Development of SiTCP Based Readout System for The ATLAS Pixel Detector Upgrade

Osaka University, Graduate School of Science, Physics Department,
Yamanaka Taku Laboratory, Master Course 2^{nd} Year

Teoh Jia Jian

February 1, 2012

Abstract

The ATLAS pixel detector will be replaced in the future HL-LHC upgrades to preserve or improve the detector performance at high luminosity environment. To meet the tighten requirements of the upgrades, a new pixel Front-End (FE) IC called FE-I4 has been developed. We have developed a readout system for FE-I4 using a general purpose DAQ board called Soi EvAluation BoArd with Sitcp (SEABAS). This readout system is meant to be used as an electronic test stand for FE-I4 as well as a readout system for the module consisting of sensor and FE-I4. Our system incorporates Silicon Transmission Control Protocol (SiTCP) technology implemented on a FPGA which utilizes the standard TCP/IP and UDP communication protocols. This technology allows the direct access and transfer of the data between the readout hardware chain and PC via high speed Ethernet. In addition, the communication protocols are small enough to be implemented in a single FPGA. Because of this our readout system is very compact, versatile and fast. We have developed the firmware and software together with the readout hardware chains and established the basic functionalities for reading out FE-I4. This thesis details the functionalities of each component including the hardware, firmware and software and how they are integrated to form a functioning readout system. The thesis also provides a step by step guide on the general DAQ process flow. On top of that, in later part of this thesis various readout functionalities and their test results will be presented. Last but not least, numerous efforts which have been performed to optimize the readout speed are also shown.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 LHC and the ATLAS detector

Located about 100 meters beneath the France-Switzerland border near Geneva lies the 27 kilometers long Large Hadron Collider (LHC). It is the world largest and highest energy particle accelerator designed to investigate fundamental interaction through proton-proton (p-p) collision at center of mass energy up to 14TeV. The collider is designed to deliver a peak luminosity of 10^{34} cm^{-2}s^{-1} with 25 ns per bunch crossing for proton-proton collision.

A Toroidal LHC Apparatus (ATLAS) is one of four major detectors installed at the LHC to probe these collisions. It is a general-purpose experiment designed to explore various physics processes such as the search for the Higgs boson, H, supersymmetry (SUSY) and extra dimensions.

In 2011, the ATLAS recorded a total of 5.25 fb^{-1} of data for pp collisions at 7 TeV centre-of-mass energy. As of the end of proton run in December 2012, ATLAS has successfully recorded 21.7 fb^{-1} of data at center-of-mass energy, $\sqrt{s} = 8$ TeV, which is more than triple the amount collected in 2011. Around 2020, ATLAS will undergo a major upgrade to prepare for the high luminosity era in the so called High Luminosity LHC (HL-LHC) upgrade. This will significantly extend its opportunities to explore new physics and allows us to measure the property of the newly discovered Higgs-like particle more accurately.

1.1.1 Overview of the ATLAS experiment

Physics goals and detector requirements

One of the most important physics goals of ATLAS is the search for Higgs boson in order to understand the mechanism of the electroweak symmetry-breaking. For low mass SM Higgs ($m_H < 2m_Z$), $H \rightarrow \gamma\gamma$ would provide the cleanest channel for the Standard Model (SM) Higgs detection. Several other channels including associated production of Higgs boson such as $t\bar{t}H$, ZH, and WH with H decays to $b\bar{b}$ would be the important one as well. On the other hand, searches for Minimal Supersymmetric Standard Model (MSSM) Higgs such as A and $H^{\pm}$ will greatly depend on the detection of τ lepton and b quarks from processes such as $A \rightarrow \tau^{+}\tau^{-}$, $H \rightarrow \tau^{\pm}\nu$ and $H \rightarrow 2$ jets.

On top of that, the ATLAS experiment also aims for the evidence of the yet undetected SUSY particles. If they exist, they are predicted to decay mostly into energetic quarks or gluons and the lightest stable supersymmetric particle (LSP). Then the most likely signature of these superparticles would be the high energy jets and large missing transverse energy (E_T^{miss}), which signal the escape of LSP from the detector. Furthermore, the high rate production of heavy quarks such as top (t) and bottom (b) quarks will allow precision measurement of their mass and interactions with other particles.

All these physics goals define a set of general requirements for the ATLAS detector [1, 2]. In particular, the ATLAS detector needs to have an excellent electrons, muons and photon identification capability, accurate and high-resolution jet and E_T^{miss} measurement as well as excellent secondary vertex detection for τ leptons and b-quarks for various Higgs boson searches. On the other hand, SUSY searches require a large acceptance in pseudorapidity (η) and very good E_T^{miss} measurement and b-tagging performance.

Overview of ATLAS detector

Figure 1.1 shows the layout of the ATLAS detector. It is forward-backward symmetric with respect to the interaction point. From inside out is the inner tracker, the solenoid magnet, the calorimeter, the toroid magnet system and the muon spectrometer [3].

The inner part of the inner tracker consists of semiconductor pixel and strip detectors whereas the outer part is the transition radiation tracker (TRT) which

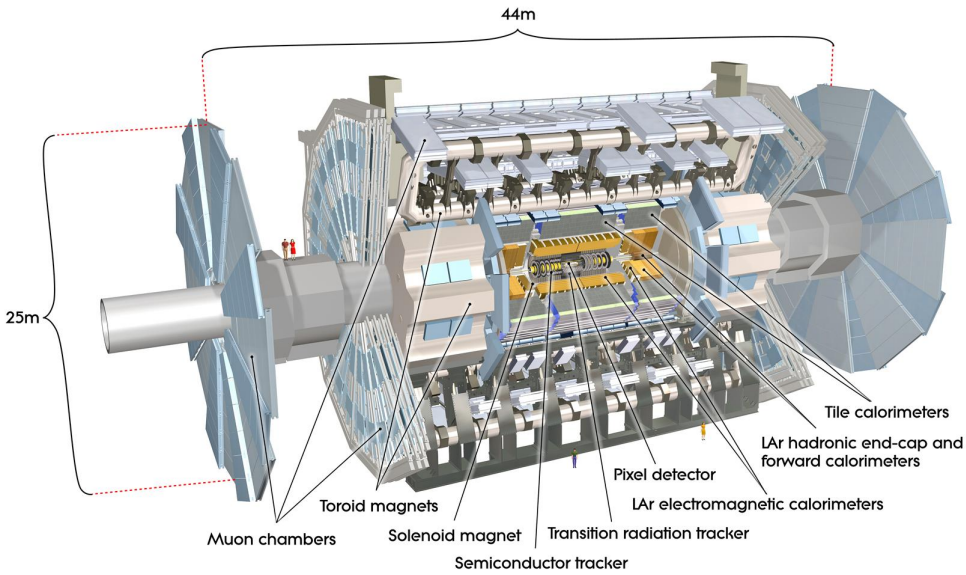


Figure 1.1: The ATLAS detector and its four major systems, i.e the inner tracker, calorimetry and magnet system as well as the muon spectrometer.

can detect transition radiation. The inner tracker is surrounded by a 2 T solenoidal magnetic field and is most important in providing high-resolution momentum and vertex measurement. The pixel detector will be described in more detail in the next section.

For precision measurement of electrons and photons, the ATLAS detector employs the Lead-Liquid Argon (LAr) electromagnetic sampling calorimeter with high granularity. On the other hand, the coarser scintillator-tile hadronic calorimeter provides jet reconstruction.

Surrounding the calorimeters is the muon spectrometer covering the pseudo-rapidity range $|\eta| < 2.7$. Couple with the large bending power of the toroid magnets, it provides precision momentum measurement for those muons exiting the calorimeters. In addition, the muon spectrometer is also capable of providing muon trigger with timing resolution of 1.5 to 4 ns.

1.2 Pixel detector

1.2.1 Semiconductor as charged particle detectors

The p-n diode junction

In most high energy physics application, semiconductors are doped with foreign material to alter their electrical property. Depending on the type of the added impurities, one obtains n-type or p-type semiconductors. When p and n-type semiconductor are brought together to create a junction, the difference in the number of electrons and holes causes a diffusion of majority carriers across the junction. This migration and electrons-holes recombination leave a region of net charge of opposite sign on each side. An electric field is induced across the junction which eventually stops the diffusion process. This is depicted in Figure 1.2.

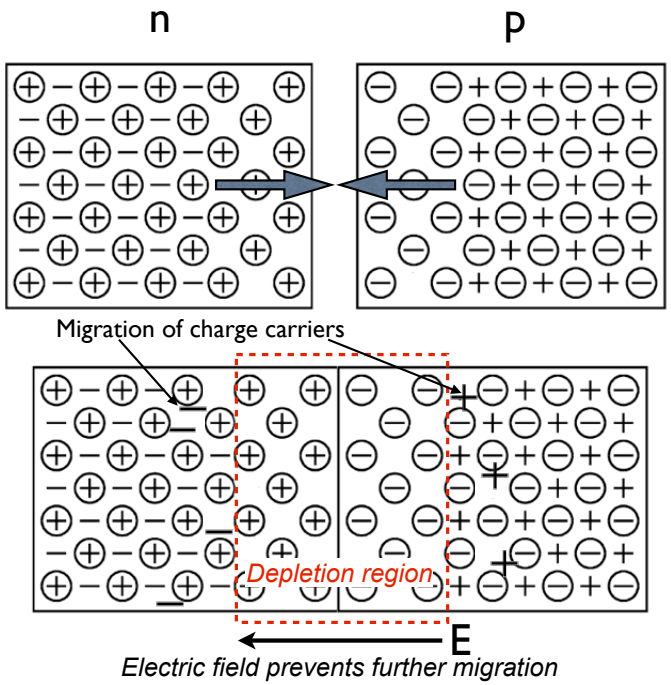


Figure 1.2: A p-n diode junction in thermal equilibrium when p and n-type semiconductor are brought together.

The region of immobile charges left behind after the mobile charge carriers have diffused away is called the depletion region. This region has an attractive characteristic that recombination can not happen here as there are no electrons or holes left behind. In addition, any charged particle entering this region will be swept out by the electric field. Thus electrons or holes created by any ionizing particles

traversing this region will be swept out and creates signal that is proportional to the ionization. These principles can be exploited for charged particle detection.

The depletion depth

Referring to the schematic of a uniformly charged p-n junction in Figure 1.3, the N_D and N_A denote the donor and acceptor impurity concentration. Likewise, the d_n and d_p are the how deep the depletion zone extent into the n-side and p-side respectively.

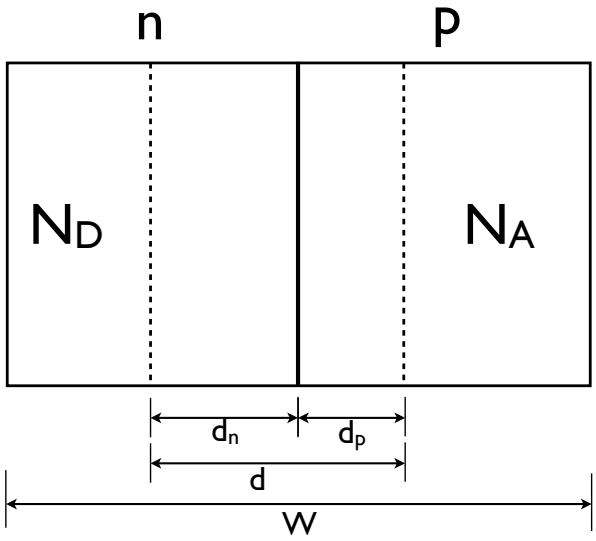


Figure 1.3: Schematic for depletion zone.

Without external bias voltage, the depletion zone is generally small. The total thickness of the depletion zone, d is given by [7]:

$$d = d_p + d_n = \left(\frac{2\varepsilon V_o}{e} \frac{(N_A + N_D)}{N_A N_D} \right)^{1/2} \tag{1.1}$$

where e is the electron's charge, V_o is the contact potential and ε is the dielectric constant. For the case of p^+-n junction, meaning $N_A \gg N_D$, then the thickness, d is approximately,

$$d \simeq d_n \simeq \left(\frac{2\varepsilon V_o}{e N_D} \right)^{1/2} \tag{1.2}$$

or by using the expression:

$$\frac{1}{\rho_n} \simeq eN_D\mu_e$$

we can express Equation 1.2 as

$$d \simeq (2\rho_n\mu_e\varepsilon V_o)^{1/2} \tag{1.3}$$

where ρ_n is the n-bulk resistivity and μ_e is the mobility of the electrons. By replacing the $\rho_n\mu_e$ by $\rho_p\mu_h$, we can get the thickness of depletion region at the p-n$^+$ junction.

The p-n diode under external voltage

Thermal equilibrium will be destroyed when an external bias voltage is applied across the junction. For a forward bias, the contact voltage will decreases and the thickness of the depletion zone will shrink. This is not favorable for particle detection. The remedy is to apply a reverse bias voltage across the junction with positive voltage at n side while negative voltage at p side. This will enlarge the depletion zone and thus the detection sensitivity of the device. Figure 1.4 illustrates the enlarged depletion zone under reverse bias voltage.

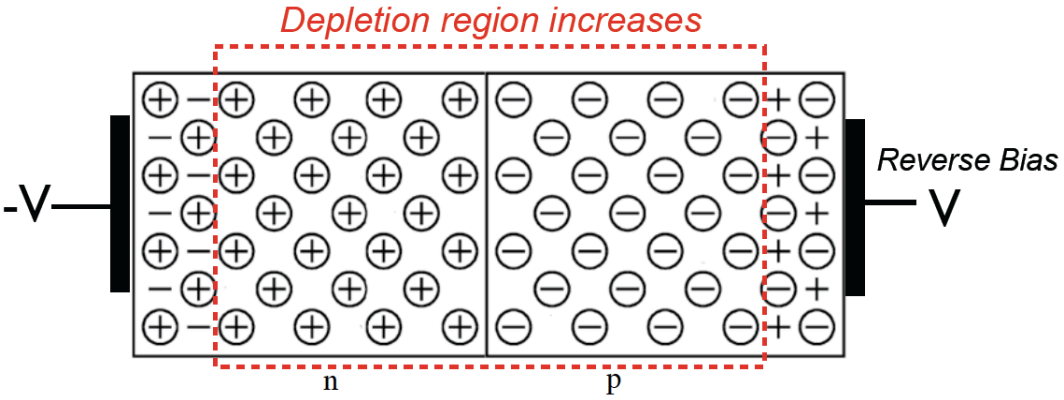


Figure 1.4: By applying reverse bias voltage, the depletion region can be widen.

From Equation 1.3 and Figure 1.3, we can see that in order to fully deplete the substrate ($d = W$), the depletion voltage, V_d must have the value that is given by:

$$V_d = \frac{\mathrm{W}^2}{2\varepsilon\rho_o\mu_o} \tag{1.4}$$

where the subscript "o" can be replace by n or p in the case of the p^+-n junction or the p-n^+ junction respectively. However, usually this maximum depletion voltage will not be achieved due to breakdown (*Zener* breakdown and *Avalanche* breakdown [9]) of the junction. To achieve higher depletion width, beside increasing the V_d, it is necessary to use higher resistivity material [9].

Creation of electron-hole pairs in semiconductors

In general, heavy charged particles lose energy by Coulomb interaction with the atomic electrons and elastic scattering from nuclei of the material. Only in the former electron-hole pairs are created. Because of its small mass, electron interacts slightly different from heavy charged particles. Apart from the interaction with atomic electron, electron also undergoes radioactive process (bremsstrahlung) at high energies. Whereas for γ or X rays, they interact with the material through photoelectric effect, the Compton effect, and the pair-production effect depending on the photon's energies.

The details of energy loss process when a charged particle traversing a semiconductor are complicated but the average energy, ε necessary to create an electron-hole pair in a given semiconductor at a fixed temperature is independent of the type and the energy of the ionizing radiation. Table 1.1 gives the minimum energy required to create an electron-hole pair in Si and Ge at room and cryogenic temperature.

Table 1.1: Average energy for electron-hole creation in Si and Ge [7].

Temperature (K)	Si	Ge
300	3.62eV	–
77	3.81eV	2.96eV

Since the forbidden band gap for Si and Ge is only of the order of 1 eV at these temperatures, we can see that only about 30% of the absorbed energy is actually spent in breaking the covalent bonds. Another point to take notice of is the small value of ε compared with the ionization energy of gas (typically 15 to 30 eV). This represents a superior spectroscopic performance of semiconductor detectors.

As the thickness of the semiconductor sensor increases, the energy loss E by a charged particle also increases. In case of silicon sensor, a minimum ionizing particle (MIP) traversing the silicon sensor will generate about 80 electron-hole

pairs per μm. Hence, for a typical ATLAS pixel detector's silicon sensor with thickness of about 250μm, roughly 20000 electron-hole pairs will be generated by a MIP traversing through the sensor. As we shall see later, due to this reason the pixel front-end readout IC is tuned to this charge value.

1.2.2 Position Measuring Silicon Pixel Detector

By using the readout electronics that measure signal charge, energy deposited in the sensor by charged particle can be measured. The 2-D position information can be obtained by splitting the signal charge among several electrodes and then measure the ratio of charge at each electrode. Alternatively, dividing the detector into smaller section that are read out separately may also help. Dividing a detector active area into pixels, one obtains a position sensitive detector where the position resolution is linear to the pixel pitch size.

A pixel detector has the advantages [8] of unambiguous hits, unlike the strip detector that produces ghost hits at high occupancy as can be seen in Figure 1.5. Besides, the smaller segmented area also reduces the capacitance with typical value close to 1fF/pixel. This also means larger signal to noise ratio. Furthermore smaller pixel volume also leads to lower leakage current ($\approx$ 1pA/pixel) which means lower noise value.

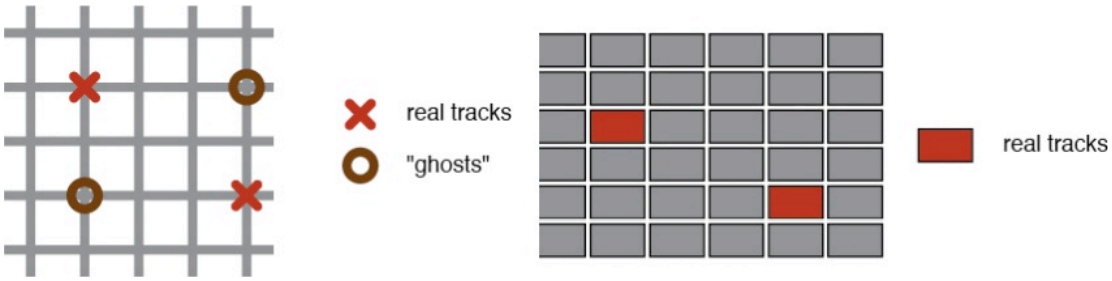


Figure 1.5: Pixel detectors produce unambiguous hits [8].

Despite that, the pixel detector needs large number of readout channels, which means large number of electrical connections and power consumption. Due to its small size, it is extremely expensive to cover large area. Thus, the pixel detector is only suitable for particle detection at the innermost region near the interaction point.

Hybrid pixel detector

One of the most common pixel detector types for charged particle detection is the hybrid pixel detector which consists of two separate but geometrically matching array of detectors and custom designed readout Application-Specific Integrated Circuit (ASIC). The readout IC has multi-channel and provides various functional components such as amplification, filtering, storage and so on. These arrays of electronics and the 2-D diode arrays are usually built on a separate substrate. They are then connected by the small conducting bumps applied by using the bump bonding and flip-chip technology [9, 11]. This hybird pixel detector structure can be seen in Figure 1.6.

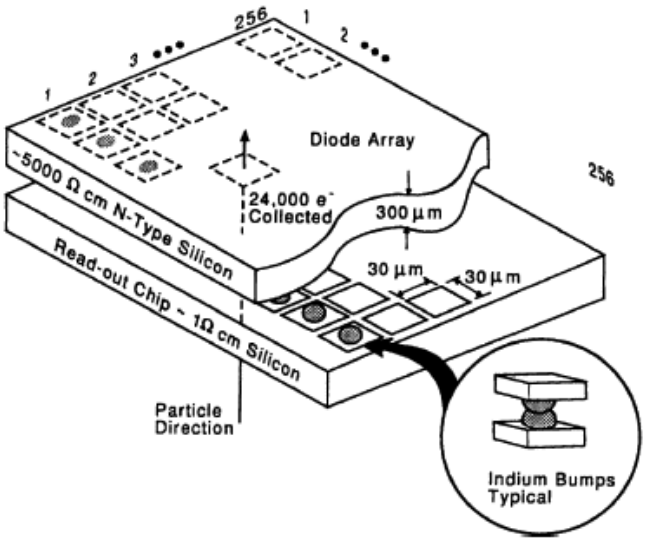


Figure 1.6: Schematic of a typical hybrid pixel detector. The zoom-in shows the indium solder bump bonds that connect the detector to readout electronics [9].

There are a variety of readout architecture and one typical example is the "column-based structures". In this type of readout system, each column of pixels is independently treated. Signal generated by traversing charged particle is amplified, filtered and temporarily stored in the pixel where it waits for the readout trigger. Upon receiving the readout trigger, the signal is then stored in an end-of-column buffer that will be read out asynchronously. This method is advantageous in the sense that any error in one pixel will affect only one column instead of the whole device.

As mentioned above, the position measurement is achieved through segmenting a large-area diode into many small pixel-like regions and to read them out

separately. Referring to Figure 1.7, the position sensing process can be explained succinctly as follows:

1. Charged particles traversing the depletion region create electron-hole pairs.
2. These charges subsequently drift to the oppositely charged electrodes.
3. Electronics in the readout ASIC amplifies, shapes and filter the signal created by the drift charges.
4. Finally, the segment showing the signal gives the position of path of the ionizing particles.

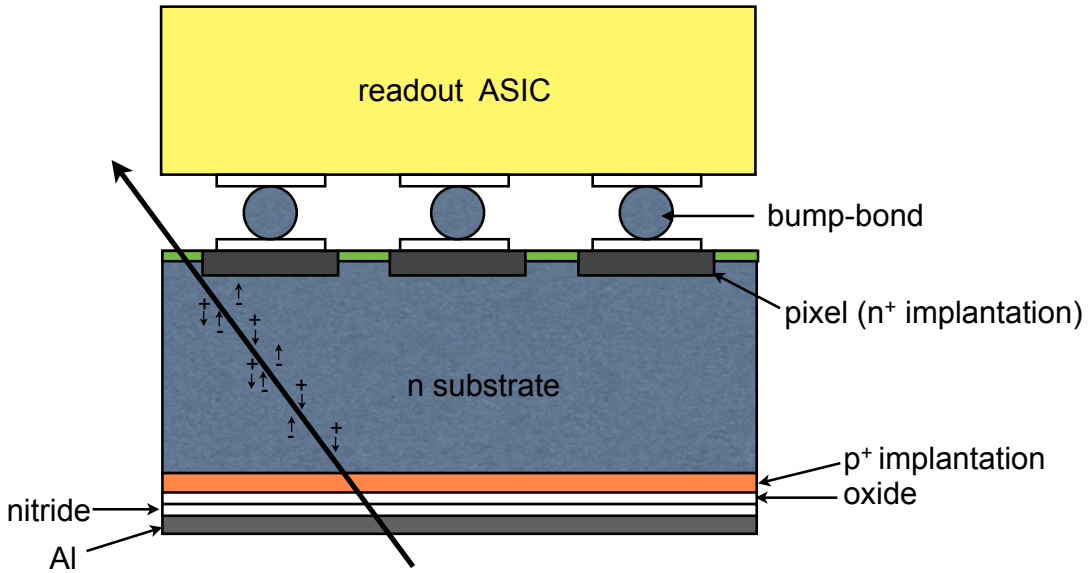


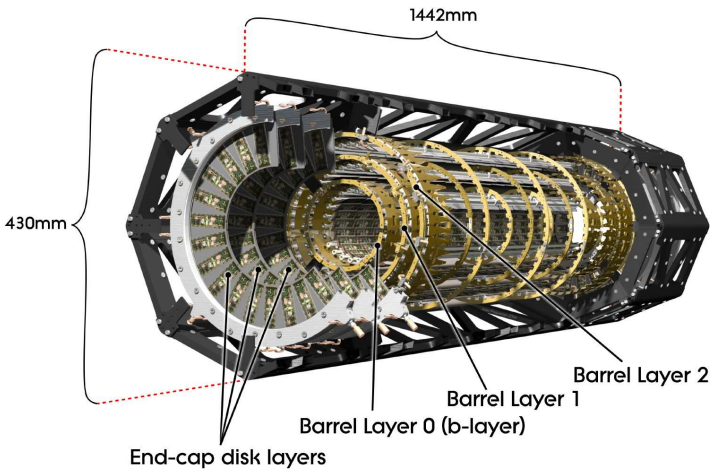
Figure 1.7: Schematic for position measurement by discrete detector. [8]

The ATLAS pixel detector is exactly based on this hybrid pixel technology and will be discussed next.

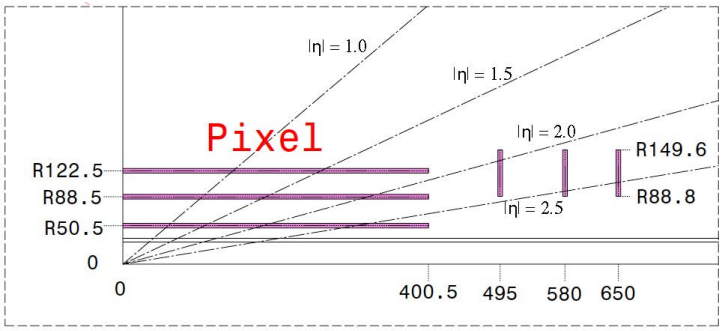
1.3 The ATLAS pixel detector

1.3.1 Performance requirements

The pixel detector as shown in Figure 1.8 (a) is the innermost component of the ATLAS detector. It has a total of about 80 million channels distributed among



(a)



(b)

Figure 1.8: (a) The ATLAS pixel detector consists of 3 barrel layers and 3 end-cap disk layers at each ends, and (b) Plan view of a quarter-section of the pixel detector and its active dimension. [4].

three barrel layers and six end-cap disk layers and covering a total active area of about 1.7 m^2 [4]. As can be seen in Figure 1.8 (b) Barrel layer-0 is positioned at radius of 50.5mm from the beam axis whereas barrel layer-1 and 2 have radius of 88.5mm and 122.5mm, respectively. It is designed to cover the region $|\eta| < 2.5$ and arranged in a way such that each track will typically cross a minimum of three pixel layers.

As described in previous section, the physics goals of the ATLAS experiment dictate the performance requirements of the detectors. For the pixel detector, excellent vertexing performance and robust pattern recognition is a must. Detailed description of the performance specification for the pixel detector can be found in

Reference [5, 6], however the general performance requirements can be summarized here:

1. Excellent 3D vertexing capability and transverse impact parameter resolution

   The transverse impact parameter resolution, σ_d is required to be better than about $15\mu m$. The primary vertex reconstruction of charge tracks with longitudinal resolution, σ_z < 1mm is also desired [4, 5]. In addition to this, secondary vertex reconstruction from b-hadron decays also has to be very accurate.

