ECE 1718 - Final Project

Juan Camilo Vega

Mohammad Ewais

Project: Audio Filtering and Mixing with the DE1 SOC Board

For this project we created an audio blending and filtering project using the exponential moving average approach. This was combined with a digital piano with an enhanced feature set compared to the lab. The block diagram of this design is shown below.

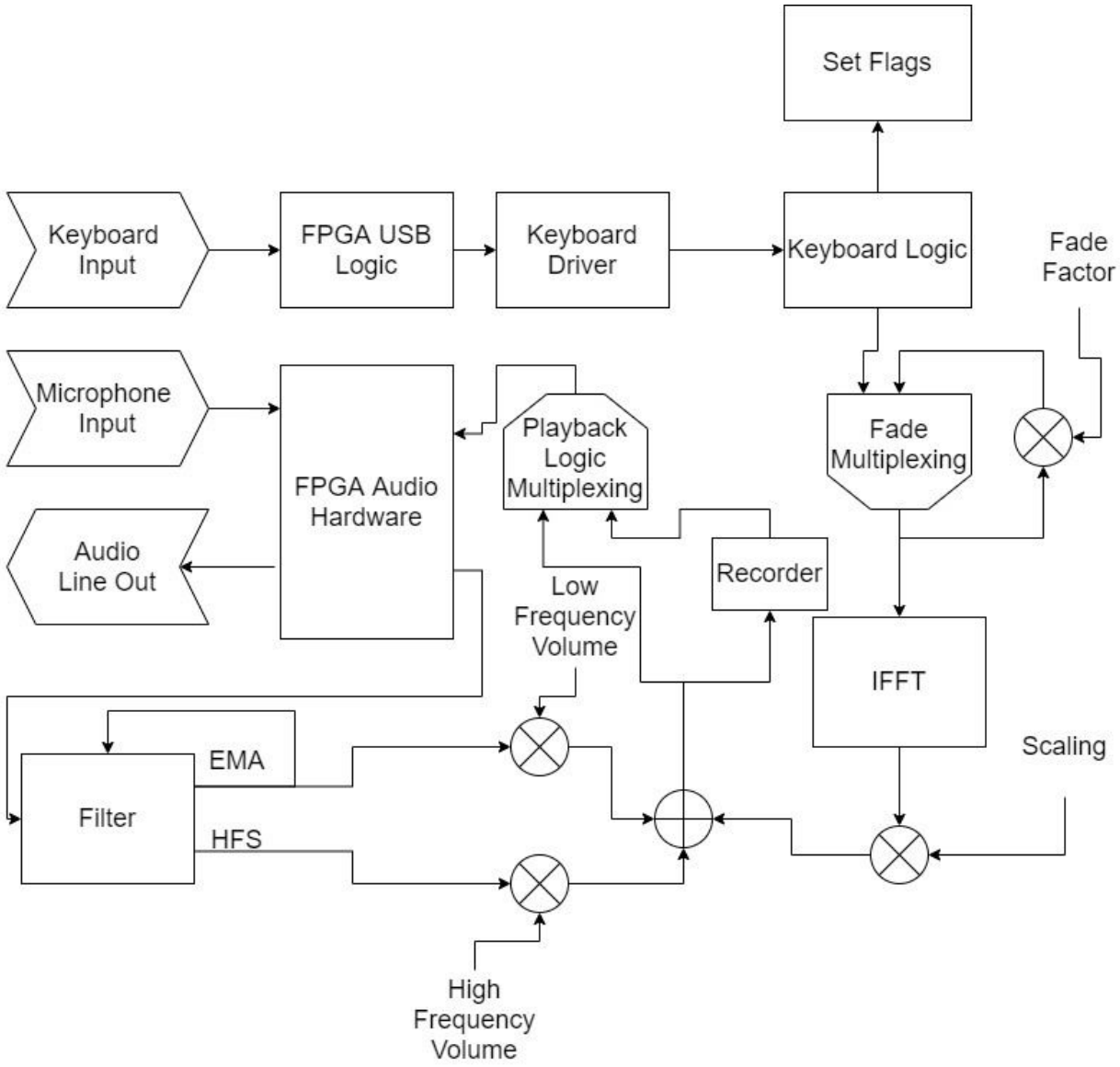
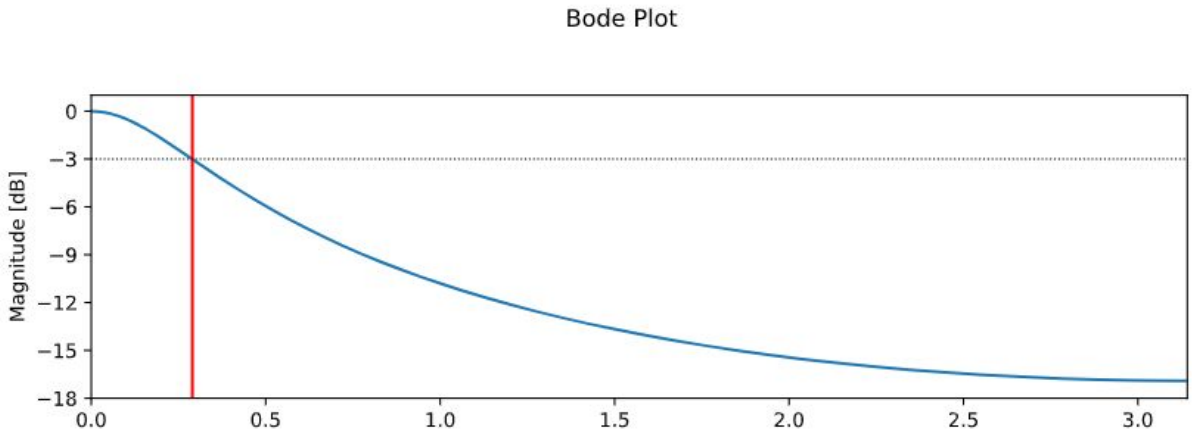


