Lab 4: A Reverb Program

Introduction:

Part of what gives a full and rich sound to listening to music in a concert hall is the fact that we the music reaches us several times---directly from the stage, and then indirectly, with some delay, after reflecting from the walls, ceiling, etc. of the room. Normally the direct path has the largest amplitude, and the reflections, or reverberations, have smaller amplitudes. The same effect is a big problem in other systems; perhaps the most important example at the moment is what is called ‘multi-path’ in communications systems such as cell phones, where the radio waves to and from the cell towers bounce of buildings and other obstacles and thus again several delayed and attenuated copies arrive at the receiver. We will demonstrate the audio version of this reverberation effect with our ADSP-BF535 system in this experiment.

Let's consider 4 delayed reflections, so that at any time n the output sample $y(n)$ is given by,

$$y(n) = a_0x(n) + a_1x(n-N_1) + a_2x(n-N_2) + a_3x(n-N_3) + a_4x(n-N_4).$$

It is relatively easy to realize this effect with the delay operation. We have to add 4 scaled delayed samples to the scaled current sample, with proper care to avoid overflow. In this experiment you will set the constants $a_0$ through $a_4$ as \{1,1/2,1/4,1/8,1/16\}.

We will modify the talkthrough program that we used in Lab2 to do this experiment. The BF535 allows the use of both hardware and software circular buffers. In this lab you will be asked to write your own index control to simulate a circular buffer in software. Although this is not as fast as a hardware circular buffer, it will have the benefit of making you think through what the index control operations need to be.

In your code you will define a linear buffer to save the current sample $x(n)$ and some number of previous samples $x(n-1)$, $x(n-2)$, ... , in the main function or process audio data function, for example,

```c
volatile short TDM_DELAY_BUFFER[SIZE_OF_DELAY_BUFFER];
```

Here, `SIZE_OF_DELAY_BUFFER` is the constant you define in a header file or within the file of “Process audio data.c”, indicating the total number of samples we will save in the buffer. For example,

```c
#define SIZE_OF_DELAY_BUFFER 48000
```
To access a previous saved sample in the circular buffer, we need an offset in time from the current sample. Since we will use this linear buffer as a circular buffer, you need to control the loop index so that when it reaches the end of the buffer it wraps around to the correct location.

The easiest way to do this is by using the modulo operator. For example, if we were going to have an 8 sample delay and a buffer size (or length of the delay line) of 16 samples, the 8-sample delayed value from the current sample could be accessed like this:

Delayed_value=TDM_DELAY_BUFFER[(N+8)%16];

where N is the current index into the TDM_DELAY_BUFFER as written by the RX interrupt handler. So, when the a given RX data comes in, the RX handler places the valid current input sample (saved in the variables of “sLeft_Channel_In” and “sRight_Channel_In” for stereo input, and in the variable of “sLeft_Channel_In” only for the monotone input) of the input frame into the correct location (say, for example purposes, at the beginning of the buffer, TDM_DELAY_BUFFER[0]). Meanwhile we can pick up the current sample and 4 previous saved values (which you can access via the technique we above), weight them as we want, and add them together), and then send the result out to the speaker (by writing into the variables of “sLeft_Channel_Out” and “sRight_Channel_Out”). After this is done, we increment the index, do the same operation till the index comes to the end of the circular buffer. At this point, we need to set the index to zero, forcing the index to return to the starting point of the simulated circular buffer.

**Exercises:**

1. Modify the talkthrough program posted on the web site to be able to simulate the reverb. In particular, allocate enough memory to the simulated circular buffer you define to be able to generate delays up to 1s with a sampling frequency of 48KHz. Use the microphone input for this program and make sure that the left and the right output channels get the same input from the microphone.

2. Simulate reverberation effects using delays of duration 100ms and 200ms (you may redefine the size of the circular buffer or modify the sampling frequency). Demonstrate your results using the microphone input and record your observations.

3. Modify your program so that we have stereo input and reverbed stereo output (with the same amount of delay in each channel). Simulate delays of 100ms and 200ms. Demonstrate your program using stereo input and record your observations.

4. To get a realistic “concert hall” effect, we need some delays of small durations. Modify your program in part 3 above by playing with the scaling factors and delay values, until you can experimentally demonstrate the concert hall effect.
using this new program and stereo inputs. You may want to keep in mind the homework problem we did from Chapter 4 of the Proakis-Manolakis book, which pointed out an important requirement of the delays to get a reverb effect.

If you have finished all the exercises above and still have some time, you can consider the next experiment. Note that this is a preliminary writeup, but if you have time it is a good idea to start on it.

II. An FIR Filter Design

One of the routing processing in DSP is a filtering operation. This operation is typically implemented in practice using a difference equation approach. In the last experiment, we have studied some basic operations, such as scaling and summing of input or output samples to obtain a processed sample, we will continue these operations in this experiment.

Digital filters are commonly classified as FIR (Finite Impulse Response) and IIR (Infinite Impulse Response) filters. FIR filters are characterized as Moving Average (MA) filters, and we will try to design an simple FIR filter in this experiment. The technique we should use to realize an FIR filter is the same as we have used in the last reverberation experiment. You should modify your program for the reverberation to realize the difference equation shown below,

\[ Y(n) = \_ x(n) + \_ x(n-1) + 1/8 x(n-2) +1/16 x(n-3) +1/32 x(n-4) \]

Exercise:
1. Modify your program for reverberation implement the above MA equation: Determine and plot the magnitude response of the above filter (use Matlab if necessary) and verify it experimentally using a sinusoidal source.

2. Using the above filter, experiment with speech signals and record your observations.

3. Modify your program to process stereo inputs separately using two different filters (one for each channel). Use the sampling rate of 48KHz and the above filter in each channel. Experiment with stereo input and discuss your observations.