

Instrumentation for silicon tracking at the HL-LHC

Licentiate thesis

REBECCA CARNEY

May 2017
Stockholm University

Abstract

In 2027 the Large Hadron Collider (LHC) at CERN will enter a high luminosity phase, delivering 3000 fb^{-1} over the course of ten years. The High Luminosity LHC (HL-LHC) will increase the instantaneous luminosity delivered by a factor of 5 compared to the current operation period. This will impose significant technical challenges on all aspects of the ATLAS detector but particularly the Inner Detector, trigger, and data acquisition systems. In addition, many of the components of the Inner Detector are reaching the end of their designed lifetime and will need to be exchanged. As such, the Inner Detector will be entirely replaced by an all silicon tracker, known as the Inner Tracker (ITk).

The layout of the Pixel and strip detectors will be optimised for the upgrade and will extend their forward coverage. To reduce the per-pixel hit rate and explore novel techniques for dealing with the conditions in HL-LHC, an inter-experiment collaboration called RD53 has been formed. RD53 is tasked with producing a front-end readout chip to be used as part of hybrid Pixel detectors that can deal with the high multiplicity environment in the HL-LHC.

A silicon sensor, which makes up the other half of the hybrid Pixel detector, must also be designed to cope with the high fluences in HL-LHC. Significant damage will be caused by non-ionising energy loss in the sensor over its lifetime. This damage must be incorporated into the detector simulation both to predict the detector performance at specific conditions and to understand the effects of radiation damage on data taking. The implementation of radiation damage in the ATLAS simulation framework is discussed in this thesis.

Collisions produced by the HL-LHC also presents a challenge for the current track reconstruction software. High luminosity is obtained, in part, by increasing the number of interactions per bunch crossing, which in turn increases the time taken for track reconstruction. Various approaches to circumvent the strain on projected resources are being explored, including porting existing algorithms to parallel architectures. A popular algorithm used in track reconstruction, the Kalman filter, has been implemented in a neuromorphic architecture: IBM's TrueNorth. The limits of using such an architecture for tracking, as well as how its performance compares to a non-spiking Kalman filter implementation, are explored in this thesis.

Contents

Preface	iii
1 Introduction	1
1.1 Accelerators	2
1.2 Particle detectors	4
1.3 Track reconstruction	13
1.4 ATLAS upgrades for HL-LHC	15
2 Kalman filter in IBM's TrueNorth	21
2.1 Neuromorphic computing	22
2.2 IBM's TrueNorth	25
2.3 Implementation and test setup	35
2.4 Results and discussion	50
2.5 Conclusion	59
3 Radiation damage modeling in Pixel detector sensors	62
3.1 Silicon detectors	62
3.2 Energy deposition in silicon detectors	77
3.3 Radiation damage in silicon	79
3.4 Digitization in Athena	86
3.5 Digitization restructure	94
3.6 Radiation damage simulation	98
3.7 Outlook	107
4 Conclusion	108
Appendices	110
A TrueNorth	111
A.1 Complete neuron description	111
A.2 Neuron parameter search	114
B Radiation Damage modelling	116
B.1 List of variable name changes	116

Preface

The work I have undertaken for licentiate can be divided into major and minor contributions. In the first year of my degree I spent a significant amount of time working on two projects which I have decided not to include explicitly in this thesis, although both have resulted in publications. The two projects worked on in the second year form the main body of this licentiate. The publications associated with the work described here are listed at the end of the preface.

The first project I worked on explored the cluster properties of ATLAS IBL Pixel modules placed at shallow angles in a test beam. The modules were oriented in the beam such that the particles passed through $50\text{ }\mu\text{m}$ of silicon per pixel. As such, the front-end readout chips had to be tuned to low thresholds ($\sim 1000\text{ e}^-$) to record the signal, which required a custom tuning scheme. I also prepared tunings at 1500, 2000, and 3000 e^- . The modules used had broken low-voltage regulators on the front-end readout chip, which I manually bypassed. Overall, I prepared six Pixel modules for the extended layout testbeam which included bypassing the broken regulators on-chip, IV characteristics to determine safe operating parameters, and front-end configuration files.

For the testbeam itself I worked with the data acquisition team at SLAC to write a bitstream converter for the output data packet, and wrote a framework for clustering to be used in data analysis. With the rest of the extended-layout working group, I helped setup the testbeam at SLAC, took shifts monitoring data-taking, and assisted in early data analysis. However, the final analysis and write-up was performed by other members of the group. This resulted in the ATLAS note, (a), listed below.

My second project was working with the first 65 nm demonstrator produced for a front-end readout chip in HL-LHC. The chip, FE65-p2, was produced with the RD53 collaboration and allowed me to work directly with three chip designers. In that project I performed verification of the digital logic of the readout chip. This involved writing testbenches that simulated hit patterns that might be produced in the detector and testing how the chip processed them. My work in chip verification revealed discrepancies in the matching of the analogue and digital pixel matrix mapping and uncovered a couple of overlapping register definitions.

Following verification, several FE65-p2 chips were produced. FE65-p2 contains variants of a radiation-hard analogue amplifier, specifically designed to operate under the high doses received in the ITk. To test the suitability of these chip, a post-doc and I prepared six of the them to be irradiated at the LANSCE facility in Los Alamos National Lab. I performed measurements of the amplifier currents before and after irradiation as well as physically mounting passive components to each testboard. At Los Alamos I wrote a basic monitoring and control application over GPIB for the Keithley power supplies and took shifts over the course of the irradiation to remotely operate the chips from 30 m away.

However, as with the extended layout testbeam studies, the final analysis and paper (b) was performed by other members of the group, so I chose not to include it in my thesis.

About this thesis

The work included in this thesis represents two projects in which I have made a significant contribution throughout.

The IBM TrueNorth project presented a unique opportunity to work with cutting edge technology produced by industry. I worked on implementing a Kalman filter in TrueNorth with David Clark, an undergraduate at UC Berkeley, supervised by Dr. Paolo Calafiura. David was tasked with producing a numerical simulation in Python to produce toy data and simulate characteristic features of the chip. This numerical simulation provided a baseline with which to compare the TrueNorth implementation to. David and I tried several approaches to implement the Kalman filter, before settling on the one described in this thesis. Whilst the formulating of crossbars was truly a joint effort, I wrote the entirety of the code for TrueNorth as well as the top-level design of the Kalman filter. I also wrote the analysis framework, conceived and performed the tests and measurements, and wrote the proceedings for CHEP 2016 (c), in which I presented our work in a 15 minute presentation. Chapter 2 will detail the design, implementation, and performance of the Kalman filter in TrueNorth.

The second project presented in this thesis involved working with the radiation damage simulation subgroup of the PixelOffline software group. I was assigned the task of migrating a non-ionising energy loss (NIEL) damage model, written in a standalone simulation framework called Allpix, into the Athena simulation framework used by the ATLAS experiment. My task involved assessing where in the framework the simulation would be best placed, and then implementing it. During this task I noticed that large portions of the core digitization code were repeated and could be optimised for clarity and performance. I am also an active editor for the internal note (d) describing this work. Chapter 3 will detail both the implementation of a NIEL damage model in the Athena simulation framework and the restructuring of the Pixel digitization package.

List of publications

- (a) S. Viel, S. Banerjee, G. Brandt, R. Carney, et. al (2015) Performance of Silicon Pixel Detectors at Small Track Incidence Angles for the ATLAS Inner Tracker Upgrade, *ATL-INDET-PROC-2015-011* .
- (b) M. Garcia-Sciveres, R. Carney, K. Dunne, et. al (2016) Results of FE65-P2 Pixel Readout Test Chip for High Luminosity LHC Upgrades, *CERN-RD53-PROC-16-001*
- (c) R. Carney, K. Bouchard, P. Calafiura, et. al (2016) Neuromorphic Kalman filter implementation in IBM's TrueNorth *Proceedings CHEP 2016*
- (d) M. Benoit, M. Bomben, R. Carney, et. al (2017) Modeling Radiation Damage Effects for Pixel Sensors in the ATLAS Detector *ATL-COM-INDET-2017-011*

Chapter 1

Introduction

Particle physics, as a discipline, seeks to understand the fundamental constituents of matter and how they interact. In practice, particle physicists confirm the existence of particles and measure them using custom detectors. For example, in 2012 the ATLAS and CMS detectors at the Large Hadron Collider (LHC) at CERN confirmed the existence of the Higgs boson [1]. The Higgs boson is the linchpin of the most successful attempt yet to describe the building blocks of the known universe, the Standard Model. The Standard Model, see Fig. 1.1, not only

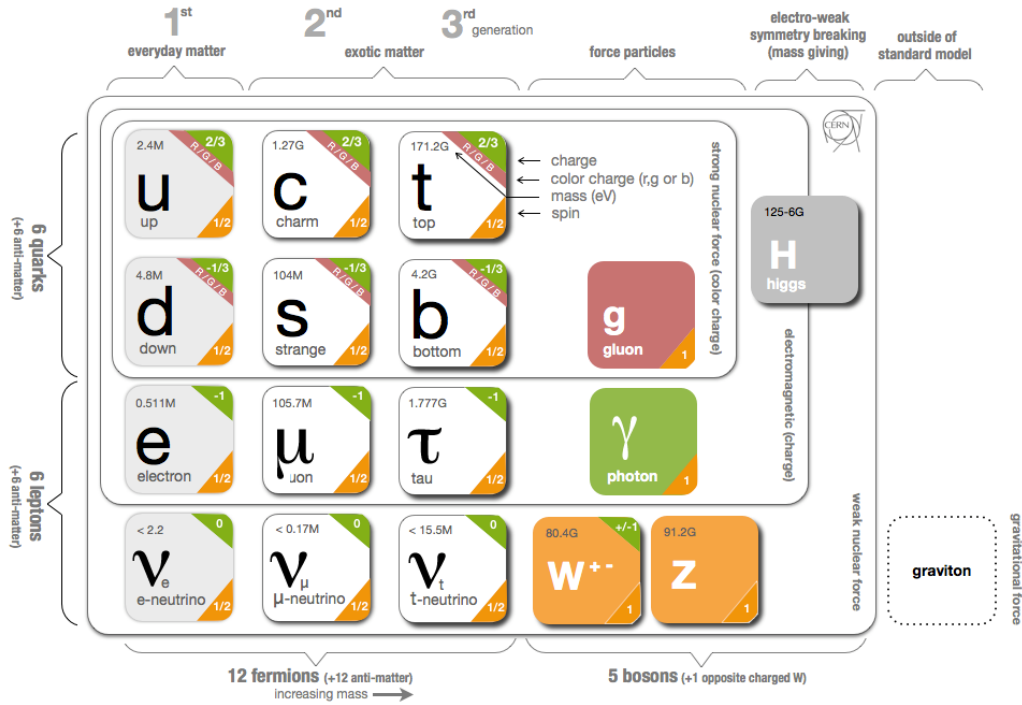


Figure 1.1: The fundamental constituents of the Standard Model [2].

predicts the fundamental matter particles, quarks and leptons, but also the mediators that allow those particles to interact, the gauge bosons, and the Higgs boson, which accounts for fundamental particles having non-zero mass and not traveling at the speed of light. Both quarks and leptons have 3 generations of particle pairs, with each generation becoming heavier than the last. All the stable matter in the universe is made from the first, and lightest, generation of these particles. If any of the second and third generation particles come into existence, they quickly decay to stable particles. The decay mechanisms, their rate, and the likelihood with which one decay will happen over another is precisely predicted by the standard model;

predictions which have been experimentally verified down to within experimental limits [3].

Elementary particles interact via four fundamental forces: electromagnetism, the strong force, the weak force, and gravity. The electromagnetic, strong, and weak forces result from the exchange of force-carrier particles, or bosons. The electromagnetic force is mediated by the photon, a massless particle, that interacts only with electrically charged particles and mediators. The strong force is mediated by massless gluons that only interact with particles that have colour charge. The strong force is responsible for binding quarks into protons and neutrons and for holding those protons and neutrons together in the nucleus. The weak force is mediated by the W and Z bosons and interacts with all particles that have a non-zero weak isospin. The weak force allows quarks to change flavours, for example in beta minus decay, a down quark can decay into an up quark along with the release of an electron and an anti-neutrino.

The matter particles shown in Fig. 1.1 also have anti-matter counterparts, so named because, despite having the same mass and spin, they have opposite fundamental charges. The W^+ and W^- bosons are each other's antiparticle and the photon and Z boson are their own antiparticle. There are 8 gluons with various combinations of colour charge that are each other's antiparticles. However, although the Standard model includes antiparticles, it does not account for the asymmetry of matter to anti-matter observed in the universe. The Standard Model's predictions agree to great precision with observation but it does not describe some key phenomena observed in nature. For example, of the four fundamental forces, all are incorporated into the Standard Model but gravity. There are theories that combine general relativity, which describes the law of gravity, and quantum field theory, which underpins the Standard Model, such as Supersymmetry (SUSY). If supersymmetric particles exist, they should interact with matter in a specific way that is detectable.

The Standard Model can also not account for the makeup of 95% of the universe. Dark matter, making up around 27% of the universe, has mass but does not interact with the electromagnetic force and is only observable by inference. Some theories predict that dark matter couples to regular matter, and that it is light enough that it could be produced in particle colliders.

To verify the predictions of the Standard Model, and now to answer the question of what physics lies beyond it, particles are collided at high energies in purpose-built accelerators. To probe the nature of matter at a fundamental level a particle collider is used in much the same way as a microscope. Instead of low energy photons interacting with matter and being observed through an eye-piece, high-energy particles collide and the remnants of their interaction are recorded in a detector. If particles are accelerated to high enough energies, heavier particles can be produced from their interaction. In this manner, the Standard Model of particle physics was experimentally confirmed, culminating in the discovery of the Higgs boson in 2012 by two experiments at the LHC.

This chapter will briefly introduce the instrumentation, data acquisition, and processing that goes into obtaining data from one of the two multi-purpose detectors at the LHC, ATLAS. The chapter will conclude with a brief discussion in upgrading specific parts of the detector for planned accelerator upgrades.

1.1 Accelerators

Many of the proposed solutions to questions about physics beyond the Standard Model require accelerating particles to high energies. To accelerate particles an oscillating electric field produced from a radio-frequency (RF) cavity is used. The cavity is designed to achieve resonant frequency and hence produce an oscillating electric field with a large amplitude. As such par-

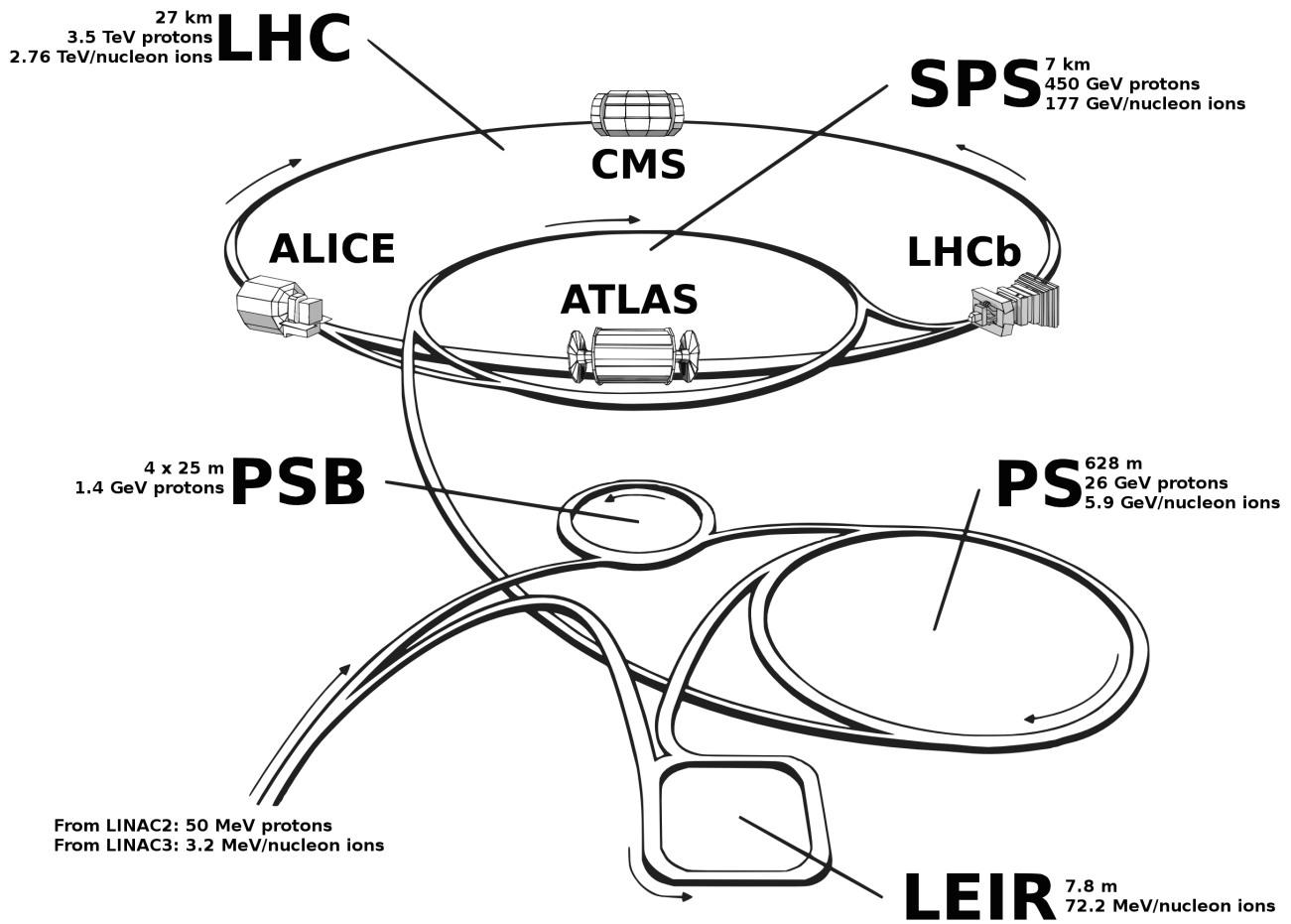


Figure 1.2: The LHC accelerator complex at CERN. High energy particles, protons or lead ions, are accelerated in different stages of the complex before being injected into intersecting storage rings in the LHC [4].

ticles are not accelerated in a continuous stream, but rather in small clusters called bunches. The bunches are injected into the accelerator just ahead of the peak of the electric field such that slower particles in the bunch are accelerated more than faster particles. This has the effect of compressing the bunches and homogenising the energy across each bunch.

The LHC is fed by several smaller accelerators which accelerate protons to 450 GeV, before they are accelerated by the RF cavities in the LHC. A cartoon of the CERN accelerator complex that feeds the LHC is shown in Fig. 1.2. Protons are first accelerated in a linear cavity accelerator, the LINAC2. They then are injected into a series of synchrotrons: the Proton Synchrotron Booster (PSB), Proton Synchrotron (PS), and Super Proton Synchrotron (SPS). Synchrotrons are circular accelerators that use synchronously ramped magnetic fields which increase with each lap made by the bunches. As the bunches become more energetic, more energy is needed to bend them.

On leaving the SPS, protons have an energy of 450 GeV. From there the protons are transferred to two beam pipes that run through the LHC until they reach the desired energy. In Run 2, each proton beam reaches 6.5 TeV, leading to 13 TeV centre-of-mass collisions which have almost twice as much energy as LHC Run 1, and seven times as much energy as the most energetic collisions at the Tevatron. One beam pipe circulates protons in a clockwise direction, the other in an anti-clockwise direction. The LHC then stores and collides the two proton beams for around 9 hours.

When the proton bunches in the LHC reach their designated energy, they are collided in specific straight portions of the accelerator. Occupying four of these sites are particle detectors, ATLAS, ALICE, CMS, and LHCb. At the interaction point, magnets are used to focus the bunches to increase the interaction cross-section and so maximise the number of collisions per bunch crossing. However, only a small percentage of the protons in each beam will actually collide. For example, in LHC Run 2, the number of interactions per bunch crossing by the end of 2016 was, on average, 25 for bunches of 1×10^{11} protons [5] [6]. The mean number of inelastic collisions per bunch crossing is also known as the pile-up, μ .

Maximising the number of interactions, or events, is particularly important for particles with a small production cross-section, σ_p ¹. The instantaneous luminosity provides a measurement for the rate of collisions divided by the production cross-section. A more commonly used measurement is the luminosity integrated with respect to time, which is usually presented in units of inverse femtobarns ($\text{fb}^{-1} = 100 \text{ fm}^{-2}$). Luminosity should not be confused with the energy of the collisions but both need to be large to produce rare processes with heavy particles in the LHC.

1.2 Particle detectors

Accelerating particles to high enough energies and luminosities is not the only challenge in particle physics. Detecting techniques and instruments must also be custom designed and produced for each experiment. Designing a particle detector requires an understanding of how particles interact with matter and on what timescales. Modern particle detectors reconstruct the presence of decay products of collisions using a combination of energy, momentum, and charge measurements which help to both identify the decay products and trace back their path through the detector to the interaction point.

In Fig. 1.3 the interactions of some collision fragments are shown for a generic general-purpose detector. Particles radiate out from the interaction point in the centre of the image. They

¹The production cross-section refers to the probability that two particles will react to produce a given product.

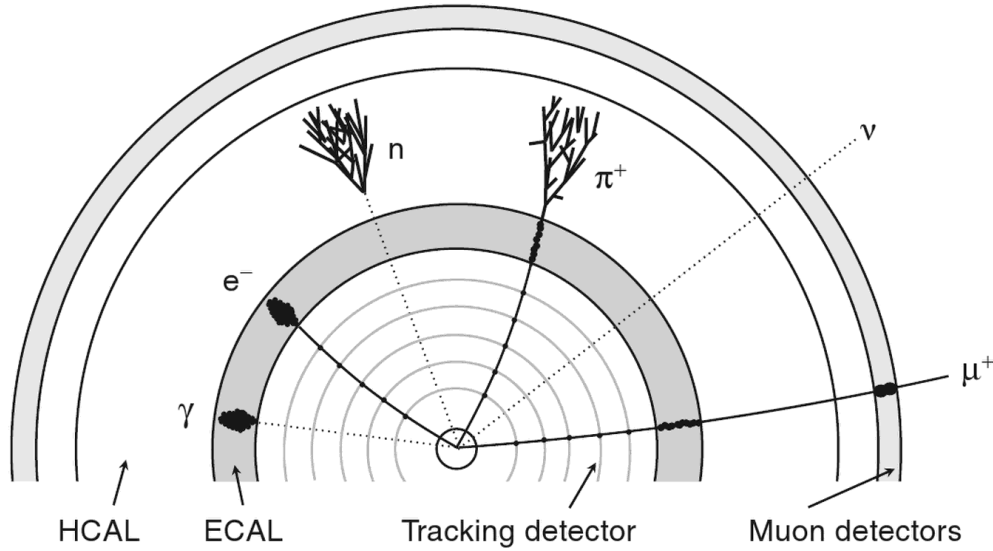


Figure 1.3: A generic general-purpose detector showing concentric layers of subdetectors with example interactions from various particles that could be produced in a proton-proton collision [7].

first pass through a multi-layer tracking detector and, if the particles have an electric charge, deposit a small amount of energy in each tracking layer. Tracking layers are usually ionisation chambers, where the energy deposited by the particle liberates charge carriers that create a measurable current which shows the detector has been hit. The hits can be used to reconstruct the trajectory of the particle by connecting the dots between layers. The procedure for track reconstruction in the ATLAS detector will be summarised in Section 1.3.

To measure the momentum of the particles, a strong magnetic field is applied through the extent of the tracker. Charged particles moving in a magnetic field will describe a helix. By finding the radius of curvature of the helix, and knowing the magnetic field strength, the momentum of the particle transverse to the field can be measured. The direction of curvature reveals the charge of the particle.

After leaving the tracker the particles are stopped in the detector volume by calorimeters. Calorimeters are built from a material with high atomic number, which will cause the particle to lose energy, called an absorber and some sampling device that measures the energy loss.

The first calorimeter in the particle's path is an electromagnetic (EM) calorimeter. The EM calorimeter is made of a material with a low radiation length, the length over which a particle's energy reduces by a factor of e^{-1} due to electromagnetic interactions, and is used to stop electrons, positrons, and photons. All other charged particles will also deposit energy but their energy loss is small relative to their momentum and therefore they will penetrate the EM calorimeter to the hadronic calorimeter. The hadronic calorimeter is comprised of an absorber with a small nuclear interaction length, the length over which a particle's energy reduces by a factor of e^{-1} due to inelastic nuclear collisions. The hadronic calorimeter is designed to have enough material such that the most energetic particles are stopped within it.

However, two particles still escape the detector volume: neutrinos and muons. Neutrinos only interact via the weak force and their presence is deduced through missing transverse energy. Muons are measured in another tracker in the last layer of the detector. Since muons are the only charged particles to make it through the calorimeter, their known mass and measured momentum can be used to measure their energy. As such, a magnet system is also placed around the muon tracker.

In this section the ATLAS detector's subdetectors are briefly described and motivated for their use in particle identification and reconstruction.

1.2.1 The ATLAS experiment

The ATLAS experiment, shown in Fig. 1.4, consists of subsystems that can be grouped into the tracker, calorimeters, and muon detectors. Particles fragments produced in proton-proton

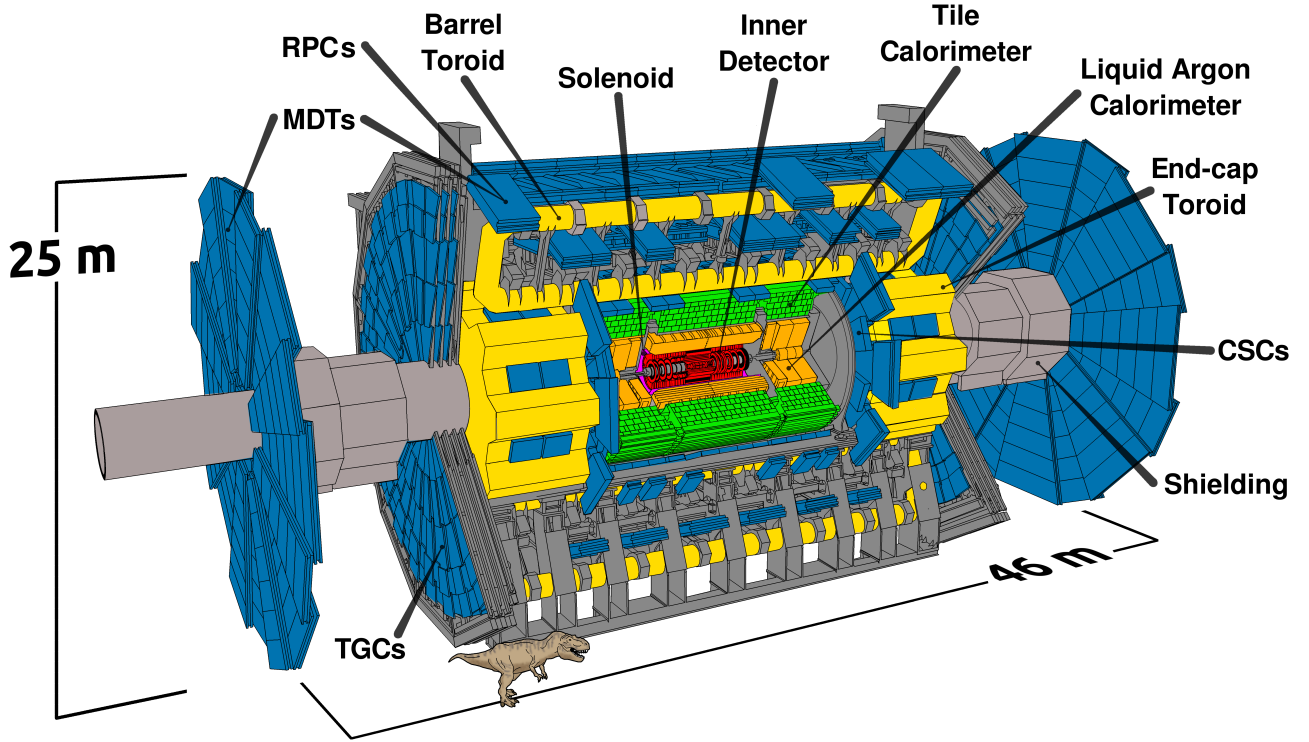


Figure 1.4: The ATLAS detector with T-Rex for scale [4]. The beam pipe runs through the axis of the detector and the particles collide within.

collisions in the beam pipe, that runs through the centre of ATLAS experiment, are identified by their energy, momentum, and the way that they interact with each subdetector. The ATLAS experiment is multi-purpose in that it is not designed to measure one specific particle or decay chain but is able to reconstruct any event produced by a proton-proton interaction.

The ATLAS detector is comprised of concentric, cylindrical layers of subdetectors with the beam pipe running through the central axis of the cylinders. The coordinate system used by the ATLAS detector is similar to a cylindrical coordinate base. The beam pipe, and hence the interaction point, is aligned with the z -axis shown in Fig. 1.5. The concentric cylindrical layers of the detectors lie at radius, R , from the z -axis and at azimuthal angle ϕ . The polar angle, θ is not used in the ATLAS coordinate system. Instead a variable called the pseudorapidity is defined as:

$$\eta = -\ln \left(\tan \left(\frac{\theta}{2} \right) \right) \quad (1.1)$$

This quantity is useful in hadronic physics reconstruction due to differences in the pseudorapidity, $\eta_1 - \eta_2$, being invariant under Lorentz boosts along the z -axis. This helps with analysing the directional distribution of particles. Very forward regions of the detector have a high eta value and those near the centre of the detector have a low eta value.

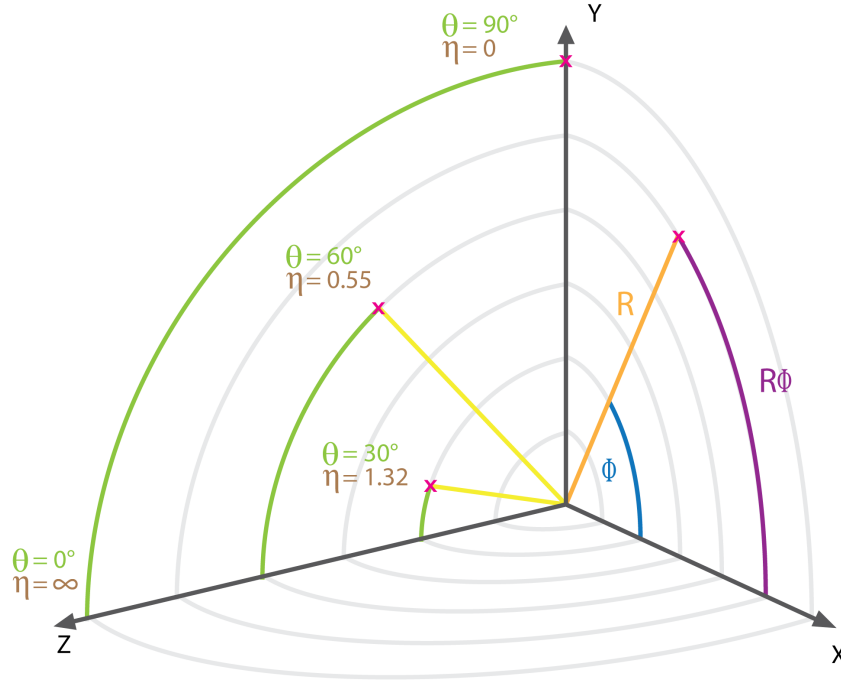


Figure 1.5: The coordinate system used by the ATLAS experiment, where the beam pipe that runs through the centre of the ATLAS detector lies along the z -axis [8].

Inner Detector

In the ATLAS detector the tracker is also known as the Inner Detector and comprises three subdetectors. Each subdetector is arranged in multiple layers in two possible structures: the barrel and endcap. The detector barrel is placed in concentric, cylindrical layers and has full coverage in ϕ and extends out to $|\eta| = 2.5$. The endcaps, also known as discs, essentially cap these barrels at both ends, as can be seen in Fig. 1.6. The innermost four layers of the ATLAS Inner Detector are hybrid detectors of pixelated silicon diodes and front-end readout chips, electrically connected by lead solder bumps, see Fig. 1.7b. These pixelated layers are jointly known as the Pixel detector, shown in Fig. 1.7a. Whilst only spanning ~ 1.44 m in length and taking up 0.0075 % of the volume of ATLAS, the Pixel detector accounts for ~ 92 % of the 100 M readout channels in the entire detector. The innermost layer of the Pixel detector, also known as the Insertable B-Layer (IBL), has $50 \times 250 \mu\text{m}^2$ pixels, the outer 3 layers and endcap modules have $50 \times 400 \mu\text{m}^2$ pixels. This high spatial resolution allows the detector to discriminate vertices.

The Pixel detector not only records which pixels were hit in an event but also the amount of charge deposited by the particle. This is done with a charge sensitive amplifier that integrates the charge induced in the pixel sensor and records how long the recorded charge exceeds some pre-tuned threshold. This process effectively digitizes the analogue charge collected in the sensor and produces a value called the Time-Over-Threshold (ToT). The ToT is used, along with which pixel was hit in the interaction, to reconstruct the tracks produced. This will be covered further in section 1.3. The Pixel detector sensor will be covered in more detail in Chapter 3.

The Pixel detector is followed by four layers of back-to-back silicon strip detectors, known as the SemiConductor Tracker (SCT). Each sensor is placed at a small stereo angle which allows space points to be constructed from two back-to-back sensor hits.