Figure 1: Diagram of the signal flow. IFFT is a misnomer, it's simply a frequency to time transform by playing the required signals.

Filter Design

In this project we filter the signal to divide the high frequency components from the low frequency components and give the user a method to individually control the volume of the two components. This is a functionality that is typical of an analog sound board but we have performed this digitally. For this filter design, we want to ensure that the sum of the high and low frequency signals sum to the original signal. This is needed to ensure that if the volume of the two sides is the same then the signal will pass undisturbed. This eliminates the possibility of using a low and high pass filter since those provide non-uniform attenuation of the signal in the middle and the sum does not add up to the original. We also wish the bode response of the signal to flatten towards a non-zero infinite asymptote so that both components still carry fragments of the entire signal (some low frequency sound in the high signal and high frequency sound in the low signal) so that if one of the volumes is turned fully down it doesn't sound choppy since it is not a cutoff but rather a drastic minimization of the other component. The algorithm in [1] shows a digital filter called the Exponential Moving Average filter with the following bode plot:



As desired it smoothly passes low frequency but progressively passes less of the higher the frequency. It asymptotes at -17 decibels, as desired it never cuts off a signal. The algorithm is EMA(t) = EMA(t-1) * B + (1-B) * V(t) where V is the input signal at time t. B is a calibration constant where the closer B is to 1 the higher the cutoff signal shown in red.

