Capstone Design Project: Final Report
Multirate Filter Design

Introduction
The goal of this Capstone Design project is to explore a set of reliable filter designs and construction methods for specific hardware applications.

Problem
One specific difficulty in filter design is the climbing computational load of a high sampling rate system. Filters are commonly implemented in these systems with a certain length (these are finite impulse response in particular) which are multiplied for each sample value and accumulated to form an output signal. At a high sampling rate, there will be more sample values present in the input, therefore making it necessary for a higher computational load at that data rate. There are some filter applications in which this high sampling rate is excessive and can be removed to lighten the computational load on the digital system.

Solution
A common solution to this issue is the use of a Multirate Filter. A Multirate Filter will decrease the sampling rate of the input signal to a more manageable value so that the computational load is much smaller. By lowering the sampling rate, there will be less computations required by the digital filter to acquire the desired output signal.

Results
Just like there are many different applications for Multirate Filters, there are many different designs for Multirate Filters. One design may not be the optimal solution for all Multirate Filter applications, but there is certainly a design that is most efficient considering the design specifications. Below are results gathered by implementing different Multirate Filter designs for a given specification. One can assume that our results can be applied to any similar design requirement.

Design Requirement:
- Lowpass filter
- Narrow passband
- Cutoff Frequency: 80Hz
- Sample Rate: 8kHz

The group has calculated the approximate computational load for a standard Single-Rate Filter design for this application. We will use this as a reference for any explorations in Multirate Filter designs.
Using the following MATLAB command we were able to find an estimated order for the required filter:

```
remezord([70 80], [1 0], [0.01 1E-4], 8000)
```

This will result in a filter order of 2511. A single rate filter of this size in the required design specifications will give us:

2512/2 MACs/sample * 8000 samples/sec = 10048 kMACs/sec *

*(MACs is Multiply and Accumulate Computations)*

We will be able to significantly reduce this number using some Multirate Filter designs.

**Single-Stage Multirate Filter**

For the given design, we are able to Decimate our signal by 50 samples (8000 samples / 160Hz = 50. 160Hz is the required sampling rate to retrieve an 80Hz or below signal without aliasing). To be cautious in our design, our group chose a decimation factor of 48:

![Fig. 1](image_url)

Above is a simple block diagram of our filter design constructed in Simulink. Each decimation filter block consists of a Lowpass filter and a Downsampler (or in the case of the Interpolation block and Upsampler and a Lowpass filter). The lowpass filter for each block was the same and designed using the `fdatool` in MATLAB:
Fig. 2

Saving the designed filter coefficients to a variable ‘b’, we were able to use the following script to generate our multirate filter decimation object (in this case, Hm_firdecim_48*):

```
M = 48; % Decimation factor
Hm_firdecim_48 = mfilt.firdecim(M,b); fvttool(Hm_firdecim_48)
```

*The interpolator was made by altering this script to utilize the mfilt.firinterp(M,b) command

Fig. 3
Running the Simulink Design Block shown in Figure 1 with a 65Hz input sinusoidal wave gives us the output shown on the final Time Scope Display:

*The resulting wave shows a frequency of 65.000Hz on the right highlighted in red

Fig. 4

Now that we are certain our design meets the given requirements, we will summarize our findings with the calculations for the computational load of our system:

\[
\text{(2512/2 MACs/sample) } \times \text{ (8000 samples/sec) } \times \text{ (1/48 decimation factor) } \times 2 \text{ (decimator plus interpolator stage)} = 419 \text{ kMACs/sec}
\]

The resultant 419 kMACs/sec is significantly less than the 10 thousand kMACs/sec calculated for the Single Rate Filter design above. Already seeing positive results, we will dive deeper into Multirate applications that will provide a more efficient system for the design requirements.
Multi-Stage MultiRate Filter Design

Above is the block diagram for a Multi-Stage Multirate filter design consisting of two decimation blocks (12 and 4 respectively), and two interpolation blocks. For Multi-Stage decimation, the product of all factors given in each block must equal the total decimation factor. In this case, we are using 12 and 4 (12 * 4 = 48). Later on, we also tested other product combinations.

As in the previous design, we must find the filter orders required for each block.

**Decimation by 12 block:**
\[
\text{remezord}([1/12\ 0.03\ 1/12], [1\ 0],[0.5e-2\ 1e-4], 2) = 225
\]
- When decimating by 12 in the first stage, the cut off must be before 1/12 of the normalized sampling frequency (explaining the first input) to prevent aliasing.
- The pass band ripple requirement (third input), must be divided by two, since we are using a Two-Stage Multirate Filter.

**Decimation by 4 block:**
\[
\text{remezord}([70\ 80], [1\ 0],[0.5e-2\ 1e-4], 8000/12) = 225
\]

Thus the design requires 452 coefficients in total (226+226). The number of coefficients is the order plus one.

