A Scalable FPGA-based Multi-channel Data Acquisition System

for Parallel Plate Ionization Chamber

by

Rafael Andres Acuna Briceno

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2021 by the
Graduate Supervisory Committee:

Hugh Barnaby, Chair
David Blyth
John Brunhaver

ARIZONA STATE UNIVERSITY

December 2021

ABSTRACT

Proton beam therapy has been proven to be effective for cancer treatment. Protons allow for complete energy deposition to occur inside patients, rendering this a superior treatment compared to other types of radiotherapy based on photons or electrons. This same characteristic makes quality assurance critical driving the need for detectors capable of direct beam positioning and fluence measurement.

This work showcases a flexible and scalable data acquisition system for a multi-channel and segmented readout parallel plate ionization chamber instrument for proton beam fluence and positioning detection. Utilizing readily available, modern, off-the-shelf hardware components, including an FPGA with an embedded CPU in the same package, a data acquisition system for the detector was designed. The undemanding detector signal bandwidth allows the absence of ASICs and their associated costs and lead times in the system.

The data acquisition system is showcased experimentally for a 96-readout channel detector demonstrating sub millisecond beam characteristics and beam reconstruction. The system demonstrated scalability up to 1064-readout channels, the limiting factor being FPGA I/O availability as well as amplification and sampling power consumption.

ACKNOWLEDGMENTS

To my family and friends. Thank you for all the support and encouragement you have provided across the years. I could not have done it with out you.

To the members of my thesis committee. Professor Barnaby, thank you for encouraging me to pursue graduate school and for your support these past years. Professor Brunhaver, I will forever be grateful for introducing me to digital design and your teachings on the subject. Dr. Blyth, I appreciate your mentorship and time spent together developing the work described in this thesis.

To Dr. Evgeny Galyaev, for giving me the chance to participate in this project and trusting me with the contributions about to be explored. Furthermore, thank you for providing support material used in this document.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Nuclear Physics program office under Award Number DE-SC0015136.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Cancer as a disease has a major impact on society across the world. Techniques and methods to treat it have been developed with a varying degrees of success which include surgery, chemotherapy, immunotherapy, ultrasound therapy, and radiotherapy as the most common treatments. Radiation therapy is based on the idea of damaging cancerous tissue through exposure to a radiation beam, while healthy neighboring tissue regenerates and repairs faster. In practice, techniques involving photon and electron beams have been proven successful with proper radiation dose to a target volume of cancerous cells [1].

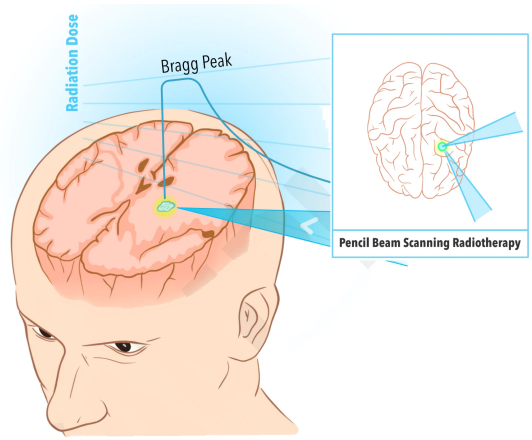


Figure 1. Protons release most of their energy shortly before the end of their path in matter. The sharp peak on the plotted linear energy transfer curve depicts the Bragg Peak.

Protons have proven to have an advantage in cancer radiation therapy due to the particles' energy loss characteristics in tissue. Protons path in matter is virtually straight with velocity dependent energy loss and therefore most of the energy deposition occurs right until the end of their trajectory. This phenomenon is defined as Bragg peak [2]. The advantage is clearly

depicted in Fig. 1, where radiation dose can be delivered to the volume while minimizing damage to bystander tissue. Since protons completely stop and deposit their energy inside the patient, the transverse beam profile, intensity, and depth are all critical features that must be directly measured. This also implies that treatment must be thoroughly planned to minimize potential damage. The verification of beam characteristics for treatment purposes is also known as quality assurance or QA.

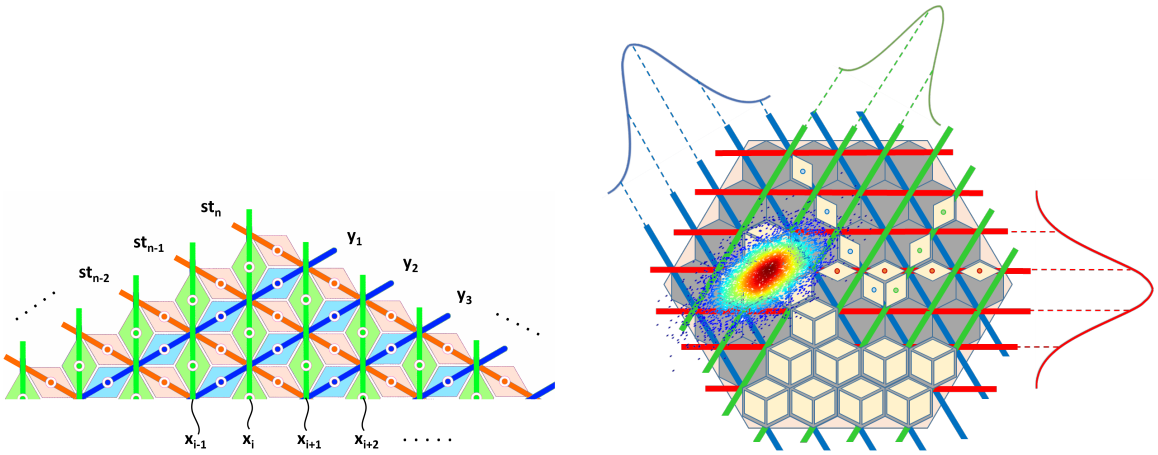
This critical application drives the need for detectors capable of direct proton beam measurement. Detectors which require high spatial and temporal resolution, wide dynamic range, as well as good linearity. The main goal is to enable proper characterization of X-Y beam position as well as provide accurate fluence measurements.

The application and its corresponding requirements present the need for a data acquisition (DAQ) system that is compliant to the detector's specifications. This thesis presents a scalable DAQ system leveraging off-the-shelf modern components, and therefore abstaining from using more expensive and less available application specific integrated circuits (ASICs). By using off-the-shelf components mounted in a standard PCB, manufacturing lead times are significantly reduced.

1.1 Detector Description

The detector is a parallel plate ionization chamber to be used as a planar proton beam fluence detector for quality assurance (QA) in cancer treatment. QA aims to ensure safe and accurate delivery of proton radiation dose to patients, while minimizing damage to healthy tissue. This detector provides sub-millimeter planar spatial resolution, as well as a temporal resolution of 25k frames per second (fps) for superior QA.

The principle of operation is briefly explained as follows. Through the application of a DC voltage bias between the detector parallel plates an electric field is created inside the chamber. Through proton beam exposure (a source of ionizing radiation) ion pairs are formed between the plates. The electric field is strong enough to reduce recombination of the ion pairs and induce the collection of electrons. This produces an output ion current, effectively turning the parallel plates into a cathode and an anode. The amount of current is dependent on the proton beam energy.



(a) Output strips arrangement (b) Beam projections

Figure 2. Anode composed of collector pads connected to three sets of output strips normal to the beam separated 120 degrees from one another; 2a. The arrangement enables the readout of three beam projections which are used for beam reconstruction; 2b.

The anode is made of copper collector pads which are fabricated with readily available standard Printed Circuit Board (PCB) manufacturing. Copper traces organized as strips are routed and connected to the collector pads with a PCB via. As depicted in Fig. 2a, three groups of strips are routed below the ionization chamber's anode, each in a different direction normal to the beam on a different PCB layer. The three sets of strips correspond to the three resulting signal intensity projections of the originating radiation beam, where each projection is a one-dimensional perspective of the beam as shown in Fig. 2b. Each output strip carrying an

ion current is fed into to the data acquisition (DAQ) system. Using the three projections it is possible to perform the reconstruction of the particle beam since the pad array and output strip arrangement are predefined and well known [3]. This idea is depicted in Fig. 2b. Image reconstruction from its projections is also referred to as tomography. Details on the reconstruction approach are based on the assumption that the proton beam flux distributes as a covariant 2-D Gaussian as described on the following work [4].

Due to the nature of the detector and its principle of operation, the frequency components composing the output signals are on the low end. Therefore, the bandwidth and sampling requirements of the DAQ system are relatively undemanding. The analog circuits should support 10kHz of signal bandwidth. The critical aspect of the system is the dynamic range as well as performing the sampling of all outputs simultaneously. Dynamic range can be defined as the range between the noise floor and the maximum output [5]. Maximizing dynamic range enables the instrument to measure beam characteristics in better detail as well as increasing the instrument flexibility.

1.2 Thesis Statement

The need for superior cancer treatment QA in proton beam therapy drives urgency for better instruments to measure accurate X-Y beam positioning and radiation beam fluence profile. A scalable and flexible DAQ system built with off-the-shelf modern components is of interest in order to avoid ASIC related lead times and costs. Additionally, the system should not only accommodate to a single detector, but rather support detectors of similar construction with different output channels and different output current ranges. Scalability and a high dynamic range of the DAQ system address these challenges.

1.3 Contribution

The contributions focus on the hardware design related to the DAQ system. This includes Printed Circuit Board (PCB) design of the PCB which digitizes detector outputs via an array of Transimpedance Amplifiers (TIAs) and Analog to Digital Converters (ADCs). Moreover, a Field Programmable Gate Array (FPGA) design that serves as the intermediary between ADCs and the software infrastructure was designed, implemented, tested, and deployed for a 96-channel detector prototype. Details about the software infrastructure are mentioned in order to provide context since the information is pertinent to some FPGA design choices.

1.4 Thesis Structure

The structure of this document is as follows. Both Chapter 2 and 3 correspond to the DAQ system. Chapter 2 focuses on the detector level context providing details on each block and how they contribute to the overall system goals. Furthermore Chapter 3 focuses on the FPGA level design. Even though it is only one part of the system, special attention is given since the device orchestrates the sampling of the detector's outputs with the software infrastructure. Chapter 4 elaborates on how the FPGA system scales with incrementing readout channels as well as providing experimental data. Experimental data showcases spatial and temporal resolution as well as detector linearity.

Chapter 2

DATA ACQUISITION SYSTEM

The data acquisition system is composed by all the elements which together, collect output data from the detector. In order to understand how each piece contributes and its context, the output of the system is explored first, then each element in the system is explored from input to output or from the detector output currents to a digitized data stream.

2.1 Streaming Readout

As mentioned in Chapter 1 the detector supports 25kfps implying a constant readout rate. This categorizes the data acquisition as a streaming readout system. Classic Data Acquisition Systems are triggered-based, therefore they rely on a hardware signal, known as the trigger, to perform data collection. Modern hardware enables a constant sampling rate and moves hardware challenges into software. For example, with a constant sampling rate, noise suppression can be taken care of in software with an easy to implement digital low pass filter. The main drawback of a streaming readout system is the inherent increase in data throughput. The solution to alleviate this concern, is explained in the following Section 2.1.1. Additionally, streaming readout systems consume all generated data as it is constantly being produced. Data can be consumed by storing the data or by performing operations on the data stream. This enriches the system by enabling software tools for processing, manipulation, display, and analysis of the stream on the fly.

