GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Fakultät für
Physik

Bachelor's Thesis

Implementation of the support of arbitrary pixel geometries in an existing testbeam analysis framework

Implementation der Unterstützung arbiträrer Pixelgeometrien in ein bestehendes Teststrahlanalyseprogramm

prepared by

Konstantin Lehmann

from Hamburg

at the II. Physikalisches Institut

Thesis period:	15th April until 22nd July 2013
First Referee:	PD. Dr. Jörn Große-Knetter
Second Referee:	Prof. Dr. Arnulf Quadt
Thesis Number:	II.Physik-UniGö-BSc-2013/04

Contents

1. Introduction **1**

2. The ATLAS Pixel Detector **3**

2.1. ATLAS . . . 3

2.2. Current Pixel Detector . . . 4

2.2.1. Planar Pixel Sensor . . . 5

2.2.2. Front End Chip . . . 6

2.3. ATLAS Upgrade: Insertable b-Layer . . . 6

2.3.1. Planar Pixel Sensor . . . 7

2.3.2. 3D Sensor . . . 8

2.3.3. Front End Chip . . . 9

3. Testbeam **11**

3.1. Nomenclature . . . 11

3.2. Setup and EUDET . . . 11

3.3. Track Reconstruction . . . 12

4. tbmon **15**

4.1. Structure . . . 16

4.1.1. Data Handling . . . 16

4.1.2. Preprocessing . . . 17

4.1.3. Analyses . . . 18

4.2. User Interface . . . 21

5. Arbitrary Pixel Geometries **23**

5.1. Pixel Geometry . . . 24

5.2. Arrangement of Pixels . . . 25

5.3. DUT Config . . . 26

5.4. ToT Calibration . . . 29

6. Results **31**

7. Outlook **35**

7.1. η-correction 35

7.2. Matching Condition 36

A. General Structure of tbmon **39**

Bibliography **41**

List of Figures **43**

Acknowledgements **45**

1. Introduction

In search of elementary particles and their fundamental interactions, recently a major discovery has been made. Physicists at the "Organisation européenne pour la recherche nucléaire" (CERN) discovered a new particle, which possibly is identical to the Higgs boson, which was predicted in 1964 [1]. It is the last missing particle of the Standard Model, which was developed during the last century.
The Standard Model describes fundamental particles, of which all matter consists. They can be divided into two groups, depending on their spin: fermions (half-integer spin) and bosons (integer spin). The fermions are matter particles, the bosons are interaction particles, which exchange three forces between particles: the electromagnetic, weak and strong interaction. Only gravity can not be included in this model. The Higgs mechanism explains a particle's mass by its interaction with the Higgs field.
One of the most important tasks of the "Large Hadron Collider" (LHC) at CERN is to measure the Higgs boson's properties. For this purpose, at LHC protons are accelerated in bunches up to energies of 4 TeV these days. The particles move with almost the speed of light along a circular beam pipe until they are forced to collide at certain spots, which is surrounded by detectors. The collision of two bunches is called "event".
At the LHC there are four different experiments. They are designed to reconstruct events by detecting the generated particles and their decay products. The detectors are optimized for different measurements. One of them, the "A Toroidal LHC Apparatus" (ATLAS), is designed to discover new physics particles. On that account it is amongst other tasks supposed to provide "efficient tracking at high luminosity" [2]. For this purpose, the Pixel Detector, which is a part of the ATLAS experiment, plays an essential role. Its setup and constituents will be explained in chapter 2. Due to the current upgrade of the Pixel Detector the corresponding setup changes will be a part of this chapter, too.
For upgrades of the Pixel Detector a lot of research on different pixel designs is done. One of the main aims is to improve radiation hardness. Another very recent issue is modifying the pixel shape, as there can be advantages in non standard design pixels. Before new modules are installed in the detector, they need to be tested. The characterization of the sensor is performed with a testbeam measurement, whose setup will be illustrated in

chapter 3.
When the data is taken at the testbeam, it needs to be evaluated. This task is performed by a program called “tbmon”, which provides dedicated analyses such as efficiency computation. As data processing requires exact knowledge of the module design, the pixel shape and arrangement must be well known to tbmon. Tbmon was programmed to only handle rectangular pixels. Since other pixel shapes are investigated in addition to that, a further development of the framework needs to be performed in order to support arbitrary pixel geometries. This is the main task of this bachelor thesis.
In chapter 4 the structure and functionality of tbmon will be explained. The main issues, when dealing with arbitrary pixel geometries, and their implementation will be illustrated in chapter 5. The results that were achieved will be presented in chapter 6. As an example two sensor setups are defined there, whose tbmon output demonstrates that the implementation operates as required. In order to analyze testbeam data, some tbmon methods need to be adjusted. In chapter 7 potential solutions for analyses techniques, that need to be reworked, are proposed.

2. The ATLAS Pixel Detector

2.1. ATLAS

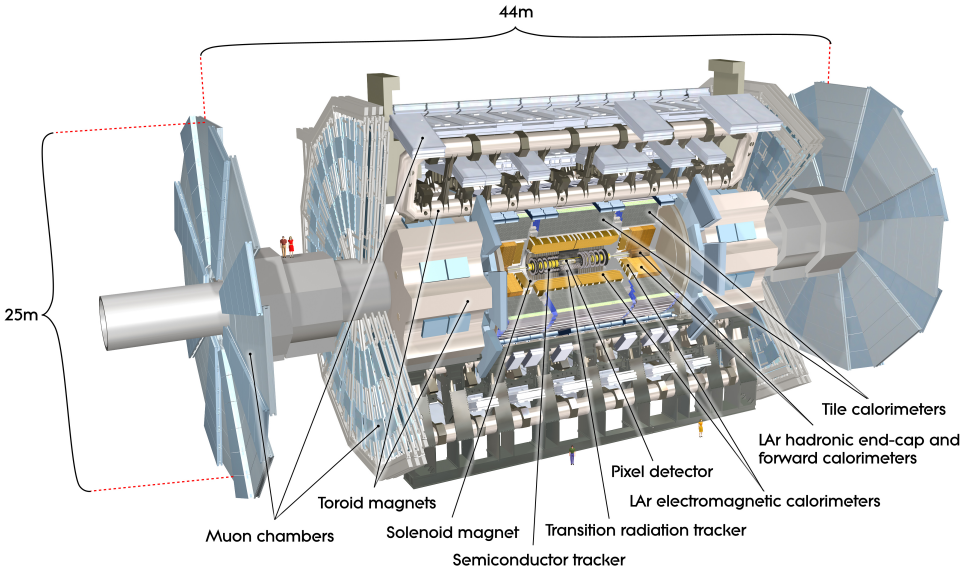

Figure 2.1.: Sketch of the ATLAS detector.

Due to the LHC design, the experiments need to fulfill special requirements. In order to achieve a high statistical significance of the measured quantities there are collisions of proton bunches with a frequency of 40 MHz. Thus all detector readout and data storage has to be managed in a 25 ns timing. Another issue is the high particle flux, which originates from the large number of interactions inside the detector. All used materials need to be radiation hard, particularly the electronics and sensors.
As depicted in figure 2.1, the ATLAS experiment [2] is a cylindrical setup of different detectors. Essentially there are the Inner Detector, the calorimeter and the muon chambers (from the inside to the outside). Most of them consist of disks arranged orthogonally to the beamline and barrel layers arranged parallel to the beamline (Z direction in the global ATLAS coordinate system). With this arrangement, the coverage of the full azimuthal

($R\phi$) and a great part of the polar angle (measured by pseudorapidity η[1]) is provided.
For reconstruction of an event and for further analyses, energy and momentum measurements are basic requirements. The calorimeter is designed to investigate an initial particle's energy by absorbing it. Especially for higher energies, as they occur at the LHC, the relative uncertainties of this measurement decrease with increasing energy. To investigate a charged particle's momentum, its quantity of deflection in a magnetic field is measured. On that account two magnets are located in the experiment: A solenoid magnet generates a field of 2 T, which is parallel to the beamline, inside the Inner Detector. The muon chambers are provided with a magnetic field by toroidal magnets.
For high precision tracking measurements the Inner Detector has been constructed. It consists of three different detector types. The outermost part is the Transition Radiation Tracker [3]. Its modules are filled with a Xe mixture, which is ionized by a passing particle. The released charges are detected and provide information about the particle trajectory. Every module is covered with a radiator foil, which emits photons, when a particle passes. These photons ionize the gas and cause an additional electric signal, that makes it possible to identify some light particles. The second detector type is a semiconductor tracker [4]. It is made out of silicon and its p type doped electrodes have a strip shape. The detector part closest to the beamline is the Pixel Detector, which is silicon based as well. In contrast to the semiconductor tracker it is made of pixels instead of microstrips. It is explained in more detail in the following section.

2.2. Current Pixel Detector

The Pixel Detector's [5] goal is to measure a particle track with excellent precision. As there are many simultaneous events at the LHC, the detector has to deal with a high number of particles. Particularly in the inner layers the particle flux per surface is very high. Thus the damage is expected to be up to $10^{15}\,\frac{\mathrm{n_{eq}}}{\mathrm{cm}^2}$ [6], where a neutron equivalent n_{eq} corresponds to the damage, that one neutron with kinetic energy of 1 MeV causes. Hence the consequences of irradiation had to be studied especially for the sensor.
Another issue is the material budget. As energy measurements are performed in the calorimeter, passing particles may only be minimally decelerated. The quantity X_0 indicates the amount of material, that causes a particle to decrease its energy to $\frac{1}{e}$ of the initial energy. For a barrel it is approximately 3.5% X_0 at $\eta = 0$ not including support structures [3].
The setup [3, 5] of the Pixel Detector is shown in figure 2.2. Since it is supposed to

[1] Pseudorapidity is defined as $\eta = -\ln\left(\tan\frac{\theta}{2}\right)$ with θ being the polar angle.

