ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

BERKELEY LAB

Self-seeded Trigger

Tracking with self-seeded Trigger for High Luminosity LHC

Master Thesis

Section of Electrical and Electronical Engineering

École Polytechnique Fédérale de Lausanne

Student	Niklaus Lehmann	n.lehmann@bluemail.ch
Supervisor	Carl H. Haber	chhaber@lbl.gov
Mentor	René Beuchat	rene.beuchat@epfl.ch
Proposed by	Lawrence Berkeley National Laboratory, University of California Berkeley, USA	
Handed in	15. August 2014	
Executed at	Lawrence Berkeley National Laboratory	

August 15, 2014

This page is intentionally left blank.

Abstract

The Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) is the largest particle accelerator built so far. An upgrade to the so called high luminosity LHC (HL-LHC) is in development to increase the energy and luminosity of the particle beam, with the goal to perform preciser measurements on known particles as well as investigate into new physics.
For the HL-LHC upgrade of the ATLAS experiment a new generation of binary readout chips, called ABC130, has been developed. This chip was designed by the ATLAS group and includes a new method of fast readout, called fast cluster finder (FCF).
To handle the amount of data generated by the ATLAS experiment, trigger signals are required. These indicate an interesting event for further analysis.

This thesis consists of a complete detector development. This includes developing the logic for reading the detector and building the particle detector. Further, the built detector was tested in a electron test beam and different characterisation measurements were performed. The analysis of the test beam data concluded the work.

In this thesis, a demonstrator to prove the principle of a self-seeded trigger was developed. An existing concept of a correlation logic was completed and implemented in a FPGA. The final design is able to match hits from two sensor modules mounted back to back. The correlation logic can find tracks within a few beam crossing (BC) clock cycles and no external trigger signal is required. The output from this logic could be used to generate a trigger signal for a complete readout of the ATLAS detector.
A LabVIEW application was developed to implement a readout system. This system allows to acquire data collected by the correlator design. It implements also different functions to test the correlation logic and to characterize the system.

For the first time, the ABC130 was tested with a beam. The beam test was also the first experiment to give a working example of a self seeded trigger with the fast cluster finder. Characterization measurements were performed besides the test beam experiment. These measurements allow to verify the performance of the built detector.
The data acquired during the test beam measurements were analyzed to get a good understanding of the built detector. Some of the observed behaviour could be explained with the use of simulation.

This work included the study of silicon strip sensor and to become familiar with the elements of the ATLAS experiment. It included further the work with an FPGA and required the analysis of the internal timing structures. Test applications were developed with LabVIEW and the analysis of data was performed with the ROOT framework.

Résumé

Le Grand collisionneur de hadrons (LHC : Large Hadron Collider) est le plus grand accélérateur de particule construit à ce jour dans le monde. Il est géré par l'European Organization for Nuclear Research (CERN) à Genève. Pour obtenir des mesures plus précises sur des phénomènes connus et effectuer des rechercher dans de nouveaux domaines de la physique, une mise à jour, nommée high luminosity LHC (HL-LHC), est en développement. L'énergie et l'intensité du faisceau de particules seraient augmentées dans le cadre de cette mise à jour.

Pour la mise à jour de l'expérience ATLAS dans le cadre du HL-LHC, une nouvelle génération de circuit intégré appelé ABC130 est développée. Elle permet d'accéder au détecteur de silicium par une nouvelle méthode de lecture rapide, le « fast cluster finder (FCF) ».

Pour être capable de traiter les grandes quantités de données générées par ATLAS, il faut des signaux de « trigger », qui indiquent un évènement intéressant et ainsi de le sélectionner pour une analyse plus profonde. Cette thèse se compose d'un développement complet du détecteur. Elle comprend l'élaboration de la logique pour lire le détecteur et la construction du détecteur de particules. En outre, le détecteur intégré a été testé dans un essai de détection de faisceaux d'électrons et différentes mesures de caractérisation ont été effectuées. L'analyse des données de faisceaux de test conclut le travail. Un démonstrateur pour valider le principe d'un signal de trigger auto-généré a été développé pendent cette thèse qui permet de prouver le principe d'un déclencheur à auto-sélection. Cette réalisation a été achevée en complétant un concept existant d'une logique de corrélation et a été mis en œuvre dans un FPGA. La réalisation finale du système est capable de corréler des impacts de particules de deux capteurs montés dos à dos. La logique arrive à détecter des traces de particules dans un délai de quelques périodes de l'horloge principale du système sans aucun signal de déclenchement externe. La sortie de cette logique peut être utilisée pour générer un signal de déclenchement pour une lecture complète du détecteur ATLAS.

Une application LabVIEW a été développée pour mettre en œuvre un système de lecture. Ce système permet d'acquérir les données collectées par le corrélateur. Il met également en œuvre des fonctions différentes pour tester la logique de corrélation et pour caractériser le système. Pour la première fois, l'ABC130 a été testé dans un faisceau. Le test de faisceau a également été la première expérience pour donner un exemple de travail d'un déclenchement à auto-sélection avec le module de recherche rapide (FCF).

Des mesures de caractérisation ont été effectuées en dehors de l'expérience du faisceau d'essai. Ces mesures permettent de caractérisé le détecteur construit.

Les données acquises lors des mesures de faisceau d'essai ont été analysées pour obtenir une bonne compréhension du détecteur intégré. Une partie du comportement observé peut être expliqué à l'aide de simulations. Ce travail comprenait l'étude du capteur de bande de silicium et la familiarisation avec les éléments de l'expérience ATLAS. Il comprend en outre le travail avec un FPGA et l'analyse des structures internes de cadencement. Les applications de tests ont été développées avec LabVIEW et l'analyse des données a été réalisée avec l'outil ROOT.

Acknowledgements

Many people were involved in this project. I'd like to thank all of them however small the contribution was.

A special thank goes to Carl H. Haber, my supervisor at LBNL. He proposed the project and supported me during the whole time with important input for the development, as well as the physics background. I'd like also to thank Haichen Wang, post doctoral researcher at Lawrence Berkeley National Laboratory (LBNL). He helped me to understand better the physical motivation of the experiment and was a great support during the analysis of the data.

Further I'd like the Ecole Polytéchnique Fédérale de Lausanne (EPFL) for the support of this project. I'd specially thank René Beuchat, my mentor at EPFL, who followed my progress from Lausanne during the entire project. His input was always well appreciated and helpful.

My thanks go also to the test beam team at SLAC and in the rest of the world. Thanks to them it was possible to prove the self-seeded trigger system. I'd like to thank Sergio Diez-Cornell, now at DESY but former post-doc at LBNL, who helped to design the demonstrator and made a big effort organizing the test beam.

A great thanks goes to my parents, Beat and Therese Lehmann, who supported me during all my studies.
Finally I'd like to thank Jonas Stalder, with whom it was a pleasure to work during the summer, and all the other people I met here in Berkeley.

This page is intentionally left blank.

Contents

1 Project definition and goals

1.1 Motivation

The Large Hadron Collider (LHC) has been running for three years and was shut down in February 2013 for maintenance and to install a first upgrade[1]. During this first run major advances in physics were made, including the discovery of a new particle, the Higgs boson.

The LHC accelerates protons to nearly the speed of light. By colliding two beams in opposite directions, conditions similar to the time at the beginning of the universe are created. In these conditions new particles are formed which are detected by different experiments like the ATLAS.

The planned upgrade to the high luminosity LHC (HL-LHC) will increase the luminosity to obtain more precise measurements of the Higgs boson and determine it's nature, as well as to look for new physics. This also requires an upgrade of the ATLAS experiment to handle the increased luminosity. This includes the development of a new inner detector [2].

The trigger system to decide which events might be interesting for analysis, undergoes a redesign. It is planned to create a track trigger, which is able to detect tracks of particles with moderate transverse momentum (P_T) directly with data from the inner detector. Different possible approaches are researched. For the self-seeded trigger, one possible method, a fast cluster finder (FCF) was introduced in the new generation of readout chips for silicon strip sensors, called ABC130. The ABC130 chip was designed by the ATLAS group and a first batch was fabricated end of 2013.

In previous works ([3], [4] and [5]) a test system for the ABC130 was created, together with a concept for a correlation logic for a double-sided sensor module (doublet). The goal is to demonstrate the possibility to find interesting tracks of particles in the doublet.

1.2 Goals

The self-seeded trigger has the goal to find interesting tracks in the new inner tracker (ITK) of the ATLAS detector. The ITK is built with several layers of silicon detectors, which are capable to capture hits of particles. A specific logic will read two layers of sensors and match them together to look for tracks.

The content of this thesis can be separated in two main parts:

1. The first part continues the work of Lorenzo Pirrami [3]. The work of Mr. Pirrami included the development of a ABC130 test environment. Unfortunately the environment couldn't be completely tested, because of problems with the communication to the ABC130 chip. The regular readout is not possible because of a design error of the ABC130 chip.

It is necessary to complete the test system. This requires an analysis of the existing design and corrections if required. Very important is to establish the readout of the FCF of the ABC130. This new method of reading the chip has to be proven to work in this thesis.

The characteristics of the new ABC130 chip are not well known yet. Pioneering work will therefore be done in regard with how to use this chip.

Mr. Pirrami developed a concept for a correlator logic. This logic has to be completed and tested. A good understanding of the correlation principles has to be acquired. This includes also how the triggering is realized in ATLAS. To test the correlation logic, a demonstrator has to be built.

2. The second part consists of the development of a higher level correlator (track correlator) to track particles across two doublets. The ultimate goal would be to build a telescope out of two doublets.

   During this work a design should be proposed how the data from two doublets could be handled.

The complete correlator design would be implemented in a new ASIC, if the approach of the self-seeded triggering is selected for the HL-LHC upgrade. Design inputs for this chip could also be part of this thesis.

1.3 Planning

In the first weeks of the project it was planned to work mostly on part 1 of the work. This helped to get familiar with the system and the used technologies. In parallel research was done on the state of the art for part 2. With the progress of the work, the focus should have been shifted towards part 2.

The original schedule had to be abandoned after some weeks, because the correlator needed more development than originally defined. During the work it was also decided to test the correlator logic in a test beam. Additional logic had to be developed to have the possibility to integrate the correlator demonstrator with the test beam environment.

Finally the work performed was a deep study of the self-seeded trigger principle with the correlation logic. Therefore the second part was only looked briefly into. The original schedule can be found in the appendix A.

2 Introduction

In this chapter the context of the work is introduced. This includes a brief explanation about the LHC and the planned update for the HL-LHC. More details are given for the ATLAS detector, with a focus on the inner detector. The current ATLAS detector is explained in general, where the introduction of the update is focused on strips of the ITK.

Figure 2.1 gives an overview of the LHC with a zoom to the inner detector and the module, which is part of the new detector. To mention is that the computer graphics are for the currently installed ATLAS detector. The image shown for the silicon detector is from the assembly of the silicon strip detector and shows just one barrel as example. The stave is a prototype assembled at Rutherford Appelton Laboratories (RAL) and the module is a prototype of assembled at Lawrence Berkeley National Laboratory (LBNL). Both module and stave were built with ABC250, the predecessor of the ABC130 chip, used in this work.

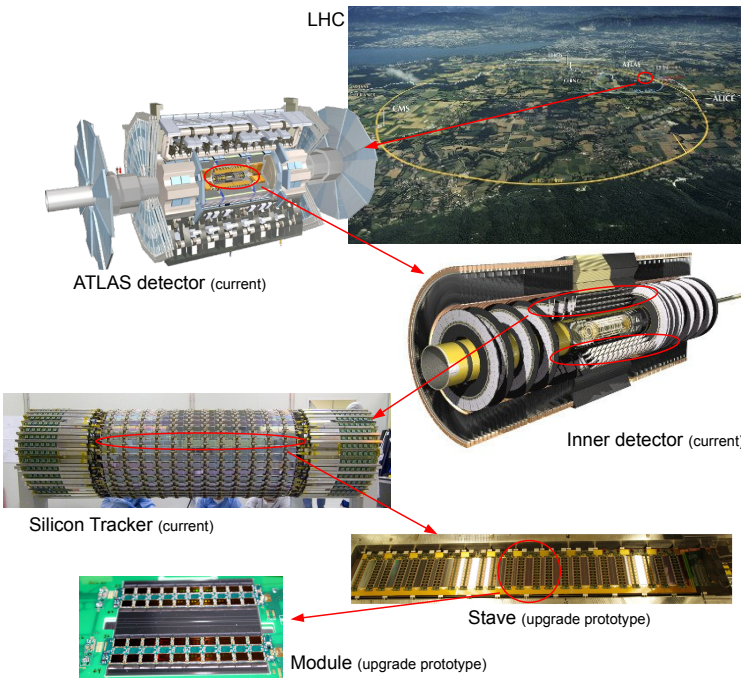


Figure 2.1: An overview of the project environment. Pictures of LHC, ATLAS, Inner detector and Silicon Tracker © CERN. Stave picture from [6]

2.1 CERN and the Large Hadron Collider

The European Organization for Nuclear Research (CERN) is located at the Swiss-French boarder and hosts multiple experiments and accelerators. CERN groups together scientist and engineers from all over the world with the goal to achieve a more profound knowledge about the fundamental structure of the universe.

The LHC is the largest an most powerful particle accelerator built so far[7]. This particle accelerator is built in an underground tunnel with a circumference of 27 kilometers. Two particle beams are guided along the tunnel with superconducting magnets at nearly the speed of light. The beams are colliding at four sites, which are the location of the four particle detectors of the LHC: ALICE, ATLAS, CMS and LHCb.

ALICE looks how the universe was just after the big bang, by looking for quark-gloun plasma. This is a state of matter supposed to have formed just after the big bang.

ATLAS is a general purpose detector and is described more in detail in section 2.2.

CMS is also a general purpose detector with the same scientific goals as the ATLAS experiment. It uses different technical solutions than ATLAS.

LHCb has the goal to find more about the differences between matter and antimatter.

2.2 The ATLAS experiment

This section introduces the currently installed ATLAS detector in general. A computer generated view of the complete ATLAS detector is shown in figure 2.2. Section 2.5 will describe the planned upgrade of the HL-LHC focusing on the relevant points for this thesis.

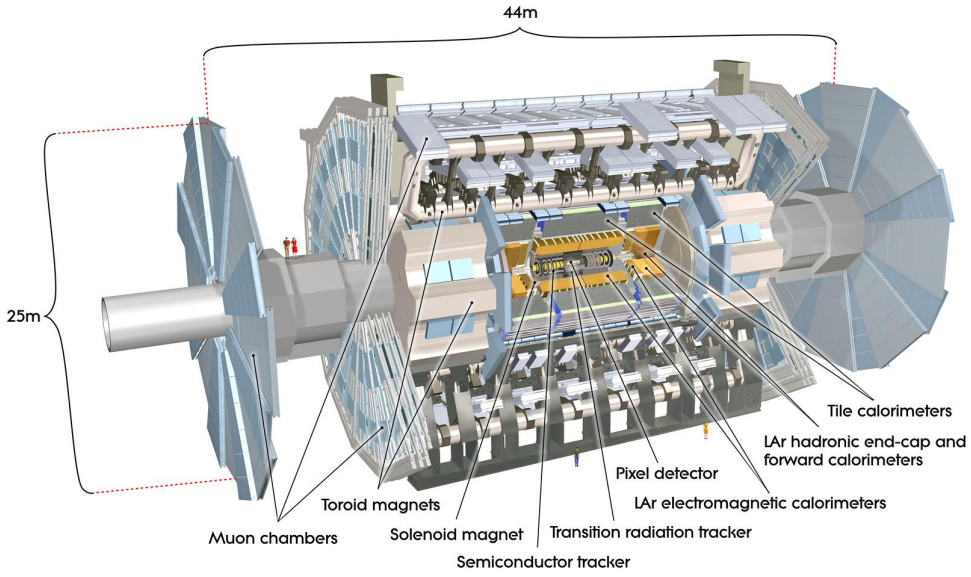


Figure 2.2: A detailed computer generated view of the ATLAS detector (ATLAS Experiment © 2011 CERN)

With a length of 46 meters, a width and height of 25m and a total weight of 7000 tons is ATLAS the largest particle detector. ATLAS is located close to the village of Meyrin in Switzerland (see map in figure 2.1). It is a general purpose detector that investigates in a wide range of physics. The results captured in the ATLAS experiment during the first run of the LHC were used to discover the Higgs boson. But ATLAS looks also for physics beyond the Standard Model of particle physics.

This detector is built around the collision point and has several layers of different detector technologies. The following sections describe the different detector parts

ATLAS is a collaboration of 174 institutes from 38 different countries. More than 3000 people are working together for this detector [8].

2.2.1 Inner Detector

The inner detector (ID) is located at the most inner point of the ATLAS and surrounds directly the collision point. The currently installed ID consists of three parts which combine high resolution detectors at the smallest radii and continuous tracking elements at larger radii. Figure 2.3 shows a schematic representation of the different elements in the ID. The complete inner detector can be separated into two regions. The barrel region has detectors arranged in concentric cylinders around the beam axis and is located around the beam crossing point. At each end are the end-caps, which hold detectors mounted on discs perpendicular to the beam axis. The ID is completely contained in the central solenoid which creates a magnetic field of 2T [9].

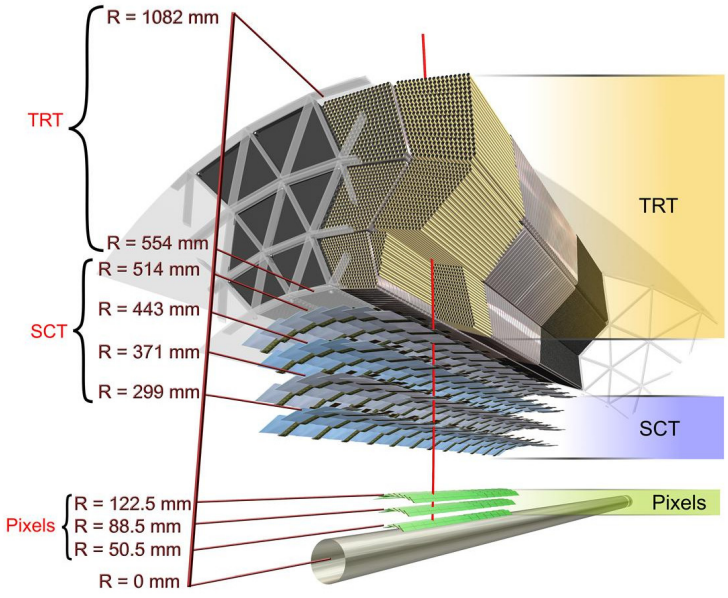


Figure 2.3: Scheme of the inner detector and its different elements (ATLAS Experiment © 2008 CERN)

Pixel Detector

The intention of the pixel detector is to track charged particles. The pixel detector is built to provide precision measurements over three different layers of barrels. It assembles semiconductor detector chips with a total amount of 80 million pixels who cover an area of 1.7m^2.

Semiconductor Tracker Detector

The semiconductor tracker (SCT) is located outside of the pixel detector at a distance of approximately 30cm to the beam. The SCT provides precision tracks in the intermediate radial length. The measurements from the SCT contribute to the measruement of the particle momentum, impact parameter and vertex position.

Transition Radiation Tracker

The most outer part of the ID is the transition radiation tracker (TRT) which consists of straw detectors. They can operate at high rates and allow the detector to discriminate between tracking hits passing the lower threshold and transition radiation which passes the higher threshold. Each straw has a diameter of about 4mm and is equipped with a wire in the center. The straws are filled with a non-flammable gas mixture, which interacts with the particles passing through.

2.2.2 Calorimeter

The calorimeter measures the energy of charged and neutral particles, except muons and neutrinos. This is done by the means of absorbers and sensing elements. The particles interact with the absorber and the results of this interactions is detected by the sensing elements. In the ATLAS detector are two systems in use: liquid argon in the inner sections of the calorimeter and scintillating plastic in the outer sections [9].

2.2.3 Muon Spectrometer

Muons are charged particles, similar like electrons, but with a mass about 200 times larger [9]. They are the only particles able to pass the absorbers of the calorimeter besides the neutrinos .

The muon spectrometer is the outer most part of the ATLAS. It measures the path of muons and allows a precise measurement of their momenta. The sensors used are similar to the straw detectors used in the TRT (see 2.2.1) but with larger tube diameters.

2.2.4 Magnet System

A supraconducting magnet system is integrated in the ATLAS and induces a magnetic field through the whole detector. The magnetic field bends the trajectory of charged particles and thus allows to determine their momenta.

The magnet system is separated in two part. A solenoid magnet around the ID with a magnetic field strength of 2 Tesla, and a toroidal magnet systems in the outer layer of the detector with a field of 4T.

2.3 Trigger system

In general a trigger signal indicates when a something has to be stored or registered. In case of particle physics experiments, the trigger signal is used to indicate the readout electronics when an event happened, i.e. when an actual particle crossed the detector. The trigger signal might be generated from other detectors or directly from the signal measured by the readout electronics.

2.3.1 Motivation for a trigger system in ATLAS

There are around 1 billion (10^9) proton-proton collision per second in the detector. It would need 100'000 CDs per second to record all raw data [8, p.7]. This is not possible and this amount of data can't be processed in real time. Most of the events are not interesting for further analysis, because already well known from preceding experiments. Therefore a trigger system is needed to select the events which are interesting for further analysis.

2.3.2 Current ATLAS trigger system

The current trigger system works in three levels. Figure 2.4 shows the concept of the implemented trigger stages.

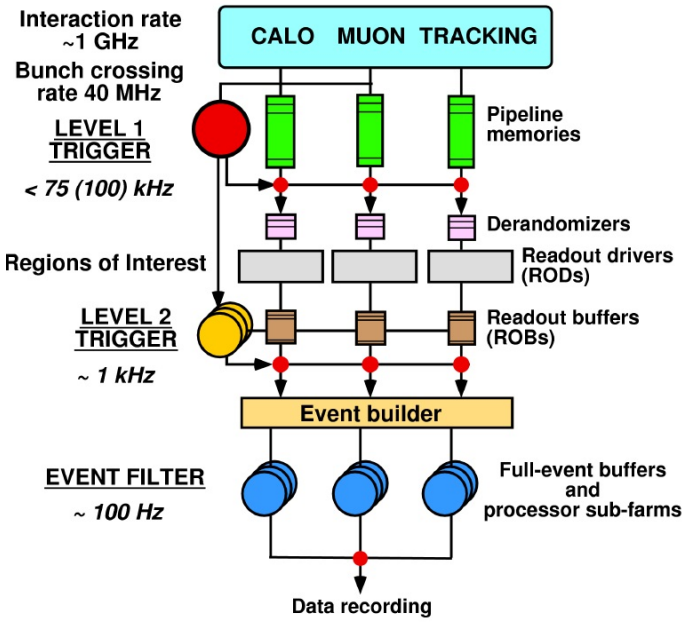


Figure 2.4: Schematic of the current ATLAS trigger system [10, Figure 1.9]

Level 1 (L1) is done in the detector and uses special purpose logic. It reaches its decision after 2μs and reduces the events to approximately 75'000 events per second. To reach a decision, the L1 trigger uses the data coming from the calorimeter and the muon spectrometer.

Level 2 (L2) is a trigger implemented in software and runs on large computer farms. These farms are built with commercial computer and network devices. The reach the decision, the L2 trigger uses the information from the complete detector including the ID.

The data selected by L2 are processed in an event filter, which is also run on a computer farm. Approximately 200 events per second are selected to interesting and are stored for offline analysis [8].

2.4 Silicon strip sensors

Silicon strip sensors are used in the SCT of the current ATLAS experiment. They will also be used in the ITK of the upgraded ID. The working principle of the strip sensors are explained here to give a basic understanding. How the sensors are integrated is described in section 2.5.3.

2.4.1 Strip detector

The basic idea of a silicon sensor is to use pn-junctions. It can also be seen as a diode. The junction is biased in reverse to completely empty the junction of charges. If a particle passes through the silicon, it looses it's energy due to bremsstrahlung, ionization and others. The ionization loss creates electron-hole pairs in the pn-junction. These charges drift under the influence of the electric field and are seen as a current which can be measured. Figure 2.5 shows a schematic view how such a sensor is built.

The term sensor usually references to the complete array of strips. One strip is a single diode. Figure 2.5 shows three such strips as a side view.

The base for the sensors used for the ATLAS strip detector is a p-type silicon substrate with a p^+-backplane. On the surface opposite the backplane, n^+-doped strips are implanted. This creates an array of diodes in the sensor. Each implanted strip is connected with a resistor to the bias voltage. The strips are covered by an oxide and by metal strips. The

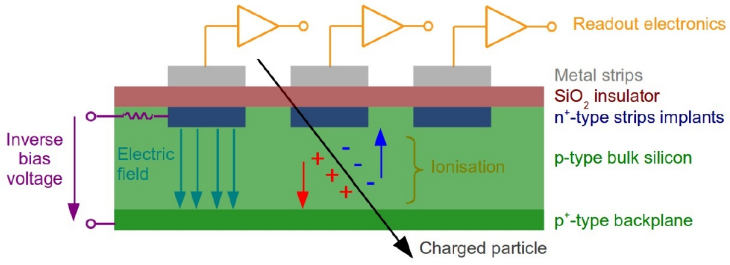


Figure 2.5: Schematic view of a silicon strip sensor looking from the end of the sensor [11, Figure 7]. Three strips are shown.

n^+ and metal strips from a capacitor called C_{ox}. The metal strips are connected to the readout electronics via wirebonds and the signal is read by the front end electronics (FEE) through AC coupling. The electronics is explained in section 2.4.3.

A sensor can be built with different geometries. The geometry is usually indicated by the name of the sensor. A pixel sensor, the basic sensing elements are square or rectangular shaped elements and a two dimensional array is fabricated. A strip sensor is based on long sensing elements with a fixed width and are assembled in rows. The length is by magnitudes larger than the width in case of strips.

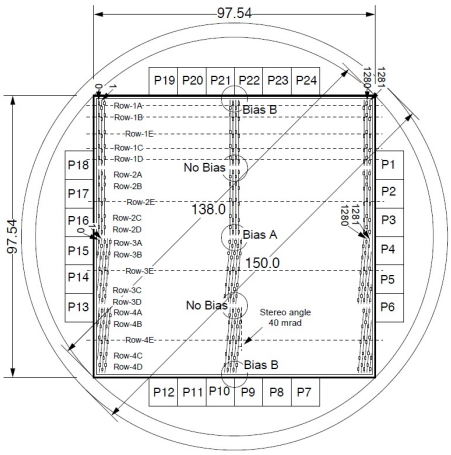
In general such a silicon sensor can be seen as a digital camera, which takes pictures of particles passing through. The resolution of the sensor is defined by the number of sensing elements assembled together.

2.4.2 ATLAS07 sensor

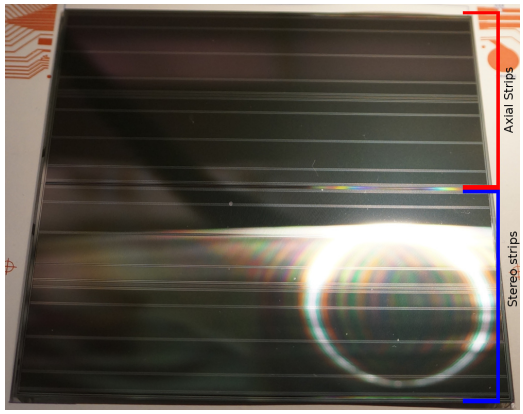
The sensor used in this thesis is a ATLAS07 strips sensor and is shown in figure 2.6. It has four rows of 1280 strips each, with a pitch of 74.5µm. The rows are each 23.8 mm long. The sensor is 300µm thick.

The two stereo rows are rotated by 40 mrad in respect with the edge of the sensor. This gives the possibility to have a 2 dimensional readout if two layers are used together. The axial two rows are parallel with the edge. For the self-seeded trigger demonstrator only the axial rows were used.

For a good collection of the signal it is important that inverse bias voltage is large enough to completely empty the pn-junctions. In case for the ATLAS07 sensor a inverse bias voltage



(a) Design specification of the ATLAS07 sensor. Each row of strips (1-4) has five rows of bonding pads (A-E)



(b) Picture of one sensor used in this work

Figure 2.6: ATLAS07, a n-on-p silicon strip sensor. Two rows have an angle to the edge of the sensor (stereo strips) and two rows are parallel (axial strips). Source for design specification: [12, Fig. 1]

of 250V has to be applied. According to [13, 32.7.1] the most probable charge deposition in a 300µm silicon sensor is 3.5fC (22000 electrons). This charge corresponds to the expected signal amplitude.

2.4.3 Readout electronics

Figure 2.7 shows a simple block diagram of an analog front end electronics.

The current generated by a passing particle is stocked in C_{ox}. This capacitor is discharged through the resistor R_{bias} at the end of the strips. Therefore, the readout electronic has to collect the charge within a given time, defined by the RC-element $R_{bias} \cdot C_{ox}$.

The charge read is amplified and integrated by a preamplifier. The signal pulse is then optimized by a pulse shaping circuit before it is digitized.

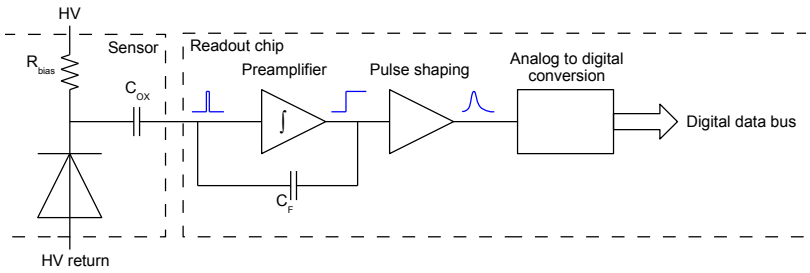


Figure 2.7: Basic blocks of a analog front end electronics for the readout of a silicon sensor. C_F defines the final gain of the amplifier stage. [14, Figure 1.2] [15, Figure 10]. Such an electronic is implemented for each channel (strip) of the sensor.

There are several different methods how the data are converted and used.

- **Analog readout**: The analog signal from the amplifier or after the pulse shaping circuit could be used directly. All the information from the signal is available with an analog readout, but requires a complex analyzing circuit.
- **Time over threshold**: The value return by the electronics is the time which the pulse from the sensor is above a set threshold. The time gives an indication of the amplitude of the impulse.
- **Multi bit ADC**: If an analog to digital converter with a a given number of bits is used, the amplitude of the signal is conserved within the resolution of the ADC. This requires a very linear amplifier.
- **Binary readout**: A simple 1-bit ADC is used, which detects only if there is a hit or not. For such a system a threshold has to be set to define the level the input signal has to reach for a detection as a hit. The amplifier does not need to be linear above the threshold, which makes the design simpler.

In case of the ABC130 the binary readout approach was selected. Each strip is read with a 1-bit ADC, which has a programmable threshold. How the FEE is implemented in the ABC130 chip is described in section 3.2.

2.5 ATLAS for the HL-LHC

2.5.1 Intention of the upgrade

As already mentioned, it is planned to increase first the energy of the collision and later the instantaneous luminosity. Figure 2.8 shows the schedule with the different phases of the upgrade. This work is for the experiment upgrade (Phase II) in 2022.

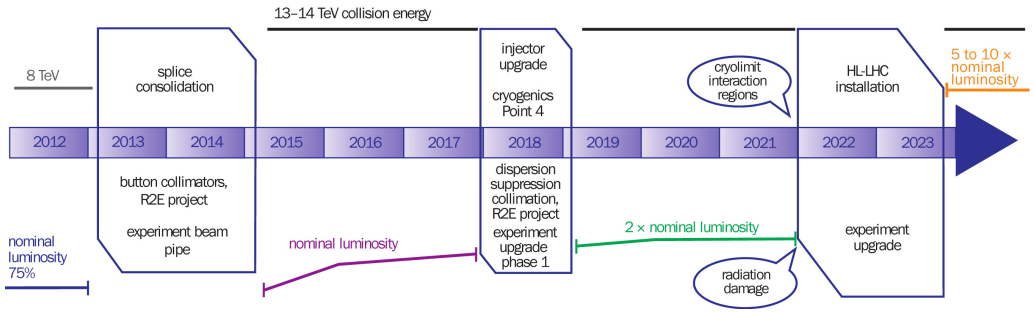


Figure 2.8: Planned schedule for HL-LHC upgrade [16]

The luminosity after the upgrade will be five times the nominal LHC luminosity [2]. It will help to look for signatures of new physics predicted by models like super symmetry or extra dimensions. The HL-LHC will also allow to measure particles from the Standard Model with a better precision.

At the time of the phase II upgrade, there will be components running in ATLAS which are more than 15 years old. Also the silicon tracking systems of the ID will reach the end of their lifetime. The ID will be completely replaced for this reason and also because the higher luminosity will increase the occupancies of the sensing elements.

The upgrade introduces also other challenges like in case of data output. ATLAS should maintain or improve its performance in triggering and reconstructing the full range of physics objects. Therefore the P_T-threshold is maintained at the same level. Maintaining this low threshold would increase the Level 1 trigger rate by at least a factor of five, which exceeds the capability of the current trigger system. A new trigger system together with an improved data acquisition system is in development for the update.

2.5.2 All Silicon Inner Tracker

The ID will be completely replaced by an all silicon tracker inner tracker (ITK). It will consist of a pixel detector closest to the beam crossing point and a silicon strip detector at the outer layer. The strip detector will have short strips in the location of the current SCT and long strips instead of the TRT. Figure 2.9 shows a conceptual layout of the ITK, where only one quadrant of the whole detector is shown. The beam crossings are located at the lower left corner of the figure.

The detectors are arranged in cylindrical barrels in the central region and as disks in the forward region. The total surface covered by the pixel detector will be 8.2m^2 with 638

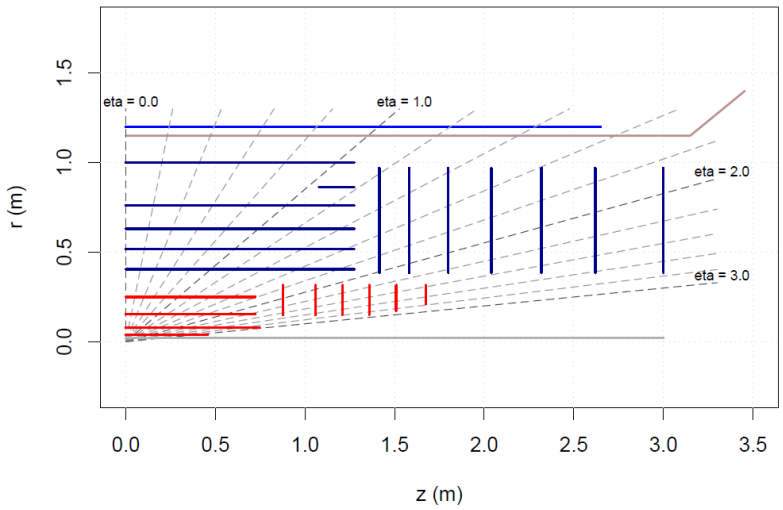


Figure 2.9: One quadrant of the layout for the new inner tracker. Pixel detector in red, arranged 4 barrel and 6 disks. Strip detector in blue with 3 short strip barrels, 2 long strip barrel layers and 7 strip disks. [2, Figure 6.1, p. 59]

million channels. The 74 million strip detector channels will cover a silicon area of 193m^2 [2, Table 6.6, p. 60]

Not only the sensing elements will be improved, but also the mechanical structure. The overall amount of material used in the tracker is reduced. This reduces the energy losses due to and reduces the multiple scatterings, which leads to a better precision.

2.5.3 Strip detector in the inner tracker

The ITK uses strip sensors as described in section 2.4.1. There are two kind of geometry used: *1.* short strips with a length of 23.82mm, on the inner three strip barrels and *2.* long strips on the outer two barrels with a length of 47.755mm. The strip sensors have either 4 rows of short strips or two of long strips. There are 1280 strips in one row with a pitch of 74.5µm.

The electrical readout of the sensor is done by the ABC130 chip (see chapter 3). Each chip has 256 input channels. 5 chips are therefore required to read one row. Practically there will be 10 chips assembled on a hybrid board which reads two rows. One or two hybrid boards are glued onto a silicon detector, depending on the length of the strips and form a module.

The strip detector of the ITK will be assembled in so called staves. Each stave will hold 26 modules, always 13 on each side of the stave. Figure 2.10 shows a sketch of a stave.

The stave has a lightweight honeycomb carbon fibre core with integrated cooling pipes. There is a copper/aluminium/kapton bus tape that integrates all the power and data connection lines to the modules on each side of the core and the modules are glued on top of it.

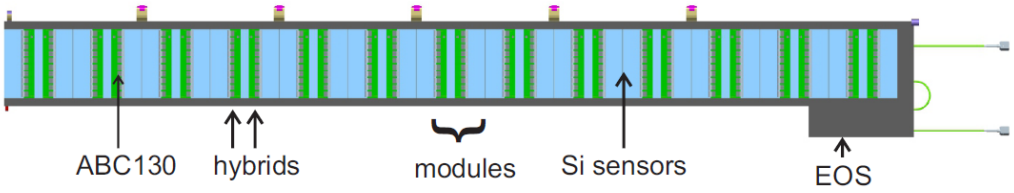


Figure 2.10: Sketch of a stave for strip modules [2, Fig. 6.28]

An End Of Stave board (end of stave board (EOS)) provides the interface to the computer farms in the means of a versatile fiber. It also distributes the power and the timing, trigger and command signals to the modules. Each hybrid has an additional chip, called Hybrid Control Chip (hybrid control chip (HCC)). The HCC makes the connection between the EOS and the ABC130 chips.

A similar structure is used for the end-cap disks. The equivalent to the stave is the petal. The readout is also done with the ABC130 chips, but the hybrids as well as the silicon sensors for the petals have a different shape. There are shorter strips on the inner radii and longer strips on the outer part of the disks.

2.5.4 Upgrade of trigger system

The requirements for the trigger system of the upgrade depend on the physics goals at the HL-LHC. The upgrade of the trigger system for phase II is driven largely by the desire to maintain a threshold of ∼20GeV for isolated electrons and muons at the higher luminosity[2, Section 2]. Including the data from the inner detector could lower the acceptance rate to a P_T of a few GeV. Further, the trigger system should be flexible enough to allow adaptions to new discoveries or background changes.

There are two main roads of a possible track trigger implementation. *1.* The Regional Readout Request (regional readout request (R3)) scheme, or *2.* addition of specific trigger logic to produce a self-seeded track trigger [17].

Regional readout request scheme

The R3 scheme works in a two level trigger system. The ABC130 has two different pipelines. The level 0 (L0) pipeline is filled with the front end data on every BC clock together with a beam crossing ID (BCID). The L0 buffer can store data up to 6.4µs. The transferred event is selected by a fixed, but programmable, latency and contains the data from 3 collisions (one before, the selected and one after).

Figure 2.11 shows a schematic view of the two level L0/L1 system. To apply this system, two possible triggers were implemented in the readout chips. The R3 trigger signal is used to readout specific regions of the detector, while a L1 trigger performs a readout of the complete detector.

A very first trigger level (L0) is generated at a maximum rate of 500kHz. This L0 would be created base on information from the calorimeters or the muon spectrometers. On each

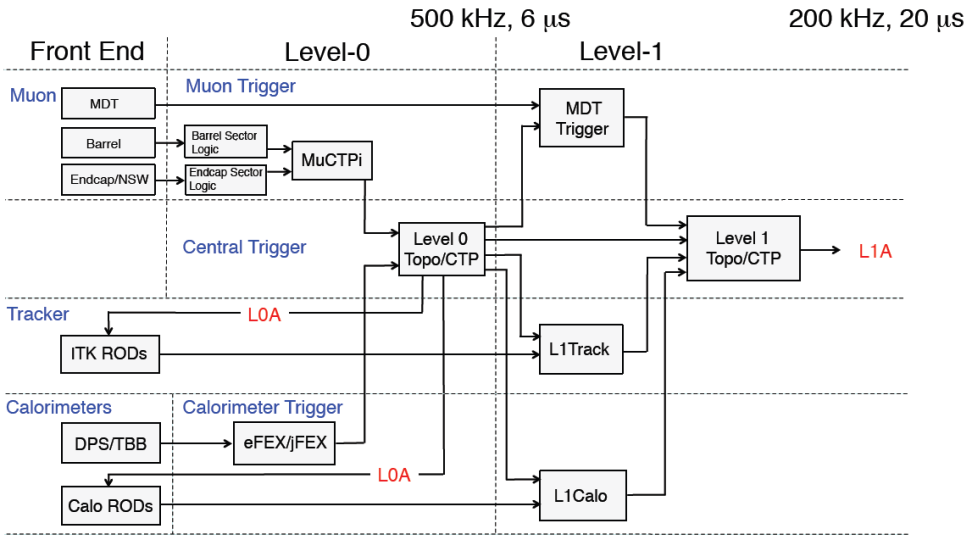


Figure 2.11: Scheme of the new L0/L1 trigger generation for the HL-LHC [2, Figure 2.2].

L0 signal an event is moved from the L0 buffer to the R3L1 pipeline and a Level 0 ID (L0ID) is added to the events in the R3L1 pipeline.

A region of interest (region of interest (RoI)) is created together with the L0. The RoI are like a cone to select only a specific part of the detector, as shown in figure 2.12. The partial readout reduces the amount of data to be read and transmitted to the L1 trigger electronics. The L1 rate is expected to be lower than 200kHz [2, Chapter 2].

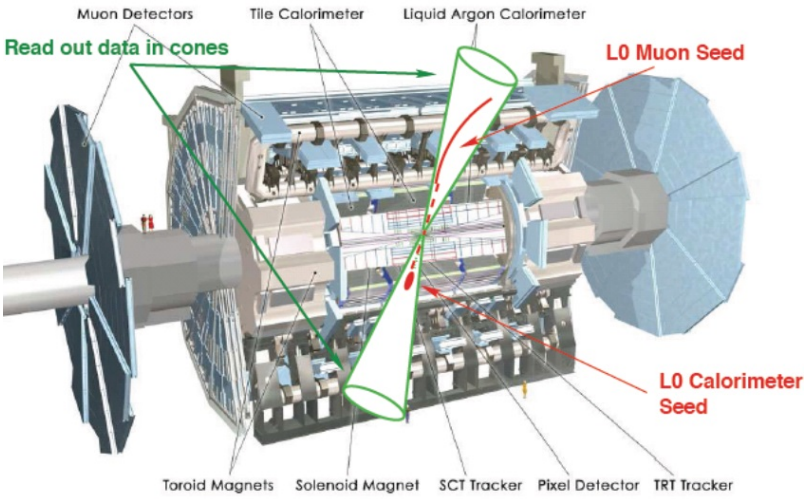


Figure 2.12: Definition of RoI from events in the calorimeter and muon detectors. [18]

Self-seeded track trigger

The self-seeded track trigger is a new concept for a trigger system, where the data from the ID is also used for the L0 and L1 trigger generation. The idea is to add logic elements, which can find tracks within the time limits of the L0 trigger. The output of this logic would be fed to the Level 0 Topo/CTP block in figure 2.11 and is used together with the muon and calorimeter data.

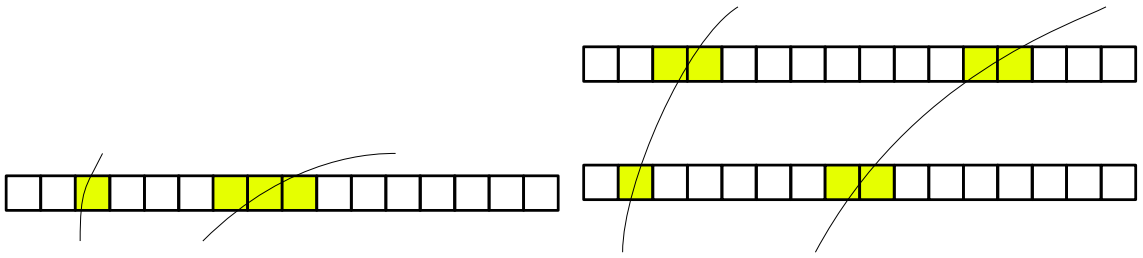
To do this, filtering algorithms based on cluster size and fast vector tracking are implemented in the front end electronics to select high P_T tracks [19]. The self-seeded track trigger is an alternative to the region of interest method. The FCF with a 640MHz readout clock gives also first information on how a fast output readout system could perform.

