Palacký University Olomouc
Faculty of Science
Department of Experimental Physics

DIPLOMA THESIS

Simulation of DESY and CERN test beam measurements for the ATLAS ITk strip project

Author:	**Bc. Radek Přívara**
Study programme:	N1701 Physics
Field of study:	Applied Physics
Study form:	Full time
Supervisor:	Mgr. Jiří Kvita, Ph.D.
Consultant:	RNDr. Jiří Kroll, Ph.D.
Submission date:	May 2020

Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Katedra experimentální fyziky

DIPLOMOVÁ PRÁCE

Simulace testovacích měření pro projekt ATLAS ITk strip na svazcích v laboratořích DESY a CERN

Vypracoval:	**Bc. Radek Přívara**
Studijní program:	N1701 Fyzika
Studijní obor:	Aplikovaná fyzika
Forma studia:	Prezenční
Vedoucí diplomové práce:	Mgr. Jiří Kvita, Ph.D.
Konzultant diplomové práce:	RNDr. Jiří Kroll, Ph.D.
Termín odevzdání práce:	květen 2020

Bibliographical identification

Author's first name and surname	Bc. Radek Přívara
Title	Simulation of DESY and CERN test beam measurements for the ATLAS ITk strip project
Type of thesis	Master
Department	Department of Experimental Physics
Supervisor	Mgr. Jiří Kvita, Ph.D.
Consultant	RNDr. Jiří Kroll, Ph.D.
The year of presentation	2020
Abstract	This work focuses on simulations of a silicon strip detector prototype developed by the ATLAS ITk Strip collaboration, composed of an ATLAS17LS sensor and read-out electronics in Star architecture, using Allpix-Squared and Athena simulation frameworks. The obtained data is processed and analysed using Python3 scripts and the results are compared to test beam measurements performed at CERN and DESY. The primary goal is to find the optimal simulation parameters to accurately reproduce experimental data obtained by test beam measurements.
Keywords	Allpix-Squared, Athena, ATLAS17LS, simulation
Number of pages	68
Number of appendices	0
Language	English

Bibliografická identifikace

Jméno a příjmení autora	Bc. Radek Přívara
Název práce	Simulace testovacích měření pro projekt ATLAS ITk strip na svazcích v laboratořích DESY a CERN
Typ práce	Magisterská
Pracoviště	Katedra experimentální fyziky
Vedoucí práce	Mgr. Jiří Kvita, Ph.D.
Konzultant práce	RNDr. Jiří Kroll, Ph.D.
Rok obhajoby práce	2020
Abstrakt	Práce se zaměřuje na simulaci prototypu křemíkového stripového detektoru vyvinutého kolaborací ATLAS ITk Strip, skládajícího se z ATLAS17LS senzoru a vyčítací elektroniky typu Star, s využitím simulačních nástrojů Allpix-Squared a Athena. Získaná data jsou zpracována a analyzována vlastními skripty v jazyce Python3 a výsledky jsou porovnány s měřeními na testovacích svazcích v laboratořích CERN a DESY. Hlavním cílem je nalezení vhodných parametrů simulace, která bude co nejpřesněji reprodukovat experimentální data získaná měřením na testovacích svazcích.
Klíčová slova	Allpix-Squared, Athena, ATLAS17LS, simulace
Počet stran	68
Počet příloh	0
Jazyk	anglický

Declaration

I declare that I wrote my diploma thesis independently under the supervision of Mgr. Jiří Kvita, Ph.D with the use of the cited sources.

In Olomouc, May 31, 2020

.....

Bc. Radek Přívara

Prohlášení

Prohlašuji, že jsem předloženou diplomovou práci vypracoval samostatně pod vedením Mgr. Jiřího Kvity, Ph.D. a že jsem použil zdrojů, které cituji a uvádím v seznamu použitých pramenů.

V Olomouci dne 31. května 2020

.....

Bc. Radek Přívara

Acknowledgement

I would like to express my sincere gratitude to my supervisor Mgr. Jiří Kvita, Ph.D. and to consultant RNDr. Jiří Kroll, Ph.D. for their guidance, helpful suggestions and words of encouragement.

Many thanks to Edoardo Rossi for sharing his insight into Allpix-Squared simulations and to Nicholas Styles and Helen Hayward for their assistance with setting up simulations in the Athena framework.

I am also very grateful to members of the ATLAS ITk strip test beam and irradiation group and of the Czech ATLAS ITk collaboration for welcoming me into their communities.

Lastly I want to thank my family and friends for their continued support.

The measurements leading to the results shown in this work have been performed at the Test Beam Facility at DESY Hamburg, Germany, a member of the Helmholtz Association (HGF).

The work on this thesis has been supported by the funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 654168.

The author gratefully acknowledges the support from the project IGA_PrF_2020_007 of Palacky University.

Contents

Introduction

The Large Hadron Collider (LHC) at CERN is planned to undergo an upgrade to High-Luminosity LHC (HL-LHC). The increased rate of interactions imposes stricter requirements on performance and radiation hardness of all the components of the ATLAS detector. As a result, the current Inner Detector will have to be replaced by the all-silicon Inner Tracker (ITk) which is designed to meet these criteria.

The ITk will use a new generation of silicon strip detectors, which have been developed by the ATLAS ITk strip collaboration and are now manufactured within the pre-production phase of the project. These sensors are extensively tested by the ATLAS ITk strip test beam and irradiation group to ensure their performance meets the requirements even at their expected end-of-lifetime.

Testing of silicon detectors is typically conducted at test beam facilities such as CERN or DESY. However, such measurements are time consuming and due to high demand for beam time from the physics community, the availability of these facilities is very limited. Simulations of detectors using complex software frameworks present a valuable way to complement these measurements, as they can be performed on virtually any hardware at any time and can provide experimental flexibility which is more difficult to achieve during test beam measurements.

This work explores the possibilities of simulating ATLAS ITk strip detectors in two simulation frameworks – Allpix-Squared and Athena. The obtained results are compared to data from test beam measurements of real prototypes. Adjustments to the simulation parameters are carried out and additional effects are implemented to improve the agreement between the simulations and the test beam data.

The first section of this work deals with theoretical description of silicon strip detectors. The second section includes a general overview of the ATLAS detector and focuses on the description of the ATLAS ITk, where the strip detectors are located. The third section presents the Allpix-Squared simulation framework. The fourth section details the simulations performed in this framework, discusses the influence of various configuration parameters and presents the obtained results. The fifth section serves as an overview of the Athena framework and in the sixth section the results obtained with this framework are discussed.

1 Strip Detectors

This section serves as an introduction to the design of silicon strip detectors. It explains their composition as well as the basic working principles. Finally, the characterization of a silicon strip detectors with a binary readout is described.

1.1 Semiconductor detectors

Semiconductor solid-state detectors offer a number of advantages over other types of detectors, such as scintillators or gas-filled detectors. Semiconductor detectors used for spectroscopy of charged particles are primarily made of silicon, in gamma-ray spectroscopy germanium is widely used. Their dimensions can be much smaller compared to gas-filled detectors as a density of a solid material is around 1000 times greater than of a gas. The energy resolution of semiconductor detectors is also very high – the ionization energy of silicon is about 3.6 eV compared to more than 30 eV for a gas-filled detector [1]. Scintillation detectors also exhibit poor energy resolution compared to solid-state detectors.

Compact dimensions, high energy resolution and, in case of segmented semiconductors, excellent spacial resolution make semicondutor detectors very desirable for use in the innermost layers of large accelerator detectors such as the ATLAS detector. There they are used as tracking detectors, responsible for reconstructing the trajectories of charged particles and for identification of primary and secondary vertices. However, one of the drawbacks is that semiconductor detectors' performance suffers due to damage caused by radiation. In the ATLAS detector these devices are placed close to the points of proton-proton or heavy-ion interactions and therefore are exposed to significant radiation. Accurate prediction of expected accumulated radiation dose and testing of irradiated prototypes of the detectors are vital to ensure they are operating at acceptable levels for the entirety of their planned lifetime.

1.1.1 Band structure

Crystalline materials with a periodic lattice exhibit bands of allowed energy level for electrons. Electrons bound to lattice sites in a crystal inhabit the valence band, whereas the conduction band is inhabited by electrons that are free to move through the solid and thus contribute to the electrical conductivity. The energy difference between the conduction and the valence band is called a bandgap. Semiconductors exhibit a fairly small bandgap – at room temperature it is 1.14 eV for silicon and 0.64 eV for germanium [2], compared to more than 5 eV for insulators [3].

An electron can be elevated to the conduction band by gaining, at minimum, an amount of energy corresponding to the bandgap. This energy can come from thermal excitations, which can happen at any non-zero temperature, from optical excitations caused by a photon, or from a collision with a charged particle passing through the volume of

the semiconductor. As a result of this excitation, an electron is created in the conduction band and a vacancy (called a hole) is created in the valence band, so in summary an electron-hole pair is created.

Both the electrons and the holes will start to move if an electric field is applied to the volume. Because of the opposite charge of electrons and holes, they will move in opposite directions. The motion caused by the electric field is called drift motion and it is parallel to the direction of the field. The charge carriers will also have a thermal velocity in a random direction. When the electric field intensity reaches a certain level, the drift velocity stops increasing and reaches a saturation velocity. Semicondutor detectors are usually operated with an applied electric field of sufficient intensity to reach the saturation velocity. For a typical saturation velocity of 10^7 cm/s [3] and a drift distance of 300 μm (a typical thickness of semiconductor strip detectors), the charge carriers are collected in less than 3 ns. Semiconductor detectors therefore have one of the fastest responses of all radiation detectors.

1.1.2 Semiconductor types

A semicondutor with no defects or additions to its crystal lattice would have all its free electrons and holes created only by thermal excitation. In this case the number of electrons and holes would be exactly equal, as a creation of a conduction band electron leaves behind a hole in the valence band. Such a semiconductor is called intrinsic. Practically it is impossible to obtain a truly intrinsic semiconductor, as some small level of impurities or defects is always present in the lattice, even in cases of high-purity silicon or germanium.

A small amount of impurities – atoms other than the primary silicon or germanium – may be intentionally added to modify the semiconductors' properties, mainly its electrical conductivity. In small impurity concentrations (less than 1 atom per million), these atoms will take place of a silicon atom in the lattice. A semicondutor with intentional impurities is called doped and based on the type of the impurity atoms, can be called either n- or p-type semicondutor.

If the impurity is a pentavalent atom (found in group V of the periodic table) it has five valence electrons. After all the covalent bonds are formed with the adjacent four silicon atoms in the lattice, one valence electron is left over. This electron is bound very lightly and is easily elevated to the conduction band. This added concentration of conduction band electrons shifts the balance between electrons and holes in such a way that the number of electrons is much higher and the number of holes much smaller than in an intrinsic semiconductor. Such a semiconductor is then called an n-type, because the negative charge carriers are in majority. Since the total amount of charge carriers is much higher than in a pure material, the electrical conductivity is also significantly higher.

An addition of a trivalent atom (from group III of the periodic table) with three valence electrons will leave one covalent bond unsaturated. The resulting vacancy is very similar

to a hole left behind by an electron. This time there are many more holes than conduction band electrons and because positive charge carriers are dominating, such a semiconductor is called a p-type. The increased concentration of charge carriers again leads to a more conductive material than an intrinsic semiconductor.

If impurities are present in such a way that the electrons and holes are balanced, that semiconductor is called compensated. These are usually designated as i-type due to their properties' similarity to intrinsic semiconductors.

Another special notation is often used for heavily doped materials with very high concentration of impurities which exhibit very high electrical conductivity. These are designated as n^+ or p^+ for a heavily doped n- or p-type semiconductor. Similarly a very lowly doped material is designated as n^- or p^-.

1.1.3 Semiconductor junction

Semiconductor detectors are based on a junction of n- and p-type semiconductors. At the junction area, electrons from the n-type volume will diffuse into the volume of the p-type where they quickly recombine with the present holes. Absence of these electrons in the n-type volume will create immobile positively charged impurities. Similarly holes from p-type volume will diffuse into the n-type volume and recombine with the electrons, leaving behind immobile negatively charged impurities in the p-type. This leads to an overall positive charge on the n-side and a negative charge on the p-side of the junction.

This accumulated opposite charge on both sides of the junction creates an electric field which suppresses concentration and further diffusion of charge carriers. This region around the junction is called a depletion region and it is very useful as a medium for radiation or charged particle detection. When a charged particle passes through the depleted region, electron-hole pairs are created as a result of interactions with lattice atoms. Every electron created is pulled by the electric field toward the n-side, while holes are pulled to the p-side of the junction. This motion of charge carriers constitutes the basic electric signal indicating a passage of a charged particle.

A semiconductor junction with no external electric field applied is called an unbiased junction. It can be used as a detector but its performance would be very poor. Due to a low intensity of the electric field the charge carriers move slowly and with a high probability of recombination, the depletion region is small and the capacitance of such a junction is high, which leads to poor noise properties [3]. These drawbacks can be remedied by an application of an electric field in a way that negative voltage is applied to the p-side and positive voltage to the n-side. This is called a reverse-bias junction.

An application of a reverse-bias voltage on a junction causes the depletion region to extend even more. For a high enough bias voltage the depletion region can extend through the entire volume, such a detector would then be called fully depleted and this is how most semiconductor detectors are operated. In the reverse-bias mode the concentration of free

charge carriers is low, therefore the reverse current flowing through the junction is very small. A more intense electric field also improves timing resolution of the detectors due to the charge carriers reaching the drift saturation velocity.

1.1.4 Strip sensor

A reverse-biased junction can provide information about the energy loss of an incident particle, as the collected charge is proportional to that particle's energy loss in the sensitive volume. However obtaining accurate position of the passing particle is impossible with a single junction.

Position information can be obtained by having multiple electrodes amongst which the generated charge will split. The ratio of these collected charges in the electrodes confers information about the particle's position. Strip detectors use a thin layer of a highly doped material upon which thin metal electrodes are deposited, all placed on the bulk volume of the semiconductor. This way a number of separate reverse-biased p-n junctions are created. The charge from each junction – due to its shape called a strip – is collected and quantified separately, this process is called reading out. The location of a passing particle is then determined based on which strip is showing the signal. An example of a strip detector is shown in Fig. 1.

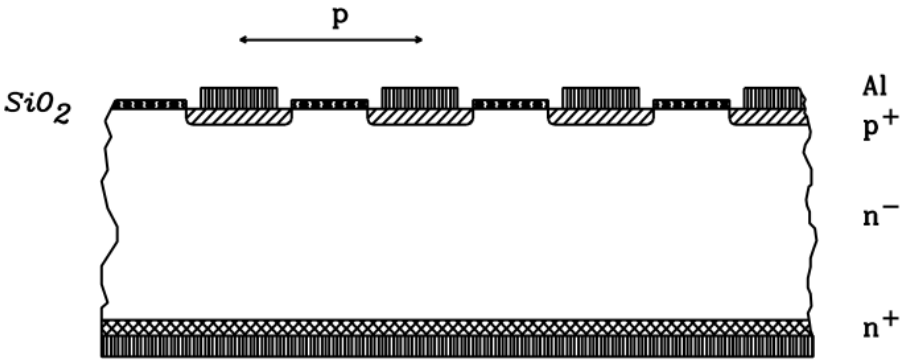

Figure 1: Cross-section of a strip detector. From [1].

There are two possible ways of reading out the strips. In a binary readout mode a global charge threshold is set. If the collected charge in a strip surpasses this threshold, a logical true value is read out from that strip and a hit is called. If the charge isn't sufficiently high, a logical zero is read out from the strip. Spacial resolution of a strip detector in binary readout mode is [1]

$$\sigma = \frac{p}{\sqrt{12}}, \tag{1}$$

where p is the distance between the centers of adjacent strips, called a strip pitch.

The second possible readout mode is analog. Analog readout preserves the signal shape and amplitude from every strip, which results in better spacial resolution and information

about the total energy loss of a passing particle. If the signal charge is collected on multiple strips, the location of a passing particle can be calculated as a center of gravity of the signal.

The choice between binary and analog readout mode depends on the spatial resolution requirements and on the necessity of energy information.

An array of strips on one side of the bulk material is only capable of determining one-dimensional location of the passing particle. A second array of strips on the opposite side of the bulk material, oriented perpendicularly to the first array, ensures that both electrons and holes are used for position measurement. This way a two-dimensional location information about the passing particle can be obtained. An illustration of a double-sided detector is shown in Fig. 2. Another possibility of obtaining a two-dimensional position with strip detectors is using two single-sided detectors with a different orientation of the strips.

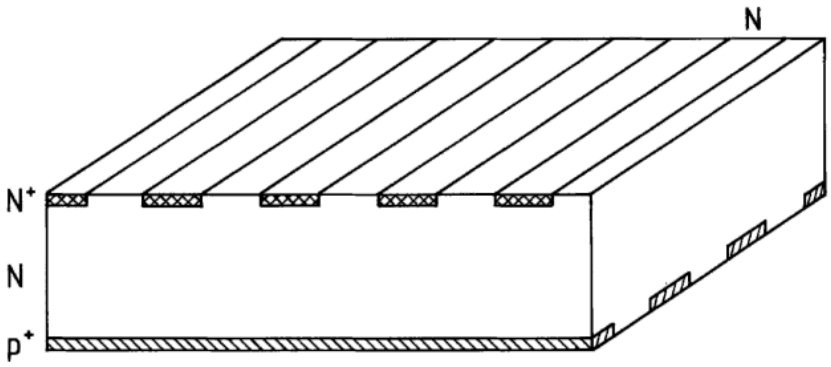

Figure 2: A double-sided strip detector. The specific configuration (n^+-n-p^+) is one of many possibilities. From [1].

1.1.5 Passage of particles through matter

The interactions of fast charged particles with speed $v = \beta c$ occur in single collisions, leading to ionization, atomic, or collective excitation. Most frequently the energy losses are small, in thin materials few collisions will take place and the total energy loss will show a large variance.

At low energies electrons and positrons primarily lose energy by ionization. While ionization loss rates rise logarithmically with energy, radiation losses rise nearly linearly and dominate at higher energies.

The mean rate of ionization energy loss caused by moderately relativistic charged heavy particles is well-described by the Bethe equation [4]

$$\left\langle -\frac{\mathrm{d}E}{\mathrm{d}x} \right\rangle = K\rho z^2 \frac{Z}{A}\frac{1}{\beta^2}\left[\frac{1}{2}\ln\frac{2m_e c^2\beta^2\gamma^2 E_{\max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2}\right], \tag{2}$$

where $K = 4\pi N_A r_e^2 m_e c^2$, N_A is the Avogardo's number, r_e is the classical electron radius,

m_e is the electron mass, ρ is the material density, Z and A is the atomic number and mass of the material, $\gamma = \frac{1}{\sqrt{1-\beta^2}}$ is the particle's Lorentz factor, I is the mean excitation energy of the material, $E_{\max}$ is the maximum energy transfer possible in a single collision and $\delta(\beta\gamma)$ is the density effect correction.

Equation (2) describes the mean rate of energy loss in the region $0.1 \leq \beta\gamma \leq 1000$ for intermediate-Z materials with an accuracy of a few percent.

The main problem with the mean energy loss is that it is weighted by very rare events with large single-collision energy deposits. With a sample size of hundreds of events an accurate value of the mean energy loss cannot be obtained. More easily measured is the most probable energy loss.

For detectors of moderate thickness the energy loss probability distribution is adequately described by the highly-skewed Landau distribution, shown in Fig. 3. The most probable energy loss is [4]

$$\Delta_p = \xi \left[\ln \frac{2m_e c^2 \beta^2 \gamma^2}{I} + \ln \frac{\xi}{I} + 0.2 - \beta^2 - \delta(\gamma\beta) \right], \tag{3}$$

where $\xi = (K/2)(Z/A)z^2(x/\beta^2)$ MeV for a detector with a thickness x in $\mathrm{g \cdot cm^{-2}}$. For $\beta\gamma \geq 100$ the most probable energy loss becomes [4]

$$\Delta_p = \xi \left[\ln \frac{2m_e c^2 \xi}{(\hbar\omega_p)^2} + 0.2 \right], \tag{4}$$

where $\hbar\omega_p$ is the plasma energy. While $\mathrm{d}E/\mathrm{d}x$ is independent of thickness, $\Delta p/x$ scales logarithmically with x.

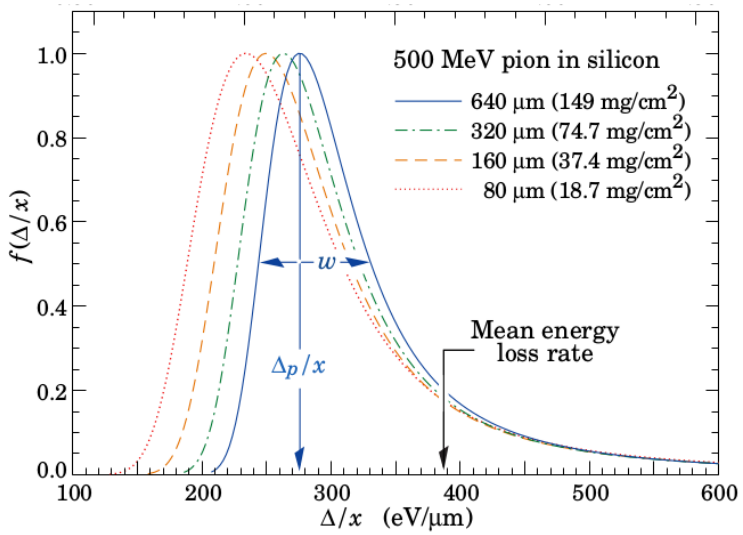

Figure 3: Energy loss distributions of 500 MeV pions in silicon, normalized to unity at the most probable value Δ_p/x. The width w is the full width at half maximum. From [4].

The Landau distribution fails to describe energy loss in thin absorbers such as silicon detectors. While Δ_p/x may be calculated adequately using Equations 3 or 4, the distributions are significantly wider than the Landau width $w = 4\xi$. For very thick absorbers

the distribution is less skewed but never approaches a Gaussian [4].

1.2 ATLAS ITk strip sensors

In the ATLAS ITk there are two main regions for the detector placement, called barrel and end-cap (EC) regions, both with specific demands on sensor shape and design. This layout of the ITk is discussed in detail in Section 2.3. The ATLAS ITk collaboration developed 2 types of barrel sensors and 6 types of EC sensors. While the geometrical shapes of barrel and EC sensors differ significantly, all these sensors are functionally equivalent. The aluminium strips on the silicon sensors are AC-coupled with n^+-type implants in a p-type float-zone silicon bulk (n^+-in-p FZ).

The current ATLAS Semiconductor Tracker (SCT) uses p^+-in-n sensor technology, for the ATLAS ITk the n^+-in-p technology was chosen, which provides higher amount of signal especially in sensors damaged by irradiation [5]. In the fluence range of 8–10·10^{14} $n_{\mathrm{eq}}/\mathrm{cm}^2$ the n^+-in-p sensors should deliver twice the charge compared to p^+-in-n sensors.

The target physical thickness is 300–320 μm, while the active thickness is expected to be at least 90 % of the physical thickness [6]. Close proximity to the accelerator interaction point and exposure to the resulting radiation mandates high radiation tolerance of the strip sensors. It is required of them to withstand the expected maximum fluence of $1.7 \cdot 10^{15}$ $n_{\mathrm{eq}}/\mathrm{cm}^2$ and the total ionising dose of 80 Mrad [7]. Both these values include a 50 % safety factor. The maximum bias voltage during operation will be 500 V, but the sensors are required to work with a bias voltage of up to 700 V.

