Seth Iris Canonigo
Janette Tse
12-5-24

Lab 5: VGA Oscilloscope

Introduction:

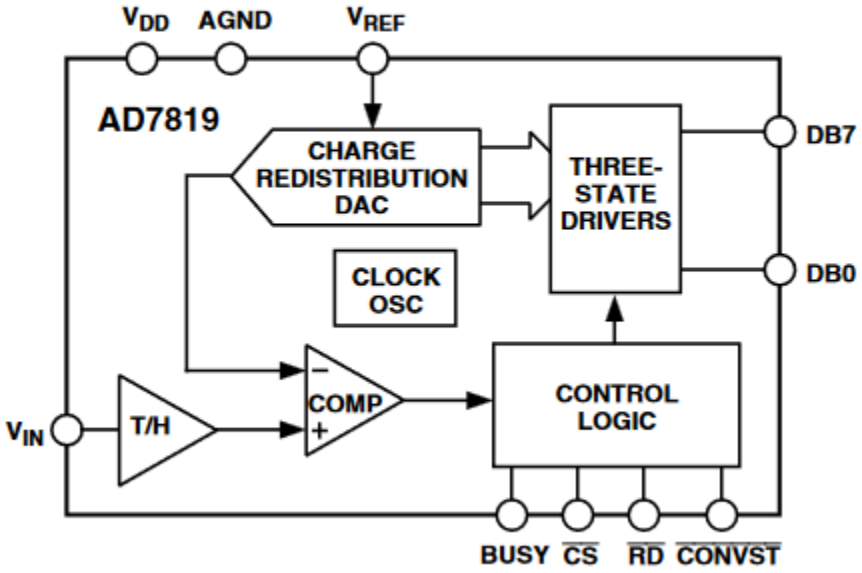
Oscilloscopes are essential tools in electronics, enabling visualization of time-domain signals. Traditional oscilloscopes rely on analog hardware, but modern designs often leverage digital systems for added flexibility. This lab project aims to build a simple digital oscilloscope using an FPGA, specifically the Basys 3 board and an AD7819 8-bit parallel A/D converter. By leveraging the processing power of the Basys 3, the oscilloscope can sample analog signals, process the data in real-time, and, using the VGA Controller from the previous lab, display the waveform on a 640x480 monitor. This project helps demonstrate the integration of digital systems, high-speed signal processing, and real-time display capabilities in a single device.

Background and Theory:

The AD7819 is a high-performance, 8-bit analog-to-digital converter (ADC) designed for fast and accurate signal digitization. It operates with a single power supply and offers a straightforward parallel data output interface, making it ideal for integration with digital systems such as microcontrollers, FPGAs, or DSPs. Its compact design and high-speed operation make it suitable for applications requiring real-time analog-to-digital conversion.

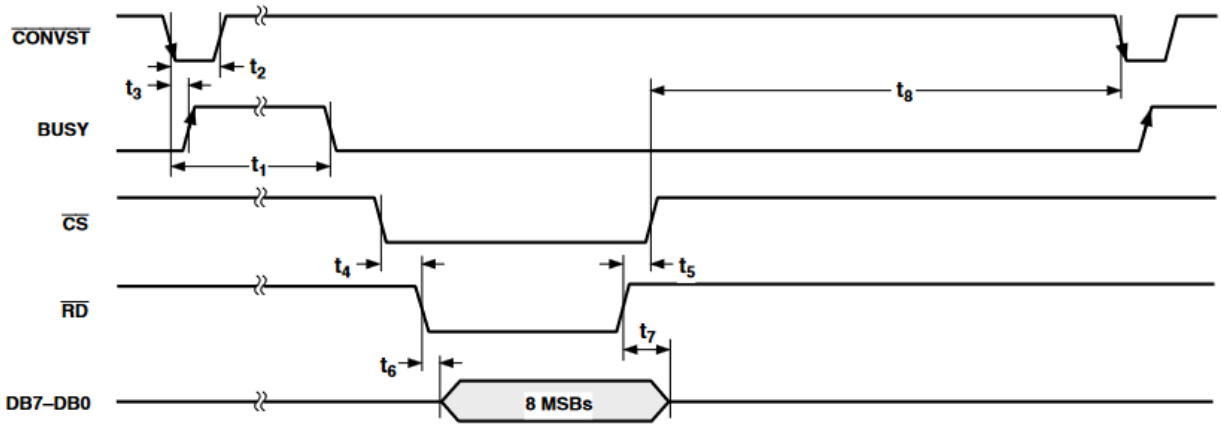
This ADC features a low conversion time of just 2 microseconds, allowing it to handle high-speed sampling tasks efficiently. The input voltage range is from 0V to the reference voltage (VREF), which can go up to 5V, providing flexibility for various signal ranges. The AD7819 uses control signals like CONVSTB, CSB, and RDB to manage its operation, ensuring precise timing for sampling, conversion, and data retrieval.

