ECE 350 Final Project

Januario Carreiro and John Yaras

December 4, 2019

1 Project Design and Specifications

For the final project, we decided to build and implement an `XY-Plotter`, which is a device that can guide a pen along two axes in order to create an image on a piece of paper. For the hardware, we settled on the <KIT NAME> as a starting point. This kit included the mechanisms necessary to draw on two axes, and we would go on to add a mechanism to lift and lower our writing instrument.

Part	Quantity
Acrylic Shelf	8
Aluminum Beams	4
Aluminum Guideway	1
Ball Bearing	13
Elastic Coupling	1
Flat Washer	12
Hex Nut	20
Hex Screw	28
Phillips Screw	36
Plastic Washer	20
Servo Motor	1
Stepper Motor	2
T Nut	8

Table 1: Parts List

2 Input and Output

To give instructions to the FPGA, a system similar to the one used by Logo's turtle graphics module was implemented:

$$\boxed{\text{OP}}\underbrace{\boxed{\text{H2}}\boxed{\text{H1}}\boxed{\text{H0}}}_{\text{Hex Digits}}$$

Table 2: Instruction Format

Here, `OP` is an ascii character code for either `F, B, L` or `R`. If `OP` is `F` or `B`, then the digits that follow are interpreted as a number of rotations of the stepper motor. If `OP` is instead `L` or `R`, then the digits that follow are interpreted as degrees. This system where the pen moves with commands relative to its own position was chosen for two reasons: (1) the stepper motors and the tracks slowly decalibrate as they move, so a coordinate system would provide wonky images, and (2) turtle geometry has a short learning curve: anyone can easily control `XY-Plotter` with minimal instruction—after all, Logo was designed as an educational programming language.

3 Changes to Processor

No changes to the processor were made. The only addition to the processor that would have been useful would have been the capability to do floating point math. However, adding support for floating points was deemed to be an irresponsible use of time given the time constraints, so we used a system of lookup tables (with significant multipliers) inserted into the DMEM in order to get around the need for floating point math.

4 Challenges Faced

5 Testing

The first thing that was tested was the stepper motor. Since neither of us had ever worked with stepper motors before, we quickly prototyped a circuit using a breadboard, wrote some Arduino code, and messed around with the Stepper motor until we could understand how to communicate with it effectively.

After making sure the stepper motors worked on their own, we built the base of the `XY-Plotter` and plopped the stepper motors in position. Then, using the Arduino code and circuit from before, we tested the stepper motors on the base to make sure that the tracks were moving properly.

Once we were sure the stepper motors and the tracks were working as expected, we moved on to trying to communicate with the stepper motors using the FPGA. At first, we wanted to communicate directly from the FPGA to the stepper motors, but we soon realized that a better design would be using the Arduino as a buffer between the FPGA and the stepper motors. This way, we could have the FPGA working to decode and interpret instructions as soon as they were received and send the decoded instructions to the Arduino without any need for stalling. The Arduino's built-in serial buffer could then be used as a queue of instructions for the the Arduino to convert to stepper motor movement without much hassle.

6 Programs/Code

```
import math
...
f = open("dmem.mif","w+")
for i in range(360):
  sinBin="{0:032b}".format(round(multAngle*math.sin(radi*i))&0xffffffff)
  f.write("%d : %s ;\n"%(offsetSin+i,sinBin))
...
for i in range(360):
  if round(fm * math.sin(radi * i)) == 0 : # cannot divide by zero
    fx="{0:032b}".format(abs(round(fm*math.sin(radi*i))))
  else :
    fx="{0:032b}".format(round(fc/(2*abs(fm*math.sin(radi*i)))))
  f.write("%d : %s ;\n"%(offsetFx+i,fx))
...
f.write("END;\n")
f.close()
```

Listing 1: Python Code for generating Lookup Tables in dmem.mif

7 Possible Improvements/Features

8 Images

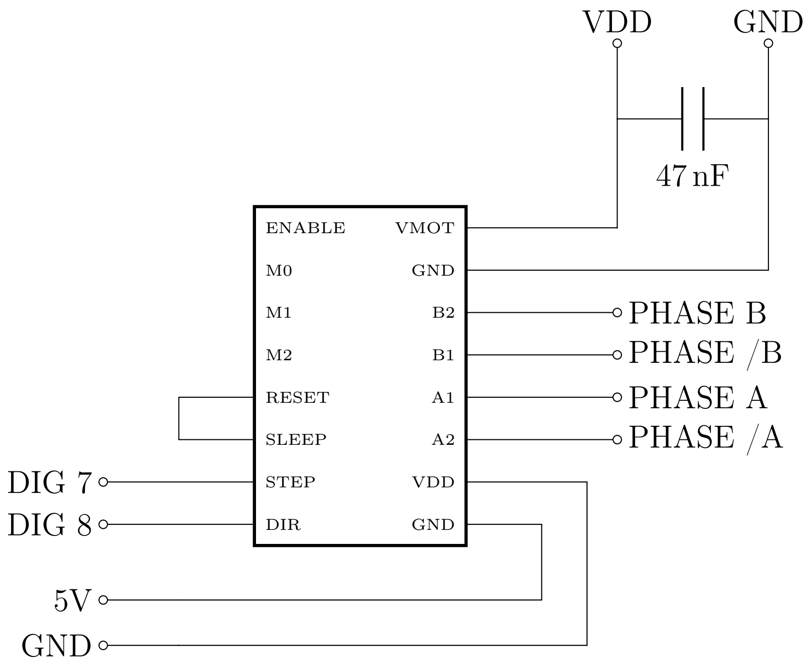

Figure 1: Circuit Diagram for Stepper Motor Drivers

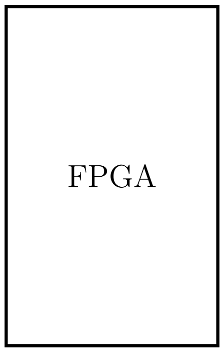

Figure 2: Circuit Diagram for FPGA Outputs