Ali AlShehab
Tyler Christensen
Nov. 4, 2013
6.111 Project Proposal

The Hardcore Digital Oscilloscope

i. Project Overview

Our final project will use the lab FPGA to implement a digital oscilloscope. The modularity of our FPGA scope allows cheap and easy upgradability and customizability as needed by the customer.

An analog signal will be sampled through an analog-to-digital converter (ADC), and the FPGA lab kit will read, analyze, and visualize the data on a VGA display output by showing the collected waveform as well as minimum voltage, maximum, mean, and frequency in alphanumeric characters. The project will have six input buttons used to set the levels of three parameters: voltage scaling, time scaling, and trigger level. These parameters will work exactly as they do on a typical retail oscilloscope.

ii. Technical Overview

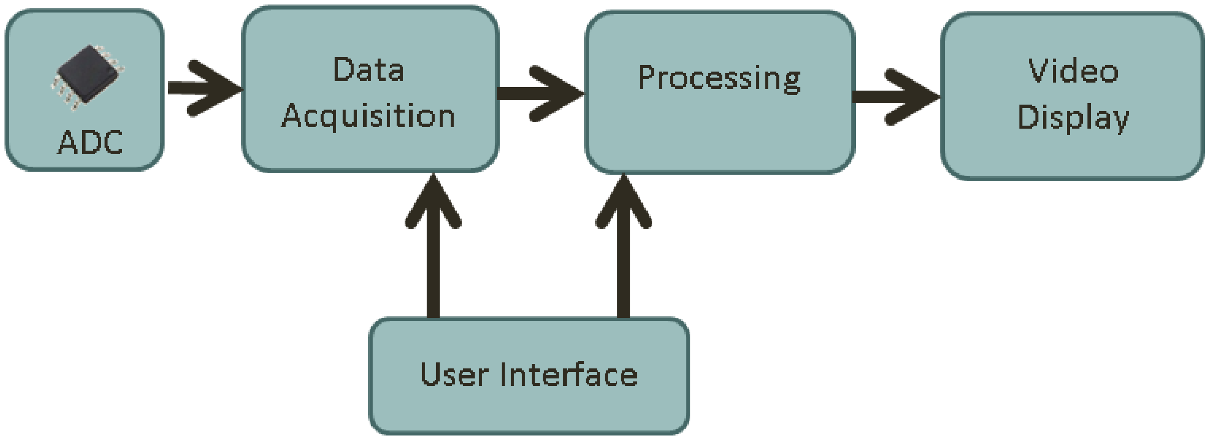

Figure 1 – **System Architecture** – A simplified block diagram showing the data flow from the input stage to the output stage. The data is sampled by an external ADC chip, fed to data acquisition for storage, processed, and finally displayed on a video output. The user interface controls voltage and time scaling of the displayed signal.

The project will be divided into two main components, the input stage (to be completed by Tyler) and display stage (to be completed by Ali). A simplified block diagram showing the layout of the logic can be seen in figure 1 (a more detailed diagram illustrating all blocks as well as bus widths can be found in figure 2 at the end of the document). The input stage begins with the waveform to be analyzed which is fed into the ADC. The output from this is directed through the triggering, sampling, computation, plotting, and scaling modules to ultimately culminate in a BRAM containing the waveform samples to be plotted on the screen. The display stage contains the frame buffer, character map, rendering, and VGA modules which are responsible for taking the plot data and waveform text and displaying these on the display.

a. Data Acquisition and Processing

The input stage consists of the data collection, storage, and analysis for the oscilloscope. The input signal originates in analog form, and is first collected by a 12-bit ADC. This information is fed into a trigger module, which compares the input with a user-configurable trigger level. When this compare signal has a rising or falling edge, a trigger event occurs and a capture module is signaled to begin capturing and storing the input data in a BRAM. If the user has selected a wider sampling time than 10MHz could create, the capturing module records every n^{th} element while the ADC sampling rate remains at 10MHz. For instance, if the user wants to fill the 700 pixel screen width with 700us of data, a 1MHz recording rate would be needed to fill the screen width.