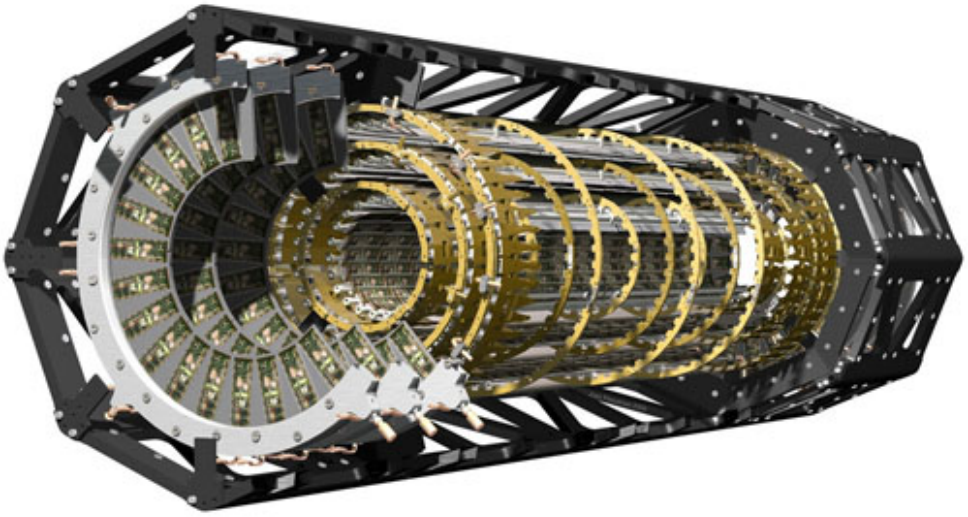

Figure 2.2.: Sketch of the ATLAS Pixel Detector.

measure three track points, it consists of three barrel layers, which cover the area of low pseudorapidity $|\eta| \lesssim 1.9$, the three disks enlarge the sphere to $|\eta| < 2.5$. As the single sensor modules are overlapping a little, every particle usually generates one signal hit per layer corresponding to three hits in total.

2.2.1. Planar Pixel Sensor

The sensor's functional principle takes advantage of a passing particle ionizing many silicon atoms. As a result for a minimum ionizing particle there are in the order of $20,000$ free electrons [3] expected in a non irradiated sensor, which drift to the readout electronics and cause a signal there.
The sensor consists of a 250 µm thick, slightly n-doped silicon wafer. The back side is p^+ doped, while on the front side of the sensor n^+ doped implants are located. In the n-bulk a depletion zone develops due to an equilibrium of the internal electric field and free charge carrier diffusion. When a passing particle creates free electrons, these electrons follow the electric field to one of the nearest n^+ implants. Every one of these n^+ implants represents a single pixel, since it detects a charge signal depending on its shape.
The irradiation of a sensor causes two kinds of damage. Ionizing effects mainly affect the sensor surface and electronics, non ionizing effects like collisions of heavy particles with the silicon lattice leave point and sometimes even cluster defects. Although the material composition does not change during this process, the lattice defects make the n-bulk behave like a p bulk. This process is called "type inversion", whose result is shown in figure 2.3. Only charge carriers that are released in the depletion zone are detected. In

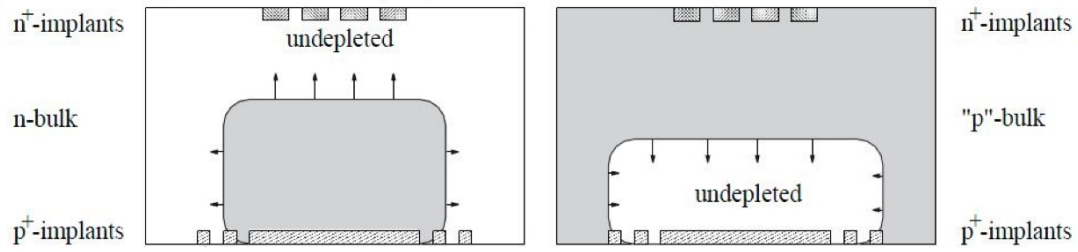

Figure 2.3.: Schematic view of type inversion in a silicon sensor. Left is before, right is after type inversion. After type inversion the pixels are surrounded by a depletion zone even without bias voltage.
Figure extracted and modified from [7].

order to enlarge the sensitive area, a reverse bias voltage is applied between the n^+ and p^+ implants. The more defects are induced, the higher bias voltage is needed to deplete all the sensor area.

The n^+ pixel implants are rectangular and they usually have a dimension of 400 µm × 50 µm. Each of these pixels is individually connected to an electronic readout circuit with a "bump bond". The circuit is part of a Front End (FE) chip.

2.2.2. Front End Chip

The FE chip [3] is a sophisticated readout chip, which receives an input current of the sensor and amplifies. Afterwards it digitizes the signal by measuring the time, during which the current is larger than an adjustable threshold. The resulting time over threshold (ToT) is buffered digitally and saved in units of 25 ns length. Afterwards the FE chip provides data storage until a trigger signal causes it to forward the data. The FE-I3, which is used in the Pixel Detector, has 18 columns and 160 rows. Due to its bump bond location, it is deliberately designed for a pixel size of 400 µm × 50 µm. Thus with every change in pixel geometry or pitch, the FE chip may need to be adjusted.

2.3. ATLAS Upgrade: Insertable b-Layer

Huge experiments at the LHC need upgrades [8] every now and then. For the detectors, the focus lies on repairing and replacing some parts, as they suffer from irradiation. During the next years the LHC wants to evolve into the High-Luminosity LHC. Thus in long terms the experiments need to adapt to these conditions. For example at ATLAS the entire Inner Detector is planned to be replaced in about ten years. At present already much research effort is in progress for the High-Luminosity LHC.

Currently (mid 2013) the LHC is shut down for a period of about two years. During this time parts of the infrastructure will be tested and reworked, so that in the beginning of 2015 the center of mass energy can be increased to 14 TeV. The experiments are being modified as well. In ATLAS a main change of the setup is the installation of the Insertable b-Layer (IBL) [8, 9]. This 4th pixel layer will be inserted between the beam pipe and the current inner pixel layer in order to improve the tracking resolution and to partially compensate radiation damage.
In search for new physics some models depend in large part on events including b-quarks. Therefore b-tagging is a powerful tool. Its uncertainties can be decreased by installing the inner layer as closely to the interaction point as possible. The beam pipe will be shrunk by approximately 4 mm to an outer radius of 24.4 mm, so that IBL is placed at an average radius of 34 mm. As a consequence, this layer will suffer from higher irradiation and process a higher data rate than the current b-layer.
In contrast to the current Pixel Detector, IBL does not comprise a disk layer. It includes several new technologies, which have been developed and tested during the last years. These technologies enable to reduce the material budget, so that IBL has only an expected radiation length of 1.5% X_0. Much effort has been expended to improve the sensor modules. At an advanced stage of the IBL project, after an extensive testing phase, two possible sensor technologies remained: Planar sensors and 3D sensors, which are both silicon based. Finally both sensor types are installed at IBL, 75% of planar sensors and 25% of 3D sensors at high $|\eta|$, where they are expected to provide a higher efficiency. In the following chapters the 3D sensors and the modifications of planar sensors with respect to the current Pixel Detector will be described.

2.3.1. Planar Pixel Sensor

In the current ATLAS experiment 250 µm thick n-in-n sensors are installed. IBL studies [10] have investigated many aspects, for example radiation hardness was examined for an n-in-p pixel design in comparison to the established n-in-n design. A major benefit of the p-bulk material is a simplified sensor production, as the devices only need to be processed on one side. In addition, type inversion does not occur after high irradiation. However, another design was favoured, although n-in-p sensors with a thickness of 150 µm were developed for IBL. Due to lower manufacturing costs, this technique might still be used for further upgrade plans.
On the back side of planar sensors guard rings are placed near the edges. They achieve the required potential drop between the p^+ doped back and the sensor edges. As this region is a non active part of the sensor, some research concentrated on reducing the inactive area

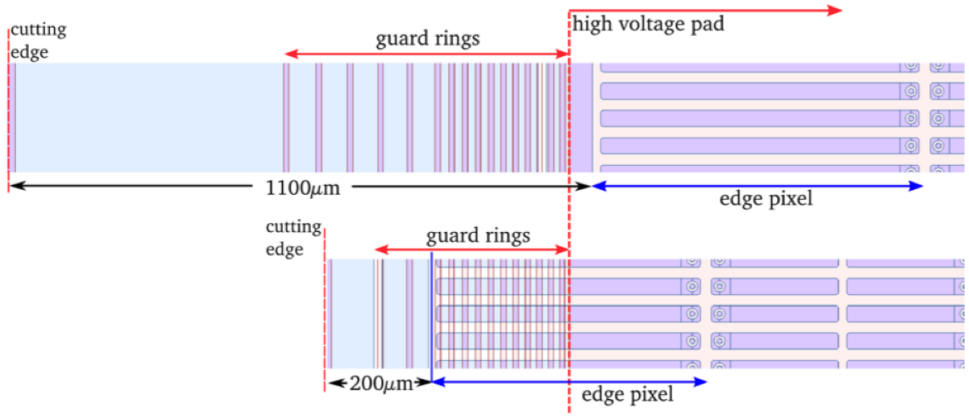

Figure 2.4.: The upper sketch depicts the arrangement of guard rings and edge pixels in the current Pixel Detector. In the slim edge design the inactive region was significantly decreased. The guard rings are located at the sensor back, the pixels at the front.

at the pixel edge by overlapping elongated pixels and guard rings. With respect to the current sensor design both the arrangement and amount of guard rings were investigated. Finally, this slim edge design with an inactive region of approximately 200 µm was chosen for IBL. The number of guard rings is reduced to 13 and they are located underneath elongated pixels. A comparison of the current sensor edge region and the more advanced design is shown in figure 2.4. The slim edge design is n-in-n based and the sensor has a thickness of 200 µm.

In general the pixel pitches are 250 µm × 50 µm and on the edges the pixels are enlarged to 500 µm × 50 µm. While in the current detector one sensor is covered by 16 FE chips, in IBL two FE chips cover one planar sensor. They are called double chip module. As between both FE chips a gap occurs, in this region two columns with a larger pixel pitch (450 µm × 50 µm) are located [11].

