RTR532 2018
Problem set #5

Total value - 100 points

Submission deadline – 2018-05-11 23:59

Aim of this problem set – to practice IIR filter development in FPGA

1. Design entry – biquad iir filter
2. Design verification with testbench

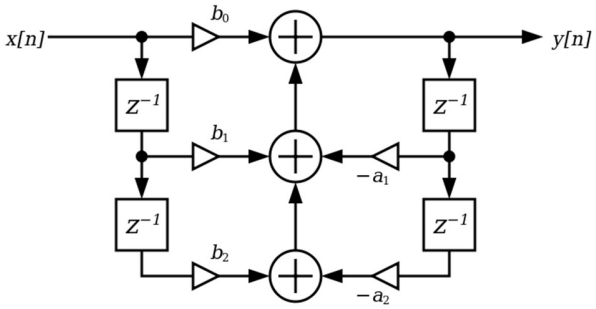
Starting notes

1. Copy the repo \rtr532-2018\problem_sets\ps05\ contents to your repository (\repo-student\problem_sets\ps05\)
2. Remember to commit changes (of .vhd files) to your local repository, before pushing it to the master remote repository.
3. **Problem sets will be graded from your repositories.**

Task 1 – Improve df1_iir_biquad.vhd (40 / 100 points)

High-order IIR filters can be highly sensitive to quantization of their coefficients, and can easily become unstable. This is much less of a problem with first and second-order filters; therefore, higher-order filters are typically implemented as serially-cascaded biquad sections (and a first-order filter if necessary).

- *https://en.wikipedia.org/wiki/Digital_biquad_filter*



Consider the biquad structure IIR filter as shown in figure. VHDL file df1_iir_biquad.vhd implements the filter. In lecture 05 the module df1_iir_biquad.vhd was functionally completed, but it could be improved.

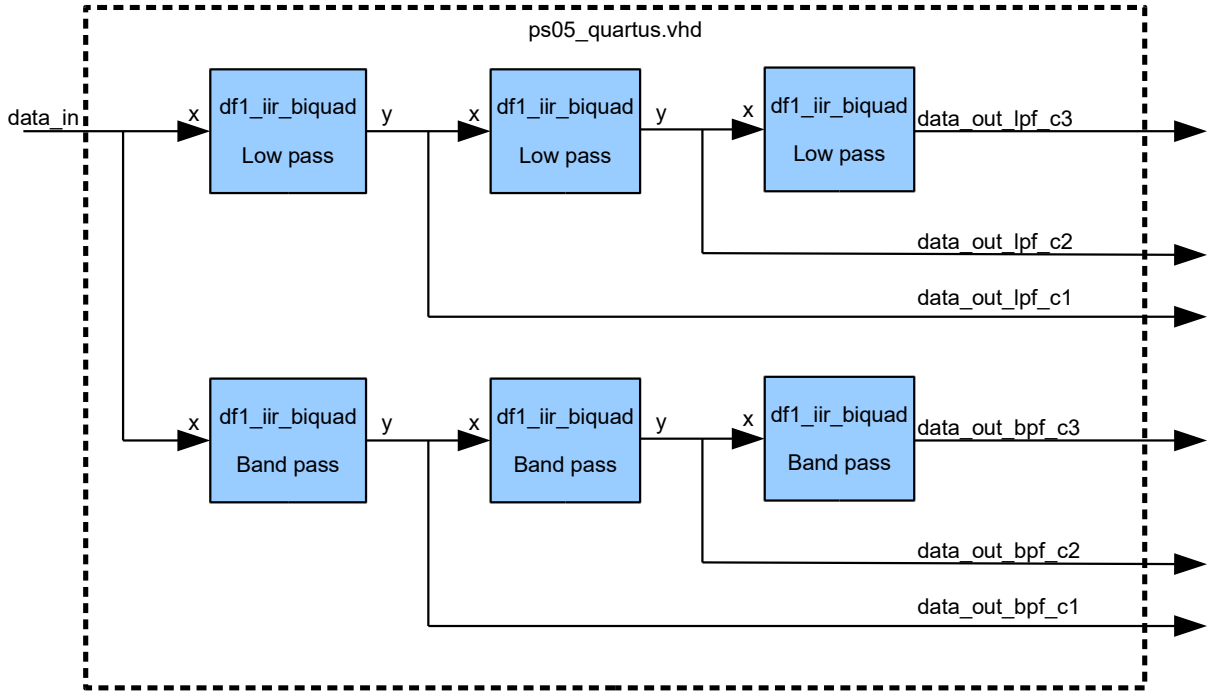
Make the following improvements:

