
Imperial College London

Research and Development for the Data, Trigger and Control Card for CMS in Preparation for Hi-Lumi LHC

David Monk

2022

Department of Physics
Imperial College London

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy of Imperial College London
and the Diploma of Imperial College London



Abstract

When the Large Hadron Collider (LHC) increases its luminosity by an order of magnitude in the coming decade, the experiments that sit upon it must also be upgraded to continue to their physics performance in the increasingly demanding environment. To achieve this, the Compact Muon Solenoid (CMS) experiment will make use of tracking information in the Level-1 trigger for the first time, meaning that track reconstruction must be achieved in less than $4\text{ }\mu\text{s}$ in an all-FPGA architecture.

MUonE is an experiment aiming to make an accurate measurement of the the hadronic contribution to the anomalous magnetic moment of the muon. It will achieve this by making use of similar apparatus to that designed for CMS and benefit from the research and development efforts there.

This thesis presents both development and testing work for the readout chain from tracker module to back-end processing card, as well as the results and analysis of a beam test used to validate this chain for both CMS and the MUonE experiment.

Acknowledgements

I would like to thank the following people for their support over the duration of my PhD:

- **Andrew Rose:** It was through you that I was first given the opportunity to work in this field and I have not looked back since. Thank you for always supporting me and guiding me from start to end.
- **Mark Pesaresi:** I would not have been able to achieve all that I have without your help and guidance during my time at CERN. You have always been there to answer any query I had. Thank you.
- **Greg Iles:** I was fortunate enough to be able to share an office with you during my time at CERN, and you were always more than happy to answer a question or offer advice on any subject.
- **Trevor, Debbie and Sarah Monk:** You have always been the inspiration for me to aim as high as possible and achieve those goals, both in academia and outside.

I'd also like to thank the following people who have provided a helping hand throughout the last 3 years: **Kristian Hahn, Sarah Seif El Nasr-Storey, Tom Williams, Kevin Lannon, Giacomo Fedi, Sioni Summers, Tom James, Kirika Uchida.**

Statement of Originality

I declare that the work presented here is my own, with the work of others explicitly referenced.

During the period of this PhD I was responsible for the following:

- The development of the DTC firmware, in particular the link decoding and stub coordinate formatting, its testing in all forms, including with the use of genuine front-end hardware.
- The Shep layer of the CMS Tracker Online Software and the integration and testing of the HERD layer on both dummy and prototype hardware. The investigation and management of the test Kubernetes cluster located in the TIF at CERN. Furthermore the use of this cluster for firmware builds and the accompanying software development required to enable CI for such builds.
- The development and testing of MUonE-specific firmware and its integration with existing DTC firmware for use in a beam test.
- The online software required to configure and monitor the DAQ system during the MUonE beam test.
- The analysis of the data collected in the beam test, with a particular focus on track finding and the appropriateness of the algorithm for use in an FPGA. Estimates of the module misalignment and offline corrections were also conducted, and a novel approach to station calibration.

Copyright Statement

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Contents

1	Introduction	21
1.1	The Large Hadron Collider	21
1.1.1	CERN Acceleration Complex	21
1.1.2	LHC Properties	23
1.1.3	High Luminosity LHC	25
1.2	The CMS Detector	26
1.2.1	Coordinate System	26
1.2.2	Tracker	28
1.2.3	Calorimeters	28
1.2.4	Muon Chambers	29
1.2.5	Triggers	29
1.3	Technologies Used at CMS	31
1.3.1	Field Programmable Gate Arrays	31
1.3.2	Silicon Detectors	33
2	CMS Phase-II Upgrade	37
2.1	p_T Modules	39
2.1.1	2S Module	40
2.1.2	PS Module	42
2.2	Back End Electronics	43
2.2.1	Architecture	44
2.2.2	ATCA Processing Cards	45
2.2.3	DTC	48
2.2.4	Track Finder	49

3	CMS Online Software	55
3.1	Introduction	55
3.1.1	Containerisation	55
3.1.2	Requirements	56
3.1.3	Architecture	57
3.2	HERD	58
3.2.1	Hardware Interface	59
3.2.2	Base Application	60
3.2.3	Plugins	60
3.3	Shep	61
3.4	Crook	64
3.5	Deployment	64
3.5.1	Balena	65
3.5.2	Kubernetes	65
4	DTC Development	69
4.1	Trigger Path	69
4.1.1	EMP xGBT interface	70
4.1.2	Framer	71
4.1.3	Stub Alignment	72
4.1.4	Link Interface	74
4.1.5	Stub Processing	77
4.2	DAQ Path	85
5	DTC Testing	87
5.1	Stub Processing	87
5.2	Front End Testing	90
5.2.1	Stub Histograms	90
5.2.2	Issues in the Data	93
5.3	GBT Full Module Tests	96
5.3.1	Initial Tests	96
5.3.2	HV Threshold Tests	96
5.3.3	Histogramming with a Radioactive Source	97
5.3.4	Latency Tests	103

6	The MUonE Experiment	107
6.1	Introduction	107
6.2	Extraction of a_μ^{HLO}	109
6.3	Experimental Setup	110
7	MUonE Test Beam	113
7.1	Experimental Setup	113
7.1.1	The M2 beam	113
7.1.2	MUonE Test Stand	114
7.2	DAQ Firmware for Serenity	119
7.2.1	Link Aggregator	120
7.2.2	Link Combiner	122
7.2.3	10 Gigabit Ethernet Link	124
7.2.4	Limitations and Scaling	124
7.3	Operations	126
7.3.1	Beam Conditions	126
7.3.2	Control and Monitoring	127
8	Beam Test Analysis	131
8.1	Data Taking Runs	131
8.2	Beam Characteristics	132
8.2.1	Intensity	132
8.2.2	Time Dependence	133
8.3	Event Characteristics	134
8.3.1	Event Size	134
8.4	Synchronisation and Alignment	135
8.5	Track Finding	138
8.5.1	Event Selection	138
8.5.2	Method	138
8.5.3	Intermediate Hit Deltas	141
8.5.4	Residuals	142
8.5.5	Multiplicity	145
8.5.6	Angular Distribution	146
8.6	Improvements to Track Finding Performance	147

8.6.1	Module Position Correction	147
8.6.2	Stub Bend	152
9	Conclusions	155
A	CIC User Data Mapping	165
B	Module Testing Plots	167
B.1	Day 1	167
B.2	Day 2	176
B.3	Day 3	178
C	Beam Test Error Propagation	197
C.1	Special case for equidistant planes	197
C.1.1	Interpolation	197
C.1.2	Extrapolation	199
C.2	Generalisation to arbitrary plane distances	199
C.3	Multiple Scattering	201

List of Tables

2.1	Boxcar multiplicities	43
2.2	L1T track object	43
5.1	Cable length latencies.	105
7.1	M2 beam dynamics	114
7.2	Stub data format	122
A.1	Framer stub data indices	165

List of Figures

1.1	CERN Acceleration Complex	22
1.2	LHC Dipole Cross-section	23
1.3	LHC Interaction Points	24
1.4	LHC Run Schedule	25
1.5	Transverse Slice of CMS	27
1.6	CMS Coordinate System	27
1.7	Cross-sections of physics phenomena	30
1.8	CMS Experimental Caverns	31
1.9	Firmware Design Process	33
2.1	Phase-II Tracker Layout	38
2.2	Level 1 Trigger Rates	39
2.3	Stub generation within p_t modules	40
2.4	CBC Electronics	41
2.5	CBC Threshold Voltage	41
2.6	Detector and processor sector overlap	44
2.7	Proposed rack layouts	46
2.8	Phase-II ATCA Prototypes	48
2.9	Track-finding performance	50
2.10	Hybrid Algorithm Design Flow	51
2.11	EMP Framework	52
2.12	IPBB dependencies file	54
3.1	Tacker Online Software Architecture	58
3.2	Infrastructure firmware and software for Phase-II Processing cards	59
3.3	HERD plugin structure	61

3.4	Shep homepage	63
3.5	Proposed Kubernetes architecture for the Track Trigger online software. . .	67
4.1	DTC block diagram.	70
4.2	lpGBT EMP datapath	70
4.3	Boxcar data format	72
4.4	DTC framer	73
4.5	Stub misalignment diagram	74
4.6	Link Interface firmware block diagram.	75
4.7	Stub interleaver cache occupancy	78
4.8	Tracker module correction geometry	79
4.9	Coordinate correction error plots	81
4.10	Correction algorithm error comparison	82
4.11	Coordinate correction resource usage	83
4.12	Systolic array	84
5.1	DTC Stub Processing buffer outputs	89
5.2	Stub histograms	91
5.3	Temporal data errors	94
5.4	GBT frame mean bit values	95
5.5	V_{cth} occupancy plot	97
5.6	CIC histogram comparison	99
5.7	Strip location histograms	99
5.8	Threshold voltage scan	100
5.9	Perspex rate difference	100
5.10	p_T cut sweep	101
5.11	Radioactive source	101
5.12	Point vs distributed comparison	102
5.13	Point vs distributed fit	102
5.14	Module and Scintillator Diagram	103
5.15	Oscilloscope connection to Serenity GPIO	104
5.16	DTC latency capture	105
5.17	Latency zoomed oscilloscope capture	106
5.18	Cable length calibration	106

6.1	Elastic scattering of a muon and electron	108
6.2	Running fine structure constant	110
6.3	MUonE Stations	111
6.4	Sensors within a MUonE station	112
7.1	Test stand diagram	115
7.2	Test stand	116
7.3	Power Supplies	117
7.4	Environmental control	118
7.5	2S-Serenity optical readout path	119
7.6	MProcessor Block Diagram	120
7.7	Link Aggregator	121
7.8	Link Combiner	123
7.9	10G Ethernet	124
7.10	Expanded MProcessor	125
7.11	Beam uptime	127
7.12	Run configuration example	128
7.13	Grafana dashboard	130
8.1	Run size	132
8.2	Intensity	133
8.3	Time dependence	134
8.4	Event Size	135
8.5	Synchronicity	137
8.6	ToF-corrected synchronicity	138
8.7	Tilt corrections	141
8.8	Track deltas	141
8.9	Monte Carlo track deltas	142
8.10	Track residuals	143
8.11	Monte Carlo track residuals	144
8.12	Fitted residuals	144
8.13	Multiplicity	145
8.14	High multiplicity event	146
8.15	Angular distribution	147

LIST OF FIGURES

8.16	Positional corrections	148
8.17	Stub delta heatmap	149
8.18	σ fit	150
8.19	μ fit	150
8.20	Stub delta comparison	151
8.21	Optimised angular distribution and multiplicity	151
8.22	Optimised track residuals	152
8.23	Stub bend	153
8.24	Effect of bend cut	154
9.1	Stub efficiency	156
9.2	Effect of sensor resolution on angular distribution	157
C.1	Base model for residual calculation	198
C.2	Extension to arbitrary plane distances	200
C.3	Multiple scattering	201

Chapter 1

Introduction

High energy physics covers a wide-ranging collection of phenomena, with the technology required to perform measurements on these similarly diverse. This chapter progressively focuses down from an overview of the Large Hadron Collider at CERN to the CMS detector, its subdetectors, in particular the tracker, and finally some technologies used for both generating and processing data in CMS. These technologies and the tracker detector of CMS will form the centrepiece of the work presented in this thesis.

1.1 The Large Hadron Collider

As of 2022, CERN is home to the most powerful particle collider in the world, the LHC[1]. It is a 26.7 km circumference circular collider, made up of two counter rotating beams that are focused to collide at four points around the ring, capable of colliding protons at a centre-of-mass energy of up to $\sqrt{s} = 13.6$ TeV. The primary motivation for the collider was “to elucidate the nature of electroweak symmetry breaking for which the Higgs mechanism is presumed to be responsible”[2].

1.1.1 CERN Acceleration Complex

Acceleration of the particles does not occur just in the LHC itself however, but is achieved in multiple stages, each increasing the energy of the protons above the previous (figure 1.1). First is a linear accelerator, LINAC4, which accelerates negative hydrogen ions from rest to 160 MeV, through four stages: up to 3 MeV in a radio-frequency quadrupole (RFQ), then 50 MeV in Drift Tube Linacs (DTLs), 100 MeV through Coupled-Cavity Drift Tube Linacs (CCDTLs) and finally up to the to output energy of 160 MeV by PI-Mode

Structures (PIMS).

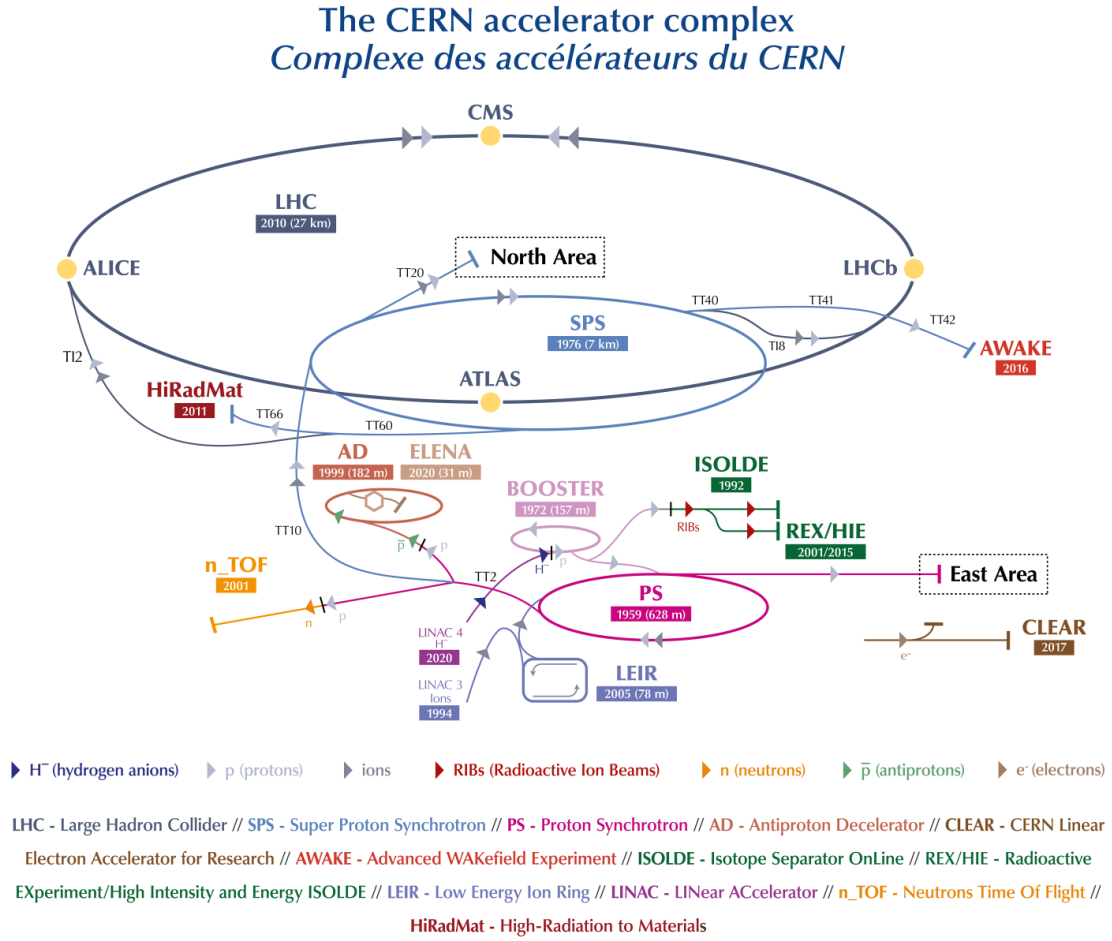


Figure 1.1: Diagram showing each of the accelerators present within the CERN Acceleration Complex. Source: [3]

The negative hydrogen ions from LINAC4 are then sent to the Proton Synchrotron Booster (PSB), an acceleration stage which strips the ions of their electrons to leave just the protons, and then boosts the energy of these protons to 2 GeV for injection into the Proton Synchrotron (PS). The PS is the first of the three major ring accelerators in the CERN complex at a circumference of 628 metres and boosts the energy of the protons up to 26 GeV.

The protons are then fed into the 7 km circumference Super Proton Synchrotron (SPS), which is capable of accelerating them up to energies of 450 GeV. Before becoming part of the acceleration complex for the LHC, the SPS was integral to a number of physics discoveries in its own right, most notably the experimental observation of the W and Z bosons[4–7]. Proton beams from the SPS are also used for fixed target experiments within

the CERN North Area, and the generation of secondary particle beams, such as the M2 muon beam.

1.1.2 LHC Properties

In order to achieve the collision energy of 14 TeV, the proton beam from the SPS is split into two, and sent in opposing directions around the ring in independent beam pipes (figure 1.2). Each beam is accelerated up to 7 TeV and steered around the ring through the use of 1232 superconducting dipole magnets. Protons are organised in the LHC into bunches spaced 25 ns apart, each typically containing $N_p = 10^{11}$ protons.

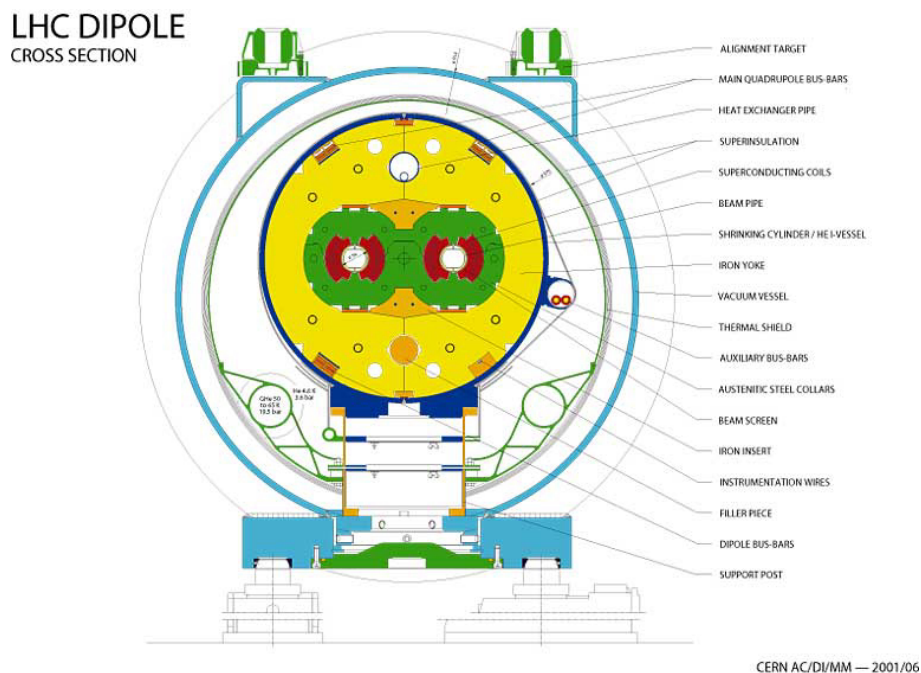


Figure 1.2: Diagram showing a cross-section of the a LHC dipole magnet, showing the two independent beam pipes. Source: [8]

Collisions of these bunches occur at four points around the ring, at each of which resides a detector. The two largest experiments, A Toroidal LHC Apparatus (ATLAS) and Compact Muon Solenoid (CMS), are located at Points 1 and 5 respectively, with Large Hadron Collider beauty (LHCb) located at Point 8 and A Large Ion Collider Experiment (ALICE) at Point 2. An diagram detailing the relative locations of each interaction point can be seen in figure 1.3

Aligned with the goals of the LHC, ATLAS and CMS were general purpose detectors designed to be optimised for recording events in which a Higgs boson was created. On July 4th 2012, the two experiments made a joint announcement detailing the discovery of

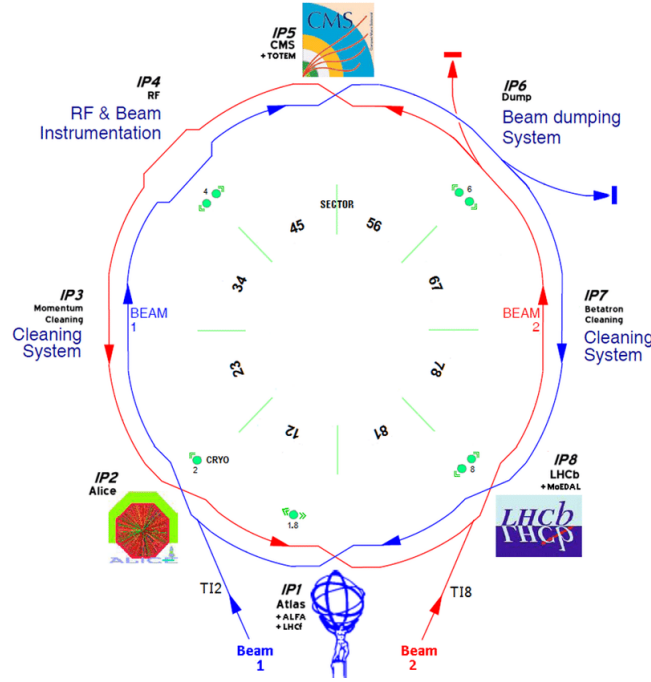


Figure 1.3: Diagram showing the positions of each interaction point on the LHC. The experiments located at a given interaction point are also shown. Source: [9]

the Higgs boson[10, 11].

The remits of LHCb and ALICE differ from those of CMS and ATLAS. The LHCb detector is designed to probe charge-parity violation in the beauty-sector, whilst ALICE makes use of the LHC's functionality to deliver ion collisions at $\sqrt{s} = 5.02$ TeV to understand quark-gluon plasmas and other phenomena of high density nuclear physics.

At the collision point for each of ATLAS and CMS, the proton beams are focused such that the instantaneous luminosity (L) is approximately $2 \times 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$. Given a typical collision frequency of 40 MHz, resulting a spacing between bunches of 25 ns, and the inelastic proton-proton cross-section $\sigma_{pp} = 1.1 \times 10^{-25} \text{ cm}^2$, the expected number of proton interactions per bunch crossing can be estimated:

$$n = L \times \sigma_{pp} \times 25 \times 10^{-9} \approx 50 \quad (1.1)$$

This quantity is given the name pileup and its value has a significant bearing on the design of the detectors used at the LHC

1.1.3 High Luminosity LHC

Data-taking at the LHC is split into multi-year “runs” the first of which began in 2010. Each run adds to the total integrated luminosity delivered by the LHC (see figure 1.4) and are separated by “long shutdowns” whereby repairs and upgrades can be made to both the accelerator and detectors.

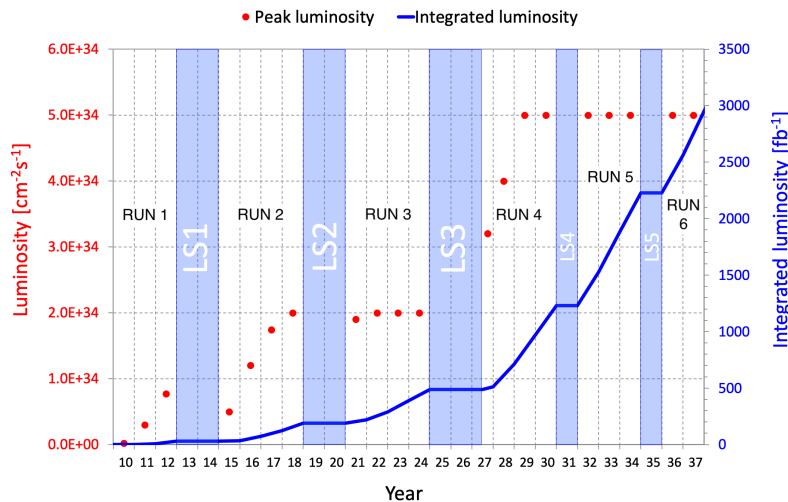


Figure 1.4: Diagram showing the instantaneous and integrated luminosity for each run of the LHC. Note that the proposed years for LS4 and LS5 have been subject to change since the generation of this diagram

Whilst the LHC operates at close to its maximal energy for a single proton, there is scope to increase the instantaneous luminosity delivered to the experiments by almost an order of magnitude, in a project termed High Luminosity LHC. This will allow for the production of datasets of an unprecedented size of up to 4000 fb^{-1} by the end of run 4.

The High Luminosity Large Hadron Collider (HL-LHC) is motivated by unanswered questions within the Standard Model. Since the discovery of the Higgs boson, research has shifted from a search for the particle to understanding its properties, particularly those with avenues for new physics. In order to do this, high statistics are necessary, with the aim to produce $\sim 10^8$ Higgs-containing events during Run 4.

The increase in integrated luminosity will be achieved primarily by three methods: increasing the number of protons per bunch from 10^{11} to 2×10^{11} ; upgrading the focusing magnets located around the interaction points; and luminosity levelling, a process whereby the beam is incrementally focused over the period of a spill, such that the luminosity remains constant whilst the number of protons in the bunch decays.

From equation 1.1 it follows from the increased luminosity will be the similar increase

in pileup. This will rise from approximately 25 up to 140-200. In order to resolve and process the increased number of interactions within each event, each of the experiments will be upgraded along with the LHC itself so that the physics performance of this new operating mode can be maximised. The details of the upgrade to CMS are the subject of chapter 2.

1.2 The CMS Detector

As described above, the CMS detector’s primary goal was the discovery of the Higgs boson, but it is capable of operating in a far more general purpose too. Since the discovery, its focus has shifted from a machine designed for discovery to that which can measure the properties of the Higgs boson, for example its coupling strength as a function of mass[12]. Its capability to perform high-precision measurements of numerous particle signatures, such as those produced by photons, muons and electrons, was integral to the Higgs discovery, which relied primarily on the two “golden” channels: $H \rightarrow \gamma\gamma$ and $H \rightarrow ZZ \rightarrow 4l$. It achieves this by leveraging the power of multiple sub detectors, organised in layers around the interaction point, as shown in figure 1.5. Moving outward from the beam pipe, there are first the trackers, with the pixel-based Inner Tracker (IT) closest and the strip-based Outer Tracker (OT) outside that. Beyond the trackers can be found the two calorimeters, first the electromagnetic and then hadronic. A superconducting solenoid magnet, capable of producing a field strength of 3.8 T encompasses the tracker and calorimeter. Outside of the solenoid are layers of muon chambers, specifically designed to detect and track muons. A highly detailed description of the detector can be found at [2].

1.2.1 Coordinate System

In all description of the CMS detector, a right-handed coordinate system is employed. As shown in figure 1.6a the origin is defined to be the approximate collision point at the centre of the detector, with the x -axis pointed inwards to the centre of the LHC ring. The y -axis points vertically upwards and the z -axis along the direction of the beam pipe toward the Jura mountains. Relative to the z -axis, the azimuthal and altitude angles are defined as ϕ and θ respectively.

A more commonly used measure of the forward angle within the detector is the pseu-

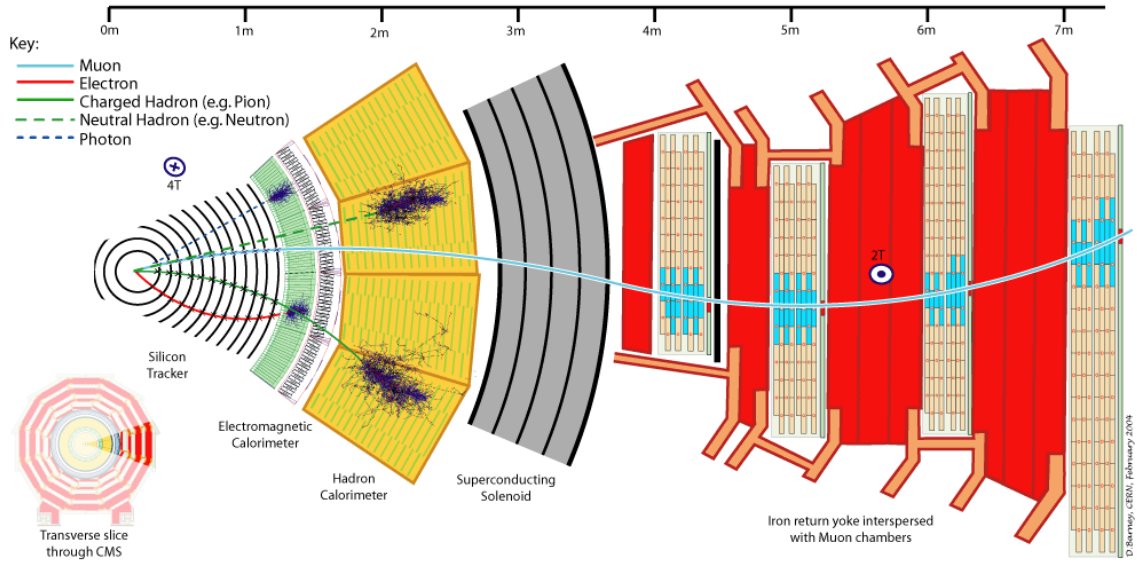


Figure 1.5: Diagram showing a transverse slice of the CMS detector. The beam pipe is located at the centre of the silicon tracker. Source: [13]

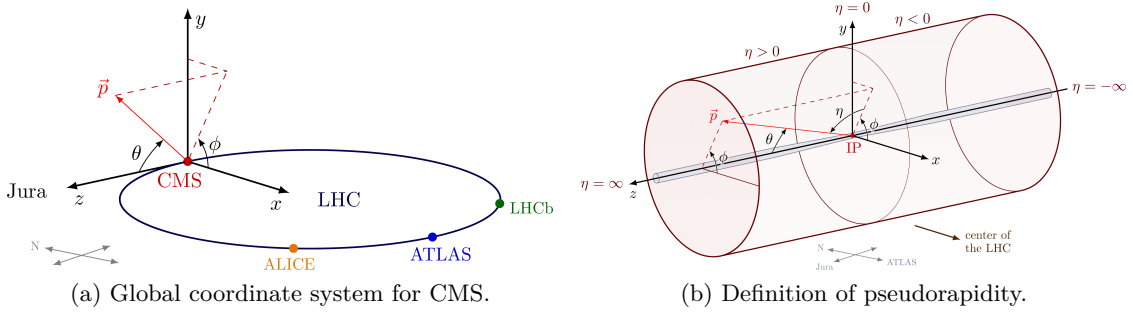


Figure 1.6: Diagrams showing the global coordinate system used in CMS, as well the definition of pseudorapidity. Source: [14]

dorapidity, which is defined as follows:

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right) \quad (1.2)$$

Whereby a point on the beamline has $\eta = \pm\infty$ and a point that sits on the xy -plane has $\eta = 0$. Quantities which are described as “transverse”, typically denoted by a subscript “T”, are sum of the contributions to that quantity projected onto the xy -plane. For example the transverse momentum, p_T , can be calculated as follows:

$$p_T = \sqrt{p_x^2 + p_y^2} \quad (1.3)$$

1.2.2 Tracker

In CMS, the role of the tracker is to record the path taken by a particle, whilst minimising the energy deposited within the tracker itself, so as to limit disturbances to the path and minimise the effect on the calorimetry measurement. The signal left by the particles in each layer of the tracker can then be used to reconstruct the path of the original particle and thus determine the original vertex. In CMS, the momentum of charged particles can be determined by observing the bend of a track within the magnetic field present in the detector. The transverse momentum of a charged particle is inversely proportional to the radius of curvature of its trajectory according to the following formula:

$$R = \frac{p_T}{q c B} \quad (1.4)$$

Closest to the beam-pipe, the IT is made up of 3 layers of pixel modules, with a size of $100\text{ }\mu\text{m}$ by $150\text{ }\mu\text{m}$. These provide accurate track information in three dimensions necessary to calculate the vertex of a track. The OT has 10 layers, coming out to a radius of 130 cm away from the beam, and is made up of strip sensors, allowing for accurate measurement in ϕ , at the cost of coarse position information in z . Both the IT and OT are made up of barrel layers located at low $|\eta|$ and discs which cover the high pseudorapidity regions up to $|\eta| < 2.5$.

1.2.3 Calorimeters

Whilst the tracker provides information on a particle's path and momentum, the other measurement required to determine the mass is its energy. In contrast to the tracker, in an ideal detector the calorimeter maximises the energy deposited within it. The calorimeter is also capable of detecting uncharged particles, such as photons and neutral hadrons, which cannot be detected within the tracker.

The first of the calorimeters, the Electromagnetic CALorimeter (ECAL), is made up of lead tungstate crystals, designed to precipitate electromagnetic cascades induced by an electron or photon. This scintillation is then collected at one end of the tapered crystal and read out by an amplifying photodiode. Beyond the ECAL is the Hadronic CALorimeter (HCAL), comprising shower-inducing brass plates and plastic scintillation plates to read out the signal.

1.2.4 Muon Chambers

The outermost section of the detector, housed within the steel return yoke of the superconducting solenoid, contains three different types of muon detector[15]. In the barrel region, Drift Chambers are employed, whilst in the end cap, Cathode Strip Chambers (CSC) are used. For all $|\eta| < 2.1$, Resistive Plate Chambers (RPC) are also installed to give high precision timing information for the muons. During Long Shutdown 2, Gas Electron Multiplier (GEM) detectors were installed into the endcap region to add further accuracy to measurements of forward muons.

1.2.5 Triggers

Limitations in processing power, storage capacity and bandwidth out of the detector mean that it is not possible to record every event that occurs within CMS. As such, a decision must be made as to which events are stored for further analysis and which are discarded. As shown in figure 1.7, the cross-sections for physics events, for example $h \rightarrow \gamma\gamma$, that are of interest are many orders of magnitude below the inelastic proton-proton cross-section, meaning that almost all events that happen in the detector do not contain interesting physics. The aim of the trigger is to discard a sufficient fraction of these uninteresting events such that the rate of acceptance can be handled by the downstream processing.

In CMS, the trigger has two tiers: the first is the Level-1 Trigger (L1T), which takes a subset of the full event data and performs basic reconstruction; the second, the High Level Trigger (HLT), takes only the events deemed worthy by the L1T and reconstructs them in higher precision using the full event dataset to make a more informed choice. Both of these triggers are considered to be “online”, meaning that the data is kept in volatile storage whilst being processed. Only data which passes both levels of the trigger is then written to tape. For the current detector, the L1T reduces the accepted event rate down from 40 MHz to 100 kHz, which is then further reduced to 1 kHz in the HLT. All accepted events are first stored at CERN, Tier-0, before being copied to one or more Tier-1 sites around the world[17]. Further reconstruction and filtering required for specific analyses are then performed on the Worldwide LHC Computing Grid, making use of distributed computing resources around the world.

As shown in figure 1.8, the counting room located in the cavern next to the detector houses the processing resources for the L1T. In a special case of an online trigger, for the L1T data is stored on the detector itself whilst a decision is made, meaning that

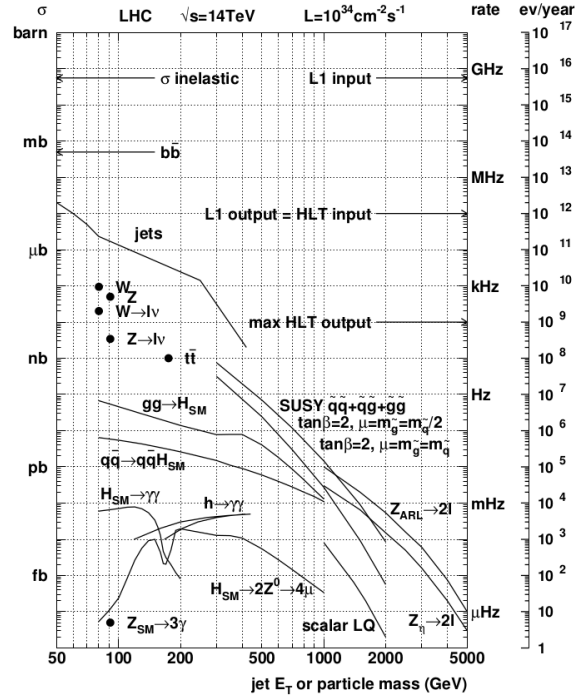


Figure 1.7: Plot of relative cross sections of Standard Model phenomena at differing energies. The σ inelastic cross-section determines the number of events produced in the LHC beam. Source: [16]

only a very small window in time, for CMS $3.2\mu\text{s}$, is allowed before the Level 1 Accept (L1A) signal must be propagated back to the detector in order for the full event to be read out or discarded. The 100m of fibre that connects the detector modules to the L1T adds unavoidable latency which must be accounted for in the time budget for the L1T. Within the current detector, there is neither the bandwidth nor the processing power to include information from the tracker within the L1T, therefore only calorimetry and muon chamber information is used to make decisions. Each of the sub-detectors are first processed independently to reconstruct objects such as electrons and photons, which are then passed to a global trigger block which makes the decision of whether to accept the event. In order to meet the latency target and maximise throughput, the L1T homogeneously uses Field-Programmable Gate Array (FPGA) technology, with serial optical links used to communicate between processors.

The computing farm which houses the HLT is located at surface level at Point 5 and comprises commercial servers with CPUs to perform the processing.

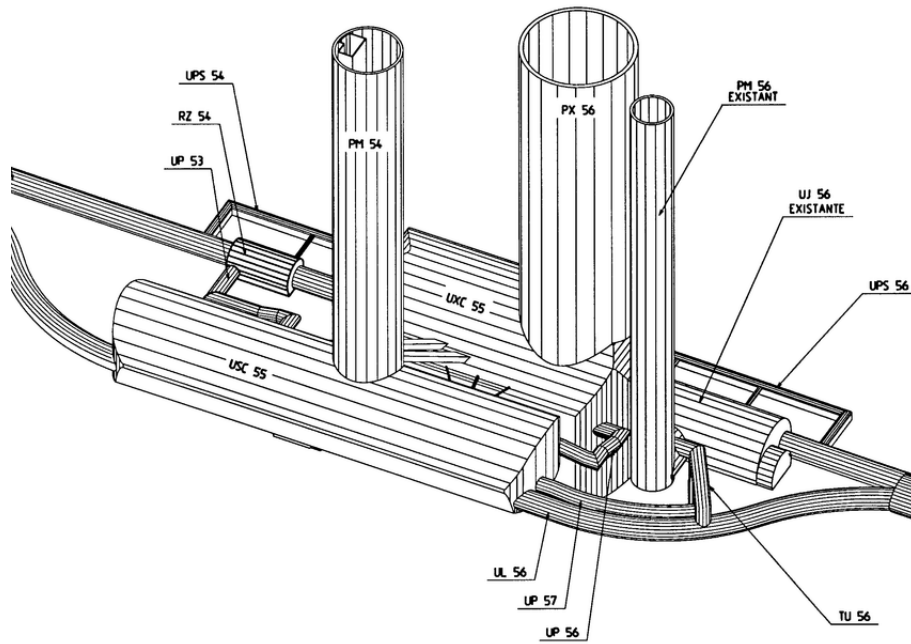


Figure 1.8: 3-D projection of the CMS experimental caverns at Point 5. The detector itself is housed in UXC 55, with services and readout electronics located in USC 55. Source: [18]

1.3 Technologies Used at CMS

The achievements in High Energy Physics would not have been possible without significant contributions and advancements in other areas of scientific research, in particular solid state physics and computing. This section provides information on two examples of technologies that are used extensively at CMS and the wider High Energy Physics field: the FPGA and silicon particle sensors.

1.3.1 Field Programmable Gate Arrays

Undertaking all of the processing for both the current and future L1Ts, the technology and architecture of FPGAs is significantly different from a CPU. It shares instead many similarities with an Application Specific Integrated Circuit (ASIC), however has the benefit of being entirely reprogrammable, at the cost of a reduction in throughput and an increase in power consumption. Unlike a CPU, which generally has a sequential processing architecture, an FPGA is capable of performing highly parallel tasks, as well as so-called “pipelined” processing. In this paradigm, multiple blocks of logic can each be working on individual bits of data in parallel, then pass the result to the next block and receive new data from the block upstream. Such an architecture allows for the incredibly high

throughputs required by the L1T to be achieved. Built into the architecture of the FPGA are also a number of specialised resources, allowing for better optimisation of general-purpose resources and to improve timing convergence. Some examples of these resources are detailed below:

- **BRAM:** sharing the same role within a system as memory used in commercial PCs, these blocks can each store a small amount of data (typically 36kb) and can be accessed through two independent ports. In comparison to a CPU, whereby RAM is generally located on a separate die, Block Random Access Memory (BRAM)s are distributed within the FPGA, allowing for access to data to be achieved in one or two clock cycles, compared to typically 15 or more required for external memory access.
- **DSP:** A Digital Signal Processor (DSP) is a specialised piece of logic that is capable of complex signal processing tasks that would otherwise use significant resources on the chip. These include Fast Fourier Transforms as well as the multiplication of two variables.
- **Transceivers:** To achieve the high data throughput for which an FPGA is so effective, the I/O that can be achieved must be similarly high. As such custom transceivers capable of up to 58 Gbps on a single serial link are provided within the FPGA package itself, with large FPGAs having up to 128 of these transceivers.

“Programs” for an FPGA are typically made through the use of a Hardware Description Language (HDL). The language used most commonly for FPGA designs at CMS is VHSIC Hardware Description Language (VHDL). As shown in figure 1.9, similar to how a C++ program must be first compiled before it can be run, firmware designs must go through a number of steps before they can be loaded onto an FPGA.

The first step is Synthesis, a process whereby the HDL design is converted into a “netlist”, whereby operators in the HDL become logic gates, and variables to wires. Specialised logic units are also inferred from the design at this stage. After synthesis is complete, an initial estimate can be made as to the resources required on the FPGA.

After synthesis comes implementation, whereby the synthesised netlist is organised such that it fits physically on the target FPGA. This is an iterative process whereby the logic blocks are placed and then connected up to each other. As well as the spatial constraint on the placement of the logic block, timing must also be considered. A signal

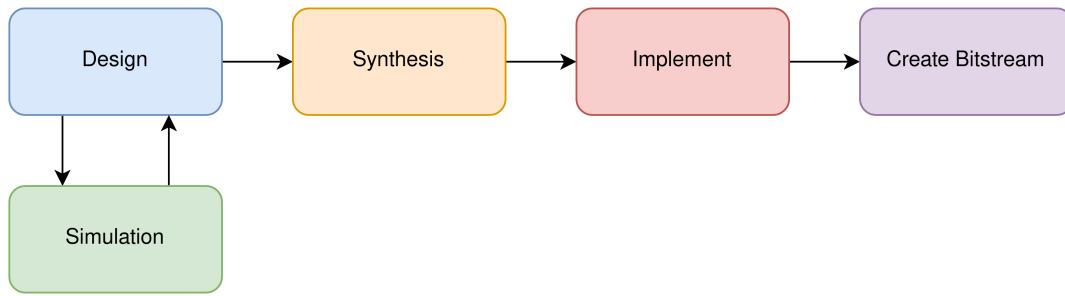


Figure 1.9: Diagram showing the overall design process when developing firmware for an FPGA.

takes a finite time to travel across the FPGA silicon, thus if the path is too long, the signal is allocated to the wrong clock edge of a process, leading to unintended behaviour within the chip. As such, the design must also be placed in such a manner that the distances that each signal must travel are minimised whilst not conflicting with the path of any other signal and meet the threshold for the given clock frequency that the FPGA runs at. A higher clock frequency requires shorter paths.

A successfully implemented design can then finally be converted to a format suitable for programming the FPGA, named a bitstream. This bitstream defines how each gate, interconnect and logic block block of the FPGA must behave to reflect the created design.

Synthesis and Implementation are both computationally expensive and time consuming actions to perform each time a new design is created. In order to speed up the development process, simulation software of the FPGA exists. This software can check for errors in the design by modelling how each signal within the design evolves over time for a given input. This gives a detailed and invaluable insight into how the design operates, which is far more difficult to achieve on the FPGA itself. As such, designs are typically iteratively developed and optimised through the use of simulation before any builds are attempted.

1.3.2 Silicon Detectors

Including the tracker of CMS, the use of semiconductor detectors is widespread in High Energy Physics, specifically those which use silicon. The band-gap and thus ionisation energy required to create an electron-hole pair in a semiconductor is typically on the order of 1-2 eV, making them ideal for use as a detector of ionising radiation: small enough to minimise the energy loss for the incident particle whilst similarly minimising the current from thermal excitations. The electron-hole pairs created in the silicon and their subse-

quent movement under an applied bias is the signal to be measured in the accompanying electronics.

At room temperature there is, however, sufficient thermal energy to already have many electrons-hole pairs produced, thus any further pairs generated by an incident particle would be swamped. For example, a square centimetre of silicon 200 μm thick (d) with a carrier density of $1.45 \times 10^{10} \text{ cm}^{-3}$ at room temperature will generate approximately 3×10^8 electron-hole pairs. A Minimally Ionising Particle (MIP), as defined by the Bethe-Bloch formula[19], will lose a constant amount of energy per unit distance travelled (dE/dx). Thus for a material with a given ionisation energy, I_0 , the number of electron-hole pairs generated can be calculated:

$$\frac{dE/dx \cdot d}{I_0} = \frac{3.87 \times 10^6 \text{ eV cm}^{-1} \cdot 0.02 \text{ cm}}{3.62 \text{ eV}} = 2.1 \times 10^4 \quad (1.5)$$

This value is over 4 orders of magnitude lower than that of the thermal pairs and thus would be impossible to measure. In order to overcome this problem, it is necessary to remove these thermally-produced electrons from the material before beginning operation as a detector. This can be achieved by the construction of a pn-junction from the silicon.

A pn-junction is created by doping the silicon with both “donor” atoms for n-type and “acceptor” atoms for p-type in adjacent regions of the substrate. These atoms replace silicon atoms within the crystal lattice and thus change the electrical properties of the region. N-type dopants, such as phosphorous, contain an extra valence electron compared to silicon and thus increase the number of electrons within the conduction band of the system. By contrast, a p-type dopant, such as boron, has one fewer valence electron compared to silicon, and thus increases the number of empty states, or holes, within the valence band of the system.

By combining n and p-type silicon together, diffusion of charge carriers will occur at the interface between the two materials until there is an equilibrium. Once this has happened, the area around the interface will no longer contain the same concentration of electrons and holes compared to the rest of the material and is therefore defined as being “depleted”.

The width of the intrinsic depletion region is dependent on the concentration of the doping, but can be further controlled by the application of an electric field, or bias, across the junction. In one direction, this bias will act to shrink the depletion region and thus

allow for current to flow freely across the junction, so called “forward bias”. In the opposite direction, or “reverse bias”, the electric field will increase the width of the depletion region further, until at its most extreme, there are no longer any available charge carriers within the silicon. It is in this mode of operation that the system becomes highly effective as a particle detector no more thermally-generated electron-hole pairs will exist in the depletion region, allowing for a signal created by an incident particle to be significant.

The application of a bias in the silicon does not change the likelihood of electron-hole pairs to be produced by an ionising particle. In the case of a particle being incident, the pairs are transported off the silicon through both diffusion and the static electric field. The movement of these electrons, and thus current, can be measured by electronics connected to the end of the silicon sensor. The pulse of current is converted to an equivalent voltage, which, provided is high enough to meet a threshold within the electronics, generates a digital “1”, indicating a hit within that part of the sensor.

Once the particle has passed through the silicon and thus no more electron-hole pairs are being produced, the current measured in the sensor will quickly fall again. The rise and fall time of the current signal is proportional to the carrier mobility of the semiconductor used within the sensor. Silicon has a sufficiently high electron mobility, thus allowing for resolution of particles from individual collisions to be possible at the 40 MHz collision frequency of the LHC.

Chapter 2

CMS Phase-II Upgrade

As described in chapter 1, one of the most significant changes that will arise through the HL-LHC upgrade is the increase in pileup from 25 to 140-200 collisions per bunch crossing. For the detectors on the LHC, this will both create more data, but the increased luminosity will also induce far more radiation damage over the duration of Run 4. The long shutdown period before the start of HL-LHC will be used by CMS in order to replace and upgrade parts of the detector such that it can continue to perform its physics goals. This chapter will present how CMS will be upgraded, with a particular focus put on the upgrade to the tracking system, both in the modules that will be installed on the detector and the off-detector data processing systems.

The current endcap ECAL and HCAL will be merged into a single calorimeter, named the High-Granularity CALorimeter (HGCAL), as a result of radiation damage in the current crystals and their poor performance. This will also enable an upgrade to the system, improving resolution in both energy and space. The readout electronics for the Muon chambers will also be upgraded before HL-LHC.

The tracker will be replaced in its entirety, necessitated by the radiation damage incurred over runs 1 to 3. It will be replaced in two tracking regions the IT and OT, with both featuring improved spatial resolution and radiation hardness over the current tracker. The improved resolution will be required in order for the detector to perform in the high pileup conditions of HL-LHC, whereby fine-grained tracking will be necessary to separate the multiple concurrent collisions that will occur in each Bunch Crossing (Bx).

The new tracker, as shown in figure 2.1, will contain a total of ten layers in the barrel region. Four layers will be IT pixel modules, with the remaining six made up of OT modules. The IT will have 12 endcap disks, ensuring coverage up to $|\eta| < 4.0$, and the

OT will have five. The proportion of layers chosen for each module type was optimised to maximise performance, whilst ensuring that space, bandwidth limitations and cost were not exceeded.

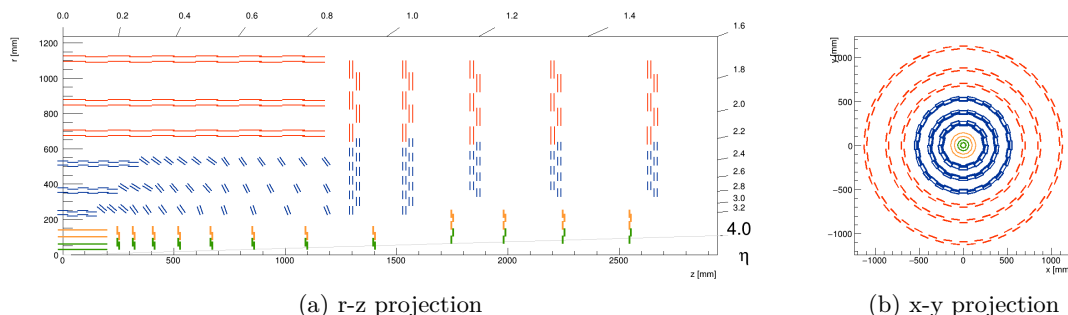


Figure 2.1: Diagrams showing the tracker layout for Phase-II. In the IT the green lines correspond to pixel modules using two readout chips and the yellow lines to pixel modules with four readout chips. For the OT the blue and red lines represent the PS and 2S modules respectively. Generated using tklayout[20].

As well as being upgraded, the OT will play an integral role in the trigger performance of CMS in HL-LHC. Given the high pileup conditions, using the same detector information and trigger menu as in Phase-I would lead to a trigger rate of 4 MHz, far beyond the design input of 750 kHz for the Phase-II HLT (see figure 2.2). In particular, the current selection algorithms, without information from the tracker, perform poorly in high pileup conditions, whereby the multiple interaction become difficult to distinguish. So that CMS can continue to perform a wide-ranging physics program, information from the OT will be used for the first time within at the L1T. The high spatial granularity of this extra information will allow for the trigger algorithms to maintain an acceptable rate whilst operating with higher pileup. Made possible through the novel, dual-layered, sensor design for the OT, the so-called “ p_T module”, only hits within the tracker compatible with particles with a momentum of $> 2 \text{ GeV}/c$ are sent to the L1T. This will reduce the required bandwidth sufficiently for the inclusion of the data to be possible.

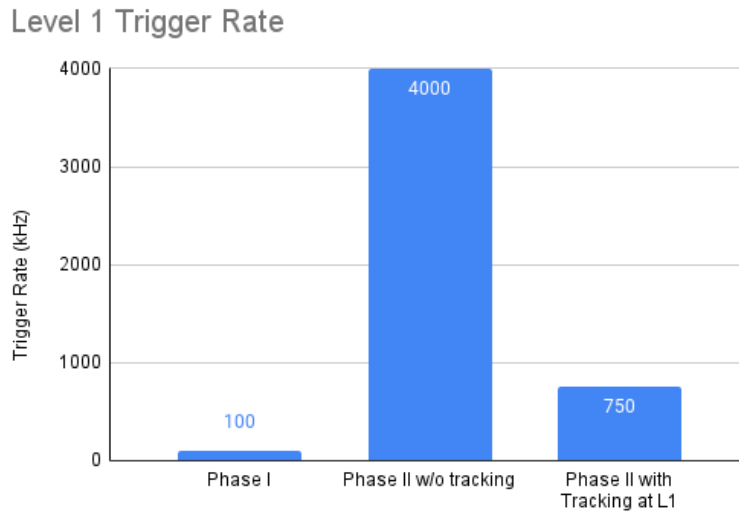


Figure 2.2: Plot showing the relative trigger rates for Phase-I, Phase-II without tracking information and Phase-II with tracking information included.

2.1 p_T Modules

For Phase-II, the OT modules have been completely redesigned, with two different types set to be installed, the so-called Two Strip (2S) and Pixel Strip (PS) modules. The PS modules will be located closest to the beam pipe within the OT, and contain two layers of silicon: the layer closest to the beam made of pixels, and the outer layer, strips. The 2S modules, which will form the outer layers of the OT, will be made up of two layers of strip sensors. Similar to the split between IT and OT in the overall tracker, PS modules are only used for the innermost layers whereby the pixel layer can be leveraged to perform accurate vertexing of particles. In the outer layers, further from the beam spot, this resolution is not necessary, thus strip modules are used to limit the bandwidth required to readout the whole tracker.

Both of these module types have the same design philosophy, whereby as a result of the high magnetic field within CMS, the bend of a charged particle passing through the two layers of the module can be estimated by comparing the relative position of the hit in both sensors, as shown in figure 2.3. A particle with high p_T will traverse the two layers in almost a straight line, leading to a small offset between layers. Conversely, a low p_T particle will be bent significantly by the 3.8 T magnetic field within CMS and thus pass through the outer layer of the module at a larger offset. This discriminating power for the modules to determine p_T allows for only hits which are compatible with tracks $> 2 \text{ GeV}/c$

to be sent to the L1T.

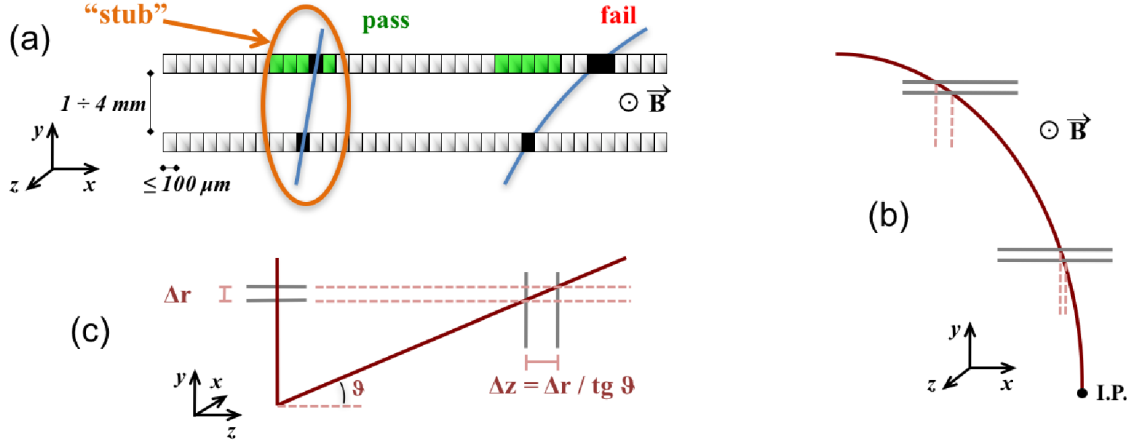


Figure 2.3: Diagram showing how stubs are created from correlated hits within a p_t module. (a) a second hit that lies within a defined window, shown in green, is combined with the primary hit to form a stub, whilst if the bend is too high, the second hit lies outside of the window. (b) a particle with constant p_t will create a stub with a larger bend in a module with a large radius. Each layer of the tracker must therefore be calibrated to account for this. (c) Modules with a larger spacing are used for the endcap disks, so that the same discriminating power in Δr can be achieved for particles with high pseudorapidity. Source: [21]

Also situated on the modules themselves are the readout electronics and power delivery systems required to both convert the analogue signals generated in the silicon sensors to a digital value, as well as correlate hits between the two layers to generate the combined signal, a so-called “stub”, which is then transmitted optically to the L1T.

2.1.1 2S Module

Making up the outer three layers of the OT, the 2S module comprises two silicon strip sensors, separated by either 1.8 mm or 1.4 mm. Each sensor is split into two columns, with each containing 1016 individual strips. Strips from both the top and bottom sensor are read out by the same ASIC, namely the CMS Binary Chip (CBC), which allows for correlations between hits in both sensors to be made. Each sensor column is read out by 8 CBCs, corresponding to 127 strips per ASIC, as well as some communication between CBCs to account for pairs of hits which cross boundaries between CBCs.

Each strip is 5 cm long with a pitch of $90 \mu\text{m}$, giving a total sensor area of $10 \times 10 \text{ cm}^2$ made of n-doped silicon on a p-doped bulk. This doping method was chosen after investigations of both n-in-p and p-in-n showed that n-in-p had a far more Gaussian noise distribution after irradiation than the other method[21]. N-in-p sensors lead to a

predominantly electron signal when the silicon is depleted.

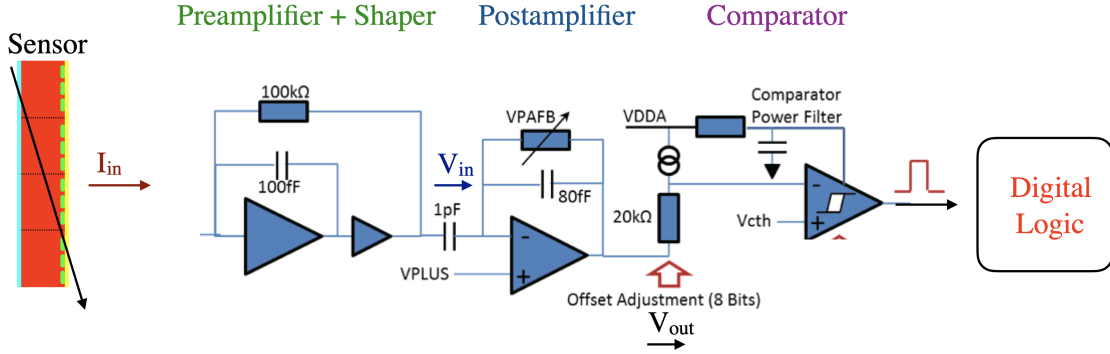


Figure 2.4: Diagram of the analog front-end electronics within the CBC.

The digitisation, correlation logic and calculation of stub properties is performed by the CBC, and all following manipulation of the data is within the digital domain. The analog circuitry of the CBC comprises three stages: Preamplifier + Shaper, Postamplifier and Comparator[22] (figure 2.4). The first two stages integrate and amplify the incoming signal, whilst the final converts the signal from the analog domain to the digital. In this comparator stage, the analog signal is compared to a reference threshold voltage, V_{cth} . If the signal crosses this threshold, a digital “1” is outputted, until the signal crosses back. The silicon sensor works in a so called “electron” mode, meaning that the excess of electrons is the signal which is measured in the readout electronics. In this case, the signal spike which is compared against a reference voltage is negative, thus a high threshold in the CBC corresponds to a more negative V_{cth} (figure 2.5).

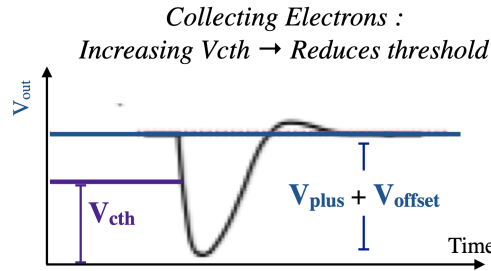


Figure 2.5: Plot of a dummy input signal to the comparator stage of the CBC, with each voltage level labelled.

Stubs generated by all eight CBCs are collated by the Concentrator Integrated Circuit (CIC) ASIC, one located in each half of the module, which then constructs the data packet to be sent to the off-detector electronics, including both stubs to be used within the L1T

and L1A data to go to the HLT. For the trigger path, stubs are collected over eight consecutive bunch crossings and then sorted in ascending order by their bend, meaning that those with the lowest bend, and by extension highest transverse momentum are sent first in the packet. In the case of more stubs being produced than can fit in the packet, or in tracker terminology, a “boxcar”, those with the highest bend are discarded, minimising the effect on the L1T. In the case of fewer stubs than the maximum multiplicity allowed within the packet are produced, a zero padding scheme is used to fill the space within the packet.

The two CICs, each serving half the sensor area, are both connected to a single transceiver module, the low-power GigaBit Transceiver (lpGBT), to serialise the data in preparation for transmission off-detector. The lpGBT also acts as an Inter-Integrated Circuit protocol (I2C) master for the module.

