USB Oscilloscope

Final Report

ECE 4901 - Fall 2012

Ethan Dumaine (EE), Sean Fischer (EE), Keun Park (EE), Edward Powell (EE)

Faculty Advisor
Ali Gokirmak
Office: ITEB 335
Phone: (860) 485-9425
Email: gokirmak@engr.uconn.edu

University of Connecticut
Department of Electrical and Computer Engineering

Abstract

Signal capture is critically important for anyone working with electronics. Typically, the tool of choice is the digital oscilloscope, which displays a captured waveform on a screen. Digital oscilloscopes are generally characterized by their sample rate, analog bandwidth, bit-resolution, and memory capacity. Typical starter oscilloscopes have ~100 MHz bandwidth, 8-bit resolution, two channels, and sample at a rate of ~1 GS/s. The capability of these oscilloscopes is limited by their internal memory capacity, which forces users to trade between measurement duration and resolution. Measurement capability improves with increased memory capacity, but at a higher price (>$1000). Cost reduction may be achieved by using the memory of an external PC in place of a memory internal to the oscilloscope. These PC-based oscilloscopes are less expensive, but are limited in performance by the data transfer rate of the PC bus connection. The recent introduction of USB 3.0, which can sustain data transfer rates of 5 Gb/s (10x faster than USB 2.0), provides an opportunity to create a PC-based oscilloscope which can outperform the present PC-based oscilloscopes while still maintaining low cost. Therefore, our goal is to produce an improved PC-based oscilloscope using USB 3.0.

Introduction

An oscilloscope is a tool which measures an electrical signal and projects that signal on a graphical display. There are two primary classes of oscilloscopes: analog and digital.

Analog oscilloscopes display the measured signal using a cathode ray tube (CRT) screen, which consists of a phosphor coated glass and an electron gun (Figure 1). The phosphor glows when electrons hit the phosphor. A beam of electrons from the electron gun is aimed by applying voltage across a set of vertical deflection plates and a set of horizontal deflection plates. The voltage across the vertical deflection plates is modulated by the measured signal, which deflects the electron beam vertically. A linearly increasing voltage is applied across the horizontal deflection plates, which sweep the electron beam horizontally across the screen.

The vertical scale of the oscilloscope is controlled by scaling the incoming signal before it is applied across the vertical deflection plates. Front end amplifiers are typically used for this purpose. These amplifiers determine the bandwidth of the oscilloscope, or the highest frequency that can be accurately measured. More formally, oscilloscope bandwidth is the point at which the incoming signal appears attenuated by 3 dB on the display (Figure 2a). If an oscilloscope with 1 MHz bandwidth is used to measure a 1 V, 1 MHz sinusoid, the signal will appear on the display with ~0.7 V amplitude and ~45^{o} phase distortion.

The horizontal scale of the oscilloscope is controlled by changing the slope of the linearly increasing voltage applied to the horizontal deflection plates. A single voltage ramp may be applied across the horizontal deflection plates based on a trigger event (Figure 2b). These trigger events synchronize the vertical and horizontal deflection plates; an improperly triggered horizontal sweep may result in a garbled display. Common trigger events are based on the

incoming signal. For example, the incoming signal may reach a particular voltage or may rise or fall through a particular voltage. Other specialized trigger events, such as an external clock or mechanical switch, may be used depending on the application.

Digital oscilloscopes translate an incoming analog waveform into a digital representation. The digitized waveform is stored in an internal memory and read by a digital display, which interpolates the measured waveform from the finite set of samples (Figure 3c). Measurement error is inherent with digital oscilloscopes due to this interpolation, thus digital oscilloscopes only produce approximations of the measured analog waveform. The accuracy of the approximation is determined primarily by the sampling rate and the precision of the digital oscilloscope.

The sampling rate determines the time resolution of the digital oscilloscope. An analog oscilloscope, which has infinite time resolution, may be thought of as a digital oscilloscope with an infinite sampling rate. In the frequency domain, the sampling rate determines the sinusoid of highest frequency which can be accurately translated into the digital domain. From signal theory, it can be shown that any periodic signal may be represented as the summation of sinusoids of various amplitudes, phases, and frequencies. The Nyquist theorem states that the sampling rate must be at least twice that of the highest frequency component of the measured signal to avoid aliasing, or distortion due to sampling (Figure 3a). According to Nyquist, a 10 Hz sinusoid may be sampled at 20 S/s without distortion; however, this is generally not sufficient in practice. As a rule of thumb, the sampling rate should be at least ten times that of the highest frequency component of the measured signal to avoid aliasing (Figure 3b).

Depending on the periodicity of the measured signal, it may be possible to sample at a rate less than the highest frequency component. The technique for achieving this is called equivalent time sampling. In equivalent time sampling, a periodic signal is sampled at different points over many acquisition cycles (Figure 3e). Equivalent time sampling depends on the periodic nature of the measured signal. Since a single period is never truly measured in its entirety, small glitches and jitter in the measured signal may not be captured using equivalent time sampling. The opposite is true for real time sampling (Figure 3d), which is the standard sampling method and the only method available for sampling non-periodic signals.

The voltage resolution of the digital oscilloscope is determined by the bit-precision, or the number of bits used to describe each sample (Figure 4a). A digital oscilloscope with 1-bit precision may assign “1” to all samples where the measured voltage is positive and “0” to all samples where the measured voltage is negative. This may be satisfactory if the voltage range under consideration is very small; however, for practical voltage ranges (-20 V to 20 V), 1-bit precision is insufficient. Typical digital oscilloscopes have at least 8-bit precision, which divides

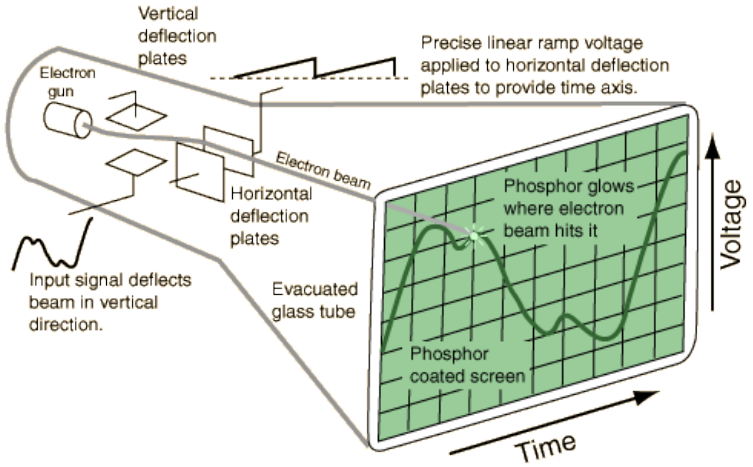

Figure 1. The CRT display of an analog oscilloscope [1].

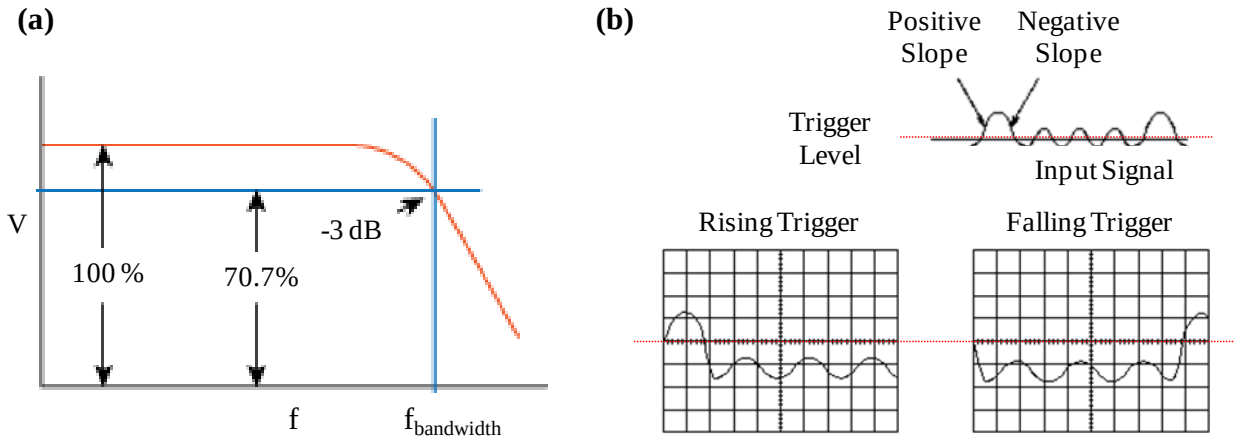

Figure 2. Definition of analog bandwidth **(a)** and common oscilloscope trigger events **(b)** [1].

the input voltage range into 256 discrete levels. High precision oscilloscopes have up to 16-bit precision, which divides the input voltage range into 65,536 discrete levels. An analog oscilloscope may be thought of as a digital oscilloscope with infinite-bit resolution.