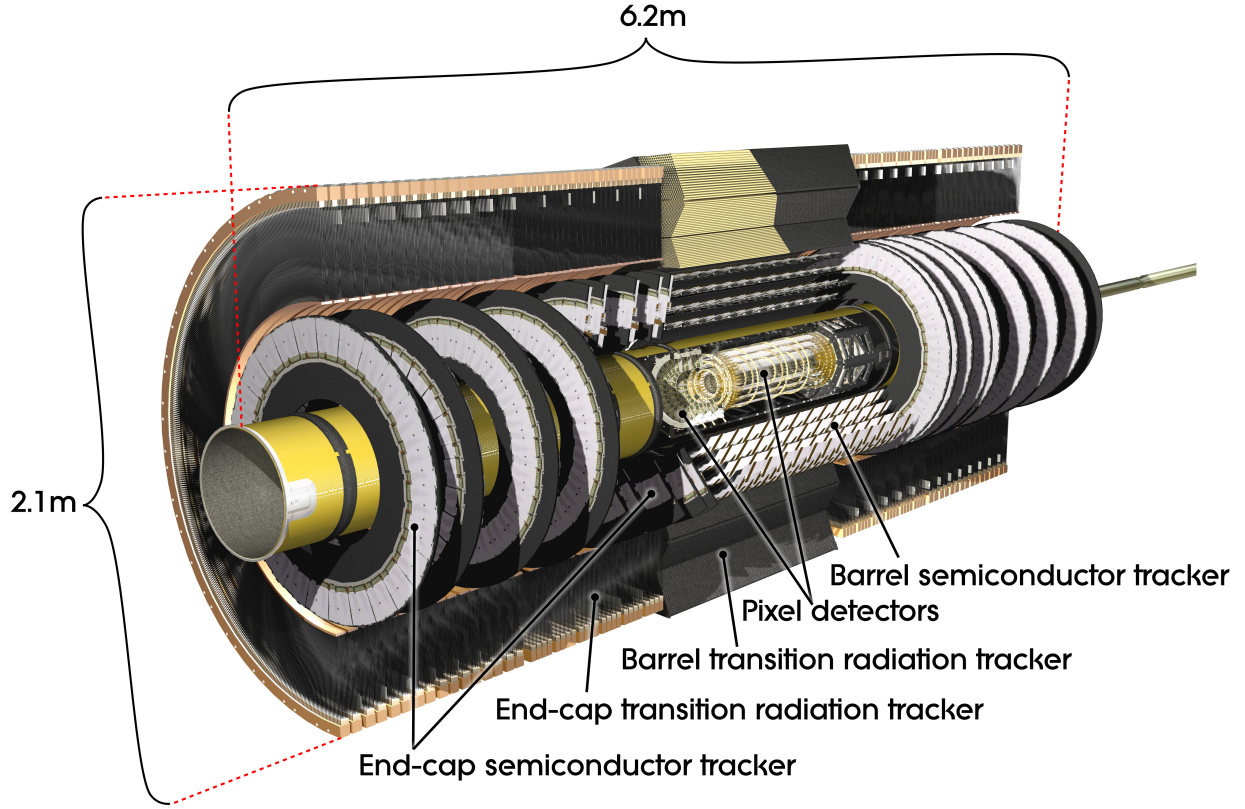
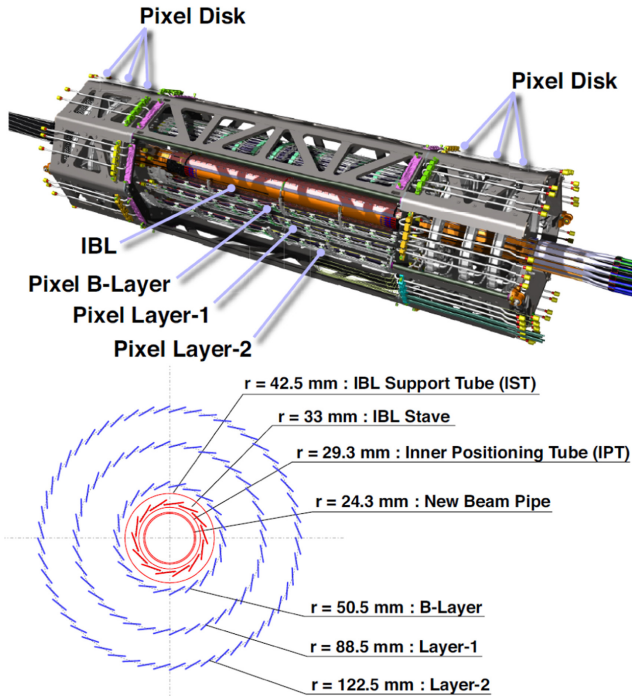
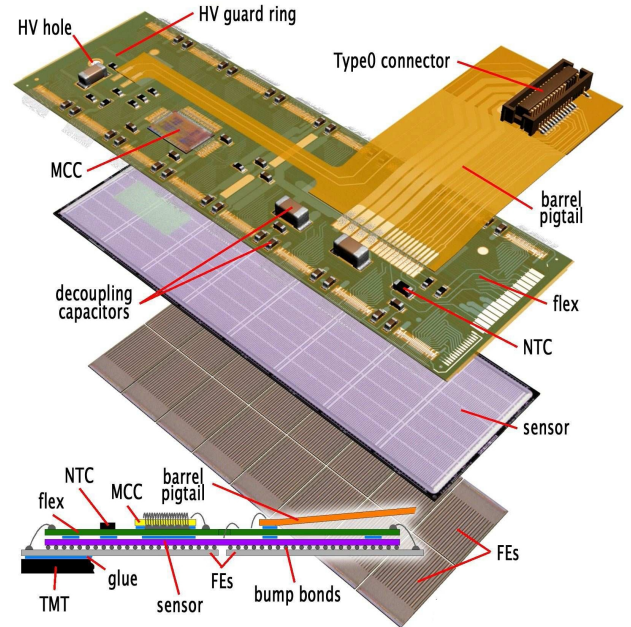


Figure 1.6: The ATLAS Inner detector, which consists of two silicon trackers: the Pixel and SCT detectors, and the Transition Radiation Tracker [9].



(a) A 3D model of the Pixel detector and a cross-sectional view of each layer. The individual staves making up the detector barrel are clearly visible in the cross-section [10].



(b) A 3D model of a hybrid Pixel module. Sixteen individual front-end readout chips (FE) are electrically connected to a pixelated silicon sensor by bump bonds. The FEs are then wire-bonded to a flexible PCB glued atop the sensor, from which data is transported off-detector [11].

Figure 1.7: The Pixel detector layout and modules.

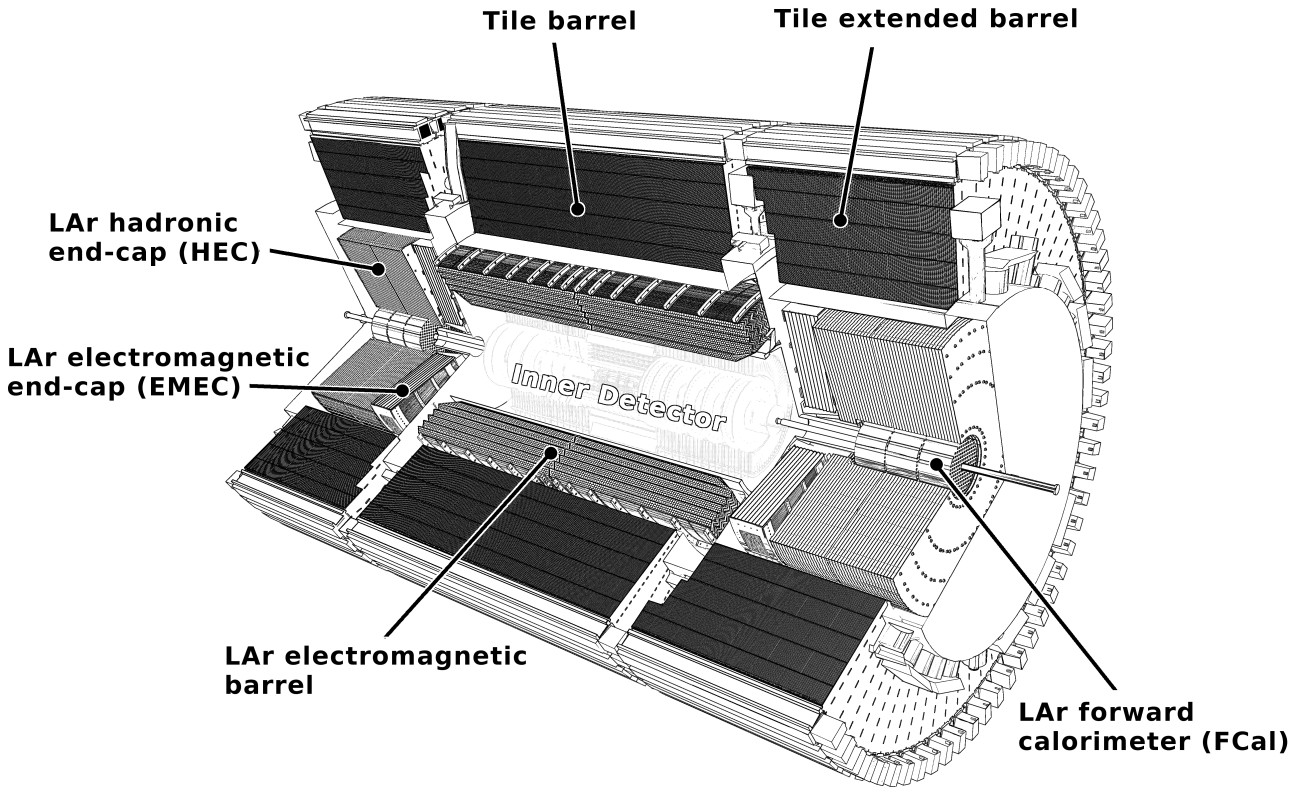


Figure 1.8: The ATLAS detector calorimeters [4].

Following the SCT is the Transition Radiation Tracker (TRT). The TRT is comprised of $\sim 350,000$ individual proportional drift tubes, or straws, with an anode running through their centre and an outer shell cathode. The TRT straws are flushed with a xenon-gas mixture which, paired with the large potential difference between the straw casing and the anode, causes cascades of electrons to drift towards the anode.

The ToT is also recorded for the TRT, but in this case only to flag large charge deposits indicative of an electron. The TRT is able to discriminate between electrons and pions traversing the detector due to materials of different dielectric constants sandwiched between the straws. As particles with a high Lorentz factor pass through these materials, transition radiation is emitted in the form of x-rays which interact with Xe gas in the straws producing large signals. Such a setup allows the TRT to identify electrons, as due to their low mass, electrons are the only charged particles with Lorentz boost above the transition radiation emission threshold [12]. The Inner Detector is surrounded by a superconducting solenoid magnet that provides a 2 T magnetic field which causes the particles trajectories to curve as they pass through the detector.

Calorimetry

The tracker is made of material that has a large radiation and nuclear interaction length, which means that particles traversing it will only lose a small fraction of their kinetic energy to electromagnetic or hadronic interactions. This is done to reduce multiple scattering which assists with track reconstruction. The opposite is true for the calorimeters that lie just outside the Inner Detector. To establish the energy of the particles produced, layers of calorimeters are used to stop the particles completely, which causes all of their energy to be deposited in the detector volume and stop particles reaching the muon detector. Both the barrel and endcap regions of the ATLAS detector have calorimeters, see Fig. 1.8.

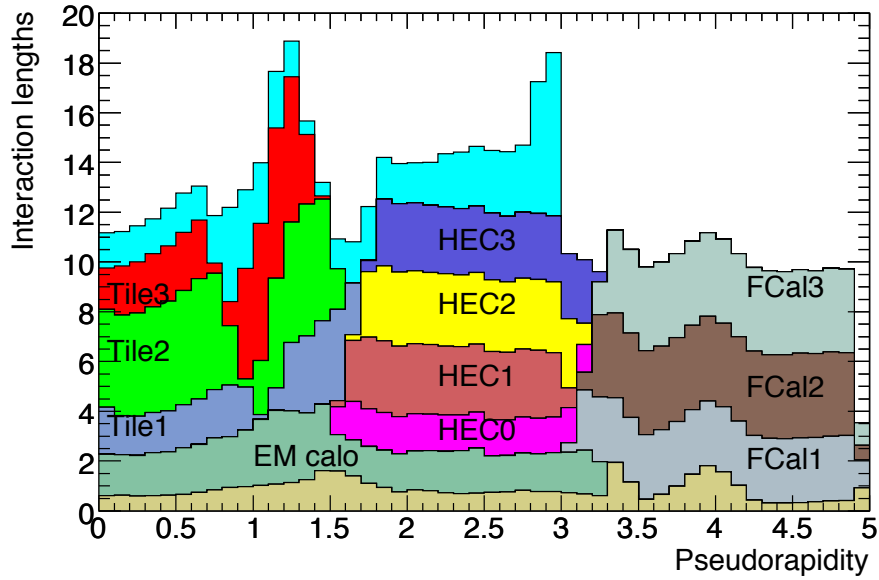


Figure 1.9: A stacked histogram of the number of interaction lengths in each part of the ATLAS detector. From the bottom (unlabelled) is the Inner Detector, followed by the EM and hadronic calorimeters, and finally (unlabelled) the muon detector [13].

- The **Electromagnetic calorimeter** is made of a material with small radiation length, lead, which causes the particles traversing it to shower in electromagnetic cascades. In between the lead are segments of liquid argon (LAr) that act as an ionisation centre and sample the energy of the particle after it has passed through the lead absorber. The segments of the EM calorimeter are tiled like an accordion to provide full coverage without gaps. This fine segmentation allows for greater spatial precision when reconstructing data. The EM calorimeter effectively samples the energy of the particles in segments, with the lead sheets causing showers and the LAr sampling the energy of the particle.
- **Hadronic calorimeter** Low-mass particles like electrons and photons will be completely stopped by the EM calorimeter, as can be seen in Fig. 1.3. Hadrons will lose some of their energy in the EM calorimeter and but will be fully stopped in the hadronic calorimeter. There are two types of hadronic calorimeter in ATLAS: the Tile and LAr end-cap (HEC). The Tile calorimeter is made of layers of iron absorbers between layers of scintillators. The scintillators are lined with fibre-optic cables connected to photomultiplier tubes that transduce the signal into a measurable current. As the name suggests the LAr endcap sits in the high η region of the detector, after the LAr EM endcap calorimeter. LAr is used in this region as a collection medium instead of scintillator due to the higher particle flux which would damage the scintillator and lead to a much shorter lifetime. The absorber in the endcap hadronic calorimeter is copper rather than lead, as it has a smaller nuclear interaction length per unit mass. The nuclear interaction length is shown for the entire detector in Fig. 1.9. This clearly shows that, despite the similarity in their construction, the LAr HEC calorimeter has many more interaction lengths in its volume than the EM calorimeter.

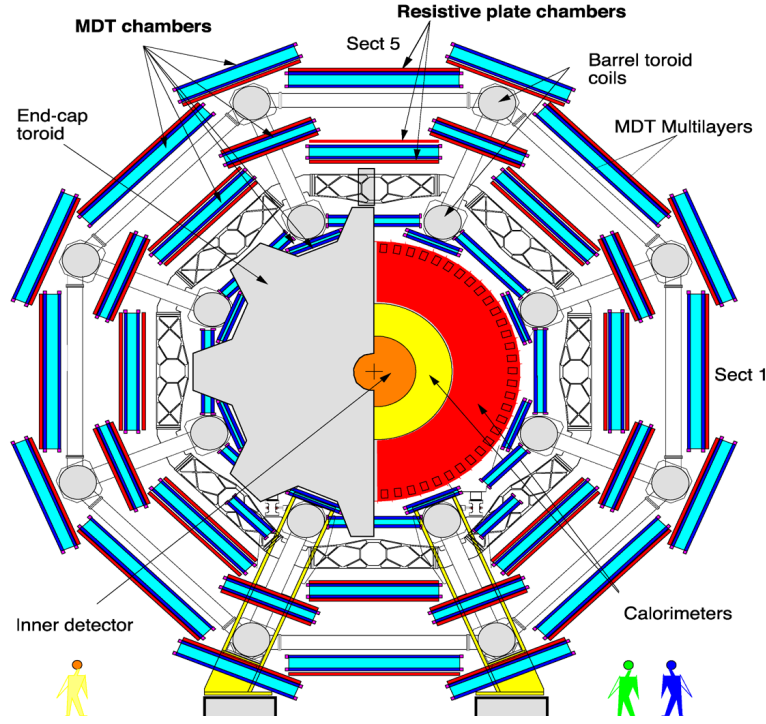


Figure 1.10: A side view of the ATLAS detector with the MDT and RPC of the muon detector clearly visible [15].

Muon detectors

Muons penetrate both calorimeters and are not stopped in their volume. This is because they do not interact hadronically and lose very little energy, compared to the electron, when interacting electromagnetically due to their high mass. As such an entire detector, solely for identifying muons is placed outside the calorimeter. Many analyses searching for beyond-the-standard-model processes use missing transverse energy and momentum as a flag that an interesting process is happening, thus if a muon escapes the calorimeters it must be accounted for. The muon detector is also part of the ATLAS trigger system so it can trigger, for example, on promising channels such as $H \rightarrow 4\mu$ decays [14].

The muon detector consist of five elements: a magnet system, two types of precision measurement detector, and two types of detector used for triggering, see Fig. 1.10.

- **Toroidal magnets**, or more specifically three 8-coil air-core toroid magnets, one to cover the barrel region and two in the forward regions. These provide a magnetic field to bend to muons and enable a momentum measurement of the particles before they leave ATLAS. The toroid compromises on magnetic-field strength to minimise the amount of material in it, which in turn reduces multiple scattering.
- **Monitored Drift Tubes (MDT) chambers** provide precise tracking information, down to around $80 \mu\text{m}$, using Argon-filled drift tubes. An alignment system built with computer-vision software and lasers to track the detector position ensures the MDT's position is known to within $40 \mu\text{m}$ [16].
- **Cathode Strip Chambers (CSC)** track muons in the forward region of the detector, due to their higher rate capability and time resolution when compared with the MDT, which would not cope with the higher particle fluence [17]. The CSC is a multi-wire proportional chamber with segmented strips which allows precision position measurements down to $40 \mu\text{m}$.

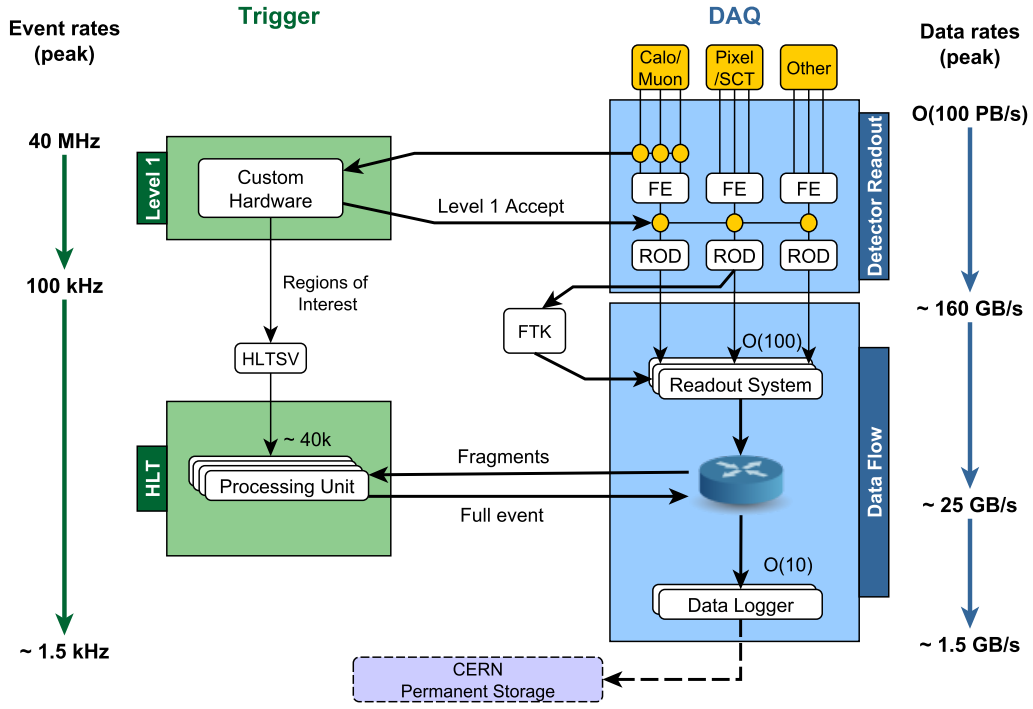


Figure 1.11: The ATLAS trigger and data acquisition system [18].

- **Resistive Plate Chambers (RPC)** provide the first-level muon trigger and have a reduced position measurement, when compared with the MDT, in the barrel region of the detector. They consist of ionisation chambers filled with an ethane-butane based gas mixture. The high electric field and relatively short drift distance give a timing jitter of less than 2 ns [14].
- **Thin Gap Chambers (TGC)** also act as muon triggers for the forward region and provides a second position measurement to complement the CSC. The TGC's are also multi-wire proportional chambers, the thin gap refers to the separation of the anode and cathode [13].

The timing and position information from the RPC and TGC are passed into the trigger system which decides which events to store and which to flush.

Data acquisition and trigger

There are proton bunches crossing every 25 ns in the ATLAS detector. In Run 2, ATLAS can produce 100 PB/s of data, so storing and analysing every event would be impossible. However, a large fraction of the events formed by proton-proton collisions in the ATLAS detector are from QCD jet production and other well understood processes. These backgrounds are not of interest to the scientific community, so a trigger is used to only record events of interest. Therefore a trigger system with the dual purpose of selecting events that contain interesting physics processes, whilst reducing the amount of data recorded by 4 orders of magnitude, is implemented. This is implemented in a three layer trigger system, see Fig. 1.11.

1. High energy muons, able to penetrate through the detector to the muon detector, indicate that a high energy event has occurred and supply information to the first level of triggers from the RPC and the TGC. Hits from all calorimeters, with a reduced granularity are also supplied. The data is held in buffers, on-detector, until the level-1 trigger decision

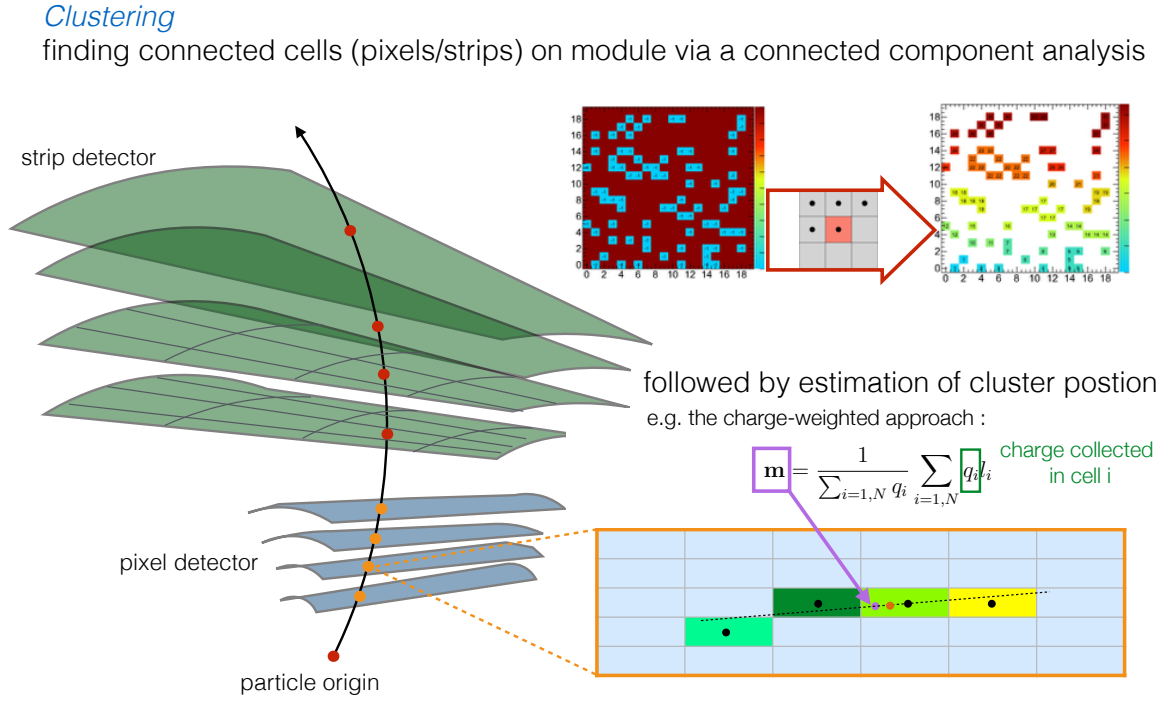


Figure 1.12: An illustration of the pixel clustering, track seeding, and propagation as part of the track reconstruction process in the silicon trackers [20].

is reached by the Central Trigger Processor (CTP). When the CTP decides to trigger an event the criteria for making that decision as well as Regions of Interest (RoI), are supplied to the level-2 trigger. The maximum trigger latency for the level-1 trigger is $2.5 \mu\text{s}$ and reduces the data rate from 40 MHz to $\sim 75 \text{ kHz}$ [19].

2. The level-2 trigger uses the full resolution of the calorimeter and muon detector data to further select interesting events along the RoI supplied by the CTP. The large amount of data present in the Inner Detector could not be used in the level-1 or 2 trigger decision, as it could not be processed fast enough. However, as of Run 2 the Fast Track trigger (FTk) has been included in level-2 trigger, see Fig. 1.11. FTk uses associative memories to store pattern-banks of hits in different layers of the Inner Detector and matches those patterns to data read out from the the SCT and Pixel detector following a level-1 trigger. The FTk is able to match patterns in $\sim 25 \mu\text{s}$, before passing that information to the rest of the level-2 trigger [19]. The level-2 trigger reduces the data rate from $\sim 75 \text{ kHz}$ down to 3.5 kHz.
3. The final stage of triggering is done by the event filter that uses offline analysis techniques to reduce the data rate to $\sim 200 \text{ Hz}$.

Once data has passed through all three stages of trigger it is stored to be reassembled later in a process called reconstruction. Reconstructed data is composed of variables that physicists can work with directly, including tracks, particle identification, and kinematics.

1.3 Track reconstruction

Once an event is deemed interesting enough to be stored by the trigger, it is written out to tape and made available ‘offline’ to physicists for reconstruction. Reconstruction involves translating raw detector information into objects useful for physics analysis. The offline software frame-

work used for data reconstruction in ATLAS is called Athena [21], and is described in section 3.4.1. Aside from reconstructing the raw data, Athena also includes a simulation framework that provides a way to produce simulated detector data to verify both our understanding of the physics interactions in ATLAS and also how the detector interacts with those particles. Having the physics and detector simulation feed directly into the reconstruction framework allows direct data and simulation comparison.

The simulated and collected data are reconstructed in a sequence of steps, listed below. Depending on the specific physics channel that is being analysed, some steps may appear in a different order, for example tracks in the muon detector might be used to find tracks in the Inner Detector, or vice versa [22].

- **Creating space-points from silicon detectors** is the first step in reconstructing particle tracks from Inner Detector hits. First, pixel clusters are created by clustering hit pixels together and taking into account particular attributes such as missing hits or delta rays². A neural network is then used to identify and flag clusters that were merged during the clustering process but that actually come from distinct tracks [23]. The centroids of these clusters, along with the layer of the detector, are then used to create a 3D space-point. The cluster centroid is formed by using pixel hits weighted by ToT. The SCT space-point is made by pairing up hits on back-to-back modules aligned at a stereo angle to each other and the layer they appear in.
- **Seed building** involves connecting space-points in different layers into triplets. These seeds can be built from three hits in the Pixel detector (PPP), three hits in the SCT layer (SSS), or a mix of the two (SPP). The seeds are built by connecting hits in similar regions and throwing away seeds that do not make specific cuts on transverse momentum and distance from the interaction point. The number of potential seeds is further reduced by confirming the seed with the existence of a hit in a fourth layer of the Inner Detector.
- **Track building** uses an adaption of a linear filter, called the combinatorial Kalman filter, that starts from a seed and adds one hit after another to it until all layers in the Inner Detector have been considered. Every seed leads to at least one hit collection, which is a track candidate. This process produces multiple track candidates that can share seeds or produce ‘fake tracks’, which appear to exhibit the properties of a track but are unphysical in some way. There is a compromise made between cuts that decrease the number of fake tracks and that keep a good reconstruction efficiency.
- **Ambiguity solving** eliminates multiple tracks that share near-duplicate information and fake tracks that are formed by valid seeds but that are not indicative of an underlying track. Ambiguity solving uses a scoring scheme that applies positive scores for unshared hits and good fit quality, and negative scores for holes, or missing hits, in layers and shared hits, and favours high transverse momentum tracks. At the end of the ambiguity solving process only tracks that pass a threshold for this scoring scheme will remain.
- **Tracks are then extended into the TRT** which increases the momentum resolution for the Inner Detector given its long lever arm [24]. TRT data is added to existing silicon hits in two ways: either after a silicon-track has been produced or before, these methods are known as the inside-out or outside-in method, respectively. The inside-out method uses a Kalman fitting-smoothing technique to extrapolate the silicon track to define a road through the TRT in which hits are added [25]. The outside-in method first transforms the TRT data using the Hough transform [26]. In Hough space intersecting lines represent space-points in real space belonging to the same track. Once the TRT hits have been

²Delta rays are knock-on electrons, that leave characteristic ionisation tracks branching from the initial track.

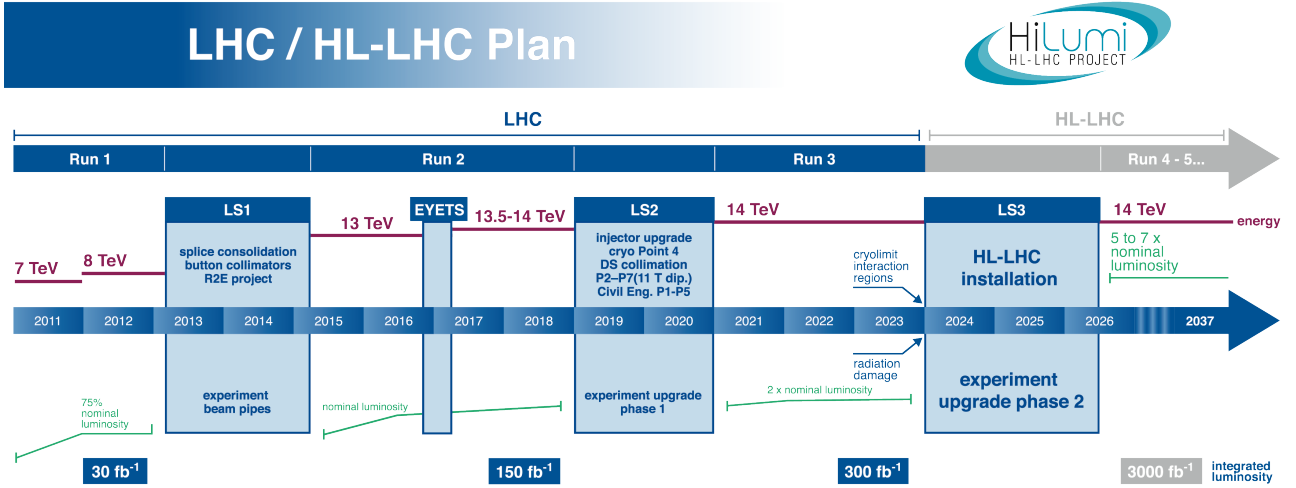


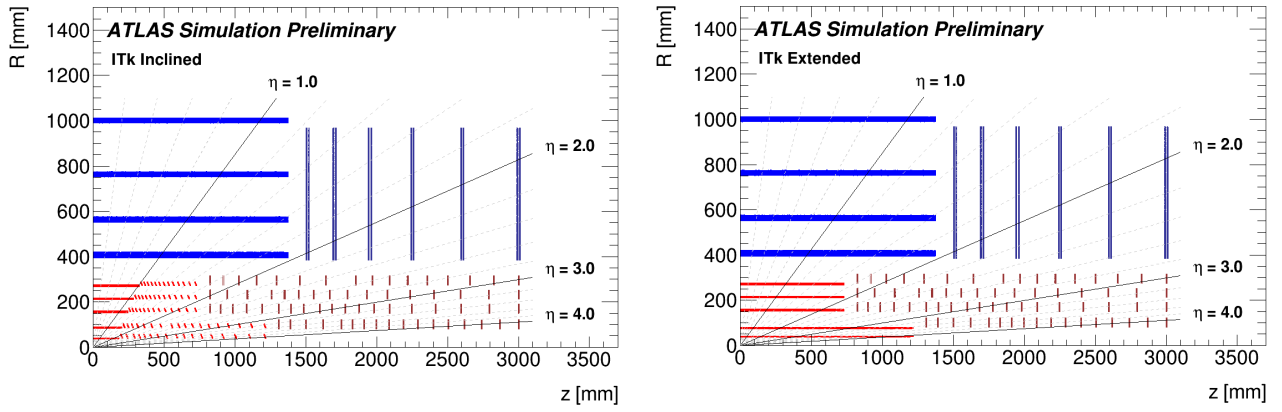
Figure 1.13: The LHC Run and Shutdown plan from the first data-taking Run up to HL-LHC [27].

clustered they are extrapolated into the Pixel and SCT layers.

- **Matching tracks with calorimeter hits** is performed after track reconstruction. Some tracks are extrapolated to different sampling layers of the calorimeter and calorimeter cells containing an energy deposit are collected. Tracks in the muon detector may also be extrapolated to the calorimeter in this way. A similar method is used to extrapolate Inner Detector tracks to a given vertex.
- **Muon tracks** are either added to the existing reconstructed track from the Inner Detector (known as a combined muon), or are built in the muon detector and used to identify Inner Detector tracks (tagged muons). Building a muon track is simpler than for the Inner Detector because of how many less hits the muon detector receives in comparison.

