John Westbrook

10 December 2023

System on Chip

Dr. Keith Schubert

Final Project Report

Introduction

Having a background in music, my idea for this project was to use the on-board mic of the NEXYS4 and display an audio input as a note on the seven-segment display. It seemed simple, as the process in my head shows in Figure 1. I ended up being very difficult. Taking data from the microphone, which was already in a digital format, and determining the frequency from it became an impossible task for the knowledge I have within the time frame I had. Eventually, a major shift in scope was needed to produce some level of finished project.

A diagram of software

Description automatically generated

Figure

Process

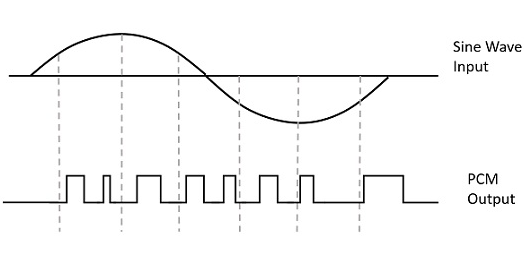
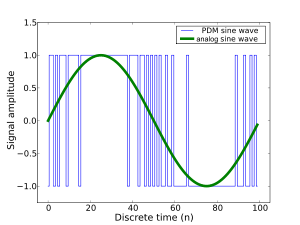
Coming into this project with a minimal understanding of converting audio formats required many hours of research just to understand what needed to be done. After many hours of scouring Google and help forums, I came to the understanding that a Fast Fourier Transform would be needed. The next issue was that I had almost no idea how an FFT worked. Xilinx has an FFT IP built-in to Vivado, which seemed to be the easiest way to accomplish an FFT. Figure 2 shows the updated process needed, after my research.

A diagram of a computer hardware system

Description automatically generated

Figure

The NEXYS4 microphone outputs Pulse Density Modulated data, which streams a one-bit output as 1s and 0s, with the density of the bits carrying the data. To make it easier to use the FFT, I first tried the PDM data to Pulse Code Modulated data, which was able to store a data size greater than one.



I was able to find one resource online about converting PDM to PCM, using a CIC Decimation filter. This was another added complication that I did not know how to use. After implementing what I could, I did achieve an output in testing, but I could not figure out if it was in a PCM format or not. I decided to move ahead with putting this data into an FFT, using an example I found involving BRAM.

This is where I discovered all the timing and control flags needed for the FFT IP. The IP user guide from Xilinx did not explain at a high enough level for me to understand, so I eventually turned to the Digilent forum for advice on how to accomplish what I wanted, and if it was possible for my current skill set. I was advised that obtaining frequency in the way I wanted would be very hard, so I shifted gears into making something that would work.

To take advantage of the frequency to note conversion I was intending, I decided to instead take an input from a keyboard and find the note of the frequency typed. The process was simple: use the Ps2 core to obtain the input, process it, and display it on the SSEG using the SSEG core. After a day or two of debugging, I was able to achieve a working product.

Conclusion

Sadly, this finished project is not as complicated as I wanted it to be. Because of the time crunch I ran into it was necessary to simplify what I was doing. I learned more about signal processing and what it’s like to try and accomplish something rarely anybody has done. Looking back, I should have tried forums from early on. The resources I was able to find weren’t high-level enough for me to understand, and a knowledgeable person would have been able to help. Asking if my idea was even reasonably doable would also have helped, because then I could have changed the project in something that was doable.

GitHub

[ELC4396\_Project](https://github.com/JhnWstbrk/ELC4396_Project/)