HDMI Display Oscilloscope

6.111 Final Project Proposal

Daniel Kramnik

1. Introduction

The goal of this project is to implement a digital oscilloscope with an HDMI output that can be connected to a large, high resolution screen superior to what is found in most digital oscilloscopes, or used as a classroom demonstration when connected to a projector. The user interface of the oscilloscope will be designed to be as similar as possible to that of a standard digital oscilloscope, with buttons and knobs mimicking the functions of an analog oscilloscope. In order to make it a useful instrument, the oscilloscope analog front end will connect to a standard oscilloscope probe via BNC connectors, be able to support several full-scale input ranges from 100mV full-scale to 100V full-scale, have two channels, and operate at 100MSPS (million samples per second) for a Nyquist rate of 50MHz.

2. Project Overview

Overall, the oscilloscope is comprised of four main stages of signal conditioning and processing, an analog front end, a capture and storage module, a data processing module, and a display module (Fig. 1). The analog front end buffers and scales the analog input signals, converts them to digital signals, and sends them to the FPGA logic board. Within the FPGA, the capture and storage module detects trigger events such as rising and falling edges and stores the inputs from the analog front end into the FPGA's onboard BRAM once the appropriate conditions are met. A data processing module sets the trigger conditions and timing based on the user's input from buttons and knobs and calculates the locations of the pixels that need to be colored in based on the samples stored in the BRAM and the values of the onscreen displays. This data is sent to the display module, which coordinates writes to the frame buffers in the off-chip DDR2 DRAM and reads from the frame buffers into the HDMI output transmitter.

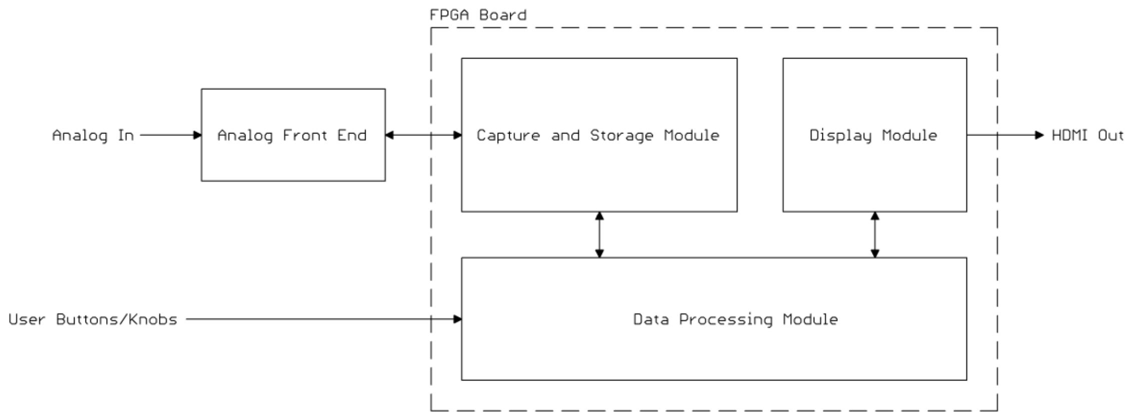

Figure 1: **Block Diagram Showing the Processing Components in the Overall Signal Path of the Oscilloscope.** Analog inputs are converted into digital signals by the analog front end, which then pass through 3 stages of digital processing inside the FPGA that produce an output to the screen. User settings from buttons and knobs affect the data processing step.

3. Implementation

3.1 Analog Front End

The analog front digitizes two channels of analog inputs that are probed with oscilloscope probes. In order to do this, the analog front end must buffer the high impedance outputs from the oscilloscope probes, offset and scale the signals so that they fall into the input range of the analog to digital converters, and apply a lowpass filter to prevent aliasing.