The two main front end filtering methods intended to be used are *1.* the Cluster size method and *2.* the offset method.

For the cluster size method logic, the readout chip looks for hits on adjacent strips. Low P_T tracks are strongly bent due to the strong magnetic field and will result in multiple hits, because they hit the sensor under a larger angle. This is illustrated in figure 2.13a. The FCF of the ABC130 implements the cluster finding and filters directly all large clusters. See section 3.4 for more details.

The offset method makes use of the double sided staves. A sketch of the principle is shown in figure 2.13b. For the track trigger the modules of both sides are aligned with each other, i.e no inclination relative to each other. Matching corresponding hits between the inner and outer layer allows to select only tracks above a certain P_T-threshold. This is done by limiting the position offset between the hits of the inner and outer layer. The intend is to add a correlator chip, who matches the clusters from the FCF outputs to find high P_T tracks. The basic logic for such a correlator was developed in [3].

An even better track analysis could be performed, when the results from different barrels of the strip detector are taken into account.



(a) Principle of the cluster method. A high momentum particle (left track) creates only a single or double hit, while a low momentum particle creates a cluster with more than two hits (right track)

(b) Principle of the offset method. A high momentum particle (left track) hits both sensors almost at the same location. Contrary a low momentum particle has a larger difference between the hit strips of the two sensors (right track)

Figure 2.13: Illustration of the offset and cluster method

3 ABC130

The newest generation of the ATLAS binary chip, called ABC130, is an ASIC containing the front-end electronics for the silicon strip sensor readout. This chip is fabricated in a 130nm technology, hence the name. It was developed to match the requirement for the HL-LHC upgrade and includes several new features.

This chapter gives a general overview of the chip and its functions. A more detailed description is given in the specification [20] and in the description of the electronic for the strip detector upgrade [17].

3.1 Overview

The ABC130 provides all functions to process the signals from 256 silicon strips, employing a binary readout architecture. Figure 3.1 shows a simplified overview of the complete chip.

The strips from the sensor are connected to an analog front end, which is described in section 3.2.

The digitized sensor data are then transferred to the buffers for the different trigger levels. The data in the buffers are then compressed depending on the requested readout scheme and serialized to be transmitted of the chip. The regular data readout through the *Data* and *Xoff* lines, together with the corresponding output logic blocks, aren't used in this

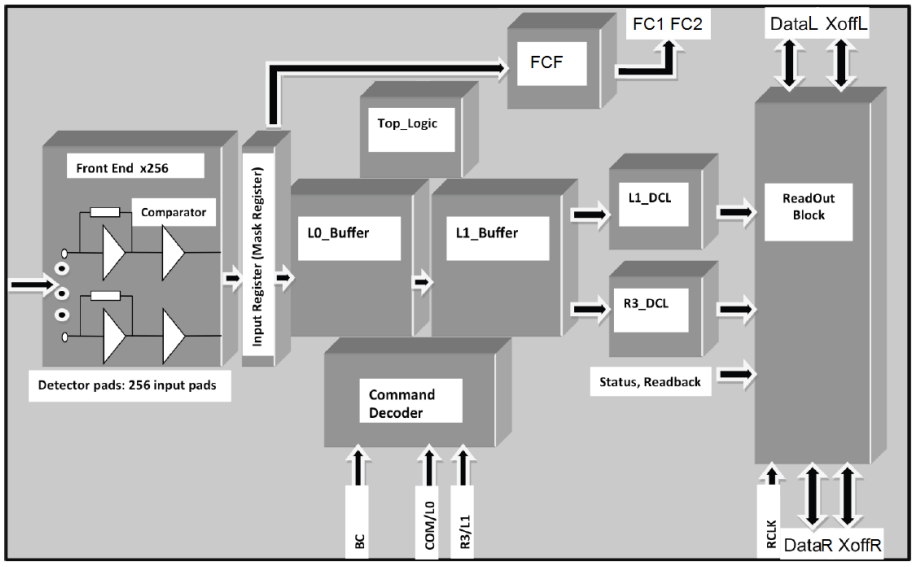


Figure 3.1: Block diagram of the ABC130 [20, Fig. 3-1]

thesis and therefore not described in detail. A design error is the cause why the readout is not working in the available chips. More about this error in section 3.5. The description of the regular readout scheme can be found in [20, section 3.1.11 to 3.1.18].

A command decoder reads and decodes the serial input lines. It is possible to send commands and trigger signal on two different lines. The commands and signals are described in section 3.3.

The FCF block is used in this work as alternative data output. Its main purpose is to find hit clusters which serve as input to the correlator logic. Section 3.4 describes the functionality of this block.

3.1.1 Pins and signals

Table 3.1 lists the available signals of the ABC130 chip. Note that the signals *COM/L0* and *R3/L1* have two signals multiplexed on the same line, i.e. two 40MHz components which form a 80MHz signal. The description in the table indicates which signal is on the falling and which on the rising edge of the BC clock signal.

Signal	**Direction**	**Type**	**Description**
In 0-255	input	analog	preamplifier inputs from strips
BC	input	sub-LVDS (SLVDS)	Beam crossing clock (40MHz)
RCLK	input	SLVDS	data readout clock (80 or 160MHz)
FastCLK	input	SLVDS	FCF readout clock (320 or 640MHz)
COM/L0	input	SLVDS	L0 trigger on falling BC edge, COM signal on rising edge
R3/L1	input	SLVDS	R3 trigger command on falling BC edge, L1 trigger command on rising edge
XOFFL	in/out	SLVDS	Chip to chip data packet flow control
XOFFR	in/out	SLVDS	Chip to chip data packet flow control
DATAL	in/out	SLVDS	Chip to chip serial data line
DATAR	in/out	SLVDS	Chip to chip serial data line
FC1	out	SLVDS	FCF output for the first fast cluster finder
FC2	out	SLVDS	FCF output for the second fast cluster finder

Table 3.1: ABC130 signals [20, table 3-1, 3-2 and 3-3]

The *XOFF* and *DATA* lines are intended for the transmission of the data packages through the chips out of the hybrid. A bit controls the direction to allow a readout in both ways. See also section 3.5.

3.1.2 Channel numbering

The ABC130 has in total 256 input channels. To make the wirebonding possible, one chip reads from two different strip rows. Figure 3.2 shows how the strip sensor is connected to the ABC130 chip.

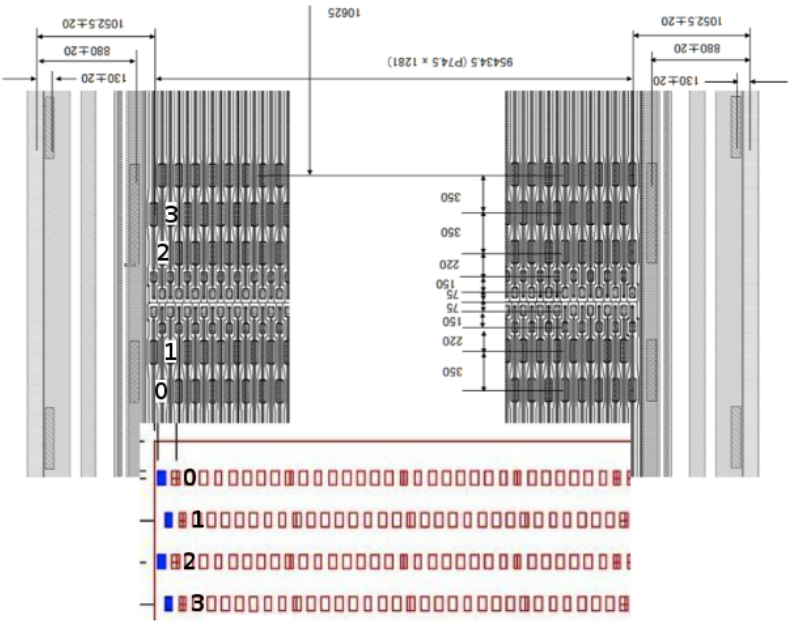


Figure 3.2: ABC130 input pads with sensor. The bonding pads of the sensor are shown in gray in the upper part of the figure. The input pads of the ABC130 are shown in red in the bottom part. A connection with a wirebond has to be made from pin 0 and strip 0 and so forth. [20, Fig. 3-10]

Because the chip reads from two different rows of strips and the interconnection is not straight forward, a quite complicated numbering is used. One has to differentiate between input channel number, register channel number and the address in the FCF output data. The table 3.2 shows how the different numberings are related together.

The rows indicated in table 3.2 do not match the row numbers in figure 2.6a. They are actually connected in inverse, i.e. row 1 in table 3.2 corresponds to row 2 in figure 2.6a and row 2 vice verso. Rows 3 and 4 from figure 2.6a are not used in this work. They would be connected to a second hybrid (see section 4.2.4) in a similar way as the first two rows.

The rows from the sensor, described in section 2.4.2, have 1280 strips. Always 128 channels are connected to one ABC130 as indicated in table 3.2. The strip number of the sensor can be calculated with

$$Strip\ number = 128 \cdot ABC130\ ID + ABC130\ FCF\ address \tag{3.1}$$

where the strip number is from 0 to 1279, the ABC130 ID from 0 to 9 and the FCF address from 0 to 127.

Sensor	Input pin	Channel number	FCF address
row 2, strip 0	0	0	FC2, address 0
row 2, strip 1	1	2	FC2, address 1
row 1, strip 0	2	1	FC1, address 0
row 1, strip 1	3	3	FC1, address 1
⋮	⋮	⋮	⋮
row 2, strip 126	252	252	FC2, address 126
row 2, strip 127	253	254	FC2, address 127
row 1, strip 126	254	253	FC1, address 126
row 1, strip 127	255	255	FC1, address 127

Table 3.2: ABC130 channel numbering. Gray rows are so called "*even*" rows, white rows are "*odd*" rows. Attention to the confusing index change between the input pin numbering (as in figure 3.2) and the channel numbers, used for the internal ABC130 register e.g. mask register.

3.2 Front end

The charge acquisition of the input channels is done in the FEE of the ABC130. There are several stages as can be seen in figure 3.3.

The current source $\mathbf{I_S}$ represents the input charge from the sensor. The charge from the sensor is integrated through the pre-amplifier and the subsequent pulse shaper adjusts the timing and form of the created pulse to meet the input requirements of the analog to digital converter.

The ABC130 uses a 1-bit ADC and stores only information if there was a hit or not. The amplitude of the original signal is lost after digitization.

Through several configurable digital to analog converters it's possible to adjust all the different blocks of the FEE. Most of the DACs are common for all channels.

BIPRE and **BIFEED** control the input transistor and the feedback current of the pre-amplifier respectively.

With **BTV**, one can adjust the threshold voltage for the comparator, whose bias current can be adjusted with **BICOM**. Each channel has an additional DAC (TrimDAC) to modify the threshold level individually. The range of the TrimDAC can be set with **BTRANGE**.

3.2.1 Calibration circuit

A calibration circuit is included for testing purposes. This circuit consists of a test capacitance $\mathbf{C_T}$, which is charged through a chopper circuit. The applied voltage is defined by the 8 bit DAC **BCAL**. The charge is injected into the channels with a common calibration

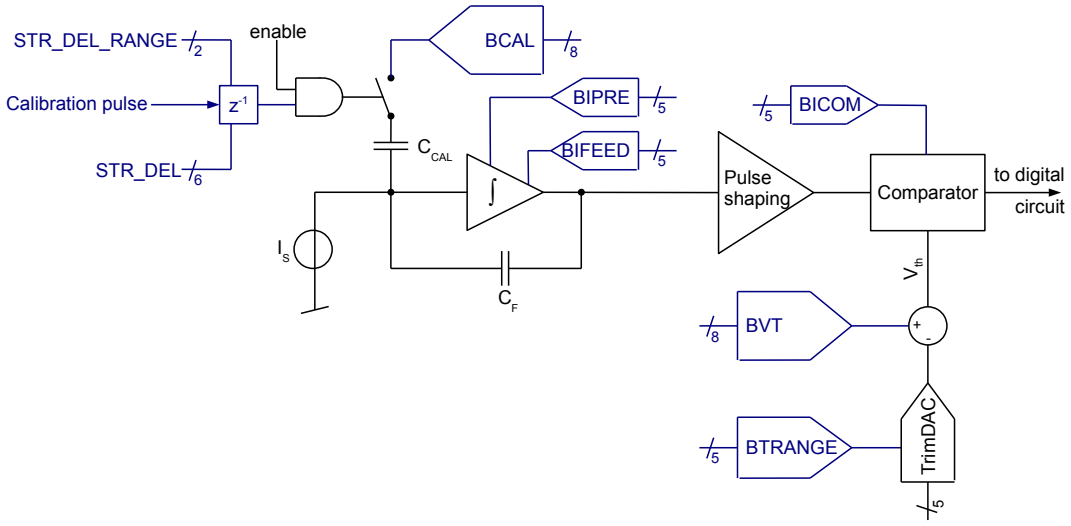


Figure 3.3: Block diagram of the analog front end electronic for one input channel. The elements drawn in blue are global signals and common for all channels. The elements in black are present for each channel.

signal after a tunable delay, called strobe delay (**STR_DEL**). For each channel an enable bit is used to activate the charge injection.

The injected charge can be calculated with the formula

$$Q = C_T \cdot \Delta V \tag{3.2}$$

where $C_T = 60fF$ and ΔV is the voltage level set by **BCAL**. ΔV can be adjusted between 0 and 150mV which creates a charge range from 0 to 9fC [1].

The injection of the charges is triggered with the calibration pulse. This pulse is generated, if a '1' is written to the **CalPulse** bit in the special register[2].

3.2.2 Analog multiplexer output

A test output pin called **AMUX** is used to measure internal voltages. The voltages are selected by writing a '1' to the corresponding bit in the analog and DCS control register. Only one voltage at a time can be activated. [20, Section 3.1.7.7] shows the possible outputs. This pin was used to verify that the ABC130 chip is responding to the commands.

3.2.3 Mask register

The binary information from the comparator is first stored in an input register. The channels in this register can be masked by setting the corresponding bits in the mask

[1] see [20, Section 3.1.7.5]

[2] **CalPulse** is bit 0. Special register address has 0x00

registers. Each channel that is masked, will be always at '0' i.e. no hit in the further stages. This mask register allows therefor to filter bad channels.

The **TM** bit in the configuration register[3] gives the possibility to use the mask register as a fixed input the L0 pipeline and FCF. This mode allows to test the compression and readout logic.

3.3 Command and trigger signals

The ABC130 has two input lines for command and triggers signals, see table 3.1. The *COM/L0* line is used to send a L0 trigger signal or a command. The *R3L1* line is used to send either a R3 or a L1 trigger signal. Examples of the different transmissions are given in section 3.3.3.

3.3.1 Commands

There are three different kind of commands. *1.* Resets, *2.* Access control and status register (ACSR) command or *3.* Special commands.

Resets

Reset commands use an 8 bit format and have a direct effect on the chips operation. The resets are used to synchronize the internal with the external counters or to return the chip in a predefined status. Table 3.3 shows the different resets and their pattern

Head				Type			Parity	Description
1	0	1	0	0	0	0	0	FCF reset
1	0	1	0	0	0	1	1	System reset
1	0	1	0	0	1	0	1	BC reset
1	0	1	0	0	1	1	0	L0ID preset
1	0	1	0	1	0	1	0	SEU register reset

- *Head*: field indicating that a reset command is sent.
- *Type*: indicates reset type
- *Parity*: set to have an even number of '1's in the command.

Table 3.3: ABC130 reset commands. The L0ID counter can be set to a user defined value. As default it is set to $FF to have the first event number 0. [20, table 3-19]

[3] **TM** is bit 16. Configuration register address has 0x20

ACSR and special commands

ACSR are commands to execute read and write operations to the internal registers. These commands have 58 bits and address a specific chip. All chips who receive the command have to decode it completely, but only the addressed executes it. This is done, so that the payload may not be interpreted as another command signal. The ABC chip ID "11111" is used as broadcast address where all chips execute the command.

The special command uses the same format as the ACSR commands with the register address set to \$00. Writing at this register address will execute an immediate specific command, like producing a calibration pulse.

Head	**Type**	**Parity**	**HCC ID**	**ABC ID**	**Register**	**R/W**	**Data**
1011	111	0	5 bits	5 bits	7 bits	1 bit	32 bits

- *Head*: field indicating that a long command is sent.
- *Type*: ABC130 long command
- *Parity*: set to have an even number of '1's in the first 8 bits of the command.
- *HCC ID*: Address of the HCC chip
- *ABC ID*: Address of the ABC130 chip
- *Register*: Register address to be accesses
- *R/W*: write '1' register or read '0'
- *Data*: data to be written to register

Table 3.4: ACSR and special commands [20, table 3-20]

The HCC chip ID field is used to send commands to the Hybrid Controller Chip (HCC) which is located at the end of a hybrid. This chip is not used during this work and therefore not further described. Commands intended for the HCC would have "110" in the type field.

3.3.2 Triggers

L0 trigger

There are two possible modes for the interpretation of the L0 signal. This is controlled with the **L0mode** bit in the control register. As default, a single *high* bit is interpreted as the L0 trigger. If **L0mode** is set to '1', then the pattern "110" is required as a L0 signal.

L1 trigger

The L1 trigger signal is a stream of 11 bits. Table 3.5 shows the pattern expected. The L0ID is a number identifying the L0 trigger event in the pipeline. The L1 trigger signal sends this number as data to request a specific value from the R3L1 buffer.

Head			L0ID							
1	1	0	x	x	x	x	x	x	x	x

Table 3.5: L1 trigger pattern. The L0ID indicates the entry read in the buffer.

R3 trigger

As the L1 trigger, the R3 trigger is formed of 11 bits. Table 3.6 shows the pattern for the R3 trigger. Again the L0ID is sent with the trigger to request a value from the R3L1 buffer.

Head			L0ID							
1	0	1	x	x	x	x	x	x	x	x

Table 3.6: R3 trigger pattern. The L0ID indicates the entry read in the buffer.

3.3.3 Examples of command and trigger transmissions

To illustrate the transmission of the commands and trigger some example are given in the figures 3.4, 3.5 and 3.6.

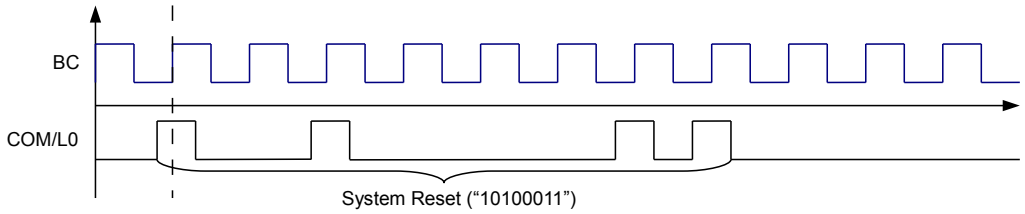


Figure 3.4: Time diagram for the transmission of a system reset command. The resets are transferred on the COM/L0 line synchronous to the rising edge of the BC clock.

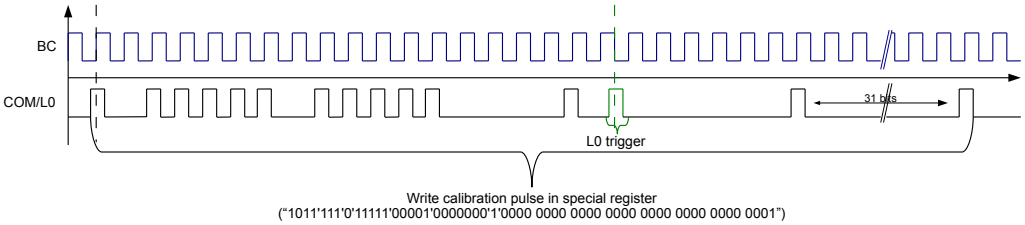


Figure 3.5: Time diagram for the transmission of a ACSR command together with a L0 trigger signal. Both signals are tran transferred on the COM/L0 line. The command, in this example write of the special register to set a calibration pulse, is synchronous to the rising edge of the BC clock. The L0 trigger is synchronous to the falling edge of the BC clock.

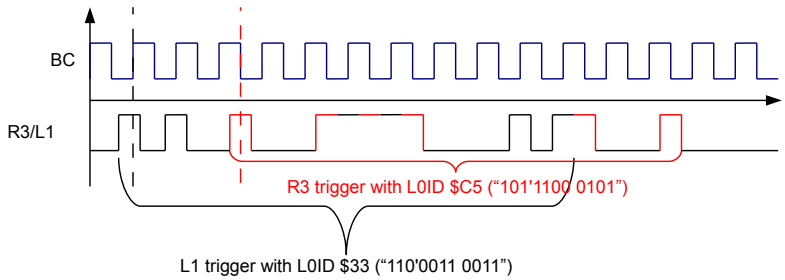


Figure 3.6: Time diagram for the transmission of a L1 and R3 trigger. Both triggers are transferred on the R3/L1 line. The L1 trigger is synchronous to the rising edge and the R3 trigger is synchronous to the falling edge of the BC clock.

3.4 Fast Cluster Finder

The fast cluster finder (FCF) was added as an option to explore the feasibility of the self seeded track trigger. It consist of two logic banks, which detect each two clusters of one or two hits. The cluster information is then serialized and can be transmitted during one BC clock cycle.

Figure 3.7 shows the block diagram of the logic. Each bank has as input the channels from on row of the silicon strip sensor. The FCF takes the channel information after the mask register, as indicated in figure 3.1. The binary information from each input channel is first stored in a input register. Bad channels can be masked by setting the corresponding bit in the mask register. The FCF logic reads only the unmasked channels.

Each fast cluster finder bank finds two clusters. A cluster is one or two adjacent strips with a hit. The maximal width of two performs a first selection of high P_T particles according to the cluster method (see section 2.5.4). Each FCF bank starts the search for clusters at both ends of the channels, i.e. channel 0 and channel 127. As soon as a cluster

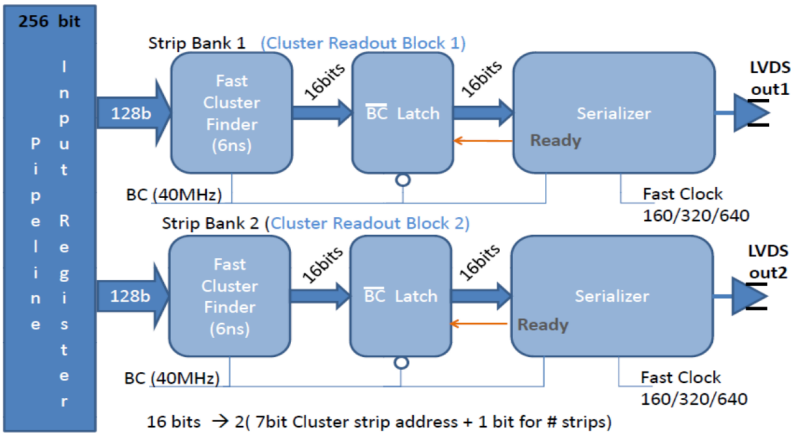


Figure 3.7: Block diagram of fast cluster finder logic [20, Fig. 3-2]

is found, the search is stopped.

The expected occupancy in the HL-LHC is about 1%, i.e. there is around 1 hit per 128 channels of the FCF input. This is also why only two clusters are sent out per FCF block. If there are more than two clusters per 128 channels, the hits in the center channels of the chip are not transmitted and lost.

3.4.1 FCF data output

The output of the each FCF bank is 16 bit long and is transferred on a special dedicated differential output line. Table 3.7 shows the output format of the FCF data. If no cluster is found, then the illegal combination of 8 '1' is sent out on both addresses.

The serialized data are sent with the most significant bit (MSb) first.

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	H size	H address							L size	L address						

Table 3.7: FCF data format. H stands for the cluster at the higher address and L for the second cluster. Size is '0' for a one hit cluster and '1' for a two hit cluster.

The FCF logic takes 6ns to execute and find the clusters. The cluster data are then latched and will be transmitted during the following BC cycle. There is a specific fast clock (FastCLK) signal that provides the readout frequency, which can be either 160MHz, 320MHz or 640MHz.

In the case of a 640MHz FastCLK all 16 bits are transmitted within one BC cycle (40MHz) and the data are processed for each beam crossing. For a 320MHz FastCLK it takes two BC cycles and only every second event can be processed, respectivly every fourth event in the case of 160MHz FastCLK. Figure 3.8 shows the timing diagram for the a 40MHz BC and 640MHz FastCLK.

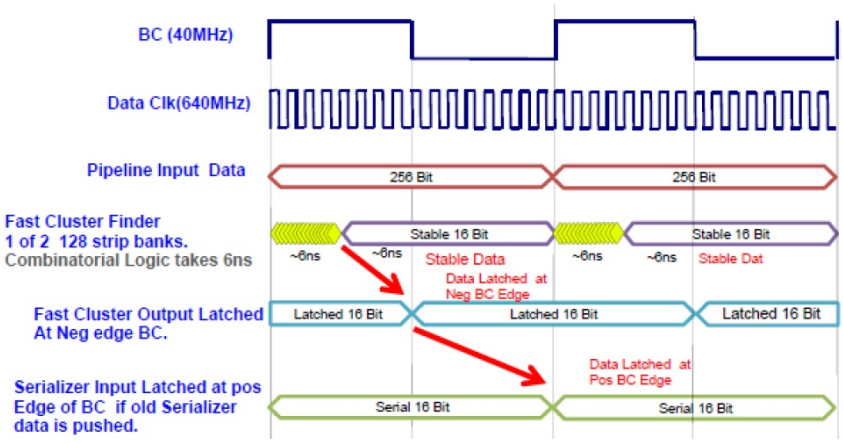


Figure 3.8: Timing diagram for the fast cluster finder signals [20, Fig. 3-3]

3.4.2 FCF framing mode

A special framing mode can be activated with the FCcontrol bits in the configuration register. In this mode, the FCF sends first eight '0' followed by eight '1'. This allows to detect the phase at the FCF outputs

3.4.3 Dependency on the FCLK to BC phase

One issue during the development and test of the doublet correlator was a large difference of the arrival time from the FCF data between the chips. A arrival time difference of up to 25 ns was observed between the ABC130s from the same hybrid. The different lengths of the FCF lines is too small to explain such a behavior.

All possible aspects from termination resistances to ground loops were investigated. The problem was finally found as a dependence of the FCF logic to the phase between the BC and the FastCLK. To verify this and to have the possibility to correct, an ODELAY element was place on the FastCLK lines in the FPGA used for testing.

A simple test was done to see this behaviour and find a good setting for the ODELAY. For each value of the ODELAY, all channels were read. The ABC130 chips were in framing mode and a soft reset was sent after each new ODELAY value. The read values for one FCF line was either `0x00FF` or `0xFF00` plus minus a few bits shifted. I counted how many of the lines were close to `0x00FF` for each ODELAY value. The result is plotted in figure 3.9. The readout of all 20 FCF lines per side, can be done at the value 0 or 10. This means that there are two stable regions where the readout can be performed.

Figure 3.10 shows the FCF signal (Ch1) from one line overlapped with the BC signal in case of two different ODELAY values. The green signal (Ch2) is the measured BC clock in case of a ODELAY value of 21. The blue signal (Ref1) shows the BC clock in case of a ODELAY = 0. The measurement was done with the trigger on Ch1. It can be observed

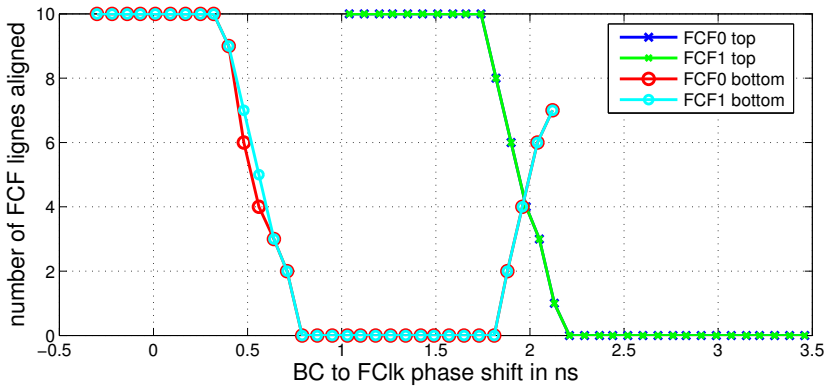


Figure 3.9: FCF readout as function of BC to FastCLK phase. The phase shift on the x-axis was measured for ODELAY 0 and calculated for the other values. The y-axis shows how many lines are close to `0x00FF`. The FCF readout can be performed either for a value 10 or a value 0.

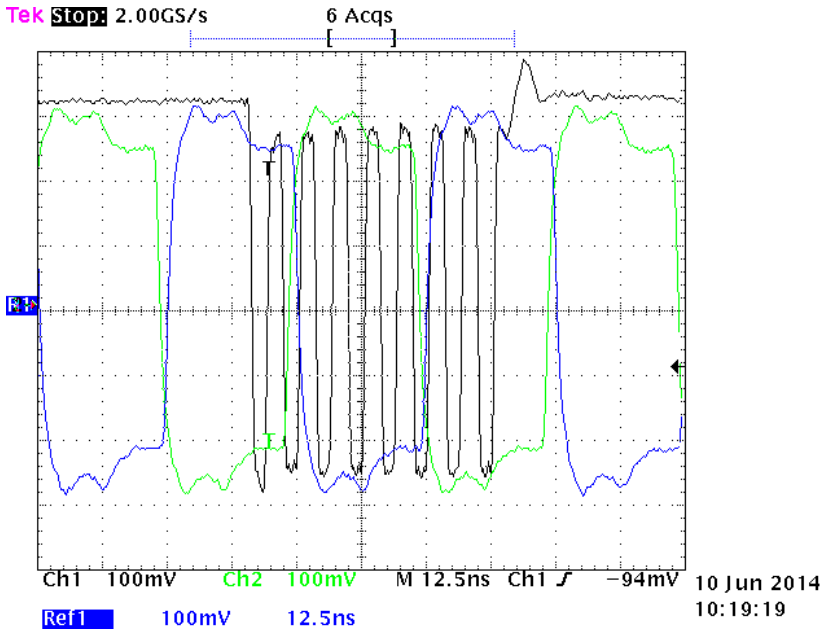


Figure 3.10: Scope measurement showing the dependency of the FCF to the phase between BC and FastCLK. Ch1 is data on the FCF line and used as trigger. Ch2 is BC clock on hybrid with an ODELAY value of 21. Ref1 is BC clock on hybrid with an ODELAY of 0.

that the BC clock is inverted for the two ODELAY values. This indicates that depending on the FastCLK ODELAY, i.e. the phase to BC, the FastCLK data are serialized either on the falling or the rising BC edge.

3.5 Data readout error

As already mentioned the regular data readout is not possible on the ABC130 chips available for this thesis. This is due to a design error in the bit that sets the output direction of the data readout lines. This bit controls the logic together with the output multiplexer. But the bit is flipped, i.e. when a line is set as driver the logic interprets it as sending '0' and if it's a receiver the logic is in the "not listening" state.

The polarity can be corrected with a focused ion beam (focused ion beam (FIB)). This procedure is expensive in means of cost and time. Therefore only a few chips were actually corrected to test the readout of the chip [6].

With the FCF lines an alternative is available which is tested and used in this work. Performing characterization tests with the FCF readout is possible thanks to the mask register. It is done, by masking all channels except four pairs, two on *odd* and two on *even* channels. The data received from the FCF are then bound to be on the unmasked channels.

A corrected version of the chip is in production, but wont be available for this work.

4 ABC130 FCF test system

As already mentioned is this thesis the continuation of the master thesis from Lorenzo Pirrami [3].

A test system was developed by L. Pirrami which allows to configure the ABC130 chips and read data from the FCF lines. It was designed to read the data from two hybrids with 10 ABC130 mounted. The setup is also intended to include the correlator logic for a doublet. Due to timing issues in the FPGA, the ABC130 chips didn't respond to any commands sent from the test system and the system was not tested entirely.

This chapter describes the test system and the developed improvements. Described are as well the calibration tests for the characterization of the ABC130.

During the first weeks of this project, it was possible to obtain a working system and communicate with the ABC130s. Section 4.3 describes the problems encountered and solved.

4.1 Calibration tests

To characterize the readout chips, different tests exist which allow to measure the gain of the FEE and the equivalent noise charge (equivalent noise charge (ENC)) at the input. This section describes the test procedures implemented.

4.1.1 FCF readout delay

The first test to perform is to adjust the timing of the readout. Ideally the data would be in phase with the BC clock. Due to delays from the transmission lines, the measured input signal has to be adjusted before any test can be performed. Figure 4.1 shows a scope capture of a data package from a FCF line.

For this purpose is the FCF deserialization logic clocked by a separate mixed-mode clock manager (MMCM), which allows to set a user defined phase shift for the readout clock. The read-in clock can be shifted by at least one BC period. To adjust the delays from the different FCF lines of one hybrid, an IDELAY element is added on each input. The IDELAYs can be set for each chip individually and allow to compensate a difference of up to 2.418 ns.

A function was created that automatically sets the phase shift and IDELAYs. It's implementation is described in section 5.4.2.

Besides the phase delay, also the delay between the command for the calibration pulse and the readout has to be set. No automatic scan function for this value was implemented, because normally this value doesn't need any modification. A working default value is set in the control applications.

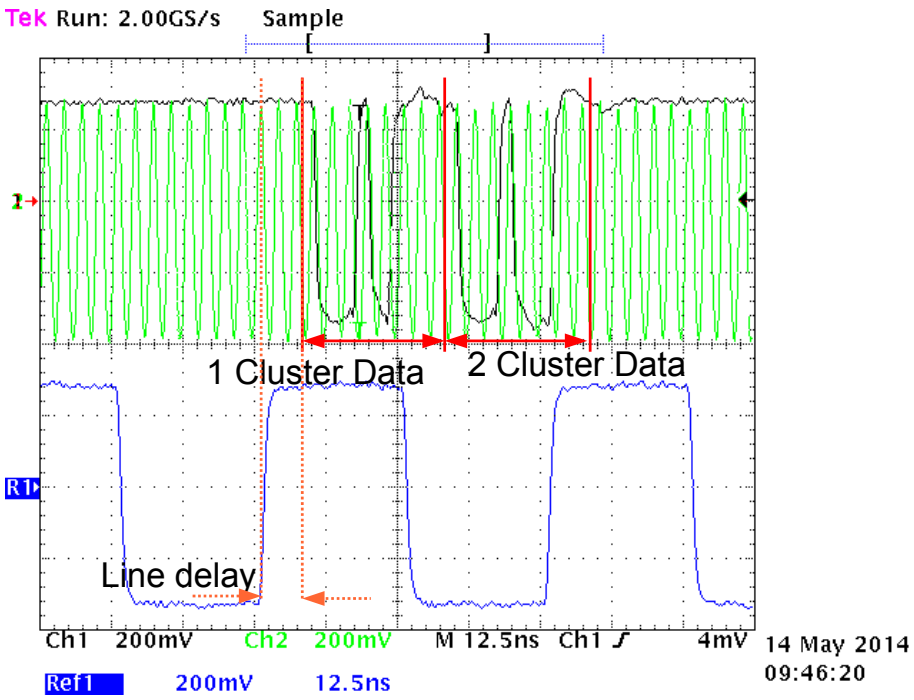


Figure 4.1: Scope capture of the FCF delay. Ch1: FCF in framing mode (black signal), Ch2: Fast clock (green), Ref1: BC clock (blue). A delay is observed between the rising edge of the BC clock and the first bit on the FCF line.

4.1.2 Strobe delay

The strobe delay defines at which point the calibration charge is injected into the input channels as described in section 3.2.1. If the charge is injected to early or to late, no hit will be observed. It is therefore important that the charge is injected at the correct time to have a correct hit rate.

With the strobe delay test, a scan over the complete range is performed. At each step a calibration charge is injected and the response is read. There are several calibration pulses performed at each value of the strobe delay and the average is used to create a plot. In an ideal case, one would want to sample the peak value of the shaper response. Figure 4.2 shows how such a curve could look like. The practical approach to selecte the strobe delay value is using the value at 40% between the delay of mid-rise and mid-fall as defined in [21, section 3.6.12].

It is important to select the strobe delay to the peak of the injected charge signal, which is presumably in the plateau of the plot. If the strobe delay is off, which means also just set to the edge of the plateau in figure 4.2, the injected charge is not correctly observed. This could result in a no-hit, even though the signal would actually pass the threshold. It is just sampled somewhere during the rising or falling time of the pulse. Such an error leads to wrong values for the gain.

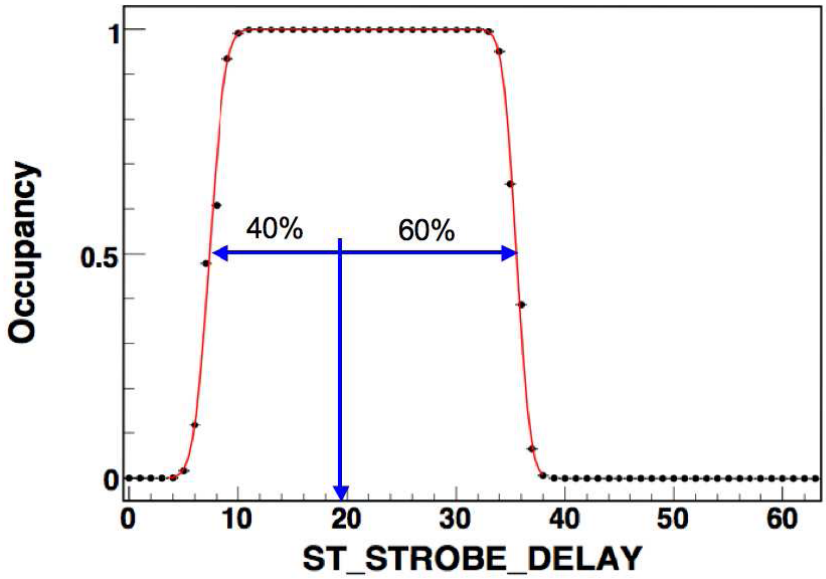


Figure 4.2: Output plot from the strobe delay test. The selected delay is at 40% of hit range. [21, Fig. 3-19]

4.1.3 Threshold scan

One of the most basic tests for a binary system is the threshold scan. This test allows to measure the noise at the output of the amplifier. This test is the basis for all other characterisation measurements.

The threshold scan is performed by injecting a constant charge and varying the threshold value of the discriminator. Scanning the threshold from 0 to maximum will give a plot as shown in figure 4.3, also known as S-curve. At each threshold level, several charge injections are performed and the average hit rate is plotted in the S-curve. In an ideal case the rate would go at a certain threshold directly from 1 to 0. Due to noise there is no immediate transition. The point where there is a 50% hit rate corresponds to the median of the injected charge and is known as V_{T50}.

The derivation of the S-curve gives an error function and the σ of this function is the output noise in mV.

By measuring the V_{T50} and σ for different charges one can calculate the gain and the ENC as described in the next section.

4.1.4 Response curve

The response curve gives information about the linearity of the FEE as well as the gain of the input stage.

The test consists of multiple threshold scans with different charges. For the ATLAS SCT two series were mainly used.

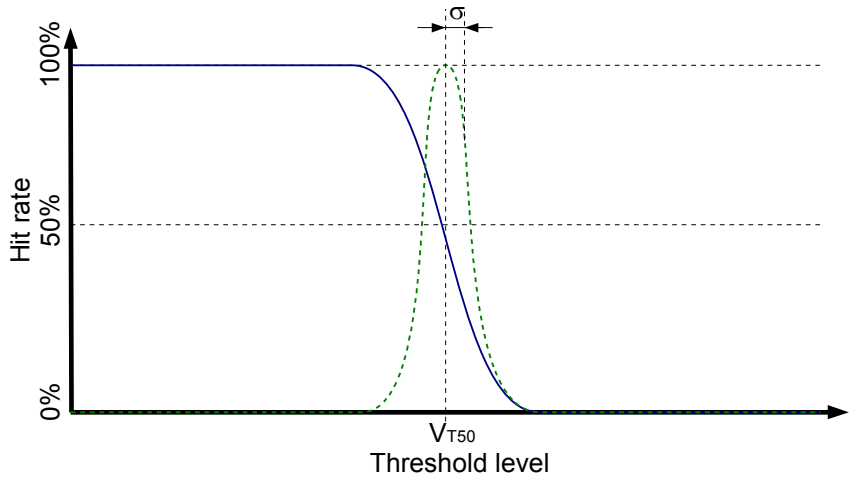


Figure 4.3: Possible S-Curve from a threshold scan (blue continuous line) and derivated error curve (green dashed line)

- 3 point gain: where the three charges are injected. Possible charge values are 1.5fC, 2fC and 2.5fC. This test is used for to check the most linear region of the chip.
- 10 point gain: a more precise test with 10 injected charges (e.g. 0.5fC, 0.75fC, 1fC, 1.25fC, 1.5fC, 2fC, 3fC, 4fC, 6fC, 8fC). Used to calibrate the chip.

For a binary system it is important to have a linear behavior of the amplifier around the desired threshold. In case of the ABC130 the threshold will be set around 1-2 fC. Because of this the 3pt gain measures low charges and also the response curve has a higher resolution for lower charge values. The gain observed at higher charge values is less important as long as it is guaranteed that the signal will always pass the threshold.

For each injected charge the V_{T50} point is measured and plotted as function of the charge. By deriving the response curve the gain of the input stage is obtained. Usually the gain is expressed in fC/mV. It depends on the input charge and declines with increasing charge.

Another important value is the equivalent noise charge (ENC). The ENC indicates the charge at the input, that would create the same output level as obtained from the noise. It can be calculated by

$$ENC = \frac{\sigma}{Gain} \tag{4.1}$$

and is usually expressed in e^- (electrons), where $1fC = 6241e^-$ and σ is the output noise in mV.

Figure 4.4 shows an example of the response curve together with the gain, output and input noise. As stated above, it is more important to have a linear amplifier below the expected charge deposition of a particle, expected to be around 4fC [15, section 10.4.2.1]. More characterisation measurements done during this work are documented in chapter 6.

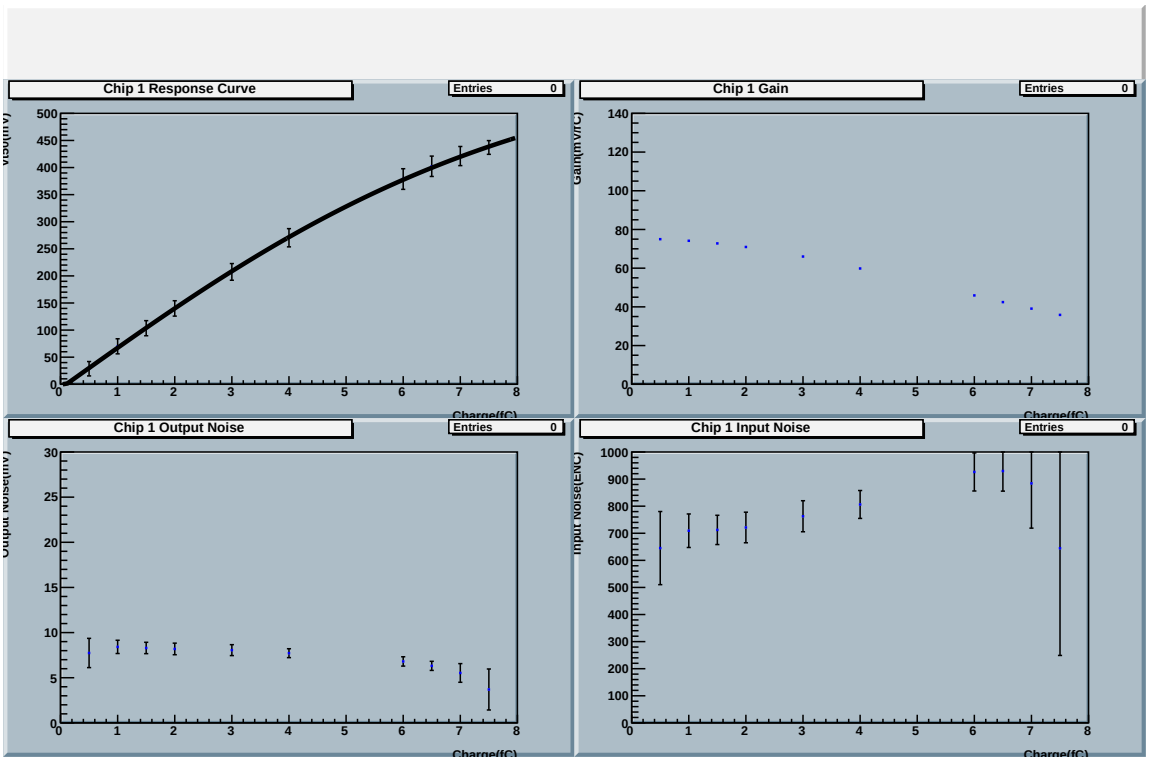


Figure 4.4: Example of a response curve plot

4.2 Hardware

The development of the hardware for the FCF test system was part of two previous bachelor thesis done by Eric Ropraz [5] and Silvan Duner [4]. Four different boards form the system:

1. The FPGA board where the logic of correlator and the test system is integrated. Used is the ML605 evaluation board for a Virtex 6 FPGA[1], see section 4.2.1.

