DESIGN DOCUMENT FOR AUDIO EFFECTS PROCESSOR

Lowell Jacobson and Hanneul Earl Han
EE E4840

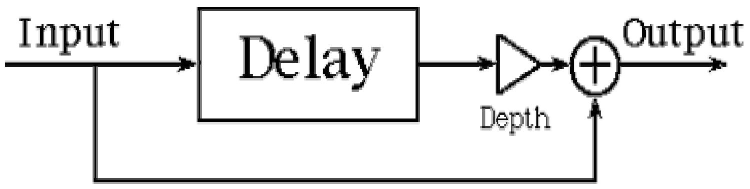
- ## Echo/Reverb/Flanging

  Echo, *Flanging*, and *Reverb* cannot be implemented at the same time. For the purposes of this project, we consider them to be incompatible because of the essential similarities between them as well as the lack of usefulness for producing something like an echoed reverb or an echoed flanging. Through the interface, the user can choose to do any of the effects, or none.

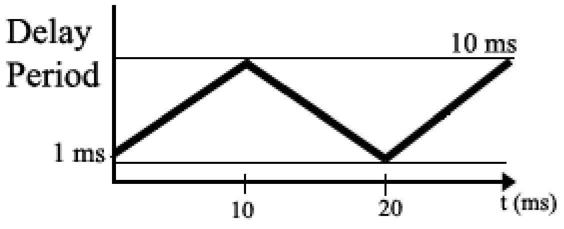
Echo/Reverb Bits (from control)

B_1	B_0	**Audio Effect**
0	**0**	**None**
0	1	*Flanging*
1	0	*Echo*
1	1	*Reverb*

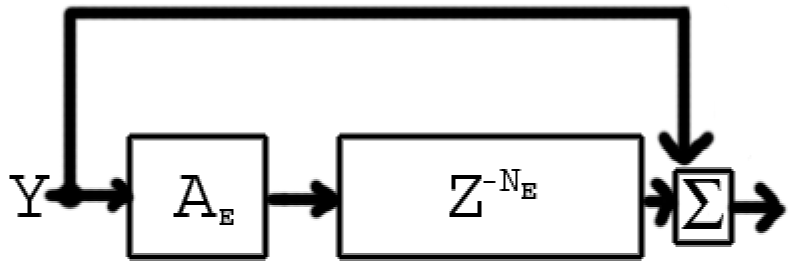
- **Implementation and Work to be done**:
  - ***Flanging***: Figure out attenuation value
    - **Delay Time** is variable and constantly swept between 1 and 10 ms and back.



- While flanging looks simple from a diagram, appearing very similar to Echo, it is in fact much more complicated due to the fact that the delay period is itself periodic (in our case, varying from 1-10 ms), rather than a constant value.
  - Flanging will be implemented similarly to echo, but with a delay time utilizing an up/down (1 -> 10 -> 1 -> 10 …) counter linked to a delay element (or an array of delay elements), in VHDL. For example, the first 1 ms of audio data input to the flanger will be delayed by 1ms at the output, the 9th by 9ms, the 10th by 10 ms, the 11th by 9 ms, and so on.

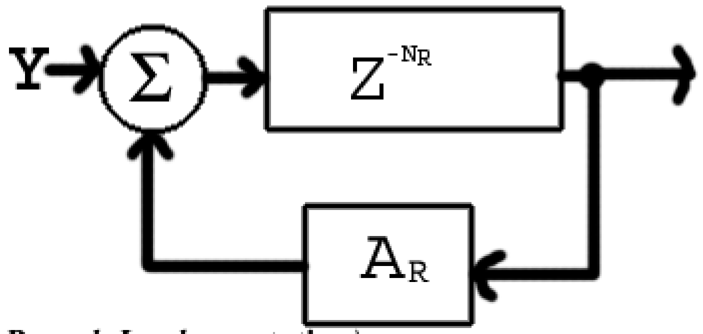


        - ***Echo***: Figure out delay time (**N_E**) and attenuation value (**A_E**)
            - **N_E** should be somewhere between 50 and 70 ms



(*Echo Implementation*)

        - ***Reverb***: Figure out delay time (**N_R**) and attenuation value (**A_R**)

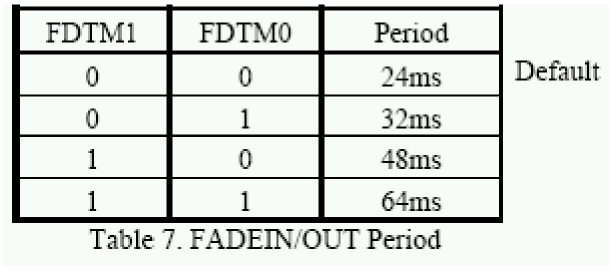


(*Reverb Implementation*)

- **Fade-in and Fade-out effects**:
  The fade effects will be controlled in real-time via the on-screen display (OSD), which will communicate with the board using `minicom`.

