# DESIGN AND PERFORMANCE OF AN AUTOMATED PRODUCTION TEST SYSTEM FOR A 20,000 CHANNEL SINGLE-PHOTON, SUB-NANOSECOND ELECTRONIC READOUT FOR A LARGE AREA MUON DETECTOR

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

DECEMBER 2016

By

Bronson Riley Edralin

Thesis Committee:

Gary Varner, Chairperson
Tep Dobry
Galen Sasaki

... My work is dedicated to my *lovely wife* ...

Joann Edralin

for supporting my dreams and aspirations to become the best I can be

... Much thanks to my family for who I am today ...

*My parents*

Francine and Patrick Edralin

*My brothers*

Chad, Kyric and Royce Edralin

# ACKNOWLEDGMENTS

I would like to acknowledge the University of Hawaii at Manoa (UH Manoa) - Instrumentation Development Laboratory (IDLab) for supporting this work. I would also like to thank my research advisor, Prof. Gary Varner, for giving me the opportunity to work and grow as an engineer in the lab. Much thanks to Dr. Isar Mostafanezhad for his leadership and work on the firmware and software. I would also like to thank Xiaowen Shi for his contributions to hardware design. In addition to this, I thank Chris Ketter for his work on the mechanical design. I also thank Matt Andrew for his expertise during the hardware design reviews.

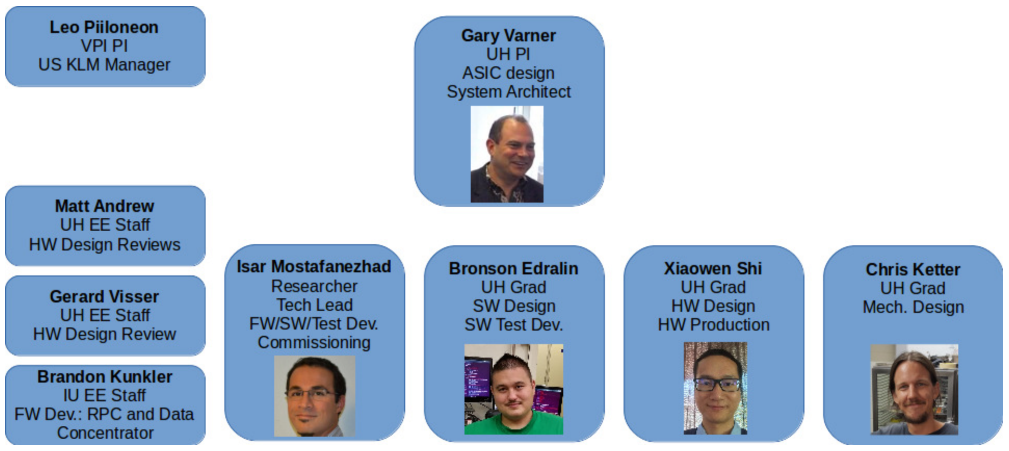


Figure 1: People involved in this work.

Thank you everyone who have contributed by using the automated software to test all the KLM electronics that include up to 1500 TARGETX ASICs, 150 MotherBoards, 150 SCRODs and 150 RHICs. Some notable testers include Denise Aliny, James Bynes, Julien Cercillieux, Alfredo Gutierrez (Wayne State), Vani Kalapciev, Khanh Le, Weng Lam Sio, Eduardo Casimiro Sanches Tanizaka (University of Sao Paulo), Cara Van De Verg, Vihtori Virta, Dr. Xiaolong Wang (Virginia Tech), Mengyuan Jerry Wu, and Kunliang Xiao. I also thank our collaborators at PNNL, KEK, Indiana University and Virginia Tech.

I have been very fortunate to be surrounded by knowledgeable and helpful individuals working in IDLab. These people include Harley Cumming, Dr. Andrej Seljak, James Bynes, Peter Orel, Dr. Oskar Hartbrich and everyone else. I would like to take the time to thank the members of my thesis committee, Prof. Galen Sasaki and Prof. Tep Dobry for their time, instruction and guidance. I would also like to thank Prof. Wayne Shiroma and my other professors for their teachings.

# ABSTRACT

The successful physics program of the Belle experiment at KEK, Tsukuba, Japan, resulted in an upgrade of the KEKB collider and of the Belle detector. The expanded Belle II international collaboration, consisting of over 500 physicists and engineers from 97 institutions spread worldwide, aims to precisely test the Standard Model of particle physics, more specifically in a search for rare B and D meson decays, and charge parity (CP) violation, by performing unprecedented precision measurements. University of Hawaii at Manoa (UH Manoa) played a lead role in the design and construction of the iTOP and KLM detectors, in particular their electronic readout. For this purpose, a pair of state-of-the-art Application Specific Integrated Circuits (ASIC) were developed in support for this world class experiment. Details of the TARGETX ASIC, a 16-channel Giga-Samples Per Second (GSPS) digitizer, developed by the Instrumentation Development Laboratory (IDLab), will be presented. Automated calibration routines were developed. Then, the performance of the ASIC was evaluated. In addition, the KLM detector readout electronics system was mass produced. Fully automated production test software was developed to systematically verify its correct operation. A serial numbering system with barcodes was set in place to properly monitor each sub-readout module and store results in a PostgreSQL database. The complete readout system will be presented, which is consisting of more than 20,000 measurement channels, and its associated electronic system.

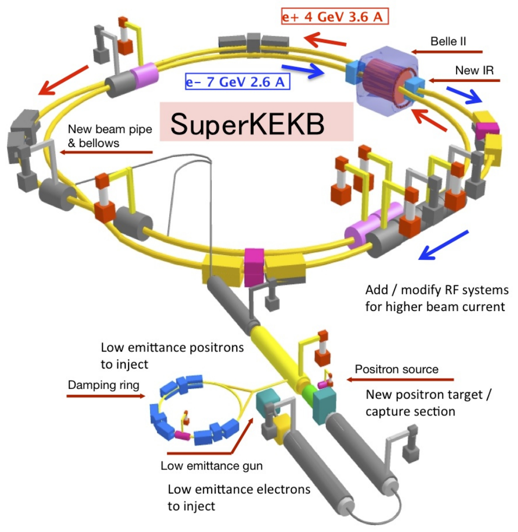
# CONTENTS

# LIST OF TABLES

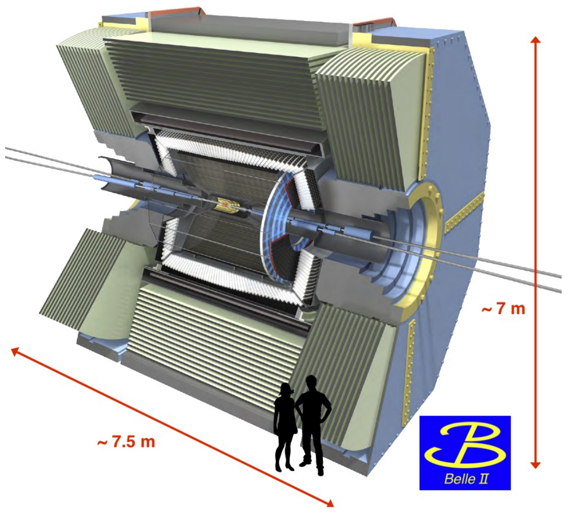
# LIST OF FIGURES

# CHAPTER 1
# INTRODUCTION

The successful physics program of the Belle experiment at KEK, Tsukuba, Japan, resulted in an upgrade of the KEKB collider, and of the Belle detector. The new Belle II international collaboration, consisting of about 500 physicists from 100 institutions spread worldwide, aims to investigate the Standard Model, more specifically search for rare B and D meson decays, and charge parity (CP) violation, by performing unprecedented precision measurements [4]. In order to do so, a set of data that is 40 times larger is required [1]. The upgraded KEKB asymmetric electron-positron collider, now named SuperKEKB, will provide the necessary luminosity. Similarly, the Belle spectrometer went through a substantial upgrade, and is now named the Belle II spectrometer. It is composed of a number of different detectors, each providing necessary information for the reconstruction of an event decay [1]. Increasing the luminosity provides a larger event rate, however this requires the spectrometer to operate in a harsher environment. As the construction of the spectrometer requires state of the art technology to withstand much elevated data rates, as well as cope with the 20 times larger radiation background [1]. This work documents the development of a readout system for one of the spectrometer components, the $K_L$ and Muon detector (KLM), where $K_L$ is the long-lived kaons [1]. The Belle II spectrometer cross-section, and SuperKEKB accelerator consisting of a linear accelerator and two 3 km circumference storage rings is shown in Figure 1.1.


e+ 4 GeV 3.6 A
Belle II
New IR
e− 7 GeV 2.6 A
SuperKEKB
New beam pipe & bellows
Add / modify RF systems for higher beam current
Low emittance positrons to inject
Positron source
Damping ring
New positron target / capture section
Low emittance gun
Low emittance electrons to inject

(a) superKEKB particle accelerator ring [1]



(b) Belle II detector [1]

Figure 1.1: The overall structure of the KEK particle accelerator ring and the Belle II detector.

## 1.1 Belle II spectrometer

Belle II is approximately a ”$4\pi$” enclosed spectrometer composed of a number of sub-detectors that provide the necessary information for event reconstruction. In addition, its inner part is placed inside a 1.5T magnetic field provided by a superconducting solenoid magnet. The magnetic field is required to measure charged particle momenta via the Lorentz force [1]. A side-view of the Belle II detector is shown in Figure 1.2.

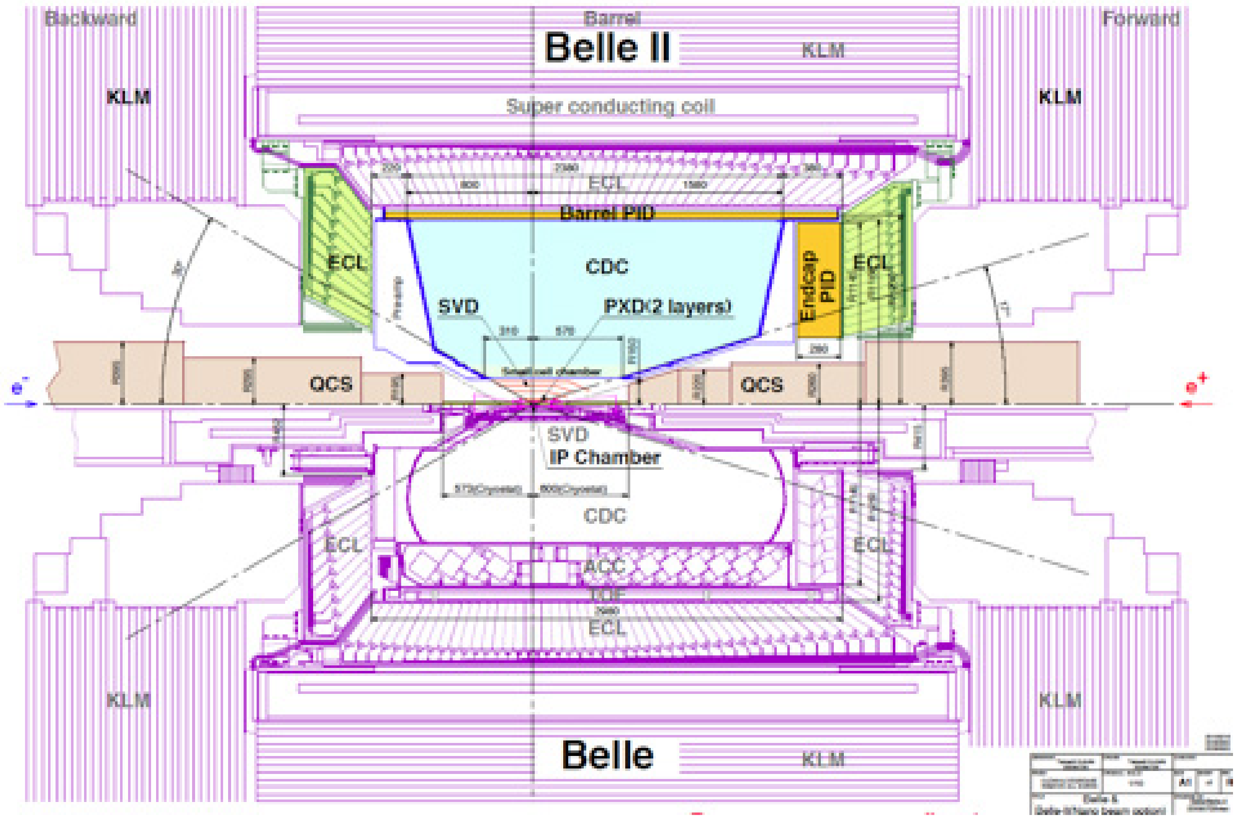


Figure 1.2: Upgraded Belle II spectrometer (top) vs previous Belle detector (bottom) [1].

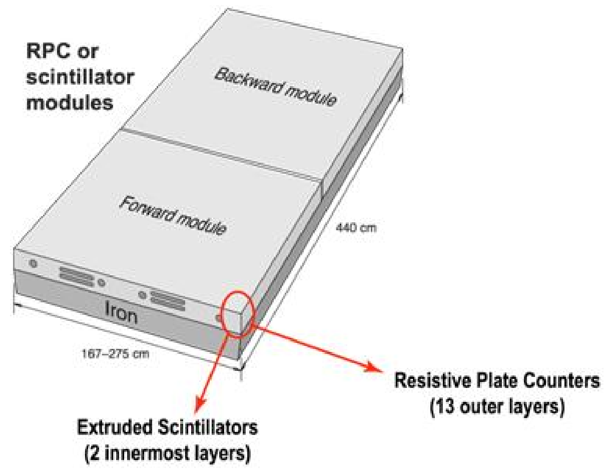
**The Belle II spectrometer is composed of the following detectors [1]:**

- “Vertex Detector” (VXD) are positioned around the electron-positron interaction point and provide a precise measurement of the position of the delay products from an interaction.
- “Central Drift Chamber”(CDC) is used to reconstruct charged particle tracks with good precision from which momentum can be determined with sufficient resolution.
- “Electromagnetic calorimeter” (ECL) consists of thallium-doped CsI crystals and measures the direction and the energy of high energy gamma rays, as well as electron identification.

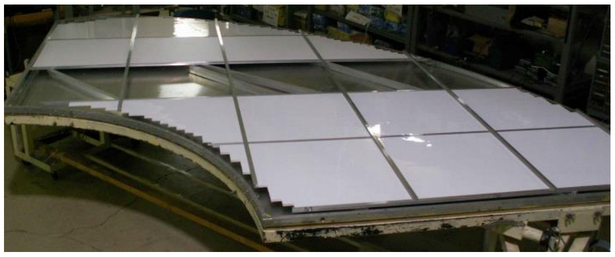
- "Imaging Time-Of-Propagation (iTOP) sub-detector and Aerogel Ring-Imaging Cherenkov timing sub-detector" are particle identification devices, based on Cherenkov photons imaging, that measure the relativistic velocity of charged particles.
- "Kaon-long and muon detector" (KLM) is built of multiple layers of charge particle detectors interspersed by iron plates in the iron return yoke of the magnet.

## 1.2 The $K_L$ and Muon detector

The KLM derives its name from the long lived kaon ($K_L$) and the muon, the particles that it is designed to detect. The outer 13 layers of the Barrel KLM part re-uses resistive plate counters (RPC). However this option was not possible for the Endcap KLM and innermost layers of Barrel KLM, because of the elevated background radiation in Belle II in combination with the RPC dead time [1]. The scintillator KLM layer instead consists of plastic fiber scintillators [4]. As an example, a kaon interacting in the metal plates will produce a hadronic shower, such shower of charged particles produces scintillation light in the plastic scintillating fibers inside the KLM detector [1]. These photons are collected into wavelength shifting fibers, which guide the photons to the photon detectors. For photon detection, solid state Silicon photomultiplier (SiPM) operating in Geiger mode were chosen [1]. In Figure 1.3 both KLM detectors are shown.


RPC or scintillator modules
Backward module
Forward module
440 cm
Iron
167–275 cm
Resistive Plate Counters (13 outer layers)
Extruded Scintillators (2 innermost layers)

(a) KLM Barrel detector.



(b) KLM Endcap detector.

Figure 1.3: The Barrel and Endcap part of the KLM detector are shown [9].

The limited space and the strong magnetic field do not allow the use of conventional photo multiplier tubes. This would naturally be the preferred choice, as they don't have any background signals or leakage currents. Silicon PhotoMultipliers (SiPM), a solid state device, are mechanically very robust devices that can measure down to single photon intensities. A number of different names for these devices might be found in literature, however most commonly are called Silicon PhotoMultipliers (SiPM), Multi Pixel Photon Counters (MPPC) or Avalanche Photodiodes (APDs). The Hamamatsu 310362-13-050C MPPC, shown in Figure 1.5, was incorporated into the design [5]. Shown in Figure 1.6, the device is composed in an array of 667 pixels, each 50 x 50 $u$m size, in a 26 x 26 pixel array [5]. Shown in Figure 1.4, the spectral response is between 300 and 900nm, heaving a peak at 500nm. According to the schematic in Figure 1.7, all of the pixels are connected in parallel [6].

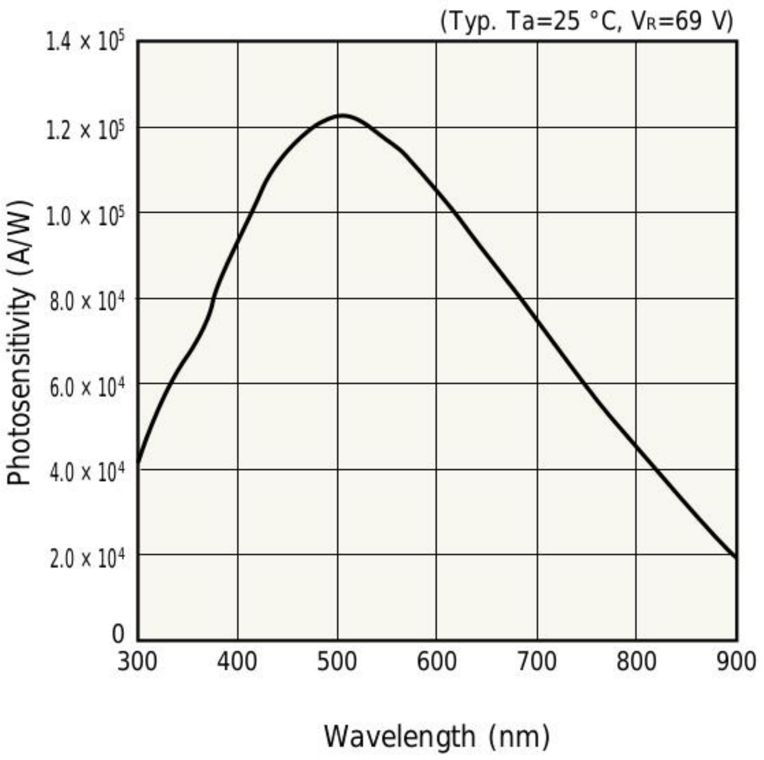


Figure 1.4: Spectral response of the SiPM [6].