The barrel sensors are designed with an expected active area of 96.640 x 96.669 mm^2, with 1280 read-out strips and a strip pitch of 75.5 μm [6]. There are two types of barrel sensors that populate the four barrel layers. One type has four rows of short strips (24.1 mm) and is used in the inner two layers, a module based on this sensor is called a short-strip module. The other type has two rows of longer strips (48.2 mm) and is suitable in the outer two layers, the corresponding module is called a long-strip module.

The EC sensors feature radial strips, which are pointing to the beam axis, to allow for a measurement of the $R\varphi$ coordinate. As a result, these sensors are wedge-shaped and have curved edges. The strip lengths of the six EC sensor types are based on how close to the beam axis they are to be positioned, varying from 19 mm in the region closest to the beam axis to 60.1 mm in the outermost region. Similar to the barrel modules, the EC modules feature either two or four rows of strips.

1.3 ATLAS ITk strip module

The basic unit of the ITk strip detector is a silicon strip module. A module is composed of a strip sensor and one or two printed-circuit boards, called hybrids, hosting the readout ATLAS Binary Chips (ABC) and Hybrid Controller Chips (HCC) in their Star configuration (ABCStar and HCCStar). The hybrids are glued to the silicon sensors with

electronics-grade epoxy.

A layout of a short-strip module is illustrated in Fig. 4, a prototype of a short-strip module is shown in Fig. 5. This prototype uses an older ABC130 architecture of the chipset, the module layout is nonetheless very similar.

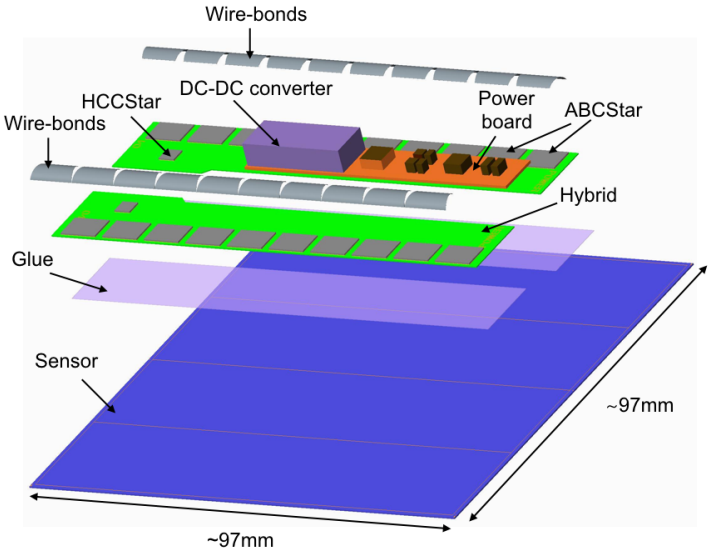

Figure 4: A layout of a short-strip barrel module. From [6].

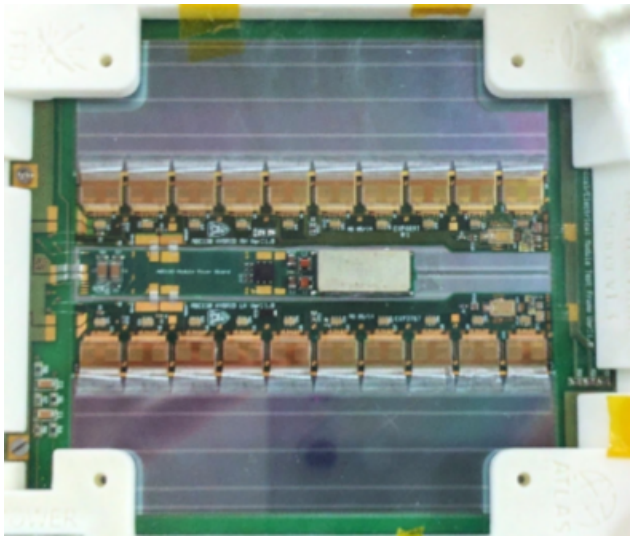

Figure 5: A prototype of a short-strip barrel module using ABC130 architecture.

As was discussed in previous subsections, a charged particle passing through the silicon sensor deposits a portion of its energy in this sensor volume, thus creating free charge carriers, which are collected on the n^+ strips. The signal induced on the aluminium strips AC-coupled with n^+ implants is then transmitted through aluminium wire bonds to individual channels of the front-end ABCStar chips, each containing 256 readout channels with pre-amplifiers and discriminators. With the ABCStar the signal on each channel is amplified, shaped and its discriminator then compares the signal charge to the charge threshold to provide a binary output. The HCCStar chips receive the signals from the

ABCStar chips, build packets and send them to the next control element of the structure hierarchy, called the End-of-Substructure (EOS) card. The HCCStar chips are also responsible for distributing the clock and control signals to every ABCStar. A diagram of the hybrids' electronics is shown in Fig. 6.

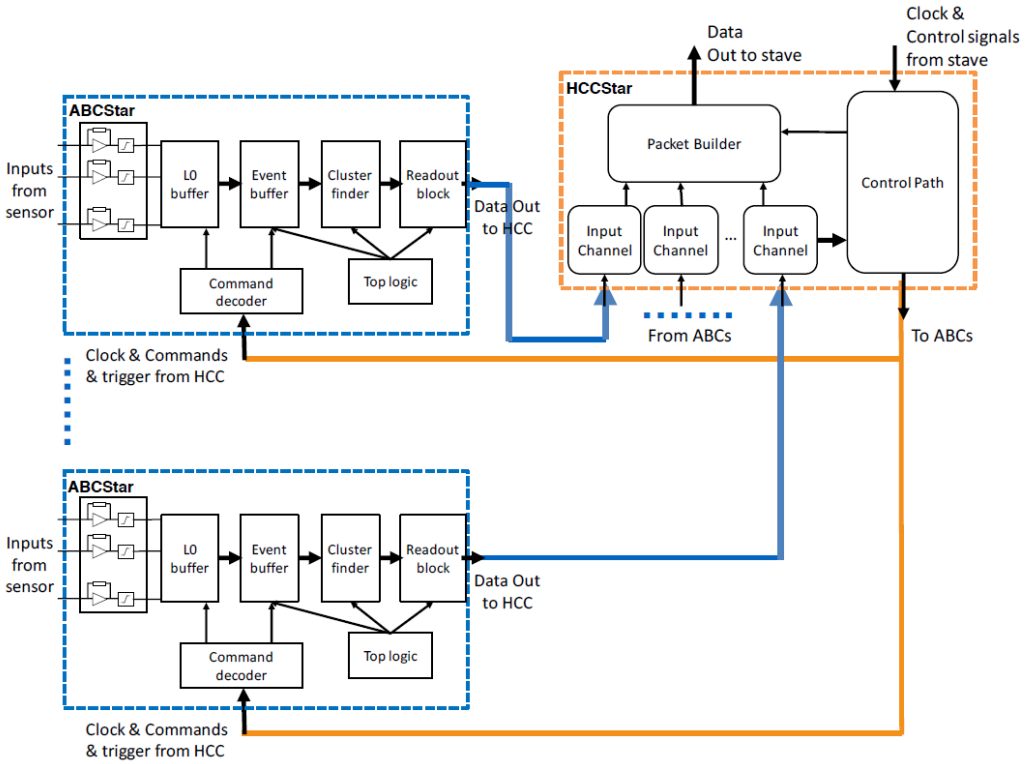

Figure 6: A schematic of the hybrid electronics. From [6].

A power board is placed between the two hybrids. It contains a low-voltage DC-DC power converter and a high-voltage switching circuit which provides the bias voltage. It also hosts an AMAC (Autonomous Monitor and Control) chip which is responsible for module-level monitoring of voltages, temperatures and a sensor leakage current. The power board is connected to the EoS card and distributes power to each hybrid.

1.4 Module characterisation

One of the most important characteristics of a module is its detection efficiency for a given charge threshold. It can be obtained either by injecting a constant charge produced by an internal calibration circuit into the individual readout channels or by a passage of a charged particle of known energy through the sensor volume. If the collected charge in a strip surpasses the given charge threshold, this event is accepted as a hit, otherwise it is rejected. This process is repeated many times at each threshold value and this way an average hit rate, defined as a ratio of hits to all events, is obtained. By plotting the average hit rate as a function of the charge threshold, an efficiency curve – also called an S-curve due to its shape – is obtained. Example efficiency curves are shown in Fig. 7.

For a threshold set at zero charge every injected charge will be accepted as a hit and

the average hit rate is 1. Likewise for very high thresholds, every event is rejected and the average hit rate approaches zero.

In an ideal case of noiseless chips the efficiency curve would be rectangular when injecting a constant charge, with efficiency staying at 1 for every threshold lower than the injected charge, then dropping immediately to zero for every threshold higher than the injected charge.

In real detectors the efficiency drop isn't sharp, but rather gradual. The primary causes for this are twofold. First, the amount of signal from a passage of a charged particle is dependent on that particle's energy loss in the volume. Since the particle's energy loss rate isn't constant but rather follows a Landau distribution, the amount of signal from particles with a constant energy varies slighty. The second reason is the existence of electronics noise, which is well described by a Gaussian distribution [6]. Therefore the variance of the signal is described by a convolution of a Gaussian and a Landau distribution. The probability density function of the Landau distribution does not have an analytic expression. The dependence of the efficiency on the threshold obtained from the measurement or computer simulation is described approximately, using a skewed complementary error function, which is discussed in Section 4.3.

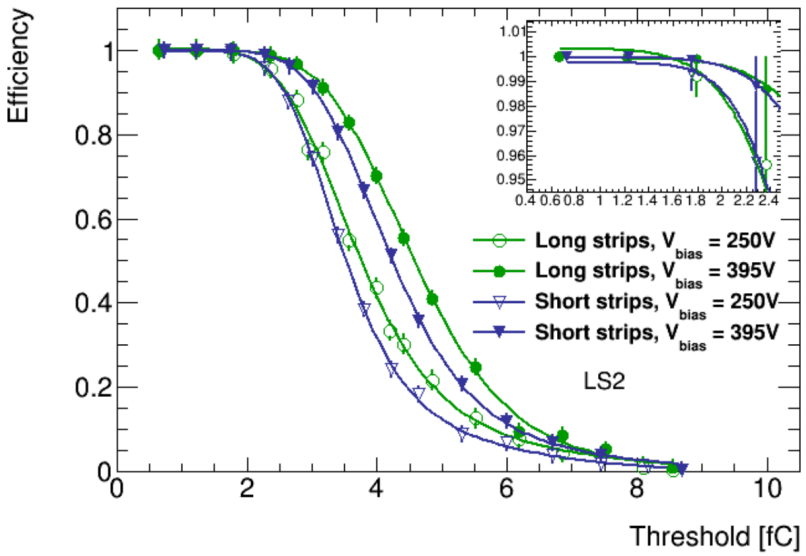

Figure 7: Examples of efficiency curves of long- and short-strip modules for two bias voltages. From [6].

The threshold for which the efficiency is exactly 0.5 corresponds to a median charge produced in the sensor by the passing particle. The amount of signal from the detector is however defined by the most probable value of the charge distribution, not the median value. Due to the asymmetry of the Landau distribution, the relationship between the median and the most probable value is not accurately defined, according to [8] the most probable value is about 8–15 % smaller than the median value.

To ensure the modules' performance is acceptable, the requirement is that at the end-

of-lifetime the modules must have a window of charge thresholds with simultaneously more than 99% detection efficiency and less than $1 \cdot 10^{-3}$ channel noise occupancy. Noise occupancy is defined as the ratio of hits caused by noise to all hits. These requirements are equivalent to having the signal-to-noise ratio of at least 10 [6]. The electronics noise is usually determined by runs without a particle beam, where the only hits detected are caused by the electronics noise itself.

Another characteristic of a module is an average cluster size, i.e. the number of strips with a hit caused by the passage of a single particle. This is also usually presented as a function of a charge threshold, an example plot is shown in Fig. 8. A cluster size relays information about a level of charge sharing among the strips. For low charge thresholds, even a small amount of charge leaking from a principal strip to an adjacent one can lead to both strips calling a hit, as their collected charges surpass the threshold. At higher thresholds this is unlikely to happen and a higher cluster size can instead be caused by different effects, such as by a creation of a δ-electron, a high-energy particle that travels a relatively large distance in the volume and can deposit enough charge in multiple strips for them to call a hit.

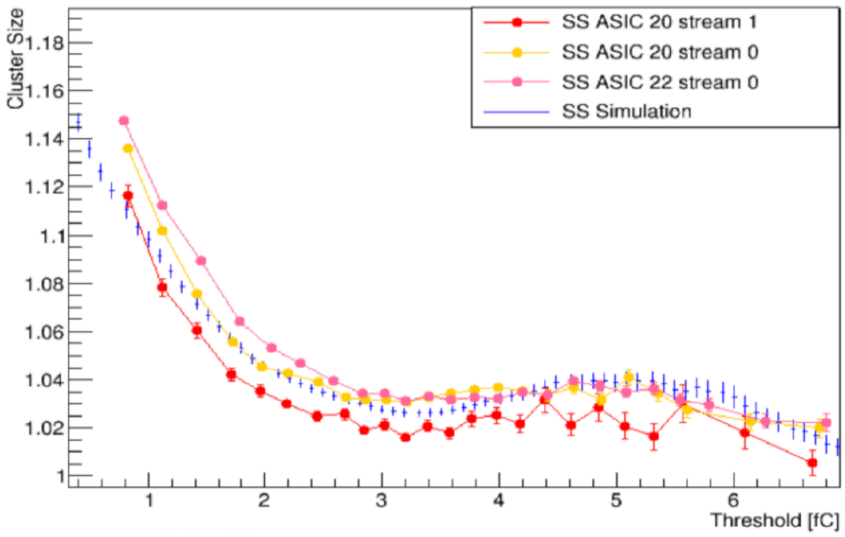

Figure 8: Cluster size plots of a prototype short-strip module obtained from test beam measurements on several ASICs and from simulations.

1.5 Test beam measurements

Test beam measurements represent a standard way to ascertain functionality and performance of a module. During the measurement, the tested module, called Device-Under-Test (DUT), is placed into a beam of known passing particles. To see how accurately a module can determine the position of a particle hit, the DUT is surrounded by a detector array called an EUDET telescope [9], which consists of six pixel detectors with good spatial resolution, three on each side of the DUT. Fig. 9 shows the DURANTA telescope at DESY. By interpolating the hits in the telescope with a line using the General Broken Lines (GBL)

algorithm [10] and comparing the intersection of the line with the DUT, a reference position of a hit in the DUT is obtained. The difference of this reference hit position and the hit position recorded by the DUT itself is called a residual. Residuals are one of the most important outputs of a test beam measurements and give information about spatial resolution of a detector.

For triggering, an EUDET telescope uses four scintillators with photomultiplier tubes, two placed in front of the first pixel detector and two placed behind the last one. A Trigger Logic Unit (TLU) monitors every detector in the telescope and ensures that the DUT is only read out when defined conditions are met, i.e. when each of the four scintillators or just a selection of them calls a hit. This eliminates the possibility of detecting random incident particles by the DUT.

To measure detection efficiency of the DUT an additional pixel detector can be placed into the telescope. The detection efficiency can then be calculated as the ratio of the number of hits in both the DUT and the pixel detector to the number of hits in the pixel detector.

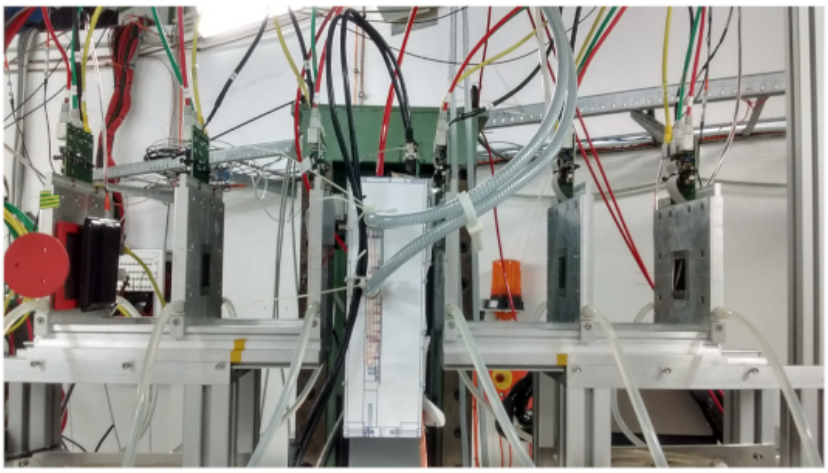

Figure 9: The DURANTA telescope at DESY, the central box houses the DUT. From [6].

2 ATLAS Experiment

A Toroidal LHC Apparatus (ATLAS) is one of four large experiments at the LHC at CERN, the other three being CMS, ALICE and LHCb. Search for the Higgs boson, dark matter particles or extra dimensions are a few of the wide range of physics topics it investigates. In 2012, both ATLAS and CMS detectors participated in the discovery of the Higgs boson.

Proton or heavy-ion beams accelerated by the LHC collide inside of the ATLAS detector. These collisions result in a number of new particles being created and moving out from the collision points in various directions. To detect these particles and to record their trajectories, energy and momentum, a total of six subdetector systems are arranged in separate layers around the points of collisions. Momentum measurement of charged particles is facilitated by a magnet system which bends their trajectories.

Due to the enormity of data coming from the subdetectors when collisions are happening, it is neither possible nor desirable to record every event. To reduce the amount of data to be saved, a system called "trigger" is used to choose the interesting events and instruct the subdetectors to record them. Analysis of the recorded events is made possible by data-acquisition and computing systems.

The ATLAS detector has the shape of a cylinder, it is 46 m long, has 25 m in diameter and weighs 7000 tonnes [11]. The four major components of the ATLAS detector are the Inner Detector, calorimeters, the Muon Spectrometer and the Magnet System. The layout of the ATLAS detector is illustrated in Fig. 10.

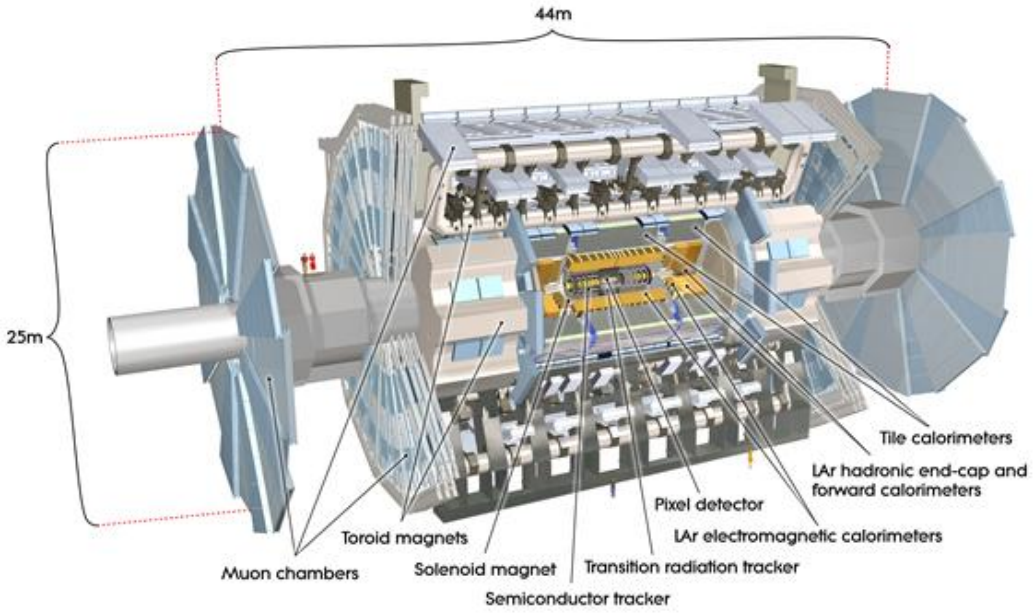

Figure 10: Computer generated image of the ATLAS detector. From [11].

2.1 ATLAS Inner Detector

As the innermost layer of the ATLAS detector, the Inner Detector has a key role in physics analyses and it is, among other things, responsible for the reconstruction of trajectories of charged particles, identification of primary and secondary vertices (points of interaction) and identification of electrons, photons, muons and tau-leptons produced in both proton-proton interactions and heavy-ion collisions.

The current ATLAS Inner Detector consists of three layers: the Pixel detector is the inner layer, followed by the SCT, where the strip detectors are located, and the Transition Radiation Tracker (TRT). The ATLAS Inner Detector layout is shown in Fig. 11.

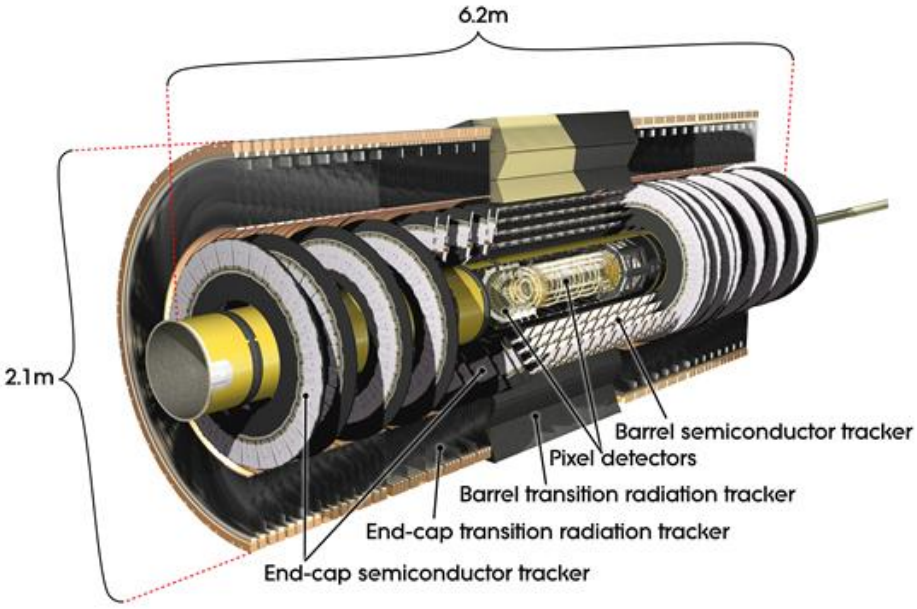

Figure 11: Computer generated image of the ATLAS Inner Detector. From [12].

When designing the detectors many parameters have to be considered. A frequency at which bunches of protons cross and interact, and an instantaneous luminosity L, which corresponds to a number of collisions per surface unit per second, determine requirements on spacial and timing resolution and read-out times of the detectors. A number of interactions per bunch crossing is called a pile-up μ.

A detector has to be able to perform at an acceptable level until its planned end of operation. To describe the total number of interactions over an entire span of operation, a concept of an integrated luminosity L_{int} is used. It is defined as a time integral of an instantaneous luminosity L

$$L_{\text{int}} = \int_0^T L(t)\ \mathrm{d}t \tag{5}$$

and its relevant unit is an inverse femtobarn (fb^{-1}). This quantity is a measure of a total amount of interactions observed during a certain time period. When multiplied by a cross-section σ of a specific interaction, it quantifies how many times this interaction

was observed

$$N = L_{\text{int}} \cdot \sigma. \tag{6}$$

When designing the detectors, an expected value of integrated luminosity by the end of operation is calculated. Since it relates to the total amount of interactions, it can be used to estimate a level of radiation damage the detectors will receive during operation. The current Pixel detector was designed to withstand the radiation damage corresponding to an integrated luminosity of 400 fb^{-1}, the SCT to a value of 700 fb^{-1} and the innermost pixel layer, called Insertable B-Layer (IBL), of 850 fb^{-1} [6].