1.4 ATLAS upgrades for HL-LHC

Around 2027 the LHC will enter a high luminosity phase which will deliver an integrated luminosity of 3000 fb^{-1} to the ATLAS detector over the course of ten years. The present ATLAS detector will run until the end of 2022 with a projected 300 fb^{-1} integrated luminosity, see Fig. 1.13. The High Luminosity LHC (HL-LHC) will provide enough data to probe the multi-TeV energy range for new physics searches and study the properties of the Higgs boson couplings in more detail. The HL-LHC will increase the instantaneous luminosity by a factor of 5 compared to Run 2, which is the current Run ongoing in 2017. This increase imposes significant technical challenges on all aspects of the ATLAS detector but particularly the Inner Detector, trigger, and data acquisition systems. Many of the components in the Inner Detector are reaching the end of their designed lifetime and will need to be replaced. At HL-LHC's instantaneous luminosity, the TRT and parts of the Pixel detector's hit occupancy would saturate, rendering them inoperable. Also, as the TRT ages straws have begun to leak and so are filled with Argon which removes their ability to detect transition radiation [24]. As such the Inner Detector will be entirely replaced by an all-silicon tracker for HL-LHC, known as the Inner Tracker (ITk). HL-LHC also serves as an opportunity to upgrade the detector with technology that wasn't available or feasible when it was originally designed and push for creative solutions to the



(a) The inclined layout. All five pixel layers have inclined pixel modules following the barrels. The inclined inner-two layers extend outwards to $\sim \eta = 4$.

(b) The extended barrel layout. The two innermost pixel barrel layers are extended out to $\sim \eta = 4$. The outermost 3 layers are extended to $\sim \eta = 2$.

Figure 1.14: The two ITk Pixel layouts under consideration for HL-LHC [29]. The layers shown below $R = 300$ are Pixel layers, those above are Strip layers.

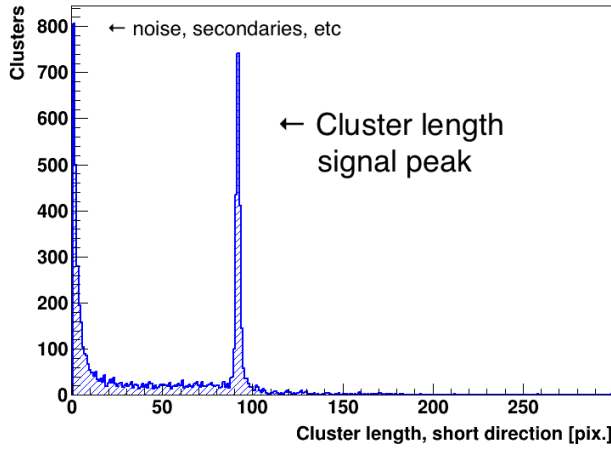
new challenges imposed by the conditions. A few upgrade projects for ATLAS HL-LHC are discussed below, a scoping document describing the complete changes ATLAS will undergo for HL-LHC is available at [28].

1.4.1 ITk layout studies

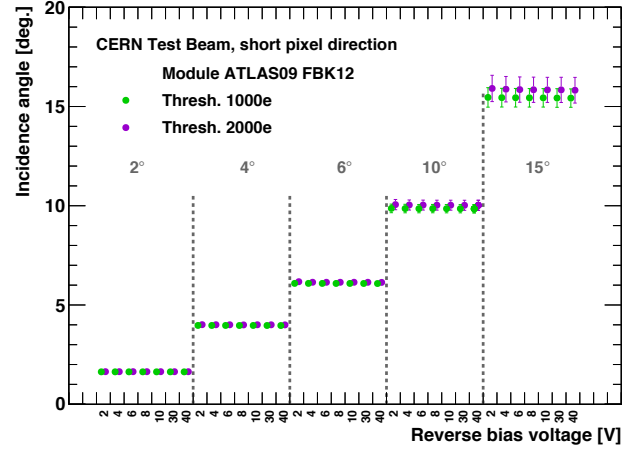
In ITk, the number of barrel and endcap layers, as well as their layout, is an ongoing study in the ATLAS collaboration. Some of the requirements for the new layout include: reducing the amount of material in the tracking volume to minimise multiple scattering and aid reconstruction, and to extend tracking coverage out to $\eta = 4$. This latter condition will provide improved sensitivity for specific physics channels [29]. There will be five Pixel layers and four Strips layers in the ITk but, at the time of writing, there are two layout possibilities for the Pixel layers, shown in Fig. 1.14b. Both layouts pose different technical challenges and module support structures. The major concepts of both are discussed below.

Extended layout

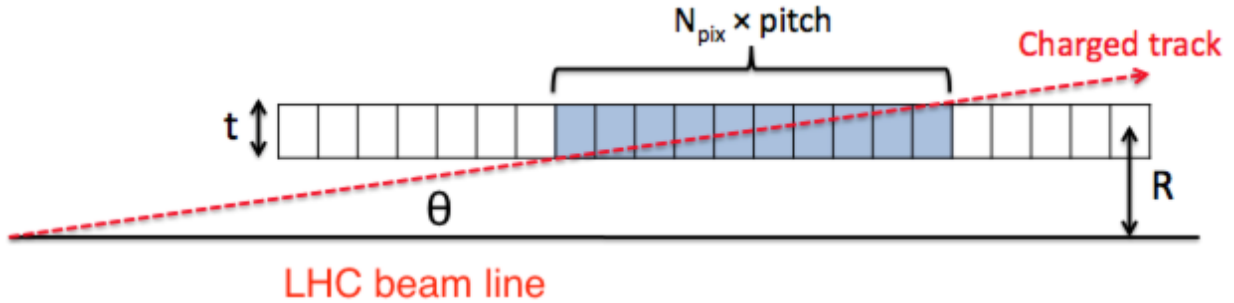
The extended layout refers to the two innermost barrel layers extending out $\eta = 3.5, 4.0$, respectively. The outermost three layers extend to $\eta = 2$ on average, see Fig. 1.14b. Extending the barrels means there is a near-parallel placement of modules with respect to high- η tracks so particles cross the sensors at small incident angles leaving a long cluster of hits, see Fig. 1.15c. Long clusters can help with the rejection of fake tracks by cutting off tracks containing clusters that are too small for a given η . Proof-of-concept measurements were made using Pixel modules from the innermost barrel layer, placed in testbeams at precise angles between 2° and 15° , which showed that there is a narrow and distinct peak in cluster size distinguishable for small angles, see Fig. 1.15a [30].



(a) The cluster length distribution of a pixel detector with $25\ \mu\text{m}$ pixels placed in a beam at an angle of 2.5° .



(b) The stability of the measured angle due to cluster length, vs. sensor bias voltage. The true angle placement is labelled in grey.



(c) A cartoon showing how particles passing through a sensor at a shallow angle results in a long cluster.

Figure 1.15: An illustration of the concept of using extended barrel modules at high η and small angles to the incident particles. Particles exhibit a distinct cluster length for a given incident angle [30].

Quantity	RD53 specification	Current Pixel readout chip
Hit rate	3 GHz cm ⁻²	400 MHz cm ⁻²
Trigger rate	1 MHz	200 kHz
Trigger latency	12.8 μ s	6.4 μ s
Pixel size	50 $\times$ 50 μ m	50 $\times$ 250 μ m
Chip size	2 $\times$ 2 cm ²	2 $\times$ 2 cm ²
Current consumption	< 8 μ A/pixel	20 μ A/pixel
Radiation tolerance	0.5-1 Grad	300 Mrad
Minimum stable threshold	600 e	1500 e

Table 1.1: A subset of the specifications for the RD53 demonstrator readout chip, which will comply with the requirements for HL-LHC. The RD53 specification is compared with the most recent Pixel detector readout chip specifications, in the innermost layer [31].

Inclined layout

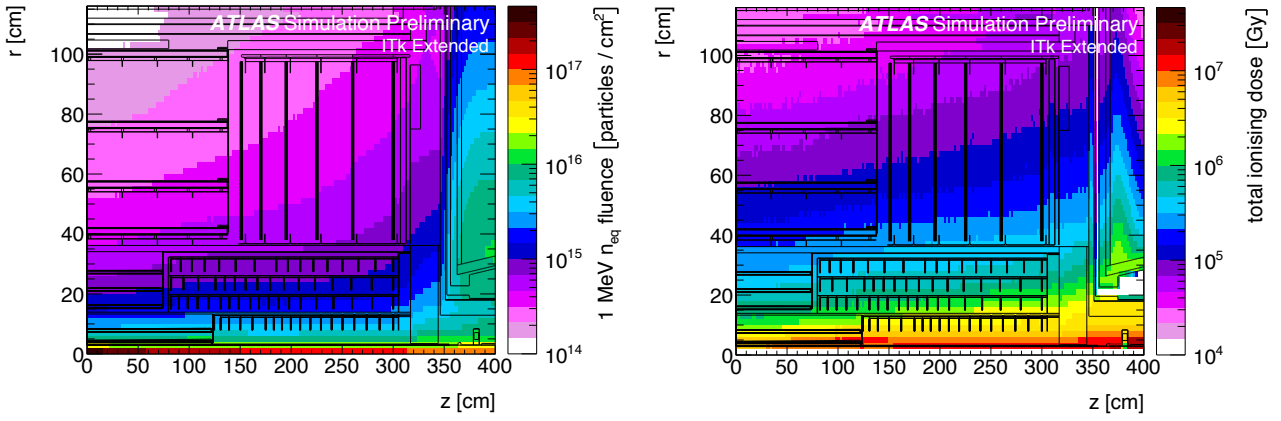
The inclined layout extends the Pixel barrel layers out to $\eta = 1$ and a series of small modules placed at various angles and separations in $1 \leq \eta \leq 4$, see Fig. 1.14a. A single particle passing through the tilted sensors will hit several sensors in a single layer whilst crossing less material than the extended layout and has more hits in the forward region. Passing through less material reduces multiple scattering which improves vertex reconstruction.

The inclined layer design moves away from more traditional mechanical support structures, which is more challenging for production than the extended layout. The final layout decision is expected to be announced in the Pixel ITk Technical Design Report in late 2017.

1.4.2 ITk Pixel modules

Along with optimising the ITk layout, the Pixel detector modules need to be upgraded to cope with the increased particle fluence, radiation damage, and data transmission requirements, which surpass the conditions in any previous run in the ATLAS detector. As with the current Pixel detector, Pixel modules in ITk will most likely be hybrid detectors consisting of a pixelated silicon sensor and a mixed-signal, front-end readout chip. The sensors themselves will experience a high fluence relative to previous Runs, for example the current innermost layer of the Pixel detector will experience a design lifetime fluence of 5×10^{15} n_{eq}cm⁻² [32] whereas the innermost layer of the ITk Pixel detector is expected to experience a fluence of $\sim 2\text{-}3 \times 10^{16}$ n_{eq}cm⁻², see Fig. 1.16a. The detector will be replaced halfway through the HL-LHC duration, so the actual sensors will only experience half the total fluence quoted here. A high neutron equivalent particle fluence will change the sensor's electrical properties over its lifetime which will affect its efficiency as a detector. The precise mechanism with which radiation damage effects silicon sensors, and strategies for coping with that damage, is detailed in Chapter 3. The effects of radiation damage must also be included in the Athena framework's sensor simulation, this is detailed in Chapter 3, Section 3.6.

The front-end readout chip that makes up the other half of a Pixel module must also be modified



(a) The simulated 1 MeV neutron equivalent fluence per square cm distribution in the ITk extended layout. The fluence map is particularly useful in planning for the radiation tolerance of silicon sensors over the detector lifetime.

(b) The Total Ionising Dose (TID) in the ITk extended layout. The TID map is particularly useful in planning for the radiation tolerance of read-out electronics over the detector lifetime. TID is typically expressed in Rad, $1 \text{ Gy} = 100 \text{ Rad}$.

Figure 1.16: Fluence and TID distributions in the ITk extended layout for an integrated luminosity of 3000 fb^{-1} of 14 TeV proton-proton collisions, which is the projected luminosity over 10 years for HL-LHC [33].

for ITk. RD53, an inter-experiment collaboration between the ATLAS and CMS experiments, has been formed with the task of designing tools and strategies for front-end readout chips to be used with hybrid pixel detectors in HL-LHC conditions [31]. The collaboration will produce a demonstrator chip, RD53A, which will include IP blocks already tested in silicon. A summary of some of the requirements put forward by the collaboration for the ITk readout chip is shown in table 1.1.

A chip, FE65-p2, was produced to test features and design strategies for the demonstrator. FE65-p2 features $50 \times 50 \mu\text{m}^2$ pixels, which is necessary to reduce the per-pixel hit rate in the HL-LHC environment. A smaller pixel size also has the advantage of an increased resolution in z , which is important for vertex separation in high pile-up environments. Reducing the pixel size by a factor of 5, when compared to the smallest pixel size in the current Pixel detector, is made possible by the use of 65 nm technology which provides the necessary logic density.

FE65-p2 was also the first instance of a readout chip employing analog logic blocks surrounded by a sea of synthesised digital logic. Previous front-end readout chips tiled blocks of analog and digital logic over the entire chip in a step-and-repeat procedure, which is a more labour intensive design process. From Fig. 1.16b, the ITk Pixel front-end chip is expected to receive an ionising dose of 1 GRad over its lifetime compared to the 250 MRad for the innermost layer of the current Pixel detector [32]. However, since the detector will be replaced halfway through HL-LHC, the actual radiation tolerance needed is only half of the TID: 500 MRad. To test the radiation-hardness of a variety of front-end designs, FE65-p2 was equipped with columns of different analogue front-end variations and then irradiated in a proton beam to see how each variation's performance would be affected over time [34]. These tests resulted in an analogue front-end design that will be used in the RD53 demonstrator chip, along with two others, which is expected to be produced in late 2017.

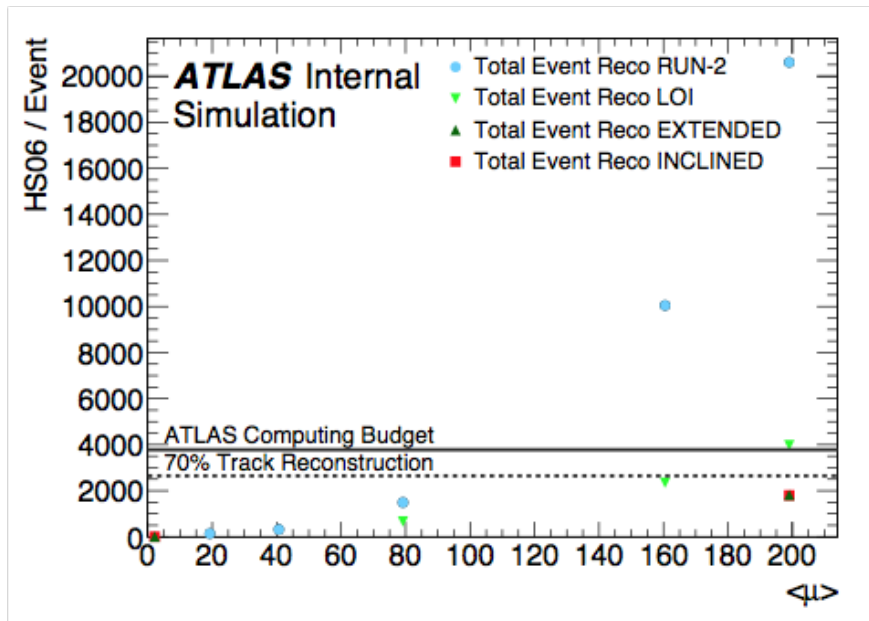


Figure 1.17: A benchmark for the computation time per Event vs average pile-up for various detector layouts. The plot shows that for the layout used in Run 2 the time taken to process each event far exceeds the computing budget. Also, due to the increasing number of seed candidates at high pile-up, the time taken to process an event grows non-linearly with μ . Plot courtesy of Heather Gray.

1.4.3 High pile-up in track reconstruction

Reconstruction is a fundamental part of data analysis as if it is done poorly it can introduce a bias into collected data or ignore interesting information by failing to reconstruct it. As such, significant computational resources are dedicated to reconstruction. The first step of track reconstruction, which builds seeds from the silicon detectors, is computationally the most expensive and grows non-linearly with pile-up, see Fig. 1.17. The Kalman filter algorithm is one of the most used techniques for reconstruction. As the number of interactions per second, the pile-up (μ), increases, the number of seed-candidates the Kalman filter needs to evaluate increases non-linearly. Various approaches to circumvent this problem are currently being explored. These include looking at new algorithms and techniques to perform reconstruction from replacing specific steps in the reconstruction process to overhauling the entire reconstruction process altogether. Another approach is to keep the same basic steps in place but to move the computations to different architectures. For example implementing a Kalman filter on an inherently parallel architecture such as FPGA's [35], Intel's Xeon-Phi architectures and GPU's [36]. Future technology, that may become commodity computing in a decade's time, is also being explored. In Chapter 2, for example, a neuromorphic chip, designed to run neuromorphic networks, has been investigated for its suitability to implement a common tracking algorithm, the Kalman filter.

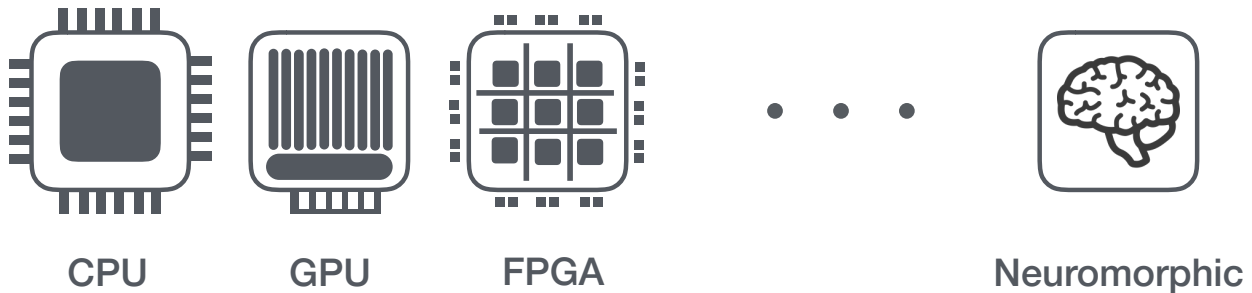
Chapter 2

Kalman filter in IBM's TrueNorth

With future colliders attaining unprecedented luminosities and data rates, new methods of computation and new architectures on which to perform them are being explored. With the advent of a post-Moore's law field of computation the range of technology and infrastructure available in the next decade is less easy to foresee and thus plan for. Future commodity computing architectures may take a form that the current HEP infrastructure must adapt to, so early explorations into the portability of common algorithms to novel architectures is necessary. An algorithm key to track reconstruction in multiple HEP experiments over the past 50 years is the Kalman filter, see its usage in ATLAS track reconstruction in chapter 1.3. As such the implementation of this algorithm on novel architectures is a promising step to understanding the usability of such a device and projecting the feasibility of its inclusion in future HEP computing.

In recent years, neuromorphic architectures have become more prominent in commercial companies such as IBM. In 2008 IBM entered a Defense Advanced Research Projects Agency (DARPA) project called SyNAPSE that involved the design of a scalable, low power neuromorphic architecture and in 2015 released the first version of this chip to select institutes for testing and development. Neuromorphic architectures are, quite literally, brain or neuron-inspired computers.

In this chapter the implementation of a Kalman filter in IBM's neuromorphic architecture, TrueNorth, will be explored and its performance used to evaluate its usage in HEP computing. A brief introduction to the concept of neuromorphic computing will be covered in section 2.1, IBM's TrueNorth chip will be introduced in section 2.2, and the implementation of a Kalman filter in section 2.3 with the results of that implementation discussed in section 2.4 .



2.1 Neuromorphic computing

Neuromorphic computing is computing based on our knowledge of how the brain processes and distributes information. The brain processes data in small elements called neurons. Complex processes in the brain arise from the amalgamation of these simple units. Efforts in brain inspired computing broadly fall into two categories [37]:

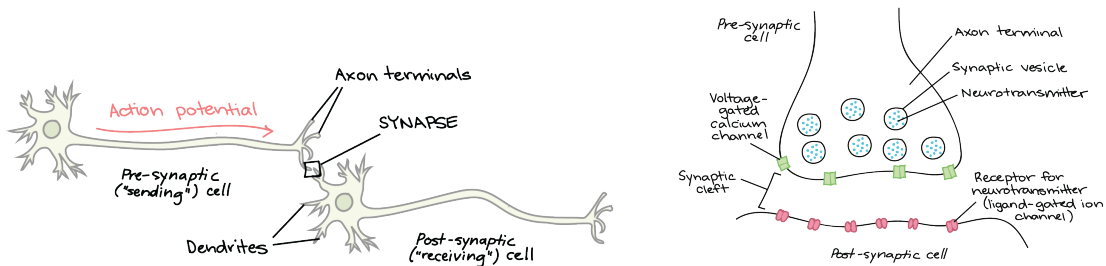
1. For the understanding of biological information processing through synthesis. This is a top-down approach to understanding the intricacies of the brain. International collaborations such as the Human Brain Project (HBP) have been formed with the explicit purpose of top-down modelling.
2. To produce future computing architectures based on biological information processing.

The latter approach seeks to take what mechanisms are understood about the brain and apply them to conventional computing methods. Despite our knowledge of how the brain operates, on the whole, still being largely incomplete, there are some key attributes that make neuromorphic computing desirable.

Brains are incredibly efficient in terms of energy consumed vs. number of operations performed when compared with a standard CPU, both in terms of computations per second and because of the information density of those computations [38]. The brain volume has an optimal wire-to-processor ratio [39]. It is fault tolerant - we know that large parts of the brain are regenerated everyday with no loss in performance [40] and that the brain exhibits learning. The latter design approach seeks to pick and choose the attributes of brain computation that cause these desirable features and use them to build a computer with deterministic output. To understand the fundamentals of neuromorphic computing then, is to understand some basic principles about the fundamental building block in neuroscience: the neuron.

2.1.1 Biological neurons

The neuron is a cell that can be deconstructed into the following substructures. Dendrites are projections from the body of the neuron that act like wires, transmitting electrochemical charge-carriers towards the soma, the cell body. The soma is also connected to an axon, a long chain of specialised cells that terminate in a series of protrusions known as the axon terminal, see Fig. 2.1a. Neurons form long chains in the body that communicate both electrically, with



(a) A very simplified cartoon of the biological neuron and its connectivity.

(b) A simplified cartoon of the synapse and post-synaptic cell.

Figure 2.1: Simplified cartoons of the neuron and synapse [41].

direct conducting channels between the cells, and chemically via synapses: chemical bridges between the axon terminal, in the pre-synaptic neuron, and the dendrites, in the post-synaptic neuron, see Fig. 2.1b. The consequence of synaptic transmission is a change in the properties

of the cell membrane of the post-synaptic neuron. The transmission can either increase or decrease the potential of the exterior of the cell membrane with respect to the interior of the cell.

Since a single neuron is likely connected to multiple axons, its membrane potential is determined by the integration of all of these inhibitory and excitatory post-synaptic potential changes. The membrane potential has an associated time constant which describes the rate at which charge leaks away till the membrane reaches some resting potential. When the membrane potential

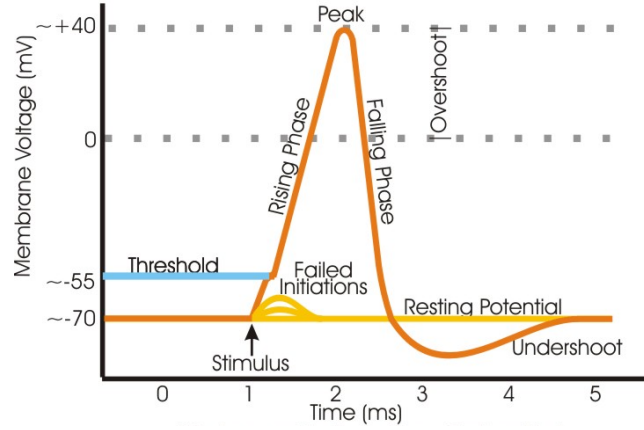


Figure 2.2: A schematic of the action potential, or spike, emitted by the soma.

reaches a specific threshold in the neuron, an avalanche of ion channels will open creating what is known as an action potential, where the neuron potential rises rapidly, falls off with a characteristic time, and undershoots as is seen in Fig. 2.2. This produces a current pulse, also called a spike, which is propagated along the axon to the synapses.

The axon is an insulated conductor, whose diameter can also vary and affect the transmission of electrical impulses produced by the soma. A popular model for the relationship between

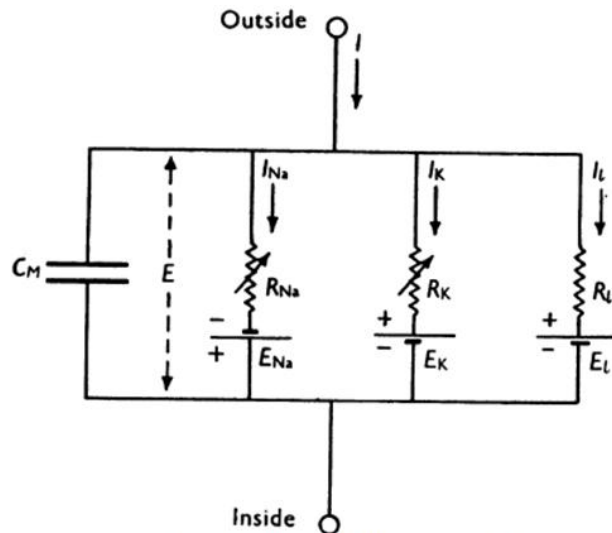


Figure 1: Circuit Model of an Axon [1]

Figure 2.3: The Hodgkins-Huxley model of the current passing across a semi-permeable cell membrane by three ion channels.

the membrane potential and the post-synaptic current is the Hodgkins-Huxley model [42] that considers each of the three dominant ion channels at the synapses as parallel RC circuits where the semi-permeable cell membrane is represented in each case with a resistor-capacitor pair

with a probabilistic model assigned to each, see Fig. 2.3. This results in a series of differential equations with 20 tuneable parameters.

Simplified versions of the Hodgkins-Huxley model exist, for example the leaky-integrate-and-fire model, which replaces the separate treatment of the ion channels by a single RC circuit with no probabilistic switching.

2.1.2 Neuromorphic networks and artificial neural networks

Neuromorphic architectures take key characteristics of the spiking biological-neuron model and create an electrical simulation of that model. Neuromorphic devices have a vast morphology but they share the following attributes:

- **Spikes to encode data** Unlike conventional computing architectures that encode instructions and data in binary words, neuromorphic devices encode data in temporally separated spikes.
- **Distributed, simple, parallel processing** All neuromorphic architectures share the concept of a neuron performing computations on inputs. The computations it performs have little variety and so complexity arises from the connectivity between the neurons and the number of tuneable parameters each neuron has.
- **Thresholding a weighted sum** Neurons calculate a weighted sum of their inputs, store that value as a membrane (or neuron) potential, and check it against a set threshold. Some models include a temporal element where the membrane potential can leak away at a given rate.
- **Spiking when over threshold** If a neuron potential is over threshold, the neuron emits a spike of its own and the neuron potential is reset. In some of the more biologically plausible architectures the reset includes a refractory period during which the neuron cannot fire again.

These devices differ from a traditional von Neumann architecture in that they have no complex instruction set, but only a model file to update the neuron parameters and connectivity of the neurons. Instead of a single cluster of device memory and a CPU, each neuron has its own memory and IO blocks such that each neuron can operate independently for truly parallel computation. In many ways neuromorphic architectures more closely resemble Artificial Neural Networks (ANN) but there are some important differences. The concept of thresholding in a neuromorphic neuron could be analogous to the activation function in an ANN neuron, aside from the concept of the neuron potential retaining values for multiple timesteps. Neuromorphic neurons that seek to closely resemble biological neurons usually contain some analogue circuitry to better model the shaping of the post-synaptic spikes as they enter the membrane which can lead to more complex behaviour than is typically exhibited by ANN neurons. Neuromorphic neurons may also be programmed to exhibit time-dependent plasticity over short or long time periods, in which neurons fire more or less frequently depending on how often spikes are sent to them. This concept has been shown to translate well to ANN vision tasks [43].

2.2 IBM's TrueNorth

The architecture considered in this study is the first version of IBM Research's fully digital neuromorphic chip, TrueNorth. Whilst many other hardware realisations of neuromorphic architectures focus on simulating biological neuron behaviour, TrueNorth's focus is on low-power, event-driven computation. The chip was developed in response to a DARPA project called SyNAPSE [44]. SyNAPSE challenged participants to meet a set of criteria including spike rate encoding, hardwired crossbars with programmable parameters, operating speed > 10 Hz, and that any hardware produced would, at scale, consume $< 10^{-12}$ J per synaptic operation. With the focus on scalability, low power, and processing elements operating with a slow clock, IBM designed TrueNorth.

In 2015 IBM Research released TrueNorth NS1e evaluation boards, see Fig. 2.4, to a select

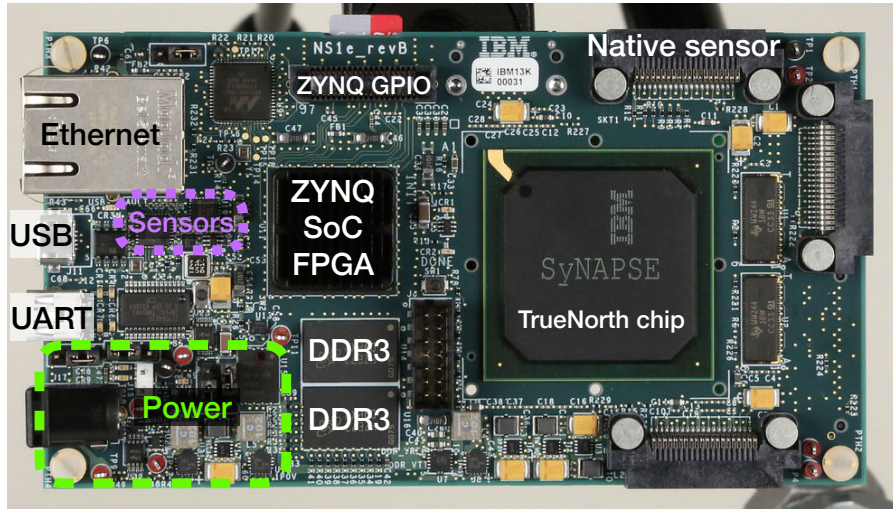


Figure 2.4: The NS1e TrueNorth evaluation board.

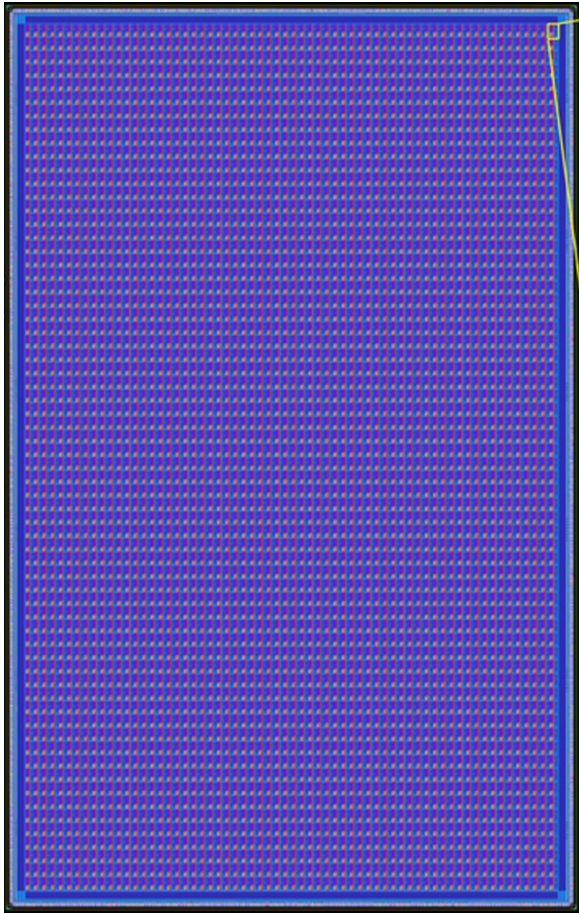
number of institutes in the USA, including Lawrence Berkeley National Laboratory, to develop algorithms, perform alpha testing on the software released, and to use the supplied TrueNorth testboards. The NS1e boards contain a single TrueNorth chip, a ZYNQ System-on-Chip ARM core and FPGA, which runs linux and interfaces directly with TrueNorth, and optional native sensor IO to send spikes directly into the chip. The software released at the time consisted of a Hardware Description Language (HDL) called the Corelet Programming Environment (CPE), a TrueNorth Neuro-Synaptic Chip Simulator (NSCS), and an extension to the popular Caffe¹ deep learning framework called Tea. Tea supported the constraints of the TrueNorth architecture and allowed users to train feed-forward neural networks and utilise other standard layer designs in Caffe's library. Early in 2016, support for Tea was discontinued and IBM released a framework known as the Energy-Efficient Deep Neuromorphic network (Eedn) [45]. Eedn shifted TrueNorth's focus from feed-forward and deep neural nets to convolutional and all recent publications affiliated with the project have focused on vision tasks.

The rest of this chapter will detail the implementation of a simple tracking algorithm in TrueNorth and evaluate its suitability and ease-of-use in future collider experiments.

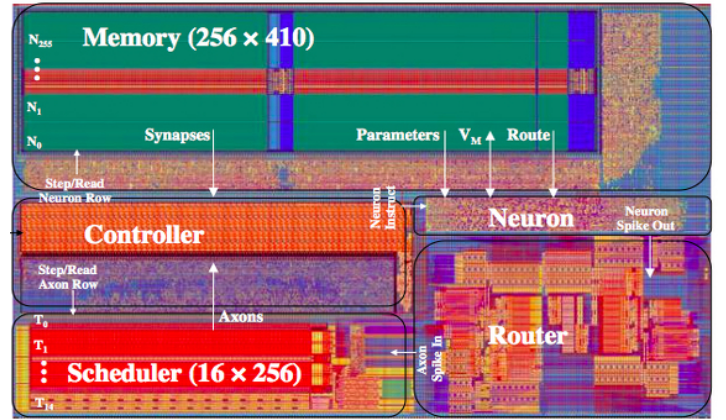
¹Available at <http://caffe.berkeleyvision.org/>.