As depicted in Fig. 3, the main hardware elements of the DAQ are Front-End Modules, which feature an array of transimpedance amplifiers as well as analog to digital converters, fol-

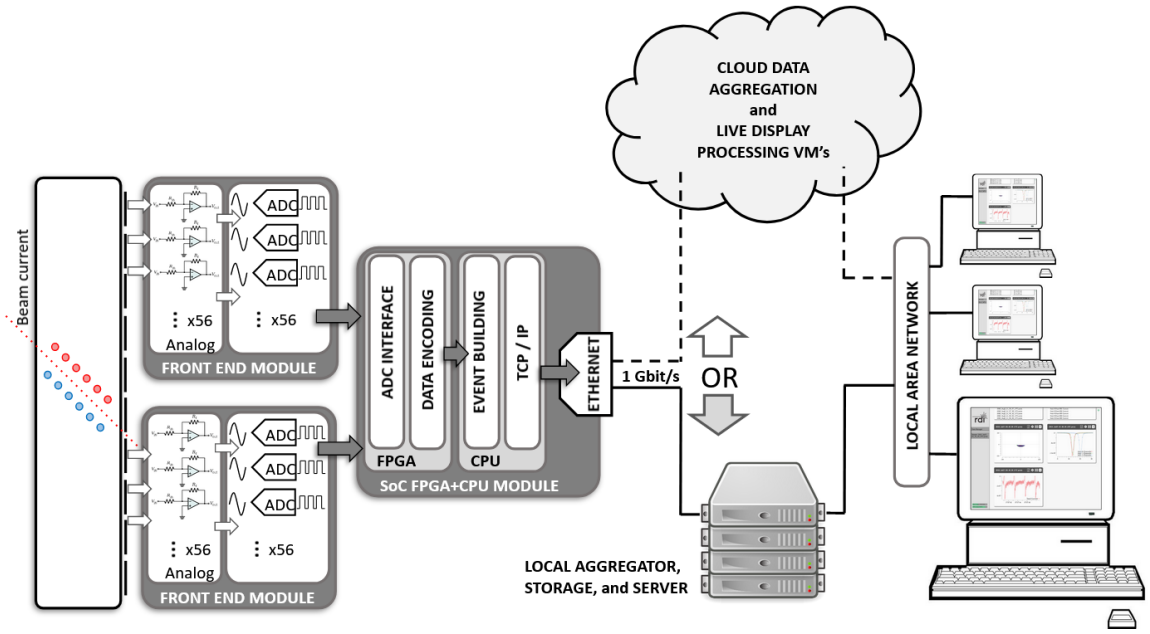


Figure 3. Data acquisition system diagram. Detector output currents are amplified and digitized on the Front-End Modules while the SoC FPGA Module interfaces with the ADC array and outputs a ProIO event stream through ethernet to the aggregator server. The modules are separate PCB boards connected to the same carrier board.

lowed by a System on Chip (SoC) FPGA which outputs a data stream in ProIO format [6] over the network into a data aggregator server. The data stream can be manipulated (e.g. filtered) on the server where extensive computing resources are available. ProIO and the reasoning why it was chosen for the data stream format will be explained shortly.

2.1.1 Streaming Readout Output Format

As previously mentioned, the main drawback of a streaming readout system is a higher memory bandwidth since data produced by the system increases. For a detector featuring 336 output readout channels, the memory bandwidth needed is $\sim$270Mbps when the sampling rate is set to be 25kSps. Ethernet supports the throughput, but it now produces a disk bandwidth concern. LEB128 encoding is performed on the data stream in order to reduce data throughput.

LEB128, which stands for Little Endian Base 128, is a binary encoding scheme used in the DWARF file format, WebAssembly, and Protocol Buffers [7]–[9]. The scheme enables the storage of a binary number in a lesser number of bytes. Each encoded byte is composed of seven data bits and one continuation flag bit. Fig. 4 conveys how an a 32-bit integer can be stored in three bytes or 24 bits, effectively reducing memory allocation. In a worst-case scenario, a very large LEB encoded 32-bit integer would require five bytes of memory, but this cases are rare.

```
msb-------------------------lsb
00000000000001011001001000001101  365069 in 32 bit integer binary format
     0010110   0100100   0001101  Split into 7 bit groups, disregard empty groups
    00100110  10001110  11100101  Concatenate continuation flags
       0x26      0x8E      0xE5  LEB128 int in hex format

          2         1         0  LSB to MSB in memory
```

Figure 4. 32-bit integer LEB128 encoding example depicting memory allocation reduction from 4 bytes to 3 bytes.

In the proton beam therapy context, it is reasonable to assume that proton flux measurements are distributed as a covariant 2-D Gaussian when integrated in the order of milliseconds [10], [11]. This idea is the main approach to perform tomography on the proton beam as depicted in Fig. 2b. Furthermore, the 2-D Gaussian shape of the beam is also seen in experimental data as shown in Fig. 5 through the three beam projections. Under this assumption, one can conclude that a significant amount of detector output values are small. These small values correspond to the detector area not being irradiated at a particular moment. Therefore, considerable data transfer throughput is saved if LEB128 encoding is performed on all the detector output values.

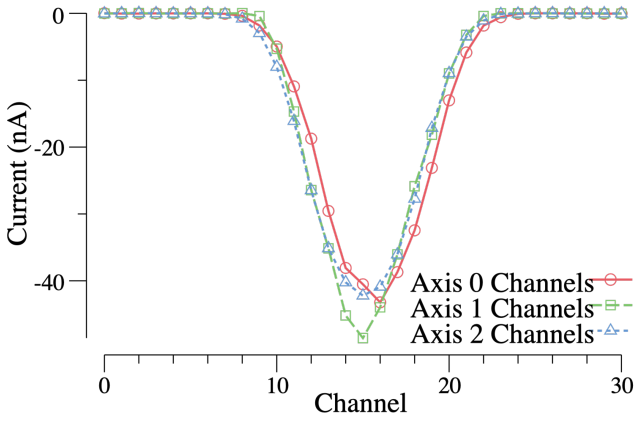


Figure 5. Experimental proton beam projections in one recorded frame from a 96-channel detector prototype. A considerable number of channels show small measurements while the detector is irradiated. Each axis corresponds to a beam projection.

LEB128 is used not only due to the transfer bandwidth reduction, but also because Protocol Buffers Varints are LEB128 encoded. *Varint* refers to a variable length integer in Protocol Buffer terminology. The Varint term will be used for the rest of this document.

Protocol Buffers or Protobufs for short, are Google's software mechanism to serialize structured data, its free and open source [12]. Protobufs are efficient for both data transfer as well as storage in a language-neutral manner. The way Protobufs are created is flexible and powerful. User defined data structures, called messages, are described in a definition file with the *.proto* file extension. The *protoc* command line tool takes the *.proto* file and the desired programming language as arguments, and generates the API necessary to serialize and deserialize messages. In the software context, serialization is defined as the process of converting data structures into a byte sequence which can be transmitted among processes or stored, while deserialization refers to the data structure reconstruction from bits.

Serialization formats like XML, JSON, BJSON (binary JSON) are used extensively for network and interprocess communication. The most notable differences compared to Protobuf, is that XML and JSON are both human readable with the exception of BJSON. In recent years

Protobuf has become a considerable contender against these widely adopted formats since it provides superior performance. Protobuf offers both lesser message storage size as well as lesser serialization and deserialization times [13], [14].

Protocol Buffer's features like efficient data storage and efficient serial encoding of user defined data structures, make it an attractive tool. Nevertheless, it still not well suited for describing rich and extensive datasets collected from DAQ systems. To address the absence of features ProIO is leveraged. ProIO is a language-neutral event-based stream format for Protobuf messages [6]. Self-description as well as direct event access are enabled by ProIO and originally missing from Protobufs. For these reasons, ProIO is used as the output format of the data stream, as it provides both efficiency and flexibility. On a final note, a ProIO event is an encapsulation of a structure describing one or several samples of the detector output. The encapsulation is practical when reading data, as well as accessing a particular section of data. The DAQ system output is a ProIO event stream which describes the detector values. Understanding the digitized output format is of importance since it is pertinent to design choices of the system which will be introduced in Section 2.3.

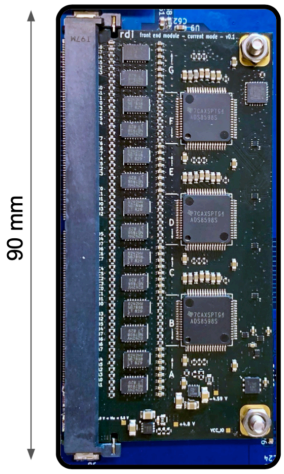
2.2 Front-End Module

The detector's outputs are connected to what is defined as Front-End Module (FEM). The main function of the FEM is to encapsulate all the components needed to convert a detector output into a digitized binary value on a modular PCB board. By making the FEM modular, scaling the detector to a different output channel configuration is straightforward. The main components of the FEM are an array of transimpedance amplifiers as well as corresponding analog to digital converters. Each FEM supports 56 input signal currents and readout channels. All the supporting circuits, like power distribution, clocking, voltage references are all mounted

in the FEM PCB. The FEM I/O is composed by an input power rail of 5V, analog input currents, and the digital signals which interface with the ADCs.

A standardized connector for the FEMs is of interest due to ease of use as well as off-the-shelf availability. The Mobile PCI Express Module (MXM) 3.0 connector was chosen [15]. Even though the MXM 3.0 standard describes several specific pin assignments as well as software interface, only the form factor of the mating 314-pin socket connector is considered.

Proper care was taken in designing the FEM PCB. Mixed signal PCB designs must isolate analog elements from digital elements due to sharp edges associated with digital signals. Digital signals can couple to analog signals potentially rendering them noisy. Some of the practices implemented in the design are separating analog and digital ground and power planes, isolating analog and digital power distribution, as well as shielding for the analog input currents.



Front-end-Module Specifications	
Channels	56
Gain	1 V/uA
Sample Rate	25 kHz
Channel Noise	50 pA
Chan Max Current	±5 uA
SNR (ENOB)	1E+05 (16.6 bits)
Linear Density	1 channel/1.6 mm
Power	48 mW/channel

Figure 6. FEM PCB board mounted on a MXM 3.0 connector. Left ICs correspond to TIA opamps and right ICs correspond to the ADCs. Specifications are quoted in the right table.

2.2.1 Transimpedance Amplifiers

The transimpedance amplifier (TIA) converts a detector output into a low-impedance voltage source through an operational amplifier (opamp) as shown in Fig. 7. The voltage output can be digitized with an ADC. The circuit bandwidth must be above 10kHz based on the detector's principle of operation. The bandwidth requirement is easily attainable with modern operational amplifiers. Additionally, the TIA gain needs to be picked strategically, as it needs to be high enough in order to satisfy a high dynamic range, but it must avoid output saturation.

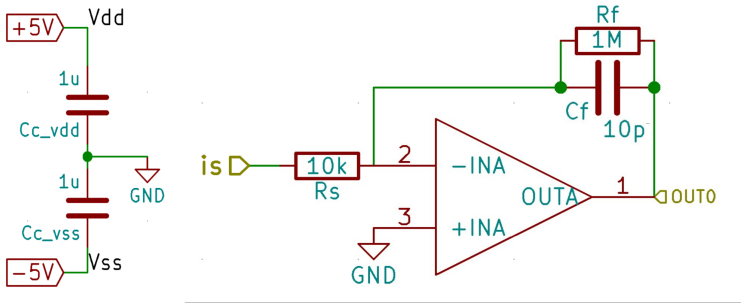


Figure 7. Transimpedance amplifier schematic. ±5V rails with decoupling capacitors. Impedance gain set with R_f resistor value.

As depicted in the TIA schematic shown in Fig. 7, the resistor R_f sets the amplifiers impedance gain and should be set according the maximum and minimum output voltage, the maximum input current, and noise requirements. In order to maximize dynamic range, a substantial R_f value is desired. As shown in Eq. 2.1a and Eq. 2.1b, which describe the output gain and the output noise of the circuit respectively; the gain of the circuit increases linearly with R_f while the output voltage noise increases proportionally to the square root of R_f [16]. Therefore, if R_f is increased, the dynamic range of the amplifier is increased as well. A good compromise was concluded to be 1V/uA or 1MΩ of gain. The type of noise described by

2.1b is Johnson–Nyquist noise which is commonly referred to as thermal noise. Thermal noise occurs due to the vibration of the charge carriers in an electrical conductor [17].

$$A_i \approx -i_s R_f \tag{2.1a}$$

$$v_{o\_nrms} \approx \sqrt{4k_B T R_f \Delta f} \tag{2.1b}$$

The high frequency characteristics of the TIA, are set by the feedback capacitor C_f which determines both amplifier bandwidth, stability, and the opamp gain-bandwidth product (GBW) requirement. In order to satisfy both the gain and signal bandwidth of 10kHz C_f needs to less than 15pF. The C_f value was set to 10pF, setting the required GBW to 40kHz with a conservative estimation [18]. Opamps that comply with the requirement are readily available.

To implement the TIAs the Analog Devices LTC6242 IC opamp was used. This part features four opamps in a single package, making it proper to reduce PCB area on the FEM PCB. The opamp is a low noise (10nV/$\sqrt{Hz}$ at 1kHz), low input bias current (1pA max), with a gain bandwidth product of 18Mhz. Due to its specifications it is adequate for the application.

