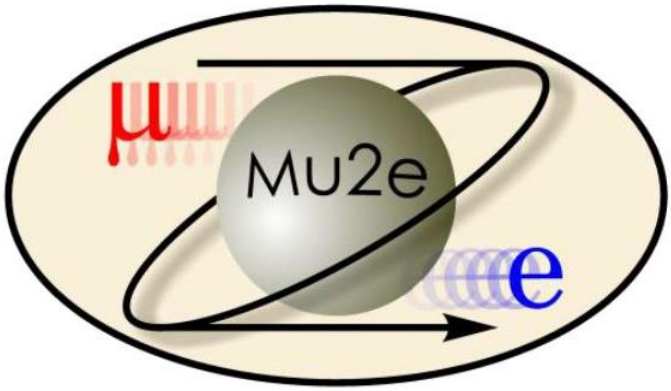
Digitization of Analog Signals using a Field Programmable Gate Array (FPGA)



Daniel Aguilera-Amaya

Illinois Institute of Technology

SIST 2017, Fermilab National Accelerator Laboratory

Supervised by Dr. Vadim Rusu

August 17, 2017

Abstract:

The Mu2e experiment is looking to detect a neutrino-less muon to electron conversion. The detection of such a rare event will hint towards previously unknown physics as such an event is dynamically suppressed by the Standard Model [1]. Detecting such an event is not easy and the experimental data must be digitized accurately to minimize error. Part of the digitization is being done with "of-the-shelf" analog-to-digital converters (ADC). To reduce cost and the number of components that could fail due to radiation exposure, it would be best to implement most if not all integrated circuits within the FPGA. This report explores the idea of designing and implementing an ADC within the FPGA.

Introduction:

Over the last half a decade or so, Mu2e has been developing a detector to detect a neutrino-less muon to electron conversion. Such an event would be evidence of Charged Lepton Flavor Violation (CLFV). The tracker is the device that will track any particle, including high energy electrons, and send out information to data acquisition. But how does the tracker do this?

To detect such events, the tracker uses metalized Mylar drift tubes (Anode), which are filled with a 20:80 mixture of Ar and CO_2 gas and have a high voltage, gold-plated, tungsten sense wire (Cathode) running through the center [2]. An example of the straws can be seen in figure 1. Every time a charged particle passes through a straw, the gas inside will ionize. Due to the electric field created by the anode and cathode, the negatively charged particles move towards the sense wire, creating an electron avalanche. This process generates a signal large enough to be differentiated from noise and be digitized.

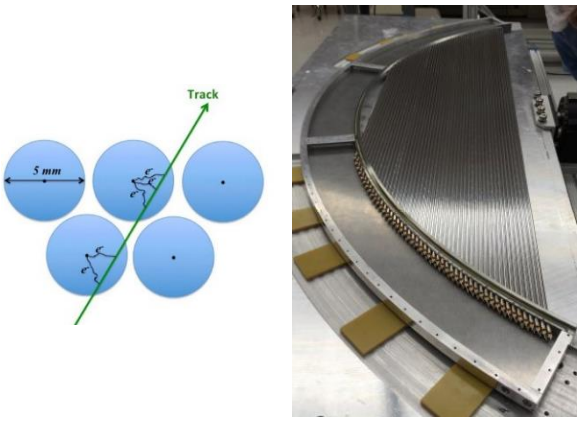


Figure 1: Side view of straws (Left) A full panel with 96 straws (Right) [2]

The digitization process starts with the preamps, which are located at both ends of the straw. Each preamplifier, or preamp, will shape and amplify the signal coming from the sense wire. From the preamps, the

signals go through distinct Schmidt triggers before going to the time-to-digital converters (TDC). The TDCs measure the time difference between arrival times of the signals with a time division resolution of 100psec [3]. The time difference is the used to estimate the position in which the event occurred at along the straw for pattern recognition. Also from the preamps, the signals are summed up and the resulting signal is fed into the input of an analog-to-digital converter (ADC). The ADC will digitize the amplitude of the analog signal to determine the change in energy as it passed through multiple straws [3]. Knowing the change in energy and its path, we can determine the particle's energy and determine what kind of particle it was. Once the analog signal has been digitized, it will be sent on to the Output Control & Buffer, Readout Controller (ROC), and lastly to Data Acquisition (DAQ). Figure 2 shows the system level layout of the digitization process.

