Developing a High Speed Data Acquisition System For Nuclear Physics Experiments Using FPGAs

JT Mills

Eastern Kentucky University, Richmond KY 40475

(Dated: August 11, 2022)

I. INTRODUCTION

The study of fundamental symmetries has become a prime focus for nuclear and particle physicists, as it provides a deeper understanding of the nature of the universe and its formation. For example, the matter-antimatter asymmetry of the early universe, also known as baryon asymmetry, remains an open question in cosmology, and the study of fundamental symmetries could provide the answer. In particular, the Neutron Optics Parity and Time Reversal Experiment (NOPTREX) collaboration is studying the violation of time-reversal symmetry. This symmetry violation is required by our current model of the Big Bang in order to explain why the universe formed as matter and not antimatter.

To investigate this asymmetry, the NOPTREX collaboration is using a low energy neutron beam to measure n-gamma resonances in heavy nuclei, which will be measured by an array of 24 sodium-iodide (NaI) scintillation gamma detectors (Figure 1)[1]. Therefore, this measurement will also require a high speed data aquisition system to process the large amounts of data from the array in real time. By using field programmable gate arrays (FPGAs) to digitize and filter the analog signals from the detectors, it is possible to implement custom built data analysis firmware based on the demands of the experiment.

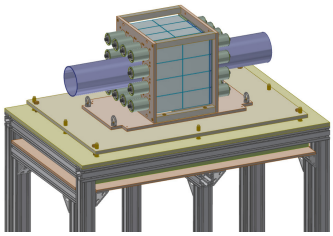


FIG. 1. The 24 NaI scintillation detector array for NOPTREX, being built at EKU.

In this research, I will use the CAEN SciCompiler software to develop a firmware for the DT5560SE open FPGA 32 channel digitizer that uses a trapezoidal filter and multichannel ana-

lyzer (MCA) to record the energy of each detector pulse, as well as charge integration to measure the detector current over a given time period. The block-coding style of the SciCompiler application simplifies the firmware design process, and will allow for a gradual introduction to computer logic. Subsequently, I will use the Vivado coding environment to develop firmware for the KRM-4ZU27DR development board. This board uses many high performance components to process high frequency digital signals, and will be used to development new methods of data acquisition for nuclear experiments. Upon the completion of this research, the DT5560SE digitizer will be used to acquire data for the NaI array being built at Eastern Kentucky University, and I will have gained a deeper understanding of computer logic and firmware development for FPGAs.

II. THE NAI(TL) ARRAY

As mentioned previously, the NOPTREX collaboration has developed an array of NaI(Tl) scintillation detectors to probe the symmetry of neutron interactions. These detectors use a sodium-iodide crystal that has been doped with thallium to detect nuclear radiation. When the radiation quanta are incident on the crystal's atomic structure, the interaction causes an excitation of the electrons within the thallium sites. Upon the de-excitation of these electrons, a photon of light in the near-visible spectrum is produced. This pulse is then funneled through the reflective of the crystal casing and into the photomultiplier tube (PMT) coupled to the crystal. This PMT is responsible for processing this pulse of light into retreiveable data.

The entry to the bottom of the PMT, known as the photocathode, s coated in a photoemissive material. By the photoelectric effect, the momentum of the incoming scintillated photon causes a certain number of photoelectrons to be released by the photocathode. Because the tube is held at high voltage, these free electrons are then accelerated up the tube and collide with a series of high voltage dynodes. This produces a cascade of electrons that eventually reach the anode of the PMT, and carry enough energy to form a detectable current. This signal is then received by the electronics and ultimately processed by the data acquisition system.

The detectors for the NOPTREX experiment use a custom made electronics board for voltage division between the dynodes and signal processing. This board allows for operation in either pulse or current modes, based on the demands of the experiment. This makes the array significantly more adaptable for later experiments. They are also contained within a mu-metal magnetic shield that has been arranged for maximum protection from exterior magnetic field interference and a spring-loaded housing assembly to ensure a light-tight enevironment.

