Audio Pre-processing Python Model

Michael Aguero & Aydan Olaez

1 Introduction

These files aim to create a python model of the audio pre-processing modules that will be implemented into SystemVerilog. Additionally these files will be used to verify said RTL models later down the road. The general outline of the audio pre-processing is: MEMS PDM Mic - CIC Decimator Filer - FIFO - Short-Time Fast Fourier Transform - Filter Bank - Spectrogram

2 MEMS PDM Mic

A MEMS PDM microphone is a sensor that converts pressure waves into a digital signal encoded in pulse density modulation(PDM). In our project to create a low-power speech identification with machine learning ASIC we chose the IM69D130 High performance digital XENSIVTM MEMS microphone:

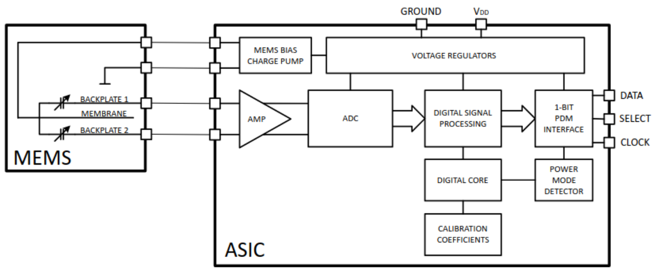

Fig. 1. IM69D130 - infineon - add citation

2.0.1 MEMS Transducer

A variable capacitance that coverts the change in air pressure to a voltage, in the python model this will be omitted.

2.0.2 Amplifier

The Amp. buffers the voltage from the transducer and provides the correct voltage the PDM modulator, in the python model this will be omitted.

2.0.3 PDM Modulator

Coverts the analog signal from the amplifier into a high-frequency 1-bit pulse-density modulated stream. PDM is a way to represent a sampled signal in single bits.

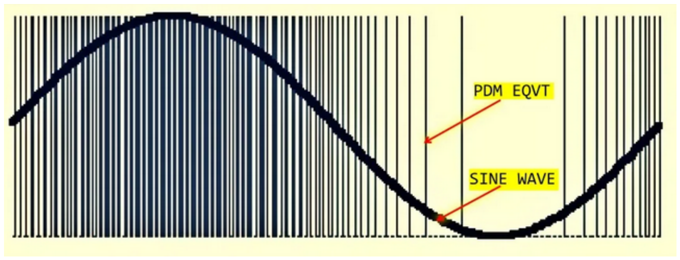

Fig. 2. PDM of sine wave - EDN.com add citation

This is done by first-order delta-sigma modulation, which is also used in other oversampling ADCs and DACs:

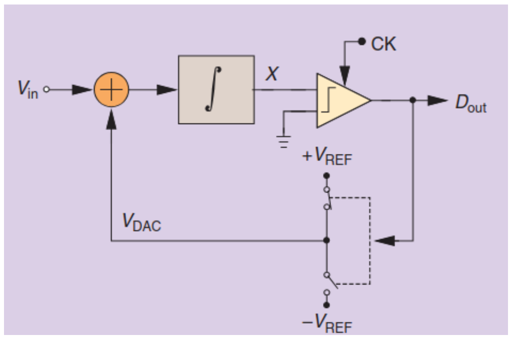

Fig. 3. The DSM - Behzad Razavi UCLA

3 CIC Decimator Filter

Since the PDM microphone will be outputting a high-frequency 1-bit stream and our proposed design will require a 16-bit output at a sampling rate of 16kHz. We must convert the pulse-density modulation into pulse-code modulation(the standard form of digital audio in hardware). This can be done by reducing the sampling rate(decimation) and increasing the word length, the preferred method to do this is a Cascaded Integrator Comb Filter.

The filter consists of an equal number of integrator and comb filters alongside a decimator. This describes a single stage CIC filter, however we can improve attenuation by increasing the stages of integrators and comb filters.

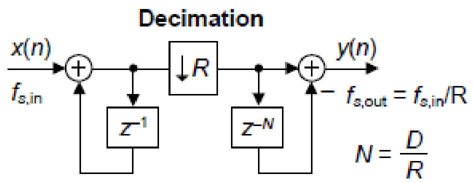

Fig. 4. single-stage CIC Decminator Filter - dpsrelated.com

For this design we choose the number of stages to be N = 5, this provides a good spot. We can analyze this decision by looking at the magnitude response and transfer function of the CIC Filter:

$$H(z) = (\frac{1 - z^{-R}}{1 - z^{-1}})^N$$

$$|H(f)| = (\frac{sin(\pi f RM)}{sin(\pi f)})^N$$

In these equations M = 1 and $R = f_i/f_o$, in this design we are assuming we are using a 1.008 MHz global clock(low-power) for the PDM microphone clock and the output sampling rate of 16kHz. This gives us a R value of 63 for our equations. In context this means that there will be 63 PDM bit per PCM sample. Additionally, upon analyzing these equations in MATLAB we find that the outputted frequencies will have less gain as the frequencies increases. This in turn will cause the higher frequencies bins on a spectrogram(input to ML model) to be low and potentially cause issues for ML speech detection, this is called pass-band droop. To combat this we can employ compensation FIR Filters since ”in typical decimation filtering application we desire reasonably flat pass band gain and narrow transition region width performance”.