FUNCTIONAL BLOCK DIAGRAM



TIMING CHARACTERISTICS[1, 2] (–40°C to +125°C, unless otherwise noted)

Parameter	V_{DD} = 3 V ± 10%	V_{DD} = 5 V ± 10%	Unit	Conditions/Comments
$t_{POWER-UP}$	1.5	1.5	µs (max)	Power-Up Time of AD7819 after Rising Edge of $\overline{CONVST}$.
t_1	4.5	4.5	µs (max)	Conversion Time.
t_2	30	30	ns (min)	$\overline{CONVST}$ Pulsewidth.
t_3	30	30	ns (max)	$\overline{CONVST}$ Falling Edge to BUSY Rising Edge Delay.
t_4	0	0	ns (min)	$\overline{CS}$ to $\overline{RD}$ Setup Time.
t_5	0	0	ns (min)	$\overline{CS}$ Hold Time after $\overline{RD}$ High.
t_6[3]	10	10	ns (max)	Data Access Time after $\overline{RD}$ Low.
t_7[3, 4]	10	10	ns (max)	Bus Relinquish Time after $\overline{RD}$ High.
t_8[3]	100	100	ns (min)	Data Bus Relinquish to Falling Edge of $\overline{CONVST}$ Delay.



200 kSPS = 200000 cycles per second = 5000 ns sampling period

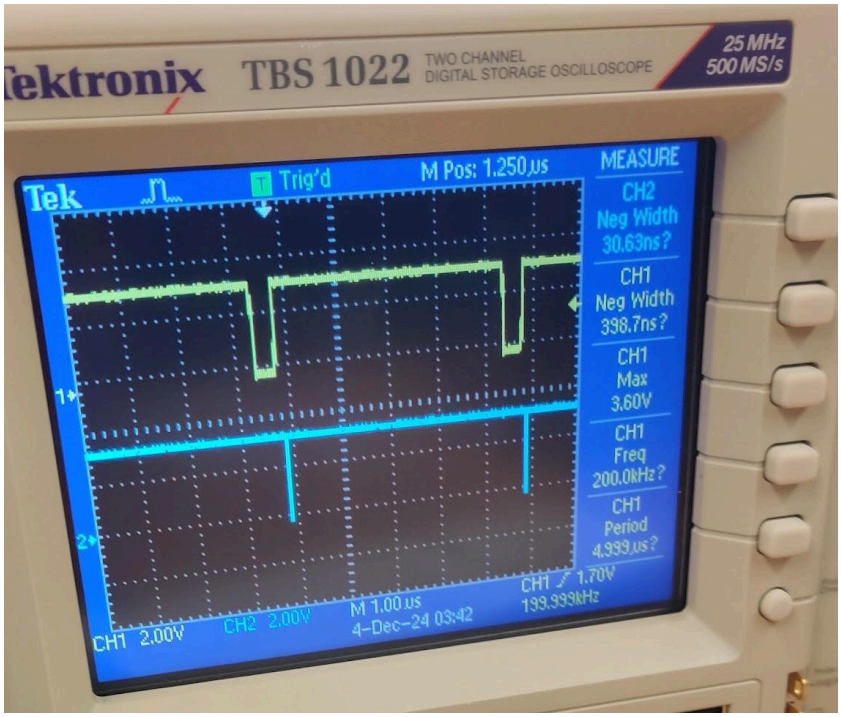
Experiment 1

The first experiment was to generate the timing signals on the Basys board matching the A/D chip matching the given timing diagram. Since the signals, convst, cs, and rd are the signals most important to the A/D converter, they are the ones that need to be timed correctly on the Basys board.

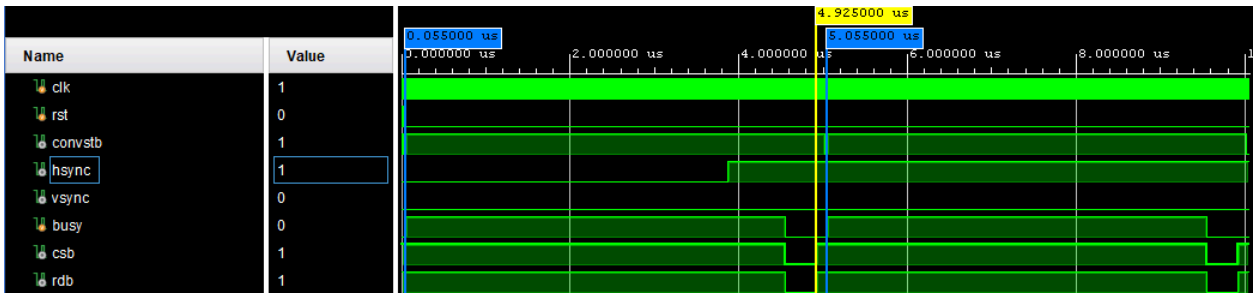
Originally we had planned to have multiple counters to account for the multiple timings, however we ultimately it more efficient to have a single counter and have the signals be active when the counter is within a certain range. One issue we ran across was the vagueness of the delays between certain signals. For example, there exists a delay between the cs and rd signals,

in which the minimum is 0 ns. Ultimately, we chose to follow the minimum delay of 0 ns between the rd and cs signals. Another issue we encountered determining how long cs and rd are in active low for, since the diagram does not explicitly state it. To solve this problem, we had simply set the signals to active low when the counter is between the end of t1 and the beginning of t8.