2. An interface board to host the low voltage differential signaling (LVDS) and SLVDS drivers used to adjust the levels of the differential lines, described in section 4.2.2

3. A support board to hold one silicon sensor with one hybrid print mounted. Two support boards are used in a chain to create a doublet. The silicon strip sensor is fixed in the middle of this board. More details in section 4.2.3

4. The FCF hybrid, which holds the ABC130 chips and has the connectivity for the FCF readout. The hybrid is glued onto the silicon sensor. The hybrid board is described in section 4.2.4.

[1] *http://www.xilinx.com/products/boards-and-kits/EK-V6-ML605-G.htm*

4.2.1 Virtex 6 Evaluation Board

The ML605 evaluation board is used to host the test system. A Virtex 6 XC6VLX240T-1FFG1156 FPGA is mounted on this board together with additional hardware for testing purposes. The complete documentation can be found in [22]. It provides features for embedded processing systems. Only few are used for the test system. The following features are used:

- **USB to UART bridge**: used for the communication between PC and the softcore processor.
- **VITA 57.1 FMC connectors**: the HPC (high pin count) and LPC (low pin count) FMC connectors available are used to connect the interface board over LVDS lines. Both connectors provide together 112 differential signal lines, 6 differential clocks. Additional 9 MGT (multi-gigbabit transceiver) signals and 3 MGT clocks are available but not used.
- **User GPIO**: for test and control purposes

The features below might be interesting for future extensions:

- **Multi-Gigabit Trasnceivers (GTX MGTs)**: for communication between two FPGA boards to exchange data for the higher level correlation.
- **10/100/1000 Ethernet PHY**: for faster communication to PC

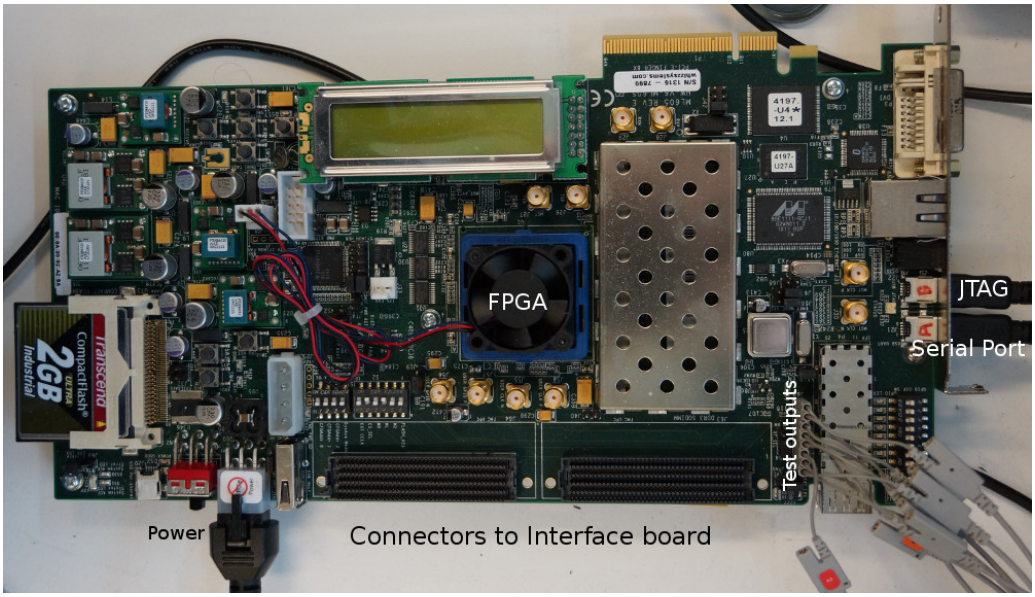


Figure 4.5: The Virtex 6 evaluation board used for the test system.

4.2.2 Interface board

The interface board is a board that provides multiple LVDS and SLVDS drivers for the communication between the FPGA and the ABC130 chips.

SLVDS is a variant of the LVDS standard and uses lower voltages better suited for the 130nm technology used to fabricate the ABC130 chips. Table 4.1 compares the voltage levels of the two differential standards. Except for the supply voltage, the levels of SLVDS are within the range of LVDS.

Characteristic	**SLVDS**	**LVDS**
Supply voltage	1.2-1.5 V	2.4-3.3 V
Min. common voltage	0.75 V	0.05 V
Max common voltage	1.05	2.35 V
Min differential voltage	100 mV	100 mV
Max differential voltage	200 mV	2.4 V

Table 4.1: Comparison of LVDS and SLVDS voltage levels [5, Section 7.3]

It was designed during the work of S. Duner [4] and is shown in figure 4.6, where the most important elements are marked. The connection to the support board is made by 6 differential connectors, visible in the right of figure 4.6. The signals from top to bottom are: FCF lines bottom module, control signals bottom module, FCF lines top module, control signals top module, test lines top module and test lines bottom module. The test lines were intended for testing the FCF hybrid boards. They aren't used for this purpose anymore. Though some of the test lines are now used for other purposes. Section 4.3.3 describes modifications done on this board.

The selected SLVDS drivers on this board are working only up to 320MHz. Currently it is therefore not possible to test the FCF outputs at maximal speed of 640MHz.

4.2.3 Support board

The support board was designed to hold the sensor with the hybrid board. It is also a result of [4]. There is the possibility to connect two support boards in series. The first will be the top module and the second the bottom module of the doublet.

A revised version of the support board corrects the position of the hybrid and does not include the test lines anymore. These lines were originally intended to create a loopback test, as long as the ABC130 chips were not yet available. These lines are no longer required and could therefor be removed.

4.2.4 FCF Hybrid

A specific PCB board was designed by E. Ropraz [5], which includes the output lines for the FCF interface of the ABC130. It is called hybrid, because it holds elements different

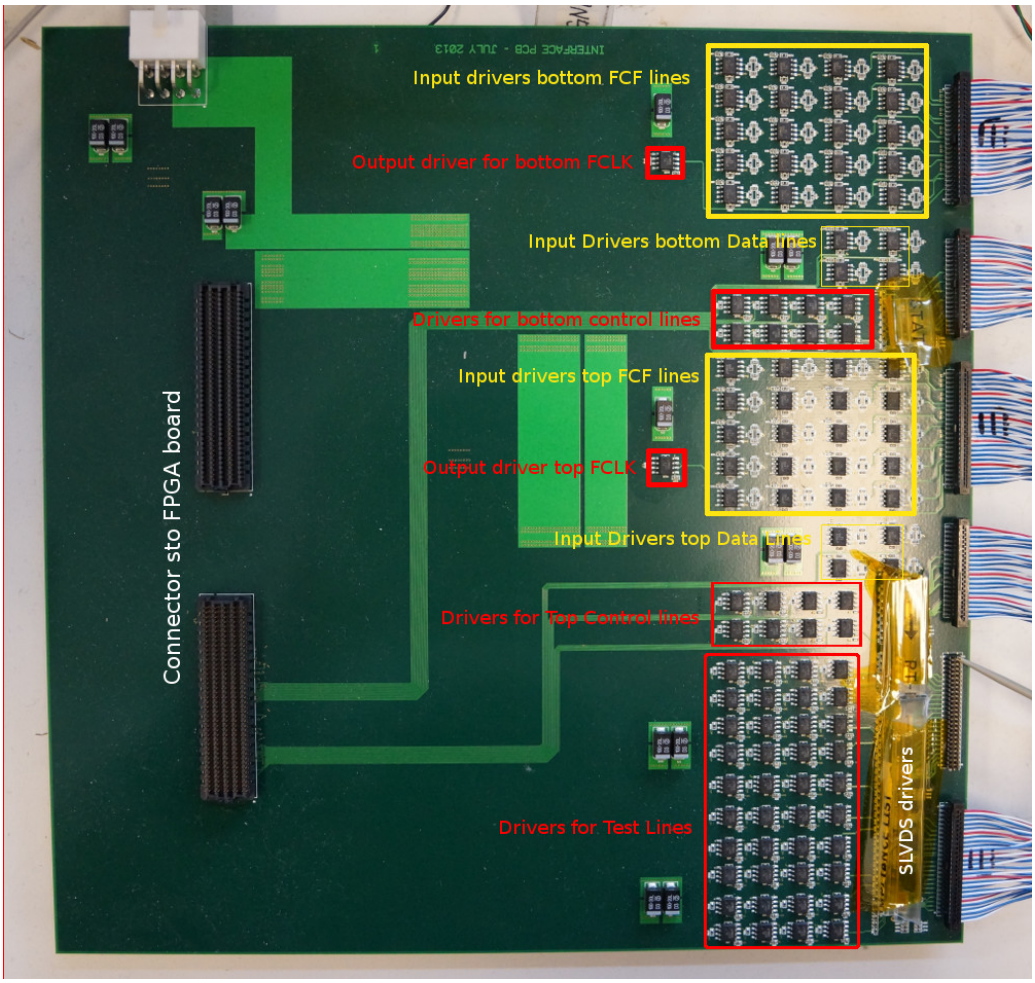


Figure 4.6: Interface board with the LVDS and SLVDS drivers.

kinds of packaging, i.e. bare die and regular SMD components. Figure 4.8 shows the hybrid board with 10 chips mounted. The hybrid in the picture is not yet mounted on a sensor and was used to test the communication and the functionality of the ABC130 chips, primarily the FCF readout.

4.3 Modifications of the existing FCF test system

The system to test the FCF readout was developed as part of [3]. During the first part of this work, the system was completed and modified to have a working environment. This section describes which problems were encountered and how they were solved, together with the modifications and improvements made on the test system.

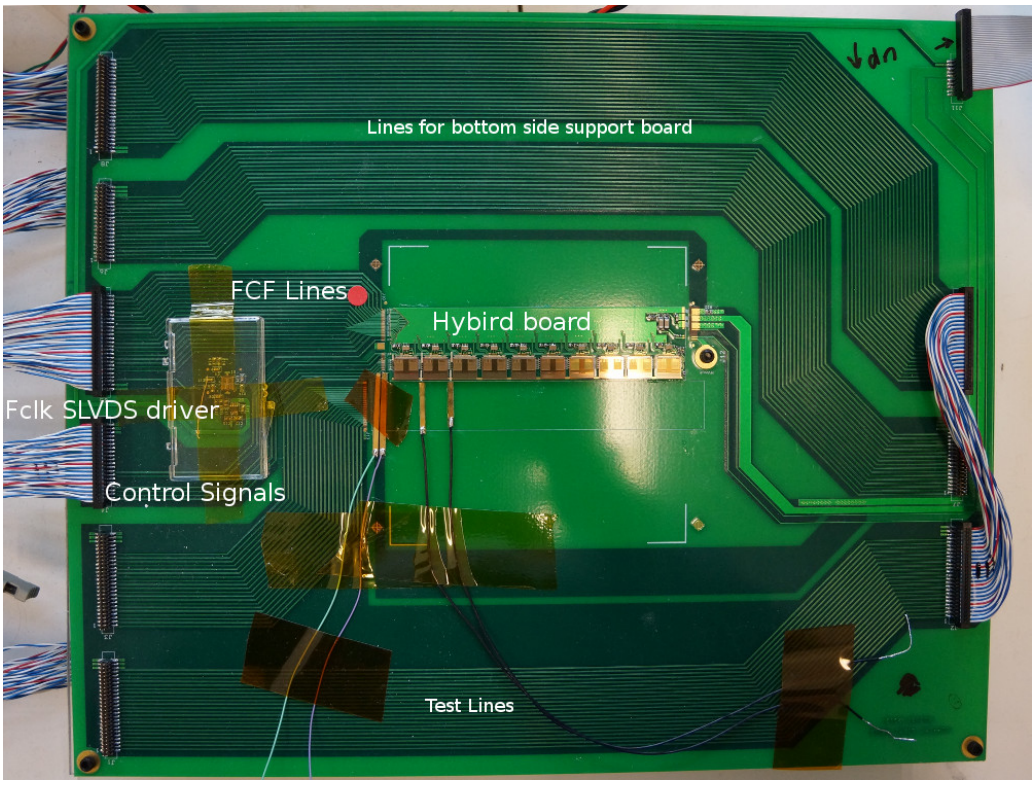


Figure 4.7: Support board version 1. Version 2 has an adjusted position of the hybrid and doesn't include the test lines anymore.

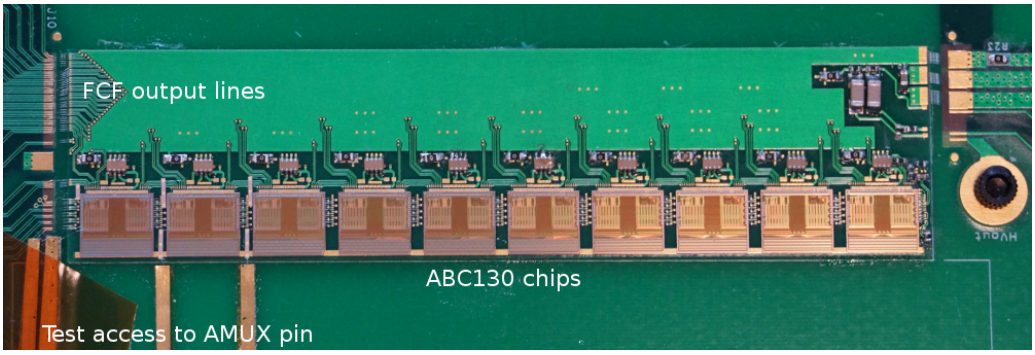


Figure 4.8: FCF hybrid board with 10 ABC130 mounted. The first three chips have additionally test access to measure the AMUX output pin.

4.3.1 Internal timing problems

The communication with the ABC130 chips did not work at the start of the work. [3] showed a correct signal on the termination resistances on the hybrid board. Why this command weren't interpreted by the ABC130s could not be determined. Most likely it was

due to some timing problem between the BC clock and the L0CMD line.

At the point I started this thesis, it wasn't possible to capture the commands anymore. To observe the internal signals of the FPGA, the test pins on the ML605 boards were used. Once the internal L0CMD line was added to such a test pin, the communication started to work and the ABC130s were responding.

The reason why the command signal was not generated properly is most likely problems with the clock signals inside the FPGA. It was observed that many timing constraints could not be met. Most of the failing constraints were not of importance, because the source signal in question was a register with a constant value once the system is initialized. Others caused problems like the command to send was not correctly loaded into the register for the serialization. The timing analysis showed also a large clock skew overall the design. For this reason, the clock distribution in the design was modified and improved. A simplified clock tree of the original and redesigned system are shown in figure 4.9 and figure 4.10 respectively. The register transfer level (RTL) schematics with all the clock signals are digitally available.

Not shown in figures 4.9 and 4.10 are the clock buffers and inverters added to the clock inside the different blocks like the command and trigger logic. During the redesign, I rewrote the VHDL code, so that flip flops with the correct edge sensitivity were instantiated.

Further I decided to remove the 640MHz part and work only with the scaled clock for now. This helps to simplify the design and due to the SLVDS driver it's currently not possible to test the 640MHz. The final correlator design was running with a 20MHz BC and a 320MHz FastCLK. These frequencies allow to read the clusters from the FCF on every BC cycle.

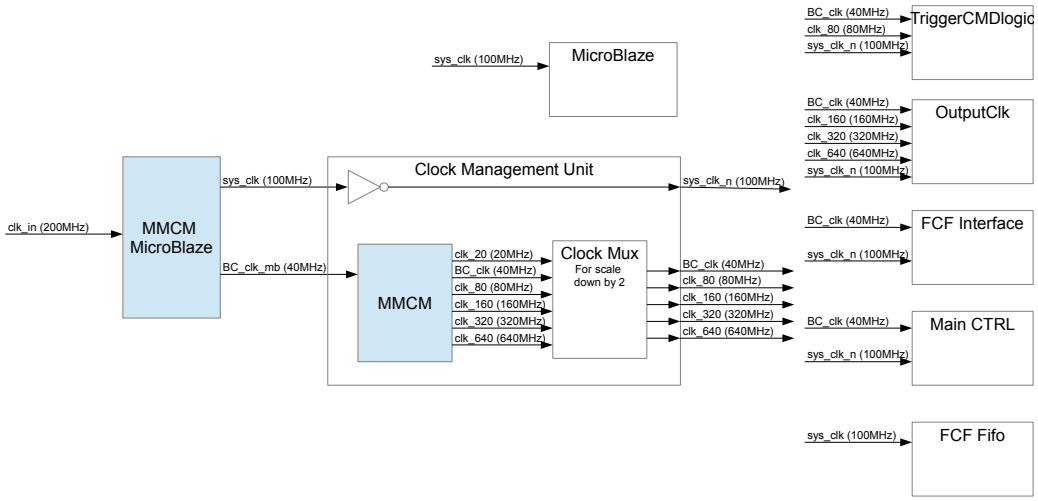


Figure 4.9: Schematic of the clock tree for the original ABC tester design. Buffers and further inverters inside the different blocks are not shown.

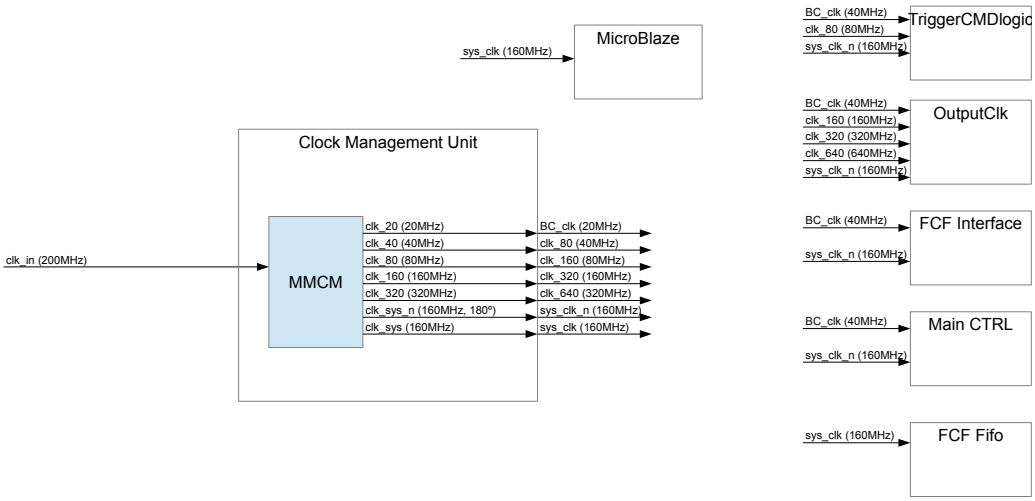


Figure 4.10: Schematic of the clock tree for the redesigned ABC Tester. Buffers and further inverters inside the different blocks are not shown. The RTL schematic with all clock signals in available in the digital appendix.

4.3.2 LabVIEW interface update

To control the test system a LabVIEW application is used. LabVIEW calls it's programs virtual instruments (virtual instrument (VI)) The original interface allows to control the complete system and send single commands. To add the functions for testing the ABC130 chips, the original VI was split into smaller SubVIs and a new interface was developed, shown in figure 4.11.

Following is a brief list of actions that can be perfomred with this application:

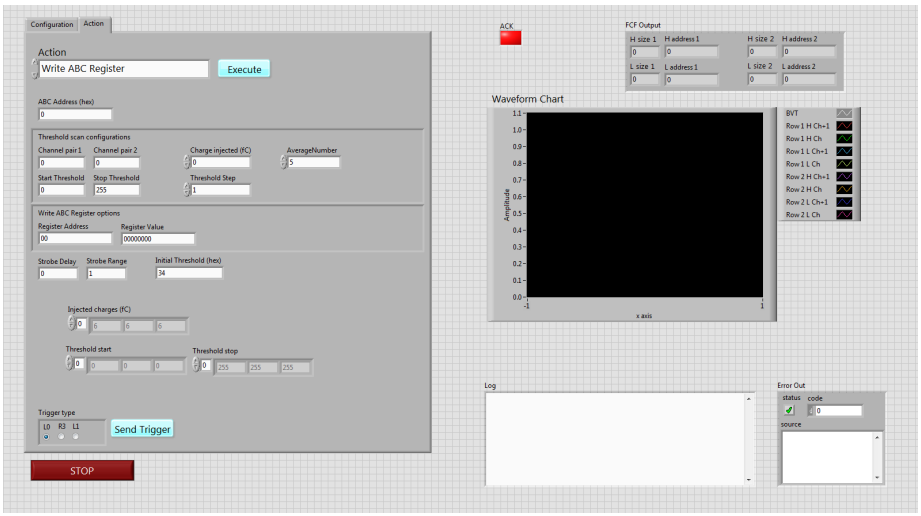


Figure 4.11: Screenshot of the LabVIEW FCF readout tester front panel.

- **Action**: allows to select the task to be executed with a click on the *Execute* button. Possible tasks are from writing a single register in the ABC130 to perform complete test like threshold scans or response curve.
- **ABC Address**: is used to define which ABC shall be used.
- **Threshold scan configurations**: fields to enter the parameters for a single threshold scan.
- **Write ABC Register options**: input to set a specific value to a register, when executing the *Write ABC Register* task.
- **FCF Output**: shows the values read by the *FCF read* task.
- **Waveform Chart**: shows a plot of the threshold scans performed to give a first indications of the result.

This interface was used for the first test to verify the functionality and characteristics of the ABC130s using the FCF lines for readout. It uses the MicroBlaze (MB) program developed in [3] to control the FPGA. For the correlation tester, a new program for the MB was written, together with a new LabVIEW application. The new application was designed to read and record the data from the correlations and is described in section 5.6.

4.3.3 Hardware modifications

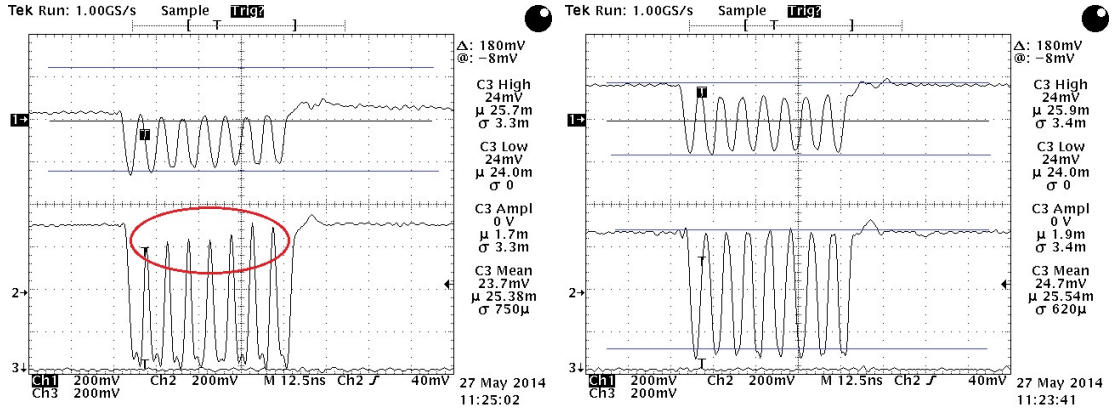
During the development and test of the system, some modifications had to be made on the hardware. All modifications done are described in this section.

AC coupling of the FCF lines

The interface board, described in section 4.2.2, contains an AC coupling on the FCF lines. This was done, because the SLVDS coming from the ABC130s have a different common mode voltage than the LVDS required for the FPGA. The circuit is described in [4, section 7.1]. Figure 4.12 shows the signals with and without AC coupling. In case of AC coupling, the input signal is not balanced. The output is still recognizable, but as can be seen not all bits at one are rising to the same voltage level. This could lead to a wrong value read by the FPGA.

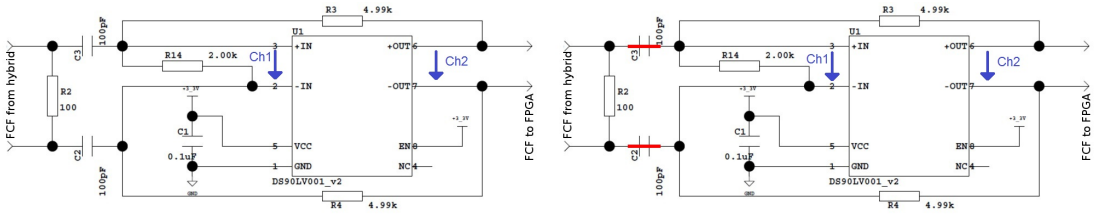
The problem here is that the FCF lines have a non-DC balanced signal. Most of the time it is at '1', only when data are arriving the line changes. As described in [23, section 1] AC coupling is not recommended for non-DC balanced signals and can cause baseline wandering. This was observed during the tests and as shown in figure 4.12a the signal at the input of the LVDS driver has non-symmetric differential signal, which can result in a wrong value read by the FPGA.

The LVDS driver used has a wide acceptance range for the differential input voltage. This range allows the driver to correctly interpret the SLVDS signal without any further circuit. By replacing the capacitors with a wire, as shown in figure 4.13b this problem could be solved. Figure 4.12b shows the correct signal after the modification. The effect



(a) Signals measured with the AC coupling (b) Signals measured without the AC coupling

Figure 4.12: The signals measured at the input (CH1) and output (CH2) of the LVDS driver used to adapt the levels from the SLVDS signals. The signals were measured with differential probes.



(a) Original SLVDS to LVDS conversion (b) Shorted capacitor

Figure 4.13: Schematic for the SLVDS to LVDS conversion used on the interface board together with the applied patch [4, Figure 24]. The arrows indicate where the signals in figure 4.12 were measured.

from the additional resistors of the AC coupling circuit is negligible. Though they should be removed in a revision of the interface board.

The solution presented here works well for a single doublet. In a stave, the modules will be powered serially and each module will be at a different voltage level. In this case a AC coupling is indispensable. The feedback resistors R3,R4 and R14 in figure 4.13 are used to compensate the drift from the non-balanced signal. The values of those resistances should be adjusted to get a working circuit with AC coupling.

Inversion of lines

A mismatch between the pin numbering of the connectors from the interface board to the support boards leads to an inversion of the differential signals, i.e. the p and n line are crossed. This problem was solved with inverters inside the FPGA. All the signals coming

from and going to the ABC130s had to be inverted.

For the FCF readout, the inversion was introduced after the deserialization to reduce the amount of logic in the input path.

The BC signal was inverted by using a second clock output on the MMCM with a 180 °C phase shift.

The FastCLK lines don't have an inverted on the FPGA, but a ODELAY element as described in section 3.4.3.

Bottom side fast clock

The originally assigned line for the bottom FastCLK on the FPGA, did not drive a strong enough differential signal to be correctly transmitted by the LVDS driver. Figure 4.14 shows the clock signals for the top and bottom module, where the amplitude of the bottom line is almost 10 times smaller than the top line. To solve this issue, the bottom clock was rerouted to an output originally intended for emulating FCF signals. This pin is connected to one of the differential connectors for the test lines. A specific cable was made to have the correct signals together. Table 4.2 indicates which pin is was finally used. After changing the clock pin, both signals looked the same.

	FPGA pin	**Interface board line**
original	AF26|AE26	IB_BOT_FCLKp|n
new	F31|E31	EM_TOP_FC2_p|n_4

Table 4.2: bottom FCLK pins originally intended and actually used.

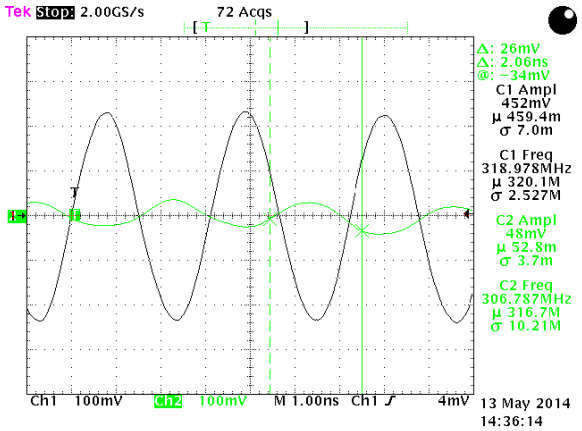


Figure 4.14: The FCLK signals measured on the interface board. Ch1 is the top line and Ch2 is the bottom line. The amplitude for the bottom line is too small to be correctly transmitted by the LVDS driver.

Trigger Logic Unit connector

The trigger logic unit (trigger logic unit (TLU)) was used during the test beam as described in chapter 7. To connect to the TLU, an RJ45 connector had to be added, which has four differential signals. There aren't enough free differential lines on the ML605 test board. Therefore some of the emulation lines on the interface board (IB) were selected for these lines. The lines used are listed in table 4.3. The LVDS drivers on the IB can only serve as outputs. The RJ45 connector was soldered directly to the inputs of the drivers for this reason .

TLU signal	**FPGA pin**	**Interface board line**	**RJ45 pin**
Reset	R26|T26	EM_BOT_FC1_p|n_4	4|5
Trigger clock	N27|P27	EM_BOT_FC2_p|n_4	1|2
Trigger	P29|R29	EM_BOT_FC1_p|n_3	7|8
Busy	L29|L30	EM_BOT_FC2_p|n_3	3|6

Table 4.3: Pins used for the connection to the TLU

5 Doublet correlator

This chapter gives an overview of the functionality and the implementation of the correlator for one doublet, which is shown in figure 5.1.

The doublet correlator reads the FCF lines from two hybrids, mounted on a double sided module. It correlates cluster hits from the two modules and generates information about track stubs.

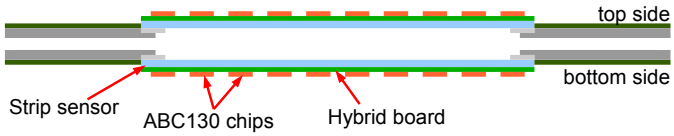


Figure 5.1: Sketch of a doublet

5.1 Doublet correlator architecture

The correlator implements the offset method, as described in section 2.5.4. For this work, the correlator was implemented for a single hybrid with 10 ABC130 chips, because the test setup will be built of modules with only one hybrid. The implemented design could be extended to work for multiple hybrids, by reusing the basic correlation logic.

The correlator is calculation based and has three stages as shown in figure 5.2.

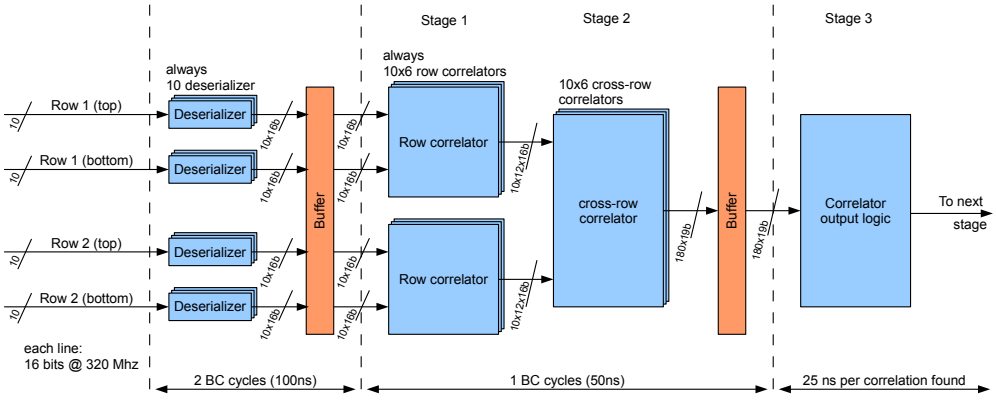


Figure 5.2: Block diagram of the correlator logic. The timing is given for the case of a 20MHz BC and 320MHz FastCLK. The final speed to be used should be two times faster.

Stage 1. performs a correlation of the information from the same strip row. Stage 2. takes as input the results from stage 1 and makes a cross-row correlation. The same algorithm is used on both stages, with additional logic in the second stage to adapt the input data and reduce fake track generation. Stage 3. filters and compresses the result and transmits the stub information to the higher level track correlator.

5.1.1 Calculation algorithm

The basic element is a combinatorial logic circuit. A more detailed description of the algorithm can be found in [3, chapter 5].

The basic idea is to calculate the distance between two hits. The result is valid if it is inside a certain window range. Figure 5.3 shows the pseudo-RTL schematic of the algorithm. The window sizes are provided by integrated memory, which allows some flexibility regarding the position of the chips.

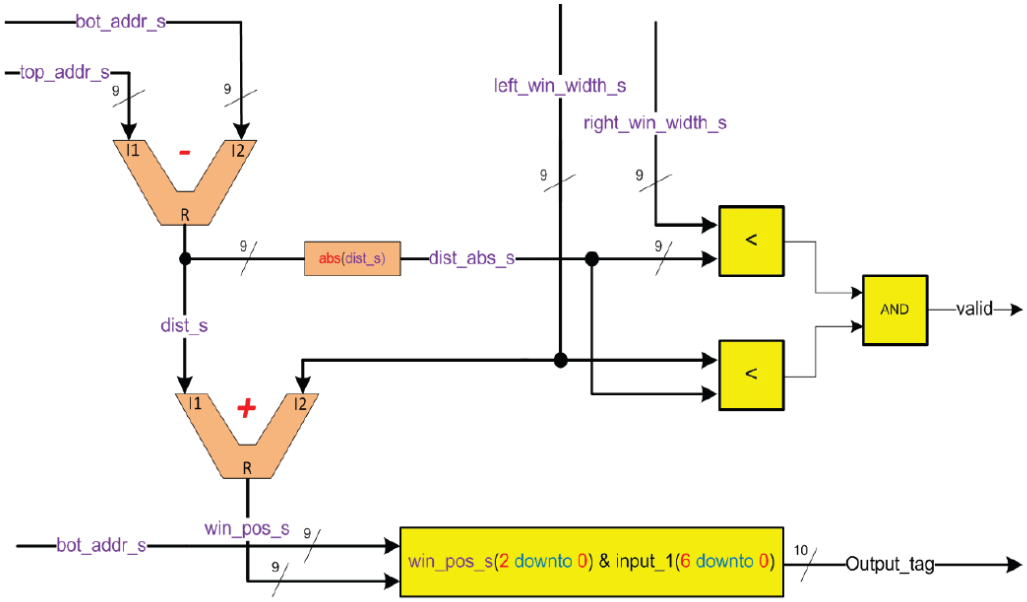


Figure 5.3: Pseudo-RTL of correlator basic block ([3, Figure 5.16]).

18 such correlation blocks are used for each pair of ABC130s. One block creates a possible track, called stub, which is encoded with different fields of a 16 bit number as described in table 5.1 and called tag.

The row correlation of stage one has six possible tracks, as shown in figure 5.4. Therefore uses aach of the two strip rows 6 correlation blocks in stage 1 and another 6 blocks are used for the cross-row correlation in stage 2.

The cross-row correlation in stage 2 checks first if there is a possible track between two rows. To reduce false positives, this is only done if a bottom cluster has three missing top tags.

bit	15-14	13-11	10-4	3-2	1-0
	missing	distance	address	cluster	row

- *missing*: no corresponding hit on top or bottom
- *distance*: difference between top and bottom hit
- *address*: bottom hit cluster address
- *cluster*: top and bottom cluster size
- *row*: strip row of top and bottom hit

Table 5.1: Internal tag format for one stub. See also [3, section 5.3.2].

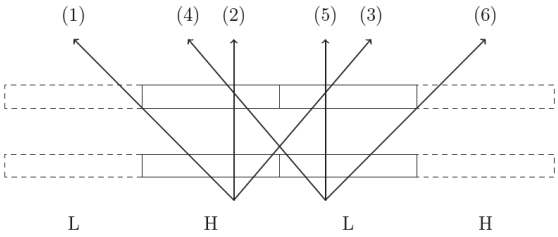


Figure 5.4: Principle of the row correlation ([3, Figure 5.17]). H and L are the two addresses coming from the FCF interfaces. The dashed blocks are the values from adjacent chips.

Modifications and filter pre-selection of good results

If only one cluster is found in a row, the ABC130 chips sends two times the same cluster information on the FCF line. This leads to equivalent tags in the correlation. E.g. if the bottom clusters are equal, then the following correlations are the same: (2) = (4) and (3) = (5) as can be seen from figure 5.4. Additional logic was implemented in the row and cross-row correlator blocks to remove these redundant tags.

Only valid tags have to be transmitted to the high level correlator. A valid tag has a bottom and top hit, which are within the defined window. A tag, as shown in table 5.1, is valid if the missing field is “00” and the distance field is not “000”. Each tag is checked after the cross-row correlation and a valid bit is set accordingly. This valid bit is calculated in the logic and set together with the other fields. The valid bit is used in the correlator output logic to select the tags, which have to be transmitted to the next stage. Section 5.1.3 describes the correlator output logic.

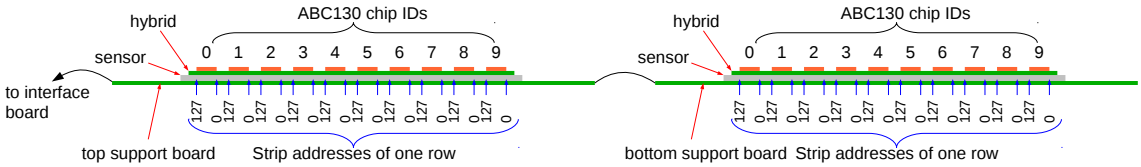
After the cross-row correlation, the chip ID is added, which corresponds to the 4 lowest bits of the ABC130 address (0 to 9 on one hybrid).

bit	18	17-14	13-11	10-4	3-2	1-0
	valid	chip ID	distance	address	cluster	row

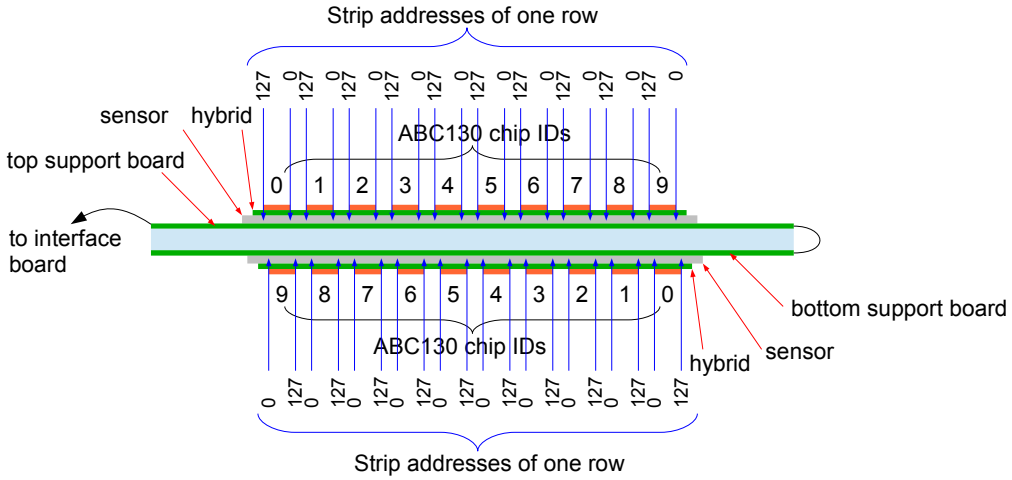
Table 5.2: Format for one correlation. The valid bit is set if the tag contains a valid correlation and the tag has to be sent to the track correlator. If the valid bit is '0' then the tag will be discarded.

5.1.2 Correlator for single hybrid doublet

The correlator logic described in the section above is applied to each pair of ABC130 chips. The test setup will hold two modules with one hybrid each. Therefor the correlation will be done between the twenty chips. As described in section 4.2.3, the two modules are connected in series and the second will be flipped to form the bottom module. This results, that the chip and channel numbering is inverted on the bottom module. Figures 5.5 and 5.6 show how the FCF lines have to be connected together for a correct correlation.



(a) Sketch of the support boards connected in series. The number of the ABC chips corresponds to the number of the FCF lines in the schematic.



(b) Sketch of the doublet test setup with the chip numbering and channel numbers per chip from one row

Figure 5.5: Sketch how the correlator logic has to be connected together in respect with the test setup of the doublet. 5.5a shows the two modules as mounted and 5.5b shows when the bottom module is flipped.

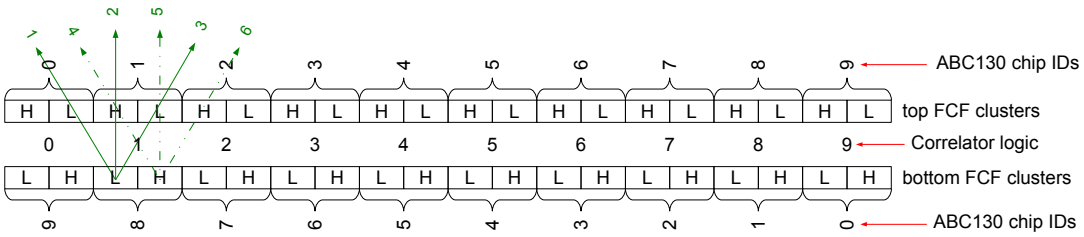


Figure 5.6: Relation of the FCF clusters for the correlator logic. The green arrows show the row-correlation one correlator logic. The numbers (1 to 6) correspond to the basic correlator calculation block described in section 5.1.1. Each of the ten logic blocks has the connection according to this pattern. H and L are the clusters as described in table 3.7

Because the bottom module is flipped, the addresses don't fit together as $0 \leftrightarrow 0$ to $127 \leftrightarrow 127$, but have to be matched like $0 \leftrightarrow 127$ to $127 \leftrightarrow 0$ (see in figure 5.5b). For the correlator algorithm to work, one of the addresses has to be adapted to the other. For example, the adapted bottom address is 127 minus the received cluster address. This change of address has to be respected also in the cross row correlation and in the stage 3, where the correlator results are prepared for the track correlator.

Not included yet is the integration of cross-row correlation over two hybrids on the same sensor module and the correlation over several modules on the same barrel. The cross-hybrid correlation requires a more profound change to the logic, so that the cluster information could be exchanged between two hybrid correlators. An other approach is to create a larger correlator, which can handle the data from two hybrids.

Regarding the cross module correlation, a simple implementation is to add input lines which will be used to receive the top cluster address of the neighboring modules.

5.1.3 Stage 3: Correlator output logic

Only valid tags are sent to the next stage. In a first step this is done by storing the valid tags in a FIFO. The logic for this is shown in figure 5.7.

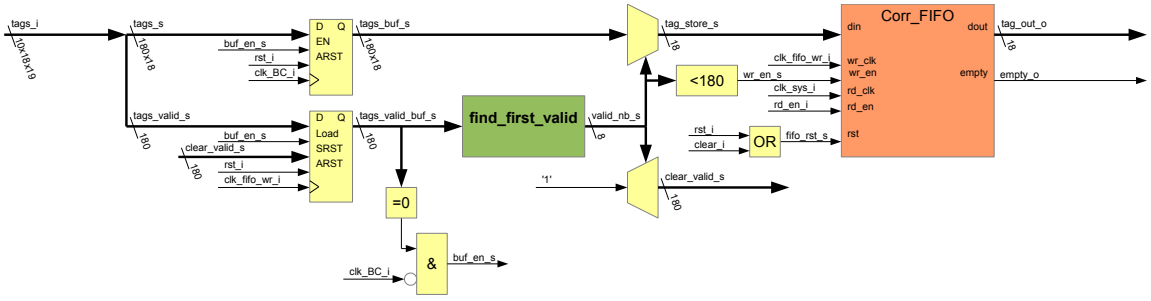


Figure 5.7: RTL schematic for the correlator output logic.

