DSP Dude: A DSP Audio Pre-Amplifier

6.111 Project Proposal

Yanni Coroneos and Valentina Chamorro

Overview

The problem we would like to address with DSP Dude is the scenario where there are multiple speakers in a single room but each can optimally produce only one set of frequencies. We imagine DSP Dude as a quickly-reprogrammable digital signal processor that can filter out frequencies a speaker can't reproduce. By chaining multiple of these units together, we can allow every speaker to only produce the frequencies it is good at reproducing. The digital nature and reprogrammability is integral to our design because we intend for this system to be deployable on short notice and in very different speaker arrangements—as is the case in public spaces which host a variety of social events.

Motivation: Digital Signal Processing

A valid question one might ask is why must these audio signals be digitally processed when there exist working analog solutions? Analog signal processors are large, difficult, and ineffective to produce because of the cost and space associated with continuous-time (CT) analog filters. A generally applicable analog signal processor will either need an array of CT filters, which takes up space, or a few CT filters and a complicated voltage multiplier in order to shift the center frequencies of the filters. This completed scheme still doesn't even solve the problem of arbitrary frequency response: what if the user needs a highly specific frequency response whose shape cannot be achieved by just a few linear combinations of CT filters? In the analog domain, this requires a completely custom circuit design—which is prohibitively expensive for just about everyone. There is also the issue of noise and unbalanced signals. Analog audio outputs on most consumer electronics are single-ended, which means that the negative voltage is also the reference ground. This can cause issues at high frequencies or high power because the current running through the ground wire will cause a voltage drop, or a back EMF in the case of a very long cable. When the ground is no longer 0 volts the reference is lost. In extreme cases this can cause a noticeable DC offset on the input to the amplifier which can result in clipping or DC current through the speaker. A digital signal processor with user re-programmable DSP functions could solve every issue previously mentioned while still remaining small in size. A system consisting of only an FPGA and a codec can take the place of all the analog electronics.

Block Diagram

To help illustrate how our project will achieve this goal, we created a block diagram that breaks our project down into its major components: AK4117 input stage, serial controllers, I2S receiver, FIR module, coefficient generation, I2S transmitter, AK4396 output stage, VGA volume visualizer, and the clock gen module.

