ELT-ESE-3 DSBL

Digital Signal processing practical work

HAN Electrical Engineering/Embedded Systems

E.J Boks MSEE MA

Assignment 4: Desktop implementation of a configurable fixed-point FIR-type bandpass filter

Target:

Write an application that makes it possible to design a digital filter via the graphical interface and test the filter with a signal generator present in the application.

Time:

4 weeks.

Required material:

Workstation in B1.29 / B1.33, or your own computer.

Theory book.

WxWidgets online manual at http://www.wxwidgets.org

Written and tested classes from Assignment 1.

Description:

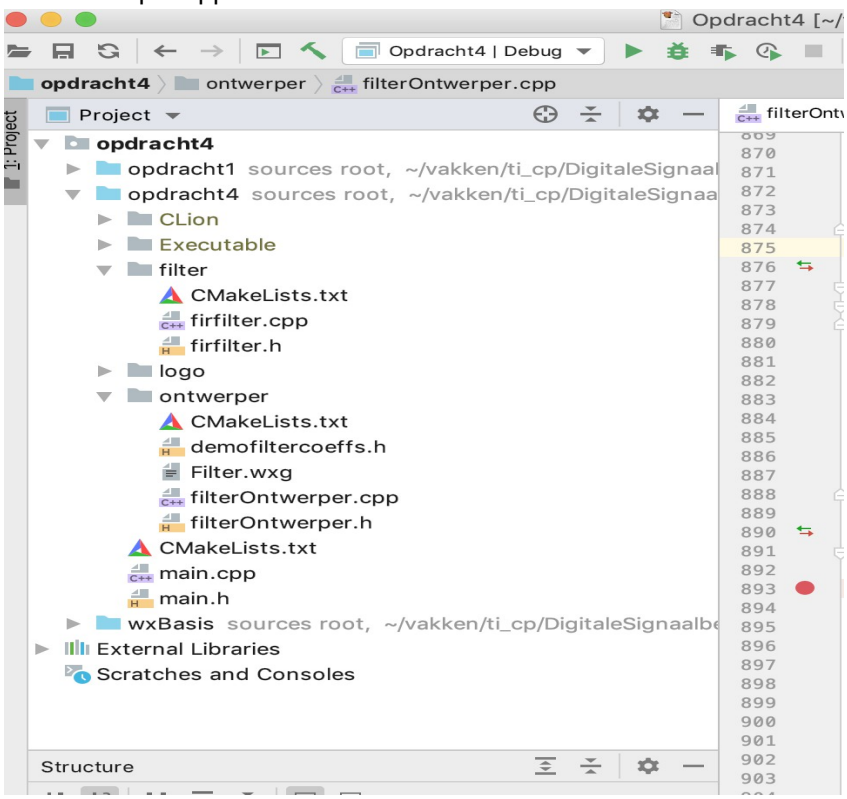
Complete an application that makes it easy for the user to set an FIR filter so that a certain signal band is stopped / transmitted. The application must therefore meet the following requirements:

- The program must be written in the C++ language.
- The wxWidgets toolkit is used for the graphical interface. The example code provided as a start provides a working but empty application.
- The user indicates the following parameters in the GUI:
  - the sampling frequency.
  - The start and stop frequencies of the pass band.
  - The order of the filter.
  - A gain factor.
  - The applied window.
  - The number of bits used for Fixed-point coding.
- The application is intented to generate FIR coefficients, which will enforce (filter) the frequency behaviour specified by the user in an FIR filter.
  - The FIR coefficients must first be calculated in floating point format.
  - The final filter FIR coefficients of having to be in fixed point (Qx format). The conversion is done on the basis of the GUI settings for the number of bits to be used (wxSpinCtrl element).
  - To cut the coefficients, the Rectangle, Triangle or Hamming window must be used. The window selection follows from the GUI (wxChoice element). The selected window takes the filter order for the number of coefficients.
  - The stated gain factor must be entered at the coefficients.
  - The filter coefficients must be stored in the wxArrayShort filterCoeffs, present in the filterVenster class.
  - The FilterFirInt16 class must be expanded to function as an FIR filter. The class constructor expects the number of taps that the filter has, a pointer to the array where the coefficients are and a scale factor with which the MAC result is compressed for it

to fit into the Int16 format.

- The user must be able to analyze the fabricated filter. Use the upper two windows and draw:
  - In the left upper window a plot of the impulse response of the designed window in the time domain.
  - In the upper right hand pane, a plot of the designed window in the frequency domain.
- The user must be able to test the fabricated filter:
  - The application has a cosine signal generator which can produce a test signal with a frequency between 0 and 4 times the sampling frequency. In addition, the amplitude can be set over the full 16 bit range.
  - The selected test frequency is presented to the filter as an input signal. The filter must be able to filter with the coefficients generated by the application.
  - In addition to the cosine signal, you can also generate:
    - A square wave with the same amplitude and frequency characteristics as the cosine.
    - an impulse
    - a step

There are four files in the supplied code: firfilter.cpp and firfilter.h, as well as filterOntwerper.cpp and filterOntwerwer.h:



firfilter.cpp must be filled in with the implementation of the FIR filter. Make sure that the filter works in fixed point and has loop unrolling (factor 4).

filterDesigner.cpp needs to be completed to enable the coefficient generation.

- The user must also be able to store the generated FIR coefficients in a C Header file. This is already prepared in the example code and does not need to be worked out by the student.

