FPGA based
Audio Effect System

Submitted by: Dean Devereaux
Student number: R00116198
Date: 5th January 2018

Supervisor: Paddy Collins
Assessor: John Horan

Department of Electrical & Electronic Engineering

Declaration

I declare that this submission is my own work and writing.

Any text, diagrams or other material copied from other sources (including, but not limited to, books, journals and the internet) have been clearly acknowledged and referenced as such in the text by the use of 'quotation marks' (or indented *italics* for longer quotations) followed by the author's name and date [e.g. (Byrne, 2008)] either in the text or in a footnote/endnote. These details are then confirmed by a fuller reference in the bibliography section.

I acknowledge that I understand the concept of plagiarism. I have studied CIT's guide to referencing (Section 3 Referencing at http://www.mycit.ie/libraryguide/) and I have read the section on Plagiarism and Cheating (pages 46 – 49) in CIT's Student Regulations handbook (available at http://www.mycit.ie/). I understand that only assignments which are free of plagiarism will be awarded marks. By submitting this assignment I agree to these terms. I further understand that CIT has a plagiarism policy which can lead to the suspension or permanent expulsion of students in serious cases.

I understand that my project report will be evaluated by the Safe Assignment/TurnItIn tool in Blackboard to detect instances of plagiarism.

Signed:

_____ Date: _____

Table of Contents

List of Figures

List of Tables

Abstract

For this project the objective will be to apply an effect to an audio signal through the use of a field programmable gate array on the basys3 development board which will be programmed using Verilog language on Vivado design suite. This audio signal will go through the development board and produce an altered signal outputted to a speaker. Three effects will be created and applied to the signal using slide switches to choose the desired effect.

In this report the research of the project will be explained which includes software and hardware. The system design which show the overall solution and explain the XADC. Testing of the tutorial programs through Vivado and what will remain to be done in semester two.

Acknowledgements

I would like to express my sincere thanks and great gratitude to my supervisor Paddy Collins

Dean Devereaux

December 2017

1. Introduction

This report will be going through the research and work done to date in semester one of the project at hand. The areas at which this project will be focusing on are digital system design and digital signal processing. The end product of this project should show that the Artix-7 FPGA can be used to create an audio effect system similar to a pedal board used by musicians. This idea has already been done by a music industry company by the name of Antelope Audio which sell a Hardware FPGA Audio effects commercially. ("Antelope Audio Hardware-based FX," n.d.)

The main goal of this project is to program the basys3 development board to digitize any analogue audio sent through the JXADC Pmod and apply any three of the following effects

- Delay
- Non-Linear Processing
- Spatial

The digitized audio will then be sent out through a second Pmod port and converted back to analogue audio using the external Pmod I2S which can then be connected to a speaker to output the audio with the applied effect. For the three effects three slide switches from the basys3 will be used to control which effects gets applied to the audio passing through.

The plan was to split the work by focusing on the XADC and DAC for semester one to get an audio signal to pass through the development board without and altercations. Then in semester two the effects would be researched and applied to the audio signal from the programming of the FPGA.

2. Literature Review/Theoretical Background

a. FPGA

Field Programmable Gate Arrays originate around the late 80's/early 90's and took on from PROM and PLDs which could be field programmable but had the disadvantage of having hardwired logic. FPGAs is an IC chip and is intended to be reprogrammable after manufacturing. To do this it uses a matrix of logic blocks known as CLB's short for configurable logic blocks. These blocks are very flexible which allows for interconnection between each CLB. FPGAs similar to ASICs uses hardware description language (HDL) which can be between the most common, VHDL or Verilog language. ("What is an FPGA," n.d.) ("Field-Programmable-Gate-Array," 2017)