2.3.2. 3D Sensor

3D sensors [12, 13] have the same functional principle as planar sensors. The main difference compared to planar sensors is that the electrodes are located vertically inside the wafer rather than horizontally on the surface. They are etched into the bulk and doped by diffusion, which results in a setup that is depicted in figure 2.5. The column on the front side is n^+ doped and connected to a FE chip via bump bonds. The back side columns are p^+ doped.

This setup allows to decrease the distance of two electrodes independently from the bulk thickness. This thickness influences the number of released electron-hole pairs before ir-

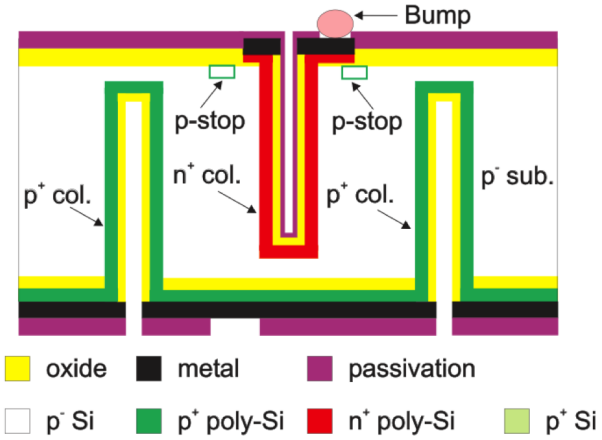

Figure 2.5.: The pixel electrodes are inside the silicon bulk. A crucial advantage is the lowered distance of two electrodes [13].

radiation. Reducing the electrode pitch implicates various advantages, as a lower bias voltage. Even after high irradiation 3D sensors full depletion can be accomplished with less than 200 V. Furthermore the charge collection time decreases, as the electrons have a shorter way to travel. In addition, this causes that the charge trapping probability is reduced. In this way the signal over noise ratio does not decrease after irradiation to the same extent.

2.3.3. Front End Chip

The higher particle flux and the changes in pixel dimension made IBL also require a different FE chip, which is the reason for a new FE chip generation development. The utilized FE-I4 [13, 15] is geared to pixel sizes of 250 µm × 50 µm. With respect to the previous chip, the number of cells increased to 26,880 (80 columns × 336 rows). Consequently the FE-I4 covers an approximately 5 times larger area.
As IBL is supposed to be assembled closer to the beamline than the current innermost detector layer, the FE-I4 has to deal with higher irradiation. It is particularly designed to withstand a neutron equivalent of $5 \cdot 10^{15} \frac{\mathrm{n_{eq}}}{\mathrm{cm}^2}$, which has been a major challenge. Another consequence of the lower radius is a higher hit rate. Hence the data is not read out column by column and saved centrally any more, but is stored in a "4-pixel digital region" instead. This region covers four pixels, saves their ToT information and forwards it, when a trigger signal arrives, or rejects them instead. This way more memory space is provided and in addition the four pixels are already able to interact with each other. When one of them detects a large hit, every pixel stores the ToT and all the data in the 4-pixel digital region are matched to the event. By arranging four pixels in a joint region the data rate is

lowered, as the data is not pushed to the periphery any more, when no trigger arrives. The FE-I4 design necessarily specifies the pixel pitch. Of course, both the planar sensors and the 3D sensors have been developed to be compatible with the FE chip: 3D sensors have the same size as a FE-I4 (single chip module), while one planar sensor fits two FE chips (double chip module).

3. Testbeam

In chapter 2 various requirements for Pixel Detector modules are described. When a new prototype is manufactured, its properties need to be investigated with high diligence. In particular the performance after irradiation is studied by comparing sensors, that were irradiated before, to different levels. At a testbeam, the Device Under Test (DUT) is examined using a particle beam, usually high energy pions (120 GeV) at CERN or electrons respectively positrons (1-6 GeV) at DESY[1] are utilized. Their track is measured using reference detector planes (telescope). The EUDET[2] testbeam setup used for the characterization and a short overview of the track reconstruction tools are presented in this chapter.

3.1. Nomenclature

Some terms used in the context of EUTelescope and tbmon are defined here.

Run	A series of events with the same assembly.
Event	All data taken during one trigger.
Hit	A signal in a pixel during an event.
Track	The expected position of a particle hit on the DUT.
Cluster	A group of neighbouring hits, which may be caused by the same particle.
Match	A track is called "matched", if it can be identified with a cluster. This cluster is named "matched" as well.

3.2. Setup and EUDET

The EUDET telescope [17] was developed to improve detector research. It is a series of six in succession aligned, equal detector planes, whereas the DUTs are assembled between the first three and the second three layers, as shown in figure 3.1. Thus a particle in the

[1] Deutsches Elektronen-Synchrotron
[2] DETector-project supported by the European Union

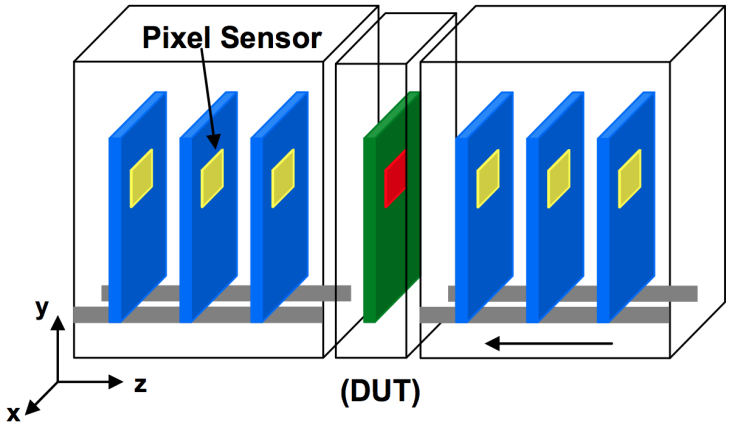

Figure 3.1.: Sketch of a testbeam setup using the EUDET telescope [16]. It is also possible to test a couple of DUTs at once.

testbeam first passes three telescope planes, afterwards the DUTs and finally three more telescope planes. An event is triggered by scintillators located upstream and downstream of the telescope.
The telescope planes consist of silicon based Mimosa26 [16] sensors covering an area of 21.2 mm $\times$ 10.6 mm. They differ from the ATLAS Pixel Detector sensors in various issues, as they are designed to work in a different environment as LHC experiments do. Their readout time of $\sim$ 100 µs is very long compared to 25 ns, but with a pixel pitch of 18.4 µm they have a significantly higher resolution. In addition, their thickness can be slimmed down to 50 µm, which suppresses multiple scattering and improves DUT characterizations. Furthermore the Mimosa sensors only have a binary output, which requires different analyses.

3.3. Track Reconstruction

After data has been taken the offline analysis tool "EUTelescope" [18] combines the pieces of information.

- At first single pixels, which recorded an occupancy of more than 0.1% (default value, which can be changed), are masked, as they are assumed to be noisy.
- Afterwards the hits in neighbouring pixels are arranged in groups, as they are probably caused by the same particle. To separate hits, who originate from two close-by particles, a "digital fixed frame algorithm" is applied. In a group it looks for the "seed pixel" and merges all hit pixels, which are inside a fixed frame around the

seed, into the cluster. The seed pixel is defined to be the pixel with the most non diagonal neighbours. If this condition is ambiguous, the diagonal neighbours are considered. The same algorithm is applied for all not yet clustered pixels of the group.

- The telescope planes' alignment needs to be known with more precision than the pixel resolution. It is determined at the same time as the tracks through the layers are built using a χ^2 minimizing algorithm individually in x and y direction. Therefore a straight line is fitted through all combinations of hits in different planes. Cuts concerning residuals reject most of the background, before a least squares fit adjusts both track and alignment constants. There are three constants per plane taken into account: shifts in x and y being orthogonal to the beam axis and rotation around the beam axis.

- In a final step every tracks' intersection point with the DUT is extrapolated. Due to multiple scattering the uncertainties depend on the beam energy. They are in the order of 2 µm.

The analysis framework EUTelescope produces several data files, one of them is a root file named "tbtrack" with a run number appended. It is deliberately created for further processing in tbmon and contains all raw data of the DUTs as well as the reconstructed hit position based on the telescope data.

4. tbmon

Tbmon is a powerful C++ based offline framework for processing testbeam data, which have already been preprocessed with EUTelescope. It is supposed to provide automatized analyses by producing sensor specific pixel maps and plots, which show physical properties. These results are used for examining new sensor techniques, as they indicate the relevance of new developments. When comparing different sensor designs, the tbmon analyses are crucial, as they apply the same standard and make results comparable. This for example happened, when pros and cons were balanced concerning planar and 3D sensors for the IBL upgrade. The tbmon outcome is also appreciable for final characterization and performance measurements, if sensors are installed in the ATLAS detector.
In order to produce comparable results when characterizing sensor prototypes, the ATLAS pixel groups decided to use tbmon as their standard analysis tool. As the requirements change over time, tbmon often needs to be upgraded. Currently in sensor development some designs are changed concerning pixel geometry. Thus a major issue is the implementation of non rectangular pixels and their configuration. As furthermore the development of other pixel geometries is in progress, there is the necessity to implement arbitrary pixel geometries. Also the pixel arrangement is crucial. The double chip modules for instance can only partially be recognized by tbmon. As they are installed at IBL, their complete characterization is very important. Thus tbmon needs to be adapted in order to evaluate all existing data.
The implementation of arbitrary pixel geometries and pixel arrangement are the main tasks, which were performed during this thesis. The progress and results on this subject will be explained in chapter 5.
As tbmon provides a powerful analysis framework and is expandable, also other groups are interested in utilizing it, especially when arbitrary pixel geometries are available. As tbmon is quite adjustable as well, many preferences and analysis settings can be changed. Unfortunately these settings are not made centrally, but they are spread over the source code. Not only for other groups, but also for the ATLAS pixel community an easier handling of preferences is desirable. For that purpose lately many settings were outsourced in config files, which are explained in chapter 4.2. The arrangement and contents of the

config files was developed during this bachelor thesis, too.
In the following section the underlying structure of tbmon will be explained. It was only changed marginally and basically remained the same.