The final link in the chain before the data is sent over the optical link is the Versatile Link Plus (VL+) module, a radiation-hardened, CERN-built optomodule which converts the electrical signals from the lpGBT to an optical signal designed to be sent over the fibre.

2.1.2 PS Module

The design of the PS module is similar to that of the 2S, but rather than two layers of strip sensors, the lower is a macro-pixel sensor, containing pixels $1467\mu\text{m}$ long with a pitch of $100\mu\text{m}$. The overall size of the module is half that of the 2S, with the strip layer containing only a single bank of strips, with the same pitch as the macro-pixels. Hits in the upper strip sensor are read out independently to the macro-pixels, by the Strip Sensor Array (SSA) ASIC. The digitised and sparsified data, is then sent to the corresponding Macro-Pixel Array (MPA) ASIC, which reads both the hits from the macro-pixel to which it is bump bonded, and then correlates this with the data from the SSA. The data format and transfer protocol for the resulting stub is similar to that of the output of the CBC for the 2S module, meaning that the same CIC chip can be used for both the 2S and PS, albeit with different configurations.

Given the higher expected occupancy of the PS modules compared to the 2S, the bandwidth of the lpGBT link off the module is increased. The forward error correction mode is reduced from FEC12 to FEC5 to allow for more data bits to be included in each frame at the cost of increased susceptibility to irrecoverable bit errors. A 10 Gbps

operating mode will also be used for the modules closest to the beam pipe. As shown in table 2.1, this allows for up to 35 stubs per boxcar to be sent to the L1T.

FE Config	CBC (2S)				MPA (PS)							
Output Format	With Bend		Without Bend		With Bend				Without Bend			
Stub Width (bits)	18	18	14	14	21	21	21	21	18	18	18	18
Output Frequency (MHz)	320	320	320	320	320	320	640	640	320	320	640	640
# e-links	5	6	5	6	5	6	5	6	5	6	5	6
Max Stub Multiplicity	16	19	20	25	13	16	29	35	16	19	34	40

Table 2.1: Table showing the the boxcar properties for different Front End (FE) configurations. An output frequency of 320 MHz corresponds to the 5 Gbps lpGBT operating mode, with 640 MHz corresponding to 10 Gbps.

2.2 Back End Electronics

With tracking information being included in the L1T for the first time for Phase-II, the use of this information within the trigger is similarly novel. Within the L1T for Phase-I, the calorimeters and muon systems generate so-called “primitives”, which are then collected together and a global decision is then made. Similarly for Phase-II, the tracker must also generate a primitive. This primitive takes the shape of a helical track, which can be parameterised using 5 variables. Each of the 5 parameters, as well as the possible range of values and precision can be found in table 2.2.

Parameter	# bits	Granularity	Range
d_0	13	0.000391 cm	± 16 cm
z_0	12	0.0100 cm	± 20.5 cm
ϕ_0	12	0.000383 rad	± 0.785 rad
$1/R$	15	$3.66 \times 10^{-7} \text{ cm}^{-1}$	$\pm 0.006 \text{ cm}^{-1}$
$\tan(\lambda)$	16	0.000244	± 8

Table 2.2: Table showing the number of bits, granularity and possible range of each variable within the L1T track object.

These tracks are formed from multiple stubs across the layers of the tracker which are consistent with originating from a single particle. In a low pileup environment, this would be a relatively simple task, however in the expected conditions arising from HL-LHC of up to 200 collisions per bunch crossing, it becomes far more difficult. This is because whilst the number of tracks scales linearly with pileup, the number of possible combinations of hits across the layers of the tracker scales exponentially.

2.2.1 Architecture

The tracker Back End (BE) data processing system comprises two stages: the Data, Trigger and Control (DTC) and Track Finder Processor (TFP). The DTC serves as the interface to the FE modules and decodes and organises the stub data into an optimal format for the TFP. The TFP then takes the stub data and fits it into tracks, which are then sent downstream.

The tracker modules on the detector are split virtually into nine individual sectors, or “nonants”, each of which is processed independently. Each nonant is served by 24 DTCs giving a total of 216 DTCs for the entire OT. Each DTC card is capable of handling 72 links to FE modules. As shown in figure 2.6, track-finding is similarly split into nonants, however each of these are rotated by 20° such that they span the boundaries in the DTC sectors. This ensures that stubs created by a track which crossed the border between two DTC sectors is correctly reconstructed. Hits which are compatible with both track-finding nonants are duplicated and a copy used for reconstruction in each TFP.

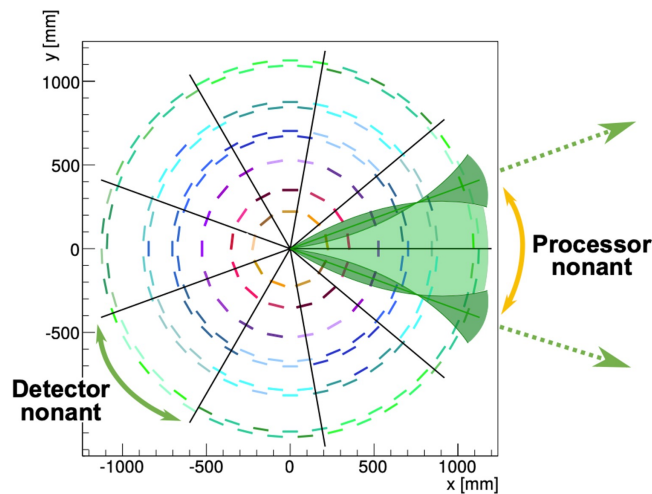


Figure 2.6: Diagram showing the difference between the detector and processor nonants. The so-called “hourglass” region between two processor nonants denotes the area where stubs are duplicated between regions in the DTC.

Based on its successful implementation in the current L1T, the TFPs are organised in a “Time-Multiplexed” architecture. In such a paradigm, all data from a single event is collected, stored and processed together, with multiple copies of the same processor each running on separate events in parallel, each offset by one Bx. This allows the time given for reconstruction to be increased from a single Bx to N Bx, where N is the time-multiplexing period. In the case of the TFP, the time-multiplexing period is 18, thus each processor

has 450 ns in which to perform the track finding.

2.2.2 ATCA Processing Cards

Another aspect of the Phase-II upgrade is the move to a new form factor for the processing cards that will be housed in the counting room at Point 5. In Phase-I, μ TCA[23] cards, such as the MP7[24] were deployed, whereas in Phase-II, the Advanced Telecommunications Computing Architecture (ATCA)[25] standard has been chosen. This new form factor is significantly larger than μ TCA at 280×322 mm and is rated by the standard to deliver a maximum of 400 W to the electronics on the card. This allows for larger and more powerful FPGAs to be used.

Up to 14 ATCA cards can be fitted into a single crate, with up to two crates installed per rack in the counting room such as in figure 2.7. These crates use a vertical mounting and airflow system, whereby cool air is brought in at the base of the crate, with hot air exhausted from the top. In order to limit the leakage of heat from each rack into the cavern, heat exchangers will be installed above each crate, allowing for heat to be transported away via water pipes. The crates contain within them a backplane to which all cards in the crate connect to, delivering power to the cards as well as other services such as networking for management. The primary I/O for each card uses optical fibres that pass out of the front of the crate to be routed to their destination.

Conceptually, the two processing cards selected for use in the tracker BE have many similarities; both were designed to be general purpose processing cards, intended for use both in other systems within CMS as well as other experiments. They can be fitted with up to two Xilinx Ultrascale+[26] FPGAs, with sufficient optical transceivers to maximise their throughput. In contrast to Phase-I processing cards, Phase-II cards have extra computation resources located on the cards for independent control and management. This takes the form of an System-on-Module (SoM), typically with either x86 or ARM architecture, that is able to run a standard GNU/Linux operating system. These SoMs provide the opportunity to develop online software that will run on the cards themselves that can be powerful, without the large overhead of building for a custom embedded system.

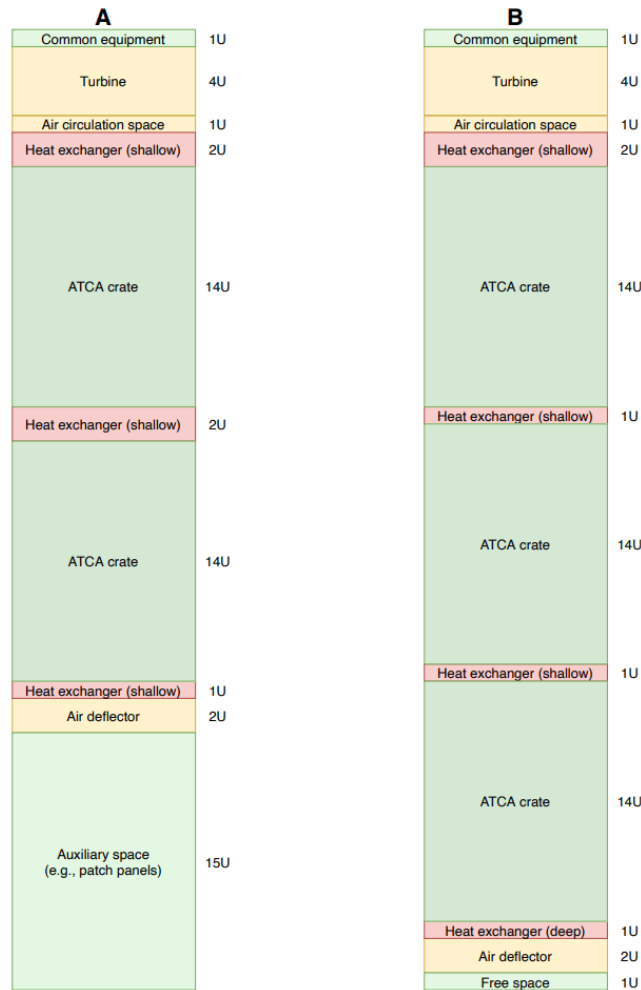


Figure 2.7: Diagram showing the proposed rack layouts to be used for the Phase-II BE electronics. Proposal A is the baseline with two 14U crates per rack; whilst proposal B increases the crate density by allowing a third crate per rack, but is limited to the same power budget as the proposal A. This means that proposal B may only be used for crates with low power requirements. Source [27]

2.2.2.1 Serenity

Initially designed at Imperial, the Serenity[28] card (figure 2.8a) is a highly versatile ATCA card. It achieves this versatility through the use of an interposer, onto which the FPGAs are placed. The FPGAs themselves are bonded onto small Printed Circuit Board (PCB)s, named daughtercards, in a similar fashion to commercial CPUs, which are then mounted atop the interposer. This decoupling of the FPGA away from the main PCB provides flexibility in the choice of FPGA for a given card, even after the main PCB has been assembled, as well allowing for either part to be replaced in the event of problem occurring.

Optical communication on the board uses a similarly modular approach, with SAMTEC Firefly transceivers[29], being the current choice. These modules are capable of provid-

ing up to twelve 25 Gbps channels each, giving a potential I/O throughput of a single daughtercard in the order of 2.8 Tbps Rx + 2.8 Tbps Tx.

Serenity cards are capable of accommodating up to two FPGAs, provided the combined power requirement is less than that for a single ATCA slot. For the DTC, for which the Serenity is the card of choice, only a single Xilinx VU13P[30] will be installed. This will be connected to 72 optical links originating in the detector, which will use pre-qualified SAMTEC modules that are rated for higher Tx power and Rx sensitivity. A third, smaller, FPGA is also included on the Serenity card, which is used to program and provide clocks to the two large FPGAs. For this a Xilinx Artix-7[31] class FPGA is used.

For Serenity, a number of SoMs have been proposed to control the board. Early prototypes have been assembled with the Com-Express[32] family of SoMs, which use x86-class CPUs and include up to 8 GB of RAM. More recent boards have since been fitted with the Xilinx Kria SoM, containing a hybrid ARM Central Processing Unit (CPU) with a smaller FPGA connected to it. Whilst evaluation is still ongoing into this SoM, it has the possibility to simplify the design of the Serenity card by removing the Artix FPGA, as its role would be absorbed into that of the SoM. The SoM has an independent gigabit ethernet connection to the wider network through the crate backplane. Images to be loaded onto the FPGAs are first downloaded over this interface onto the SoM, before the FPGAs are programmed over a PCIe link between the SoM and Artix, which then translates the signal to JTAG to then send to the large FPGA.

2.2.2.2 Apollo

For the TFP, a board designed by a conglomeration of American Institutes named Apollo (figure 2.8b) has been proposed[33]. It shares a number of similarities with the Serenity design, with a dual FPGA design and the use of SAMTEC Firefly optics. Instead of the FPGAs being connected to the board through an interposer, different versions of the board are achieved by splitting the processing components (FPGA and optics) from the service components (power, timing, management interface). Each collection of components is assembled onto an individual PCB, named the Command and Service modules respectively, which are then connected together to form a complete ATCA card. This approach has both the advantage and disadvantage of having the FPGA soldered directly onto the PCB. By being soldered and without the interposer, there is one fewer discontinuity in the the high speed lines which carry data to and from the FPGA and the optical modules, which

leads to a cleaner signal being transmitted with fewer errors. The cost of this however, is that the FPGA cannot be easily replaced should it, or another component on the whole PCB fail.

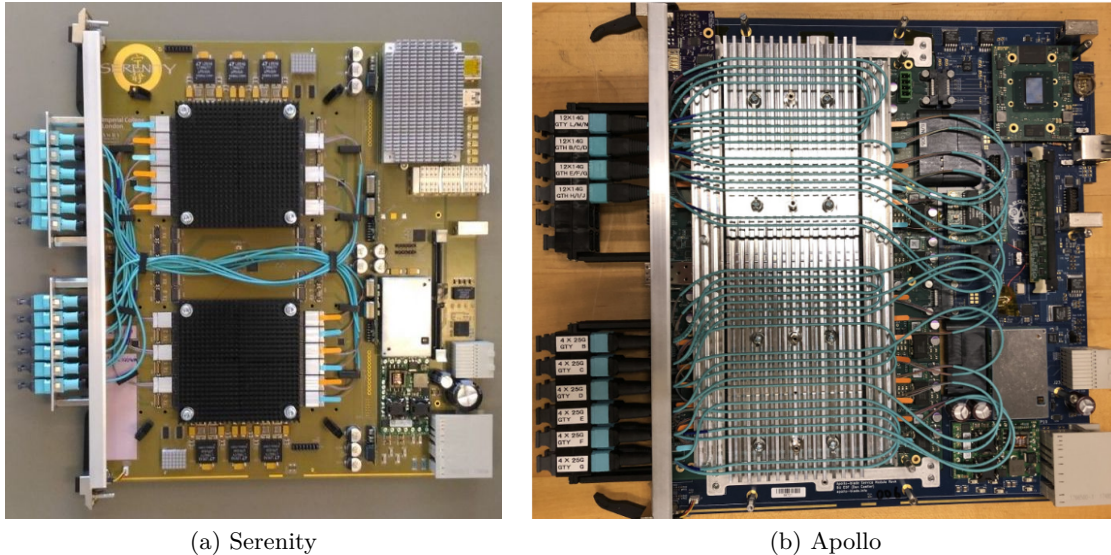


Figure 2.8: Photographs of the two ATCA prototype processing cards to be used for the CMS Phase-II track trigger.

2.2.3 DTC

Acting as the interface between the on-detector electronics and the Back End processing, the DTC serves numerous roles within the Track Trigger. Considering first upstream data, the DTC manages both the trigger datapath, containing stubs to be used within the L1T, and the Data Acquisition (DAQ) path, which contains data accepted by the L1T and must be sent on to the HLT. For the trigger path, the stub data sent from each connected module contains positional information which is local only to that module. This must be first decoded off the optical link and then converted to a global coordinate within the tracker nonant from which the module is located, so that it may be used downstream with in the TFP. Furthermore, the routing logic required to convert the input data stream from the positional domain to a temporal domain required for the time-multiplexed architecture of the TFP is also contained within the DTC. The DTC must also be capable of configuring and monitoring all tracker modules which are connected to it, for example temperature data to be used by the overall run control for the detector.

2.2.4 Track Finder

The act of reconstructing tracks in an online environment presents, computationally, the largest challenge of the Phase-II level 1 track trigger. As described above, the number of possible combinations of hits to check if they are compatible with a track scales exponentially with pileup. Throughout the research and development period of the Phase-II upgrade, a number of algorithms have been proposed to efficiently perform the task of track finding. The current design, named the “hybrid” approach, is the result of merging two previous approaches: “TMTT”[34] and “Tracklet”[35]. The overall algorithm can be separated into two steps, with each step being derived from one of the two original proposals.

The first of these steps is to seed the algorithm, by creating possible tracks from pairs of stubs in different module layers and projecting these out to the remaining layers of the tracker. With the six barrels layers labelled L1-6 and disks D1-5, the seeding pair combinations used are: L1+L2, L3+L4, L5+L6, L1+D1, L2+D1, D1+D2, D3+D4. This ensures high efficiency by reducing the likelihood of a track being missed by all combinations as well as ensuring the full coverage of the track finding η region is covered. The seeds, or “tracklets” are projected both radially inwards and outwards from the pair of hits, and a narrow window around that projection is searched for hits in other layers that are compatible. The initial tracklet as well as the stubs in the other layers are then combined into an track object. Only track objects with 4 or more hits in separate layers are considered as being valid at this stage. If multiple track objects share a hit, then the two objects are merged into one. This is a necessary step, given the high probability of duplicate initial tracklets being formed from different combinations of seeding layers.

The valid track objects are then fitted to a set of helix parameters in the second stage. This fitting is performed using a Kalman filter, which is capable of both the fitting as well as excluding hits within the track object which are not consistent with the fitted track. The performance of the algorithm, both in efficiency and resolution have been extensively studied using software simulation. The results of these studies can be found in figures 2.9a and 2.9b.

When implemented in firmware, the design is further separated into more granular modules, each of which can be seen in figure 2.10.

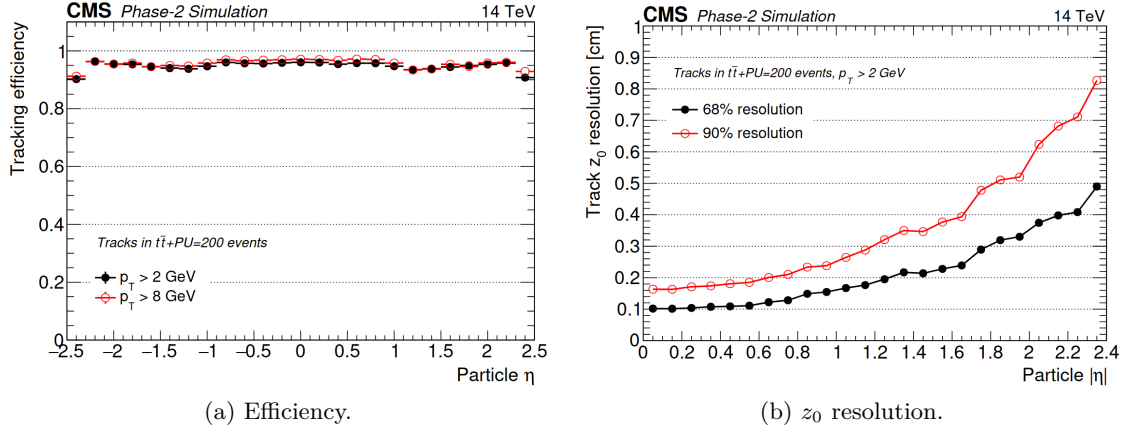


Figure 2.9: Plots showing the efficiency and z_0 resolution of the hybrid track finding algorithm. Source [36]

2.2.4.1 EMP

Developed alongside the design on the new ATCA processing cards, the Extensible, Modular data Processor (EMP) framework is a collection of firmware packages to enable rapid development of algorithms for FPGAs. The framework is inspired by that of an operating system for a standard PC, whereby the management of input and output links, timing and other such functions are abstracted away, allowing end users to focus on designing algorithms (figure 2.11). Typically, developers will then only write firmware which resides within the “payload” block of the FPGA, which has a simple interface of an input and output bus, along with clock pins for timing use. By having a payload block which is distinct from the infrastructure firmware, it also allows for algorithm designs to become transferrable between classes of FPGAs. But for the number of links available and resources within the payload, an end user generally does not need to put consideration into the exact model of FPGA which a given algorithm is designed for. For instance, early Serenity boards had Xilinx KU15P or KU115 FPGAs mounted to them, however these will be replaced by the larger VU13P in the final designs. Nevertheless, firmware which was designed for the former can easily be migrated to the new FPGAs once they become available provided it is sufficiently generic.

Along with the overall framework, EMP also comes with a set of predefined data types. For example, the input and output buses of the payload use the `lword` data type, which is made up of 64 data bits, as well as “valid”, “start”, and “strobe” metadata bits.

Alongside the firmware packages, the EMP framework also includes accompanying

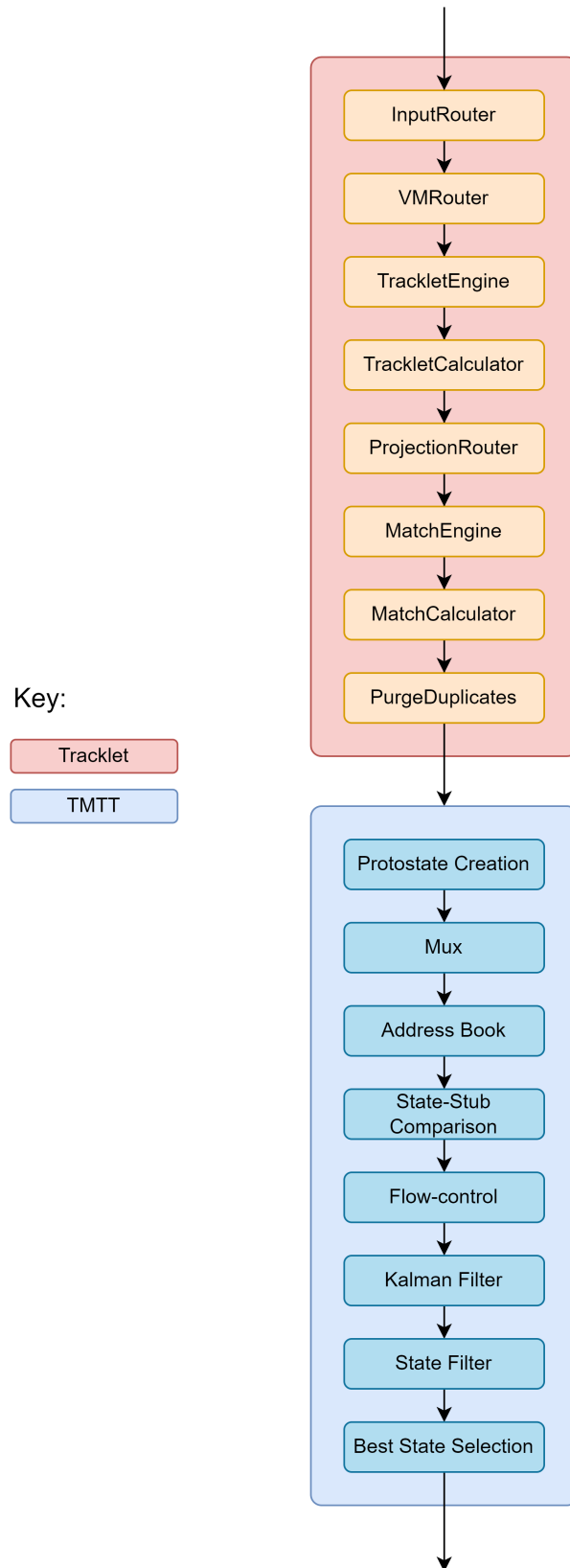


Figure 2.10: Diagram showing the individual firmware modules for the entire track finding chain. Details on each of the modules can be found in [35] and [34] for the tracklet and TMTT parts respectively.

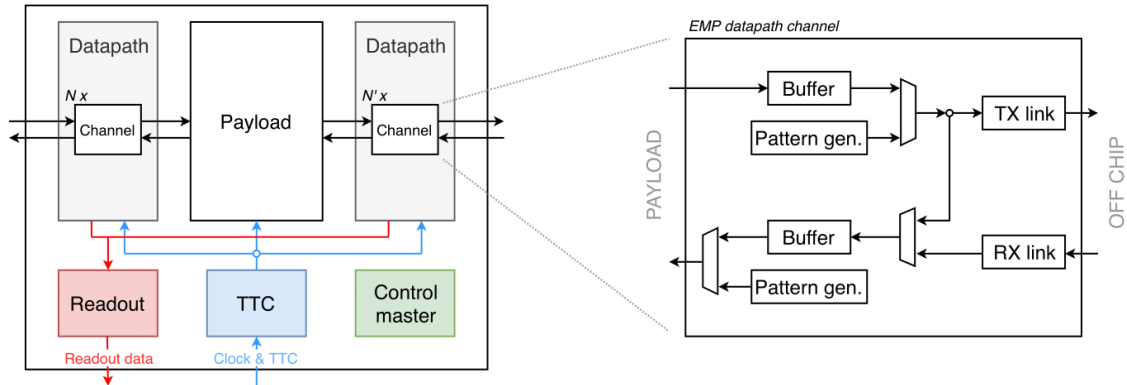


Figure 2.11: Diagram showing the overall structure of the EMP framework.

software to configure and control a processing card running an EMP firmware design. The Command Line Interface (CLI) for this control is named `empbutler` and allows for the configuration of the clock source used by the FPGA as well as the input and output links, for instance to inject a given pattern on the input links which can then be played through the payload algorithm to test it.

Whilst it is possible to build an EMP-based FPGA design through the Vivado Graphical User Interface (GUI), this can become cumbersome when working with large designs and limits the build to happen only on the user’s machine. As an alternative, the IPBus Build (IPBB)[37] tool was developed, to give the ability to script FPGA builds through a CLI. These tools have since been extended further to incorporate firmware builds into Gitlab Continuous Integration (CI) pipelines, whereby they are run on each commit to a given repository. This functionality was made possible through both additional software development and the construction of a dedicated cluster of servers at CERN on which the builds occur.

The cluster currently contains 8 build machines, each with 24-core hyperthreaded CPUs, and 128GB of RAM that are capable of multiple builds each in parallel. To avoid the duplication of installations and version discrepancies between machines, Vivado is installed on a separate “supervisor” server, which is then mounted over a Network File System (NFS) share to all of the build machines. Whilst this method reduces build performance compared to a local install, the reduction was only found to be approximately 1 % when the NFS mount used a 10 Gbps ethernet link between the machines. Kubernetes is used as the method for orchestrating and managing the cluster, effectively creating a single pool of resources from all the machines. The supervisor server also acts as the master node for the cluster and is the gateway for Application Programming Interface (API)

access. In Gitlab, there is then the functionality to associate a kubernetes cluster with a given group of repositories, meaning that an project within the group has access to the build resources.

The Gitlab pipeline itself is split into multiple jobs, each performing a single stage in the firmware build process described in section 1.3.1. An initial job also exists at the start of the pipeline to create the firmware project and to pull in all of the modules required for the build. These dependencies are stored in a separate file within the project and can be customised to use a specific branch, tag or commit of another repository for the build. An example dependencies file can be seen in figure 2.12. With the project created, each of the firmware build stages are executed sequentially. Each of the stages requires a different amount of computing resources, both in the number of CPU cores and RAM. For instance, the building of the pre-generated IP cores can be parallelised, utilising many cores, whereas the implementation phases is limited to a single core but requires a significant amount of RAM, particularly when targeting large FPGAs. As such, the pipeline allocates different resources in the cluster to each job, meaning that multiple builds make the most efficient use of the build cluster's total resources.

Jobs in the Gitlab CI are containerised, meaning that no data is intrinsically stored between jobs. Furthermore, it is not guaranteed that consecutive jobs in the same pipeline will run on the same server in the cluster, meaning that saving data locally between jobs would be redundant. Nevertheless, in the firmware build pipeline it is necessary to store the project and the results of the build to pass onto the next stage. In order to do this, the in-built artifacts feature was initially used, but limitations were found in the size of the artifact that was able to be stored between jobs and thus a bespoke data-passing solution was developed. Similar to the artifacts system, a Simple Storage Service (S3)[38] compatible data storage backend was used, with the final result of a given stage being packaged up and compressed before being sent to be stored. At the start of the subsequent job, the same package would be pulled from the S3 bucket and decompressed to be used in that job. By using a custom solution such as this, it also allowed for the storage server to be local to the build machines and make use of 10 Gbps network speeds.

Finally, to facilitate the ease of loading completed designs onto the test hardware itself, the final result of the build is stored in a separate bucket on the same S3 server and can be downloaded from anywhere on the CERN network.

The image shows a screenshot of a file named 'dependencies.yml' in a code editor. The file is 1.39 KB in size. It contains a YAML configuration for a CI pipeline, listing various dependencies with their sources, branches, tags, and URLs. The dependencies are organized into a 'sources:' section. The list includes: 'emp-fwk' (branch: k_fe_mgt, url: https://gitlab.cern.ch/p2-xware/firmware/emp-fwk.git), 'ttc_legacy' (tag: v2.1, url: https://gitlab.cern.ch/ttc/legacy_ttc.git), 'gbt-fpga' (tag: gbt_fpga_6_1_0, url: https://gitlab.cern.ch/gbt-fpga/gbt-fpga.git), 'lpGBT-fpga' (tag: v.2.1, url: https://gitlab.cern.ch/gbt-fpga/lpgbt-fpga.git), 'ipbus' (tag: v1.9, url: https://github.com/ipbus/ipbus-firmware), 'histogram' (branch: master, url: https://gitlab.cern.ch/dmonk-emp-projects/histogram.git), 'tcDs2' (tag: v0_1_1, url: https://gitlab.cern.ch/cms-tcds/cms-tcds2-firmware.git), 'tclink' (commit: fda0bcf, url: https://gitlab.cern.ch/HPTD/tclink.git), 'gbt-sc' (tag: gbt_sc_4_1, url: https://gitlab.cern.ch/gbtsc-fpga-support/gbt-sc.git), 'dtc' (branch: master, url: https://gitlab.cern.ch/cms-tracker-phase2-data-processing/BE_firmware/dtc.git), 'dtc-stub-processing' (branch: master, url: https://gitlab.cern.ch/cms-tracker-phase2-data-processing/BE_firmware/dtc-stub-processing.git), 'verilog-ethernet' (tag: v1.0.0, url: https://github.com/KristianHahn/verilog-ethernet.git), and '10Gb-ethernet' (tag: v1.0.0, url: https://gitlab.cern.ch/cms-tracker-phase2-data-processing/BE_firmware/10Gb-ethernet.git).

```
1 ---
2 sources:
3   emp-fwk:
4     branch: k_fe_mgt
5     url: https://gitlab.cern.ch/p2-xware/firmware/emp-fwk.git
6
7   ttc_legacy:
8     tag: v2.1
9     url: https://gitlab.cern.ch/ttc/legacy_ttc.git
10
11   gbt-fpga:
12     tag: gbt_fpga_6_1_0
13     url: https://gitlab.cern.ch/gbt-fpga/gbt-fpga.git
14
15   lpGBT-fpga:
16     tag: v.2.1
17     url: https://gitlab.cern.ch/gbt-fpga/lpgbt-fpga.git
18
19   ipbus:
20     tag: v1.9
21     url: https://github.com/ipbus/ipbus-firmware
22
23   histogram:
24     branch: master
25     url: https://gitlab.cern.ch/dmonk-emp-projects/histogram.git
26
27   tcDs2:
28     tag: v0_1_1
29     url: https://gitlab.cern.ch/cms-tcds/cms-tcds2-firmware.git
30
31   tclink:
32     commit: fda0bcf
33     url: https://gitlab.cern.ch/HPTD/tclink.git
34
35   gbt-sc:
36     tag: gbt_sc_4_1
37     url: https://gitlab.cern.ch/gbtsc-fpga-support/gbt-sc.git
38
39   dtc:
40     branch: master
41     url: https://gitlab.cern.ch/cms-tracker-phase2-data-processing/BE_firmware/dtc.git
42
43   dtc-stub-processing:
44     branch: master
45     url: https://gitlab.cern.ch/cms-tracker-phase2-data-processing/BE_firmware/dtc-stub-processing.git
46
47   verilog-ethernet:
48     tag: v1.0.0
49     url: https://github.com/KristianHahn/verilog-ethernet.git
50
51   10Gb-ethernet:
52     tag: v1.0.0
53     url: https://gitlab.cern.ch/cms-tracker-phase2-data-processing/BE_firmware/10Gb-ethernet.git
```

Figure 2.12: Example dependencies file used within the IPBB Gitlab CI build pipeline.

Chapter 3

CMS Online Software

3.1 Introduction

The upgrade to the CMS tracker and BE electronics in Phase-II provides the opportunity to not only upgrade the hardware, but also the software used to manage and control all aspects of it. In particular, the use of SoMs on each of the processing cards used in the BE system means that there is far greater scope to run this software as close to the hardware as is possible. There will be approximately 400 of the BE processing cards to monitor and control in the final system, which also provides a challenge of building the software to scale effectively. In order to overcome these challenges, the online software built for the CMS tracker leans on industry trends such as containerisation[39] and RESTful[40] applications.

3.1.1 Containerisation

The emergence of containerised computing has caused a significant change in how software is deployed. The philosophy of containers is similar to that of virtual machines, a more mature technology. Both allow for a program to be executed in an environment that is isolated from the host. Similarly, the dependencies for a given piece of software are only installed within this isolated environment, thus reducing the likelihood of conflicts between multiples pieces of software. Containers differ from virtual machines by sharing the base kernel of the operating system between the host and containers, as opposed to virtual machines where a new kernel is installed. Sharing the kernel reduces both the size of containers relative to virtual machines and the overhead in operating them. It is the intention of the online software group to use containerised computing throughout the

software stack.

3.1.2 Requirements

The requirements of the online software for CMS are detailed in the Data Processing Systems (DPS) specification document[41]:

- **Control:** The online software must be able to configure the tracker electronics in multiple different scenarios: standard data-taking runs, initiated by central CMS run control applications; and expert operations (e.g.front-end calibrations) involving either the entire tracker or an individual system, initiated through a GUI. It must support the retrieval of configuration parameters from databases, or equivalent data stores that ensure traceability of the system's configuration during data-taking periods. It should also support the editing of these configuration parameters in a manner that minimises the likelihood of human error.
- **Monitoring:** The online software must continuously collect monitoring data from the electronics, to identify erroneous/anomalous behaviour in the FPGAs, ASICs and supporting components. In order for on-call experts to be able to diagnose and fix problems quickly, in addition to providing a high-level summary of the system's status, monitoring GUIs must allow for rapid access to pertinent fine-grained monitoring data.
- **Test stands, installation and commissioning:** The online software should be deployable and usable in test stand environments (both within the CERN network, and offsite), and should support the independent operation of certain subsets of systems as is required for installation and commissioning on the detector site, and for the tracker integration facility.
- **Multiple systems & ATCA boards:** The SoM components of the online software framework should make use of software developed by different groups that implement low-level control and monitoring procedures. They should also work on both the Serenity and Apollo platforms, meaning that the online software design needs to be flexible enough to potentially support different SoM hardware and capable of handling the different functionalities of the blades.

- **Deployment:** The online software should be designed to support its rapid redeployment across the different levels of the system (SoMs and off-board computers) while providing the ability to rigorously track and control the specific versions deployed and the history of past versions. It must be deployable within the central DAQ infrastructure provided at Point 5 and meet the requirements set by CMS sysadmins, for example using OS images based on CERN CentOS.

3.1.3 Architecture

The proposed design for this software comprises three hierarchical layers. Starting at the lowest level, software on each SoM must exist to collect monitoring data and execute commands. This software is collectively known as HERD. A benefit of the Time-Multiplexed architecture within the track trigger means that there will exist many cards with common configurations. Interfacing with the HERD instances is the Shep layer, which resides off the processing cards and can control one or more HERD instances at a time. At the Shep level, HERD instances are combined into logical groups, such as the DTC and TFP, with a cumulative state. Finally, the top layer, known as crook, will allow control of the entire system and receive the states of each Shep group. Collectively, this stack of software would then be known as ShepHERDsCrook. An overview of the architecture can be seen in figure 3.1

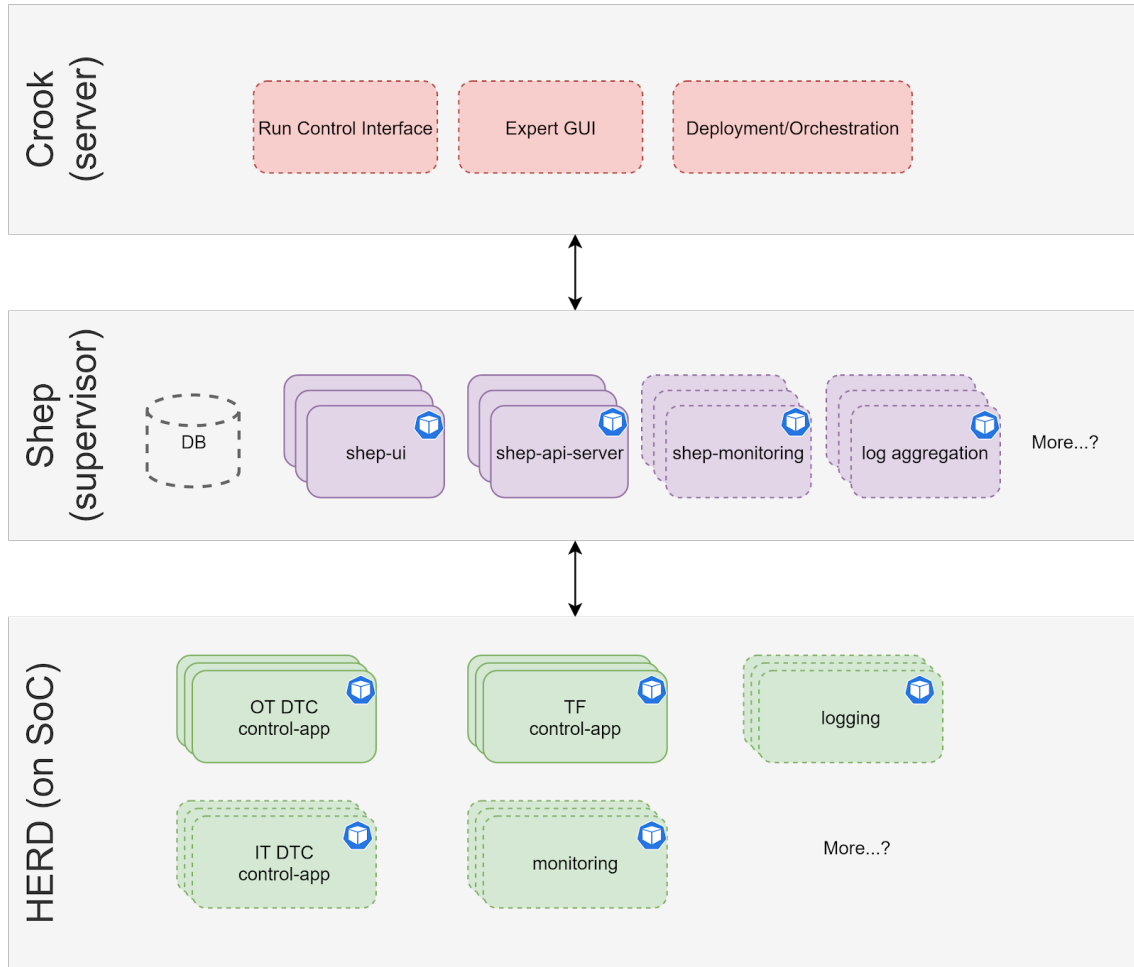


Figure 3.1: Diagram showing the overall architecture of the CMS Tracker Online Software stack.

3.2 HERD

Given the location of the software on the processing cards themselves, the primary role of HERD is to interface with the hardware contained on the card. It must similarly outwardly expose a coherent interface so that the board may be controlled and monitored remotely. The software designed must therefore have both elements that are common across all HERD instances, but also aspects which are specific at both a physical level (Serenity vs Apollo) as well as between the functions of the boards (DTC vs TFP). To achieve this, a plugin-based approach was used, whereby a base HERD application can be loaded with the necessary plugins required for a given board and application.

3.2.1 Hardware Interface

Whilst the majority of the algorithmic aspects of a firmware design are fixed at compile time, it is still possible to build in functionality to control data flow and update aspects such as variables and Look Up Table (LUT)s. A method for interfacing with FPGAs over multiple transport protocols, including ethernet, AXI, and PCIe, named IPbus[37], was developed as part of the Phase-I CMS trigger, to achieve this control. IPbus can be considered as the collective term for the whole system of interfacing an FPGA over ethernet, but is in fact made up of a number of individual aspects. First is the IPbus protocol itself, which is the communicative language and rules used by both the client and server. The IPbus firmware library provides an open-source implementation of the firmware required to communicate over the protocol, which has been developed for and tested across multiple experiments. uHAL is a software library that provides the interface for other software stacks to communicate over the IPbus protocol.

The two most common IPbus utility firmware blocks used in designs are the IPbus register and BRAM modules. The former provides read and/or write access to a single 32-bit word within the firmware, which can then be used for either status or control bits within an algorithm. A common use for a control bit is as a reset line for a block of logic within the firmware. IPbus BRAMs allows for significantly more data to be read out or supplied to the algorithm, with examples being the EMP link buffers, and the module correction LUT for the DTC stub processing, both of which are controlled through the HERD application.

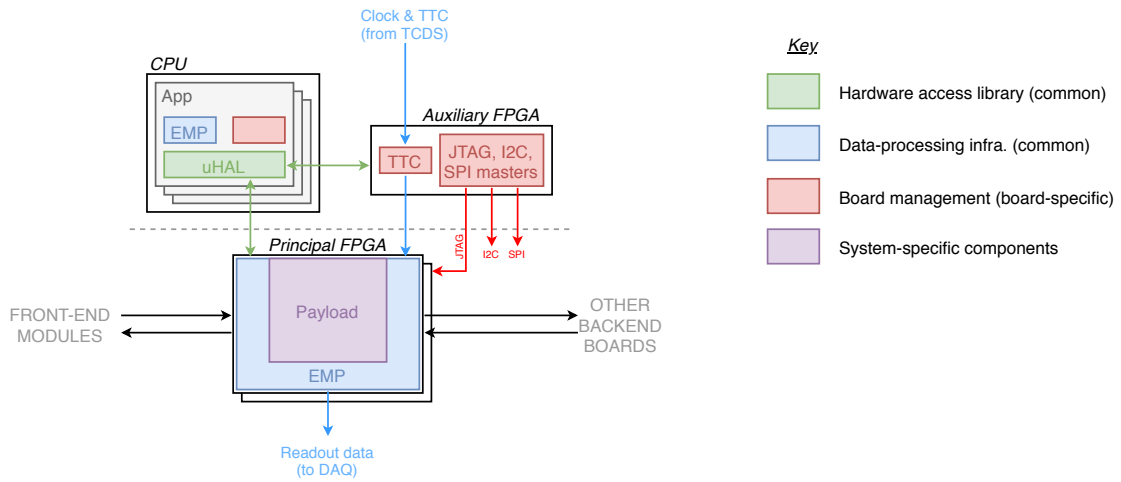


Figure 3.2: Diagram showing structure of the infrastructure firmware and software used in the BE electronics.

3.2.2 Base Application

The base application for HERD has two primary roles: provide the framework for which plugins are loaded into and host the HTTP server used for external communication. Based on the SWATCH framework[42] for Phase-I, the framework provides the capability to run both single commands and state machines.

The network interface underwent a number of revisions during development. Initial versions used a combination of ZeroMQ[43] and MessagePack[44] for communication, however this was superseded by the use of HTTP and JSON to form a RESTful interface. The reason for the change was mainly to improve the ability for the software to be both used in small test stands in laboratories as well as the final system. Compared to MessagePack, JSON is human-readable, meaning that it was possible to simply navigate to the address of the board in a web browser and receive meaningful data, rather than have to install a full off-board software stack in order to start communicating. Read-only operations in HERD, such as providing the details of the devices connected to the board, are conducted using GET requests to endpoints in the server, whilst operations which require the transmission of data to the board, such as configuration parameters, use POST requests.

3.2.3 Plugins

All of the cards managed by the ShepHERDsCrook framework will run firmware images that use the EMP framework. As described in section 2.2.4.1 this provides many of the common functionalities required by both the DTCs and TFP, such as clock distribution and board-to-board I/O. Many of these aspects are controlled by the use of IPBus and in turn through the SoM. The EMP plugin for HERD provides the commands and relevant parameter sets required to perform these tasks. Each of these commands is registered to the base application when the plugin is loaded, making their associated endpoints available to the HTTP server. The commands available through the EMP plugin closely match those available through the empbutler CLI.

For the board specific commands, there exists Serenity and Apollo plugins. The commands registered through these plugins include those to power on and off each FPGA on the board, as well as program the FPGAs. In the case of the latter command, each card has a significantly different implementation. Serenity makes use of a Peripheral Component Interconnect express (PCIe)/JTAG link to program each FPGA on the board, whilst Apollo uses an AXI Chip-to-Chip link between the FPGA built into the Zynq processor

on the SoM and the FPGAs on the board itself.

For a complete system, both the EMP and board specific plugin would be loaded, along with a plugin to control the relevant payload algorithm, as shown in figure 3.3.

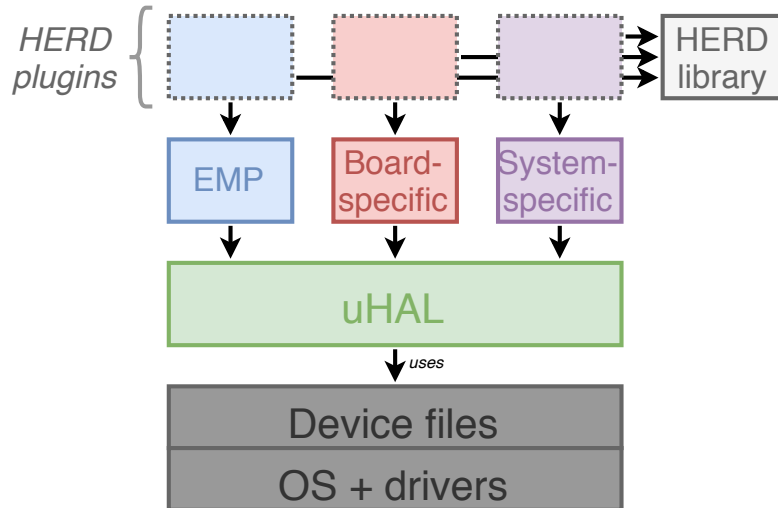


Figure 3.3: Diagram showing the plugin structure for a HERD application.

For development purposes, a “dummy” plugin was also developed, which contained within it a selection of commands and a state machine that could be used to test other aspects of the software stack. Unlike the other plugins this plugin required no hardware connectivity, thus could be run on any machine.

3.3 Shep

Forming the intermediate layer of ShepHERDsCrook, Shep is the first part of the software which resides on machines located off the processing boards themselves. Given the requirement for the software to be scalable from single board test stands to the full system at Point 5, Shep must both be capable of operating as an access point to the software stack, as well as being part of a larger overall system involving the use of Crook.

A design philosophy taken when developing the Shep software was to have no assumptions on the capabilities of the HERD endpoints to which it would connect. Instead, the commands and state machines available to the user through Shep would first be polled from the HERD endpoints themselves, collated and presented to the user. This philosophy ensures that the Shep software can have a single common version that is compatible with all possible HERD endpoints.

Shep itself is made up of three separate applications, the API server, database and

GUI server. The API server provides an API that is capable of sending requests to one or multiple HERD endpoints concurrently. Within the API is also the capability to register, delete and edit the records of both a single board and a collection. Given the stateless nature of containers, meaning that when one is deleted, so is all stored data within that container, a separate database application is required to store information about the boards registered to the Shep instance. The database is a standard postgresSQL[45] instance, with variables such as the IP/hostname, date of registration and Universally Unique IDentifier (UUID) stored for each board. Information for a collection of boards is similarly stored in the database, with an additional field containing a list of foreign keys, each of which corresponds to a row within the boards table in the database.

The API server is built upon the FastAPI framework[46], which is a Python library for developing APIs. Requests sent to endpoints representing collections of boards are fanned out using asynchronous processes, with the results all collected before they are returned. Asynchronicity works well in this instance, given that the majority of the time-cost of a request is in waiting for the response from the HERD endpoint, rather than in processing at the Shep level. The concurrent request functionality was successfully demonstrated with up to 8 endpoints without any noticeable increase in response time.

Whilst it is possible to achieve all necessary functionality in controlling HERD endpoints through the API server by making HyperText Transfer Protocol (HTTP) requests, this is not user-friendly, particularly for users which are not familiar with the software structure. Instead the GUI server provides a web-based interface for the API. The GUI has undergone a number of iterations throughout the development of Shep, with early version based on templated HyperText Markup Language (HTML) code and the Django[47] web framework. The most recent version uses declarative JavaScript in lieu of the HTML, using the vue.js framework[48]. Regardless of the implementation, the design has remained constant throughout.

A user is presented initially with a homepage in the GUI, giving a list of available boards and also all collections of boards which have been created, see figure 3.4. From here, new boards can either be manually created, as well as new collections of boards.

By clicking on the UUID for a boards, the user is presented with the board landing-page, whereby the boards details can be seen. Tabs to control state machines and enact commands can also be selected from this page. Board details and available commands are polled from the board itself during the loading of this page, ensuring that are as up to

Shep Board Manager

Admin

Applications

New Application

Name	Date Created	Boards
stack-test	25 Jun 2020, 11:45 a.m.	4a34ab04, 445429f3, 186f4147,

Boards

UUID	IP Address	Board Type	Date Registered	Status
4a34ab04	192.168.10.47	herd-control-app	25 Jun 2020, 10:41 a.m.	Offline
445429f3	192.168.10.158	herd-control-app	24 Jun 2020, 7:39 p.m.	Online
186f4147	192.168.10.184	herd-control-app	24 Jun 2020, 7:39 p.m.	Online

Figure 3.4: Screenshot of the Shep GUI homepage.

date as possible.

3.4 Crook

As a result of its required close integration with upstream control applications within the overall CMS run control, this software has yet to be developed, and therefore the interfaces are currently unknown. Without such information, it is not possible to develop a software layer that can interact with it. The configuration files used to generate and manage the kubernetes cluster will also form part of the Crook package.

Some cluster-level software considerations have been made however, in particular for logging and monitoring. Prometheus[49], a cloud-native aggregation platform for both logging and monitoring information will be used for the online software. A single instance, or a cluster of federated instances will collect logging information from all HERD and Shep instances within the cluster. Monitoring data will include both hardware health information from all nodes in the cluster as well as records of the performance of the algorithms running in the FPGAs.

All DTCs will also record temperature information from all connected FE modules within the tracker, giving approximately 15,500 individual variables which must be recorded. At a polling frequency in the order of seconds, this presents a challenge to the scalability of the aggregation system.

3.5 Deployment

As described above, it is proposed that every processing card used in the track trigger will have an SoM attached to it. This is a paradigm shift from the current computing architecture and it is hoped to bring significant benefits to performance. Computing on the edge, as would be done using these SoMs however comes with its own set of challenges. The majority of these arise from significant increase in the number of individual computing units. In an architecture of few, but large, servers controlling many cards each, it is possible to manage these servers in an ad-hoc manner. It would be possible to update and solve issues on each machine individually without a great cost in efficiency. Trying to scale such a service to the order of thousands of SoMs, such an approach no longer becomes feasible. In order to provide a solution to these challenges, the Online Software group looked to a similar paradigm, that of the Internet of Things (IoT), where such problems can also arise.

3.5.1 Balena

Balena[50] is an orchestration and compilation toolkit designed specifically to support IoT devices. On the edge it includes the operating system and container architecture in which services run, allowing for both Infrastructure as Code (IaC) and direct control such as Secure SHell (SSH) access to each device in the cluster. In a similar manner to the CI pipelines offered by Gitlab, it also offers the functionality of building containers to run on the cluster, triggered by commits to a repository and then deploy this new code directly to devices.

There are, however, a few drawbacks to using balena. Firstly, from a technical standpoint, it is not possible to have a single cluster which is a mixture of both ARM and x86 architectures. This is a limitation of the software but may present itself an issue given that SoMs of both architectures are proposed for the track trigger. Secondly, whilst the core functionality of balena is open source and free, many quality-of-life features require a paid licence to use. This contradict CERN’s policy on being open source and thus these features may have to be forgone.

Finally, whilst balena provides the full pipeline from code to deployment, much of this pipeline has already been fulfilled by a self-hosted instance of gitlab provided by CERN. Within gitlab it is possible to store code in a remote repository, but also test and then build that code using gitlab’s CI pipelines. This leaves only the deployment aspect missing before the functionality of balena is possible from gitlab itself. Whilst not intrinsic to gitlab, much effort has been placed into integrating gitlab and Kubernetes, which could provide a solution to deploying HERD code onto the SoMs.

Balena was tested for the online software stack using a small scale cluster of Raspberry Pi devices as a proof of concept for the toolkit. Whilst successful, the drawback of the inability to have a heterogenous cluster of architectures was found to be an issue once a set of Com-Express devices were also included in the test setup. This was the primary driving factor in exploring alternatives, such as Kubernetes.

3.5.2 Kubernetes

Kubernetes is “an open source container orchestration engine for automating deployment, scaling and management of containerised applications”[51] primarily developed by Google to allow them manage their services through the use of containers. This is achieved through a concept called IaC. In the IaC paradigm, hardware is not considered as individual, but as

a collective pool of resources which can be assigned as needed for the applications to run. Furthermore, the scale and nature of what is run on the cluster is defined in configuration files, rather than multiple actions by the user. It is already intended that all layers above the HERD will be run on a kubernetes cluster.

It may also be possible to run the HERD instances within a cluster too. As the HERD instances must communicate with hardware on the Serenity and Apollo cards, they must be locked to those nodes, with a single copy per node. Such a configuration is possible in kubernetes and is named a DaemonSet. A DaemonSet deploys a single instance of a container to every node in the cluster, with filtering possible based on node labels. The benefit of IaC comes when a new node is added to the cluster, for instance to replace a faulty card, this node, if appropriately labelled, will automatically have the correct HERD instance loaded onto it. This means that the cluster can scale without any extra input from an administrator, other than labelling nodes as they are added. This will be a significant benefit in the final system of approximately 400 HERD endpoints. It is also possible to have a heterogeneous kubernetes cluster containing both x86 and ARM based nodes, a limitation in the balena approach.

Using kubernetes for the Shep layer will also see benefits over conventional monolithic designs. The first is in horizontal scaling of the applications. A functionality of kubernetes allows for individual deployments, such as those for the Shep API server and GUI respectively, to be scaled in real-time with demand. Each deployment is load-balanced within kubernetes such that a deployment with one or more pods looks identical to other services within the cluster and outside it. Given that both of the Shep servers are stateless, they are well suited to this scaling. The use of deployments in this manner should optimise computational resources at Point 5.

There is a strong push within the CERN community towards the use of Kubernetes within the upcoming upgrades to all detectors, making this approach more likely to be accepted compared to the proprietary software stack of balena. An example architecture of the online software within the Kubernetes paradigm can be found in figure 3.5.

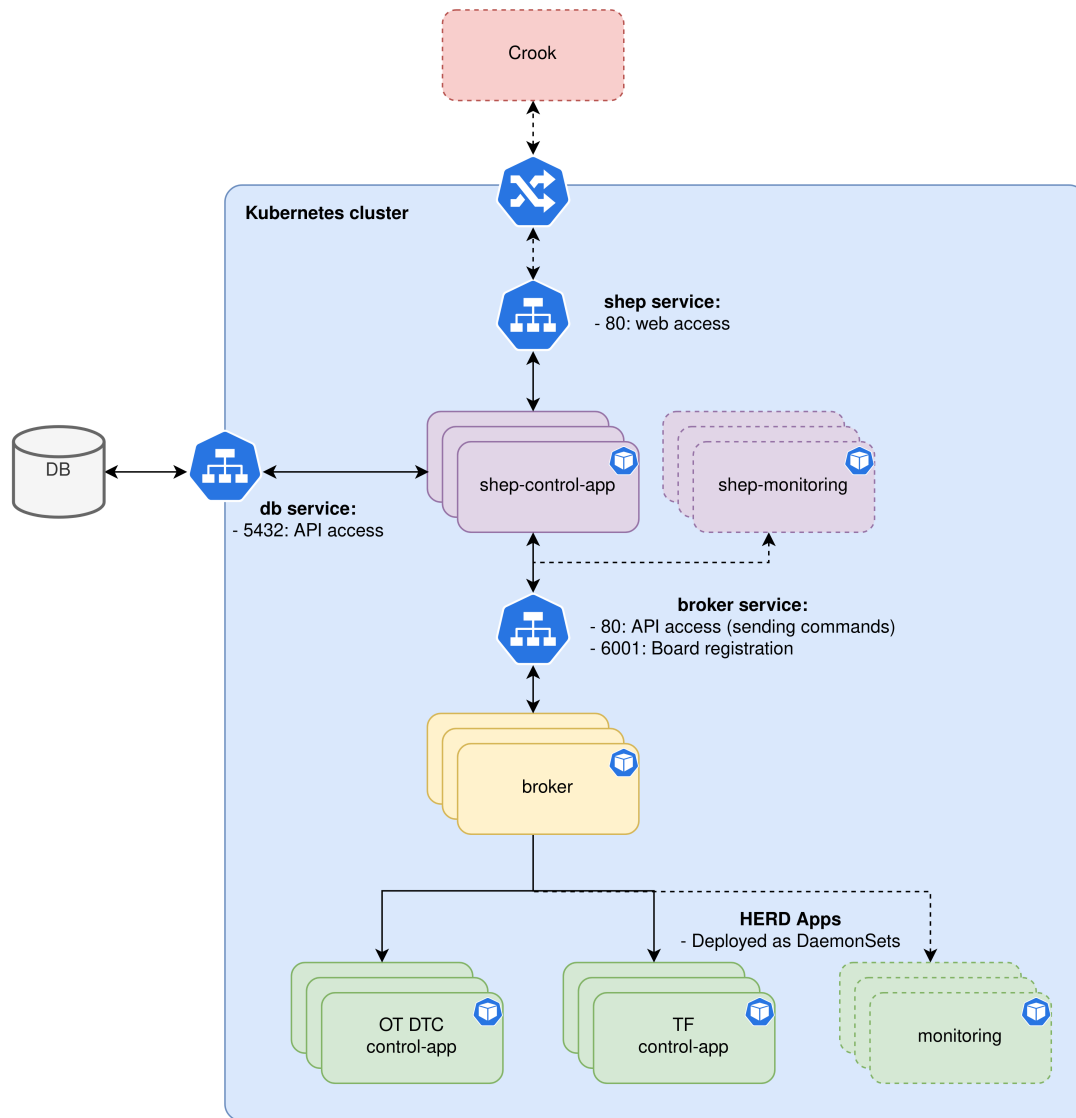


Figure 3.5: Proposed Kubernetes architecture for the Track Trigger online software.

Chapter 4

DTC Development

The DTC will perform a role for both the trigger and DAQ systems in CMS: for the trigger it must decode, format and route stub data to be reconstructed as tracks and for the DAQ it must collate and packet data from accepted events. The firmware to perform these two function is mostly separate within the FPGA, thus is presented in this chapter in two separate sections. The majority of the work was on the former of the two paths, thus it is similarly focused on in this chapter.

4.1 Trigger Path

Within the DTC, the trigger path comprises two distinct blocks in firmware: the link interface and stub processing. The former manages the bidirectional connection between the DTC and tracker modules on the detector. The latter performs a conversion of the stub data from a local coordinate on the module itself to a global coordinate within the ϕ processing nonant and then routes the stubs to the correct downstream TFP board for track reconstruction. An overall block diagram for the DTC Trigger path can be seen in figure 4.1.