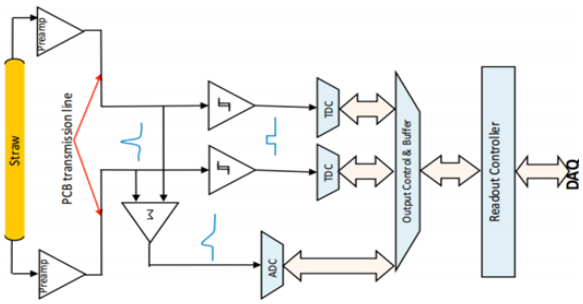


Figure 2: Digitization high level [3]

The current setup works well, but because the tracker will be in vacuum, all the electronics will be inside the tracker. Consequently, the electronics will be exposed to prolonged periods of radiation. Radiation can affect the programmability and even functionality of an integrated circuit (IC). To minimize the effects of the radiation two things can be done: changed failing components as needed or buy components that can withstand prolong exposure to radiation. However, buying an FPGA that can withstand large amounts of radiation and programming the "off-the-shelf" ICs into it can keep the cost down while minimizing the possibility of components failing. In the current setup, the ADC is the only "off-the-shelf" IC. By using low voltage differential signaling (LVDS), we can implement the ADC functions into the FPGA.

Materials:

The materials used for this research were all part of the samrtfusion2 starter kit. The kit used Microsemi's M2S050-FGG484 FPGA. To interact with the FPGA, the starter kit used Emcraft Systems' extension board which contained a breadboard area. To communicated with and program the FPGA, the kit came with a FlashPro4 serial programmer. The kit can be seen in figure 3.

Figure 3: FPGA & Extension Board (Top) FlashPro4 Programmer (Bottom)

In addition to the hardware, software was a big part of this research. To write the VHDL code, compile the code, and program the FPGA, it was necessary to use Microsemi's Libero v11.8 software. Libero made it simpler to program the FPGA as it takes care of placing and routing and allocating resources. A diagram of the initial system for sampling can be found in appendix A. Additional software used was ModelSim 5.4c and SoftConsole 4.0. ModelSim was used to run simulations to identify issues regarding synchronization and timing. Lastly, SoftConsole was used to run the code using the programmed FPGA to aid in debugging and to read information from RAM.

Methods:

To sample an analog signal, a digitizing scheme needs to be selected. For this project LVDS was chosen because it uses less power which means less heat generated. The way LVDS works is very simple; it has two inputs, one is the analog signal to be sampled and the other is the sampling clock. LVDS will compare if the sampling clock voltage level is below or above the analog signal. If its above, then LVDS will output logic low, and if it's below then LVDS will output logic high. Figure 4 shows LVDS in action.

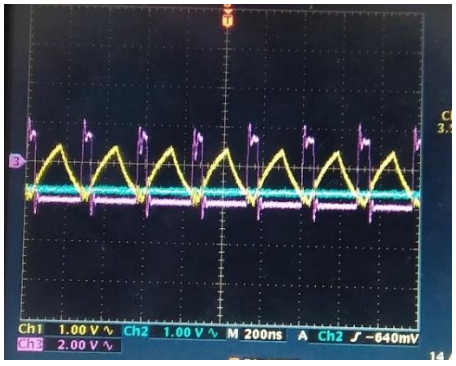


Figure 4: Sampling Clock Input (Yellow), Analog Signal Input (Blue), LVDS Output (Purple)

However, before LVDS can be used to sample anything, a sampling clock must be determined. Because the FPGA is a digital device, the clocks generated through it will be square-wave like. Unfortunately, this will not work with the LVDS scheme. To fix this

issue, the sampling clock will go through an RC filter that will turn the square-wave clock into a triangle-like wave. The values for the RC filter were chosen for a 3.5MHz sampling clock. The capacitance for the RC filter comes from the input and output pins. The resistor was chosen through trial and error to ensure that the voltage drop across the resistor was not too large and that the resulting sampling clock was as close to a triangle wave as possible. Figure 5 shows the before and after the sampling clock goes through the RC filter.

