Abstract:

This lab report covers the design of a 32-tap FIR filter using VHDL. An MCP-3202 A/D converts the analog signal to digital at a rate of 49.60 kHz. In between samples, the data is processed, and a new output is prepared. The Microchip MCP-4822 D/A converter runs in parallel with the ADC and converts the serial signal to an analog voltage. The 32 calculations for the FIR filter were designed to be completed sequentially, rather than in parallel, to reduce the hardware demand. The design was implemented on a Xilinx Spartan 6 FPGA and tested in the lab to verify correct operation. As of June 6th 2021, the circuit is working as intended.

Relevant Theory:

A FIR filter can be used to implement almost any sort of frequency response digitally. An FIR filter is usually implemented by using a series multipliers and adders to create the filter's output. The output of a FIR filter depends on previous input and is characterized using the following expression:

$$R_i = \sum_{k=1}^{N} D_k \cdot C_k$$

where R_i is an output data value, D_k is an input data value, and C_k is a filter coefficient. The number of taps, N, represents the number of previous data points to be used in the calculation of the filter output. The output can be visualized using the diagram shown in figure 1. In order to compute an output R_i, the previous N values of the input x are multiplied by N coefficents. The previous N inputs of x are stored in a data array.

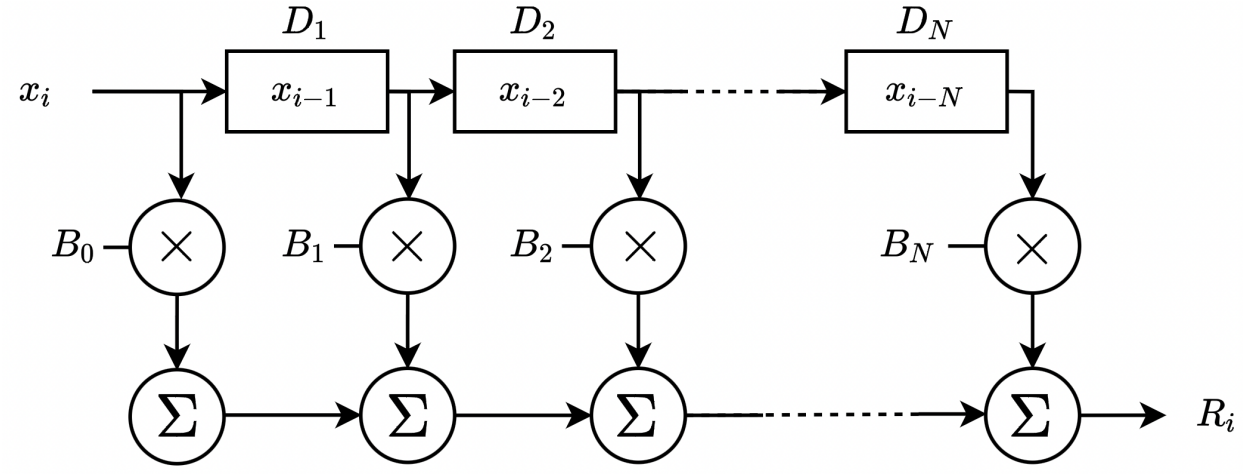


Figure 1: A discrete-time FIR filter of order N

Design:

Since we are working with a 12-bit A/D, both D and C will be 12-bit values. The size of R after n multiplications and additions will most likely have a magnitude greater than 12 bits. The maximum size that R could be is $12 \times 2 + 5 = 29$ bits. Since we are also using a 12-bit D/A, twelve bits from the calculated output value R are selected. Since it's hard to know which bits will produce the desired results, we used two toggle switches to select one of four different subsets of the 29 total bits. Since we are working with signed numbers, the MSB (the sign bit) is maintained with each range selected.

Typical FIR filter design assumes we are working with signed numbers while the MCP-3202 A/D creates an unsigned binary number ranging from 0 (for zero volts in) to 4095 (for voltage in equal to V_{DD}). Thus, the analog data is level-shifted up $V_{DD}/2$ and then 2048 is subtracted from each incoming binary number from the A/D.

After a new filter output is computed, the result is then level shifted back down by adding 2047 to the signed outcome of the FIR filter. The data can then be run through the MCP-4822 D/A to create the filtered analog output. Both the A/D and D/A are run off a 892.857 kHz clock. Since both the A/D and D/A require 18 clock cycles to sample/transfer a 12-bit data value, the sampling frequency is 892.857/18 = 49.6 kHz. Thus, every 20.16 µs a new FIR output value is computed. The calculations are run off a 3571.43 kHz clock, which is four times the speed of the serial clock running the A/D and D/A. This allows the calculations to be completed synchronously with the A/D conversions, while allowing enough time for all 32 multiplications and additions to finish.