2. High pattern recognition efficiency

   The requirement for the efficiency for high p_T isolated track reconstruction is $\geq$95% with a fake rate of $\leq$1%. Other than this, the reconstruction efficiency for all tracks with p_T $\geq$0.5 GeV should be $\geq$95% as well [5].

1.3.2 The pixel sensor and readout electronics

The 80 million pixels are arranged into 1744 pixel modules with identical function at the sensor and integrated circuit level. Each pixel module consists of a 256$\pm$ 3μm thick n^+-in-n silicon sensor with 6.08 x 1.62cm^2 of active surface. The sensor contains 47232 pixels which are individually bump bonded to 16 Front End (FE) readout ICs. All the FE ICs in turn are connected to a module-control chip (MCC) which provides the necessary communication with the off-detector electronics via opto-links [4]. The schematic cross section of the ATLAS pixel detector module is illustrated in Figure 1.9. The back and front side of the ATLAS pixel detector module with 16 FE readout ICs is shown in Figure 1.10.

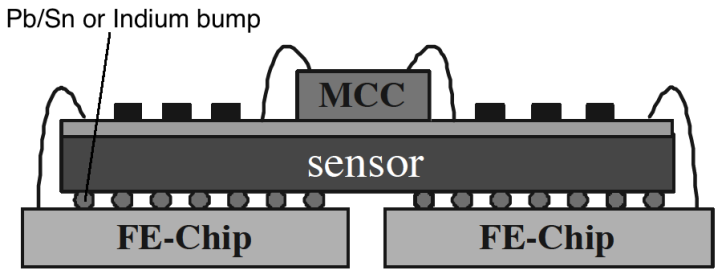
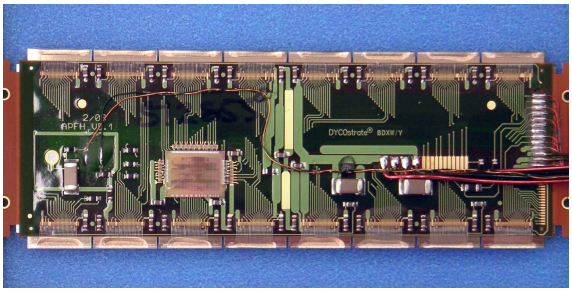
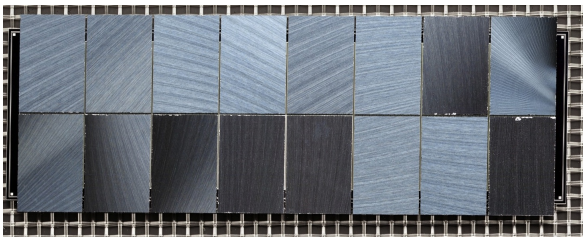


Figure 1.9: A not to scale cross section of the ATLAS pixel module. [12]



(a)



(b)

Figure 1.10: (a) The back side of the pixel module with MCC. (b) The FE side of the pixel module, 16 FE chips are connected to a single n^+-in-n silicon sensor.

Current readout IC for the ATLAS pixel detector is called FE-I3. It contains 2880 pixel cells arranged in a 18 x 160 matrix. Each pixel cell is 50 x 400 μm^2 in size and is fabricated in the IBM 0.25 μm CMOS technology. Due to the high radiation environment at region close to the beam pipe, this pixel FE readout IC has some strict requirements to follow, among those are [13]:

1. Fast timing
   The LHC beams are operated at 40MHz with roughly 2800 collisions per 90μs bunch revolution time. An unique crossing ID has to be associated with each event. To meet this requirement, a very fast front-end design with peaking time of about 30ns and the capability of accurately assigning unique crossing ID to every hit from contiguous crossings are required.

2. Radiation hardness
   The pixel electronics should remain within specification after a lifetime radiation dose of 500kGy or $10^{15} n_{eq} cm^{-2}$ (where $n_{eq} cm^{-2}$ is an equivalent 1 MeV neutron fluence producing the same bulk damage in a specific semiconductor in non-ionizing energy loss (NIEL) scaling). At a maximum luminosity of $10^{34} cm^{-2} s^{-1}$, the innermost layer is expected to reach this radiation dose in about 5 years while 10 or more years for the other layers.

3. Low power consumption per readout channel
   A limit of 40 μW per channel is set for the FE-I3 to facilitate the cooling of the sensor and electronics.

4. Threshold, noise and signal height.
   After the exposure to lifetime fluence of $10^{15} \mathrm{n}_{eq} \mathrm{cm}^{-2}$, radiation damage causes the sensor's charge collection efficiency to decrease and the leakage current to increase. The front-end readout IC must have the ability to cope with leakage current of up to 100nA per pixel while keeping the system efficiency higher than 97%. Additionally, the readout IC must also be able to cope with a threshold as low as 2000 electrons and a noise of a about 200 electrons.

5. High data rate
   The readout architecture must be designed to record and readout hits data at more than 99% efficiency with hit intensity of up to 5 x 10^7 hits/cm^2/sec. Data loss due to analog signals delay, insufficient buffer size, multiple hits etc. must be kept to a minimum level.

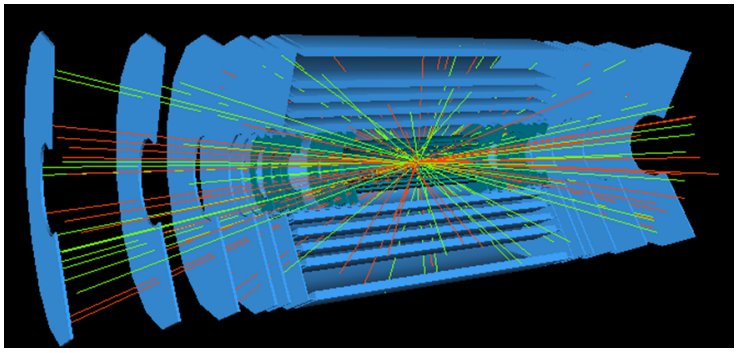
1.4 ATLAS pixel detector upgrade

The high-radiation environment imposes stringent conditions on the pixel detector sensors and electronics. Radiation damage will gradually degrade the sensor and its electronics and in turn its detection performance. To preserve the tracking performance after a few years of operation or for the operation at even higher luminosity environment, those defect pixel modules have to either be replaced or some compensation scheme must be made in place. Besides, over the next 10 years, the LHC will undergo a series of upgrades in particular the High Luminosity LHC (HL-LHC) upgrade where higher luminosity would be achieved. To keep up with this environment, the ATLAS detector has to undergo a major modification as well.

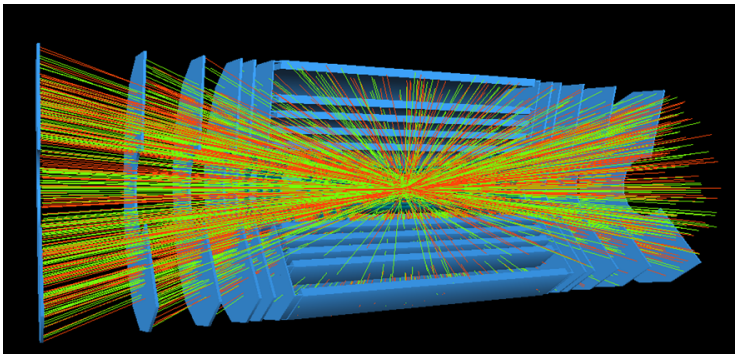
1.4.1 High Luminosity LHC upgrade

Around year 2022 LHC will undergo a major upgrade under the so called HL-LHC project. The ultimate goal is to collect $3000 fb^{-1}$ of data by the year 2030. To

achieve this goal, the instantaneous luminosity will have to be increased to 5 x 10^{34} $cm^{-2}s^{-1}$. This increase in luminosity will greatly enhance the discovery potential of LHC but there is a high price to pay. At this luminosity, the number of pile-up events is expected to reach ∼140 per bunch crossing. As a comparison, the nominal operation of LHC at design luminosity of $10^{34}cm^{-2}s^{-1}$ will generate roughly 28 pile-up events per bunch crossing [14, 15]. This harsher radiation environment and higher event rate poses unprecedented challenges to the detectors especially those which are closer to the interaction region. Thus for the ATLAS detector a complete replacement of its inner tracker which consists entirely of silicon pixel and silicon strip detectors is foreseen.



(a)



(b)

Figure 1.11: The layout of the new Inner Detector accompanying the HL-LHC upgrade. Detector occupancy illustration (simulated) under 23 pile-up events (a) and 10 times increase in pile-up events after the luminosity upgrade (b) [14].

Currently the new inner tracker is still under intensive design and development phase. However, the baseline design consists of 4 pixel and 5 silicon strip barrel layers while the end-caps have 6 pixel and 5 silicon strip disk layers at each end [14]. The current estimates for the HL-LHC ATLAS inner tracker replacement are

roughly 6 m^2 and 150m^2 for pixel detector and silicon strip detector respectively [16]. Figure 1.11 illustrates the baseline layout of the new inner tracker and the simulated pile-up events before and after the luminosity upgrade.

1.4.2 Pixel FE readout electronic in high luminosity era

Motivated by the pixel detector upgrade, a newer generation of front-end readout IC which can satisfy the criteria of higher granularity, better radiation resistivity and lower material budget has been developed. The new readout IC is called FE-I4 [17, 19] and is the largest readout IC produced to date in high energy physics experiment. It is designed in a 130nm CMOS technology but in the future pixel readout IC fabricated in 65nm CMOS technology is foreseen [18]. The goals are to achieve the same or even better tracking and vertexing performance than the current pixel detector and at the same time give better radiation tolerance.

A more detailed description of the new FE-I4 readout IC will be given in Chapter 2.

1.5 Research theme and scope

The main theme of my research is to design and develop a readout system which can act as an electronic test bench for FE-I4 and provide various functionalities for the IC characterization and performance analysis. It can also serve as a DAQ system for pixel module with various silicon sensor technologies such as the planar pixel sensor, 3D silicon sensor or even the diamond pixel sensor.

This research is mainly the continuation of the previous works that have been kick-started by Y. Takubo (KEK) et al.. Throughout this research, we will mainly focus on the further design and development of the software and firmware of the DAQ system to read out FE-I4. Another main topic is to find out ways to improve the readout speed. Testings of FE-I4 were carried out mainly to demonstrate the practicability and functioning of the readout system as well as to serve as an assessment for the correctness of the readout system implementation.

1.6 Research motivations and objectives

This research is mainly motivated by the advent of the new ATLAS pixel FE readout IC. Our motivations and goals that we wish to achieve in this undertaking are as the following:

1. The need of a new readout scheme.

   (a) FE-I4 is considered as a second generation pixel readout IC for the ATLAS pixel detector with many novel new features in comparison to its predecessor. Examples of such features that will directly affect the data readout include the number of pixels, output data rate, charge measurement resolution, readout logic, configuration registers, etc. The development of the original readout system for FE-I3, TurboDAQ system, has long been stopped and the adaptation of its software and hardware to read out FE-I4 is not feasible. Even though there are well-establish alternatives to this system, a decision has been made to design a new readout system from sketch by incorporating several new technology which we think could make our system faster, more flexible and versatile.

   (b) The current design concept for the pixel module for the ATLAS HL-LHC upgrade is a 4-chip module. The 2 by 2 matrix is based on the arrangement limitation as well as the requirements of reducing the amount of materials, sharing of data, command and clock line among chips, benefit of localization in the instance of malfunctioning and so on. Hence it is highly desired to read out 4 chips simultaneously. Nonetheless, the development of the 4-chip readout scheme was postponed due to hardware problem and thus will not be covered in this thesis.

   (c) Currently there exist two well established FE-I4 readout systems namely the Reconfigurable Cluster Element (RCE) system and the USBpix system. The RCE system is optimized for exploring new DAQ architectures for ATLAS upgrade applications. It is very fast and is capable of reading out multiple 4-chip modules. Yet, it is based on a ATCA packaging standard which means the the hardware package is complex and huge. On the other hand, the USBpix is a very compact modular readout system. However, currently it is not yet ready for reading out

multi-chip module. Hence, our goal is to design a readout system that could fuse together the merit of each systems, i.e fast, compact and versatile.

2. Compactness
   As a pixel FE electronics test system, a more flexible and compact hardware setup is more attractive and preferable than a bulky one based on the VME or CAMAC system which require large dedicated hardware. Our aim is to create a system that is both lightweight and highly portable in view of the fact that the readout IC and the sensor testing often need to be carried out in different facilities around the world.

3. Flexibility and versatility
   In terms of software functionalities, a user wants a system that is highly adaptable. There will be instance where a user needs to device their own test routines to extract relevant data which cannot be obtained by the standard or preset test routines. This is especially true during the period of extensive readout IC testing and characterization where the acquisition of new information is essential to fully understand the IC's behavior. It is our goal to design a software and firmware framework that can be rapidly adapt to suit the user needs by using existing block without involving any extensive modification to the structure.

The objectives which we aim to achieve are:

1. to prove that our readout system is able:
   (a) to communicate with FE-I4 and correctly carry out configuration.
   (b) to read out the FE-I4.
   (c) to correctly decode and extract relevant data from the output data stream.

2. to provide and test essential functionalities for FE-I4 testing.

3. to evaluate and improve as much as possible the readout speed.

1.7 A preview of the readout system

It is in the reader interest that an overview of our readout system be given before moving on to the following chapters. Figure 3.2 and Figure 3.3 shows the single chip setup. We have a general purpose DAQ board which is connected to the Single Chip Card (SCC) via an adapter card. The main DAQ board communicates with the PC via an Ethernet connection. The software takes care of all the control routines while the communication firmware and data flow control firmware are implemented in the two Field Programable Gate Array (FPGA) onboard the main DAQ board.

In the next chapter, detailed description of the pixel sensor and its front-end electronic as well as the description of FE-I4 will be given. Chapter 3 will give an in-depth discussion on various components of the readout system and the implementation details of the firmware and software. Various software functionalities and test results will be presented in Chapter 4 while readout system's performance evaluation will be made available in Chapter 5. Finally the last chapter will be dedicated to a brief discussion on the outlook of these research.

Chapter 2

The New ATLAS Pixel FE Readout Chip —FE-I4

2.1 Motivation for the new FE chip

The performances of the pixel detector after the LHC luminosity upgrade have prompted a redesign of the current pixel FE-I3 readout chip. Studies have shown that current FE-I3 cannot handle the high hit rates environment, leading to unacceptable inefficiency for HL-LHC. Figure 2.1 shows the schematic of the FE-I3's column-drain based readout architecture and the arrangement of pixels. FE-I3's pixel arrays are organized in Double Column (DC) with the buffers located at the End Of Double Column (EODC). The FE-I3 readout architecture employs a column-drain based data transfer concept which means that each pixel with the comparator fires will send timing, charge and address information down to the End Of Double Column buffers. Until the data transfer is finished, the active pixel will be unavailable to record the next hit. As the hit rate increases, the Double Column bus becomes saturated and as a consequence, the inefficiency starts to increase.

As can be seen in Figure 2.2, the inefficiency of the FE-I3 chip is no longer tolerable starting from about three times the LHC design luminosity. The inefficiency caused by double-hit or pile-up is directly proportional to the hit rate and the pixel size. Since all the pixels in the same Double Column share the same Double Column data bus, while one pixel is transferring its hit data, the other must wait. During this waiting time, no new hit can be stored in the local buffer and hence the new hit is lost. This is called as the busy/waiting inefficiency. Every hit that is recoded by the pixel will be associated with a time stamp called bunch

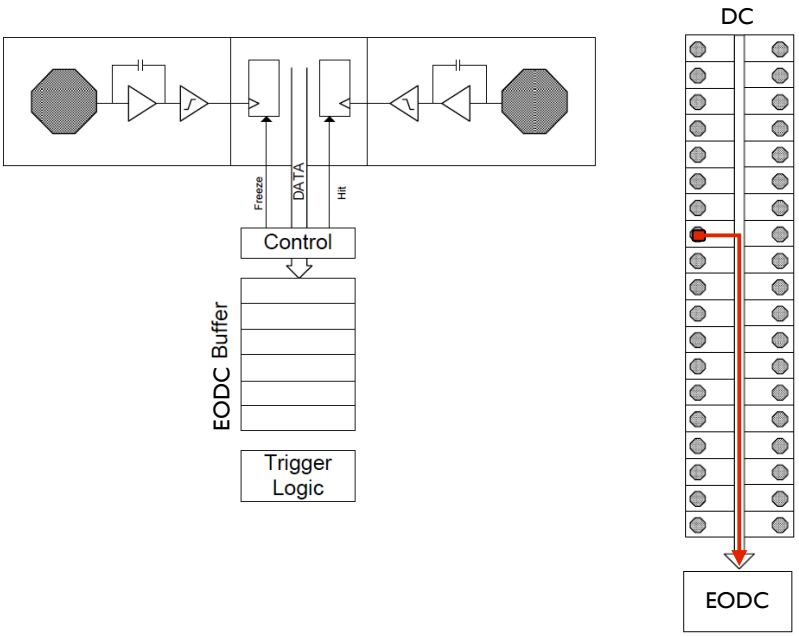


Figure 2.1: The standalone pixel logic, the Double Column and the End Of Double Column (EODC) buffer concept of the FE-I3. The pixel that fires stays inactive until its data is transferred to the next available end of column buffer [21].

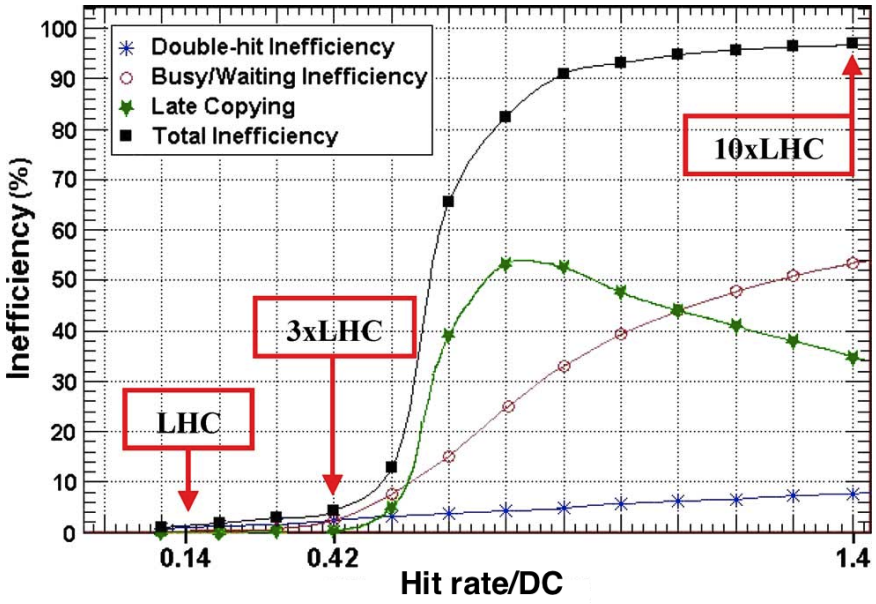


Figure 2.2: Inefficiency of FE-I3 readout chip at r=5cm as a function of hit rate per double column. At 3 times the LHC design luminosity, the inefficiency starts to increase drastically up to an unacceptable value. [20]

crossing ID by the pixel digital circuit. At the End Of Double Column buffer, the Level 1 trigger time stamp will be compared with the bunch crossing ID. If these two have the same time stamp, the hit data will be read out. However when the hit is associated to a wrong bunch crossing ID due to late copying, the hit will not be read out. This data lost contributes to the so called late copying inefficiency. Both of these inefficiency rise drastically starting from approximately three times the LHC nominal luminosity. Thus, in order to preserve the tracking performance at the luminosity beyond the current LHC design value, a new front-end readout Integrated Circuit (IC) is needed.

2.2 FE-I4 specification and design

The following discussion is mostly about what is most relevant to our readout system development. There are currently two FE-I4 versions namely FE-I4A and its improved version, FE-I4B. More details can be found in references [30, 31].

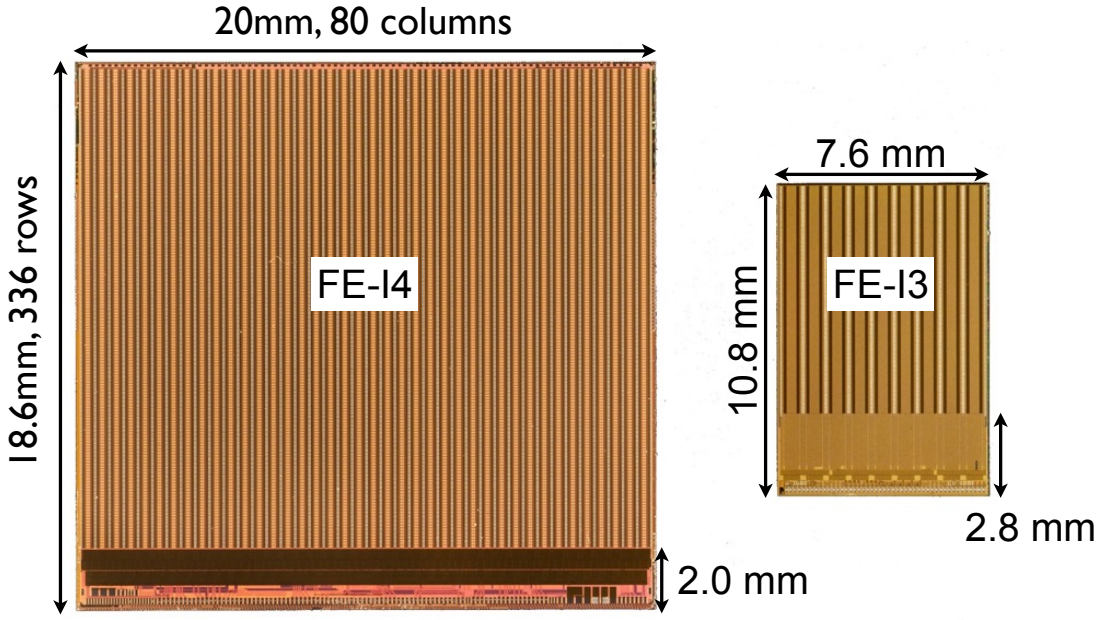


Figure 2.3: The physical size of FE-I4 and FE-I3 pixel readout IC

Figure 2.3 shows the physical size of FE-I4 and FE-I3. FE-I4 is fabricated in a 130 nm CMOS process. It is the largest front-end IC used in high energy physics experiment to date with a total of 26880 pixels arranged in 80 x 336 matrix on a 20 x 18.6 mm^2 chip. The larger chip size leads to more efficient system integration and

thus reduces the amount of material per detector layer. Furthermore, larger chip size also leads to significant manufacturing cost reduction. The pixel size is about 40% smaller than its predecessor at 50 x 250 μm^2. This reduction in pixel size significantly reduces the pile-up inefficiency and improves single point resolution in z-direction. Moreover, FE-I4 has an active area close to 90%. In comparison, FE-I3 only has an active area of 74%. This higher active area of FE-I4 reduces the material and greatly enhances the vertexing capability.

Moreover, to reduce the cable budget and power losses, both the digital and analog current is limited to 10μA/pixel. All Input/Output (I/O) of the FE-I4 are LVDS signal. The nominal I/O bandwidth is raised four times higher than FE-I3 at 160Mb/s. The comparison between FE-I3 and FE-I4 is listed down in Table 2.1. A more complete FE-I4 specification can be found in Appendix A.

Table 2.1: Comparison of the specification between FE-I3 and FE-I4 [13, 17]

	Value		Units
	FE-I3	FE-I4	
Chip Size	7.6 x 10.8	20.0 x 18.6	mm^2
Pixel Size	50 x 400	50 x 250	μm^2
Pixel Array	18 x 160	80 x 336	Col x Row
Active Fraction	74	89	%
Analog Voltage	1.6	1.5	V
Digital Voltage	2.0	1.2	V
Analog Current	16	10	μA/pixel
Digital Current	10	10	μA/pixel
Data Rate (nominal)	40	160	Mb/s

2.2.1 Layout and numbering convention

Figure 2.4 shows the layout of the FE-I4 pixels and their numbering convention. All the pixels are organized in column pairs. A single column pair consists of 672 pixels which are divided into one Digital Double Columns (DDC) or we call it Double Column for short and two analog columns. An analog column contains the pixel analog circuit which is bonded to the sensor. Each Double Column is

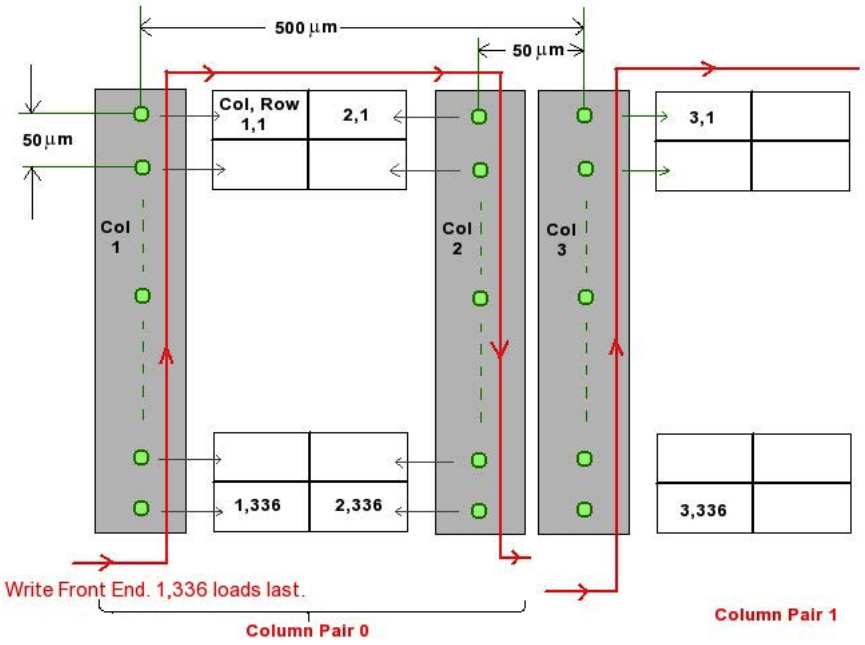


Figure 2.4: The pixel numbering convention. The white box correspond to the DDC while the gray columns correspond to the analog columns. Pixel (1, 1) is at the top left corner. The green dots represent the bump bonding location. [31]

bounded by an analog columns at each sides. Double Columns are numbered from 0 to 39 while the analog columns are numbered from 1 to 80. The Double Column-0 contains the analog column 1 and 2, the Double Column-1 contains the analog column 3 and 4, and so on. All operation on FE-I4 is based on the Double Column numbering scheme, thus it is worth taking some time to familiarize with it. Each pixel address is specified by its column and row number as (Column, Row).