With the signals generated in verilog, we simulated in our testbench as well as the oscilloscope as shown below.



Channel 1: rdb
Channel 2: convstb



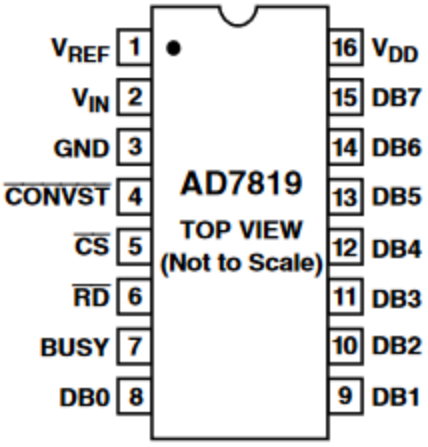
Experiment 2

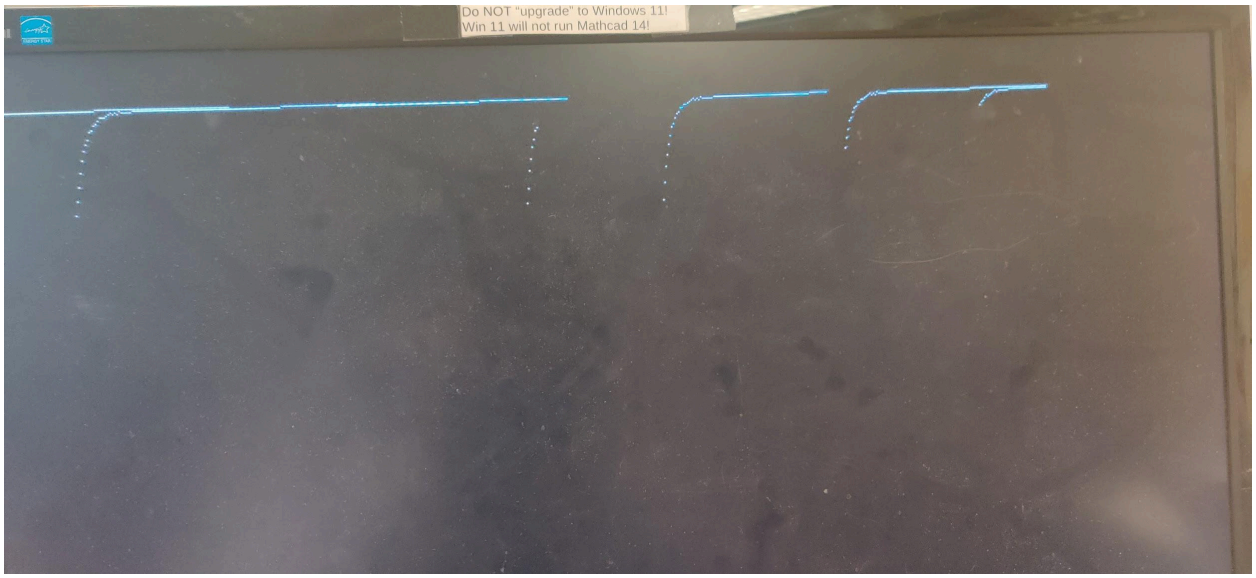
For experiment 2, we use the AD7819 A/D converter to generate an array of data busses from an AC source, like an arbitrary waveform generator. The data busses generated would then serve as inputs to the Basys board, which in correct timing, can write the data into a buffer, which then can be shown as a pixel using the VGA controller from the previous lab.

The buffer would reset itself to the beginning and continue the cycle over again and therefore generate a sine wave. We had troubles with displaying the lit up pixels on the screen based on the coordinates. The signals were almost always cut off at the end or at the top. The code had some troubles displaying the full sine wave; it had come up but wasn't able to display going down and had stuck itself to the up bar till it resets to do the same thing over again. This meant the problems we had lay in the buffer or signals, since only when the buffer matches the signal, it will light up the pixel on the screen but it will light up when it is on a horizontal line.

The solution we had was changing the rst to ~rst to somehow continue the wave till it goes down but it hadn't shown anything. The other we have tried out was adding more code to display more pixels intentionally making it show the down path of the sine wave but this didn't work and instead showed the sine wave vertically or no change at all. These were in if and else statements so we believe the set up was wrong somewhere in the if/else statements. We also ran tests using the actual oscilloscope to make sure the correct signals are being displayed.

PIN CONFIGURATION
DIP/SOIC





Conclusion

These experiments help us understand analog signals and how it works digitally. And despite our failure to produce an accurate waveform, the oscilloscope project provides a foundation for further enhancements, such as higher resolution displays, more advanced triggering, and

multi-channel signal acquisition. This approach showcases how FPGA-based systems can bridge the gap between analog signals and digital processing in modern electronics.