    - The bits **FDIN** and **FDOUT** are located at registers **D2** and **D1**, respectively, at address **06H** (Operation Mode) on the register for the *AK4565*.
        - If either bit is set to zero, that function will not be implemented. If both are set to zero, neither will be implemented.
        - Note that as soon as a bit is set to 1 for **FDIN** or **FDOUT**, that effect will be implemented by the CODEC, thus making real-time communication essential for this effect.
            - If **FDOUT** is held at 1, the sound will stay muted until it is set to 0
        - If both bits are set to 1, the sound will fade out, then fade back in with the specified parameters

- The bits **FDTM1** and **FDTM0** control the period of the fade-in or fade-out. They are located at registers **D7** and **D6** respectively, at address **03H** (Timer Select). Through manipulation of these bits, the period of the fade can be adjusted as follows:



FDTM1	FDTM0	Period	
0	0	24ms	Default
0	1	32ms	
1	0	48ms	
1	1	64ms	

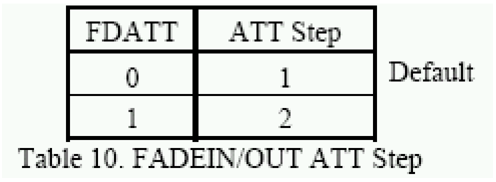
Table 7. FADEIN/OUT Period

  - If these bits are to be changed, they must be changed ***before*** **FDIN** or **FDOUT** are changed, because when **FDIN** or **FDOUT** are set to 1, they use the current values of the **FDTM** bits to determine the period for that particular fade.
  - The period does not determine the overall length of the fade. Rather, the step per period (see **FDATT**) and the strength of the signal determine how quickly the fade seems to work.
    - The fade length formula is as follows:

$$\text{Time}_{\text{Fade}} = (A_{\text{Signal}} - A_{\text{Floor}}) \frac{\text{period}}{\text{dB}_{\text{STEP}}}$$

Where A_{Signal} is the Amplitude of the signal, in dB and A_{Floor} is the lower limit in dB that the human ear is able to discern at the current Signal – to – Noise ratio of the listening environment.

- The bit **FDATT** controls the steps in dB that the audio signal increases or decreases per period:



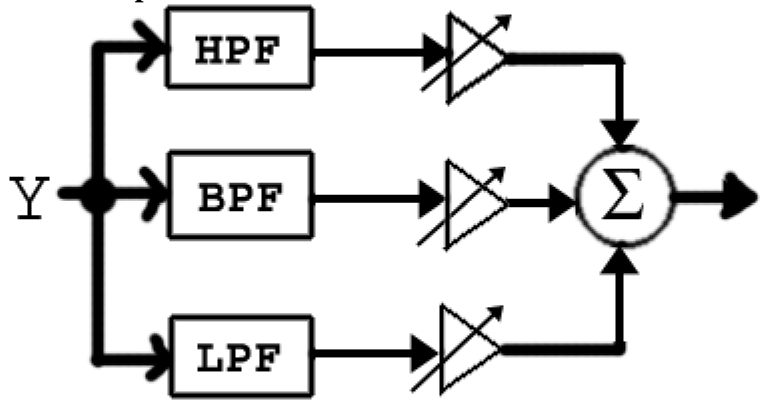
FDATT	ATT Step	
0	1	Default
1	2	

Table 10. FADEIN/OUT ATT Step

  - With default settings, when **FDATT** is 0, the signal's magnitude changes by 0.5 dB/period. When **FDATT** is 1, the signal's magnitude changes by 1 dB/period. The direction of the change is dependent on whether a fade-in or fade-out is currently in progress.

- **Tone Control**
  Tone Control will also be done in real time, via the on-screen display. Tone Control works by splitting the audio signal into a mid, bass, and treble through a band-pass, low-pass, and high-pass filter, respectively. Each of the three frequency ranges will have its own variable-gain element to increase or decrease the strength of that segment of the audio spectrum. The gain of each range will be adjustable, in real-time, by the on-screen display. Following the variable gain elements, the signals are recombined to form a re-mixed signal.