4.1. Structure

Tbmon is split into two parts: First some eventbuilders prepare the data. As the DUTs' raw data is used as input, some preprocessing steps have to be performed. Afterwards tbmon continues by applying different analyses, which produce the main output. The most important eventbuilders and analyses will be illuminated in this chapter, after a basic overview of the data handling in tbmon is given.
The inner structure of eventbuilders and analyses are briefly explained in appendix A. There is also an overview of the method calls.

4.1.1. Data Handling

In tbmon the testbeam data are stored in an object of the class "Event". The most important objects of each event are listed below:

- Track: Some properties of the track, that come from EUTelescope, e. g. χ^2.
- TrackX, TrackY: Position of the track in µm in the DUT coordinate system.
- Hits: Vector of all the single hit properties in the DUT, e. g. row, column, ToT.
- Clusters: Vector of all the clusters, each containing an arbitrary amount of hits.
- Many flags (values are called "kGood", "kBad", "kUnknown") summarizing the important properties of the event. They often indicate a cut on a track property.
  - fTrack: Indicates whether the track passed all cuts, e.g. a minimum number of reference DUTs, who matched the same track are required.
  - fTrackCentralRegion: Indicates whether the track does (kBad) or does not (kGood) match an edge pixel.
  - fTrackMaskedRegion: Indicates whether the track matches a masked pixel.
  - fTrackRegion: It has the value kBad, if either fTrackCentralRegion and fTrackMaskedRegion are kBad or the track may be identified with a cluster containing a masked pixel.

4.1.2. Preprocessing

Before the DUTs can be analyzed, during the preprocessing some jobs need to be performed. They are listed in the order they are performed in:

Maskreader

Before a DUT is exposed to the testbeam, potentially the related sensor was already examined in a laboratory. For example measuring the signal of irradiation with an undirected β^- emitter often shows dead pixels. In order to mask them for data taking an external mask file can be applied to tbmon in Maskreader.

Hotpixelfinder

In this eventbuilder, unmasked noisy and dead pixels are supposed to be found based on the testbeam data. Since it can be useful to distinguish between different reasons for masking, tbmon stores them in the integer variable `type`. It is saved in a binary way, as one pixel can be masked for several reasons. In this case different types can be added without information loss. The allocation of reason to number is shown here.

`type`	Reason for masking
0	No mask
1	Noisy
2	Neighbouring pixel is noisy
4	Dead
8	Whole column or row is masked in DUT config
16	Masked by Maskreader

For every DUT all the hits are inserted into a hit map, which counts the number of hits for each pixel during one run. If a pixel has registered no hit, it is masked as dead. If a hit on a certain pixel is beyond the LV1 cut in more than $5 \cdot 10^{-4}$ of all cases, it is masked noisy. In addition, all its neighbouring pixels including the diagonal ones are masked, as sometimes fake hits occur due to electromagnetic induction.

Checkcentralregion

This eventbuilder sets the flags fTrackCentralRegion, fTrackMaskedRegion and fTrackRegion. The latter one might be modified in Clustermasker.

Clusterfinder

The Clusterfinder combines all individual hits into clusters. Its functionality is different from the clustering algorithm in EUTelescope, since it merges all neighbouring (including diagonal) hit pixels into a cluster. The pixels' mask is ignored at this point.

Clustermasker

After hit pixels have been grouped in clusters, these are checked for masked pixels. If a cluster contains a masked pixel, it is rejected. If furthermore the track is expected to be inside the cluster, the event flag fTrackRegion is set to kBad.

Eubuildtrack

If more than one DUT is examined in the testbeam setup, the DUTs can be aligned with respect to each other. From a track on the considered DUT a cone is applied towards the other DUTs. If their track is not located inside the circular base of the cone, the events are rejected. At this stage a more strict cut is applied than in EUTelescope due to the higher precision in Checkalign.

Checkalign

When the alignment is performed in EUTelescope, shifts of up to 5 mm occur. When running Checkalign in tbmon only the DUTs are analyzed and adjusted. More precise results are achieved at this stage, due to different clustering and alignment algorithms, a more strict Hotpixelfinder and run by run analyses. Checkalign observes one hit cluster residuals in x and y direction as a function of both x and y for every DUT. By fitting a polynomial of degree one to the trend of this curves the DUTs' rotation and shift can be calculated.

Translator

Translator reads the shifts, that were calculated by Checkalign, and applies them to every events' track position by simply adding them. Additionally the user can add manual shifts.

Getetacorr

The eventbuilder's name is an abbreviation for "get η-correction". This calculation is performed for improving the reconstruction of a hit position out of the cluster information. Although the former η-correction generally does not yield best results for arbitrary pixel geometries, it is still implemented. A brief description and an idea for another technique is presented in chapter 7.1.

4.1.3. Analyses

Like in the previous section here different tbmon analyses are briefly explained. They do not have to be processed in a certain order.

Beamprofile

This analysis produces various histograms concerning the profile of the beam based on the telescope's data.

Clusterchecker

Here some cluster properties are investigated and saved in terms of histograms, for example the cluster size of matched clusters.

Correlations

Correlations analyzes the correlation between hits on different DUTs by plotting track position against hit position. As the track position equals the hit position on a reference plane in good approximation, the resulting plot depicts the correlation. It allows an estimation of the alignment quality.

Efficiency

Calculation of the DUT efficiency is one of the main tasks among the analyses. Considered are N events marked by the index i, each implying one track. In every event either a hit, that matches the track, or no matching hit is observable. For every single event efficiency e_i the following equation is valid:

$$e_i = e_i^2 = \begin{cases} 0 & \text{no matching hit observed} \\ 1 & \text{matching hit observed} \end{cases} \tag{4.1}$$

The matched efficiency e_{match} is calculated by the average value

$$e = < e_i > = \frac{1}{N} \sum_{i=1}^{N} e_i. \tag{4.2}$$

The uncertainties Δe are evaluated by calculating the standard deviation σ using equations (4.1) and (4.2):

$$\begin{aligned} \sigma^2 &= < e_i^2 > \; - \; < e_i >^2 \\ &= \frac{1}{N} \sum_{i=0}^{N} e_i^2 - \quad e^2 \\ &= \quad e \quad - \quad e^2 \\ \Delta e = &\quad \frac{\sigma}{\sqrt{N}} \quad = \quad \frac{\sqrt{e - e^2}}{\sqrt{N}} \end{aligned} \tag{4.3}$$

Equations (4.2) and (4.3) also hold for the "any hit efficiency" e_{any} stating how often during one event any (not necessarily matched) hit was detected.
In tbmon only three indices need to be enumerated: The number of tracks seen by the telescope `ntracks` $= N$, which corresponds to the number of events, the number of hits matching a track `nhits` $= \sum_i e_{\text{match},\, i}$ and the number of events having any hit `anyhit` $= \sum_i e_{\text{any},\, i}$ are counted. Only events, which passed all cuts (flag fTrack) are

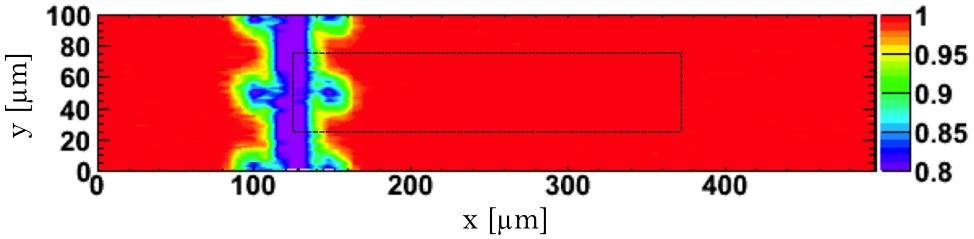

Figure 4.1.: In-pixel efficiency plot of a neutron irradiated, planar pixel sensor with a bias voltage of 1000 V and a pixel pitch of 250 µm × 50 µm. The dashed rectangle illustrates one pixel, one half of the neighbouring pixel is plotted as well. The efficiency information of the whole DUT is folded into this plot. A decreased efficiency can be recognized at the bias electrode pixel edge. Figure extracted and modified from [19].

taken into account.

Tbmon also prints out the track, hit and any hit distribution of the DUTs in histograms. An inefficiency map is created as well. Its entries are calculated by $1 - \frac{\#\text{hits}}{\#\text{tracks}}$. Furthermore the variation of efficiency inside a pixel is interesting. It can be resolved, as the telescope's precision is about 2 µm. For the same pixel geometries, all efficiency measurements are folded into one pixel to increase the statistics. These plots are instructive, as they depict the pixel setup. As shown in figure 4.1, for irradiated planar pixels, for example, clearly the bias electrode's position is recognized, as the efficiency decreases there.

Lvl1cut

This analysis creates two one dimensional histograms showing the Lvl1 distribution of matched and of any hits. A Lvl1 value of 6 is expected due to the triggering. If the histograms differ to a large extend, cuts can be applied to reject noise.

Maxcellresiduals

Here, mainly properties of the cell with the highest ToT per DUT and event are analyzed, for example the residuals both cut and uncut. But also other values like the ToT of the largest cluster and the χ^2 are filled into histograms.

Qefficiency

Qefficiency analyzes the charge distribution, as an inhomogeneous spread indicates critical parts of the sensor, in which less charge is collected. All over the DUT histograms and in-pixel histograms are created. In this way worrying spots can be observed independently of the efficiency calculation.

Qshare2D

In testbeam measurements at low angles a cluster usually contains only one pixel. Thus the region in which the charge is split up between two pixels is interesting.

Qshare2D analyzes this charge sharing by producing in-pixel plots, which include half of the neighbouring pixel in x and y. They show the probability and amount of charge sharing dependent on the position.

Sumtot

In this analysis various ToT distributions are computed and saved in terms of histograms. The ToT dependency on cluster position on the DUT is plotted as well. If a ToT calibration[1] is available, also the charge is calculated.

