EC581 DSP Projects: Lab Project #4

Dept. of Electrical and Computer Engineering
Rose-Hulman Institute of Technology

Digital Audio Gain Compressor

Last Modified on 3/22/2002 (KEH)
Modified for 6711 DSK on 29-Mar-2004 (may)

1. Introduction

Audio systems such as portable voice memo recorders, voice radio communication equipment, public address systems, hearing aids, and even electronic surveillance equipment (bugs), often incorporate some form of audio gain compression (AGC), also called automatic volume control (AVC). An AGC system compresses the "dynamic range" of an audio signal by automatically decreasing the system gain if the average magnitude of the audio signal (averaged over a 30-ms time window, or time "frame") gets too high. Likewise, the AGC system will increase the system gain if the average signal intensity gets too low.

A public address system or radio transmitter typically includes an AGC so that if the speaker backs away from the microphone, the AGC will automatically increase the gain in order to keep his voice at the same average intensity level. Likewise, if the speaker moves closer to the microphone, the AGC system will automatically decrease the gain, in order to again keep his voice at the same average intensity. Without the AGC system, a "sound man" would have to monitor the system output, and manually "ride the gain" control. Unfortunately, the sound man cannot react to sudden changes in audio level instantly, hence there are usually sudden short-term surges and fade-outs when a human is in the loop.

What about the frames of silence that occur between bursts of speech or music? We do not want the audio gain to be turned up "wide open" between bursts of audible speech, since this would only serve to emphasize the background noise. To keep this from happening, if the average signal intensity falls below an arbitrarily assigned "no signal" threshold value, then we shall assume that this is a "silent frame". The gain for a silent 30-ms frame will be set to zero, rather than increased. This guarantees that there will be silence between bursts of speech, rather than greatly amplified background noise.

2. Digital Audio Compression System Project Specifications

In this laboratory project you are asked to write a real-time C program for the TI C6711 DSK that samples audio data at a rate of 48000 samples/second. Your program must break the speech up into 30 ms frames. This is to be done by storing the speech samples in a 0.030 s * 48000 samples/second = 1440 element `buffer`. i.e. shorten `buffer` to 1400*2 samples[1]. During each sample period, you should update a sum variable. The absolute value of the new sample should be added to the sum. Then, when the buffer wraps (the buffer array index variable is set back to zero), the sum of the last 1440 absolute values can be divided by 1440

[1] Since you are changing the size of `buffer`, be sure to get a new copy of `DSK_Support.c`. The old version doesn't' resize the EDMA configuration.

to determine the average audio magnitude value (`aveMagOfFrame`) for the 30 ms frame. Then all the values in the current buffer are then be multiplied by the scaling constant `desiredAveMag/ aveMagOfFrame`, and sent out to the D/A converter. This should happen with each sample, so that after 1440 more samples, the entire "amplitude scaled" frame will be sent to the D/A converter. Note that each frame will be scaled (each of the 1440 samples in the frame will be multiplied by the scaling constant) to have the same average magnitude value, "`DesiredAveMag`". Also, note that the output of this digital gain compression system will lag the present input by 30 ms, which is a barely perceptible delay.

If the `aveMagOfFrame` falls below a "zero threshold" value (`zeroThreshold`), which you will have to experimentally determine, set the entire 1440-sample frame to zero, as you send it out to the D/A converter.

You will need to experiment with appropriate `desiredAveMag` and `zeroThreshold` values. An easy way to try different values is to declare `desiredAveMag` and `zeroThreshold` outside the function and then attached a slider to it. Do this by creating a file called slide.gel and putting the following in it:

```
menuitem "test"
      slider zeroThresh(0, 10000, 10, 1, amp) {
           zeroThreshold = amp;
      }
```

Next go to `File:Load Gel` … and select slide.gel. This creates a menu under the `Gel` menu called `test`. Under test you will see a slider called `zeroThresh`. Check now and see if they are there. The slider starts at 0 and goes to 10000. Watch `zeroThresh` and adjust the slider. Does it change?

Another debugging tool is to add:

```
LOG_printf(&logTrace, "Ave = %d", aveMagOfFrame);
```

to your `processBuffer()` routine. This will print out the value of `aveMagOfFrame` every time a buffer is processed. See page 9-23 step 30 for instructions on setting `logTrace`.

You will also have to decide whether to implement this routine in floating point or fixed point (fixed point may be more challenging, but it can be implemented more cheaply.)

You should test your program with both a microphone and a 400-Hz sine-wave generator, whose gain is varied. You should be able to demonstrate nearly constant output (no more than 1 dB variation) while the output is varied over a least a 10:1 (20 dB) amplitude range.

Demonstrate the proper operation of this AGC system to the instructor by showing that you can move the microphone different distances (from 2 to 24 inches) from your mouth, and yet the average voice intensity will remain virtually constant.