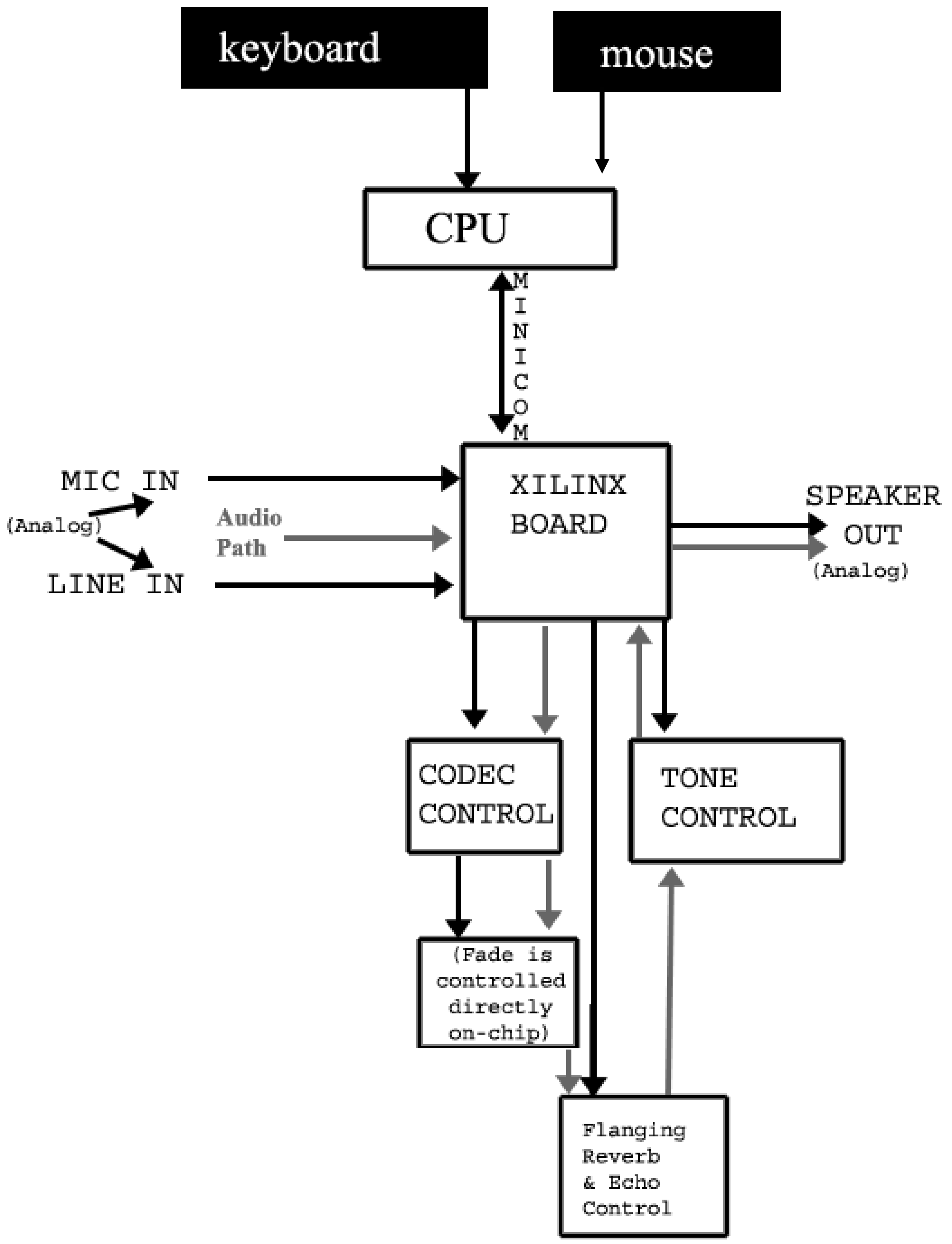
Implementation of tone control:



  - o The filters themselves will all be FIR filters implemented in VHDL. Since FIR filters are very easy to make, this will not take up much memory. In addition to their small size, the FIR filters have the added benefit of not shifting phase, which is desirable in an audio-processing application such as this.
    - Both the **HPF** and the **LPF** will be simple low-order filters (assuming audio quality is sufficient), both for size considerations and for the fact that we want to ignore everything below or above, respectively, the cut-off frequency.