3.1 FIR Filter

The combat pass-band drop the FIR filter’s freq. magnitude response should ideally be an inverted version of the CIC filter pass-band magnitude response:

$$H(z) = \frac{9/8}{1 - (1/8)z^{-1}}$$

Add citation, The FIR filter computes each output sample as a weighted sum of recent input samples, in this there are taps. Which consists of a register, filter coefficient, multiplier, and adder. Increasing taps will allow us to increase compensation, however for the ML model a range of taps between 15 and 31 will work. In this design we will choose for the Compensation FIR Filter to have 15 taps. Below is an example how a FIR Filter can help attenuate the output of the CIC Decimator Filter:

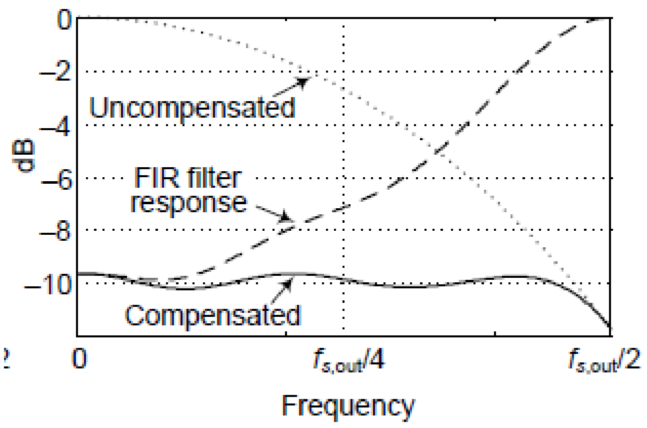

Fig. 5. 3 stage CIC Decimation Filter - dspsrelated.com

4 FIFO

To separate the streaming audio from the frame-based STFFT we will be using a FIFO as buffer with a ready-valid interface, this is well-documented and will be left out of the model description.

5 Fast Fourier Transform Core

In this section we will introduce the concept of the Fast Fourier Transform algorithm. However we will not be designing a FFT Core in SystemVerilog. Instead we will be using an open source pipe-lined FFT by ZipCPU.

5.1 ZipCPU FFT Core

Gisselquist Technology's ZipCPU created an open source pipelined FFT generator, this allows us to generate a custom FFT core for the ASIC. This can be done by downloading and building the repository that is available on GitHub by ZipCPU. Once the $make$ command is complete there will be a $fftgen$ program in the $sw/$ directory. Using this executable alongside parameters we can build the custom core:

```
./fftgen -f 512 -n 24 -x 2 -p 5
```

Here is a list of what the parameters do:

- -f: sets N, the sample size. In our case we have a 512 point FFT
- -n: sets the input width. In our case we have a 24 bit input width
- -x: increases the twiddle factor bit size. In our case we have a twiddle factor bit length of 14 (increasing the bit length for the Twiddle Factor helps prevent truncation errors)
- -p: sets the numbers of DSPs used. In our case we are using X DSPs

5.1.1 Double Clock FFT Core I/O

Below is a figure of the FFT Core and its inputs/outputs:

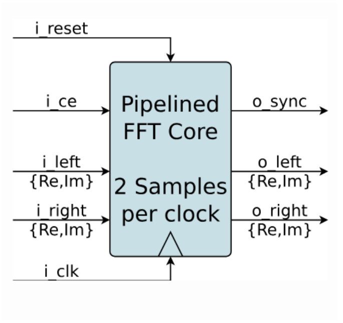

Fig. 6. Dbl-Clk FFT - ZipCPU

5.2 Discrete Fourier Transform (DFT)

The DFT is the finite version of the continuous Fourier Transform for signals at N instants separated by T sample times. Now let N samples be:
$f[0], f[1], f[2], ..., f[k], ..., f[N-1]$
This shows us the bounds of the summation. Now using the definition of the continuous Fourier Transform, we can denote the DFT as shown below since we are treating the data as if it was periodic:

$$F(n) = \sum_{k=0}^{N-1} f[k]e^{\text{-j*(2pi/N)*nk}}$$

6 Short-Time Fast Fourier Transform

The STFFT is a form of the previously mentioned FFT, however in essence it computes multiple smaller FFT over certain windows. This allows the STFFT to determine audio signals with overlapping frequencies and frequencies that change over time(like speech).