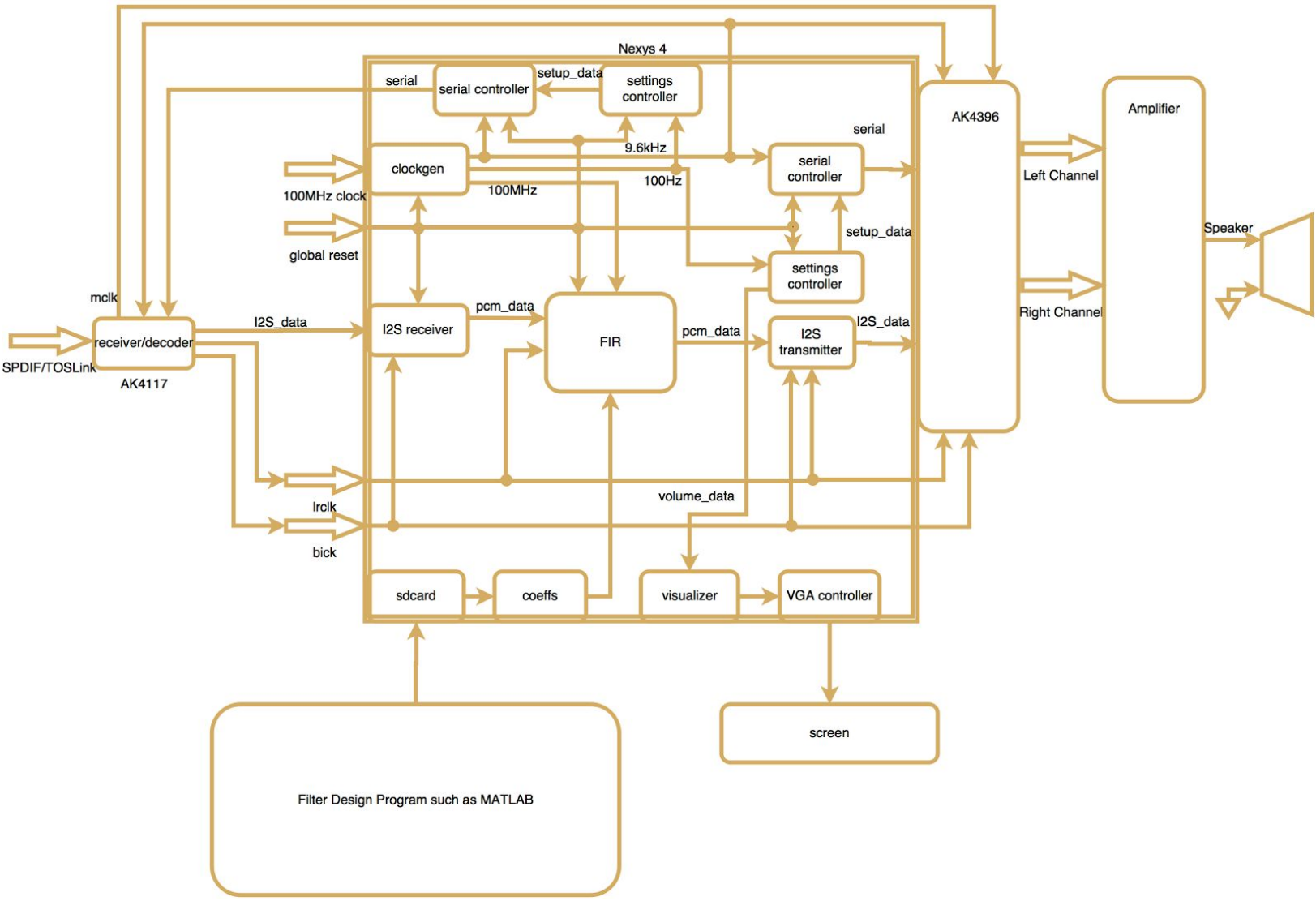

Figure 1 - block diagram of expected functionality

On the far left of Fig.1, in the input stage, the AK4117 receiver IC takes in the SPDIF audio signal from the computer and outputs I2S. In order to do that though, it must be properly configured by our serial controller. After the I2S is input to the FPGA, we run it through our I2S

receiver module to decode the I2S signal into a signed 24bit number containing the audio data and pass it to the FIR filter module. An external program will calculate the FIR coefficients that will also be passed to our FIR filter module which can then appropriately filter the audio signal. After filtering, our I2S transmitter module will convert the signed filtered data into I2S so that the AK4396 codec on our output stage can convert it to analog for our speakers to output. Finally, since the codec can also be configured serially and one of those settings is volume, we will send our desired volume value to a VGA visualizer so that the user can see the current volume.

AKM4117 Receiver IC Input Stage

The audio input is in the form of SPDIF over TOSLink, which is an optical communication protocol necessary for ground isolation and low distortion; however, it is difficult to decode. Specifically, SPDIF is encoded in a Bi-Mark phase encoding that has the clock and data on a single wire. Proper and reliable decoding requires a phase-locked-loop (PLL) which the AK4117 receiver IC provides for us. After being configured by the serial controller, the AK4117 will recover the audio data and sample rate from the SPDIF signal and output the audio data as I2S that gets fed into our I2S receiver module.

Serial Control Module

Both the AK4117 receiver IC and the AK4396 codec first need to be configured to operate for certain sampling frequencies and audio data widths. This is the job of our FSMs inside the serial controller module. On every 100Hz clock pulse, a new configuration command is sent out to both chips. Both chips operate on 16bit SPI, which is a three-wire protocol consisting of a chip-select, clock, and data wire. When chip-select is low, data for the IC is read on the rising edge of clock. We program the AK4117 and the AK4396 to operate in I2S audio mode with 24bits of audio data per channel.

