EELE 466
Austin Rosenbaum
Final Project: Logarithmic Shift

Final Report

Introduction

The goal of this project was to modify the Frequency Processing block shown in **Figure 1** to achieve some perceptably different effect on the incoming audio signal. The simulink model for this project was developed in Lab 5. The specific change I attempted to make was a logarithmic shift in the frequency domain to better preserve the harmonic relationships in the source audio signal.

Figures and Descriptions

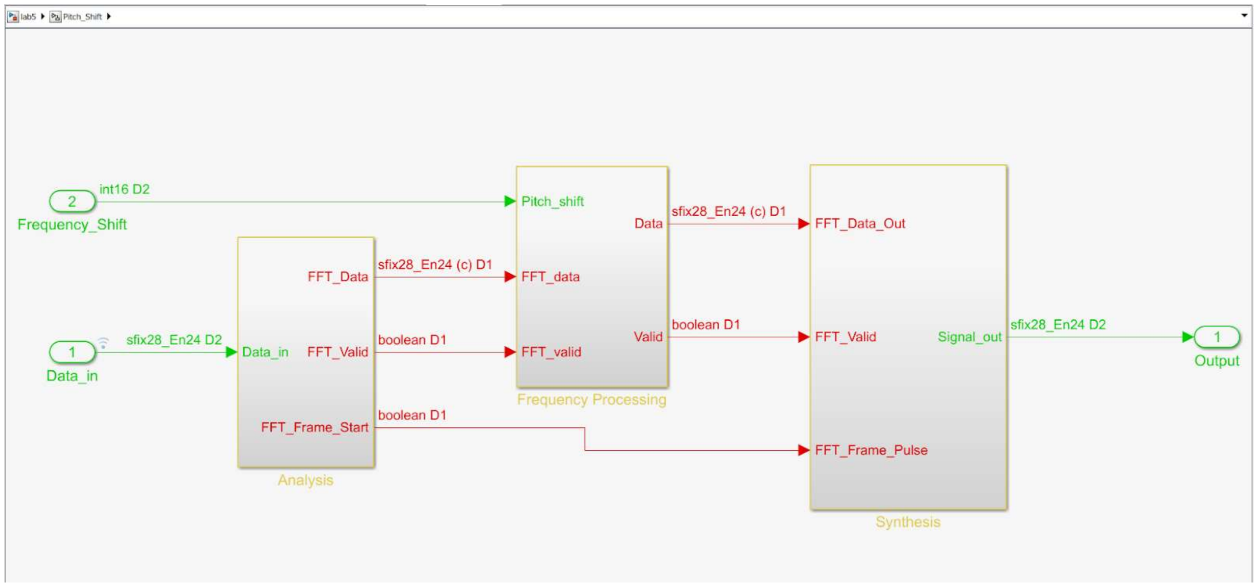

Figure 1 Toplevel pitch shift block

Figure 1 shows the three main blocks required for this lab. The Analysis block was responsible for generating the 1024 bit frames and running them through the FFT to produce the frequency domain signal. The Frequency Processing block contained primarily the Matlab script which applied the shift to the frequency domain vector from the FFT. Lastly, the Synthesis block applied the IFFT to the shifted output and performed the overlap-and-add to reconstruct the waveform in the time domain.