The coefficients are stored in a 128x12 bit ROM memory with asynchronous read. There are four sets of coefficients stored in memory that have been used in testing. Two switches (sw0 and sw1) select the set of coefficients that are used. The coefficients are listed below:

1. Coefficients used to debug the FIR filter. The first coefficient is one ($C_0 = 1$) and all the rest of the coefficients are zero. When using these coefficients, the signal is essentially passed through the filter unchanged. The only noticeable difference is that the signal is attenuated slightly. This is because the maximum output voltage of the DAC (2.048 volts) is less than the maximum input voltage of the ADC (3.3 volts). Thus, the transfer characteristic due to this mismatch in maximum voltage is: $V_{out} = 0.621\, V_{in}$.
2. Also coefficients used to debug the FIR filter. All the coefficients are 1.
3. Coefficients for a FIR filter with a 300 Hz cutoff frequency and a sampling rate of 50 kHz.
4. Coefficients for a FIR filter with a 5kHz cutoff frequency and a sampling rate of 50 kHz.

The data values are stored in a 32x12 bit RAM with a circular buffer. After the ADC has recorded a sample, the data value is stored in the RAM. The data is stored at the address specified by the value of the counter *addr_offset* which is incremented each time the data is

sampled. The value specified at *addr_offset* represents the most recent value loaded into the data RAM. The value at *addr_offset* + 1 is the oldest value.

Since the index of the newest data point is constantly changing, the multiplication between data and coefficients requires two counters. Thus, when computing the output of the FIR filter, the signal *addr_count* is used to sequentially step through the calculations while the counter *addr_offset* keeps track of the newest data point. The coefficient C_0 needs to be multiplied by the oldest piece of data and the coefficient C_{31}. Thus, the data value at *addr_count* + *addr_offset* needs to be multiplied by the coefficient at *addr_count.*

The input/output resources used in the design are shown below:

- **sw7** and **sw6** select which set of bits are to be sent to the DAC.
- **sw2** selects which channel the DAC will use for output.
- **sw0** and **sw1** select which coefficients are to be used in calculations.
- **pmod1** was used for the ADC and DAC inputs/outputs. The
  - **<1>:** *cs* – ADC chip select
  - **<2>:** nc – no connect
  - **<3>:** *dout* – ADC serial data out
  - **<4>:** sclk – ADC serial clock
  - **<5>:** *sdata* – DAC serial data
  - **<6>:** *cs* – DAC chip select
  - **<7>:** *load* – DAC load command
  - **<8>:** *sclk* – DAC serial clock

Other input/output resources used for debugging:

- **tek1** and **tek2**: Used for extracting signals from the FPGA to view on the Digilent scope.