Since we are now continuously moving a window across an incoming audio input to simulate many smaller FFTs across the signal. We must introduce different forms of measurement for the FFTs range. First we have the window size/frame size(in our case window and frame size are equal). The window size is the signal we are going to apply a singular FFT to. The window size can be measured in time and number of samples:

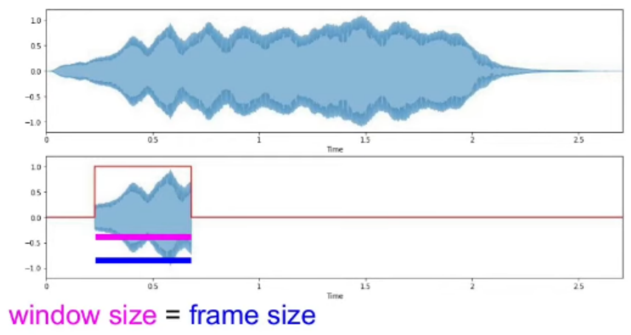

Fig. 7. Window/Frames - Valerio Velardo

Next is the hop size, this is the time or amount of samples before a new window will begin. It is essential to have windows overlapping to prevent aliasing and errors in output(hop size is less than window size): Now it is important to keep in mind the time-

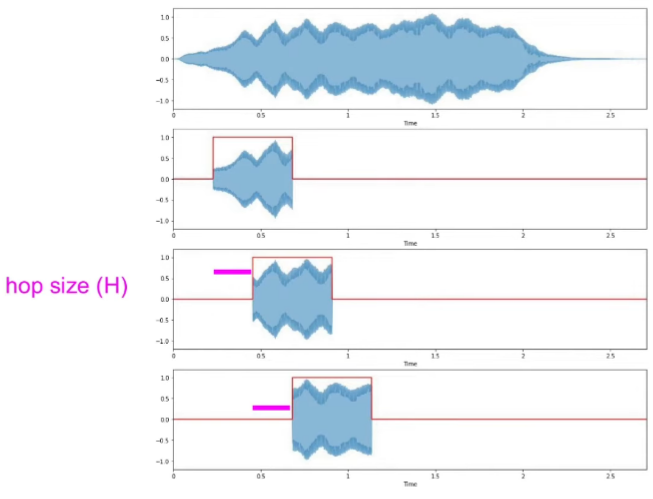

Fig. 8. Hop Size - V. V add citation

frequency resolution of the STFFTs output. As the frame size increases the frequency resolution increase however the time resolution decreases. Additionally, the hop size dictates how many FFTs are computed in a given audio sound. Before we find suitable values for the window and hop size. I will denote that the FFT core will be 512-points to be accurate but efficient. This allows us to find the frequency resolution of just the FFT core not the STFFT:

$$\Delta f = \frac{16kHz}{512} = 31.25Hz$$

However to find the true frequency resolution we must look at the frame size. However, what do we denote as a good frame size or a bad one. Let us first look at our application: speech. It is found that the pitch period for speech is approximately 5-12ms(add citation). Since we want our output to have varying harmonics to gather data from we should choose double the max pitch period, 25ms. This works out as the range which speech appears stationary over smaller region is 20-30ms(add citation). Additionally we can show some math behind our reasoning as well:

$$\Delta f \simeq \frac{1}{frame}$$

Given that our frame size is 25ms:

$$\Delta f \simeq \frac{1}{25ms} \simeq 40Hz$$

Back to hop sizes, for a hop size we want to choose middle ground time period. If the hop size is to long(closer and closer to the frame size) there will be less FFTs computed resulting in a lower time resolution. However if we were to have an extremely small hop size this would be computationally expensive but also decrease the frequency resolution. Knowing this we must use speech dynamics to determine the range of a potential hop size. Earlier we mentioned that pitch periods range from 5-12ms. Since this period denotes areas of similar signals we can pick safely pick $\simeq 10ms$ as the hop size. The frame and hop size are subject to slight alterations in the future.

6.1 Output of STFFT

The STFFT will output for each frame 512 bins, however bins 256-511 are complex conjurgates that are not needed. Each bin will be around 40Hz due to the frequency resolution we calculated earlier. Before we go to the log-mel spectrogram we must convert to magnitude. In practice this will just be the Real component of the signal squared.

7 Log-Mel Spectrogram

After the STFFT we will convert the output to the log-mel scale. This is done as human hearing perception works off a logarithmic scale, frequencies increase humans have a harder time distinguish between them.

$$m = 2595 * log(1 + \frac{f}{50})$$

To extract the Mel Spectrogram, you must take STFFT then convert the amplitude into decibels(dBs) and then convert the frequencies to the Mel scale.

7.1 Freq to Mel Scale

To convert frequencies to the Mel scale you must first choose the n number of mel bands(will add more later). Then construct the mel filter banks:

1) Convert lowest and highest frequency to Mel using the formula
2) Create n bands equally spaced points
3) Convert points back to Hz
4) Create triangular filters***

Then finally you mist apply the mel filter banks to the spectrogram.

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.