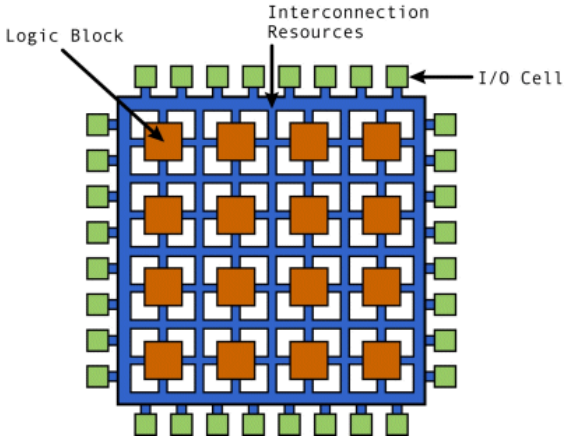

Figure 1 FPGA Architecture ("All About FPGAs," 2006)

b. Basys 3 Development Board

The Basys3 development board was produced by Digilent and is the board which was available for this project. This development board uses the Xilinx FPGA Artix-7 35T which is low cost and can be used for high performance logic. This board uses a 100MHz clock and has 1,800Kbits of RAM with 90 DSP slices. Due to the basys boards change from Spartan-3E to Artix-7 FPGA the basys 3 comes with some useful peripherals including

- 16 Slide Switches
- 16 LEDs
- 5 Push Buttons
- 12-bit VGA Connector
- 4 digit 7-segment displays
- 4 Pmod ports
- Internal XADC

Figure 2 Basys 3 development board front view

("Basys 3," n.d., p. 3)

For this project the Pmods, internal A/D converter and switches will be the primary parts used to achieve an audio effects system. The LEDs and push buttons will be used for secondary functions in regards to this ("Basys 3 Reference Manual," 2016)

Figure 3 shows the many ways the Artix-7 can be programmed. For this project the FPGA was programmed via USB-JTAG which allowed a project from Vivado to be sent to the board once power was on and the board was connected to the PC. This was done through the port 'J4'. The configuration data from the Vivado project would be stored in bitstream files which are needed for the logic functions and circuit connections of the board. These file are created within Vivado before sending the program to the board and can take a bit of time to send unless they are compressed before programming the device. When programmed the bitstream files are stored within the SRAM memory of the Artix-7. ("Basys 3 Reference Manual," 2016)

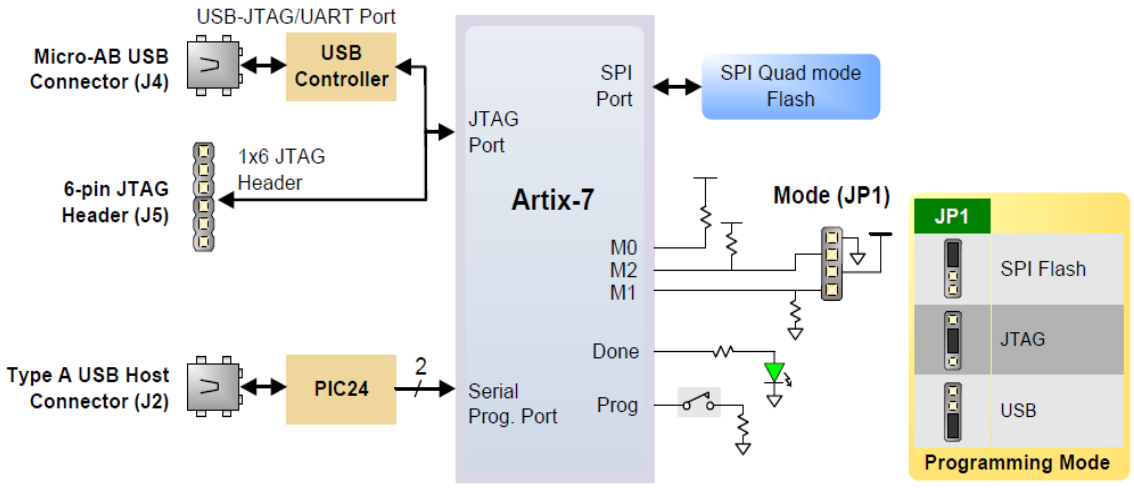

Figure 3 Basys3 configuration options ("Basys 3 Reference Manual," 2016)