2.2.2 Analog to Digital Converter

In order to digitize the output voltage of the TIA amplifiers, ADCs are used. A proper ADC architecture that supports both the effective sampling of the input bandwidth as well as a high-resolution output is needed. Both Successive-Approximation-Register (SAR) as well as Sigma Delta ADCs comply with the requirements. SAR ADCs were chosen for the balance between high resolution, sampling rate, as well as their ease of use. The simplicity of SAR ADCs is related to its main principle of operation. Conversions are based upon a binary search algorithm where the final digital output is searched among all the possible quantization values.

An output bit is determined from most significant bit (MSB) to least significant bit (LSB) a clock cycle at a time and converging into a value with each step [5].

According to Nyquist-Shannon theorem, in order to accurately reconstruct an analog signal using discrete data points while avoiding aliasing, the sample rate needs to be at minimum two times the highest frequency component of the signal [5]. This sampling rate is also known as Nyquist rate. In order to account for possible high frequency noise, an effective anti-aliasing filter is needed to avoid high frequency components mixing with the frequency range of interest. Sampling the input signal above Nyquist rate, also known as oversampling, improves SNR and avoids the need for an impractical high order anti-aliasing filter. The anti-aliasing filter, as well as the desired dynamic range need to be taken into account in order to select an effective sampling rate.

The Texas Instruments ADS8598S ADC chip was chosen for this application. It features an eight-input 18-bit SAR ADC featuring a diverse set of settings. The most notable settings used in the system, are an input range of $\pm$5V, an on-chip third-order Butterworth anti-aliasing low-pass filter [19] with a 3-dB cutoff frequency at 10.6kHz and an averaging digital filter. For every analog to digital conversion, a total of eight samples are averaged when using an 8x oversampling mode. The averaging of multiple samples enables thermal noise (or white noise) reduction due to thermal noise having an average value of zero [5]. The system is set to perform digital conversions at 25k Samples per second (Sps). With the 8x oversampling enabled, the ADCs effectively sample the analog inputs at 200kSps. The ADC features are depicted in Fig. 8.

Moving average filters, are one of the simplest digital filters due to its low hardware cost. The averaging is performed efficiently since it only requires an accumulator while the division by a base 2 number is performed by a simple bit shift. Therefore, improving SNR and dynamic range with a minimal amount of hardware. In this case, downsampling also takes place since

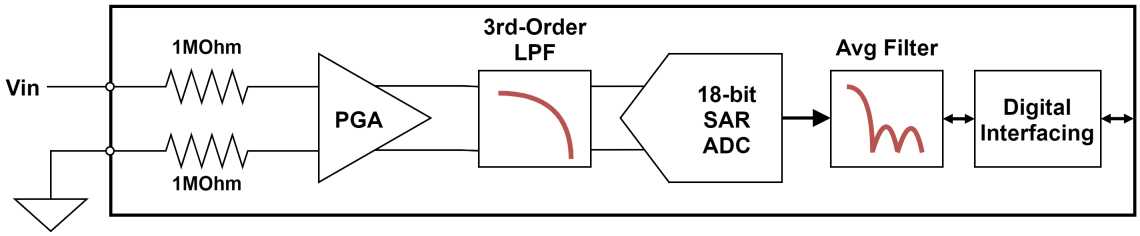


Figure 8. Simplified ADS8598S ADC diagram for a single input featuring 1MΩ input resistor followed by a programmable amplifier used for both ±5V and ±10V compatibility.

the output data rate is reduced by the oversampling factor. This keeps the sample rate low and consequently computation and processing resources consumption reasonable.

2.3 System on Chip FPGA

SoC FPGA devices integrate both a processor system with a FPGA into a single device and package [20]. The incorporation of an FPGA and an ARM CPU, provides a flexible embedded system platform for a wide range of applications. The FPGA is ideal handling real-time or time critical tasks with the power of reconfigurable hardware, while the ARM CPU shines where software flexibility is needed.

An off-the-shelf System on Module (SOM), a PCB module featuring a Cyclone V SoC FPGA, was used as the embedded platform for the DAQ. The MitySOM-5CSx SOM features the SoC FPGA as well as all its supporting electronics in a compact PCB as depicted in Fig. 9. The MitySOM connects into a carrier board using the previously mentioned MXM 3.0 connector. Therefore, the same connector is used for the major hardware components of the DAQ system.

The CPU in the SoC device, is referred to as Hard Processor System (HPS). In this application is used as a computer running a Linux OS environment which can be accessed seamlessly over the network. As shown in Fig. 10, the HPS enables several peripherals including an Ether-

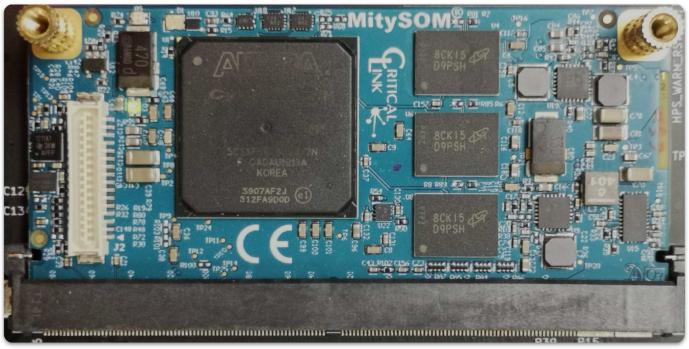


Figure 9. MitySOM-5CSx featuring a Cyclone V SoC FPGA mounted on a MXM 3.0 connector. Both RAM and power handling are seen on the right.

net Media Access Controller (EMAC), Secure Digital (SD) card controller, as well as I2C lanes for several sensors on the detector. The EMAC establishes network communication, while the SD card controller interfaces with an SD card which is used as the HPS disk. Furthermore, a UART interface is enabled for debugging purposes. Lastly, the HPS also features a DDR3 memory controller with 800MT/s transfer speed for the 1GB of available RAM.

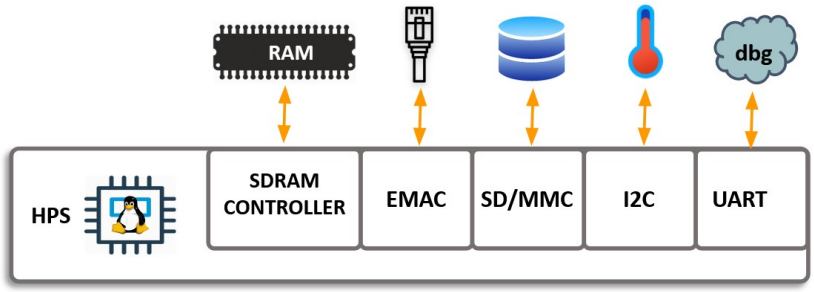


Figure 10. Cyclone V SoC HPS enabled peripherals. EMAC enables network capabilities, SD/MMC enables disk communication.

The main goal of the Cyclone V SoC chip is to transfer a ProIO data stream which describes the output of the detector over the network into a server. The FPGA device performs time critical tasks, which include driving ADC operations and readout, performing data stream Varint encoding, as well as writing data into a reserved memory space in the HPS. On the other hand,

a software application running in the HPS reads the Varint encoded data from the reserved memory and builds ProIO events.

Due to the limited computational resources available in the embedded HPS, Varint encoding is offloaded into the FPGA where hardware resources are extensive. From Fig. 4, the encoding is mainly bit manipulation which can be performed more efficiently with a hardware solution. By offloading the HPS of the Varint encoding load into the FPGA, the HPS resources are better allocated towards ProIO event building while keeping reasonable CPU utilization. Details regarding the FPGA design and the interfacing between the HPS and FPGA will be explored in detail in Chapter 3.

2.4 Data Aggregation Server

The HPS software application is directly running in the Linux environment as a service. The data acquisition process is enabled as soon as a connection is established with the server through TCP/IP protocols. The HPS outputs a data stream in ProIO format via a WebSocket [21] to an aggregation software application which can be either run on the cloud or privately.

Users are able to interface with the aggregating software via a WebServer graphical user interface (GUI) running on the server. The Web GUI can be accessed via a local client PC and provides live data display as seen in Fig. 11. The ProIO stream is constantly received but not necessarily stored on disk. ProIO can also be used as a serialization layer for inter-process communication, meaning that data can be displayed but not recorded if desired. Furthermore, the Web GUI also provides control to the user for when to start and end recordings. Additionally, it also enables users to modify display characteristics and filter live data.

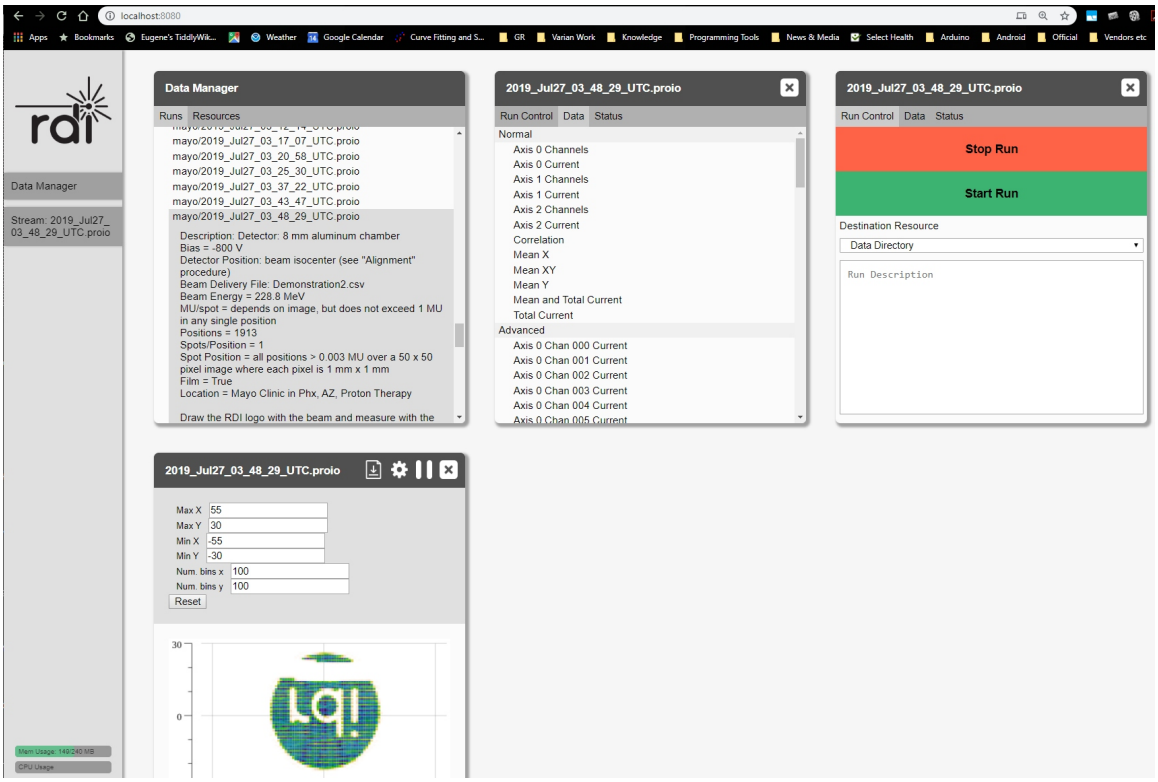


Figure 11. Web GUI live data display [22]. Enables an intuitive interface to the detector data stream. Features data storage control, display from a live stream, and display of past stored runs. Several plotting controls are offered.

2.5 Summary

The detector data acquisition was presented with special emphasis on the benefits of a streaming readout system and how the drawbacks are alleviated. The hardware components which sample detector ion currents were introduced. The operational amplifier used for the TIAs, the LTC6242, features low noise, low input bias, and sufficient gain bandwidth product. The SAR ADC selected, the ADS8598S, features an anti-aliasing filter and an 8x oversampling mode of operation. Using both components, the 10kHz bandwidth input signals are effectively amplified and sampled while maximizing dynamic range. The Cyclone V SoC FPGA was introduced as the link between the hardware and software infrastructure where its inputs are

digitized detector values while the output is a ProIO data stream to be sent over the network to an aggregation server where the data stream can be displayed and stored.