Figure 2 is a block diagram of the signal path in a single channel of the oscilloscope. The oscilloscope probe forms a compensated 10:1 voltage divider with the input filter and an AC/DC coupling relay switches in a series capacitor that can be used to AC couple the input signal. Next, an input buffer comprised of a unity gain JFET-input opamp acts as an impedance converter between the high impedance output of the divider and the low impedance required to drive the bipolar-input opamps later in the signal processing chain. A switchable attenuator and preamplifier form a variable amplifier that can be used to set the range of voltages that the oscilloscope is sensitive to; we want to display anywhere from 100mV full-scale to 100V full-scale on the screen. Next, an anti-aliasing filter blocks frequencies below the Nyquist rate of the ADC. It is necessary to put the anti-aliasing filter as close as possible in the signal processing chain to the ADC to reduce high frequency noise from any high-bandwidth stages that would otherwise be additively aliased into the digitized signal. Finally, an offset generator adds a DC offset to the signal that puts it into the ADC's input range. A precision DAC with an output current buffer and two voltage scalers is used to set the reference voltage of the ADC and the DC offset of the signal. Appendix 1 contains a complete schematic of the analog front end.

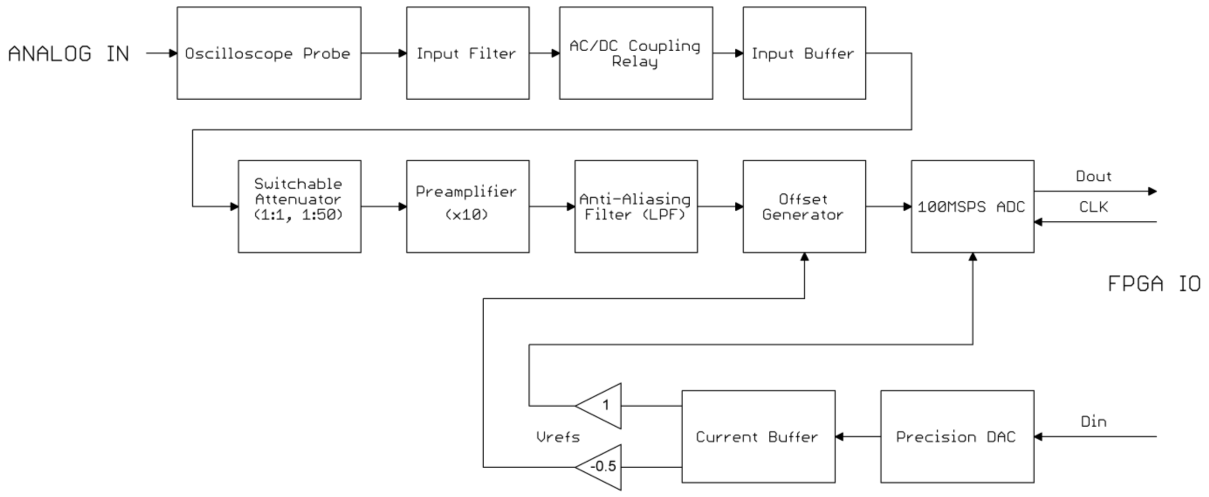

Figure 2: **Block Diagram of the Signal Path in One Channel of the Analog Front End.** An analog input is captured by an oscilloscope probe and passes through a signal conditioning chain that ends with a 100MSPS analog to digital converter that digitizes the signal and sends it to the FPGA logic board.

3.2 Capture and Storage Block

The capture and storage module detects trigger events such as a rising or falling edge of an input signal and stores raw ADC data to the BRAM once such an event occurs. It also coordinates access to the BRAM between the ADCs and the data processing module. The main components in the capture and storage block are a 100MHz clock generator that sets the ADCs' sample rate, FIFO buffers that capture the inputs from the ADCs, a trigger module that communicates with the data processing module and BRAM, and detects when to store the ADC data, and a BRAM that holds the ADC samples. Figure 3 is a block diagram that shows the interconnection of these components.

In order to detect edge trigger events, the ADC data input passes through a FIFO buffer, and the first and last samples of the selected trigger channel are compared against the trigger threshold in order to detect if a rising or falling edge has occurred. An extended FIFO is necessary to prevent triggering on noise when the signal level is close to the trigger threshold, otherwise, the waveform will appear to jitter on the screen due to small variations in the timing of the trigger event between frames. The trigger module only stores data to the BRAM if the data processing module has sent the correct signals. This is used to prevent triggering faster than the refresh rate of the oscilloscope display. Also, it is possible for a periodic waveform to have many trigger level crossings in one cycle, so it is necessary to prevent triggering on all of them (this is what the holdoff signal is used for). The user will set the trigger holdoff using a knob, much like on an analog oscilloscope. When the screen buffer needs to be drawn, the data processing module will request access to the BRAM by asserting the access signal and forward the data to the display module.