2.2.1 Top level design: topology, timing, and packets

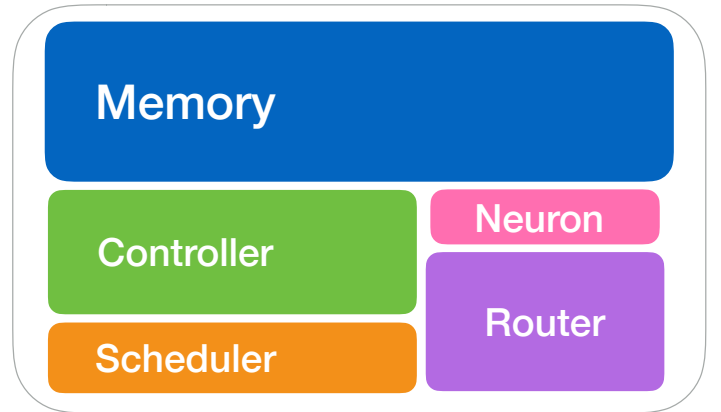
The TrueNorth architecture is comprised of parallel processing units called cores, of which there are 4,096 on a single chip. TrueNorth's cores are tiled in a 64×64 grid, as seen in Fig. 2.5a, where each core can be connected to any other core on the chip by a two-dimensional mesh network. Packets enter the chip via one of the IO ports and hop from one core to another until they reach the correct one. Every core contains a router which communicates data packets between cores and delivers them to the core's scheduler when they reach the correct destination.



(a) The TrueNorth chip top-level layout. Each highlighted rectangle on the chip is one of 4,096 cores.



(b) An annotated layout of the TrueNorth core.



(c) A schematic of the top-level blocks in a TrueNorth core, where the size of each block is proportional to the area taken up in the real layout.

Figure 2.5: IBM's TrueNorth chip layout and core detail.

TrueNorth cores contain a router, scheduler, token controller, a core memory block, and the computational element: a neuron block, as seen in Fig. 2.5b and 2.5c. The need for both a scheduler and token controller is due to TrueNorth's mixed Synchronous-Asynchronous design. Instead of a fast global clock net, which consumes idle power, communication between cores on the chip happens asynchronously with a four-phase handshake protocol. Packets passed from the router to the scheduler are stored in the scheduler's 16×256 SRAM. The dimensions of the SRAM correspond to the 256 inputs to the programmable core² and the 4 bits of delay that can be programmed into a packet and thus inputs to the router are stored in the corresponding

²Programming the core is covered in Section 2.2.2

cell of the SRAM. The scheduler, too, is event-driven and queues packets as they arrive from the router asynchronously.

The actual computations in the core happen in the 256 neurons in the neuron block and are synchronous. The clock supplied to each neuron in the neuron block is generated by the token controller and so is local to each core. The clock pulses for performing computation are only generated if the input to the neuron has a packet waiting and if the connection to that input is active, thus although the logic in a neuron is synchronous its usage is event driven.

Once the token controller has established that a neuron will be active it sends the exact number of clock pulses needed to perform the computation, it requests the neuron model parameters stored in the core memory block and combines them with the input packets stored in the scheduler and sends both of these to the neuron, it takes any outputs produced by the neuron block at the end of the computation and sends them to the router and updates the neuron parameters in the core memory block.

Although each core performs a combination of synchronous and asynchronous tasks, these are all initiated by the arrival of a signal edge in the token controller called a tick. The sending of data packets from a source core to its destination, along with any processing done by the core upon the packet's arrival, must be completed before the arrival of the next tick. As such it is not called a clock in the traditional sense but rather a global synchronization point. If communication or computation is not completed within one tick a number of errors will be flagged by the chip. These will be addressed in Section 2.4.4.

2.2.2 The programmable core: neurons, axons, and the crossbar

The elements of TrueNorth discussed up to this point are necessary to understand the chip's unique constraints, but are also fundamental to the chip's architecture and cannot be changed by the user. In this section the programmable elements of the TrueNorth core will be discussed.

Alluding to the biological counterparts they imitate, the inputs, connections within the core, and computational elements are known as axons, synapses, and neurons, as shown in Fig. 2.1a. A single neuron can have multiple axons connected to it, as shown in Fig. 2.6a and likewise a single axon can be connected to multiple neurons as in Fig. 2.6b. Combining these two concepts results in a crossbar: a connectivity map of the axons and neurons where connections are denoted by the presence of a binary synapse, as shown in Fig. 2.6c.

Axons, aside from being inputs to the core, have two additional properties. They can buffer an incoming packet for up to 15 ticks before it is sent to the crossbar, this is the 16×256 SRAM described in section 2.2.1, and they can be assigned a label, $G \in \{0, 1, 2, 3\}$. Both of these values can be set by the user. The axon label is stored, along with synaptic connectivity and neuron parameters, in the core memory. The axon label is used by the neuron to indicate how packets arriving from an axon with that label should be handled, thus a single neuron can handle packets up to four ways.

Once a neuron receives and processes an incoming packet from an axon, it can emit one of its own. A neuron's output can be fed back to a new axon in either the same core or another core but may only be connected to a single axon. Splitting the output of a neuron over multiple axons will be covered in section 2.2.5. The user sets both the crossbar, by indicating whether a synapse is present or not, and neuron output connectivity.

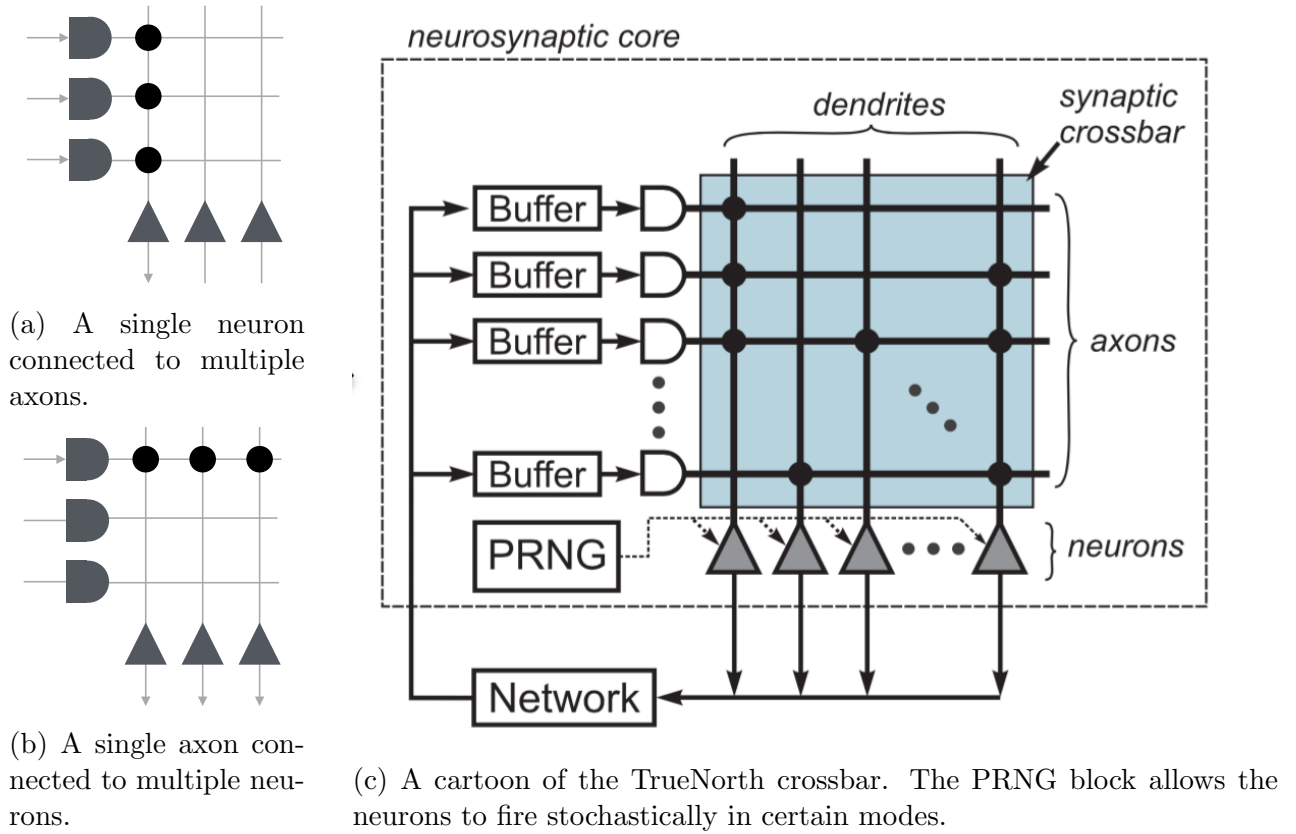
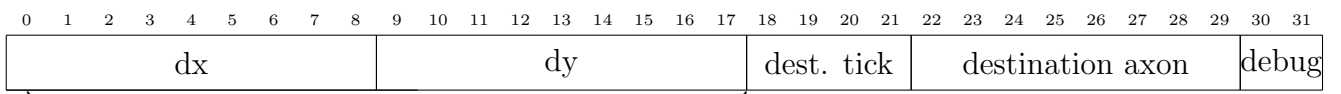


Figure 2.6: TrueNorth elements and the crossbar. Axons, inputs, are represented as semicircles. Neurons, which perform computations and output the result, are represented as triangles. Synapses, which indicate if a neuron and axon are connected, are shown as black circles.

2.2.3 Spikes and the neuron update equation

TrueNorth communicates and processes data packets called spikes. A spike is a 32 bit packet containing routing information to indicate which core it belongs in, a destination tick which indicates when it should be injected into the crossbar, a destination axon in the crossbar, and some debugging information, see Fig. 2.7. Once a spike is injected into the network it first



Removed by routing network when handed to scheduler

Figure 2.7: Composition of a 32 bit TrueNorth spike, as processed by the routers in each core. When the spike reaches its intended core the first 18 bits are stripped and only the tick, axon, and debug bits remain.

moves in the dx direction till it reaches the router in its destination column. At this point the first 9 bits are stripped away and the spike continues moving in the dy direction of the core network till it reaches the router in its destination core, where the second 9 bits are stripped away. The remaining 14 bit packet is sent into the scheduler to be processed by the core, as described in the previous section.

Once a spike packet has been routed through the crossbar to the correct neuron, it is weighted according to the label of the axon it came from, since there are 4 possible axon labels a single neuron can store up to 4 weights. The weighted input is added to the neuron potential, V_j ,

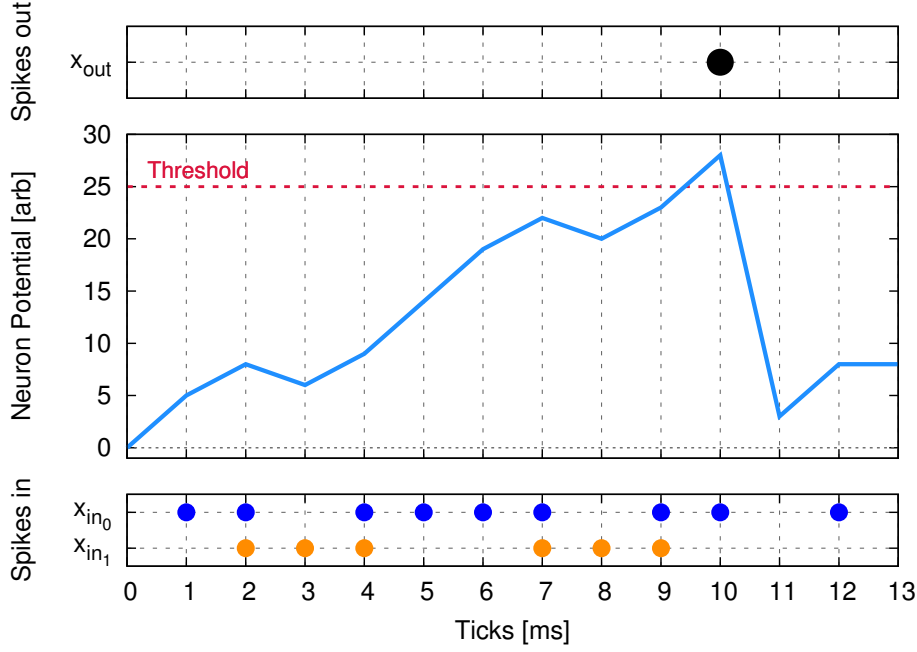


Figure 2.8: The evolution of a neuron potential over multiple ticks. The neuron has two inputs which are weighted differently: one increases the value of the neuron potential and the other reduces it. After ten ticks $V_j > \alpha_j$ and the neuron emits a spike. V_j is then reset by $V_j - \alpha_j$.

where $j \in [1, 256]$ indexes neurons on the core. Once all inputs to the neuron have been processed in this way V_j is changed by a fixed bias, or leak, and then compared to a threshold, α_j .

If $V_j \geq \alpha_j$ the neuron emits a spike, this is more commonly referred to as the neuron firing. Once the neuron has fired V_j will be changed according to a programmed reset scheme. The neuron potential is also compared to a negative threshold, however if below this threshold it will not spike but will also be reset. The process described here is illustrated in Fig. 2.8. A cartoon of the neuron circuit is shown in Fig. 2.9. The sequence of operations above constitute the neuron update, which is described more completely by the equation 2.1. The parameters in this equation are described in table 2.1. Both this update equation and table are a subset of the complete neuron update, which has additional parameters to control leak

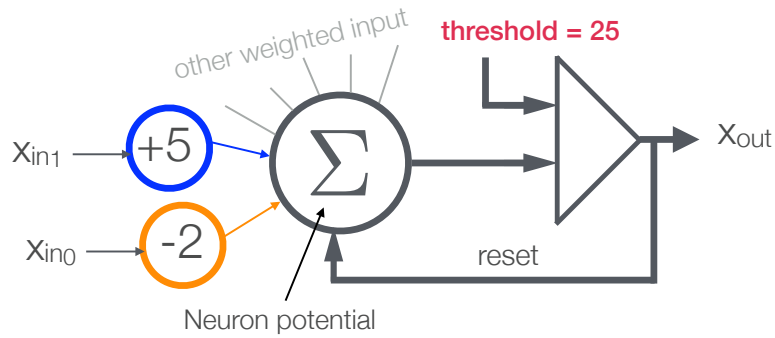


Figure 2.9: A cartoon of the neuron circuit. Input spikes are weighted according to the label assigned to the axon they entered the crossbar on and summed. This value is then checked against a fixed threshold and if greater than or equal to threshold emits a spike and is reset according to some scheme.

and stochasticity that are not included in the Kalman filter, this is described in appendix A.1.

The neuron potential update at each timestep:

$$V_j(t) = V_j(t-1) + \sum_{i=1}^{256} A_i(t) \times w_{i,j} \times s_j^{G_i} \quad (2.1a)$$

The potential, V_j is capped at a ceiling and floor defined as:

$$\begin{aligned} \text{ceiling} &= \begin{cases} \alpha_j + M_j, & \text{if } \gamma_j = 2 \\ 393216, & \text{otherwise} \end{cases} \\ \text{floor} &= \begin{cases} -\beta_j - M_j, & \text{if } \gamma_j = 2 \\ -393216, & \text{otherwise} \end{cases} \end{aligned} \quad (2.1b)$$

The neuron will cross the positive threshold and fire if:

$$V_j > (\alpha_j + \eta_j) \quad (2.1c)$$

The neuron will cross the negative threshold if:

$$V_j(t) < -\beta_j \quad (2.1d)$$

If V_j crosses the positive threshold it will be reset according to:

$$V_j(t) = \begin{cases} \text{Vrst}_j, & \text{if } \gamma_j = 0 \\ V_j(t) - \alpha_j - \eta_j, & \text{if } \gamma_j = 1 \\ V_j(t), & \text{if } \gamma_j = 2 \end{cases} \quad (2.1e)$$

If V_j crosses the negative threshold it will be reset according to:

$$V_j(t) = \begin{cases} -\text{Vrst}_j, & \text{if } \gamma_j = 0 \\ V_j(t) + \beta_j + \eta_j, & \text{if } \gamma_j = 1 \\ V_j(t), & \text{if } \gamma_j = 2 \end{cases} \quad (2.1f)$$

scope	symbol	range	description
Basic neuron update	i, j	$[1, 256]$	Axon and neuron indices, respectively
	$V_j(t)$	$[-393216, 393216]$	Neuron potential at tick t
	$A_i(t)$	$[0, 1]$	Presence of spike in axon i at tick t
	$w_{i,j}$	$[0, 1]$	Presence of synapse connecting axon i and neuron j
	G_i	$[1, 4]$	Axon label
	$s_j^{G_i}$	$[0, 2^8 - 1]$	The weight assigned to spikes incident on neuron j from axons with label G_i
Threshold and reset	α_j	$[0, 2^{18} - 1]$	Positive threshold. If V_j exceeds this the neuron will fire and be reset.
	β_j	$[0, 2^{18} - 1]$	Negative threshold. If V_j drops below this the neuron will not fire but will be reset.
	M_j	$[0, 2^{18} - 1]$	Acts as a mask for the reset stochasticity and also extends the ceiling for the special reset case where $\gamma_j = 2$
	γ_j	$[0, 2]$	Membrane potential reset mode
	η_j	$[0, 2^{18} - 1]$	Masked stochastic threshold.
	V_{rst_j}	$[0, 2^{15} - 1]$	Reset voltage in special reset mode

Table 2.1: A partial list of the changeable neuron parameters in TrueNorth. The complete list is available in table A.1.

The entirety of the spike packet is information about when and where the packet should be injected into the network, it contains no additional information and so is essentially a digital 1. Digital spiking is a key difference between TrueNorth and other neuromorphic or ANN chips and affects the way data is encoded and processed. For example non-spiking architectures usually encode data in binary and can represent non-integer values using floating or fixed point representations. The neuron has no native way of handling binary encoding so unary is used. In unary, the ordering of ones and zeros in a data word does not change the meaning of the word. For example, in binary a word with 3 bits can encode 2^3 different values whereas in unary only 3 different values can be encoded. To illustrate this consider a word with two 0's and one 1.

In binary:

$$100 \neq 010 \neq 001$$

However, in unary:

$$100 = 010 = 001$$

Encoding in unary can be done by defining a data word in terms of a fixed number of ticks

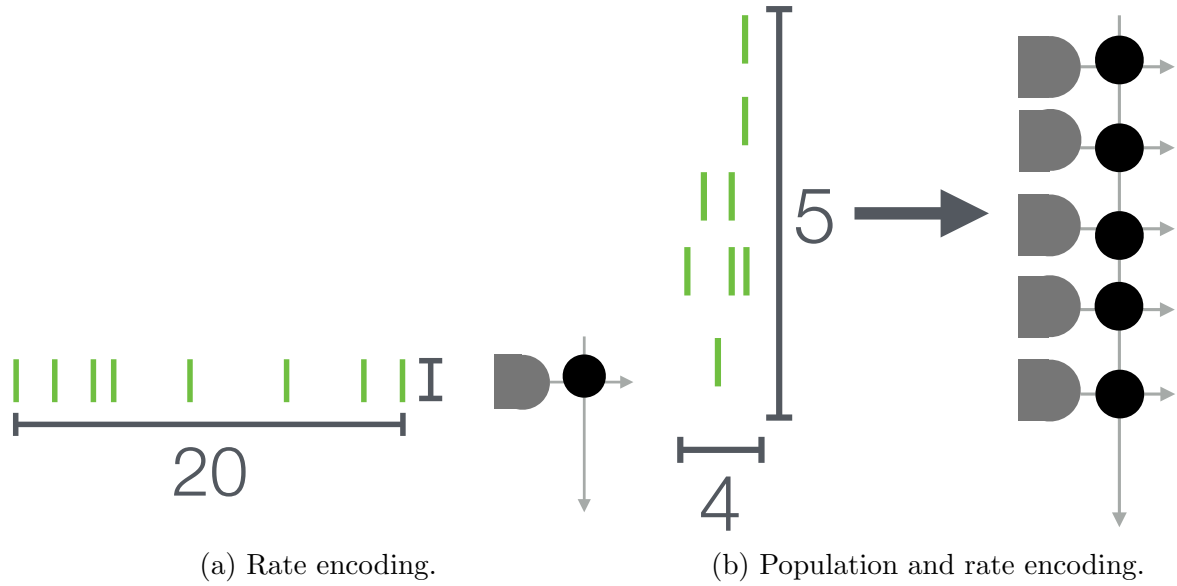


Figure 2.10: Rate and population encoding with axons. The same value (0.4) is encoding in a 20-spike word using 8 spikes. In rate-only encoding this takes 20 ticks and 1 axon to complete. In rate-and-population encoding this takes 4 ticks and 5 axons.

(the above example would take 3 ticks to encode) which is known as rate encoding or by a combination of rate and population encoding. In this context population encoding describes splitting up a word over multiple axons, as is shown in Fig. 2.10b. Splitting up a word over multiple axons allows it to be processed in less ticks but takes up more of the crossbar. This becomes an important consideration when many inputs are needed to the same crossbar and hence there is limited space to speed up a computation. This topic will be explored further in section 2.4.

2.2.4 Arithmetic in TrueNorth

With an understanding of the basic functionality of neurons and connectivity, a TrueNorth user can begin to program more complex operations. This section will outline how to implement basic arithmetic concepts in TrueNorth.

Adding two spike trains together simply involves connecting them to the same neuron on the crossbar, as in Fig. 2.11a. This is because weighted addition is intrinsic to the neuron functionality. Likewise, subtracting spike trains uses the same concept but assigns a different axon type to change the sign of the weight, as is illustrated in Fig. 2.11b. Multiplying incoming spikes by a weight and dividing that weight by comparing it to a threshold are also operations intrinsic to the neuron.

To multiply a spike train by an integer, the neuron weight alone can be applied. For example, if a spike train was connected to a neuron with $s_j^{G_i} = 2$ and $\alpha_j = 1$ then the output of that neuron would spike twice the rate of the input train, so if there were 20 spikes in an input word of there would be 40 spikes in the outgoing word. Setting $s_j^{G_i} > \alpha_j$ multiplies the incoming spike rate, and setting $s_j^{G_i} < \alpha_j$, divides the incoming spike rate. To multiply an incoming spike train by a numerical weight, the weight must be represented as a rational number of the form:

$$\text{weight} = \frac{s_j^{G_i}}{\alpha_j} \quad (2.3)$$

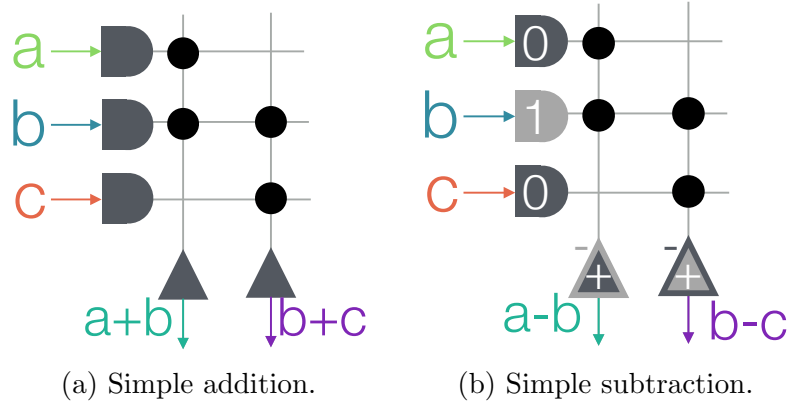


Figure 2.11: Crossbars showing basic signed addition of multiple spike trains in a core. In the case of subtraction two axon labels are used so the neuron knows which inputs to add and which to subtract. Note that each neuron can interpret these labels differently.

The neuron weight register is only 8-bits, so the precision of weights that can be represented is limited. The range of weights can be extended by duplicating a spike train over multiple axons and adding those inputs together, as is shown in Fig. 2.12a.

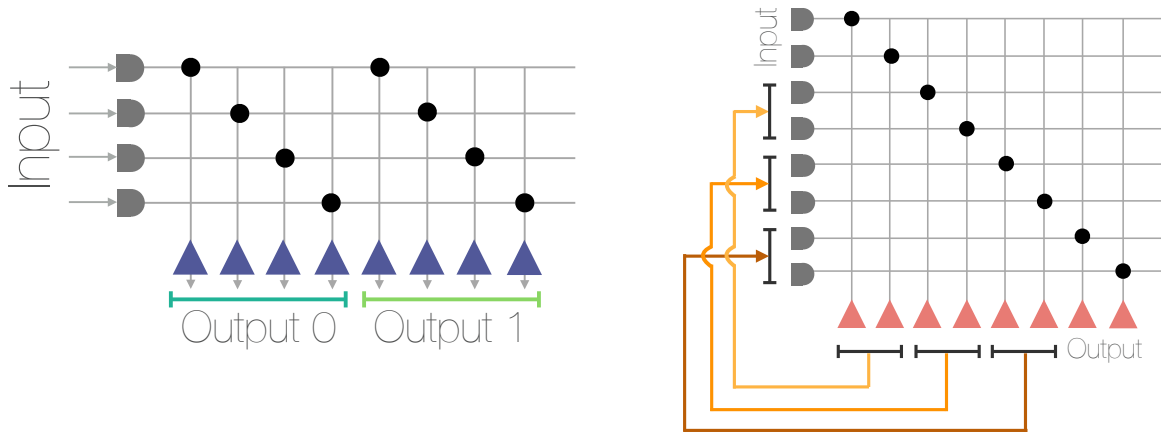
Increasing the range of the threshold, or denominator of the rational weight representation requires splitting it into two factors: threshold = factor a $\times$ factor b, and using two neurons. The first neuron has whatever neuron weights are needed assigned and $\alpha_j = \text{factor a}$. The first neuron's output is fed into a second neuron with $s_j^{G_i} = 1$ and $\alpha_j = \text{factor b}$, as shown in Fig. 2.12b. Increasing the range of the threshold, however, is rarely needed since it is already an 18 bit register. A combination of these two methods can be used for high precision weights.



(a) Neuron weight range can be extended by duplicating inputs over multiple axons, where the sum of all weights associated with the axons is equal to the desired weight, and summing the result.

(b) Neuron threshold range can be extended by splitting the threshold up into factors where $\alpha_0 \times \alpha_1$ are equal to the desired threshold, and passing the output from the first neuron to the second neuron.

Figure 2.12: Extending the range of values the neuron weight and threshold is able to represent is possible to increase representation precision. In the case of extending the neuron weight range, at least two axon types are needed. The first (in red in Fig. (a)) has the max value of 255, the second (in grey) will have the value (weight % 255).



(a) A simple 1:2 splitter corelet for a 4-pin Connector. As with the delay corelet, the crossbar is a simple diagonal and adding additional splits would just require duplicating this pattern along additional neurons.

(b) A simple delay corelet for a 2-pin Connector. With each recurrent connection the input can be delayed up to 15 ticks, so the min. and max. programmable delay for this corelet is 4 ticks and 46 ticks, respectively.

Figure 2.13: A couple of simple corelet concepts.

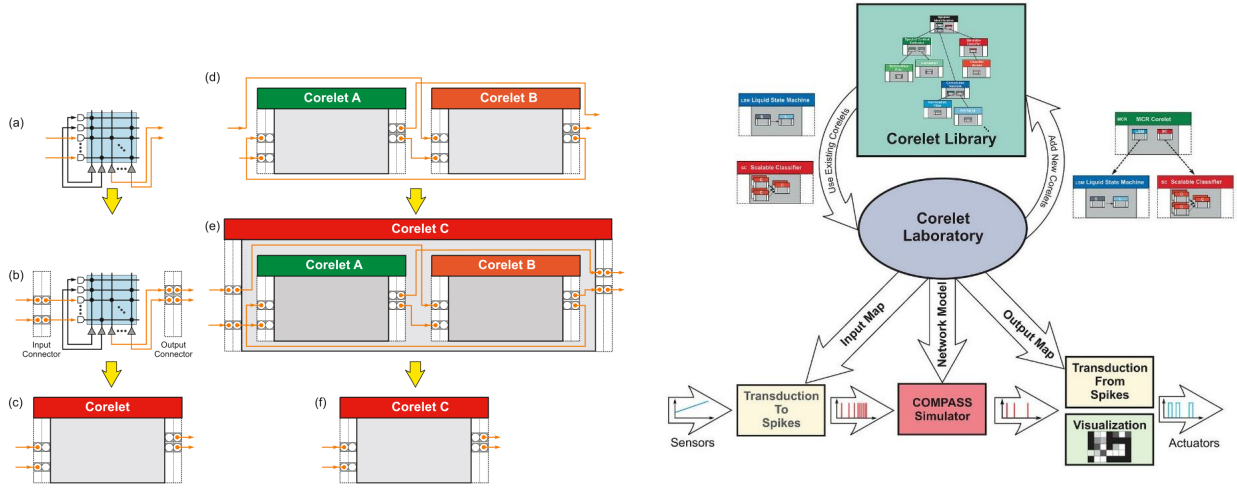
2.2.5 Key corelet concepts

With many ways to organise cores and parameters in TrueNorth, IBM has defined a workflow for programming and deploying models. The connectivity, axon and neuron parameters, and crossbar in TrueNorth are programmed using a Hardware Description Language (HDL) called the Corelet Programming Environment (CPE), which is built and written using Matlab. The CPE defines several objects including the neuron, core, and a top level container called a corelet. Corelet objects can contain crossbars for multiple cores and define multiple neuron types. They can also be built from other corelets, so a single top-level corelet can be defined from multiple smaller ones, see Fig. 2.14. A couple of popular corelets in the corelet library, supplied with CPE, include a splitter, that takes a corelet with an output Connector of size n and produces x copies, also of size n , as shown by the crossbar in Fig. 2.13a. Another popular corelet is the delay corelet, which simply delays some input for t ticks, see Fig. 2.13b. These corelets have in common that they compensate for limitations in the TrueNorth architecture: the delay corelet compensates for the maximum delay per input being only 15 ticks, and the splitter corelet allows for neuron outputs to be sent to multiple destinations, something also not supported by the native architecture.

One of the complications of programming a corelet is that the inputs and outputs to the object cannot simply be assigned to individual pins directly but must be interpreted through a class called the Connector. A Connector must be defined to have a finite size. This becomes a problem when only a few of the pins in the Connector are desired in some cases but all are needed in others, there is no option in the CPE that allows pins to remain unused or unconnected for a specific instance.

For example, consider the case where a corelet's output Connector was size four. Later in the design process it was decided that the output of that corelet should be used in two places in the design, but only two of the pins in the Connector were needed in the second instance. Because only a partial number of pins in the Connector are needed the user would either have

³Outputting a different sized Connector than was input is not supported by the corelet library version of the splitter, so a custom one had to be written since the need for this functionality arose often.



(a) Creating a top-level design in CPE: (a) describe core and neuron parameters, (b) assign Connectors to axons and neurons, (c) synthesise as corelet, (d,e,f) connect to other corelets in top-level corelet.

(b) The minimal workflow in creating, deploying, and testing a new corelet in TrueNorth. Aside from creating a corelet and combining it with others from the corelet library, stimulus must also be transduced to spikes.

Figure 2.14: Corelet creation and deployment workflows.

to decide to reserve two extra axons in the recipient corelet and connect the size four Connector, effectively wasting two axons, or would have to write a custom splitter class to split a Connector of size four into two Connectors of size two³. Naturally this second solution involves using an additional corelet, and hence core, and will add latency to the design - all of which could easily be averted if the user could connect pins directly, rather than having to go through Connectors. At the time of writing it is not possible to force multiple corelets, which take up less than one core, to be deployed on a single core. In other words there is no way to optimise core usage by placing multiple corelets on a single core. This is a feature that IBM have spoken of including in future updates to CPE. Thus, at the time of writing, the number of cores used in a design is greater than or equal to the number of corelets in the design.

Once a corelet has been described it is synthesised into a model file that can be run by the TrueNorth NSCS or loaded onto the chip. When the model is deployed it is not possible to change any of the parameters or connectivity - so no online-learning is possible with this architecture. The significance of this will become apparent in section 2.3.2.

2.3 Implementation and test setup

In Chapter 1.3, the Kalman filter was introduced and its use in track reconstruction in the ATLAS experiment was described. In this section the implementation of a subset of the Kalman filter equations in the TrueNorth architecture will be detailed.

2.3.1 The Kalman filter and the steady-state Kalman filter

The Kalman filter [46] is a recursive set of equations that produces an estimator of the state of a discrete-time controlled process in a way that minimises the mean squared error of the estimator with respect to the true state. The Kalman filter operates on a stream of noisy

measurements and essentially ‘filters’ out the noise to recover the underlying state. The state, $x \in \mathbb{R}$, is defined according to the linear stochastic difference equation:

$$x_t = \mathbf{F}x_{t-1} + \mathbf{D}u_{t-1} + w_{t-1} \quad (2.4)$$

Where $\mathbf{F}$ is the state transition matrix which relates the state at the previous timestep, $t - 1$, to the state at the current timestep t , u is the control input, $\mathbf{D}$ relates the control input to the state, and w is a white, wide-sense stationary, normally-distributed stochastic noise called the process noise. The state is not measured directly but rather through a device that has some additional properties, thus measurements, y , of the state are given by:

$$y_t = \mathbf{H}x_{t-1} + v_{t-1} \quad (2.5)$$

Where $\mathbf{H}$ is the measurement matrix that relates the measurement to the state and v is also a white, wide-sense stationary, normally-distributed stochastic noise assumed to be independent of w , called the measurement noise. The covariance matrices of w and v are defined to be Q and R , respectively. Unlike similar state estimators such as the Weiner algorithm, the Kalman filter is light-weight to implement as it recursively incorporates a weighted history of the entire state at each time step, instead of requiring that all states in the history are retained and operated on together. Because the Kalman filter incorporates all past estimates of the state, and due to the statistical properties of the estimators, the Kalman filter estimators are more precise than predictions made using a single measurement.

