FPGA Implementation of the Fast Fourier Transform

Garrett Massman and Cory Walker

September 13, 2014

Introduction

In this project we implemented a fast Fourier transform algorithm on a Xilinx FPGA using VHDL. The primary motivation of our project was to digitally analyze musical signals, but the use cases for an FFT chip far surpass that specific use case.

As a very general overview, our completed device functions by sampling an analog signal for a set amount of time and storing the data into an input buffer in BRAM. Next, the signal is is processed using an FT controller and a complex ALU and stored in an output buffer. Finally, a microcontroller can then read the output buffer over a standard SPI interface. From there the processed frequency domain data can be sent to a computer for further processing or any other device.

Theory

To understand the discrete Fourier transform, one must first analyze its analogous continuous time form. This is written most simply as

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t}\, dt$$

which defines the transform of a continuous time signal $f(t)$. Without getting into too many details, it defines the signal as a combination of sinusoids, making it a very useful tool for real world applications.

The discrete Fourier transform (DFT) is simply a reduction of the continuous Fourier transform into a discrete sample space. In other words, if we let x_n represent a sampled version of the continuous time function $x(t)$ with a total of N samples, we can replace the integral with a summation over the series, as shown below.

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-j2\pi kn}{N}}$$

As with the continuous time case, this series gives us a glimpse into the component elements of our original signal. However, it is rather costly to

compute, requiring a time complexity of $O(N^2)$. For each value X_k, a series of values from $n = 0$ to $N - 1$ must be generated and summed, using up valuable computer resources. Thus, calculating the DFT in this way is very inefficient, so we instead turned to the fast Fourier transform.

In order to perform this operation more quickly, we utilized the Cooley-Tukey FFT algorithm. However, the one requirement is that our input is strictly a power of 2. That is because the algorithm works by recursively finding the FFT of smaller and smaller sample sizes of x_n, which arises from the fact that the transform itself is periodic. The transform function can be broken into the sum of its even and odd components,

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m+1)k}$$

This equation can further be simplified by making the substitutions

$$E_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)k} \qquad O_k = \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2}mk}$$

giving us

$$X_k = E_k + e^{-\frac{2\pi i}{N}k} O_k$$

The expression $e^{-\frac{2\pi i}{N}k}$ is commonly called the *twiddle factor*. Furthermore, because of the periodicity of the transform, we can calculate respective even and odd components simultaneously

$$E_k = E_{k+\frac{N}{2}} \qquad O_k = O_{k+\frac{N}{2}}$$

Stuff That's Already Been Done (change section name plz)

Motivation

System Overview

A critical task in building our FFT device was converting an arbitrary waveform into the SPI signal input that the FPGA expected. This requires the

use of a fast analog to digital converter. The Xilinx Spartan 3E FPGA does not come with an onboard ADC. Because of this, a large percentage of the work was actually electronics work done outside of the FPGA:

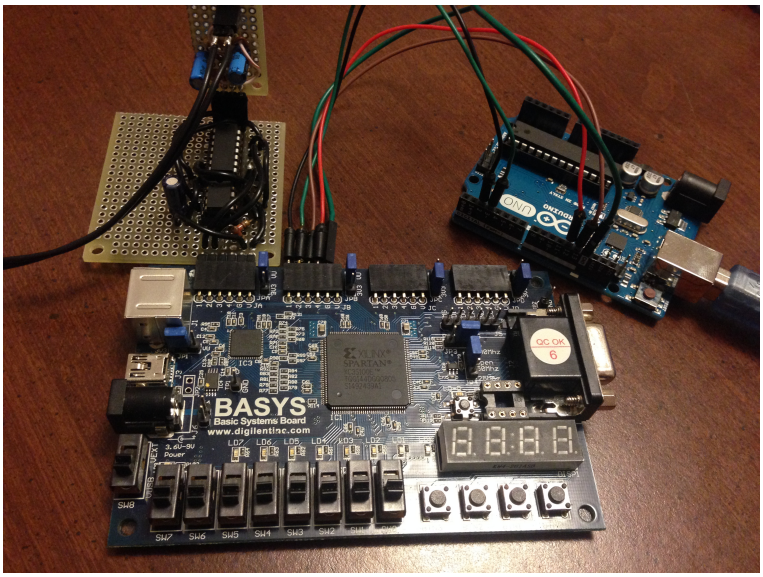

Figure 1: Full system including ADC PMOD and audio signal module.

These external components condition the signal and then send the digital signal into the FPGA at the right voltage level. We call these external components macro-components and we call the FPGA internal blocks micro-components.

Macro-Component Descriptions

Micro-Component Descriptions

The final block diagram of our device is as follows:

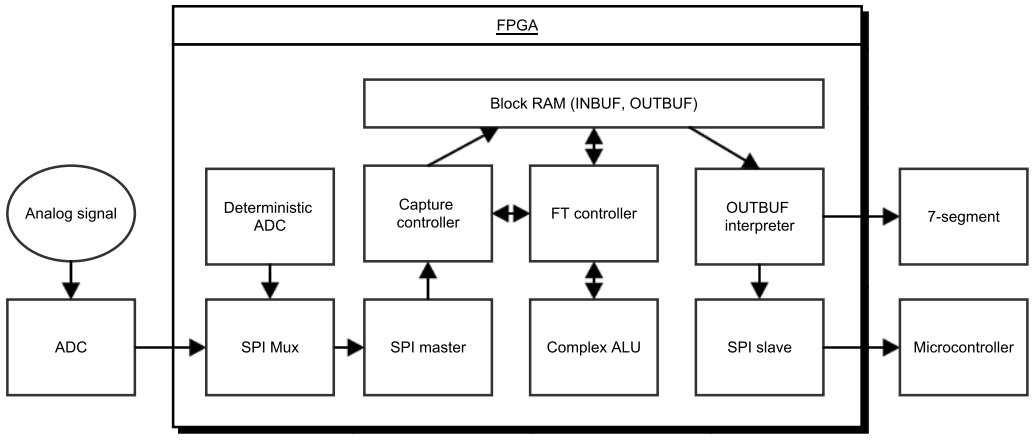

Figure 2: Block diagram of the FFT chip

All of the blocks inside the FPGA device are referred to as micro-components.

TODO: DESCRIPTION OF EACH COMPONENT HERE

Steps Taken

Simulation Results

Roadblocks

Future Improvements

Conclusion

Appendix

References

[1] Doin, Jonny. *SPI Master/Slave Interface.* OpenCores, 16 May 2011. Web. 13 Sept. 2014. <`http://opencores.org/project,spi_master_slave`>.

[2] Reynwar, Ben. *FFT on an FPGA.* FFT on an FPGA. N.p., n.d. Web. 13 Sept. 2014. <`http://www.reynwar.net/ben/docs/fft_dit/index.html`>.

[3] Roberts, Michael J. *Signals and Systems: Analysis Using Transform Methods and MATLAB.* New York: McGraw Hill, 2012. Print.

[4] Satoh, Keiichi, Jubee Tada, Kenta Yamaguchi, and Yasutaka Tamura. *Complex Multiplier Suited for FPGA Structure.* Computers and Communications (2008): 341-44. Web. 13 Sept. 2014.

[5] Wikipedia contributors. *Cooley–Tukey FFT algorithm.* Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 27 Jun. 2014. Web. 13 Sep. 2014.

[6] Wikipedia contributors. *Discrete Fourier transform.* Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 2 Sep. 2014. Web. 13 Sep. 2014.