Project 4: Five Band Stereo Audio Equalizer

Using your Verilog components already designed this semester, create a 50 KHz, 2 channel (stereo), 5 band audio equalizer. To do this, implement 5 FIR filters for each channel. Get the coefficients for each band from the EqualizerDesign (Installation instructions) tool shown below:

When you run the program, the filter coefficients ($h_i$) for all 5 bands will be written to a file. The coefficients will be communicated to your Equalizer from a C program running on the Leon. For each band, your C code will also communicate a 16 bit attenuation factor ($K_i$) to set the output levels. For each sample from the ADC, apply all 5 FIR filters in parallel, multiply each output by the attenuation factor for that band, sum the results together and play this signal through the DAC. The block diagram for this system is shown below:
When you put then entire system together it will look like the figure below so that the C code and the Equalize module can talk to the other APB peripherals.

The following is a recommended procedure to complete the project. Note – not every detail is included in the procedure below. Every step should be verified in simulation and in implementation.

1. Create an “expensive wire” demo in stereo in hardware. Configure the SPI peripherals with your C code from Project 3. After they are configured, turn the control of your SPI peripherals over to the EqualizerController APB peripheral (Device 8) which communicates with the ADC APB module and the DAC APB
1. Module to create a loopback connection. Test with an audio source. Verify that you are not dropping any samples with the oscilloscope.

2. Modify Project 2b to multiply the filter output by a 16 bit attenuation factor K (maximum value = 1.0) that attenuates the filter output. The value of K should be programmable from your C code along with the filter coefficients. Make sure you only use 1 multiplier for the filter and the attenuation. Add the low pass filter to the expensive wire demo from step 1 working on 1 channel. Verify your results by comparing them to the results from project 2.

3. Create a new Verilog module that instantiates 10 FIR-K modules. Add the necessary logic on the input and output sides so that each FIR-K can be accessed separately. Verify that all 10 FIR-K modules work individually like in #2 above.

4. Create the verilog model of the Equalizer Controller.

5. Create the C code to transmit the equalizer coefficients to the 10 (5 left and 5 right) FIR-K modules. Add the summing hardware to Equalizer controller. Set K to 1.0 for all channels and test with the equalizer with pure tones in each band. Verify that the output matches the amplitude predicted by the equalizer frequency response. This will require zooming in very closely to the ripple in the passband.

6. Connect a second serial port on the PC to RS232-1 and use HyperTerminal to send a simple command to the Leon to switch between 4 pre-programmed equalizer profiles that you create.
   a. Hyperterminal Setup:
      i. Serial port configuration: 38400-N-8-1, No flow control
      ii. File-> Properties->Settings-> ASCII Setup: Local echo off/Append line feeds to incoming line ends.
   b. printf and getchar() can be used to communicate with the PC. Hyperterminal delays printing to the screen until it receives a newline.
   c. On the Run->External Tools -> External Tools Configurations… in Eclipse you need to set UART Loopback = NO to get the second serial port to work.
   e. Base address on the UART for rs232-1 is 0x80000100 if you want to change the baud rate.

7. Create a Windows Application using C#, Matlab, LabVIEW, python, etc. to graphically control the equalizer configuration with slider bars. Create 4 buttons to implements the 4 pre-programmed equalizer profiles.

8. Verification – you need to prove that the equalizer works in both simulation and in hardware. The easiest way to do this is to set the attenuation factors to 1 and pick two frequencies where the passband ripple is at a maximum and a minimum.
   a. Simulation: Incorporate the adc_tester.v and dac_tester.v Verilog models for the ADC and DAC into your simulations. Verify that the amplitude and phase out of your filter match your prediction.
   b. Hardware: Use Band Pass Filter Demo (Installation instructions) or some other Function Generator to generate the same frequencies as in the simulation. Verify that the amplitudes differ by approximately the same...
ratio as predicted by the frequency response of the Equalizer (see RelativeAmplitudeMeasurementsFromFreqResp.pdf for details).

c. Gui: Verify that when you change the attenuation of any band from 1.0 to 0.5 that the DAC output changes by a factor of 2 for signals in that pass band.

9. Upgrade your GUI and your hardware to use 10 bands for each channel as described in the 10 band equalizer on the Useful Docs page.

Demonstration:

Be prepared to quickly change the equalization (Attenuation Factor for the 5 bands) in your C code to demonstrate that your system is working with both music and the sine waves from function generator.

Report required.

Extra Credit:

- It is possible that your design wastes a lot of RAM. Modify your design so that your 10 band equalizer uses only 12*279*16 bits of RAM for the coefficients and the input values.
- Display the time domain and frequency domain signals of the equalizer output on the PC.