The input to the trigger path is seventy-two links, each connected to a single tracker module. Each link is individually decoded its own Link Interface instance, with stubs again individually formatted into the global coordinate system of the tracker. As described in chapter 2, track finding is time-multiplexed by a factor of eighteen, meaning that the output of the DTC is thirty-six links, corresponding to eighteen Bx for each processing nonant covered by the given DTC. The role of the routing block is organise the stubs from each of the input links such that they are sent to the correct output link corresponding to

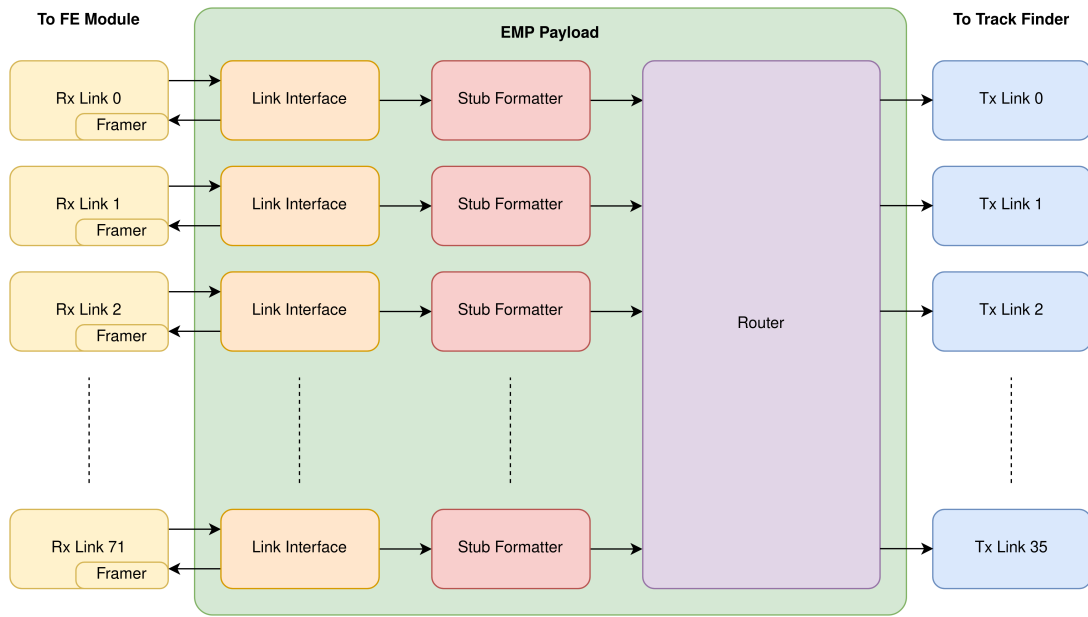


Figure 4.1: DTC block diagram.

their Bx.

4.1.1 EMP xGBT interface

In order to communicate with the detector modules, it was necessary to write firmware for the EMP datapath specific to the GigaBit Transceiver (GBT) and lpGBT protocols. These linked together the previously written lpGBT-FPGA firmware block that was developed by CERN with the standard EMP interfaces for a data link. The full datapath for the lpGBT version can be found in figure 4.2.

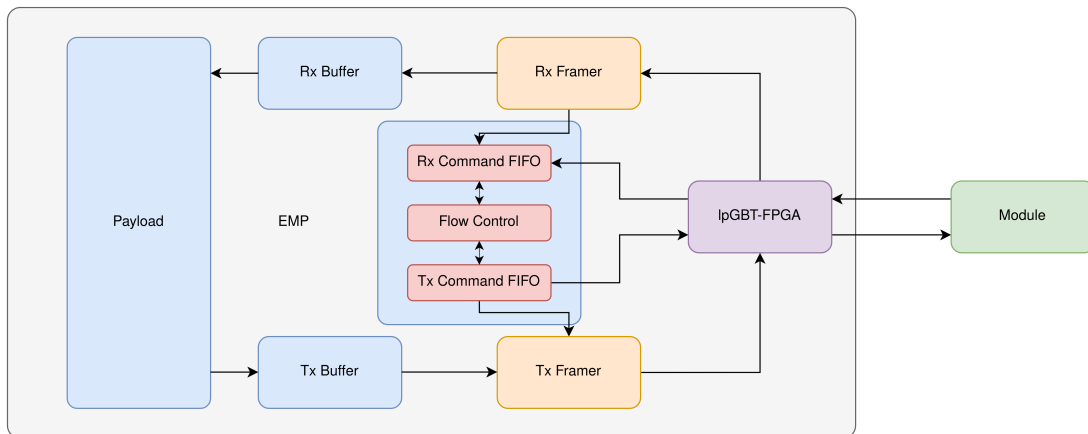


Figure 4.2: Block diagram of the lpGBT datapath for EMP.

The interface consists of two parts: the framer and flow control logic. The framer acts to convert the data between the formats used in the EMP payload and the lpGBT-FPGA block, both in the Receiving (Rx) and Transmission (Tx) directions. The flow control logic allows for addition commands and information to be passed on the lpGBT link outside of the EMP payload. This includes the “slow-control” commands used to configure the link itself.

4.1.2 Framer

As shown for the 2S module in figure 4.3, a boxcar is transmitted from the module to the DTC over 8 Bx, totalling 320 bits. Data from the lpGBT-FPGA core is presented to the EMP datapath in the form of a single, wide word, once per Bx, equivalent to a frequency of 40 MHz, and contains $1/8^{\text{th}}$ of the total boxcar. There are two possible widths for this word, 80 bits for a GBT frame and 112 bits for a widebus frame[52]. The widebus frame foregoes the Forward Error Correction (FEC) bits in order to maximise bandwidth. In the case of the OT modules, this widebus data format was chosen. The data format used is determined upstream at the module level, and then reflected at build time within the FPGA design.

The data sent in the 40 MHz word is organised in a column-wise fashion, meaning the eight bits of data from each External Link (e-link) are sent together. This contrasts with the row-wise data format used for the information contained within the boxcar. Thus in order to optimise the decoding process of the boxcar, it is necessary to convert between the two formats within the framer. The higher frequency of the EMP payload clock, typically eight or nine times the 40 MHz LHC base-clock, decoding can be further optimised if a row of the boxcar, equalling five bits for the 2S module, is sent on every clock cycle.

In the framing process, the five sets of 8 bits corresponding to the stub data e-links are first collated into a single two-dimensional array with shape (5, 8). A transpose operation is then performed on the array so that its primary index now has length 8, before iterating over the array, outputting a slice of length 5 bits on each clock cycle. A diagram of this operation can be seen in figure 4.4. The data then sent to the payload region is organised identically to that given in the specification.

Given the eight clock cycles on which data is output from the framer, it follows that the payload must have a minimum frequency of 320 MHz. Frequencies higher than this are also possible, whereby the extra clock cycles, one at 360 MHz or two at 400 MHz, are

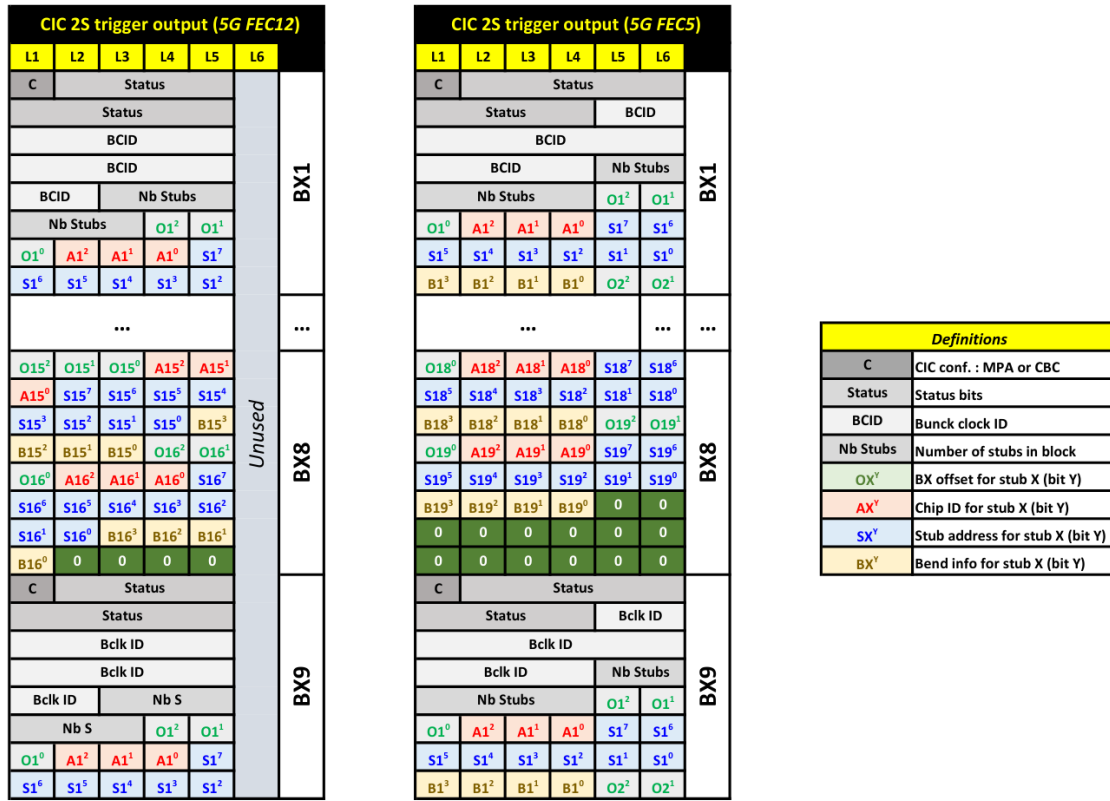


Figure 4.3: Specification of CIC output trigger data format (2S modules). Source [53]

used to send invalid data, typically all zeros. Data validity is indicated by the uppermost bit in the `lword`, using 1 for valid data and 0 for invalid. Inside the payload, this bit is transferred to the strobe bit within the `lword` to allow for the entire 64-bit data field to be used for processing.

For downstream data sent from the DTC to the modules, a reverse of the process described above is performed. Data from the multiple clock cycles equivalent to a single 40 MHz clock cycle are collated together and then sent to the lpGBT-FGPA core as a single word.

4.1.3 Stub Alignment

There exists two possible modes of data misalignment within the trigger path: intra-link and inter-link. The former exists if the five or six e-links which comprise the stub data from the CIC to the lpGBT are out of sync. In this case, a bit from an out of sync e-link would arrive early or late compared to the others (see figure 4.5). This would then lead to that bit being in a different word sent from the framer. Such an issue can be corrected in two different points in the trigger path: the first is at a module level, whereby delays can

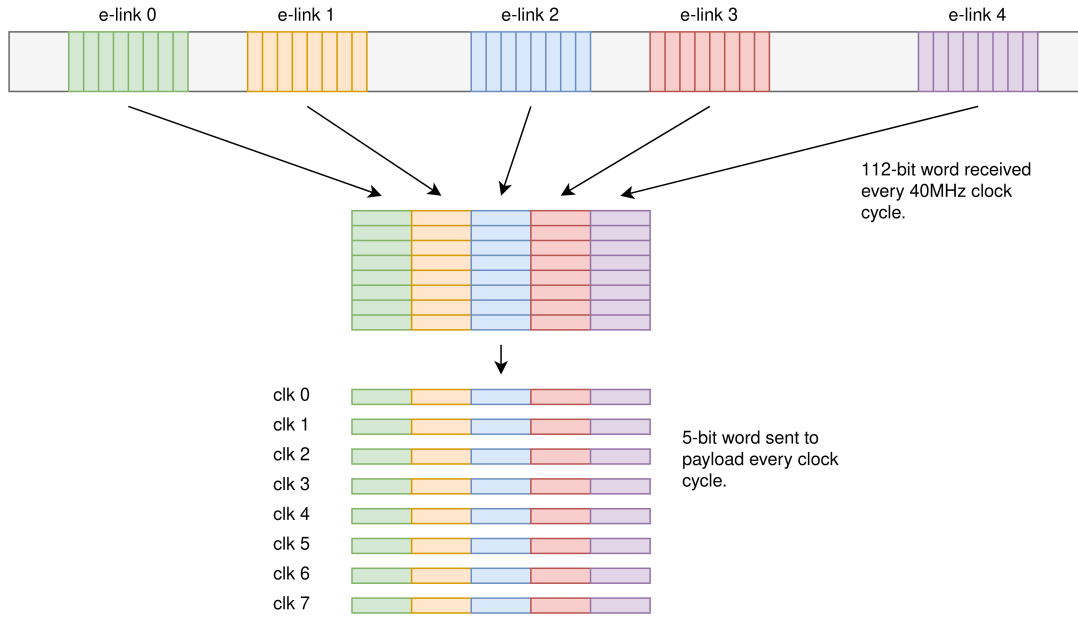


Figure 4.4: Diagram showing the framing process for stub data originating from a single CIC on the Front End module.

be introduced onto individual e-links to correct for a delay once it has been identified and the second is to add a similar mechanism into either the framer block or to the payload pipeline before the data is used for stub extraction. In this latter case, data would be buffered in a shift register with a configurable tap point for each e-link to define the delay. Correction to misalignments cannot however, be conducted during normal operation of the front end modules, as in order to identify a delay, a known, fixed pattern must be sent by the modules, which can only be achieved as part of the calibration phase before the start of data taking. The pattern chosen consists of bits sent in succession on each e-link, with the ordering: [1, 1, 1, 0, 1, 0, 1, 0].

Inter-link misalignment arises from the possibility of data arriving from two independent modules at different times to the FPGA. Given the multiplicity of fibre lengths that will be used within the Tracker for CMS and the finite speed of light within an optical cable, it may be possible to see differences in arrival times of multiple clock cycles at the FPGA. As with the intra-link corrections, this issue can be addressed both at the module level and the DTC. Minor corrections can be achieved by again adding a delay to the data sent by the module, though this is limited to a maximum of 25 ns, or one bunch crossing. Should the difference in arrival time be greater than this, instead the correction must be achieved in the DTC itself. The corrective method would again be similar to that for



Figure 4.5: Diagram showing a single e-link, with index 4, which is late sending data to the lpGBT. Each colour corresponds to different data packet.

The input and output ports for each detector link in the datapath are each connected to a copy of the Link Interface firmware within the EMP payload. This firmware block is the combination of multiple sub-modules, as shown in figure 4.6, so that it is capable of independently handling both the trigger and L1A upstream data, as well forming the commands sent downstream to the modules.

Before extracting both the header information and following stubs from a boxcar, the start position of the boxcar in the data stream must first be determined. Given there is no explicit marker in the stream to denote the start of a boxcar, the beginning must

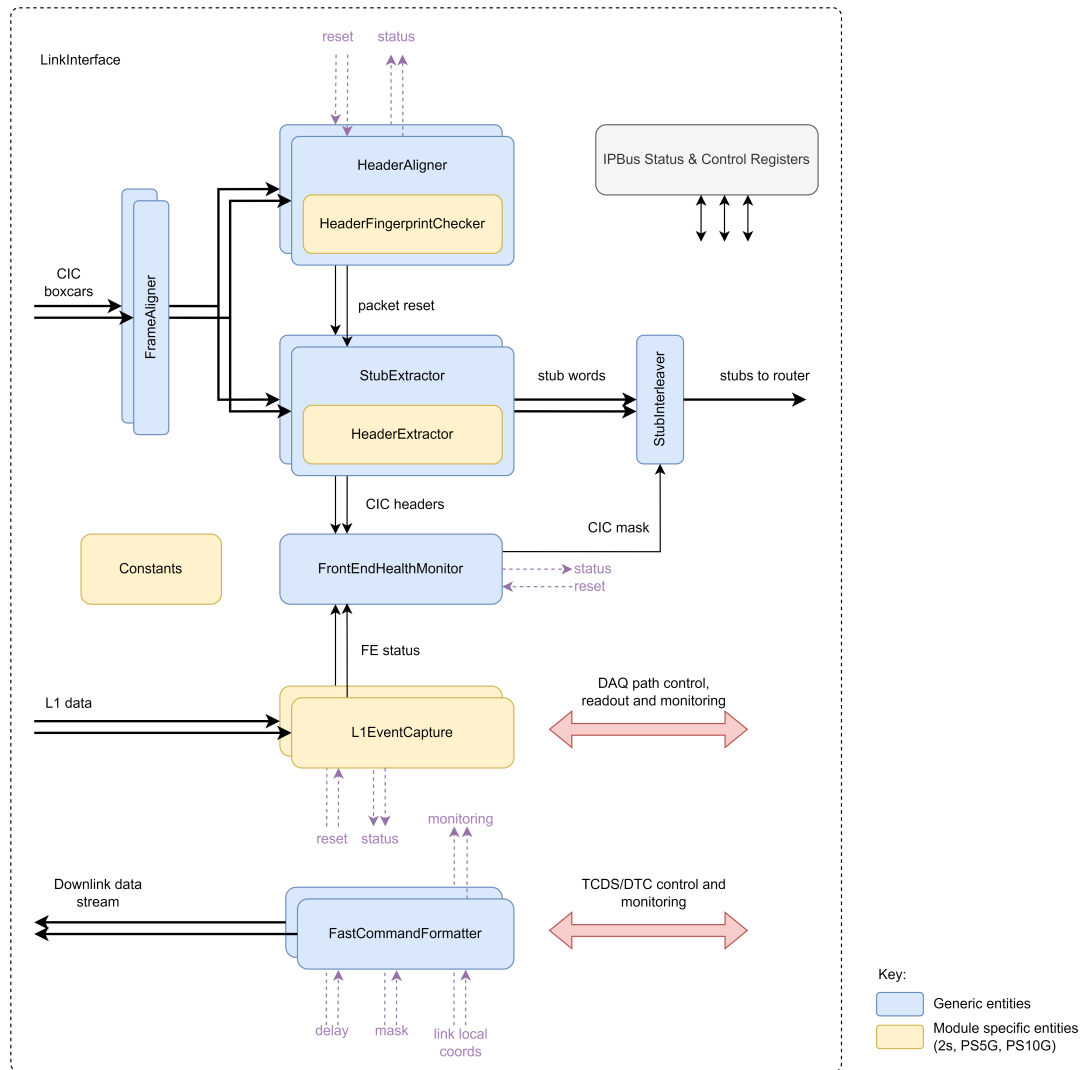


Figure 4.6: Link Interface firmware block diagram.

instead be inferred through information which remains constant between boxcars. With the knowledge that every boxcar is 64 frames long, it is then possible to compare current frames with those from 64 frames previous to infer the start of a packet using particular properties of data format.

The boxcar contains within it three markers which can be used for these comparisons: the module type bit, which must remain constant for each boxcar, the padding zeros used at the end of the boxcar, which exist even when the maximum number of stubs are sent, and finally the Bx ID, which although does not remain constant between boxcars, increments by a fixed amount. By finding a point in the data stream whereby the information 64 frames previous matches with these markers, the start of the header and thus packet can then be reliably determined. Once found, determining the start of following boxcars is trivial, as it will simply be 64 frames after the previous and thus can be implemented through the use of a counter in the firmware.

Each of the different module types similarly has a different fingerprint against which the header must be identified. As such, the header identification block within a given DTC link is unique to the module type to which it connects. Nevertheless, all headers can be identified by the three characteristics detailed above. The bespoke nature of this algorithm is reflected in the firmware design of the DTC, whereby the header identifier is specific to a DTC, compared to the largely common aspects of the remaining blocks in the design.

4.1.4.2 Stub Extraction

Each of the header data fields are extracted from the data stream through the use of a fixed pattern, corresponding to the fixed nature of the header. Conversely, as the number of the stubs in a boxcar is not constant, the extraction of the stubs is done in a more dynamic fashion. Each word which contains stub data is concatenated together as they arrive until the resulting bit vector is large enough to contain a full stub, at which point that stub is read out and the bit vector is shortened by the equivalent size, leaving only the bits for the next stub in the vector. This process is repeated until the number of extracted stubs equals the value of the stub multiplicity field within the boxcar header. The break condition is necessary to avoid extracting “ghost” stubs from the boxcar which would be made by misinterpreting padding zeros as valid data.

4.1.4.3 Stub Interleaving

As a result of the architecture of the routing stage of the DTC, a final processing block is required within the decoding. The router expects a single stream of stub data per input link from a module, however boxcars from each CIC on the module are decoded independently. Therefore it is necessary to combine the two stub streams from each CIC into a single stream to be sent to the router. In order for the firmware to have sufficient bandwidth to transport all stubs for both CICs without truncation, the FPGA for the DTC must run at 360 MHz, equating to 72 clock cycles per 8Bx boxcar period. This number of clock cycles is equal to twice the maximum number of stubs defined to be sent by the 10G PS module (see table 2.1), the highest bandwidth configuration, meaning that there will be no truncation with a stub produced on each clock cycle for any module configuration.

In order to achieve the interleaving, a relatively simple algorithm is implemented. In the case when only one of the two decoders sends a stub on a given clock cycle, that stub is simply immediately sent onwards to the output data stream. In the case where both decoders send stubs on the same clock cycle, the stub from the second decoder is cached, whilst the other is sent to the output. In the condition whereby neither decoder sends a stub on a given clock cycle, cached stubs are then sent to the output. Through software simulation of the scenario where the maximum number of stubs are sent from both CICs, it was shown that having a cache depth of three was sufficient to avoid any truncation within the interleaving stage (see figure 4.7).

4.1.5 Stub Processing

Stub processing consists of two sets of firmware blocks described in figure 4.1: the stub formatter and router. Here the two primary functions of the DTC for the trigger path are respectively performed: conversion to a global coordinate system and routing to the correct output link through the use of the stub's Bx information.

4.1.5.1 Stub Formatting

Motivated by the need to minimise the bandwidth leaving the detector, whilst maintaining maximal trigger throughput, the stub data format within a boxcar contains position information local only to the individual module from which it is sent. In order to perform track finding however, a global coordinate, at least global within a given nonant, is required.

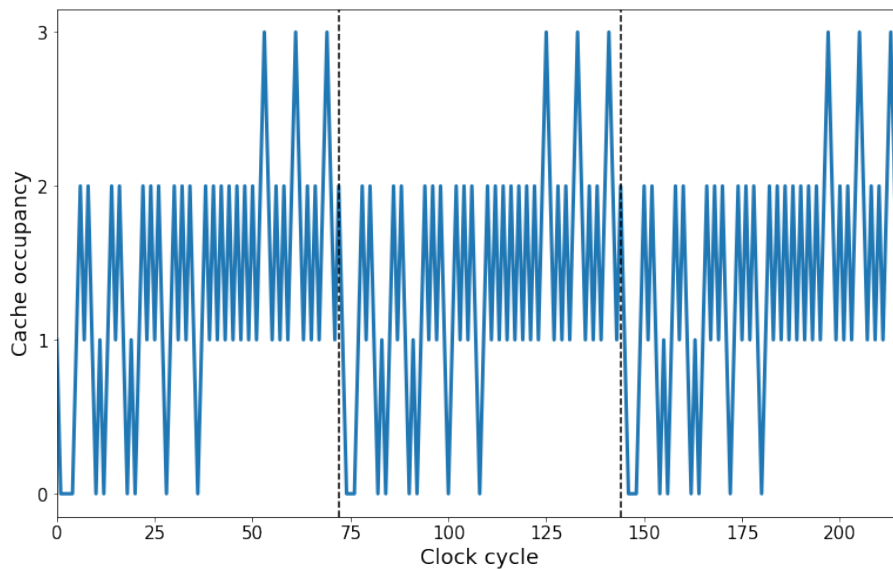


Figure 4.7: Plot of cache occupancy for three boxcar periods at 360 MHz.

The conversion from local to global coordinates is achieved by using the index of the link from which a stub arrived as the address of a lookup table containing the positions of the centres of each module connected to the specific DTC. The precise location of the hit can then be determined by adding on the local position of the hit to this centre coordinate.

When converting from the Cartesian coordinate local to the module to a cylindrical coordinate system for the full tracker, a non-linear correction must be applied to any hit located away from the centre of the module. The correction becomes even more significant when the modules are not arranged normal to transverse rays emanating from the beam line. In the Phase-I and early iterations of the Phase-II tracker, barrel modules were tilted in the axis parallel to the beamline, defined here as θ . Modules at higher η between the barrel and endcaps are also tilted in the orthogonally to the beamline, defined as β . For a two-dimensional example, for instance a barrel module where $\beta = 0$, shown in figure 4.8 it is clear that there is a significant difference in ϕ and in the true radius of the hit.

In two dimensions, for instance a transverse slice of a barrel module, the true radius and the difference in angle between the centre and hit can be calculated analytically: the angle, ϕ_{corr} is given by

$$\begin{aligned} \tan \phi_{corr} &= \frac{x \cos \theta}{r_{centre} + x \sin \theta} \\ \phi_{corr} &= \tan^{-1} \left(\frac{x \cos \theta}{r_{centre} + x \sin \theta} \right) \end{aligned} \quad (4.1)$$

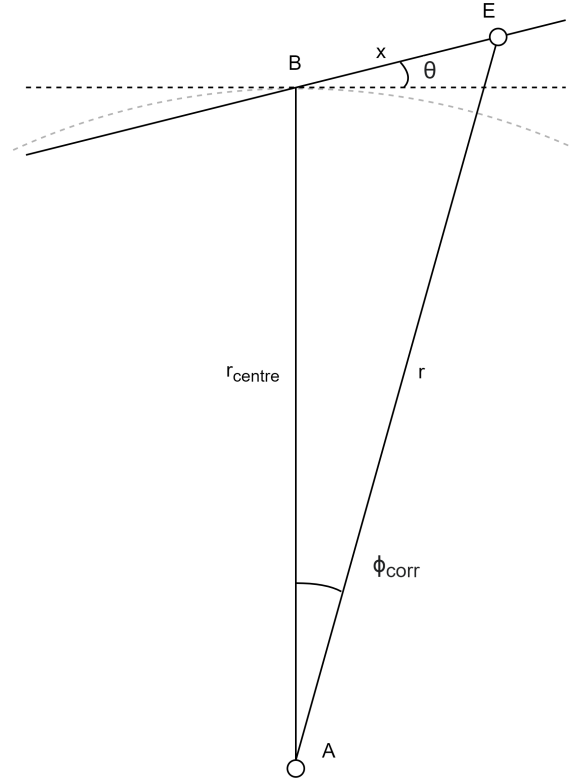


Figure 4.8: 2D diagram showing the position of a hit (labelled as E) away from the centre of a tilted module by an angle of θ module and by a displacement x . In this example, x would be positive. r_{centre} denotes the radius given by the LUT and r , the true radius of the hit. ϕ_{corr} denotes the correction required for the ϕ measurement.

where x is the displacement along the module from the centre to the hit. It then follows that the true radius can be given by

$$r \sin \phi_{corr} = x \cos \theta \quad (4.2)$$

$$r = \frac{x \cos \theta}{\sin \phi_{corr}} \quad (4.3)$$

Clearly neither of these corrections are linear, even at relatively small displacements, and therefore would be very difficult to implement in an FPGA, beyond a large lookup table containing results for every point in every module. The requirement for linearisation would be to have a correction that is in the form (in two dimensions):

$$\begin{pmatrix} r \\ \phi \end{pmatrix} = \begin{pmatrix} r_{centre} \\ \phi_{centre} \end{pmatrix} + x \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \quad (4.4)$$

where α_1 and α_2 are constants. In the limit where $x \ll r_{centre}$, α_2 can be reduced to

$$\alpha_2 = \frac{\cos \theta}{r_{centre}} \quad (4.5)$$

which is constant for a given module. The radius however, is far more difficult to linearise. If instead the assumption is made that $\phi_{corr} \ll 1$, r can also be written as

$$r = r_{centre} + x \sin \theta \quad (4.6)$$

hence α_1 is simply

$$\alpha_1 = \sin \theta \quad (4.7)$$

By combining these corrections, equation 4.4 can now be written as

$$\begin{pmatrix} r \\ \phi \end{pmatrix} = \begin{pmatrix} r_{centre} \\ \phi_{centre} \end{pmatrix} + x \begin{pmatrix} \sin \theta \\ \cos \theta / r_{centre} \end{pmatrix} \quad (4.8)$$

Equation 4.8 can be extended to three dimensions, for a module rotated in both axes. The coordinate correction is then given by the equation

$$\begin{pmatrix} r \\ \phi \\ z \end{pmatrix} = \begin{pmatrix} r_{centre} \\ \phi_{centre} \\ z_{centre} \end{pmatrix} + \begin{pmatrix} \sin \theta \cos \beta & \sin \beta \\ \cos \theta / r_{centre} & 0 \\ -\sin \theta \sin \beta & \cos \beta \end{pmatrix} \begin{pmatrix} strip \# \times pitch_{strip} \\ pixel \# \times pitch_{pixel} \end{pmatrix} \quad (4.9)$$

where *strip #* is analogous to x and *pixel #* is the equivalent displacement in the z axis and scaled by the respective pitch in each dimension of the sensor. Considering a “worst case” theoretical module, whereby it is close to the beam line and is rotated in both axes, the exact, linearised and relative error are shown in figure 4.9. The greatest error is seen in the the correction for ϕ , with a relative error of approximately 10% for the farthest corner of the module.

According the most current tracker geometry, as described in [54], all modules in the outer tracker have $\theta = 0$. In this case, all modules are normal to the beam line in the transverse orientation, thus the correction is greatly simplified. The correction matrix

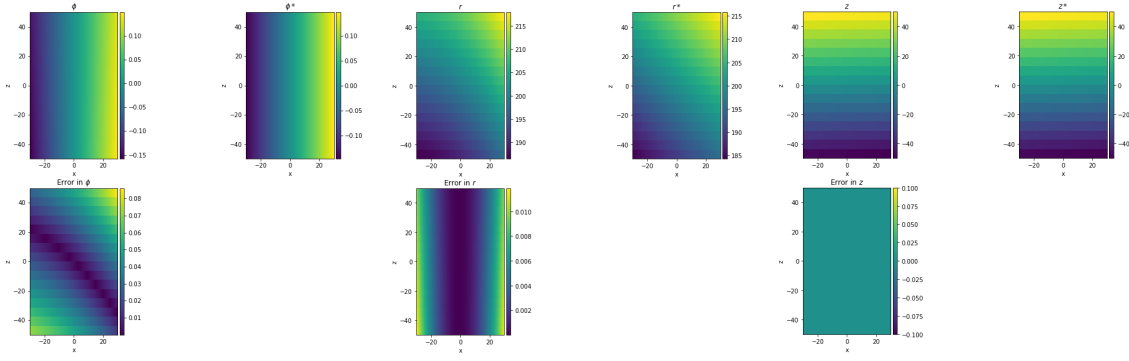


Figure 4.9: Plots showing the exact correction, approximate correction and relative error in r , ϕ and z . Coordinates were calculated for a module 30 mm wide, 50 mm long at a radius of 200 mm and a tilt of $\theta, \beta = 0.2$.

becomes

$$\begin{pmatrix} 0 & \sin \beta \\ 1/r_{centre} & 0 \\ 0 & \cos \beta \end{pmatrix} \quad (4.10)$$

with only three non-zero entries. Considering this simplicity, it may be possible to include higher order corrections to some coordinates. When $\theta = 0$, Pythagoras' theorem may be used to calculate the true radius

$$\begin{aligned} r^2 &= x^2 + (r_{centre} + z \sin \beta)^2 \\ r &= \sqrt{x^2 + (r_{centre} + z \sin \beta)^2} \end{aligned} \quad (4.11)$$

This function for r is even with respect to x , hence the first non-constant term will be $\mathcal{O}(x^2)$ which is not linear. For ϕ , if again the small angle assumption is made, the correction may be written as

$$\begin{aligned} \phi_{corr} &= \tan^{-1} \left(\frac{x}{r_{centre} + z \sin \beta} \right) \\ &\approx \frac{x}{r_{centre} + z \sin \beta} \\ &\approx x \left[\frac{1}{r_{centre}} - \frac{z \sin \beta}{r_{centre}^2} \right] \end{aligned} \quad (4.12)$$

which now includes an extra corrective term which reduces the error in ϕ , especially in the endcap, where $\beta = \pi/2$. As for z , the correction is linear and exact, hence no approximations need to be made.

As described in section 2.1.1 however, each of these modules is segmented into eight rows, each controlled by a single front end ASIC. As the module itself is flat, the outer segments of the module will no longer be normal to the beam line and instead be at an angle. By using a LUT to get the centre of each of these segments, then applying a correction across a smaller area, the error on the correction will be reduced. This does however, come at the expense of an increase by a factor of 8 in the address space for the LUT.

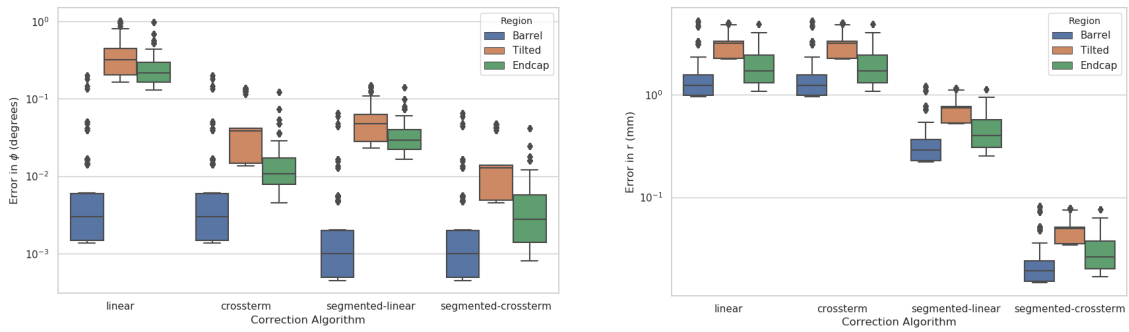


Figure 4.10: Plots to show the absolute accuracy of different algorithms. “Segmented” algorithms split each module by eight in the ϕ direction. The “crossterm” algorithms include an extra corrective term in ϕ and also a correction of tilt of θ for segments away from the centre of the module.

Figure 4.10 shows box plots for the four approaches explored for the coordinate correction. The data was further split into barrel, tilted and endcap modules, as these were distinct categories in the geometry. The inclusion of a segmented modules gave the largest reduction in the error, however the inclusion of the cross term in ϕ and the tilted correction in r also improved the accuracy of the correction significantly. These extra corrections come at the expense of increased resource usage within the FPGA. As shown in figure 4.11, the size of the BRAM LUT increases by eight between the segmented and non-segmented algorithms, in proportion to the increase in granularity offered. The cross term and tilted module correction both require an extra DSP per incoming link. The BRAM utilisation required for these lookups is orders of magnitude less than what would be required to perform an exact lookup for each point on each module. The increase in BRAM usage by a factor of approximately 6,000 for an exact lookup would not be feasible for an FPGA and would likely also increase the latency in traversing such a large table.

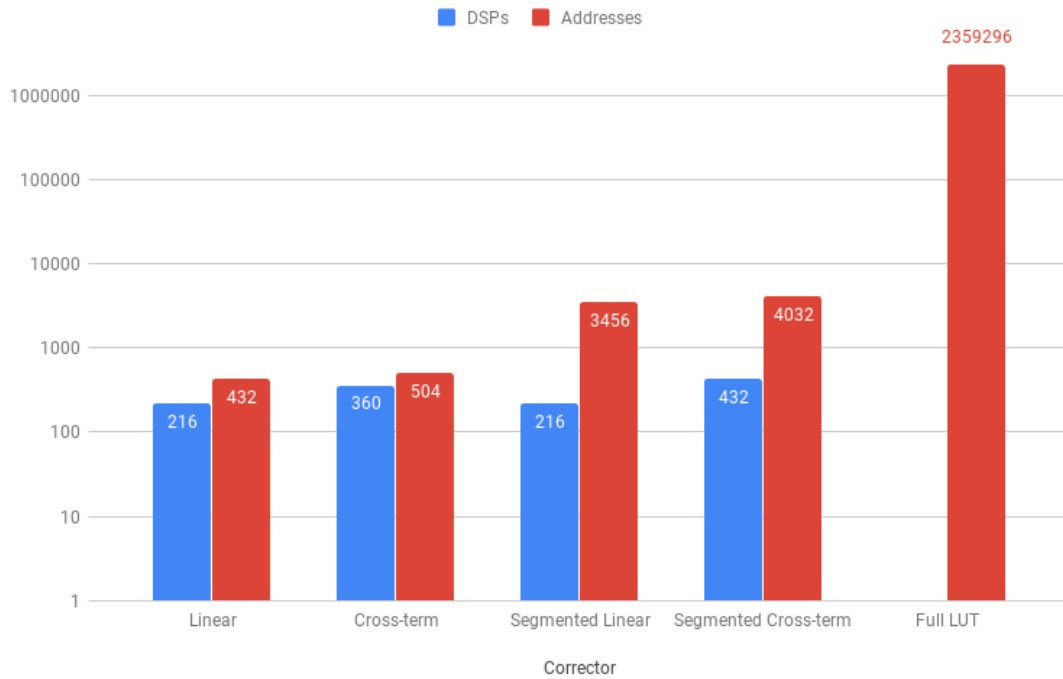


Figure 4.11: Plot showing the approximate resource usage for the different correction algorithms explored. Note the logarithmic scale on the y-axis. For comparison, an approach using only LUTs is also included.

4.1.5.2 Routing

The second aspect of the DTC trigger path is a necessity that arises due to the architecture of the track trigger system itself. By having multiple cards each processing individual events in parallel, stubs must be converted from being spatially divided, by the nature of multiple input links, to being divided by their bunch crossing. In order to do such a conversion, a block of logic was developed to reside in the DTC FPGA.

The block can be best described as a systolic array of First In First Out (FIFO)s. A systolic array is defined as a two-dimensional lattice of interconnected Processing Unit (PU)s, each of which perform a simple task. In the case of the router, these PUs take data from both the neighbouring PU in which it shares a row as well as the neighbouring PU in the same column. It then sends data to the next PU in the row as well as the following PU in the column. In order to most optimally use the physical layout of the FPGA and its two banks of input links, the router contains two arrays, each with 36 rows and 18 columns (figure 4.12). The input to the array comes from the rows, each of which is fed data from a single module. The output of the array is then the bottom of each column,

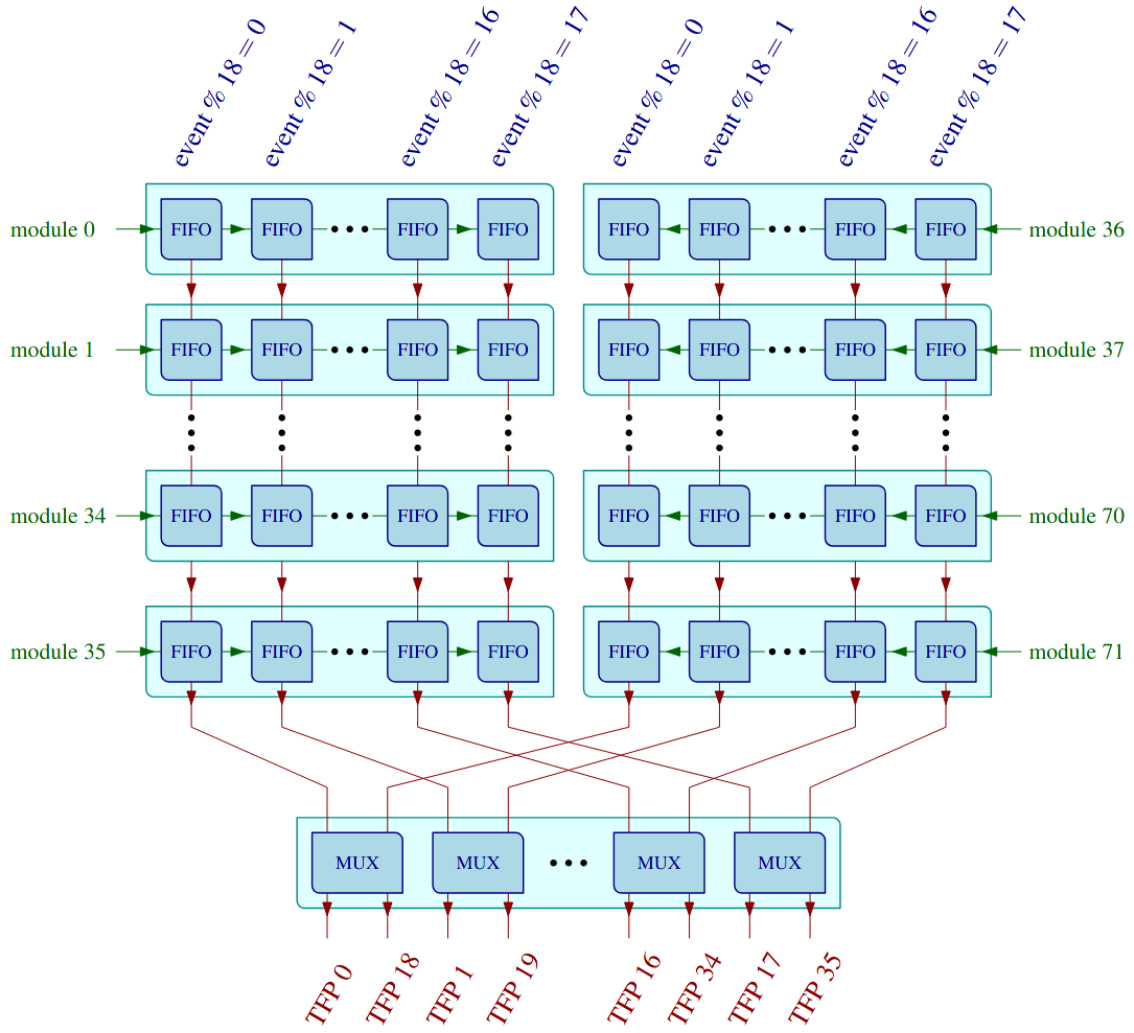


Figure 4.12: Diagram showing the systolic array architecture of the router block.

corresponding to a given bunch crossing within the time-multiplexing period of 18 bunch crossings. On each clock cycle, a PU can receive a stub from both the neighbouring row PU and column PU. These stubs are stored within a FIFO within the PU. If the stub has a bunch crossing ID which matches that of the column which the PU belongs to, the stub is then sent downwards to the next PU in the column, else it is sent to the next PU in the same row. The FIFO within each PU ensures that each PU can handle multiple input stubs on each clock cycle without truncation and the FIFO will empty during gaps in the input data stream.

4.2 DAQ Path

Alongside the trigger path, the DTC also serves a role in the transport of events accepted by the Level 1 (L1) trigger to the HLT. In the case of an L1A, this signal is propagated to all DTCs, which in turn send this signal to the connected modules. On receipt of the signal, the module then sends all hit data from the bunch crossing, irrespective of the bend, over a dedicated L1A path back to the DTC. From the CIC to the lpGBT, one e-link is dedicated to this L1A upstream datapath. The DTC itself connects to a DAQ and TCDS Hub (DTH)[55] card located within the same crate as the DTC, which finally transports the event data up to the HLT farm located at ground level at Point 5. Each DTC card has four 25 Gbps output optical links to the DTH, thus the data from the modules must be collated into a large buffer to then be sent out on these links. Given the significantly reduced bandwidth of this path compared to the trigger path, the logic allocated within the FPGA for the path is far smaller.

Chapter 5

DTC Testing

Alongside the development of any firmware or software project, the necessity to test is paramount. Similarly the testing was structured such that smaller units of the system were tested individually before integrating multiple units together to form larger tests. Testing of the DTC was performed in two stages, driven by the availability of hardware at the time. This began with the Stub Processing block which lies only within the payload of the FPGA, before the arrival of prototype FE modules provided scope to test the Link Interface firmware block and optical link between module and BE processing card. The tests presented in this chapter follow the same order.

5.1 Stub Processing

As it is possible to develop and test the Stub Formatter and Stub Router, collectively known as the Stub Processing block, without the need to connect genuine FE hardware, these were the first to be written. Test data could be loaded into the input EMP buffers, played through the algorithm and then captured on the output buffers. This output would then be compared with an expected output to check for errors. The inputs and expected outputs for the Stub Processing block were generated by a software emulation of the firmware, written in C++, as part of CMS SoftWare (CMSSW)[56]. The input used was a set of events containing the decay of a top and anti-top pair along with 200 soft QCD “pileup” interactions. Given the high multiplicity of the decay, this event type is considered a “worst-case scenario” for the tracking system, thus was used as a benchmark for testing all aspects of the chain. A third point of comparison can be generated by simulating the firmware within the EMP framework, which has the functionality to write

to and read the buffers in the same manner as on the FPGA itself.

At the time of the tests, no hardware was available with the target FPGA for the DTC, the VU13P. Instead, the Serenity version 1.1 prototype used for the DTC tests was fitted with a KU15P daughtercard, which is approximately half the size of a VU13P. As such, the full algorithm developed for the Stub processing, in particular the routing block, had to be downsized in order for it to be contained within the smaller FPGA. The so-called “half” DTC had half the input and output links of the full, with stubs being routed to just one of the track-finding nonants, but for all 18 Bx. As a single copy of the stub formatter is required for each input link, this aspect of the design scaled easily. The router also contains two systolic arrays, each processing one half of the full 72 links, with the outputs multiplexed together in the final step. As such, in the half implementation, only one of these arrays is instantiated, with the multiplexing step not required. Stubs that would be routed to the track-finding nonant whose links are not connected to the DTC had their stubs discarded at this final stage.

Despite these changes, the half implementation still gives a representative example of the DTC firmware, and its demonstration in hardware was a primary objective of the DPS working group within CMS. As a result of limitations in the software emulation of the DTC, only the full implementation could be simulated in this way, whilst both the half and full designs could be simulated in firmware. As such, software and firmware were compared using the full implementation, whilst firmware simulation and the results of running on hardware were compared using the half design. There were however, drawbacks to the testing approach. In particular any issues that may arise from implementing the larger design on an FPGA, such as convergence of timing and resource usage could not be determined. Similarly the functionality of the multiplexing stage which was omitted in the smaller design could not be verified in hardware.

The results of these two comparisons can be seen in figures 5.1a and 5.1b. It can be seen that the two comparisons of the firmware design match identically, showing that the design loaded onto the FPGA matches that declared by the simulated VHDL. The comparison between software and firmware is broadly the same, however there are a few small inconsistencies between the two. This comes as a result of different truncation methods being used in the emulation, meaning that some events with high multiplicity behave differently between the two.

Overall, the stub processing block was shown to operate well, and should not present

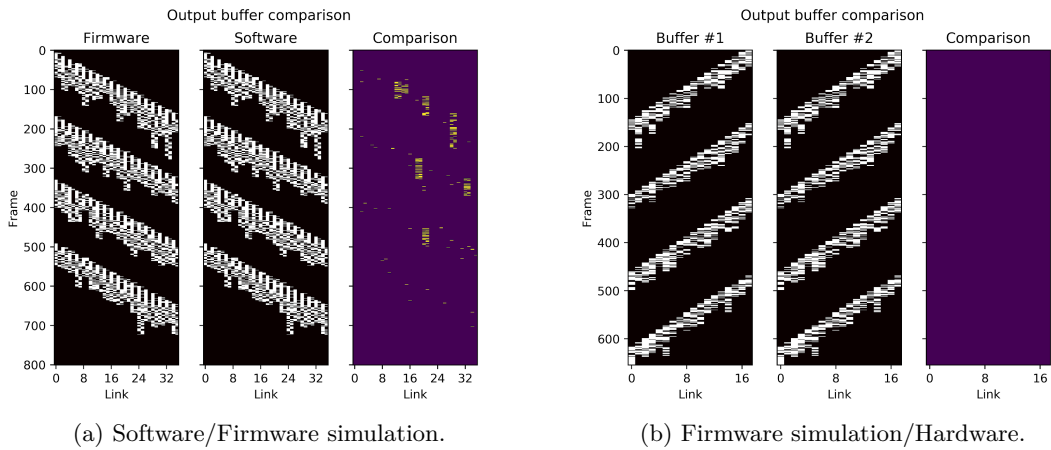


Figure 5.1: Plots the value of each word in a 1024 frame capture of the EMP buffers, for the software emulation, firmware simulation and on the FPGA itself. The third subplot of each figure shows the difference between the other two plots. In plot 5.1b, the left hand plot is firmware simulation, whilst the middle is taken from the FPGA itself and the right is again the comparison.

a bottleneck to the use of tracking in the L1T.

5.2 Front End Testing

With the stub processing block demonstrated, focus shifted from its development to that of the link between the DTC and the FE modules to which it will be connected in the final system. This development came at the same time as the first prototype “skeletons” of the FE modules began to arrive at CERN for testing. The skeletons provided had all of the FE ASICs installed on the module, however had no silicon sensor. As such, they could not be used to record genuine stubs, however it was possible to create “synthetic” stubs within the CBCs to simulate stubs being generated. Another difference was that the early skeletons used the GBT, rather than lpGBT for communication. Whilst similar, the two transceivers required different blocks in the BE FPGA, which were necessary to develop before testing could begin. This gave the opportunity to test with real hardware, which was necessary when testing firmware which lies outside of the scope of the EMP payload.

The aim of early tests with the skeleton was to verify that it was possible to communicate between the Serenity processing card and module. In particular, to receive data on the Serenity that had been generated within the CBCs. A further challenge in the early tests was that no software existed to configure the FE modules via a Serenity card. Software existed to do such a task for the FC7[57] card, but this had not been ported over to Serenity at the time. This meant that it was necessary to first connect both the Rx and Tx fibres to an FC7 in order to configure the module, before reconnecting the Rx fibre to the Serenity card so that it could receive data. The module can only operate when it receives an external clock provided over the optical link, therefore it was necessary to not break the Tx fibre connection to the module from the FC7, so that this was not lost. This meant that the module had a fibre connected to both the FC7 and Serenity during data taking. Whilst necessary in order to receive any kind of data in the Serenity card, this setup also led to a number of issues during testing which are discussed below. Nevertheless, the tests were successful and it was possible to receive stub data from the module in a Serenity card.

5.2.1 Stub Histograms

Using the EMP framework in both simulation and on the FPGA itself, it is possible to capture 1024 frames of data on both the Rx and Tx links. These were very useful in the initial stage of development, allowing for deep inspection of the data flow within the

payload. The sample size, however is relatively small, meaning that bugs in the firmware that do not present themselves on each boxcar may be missed. In order to capture data over longer timespans, a histogramming module was added to the payload. The input to this histogram was the address of the processed stub, meaning a given bin value was incremented by one each time that a stub was sent to it. An EMP capture, showing the output of such a histogram, along with an example plot of extracted stubs can be seen in figure 5.2.

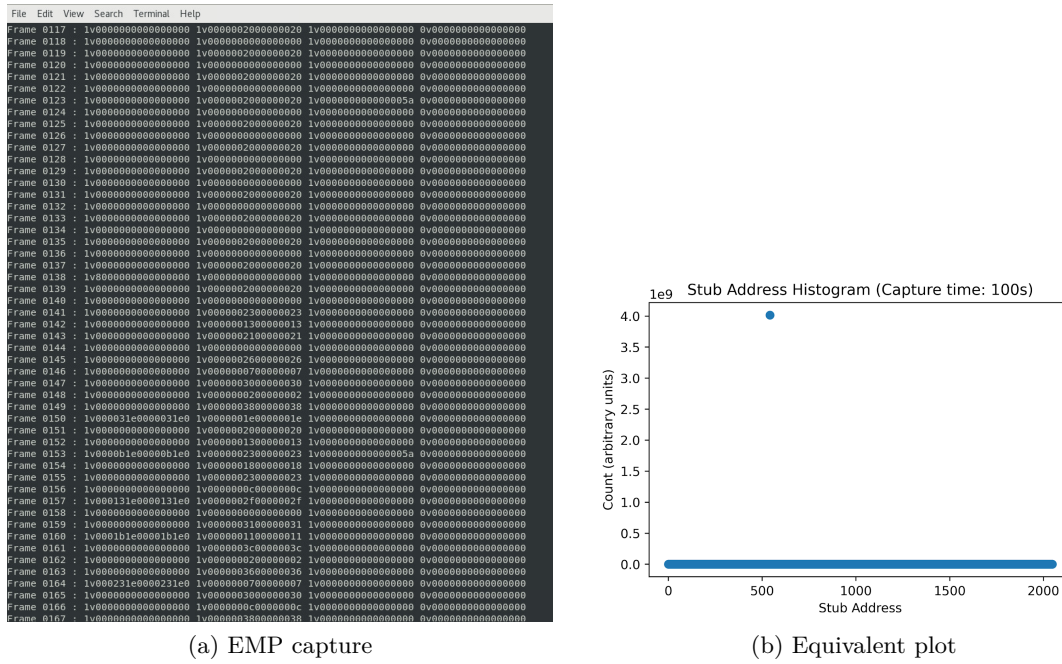


Figure 5.2: EMP capture showing both the extracted stubs (first column) and histogram output (third column). This histogram was taken during the early stages of development of the stub extractor, at a point where the output of the extractor did not check for “ghost” stubs (see section 4.1.4.2). This is the cause of the two peaks in the histogram, one at address zero and the other at the address of the real stubs. Right hand image shows how a similar capture would look when plotted as a histogram.

When testing for errors in the link between the FE module and Serenity, or within the the FE module itself, it is important to ensure that the firmware for the histogram does not introduce any errors itself. The underlying process for a histogram is as follows:

1. Read the value corresponding to the input address
2. Add one to this value.
3. Write the updated value to the same address

In a sequential processing paradigm, implementing the algorithm described above

seems trivial, however it becomes far more complex when designing firmware for an FPGA. In such a design a new address arrives on every clock cycle in a continuous stream, however the process shown above requires more than one clock cycle to complete. Simply reading from a Random Access Memory (RAM) address requires more than one clock cycle for instance. The dual-port design of RAM in Xilinx FPGAs allows for such a process to be possible. A dual-port RAM means that on a given clock cycle, the same block of RAM can be accessed twice, independently, one port being used to read the value at a given address, whilst the other writes an updated value to another independent address.

The dual-port nature of the RAM enables the histogram design, however does not implicitly protect the algorithm from errors, particularly if the same address is attempted to be updated on consecutive clock cycles. In such a case, the second read of the RAM would be made before the value at that address has been updated and thus data lost. As the whole algorithm in fact takes more than two clock cycles, there are a number of possible cases whereby data could be lost.

The first case of the same address being attempted to be accessed on consecutive clock cycles, is mitigated by changing the increment for the value of the bin to two instead of one. This means that the second write will have the correct value. The second condition is caused when the read address collides with the address of the bin being processed. For this case, the read value is taken from the value being processed, rather than reading from the RAM at all, thus ensuring that it is accurate. Finally, if a read and write are attempted on the same address on the same clock cycle, a similar mitigation strategy is used to the second case, whereby the write value is sent immediately to the read value, rather than to the RAM.

In order to test for each of these conditions, as well as combinations of all of them, a test pattern was generated that included each of the cases. Using a simulation of the firmware, it was then possible to view the exact values of each bin for every clock cycle, meaning that any inaccuracies could easily be flagged. In hardware, such inspection is not possible, so instead the totals of a bin was checked after the pattern has been processed and then compared with the count of the number occurrences of that address in the pattern.

With both ports in use during operation, it not is possible to also readout the contents of the histogram. Instead, operation must be paused so that a readout and reset can be performed. During this time, each address in the RAM is sequentially read and then zero is written to the same address. These operations leverage the both ports of the RAM in

order to reduce the time for a readout to complete. In order to then record the contents of the histogram in software, bin values are sent into an identically sized IPbus RAM, where one port operates at the payload clock frequency to input the data and the other at the IPbus clock in order to send the data to the SoM on the Serenity card.

Two different methods were developed to readout and reset the counts within the histogram. The first, and most simple was to implement a counter in the firmware that counted up to a configurable value, before sending a signal to the histogram to read out to IPBus buffers. The second was a threshold trigger, whereby the maximum bin value across the histogram was tracked and once it reached a similarly configurable value, the trigger was sent to read out. The first was useful for situations whereby the period of time over which the histogram was running was required to be known and thus provide an indication of rate, whilst the second allowed for longer runs without the concern of overflow in the histogram RAM blocks.

5.2.2 Issues in the Data

During initial testing, a significant level of bit errors were observed in the data captures from the modules. In particular, certain indices within the data word showed “noise” (see figure 5.3) on top of the expected output. In particular the errors manifested as extra “ones” in the bit stream, instead of bit-shifts. This can be seen in figure 5.4, whereby the average value of each index is plotted. For the indices whereby errors are seen, the average value is higher than expected, suggesting that the data being sent at that specific index is not as intended. The bit indices at which this noise was seen are limited to a very specific set. It can be found either at two pairs of indices located at 38,39 and 51,52 or 78,79 and 93,94. All other bits in the frame are as expected.

It is possible that these erroneous bits originated in the CIC itself, as for instance, the pair 38,39 correspond to bits found within the `stub0` e-link for `CIC0`. The pair 51,52 however correspond to an e-link which is not connected at the CIC and therefore should always be zero. In good captures of data, this was always the case. These errors were also seen when receiving stub data using alternative DAQ hardware, such as the FC7.

In order to identify and isolate the cause of the errors, each part of the data transmission chain was tested. To eliminate the chance of the errors being caused by a faulty module, multiple prototypes were tested and in all cases the same noise was seen. A simplified firmware design, whereby the the 112-bit widebus word sent from the module was

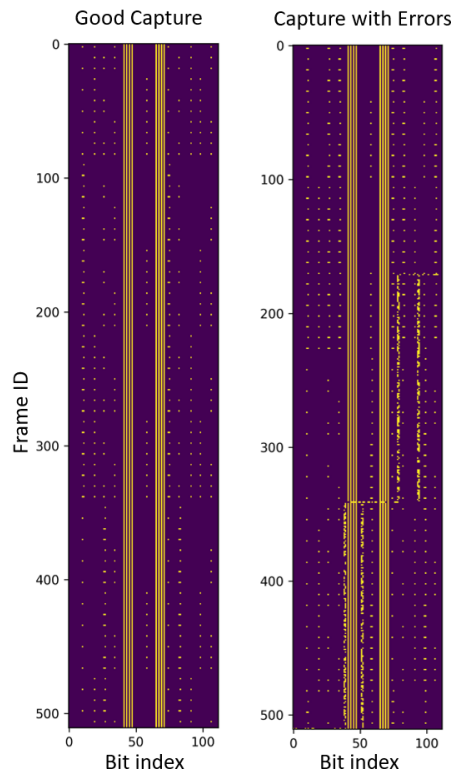


Figure 5.3: Plot showing two temporal captures of data from the GBT module. The left-hand plot shows expected behaviour when the CIC is sending only headers and no stubs. The right-hand plot shows the same CIC configuration, but errors can now be seen.

split into two consecutive 64-bit words which were then captured by EMP to be analysed. By doing so, any corruption caused by the downstream firmware blocks could similarly be eliminated.

The possibility of the errors arising as a result of clock dephasing, caused by the module and Serenity card being driven by independent clocks, was tested by coupling the clocks of both pieces of electronics. This was achieved by using the FC7 used to configure and provide a clock to the module to also deliver a clock signal to the Serenity card. A pair of SMA cables connected the FC7 to a clock breakout board which converted the 40 MHz LHC clock to the 160 Mhz required for Serenity. Despite this connected setup, errors were still observed when capturing data. Similarly, the nature of the errors not being a bit shift would suggest that a simple clock misalignment was not the cause.

As described above, the setup used when observing the errors was not how the module and Serenity were intended to be used, but a necessary compromise whilst software and firmware were developed to enable the use of the Tx link from Serenity to the module. Upon the completion of this work and the ability to communicate in both directions from

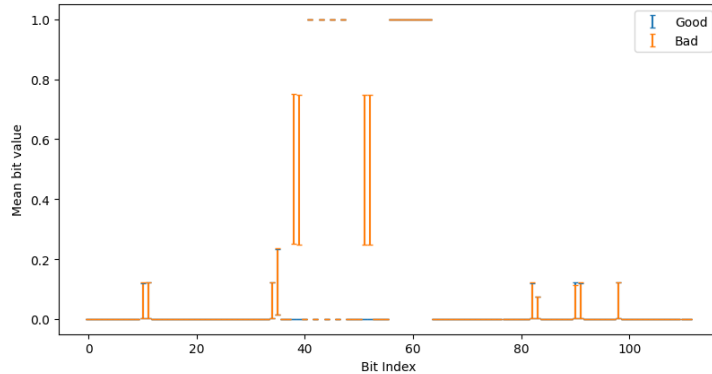


Figure 5.4: Plot showing the mean bit values for each index within a GBT frame. The data for good and bad captures overlap at all indices but for those which contain errors.

the Serenity enabled, the errors observed here were no longer seen. It is most likely therefore that the bit errors were as a result of a small frequency mismatch between the clocks on the Serenity card and the FC7 driving the module, however this asynchronicity manifested itself with the transceiver in a far more complex manner than a simple bit shift over time.

The data allows the possibility that these errors were caused by a timing issue at the e-link level to be ruled out. Such an issue would result in a bit shift within the 8 bits associated with a e-link, meaning that adjacent indices would be non-zero. The errors at 38, 39 correspond to the same e-link which is sending non-zero data, however mean value of the capture with errors is far higher than without. This means that extra ones are being added to the stream, not simply the bits being shifted to a different index.