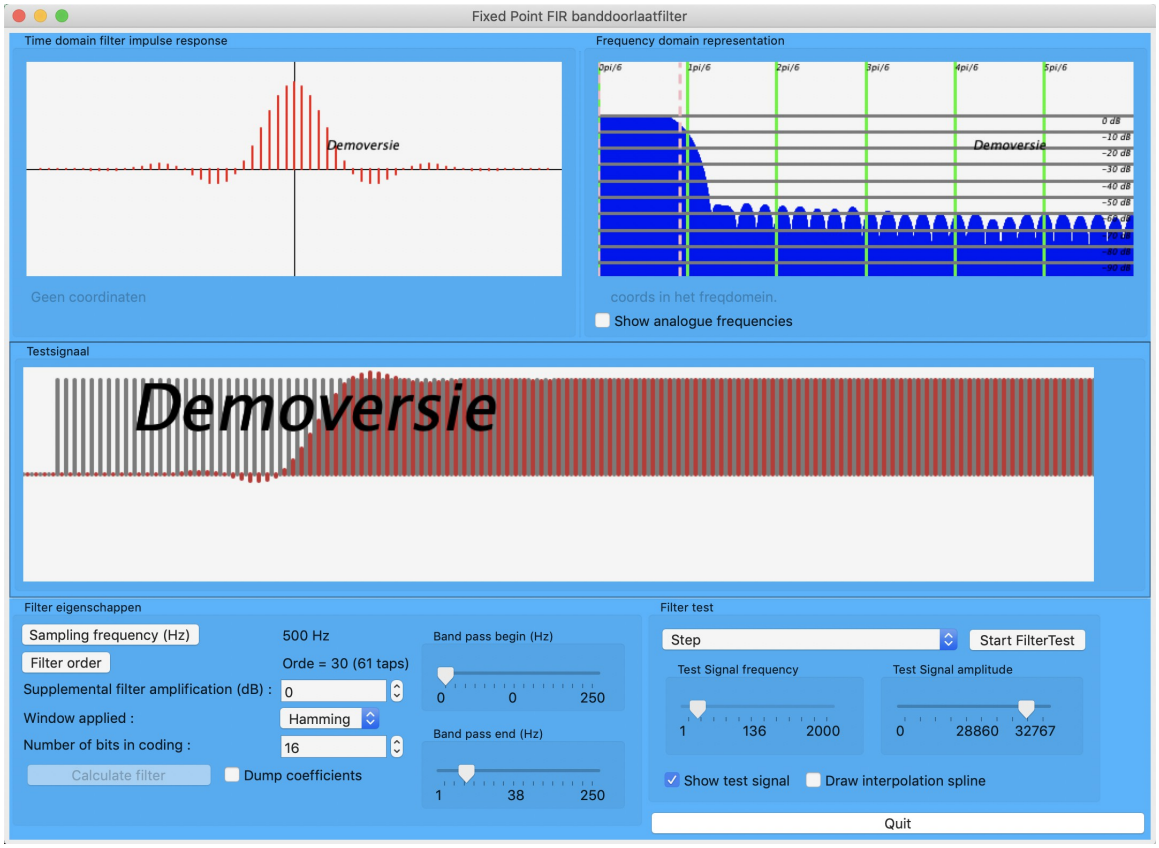
Assignment:

You have to extend the sample code already supplied to a whole that works according to the description given above.

The parts that you must complete are these:

1. An interpretation of the FilterFirInt16 class. The filter is based on the CircularBuffer class from assignment 1. Two circular buffers are used, one for the coefficients and one for the data.
2. The *triangle(const Int32)*, *hamming(const Int32)* and *sinc(const double)* functions in filterOntwerper.cpp
3. Work out the functions *computeFixedPoint()* and *computeFloatingPoint()* that perform the conversion between floating point h [n] and fixed-point. These functions depend on the entered amount of coding bits. Note that one of the number of bits specified in the GUI is always used for the sign of the value.
4. Work out the function *berekenFilterHandler (wxCommandEvent &)* that calculates and draws the filter coefficients, taking into account the chosen window (triangle or Hamming).
5. Calculate the function *berekenFreqResponse ()* calculates the frequency image and determines minima and maxima (in dB) of the response over 0 to π.
6. The function *tekenFreqSpectrum ()* which draws the frequency image of the impulse response.
7. [optional] In `void FilterVenster::tijdViewMuisBewegingHandler(wxMouseEvent &event)` en `void FilterVenster::freqViewMuisBewegingHandler(wxMouseEvent &event)` information is determined about the graphical coordinate, which says something about h [n] or H (Ω), for example the filtering at a certain frequency or the value of an impulse response sample. View the demo version of the assignment or ask the teacher for more information. Elaboration of this option yields a bonus point for this assignment!

With a properly working application an image is obtained that looks like the image below:



Delivery:

Show the teacher your working application.

Write a small report, containing the following:

- source code application
- screendump of the working application

Delivery takes place together with the other assignments. Do not proceed without verbal approval of the work with the teacher.

Tips

First view *firfilter.h* and *filterOntwerper.h*. Here you will find all the variables that you have to use. See also where the wxWidgets event handlers are and how they store the information offered by the user.

The *filterOntwerper.cpp* file contains the GUI section for a large part already prepared. You only have to fill in the signal processing parts. Let the wxGlade comment with peace (do not erase). The graphic tool wxGlade requires these, in case you want to edit the GUI layout.