III. FIELD PROGRAMMABLE GATE ARRAYS

Field programmable gate arrays (FPGAs) differ from conventional CPUs in their ability to have their architecture rearranged based on the demands of a specific task. The architecture of an FPGA consists of a system of configurable logic blocks (CLBs) made up of look-up tables (LUTs) to define the logic structure and flip-flops to store data. In addition to the CLBs, digital signal processor (DSP) slices are also incorporated to handle more complex tasks such as addition, multiplication, and accumulation. These components are connected by interconnects, which serve as pathways for the signals to travel through. By combining these systems with I/O ports and RAM memory, it is possible to create a fully customizable FPGA that can be reprogrammed on the fly using a hardware development language (HDL). While the use of a programming language may cause FPGA design to seem similar to computer software coding, it is actually much different as an FPGA requires the definition of the specific arrangement of its hardware. The adaptability of FPGAs make them a viable candidate for high speed data acquisition, as they are capable of efficiently processing data in real time and thereby reduce the amount of post-recording data analysis for the experiment.

A. DT5560SE

To manage the DAQ system for the NOPTREX NaI detector array, we will use the CAEN DT5560SE open FPGA digitizer featuring 32 analog input channels, 14-bit ADC, and 125 MS/s processing capabilities (Figure 2)[2]. The board also features 6 extra digital I/O ports, two optical link ports, and two sync ports which be used to synchronize the internal clock with that of another board. The board can be connected to a PC using either a USB or Ethernet connection, allowing access to the board settings, analog front-end settings, and the FPGA flashed firmware. The high processing power of this device in conjunction with the customizability of the FPGA will allow for high speed, real-time signal processing for each of the 24 detectors.

The board defaulted with a precompiled multichannel pulse height analysis firmware that included an oscilloscope and energy spectrum. This default firmware allowed for each of the 32 analog channels to collect and process data. In addition, the package included the open-source DT55xx readout software to allow for a graphical user interface with the firmware. The readout software displays the oscilloscope output, energy spectrum, and access to the signal processing and analog front-end settings. Also, it allows for statistical analysis of the energy spectrum. However, in order to customize the signal processing methods to include charge integration triggered by an external

FIG. 2. The CAEN DT5560SE 32 channel Open FPGA digitizer.

signal, it will be necessary to use the CAEN SciCompiler software to develop a new firmware for the board, and subsequently the CAEN software development kit to create a software capable of interfacing with the firmware.

IV. CHARGE INTEGRATION FIRMWARE DESIGN WITH CAEN SCICOMPILER

A. SciCompiler

The SciCompiler software, developed by Nuclear Instruments in conjunction with CAEN, allows for high level block coding of the firmware for the DT5560SE digitizer and the Xilinx FPGA. Rather than starting from scratch using HDL code, this software makes the firmware development process more approachable by allowing for the use of premade digital components for a wide variety of nuclear physics applications. It also includes registers for reading or writing data, digital or analog I/O ports, and a collection of digital logic and timing components. The software's ease of access and compatibility with the digitizer makes it the obvious choice for the development of the NOPTREX NaI array DAQ system.

In order to use the full functionality of the SciCompiler software, it is necessary to install the Xilinx Vivado software and Microsoft Visual Studios. Vivado can be used to develop FPGA firmware and produce bitstreams with Vivado hardware development language, or VDHL. It allows for the entire prototyping, design, debugging, and construction process within the application. When a firmware is developed and compiled in SciCompiler, the SciCompiler software automatically connects to the Vivado application to construct the design, compile the firmware, and produce the optimal bitstream for the FPGA to execute the customized design.

In addition to producing the firmware, SciCompiler also creates a software development kit based on the firmware requirements in order to assist with the software development process. This

kit includes libraries for both Python and Visual C++ that can be opened in Microsoft Visual Studios, as well as an example program to demonstrate the connection to the board. By using these libraries and examples, it is possible to design a software application capable of interfacing with the components included in the compiled firmware. In short, the SciCompiler software provides all the necessary components to make the FPGA firmware development process accessible to those at the introductory level.

B. Charge Integration

In conventional nuclear detection, there are two primary methods for operating a detector: pulse or current mode[3]. In pulse mode, the detector utilizes a higher gain and possibly a preamplifier to analyze individual pulses of radiation. This application is useful for cases in which the energy or timing of the events is needed. On the other hand, current mode provides an average of the voltage produced by the radiation of a given time. This method is useful for applications in which the count rate is very high and the pileup rate would make counting pulses more difficult. While Both of these methods exhibit their own strengths and weaknesses for different applications, for the purposes of this experiment it will be best to utilize the current mode operation. This mode naturally lends itself to the charge integration method of signal analysis, as it also provides an average of the activity over a given time period. This removes the need for complicated filters and excessive memory usage for real-time analysis.

