Elementary DSP Operations

Now that we understand how to perform simplest operation A/D and D/A conversion on the Blackfin 535 using VisualDSP++ and the Talkthrough program, we are ready to begin to study how to implement three elementary DSP operations. These are sample shifter (data move), scalar (constant multiplier), and adder operations. The constant multiplier scales each sample by a constant, while the adder adds two signal samples to form a sample of a third signal. As discussed in class, the “multiply-accumulate” operation is the central operation in many DSP algorithms, including convolution, difference equation implementation, DFT transforms, etc. Thus, with these three operations we can then move on to more complex processing such as reverberation, convolution with FIR filters, and implementation of difference equations for IIR as well as FIR filtering.

In this lab we will look at simple use of sample shifting. The sample shifter will be able to provide a shift of as little as one to several sample-intervals or even shifts of few seconds.

In particular, in this lab we will simulate an acoustic delay-line by delaying the input signal by noticeable amount. Before you begin, please think about the factors that affect how long a delay you might be able to realize with a given hardware configuration. In other words, suppose you want to delay an audio signal by 1 second. What aspect of your A/D and D/A will affect what your hardware must be able to provide to achieve this delay? Please try to answer this in your notebook.

The single most important programming concept in this experiment is the use of circular buffer to simulate a delay. The incoming data is stored in the buffer, whose length is equal to the total amount of delay (in number of samples) that we want to simulate. The current stored value in a buffer location is first transmitted to the serial port and then the incoming sample value is stored at that buffer location (overwriting the currently stored value). The buffer location pointer is then advanced by one and the process continues. Because the buffer is set up to be circular, each stored sample is thus read out (that is, sent to the serial port) after one rotation through the buffer (which is equal to the length of the buffer, again counting in samples), thus providing the necessary delay.

Exercise 1
Have a look at the main.c and Process ISR.c programs in the currently supplied version of the Talkthrough suite (remember to make your own copy before you modify anything), and be sure to allocate enough memory to rxbuffer and txbuffer to generate delays up to 1 second. (Make sure you explain how you arrived at this value, and how you implemented it, in your lab notebook.) We will use microphone input for this program, so you should make sure that the left and the right output channels get the same input from the microphone.
**Exercise 2**
Now use the setup from Exercise 1 to simulate delays of 100ms, 500ms and 2 seconds. Demonstrate your program using microphone input and record your observations.

**Exercise 3**
Try to modify your program so that the left output channel sends out the current sample $x(n)$ while the right output channel sends out the delayed sample $x(n-k)$. Simulate delays of 500ms and 1 sec (that is, make the delay $k$ for the right output channel the correct number of samples so that the delay on that output is first 500ms, and then 1 second). Demonstrate your program using stereo input and record your observations.