1. Module has ***rst*** input of type std_logic. Make changes to the module to ensure that when ***rst = '1'*** all internal signals and output is low level or zero. Hint - don't forget to reset signals not only in process.
2. Filter coefficients (b0, b1, b2, a1, a2) currently are defined as constants in the filter module. Make changes to the module so that the coefficients can be passed to the module as a signal through PORT MAP (not Generic map), type std_logic_vector(31 downto 0).
3. Create a new input of the module called "sample_coefs" of type std_logic. Make new internal signals (registers) to store the correct coefficient values. The signal values should be updated on the rising edge of "sample_coefs". Use the internally stored coefficients in filter multiplications (this is a common practice, to ensure that coefficient values are correctly updated).
4. Create a new input of the module called "enable_filter" of type std_logic. Filter should not be running when "enable_filter" is '0'.

No changes should be made to the filter part (bit widths, truncations, math, multiplications) as it was proven to work during the lecture.

Task 2 – Implementing the cascaded filter structure (20 / 100 points)

Use the improved df1_iir_biquad.vhd from Task1 module to make the following filter structure:



Coefficients for modules:

Low pass filter coefficients are 2nd order Butterworth fc = 12000Hz, Fs = 48000Hz, PBR = .08 dB, SBR = .03 dB. Use the coefficient form “bin 2.30” directly in VHDL signal value assignment.

Coefficient	Value dec	Value bin 2.30
B0	~ +0.292893219	00_01_0010_1011_1110_1100_0011_0011_0011
B1	~ +0.585786438	00_10_0101_0111_1101_1000_0110_0110_0110
B2	~ +0.292893219	00_01_0010_1011_1110_1100_0011_0011_0011
A1	~ 0.0	00_00_0000_0000_0000_0000_0000_0000_0000
A2	~ +0.171572875	00_00_0000_0000_0000_0101_0111_1101_1000

Band pass filter coefficients are 2nd order Butterworth filter with frequencies: f0 = 2000Hz, fl = 1500Hz, fu = 2500 Fs = 48000Hz, PBR = .08 dB, SBR = .03 dB. Use the coefficient form “bin 2.30” directly in VHDL signal value assignment.

Coefficient	Value dec	Value bin 2.30
B0	~ +0.061511769	00_00_0011_1110_1111_1100_1111_0000_1111
B1	~ 0.0	00_00_0000_0000_0000_0000_0000_0000_0000
B2	~ -0.061511769	11_11_1100_0001_0000_0011_0000_1111_0001
A1	~ -1.816910185	10_00_1011_1011_0111_1011_1110_0101_0111
A2	~ +0.876976463	00_11_1000_0010_0000_0110_0001_1110_0010

Task 3 – Feeding the test signal and observing the result (40 / 100 points)

Third task is to verify the filter functionality. Create a testbench that generates the test signal and observe the outputs. Each next cascade filtered result should be better than previous. As result for this task I expect to be able to open Quartus project, click Tools – Run Simulation Tool – RTL Simulation and it launches for a time length where operation of the design can be observed.

Generating the stimuli

Clk - Generate sample frequency clock of 48 kHz

Rst – generate both high and low reset to ensure correct start up

Data_in – sum of 1.5kHz sinusoid, 3kHz sinusoid, 4.5kHz sinusoid, 6 kHz sinusoid, 15 kHz sinusoid, 20kHz sinusoid (hint – use previous lecture DDS principle for generating the sinusoids). Otherwise, instead of sum of sinusoids, you can generate a pulse train signal (on/off) with frequency that will show if the filters work correctly (especially the bandpass part).

Note how signal outputs after 1, 2, or 3 filters change and become better filtered.