For the NOPTREX DAQ system, I have developed a charge integration firmware for processing the signals (Figure 3). The SciCompiler charge integration module takes a trigger signal and an integration time input and measures the area under the input signal over the given integration time, which can be set by a register. The module outputs a single value for the area for each integration window. It also features a pileup inhibitor setting and a baseline subtraction, which can also be set by registers. This module has been implemented into the various iterations of firmware that I have created throughout this project.

In the preliminary firmware, I used a leading edge trigger module to trigger on the analog detector signal. By setting the trigger threshold using a register, the leading edge trigger module outputs a digital signal to the charge integration module for every pulse that exceeds the threshold. However, this design lead to the integration being activated rapidly and for very short periods of time, which produced a low amount of area counts and statistically unsatisfying data (Figure 4). In order to configure the firmware to match the experimental requirements for NOPTREX, I

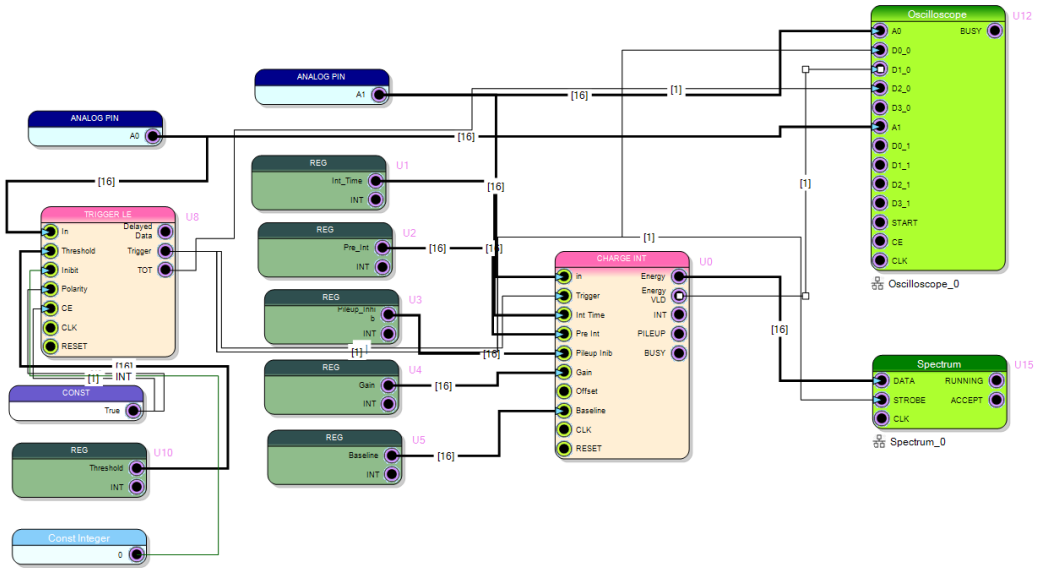


FIG. 3. The SciCompiler block diagram for the charge integration firmware.

developed more versions with different trigger settings.

A later version of the firmware featured an external trigger signal consisting of a rectangular pulse produced by a signal generator. This low frequency external pulse activated the leading edge trigger module, which then enabled the charge integration. The integration time was set to the period of the external trigger pulse. This lead to better results, since it required fewer integration windows and each window measured a greater area (Figure 5).

V. FIRMWARE FOR THE NOPTREX EXPERIMENT

To adapt this initial charge integration firmware to fit the goals of the NOPTREX experiment, a new iteration needed to be made (Figure 6). In this version, the charge integration module is triggered by a 20Hz external signal, which in the case of the NOPTREX experiment at the Los

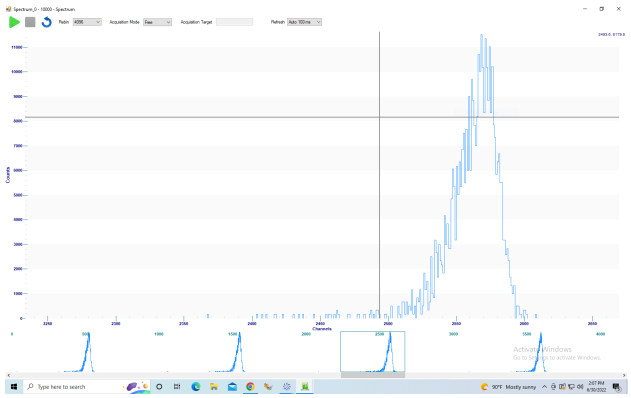


FIG. 4. The results of a Cesium 137 spectrum using the first firmware iteration.