The errors seen in the data were eventually resolved once it was possible to use both the Tx and Rx links from the Serenity board to the Front End Hybrid (FEH). By doing so, the same clock used by Serenity to receive data was also sent over the Tx link to the FEH and thus was guaranteed to be continuously synchronous and removed the need to swap fibres during the configuration of the module. It is likely therefore that the cause of the issues in the data was a result of a small difference in the clock frequency between the Serenity and FEH, which then lead to transient errors that arose with a frequency of the difference between the two clock and thus could only be seen over macroscopic time frames.

5.3 GBT Full Module Tests

5.3.1 Initial Tests

With the arrival of a full module and sensor at CERN, it was first necessary to repeat some of the tests that had been conducted with the skeleton, to ensure that the optical connection between the module and Serenity board worked, as well as the functionality of the ASICs on the module.

As this module contained some of the earliest components produced for the FE modules, the protocol for the optical connection was subtly different compared to that of the previous skeleton: the word size for the Rx direction at the Serenity was the same, however the Tx word size was smaller, using the 80-bit `GBT_FRAME` compared to the now standard 112-bit `WIDE_BUS`. This meant that the firmware used for testing and development with the skeleton had to be modified and rebuilt before it could work with the module itself. The modular nature of the firmware however meant that the changes required were small and easily tracked.

With these changes in place, the first test was to generate stubs synthetically within the CBCs and then record these stubs in a histogram in the BE board.

5.3.2 HV Threshold Tests

To generate real stubs, a High Voltage (HV) must be applied to deplete the silicon sensor and carry the electron-hole pairs generated to the readout chips. In order to achieve this depletion, the polarity required for the sensor must first be determined. This was achieved by applying a small voltage across the sensor in both polarities and observing the current produced. In the case of the incorrect polarity, the resistance of the sensor is very low, thus a high current generated. To mitigate any damage that could have been incurred in this scenario, a limit was placed on the current within the Power Supply Unit (PSU). Once the polarity was correct, the voltage across the sensor was then incrementally increased, each time observing the current produced. A higher voltage has the benefit of both ensuring that the sensor is full depleted as well as reducing the response time of the sensor. Initial tests were run at -100 V , whilst the majority of the data was taken with the sensor at -300 V .

With the HV supply active, ambient light from the laboratory is sufficient to induce a current in the silicon that saturates the supply, as such, the entire module had to be

covered with a box to ensure darkened conditions before the HV supply could be switched on.

The next task was to calibrate and characterise the module using the Phase-II Acquisition & Control Framework (Ph2ACF) collection of software and firmware. The procedure involved determining the “pedestal” for each strip, defined as the threshold voltage at which each strip has a 50% occupancy.

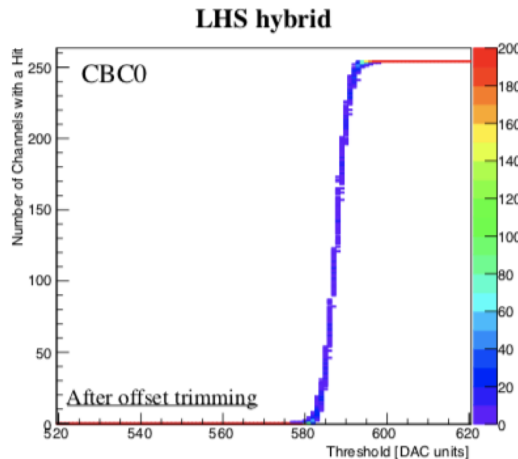


Figure 5.5: Example plot of occupancy against V_{cth} for all strips in a CBC.

A scan of strip occupancy against V_{cth} will yield data in the form of an “S-curve” (see figure 5.5). From this, the pedestal value can be extracted by the intersection of this curve with a line drawn at 50% occupancy. This calibration is run at a low level within each CBC, however an analogous measurement can be made at the DTC. By sweeping V_{cth} but this time recording the total number of stubs generated over a constant time period, a similarly shaped plot should arise. In the case of the module tests, the contributions of all strips within all CBCs were summed over a period of 13.4 s. This period equates to 2^{32} clock cycles of the 320 MHz payload clock.

Similar to the occupancy test, the data shows a significant rise in the number of recorded stubs as a function on V_{cth} . In order to maximise the signal to noise ratio, whilst also not reducing the sensitivity to genuine stubs, V_{cth} was selected to be 550 by default.

5.3.3 Histogramming with a Radioactive Source

Once a baseline noise level had been found, the next aim for the tests was to attempt to record stubs produced from genuine particles generated by a radioactive source. In the case of these tests, the source was a sample of Bismuth-207 (^{207}Bi), which has a primary

decay mode of electron capture, whereby a proton within the nucleus is converted to a neutron through combining with an electron. This process causes the nucleus to become excited and subsequently releases a high energy photon. A purely gamma source as this process suggest would not have been well suited for the module tests, however ^{207}Bi also undergoes a secondary process of internal conversion, which suppresses the emission of γ particles through the interaction of the electromagnetic fields between the nucleus and its orbital electrons[58]. This interaction instead leads to the emission of an electron within the lower orbital with a specific energy equal to the γ photon energy minus the binding energy of the electron. As both of these quantities are constant to the first order, the emission spectrum for electrons from the atom is a spike, rather than a continuous distribution.

For the tests with the module, having a source with well-defined electron energies was preferable to a conventional β source. In ^{207}Bi , internal conversion happens in approximately 11% of all decays[59], meaning that the stated activity for the source of 34.9 kBq must be multiplied by 0.11 in order to determine the electron activity of the source.

In order to protect the module from damage incurred during setup, the source was placed atop a perspex sheets, which was itself above the silicon sensor of the module. To record stubs, the module was power-cycled and reconfigured, then a count of stubs for each strip in the module was saved in the DTC over a period of 13.4s, before the counts were read out over IPBus to the SoM.

Before these counts could be plotted as a histogram, the first four CBCs had to be rearranged from their ordering within the CIC to the physical ordering on the module. Furthermore, a number of noisy strips were identified during data taking; these were similarly masked out of the final histograms.

5.3.3.1 Data Analysis

Initial tests focused on the behaviour of the histogram when the source was moved to different locations on the sensor. For instance, as shown in figure 5.6, placing the source over each of the two columns, and thus CICs, provided information on which physical set of strips corresponded to a given CIC.

By moving the source along the sensor orthogonally to the strips, as shown in figure 5.7, the physical location of the first strip could also be determined. This test also highlights well the spatial resolution of the sensor in this axis.

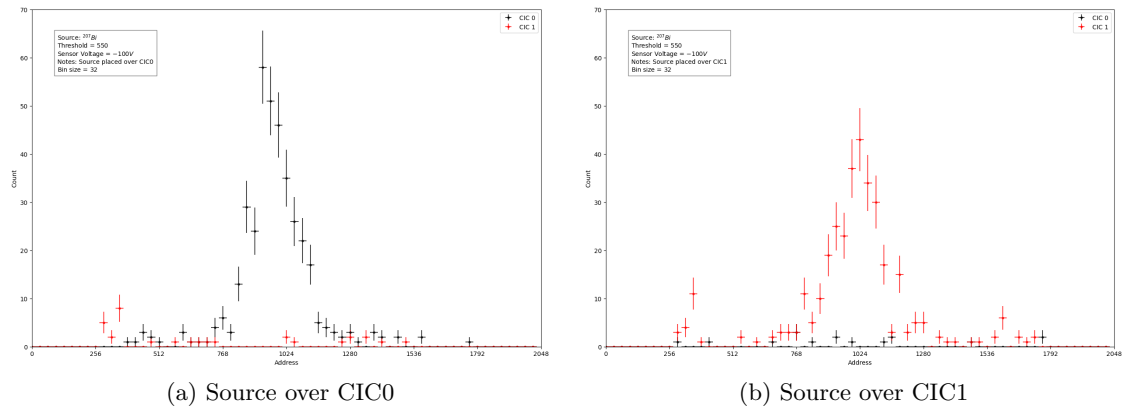


Figure 5.6: Plots showing histograms of the sensor strips when the radioactive source is placed over each in turn.

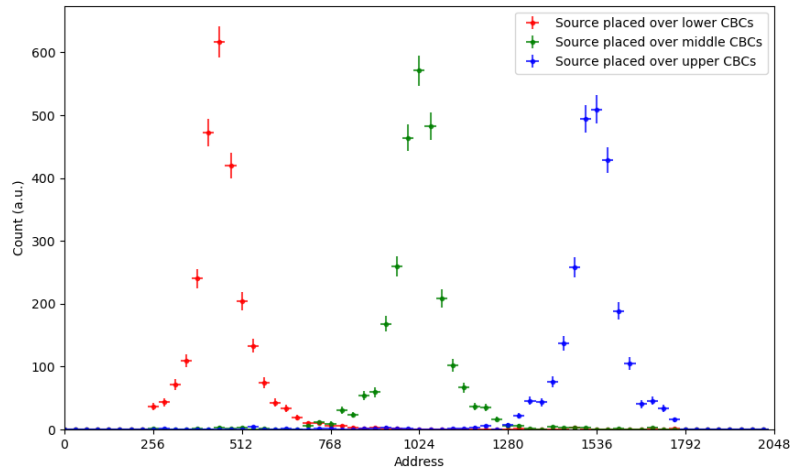


Figure 5.7: Plot showing three histograms, each taken with the source located at different positions on the sensor.

With the spatial mapping complete, testing progressed on to sweeps of different variables that could be changed and their behaviour observed. First was the voltage of the HV PSU, with histograms taken at -100 V, -150 V, -200 V, -300 V, -400 V. The results of this scan can be seen in figure 5.8. The histograms at -300 V and -400 V contain a similar number of hits, suggesting that the sensor has reached full depletion at these voltages. This is as expected for a sensor without radiation damage. Despite running for 2^{32} events at 40 MHz, equivalent to approximately 107 seconds, the statistics for the voltage sweep were limited. In an effort to improve the hit rate, runs were conducted with the perspex cover removed from above the sensor. So that the source was not directly in contact with the silicon sensor, a sheet of paper was used to separate them. As shown in figure 5.9, this gave an order of magnitude improvement in the number of hits recorded within the

capture period.

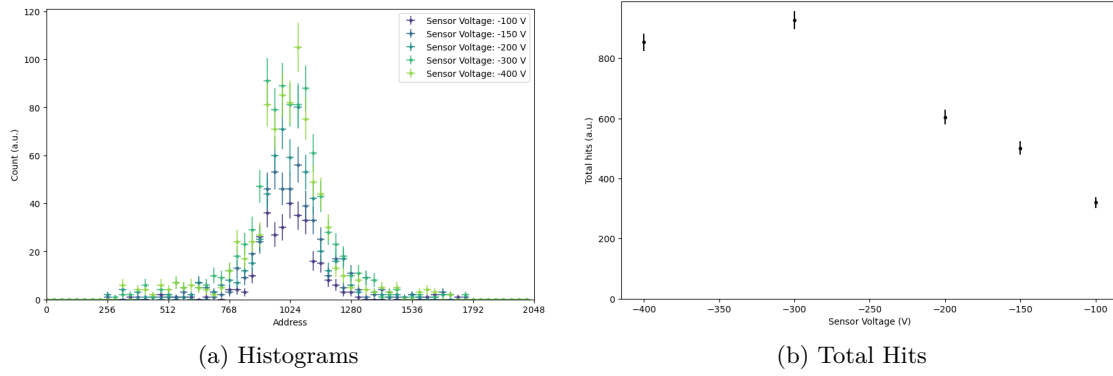


Figure 5.8: Plots showing the histograms recorded at each voltage value within the scanned range.

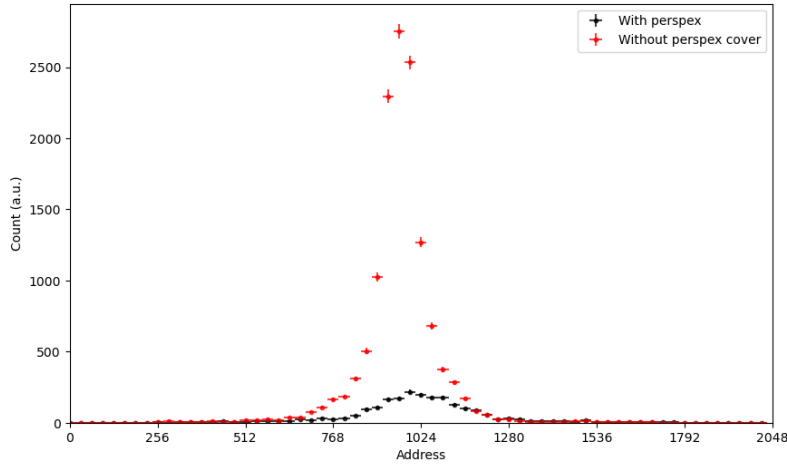


Figure 5.9: Plot the difference in hit rate for both with and without the perspex cover.

Within the settings for the CBCs, it is possible to define a p_T cut, whereby hits with a displacement of greater than this value between the top and bottom sensors are rejected. This value is measured in the number of strips in this displacement and can vary between zero strips, i.e. requiring both hits to be in the same strip index, to fourteen. A scan of histograms was taken, with records made at each value in the range and can be seen in figure 5.10. As seen in figure 5.10b, there is a strong linear relationship between the number of hits recorded in the histogram and the p_T cut value.

In order to estimate the efficiency of the sensor, first the expected distribution of particles from the source must be characterised. The simplest approximation would be to make the source isotropic, originating from a single point above the sensor. Furthermore, given that each half of the sensor has no z resolution, this dimension can be excluded.

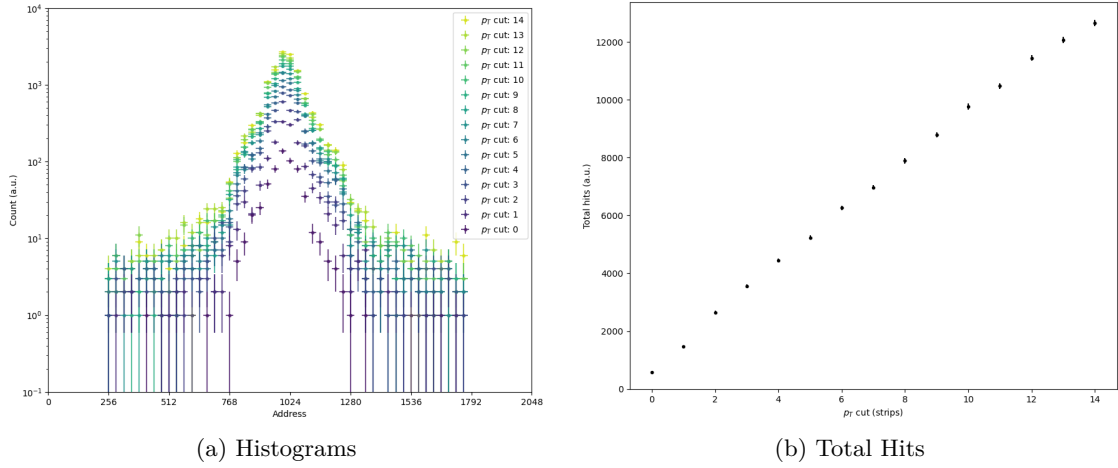


Figure 5.10: Plots showing the histograms taken with each p_T cut value. Note the log scale on graph 5.10a.

Thus, for a point source with activity A , the flux density, J_{point} , incident at a point on a plane, with the source a distance h above that plane, can be given by:

$$J_{point}(x) = \frac{A}{h^2 + (x - x_0)^2} \quad (5.1)$$

Where x_0 is the point on the plane closest to the source.



Figure 5.11: Photograph of ^{207}Bi source used for tests, placed above a scintillator and PhotoMultiplier Tube (PMT).

Instead of a single point, the source used for the tests was in fact a disc of active material (see figure 5.11). As such, in the single plane dimension considered, the source could be better described with a rectangular function[60], $\Pi(x/w)$, where w is the width of the source. As such a distributed source can be equated to the sum of many point sources, a convolution between the the formula for the flux density of the point source and

the rectangular function gives the flux density, J_{dist} for the distributed case:

$$J_{dist}(x) = (J_{point} * \Pi)(x) \quad (5.2)$$

where

$$(J_{point} * \Pi)(x) = \int_{-\infty}^{\infty} J_{point}(\tau) \Pi(x - \tau) d\tau \quad (5.3)$$

This produces the following formula:

$$J_{dist}(x) = \frac{A}{h} \left[\tan^{-1} \left(\frac{1}{h} \left(\frac{w}{2} - (x - x_0) \right) \right) + \tan^{-1} \left(\frac{1}{h} \left(\frac{w}{2} + (x - x_0) \right) \right) \right] \quad (5.4)$$

A plot of flux density against x for both a point and distributed source can be seen in figure 5.12. Each of the distributions were used to fit against the histogram data produced by the module, seen in figure 5.13. The distributed source formula provided a more representative estimate. Both formulae, however overestimate the flux for positions on the module far from the source. This is most likely a result of the bend cuts within the CBCs suppressing hits with large angles of incidence.

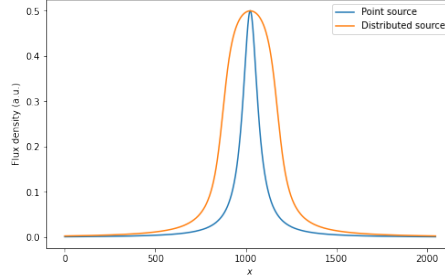


Figure 5.12: Plot of the flux density incident on a plane from both a point and distributed source placed at the same point above the plane.

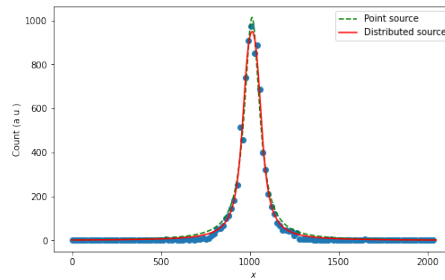


Figure 5.13: Histogram of data taken from the module with both the point and distributed source formulae fitted.

5.3.4 Latency Tests

A further test that was possible with access to both the silicon module and the radioactive source was to measure the end-to-end latency of the readout chain. This was defined as the time taken between a particle being incident on the silicon sensor, to the produced stub being output within the DTC firmware on the processing card.

Determining the start time required a trigger independent from the sensor, in the case of this experiment, a scintillator and PMT were used (see figure 5.14). As a particle passes through the scintillating material, it emits photons which are then amplified and converted to an electrical signal within the PMT. This signal was then sent over a coaxial LEMO cable to an oscilloscope where it could be displayed.

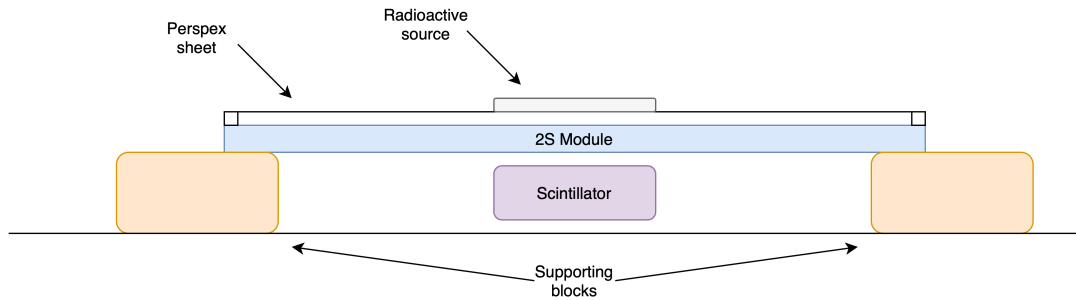


Figure 5.14: Diagram showing the experimental setup used for the latency measurements.

The time at which the corresponding stub was extracted within the firmware of the DTC was calculated by first creating a custom firmware image, in which the valid bit for the stubs within the Link Interface was routed out to an output pin on the FPGA package which was electrically connected to one of the General Purpose Input/Output (GPIO) pins on the Serenity card. This pin could then be connected directly to the same oscilloscope as the scintillator, figure 5.15.

With both signals now present on the oscilloscope, the time difference between the two would then yield the latency of the system. Having a coincident signal from both the scintillator and DTC however, proved to be rare. By using the infinite persistence function of the oscilloscope it was possible to record all traces over a period of minutes, allowing for sufficient statistics of coincident events to be recorded. A threshold trigger on the PMT signal subsequently improved the purity of the data. An example capture of this data can be seen in figure 5.16.

The primary signal coming from the DTC is spread across 200 ns, corresponding to

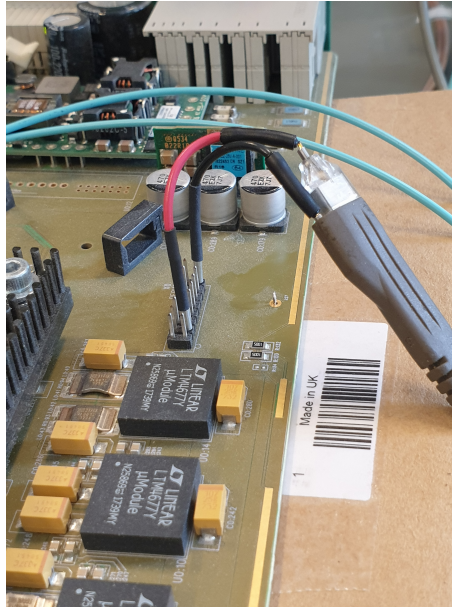


Figure 5.15: Connection of the oscilloscope to the GPIO pins on the Serenity card.

the period of the boxcar sent from the CIC. A hit in the sensor and thus also a signal from the scintillator, can occur at any time in that period, meaning that when fixing the scintillator signal in time the corresponding DTC signal is spread over the 200 ns. As a stub from the 8th Bx has the shortest time between being generated and sent over the link, this corresponds to the leftmost signals on the oscilloscope. The secondary signals from the DTC seen in figure 5.16 can be attributed to stubs from the following boxcar after the primary signal.

It was possible to improve the accuracy of the latency calculation by modifying the firmware to select only stubs from the 8th Bx, and hence providing the minimum possible latency. By inspecting captures using this firmware (figure 5.17), the latency was measured to be 812 ± 4 ns.

5.3.4.1 Calibration

The latency measured in the capture was not however, the true latency between the incident particle and the stub generated in the DTC. The sensor module and scintillator were located on the opposite side of the laboratory to the the Serenity card running the DTC firmware and the oscilloscope. This meant there was a significant delay caused by the cable connecting the scintillator and oscilloscope, as well as non-zero delays within the PMT itself. The cable delay was directly measured by connecting both the cable to be measured and a secondary cable to the standard and inverting outputs of a signal

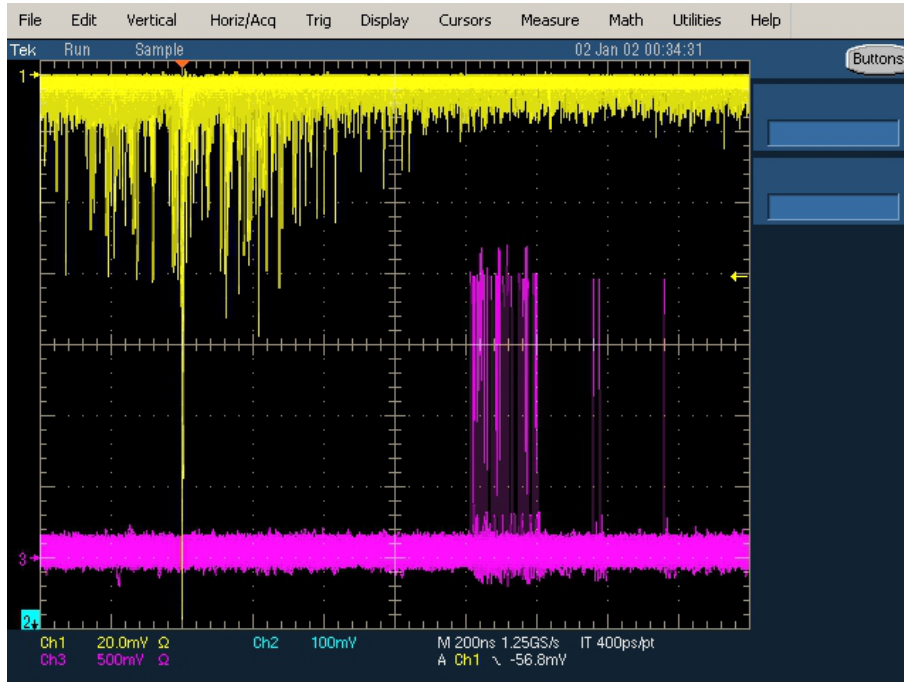


Figure 5.16: Screen capture of oscilloscope data showing both the scintillator (yellow) and DTC (pink) signals.

generator and an oscilloscope. The frequency of the signal generator was then tuned such that both of the signals were in phase, figure 5.18, at which point the latency difference between the two cables is half the period of the signal generated. For the cable connecting the scintillator to the oscilloscope, this was measured as 55.6 ± 0.2 ns.

Full table of the latencies associated with the measurement can be seen in table 5.1.

Component	Latency (ns)
PMT	1.5 ± 0.1
PMT Cable	7.5 ± 1
LEMO Cable	55.6 ± 0.2
Total	64.6 ± 1.3

Table 5.1: Cable length latencies.

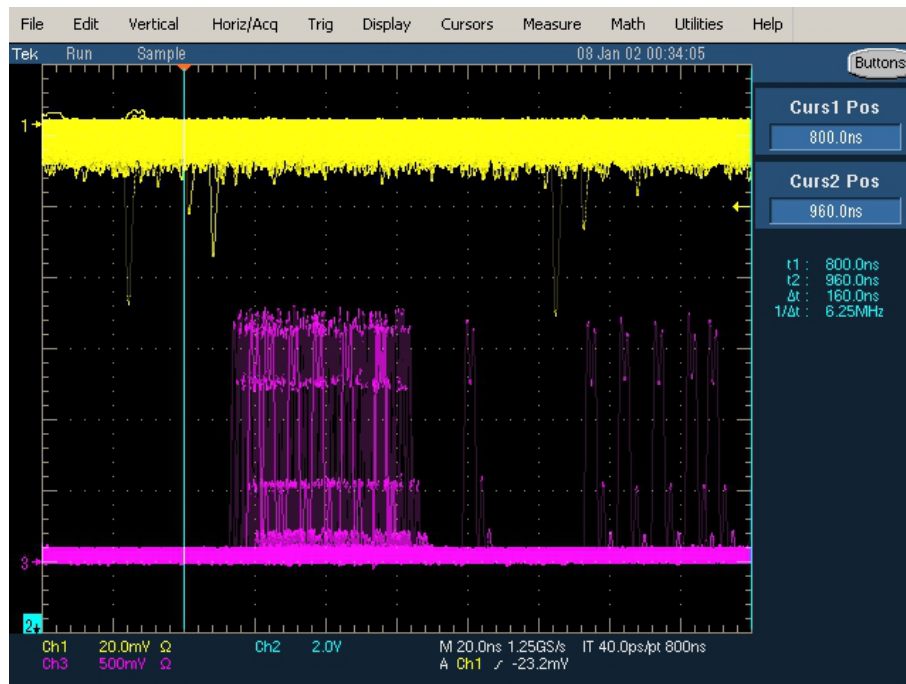


Figure 5.17: A zoomed in capture of the oscilloscope data, showing only the signal region for the DTC. The blue cursor is located at a latency of 800 ns from the trigger.

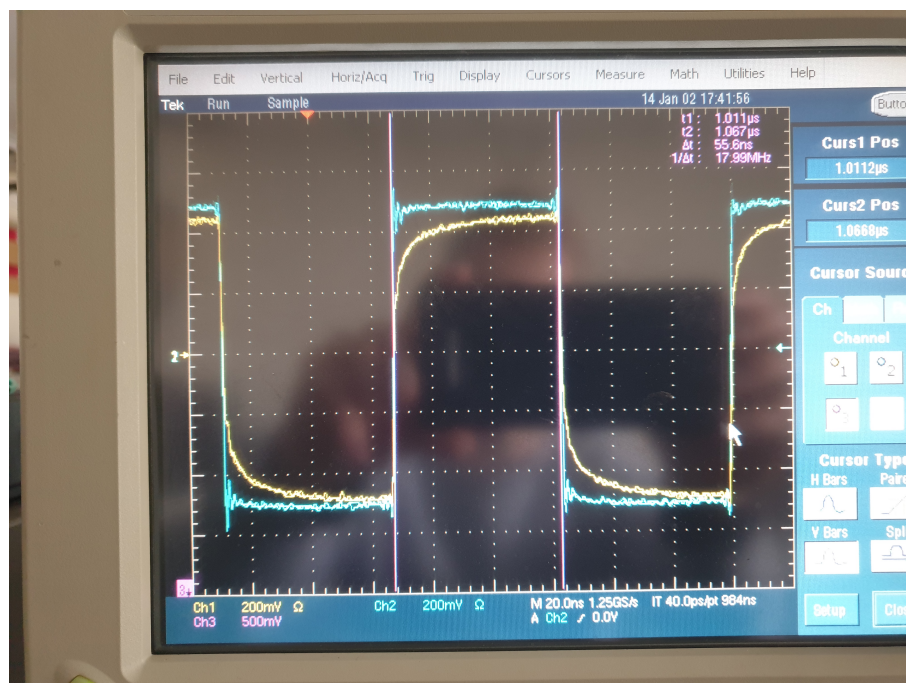


Figure 5.18: A photograph of the oscilloscope output when the signal generator is tuned. The time difference for half a period is shown by the two cursors.

Chapter 6

The MUonE Experiment

6.1 Introduction

In recent years, the muon sector has shown promise with results suggesting discrepancies with the Standard Model. Fermilab’s Muon $g - 2$ [61] has presented a significant deviation away from the Standard Model, in the measurement of the anomalous magnetic moment of the muon.

For a tree level calculation of the magnetic dipole moment, the result would be exactly two, however corrections must be added for additional loops. This “anomalous” magnetic moment is parameterised in the following equation:

$$a = \frac{g - 2}{2} \tag{6.1}$$

For the muon, there are 3 contributors to a , QED, electroweak and hadronic. The first two contributions can be well modelled, however the hadronic, denoted a_μ^{HLO} , and thus QCD-based, contribution is non-perturbative, making it far more difficult to calculate. Examples of interactions which contribute to the magnetic moment of the muon can be seen in figure 6.1.

Given the difficulty in calculating a_μ^{HLO} , an accurate experimental measurement of the value could provide an independent verification of the $g - 2$ result. It would allow for a reduction in the associated uncertainty for this contribution to $g - 2$, whilst an unexpected value may provide an insight into the source of the discrepancy between predicted and measured values for $g - 2$. The t-channel process, $\mu e \rightarrow \mu e$, can provide direct sensitivity to a_μ^{HLO} through a calculation of the running fine structure constant $\alpha(t)$. Integrated

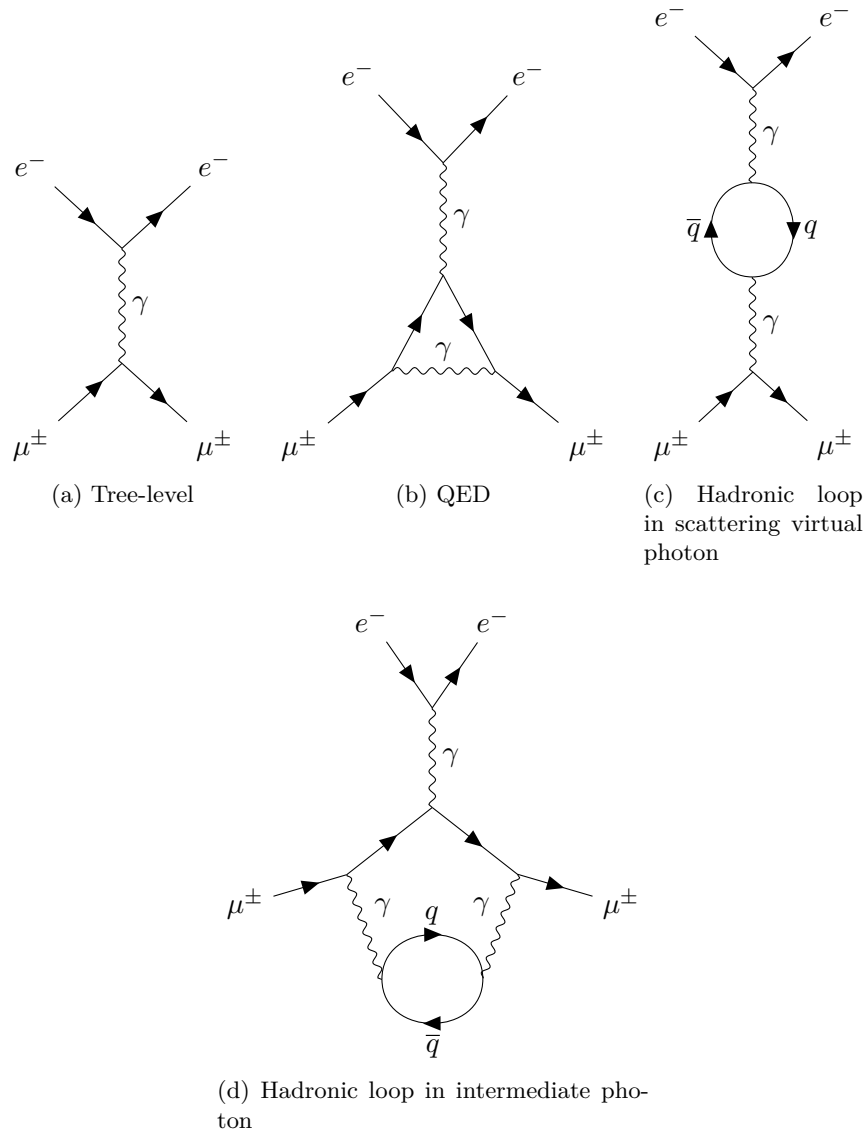


Figure 6.1: Feynman diagrams showing the contributions to α for $\mu e \rightarrow \mu e$ t-channel scattering.

over $-\infty < t < 0$, the hadronic contribution to the anomalous magnetic moment can be recovered from the hadronic contribution to α [62]:

$$a_\mu^{HLO} = \frac{\alpha}{\pi} \int_0^1 dx (1-x) \Delta\alpha_{had}[t(x)] \quad (6.2)$$

where t is the space-like squared four-momentum transfer, parameterised such that $0 < x < 1$:

$$t(x) = \frac{x^2 m_\mu^2}{1-x} \quad (6.3)$$

6.2 Extraction of a_μ^{HLO}

In order to calculate $\Delta\alpha_{had}(t)$ and thus, by extension, a_μ^{HLO} , the measured differential cross-section is compared to a purely QED Monte Carlo simulation. A deviation from 1 of the ratio, $R_{had}(t)$, of measured divided by Monte Carlo data in each bin can thus be attributed to the hadronic contribution, $\Delta\alpha_{had}(t)$. Once sufficient data is collected to cover the full range of momentum transfers possible within the experiment, a two parameter fit is performed, using a function which closely matches Jegerlehner's numerical parameterisation of $\Delta\alpha_{had}(t)$ [63, 64], the functional form of which can be seen in figure 6.2. This function, which can then be extended to cover $-\infty < t < 0$, is used within the integral given in equation 6.2 to provide a measurement of a_μ^{HLO}

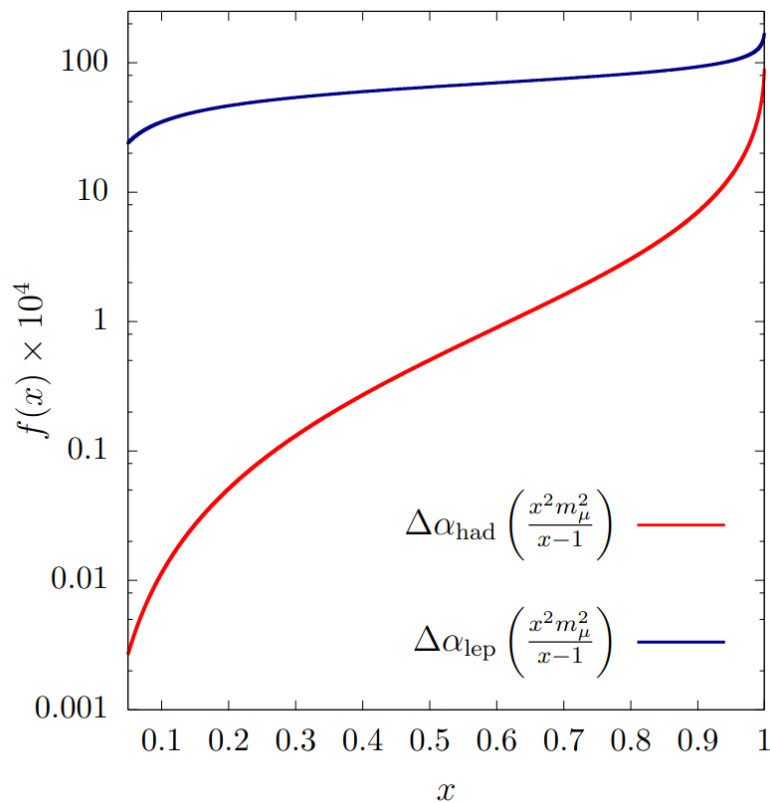


Figure 6.2: Plot showing the functional form of both the leptonic and hadronic contributions to the running fine structure constant. Source: [62].

6.3 Experimental Setup

MUonE will be a fixed target experiment, whereby a high energy, high intensity muon beam will collide with a low-Z (Beryllium) target. The resulting elastic scattering between muon and atomic electrons will be accurately tracked downstream by a series of silicon sensors. The M2 beamline at CERN, capable of delivering muons with a momentum of approximately 150 GeV/c at a nominal intensity of 1.3×10^7 muons per second is an ideal candidate for the experiment. In particular the high rate of muons will allow for a high level of statistics to be achieved in a reasonable timeframe for the experiment.

The goal of the experiment is to accurately measure the differential cross-section of elastic $\mu e \rightarrow \mu e$ scattering with respect to the momentum transfer during the scattering, thus one of the the largest sources of error will be in Coulomb scattering after the initial interaction, thereby inducing an error in the angle of the outgoing electron track. This can be minimised by having a thin target, but at the cost of a lower interaction rate. In order to ensure sufficient statistics can be accumulated in an acceptable time, the MUonE experiment will have a modular design, with many individual “stations”, each with a

15mm target at their start, thus allowing for an “effective” target thickness to much greater than a single target. The station will then contain within it 6 silicon tracking modules to record the paths of scattered particles. In the final design, there will be 40 of these stations, positioned end-to-end along the beam line (see figure 6.3). Downstream of the final station will also be an ECAL to aide particle identification and to measure electron energy.

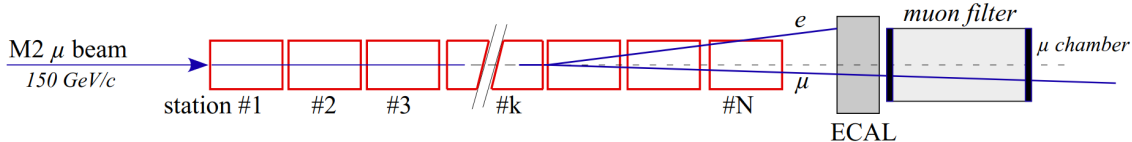


Figure 6.3: Diagram of the station-level composition of the MUonE experiment. Source: [62].

MUonE will make use of the the research and development effort being made in the CMS Phase-II upgrade by using the same 2S modules designated for the OT within the stations. These sensors are ideal for this experiment as a result of their high spatial resolution, capacity to readout hits at 40 MHz and existing development work for DAQ. The 2S module, being a strip sensor, can provide accurate positional information in one axis, at the cost of limited precision orthogonal to that axis. In order to account for this, the modules within the station will be organised into pairs, each with one module rotated 90° from the other around the axis of the beamline so as to provide a combined coordinate which is accurate in both dimensions. As shown in figure 6.4, two pairs of such modules, with the strips running horizontally and vertically will be installed at the beginning and end of the station respectively. In the middle will be a third pair, with the modules instead rotated at 45° from the other pairs. These will be used to discriminate between real and “ghost” pairs of hits in each of the other pairs. Such pairs arise when more than one hit is recorded in both the x and y -oriented modules in a pair; in such the case of two hits per module, four combinations of these hits can be created, two of which are genuine and the other two are “ghosts”. The hits in the rotated plane will only align with the genuine hits, thus allowing for discrimination to be achieved.

Similar to the CMS OT, the modules will be read out to a Serenity BE processing card over optical links. Decoding, formatting and intra-station track-finding will be performed on the FPGAs on the card itself, before the data is saved to disk to be processed offline. Whilst the occupancy of the modules will be similar to that in CMS, the processing required in the DAQ system will differ significantly. MUonE will operate in a “triggerless

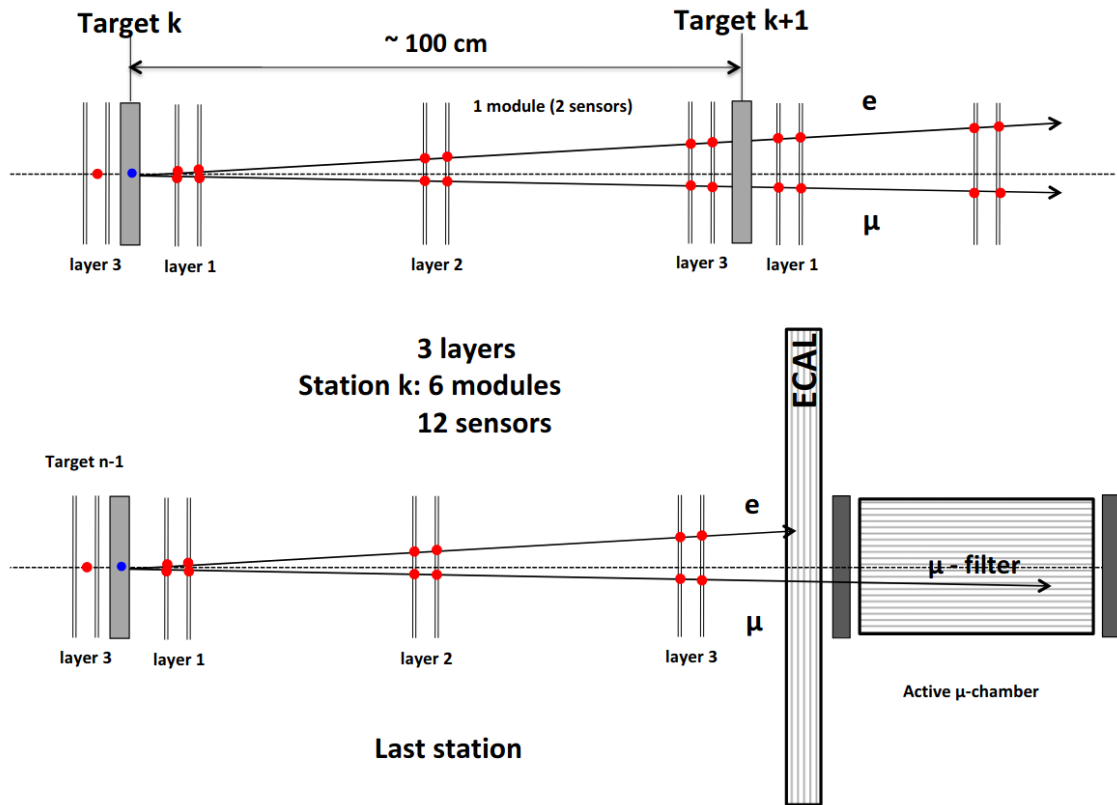


Figure 6.4: Diagram showing the positions of the six tracking sensors within a given station. Source: [62].

readout” mode, whereby the data will be streamed through the processing card, with any selections made in real-time. This differs from CMS, whereby the data will be buffered and a trigger propagated back to the modules to allow for a full readout of the data to happen.

In comparison to HL-LHC, the data rate that will be produced by MUonE will be low, thus a trigger system will not be strictly required. Nevertheless, investigations are ongoing to understand the feasibility and potential reductions in offline computing resources that could be achieved by an online trigger. In particular, the selection of only events containing two tracks, one each for the muon and electron respectively, would reduce the accepted event rate significantly, whilst putting a further constraint on the vertex of the two tracks emanating from a target would reduce the rate even further.

Chapter 7

MUonE Test Beam

In collaboration between CMS and the MUonE experiment, a beam test was conducted in October 2021 at a facility in CERN. Whilst previous tests with modules in a beam[65, 66] had used the DAQ path to readout, in this test, the trigger path was readout at 40 MHz for the first time. In this chapter, the test setup is described, including the characteristics of the beam used and the hardware installed. Additional firmware that was required to successfully read and store the data generated is presented, as well as the operational procedures and challenges faced.

7.1 Experimental Setup

7.1.1 The M2 beam

The M2 Beam, situated in the CERN North Area is one of 6 secondary particle beams formed by colliding protons from the SPS with fixed targets, typically beryllium. These collisions generate secondary hadrons, typically pions, the decay products of which can then be selected through spectroscopy to provide a beam for a given experiment.

In the case of the M2 beam, it is not these secondary hadrons which are desired, but instead the muons into which they decay. This is enabled by having a beam section, 600 m in length, providing sufficient time for both pions and kaons to decay into muons. The former by the simple decay $\pi \rightarrow \mu + \nu_\mu$ and the latter by first decaying to 3 pions and then each charged pion decaying to a muon. At the end of this decay section, a hadron absorber filters out remaining hadrons, before the beam is bent and focused into the EHN2 experimental hall, where the beam has the characteristics as shown in table 7.1.

Given the cyclical nature of the SPS, muons arrive at the experimental hall in so

Beam Parameters	Measured
Beam momentum p_μ/p_π	160/172 GeV/c
Proton flux on T6 per SPS cycle	1.5×10^{13}
Muon flux at EHN2 per SPS cycle	2.5×10^8
Beam polarisation	$-80\% \pm 4\%$
Spot size at COMPASS target ($\sigma_x \times \sigma_y$)	8 mm $\times$ 8 mm
Divergence at COMPASS target ($\sigma_{dx} \times \sigma_{dy}$)	0.4 mrad $\times$ 0.8 mrad
Muon halo within 50 mm from beam axis	16 %
Halo in experiment ($6\text{ m}^2 \times 4\text{ m}^2$) at $ x, y > 150\text{ mm}$	7 %

Table 7.1: Beam dynamics of the M2 muon beam when incident at the COMPASS experiment, located just upstream of the MUonE test stand.

called “spills”. SPS spills are shared out over all experiments connected to the accelerator, including notably the LHC which is downstream from the SPS. Whilst bunches in the LHC are spaced by 25 ns, the spill structure of the SPS is at a far lower frequency, with spills typically in the order of seconds within the accelerator. As such, there was no correlation between the time of arrival of an incident muon from the M2 beam and the LHC clock used to synchronise the detector components.

As well as the primary operating mode of a muon beam, M2 can also deliver a high intensity hadron beam with momenta up to 280 GeV/c and a low energy, low intensity electron beam that can be used for calibrations.

7.1.2 MUonE Test Stand

For the test beam in October 2021, four 2S modules were provided through three institutes: one from Karlsruher Institut für Technologie (KIT), one from Brussels and two from Perugia. The former two constituted the CMS contribution to the test beam, whilst the latter two were manufactured specifically for the MUonE experiment. As such, the CMS modules were placed in the beam within a separate box to the two MUonE modules, which were placed in a prototype station apparatus (figure 7.1). Modules placed in the station required a new mounting frame to the one provided by CMS, hence the separation between the two. As shown in figure 7.2, the CMS modules were upstream to the station. Attached on the upstream side of the CMS was also placed a scintillator and PMT to provide triggering information to the DAQ system.

The area used for the test beam was located downstream of the currently inactive Common Muon and Proton Apparatus for Structure and Spectroscopy (COMPASS) experiment, at the end of EHN2. The modules and surrounding housing were placed on an

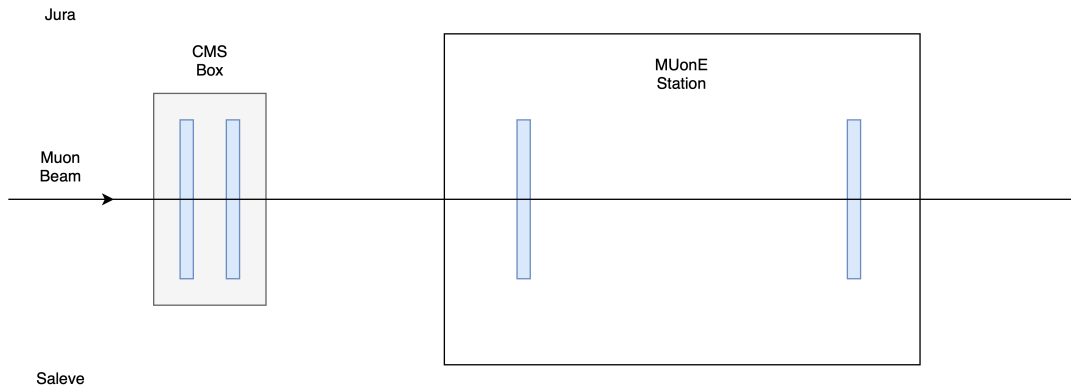


Figure 7.1: Diagram showing a top-down view of the test stand setup, with each of the four 2S modules highlighted in blue.

elevated concrete platform in order to be aligned with the muon beam. Power supplies for the tracking station and BE readout electronics were placed in a rack to the side of the platform, reducing radiation exposure (figure 7.3a). As a result of cable-length limitations for the CMS modules, their power supplies and readout electronics were placed on the elevated platform along with the modules themselves, on a table set against one of the concrete walls (figure 7.3b).

7.1.2.1 Prototype Station

Designed as an initial iteration of the tracking station for MUonE, the prototype station can house up to six 2S modules and, as shown in figure 7.4a, contains the connectivity to cool, power and control the modules. Water, fed through tubes attaching to the top of the station was used as the primary cooling method for the modules, whereby copper pipes, within which the water flowed, cooled a metal plate, upon which the module was mounted (figure 7.4b). Further environmental control within the station was achieved through the use of dry air, provided as part of the experimental hall infrastructure and further cooled by a chiller placed on the concrete platform adjacent to the station (figure 7.4c).

Similar to the environmental control of the station, power for the modules was provided by an external supply, generating both the Low Voltage (LV) and HV power independently. The housing onto which the modules are mounted sits freely compared to the outer frame, with a set of servos used to translate and rotate the modules within the station. This freedom enables module alignment fine tuning during calibration.

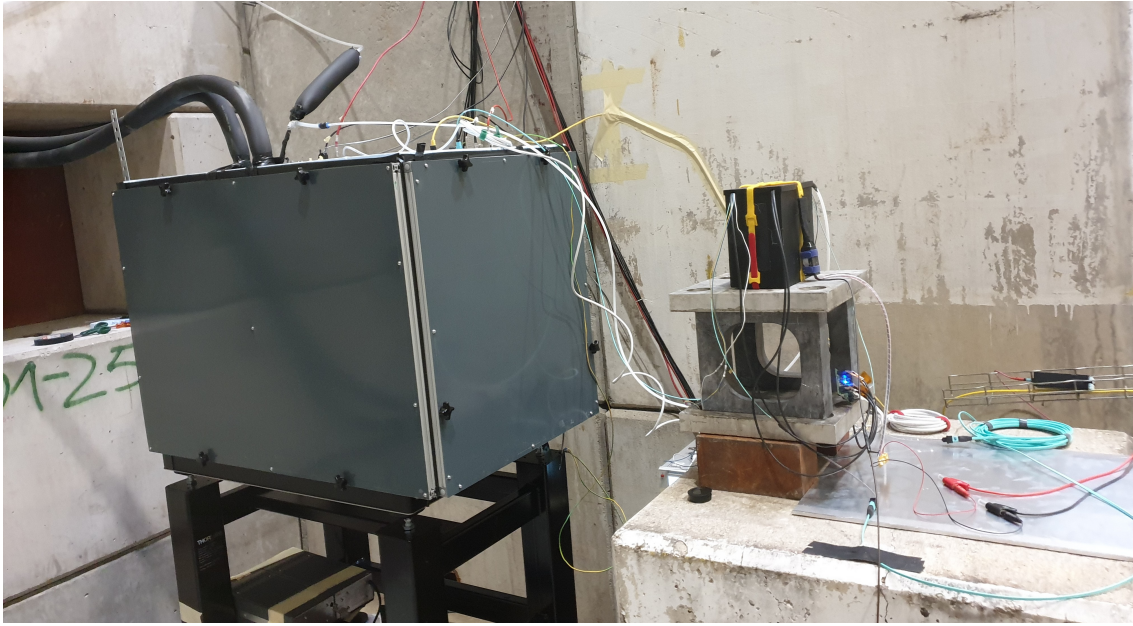


Figure 7.2: Photograph of the test stand within EHN2 with the particle beam travelling from right to left. The prototype station can be seen in the left half of the photograph, including the connections for dry air, sensing, control and readout. The black box in the right half of the image housed the both 2S modules supplied by the CMS collaboration. Note also the scintillator and PMT attached to the front of the CMS box.

7.1.2.2 DAQ System

Two DAQ chains were implemented for the test beam: a so-called “uDAQ” chain utilising an FC7 for readout and a Serenity-based chain for primary data-taking. The uDAQ chain used pre-existing firmware and software that has already been used for previous beam tests and readout tests with 2S modules and allowed for the modules to be correctly calibrated and configured. The limited readout capacity of the FC7 necessitated the use of the scintillator as a trigger for reading out the module. This scintillator signal was also separately read out to a connected PC via the FC7, which in turn exposed the signal as a count to be used in the online software.

The successful demonstration of the Serenity-based DAQ chain was one of the primary goals of the beam test. Both MUonE and the CMS L1 track trigger both require a system in which multiple 2S modules are read out in real time through a Serenity (in the case of CMS, the Serenity is a DTC card). The difference between the two systems is the downstream location of the data after the Serenity card. In the case of CMS, stubs are sent to another processing card for track finding, whereas for the MUonE test, the stubs were instead sent to a PC for storage and offline analysis. The different downstream



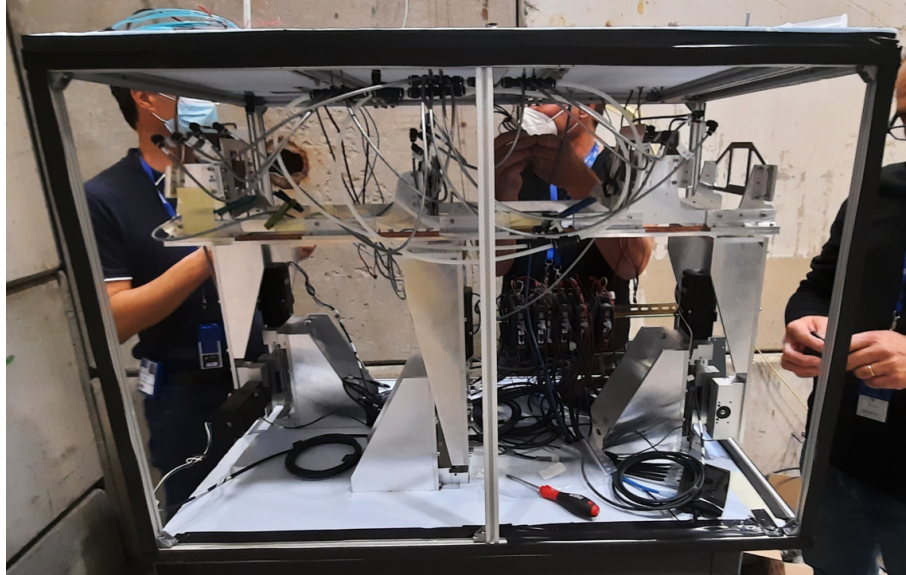
(a) Power supplies and back end electronics rack.

(b) Power supplies and FC7s located on the elevated experimental platform.

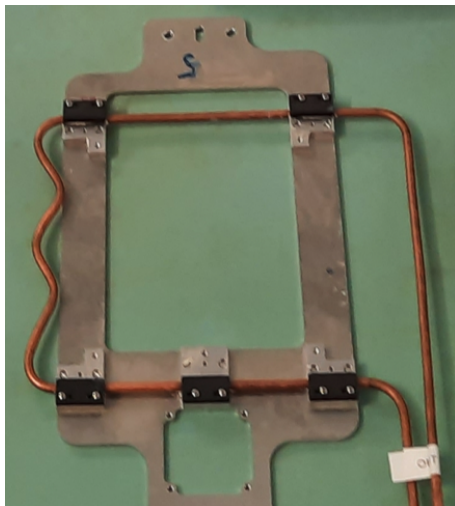
Figure 7.3: Power supplies and readout electronics used for the beam test. Elevated experimental platform can be seen behind the concrete wall in figure 7.3a.

interface necessitated the development of a new firmware block which implemented the ethernet protocol at a 10 Gbps speed.

Connection between the modules and either the FC7 or the Serenity was facilitated by multi-mode optical fibres (figure 7.5). Modules within the prototype station were connected through a single 12-channel fibre passed through a gromit in the top of the station cover. At each end of the fibre, it branched out to 12 individual Lucent Connector (LC) pairs to allow for connection to the module and DAQ electronics. For the FC7, the LC pair connected directly to an AMC mezzanine card installed on the FC7, whilst for Serenity, which had a 12 channel firefly optical module installed, the pairs were connected to an MTP patch panel which was then connected to Serenity by another 12-channel optical fibre.



(a) Tracking station interior



(b) Cooling plate



(c) Chiller

Figure 7.4: Interior of the MUonE tracking station and environment control.

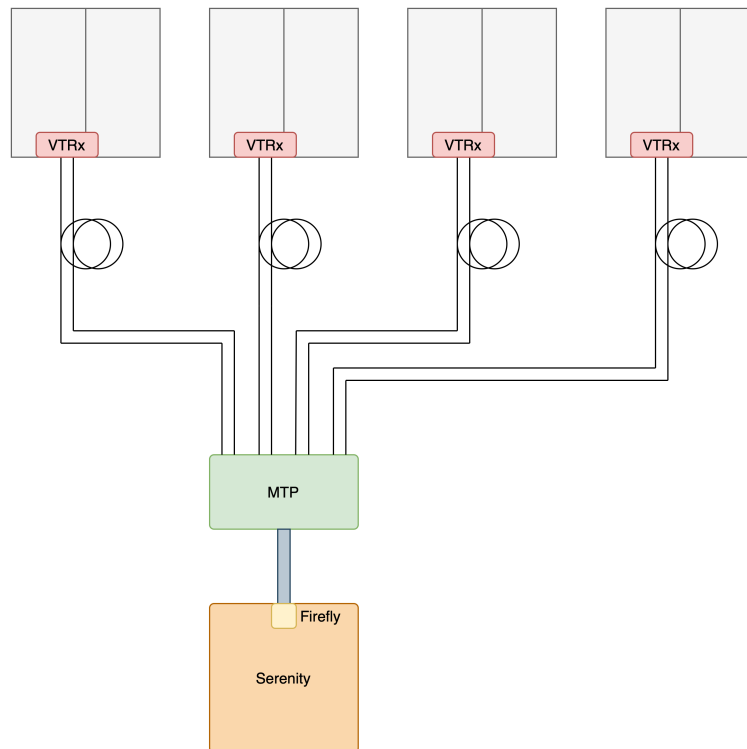


Figure 7.5: Diagram showing the optical readout path between the 2S modules and Serenity BE processing card.

7.2 DAQ Firmware for Serenity

Whilst the role of the Serenity in the beam test was very similar to that of the proposed DTC for CMS, the significantly fewer modules available to connect to it meant that using the firmware designed for the DTC would have used an unnecessary amount of resources within the FPGA. Instead, a simpler routing block was designed to meet the needs of the beam test and potentially the MUonE experiment in the future. The overall design of the firmware developed for the Serenity used in the Beam Test was given the name MProcessor. As seen in figure 7.6, the MProcessor used the same Link Interface firmware block as the DTC, as a single copy of this firmware is required for each module link in order to decode and control it. After this however, instead of the stubs being presented to the formatting and routing stages of the DTC, two new blocks were designed: the so-called “Link Aggregator” and “Link Combiner”, the first of which sorted and collated stubs based on their Bx ID, whilst the latter combined these collections of stubs into a single stream to be sent out on the ethernet output link.