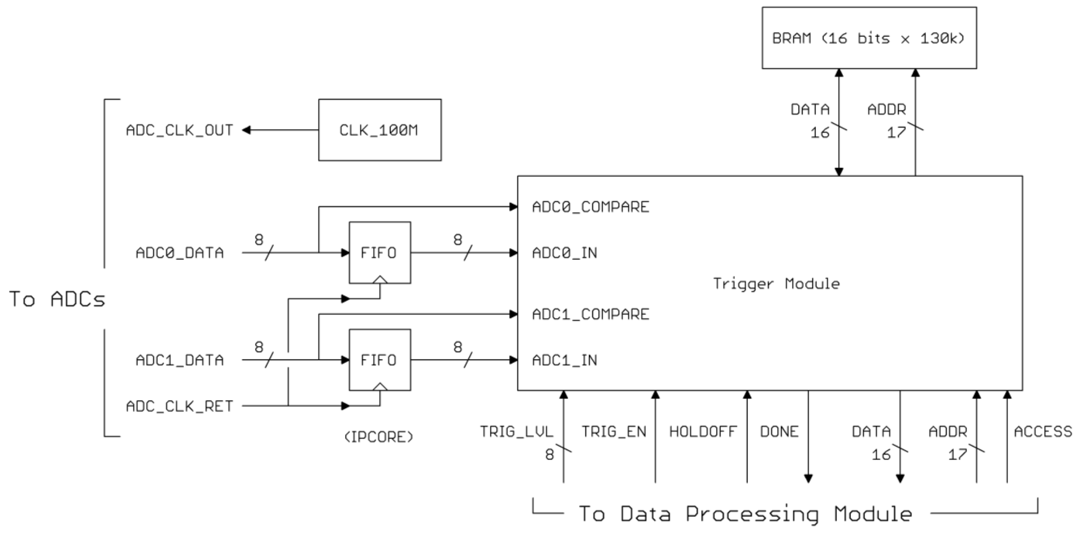

Figure 3: **Block Diagram Showing the Internal Modules and their Connections within the Capture and Storage Block.** The capture and storage block is organized into several main components, an ADC clock generator, ADC input FIFO buffers, a trigger module, and an ADC sample BRAM. The trigger module is at the core of the block, determining when to store samples to the BRAM and when to give the data processing module access to the BRAM to retrieve stored samples.

3.3 Data Processing Block

The data processing block coordinates the timing of the capture and storage module and the display module so that triggering and sampling occurs at the correct time. It also processes the raw ADC data to draw the next frame in the display block's frame buffer. The two main modules inside the data processing block are a display setting finite state machine and a pixel calculator module (Fig. 4).

The display setting control FSM stores the states of all the user-selectable controls such as the trigger channel, channel display offset, vertical scale, and timebase based on the input from the buttons and knobs on the front panel of the oscilloscope. It then select values for the switchable attenuator and reference DAC in the analog front end and sends the pixel calculator the values of the user-selectable controls. It also sends the trigger level directly to the trigger module.

The pixel calculator is responsible for calculating the coordinates and colors of the pixels that need to be drawn to the screen. This includes the graticule (horizontal and vertical grid lines), the current values of the user-selectable settings that are stored in the display setting control FSM, and the input channel traces. The display block will control the data processing module's access to the DDR2 DRAM where the frame buffer is drawn so that memory writes do not conflict with memory reads as the current frame is being drawn; the display module does not perform any additional processing on the pixel values that are calculated in the data processing block.