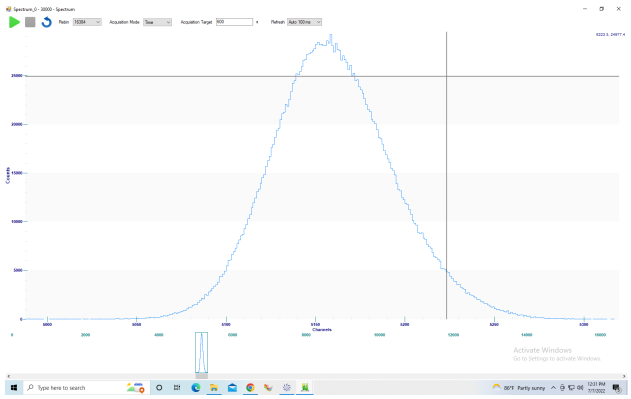


FIG. 5. The results of a Cesium 137 spectrum using the second firmware iteration.

Alamos Lab Neutron Spallation Source will come from the proton accelerator that produces the neutron events. This signal is delivered to the rising edge trigger module which subsequently triggers a pulse generator to produce an internal signal at a frequency that can be set by a register. This signal causes the charge integration module to perform periodic integrations over the period of this signal until the next external signal is received. Each of the results from these integration windows is then saved to a list module, which produces a downloadable single-column text file containing each integration result in a separate row. This feature will simplify the software development process, as the program will just have to read the contents of the list module for each external signal period and write those contents to a text file.

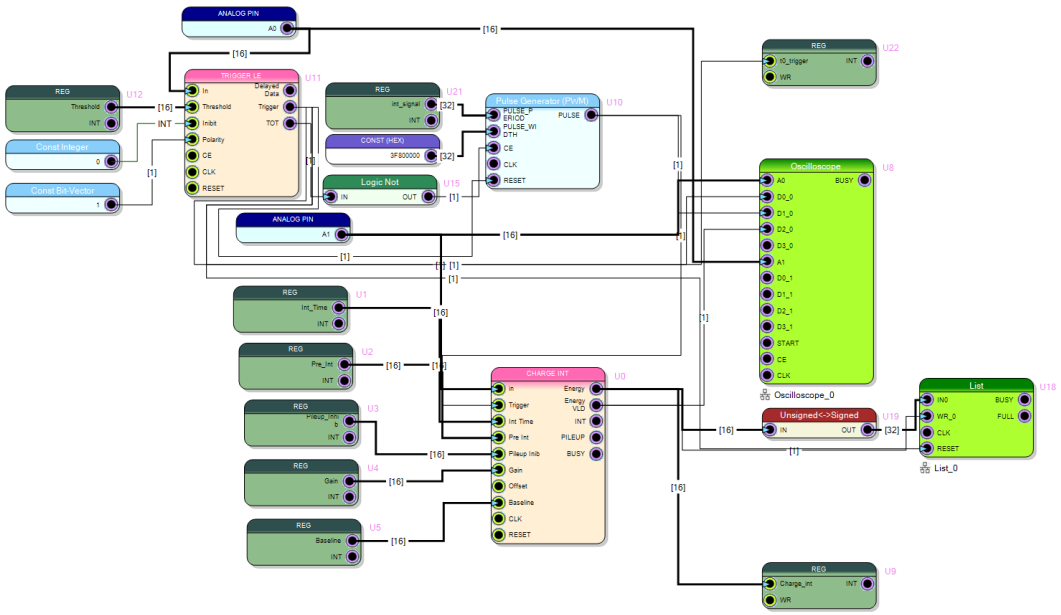


FIG. 6. The second iteration of the charge integration firmware.