2.2.2 Digital architecture

Thanks to the smaller feature size process, the local buffers can be made next to each pixel instead of putting them all at the End of Double Column. This feature allows the hit data to be stored locally inside the double column bus without moving around unless it is requested by L1 Trigger. By not transferring unnecessary hit data, the source of inefficiency as in FE-I3 at high hit rate environment can be reduced. Instead, what is left as a dominant source of inefficiency in FE-I4 is the single pixel pile-up inefficiency. Simulation on FE-I4 with real physics data has shown that the pile up inefficiency at 3.7cm from the beam axis could be kept as low as ∼0.42% at a luminosity of 3 x 10^{34}cm^{-2}s^{-1} [20].

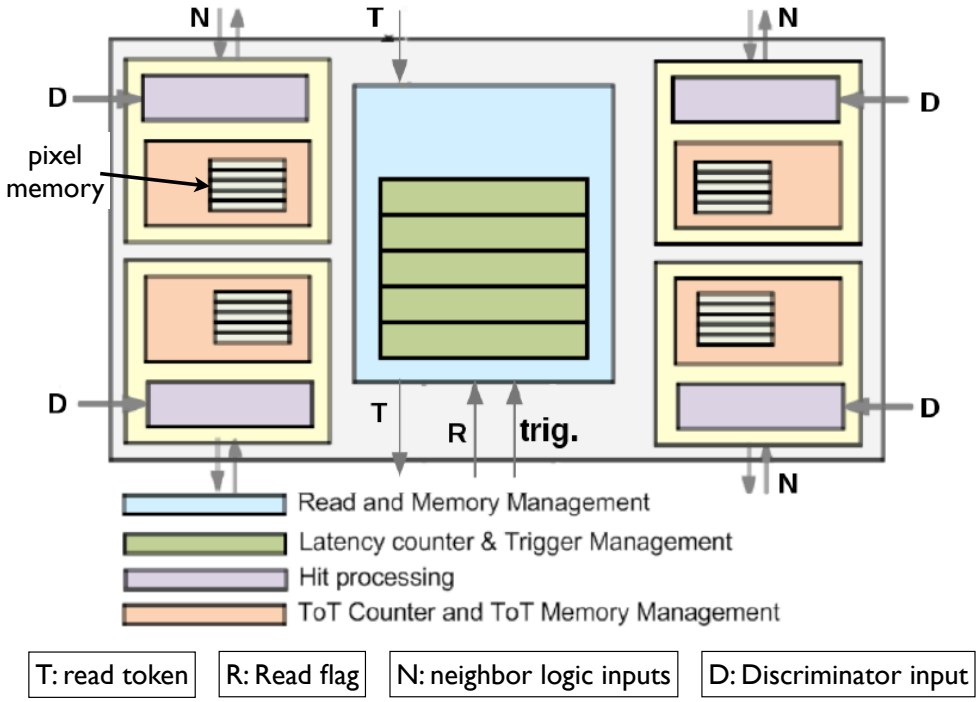


Figure 2.5: The FE-I4 digital architecture. Unlike FE-I3, FE-I4 implement shared pixel logic and local data storage. Hit data is not transferred to End of Double Column Logic unless a trigger is received at the correct timing. [31].

As can be seen in Figure 2.5, four pixels are tied together to form a central Pixel Digital Region (PDR). This has the effect of area reduction and power saving. Each Pixel Digital Region contains four groups of hit processing logics, Time Over Threshold (TOT) counter and memory manager as well as 5 pixel memories. The region has 5 latency counters and trigger management units which are mapped one to one to the pixel memories. At any given time, at least one Time Over Threshold counter must be idle so that one of the pixel can record a hit.

Besides the discriminator in the pixel analog circuit, there is yet another digital discriminator in the pixel digital logics. This digital discriminator imposes a digital threshold to the width of the analog discriminator output of each pixel. Essentially it is a Time Over Threshold comparator. Depending on the setting of this digital discriminator, even if a signal passed the screening of the analog discriminator, if the signal's Time Over Threshold values is too small it will not be qualified for reading out. Thus, throughout this thesis a qualified hit refers to a hit that produces a signal that passes both the analog and the digital discriminators.

2.2.3 Analog readout chain

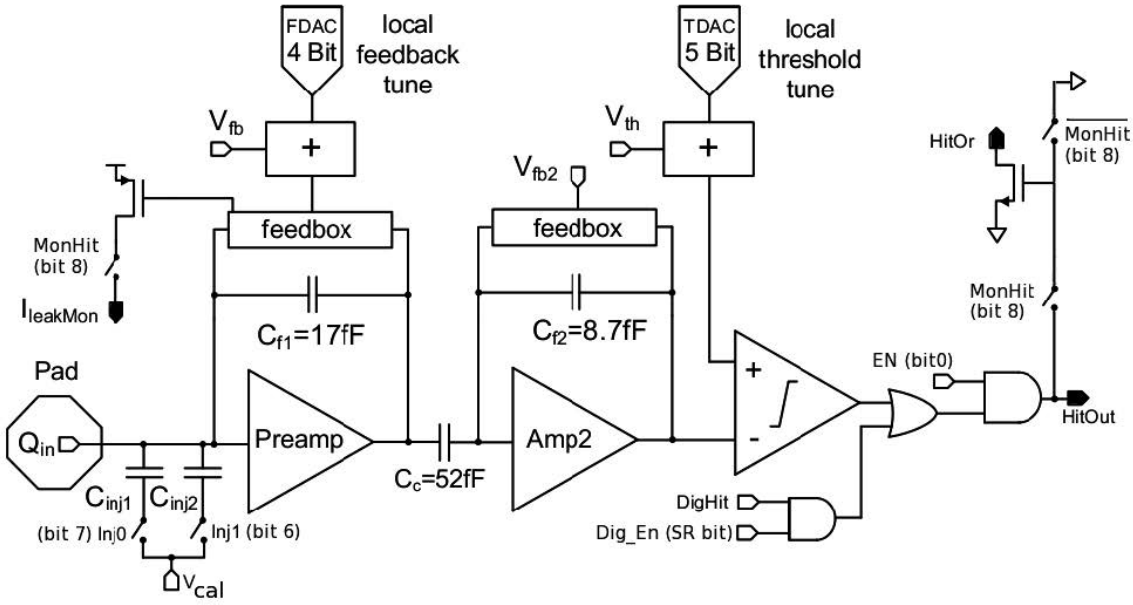


Figure 2.6: The schematic diagram of the analog pixel for FE-I4. [31]

Figure 2.6 illustrates the analog pixel schematic. From left to right are the input pad from sensor, the internal charge injection circuitry, the pre-amplifier first stage and the amplification second stage followed by the discriminator. The sensor is DC coupled to FE-I4. The charge injection circuitry consists of a large and a small charge injection capacitor with capacitance of 3.90fF and 1.95fF respectively. Besides providing a high signal gain, the preamplifier also has an adjustable shaping provided by the feedback circuitry. This feedback circuitry gives the preamplifier a linear return to baseline, which in turn leads to a discriminator pulse width that is proportional to the input charge. Therefore we can use the Time Over Threshold value to obtain the signal amplitude. In addition, this feedback filter of the preamplifier also provides compensation to the DC leakage current from the sensor with a leakage current tolerance up to 100 nA [31].

The pre-amplifier is AC coupled to the second stage. The ratio of the coupling capacitance (C_c) to the second stage feedback capacitance (C_{f2}), C_c/C_{f2} gives an additional amplification factor to the pre-amplifier feedback capacitance, C_{f1} [31]. This increase in feedback capacitance has the advantages of increased charge collection efficiency, less power consumption and higher signal rise time [21]. Finally the discriminator is made out of a 2-input voltage comparator and a threshold

generator which has a global control and can also be adjusted locally through the 5-bit Threshold Trim DAC (TDAC) register.

There are two local tuning registers in the pixel analog circuit namely the TDAC and the Feedback DAC (FDAC). The TDAC is used to tune the discriminator threshold. It is 8-bit complement, which means smaller TDAC value corresponds to higher threshold value and vise versa. On the other hand, the FDAC register is meant for adjusting the feedback current of the preamplifier. Larger FDAC value means larger feedback current.

2.3 Operation and control

Everything about operating the chip has been clearly described in reference [30, 31]. Thus, only a concise description relevant to the DAQ development will be given here.

2.3.1 Registers

There are two kinds of registers. First is the global register which affect the parameters value that is common to all the pixels on FE-I4. Second is the local pixel register that control parameters for each pixels. FE-I4A is controlled by 75 global registers whereas FE-I4B has 81 global registers. Both of them have 13 Single Event Upset (SEU) hard local pixel registers for the control of the analog pixel logic. All the global registers have different length and are grouped into 35 sixteen-bit long words. In addition, there is a 672-bit long shift register (SR) which is mapped one to one to all the pixels in a Double Column. It is used for reading and writing the 13 SEU hard pixel registers. A complete description about all the register types and their functions are available in chapter 8 of reference [30, 31].

2.3.2 I/O protocol

The data transfer to and from FE-I4 is through the serial pseudo-LVDS (Low Voltage Differential Signal) signal synchronized by an external clock. It is called as pseudo-LVDS because the FE-I4 driver does not use the standard IEEE LVDS 1.2V offset standard. Nonetheless, the pseudo-LVDS driver can still communicate with commercial LVDS drivers and receivers. The LVDS circuit has a maximum

clock rate of 320MHz but nominally 160MHz is used for data output whereas input rate is 40MHz.

2.3.3 Command and command decoder

FE-I4 uses a very simple serial protocol for all the configuration commands. Commands are subdivided into three classes, i.e "Trigger", "Fast", and "Slow" commands. Both of the Trigger and Fast command type only work while FE-I4 is in the "Run Mode". While FE-I4 is set to "Configuration Mode", only Slow commands will be accepted by the command decoder. Different command types can be easily identified by their unique header as can be seen in Table 2.2. Trigger command (LV1) is the shortest with 5 bits in total. It triggers the acquisition of new hit data from FE-I4. Fast command are slightly longer with 9 bits in total. The BCR and ECR fast commands are responsible for reseting the bunch and event counters respectively. The CAL command is used for issuing a calibration pulse in FE-I4.

Table 2.2: Unique identifier for different command types. LV1 is the Trigger command type while BCR, ECR and CAL belong to Fast command type [31].

Name	Field 1	Field 2		
size (bits):	5	4	Description	Command Type
LV1	111011	-	Level 1 Trigger	Trigger
BCR	10110	0001	Bunch Counter Reset	Fast
ECR	10110	0010	Event Counter Reset	Fast
CAL	10110	0100	Calibration Pulse	Fast
Slow	10110	1000	Slow command header	Slow

Table 2.3 shows the different Slow commands and their interpretation. The Slow commands are mainly related to reading and writing the global registers, the shift register and the local pixel registers. They are also used to issue the Global Reset command and the global pulse which has various functionalities. On top of that, Slow commands can also be used to switch between the "Run Mode" and "Configuration Mode". Unlike the Trigger and Fast command type, the Slow commands may have 4 other fields in addition to the header part. Note

that the Data part (Field 6) of the WrRegister command is 16 bit long while for WrFrontEnd command is 672 bit long. For RunMode command, the Mode (Field 5) should be set to 111000 for Run Mode and 000111 for Configuration Mode.

Table 2.3: Slow commands types and their format. [31].

Name	**Field 3**	**Field 4**	**Field 5**	**Field 6**	
size (bits):	4	4	6		**Description**
RdRegister	0001	ChipId	Address	-	Read addressed global memory register
WrRegister	0010	ChipId	Address	Data	Write into addressed global memory register
WrFrontEnd	0100	ChipId	xxxxxx	Data	Write conf data to selected shift register(s)
GlobalReset	1000	ChipId	-	-	Reset command; Puts the chip in its idle state
GlobalPulse	1001	ChipId	Width	-	Has variable pulse width and functionality
RunMode	1010	ChipId	Mode	-	Sets RunMode or ConfMode

The FE-I4 command decoder handles the decoding of all the command input and is also responsible for the generation of reset signal for the logic. It is worth noting that the decoder is not multi-treated which means it can only handle one command at any single time. There is no way to send multiple commands at the same time.

2.3.4 Encoding and data format

8b/10b encoding

The FE-I4 output data is 8b/10b encoded by default and can be turned off if needed. The 8b/10b encoding is crucial for clock recovery, data framing and phase alignment. The code specifies the encoding of a 8-bit symbol (256 unique data words) and an additional 12 special (K-Code Group) characters into a 10-bit symbol. Detailed implementation of 8b/10b can be found in the paper by A.X. Widmer and P.A. Franaszek of IBM [22].

The 12 special symbols, referred to as control symbols, are shown in Table 2.4. These control symbols are unique in that their bit pattern never occurs in a string of data symbols. Among them, K.28.1, K.28.5, and K.28.7 are "comma symbols" which are used in FE-I4 for data stream framing and alignment. These frames are:

1. **IDLE**

   The FE-I4 Data Output Block (DOB) is always sending out IDLE code

whenever there is no data to transfer. The IDLE code is the K28.1 comma.

2. **EOF**
   The K28.5 comma is used as the End of Frame (EOF). The EOF serves the purpose of separating two events. It can also be used in frame synchronization when events follow closely one after another (no IDLE state).

3. **SOF**
   The Start of Frame (SOF) is the K28.7 comma. It marks the start of a valid transmission and is also used in frame synchronization.

Table 2.4: The 12 unique K-Code symbols. FE-I4 uses the K28.1, K28.5 and K28.7 symbols for data stream framming and synchronization.

Code Group	kin/kout	8-bit DIN HGF EDCBA	10-bit DOUT (RD-) abcdei fghj	10-bit DOUT (RD+) abcdei fghj	Frame
K28.0	1	000 11100	001111 0100	110000 1011	
K28.1	1	001 11100	001111 1001	110000 0110	IDLE
K28.2	1	010 11100	001111 0101	110000 1010	
K28.3	1	011 11100	001111 0011	110000 1100	
K28.4	1	100 11100	001111 0010	110000 1101	
K28.5	1	101 11100	001111 1010	110000 0101	EOF
K28.6	1	110 11100	001111 0110	110000 1001	
K28.7	1	111 11100	001111 1000	110000 0111	SOF
K23.7	1	111 10111	111010 1000	000101 0111	
K27.7	1	111 11011	110110 1000	001001 0111	
K29.7	1	111 11101	101110 1000	010001 0111	
K30.7	1	111 11110	011110 1000	100001 0111	

Record formatting

FE-I4 data output is a single treaded serialize bit stream. Before 8b/10b encoding, three 8-bit raw data words are grouped into a 24-bit long word called record. There are six record types as shown in Table 2.5.

Valid data output from the Data Output Block needs to follow certain sequences. For example, a valid pixel data sequence is shown in Figure 2.7

Table 2.5: Record types for FE-I4B. Bit order follows the [MSB : LSB] convention. [31]. FE-I4A has slightly different bit order and length for LV1ID and BCID.

Record	**Field 1**	**Field 2**	**Field 3**	**Field 4**	**Field 5**
Data Header (DH)	11101	001	Flag	LV1ID [4:0]	BCID [9:0]
Data Record (DR)	Column [6:0]	Row [8:0]	ToT_1 [3:0]	TOT_2 [3:0]	
Address Record (AR)	11101	010	Type	Address [14:0]	
Value Record (VR)	11101	100	Value [15:0]		
Service Record (SR)	11101	111	Code [5:0]	Number [9:0]	
Empty Record (ER)	abcdefgh	abcdefgh	abcdefgh		

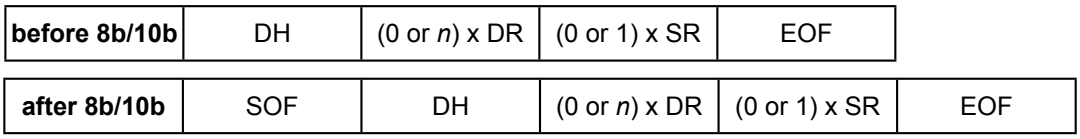


Figure 2.7: Valid pixel data sequence.

2.4 Data acquisition

A typical data acquisition process first starts with configuration of FE-I4 and all the pixels to put them in the Acquisition Mode or the Run Mode. A large enough signal pulse that is generated either from a source such as charged particle or artificial pulse generator then produces a hit in the pixel. Finally a trigger signal or readout flag is sent to each pixel to request for the data that has been stored in the pixel. In this section we will briefly discuss about the triggering mechanism and the signal pulse generation in FE-I4.

Triggering

Whenever a large enough signal passes the discriminator threshold and the digital discriminator threshold, the signal is qualified to be stored in the pixel local buffers awaiting for readout. At the same time, the leading edge of this signal will cause a latency counter within the PDR to start counting down with the system clock. One clock lasts for 25ns (40 MHz). The latency counter is a 8-bit counter and the count down starts at 255. When the counter reaches a pre-defined trigger latency value,

the readout processor checks for a trigger signal. If a trigger signal is detected at the same time, the hit data is then qualified for reading out, otherwise it will be deleted and the latency counter will be reset.

Calibration pulses

As mentioned in Section 2.2.3, FE-I4 has two capacitors in each pixel analog circuit that are capable of injecting negative charges to the preamplifier. In addition, FE-I4 is also capable of sending digital pulses downstream of the discriminator directly into the Pixel Digital Region to simulate hits. From here on, the former will be referred to as "Analog injection" while the latter as "Digital Injection". Both of these methods use the internal pulse generator block, hence hits can be produced at each pixel even without a sensor connected. On top of that, external pulses can also be used in replacement of the internal pulse generator to perform the Analog and Digital Injection.

The pixel column configuration procedure for Analog and Digital Injection is different. For Analog Injection, the column selection follows this formula:

$$\text{selected columns} = \begin{cases} 2n \\ 2n+1 \end{cases} \tag{2.1}$$

where n is the Double Column address ranging from 0 to 39. On the other hand, for Digital Injection, the column selection follows a slightly different formula as follows:

$$\text{selected columns} = \begin{cases} 2n+1 \\ 2n+2 \end{cases} \tag{2.2}$$

Nonetheless, the general idea is that when a CAL fast command is received by FE-I4, a start pulse will be generated by the pulse generator. Upon receiving the start pulse, corresponding digital or analog pulse will be generated. Various properties of the output pulse such as delay, pulse width, pulse height, rise time and so on can be programmed by approximately 10 global registers. The detail for pulse generator configuration is omitted here but the relation between the start input and the digital and analog output is shown in Figure 2.8.

Referring to Figure 2.6, analog calibration injection is achieved through two selectable capacitors, C_{inj1} (smaller capacitance) and C_{inj2} (larger capacitance).

A calibration voltage (VCAL) which can be controlled by the PlsrDAC global register is applied across the capacitors. The calibration voltage is switched very quickly (<10ns [31]) to ground and held at ground level for the duration of t_{w2}. Charge is injected at the falling edge of this calibration voltage output.

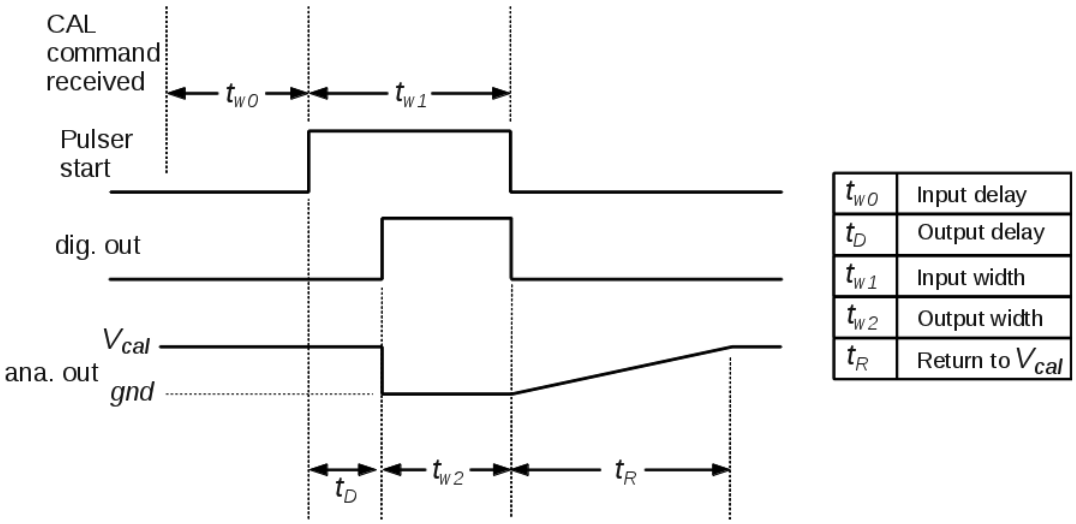


Figure 2.8: The timing diagram for the pulse generator output pulse. All the variables $t_{w0}, t_D, t_{w1}, t_{w2}$ and t_R are programmable [31].

Chapter 3

The SEABAS Readout System

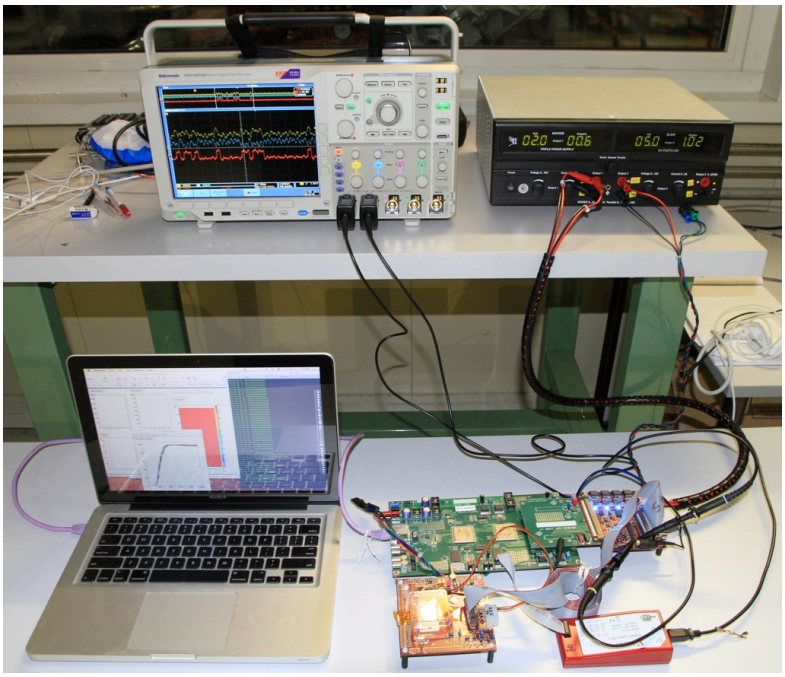


Figure 3.1: The complete setup of the SEABAS FE-I4 test system.

Figure 3.1 shows the complete setup of our SEABAS FE-I4 test system which includes a oscilloscope for signal checking and debugging, a power supply, DAQ boards and a computer containing the DAQ software. Additional power supplies may be needed depending on the FE-I4 chip powering scheme and for sensor biasing.

Each and every single pieces of the DAQ boards and the DAQ software will be given an in-depth description in the following sections.

3.1 Readout system hardware

3.1.1 System overview

Soi EvAluation BoArd with Sitcp (SEABAS) readout system aims at using a minimal set of hardware which is compact and flexible. The hardware includes a main SEBAS DAQ board, adapter cards and a Single Chip Card (SCC) which house the FE-I4 chip. The communication between the SEABAS main DAQ board and the DAQ software is achieved with an Ethernet interface. Data flow and communication relay are controlled by firmwares implemented in the Field-Programable Gate Arrays (FPGA).

For the record, two setups have been used throughout the development of this readout system. First is the single-chip setup with USBpix adapter card as a temporary interface. Towards the end of the development phase, a setup with an original daughter card specifically designed to suit our SEABAS readout system is used. Both of them will be described here.

3.1.2 Single FE-I4 setup with USBpix

Figure 3.2 shows the single FE-I4 setup. In this setup a SEABAS-USBpix daughter card is used to route the signals essential for operating FE-I4. In particular, the signal lines include the input command and input reference clock to FE-I4, the FE-I4 output data, power supply channel on/off signal and various others.

We have borrowed the USBpix FE-I4 adapter card made by the SiLab of University of Bonn [24] as a temporary solution while waiting for a dedicated adapter card to be made to suit our system. The USBpix FE-I4 adapter card is connected to SEABAS-USBpix daughter card via a KEL 100-pin connector. All the signals sent from SEABAS to the adapter card are single ended. Remember that FE-I4 works with LVDS signal, the adapter card therefore will convert the single ended signal to LVDS signal before sending it to FE-I4 and vise versa. In addition to the LVDS transceiver, the adapter card also hosts the bias voltage regulators for FE-I4. One thing to take note is that when using this setup, we did not manage to configure the power supply controller onboard the adapter card via the Inter-Integrated Circuit (I^2C). Because of this, the FE-I4 analog and digital voltage supplies (VDDA and VDDD) were set at the default DAC value of $\sim$1.5V. Even though FE-I4 specifies a 1.2V VDDD, the chip can still be operated at VDDD of

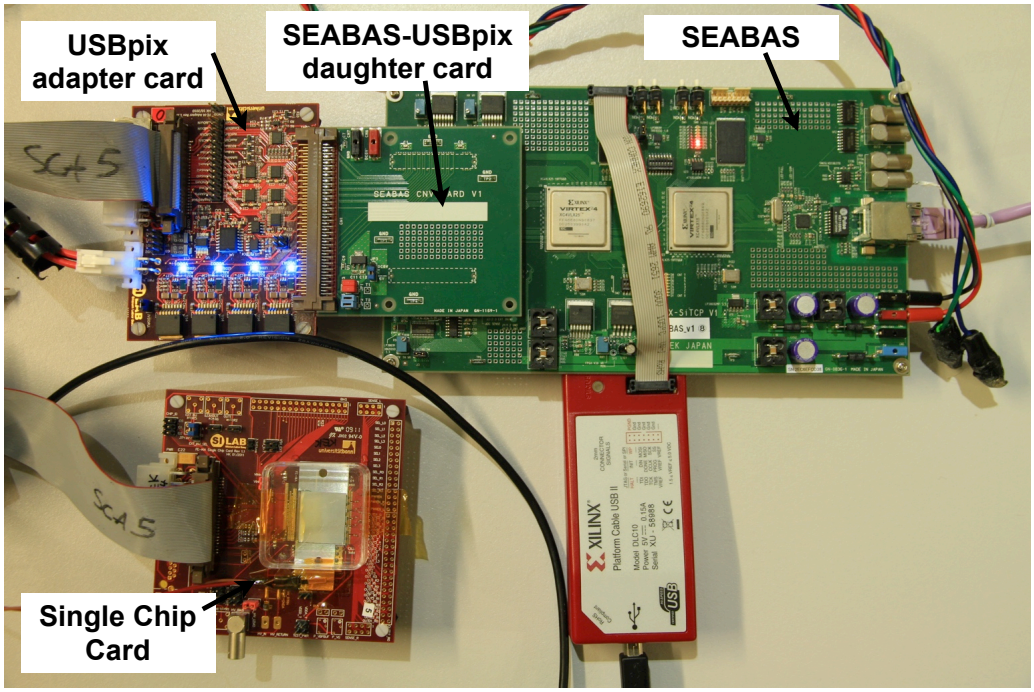


Figure 3.2: The setup using USBpix adapter card.