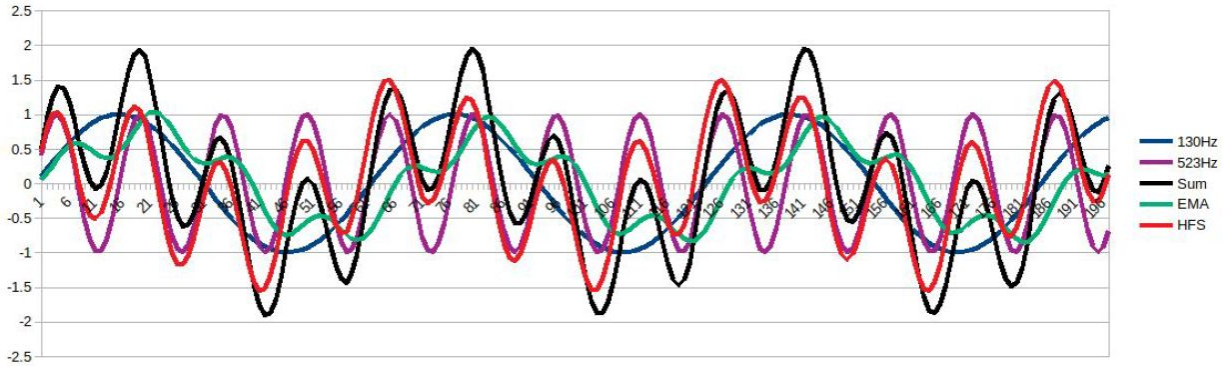
To satisfy the requirement of the sum, we note that since low frequencies pass through this algorithm, the signal that is blocked will likewise contain more high frequency components than low frequency components thus we can define a new quantity we call the high frequency signal (HFS) as:

HFS(t) = V(t) - EMA(t). Since we have access to EMA(t-1) we can combine it with the equation above to get:

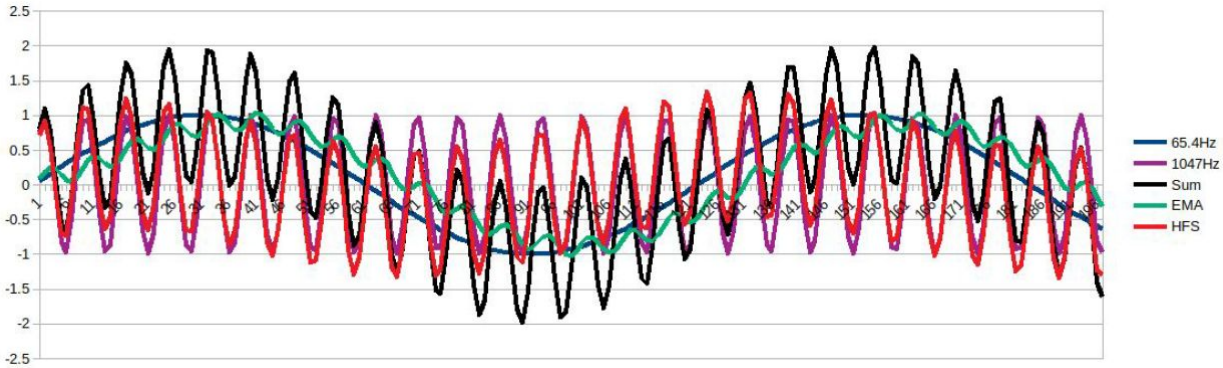
HFS(t) = B * (V(t) - EMA(t-1))

As shown in the diagram the filter produces both of these signals and the EMA is looped back to the filter for the next cycle.

We used excel to test this algorithm to find the optimal B. We won't show all our steps, but we will show the results of the optimal B we found that was 0.9. We produced a 130Hz and 523Hz signal which correspond with C3 and C5 in the piano, and we summed the two signals feeding them into a filter with B=0.9. The diagram shows the result of this test. The EMA signal follows the 130Hz input closely with a bit of a lag and with some HFS components showing. The HFS follows the 523Hz signal very closely too. The lag is not very concerning since the ear can not perceive the phase of the individual harmonics and the sum still gives the original signal (the two phase shifts cancel out).



We also ran the test with 65.4Hz and 1047Hz (C2 and C6) and the results are shown below. The correlation becomes much stronger since these two signals are closer to the extremes.