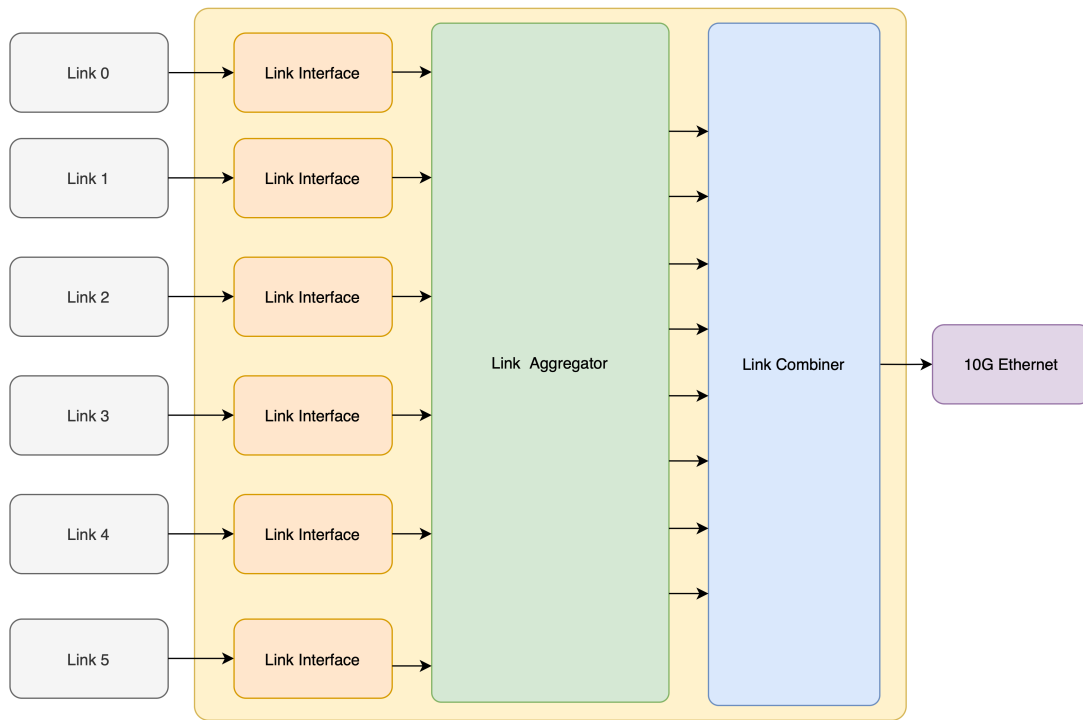


Figure 7.6: Block diagram showing the showing the firmware blocks implemented in the DAQ system used on the Serenity card.

7.2.1 Link Aggregator

The Link Aggregator shares a common goal with the DTC Router, by taking stubs from multiple input links and collating them by their Bx ID. In the case of the MUonE experiment, stubs were collected only over the eight Bx of a boxcar sent from the FE modules, compared to eighteen Bx for the DTC. This was achieved through two layers of FIFOs, with a rotating pointer between them (figure 7.7). As stubs enter the Link Aggregator block, they are first sorted into eight FIFOs per link, corresponding to the eight bunch crossings possible from a boxcar.

Eight independent pointers, one per Bx, then iterate over these FIFOs, taking a stub from each link in turn and placing it into a deeper FIFO. If a FIFO is empty, then no data is transferred on that clock cycle. The result of this stage is eight deep FIFOs, each containing all stubs from a given Bx.

Each stub contained sufficient information such that no further information would be required to use the stub in analysis, meaning that in addition to the 17 bits provided over the front end link, the full BxID should be included as well as a link ID for the origin of the stub. Table 7.2 shows the full breakdown of the bit fields used for the stub during

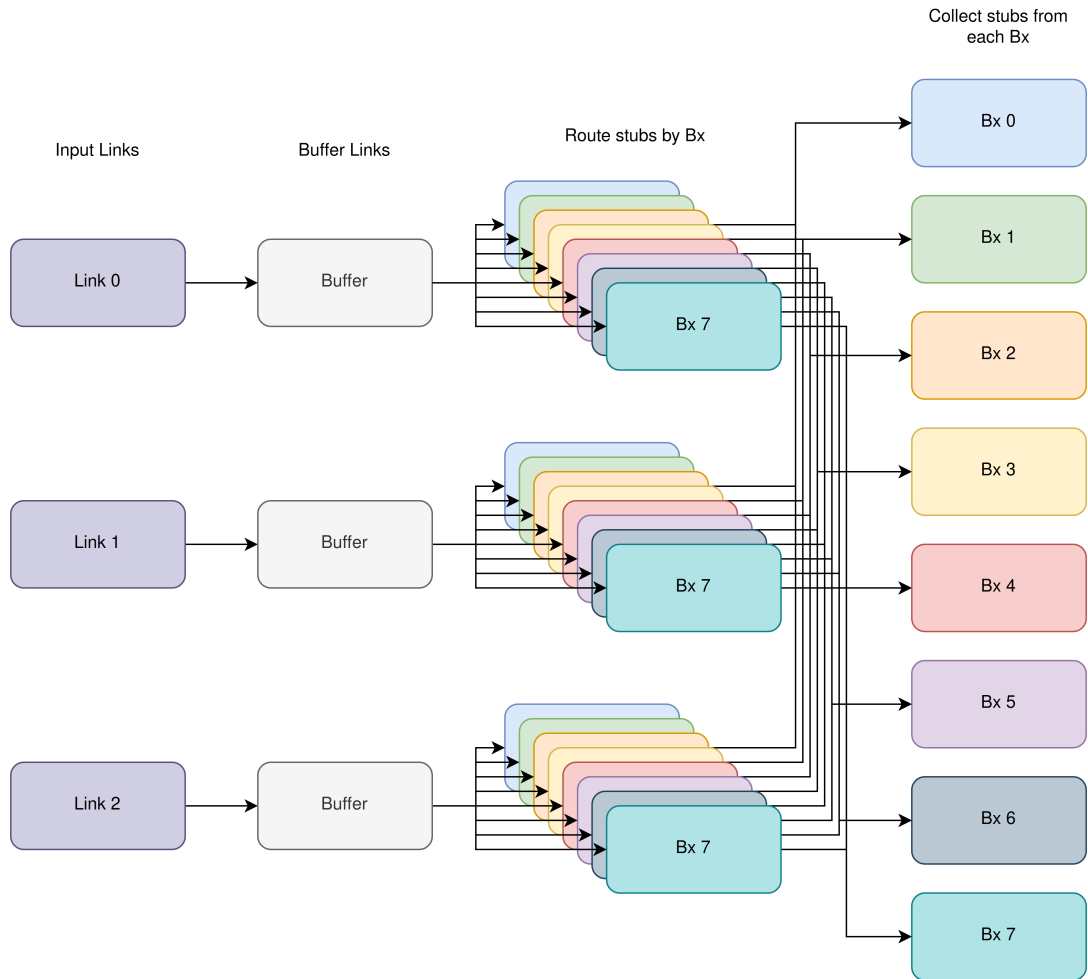


Figure 7.7: Block diagram showing the organisation of the FIFOs within the Link Aggregator for three input links.

processing. Throughout the Link Aggregator, Link Combiner and the output ethernet link, the EMP `lword` was used as the interface, meaning that 64 bits were available per clock cycle. Given the 32-bit length of the stub, two stubs could fit into a single `lword`, giving the twofold increase in throughput. If the total number of stubs for a given Bx is odd, half of the `lword` is left empty and filled with zeros. It is not possible for the 2S modules to send a stub with all fields being zero, as the address 0 does not exist on the module. This ensures that the empty half-word cannot be interpreted as a genuine stub in the analysis.

Bits (inclusive)	Value
3 - 0	Bend
11 - 4	Address
14 - 12	CBC ID
26 - 15	Bx
27	CIC ID
31 - 28	Link ID

Table 7.2: Table showing the bit fields for a stub output from the Link Aggregator.

7.2.2 Link Combiner

In the case of the beam test, a single 10 Gbps ethernet link was instantiated, therefore it was necessary to combine the eight collections of stubs into a single stream to be sent out on the link, a task performed by the the Link Combiner. In the first stage of the Link Combiner, the collections of stubs are buffered into a set of identical type of FIFOs to the output of the Link Aggregator. After which a pointer then iterates over the eight FIFOs in turn, emptying one before moving to the next (figure 7.8). This meant the stubs were still grouped by Bx and these collections were also ordered to allow for ease of decoding and storage at the other end of the ethernet link.

In cases where an odd number of stubs were collated in total from a given Bx, and therefore one for the words was half full, this would simply be seen in the data stream as a gap of all zeros. It may be possible for multiple odd collections to be merged in order to maximise bandwidth, however the development and potential processing overhead cost of doing so was not deemed worthwhile.

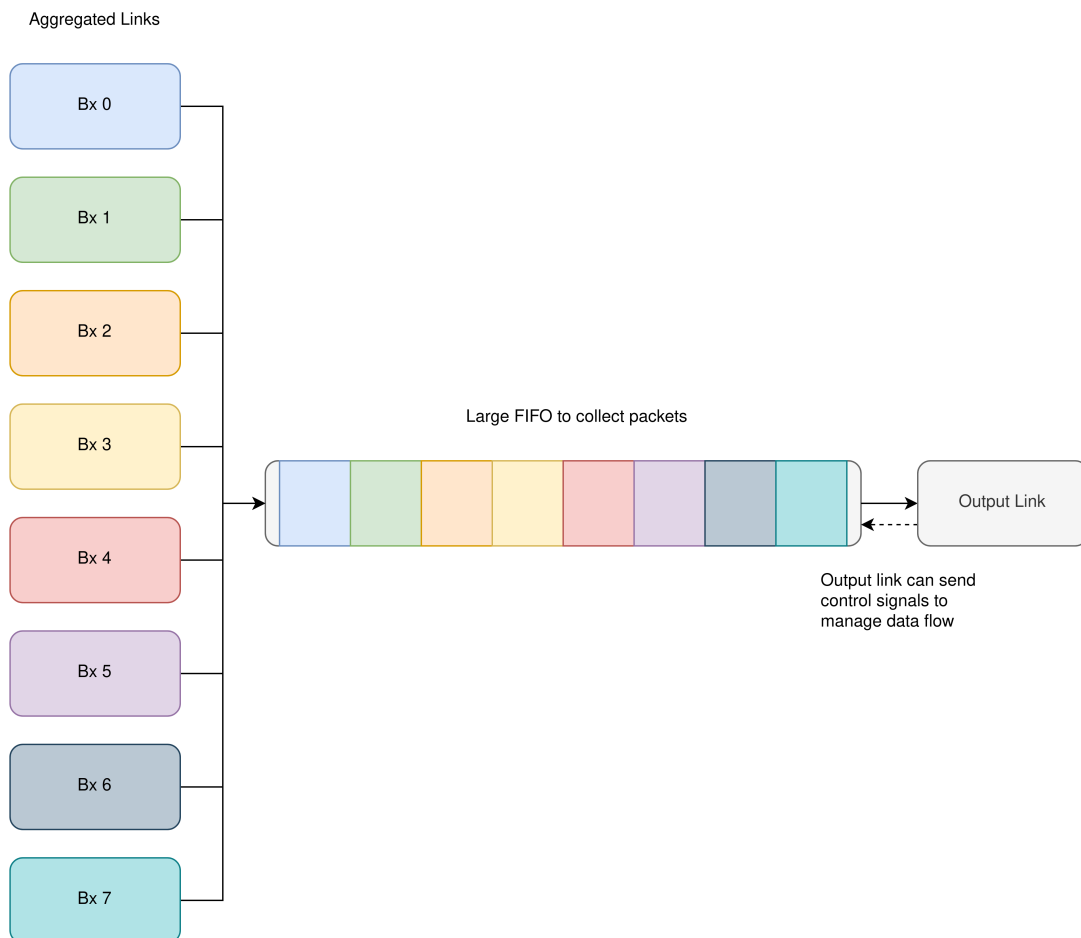


Figure 7.8: Block diagram the arrangement of FIFOs in the Link Combiner firmware block.

7.2.3 10 Gigabit Ethernet Link

Alongside the development of the stub processing firmware, it was also necessary to integrate into the design firmware to interface with the transceiver used for the 10 Gbps ethernet link (figure 7.9). Whilst the ethernet stack firmware block was a pre-existing IP core, its interface was AXI, meaning that the data packets had to be translated into this protocol before being sent.

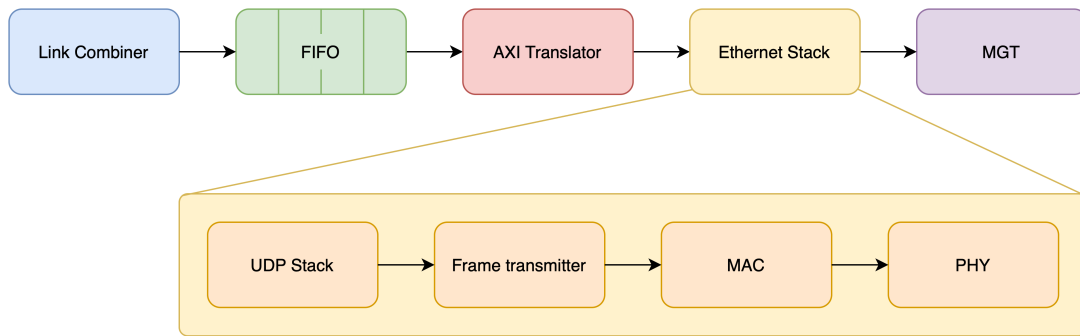


Figure 7.9: Block diagram the arrangement of FIFOs in the 10G Ethernet firmware block.

7.2.4 Limitations and Scaling

With the simplicity of the algorithm used for the Link Aggregator, there are also limitations when scaling the number of input links and data rate. The number of stubs per Bx that can be sent from the Link Aggregator to the Link Combiner is set to sixteen. This limitation arises from the time needed to empty the aggregated FIFO at the end of the Link Aggregator block, whereby only one stub can be read out on each clock cycle. At 320 MHz, which was the payload frequency used for all development of the firmware development for MUonE, this gave a remaining 48 clock cycles on which to route stubs to the aggregated FIFOs. This in turn provides a limit of $48/N$ stubs per link that can be sent per boxcar, where N is the number of input links. In the case of the beam test, $N=4$, so at most 12 stubs could be sent from each link per Bx. In the case of more stubs being available at either of the stages described, these would be lost as the FIFOs are reset such that all remaining data is lost at the end of each boxcar period. In such cases, these events would be flagged as incomplete in the downstream data collection to ensure that no biases were induced as a result. Similarly in the Link Combiner, if each Bx FIFO contains 16 stubs, there would not be sufficient clock cycles over which the data could be

transmitted to the output link. In this case, the higher Bx IDs (e.g. 6, 7) would be lost as they are the last to be read out by the algorithm. The data rate for the beam test was expected to be far below these limits, therefore these limitations were acceptable.

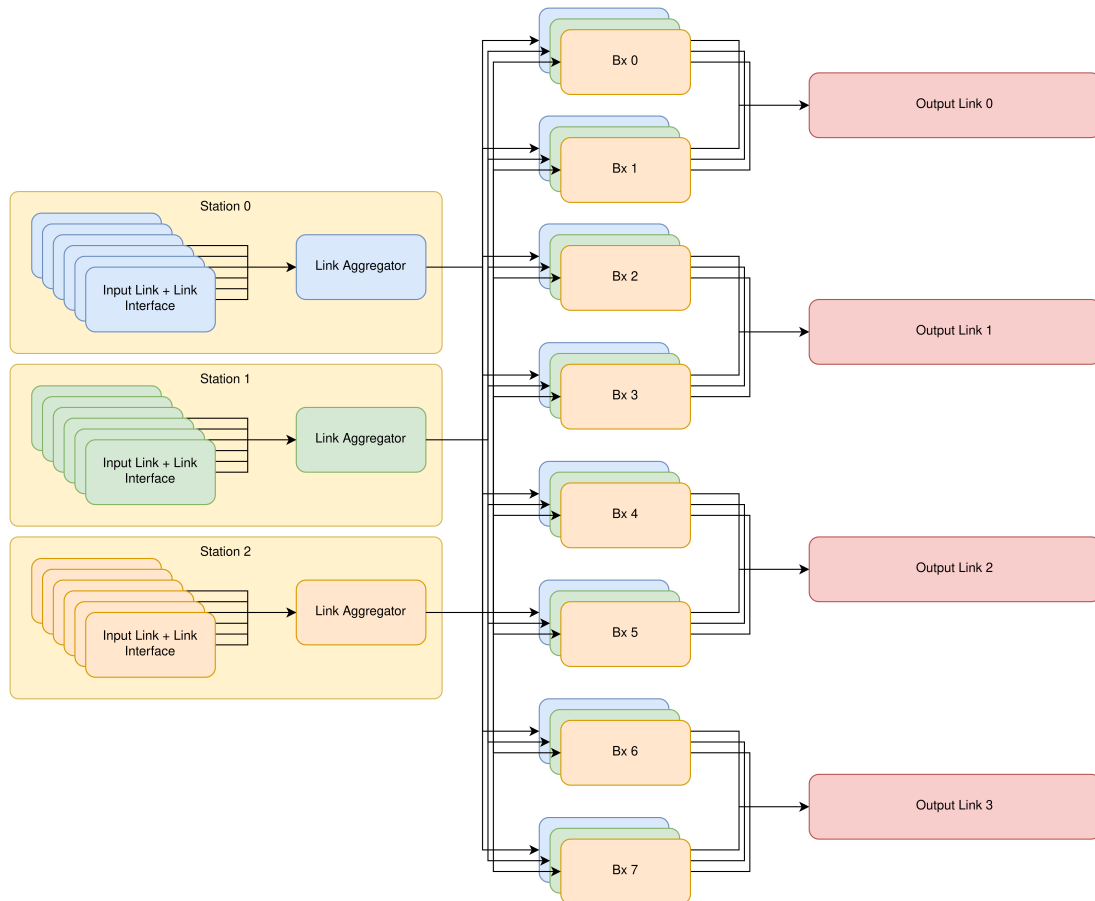


Figure 7.10: Block diagram showing an MProcessor connected to three stations.

In the final MUonE experiment, it is expected that multiple stations will all connect to a single Serenity card. As each station contains six modules, the Serenity card will therefore have to handle $S \times 6$ where S is the number of stations up to a maximum of 20. For any $S > 1$, using the Link Aggregator and Combiner as described above would not be feasible. Since the link number scales, the maximum number of stubs per link per Bx inversely decreases. For 48 clock cycles of stub processing, 6 modules would allow for 8 stubs per link per Bx, however that number reduces to 4 stubs in the case of two stations and a total of 12 modules. It is therefore envisaged that links will be first aggregated within their constituent station, meaning a maximum of 6 links per aggregator. The number of output links will not, however, scale linearly with the number of stations connected. Instead it is planned to have four 10 Gbps ethernet links to transport all stubs from all stations. In

this case, the MProcessor would contain four link combiners, each outputting data to a single 10 Gbps . Each Link combiner would take all data from two Bx, e.g. Bx 0, Bx 1 for Link Combiner 0, collating data from each of the stations (figure 7.10). This ensures that all event data from a given event is sent on a single link, easing downstream analysis and storage. Given the independence of each event, this method should be effective in load balancing the data across output links. It has already been demonstrated in the MProcessor for the beam test that a single link can handle 8 input data streams, therefore up to four stations would be equivalent in data rate. Expanding beyond this in a single Serenity card would require further studies and testing to ensure that there would not be significant truncation.

7.3 Operations

7.3.1 Beam Conditions

For the beam test of November 2021, the MUonE experiment was not the primary experiment on the M2 beam line. Instead it was a piggyback experiment to NA64[67], located upstream from the MUonE test stand and as such, much of the running schedule and beam conditions were dictated by the upstream experiment. As seen in figure 7.11, during only approximately an eighth of the available running time was it possible to record data at the test stand. The time apportioned to “no beam” includes time when the beam was shut off for technical stops for the experiments, as well as when the beam was running in either hadron and electron modes. In both of these beam modes, used in NA64 for calibration of the detector, no particles were incident on the MUonE test stand and thus no relevant data could be recorded. On top of the time without beam, almost 25% of the total running time was lost to being unable to record data whilst beam was present. These losses arose as a result of technical issues within the test stand or DAQ system which could not be rectified remotely and required manual intervention within the beam area. The particular inflexibility within the NA64 for political reasons for short interventions to fix these issues meant that in some cases days were lost before access was granted again.

Beam quality was also poor during the beam test. The luminosity of the beam on arrival at the experimental station was diminished by both the NA64 experiment and COMPASS, which whilst inactive during the beam test, contributed significant material budget to the area upstream of the MUonE test stand. Furthermore, focusing magnets

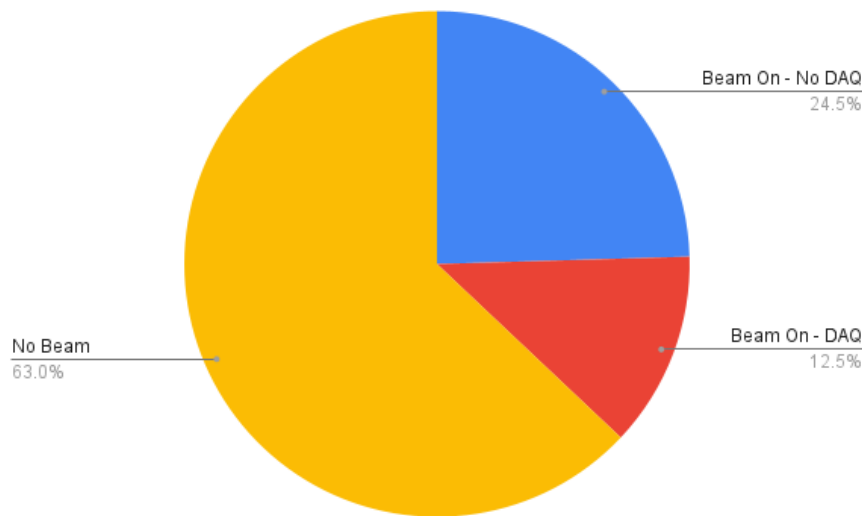


Figure 7.11: Chart showing the beam uptime and DAQ availability during the November beam test.

employed by NA64 distorted the beam downstream of their experiment, reducing its intensity and moving it away from its nominal location. This meant that for the majority of running, the MUonE test stand was at best on the edge of the beam and was never centred on it.

7.3.2 Control and Monitoring

Successful acquisition of data during a beam test is not possible without a reliable system to both monitor and control each aspect of the experiment. Whilst development of software to remotely control a Serenity card for use by CMS had been conducted prior to the test beam, it was not in a sufficiently mature state to be used. Instead, control of the Serenity DAQ system, including both itself and the the connected modules was achieved through a set of scripts running on the SoM attached to the Serenity card. These scripts were capable of configuring the modules through the Ph2ACF software package, and also the MProcessor by use of IPBus registers located within the firmware.

The registers provided the ability to reset each component of the MProcessor individually, as well as set input masks for both the Link Aggregator and Link Combiner. The masks were designed as a method for controlling the bandwidth allowed into each block by blocking off certain links in the case of the Link Aggregator and certain Bx IDs for the Link Combiner. The final output of the Link Combiner was also controlled by an enable bit, allowing for the link between sink PC and Serenity to be validated before the

commencement of data taking.

A record of the parameters required to configure each of these IPBus registers was stored in a MongoDB database in JSON format. An example entry to this database can be seen in figure 7.12. Along with the register values, the firmware package identifier to be loaded onto the FPGA was included as a field within the database. A unique timestamp for each entry enabled traceability between a given data-taking run and its configuration parameters.

```
1 {
2   _id: ObjectId('618dd5213026fd0c02d01945'),
3   created: ISODate('2021-11-12T02:44:49.105Z'),
4   'package': {
5     image: 'mprocessor',
6     tag: 'reduced-histogram-size-7453c79b'
7   },
8   'fe-modules': {
9     threshold: 550,
10    reconfigure: true,
11    disablePerugiaChip7: true
12  },
13  registers: {
14    payload: {
15      csr: {
16        windowH: 1,
17        windowL: 0,
18        histogram_sel: 3
19      },
20      mprocessor: {
21        header_user_bits: '0xd451d007',
22        input_link_mask: {
23          link_aggregator: '0b1111',
24          link_combiner: '0b11111111',
25          head_start_select: 1
26        },
27        link_combiner_control: {
28          enable: 1
29        }
30      },
31      eth10g: {
32        ctrl: {
33          reg: {
34            echo: 0,
35            run: '0x02c',
36            aggregate: 1,
37            aggregate_limit: '0x101',
38            rst: 0,
39            heartbeat: 1
40          },
41          local_ip: '0xC0A80352',
42          remote_ip: '0xC0A80302',
43          ports: {
44            local: '0x04D2',
45            remote: '0x04D2'
46          },
47          watchdog_threshold: '0x80',
48          heartbeat_threshold: '0x13000000'
49        }
50      }
51    }
52  }
53 }
```

Figure 7.12: Example entry to the configuration database.

The database itself was hosted as a docker container on the sink PC and entries were pulled down to the Serenity using the unique `ObjectId` created at the same time as the entry. The NoSQL format of MongoDB was specifically chosen for the configuration database for its flexibility in data format. As opposed to a classic Structured Query Language (SQL) database, which requires a specific schema to be known before at creation of the database, MongoDB documents are stored as individual JavaScript Object Notation (JSON) objects and thus can have entirely different structures. Given the scripts, and thus the parameter structure, evolved during development before and during the beam test, this flexibility was highly beneficial.

Whilst the control aspect of the beam test online software relied heavily on custom scripts, much of the monitoring was achieved through the use of existing commercial software. Statistics were generated and aggregated using the Prometheus software stack[49] and its data model. Prometheus uses a pull-style model for data collection, whereby each node exposes a set of variables as a web page, which is then successively scraped at a set frequency by the aggregation server. In the case of the test stand, three nodes were created to be scraped: the sink PC, the Serenity ComExpress and a monitoring PC for the FC7, to which the PMT was connected, which gave an independent reading of the live beam intensity.

Each node exposed a different set of metrics, pertaining to their individual roles within the system. The sink PC metrics focused on the PC hardware itself, providing information on CPU, RAM, disk and network usage, since these were critical to its role in the system. A script which periodically polled all MProcessor IPBus status registers on the the Serenity card was developed to work with the Prometheus package, allowing for the status of the FPGA to be logged and monitored. A similar custom script exposed the hit count of the PMT to the Prometheus stack, providing an independent method of measuring the beam activity.

Given its widespread adoption for displaying Prometheus metrics, Grafana was chosen to create the dashboard shown to the end user. An example dashboard used during data taking can be seen in figure 7.13. For metrics where history is important, such as the beam activity, a time series plot was used. For more instantaneous metric, such as RAM and disk usage, a gauge was used.

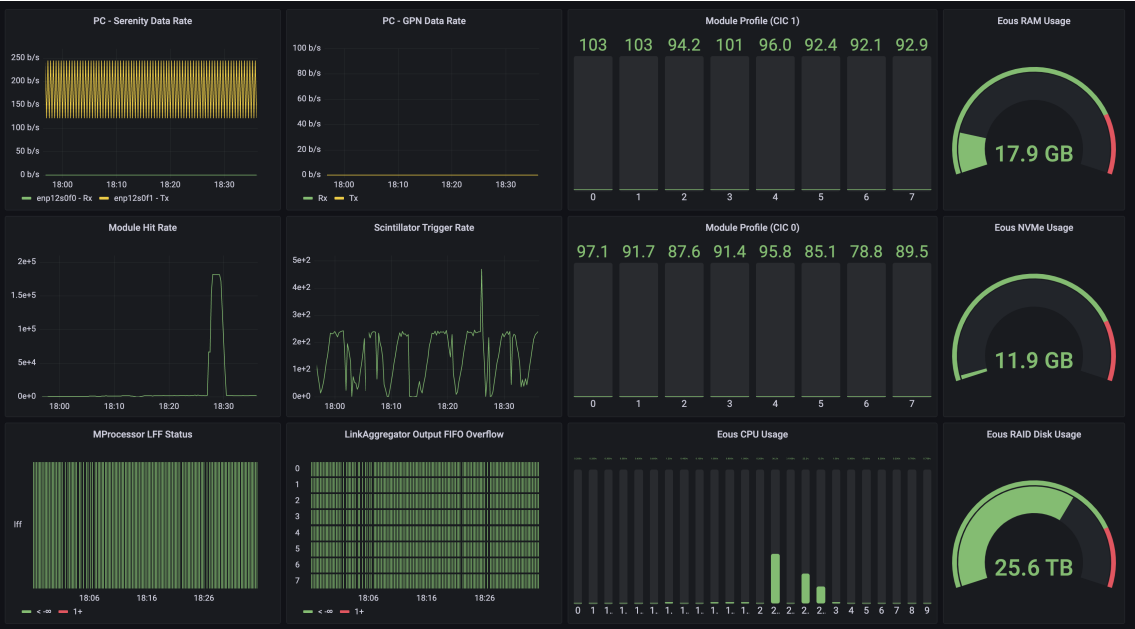


Figure 7.13: Example Grafana dashboard used during the beam test.

Chapter 8

Beam Test Analysis

The analysis of the data collected during the test beam has a primary focus of determining possibility of performing track fitting and quantifying the performance of this fitting. Before such a goal could be achieved, it was first necessary to understand the composition of the data collected, in terms of characteristics of both the beam itself and the events it produced. In particular, the distribution of the event multiplicities has a large impact on the performance of the tracking algorithm. Before any tracking may be performed, it must also be determined if the data collected from each module individually was synchronous, allowing then for tracking to be performed with the confidence that events sharing the same Bx ID originate from the same incident muon.

8.1 Data Taking Runs

In total, 33 runs were conducted during the beam test, with 27.1 TiB of data being recorded. 19 of the runs were data-taking, 6 were magnet scans, one V_{cth} scan, one Delay-Locked Loop (DLL) timing scan and 6 DAQ tests. As a result of the beam issues described in the previous chapter, the majority of the data taking runs occurred with either the beam centre away from the test stand or the modules misaligned in an attempt to locate the beam. Figure 8.1a shows the amount of data recorded for each run, as well as the size of the “skimmed” data set for each run, whereby any data packet which contained no stubs was deleted. A plot of the fraction of the original data in the skimmed data set can be seen figure 8.1b. The early runs can be seen to have particularly low data fractions, with a noticeable improvement starting with run 38, whereby the beam was aligned to be closer to the station. Run 47 has a particularly high data fraction as a result of this run

being a DAQ test with a low module threshold, leading to a high number of stubs being generated in the front end. The other run with a particularly high data fraction was run 49, which was the threshold scan. For high threshold values in the scan, a similarly large number of stubs would be produced, resulting in the high fraction of stub data in the recorded set.

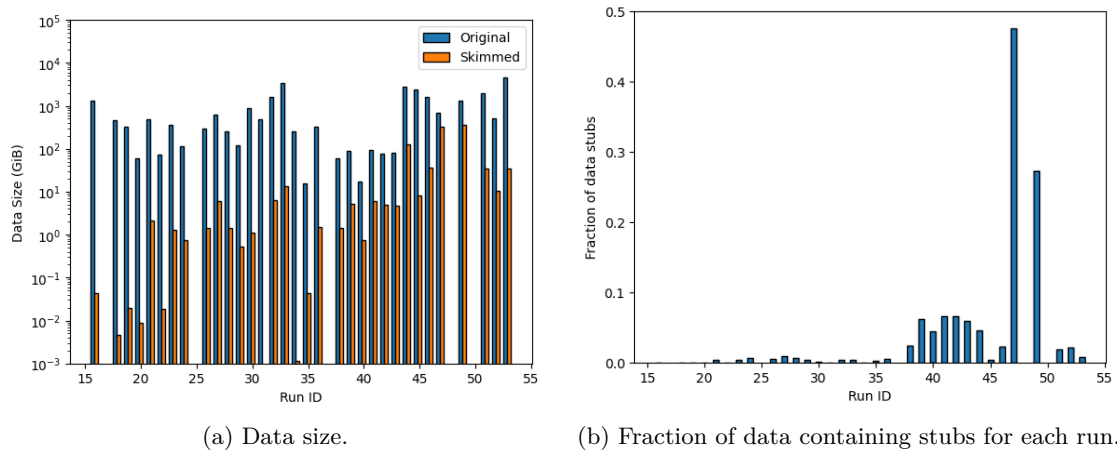


Figure 8.1: Plots showing the data size for each recorded run and the fraction of that data which contained valid stubs.

Of the 33 runs, two are considered “golden” runs for the measurement of synchronicity and track finding. The first is run 44, which is the only data-taking run where all modules were aligned and thus track finding would be possible. The second is run 46, which although has misaligned modules, contains a large number of stubs within the dataset, making the measurement of synchronicity between modules possible.

For the two scan runs, 49 for V_{cth} and 51 for DLL, the value of the variable being scanned was encoded into the IPBus user data bits sent in the event header.

8.2 Beam Characteristics

Before considering any advanced analysis such as track reconstruction, first the basic properties of the beam and thus the data recorded must be understood.

8.2.1 Intensity

One of the most fundamental measurements that can be made with the data gathered from the beam test is to plot the intensity of the beam with respect to position. This can be achieved by generating a histogram of the strip number in each module across the

full data set. Groups of 32 strips were combined into bins in order to smooth some of the statistical noise. An example intensity plot for run 46 can be found in figure 8.2. From these plots it can be seen that the intensity incident on modules 0 and 1 was higher than 2 and 3. For modules 2 and 3, there is also a gradient in intensity, suggesting that these modules were on the edge of the beam compared to 0 and 1.

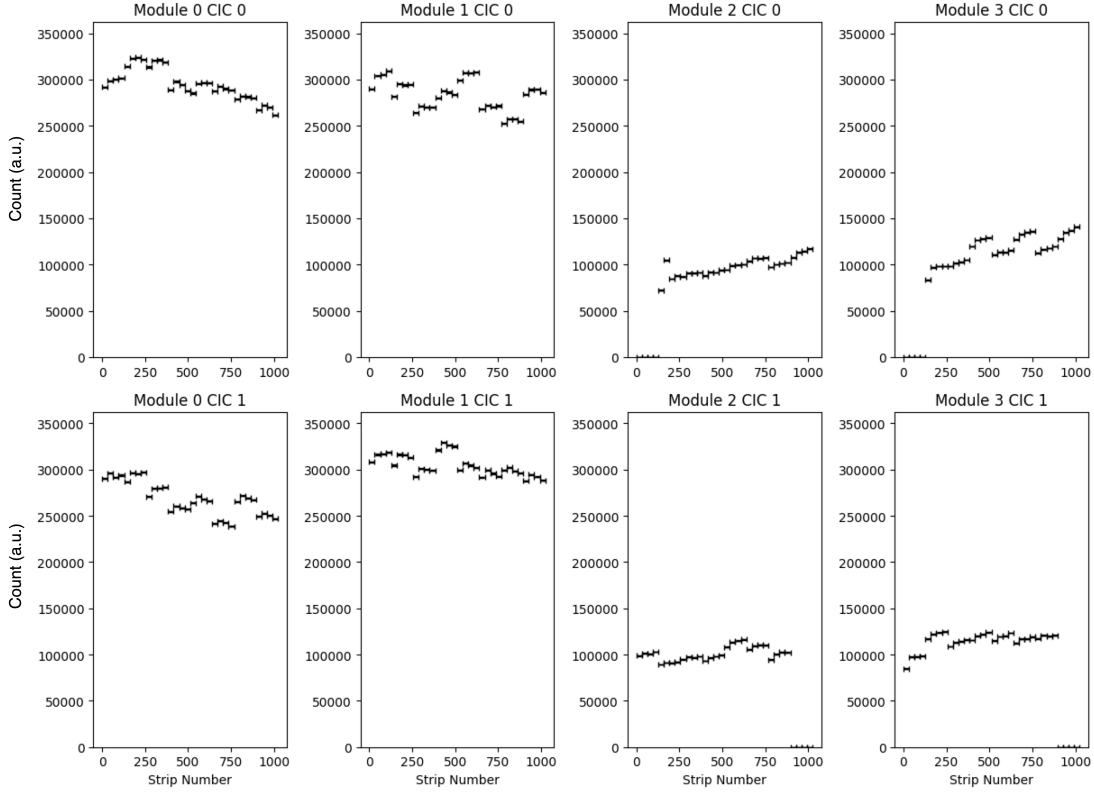


Figure 8.2: Histograms of hit count vs strip number, grouped by module. Note the empty bins modules 2 & 3, as a result of the last CBC in each CIC being disabled.

8.2.2 Time Dependence

In a similar method to the module intensity plots, the time dependence of intensity can be visualised by taking a histogram of the packet Super IDs. This property of the packet is unique within a run and smoothly incrementing, thus the number of hits within each super ID provides a good indication of the beam intensity. A plot of this histogram for run 46 can be found in figure 8.3. The time dependence of the plot shows how the beam varied during the course of an entire run, for instance if a parameter of one of the bending magnets was changed, which in turn affects the intensity of the beam incident on the modules. In the case of figure 8.3, the change moved the beam closer to the station, thus

increasing the intensity.

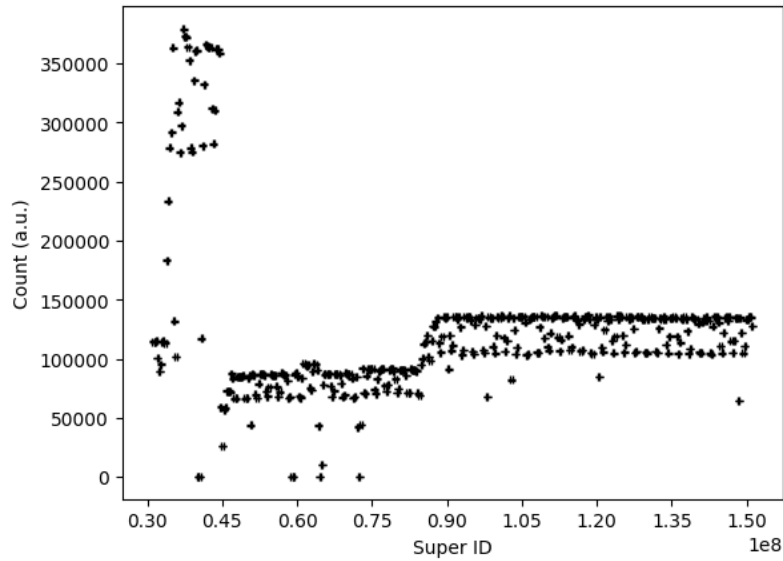


Figure 8.3: Plot showing a histogram of packet Super ID, which is analogous to time.

8.3 Event Characteristics

8.3.1 Event Size

When considering the properties of individual events within each run, a first metric to use is the size of the event. The distribution of the event sizes can provide insight into the processing bandwidth required to ensure that truncation is avoided or at worse limited to rare cases. This distribution can be seen in figure 8.4a. The same data, but grouped by CIC can be seen in figure 8.4b. The former plot shows that the maximum event size observed in run 46 was 20 stubs, which, if all events in a boxcar were that size, would be too large to be processed in its entirety within the MProcessor. If the other events with the boxcar are sufficiently small, then it can still be processed and sent over then link. Given that 99% of non-zero events have a size less than 4 stubs, this should not be a concern.

It should be noted however, that beam was not centred on the modules when this data was recorded, thus the intensity and hence data rate would have been lower compared to a more centred beam. Such a measurement will need to be repeated with a centred beam to better estimate whether truncation will exist in the system and if, to what level.

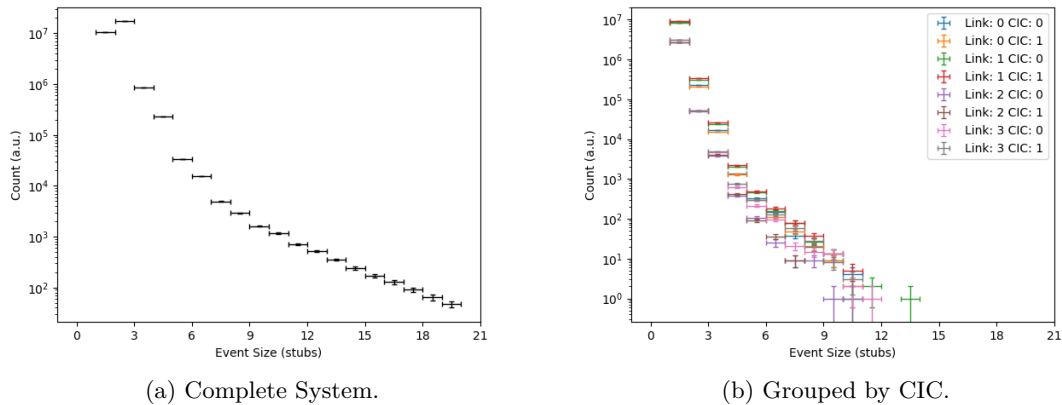


Figure 8.4: Plots showing histograms of event size for both the complete DAQ system and also grouped by each individual CIC.

8.4 Synchronisation and Alignment

Before any meaningful attempts can be made at reconstructing tracks between the multiple modules placed in the test beam, it is necessary to ensure that they are all synchronised. As all modules are connected to the same clock source, the Serenity card, the primary source of misalignment in time will be from the differences in fibre length between each of the modules. The fibre used to connect the modules 2 and 3 was longer than that which connected 0 and 1, meaning that the clock pulse will arrive at modules 2 and 3 slightly after the same pulse arrives at 0 and 1.

In order to test synchronisation between two modules, it is necessary to have an event happen on both modules at as close to the same point in time as possible. A muon with momentum 160 GeV travels at approximately 99.9999% of the speed of light, thus will be incident on each of the modules almost instantaneously. In the case of the two CMS modules, which were only centimetres apart, the difference in incident time between the particle passing through both modules is on the order of 0.1 ns, well below the timing resolution of the modules. For the most extreme set of modules in the test stand, the time of flight between them for a muon was 6 ns, again within the 25 ns window for recording an event, but more significant. In order to test this synchronisation, it was necessary to define a “coincident” event within the data. This was considered as a pair of hits, one in each of the modules under consideration, which had identical BxIDs and Super IDs. No positional constraints were put on the two hits. All events within the data could then be inspected to look for these coincident events. In order to measure synchronicity, it is also

necessary to consider coincident events with differing BxIDs. For instance, if there were an excess of events with a hit in the second module in the bunch crossing after the hit in the first module, it would suggest a misalignment in time. As such, the coincident count was conducted for multiple different offsets, from -5 to +5 Bx away from synchronous. The results of these counts can be seen in figure 8.5.

Before even normalising for time of flight, the peak of each combination of modules is at zero offset, suggesting good synchronisation. Particularly within the two pairs of modules there can be seen very good synchronicity. By taking the mean of the middle 3 bins, it can be seen that modules 0 and 1 have an average offset of just 0.2 ns, 2 and 3 have an average offset of 2.8 ns. Accounting for the time of flight for a muon to travel the 0.7 m between the station modules, the average offset is reduced to -0.2 ns, similar to that of the upstream modules.

Simply normalising for time of flight does not however, account for all of the offset between CMS modules and those in the station: there remains still an approximately 4-5 ns difference in the average offset between the two sets of modules (figure 8.6), which can most likely be explained by the aforementioned difference in fibre lengths between the two sets.

From this, it is clear that the modules are well synchronised, giving confidence that stubs generated in each of them on the same bunch crossing do indeed correspond physically to the same instant in time.

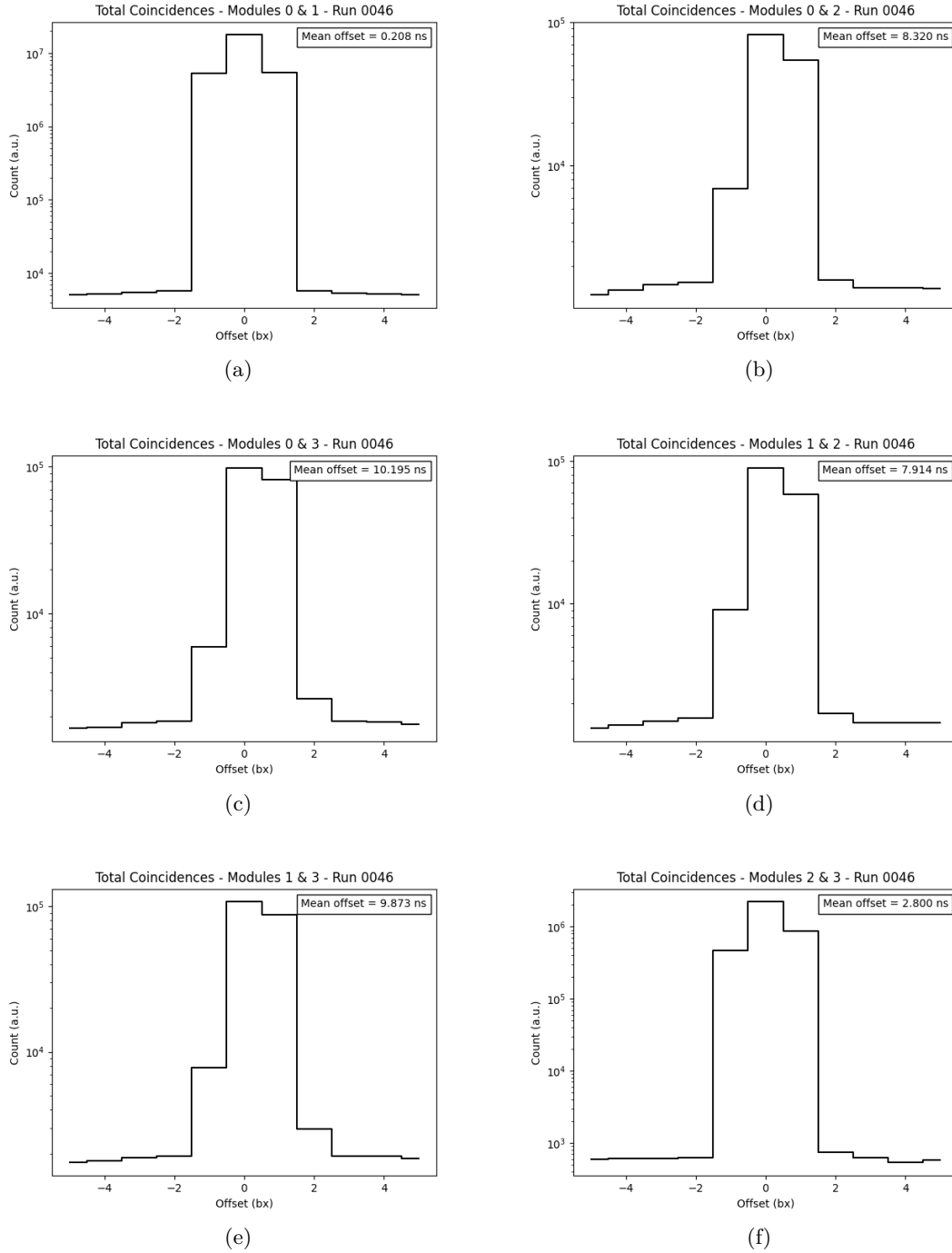


Figure 8.5: Plots showing histograms of synchronicity between each combination of modules in the test stand.

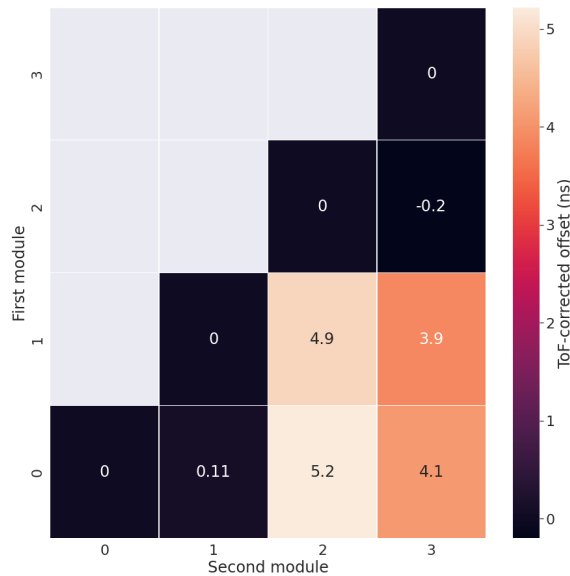


Figure 8.6: Heatmap showing the corrected average offsets after accounting for the time of flight differences.

8.5 Track Finding

8.5.1 Event Selection

In order to achieve any meaningful reconstruction of straight line tracks, at least one hit is required in each of at least three modules. Furthermore, so as to most accurately represent a full MUonE station, the three modules in which a hit must be present are modules 1, 2 and 3. These modules were approximately equidistant from each other, as the CMS test box was placed approximately a further 0.7 m upstream from the station. Hits in module 0, the furthest upstream, were ignored for track reconstruction as this module was not connected to the Serenity DAQ system for all runs.

8.5.2 Method

The method used to reconstruct events which met the criteria for tracks described above was designed to mirror as closely as possible the intended design for the online track reconstruction that would run on the FPGA using real-time data. This would allow for lessons learnt during development of the analysis code to be reflected in the design of the firmware. The algorithm used can be broadly split into two stages: pair construction and propagation, followed by intermediate hit association.

In the first stage, all possible combinations of hits in the first and third modules are created as stub pairs and stored in a list. For each of these pairs, the intersection of a

straight line passing through the pair of hits with the intermediate module is calculated. The gradient of this straight line can be calculated by taking the difference in position in the x -plane and dividing by the difference in position in z :

$$\frac{\Delta x}{\Delta z} = \frac{x_2 - x_0}{z_2 - z_0} \quad (8.1)$$

where variables subscripted by the number 0 belong to the first module of the triplet, and by 2 to the last. The intersection with the intermediate module can then be calculated by projecting this gradient by the distance between the first and intermediate modules and adding an offset from the first module:

$$x_{projection} = x_0 + \frac{\Delta x}{\Delta z} (z_1 - z_0) \quad (8.2)$$

By taking a coordinate system such that $z_0 = 0$, equation 8.2 can be simplified to

$$x_{projection} = x_0 + \frac{z_1}{z_2} (x_2 - x_0) \quad (8.3)$$

$$= x_0 \left(1 - \frac{z_1}{z_2}\right) + x_2 \frac{z_1}{z_2} \quad (8.4)$$

The ratio between z_1 and z_2 is fixed during operation, meaning both $\left(1 - \frac{z_1}{z_2}\right)$ and $\frac{z_1}{z_2}$ are constants that can be pre-computed. The projected intersection is therefore the sum of two linearly-scaled variables, an operation that be easily optimised to run on an FPGA.

For the initial analysis, all modules were assumed to each be spaced exactly 0.719m from each other and perfectly aligned in the x direction. In this approximation, equation 8.4 can be further simplified to:

$$x_{projection} = 0.5 \cdot x_0 + 0.5 \cdot x_2 \quad (8.5)$$

With the intersection point calculated for the pair of hits, all hits in the intermediate module are then compared against this intersection for those which may correspond to the track. A delta is calculated, by taking the absolute difference in x between the real hits and the intersection point:

$$\delta_x = |x_1 - x_{projection}| \quad (8.6)$$

The lowest delta and corresponding hit in the intermediate module are associated with the the track pair candidate. At the time of calculation of the delta, if it is below a threshold

window size, the track candidate is also considered “valid”. In the firmware design, only these “valid” tracks would be sent off the FPGA for further analysis.

8.5.2.1 Tilt Correction

The two modules housed within the MUonE station were tilted to an angle of 233 mrad. As such, the x and z coordinates of hits in these modules must be corrected, compared to a perpendicular module. Because θ_{tilt} is constant, both of the corrections are linear:

$$x \rightarrow x \cos \theta_{tilt} \quad (8.7)$$

$$z \rightarrow z \sin \theta_{tilt} \quad (8.8)$$

The correction to the intersection point of the projected track is more complex however. As shown in figure 8.7, the projected point is the intersection of the projected track with a tilted plane. $z_{projection}$ can be calculated as follows:

$$z_{projection} = x_{projection} \tan \theta_{tilt} \quad (8.9)$$

Thus equation 8.2 can be rewritten with this correction:

$$x_{projection} = x_0 + \frac{\Delta x}{\Delta z} (z_1 + z_{projection} - z_0) \quad (8.10)$$

$$= x_0 + \frac{\Delta x}{\Delta z} (z_1 + x_{projection} \tan \theta_{tilt} - z_0) \quad (8.11)$$

$$x_{projection} \left(1 - \frac{\Delta x}{\Delta z} \tan \theta_{tilt} \right) = x_0 + \frac{\Delta x}{\Delta z} (z_1 - z_0) \quad (8.12)$$

$$x_{projection} = \frac{x_0 + \frac{\Delta x}{\Delta z} (z_1 - z_0)}{1 - \frac{\Delta x}{\Delta z} \tan \theta_{tilt}} \quad (8.13)$$

Where Δx and Δz both include the tilted corrections also. Whilst both corrections are simple to apply in offline analysis, and are used for all proceeding results, only the correction in x can be easily used in an FPGA as the correction in z would require multiple division operations. In all track finding calculations detailed above, the x variables are only used for multiplications, however the separation between modules, and hence the z coordinates, are used for divisions. The latter operation is poorly optimised for FPGAs as it requires an iterative method to calculate. Division of an integer is also not closed, meaning that further care is required in order to limit the loss of accuracy.

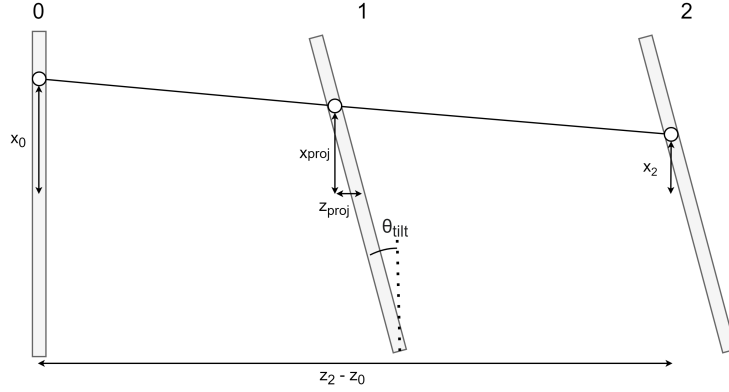


Figure 8.7: Diagram showing the corrections required for tilted modules. Modules 1 and 2 are parallel.

8.5.3 Intermediate Hit Deltas

Without the constraints of running in an FPGA, offline reconstruction can allow for all potential track candidates and deltas to be recorded. This was done for both runs 46 and 44, the former having the upstream module misaligned compared to the other two as a result of operational error, and the latter with all three aligned, and as such run 46 should have no possible valid tracks, as no straight line could pass through all three modules. A histogram of all calculated deltas can be seen in figure 8.8b.

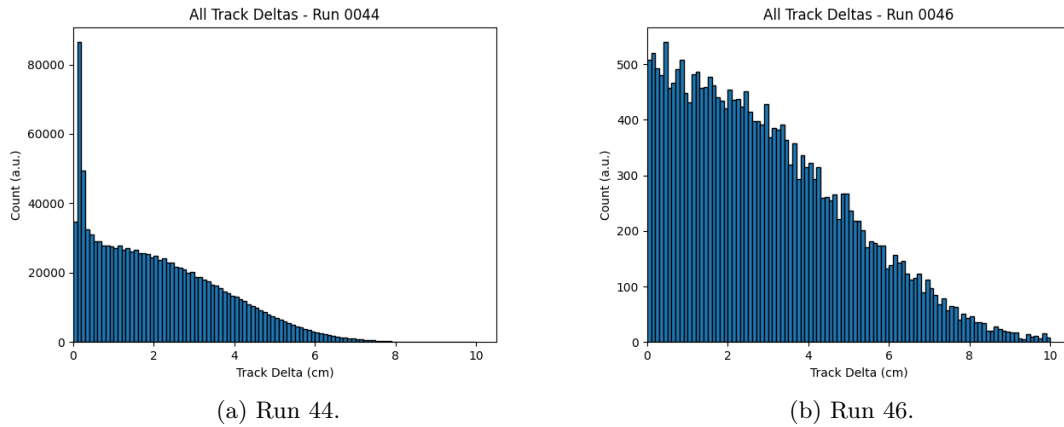


Figure 8.8: Plots showing histograms of track deltas for the two golden runs. The strong excess of hits close to zero for run 44 that does not exist in run 46 can be attributed genuine tracks.

Given the impossibility of genuine tracks within this sample, the deltas plotted should come from combinatorics of random hits across all three modules. In order to test this

theory and understand the distribution of the deltas, a toy Monte Carlo model was created to simulate random hits across three modules. The same track finding algorithm was then used to calculate the delta created by those three hits. A comparison between the Monte Carlo and real data can be seen in figure 8.9, which shows there is very good agreement between this model and the background data when modules are not aligned.

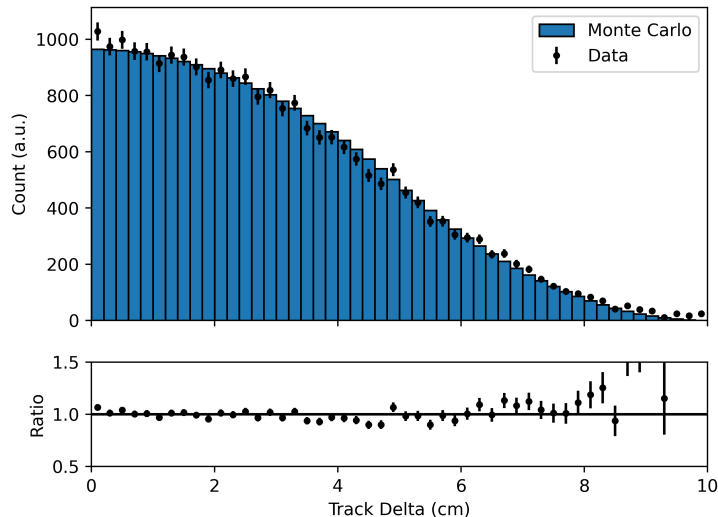


Figure 8.9: Ratio plot between the calculated track deltas for simulated Monte Carlo and real data in run 46.

Figure 8.8a shows the histogram of track deltas, but this time for run 44, which should contain genuine tracks. In this plot, it is possible to again see the smooth background of combinatorial noise, but also a strong excess above this background between 0 and 0.5 cm. This excess can be considered to have been generated by real tracks and thus provides an indication for choosing the acceptance window for valid tracks. By choosing a window size of 0.5 cm, it is possible to reduce the number of possible pair combinations down by a factor of five.

8.5.4 Residuals

Whilst the use of the absolute value of the delta is effective as a metric for determining the validity of a track and will be used in future implementations of tracking in an FPGA, simply the difference between the expected and real position can also provide insight into the nature of the tracks. These differences, or residuals, can be collected in a similar manner to the stub deltas and the distribution plotted in a histogram. As shown in figure

8.10a, the distribution can be both positive and negative, but shares many similarities with that of the stub deltas. The excess of hits can again be seen above a smooth combinatorial background. By taking a histogram of just the points close to the peak of this excess, it is possible to see that it is centred at a non-zero value. This suggests the existence of a misalignment in the three modules, as in a perfect system, the mean of the residuals would be at zero. The width of this peak gives the achievable resolution of the system for a track's position.

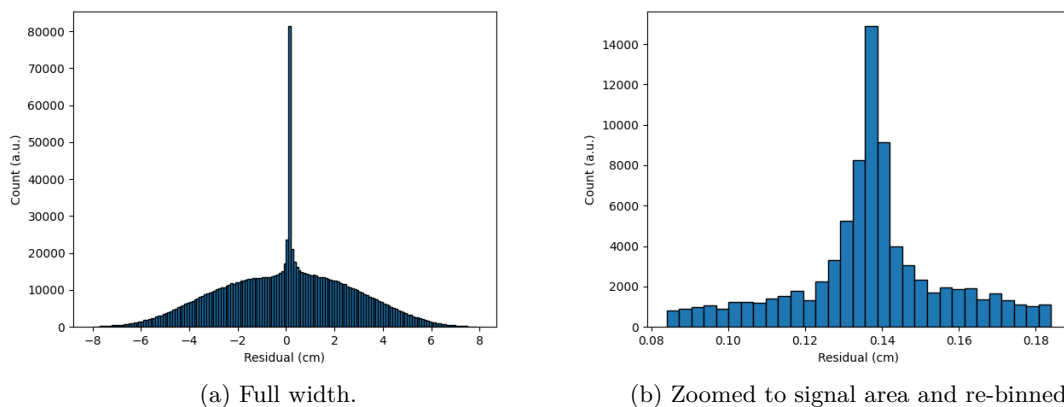


Figure 8.10: Plots showing histograms of track residuals, both for the full range of values and zoomed to the signal area.

Before attempting a fit to determine the best estimate of the mean and width of the signal distribution, first the background must be subtracted away. This was achieved by using the same Monte Carlo generator used for figure 8.9 and scaling the bins to the data outside of the signal region ($|r| > 1$). A ratio plot between data and Monte Carlo can be found in figure 8.11 and shows that there is good agreement between Monte Carlo and data for the background regions, meaning that this generator can be used to remove the background. The small asymmetry seen can likely be attributed to the binning of the histogram and that the two distributions were not fully aligned.