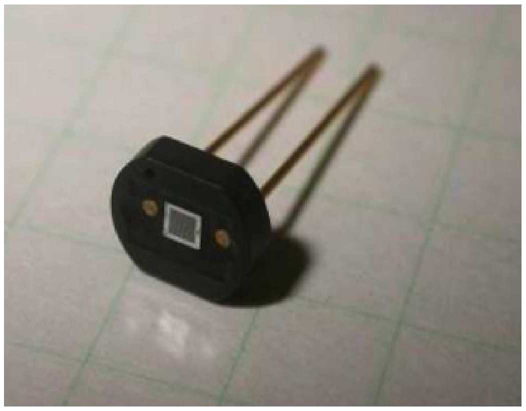


Figure 1.5: Hamamatsu S10362-13-050C MPPC [5].

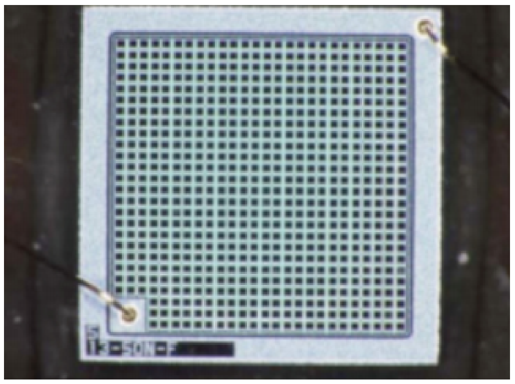


Figure 1.6: Closeup of the active surrounding 667 pixels, each 50 x 50 $\mu m^2$ area, in a 26 x 26 pixel array [5].

When a reverse bias voltage is applied slightly higher than the breakdown voltage to the device, the electric field in the pixel becomes high enough to cause a discharge (Geiger discharge) even with single absorbed photons. Multiple pixels in a single device allow photon counting. The quenching resistor stops the avalanche, and subsequently re-biases the electric field inside the device after a discharge event [6].

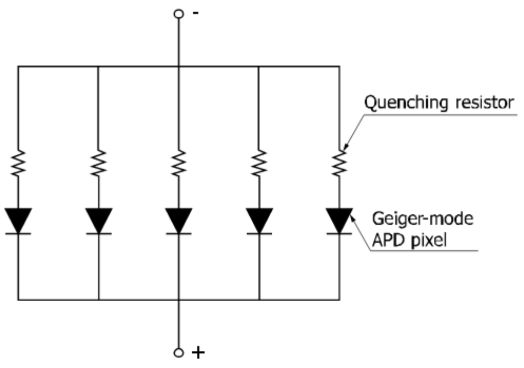


Figure 1.7: MPPC pixel array and quenching resistors [6].

The drawback of these devices is that they tend to have a high number of dark counts. Pulses are produced not only by photon-generated carriers, but by thermally-generated carriers as well. Pulses produced by the thermally-generated carriers are called the dark pulses [6]. As the reverse bias increases, the gain of the device increases along with the dark count rate. The dark pulses and signal pulses are observed, which are also multiplied to a constant signal level (1 p.e.). These dark pulses are not distinguishable by the shape from photon-generated pulses [6].

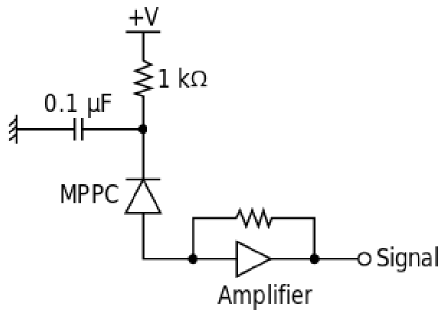


Figure 1.8: Common electrical wiring diagram for a SiPM device [6].

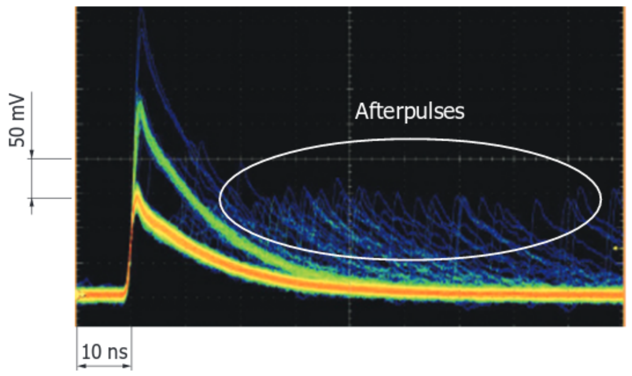


Figure 1.9: Response to mostly single photons including afterpulses (dark counts) [6].

Signals emerging from the SiPMs are short pulses in which height is proportional to the number of incident photons on the device. Usually, these signals are in tenths of a milli-volt, which does not fit the input levels of ADC converters. For this purpose, a wide band preamplifier is needed. An amplified signal is suitable for digitization, but given the signal's time scale, requires a Giga-Samples Per Second (GSPS) digitizer.

A readout electronic system measuring 20,000 channels constructed with commercial components is unrealistic in terms of cost, size and power consumption. Therefore, a customized electronic system is needed.

# CHAPTER 2
# THE KLM READOUT ELECTRONIC SYSTEM

For the construction of the Belle II spectrometer, the structure of the magnet yoke and the spectrometer support skeleton were retained from the Belle spectrometer [1]. As a result, the available space for electronic equipment remained the same. The KLM instrumentation racks for the equipment can be seen in Figure 2.1.

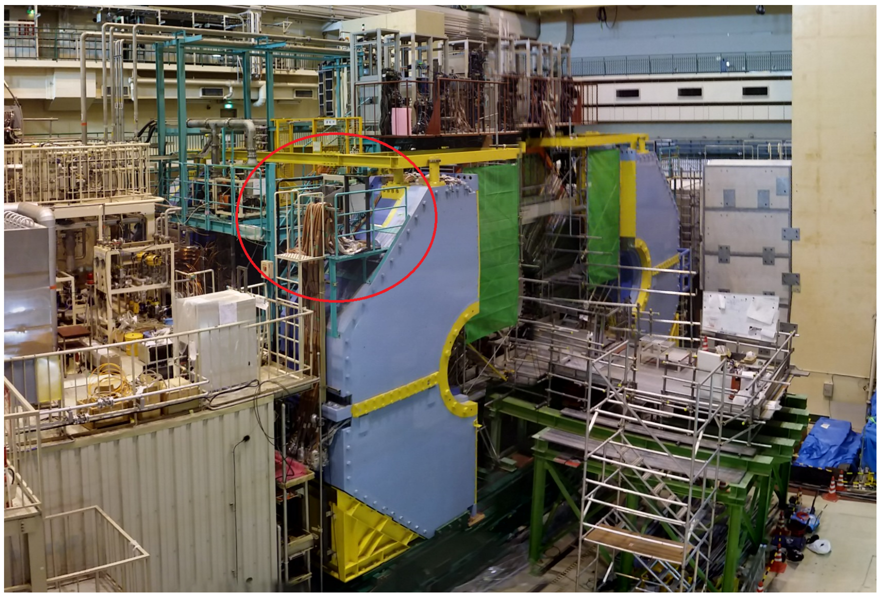


Figure 2.1: Image of the Belle II spectrometer under upgrade in Tsukuba hall in Japan. The KLM readout system resides as noted in the image [10].

## 2.1 The readout system

In order to handle the required amount of measurement channels, an integrated rack mount system was developed. The KLM electronics consist of full waveform sampling and digitizing TARGETX ASICs that were designed by Dr. Gary Varner in IDLab at University of Hawaii at Manoa. These ASICs are assembled on TARGETX ASIC daughter cards (TXDC), which are later seated on a Motherboard main board, as seen in Figure 2.3. The rest of the electronics was designed by Xiaowen Shi in IDLab. The Standard Control and Readout Device (SCROD), that incorporates the Spartan 6 Field Programmable Array (FPGA), is also seated on the Motherboard which controls the operation. The SCROD also serves as a communication interface between the outside world via Ethernet and the TARGETX ASICs. The motherboard is connected to an extension board named the Ribbon Header Interface Card (RHIC), which interconnects the MPPC signals from the detector.

In order to instrument 20,000 readout channels, 136 modules are required for the KLM detector where each module processes up to 150 scintillator bars or channels, each reading an MPPC. In summary, each KLM Readout module consists of:

- 1 KLM System Control and Readout Module (SCROD) Rev A5
  - It is a digital electronics circuit based on a Spartan 6 FPGA, which controls the operation of the TARGETX ASICs and provides communication with the external world.
- 7 - 10 TARGETX Daughtercards (TXDC)
  - Full waveform sampling/digitizing TARGETX ASIC with 1 Giga-Samples Per Second (GSPS) and 15 out of 16 channels per daughtercard are used.
- 1 KLM Motherboard Rev C
  - Serves as a platform for the daughtercards and SCROD.
- 1 KLM Ribbon Header Interface Card (RHIC)
  - Provides MPPCs with bias voltage and route signals to or from MPPCs to Motherboard.

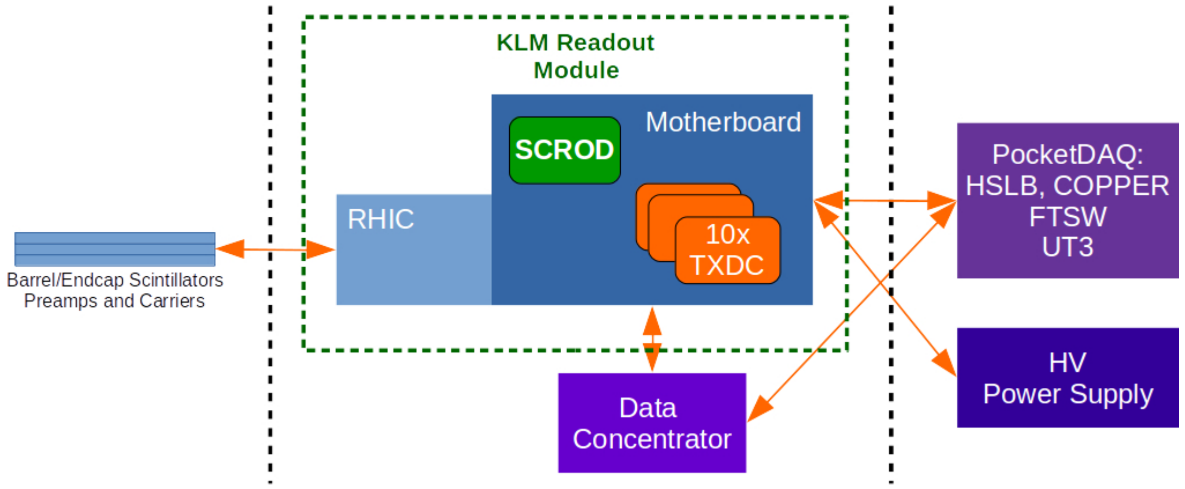
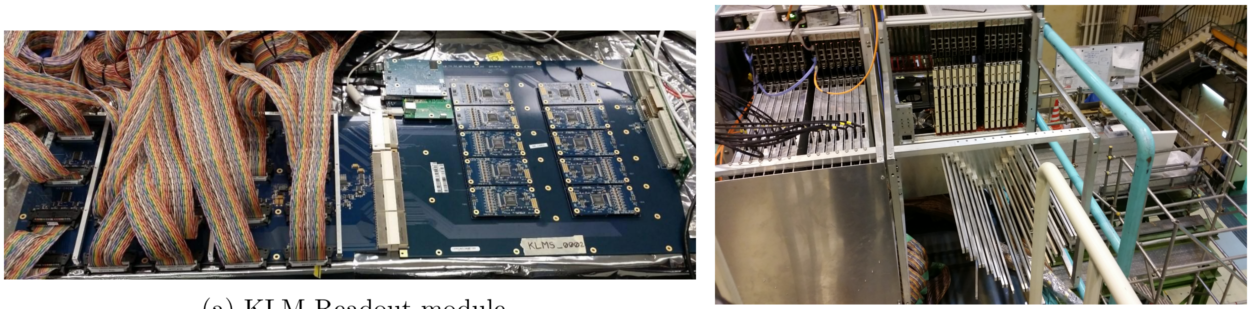

KLM Readout Module
Motherboard
SCROD
RHIC
10x TXDC
Barrel/Endcap Scintillators Preamps and Carriers
PocketDAQ: HSLB, COPPER FTSW UT3
HV Power Supply
Data Concentrator

Figure 2.2: Block diagram of hardware components for a single module reading 150 channels.



(a) KLM Readout module

(b) Custom rack

Figure 2.3: Multiple KLM Readout modules get inserted into a modified rack system.

The required specifications for the KLM Readout Modules are summarized in Table 2.1.

Table 2.1: Summary of the KLM Readout Module specifications [10].

| Parameter | Value | Comment |
|---|---|---|
| Scintillator channels | 20k | |
| TARGETX ASICs (including daughtercards) | 1250 | |
| Channels per ASIC | 16 | |
| 9U VME Motherboards | 136 | (Total Modules in Endcap) + (Total Modules in Barrel) |
| Number of Endcaps | 2 | |
| Endcap Layers | 14 | |
| Endcap Segments | 4 | |
| Total Modules in Endcap | 104 | Not using layer 2 ( 2*14*4  2*4 = 104 ) |
| Number of Barrels | 2 | |
| Barrel Layers | 15 | Responsible for only 2 of those that are scintillators while the other 13 layers are RPC |
| Barrel Segments | 8 | |
| Total Modules in Barrel | 32 | ( 2*(15-13)*8 = 32 ) |
| Sample Rate (GSPS) | 1.0 | |
| Single Sample Resolution (bits) | 10 - 12 | |

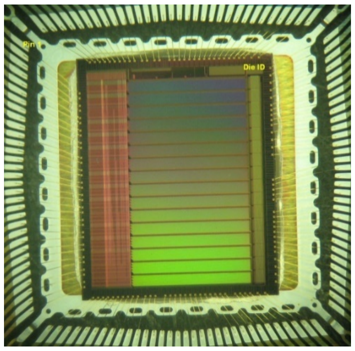
## 2.2 TARGETX ASIC

The TARGETX ASIC was fabricated in the TSMC 250nm process. This transient waveform recorder was initially designed to monolithically and inexpensively instrument large deployments of semiconductor photon detectors for large neutrino and muon detectors [8]. The general nature of the signal recording, the narrow digitization selection window and fast single conversion make it useful in a number of different applications. In order to support large arrays, self-triggering capabilities have been incorporated to permit event-of-interest identification as well as data sparsification [8].

The die photograph of the TARGETX is shown in Figure 2.4a. Functioning as an advanced Analog to Digital Converter (ADC), it is able to sample Radio Frequency (RF) signals at 1 Giga-Samples Per Second (1 GSPS). A storage array of 512 sets of 32 memory storage cells per channel means there is 16,384 memory storage cells per channel [8]. With this amount of storage per channel and assuming 1 GSPS, the TARGETX is able to store up to 16.3 $us$ depth [8]. With this much data storage, the user may be able to look back in time for the event of interest. Trigger encoding was used for reducing the number of pins. In addition, the ASIC offers self-triggering through the utilization of a signal over threshold circuit available inside the chip. The TARGETX specifications are summarized in Table 2.2.

Table 2.2: TARGETX ASIC Specifications [10].

| Parameter | Value |
|---|---|
| Channels per ASIC | 16 |
| Sampling rate | 1 GSPS |
| Sampling Array | 2x32 |
| Storage Array | 512x32 |
| Input Noise | 1 - 2 mV |
| DC RMS dynamic range | 11 bits effective |
| Signal voltage range | 1.9 V |
| | |
| LVDS sampling clock speed | 16 MHz |
| LVDS digitization & readout clock | 64 MHz (16 chan at once) |
| Single Sample Resolution (bits) | 10 - 12 |


(a) TARGETX ASIC die [7].

(b) TARGETX Daughtercard (TXDC).

Figure 2.4: The ASIC is encapsulated in the 128 LPQF package soldered on the TARGETX Daughtercard (TXDC) board [10].

### 2.2.1 ASIC Architecture and Operation

The TARGETX is part of a family of chips that was intended for detectors with the need for sampling rates of 0.5 - 1.2 Giga-Samples per second (GSPS). Triggered readout rates can reach up to 100 kHz depending upon occupancy, sample resolution and serial readout speed [8]. A serial configuration port is available to load the chip registers. In order for the chip to function properly, these registers will have to be calibrated to find the optimum register values.

The TARGETX chip's front-end of the channel is composed of a switched capacitor sampling array, shown in Figure 2.5, subdivided into 2 sampling windows each having 32 cells. The sampling time is controlled by a voltage controlled delay line (VCDL) actuating the charging of capacitors. Sampled signals from the two sampling windows are intermittently transferred into a capacitor based storage array, providing 16,384 samples for each channel. The digitization of the samples from the storage windows is done by Wilkinson converters [8].

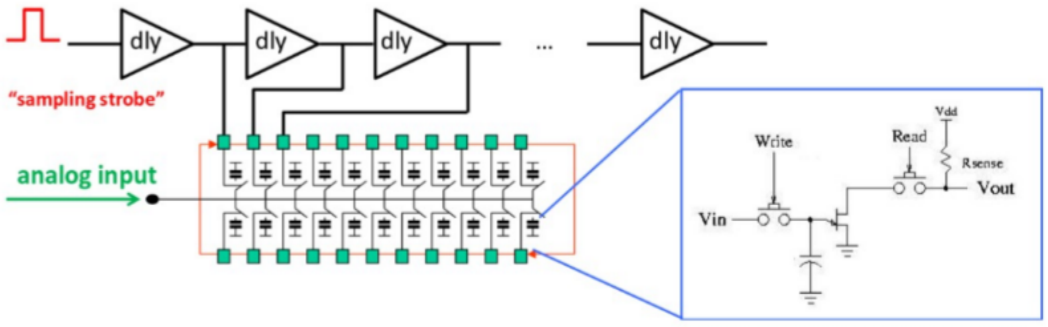


Figure 2.5: Simplified block diagram of the sampling window.

The chip requires a single 2.5V dc power supply for operation. A digital logic circuit commonly implemented on an external FPGA is needed to drive the chip. The firmware on the FPGA runs a state machine to provide the ASIC chip with an address defining the writing location in the storage array. This process runs continuously as a circular buffer. When a trigger is activated, the address of a selected storage cell is marked to prevent overwriting during conversion. Next, the FPGA starts the digitization by providing the Wilkinson ADC converters with a clock. The conversion is done simultaneously for all channels, starting with all 32 storage cells of the selected storage window at once. A "done" signal strobes from the ASIC chip when the conversion completes. At this point, the streaming of data begins through 16 LVDS pins. Once the transmission is complete, the next conversion can take place. The block diagram of the chip is summarized in Figure 2.6.