1.5V. As this is a temporary setup, it is sufficient for DAQ software and firmware development and testing phase.

The Single Chip card is where the FE-I4 is mounted on and wire bonded to the bond pads on the card. The Single Chip Card is connected to the adapter card via the RJ45 connector (via Ethernet cable) or the KEL 50-pin connector (via a flat ribbon cable). When the RJ45 connector is used, power lines must be routed through a dedicated power connector. Conversely, all data lines and power can be routed via the flat ribbon cable alone.

3.1.3 Single FE-I4 setup using 4-chip daughter board

Figure 3.3 shows the single FE-I4 setup using a 4-chip daughter board. The 4-chip daughter board is designed and built by Dr. Y. Ikegami and Dr. Y. Takubo from KEK. This daughter board is meant for the final implementation of the FE-I4 4-chip readout system, therefore there are a total of four in/out channels with RJ45 connectors. Each channel routes three critical signals between SEABAS and FE-I4. Those signals are the 40MHz reference clock, command signal to FE-I4, and the data output from FE-I4.

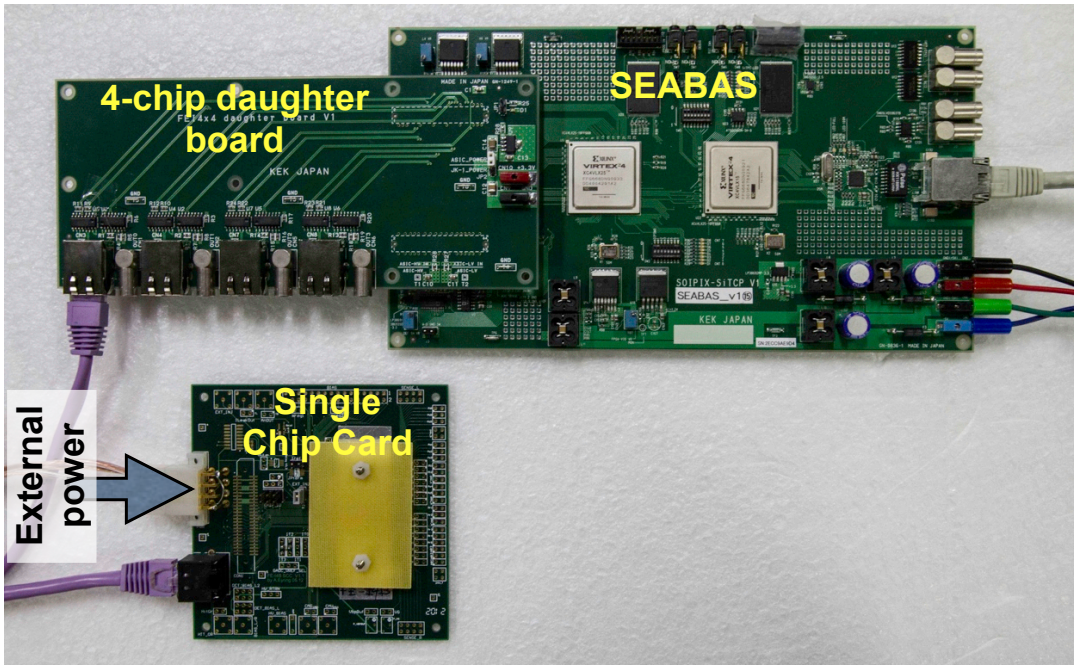


Figure 3.3: The single FE-I4 setup using 4-chip daughter board

The design of this daughter board utilizes eight DS90CP22 2x2 cross point switch [23] as the LVDS buffers for low power and high speed operation. By using LVDS signal for both the input and output data lines, it enables point-to-point high speed interconnection plus low noise and pulse width distortion. The DS90CP22 switches are set to repeater mode which operates as a 2 channel LVDS buffer. In repeater mode, the LVDS signal amplitude is restored before being forwarded to FE-I4.

Unlike the USBpix adapter card, there is no on board power supply unit to regulate the voltage to FE-I4. Hence, an external low voltage power supply is needed to power up FE-I4. In this case, the VDDA and VDDD can be controlled by adjusting the output of the low voltage power supply.

3.1.4 SEABAS Board

Figure 3.4 shows the SEABAS DAQ board. SEABAS is a general purpose DAQ board developed by the Silicon On Insulator (SOI) collaboration at KEK [26]. Its main features include:

- one user FPGA
- one SiTCP FPGA

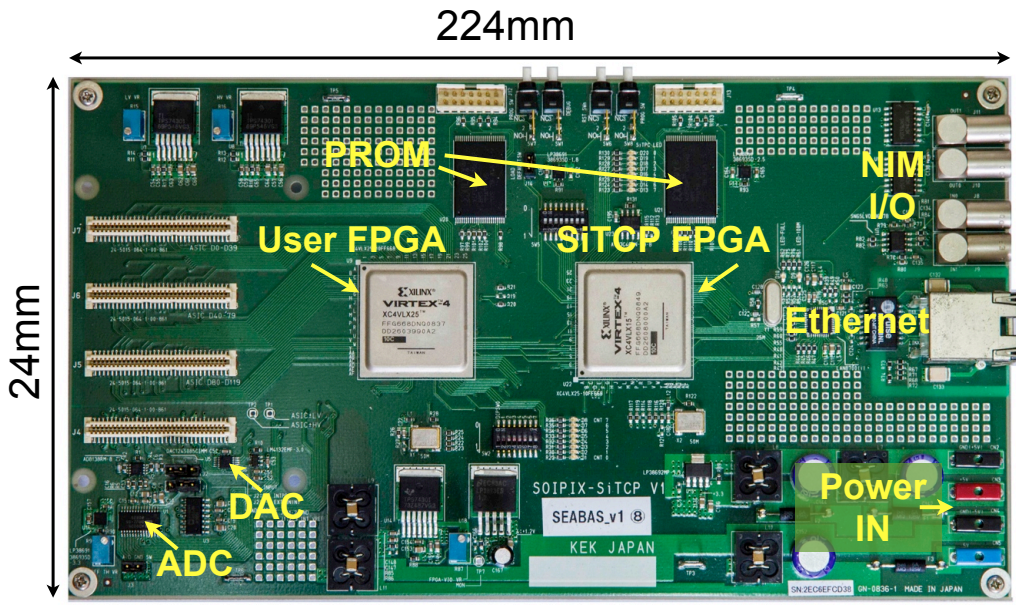


Figure 3.4: The SEABAS general purpose DAQ board.

- two PROMs
- 100 Base_T Ethernet
- 65Mhz 12-bit ADC
- 4 channels 12-bit DAC

SEABAS user guide can be found in reference [27]. A voltage of ±5V should be supplied to power the board. Depending on the setup, the current is 1A to 1.7A. For first time user, the following network setting should be used:

- SEABAS's MAC address: 01-00-C0-A8-00-10
- SEABAS's IP address: 192.168.0.16

The two FPGAs are for the implementation of the SiTCP firmware (SiTCP FPGA) and the device specific firmware (User FPGA). The SiTCP FPGA is a Xlinx Virtex-4 XC4VLX15 device with 864Kb Block RAM capacity while the User FPGA is a Xilinx Virtex-4 XC4VLX25 device with a maximum 1296Kb Block RAM capacity.

SiTCP

Silicon Transmission Control Protocol (SiTCP) basically is a hardware based TCP/IP processor which is specifically designed to simplify the communication with front-end device [29]. On top of the TCP protocol, a control mechanism over User Datagram Protocol (UDP) is also provided. SiTCP processes TCP/IP, UDP and Ethernet/IP protocol where user needs only to establish the control protocol for their specific application. In other word, the communication and data transmission between PC and SEABAS is entirely taken care automatically by the SiTCP FPGA. This significantly simplifies the development of the readout firmware or any further upgrades. SiTCP was developed by T. Uchida of KEK. SiTCP has the following advantages:

1. Small hardware size

   The standard TCP communication protocol are very complex and large. It needs powerful CPU for high rate processing. In general, embedding a powerful CPU in a single ASIC is hard to achieve without extra devices. SiTCP only adopts the minimum set of TCP/IP protocol, hence the circuit size can be made very small and thus can be implemented in a single FPGA. This makes the system inherently more flexible than other hardware bus systems like CAMAC, VME and etc.

2. High-speed data transfer

   The SiTCP is compatible with the commercial Ethernet protocols. The SEABAS version 1 which we are using uses the 100Base-T standard while the newest version of SEABAS can provide up to 1Gbps data transfer rate. The utilization ratio, which is defined to be the bandwidth ratio of effective data to the line bandwidth, of SiTCP has been shown to reach up to 95% [28, 29].

3. Simple user interface to external circuit

   The SiTCP user interface is designed to behave like a synchronous FIFO memory device with minimal interfaces to an external circuit. Once the connection between SiTCP and its communication partners (i.e. the User FPGA and PC) is established, user needs only to concern with providing proper read and write flags for the data transferring to work.

3.2 Firmware implementation

As mentioned above, SEABAS holds two FPGAs. One of the FPGAs has SiTCP firmware pre-installed while the other is for the implementation of user specific firmware, which is what we are developing.

The firmware is written in Verilog, a hardware description language used to model analog and digital circuits at the register-transfer level of abstraction. The Verilog's syntax is similar to the C programming language. In general, the user firmware takes care of the following tasks:

- forwarding command bit stream from the control software to FE-I4.
- generation of Fast commands (LV1, CAL etc.) that need precise timing.
- receiving FE-I4 data output and data parsing.

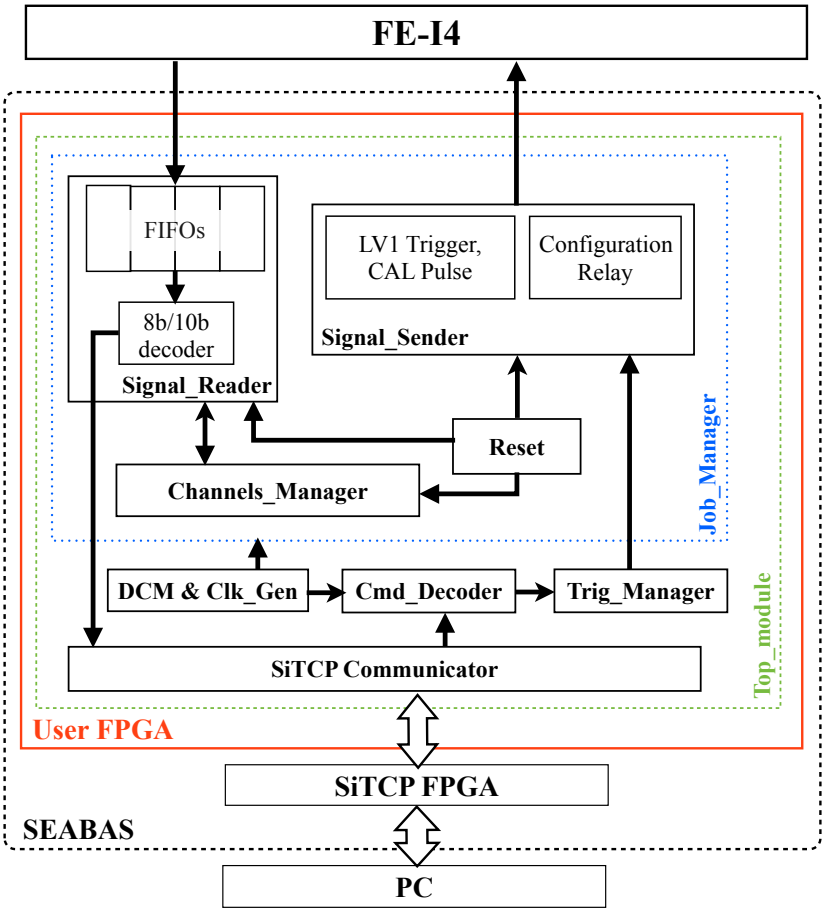


Figure 3.5: The SEABAS firmware design.

The firmware is designed such that it consists of the modules that encapsulate design hierarchy. Each module handles a specific set of tasks and communicates with other modules through a set of declared input, output, and bidirectional ports. An overview of the SEABAS firmware modular structure is given in Figure 3.5. The functionalities of each module are described as follows:

1. **Top_module**
   All I/O ports are declared in this module and are constrained in a separate User Constrained File (UCF). The top module contains the Digital Clock Manager (DCM) and Clock Generator (Clk_Gen) block, SiTCP Communicator block, Command Decoder (CMD_Decoder) block and Trigger Manager (Trig_Manager) block. The function of each block is as follows:

   - DCM and Clk_Gen
     This component is used to derive and control the various clocks needed within the system. It contains a delay-locked loop (DLL) to completely eliminate clock distribution delays. The Digital Clock Manager is also used to synthesize clocks with different frequencies and phase shift. The Clk_gen block is coupled to the Digital Clock Manager to produce additional clock frequencies. The clocks generated by these two blocks include:

     (a) 25MHz clock
     This is the primary clock used to drive most of the Finite State Machines (FSM) in the Top_module. It also serves as a driver clock for the SiTCP Communicator as well as the readout FIFOs in the Signal_Reader module.

     (b) 160MHz clock and 160MHz clock with the phase shifted by 135° and 180°
     These clocks are selectable and are fed directly into the Signal_Reader Block to drive the receiver FSM. In addition, the 160MHz clock is used to drive the Channels_Manager block and used as an input for the Clk_Gen block to generate 20MHz and 40MHz clocks.

     (c) 20MHz clock
     The sole purpose of this clock is to function as the write clock for the asynchronous FIFOs in Signal_Reader block.

(d) 40MHz clock

This is the reference clock that is used to drive the FE-I4 logics. As 40MHz is the reference frequency for FE-I4 input, it also means that all the FSMs in Job_Manager that deal with the inputs to FE-I4 is driven by this clock.

In addition, power-up reset is implemented in the firmware based on the Digital Clock Manager's LOCKED signal. When this signal is high it means the clock outputs already established the correct frequency and phase.

- SiTCP Communicator

  As its name implies, this block defines and constrains the I/O ports that link to SiTCP FPGA. It contains the TCP/IP as well as the UDP interfaces. All the data transfer are entirely taken care by this block automatically.

- CMD_Decoder

  This block identifies header from the data stream that is sent from the control software and decide what to do next. The headers are associated with a set of tasks, for example, to activate other modules or to assign values to registers from the payload extracted from the data stream.

- Trig_Manager

  This block consists of three FSMs that set the triggers for Signal_Sender Module. It tells the Signal_Sender module when it can start sending out the command bit stream to FE-I4. In particular, it manages the interval between each successive CAL and LV1 command and stops sending the commands when the desired number of LV1 triggers is reached.

2. **Job_Manager**

   This module consists of the Signal_Sender and the Signal_Reader modules as well as the Channel Manager and Reset blocks. In the Reset block, one master counter issues the internal reset signal that is used by all other FSMs contained in the Job_Manager module. On the other hand, the Channel Manager simply flags the FIFOs for readout whenever it is ready.

3. **Signal_Sender**

   Through this block the command or configuration bit stream is routed to FE-

I4. As the CAL and LV1 command need precise timing, they are the only commands that are directly managed and sent from the firmware without intervention of the control software.

4. **Signal_Reader**
   As described in Section 2.3.4, a valid data record is framed by the SOF, DH and EOF frames. Since FE-I4 is always sending out IDLE states when there is no hit data, a FSM is used to check the data stream for the SOF and EOF frames. When a SOF frame is recognized, it starts to extract the correct portion of data and pass it to the 8b/10b decoder for decoding. The decoded data is then stored in an asynchronous FIFO which is embedded in the Signal_Reader module. Once the FSM finds an EOF frame, it stops taking out data. At the same time the SiTCP Communicator starts to read out the stored data from the FIFO.

3.3 Software framework

The software is written in C++ to exploit the features that object-oriented programming style offers. The concept is to have a program that composed of self-sufficient reusable functional modules ("classes") with a collection of interacting objects that contain all the information needed to manipulate its own data structure. We try to limit the interdependencies between each module to reduce system complexity and thus increases its robustness and maintainability.

Figure 3.6 illustrates the modular structure of the DAQ and the control software. The control software has two main classes which other classes are built upon, that is the Configuration Class (CONFIG) and the Data Acquisition (DAQ) Class. The Operation (OPR) and Injection (INJ) sub-classes are derived from CONFIG and DAQ top level classes respectively. In addition, there are two stand alone classes that are independent from the other, namely the SiTCP Controller and Data Output (DOUT) classes. Each instance (object) of classes is capable of receiving messages, processing data, and sending messages to other objects. The functions of the main classes will be explained next.

1. **SiTCP Controller**
   At the control software end there is also a SiTCP controller which talks to its counterpart in the firmware. It is primarily responsible for establishing

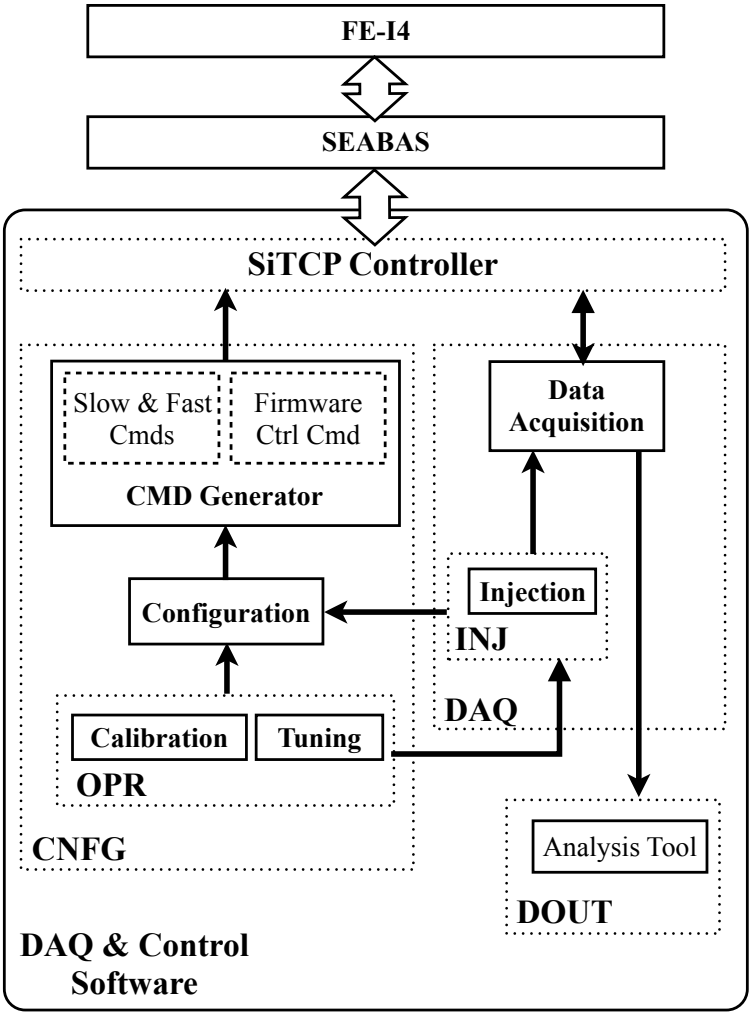


Figure 3.6: The modular framework of SEABAS readout system software.

connection to SEABAS DAQ board and providing UDP control. TCP/IP data transfer is established entirely by using the standard socket functions that are readily available in network programing language.

2. **Configuration class**

   In this class there is a Configuration and a Command Generator block. The Configuration block essentially takes care of all the routines that provide configuration of all the registers in FE-I4 including both the global registers and the local pixel registers. It provides the algorithms to select which pixel to activate or mask based on the user selection. Two simple text files, one contains all the register values and the other contains all the parameters essential for the execution of the program are read at the very beginning of

the program execution. The stored FE-I4 register values are then passed to the Command Generator block which performs data reformatting according to the description in Section 2.3.3. Additionally, an extra header and trailer are inserted before sending it to the firmware. The reformatted data stream has a simple structure as shown in Figure 3.7. The purpose of adding a trailer is to separate two configuration bit stream. On the other hand, each unique 8-bit-long header corresponds to a certain task to be carried out by the firmware as defined in Table 3.1.

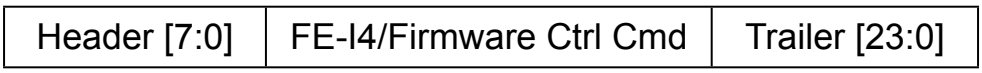


Figure 3.7: Reformatted configuration data stream. The payload can be either the configuration command for FE-I4 or the control command for the firmware.

Table 3.1: Header to be read by the Command Decoder block in firmware.

Header	**Description**
1110 1001	write global registers.
1110 1101	write pixel 672-bit shift register.
1110 1010	start signal for issuing CAL or LV1 command.
1110 1011	start trigger for various blocks in Job_Manager module.
1110 1111	assign payload bits to registers in firmware.

3. **DAQ class**

   This is another class that contains an Injection module, a DAQ block and a Decoder. The primary function of the Data Acquisition block is to monitor the kernel buffer for any data packet that has been transferred from the SEABAS. Any data stored in the kernel buffer is retrieved and temporarily stored in the user buffer. The Data Acquisition block then unpacks the data into a 24-bit data record. From this 24-bit data record, relevant information is extracted based on the format described in Section 2.3.4.

3.4 DAQ Operation Overview

Having discussed about the FE-I4 chip, the readout hardware and the implementation details of the firmware and software, it is by all means appropriate to go through the whole process of acquiring data from the FE-I4 chip. The overall DAQ flow using FE-I4 internal calibration pulse generator is illustrated in Figure 3.8.

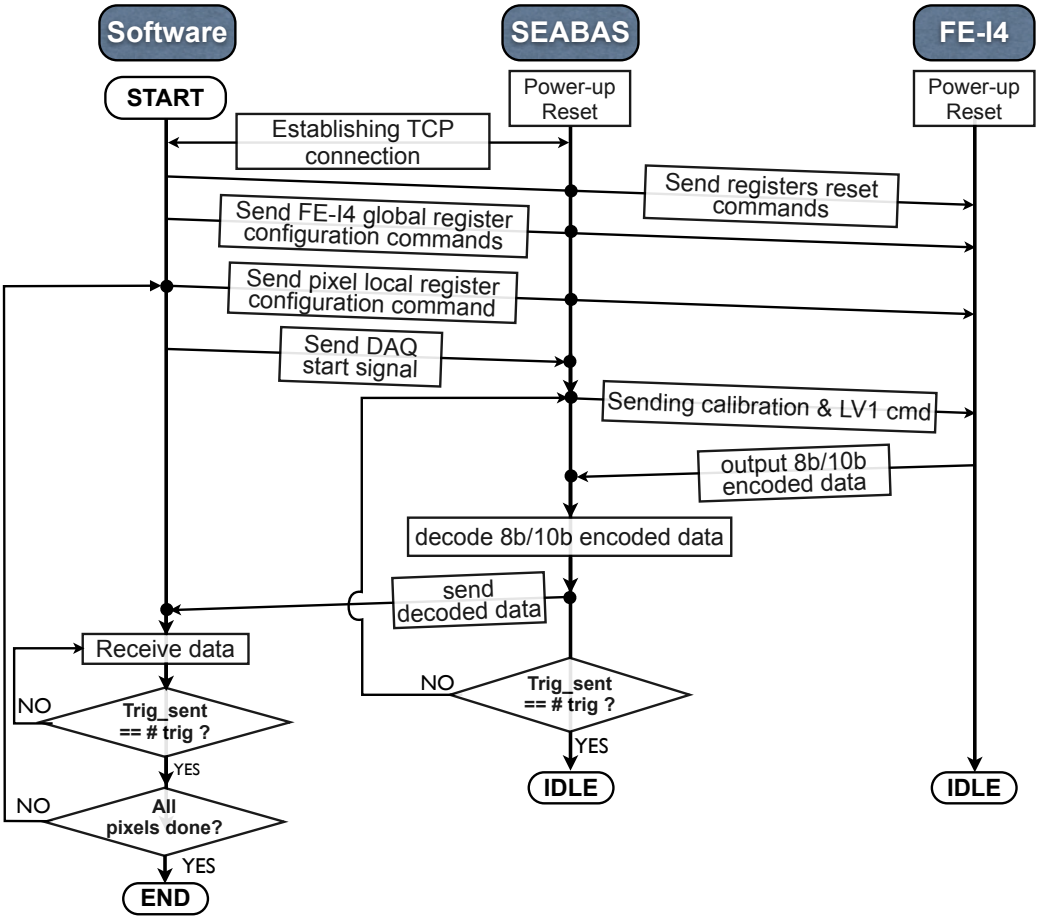


Figure 3.8: The overall DAQ flow chart.

An succinct overview of the DAQ process flow is as follows:

1. Establish connection between PC and the SEABAS.
2. Reset the user firmware and FE-I4.
3. Send configuration commands to configure FE-I4's global registers.
4. Send configuration commands to configure selected pixels' local registers.

5. Signal the user firmware to start the DAQ process.
6. Send calibration pulse command and Level 1 trigger command from firmware to FE-I4.
7. FE-I4 output hit data in 8b/10b encoded format to SEABAS.
8. User firmware decodes the 8b/10b encoded data and send it to PC.
9. DAQ software receives and extracts relevant information from the received data stream.
10. Repeat the whole process on other pixels.

Chapter 4

DAQ and Functionality Test

At this stage, reader should have a general idea about how FE-I4 and each component of the readout system function as well as the overall process of data taking. In the first half of this chapter we will examine the results of various tests to establish the basic DAQ operation. One of the main objectives of this research is to provide the functionalities to test the FE-I4 electronics. In the second half of this chapter, numerous other functionalities which are essential for the role of our readout system as an electronic test stand for FE-I4 will also be discussed. A few terms that will be used frequently are defined in advance. They are as follows:

1. **Scan**
   A series of loops in which a particular register value is reconfigured before the start of each loop.

2. **Time Over Threshold (TOT)**
   A measurement of the time a certain signal is above a threshold. The unit used in FE-I4 case is 25ns or 1 bunch crossing (BC).

3. **Mask stage**
   To avoid the overflow of User FPGA's buffer, typically only a fraction of the pixels is selected for injection test. The injection procedures is repeated to cover all the pixels on FE-I4. Each of this repetitions is referred to as a mask stage.

4.1 Basic DAQ operation test

To confirm that our readout system is working as designed, a few basic DAQ operation tests have been performed. Through each test, the following points will be proved:

- communication with FE-I4 is successfully established.
- configuration of the specific global registers and the local pixel registers can be performed correctly.
- calibration command and Level 1 trigger command can be sent to FE-I4.
- both analog and digital injection can be performed.
- decoding and data extraction is correctly carried out.