Subtraction was performed on a bin-by-bin basis, whereby the Monte Carlo and real data were both histogrammed with identical bins, with the Monte Carlo then subtracted from the real data. Once the background had been removed, a fit can then be made to the signal, which can be found in figure 8.12. In order to increase the quality of fit, a finer binning was used for the signal region. In performing this, it was noticed that the previously smooth peak separated into multiple peaks. The reason for these is the discrete

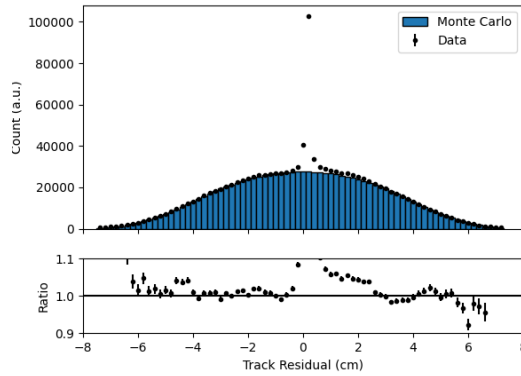


Figure 8.11: Ratio plot between the calculated track residuals for Monte Carlo and real data in run 44.

nature of the strips within the sensor module. This quantisation means that the residual is not continuous, which is reflected in the distribution.

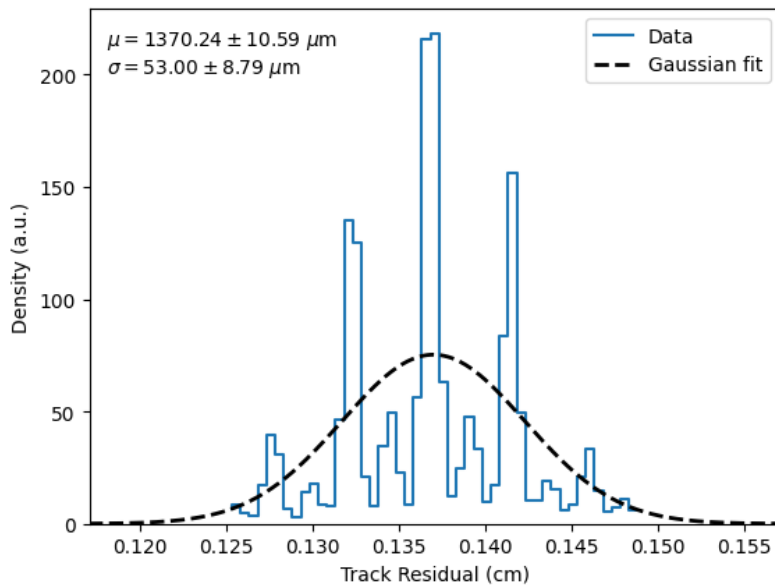


Figure 8.12: Plot showing the real data as a normalised histogram with the fitted Gaussian distribution overlaid. when fitted to a more coarsely-grained binning system.

With a Gaussian fit to the data, the mean and standard distribution of the fit can be extracted. The mean gives an indication of the misalignment of the system, whilst the standard deviation provides a measure of the track finding resolution.

Generating a simulated Monte Carlo background also provides an opportunity to estimate the fake rate for a given track finding acceptance window. This fake rate can be calculated as the number of Monte Carlo tracks within the acceptance window divided

by the total number of tracks recorded in data. Using this formula, the fake rate for an acceptance window of 0.5 cm was estimated to be 58.6%.

8.5.5 Multiplicity

A challenge in reconstructing tracks, particularly when attempting to do so online, will be the multiplicity of combinations of stubs in events. Given that each pair of hits in the outermost two modules must be used to generate a track candidate, this scales quadratically with the number of hits:

$$N_{pairs} = \#hits_0 \times \#hits_2 \quad (8.14)$$

Offline, each of these pairs is analysed sequentially, to do so in an FPGA would dramatically increase the latency of the process. Instead, each pair would be processed in parallel or a hybrid manner, using multiple blocks of identical logic, each processing a single pair. A limit must therefore be put on the maximum multiplicity that the reconstruction can handle, defined by the resource constraints of the FPGA. Events which produce more pairs than can be handled by the design would then have to be truncated or not processed at all and instead sent to be analysed offline with the event being flagged as too large for online processing.

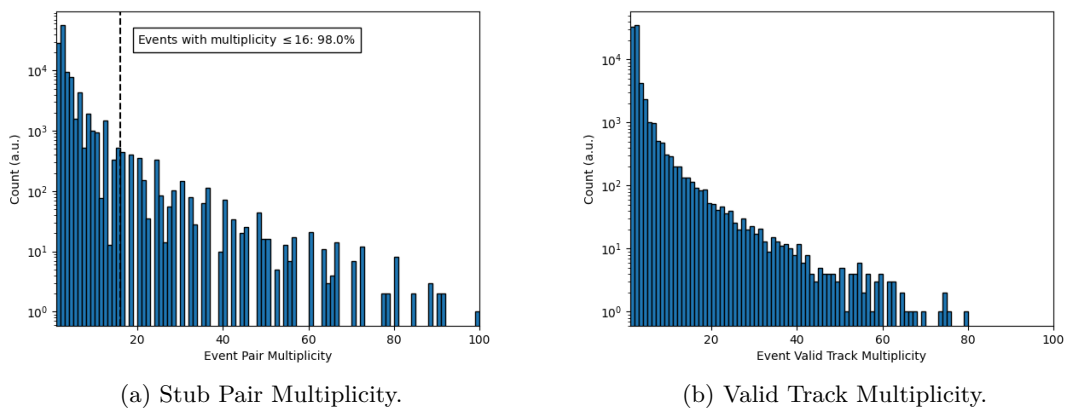


Figure 8.13: Plots showing histograms of multiplicity for both stub pairs and valid tracks in a given event. A track acceptance window of 0.5 cm was used to calculate valid tracks.

As shown in figure 8.13a, approximately 2% of events have multiplicities greater than 16, a value chosen as an estimate of the parallel capacity for online reconstruction and by looking at the multiplicity of valid tracks per event, it is clear that a significant number of events contain a large number of genuine tracks, rather than the cause of the high

multiplicity being random noise. An example of an event with high valid track multiplicity is illustrated by figure 8.14. In this example event, there seem to be a number of correlated hits across each of the three modules, potentially corresponding to a shower of particles produced by a scattering event upstream from the station. Such an event would not provide a clean signal for use in physics analysis, thus could likely be discarded.

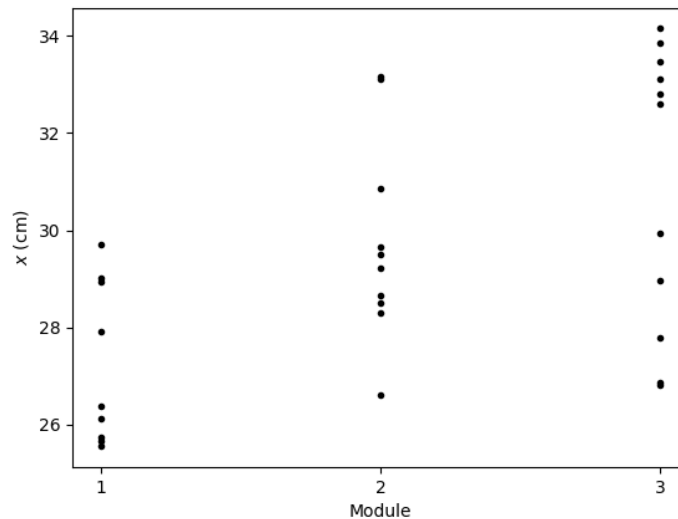


Figure 8.14: Scatter plot showing hit positions across each module for a high multiplicity event.

8.5.6 Angular Distribution

In order to calculate the intersection of a pair of hits with the intermediate module, the gradient $\frac{\Delta x}{\Delta z}$ must be calculated. Though not explicitly calculated in the formula used in hardware 8.4, it can be done so in the offline analysis of the data and then converted to an angle by taking the inverse tangent of the gradient:

$$\tan \theta = \frac{\Delta x}{\Delta z} \quad (8.15)$$

A histogram of the track angles for valid tracks can be found in figure 8.15. Given the maximum possible angle that could be reconstructed in the station was 0.05 rad, the plot shows that the majority of reconstructed tracks were on the edge of missing the station. This provides further evidence that the beam was not centred on the station during running and that greater control over beam position could have lead to improved statistics and results.

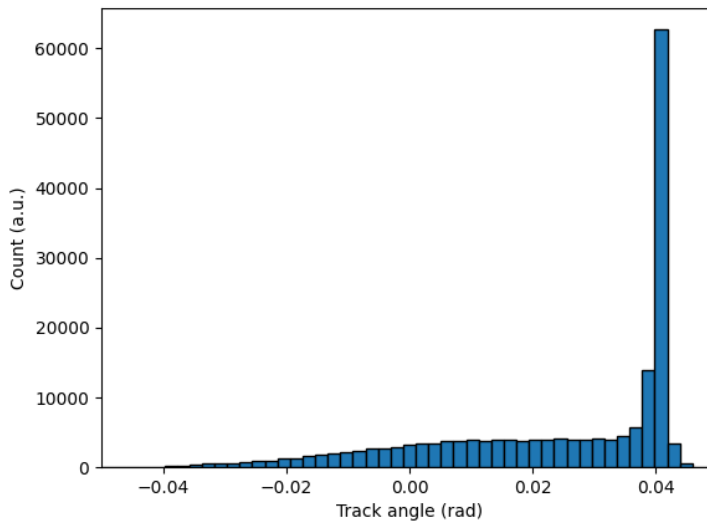


Figure 8.15: Histogram of the angular distribution of valid tracks in run 44.

8.6 Improvements to Track Finding Performance

8.6.1 Module Position Correction

Within the test beam setup, whilst the two modules located within the MUonE station were fixed into a known position, the CMS modules were not. As such, the exact position of these modules relative to the MUonE station can be estimated, but not known exactly. “Moving” module 1 can be achieved in the offline analysis by adding a correction in both x and z to the assumption made for initial track finding, which can then be repeated and compared with the original results. A diagram of the module position corrections can be seen in figure 8.16.

As shown in figure 8.8a, the peak of the excess track deltas is at a non zero value. In a perfectly aligned system, if this excess is generated by the genuine tracks created by muons, it should peak at a delta of zero, as the hits match a straight line created by the particle. A simple metric to measure changes in alignment against is therefore the sum of all deltas calculated during the track fitting. An improved alignment, and thus a movement of this peak towards zero will reduce the total sum of deltas, hence minimising the sum of the deltas should provide the optimum alignment of the modules in the two dimensions described above.

Taking the equation for the projection of the stub pair onto the intermediate module, equation 8.2, moving Module 1 further from the MUonE is equivalent to increasing the

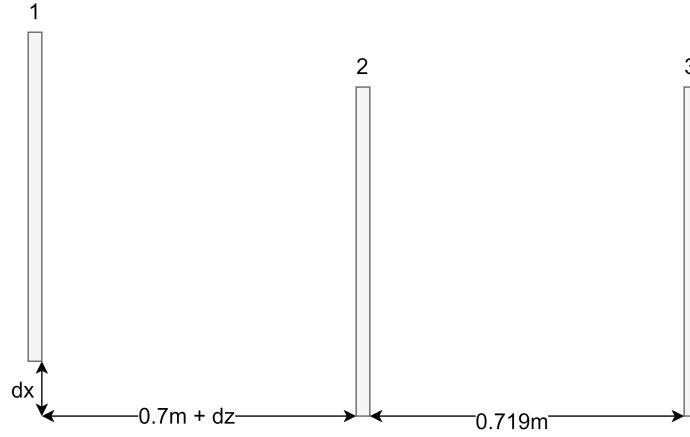


Figure 8.16: Diagram showing the position corrections made to improve the alignment of the modules. Positive dz is defined as increasing the separation between module 1 and module 2, whilst dx is defined as moving module 1 towards the Jura Mountains.

distances z_1 and z_2 by the same small amount, δz . The effect of this on equation 8.2 gives the following formula:

$$x_{projection} = x_0 + \frac{x_2 - x_0}{z_2 + \delta z - z_0} (z_1 + \delta z - z_0) \quad (8.16)$$

Which in the coordinate system where $z_0 = 0$, simplifies to:

$$x_{projection} = x_0 + \frac{z_1 + \delta z}{z_2 + \delta z} (x_2 - x_0) \quad (8.17)$$

$$x_{projection} = x_0 \left(1 - \frac{z_1 + \delta z}{z_2 + \delta z} \right) + x_2 \frac{z_1 + \delta z}{z_2 + \delta z} \quad (8.18)$$

Provided that δz is constant, equation 8.18 remains the sum of two constant multiplications.

A correction in the x -axis to module 1 can be achieved by the transformation $x_0 \rightarrow x_0 + \delta x$, yielding the following formula from equation 8.18:

$$x_{projection} = (x_0 + \delta x) \left(1 - \frac{z_1 + \delta z}{z_2 + \delta z} \right) + x_2 \frac{z_1 + \delta z}{z_2 + \delta z} \quad (8.19)$$

In order to understand the effect of these corrections, a two-dimensional parameter sweep was conducted, with the sum of the stub deltas calculated at each point. The results of this sweep can be found in figure 8.17.

The sweep shows that there is not a single point which minimises the stub deltas, but

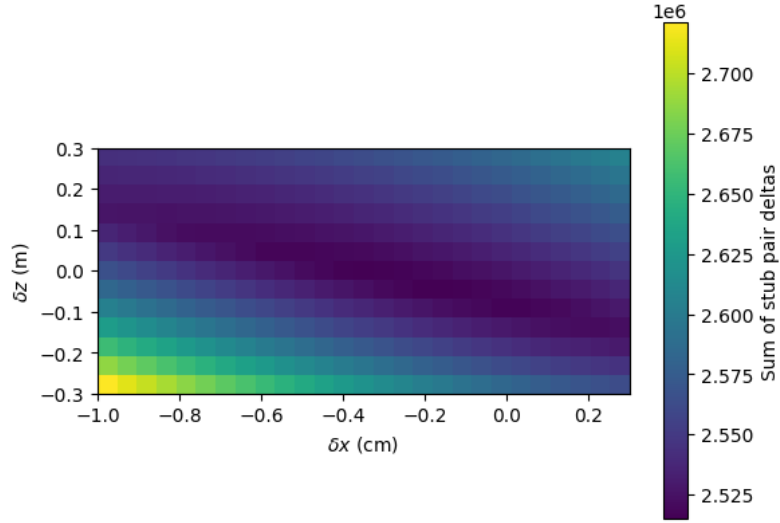


Figure 8.17: Two-dimensional heatmap showing the effect on the total sum of the stub deltas when moving the module in both x and z . Note the different units on each axis.

instead a line. This can be understood by considering that moving the module parallel to the direction of the tracks will have no effect of the position of the projected intersection and thus delta. If all genuine tracks are parallel to each other, then there is no focusing effect by moving the module along the beam line.

A secondary method for optimising the offset position of the module is to use the fitted standard deviation of the residuals. Minimising the standard deviation and thus achieving the greatest resolution in track finding will again indicate the most likely position of the module in the station. In particular, the relative separations of the modules will have the largest effect when using the standard deviation as the metric of merit, as this is where any focusing effects will be observed.

Figure 8.18 shows that moving the module in the z direction has a strong effect on the fitted standard deviation, and that there is a position where the modules are most focused. By performing a quadratic fit around the minimum point, the required z -correction can be quantitatively estimated. The value for the offset was found to be 4.52 ± 0.61 cm. By performing a similar scan in the x -direction, no significant effect on the resolution was seen. As such, the standard deviation of the residuals can be seen as an independent constraint on the z correction.

Whilst scanning in x gives no effect on the standard deviation of the residual peak, it has a strong effect on the mean, or centre, of it. In a system where the modules are perfectly aligned, it would be expected that the mean of the peak would lie at zero.

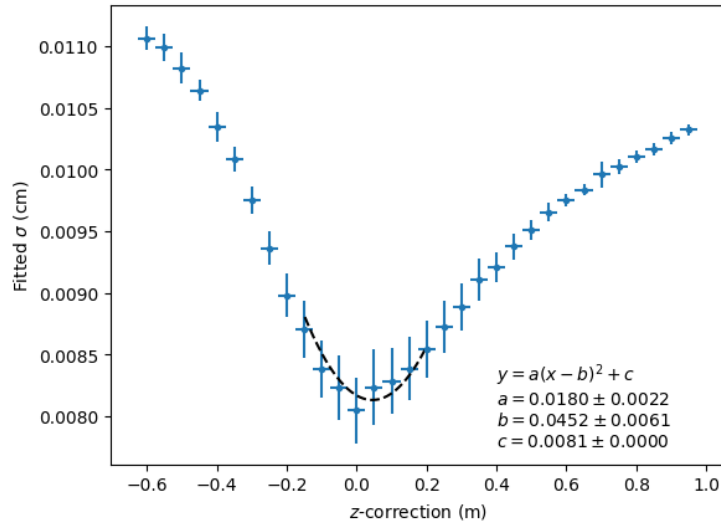


Figure 8.18: Plot showing binned fitted σ against the z correction to module 1. A quadratic fit is also applied around the minimum.

Figure 8.19 shows a plot of the peak mean against x -correction and it can clearly be seen that the two have a linear relationship. Fitting a straight line to the data and then calculating the intersection of this straight line with $\mu = 0$ gives an optimal x correction of -0.648 ± 0.000 cm.

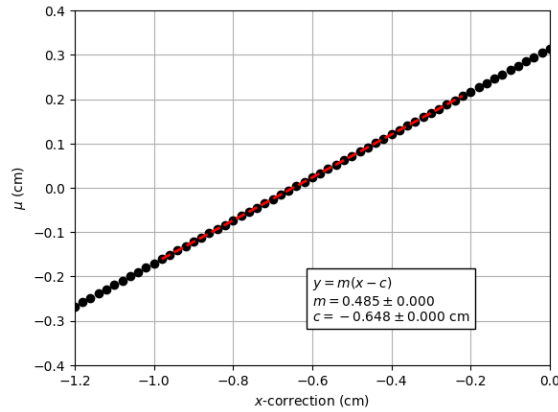


Figure 8.19: Plot showing fitted μ against the x correction to module 1. A linear fit is also applied to data close to $\mu = 0$.

With the information provided by the individual fits on the z and x corrections for the module, a number of the previously shown plots can be regenerated. For example, the peak in the stub delta plot (figure 8.8a), has shifted considerably towards zero, as shown in figure 8.20. This shift also allows for the window size used to determine valid tracks to be reduced, from 0.5cm to 0.015cm. This in turn increases the exclusion fraction from 80.6% to 95.1%, without removing any of the genuine tracks.

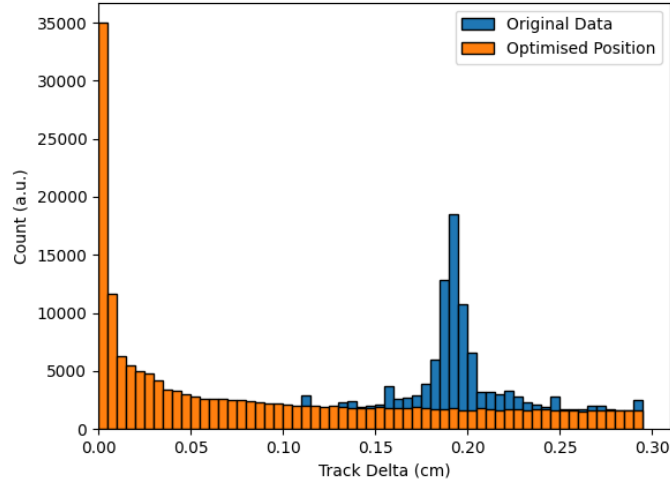


Figure 8.20: Histogram of the stub deltas for both the original data and with the corrected module position.

The effect of this cut can be seen clearly in the angular distribution of the tracks and the event multiplicity, shown in figure 8.21. The plot of angular distribution shows only a minimal reduction in tracks in the signal area, but a significant reduction in the background. It can thus be inferred that the cut is not simply reducing the number of tracks uniformly, but instead increasing the purity of the resulting tracks. The shift in the peak comes as a result of the moved module changing the angle of the stub pairs. The effect on the multiplicity is similarly strong, whereby no event has more than 9 valid tracks, down from 80 in the original data. The fake rate with the tighter acceptance window is also significantly improved, from 58.6% originally down to just 7.8%.

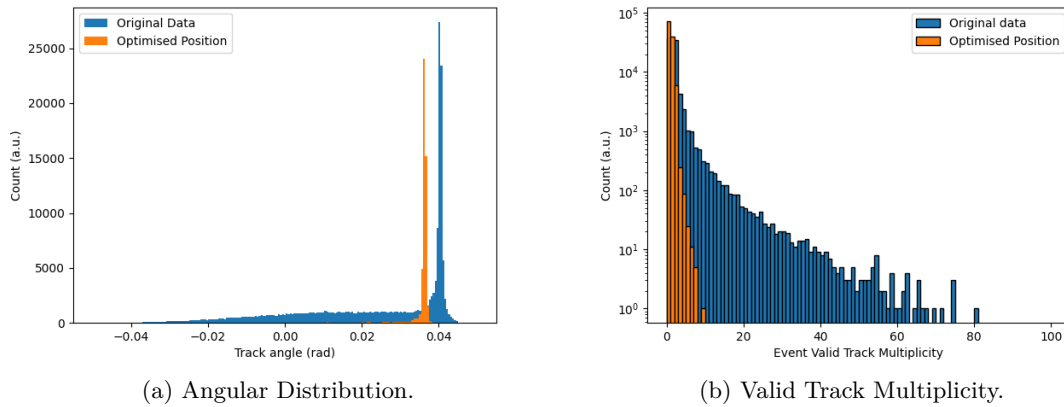


Figure 8.21: Plots showing histograms of track angle and event multiplicity, for both the original track generation and with the position optimised data.

The same fitting which was performed on the track residuals can also be repeated for the optimised data, yielding the results shown in figure 8.22. The fit to the overall peak is slightly improved compared to the original data, with a width of $48.39 \pm 3.05 \mu\text{m}$. Using the formula as derived in Appendix C:

$$\sigma_r = \sigma_m \sqrt{2R^2 + 2R + 2} \quad (8.20)$$

where in this case $R = 0.94$, the intrinsic module resolution can be estimated to be $39.5 \mu\text{m}$. It can also be noted in figure 8.22a that the peak is comprised of multiple sub peaks, each with a smaller width than the overall. The centermost peak has a fitted width of just $15.03 \pm 1.32 \mu\text{m}$, which equates to an approximate module resolution of $12.2 \mu\text{m}$.

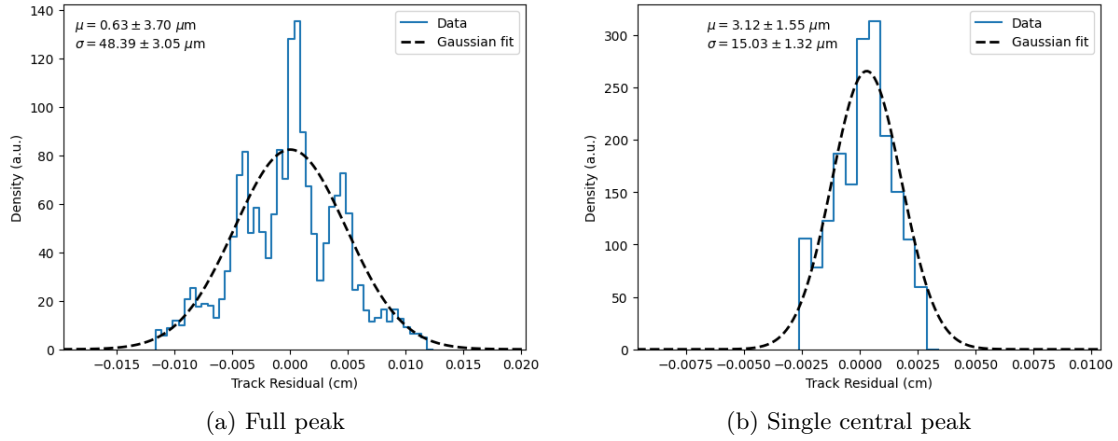


Figure 8.22: Plots showing fitted histograms of the track residuals. In both plots, the combinatorial background has been subtracted before fitting.

8.6.2 Stub Bend

In the analysis presented above, a variable provided in the stub data but not used for track finding is the bend. The bend, a term derived from the use of the sensor modules in the CMS tracker, corresponds to the offset between the hit in the first layer of the sensor compared to the second. Given the absence of a magnetic field in the test beam to bend the incident particles, the calculated bend can instead be thought of as the angle through which the particle traverses the module. Given a sensor separation of 1.8 mm and a strip

pitch of $90\text{ }\mu\text{m}$, the angle can be derived from the bend by the following formula:

$$\tan \theta = \frac{\text{bend} \times 90 \times 10^{-6}}{1.8 \times 10^{-3}} \quad (8.21)$$

$$\theta = \tan^{-1}(\text{bend} \times 0.05) \quad (8.22)$$

Given the dimensions of the test stand, the magnitude of the maximum possible bend for a straight track can also be calculated:

$$\frac{0.1}{1.425} \cdot \frac{1.8 \times 10^{-3}}{90 \times 10^{-6}} = 1.40 \text{ strips} \quad (8.23)$$

Thus any stub with bend greater than this value can be removed as they would not correspond to a genuine track as the angle would be greater than the acceptance of the system. To understand whether this cut will have an effect on track finding, first a histogram of the bend values for each module was taken and can be seen in figure 8.23. By taking module 1 as an example, it can be seen that the majority, 95.3%, of stubs lie within $|\text{bend}| < 1.4$.

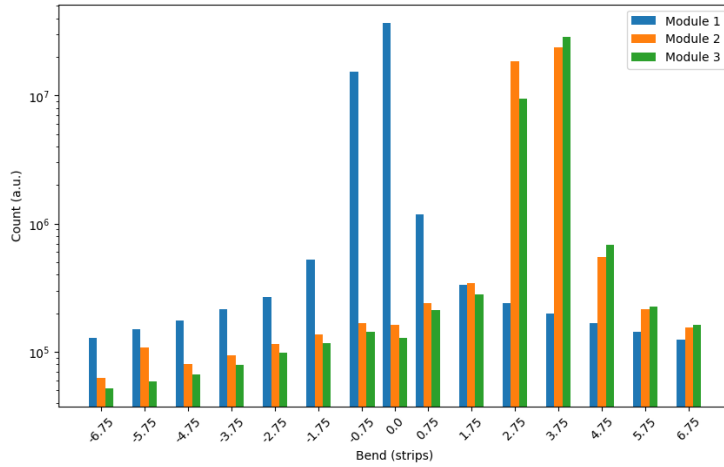


Figure 8.23: Histogram of the stub bend for run 44. The bend can only take the discrete values shown on the x -axis. Note also the offset in the peak of the bend for the tilted modules 2 and 3, which must be corrected for before any cuts can be made.

For modules 2 and 3, which are tilted, a corrective offset of 233 mrad , corresponding to 4.75 strips, must first be applied to re-centre the bends before the cut can be made.

Given a cut on bend can be performed on each stub independently, this operation can happen before track fitting and has the possibility to reduce the number of pair combinations required to be calculated for each event. The results of the bend cut on angular distribution and pair multiplicity can be seen in figure 8.24. The bend cut results in

a noticeable decrease in the number of tracks, particularly in the signal region, suggesting that some the hits used in these tracks may have inaccurate bends as the bend given by the stub would correspond to an incident particle traversing the system outside its acceptance. The cut also reduces the pair multiplicity, with the fraction of events with a multiplicity greater than 16 decreases from 2% to 1.4%.

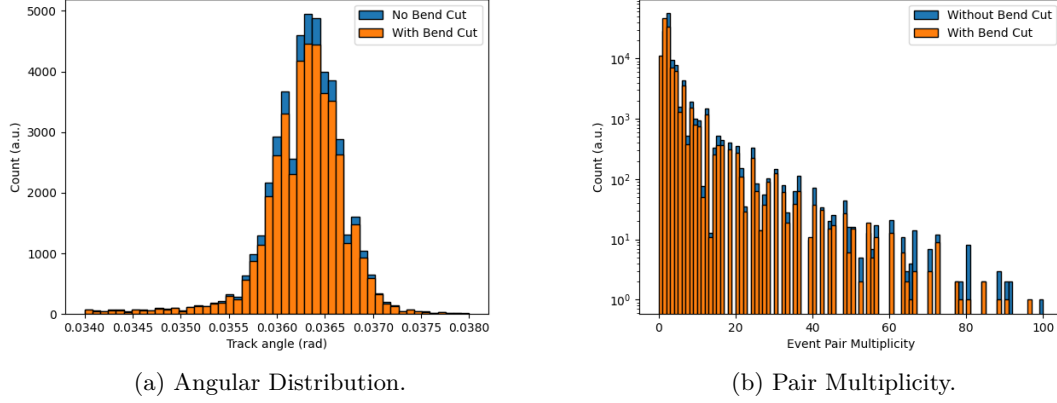


Figure 8.24: Plots showing histograms of track angle, zoomed to the signal region, and pair multiplicity, for both without and with the bend cut applied.

Chapter 9

Conclusions

The results of the test beam provide the culmination of work completed both for CMS and MUonE. The successful readout of module data at the full 40 MHz rate was possible through the development of DTC firmware to communicate with the FE modules and the extensive testing conducted prior to the beam test. The monitoring and control systems employed during the beam test built upon the work done in developing an online software stack for the Phase-II CMS tracker.

To show that it is indeed possible to link all pieces in the chain from module sensor through to the output of the Serenity board, all at a 40 MHz rate, is an important step towards the complete tracker system for CMS. Similarly, the ability to read data at the full rate is necessary for the MUonE experiment to collect sufficient statistics over a reasonable data-taking period.

The synchronous readout of multiple modules had not been attempted in any previous beam tests, thus the results described above, in particular those shown in figure 8.5 are the first indication that this functionality can be achieved. The slight asynchronicity between the modules located within the station and those outside also highlights the need to carefully consider fibre lengths and other such factors during calibration of a system containing these modules.

The stub efficiency result can also be compared against that of the previous beam test, with generally good agreement between both (figure 9.1). The slight reduction in efficiency is most likely caused by two factors: the methodology for calculating the efficiency and timing effects due to the asynchronous beam.

It is possible to consider the effect of the measured sensor resolution on the physics goals of the two experiments. Good sensor resolution, and by extension good vertex resolution

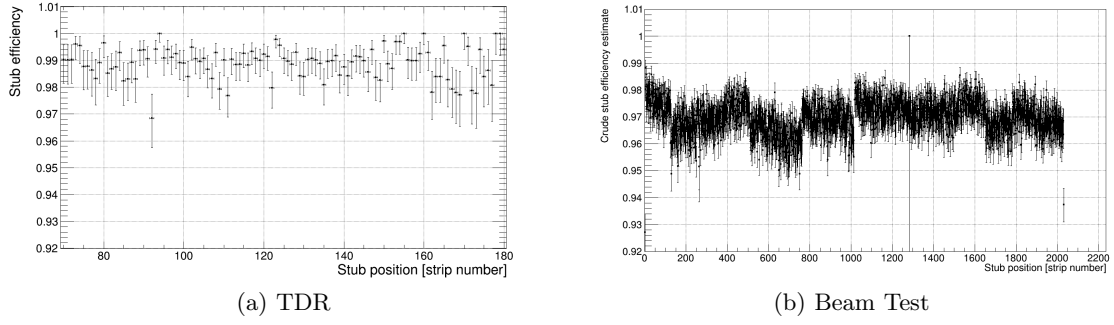


Figure 9.1: Plots showing the measured efficiency given in both the Tracker TDR[21] and this beam test.

of tracks, will be necessary for the successful running of the L1T at CMS. Poor resolution will mean that it will not be possible to reduce the trigger rate sufficiently at Level-1 and thus downstream triggers will be similarly saturated. For MUonE, it has been shown that sensor resolution, and by extension the possible angular resolution achievable, will have a significant effect on the ability to measure the angular distribution of the scattered electrons and muons. Figure 9.2 shows how improving from a resolution of $40\text{ }\mu\text{m}$ to $10\text{ }\mu\text{m}$ has a tangible effect on the angular distributions that can be measured. The results shown from this beam test, in particular figure 8.22, show that it may be possible to achieve a sensor resolution in the order of $10\text{ }\mu\text{m}$, if the selection of tracks with residuals making up only the central peak can be done. It will be the aim of future beam tests and subsequent analyses to investigate the possibility of this, as the sensor resolution and in turn the accuracy of the angular distribution has an upstream effect on the physics that can be investigated by the MUonE experiment. In particular, the running time required for a statistically significant result will increase with a poor resolution, as more data will be required to perform a fit on the distributions. The exact effect on the completion time of the experiment would require a full analysis and is thus beyond the scope of this thesis.

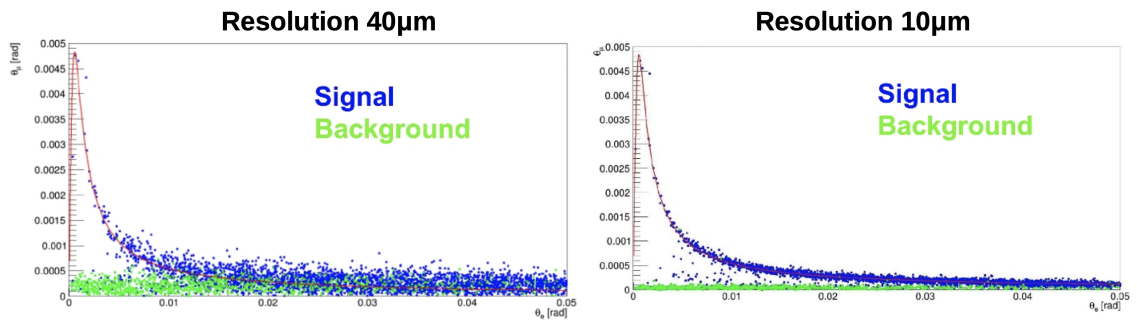


Figure 9.2: Plots of the angular distribution for simulated muon and electron elastic collisions at different sensor resolutions. Data was generated using the GEANT4[68] toolkit.

Bibliography

- [1] L. Evans and P. Bryant, *LHC Machine*, *Journal of Instrumentation* **3** (2008), ISSN: 17480221 (cit. on p. 21).
- [2] S. Chatrchyan et al., *The CMS experiment at the CERN LHC*, *Journal of Instrumentation* **3** (2008), ISSN: 17480221 (cit. on pp. 21, 26).
- [3] E. Lopienska, *The CERN accelerator complex, layout in 2022.*, (2022), URL: <https://cds.cern.ch/record/2800984> (cit. on p. 22).
- [4] M. Banner et al., *Observation of single isolated electrons of high transverse momentum in events with missing transverse energy at the CERN pp collider*, *Physics Letters B* **122** (1983), ISSN: 03702693 (cit. on p. 22).
- [5] G. Arnison et al., *Experimental observation of isolated large transverse energy electrons with associated missing energy at $s=540$ GeV*, *Physics Letters B* **122** (1983), ISSN: 03702693 (cit. on p. 22).
- [6] P. Bagnaia et al., *Evidence for $Z0 \rightarrow e+e-$ at the CERN pp collider*, *Physics Letters B* **129** (1983), ISSN: 03702693 (cit. on p. 22).
- [7] G. Arnison et al., *Experimental observation of lepton pairs of invariant mass around 95 GeV/c² at the CERN SPS collider*, *Physics Letters B* **126** (1983), ISSN: 03702693 (cit. on p. 22).
- [8] V. Duviol, “Cross section of LHC dipole. Dipole LHC: Coupe transversale.”, 2001, URL: <http://cds.cern.ch/record/843195> (cit. on p. 23).
- [9] J. Wenninger, *Machine protection and operation for LHC*, in proceedings of the *2014 Joint International Accelerator School: Beam Loss and Accelerator Protection, Proceedings*, 2014 (cit. on p. 24).

- [10] S. Chatrchyan et al., *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*, *Physics Letters, Section B: Nuclear, Elementary Particle and High-Energy Physics* **716** (2012) 30–61, ISSN: 03702693 (cit. on p. 24).
- [11] G. Aad et al., *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, *Physics Letters, Section B: Nuclear, Elementary Particle and High-Energy Physics* **716** (2012) 1–29, ISSN: 03702693 (cit. on p. 24).
- [12] V. Khachatryan et al., *Precise determination of the mass of the Higgs boson and tests of compatibility of its couplings with the standard model predictions using proton collisions at 7 and 8 TeV*, *European Physical Journal C* **75** (2015), ISSN: 14346052 (cit. on p. 26).
- [13] A. T. Margarian et al., *Interactive Slice of the CMS detector*, Conf. Proc., Technical Design Report CMS **C590914** (June 2016) 599, ed. by A. Tapper, URL: <https://cds.cern.ch/record/2205172%20http://cds.cern.ch/record/706847%20https://cds.cern.ch/record/1556311> (cit. on p. 27).
- [14] I. Neutelings, *CMS coordinate system*, 2021, URL: https://tikz.net/axis3d_cms/ (cit. on p. 27).
- [15] J. G. Layter, *The CMS muon project: Technical Design Report*, Technical design report. CMS, Geneva: CERN, 1997, URL: <https://cds.cern.ch/record/343814> (cit. on p. 29).
- [16] G. L. Bayatyan et al., *CMS TriDAS project: Technical Design Report, Volume 1: The Trigger Systems*, Technical Design Report CMS, CERN, 2000 599, URL: <http://cds.cern.ch/record/706847> (cit. on p. 30).
- [17] G. L. Bayatyan et al., *CMS computing: Technical Design Report*, Technical design report. CMS, Geneva: CERN, 2005, URL: <https://cds.cern.ch/record/838359> (cit. on p. 29).
- [18] B. Goddard et al., *Preliminary Report on the Consequences of LHC Civil Engineering for the SPS and LEP*, tech. rep., 1996 (cit. on p. 31).
- [19] D. E. Groom, *Passage of particles through matter*, *European Physical Journal C* **3** (1998), ISSN: 14346044 (cit. on p. 34).

-
- [20] G. Bianchi, *TkLayout: A design tool for innovative silicon tracking detectors*, *Journal of Instrumentation* **9** (2014), ISSN: 17480221 (cit. on p. 38).
- [21] The CMS Collaboration, *The Phase-2 Upgrade of the CMS Tracker*, tech. rep., Geneva: CERN, June 2017, URL: <https://cds.cern.ch/record/2272264> (cit. on pp. 40, 156).
- [22] M. Raymond et al., *The CMS binary chip for microstrip tracker readout at the SLHC*, en, *Journal of Instrumentation* **7** (Jan. 2012) C01033, Publisher: IOP Publishing, ISSN: 1748-0221, URL: <https://iopscience.iop.org/article/10.1088/1748-0221/7/01/C01033/meta> (visited on 04/27/2021) (cit. on p. 41).
- [23] R. Larsen, *Introduction to MicroTCA*, SLAC MicroTCA Standards Review (dated Jun. 4-5, 2012) (2012) (cit. on p. 45).
- [24] G. Iles, J. Jones, and A. Rose, *Experience powering Xilinx Virtex-7 FPGAs*, *Journal of Instrumentation* **8** (2013), ISSN: 17480221 (cit. on p. 45).
- [25] PICMG, *AdvancedTCA® Base Specification*, tech. rep., Mar. 2008, URL: <https://www.picmg.org/product/advancedtca-base-specification/> (cit. on p. 45).
- [26] S. Ahmad et al., *A 16-nm Multiprocessing System-on-Chip Field-Programmable Gate Array Platform*, *IEEE Micro* **36** (2016), ISSN: 02721732 (cit. on p. 45).
- [27] M. Hansen and J. Hegeman, *Proposed CMS Phase-2 ATCA rack layouts*, tech. rep., Sept. 2021, URL: <https://edms.cern.ch/document/2429324/1> (cit. on p. 46).
- [28] A. Rose et al., *Serenity: An ATCA prototyping platform for CMS Phase-2*, in proceedings of the *Proceedings of Science*, vol. 343, 2018 (cit. on p. 46).
- [29] Samtec, *FireFly™ Application Design Guide*, tech. rep., May 2021, URL: <http://suddendocs.samtec.com/ebrochures/firefly-brochure.pdf> (cit. on p. 46).
- [30] AMD Xilinx, *UltraScale™ Architecture and Product Data Sheet: Overview*, tech. rep., Nov. 2022, URL: <https://docs.xilinx.com/v/u/en-US/ds890-ultrascale-overview> (cit. on p. 47).
- [31] AMD Xilinx, *7 Series FPGAs Data Sheet: Overview*, tech. rep., Sept. 2020, URL: https://docs.xilinx.com/v/u/en-US/ds180_7Series_Overview (cit. on p. 47).
- [32] PICMG, *COM Express Module Base Specification*, tech. rep., July 2005, URL: <https://www.picmg.org/product/com-express-module-base-specification/> (cit. on p. 47).

- [33] A. Albert et al., *The apollo ATCA platform*, in proceedings of the *Proceedings of Science*, vol. 370, 2019 (cit. on p. 47).
- [34] C. Amstutz et al., *An FPGA-based track finder for the L1 trigger of the CMS experiment at the high luminosity LHC*, Real Time Conference (RT), 2016 IEEE-NPSS, IEEE, 2016 pp. 1–9 (cit. on pp. 49, 51).
- [35] E. Bartz et al., *FPGA-based tracking for the CMS Level-1 trigger using the tracklet algorithm*, *Journal of Instrumentation* **15** (2020), ISSN: 17480221 (cit. on pp. 49, 51).
- [36] The CMS Collaboration, *The Phase-2 Upgrade of the CMS Level-1 Trigger*, tech. rep., Geneva: CERN, Apr. 2020, URL: <http://cds.cern.ch/record/2714892> (cit. on p. 50).
- [37] C. G. Larrea et al., *IPbus: a flexible Ethernet-based control system for xTCA hardware*, *Journal of Instrumentation* **10** (2015) (cit. on pp. 52, 59).
- [38] B. McLaughlin and S. Perrott, “Amazon Simple Storage Service (S3)”, *AWS Certified SysOps Administrator Study Guide, 2E*, 2020 (cit. on p. 53).
- [39] M. J. Scheepers, *Virtualization and Containerization of Application Infrastructure : A Comparison*, 21st Twente Student Conference on IT (2014) (cit. on p. 55).
- [40] Mark Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*, 2011 (cit. on p. 55).
- [41] L. Ardila et al., *Specification of the Phase-2 Tracker Backend Electronics*, tech. rep., Feb. 2020 (cit. on p. 56).
- [42] S. Bologna et al., *SWATCH: Common software for controlling and monitoring the upgraded CMS Level-1 trigger*, in proceedings of the *Journal of Physics: Conference Series*, vol. 898, 3, 2017 (cit. on p. 60).
- [43] F. Akgul, *ZeroMQ*, Packt Publishing, 2013, ISBN: 178216104X (cit. on p. 60).
- [44] Sadayuki Furuhashi, *MessagePack*, Apr. 2013, URL: <https://msgpack.org/index.html> (cit. on p. 60).
- [45] The PostgreSQL Global Development Group, *What Is PostgreSQL?*, Postgresql.Org (2021) (cit. on p. 62).

-
- [46] R. Bogachev and N. Zarikovskaya, *Overview of the capabilities of the FastAPI web framework for implementing the backend of web systems*, in proceedings of the *Proceedings of the All-Russian Scientific Conference on Achievements of Science and Technology (DNiT-2021)*, Krasnoyarsk Science and Technology City Hall, 2021 pp. 289–293, ISBN: 9785604516539 (cit. on p. 62).
- [47] Django Software Foundation, *The Web framework for perfectionists with deadlines / Django*, 2018 (cit. on p. 62).
- [48] S. bin Uzayr, N. Cloud, and T. Ambler, “Vue.js”, *JavaScript Frameworks for Modern Web Development*, 2019 (cit. on p. 62).
- [49] B. Rabenstein and J. Volz, *Prometheus: A Next-Generation Monitoring System (Talk)*, Dublin: USENIX Association, May 2015 (cit. on pp. 64, 129).
- [50] A. Marinos, *Announcing balenaEngine: a container engine for IoT based on Moby technology from Docker*, balena.io () (cit. on p. 65).
- [51] E. A. Brewer, *Kubernetes and the path to cloud native*, Proceedings of the sixth ACM symposium on cloud computing, 2015 pp. 167–167 (cit. on p. 65).
- [52] J. Mendez, *GBT-FPGA Tutorial*, FPGA Working Group, Sept. 28, 2016, URL: https://indico.cern.ch/event/489996/contributions/2291863/attachments/1345764/2028939/GBTTutorial_-_TWEPP2016.pdf (cit. on p. 71).
- [53] L. Caponetto et al., *CIC1 technical specification*, URL: https://espace.cern.ch/Tracker-Upgrade/Electronics/CIC/Shared Documents/Specifications/CIC_specs_v3.pdf (2018) (cit. on p. 72).
- [54] *Tracker Layout repository*, 2022, URL: <http://cms-tklayout.web.cern.ch/cms-tklayout/layouts-work/recent-layouts/> (cit. on p. 80).
- [55] G. Badaro et al., *The Phase-2 Upgrade of the CMS Data Acquisition*, *EPJ Web of Conferences* **251** (2021) (cit. on p. 85).
- [56] The CMS Collaboration, *CMS Offline Software*, URL: <https://github.com/cms-sw/cmssw> (cit. on p. 87).
- [57] M. Pesaresi et al., *The FC7 AMC for generic DAQ & control applications in CMS*, *Journal of Instrumentation* **10** (2015), ISSN: 17480221 (cit. on p. 90).
- [58] K. S. Krane, D. Halliday, et al., *Introductory nuclear physics*, vol. 465, Wiley New York, 1988 (cit. on p. 98).

- [59] P. Zyla et al., *Review of Particle Physics*, **PTEP** **2020** (2020) 083C01 (cit. on p. 98).
- [60] R. Bracewell, *The Fourier Transform & Its Applications*, English, 3rd edition, Boston: McGraw-Hill Science/Engineering/Math, June 1999, ISBN: 978-0-07-303938-1 (cit. on p. 101).
- [61] B. Abi et al., *Measurement of the Positive Muon Anomalous Magnetic Moment to 0.46 ppm*, **Physical Review Letters** **126** (2021), ISSN: 10797114 (cit. on p. 107).
- [62] G. Abbiendi, *Letter of Intent: the MUonE project*, tech. rep., Geneva: CERN, June 2019, URL: <https://cds.cern.ch/record/2677471> (cit. on pp. 109–112).
- [63] S. Eidelman and F. Jegerlehner, *Hadronic contributions to $(g-2)$ of the leptons and to the effective fine structure constant $\alpha(MZ2)$* , **Zeitschrift für Physik C Particles and Fields** **67** (1995), ISSN: 01709739 (cit. on p. 109).
- [64] F. Jegerlehner, *The running fine structure constant $\alpha(E)$ via the Adler function*, **Nuclear Physics B - Proceedings Supplements** **181-182** (2008), ISSN: 09205632 (cit. on p. 109).
- [65] W. Adam et al., *Beam test performance of prototype silicon detectors for the Outer Tracker for the Phase-2 Upgrade of CMS*, **Journal of Instrumentation** **15** (2020), ISSN: 17480221 (cit. on p. 113).
- [66] R. Koppenhöfer et al., *Beam test results of silicon sensor module prototypes for the Phase-2 Upgrade of the CMS Outer Tracker*, in proceedings of the **Journal of Instrumentation**, vol. 16, 12, 2021 (cit. on p. 113).
- [67] S. Andreas et al., *Proposal for an Experiment to Search for Light Dark Matter at the SPS*, arXiv preprint arXiv:1312.3309 (2013) (cit. on p. 126).
- [68] S. Agostinelli et al., *GEANT4 - A simulation toolkit*, **Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment** **506** (2003), ISSN: 01689002 (cit. on p. 157).

Appendix A

CIC User Data Mapping

CIC 0

E-link	Pin	Data Index
<i>Not Connected</i>	RX0_R - DIO1	Uplink[7:0]
CIC_OUT_1 (stub0)	RX1_R - DIN32	Uplink[103:96]
CIC_OUT_2 (stub1)	RX2_R - DIN0	Uplink[39:32]
CIC_OUT_3 (stub2)	RX3_R - DIN24	Uplink[87:80]
CIC_OUT_4 (stub3)	RX4_R - DIO5	Uplink[15:8]
CIC_OUT_5 (stub4)	RX5_R - DIN28	Uplink[95:88]
CIC_OUT_6 (l1a)	RX6_R - DIN4	Uplink[47:40]
FastCommand	TX0_R - DOUT24	Downlink[87:80]

CIC 1

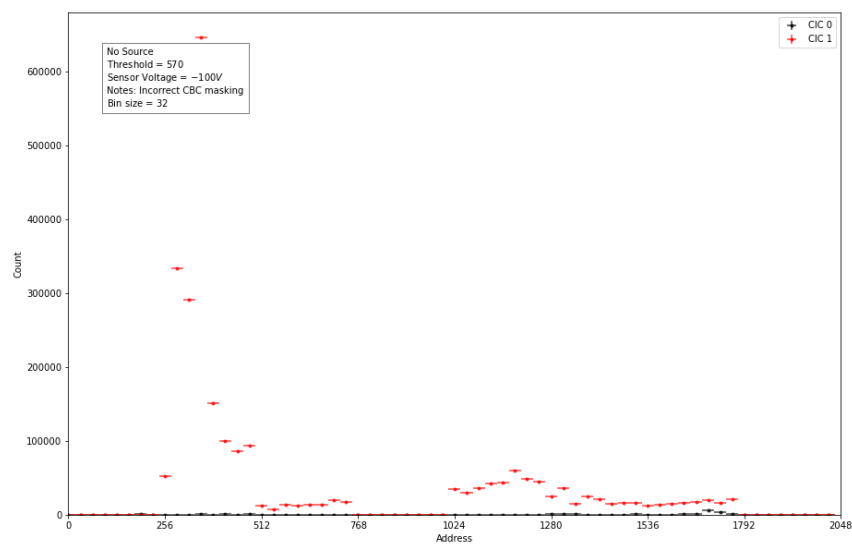
E-link	Pin	Data Index
<i>Not Connected</i>	RX0_L - DIN8	Uplink[55:48]
CIC_OUT_1 (stub0)	RX1_L - DIN12	Uplink[63:56]
CIC_OUT_2 (stub1)	RX2_L - DIN36	Uplink[111:104]
CIC_OUT_3 (stub2)	RX3_L - DIO13	Uplink[31:24]
CIC_OUT_4 (stub3)	RX4_L - DIN20	Uplink[79:72]
CIC_OUT_5 (stub4)	RX5_L - DIO9	Uplink[23:16]
CIC_OUT_6 (l1a)	RX6_L - DIN16	Uplink[71:64]
FastCommand	TX0_L - DOUT20	Downlink[79:72]

Table A.1: Table showing the indices within the user data for each e-link on the lpGBT.

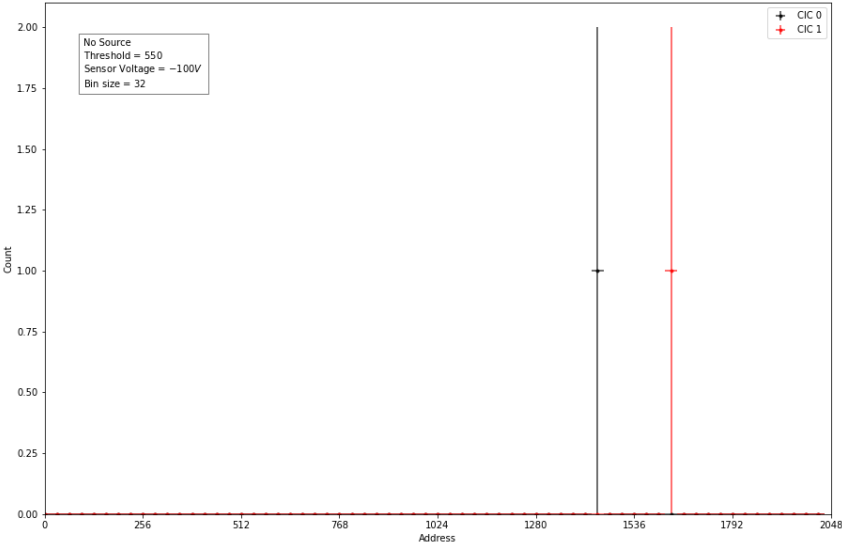
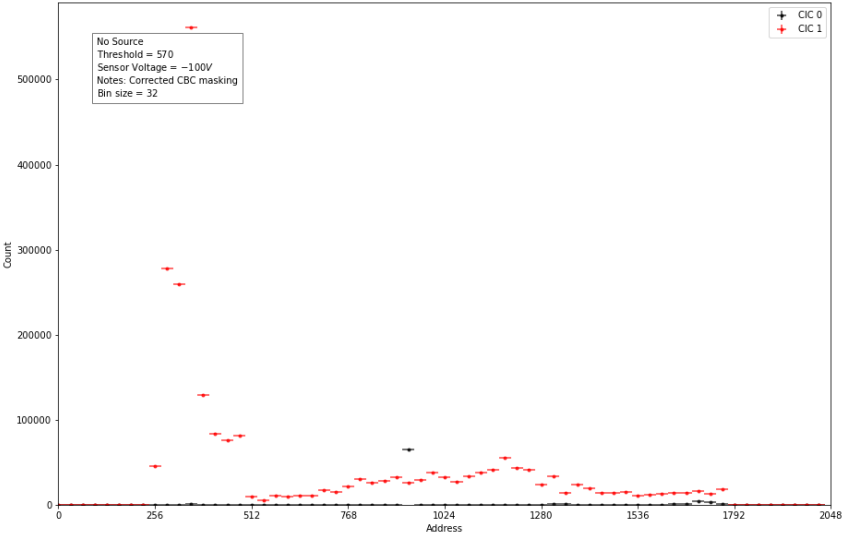
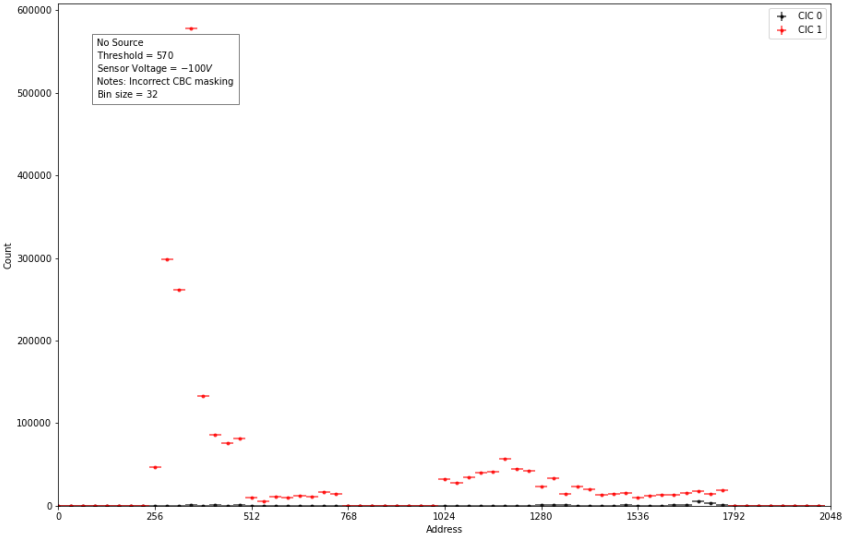
Appendix B

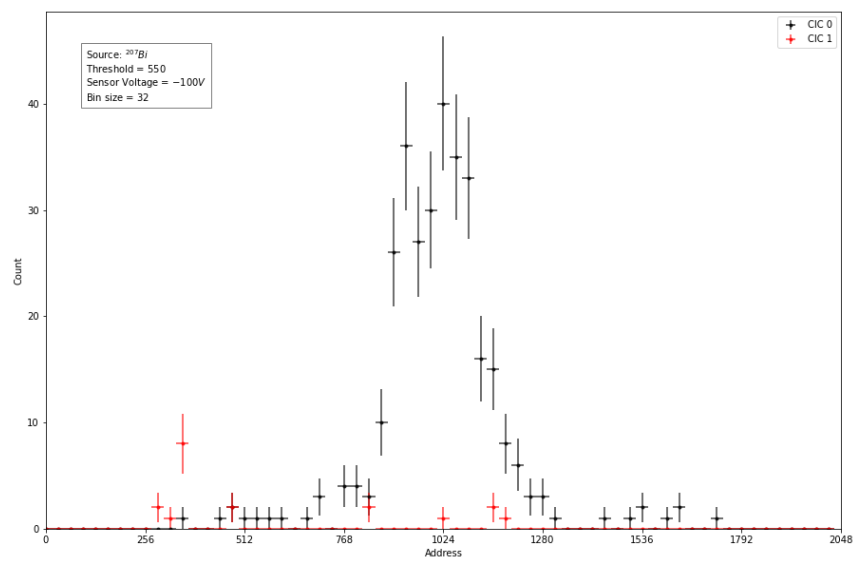
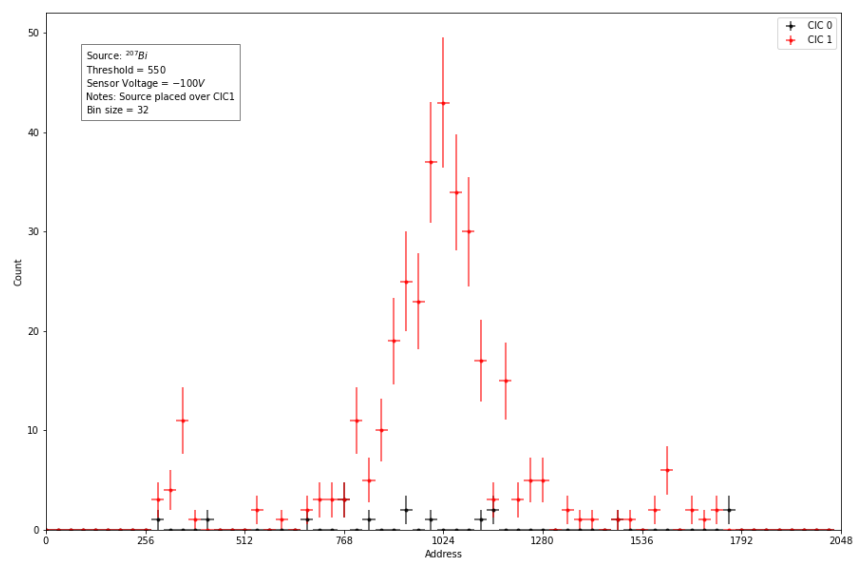
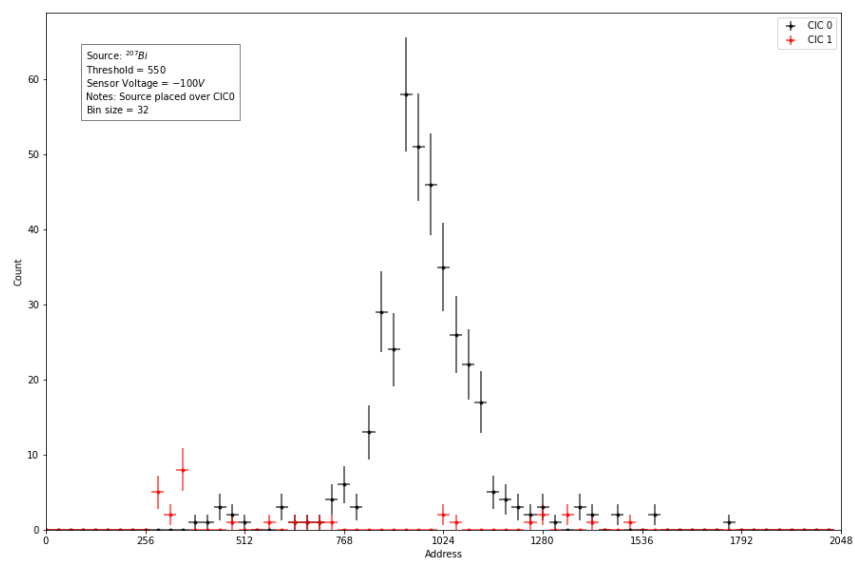
Module Testing Plots