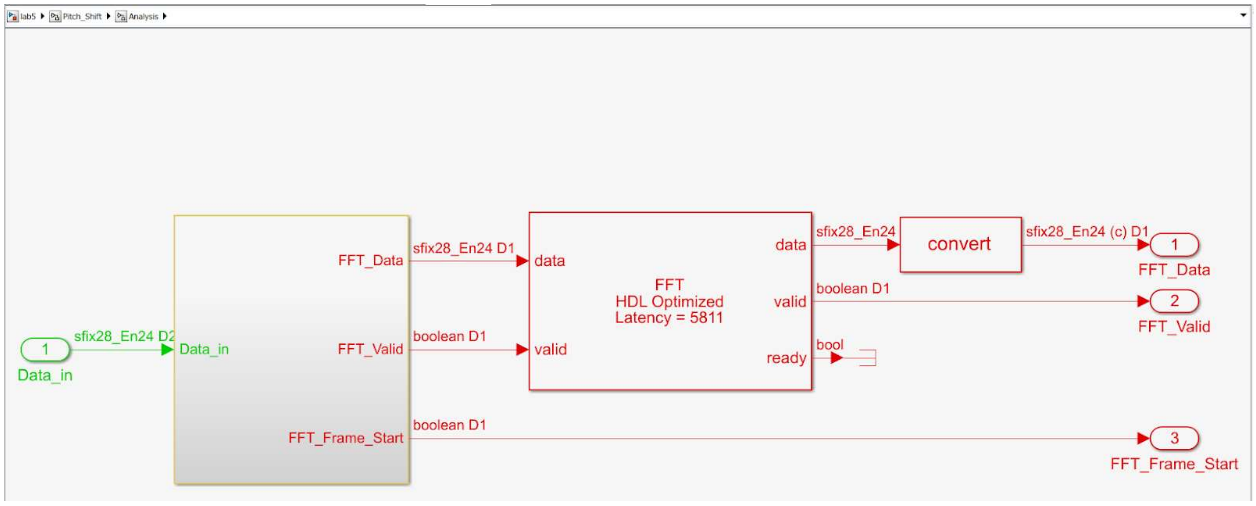

Figure 2 Analysis block with memory block (left) and FFT block.

Figure 2 shows the Analysis block in more detail. It is made up of two major parts, the FFT block and the FFT control block. The FFT control block produced the valid signal to tell the FFT block when to begin processing, as well as the correctly framed data. Additionally the control block produced the FFT frame start signal which controlled the push and pop signals for the FIFO blocks in the Synthesis block, shown in detail in **Figure 5** and **Figure 6**.

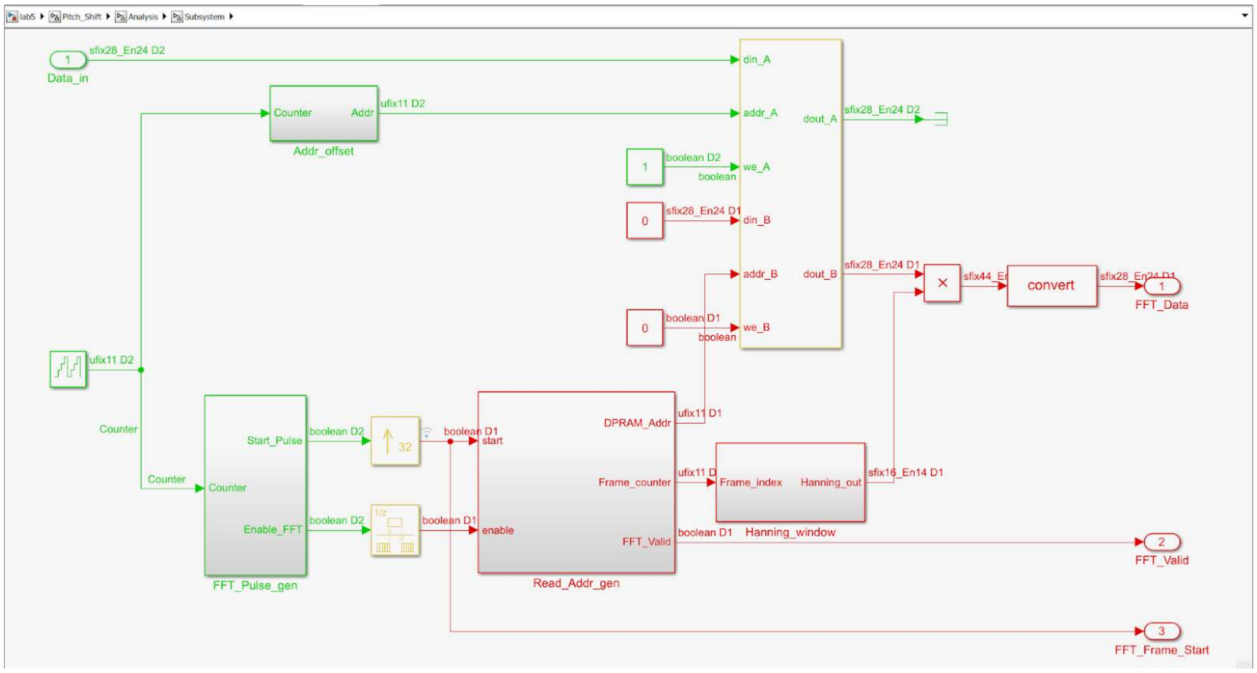

Figure 3 Dual-port RAM and Data Shifting block

The block shown in **Figure 3** contains the dual-port RAM used for storing the incoming audio signal. The write pointer for the input signal is controlled by a counter that counts to the maximum address size of 2048. This is offset from the actual counter by 1 to avoid conflicts with the output read pointer. Once every 256 counts of this counter, a pulse is generated that triggers a 1024 point read from the output port of the RAM. This only occurs after the fourth pulse generated to allow sufficient time for the RAM to fill with the incoming signal.