The output logic functions without any trigger signal and looks only at the tags generated by the correlation logic. The results from the correlation logic are stored in two buffers, one for the tags and one for the valid bits. A dedicated logic is implemented to select the valid tags and write them to the FIFO. As long as there are valid tags to be written to the FIFO, no new tags are stored. This assures that all tags from a correlation are stored in the FIFO. A drawback is that the output logic is blind to new data, as long as there are still valid tags in the buffer. If there are many valid tags, the transfer to the FIFO might take longer than one BC cycle and the 2nd of two sequential hits might be lost.

Find first valid tag

The **find_first_valid** block searches for the first '1' bit in the valid bit buffer. It takes as input a vector consisting of the valid bits from each generated tag. On each cycle the bit number of the lowest bit at '1' is returned. E.g. the vector "0010 0100 1000" would return 3 and the vector "1010 0110 0000" results in 6. These examples are only for 12 bits and the logic implemented works for 180 bits.

After finding the first valid tag, the selected tag is stored in the FIFO and the corresponding valid bit is cleared. The next valid tag is then stored in the following clock cycle and so on until all valid tags are written to the FIFO. The rate of data arriving has to be slow enough to allow every valid tag to be written into the FIFO. Otherwise, some tags could be lost.

Listing 5.1 shows the VHDL code for the **find_first_valid** block[1]. The goal of this logic was to use only as many cycles as there are valid tags to store them. Because there are 180 possible tags, the selection logic has many stages of multiplexers and comparators and creates a long path delay. In the first implementation a delay of 19ns was achieved and the write clock had to be set to 40MHz. Using a 20MHz BC, only two tags can be written per BC.

Code Sample 5.1: Find first valid algorithm

```
1  valid_nb_s(7) <= '1' when
     (tags_valid_buf_s(127 downto 0) = X"00000000000000000000000000000000")
3    else '0';
   v6 <= X"0000000000000000" & tags_valid_buf_s(179 downto 128) when
5   (valid_nb_s(7) = '1')
    else tags_valid_buf_s(127 downto 0);
7  valid_nb_s(6) <= '1' when
     (v6(63 downto 0) = X"0000000000000000")
9    else '0';
   v5 <= v6(127 downto 64)  when
11   (valid_nb_s(6) = '1')
     else v6(63 downto 0);
13 valid_nb_s(5) <= '1' when
     (v5(31 downto 0) = X"00000000")
15   else '0';
   v4 <= v5(63 downto 32) when
17   (valid_nb_s(5) = '1')
     else v5(31 downto 0);
```

[1] inspired by the response from
http://stackoverflow.com/questions/2368680/count-leading-zero-in-single-cycle-datapath

```
19 valid_nb_s(4) <= '1' when
     (v4(15 downto 0) = X"0000")
21   else '0';
   v3  <= v4(31 downto 16)
23   when (valid_nb_s(4) = '1')
     else v4(15 downto 0);
25 valid_nb_s(3) <= '1' when
     (v3(7 downto 0) = X"00")
27   else '0';
   v2 <= v3(15 downto 8) when
29   (valid_nb_s(3) = '1')
     else v3(7 downto 0);
31 valid_nb_s(2) <= '1' when
     (v2(3 downto 0) = X"0")
33   else '0';
   v1 <= v2(7 downto 4) when
35   (valid_nb_s(2) = '1')
     else v2(3 downto 0);
37 valid_nb_s(1) <= '1' when
     (v1(1 downto 0) = "00")
39   else '0';
   v0 <= v1(2) when
41   (valid_nb_s(1) = '1')
     else v1(0);
43 valid_nb_s(0) <= not v0;
```

Final tag

For the test beam, an additional field of 14 bits was introduced to include the trigger ID. The final tag stored in the output FIFO is described in table 5.3. The trigger ID can either come from an internal counter, which is incremented on every calibration pulse or TLU trigger signal. Or it is taken from the TLU input. To take into account the delay between the arrival of the correlations and the TLU trigger ID, a two stage FIFO was introduced. By delaying the transfer from the first to the second FIFO, the trigger ID received from the TLU should be correct. However an event based readout was implemented in the softcore program during the test beam, which made the two stage FIFO obsolete.

bit	31-18	17-14	13-11	10-4	3-2	1-0
	trigger ID	chip ID	distance	address	cluster	row

Table 5.3: Correlator output tag format

5.1.4 Time diagram

The correlation itself can be executed within one BC cycle. With all the buffer and phase recovery, it takes several cycles from the event until the valid tags are available.

Figure 5.8 shows the time diagram for the different signals. The red arrow indicates an event loaded into the ABC130 pipeline. The FCF takes about 6 ns and the result is registered on the falling BC edge (first green arrow). The clusters are then serialized and transmitted to the FPGA, where the other green arrows indicate how the data passes from

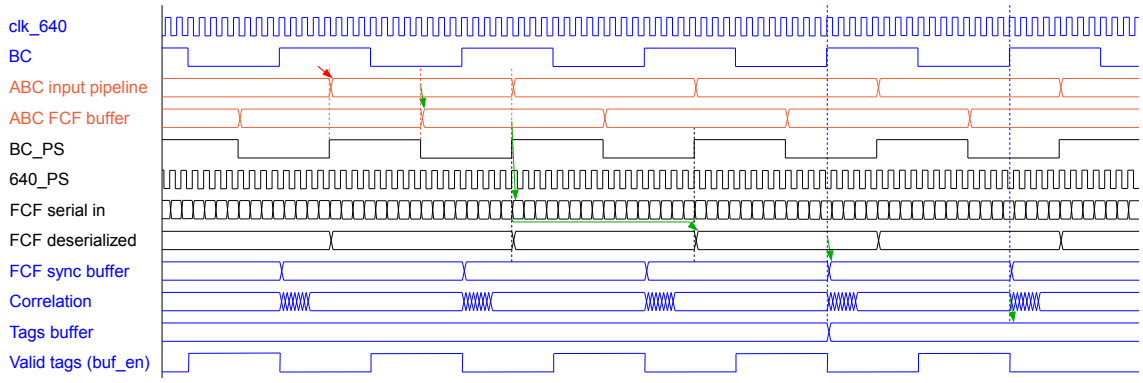


Figure 5.8: Timing diagrams for the data coming from the ABC130 to the correlator. The blue signals are FPGA internal and synchronous with the main BC clock. The orange signals are ABC130 signals. The black signals are the FCF input signals and synchronized with the recovered phase from the FCF lines.

one signal to the next. The cluster data are buffered and synchronized with the system clock after deserialization and before entering the correlation logic.

The result from the correlator is available after four BC cycles. Another few BC cycles will be required to transfer the data to the next stage.

5.2 Doublet correlator FPGA design

This section describes the design developed to test the doublet correlator. The design is based on the FCF tester design from [3].

5.2.1 Top level and ABC interface

The top level of the design connects the MicroBlaze (MB) softcore processor to the ABC interface.

The softcore processor is used for the communication with the LabVIEW application on the computer. This gives the possibility to integrate and modify easily the user control of the system. Further can the MicroBlaze be used to perform part of the characterisation tests and reduce the time needed by the test described in section 4.1.

The communication between the ABC interface and the MB is realized with an external peripheral controller (EPC) interface, which has a 32 bit parallel data bus and a 8 bits for the address. The address is used to write to the different registers or to execute a specific command. The ABC interface contains all the logic to communicate with the ABC130 chip together with the correlator logic. Figure 5.9 shows the different blocks described in the following sections.

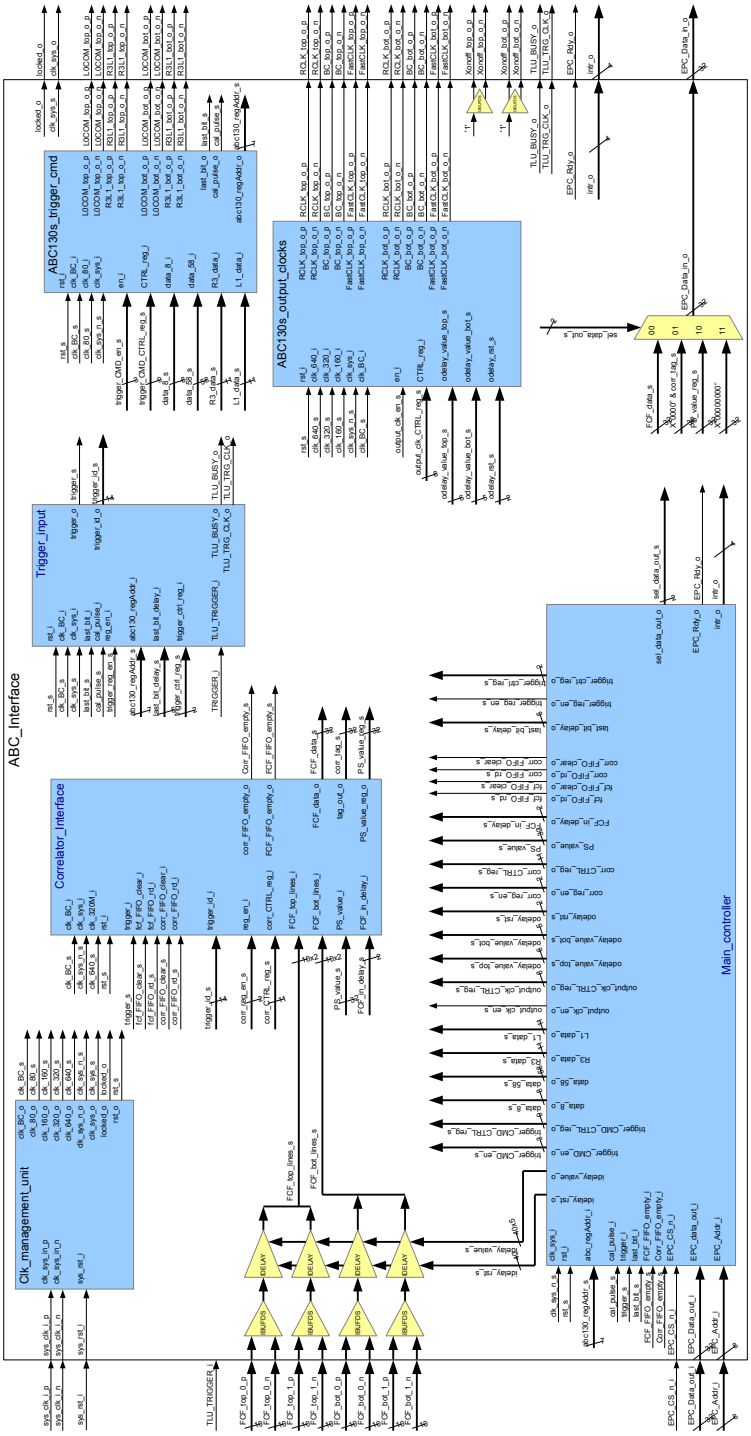


Figure 5.9: RTL schematic for the ABC Interface

5.2.2 Main controller

This block holds the logic to communicate with the MicroBlaze and generates all the control signals for the other blocks. It contains the register for setting the different delays and to select the read value. The delay register, shown in table 5.4, is used to define the ODELAY values, as well as delays on internal signal lines.

Delay register					**Address**		\$01
bit	18	17	16 - 12	11 - 7	6	5	4 - 0
	enable		ODELAY value		FCF data delay		last bit delay
	bottom	top	bottom	top	bottom	top	

- The *last bit delay* is used to set the amount of BC cycles between the transmission of the calibration pulse and the readout of the FCF value.
- The *FCF data delay* bits can be used to delay either the top or bottom data by 1 BC cycle. This could be used in case the value from the two sides are not read in the same BC cycle.
- The *ODELAY* values are used to apply an output delay on the fast clock lines. This allows to adjust the phase between the FastCLK and the BC clock, to prevent the problematic described in section 3.4.3. The values are updated when a '1' is written to the corresponding *enable* bit.

Table 5.4: Delay register to set delay values

Each FCF input line has an IDELAY to take into account the delays between the lines. These can be set by writing to eight registers, where each register holds the value for five IDELAY elements. Table 5.5a shows the bits of each register and table 5.5b lists which address corresponds to which IDELAY. Similar as for the ODELAY, the value set in the *IDELAY value* field is applied when a '1' is written to the corresponding *enable* bit.

A value is read in two steps. First write to the specific address to set the output multiplexer, then perform an EPC read at address \$00. Table 5.6 lists the addresses to read a specific value.

Another task of the main controller is to generate the interrupt signals for the MicroBlaze. There are four interrupt lines, which go all to a GPIO IP connected to the softcore processor. Table 5.7 shows the signifiaction of the different lines.

5.2.3 Clock management unit

The clock management unit is the source of all clock signals, except for the phase recovery in the FCF deserialization. One MMCM is used to generate all the different clock signals, including the clock for the MicroBlaze.

IDELAY register										**Address**	\$01
bit	31	30	29	28	27	26 - 25	24 - 20	19 - 15	14 - 10	9 - 5	4 - 0
	enable					unused	IDELAY value				
	E	D	C	B	A		E	D	C	B	A

(a) Bit functions for the IDELAY registers

Register	**Address**	**Description**
IDELAY register 0	\$12	top lines 0 for chip 0 to 4
IDELAY register 1	\$13	top lines 0 for chip 5 to 9
IDELAY register 2	\$14	top lines 1 for chip 0 to 4
IDELAY register 3	\$15	top lines 1 for chip 5 to 9
IDELAY register 4	\$16	bottom lines 0 for chip 0 to 4
IDELAY register 5	\$17	bottom lines 0 for chip 5 to 9
IDELAY register 6	\$18	bottom lines 1 for chip 0 to 4
IDELAY register 7	\$19	bottom lines 1 for chip 5 to 9

(b) List of register to set the IDELAYs

Table 5.5: Addresses and corresponding FCF lines for the IDELAY registers. The letters A to E in 5.5a correspond to the numbers defined in 5.5b

Register	**Address**	**Description**
FCF value	\$0D	read a value from the FCF output FIFO
Correlator value	\$0E	read a value from the correlator output FIFO
PS value	\$0F	read the current value set in the phase shift register
TLU value	\$1B	read a trigger value form the TLU output FIFO

Table 5.6: Read data register addresses

5.2.4 Correlator interface

The correlator interface is the most complex block and holds the logic to read and deserialize the FCF lines together with the correlation logic. Figure 5.10 shows the RTL block diagram.

There are two register to control the correlator interface. The first defines the functionality of the correlator and FCF readout logic and is described in table 5.8. The second register is used to set the phase shift value for the FCF readout and is shown in table 5.9.

Interrupt line	Description
line 3 - 0	“001” if the transmission of a command or trigger is completed “010” if a FCF value is ready to be read. “011” if a TLU value is ready to be read. “100” if a correlation tag is ready to be read. “110” if a error occurred while reading (no data in the FIFOs)
line 3	‘1’ if new data are available in one of the output FIFOs

Table 5.7: Interrupt signals from the ABC interface to the MicroBlaze. When line 3 is assigned, the MB has to check which output has new data.

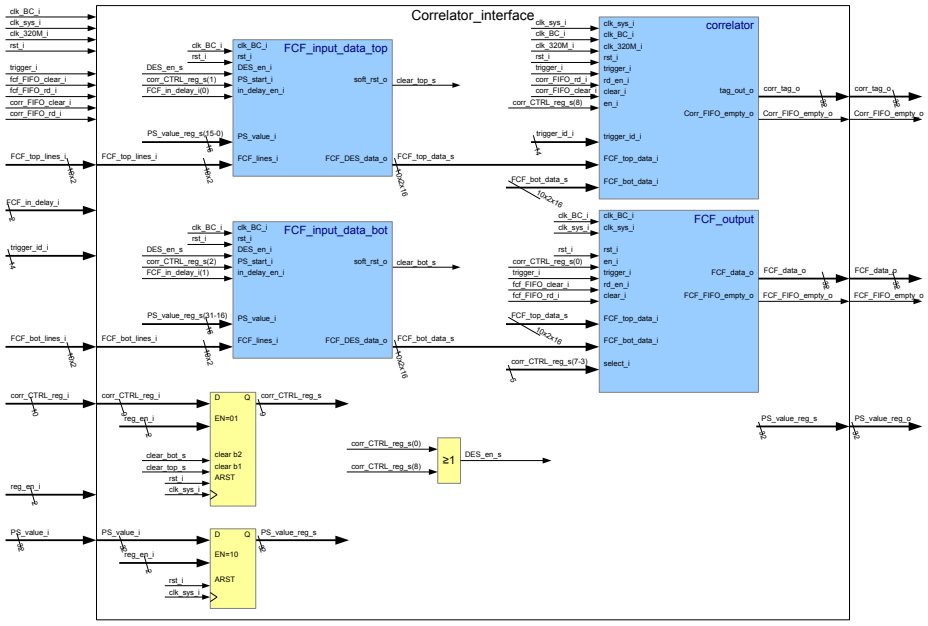


Figure 5.10: RTL schematic of the correlator interface

FCF input data

There are two identical blocks to read the FCF lines. One is for the top module and the second for the bottom module. The pseudo RTL schematic is shown in figure 5.11.

The *phase_shifter* block contains a MMCM which generates the clock signals to read the input lines, together with the control logic. This allows to apply a phase shift to take into account the delay from the lines.

For each chip of one side exist two 16 bit Serial In/Parallel Out (SIPO) register. They are shifted in with the FastCLK. The content of the SIPO is copied to a buffer on each BC cycle. The buffer synchronizes the data between the phase shifted clock and the system clock.

Correlator control register				**Address**		$0B
bit	8	7 - 4	3	2	1	0
	correlator enable	Chip select	select side	Phase shift start bottom	Phase shift start top	FCF output enable

- all bits '1' for active and '0' for inactive
- Setting the *Phase shift start* bits to '1' updates the phase shift for the FCF readout with the value set in the phase shift register (table 5.9). They are reset automatically.
- *FCF output enable* and *correlator enable* activate writing to the FCF FIFO and the correlation logic respectively.
- *Chip select*, the chip ID, and *select side*, '1' for bottom or '0' for the top module, control the multiplexer of the FCF readout.

Table 5.8: Correlator control register bits

Phase shift register		**Address** $01
bit	31 - 16	15 - 0
	Phase shift top value	Phase shift bottom value

Table 5.9: Phase shift register. The phase shift values have a resolution of $1/56 \cdot 1/f_{VCO}$, where f_{VCO} is defined in the instantiation of the MMCM. For the 320MHz design is $f_{VCO} = 960MHz$.

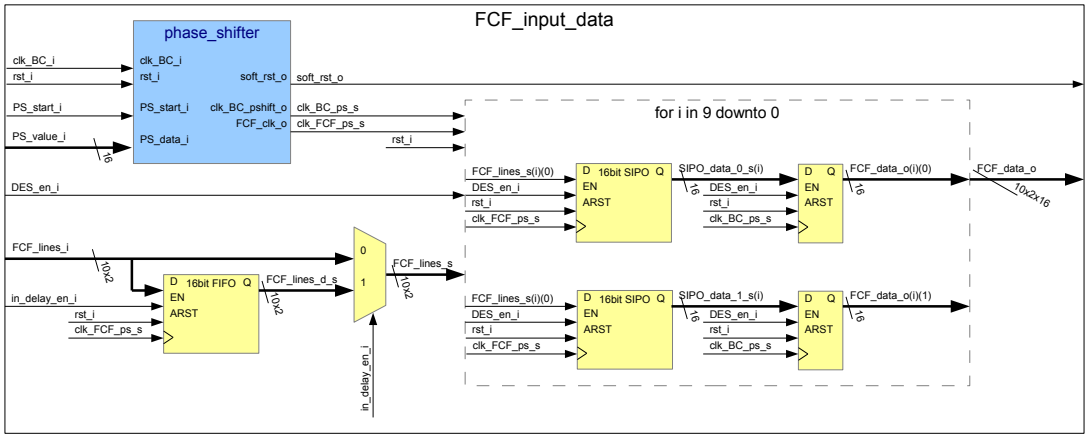


Figure 5.11: RTL schematic for the FCF input logic

Correlator

This block includes the complete correlation logic as described in section 5.1. The correlator block can be enable with a bit in the correlator control register.

FCF output

The FCF output is used to read the FCF values from the different lines. This block contains a multiplexer for the FCF lines and a FIFO where the FCF data are stored after a trigger signal.

5.2.5 Trigger input

This block contains the logic to generate the signal for the FCF readout together with an internal trigger counter. Further the interface to the TLU from the EUDET telescope is integrated in this block.

As described in section 5.2.4 is the read FCF data only stored in the FIFO, when a trigger signal is generated. The trigger input block generates an internal trigger signal, whenever a calibration pulse command is sent.

Another source for the internal trigger signal is the TLU. This trigger unit is used during the test beam and generates a trigger signal once a hit in the EUDET telescope is registered.

The trigger input logic provides also a trigger ID signal, which is added to each tag from the correlator. This allows to associate the tags to the trigger signal in the post processing. The ID is 14 bits wide and can be either the internal trigger counter or the ID transmitted from the TLU. The width was chosen to have no more than 32 bits for each correlator tag.

Table 5.10 shows the bits used to activate the TLU and to control the trigger input block.

Trigger control register			**Address**	\$1A
bit	2	1	0	
	enable TLU input	select ID source	reset internal trigger counter	

- *select ID source*: '1' for TLU and '0' for internal counter.
- *reset internal trigger counter*: executes when a '1' is written and is cleared automatically.

Table 5.10: Trigger control register

5.2.6 ABC130s trigger and command

The trigger and command block contains the elements to create the trigger signals and the command signals sent to the ABC130 chips.

A control register, shown in table 5.11, is used to start the different commands. The data to be sent for a R3 or L1 trigger or a command can be set by writing to a specific address.

ABC130 trigger and command control register							**Address**		$03
bit	8	7	6	5	4	3	2	1	0
	L0 enable	Cmd enable	L1 enable	R3 enable	select trg/cmd	bottom	top	Cmd size	start

Table 5.11: ABC130 trigger and command control register.

Because the EPC connection between the MicroBlaze and the ABC interface is only 32 bits wide, the long command data (58 bits) has to be written in two access, plus one to copy the 58 bits into the output buffer. Table 5.12 lists the register to be written to set the data.

Register	**Address**	**Description**
Short Command data	$04	write the short (8 bits) commands
Copy long command	$05	copy data for the long (58 bits) commands
R3 data	$06	write data for R3 trigger (8 bit)
L1 data	$07	write data for L1 trigger (8 bit)
Long command LSB	$08	write bits 0 to 31 of long command
Long command MSB	$09	write bits 32 to 57 of long command

– The data are always on the LSb of the EPC data lines.

Table 5.12: Data register addresses to set the ABC130 triggers or commands executed

5.2.7 ABC130s output clocks

This block manages the output clocks provided to the two hybrids. Each clock can be individually activated by the means of setting the corresponding bit in the output clock control register, shown in table 5.13.

Output clock control register					**Address**	$02
bit	5	4	3	2	1	0
	bottom BC	bottom RClk	bottom FClk	top BC	top RClk	top FClk

– All bits are zero active!

Table 5.13: Output clock control register bits

The ODELAY for the fast clock signals are also integrated within this block. The delay values are set in the main control register (see table 5.4).

5.3 Simulation of the logic

The complete ABC interface was simulated in ModelSim to verify the functionality. For this purpose, a testbench was written, which allows to generate the FCF input signals. Two specific sets of input data were used which were chosen to test all possible combinations of correlations. Tables 5.15 show the applied input hits for the test input.

Table 5.14 shows the correlations for the test inputs. Only the valid tags are listed in the table. All other tags should be invalid.

chip	distance	address	size	row	(hex)
0000	001	1110001	00	00	00F10
0010	110	0000011	10	00	0B038
0011	111	0000000	10	00	0F808
0011	001	1111101	00	00	0CFD0
0111	111	0000001	00	11	1F813
0111	101	0110110	00	01	1EB61

(a) Test input 1: Results

chip	distance	address	size	row	(hex)
0000	011	0101011	10	01	01AB9
0000	110	0101011	00	01	032B1
0011	001	0000000	00	00	0C800
0011	001	1111101	00	00	0CFD0
0011	001	1110001	00	11	0CF13
0100	110	0000011	10	11	1303B
0101	111	0000000	10	11	1780B
0111	111	0000001	00	11	1F813
0111	101	0110110	00	01	1EB61

(b) Test input 2: Results

Table 5.14: Expected results for the test input.

The expected result was obtained with the simulation. Figure 5.12 shows the waveforms for the test input 1. Only the FCF lines with data are shown, the others are always at the same level. Note that the FCF lines are by default at '0' instead of '1'. This is because the FCF lines on the setup get inverted in the connection from the support board to the interface board. For the correlation results also the valid bit is seen in the figure (the MSb of each tag). An additional non-valid tag is also shown at the bottom, which has the valid bit at '0'.

0		**1**		**2**		**3**		**4**		**5**		**6**		**7**		**8**		**9**	
H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L
11	11	FF	FF	**126**	**2**	FF	FF	127	127	FF	FF	FF	FF	74	74	FF	FF	FF	FF
113	113	FF	FF	3	3	0	125	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H
9		**8**		**7**		**6**		**5**		**4**		**3**		**2**		**1**		**0**	

(a) Test input 1: Row 0

0		**1**		**2**		**3**		**4**		**5**		**6**		**7**		**8**		**9**	
H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	1	1	FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	1	54	FF	FF	FF	FF	FF	FF
L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H
9		**8**		**7**		**6**		**5**		**4**		**3**		**2**		**1**		**0**	

(b) Test input 1: Row 1

0		**1**		**2**		**3**		**4**		**5**		**6**		**7**		**8**		**9**	
H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L
83	86	FF	FF	124	124	127	127	FF	FF	FF	FF	FF	FF	74	74	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	0	125	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H
9		**8**		**7**		**6**		**5**		**4**		**3**		**2**		**1**		**0**	

(c) Test input 2: Row 0

0		**1**		**2**		**3**		**4**		**5**		**6**		**7**		**8**		**9**	
H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L
FF	FF	FF	FF	FF	FF	11	11	**126**	**2**	FF	FF	1	1	FF	FF	FF	FF	FF	FF
43	43	FF	FF	FF	FF	113	113	3	3	0	0	FF	FF	1	54	FF	FF	FF	FF
L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H
9		**8**		**7**		**6**		**5**		**4**		**3**		**2**		**1**		**0**	

(d) Test input 2: Row 1

Table 5.15: Inputs for the correlator testbench. A channel number in **bold** means a 2 hit cluster. "FF" is an invalid cluster.

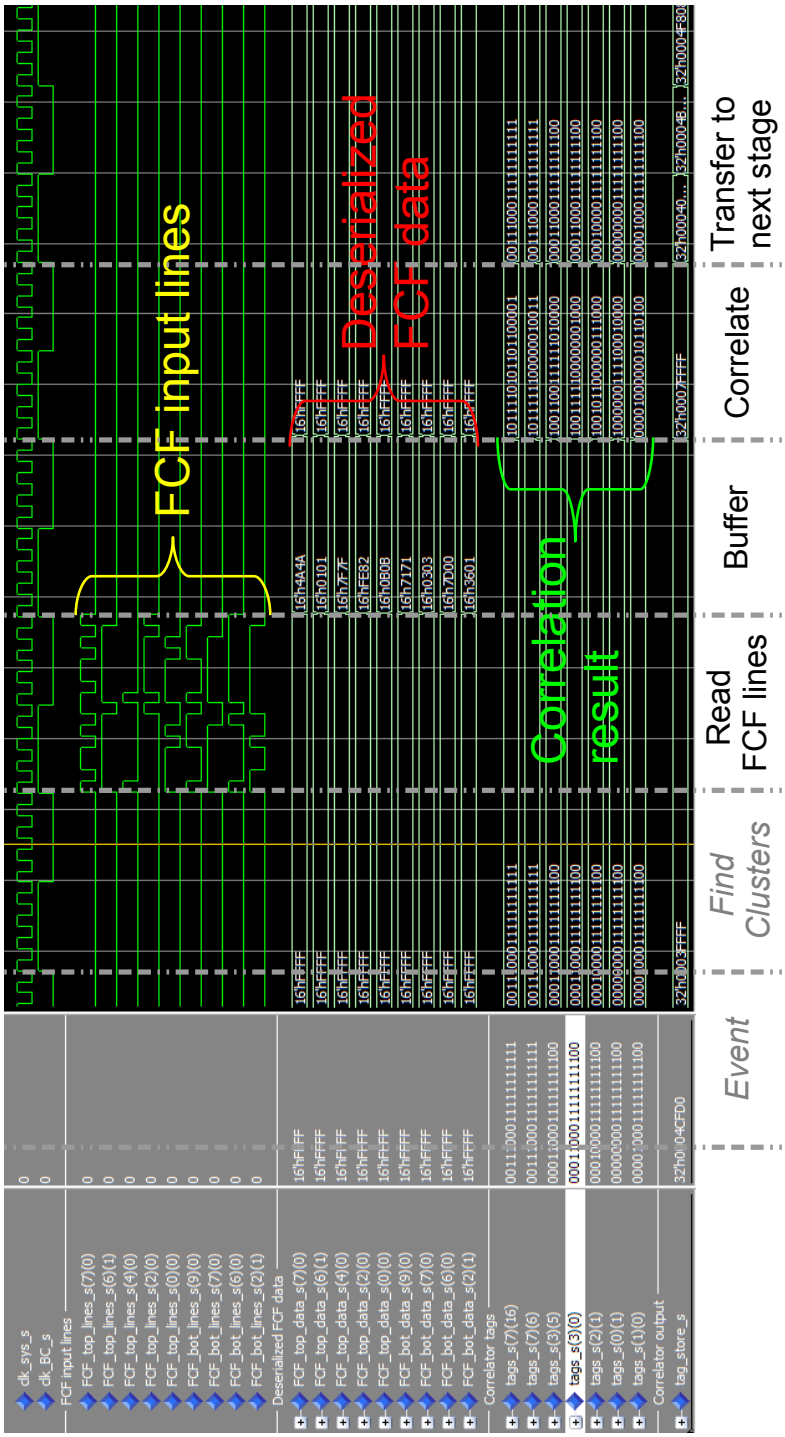


Figure 5.12: Results from the simulation of the correlator design.

5.4 Softcore program

A new program for the MicroBlaze (MB) processor was written to test the correlation logic and to have a simpler interface, that simplifies the integration of new functions.

5.4.1 Main function

The program is based on interrupts, which can come from either the serial port or from the ABC interface. Figure 5.13 shows the basic main loop.

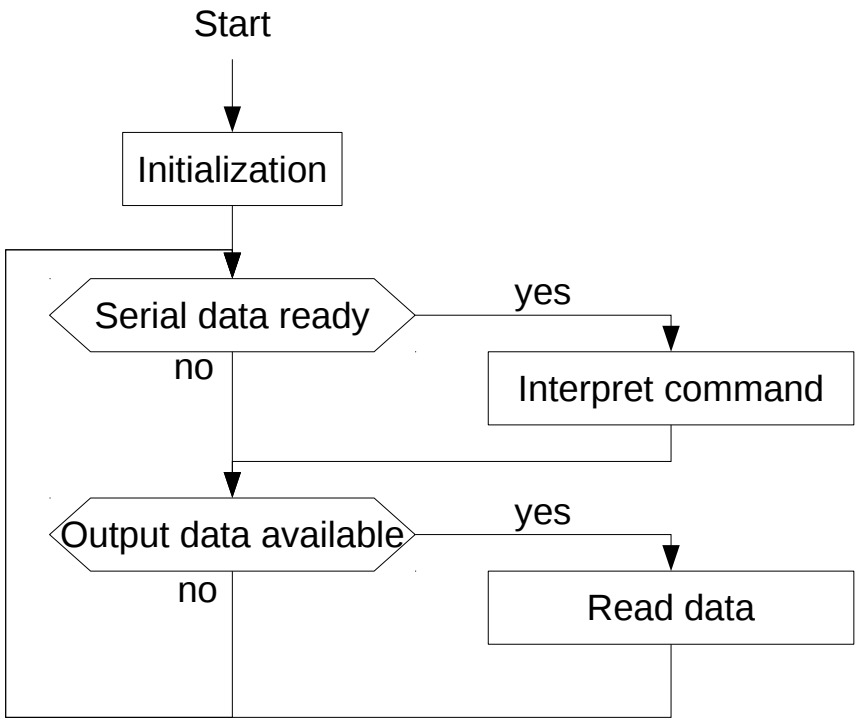


Figure 5.13: Flow chart for the main function of the MB program.

The `Interpret command` function reads the data from the serial port and decodes the command to be executed. Section 5.4.2 describes the format of the commands and how they were implemented.

The `Read data` function reads the data from the different output FIFOs in the ABC interface. There are two different output formats implemented. A direct and an event based output as described below.

Direct data output

In the direct data output, all the data coming from the different sources are sent to the computer as soon as received. There are three possible string returned as described in table

5.16

Source	Format	Description
TLU	`acktXXXXXXXX\n\r`	Data from the TLU. Bits 31-18: trigger ID, bits 17-5: TDC value, bits 4-0: TDC clock bin
Correlation	`ackcXXXXXXXX\n\r`	Correlation output tag as described in table 5.3
FCF	`ackfXXXXXXXX\n\r`	data from the FCF deserializer. Bits 31-16: FCF line 2, bits 15-0: FCF line 1

- XXXXXXXX: stands for a 32 bit number, transmitted in ASCII hex (0-F) characters.
- `ack` and `\n\r` are used to mark the size of one data package.

Table 5.16: Formats for the data returned in the direct output mode

Event data output

The event data output waits until a trigger from the TLU is received, before the data are sent out. The correlations are read from the ABC interface as soon they arrive and are stored in an array. Once a new trigger information is available and no correlations are in the output FIFO, the trigger information is sent together with all stored correlations. Table 5.17 shows the format, how the events are transmitted. Depending on how many correlations are found in one event, the length may variate.

Trigger info	Correlations	End
`ackeXXXXXXXX`	`YYYYY...`	`80000\n\r`

- XXXXXXXX: stands for a 32 bit number, transmitted in ASCII hex (0-F) characters.
- YYYYY: stands for a 20 bit hex number. There may be up to 40 correlations sent in one event.
- `ack` and `\n\r` are used to mark the size of one data package.
- There is no space between the different fields.

Table 5.17: Format for the data returned in the event output mode

5.4.2 MicroBlaze commands

To control the MicroBlaze, a simple protocol over the serial line was introduced. Each command sent to the MB consists of two ASCII characters. Depending on the command more characters have to be transmitted which correspond to the input value.

The MB responds to each command with an acknowledge, which can be one of the possibilities listed in table 5.18. Depending on the command, the response can contain some data. The data will be sent after the `ack` string. To mark the end of the return value, a *line feed* (`\r`) and *new line* (`\n`) is sent.

Ack type	Description
`ack`	command execution ok
`err`	error in execution
`mis`	not enough data in payload
`ukn`	unknown command

Table 5.18: Acknowledges return by the MicroBlaze after a command

Table 5.19 lists some of the implemented commands. There are more commands implemented, but those were used for testing or debugging and are documented inside the code. The full code is in the digital appendix. The most important and complex commands are described in detail below.

Configure phase shift

For a correct read in of the FCF signal, the phase shift input clock has to be adjusted to the system. A command was written to perform this task in an automated way. The command for the phase shift is `ps` and the input parameter can be found in table 5.19. Each side is always configured individually. The input `m` it is possible to select which side or if both sides shall be configured one after the other with the same initial chip number.

The address of the initial chip is given as the parameter `a` to the function. This chip will be used for the phase shift calibration and should be the chip with the longest path delay on the hybrid. All the other chips will get their IDELAY set according to the found phase shift value.

The calibration consist of three steps which are executed for each side:

1. Coarse phase shift set with the framing mode
2. Fine phase shift trim
3. IDELAY adjustments

Command	Input	Description
`fw`	`rrxxxxxxxx`	Write FPGA register.
`aw`	`maarrxxxxxxxx`	Write ABC130 register.
`rs`	`m[f|s|l|b|S|o]`	Send reset command. Following reset can be used: f = FCF, s = System, l = L0ID, b = BC, S = SEU, o = soft
`ps`	`map`	Configure phase shift. Add `p` to print IDEALY values. See section 5.4.2
`cc`	`[m|0|1|*]maa`	Configure chip to masked mode (`m`), set all mask register to zero (`0`) or one (`1`), else set to operational mode with all channels masked.
`av`	`Macccccccdddd`	Send multiple calibration pules and count how many times the correct value was read on the FCF line. See 5.4.2 for more details.
`cp`	`maa`	Send calibration pulse
`tf`		Read a value from each FCF line.
`hh`		Print list with all commands implemented.

Common values for input data:

- `r` = register address in hex (8 bits)
- `x` = register value in hex (32 bits)
- `a` = ABC130 address in hex (4 or 8 bits)
- `m` = module side (1 = top, 2 = bottom, 3 = both)

Table 5.19: List of MicroBlaze commands

Coarse phase shift set: Before starting the configuration, the phase shift is set back to 0 as well as the IDELAY of the initial chip. This is requeried for a correct reference.

In the first step, the initial chip is set to the framing mode. The FCF value is then read and based on how many times the bits have to be shifted, an estimate of the required phase shift is calculated.

By increasing the phase on the read-in clock, the read value gets shifted to the left. The coarse phase shift step estimates the required phase shift, by counting how many times the read value has to be shifted until the correct value is obtained. To illustrate the method, consider the following example:

The value `0x03FC` is read on the FCF line. The value should read `0x00FF` to be correctly aligned, what requires 14 times a rotation to the left. By multiplying with the FastCLK period, a first estimate of the required phase shift value is obtained.

Fine phase shift trim: The find a more precise phase shift value, the initial chip is set into operational mode and the mask register is set to a 2 hit cluster on channel 42 for both rows to create an output of `0xAAAA` on both FCF lines. This pattern was chosen to have a change on every bit of the read value.

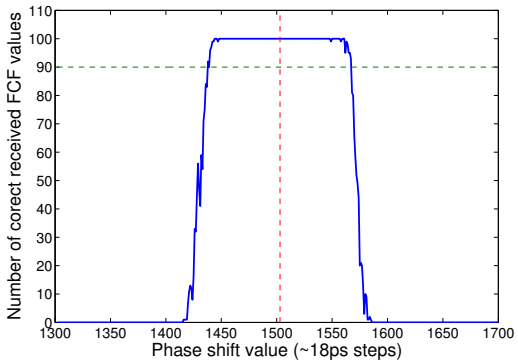


Figure 5.14: Results from a phase shift scan with 100 samples per value. The green line indicates the 90% limit which is used to define the range. The red line marks the center of the range, which will be used as the phase shift value.

The function scans then over a range of phase shift values, with the estimate from the coarse search as a base. By reading multiple times for each step a range can be found, where the phase shift is correct. Such a scan is shown in figure 5.14. The selected value is in the center of the range, where the number of samples correctly read exceed 90%.

IDELAY adjustments: The last step is to adjust the IDELAYs of the other FCF lines. This is done in a similar way as for the fine phase shift trim. For each FCF line, the same pattern is applied through the mask register. A scan is then made over the IDELAY value and again the middle value in the 90% range is applied to the line. The IDLEAY scan is done for each of the remaining 9 chips.

Average function

The average function is used to sample up to eight channels at the same time. The channels have to be four pairs of two consecutive channels.

This function was implemented to speed up the test procedures for the characterisation of the ABC130. It takes the following arguments:

- `M` = module side (0 = top, 1 = bottom)
- `a` = ABC130 address in hex (0 to 9)
- `c` = 4 times a 7 bit address (0..127) as 2 hex values, where the lower address of each pair is given. The first two addresses are for row 0 and the second two for row 1.
- `d` = 4 digit decimal number of how many samples to be taken.

The selected ABC130 chip is then configured to have all channels masked, except the selected 4 pairs. The given number of calibration pulses are then sent and the read value on the FCF line is verified. For each of the eight channel a separate counter measures how many times it was hit.

The values of the counters is then return in a comma separated list, in the same order as the channels were given.

Below follows an example of how the execution of an average command could look like in the console on the LabVIEW interface. The channels set are 32,33,47 & 48 on row 0 and 10,11,101 & 102 on row 1. Selected is the chip 3 on the bottom side for 100 samples.

```
-->av13202F0A650100>
ack75,83,82,69,71,82,85,76,
```

The return value means that on row 0 following number of hits were observed: ch. 32 = 75 hits, ch. 33 = 83 hits, ch 47 = 82 hits, ch. 48 = 69 hits. And on row 1: ch. 10 = 71 hits, ch. 11 = 82 hits, ch. 101 = 85 hits, ch. 102 = 76 hits.

5.4.3 Code structure

This section describes how the C program was written and how it was split into different modules.

main.c/.h: Contains main function as described in section 5.4.1.

uart.c/.h: Definitions and functions for the communication over the serial port.

platform.c/.h: Initialisation and definition of the MicroBlaze system and all peripherals.

globals.c/.h: Helper function for data type conversions and global definitions.

isr.c/.h: Interrupt handler function and definitions and access to interrupt flags.

FPGA_control.c/.h: Definitions and functions to access the abc interface logic on the FPGA.

abc130_control.c/.h: Contains all definitions and functions to control and configure the ABC130 chips.

commands.c/.h: Definitions and functions for the interpretation of the serial commands. All commands are have a specific function and are listed, together with the help text, in the arrat `cmds`. The main interpreter function, `decode_command` searches through the `cmds` array upon receiving a command and with the use of pointer to function calls the specific interpreter for the received command.

5.5 Doublet correlator demonstrator

To demonstrate the principle of the self-seeded triggering, a demonstrator doublet was built. This section describes the mechanical part fo the demonstrator. A description of the assemlby process is given in appendix B. The mechanical components were designed by Sergio Diez-Cornell and built during this thesis by Carl Haber, Haichen Wang and myself. The wirebonding was done by Rhonda Witharm.

5.5.1 Doublet test frame

The frame is an aluminium construction and holds the two support boards. Each board is mounted on an anodized aluminium plate for support. A hole is cut where the sensor will be inserted. To connect the HV bias the sensor, is glued to a small non-anodized frame. Figure 5.16 shows how the frame holds the different elements.

One board is fixed to the supporting frame, where as the other has the liberty to move in one horizontal direction. A micrometer allows to push the mobile side in a precise way. Two springs are attached between the plate and the frame, to pull the mobile plate back.

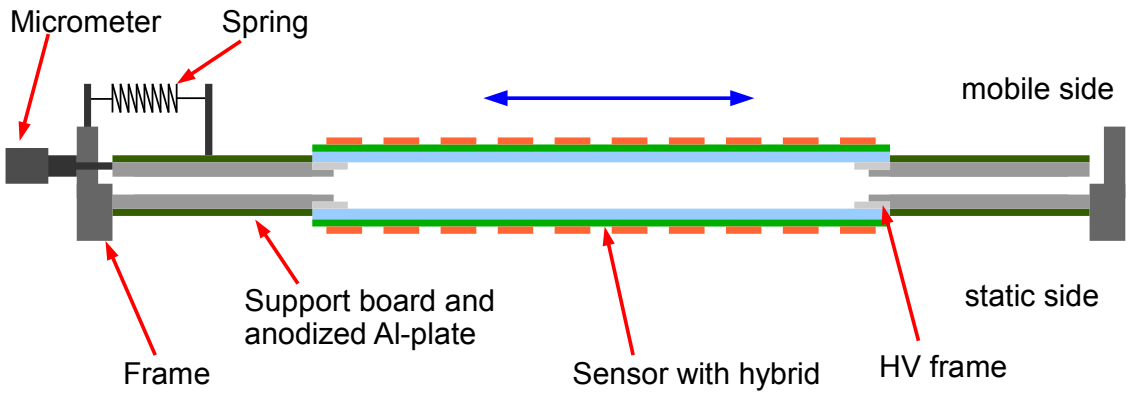
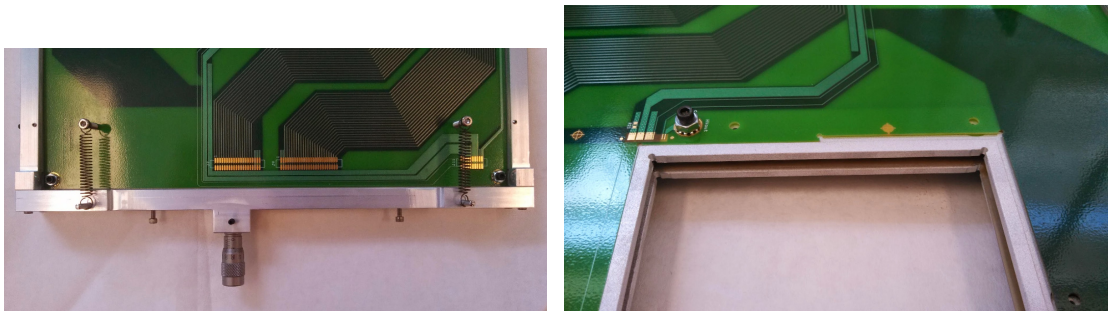


Figure 5.15: Sketch of the assembled doublet. It represents a cut through the middle of the sensor. The micrometer pushes the mobile part to the right, while the springs pulls it back to the left.