The current ATLAS Inner Detector was designed for:

- 10 years of operation,
- an instantaneous luminosity $L = 1.0 \cdot 10^{34}\ \text{cm}^{-2}\,\text{s}^{-1}$,
- centre-of-mass energy of 14 TeV,
- 25 ns between bunch crossings,
- an average pile-up of $\mu = 23$.

2.2 ATLAS Inner Tracker

Preparations are currently under way for an upgrade of the current LHC to HL-LHC which plans to begin operation during 2027. It will be capable of achieving significantly higher instantaneous luminosities which will result in a much higher average pile-up and more radiation damage to the detectors. These operating conditions mandate a complete replacement of the entire ATLAS Inner Detector with an all-silicon detector called the ATLAS ITk. Similar to the current Inner Detector, the ITk layout will consist of an inner pixel detector and a large area strip detector as an outer layer. The higher average pile-up poses requirements on tracking performance of the ATLAS ITk, which should be equal or better than the performance of the current Inner Detector.

The ATLAS ITk is being designed for:

- 10 years of operation,
- an instantaneous luminosity $L = 7.5 \cdot 10^{34}\ \text{cm}^{-2}\,\text{s}^{-1}$,
- a total integrated luminosity of 4000 fb^{-1},
- 25 ns between bunch crossings,
- an average pile-up of $\mu = 200$.

2.3 ATLAS ITk Layout

In the ITk the silicon detectors are arranged in concentric cylinders – this region is therefore called the barrel – around the beam axis, with inner five layers of pixel detectors followed by four layers of strip detectors. The barrel region extends 1.4 m from the interaction point in both directions along the beam axis. Six disks of strip detectors and many rings of pixel detectors close up the forward regions which are called end-caps. To improve coverage and detection performance, some of the pixel modules are inclined towards the

interaction point. The layout is illustrated in Fig. 12, its implementation in a simulation framework is shown in Fig. 13.

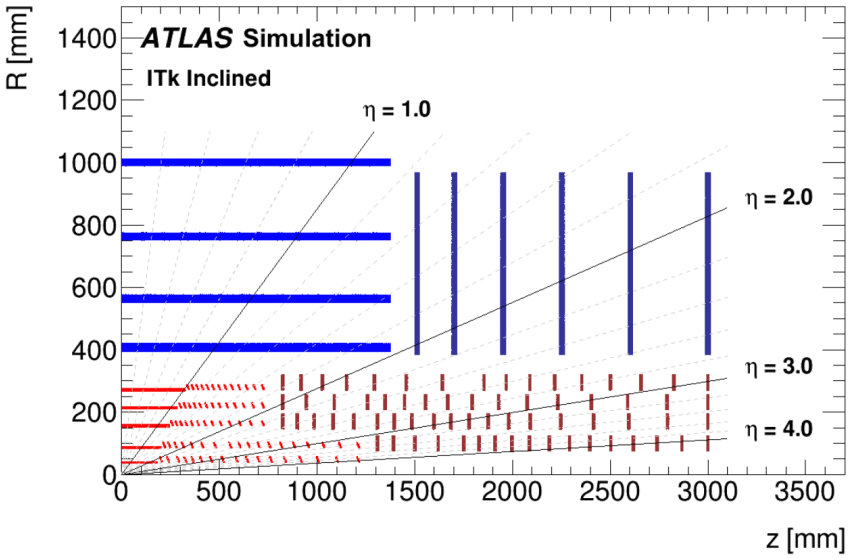

Figure 12: The ITK Layout, only one quadrant is shown. The pixel detectors are illustrated in red, the strip detectors in blue. The horizontal axis is the axis along the beam line with zero being the interaction point. The vertical axis is the radius measured from the interaction point. From [6].

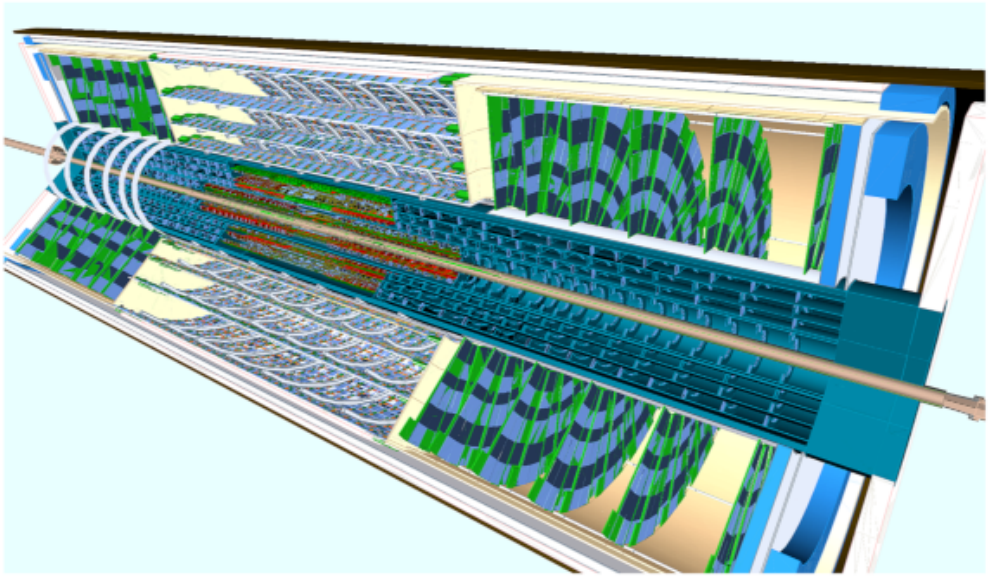

Figure 13: An implementation of the ITk in a simulation framework. From [6].

2.4 ATLAS ITk strip detector

The silicon detector modules are not placed individually in the layout. Instead, they are grouped on a substructure called a local support. In the barrel region this local support is called a stave and in the end-cap it is called a petal due to their respective shapes. The local supports are illustrated in Fig. 14. A low-mass core of a petal or a stave provides the

mechanical rigidity and support for the modules and it houses the electrical and cooling services. An EoS card connects and interfaces the modules with the electronics outside of the detector and it routes all the power and data links to and from the modules.

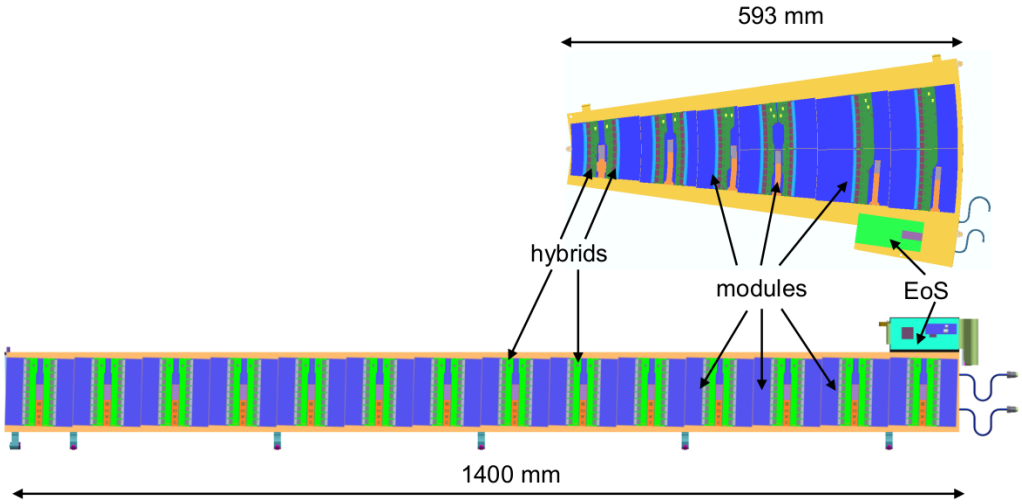

Figure 14: End-cap petal (upper) and barrel stave (lower) local supports. From [6].

The four barrel layers consist in total of 392 staves, each stave is populated with 28 strip modules. The inner two layers use short-strip modules, the outer two layers deploy long-strip modules.

In the two end-cap regions a total of 12 disks are placed. Each disk consists of 32 identical petals. Each petal has a total of 18 modules (9 on each side) and to optimally cover the wedge-shaped petal surface, six different sensor geometries are deployed. The specifications of the sensors are discussed in Section 1.2.

3 Allpix-Squared Framework

Allpix-Squared (alternatively AllPix^2) [13] is a framework for simulation of generic silicon pixel detectors. Written in the C++ programming language, its important feature is modularity – each part of a simulation is handled by a different module. Some modules work independently on each other (for example application of an electric field) and some require output from a module that precedes it in the simulation chain (e.g. a module which handles the propagation of charge carriers requires these charge carriers to be first created in the sensor volume by a previous module). Allpix-Squared depends on several libraries, specifically Geant4 [14] for the passage of particles through the sensor and the deposition of charge carriers in the sensor, ROOT [15] for data handling and storage and Eigen3 [16] for matrix algebra operations.

3.1 Installation

An Allpix-Squared simulation can either be run locally on one's own computer or on the LXPLUS, a cluster of machines running CERN CentOS 7 [17]. The latter is simpler, because it removes the need to build Geant4 and ROOT libraries by allowing to load these dependencies from the CernVM File System (CVMFS), a software distribution service.

The process is fairly simple and straightforward. First step is to clone the git repository from CERN GitLab. Then the necessary setup scripts for ROOT and Geant4 libraries have to be sourced to load these dependencies. Finally, the repository is built and installed using CMake, by following the typical order of commands. Listing 1 shows all the necessary steps to install Allpix-Squared on the LXPLUS.

```
git clone https://gitlab.cern.ch/allpix-squared/allpix-squared.git
cd allpix-squared
source etc/scripts/setup_lxplus.sh
mkdir build; cd build
cmake ..
make -j
make install
```

Listing 1: Installing Allpix-Squared on the LXPLUS.

3.2 Configuration

Running a simulation in Allpix-Squared requires three types of configuration files:

1. Main configuration file.
2. Geometry configuration file, which describes the experimental geometry.
3. Model configuration file with specific model description, such as size of the detector, number of pixels, etc.

Format of these configuration files was designed in such a way as to be easily readable even for people without extensive background in programming languages. Ease of use is one of the philosophies of Allpix-Squared and, combined with complete documentation available online [18], enables users to learn the basics of the framework very quickly.

3.2.1 Main configuration

Main configuration file passes to the program global simulation settings and details a list of modules to be instantiated along with their respective parameters. In the code, each module is prefaced by a module header in square brackets followed by mandatory and optional parameters which tune the behavior of a module. The code describing configuration of the main module `AllPix` is shown in Listing 2.

```
[AllPix]
log_level = "STATUS"
log_format = "DEFAULT"
detectors_file = "geometry.conf"
number_of_events = 50000
```

Listing 2: Example of a main configuration file in Allpix-Squared.

A full list detailing all the modules used for a simulation and their parameters can be found in the Allpix-Squared Manual [18]. A minimal set of modules necessary to run a simulation contains:

- Main module – sets global parameters and points to geometry configuration file.
- GeometryBuilderGeant4 – constructs specified experimental geometry.
- DepositionGeant4 – deposits energy in a detector volume and creates charge carriers.
- ElectricFieldReader – applies an electric field to a detector volume.
- GenericPropagation – propagates deposited charge carriers in a detector volume.
- SimpleTransfer – prepares sets of propagated charges for processing.
- DefaultDigitizer – translates the collected charge into a digitized signal.
- ROOTObjectWriter – stores detailed simulation results into a ROOT file.

3.2.2 Geometry configuration

The placement of individual detectors used in a simulation is described in the geometry configuration file. Every detector is described by a block of code with a unique name in the block's header. Detectors are in fact represented by individual modules, whose parameters describe their position and orientation in the experiment and also specify the type of a module. Users can either use one of the available models of widely used detectors (such as TimePix) or create a custom model. An example of a geometry configuration file of a single TimePix detector is shown in Listing 3. A visualization of an EUDET-type telescope consisting of six Mimosa detectors and a TimePix DUT is shown in Fig. 15.

```
[dut]
type = "timepix"
position = 0 0 0
orientation = 0 90deg 0
```

Listing 3: Example of a geometry configuration file with one detector named "dut."

Allpix-Squared allows the simulation of the full test beam setup – a telescope consisting of a number of pixel detectors and the tested module. Output of the simulation is then stored in a format compatible with specialized frameworks for track reconstruction and analysis – EUTelescope [19] or Corryvreckan [20]. It is also possible to simulate only the DUT, the simulation output is then stored in ROOT files, which is a standard format in particle physics experiments.

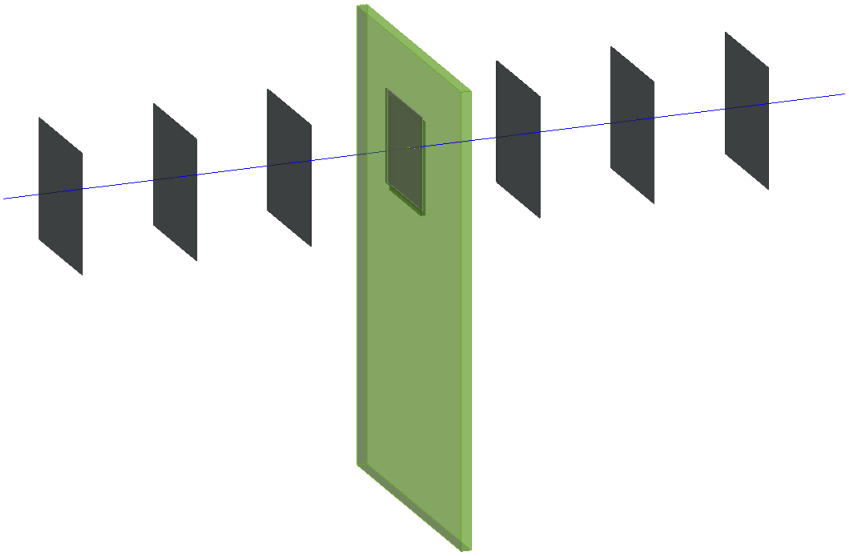

Figure 15: Example of experimental geometry using six mimosa detectors and one timepix detector. This is a typical experimental setup for test beam experiments. Blue line represents a passing particle, sensitive detector volumes are illustrated in black, non-sensitive volumes in green.

3.2.3 Model configuration

In order to use a detector in a simulation, an appropriate model of such a detector has to be created. Model configuration file contains information about dimensions and segmentation of a silicon sensor. It is also possible to increase the accuracy of a model by adding layers, which can simulate kapton tapes or printed-circuit boards which are present on real strip detectors. An example definition of a strip detector model is shown in Listing. 4, a visualization of this particular model is in Fig. 16.

```
type = "monolithic"
number_of_pixels = 1280 1
```

```
pixel_size = 74.5um 10cm
sensor_thickness = 310um
sensor_excess = 10um

[support]
thickness = 1um
size = 5cm 5cm
location = "absolute"
offset = 0 0 -5cm
material = "kapton"

[support]
thickness = 1um
size = 5cm 5cm
location = "absolute"
offset = 0 0 5cm
material = "kapton"
```

Listing 4: Code of a custom model configuration file.

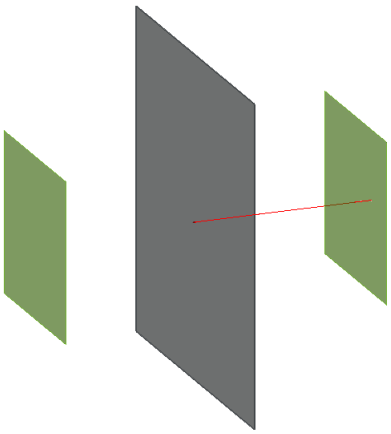

Figure 16: Example of a strip detector model consisting of a main sensor area (grey) and two kapton support layers (green). The incident particle is displayed in red.

4 Allpix-Squared Module Simulation

The purpose of the simulation is to perform a module characterization, that is to obtain two primary characteristics of a simulated module – an efficiency curve and an average cluster size plot – and to explore the influence of various configuration parameters on these characteristics. The obtained results are compared to the data from test beam measurements to ascertain the agreement of a real module and its simulated model. To ensure the best agreement, adjustments of the simulation parameters are carried out and additional effects are implemented.

A prototype long-strip barrel module based on ATLAS17LS sensor was characterized with test beam measurements carried out at DESY (Deutsches Elektronen-Synchrotron) in Hamburg, Germany in April 2019. The local accelerator is capable of supplying a beam of electrons in an energy range of around 4–6 GeV, the measurements were performed using 5.8 GeV electrons.

4.1 Simulation Setup

A simple silicon strip detector model is used, its parameters are identical to the ATLAS17LS sensor. The sensor is divided into 1280 strips with a strip pitch of 75.5 μm and a thickness of 300 μm. Because the actual thickness of the sensitive sensor volume is not known with absolute precision, different thicknesses of the sensor are also simulated.

The simulation is performed using a simplified setup without an EUDET-type telescope. In order to ensure the reproducibility of the simulation, the main configuration parameters of the simulation are listed.

A total of 50 000 events are simulated for every configuration, in every event only one particle is emitted. Specifically, electrons with an energy of 5.8 GeV are used, forming a Gaussian beam with a standard deviation $\sigma = 1$ mm.

Physics list (a list of physics processes in the simulation) used is FTFP_BERT_LIV, which is a default physics list recommended by the developers of Allpix-Squared for test beam measurement simulations. However in search of the best agreement between the simulation and the test beam results, other physics lists are also explored in Section 4.6.

During the test beam measurement at DESY, various incident angles of the particle beam were measured. This was achieved by rotating the central box containing the DUT. In the simulations, the same angles of incidence are set and simulated. The results are discussed in Section 4.14.

The parameter "max_step_length" is set to 1 μm. It defines the maximum distance a particle can travel before its properties are updated. The effect of varying the value of the "max_step_length" parameter is explored in Section 4.7.

A linear electric field was applied to the sensor model, the full depletion voltage, given by the parameters of the real sensor, is set to -300 V, while bias voltage is set to -400 V to correspond to the test beam measurements. An alternative electric field model generated

in TCAD [21] is also simulated, the results are discussed in Section 4.12.

The parameter "charge_per_step" is set to 50. It defines how many charge carriers are to be simultaneously propagated during one step. The influence of this parameter is discussed in Section 4.8. Temperature, which also affects charge propagation, was set to 290 K (approximately 17 °C).

During the digitization of the charge collected on every strip, Allpix-Squared allows to set the standard deviation of the Gaussian electronics noise, which is added to the input charge. In the simulations it is set to 0.138 fC (864 e) which was the average noise value established during test beam measurements of the real sensor.

The output of the simulation is configured to be stored in a single ROOT file. In order to decrease the size of an output file, only the PixelCharge object, which contains information about the deposited charge in every individual strip, is stored.

4.2 Simulation Run

All the simulations were run on the LXPLUS using Allpix-Squared version 1.3.2. An example of the simulation output is shown in Listing 5.

```
|09:22:46.052|  (STATUS) Welcome to Allpix^2 v1.3+270^g6cc43a8b
|09:22:46.053|  (STATUS) Initialized PRNG with system entropy seed
    410571489608774157
|09:22:46.531|  (STATUS) Loaded 8 modules
|09:22:46.679|  (STATUS) Initialized 8 module instantiations
|09:43:43.296|  (STATUS) Finished run of 50000 events
|09:43:44.013|  (STATUS) [F:ROOTObjectWriter] Wrote 65772 objects to 1 branches
    in file: /afs/cern.ch/user/r/rprivara/tb/output/output.root
|09:43:44.015|  (STATUS) Finalization completed
|09:43:44.015|  (STATUS) Executed 8 instantiations in 20 minutes 57 seconds,
    spending 80% of time in slowest instantiation GenericPropagation:dut
|09:43:44.015|  (STATUS) Average processing time is 25 ms/event, event
    generation at 40 Hz
```

Listing 5: Example of an Allpix-Squared simulation output.

A simulation of 50 000 events took approximately 21 minutes with the main output file about 60 MB large.

A Python3 script was created to allow easier rerunning of the simulation with different parameters. It modifies the Allpix-Squared configuration files according to user input, runs the simulation and saves the output with a file name which reflects the parameters.

4.3 Data Analysis

In order to obtain the efficiency and average cluster size plots, it is necessary to perform a threshold scan. This procedure was explained in Section 1.4. It involves systematically raising the charge threshold from 0 fC up to a certain point, usually around 7–8 fC, and observing how many strips have collected charge above the given threshold. If the charge

in a strip surpasses this threshold, it is accepted as a hit. By counting the number of accepted hits as a fraction of the total amount of events (particles sent) an efficiency for a current threshold is obtained. Furthermore, the number of strips with a hit gives information about an average cluster size for a given threshold.

There are two possible ways to perform a threshold scan in a simulation. One possibility of threshold scanning is to set the threshold in the main Allpix-Squared configuration file and rerun the simulation for every threshold value. For a typical threshold scan from 0 to 8 fC with a step of 0.1 fC this would mean running the simulation 81 times and would take over 28 hours (assuming 50 000 events in each run as in the simulation mentioned above). A more efficient and faster way is to run the simulation only once with the threshold set to zero in Allpix-Squared configuration and to apply the actual thresholds on the resulting output dataset.

A Python3 script was written to handle analysis of the simulation output. A user can first configure parameters of the threshold scan (the lowest and the highest threshold and a threshold step). The script then processes output ROOT files of the simulation and creates plots of efficiency and average cluster size.

For every threshold, it is done by iterating over a set of `PixelCharge` objects, which contain information about every strip with non-zero deposited charge during a single event. To ensure easier handling of the data and to allow for further application of additional effects, an array is created containing values of deposited charge in every strip during an event.

A number of strips with a charge above the current threshold is obtained, this is the cluster size for a given event and it is entered into a two-dimensional histogram, shown in Fig 17. A profile of this histogram then creates the average cluster size graph. Efficiency is easily calculated by counting the number of events with non-zero cluster size for a given threshold and then scaling down the resulting graph by a total number of events.

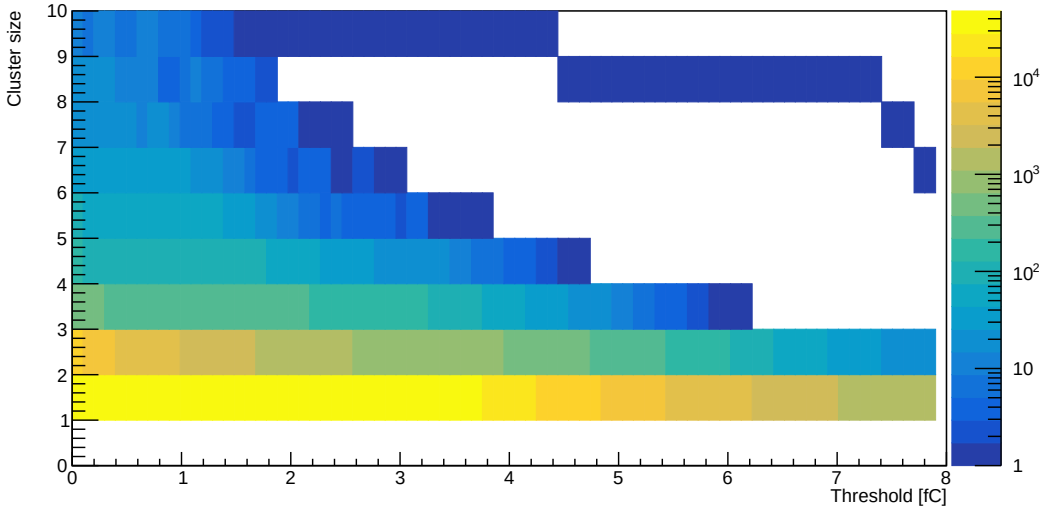

Figure 17: Two-dimensional histogram of cluster size as a function of a charge threshold.