After the 700-element sampled data memory is filled, the computation module is signaled to begin analyzing this data. The analysis consists of two important parts: vertically scaling the data for displaying, and calculating key parameters including minimum voltage, maximum voltage, mean voltage, and frequency. The output from the computation is scaled to 9-bits to reflect the pixel height of the display window, and still contains a 700 sample width. This bit array is stored in a 700x9-bit BRAM array to be read by the display module. The parameters are read by the display module with 15 bits of precision for voltage and 20 bits for frequency. Although the voltages to be displayed are floating point numbers, the modules will communicate through integers with a pre-determined decimal place.

b. VGA Display

The display stage takes the 700x9-bit BRAM, calculated voltage and frequency measurements, and user-input frequency and time division scaling parameters from the input stage, and displays the data visually on a monitor. This will be accomplished using two frame buffers, utilizing a total memory footprint of 1024x768x8x2=12.6Mbit. Since the FPGA does not contain this much memory internally, the external ZBT memory chips on the labkit will hold the frame buffers. A frame buffer module contains two frame buffers: a construction buffer and an output buffer. At any given time, the construction buffer is written to while the output buffer is sent to the VGA module. When the drawing of a frame is completed, the buffers are flipped and the next frame is then drawn to the screen. Every time a new frame is to be drawn, the frame buffer module must collect the data to be displayed and store it in the construction buffer.

The buffered image comes from two places: the plotting module and the rendering module. The plotting module simply contains the BRAM calculated in the input stage of the project. To display this, each of the 700 elements corresponding to a scope trace display column is read. By iterating through all 700 elements, the entire scope trace is drawn from the 700x9-bit BRAM by coloring the pixel corresponding to the value read in the array for each column. Also, the rendering module displays the measurements and scaling parameters by displaying alphanumeric characters on the screen. The data is decoded into decimal format, and the display will be fed pre-programmed alphanumeric characters to write the information on the screen. The rendering module draws the grid lines behind the scope trace. Once all of these tasks have been completed, the construction frame will be complete and contain the next frame to be drawn. The system will remain idle until the frame in the output buffer is fully drawn and it is time to start constructing the next frame.

iii. Analysis of Performance Limiting Modules

The most resource intensive modules in the system are the capturing, computation, and frame buffer modules. Since it is important to consider the possible technical limitations to the implementation before solidifying a design, it is necessary to consider each potential performance bottleneck.

The capturing module will need to be capable of sampling the input at 10MHz and storing the results at up to this speed. This should not be difficult to achieve, since the FPGA memory can store data at up to the clock frequency of the system. Additionally, the latency of the memory is irrelevant since the data will not be read back until the entire data-set is collected.

The computation module will need to perform calculations on the data collected before it can be displayed. So long as this calculation is completed in much less time than it takes the VGA module to display a frame (one 60th of a second), there will be no performance hit incurred from this computation. Meeting this requirement leaves more than 600 cycles of computation per element sampled. Since the only computation necessary will be amplitude scaling and offset, this will easily be accomplished in vastly less than 600 cycles for each column.

Finally, the frame buffer will require performance considerations. Each frame will be drawn in a 60th of a second, so there are 450,000 cycles available for drawing the next frame. The trace can be drawn in 700 cycles. The display will contain no more than 100 characters and with these stored in 8x10 arrays, no more than 8000 cycles are needed to draw the characters. Finally, a grid can be drawn in approximately 10,000 pixels. Summing these, the entire frame buffer can be synthesized in less than 20,000 cycles which is more than an order of magnitude faster than is required. Therefore, the attainable performance of the frame buffer will be more than sufficient for this task.

iv. Testing

The two halves of the project can be tested independently. This will make it possible for each of us to debug code without needing the other person to have completed his part of the project. The output display module can be easily tested on its own by simply providing it with data from a simulated waveform and test characters for the measurements. As long as the display module can display this test data, it should have no problem being fed real-time information. The input stage is slightly more cumbersome to test since it has no visual output, although it should still be possible to test independently. Initial testing will likely be easiest through test benches as the results can be easily saved to data files and drawn using MATLAB. Final testing could be accomplished by exporting the plot BRAM data to the lab computer and drawing this data.

v. Hardware

The only hardware required for this project is the ADC feeding the input capturing stage. The TI ADC12020CIVY chip will be used for data capture. This is a 20MHz free-running chip which will be clocked at 10MHz from the labkit. This chip can be sampled through TI, so obtaining the part should not be difficult. Unfortunately there are no parallel ADC chips in a through-hole package over 2MHz, so a breakout board will be made to hold this chip. This board will be mounted close to the user input pins to reduce inductance and signal rise time.