4.1.1 Communication between SEABAS and PC

Before proceeding with any further tests, it is essential to verify that signal transmission from computer to FE-I4 has been established. In order to see if the LVDS configuration command signal and clock signal are routed to FE-I4 correctly, we probed the corresponding output ports either on the USBpix adapter card or the 4-chip daughter board with oscilloscope.

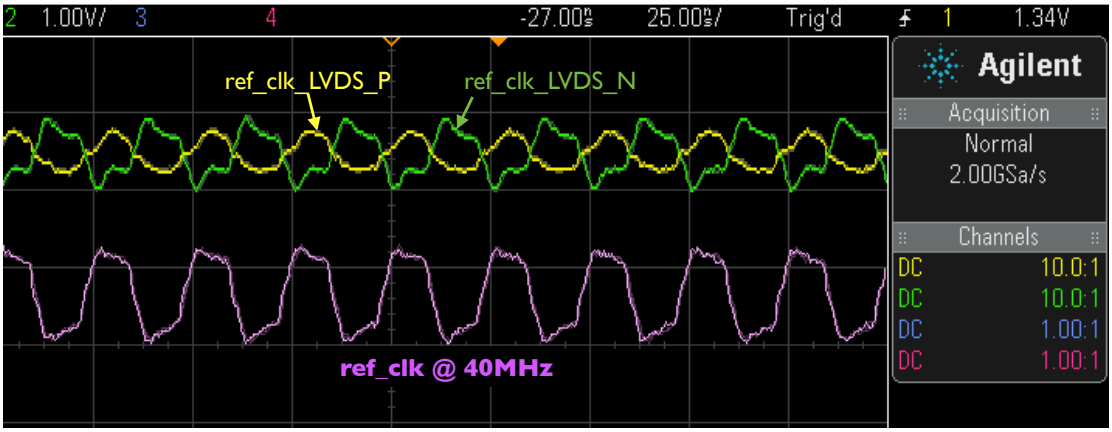


Figure 4.1: The 40MHz reference clock that is fed into the FE-I4 clock generator block.

The 40MHz reference clock sent from SEABAS to FE-I4 was verified as shown in Figure 4.1. The horizontal time scale is 25ns per unit grid. The command signals' bit pattern was also verified to be exactly what had been sent from the computer. One example of slow command bit pattern is shown in Figure 4.2. The correct value for each field formatted according to item 2 (WrRegister) in Table 2.3 can be clearly seen. In both Figure 4.1 and Figure 4.2, the yellow signal line is the LVDS positive signal, the green line is the LVDS negative signal and the purple line is the difference between the two.

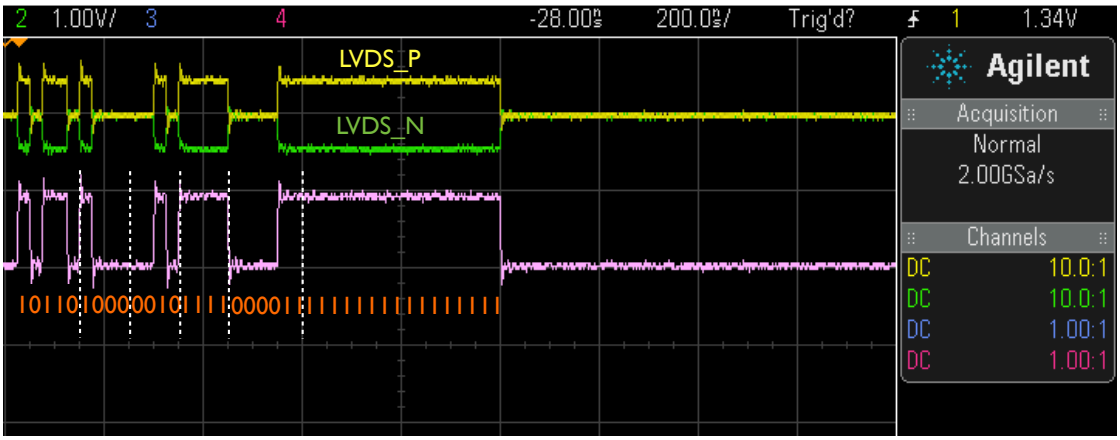


Figure 4.2: Slow command (WrRegister) bit pattern that was sent to FE-I4. In this case, the bit pattern was 10110 _ 1000 _ 0010 _ 1111 _ 000011 _ 1111111111111111.

4.1.2 Digital Injection Test

The main purpose of the Digital Injection Test is to examine the functionality of the pixel digital region circuit as well as the readout chain. Nonetheless, it can also serve as a reliable mean to test the five points mentioned in Section 4.1.

In the Digital Injection Test, strobe signals with a fixed length are sent downstream of the pixel discriminator directly into the pixel digital region circuit to simulate a hit in the pixel. The number of hits should be the same as the number of strobe signals that have been sent. In addition, the Time Over Threshold value for every hit should be the same as the strobe signal length.

All pixels on FE-I4B were used in this digital test. In total, 100 strobe signals with length lasting for 13 bunch crossing were sent. If the hit map shows the correct hit pattern with the correct Time Over Threshold value, then we can be convinced that the decoder and data extraction is implemented correctly.

Figure 4.3 (a) shows the resultant hit map. For this particular FE-I4, all but two pixels returned 100 hits. These two pixels which do not register any hit are known to be as dead pixels, and hence masked. Figure 4.3 (b) shows that the correct Time Over Threshold value, which is 13, was returned from all the hits. We confirmed by these results that the digital injection, the decoding, and data extraction were correctly carried out.

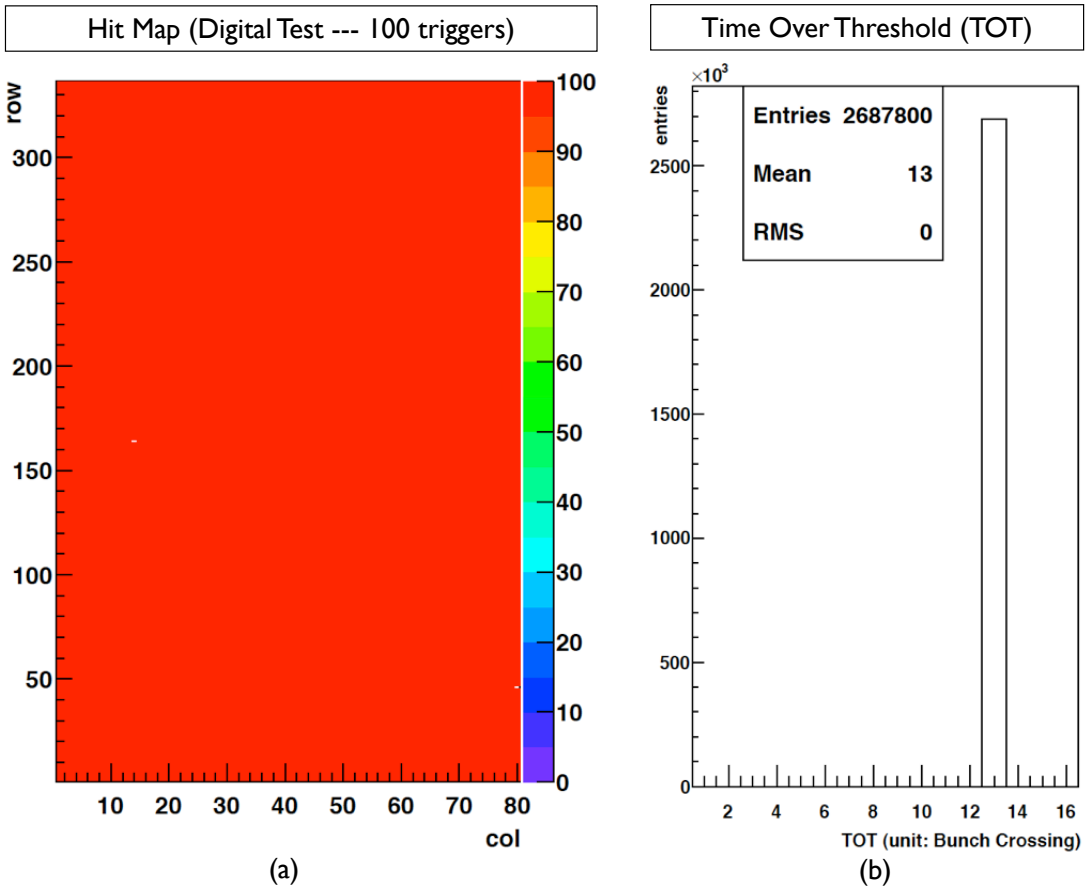


Figure 4.3: (a) The hit map for Digital Injection Test. The horizontal axis corresponds to the column number while the vertical axis is the row number. The color scale gives the total number of hits. (b) Time Over Threshold value for all the digital injection hits.

Apart from this, by setting the column mask in the global register or the row mask in the local pixel register, we can test whether these registers can be configured properly or not. With this test we can also check the ability of selecting any combination of pixel for testing. In Figure 4.4 we can see the effect of masking

column 33 to column 64 and activating the row mask in the local pixel register for all pixels other than every third row of pixels. We confirmed that the configuration of the global register and the local pixel registers can be performed correctly. Likewise, the fact that we obtained the correct hit pattern means the decoder has decoded the data accurately.

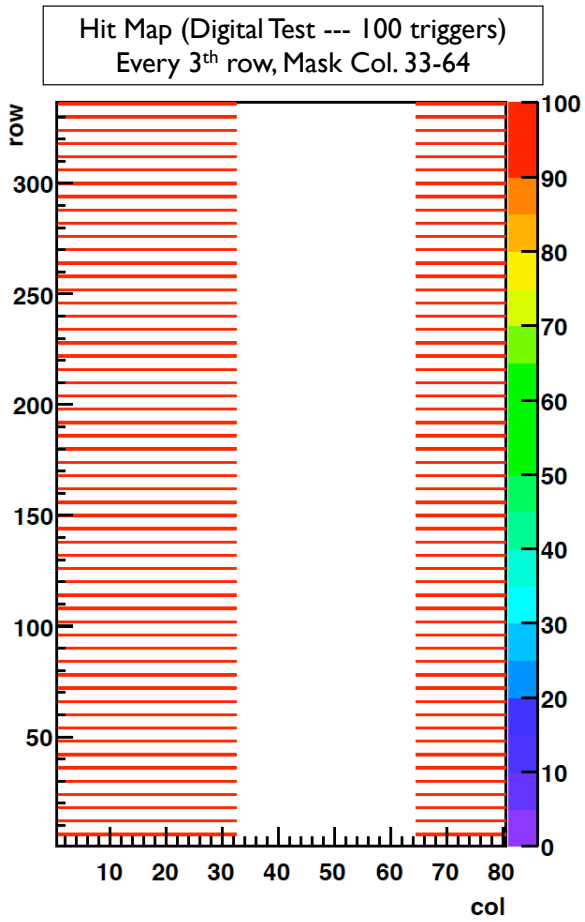


Figure 4.4: The hit map obtained by masking column 33 to column 64 and every third row of pixels.

4.1.3 Analog Injection Test

The Analog Injection Test is meant for testing the pixel analog circuit components such as the charge injection capacitors, pre-amplifier, amplifier and the discriminator. From the hit map, pixels that have defect can be isolated and masked. For the Analog Injection Test to work, the pixel's charge injection capacitors need to

be activated by configuring the corresponding local pixel registers. Furthermore, the global threshold and the calibration voltage also need to be set by configuring the global registers called Vthin_Alt_Fine and PlsrDAC, respectively. Thus, analog injection can serve as a good indicator whether these local pixel registers and the global registers can be configured properly.

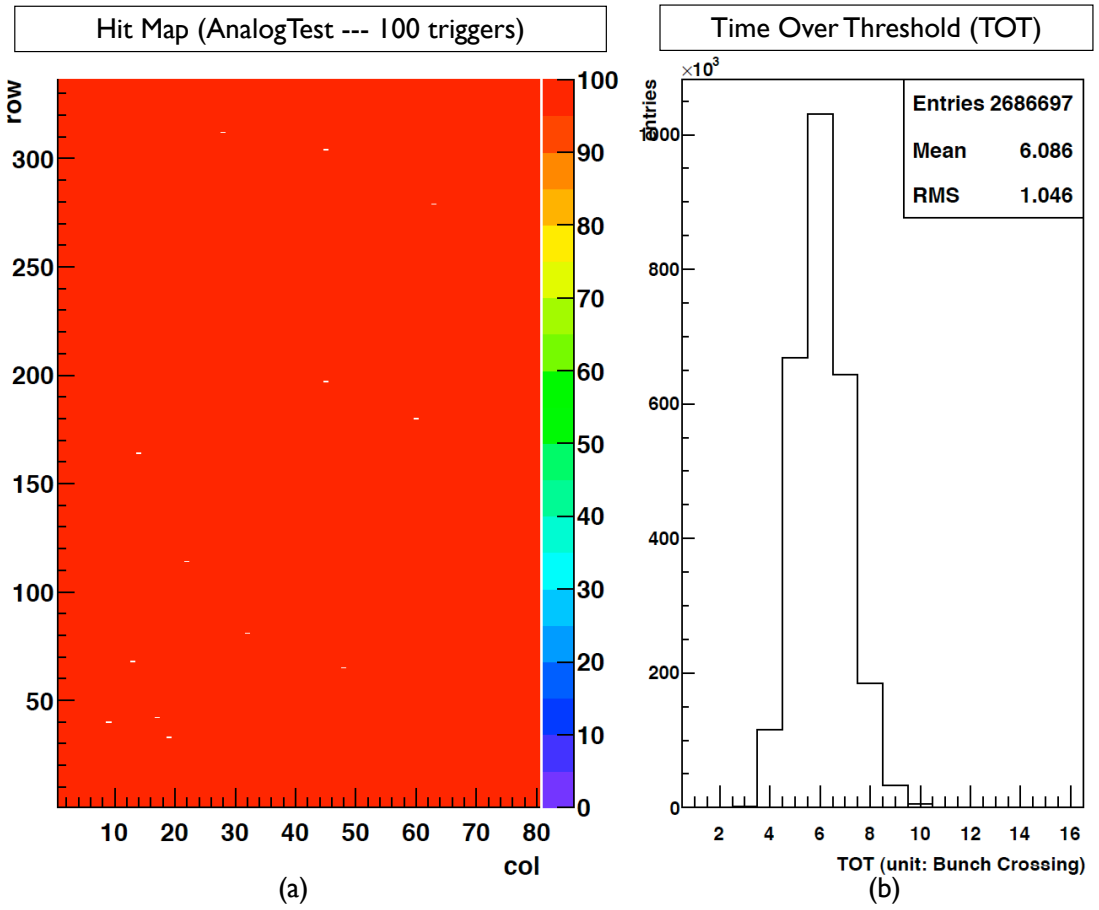


Figure 4.5: (a) The hit map for Analog Injection Test on all pixel on FE-I4B. (b) The measured Time Over Threshold value by an Analog Injection Test.

To perform an Analog Injection Test, charge is injected to the pixel analog circuit's pre-amplifier through the charge injection capacitors. To make sure every pixel has its analog discriminator fires, a large input charge well above the threshold is injected. This can be ensured by setting the global register, PlsrDAC, to an appropriate value. In this test, 100 triggers were sent to FE-I4. We then expect the number of hits to be the same as the number of triggers.

Figure 4.5 (a) shows the hit map of the Analog Injection Test for all pixels of

FE-I4B by using PlsrDAC value of 300. Most of the pixels returned a full 100 hits. A maximum hit value of 100 means the decoder and data extraction did not mess up in obtaining hit information. If that is the case, some pixels might register hit number that is more or less than the number of triggers that had been sent. On the contrary, those pixels without a hit might be due to the fact that their threshold value is way too high or their charge injection capacitors have broken. Figure 4.5 (b) shows the measured Time Over Threshold distribution.

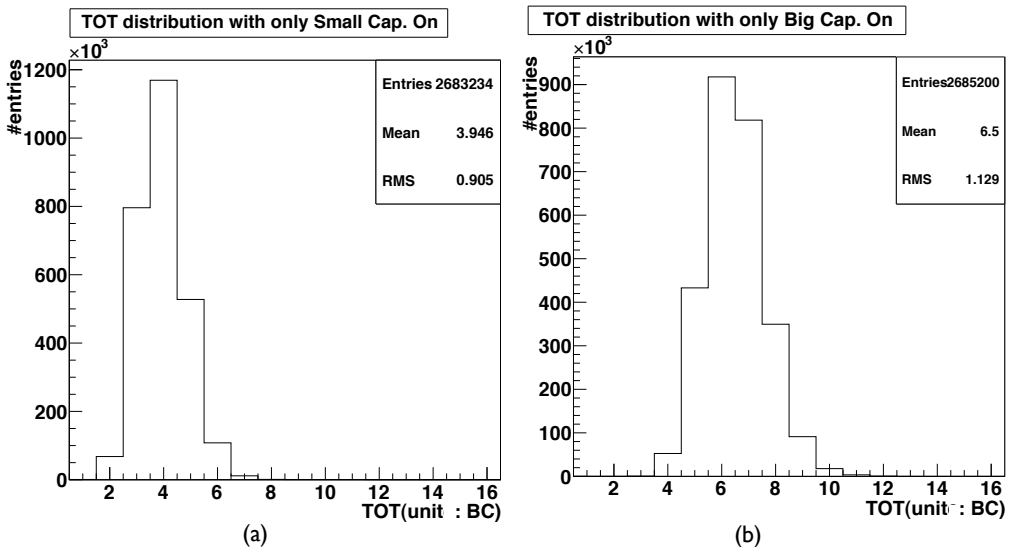


Figure 4.6: Time Over Threshold distribution in Analog Injection Test with the same setting other than charge injection capacitor setting: (a) small capacitor ON, (b) big capacitor ON.

Recall that in analog injection charge is injected by the two charge injection capacitors, one has larger capacitance than the other. We can turn either one off and run the Analog Injection Test with the same setting. The average Time Over Threshold value should be larger when the high capacitance capacitor is turned on and vice versa. By performing this test, we can verify that the specific local pixel registers can be configured correctly.

Figure 4.6 shows the observed change in Time Over Threshold value. As expected, the average Time Over Threshold value was larger for the case where the large capacitor was selected.

4.1.4 Latency Scan

As a reminder, the leading edge of a valid hit signal in each pixel starts a 8-bit latency counter. The latency counter counts down from 255 with the system clock until it reaches the trigger latency value programmed in the FE-I4 global register. If the LV1 trigger does not reach the pixel when the latency counter is equal to the programmed trigger latency value, the hit is not read out and is discarded. This process is illustrated in Figure 4.7. For the instance when Digital or Analog Injection Test failed completely (blank occupancy plot), one possibility might be that the trigger latency setting is wrong. This is when the Latency Scan comes in handy by telling us the correct trigger latency to use.

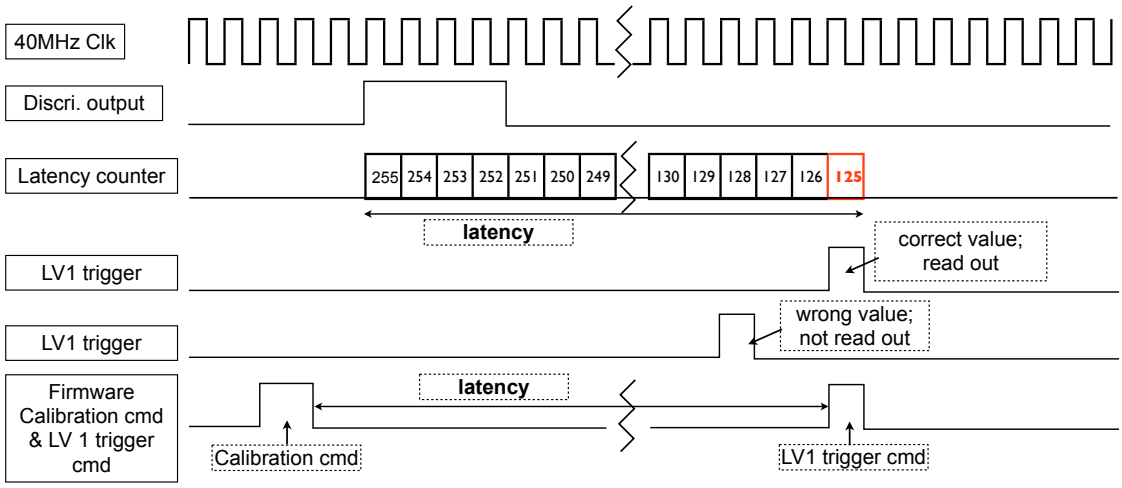


Figure 4.7: The total hits (ordinate) versus trigger latency (abscissa) histogram. The peak gives the correct trigger latency value that should be used.

The precise timing when a LV1 trigger should arrived is controlled in the firmware. As shown in Figure 4.7, if the true latency is 125, this latency value must also be correctly set in the firmware. Otherwise, no hit is ever registered in any pixel. To check whether this is true, we perform the so called Latency Scan.

In the Latency Scan the same number of triggers are sent to the selected pixels at each 255 possible trigger latency (global register, Trig_lat) values in either the digital or the analog injection. If the latency value is set correctly in the firmware, the histogram (when the trigger multiplier register, Trig_cnt is set to 1) is expected to have a single peak at the programmed trigger latency value. This is confirmed as shown in Figure 4.8. In this case, the latency value is set to 125.

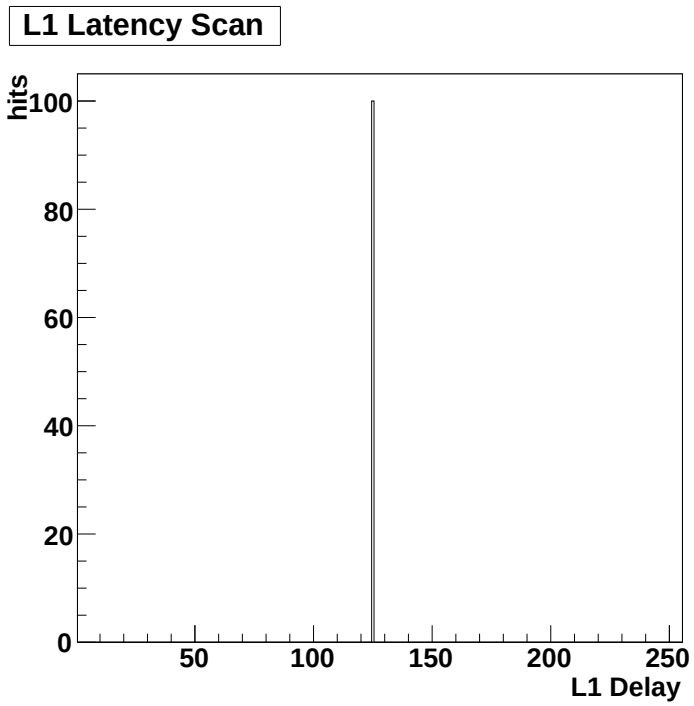


Figure 4.8: The number of hits (ordinate) versus trigger latency (abscissa). The peak gives the correct trigger latency value that was used.

Note that the trigger latency register is 8-bit compliment thus the trigger latency is not the actual LV1 latency value. Instead, for the real LV1 latency of 125 clocks, for example, the trigger latency register value should be set to 255-125 = 130.

4.2 Functionality test

We have developed several essential readout functionalities in which their operation is built upon the Analog Injection Test mentioned in the previous chapter. These functionalities play a major role as an electronic test stand for FE-I4 as well as the DAQ system for the module consisting of sensor and FE-I4. Furthermore, they also serve as further verification for the functioning of the control software and firmware. In the subsections below, the description for each functionality is explained.

The functionalities that we describe here include:

1. Injected Charge Scan
2. Threshold tuning
3. Time Over Threshold Scan
4. Time Over Threshold tuning

4.2.1 Injected Charge Scan

The main purpose of this scan is to find the threshold and measure the noise for each pixel. The FE-I4 specification states that the single pixel Equivalent Noise Charge should be lower than 300 electrons. In order to measure the threshold and noise correctly, we have to first get the correct charge conversion factor from the global register, PlsrDAC. PlsrDAC is a 10-bit global register that controls the calibration pulse voltage which in turn determines how much charge is injected into each pixel. In the Injected Charge Scan, analog injection is performed and the hit efficiency is scanned against the injected charge at a fixed discriminator threshold.

The calibration voltage, V_{cal} can be expressed in term of the PlsrDAC as follow:

$$V_{cal} = a + b(\text{PlsrDAC}) \text{ (mV)} \tag{4.1}$$

where a is the the V_{cal} offset and b is the slope of V_{cal} versus PlsrDAC plot. Likewise, the relation between the capacitance, the calibration voltage and the charge (in unit of the number of electrons) can be expressed as below:

$$N_{electron} = \frac{C_{inj}V_{cal}}{e} = \frac{C_{inj}[a + b(\text{PlsrDAC})]}{e} \tag{4.2}$$

where e is the electron charge and C_{inj} is the summed capacitance of both the injection capacitors. The typical value of C_{inj} is 5.85fF if both capacitors are used. The V_{cal} offset is small and thus is out of consideration in the context of the readout system development.

By measuring the V_{Cal} for each PlsrDAC value, we can get the conversion factor, b. The result can be seen in Figure 4.9. The value b was measured to be

around 1.4mV/PlsrDAC. By using Equation 4.2, the value in PlsrDac register can now be converted to the corresponding number of electrons.

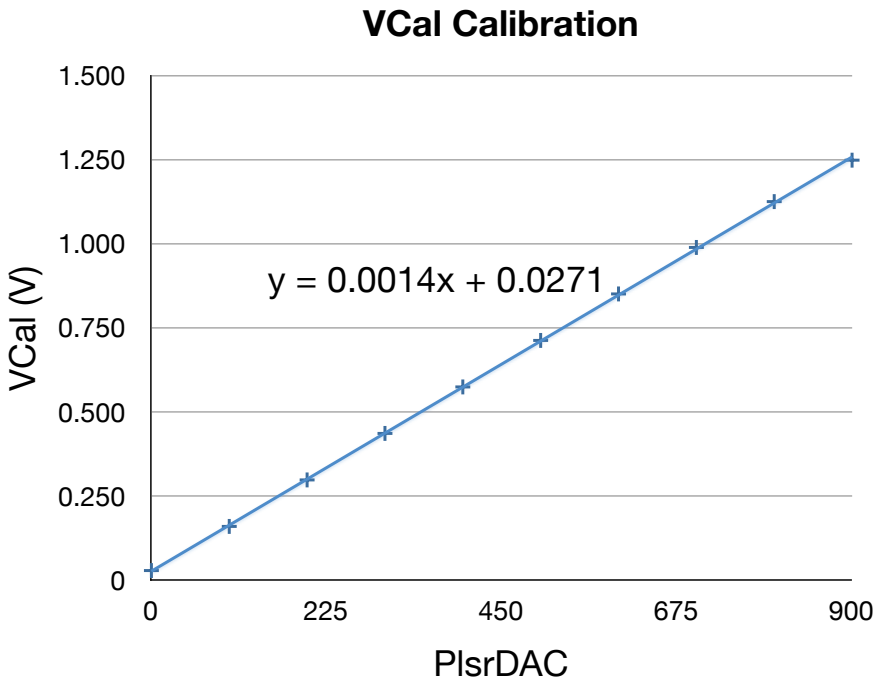


Figure 4.9: The calibration voltage (VCal) versus PlsrDAC value. The slope was measured to be approximately 1.4mV/PlsrDAC.