The section of the code used to perform the threshold scanning is shown in Listing 6.

```
for aThr in np.arange(thrStart, thrEnd, thrStep)
    for anEvent in pixelChargeSet:
        for stripHit in anEvent:
                stripCharge[stripHit.getIndex().X()] = stripHit.getCharge()
        clusterSize = len(np.where(stripCharge > aThr)[0])
        if clusterSize > 0:
                effHist.Fill(aThr)
                clusHist.Fill(aThr, clusterSize)
effHist.Scale(1/nOfEvents)
```

Listing 6: The analysis code to compute efficiency and cluster size.

A separate Python3 script was written to provide a configurable way to plot and save these histograms. Additionaly, it fits the efficiency data points with a skewed complementary error function

$$f(x) = 0.5 \cdot E_0 \cdot \mathrm{erfc}\left(\frac{x - x_0}{\sqrt{2} \cdot \sigma} \cdot \left[1 - a \cdot \tanh\left(\frac{b \cdot (x - x_0)}{\sqrt{2} \cdot \sigma}\right)\right]\right), \tag{7}$$

where meaning of the fit parameters is:

- E_0 – maximum efficiency,
- x_0 – median charge (50 % efficiency),
- σ – width of the distribution,
- a, b – skew parameters.

4.4 Evaluating the test beam and simulation data agreement

To quantitatively judge the level of agreement between two sets of data, typically from a simulation and from test beam measurements, an analogy of χ^2 statistic is used, where the agreement is described by a number

$$\chi^2_{(\text{test beam - sim})} = \sum_i \frac{(y_{i,\text{tb}} - y_{i,\text{sim}})^2}{\sigma^2_{i,\text{tb}} + \sigma^2_{i,\text{sim}}}, \tag{8}$$

where i is the index of the test beam data points, $y_{i,\text{tb}}$ is the efficiency from the test beam data, $y_{i,\text{sim}}$ is the efficiency from the simulation data and $\sigma_{i,\text{tb.}}$ and $\sigma_{i,\text{sim}}$ are their corresponding errors.

The efficiency from the test beam data is simply the value of the i-th data point. However, since the threshold steps used in test beam measurements are independent from the threshold step in the simulation data analysis, the data points do not match and the efficiency from the simulation has to be either interpolated or obtained from the fit function. Usually the interpolation method is used because the fit function can sometimes fail to accurately describe the efficiency drop area, especially for non-perpendicular incidence angles, as discussed in Section 4.14.

4.5 Simulation results

The obtained results from an Allpix-Squared simulation for perpendicular incidence were compared to the test beam measurement data. An initial comparison shows that the efficiency curves (Fig. 18) are not in agreement at all – median charge, obtained from the x_0 parameter of the fit function (7), has a value of (3.65 ± 0.01) fC for the test beam data and (4.058 ± 0.002) fC for the simulation. This shows a very significant discrepancy of about 11 %.

Initial results regarding the average cluster size as a function of charge threshold show that the shapes of the curves (Fig. 19) from the test beam and simulation data are very similar and in good agreement for middle thresholds (2–4 fC). There is a noticeable offset at lower thresholds and also at the highest ones, but there the uncertainty from the test beam measurements is much higher, as indicated by error bars, due to a lower number of events.

As was stated at the beginning of this section, a primary goal is to reduce this disagreement between simulation and test beam data by exploring how individual parameters of the simulation affect the characteristics.

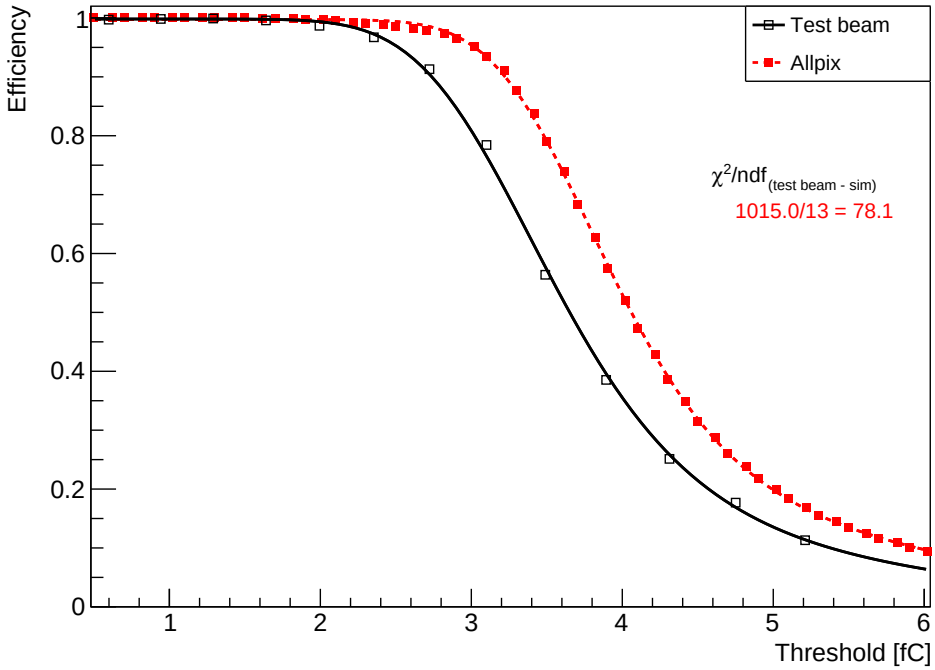

Figure 18: Efficiency curves from simulation and test beam data for perpendicular incidence.

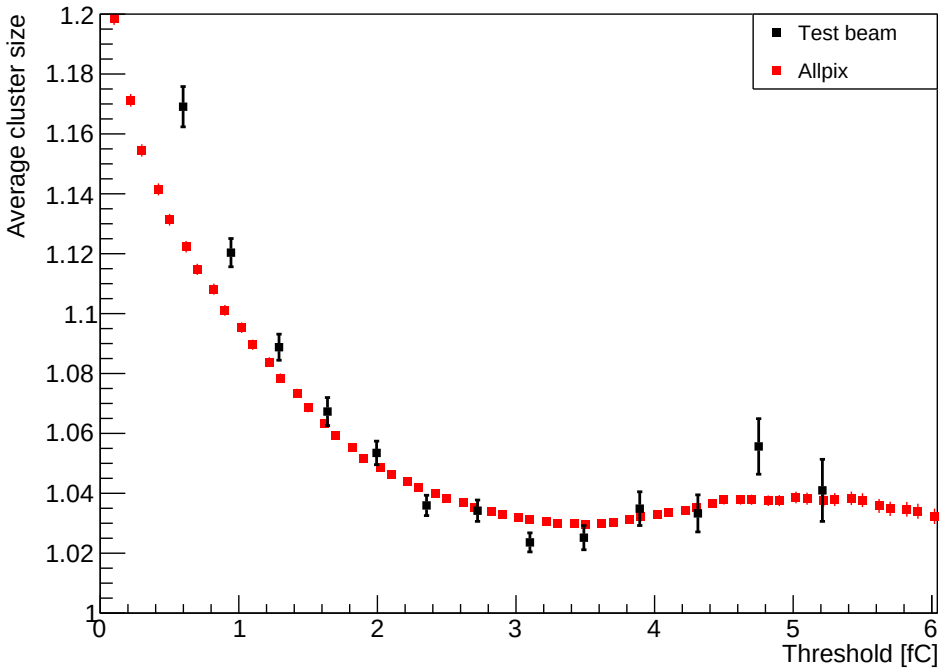

Figure 19: Average cluster size from test beam and simulation data for perpendicular incidence.

4.6 Physics lists

Physics processes allowed and used during a simulation are defined in a physics list. There is a number of precreated reference physics lists which can be used in usual circumstances and a creation of a custom list is also possible.

Every physics list consists of several components describing classes of processes, for example hadronic, electromagnetic and decay. These components include a number of processes which are built from cross section sets and interaction models.

Allpix-Squared developers recommend usage of the FTFP_BERT_LIV reference physics list. The FTFP_BERT physics list is generally recommended for collider physics applications [22] and the affix – in this case "_LIV" – indicates alternative electromagnetic physics models.

Simulations in Allpix-Squared were performed using two standard physics lists and a number of alternative electromagnetic physics models, including the recommended one:

- FTFP_BERT – a recommended physics list for collider physics,
- QGSP_BERT – a former default physics list, now mostly replaced by FTFP_BERT,
- FTFP_BERT_LIV – a default physics list in Allpix-Squared, alternative electromagnetic physics models based on Livermore set of models,
- FTFP_BERT_PEN – alternative electromagnetic physics models based on Penelope-2008 set of models,
- FTFP_BERT_EMV – a less precise, but faster set of electromagnetic physics models,
- FTFP_BERT_EMZ – the most accurate set of electromagnetic physics models selected from other packages.

From the obtained efficiency curves (Fig. 20) and average cluster size plots (Fig. 21) it is clear that the choice of a physics list or its electromagnetic models has a negligible influence on the outcome of the simulation. However this could change if an exotic physics list was used.

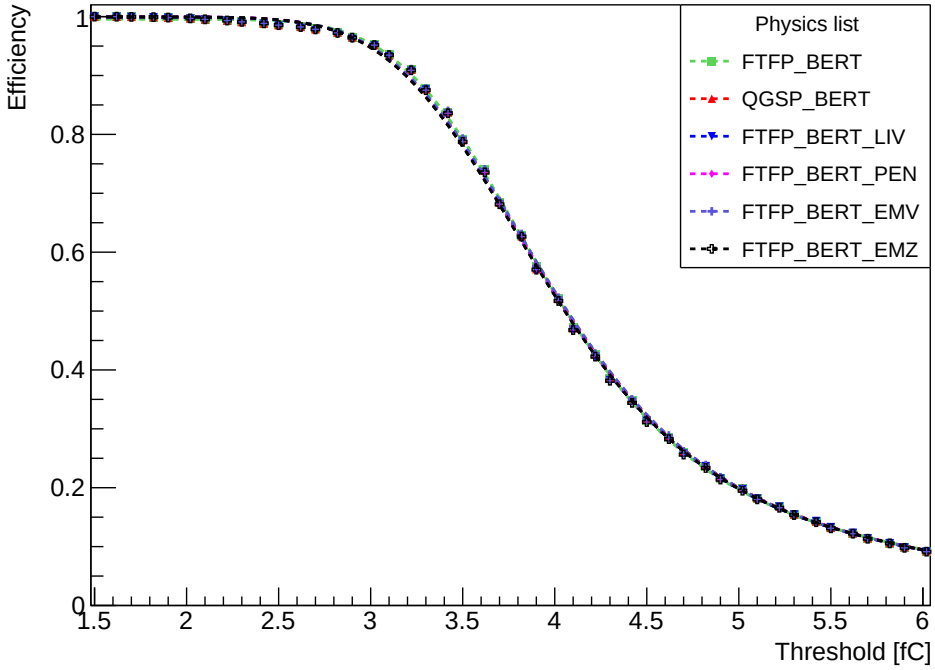

Figure 20: Efficiency curves obtained from Allpix-Squared using different physics lists.

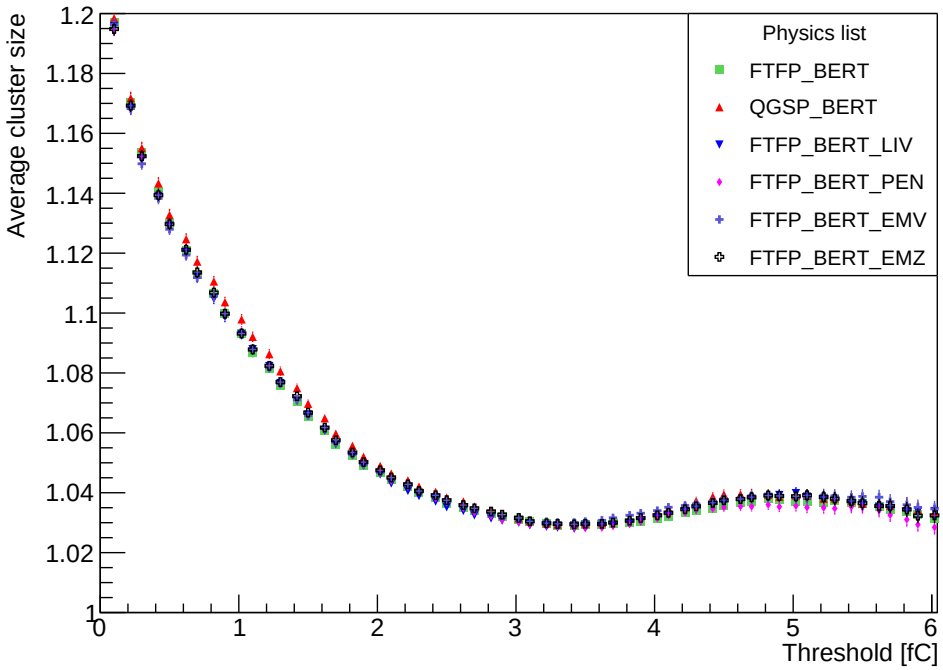

Figure 21: Average cluster size plots obtained from Allpix-Squared using different physics lists.

4.7 Step length parameter

The influence of the "max_step_length" parameter on the simulation outcome was tested. The Geant4 toolkit, which handles the propagation of particles in a volume, does not continuously check and modify particles' properties (e.g. particle's current energy due to ionization and other energy losses). Instead it only checks their status once a certain distance, called a step, was travelled from the previous check. The parameter "max_step_length" allows a user to configure the maximum length of a step. A shorter step distance should mean a more accurate simulation of particle behavior at the cost of longer computational time. The default value of the parameter is 1 μm, simulations were also performed using values of 0.1 μm and 5 μm.

While there is a very slight difference visible in the efficiency curves (Fig. 22) and at higher thresholds in the average cluster size plots (Fig. 23), it is certainly not significant enough to believe changing this parameter could improve the agreement between the simulation and the test beam data.

The effect of this parameter on computational and simulation time was insignificant, every simulation took approximately 21 minutes, with the differences being in the order of seconds.

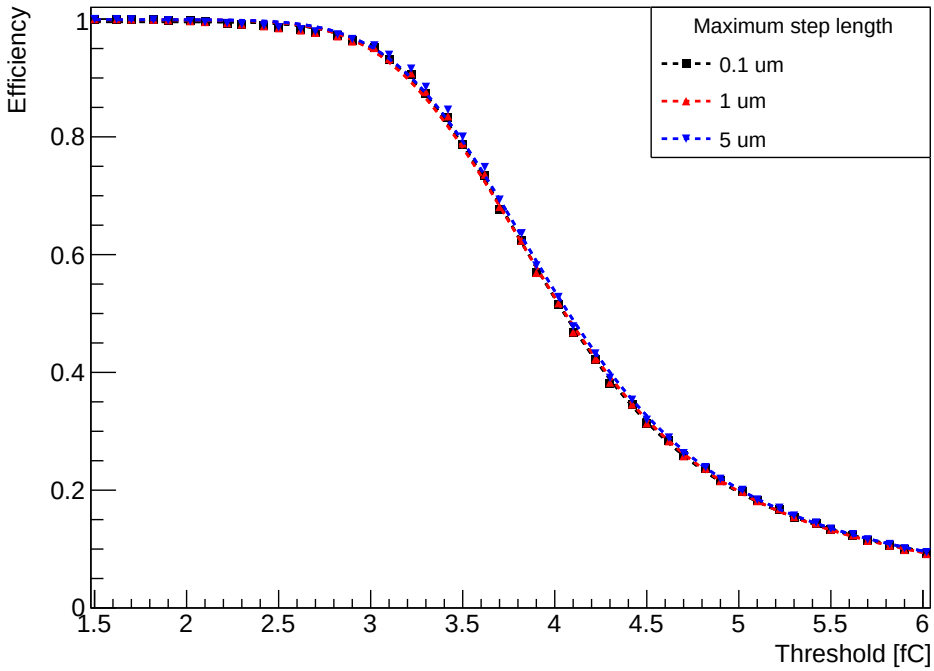

Figure 22: Efficiency curves obtained from Allpix-Squared using different values of the max_step_length parameter.

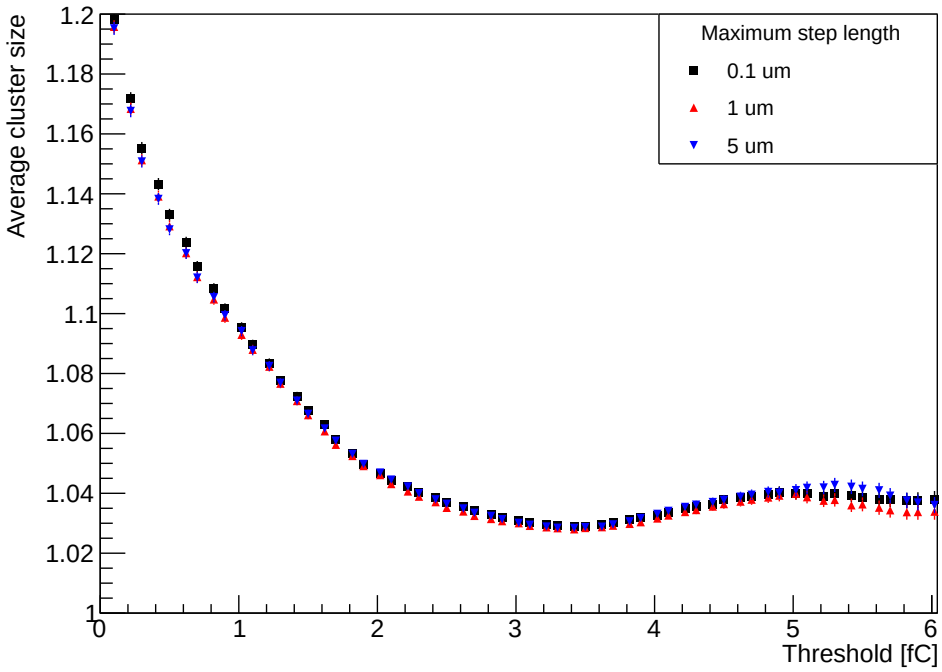

Figure 23: Average cluster size plots obtained from Allpix-Squared using different values of the max_step_length parameter.

4.8 Charge grouping parameter

A parameter called "charge_per_step" modifies the behavior of the simulation during its charge propagation stage. When charge carriers are propagated through the sensor volume, they can be grouped together and propagated simultaneously as a single charge carrier. If the value of the "charge_per_step" is set to one, this grouping is disabled and every single charge carrier is propagated individually. A higher value of the parameter results in a faster, albeit less accurate simulation. Simulations were performed using "charge_per_step" parameter values of 10, 50 and 100.

The effect of the "charge_per_step" parameter value on the efficiency curves (Fig. 24) and the average cluster size plots (Fig. 25) is negligible. However the effect on computational time is notable – with the parameter set to 100 the simulation took around 14 minutes, for a value of 50 it took around 21 minutes and when set to 10, it took 86 minutes. It also heavily affects the output file size, which were around 40, 64 and 267 MB for the parameter values of 100, 50 and 10 respectively. If either the file size or the computational time is a consideration, setting this parameter to a higher value can be helpful.

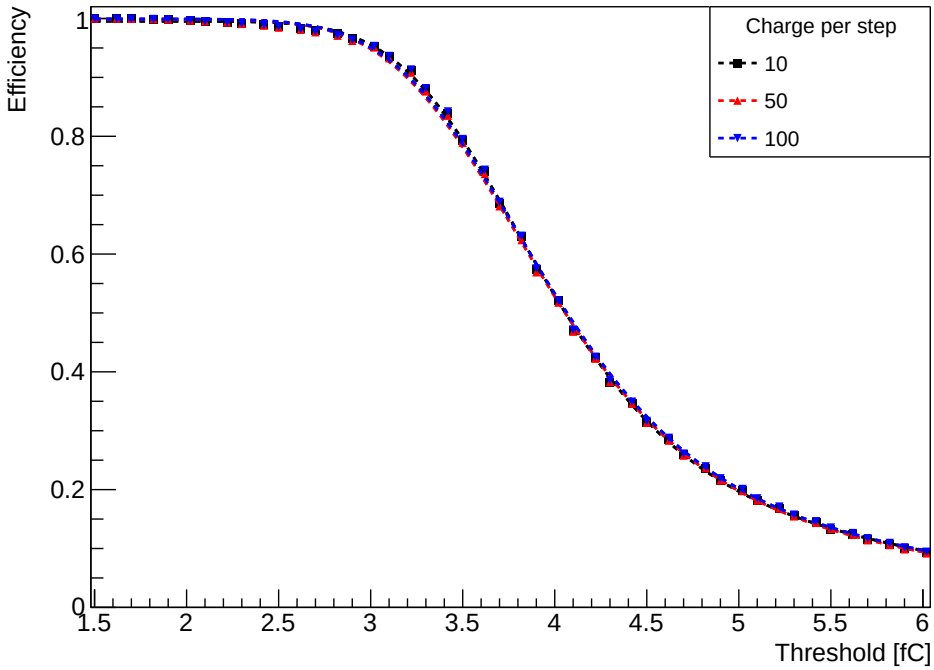

Figure 24: Efficiency curves obtained from Allpix-Squared using different values of the "charge_per_step" parameter.

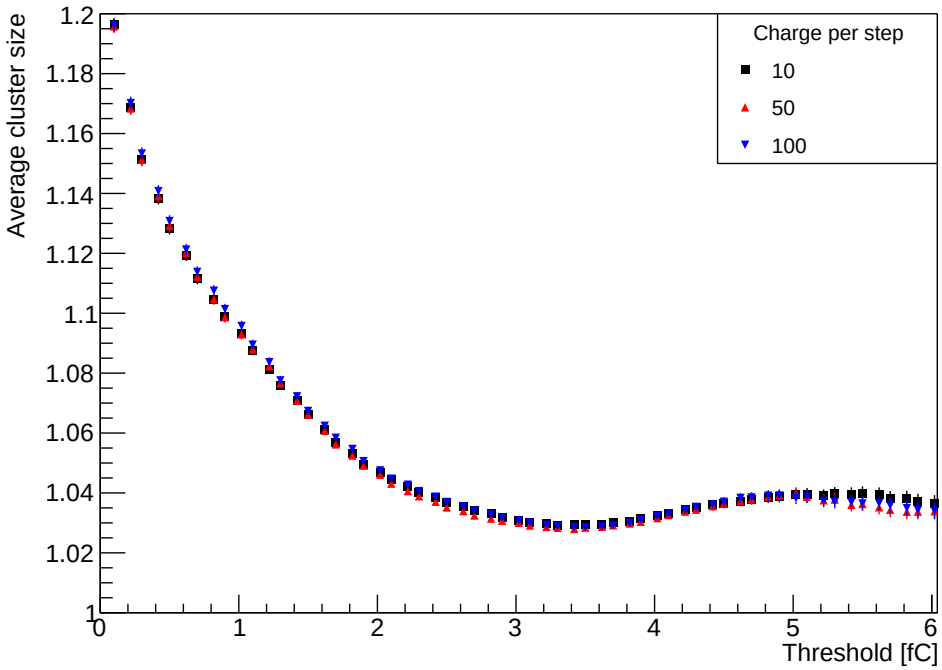

Figure 25: Average cluster size plots obtained from Allpix-Squared using different values of the "charge_per_step" parameter.