One important aspect of this block is to transition the sample rate up so the frames created could be processed faster than the input sample rate. The block labeled 'Read_Addr_gen' is responsible for generating the 1024 read address locations as well as offsetting the read pointer by 768 (1024*0.75) to create the overlapping frames for the FFT block to follow. The frame index output of this subsystem allowed the Hanning window to be applied to the outgoing data from memory. Finally, the output was sent to the FFT along with the appropriate valid signals and pulse.

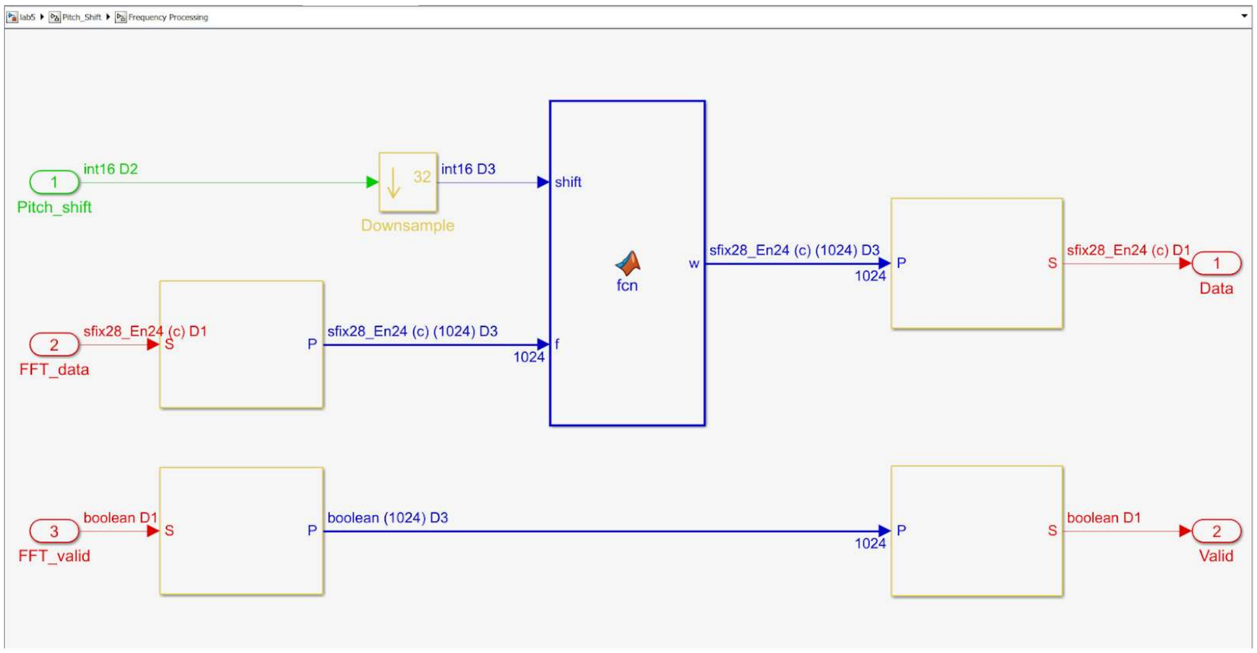

Figure 4 Frequency processing block

Figure 4 shows the frequency processing block in full detail. In order to be able to process the frames as vectors in the Matlab script, they needed to be parallelized using the serial to parallel conversion blocks seen at the left. Once the frequency adjustments had been made, the resulting vector was then serialized to be passed to the rest of the system. It can also be noted the FFT_valid signal was converted from serial to parallel and then back with nothing in between. This was done to match any latency that might have been caused by the similar conversion of the FFT data, and was mostly done as a cautionary measure. Also, the shift value was downsampled to match the parallelized data because Matlab required the input rates to its function blocks be the same.