The assembly of the doublet is described in appendix B. Figure 5.18 shows one side of the complete doublet.

5.5.2 Wirebonding of the sensor

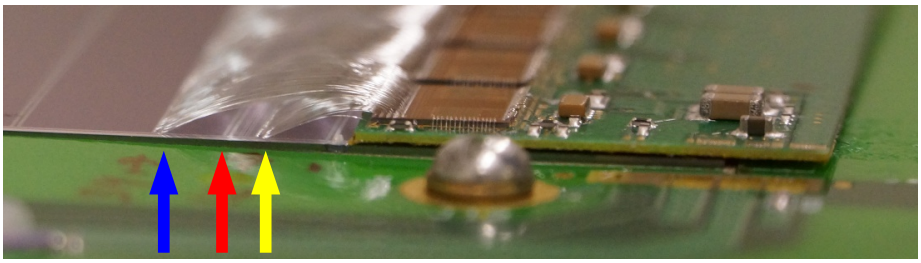
For the doublet demonstrator a ATLAS07 sensor was used. This sensor is described in section 2.4.2. The ATLAS07 sensor was not designed for use with the ABC130 and therefor good care had to be taken, how the bonds were made. There were three rows of pads available for the bonding, as indicated in figure 5.17a. The pads indicated with the red arrow had the problem, that they were misaligned as can be seen in figure 5.17b.



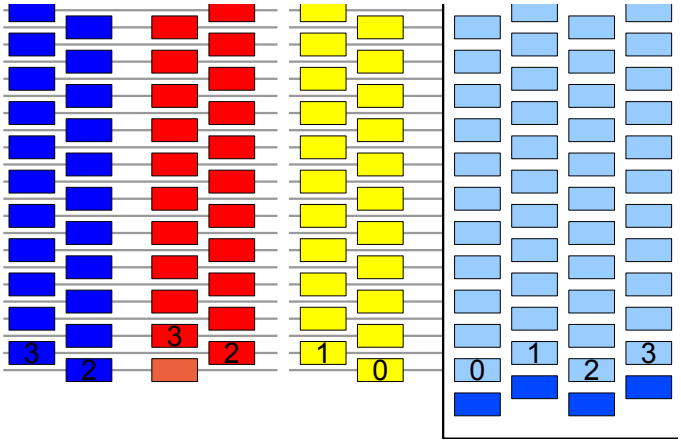
(a) Springs mounted on support board and frame. Micrometer to set position

(b) Close up picture of the HV frame as a support and contact for the sensor bias voltage.

Figure 5.16: Detail pictures of the unfinished doublet frame



(a) Side view of the wirebonds from the sensor to the ABC130 chips. The blue and red arrow are pads for row 0, and the yellow arrow is for row 1. Row 0 was bonded to the blue pads on the first sensor and to the red pads on the second.



(b) Schematic view of the pad layout (not to scale). The colors match the pad rows indicated by the arrows. As can be seen, the red row cannot be bonded so that the pads line up on both rows. To use this row, all connections have to be shifted by one strip.

Figure 5.17: Picture of the wirebonds and the schematic view of the pad layout.

The bonds to the blue row were too long and many problems occurred during the bonding. Several of the bonds weren't kept straight and can be touching the neighbors. Because of this, we decided to use the red row for the second sensor. This gave much better results in the bonding, but added an offset of one strip to row 0, as explained in the figure 5.17b.

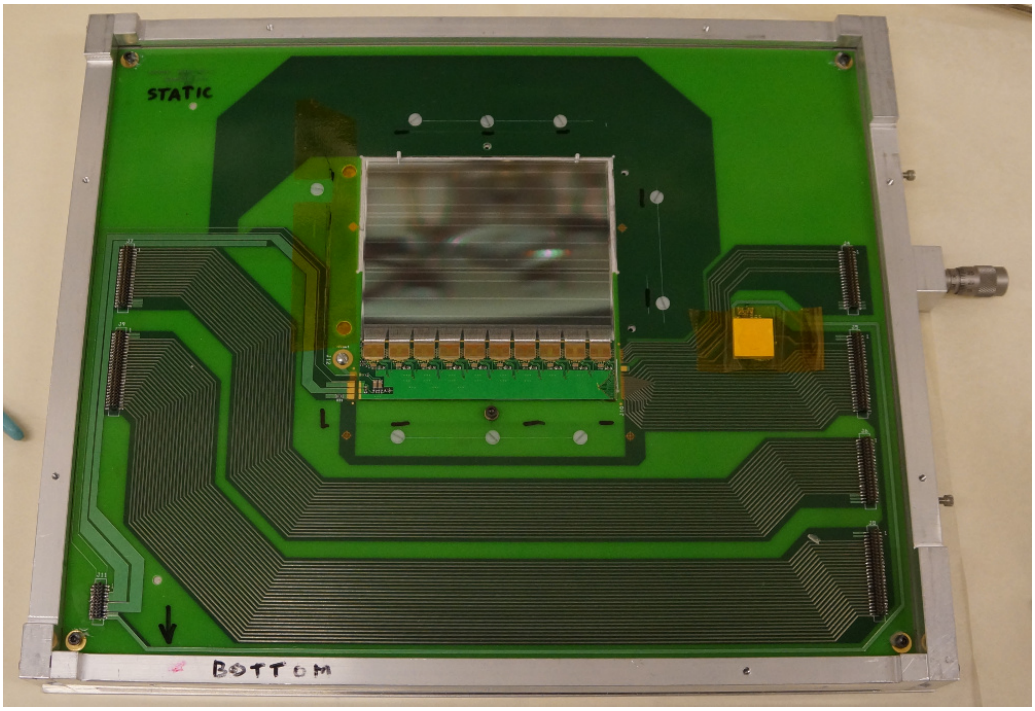


Figure 5.18: The doublet completely mounted

5.6 Data acquisition software

The data acquisition and analysis software consist of multiple parts. Multiple LabVIEW applications, called virtual instrument (VI), are used to collect the data generated by the correlator demonstrator. Figure 5.19 shows how the different parts interact together.

ROOT[2] is a C++ framework developed and maintained by CERN. It provides functions to handle and analyze large data sets in an fast and efficient way. Further it integrates many possibilities to represent the data in plots. To store and access the data ROOT uses *.root* files.

The silicon tracker data acquisition software (SCTDAQ) is a software package originally developed for testing the electronics of the semiconductor tracker. It consists of precompiled libraries and macros for ROOT. It is maintained by the ATLAS group at RAL and was since the use at the current ATLAS detector further developed. Now it includes many

[2] see *http://root.cern.ch/*

macros to test the ABC130 chip. It does not yet integrate the FCF readout and therefor an interface had to be created, to reuse the analysis macros from SCTDAQ.

For this purpose Jonas Stalder wrote a library used to create *.root* files from LabVIEW [15]. The created *.root* files can then be used by macros from the SCTDAQ or directly in ROOT itself.

I developed the different VIs to used to read the data from the demonstrator.

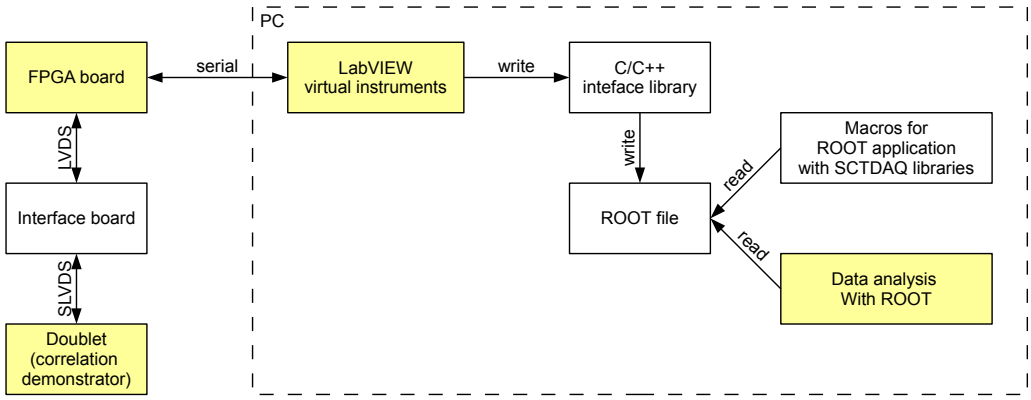


Figure 5.19: Overview over the data acquisition and analysis software. The yellow colored boxes indicate the elements where I contributed partially or completely in this work.

5.6.1 Correlator tester

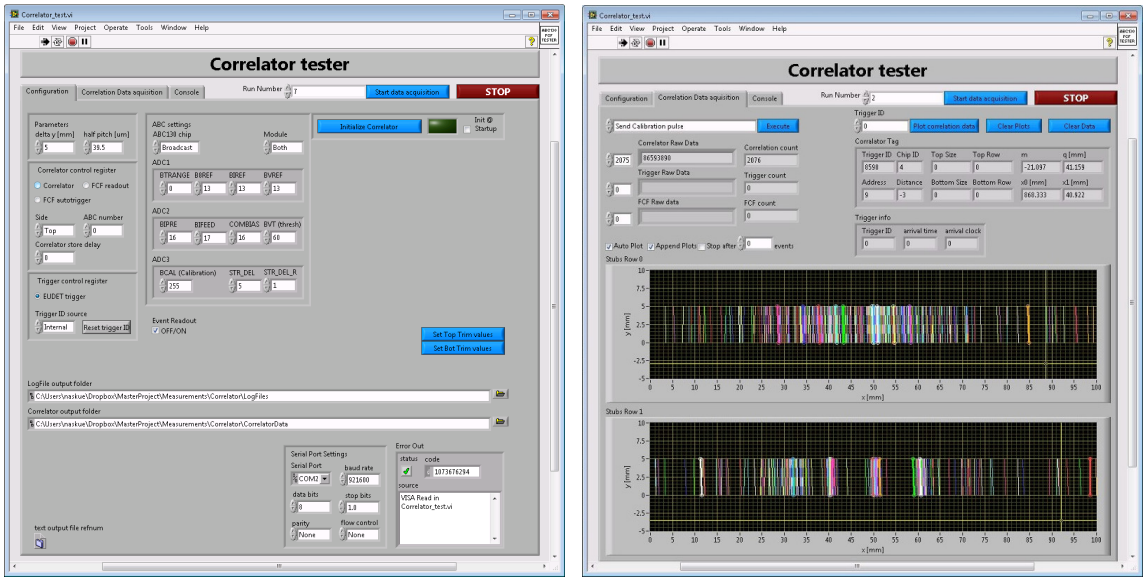
The correlator tester is the main VI to configure the FPGA and acquire the correlations generated. It communicates with the MicroBlaze with a serial line. It implements further functions to configure the ABC130 chips.

The correlations read are stored in a *.root* file for further analysis. Additional trigger information read from the TLU is written to a text file.

Figure 5.20 shows the main view for the data acquisition. The user manual in the appendix C explains in more details the whole data acquisition software. To execute commands on the MB a console is implemented in the correlator tester VI (see figure C.5 in the appendix).

The principal functions available in the correlator tester are:

- Sending commands to the MB.
- Configure the correlation logic of the FPGA.
- Configure the ABC130 ADC registers.
- Start a run to acquire correlations.
- Plot the correlations directly on the user interface.
- Save the correlations in *.root* files for offline analysis.



(a) The configuration view to set different options.

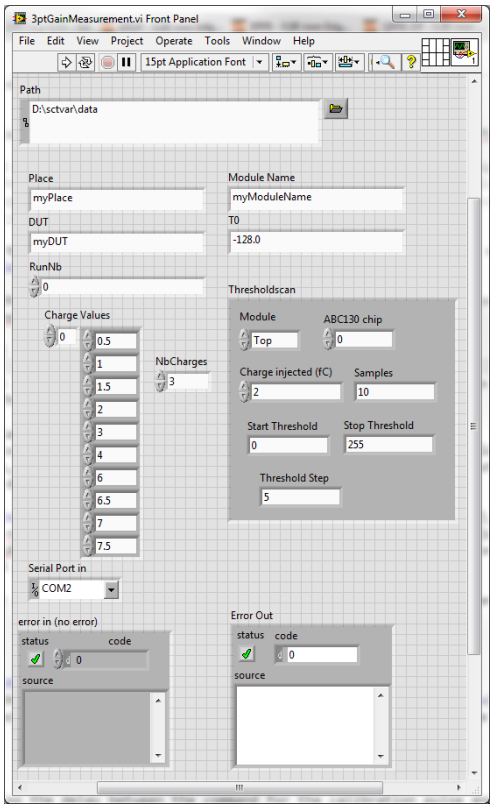
(b) The data acquisition view with recorded correlations.

Figure 5.20: Screenshots of the correlator tester user interface.

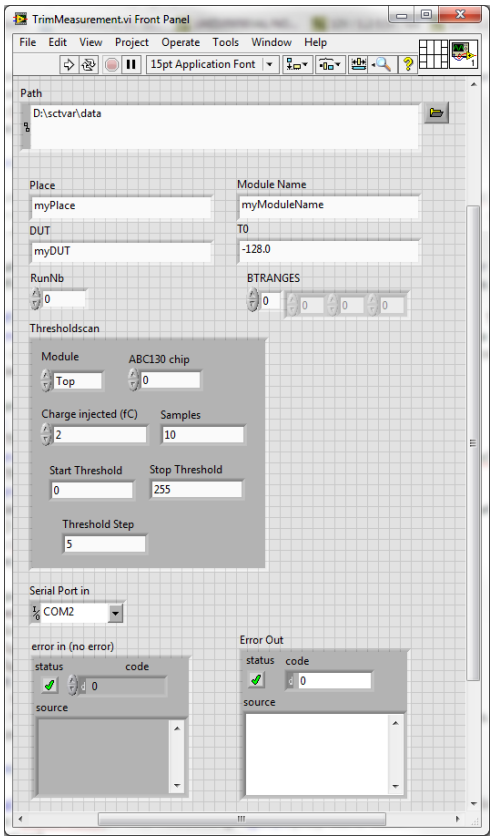
5.6.2 Characterisation test applications

A set of VIs was developed to perform the different characterisation test described in 4.1. The following VIs were made:

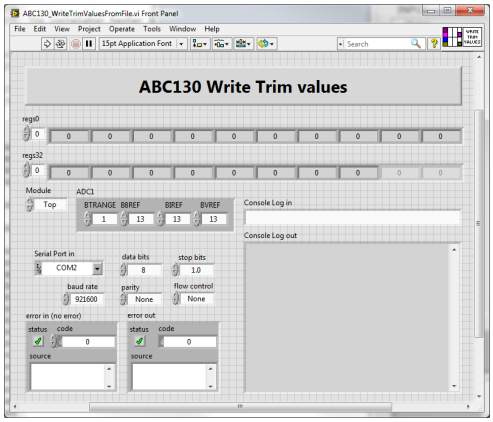
- **Response curve measurement**: VI that allows to perform a response curve. It's possible to define the used charges and therefore allows to perform a 3 point gain too. Shown in figure 5.21a.
- **Trim range measurement**: VI which scans over the trim range and stores the results in a *.root* file. The output file can be used in the macro *FCF_ABC130TrimRangePlot.cpp*. See [15, section 9.2.1.2] for the use of the macro. This VI is shown in figure 5.21b.
- **Write trim file**: VI to write the trim file values to the ABC130 chips, which generated by the macro *FCF_ABC130TrimRangePlot.cpp*. This VI, shown in figure 5.21c, needs to be corrected as described in section 6.2.2.
- **Threshold scan**: VI to perform a single threshold scan on 4 channel pairs, see figure 5.21d



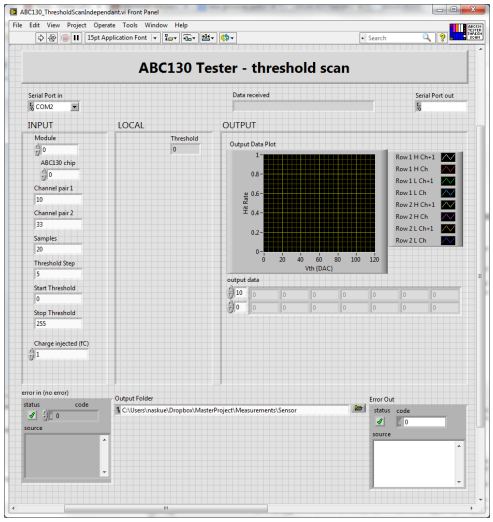
(a) The data acquisition view with recorded correlations.



(b) Trim range measurement VI



(c) Write trim file



(d) Threshold scan VI

Figure 5.21: Screenshots of the correlator tester user interface.

6 Noise and gain measurements

A brief characterisation of the ABC130s was performed by executing the tests described in section 4.1. In an early stage of the project, only a few channels of one chip were tested. After the test beam, all channels and both modules built were measured and analyzed.

A more complete characterization of the ABC130 was done in the bachelor thesis of Jonas Staler [15], performed during the same time period as my work.

The strobe delay scan was performed at the beginning and the found value was then applied during the measurements described in this chapter.

6.1 First characterizations with FCF

After establishing the correct FCF readout system, some tests were performed to verify the functionality and characteristic of the chips. These test were performed with the FCF tester LabVIEW application described in section 4.3.2. The chips weren't bonded yet to a sensor at the time these measurements were performed.

The response curve for the tested channels is shown in figure 6.1. For low charges the channels are relatively close together. A wider spread in the V_{T50} is observed at higher

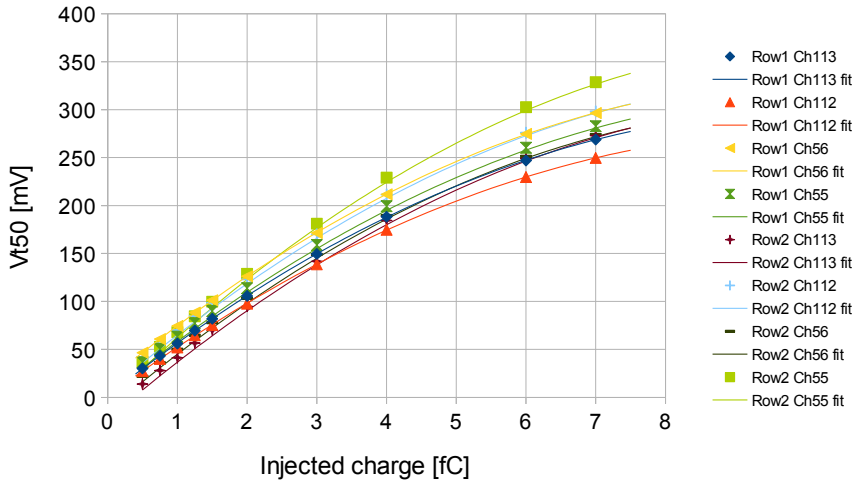


Figure 6.1: ABC130 chip 0 response curve of 8 channels (113, 112, 56 and 55 on both rows) measured through the FCF readout. 100 samples per threshold value taken. Marks are the measured points and the thin lines represent the fitted curves.

charge values. The design of the ABC130 input stage was focused to have a linear gain for low charges (around 1 to 2 fC) where the threshold will be set. For higher charges, the amplifier becomes non-linear. This does not influence the results, because any charge in the non-linear region will be seen as a hit.

To obtain the gain, a quadratic fit is performed on the measured values for the response curve and the derivative is taken.

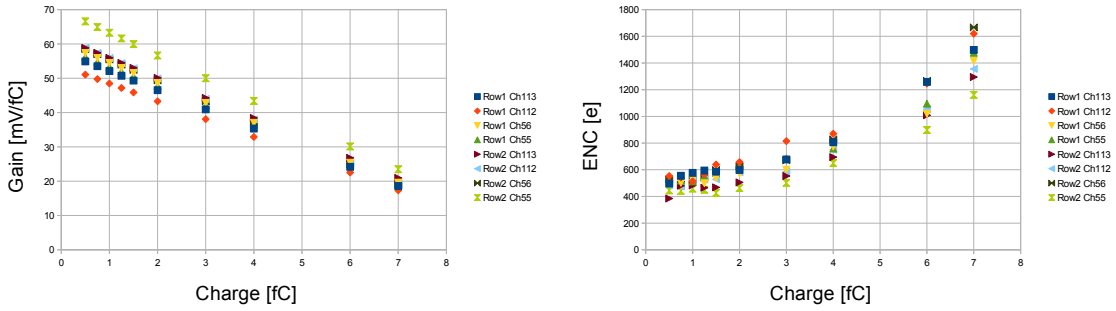
$$V_{T50} = a \cdot q^2 + b \cdot q + c \tag{6.1}$$

where q is the injected charge. a, b and c are the parameters from the fitting and will be used to calculate the gain:

$$Gain = 2a \cdot q + b \tag{6.2}$$

The quadratic fit is only one method. SCTDAQ uses an exponential fit for the response curve. In case of the 3 point gain test, a linear fit is done by SCTDAQ. The obtained gain is shown in figure 6.2a. It can be observed, that the gain is largest for small charges. This is again from the design, which is optimized for small charges.

The equivalent noise charge (ENC) is obtained by dividing the output noise with the gain at the corresponding injected charge. Figure 6.2b shows the obtained input noise. Due to the smaller gain, the observed noise is much larger for high charges.



(a) Gain as function of injected charge

(b) ENC as function of injected charge

Figure 6.2: ABC130 chip 0 gain and ENC for channels 113, 112, 56 and 55 of both rows. Calculated from response curve in figure 6.1

Reference results were presented by Bruce Gallop at the Valencia ATLAS strip meeting [24]. Those tests were performed on some ABC130s, which were corrected by means of FIB (see section 3.5).

Overall it can be stated, that I obtained similar values for the gain. The noise is with 500e$^-$ approximately 100e$^-$ higher.

6.2 Full characterisation tests

The correlator test system includes also the average function to perform multiple charge injections (section 5.4.2). This function allowed to speed up the calibration tests and made it possible to perform a test on all 10 chips of a module within a reasonable time.

LabVIEW applications, called VIs, were created in order to perform the characterisation tests. These applications store the acquired data in ROOT files, which can then be analyzed in the SCTDAQ. See section 5.6 for a description of the data acquisition software.

6.2.1 3 point gain

A 3 point gain was performed on both modules from the doublet. Figure 6.3 shows the V_{T50} point, the gain and the input noise for all channels of chip 5 to 9 from the bottom module. The macros producing the plots were reused from the test of the ABC250 chips. As it was understood, the plots do not correspond to the rows, but to all pins of 5 ABC130s.

The gain is around 70 mV/fC, which is similar as measured at the beginning of the project. But it is smaller than observed with the single chip board which was around 80 mV/fC [15, Chapter 11]. Some chips seam to have a lower gain around 65 mV/fC. In case of the figure above, this is chip 6 and 7 (channels 256 to 767).

The output noise is around 750 electrons. This is also higher than observed in the single chip board where around 550 electrons were observed. The increased noise is expected and can be explained with the larger strips from the sensor. The single chip board was connected to a mini-strip sensor with strips 1 cm long, while the doublet has 2.5 cm long strips.

6.2.2 Trim range

Because the variation of the V_{T50} from channel to channel is quite large, a trimming is applied. This is done by writing to the trim value registers in the ABC130. A VI was written to perform measurement over different trim values. The result from this measurement can then be applied to a ROOT macro to obtain a trim file.

The current application of this trim file is incorrect and the channels don't get trimmed as expected. Figure 6.4 shows the V_{T50} per channel for the same chips as in figure 6.3. It can clearly seen, that the variations between the channels didn't decrease as it should.

6.2.3 Response curve

The response curve was performed on the static side of the doublet. The gain observed is as for the 3 point gain. At low charges, more bad channels were observed. Actually not all channels had a correct S-curve for a charge of 0.5 fC, i.e. the rate was not at 100% at a threshold of 0. The measured response curve of chip 0 is shown in figure 6.5.

The response curve measured on the doublet is similar to the one measured on the single chip module (represented in figure 6.6). The noise measured on the doublet is much larger than for the single chip module. This can be explained by the longer strips of the doublet. The noise increases with to the detector capacitance C_{ox}[14, section 3.10.1]. The longer the

strip the larger this capacitance becomes. The doublet has strips of 2.5cm while the single chip board is connected to a mini sensor with 1cm long strips.

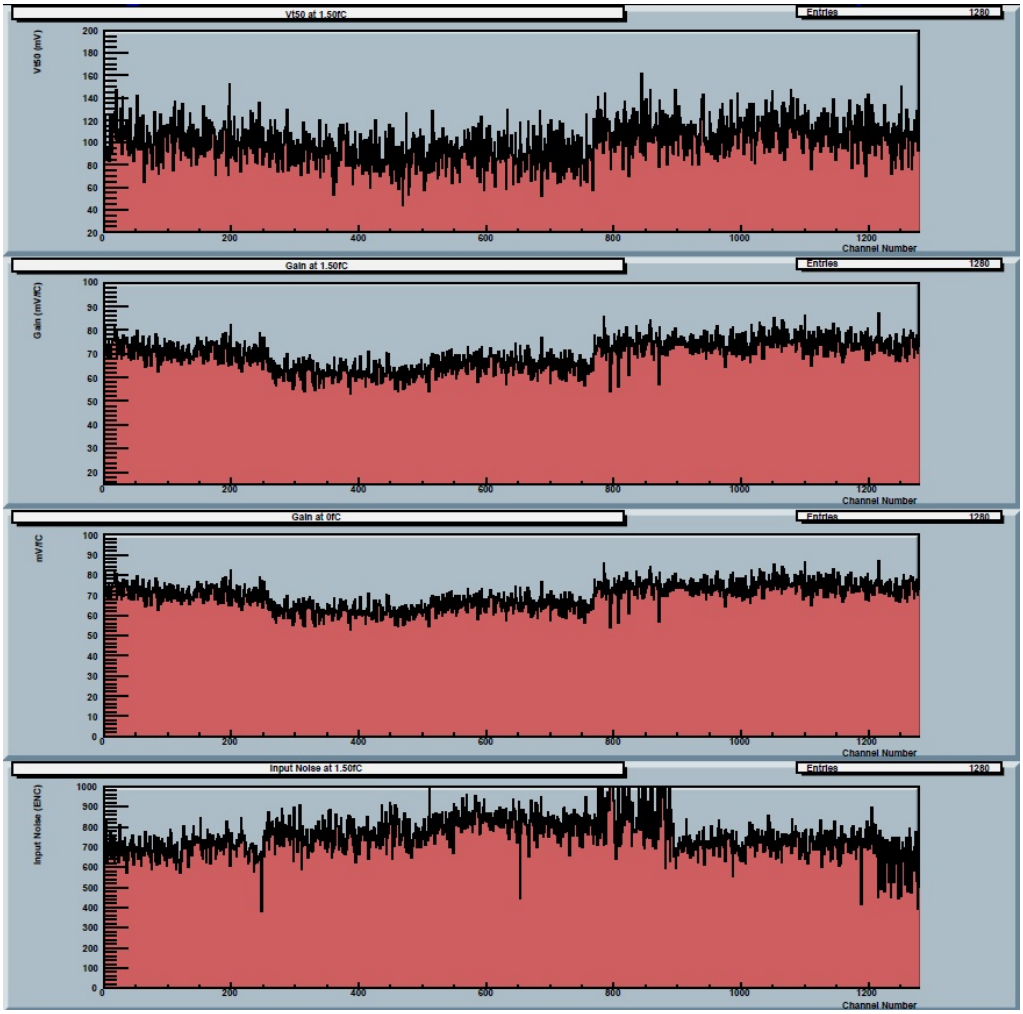


Figure 6.3: V_{T50}, Gain and input noise for each channel of chips 5 to 9 of the untrimmed bottom side module at a injected charge of 1.5 fC.

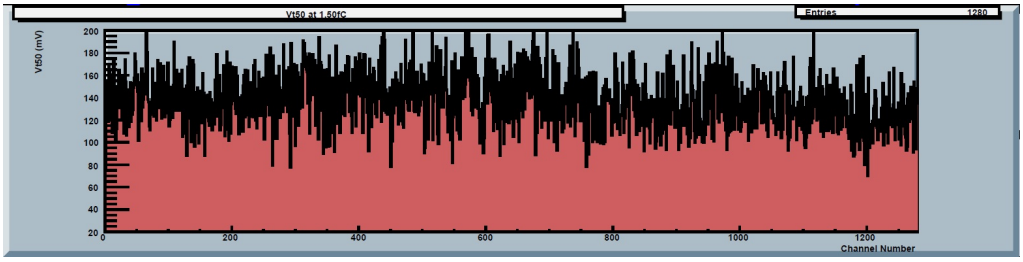


Figure 6.4: V_{T50} for the wrong trimmed module at 1.5 fC injected charge.

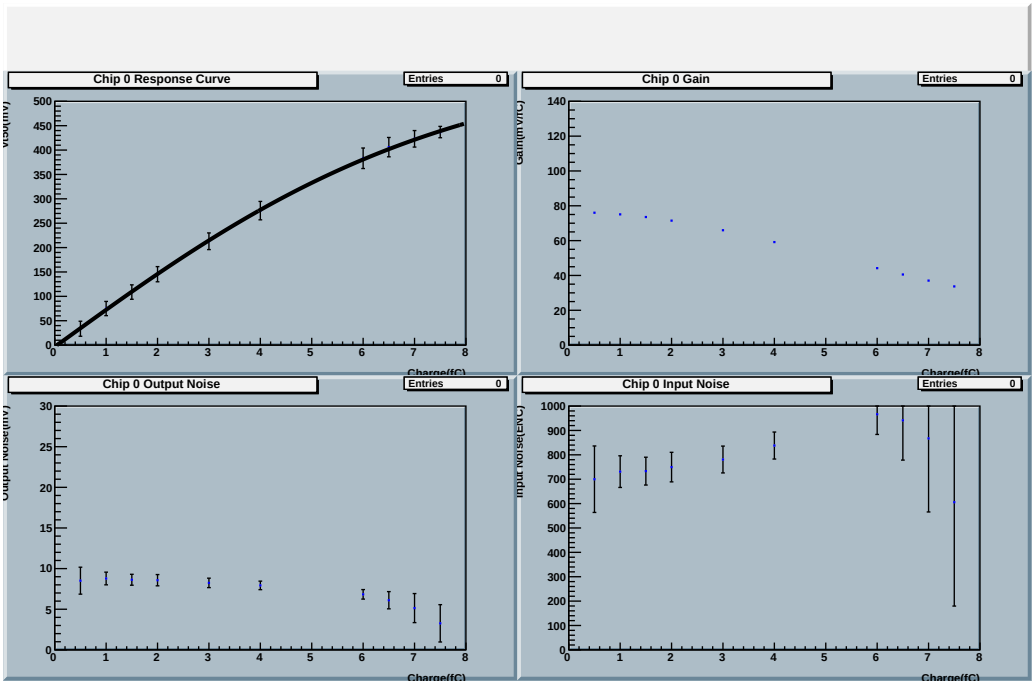


Figure 6.5: Response curve for chip 0 of the static side module.

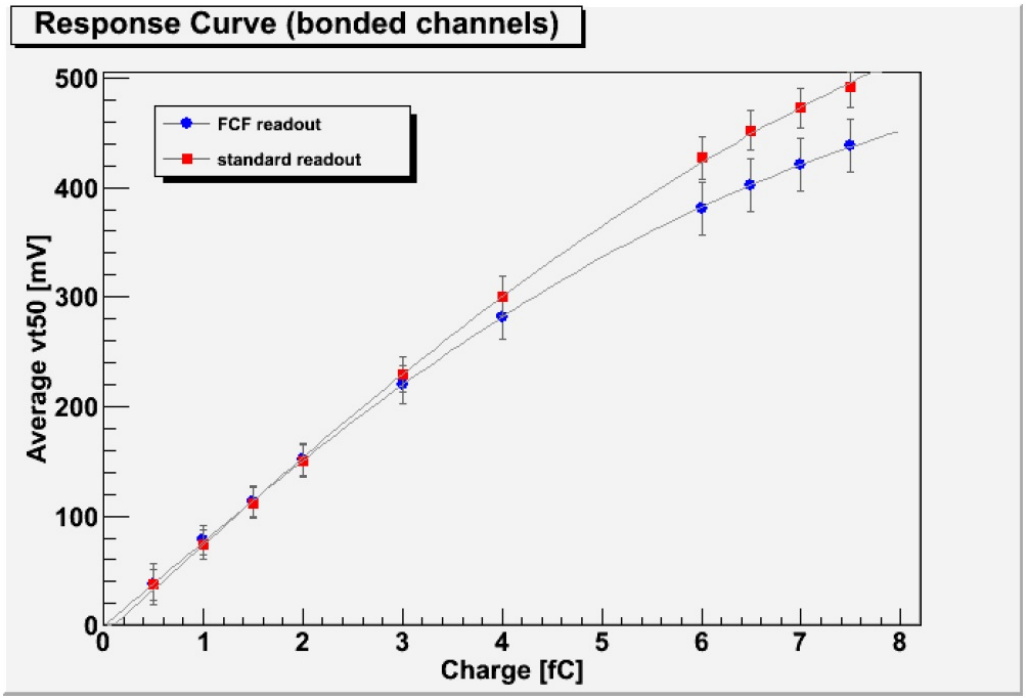


Figure 6.6: Response curve for the single chip boards. One board was measured with FCF and the other with the standard readout [15, Figure 81].

7 Test Beam

The correlator demonstrator was tested in a electron test beam at the Stanford Linear Accelerator Center (SLAC).

In parallel with the correlator demonstrator experiment, a calibration measurement of the ABC130 chip was performed on a single chip board. These measurements are not directly related to my work and are therefore not discussed in detail. They were part of the bachelor thesis of J. Stalder [15].

The results obtained during the test beam experiments are the first beam results with the ABC130 chip. It was also the first time, that a correlation logic was used in a test beam with the fast readout provided by the FCF output.

During the first days of the test beam, the focus was more on the measurements of the single chip board. This time was used to improve the readout system and do some preliminary tests. The main measurements with the doublet were done in two nights. Table D.1 lists the runs and the different parameters. The run numbers indicated in the figures of this chapter correspond to this table.

This chapter describes the test beam setup together with the analyzed measurements. The different section describe the following topics:

- Section 7.1 motivates why the test beam experiment was performed.
- Section 7.2 gives a very short introduction of the Stanford Linear Accelerator Center.
- Section 7.3 explains the different experiments and how they were installed.
- Section 7.4 analysis the measurements with regard of the beam profile and the distribution of the test beam observed.
- Section 7.5 describes the threshold scan performed with on the demonstrator.
- Section 7.6 presents the results from the test of the correlation logic.
- Section 7.7 describes the simulations performed to explain observations made on the test beam data.
- Section 7.8 shows the matching of the measured correlations and the tracks found with the telescope.
- Section 7.9 summarizes the experiment and results and describes possible improvements for the correlation logic and the readout system.

7.1 Motivation for the test beam

To test the correlation logic with real particles, it is possible to use either cosmic rays, a radioactive source or a test beam. Cosmic rays are everywhere available but it requires a long time to collect enough data. Also the direction of the cosmic rays is hard to control, what makes it less suitable to thes the correlator. A radioactive source can also be used in the laboratory, but has a low energy of only 1MeV. This energy level is limit if it has to go through multiple detectors as it is the case for the correlator. A test beam provides a high enough energy and can the direction of the beam is well defined. The test beam provides the most controlled environment and is best suited for to test the correlator. The test beam was chosen for these reasons, and because a time slot became available.

During the progress of the work a time slot, at the SLAC became available. SLAC is within driving time of LBNL and therefor conveniently located. Part of the ATLAS group performed an experiment earlier in the summer, what allowed us to get an impression of a test beam environment. Further the EUDET telescope[1] was installed at SLAC for this experiment and remained on place for the rest of the summer. This telescope is well known and can be used as a reference for measurements performed.

During the discussion for organizing the test beam, the idea of using the same time to perform characterisation measurements on the ABC130 came up. The gain of the ABC130 is not entirely understood yet. The test beam would provide a measurement with a well defined charge deposition to verify the gain. Performing threshold scan in the test beam allows to measure the gain with a input charge other than the internal calibration capacitor. See [15] for more details about the single chip measurements.

7.2 About SLAC

SLAC holds the longest linear particle accelerator and was the site of fundamental discoveries in particle physics, such as the prove of the existence of quarks. A view of SLAC is shown in figure 7.1.

The accelerator center includes now the Linac coherent light source (LCLS), which produces very bright X-ray pulses. This allows to capture atoms and molecules in motion [25].

The end station (A) test beam (ESTB) uses 5 Hz of the LCLS beam for detector testing and development, as in our case to test the correlator demonstrator. This beam is directed to end station A (ESA) where the devices under test are located.

7.3 Setup and installation

From the 5^{th} of July to the 15^{th}, the doublet was installed at SLAC and measurements were taken.

[1] for more info, see *http://www.eudet.org*

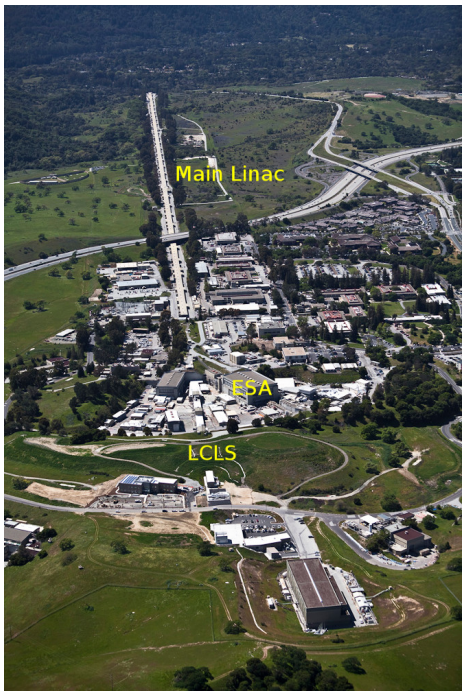


Figure 7.1: Overview of SLAC (source: *www-group.slac.stanford.edu*)

The test beam experiment consists of three detectors: the correlator demonstrator, the ABC130 single chip board and the EUDET telescope. They were placed at the end of the beam in ESA. The telescope consists of 6 pixel sensors, where three sensors were placed before and three after our devices. Figure 7.2 shows the setup.

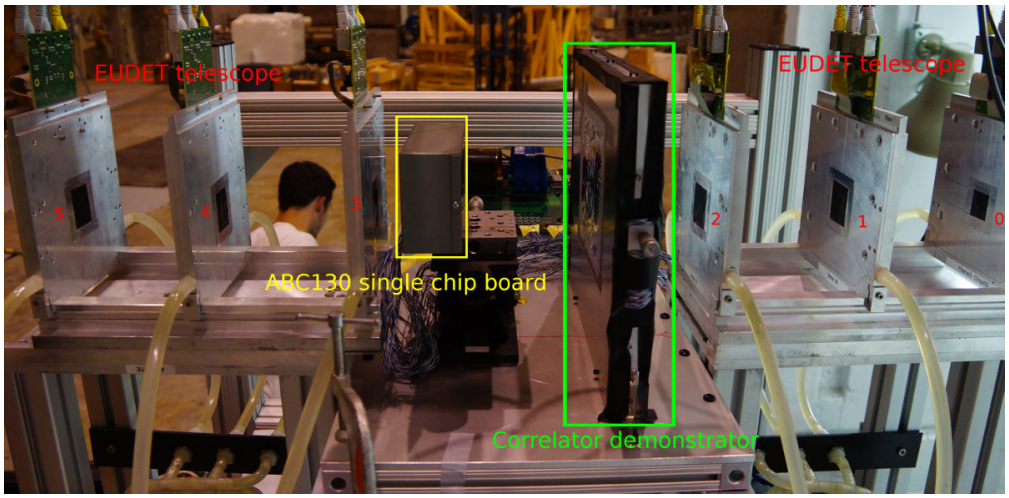


Figure 7.2: Setup of the devices under test in the middle of the telescope. The beam is comeing from the right of the picture.

The doublet was mounted directly on a support structure. This structure was fixed at a height, to place the pixel detectors from the telescope in the middle of the two strip rows bonded to the ABC130 chips.

All experiments (telescope, single chip board and correlator demonstrator) were mounted on a XYZ table that could be remotely moved around. This table could be used to place the detectors in the beam spot. All three experiments had to be aligned together so that all of them see the beam.

The beam was tuned with looking at the results from the telescope. The goal was to have a beam spot size of 1 cm in diameter with around 10 electrons per bunch.

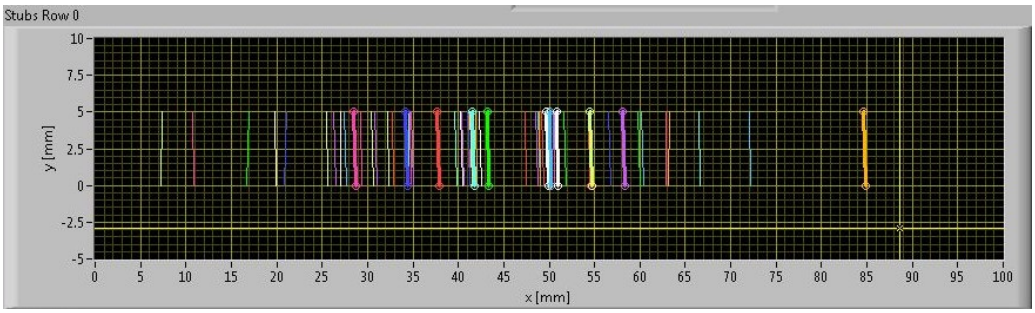
The final beam spot obtained was somewhat larger with 2 cm diameter. A very low intensity with only few 10s of electrons. The intensity was varied depending on the experiment. A higher intensity was used for the single chip board, to reduce the time required for one measurement and still obtaining a sufficient number of hits in a reasonable time. Most of the time there were around 20 electrons per bunch. The energy of the beam was always around 9-10 GeV. There were slight variations due to changes in the LCLS experiment running at the same time.

Figure 7.3: Panoramic view of ESA. The setup is in the middle of the picture, next to the two persons. The beam comes out of the tunnel and goes to the right.

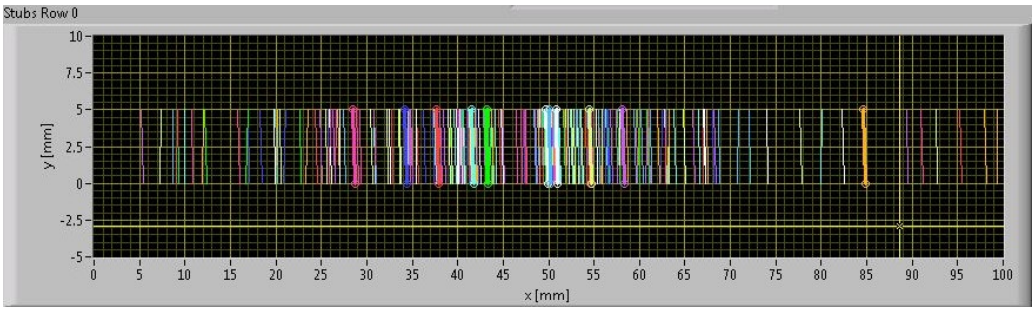
7.4 Beam profile

A method to verify that the beam was actually observed is to look at the location of the seen correlations and control the beam profile. In a very coarse way, this was possible to observe on the real time plots of the LabVIEW application. Figure 7.4 illustrates this with screenshots of the user interface at different times of a run. The screenshots show the graph of one row, where the correlations are plotted as vertical lines. The x-axis is the x position on the sensor. The y axis corresponds to the two sides. The hit location of the mobile (bottom) side is at y=0 and the static (top) side at y=5. The x position is set on both sides and in case of a non-zero correlation distance, the line is slightly tilted.