Chapter 3

FPGA SYSTEM DESIGN

A Field Programable Gate Array (FPGA) is an integrated circuit that is designed to be reconfigurable. Via an array of Configurable Logic Blocks (CLB), the device can be used to implement a multitude of logic functions. The CLBs connect through an extensive and sophisticated interconnect which is also reconfigurable. Modern FPGAs feature a set of Digital Signal Processing (DSP) slices as well as Block RAM (BRAM) slices to further increase their versatility. FPGA devices also have their own set of configurable I/O circuitry and analog Phase Locked Loops (PLLs) [23]. In short, FPGAs combine the power of hardware design with the reconfigurability of software

In order to implement a design using an FPGA, a similar workflow to an application specific integrated circuit (ASIC) is followed. Synthesis and automatic place and route (APR) are usually performed with vendor software tooling, where the design and FPGA configuration are both described with Hardware Description Languages (HDLs). SystemVerilog, one of the most popular HDL languages, was used to implement a digital design on the FPGA device.

In this work the implementation is referred to as FPGA design. The main specifications of the FPGA design are, concurrent interfacing with each ADC chip in the ADC array, performing Variable Integer Encoding (Varint) on every sample, and writing the encoded samples to a reserved RAM section. Additionally, the operations corresponding to one single sample should be performed in less than 40us. This maximum time specification is related to the desired 25kfps on the detector output where one *Sample* refers to all the digitized values of the ADC array corresponding the analog to digital conversion of all the detector outputs.

The design development was performed using Intel Quartus Prime tools and implemented in a Cyclone V SE FPGA. The terms *soft* and *hard* are used to describe hardware on the rest of this Chapter. In this context soft refers to reconfigurable hardware, while hard refers to non-reconfigurable hardware on the chip. The Cyclone V SE FPGA family features a 32-bit dual-core ARM Cortex-A9 processor embedded in the same package as the FPGA. The processor is referred to as HPS which stands for Hard Processor System. While running a Linux OS, the HPS software application provides an intuitive event structure to the data stream by building ProIO events which are sent to a data aggregation server over the network. Details on the ProIO data stream format are provided in Section 2.1.1.

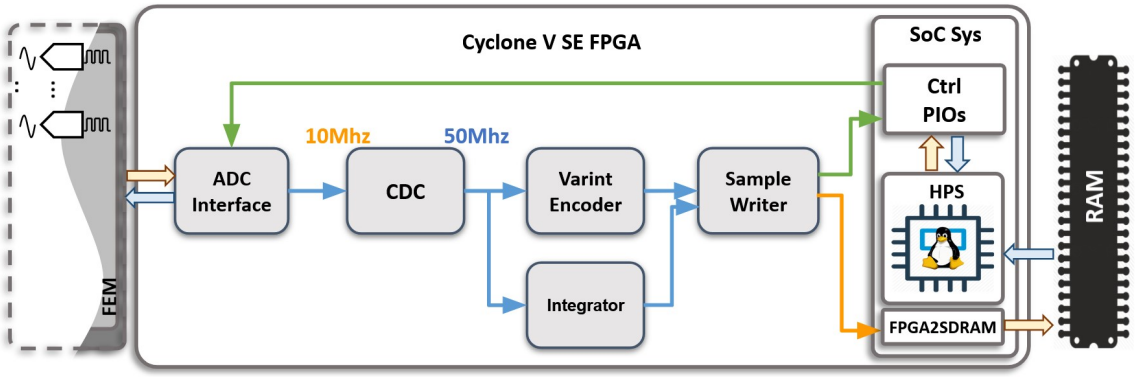


Figure 12. Top level FPGA design block diagram depicting a FIFO structure. Blue arrows represent Ready/Valid interfaces. Yellow arrow represents an Avalon Memory Mapped interface used to write the data stream directly to RAM. Control PIOs manage data acquisition and HPS data reading operations from RAM.

The FPGA design presented in this work is depicted in Fig. 12, and clearly follows a First In First Out (FIFO) structure. The main motivation for the simplicity of the system is making the design scalable, meaning that adapting the FPGA fabric to be compatible with a higher number of data readout channels should be supported by the architecture. A SystemVerilog package is leveraged to enable design flexibility. System wide parameters defined in the package, provide a simple mechanism to adapt the design procedurally at build time [24]. Furthermore, procedu-

rally generated RTL implies more maintainable, reliable, and readable RTL code [25]. In the following section, both the interfaces used between FPGA design blocks as well as the interfaces communicating with external elements to the FPGA will be briefly introduced followed by details regarding each FPGA block.

3.1 Block Interfacing

The FPGA design blocks conveyed in Fig. 12, all interact among themselves via Ready/Valid handshakes. The interface is depicted with blue arrows. Even though this is not a standardized handshake, it is a simple yet versatile protocol to manage data flow in a pipeline [26]. The states of this communication protocol are detailed in Table 1.

Table 1. Ready/Valid state signal description

Ready	Valid	Description
0	0	Idle
0	1	Waiting for receiver to be ready
1	0	Waiting for transmitter data to be valid
1	1	Data transfer is performed

3.1.1 Peripheral Interfacing

The ADC interfacing block communicates with the external ADC array via a serial interface similar to a Serial Peripheral Interface (SPI). More details about the interface will be discussed in Section 3.3. The Sample Writer interfaces with the HPS via the FPGA2SDRAM port through an Avalon Memory Mapped (AMM) [27] interface. The port provides a direct interface to RAM, and therefore to the HPS memory space. Due to the relatively vast amount

of memory available (1GB) in RAM, the port is attractive for data transfers. The HPS features hard Advanced eXtensible Interface (AXI) [28] buses for communication among its blocks. Moreover, AMM was chosen as the interface of the FPGA2SDRAM port since the soft IP provided through Intel Quartus Prime tools feature AMM interfaces. Therefore, it makes most sense to keep bus protocols consistent in soft hardware.

3.2 Platform Designer SoC System

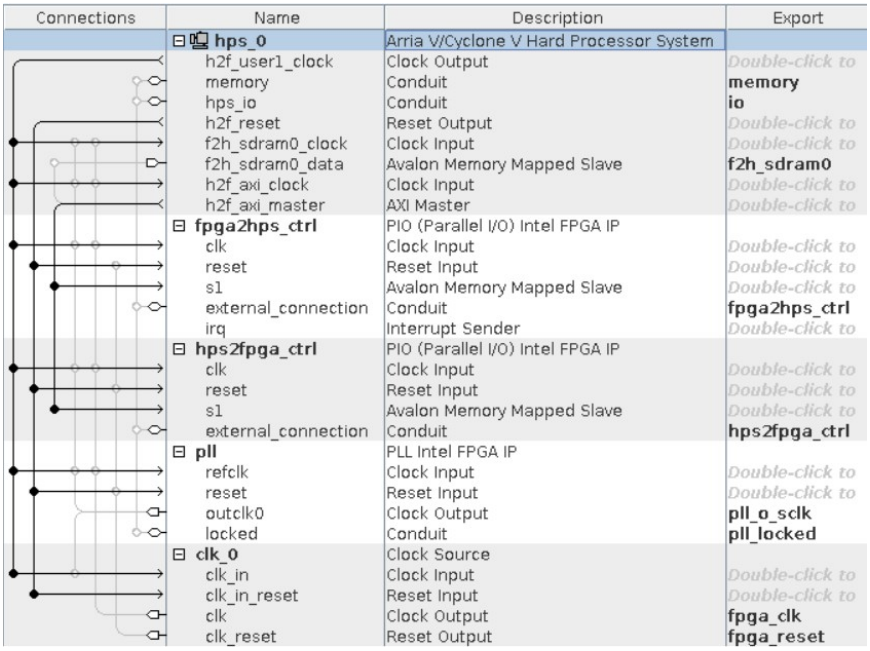


Figure 13. SoC System diagram from the Platform Designer GUI depicting the system components and their connections. Two PIOs are used for control.

In order to interface with the HPS instance from an FPGA design perspective, the Platform Designer Quartus sub tool was used. Platform Designer is a system integration tool, which encapsulates and automatically generates the interconnect among Quartus IP blocks [29]. Furthermore, the HPS configuration, including interfaces, memory timing settings, buses, clocks

and resets, are all set in Platform Designer. For the rest of the Chapter, this encapsulation will be referred to as the SoC System.

From Fig. 13, it can be observed that the major elements of the SoC System are the HPS, two Parallel Input/Output IP blocks (PIO), as well as a PLL. The PLL generates a clock (SCLK) used for the ADC array data readout and the ADC Interface. Both PIOs are used for FPGA/HPS communication, where each PIO corresponds to a communication direction between the devices, FPGA-to-HPS and HPS-to-FPGA. Lastly, the export column from Fig. 13 represents input and output ports available to the FPGA design to interface with the SoC System. All the logic composing the connections among the IP blocks is procedurally generated by the Platform Designer sub tool.

Table 2. FPGA and HPS interfacing signals

PIO	Signal	Description
HPS2FPGA	adc_powerup_en	Powers up ADC chips from standby
HPS2FPGA	data_aq_en	Enables data acquisition periodically at 25kSps
FPGA2HPS	read_ram_block	Interrupt signal for HPS to perform RAM read
FPGA2HPS	ram_block_num	Convey which block to read from RAM

The HPS2FPGA control is composed of two *enable* signals. The FPGA2HPS signals are related to the data transmission to the HPS. Details about the signals are conveyed in Table 2. More information on FPGA-to-HPS data transmission will be provided in Section 3.7.

3.3 ADC Interface

The chosen ADS8598S SAR ADC features three different interfaces to perform data readout and eight-input channels, the part is discussed in Section 2.2.2. The ADC array is composed of multiple ADC chips, which all require their own corresponding interface to the FPGA. Since

support for a high number of detector readout channels is desired, FPGA I/O must be used thoughtfully. The serial readout ADC interface is used, since it requires the least I/O by using only six PCB traces. The serial ADC interface signals are described in Table 3.

Table 3. ADC interface signal description

Signal	Description
convst	Start conversion at rising edge
csn	Active low chip select enables data readout
adc_busy	ADC in busy status while conversion ongoing
adc_sdata	ADC output serial data
adc_reset	Active high ADC chip reset
SCLK	Serial clock for data readout

Creating independent interfaces from the FPGA to each ADC would be impractical as well as non-scalable. Therefore, the output control signals are shared among every ADC in the ADC array. The inputs correspond to the output serial data and a busy status signal. Every data signal needs a direct connection to the block, in contrast, the busy status signals are OR reduced and the result fed into the block. By sharing control, the ADC interfacing is done concurrently on the ADC array.

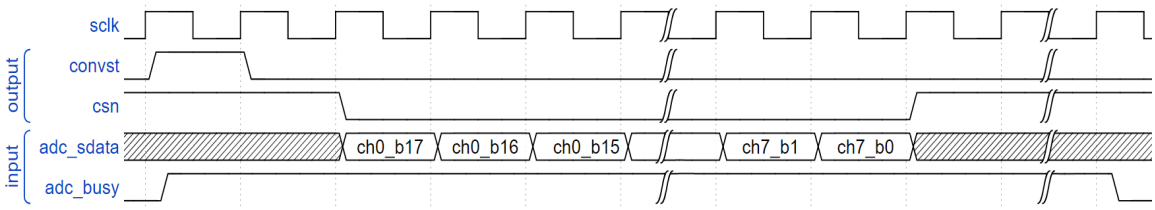


Figure 14. ADC Serial interface waveforms. Conversion begins on rising edge of *convst* and readout takes place when *csn* is deasserted. Readout needs to take place while *adc_busy* is asserted.

As depicted in Fig. 14, the *convst* signal rising edge starts the conversion on the ADC array. Once the ADC array is in a busy state, *csn* is deasserted in order to start the data readout

procedure. This is possible since the ADC is capable of readout during a busy state with the drawback that all data must be readout before the conversion is completed. Channel 0 is output first followed by the rest of the channels in ascending order, while the bits of the channels are output in descending order. Each ADC has eight input channels with 18-bit resolution; readout consumes 144 SCLK cycles.

The SCLK frequency is set to 10Mhz, therefore the readout procedure takes approximately 14.5us to complete. By sampling at 25KSps, all the pipeline operations associated to a single data sample need to be performed in less than 40us. Due to the serial interface as well as the slow clock, the readout operation is the most time-consuming operation in the pipeline.