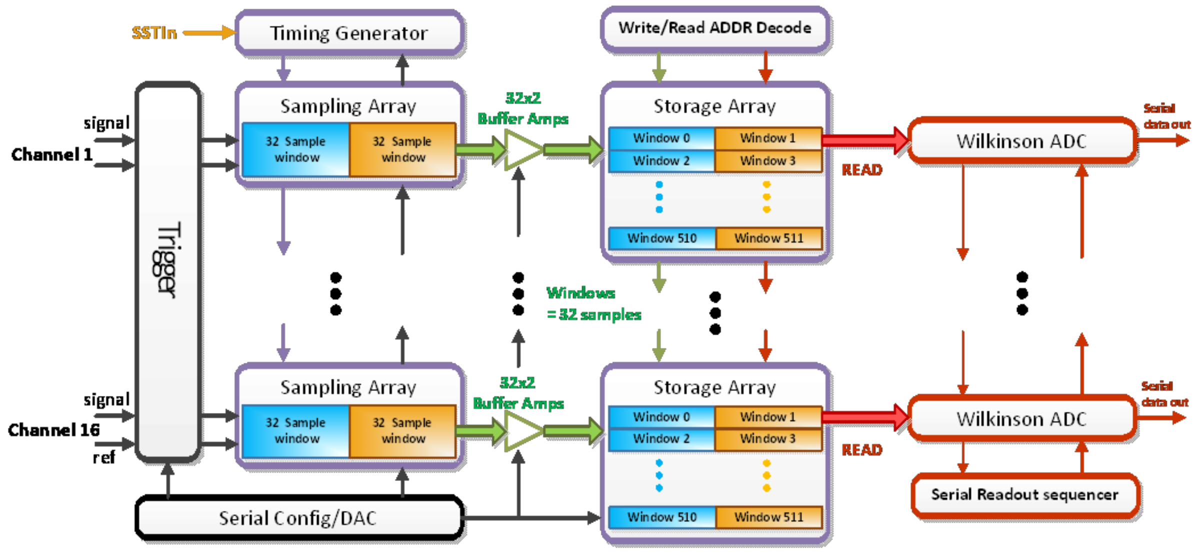


Figure 2.6: TARGETX's operational overview [8].

## 2.2.2 Timing Generator and its Timing Registers

Driven by the SSTin Low Voltage Differential Signal (LVDS) input, the configurable Timing Generator provides all the timing signals necessary for the chip to operate smoothly where it provides continuous sampling and transferring to a larger storage array bank in groups of 32. While one group of 32 in the sampling stage is busy acquiring new data, the other group of 32 in the sampling stage is buffering its data and transferring it into the storage array [8]. This is known as the ping-pong effect.

Schematic of the base timing generator cell can be seen in Figure 2.7. Inside the TARGETX delay x64 block, 64 delayed versions of the SSTin is generated with the desired delay. The delay line loop feedback adjusts VadjN for optimum sampling by comparing the SSTin and SSToutFB phase, which is connected to a charge pump whose strength is determined by the Qbias value [8]. An external capacitor stores the value of VadjN.

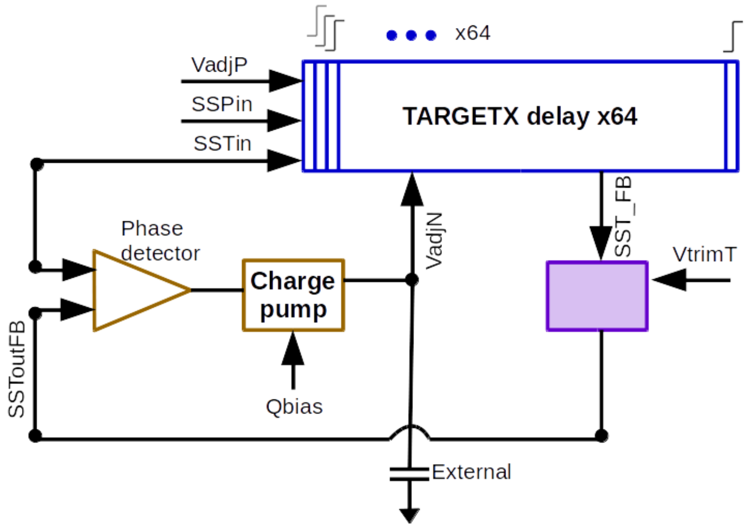


Figure 2.7: The Sample Timing Generator [8].

The VadjN value can be adjusted to select the sampling speed of the TARGETX shown in Figure 2.8.

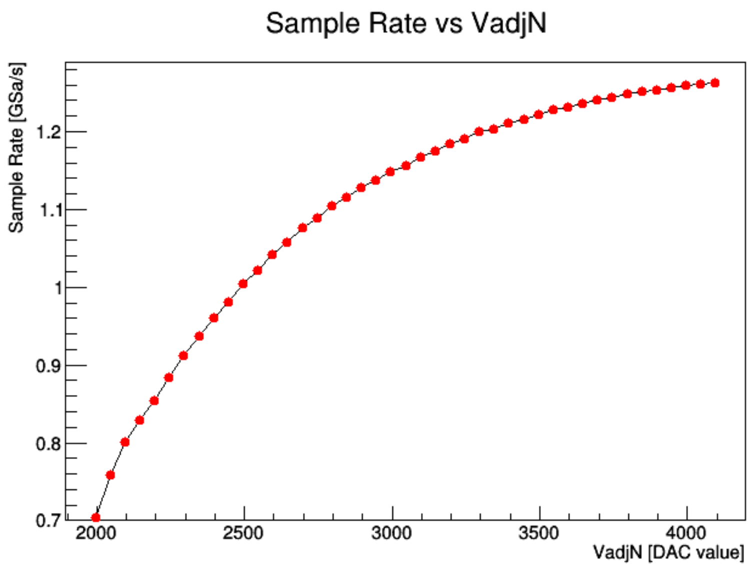


Figure 2.8: The sampling rate may be controlled by adjusting VadjN.

In order to provide seamless sampling, the strobes SSPin and SSTin must be repeated with the same frequency. Figure 2.9 shows an example of data acquisition at 1 GSPS. It starts off with SSTin and SSPin being low, but sampling will start as soon SSPin is asserted to high [8]. Later, with SSPin still being high, SSTin will be asserted high. Then, the switches will open and the instantaneous value at the input to the switch is then stored on the sampling capacitors. For this reason, the rising edge of SSTin is timing critical and much effort must be made to ensure its integrity. The difference between the rising edge of SSTin and SSPin will determine the width of the sampling strobes called SMTP. Two groups of 32 SMTP signals are generated from SSTin and SSPin. It is important to know that the sampling is actually done on the falling edge of SMTP, but the width of the signal is also an important factor since the width determines the tracking part of the sampling stage.

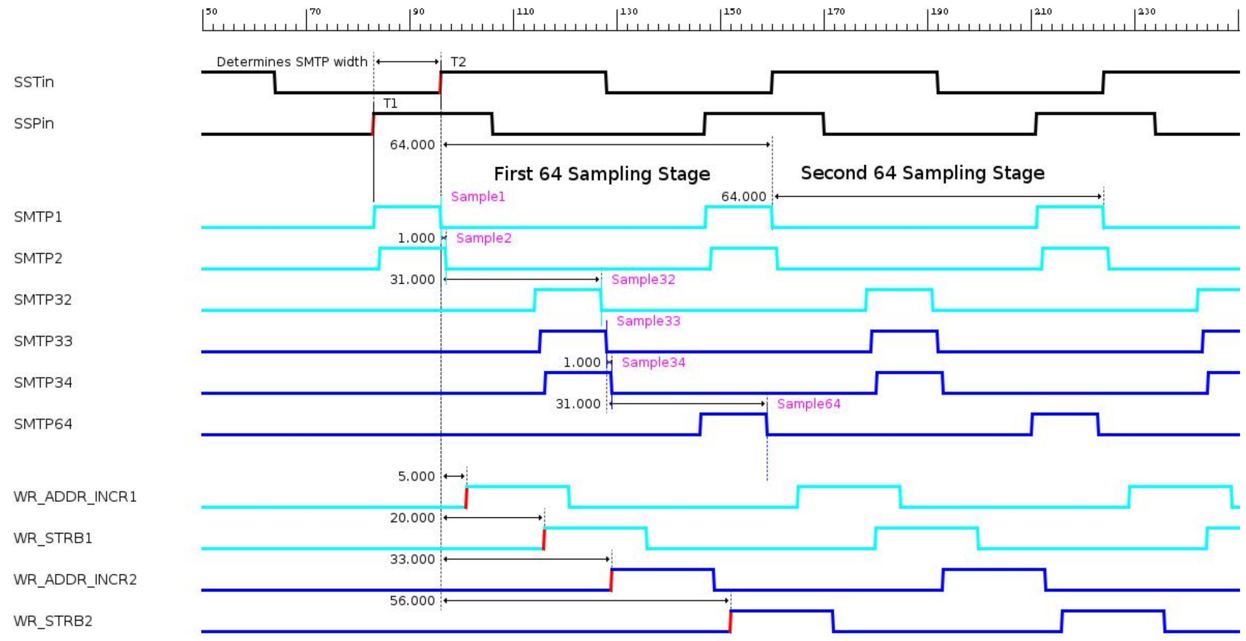


Figure 2.9: Timing Diagram for a Calibrated TARGETX during 1 GSPS data acquisition.

A sequential selection of the Write Addresses may be done with the Write Address Increment (WR_ADDR_INCR) timing signal. This timing signal is actually a clock for a synchronous counter. The outputs from the counter are fed into two decoders, which are used to select the row and column of the storage array that will be transferred to. For this reason, the width of the WR_ADDR_INCR timing signal is unimportant since we are only concerned with the rising edge. The actual transferring from the sampling array to the storage array does not happen until the Write Strobe (WR_STRB) signal gets high. There is actually an AND gate at the end where the WR_STRB signal acts as an enable for the transferring. The rising edge is critical since you only want to start transferring from the sampling array to the storage array when the sampling is done within that particular sampling array. The width is also critical since there may be some noise associated with the switching that occurs when the selection is made. The width must be wide enough for the ringing to settle otherwise we would see too much noise in the results. Because there are two sampling arrays, there must be two sets of the timing signals to manage the sampling and transferring to storage arrays.

### 2.2.3 Calibration of the Timing Registers

There are other registers that were optimized previously but the following timing registers that were focused on can be seen in Table 2.3.

Table 2.3: Timing Registers to be optimized.

| Register Name | Register Number | Register Value |
|---|---|---|
| SSPin LE | 64 | 51 |
| SSPin TE | 65 | 7 |
| WR_ADDR_INC1 LE | 66 | 5 |
| WR_ADDR_INC1 TE | 67 | 25 |
| WR_STRB1 LE | 68 | 20 |
| WR_STRB1 TE | 69 | 40 |
| WR_ADDR_INC2 LE | 70 | 33 |
| WR_ADDR_INC2 TE | 71 | 53 |
| WR_STRB2 LE | 72 | 56 |
| WR_STRB2 TE | 73 | 12 |
| SSToutFB | 75 | 58 |

Manually setting each timing register with an arbitrary value and then evaluating the waveform by simply generating plots for viewing is inefficient. A more quantitative and automated approach is taken. Using the algorithm implemented in Python, it starts with programming the default register values serially before running the calibration software created. A single timing register is programmed serially with an arbitrary value. Then, a sinusoid waveform is fed into the input of the ASIC. The waveform is sampled, digitized and later read out. A fit is performed and later evaluated to test the quality of the sampled waveform. Numerous register values are tested and evaluated. Eventually, the optimized register value is realized.

**The algorithm can be seen below:**

1. Control a function generator to inject a 40MHz sinusoid with 600mVpp amplitude and 1.5V offset.

2. Readout and construct waveform “X”

3. Scale amplitude of waveform “X” to unity amplitude

4. Construct an expected sinusoid “E” by sampling a 40MHz sinusoid with unity amplitude at 1 GSPS

5. Use a matched filter, shown in Equation 2.1, to achieve synchronization for fitting with normalized actual waveform “X” and expected waveform “E”

$$h(t) = s(T - t) \tag{2.1}$$

   where:

   $s(t)$: is the signal

   $h(t)$: is the matched filter

6. Plot synchronized waveforms “X” and “E” onto same plot and call it Fitting.

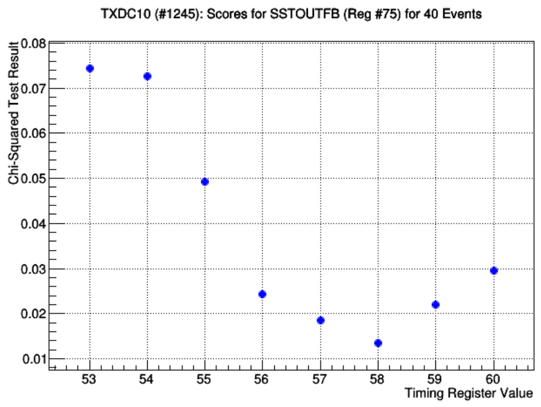
7. Plot residuals for “X” and “E”.

8. Using Equation 2.2, calculate the modified Chi-Squared Test score of “X” and “E”. Then log these raw values

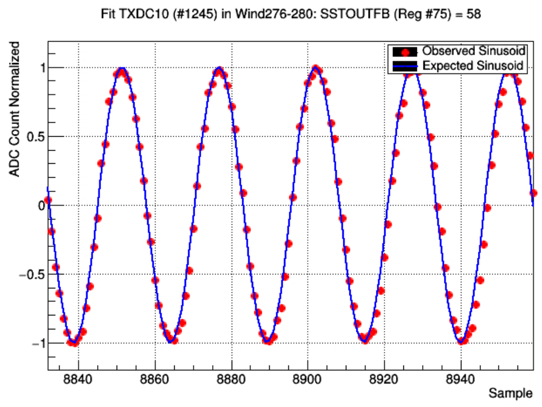
$$X^2 = \sum \frac{(Observed - Expected)^2}{(Number\ of\ Samples)} \tag{2.2}$$

9. Use the average of the modified Chi-Squared Test scores for waveforms with multiple events to determine the optimum bias register value. The minimum score represents the optimized register value.

Sweeping the timing register value has an effect on the waveform quality and the optimum register value is the one with the minimum score. Figure 2.10 shows an example of the SSToutFB register during an optimization sweep. The optimized timing register map may be seen in Table B.1 of the Appendix.



(a) Optimization sweep of SSToutFB register value



(b) Sinusoid fit performed.

Figure 2.10: (a) The SSToutFB timing register values were sweeped from 53 to 60. Then, the optimum register value of 58 was found. (b) The sinusoid fit with the optimum register value is shown.

# CHAPTER 3
# DESIGN OF AN AUTOMATED PRODUCTION TEST SYSTEM

## 3.1 Test Setup and Procedure

The testing flow without the Pre-Testing stage was proven to be inefficient once a bad batch was encountered where many of the TARGETX ASIC chips were either shorted to ground or power. A vendor mistakingly created shorts during the wire-bonding and packaging process. Since each Motherboard Production test takes up to a few hours per motherboard, it would be really inefficient without a quick check before running extensive tests. A Pre-Testing stage was created along with a new evaluation board to check for shorts. The new board incorporated clamshell packaging that allowed the user to pre-test the TARGETX ASIC chips individually before sending it to be assembled on a daughtercard. The testing flow may be seen in Figure 3.1.

Pre-Testing
Motherboard Production Testing
RHIC Production Testing

Figure 3.1: Testing flow include three stages: the Pre-Testing stage, the Motherboard Production Testing stage, and the RHIC Production Testing stage.

The Motherboard Production Testing stage includes extensive testing and evaluation of all 10 TARGETX ASIC daughtercards (TXDC). By testing the 10 TXDCs and the waveform quality, every component except for the RHIC is essentially tested as a single system. This systematic testing setup may be seen in Figure 3.2. Firmware was developed, by using VHDL, to program the SCROD that integrated the Spartan 6 FPGA into its system. The SCROD is viewed as the brain of the operation by constantly listening for commands from the outside world via Ethernet. Data from all 16 channels of the TXDC is passed serially within each channel while simultaneously being read across all channels to the SCROD. The SCROD, then, would relay the information to the PC via Ethernet by sending UDP packets. The software will be explained in more detail later. It was essential for this test to be passed before the next phase of production testing.

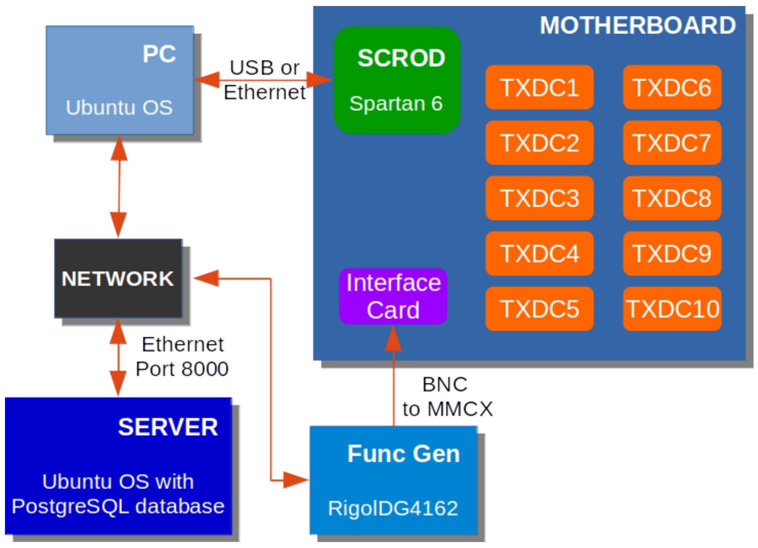


Figure 3.2: Test Setup for Motherboard Production testing.

The RHIC Production tests was created to include a similar setup as it would be installed in Japan. A custom crate was created for this task where the motherboards including the RHIC boards would be installed. The crate setup is shown in Figure 2.1. The health of the boards including the interconnects are examined by verifying the currents and temperatures. A trigger scan is also performed, which verifies any triggers from the TARGETX ASIC. If a trigger was not verified, there could either be a short in the RHIC board or a problem with the ASIC itself. Most issues were fixed with proper debugging. Other tests include the trigger efficiency of each KLM Readout module.

## 3.2 Software Overview

Any language such as C, C++, Matlab, Ruby, or Python could be chosen to design and implement the automated production software. Each language has its own set of advantages and disadvantages. Using languages such as C and C++ may result with a software that is highly optimized. One big problem with low-level languages such as C and C++ is the amount of overhead in writing the software since the programmer needs to worry about low-level concepts such as memory management. Another issue is the readability since these languages are often viewed as being cryptic. Scripting languages such as Matlab, Ruby, and Python are great choices to consider when a programmer needs to create automated production test software for the following reasons. Many libraries exist for developers to use at their disposal, which could result in higher productivity. Since these are interpreted languages, most of its implementations execute instructions directly without having to compile them into machine-language instructions. One advantage of using interpreted languages is its cross-platform trait. Another advantage of using interpreted languages is the ability to debug during run-time. Out of those three languages, Python and Ruby are more desirable for them being open-source, but Python is the clear choice for readability. Python resembles the English language where it uses words such as “not” and “in”. Additionally, the language has its own set of rules, known as PEP 8, which forces every Python developer to format their code in a certain standard. Because of Python’s readability and clear code organization, future collaboration on the software would become easier. Python has been around for over twenty years and its popularity is still growing with more communities of developers on blogs such as Stack Overflow. For all these reasons, Python was chosen to be the key language for the software.