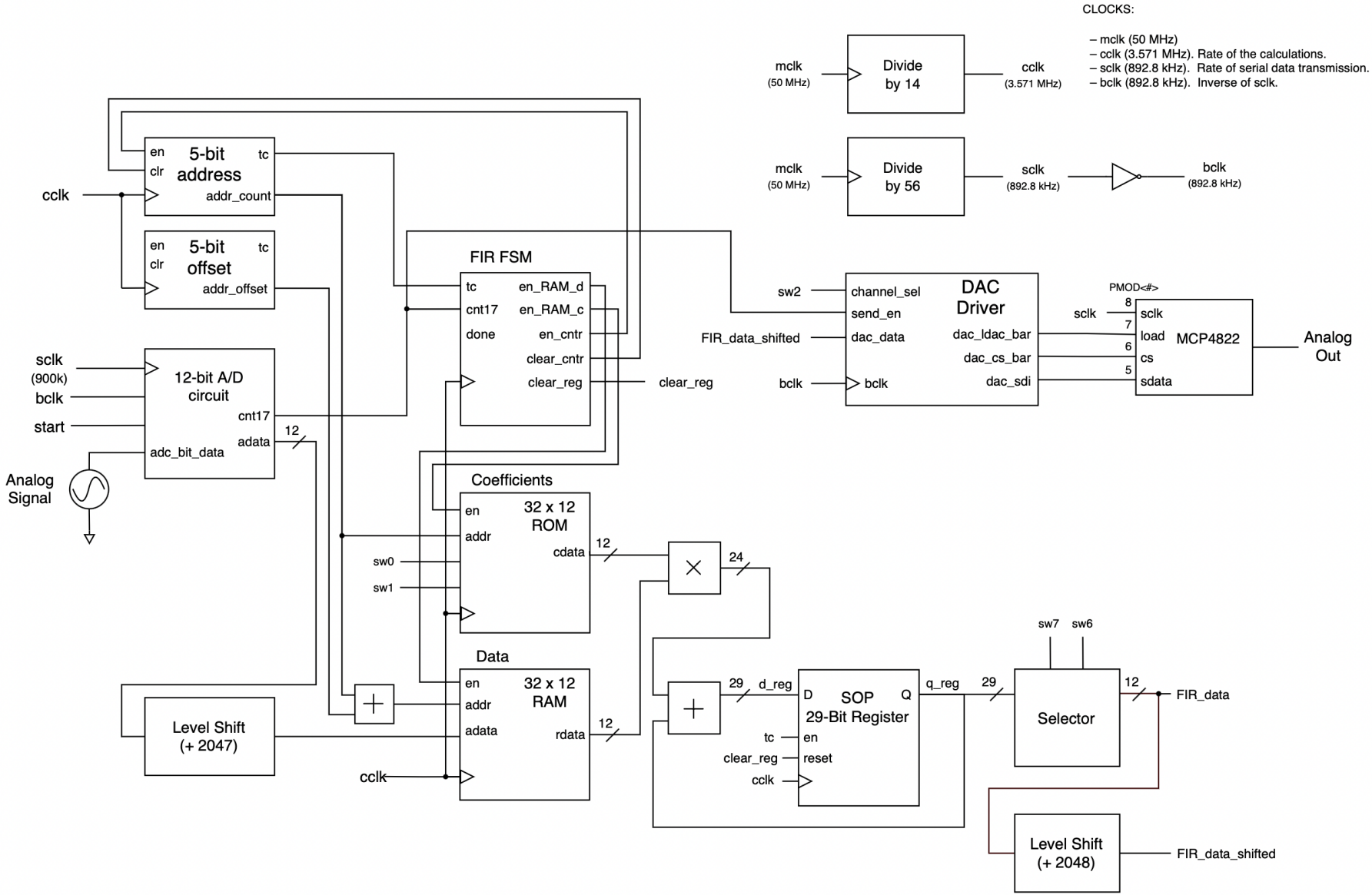
The block diagram, FSM, and timing diagram are shown on the following page. The design utilizes two modules design by Dr. Larry Aamodt; a ADC driver and a DAC driver. The design files and VHDL descriptions for these drivers can be found on the course website at: https://gab.wallawalla.edu/~larry.aamodt/engr435/index.html

Slight modifications to the DAC driver were made to meet project objectives. These are as follows:

- **Complete this**: .

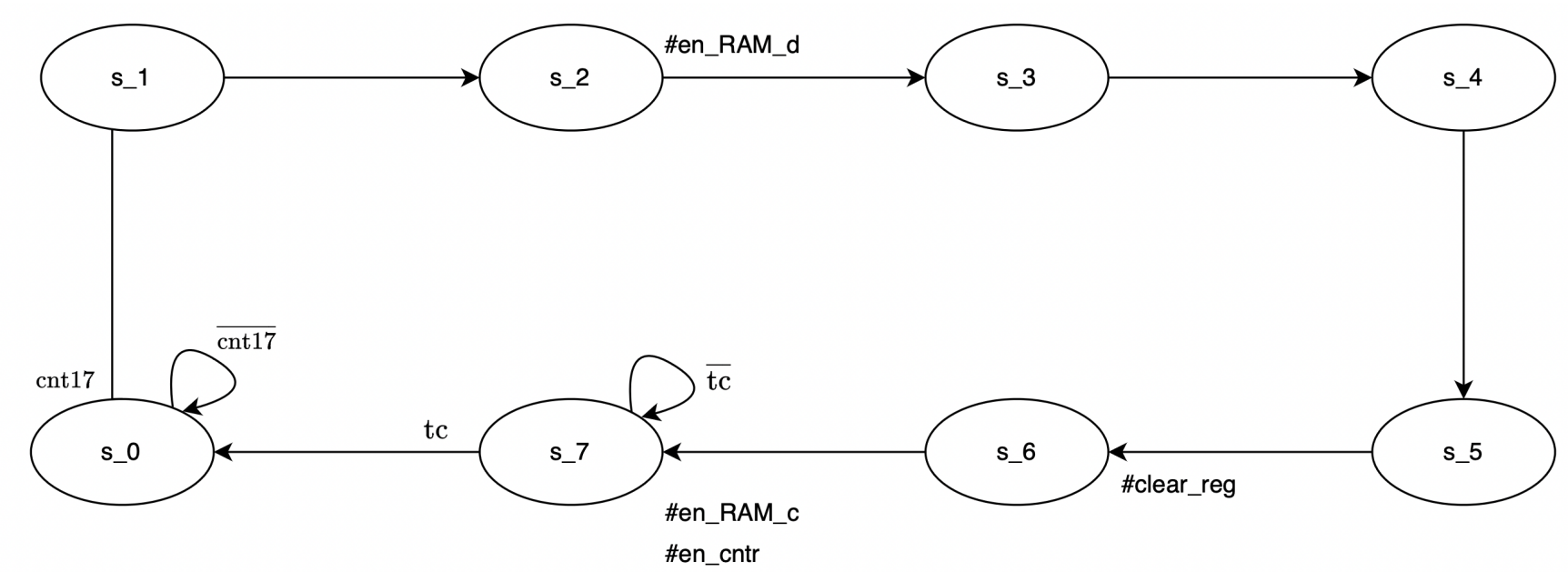
- Block Diagram:

Below is the block diagram for the FIR filter. Block diagrams for the ADC driver and DAC driver were provided by Dr. Aamodt and can be seen on the class website: https://gab.wallawalla.edu/~larry.aamodt/engr435/index.html



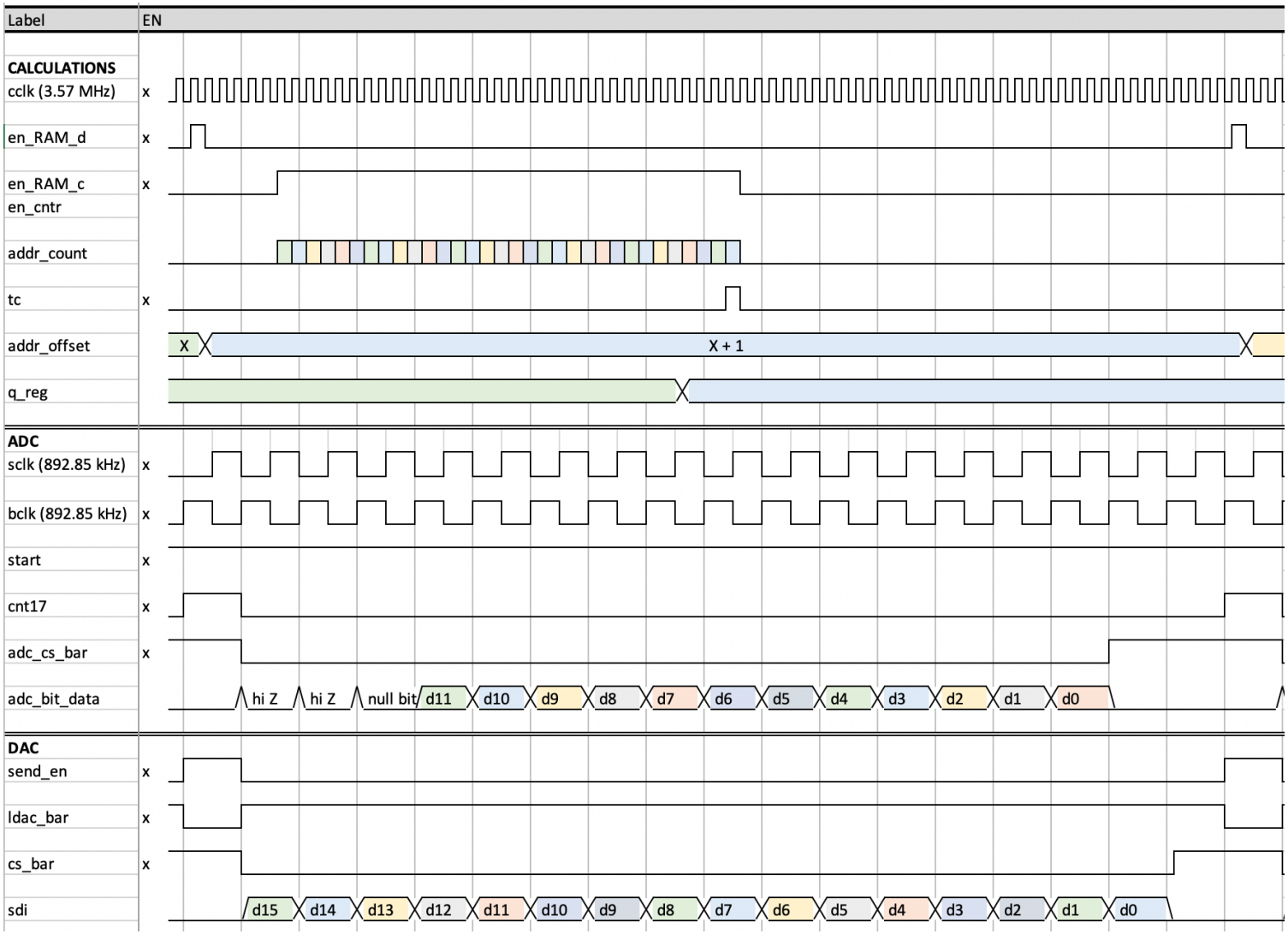
- Finite State Machine:

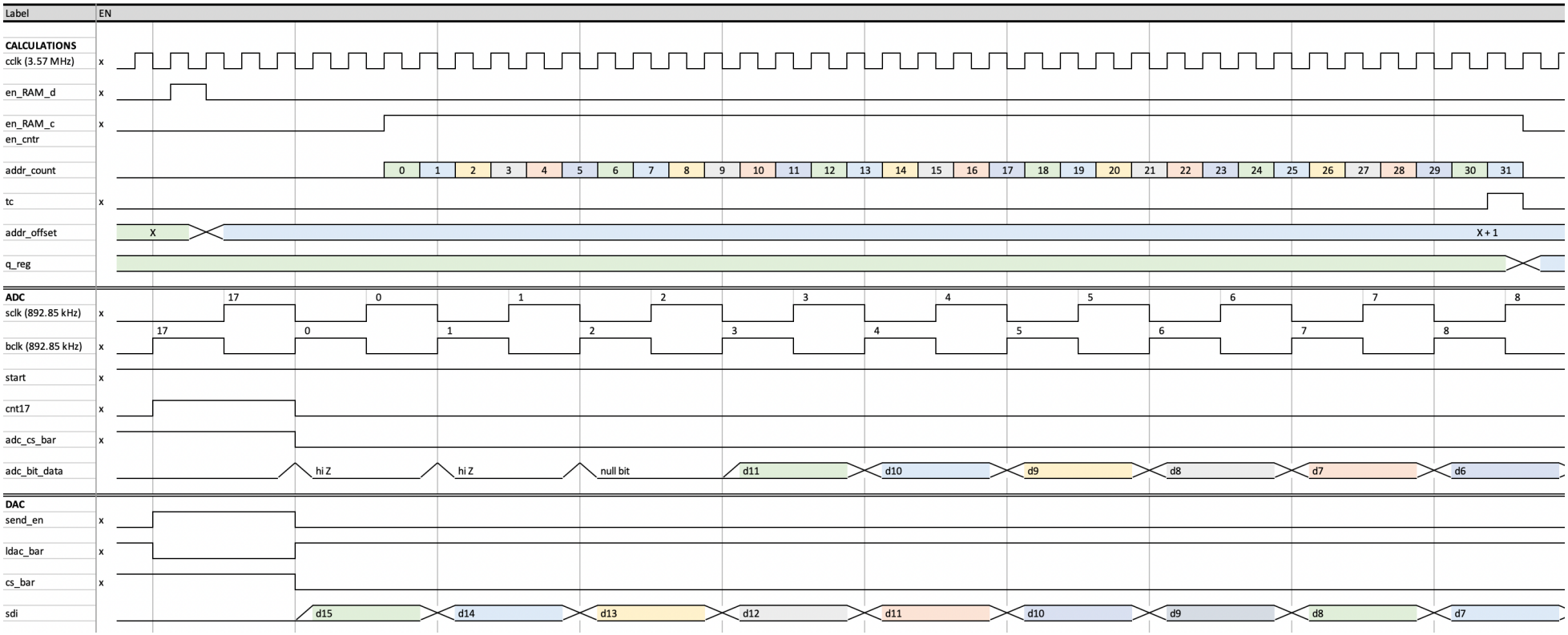
Below is the Finite State Machine diagram for the FIR Filter FSM. Both the ADC and DAC controller also have FSMs embedded in their designs. The FSMs of those devices can be seen on the course webpage:
https://gab.wallawalla.edu/~larry.aamodt/engr435/index.html



- Timing Diagrams:

Below are a couple of timing diagrams for the FIR filter. The complete timing diagram can be seen in the interactive Excel worksheet: "Froelich_FIR_Timing.xlsm". The Excel spreadsheet with the timing diagram has been submitted to the D2L website.