The Kalman filter has its origins in an iterative linear least-squares estimator of the form:

$$S^2 = \sum_{i=1}^T \frac{(y_i - \mathbf{H}\hat{x}_i)^2}{\sigma_i^2} \quad (2.6)$$

Where y , x , and $\mathbf{H}$ are defined as before, S is the mean square error of the estimator and is distributed normally, $\sigma^2 = v$, and is the variance of the measurement, or noise on the measurement. This equation can be extended to n dimensions by writing it in matrix form as:

$$S^2 = (\vec{y} - \mathbf{H}\hat{x})^\top \mathbf{R}^{-1}(\vec{y} - \mathbf{H}\hat{x}) \quad (2.7)$$

where y, x are now assumed to be $n \times 1$ vectors and $\mathbf{R}$ is an $n \times n$ matrix of the covariance of the measurement noise. To solve for the estimator, $\hat{x}$, S^2 is minimised with respect to x :

$$\begin{aligned} \frac{dS^2}{dx} &= -2\mathbf{H}^\top \mathbf{R}^{-1}(y - \mathbf{H}\hat{x}) = 0 \\ \therefore \hat{x} &= (\mathbf{H}^\top \mathbf{R}^{-1} \mathbf{H})^{-1} \mathbf{H}^\top \mathbf{R}^{-1} y \end{aligned} \quad (2.8)$$

Which can be simplified further by defining:

$$\mathbf{P} = \text{covar}[x] = (\mathbf{H}^\top \mathbf{R}^{-1} \mathbf{H})^{-1} \quad (2.9)$$

and therefore equation 2.8 becomes:

$$\hat{x} = \mathbf{P} \mathbf{H}^\top \mathbf{R}^{-1} y \quad (2.10)$$

This formulation assumes the entirety of measurements have been made *a priori* and, depending on how many timesteps are used, can involve a large matrix inversion. To break the equation into smaller steps, and allow it to be evaluated iteratively, it can be rewritten as:

$$\hat{x}_t = \hat{x}_{t-1} + \mathbf{P}_t \mathbf{H}_t^\top \mathbf{R}_t^{-1} (y_t - \mathbf{H}_t \hat{x}_{t-1}) \quad (2.11)$$

with the state covariance given by:

$$\mathbf{P}_t = (\mathbf{P}_{t-1}^{-1} + \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{H}_t)^{-1} \quad (2.12)$$

The part of the matrix multiplication in equation 2.11 which involves $\mathbf{P}$ is still computationally intensive at each iteration and can be reduced using the following identity:

$$\mathbf{K}_t = (\mathbf{P}_{t-1}^{-1} + \mathbf{H}_t^\top \mathbf{R}_t^{-1} \mathbf{H}_t)^{-1} \mathbf{H}_t^\top \mathbf{R}_t^{-1} = \mathbf{P}_{t-1} \mathbf{H}_t^\top (\mathbf{R}_t + \mathbf{H}_t \mathbf{P}_{t-1} \mathbf{H}_t^\top)^{-1} \quad (2.13)$$

where K is known as the Kalman gain matrix which minimises the *a posteriori* error covariance, $\mathbf{P}_t$ [47]. The Kalman gain is then incorporated into equation 2.11 and 2.12 so that they becomes:

$$\begin{aligned} \hat{x}_t &= \hat{x}_{t-1} + \mathbf{K}_t (y_t - \mathbf{H}_t \hat{x}_{t-1}) \\ \mathbf{P}_t &= (\mathbb{I} - \mathbf{K}_t \mathbf{H}_t) \mathbf{P}_{t-1} \end{aligned} \quad (2.14)$$

Another way to think of this is that as the measurement noise covariance, $\mathbf{R}$, in the Kalman gain expression, tends to zero the measurement y is trusted more whilst the predicted state $\mathbf{H}\hat{x}$ is trusted less and vice versa. The equations in 2.14 constitute the discrete Kalman filter measurement update equations and combine a new measurement with an *a priori* state estimate to obtain an improved *a posteriori* measurement. Not included in these equations is the fundamental *prediction* step which is given by the linear stochastic difference equation 2.4 at the beginning of this section and the covariance prediction equation [48]:

$$\mathbf{P}_t^- = \mathbf{F} \mathbf{P}_{t-1} \mathbf{F}^\top + \mathbf{Q} \quad (2.15)$$

To distinguish between the update equations that use the state and covariance estimate from the previous timestep (equations 2.15 and 2.4), and the measurement update equations derived in this section (equations 2.13 and 2.14), the time update states are denoted with a ‘-’, as seen in equation 2.15. Thus the full discrete Kalman filter update equation is summarised below:

$$\hat{x}_t^- = \mathbf{F} \hat{x}_{t-1} + \mathbf{D} u_{t-1} + w_{t-1} \quad (2.16a)$$

$$\mathbf{P}_t^- = \mathbf{F} \mathbf{P}_{t-1} \mathbf{F}^\top + \mathbf{Q} \quad (2.16b)$$

$$\mathbf{K}_t = \mathbf{P}_t^- \mathbf{H}_t^\top (\mathbf{R}_t + \mathbf{H}_t \mathbf{P}_t^- \mathbf{H}_t^\top)^{-1} \quad (2.16c)$$

$$\hat{x}_t = \hat{x}_t^- + \mathbf{K}_t (y_t - \mathbf{H}_t \hat{x}_t^-) \quad (2.16d)$$

$$\mathbf{P}_t = (\mathbb{I} - \mathbf{K}_t \mathbf{H}_t) \mathbf{P}_t^- \quad (2.16e)$$

A point of interest for this set of equations is that since the covariance of all state estimator and measurement is wide-sense stationary, and white with zero mean, over time they will average out to be zero. This will quickly cause the estimation error covariance, $\mathbf{P}$, and the Kalman gain, $\mathbf{K}$, to stabilise and remain constant, hence both of these values can be pre-computed or trained depending in the application.

A subset of the Kalman filter estimator is the steady-state Kalman filter which assumes $\mathbf{P}$ and $\mathbf{K}$ to be constant. If we additionally simplify the system to omit any control input (so no $\mathbf{D}u$ in equation 2.16a), and set the measurement matrix to the identity (so $\mathbf{H} = \mathbb{I}$) the Kalman filter equations reduce to:

$$\hat{x}_t^- = \mathbf{F} \hat{x}_{t-1} + w_{t-1} \quad (2.17a)$$

$$\hat{x}_t = \hat{x}_t^- + \mathbf{K}_t (y_t - \hat{x}_{t-1}) \quad (2.17b)$$

where the state estimate covariance, $\mathbf{P}$, and Kalman gain, $\mathbf{K}$, are not shown as they are constants in time. This equation further simplifies to:

$$\begin{aligned} \hat{x}_t &= (\mathbf{F} - \mathbf{K}) \hat{x}_{t-1} + \mathbf{K} \vec{y}_t \\ &= \mathbf{A} \hat{x}_{t-1} + \mathbf{B} \vec{y}_t \end{aligned} \quad (2.18)$$

In this work, $\mathbf{A}$ is defined as the matrix multiplying the recurrent state estimate and $\mathbf{B}$ is the matrix multiplying the observations/measurements of the system. In this case, $\mathbf{B}$ is equal to the Kalman gain, $\mathbf{K}$. Since steady-state dynamics are assumed, both the $\mathbf{A}$ and $\mathbf{B}$ matrices can be pre-calculated, or trained.

Attempts were made to implement a full Kalman filter in the TrueNorth architecture in a couple of ways. There are many existing methods to implement a Kalman filter in an artificial neural network (ANN). These perform the covariance matrix and Kalman gain matrix updates using recurrency and/or backpropagation depending on the type of Kalman filter used [49]. This method could not be implemented in TrueNorth as no neuron parameters or connectivity can be changed during runtime and additionally any training of an ANN-style network in TrueNorth is extensive. For such a method to be implemented the chip would have to be stopped and started continuously to update the parameters, resynthesise the model file, and load it onto the chip.

Another method considered was that of the Neural Engineering Framework [50] which takes linear control systems and implements them in populations of neurons. In this definition, a neuron population refers to many neurons performing a very similar operation with slight variations. Unfortunately part of the formulation required that a non-linear leak be implemented in the neurons and that the post-synaptic current have a finite width, thus altering the value arriving at the neuron: essentially, this method does not work for digital spikes and so had to be abandoned.

After exploring tracking algorithm implementations in existing neuromorphic architectures we came to the conclusion that TrueNorth's neuron model is too simple to emulate their approaches. Instead, using the inherent ability of a neuron to do linear algebra, the Kalman filter was formulated using explicit matrix multiplication and addition. The following sections will describe this implementation.

2.3.2 Implementation and design

A steady-state Kalman filter was implemented in TrueNorth using linear algebra and was broke down into two processes: matrix element multiplication and addition which are easily implemented as was seen in section 2.2.4. For example, in a 2-dimensional state estimate produced

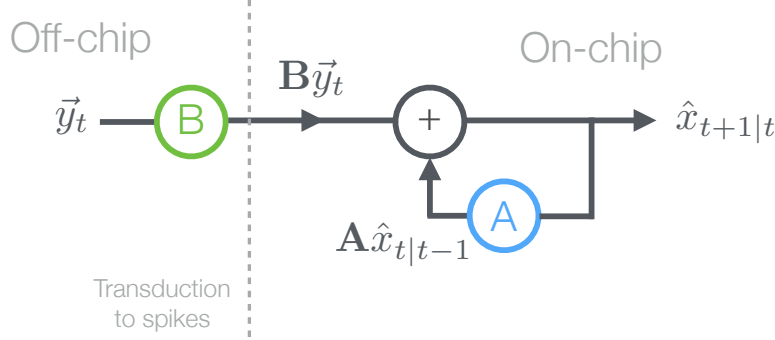
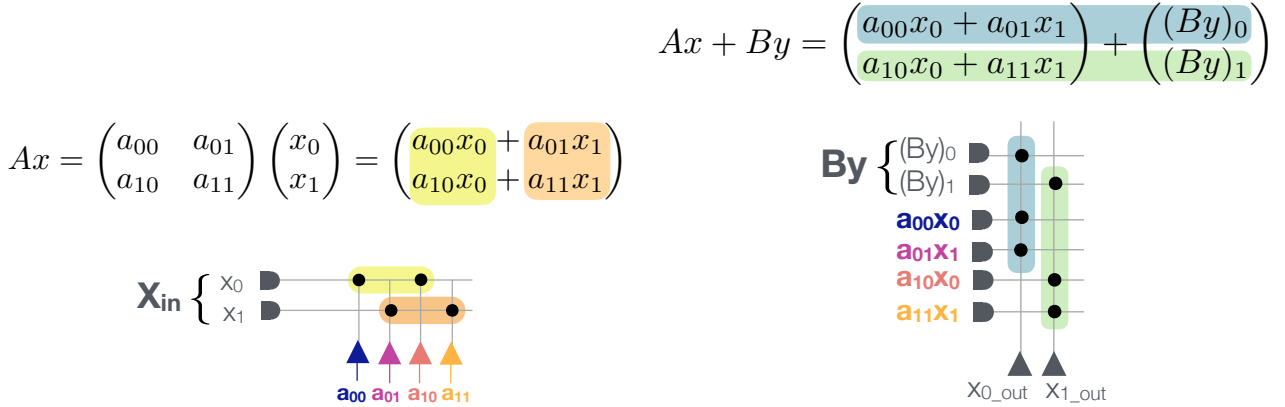


Figure 2.15: A circuit diagram for the steady-state Kalman filter implementation on TrueNorth, from equation 2.18.

by a Kalman filter, equation 2.18 would involve multiplying a (2×1) vector by a (2×2) matrix and adding it to the noisy measurement. These steps are broken down visually into crossbars as shown in Fig. 2.16.

The system measurements, the input to the Kalman filter, are multiplied by matrix $\mathbf{B}$, however this step is not performed on the chip, as is shown in Fig. 2.15. This is because the measurements have to be pre-scaled to lie in the range $[-1, 1]$, off-chip, before they can be transduced to spikes, so an additional multiplication is simple to perform. It is possible to port this operation onto the chip but it will introduce both additional latency and take up additional cores to perform. It is easy to see how this formulation could be extended for higher-dimensional states.



(a) Each matrix element is represented by a different neuron. This can be thought of as the columns of the resultant matrix corresponding to rows in the crossbar, as highlighted in the image.

(b) In the case of addition, values that should be added together are sent to the same neuron. Here the rows of the matrix correspond to columns in the crossbar.

Figure 2.16: The Kalman filter state estimate weighting and addition to the measurement step as shown in crossbars for a two-dimensional system. The outputs of the coloured neurons in Fig. (a) are sent to the corresponding axons in Fig. (b).

For example, the multiplication block in Fig. 2.16a would require n axons, and n^2 neurons and synapses, for an n -dimensional state. The addition block in Fig. 2.16b would require $n + n^2$ axons and synapses, and n neurons, for an n -dimensional state.

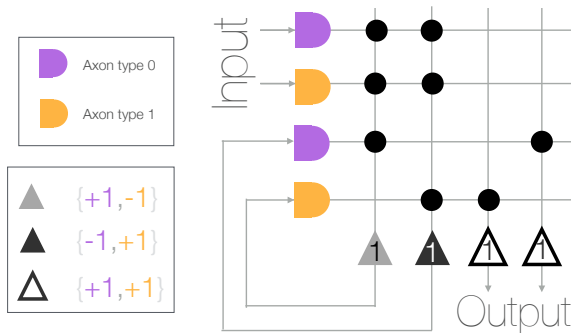
Positive and negative values

With the basic building blocks of the steady-state Kalman filter implementation formed, an additional consideration is how to deal with signed numbers. Whilst it may be possible to transform dynamical system measurements such that $y \in [0, 1]$, this does not guarantee that the corresponding $\mathbf{A}$ and $\mathbf{B}$ matrices will also be positive-valued, thus inputs are pre-scaled to the range $y \in [-1, 1]$. Standard binary words carry an extra bit for encoding the sign of the value and the logic that processes that word is designed to expect a sign-bit at a specific place in the packet. There is no such mechanism available to encode a sign bit in a spike train explicitly since spikes are integrated in time. One could envision an additional channel in the core reserved solely for a sign spike that would gate some special logic to treat the respective word as signed but the special logic quickly becomes complicated and was not pursued.

Instead, positive and negative values are handled in parallel and combined in the addition core, as in equation 2.19:

$$\begin{aligned} V_j^+(t) &= V_j^+(t-1) + \sum_{i=1}^{256} A_i(t) \times w_{i,j} \times \begin{cases} +1, & \text{if } G_i = 0 \\ -1, & \text{if } G_i = 1 \end{cases} \\ V_{k \neq j}^-(t) &= V_{k \neq j}^-(t-1) + \sum_{i=1}^{256} A_i(t) \times w_{i,k} \times \begin{cases} -1, & \text{if } G_i = 0 \\ +1, & \text{if } G_i = 1 \end{cases} \end{aligned} \quad (2.19)$$

These equations are very similar to the standard neuron update equation but here explicit neurons are assigned to encode the positive and negative values and at each tick this pair will have potentials with equal and opposite-sign; in other words the sum of the neuron potentials in the pair will always be zero. However if one of the neurons spikes, the pair becomes unbalanced, thus additional logic is used to reset the neuron that did not fire, this is illustrated by Fig. 2.17. This additional logic, for an n -dimensional system, will add $2n$ axons and synapses, and



$V_{\text{Neuron \#0}}$	$V_{\text{Neuron \#1}}$	$V_{\text{Neuron \#2}}$	$V_{\text{Neuron \#3}}$	Spikes in	Spikes out
$0 + 1 - 1 = 0$	$0 - 1 + 1 = 0$	0	0	1,1	0,0
$0 - 1 = -1$	$0 + 1 - 1 = 0$	0	0	0,1	0,0
$-1 + 1 + 1 - 1 = 0$	$0 - 1 = -1$	0	$0 + 1 - 1 = 0$	1,0	0,1
$0 + 1 - 1 = 0$	$-1 + 1 - 1 = -1$	$0 + 1 - 1 = 0$	0	1,0	1,0
0	$-1 + 1 = 0$	$0 + 1 - 1 = 0$	0	0,0	1,0
Total spikes				+3, -2	$2 - 1 = 1$

(a) An example of the paired signed neuron logic.

(b) A table showing the evolution of the paired neuron V_j for various spike patterns.

Figure 2.17: Signed addition reset logic, if one of the neurons in the pair fires it will add +1 to the neuron potential of its pair, thus ensuring that the sum of the neuron potentials in the pair remain 0.

n neurons to a signed addition crossbar. It should be noted that the signed addition crossbar neurons should have a negative threshold set to at least 2α for the addition to be performed correctly.

The final crossbar of the n -dimensional serial Kalman filter, with this signed addition concept included, uses $2n(n+3)$ axons, $2n(n+2)$ neurons and $4n(2n+1)$ synapses. The complete crossbar for the case $n=2$ is shown in Fig. 2.18.

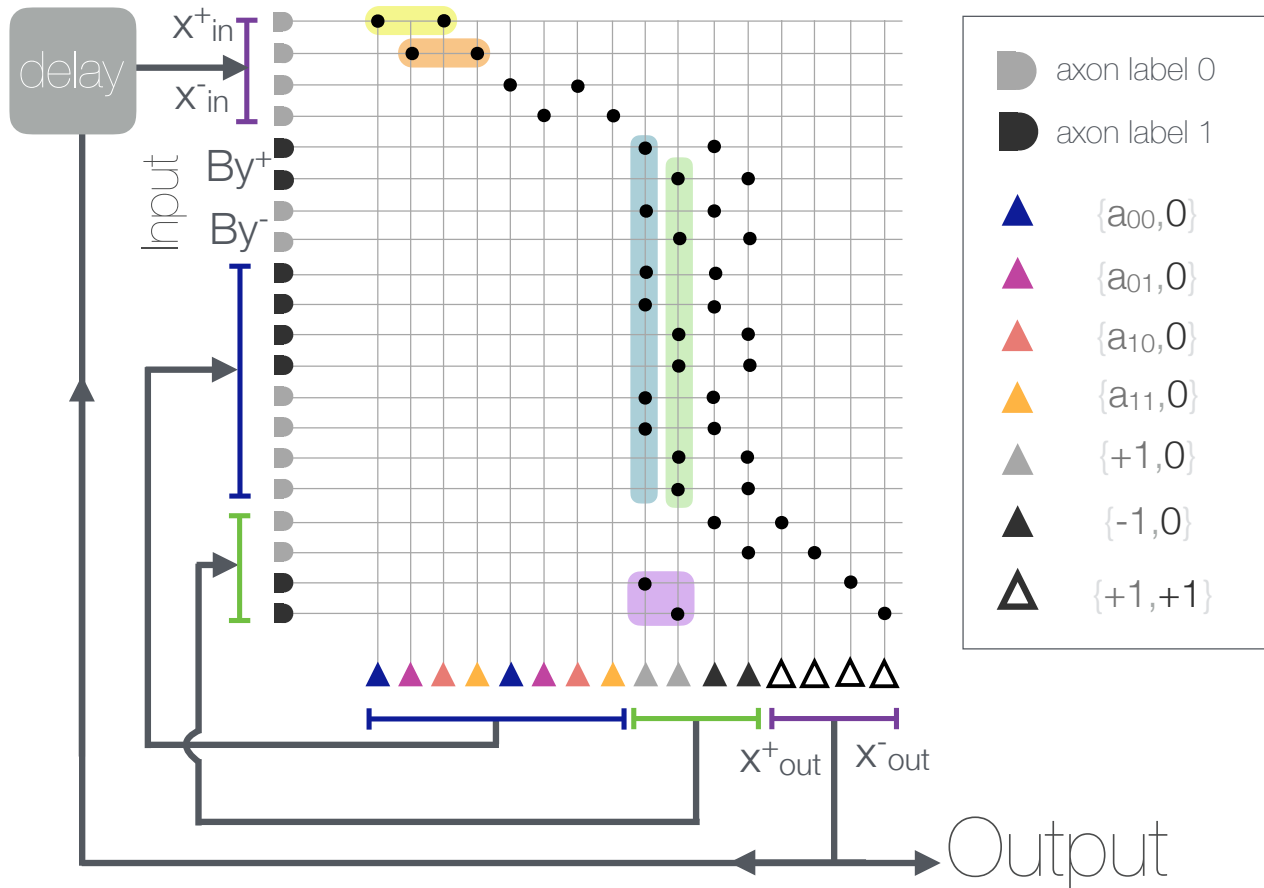


Figure 2.18: The complete 2-dimensional Kalman filter crossbar. Not shown explicitly is the delay corelet to correctly time in the state estimate with the measurements, as this depends on the size of the word used to encode the data. The split at the output is done using a splitter corelet, also not shown here. The highlighted portions show the multiplication, addition, and signed-reset logic for the positive channel described previously.

Serial and parallel implementations

Up till this point the crossbar logic described will only work with a purely ‘serial’ encoding, a rate encoding, as discussed in section 2.2.3, where a single axon-neuron pair will process a word and any variations in data are encoded in time-separated spikes. This section will explain the modifications necessary to the Kalman filter crossbars so that they can process data encoded across both time and multiple axons, this will be known as the parallel encoding.

First, the serial word must be parallelised by dividing it over a number of axons and neurons, hereafter called the blocksize, denoted by b . For example if a data word takes 32 ticks in the serial encoding, then it could be split into a parallel block of 8 axons and take 4 ticks to encode. A data word is thus defined as:

$$\text{data word size} = \text{window of time [ticks]} \times \text{blocksize [axons]} \quad (2.20)$$

The naive assumption, when formulating the parallel representation, would be to replace each synapse in the serial crossbar with a diagonal crossbar, and replicate each neuron and axon b times, see Fig. 2.19. To understand why this doesn’t work, first consider what is required of

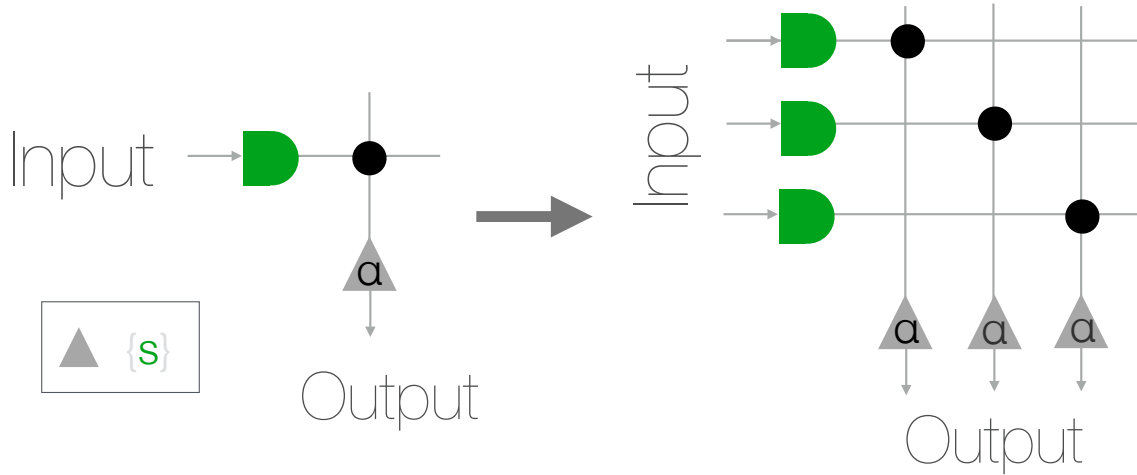


Figure 2.19: Naively, to parallelise a corelet simply replace each axon-neuron pair with b axons and neurons and a diagonal crossbar.

the parallel representation. For the equivalent stimulus, it must produce the same number of spikes for a given word. In the serial case this word is defined as taking a fixed number of ticks and the output is over 1 neuron. The equivalent parallel case must output the same number of spikes, when summed over the b neurons and the reduced number of ticks. Consider table 2.2 for a word of size 6 ticks, multiplying an incoming train by $\frac{s}{\alpha} = \frac{2}{3}$ and compare it to the output of the naive parallel solution in table 2.3 for $b = 3$ and 2 ticks. From the first word it is clear that the number of spikes emitted and, in fact, the total number of spikes is not the same. This is a problem for the Kalman formulation which is explicitly calculating the state estimate and for an accurate output relies on the multiplication and timing being accurate. The naive formulation does not work because not all neurons in the block receive the same input and so a different number of spikes are integrated over, which cannot be rectified without enforcing a binary-like encoding. Instead, consider a setup where every neuron receives the same input, i.e. with a full crossbar, as shown in Fig. 2.20. Although every neuron is receiving the same input, it is receiving b times too much. To deal with this, a linear ladder of thresholds is used. An analogous method is used in a linear ADC, where an input voltage is sampled over progressively increasing resistors. Using the crossbar and neuron parameters shown in Fig. 2.20 then, the

word	spikes in	spikes out	V_0	word	spikes in	spikes out	V_0
1	1	0	$0 + 2 = 2$	3	1	0	2
	1	1	$2 + 2 - 3 = 1$		0	0	2
	1	1	$1 + 2 - 3 = 0$		0	0	2
	1	0	$0 + 2 = 2$		1	1	1
	1	1	$2 + 2 - 3 = 1$		0	0	1
	1	1	$1 + 2 - 3 = 0$		0	0	1
2	1	0	2	4	0	0	1
	0	0	2		1	1	0
	1	1	1		0	0	0
	0	0	1		1	0	2
	1	1	0		0	0	2
	0	0	0		1	1	1
Total spikes in: 14, total spikes out: 9							

Table 2.2: The response of a single neuron to a serial, or rate-encoded, stream of 6-tick words. The neuron is set up to have $s = 2$, $\alpha = 3$, $\gamma = 1$, i.e. if the neuron fires V is reduced linearly by α . See table 2.1 for more details.

word	spikes in	spikes out	$V_{0,1,2}$
1	1, 1, 1	0, 0, 0	2, 2, 2
	1, 1, 1	1, 1, 1	1, 1, 1
2	1, 0, 1	1, 0, 1	0, 1, 0
	0, 1, 0	0, 1, 0	0, 0, 0
3	1, 0, 0	0, 0, 0	2, 0, 0
	1, 0, 0	1, 0, 0	1, 0, 0
4	0, 1, 0	0, 0, 0	1, 2, 0
	1, 0, 1	1, 0, 0	0, 2, 2
Total spikes in: 14, total spikes out: 8			

Table 2.3: The response of a block of three neurons to a parallel stream of 2-tick words. The neurons all have $s = 2$, $\alpha = 3$, $\gamma = 1$, i.e. if the neuron fires V is reduced linearly by α - the same as in table 2.2.

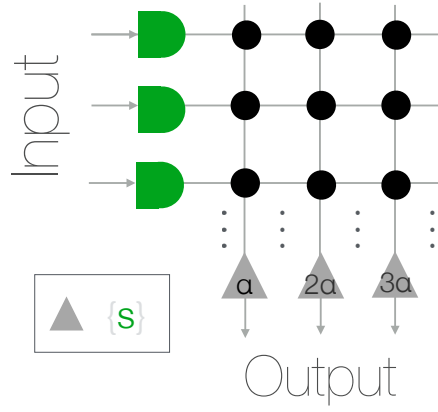


Figure 2.20: A full crossbar is used at the input axons so all neurons in the block receive the same input. The neurons are then setup to implement the same weights for the input axons but have a linearly increasing threshold from left to right. There is some additional logic, not shown in this diagram, to implement the correct reset across the neuron block.

stimulus applied in the serial and naive-parallel case produces an output as shown in table 2.4. Here, the total number of spikes emitted after each word is the same, so this is the axon/neuron setup that will be used in the parallel implementation.

An additional caveat is that for all the neurons in the block to sample the spikes correctly at each tick, they must all be reset by the same amount: by $(\alpha_0 \times \# \text{neurons fired in block})$, i.e. by α_0 times how many neurons fired in that timestep.

The first approach to the reset was to use neurons set up with $\gamma = 2$, which is the ‘no-self-reset’

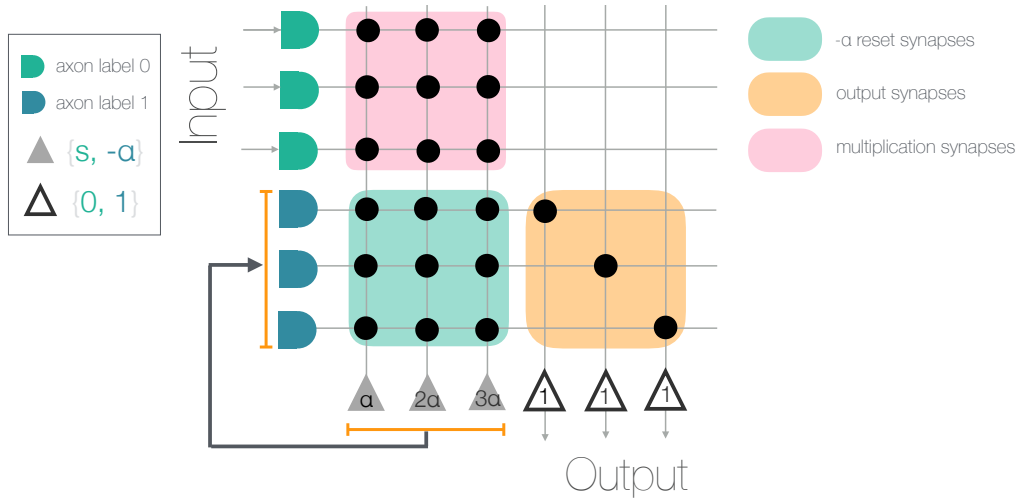


Figure 2.21: The parallel multiplication crossbar for $\gamma = 2$. This is no-self-reset-when-fire mode for the neurons, so if the neurons fire the spikes are both fed back to reset the neuron potential across the block and output from the core. This setup does not work as intended due to the cap on the neuron potential ceiling, see text for details.

mode for the neuron: when the neuron spikes its potential remains the same. The crossbar for this setup is shown in Fig. 2.21. The first part of the crossbar, spanning neurons and axons $\{1 - 3\}$, is the input block that sends all input spikes to all neurons in the block. The second full crossbar, spanning axons $\{4 - 6\}$ and neurons $\{1 - 3\}$, is the reset crossbar. Consider the case that the first neuron in the block fires, from Fig. 2.21 a spike will be queued in axon 4. In the next tick, the spike will be sent to neurons 1, 2, 3, 4. In neurons $\{1 - 3\}$ the spikes will be weighted by $-\alpha_0$, thus resetting all the neurons. The same principle holds if more neurons in

word	spikes in	spikes out	$V_{0,1,2}$
1	1, 1, 1	1, 1, 0	0, 0, 0
	1, 1, 1	1, 1, 0	0, 0, 0
2	1, 0, 1	1, 0, 0	1, 1, 1
	0, 1, 0	1, 0, 0	0, 0, 0
3	1, 0, 0	0, 0, 0	2, 2, 2
	1, 0, 0	1, 0, 0	1, 1, 1
4	0, 1, 0	1, 0, 0	0, 0, 0
	1, 0, 1	1, 0, 0	1, 1, 1
Total spikes in: 14, total spikes out: 9			

Table 2.4: The response of a block of three neurons to a parallel stream of 2-tick words. The neurons all have $s = 2$. The thresholds are: $\alpha_0 = 3$, $\alpha_1 = 6$, $\alpha_2 = 9$. The neurons are set up such that all are reset by the same amount: ($\alpha_0 \times \#$ neurons fired in block).

the block fire.

The reason this setup doesn't work is because of the ceiling applied to the potential in the case that $\gamma = 2$, from equation 2.1b:

$$V_j = \begin{cases} \alpha_j + M_j, & \text{if } \gamma_j = 2 \\ 393216, & \text{otherwise} \end{cases}$$

The parameter M_j can be set to be non-zero to increase the ceiling but doing so causes stochastic behaviour in the neuron. Capping V_j in this way causes a bias and does not produce the correct output, thus the reset portion of the crossbar and neuron parameters had to be redesigned.

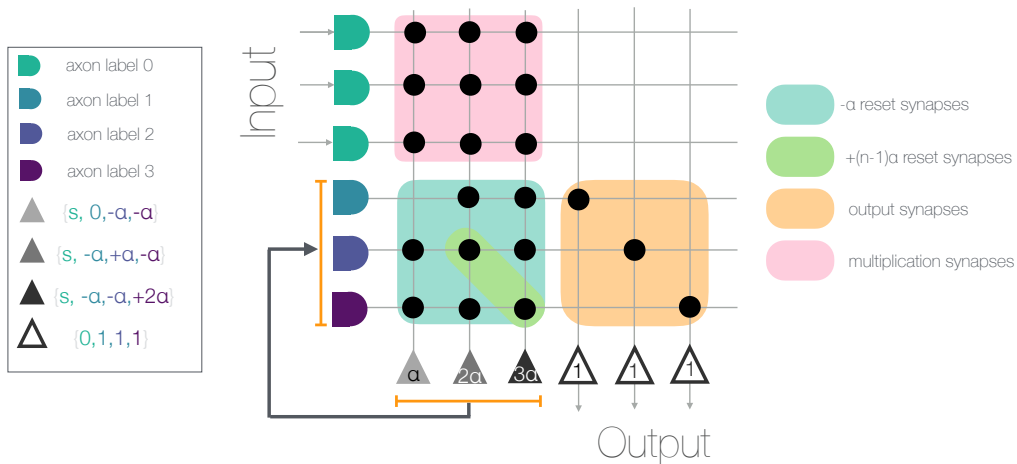


Figure 2.22: The parallel multiplication crossbar for $\gamma = 1$. In this mode neurons that fire reset themselves but are also reset by other neurons in the block firing. Unfortunately this particular crossbar setup only works for $b < 4$, see text for details.

If, instead of the neurons in the block having $\gamma = 2$, all have $\gamma = 1$ then the setup shown in Fig. 2.22 could be used. Although the crossbar looks the same as for the previous case, note

that two extra neuron settings and 2 extra axon labels used. In this setup if a neuron fires then its potential is reset linearly by the neuron itself. The spike emitted is then fed back into the crossbar to change the potential of the neurons that didn't fire. Note that all the block-neurons, except the first, also have a feedback loop that is connected to them which assigns a weight of $(j-1)\alpha_j$, this is to compensate for their potential already having been reduced by neurons to the left of them in the block.