The basic overall software structure can be seen in Figure 3.3. The Main module represents the user-interface. Its main function is to allow the user to easily modify the settings of each test. The Production Test module represents the automated test that executes once the user presses the “Start” button. A script was written for the Client module that is designed to communicate with a remote server that is listening on Port 8000. The sole job of the Client module is to pass either commands or data to the remote server so it could either upload or download results to the PostgreSQL database that was seated on the server. An Ethernet driver is also written in Python, which is being used by the Get Data driver to communicate with the SCROD in order to collect data from the TARGETX ASICs. Implementing an Instrument Control Module is necessary for automating the test since the waveform function generator needs to be turned on and off during certain phases of testing. Any form of processing and plotting is represented by the Process and Plot modules.

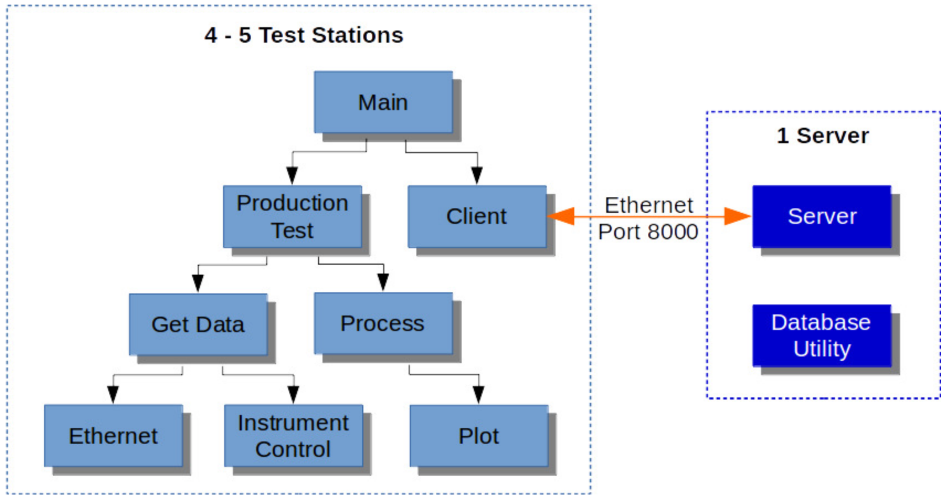
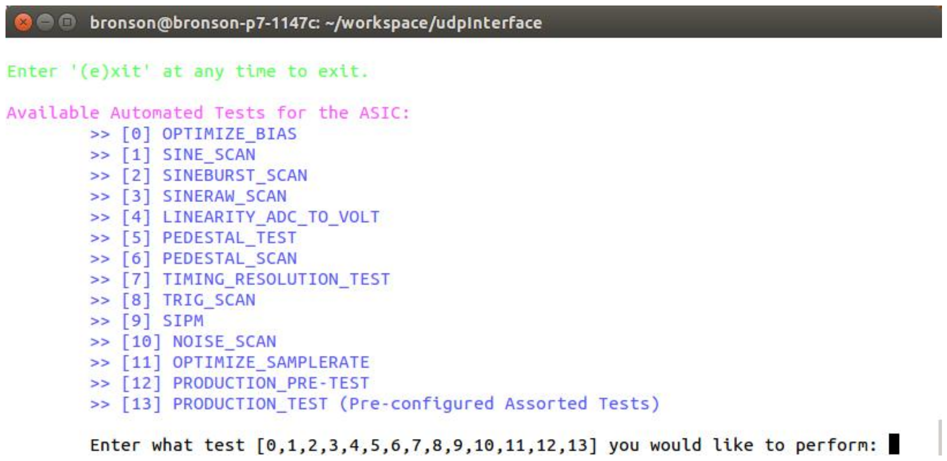


Figure 3.3: Software Overview.

### 3.2.1 Graphical User Interface (GUI)

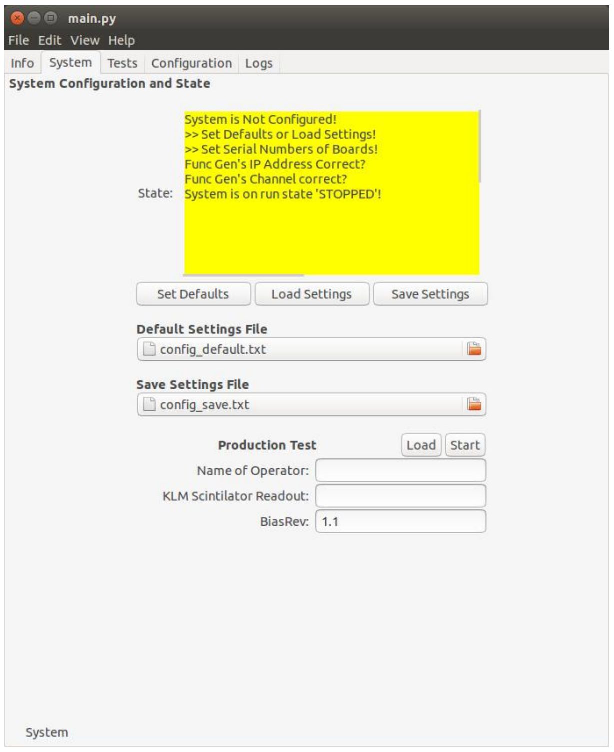
A Command Line Interface (CLI) is useful when users choose to run each test on command-line. Shell scripts could be created by placing a custom batch of commands into a file and then using the command "chmod +x <filename>" to declare the file an executable. This approach is somewhat automated but it is really problematic in editing the shell script or software each time something goes wrong. Customization of the automated test becomes problematic for the user especially if the person is not trained to know where to edit the code. For more customization options, a User-Interface (UI) was created. A screenshot of the UI can be seen in Figure 3.4.



```
bronson@bronson-p7-1147c: ~/workspace/udpInterface

Enter '(e)xit' at any time to exit.

Available Automated Tests for the ASIC:
        >> [0] OPTIMIZE_BIAS
        >> [1] SINE_SCAN
        >> [2] SINEBURST_SCAN
        >> [3] SINERAW_SCAN
        >> [4] LINEARITY_ADC_TO_VOLT
        >> [5] PEDESTAL_TEST
        >> [6] PEDESTAL_SCAN
        >> [7] TIMING_RESOLUTION_TEST
        >> [8] TRIG_SCAN
        >> [9] SIPM
        >> [10] NOISE_SCAN
        >> [11] OPTIMIZE_SAMPLERATE
        >> [12] PRODUCTION_PRE-TEST
        >> [13] PRODUCTION_TEST (Pre-configured Assorted Tests)

        Enter what test [0,1,2,3,4,5,6,7,8,9,10,11,12,13] you would like to perform:
```

Figure 3.4: Screenshot of the old user-interface that was created for automated testing.

The UI works well for systems with few customizations but it was deemed to be not enough for our test system. There was an excessive amount of information overhead to start each test. Networking information, test configuration settings and serial numbers were needed to be inputted into the software for each test. Inputting the information manually each time could result in much time wasted. One could put some of these settings beforehand in a configuration text file and allow the users to manually change the settings but this approach may lead to errors. Extra spaces or editing in the wrong place of the configuration text file are common errors that could occur when you let the user change the settings directly. With an UI, it is impossible to include all the configuration settings so the user would have to edit the code directly when a special case is needed. The excessive problems, with the UI, led to the development of a Graphical User Interface (GUI) as seen in Figure 3.5.



System Configuration and State

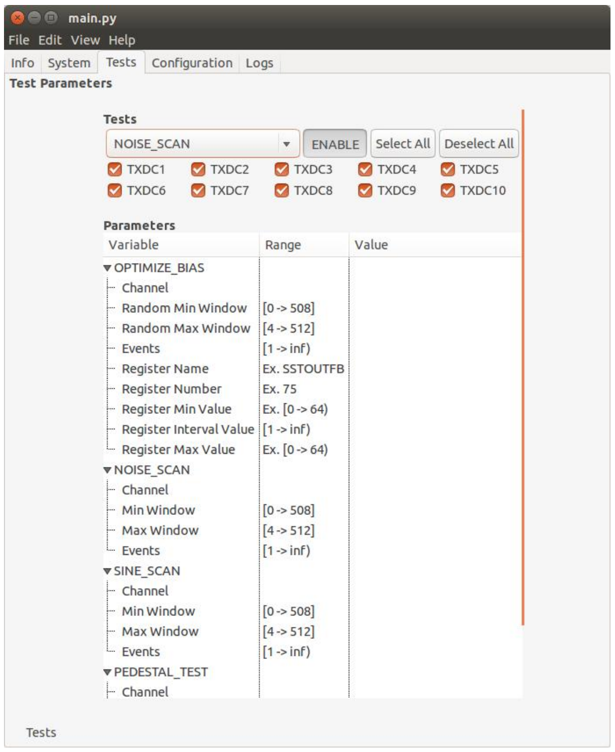
State:
System is Not Configured!
>> Set Defaults or Load Settings!
>> Set Serial Numbers of Boards!
Func Gen's IP Address Correct?
Func Gen's Channel correct?
System is on run state 'STOPPED'!

Default Settings File: config_default.txt

Save Settings File: config_save.txt

Production Test
Name of Operator:
KLM Scintilator Readout:
BiasRev: 1.1

(a) GUI System section.



Test Parameters

Tests: NOISE_SCAN

✓ TXDC1 ✓ TXDC2 ✓ TXDC3 ✓ TXDC4 ✓ TXDC5
✓ TXDC6 ✓ TXDC7 ✓ TXDC8 ✓ TXDC9 ✓ TXDC10

Parameters

| Variable | Range | Value |
|---|---|---|
| OPTIMIZE_BIAS | | |
| Channel | | |
| Random Min Window | [0 -> 508] | |
| Random Max Window | [4 -> 512] | |
| Events | [1 -> inf) | |
| Register Name | Ex. SSTOUTFB | |
| Register Number | Ex. 75 | |
| Register Min Value | Ex. [0 -> 64) | |
| Register Interval Value | [1 -> inf) | |
| Register Max Value | Ex. [0 -> 64) | |
| NOISE_SCAN | | |
| Channel | | |
| Min Window | [0 -> 508] | |
| Max Window | [4 -> 512] | |
| Events | [1 -> inf) | |
| SINE_SCAN | | |
| Channel | | |
| Min Window | [0 -> 508] | |
| Max Window | [4 -> 512] | |
| Events | [1 -> inf) | |
| PEDESTAL_TEST | | |
| Channel | | |

(b) GUI Tests section.

Figure 3.5: A Graphical User Interface (GUI) was created to improve user-experience.

Much development was put into the features of the GUI. The GUI would not let the user start testing until the "Set Defaults" or "Load Settings" button was pressed. The "Set Defaults" button would load the default settings for all the tests including the configurations for the function generator and more. The default configuration file could never be overwritten with the "Save Settings" button but the custom settings configuration file could be overwritten with it. This custom configuration file could be loaded by pressing the "Load Settings" button. This gave the tester more flexibility in testing. Custom settings could be saved without overwriting the default settings that was programmed into the system. Another really good feature of this program is that it allows the tester to now select or deselect any TARGETX ASIC daughtercards (TXDC) to test by simply clicking on the check-buttons associated with the TXDC numbers. These check-buttons were unique to each test by using the drop-box menu under "Tests" to select what Test setting to choose.

### 3.2.2 Serial Numbering System and Logging with Remote Database

With over a thousand of TARGETX ASICs and hundreds of different boards to test, the complexity increased in tracking each hardware component. Test results and records for these tests were needed to be properly logged since different time-shifts were set up for each tester. A serial numbering system was created to track each system and its components. In Figure 3.6, the KLM readout module is given a virtual serial number that packages all its hardware components together. The serial number formats were created to be unique as it represents the individual board(s) with its revision number:

- **KLMS_0000** = KLM Readout Module
  - **MB_C0000** = Motherboard Rev C
  - **S_A50000** = SCROD Rev A5
  - **RHIC_C0000** = RHIC Rev C
  - **0000** = TARGETX Daughtercard (TXDC)

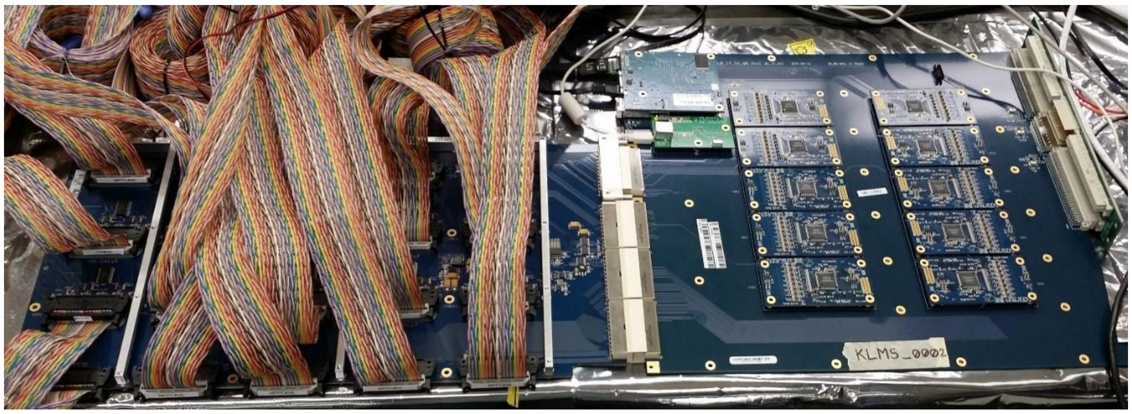
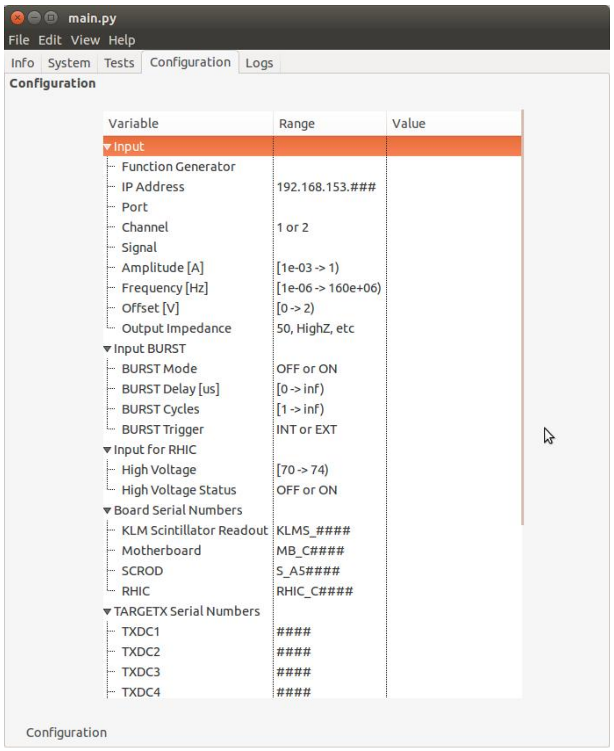


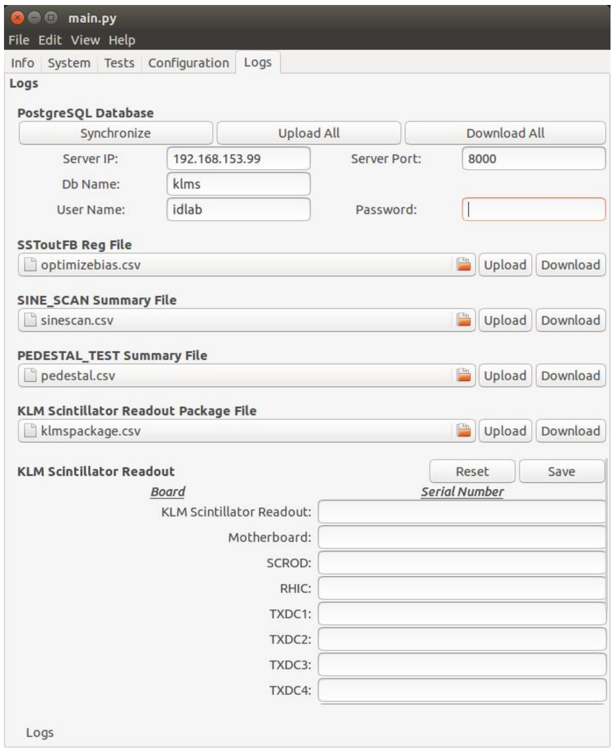
Figure 3.6: KLM Readout Module is serial numbered and barcoded with a virtual serial number that packages the serial numbers each physical hardware component.

With the serial numbering system in place, unique folders for each KLM Readout module and its components could be made by incorporating the serial numbers into the naming scheme. The results could also be easily identified by logging the serial numbers of the hardware being tested. Each module was extensively tested for functionality. For each motherboard, calibrated bias register values for each TARGETX ASIC were measured and entered into a database. The information logged was part of a slow control setup. Systematic tests of each module were done extensively at UH Manoa. Given the number of assembled parts that needed to be tracked and tested, an in-house inventory tracking and management database was implemented using PostgreSQL and Python. Barcode scanners were used to automate the part identification and helped with data entry. The software allowed the user to run automated scripts and interface with the database by navigating through the GUI shown in Figure 3.7. Result logs could easily be uploaded or downloaded to the remote database.



| Variable | Range | Value |
|---|---|---|
| Input | | |
| Function Generator | | |
| IP Address | 192.168.153.### | |
| Port | | |
| Channel | 1 or 2 | |
| Signal | | |
| Amplitude [A] | [1e-03 -> 1) | |
| Frequency [Hz] | [1e-06 -> 160e+06) | |
| Offset [V] | [0 -> 2) | |
| Output Impedance | 50, HighZ, etc | |
| Input BURST | | |
| BURST Mode | OFF or ON | |
| BURST Delay [us] | [0 -> inf) | |
| BURST Cycles | [1 -> inf) | |
| BURST Trigger | INT or EXT | |
| Input for RHIC | | |
| High Voltage | [70 -> 74) | |
| High Voltage Status | OFF or ON | |
| Board Serial Numbers | | |
| KLM Scintillator Readout | KLMS_#### | |
| Motherboard | MB_C#### | |
| SCROD | S_A5#### | |
| RHIC | RHIC_C#### | |
| TARGETX Serial Numbers | | |
| TXDC1 | #### | |
| TXDC2 | #### | |
| TXDC3 | #### | |
| TXDC4 | #### | |

(a) GUI Configurations section



Server IP: 192.168.153.99 Server Port: 8000
Db Name: klms
User Name: idlab Password:

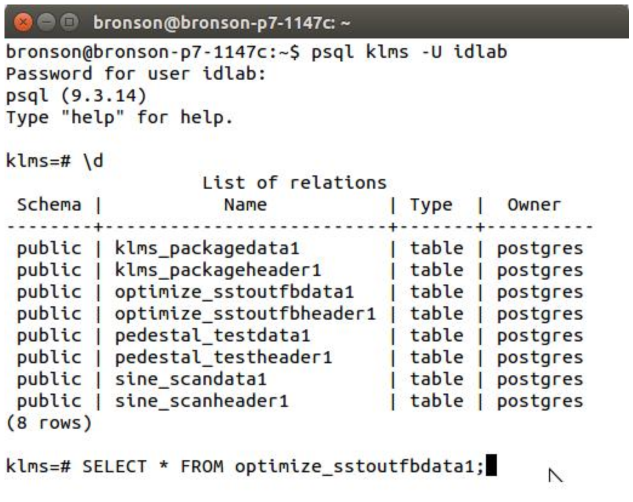
SSToutFB Reg File: optimizebias.csv
SINE_SCAN Summary File: sinescan.csv
PEDESTAL_TEST Summary File: pedestal.csv
KLM Scintillator Readout Package File: klmspackage.csv

(b) GUI Logs section

Figure 3.7: Configurations for the test setup may be altered using the GUI. The remote database may also be accessed through the GUI.