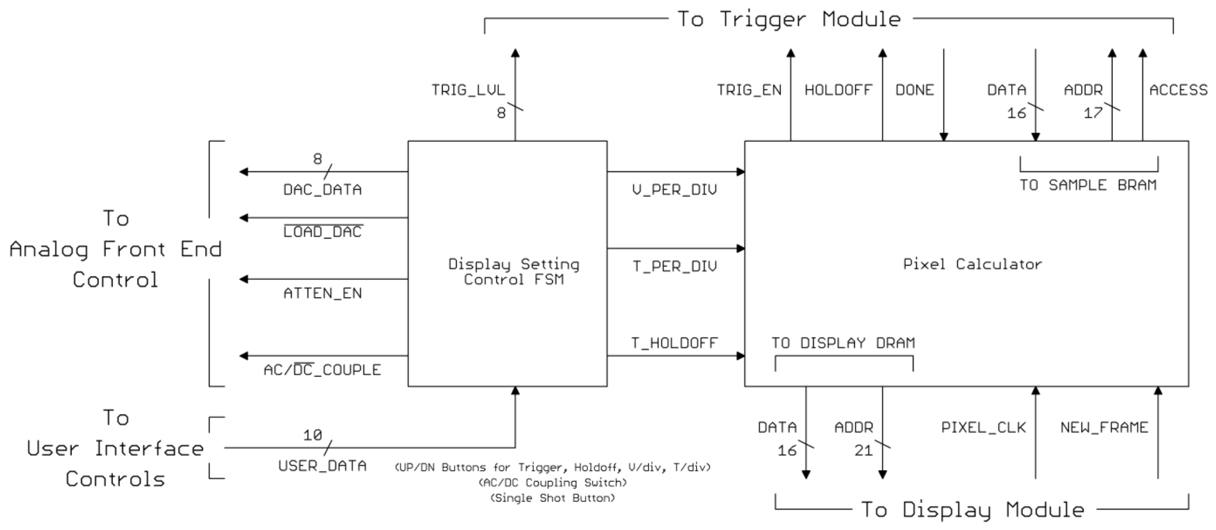

Figure 4: **Block Diagram Showing the Internal Modules and their Connections within the Data Processing Block.** The two main submodules are a display setting control finite state machine that keeps track of the user input from the front panel buttons and knobs and a pixel calculator that draws the next frame into the display module's frame buffer.

3.4 Display Block

The display module draws the oscilloscope display to the screen using an HDMI transmitter. It is built out of three main components, a DDR2 DRAM controller module, an HDMI serializer module, and a memory coordinator module (Fig. 5). The DDR2 controller generates the control signals necessary to interface with the DDR2 chip and the HDMI serializer converts parallel video data in VGA format into serial HDMI packets. These two modules are Xilinx IPCores, while the memory coordinator is written as part of this project.

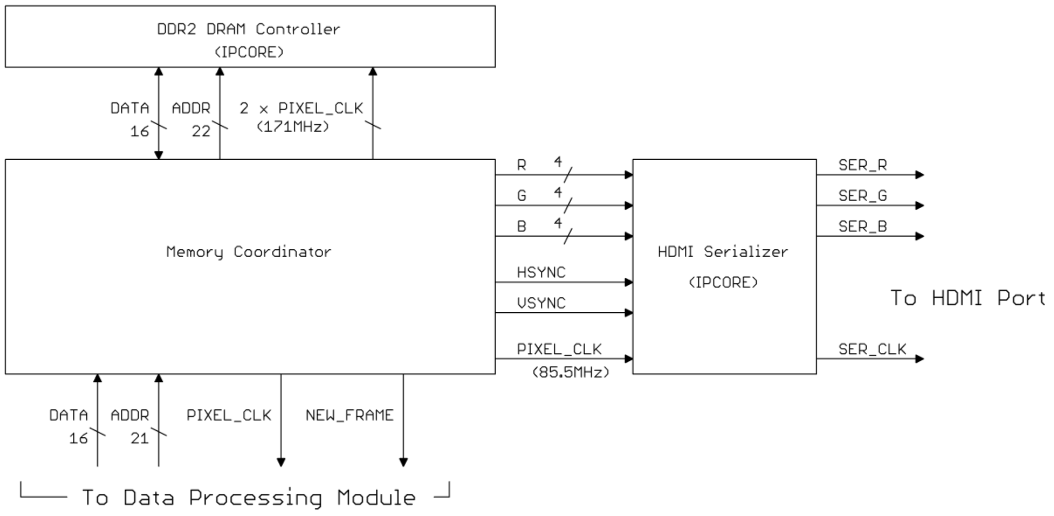

Figure 5: **Block Diagram Showing the Internal Modules and their Connections within the Display Block.** A memory coordinator switches between writing pixels to a buffer in DDR2 memory that contains the next frame and reading pixels from a buffer in DDR2 memory that contains the current frame. An HDMI serializer converts the VGA-style data stored in memory into HDMI serial data.