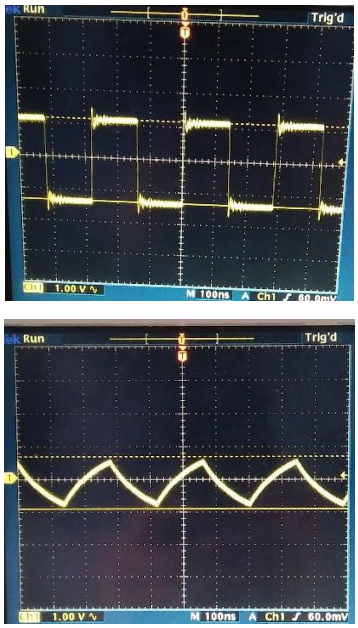


Figure 5: Sampling clk generated by the FPGA (Top) and sampling clk after the RC filter (Bottom)

With the appropriate sampling clock, the analog signal was sampled. However, the output of the LVDS at this point was still just high or low logic. To turn the output into useful information, it is necessary to sample the output of the LVDS. The sample or "count" being store to the random access memory (RAM) is synchronous to the sampling clock (3.5MHz). The secondary sampling clock will define the resolution of the ADC as the ratio (secondary/primary) of the clocks is the number of discrete voltage levels. For this setup, it the secondary clock was chosen to be 350MHz for a resolution of 100 discrete voltage levels. The 350MHz secondary sampling clock will look at the LVDS output and counts how many clocks cycles the output was high. The count stored to RAM will be a number between zero and 100. This process will repeat until the 9-bit RAM is fully populated with 512 samples.

Once the RAM is filled up with count samples, it is possible to read them out and turn them into voltage levels. However, to do this it is necessary to use a program called Termite, which allows for the serial readout of the data stored in the RAM. Using Termite and SoftConsole to run the program, samples were exported from the RAM into an excel spread sheet. From here the samples can be analyzed and turned into voltage values. The simplest way to do this is to take each count and turn into a ratio between the count and the highest possible count ($count/count_{Max}$).

To turn that into a voltage level, multiply the ratio with the peak-to-peak voltage. This will give us 2^9 voltage samples.

Results & Discussion:

With the samples now as voltages, it is possible to plot them in sequential order (sample 1 to sample 512). Figure 6 shows the resulting digitized waves for a couple different frequencies sine waves.