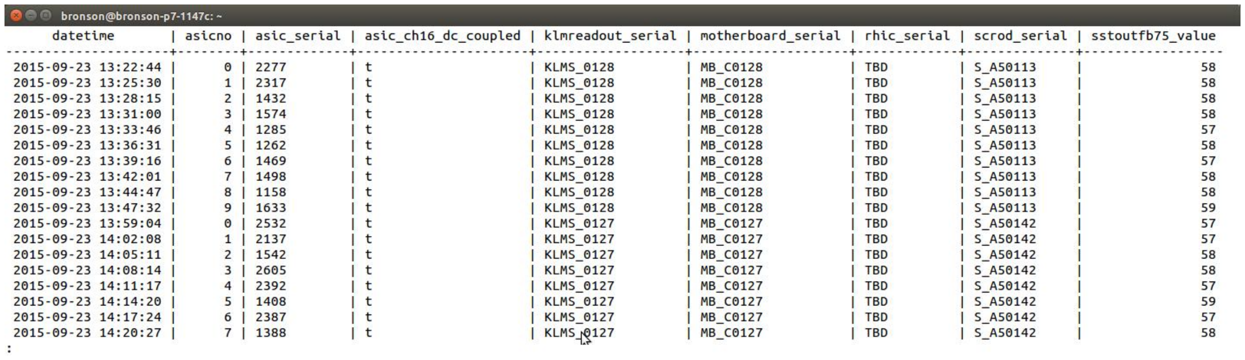
A summary of the results from the "optimize bias," "sine scan," and "pedestal test" are saved onto the PostgreSQL database. Results can either be pulled using the GUI or simply on command line. Data tables created in the PostgreSQL database can be viewed on command line by using the command "\d." A screenshot of the tables can be seen in Figure 3.8.

The calibrated SSToutFB register values, for each TARGETX ASIC, are logged and saved into the database. A sample screenshot of the viewed data on command line can be seen in Figure 3.9.



```
bronson@bronson-p7-1147c:~$ psql klms -U idlab
Password for user idlab:
psql (9.3.14)
Type "help" for help.

klms=# \d
                  List of relations
 Schema |           Name            | Type  |  Owner
--------+---------------------------+-------+----------
 public | klms_packagedata1         | table | postgres
 public | klms_packageheader1       | table | postgres
 public | optimize_sstoutfbdata1    | table | postgres
 public | optimize_sstoutfbheader1  | table | postgres
 public | pedestal_testdata1        | table | postgres
 public | pedestal_testheader1      | table | postgres
 public | sine_scandata1            | table | postgres
 public | sine_scanheader1          | table | postgres
(8 rows)

klms=# SELECT * FROM optimize_sstoutfbdata1;
```

Figure 3.8: Displaying the saved data in the PostgreSQL database using command line.



```
      datetime       | asicno | asic_serial | asic_ch16_dc_coupled | klmreadout_serial | motherboard_serial | rhic_serial | scrod_serial | sstoutfb75_value
---------------------+--------+-------------+----------------------+-------------------+--------------------+-------------+--------------+------------------
 2015-09-23 13:22:44 |      0 | 2277        | t                    | KLMS_0128         | MB_C0128           | TBD         | S_A50113     |                58
 2015-09-23 13:25:30 |      1 | 2317        | t                    | KLMS_0128         | MB_C0128           | TBD         | S_A50113     |                58
 2015-09-23 13:28:15 |      2 | 1432        | t                    | KLMS_0128         | MB_C0128           | TBD         | S_A50113     |                58
 2015-09-23 13:31:00 |      3 | 1574        | t                    | KLMS_0128         | MB_C0128           | TBD         | S_A50113     |                58
 2015-09-23 13:33:46 |      4 | 1285        | t                    | KLMS_0128         | MB_C0128           | TBD         | S_A50113     |                57
 2015-09-23 13:36:31 |      5 | 1262        | t                    | KLMS_0128         | MB_C0128           | TBD         | S_A50113     |                58
 2015-09-23 13:39:16 |      6 | 1469        | t                    | KLMS_0128         | MB_C0128           | TBD         | S_A50113     |                57
 2015-09-23 13:42:01 |      7 | 1498        | t                    | KLMS_0128         | MB_C0128           | TBD         | S_A50113     |                58
 2015-09-23 13:44:47 |      8 | 1158        | t                    | KLMS_0128         | MB_C0128           | TBD         | S_A50113     |                58
 2015-09-23 13:47:32 |      9 | 1633        | t                    | KLMS_0128         | MB_C0128           | TBD         | S_A50113     |                59
 2015-09-23 13:59:04 |      0 | 2532        | t                    | KLMS_0127         | MB_C0127           | TBD         | S_A50142     |                57
 2015-09-23 14:02:08 |      1 | 2137        | t                    | KLMS_0127         | MB_C0127           | TBD         | S_A50142     |                57
 2015-09-23 14:05:11 |      2 | 1542        | t                    | KLMS_0127         | MB_C0127           | TBD         | S_A50142     |                58
 2015-09-23 14:08:14 |      3 | 2605        | t                    | KLMS_0127         | MB_C0127           | TBD         | S_A50142     |                58
 2015-09-23 14:11:17 |      4 | 2392        | t                    | KLMS_0127         | MB_C0127           | TBD         | S_A50142     |                57
 2015-09-23 14:14:20 |      5 | 1408        | t                    | KLMS_0127         | MB_C0127           | TBD         | S_A50142     |                59
 2015-09-23 14:17:24 |      6 | 2387        | t                    | KLMS_0127         | MB_C0127           | TBD         | S_A50142     |                57
 2015-09-23 14:20:27 |      7 | 1388        | t                    | KLMS_0127         | MB_C0127           | TBD         | S_A50142     |                58
:
```

Figure 3.9: Displaying the calibrated SSToutFB register value data in the PostgreSQL database using command line.

# CHAPTER 4
# CHARACTERIZATION OF THE READOUT SYSTEM

## 4.1 ASIC Calibration Results

### 4.1.1 Pedestals

In the current firmware of the FPGA, 4 windows are readout from the ASIC and stitched together to make a waveform. Since there are 32 samples per window, a single waveform will have 128 samples. Because the TARGETX incorporates the Wilkinson ADC architecture for digitization, there is an offset for the digital value called ADC count. Each sample has their very own distribution in mean and standard deviation. The very last sample of each window has an even greater offset but seem to share the same amount of standard deviation as the other samples. These set of raw offset values are called pedestals. In order for us to get a clean waveform, we must compute the average offset per sample and subtract this every time we readout a waveform. Figure 4.1 shows a plot of the pedestals.

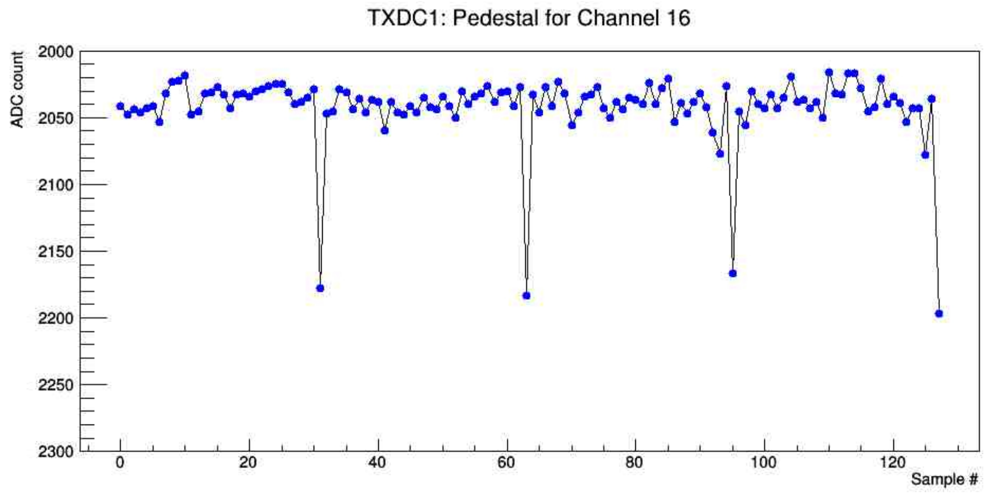


Figure 4.1: Pedestals of a waveform that includes 4 windows.

Subtracting the pedestals per waveform readout is done in different ways depending whether it is DC coupled or AC coupled at the input. DC coupled input means it has a zero ohm resistor in series at the input while an AC coupled input has a capacitor in series at the input. If it is AC coupled at the input, an operating dc bias must be supplied on board. If it is DC coupled at the input, the tester would have to provide a DC bias offset when injecting a signal.

**The algorithm for obtaining the pedestals is the following:**

- AC Coupled Input
    1. Turn OFF function generator
    2. Generate pedestals
    3. Turn ON function generator
    4. Collect data
- DC Coupled Input
    1. Turn ON function generator
    2. Change amplitude to 1mVpp (smallest)
    3. Generate pedestals
    4. Change amplitude back to default amplitude such as 600mVpp
    5. Collect data

### 4.1.2 Linearity

A linearity test was performed to view the usable linear region. The dynamic range of the TARGETX ASIC was expected to be between 1.9V to 2V. Using the results, the transfer function could be extracted. A plot of the results may be seen in Figure 4.2. There was definitely some variation of the slopes between each ASIC. This is probably due to the external 10% tolerance capacitor chosen to assist the voltage ramp circuit within the chip. Using the Wilkinson ADC architecture, the voltage ramp circuit plays a direct role in the digitization of the samples. A voltage ramp with a higher slope output would directly result with an increased slope in the linearity test. The test was performed with data not being pedestal subtracted.

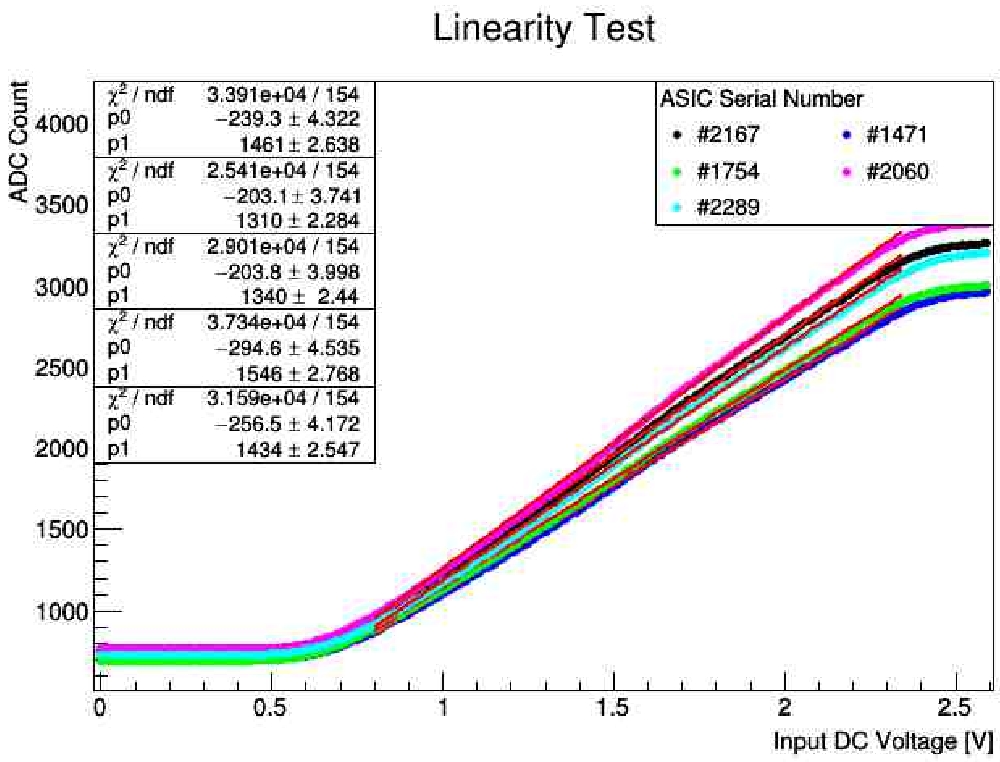


Figure 4.2: Linearity test performed to extract transfer function.

After the linear regime was found, another plot was performed on only the linear region. From these results, the TARGETX dynamic range was found to be approximately 1.6V. Further optimization of some of the registers may be done to improve the dynamic range. For proper characterization, the data was pedestal subtracted with a linear fit performed. The plot may be seen in Figure 4.3.

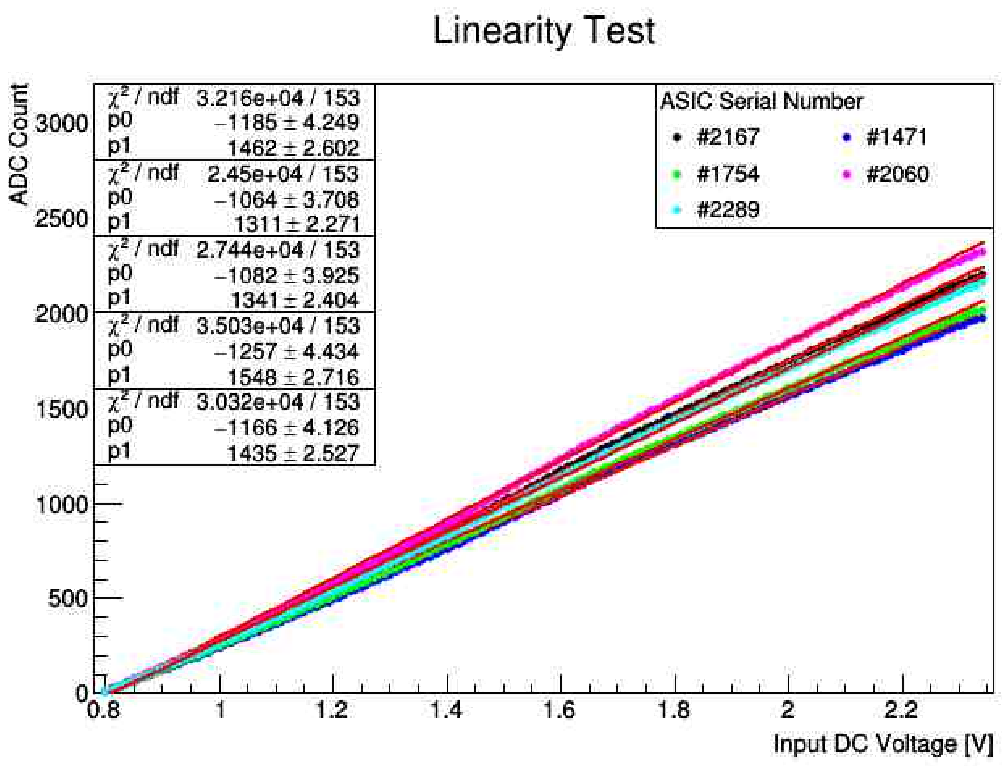


Figure 4.3: The fitted lines are performed only on the linear region of the test.

Using the transfer function found from the test, conversion from ADC Count to Voltage can now be done for the ASIC with serial number #2167 using Equation 4.1.

$$Voltage\ [V] = \frac{ADC\ Count}{1462} \tag{4.1}$$

Residuals were calculated for the linearity test within the linear region using Equation 4.2. Plot of the residuals may be seen in Figure 4.4. It was discovered that residuals for five different ASICs almost directly matched each other. Because of this, a series of sub-fits may be performed on this curve to extract the transfer function for calibrating linearity. If linearity was a concern, the ASICs could be calibrated to have the same slope in the linearity test by adjusting a register called ISEL, which plays a role with the voltage ramp circuit in the chip.

$$residuals = observed - expected \tag{4.2}$$

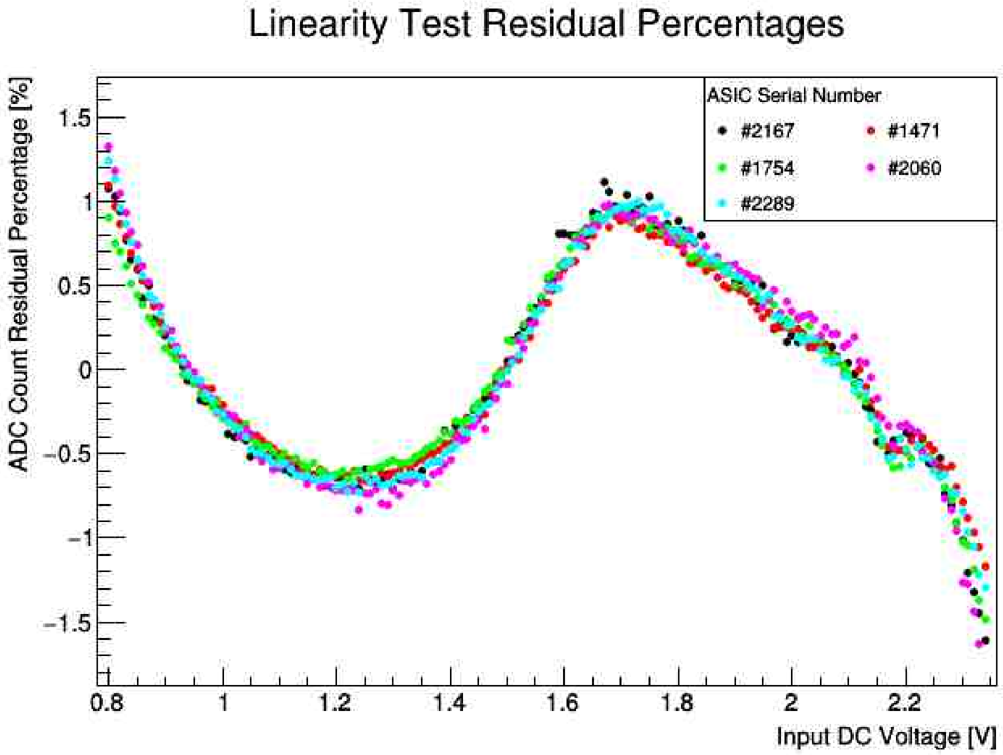


Figure 4.4: The linearity test residual percentages, with respect to the maximum ADC Count usable range, are plotted. Since the results align with each other, it could be used for calibrating the linearity of the ASIC.

### 4.1.3 Noise

In order to perform a Noise Analysis, there must be no signal attached to the input, but there still should be a proper DC input bias voltage called Vped. It's operating dc voltage was discovered by the linearity test seen in Figure 4.3. It is important for Vped to be stable during operation since it plays a role with noise. In performing a readout, the data must be subtracted by the pedestals to examine the actual noise from the system. After performing the analysis, it was discovered that there is roughly 1 mVrms noise for a single channel. In Figure 4.5, a histogram of the results was plotted.