I2S Receiver Module

After being configured by the serial control module, the AK4117 will output valid I2S which can be decoded by our I2S receiver. I2S is a three-wire protocol where there is a sample rate clock, a bit clock, and a data line. When the sample rate clock is low, bits for the left channel of audio data are read off the data line on every falling edge of the bit clock. When the sample rate clock is high, bits for the right channel of audio data are read off the data line on every falling edge of the bit clock. Correct reception entails detecting both the rising and falling edges of the sample rate clock, as well as the falling edges of the bit clock. After recovering the data, the I2S receiver module will output signed 24bit data representing each of the left and right channels of audio.

FIR Module

The central part of DSP Dude is the FIR module that handles the actual audio filtering. At every period of the sample, the FIR module reads the next output from the I2S receiver module. For each sample, the FIR module calculates the result of a multi-tap FIR filter using the provided coefficients and outputs a new transformed sample at the same rate as the samples come in. This is possible because the FIR module is clocked at 100MHz, which is well above our

maximum sample rate of 192KHz. Assuming each FIR iteration takes 3 clock cycles, we can support a 100MHz/192KHz/2/3=86 tap FIR filter. The extra division by 2 is because we must calculate both left and right channel outputs. The coefficients we use for our FIR module will come from the FIR coefficient generator module and the outputs of the FIR module will be two signed 24bit numbers representing the left and right audio channel data. These will be fed into the I2S transmitter module.

FIR Coefficient Generation

The coefficients necessary for our FIR module are generated externally by an engineering design program such as MATLAB. The user will be able to design any type of FIR, linear phase filter and MATLAB can compute a series of coefficients and store them on an sdcard. The FIR coefficient module inside DSP Dude will read the sdcard in order to acquire the FIR filter coefficients and then pass them on to the FIR module. This also means that it's possible to store several different types of filters on a single sdcard and have DSP Dude switch between them at any point. The sdcard will not use a filesystem and will, instead, just store raw bits in a linear ordering. Different sets of coefficients will be marked with special header sequences.

I2S Transmitter Module

The I2S transmitter module will take as input two 24 bit numbers representing left and right audio channel data. It will then encode them into the same I2S protocol that was used for reception and feed them into the AK4396 codec output stage. It is essentially the reverse of the I2S receiver module.

AKM4396 Codec Output Stage

After configuration by the serial module, the AK4396 codec can take in an I2S audio data signal from the I2S transmitter module and output left and right channel analog audio. This will be fed to speakers. Unfortunately, this codec is a complicated beast: we are wiring it to operate in serial PCM mode with split ground planes and split power rails. This is because The Nexys 4 operates on 3.3V logic and the analog circuitry within the codec operates on 5V (see figure 2

below).