Due to noise contribution from the electronics as well as the sensor, the hit efficiency as a function of injected charge is smeared out into more or less a "S" shape. This smeared distribution can be modeled with the normal cumulative distribution function as in Equation 4.3:

$$\text{eff}_{hit}(Q_{inj}) = \frac{1}{2}\left(1 + \text{Erf}\left(\frac{Q_{inj} - Q_{th}}{\sqrt{2}\sigma_{noise}}\right)\right) \tag{4.3}$$

where Q_{inj} is the injected charge, Q_{th} is the threshold and σ_{noise} is the noise expressed in equivalent noise charge and is proportional to the slope of Equation 4.3 at $\text{eff}_{hit} = 0.5$. The threshold is defined as the injected charge that gives 0.5 of hit efficiency.

As illustrated in Figure 4.10, by injecting various charge, a S-curve can be mapped out. In this case, the Injected Charge Scan was carried out on a single

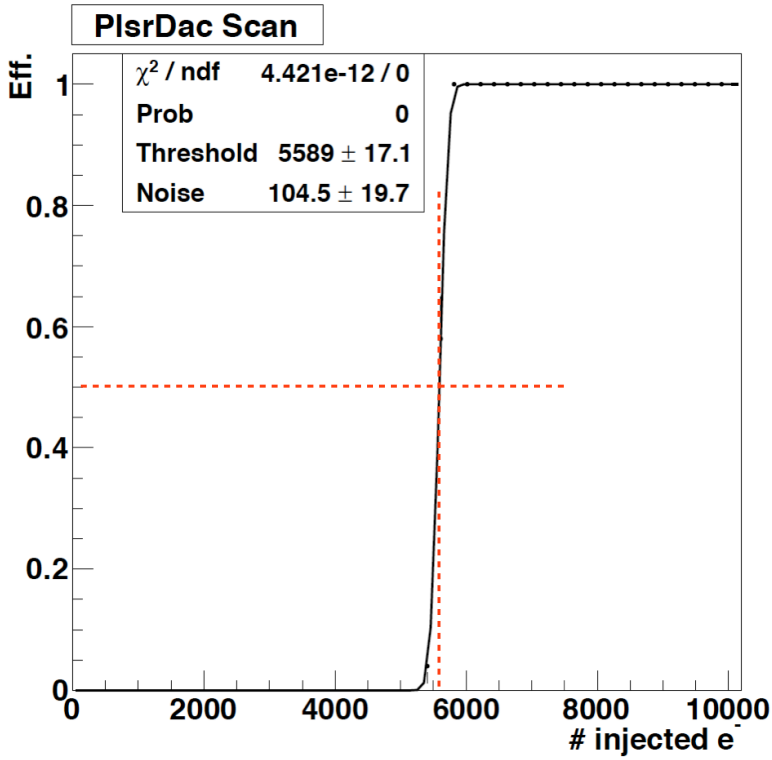


Figure 4.10: The Injected Charge Scan. The vertical axis is the efficiency (total hits/total triggers input) and the horizontal axis is the number of injected electron converted from PlsrDAC by using Equation 4.2.

pixel. Fifty events were taken at each PlsrDAC value from 0 to 200 with the step of 4. From Figure 4.10, the threshold for this particular pixel is measured to be 5589 electrons while the noise is 104 electrons. As required by the FE-I4 specification, the single pixel equivalent noise charge is less than 300 electrons [31].

By performing Injected Charge Scan for each pixel, the mean threshold and the average noise for the whole FE-I4 can be obtained. To cover all the pixels, typically 120 mask stages are required. Figure 4.11 shows the threshold distribution for a single FE-I4 while Figure 4.12 shows noise distribution for a single FE-I4.

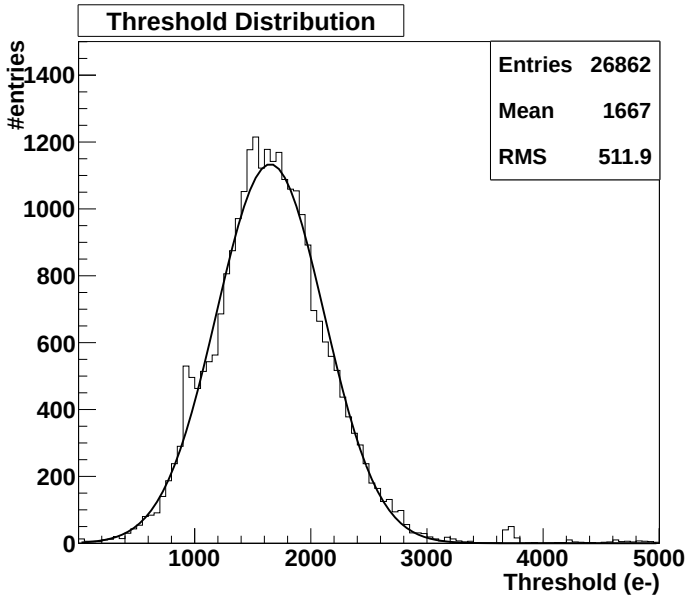


Figure 4.11: The threshold distribution of the whole FE-I4. The vertical axis is the number of pixels while the horizontal axis is the threshold value in unit of electrons. In this case the mean threshold, Q_{th_mean} is about 1667 electrons with a typical threshold dispersion of 512 electrons for an untuned FE-I4.

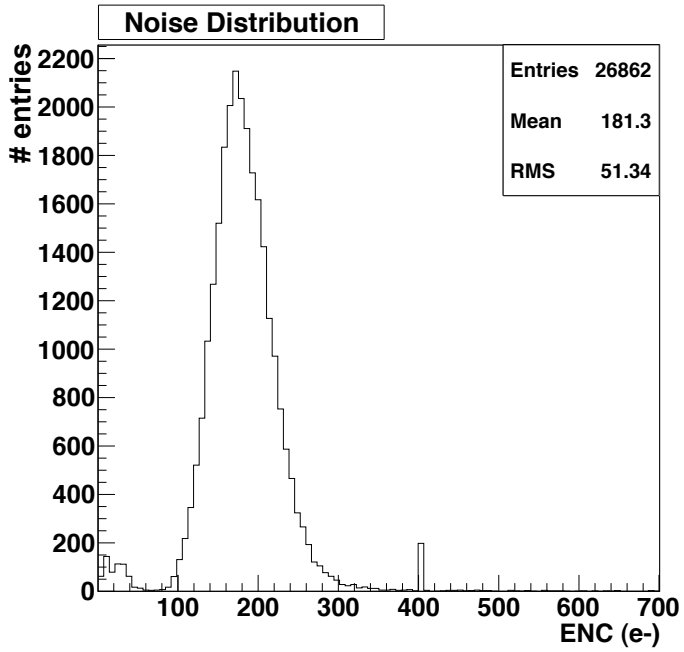


Figure 4.12: The noise distribution of the whole FE-I4. The average noise value of 181 electrons comply with the FE-I4 specification.

4.2.2 Threshold tuning

In actual particle detection experiment, the threshold of each pixel must be properly set in order to optimize the particle detection efficiency and the position resolution. Ideally, all pixels should have the same threshold. This is also true in FE-I4 testing phase where we need to study its characteristic or to study how certain electrically properties affect the threshold. Thus the purpose of threshold tuning is to bring the threshold of all pixels as close as possible to the target threshold value.

There are two threshold tuning methods. One is the coarse tuning, namely the Global Threshold Tuning and the other is the fine threshold tuning, or Local Threshold Tuning. The FE-I4 specification specifies that the spread of the tuned threshold must be less than 100 electrons. This requirement serves as a judge on the accuracy of our threshold tuning procedures.

Global Threshold Tuning

FE-I4's global threshold value which is common to all pixels is controlled by a global register called GDAC. Before performing the pixel by pixel threshold adjustment, it is vital to make the FE-I4 average threshold closer to the target threshold by first selecting a global threshold register setting. This allows finer adjustment of pixel threshold to be performed later.

The procedure for the Global Threshold Tuning is to measure the average threshold of FE-I4 for four global threshold register values. Then the average threshold versus global threshold register plot is interpolated to get the proper global threshold register that matches the target threshold value.

The result can be seen in Figure 4.13. In this test, the result was a global threshold register value of 132 at target threshold of 3000 electrons. Note that the global threshold tuning only brings the average threshold closer to the target threshold value, it does not reduce the spread in threshold. The threshold distribution before and after the Global Threshold Tuning can be seen in Figure 4.14.

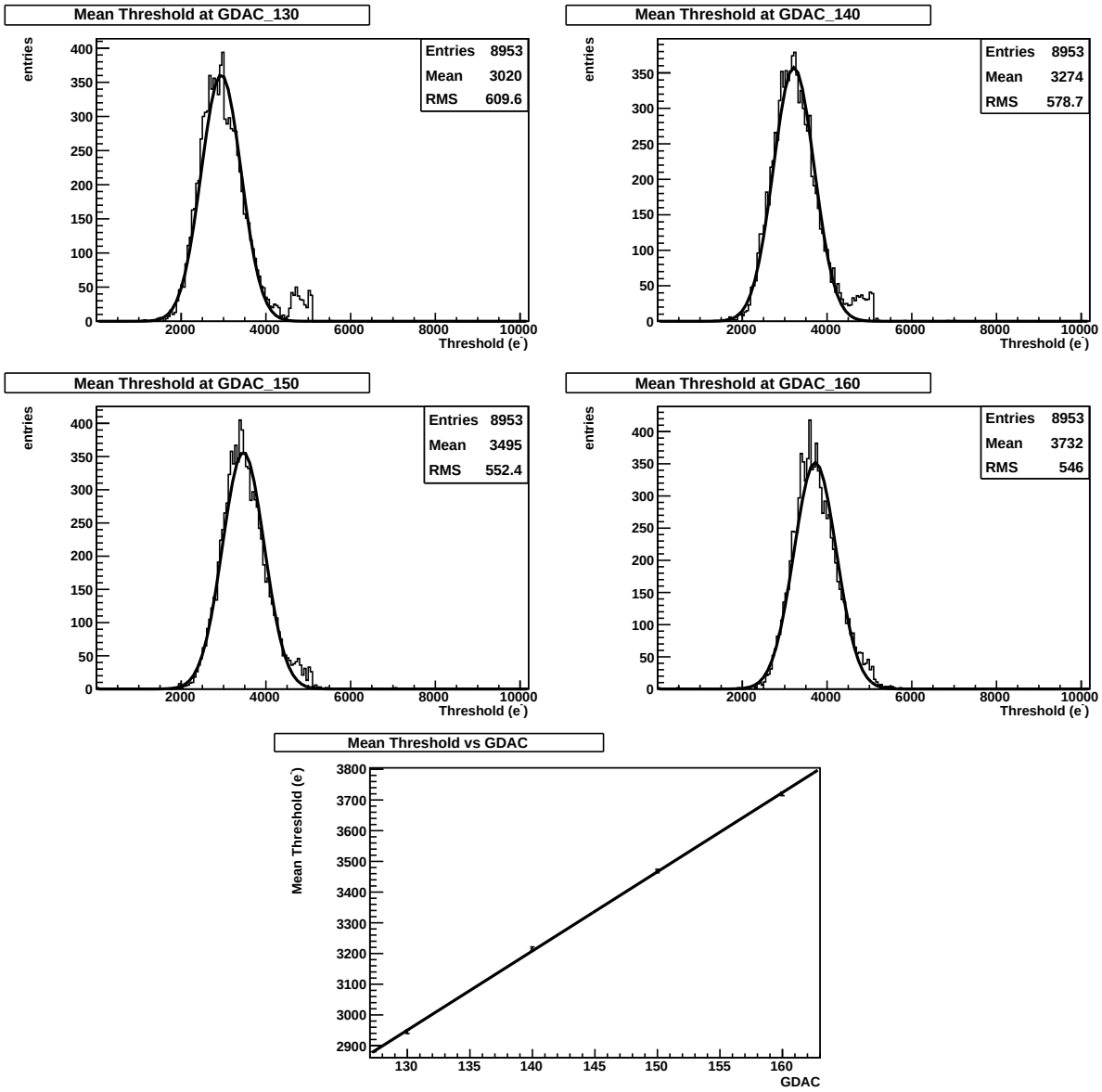


Figure 4.13: The global threshold tuning. Threshold distribution at four different global threshold register values (130, 140, 150 and 160). The average threshold versus global threshold register was interpolated to find the correct global threshold register value for a target threshold of 3000 electrons.

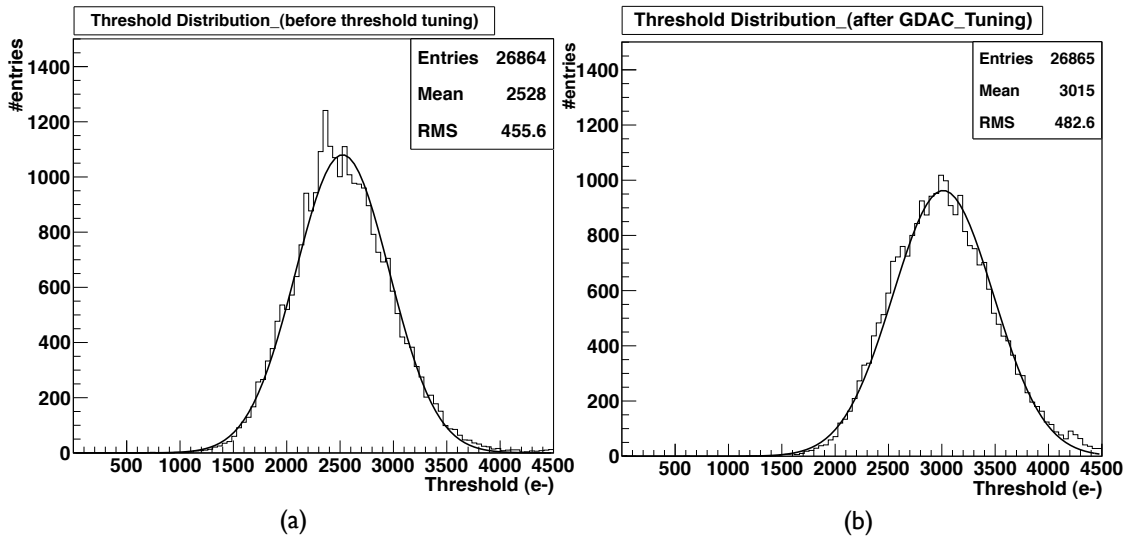


Figure 4.14: The threshold distribution: (a) before and (b) after performing the global threshold tuning. The average threshold of the whole FE-I4 was brought near to the target of 3000 electrons. Nonetheless, the spread in threshold still remained at a high value.

Local Threshold Tuning

Each pixel has a 5-bit local pixel threshold tuning register, called TDAC. A finer threshold tuning can be performed by adjusting this local pixel register. To perform a linear search for the correct TDAC which gives the closest value to the target threshold is extremely time consuming. A faster way is to use the binary search algorithm.

Think of the 32 values of TDAC as a sorted array. What binary search does is to find the TDAC value that gives the threshold value that is the nearest or equal to the target by stepping through 2^{N-n} unit per step, where N is the number of bits and n is the n^{th} step. To be more specific, in each step the threshold corresponding to the middle element of the TDAC array is compared with the target threshold. If the threshold of the middle TDAC element is smaller than the target threshold, then in next iteration the same action is repeated on the sub-elements to the left of the current TDAC middle element and vice versa. In the case of equality, the algorithm stops and returns the corresponding TDAC, threshold and noise for that particular pixel. Otherwise if there is no more element to search for, the algorithm returns the TDAC that gives the nearest value to the target threshold. A flow chart for the binary search is illustrated in Figure 4.15.

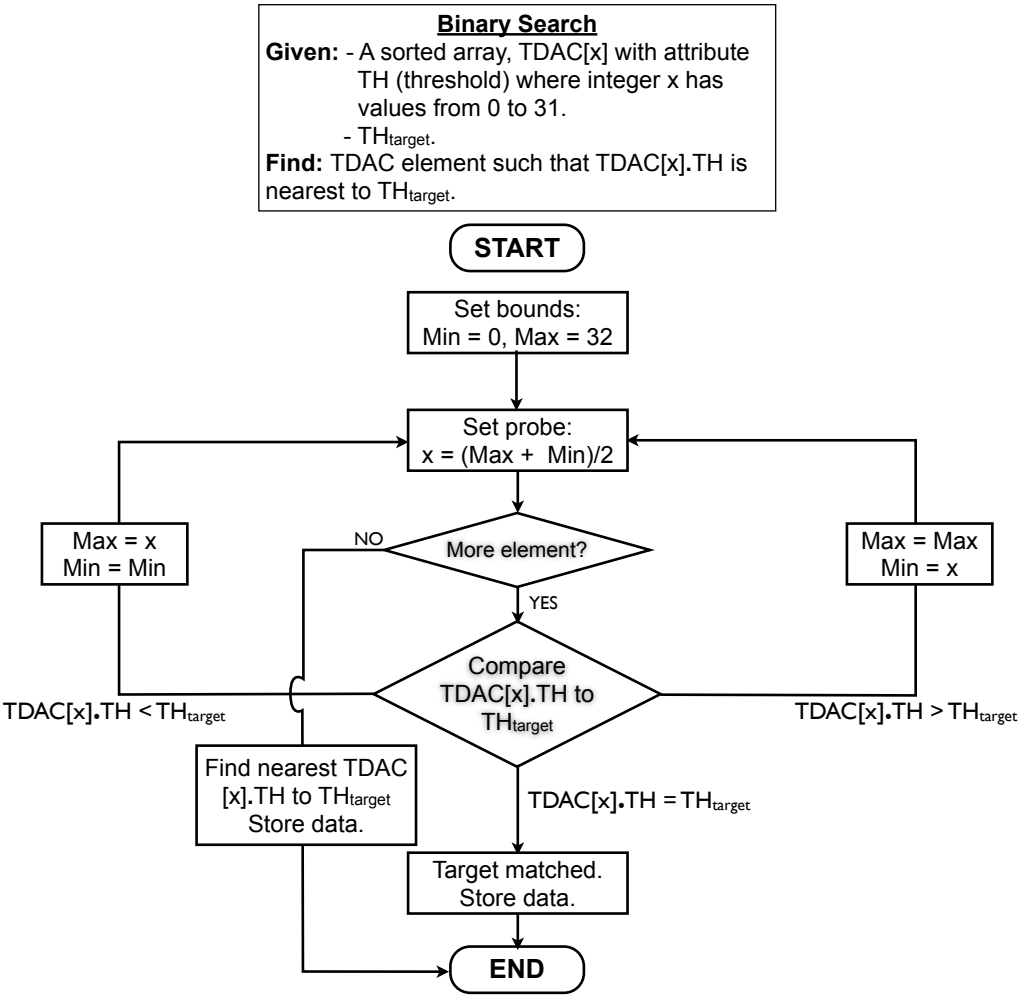


Figure 4.15: The binary search flow chart has three comparison paths: one for equality, one greater than and one for less than. The equality path is rarely taken as most of the time the threshold does not match with the target threshold.

The expected number of iterations in a successful search is $\log_2(N)$ and the worst case is $\log_2(N)+1$. In our case N = 32 thus the average number of iteration to cover all the 32 TDAC values is 5 while the largest number of iterations is 6. The extra one iteration comes from the rare case of TDAC reaching zero. A binary search decision tree covering all 32 values of the 5-bit TDAC register is shown in Figure 4.16.

An implementation example of the binary search for one pixel during the TDAC tuning is shown in Figure 4.17. The TDAC initial value was 16 and depending on the comparison with the target threshold, TDAC value in subsequence steps was

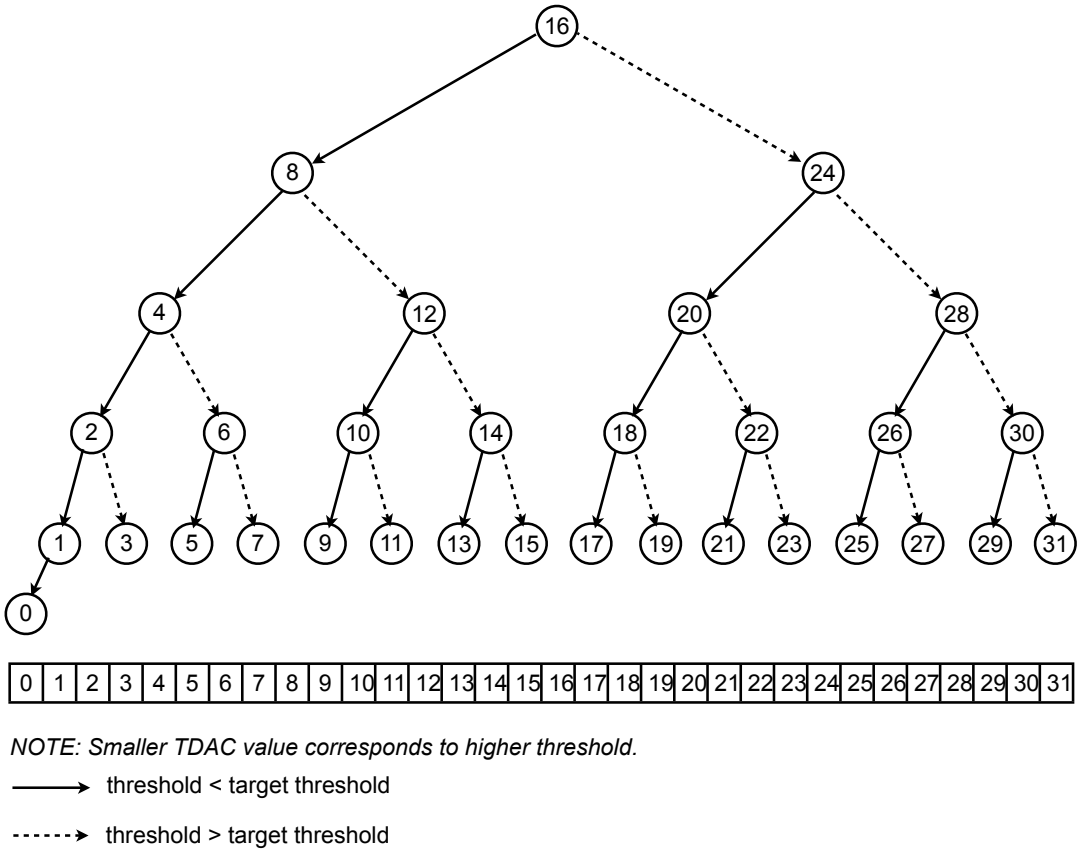


Figure 4.16: A binary search decision tree for our 5-bit TDAC tuning. The threshold value in any node's left sub-tree is smaller than the target threshold while the threshold value in any node's right sub-tree is larger than the target threshold.

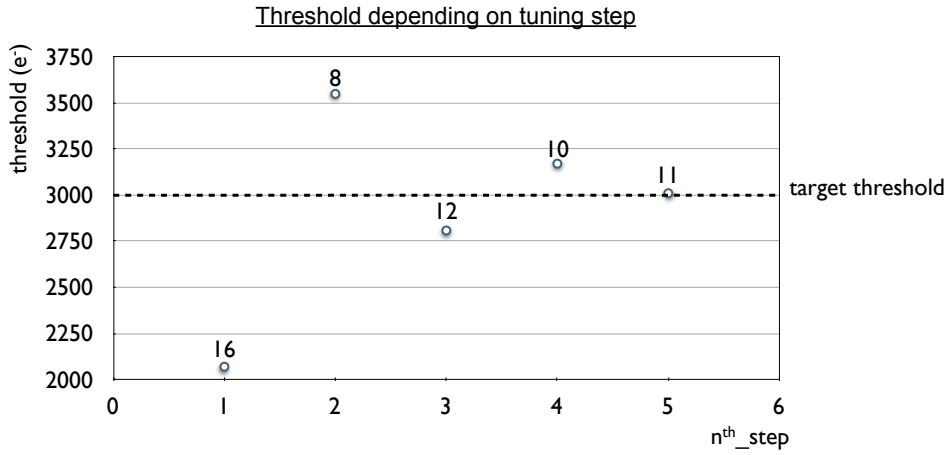


Figure 4.17: Implementation example of binary search for a single pixel. The threshold converged to target threshold of 3000 electrons. The number beside the data point is the TDAC value corresponding to that tuning step.

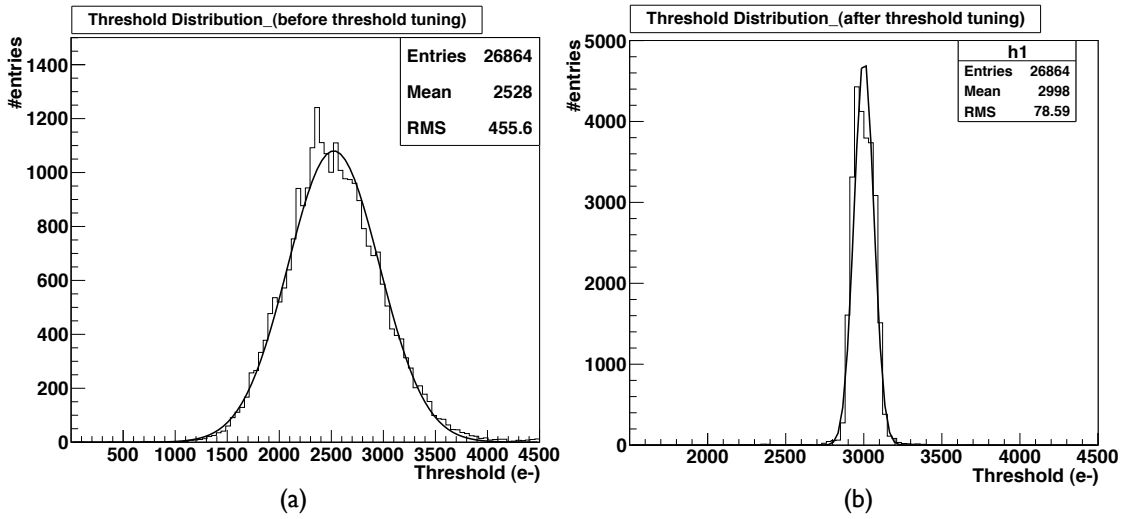


Figure 4.18: Threshold distribution for FE-I4 before and after the tuning.

reduced by 8, 4, 2, and 1. With each advancing step, the threshold converged to the target value.