vi. Deliverables and Conclusion

The scope of this project described thus far is the minimum description of what will be created. The base product will allow the user to select time scale, voltage scale (within some limits set by the ADC resolution), and trigger level. The scope will show this trace along with some information about the captured data. As described, this is a fully functional digital oscilloscope that performs similar to the basic functionality of a retail oscilloscope. This project also has a variety of upgrades that could be implemented as stretch goals for the final project:

- A 12-bit ADC provides only a factor of 8 in extra data precision when a 9-bit signal is displayed on the monitor. Therefore, if voltage scaling is desired to give finer resolution than 8 times less than the full-scale voltage range of the ADC chip, the input would need to be attenuated. A variable gain amplifier could be added to the input signal to perform this attenuation as the voltage scaling is modified.
- The basic implementation requires the user to manually select an appropriate time-scale setting. A retail digital oscilloscope typically has an auto-set button to make this task easier. A similar feature could be added to this project by analyzing the frequency, and adjusting the time scaling to show a pre-programmed number of cycles on the screen.
- The alphanumeric rendering module is implemented using 1-bit black and white character maps. This could be improved to 8-bit color character maps in flash memory to make the characters look less blocky.

vii. Appendix

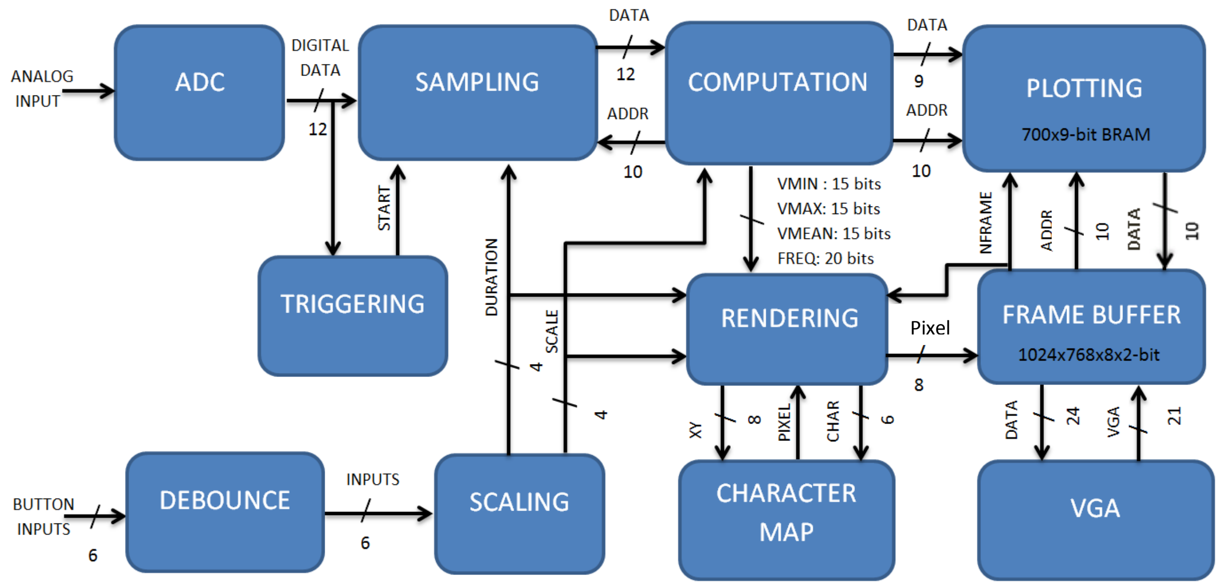

Figure 2 – **Architecture** – a detailed block diagram showing all the modules and their input and output signals. The two input ports on the left provide the waveform itself as well as scaling parameters programmed through pushbuttons. With the help of the triggering module for timing, the waveform is sampled and stored in a BRAM. The scaling and computation modules generate the scaled waveform samples to be stored by the plotting module. This information, as well as waveform parameters gathered in computation, is rendered into a frame buffer. Alphanumeric characters are stored and drawn from the pre-programmed character map. Finally, this data is drawn to the display through the VGA output module.