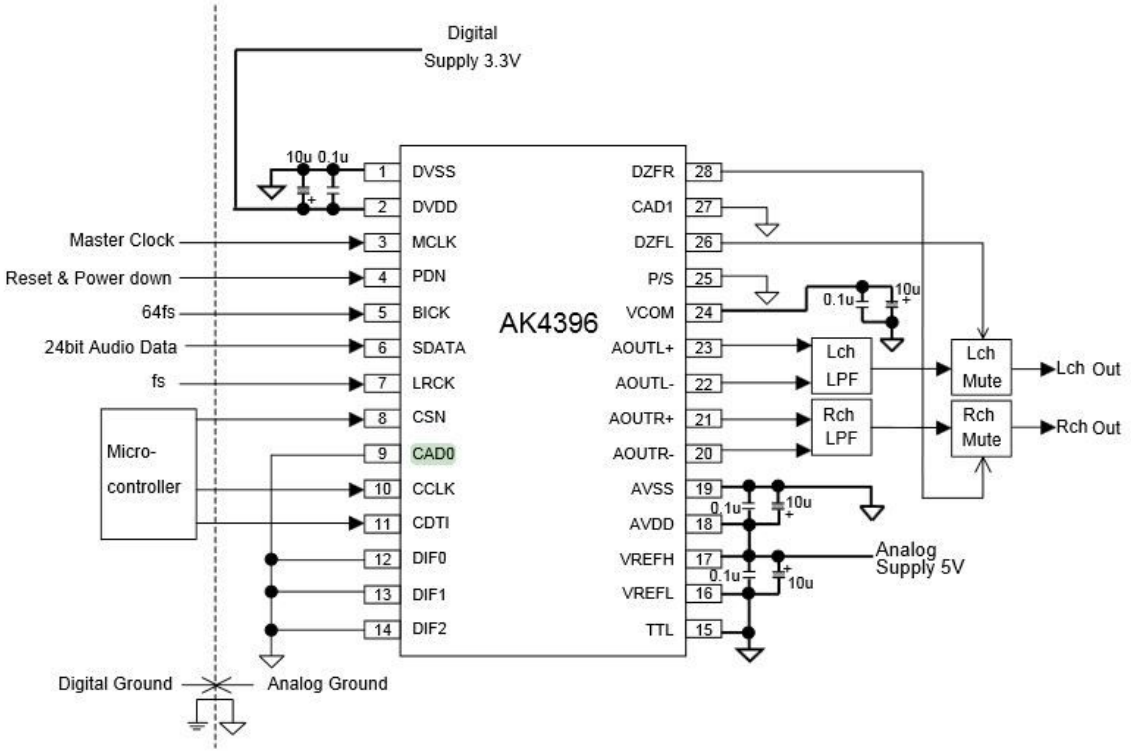

Figure 2 - AK4396 codec wiring diagram

VGA Volume Visualization

To allow users to see the operating volume of the system, there will be a module that outputs a visual representation on the VGA of the volume. As mentioned before, the codec can be configured serially and one of its configurable settings is volume. Using the labkit we can set an 8bit value with the switches that can then be passed to both the codec serial controller and the VGA visualizer.

Clock Gen Module

In order for all of our modules to work correctly, they need different clocks. The AK4396 codec needs a sample rate clock (LRCK 192 kHz), a bit rate clock (BICK 9.216 MHz), a master clock (MCLK 24.5760 MHz), and a serial clock. The serial controller needs the same serial clock as the codec, the I2S transmitter needs the bit rate clock, and the FIR module needs a 100MHz clock. The internal 100MHz crystal clock on the Nexys 4 is divided down to 9.6KHz for the serial clock and the AK4117 generates the BICK and MCLK from the the LRCK that it recognizes on the SPDIF input. The receiver outputs all three clocks which are then fed directly to the AK4396. A clock gen module on the FPGA, takes in the MCLK from the AK4117 and generates a 100MHz clock such that the FIR module is in phase with the data being received from the AK4117. The clock gen module also generates the serial clock for the serial control modules but those do not have to be in phase with the receiver IC.

Gantt Chart

	10/25/2015	11/1/2015	11/8/2015	11/15/2015	11/22/2015	11/29/2015	12/6/2015
clock gen module							
serial controller module							
I2S transmitter module							
settings controller module							
codec outputs sound							
generate FIR coefficients							
r/w coefficients to sdcard							
I2S receiver module							
FIR module							
interface amplifier and speakers with FPGA							
volume visualization module							
VGA controller module							
check off							

Labor Breakdown

1. Yanni:
    a. Serial control for the receiver and codec
    b. Interfacing with the sdcard
2. Valentina:
    a. I2s for the receiver and coded
    b. Interfacing with the VGA
3. Both will work on the FIR filter module and FIR coefficients generator

Conclusion

In the end, we hope to have a functional digital signal processor that can be easily reprogrammed by the user to output the optimized audio signal for the specific speakers being used. Based on our experience in audio sampling and interfacing with the Artix 7 FPGA we expect a few areas to give us some trouble. Namely, ensuring that all our many clocks are properly synchronized.