The development board has 32Mbit serial flash device which is connected to the Artix-7 with an SPI bus shown in figure 4. The Artix-7 only uses about 2Mbytes for configuration files which leaves almost 48% remaining for user data on the board. ("Basys 3 Reference Manual," 2016)

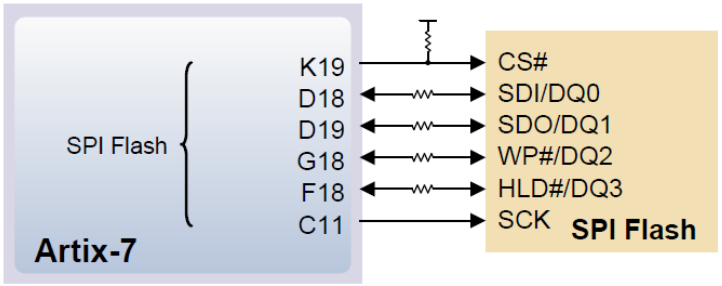

Figure 4 External Memory ("Basys 3 Reference Manual," 2016)

i. XADC

The internal XADC found on the basys3 is what will be used for converting analogue audio to digital audio. The typical conversion rate for audio found today would be 16-bits to 24-bits at a sample rate of 44.1 kHz.

The internal A/D converter has a dual channel which can convert 12-bits max each and a sample rate of up to 1MHz which is more than enough for the objective. The XADC which is part of the Artix-7 chip is connected to the Pmod JXADC through an auxiliary analog input pin and this will allow the XADC to convert signals received by the JXADC ("Artix-7 Datasheet'," n.d.)("Basys 3 Reference Manual," 2016)

c. Pmod I2S

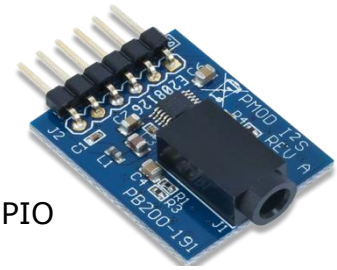

Figure 5 Pmod I2S

The I2S will be used with the basys 3 to output the altered audio. This device can be connected with a Pmod port of the development board and convert the digitized audio back to analogue. This external device is perfect as it is capable of performing 24-bit D/A conversion and uses a GPIO interface with the 6-pins to communicate with the host board. A beneficial feature to the pmod is the headphone jack which can support any audio data format. This is also a good feature as it allows for the pmod to be connected directly to a speaker without any DIY procedures ("Pmod I2S," 2016)

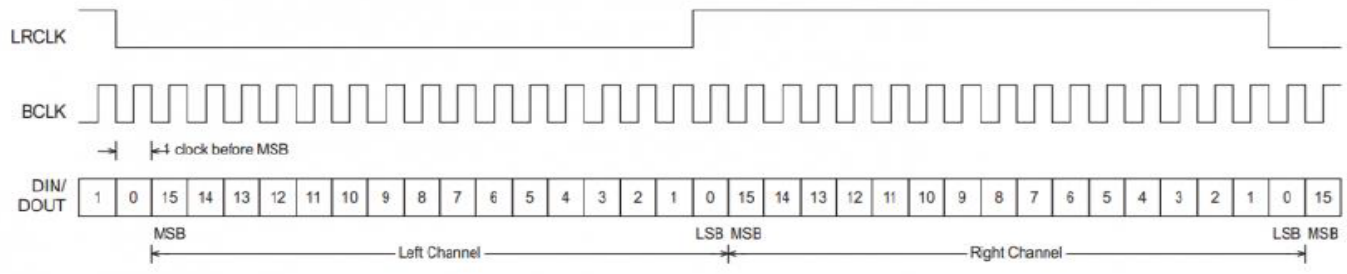

Figure 6 I2S Timing Diagram ("Pmod I2S," 2016)

