Anthony Lin
Matthew Morscher
Dr. Song
16 February 2017

ECE 530 Final Report

Introduction

For our project, we implemented an analog-to-digital converter (ADC) on the Pynq-Z1 board. We took an input from a composite video source (GoPro) as seen in *Figure 1* and sampled it at 1 MSPS (million samples per second). After collecting the data, we used a Python program (see Appendix – Python Code) to develop code to convert it to a grayscale digital image. We initially planned to display the image real-time through the HDMI port, but were unable to get the port working on time.

Theory

In theory, we would use the XADC (ADC unit on the Zynq chip) on the Pynq board to convert a composite video signal to a full-color digital picture. Unfortunately, the XADC built into the Pynq board does not have a fast-enough sampling rate. The XADC only has a max sampling rate of 1 MSPS, but to collect color samples and make the quality the best possible, we would need about 500 MSPS. As you can see in *Figure 4*, there is a burst immediately following the Horizontal Syncs (HSyncs), which provides the frequency that the color information is encoded for that frame. The frequency needed to retrieve that information was not something we were able to obtain. Also, ideally if HDMI was working, we would have a live stream of the GoPro feed being converted from analog to digital.

Design/Analysis

You can see in our block design in *Figure 2*. This block design is for the XADC on the Pynq board and the theoretical output of HDMI. The top half, which is the working implementation, is for the XADC. You can see our hardware setup in *Figure 1*. Here, we connected a GoPro to a breadboard using the USB connection to obtain the composite video data. We then connected the breadboard to the analog ports of the Pynq board. After doing this, we were successfully able to run our hardware and code to obtain the digital data of the analog picture.

The sampling rate for the XADC was the main limiting factor. Due to this, our final digital picture quality was quite poor, which you can see in *Figure 8*. If we had a higher sampling rate of around 8 MSPS, we could obtain a complete grayscale image with good quality. If we had an even higher sampling rate of around 500 MSPS, we could obtain a complete color image with good quality. Unfortunately, with the 1 MSPS ADC we could only capture around 80 pixels per line (column pixels).

Testing, Measurements, Procedure, and Results

To start, we began by connecting the GoPro to an oscilloscope to view the composite video analog data. Next, we created the ADC hardware and software (see *Figure 2*). After doing that, we connected the GoPro to the Pynq board analog ports (see *Figure 1*). We then collected 65536 samples of data from the ADC. We then took that data and plotted it using Python. We analyzed these plots (see *Figure 5, 6, and 7*) and determined where the syncs occur. Based upon this, we wrote another program in Python that could decode a digital picture from this data (see *Figure 8*).

In the end, we could sample at 80 columns per Hsync. Due to this low number, the quality of our final digital picture (see *Figure 8*) was not very good. As mentioned before, this is due to the ADC not being able to sample at a rate of more than 1 MSPS. To maintain the original aspect ratio, we scaled the horizontal axis.

Conclusion

Overall, we achieved most of our goals for this project however we were unable to complete the HDMI portion of our project. We feel this is due to the limited documentation on HDMI on the Pynq board, as it is a new board and the rather short time frame to complete the project. Otherwise, we completed all our initial goals we decided upon in the beginning. Even though we did not get HDMI function, we still believe our project was a success.

References

- http://web.mit.edu/6.111/www/f2008/handouts/L12.pdf
- https://github.com/Xilinx/PYNQ
- http://www.pynq.io/board.html