4.2. User Interface

When merging the preferences into config files, the handling by a user needs to be considered. Particularly his arrangement and storage of data files need to be in agreement with the config files' construction. It is assumed that a user wants to

- have all settings saved together with the finished analysis results.
- reproduce the analysis without modifying anything.
- perform different analysis techniques on the same data set without overwriting them.
- be able to save all DUT config files together.
- be able to save all raw data from different measuring periods together.

When executing tbmon the user needs to specify an output folder, where all the created data is supposed to be saved. If this folder does not exist yet, tbmon creates it and sets up all required files including default config files. Afterwards it terminates, so that the user is given the possibility to modify the config files or to replace them by own ones. If the output folder already exists when launching tbmon, preprocessing and analyses are executed.

In figure 4.2 the arrangement of tbmon files is shown. An output folder always includes both the tbmon output, which is stored in the folders "preprocessing" and "plots" and the config information. The raw data can be saved at any place, since it is referenced. Three different config files are available, whose encoding is easy to understand, as it is very similar to C++ code and a prototype already exists, which only needs to be modified. The main config file contains general settings, eventbuilder settings, plot settings and location of testbeam data. The information about the DUTs is split up: All cuts and tbmon values are stored in the main config, the pure arrangement and shape of pixels is

[1] For more information see chapter 5.4

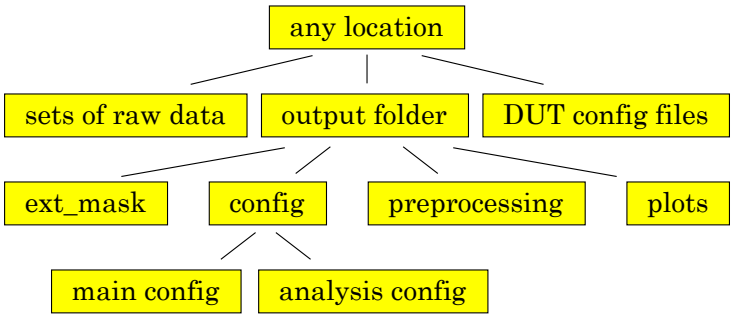

Figure 4.2.: Storage of files for tbmon analyses. The main advantage is that all configuration is saved together with the results. In this way the analyses are reproducible at a later time.

specified in a DUT config file, which is referenced by main config. Thus the user creates only one DUT config file for each sensor design and afterwards refers to it in multiple evaluations. An example for a DUT config file is given in chapter 5.3.
In analysis config all cuts, that are applied during analyses, are defined. Hence every analysis can hold an arbitrary amount of strings, which are converted into different variable types and used during runtime.

5. Arbitrary Pixel Geometries

When analyzing a DUT in tbmon the arrangement and shape of the pixels need to be defined accurately. So far the DUT description has been quite simple, as depicted in figure 5.1. Up to now it was possible to define one rectangular standard pixel as well as one edge pixel by hard coding their pitch in x and y. By entering the number of rows and columns the DUT was defined completely.

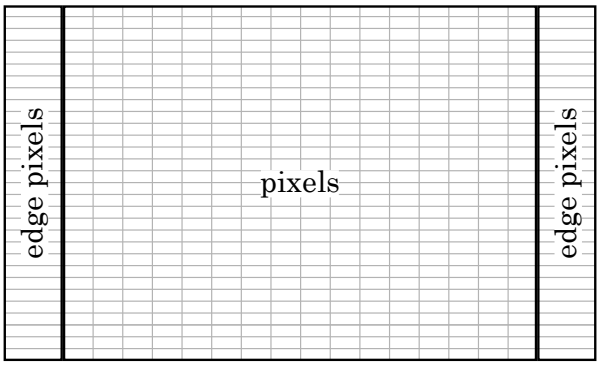

Figure 5.1.: Former description of DUTs: In the middle there is a large amount of standard pixels. The extension in x and y depends on the number of columns and rows of the FE chip. At the left and right edges there is one column of edge pixels having a different pitch.[1]

A FE chip is designed for a particular pixel geometry, as the pitches of the bump bonds need to be consistent with the pixel pitches. When other pixel geometries are investigated, the aim is to test them with an established FE chip, since development of a new chip would be too much effort. Although that might result in uncommon shapes, subsequently rather the examined geometries than the FE chip are slightly changed. An "L shaped" pixel is a good example: Originally longer and at the same time narrower rectangular pixels were supposed to be tested. Adjusting to the FE-I4 design is the reason for designing L shaped pixels instead, as illustrated in figure 5.2.

Besides the double chip module there is another DUT design, that has been examined in a testbeam already, but can not be evaluated. It contains rectangular pixels, where every

[1] Figure not to scale.

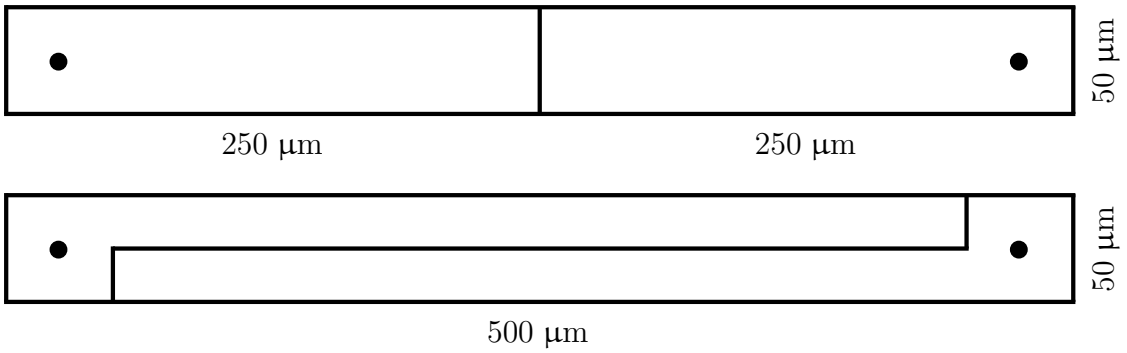

Figure 5.2.: Two pixel geometries in comparison. The FE chip was initially designed for the upper one. To study narrower pitches and still contact the bump bonds indicated by the solid black circles, an L shaped pixel is used (lower sketch).

second row is shifted in x by half a pixel pitch. For this kind of DUT the pixel geometry is less important, but the pixel arrangement does matter. These two examples lead to two major issues.

1. How to define a pixel geometry in tbmon?
2. How to arrange the pixel positions?

How they were solved in tbmon is illustrated in the following sections.

5.1. Pixel Geometry

Before the first question can be answered, the kind of geometries, that are supposed to be defined, needs to be ascertained. In the coming years, most likely only pixels based on rectangular shapes will be designed. That makes it easier to define a pixel geometry in tbmon, as it can be based on an arbitrary number of rectangles. Their size and arrangement define the pixel shape. For L shaped pixels this is shown in figure 5.3.

In Tbmon all the relevant information about a DUT is handled in an identically named class. As it is supposed to hold a list of pixel geometries, it makes sense to define another class `PixelGeometry`. All necessary information concerning a pixel geometry is stored there. As rectangles are a part of a `PixelGeometry`, it is again useful to write another class called `Rectangle` providing all the required information. Once again an arbitrary number of them can be memorized by `PixelGeometry`.

The rectangle size and arrangement needs to be defined. Thus every `Rectangle` comprises the position of its lower left and upper right corner in a pixel geometry specific coordinate

Figure 5.3.: A pixel, based on rectangular shapes, is spanned by smaller, red colored rectangles. There can be different possibilities, two of them are shown in this figure for the L shape.

system. So, simply by composing the single rectangles, a pixel geometry is defined. A `PixelGeometry` contains `Rectangles`, a variable indicating if the pixel geometry is used on the edges and another one denoting mirroring properties, which will be explained in chapter 5.3.
There is also some redundant information stored. It is calculated by extraction of other properties and saved, as it might be accessed quite often during the program execution. That way some time for constantly recurring computation is saved. In a `Rectangle` especially its area, in a `PixelGeometry` its area, its geometric center (center of mass) and its maximum expansion in x and y are stored.

5.2. Arrangement of Pixels

The previous section explains, how the pixel shape is defined. Now the single pixels need to be arranged on the DUT. In tbmon it is necessary to adjust each pixel to one cell of the FE chip (every combination of column and row), as they obligatorily need to be associated with each other due to the readout. Thus an array, that has the same number of columns and rows as the FE chip, is created. Every entry is associated with one pixel shape by a pointer to `PixelGeometry`. This setup is shown in figure 5.4. In this way tbmon knows, which FE cell is connected to which pixel shape. To define the exact position of every pixel, the array also contains information about the lower left corner of the pixel geometry in a DUT specific coordinate system. In summary, the DUT has an array and each of its cells is logically connected to one pixel (by a pointer to `PixelGeometry`) and contains its lower left coordinate.
The described method was implemented in tbmon, but nevertheless also another one was considered. It would not contain an array, but would have the information stored by their periodical properties. Every `PixelGeometry` would then contain the indices of the FE cell, where it first appears. In addition, it would save the information after how many

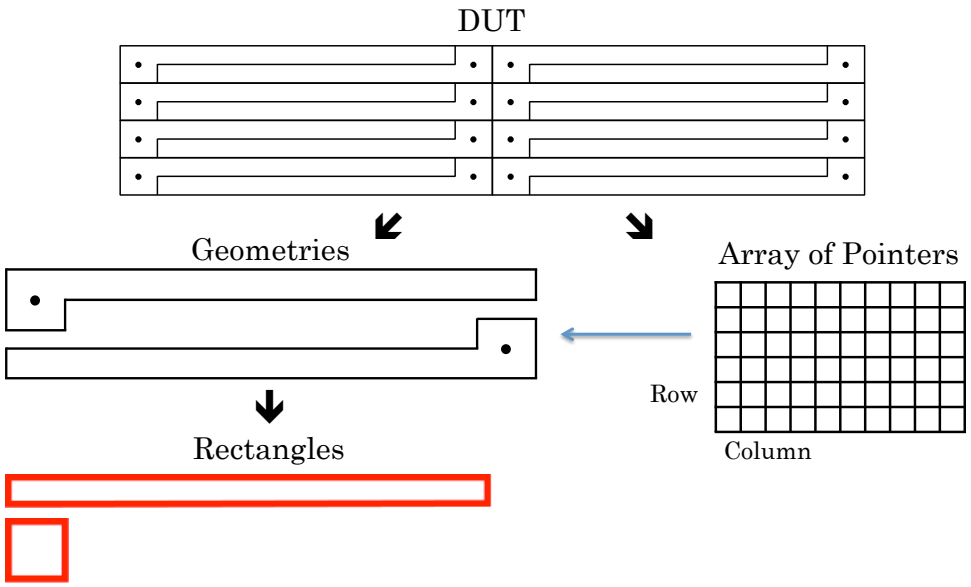

Figure 5.4.: Inner structure of a `DUT`. Every `DUT` holds a list of different pixel geometries, which again hold a list of underlying rectangles. Besides that there is an array with the same number of columns and rows as the FE chip. It represents the DUT area and by pointing to geometries states the arrangement of pixels on the DUT.