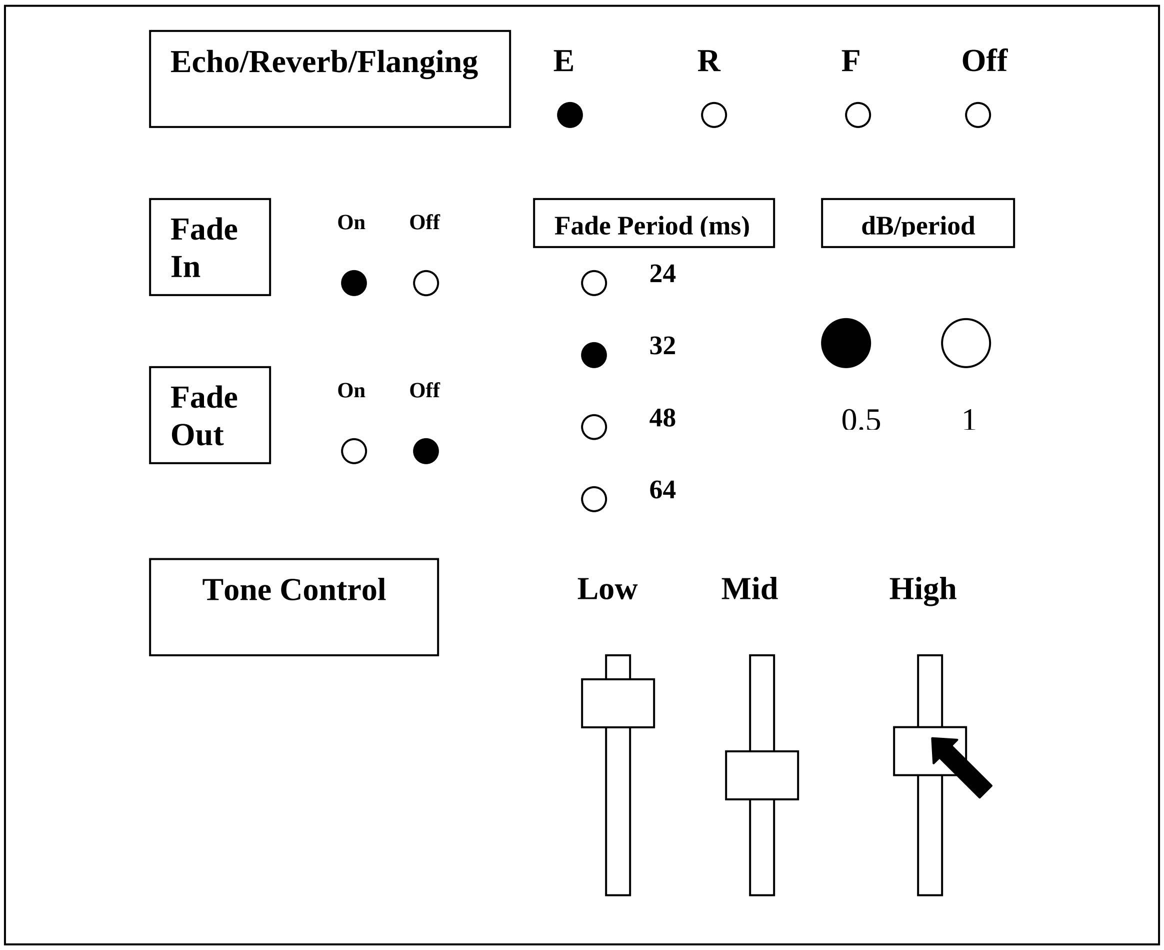
  - o **Target Filter Values (Cutoff/Crossover frequencies):**
    - HPF: f > 2 KHz
    - BPF: 200 Hz < f < 2 KHz
    - LPF: f < 200 Hz
  - o **Work to be done:**
    - Write filters in VHDL with the given cutoff values
    - (If audio quality is not good enough) Implement higher order filters in VHDL for the BPF and LPF.

Logical Diagram for Audio Signal and Controls for Audio Effects Processor



- **Audio On Screen Interface**

- The Audio Interface will list on screen audio manipulation options for user. The user will enter his selection via the mouse.



The C programming will print the graphics for the interface on the screen, and enable mouse support. The user will use the mouse and click on the appropriate box to enable and disable a certain option. The C program will also communicate in real-time with the Xilinx board, and update the values of the registers for Fade, as well as enabling various VHDL elements for Echo/Reverb/Flang and controlling the variable gain elements of the Tone Control.

- **Implementation Work**

- Code mouse graphic and support scroll up and down refresh of the image
- Enable mouse-click support
- Communication with Xilinx board