In a digital oscilloscope, the digital representation of a measured signal is stored in an internal memory (Figure 4b). The capacity of this memory is finite, and the rate at which it fills is determined by the bit-resolution, the sampling rate, and the number of active channels. The record length of a digital oscilloscope is the number of samples that can be stored in an internal memory per channel. For periodic signal measurement, the horizontal sweep may be triggered in an analog oscilloscope much like a horizontal sweep is triggered in an analog oscilloscope. For measurement of non-periodic signals, a finite record length forces a trade-off between measurement duration and resolution. The maximum duration of high resolution measurements (high sampling rate and high bit-precision) is less than that of low resolution measurements.

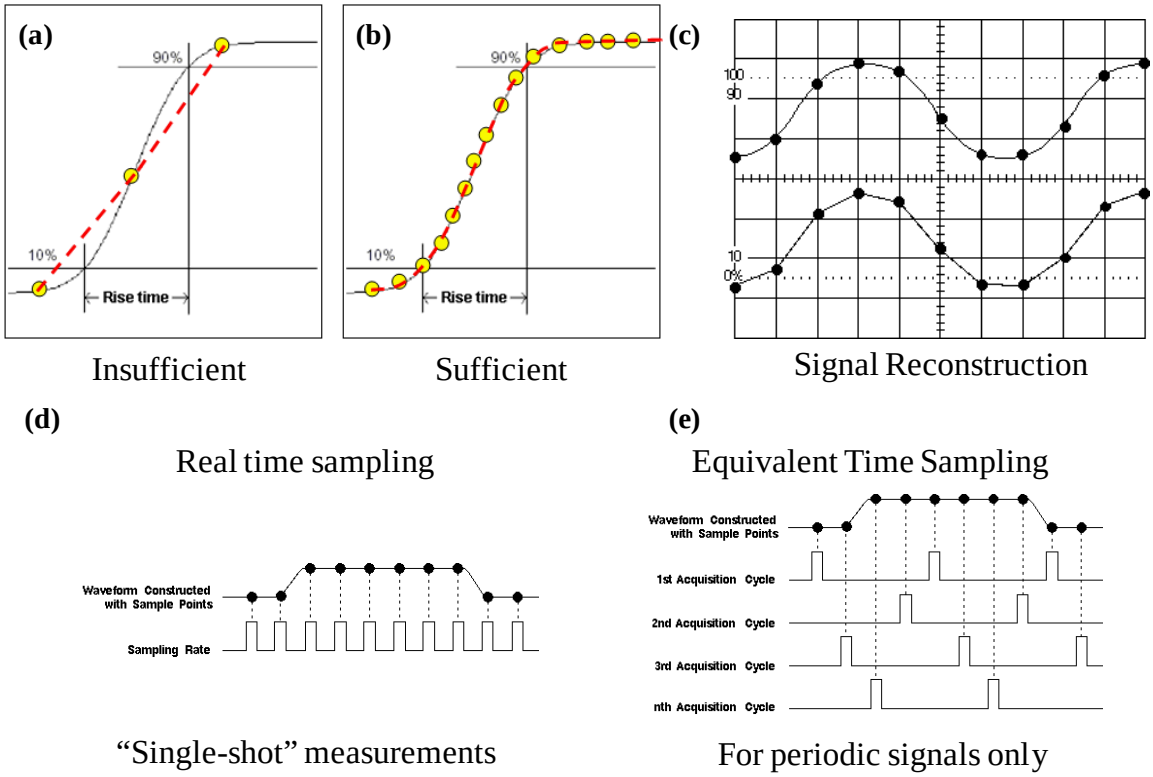

Figure 3. Insufficient **(a)** and sufficient **(b)** sampling of an analog signal. Sinc and linear signal interpolation **(c)**, real time sampling **(d)** and equivalent time sampling **(e)** [1].

Since digital oscilloscopes may only produce an approximation of the measured signal, it is reasonable to question their use. After all, analog oscilloscopes have infinite time and voltage resolution without having to worry about sampling rates, bit precision, or record length. The strength of the digital oscilloscope lies in the ability to store, process, and transmit data. Today's digital oscilloscopes typically have the ability to transmit data to a computer or mass storage device, perform mathematical functions, and perform frequency domain analysis. Such storage and analysis is considerably more involved using analog oscilloscopes, thus the ability of the digital oscilloscope to store, manipulate, and communicate data generally compensates for the loss of precision due to sampling.

Digital oscilloscopes may be categorized based on their architecture. Different architectures may be better suited for particular applications. The cheapest type of digital oscilloscope is the PC-based oscilloscope, which stores captured waveform data in the dynamic random access memory (DRAM) of a computer. The performance of these oscilloscopes is limited by the speed of the bus between the oscilloscope and PC. As data busses become faster, the performance of PC-based oscilloscopes will become more competitive with the performance of traditional lab bench oscilloscopes. This will enable electronics enthusiasts, companies, and educational institutions around the world to afford quality oscilloscope technology.

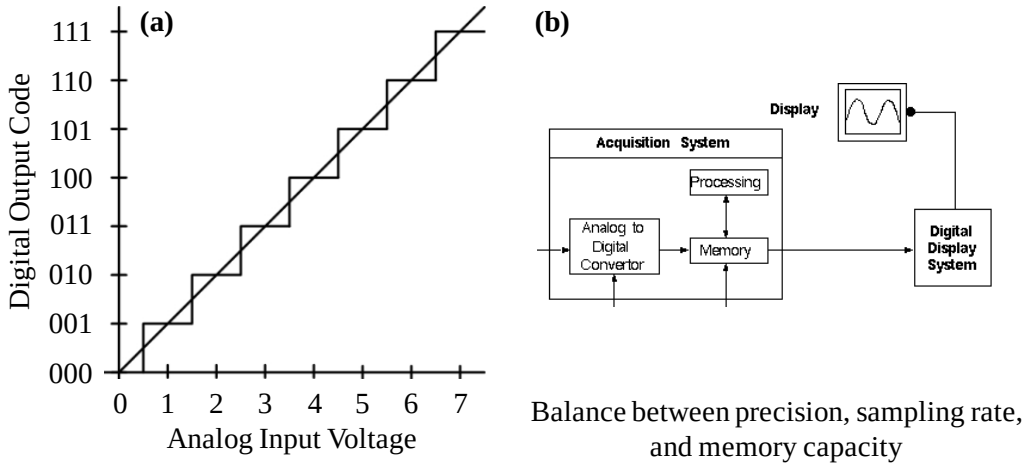

Figure 4. Analog to digital transfer function **(a)** and role of internal memory in digital oscilloscope **(b)** [2].

Our goal is to produce a PC-based oscilloscope with the specifications shown in Table 1. The specified sampling rate and bandwidth is limited by the speed of the data bus (USB 3.0) and cannot compete with the high-end oscilloscopes, which generally have bandwidths in the GHz range. However, the typical memory capacity of today's PCs is at least 1-2 GB of RAM, which is comparable to the internal memory of today's high end digital oscilloscopes. Increased capacity offered by solid state and magnetic hard drives may also be utilized as well depending on the speed of the measurement. The advantage of this PC based oscilloscope over other oscilloscopes is its cost. Similarly performing lab bench oscilloscopes may cost >$500 [3].