The I2S has 6 pins three of which are clock signals. Pin1 connects with the Master Clock (MCLK) of the host device. This synchronises both devices allowing no delay to the output signal. Pin2 is the Left-Right Clock (LRCK) and this takes the sample rate used on the connected host device and applies it to the signal being output through the I2S. Pin3 then which is the Serial Clock (SCK) controls the bit rate of the I2S. This rate is determined by the MCLK and LRCK. ("Pmod I2S," 2016)

Pin	Signal	Description
1	MCLK	Master Clock
2	LRCK	Left-right Clock
3	SCK	Serial Clock
4	SDIN	Serial Data input
5	GND	Power Supply Ground
6	VCC	Positive Power Supply

Table 1 Pinout Description

("Pmod I2S," 2016)

d. Vivado Design Suite

Vivado is a software application created by Xilinx and is the software which will be used for programming the basys 3 development board. The HDL used with vivado are VHDL and Verilog. Vivado has many tutorials on the Digilent website including how to set up a project and a few of these tutorials are based on the Basys 3 development board which include code and setup of projects. An I/O and XADC demo for the basys3 was discovered on the website and used as a tester and building block for the internal A/D configuration. ("Basys 3 General I/O Demo," n.d., "Basys 3 XADC Demo," n.d., "Getting Started With Vivado," n.d.)

To date vivado is currently on version 2017.4. For this project the version used was 2017.1 due to it being the latest full software version to be downloadable. When using vivado a constraint file is needed to work on the basys 3 in order to identify which devices on the development board. Digilent will have the constraint file titled 'Basys3-Master' which can be used for all tests and simulations of the project. ("Getting Started With Vivado," n.d.)