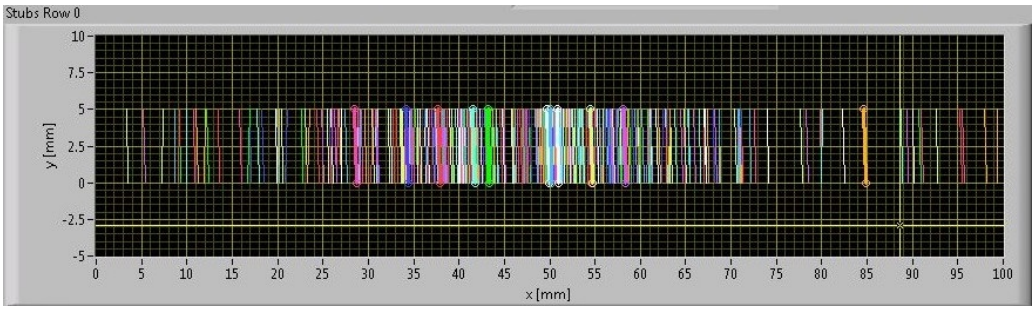
One can observe that there are more correlations detected in the middle of the sensor than at the edges, which means that the beam is centered on the middle four chips.



(a) at start of the run



(b) in the middle of the run

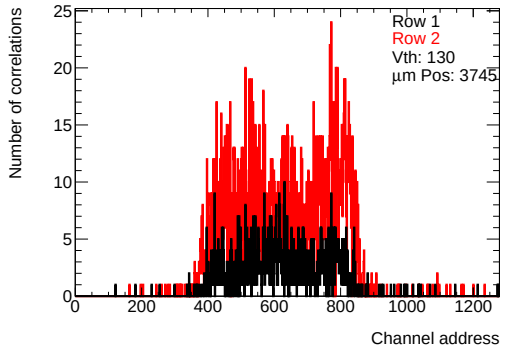


(c) at the end of the run

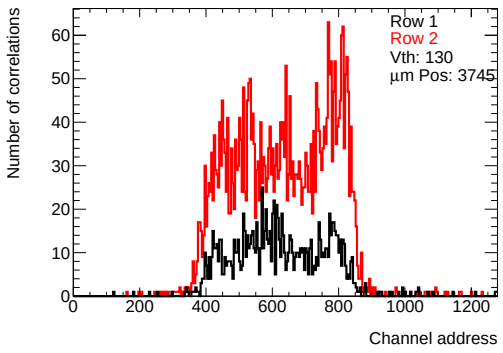
Figure 7.4: Correlations shown in LabVIEW at three different times of a run

For a better analysis of the beam profile, the number of correlations per address or per chip is plotted. The use of the chip ID would be the same as the use of only 10 bins for the histogram creation. It has the advantage, that the distortion of the address distribution from the FCF readout is removed. Figure 7.5 shows the address distribution for one run.

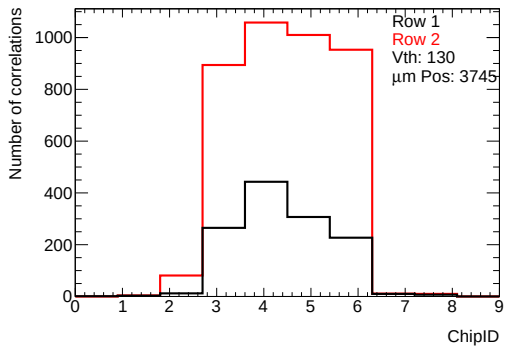
It can be seen, that most of the correlations are observed by the chips in the center of the sensor (chips 3 to 6). Also row 2 has more hits than row 1. This indicates that the beam spot was not centered in the middle of the two rows. The shape observed corresponds the the observations from the other experiments. The sensor of the ABC130 single chip board



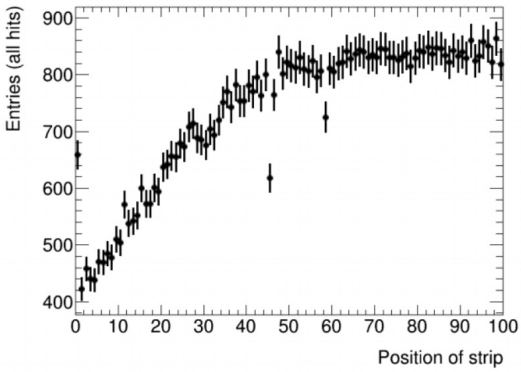
(a) individual channels, run 79



(b) histogram bins of 4 channels, run 79



(c) per chip, run 79



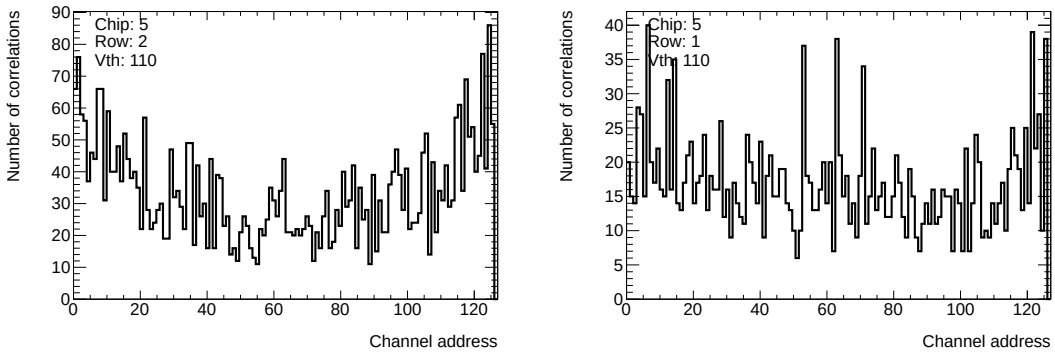
(d) Beam profile captured with the ABC130 single chip board

Figure 7.5: Distribution of the correlation address with different histogram binning. Beam profile from single chip board as comparision.

is with 1x1 cm^2 much smaller than the doublet. Nevertheless it was possible to capture the edge of the beam spot with the single chip board as shown in figure 7.5d. The conclusion was that the beam has a flat center part, with a tail on both sides.

The distribution of the correlations is not flat in the center part. There are some peaks observed, which correspond to the edge of a chip. Because the FCF logic returns only the first two clusters from each end of a chip, additional hits in the center could be missed. This effect starts to show, when three or more hits per chip are observed. This was shown in a small simulation of the FCF logic and is described in section 7.7.1. The large variation of hits from channel to channel is most likely a result from the bad trimming, see section 6.2.2.

Figure 7.6 shows the address distribution for a chip in the center of the beam spot. The addresses at the edge of the chip, i.e. close to 0 and close to 127 see more correlations than in the center of the chip. We tried to tune the beam for an intensity of 5 to 10 electrons per bunch. As the measurements from the telescope show, the intensity was around 17



(a) (b)

Figure 7.6: Distribution of the correlation address per chip for both rows. Both plots runs 52 to 71.

electrons per bunch. Given that the bunch is distributed over multiple chips, each chip should see only a few hits. As already observed earlier, there are more hits on row 2 than row 1. This is also confirmed when the address distribution is compared for the two rows, see figure 7.6. For row 2 the bias due to the FCF logic is clearly visible, while row 1 has a flatter profile.

7.5 Threshold scan

To characterize the system a threshold scan is performed. For this the global threshold is varied over the possible range and the number of correlations observed during a fixed number of events is captured. This procedure is also explained in section 4.1.3. For very low thresholds, in the range below < 20 DAC counts, one would expect a low number of correlations. This is because most of the channel will see noise and there will be mostly clusters of three or more strips, which are ignored by the FCF. We expect a high number of correlations for thresholds higher than 20, which will decrease with increasing threshold and should give something like an S-curve.

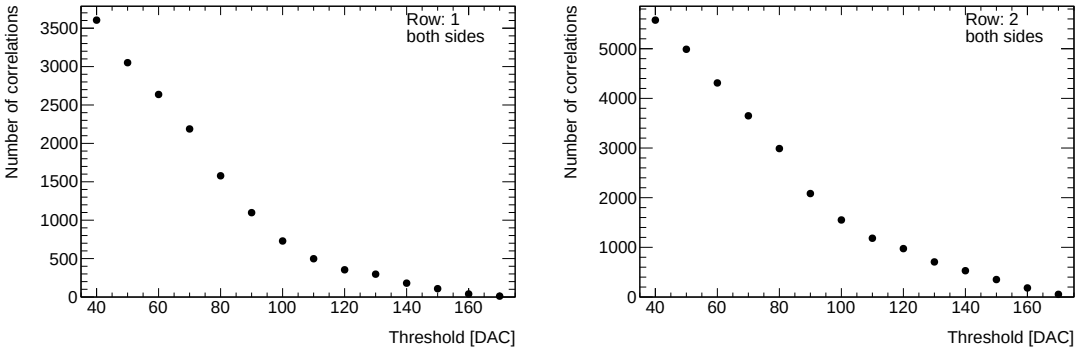
During the test beam three different threshold scans were performed. One where the threshold on both sides was changed together and two where only one side was modified, while the other was at a fix low threshold.

The data acquisition system had problems when the threshold was set very low (less than 40 DAC counts). There was not enough time during the test beam to fix this issue and therefore no data was taken with low thresholds (<40 DAC). Presumably the issue was that the communication between LabVIEW and the MB could not handle the amount of correlations found and the MB went into an undefined state. The FPGA had to be reprogrammed after such a case.

7.5.1 Both sided threshold scan

The first scan was performed earlier than the other two and before fixing the sliding mechanic (see section 7.6). The two sides were not aligned during this measurement.

The threshold was varied from 40 to 170 DAC counts in steps of 10 and 4000 events were recorded for each step. Figure 7.7 shows the result of the threshold scan for each row individually.



(a) (b)

Figure 7.7: Threshold scan with threshold varied the same on both sides. Both plots run numbers 9 to 33 on Jul. 10

As expected we see very high number of correlations for small thresholds and a decrease in the number of seen correlations with increasing threshold. The charge deposition from a particle beam is different than the charge injection. Therefor we don't observe an S-curve as it was the case for the charge injection. The threshold scan in the beam observes the energy loss of the particle. This should give a Landau distribution convoluted by a Gaussian.

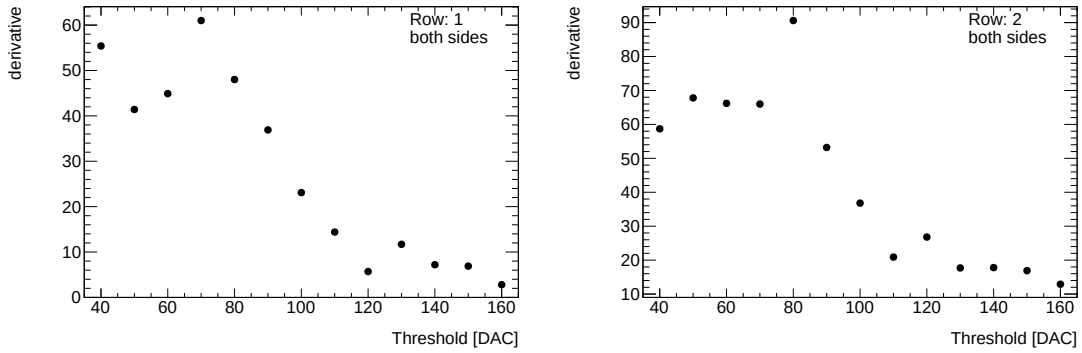
Calculating the derivative of the obtained threshold plot should indicate a peak which marks the V_{T50} point. The negative derivative of the measured threshold is shown in figure 7.8. To calculate the derivative, the finite difference formula was used:

$$\frac{\mathrm{d}y}{\mathrm{d}x} \approx \Delta y/\Delta x = \frac{y_{i+1} - y_i}{step} \tag{7.1}$$

where $step = 10$ for this run, and y_i is the number of correlations observed at a point i.

It is not simple to interpret the found derivative as the expected distribution. The measurement should have been done with a higher resolution to get a clearer image. There is a peak around 70 DAC counts visible which would correspond to the V_{T50} point. This is lower than 120 DAC counts, measured V_{T50} with the single chip board [15]. A smaller second peak is visible at 120 DAC counts, which would correspond to the single chip board.

Further analysis could be performed, by filtering the data more. Because the two sides were not aligned, care has to be taken, which correlations are removed from the data set.



(a) (b)

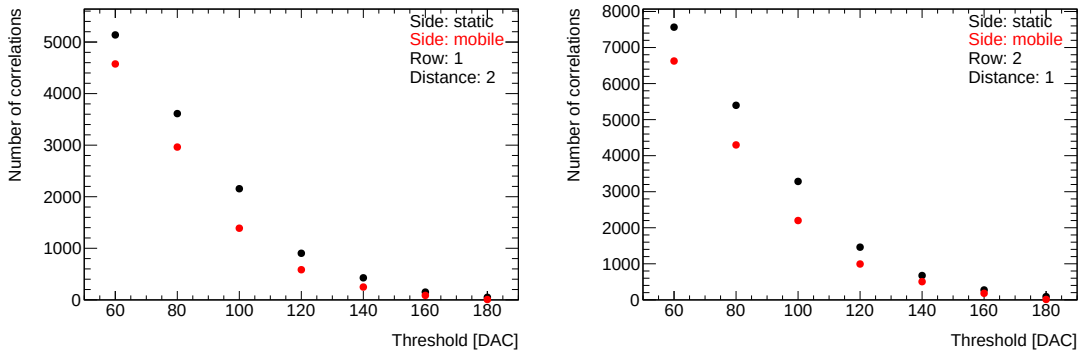
Figure 7.8: Derivation of the plot in figure 7.7.

7.5.2 Threshold scan on static and mobile side

Two additional threshold scans were performed at the end of the beam time. Those were done with the two sides aligned. Further one side was kept constantly at a threshold of 40 DAC, while the other was scanned from 60 to 180 DAC counts in steps of 20. The low threshold was selected to limit the hits only on the side, where the threshold scan was performed. The goal of this measurement was to characterize each side of the doublet individually.

To suppress a possible error from false correlations, the data for the single sided threshold scans were filtered. Only single hits and also only correlations with the correct distance were taken into account. Though none of these filters had a big impact on the result.

The threshold curves obtained from the single sided threshold scan are similar as with



(a) (b)

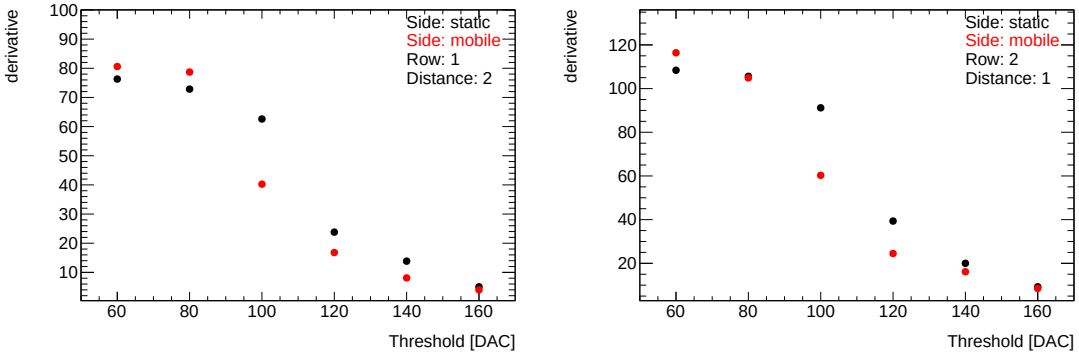
Figure 7.9: Threshold scan with threshold varied on only one side, while the other was kept at 40 DAC. Run numbers 39-45 for mobile and 72-78 for static side.

the both sided scan. They are shown in figure 7.9. The absolute number of correlations is much larger than for the double sided scan. This comes mostly from the fact that the two sides were aligned within the correlation range for the single sided test. On the mobile side scan are less correlations observed than for the static side.

There was a change in the beam intensity between the two measurements. For the mobile side scan, the telescope registered around 25 electrons per bunch. While only around 16 electrons per bunch were seen for the static side scan. This intensity is contradicting the observations.

More likely it is due to the correlation logic. The mobile side was connected to the bottom lines. The logic searches form the bottom hits corresponding top hits. Therefore less bottom hits have apparently a bigger impact than a less top hits.

Looking at the derivative of the single sided threshold scans, there is no longer a peak observable (see figure 7.10). The derivative was calculated again with equation 7.1, but with a step size of 20. This is most likely due to the larger step size or the peak could be at 60 or less DAC counts. This scan should be repeated in a future test beam experiment with a smaller step size.



(a) (b)

Figure 7.10: Derivation of the plot in figure 7.9.

7.6 Micrometer scan

As described in section 5.5.1 a micrometer allows to move one side in parallel to the other.

The two sides of the doublet have been aligned during the fabrication for a given micrometer position. Because of mechanical problems with the setup, this position was lost and the micrometer had to be repositioned. The problems were mainly due to screws, either not well loosened or not tight enough. After fixing the setup, the movement worked well, but the correct position had to be found.

Measurements were performed at different micrometer positions until a peak in the correlation distance was found. Once this peak was discovered, a scan in steps of one pitch (80µm) was performed over a range of 9 strips.

7.6.1 Beam angular distribution

The test beam provided should have a very small angular distribution. This is obtained with two collimators, which are place as indicated in figure 7.11. The maximal angle that allowed by the collimator is 27.5 m$^\circ$.

$$\alpha = arctan\left(\frac{12mm}{25m}\right) = 27.5m^\circ \tag{7.2}$$

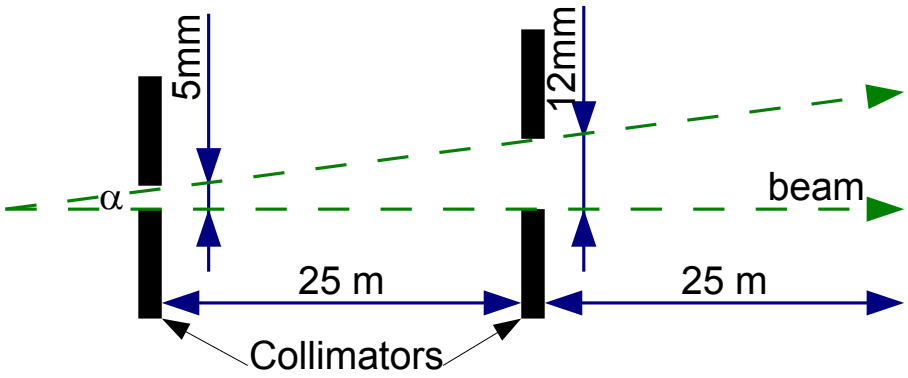


Figure 7.11: Sketch of collimators used in the test beam

The angular resolution of the doublet can be calculated from the strip pitch, i.e. 75 µm, and the distance between the two sides, which is approximately 5mm.

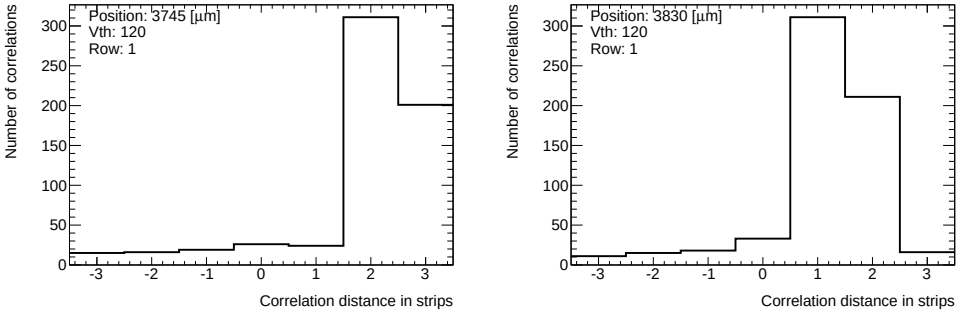
$$\alpha = arctan\left(\frac{75\mu m}{5mm}\right) = 860m^\circ \tag{7.3}$$

The angular resolution of the doublet (equation 7.3) is 30 times bigger than the angular distribution of the beam. This means that all the electrons should pass straight through the doublet. In case of alignment of the two sides, the correlation distance should be 0 for all hits.

7.6.2 Distance distribution per micrometer position

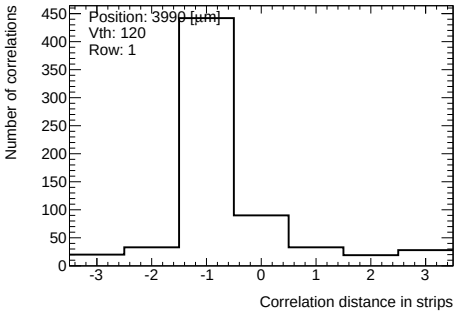
For each micrometer position (µm-pos) the number of correlations per distance is plotted. The distance is defined as the difference between the top and bottom hit and is the field with the same name as defined in table 5.3. Figure 7.12 shows some plots of the correlation distance for different µm-pos.

There is a clearly defined peak in the distance for each µm-pos. From the analysis of the beam, it is expected that only one possible distance should occur in the correlations. As can be seen in figure 7.12, there is a baseline of correlations on all distances. Also against the expectation there were some correlations observed, independent of the micrometer position.

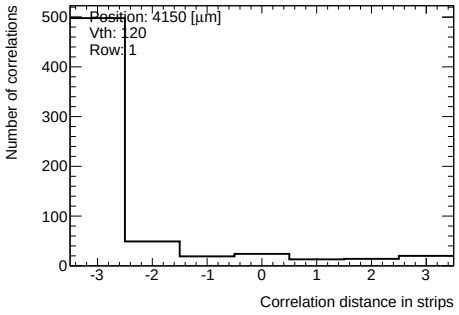


(a) Run 52

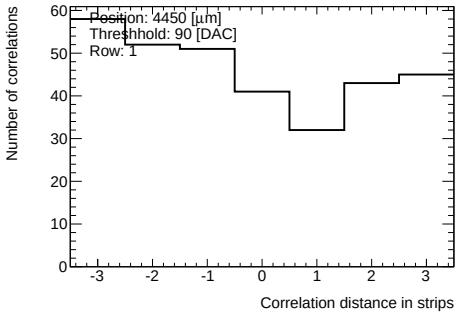
(b) Run 61



(c) Run 65



(d) Run 69



(e) Run 33, Jul. 14

Figure 7.12: Correlation distance plots. The main peak is wandering with the µm-pos. Outside the aligned range (7.12e) are still correlations observed with a flat distance distribution.

Because a flat profile of the correlation distance was observed in the first measurements, we weren't sure if the movement was working or if there was an error in the logic. Because the step size did not exactly correspond to the pitch of the strips (80µm step instead of 75µm pitch), the strips will not be perfectly aligned. This can be seen for the position 3745µm (figure 7.12a), where a peak at distance 2 is observed, but also distance 3 has a large value.

It can be assumed that the strips are overlapping. The same is observed for the position 3830µm. In case of position 3990µm (figure 7.12c) the two sides are better aligned, because there is only a very samll side peak.

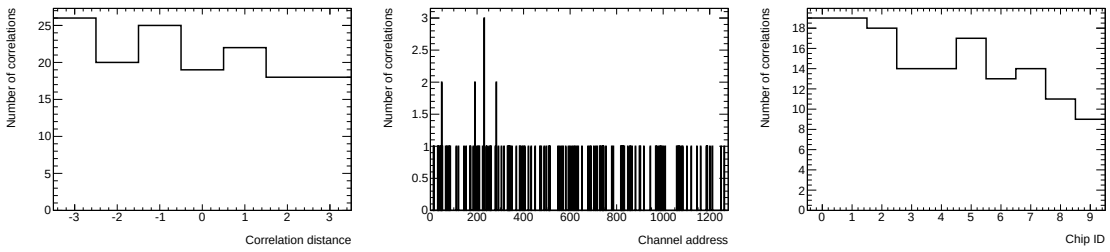
Outside the range where the two sides are aligned, a flat profile is observed as can be seen in figure 7.12e.

There are multiple possible reasons behind these wrong correlations which are discussed below.

Correlations from noise hits

One possibility of wrong correlations are noise hits. This is tough very unlikely and only a very small percentage of the correlations should come from this source. To create a correlation from noise hits, there have to be matching hits on both sides. Also the threshold was chosen quite high with 110-120 DAC counts (250-280mV). Further, almost no correlations were observed at SLAC, while the test beam was off.

After the test beam, a measurement was performed, where the correlations were captured during three hours with a threshold of 60 DAC. No specific particle source was present, so the correlations observed in this test could come from either noise or cosmic rays. During this three hours 148 correlations were found, i.e. a very slow rate of less than 1 correlation per minute. The correlation distance and the distribution over the sensor for this run are shown in figure 7.13.



(a) correlation distance (b) correlation address (c) correlations per chip

Figure 7.13: Correlations measured in a period of 3 hours in the laboratory at LBNL. Data from run 2 on July 29.

The distance distribution is very flat, as well as the distribution over the sensor. Looking at the correlations per chip, there is a slight tendency towards the lower chip IDs.

From this test, it can be concluded, that noise or cosmic ray hits should have a very low influence on the system. First, the threshold set during the noise run was lower than for the measurements performed during the test beam. Second the runs at the test beam were done for around 3000 events which takes 10 minutes at the 5 Hz beam rate, i.e. there should be less than 10 noise hits within the data taken.

Unfortunately no noise measurement was not performed at ESA. There could have been a higher background radiation. It has to be still very small, because it was possible to

detect that there was no beam with the correlator. Whenever the beam was switched off, no (or only very few) correlations were observed.

Multiple scattering

Before arriving at the doublet, the electrons in the beam passes through different parts of the telescope. It is therefore possible, that some of them get deviated and arrive at a different angle on the doublet, an effect called scattering. According to [13, Section 31.3], a Gaussian approximation is sufficient for many applications. The central 98% of the projected angular distribution can be calculated with the formula 7.4

$$\Theta_0 = \frac{13.6MeV}{\beta cp} z \sqrt{\frac{x}{X_0}} \left[1 + 0.038 \cdot ln\left(\frac{x}{X_0}\right)\right] \tag{7.4}$$

where p, the momentum, βc, the velocity and z the charge number. The ratio x/X_0 is the thickness of the scattering material in radiation lengths.

To simplify the calculations following assumptions are made. In front of the doublet are three Mimosa pixel sensors located. Each pixel sensor has a thickness of 50 µm. Therefore the thickness of the scatter material in equation 7.4 can be estimated as 150µm of silicon. The radiation length of silicon is 9.36 cm and the charge number is 1. $\beta = v/c \approx 1$, because the electrons at the test beam are accelerated to almost speed of light. The momentum p times speed of light c equals the energy of the test beam, which was around 10 GeV. The ln term is negeleced because $x/X_0 < 1$ it would reduce the angle by approximately 25%. This gives for Θ_0

$$\Theta_0 \approx \frac{13.6MeV}{10^4 MeV} \sqrt{\frac{0.15mm}{93.6mm}} \approx 54.4 \cdot 10^{-6} rad = \underline{3.12m^\circ} \tag{7.5}$$

This angle is almost 300 times smaller than the resolution of the doublet. Therefore we can assume that the scattering due to the pixel detectors in front of the doublet doesn't cause the observed baseline.

Another possibility would be to estimate the scattering from the fixtures of the mimosa sensors. The pixel sensor have a cooling system and are mounted in an aluminium fixture. A rough estimate is to consider something like 2mm of aluminium for each sensor. Aluminium has a radiation length[2] of 8.9 cm . Using again equation 7.4 with the same approximations as for the silicon, we get a scattering angle of $20.2m^\circ$ for the aluminium structure. This angle is still a magnitude smaller than the doublet resolution.

A more precise calculation could be made. Given that these estimates give angles smaller than the doublet resolution by a magnitude, it would lead to the same conclusion. Which is that the material in front of the doublet shouldn't have any effect on the correlation distance. It can not explain the baseline observed.

[2]The radiation length for silicon and aluminium was taken from: *http://hepwww.rl.ac.uk/Atlas-SCT/engineering/material_budget/material/calculator.xls*

Correlations between two electrons

The correlation logic has no selection in case of multiple correlations per hit. E.g. if there are two matching top hits for a bottom hit, both correlations are considered valid and sent to the DAQ system.

If two electrons pass through the doublet within a couple strips, it is possible that there is an additional correlation. One could imagine the following example of hit addresses:

$$\begin{aligned} top\ hits &= 120\ and\ 124 \\ bottom\ hits &= 119\ and\ 123 \end{aligned}$$

There are two correlations with a difference of 1. But there is also a correlation from top channel 120 to bottom channel 123 with a distance of -3. Looking at figure 5.4 it would mean that the correct tracks (2) and (5) were found, but also the wrong track (4).

By adding additional logic to allow only one correlation per bottom hit, those wrong tracks could be removed. The priority should be given to the tracks with the smallest distance.

In the simulation described in 7.7.2, I tried to reproduce the same results seen in the test beam.

Bad trimming

As only later discovered, there is still an error in the application of the trimming file. This leads to a large variation of the threshold value for the different channels. Therefore some channels require a much smaller threshold voltage compared to others to see a hit. This could result in errors in case of two hit clusters, where only one channels registers the hit.

It is also possible that a three hit cluster is split into two one hit clusters, in case the middle channel does not detect a hit. Such a case could also add additional wrong correlations.

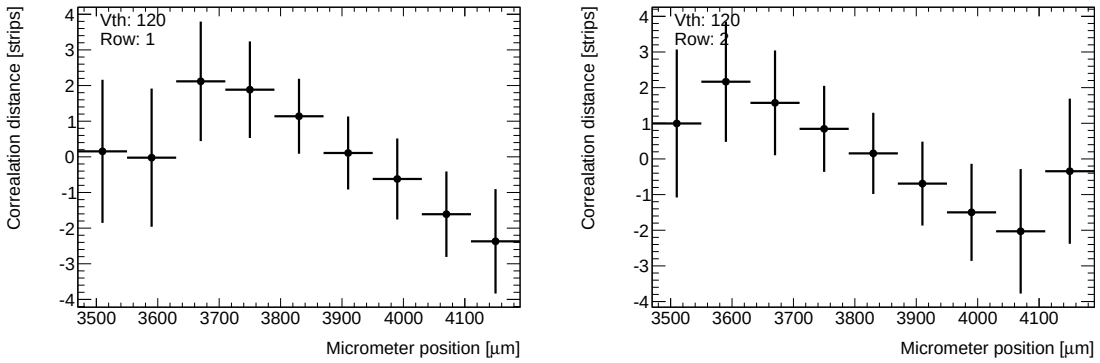
7.6.3 Mean distance versus micrometer position

A further verification of the correct behaviour for the correlator is taking for each position the mean value of the distance distribution and plot it against the µm-pos. Those plots for both rows are shown in figure 7.14.

The mean distance has a mostly linear dependency of the position within a certain range. Expected would be a linear dependency with a slope of 1 strip per step of 75µm. In other words, for a movement equivalent to one pitch size, the mean distance should also move by one. Once the movement passes plus or minus three strips, no more correlations should be found.

The found mean value has a slightly smaller slope than expected. This comes from the fact, that there is this baseline of wrong correlations as described above. These wrong correlations distort the mean towards 0.

Outside the aligned region, the mean value returns towards 0 with a flat distance distribution. The number of correlations is also drastically reduced, once the mobile side is moved out of the valid region. For row 1 there is already a peak at distance 3 for the position

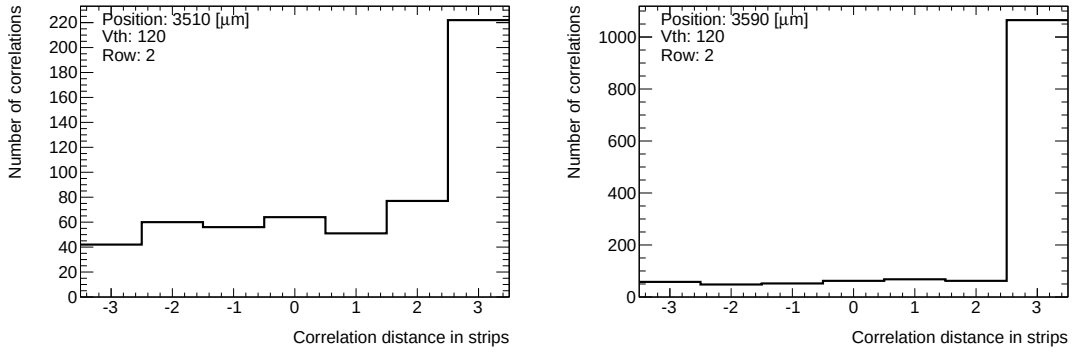


(a) (b)

Figure 7.14: Mean distance as function of the µm-pos for both rows. Runs 52, 55, 57, 60, 61, 63, 65, 67 & 69

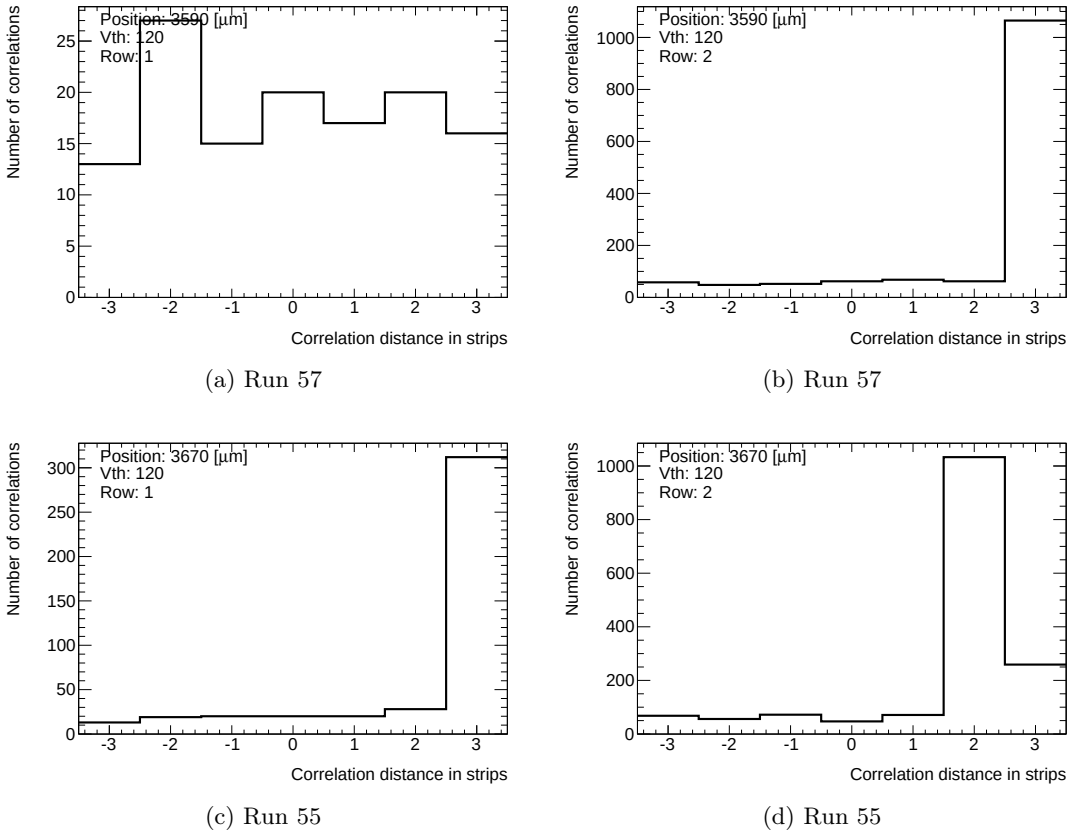
$3510\mu m$ (see figure 7.15. The peak is with 220 correlations much lower than for the next position, where it has more than 1600 correlations. This explains why the mean distance is not at 0 for row 2 at this position. It indicates further, that the selected positions did not correspond perfectly to a position where the strips on both sides are perfectly aligned.

As described in section 5.5.2, there is an offset of 1 strip in row 1 compared to row 2. This offset coming from the wirebonding is clearly visible in figure 7.14. For the position $3590\mu m$ row 2 has the peak at 3, whereas row 1 is out of range. For the next step at $3670\mu m$ the peaks are at 2 for row 2 and 3 for row 1. This is also shown in figure 7.16.



(a) Run 60 (b) Run 57

Figure 7.15: Distance distribution of row 2 at the edge of the valid range.



(a) Run 57

(b) Run 57

(c) Run 55

(d) Run 55

Figure 7.16: Comparison of the distance distribution for the two rows.

7.7 Simulations

Some of the observed behaviour were studied more in detail and verified with simulations, to get a profounder understanding of the system and to better interpret the data.

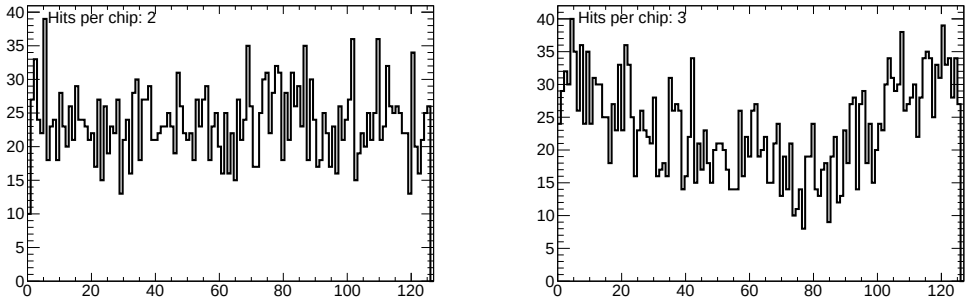
7.7.1 FCF readout address distribution pattern

To verify the pattern observed on the address distribution, a small simulation was made. A given number of hits were created with a uniform distribution over the 128 channels of one chip. The highest and lowest address of the hits were selected, as it is done in the fast cluster finder logic.

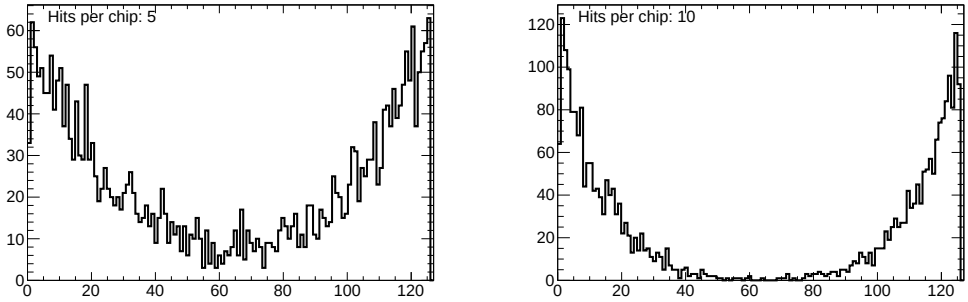
As can be seen in figure 7.17 the valley shaped distribution starts to appear with 3 hits. As expected, the effect increases with more hits per chip.

Comparing the simulated distribution to the measured distribution, it should be possible to get the same shape of the address distribution. Figure 7.18 does this for two chips inside and outside the correlation range.

For the chip within the correlation range, the simulation and the measurement have a

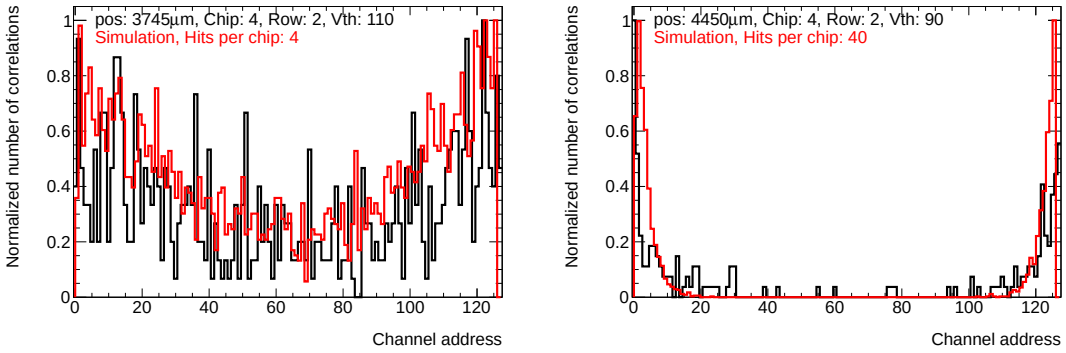


(a) (b)



(c) (d)

Figure 7.17: Simulated random address distribution for one chip with a given number of hits per chip.



(a) Simulation and run 71 (b) Simulation and run 33, Jul. 14

Figure 7.18: Comparison of simulated to measured address distribution. Once for a chip within the correlation range, and once outside the correlation range.

similar figure for 4 hits per chip. The four chips in the middle of the sensor (chip 3 to 6) see almost the same amount of correlations. From this we'd expect to have a bunch size of 16 electrons per event. From the telescope data a bunch size of 17 electrons per event was detected for the same run analyzed here (see figure 7.19). The simulation doesn't consider real correlations, but only the FCF algorithm. Because the measurements are correlations, additional hits not creating any valid correlation are not seen in the data. Unless there are other effects adding to the creation at the valley profile, it would mean an even higher hit number. Not really matching the observed address distribution is the fact, that the number of correlations is much smaller than the number of hits.

In case of the measurement outside the correlation range, the shape is very strong biased and it resembles the simulation with 40 hits per chip. This number of hits is definitely too big. Something in the design of the correlation logic has to favor wrong correlations at the edge of the chips or cross-chip correlations. This also indicates some additional effect leading to the valley shaped address profile.

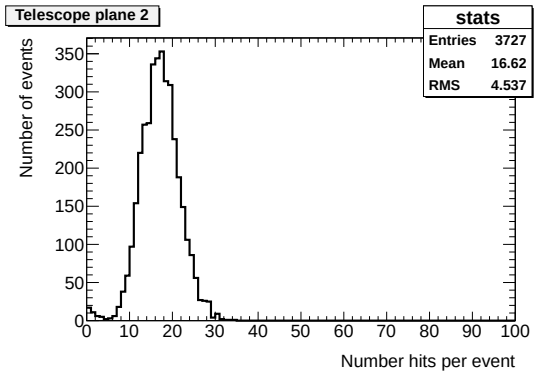


Figure 7.19: Number of hits per event measured with the telescope plane 2. Data from telescope run number 1503.

7.7.2 Random correlation hits

One hypothesis to explain the baseline in the correlation distance and the observations of correlations when the doublet was not aligned is the correlation of two separate hits (see section 7.6.2).

To verify this a function was written for the MicroBlaze, which creates random masks for correlations. As input, this function takes the number of desired hits and the distance of the correlation. The number of hits is also random uniform spread between 0.5 and 1.5 times the given input.

The hits are uniformly spread over the middle four chips. These chips did see most of the hits during the test beam. The mask registers are set for each chip individually so that correlations are generated always with the same distance. This corresponds to the test beam which should be focused enough to allow only one distance according to the micrometer position.

Different than in the test beam is the masking of all other channels. Therefore no noise hits are possible. Given that there are only few, this can be neglected.

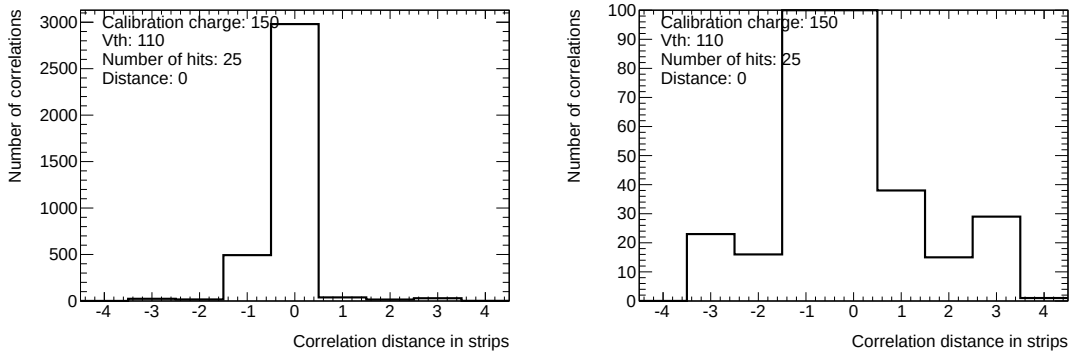
After each call, a calibration pulse is sent to all chips and the found correlations are read out. For each generated mask, the pulse is applied only once.