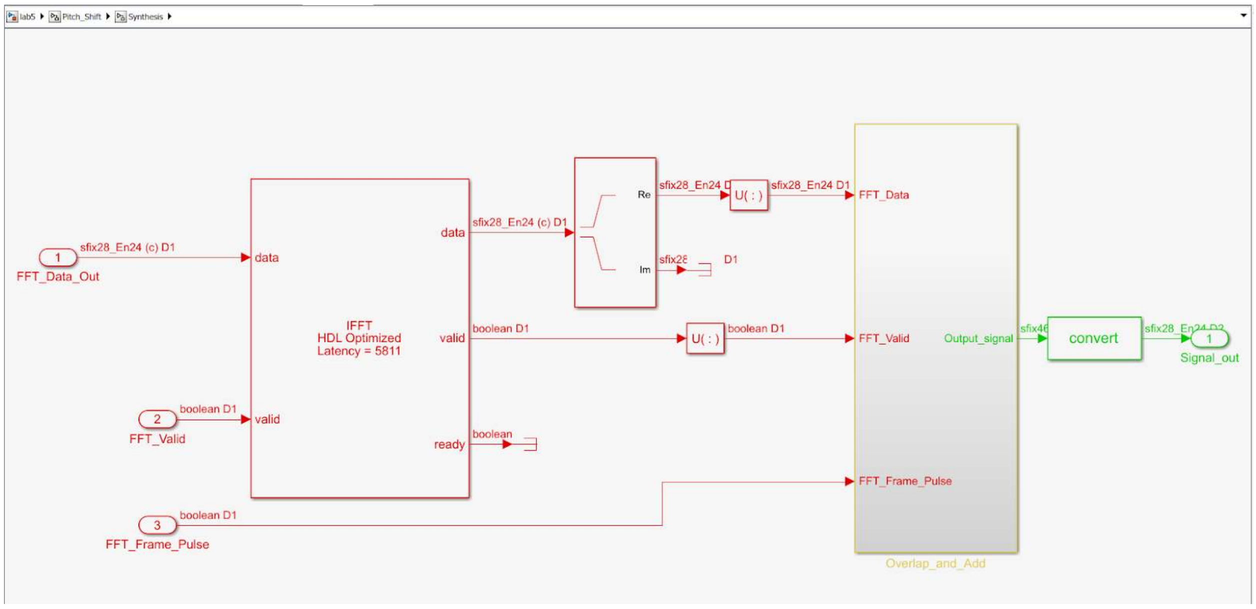

Figure 5 Synthesis block with IFFT and Overlap and Add block (right)

Figure 5 is the Synthesis block, which had two primary parts: the IFFT block and they Overlap and Add block. The IFFT converted the processed data from the frequency domain back to the time domain, which was then split in to its real and imaginary parts, of which only the real was used further. The Overlap and Add block was responsible for stitching together four IFFT outputs to reconstruct the modified original signal. This was necessary as the original signal was sampled in 1024 bit windows offset by 256 bits and then run through a Hanning window, and in order for successful reconstruction, four windows needed to be combined to compensate for the losses which occurred during the Hanning process.