B.1 Day 1

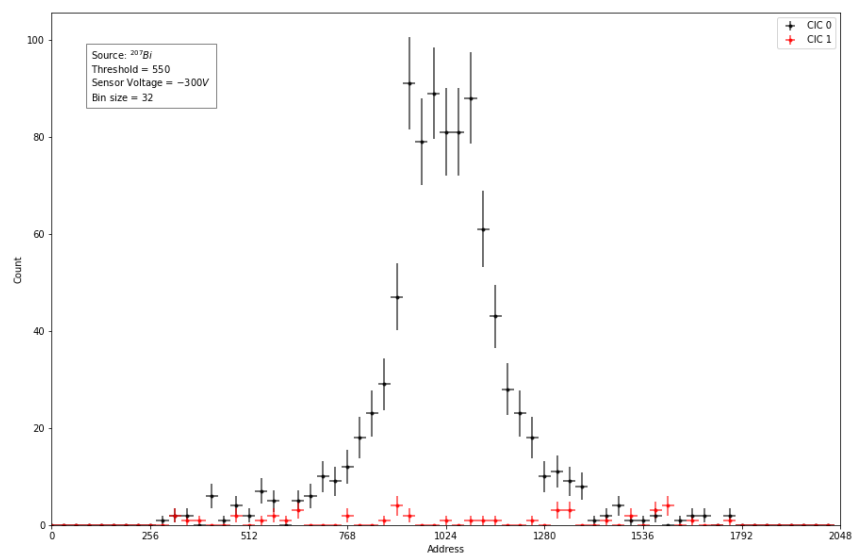
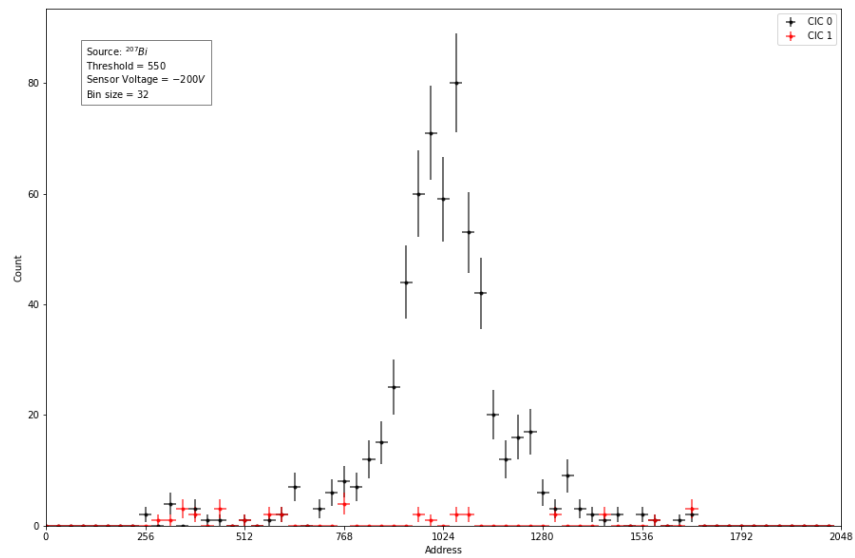
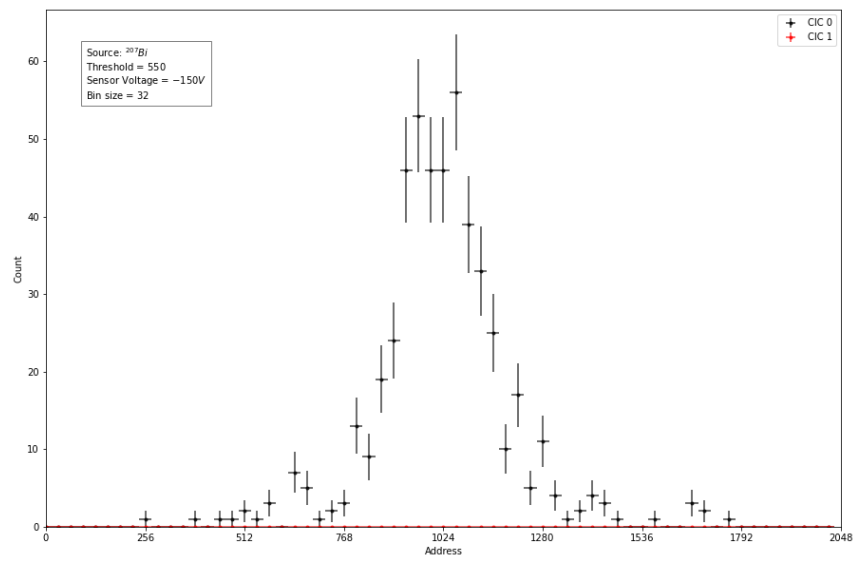


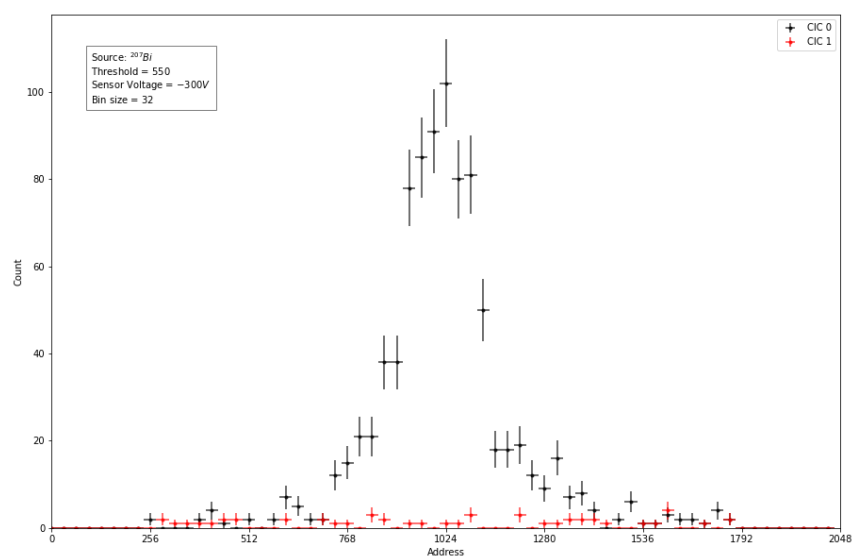
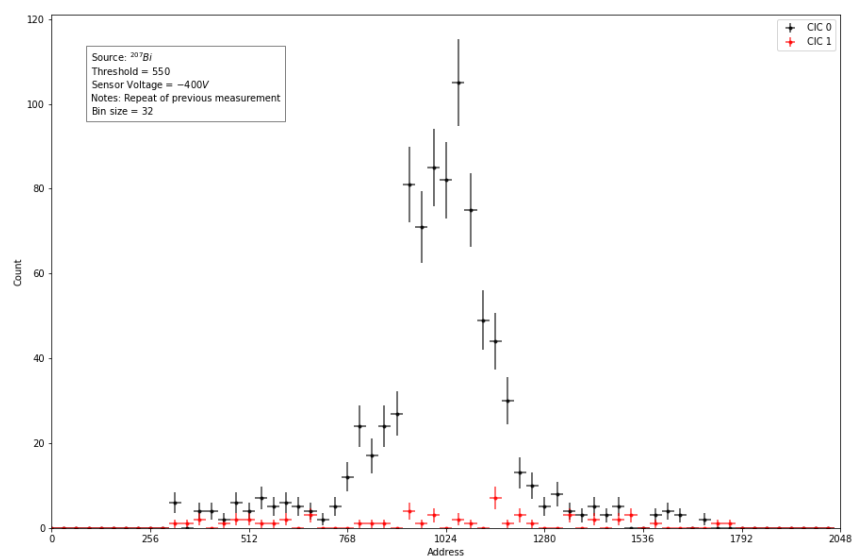
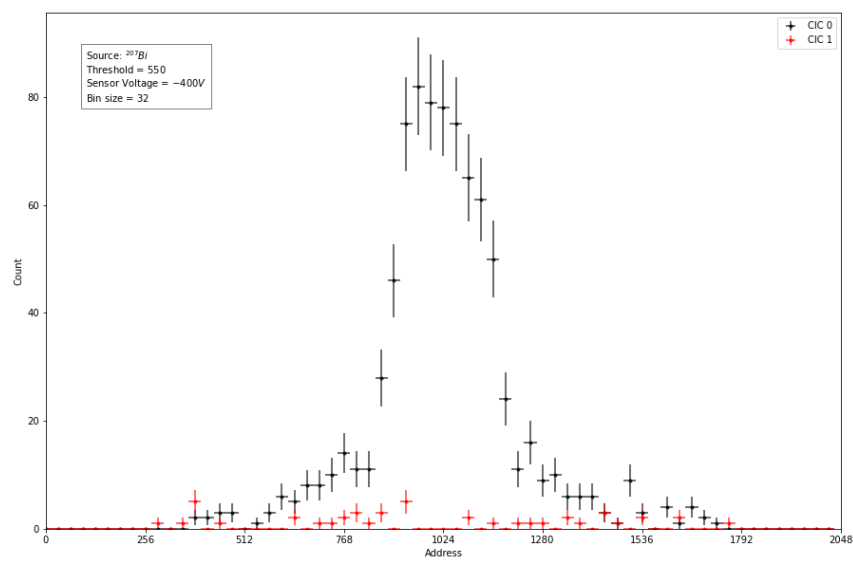
APPENDIX B. MODULE TESTING PLOTS



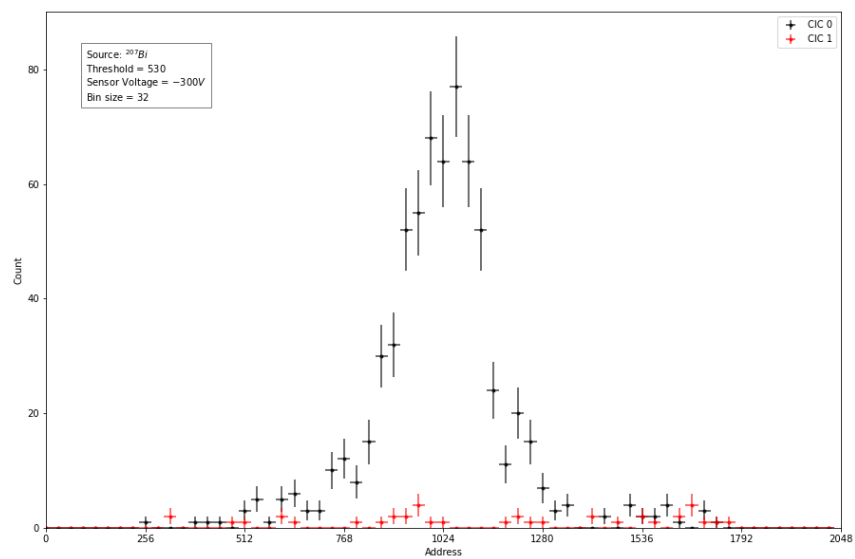
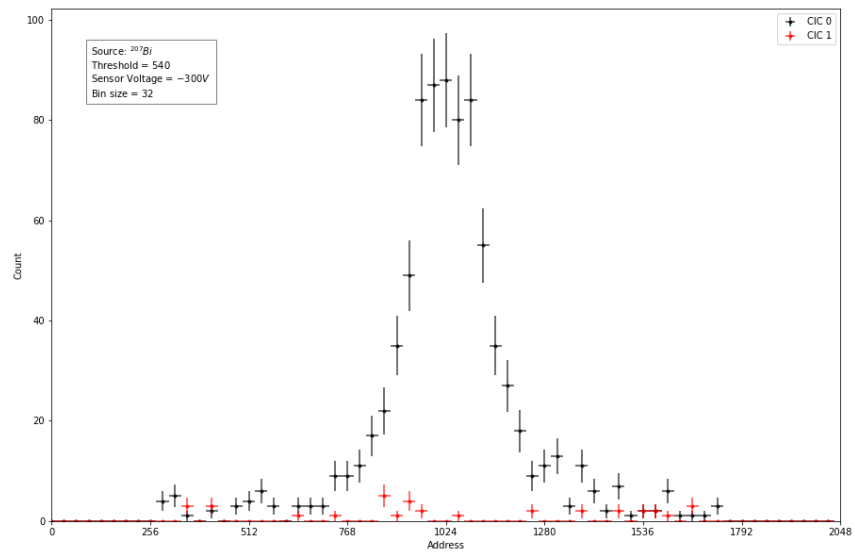
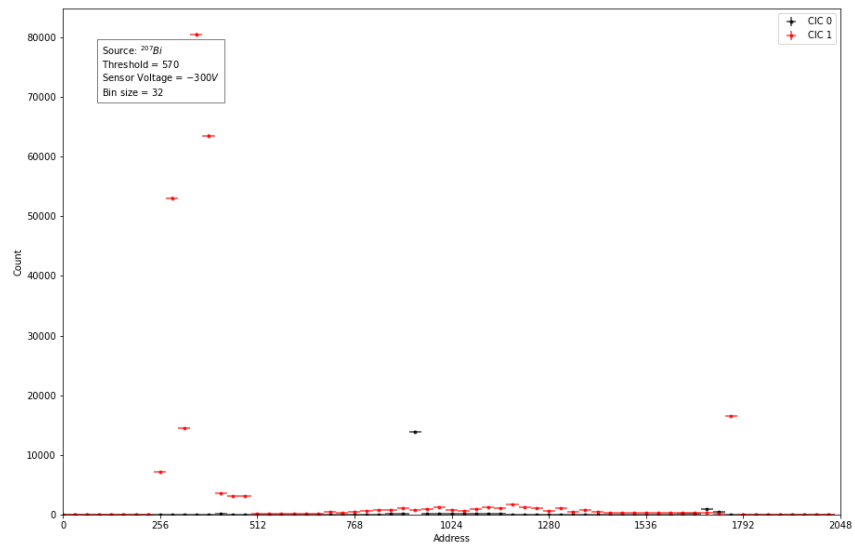


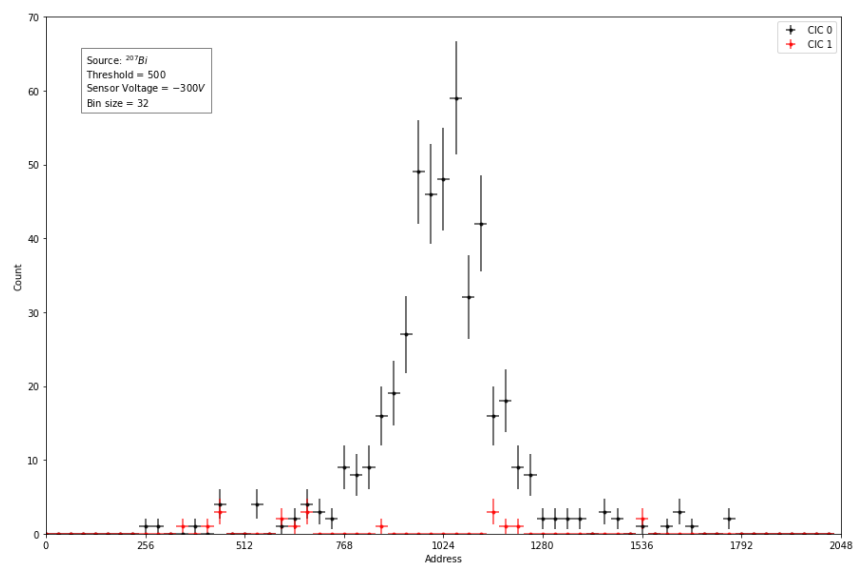
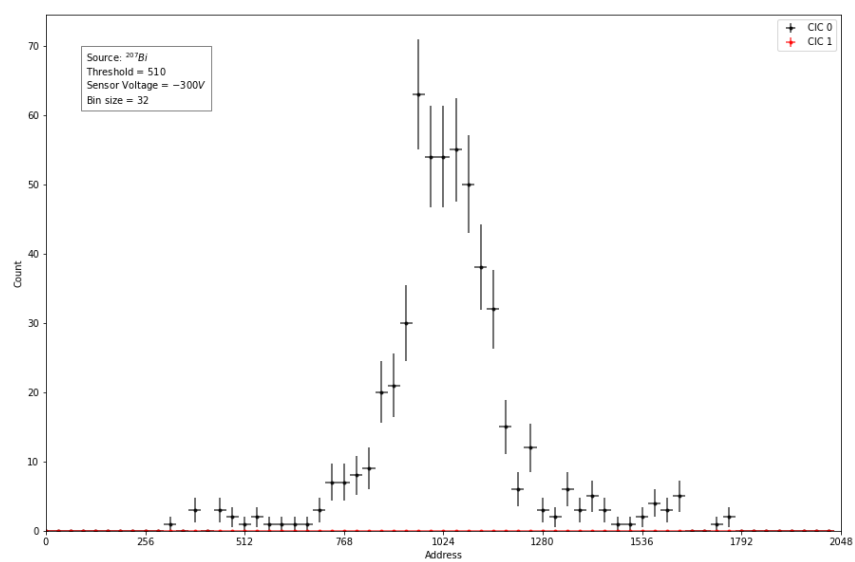
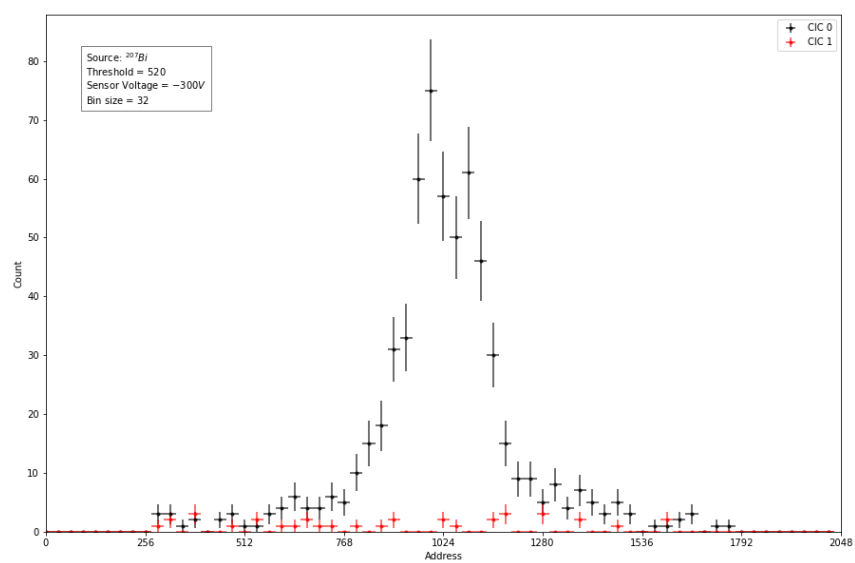
APPENDIX B. MODULE TESTING PLOTS



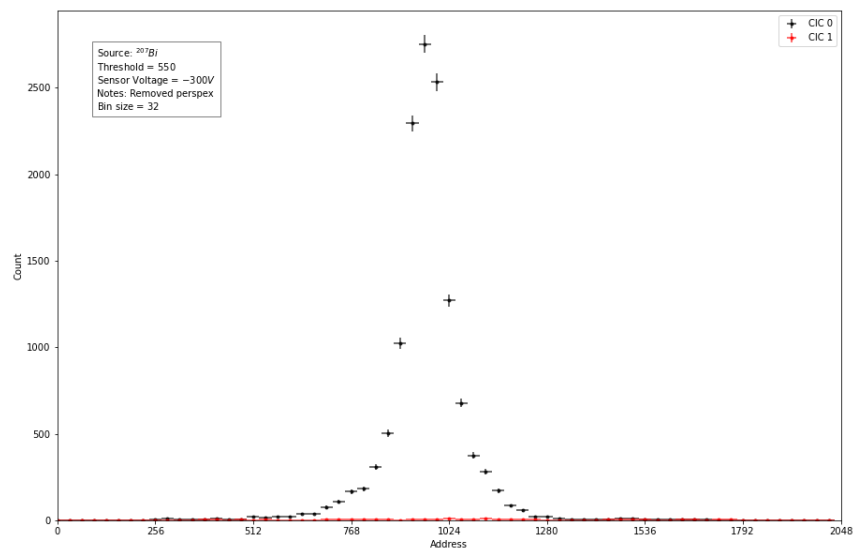
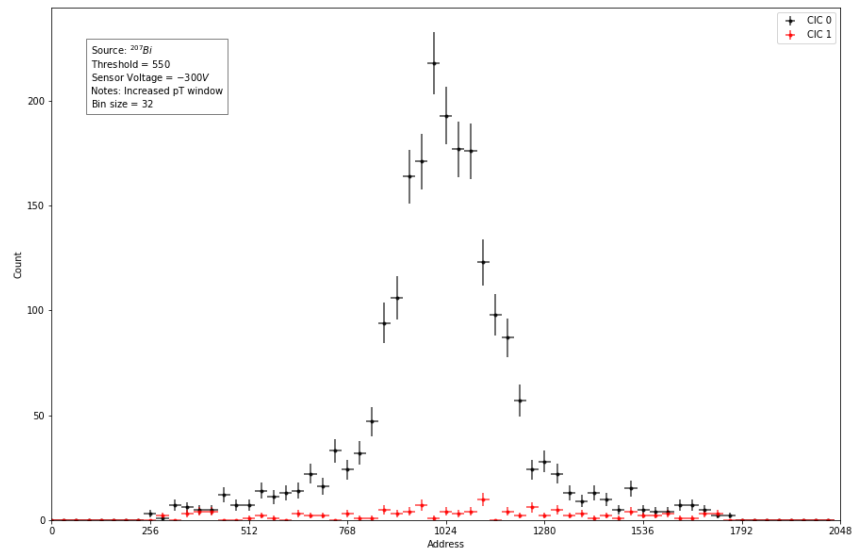
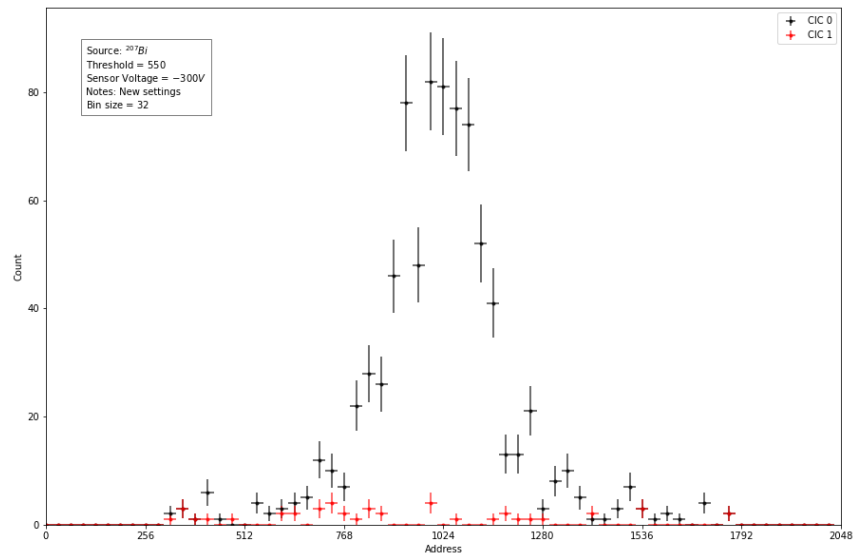


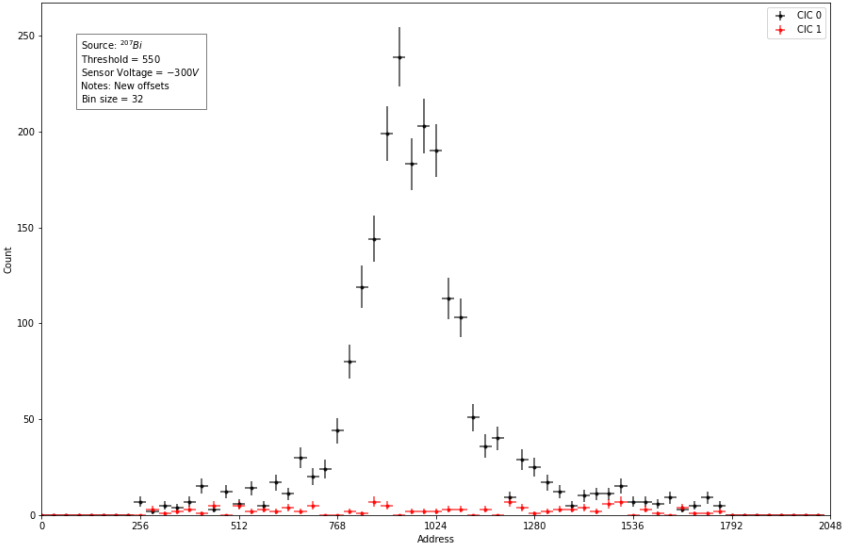
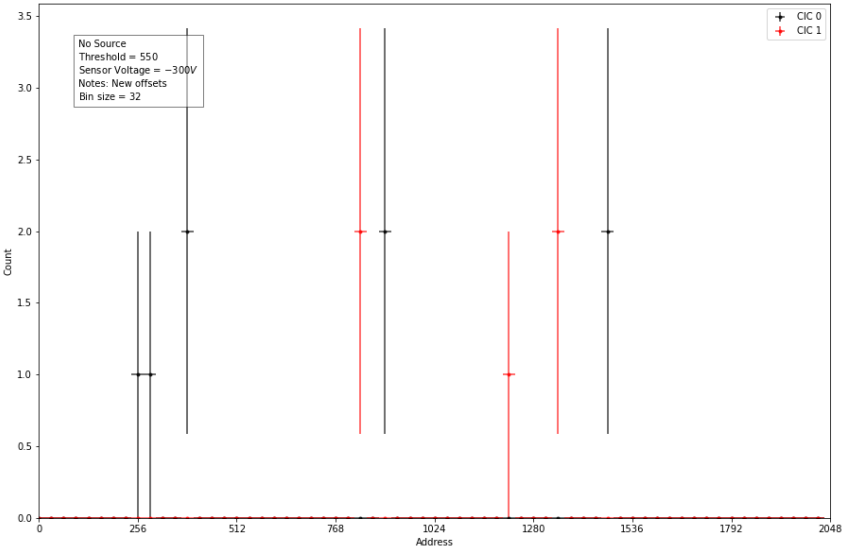
APPENDIX B. MODULE TESTING PLOTS



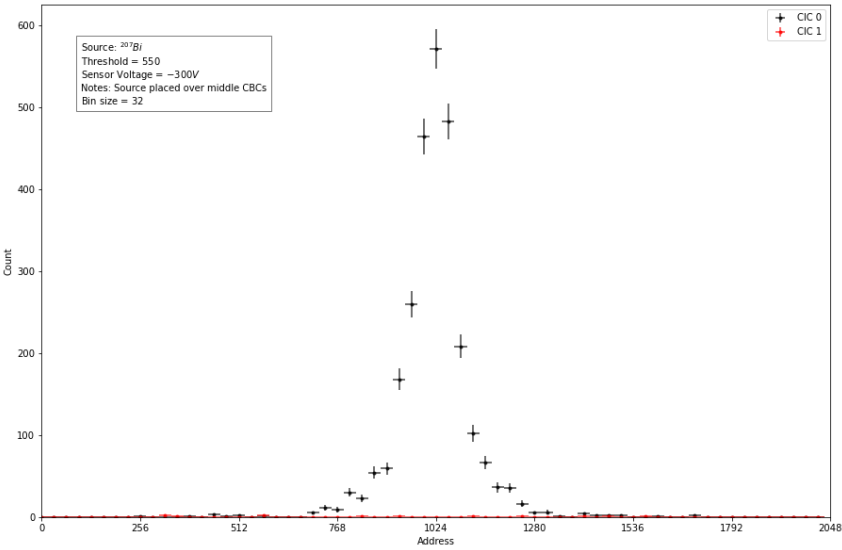
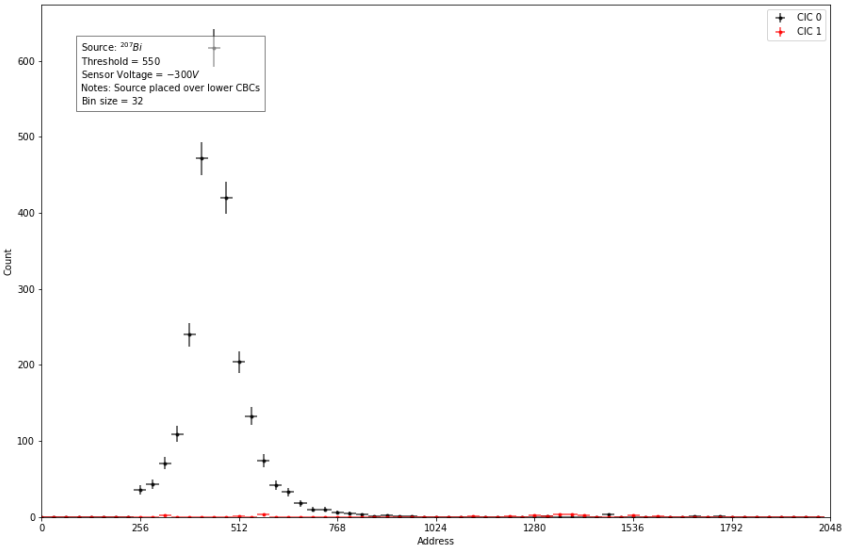


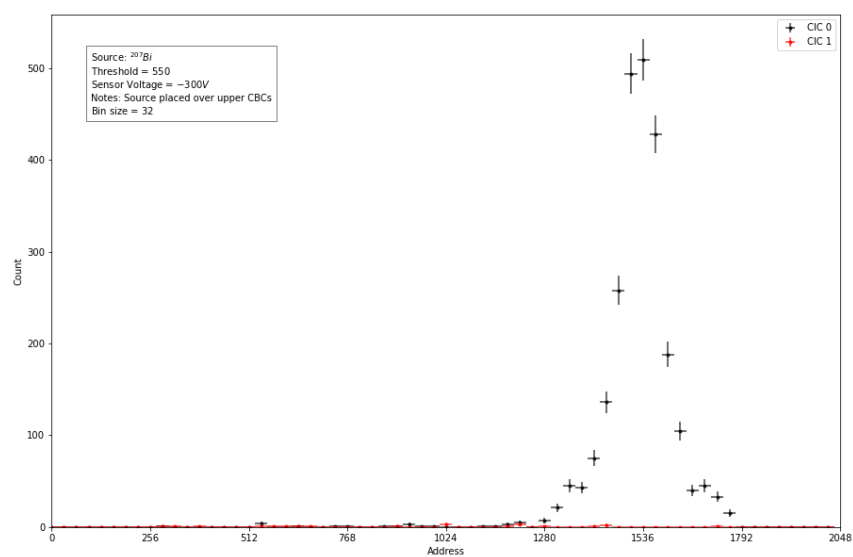
APPENDIX B. MODULE TESTING PLOTS



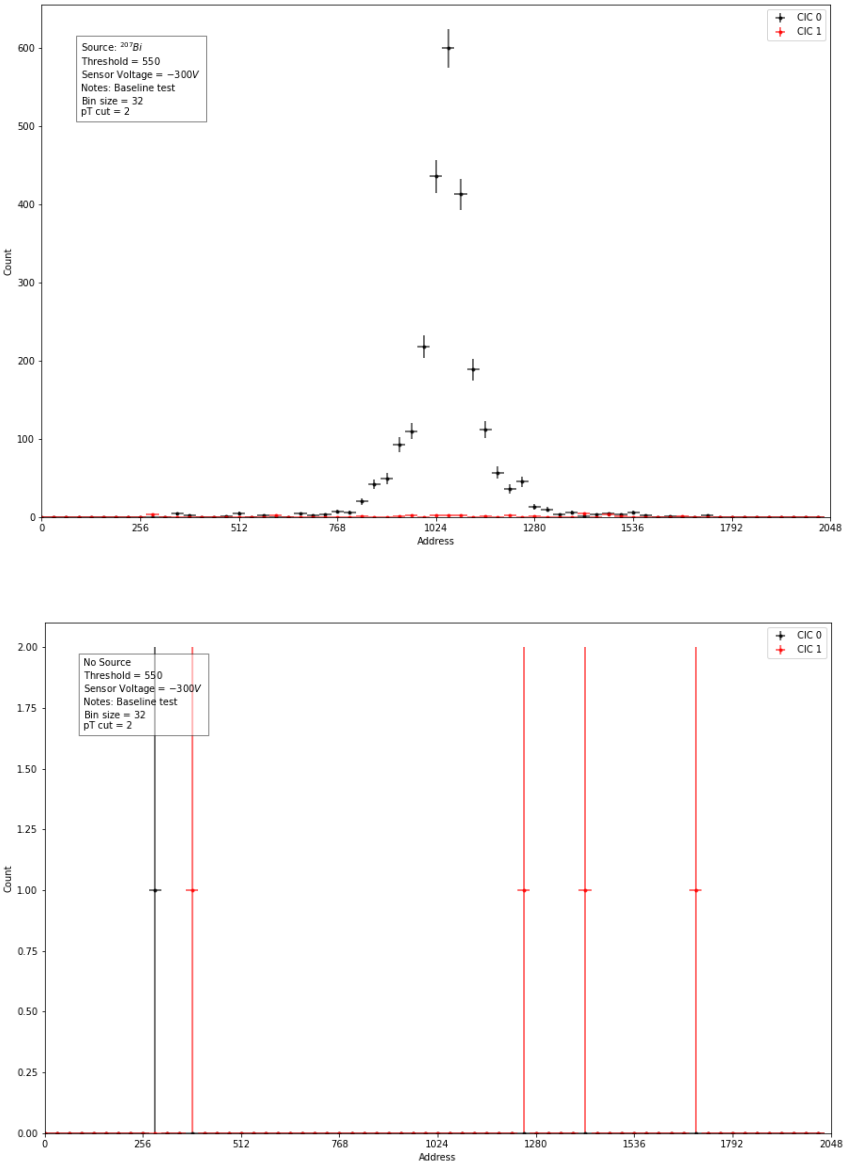


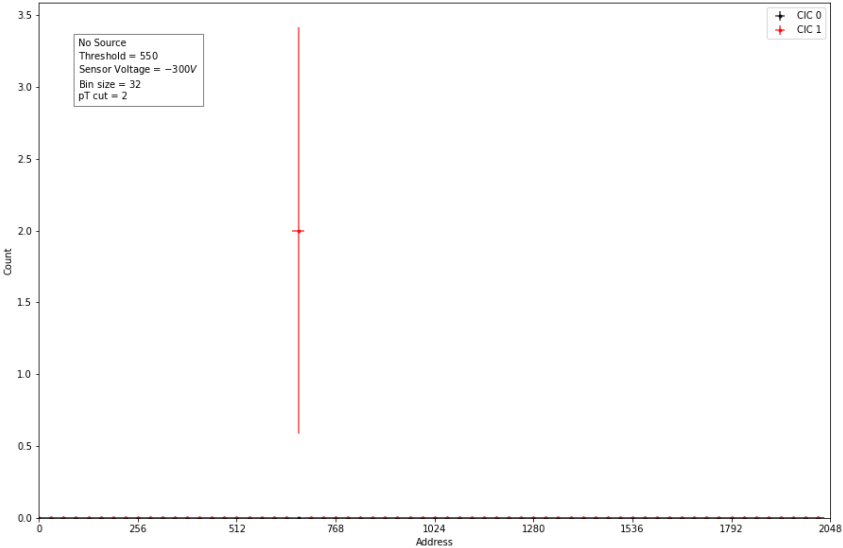
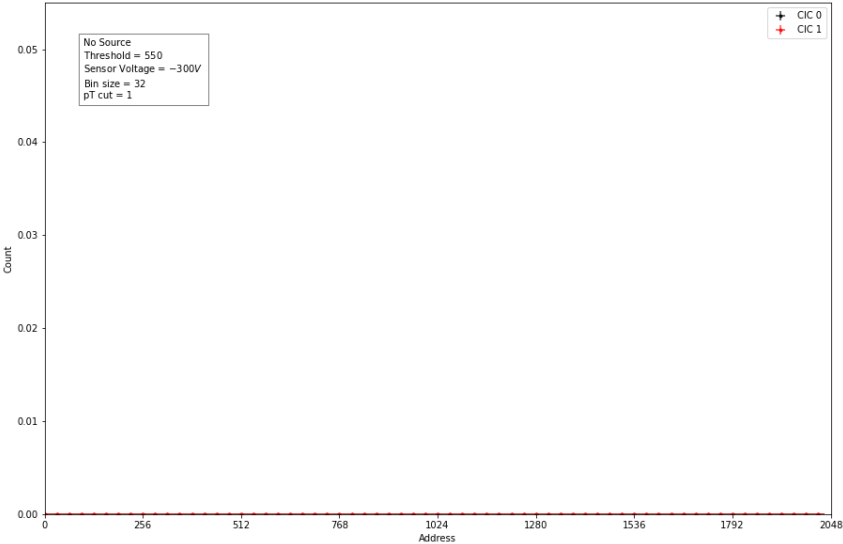
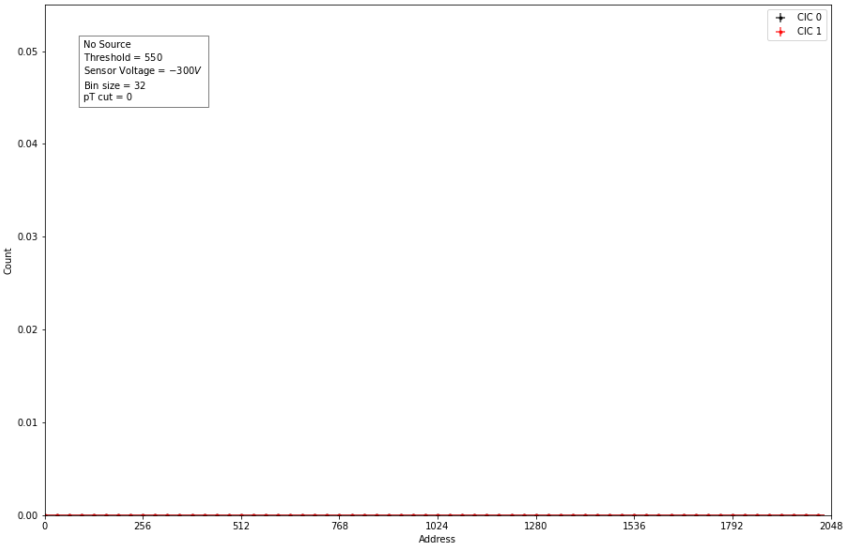
B.2 Day 2



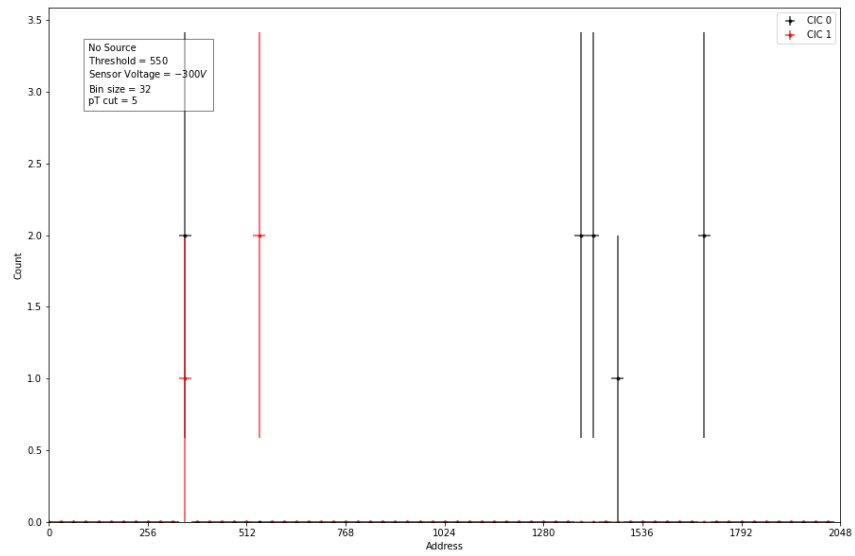
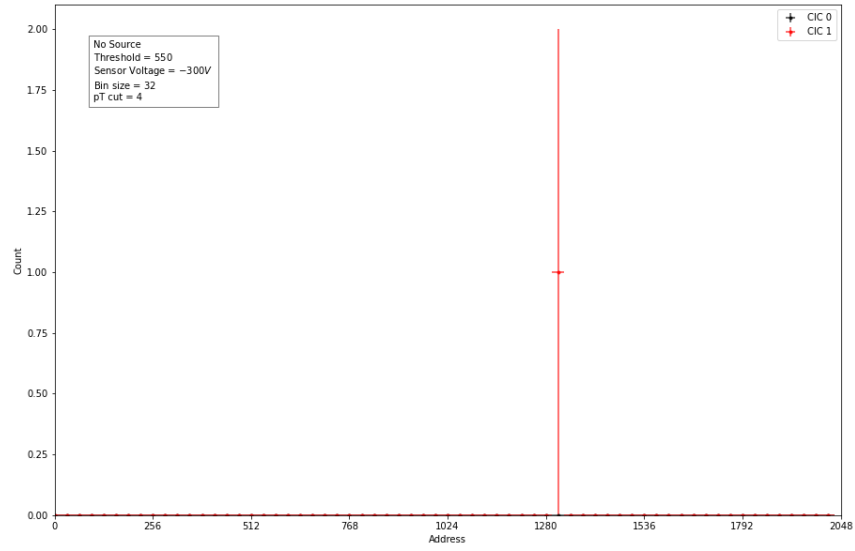
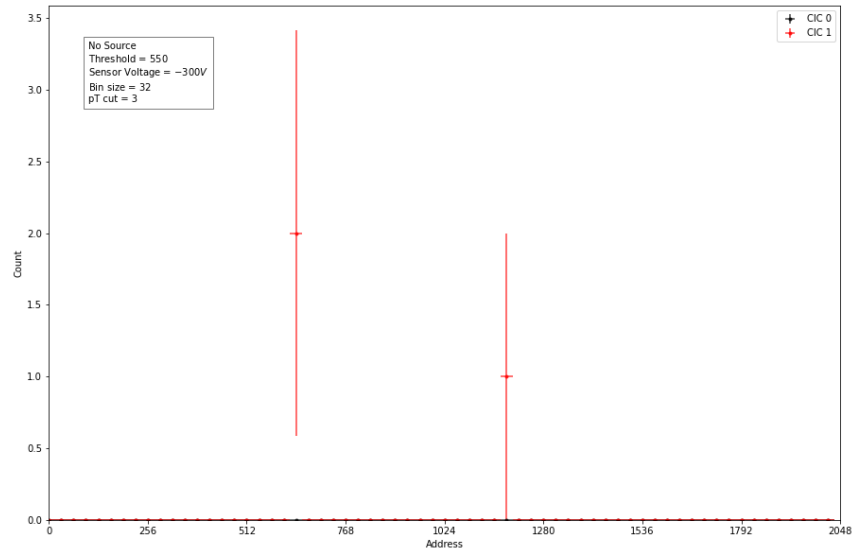


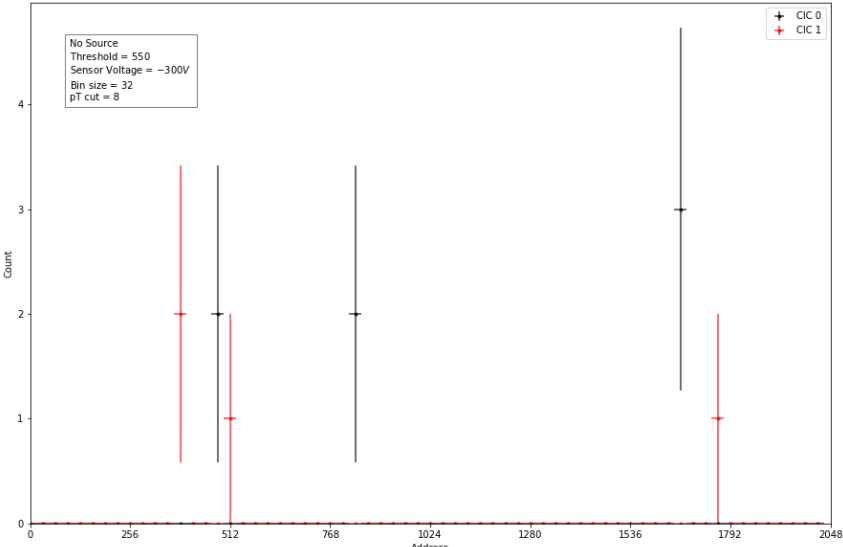
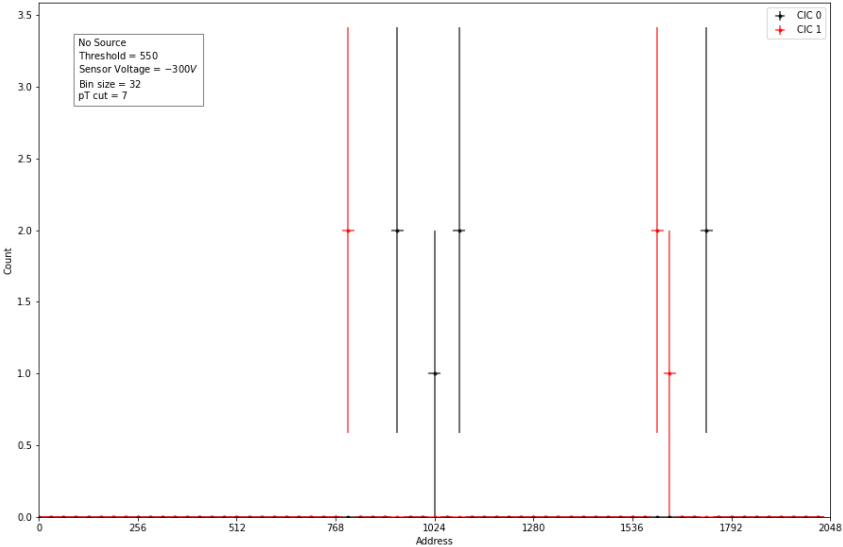
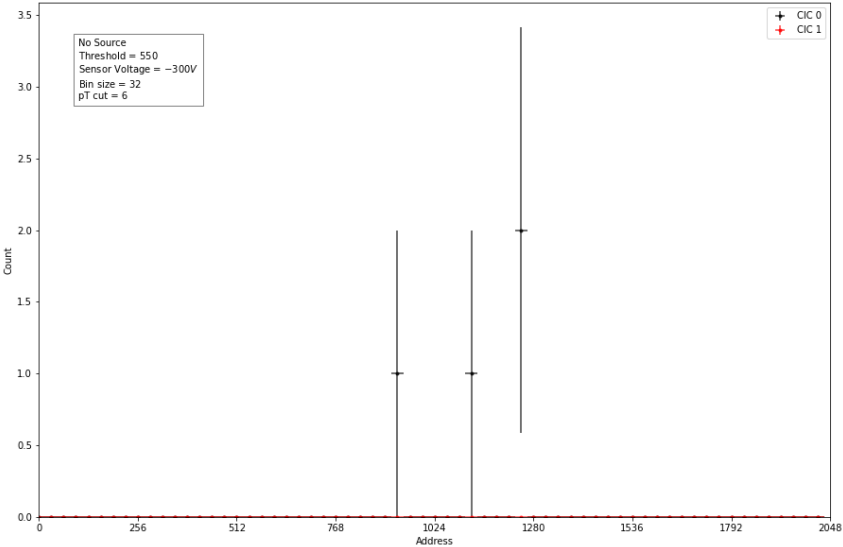
B.3 Day 3



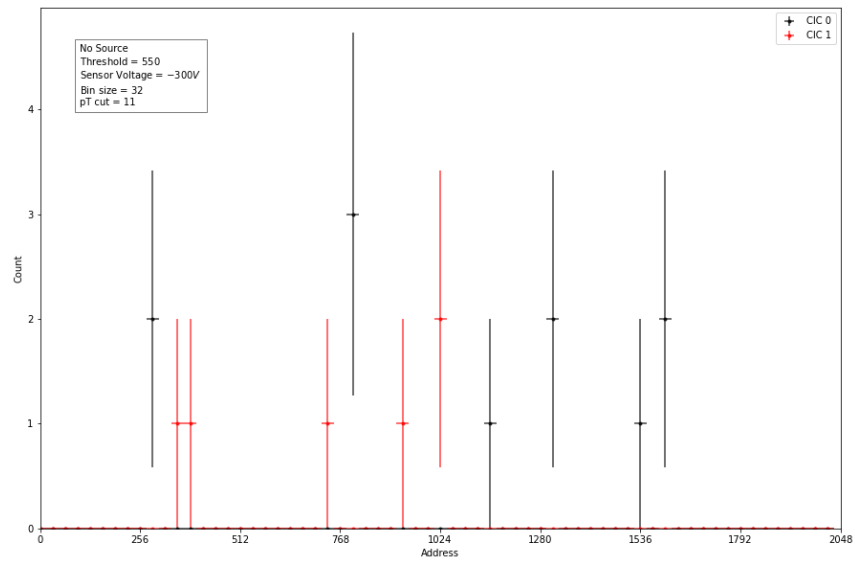
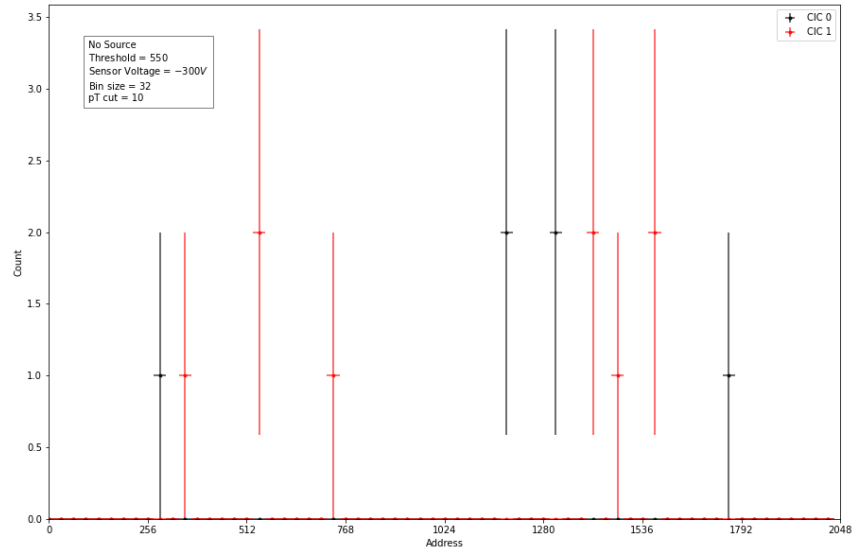
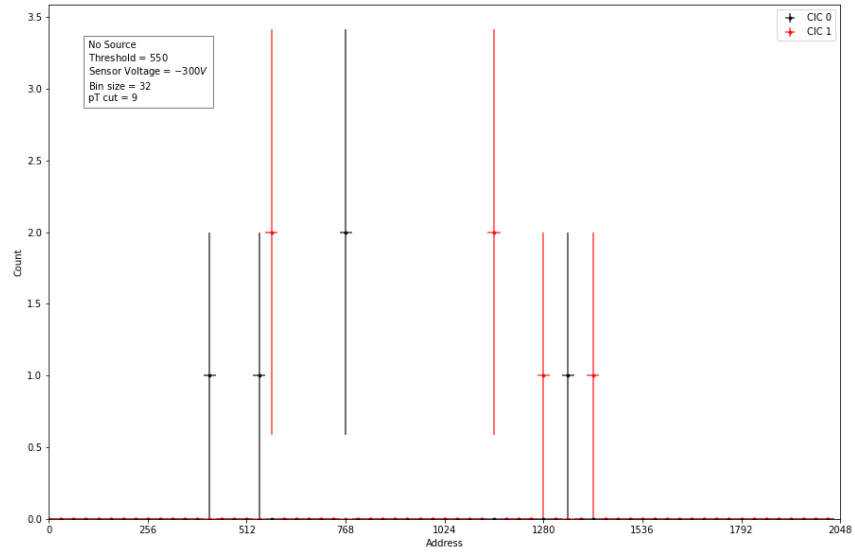


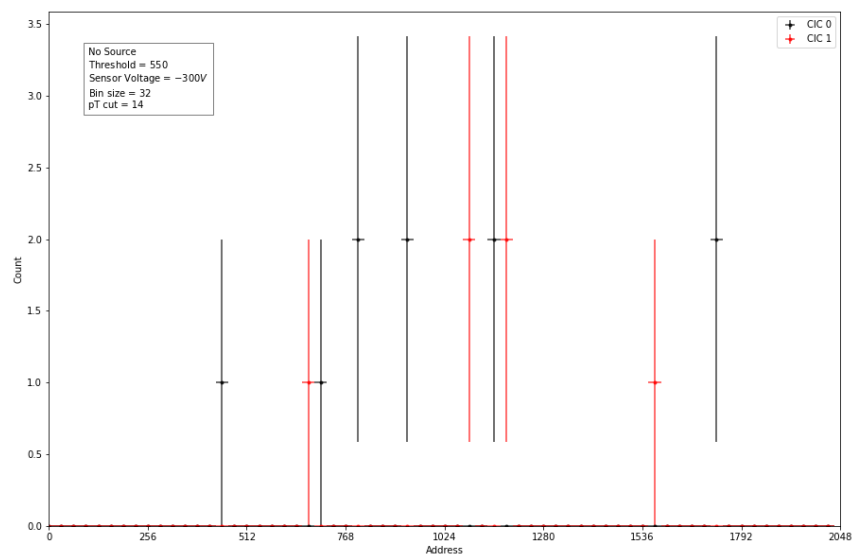
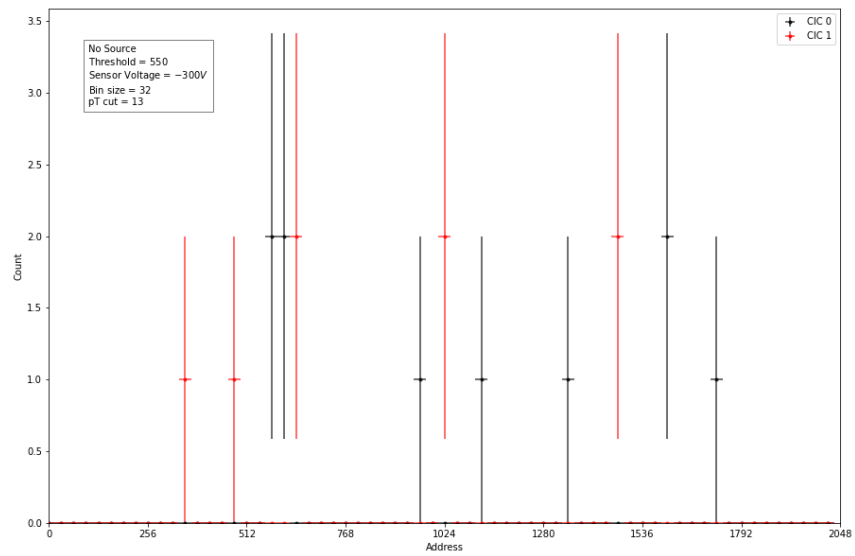
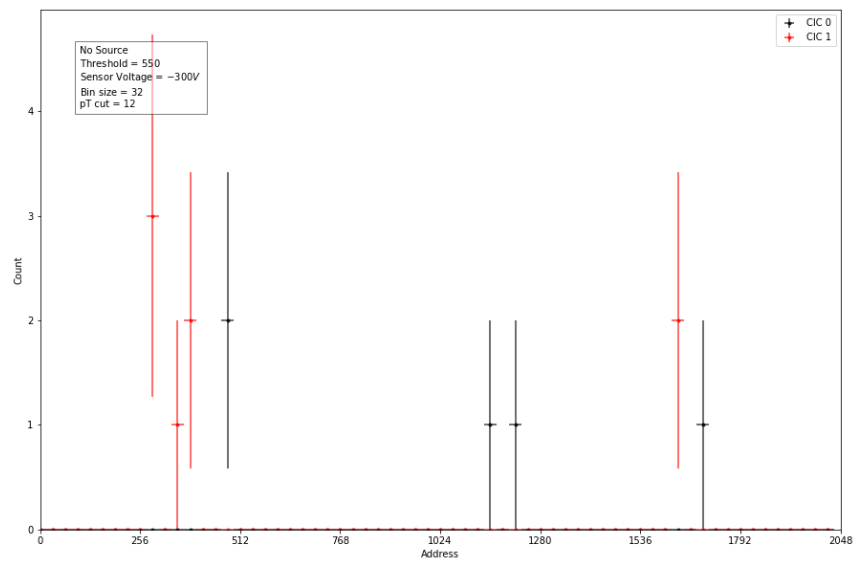
APPENDIX B. MODULE TESTING PLOTS



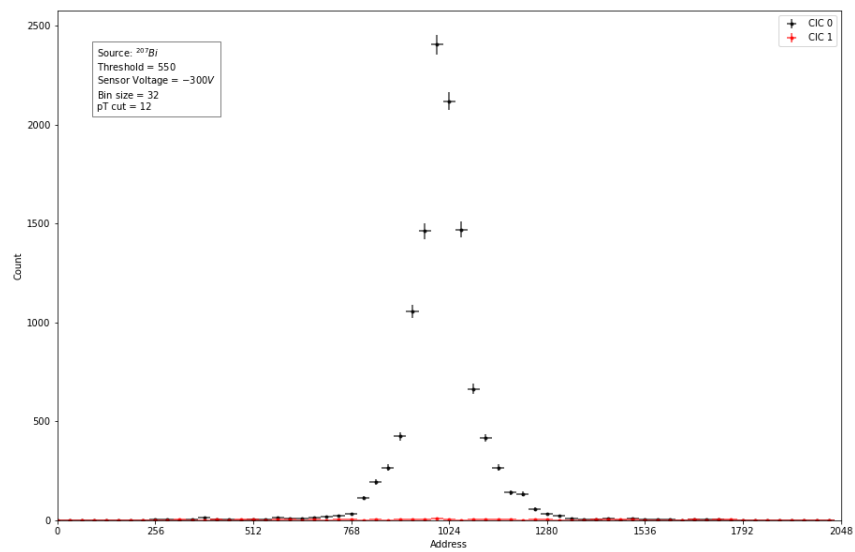
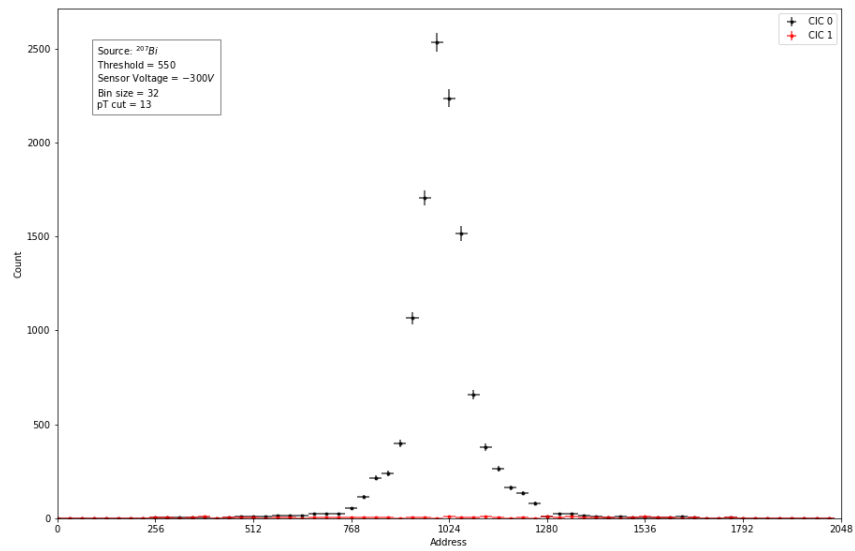
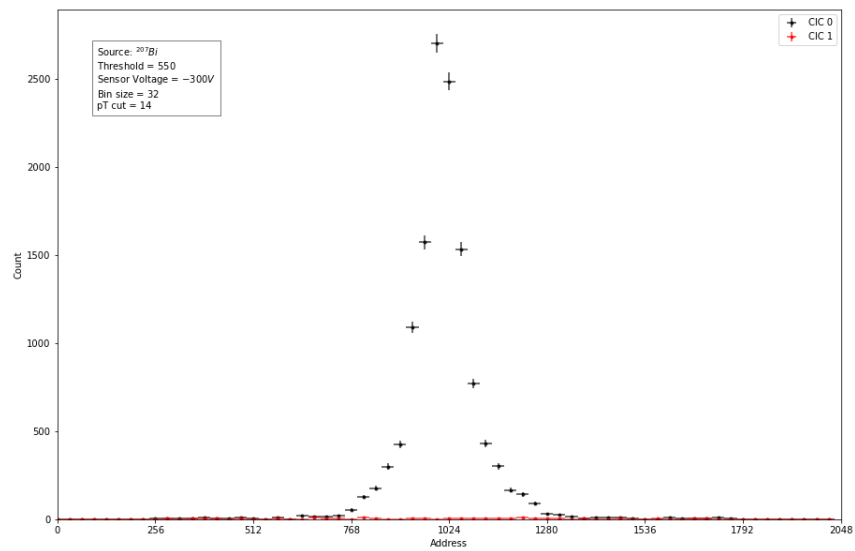