Correlations in range

Figure 7.20 shows the correlation distance obtained for the randomly generated hits. Each event had an average of 9 hits and the correlation distance was set to 0. The threshold was set to 110 DAC counts, which corresponds a threshold used in the test beam. The injected charge was set to a value of 150 corresponding to 5.25 fC.

It can be observed that almost all correlations are found with the correct distance. A baseline of around 25 hits over the whole distance range is observed similar as in the test beam. A higher count of correlations is also observed at a distance of -1. This comes most likely from double hit correlations, where only one of the two hits was detected.

The ratio between the peak and the baseline depends largely on the selected threshold and the injected charge.



(a) (b) zoomed on y axis

Figure 7.20: Correlation distance for random generated hits.

Correlations out of range

If the random correlations are generated with a distance larger than allowed by the window, there are still correlations observed as it was the case during the test beam. The number of correlations observed in this simulation is much smaller than when the distance was set within the valid range. The correlation distance is distributed over the whole valid range as can be seen in figure 7.21.

Comparison simulation and measurement

Run 33 and 61 from Jul. 14 will be compared to the simulation. The actual distance for run 33 is unknown but should around -7 according to the micrometer position and the observed

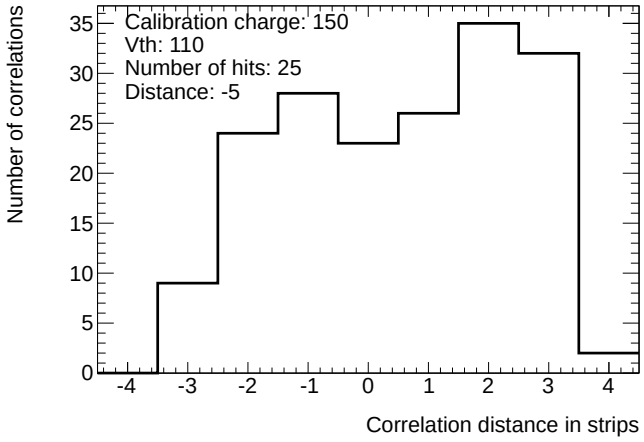


Figure 7.21: Random generated correlation with a distance outside the range.

correlations in other runs. Run 33 acts as a run out of range. Run 61 has a distance of 1 on row 1, resp. 0 on row 2 and is in range. Table 7.1 compares the measurements to the simulations.

A higher number of correlations was observed in the test beam in case the distance is outside the range. If the distance is in range, the simulation sees correlations in every event. This was not case during the test beam measurements. There only a third of the events had a correlation observed.

The simulation showed that multiple hits can generate wrong correlations. A more profound analysis has to be done to find more sources of wrong correlations. E.g. the telescope data can be used to eliminate wrong hits in the data set.

	Nb. events	**Nb. correlations**	**Correlations per event**	**Events with correlations**	**Mean nb. correlations**
Sim out of range	400	180	0.45	144	1.24
Run 33	1000	1135	1.14	469	2.34
Sim in range, dist 1	400	3595	8.99	400	8.99
Run 61	3000	2263	0.75	1065	2.12

Table 7.1: Comparison simulation to test beam measurement

Trimming

The trimming applied during the test beam was not correctly set. To study the influence of this bad trimming, the random correlation generation was applied to the untrimmed and the bad trimmed doublet.

With the trim file applied, a higher number of correlations was observed. Most of them were nevertheless observed in the set distance. Therefore the bad trimming is not creating additional wrong correlations. Figure 7.22 compares the two simulations together

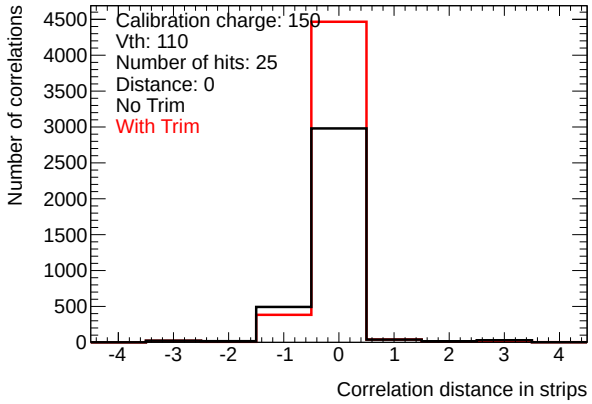


Figure 7.22: Random generated correlation with applying the trim file and without.

7.8 Matching correlations to telescope

For the runs analyzed the trigger number from the TLU was registered and stored together with the correlations. It is therefore possible to match the data from the correlator together with the telescope data.

The matching of the telescope data and the correlations was done by John Keller at Deutsches Elektronen-Synchrotron (DESY).

The correlations and the data from the telescope have both the same trigger ID number. For all found hits within the same ID, a histogram was made with an entry for each possible correlation-cluster pair. Filling the histogram over all events of one run gives then a plot as shown in figure 7.23. Expected would be a diagonal line, which corresponds to matching data. This line appears, because the matching hits will be all placed on this line, while random pairs are distributed over the whole area.

To mention is that the telescope had a slight angle with respect to the beamline. This reduced the width of the x-axis in the figure to 15mm instead of 20mm which is the full width of the telescope pixel sensors.

A diagonal line is observed in figure 7.23. This line proves that the data from the correlator are matching together with the clusters found by the telescope.

The horizontal lines observed in the plot are coming from the increased number of hits at the edge of the chip, as described in 7.4.

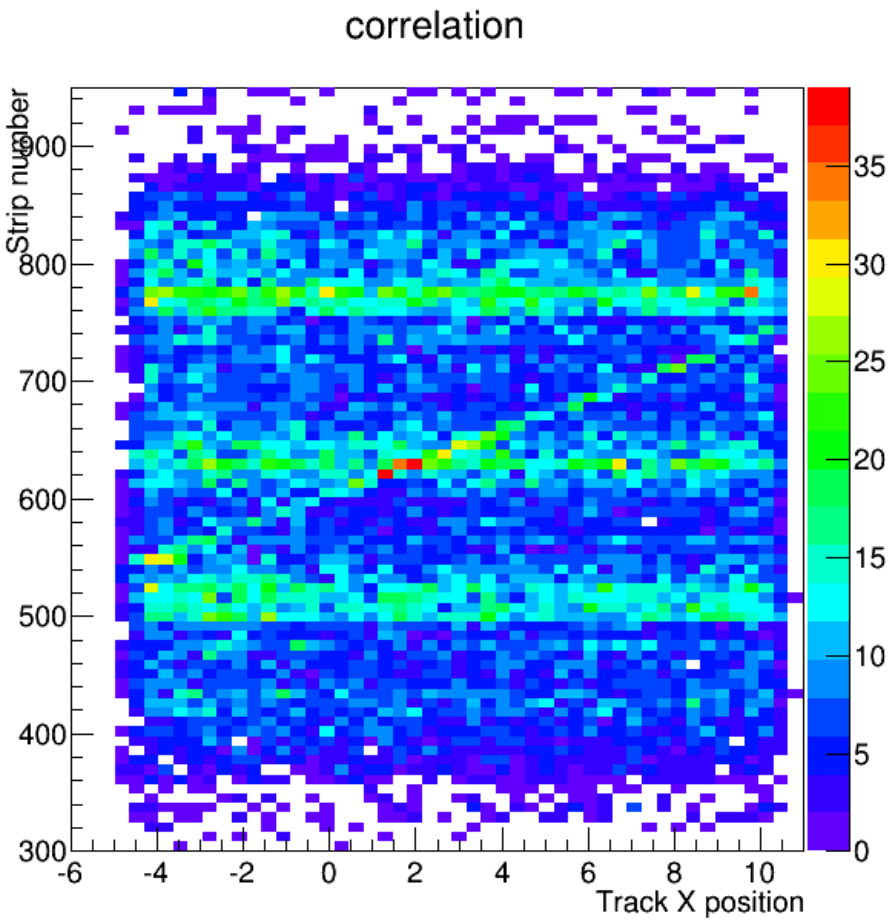


Figure 7.23: Matching of the data between telescope clusters and doublet correlations. x-axis is horizontal position recorded by telescope and y-axis is horizontal position recorded by the doublet.

7.9 Summary of the test beam experiment

The test beam experiment was a success. It was possible to prove that the correlation with a doublet is possible and particle tracks can be found.

The beam spot could be recognized and the expected profile is observed.

The threshold scan performed with the doublet was not optimally done and if there is the chance it should be repeated. It is not yet understood how the correlation influences the threshold scan and if this output can be used to perform characterisation measurements.

The micrometer scan to test the correlation logic did work after fixing the setup. It was possible to observe a peak in the distance distribution move according to the micrometer position. A baseline of the correlation distance is though observed over any alignment. It was possible to explain this behaviour to some extend with a simulation. There may be

more sources for the observed baseline.

It was possible to match the correlation data together with the telescope measurements. More analysis can be done on this part and it should be possible to find wrong correlations by comparing with the telescope tracks.

During the measurements of the test beam and also during the analysis of the data some points were observed which could be improved on the current correlator design. These improvements are described below.

7.9.1 Improvements for the correlator tester and data acquisition

The most important point in the test and data acquisition system is to improve the speed of the communication between the FPGA and the computer. Following is a list with points to improve the system:

1. **Faster connection**: One solution to speed up the communication is to use a faster transmission channel. The used FPGA board has a Ethernet port already available which would be suited for such a communication.

2. **Compress data**: An even simpler solution would be to transmit the data directly in binary format, instead of ASCII characters. The use of ASCII characters made the handling and debugging much easier but is not required otherwise.

3. **LabVIEW application**: The LabVIEW application is not optimized and may not be well suited. The acquired correlations were stored in arrays to directly show them on the front panel. This feature is also most useful for debugging, but the array can grow rather large. Because the space is allocated during run time, this leads to increasing execution time.

Several LabVIEW applications were created to perform the different characterisation tests. To improve the usability, they should be correctly integrated into one program.

7.9.2 Improvements on the correlation logic

Regarding the correlation logic, the most important improvement is the reduction of false correlations. The list below gives some possible corrections together with other issues and how they could be solved.

1. **False correlation from close hits**: One source of wrong correlation is the case where two hits are within the correlation range of each other. A limitation to only one track per bottom hit should be added, to reduce false hits in this case.

2. **False correlation from cross-chip hits**: Another improvement would be a better cross-chip correlation. A double hit right on the border of two chips is currently not handled. Therefore this would result in two individual clusters, one on each chip, and can give multiple correlations. An additional test has to be introduced to catch such a case.

3. **Window flexibility**: Currently there is the integrated memory which defines the allowed distance for correlations. This memory has to be configured before the synthesis of the design. A programmable memory would be more favorable. This would allow to adjust the window during runtime and give more possibilities for testing. A drawback of this suggestion is that the logic would increase drastically, if it is programmable for each channel. Also the currently used memory has a much faster access time.

4. **Fast data output**: The correlation output logic is currently the bottleneck in the data path. An improvement could most certainly be done on the logic to select the valid correlations. One could imagine to split the used algorithm into different blocks and read out block after the other.

   Further could a buffer of the calculated correlations be useful. This would allow to store correlations in consecutive BC cycles and transmit them when less correlations are arriving.

8 Track correlation

This chapter describes the concept for the correlation of a track through two doublets.

A simplified sketch how the track correlator is integrated into the whole trigger system is shown in figure 8.1.

The correlator is the logic block described in chapter 5 and correlates the hit information of one doublet. The development and test of the doublet correlator was the main part of this work. The output logic should be modified to fulfill the specifications for the track correlator.

The track correlator has to be developed and is specified in this chapter.

The output from the track correlator serves as input of a trigger logic which decides when a trigger signal has to be issued. This logic is not specified here, nor the required input signal. Most likely it would be part of the general trigger logic for the ATLAS.

8.1 Timing and speed analysis

This section analysis the required working time for the track correlator and the depending transmissions speed.

8.1.1 Result creation

The L0 buffer of the ABC130 can store data up to 6.4µs. Because some time will be needed for the trigger logic to execute, the track correlator should generate its result with some margin to the 6.4µs.

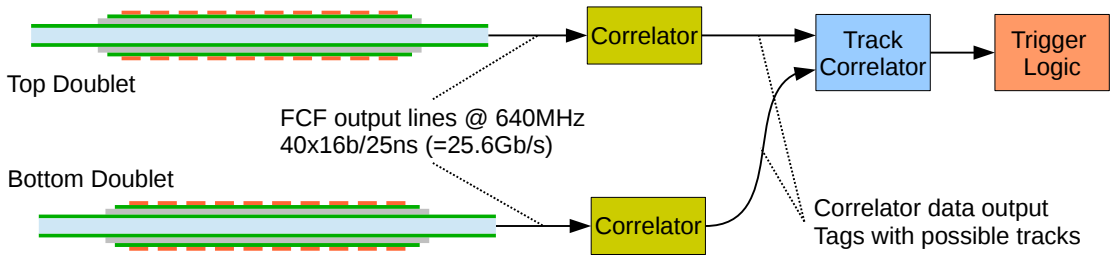


Figure 8.1: Sketch of high level track correlator.

8.1.2 Transmission speeds

FCF lines

The data from the doublets to the correlators is transmitted through the FCF output lines. There are 2 lines per ABC130 chip and 10 chips per module, i.e. a total of 40 FCF lines per doublet. Each FCF output sends 16 bit per 25ns (every beam crossing), which creates 25.6 Gb/s of data that are transmitted with 40 serial lines.

The goal transmission speed for the FCF is 640Mb/s. If this speed can't be achieved, it exists the option to reduce the speed to 320Mb/s. In the latter case, only every 2^{nd} event could be handled by the track correlator.

Correlator to track correlator

The information transmitted from the correlator logic to the track correlator should contain only the valid track stubs from each doublet.

Each stub is encoded as a tag of 22 bits as described in table 5.3, not using the trigger ID. Given that each FCF returns only two hits, there are maximal 4 hits per chip found. If each hit results in a correlation, this would lead to 40 correlations transmitted from the correlator to the track correlator. Preferably this transmission is also performed within one BC cycle, what would lead to a required maximal transfer rate of 35 Gb/s.

$$\begin{aligned} \textit{Data rate} &= \frac{\#tags/chip \cdot \#bits/tag \cdot \#chips\ correlated}{transmission\ delay} \\ &= \frac{4 \cdot 22b \cdot 10}{25ns} = \frac{880b}{25ns} = \underline{35.2Gb/s} \end{aligned} \tag{8.1}$$

A more detailed analysis should be done, if the number of correlations is always the same or if there is a variation. In case of varying number of correlations, it is possible that the rate could be done over several BC cycles to reduce the rate. This would require some protocol to distinguish multiple events and a buffer has to be introduced. Further it should be guaranteed that the result is obtained always at the same time.

It could also be possible to compress the tags, e.g. by transmitting the chip ID only once in case of multiple correlation per chip.

Taking the functional concept into account, it might also be interesting to transmit the data in a different form than currently available at the output of the correlator, as long as the data can be reduced, e.g. transmit directly the slope and intersect or the perpendicular bisector of the stubs. Care has to be taken, that no important information will be lost by the conversion into another format.

8.2 Functional concept

The track correlator takes as input the stubs from the correlator logic and finds possible track of particles passing through both doublets. Following points should be considered:

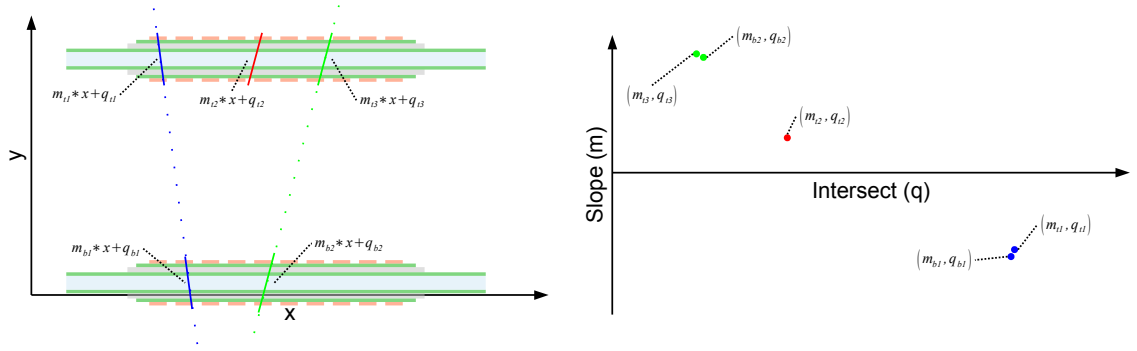
- Tracks can either be
  - straight, or
  - curved, which requires the presence of a magnetic field.

  The track correlator should be able to detect both kinds.
- The finding algorithm should be configurable to set the parameters regarding the precision of the system.
- A configurable range should be included to filter tracks within a certain momentum range. This range could be open towards high momentum.

8.2.1 Straight track finding method

To find straight tracks, the stubs from the two doublets have to be matched. This could be done by calculating the slope and intersection of the line described by the stub. The found lines give a point in the slope/intersection plane, where points close together indicate a track. Figure 8.2a shows a possible situation with two straight tracks. The points in figure 8.2b correspond to the stubs. A certain imprecision has to be expected. A track can be found by finding groups of two points close together. E.g. the blue and green points are close enough to be a track, where as the red point is alone and therefore a false stub.

The imprecision of the system has to be calculated from the characteristics of the sensors (e.g. strip pitch, distance between the modules).



(a) Test setup sketch with two tracks and one false stub. The dotted lines indicate the tracks and the short thick lines the stubs.

(b) Slope/Intersection plane with the points from the stubs.

Figure 8.2: Method for finding straight tracks.

8.2.2 Curved track finding method

A charged particle that passes a magnetic field is expected to have a circular track. The stubs are lines tangential to the circle described by the track. Following methods could be applied to correlate the stubs to circular tracks:

Associate circles Possible circles for each stub are calculated and matching circles for two stubs have to be found. This could be done by placing characteristics of the circle, like coordinate of the center and the radius, in a new space and looking for points close together. Similar to the slope/intersect plane of the straight lines.

Perpendicular bisector For each stub the perpendicular bisector is calculated. Then intersections between the different bisectors are located. Two stubs are on the same circle, if the distance from the stub to the intersection point is equal for both stubs. Figure 8.3 sketches a possible situation. Four distances between an intersection point and a stub are marked, where the pair d1, d2 mark a valid point and d3, d4 an invalid intersection.

Again a certain imprecision of the system has to be determined. This imprecision will lead to a variation of the direction of the bisectors and give a triangular shaped region, where the correct bisector is located. The green dashed lines indicate such an imprecision in figure 8.3.

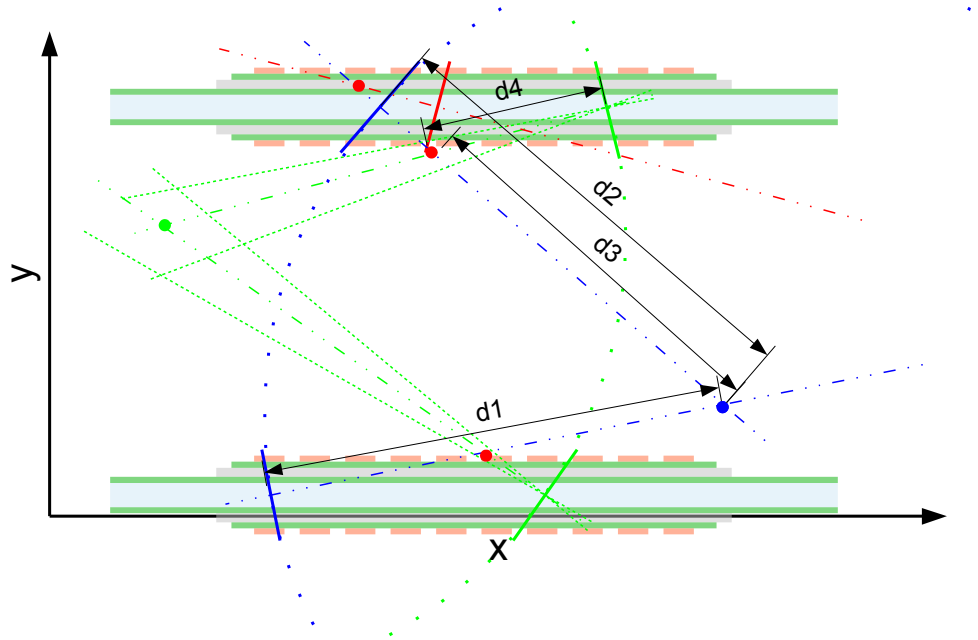


Figure 8.3: Method how curved tracks could be found with the perpendicular bisector. The green and blue dotted curves are two circular tracks with the corresponding stubs as short thick lines. The dash-dotted lines are the perpendicular bisecotrs and the points mark the intersections. The red points are wrong intersections.

8.2.3 Correlating in three dimensions

The two methods described above are working in one plane, i.e. matching together stubs from the same rows. Depending on the distance between the two doublets, it is also possible to have tracks going from one row to another. This will depend on the position of the modules and in which region of the ITK they are located. A simple solution could be to connect the doublet together depending in which range of the ηthey are located. Figure 8.4 illustrates this problem.

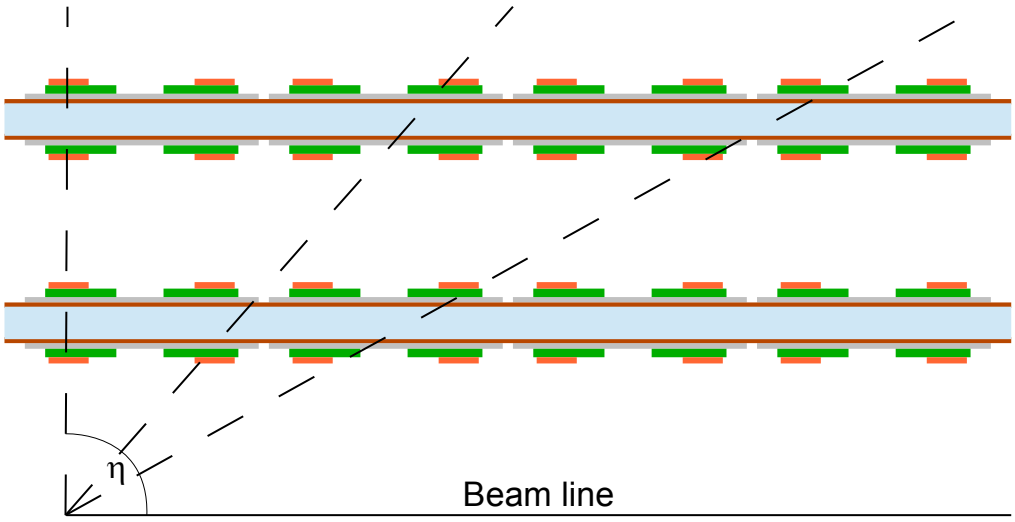


Figure 8.4: Illustration of track correlations across multiple modules.

9 Conclusion

9.1 Project goals

The main part of the work performed was on completing and testing the doublet correlator.

In the original project description was the development of a telescope and the track correlation logic. Finally the development of the telescope was only briefly looked into. A short concept was made, which includes some methods how a track correlator using two modules could be built.

The completion of the doublet took much more time than anticipated and a larger part of additional logic had to be developed. Therefore the schedule originally defined had to be abandoned. The measurements with test beam at SLAC gave a great opportunity to test the correlation logic in a real particle beam, but meant a lot of extra work. The analysis of the data measured at SLAC, together with the simulations done at LBNL concluded the work.

The amount of work required to complete the doublet correlator was underestimated in the initial analysis. The development of a track correlator was put aside and the focus was changed to analyze in detail the doublet.

9.2 Possible improvements

During the work of this thesis, the ABC130 chip could be tested in a test beam. Further the principle of the self-seeded triggering was proven to work. The system has still several points which can be improved. Most of this possible improvements were already discussed in sections 7.9.2 and 7.9.1. Summarized the improvement should be:

- For the doublet correlator:
  - improving the algorithm to reduce the number of false correlations.
  - optimize the correlator output block.
  - implement an interface with a higher data rate is required for a track correlator. The current implementation does not yet support the required transmission speed.
- For the data acquisition system:
  - make the system more stable in case of a high correlation rate.
  - integrate the different VIs for testing the modules, e.g. trim range test or response curve, into one application. This would improve the usability and simplify test of other modules built with a FCF readout.

- A future work should also look into the increase of the frequency for a 640MHz FastCLK. For this the interface board has to be redesigned with an other LVDS to SLVDS conversion. Right now there is no chip available, that could do such a conversion. An alternative method would be a ground offset for the support board an modules to obtain the correct common mode voltage at the differential inputs of the ABC130 chips. Regarding the FPGA design, most care has to be given to the deserialization of the FCF signals. This is the part of logic which has to work at 640MHz. The correlation logic has currently a path delay of around 20 ns. This is very close to the period of a 40MHz BC and it could be considered to add a buffer in the middle of the correlation logic, to create a pipeline.
- The threshold scan measurement should be repeated. This could be done at LBNL by capturing cosmic rays.

9.3 Achievement and results

This thesis contains all steps of a detector development. From the design of the readout electronics with the corresponding data acquisition system, over the measurements performed in a test beam to the analysis of the acquired data.

The tasks finally performed in case of this thesis are listed below. Contribution from other persons are also mentioned below.

- This work is a small part in a very complex machine, called ATLAS. The structure of the experiment was studied with a focus on the silicon strip detector. This also included to become familiar with the readout electronics.
- A new generation of the ATLAS binary readout chip, ABC130, was used in this thesis. This chip was designed by the ATLAS group. The data readout with the new fast cluster finder (FCF) was realized for the first time.
- An existing FPGA design, developed by L. Pirrami, to test the ABC130 chips was completed and improved. This contained an optimization of the FPGA internal clock distribution. The test system uses different PCB boards designed by E. Ropraz and S. Duner. All the FPGA programming was realized in VHDL.
- Starting on a basic correlation logic block, a full correlator for a doublet was implemented in the FPGA. The basic correlation logic was concepted and written by L. Pirrami. The doublet consists of two silicon strip sensors. Both sensors are read 10 ABC130 chips, assembled on a hybrid board.
- A correlator demonstrator was built during the execution of this thesis. The mechanical concept was designed by S. Diez-Cornell. The assembly and fabrication was done by C. Haber, H. Wang, R. Witharm and myself.
- A LabVIEW application was created to control and read the correlation logic. This application stores the correlations acquired in a *.root* file, which can be analyzed in the ROOT framework. The library to create the *.root* file was written by J. Stalder.

- The demonstrator was tested in a test beam at the Stanford Linear Accelerator Center (SLAC). The test beam was organized by S. Diez-Cornell (DESY, former LBNL) and A. Blue (University of Glasgow). The installation of the experiments and the measurements were performed by a group formed of the following people: C. Haber (LBNL), J. Keller (DESY), J. Pasner (UC Santa Cruz), J. Stalder (LBNL), A. Risbud (LBNL), H. Wang (LBNL) and myself (LBNL). More people were involved in the background and supported the test beam experiments from distance or on place, which are M. Warren(University College London), B. Gallop(RAL), P. Phillips (RAL) and M. Swiatlowski (SLAC).

- The acquisition of the trigger signal from the TLU had to be integrated in the FPGA design. This gave the possibility to match the data from the correlator together with the other detectors used.

- VIs were programmed to perform characterisation tests on the full hybrid board. These programs also create *.root* files which can be analyzed by the SCTDAQ. The macros to control the SCTDAQ utilities were created by J. Stalder.

- Characterization measurements were done on the complete doublet with the developped VIs. A similar performance was observed as in case of the single chip board, characterized by J. Stalder.

- The test beam data were analyzed in ROOT. The data show a correct function of the correlation logic. Wrong correlations were observed and the source was further investigated.

- A threshold scan was performed during the test beam. The results from this scan are not yet fully understood. They are different from the single chip measurements.

- With a simple simulation the address distribution coming from the FCF logic was studied and the test beam data could be verified.

- With the generation of random masks, to create correlations, the source of the wrong correlations was investigated. Several improvement for the correlation logic were seen and an approach to implement those is described.

- A basic concept for a track correlator was developed. This concept can be used as a starting point in another work.

The test beam performed during this thesis was also the first time the ABC130 was used in a particle beam. The results from this beam experiment can help to understand better the chip.

The correlation logic for a self-seeded trigger was proven to work in a test beam. It was possible to acquire captured correlations without external trigger signals, using the new fast readout of the ABC130 chips.

I, Niklaus Lehmann, hereby sign and confirm that I've written this thesis without using other than the listed resources.

August 15, 2014 Berkeley

Bibliography

[1] CERN, "First three-year LHC running period reaches a conclusion," March 2014. http://press.web.cern.ch/press-releases/2013/02/first-three-year-lhc-running-period-reaches-conclusion.

[2] CERN, "ATLAS Letter of Intent Phase-II Upgrade," *LHCC-I-023*, December 2012.

[3] L. Pirrami, "Fast triggering sensor module for the Large Hardron Collider," Master Thesis, Ecole d'Ingénieurs et d'Architectes de Fribourg part of the University of Applied Science Western Switzerland, February 2014.

[4] S. Duner, "Trigger Functions for the High Luminosity LHC," Bachelor Thesis, Ecole d'Ingénieurs et d'Architectes de Fribourg part of the University of Applied Science Western Switzerland, August 2013.

[5] E. Ropraz, "Sensor Module for the High Luminosity LHC," Bachelor Thesis, Ecole d'Ingénieurs et d'Architectes de Fribourg part of the University of Applied Science Western Switzerland, August 2013.

[6] S. Diez-Cornell, "(My) highlights of ATLAS Phase II strips tracker meeting at Valencia," February 2014.

[7] CERN, "The Large Hadron Collider," March 2014. http://home.web.cern.ch/topics/large-hadron-collider.

[8] CERN, "ATLAS fact sheet," 2011.

[9] CERN, "ATLAS experiment," March 2014. http://atlas.ch/.

[10] S. Pirner, "Intelligent Filtering Algorithms for an upgraded Inner Tracker at ATLAS," Bachelor Thesis, Fakultät für Physik und Astronomie, Ruprecht-Karls-Universität Heidelberg, 2010.

[11] M. Defferard, "Sensor Module Studies," Bachelor Thesis, Ecole d'Ingénieurs et d'Architectes de Fribourg part of the University of Applied Science Western Switzerland, 2012.

[12] ATLAS SCT collaboration, "Supply of Silicon Microstrip Sensors of ATLAS07 specification," October 2007. Techical Specification.

[13] (Particle Data Groupe) J. Beringer *et al.*, "Review of Particle Physics," *Phys. Rev. D86, 010001*, 2012. and 2013 partial update for the 2014 edition.

[14] H. Spieler, *Semiconductor Detector Systems*. Oxford Science Publications, 2005.

[15] J. Stalder, "Analysis Tool of High Speed data readout," Bachelor Thesis, Ecole d'Ingénieurs et d'Architectes de Fribourg part of the University of Applied Science Western Switzerland, August 2014.

[16] CERN, "The HL-LHC project," March 2014. http://hilumilhc.web.cern.ch/HiLumiLHC/about/.

[17] T. Affolder *et al.*, "System Electronics for the ATLAS Upgraded Strip Detector," *Release 2.0*, February 2013.

[18] T. Wengler, "The ATLAS Upgrade Programme," ECFA High Luminosity LHC Experiments Workshop, Aix-les-Bains, France, October 2013.

[19] A. Schöning, "A self seeded first level track trigger for ATLAS," Workshop on Intelligent Trackers (WIT2012), May 2012.

[20] ATLAS group, "ABC130 specification," 2013. 4.6Draft.

[21] B. M. Demirköz, *Construction and Performance of the ATLAS SCT*. PhD Thesis, Balliol College, Oxford University, April 2007.

[22] XILINX, "ML605 Hardware User Guide UG534," October 2012. v1.8.

[23] Texas Instruments, "AC-Coupling Between Differential LVPECL, LVDS, HSTL and CML," October 2007. Application Report SCAA059C.

[24] B. Gallop, "ABC130 Testing," Valencia - ATLAS strips, February 2014.

[25] SLAC, "What is LCLS," July 2014. http://lcls.slac.stanford.edu/WhatIsLCLS_1.aspx.

Glossary

ABC130	new block Binary Chip in 130nm technology
ACSR	access control and status register
ADC	analog to digital converter
ALICE	A Large Ion Collider Experiment
AMUX	analog multiplexer Output on ABC130
ASCII	American Standard Code for Information Interchange, character encoding
ASIC	application specific integrated circuit
ATLAS	A Toroidal LHC ApparatuS
BC	beam crossing clock signal (defined 40 MHz, used 20 MHz for correlator tester)
BCID	beam crossing ID
CERN	European Organization for Nuclear Research
CMS	Compact Muon Solenoid
C_{ox}	Oxide capacitance on the silicon strip sensors
DESY	Deutsches Elektronen-Synchrotron
doublet	assembly of two silicon detector modules back to back
ENC	equivalent noise charge
EOS	end of stave board
EPC	external peripheral controller
EPFL	Ecole Polytéchnique Fédérale de Lausanne
ESA	end station A
ESTB	end station (A) test beam
EUDET	European Detectors, European program for detector research and development
FastCLK	Fast cock for FCF output lines
FCF	fast cluster finder
FEE	front end electronics
FIB	focused ion beam
FIFO	first in / first out
FPGA	field programmable gate array

HCC	hybrid control chip
HL-LHC	high luminosity LHC
HV	High Voltage to deplete the silicon detector
IB	interface board
ID	inner detector
IDELAY	Input delay block for FPGA input pins
ITK	inner tracker
L0	Level 0 Trigger
L0ID	Level 0 ID
L1	Level 1 Trigger
L2	Level 2 Trigger
LBNL	Lawrence Berkeley National Laboratory
LCLS	Linac coherent light source
LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty
LSb	least significant bit
LVDS	low voltage differential signaling
MB	MicroBlaze
MMCM	mixed-mode clock manager
µm-pos	micrometer position on the doublet demonstrator sort
MSb	most significant bit
ODELAY	Output delay block for FPGA output pins
PCB	printed circuit board
P_T	transverse momentum of particles
R3	regional readout request
R3L1	Buffer for events selected by R3 or L1 triggers
RAL	Rutherford Appelton Laboratories
RoI	region of interest
ROOT	Data analysis framework
.root	File format for a data file used by ROOT
RTL	register transfer level
SCT	semiconductor tracker
SCTDAQ	silicon tracker data acquisition software
S-curve	Resulting plot from the threshold scan
sensor	Usually means a silicon strip sensor, containing multiple strip elements.

σ	Output noise
SIPO	Serial In/Parallel Out
SLAC	Stanford Linear Accelerator Center
SLVDS	sub-LVDS
SMD	surface mount devices
strip	One sensing element of a silicon particle detector. Shaped like a line, hence the name strip.
stub	Part of a particle track, built from two hits of a doublet
tag	binary vector with information about correlations
TLU	trigger logic unit
TRT	transition radiation tracker
VHDL	VHSIC (very high speed integrated circuit) hardware description language
VI	virtual instrument
V_{T50}	Threshold voltage with a 50% hit rate for an input charge

List of Figures

List of Tables

A Planning

This appendix present the original schedule made at the beginning of the work. The text was originally intended for section 1.3.

In the first weeks the focus will be set mostly on part 1 of the work. This helps directly to get familiar with the system and the used technologies. In parallel research can be done on the state of the art for part 2. With the progress of the work, the focus will be set more and more on part 2. For more details see the planned schedule in figure A.1.

The planning is made until the hand date in of the report. Because the presentation is also given at LBNL, there is an additional month, which will be used to complete loose ends of the work.

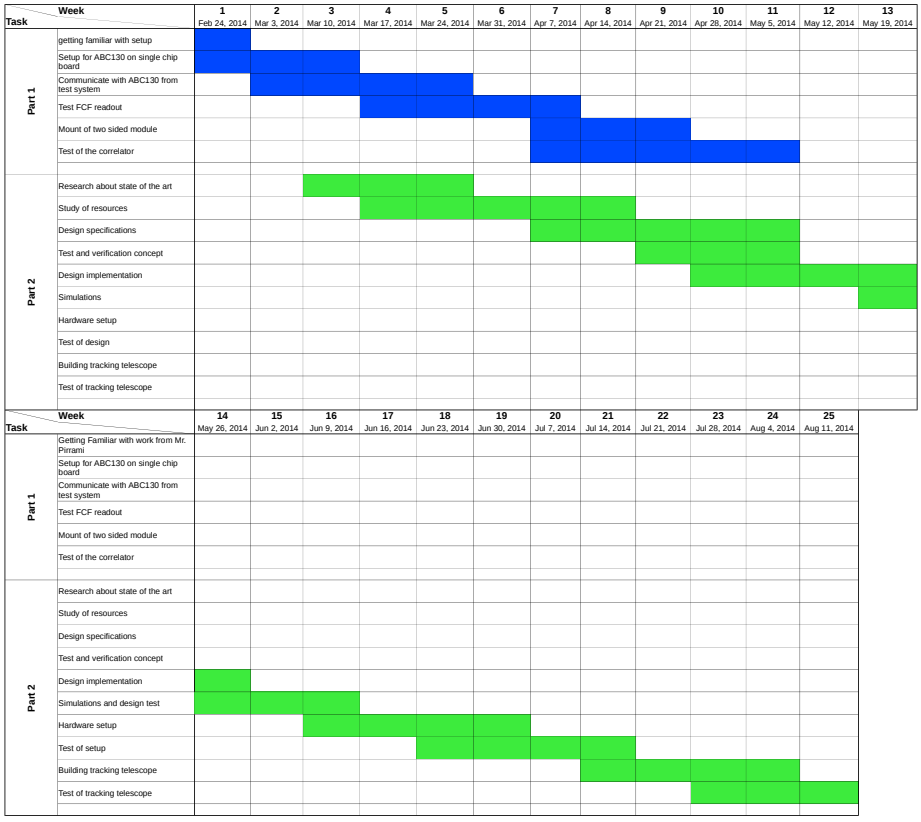


Figure A.1: Planned time schedule for the project

B Doublet assembly

During this thesis, two hybrids were mounted on a ATALS07 silicon strip sensor. These two modules were used to build the correlator demonstrator, called doublet.

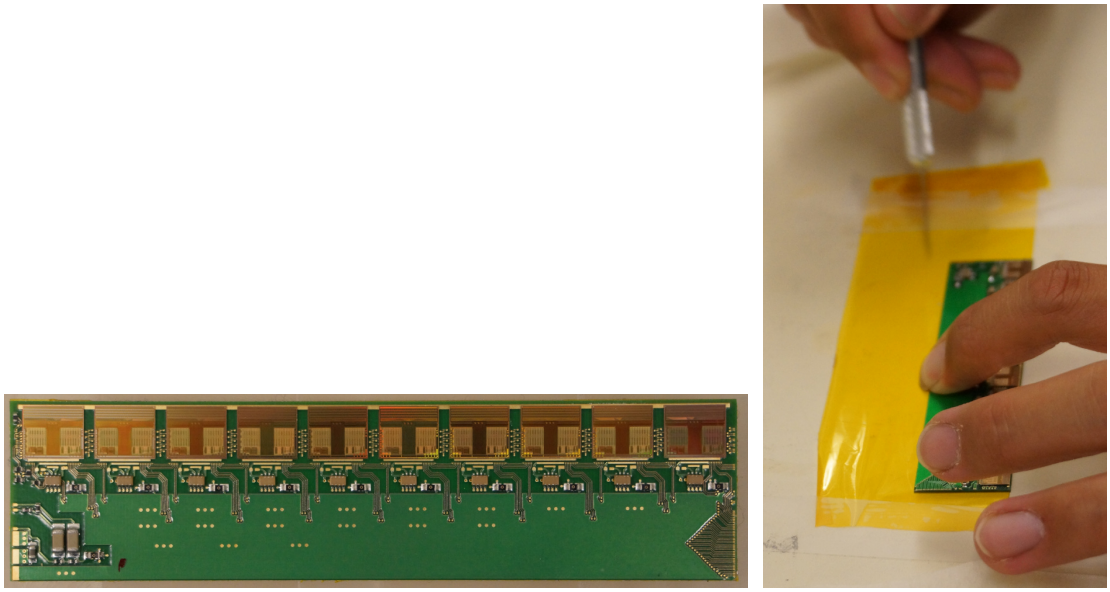
Because this was the first time such a doublet was constructed, only minimal fixtures were available and most alignment was done by hand.

B.1 Gluing of the hybrid

B.1.1 Preparation

The FCF hybrid was fully mounted and the ABC130 chips bonded. To test the function of the chips, the hybrid was bonded to a support board. After the successful test of the chips, it was removed again.

Before starting the glue application, the back of the hybrid had to be covered with kapton tape to insulate the back copper plane.



(a) Fully assembled hybrid before gluing. (b) Insulate backside

Figure B.1: The hybrid board and cutting of the kapton tape glued to the back of the hybrid.

B.1.2 Glue application

A glue mask is attached to the back of the hybrid. The mask consists of a two layered tape. The first layer is a 125 µm thick kapton polyamid tape. This is covered by a second 110 µm thick frisket film. The second layer will be completely removed, while the outer part of the mask, called from hereon frame, will stay on the hybrid. The inner part is removed and leaves only a specified amount of glue. The frame is used to prevent the glue from spoiling over and also to define the distance between the sensor and hybrid.

Figure B.2: Mask for the glue attached to back of the hybrid. The wings in the four corners have to be removed. The frame on the border of the hybrid will stay. The inner part will be removed after the glue is applied.

The two component epoxy glue is then spread over the mask and the extend is removed with a razor blade. After applying the glue, the thin layer of the mask is removed with the inner part. Only the frame remains together with the glue on the hybrid.

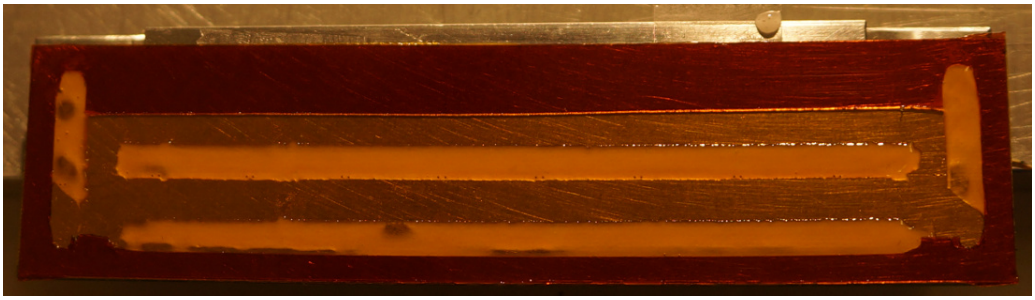


Figure B.3: Glue applied on hybrid and the mask removed.

B.1.3 Placing hybrid on sensor

The sensor is held on a vacuum fixture. A printout of the support board is attached on the fixture, to give an indication of the location where the hybrid has to be placed.

The hybrid is then placed and aligned by hand. The width of the hybrid is the same as the sensor. The edges are moved to match on both sides. For further alignment, the front edge of the hybrid (where the ABC130 chips are located) is placed as parallel as possible to the bonding pads on the sensor.

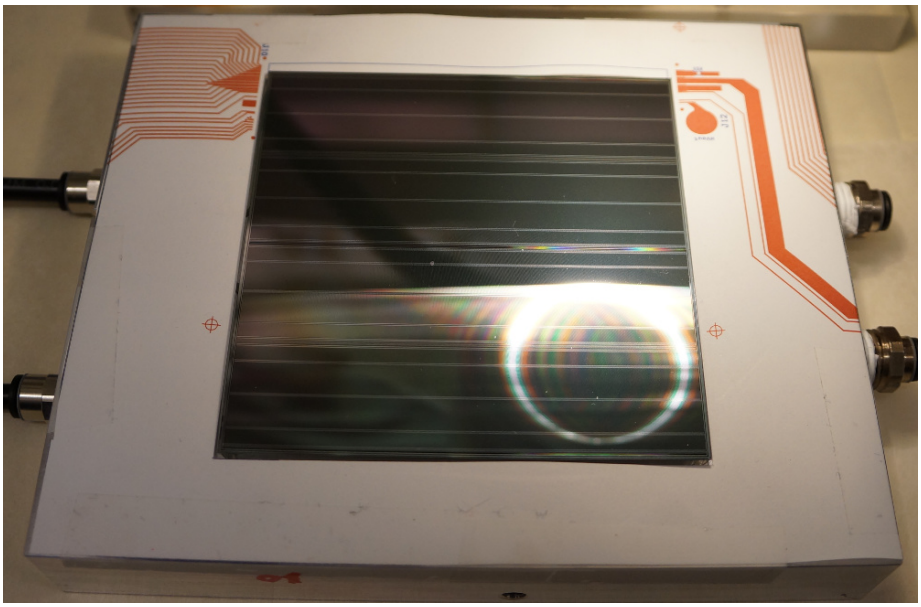


Figure B.4: Sensor placed on vacuum jig.