Running the Simulink design block shown in Fig. 5 with the same 65 Hz sinusoidal input, we obtain the Time Scope Display shown below:
In terms of the calculations required per second
(226/2 MACs/sample) * (8000 samples/sec) * (1/12 decimation factor)
+ (226/2 MACs/sample) * (8000/12 samples/sec) * (1/4 decimation factor) = 188kMACs/sec

By utilizing a Multi-Stage Multirate Filter and decreasing the amount of computations per stage, one can observe that the total MACs has decreased as well. The test results shown above demonstrate that using Multi-Stage Multirate Filter Designs can increase the efficiency of the system.

Changing Decimation Factors:

We were able to increase the efficiency of our Multirate Filter by changing the decimation factors used for downsampling and upsampling. We chose an 8 to 6 decimation design. Below are our calculations for each filter block:

Decimation by 8 block:
\[
\text{remezord}([1/8, -0.03, 1/8], [1 0], [0.5e-2, 1e-4], 2) = 225
\]
*We designed this block and the one below using the same process for the 12 and 4 decimation blocks in the previous design

Decimation by 6 block:
\[
\text{remezord}([70 80], [1 0], [0.5e-2, 1e-4], 8000/8) = 338
\]

Thus the design requires 565 coefficients in total (226+339). This is more than the previous design, but we will see the benefits computation wise once we calculate the filter’s computational load.
After testing our results to verify that our Multirate Filter design matched the design requirements, we found the calculations required per second needed for the filter to operate:

\[
\frac{226}{2} \text{MACs/sample} \times \left(\frac{8000 \text{ samples/sec}}{1/8 \text{ decimation factor}}\right) + \frac{339}{2} \text{MACs/sample} \times \left(\frac{1000 \text{ samples/sec}}{1/6 \text{ decimation factor}}\right) = 141k\text{MACs/sec}
\]

This result is even lower than the 12 by 4 decimation design. We can conclude that in a Two-stage Multirate Filter design, choosing the two factors for decimation to be as close to each other as possible will reap the most benefits in terms of computational load.

**Multirate Filter Design in LabVIEW**

Another factor we wanted to touch on was designing multirate filters for use on an FPGA. In particular we used a National Instruments myRIO device which features a Xilinx Zynq chip. You can target the FPGA board using a customizable LabVIEW VI.

Since we are implementing our filter onto the myRIO device, we decided to change our design requirements to support 44.1kHz audio. Applying the design to audio, we will be filtering all sounds above middle C (261.626 Hz):

**Design Requirement:**
- Lowpass filter
- Narrow passband
- Cutoff Frequency: 261.626 Hz
- Sample Rate: 44.1kHz
Our design takes an audio signal from the Audio In port on the myRIO device and sends out the filtered signal through the Audio Out port. The myRIO also communicates wirelessly to the Host machine to display a frequency diagram of the filtered signal.
**FPGA Filter Design**

We decided that a polyphase filter design would be best for implementing a multirate filter. A polyphase design allows for a distributed workload among a number of filters. Depending on the application (decimation or interpolation), we will split the stream going into or out of the filters, respectively. Below is a picture of our design:

![FPGA Filter Design Diagram](image)

**Difficulties:**

One main difficulty that seemed to be reoccurring was the actual implementation of the filters onto the myRIO board. There was definitely a learning curve for using LabVIEW and getting it to coincide with the myRIO board at first. Trying to understand the FIR compiler and how it works with COE files was a little confusing at first. It also took some time to get used to working within the LabVIEW environment and utilizing the various functions and palettes. The compile times were also a recurring problem. Initially, the compile times for each filter and the project in general were very high, some reaching over four hours. It became very inefficient to troubleshoot the project because correcting and recompiling was very time consuming.

Once our group developed some designs for a few of the filters, a problem was coming up with a streamlined solution for dividing the coefficients between multiple decimator/interpolator blocks. At first we tried using an array that would divide up between the two, but this implementation didn’t quite work out. Another solution we tried was to convert the information to matching data types and work with them that way, but this lead to other errors.
Future Work:
- Working implementation of polyphase filter on the myRIO board
- Testing a Bandpass Filter Design
- Using the Polyphase Design in a Multi-Processor environment
- Developing an efficiency test/testbank that can calculate MACs and efficiency on the myRIO board.
- Using audio effects in the middle of the Polyphase Decimator and Interpolator filters to reap processing speed benefits
References

MATLAB Design

LabVIEW Design

https://decibel.ni.com/content/docs/DOC-16650

http://edge.rit.edu/content/P14224/public/Software/MyRIO%20Software/Control%20Software/documentation/myRIO%20Custom%20FPGA%20Project%20Documentation.html