4.9 PAI model

Usage of the Photo-absorption ionisation (PAI) model during a simulation is recommended for simulation of the passage of relativistic charged particles through very thin absorbers [23].

To ascertain the actual influence of the PAI model a simulation was performed with

the model enabled and disabled. Small differences in the efficiency curves (Fig. 26) are noticeable at middle thresholds (3–4 fC), there is almost no difference in the average cluster size plots (Fig. 27).

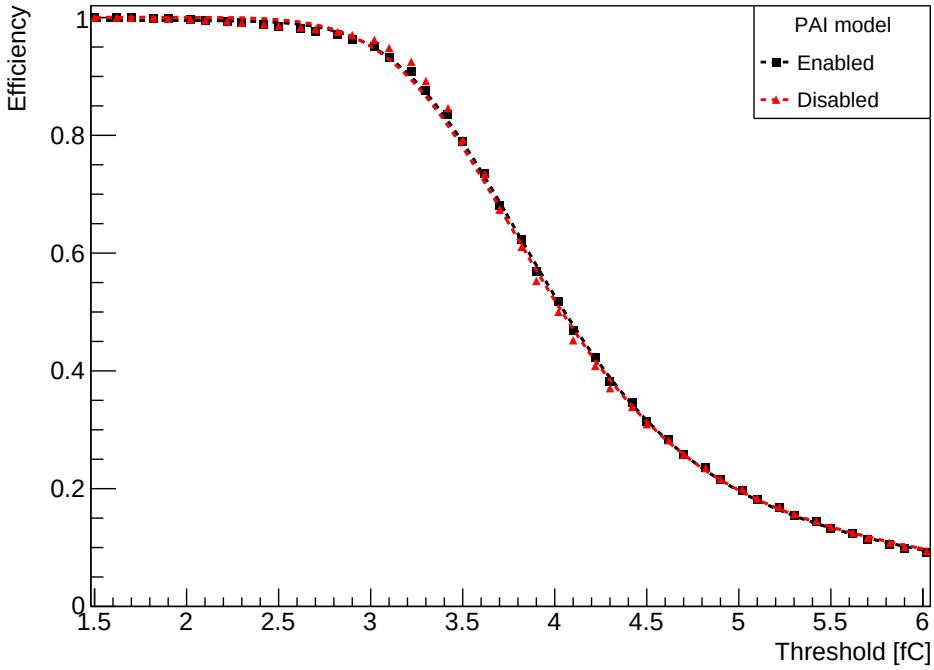

Figure 26: Efficiency curves obtained from Allpix-Squared with PAI model enabled and disabled.

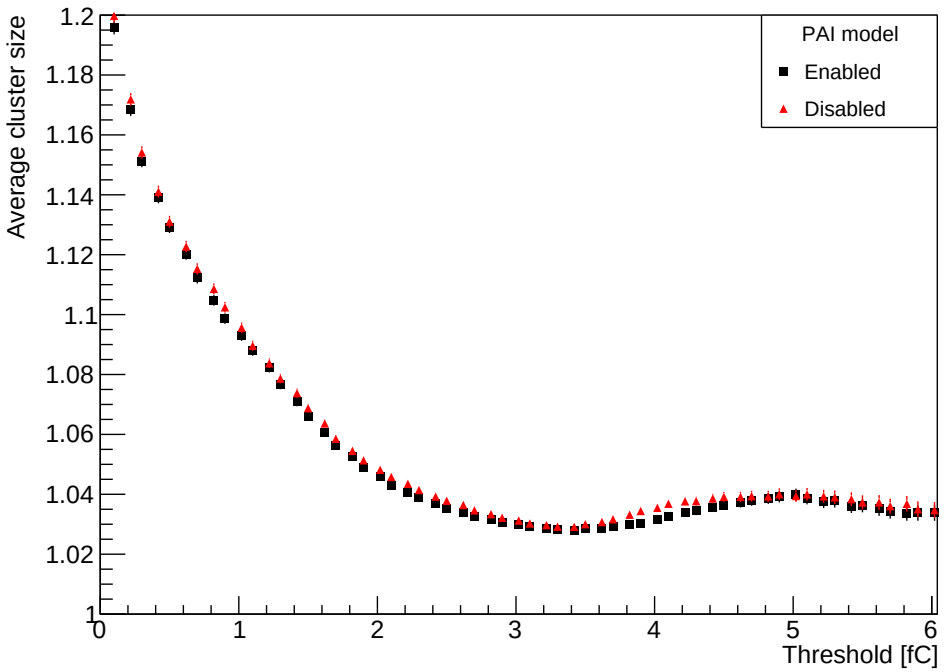

Figure 27: Average cluster size plots obtained from Allpix-Squared with PAI model enabled and disabled.

4.10 Cross talk simulation

A simple model of a first-order cross talk effect [8] was implemented. This effect has its origin in the existence of capacitive couplings between a given strip and its adjacent strips and a strip and the backplane. An equivalent diagram of three strips and the capacitive couplings is shown in Fig. 28. These couplings cause leakage of charge collected on a strip to its adjacent strips and to the backplane. In the ideal case there would only be a capacitive coupling between a strip and its Front-End electronics, ensuring read-out without any cross talk effect.

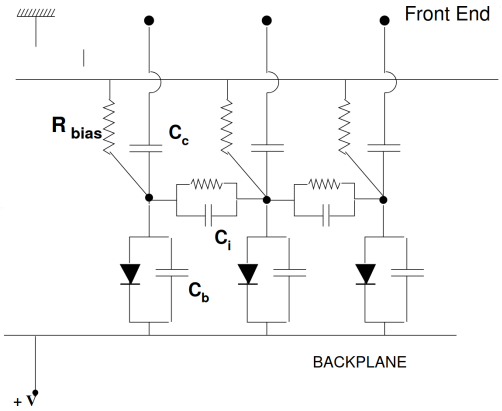

Figure 28: An equivalent diagram of three strips with their capacitive couplings to the Front-End electronics, the adjacent strips and the backplane [8].

The important distinction here is that the charge leaking from a strip to its neighbors is still retained in the sensor and is read out as a signal, albeit from a different strip, while the charge leaking to the backplane is for all intents and purposes lost.

In this model, the amount of charge leaking to other parts of the module is based solely on the capacitive couplings. The combined fraction of charge leaking to both the adjacent strips of a given principal strip, which is governed by the value of interstrip capacitance C_i, would be calculated as [8]

$$Q_{\mathrm{StS}} = \frac{C_{\mathrm{i}}}{C_{\mathrm{i}} + C_{\mathrm{bph}} + C_{\mathrm{c}}}, \tag{9}$$

where C_{b} is the capacitance to the backplane and C_{c} is the capacitance between a strip and its front-end electronics. Using the formula (9), the fraction of charge leaking to a given part of the module can be calculated by inputting the corresponding capacitance into the numerator, the denominator is always the sum of all the capacitances. The sum of all the fractions of charges leaking and the fraction of charge flowing to the Front-End electronics is 1 and therefore the conservation of charge is observed.

Based on capacitance values

- $C_{\mathrm{c}} = 25$ pF/cm [24],
- $C_{\mathrm{b}} = 0.25$ pF/cm,
- $C_{\mathrm{i}} = 0.8$ pF/cm [25],

the fractions of charge leaking to the adjacent strips and to the backplane are

$$Q_{\mathrm{StS}} = \frac{C_{\mathrm{i}}}{C_{\mathrm{i}} + C_{\mathrm{b}} + C_{\mathrm{c}}} = \frac{0.8}{0.8 + 0.25 + 25} \doteq 3.07\ \%, \tag{10}$$

$$Q_{\mathrm{StB}} = \frac{C_{\mathrm{b}}}{C_{\mathrm{i}} + C_{\mathrm{b}} + C_{\mathrm{c}}} = \frac{0.25}{0.8 + 0.25 + 25} \doteq 0.96\ \%. \tag{11}$$

The backplane capacitance value is calculated as the bulk capacitance of the sensor, divided by the number of strips and by the strip length. The bulk capacitance C_{bulk} is not measured directly but rather as $1/C_{\mathrm{bulk}}^2$, the value obtained by measurements is $1/C_{\mathrm{bulk}}^2 = 0.095\ \mathrm{nF}^{-2}$ measured at bias voltage of -400 V [25]. The backplane capacitance therefore is

$$C_{\mathrm{b}} = \frac{\sqrt{\frac{1}{0.095\ \mathrm{nF}^{-2}}}}{1280 \cdot 10\ \mathrm{cm}} \doteq 0.25\ \mathrm{pF/cm}. \tag{12}$$

The cross talk effect is implemented in the Python3 script which handles analysis of the raw .ROOT output file, not in the Allpix-Squared simulation itself. Compared to implementing it in Allpix-Squared this is advantageous as there is no need to rerun the simulation when parameters of the cross talk effect are modified. Rerunning the analysis script over a simulation output file is much faster than rerunning the entire Allpix-Squared simulation.

First an array is created containing indices of strips with non-zero collected charge. When iterating over these strips, the amounts of charge leaking from a principal strip to adjacent strips (`charge_StoS`) and to the backplane (`charge_StB`) are calculated using the charge leakage fractions (10) and (11). These charges are then subtracted from the principal strip and the charges in the adjacent strips are increased by the `charge_StoS` value.

```
chargeMask = np.nonzero(stripCharge)
for stripIndex in chargeMask:
    charge_StoS = stripCharge[stripIndex] * Q_StS / 2
    charge_StoB = stripCharge[stripIndex] * Q_StB

    stripChargeAdj[stripIndex] -= 2 * charge_StoS + charge_StoB
    stripChargeAdj[stripIndex-1] += charge_StoS
    stripChargeAdj[stripIndex+1] += charge_StoS
```

Listing 7: Implementation of a cross talk effect.

Results obtained with the cross talk effect enabled show a major improvement of agreement between the test beam and simulation efficiency (Fig. 29). The average cluster plots (Fig. 30) show a slight improvement at very low thresholds and also show differences at middle and high thresholds (3–5 fC), but due to large errors in the test beam data it is difficult to ascertain whether this represents an improvement.

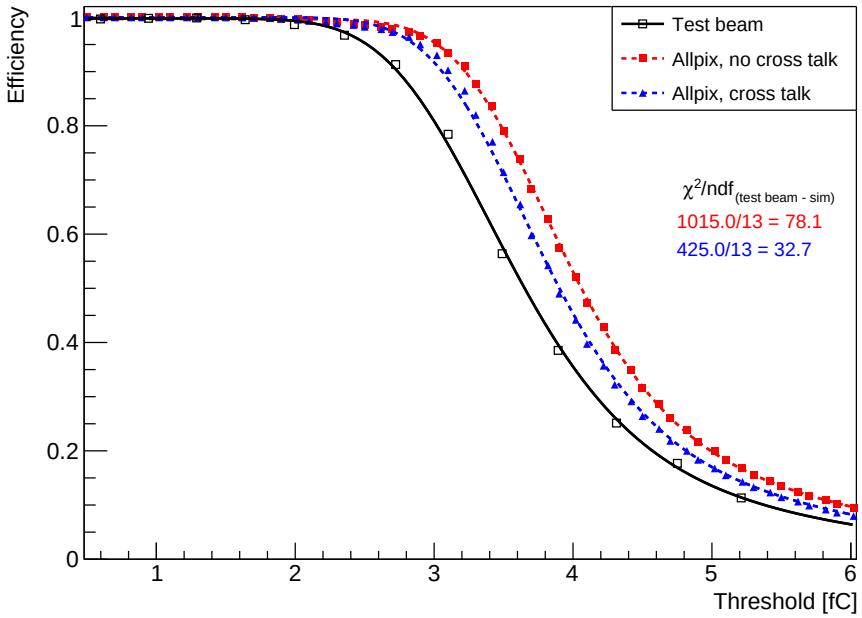

Figure 29: Efficiency curves obtained from the test beam data and from Allpix-Squared simulation with the cross talk effect enabled and disabled.

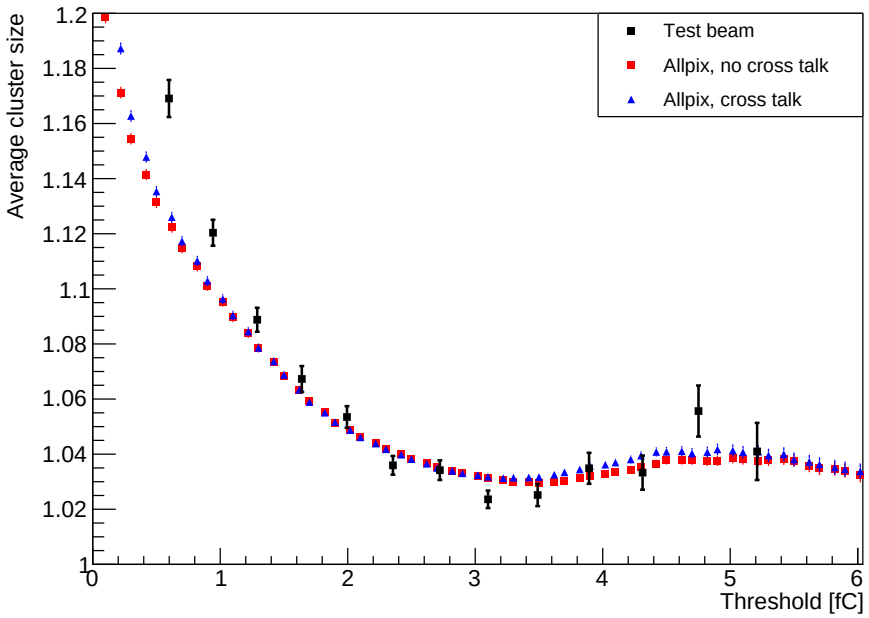

Figure 30: Average cluster size plots obtained from test beam data and from Allpix-Squared simulation with the cross talk effect enabled and disabled.

4.11 Sensor thickness

The target sensor thickness is 300–320 μm with the active volume of more than 90 % [6]. This means that the active volume can realistically be 270–320 μm. The sensor model thickness defined in an Allpix-Squared model configuration file describes the active volume.

Sensor thicknesses of 270, 280, 290, 300 and 310 μm were simulated. The cross talk effect discussed in a previous subsection was applied during the analysis of the output files.

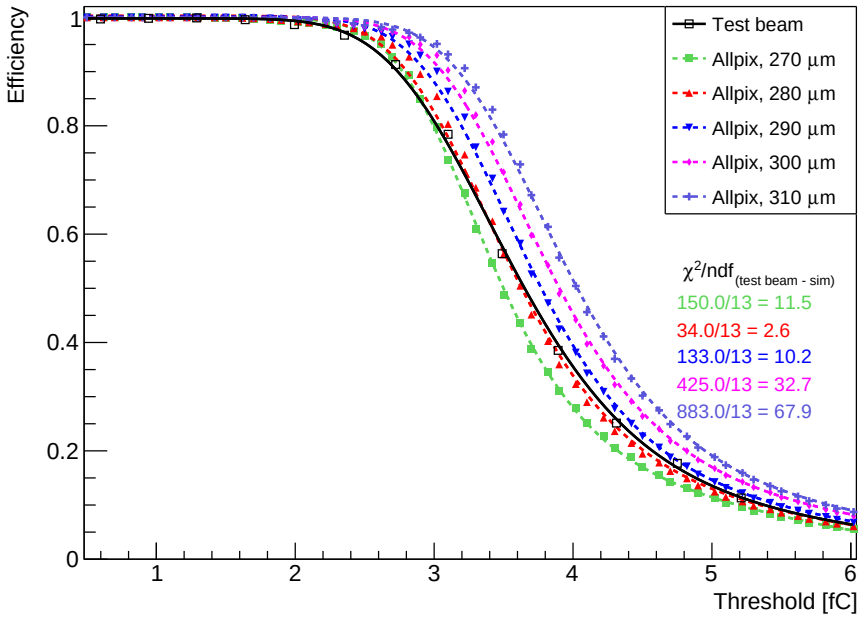

Figure 31: Efficiency curves obtained from the test beam data and from Allpix-Squared simulation for different sensor thicknesses.

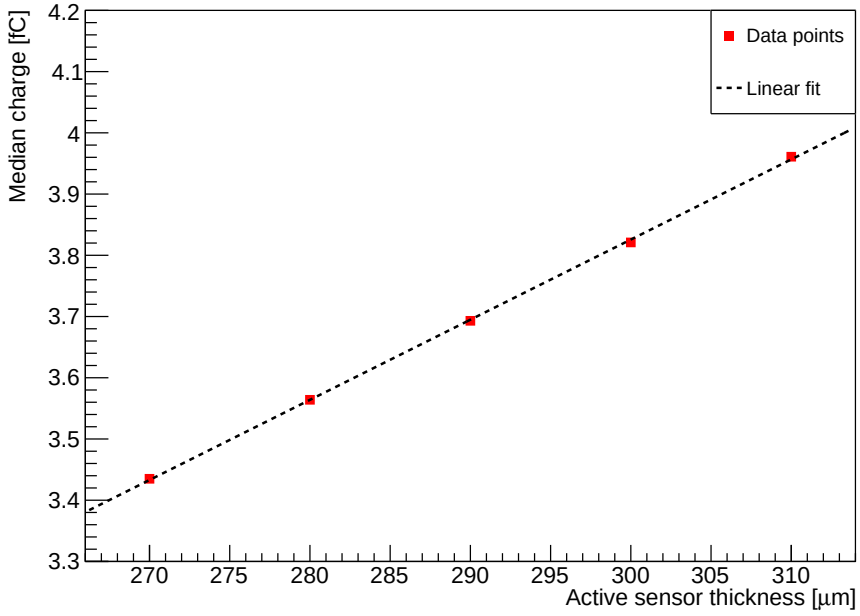

Figure 32: Scaling of the median charge value with the active sensor thickness.

A choice of the active sensor thickness heavily influences the efficiency curve (Fig. 31). An increased sensor thickness causes particles to travel longer in a sensor volume, therefore depositing more charge. This way, more events are classified as hits and the efficiency starts dropping at higher thresholds than in case of a thinner sensor. In fact the median charge value shows linear scaling with the active sensor thickness (Fig. 32). Based on the χ^2 metric used to evaluate data agreement, the active thickness of 280 μm matches the test beam data the best.

The average cluster size (Fig. 33) is in general higher for thicker sensors, again due to a higher collected charge per passing particle leading to more charge being leaked to the

adjacent strips.

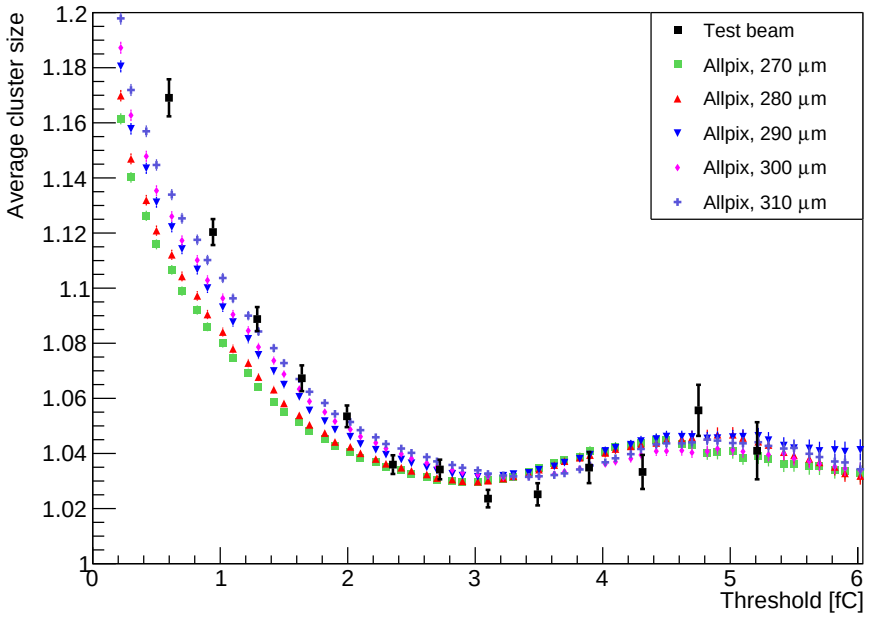

Figure 33: Average cluster size plots obtained from the test beam data and from Allpix-Squared simulation for different sensor thicknesses.

4.12 Electric field models

For studying the electric field model influence the thickness was set to 280 μm based on the findings discussed in Section 4.11 and the cross talk effect is enabled. The resulting efficiency curves (Fig. 34) show only a slight difference between the simulations using the linear and the TCAD electric field models. The average cluster size plots (Fig. 35) show higher cluster sizes at lower thresholds and better agreement with the test beam data.

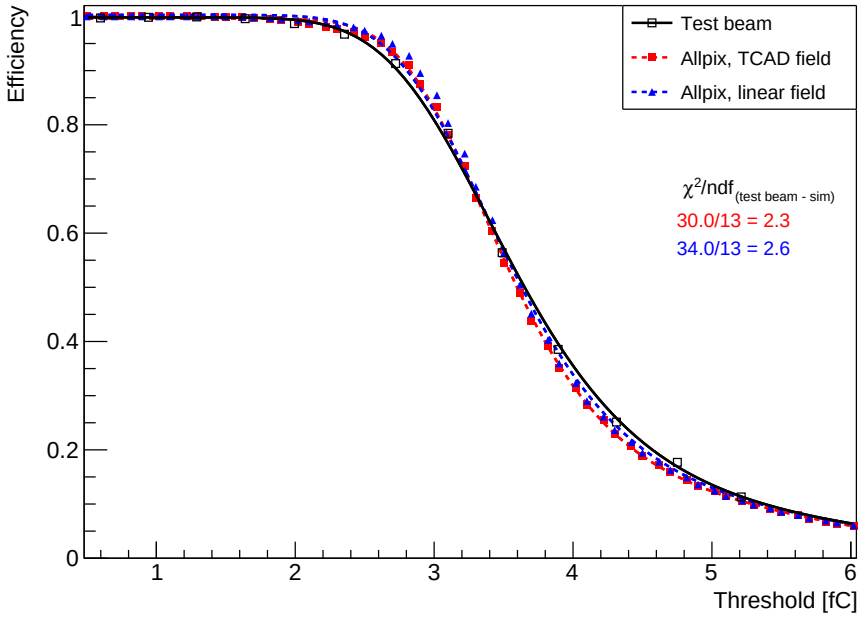

Figure 34: Efficiency curves obtained from Allpix-Squared simulations with different electric field models.