VI. FIRMWARE TESTING

To test the functionality of the firmware design, I used the SciCompiler resource explorer application in place of a dedicated readout software to evaluate the output. he resource explorer allows the user the view the output of the oscilloscope module, as well as download the contents of the list module to a text file. In order to diagnose the operation of he firmware throughout the development process, I connected various digital signals to display on the oscilloscope output. These included the main trigger signal from the 20Hz external signal and leading edge trigger module, the status of the charge integration trigger signal generator, and the signal notifying that a charge integration result had been produced. These signals, along with the raw analog input from the detector and the 20Hz input signal, made up the oscilloscope output. This view allowed me to troubleshoot the issues with the firmware during the development process.

Following the initial troubleshooting measures, I began the main testing phase for the firmware. To do this, I used an oscilloscope with a waveform generator tool to feed waveforms into the firmware to mimic the data received during an experiment. I used this method rather than a simple detector and radioactive source setup because a radioactive source produces a uniform rate of activity over time. This means that the charge integral is constant, and it doesn't prove the ability of the firmware to discover anomalies in the data, such as a nuclear resonance. For this

reason, I used previously recorded data from a prior NOPTREX experiment to test the ability of the firmware to adequately process the signals in real-time and reproduce the results of the experiment.

For the purposes of this test, I used the raw data recorded for the NOPTREX Double Lanthanum Experiment at Los Alamos Lab Neutron Spallation Source. This data demonstrates the gamma emission of a Lanthanum target bombarded by polarized, low-energy neutrons. The goal of this experiment is to use the nuclear resonances apparent in the data to reveal parity violating asymmetries in the neutron-nuclei interactions. Therefore, by processing this raw data using the charge integration firmware and attempting to reproduce the nuclear resonance data, we can support the efficacy of the firmware design.

With the raw data contained in a ROOT file, I was able to use the Python Uproot module to extract the waveform data for one neutron event and convert the binary file to a list of integers. This file included the voltage yield of the detectors for each neutron event within a TTree. After extracting the data and converting to integer form, I was able to arrange the comma separated values file to fit the requirements of the oscilloscope. I then used the oscilloscope to feed the neutron event waveform (Figure 7) to the digitizer. I then collected multiple runs of the charge integration firmware to evaluate its performance. As expected, I found that longer integration windows resulted in a lower resolution for the final waveform, while shorter integration windows resulted in a waveform with a resolution closer to that of the original waveform (Figure 8).

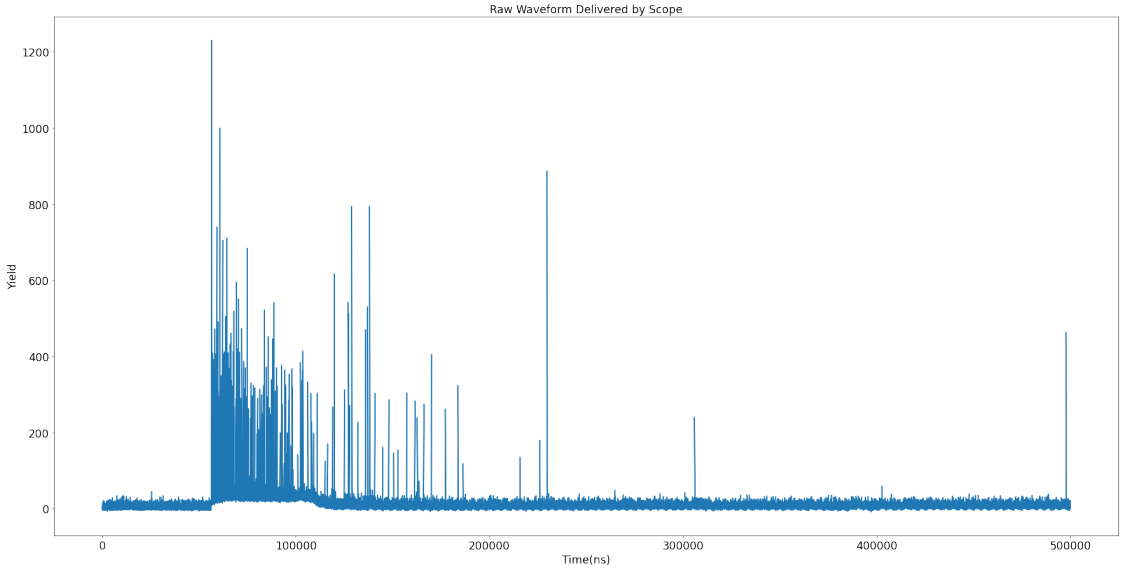
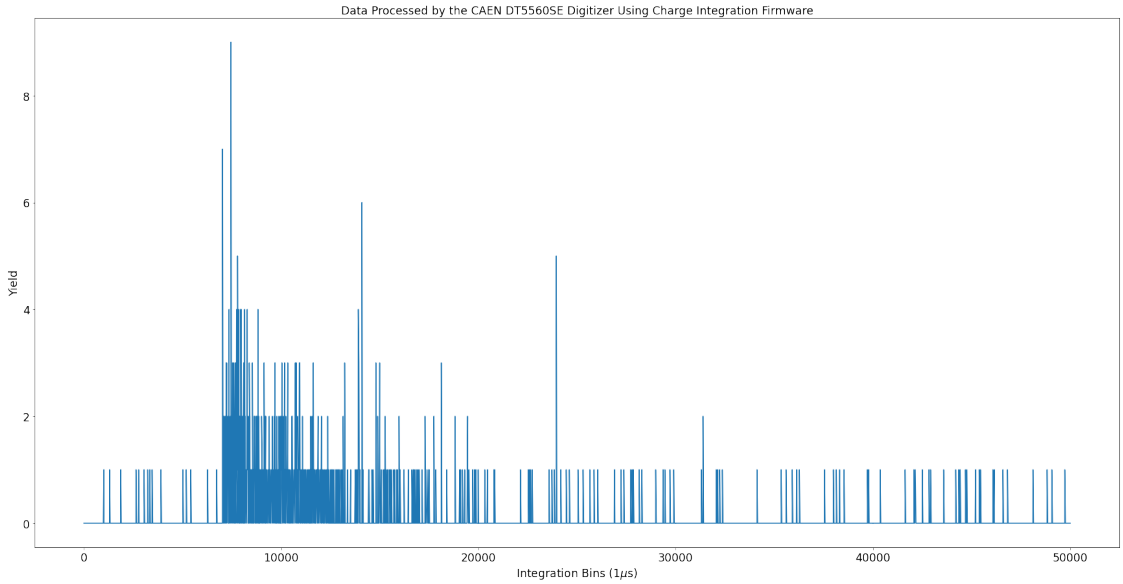