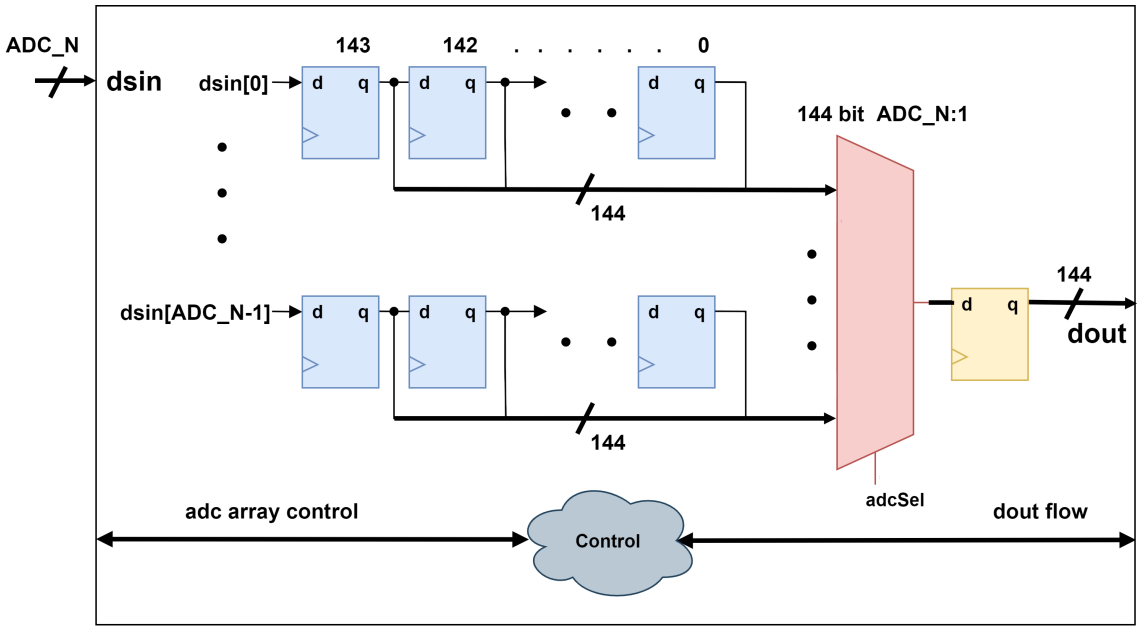


Figure 15. ADC Interface block diagram. Uses one Serial Input Parallel Output shift register per each ADC in the array to parallelize data into a 144-bit bus. Control logic is omitted.

To deserialize the serial data a Serial Input Parallel Output (SIPO) 144-bit shift register is used per ADC chip in the system as depicted in Fig. 15. Once the data for one sample is readout, transfers via Ready/Valid handshakes to the following stage in the pipeline begin. Per each

transfer, the data corresponding to an ADC is transmitted. The output order is deterministic and ascending, starting with ADC[0]. There are two techniques that are possible to transmit the data. The more elegant approach involves creating a parallel input data port on the shift register chains to be able to write all the shift register bits in the same clock cycle. The channel values can be transferred to the output, by cascading the register chains through their parallel output ports since the data output order is known.

The cascading approach was implemented, however it proved to be suboptimal due to consuming ∼25% more adaptive logic modules (ALM) compared to a less elegant large 144-bit ADC_N:1 multiplexer (mux) solution for output selection. *ADC_N* refers to the number of ADCs in the system. ALMs are Intel's implementation referring to a CLB. FPGAs are optimized from the physical design perspective to map shift registers efficiently [30]. Writing to the shift registers both serially and in parallel, requires the instantiation of a mux to select the write mode for every register in the chain. This action renders the mapping optimization unusable. Consequently, a large mux is used to select the output data. To account for the potentially deep logic paths, the output mux is registered.

The ADC interface is the most resource intensive block in this FPGA design. This is due to the long shift registers used to store channel data as well as the output selection multiplexer. Effectively, concurrency is being traded for resource utilization. The latter is preferred since the ADC Interface is the most time-consuming block in the FPGA pipeline.

3.4 Clock Domain Crosser

The remaining portion of the FPGA design runs at a clock frequency of 50Mhz which is referred to as FCLK. Since the ADC interface block is clocked by the slow 10Mhz SCLK, it is imperative that Clock Domain Crossing (CDC) techniques are needed. Clock crossing is

a critical aspect in FPGA and ASIC designs. In synchronous systems, signals must always meet their timing requirements in order to avoid metastability. Metastability is defined as an undefined logic state where the output is unpredictable. Improper clock domain crossing can cause metastability and therefore data loss, data incoherency, and even device failure [31]. The pipeline was designed to support independent clock sources for both FCLK and SCLK, meaning support for an asynchronous relationship between the clocks.

To perform CDC from a slow to a fast clock a 1 deep, 2 register FIFO synchronizer block is implemented. This CDC block was adapted from the work in [32] which discusses several different CDC techniques for different scenarios. Since the FPGA design throughput is constant and well known, the pipeline has been designed accordingly. Therefore, any irregular stalls in the pipeline are not of a concern making the 1 deep CDC FIFO sufficient.

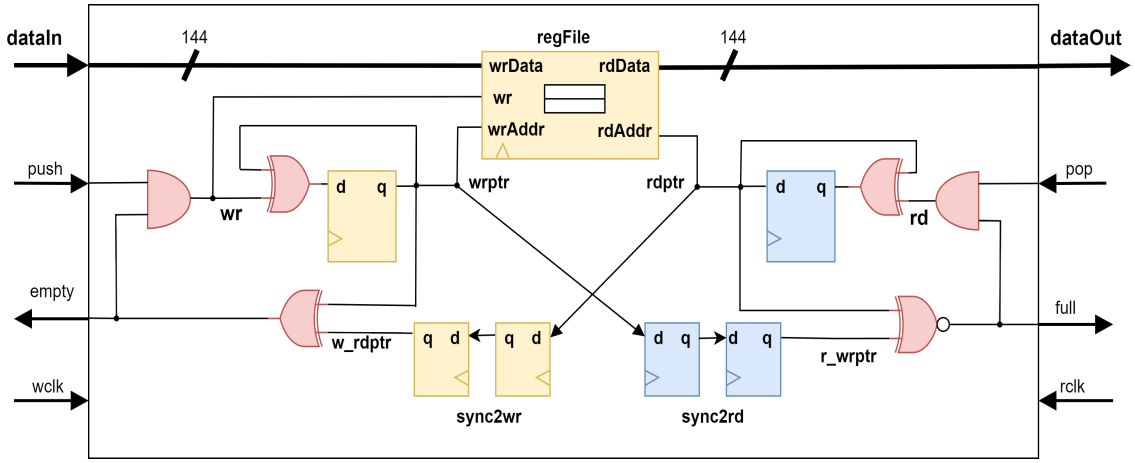


Figure 16. CDC FIFO block diagram, implemented with a 2-deep register file. Addresses are represented with a single bit and address increments are performed by XOR toggles. Yellow sequential elements are driven by the write clk (*wclk*) while blue elements are driven by the read clock (*rclk*).

The most interesting characteristic of the CDC block comes from the fact that the register bank has two registers. Therefore, both read and write address pointers can then be represented with a single bit. While performing reading and writing operations, the new pointers are calcu-

lated with simple bit toggles as depicted in Fig. 16. A synchronizer is needed per each pointer to generate the empty and full status signals. Once the pointers are synchronized to the clock domain where they are needed, the pointers are compared.

3.5 Integrator

A further requirement of the FPGA design is calculating the sum of all channel values in the ADC array. The Integrator block performs an accumulation which is analogous to the total proton beam fluence detected by the sensor at any given Sample. The result also describes the total output current from all detector channels. The input data corresponds to the channel values from one ADC transfer. A channel value is added to the total summation each cycle. Once all channel values are accumulated, the result can be transmitted to the next stage.

3.6 Varint Encoder

The Varint Encoder block also has a 144-bit input data bus for the data corresponding to one ADC chip. The encoding of the data stream is performed in this block where a channel value is encoded each cycle. By applying Varint encoding, the channel values are represented in the least number of bytes possible without modifying explicitly the data bits. As seen in both Fig. 4 and in Algorithm 1, encoding a value is mainly bit manipulation. Therefore, the encoding is performed with a combinatorial solution.

In the software context, the Sample data is a repeated packed field in the Protobuf message, meaning that data is required to be stored consecutively in memory space. This action is referred to as packing. The memory used to pack data inside the Varint Encoder block is

Algorithm 1 Varint encoding pseudocode

```
    Input:value
    Output:varint,len
 1: repeat
 2:     varintByte = lower 7 value bits
 3:     value >>= 7                          ▷ get rid of lower 7 bits
 4:     if value is not 0 then               ▷ more bytes pending
 5:         varintByte |= 128                ▷ set MSB continuation flag
 6:     end if
 7:     varint[len] = varintByte
 8:     len++
 9: until value is 0
```

a FPGA Block RAM (BRAM) with 256-bit wide entries. The block unit performs both the Varint encoding with a combinatorial solution, as well as packing the resulting Varints.

3.6.1 Combinatorial Varint Encoding

To implement the combinatorial encoding, the input A is first sliced into 7-bit groups which correspond to the 7 data bits of a Varint byte as shown in Fig. 17. To create the continuation flag bits, every 7-bit slice is OR reduced to check if any bit is set. The slice is said to be populated, if that is the case. Once the populated bytes are determined, the Varint continuation flags are generated by checking if any of the more significant bytes are populated. The Varint length is generated by a priority multiplexer where the select signal is driven by the populated byte nets (*pop[*]*).

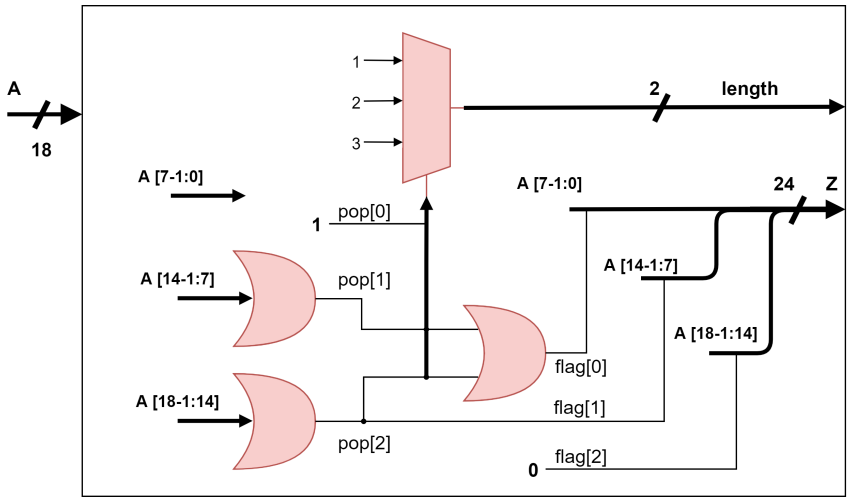


Figure 17. Combinatorial Varint encoding. Byte population is determined by *pop[*]* nets. Varint continuation flag bits correspond to *flag[*]* nets. Varint bytes composed of data and flag are concatenated towards the output.

3.6.2 Varint Memory Packing

While encoding a Varint is straight forward, it is more challenging to pack all the Varints related to a Sample into a 256-bit wide BRAM. The BRAM can be thought as the storage element of a FIFO containing packed Varints. The packing procedure is described in Algorithm 2, where the for loop iteration represents one clock cycle in the hardware implementation. It is implemented with three groups of logic, storage, occupancy, and concatenation as shown in Fig. 17.

Two sequential elements are used for storage. The BRAM with a read latency of 1 clock cycle and a 272-bit D-FF used as a temporary storage element. A concatenation of the previous Varints with a new Varint is performed to pack the data, the result is fed to the D-FF. Once the length of the concatenation is bigger or equal to the BRAM entry, the lower 256 bits are written into the BRAM and any remaining bytes are stored on the lower bytes of the D-FF. The process repeats as input data enters the block.