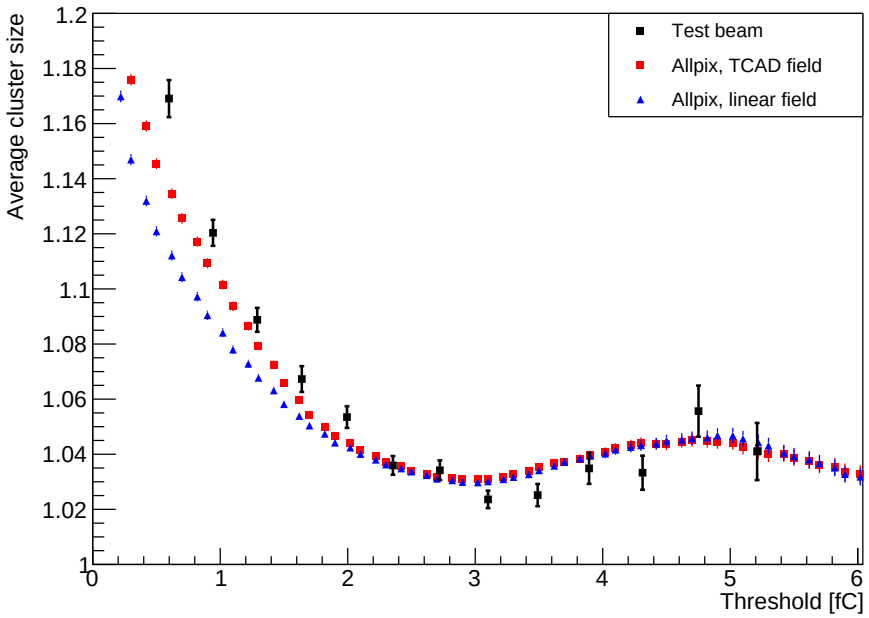

Figure 35: Average cluster size plots obtained from Allpix-Squared simulations with different electric field models.

While the TCAD-generated electric field model improves the accuracy of the simulation, especially regarding the average cluster size, the effect on computational time is noteworthy. The simulation using a TCAD field model took approximately four times longer than when using the linear electric field model.

4.13 Final simulation results

The results discussed in previous sections show that of all the explored simulation parameters or effects, three had a significant influence – the electric field model, the sensor thickness and the implementation of the cross talk effect. A final simulation was performed using Allpix-Squared with the TCAD electric field model, the sensor thickness of 280 μm and the cross talk effect enabled.

The comparison of the initial simulation results shown in Section 4.5 with the final simulation with tuned parameters shows that a major improvement was achieved in regarding the agreement with the test beam data. The efficiency curve is in a very good agreement with the test beam results (Fig. 36). The average cluster size agreement improved mostly at low thresholds, although the simulation results still don't match the test beam results perfectly (Fig. 37). At higher thresholds the agreement is more difficult to assess because of large errors from the test beam measurements.

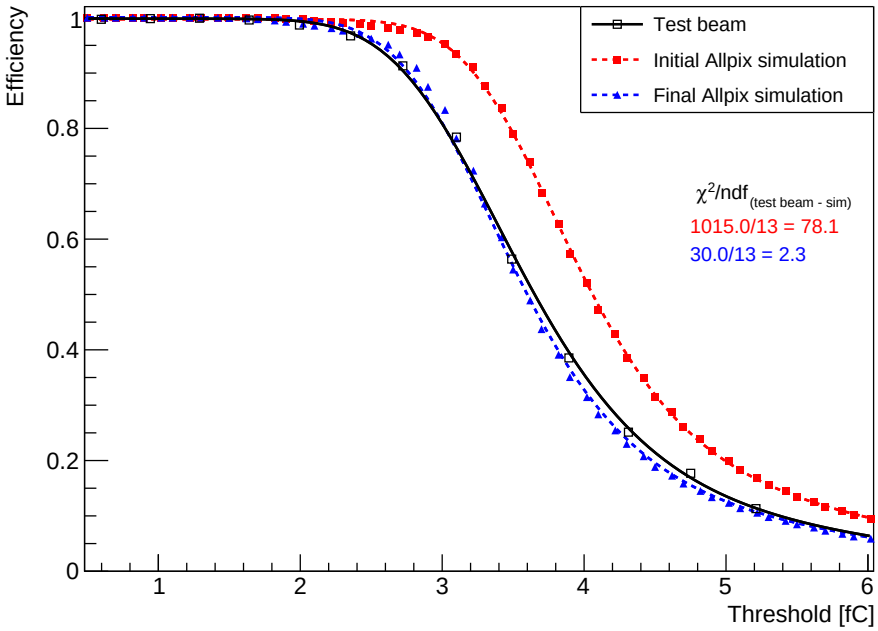

Figure 36: Comparison of the efficiency curves obtained from the initial and final Allpix-Squared simulations.

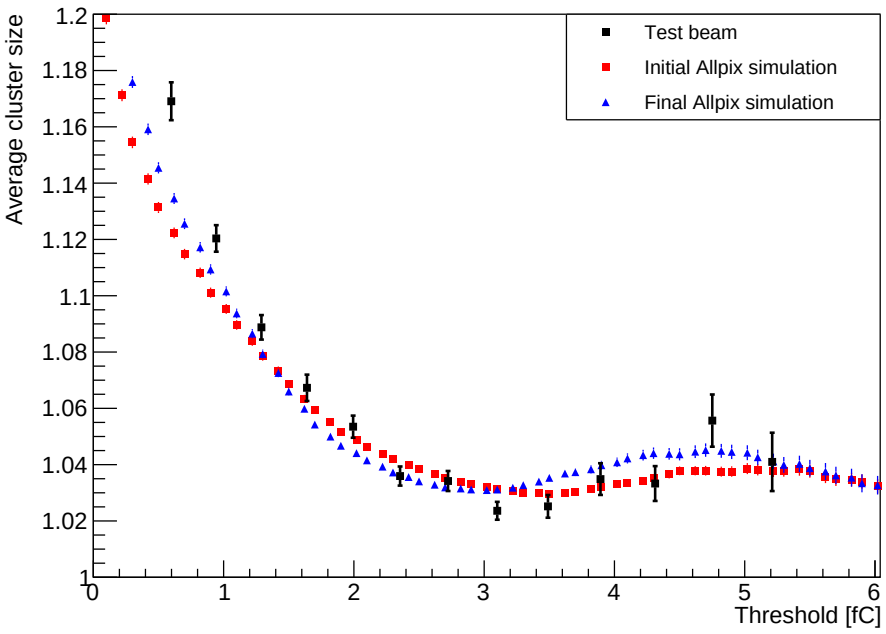

Figure 37: Comparison of the average cluster size obtained from the initial and final Allpix-Squared simulations.

4.14 Variable incident angles

During the test beam measurements of the module prototype at DESY several incident angles were measured in two possible rotation orientations, called a forward-angle and a side-angle rotation. Orientation of the axes for perpendicular incidence is illustrated in Fig. 38, the beam axis usually coincides with the z-axis and the module can be either rotated around the y-axis – a side-angle rotation – or around the x-axis – a forward-angle rotation.

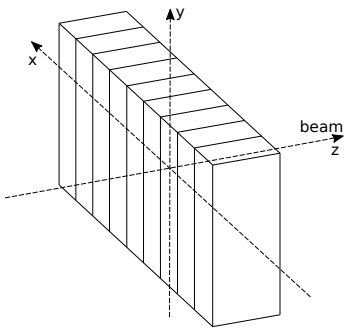

Figure 38: Axes orientation for perpendicular incidence during a test beam measurement.

Apart from the standard perpendicular incidence, two side-angle rotations of 5° and 12° and one forward-angle rotation of 23° were measured during the test beam measurements.

Simulations in Allpix-Squared were performed with these rotation configurations, using the previously established best-fit sensor thickness value of 280 μm, the cross talk effect was applied and the TCAD-generated electric field was used. A Python3 script was written to automate the process of running multiple simulations with different parameters. It modifies the geometry configuration file based on user-provided incident angles, runs the simulation and then includes the incident angle in the name of the output ROOT file to make it discernible from the others.

Efficiency curves for the side-angle rotations (Fig. 39) show a clear trend – the higher the incident angle, the sooner the efficiency starts dropping. This trend is visible both in the simulation and the test beam results. This effect can be explained by considering that the incident particle passes through multiple strips and therefore deposits less charge in each strip (compared to perpendicular incidence). This holds despite the overall deposited charge increase caused by an effective sensor thickness increase.

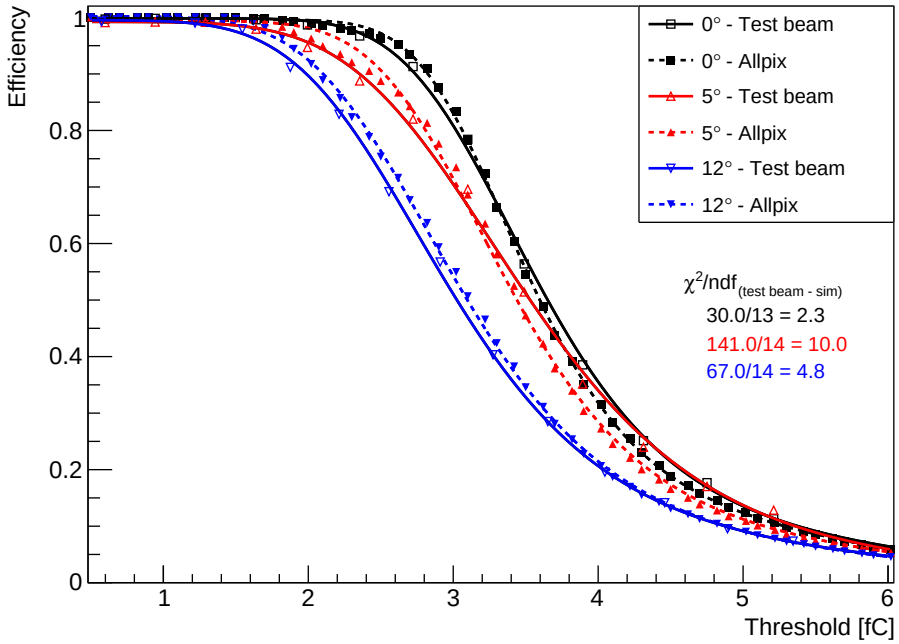

Figure 39: Efficiency curves obtained from the test beam data and from Allpix-Squared simulation for side-angle rotations.

The average cluster size changes dramatically when side-angle rotations are performed (Fig. 40). At low thresholds average cluster size is very high and then drops sharply. At thresholds higher than approximately 2.5 fC the curves are of similiar shape. Generally for higher incident angles the average cluster size is higher as well due to the incident particle itself passing through multiple strips and therefore causing more hits to be called.

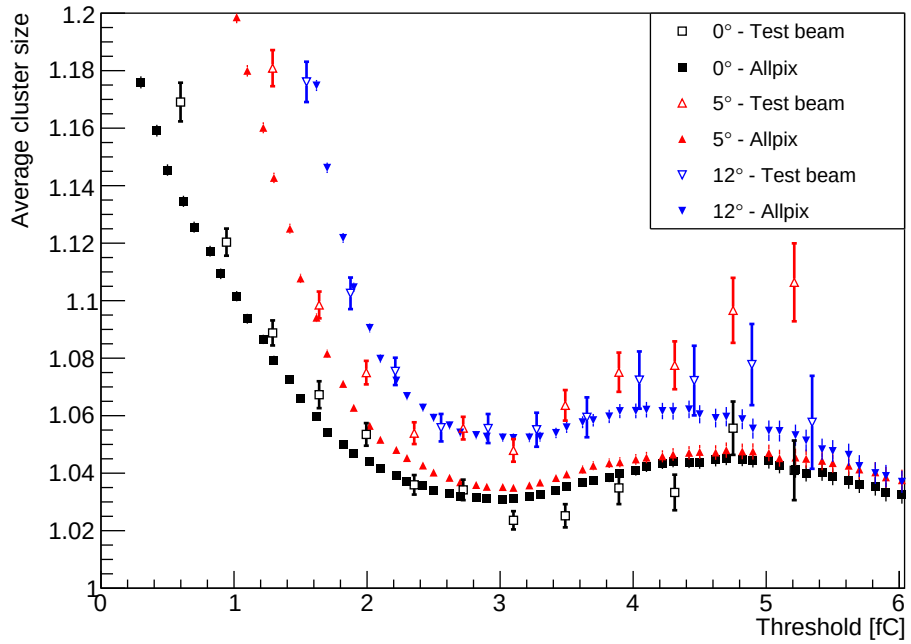

Figure 40: Average cluster size plots obtained from the test beam data and from Allpix-Squared simulation for side-angle rotations.

An Allpix-Squared simulation of the forward-angle rotation shows that the efficiency starts dropping much later than in the case of perpendicular incidence (Fig. 41). Similar to the side-angle rotation there is an increase of the effective sensor thickness, however since the incident particle passes through only one strip, the charge doesn't split among multiple strips. This trend can be also seen in the test beam data, however the agreement with the simulation results is very poor.

The test beam data show a major increase of the average cluster size at lower thresholds when forward-angle rotation is performed (Fig. 42). However this change is not reflected in the simulation results.

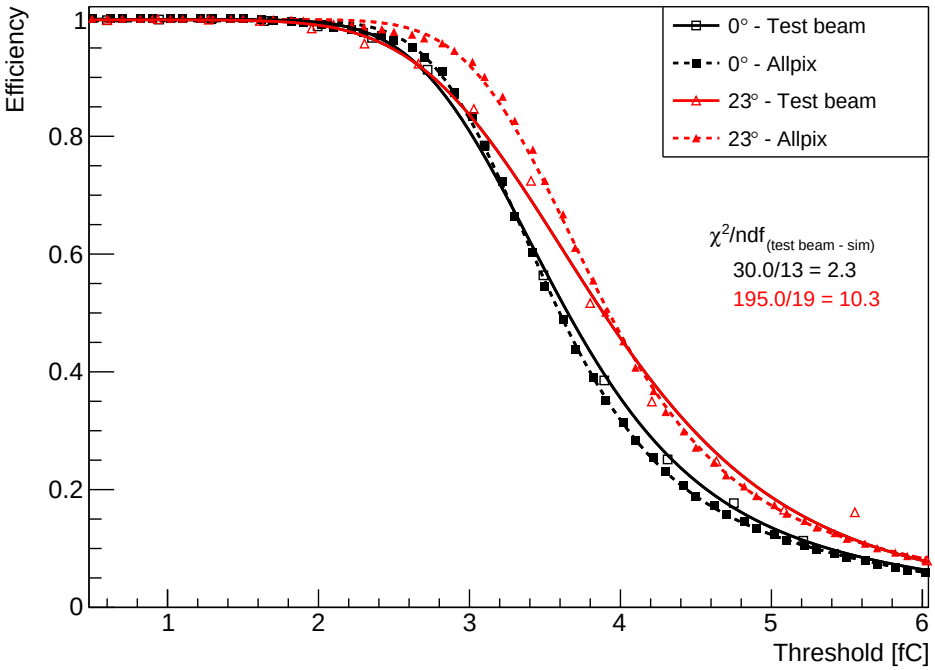

Figure 41: Efficiency curves obtained from the test beam data and from Allpix-Squared simulation for forward-angle rotations.

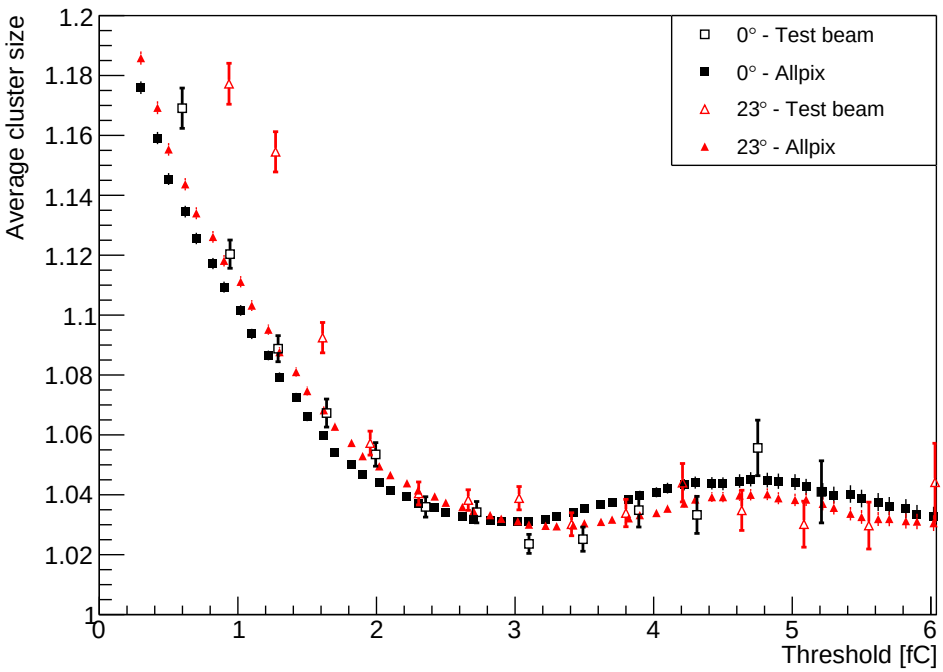

Figure 42: Average cluster size plots obtained from the test beam data and from Allpix-Squared simulation for forward-angle rotations.

5 Athena Framework

Athena represents the ATLAS software framework that manages almost every step of the ATLAS production workflow: event generation, simulation of interactions, reconstruction of tracks, digitization of hits and analysis. It is a control framework that represents an implementation of an underlying architecture, GAUDI [26], which has been originally developed by the LHCb collaboration. ATLAS collaboration participated in extending this architecture and as a result, a kernel (not experiment-specific) implementation has been created. Athena is a combination of this kernel framework and ATLAS-specific enhancements, such as the event data model and event generator framework.

5.1 Athena repository

The CERN GitLab service is used to host the Athena code. The repository holds all the code related to every workflow step mentioned in the previous paragraph. However, the ATLAS software releases aren't usually built with the complete code base, but rather as a number of self-contained subsets of the code which deal with a specific step of the ATLAS production process. These resulting builds, called Athena projects, carry the name of the process for which they are responsible, for example `AthSimulation`, `AthAnalysisBase` or `AthDerivation`. An Athena project, which is a complete build of almost the entire repository, also exists and is used primarily to ensure the repository can be built consistently.

Athena project builds that are approved and ready for production are installed onto the ATLAS production CVMFS server. Every release is tagged with a release number, formatted as $A.B.X[.Y]$, where A refers to the release series and B is the release flavour, which usually corresponds to a particular Athena project in the repository [27]. X and Y are the major and minor release numbers which increase monotonically as bugs are fixed and features are added. Table 1 explains the purpose of the principal release flavours.

Release flavour	Purpose
0	Tier-0 reconstruction and corresponding simulation production
1	High Level Trigger online data taking
2	Derivations and Analysis
3	Simulation development
5	The union of .0 and .1 (also known as TrigMC)
9	Upgrade studies

Table 1: Explanation of release flavours and their specific purpose.

This thesis deals with simulations of ITk strip modules, which are being developed to meet higher performance requirements during runs of the upgraded High-Luminosity LHC. Therefore the proper release flavour for this type of work is 21.9, which deals with

studies related to the HL-LHC upgrade.

5.2 Setup on the LXPLUS

In order to use Athena on the LXPLUS with specific changes to the geometry or to other packages, a sparse checkout of the `athena` repository is recommended. This is done by using ATLAS-specific setup commands (`setupATLAS` and `lsetup`), initiating the `athena` work directory and adding packages to the directory to allow for their modification. Listing 8 shows the complete set of commands to setup Athena this way.

```
setupATLAS
lsetup git python
git atlas init-workdir -b 21.9 https://gitlab.cern.ch/atlas/athena.git
cd athena
git atlas addpkg InDetSLHC_Example
git atlas addpkg SCT_Digitization
git clone -b master https://gitlab.cern.ch/Atlas-Inner-Tracking/ITKLayouts.git
mkdir build
cd build
asetup 21.9,latest,Athena
cmake ../athena/Projects/WorkDir
make -j
source x86_64-slc6-gcc62-opt/setup.sh
```

Listing 8: Sparse checkout of Athena on the LXPLUS.

The `InDetSLHC_Example` package is checked out, which is used to run Athena with changes to the Inner Detector geometry. The `SCT_Digitization` package, where parameters of the digitization process can be configured, is added to the check out as well. Finally the `ITKLayouts` package is cloned, which allows to make changes to the layout geometry. Then the latest build of Athena is set up and the local work directory is built using `CMake`. Finally a setup script is sourced, thus overwriting the selected packages in the default Athena build by the modified local versions.

In order to pick up the local version of geometry, rather than use one of the geometries defined in database, the `UseLocalGeometry` flag must be set to `True`.

5.3 Simulation infrastructure

The Athena infrastructure is illustrated in Fig. 43. The main components of Athena are Services, Algorithms and Tools [28] [29].

Services are software entities available in the entire framework, which perform common tasks such as printing of messages during job runs, configuring tools or algorithms or handling data storage.

Algorithms are the main applications which perform specific tasks during the event processing loop, usually by taking input data, manipulating it and producing new output

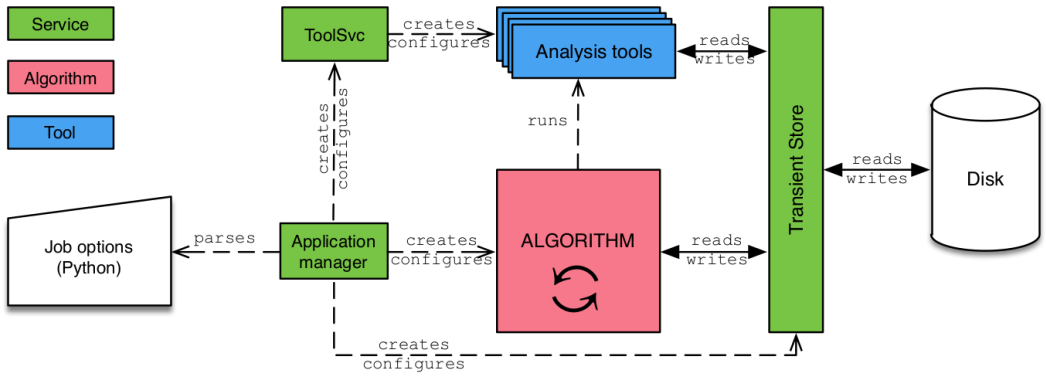

Figure 43: A scheme of Athena infrastructure. From [28].

data. The algorithms included in Athena are almost exclusively written in C++. Algorithms are configured with the use of Python jobOptions file. Every algorithm consists of three methods:

- `Initialize()` – runs at the start of a job,
- `Execute()` – runs once per every event,
- `Finalize()` – runs at the end of a job.

Tools represent the code which actually performs the desired operations during the `Execute()` method of the algorithms.

Two types of data storage are being used. A transient store, called StoreGate, holds data objects which are manipulated by algorithms during the event processing. In this case the data is held in computer memory. Final processed data objects are then stored to a persistent store (a disk) in POOL/ROOT format.

5.4 Athena job

A typical Athena job, which represents the entire simulation process, consists of several processing stages [30]:

1. fundamental event simulation, typically consisting of hard non-elastic proton-proton collisions and generation of stable particles,
2. simulation of interactions of the final state particles with the detector,
3. digitization of the detector hits to provide an output format equivalent to the real detector.

The simulation of fundamental ATLAS events is handled by a variety of event generator codes. These include general-purpose event generators, particle guns, cosmic ray and cavern background simulators, and fundamental process simulators covering physics from

Standard Model QCD and EW physics to exotic processes. Every event generator compatible with Athena produces event records in the HepMC format, these records are written to the transient store and eventually to disk in POOL/ROOT format.

Final state particle interactions with the detector volume are simulated using Geant4. This step consists of simulation of material interactions and generation of hits in the active volume of the detector. The Geant4 geometry is built from a geometry database and GeoModel interface, common to simulation and reconstruction applications. The resulting data objects describing the detector hits are stored in transient or persistent storage and logged. Configuration and application control is done through a Python interface.