However, although this crossbar produces the correct behaviour, it ceases to work for a blocksize > 4 as there are only 4 axon labels. If axons are desired to increase the range of the neuron weight this crossbar can only be used for blocksize < 3 . This constrain on the blocksize is compensated for by introducing an additional neuron-axon pair for each neuron in the block except the first, as seen in Fig. 2.23. This introduces additional complexity to the crossbar and almost doubles the number of neurons and axons but it does produce the correct behaviour.

One final problem is the limited size of the neuron weight register. The reset crossbar requires weights of up to $|(b-1)\alpha_0|$ to reset the last neuron on the block and the neuron weight register range is only $[-(2^8 - 1), 2^8 - 1]$, whereas the threshold range is $[-(2^{18} - 1), 2^{18} - 1]$. There are several ways to deal with this constraint. The first is making all $\alpha_j \leq 255$, but this lends little precision to the input weight applied at the top of the crossbar, recalling that a spike is weighted by $\frac{s_j}{\alpha_j}$ for each neuron.

Another, more practical solution, is to use the neuron weight extension method outlined in

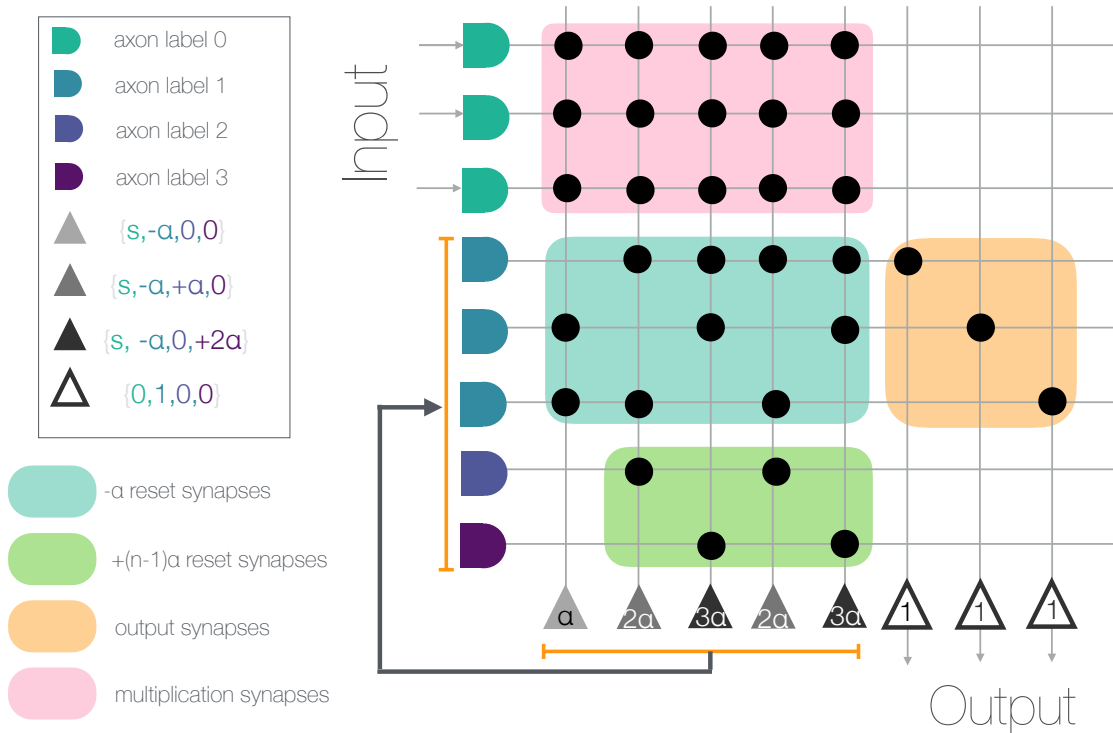


Figure 2.23: The parallel multiplication crossbar for $\gamma = 1$. In this mode, neurons that fire reset themselves but are also reset by other neurons in the block firing. This setup works for $b \geq 2$ and allows an extra axon label to extend the neuron weight precision. See text for details.

section 2.2.4. For example, if $\alpha_0 = 300$, then the axons feeding back spikes from neurons with $\alpha = 2 \times 300 = 600$ would be duplicated once: with the first axon having $s = 255$, and the second having $s = 45$, totaling $s = 300$, as desired. Then for the axons feeding back spikes from neurons with $\alpha = 3 \times 300 = 900$, the axons would correspond to weights $\{255, 255, 100\} = 600$ and so three axons would be needed, etc. The variable extending the range of thresholds that can be reset is p , i.e. if $p = 2$, there will be twice as many axons in the reset portion of the parallel multiplication crossbar, which extends the thresholds that can be used in the neurons

up to $2 \times 255 = 510$.

Once again, however, the limited number of axon labels become a problem. Consider that the weights at the input of the block likely need to have their range extended: thus at least 2 axon labels are needed to do this. An additional axon label is needed for the subtraction portion of the reset-crossbar, which leaves one remaining axon label for the addition portion of the reset crossbar whereas two are needed to extend the reset weight. The only way around this, if one still desires to use a neuron threshold > 255 , is to have $\alpha \% 255 = 0$, i.e. have the threshold be a multiple of 255.

Naturally this additional constraint reduces the precision with which the non-spiking weights can be represented but presents a viable way to program the parallel encoding and so was the setup used for the testing of the parallel Kalman filter. The implications of this will be explored in section 2.4.4.

The parallel corelet design, for multiplying an input spike train, takes up an entire core and

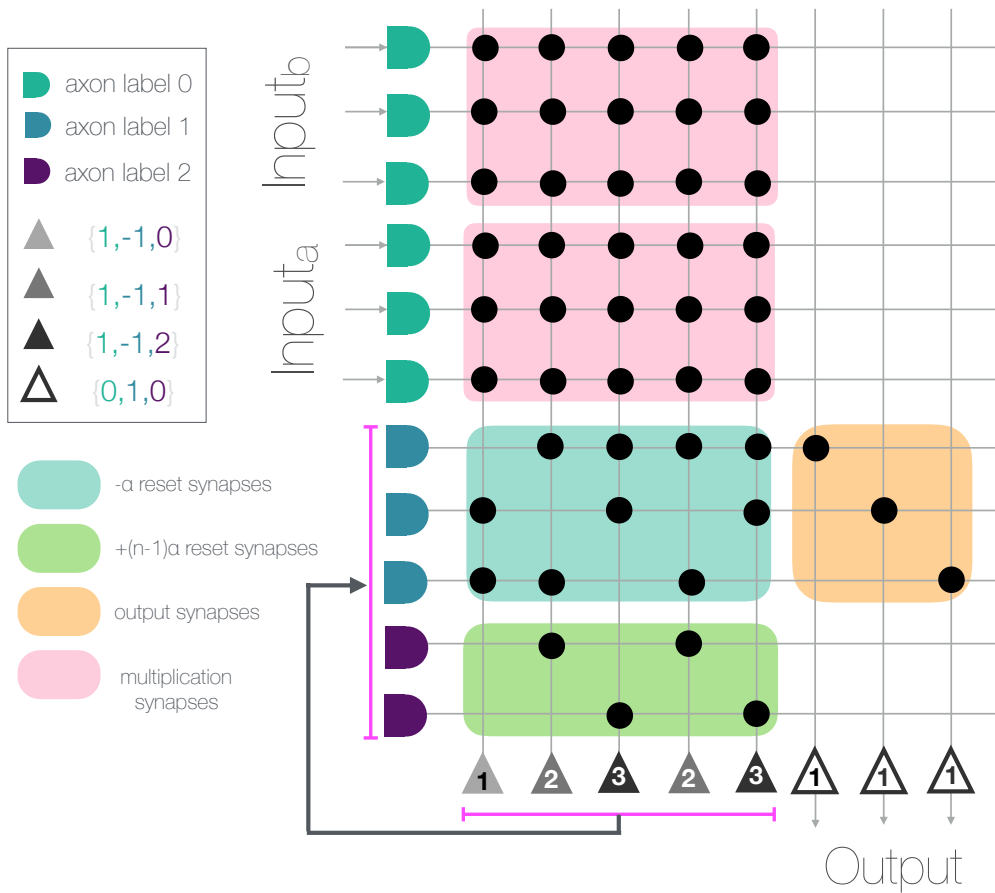


Figure 2.24: The parallel same-sign addition corelet for $b = 3$.

replaces one axon-neuron pair in the original serial crossbar: so how does this design scale for the rest of the Kalman filter?

The $\mathbf{Ax}$ portion of the crossbar can be evaluated on an element-by-element basis, in other words replacing each axon-neuron pair in the multiplication portion of the crossbar (highlighted in yellow and orange in Fig. 2.18) will not affect the calculation.

Addition does not mix the state dimensions so it can be split, one dimension at a time, into same-sign addition and addition that combines positive and negative channels. These corelets are also built from the design concepts used in parallel multiplication corelet. For example, the same-sign addition corelet for $b = 3$ is shown in Fig. 2.24, which utilises the same reset scheme as the parallel multiplier but has no need to duplicate axons to extend the weight or

threshold range of the neurons since they are both < 255 . This corelet scales, for adding two values together as:

$$\begin{aligned} \text{Axons: } & 4b - 1 \\ \text{Neurons: } & 3b - 1 \end{aligned} \quad (2.21)$$

So clearly the number of axons is the limiting factor for same-sign parallel addition.

In the case of combining positive and negative trains, or different-sign parallel addition, the crossbar is largely the same as for the same-sign addition corelet but with the extra addition-reset logic that was described in section 2.3.2. For example, the different-sign parallel addition corelet for $b = 3$ is shown in Fig. 2.25. The template for the axon-neuron usage in the general parallel case is shown in Fig. 2.25, from which it is clear that this corelet scales like:

$$\text{Axons, neurons: } 6b - 2 \quad (2.22)$$

So this corelet is always square. It is important to note that, because of the limited size of the

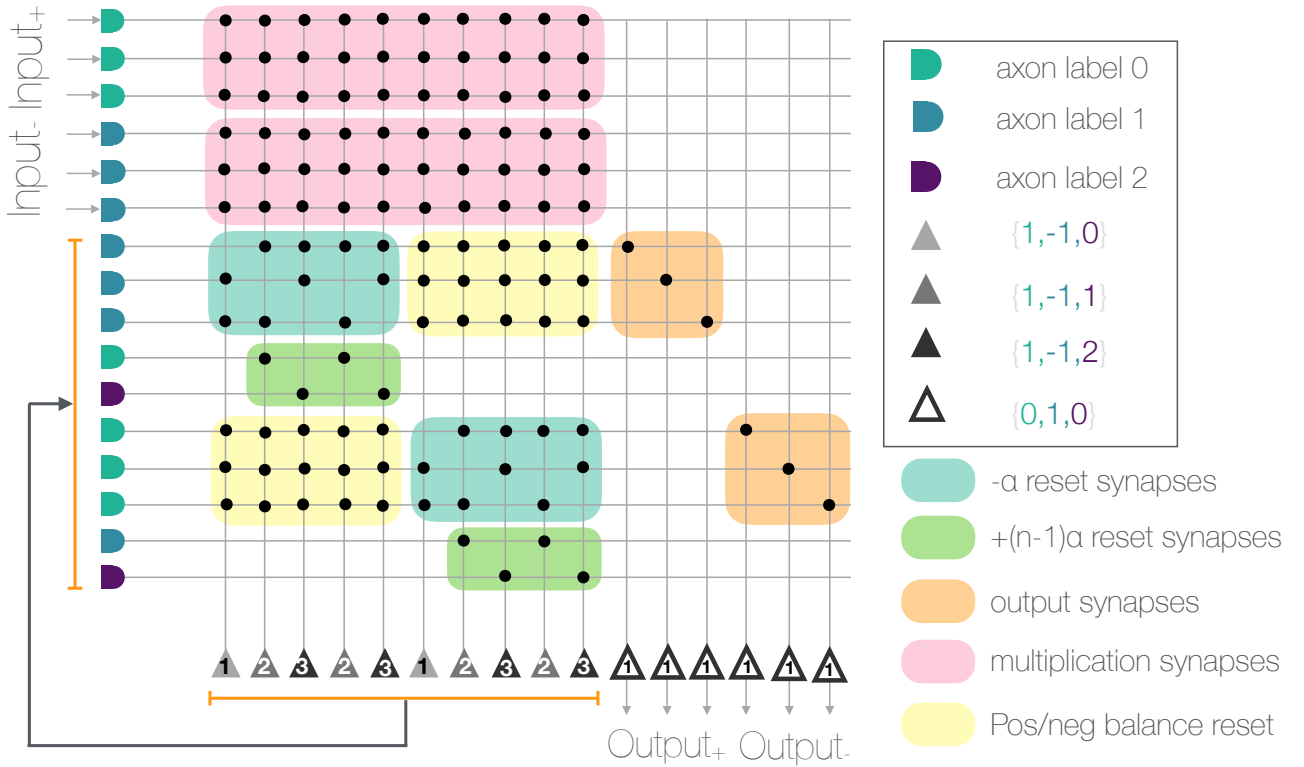


Figure 2.25: The parallel different-sign addition corelet for $b = 3$.

crossbars in TrueNorth, this version of the Kalman filter cannot be put on a single core and requires multiple splitters to use. The number of splitters, as with the number of duplications of the various corelets needed, also scales with the dimensionality of the state being tracked. The limitations of the core size on the maximum achievable accuracy and speedup of the parallel version of the Kalman filter is explored in the next section.

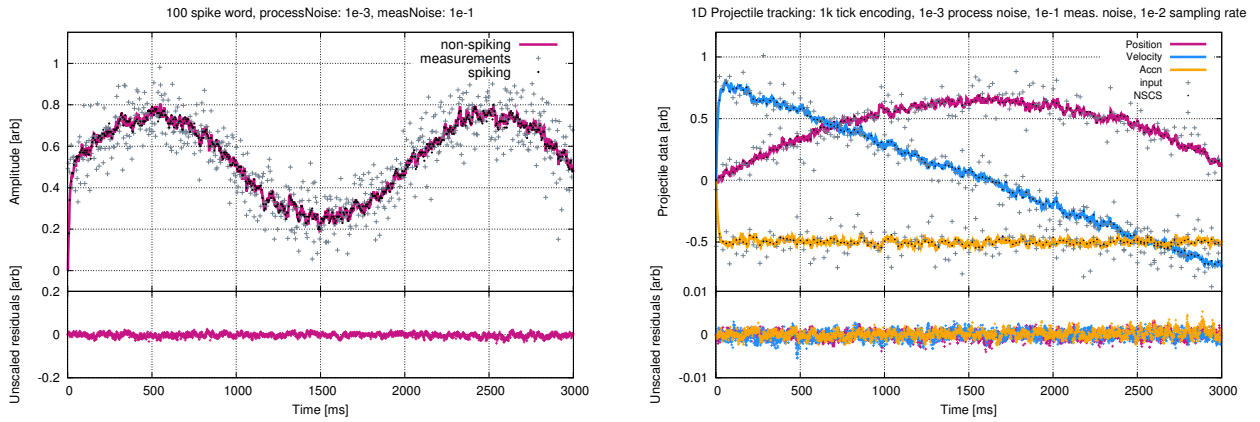
2.3.3 Test setup and simulation

To test these implementations of a spiking Kalman filter a variety of setups were used. The spiking implementation was written as a corelet and deployed on both the TrueNorth chip and NSCS simulator. Multiple tests were run to check that the output of the two matched exactly and this was always found to be the case, therefore many of the tests were run on NSCS as it is faster than running on TrueNorth with a 1 kHz tick.

During runtime, there is no way to probe registers in TrueNorth, for example the neuron potential V_j , even when using NSCS, so to ensure the processes going on in the chip matched what we thought we understood, a spiking simulation was written in Python. This simulation took the update rules described in the previous section and wrote them explicitly, thus it also ran on spikes. This turned out to be very useful and helped us to debug a number of issues that we could not have with NSCS alone. It also allowed us to try things which weren't possible with the chip simulation, for example extending the range of the registers beyond what is physically possible on the chip.

A Python simulation was also used as a baseline to compare the TrueNorth spiking estimates to. The Python Kalman filter baseline was implemented using simple linear algebra techniques.

Two datasets were generated to test the Kalman filter: a 2D system which consists of the amplitude and phase of a sine wave and a 3D system that contains the position, velocity, and acceleration of a 1D projectile, examples of both of these are shown in Fig. 2.26. Data was generated for a variety of starting conditions and combinations of measurement and process noise. The default conditions, unless specified, were a process noise of 1×10^{-3} , measurement noise of 1×10^{-1} , a timestep of 1×10^{-3} and the simulation was usually run for 1000 timesteps.



(a) An example of the 2D dataset generated, tracking the amplitude and phase of a one-dimensional sine wave.

(b) An example of the 3D dataset, tracking the displacement, velocity, and acceleration of a projectile.

Figure 2.26: Examples of the 2D and 3D datasets generated to test the Kalman filter. The grey crosses are the noisy measurements input to the Kalman filter. The solid coloured lines show the non-spiking, or ideal, estimate of the state from the non-spiking Kalman filter. The black dots show the estimate of the state from the spiking Kalman filter.

2.4 Results and discussion

This section will evaluate the performance of the steady-state Kalman filter model implemented as described in section 2.3.2 and hence explore the limitations of the TrueNorth architecture in performing linear algebra. Most of the topics explored in this section are relevant to both the serial and parallel versions of the implementation, aside from the discussion on the compromises necessary to ensure speedup through parallelism.

2.4.1 Evaluation metrics

The difference between the state as estimated by the spiking Kalman filter, $\hat{x}_{spiking}$, and the noiseless state, x , was first considered as a metric with which to evaluate the performance of the TrueNorth Kalman filter. This metric directly evaluates how well the spiking implementation filters the measurement noise and recovers the underlying state, i.e. how well it performs as a Kalman filter. However, this was deemed less useful than comparing the spiking Kalman filter output to the non-spiking Kalman filter output, which evaluates how well the spiking version of the algorithm performs when compared with its non-spiking counterpart. The spread of the residuals was settled on as the evaluation metric as it averages the performance of the tracker over many timesteps. Information about the overall structure of the residuals is lost but any biases will be referred to explicitly in the write-up. The metric is defined as:

$$\frac{1}{N} \sum_{k=1}^N (x_{non-spiking} - x_{spiking})^2 \quad (2.23)$$

or the variance of the residuals. In this text it will be referred to as the Mean Square Error (MSE).

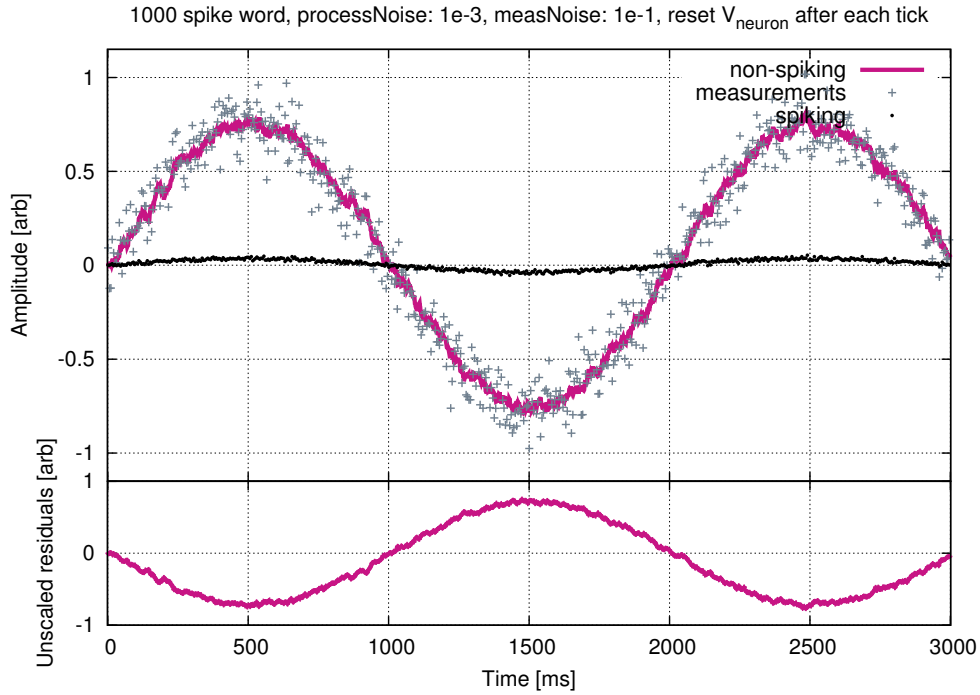
2.4.2 Neuron remainder and reset schemes

Picking the correct reset scheme for a parallel multiplier was discussed at some length in the implementation section of this chapter. In this section, the effect of other reset schemes on the behaviour of the Kalman filter will be discussed.

A reset scheme describes what happens to V_j after neuron _{j} fires. All neurons in the Kalman filter implementation use a ‘linear’ reset, where V_j is reduced by $(\alpha_0 \times \# \text{ neurons fired in block})$. In the linear reset scheme it is possible that the neuron weight is not a multiple of the threshold. In that case, after the neuron fires, there will be some remainder left over in the neuron potential which is effectively the quantization error of the weight representation in the neuron. The question, then, is whether to allow this remainder to (1) be thrown away before the next tick⁴, (2) be removed at the end of each word, (3) persist. The effect of these three reset schemes is shown in Fig. 2.27, on the 2D sine-curve dataset.

Clearly, reset scheme (1) is unusable as losing the remainder after each tick stops the filter being able to output the correct magnitude of the state, even though it shows some tracking capability - thus it is not explored further. However, previous uses of TrueNorth [51] show good performance for their intended task by setting the neuron potential to zero at the end of each word. This reset scheme, reset (2), is implemented by a complex crossbar involving set and reset pins. Although, from the single use-case shown in Fig. 2.27b it is clear that the residuals show a small bias when compared with Fig. 2.27c, how this scaled for larger encoding

⁴This would essentially force V_j to zero after it fired, something that could be implemented by setting $\gamma = 0$.

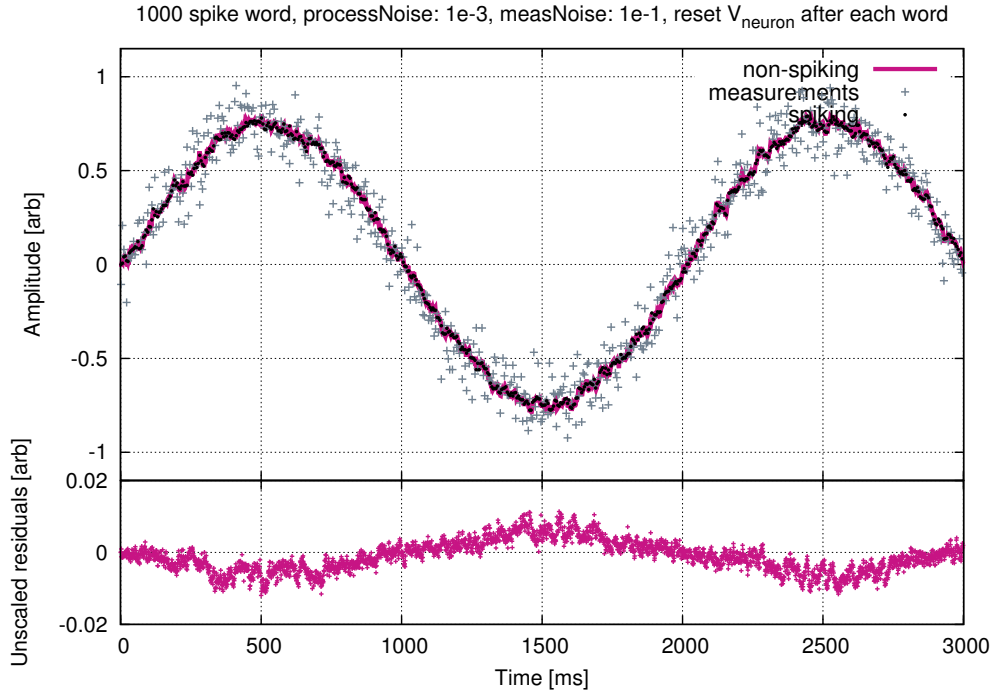


(a) V_j set to zero after every tick, i.e. remainder always discarded.

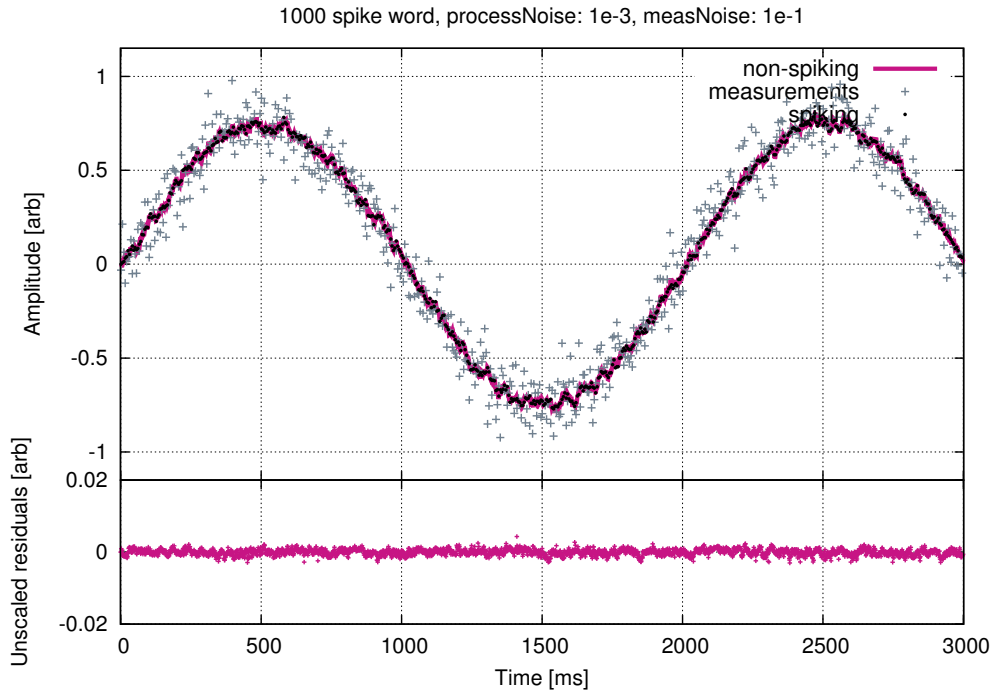
Figure 2.27: The effect of various reset schemes on the performance of the spiking Kalman filter on a 2D tracking problem, encoded with 1,000 spikes per word, using the maximum weight register size. Note the difference in the residuals scale to the other two plots in this figure.

representations was explored for both the 2D and 3D datasets, shown in Fig. 2.28. It is clear that this reset scheme consistently produces a result that is further from the ideal non-spiking estimate, regardless of the number of spikes used to represent the data word.

Since both reset schemes (1) and (2) showed worse performance when compared to the equivalent non-spiking filter than reset scheme (3), which preserves the neuron remainder, all performance tests were run with reset scheme (3) and the Kalman filter implementation was implemented using it.

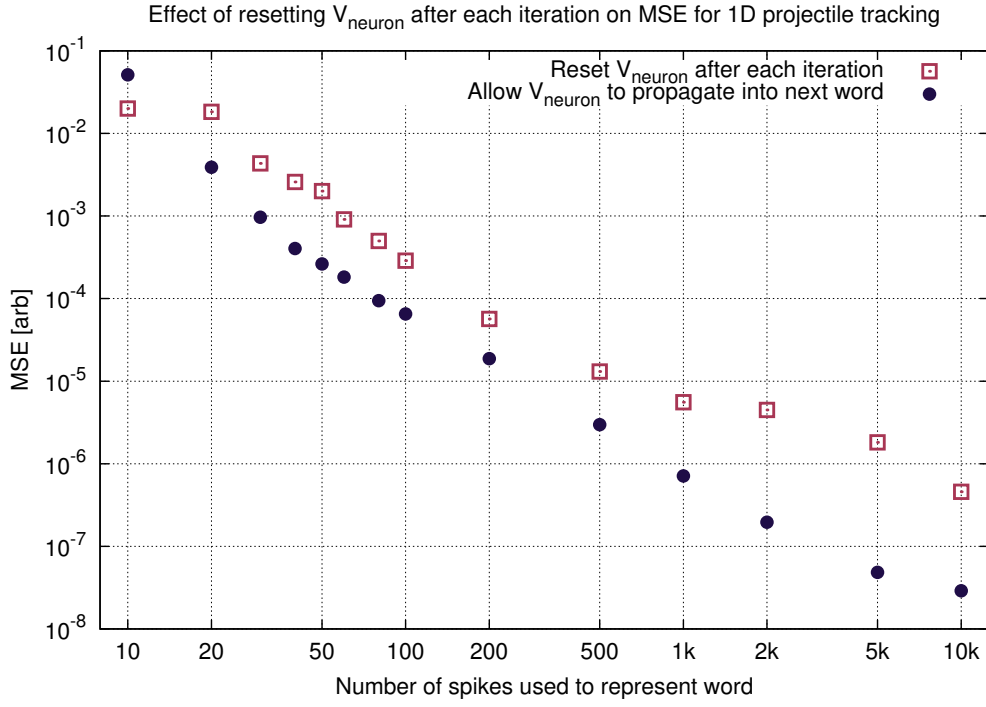


(b) V_j set to zero after every data word, i.e. remainder discarded after every word.

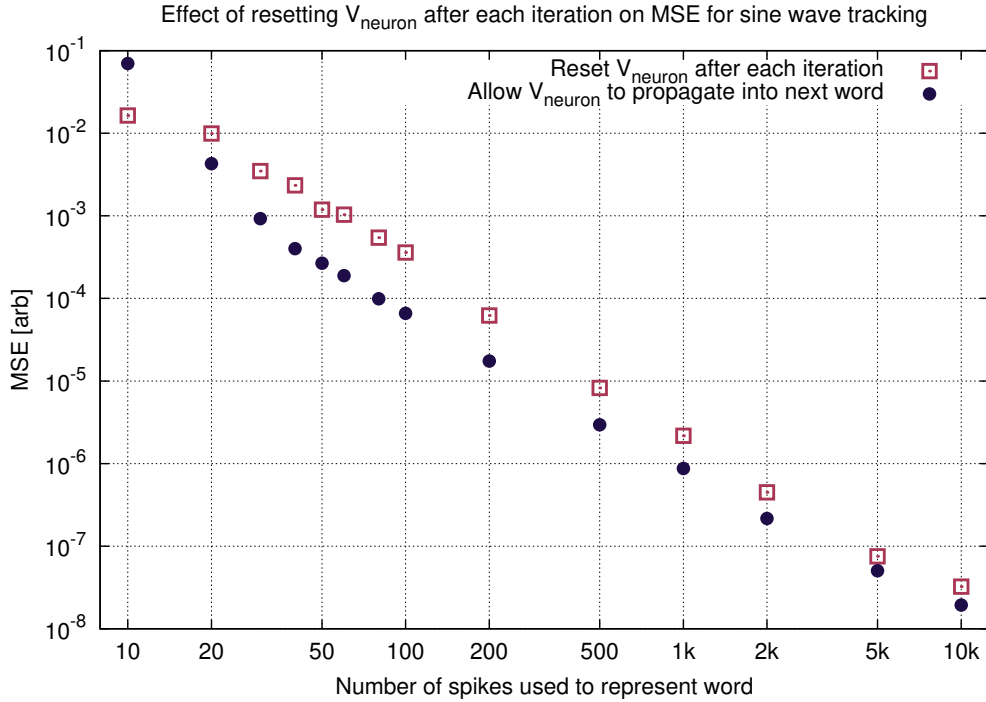


(c) V_j is linearly reset by $\alpha_0 \times \text{spikes fired in neuron block}$, i.e. the remainder persists between words.

Figure 2.27: The effect of various reset schemes on the performance of the spiking Kalman filter on a 2D tracking problem, encoded with 1,000 spikes per word, using the maximum weight register size. (*cont'd*)



(a) Comparison of reset schemes using the 3D projectile dataset. The y-axis is the variance of residuals between spiking and non-spiking Kalman filters.



(b) Comparison of reset schemes using the 2D sine wave dataset. The y-axis is the variance of residuals between spiking and non-spiking Kalman filters.

Figure 2.28: A comparison between reset schemes (2) and (3) for 2/3D datasets across a range of word-lengths measured in spikes. Consistently, for both datasets and across all ranges of encoding word, the reset (3) scheme, which allows the remainder in V_j to persist, outperforms resets scheme (2), which resets V_j after every word.

2.4.3 Input and weight representation precision

TrueNorth processes digital spikes which encode data in trains using either rate encoding or a combination of rate and population encoding. The number of spikes used to represent a given data word changes the accuracy with which that data is represented. For example, a 10-spike word will have a stepsize an order of magnitude larger than a 100-spike word. The more spikes used to encode data, the better the representation of that data and the closer the spiking Kalman filter performance approaches the non-spiking Kalman filter.

However, this effect is mitigated by how well the weights in the $\mathbf{A}$ and $\mathbf{B}$ matrices are represented. Incoming spikes in the neuron are multiplied by a neuron weight $s_j^{G_i}$ and divided by a threshold α_j , which are both integers. So the ratio of those integers is meant to be equivalent to the matrix weight - however, both the neuron weight and threshold are finite sized registers and so introduce an error on the state estimate. The combination of encoding representation and weight representation affects the accuracy of the state estimate in the Kalman filter, as is shown in Fig. 2.29 for both the 2D and 3D datasets.

Although there is a consistent improvement in the output of the spiking filter with respect to the non-spiking filter with using more spikes to encode a state, there is a limit to which this holds. This limit is dependent on how well the weight is represented in the neuron. For example, using the default neuron weight range of $[-255, 255]$, increasing the number of spikes used to encode a word beyond 1,000 does not improve the accuracy of the spiking Kalman filter relative to its non-spiking counterpart. This plot is capped at 10,000 spikes per word as, with the default tick rate, that equates to 10 s to parse an entire word into TrueNorth. The weight is capped at (1028×255) as it produces the same range of values as the threshold. The value of (256×255) is also significant as it is the maximum range that could be achieved by weight duplication, as described in section 2.3.2.