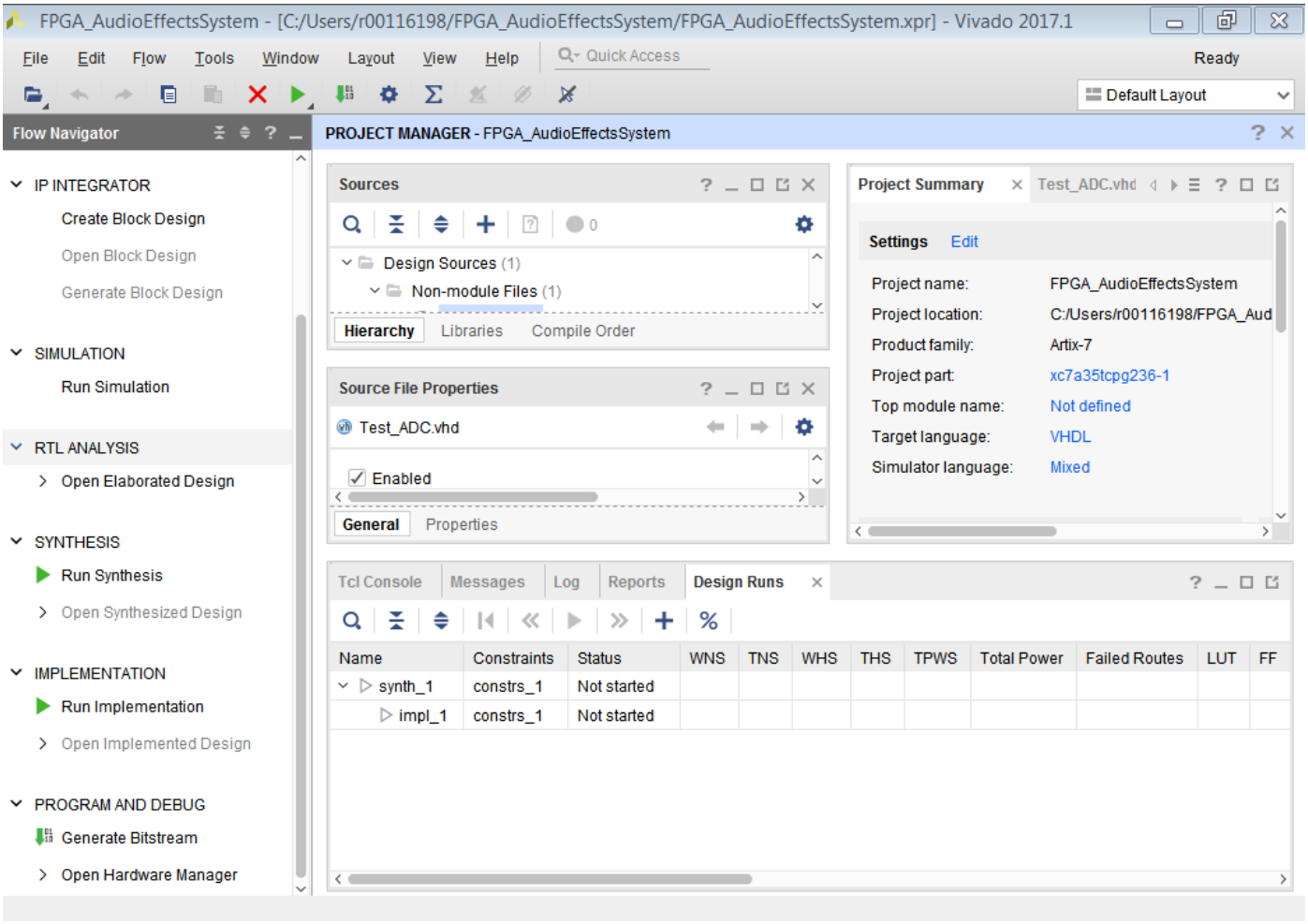

Figure 7 Vivado User Interface

3. System Design

a. Overview

FPGA development board which inputs an analogue audio signal and converts to a digitized audio signal using an internal ADC. The board will then apply any three effects being controlled by slide switches before the signal is output to an external DAC and through a headphone jack to a speaker.

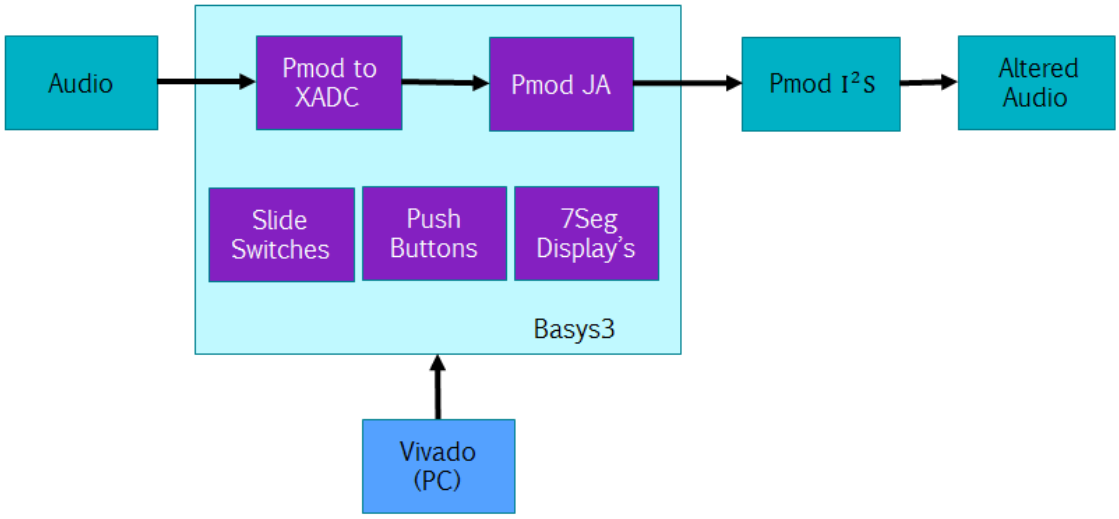

Figure 8 Audio Effects System Block Diagram

b. XADC

In order to apply an effect to the audio it must be first digitized. The basys 3 board has an internal XADC which is part of the Artix-7 chip and must be configured using the DRP (Dynamic Reconfiguration Port). The XADC on the board is capable of converting an analog audio signal into a 12-bit digital signal. Audio is between 20Hz-20kHz and to sample this the sample rate must be at least twice the size of the signal. Typically audio is sampled today is at 44.1kHz and this was the sample rate preferred. The XADC can go as low as 154kHz which oversamples more than three times the average but reduces the noise found in the signal.