Figure 4.18 (a) shows the threshold distribution before the Local Threshold Tuning and Figure 4.18 (b) shows the threshold distribution after a successful tuning was performed. After the tuning, the mean threshold has been brought very near to target threshold of 3000 electrons while the threshold dispersion was reduced from around 455 electrons to just 78 electrons. This tuned threshold dispersion satisfies the FE-I4 specification of less than 100 electrons [31].

4.2.3 Time Over Threshold Scan

Other than position information, FE-I4 is also capable of indirectly measuring the charge deposited by particle traversing the sensor through the measurement of Time Over Threshold. This measured charge information can be used to improve the track reconstruction and position resolution. Figure 4.19 illustrates the Time Over Threshold value.

To measure the Time Over Threshold value for a certain input charge value, we perform the so called Time Over Threshold Scan. It is carried out by doing analog injections at a fixed input charge to each pixel and measuring the average Time Over Threshold value of each pixel. Figure 4.20 shows the result of the Time Over Threshold Scan where the average Time Over Threshold is 7.5 bunch crossing.

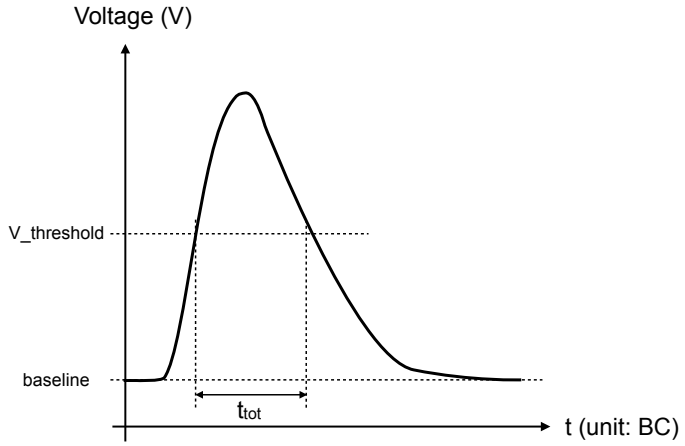


Figure 4.19: Time Over Threshold (TOT) is the period of time, t_{tot} during which input signal to the discriminator is above a predefined threshold.

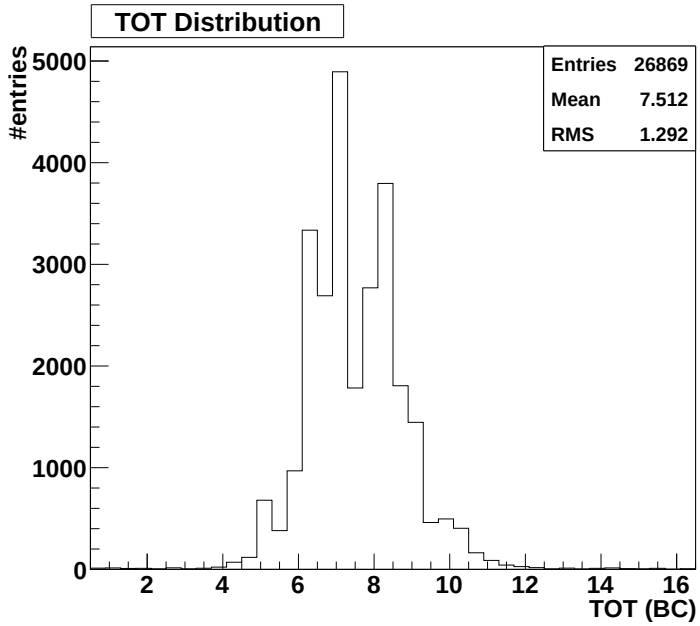


Figure 4.20: Time Over Threshold (TOT) distribution for the whole FE-I4 at a fixed input charge.

4.2.4 Time Over Threshold tuning

The purpose of Time Over Threshold tuning is to obtain a uniform Time Over Threshold response to certain input charge across all pixels on FE-I4. The Time

Over Threshold corresponding to a certain input charge can be adjusted by changing the preamplifier feedback current. The feedback current controls how fast the falling edge of the preamplifier output signal returns to the baseline. As illustrated in Figure 4.21, the higher the preamplifier feedback current, the faster the signal returns to baseline and hence the smaller the Time Over Threshold value.

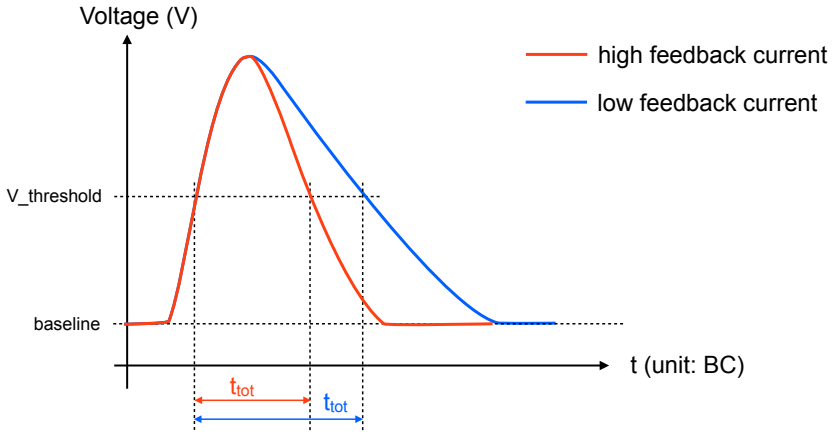


Figure 4.21: Effect of the feedback current on the Time Over Threshold value. The higher the feedback current, the smaller the Time Over Threshold value.

Similar to threshold tuning, there are two kinds of Time Over Threshold Tuning. One is the Global Time Over Threshold Tuning and the other is the Local Time Over Threshold Tuning.

Global Time Over Threshold tuning

PrmpVbpf is a 8-bit coarse adjustment for the master feedback current of the preamplifier which is common to all pixels. Before performing the fine tuning, the preamplifier feedback current has to be tuned at the front-end IC level by using the so called Global Time Over Threshold Tuning.

The Global Time Over Threshold Tuning aims at obtaining an optimum PrmpVbpf value that will be used in the fine tuning process. This optimum PrmpVbpf value should bring the average Time Over Threshold close to a target value at a fixed input charge. The Global Time Over Threshold Tuning algorithm steps through all 255 possible PrmpVbpf values. At each step, a Time Over Threshold Scan is performed for the selected pixels. Figure 4.22 shows the Time Over

Threshold value as a function of PrmpVbpf. In this case, the input charge was set to 20000 electrons and the target Time Over Threshold was 10 bunch crossing. From the plot, the optimum PrmpVbpf is determined to be 78.

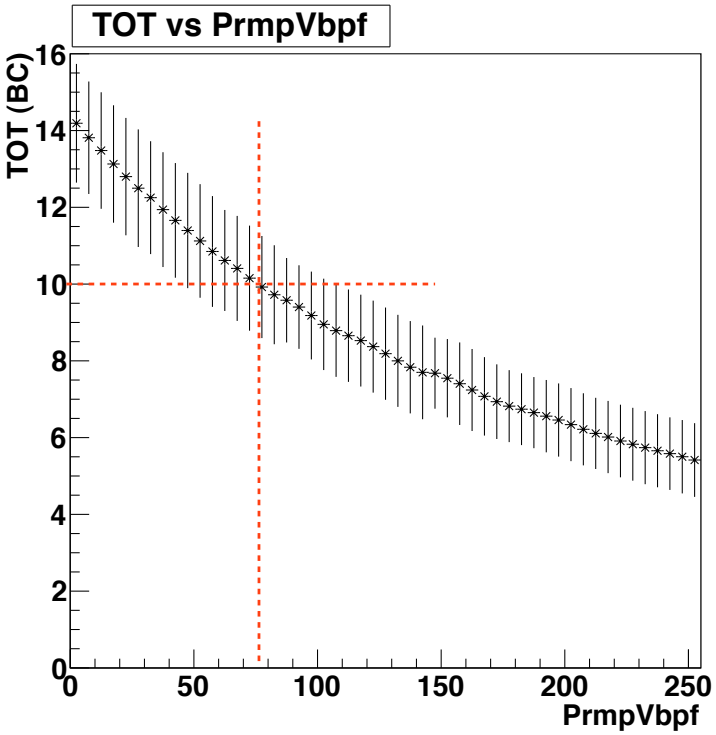


Figure 4.22: Average Time Over Threshold per FE-IC as a function of PrmpVbpf setting. The error bar represents the spread in the Time Over Threshold distribution.

Local Time Over Threshold Tuning

The Local Time Over Threshold Tuning aims at getting a uniform Time Over Threshold response across all pixels for a fixed injected charge. This tuning is performed by adjusting the feedback current of each pixel through the 4-bit local pixel feedback current tuning register, FDAC.

This tuning is performed after obtaining the optimum PrmpVbpf setting for the average Time Over Threshold per front-end IC. In order to cover all values in the FDAC range with minimum number of steps, this tuning employs the same binary search algorithm as in the threshold tuning discussed in Section 4.2.2. Similarly, the middle FDAC value is chosen to be the binary search starting point. At

each step, a Time Over Threshold Scan is performed and the mean Time Over Threshold for each pixel is recorded. Then, the FDAC value that gives the closet Time Over Threshold value to the the target value is selected for each pixel.

A comparison of the tuned Time Over Threshold value with the untuned value is shown in Figure 4.23. In this case, the Time Over Threshold was tuned to 10 bunch crossing at 20000 electrons.

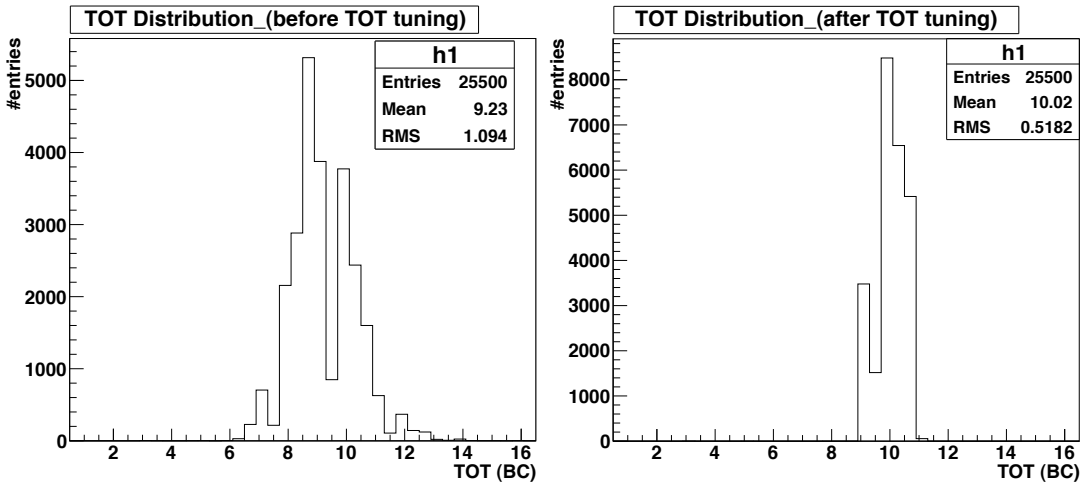


Figure 4.23: The Time Over Threshold tuning. In this case, the target Time Over Threshold value was 10 bunch crossing at input charge of 20000 electrons.

Chapter 5

Readout Speed

5.1 Target and requirement

One of the most important attributes of a readout system is the readout speed. It is especially crucial in the case of multi-chip (4-chip in our case) readout as the readout time consumption scales linearly with the number of front-end IC to be read out. Thus, readout speed optimization on a single chip setup is needed in preparation for the 4-chip readout system.

We started off by defining the readout speed to be how fast it takes to complete an operation for all pixels on a single FE-I4. A complete operation includes everything from the start of the execution of the control software until the end of its execution. The threshold tuning has been chosen to be the reference operation for the readout speed measurement since it covers all DAQ sub-operations and involves a large number of sub-operation steps. This provides a reliable statistics to compare the optimization result.

Of two of the currently available FE-I4 readout system, the RCE readout system is the fastest. It take about 23 minutes to complete a threshold tuning [25]. For the USBpix system, typically about 40∼50 minutes is needed. The initial target is to achieve a complete threshold tuning in 1 hour or faster.

With the target set, we need to identify which stages of a typical readout operation are most likely to create bottlenecks and contributes to a reduction in readout speed. A reasonable guess would be the return path, i.e from FE-I4 to SEABAS and from SEABAS to the computer, where heavy data traffic is expected. On this path, the most probable source of data congestion would be the receiver and decoder module in the software and the receiver module in the User FPGA's

firmware. With these in mind, we can start the investigation one by one.

5.2 Data transfer rate of SEABAS

The SEABAS DAQ board currently in use is version 1.1 which is capable of providing 100Mbps (12.5MBps) data transfer rate. The first step in readout speed optimization is to verify this specification. Secondly, we also need to find out which data receiver's coding style gives the highest data transfer rate.

Since we are interested in measuring only the data transfer rate between the SEABAS DAQ board and the computer, there is no need to involve FE-I4. Instead of getting data from FE-I4, a driver operated by a 12.5MHz clock was prepared in the firmware to send a 8-bit dummy data packet continuously. This ensures the data packet is always sent out at a rate of 12.5MBps. Referring to the simplified flow chart for the data transfer from SEABAS to computer in Figure 5.1, the firmware starts sending out a stream of 8-bit TCP data packets as soon as it receives the start signal from the computer. Those TCP data packets are directly placed into the kernel buffer as soon as they are available at the network input port. This process of placing incoming TCP packets into the kernel buffer is managed by the operating system which we have left it untouched. Once the data packets are stored in the kernel buffer, we check how much data there is for reading, and then write it into the user defined buffer where our execution code can have access to. While taking data out from the kernel buffer, the total amount of data is also counted. When enough data is taken, a STOP signal is sent to the firmware to stop the data transfer and the whole process ends.

We are interested in the time it takes the data packet to go from SEABAS to PC. However, it is hard to measure it precisely without an elaborate setup. In contrast, it is much easier to estimate the data transfer rate by measuring the overall time from START to END. In other word, the total time is equal to t_1 + t_2 + t_3 + t_4 as indicated in Figure 5.1. We define the TCP transfer rate as:

$$\text{transfer rate} = \frac{\text{total data}}{\text{total time}} = \frac{D}{(t_1 + t_2 + t_3 + t_4)} \tag{5.1}$$

To retrieve data from the kernel buffer, first we have to check whether any data sent from SEABAS has arrived at the input port to PC. This is achieved by calling the standard socket function called select(). Then the socket function, recv(), is

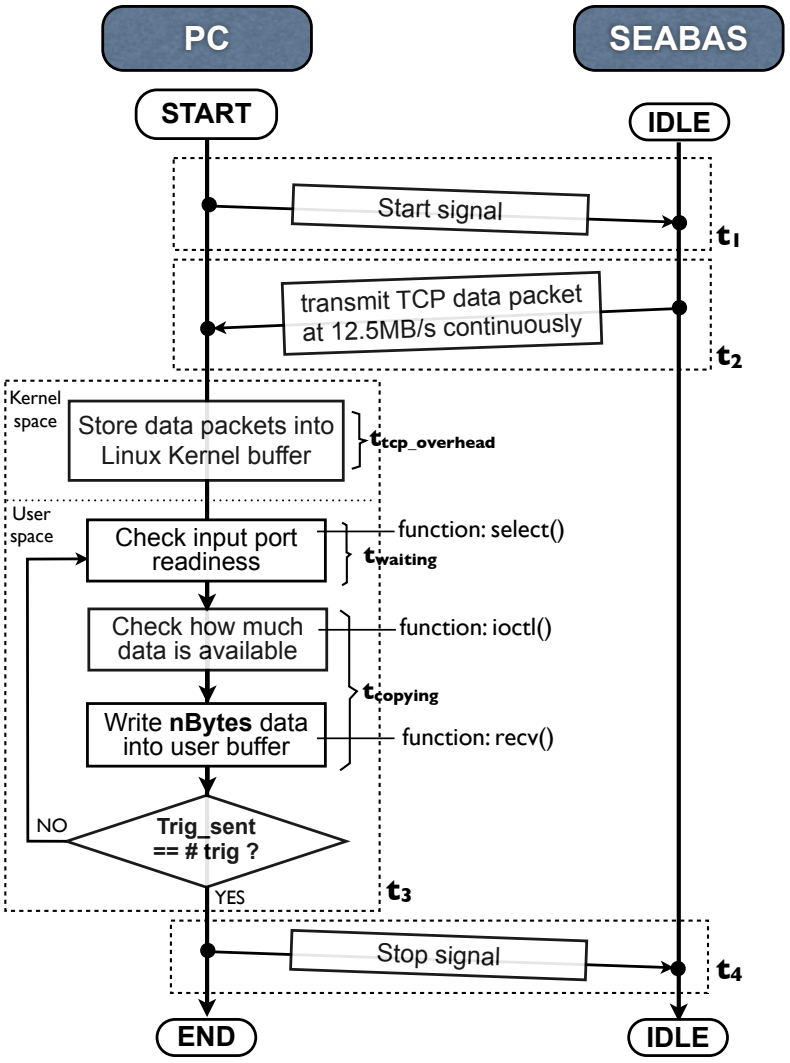


Figure 5.1: Flow chart for measuring the data transfer speed by SEABAS.

used to transfer the available data in the kernel buffer into the user buffer. We can specify how much data to transfer per recv() function call. Here we compared three options:

- Option 1: transfer 1Byte per function call
- Option 2: transfer 6.25MBytes per function call
- Option 3: transfer all immediately readable data (nBytes) per function call

For Option 3, there is an additional step to perform. We need to check how much data is available in the kernel buffer that is ready for reading out. This is carried out by calling the socket function, ioctl(). On success, the ioctl() function

returns the number of bytes of data that is available for reading. This return value is then passed to the recv() function. For Option 1 and Option 2, this step is skipped as we have already specified how much data to transfer out. Furthermore, in the case of Option 2 where a large amount of data is specified, the recv() function will simply block the program and wait for all the specified amount of data to be fully transferred.

Besides, for each of these options, two different computer systems were also compared. One is a 2.4GHz Core 2 Duo, 4G RAM machine running Mac OS X and the other is a 2.8GHz Quad-core Intel i7, 8G RAM machine running Scientific Linux 6.3.

Table 5.1: Measurement of data transfer rate from SEABAS to PC.

PC	$\frac{\textbf{nBytes}}{\textbf{function call}}$	**Total data, D (MB)**	**Total time (s)**	**Data transfer rate (MB/s)**
Mac	1	12.5	50.833	0.2459
	6.25×10^6	12.5	2.324	5.378
	n	12.5	2.120	5.895
Linux	1	12.5	795.732	0.01571
	6.25×10^6	12.5	1.048	11.922
	n	12.5	1.032	12.118

Table 5.1 shows the result of the measurement of data transfer rate from SEABAS to PC. At first glance, the transfer rates are 2 times higher for the Linux machine than the MAC OS X machine. For both machine the transfer rate is generally higher when more data is transferred out from the kernel buffer per function call. In any case, transferring just 1 byte at a time seriously impairs the transfer rate especially for the Linux machine. This might be caused by the fact that different Operating System implements different TCP tuning technique. Taking just 1 Byte per function call obviously hurt the network performance of Linux system. Moreover, this method involves a large number of repetitions which obviously adds to the time, t_3.

We also observed that by taking out a large amount of data per function call on Linux machine, we could reach a transfer rate close to 95% of the SEABAS's Ethernet 12.5Mbps specification. Nonetheless, in real data taking scenario we might not always have a long continuous stream of data coming from FE-I4 at all

time. What is more, we cannot know for sure how much data is being transferred at any given time. Thus, specifying a large amount of data to be transferred from the kernel buffer per function call may actually block the program to run until the specified amount of data is returned. If not enough data is sent from FE-I4, the time spent for waiting the extra data is wasted. Therefore, this option is not implemented.

From the table, we can see that the readout scheme using n-bytes per function call in Linux system has the highest transfer rate. It recorded a 12.118MB per second of data transfer rate which is roughly 97% of the SEABAS's 12.5Mbps specification. In this method we transfer out only what is readily available at the moment of the function call. In this manner we save the time spent on waiting. One thing to be kept in mind is that the actual time taken for data transfer is only t_2, hence the exact transfer rate is actually faster than what we have measured here. This data retrieving method of n-bytes per function call has been incorporated into our readout system.

Moreover, the sharp difference between taking out just one Byte per function call and the other two methods shows us that the data transfer speed is limited by t_3 instead of the actual data transfer from SEABAS.

5.3 Optimization

As mentioned before, the threshold tuning is the reference operation for the readout speed measurement. The target is to make the tuning time under 1 hour. The parameters in a threshold tuning that have direct influence on the readout speed are the number of scan points, the number of mask steps and the number of triggers. To set a benchmark threshold tuning that can be compared against, the parameters have been set to:

1. the number of scan points: 50
2. the number of mask stages: 120
3. the number of triggers: 50

Next, the total readout time for the threshold tuning operation was broken down into its sub-components so that they become simple enough to be optimized

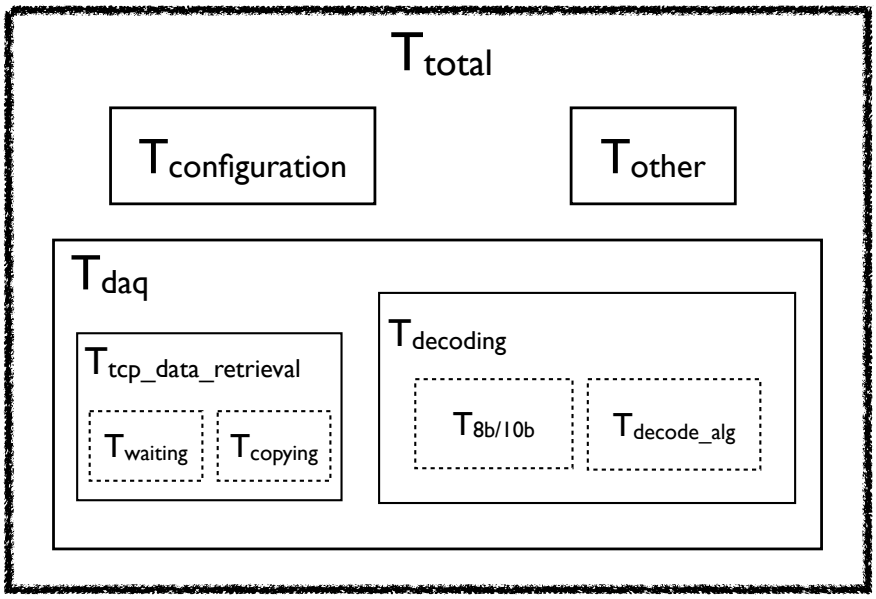


Figure 5.2: Total readout time broken down into its sub-categories.

directly. Referring to Figure 5.2, the total time, T_{total} can roughly be grouped into the major categories as below:

1. $T_{configuration}$ — This includes all the time needed to configure FE-I4 so that it is in a correct state of operation.

2. T_{daq} — This can be sub-divided into two smaller categories as follows:

   (a) $T_{tcp\_data\_retrival}$ — This is the time needed to retrieve the TCP data packet sent from SEABAS to the user buffer on PC. It includes the waiting time for the TCP data packet to arrive at kernel buffer, $T_{waiting}$, and the time needed to copy the data from the kernel buffer to user buffer, $T_{copying}$.

   (b) $T_{decoding}$ — This is the overall decoding time that can be further broken down into $T_{8b/10b}$ and T_{decode_alg}. $T_{8b/10b}$ is mainly the time to decode the 10-bit 8b/10b encoded symbols to the 8-bit raw data. Likewise, T_{decode_alg} encompasses the rest of the decoding algorithm, for example, frame matching, data parsing, data extraction and so on.

3. T_{other} — This includes other routines that cannot fit into the major categories, for example, creating, filling and saving a ROOT files as well as any

overhead in every function call.

Three factors that greatly influence the readout speed are listed below:

1. interval between each trigger, t_{trig}
2. TCP data packet retrieval method
3. data decoding

We will investigate the effect of each of these factor on the readout speed. The rest of the tests were carried out based on the 2.8GHz Quad-core Intel i7, 8G RAM machine running Scientific Linux 6.3.

5.3.1 Minimizing the trigger interval, t_{trig}

Remember that the interval between each trigger, t_{trig}, is controlled by the firmware. The t_{trig} affects the $t_{waiting}$ an thus the total readout time. To see how the interval between each trigger affects the readout speed, a simple test was performed. By comparing the total time taken to complete several threshold tunings for just two columns with different interval between each trigger, we can immediately conclude that the smaller the interval, the faster the readout speed. This can be seen in Table 5.2.

Table 5.2: Total readout time for threshold tuning scans on two columns with different interval between each injection, t_{trig}.

$\mathbf{t_{trig}}$ **(ms)**	Total time (s)
2.4	114.3
1.2	70.5
0.6	49.6
0.3	44.8

Even though a small t_{trig} value can provide faster readout speed, it cannot be set to a value that is extremely small. This is because if the next trigger arrive too early, incomplete data transfer from the FIFO will occur. This in turn cause the DAQ program to fail. In view of this, t_{trig} value has to be carefully set. Currently the t_{trig} value is determined by trial and error. A better solution will be discussed in the next chapter.