Instead of leaving the gap between the hybrid edge and the first row of bonding pads, visible in figure B.5, the hybrid should be closer to the pads or actually almost cover them. This is to reduce the distance to the bonding pads on the second row to have less problems during the wirebonding.

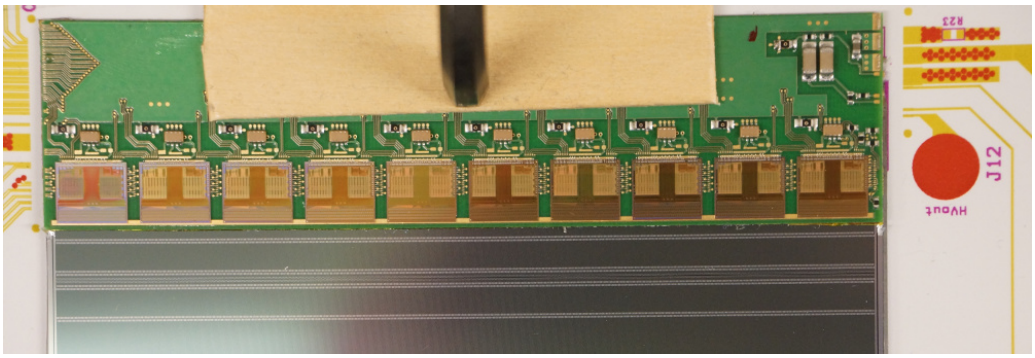


Figure B.5: Hybrid board placed on sensor.

To keep the hybrid from bending upwards in the center, a weight has to be applied in the middle of the hybrid. This was done with an improvised use of an allan wrench. The wooden piece protects the hybrid and distributes the weight more equally.

Once the glue was cured, the wiredbonds were made from the ABC130 to the strips to complete the module.

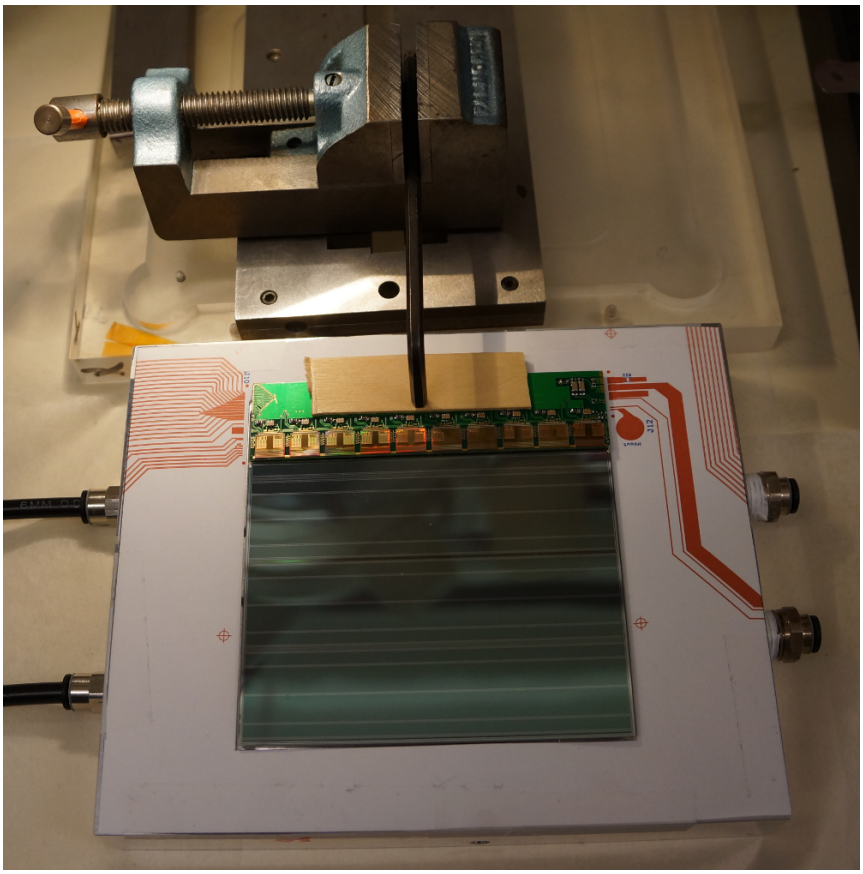


Figure B.6: A weight was applied in the center of the hybrid to keep it in place and to prevent upwards bending of the center.

B.2 Glue modules to support board

The completely bonded modules have to be glued to the HV frame of the doublet frame. For this process a conducting silver epoxy glue has to be used. It is also possible to apply the conducting epoxy only on part of the HV frame and use 5 minute epoxy for the rest.

Before starting to apply the glue, the plastic pins have to be put into the corresponding holes. These holes were made by drilling to both sides together and are used to align the two sensors. The pins are marked with a red circle in figure B.7.

The glue should be placed on the inner edge of the frame. This is to prevent glue coming up on the side of the sensor.

The module is placed on the frame and then pushed towards the pins. The pins help to find the right alignment. The module can be slightly pushed down with a cleanroom swab to spread the glue.

No weight is applied during the curing of the glue. The final step consists in makeing the wirebonds from the hybrid to the support board.

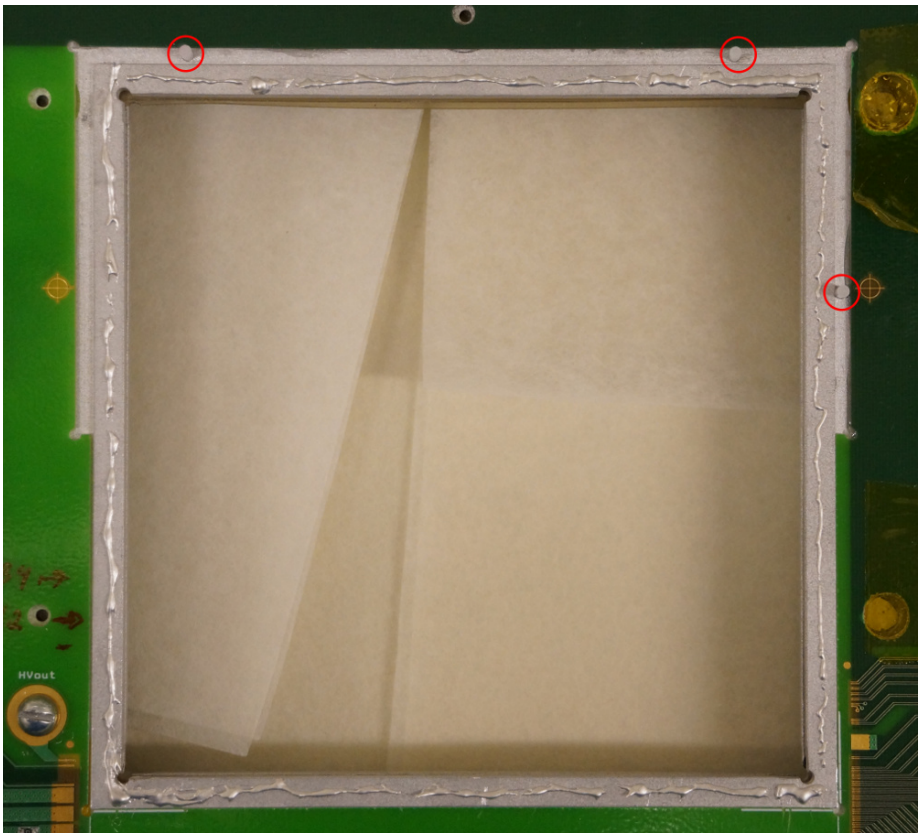


Figure B.7: Silver epoxy applied on the HV frame to make contact between the back plane of the sensor and the bias voltage.

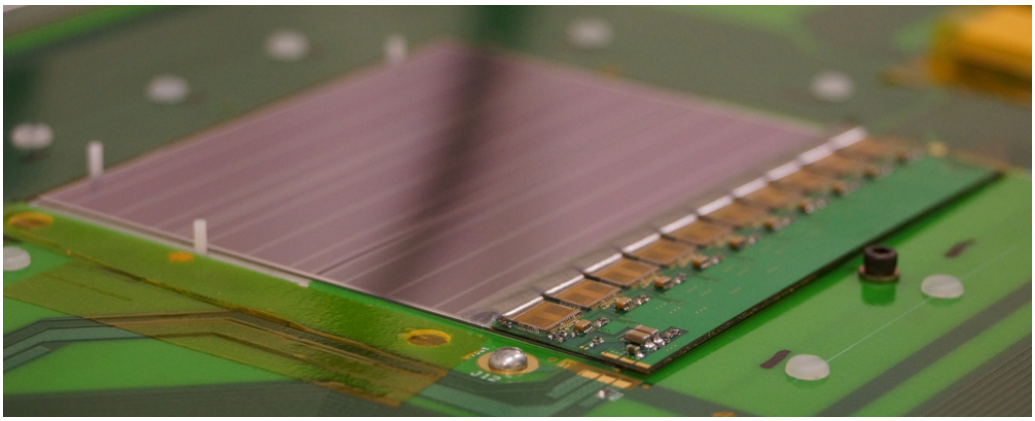


Figure B.8: Module glued to the HV frame in the center of the support board.

C Correlator DAQ user manual

Here is a brief manual on how to control the correlator data acquisition software.

All paths are given relative to the folder *C:\Users\naskue\Desktop\Correlator* on the computer used at LBNL.

C.1 Configuration of the FPGA

C.1.1 Programming the FPGA

The FPGA is configured with the *iMPACT*. The JTAG chain has to be initialized and the devices configured. If the project file *.\FPGA\Impact\Correlator.ipf* is opened, everything should be already set.

To program the FPGA, right click on the xc6vlx240t device and select program, as shown in figure C.1. A blue field with **Programming successful** should appear.

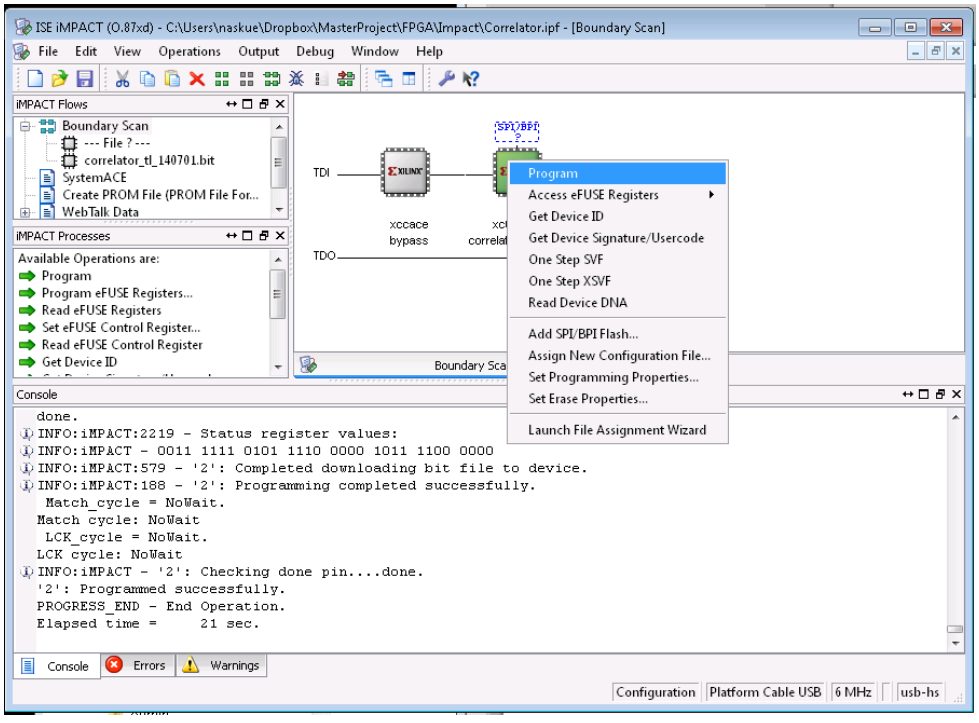


Figure C.1: View of iMPACT

C.1.2 Starting the MicroBlaze processor

After programming the FPGA, the C program has to be loaded into the memory of the MicroBlaze softcore processor. This is done with the *Xilinx SDK*.

Open the workspace *.\FPGA\FPGA_design\ABC_Correlator\MB_Correlator_Prog*.

To start the execution of the program, press on the drop down symbol next to the green arrow and select **Correlator_Control**.

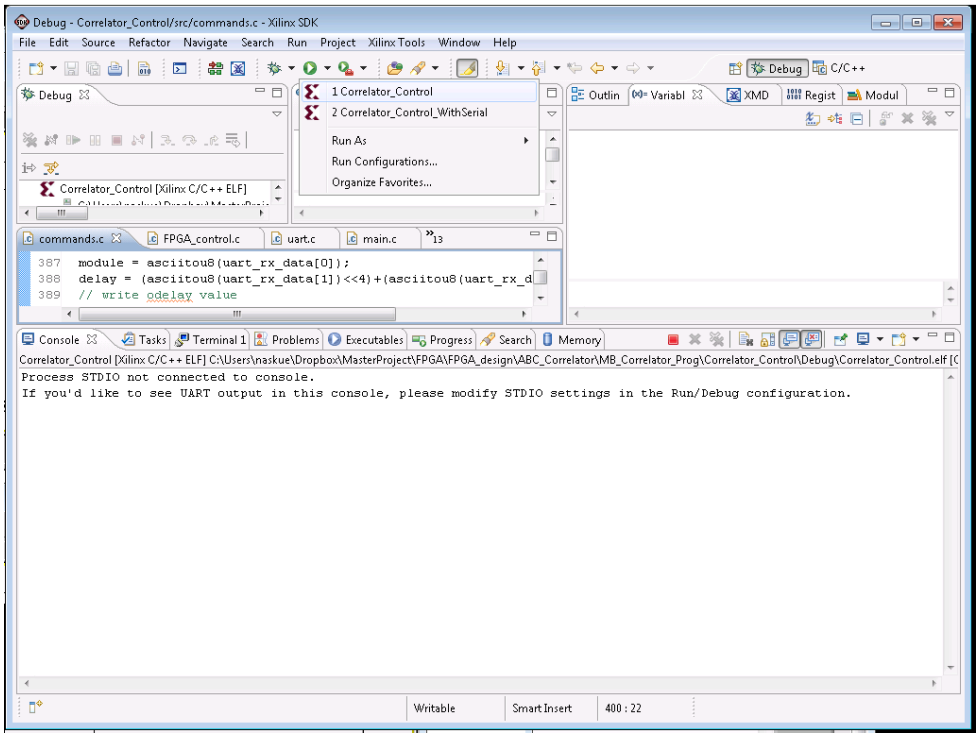


Figure C.2: Launch execution in Xilinx SDK

C.2 Correlator Tester LabVIEW program

The LabVIEW application is located at *.\LabVIEW\Correlator_test\Correlator_test.vi*.

The application has three different tabs described below.

C.2.1 Configuration tab

The configuration tab allows to change different settings on the ABC chips or the FPGA. Figure C.3 shows this tab. Following a list with a brief description of the diffent controls:

- **Parameters** for plotting the correlations
  - *delta y*: distance between the two modules

  - *half pitch*: 1/2 of the module strip pitch

- **Correlator control register** to configure the correlation and FCF readout logic in the FPGA
  - *Correlator*: enables or disables the output of correlation data
  - *FCF readout*: enables or disables the output of FCF data after a calibration pulse or trigger signal
  - *FCF autotrigger*: enables the automatic storage of FCF data
  - *Side, ABC number*: selects which chip is read out

- **Trigger control register**
  - *EUDET trigger*: activates the logic to read and confirm the trigger signal from the TLU

- **ABC settings**
  - *ABC130 chip, Module*: which chip are controlled
  - *ADC1, ADC2, ADC3*: setting for the ADC registers. *BVT* is the most important setting, which controls the threshold.

- *Event Readout*: Select how the data is read back. If enabled, correlations are grouped to events sent out after on each trigger. If disabled, events and triggers are returned independently. The measurements should be taken with event readout.

C.2.2 Data acquisition tab

In the data acquisition tab, the found correlations are shown if a run is started and if **auto plot** is enabled. **Append plots** sets if the correlations are accumulated on the plots. The plots can be cleared by deactivating and reactivating either **Auto Plot** or **Append plots**.

To reduce the amount of memory used by the program, it's better to switch off the **Auto Plot** when not required.

C.2.3 Console tab

The console tab is used to communicate with the FPGA. In the log field, all sent commands are listed. The responses are only logged, if the box **Take Log of incomming data** is activated.

The entry **User Command** allows to send manually commands to the FPGA. It can be selected by pressing F1. Typing `hh` into the **User Command** field should send back a list of all possible commands and how they could be used.

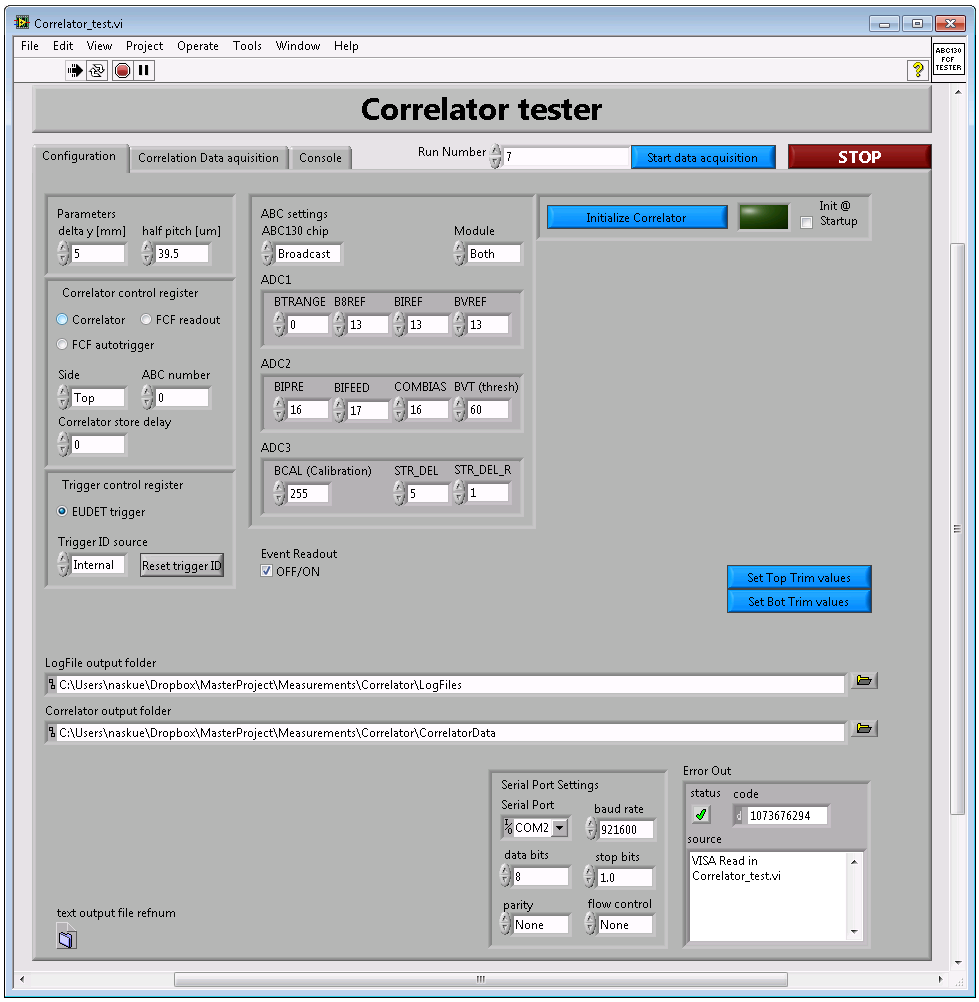


Figure C.3: View of the configuration tab

C.3 Check if system is still working

C.3.1 Check response from FPGA

To quickly check if the FPGA is still running correctly the following steps can be executed:

1. In the configuration tab, un-check the boxes **Correlator**, **FCF autotrigger** and **EUDET trigger**

2. Make sure that **FCF readout** is checked.

3. In the console tab, set the box **Take Log of incomming data**

4. type *tf* in the field **User Command**

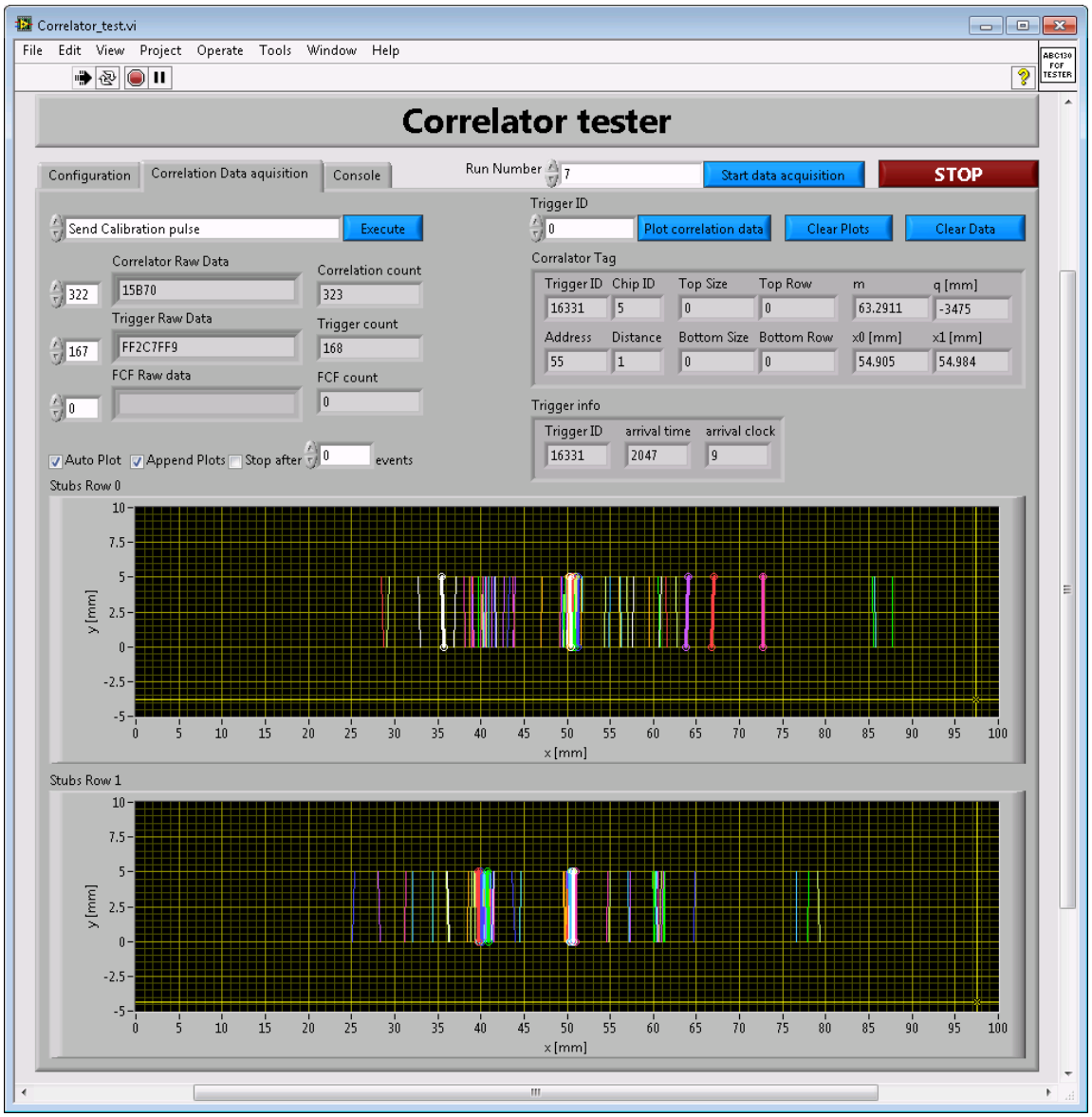


Figure C.4: View of the data acquisition tab

5. a list of all chips with the values `FFFFFFFF` like in the middle of figure C.5 show be plotted.

The `tf` command reads all FCF lines, which will contain no valid data as long as the mask registers are not set, which should be the case.

Setting or clearing each checkbox in the configuration tab should send a command to the FPGA which should be acknowledged by *ack*. This appears in the log only if the **take log** box is active.

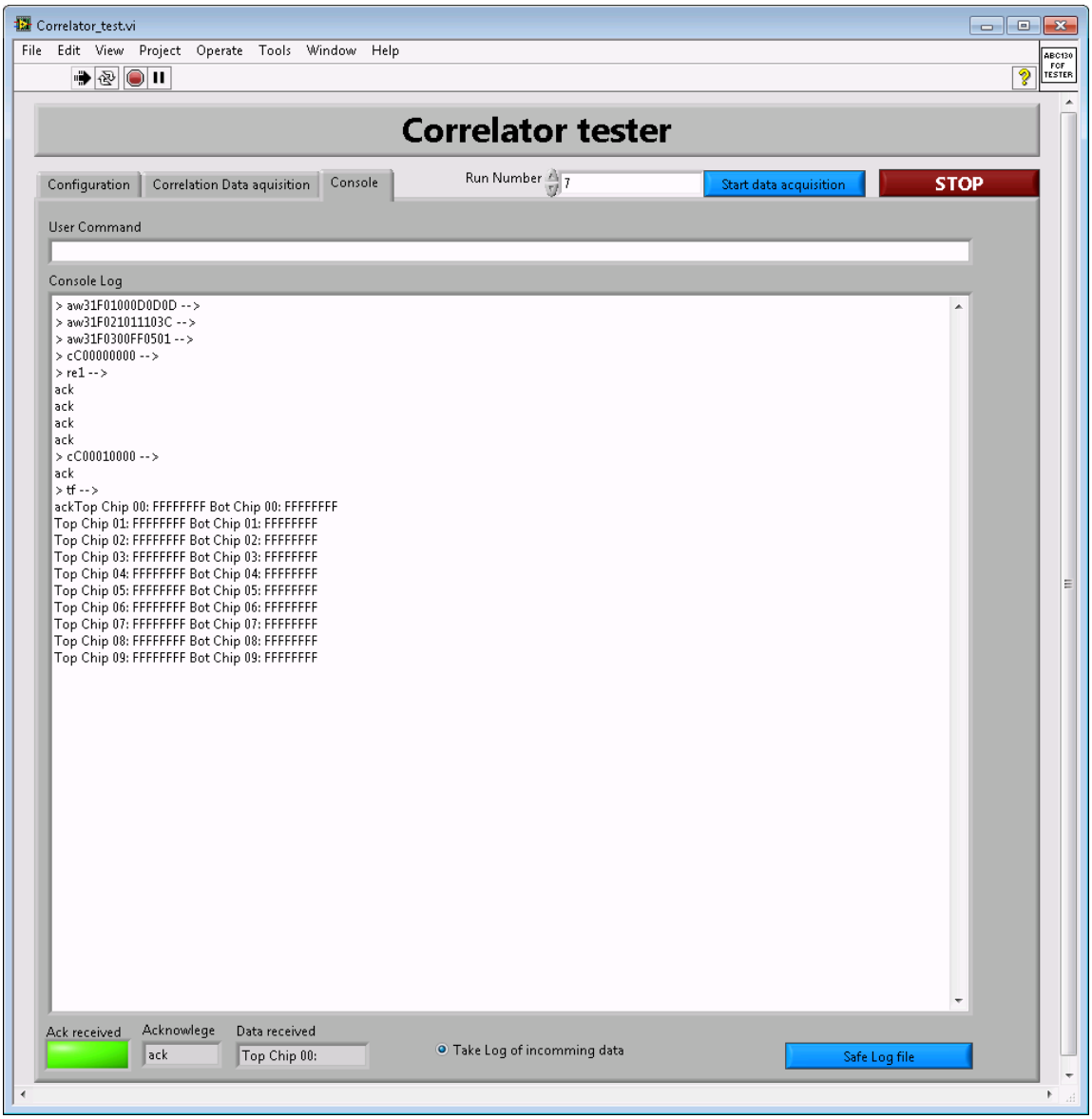


Figure C.5: View of the console tab

C.3.2 Check if correlations are found

When the beam is running correlations should be found and are sent back automatically. The readout is activated with the **Correlator** box in the configuration tab (figure C.3). The **Event readout** should be off to see the correlations without a trigger.

If **Take Log** is active, lines in the format of `ackc6710EB03` should appear for the direct readout mode. The 8 digit hex number should change depending on where the correlaton is found.

In case of the event readout, the format is something like `acke1A9BFFF20DF800CF871307012027127E317`

The length varies with the number of correlations found. For the event readout to work properly, the TLU has to be active and sending trigers.

C.3.3 Check if triggers are arriving

To receive triggers from the TLU the checkbox **EUDET trigger** has to be active. If the TLU is sending triggers, then data packages like `ackt19033FF7` should arrive, only in the direct readout mode. The 8 hex digits hold the trigger information.

C.4 Start a new data run

A new data acquisition run can be started by pressing the button **Start data acquisition**. The dialog in figure C.6 will pop up.

The acquisition will create two files. A *.root* file, which contains the correlations, and a *.txt* file for the received triggers. For both files some parameters can be entered, which will be stored together with the data. Check the fields and change them if needed. The threshold should be automatically taken from the configuration tab and the run number is incremented whenever a new run is started.

If **Event controlled stop** is activated, the acquisition will automatically stop after the entered number of received trigger signals. If no trigger is arriving, this auto stop will not work.

Once the the data is started, the correlations can be plotted in the data acquisition tab (figure C.4). This can be used for a quick check, but should not be active for a longer meausrement.

During the data acquisition the **FCF readout** and **FCF autotrigger** are deactivated automatically and the correlator is enabled. The **Take log** in the console tab, should be deactivated

Also **EUDET trigger** has to be activated to capture the triggers from the TLU.

A run can be stopped either after a certain number of triggers or by pressing on the **Start data acquisition** button again. During a run it reads **recording...** though. When the run is stopped, the correlator is deactivated.

The dialog from figure C.7 will pop up. An additional remark can be added to the text file, if wanted.

C.5 Initialize the system

To initialize the whole system, all three procedures below should be executed.

After a reset of the ABC130s,

C.5.1 Set the phase shift for the FCF readout

The FPGA contains logic to adjust for the delays on the transmission lines. The easiest way to do this is with the console in LabVIEW. Below are the commands listed that should

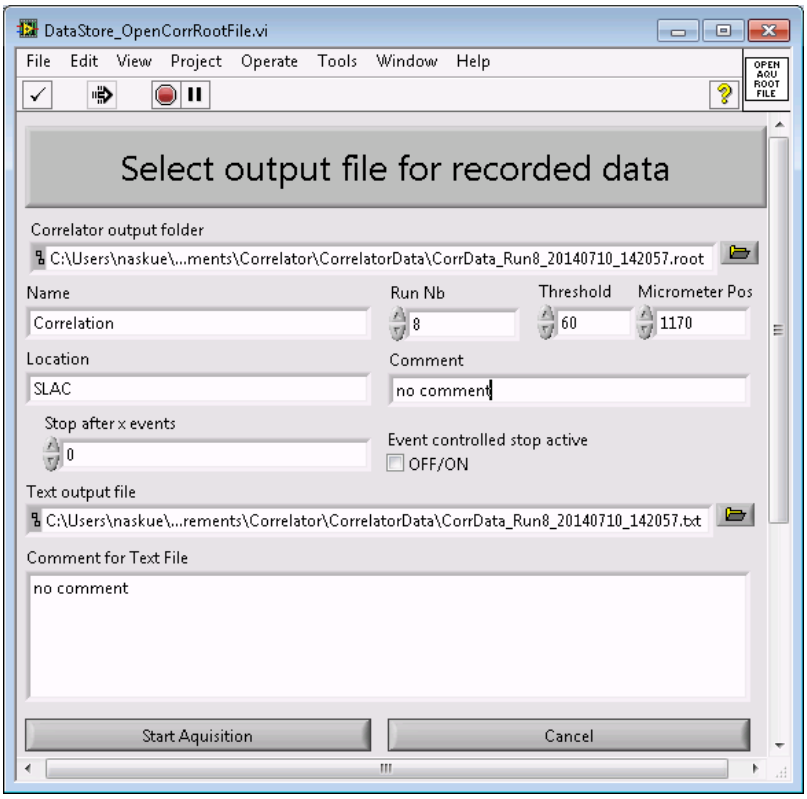


Figure C.6: Dialog for new data run

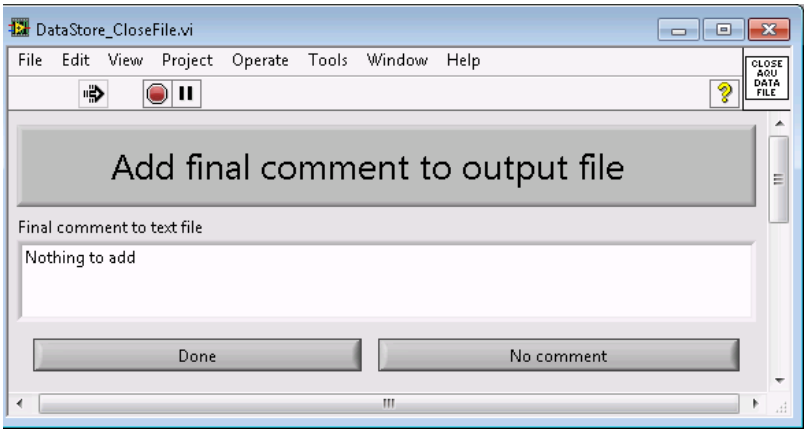


Figure C.7: Dialog at end of data run

work for the setup at SLAC, with the expected results:

1. `rs3o` : reset all chips. Returns `ack`.
2. `so206` : adjust the output clock on the bottom module. Returns `ack`.

3. `ps19p` : adjust phase shift and IDELAYs for top module. Prints a list with two entries for chips 0 to 8 and `ack` together with a 8 digit hex number.
4. `ps28p` : adjust phase shift and IDELAYs for top module. Prints a list with two entries for chips 0 to 7 & 9 and `ack` together with a 8 digit hex number.
5. `tf` : check if phase shift was correct. Every chip should read `AAAAAAAA`.

C.5.2 Trim the ABC chips

To set the trim values for all chips, a separate LabVIEW programm has to be executed.

1. Open *.\LabVIEW\Correlator_test\ABC130_WriteTrimValuesFromFile.vi*
2. Select **Module** = Top (to select the static sensor)
3. Run (press arrow button or CTRL+R)
4. Select the file *.\Measurements\FCF Doublet (static side)_tr1_20140707.trim*
5. Click ok and wait until finished
6. Select **Module** = Bottom (to select the mobile sensor)
7. Run (press arrow button or CTRL+R)
8. Select the file *.\Measurements\FCF doublet (mobile side)_tr1_20140706.trim*
9. Click ok and wait until finished

C.5.3 Set the chips to active

After the ABC130s are trimmed, the mask register has to be cleared, so that all the channels can be read. This is also done by entering a command into the LabVIEW console:

1. `cc031f` : clear mask on all chips

C.6 Errors

To reconfigure the system, in case of a freezing or no more responses, following steps can be tried.

C.6.1 LabVIEW

The LabVIEW execution should normally be stopped, by pressing the big **Stop** button. If it doesn't respond anymore, the execution can also be canceled by pressing the small red round button, with the mouse over text *Abort execution.*

To restart, just press the arrow button or press CTRL+R.

C.6.2 MicroBlaze

If the FPGA is not responding to the commands, it can be restarted with the SDK. This can be done with launching the execution as described in C.1.2. On the bottom right of the window is a progress bar.

C.6.3 ABC130

To reset the ABC130, send the command `rs3o` from the console tab in the LabVIEW application. This issues a soft reset on all chips and therefore they have to be reconfigured.

C.6.4 FPGA

It can happen, that the MicroBlaze is not responding anymore at all and progress bar in SDK stays at 27%. In that case, it's necessary to reprogram the FPGA, as described in section C.1.1.

If the FPGA was reprogrammed, the initialization procedure has to be redone. See section C.5.

D Test beam run table

Table D.1 lists the parameters of the runs performed during the test beam, at least those who were used in the figures.

More and more parameters were noted with progressing time of the test beam. Not everything was noted for all the runs.

As described in section 7.6, it was necessary to fix the moving mechanics. Therefore the original position of the micrometer was lost and a new "zero" position was defined. This change was done on the July 14. So all runs after this date have the corrected position and the runs before have the uncorrected position.

The value **Beam E** gives a rough indication of the beam intensity. It is a value read from an application listing different beam parameters and was provided by SLAC.

The threshold (**BVT**) is given in DAC counts and if only one value is written in the table, it is applied to both sides. In case of two values, there was a different threshold applied to the modules. The notation used is top/bottom.

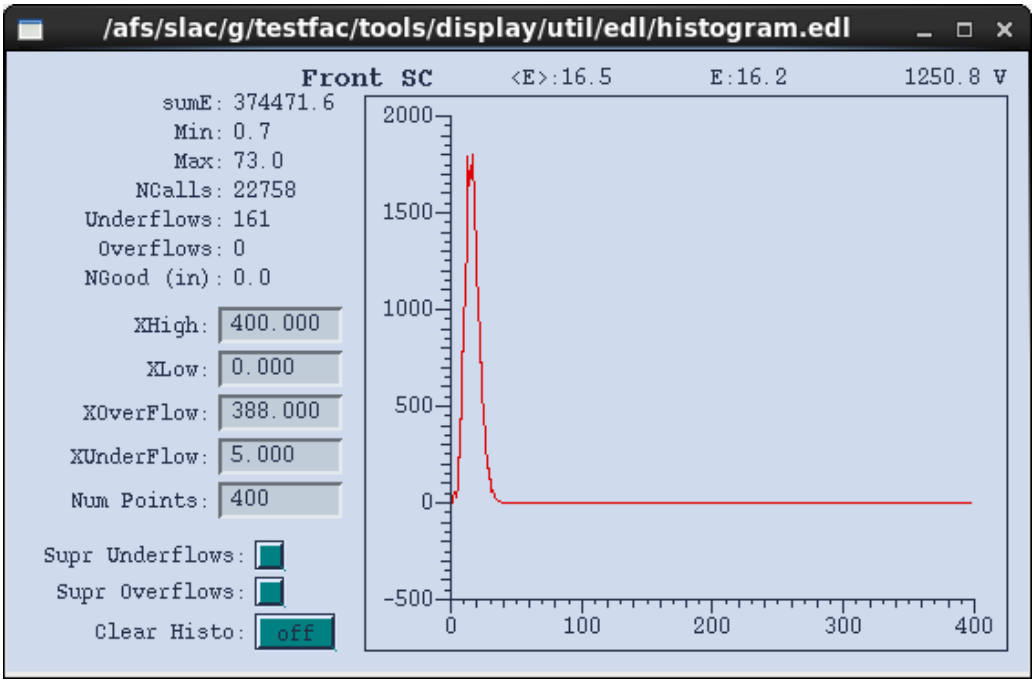


Figure D.1: Screenshot of the window with the E value for the beam, indicating approximately the beam intensity.

Date	Run Nb	Events	µm-pos	BVT	Telescope Run Nb	Beam E	Comment
Jul. 10	9	1000	1170	40			
Jul. 10	10	1000	1170	40			
Jul. 10	11	1000	1170	40			
Jul. 10	12	1000	1170	40			
Jul. 10	15	4000	1170	50			
Jul. 10	17	4000	1170	60			
Jul. 10	18	4000	1170	70			
Jul. 10	19	4000	1170	80			
Jul. 10	20	4000	1170	90			
Jul. 10	21	4000	1170	100			
Jul. 10	23	4000	1170	110			
Jul. 10	24	4000	1170	120			
Jul. 10	27	4000	1170	130			
Jul. 10	28	4000	1170	140			
Jul. 10	30	4000	1170	150			
Jul. 10	31	4000	1170	160			
Jul. 10	33	4000	1170	170			
Jul. 14	33	1050	4450	90			fix of moving mechanics
Jul. 14	39	2500	3745	40/80	1473	60.3	
Jul. 14	40	2500	3745	40/100	1474	60.4	

Continued on next page

Table D.1 – continued from previous page

Date	Run Nb	Events	µm-pos	BVT	Telescope Run Nb	Beam E	Comment
Jul. 14	41	2500	3745	40/120	1475	60.1	
Jul. 14	42	2500	3745	40/140	1476	60.1	
Jul. 14	43	2500	3745	40/160	1477	60.1	
Jul. 14	44	2500	3745	40/180	1478	60.2	
Jul. 14	45	2500	3745	40/60	1479	60.3	
Jul. 15	52	3000	3745	120	1485	39.8	
Jul. 15	55	3000	3670	120	1487	39.5	
Jul. 15	56	3000	3670	110	1488	39.4	
Jul. 15	57	3000	3590	120	1489	38.8	
Jul. 15	58	3000	3590	110	1490	39.0	
Jul. 15	59	3000	3510	110	1491	39.0	
Jul. 15	60	3000	3510	120	1492	39.6	
Jul. 15	61	3000	3830	120	1493	38.0	
Jul. 15	62	3000	3830	110	1494	38.7	
Jul. 15	63	3000	3910	120	1495	39.5	
Jul. 15	64	3000	3910	110	1496	39.5	
Jul. 15	65	3000	3990	120	1497	39.7	
Jul. 15	66	3000	3990	110	1498	39.5	
Jul. 15	67	3000	4070	120	1499	40.8	
Jul. 15	68	3000	4070	110	1500	39.3	

Continued on next page

Table D.1 – continued from previous page

Date	Run Nb	Events	µm-pos	BVT	Telescope Run Nb	Beam E	Comment
Jul. 15	69	3000	4150	120	1501	39.4	
Jul. 15	70	3000	4150	110	1502	39.3	
Jul. 15	71	3000	3745	110	1503	39.4	
Jul. 15	72	2500	3745	180	1504	39.3	
Jul. 15	73	2500	3745	160	1505	39.3	
Jul. 15	74	2500	3745	140	1505	39.4	
Jul. 15	75	2500	3745	120	1506	39.5	
Jul. 15	76	2500	3745	100	1506	39.5	
Jul. 15	77	2500	3745	80	1507	39.5	
Jul. 15	78	2500	3745	60	1508	39.5	
Jul. 15	79	10000	3745	130	1509	39.5	
Jul. 15	80	10000	3745	130	1509	39.5	end of test beam
Jul. 29	2	3h	5080	60	-	-	Noise run at LBNL

Table D.1: Test beam run information

E Digital appendix

The attached CD contains the source code of the developed software and other resources. Below is the content of the CD listed. Folders or compressed files are put in **bold**.

- SelfSeededTrigger_NiklausLehmann_Thesis.pdf: This document
- FPGA_BlocDiagram.pdf: RTL schematics of the correlator FPGA design
- WeeklyReports.pdf: Weekly progress reports created during the thesis
- **References**: Digital available references
- **Presentations**: Progress presentations of the work
- **ABC_Correlator.zip**: Latest version of the FPGA project with the correlation logic
- **ABC_Correlator_TestBeam.zip**: Correlation logic FPGA design used during the test beam
- **AnalysisMacros.zip**: ROOT macros to generate plots from the test beam data
- **AnalysisPlots.zip**: Plots generated from the test beam data
- **AnalysisRootFile.zip**: Assembled *.root* files for the analysis
- **CharacterizationMeasurements.zip**: Measurement results from the characterization measurements
- **LabVIEW_CorrelatorTester.zip**: VIs for the correlator test
- **LabVIEW_FCF_tester.zip**: First version of the FCF test application
- **NoiseRun.zip**: Measurement data from the noise run performed at LBNL
- **Pictures_Videos.zip**: Pictures and videos from the doublet assembly and the SLAC test beam
- **RandomCorrelations.zip**: Measurement data from the random generate correlation tests
- **TestBeam_Correlation_Data.zip**: Measurement data from the test beam