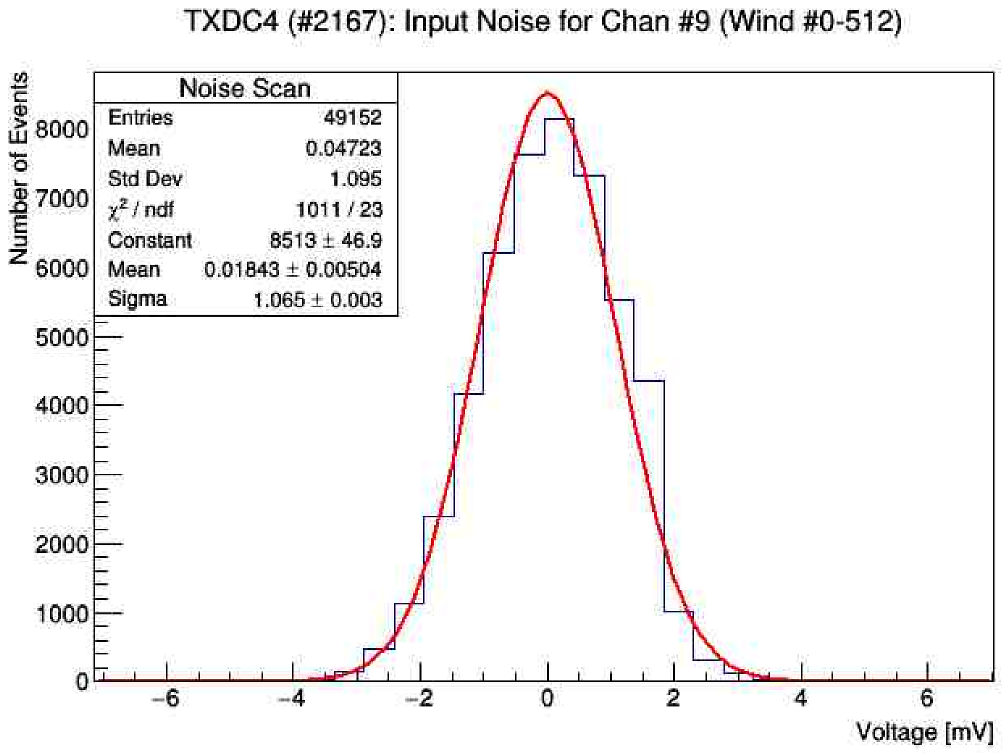


Figure 4.5: Input Noise histogram for a single channel.

Since the waveform is pedestal subtracted, the offset of the input noise should be roughly around zero. In Figure 4.6, an error bar plot is created with the mean and standard deviation per sample. The results show that the standard deviation per sample cell is roughly the same.

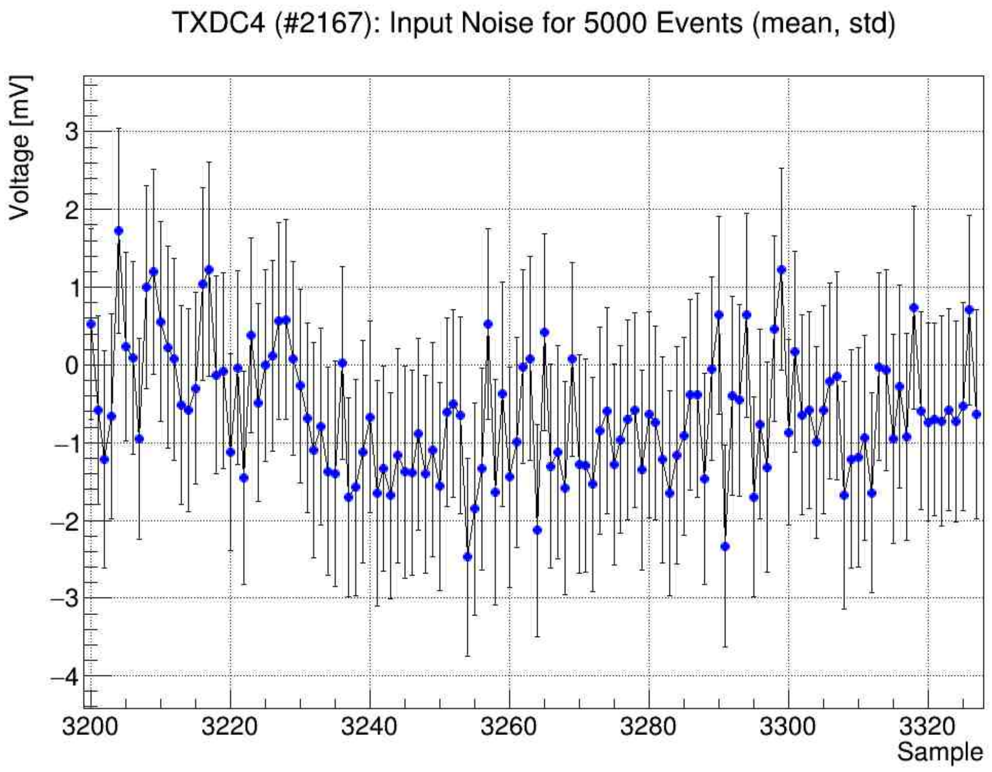


Figure 4.6: 5000 events of pedestal subtracted data for input noise were taken for windows 100 to 103 on a single channel. Each dot represents the mean, and the error bars represent the standard deviation per sample.

### 4.1.4 Waveform Quality Analysis

The waveform quality was quantified by a fit as seen in Figure 4.7.

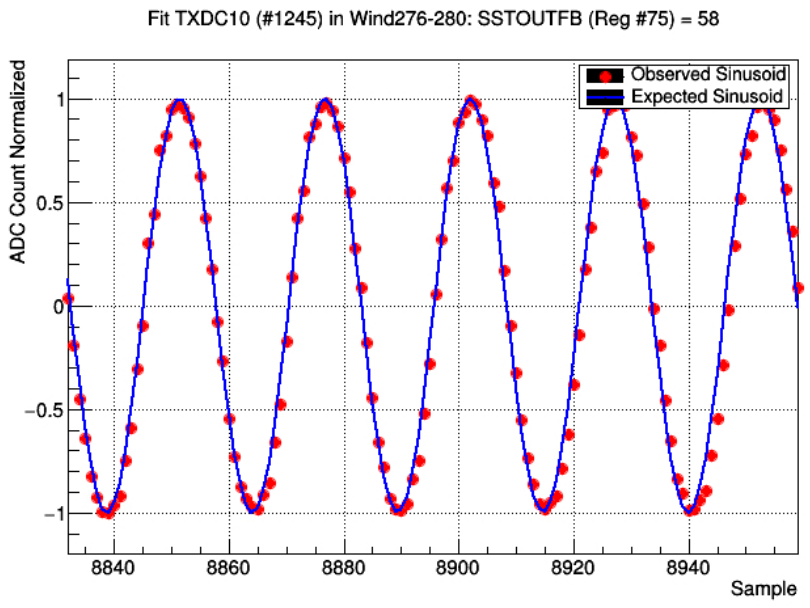
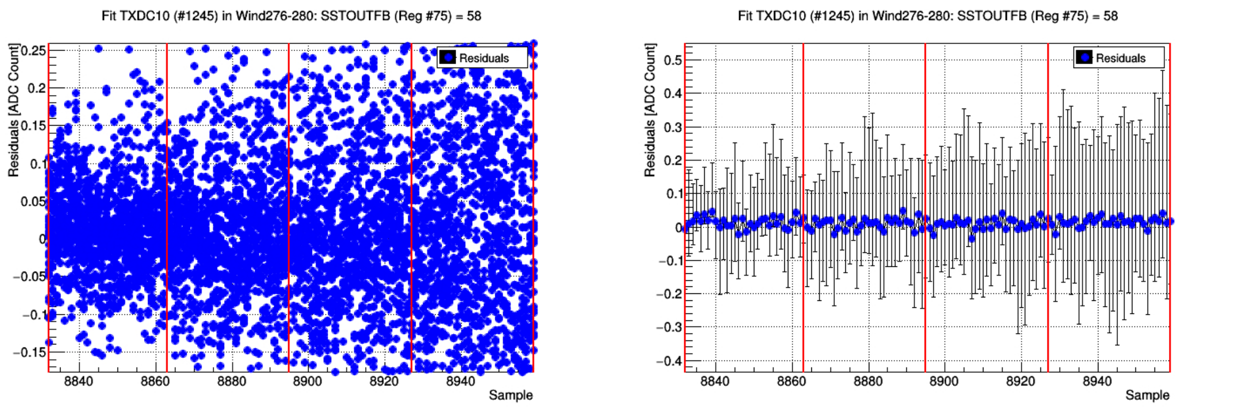


Figure 4.7: A sinusoid fit was performed.

In Figure 4.8a, the raw residuals are plotted by subtracting the observed sample value from the expected value using the fit shown in Figure 4.7. Figure 4.8b shows the errorbar plot of the residuals' mean, min and max. The residual plots give a visual representation on the quality of fit was while the modified chi-test score, from Equation 2.2, quantifies it.



(a) Residuals plot.

(b) Residuals error bar plot with mean, min and max.

Figure 4.8: Residuals plot was used to take a closer look at the quality of the fit.

### 4.1.5 Timing Resolution

Using the Zero Crossing Algorithm quantified by Equation 4.3, the period may be determined for a sinusoid waveform as shown in Figure 4.9.

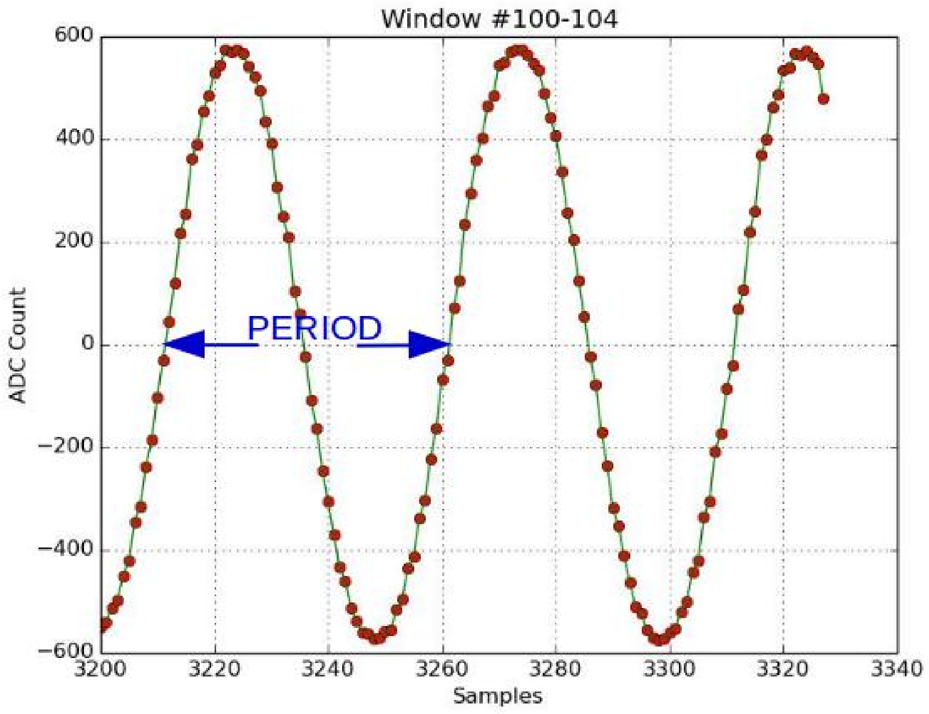


Figure 4.9: Use Zero Crossing Algorithm Equation 4.3 to help calculate the period of the sine waveform.

$$t_{zero} = t_1 + \frac{|A_1|}{|A1| + |A_2|}(t_2 - t_1) \tag{4.3}$$

where:

$t_{zero}$: zero crossing time value

$t_1$: is the first time value

$A_1$: is the voltage value of the first time value

$t_2$: is the second time value

$A_2$: is the voltage value of the second time value

After taking 9818 period measurements of a 30MHz signal, each measurement was subtracted by the expected $Period = \frac{1}{30MHz} = 33.33ns$. Shown in Figure 4.10, the measurements are plotted as a histogram. From the histogram, you can clearly see a peak forming on the right of the main peak that is centered around zero. There may also be a peak forming on the left, which starts to combine with the distribution in the middle. It became clear that the 180ps RMS timing between samples can be improved.

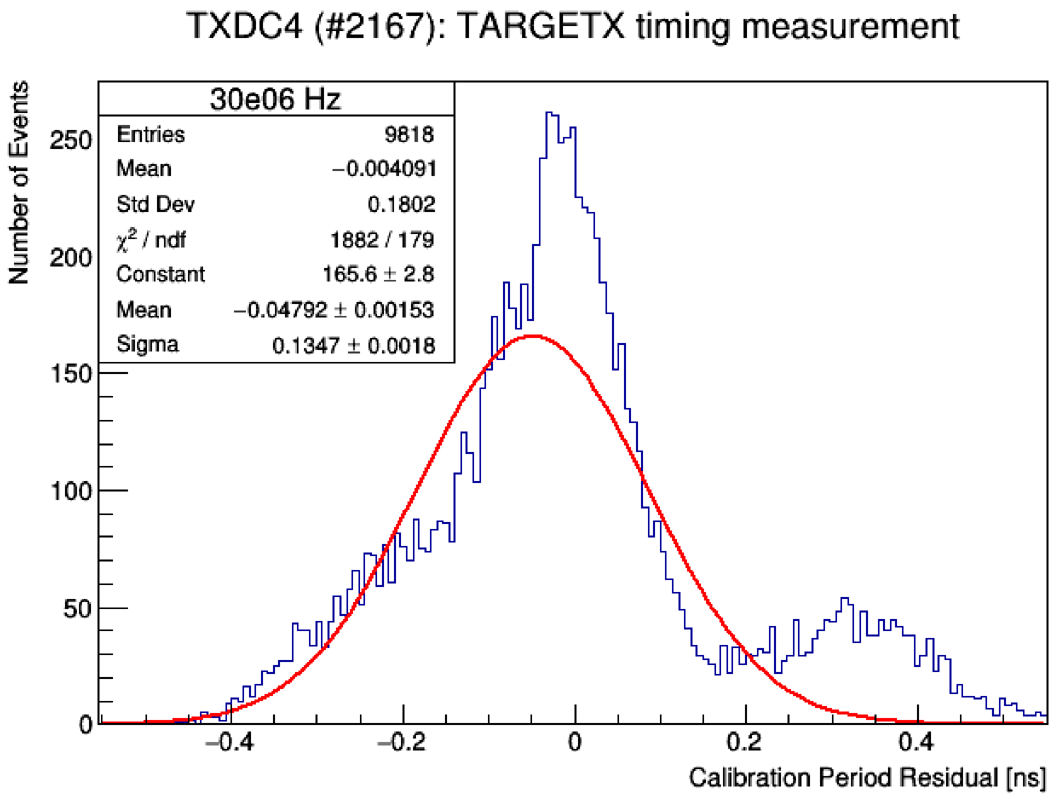
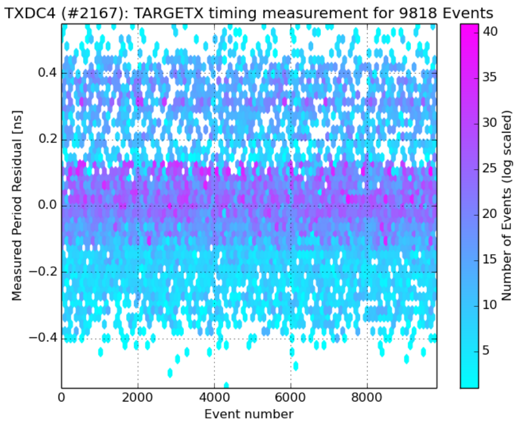
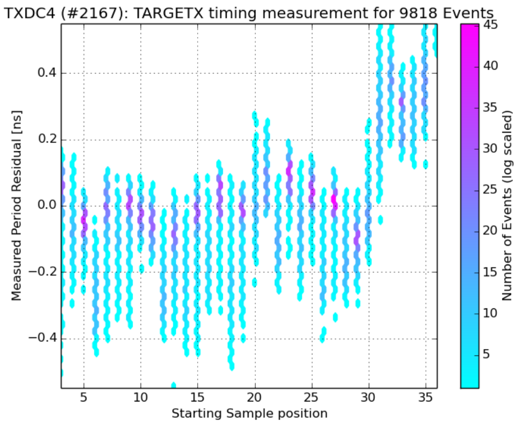


Figure 4.10: Before timing corrections, we measured roughly 180ps timing resolution.

In Figure 4.11a, it became apparent that there were distributions forming with different means or averages. Further investigation was needed to be taken to understand why this was happening. After plotting the period residuals vs the starting position of each signal where the period was calculated from, it became clear that there was a dependence on starting position. The period residuals vs the starting position plot may be seen in Figure 4.11b.



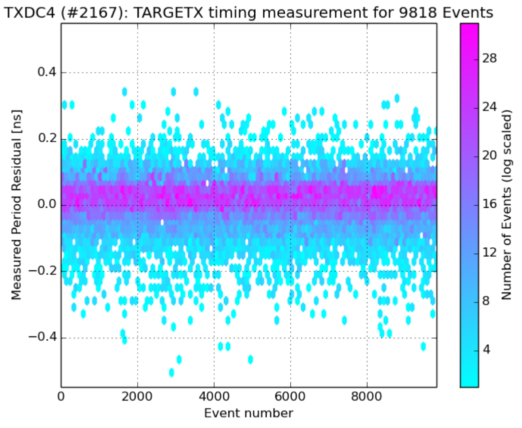
(a) Period Residuals vs Event Number.



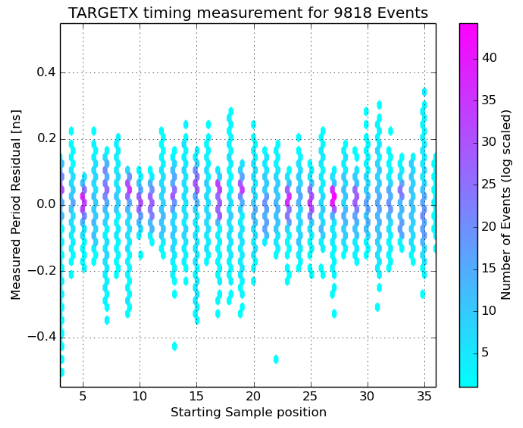
(b) Period Residuals vs Starting Position.

Figure 4.11: Before timing corrections, measured period residuals were plotted vs event number and starting position.

Each period measurement were binned according to the starting position time. The averages were calculated for each set of bins. Then, each bins of measurements were subtracted by their own set of averages. After performing this correction, the period residuals for all the measurements were now converging towards zero. The plots may be seen in Figure 4.12.



(a) Period Residuals vs Event Number.



(b) Period Residuals vs Starting Position.

Figure 4.12: After timing corrections, measured period residuals were plotted vs event number and starting position.