During the digitization process the hits from an event are overlaid with background process hits. The response of every subdetector element is then simulated in order to create raw data object (RDO) files, which are equivalent to read-out files of the real detector.

5.5 Event Generators

Event generators supported by Athena can be divided into three categories [31]. The first category describes full generators, which simulate particle collisions but are also capable of performing parton shower and fragmentation. Examples of such event generators are Pythia [32], Herwig [33] or Sherpa [34].

The second category includes add-on generator packages. These packages do not generate events from particle collisions, but rather modify existing event records. They retrieve the HepMC container from the transient store, modify the events and write them back into the store.

Most of the generators are part of the third category. These basic parton-level generators require a full generator from the first group to perform the parton shower and the fragmentation.

5.6 Running a simulation with jobOptions file

The configuration of Athena before a simulation run starts is done through Python files that contain so-called job options. Upon execution, each of the files passed to the Athena executable is loaded in sequence. The jobOptions file allows to set global simulation settings, such as the number of events to be processed, and also customize the event processing stage by adding more algorithms to the loop. Listing 9 shows an example of such a jobOptions file.

```
from InDetSLHC_Example.SLHC_JobProperties import SLHC_Flags
SLHC_Flags.UseLocalGeometry.set_Value_and_Lock(True)

from AthenaCommon.AthenaCommonFlags import athenaCommonFlags
athenaCommonFlags.EvtMax = 10000
```

```
from AthenaCommon.AlgSequence import AlgSequence
seq = AlgSequence()

from SiHitAnalysis.SiHitAnalysisConf import SiHitAnalysis
seq += SiHitAnalysis("SiHitAnalysis")
```

Listing 9: Example jobOptions file.

In this example the previously mentioned `UseLocalGeometry` flag is set to `True` to allow for use of local geometry layouts. The number of events to be processed is then set to 10000 with the use of the `EvtMax` flag. Finally an instance of the `AlgSequence` object is created and user can add arbitrary algorithms to the sequence. Here the `SiHitAnalysis` algorithm is added, which ensures detailed information about detector hits from every event are saved in the persistent store.

5.7 Running a job with a wrapper script

Running more complex jobs on production scale requires more flexible objects than the jobOptions file. The ATLAS software framework provides dedicated Python scripts, called transformations, that take a skeleton Athena jobOptions file as an input and a set of command-line parameters. These transformations provide a convenient and flexible mechanism to use predefined job options fragments [31].

On the LXPLUS an Athena job is usually run with a bash wrapper script. This script sets variables such as input and output file names that are then passed to Athena. The main part of the wrapper script are the transformation scripts, one for simulation (`Sim_tf.py`), reconstruction (`Reco_tf.py`) and digitization (`Digi_tf.py`), which are executed with a number of arguments and so-called steering files, further configuring the job. Listing 10 shows a part of the wrapper script where the `Sim_tf.py` transformation script is executed.

```
run Sim_tf.py
    --inputEVNTFile    "$evnt"
    --maxEvents        10000
    --preInclude
          all:'SLHC.py,
               SLHC_Setup.py,
               SLHC_Setup_Strip_GMX.py'
    --preExec
          all:'from InDetSLHC_Example.SLHC_JobProperties import SLHC_Flags;
               SLHC_Flags.UseLocalGeometry.set_Value_and_Lock(True);'
    --postInclude
          all:'PyJobTransforms/UseFrontier.py,
               SiHitAnalysis.py'
```

Listing 10: Simulation transformation script.

The steering files can be loaded either before or after the step is executed by using the `preInclude` or `postInclude` options. In the listing above, the steering files responsible for the setup of the local geometry (`SLHC.py` and so on) use the `preInclude` option, because this needs to be configured before the run starts. On the other hand, the `SiHitAnalysis.py`, which ensures detector hit histograms are saved, can be loaded after the run and therefore uses the `postInclude` option. Additional configuration commands can be executed, also either before or after the run, with the `preExec` and `postExec` options.

Steering files represent a flexible way to add functionality to an Athena job simply by passing preexisting fragments to the transformation scripts, thus eliminating the need to modify the Athena source code itself.

6 Athena Module Simulation

Athena framework is a complex tool designed for the simulation of the entire ATLAS detector. Performing a simulation of a single strip module was never one of its intended uses and as a result, many steps have to be taken before a desired simulation setup is reached.

The first step is to disable unnecessary parts of the ATLAS detector, in ideal case this would mean leaving only the strip detector portion of the Inner Detector enabled. Then the strip detector layout is to be modified to contain only one strip module. This module then has to be targetted by a beam of electrons with the same properties as during the test beam measurements and in Allpix-Squared simulations. During the simulation, necessary information from the single module has to be obtained, preferrably in a format compatible with the analysis script used for Allpix-Squared simulation. Finally the configuration of the Athena simulation has to be unified with the one from Allpix-Squared, as much as the differences of the two frameworks allow.

6.1 ATLAS Subdetectors

In a default Athena configuration a number of ATLAS subdetectors are enabled. During a simulation run a table is printed out to inform the user of what configuration is being used. An example of such a printout is shown in Table 2.

:	bpipe	pixel	SCT	TRT	BCM	DBM	Lucid	ZDC	ALFA	AFP	FwdRegion	em	HEC	FCal	Tile	MBTS	MDT	CSC	TGC	RPC	sTGC	Micromegas
dcs :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
detdescr :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
digitize :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
geometry :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
haveRDO :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
haveRIO :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
makeRIO :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
overlay :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
pileup :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
readRDOBS :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
readRDOPool :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
readRIOBS :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
readRIOPool :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
simulate :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
simulateLVL1 :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
writeBS :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
writeRDOPool :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON
writeRIOPool :	ON	ON	ON	--	--	--	--	--	--	--	--	ON	ON	ON	ON	--	ON	ON	ON	ON	ON	ON

Table 2: ATLAS subdetectors enabled by default during a simulation run.

The framework keeps track of its subdetectors by the usage of objects called DetFlags. Such an object exists for every individual subdetector and instructs Athena whether it is to be simulated and which particular steps are to be carried out with it, such as reconstruction of tracks or digitization of hits. User can modify these flags in a steering file that is loaded before the simulation initializes, thus enabling or disabling specific subdetectors.

The optimal case is to simulate only the strip detectors and to disable every other subdetector. Disabling most of the subdetectors doesn't cause any issues during a simulation run, however that cannot be said for the beampipe and pixel subdetectors. In the current Athena 21.9 nightly build 2020-03-09T2138 a simulation without these two subdetectors crashes during event processing. However, leaving these two subdetectors enabled is not a problem as neither the beampipe nor the pixel detectors overlap with the strip detectors

and their presence doesn't complicate the intent to simulate a single strip module.

The desired subdetector configuration is realized by setting the DetFlags on for the beampipe, pixel and strip detectors and setting them off for everything else. Below are the few first lines of the DetFlags steering file which is used. In Listing 11 first the beampipe, the pixel subdetector and the SCT are turned on and then the rest of the subdetectors are turned off.

```
from AthenaCommon.DetFlags import DetFlags
DetFlags.bpipe_setOn()
DetFlags.pixel_setOn()
DetFlags.SCT_setOn()
DetFlags.Calo_setOff()
DetFlags.TRT_setOff()
...
```

Listing 11: Example of DetFlags usage.

Note that the strip detector is internally called SCT, which is the name of the strip portion of the current Inner Detector, but the proper layout of the planned ITk is simulated. This is simply an out-of-date naming convention. The printout during a simulation (Table 3) confirms the subdetector configuration changes according to user input.

:	bpipe	pixel	SCT	TRT	BCM	DBM	Lucid	ZDC	ALFA	AFP	FwdRegion	em	HEC	FCal	Tile	MBTS	MDT	CSC	TGC	RPC	sTGC	Micromegas
dcs :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
detdescr :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
digitize :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
geometry :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
haveRDO :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
haveRIO :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
makeRIO :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
overlay :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
pileup :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
readRDOBS :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
readRDOPool :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
readRIOBS :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
readRIOPool :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
simulate :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
simulateLVL1 :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
writeBS :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
writeRDOPool :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
writeRIOPool :	ON	ON	ON	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Table 3: A custom configuration of ATLAS subdetectors during a simulation run.

6.2 Custom detector layout

The layout of the ATLAS ITk, as illustrated on Fig. 12, is already prepared in the 21.9 branch of Athena. However, the goal is to simulate a single strip module. To achieve this the layout of the detectors has to be first pruned to one module which can then be positioned and oriented in a desirable way.

The layout of pixel detectors is not changed in any way, as they are not relevant to a simulation of a strip module. Strip modules are organised in four concentric layers in the barrel region and six disks in both end-cap forward regions. This default layout is illustrated in simulation output in Fig. 44.

The easiest way to simulate a single module is to use one from these existing layers. End-cap modules are ruled out immediately, as their geometry is very different from the module prototype tested at test beam measurements. That leaves the four barrel

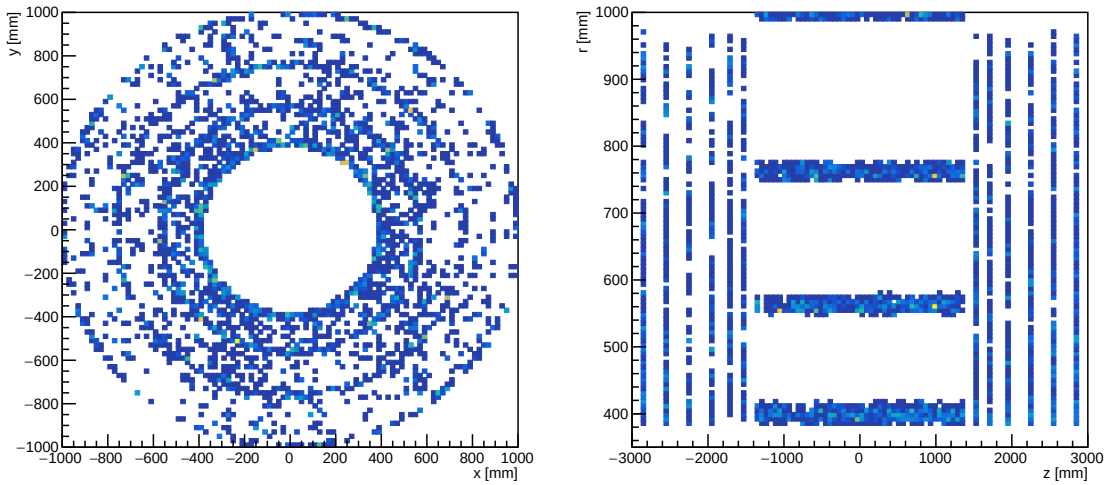

Figure 44: Simulation of hits in the default ITk layout.

layers whose modules differ primarily in strip length, as was discussed in Section 2.3. The innermost barrel layer was chosen, however the remaining three layers would also be perfectly acceptable because the strip length of the modules can be easily changed.

To obtain the single module layout the following steps have to be taken: first the end-cap layers and the outer three barrel layers are to be removed. The next step is to limit the number of staves, the local support structures for strip barrel modules, in the remaining barrel layer to one. Then the stave is to be modified to contain only one strip module.

In Athena, the layout and geometry of the ITk is defined in two types of files – the geometry of pixel detectors in XML files, the geometry of strip detectors in GMX files. The GMX files are eventually transformed to XML to unify the formats.

The strip detectors are organised in a hierarchy of logical volumes, the largest one being an entire barrel or end-cap layer and the smallest ones being individual components of a strip module. The largest logical volume is created first as a mother volume and every smaller volume is placed inside as a daughter volume.

To better illustrate the format of the GMX files and how the layout is defined in it, a code to create the innermost barrel layer is shown in Listing 12. Some lines of the original code were removed in the listing to make it easier to understand.

```
<logvol name="Barrel0" shape="shBarrel0" material="N2" alignable="true">
    <multicopy name="PlaceStavesB0" n="N_StavesInCyl_0">
        <transformation name="XfStavesB0">
            <translation x="CylRadius_0"/>
            <rotation zcos="1" angle="2 * PI / N_StavesInCyl_0"/>
        </transformation>
        <assemblyref ref="StavePairSS"/>
    </multicopy>
</logvol>
```

Listing 12: Creation of a barrel layer in GMX layout format.

The crucial part of the code is a `multicopy` mechanism, which places copies of a given volume in a systematic and controllable way. In this case it places around the beam axis (z-axis) copies of a logical volume representing a whole stave and it does so a number of times defined by the `N_StavesInCyl_0` parameter, by default 28 times. The initial position is reached by moving to a radius defined by `CylRadius_0` (399 mm) and the first stave is placed, then a new position is reached by rotating around the beam axis by $2\pi/28$ and another stave is placed. This procedure is repeated a total of 28 times and this way the 28 staves cover the entire 2π angle around the beam axis.

Obtaining a barrel layer with only one stave is as simple as setting the value of `N_StavesInCyl_0` parameter to 1 which instructs the `multicopy` mechanism to place only one copy of the stave.

Removal of an entire layer can be done by commenting out or removing the entire block of code where the appropriate logical volume is defined. This is how the outer three barrel layers and the end-cap layers are removed.

Now the strip detector layout consists of only one stave, but that still contains 28 modules. The code for creation of a stave is similar to that above, again the `multicopy` mechanism is responsible for placing copies of strip modules on the stave and additional support volumes are placed around the modules. To get a single module the `multicopy` parameter is again changed to 1, the remaining module is centered at $x = 0$ and $z = 0$ and the support volumes are removed.

The final geometry layout with a single strip module is illustrated in a simulation output in Fig. 45.

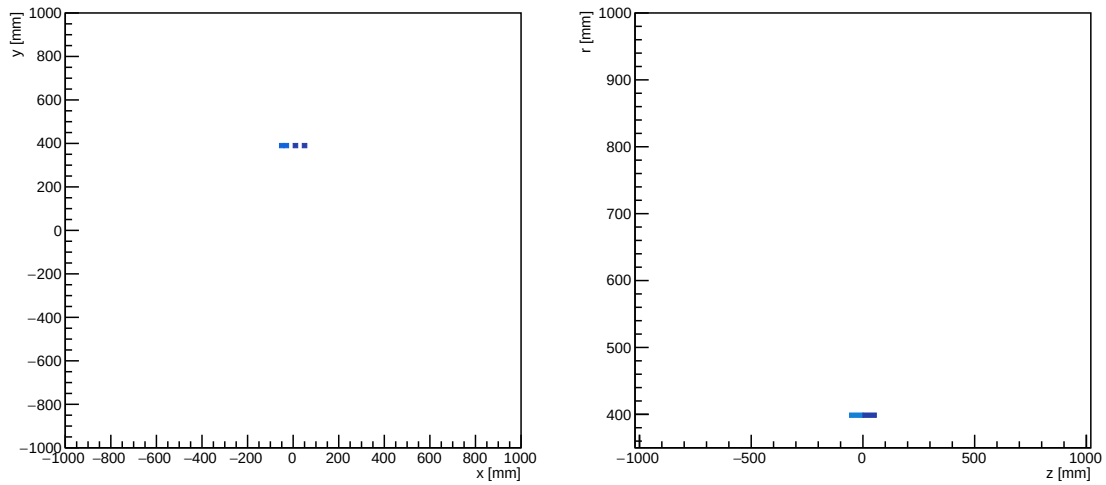

Figure 45: Simulation of hits in the modified single module layout. The module is placed at $x = 0$, $z = 0$ and radius $r = 399$ mm.

6.3 Event generation

For a simulation of a single strip module a simple event generator, called a particle gun, is used. There are many ways to configure this generator, one of them is defining the

distribution of the particle source position and of the particle momentum. A part of the code for setting up a particle gun event generator is shown in Listing 13.

```
import ParticleGun as PG
pg = PG.ParticleGun(randomSvcName=simFlags.RandomSvc.get_Value(), randomStream="
    SINGLE")
pg.sampler.pid = 11
pg.sampler.pos = PG.PosSampler(x=[-1,1], y=390, z=[-1,1])
pg.sampler.mom = PG.MXYZSampler(px=0, py=5800, pz=0)
```

Listing 13: Particle gun event generator configuration.

The particle gun was set up to fire 5.8 GeV electrons along the y-axis from the radius of 390 mm, the x and z positions of the gun can vary in a small interval of $(-1; 1)$ mm. The source is therefore located 9 mm below the strip module and is emitting particles perpendicularly upwards into the module.

The output of the event generator is obtained by running a separate Athena job with a jobOptions file containing the event generator configuration. This generates a POOL/ROOT file which can be then used as input for the main Athena simulation of a single strip module. This approach is advantageous as the generated events are stored persistently and don't have to be recreated for every subsequent Athena simulation, which significantly speeds up the entire simulation process.

6.4 Simulation setup

When setting the Athena simulation parameters the goal is to ensure it is configured similarly to the Allpix-Squared simulations. However due to fundamental differences between the two frameworks, a complete unification of the settings is not possible.

The silicon sensor model is again based on the ATLAS17 long strip sensor, with one row of 1280 strips 96.721 mm long. The strip length is slighty different from the Allpix-Squared model, which was 100 mm. However, this difference is not relevant because changing dimensions perpendicular to the applied electric field doesn't affect the transport of the charge carriers in the silicon volume. The strip pitch is 75.5 μm. The active sensor thickness was set to 280 μm based on findings discussed in Section 4.11.

The bias voltage is to -400 V, while the depletion voltage is set to -300 V, in correspondence with the Allpix-Squared simulations and the test beam measurements.

The QGSP_BERT physics list is used. While the FTFP_BERT_LIV is used in Allpix-Squared simulations, the results discussed in Section 4.6 show that there is a negligible difference between these two physics lists.

Electronics noise is disabled in the Athena simulation, as well as the cross talk effect model implemented in Athena. The reason for disabling these effects is that, contrary to Allpix-Squared, they do not affect the amount of charge in individual strips, but rather add noise or cross talk hits to the detector. Since the simulation output is processed

by analyzing the strip charge information, neither the noise nor the cross talk make any difference and there is no reason to leave them turned on.

The unability to simulate cross talk is not an issue, since a simple cross talk model is implemented in the analysis script. The lack of noise in the simulation is problematic, because, compared to the test beam measurement data, the simulation results are less realistic. They can, however, still be in principle compared to Allpix-Squared simulation with zero noise.

6.5 Data analysis

The preferred way to perform analysis of the Athena simulation data is to use the same Python3 script used for Allpix-Squared simulations. Therefore information about the amount of charge in every individual strip of the module has to be obtained. This can be done by including the steering file `RDOAnalysis.py`, which is responsible for creating such output. Fortunately, the format of the file is practically the same as from the Allpix-Squared simulation – a set of objects in the `SCT_RDOAna` tree stores information about every strip with non-zero charge during a single event. The only differences are the naming conventions of these objects and different methods which these objects use, both can be easily addressed in the analysis script.

User passes a `source` string to the main function of the analysis script, either "allpix" or "athena", to ensure the script navigates the simulation data files and extracts the necessary information – the number of events and the tree with charge information – properly based on their source. The part of the analysis script which handles this information extraction is shown in Listing 14.

```
if source == "athena":
  hitTree = rootFile.Get("SCT_RDOAnalysis").Get("SCT_RDOAna")
  nOfParts = 0
  for event in hitTree:
    if len(list(event.charge)) > 0:
      nOfParts += 1
elif source == "allpix":
  hitTree = rootFile.PixelCharge
  nOfParts = int(str(rootFile.config.Get("Allpix").Get("number_of_events")))
```

Listing 14: Analysis code to obtain the number of events and the hit tree objects.

If the data file is from an Allpix-Squared simulation, every necessary information is extracted straight from the file, because the framework stores complete configuration of the simulation in the output file. On the other hand, Athena output files do not include such information and therefore obtaining the number of events is done by iterating over the tree with hit information and counting the events.

After this initial information extraction the main loop performing the threshold scanning has to be adjusted to allow for processing of output files from both Allpix-Squared

and Athena frameworks. Based on the `source` string the appropriate code branch is chosen, the extraction of the charge information is slighty different in both branches but ultimately an array is created with the amount of collected charge on every individual strip. This array is then passed to the rest of the threshold scanning loop to calculate the efficiency and cluster size for a given threshold. The analysis code is shown in Listing 15.

```
for aThr in np.arange(thrStart, thrEnd, thrStep)
  for anEvent in hitTree:
    stripCharge = np.zeros(nOfStrips)
    if source == "allpix":
      for stripHit in anEvent:
        stripCharge[stripHit.getIndex().X()] = stripHit.getCharge()
    elif source == "athena":
      nonEmptyStrips = list(anEvent.strip_sdo)
      nonEmptyStripsCharge = list(anEvent.charge)
      for i in range(len(nonEmptyStrips)):
        stripCharge[nonEmptyStrips[i]] = nonEmptyStripsCharge[i]

    clusterSize = len(np.where(stripCharge > aThr)[0])
    if clusterSize > 0:
      effHist.Fill(aThr)
          clusHist.Fill(aThr, clusterSize)
    effHist.Scale(1/nOfEvents)
```

Listing 15: Analysis code to calculate efficiency and cluster size from Allpix-Squared or Athena simulation.

The threshold scanning is again performed from 0 to 8 fC with a step of 0.1 fC. The obtained efficiency points for a given threshold are also fitted with a skewed complementary error function defined in Section 4.3.

6.6 Simulation results

Results obtained from the Athena simulation were compared to the final Allpix-Squared simulation results shown in Section 4.13 and to the test beam measurement data. The comparison of the efficiency and the average cluster size shows that the Athena simulation results differ significantly from the other two sets of results.

The efficiency starts dropping at much lower thresholds in the Athena simulation (Fig. 46). As a result, the median charge, which corresponds to the threshold at which the efficiency is 50 %, is (3.028 ± 0.004) fC, while the median charge from the Allpix-Squared simulation and from the test beam data is (3.601 ± 0.002) fC and (3.65 ± 0.01) fC, respectively. The median charge from the Athena simulation is therefore lower by approximately 16–17 %.

Average cluster size as a function of charge threshold obtained from the Athena simulation (Fig. 47) is also very different from the Allpix-Squared and the test beam measurement results. At lower thresholds, up to 2–2.5 fC, the average cluster size from the Athena simulation is larger, which suggests stronger charge sharing among the strips. However,

at higher thresholds the average cluster size approaches unity, which means that there is almost no charge sharing among the strips.

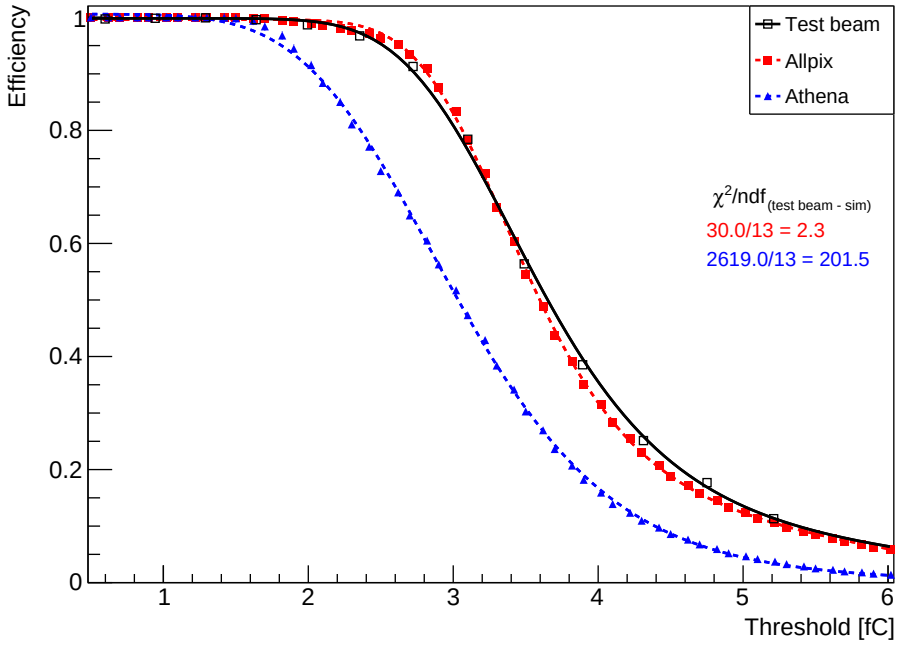

Figure 46: Efficiency curves obtained from Athena and Allpix-Squared simulations and from test beam measurements.