Algorithm 2 Varint packing pseudocode

```
    Input: Eight 18 bit channel vals
 1: for val in channel vals do
 2:     varint, len = VarintEnc(val)
 3:     p_occ = occ+len
 4:     wrEntry = p_occ >= MEM_WIDTH
 5:     if wrEntry then                                  ▷ Update entry occupancy
 6:         nxt_occ = p_occ - MEM_WIDTH
 7:     else
 8:         nxt_occ = p_occ
 9:     end if
10:     p_pkdVarints = (varint<<occ*8) | pkdVarints          ▷ Concatenate varint
11:     if wrEntry then                                  ▷ Store in BRAM or DFF
12:         bramWrData = p_pkdVarints[MEM_WIDTH-1:0]
13:         nxt_pkdVarints = p_pkdVarints[MSBs:MEM_WIDTH]
14:     else
15:         nxt_pkdVarints = p_pkdVarints
16:     end if
17: end for
```

The *occupancy* tracks the length of the packed Varints (in bytes) being stored in the D-FF. Occupancy is tracked by adding the length of each value encoded. The adder inputs are the previous occupancy value and the new encoded value length. There are two possible scenarios when updating the occupancy. If the addition result is bigger than the BRAM entry width, the BRAM entry byte width is subtracted from the addition and the resulting value registered, otherwise the addition is registered bypassing the subtraction.

A left shifter is used to perform part of the concatenation. It shifts the new Varint left by $occupancy*8$. The shifter output leaves a group of zeros in the LSBs which are to be populated by the previously packed data. The concatenation is performed with a bitwise OR between the shifter output and packed data.

As seen on Fig. 18 the block has an output *payload* interface. The payload refers to the number of bytes that are stored inside the BRAM. A payload transfer enables the pop interface and disables the push interface until all the data stored in the BRAM is read and the FIFO

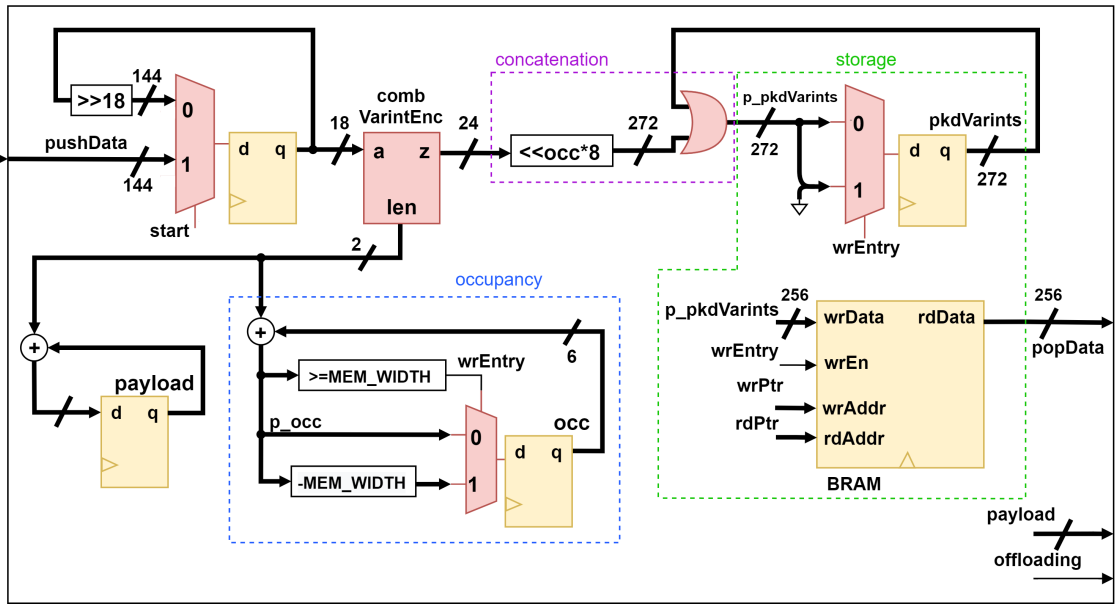


Figure 18. Varint encoder block diagram. A value is encoded and packed each clock cycle. Packing implemented by concatenating a new Varint with previously packed Varints. A BRAM entry is written when the length of the packed data is bigger than the entry.

becomes empty. While the block is on the described data readout state, the *offloading* status signal is asserted.

3.7 Sample Writer

The Sample Writer block serves as the link between FPGA and HPS. The FPGA2SDRAM port enables direct RAM memory writes independently from the HPS, making it the best option to transfer the data stream. The FPGA2SDRAM port is set to use an Avalon Memory Mapped Slave interface with a 256-bit bus. The Sample Writer generates the AMM Slave interface signals which drive the FPGA2SDRAM port. Furthermore, the block also drives the FPGA2HPS PIO to convey when and where in the reserved RAM section data is available.

3.7.1 Data Sample Description

Details about how a data Sample is arranged in memory need to be understood first before explaining the implementation of the Sample Writer block. A Sample is composed of one meta data entry and one entry per every ADC. This does not mean that there is an entry that corresponds to a specific ADC, but rather it is the total memory allocated to the whole group of packed Varints in a Sample. The memory allocation is depicted in Fig. 19.

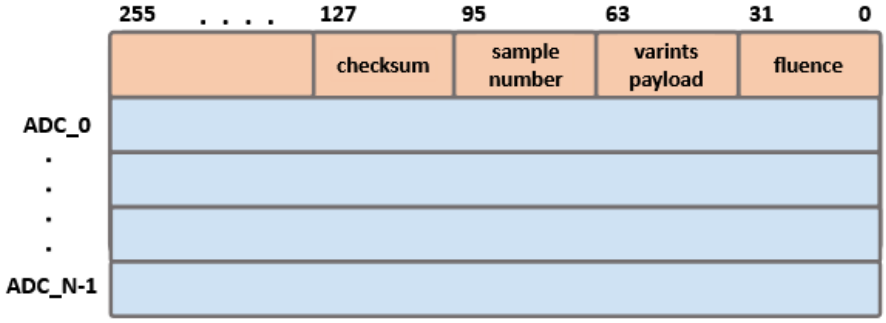


Figure 19. Sample memory allocation in a 256-bit bus. The first entry contains metadata, followed by ADC_N entries allocated for packed repeated Varints describing channel values.

A total of four 32-bit words compose the meta data entry. The meta data words associated with each Sample are the sum computed in the Integrator block, the packed Varints payload, the sample number since acquisition was enabled, and a parity byte checksum. The checksum is used to detect any bit errors that might have been introduced in the transmission of the data from the FPGA into the SDRAM [33]. Moreover, the *payload* refers to the amount of valid bytes stored in the data memory section of a Sample.

3.7.2 Buffering Scheme

A reserved RAM section serves as a buffer where the FPGA writes Samples, while the HPS reads Samples with the application software. The HPS is suboptimal when processing real-time events since it runs a Linux OS. To address the issue a group of samples are read at a time or with a single function call. The group of samples is defined as a *Sample Block* which is composed of a total of 64 adjacent Samples. A Sample Block is the minimum amount of data that the HPS reads at a time. Furthermore, the *Sample Buffer* is composed of a group of Sample Blocks as depicted in Fig. 20.

The FPGA conveys to the HPS when a Sample Block is available in RAM via an interrupt signal. The FPGA2HPS PIO described in Table. 2, is used to output the signal interrupt which triggers a Sample block read operation in the HPS and conveys the Sample block number to be read. Both the Samples, as well as the Sample blocks, are written in ascending order. Effectively, all the Sample Blocks are a circular buffer for data. An effective number of Sample Blocks is dependent on the number detector readout channels. For 336 channels, it was observed that 64 blocks were sufficient.

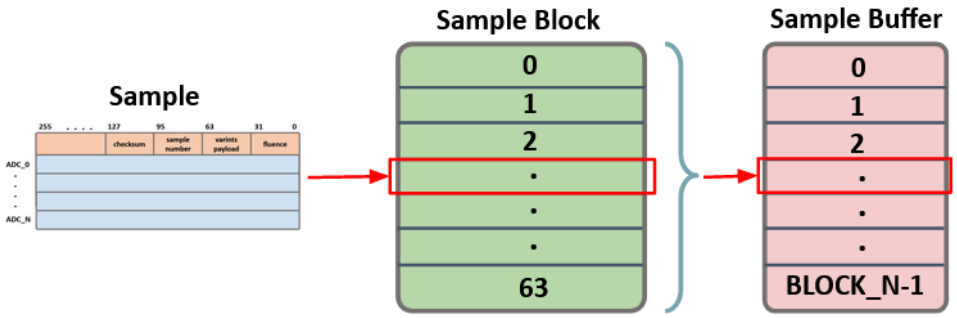


Figure 20. The Sample Buffer is a reserved memory space in RAM composed of several adjacent Sample Blocks where each Sample Block is composed of 64 Samples. A Sample Block is the minimum amount of data that the HPS can read at a time.

3.7.3 Sample Writing Procedure

The writing procedure starts once both the Integrator and Varint Encoder finish their tasks. Both the sum and the payload are transferred to the Sample Writer first. After both the sum and payload are received, Varint encoded data transfers from the Varint Encoder into the block take place. Per each data transfer there is a corresponding write operation from the Sample Writer to RAM. While the data traverses from the input to the AMM data output, the checksum is computed. The metadata is written after writing all the encoded data, ending the Sample writing procedure.

In order to generate the AMM interface write address, two counters are used. One counter defines the base of a Sample, while the other points to a specific Sample entry. Correspondingly, a coarse pointer and a fine pointer. Both pointers are added, as well as the RAM base address (which points to the Sample Buffer reserved memory) to generate the write address.

3.8 Summary

The FPGA design was presented while providing details on how each block contributes to the overall system. The pipeline was designed with scalability in mind and can be adapted to interface with a different number of ADCs. The interfacing to each ADC is done concurrently both to keep I/O usage reasonable and to keep readout time consumption manageable. Varint encoding is performed on each Sample in order to accelerate ProIO event building performed by the HPS software application.

Chapter 4

DISCUSSION AND RESULTS

In previous chapters, the data acquisition system of the instrument was presented. Each element of the system was explored, TIAs, ADCs, and an SoC FPGA. In depth details were provided on the FPGA design, since its a key element acting as a mediator between software and hardware. In this Chapter system scalability is discussed and experimental data collected at Mayo Clinic Arizona is presented.

4.1 FEM Specifications

Amplification and sampling of the detector outputs was encapsulated on the Front-End Module. Therefore, the PCB module determines system capabilities and properties such as dynamic range and noise. As shown in Fig. 6, several sampling properties have been calculated and measured. The test setup consisted on connecting the FEM to a detector with no ionizing source (no beam). The equivalent circuit of the setup resembles a floating input. Data was collected and analyzed resulting in $\sim$100dB dynamic range sampling at 25kSps, further results are shown in Table 4.

Table 4. FEM Specifications

Channels	Gain	Input Range	SNR	ENOB	Power
56	1V/uA	$\pm$5uA	100dB	16.6bits	48mW/chn

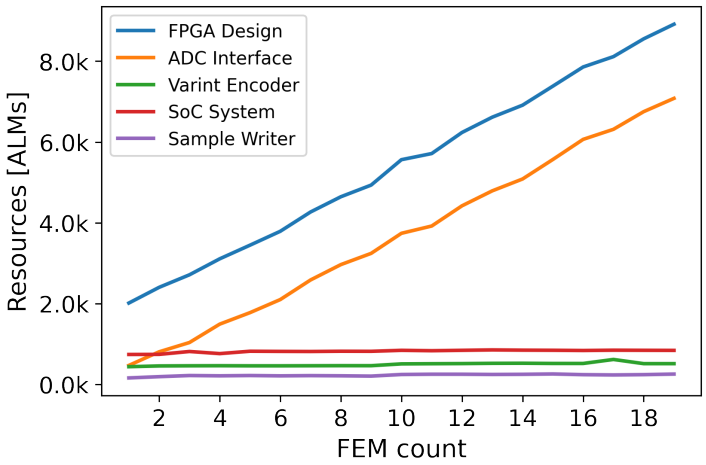
The main concern, is the power utilization of 48mW per channel or 2.68W per FEM. Power consumption was obtained with worst case scenario calculations and not with a direct measure-

ment in order to keep power distribution robust and conservative. Also note that power handling on board is composed of Low-Dropout (LDO) regulators to avoid high currents related to switching regulators, which could add noise to the analog circuitry. Therefore, low power efficiency LDOs were preferred. Thoughtful design of the power distribution tree is needed for a DAQ system with over a couple of hundred readout channels.