This plot shows that whilst picking a good encoding scheme will improve the accuracy of an output, this effect is mitigated by how well the weights in the system are represented.

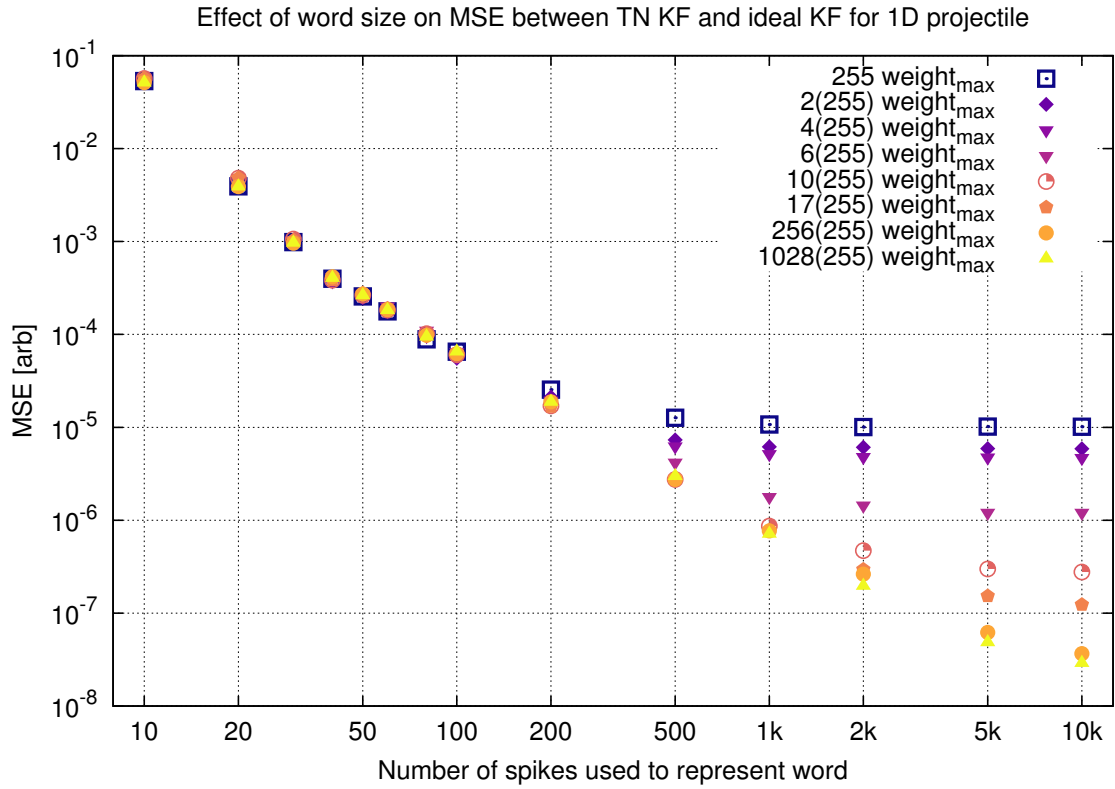


Figure 2.29: The effect of data word size, measured in spikes, on the Mean Square Error of the prediction spiking Kalman filter when compared with the equivalent non-spiking filter. The plot also shows the effect of effective weight register size, as the number of axon-neuron pairs assigned to increase the range of the weight register from 255, the performance of the spiking Kalman filter improves.

2.4.4 Latency and tick: timing limitations

Speedup through parallelism

In section 2.3.2, a way to implement the parallel Kalman filter was discussed, and the number of neurons, axons, and cores needed to implement it was explained. This section will explore effect of the finite size of the core on the available speedup and weight precision available in that parallel setup.

The limiting factor on the maximum speedup available from parallelising spike-train processing is the size of a TrueNorth crossbar. There are 256 neurons and axons available on a single core, so any design that requires more of them to be used cannot be synthesised. In the parallel Kalman filter design the corelet that uses the most neurons and axons for a given blocksize is the multiplication block, thus there is a constraint that:

$$b \left[q + \frac{p(b+1)}{2} \right] \leq 256 \quad (2.24)$$

Where b is the blocksize or parallelism, q is the amount of duplication of the input used to increase the neuron weight, and p is the amount of duplication used in the reset crossbar to extend the range of the neuron threshold that can be used.

Out of the three parallel corelets detailed: multiplication, same-sign addition, and different-sign addition, this corelet places the largest spatial constraints on the implementation. However, this equation assumes that the weight representation precision should be maximised. If we assume the minimum-possible weight representation precision, i.e. that:

$$p = q = 1 \text{ and also } \alpha_j < \frac{255}{b-1}$$

then this formula reduces to:

$$\begin{aligned} 3b - 1 &\leq 256 \\ \therefore b &\leq 85 \end{aligned}$$

However, in this scheme of minimum weight-precision, the different-sign parallel addition corelet becomes the limiting factor:

$$\begin{aligned} 6b - 2 &\leq 256 \\ \therefore b &\leq 43 \end{aligned}$$

So with the minimum possible weight representation in the multiplication core the maximum speedup that can be attained through parallelism is 43.

If the constraints on minimising the weight precision in favour of speedup begin to be relaxed then the condition:

$$p = q = 1$$

remains. In other words because the threshold is no longer less than $\frac{255}{b-1}$, the positive reset axons in the crossbar will have to be duplicated. In that case the maximum blocksize, and hence speedup attainable is limited by the size of the multiplication core:

$$\begin{aligned} \frac{b(b+3)}{2} &\leq 256 \\ \therefore b &\leq 21 \end{aligned}$$

If the constraint that $p = 1$ is also relaxed, i.e. that the threshold (and denominator of rational representation of matrix element weight) can be > 255 , then the maximum attainable speedup changes again, according to equation 2.24. This is plotted for all possible values of p and b , given a 256×256 core in Fig. 2.30.

From this plot it can be seen that the maximum weight resolution attainable with a speedup of 2 is: 5×10^{-5} , with a speedup of 3 that reduces to 1×10^{-4} , etc. The reason there are bands of similar-resolution representation for different speedup, and vice-versa, is because of the constraint that $p, b \in \mathbb{Z}$, and that equation 2.24 is non-linear. Practically this means that there are bands where the latency of the system can be reduced by adding more neuron-axon pairs without penalising the precision of the weight representation. And equivalently, there exist ranges where the weight precision can be increased without reducing the speedup due to parallelism.

It is important to note that these plots represents the minimum stepsize in weight attainable for a given speedup, but that that stepsize might not always be the optimal representation of the weight element as some axons might need to be dedicated to increasing the range of the numerator in the rational representation of the weight beyond 255, i.e. setting $q > 1$. Appendix A.2 provides the p, q, s, α_0 search used for setting up the multiplication corelet for a given b . Note that if $b \in 2, 3$ this algorithm can be modified as there are enough axon labels available for relaxing the constrain that $\alpha \% 255 = 0$.

Increasing the tick rate

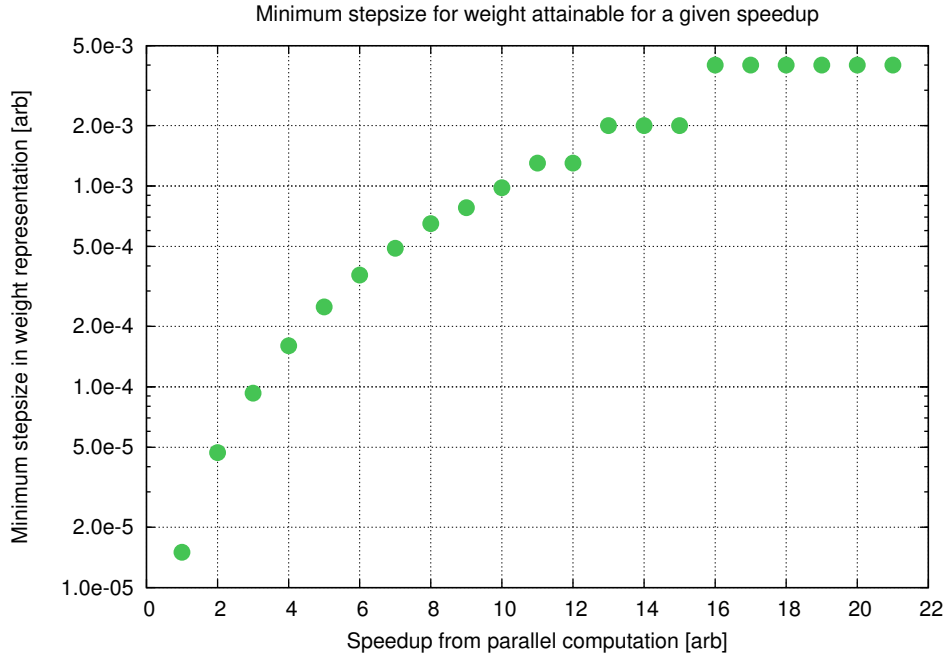
The tick is TrueNorth’s global synchronization point and a number of operations must be completed within the span of a tick, else the chip will throw an error state and cease operation. These error states, and the effect of increasing the tick rate on the evaluation of the Kalman filter model, will be discussed in this section.

If the situation arises where there are a high density of spikes to compute for a given core, it is possible that the neuron will not finish computing all the spikes queued in the scheduler memory before the next spike arrives. In this case, an error is thrown by the token controller which shows up as a ‘TOK’ core error. One can imagine that if a dense spike-train coupled with a parallel implementation was implemented on a core, the dense crossbar and input, for a small enough tick, would cause this error to be thrown.

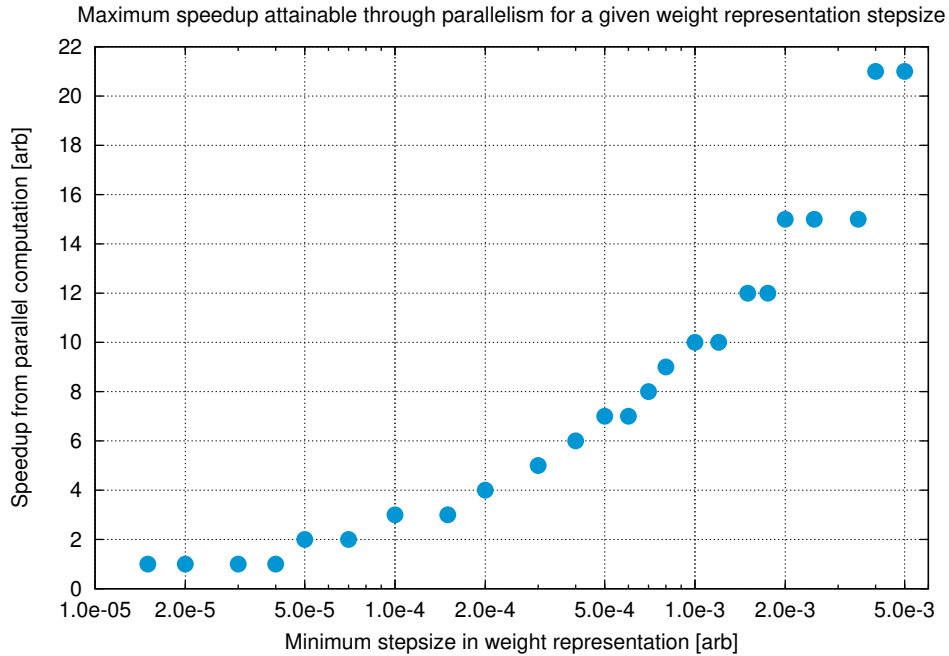
Another timing error that can occur is with routing spikes. If not all the spikes, issued from neurons in the previous tick, have arrived at the router of their destination core before the next tick is issued, a scheduler error or ‘SCH’ core error is thrown. However, these are not always caught, for example if a spike is off by a whole tick, no error will be thrown and it will simply be queued incorrectly in the wrong tick. Situations where this could occur are if the entirety of the chip is already being utilised and a spike packet has to traverse many cores to reach its destination, and/or if there are many spikes in the network that have to be routed within a given tick.

To try and understand the maximum speedup that could be achieved for this implementation of the Kalman filter, the tick rate was increased. The tick rate is generated externally and fed into the chip. On the NS1e board, the ZYNQ SOCFPGA generates the tick according to a parameter assigned in the model.conf file produced by CPE, the parameter is called “tickPeriod” and is measured in ms.

The default value for this parameter is 1, i.e. a 1 kHz global tick. A test was set up to reduce the tick period in steps to see at what point the errors described began to occur and which ones occurred first depending on model, and serial or parallel implementation. The serial



(a) As more neuron-axon pairs are dedicated to speedup through parallelizing the input, the precision with which the weight is represented decreases. Because of the non-linearity of the relationship between speedup and the number of neuron-axon pairs for extending the weight representation, certain speedups can be used without penalising the precision of the weight representation, e.g. a speedup of 12 enforces the same penalty as a speedup of 14.



(b) Equivalently, increasing the precision of weight representation reduces the number of neurons and axons available to parallelise the data and there are also increases in precision that do not penalise the speedup.

Figure 2.30: The relationship between speedup achieved through parallelizing a spike train over a population of neurons and axons and the maximum resolution for weight representation using the neuron's weight and threshold.

implementation of the 2D model as it has the lowest synaptic density and hence core usage of the models produced. To put that in perspective, the 2D serial Kalman filter can be synthesised in 3 cores: a single core for the Kalman filter corelet, and a core each for the delay and splitter corelets - which is $< 0.1\%$ of the entire cores in the chip. Also, since the 2D model was used, spike trains oscillated between being sparse and dense as the sine wave being tracked moved around 0.

First the tick rate was double to 2 kHz, but any tick rate higher did not seem achievable, although no error was thrown by the chip. Upon further investigation it turned out that the limiting factor was the chip's bandwidth. Ideally, each of the 6 ports on TrueNorth is rated for 15 Mspikes/s, which equates to around 480 Mbps bandwidth. Only two of these ports are accessible on the NS1e board so the maximum total throughput of the board is limited to 480 Mbps. In practice, what happens is that if the ZYNQ core cannot match the data rate requested by the model file, because of the limited bandwidth on TrueNorth, it simply continues to run and throttles the tick rate to match - in that way the correct computations are performed but the performance is limited by the port bandwidth (Rodrigo Alvarez, personal communication, Aug. 2016).

Again it is worth emphasizing the point that this throttling arose just beyond a tick of 2 kHz, using $< 0.1\%$ of the total chip with a serial spike train. If this minimal model cannot be sped-up there is little hope that this version of the chip can be used for latencies beyond 1 kHz for more complicated models.

2.5 Conclusion

In this project a steady-state Kalman filter implementation was designed for the TrueNorth architecture and its prediction accuracy benchmarked against a numerical implementation in Python. The aim of this study was both to understand how to approach programming tracking algorithms in TrueNorth but also to understand the compromises made to achieve low power. This section will briefly summarise the study and recommend changes to make to this or a similar architecture more suitable for use in the ATLAS experiment.

Overall it was shown that a steady-state Kalman filter can be programmed in the TrueNorth architecture. Given that each neuron in a crossbar can add, subtract, multiply, and divide, a linear algebra based approach was chosen and in the case of encoding spikes serially, was simple to implement.

However, there are a number of factors that must reach a compromise when programming in TrueNorth: latency, data representation precision, and weight representation precision. It has been shown that to achieve a simple implementation that takes a minimal amount of real estate on the chip, there is a large latency as many ticks must be used to represent the data. This latency can be reduced by reducing the number of spikes used to represent the data, but this is done at the expense of the precision to which the data is represented.

If, instead, the latency is reduced by using a parallel implementation, then the accuracy of the weight representation can be compromised. If a parallel implementation is used, at least two spikes must be placed in parallel to accurately reset the neurons. If the complex neuron scheme described in the penultimate section is not employed, then a bias is introduced to the tracking. Whilst it is possible to speed up the serial implementation by increasing the global tick rate of the chip, even the most simple model could not be sped up by more than twice the default tick rate of 1 kHz, and any, more complex, models would be unlikely to perform better.

Aside from the problematic compromises that must be made with speed and accuracy there are specific implementation details that, if improved, would greatly simplify the implementation. If the number of axon labels was increased from four to five, then the constraint that the threshold has to be an integer multiple of 255 could be lifted. This would greatly simplify the weight selection scheme and allow for better weight representation in the parallel implementation.

Whilst spiking is a fundamental aspect of neuromorphic computing, the digital spikes employed in TrueNorth lack the complexity to encode data in an efficient way and unary encoding must be used. Whilst unary encoding is an inefficient use of bits, a single bit-flip only changes the value being encoded by $+/-1$. In binary, by comparison, the placement of a bit-flip inside a word changes the value of the word by the placement of the flipped bit. Thus, unary encoding and hence spiking, is more resilient to small errors being introduced. For example, this can be seen with overflow in the Kalman filter. If two spike trains of length three ticks are added together, where one train contains three spikes and one train contains one spike, then 4 spikes total will be emitted. One spike will effectively leak into the next window but there is no register that will ‘roll over’, presenting the sum as zero. Spikes are inherently fault tolerant but it is questionable whether fault tolerance is a good compromise for the latency issues described earlier in this thesis.

In TrueNorth specifically, some issues with how the chip is programmed could be easily solved and save both space on the chip and a lot of time for developers.

One of the most inefficient uses of crossbars is the splitter. The splitter is only necessary because neurons cannot send spikes to multiple axons and it is not clear why this feature cannot be added in a newer version of the chip.

Currently, to connect two corelets together, a Connector class must be used. The Connector classes must match in dimension, there is no way to connect, for example, two of the pins on a four pin Connector and leave the other two floating. The practice of not connecting all the pins in a model, although not always advisable, is at least possible to do in other hardware description languages, such as VHDL. Instead, additional corelets must be made to convert one Connector size to another, which is a waste of the chip’s resources and adds additional latency. However, one feature that would greatly improve the developer experience is in the chip simulation. There is no way to probe the internal registers of the chip simulation during run-time. In FPGA programming, register values are routinely probed over the course of a simulation to assist in debugging logic. In TrueNorth the most important hidden register is the neuron potential, V_j . With no way to know the value of the neuron potential during run-time, a python-based model of the chip was written to emulate its functionality. Because the neuron potential was just an integer number in Python, it could be printed out and used to debug timing and connectivity logic.

Whilst the implementation of a steady-state Kalman filter in a neuromorphic architecture is possible, in a chip like TrueNorth many compromises must be made to obtain accurate tracking at a reduced latency. If neuromorphic chips are to be explored for future implementations of the Kalman filter, it seems reasonable that chips which have a higher degree of freedom, for example that make computations with analogue spikes, or that have a true LIF neuron model would be better places to start. It is also imperative, when deploying a Kalman filter, that the chip has the ability to change its weight matrix at runtime. The most commonly used Kalman filter types in track reconstruction for the ATLAS experiment, are a combinatorial and extended Kalman filter both of which have their covariance matrices updated every iteration.

Thus, it seems unlikely that a digitally spiking neuromorphic chip, such as TrueNorth, would be used to implement tracking algorithms for use in track reconstruction in the ATLAS experiment. However, the lessons learnt with respect to fault tolerance, neuron model complexity limiting

the algorithms that can be implemented, and optimal register size could be used to investigate, or even design, more suitable architectures going forward.

Chapter 3

Radiation damage modeling in Pixel detector sensors

3.1 Silicon detectors

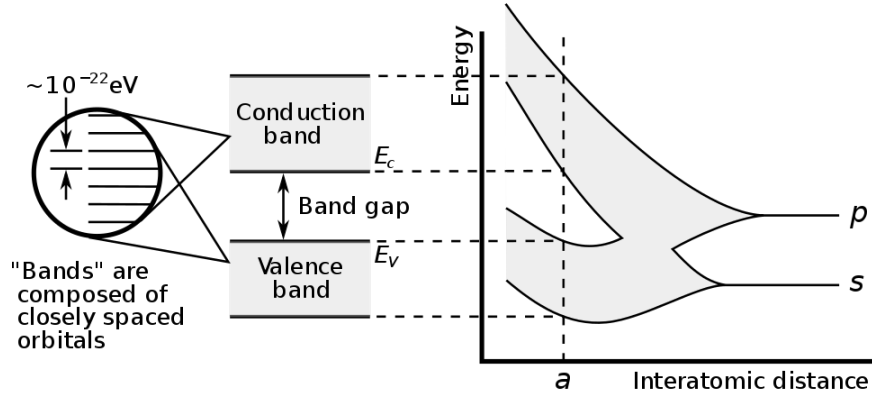
Since the 1950's, silicon has dominated the semiconductor industry for its abundance, ease of use, and its physical properties: ideally suited for creating layered junctions of varying conductivity, the building blocks of transistors.

The physical properties of silicon that make it desirable as a component in industry are not dissimilar to those that make it an excellent detector in particle physics. This chapter will first explore the properties of silicon and its use as a detecting medium, before considering the effects of high energy particles on silicon itself, and finally describing charge deposition and transport. The chapter concludes with a description of how the simulated effects of radiation damage are incorporated into the digitization software used by the ATLAS experiment as part of the simulation and reconstruction framework, Athena.

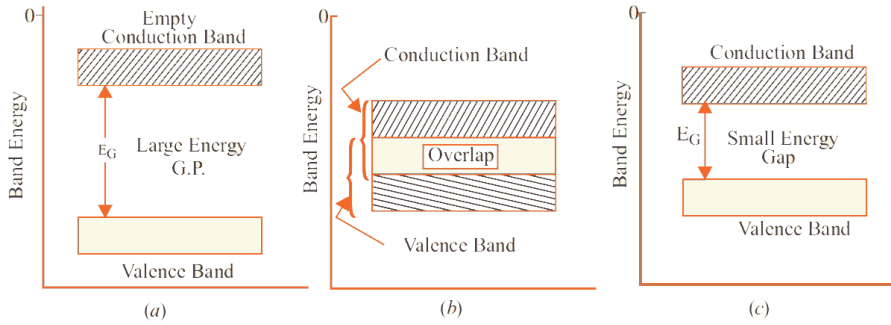
3.1.1 Semiconductors

Silicon is a semiconductor, a material that is neither as conductive as a metal nor as resistive as an insulator but that can be manipulated to behave more like either. Electrical conductivity arises from electrons being able to jump easily from atom to atom in a lattice. When atoms form a lattice, the degenerate energy levels in each atom split due to the the requirement of non-overlapping wavefunctions for Fermions. These closely spaced split levels form a band of near-continuous energy states, see Figure 3.1. The valence band is occupied by the highest energy electrons in the atom, which participate in chemical bonds. The conduction band is the energy at which charge carriers are dissociated from atoms and can move freely within the lattice. In conductors, the valence band contains only weakly bound electrons and so is also the conduction band. In non-conductors the valence and conduction bands have a separation energy that corresponds to the energy needed to break a lattice bond, this is known as the band gap. The band gap is essentially a range of forbidden energy states that electrons cannot occupy.

Insulators have a large band gap separating their valence and conduction bands, which means only large amounts of energy will liberate valence electrons from their stable bonds. Semiconductors have a smaller band gap than an insulator, on the order of eV, so if energy is introduced to the lattice, electrons in the valence band will be able to occupy the conducting band and



(a) As interatomic distances are reduced, degenerate energy levels in the lattice split into closely spaced bands [52].



(b) Relative valence and conductance band spacing for insulators, conductors, and semiconductors [53].

Figure 3.1: Band structure properties of various materials.

move freely. Likewise, the space left behind when an electron leaves the valence band, a hole, can move between atoms in the lattice. The hole is a lack of electron and so can effectively be considered a positive-charge carrier.

The mean number of electrons occupying the conduction band in a semiconductor, for a given temperature, can be calculated using Fermi-Dirac statistics. This is also known as the density of free charge carriers or the charge carrier concentration and is proportional to the conductivity of the semiconductor. The density of free charge carriers in the lattice is given by:

$$n_i = \int_{E_c}^{\infty} N(E)F(E)dE \quad (3.1)$$

where E_c is the lowest energy, or bottom, of the conduction band, $N(E)$ is the density of states in the conduction band, and $F(E)$ is the Fermi-Dirac function. The Fermi-Dirac function describes the probability of the state at energy E being occupied for a given temperature, T :

$$F(E) = \frac{1}{1 + e^{(E-E_F)/kT}} \approx e^{-(E-E_F)/kT} \quad (3.2)$$

where k is Boltzmann's constant, T is the temperature, and E_F is the Fermi level's energy or chemical potential. Another way of viewing the formula is to say that E_F is the energy, for a given temperature, at which there is a 50% chance that an electron is occupying it. In reality, E_F usually lies in the band gap and the Fermi-Dirac distribution is used to calculate the occupancy of the conduction band.

In silicon it can be shown that $(E_c - E_F) > 10 kT$ holds till ~ 650 K, so the approximation on the right hand side of equation 3.2 holds. For an equal carrier concentration of electrons and

holes in a semiconductor, an expression can be found for the intrinsic Fermi level. It can be shown that the intrinsic Fermi level lies halfway between the conduction and valence bands at room temperature. From the density of states:

$$N(E)dE = 4\pi \left(\frac{2m_q^*}{h^2} \right)^{2/3} E^{1/2} dE \quad (3.3)$$

where m_q^* is the effective mass of the charge carrier and h is Planck's constant. Substituting and evaluating equations 3.3 and 3.2 in equation 3.1 gives the concentration, n_{qi} of free charge carriers:

$$n_{qi} = 2 \left(\frac{2\pi m_q^* kT}{h^2} \right)^{3/2} e^{-\delta E/kT} \quad (3.4)$$

where $\delta E = \begin{cases} E_c - E_F, & \text{for electrons} \\ E_F - E_v, & \text{for holes} \end{cases}$

where E_v is the energy at the top of the valence band. If the lattice is electrically neutral then the number of free electrons will equal the number of free holes, so from equation 3.4, the Fermi level is:

$$E_{Fi} = \frac{E_c - E_v}{2} + \frac{3kT}{4} \ln \left(\frac{m_h^*}{m_e^*} \right) \quad (3.5)$$

At 300 K, the second term in equation 3.5 is three orders of magnitude smaller than the first term and thus the Fermi level is essentially halfway between the valence and conduction bands, see Figure 3.2. Equation 3.5 shows the Fermi level getting closer to the conducting band with

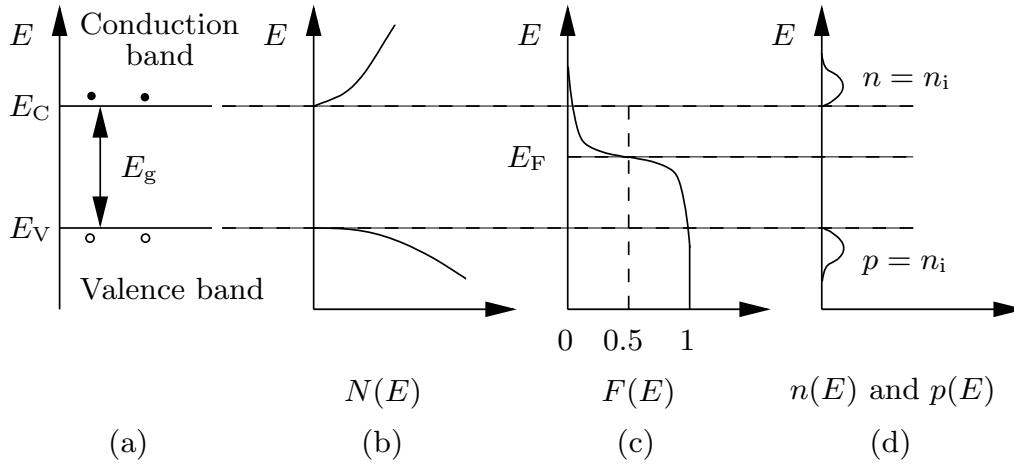


Figure 3.2: Properties of an intrinsic semiconductor above 0 K, (a) band structure showing thermally liberated charge carriers, (b) density of states, (c) Fermi function, (d) the electron and hole concentrations [54].

increasing temperature. Another way of interpreting this is that the number of free charge carriers increases with temperature. The intrinsic charge carrier concentration scales as:

$$\begin{aligned} n_i^2 &= n_e \cdot n_h \\ &= 4 \left(\frac{2\pi kT}{h^2} \sqrt{m_e^* m_h^*} \right)^3 e^{-E_g/kT} \end{aligned} \quad (3.6)$$

where n_i is the intrinsic carrier concentration and E_g is the band-gap. In silicon the band-gap is 1.12 eV and at 300 K the effective mass of the electron and hole are 1.09 and 1.15 times the electron rest mass, respectively. Thus at 300 K the intrinsic carrier concentration is $\sim 1 \times 10^{10} \text{ cm}^{-3}$.

Doping

Intrinsic semiconductors have an equal number of holes and electrons and are made of one element. However, it is possible to change the relative carrier concentration such that either holes or electrons are the dominant charge carriers. This process is called doping.

Doping replaces a number of atoms in the lattice with elements that have either one less or one more electron in their outer shell. In the case of silicon this means that atoms in the lattice will be switched with either group 3 or group 5 atoms, usually phosphorus or boron. If a group 3 atom is used there will be one less valence electron available to participate in bonding, effectively creating an additional hole. The dopant atom is thus known as an electron acceptor. If a group 5 atom is used, there will be an extra electron present in the valence band not used for bonding and the dopant atom is known as an electron donor. Thus adding either a donor or acceptor dopant to the lattice will change the carrier concentration such that $n_e \neq n_h$. A typical dopant concentration, for example in doped silicon for use in high energy physics, is between $10^{12} - 10^{19}$ [55]. From equation 3.6, this means that there are 10^{2-9} more of the dominant charge carriers and 10^{2-9} less of the minority charge carriers. Donors effectively

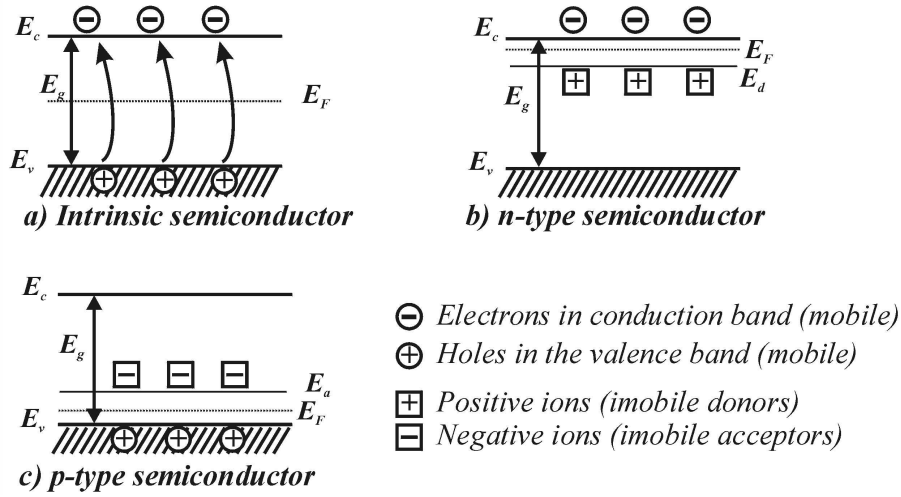


Figure 3.3: The effect of dopants on the Fermi level in semiconductors [56].

introduce an energy level close to the conduction band which, from equation 3.4, increases the Fermi level, reducing the number of holes acting as charge carriers. A material doped with a donor is known as n-type. This can also be thought of as there being more free electrons available to recombine with holes. Conversely, acceptors introduce an energy level close to the valence band, decreasing the Fermi level and reducing the number of free electrons, as can be seen in Fig. 3.3. A material doped with acceptors is known as p-type.

3.1.2 The pn-junction

A pn-junction is the metallurgical transition in a material between a p-type semiconductor and an n-type semiconductor, see Fig. 3.4, and is the basis of a diode. In thermal equilibrium a pn-junction has a net electrical current of zero but charge continuously flows across the junction boundary. The net current can be broken down into two equal and opposite components: drift and diffusion.