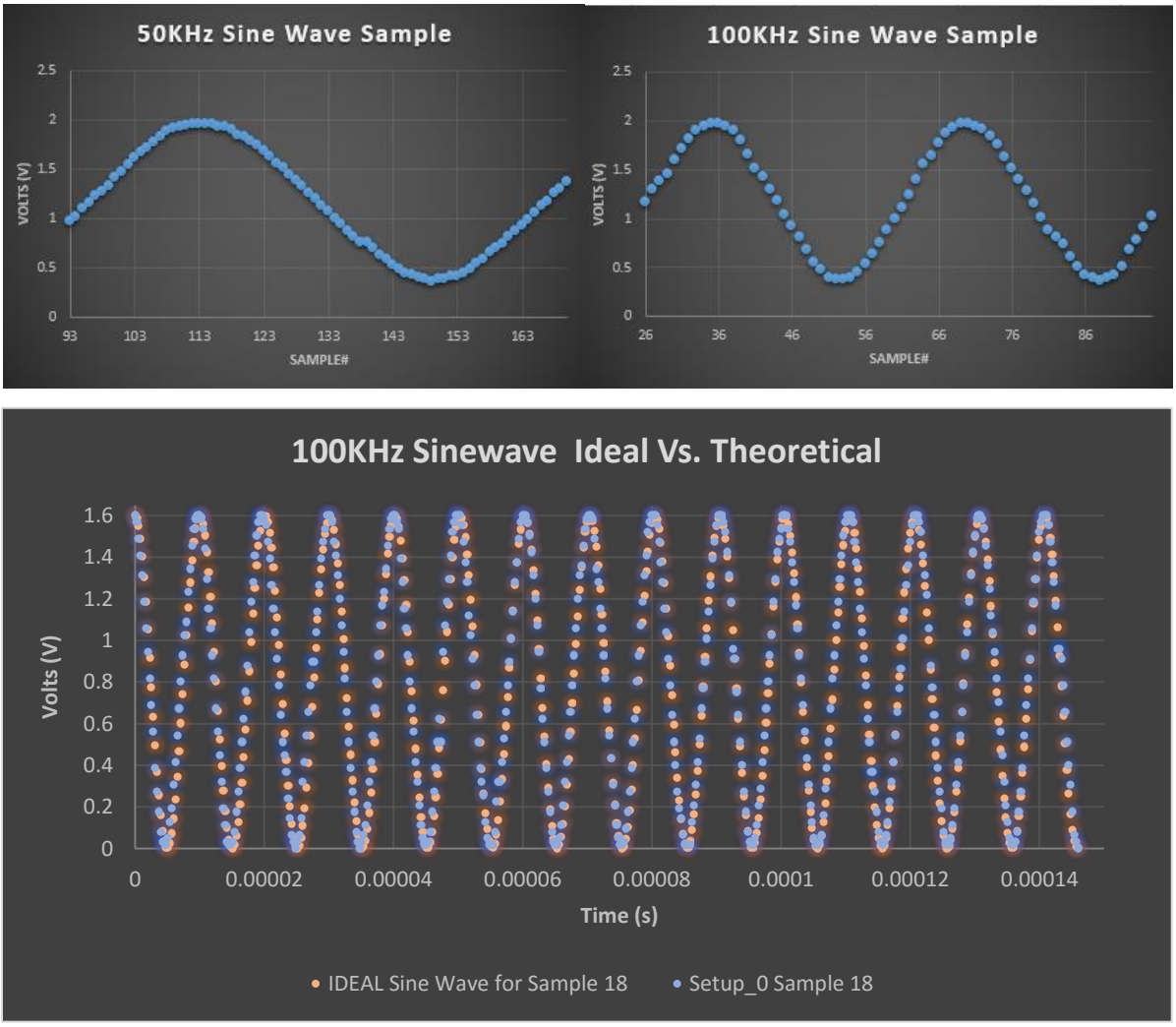


Figure 6: A period of a 50KHz sine wave sampled using FPGA (Top Left) Two periods of a 100KHz sine wave sampled using FPGA (Top Right) Sampled 100KHz sine wave on top of ideal sine wave (Bottom)

However, from these plots one can tell that the ADC set up is not perfect. The current set up has two main issues that need to be tackled: limitations on the analog signal's frequency and determining the performance of the ADC.

The first issue is that the current set up is limited to sampling analog signals with

frequencies lower than 350KHz without reducing the sampling resolution. The reason for this limitation is that the resolution is directly proportional to the ration of secondary sampling clock to primary sampling clock. This becomes an issue because the FPGA itself cannot produce a secondary sampling clock higher that 350MHz that could work properly.

One way that this issue can be tackled is to replace the limiting secondary clock with something that can determine how long the LVDS output was high level and when it was low level. The current set up has the secondary sampling clock keeping track of how many clock cycles the output is high. An improvement here would be to use TDC within the ADC. Instead of counting the how many clock cycles are high level, the TDC will time stamp every time it sees a rising edge and falling edge. With this information, it is easy to determine for how long the output of the LVDS was high level by looking at difference between a rising edge and falling edge. This will allow for sampling of signals with much higher frequencies. However, one downfall of this solution is that this set up uses two separate RAMs (one for rising edge and one for falling edge). A diagram of the TDC implementation is shown in appendix A.