4. Implementation and Tests

Vivado design suite was used throughout the semester to program the Basys3. To test the board I/O a general I/O demo from Digilent was used to guarantee that the board was functioning correctly. This demo turned on the LEDs using the slide switches. Counted from zero to nine on the four seven segment displays which would reset when a push button was pressed and display a multitude of colours through the VGA.

Also used to test the Basys3 was the XADC demo which used the XADC to convert the input through JXADC. This demo shows the digital result from JXADC on the four seven segment display and on the 16 LEDs which would turn on/off one after another based on the voltage being passed through. When testing this program on the basys3 a function generator was used to apply a signal to the JXADC.

The XADC demo was also used to understand the use of Vivado and how to properly get the XADC working due to the lack of knowledge of both software and hardware. The program was altered to output through Pmod JA and have a minimum sample rate of 154 kHz. For both demos Verilog language was used.

5. Semester Two

For semester two the I2S will have to be set up with the Basys3 to output the audio signal. Once this is finalized then the effects can be must be researched and can be applied to the programming for alteration of the digitized audio.

The slide switches will also have to be programmed for choosing an effect to use and push buttons for incrementing or decrementing the effects influence on the digitized audio.

6. Conclusion

The objective of the project was to create an audio effects system using an FPGA. This semester the goal was to digitize an audio signal using the basys3 development board and output that signal through an external DAC to a speaker without any altercations. What was accomplished was the research for hardware and software. A signal from a function generator was sent to the board and displayed on the four seven segment display using the XADC Demo tutorial. Near the end of the semester the demo program was altered to output through Pmod JA and a signal was getting through but it was too late to test with the I2S Pmod.

A weaknesses found was the lack of knowledge about the Viavado design suite and how to work it correctly with the basys3. This caused a lot of delays and back logged the overall project plan for this semester however a lot has been learnt about the development board and software due to research which should help with speeding things up in the next semester. Another weakness is the ability to send an audio signal through the JXADC Pmod with no actually input Pmod available with an audio auxiliary cable. This will have to be a DIY job for semester two. Semester one objectives could be finished off in January before semester two starts in February. This would be ideal to get the project plan back on track and take advantage of the free time.

References

All About FPGAs, 2006.
https://reference.digilentinc.com/reference/programmable-logic/basys-3/start
[Accessed 28th December 2017]

Antelope Audio Hardware-based FX, n.d.
https://en.antelopeaudio.com/hardware-based-fpga-effects/
[Accessed 10th October 2017]

Artix-7 Datasheet', n.d.
https://www.xilinx.com/support/documentation/data_sheets/ds181_Artix_7_Data_Sheet.pdf
[Accessed 10th October 2017]

Basys 3, n.d.
https://reference.digilentinc.com/reference/programmable-logic/basys-3/start
[Accessed 24th September 2017]

Basys 3 General I/O Demo, n.d.
https://reference.digilentinc.com/learn/programmable-logic/tutorials/basys-3-general-io/start
[Accessed 24th September 2017

Basys 3 Reference Manual, 2016.
https://reference.digilentinc.com/basys3/refmanual
[Accessed 24th September 2017

Basys 3 XADC Demo, n.d.
https://reference.digilentinc.com/learn/programmable-logic/tutorials/basys-3-xadc/start
[Accessed 24th September 2017

Field-Programmable-Gate-Array, 2017.
https://en.wikipedia.org/wiki/Field-programmable_gate_array#Architecture
[Accessed 28th December 2017

Getting Started With Vivado, n.d.
https://reference.digilentinc.com/vivado/getting_started/start
[Accessed 25th October 2017]

Pmod I2S, 2016.
https://reference.digilentinc.com/reference/pmod/pmodi2s/reference-manual
[Accessed 6th October 2017]

What is an FPGA, n.d.
https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html
[Accessed 28th December 2017]

