6.111 Project Proposal: Digital Storage Oscilloscope

Daniel D. Richman, Jorge A.Troncoso

1 November 2016

1. Overview

Our project is a full featured digital storage oscilloscope. Using the ADC on the Nexys 4 development board, we will implement a data acquisition system for signals up to 500 kHz (the Nyquist limit for the ADC). The oscilloscope will offer a customizable triggering system and will display the signals on a computer monitor. We also aim to provide provide signal analytics such as cursors, measurements, and calculations such as a Fourier transform to show a frequency-domain representation of the input signal.

Our project has two major pipelines. The data acquisition and processing modules are shown in yellow on the block diagram and include external analog circuitry, ADC controller, trigger, sample buffer, and signal processing components. The second is the VGA display system and its associated sprites, shown in green.

2. Data Acquisition Modules

2.1 External Analog Circuitry (DDR)

The Nexys 4 analog to digital converter (ADC) only supports input signals between 0 and 1 V, so we include a simple external analog biasing and scaling circuit, shown below. The output of the circuit is $V_{out} = \frac{1}{2}V_{cc} + [V_{in} - V_{cc}/2]\left(\frac{Rscale/2}{Rscale/2 + Rin}\right)$.

V_{cc} = 1.0V to satisfy the ADC requirements. We select R_{in} so that the circuit has a high input impedance, so as not to load the input signal. We then select R_{scale} based on the range of the input signal. For instance, we can choose R_{in} = 2MΩ, R_{scale} = 1MΩ, which will shrink the signal by a factor of 1/4 before biasing it so that it sits in the range 0-1V. The input impedance of this circuit is 2.5MΩ, which is comparable to an oscilloscope. (We have also designed a fancier op-amp based circuit if necessary.)

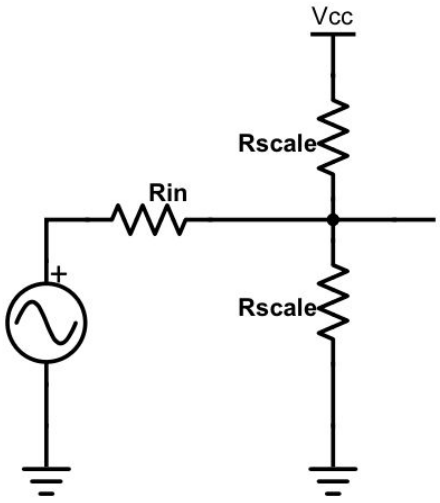
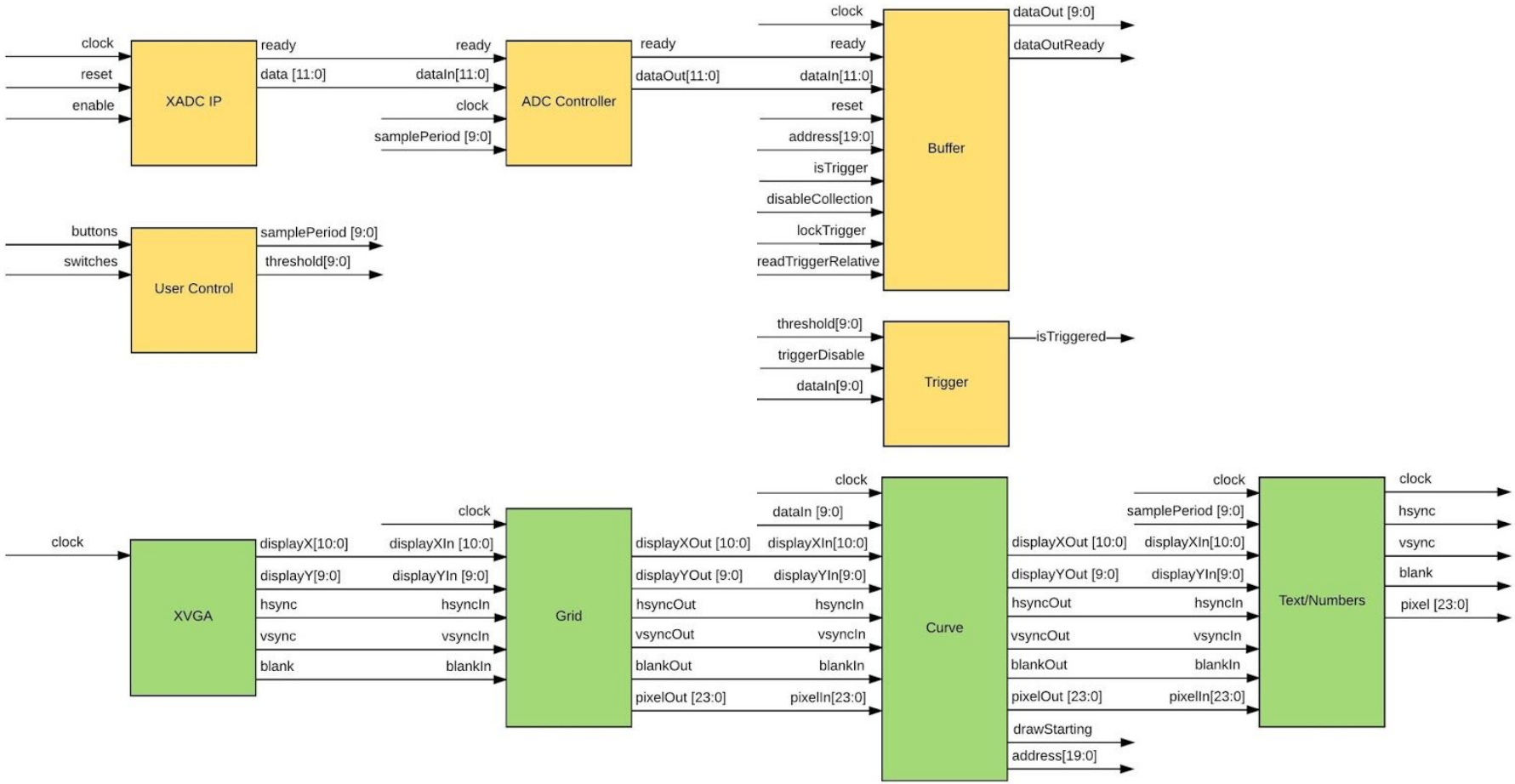