Credits:

We use a credit system to determine when we can read/write to the audio filter. The audio hardware has memory mapped registers that tell us, for the left and right FIFO the amount of data available in the read FIFO and the space left on the write FIFO. At initialization, we have no credits, and we read these four values. We then set our number of credits to be the smallest of these four values as it is the number of reads/writes we can safely do. We then proceed with the algorithm decrementing the credit count by 1 every time we do a read/write and when the credit count reaches 0 we read these registers again. Since we are faster than the 8KHz sample frequency, usually we will have no data available on the FIFO and will loop until there is data available when we finally get a credit.

Getting the output signal:

The logic for this is shown in Figure 1. On the keyboard side, we have an array called FADE which is set to 0x3FFFFFFF when a key is pressed and is reset to 0 when the key is released unless the dampening pedal is pressed (space). In that case, the signal is not reset but a flag is raised saying that the signal should become 0 when the pedal is released. This flag is reset if the key is pressed again allowing us to track the unpressed keys that still have signal playing. This is how the pedal is implemented. Each cycle the FADE is multiplied by the FADE Factor which was set to 0.9998 allowing the volume to decrease over time in an exponential factor matching the curve normal to pianos. The Fade is then scaled by multiplying it with a volume parameter and the result is added to the output signal. The other source of the output signal is the filtered mic input. After being filtered we have the HFS and EMA values. We multiply them by their respective volumes and they are added to the output signal. This signal is sent to the recorder that records it if the record flag is set and another copy goes to the output mux that selects if the playback signal or the output signal should play. This then gets written to the audio write FIFO. This whole logic is repeated for the left and right side though the keyboard stuff is not repeated since it is the same for both sides.

Flags and Volume:

The keyboard was also used for flags and volumes. O set the recording flag, P set the output mux to play back the recorded data. A and Z raised and lowered respectively the volume of the low frequency components (EMA). S and X raised and lowered respectively the volume of the high frequency components (HFS).

Difficulties and Improvements:

Our biggest challenge in this assignment was dealing with the data types. We needed to use 64 bit data and we found out after a lot of debugging that in ARM the long data type is 32 bit, not 64 bit. We had to instead use int_64 but it was hard to discover this since all we had was noise in our output and we had to find out why. We also incorrectly took the FIFO data to be unsigned (we assumed it was the instantaneous sound pressure and didn't consider that it could be negative) and that caused many issues as we lowered the volumes and our negatives would become very high positives. The last challenge was that the mic-in only accepted older 3.5mm jacks. In other words, the board expects the older audio technology where speakers would have their own green colored 3.5mm jack, and mics would have a separate pink colored 3.5mm jack. Newer combined 3.5mm jacks simply didn't work with the mic-in. This was especially hard because we had to buy older technology speakers+mics (which we couldn't find), or find stand alone microphones which were very expensive (high quality recording microphones). Eventually, we found a splitter cable (one 3.5mm to 2 3.5mm for speakers and mic) and were able to use it with the board.

In terms of possible project improvements, we did everything we wanted to do and many of the improvements are trivial. If we have more audio input channels we can easily extend this to become a digital sound board adding the resultant signal from each filtered mic together (after applying the individual volumes). Another cool feature would be the ability to layer signals together. That is to play back a signal and then sing on top of that signal to allow one singer to create a vocal orchestra. This would not be hard to do as it would just involve looping back the playback line to the addition for the output signal and allowing based on a flag to add it in. Third, since the recorded data is a simple data structure, we can enable saving that data to a file and loading it from a file easily. A more complex improvement would be to add audio effects to the signal though the math for those can be more involved.

Conclusion:

We have demonstrated that we can build an audio sound board on the DE1 board that mixes a digital keyboard with a singer. This mixing also allows the user to individually tune the volume of the low and high frequencies in this signal and the resultant signal is very clean with no distortion if the two volumes are the same.

[1]P. Pieter. Exponential Moving Average. https://tttapa.github.io/Pages/Mathematics/Systems-and-Control-Theory/Digital-filters/Exponential%20Moving%20Average/Exponential-Moving-Average.html. pdf, 6, 2019