columns and rows it occurs again. So a DUT filled with L pixels for instance would be defined by: "Pixel geometry 1 first occurs in column 0 and row 0, it is repeated in every row and in every second column. Pixel geometry 2 first occurs in column 1 and row 0, it is repeated in every row and in every second column."
The latter technique's advantage is smaller memory usage, while on the other hand computation time grows, when for each track coordinate tbmon needs to find out the pixel geometry, which covers that coordinate. A simple test framework showed that for two pixel geometries in total the array technique has a 30% to 40% faster access time. This effect is expected to even grow for more different geometries used on one DUT. Furthermore in the array some more data could be saved, for example a pixel's mask value and its position on the DUT. Due to this comparison the array method was chosen.

5.3. DUT Config

The previous sections explained, how the DUT setup is determined. Since no hard coded geometry properties are desirable, the user can define the DUT design by using the DUT config file.

```
mainDUTConfig:
{
  DUTsizeX = 20000.0;
  DUTsizeY = 16800.0;
  DUTthickness = 200.0;
  cols =  80;
  rows = 336;
};
pixelConfig:
(
  {
    firstPixelRow = 0;
    firstPixelCol = 1;
    firstPixelX = 50.0;
    firstPixelY =  0.0;
    periodPixelCol = 2;
    periodPixelRow = 1;
    periodPixelX = 500.0;
    periodPixelY =  50.0;
    countPixelCol =  40;
    countPixelRow = 336;
    edge = false;
    periodMirroringCol = 0;
    rectangleConfig:
    (
      {
        lowerLeftX = 0.0;
        lowerLeftY = 0.0;
        upperRightX = 400.0;
        upperRightY =  25.0;
      },
      {
        lowerLeftX = 400.0;
        lowerLeftY = 0.0;
        upperRightX = 450.0;
        upperRightY =  50.0;
      }
    );
  },
  {
    # second L pixel
  }
);
```

Listing 5.1: Excerpt of the DUT config file defining and arranging L shaped pixels on a FE-I4.

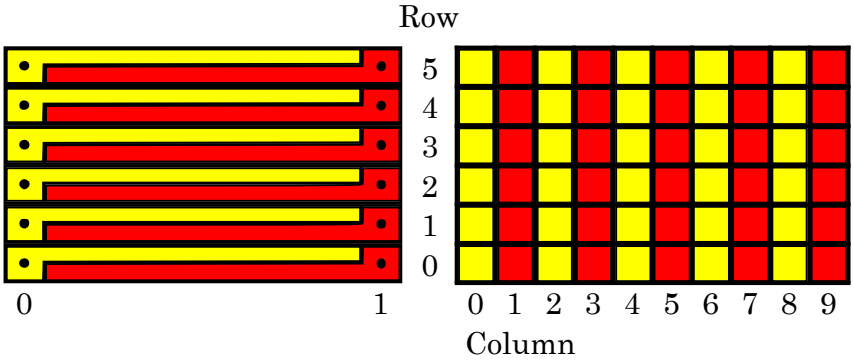

Figure 5.5.: The left side shows the actual pixel shape, the right side represents the array of pointers. The colors illustrate, to which pixel geometry the array entries point.

A `DUT` is constructed in a way that pixel geometries do not necessarily have to be arranged with periodical properties. However, a real DUT will have some kind of periodicity, so its config file is designed in such a way. To explain its structure and the necessary values, a DUT, which is made up out of two different L shaped pixels, is defined as an example in listing 5.1.

The DUT config file is split into two parts, while the first part contains basic information about the sensor, in the second part pixel geometries together with their arrangement are defined. All variables, whose last letter is "`X`" or "`Y`", are of the type `double` and require statements in µm, the other variables are integers.

In the first part the sensor dimensions are entered, because they are needed for some consistency checks. The thickness is not used yet, but will be needed, when cluster center calculation changes in the context of η-correction. Afterwards the number of columns and rows is inserted.

In the second part pixel geometries are defined. In this config file the configuration of the lower right L shaped pixel (red marked in figure 5.5) is described. To know, which FE cells match this pixel geometry, the user states, where this geometry occurs first. In figure 5.5 that corresponds to the lowest and most left appearance of a red pixel (lines 12 and 13 in the config file). Afterwards the lower left coordinates of this pixel geometry need to be entered (lines 14 and 15).

Thereafter the user needs to state after how many columns and rows this pixel shape occurs again. For the red pixel geometry that means: every row and every second column (lines 16 and 17). Also the distance of this periodicity needs to be entered: In x this pixel geometry occurs again after 500 µm, in y after 50 µm (lines 18 and 19). Thereafter the number of iterations is entered: As the FE chip has 80 columns and every second column is associated with this pixel geometry, that corresponds to 40 columns in total

(line 20). Since it occurs in every row, the number of rows is entered in line 21. The variable `edge` states, whether this pixel geometry is used on the edges. In this case some separate analyses, like edge efficiency, can be performed for these pixels. This property is neglected at this stage, as the edge pixels would need to be defined separately and would exceed this example.
As figure 5.5 depicts, the bump bonds between sensor and FE chip are on different sides. Even-numbered columns have it on the lefthand side, for odd-numbered columns it is placed on the righthand side. When one pixel geometry spans both kinds of columns (for instance the rectangular pixels in the upper figure 5.2), they are assembled in a mirror-inverted way. As this has to be considered for in-pixel plots, the property `periodMirroringColX` takes that into account, by analyzing the periodicity of mirrored columns, which the user needs to enter. For L pixels this is not of importance, since each geometry is only associated with either even- or odd-numbered columns.
Now the underlying rectangles are defined, where the extensions are interpreted in the pixel's coordinate system. All rectangles are placed in the first quadrant, as close to the x- and y-axis as possible. In this example first the long slim rectangle (lines 27-30) and afterwards the square, which includes the bump bond, is created (lines 33-36). In the config file the second L shaped pixel is omitted, as indicated by the comment (line 41). It would be defined in the same way as the first one.
The DUT config file only includes sensor specific information. Hence it only needs to be created once for each sensor layout and can be applied by every user. This is a major advantage of the conifg arrangement.

5.4. ToT Calibration

In testbeam analyses it is desired to convert the measured ToT into precise charge values. The charge as a function of ToT can be approximated by a polynomial. For all pixels the polynomial function is the same, but its constants differ. They are determined separately

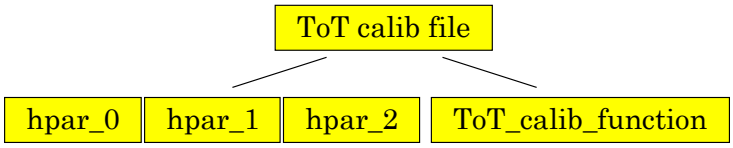

Figure 5.6.: Contents of a ToT calibration file: ToT_calib_function of type TF1 represents the charge dependency on ToT. The number of its constants equals the number of histograms (type TH2D). Thus in this example the calibration function is a polynomial of degree two.

for each pixel, before the testbeam measurements are performed. These values are filled into histograms, which are saved together with the ToT function in a root file and can be read by tbmon. The root file contents are shown in figure 5.6. Tbmon evaluates the function for each pixel applying the corresponding constants.

6. Results

Eventually tbmon is supposed to make full analyses for DUTs with arbitrary pixel geometries available, as it has done for rectangular pixels. This implies that general analyses are performed like before, but pixel based analyses are executed for each pixel geometry separately. The same applies to plots, which are supposed to show the different pixel shapes including evaluations as for instance in-pixel efficiency and charge sharing.
For these purposes two issues need to be covered: tbmon must be able to store and apply arbitrary pixel geometries and their arrangement. Therefore the necessary methods, which build the interface to eventbuilders and analyses, need to be supplied as well as the essential framework, that for example comprises communication with the DUT config. These foundations have been established during this thesis, as described so far.
To make tbmon provide its entire scope, eventbuilders and analyses need to apply the created interfaces to the data. These adaptions are not made yet and are indispensable for proper results. Two principal issues, which need to be covered, are the η-correction and the matching conditions. Possible techniques are introduced in chapter 7.

```
1  Info    : You have defined 2 pixel geometries in total.
2  Info    : There are 0 pixels masked by DutConfig.
3  Info    : 26880 pixels of 26880 pixels are set.
4  Info    : The area filled out by pixels is 3.360000 cm^2, while
       the actual dut area is 3.360000 cm^2.
5  Warning: The DUT does not seem to have edge pixels.
6  Info    : Pixel geometry 0 has its geometric center at
       (245.000000,15.000000).
7            Its max pitch is 450.000000 mum * 50.000000 mum.
8            Its area is 12500.000000 mum^2.
9  Info    : Pixel geometry 1 has its geometric center at
       (205.000000,35.000000).
10           Its max pitch is 450.000000 mum * 50.000000 mum.
11           Its area is 12500.000000 mum^2.
```

Listing 6.1: Text output of `checkDUT`. It shows that the definition and arrangement of pixel geometries works.