5.3.2 Data retrieval method

As mentioned in Section 5.2, we have chosen the Option 3 — take all immediate readable data (n-bytes) per function call, as the data transfer method. When combining this method with the decoder, it can give two possible algorithm varieties. Referring to Figure 5.3, first variation is to loop back the data transfer function call until there is no more data available in the kernel buffer for transferring. This

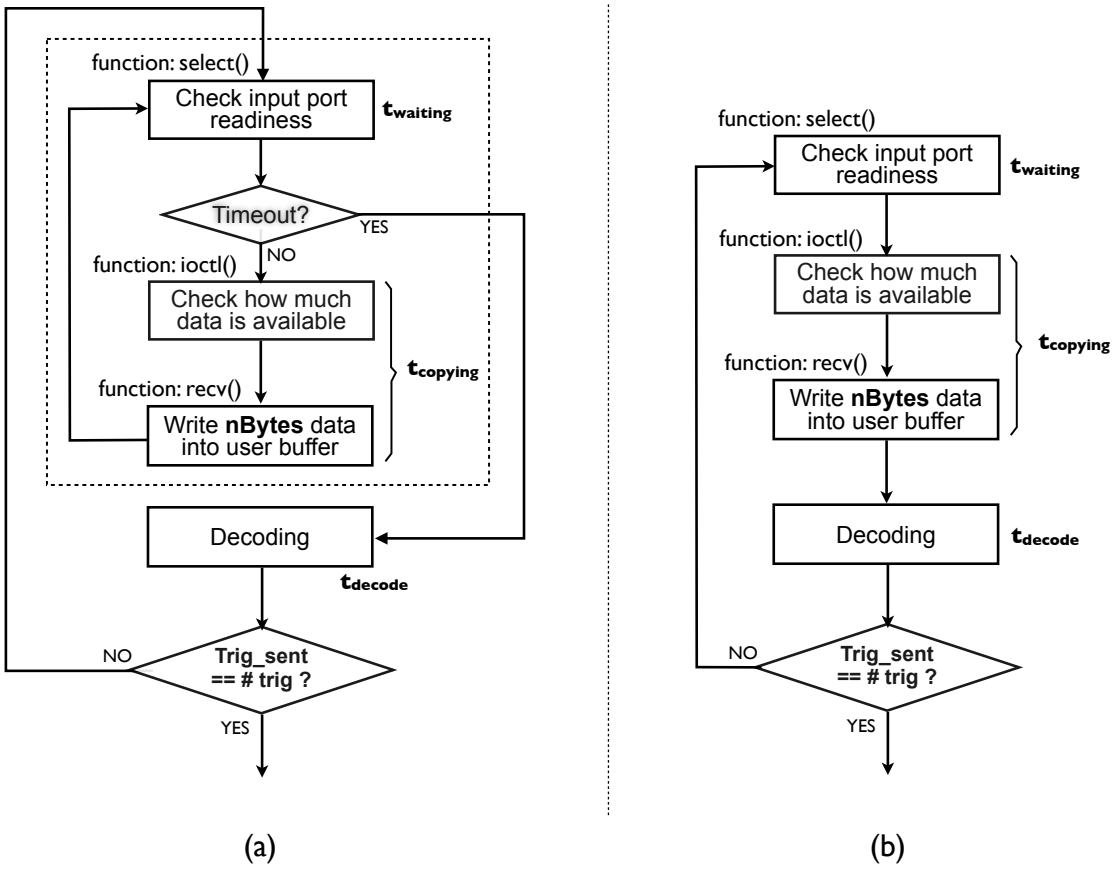


Figure 5.3: Two flow charts showing two variations in TCP data retrieval method. (a) n-bytes per function call — loop back method. (b) single n-bytes per function call method.

happens in the inner loop. After receiving all data, they are passed to the decoder. We will refer to this method as "n-bytes per function call — loop back". Second is to couple the data transfer function call directly to the decoder. We call the data transfer function only once. Then the received data is passed to the decoder for processing. Let us call this method as "single n-bytes per function call". In both methods, after the decoder, we check whether we have received all data record corresponding to the number of events we specified. If not, the whole process is

repeated again.

We compare these two methods in order to see which gives the best result on readout speed. For the sake of completeness, we also show the result of the Option 1 which has been mentioned in Section 5.2. In this method, the data transfer function call is directly coupled to the decoder. Here we refer these methods as:

1. Method 1: single byte per function call
2. Method 2: n-bytes per function call — loop back
3. Method 3: single n-bytes per function call

Table 5.3: Readout time for threshold tuning using different TCP data packet retrieval methods. The threshold tuning was performed on 76 columns on FE-I4A. Method 3 gave the fastest speed with $T_{total} \approx 48$ minutes.

Method	$\mathbf{t}_{trig}$ **(ms)**	$\mathbf{T}_{config}$ **(s)**	$\mathbf{T}_{other}$ **(s)**	$\mathbf{T}_{waiting}$ **(s)**	$\mathbf{T}_{copying}$ **(s)**	$\mathbf{T}_{decoding}$ **(s)**	$\mathbf{T}_{total}$ **(s)**
Method 1	1.2	225.00	325.70	794.23	305.58	6837.03	8487.54
Method 2	1.2	218.82	223.48	1306.71	4.97	2210.48	3964.46
Method 3	1.2	218.25	240.28	287.05	1.50	2162.02	2909.10

Table 5.3 shows the result for each method. Trigger interval of 1.2ms was used in all the tests. In general T_{config} stayed roughly the same for all three methods since it has nothing to do with the readout of data from FE-I4. As expected, method 1 recorded the longest time to complete a threshold tuning. It took 8487s or roughly 2.4 hours. Every other time categories also registered a high value for this method. The main reason is the excessive repetition of the operation such as calling the select(), ioctl(), recv() and other functions in the DAQ algorithm. Typical data size for a complete threshold tuning is around 1GB. When taking out just one byte per function call, we have to repeat $1x10^9$ times of data retrieval. The overhead in each operation or in each function call alone will aggregate to a considerable amount of time. Furthermore, the decoder algorithm matching this method is based on a straight forward but inefficient bit shifting approach. This, combined with the overhead of each function call contributes to the large decoding time.

Focus should be given to method 2 and method 3. They are similar in almost every aspect except the small difference in the data transfer method. However,

1055s difference in total readout time arose because of this slight dissimilarity. In method 2, we cannot know in advance how many data is being sent from SEABAS at any given time. For that reason, we cannot know when we should exit the data transfer loop. We use the select() function to check the input port for its readiness for reading. This function blocks the DAQ program until there is data available for reading out or it reaches a preset timeout period, whichever comes first. The only breaking condition is the timeout, which means all the data sent from SEABAS for certain cycle has been completely received. Until this timeout occurs, nothing useful can be performed. In the end, the timeout adds up to an inordinate amount of time of waiting for nothing.

To get rid of this wasteful operation, we modified the algorithm to directly pass the received data to the decoder after each data receive function call. That is the method 3. It gave the fastest time in almost all the readout time categories. We saved around 1020s in $T_{waiting}$ alone which is a saving of more than 70%. Another 50s came from the $T_{decoding}$ but it was offset by a slight increase in T_{other}. In effect, we managed to reduce the total time spent in a threshold tuning down to $\approx$ 48 minutes.

Note that these tests were carried out on just 76 columns on a FE-I4A chip. Four other columns are excluded from the test as they did not pass the analog injection test. But since the total readout time is roughly scale with the number of columns under test, we can estimate that a complete threshold tuning on 80 columns to be around 50 minutes.

From the data, we see that the decoding time makes up nearly 75% of the total time. Obviously there is still room for speeding up the program by optimizing the decoding algorithm. Hence, next section will be dedicated for improving the decoder.

5.3.3 Reducing the decoding time

As discussed in previous section, the decoding time, $T_{decoding}$ can roughly be divided into the $T_{8b/10b}$ and T_{decode_alg}. To estimate how much fraction of $T_{decoding}$ is occupied by $T_{8b/10b}$, we perform a threshold tuning on 2 columns of FE-I4 and extract the time consumption of 8b/10b decoder. We observed that more than 50% of the total decoding time was taken up by the 8b/10b decoder in the software.

The large contribution of 8b/10b decoding comes from the fact that it consists

of a large number of "if-else if" conditional statement. In total there are 255 of this statement matching the 255 possible values for a 8-bit integer. For each 10-bits symbol, the odds of getting a match in the first few statement are small. Most of the time a 10-bit symbol needs to go through lots of this statement before getting a match. Naively speaking, out of a typical 1GB data transfer from a threshold tuning, the number of 10-bit symbols is an order of 10^8. The time spent in matching these symbols simply sums up to an enormous amount of time.

One way to eliminate entirely this contribution from 8b/10b decoding operation is to implement the 8b/10b decoder in the firmware. The resultant time saving can be clearly seen in Table 5.4.

Table 5.4: Readout time for threshold tuning with implementation of 8b/10b decoder in the firmware.

Decoder Scheme	$\mathbf{t}_{trig}$ **(ms)**	$\mathbf{T}_{config}$ **(s)**	$\mathbf{T}_{other}$ **(s)**	$\mathbf{T}_{waiting}$ **(s)**	$\mathbf{T}_{copying}$ **(s)**	$\mathbf{T}_{decoding}$ **(s)**	$\mathbf{T}_{total}$ **(s)**
Software	1.2	218.25	240.28	287.05	1.50	2162.02	2909.10
Firmware	1.2	310.85	236.78	937.47	4.97	743.36	2233.40
Firmware	0.3	311.1	201.53	160.541	1.32	606.33	1280.82

The first row is essentially the same result shown as Method 3 in Table 5.3. Comparing the first and the second row of Table 5.4, every category is basically the same except the sharp difference in the T_{decode} and $T_{waiting}$. As expected, a time reduction of more than 60% has been achieved in T_{decode}. In spite of this, a factor of 2.3 increase in $T_{waiting}$ partly negated the saving in T_{decode}. This could be explained by the fact that the implementation of 8b/10b in firmware also reduces the total amount of data sending out. After passing through the decoder, the 10-bit data word is reduced to a 8-bit data word. This reduction in data means the bandwidth is not fully utilized, therefore more time is spent on waiting for next data stream to reach PC. To remedy this, we can reduce the trigger interval, say, by a factor of ten, that is from 1.2ms to 0.12ms. The result is as shown in the third row of Table 5.4. Accompanied this tenfold reduction in trigger interval is approximately a tenfold decrease in the $T_{waiting}$. In effect, we have got a total time of just under 1300s or 22 minutes, which is far lower than our initial target.

5.3.4 Final result

As a summary, the readout speed optimization method that we have implemented in our readout system are:

1. optimization of trigger interval, t_{trig}.
2. single n-byte per function call.
3. 8b/10b firmware decoder.

Previous tests were carried out on 76 columns of a FE-I4A IC. Running a threshold tuning on all columns on a FE-I4B with all these method combined together gave the following result as shown in Table 5.5. A complete threshold tuning on all 80 columns took around 23 minutes.

Table 5.5: Readout time for threshold tuning with t_{trig}= 0.3ms, single n-byte per function call method and 8b/10b firmware decoder.

$\mathbf{t}_{trig}$ **(ms)**	$\mathbf{T}_{config}$ **(s)**	$\mathbf{T}_{other}$ **(s)**	$\mathbf{T}_{waiting}$ **(s)**	$\mathbf{T}_{copying}$ **(s)**	$\mathbf{T}_{decoding}$ **(s)**	$\mathbf{T}_{total}$ **(s)**
0.3	325.28	192.34	170.73	1.50	657.83	1347.68

Chapter 6

Future prospect

Although the readout system can perform most of the essential tasks, it is still far from complete. Our works, albeit significant, constitute only a foundation for grander projects. More works and improvements can still be added in many aspects of the readout system. For one thing, the full potential of the SEABAS DAQ system, or more specifically the SiTCP technology that SEABAS DAQ board is outfitted with has yet to be fully realized. In other respect, one of our eventual goals of simultaneously reading out four or more FE-I4 chips has not yet been fulfilled. In this chapter, several propositions and plans will be presented.

On top of the list are the upgrades that could be performed on the system as a whole, they are:

1. **4-chip readout.**
   As soon as the improved version of the 4-chip daughter card is made available, our current single chip readout scheme should be extended to a four chip readout scheme. The idea is that a single serialized data stream coming from PC to SEABAS is branched off to each chip and vice versa. Destination of commands to each FE-I4 must be specified correctly. When reading out the data, data stream must be labeled with correct FE-I4 ID from which it is originated. When data can be read out correctly, readout speed optimization should also be carried out.

2. **Upgrade to a newer SEABAS version.**
   Two immediate benefits that we can reap by this upgrade are the larger memory and higher Ethernet bandwidth. The newer SEABAS (version 2) has a 50% and 30% increase in the User FPGA's Distributed and Block

RAM memory, respectively. In addition, it boosts a 1Gigabits per second data transfer rate. These upgrades are beneficial in the case of reading out four FE-I4s. More importantly, the larger memory allows for larger FIFOs and thus more pixels can be read out at the same time. This reduces the mask stages and consequently we can reduce the configuration time and waiting time. On the other hand, the higher bandwidth is crucial for accommodating the increase in the amount of data especially when reading out four FE-I4s. Hence preventing the build up of a bottleneck at the PC-SEABAS interface. The increase in readout speed may not be too significant though. As a crude estimation, a complete Threshold Tuning generates roughly 0.75GB of data. A 1Gbps transfer rate means we can reduce the $T_{waiting}$ by roughly 56s.

3. **Custom-made DAQ main board and 4-chip daughter board improvement.**
   The SEABAS DAQ main board is a general purpose DAQ board with various useful features. Nonetheless, not all are currently used in our readout system for FE-I4. If circumstances allow and if the necessary resources are available, it may be good to have a DAQ board that is customize to suit only our needs. For the new DAQ main board, we may:

   - Retain only the components or features that are necessary, and remove the redundant one such as the onboard ADC, DAC, extra 64-pin connectors, signal probing nodes and etc. This may result in a more compact physical size.
   - Add a power regulator which can regulate and supply voltages for both FE-I4's digital and analog circuits. With this option, we do not need too many low voltage supplies.
   - Add external memory such as SRAM to increase the amount of pixels that can be readout simultaneously.

   In the case of the 4-chip daughter board, rearrangement of the components may be made to make it more compact.

We have shown in previous chapter that we managed to reduce the readout speed considerably. To further improve the readout speed, two possible solutions are:

1. **Implementation of frame matching, data packet unpacking and extraction in firmware.**
   In last chapter, we see that the implementation of the 8b/10b firmware decoder significantly reduces the readout time consumption. But on close inspection, the frame matching, data unpacking and the extraction of hit information which are performed in the software still contribute to more than 50% of the total readout time. It may be possible to implement these routines in the firmware. If successful, it is possible to further increase the readout speed.

2. **Parallel computing**
   Multi-core processor has proliferated over the last few years. The possible gain in readout performance by using parallel computing in reading out multiple front-end ICs could be huge. A naive implementation could be just to run the same DAQ program for each of the front-end chip. In any case, the computation time should be able to be reduced by at least a factor of four than in the serial computation case.

For the firmware, the following improvement could be made:

1. **Active trigger interval control.**
   As we have seen in the previous chapter, the trigger interval, t_{trig} has extensive effect on the readout speed. However, its value cannot be too small as it causes an immature truncation of data transfer. Currently, the trigger interval is fixed to a certain value which has to be specified by a user. The default value is set at the minimum value that is thought to be suited for all functionalities. However, the fact is that there is no one-size-fits-all value for all the tests or scans. Instead of being such a static value, it should be a dynamic value that changes based on the readout condition. Hence, a mechanism to determine the optimum trigger interval during runtime should be implemented.

2. **Dynamic reconfigurable system clock.**
   Clock synchronization between the User FPGA data receiver and the FE-I4 data output transmitter is very important. At present, we have no efficient way of checking this as the digital clock manager in the firmware that generates clocks for driving certain logic block is not reconfigurable. To reconfigure

even only the digital clock manager means we have to reload the updated firmware onto the FPGA, which is very inconvenient. Hence, the ability to change the digital clock manager's attributes in order to select a different phase shift or frequency is highly desirable.

On the other hand, some improvement that can be made to the software are:

1. **User customizable software framework.**
   Other than the standard ready-made functionalities, there would be instances where users need to device their own custom test or scan to suit their specific needs. Hence, it is very preferable to provide a set of generic functionalities that can be selectively changed or specialized by user code to produce application specific software. On top of that, a universal, reusable software framework is also vital in ensuring software extensibility. A typical example of this type of software framework is the ROOT. To accomplish this goal, code refactoring should first be performed to limit the interdependencies between software components, thus reducing system complexity and increasing its robustness and maintainability.

2. **More user-friendly interface.**
   Graphic user interface (GUI) should be provided to facilitate users' operation. An example GUI can be a 80 x 336 matrix that represents each pixel. This could be useful for user to select or mask any pixel they want.

Of all these possible improvements or upgrades, the extension to 4-chip readout and the active trigger interval control are high on our list of priorities. We plan to start developing the 4-chip readout system as soon as the new 4-chip board is ready. At the mean time, we will work on the implementation of the active trigger interval control in the firmware.

Chapter 7

Conclusion

As a conclusion, we have successfully developed the firmware and software of the readout system for the new ATLAS pixel front-end readout IC, FE-I4. Our readout system is intended to be used as an electronic test stand for FE-I4 as well as a readout system for the module consisting of sensor and FE-I4. The main features of our readout system is the SEABAS DAQ board which utilize the SiTCP technology. It offers us the qualities of compact physical size and flexibility. Furthermore, the firmware and software are designed to be versatile enough to suit most of the users' needs.

DAQ operation tests have been performed and we confirmed the following points:

1. communication with FE-I4 has been successfully established.
2. configuration of specific global registers and local pixel registers can be performed correctly.
3. calibration command and Level 1 trigger command can be sent to FE-I4.
4. decoding and data extraction can be carried out correctly.

On top of that, we have prepared and tested several essential functionalities such as Injected Charge Scan, Threshold Tuning, and so on. These scan and tuning operations have been shown to perform as designed. In the effort to improve the readout speed, we found that the combination of the "transferring n-bytes per function call" method, small trigger interval and the 8b/10b firmware decoder gives a significant improvement of readout speed. The time consumption of a complete

threshold tuning, which we used as a reference operation, has been shown to be around 22 minutes.

All in all, we have accomplished the objectives that we set out to achieve. Nonetheless, the readout system development is still in its infancy. Many improvement can still be made especially the upgrade to read out four FE-I4s simultaneously as well as the implementation of the active trigger interval control.

Appendix A

FE-I4B Specification

Item	Value	Units
Pixel size	50×250	μm^2
Bump pad opening diameter	12	μm
Input	DC-coupled -ve polarity	
Maximum charge	$100,000$	e^-
DC leakage current tolerance	100	nA
Pixel array size	80×336	Col $\times$ Row
Last bump to physical chip edge on 3 sides	≤ 100	μm
Last bump to physical edge on bottom	≤ 2.0	mm
Normal pixel input capacitance range	100-500	fF
Edge pixels input capacitance range	150-700	fF
In-time threshold with 20 ns gate (400 pF)[1]	≤ 4000	e^-
Hit-trigger association resolution	25	ns
Same pixel two-hit discrimination (time)	400	ns
Single channel ENC sigma (400 fF)	< 300	e^-
Tuned threshold dispersion	< 100	e^-
Charge resolution	4	bits
ADC method	ToT	
Radiation tolerance (specs met at this dose)	300	Mrad
Operating temperature range	-40 to $+60$	°C
Average hit rate with $< 1\%$ data loss	400	MHz/cm^2
Readout initiation	Trigger command	
Max. number of consecutive triggers	16	
Trigger latency (max)	6.4	μs
Maximum sustained trigger rate	200	kHz
External clock input (nominal)[2]	40	MHz
Single serial command input (nominal)[2]	40	Mb/s
Single serial data output (nominal)[2]	160	Mb/s
Output data encoding	8b/10b	
I/O signals	LVDS	

*FE-I4B specification, taken from Reference [31].

Appendix B

FE-I4 Registers

Here we list down a few registers that are mentioned in this thesis. Their description are taken from [30, 31]. A complete list can be found in Reference [30, 31].

B.1 Global registers

1. **DisableColCnfg**
   Size: 40-bit
   40 bits that set the Double Column mask. Enable a particular bit will disable the digital portion of the selected Double Column.

2. **PlsrDAC**
   Size: 10-bit
   It set the calibration injection voltage value (V_{Cal}).

3. **PrmpVbpf**
   Size: 8-bit
   It controls the global feedback current of the preamp that is common to all pixels. It sets the fall time of preamp output which in turn determines the ToT LSB scale.

4. **Trig_Count**
   Size: 4-bit
   Number of consecutive triggers to send upon issue of trigger command. Since the trigger command itself takes 5 clock cycles to issue. If there were no multiplier, the most often we could trigger would be every 5 clock cycles. Value 0000 means 16 consecutive triggers.

5. **Trig_Lat**
   Size: 8-bit
   8-Bit complement of trigger latency in clock cycles (true latency = 255 - Trig_Lat). 255 is an invalid value.

6. **Vthin_Fine and Vthin_Coarse**
   Size: 16-bit
   Together, they are known as GDAC. Each of them is a 8-bit register for the fine and coarse threshold adjustment, respectively. The scale is non-linear ranging from 0 to 1V.

B.2 Local pixel registers

For each pixel there is one 13-bit long register called PxStrobes. Each bit in PxStrobes has its own function. There are shown as follows:

1. **bit 0**
   Must be set to 1 to enable each pixel.

2. **bit 1 to 5**
   Also known as TDAC. It adds a local voltage offset to the global threshold in each pixel. Bit 1 is the most significant bit.

3. **bit 6**
   Set this bit to 1 to enable charge injection through the large injection capacitor.

4. **bit 7**
   Set this bit to 1 to enable charge injection through the small injection capacitor.

5. **bit 8**
   Set this bit to 1 to monitor leakage current. Set to 0 to include pixel in hit bus.

6. **bit 9 to 12**
   Also know as FDAC. It adjusts the amplifier feedback current for each pixel. Bit 12 is the most significant bit.

7. **bit 13**

   It is also called Shift Register. Its primary function is to act as a data input to the other 12 bits. Apart from that, it must be set to 1 to enable digital injection.

Acknowledgement

Completing a Master degree is truly a challenge. It would not be possible without the aid and support of countless people. It is with immense gratitude that I acknowledge the support and help of my supervisor, Assoc. Prof. Kazunori Hanagaki. I wish to thank him for giving me a chance to study under him at the first place. His persistent guidance and advice not only on research matters, but also on the attitude one should possess as a physicist definitely has enlighten me. Apart from this, I could not thank him enough for his trust and various opportunities that he has given me for joining various conferences and physics schools alike.

My sincere thanks also goes to Dr. Yosuke Takubo, for he has guided me patiently from the very beginning of this project until the end. His insightful ideas have helped me to solve many problems regarding the readout system design especially those related to the firmware. Besides, I also share the credit of my work with Dr. Yoshinobu Unno and Dr. Yoichi Ikegami from KEK.

I consider it an honor to work with Dr. Maurice Garcia-Sciveres of LBNL. His expertise in FE-I4 IC has helped me in solving several long-standing problems which would otherwise prohibited the success in correctly reading out FE-I4. In addition, I must thank him for giving me a chance to visit LBNL where I have had rewarding experience not only in research works but also in new research practice and lifestyle.

My sincere thanks also goes to Professor Yamanaka Taku for his care and guidance. I thank my fellow labmates in ATLAS Osaka Group: Hirose Minoru, Wataru Okamura, Masaki Endo, Higashino Satoshi, Tsuji Ryoji and other Taku Yamanaka Lab members as well as Assoc. Prof. Osamu Jinnouchi and Tomonori Kubota of Tokyo Institute of Technology for the stimulating discussions and supports. Last but not the least, I would like to thank my parents for their support and for their forgiveness and understanding for not being able to go home so often.

References

[1] ATLAS Collaboration, ATLAS Detector and Physics Performance Technical Design Report I, ATLAS TDR 14, CERN/LHCC 99-14 (1999).

[2] ATLAS Collaboration, ATLAS Detector and Physics Performance Technical Design Report II, ATLAS TDR 15, CERN/LHCC 99-15 (1999).

[3] ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider, JINST 3 S08003 (2008).

[4] G.Aad et al., ATLALS pixel detector electronics and sensors, JINST 3 P07007 (2008).

[5] ATLAS Collaboration, Inner Detector Technical Design Report Vol. I, CERN/LHCC 97-16 (1997).

[6] ATLAS Collaboration, Inner Detector Technical Design Report Vol. II, CERN/LHCC 97- 17 (1997).

[7] W. R. Leo, Techniques for Nuclear and Particle Physics Experiments, Germany, Springer-Verlag, 1987.

[8] T.Bergauer, Lectures on Silicon Detector in High Energy Physics, IPM 1st Detector School, Teheran.

[9] G. Lutz, Semiconductor Radiation Detectors, New York, Springer, 2007.

[10] H.Spieler, Semiconductor Detector System, New York, Oxford University Press, 2005.

[11] N. Wermes, Pixel detectors for particle physics and imaging applications, Nucl. Instr. and Meth. A 512 (2003), 277-288.

[12] H. Fabian, The ATLAS Pixel Detector, arXiv:physics/0412138v2 [physics.ins-det] (2005).

[13] I. Peric et al., The FEI3 readout chip for the ATLAS pixel detector, Nucl. Instrum. Meth. A 565 (2006) 178.

[14] P. Vankov, ATLAS Upgrade for the HL-LHC: meeting the challenges of a five-fold increase in collision rate, arXiv:1201.5469v1 [physics.ins-det].

[15] O.S. Bruning, HL-LHC parameter space and scenarios, Proceeding of the Chamonix 2012 Workshop on LHC Performance, Chamonix, France, 2012, pp.315-324.

[16] P. Allport et al., Progress with the single-sided module prototypes for the ATLAS tracker upgrade stave, Nucl. Instrum. Meth. A 636 (2011) 90.

[17] M. Garcia-Sciveres, et al., The FE-I4 Pixel Readout Integrated Circuit, Nucl. Instr. and Meth. A 636 (2011) 155-159.

[18] M. Garcia-Sciveres, et al., Toward Third Generation Pixel Readout Chips, Proceeding of the International Workshop on Semiconductor Pixel Detectors for Particles and Imaging, Inawashiro, Japan, 2012.

[19] M. Barbero, et al., A New ATLAS Pixel Front-End IC For Upgraded LHC Luminosity, Nucl. Instr. and Meth. A 604 (2009) 397.

[20] D. Arutinov et al., Digital Architecture and Interface of the new ATLAS Pixel Front-End IC for Upgraded LHC Luminosity, IEEE Trans. Nucl. Sci. 56, 2 (2009)

[21] M. Karagounis et al., Development of the ATLAS FE-I4 pixel readout IC for b-layer upgrade and Super-LHC, Proceedings of the Topical Workshop on Electronics for Particle Physics 2008, in Naxos 2008, Electronics for particle physics, CERN-2008-008: 70-75 (2008)

[22] Albert X. Widmer and Peter A. Franaszek. A dc-balanced, partitioned-block, 8b/10b transmission 2245 code. IBM Journal of Research and Development, 27(5):440-451, 1983.

[23] National Semiconductor Corporation, DS90CP22 2X2 800 Mbps LVDS Cross-point Switch Data Sheet, February 2006.

[24] The USBpix test system homepage: http://icwiki.physik.uni-bonn.de/twiki/bin/view/Systems/UsbPix

[25] The RCE pixel test system homepage: https://twiki.cern.ch/twiki/bin/viewauth/Atlas/RCEPixelLab

[26] SOIPIX group homepage: http://rd.kek.jp/project/soi/.

[27] T. Uchida and Y.Arai, SEABAS User Manual Rev. 02, 2008.

[28] T. Uchida and M. Tanaka, Development of a TCP/IP Processing Hardware, IEEE Nucl. Sci. Symposium, NS33-6, pp1411-1414 (2006).

[29] T.Uchida, Hardware-Based TCP Processor for Gigabit Ethernet, IEEE Trans. Nucl. Sci., Vol 55, No.3, 2008.6, pp. 1631-1637.

[30] FE-I4 Collaboration, The FE-I4A Integrated Circuit Guide, Version 11.6 (2011).

[31] FE-I4 Collaboration, The FE-I4B Integrated Circuit Guide, Version 2.3 (2012).