Voltage Range	40 V
Analog Bandwidth	25 MHz
Sampling Rate	250 MS/s
Bit-precision	8 bit
Channels	1
Record Length	PC Limited
Functions	peak-to-peak, frequency, FFT, average
Controls	Horizontal, vertical, trigger, coupling
Operating Temperature	0 – 50º C
Dimensions	10 cm x 5 cm x 2.5 cm
Operating System	Windows XP, Vista, 7, 8
Cost	< $300

Table 1. Technical specifications for our PC-based oscilloscope.

Design Overview

The conceptual layout of our PC-based oscilloscope is shown in Figure 5. Our design consists of the following major components:

1) Analog Devices 9481 ADC
2) Spartan 3E FPGA (XC3S500E, CP132 Package)
3) Cypress FX3 USB Controller

The goal is to design a printed circuit board (PCB) which properly powers, interconnects, and protects these components. In addition, the FPGA and FX3 must have the ability to be reconfigured through JTAG. The following sections outline the design considerations of this PCB, as well as the software and firmware written for the FPGA, FX3, and PC.

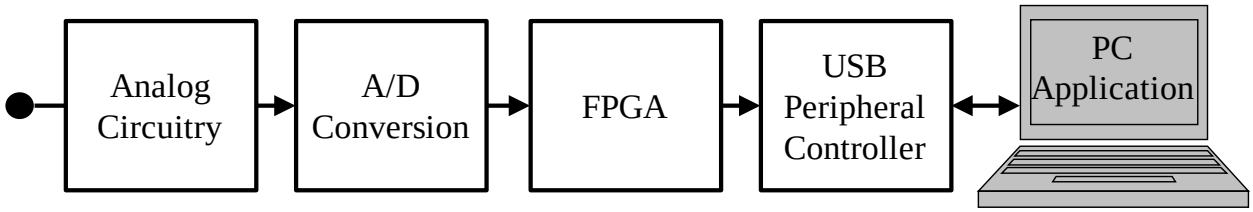

Figure 5. Conceptual layout of our PC-based oscilloscope.

Analog Circuitry

Input signal amplification. Any input signal with amplitude less than 20 V must be amplified or attenuated to match the input range of the analog to digital converter (ADC). The high speed ADC which we have selected (Analog Devices 9481) operates with least distortion when receiving a differential analog input, which eliminates common mode noise. To achieve both of these requirements, we have selected a differential amplifier with a flat frequency response curve beyond 25 MHz (Analog Devices 8138, Figure 6a). The gain of the circuit is controlled by the values of R_F and R_G (Figure 6b). A digital potentiometer controlled by command bits sent from the FPGA will change R_G.

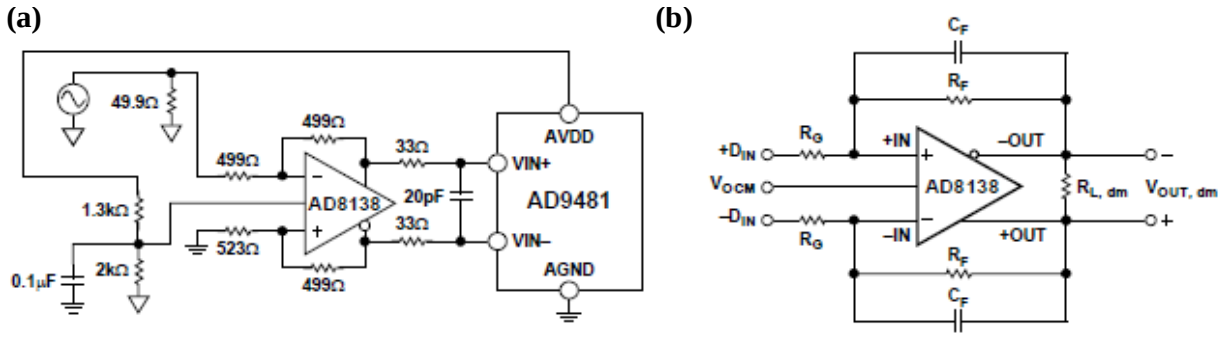

Figure 6. Differential amplifier (AD8138) driving ADC (AD9481) **(a)**. Circuit used for gain calculation **(b)** [3].

Power Distribution. The total current demand of an integrated circuit varies with time. In high speed applications, power supplies struggle to meet the changing current demand due to parasitic inductance. The result is increased ripple and sagging in the supply voltages. Fluctuating supply voltages are particularly harmful in high speed integrated circuits, which generally have more stringent requirements with respect to supply voltage ripple. Power supplies may be stabilized through linear regulators and decoupling capacitors. The first element in any power distribution system is the linear regulator, which maintains a (nearly) constant output voltage and follows changing current demand for frequencies in the kHz range. For frequencies above this range, the integrated circuit may endure many clock cycles where the supply voltage is less than the nominal value before the linear regulator can respond. To overcome this issue, decoupling capacitors (which provide local energy storage) may be added near the pins of the integrated circuit. Decoupling capacitors ranging from 1 to 0.01 μF can respond to changing currents in the kHz to MHz range. Parasitic inductance from contact pads, interconnect, vias, current loops, and equivalent series inductance in capacitors reduces the response time of capacitors and regulators. Therefore, a robust power distribution system not only requires accurate linear regulators and numerous decoupling capacitors, but an effective PCB layout as well.

The linear regulator we have selected (Analog Devices 125, Figure 7) receives input voltages up to 5.5 V (USB supplies 5.0 V) and can regulate output voltages between 0.8 and 5.0 V. The primary voltage supplies required by the major circuit components (ADC, FPGA, and FX3) are provided in Table 2. At least three separate supply voltages are required and ~500 mA of current may be drawn (worst case scenario). Multiple regulators will be used to satisfy these demands.

Each of these supply voltages must be decoupled with capacitors, which respond best to transient current demand within a frequency band centered around a resonant frequency

$$f_{res} = \frac{1}{2\pi\sqrt{LC}} \tag{1}$$

where L is the inductance associated with the capacitor in the PCB (contact pads, interconnect, series ESL) and C is its capacitance. The goal is to select a spread of decoupling capacitors which may supply the transient current needs of the integrated circuit over a wide frequency range. If we choose to have one capacitor per decade and we assume the parasitic inductance is on the order of 1 nH, we obtain the capacitor values shown in Table 3. As a rule of thumb, the number of capacitors should double for every decade decrease in size [4]. The number of capacitors needed depends on the number of supplies and the number of pins associated with each supply. This design utilizes 1.2 V, 2.5 V, and 3.3 V supplies with 21, 4, and 10 pins, respectively. As a first approximation, the number of required decoupling capacitors for each supply is shown in Table 3.

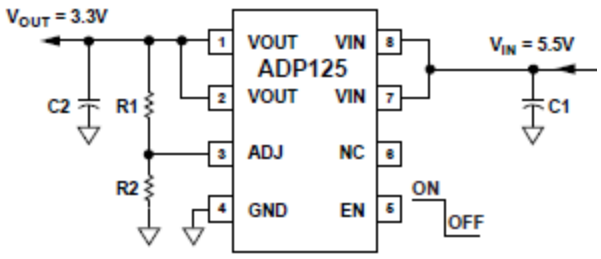

Figure 7. Adjustable linear regulator from Analog Devices (Vin < 5.5 V, 0.8 V < Vout < 5.0 V, Iomax = 500 mA) [5].

Chip	Input	Supply Voltage (V)	Max Supply Current (mA)
AD 9481	AVDD	3.3	145
AD 9481	DRVDD	3.3	43
XC3S500E	VCCINT	1.2	106
XC3S500E	VCCO	3.3	1
XC3S500E	VCCOAUX	2.5	31
FX3	VDD	1.2	200
FX3	AVDD	1.2	-
FX3	VIO1	3.3	-

Table 2. Supply voltages and maximum current requirements for the major components.

f_{res}	C	1.2 V	2.5 V	3.3 V	Total
0.5 MHz	470 µF	1	1	1	3
5 MHz	4.7 µF	1	1	1	3
20 MHz	470 nF	2	1	1	4
60 MHz	47 nF	4	1	2	7
180 MHz	4.7 nF	6	1	2	9
580 MHZ	470 pF	11	2	4	17

Table 3. Decoupling capacitors (43 total).