The memory coordinator module switches between writing pixel data of the next frame to the DDR2 memory and reading pixel data of the current frame from DDR2 memory. Each DDR2 address points of 16 bits of data, which is enough to encode HSYNC, VSYNC, and 4 bits of each color, R, G, and B. Because the memory is external and cannot be partitioned into separately-controlled blocks for each of the two frame buffers, reads and writes cannot occur in parallel. Additionally, each pixel in memory must be marked invalid after it is read into the HDMI serializer because there is no way to reset only one of the frame buffers without clearing the entire contents of the memory. Figure 6 shows the memory map of the frame buffers and the format of their contents. The render pointer (render_ptr) points to the address of the current pixel being read into the HDMI serializer while the pixel calculator pointer (pixel_calc_ptr) points to the address of the current pixel being written into the next frame buffer.

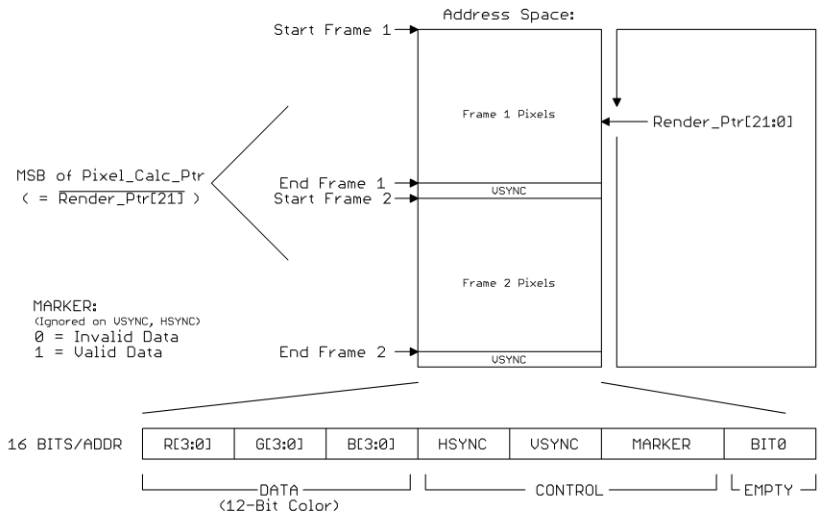

Figure 6: **Diagram of the DDR2 Memory Mapping in the Display Block.** Each 16-bit-wide location in memory contains an R, G, and B value, as well as HSYNC and VSYNC signal values, and a marker bit that determines whether the data in that location is valid. There are two frame buffers; one that the render pointer reads from and another that the pixel calculator pointer writes to.

4. Testing and Timeline

Work on the FPGA side of the project will start with the display block and work towards the ADC input side of the capture and storage module. Because it is difficult to test whether correct pixel data for a 1366x768 screen is being generated using Modelsim, much of the display and data processing submodule testing will be done on the FPGA itself, while the user input FSM and trigger module can be tested using Modelsim test benches. The analog front end will be assembled block by block and the analog stages will be tested using an oscilloscope, while the digital outputs will be tested using a logic analyzer. The following table shows the planned timeline of the project week-by-week:

Week of	**Tasks to Accomplish**
11/4	Schematic, board layout, and sendout of analog front end PCB.
11/11	Display block written and tested with Pong game from lab 3.
11/18	Analog front end board assembled and tested (arrives on Wed. 11/20).
11/25	Basic single-shot capture and storage block and bare-minimum data processing block written and tested. Display a single waveform capture to the screen.
12/2	Debug code, work on adding continuous sampling and user interface stretch goals.

See Also:

Appendix 1: Schematic of Analog Front End