Appendix – Pictures

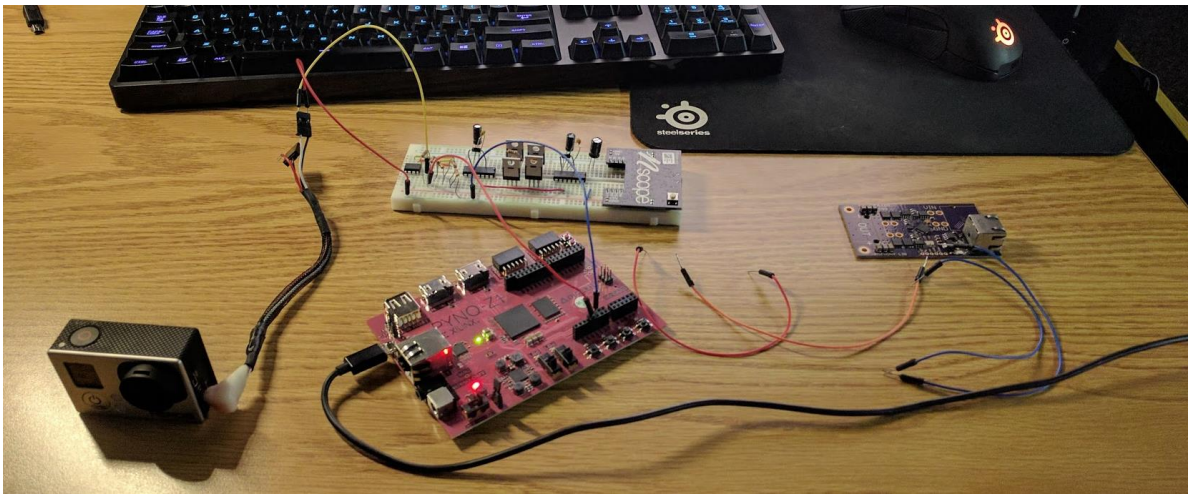

Figure 1 Hardware Setup

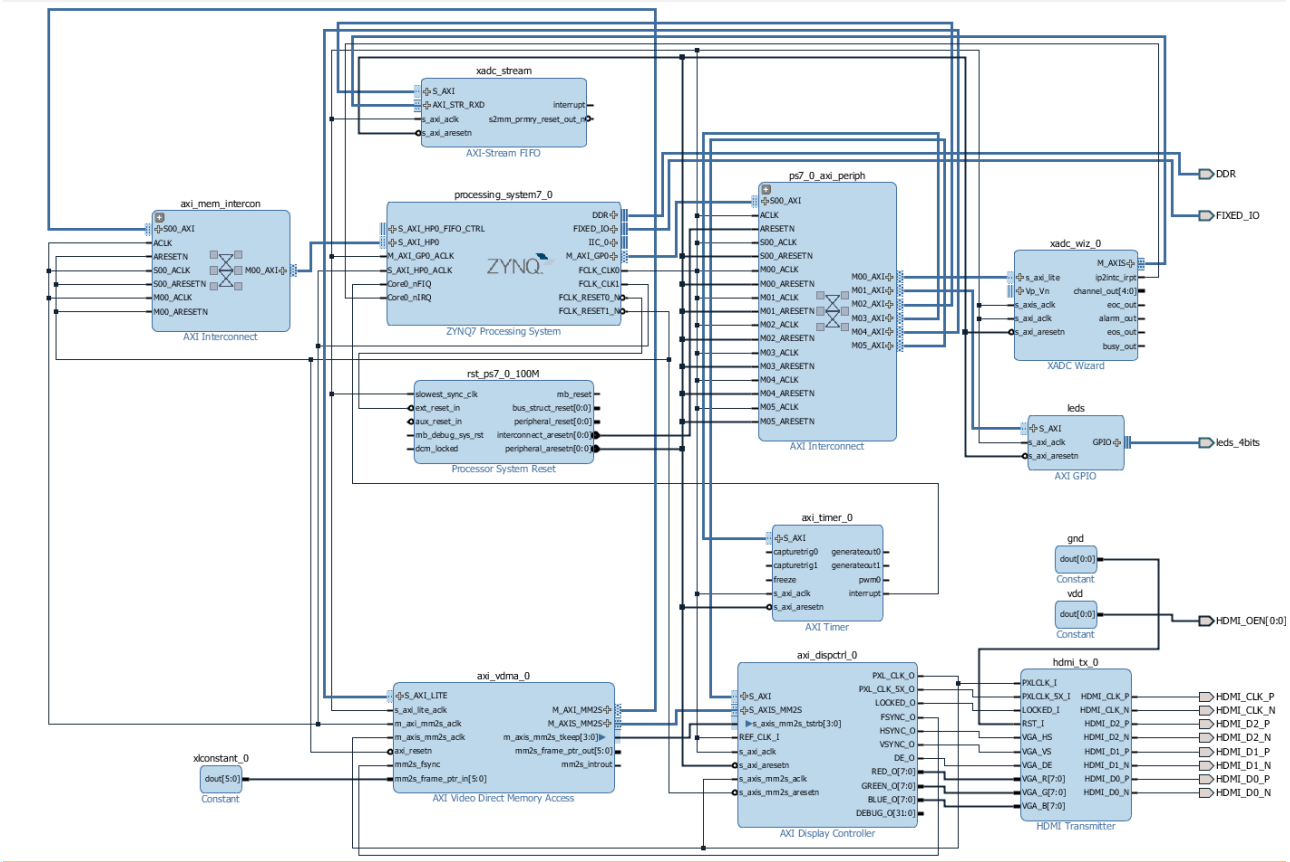

Figure 2 Block Diagram

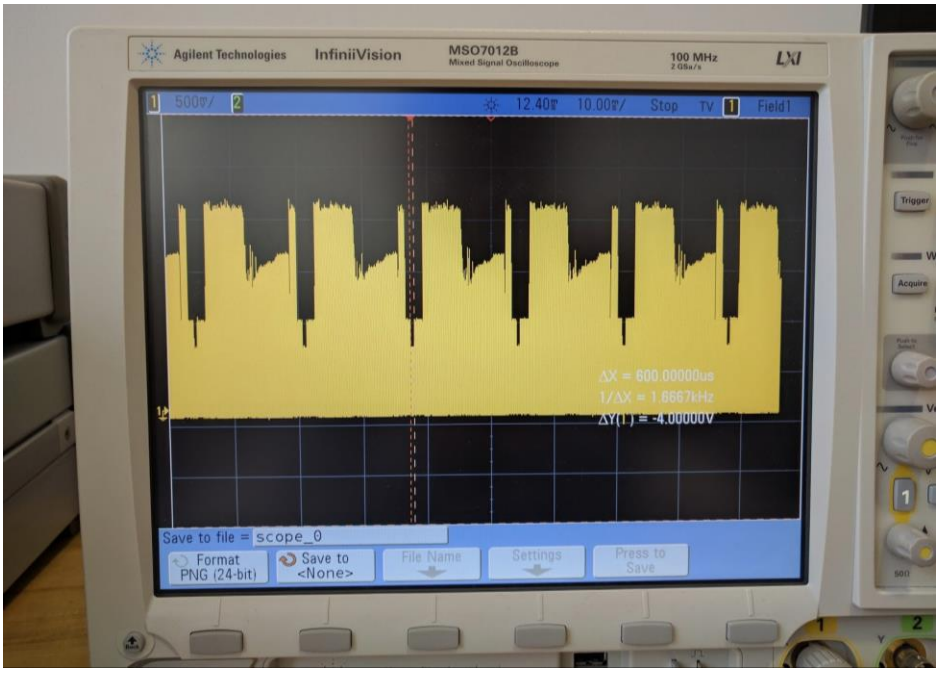

Figure 3 Oscilloscope Capture of D115 Room From GoPro

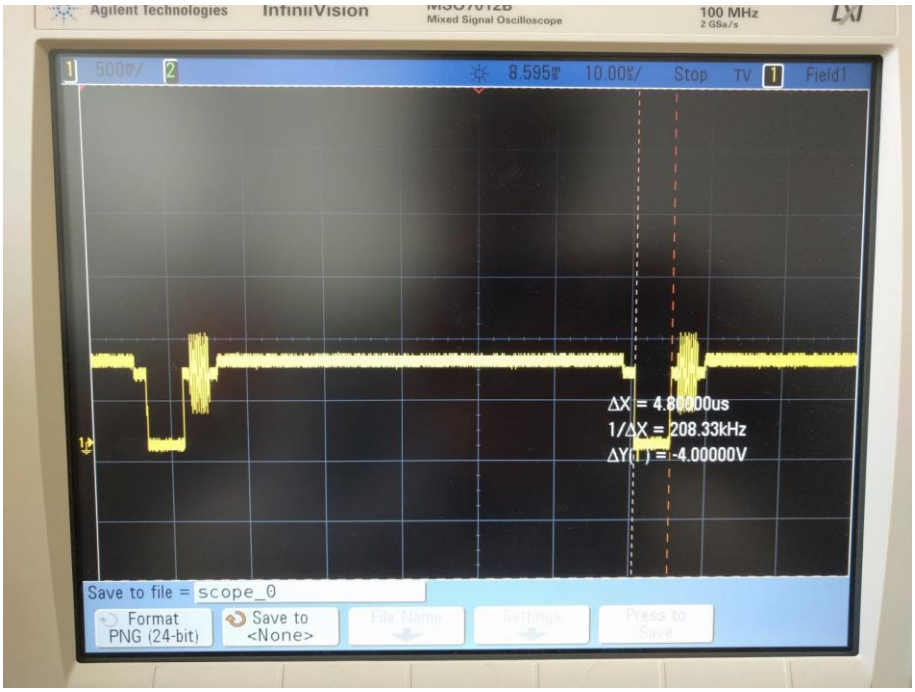

Figure 4 Oscilloscope from Figure 3 Zoomed In

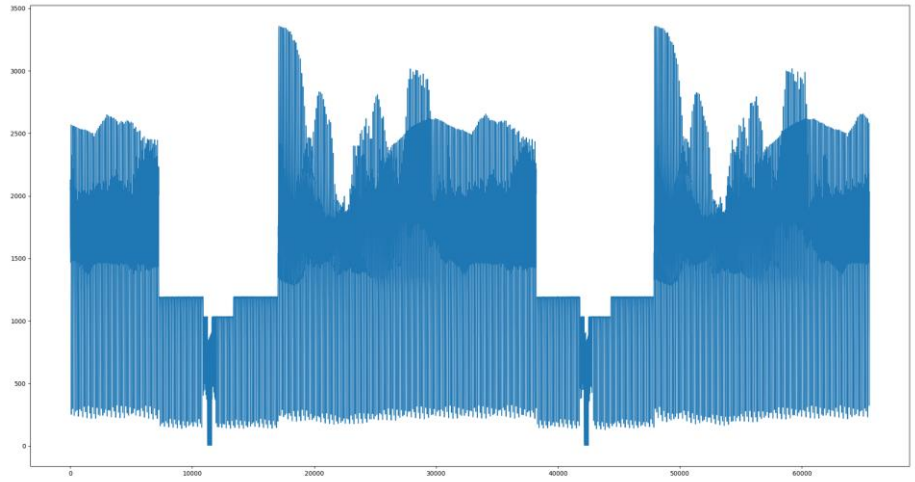

Figure 5 Plot of Digital Data of Picture with Anthony and Matthew

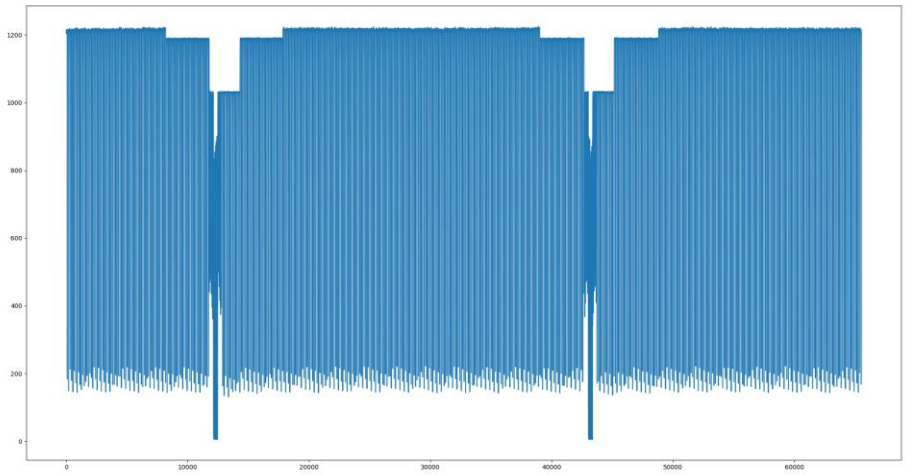

Figure 6 Plot of Digital Data of Completely Black Picture

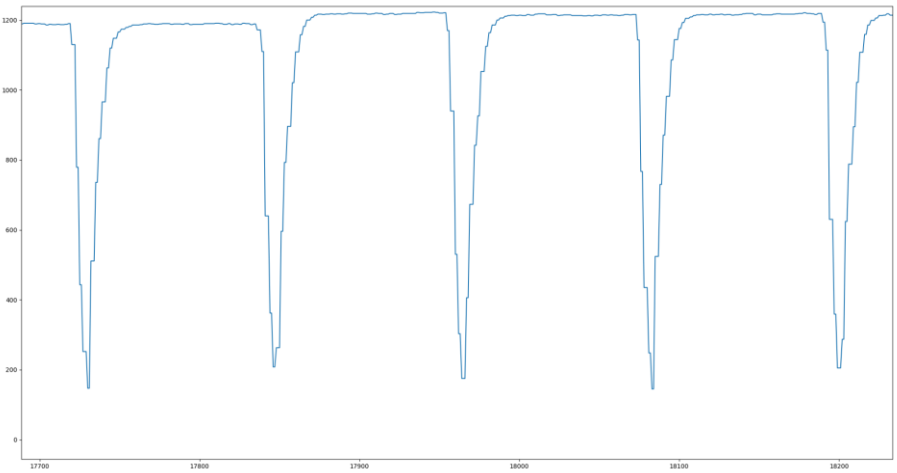

Figure 7 Plot Zoomed in of Black Level of Image

Figure 8 Digital Picture of Anthony and Matthew

Appendix – Python Code

```
import numpy as np
from PIL import Image

BLACK = 0
WHITE = 255
BLACKLEVELLOW = 1180
BLACKLEVELHIGH = 1200