FIG. 7. The original waveform from the raw Double Lanthanum data.



[h]

FIG. 8. The resulting waveform from processing the Double Lanthanum waveform with the charge integration firmware.

VII. FUTURE TASKS

The original Double Lanthanum data analysis used a pulse mode approach to characterize the individual peaks of each neutron event waveform, and used these peaks from every neutron event combined to create a time-of-flight histogram and an energy histogram[4]. This time-of-flight histogram shows the variations in gamma intensity based on the different energies of the incoming neutrons. Since the higher energy neutrons will arrive at the target more quickly than those with lower energy, the resulting gamma intensity over time can reveal the nuclear resonances in the target based on the energy of the incident neutrons. For the ultimate test of the abilities of the charge integration firmware, it will be necessary to recreate this time-of-flight histogram and compare it to that of the original analysis to evaluate the ability of the firmware to discover nuclear resonances. This will require delivering a series of different neutron event waveforms to the digitizer rather than a single event. Because each event contains roughly 500,000 data points, this series of events encompasses a huge amount of data. The next important task in conditioning the firmware will be to compile this huge data file and characterize it to fit the requirements of the oscilloscope waveform generator.

To complement this final test, it is also necessary to develop a functioning readout software for the charge integration firmware. This is because the chain of neutron events will produce a list of

integration results for each event, and a software is required to separate these lists and compile them accordingly. This software will need to write to the read registers in the firmware for the settings of the charge integration and trigger modules. It must also read the data from the list module and the oscilloscope output. When an external 20Hz trigger is received, the software should save the previous list to a new branch of a ROOT binary file, reset the list, and begin the integration once again. This can all be simplified using the software development kit provided by the SciCompiler software. Upon compilation of a firmware, SciCompiler creates a set of register files, as well as libraries to demonstrate the commands used to communicate with the digitizer. Future work will focus on developing and testing this software, and subsequently using it to perform the final test on the firmware.

VIII. REFERENCES

[1]: Jak Doskow. Indiana University Bloomington.

[2]: https://www.caen.it/

[3]: G. F. Knoll. Radiation Detection and Measurement, Fourth Edition. 2010.

[4]: David Matthews. University of Kentucky