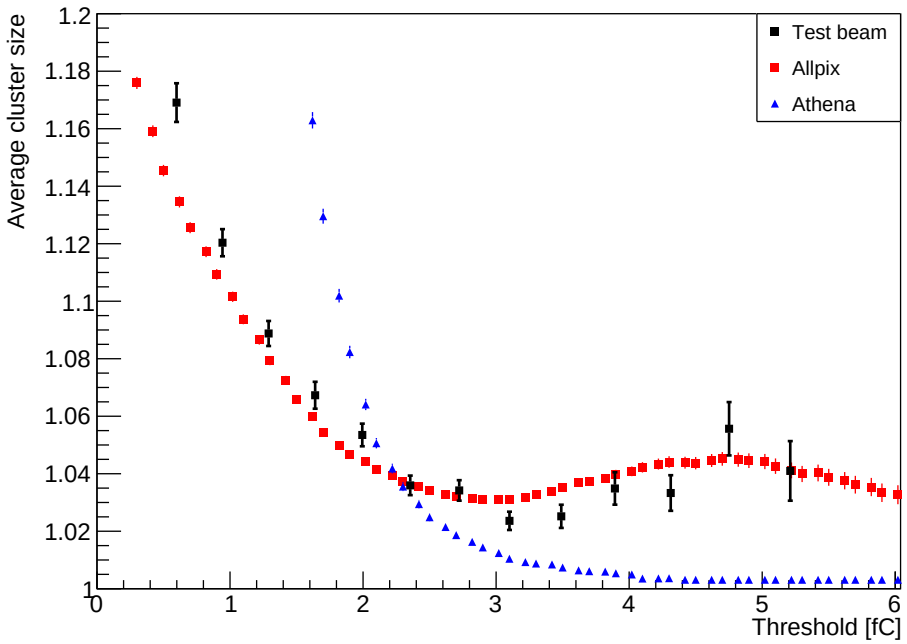

Figure 47: Average cluster size plots obtained from Athena and Allpix-Squared simulations and from test beam measurements.

6.7 Production cuts

A possible explanation for no charge sharing at higher thresholds in Athena simulations could be a missing production of δ-electrons, which are created in events where a large momentum is transferred in a collision between the incident particle and an electron. As

a result, such an electron with energies of tens to hundreds of keV can travel up to 100 μm in the silicon, depositing energy along its path through ionisation.

The production of secondary particles, including the δ-electrons, is controlled in Athena – more precisely in Geant4 which handles the simulation in Athena – by setting production cuts. A production cut is a distance which a newly created secondary particle has to be able to travel in a given volume. If it cannot do so, typically because its energy is too low, it is immediately terminated and its entire energy is deposited into the current volume.

The production cut is set in Allpix-Squared simulations by default as one fifth of the smallest strip dimension, in this case one fifth of the 75.5 μm strip pitch – 15.1 μm. In Athena the default production cut is 50 μm and simulations were performed using both the default value and the value used by Allpix-Squared.

The efficiency plots (Fig. 48) show a slight overall decrease in efficiency, at higher thresholds this decrease is more prominent. This can be explained by more secondary particles being able to propagate through the sensor, thus carrying away more energy from the principal strip, ultimately decreasing the amount of charge collected on it.

The expected result of a lower production cut is a higher average cluster size, since more particles are propagating through the sensor volume and causing hits in other strips than the principal one. On the other hand, by reducing the amount of charge on the principal strip, the threshold for calling a hit is more difficult to surpass.

The Athena simulation results show either no change or lower average cluster size, depending on the threshold (Fig. 49). While the lower average cluster size at lower thresholds is desirable in terms of increasing the agreement with the test beam and the Allpix-Squared simulations, the cluster size remains very high compared to the test beam results. The main motivation for exploring the effect of production cut was to increase the average cluster size at higher thresholds, which, however, did not happen.

6.8 Median charge scaling

Additional simulations were carried out in both Athena and Allpix-Squared frameworks for various sensor thicknesses ranging from 25 μm to 320 μm with the goal of describing how the median charge scales with the sensor thickness (Fig. 50). The median charge value is obtained from the x_0 parameter of the fit function (7).

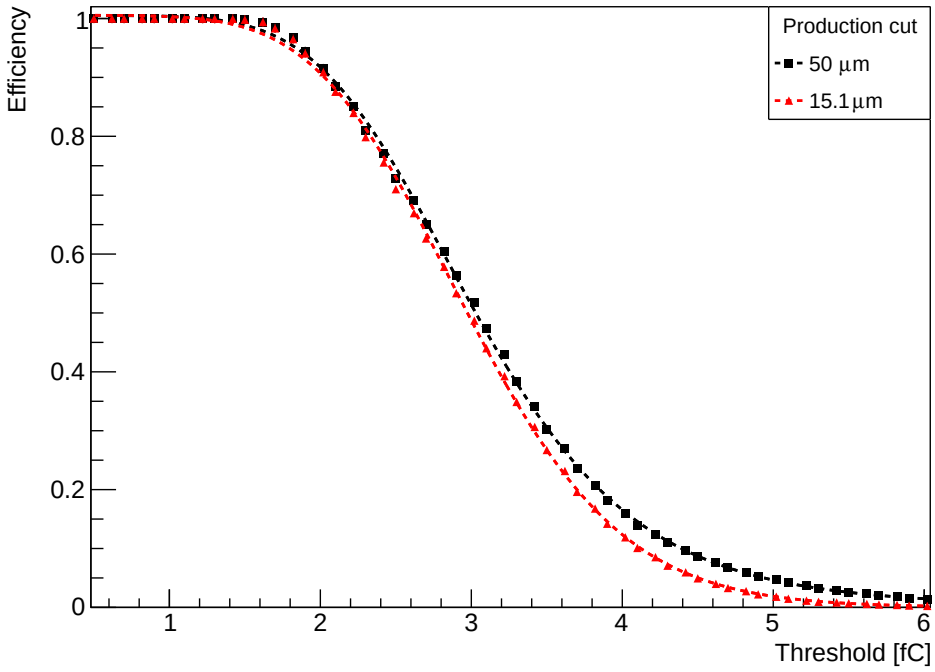

Figure 48: Efficiency curves obtained from Athena simulation with different values of the production cut.

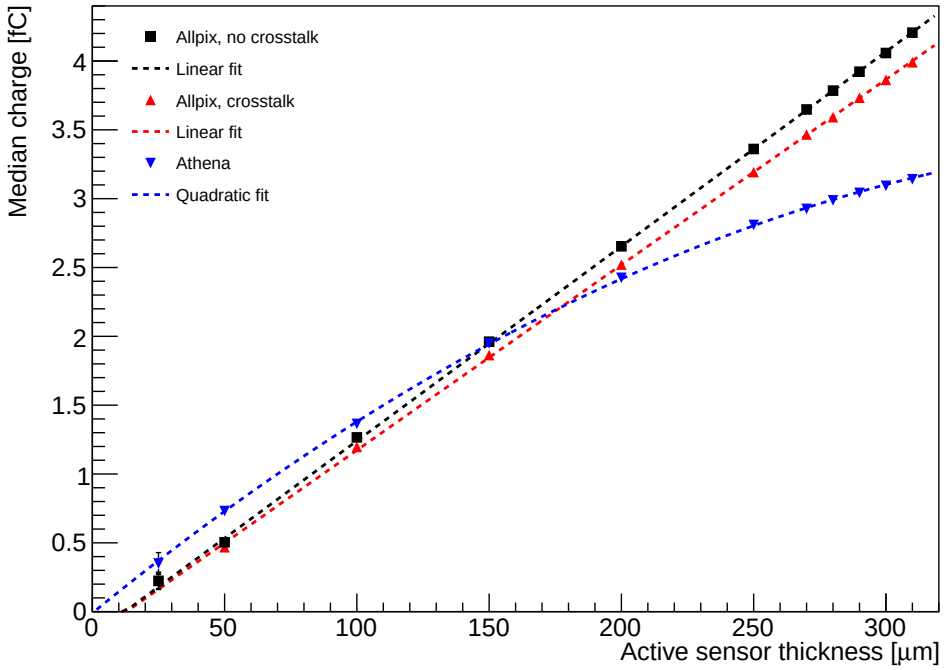

Figure 50: Median charge scaling with sensor thickness in Allpix-Squared and Athena simulations.

Allpix-Squared simulations exhibit linear scaling of the median charge in the entire range of simulated thicknesses. Application of cross talk effect on data from Allpix-Squared simulations has the effect of lowering the slope of the linear scaling, as it effectively reduces the sensor thickness – in the used cross talk model the capacitive coupling between a strip and the backplane reduces the amount of charge on a strip by a fixed percentage.

On the other hand, the Athena simulation results show a more complicated scaling,

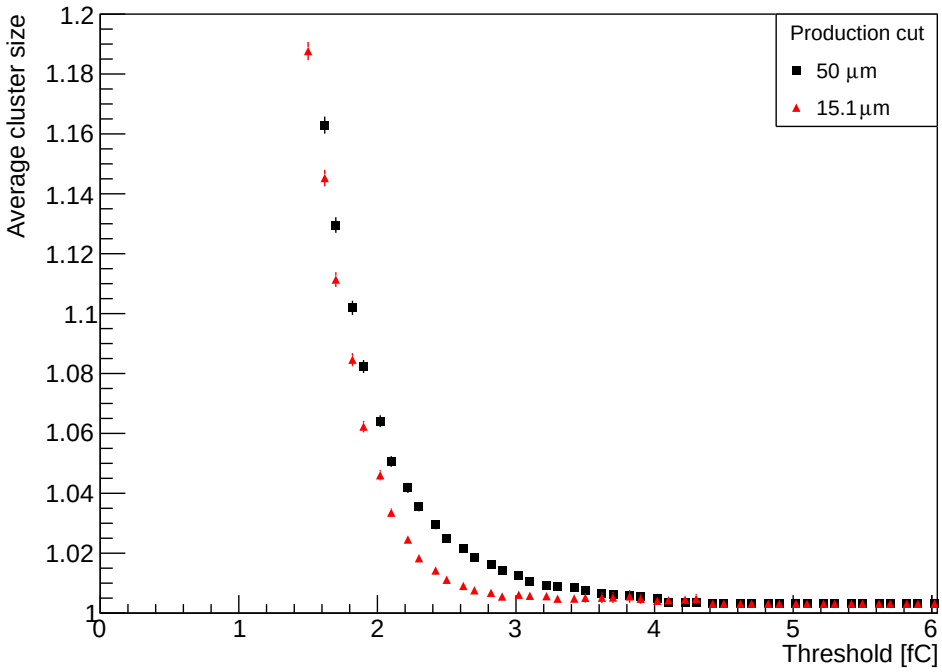

Figure 49: Average cluster size plots obtained from Athena simulation with different values of the production cut.

better described by a quadratic fit than a linear one. However, it could be approximated by a linear function in the realistic sensor thickness range of around 270–320 um (Fig. 51), but with a very different slope than in Allpix-Squared.

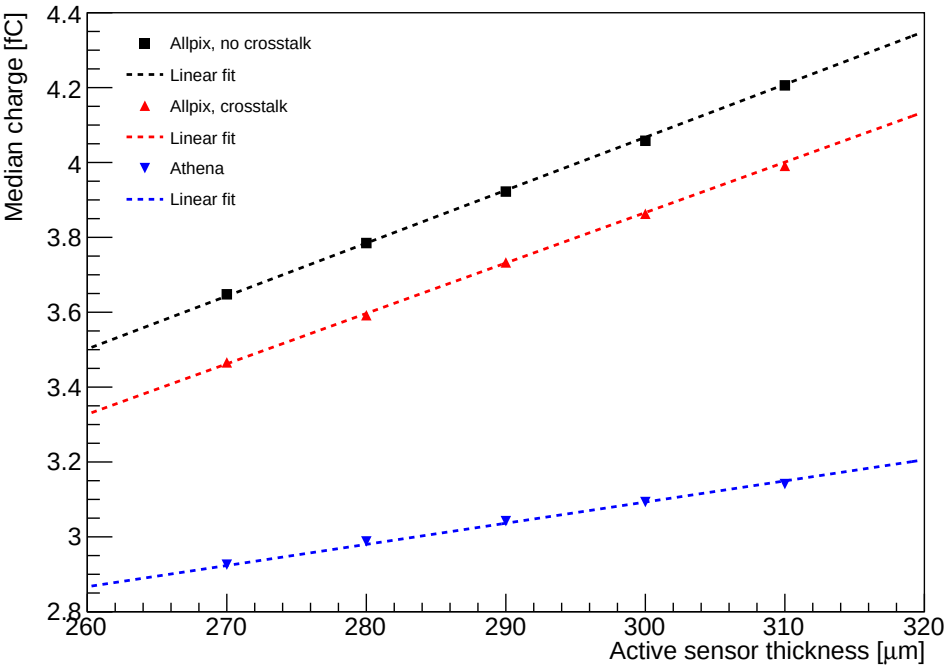

Figure 51: Median charge scaling with sensor thickness in Allpix-Squared and Athena simulations in the realistic sensor thickness range.

This shows significant differences in the digitization model. Since both simulation frameworks use Geant4 and the initial conditions – primary particle type, energy, sensor

thickness and so on – are set very similarly, the amount of deposited energy in the silicon volume should be similar in simulations using either framework. This deposited energy is then transformed to charge carriers, the number of which should again be similar between Allpix-Squared and Athena simulations. During the digitization process this deposited charge is read out and it is most likely this step of the simulation where the differences originate.

One of the possible explanations for this disagreement could be related to the fact that the current digitization model implemented in the Athena framework was written for the SCT detector, which differs from the ITk. The main difference is the sensor type – p^+-in-n for SCT strip detectors, n^+-in-p for the ITk strip detectors.

It is not clear what changes to the digitization model have to be made to improve the agreement between the results. Since the configuration of the digitization process is much more complicated in Athena than in Allpix-Squared, attempting to achieve any significant improvement would probably mean modification of the Athena code and would go beyond the scope of this work.

Summary

The purpose of this work was to perform simulations of the ATLAS ITk long-strip silicon detector based on the ATLAS17LS sensor in two simulation frameworks – Allpix-Squared and Athena – and to compare the obtained results with data from test beam measurements. The focus was placed on two main detector characteristics, the efficiency and the average cluster size. The goal was to improve the agreement between the simulations and the test beam results by exploring the influence of various simulation parameters and adjusting them.

Simulations of a strip detector model were performed in Allpix-Squared. Since this framework is developed specifically for simulation of silicon detectors, setting up a simulation was fairly simple. The influence of several parameters was established, among which the biggest impact on the detector characteristics could be credited to sensor thickness. The choice of either a TCAD-generated or a linear model of an applied electric field also had a significant effect. A simple cross talk effect model was implemented in the analysis script which further influenced the results. These factors contributed to a major overall improvement in agreement between the test beam measurement data and the Allpix-Squared simulations. Simulations of various incident angles were also performed and the results showed the same trends seen in the test beam measurement data.

Simulations of a single strip detector were also performed in the Athena framework. Athena is designed for simulation of the entire ATLAS detector and therefore setting up a single detector simulation was a much more complicated task than in Allpix-Squared. The initial simulations of the perpendicular incidence showed the Athena results differ significantly from the Allpix-Squared and the test beam measurement results. A study of sensor thickness influence showed very different scaling of the median charge with the thickness, which could suggest differences in the digitization process of the two frameworks. While no improvement of the agreement between the Athena simulations and the test beam data was achieved, the indication of a large degree of discord between the framework and the test beam measurement is still a valuable result.

List of abbreviations

ABC	ATLAS Binary Chip
ALICE	A Large Ion Collider Experiment
AMAC	Autonomous Monitor and Control
ASIC	Application Specific Integrated Circuit
ATLAS	A Toroidal LHC Apparatus
CERN	Conseil européen pour la recherche nucléaire (European Organization for Nuclear Research)
CMS	Compact Muon Solenoid
CVMFS	CernVM File System
DESY	Deutsches Elektronen-Synchrotron
DUT	Device-under-test
EC	End-cap
EW	Electroweak
EoS	End-of-Substructure
GBL	General Broken Lines
HCC	Hybrid Controller Chips
HL-LHC	High-Luminosity LHC
IBL	Insertable B-Layer
ITk	Inner Tracker
LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty
PAI	Photo-absorption ionisation
QCD	Quantum chromodynamics
RDO	Raw data object
SCT	Semiconductor Tracker
TRT	Transition Radiation Tracker

List of Figures

References

[1] LUTZ, G.; *Semiconductor Radiation Detectors.* 1st ed. Springer, 2007. ISBN 978-3-540-71678-5

[2] STREETMAN, Ben G.; BANERJEE, S.; *Solid State electronic Devices.* 5th ed. New Jersey: Prentice Hall, 2000 . p. 524. ISBN 0-13-025538-6.

[3] KNOLL, Glenn F.; *Radiation Detection and Measurement.* 3rd ed. Michigan: John Wiley & Sons, 2000. ISBN 978-0471073383.

[4] TANABASHI, M. et al. (Particle Data Group); *Review of Particle Physics (2018).* Phys. Rev. D 98, 030001 (2018).

[5] HARA, K. et al.; *Testing of bulk radiation damage of n-in-p silicon sensors for very high radiation environments*, Nucl. Instr. Meth. Phys. Res. A636 (2011), S83–S89.

[6] ATLAS Collaboration; *Technical Design Report for the ATLAS Inner Tracker Strip Detector*, CERN-LHCC-2017-005, `https://cds.cern.ch/record/2257755`

[7] `https://twiki.cern.ch/twiki/bin/view/Atlas/RadiationBackground\SimulationsStep3X`. [accessed on 2020-04-28]

[8] VOS, M. et al.; *Charge collection with binary readout from a test beam perspective*, CERN-ATL-INDET-2003-011. `http://cds.cern.ch/record/685474`.

[9] JANSEN, H. et al.; *Performance of the EUDET-type beam telescopes.* EPJ Techn. Instrum. (2016) 3: 7.

[10] KLEINWORT, C; *General broken lines as advanced track fitting method.* Nucl. Instr. Meth. Phys. Res. A673 (2012), 107-110.

[11] ATLAS Collaboration; *ATLAS: Detector & Technology* [online]. CERN, 2020. `https://atlas.cern/discover/detector`. [accessed on 2020-03-15]

[12] ATLAS Collaboration; *ATLAS: Inner Detector* [online]. CERN, 2020. `https://atlas.cern/discover/detector/inner-detector`. [accessed on 2020-03-15]

[13] SPANNAGEL, S.; WOLTERS, K.; HYNDS, D; *Allpix^2: A Modular Simulation Framework for Silicon Detectors,* Nucl. Instr. Meth. Phys. Res. A901 (2018), 164-172.

[14] AGOSTINELLI S. et al.; *Geant4 – a simulation toolkit.* Nucl. Instr. Meth. Phys. Res. A506 (3) (2003), 250-303.

[15] BRUN R.; RADEMAKERS F.; *ROOT - An Object Oriented Data Analysis Framework,* Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. Meth. Phys. Res. A389 (1997), 81-86. See also `http://root.cern.ch/`

[16] GUENNEBAUD, G.; JACOB, B.; *Eigen v3*. 2010. http://eigen.tuxfamily.org [accessed on 2020-03-20]

[17] CERN IT Department; *LXPLUS Service*. 2020. http://information-technology.web.cern.ch/services/lxplus-service. [accessed on 2020-04-24]

[18] SPANNAGEL, S., WOLTERS, K., HYNDS, D; *User Manual for the Allpix² Simulation Framework*. https://project-allpix-squared.web.cern.ch/project-allpix-squared/usermanual/allpix-manual.pdf. [accessed on 2020-05-15]

[19] EUTelescope Collaboration; *EUTelescope: a generic beam telescope data analysis framework*. https://eutelescope.github.io/

[20] SPANNAGEL, S.; WILLIAMS, Jean M.; KROEGER, J.; *User Manual for the Corryvreckan Test Beam Data Reconstruction Framework, Version 1.0*. https://cds.cern.ch/record/2703012/

[21] Synopsys Inc.; *Technology Computer Aided Design (TCAD)*, https://www.synopsys.com/silicon/tcad.html. [accessed on 2020-05-08]

[22] Geant4 Collaboration; *Physics List Guide* [online]. 2017. http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html. [accessed on 2020-03-15]

[23] APOSTOLAKIS, J. et al.; *An implementation of ionisation energy loss in very thin absorbers for the GEANT4 simulation package* Nucl. Instr. Meth. Phys. Res. A453 (2000), 597-605.

[24] MIKESTIKOVA, M. et al.; *Electrical characterization of surface properties of the ATLAS17LS sensors after neutron, proton and gamma irradiation*, submitted to Nucl. Instr. Meth Phys. Res. A, manuscript NIMA PROCEEDINGS-D-20-00044 (2020). https://indico.cern.ch/event/803258/contributions/3582831/attachments/1962502/3262248/246-Mikestikova-SurfaceOverview.pdf

[25] KLEIN, Christoph T. et al.; *Initial Tests of Large Format Sensors for the ATLAS ITk Strip Tracker*, submitted to Nucl. Instr. Meth Phys. Res. A, manuscript NIMA PROCEEDINGS-D-20-00056 (2020). http://cds.cern.ch/record/2703672

[26] MATO, P. et al.; *Status of the GAUDI event-processing framework*. 2001. http://cds.cern.ch/record/1745147.

[27] ATLAS Collaboration; *ATLAS Software documentation*. 2017. https://atlassoftwaredocs.web.cern.ch/athena/

[28] CATMORE, J.; *The ATLAS data processing chain: from collisions to papers*. 2016. https://indico.cern.ch/event/472469/contributions/1982677/attachments/1220934/1785823/intro_slides.pdf

[29] CATMORE, J.; *An introduction to the ATLAS software*. 2010. http://atlas.fis.utfsm.cl/atlas/ATLASSOFT.pdf

[30] BUCKLEY, A.; *Simulation strategies for the LHC ATLAS experiment*. ATL-SOFT-PROC-2010-004. https://cds.cern.ch/record/1307557/

[31] AY, C. et al.; *Monte Carlo generators in ATLAS software*. J. Phys. Conf. Ser. 219 (2010) 032001

[32] SJÖSTRAND, T.; MRENNA, S.; SKANDS, P.; *An Introduction to PYTHIA 8.2*, Comp. Phys. Comm. 191 (2015), 159-177.

[33] BÄHR, M.; GIESEKE, M.; GIGG, Martyn A. et al.; *Herwig++ physics and manual*. Eur. Phys. J. C 58, 639–707 (2008).

[34] GLEISBERG, T.; HOCHE, S.; KRAUSS, F.; SCHONHERR, M. et al.; *Event generation with Sherpa 1.1*, JHEP 02 (2009) 007