Ground and power planes will be needed for each supply. This may result in an excessive number of PCB layers, increasing cost beyond acceptable limits (6 layer boards ~$300 from PCB Express). To overcome this issue, we may resort to a single 3.3 V power plane and use voltage dividers to achieve the lower voltage levels. The drawback of this approach is increased power dissipation due to undesirable quiescent currents through the dividers.

Circuit Protection. For the ADC to work after everything has been wired correctly, a signal must come in through the input of the chip without any signal distortions. Signal can be distorted by background noise created by ambience, malfunction of a device being measured where the voltage goes outside the voltage range of the ADC chip, or signal which could be just out of the ADC chip range. Wherever the distortion may come from, it must be compensated so that the ADC can receive a signal staying within the upper and lower boundaries of the chip.

To compensate for white noise, many techniques can be used to get rid of it. A few techniques are AC coupling, DC coupling, average acquisition filtering, and bandwidth limit filtering. AC coupling uses a capacitor in series with the input to get rid of DC voltage or in other words, low frequency. Therefore, when measuring a low frequency AC signal, AC coupling cannot be used to effectively measure the signal. In order to measure low frequency, bandwidth limit filter can be used accurately measure low frequency. Problem is, the user must know he is looking for low frequency in a signal.

To compensate for voltages outside of ADC chip boundaries, a DC voltage can be added or subtracted to the AC signal. The first problem is detecting that the signal is outside of the boundaries. If it is outside of the boundaries and is fed into the input of ADC, it can potentially destroy the chip. To prevent this from happening, zener diode and schottky diode can be used. However using this method only prevents signal from entering into ADC chip. Since the device being made is an oscilloscope, the signal must be measured regardless of the input signal range. To adjust the input signal so that it can be fed into the ADC chip, operational amplifiers and instructions from the FPGA can be used to help adjust the input voltage. Figure 8 is the schematic to check whether or not the incoming signal is too high or too low. For example, if signal is above 4V, the FPGA will get a code of 1111. Then FPGA will give instructions to input negative DC voltage so that the signal is within the boundaries or to tell the signal amplifier to de-amplify the signal.

For the prototype, protections for the chip were not added to save time/money and to show results. However it was made sure that the input signal that went into the ADC chip was within boundaries. For the future board, PCB will be used since faster chips cannot be used on a breadboard due to high frequency.

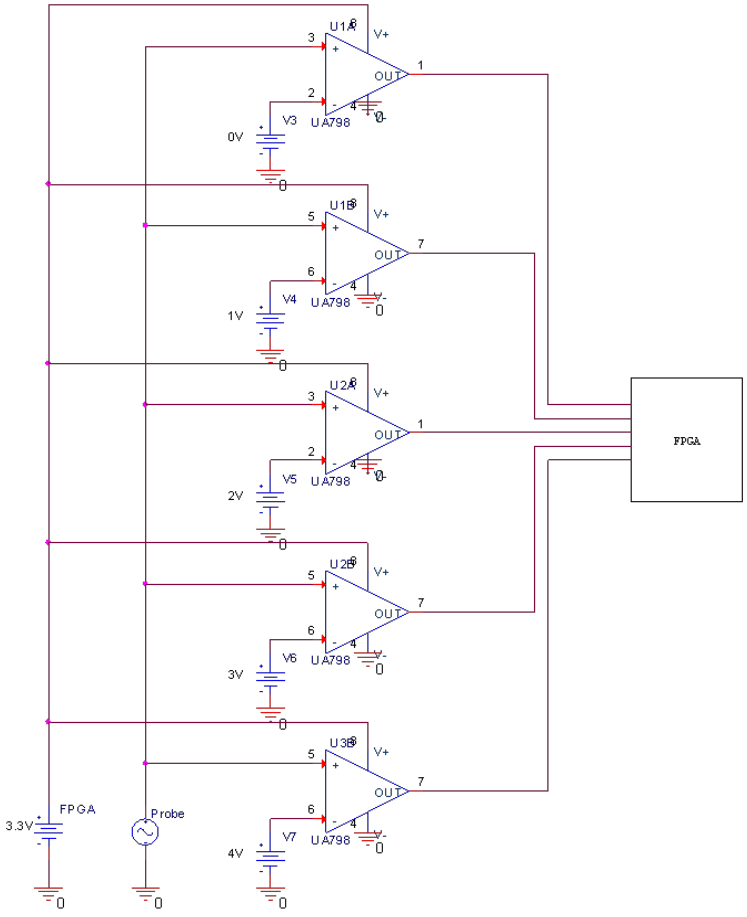

Figure 8. Circuit overvoltage protection.

Analog to Digital Conversion

Sampling Rate. According to the Nyquist theorem, the sampling rate of the ADC must be twice the bandwidth in order to avoid aliasing. The frequency at which this occurs is called the Nyquist frequency. In practical application the sampling rate must be greater than the Nyquist frequency, and we have selected a sampling rate which is 10x higher than our bandwidth of 25 MHz. The Analog Devices 9481 ADC can achieve a sampling rate of 250 MHz.

Bit Precision. The amount of data produced by the ADC every second is

$$\text{bits/second} = \text{sampling rate} * \text{precision} \tag{2}$$

The maximum data transfer rate using USB 3.0 is 5 Gb/s. If we sample at a rate of 250 MHz, we can use at most 20-bit precision. We have selected 8-bit precision to reduce the cost of our ADC.

Output Signaling Method. Fast ADCs communicate primarily through parallel, low voltage differential signaling (LVDS) method. LVDS works by moving current (3.5 mA) through a 100 Ω resistor in parallel with a pair of differential inputs. Current may move in either direction through this resistor (depending on the activation of transistors in the transmitter, Figure 9). Logic 1 and logic 0 are determined by the polarity of the voltage across the resistor (1→ +350 mV, 0 → -350 mV). The advantage of differential signaling (in general) is that the pair of wires share a common mode voltage (and a common ground), which is the sum of a DC offset (typically 1.2 V) and noise introduced during transmission. Since all that matters is the voltage difference between the two wires, noise common to both wires is not seen by the receiver. The complementary magnetic fields produced by the wires cancel each other out, reducing overall noise in the circuit. The disadvantage of LVDS is that the amount of interconnect must be doubled. Interconnect routing becomes more difficult as well, since the noise cancelling properties of the differential pair rely on the two wires being in close proximity with one another.

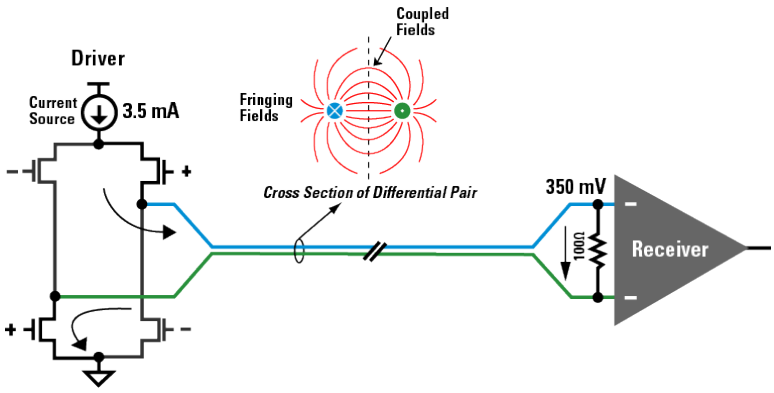

Figure 9. Low voltage differential signaling [6].

The Spartan 3E has the ability to receive both single ended and differential inputs, making LVDS a viable signaling option for this design. Despite its appealing characteristics, however, we have decided to avoid LVDS. The AD 9481 utilizes a parallel CMOS signaling method. CMOS signaling is single-ended, meaning that it does not reject common mode noise. While more susceptible to noise, CMOS signaling will require half of the interconnect needed for LVDS and will simplify PCB development.