Figure 1: Analog biasing circuit


clock
reset
enable
XADC IP
ready
data [11:0]
ready
dataIn[11:0]
clock
samplePeriod [9:0]
ADC Controller
ready
dataOut[11:0]
clock
ready
dataIn[11:0]
reset
address[19:0]
isTrigger
disableCollection
lockTrigger
readTriggerRelative
Buffer
dataOut [9:0]
dataOutReady
buttons
switches
User Control
samplePeriod [9:0]
threshold[9:0]
threshold[9:0]
triggerDisable
dataIn[9:0]
Trigger
isTriggered
clock
XVGA
displayX[10:0]
displayY[9:0]
hsync
vsync
blank
clock
displayXIn [10:0]
displayYIn [9:0]
hsyncIn
vsyncIn
blankIn
Grid
displayXOut [10:0]
displayYOut [9:0]
hsyncOut
vsyncOut
blankOut
pixelOut [23:0]
clock
dataIn [9:0]
displayXIn[10:0]
displayYIn[9:0]
hsyncIn
vsyncIn
blankIn
pixelIn[23:0]
Curve
displayXOut [10:0]
displayYOut [9:0]
hsyncOut
vsyncOut
blankOut
pixelOut [23:0]
drawStarting
address[19:0]
clock
samplePeriod [9:0]
displayXIn[10:0]
displayYIn[9:0]
hsyncIn
vsyncIn
blankIn
pixelIn[23:0]
Text/Numbers
clock
hsync
vsync
blank
pixel [23:0]

2.2 ADC Controller Module (DDR)

The physical ADC unit is controlled by the XADC IP core from Xilinx, configured to 1 MSPS in single-channel mode. We can also read analog values sequentially from multiple channels as an extra feature. The IP core *ready* and *data* outputs are connected to our ADC controller module. The module downsamples the data as requested by the *samplePeriod* input and outputs its own *ready* and *data* signals. Downsampling the data at this stage allows us to avoid the complexity of doing it once the data is already in the buffer. However, the user will not be able to zoom the time axis of the display without waiting for new data to arrive, since the old data was sampled at a slower rate.

2.3 Trigger Module (JAT)

The trigger module checks if the incoming data crosses the trigger threshold. If the incoming data crosses the threshold and *triggerDisable* is a logic low, the *isTriggered* signal is asserted for one clock cycle. Otherwise, the *isTriggered* signal remains a logic low. The trigger can be configured to fire on either rising or falling edges.

2.4 Buffer Module (DDR)

The buffer module stores ADC samples for further processing. In our initial design its input will be the ADC module's data output, although its input could also come from a math module or other preprocessing unit. The buffer instantiates two BRAMs, an *active BRAM* and a *locked BRAM,* that it uses as ring buffers. Incoming samples are stored in the active BRAM. When the *isTrigger* input goes high, the address of the current sample is recorded as the most recent trigger sample in the active BRAM. When the *lockTrigger* input goes high, the active and locked BRAMs swap roles.

The buffer module only supports read access to the locked BRAM. If *readTriggerRelative* is high, then the *readAddress* input is interpreted relative to the address of the last trigger sample in the locked BRAM. Otherwise, the *readAddress* input is interpreted relative to the most recent sample in the locked BRAM. In either case, the requested sample is output on *dataOut* and *dataOutReady* is set to high for one clock cycle.