Label
EN
CALCULATIONS
cclk (3.57 MHz)
en_RAM_d
en_RAM_c
en_cntr
addr_count
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
tc
addr_offset
X
X + 1
q_reg
ADC
sclk (892.85 kHz)
17 0 1 2 3 4 5 6 7 8
bclk (892.85 kHz)
17 0 1 2 3 4 5 6 7 8
start
cnt17
adc_cs_bar
adc_bit_data
hi Z
hi Z
null bit
d11
d10
d9
d8
d7
d6
DAC
send_en
ldac_bar
cs_bar
sdi
d15
d14
d13
d12
d11
d10
d9
d8
d7

Testing and Issues Faced:

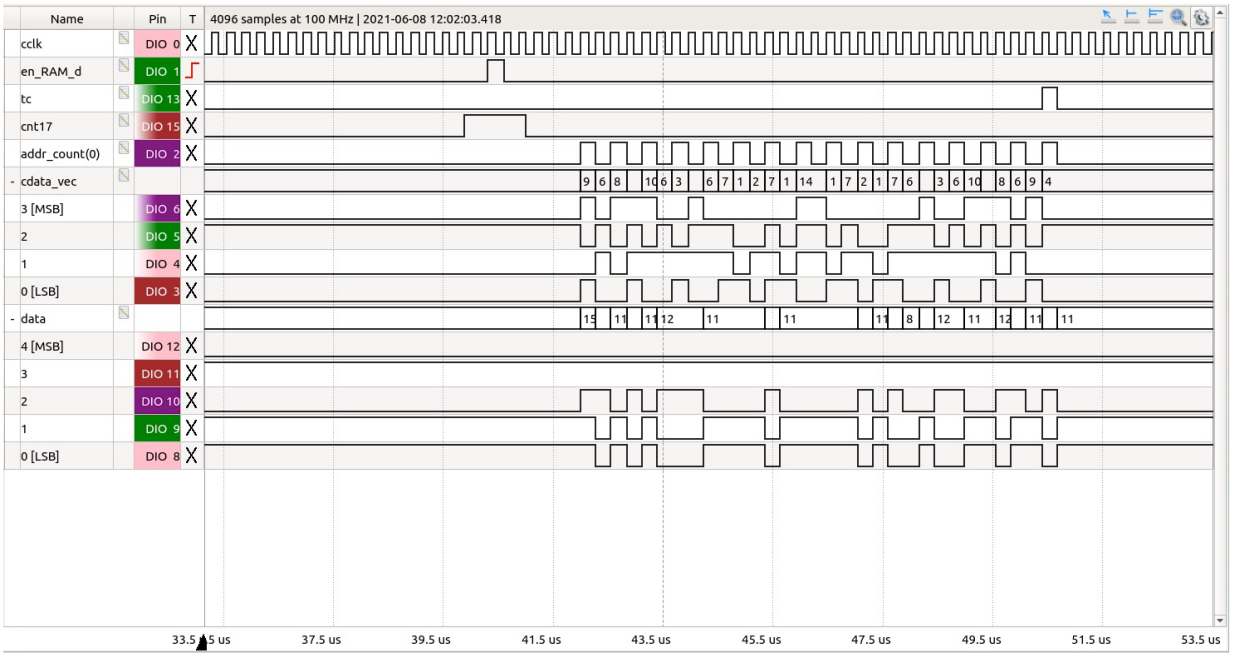
The device was tested utilizing the Digilent analog discovery to view control signals and verify correct operation. Initially, the ADC and DAC were disconnected from the circuit and the ADC was replaced with a RAM. This test allowed us to verify that the filter calculations were indeed functioning as intended. In performing this test, I did notice a couple of issues that inhibited correct operation:

1. The coefficient ROM was read synchronously, while the data RAM was read asynchronously. This led to an offset between the data value and the correct coefficient of 1 clock cycle. This issue was resolved by modifying the coefficient ROM to read asynchronously.
2. The accumulator register was not reset after a calculation. Thus, the register just kept increasing until it overflowed. This issue was resolved by adding the control signal *clear_reg.* The signal *clear_reg* is generated after the calculation result has been saved in the DAC input register and before the next calculation.

Following this test, the ADC and DAC were hooked up to the circuit. Initially, I set the DAC and ADC running, assuming that they would be in sync since each took the same number of clock cycles to complete a conversion. But after looking at the signals on the scope it was evident that the ADC and DAC were not in sync as I had anticipated. Thus, I connected the *cnt17* output from the ADC to the *start_dac* signal of the DAC driver so that the completion of an ADC conversion would trigger the DAC. This gave predictability to the timing of the ADC and DAC modules. There were a couple other minor errors that I debugged, but none significant enough to include in this report.

Results and Conclusions:

A lot of work could be put into testing the design and optimizing the filter performance, however, for the context of this lab we only needed to verify that the filter did seem to work as intended. The current design is operational and resembles a filter, however, the response of the filters tested do not perfectly match my expectations in terms of cutoff frequency and roll-off. That being said, the data that I recorded was extremely similar to that of other colleagues, leading me to believe that the implementation is indeed correct. On the next page are some images and short descriptions that discuss some of the results seen.



The figure above shows some of the key control signals and data values used when calculating the FIR filter output. The *cdata_vec* bus shows the last 4-bits of the coefficient in decimal and the *data* bus contains values read in from the ADC.