With the timing corrections made, less than 100ps timing error was realized per sample. These tests were only done to see what kind of timing error can be achieved. For our application, timing calibrations were not needed since the error was within our specifications.

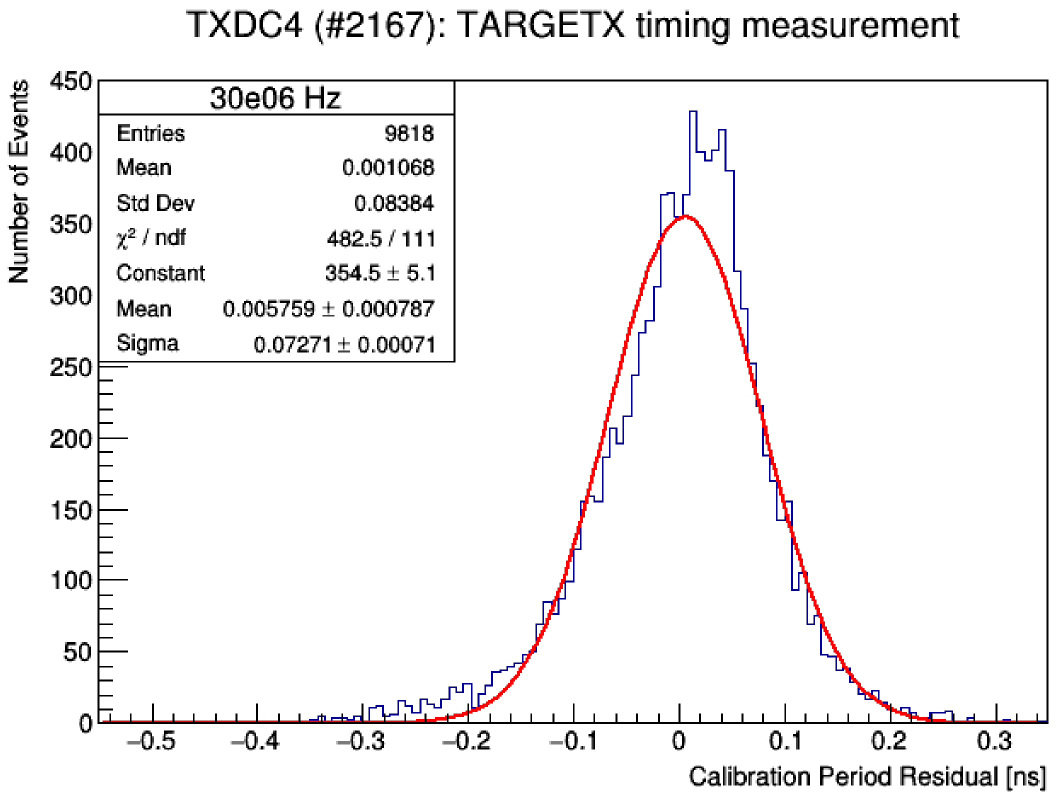


Figure 4.13: After timing corrections, 84ps timing resolution was measured.

## 4.2 Production Test Yield

From the Motherboard Production test, the “pedestal scan” will plot all the pedestals from each storage cell for all 16 channels on the chip. Shorts and unexpected pedestal offsets may be viewed. Single sample offsets would be passed but a burst of sample offsets in a row would not be passed. Figure 4.14 is an example of a plot where the ASIC chip would be passed. If there was a burst of pedestal offsets, then it would have been failed.

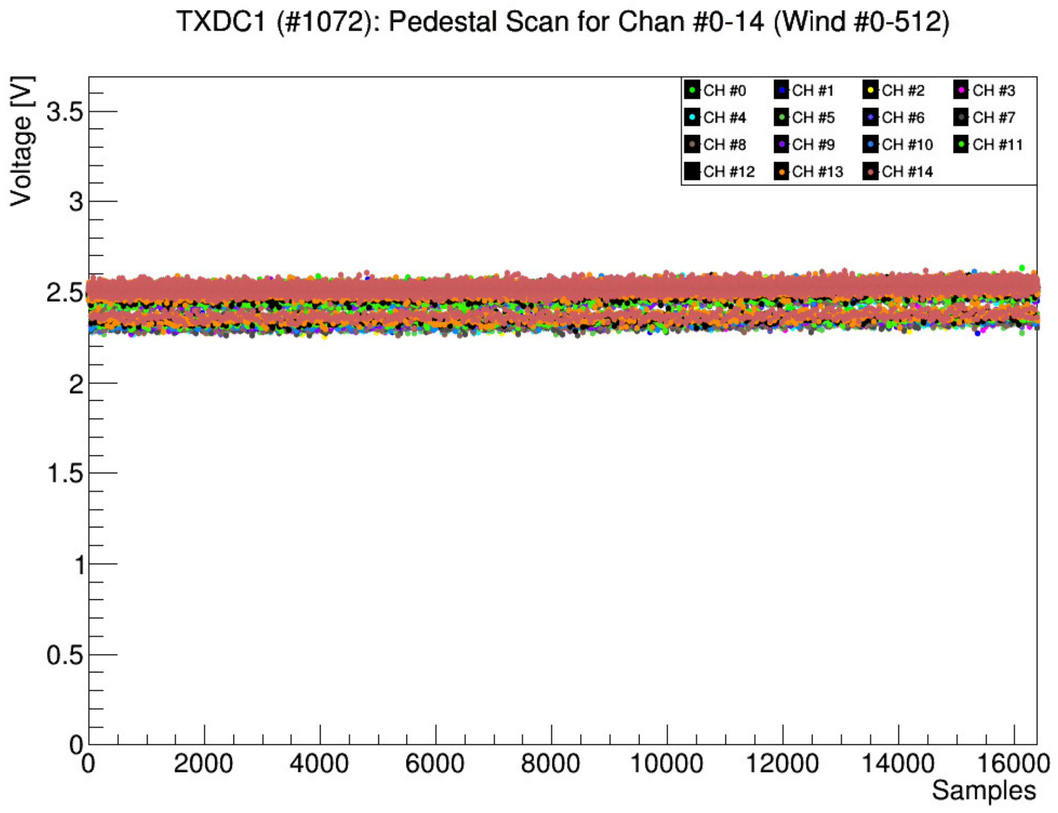
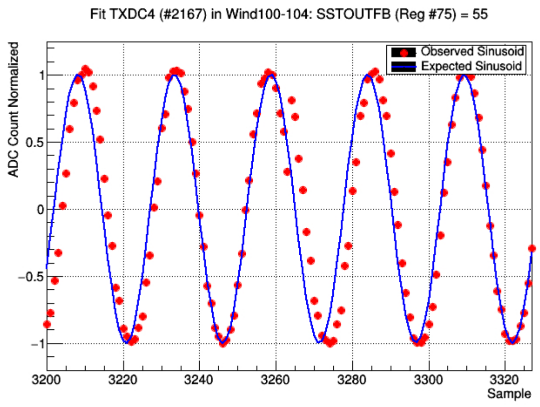
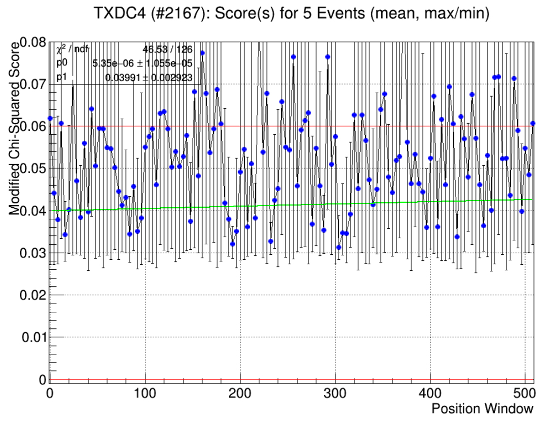


Figure 4.14: This particular test was used to look for shorts and unexpected pedestal offsets.

For each TARGETX, the SSToutFB register was optimized. Then, a “sine scan” was performed for all 512 windows of the chip. During this scan, the modified Chi-Squared test scores were calculated using Equation 2.2 to quantify the quality of the fit for each waveform. Then an error bar plot of the mean, min, and max of the scores were plotted. Figure 4.15 shows an example where the TARGETX did not pass the test.



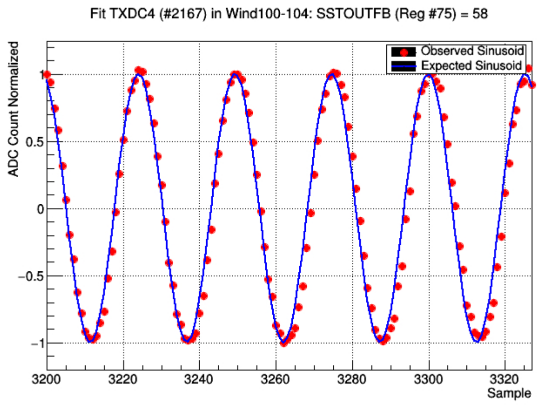
(a) Sinusoidal fit of the waveform.



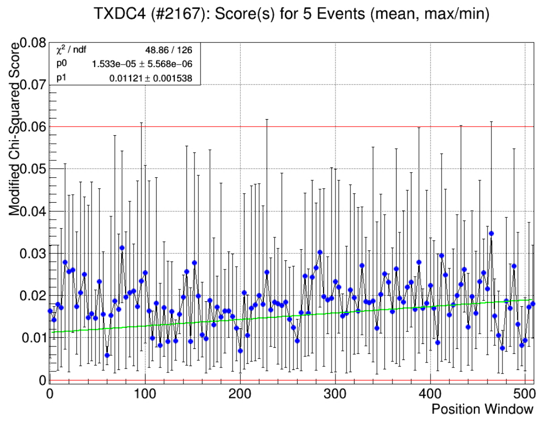
(b) Sine scan test.

Figure 4.15: Example of where optimization of the SSToutFB timing parameter failed.

After a failed "sine scan" occurs, the "optimize bias" test may be ran again. If this works, the result will be seen in Figure 4.16. If it continues to fail, this means that the chip would have to be thrown out. This does not necessarily mean that the chip is bad but it probably requires a more extensive optimization of its other register values, which is preferred to be avoided given the time required and the added operational complexity. Optimizing all the registers for each chip would be inefficient and the tracking system would have more register values to be keep track of.



(a) Sinusoidal fit of the waveform.



(b) Sine scan test.

Figure 4.16: Example of where optimization of the SSToutFB timing parameter passed.

An example of results from the RHIC Production tests are shown below for "trigger scan" where it verifies a trigger from all 10 TARGETX daughtercards (TXDC) on a motherboard. When a "trigger scan" is performed, the SCROD will command each TXDC to trigger on different thresholds. The SCROD will keep track of each trigger per TXDC channel by counting the number of triggers per threshold. Then, it would relay the information to the PC. These values were plotted on a heat-map shown in Figure 4.17. From this plot, it is unclear whether or not the "trigger scan" was successful.

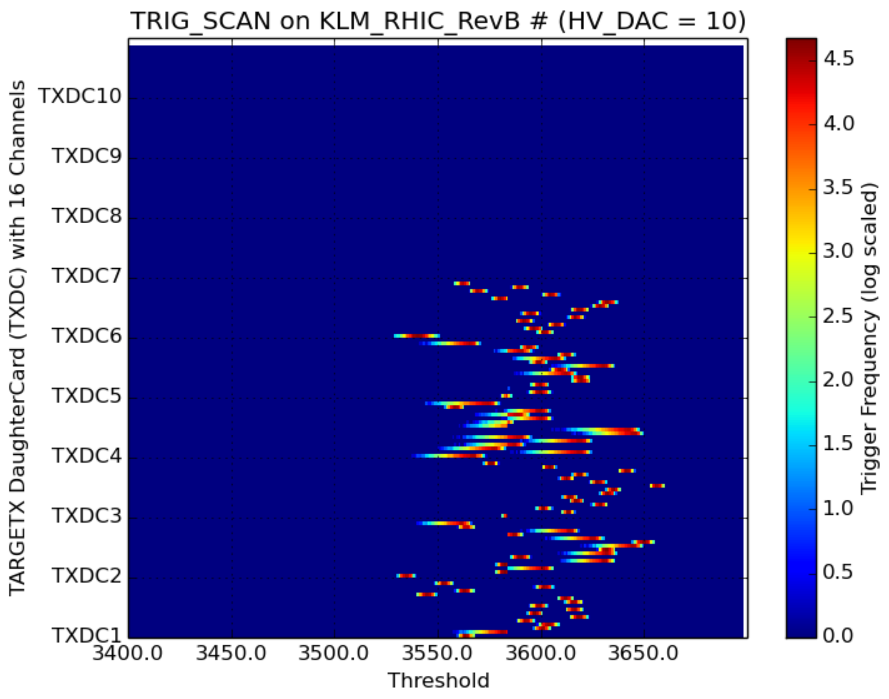


Figure 4.17: Before corrections, this is a plot of the trigger scan.

For each channel, there are different threshold offsets. Each offset was subtracted so the triggers could be properly viewed and evaluated. From Figure 4.18, triggers from TXDC #1 - 6 were only seen. There are many reasons why triggers were not seen but in this case it was because TXDC #7 - 10 slots did not have any TXDC cards on there. Other reasons for why a "trigger scan" may fail is the possibility of shorts on the RHIC board or possible problems with the ribbon cables. In this particular test for Figure 4.18, TXDC #1, 3, 6 were missing ribbon cables. This is why it was only triggering within the noise and nothing else. With proper debugging, corrections to the problem may be made. Unfortunately, if problems keep persisting, then the TXDC may have to be replaced with a new one that will need to be tested extensively using the Motherboard Production test before running anymore RHIC Production tests. If it still does not work, then the problem could be with the RHIC board itself.

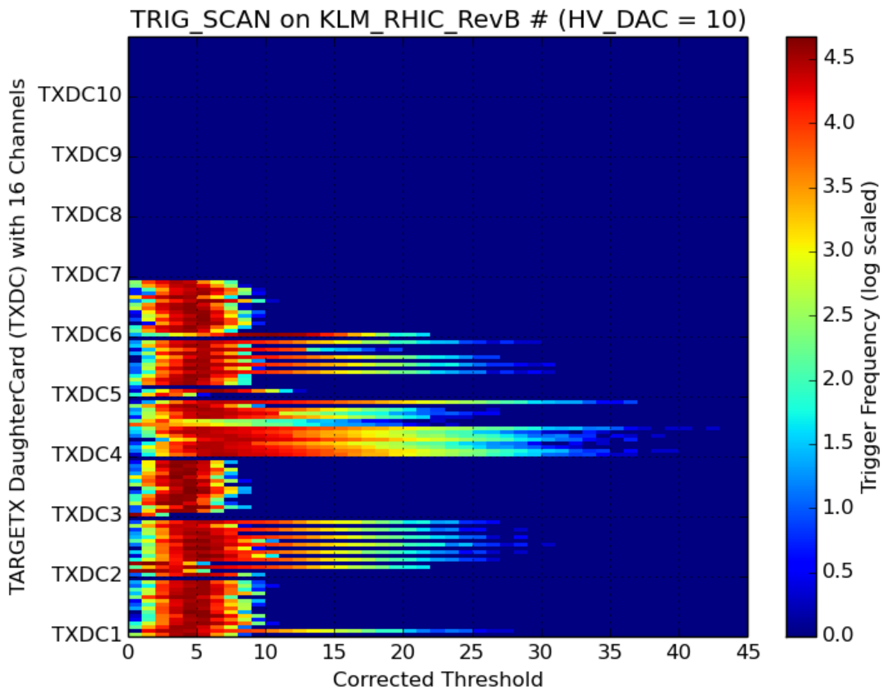


Figure 4.18: After corrections, we can more accurately view the results of the trigger scan.

After production testing for the 20,000 channel single-photon, sub-nanosecond electronic readout for a large area muon detector, the yield results summary can be seen in Table 4.1. Since the RHIC board was mainly an interconnect board, it was not surprising that the yield for that particular board would be the highest. Failures for the RHIC boards and Motherboards were mainly due to shorts. Some of the SCRODs failed because it was not able to be programmed. The TARGETX ASICs mainly failed because more extensive optimization of the registers were needed compared to only optimizing the SSToutFB timing register.

Table 4.1: Production Test Yield Summary.

| Board | Pass | Fail | Pass Percentage |
|---|---|---|---|
| SCROD | 156 | 13 | 91.66% |
| Motherboard | 156 | 9 | 94.23% |
| TARGETX ASIC | 1464 | 108 | 92.62% |
| RHIC | 156 | 4 | 97.43% |

# CHAPTER 5
# CONCLUSION

Hardware verification and testing for a 20,000 channel single-photon sub-nanosecond electronic readout for a large area muon detector required much development of automation software. An ethernet driver was developed to allow communication between a data logging PC and the front-end System Control and Readout Module (SCROD). In order for the testing to be automated, scripts and libraries needed to be created for the instrument control module, which remotely controlled the Rigol DG4162 Waveform Function generator during testing. Before production testing, the TARGETX ASIC's registers were calibrated using automated routines that was developed using signal processing techniques. Characterization test scripts were developed and used extensively during production testing. Due to time constraints for testing, parallel processing was also implemented. To increase productivity of the testers and allow ease of training, a GUI was created. An in-house inventory tracking and management database was implemented using a PostgreSQL database and Python. Serial numbers and barcode scanners were used to automate the part identification and helped greatly with data entry. Summary results from the production tests were also logged in a remote database.

With the success of verifying all the electronics for the KLM sub-detector of the Belle II detector for the upgraded SuperKEKB particle accelerator in Japan, the electronics were installed in Fall 2016.