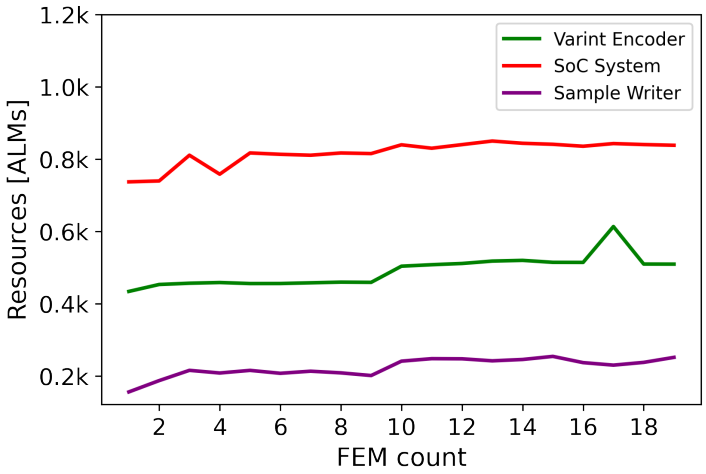
4.2 FPGA Design Scaling

FPGA design scaling is of importance, since it represents a possible system bottleneck. FPGA limitations are be explored in this section. Resource utilization is analyzed in terms of Adaptive Logic Modules (ALMs), which is Intel's implementation of a Configurable Logic Block for the Cyclone V family. This discussion is limited to a hardware design with no real emphasis on the DAQ system software due to being out of the scope of this thesis.

In order to determine how resource utilization scales with an increasing number of FEM modules, the FPGA design was built 19 times, each with one additional FEM module or additional 56 readout channels. The building process was done with balanced synthesis and fitter tool efforts on the Cyclone V SE 5CSEBA4U23I7 part, same part featured on the MitySOM. There were no pin location assignments for the project builds since there is no physical system setup related to all the FPGA builds. Due to pin location assignments not explicitly declared, the fitter tool assigns I/O at its discretion. The I/O correspond to signals needed for communication with the ADC array. The design supports up to 19 FEMs or 1064 channels, I/O being the limiting factor. It is possible to circumvent this limitation by borrowing pins from the unused HPS peripherals to the FPGA (via Quartus Prime) in order to extend I/O capabilities, but this solution is yet to be explored. The interface used to output the ProIO data stream, 1Gbit ethernet, can be expected to show saturation with the data rate needed for 1k readout chan-



(a) Design resource utilization



(b) FCLK domain units resource utilization

Figure 21. FPGA design resource utilization is shown in 21a. The ADC interface block clearly becomes dominant on design resources as FEMs are added to the system. Units in the fast clock domain shown in 21b do not increase noticeably as the design scales.

nels. Therefore, both I/O and output data throughput are design bottlenecks set by the same limitation.

Data from the FPGA design builds was gathered, and resource utilization was collected for both the design as well as the main unit blocks. Both the Clock Domain Crosser and the Integrator are not included since their resource utilization is fairly low compared to the rest. It

is observed that the system scales linearly mainly due to the contribution of the ADC interface block as shown in Fig. 21a. The remaining blocks all scale gracefully, no significant amount of resources are utilized as channels are added to the system as depicted in Fig. 21b. The 50Mhz clock speed and the low 25kSps rate, inhibits the need for parallelism on the fast clock domain. The ADC interface is clocked at a slow 10Mhz where 144 cycles are needed for ADC data readout, therefore parallelism is strongly desired. The unit high resource utilization is justified in this case. From Fig. 21a it is observed that for every FEM added to the system, or for every 56 channels added, additional 390 ALMs are utilized or around 7 ALMs per individual channel.

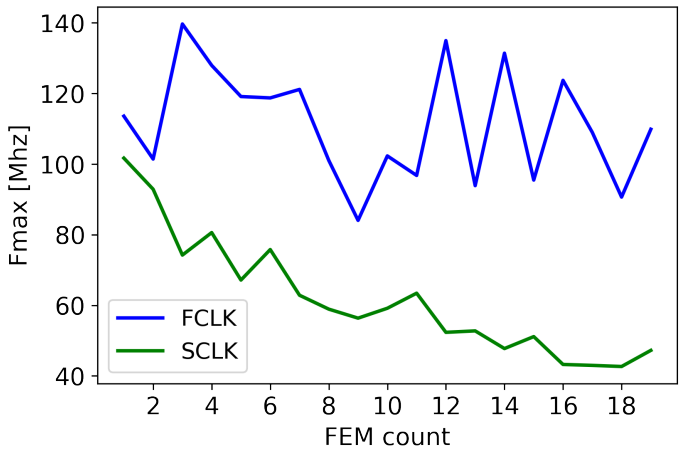


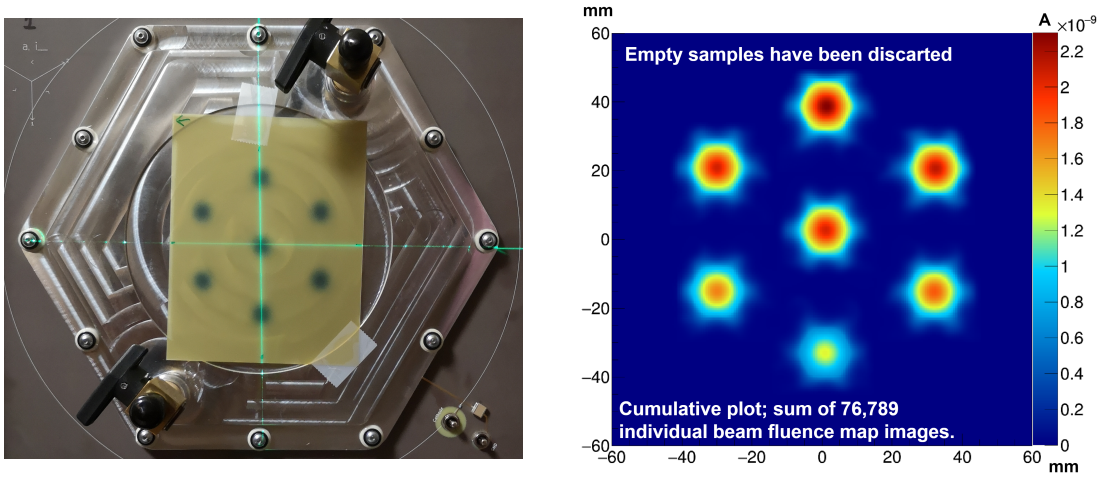
Figure 22. Max frequency clock speeds on FPGA design. Fast clock domain averages 110Mhz, while the slow clock domain decreases as channels are increased in the system.

Maximum clock frequency (Fmax) is also of interest since a higher speed could be used to increase operations in the pipeline. From Fig. 22 it is observed that there is no clear pattern on FCLK as channels are added. It is concluded that there is no relationship since no considerable hardware is being added into the fast clock domain as the system grows. The average Fmax on FCLK is approximately 110Mhz. Furthermore, SCLK Fmax clearly decreases as channels

are added. The clock drives the ADC Interface, which uses a 144bit ADC_N:1 output mux for output selection as discussed in Section 3.3. Deep logic paths associated with the output mux are the speed limitation of SCLK. The speed limitation is not a concern due to the ADS8598 ADC chip serial output readout being limited to a maximum clock frequency of 20Mhz.

4.3 Detector Experimental Measurements

Preliminary experimental data was taken with a 96-channel detector prototype measuring a proton beam source at Mayo Clinic Arizona. The prototype featured a planar area of 85cm^2. Desired detector characteristics like spatial resolution, temporal resolution, and linearity are explored briefly. The main goal from this test was to collect preliminary data for the design of a larger 336-channel detector iteration featuring an area of 1145cm^2.



(a) Detector setup

(b) Beam reconstruction

Figure 23. Experimental setup at Mayo Clinic proton beam facility shown in 23a. Seven different locations were irradiated with different energies, 10 proton spots per location. Reconstruction artifacts are visible along the three directions of the planar readout strips as shown in 23b.

The detector experimental setup is depicted on Fig. 23a where the green target depicts the beam isocenter. The detector was irradiated with 10 proton spots per location at seven different locations, each with a different beam energy. Beam reconstruction was performed as depicted in 23b. A fluence measurement is obtained, per the ion detector currents. Reconstruction artifacts visible in the reconstructed image, are a result of resolving a high degree of spatial degeneracy of the beam intensity shape using three projections. Three is the minimal number of planar projections needed to perform beam reconstruction. The artifacts are related to the 120 degree of angular separation between the three output strip sets where each corresponds to a beam projection shown in Fig. 2a.

Temporal resolution of the DAQ system enables single spot and spot structure visualization. A beam spot refers to a continuous proton delivery. In order to depict temporal resolution and how it relates to the proton beam, all the channels are added to obtain the detector's total output. Fig. 24 shows three plots with different time scales decreasing from top to bottom. A group of consecutive proton beam spots is shown on the top plot, while the middle plot depicts a single proton spot and its structure. Furthermore, the bottom plot shows the proton spot in detail. Due to the detector temporal resolution sub millisecond characteristic are seamlessly captured.

Linearity is a strongly desired detector feature. A total of 100 proton spots for 10 different levels of beam output delivery were administered to determine the detector response to dosage. Monitor Units (MU) is the usual output measure in clinical accelerators. The total output current of the detector corresponding to each spot was recorded. As shown in Fig. 25, the detector features excellent linear performance. The average total output current is plotted per dosage level with its corresponding error bars depicting two standard deviations from the average.

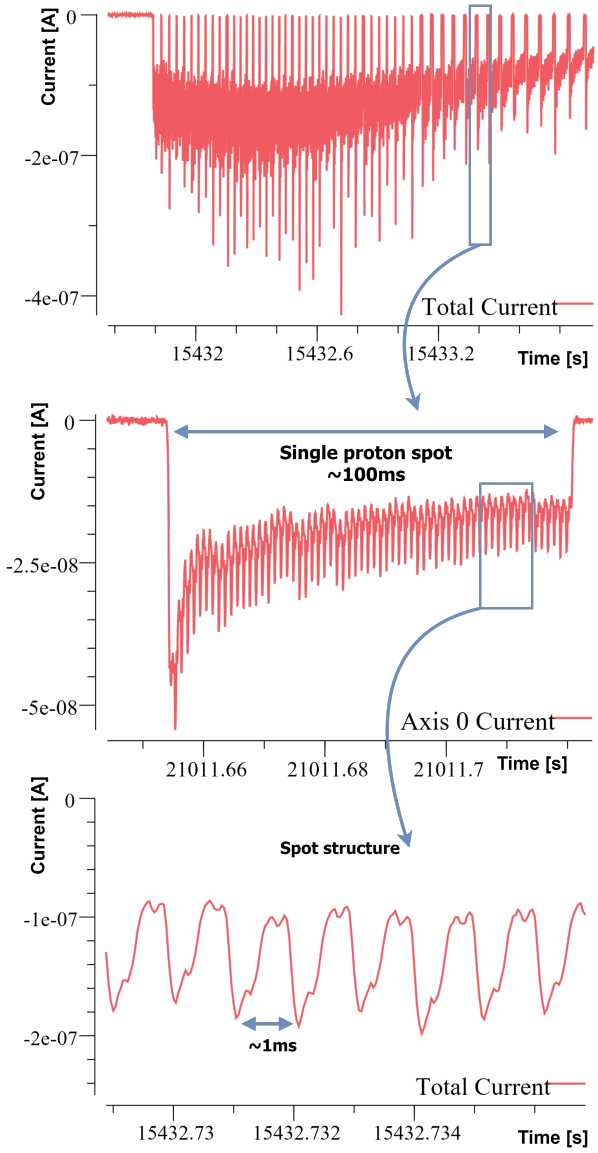


Figure 24. Temporal Resolution depicted by plotting consecutive beam spots in different time scales. Spot structure is visualized seamlessly, sub-millisecond characteristics can be measured.