When testing the integration of the ADC, the DAC, and the circuitry for computation, the first step was to utilize the first set of coefficients (a one followed by 31 zeros) to pass the waveform through the ADC and DAC. If the location of the one is shifted within the array, the output waveform will be phase-shifted in comparison to the input signal. This phase shift can be seen below:

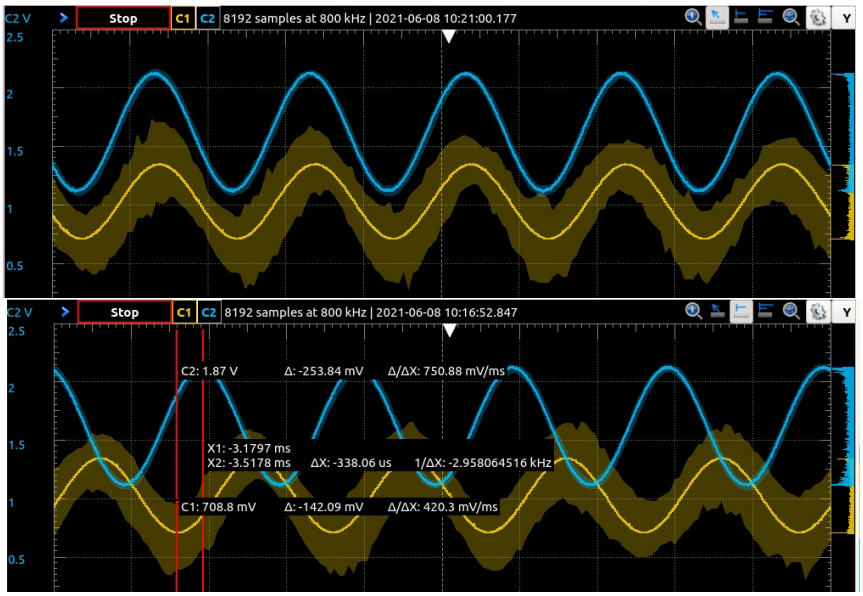
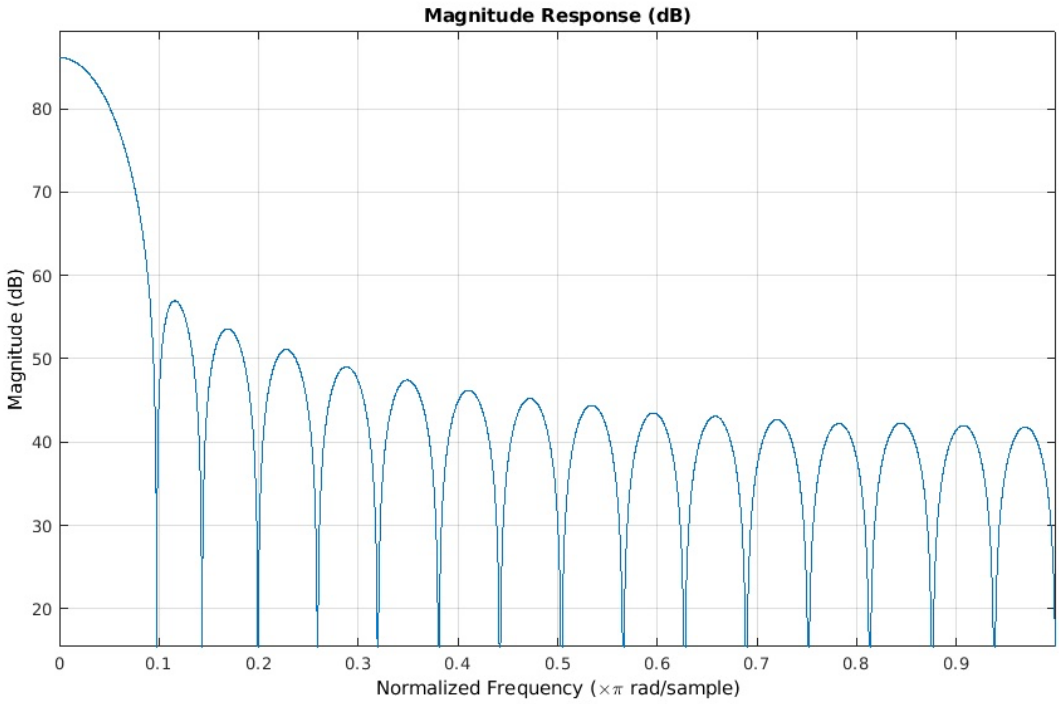


Figure 2: Input (Blue), Output (Yellow)