Package Selection. The AD9481 is fabricated in a 44-lead thin plastic quad flat package (TQFP) and has 27 pins which must be connected to the FPGA. The FX3 is packaged using a 121-ball fine ball grid array (FBGA) and has 40 pins which must be connected to the FPGA. Therefore, we have selected the XC3S500 fabricated in a 132-ball chip-scale package (CP132). This option provides a maximum of 92 I/O pins, which is sufficient to connect both the ADC and FX3 while providing room for additional pins.

Field Programmable Gate Array

The purpose of the FPGA is to extract data from the ADC and then move that data into the FX3 through the FX3's GPIF II interface. The FPGA is programmed in VHDL using Xilinx ISE Design Suite. The architecture of the circuit is shown in Figure 10. Each box represents a module which is instantiated within a *MAIN* module. These modules work together to achieve continuous sampling.

ADC Interface. This module communicates with the ADC to extract sample data. At present it is written to interface with AD7822 (2 MHz, 8-bit, parallel CMOS output ADC). The AD7822 has a sampling protocol shown in Figure 11. A conversion is initiated on the falling edge of *CONVST*. After a conversion time (t_1), the conversion is complete and *EOC* is asserted. *CS* and *RD* must be asserted to read the digitized data; these signals may be tied to *EOC*. If the circuit is in the active state and *ADC_Interface* is not sampling, *ADCack_o* is asserted, which tells *Control Unit* that *ADC_Interface* is ready for the next sample. *Control Unit* asserts *ADCstb*, which tells *ADC_Interface* to assert *CONVST* and begin the next sample. After a conversion is complete, *ADC_Interface* indicates a valid sample by asserting *ADCwe*, which tells the *Control Unit* that data is ready to be written to the next register in the *Registers* module. Though currently written to interface with an AD7822 *ADC_Interface* will be modified to interface with the faster AD9481.

Registers. This module is a 64-bit register with write enable (*REGen*) and byte addressability (*REGadr*). *Registers* asserts *REGack* to acknowledge a successful write. *REGadr* increments after each successful write, thus sample data from *ADC_Interface* is successively written to each byte of the register.

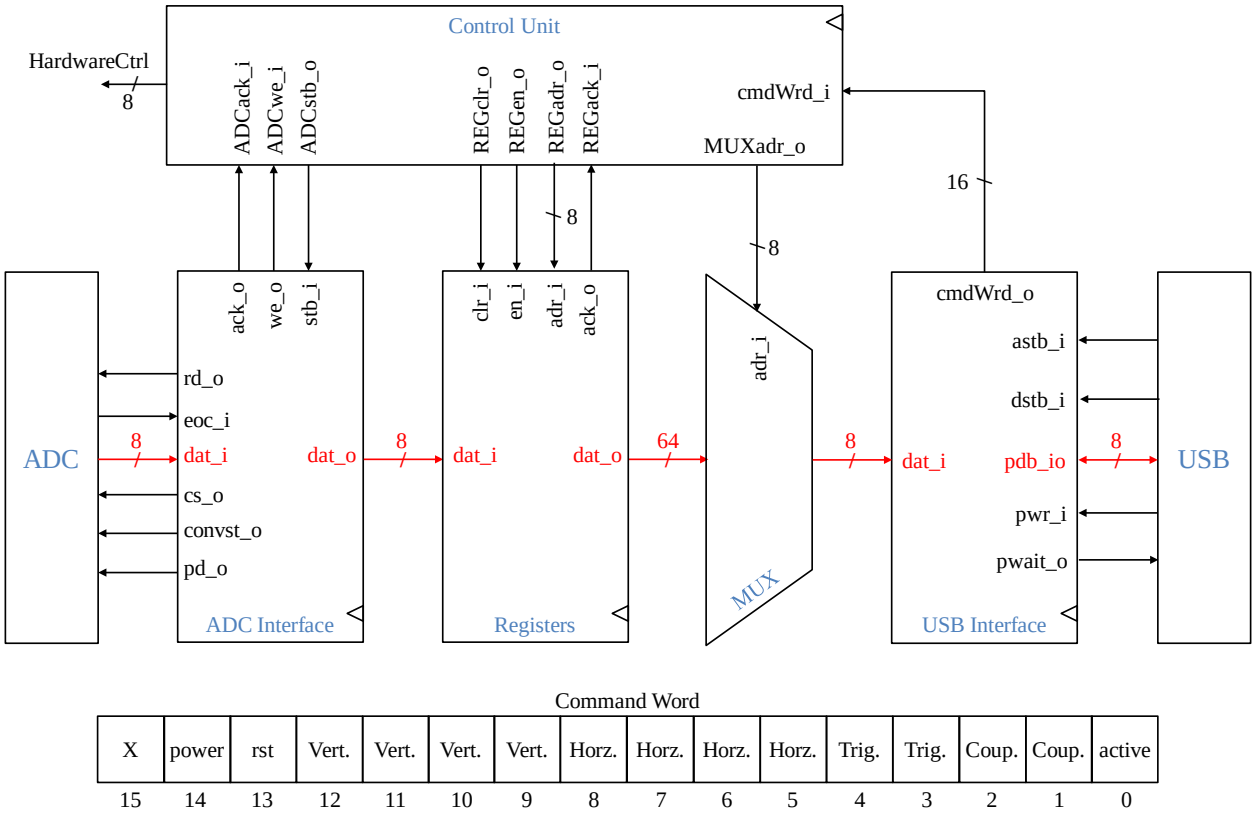

Figure 10. Hardware design within the FPGA.

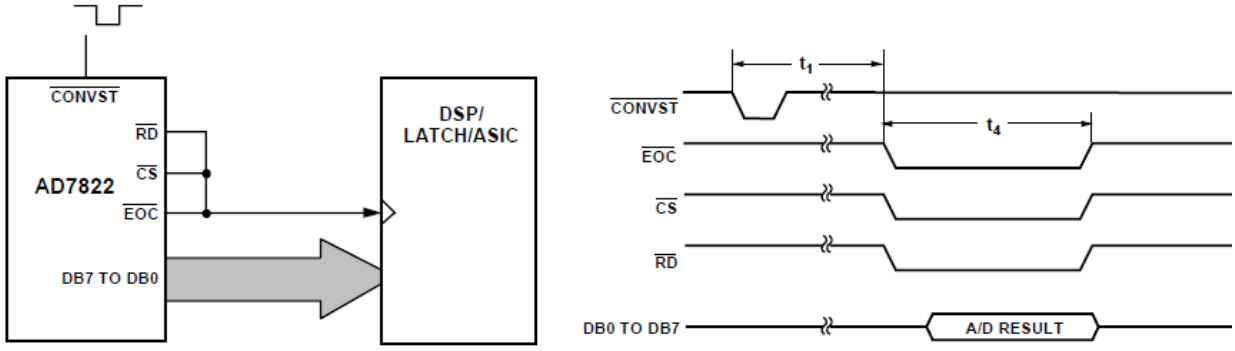

Figure 11. Sampling protocol for AD 7822, the 2 MHz ADC used for prototype [7].

Mux. This module is a 64-bit to 8-bit multiplexer. The output is selected based on *MUXadr*, which always lags behind *REGadr* by one address. Data which is connected to the output of this multiplexer will be sent to the PC through *USB_Interface*.

USB_Interface. At present, *USB_Interface* communicates with a Cypress FX2 peripheral controller. The firmware in the FX2 (provided by Digilent) makes the USB port on the Nexys-2 development board behave like a parallel port. *USB_Interface* is a generic module provided by Digilent to communicate with the FX2 through the parallel port. The interface provides four types of data transfers: Address Read, Address Write, Data Read, and Data Write. The type of

transfer is determined by *ASTB* and *DSTB*, which indicate an address transfer or data transfer, respectively, when asserted. The direction of the transfer is indicated by *WRITE* (1 → data sent to PC from FPGA, 0 → data sent to FPGA from PC). *USB_Interface* indicates it is ready for the next transfer when *WAIT* is low. There are eight data registers and one address register in *USB_Interface*. When a data transfer is taking place, the information stored in the address register determines which data register is being accessed. We modified one of the data registers (Register 0) to store sample data from the *MUX* module when the circuit is actively sampling. For more information on the *USB_Interface*, see the Digilent Software Development Kit [8].

