Synopsis - Week 12 Integrated Project: The Nyquist Frequency and Fourier Series

Prof. Jordan C. Hanson

October 16, 2023

1 ADC and DAC Setup

Peripheral modules from Digilents are often called Pmods. Pmods allow peripheral devices to connect and interact with the PYNQ-Z1 system and give it special abilities. Recently in lecture we have learned how a flash ADC works. There are several varieties of ADC, and we happened to cover the flash ADC. A digital-to-analog converter (DAC) converts binary numbers in the form of high and low voltages to an analog voltage on the output. *In this lab activity, we will produce analog voltages via a DAC and feed them into an ADC.*

2 Create Code to Send and Plot a Fourier Series

2.1 Set up the system correctly

Set up a new Jupyter workbook with the following libraries and toos:

```
#Setup programmable logic, including Pmods for DAC and ADC, and other libraries
%matplotlib inline
import matplotlib.pyplot as plot
from time import sleep
import numpy as np
import matplotlib.pyplot as plt
from pynq.lib import Pmod_ADC, Pmod_DAC
from pynq.overlays.base import BaseOverlay
ol = BaseOverlay("base.bit")
dac = Pmod_DAC(ol.PMODB)
adc = Pmod_ADC(ol.PMODA)
```

The above code imports matplotlib, to graph the ADC output from the DAC input. We also program the firmware to handle PMODA controlling the ADC, while PMODB controls the DAC. They are connected physically by a wire. Numpy is imported for math utilities, including creating vectors of data.

2.2 Send a Sine Wave through the DAC to the ADC

Along the lines of the *Fouier Series and Sampling Tutorial*, create code that forms a Fourier series for a square wave. Loop over times as in the example code below, but do not just write the value of a sine function to the DAC. Instead, write the appropriate sum of several terms in the Fourier series.

```
#Transmission loop
sampling_frequency = 10.0 #Units: Hertz
delta_t = 1.0/sampling_frequency #Units: seconds
t_max = 10.0 #Units: seconds
times = np.arange(0, t_max, delta_t)
dc_offset = 1.0 #Units: volts
amplitude = 0.5 #Units: volts
frequency = 0.5 #Units: Hertz
samples = []
for t in times:
```

```
#Don't do this (only one sine term):
    #dac.write(amplitude*np.sin(2.0*3.14159*frequency*t)+dc_offset)
    #Instead make it the sum of several sine or cosine terms
    #as given by the Fourier series
    sleep(delta_t)
    samples.append(adc.read())
```

2.3 Plotting the Results

In the following code, a plot is created from the pyplot module from the matplotlib library. The list of times defined in the last section is plotted on the x-axis, and the ADC voltages are plotted on the y-axis. The title of the plot should display the sampling frequency and sine wave frequency.

```
#Plotting section
plot.plot(times,samples,'-o')
plotTitle = "Frequency: "+str(frequency)+" Hz, Sampling Frequency: "+str(sampling_frequency)+" Hz"
plot.title(plotTitle)
plot.xlabel('Time (seconds)')
plot.ylabel('Amplitude (Volts)')
plot.grid(True, which='both')
plot.legend(loc='upper left')
plot.axis([0, t_max, 0, 2.0])
plot.show()
```

2.4 Questions to be Answered

1. (a) What is the effect of sending the Fourier series over the line from the DAC to the ADC? (b) What kinds of distortions or oddities occur that make the ADC signal different from a square wave?

2. (a) What happens to your observations if you include a higher number of terms in the Fourier series? (b) While sending a relatively higher number of terms over the line, reduce the sampling frequency. What do you notice?