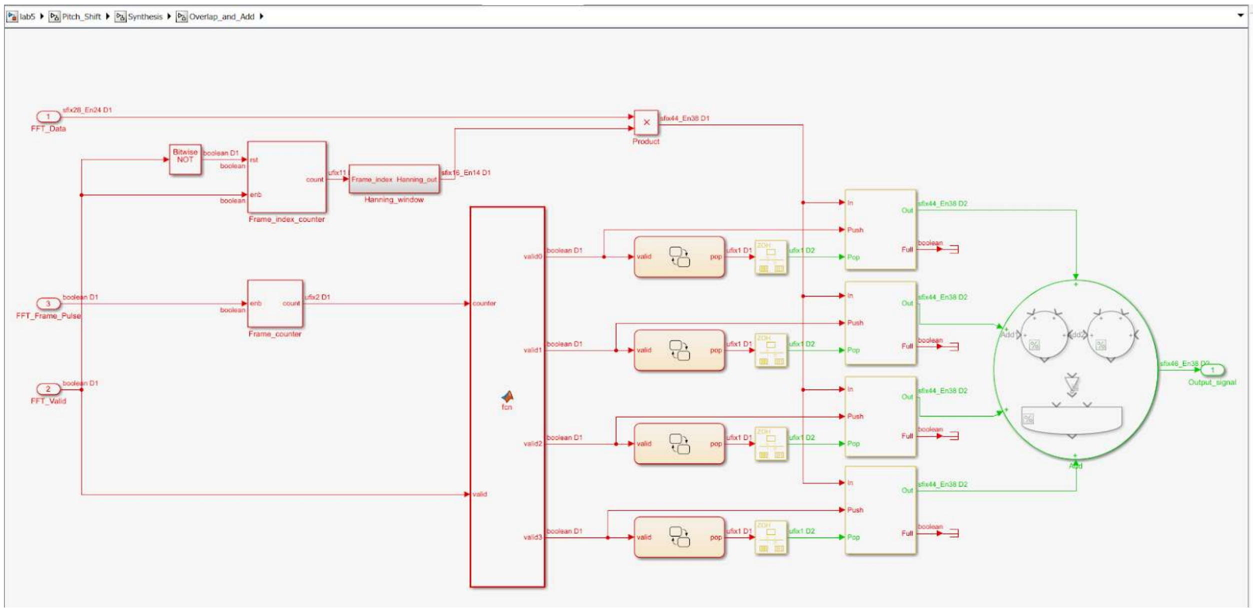

Figure 6 Overlap and Add block

Figure 6 shows the Overlap and Add block in detail. Here we see the reapplication of the Hanning window to once again prevent discontinuities in the output signal. Following this is the four FIFO blocks controlled by individual identical state machines. These state machines were themselves controlled by a Matlab function, which provided the appropriate valid signal for the state machines and the FIFO push based on the frame pulse and the FFT_valid signal. When a pulse was generated on the frame pulse signal, the matlab script would pass the FFT_valid signal through to one state-machine and FIFO pair. The state machine would assert its output on the falling edge of the valid signal, and hold it indefinitely, causing the FIFOs to continuously output any data they got in at their output rate. The valid signal would continue to toggle and control the push signal of each FIFO. The FIFOs operating in this manner essentially became streaming buffers, allowing for the overlap to occur prior to the signals being added together.