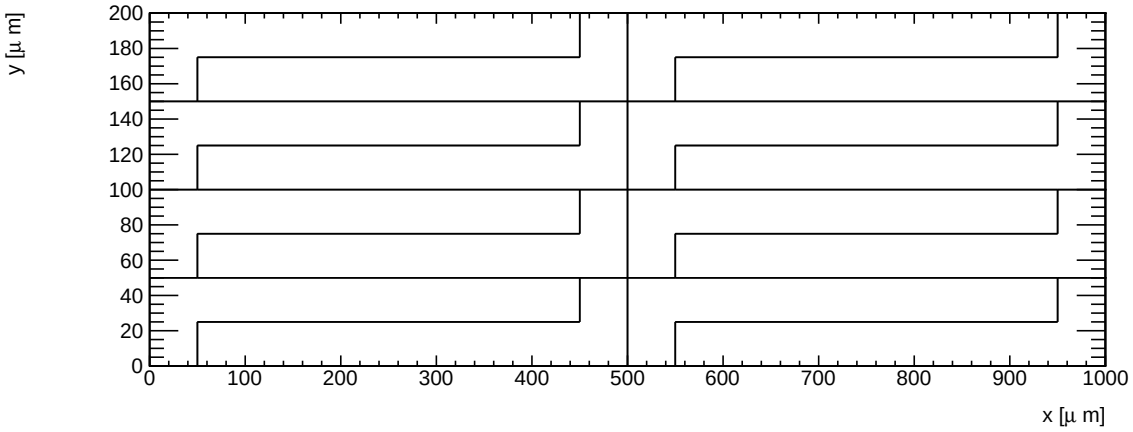

Figure 6.1.: Tbmon plots a DUT containing L shaped pixels. In this figure an enlarged view of the DUT is shown.

A tool, that examines the consistency of the DUT config, is useful, as inexperienced users might commit some errors composing a DUT config file. This additional feature has been implemented and can be performed using tbmon. If the config syntax is correct, the check of consistency can be performed and produces error-, warning- and information messages. For example it gives an error, if not every FE chip cell is logically associated with a pixel geometry. Warnings alert that perhaps not the entire range of functions is available. Information messages give a basic feedback about the DUT. In listing 6.1 the output for L shaped geometries[1] is shown.
The text output of tbmon indicates that the DUT config file operates as required. There are mostly information messages, which state the expected properties. The only warning is produced, because the property "edge" has not been defined for pixels, that are associated with edge columns or rows. This setting was deliberately neglected.
In addition, tbmon also produces a graphical output illustrating the arrangement of pixels on the sensor. This is another way for the user to ascertain that there occur no mistakes in DUT config. A sketch of the DUT filled out with L shaped pixels is depicted in figure 6.1.

As mentioned in chapter 4, not only the pixel shape, but also the arrangement of rectangular pixels is currently changed in sensor development. For testbeam measurements a DUT was built containing usual rectangular pixels with dimensions of 250 µm × 50 µm, where every row is shifted by half a pixel pitch in x with respect to the adjacent rows. This setup can now be defined as well. The graphical tbmon output is shown in figure 6.2. In this way also the double chip module, which is part of IBL, can be described.

The text output and the DUT plots, which were added to tbmon, do not only indicate if

[1] Compare with chapter 5.3

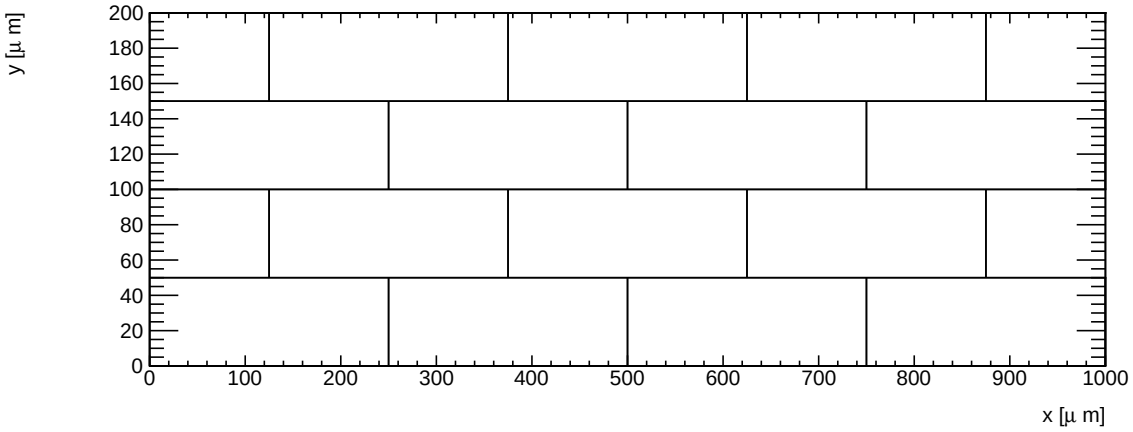

Figure 6.2.: Staggered pixels can be designed recently. Here an enlarged part of the tbmon output illustrates the pixel arrangement.

the config file is accurately set up, but they also show that the storage of arbitrary pixel geometries operates as required. The arrangement of pixels is handled correctly as well. Thus tbmon is now able to "understand" the shape of every pixel geometry, which is based on rectangles. In addition, it is able to arrange them in any order, which corresponds to the FE chip.

7. Outlook

During this thesis tbmon has been adapted to arbitrary pixel geometries. However, for complete results, several adjustments modifying the way of analysis need to be accomplished. For example the η-correction, which was briefly mentioned in the context of eventbuilders, needs to be adapted to arbitrary pixel shapes. A different technique is outlined in this chapter. Another issue, that needs to be covered, is the matching condition between clusters and tracks. A possible solution for this issue is presented as well.

7.1. η-correction

A center of mass method is a plausible approximation for the calculation of a cluster center position of rectangular pixels. Nevertheless for testbeam measurements with low angles, diffusion is the main reason for charge spread. As the dimensions of the generated charge cloud are quite small with respect to the pixel pitches, barely clusters with more than two hits occur. In this case the linear center of mass algorithm is not reasonable any more, as charge sharing only occurs in close vicinity to pixel edges. Thus the η-correction [20] is used in tbmon. It takes this effect into account by correcting the center of mass position applying a shift towards the joint edge for two hit clusters. The described method does not work correctly for arbitrary pixel shapes. Thus another technique could be the following one.

When the sensor atoms get ionized, the released charges drift to the electrodes. As diffusion is the most important cause for charge spread, the charge cloud has a Gaussian shape. In a simplified model it can be assumed to have a cylindrical shape, which projected onto the sensor area corresponds to a circle. If the charge cloud covers more than one pixel, a multiple hit cluster is observed. The deposited charge in one pixel depends on the cloud area, which overlaps the pixel. Then the equation

$$\underbrace{\frac{\text{cloud area overlapping the pixel}}{\text{total cloud area}}}_{\text{area fraction}} = \underbrace{\frac{\text{charge in pixel}}{\text{charge in cluster}}}_{\text{charge fraction}}$$

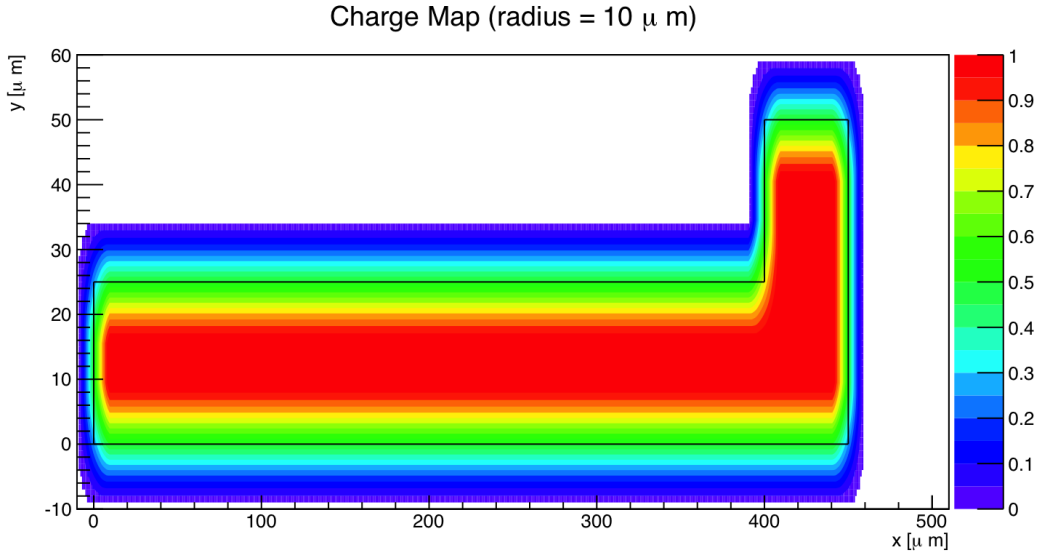

Figure 7.1.: In the charge map the radius of the charge cloud is defined to be 10 μm. To determine the value at a certain position in the diagram, a circle is made around this position. This value is calculated according to the part of the circle being inside the pixel.

is true. The charge information is known and equals the area fraction. As the charge cloud radius is determined using the diffusion equation, information about the track position can be obtained. For every possible track position the area fraction can be computed by applying a circle around this track coordinate. If it equals the charge fraction, the reconstructed track location is reasonable.

To illustrate this description, figure 7.1 shows a charge fraction map for an L shaped pixel. Track reconstruction benefits, as the charge fraction in a pixel can be associated with a certain region. For instance a charge fraction of 30% states that the track coordinate is in the cyan area. Adding the same information from other cluster pixels, a more significant map can be developed. Analyzing these values can lead to a more precise track reconstruction, which is valid for arbitrary geometries. As this calculation is based on discrete values, it also enables computation of uncertainties.

7.2. Matching Condition

After combination of hits to clusters, they are matched to tracks by comparing the cluster position with the track coordinates. In the former version of tbmon the matching condition was that the track is located within a two pixel pitch frame around any cluster pixel.