4.4 Summary

Data acquisition system limitations and preliminary experimental results were shown. Specifications for the Front-End Modules were presented achieving high dynamic range, with

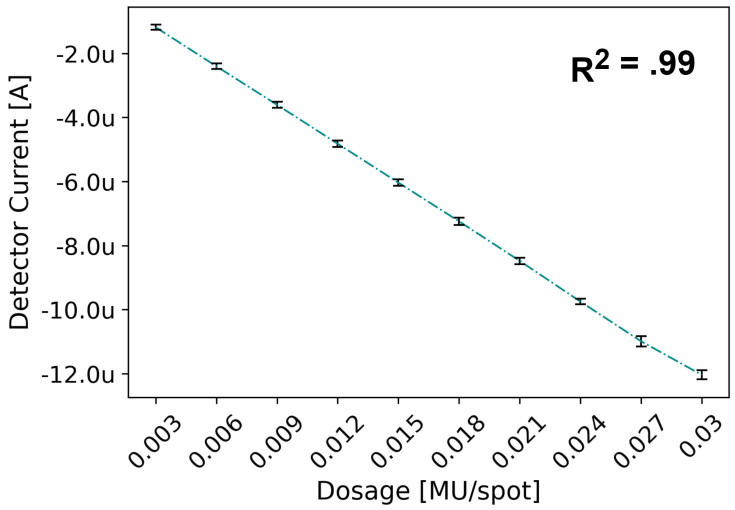


Figure 25. Detector Linearity was determined by delivering a total of 100 spots, at 10 different output dosages. Starting at 0.003 MU/spot and incrementing 0.003 MU/spot each step. Each point represents the average sum of all detector readout strips, with error bars of 2 standard deviations. Excellent linearity was concluded.

an input range of ±5uA. The main concern when scaling the system is high-power consumption of 2.68W per FEM under worst case scenarios. From the FPGA perspective, the design is limited by I/O availability rather than resources. The ADC interface in the FPGA design consumes the most resources as the system scales. The maximum amount of readout channels supported by the hardware is 1064. Temporal and spatial resolution of the system were shown in a clinical experimental setting at the Mayo Clinic Arizona proton beam. Moreover, detector linearity was verified.

Chapter 5

CONCLUSION

This work successfully developed a flexible and scalable data acquisition platform for a parallel plate ionization chamber instrument used for proton beam fluence measurement. The DAQ is part of a comprehensive high performance quality assurance system composed by the detector, the DAQ, and software infrastructure. Quality assurance aims to ensure safe and accurate delivery of proton radiation dose to patients.

The approach taken was to leverage modern hardware and software tools to enable data acquisition flexibility. Additionally, the main components of the DAQ system, the front-end modules as well as the MitySOM featuring a Cyclone V SoC FPGA, were used with modularity being one of their advantages. Modularity makes adapting the DAQ system to a new detector configuration seamless as the components can be easily reused.

The FPGA architecture was designed with simplicity in mind in order to enable scalability. As the system scales, the dominant design block in terms of FPGA resources, is the ADC Interface block. Since ADC readout is the most time-consuming procedure in the FPGA, it is best to perform the operation concurrently among all ADCs. Effectively, concurrency is being traded for FPGA resource utilization. As far as the remaining blocks in the FPGA design, they all scale gracefully with increasing channels. Data from FPGA builds shows that the FPGA design can scale up to 1064 data readout channels until running into I/O limitations. The interface used to output the ProIO data stream, 1 Gbit ethernet, would also start to saturate considering the data rate needed for 1064 detector readout channels.

The software tools that enabled flexibility were ProIO, and inherently Protocol Buffers. ProIO was used to enable a software event structure to Protocol Buffers providing an intuitive

API for inter process communication as well as efficient disk storage of detector data. The Varint encoding of the data stream is performed on the FPGA where hardware resources are extensive. HPS CPU utilization is better allocated for ProIO event building.

With the data acquisition system, preliminary detector data was obtained and presented. The 96-channel detector prototype featured 85cm^2 of active area. Beam reconstruction was briefly showcased. Furthermore, with the 25kfps from the DAQ system, temporal resolution demonstrated to resolve proton spot time structures seamlessly. Additionally, detector linearity was verified. The detectable currents are on the 100pA to 5uA range, showcasing high dynamic range.

This work has shown that by leveraging available off-the-shelf hardware and software, flexible data acquisition systems can be designed. Furthermore, the approach taken could be a reference for multi-channel data acquisition systems where the measured signal bandwidth is similar.

5.1 Future Work

One Cyclone V SoC FPGA can scale to a 1064 channel setup as mentioned in Section 4.2. In order to bypass this limitation to design a several thousand channel data acquisition system, it would be possible to utilize several Cyclone V SoC FPGAs and leverage Precision Time Protocol (PTP) under the IEEE 1588-2008 standard in order to synchronize the devices [34]. PTP enables devices connected to a network to synchronize to a grandmaster clock with accuracy in the sub-microsecond range, which is sufficient for the desired temporal resolution. An example of the approach is presented in [35]. The Cyclone V HPS supports PTP and drivers are found on current Linux releases. With this setup, each Cyclone V would output a ProIO data stream to the aggregation server.

REFERENCES

[1] F. M. Khan and J. P. Gibbons, *Khan's the physics of radiation therapy*. Lippincott Williams & Wilkins, 2014.

[2] J. Daintith, *A dictionary of chemistry*. OUP Oxford, 2008.

[3] P. A. Penczek, "Chapter one - fundamentals of three-dimensional reconstruction from projections," in *Cryo-EM, Part B: 3-D Reconstruction*, ser. Methods in Enzymology, G. J. Jensen, Ed., vol. 482, Academic Press, 2010, pp. 1–33. DOI: https://doi.org/10.1016/S0076-6879(10)82001-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0076687910820014.

[4] D. Blyth, N. Mullins, E. Galyaev, and J. Holmes, "Nonnegative gaussian process tomography for generalized segmented planar detectors," *Journal of Instrumentation*, vol. 15, no. 06, P06021–P06021, Jun. 2020. DOI: 10.1088/1748-0221/15/06/p06021. [Online]. Available: https://doi.org/10.1088/1748-0221/15/06/p06021.

[5] M. J. Pelgrom, *Analog-to-Digital Conversion*, 4th. Springer-Verlag New York, 2013.

[6] D. Blyth, J. Alcaraz, S. Binet, and S. V. Chekanov, "Proio: An event-based i/o stream format for protobuf messages," *Computer Physics Communications*, vol. 241, pp. 98–112, 2019.

[7] Unix International, *Dwarf debugging information format, version 3*, 2005. [Online]. Available: http://dwarfstd.org/doc/Dwarf3.pdf (visited on 09/26/2021).

[8] WebAssembly Community Group. "Values," [Online]. Available: https://webassembly.github.io/spec/core/binary/values.html#integers (visited on 09/26/2021).

[9] Google Inc. "Encoding," [Online]. Available: https://developers.google.com/protocol-buffers/docs/encoding (visited on 09/26/2021).

[10] R. Pidikiti, B. C. Patel, M. R. Maynard, J. P. Dugas, J. Syh, N. Sahoo, H. T. Wu, and L. R. Rosen, "Commissioning of the world's first compact pencil-beam scanning proton therapy system," *Journal of applied clinical medical physics*, vol. 19, no. 1, pp. 94–105, 2018.

[11] J. Shen, E. Tryggestad, J. E. Younkin, S. R. Keole, K. M. Furutani, Y. Kang, M. G. Herman, and M. Bues, "Using experimentally determined proton spot scanning timing parameters to accurately model beam delivery time," *Medical physics*, vol. 44, no. 10, pp. 5081–5088, 2017.

[12] Google. “Protocol Buffers,” [Online]. Available: https://developers.google.com/protocol-buffers (visited on 09/29/2021).

[13] J. Feng and J. Li, “Google protocol buffers research and application in online game,” in *IEEE Conference Anthology*, 2013, pp. 1–4. DOI: 10.1109/ANTHOLOGY.2013.6784954.

[14] S. Popić, D. Pezer, B. Mrazovac, and N. Teslić, “Performance evaluation of using protocol buffers in the internet of things communication,” in *2016 International Conference on Smart Systems and Technologies (SST)*, 2016, pp. 261–265. DOI: 10.1109/SST.2016.7765670.

[15] Nvidia. “Mobile PCI Express Module Electromechanical Specification.” (2012), [Online]. Available: https://www.module-store.de/media/pdf/d9/a4/43/MXM_Specification_v31_r10.pdf (visited on 10/01/2021).

[16] J. Grame, *Photodiode Amplifiers: Op Amp Solutions*, 1st. McGraw-Hill Education, 1995, pp. 87–90.

[17] N. AlHinai, “Chapter 1 - introduction to biomedical signal processing and artificial intelligence,” in *Biomedical Signal Processing and Artificial Intelligence in Healthcare*, ser. Developments in Biomedical Engineering and Bioelectronics, W. Zgallai, Ed., Academic Press, 2020, pp. 7–9. DOI: https://doi.org/10.1016/B978-0-12-818946-7.00001-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128189467000019.

[18] Texas Instruments. “1 mhz, single-supply, photodiode amplifier reference design.” (2014), [Online]. Available: https://www.ti.com/lit/ug/tidu535/tidu535.pdf (visited on 09/25/2021).

[19] S. Butterworth *et al.*, “On the theory of filter amplifiers,” *Wireless Engineer*, vol. 7, no. 6, pp. 536–541, 1930.

[20] Intel Corp. “Intel FPGAs & SoC FPGAs,” [Online]. Available: https://www.intel.com/content/www/us/en/products/details/fpga.html (visited on 09/26/2021).

[21] I. Fette and A. Melnikov, *The websocket protocol*, 2011.

[22] Radiation Detection and Imaging (RDI), LLC. “RDI Live,” [Online]. Available: https://github.com/rditech/rdi-live (visited on 09/27/2021).

[23] N. H. Weste and D. Harris, *CMOS VLSI design a circuits and systems perspective*, 4th. Pearson, 2010, pp. 628–629.

[24] S. Sutherland, *RTL Modeling with SystemVerilog for Simulation and Synthesis Using SystemVerilog for ASIC and FPGA Design*. Sutherland HDL, Incorporated, 2017, pp. 101–112.

[25] O. Shacham, S. Galal, S. Sankaranarayanan, M. Wachs, J. Brunhaver, A. Vassiliev, M. Horowitz, A. Danowitz, W. Qadeer, and S. Richardson, "Avoiding game over: Bringing design to the next level," in *DAC Design Automation Conference 2012*, 2012, pp. 623–629. DOI: 10.1145/2228360.2228472.

[26] Zipcores Electronic Systems Engineering S.L. "Using the valid-ready pipeline protocol." (2013), [Online]. Available: http://www.zipcores.com/datasheets/app_note_zc001.pdf (visited on 09/26/2021).

[27] Intel, *Avalon Interface Specifications*, Intel, 2021, pp. 12–36. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf (visited on 09/28/2021).

[28] ARM, *AMBA AXI and ACE Protocol Specification*, ARM, 2021. [Online]. Available: https://developer.arm.com/documentation/ihi0022/hc?_ga=2.145967412.1021591712.1586156324-1048184626.1580228297 (visited on 10/05/2021).

[29] Intel Corp., *Intel quartus prime pro edition user guide: Platform designer*, Intel Corporation, 2021. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qpp-platform-designer.pdf (visited on 09/26/2021).

[30] Altera, *Recommended hdl coding styles*, Computer Software, Altera, 2013. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/qts/qts_qii51007.pdf (visited on 09/26/2021).

[31] R. Ginosar, "Metastability and synchronizers: A tutorial," *IEEE Design Test of Computers*, vol. 28, no. 5, pp. 23–35, 2011. DOI: 10.1109/MDT.2011.113.

[32] C. E. Cummings, "Clock domain crossing (cdc) design & verification techniques using systemverilog," *SNUG-2008, Boston*, 2008.

[33] G. R. Allen, L. Edmonds, C. W. Tseng, G. Swift, and C. Carmichael, "Single-event upset (seu) results of embedded error detect and correct enabled block random access memory (block ram) within the xilinx xqr5vfx130," *IEEE Transactions on Nuclear Science*, vol. 57, no. 6, pp. 3426–3431, 2010. DOI: 10.1109/TNS.2010.2085447.

[34] J. C. Eidson, M. Fischer, and J. White, "IEEE-1588™ Standard for a precision clock synchronization protocol for networked measurement and control systems," in *Proceedings*

of the 34th Annual Precise Time and Time Interval Systems and Applications Meeting, 2002, pp. 243–254.

[35] W. Hennig, V. Thomas, S. Hoover, and O. Delaune, "Network time synchronization of the readout electronics for a new radioactive gas detection system," *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1182–1189, 2019. DOI: 10.1109/TNS.2018.2885488.