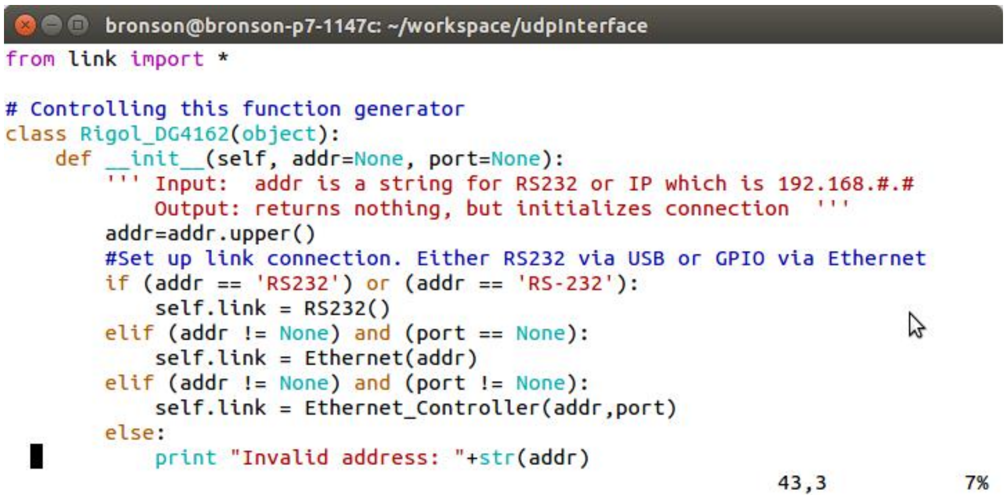
# APPENDIX A
# SAMPLE CODE SNIPPETS

In this chapter, some snapshots of the software were taken to explain parts of it. The KLM production test software (Release 10/12/15) may be downloaded at the following url:

`http://www.phys.hawaii.edu/~idlab/bronson/`

## A.1 Instrument Control

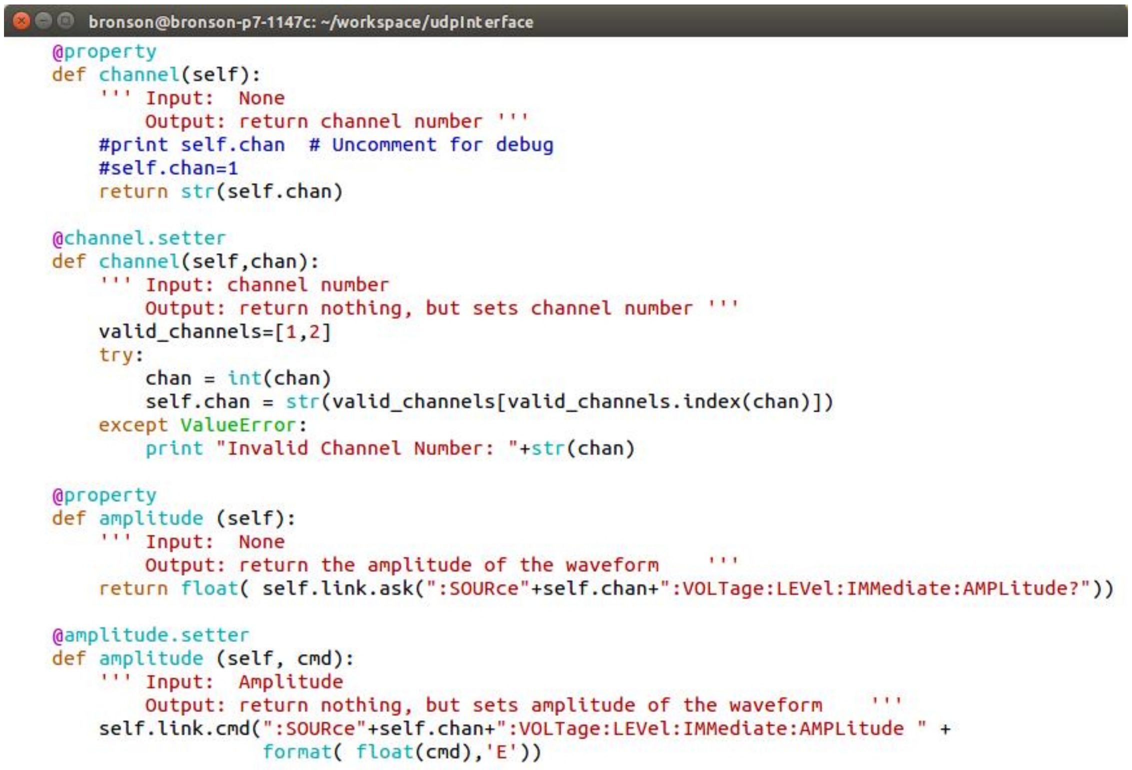
For the purpose of instrument control, Ethernet and USB drivers were implemented in Python on a file called "link.py". Modern-day instruments such as function generators, oscilloscopes and power-supplies have firmware programmed to accept a standardized set of commands called Standard Commands for Programmable Instruments (SCPI). Using Python, a library was built incorporating some of these commands related to controlling the Rigol DG4162 Waveform Function Generator. The Python file called "rigol_DG4162.py" includes the built "class" for the library. Ethernet or USB can be chosen as the communication link when you create an instance of the class. A screenshot of the initialization part of the class is shown in Figure A.1.



```
bronson@bronson-p7-1147c: ~/workspace/udpInterface
from link import *

# Controlling this function generator
class Rigol_DG4162(object):
    def __init__(self, addr=None, port=None):
        ''' Input:  addr is a string for RS232 or IP which is 192.168.#.#
            Output: returns nothing, but initializes connection  '''
        addr=addr.upper()
        #Set up link connection. Either RS232 via USB or GPIO via Ethernet
        if (addr == 'RS232') or (addr == 'RS-232'):
            self.link = RS232()
        elif (addr != None) and (port == None):
            self.link = Ethernet(addr)
        elif (addr != None) and (port != None):
            self.link = Ethernet_Controller(addr,port)
        else:
            print "Invalid address: "+str(addr)
                                                              43,3          7%
```

Figure A.1: Source code for the initialization part of the Rigol DG4162 Waveform Function Generator library class.

In Figure A.2, a screenshot is shown of the body part of the library class for the function generator. An object oriented programming technique called "Getters and Setters" was used to create methods for the commands. This technique will promote more readability in the software when the code at the higher-level starts using them.



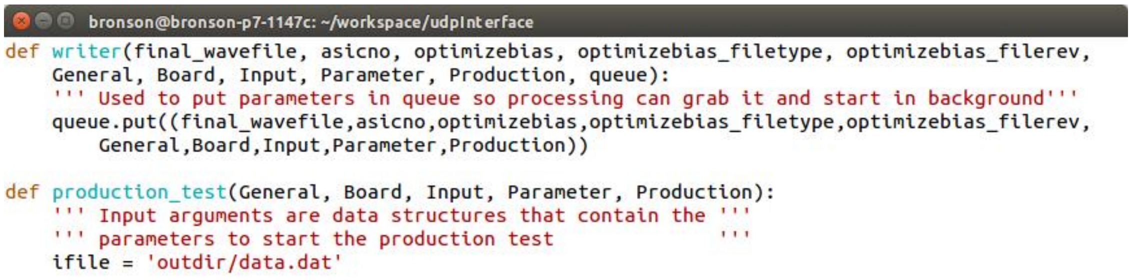
bronson@bronson-p7-1147c: ~/workspace/udpInterface

```
    @property
    def channel(self):
        ''' Input:  None
            Output: return channel number '''
        #print self.chan  # Uncomment for debug
        #self.chan=1
        return str(self.chan)

    @channel.setter
    def channel(self,chan):
        ''' Input: channel number
            Output: return nothing, but sets channel number '''
        valid_channels=[1,2]
        try:
            chan = int(chan)
            self.chan = str(valid_channels[valid_channels.index(chan)])
        except ValueError:
            print "Invalid Channel Number: "+str(chan)

    @property
    def amplitude (self):
        ''' Input:  None
            Output: return the amplitude of the waveform    '''
        return float( self.link.ask(":SOURce"+self.chan+":VOLTage:LEVel:IMMediate:AMPLitude?"))

    @amplitude.setter
    def amplitude (self, cmd):
        ''' Input:  Amplitude
            Output: return nothing, but sets amplitude of the waveform    '''
        self.link.cmd(":SOURce"+self.chan+":VOLTage:LEVel:IMMediate:AMPLitude " +
                      format( float(cmd),'E'))
```

Figure A.2: Source code for the body part of the Rigol DG4162 Waveform Function Generator library class.

## A.2 Parallel Processing

To decrease the run-time of the tests, parallel processing was introduced into the software. It was designed in a way where processing could be done in the background while the main process was busy collecting data from the SCROD. After data collection was done for a test, the software will place a series of parameters into the queue by using a writer function as shown in the source code of Figure A.3.

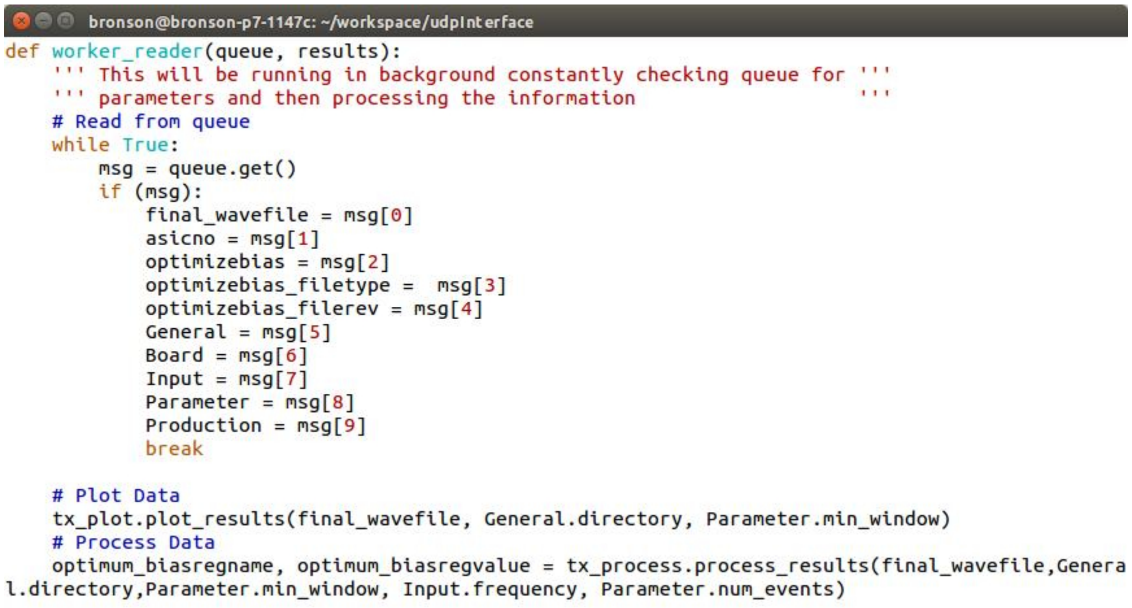


bronson@bronson-p7-1147c: ~/workspace/udpInterface

```
def writer(final_wavefile, asicno, optimizebias, optimizebias_filetype, optimizebias_filerev,
    General, Board, Input, Parameter, Production, queue):
    ''' Used to put parameters in queue so processing can grab it and start in background'''
    queue.put((final_wavefile,asicno,optimizebias,optimizebias_filetype,optimizebias_filerev,
        General,Board,Input,Parameter,Production))

def production_test(General, Board, Input, Parameter, Production):
    ''' Input arguments are data structures that contain the '''
    ''' parameters to start the production test             '''
    ifile = 'outdir/data.dat'
```

Figure A.3: Source code for the writer part of the parallel processing script.

A subprocess running in the background is called a worker or reader that constantly checks the queue. Whenever a set of parameters is placed into the queue, the reader will grab the information and use it to start processing data for the particular test. A sample screenshot of the reader can be seen in Figure A.4.

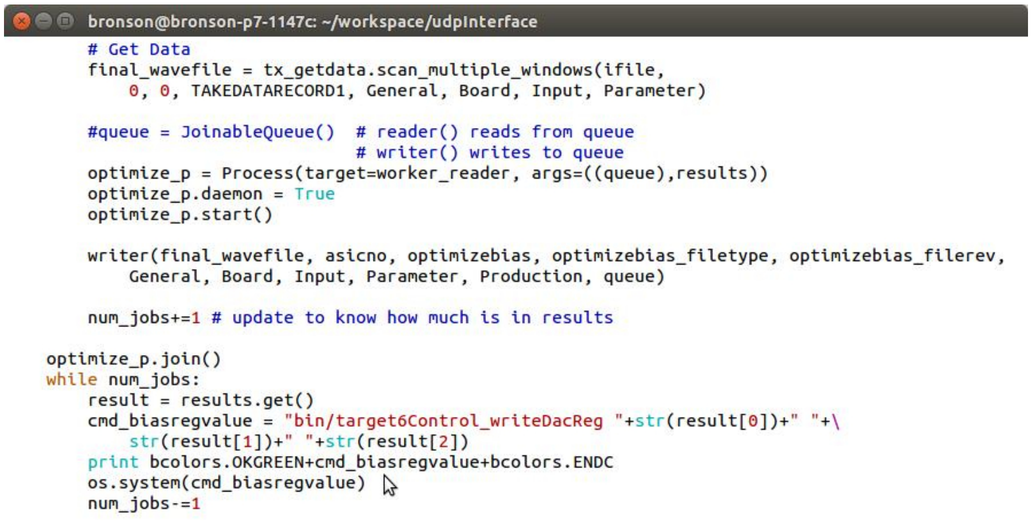


bronson@bronson-p7-1147c: ~/workspace/udpInterface

```
def worker_reader(queue, results):
    ''' This will be running in background constantly checking queue for '''
    ''' parameters and then processing the information                   '''
    # Read from queue
    while True:
        msg = queue.get()
        if (msg):
            final_wavefile = msg[0]
            asicno = msg[1]
            optimizebias = msg[2]
            optimizebias_filetype =  msg[3]
            optimizebias_filerev = msg[4]
            General = msg[5]
            Board = msg[6]
            Input = msg[7]
            Parameter = msg[8]
            Production = msg[9]
            break

    # Plot Data
    tx_plot.plot_results(final_wavefile, General.directory, Parameter.min_window)
    # Process Data
    optimum_biasregname, optimum_biasregvalue = tx_process.process_results(final_wavefile,Genera
l.directory,Parameter.min_window, Input.frequency, Parameter.num_events)
```

Figure A.4: Source code for the reader part of the parallel processing script.

In Figure A.5, a sample screenshot is in the body of the production script that uses the writer and reader together. Before the script moves onto another test, it will use the command ".join()", which tells the main process to wait until all subprocesses running in the background are done.



```
bronson@bronson-p7-1147c: ~/workspace/udpInterface
    # Get Data
    final_wavefile = tx_getdata.scan_multiple_windows(ifile,
        0, 0, TAKEDATARECORD1, General, Board, Input, Parameter)

    #queue = JoinableQueue()  # reader() reads from queue
                              # writer() writes to queue
    optimize_p = Process(target=worker_reader, args=((queue),results))
    optimize_p.daemon = True
    optimize_p.start()

    writer(final_wavefile, asicno, optimizebias, optimizebias_filetype, optimizebias_filerev,
        General, Board, Input, Parameter, Production, queue)

    num_jobs+=1 # update to know how much is in results

optimize_p.join()
while num_jobs:
    result = results.get()
    cmd_biasregvalue = "bin/target6Control_writeDacReg "+str(result[0])+" "+\
        str(result[1])+" "+str(result[2])
    print bcolors.OKGREEN+cmd_biasregvalue+bcolors.ENDC
    os.system(cmd_biasregvalue)
    num_jobs-=1
```

Figure A.5: Source code for the body part of the parallel processing script.

# APPENDIX B
# REGISTER MAP WITH OPTIMIZED VALUES

Table B.1: Register Map for TARGETX ASIC with register values optimized for production.

| Register Name | Register Number | Register Value |
|---|---|---|
| Sbbias | 48 | 1300 |
| Vdischarge | 49 | 0 |
| Isel | 50 | 2650 |
| Dbbias | 51 | 1100 |
| Qbias | 52 | 1500 |
| Vqbuf | 53 | 1062 |
| VtrimT | 54 | 1209 |
| VadjP | 56 | 1152 |
| VAPbuff | 57 | 0 |
| VadjN | 58 | 2235 |
| VANbuff | 59 | 0 |
| Vbias | 61 | 1130 |
| TRGGbias | 62 | 1100 |
| Itbias | 63 | 1100 |
| SSPin LE | 64 | 51 |
| SSPin TE | 65 | 7 |
| WR_ADDR_INCR1 LE | 66 | 5 |
| WR_ADDR_INCR1 TE | 67 | 25 |
| WR_STRB1 LE | 68 | 20 |
| WR_STRB1 TE | 69 | 40 |
| WR_ADDR_INCR2 LE | 70 | 33 |
| WR_ADDR_INCR2 TE | 71 | 53 |
| WR_STRB2 LE | 72 | 56 |
| WR_STRB2 TE | 73 | 12 |
| MonTiming SEL | 74 | 40 |
| SSToutFB | 75 | 58 |
| CMPbias2 | 76 | 737 |
| Pubias | 77 | 3112 |
| CMPbias | 78 | 1152 |
| TPGreg | 79 | 2730 |

# APPENDIX C
# LIST OF ACRONYMS AND ABBREVIATIONS

- ADC - Analog to Digital Converter
- ASIC - Application Specific Integrated Circuit
- DHCP - Dynamic Host Configuration Protocol
- GUI - Graphical User Interface
- HEP - High Energy Physics
- IDLab - Instrumentation Development Laboratory
- IP - Internet Protocol
- iTOP - Imaging Time of Propagation detector
- KEK - High Energy Accelerator Research Organization in Tsukuba, Ibaraki Prefecture, Japan
- KLM - Kaon long and Muon detector
- LAPPD - Large Area Picosecond Photo Detector
- MB - Motherboard
- MCP - Micro-Channel Plate
- MOSFET - Metal-Oxide-Semiconductor Field Effect Transistor
- MPPC - Multi-Pixel Photon Counters
- OS - Operating System
- PID - Particle Identification
- PMT - Photo-multiplier tube
- RHIC - Ribbon Header Interface Card
- RPC - Remote Procedure Call
- RF - Radio Frequency
- SCPI - Standard Commands for Programmable Instruments

- SCROD - Standard Control and Readout Device
- SQL - Structured Query Language
- SSH - Secure Shell
- TARGET - TeV Array Readout GSa/s Electronics with Trigger
- TARGETX - Generation X in TARGET series, KLM Readout ASIC
- TCP - Transmission Control Protocol
- TSMC - Taiwan Semiconductor Manufacturing Company, Limited
- TXDC - TARGETX Daughtercard
- UDP - User Datagram Protocol
- UH Manoa - University of Hawaii at Manoa
- XML - Extensible Markup Language

# BIBLIOGRAPHY

[1] Belle-II Collaboration (Abe, T. et al.) arXiv:1011.0352 [physics.ins-det]. *Belle II Technical Design Report.* Technical report, KEK, 2010.

[2] I. Mostafanezhad, G. Varner, B. Edralin. *Production and testing of a high-performance, low-cost readout system for the Belle II upgrade: KL and Muon (KLM) scintillator sub-detector*, Oct 2016.

[3] M. Barrett. *Particle identification with the iTOP detector at Belle-II.* Technical report, Belle-II iTOP group, 2013.

[4] Belle II Collaboration. *Experiment Belle II at the SuperKEKB Accelerator*, 2015.

[5] Belle-II Collaboration (D. M. Asner, et al.). *US Belle II Project Technical Design Report*, 2016. [Online]. Available: `http://belleweb.pnnl.gov/forTDRreview/TDR-SLAC-13Dec8.pdf`.

[6] Hamamatsu. *MPPC Module*, 2016. [Online]. Available: `https://www.hamamatsu.com/resources/pdf/ssd/mppc_kapd9003e.pdf`.

[7] G. Varner, B. Edralin, I. Mostafanezhad. *The TARGETX ASIC for the Belle II Muon Detector Scintillator Upgrade*, Nov 2015.

[8] Instrumentation Development Laboratory. *TARGETX 16-channel, GSPS Transient Waveform Recorder with Self-Triggering and Fast, Selective Window Readout.* Physics and Astronomy Department, University of Hawaii, May 2015.

[9] I. Mostafanezhad. *KLM-CRT Data logging.* Belle II Summer School at PNNL, August 2015.

[10] I. Mostafanezhad. *Overview of KLM Electronics.* Belle II Summer School at PNNL, August 2015.