The state of the circuit is determined by the data stored in Register 6 and Register 7 of the *USB_Interface*. These registers form the *Command Register*, which has command bits as shown in Figure 10. The circuit in Figure 10 has three primary states, *Off*, *Reset*, and *Active*. The ADC is put in a low power mode while the circuit is in the *Off* state and all modules remain idle. In *Reset*, all registers are cleared, addresses are initialized, and counters are returned to zero. The circuit continuously samples and presents data to the *USB_Interface* when in the *Active* state.

Control Unit. Based on the bits in the *Command Register*, *Control Unit* (Figure 12) asserts all necessary control signals to move data from module to module. Internally, the *Control Unit* consists of three modules (*ADC_CU*, *REG_CU*, and *MUX_CU*) which are responsible for controlling their respective module (outside of *Control Unit*). These modules communicate with each other to continuously store new sample data and move sample data from previous samples to the *USB_Interface*.

ADCstb commands the *ADC_Interface* to begin the next sample. *ADCstb* is asserted only if the command register indicates the circuit is *Active*, *ADC_Interface* acknowledges that it is ready for the next sample, and *Registers* indicates that it is ready for the next sample. When *ADC_Interface* indicates a valid sample (*ADCwe*), *REG_CU* enables *Registers* and the sampled data is loaded into the byte location indicated by *REGadr*. When *Registers* acknowledges a successful write (*REGack*), *REG_CU* stops asserting *REGen*, increments *REGadr*, and tells *ADC_CU* that it is ready for the next sample. *MUX* is purely combinational and *MUXadr* always lags behind *REGadr* by one address.

Clock Management. The Spartan 3E may accept a clock input from any external oscillator between 5 and 300 MHz, depending on the clock management tool in use. The clock management tools of interest are the Delay Locked Loop (DLL) and the Digital Frequency Synthesizer (DFS). The DLL eliminates clock skew and allows the user to phase shift an incoming clock signal by 90°, 180º, and 270º, double the incoming clock frequency, or invert the doubled clock frequency. The input clock frequency can range between 5 and 90 MHz for the DLL. The DFS divides the incoming clock frequency by an integer ($1 \le q \le 32$) and multiplies the dividend by another integer ($2 \le p \le 32$), producing frequencies between 5 and 307 MHz.

The DFS accepts incoming clock frequencies between 0.2 and 333 MHz. The FX3 has internal oscillator (19.2 MHz) which can be connected to peripheral devices. The clock from the FX3 will be connected to the FPGA and scaled by the DFS to a frequency of 300 MHz. This will allow the FPGA to drive ADC at full speed.

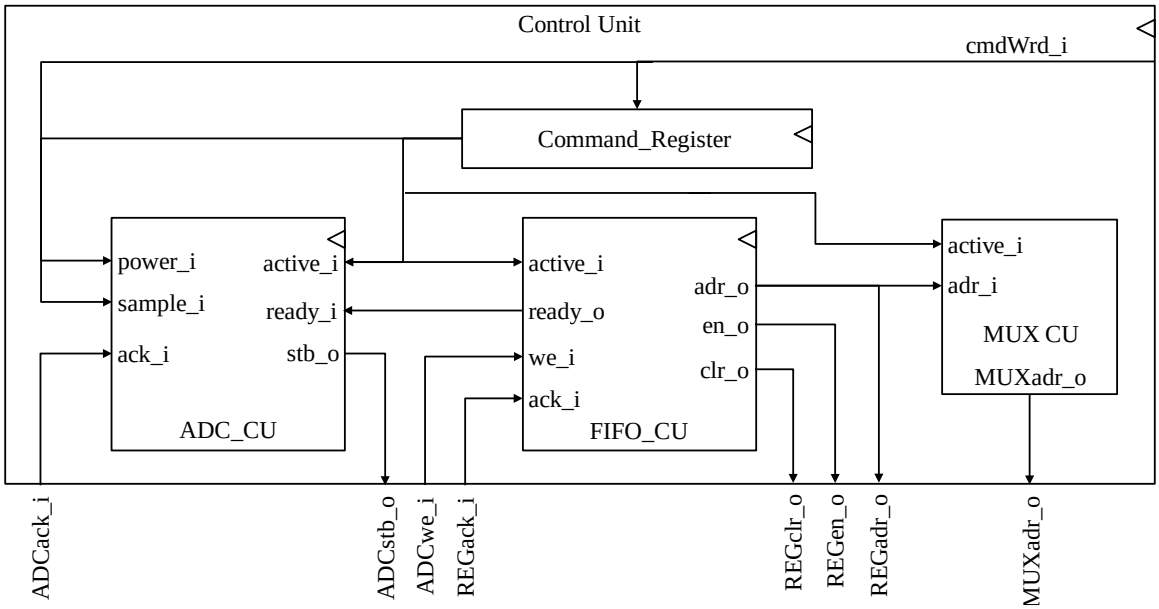

Figure 12. Control unit of FPGA circuit.

Cypress FX3

FX3 Interface. The FX3 has a general programmable interface, called GPIF II, which can operate at a maximum frequency of 100 MHz. The interface is a programmable state machine which can use up to 256 states, support a 32-bit parallel data bus, and 14 configurable control pins. We intend to configure the GPIF II as a slave first-in first-out (FIFO) interface (Figure 13). The FPGA will act as a master, driving data into the internal FIFO registers of the FX3. The FX3 will then export the data to the PC through the USB, where it will be accessible through the Cypress API in C++.

The *USB_Interface* module programmed within the FPGA will be rewritten to move data across the Slave FIFO Interface (Figure 14). *SLCS* is a chip select signal which must be asserted for any data transfer to take place. *SLWR* is a write strobe, it must be asserted to initiate a data transfer from FPGA to FX3. *SLRD* is a read strobe, it must be asserted to initiate a data transfer from FX3 to FPGA. *SLOE* is an output enable signal for sending data from the FX3 to the FPGA. *FLAGA* and *FLAGB* indicate that the FX3 is ready to receive data. *DATA* is the 32-bit data bus between the FPGA and FX3. *PCLK* is the interface clock is to synchronize the interface. The protocol for data transfer across the interface is summarized in Figure 14.

Since the maximum operating frequency of the GPIF II interface is 100 MHz, and the FPGA will collect data at a rate of 250 MHz, the sample data from the ADC must be stored within the FPGA for at least three sample periods before export to the FX3. To fully utilize the 32 bit bus, we plan to transfer four data samples across the GPIF II interface in parallel at ~62 MHz.

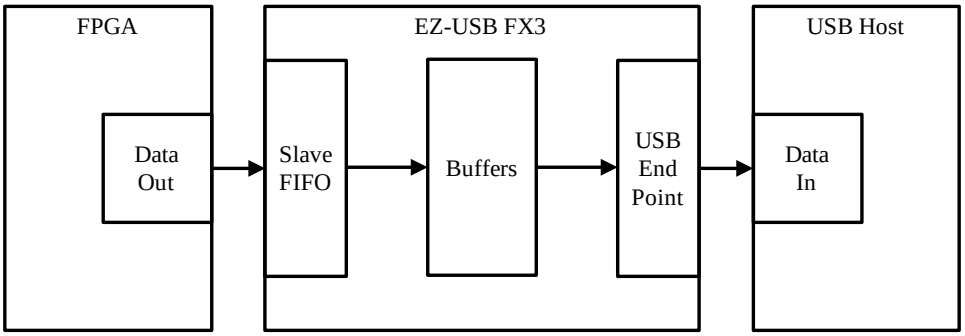

Figure 13. Synchronous slave FIFO interface using cypress EZ-USB FX3 peripheral controller.