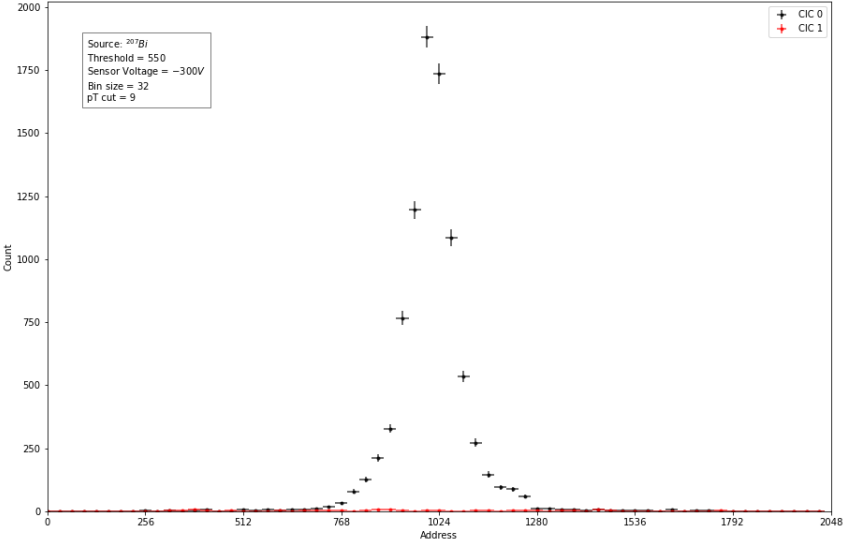
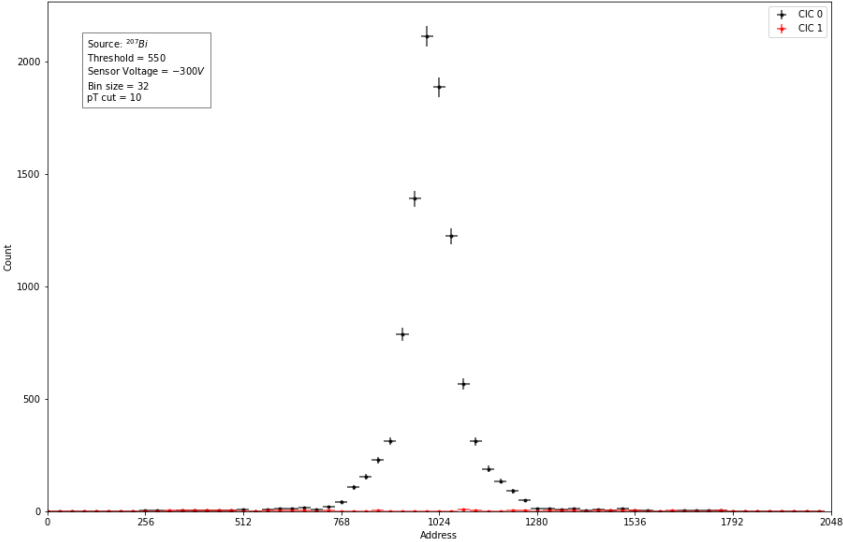
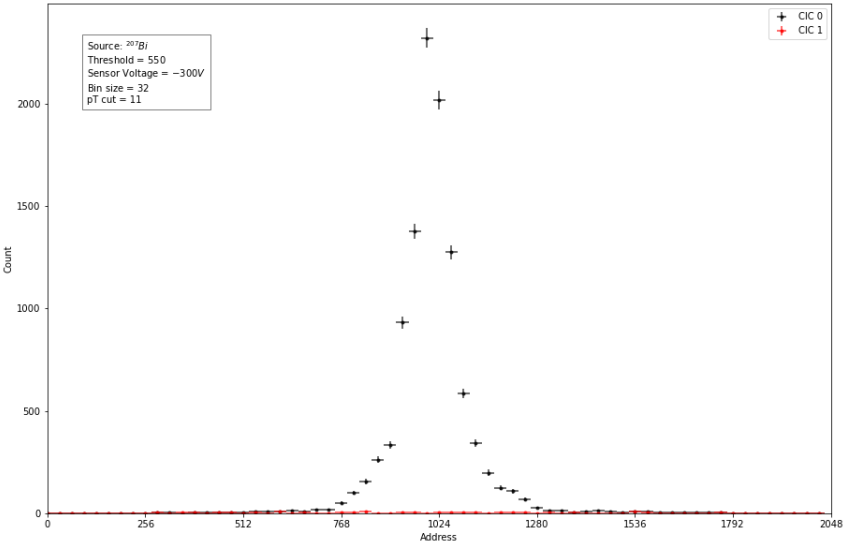
APPENDIX B. MODULE TESTING PLOTS



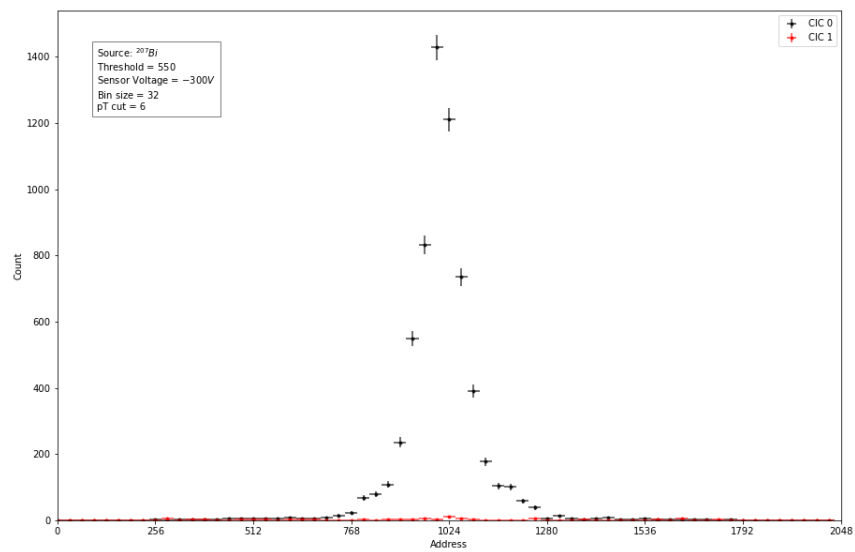
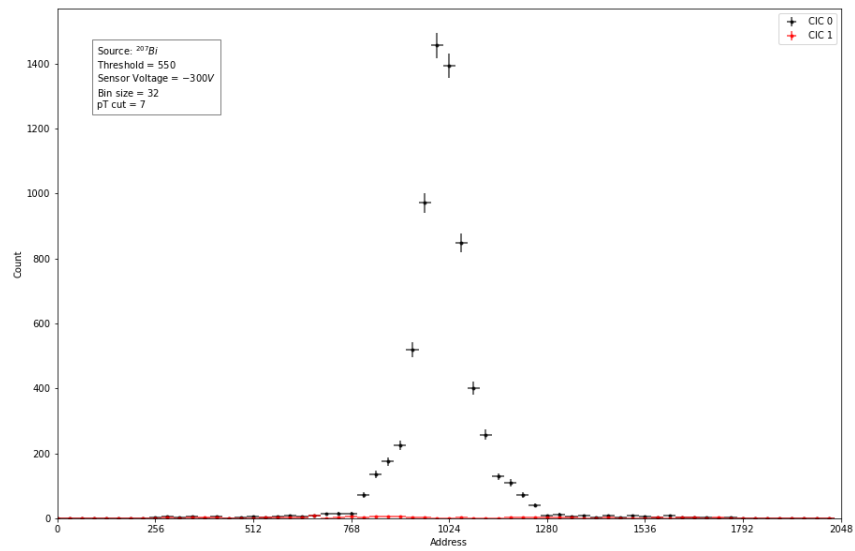
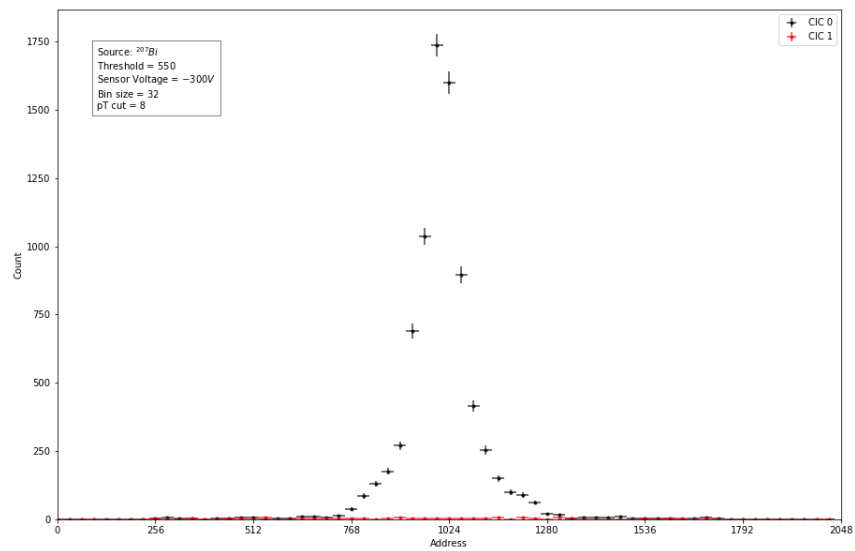


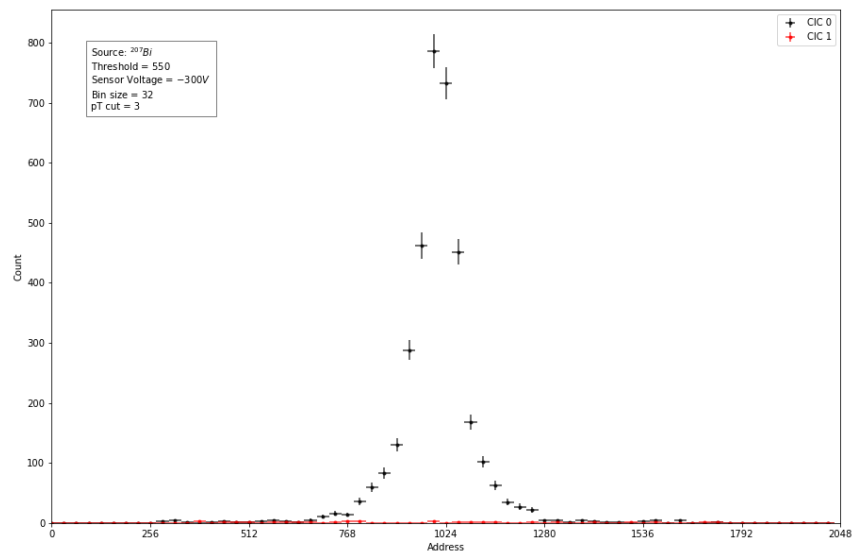
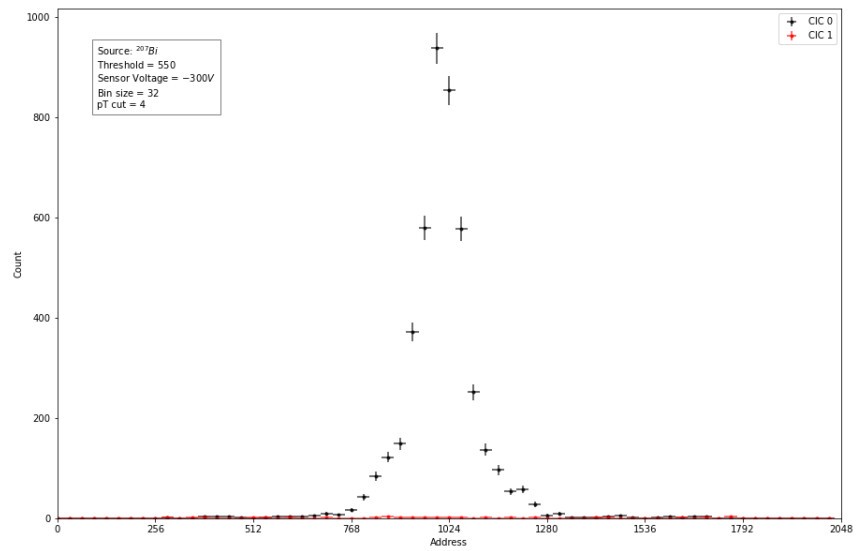
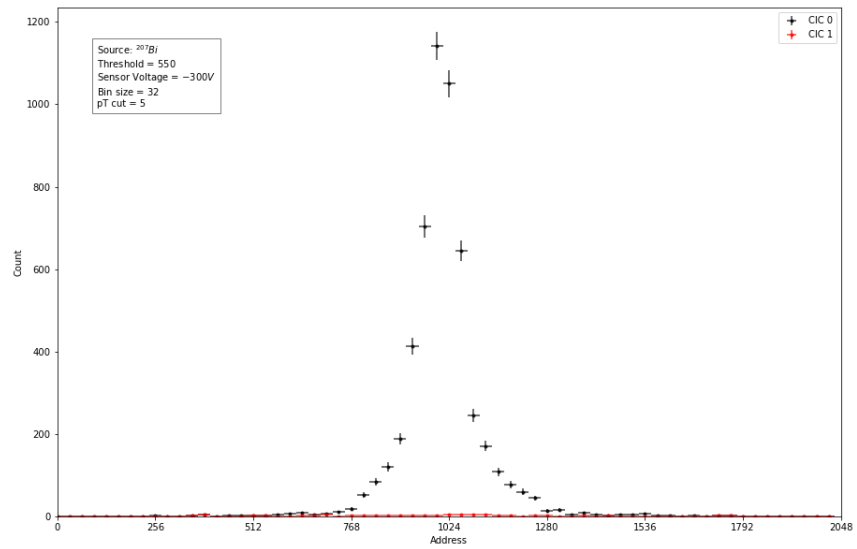
APPENDIX B. MODULE TESTING PLOTS



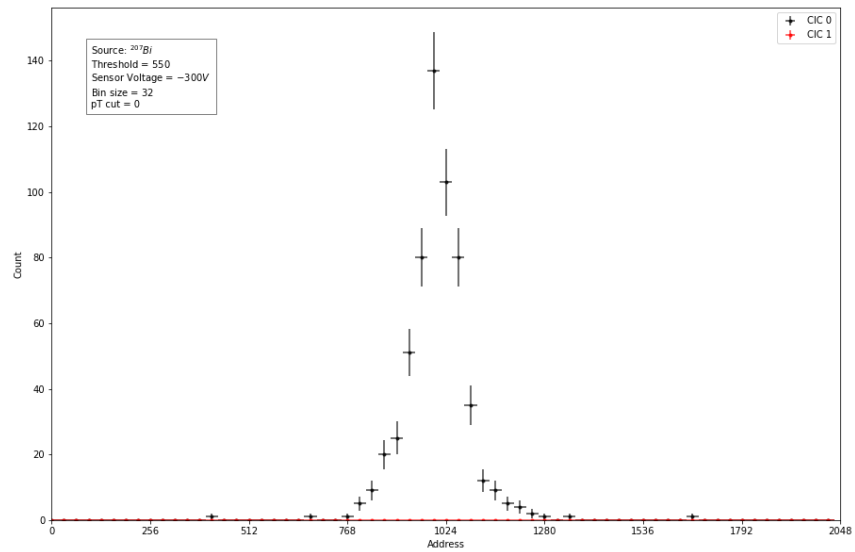
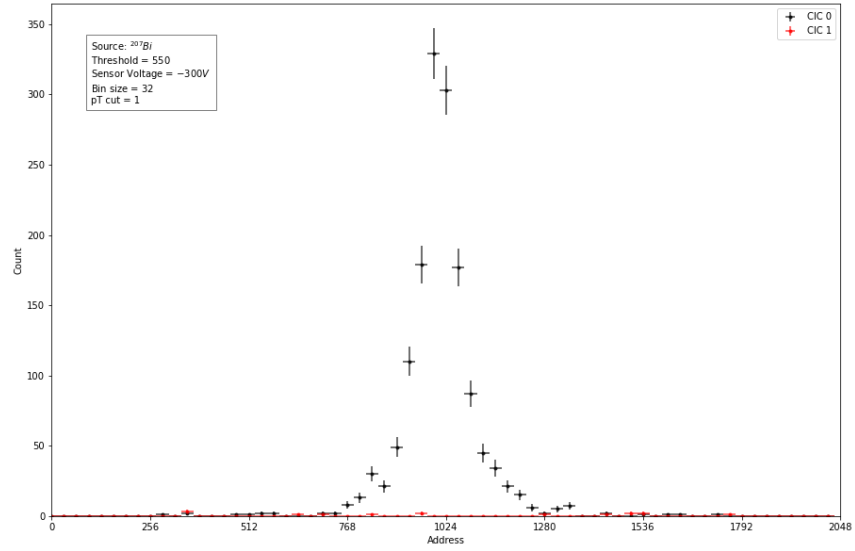
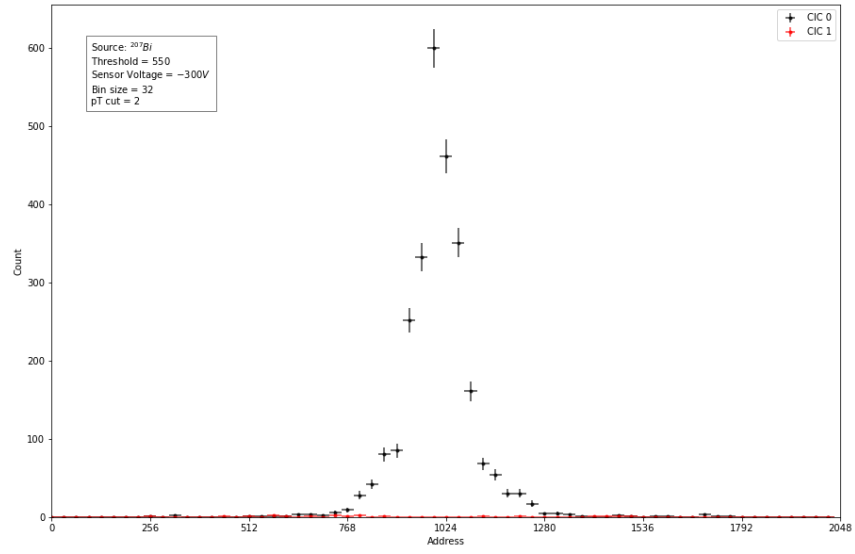


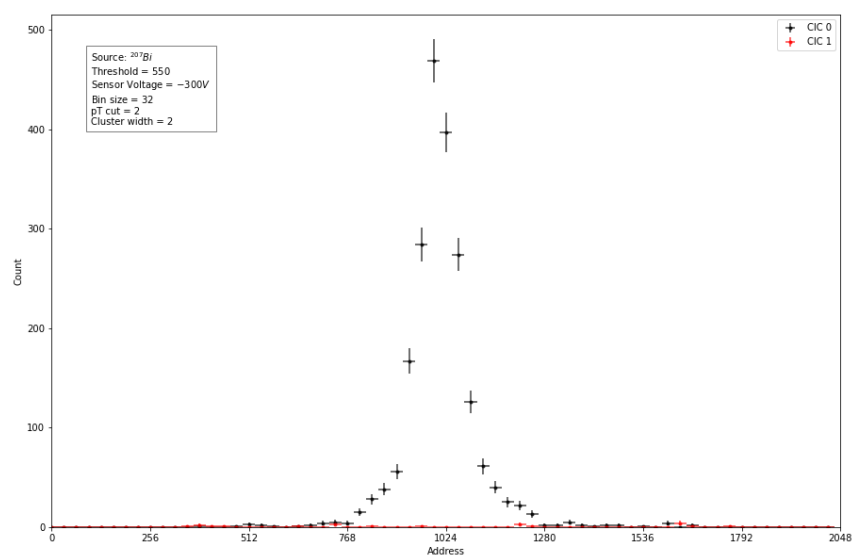
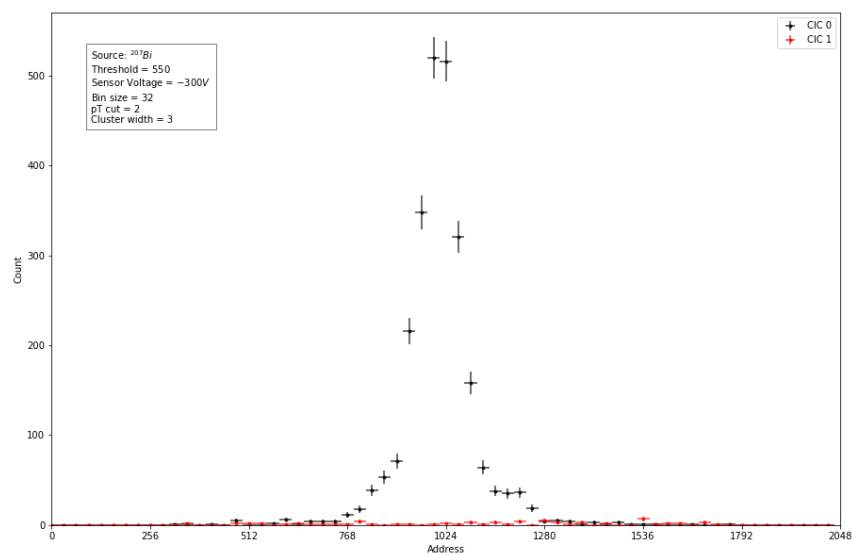
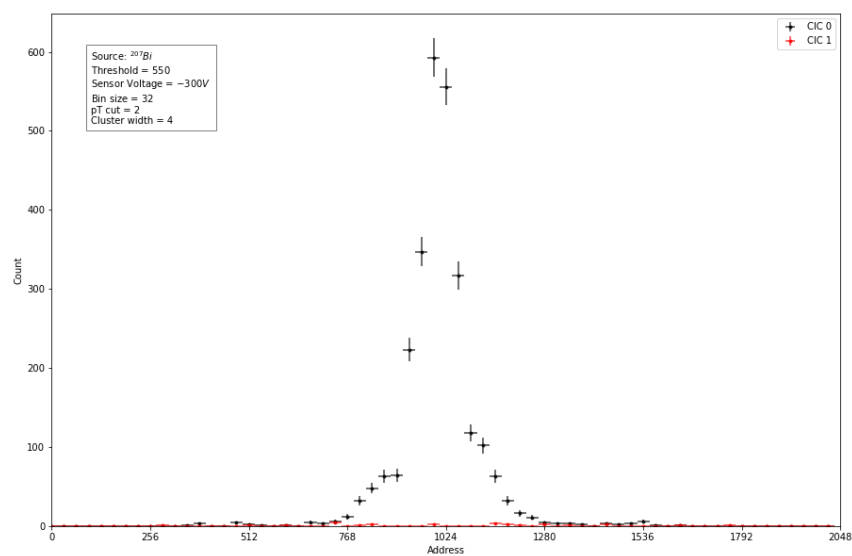
APPENDIX B. MODULE TESTING PLOTS



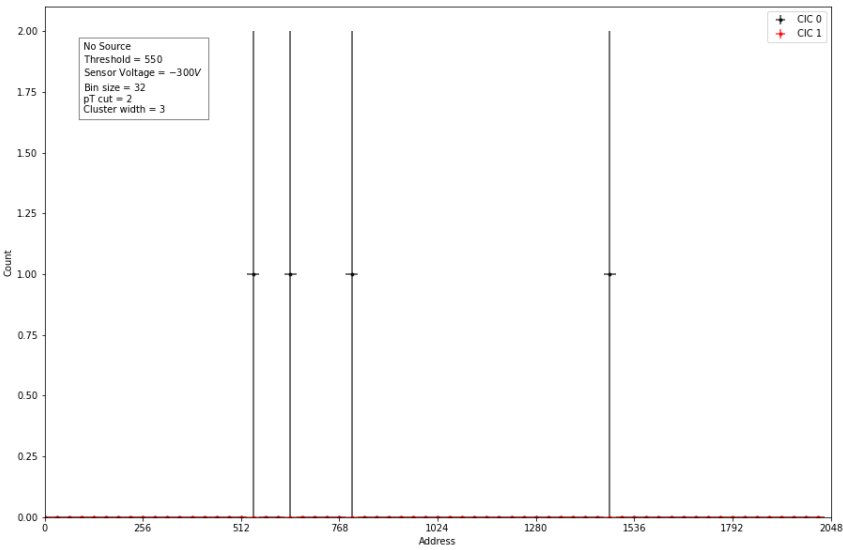
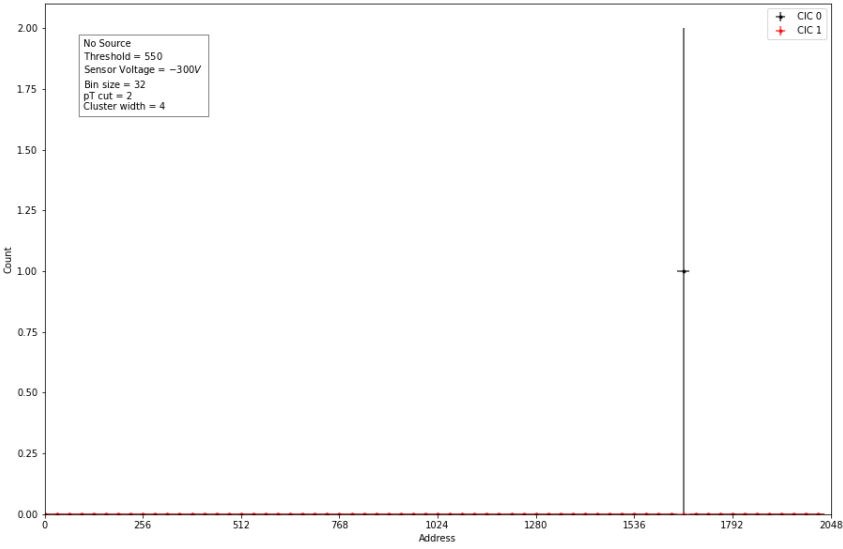
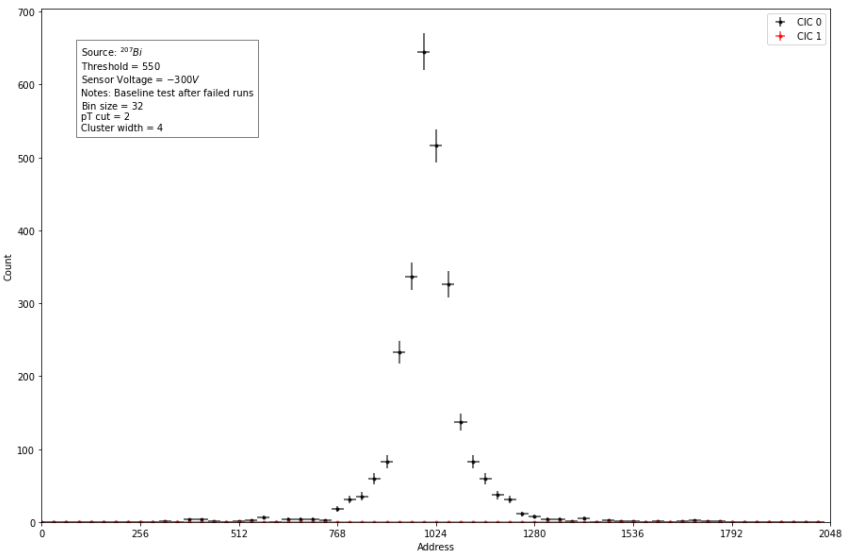


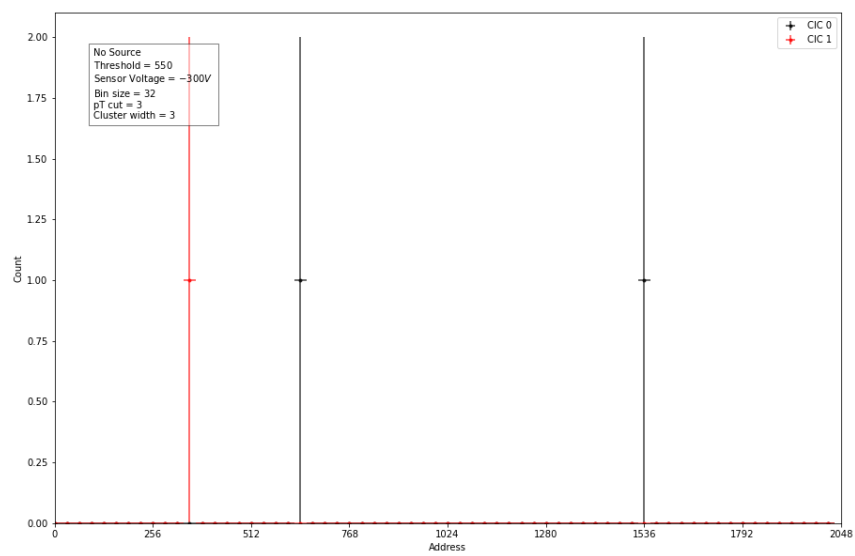
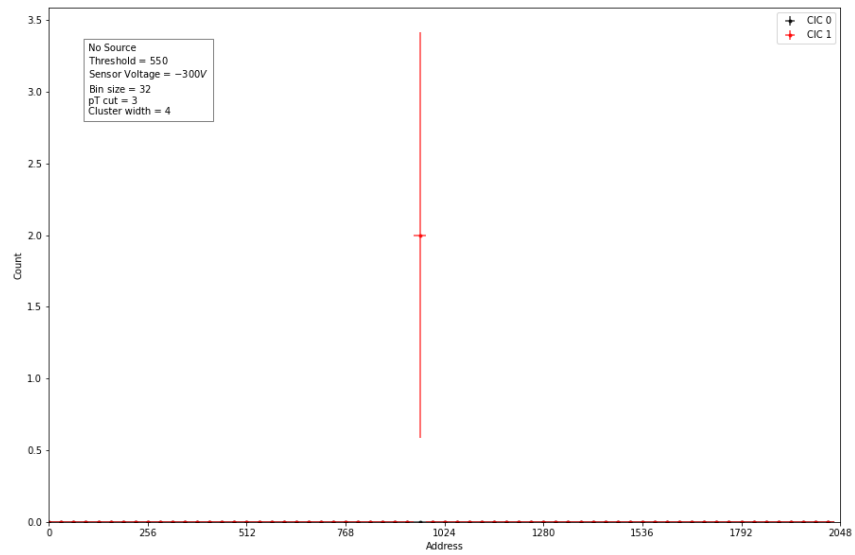
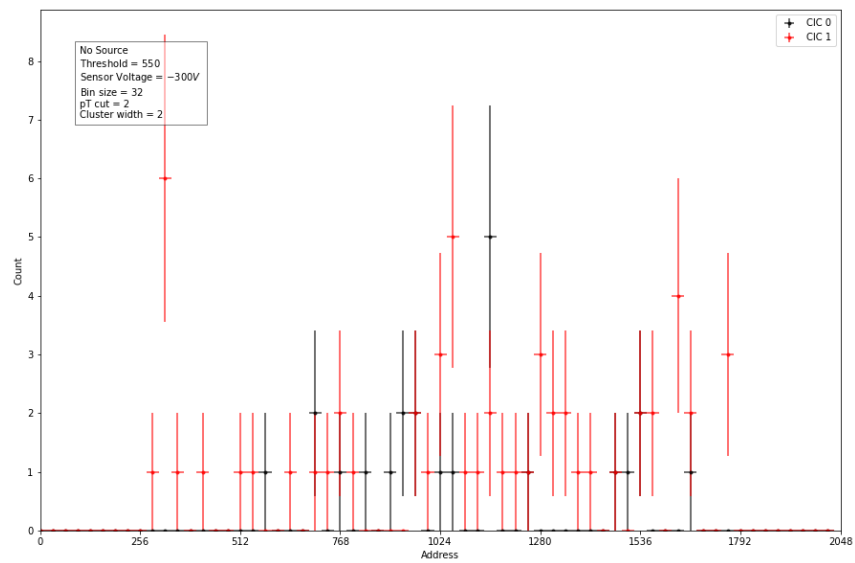
APPENDIX B. MODULE TESTING PLOTS



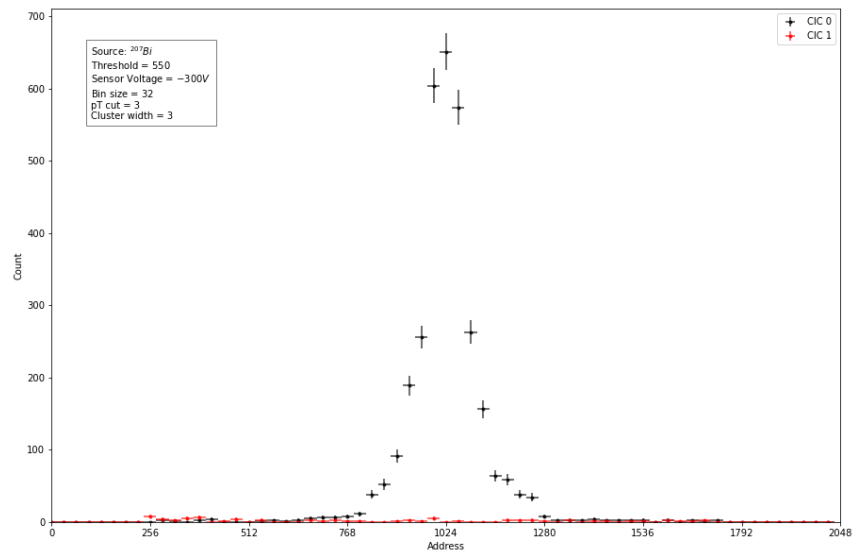
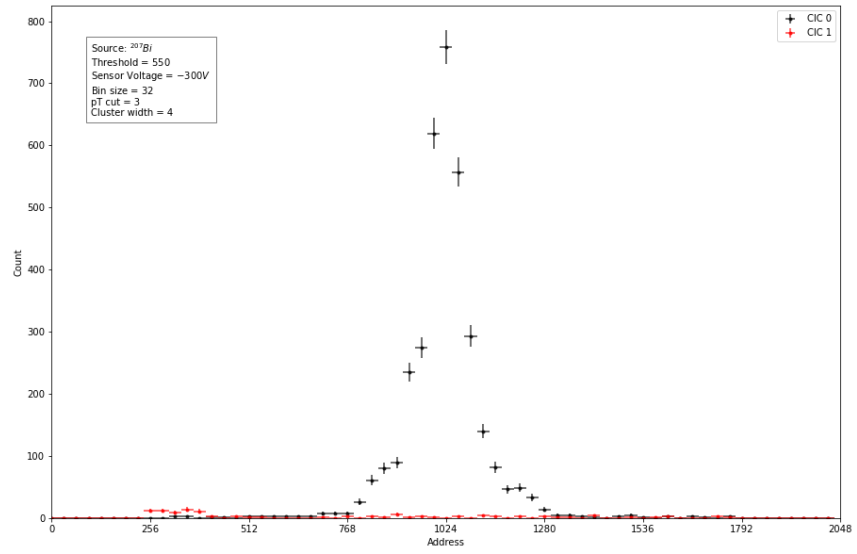
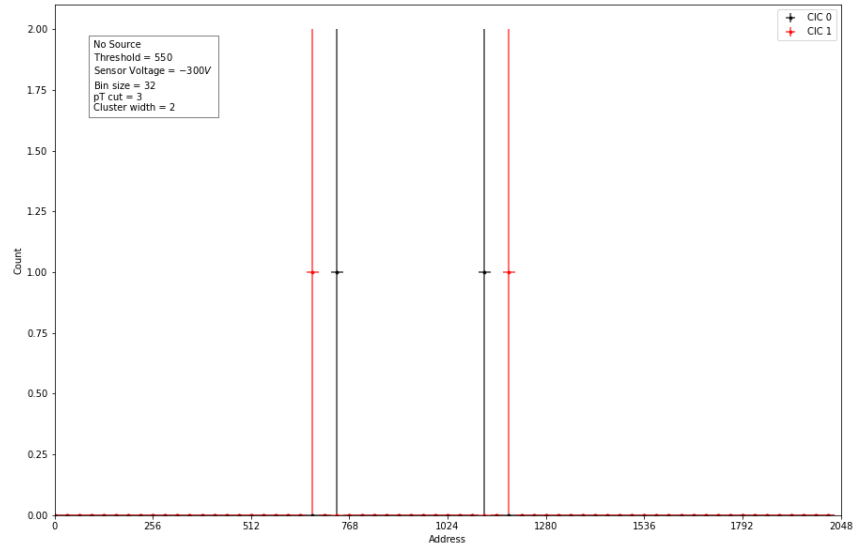


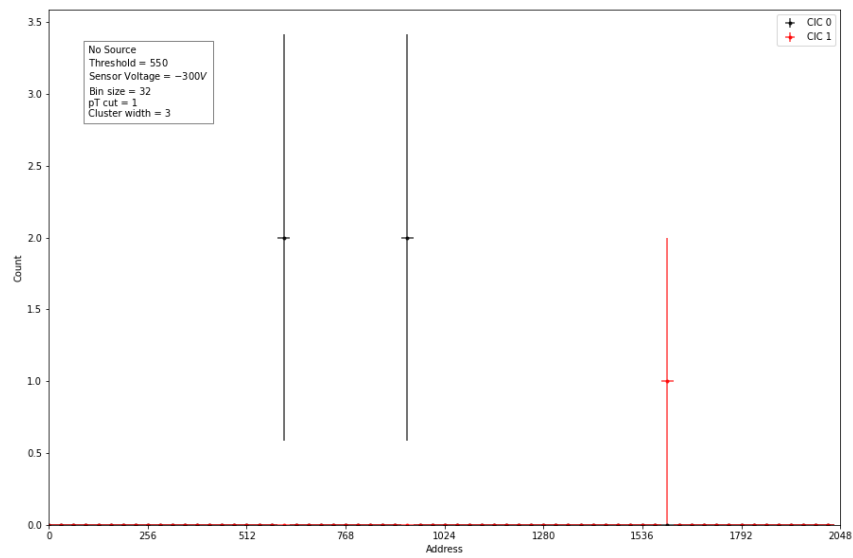
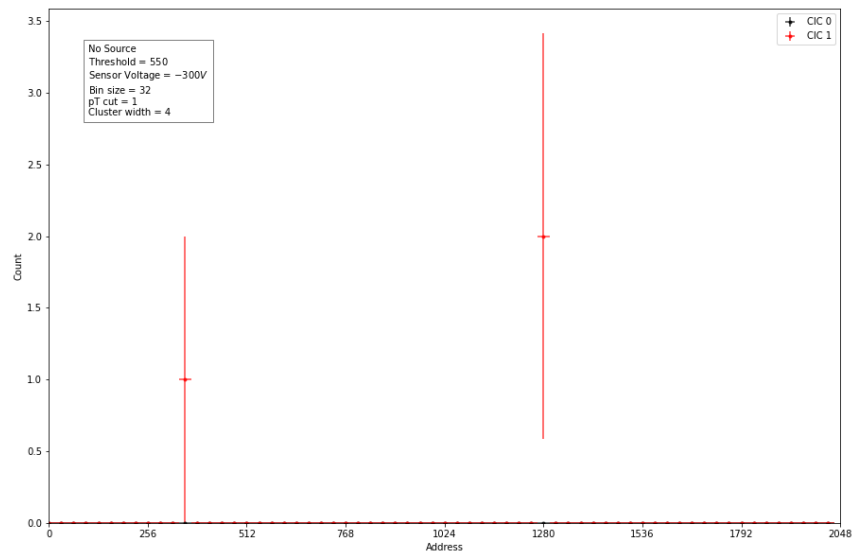
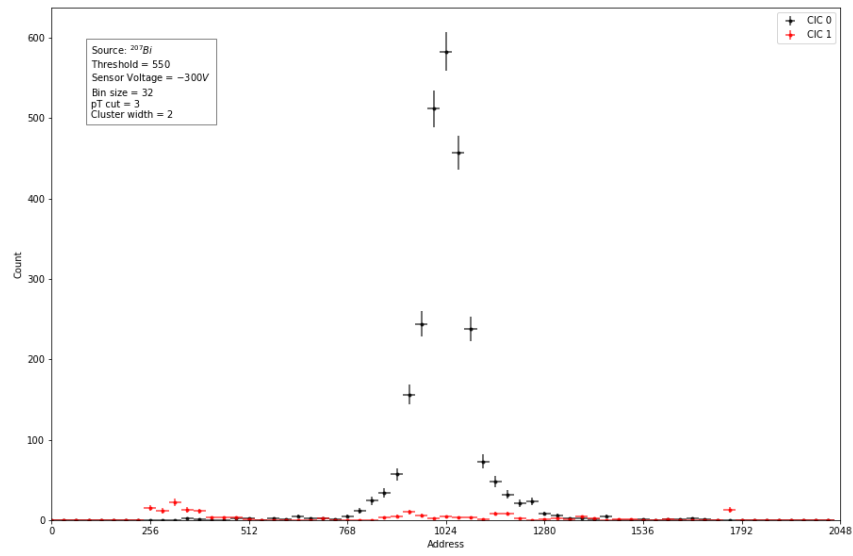
APPENDIX B. MODULE TESTING PLOTS



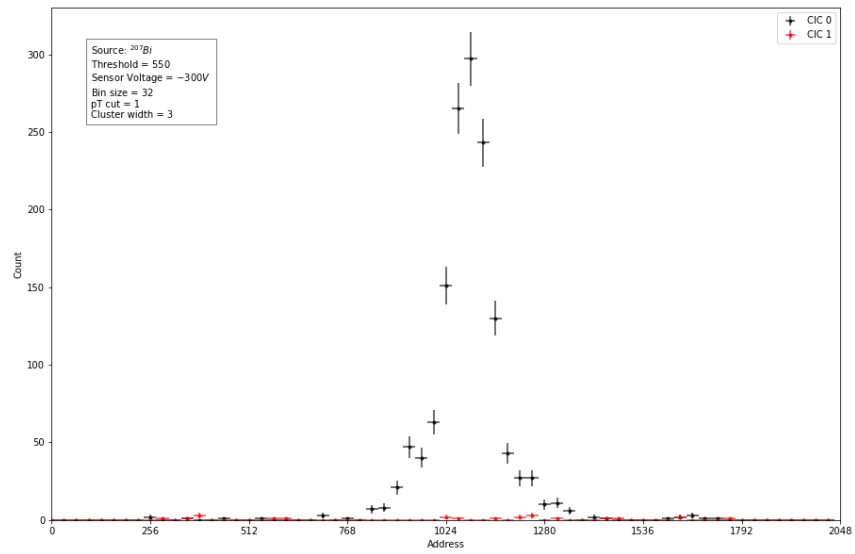
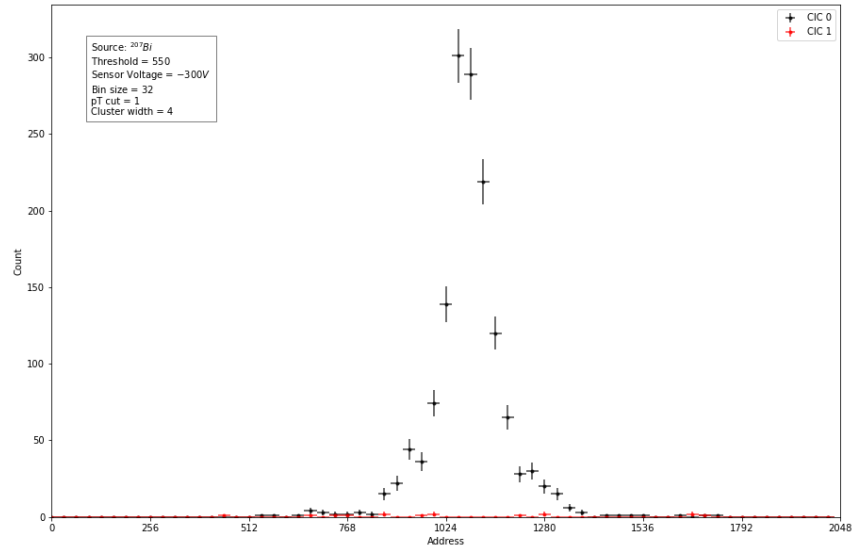
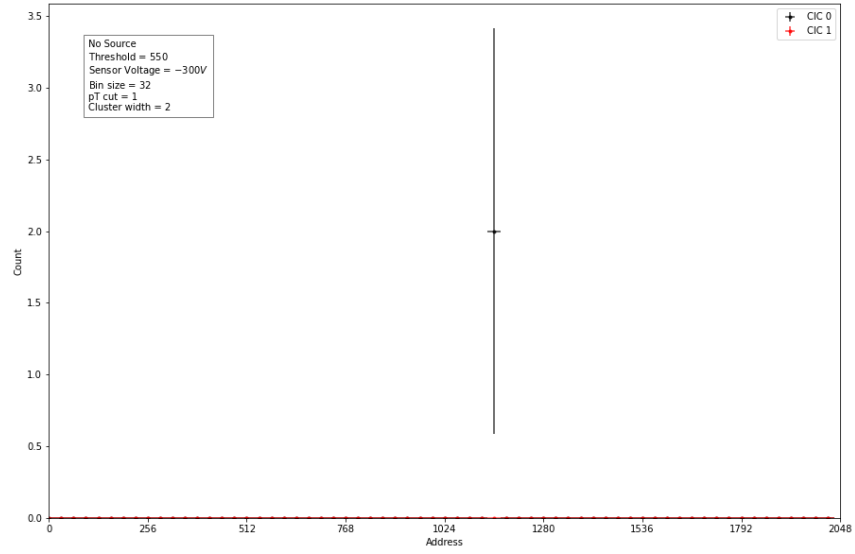


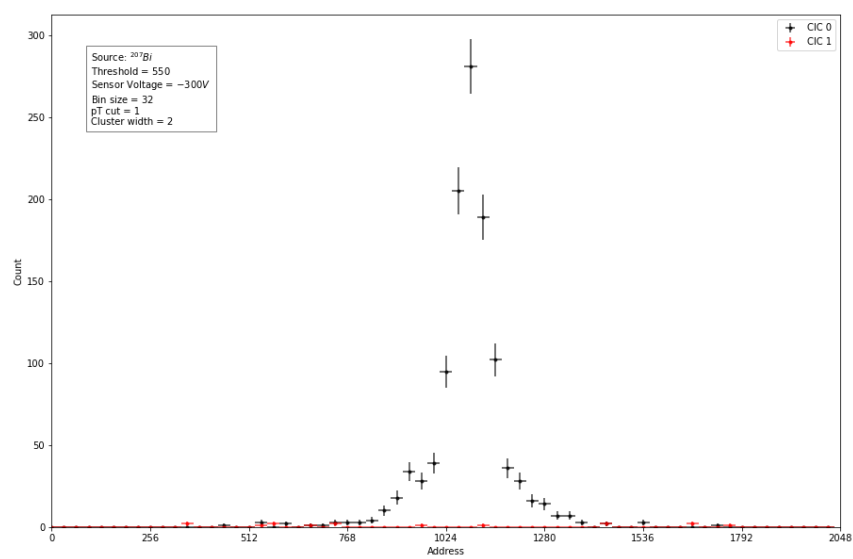
APPENDIX B. MODULE TESTING PLOTS





APPENDIX B. MODULE TESTING PLOTS





Appendix C

Beam Test Error Propagation

C.1 Special case for equidistant planes

To consider the difference in residual error between using interpolation and extrapolation to conduct the track fitting, a simplified model can be used. The model consists of three equidistant planes, upon which hits are recorded (see figure C.1). On each plane, the x value of the hits is normally distributed:

$$X_1 = N(\mu_1, \sigma_1) \tag{C.1}$$

$$X_2 = N(\mu_2, \sigma_2) \tag{C.2}$$

$$X_3 = N(\mu_3, \sigma_3) \tag{C.3}$$

Assuming the beam is collimated and the resolution of the the modules on each plane is identical, the variance in the distribution on each plane should be identical, thus σ_1 , σ_2 , σ_3 can be replaced by a single value, σ_m .

C.1.1 Interpolation

For the interpolation algorithm, in order to calculate the residual error between a track crossing the three planes, first the intersection of with the intermediate plane must be calculated:

$$x_{projection} = x_0 + \frac{1}{2}(x_2 - x_0) \tag{C.4}$$

$$= \frac{1}{2}(x_2 + x_0) \tag{C.5}$$

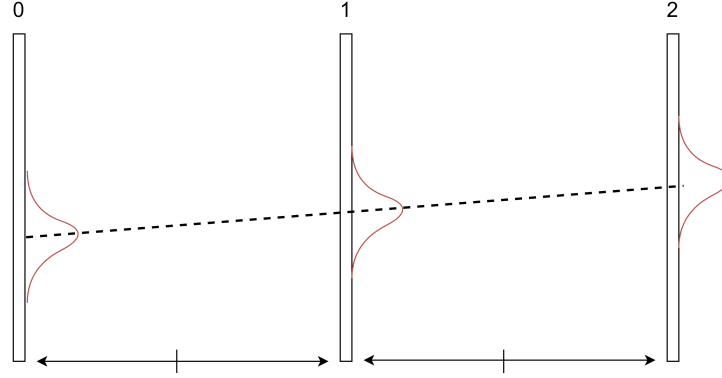


Figure C.1: Diagram showing the model used for deriving the residual distributions.

This is then compared with the recorded value in the intermediate plane to calculate the residual:

$$r = x_1 - x_{projection} \quad (C.6)$$

Considering X_1 , X_2 , X_3 as independent random variables, the above equations can be rewritten in terms of these random variables to give:

$$X_{projection} = \frac{1}{2} (X_2 + X_0) \quad (C.7)$$

$$X_r = X_1 - X_{projection} \quad (C.8)$$

where X_r is the distribution of the residual error. When performing addition and subtraction of normally distributed variables, the standard deviations of the two distributions are added in quadrature:

$$\sigma_3^2 = \sigma_1^2 + \sigma_2^2 \quad (C.9)$$

Thus, the standard deviation of the residual distribution can be calculated:

$$\sigma_{projection}^2 = \left(\frac{1}{2}\right)^2 (\sigma_2^2 + \sigma_0^2) \quad (C.10)$$

$$= \frac{1}{4} (\sigma_m^2 + \sigma_m^2) \quad (C.11)$$

$$= \frac{1}{2} \sigma_m^2 \quad (C.12)$$

$$\sigma_{projection} = \frac{\sigma_m}{\sqrt{2}} \quad (C.13)$$

$$\sigma_r^2 = \sigma_1^2 + \sigma_{projection}^2 \quad (C.14)$$

$$= \sigma_m^2 + \left(\frac{\sigma_m}{\sqrt{2}}\right)^2 \quad (C.15)$$

$$= \sigma_m^2 \left(1 + \frac{1}{2}\right) \quad (C.16)$$

$$\sigma_r = \frac{\sqrt{3}}{\sqrt{2}} \sigma_m \quad (C.17)$$

C.1.2 Extrapolation

When calculating the residual for the extrapolation case, a very similar method is used:

$$x_{projection} = x_1 + (x_1 - x_0) \quad (C.18)$$

$$= 2x_1 - x_0 \quad (C.19)$$

$$r = x_2 - x_{projection} \quad (C.20)$$

whereby the gradient is calculated between planes 0 and 1, then projected from 1 to 2. The standard deviation of the distribution of the residuals can also then be derived for this case:

$$\sigma_{projection}^2 = (2\sigma_1)^2 + \sigma_0^2 \quad (C.21)$$

$$\sigma_{projection} = \sqrt{5} \sigma_m \quad (C.22)$$

$$\sigma_r^2 = \sigma_1^2 + \sigma_{projection}^2 \quad (C.23)$$

$$= \sigma_m^2 (1 + 5) \quad (C.24)$$

$$\sigma_r = \sqrt{6} \sigma_m \quad (C.25)$$

Thus it's shown that the use of extrapolation over interpolation results in an increase in the spread of the residuals between predicted and actual position by a factor of 2.

C.2 Generalisation to arbitrary plane distances

The model used in the special case above can be generalised into a more complex model by allowing the separation between the three planes to be arbitrary. As shown in figure

C.2, these distances are now given the values z_{01} and z_{12} .

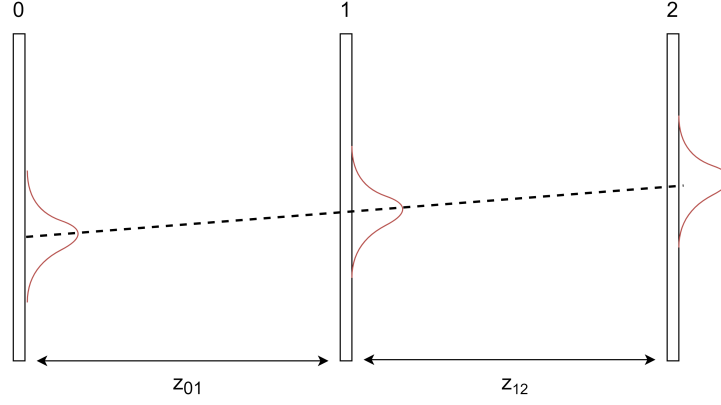


Figure C.2: Diagram showing the the extended model with arbitrary plane separations.

Equation C.4 can now be rewritten in terms of these new variables:

$$x_{projection} = x_0 + z_{01} \frac{x_2 - x_0}{z_{01} + z_{12}} \quad (C.26)$$

$$= x_0 + R(x_2 - x_0) \quad (C.27)$$

$$= x_0(1 - R) + x_2 R \quad (C.28)$$

where $R = \frac{z_{01}}{z_{01} + z_{12}}$. Using this formula for the projection of x , the standard deviation of the residuals can be again calculated:

$$\sigma_{projection}^2 = (\sigma_0(1 - R))^2 + \sigma_2^2 R^2 \quad (C.29)$$

$$= \sigma_m^2 \left((1 - R)^2 + R^2 \right) \quad (C.30)$$

$$= \sigma_m^2 (2R^2 - 2R + 1) \quad (C.31)$$

$$\sigma_{projection} = \sigma_m \sqrt{2R^2 - 2R + 1} \quad (C.32)$$

$$\sigma_r^2 = \sigma_1^2 + \sigma_{projection}^2 \quad (C.33)$$

$$= \sigma_m^2 + \left(\sigma_m \sqrt{2R^2 - 2R + 1} \right)^2 \quad (C.34)$$

$$= \sigma_m^2 (1 + 2R^2 - 2R + 1) \quad (C.35)$$

$$\sigma_r = \sigma_m \sqrt{2R^2 - 2R + 2} \quad (C.36)$$

Substituting in $R = 1/2$ for the case of equidistant planes, the result from equation C.17

is returned.

The same method can then also be repeated for the extrapolation algorithm:

$$x_{projection} = x_1 + z_{12} \frac{x_1 - x_0}{z_{01}} \quad (C.37)$$

$$= x_1 + R' (x_1 - x_0) \quad (C.38)$$

$$= x_0 R' + x_1 (1 + R') \quad (C.39)$$

where $R' = \frac{z_{12}}{z_{01}}$. The residual distribution then has the following standard deviation:

$$\sigma_r = \sigma_m \sqrt{2R'^2 + 2R' + 2} \quad (C.40)$$

Which again, by setting $R' = 1$ yields the result from equation C.25.

C.3 Multiple Scattering

A further source of error in the distribution of the track-fitting residuals arises from multiple scattering occurring whilst the particle passes through the station. This can, as shown in figure, can happen in the intermediate module, leading to the particle passing through the final module at an offset. This offset, shown in the figure as Δ comes as a result of the particle travelling at an angle offset from the original direction, labelled θ_{MS} .

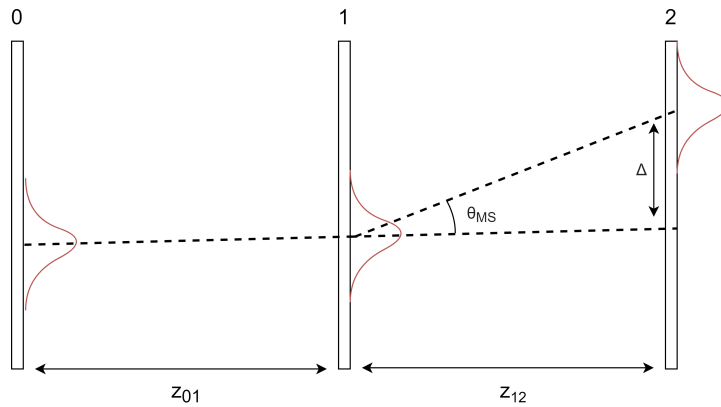


Figure C.3: Diagram showing a further extension with an additional multiple scattering event in the middle plane.

This extra correction can be integrated into equation C.26 for interpolation in the

following way:

$$x_{projection} = x_0 + z_{01} \frac{(x_2 + \Delta) - x_0}{z_{01} + z_{12}} \quad (C.41)$$

$$= x_0 (1 - R) + x_2 R + \Delta R \quad (C.42)$$

For estimating the error in the track fitting residuals, it is more useful to convert Δ to an angle using small angle formulae, to get $\Delta = z_{12}\theta_{MS}$, where the standard deviation of the angle of multiple scattering is σ_{MS} . The standard deviation of the projected x position is thus:

$$\sigma_{projection}^2 = (\sigma_0 (1 - R))^2 + \sigma_2^2 R^2 + \sigma_{MS}^2 z_{12}^2 R^2 \quad (C.43)$$

$$= \sigma_m^2 (2R^2 - 2R + 1) + \sigma_{MS}^2 z_{12}^2 R^2 \quad (C.44)$$

And for the residual:

$$\sigma_r^2 = \sigma_m^2 + \sigma_m^2 (2R^2 - 2R + 1) + \sigma_{MS}^2 z_{12}^2 R^2 \quad (C.45)$$

$$= \sigma_m^2 (2R^2 - 2R + 2) + \sigma_{MS}^2 z_{12}^2 R^2 \quad (C.46)$$

In the case of extrapolation, the correction for multiple scattering can be again applied, starting with equation C.37:

$$x_{projection} = x_1 + z_{12} \frac{x_1 - x_0}{z_{01}} + \Delta \quad (C.47)$$

$$= x_0 R' + x_1 (1 + R') + \Delta \quad (C.48)$$

Giving a standard deviation for the track fitted residuals of:

$$\sigma_r^2 = \sigma_m^2 (2R^2 + 2R + 2) + \sigma_{MS}^2 z_{12}^2 \quad (C.49)$$

List of Acronyms

2S Two Strip

ALICE A Large Ion Collider Experiment

API Application Programming Interface

ASIC Application Specific Integrated Circuit

ATCA Advanced Telecommunications Computing Architecture

ATLAS A Toroidal LHC Apparatus

BE Back End

BRAM Block Random Access Memory

Bx Bunch Crossing

CBC CMS Binary Chip

CI Continuous Integration

CIC Concentrator Integrated Circuit

CLI Command Line Interface

CMS Compact Muon Solenoid

CMSSW CMS SoftWare

CPU Central Processing Unit

COMPASS Common Muon and Proton Apparatus for Structure and Spectroscopy

CSC Cathode Strip Chambers

DAQ Data AcQuisition

DLL Delay-Locked Loop

DPS Data Processing Systems

DSP Digital Signal Processor

DTC Data, Trigger and Control

DTH DAQ and TCDS Hub

e-link External Link

ECAL Electromagnetic CALorimeter

EMP Extensible, Modular data Processor

FE Front End

FEC Forward Error Correction

FEH Front End Hybrid

FIFO First In First Out

FPGA Field-Programmable Gate Array

GBT GigaBit Transceiver

GEM Gas Electron Multiplier

GPIO General Purpose Input/Output

GUI Graphical User Interface

HCAL Hadronic CALorimeter

HGCAL High-Granularity CALorimeter

HDL Hardware Description Language

HLT High Level Trigger

HL-LHC High Luminosity Large Hadron Collider

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

HV High Voltage

I2C Inter-Integrated Circuit protocol

IaC Infrastructure as Code

IoT Internet of Things

IPBB IPBus Build

IT Inner Tracker

JSON JavaScript Object Notation

KIT Karlsruher Institut für Technologie

L1T Level-1 Trigger

LHC Large Hadron Collider

LHCb Large Hadron Collider beauty

L1 Level 1

L1A Level 1 Accept

lpGBT low-power GigaBit Transceiver

LUT Look Up Table

LV Low Voltage

MIP Minimally Ionising Particle

MPA Macro-Pixel Array

NFS Network File System

OT Outer Tracker

PCB Printed Circuit Board

PCIe Peripheral Component Interconnect express

Ph2ACF Phase-II Acquisition & Control Framework

PMT PhotoMultiplier Tube

PS Pixel Strip

PSB Proton Synchrotron Booster

PSU Power Supply Unit

PU Processing Unit

RAM Random Access Memory

RPC Resistive Plate Chambers

Rx Receiving

S3 Simple Storage Service

SoM System-on-Module

SPS Super Proton Synchrotron

SQL Structured Query Language

SSA Strip Sensor Array

SSH Secure SHell

TFP Track Finder Processor

Tx Transmission

UUID Universally Unique IDentifier

VHDL VHSIC Hardware Description Language

VL+ Versatile Link Plus