Signal Spectrograms

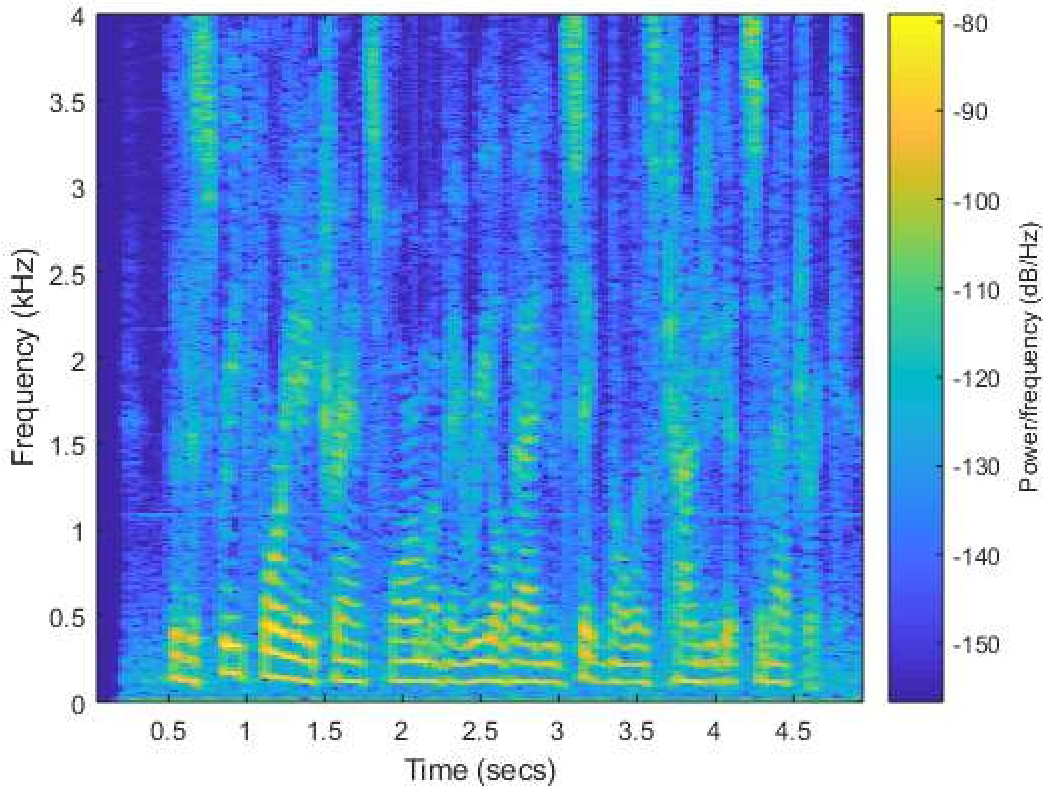

Figure 7 Spectrogram of the source audio signal with no frequency manipulation

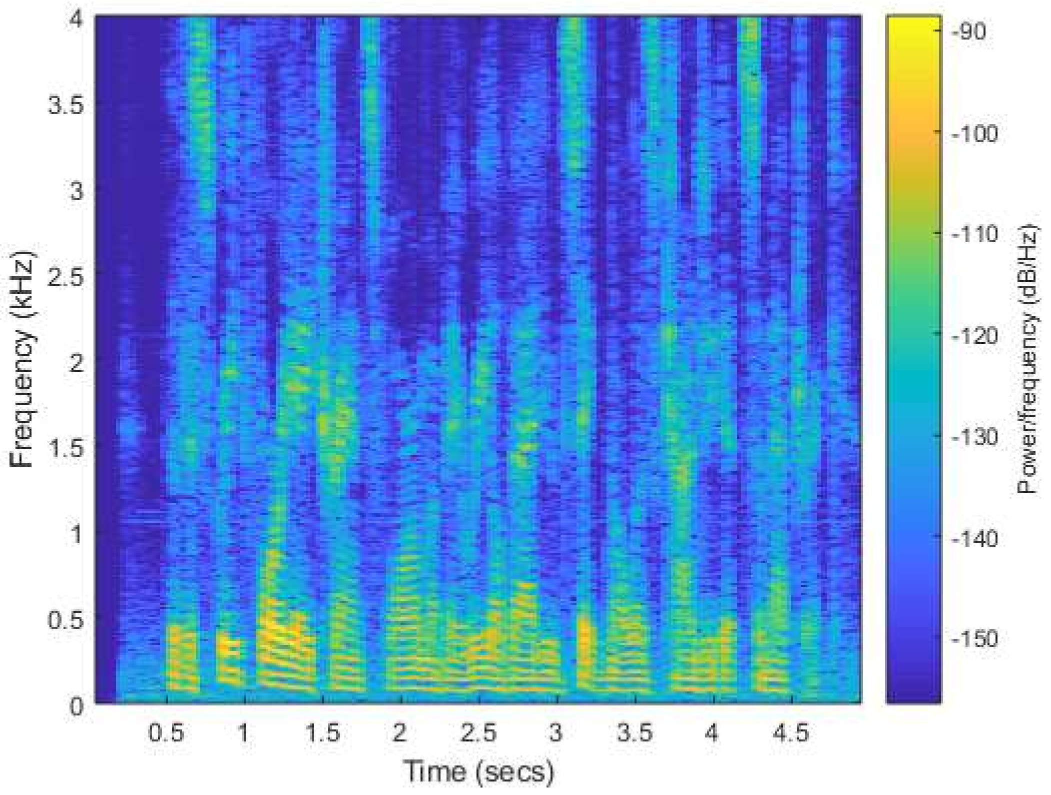

Figure 8 Spectrogram of the modified audio signal using the logarithmic shift and a frequency shift input of +50

Conclusion

As can be seen by **Figure 7** and **Figure 8**, there is a notable difference in the two audio signals. The most noticeable part is arguably the yellow colored strips in the < 500 Hz area. The modified signal has these strips much closer together vertically. The final sound effect achieved a much more guttural sound than the original.

The shift was performed by taking the natural logarithm of the incoming shift value and then offsetting into the data set by this value. One interesting note is that the linear shift offset was preserved so the frequencies are still moved significantly. However, the actual shift becomes much more subtle when done with the logarithmic scaling, and in my opinion makes the sound a bit more interesting.