The second issue that can be seen from the plots is that the samples can deviate from what the theoretical values should be. In figure 6, you can see the difference between the ideal sine wave (Orange) and the sampled sine wave (Blue). To determine whether these deviations are acceptable or not, it is necessary to quantify the performance of the implemented ADC. A common way to calculate the performance of an ADC is to determine its integral non-linearity and differential non-linearity. Integral non-linearity (INL) is the maximum vertical deviation of your sample from the ideal/theoretical signal. Differential non-linearity (DNL) in the other hand looks at the width of every "bin" and compares it the ideal width of an ideal ADC. The values determined from IDN and DNL are always in terms of least significant bit (LSB). This allows direct comparison of INL and DNL of different ADCs.

Determining the INL and DNL of the ADC implemented will help determined a few different things. One is that it will tell us how well this ADC compares to the of-the-shelf ADC that is being currently being used. Similarly, the INL and DNL can be used to determine the performance deviations as the code is changed and as the hardware setup changes.

Conclusion:

As a proof of concept implementing the ADC within the FPGA is possible. However, there are still improvements that need to be made to increase the resolution and accuracy of the ADC, while minimizing the number of resources used within the FPGA. At the moment, 0.18% of all D flip-flops, 0.22% of all look up tables, and 2.62% of all user input/outputs (I/O) are being used. Since there are two FPGAs per panel, the FPGA only needs to digitize signals from 48 straws, which means that the I/O usage percentage needs to be 2% or lower to cover all straws. In addition to minimizing the use of resources, the performance of the programmed ADC needs to be calculated to make sure that it is just as good as the of-the-shelf ADC before replacing it.

References:

[1] Glenzinski, Doug. (2016). *Mu2e for Physicists.* Fermi National Accelerator Laboratory Mu2e for Physicists. [Online] Available: http://mu2e.fnal.gov/public/index.shtml

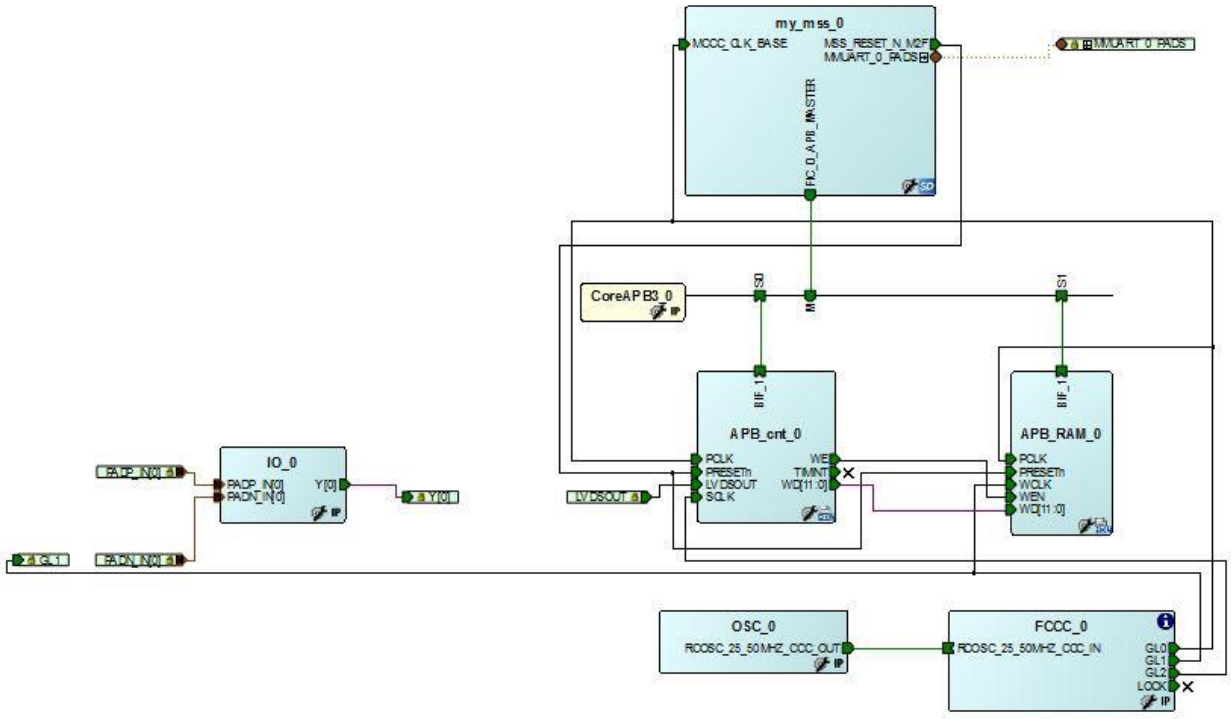
[2] Mukherjee, A. Wagner, B. *Tracker for the Mu2e Experiment at Fermilab*. Fermilab Wilson Hall, 08/16/2013

