# ENGR-435
# Lab 5 - FIR filter

## Objective

The objective of this is to design, implement, and test a finite impulse response (FIR) filter.

## References

VHDL example code in the textbook and prior labs..
Xilinx documentation, particularly the example coding in the XST synthesis manual. (see class webpage)

## Design Flow for this week

The general flow for this lab is:
- Create a system block diagram
- Create state diagram(s) as needed.
- Create a timing diagram (to help you think through the sequence of signals needed)
- Create a VHDL description for the FIR algorithm and control logic
- Synthesize, place, route, and create the bit map file using Xilinx software
- Download design to the FPGA board
- Test
- Iterate as required

## Functional Description

An analog input signal will be sampled with a MCP3202 A/D converter. The sampled value is a binary number that will be the input to a Finite Impulse Response (FIR) filter. Filter function, i.e. the type of filter, is defined by the coefficients installed in the filter. It could be low-pass, high-pass, etc. For this lab we will focus on low-pass. Filter output is a binary number that will be sent to the DAC to reconstruct an analog waveform.

## Performance Specifications

12 bits, unsigned, is created by the MCP3202. Input frequencies can range from 50Hz to 20000Hz. The output of the filter will be converted back to analog using the MCP4822 DAC. The analog input voltage range is zero to +/- Vdd/2 or about +/- 1.65 volts. Analog output is +/- about 1.024v. A suggested sample rate for the A/D and D/A is 50Khz (a nice number and very adequate for audio).

## FIR details

The FIR can be defined as:

$$R_i = \sum_{k=1}^{n} D_k \times C_k$$

where $R_i$ = an output data value, D is an input data value, and C is a filter coefficient
D and C are 12 bit binary numbers.
$R_i$ can thus have a magnitude greater than 12 bits
n is the number of “taps” on the FIR filter and is equal to the number of saved data points. Data point one could be considered the most recent data point and data point n the oldest. 32 “taps”or more are needed for a good filter.

In words, get a new data sample from the A/D and place it in memory while throwing away the oldest data point. Multiply each saved data point (n of them) by a corresponding coefficient (n of them) and sum up the results. The result is the filter output.

Typical FIR filter design assumes we are working with signed numbers while the MCP3202 A/D creates an unsigned binary number ranging from 0 (for zero volts in) to 4095 (for voltage in equal to Vdd). Assuming the analog signal starts out bipolar (for example a sine wave going + and - centered on zero volts) add a DC voltage with magnitude Vdd/2 to create a level shifted (offset) analog signal. When the signal has zero amplitude the input to the A/D will be Vdd/2 and that voltage will represent a data magnitude of zero (if testing with DC coupling to the A/D, use signals from the signal generator which allows easy addition of the offset voltage).

To convert the incoming binary number to a signed number subtract 2048 from each value (two’s complement format is desirable) Use signed arithmetic when implementing the FIR algorithm.

## Additional details

Note that the size of R after n multiplications and additions can have a magnitude greater than 12 bits. You need to account for that in your design. A decision will have to be made regarding which bits of R will be used as the filter output.

Coefficients can be placed in “ROM” memory via your VHDL description file. For initial testing use simple coefficients, such as a value of one, to determine if multiplication and addition are working. Creating “real” coefficients (i.e. ones that determine an actual filter function such as low-pass) is not required as your part of the lab per se. Coefficients will be supplied or possibly created by the class.

## Expectations and a request

I expect that the design you do will be based on the problem statement, your careful consideration of it, and your creative formulation of a solution. I ask that you don’t google a solution. Create your VHDL description based on your block diagram and the problem definition. See me for help as needed.

## <u>To</u> <u>Turn</u> <u>In</u>

Submit a short report containing the following:

A well crafted, grammar correct, abstract. (Typically this is one paragraph)
Block diagram(s), state diagram(s), timing diagrams, other design documentation, and notes.
Comments regarding problems encountered, things learned, anything you wish others to know.
Results. What worked (everything we hope, but if not, clearly state your results).
Screen shot of the Device Summary matrix in ISE.
  All warning messages must be explained and justified.
VHDL file(s) and your .bit file are to be copied to D2L.

Notes:
There are two analog inputs:

AC coupled analog input. A capacitor connects the input to the opamp This is a good way to connect a signal generator that goes plus and minus from zero. Maximum signal is +1.65 and -1.65 volts with the generator going plus and minus

DC coupled analog input. You can connect a DC voltage that ranges from zero to 3.3 v. This connection is through a 1000 ohm resistor to protect the opamp. But it means that the A/D will not produce a zero value output, it will always be a few counts above due to voltage drop across the 1000 ohms.