Appendix A: XADC Demo

Initialise

```
module XADCdemo(
    input CLK100MHZ,
    input vauxp6,
    input vauxn6,
    input vauxp7,
    input vauxn7,
    input vauxp15,
    input vauxn15,
    input vauxp14,
    input vauxn14,
    input [1:0] sw,              --Switch 0 and 1
    output reg [15:0] LED,     -- 16 LED's
    output [3:0] an,             --AD6,AD7,AD14,AD15, Show voltage difference
    output dp,
    output [6:0] seg             --SegDisplay );

    wire enable;
    wire ready;
    wire [15:0] data;
    reg [6:0] Address_in;
    reg [32:0] decimal;
    reg [3:0] dig0;
    reg [3:0] dig1;
    reg [3:0] dig2;
    reg [3:0] dig3;
```

```
    reg [3:0] dig4;
    reg [3:0] dig5;
    reg [3:0] dig6;
    //xadc instantiation connect the eoc_out .den_in to get continuous conversion
    xadc_wiz_0  XLXI_7 (.daddr_in(Address_in), //addresses can be found in the artix 7 XADC user guide DRP register space
                        .dclk_in(CLK100MHZ),
                        .den_in(enable),
                        .di_in(),
                        .dwe_in(),
                        .busy_out(),
                        .vauxp6(vauxp6),
                        .vauxn6(vauxn6),
                        .vauxp7(vauxp7),
                        .vauxn7(vauxn7),
                        .vauxp14(vauxp14),
                        .vauxn14(vauxn14),
                        .vauxp15(vauxp15),
                        .vauxn15(vauxn15),
                        .vn_in(),
                        .vp_in(),
                        .alarm_out(),
                        .do_out(data),
                        //.reset_in(),
                        .eoc_out(enable),
                        .channel_out(),
                        .drdy_out(ready));
```

LEDs setup

```
//led visual dmm
always @( posedge(CLK100MHZ))
begin
  if(ready == 1'b1)
  begin
    case (data[15:12])
      1:  LED <= 16'b11;
      2:  LED <= 16'b111;
      3:  LED <= 16'b1111;
      4:  LED <= 16'b11111;
      5:  LED <= 16'b111111;
      6:  LED <= 16'b1111111;
      7:  LED <= 16'b11111111;
      8:  LED <= 16'b111111111;
      9:  LED <= 16'b1111111111;
      10: LED <= 16'b11111111111;
      11: LED <= 16'b111111111111;
      12: LED <= 16'b1111111111111;
      13: LED <= 16'b11111111111111;
      14: LED <= 16'b111111111111111;
      15: LED <= 16'b1111111111111111;
     default: LED <= 16'b1;
     endcase
  end
end
```

Conversion

```
reg [32:0] count;
//binary to decimal conversion
 always @ (posedge(CLK100MHZ))
 begin
   if(count == 10000000)begin
   decimal = data >> 4;
   //looks nicer if our max value is 1V instead of .999755
   if(decimal >= 4093)
   begin
        dig0 = 0;
        dig1 = 0;
        dig2 = 0;
        dig3 = 0;
        dig4 = 0;
        dig5 = 0;
        dig6 = 1;
        count = 0;
   end
   else
   begin
        decimal = decimal * 250000;
        decimal = decimal >> 10;

        dig0 = decimal % 10;
        decimal = decimal / 10;
```

```
        dig1 = decimal % 10;
        decimal = decimal / 10;

        dig2 = decimal % 10;
        decimal = decimal / 10;

        dig3 = decimal % 10;
        decimal = decimal / 10;

        dig4 = decimal % 10;
        decimal = decimal / 10;

        dig5 = decimal % 10;
        decimal = decimal / 10;

        dig6 = decimal % 10;
        decimal = decimal / 10;

        count = 0;
   end
 end
count = count + 1;
end
```

Slide Switches

```
always @(posedge(CLK100MHZ))
begin
  case(sw)
  0: Address_in <= 8'h16;
  1: Address_in <= 8'h17;
  2: Address_in <= 8'h1e;
  3: Address_in <= 8'h1f;
  endcase
end
```