As a pixel with an arbitrary geometry does not have unambiguous pitches any more, this condition needs to be changed. A desirable solution would also take into account the cluster shape and ToT distribution to match a track with a cluster. As these pieces of information are combined in the cluster center, a comparison of the cluster center and the track coordinate is a reasonable approach. If they are closer than a “matching radius”, they are matched.
The matching radius could be defined globally and enables a more precise adjustment than the former method. If the cluster center calculation is performed like described in chapter 7.1, also reconstruction uncertainties for the hit position can be calculated and taken into account. In this way also the pixel shape would be considered, when matching clusters and tracks.

A. General Structure of tbmon

To provide a flexible program code, tbmon is arranged modular. All eventbuilders and analyses have according to their methods the same, inner structure. For a user this makes it possible to easily add new eventbuilders or analyses, as their methods are called automatically. The user only needs to register the method once in a config file. In table A.1 the sequence of tbmon is shown.

There are three different config files enabling the user to change cuts and settings without

	Methods	Eventbuilders	Analyses
1	`main`		
2	`core::loop`	`init`	
3			`init`
4	`looper::loop`		
5	`for runs`		
6	`looper::initRun`		
7		`initRun`	
8			`initRun`
9	`looper::eventLoop`		
10	`for events`		
11	`core::buildEvent`	`initEvent`	
12		`buildEvent`	
13	`looper::eventLoop`		`event`
14	`looper::finalizeRun`		`finalizeRun`
15		`finalizeRun`	
16	`looper::finalize`		`finalize`
17		`finalize`	

Table A.1.: Sequence of eventbuilders and analyses in tbmon: The left column names the position, at which the eventbuilders' and analyses' methods are called. Their method names are mentioned in the right columns.

modifying and compiling the code. In `main` they are read and saved into a `TBConfig`. Afterwards an object of `TBCore`, which holds all the testbeam data and calls the `Looper`, is created. The `Looper` arranges the loop over both the runs and the events. In the beginning (table A.1, between line 5 and 6) it reads the run based testbeam data file "tbtrack.root".

Bibliography

[1] P. W. Higgs, *Broken Symmetries and the Masses of Gauge Bosons*, Phys. Rev. Lett. **13** (Oct, 1964) 508–509. http://link.aps.org/doi/10.1103/PhysRevLett.13.508.

[2] ATLAS Collaboration, *ATLAS Detector and Physics Performance Technical Design Report*, Tech. Rep. CERN-LHCC-99-014 and CERN-LHCC-99-015, CERN, Geneva, 1999.

[3] J. Große-Knetter, *Vertex measurement at a hadron collider. The ATLAS pixel detector.* Habilitation, Universität Bonn, 2008.

[4] J. Weingarten, *System test and noise performance studies at the ATLAS pixel detector.* PhD thesis, Universität Bonn, 2007.

[5] N. Wermes and G. Hallewel, *ATLAS pixel detector: Technical Design Report.* Technical Design Report ATLAS. CERN, Geneva, 1998.

[6] M. Cristinziani, *The ATLAS pixel detector*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **582** (2007) no. 3, 714–718.

[7] F. G. Hügging, *Der ATLAS-Pixelsensor.* PhD thesis, Universität Dortmund, 2001.

[8] C. Gemme, *The ATLAS Upgrade programme*, Tech. Rep. ATL-PHYS-PROC-2012-104, CERN, Geneva, Jun, 2012.

[9] J. Bilbao de Mendizabal, *The ATLAS Insertable B-Layer (IBL) Project*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment (2013) .

[10] G. Marchiori et al., *Recent Results of the ATLAS Upgrade Planar Pixel Sensor R&D Project*, arXiv:1109.3047 [physics.ins-det].

[11] T. Wittig, *Slim edge studies, design and quality control of planar ATLAS IBL pixel sensors*. PhD thesis, Technische Universität Dortmund, 2013.

[12] C. Da Via et al., *3D silicon sensors: Design, large area production and quality assurance for the ATLAS IBL pixel detector upgrade*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **694** (2012) no. 0, 321 – 330.

[13] The ATLAS IBL Collaboration, *Prototype ATLAS IBL modules using the FE-I4A front-end readout chip*, Journal of Instrumentation **7** (Nov., 2012) 1010P, `arXiv:1209.1906 [physics.ins-det]`.

[14] P. Grenier, *Silicon Sensor Technologies for the ATLAS IBL Upgrade*, Physics Procedia **37** (2012) no. 0, 874 – 881.

[15] M. Capeans et al., *ATLAS Insertable B-Layer Technical Design Report*, Tech. Rep. CERN-LHCC-2010-013. ATLAS-TDR-19, CERN, Geneva, Sep, 2010.

[16] C. Hu-Guo et al., *A ten thousand frames per second readout MAPS for the EUDET beam telescope*, in *Topical Workshop on Electronics for Particle Physics*, pp. 47–51. 2009.

[17] J. Weingarten et al., *Planar pixel sensors for the ATLAS upgrade: beam tests results*, Journal of Instrumentation **7** (2012) no. 10, P10028. `http://stacks.iop.org/1748-0221/7/i=10/a=P10028`.

[18] J. Behr, *Test Beam Measurements with the EUDET Pixel Telescope*, 2010. EUDET-Report-2010-01.

[19] I. Rubinskiy, *Irradiation and beam tests qualification for ATLAS IBL Pixel Modules*, Tech. Rep. ATL-INDET-PROC-2012-007, CERN, Geneva, Mar, 2012.

[20] V. Heijne, *Characterisation of the Timepix Chip for the LHCb VELO Upgrade*. Master thesis, University of Amsterdam, 2010.

List of Figures

2.1. Sketch of the ATLAS detector. 3
2.2. Sketch of the ATLAS Pixel Detector. 5
2.3. Schematic view of type inversion in a silicon sensor. Left is before, right is after type inversion. After type inversion the pixels are surrounded by a depletion zone even without bias voltage. Figure extracted and modified from [7]. 6
2.4. The upper sketch depicts the arrangement of guard rings and edge pixels in the current Pixel Detector. In the slim edge design the inactive region was significantly decreased. The guard rings are located at the sensor back, the pixels at the front. 8
2.5. The pixel electrodes are inside the silicon bulk. A crucial advantage is the lowered distance of two electrodes [13]. 9

3.1. Sketch of a testbeam setup using the EUDET telescope [16]. It is also possible to test a couple of DUTs at once. 12

4.1. In-pixel efficiency plot of a neutron irradiated, planar pixel sensor with a bias voltage of 1000 V and a pixel pitch of 250 µm × 50 µm. The dashed rectangle illustrates one pixel, one half of the neighbouring pixel is plotted as well. The efficiency information of the whole DUT is folded into this plot. A decreased efficiency can be recognized at the bias electrode pixel edge. Figure extracted and modified from [19]. 20
4.2. Storage of files for tbmon analyses. The main advantage is that all configuration is saved together with the results. In this way the analyses are reproducible at a later time. 22

5.1. Former description of DUTs: In the middle there is a large amount of standard pixels. The extension in x and y depends on the number of columns and rows of the FE chip. At the left and right edges there is one column of edge pixels having a different pitch. 23

5.2. Two pixel geometries in comparison. The FE chip was initially designed for the upper one. To study narrower pitches and still contact the bump bonds indicated by the solid black circles, an L shaped pixel is used (lower sketch). 24
5.3. A pixel, based on rectangular shapes, is spanned by smaller, red colored rectangles. There can be different possibilities, two of them are shown in this figure for the L shape. 25
5.4. Inner structure of a DUT. Every DUT holds a list of different pixel geometries, which again hold a list of underlying rectangles. Besides that there is an array with the same number of columns and rows as the FE chip. It represents the DUT area and by pointing to geometries states the arrangement of pixels on the DUT. 26
5.5. The left side shows the actual pixel shape, the right side represents the array of pointers. The colors illustrate, to which pixel geometry the array entries point. 28
5.6. Contents of a ToT calibration file: ToT_calib_function of type TF1 represents the charge dependency on ToT. The number of its constants equals the number of histograms (type TH2D). Thus in this example the calibration function is a polynomial of degree two. 29

6.1. Tbmon plots a DUT containing L shaped pixels. In this figure an enlarged view of the DUT is shown. 32
6.2. Staggered pixels can be designed recently. Here an enlarged part of the tbmon output illustrates the pixel arrangement. 33

7.1. In the charge map the radius of the charge cloud is defined to be 10 µm. To determine the value at a certain position in the diagram, a circle is made around this position. This value is calculated according to the part of the circle being inside the pixel. 36

Danksagung

An dieser Stelle möchte ich mich bei allen, die zum Gelingen dieser Arbeit beigetragen haben, bedanken.
Mein Dank gilt Herrn Prof. Dr. Arnulf Quadt, der mir diese Bachelorarbeit am II. Physikalischen Institut ermöglicht hat.
Bei jeglicher Art von Fragen konnte ich mich an die gesamte Hardwaregruppe wenden und bekam immer hilfreiche Antworten. Dazu zählen unter anderem Julia Rieger, die sich mit der nötigen Physik, und Johannes Agricola, der sich mit der nötigen Informatik auskannte. Ganz besonders danke ich Matthias George, der sich viel Zeit genommen hat, mir einzelne Funktionen aus tbmon nahezubringen, Testbeamaufbaus zu erläutern, und mir immer Rede und Antwort stand.
Dank für das Korrekturlesen der Arbeit gilt Jens Weingarten, Matthias George und Filip Podjaski.
Dafür, dass ich während der Bachelorarbeit häufig Abwechslung auf dem Haus genießen konnte, haben meine Bundesbrüder gesorgt. Vielen Dank für die schöne und lehrreiche Zeit in Göttingen!
Ganz besonders möchte ich meiner Familie danken, die mich bei meinem Studium unterstützt – Sowohl finanziell, als auch mental. Zu letzterem haben in besonderem Maße meine Geschwister beigetragen, die mich immer wieder am Telefon zum Lachen brachten.

Erklärung nach §13(8) der Prüfungsordnung für den Bachelor-Studiengang Physik und den Master-Studiengang Physik an der Universität Göttingen:

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe.
Darüberhinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nichtauszugsweise, im Rahmen einer nichtbestandenen Prüfung an dieser oder einer anderen Hochschule eingereicht wurde.

Göttingen, den 15.07.2013

(Konstantin Lehmann)