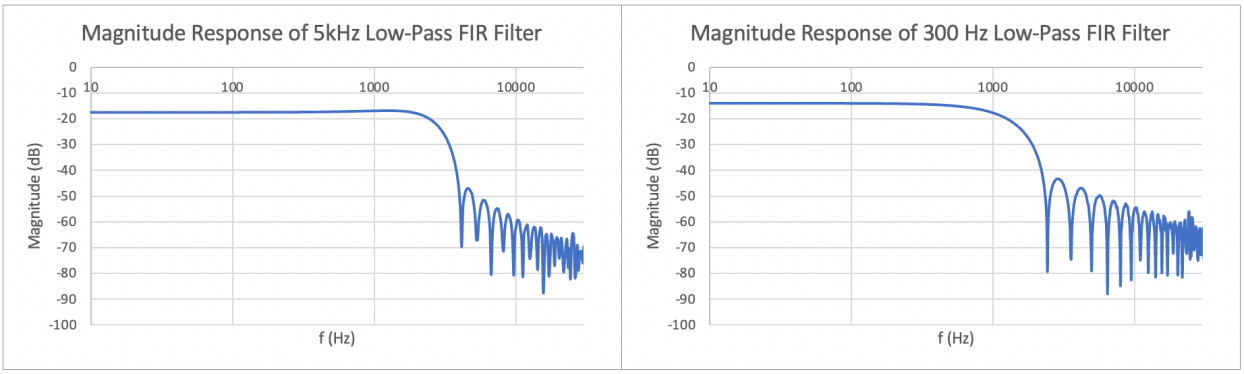
When testing the filter performance, we used two sets of coefficients:

1. The 300 Hz low-pass filter designed for a switching frequency of 50 kHz.
2. The 5kHz low-pass filter designed for a switching frequency of 50 kHz.

To determine the theoretical filter performance the MATLAB was used to plot the magnitude of the FIR filter based on the coefficients. For the 300 Hz lowpass filter with a switching frequency of 50 kHz, the result is as shown.



As seen the filter is almost like a type of comb filter. Using the Digilent, a frequency sweep from 10 Hz to 100 kHz was generated. The data was saved to a csv and then plotted in Excel. Below are the magnitude results for the filter with the 300 Hz cutoff frequency.



The magnitude response of the 5 kHz filter is much closer to what we are expecting. The 3 dB down point on that FIR filter is around 3 kHz. This is slightly less than the expected value of 5 kHz. The 3 dB down point on the 300 Hz filter is around 900 Hz, three times higher than expected.

In addition to plotting the magnitude response, I also tested the circuit the "old-fashioned way" by passing several sin waves through the filter and viewing the output magnitude on the scope for different frequencies. For the 5 kHz filter the following graph was generated.

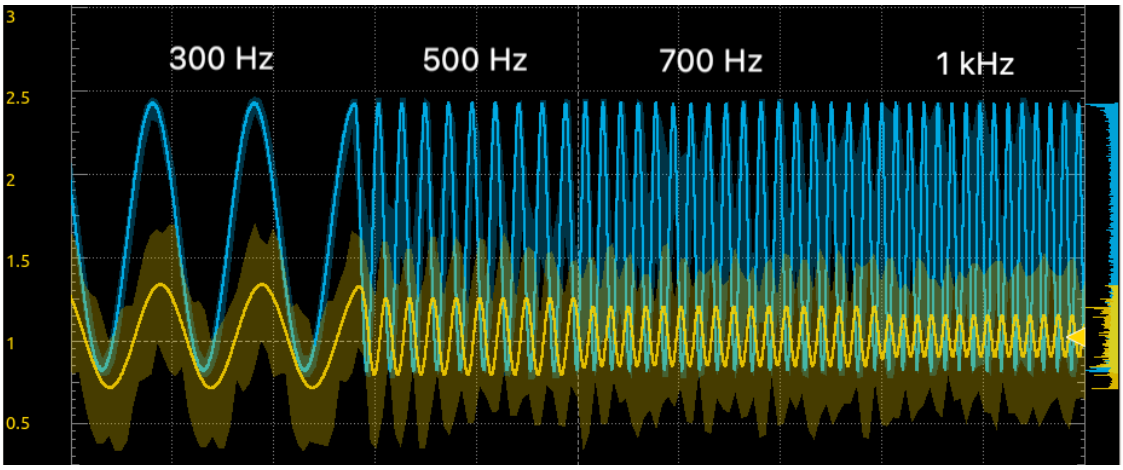


Figure 3: Input Sinewave (Blue), Output Sinewave (Yellow) for 5 kHz LPF

VHDL:

The project VHDL files, bit file, constraints file, and design summary file have all been submitted on D2L.