Class D Amplifier with Digital Crossover

Ethan Labelson, EE119c, Spring 2025

Summary

For my EE119c project I implemented a class D audio amp. the amp takes input from an I2S ADC (24 bit, 192 KHz) and outputs 4 channels of PWM: high pass and low pass filters for the left and right channels. The filters are implemented using 2 biquads back to back, and provide a smooth transition with minimal peaking. This allows for the filters to act as a digital crossover and can could be connected in a bi-amp configuration to drive the tweeter and woofer of a speaker seperatly. The power stages are setup for bridge tied load (BTL) which means that both the positive and negative side of the speaker each have their own half bridge. Because of this, I utilize BD modulation (negating the audio signal before it goes through the amplifier) to keep the positive and negative drive signals in phase and minimize the distortion. During the project, I drew some inspiration from this project. The write up/documentation was useful for planning my project, and there are some good tips for sigma delta converters. I also want to make clear that I did not view the source code for the project. The document I found most useful was this documentation pdf for the project.

I2S

The I2S ADC is used for both digital audio input and as a clock source. The onboard 24.576 MHz oscilator provided the global clock signal for my FPGA. The I2S interface was one of the first things that I wrote for my project, and initally had my FPGA configured as the I2S master connected to an ESP32 receiving bluetooth audio. However, I discovered that the bluetooth was dropping audio samples (likely due to a slight clock mismatch) so I swapped over to the ADC with much better results. I used the NXP I2S reference manual to guide my I2S implementation.

Filters

My crossover filters are biquads. I use 2 biquads chained together for a steaper crossover. The filters are run at 192 kHz, just like the incoming audio. They are currently setup with a 2000 kHz crossover frequency for easy demonstration. To compute filter coefficents, I used a spreadsheet I found here: MiniDSP biqud programming main page; direct spreadsheet download.

Sigma Delta

I spent a while tuning my sigma delta coefficents. In the end, most of the issue was caused my a misunderstanding I had about how sigma delta was supposed to work. I was trying to just take the top n bits of the sigma delta output as audio. Instead, the single bit sigma delta stream needs to be decimated into a PCM-esque format before it can be fed into a PWM modulator and output. When trying to tune my constants, I used this calculator/simulator. I went with some some simple powers of 2 when I finally figured out what was going on, but using a calcultor may have given slightly better results.

Performance

At high volumes, the amp produces decent quality audio output. The audio is mostly static free, and sounds like a "medium" quality amp. At lower volumes, there is a lot of static and distortion. I think this may be due to underflow in the sigma delta or filters. Further testing is needed to determine which. I think there is also appreciable noise due to ground loops in the system, dispite my efforts to reduce them.

Reflection

Not sure if this is a needed section, but I have some thoughts on what I could have done better with the project. I think I should have proposed something more like a deadicated DSP chip. The digital audio processors from analog devices, for example, offer a number of digitally programmable filters. I think it would have been cool to focus more on the DSP aspect of things (and add some bells and whistles to make it a full 119c project) and just use an off the shelf class D chip to take care of the sigma delta and pwm output. It's not really relevant to 119c at this point, but I may try to transform this project into something in that direction in the future (just for fun). I wish I had more time to explore things I could've done with DSP to have a properly high quality bi-amp setup.

File Structure

The project is seperated into different folders in an attempt to organize files by purpose. Generally, each folder has top level entity which ties together everything from the subdirectories into a single usable entity. Not sure if this is common practice the way I did it, but I think it made organizing the project much easier. The top level file for the whole project is top_interconnect.vhd. There is one folder called “Unused” which has some files that I wrote but are not active in the final implementation. Mostly this is my untested I2C implementation as well as the old I2S receiver configuration when the FPGA was getting data from the ESP32.