3. Display System Modules

3.1 Architecture/XVGA Module (DDR)

Our display architecture consists of a sprite pipeline. Different sprites have different latencies, so all XVGA control signals (*displayX, displayY, hsync, vsync, blank*) pass in series through all sprites. (We refer to the signals *hcount* and *vcount* as *displayX* and *displayY*.) Each sprite is responsible for appropriately delaying the control signals so that they reach the monitor at the same time as the appropriate pixel.

Each sprite also outputs a *pixel* value, which it either passes through from the previous sprite or overwrites with its own data. If more sophisticated display becomes necessary, we will add a z-index line.

3.2 Grid Sprite (JAT)

The grid module draws a black background and grid lines on the screen.

3.3. Curve Sprite (JAT)

The curve module is responsible for drawing the input waveform on the screen. At the start of each frame, it asserts the *drawStarting* output, which is connected to the buffer's *lockTrigger* input. On input (*displayX*, *displayY*), it requests the value of the sample at address *displayX* from the buffer. It then scales the value of the sample to match the user-requested scaling (see section 4) and outputs a colored pixel if the scaled value should be plotted at position *displayY* on the screen. Otherwise, it passes through the previous pixel from the grid module.

3.4 Text Sprite (JAT)

The Text module is a sprite that draws characters at a specified position on the screen. We anticipate a font that supports letters, numbers, and a few symbols. The module has a parameter, *DISPLAY_LENGTH*, that controls the number of characters it can show. It also has a *characterString* array input that specifies the value of each character. The enabled pixels for each character are stored in lookup table. When drawing a character, the module requests data from the lookup table and sends the appropriate pixel values to the screen. There will probably be more than one Text module in our system. For simplicity, only one is shown in the block diagram.

4. User Control Module (JAT)

The user control module is responsible for reading inputs from the user (e.g. buttons and switches) and distributing the signals to other modules that take user inputs. These inputs include parameters like sample rate, vertical scaling, and trigger level. This module will have internal debounce modules to produce a clean output.

5. Bonus Features

These features represent stretch goals and are not shown in the block diagram.

5.1 Multiple Channels

The ADC on the FPGA supports two channel input at the highest sampling rate, or more channels at a slower rate. We would like to add support for multiple channels, which would simply require multiple copies of modules in the signal acquisition pipeline and multiple curve sprites.

5.1 Cursors and Measurements

One of our stretch goals is to add cursors to measure the amplitude and period of a signal. The cursors and text showing the amplitude and period measurements can be implemented as sprites. The data they display would come from the math modules.

5.2 Math Analysis

An important feature of real oscilloscopes is mathematical analysis, providing autoset and cursor measurement information as well as possibly new curves to display, such as a frequency-domain

representation of the input signal. Each of these features is a stretch goal for us. We would particularly like to add FFT support using the Xilinx FFT IP core.

5.3 Autoset

Autoset is an additional feature we would like to add. When the user presses the Autoset button, the Autoset module computes nice-looking display settings (trigger level and scaling) based on the samples in the buffer.

5.4 Frame Averaging

We would like to support frame averaging, which displays an output waveform that is averaged over several frame periods to remove noise. This would require a new averaging module and additional support from the buffer module.

5.5 Run/Stop

We plan to add a Run/Stop feature to our oscilloscope. This freezes the buffer so that same waveform is displayed on the screen at all times. This can be implemented in conjunction with a single-shot triggering mode, which stops the oscilloscope as soon as the trigger fires once.

6. Memory Usage

We use memory for three purposes: buffers for sample storage, mathematical scratch space, and display sprite assets. The 4,860 Kbit of block ram (BRAM) on the Nexys 4 DDR board provides ample storage for our purposes. Although the Nexys 4 DDR also has 128 Mbytes of DDR2 SDRAM, interfacing to this memory is more involved and should not be necessary.

Displaying a curve needs about 1,000 samples (one for each vertical line of the display on which we will draw part of the signal). To continue capturing during the display of a frame, we will use two ring buffers, one fixed while the display is drawing and the other storing data at the user-configured sample rate. The ADC samples with 12 bits of precision, so our minimum sample memory is (12 bits/sample)(1,000 samples/buffer)(2 buffers/system) = 24 Kbit of BRAM.

One of our stretch-goal features is to add various mathematical analysis tools, including simple features like addition and subtraction of two ADC channels as well as more complicated features like a Fast Fourier Transform. The specifications for the Xilinx FFT IP Core block suggest that it requires about 1,000 Kbit of BRAM. Math mode features will also require additional storage buffers to interface with the display module.

The third major memory consumer is the text display system. Sprites display text by using a lookup table that stores the enabled pixels for each character in the font. Each character will be approximately 15 x 10 pixels, for a total of 150 bits per character. We estimate about 40 characters (uppercase alphabet, digits, and some punctuation), so the total font memory required is about 6 Kbit.

7. Conclusion

Our digital storage oscilloscope will allow analysis of real-time input signals and provide many useful features. The modularity of the system allows easy extensions, particularly in terms of adding more analysis and display features, which simply require new math modules. We hope that our product will outsell Tektronix!