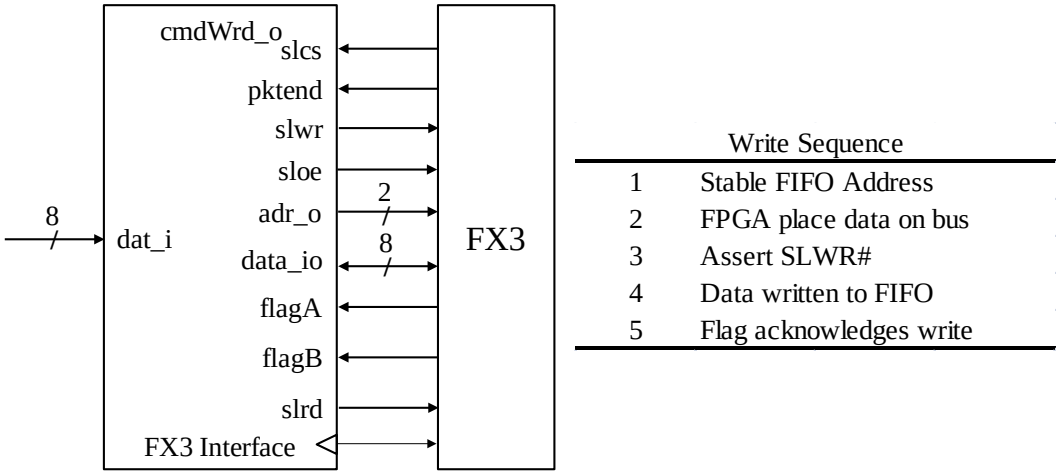

Figure 14. Signals between FPGA and Slave FIFO interface of FX3 and the write protocol to move data from FPGA to FX3.

FX3-PC Interface. Any type of oscilloscope tool needs to have several different components in order to be useful. Oscilloscopes need a way to capture the analog signal, and a way to process that data and display it on some type of screen. One advantage of a PC based oscilloscope is that the PC is used for the data processing and display, meaning that if you were to purchase one of these tools, you would not have to pay for the additional cost of a processor and a screen. A disadvantage of this method is that people who do not have access to a computer will not be able to use the device. The other key advantage of PC-based oscilloscopes is that they simplify that data transfer process. The data collected by standalone oscilloscope will often need to end up on a computer anyway, and the transferring of that information to a computer is an extra step that is not necessary with PC based oscilloscopes. The PC is the scope, and the information is already there.

From the perspective of the user, a well-designed PC oscilloscope can be very easy to use compared to a normal standalone device. From the perspective of the designer however, it is a very different story. A standalone digital scope mostly needs to capture the analog signal, convert it to a digital one and display it on the screen. Most of the modern scopes will also offer options to transfer the information to a computer, but many times these transfers will that place after the measurement has completed, and speed may not necessarily be a consideration. For the most part, a standalone digital oscilloscope is "standalone", it is not required that they have than just a very basic way to interface with other processors. In a PC oscilloscope, the processor for the oscilloscope's data is the computer's processor, so controlling the functionality of a PC oscilloscope requires a certain level of familiarity with writing computer programs.

The processor of a modern computer is very powerful, and the processor in a standalone oscilloscope may be less powerful, but that standalone processor only has the one job of being an oscilloscope processor. The PC processor, by comparison is already loaded down with a lot of processes. Designing a PC application that will be able to operate the oscilloscope at high speeds will require understanding how to use the computer's capabilities to the fullest potential. It is likely that the application may need to be modeled after other existing applications that available for the few existing USB 3.0 devices that take advantage of the new protocol's faster speed.

Cypress's FX3 chip is one of the very few options available to developers of USB 3.0 devices. Trying to make a USB 3.0 compatible device from scratch would be enormously complicated, and using the Cypress chip in the design will greatly simplify things. Cypress does provide a Development Kit and some additional documentation to help developers with the design process, but their product is generally marketed towards companies that have teams of professional computer and electrical engineers with professional equipment and years of experience in this area. The senior design team had to do a significant amount of learning this past semester, and they will need to do some more learning during this next semester in order to be able to use this chip effectively.

PC Application

The idea driving the development of the computer application and the graphical user interface (GUI) for this device is to keep it simple and familiar. Our intention is to maintain the functionality that would be available on a standalone oscilloscope. We also want to make it similar in design to that of the standalone unit by keeping it simple and easy to use. The basic idea is, if you've ever used an oscilloscope before you won't have to relearn how to use it, and if you've never used one before you should be able to learn basic operation fairly quick. Graphing and manipulating the graph and data will also be made simple user friendly.

It used to be that the best way to create a functional user interface was to use C++ or a similar computer language which requires a lot of time and a skilled user. These days we have many other tools at our disposal such as MATLAB and LabVIEW. For this instance we will be using LabVIEW for its ease of use and it graphing functionality. The only foreseeable obstacle with the software integration is communicating between LabVIEW and the USB controller. The data streaming over USB 3.0 will need to be pulled out of the packets and put into a form LabVIEW can analyze, and a format that is exportable to EXCEL in .csv format.

LabView will read the information steaming in through USB to the computer and handle the information. The raw data will be analyzed and displayed onscreen in a graph similiar to the display on a traditional oscilloscope. The controls on the interface will be very similar to those on an oscilliscope allowing adjustments to the graph and measurement of important data such as the peak. It will also contain the functionality required for exporting the raw data in .csv format. In order for this interface to work LabView must be able to communicate with the USB device which, may result in the need to develop a custom driver. A mock up of our intended user interface is shown in Figure 15.

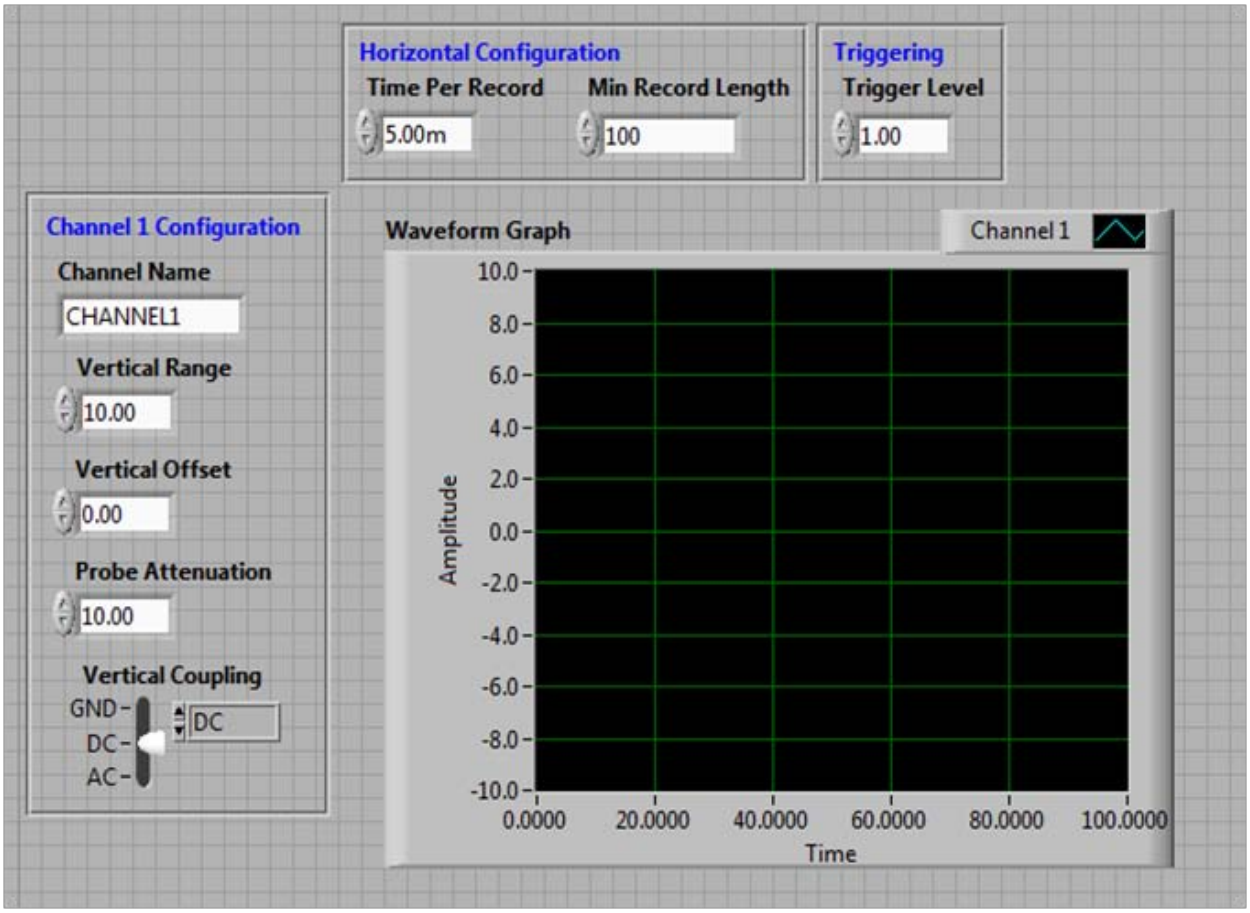

Figure 15. User interface programmed in LabView.