WHITELEVELLOW = 2200
COLUMNS = 80
VSYNCVALUE = 1100
SCALE = 4

with open("sampledatafulleframe.txt") as f:
   data = f.read()

def mapVal(valueIn, baseMin, baseMax, limitMin, limitMax):
        if(valueIn > baseMax):
                return limitMax
        elif(valueIn < baseMin):
                return limitMin
        return (((limitMax - limitMin) * (valueIn - baseMin) / (baseMax - baseMin)) + limitMin)


data = data.split('\n')
values = [];

array = np.zeros((500,COLUMNS*SCALE), np.uint8)
hcount = 0
vcount = 0

for row in data:
   values.append(int(row))

startOfFrameMaybe = False
startOfFrameYes = False
endOfFrame = False
startOfFrameCount = 0
endOfFrameCount = 0
lastValue = 0
value = 0
i = 0

while(i < len(data)):
        value = int(data[i])
        i += 1
        # Check for Vsync
        if (value <= BLACKLEVELLOW):
                startOfFrameCount += 1
                if (startOfFrameCount > 100):
                        if (startOfFrameYes == False):
                                startOfFrameMaybe = True
                        else:
```

```
                                        endOfFrame = True
                                        startOfFrameYes = False
                # Check if frame should start
                elif (value > BLACKLEVELLOW and value < BLACKLEVELHIGH and startOfFrameMaybe == True):
                        startOfFrameCount = 0
                        startOfFrameYes = True
                        print('Start of frame')
                        break
                # Reset count otherwise
                else:
                        startOfFrameCount = 0

while (i < len(values)):
        value = values[i]
        if (value < VSYNCVALUE):
                endOfFrameCount += 1
                if (endOfFrameCount > 50):
                        print('End of frame')
                        break
        else:
                endOfFrameCount = 0
        if (value < BLACKLEVELLOW and lastValue > BLACKLEVELLOW):
                vcount = 0
                hcount += 1
                lastValue = value
        elif (value > BLACKLEVELLOW):
                if (vcount < COLUMNS):
                        for diff in range(SCALE):
                                array[hcount][SCALE*vcount+diff] = mapVal(value, BLACKLEVELLOW, 3000, BLACK,
WHITE)
                        vcount += 1
                        lastValue = value

        i += 1

im = Image.fromarray(array)
im.save("result - stretched.png")
img = Image.open('result.png')
img.show()
```