[3] Tommaso, Vincenzi. 2015. *Serial Communication for Mu2e Tracker Electronics*. Mu2e Collaboration, 15.

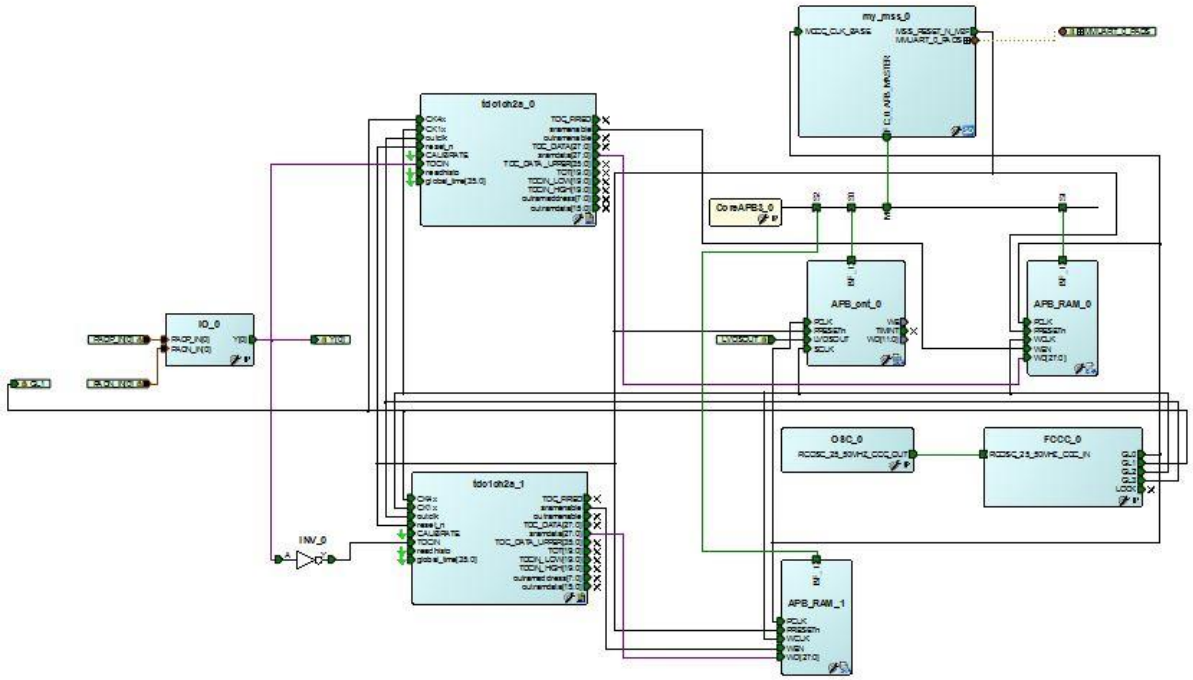
Acknowledgements:

In closing, I would like to thank a few people that helped me through the duration of this summer. First, thank you to my supervisor, Dr. Vadim Rusu, who took the time explain concepts to me, guide me, and push me to grow in and out of the work place. I would also like to thank my mentors, David Peterson and Donovan Tooke, for helping me prepare for presentations and talks. Lastly, a big thank you to the SIST committee for giving me this opportunity to learn and grow with Fermilab this summer.

Appendix A:



Initial set up to sample analog signals using LVDS and a secondary clock for the clock cycle count.



Proposed improvement to ADC by using TDCs for secondary sampling instead of using a secondary clock.