Timeline

The project timeline is shown in Figure 16. The phases were built around the goal of having a working prototype by the end of the first semester. Creating a working prototype by the end of the first semester is a key milestone because it leaves time to work on a final design during the second semester. This milestone also leaves the second semester as a 'cushion' of extra time, should any of the first semester phases not go according to plan.

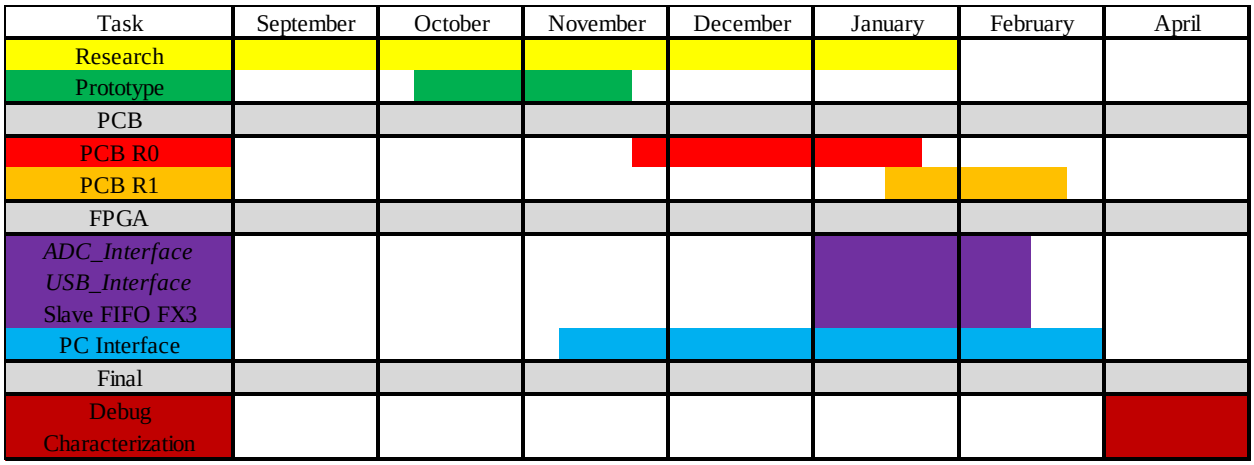

Figure 16. Project timeline.

The timeline of our project is very important because the nature of what we are working on is very time sensitive. USB 3.0 is still a very new technology. It was first introduced in November of 2008, and many companies have yet to design peripherals for it or include in their existing products. Apple for example, finally announced laptops with USB 3.0 capabilities in June of 2012. A device such as a USB 3.0 capable microcontroller or FPGA would potentially make this project a much simpler one, but such devices do not yet exist. Our project is time sensitive because soon someone will make a USB 3.0 microcontroller, and our design will seem expensive and overly-complex by comparison.

Budget

The goal of the project is to construct a device with a cost of less than $300. This is the main limitation on our budget, so even when the additional cost associated with the development are considered, it puts the total budget well below the $1,000 maximum set by the ECE department. The component cost breakdown is shown in *Table X*. The main concern is the cost of the PCB, which increases in cost as the number layers increase (6 layer estimate in Table 4).

The development costs are much higher than the device costs mostly because for the development stage, the chips being used (the FX3 and the Spartan 3E) are much more expensive when they are purchased pre-mounted on a development board or 'kit.' However, purchasing just the chips themselves was not a good option for development, because, for example, the chip may come in a ball-gate array package, which is very difficult to connect to anything except a pre-

designed printed circuit board. To date, the development costs total $320, which includes the cost of the AD7822 and the Cypress FX3 development kit. The other major development cost is the cost of printed circuit boards. When included as a part of the device cost, the cost of a PCB may be reduced, because if they purchased in bulk they can be inexpensive. However, they represent a significant development cost, because there is a minimum dollar amount that has to be considered when ordering PCBs. Ordering just a single, small PCB for device design will represent a much a higher cost per PCB than ordering many PCBs for some scale of device production.

Part ID	Digikey Part #	Description	Unit Price	Quantity	Total
AD 9481	AD9481BSUZ-250-ND	ADC 8-bit, 250 MHz	29.83	1	29.83
XC3S500E-CP132	122-1484-ND	Spartan 3E FPGA	23.01	1	23.01
CYUSB301X	EZ-USB FX3	FX3 USB Controller	38.45	1	38.45
ADP 125	ADP125ARHZ-ND	Adjustable Lin. Reg.	1.90	3	5.70
AD 8138	AD8138ARMZ-ND	Differential Amplifier	8.44	1	8.44
ISL 96017	ISL96017WIRT8Z-TK-ND	Digital Potentiometer	1.28	1	1.28
TAJC477K004RNJ	478-4772-1-ND	CAP TANT 470UF 4V 10% 2312	2.36	3	7.08
C2012X7R1A475M	445-1605-1-ND	CAP CER 4.7UF 10V 20% X7R 0805	0.24	3	0.72
CC0805KKX7R8BB474	311-1364-1-ND	CAP CER 0.47UF 25V 10% X7R 0805	0.12	4	0.48
CX0805MRX7R7BB473	311-1244-6-ND	CAP CER 0.047UF 16V 20% X7R 0805	Call	7	
CX0805MRX7R0BB472	311-1238-1-ND	CAP CER 4700PF 100V 20% X7R 0805	Call	9	
CC0805KRX7R9BB471	311-1124-1-ND	CAP CER 470PF 50V 10% X7R 0805	0.10	17	1.70
Estimate		Resistors	0.05	100	5.00
PCB Express		PCB Manufacture	300.00	1	300.00
BNC Connector	461-1188-ND	Probe Termination	14.81	1	14.81
				TOTAL	436.50

Table 4. Estimated device cost from DigiKey. Resistor price based estimated. PCB manufacture price from PCB Express.

References

1) *Tektronix: XYZs of Oscilloscopes.* Tektronix.
2) *Evaluating Oscilloscope Fundamentals.* Agilent Technologies.
3) Tektronix, http://www.tek.com/oscilloscope.
3) *AD8138 Data Sheet.* Analog Devices.
4) *Power Distribution System (PDS) Design: Using Bypass/Decoupling Capacitors.* Xilinx.
5) *ADP125 Data Sheet.* Analog Devices.
6) *LVDS Circuit Operation*. Digital image. *Wikipedia*.
7) *AD7822 Data Sheet.* Analog Devices.
8) *Digilent Parallel Interface Model Reference Manual*. Digilent.

Personnel Information

Advisor

Ali Gokirmak
Office: ITEB 335
Phone: (860) 485-9425
Email: gokirmak@engr.uconn.edu

Team 174

Ethan Dumaine
Email: ethan.dumaine@engr.uconn.edu

Sean Fischer
Email: sean.fischer@uconn.edu

Edward Powell
Email: edward.powell@uconn.edu

Keun Park
Email: kmp08005@engr.uconn.edu