The n-type material has an excess of free electrons compared to an intrinsic semiconductor and the p-type material has an excess of holes. During the junction formation, free charge carriers start off in a region of higher concentration and, through random thermal motion, will diffuse

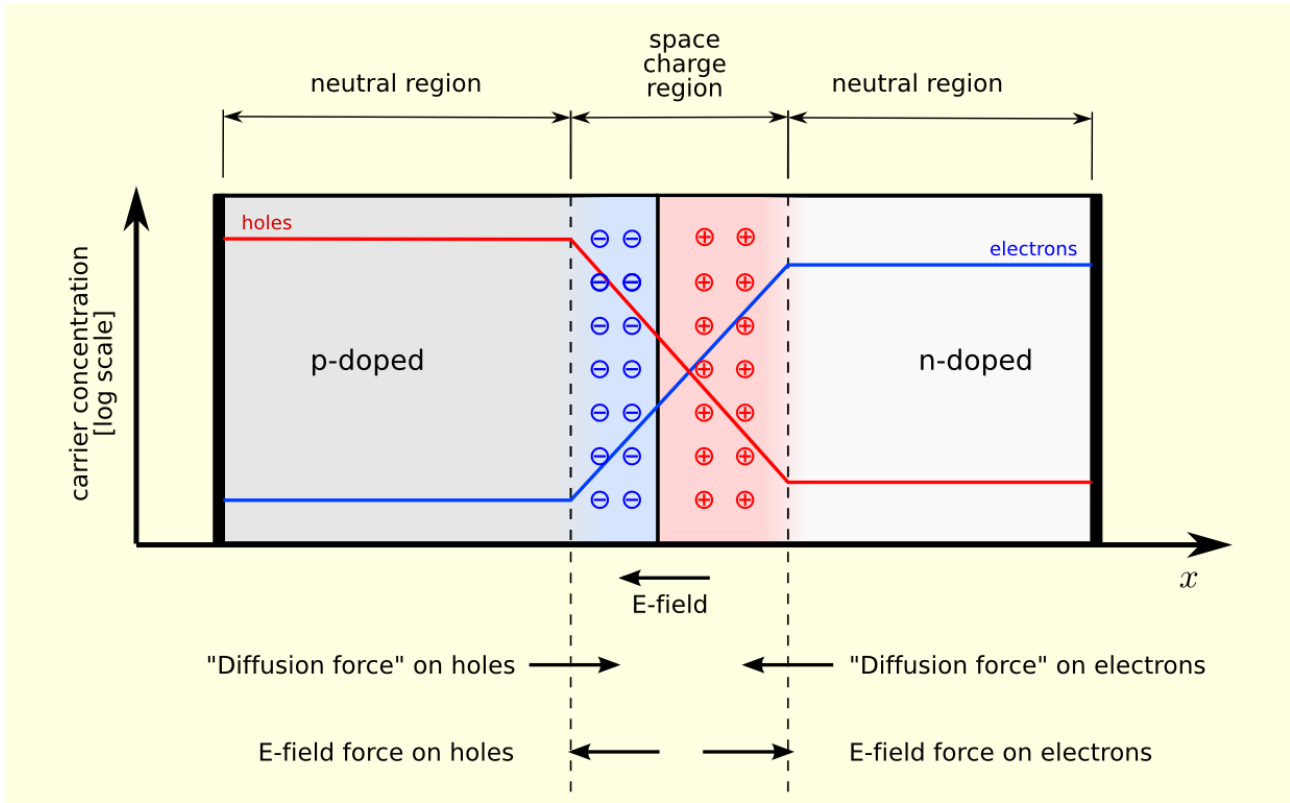


Figure 3.4: Cartoon of a pn-junction's band diagram in thermal equilibrium. The blue and red lines represent the free electron and hole concentrations, respectively [57].

across the junction. This diffusion process is characterised by the Einstein relation:

$$\vec{J}_{\text{diff}} = D_q \vec{\nabla} n_q \quad (3.7a)$$

$$D_q = \frac{kT}{e_q} \mu_q \quad (3.7b)$$

where J_{diff} is the diffusion current, D is the diffusion constant defined in equation 3.7b, ∇n_q is the gradient of the concentration of the charge carrier, q , e_q is the electric charge, and μ_q is the mobility of the charge carrier.

As diffusion proceeds across the junction, a space-charge is built up around it. This occurs because the free charge carriers diffuse away from their parent atom and leave behind ions in the lattice that are fixed in space. As the majority charge carrier in an n-type material is an electron, the space-charge region in the n-type material will be positively charged, and in the same way the p-type space-charge region will be negatively charged.

As many free charge carriers diffuse across the material boundary, a space-charge region will build up around the pn-junction. This results in an intrinsic potential difference across the junction, see Fig. 3.4. The potential difference caused by thermal diffusion is known as the 'built-in' potential, V_{bi} , and is characterised by:

$$V_{\text{bi}} = \frac{kT}{e_q} \ln \left(\frac{N_D N_A}{n_i^2} \right) \quad (3.8)$$

where e is the fundamental electric charge, N_D is the n-type (donor) dopant concentration, N_A is the p-type (acceptor) concentration, and n_i is the intrinsic carrier concentration. This formula assumes that all dopant atoms have been ionised by diffusion. If this is not the case N_D and N_A are replaced with their respective free charge carrier concentrations. The potential

difference across the junction will cause minority free charge-carriers in the material to traverse the pn-junction. For example, the built-in potential gradient goes from high to low from the n-type material towards the p-type. This means that electrons placed in that field will travel from the p-type material towards the n-type material, in the opposite direction to the diffusion current. This drift current is given by:

$$\vec{J}_{\text{drift}} = e_q n_q \vec{v}_q = e_q n_q \mu_q \vec{\nabla} V_{\text{bi}} \quad (3.9)$$

where e is signed charge, n is the free charge carrier concentration, v is velocity, μ is the mobility, and q denotes that these values are different for holes and electrons.

With no external fields applied, the junction is in a thermal equilibrium, where the diffusion current is equal to the drift current. The space-charge region has less free charge carriers than the surrounding lattice, thus this region is said to be ‘depleted’ of free charge carriers and is often called the depletion region.

Applying an external bias

The diffusion current in thermal equilibrium is limited by the built-in potential which is caused by the build-up of space-charge around the pn-junction. This diffusion current can be increased or suppressed by applying an external bias voltage to the junction. If a bias voltage is applied such that the p-type material is at a higher potential than the n-type material, an electric field will be created which points in the opposite direction to the built-in electric field. Increasing the value of the bias voltage will cause the built-in voltage to be cancelled out and allow free charge carriers to flow freely across the junction under diffusion and in the drift of the applied field. This setup is known as forward biasing, see Fig. 3.5a. Forward biasing effectively gives

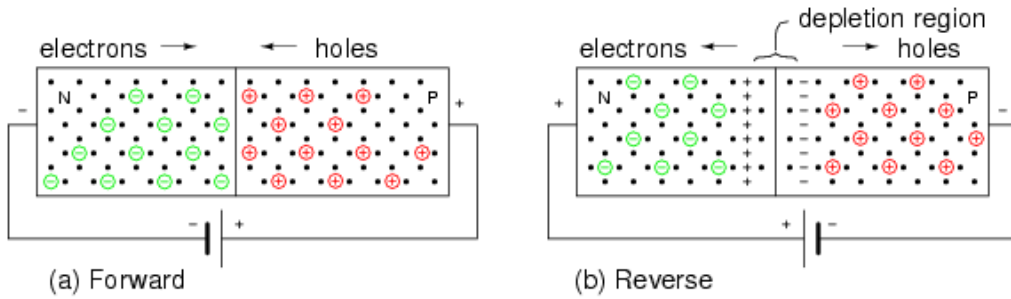


Figure 3.5: Applying a bias voltage across the pn-junction either causes it to behave like a conductor, forward bias, or increases the depletion region, reverse bias [58].

electrons and holes enough energy that they can traverse the pn-junction and recombine with holes, creating a continuous current. Applying a forward bias changes the electrical properties of a diode, causing it to behave like a conductor and removing the depleted space-charge region.

If instead, the bias voltage is applied such that the n-type material is at a higher potential than the p-type material, the pn-junction is said to be reverse biased. This causes an electric field which adds to the electric field from the built-in potential, further suppressing diffusion across the boundary. Under reverse bias, the excess free charge carriers in each material move away from the pn-junction, increasing size of the space-charge region and further depleting it of charge carriers, see Fig. 3.5b.

When the bias voltage reaches a critical value called the depletion voltage, V_{dep} , the depletion region extends across the entire depth of the material. At full depletion the number of free charge carriers in the bulk is minimised. This is equivalent to suppressing the diffusion current,

or thermal noise, of the intrinsic junction. When the junction is initially reverse biased, any minority carriers on either side will flow across the junction causing a current spike. However, after this initial flow of current, the junction effectively acts like an insulator. Current only flows across the diode if an external source of charge carriers is injected into the lattice under reverse bias.

The width, w , of the depleted space-charge region is related to the applied bias voltage by equation 3.10, which derives from solving Poisson's equation for the junction:

$$w = \sqrt{\frac{2\epsilon}{e_q}(V_{bi} - V_{bias}) \left(\frac{N_A + N_D}{N_A N_D} \right)} \quad (3.10)$$

where ϵ is the material permittivity, V_{bi} is the built-in potential at thermal equilibrium, and V_{bias} is the applied bias voltage. This relationship simplifies further for the case when one carrier concentration exceeds the other by orders of magnitude. For example, in the case of $N_A \gg N_D$, the depletion region extends further into the n-type material than the p-type material for a given bias voltage. The width of the depletion region is then given by:

$$\begin{aligned} w &= \sqrt{\frac{2\epsilon}{e_q \cdot N_D}(V_{bi} - V_{bias})} \\ &= \sqrt{2\epsilon \cdot \mu_e \cdot \rho_d(V_{bi} - V_{bias})} \end{aligned} \quad (3.11)$$

where μ_e is the electron mobility, and ρ_d is the resistivity of the n-type material.

Diode properties

Finding the depletion voltage *a priori*, in devices constructed from pn-junctions, such as the diode, is not always practical as it relies on knowing the junction properties to a good accuracy. In real devices the full depletion voltage can be found by measuring the capacitance, C , of the junction:

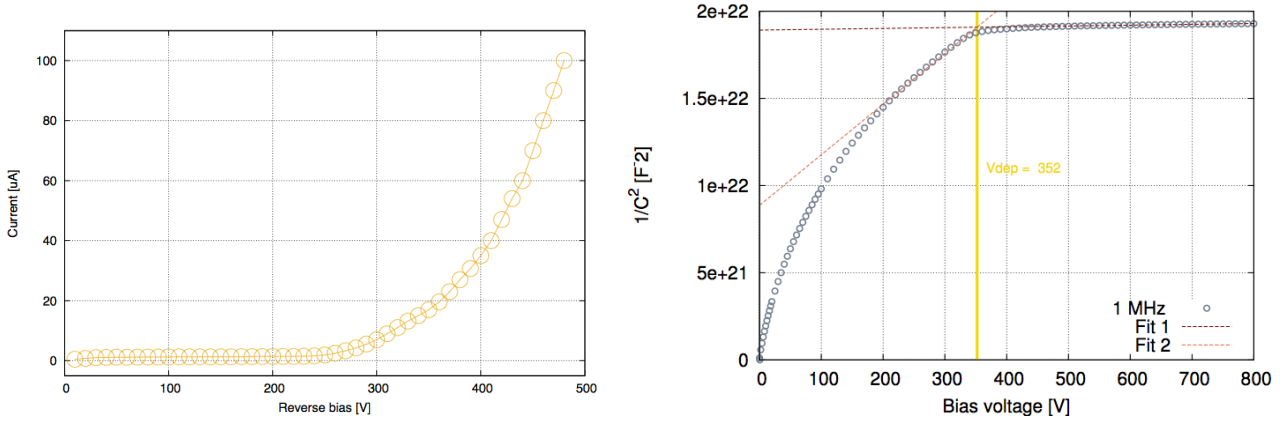
$$C = \frac{\epsilon}{d} = \sqrt{\frac{\epsilon \cdot e_q \cdot N_D}{2} \cdot \frac{1}{(V_{bi} - V_{bias})}} \quad (3.12)$$

As the space-charge region grows, more non-conducting material is effectively placed between conducting (undepleted) regions, and the capacitance decreases. When full depletion is reached the capacitance of the diode saturates. As such, a CV characteristic of a diode reveals the full depletion voltage, an example of this is shown in Fig. 3.6b.

Over-depleting, or reverse-biasing the diode at a voltage greater than V_{dep} , initially increases the charge carrier velocity but eventually that too reaches a saturation value, v_s . The mobility thus decreases with increasing electric field strength:

$$\mu = \frac{v_s}{E_c (1 + (E/E_c)^\beta)^{1/\beta}} \quad (3.13)$$

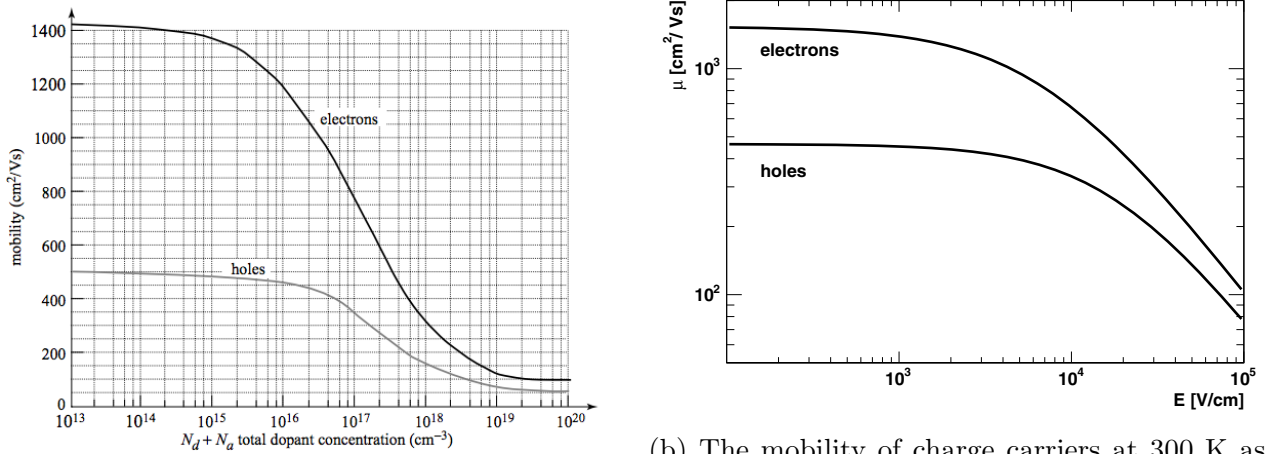
where v_s is the saturation velocity, or maximum velocity with which a charge carrier can move in the bulk, E is the electric field magnitude at a specific point in the bulk, and β and E_c are fit parameters of the model [59]. As the velocity saturates and the electric field increases, the mobility drops off, which is shown in Fig. 3.7b. This behaviour is intrinsically linked to the mean free path the charge carrier takes before it collides with a lattice atom. As the electric field strength across the pn-junction increases, so too does the velocity of the charge carrier, but the mean free path stays the same. Thus, eventually lattice collisions become the limiting



(a) Example IV characteristic of 200 μm planar silicon pixel sensor, sensor is reverse biased. Breakdown occurs ≈ 280 V.

(b) Example CV characteristic of a silicon diode. V_{dep} of the diode is found by fitting the knee in the curve, sensor is reverse biased.

Figure 3.6: Diagnostic characteristics of a planar pixel detector (a) and simple diode (b).



(a) The mobility of charge carriers in doped silicon at 300 K with no external bias applied [60].

(b) The mobility of charge carriers at 300 K as a function of Electric field strength, which is calculated with equation 3.13[54].

Figure 3.7: The effect of doping concentration and electric field on charge carrier mobility.

factor on charge carrier velocity and the mobility decreases. This phenomenon is temperature dependent, decreasing the temperature increases the mean free path, and with it the saturation velocity.

Similarly, increasing the effective carrier concentration also decreases the mobility of the charge carrier, as collisions with the ionised bulk following depletion disrupt the passage of the charge carriers. This is characterised by the phenomenological relation:

$$\mu = \frac{\mu_{\text{max}} - \mu_{\text{min}}}{1 + (N/N_{\text{ref}})^\alpha} + \mu_{\text{min}} \quad (3.14)$$

where N is the dominant carrier concentration and μ_{max} , μ_{min} , N_{ref} , and α are parameters found by fitting to experimental data [61]. Thus keeping the carrier concentration low by using a high-purity material with a low dopant concentration for the bulk of the diode will keep the mobility of charge carriers high.

The bias voltage cannot be increased indefinitely. After a certain value of V_{bias} , the junction potential will break down due to electrons being accelerated to such high energies they end up liberating other electrons from the lattice. These liberated charge carriers also have significant

energy and liberate others creating an avalanche of charge carriers flowing across the junction: a runaway, or ‘breakdown’, current. Breakdown itself is not destructive but if significant current flows then the heat dissipated by the charge carriers can deform the lattice permanently and destroy the junction [62]. An example of this is the current-voltage characteristic of an n-type bulk diode made of pixelated silicon shown in Fig. 3.6a, which exhibits breakdown starting from 280 V.

In the next section the physical properties of a reverse-biased diode will be used to construct a solid-state ionisation chamber for detecting charged particles.

3.1.3 Silicon pixel detector

The reverse-biased silicon diode forms an excellent basis for a solid-state ionisation chamber. The potential difference across the junction depletes charge carriers from the diode. This means that a small amount of injected charge will produce a measurable current that would otherwise be lost in thermal noise from the undepleted motion of thermally liberated charge carriers. A charged particle traversing the diode’s bulk will impart energy to the lattice, liberating charge carriers which move in the electric field and produce a current. The current is then integrated by a charge sensitive amplifier and recorded. In this section, specific modifications to the basic diode, described in the previous section, to build a particle detector will be discussed.

Junction manufacturing

In the previous section, the relative carrier concentrations of the dopants forming the pn-junction were discussed but it was assumed that the physical dimensions of each material were the same. However the silicon detectors used in modern particle physics consist of a bulk of one dopant type and a thin implant of the other. One of the reasons for this is to do with how such devices are commercially manufactured. In the years following the invention of the junction transistor, which, like the diode, is based around the properties of the pn-junction, methods to quickly produce junctions with controlled doping concentrations were explored [62]. The grown junction method involves extracting semiconductor from a melt in a single crystal structure called an ingot. The silicon melt is infused with a specific carrier concentration. Halfway through the growth process the dominant dopant type is switched by adding a new material to the melt, the growth then progresses, see Fig. 3.8a. After cooling the ingot can be cut to give the desired p-n geometry.

The grown junction method was discontinued in favour of the alloy junction method which is more suited for mass production. The alloy junction method starts with a singly-doped crystal which is sliced to a desired thickness. A pellet of the opposite dopant type is placed on the crystal and the materials are heated until a molten boundary is formed at the junction. When the crystal cools it recrystallises with the doped material in the pellet infused in the bulk as an alloy, see Fig. 3.8b.

The electrical properties of semiconductor junctions are highly dependent on the concentration of the relative charge carriers in the device, as was seen in the previous section. However, the amount of recrystallisation that occurs is hard to control and so this method was eventually replaced by planar technology. Planar technology uses solid-state diffusion of a dopant into the surface of the bulk. The depth to which the dopant penetrates is easily controlled and sections of the bulk can be masked off using a SiO_2 mask, see Fig. 3.8c. Thus, modern commercial semiconductor processes consist of a bulk-doped material with a planar layer of the opposite dopant material diffused on top. This forms the basis for a modern diode or silicon sensor.

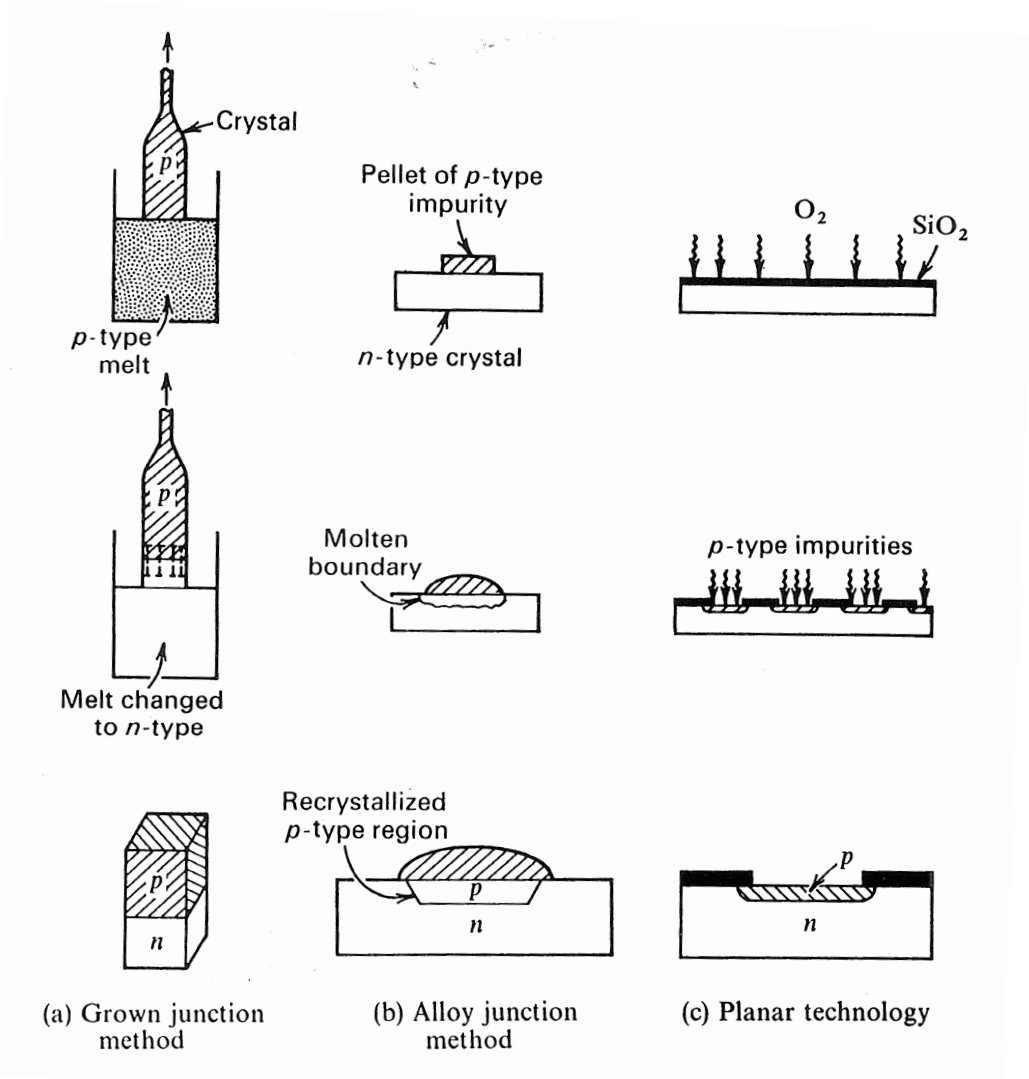


Figure 3.8: The evolution of pn-junction growth techniques. Rows progressively show the production of the junction whilst each column shows a different technique, including (a) the grown junction method, (b) the alloy junction method, and (c) the planar method [62].

Ohmic contacts

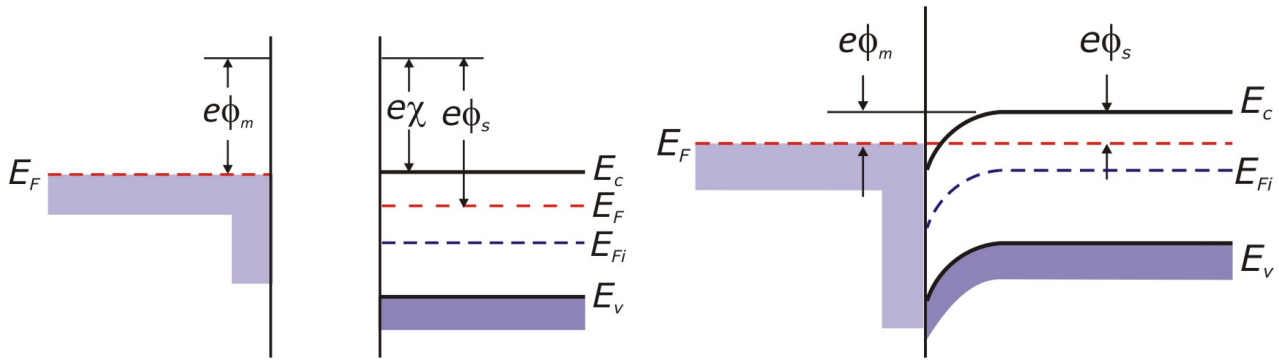
To bias a semiconductor diode, and create a depleted space-charge region, metal electrodes are attached to the front and backside of the junction. When a metal and a semiconductor are brought into contact the relative difference between the work function¹, ϕ_{metal} , of the metal and electron affinity², $\chi_{\text{semiconductor}}$, of the semiconductor create a band-gap characterised by the barrier potential, ϕ_{barrier} :

$$e\phi_{\text{barrier}} = e(\phi_{\text{metal}} - \chi_{\text{semiconductor}})$$

For good conductance between the semiconductor and the metal, the metal should be chosen such that the barrier potential is as small as possible. This means finding a metal work function that closely matches the semiconductor's electron affinity. If the barrier potential is large

¹The work function is the minimum energy required to remove an electron from a material and into the vacuum.

²The electron affinity of a semiconductor is the energy required to release an electron from the conduction band into the vacuum.



(a) The energy bands in a metal (left) and an n-type semiconductor (right), compared side-by-side. The work function of the metal is less than or equal to the electron affinity of the semiconductor.

(b) An Ohmic junction formed by a metal and n-type semiconductor. The semiconductor's conductance lower bound drops towards the junction, allowing charge carriers to easily flow across the boundary.

Figure 3.9: The formation of an Ohmic junction from a metal and an n-type semiconductor. In the diagrams E_c is the bottom of the conduction band, E_F is the Fermi level of the n-type material. E_{Fi} is the Fermi level of the intrinsic semiconductor, E_v is the top of the valence band, ϕ_m , ϕ_s are the work functions of the metal and semiconductor, and χ is the electron affinity of the semiconductor. The purple blocks indicate continuous bands [63].

compared to the semiconductor's conducting band, a Schottky rectifying junction will be formed where current will only flow by applying a forward bias between the metal contact and the semiconductor.

Ideally, for an n-type bulk, the metal contact must have a work function, $e\phi_{\text{metal}}$, that is less than or equal to the electron affinity, $e\chi_{\text{semiconductor}}$ of the semiconductor giving barrier energy that is less than or equal to zero, in other words an Ohmic³ contact is formed. This means that electrons can flow freely from the semiconductor into the electrode and vice versa, i.e. the metal-semiconductor junction will conduct under forward or reverse bias.

Aluminium is a commonly used metal for Ohmic contacts as it has a relatively low work function of 4.08 eV when compared to silicon's electron affinity, 4.05 eV. Similarly, for a p-type electrode, the metallic work function must exceed electron affinity and band gap of the semiconductor. Thus, to increase the Fermi energy and produce a good Ohmic contact with aluminium, a high dopant concentration exceeding 10^{19} cm^{-3} is used at each side of the sensor [55]. Since this doping concentration is orders of magnitude greater than the bulk, it is denoted with a +, e.g. n^+ , p^+ .

Pixel isolation strategy

Pixels in a silicon sensor consist of highly doped implants in the bulk which, segmented in x and y. By design, the pixels need to be isolated from each other, electrically, to provide spatial resolution. This section explores how additional electrical isolation is provided for the n^+ pixel implants used in the Pixel detector sensors in the ATLAS experiment.

When charged particles pass through the sensor the SiO_2 layer on the detector surface is ionised and builds up a positive space-charge. The positive space-charge in the oxide layer causes a localised electric field between the pixel implants which causes a layer of electrons to accumulate at the oxide-bulk interface. If the implants conduct electrons, in other words if these are n^+

³An Ohmic contact is a conductive contact with a linear voltage-current relationship.

implants, then the electron accumulation layer can short pixels together. Therefore electrical isolation is needed.

This is done either using p-stops, which are isolated p-doped islands in between n^+ implants, see Fig. 3.10a, or p-spray in which the p-dopant is simply sprayed on top of the implants and coats the bulk in a diffuse layer. The doping concentration of the n-implants is high enough that an additional masking layer is not needed in the case of the p-spray, see Fig. 3.10b. P-stop

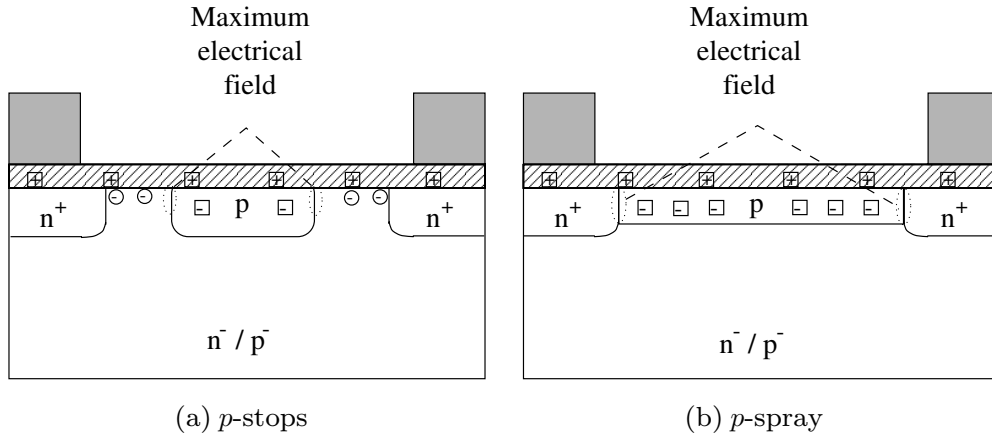


Figure 3.10: p-type isolation strategies for n^+ pixel implants. The squares show fixed space-charge regions, the circles are free charge carriers. Not shown are the additional metal and nitride layers [54].

isolation was used to isolate pixels in early sensors, as the size and concentration of the dopant can be easily controlled with common lithographic techniques. However, the high potential of the p-stop/bulk boundary leads to lower breakdown voltage as the sensors are subjected to a higher particle fluence [64]. This means that the p-stop isolation strategy is less commonly used in silicon sensors that will be exposed to a high particle fluence over their lifetime.

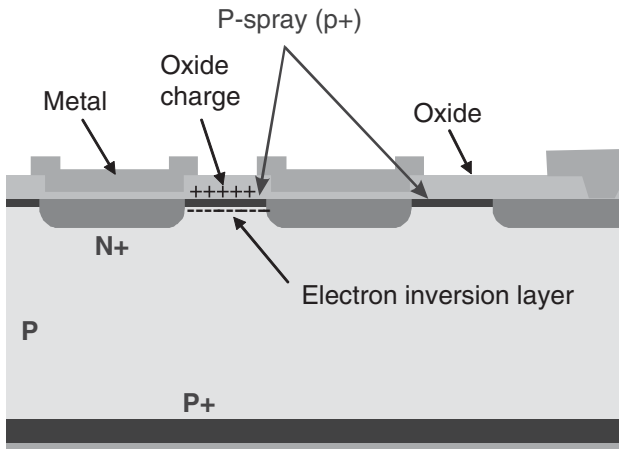
When using p-spray isolation, however, care must be taken to pick the correct dopant concentration. If the p-spray dopant concentration is too low, the electric field from the oxide layer will still penetrate to the bulk and short the pixels together. If the dopant concentration is too high, the junction formed at the p-spray and implant boundary breaks down at lower voltages, like the p-stop, see Fig. 3.11b.

A moderated p-spray technique that has a thicker p-spray layer between pixels is used to combat the disadvantage of the p-spray. The thick, inter-pixel layer isolates the bulk from the oxide whilst the concentration adjacent to the n^+ implants is made just sufficient to isolate the bulk whilst not causing early breakdown in the device.

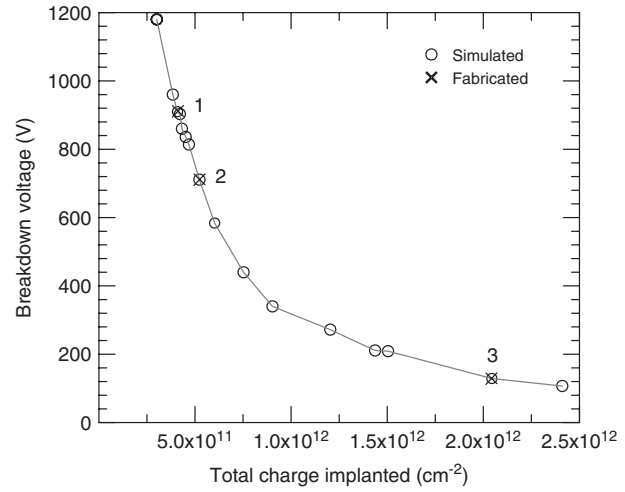
An additional advantage of using p-spray isolation instead of p-stop is that the application of a higher p-spray dose between the pixels is done by etching a deposited nitride layer later in the process and reapplying the p-spray. This does not count as an additional processing step, unlike the p-stop oxide etching, as nitride application is a commonly used process in semiconductor fabrication, for example in the passivation layer. The final metallisation and isolation strategy for an individual n^+ pixel is shown in Fig. 3.12.

Guard ring design and the bias grid

During operation, the pixel sensor is reverse-biased to deplete the bulk. The front-end chip which reads out the sensor, digitizes, and stores information about the pixel hit, is also biased. The front-end chip bias voltage is orders of magnitude smaller than the sensor, thus it makes sense to keep the side of the sensor that is in electrical contact with the front-end at a similar



(a) A cartoon showing the effect of the charged oxide layer on the bulk, shorting the n^+ implants with an electron accumulation layer.



(b) The effect of p-spray dopant concentration on the breakdown voltage of an n^+ strip sensor.

Figure 3.11: The effects of using a p-spray isolation layer that is too thin (a) and too thick (b) on the electrical properties of a silicon n^+ implant sensor [65].

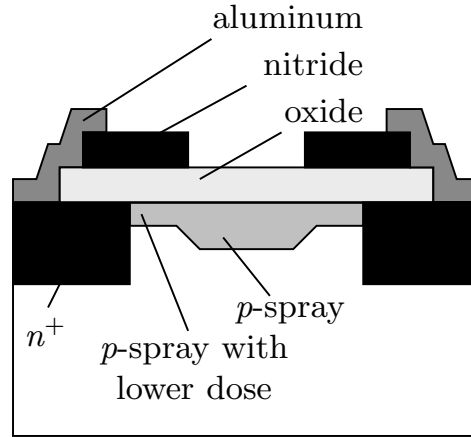


Figure 3.12: The inter-pixel material layers. n^+ implants are read out by aluminium electrodes, a p-spray isolates the pixels from each other and shields the bulk from any positive-charge build-up in the oxide layer, and the nitride is used as a mask for the edge p-spray region [54].

voltage to avoid sparking.

During fabrication the silicon sensors are diced from wafers into single units. This cutting process causes mechanical damage to the edge of the lattice and effectively causes it to become conductive. If the depletion region reaches this edge, the lattice damage acts like a generation centre and increases the leakage current. If a metal contact, at a different potential than the cutting edge is nearby, an electron accumulation layer can form and create a high field leading to early breakdown, as was the case with the p-stop implants. To prevent this, the potential difference between backside and frontside is stepped down in stages, using structures called guard-rings on the implant side where the junction is located.

Guard rings are based on the principle of punch-through. A floating implant, like that shown in the right side of Fig. 3.13a, is at V_{bi} - which is very close to the potential of the bulk. As the backside bias is increased there comes a point where the depletion region around the grounded electrode makes contact with the floating implant: this is known as the punch-through voltage, V_{pt} and the floating electrode will remain at this voltage even as the backside bias is increased,

see Fig. 3.13b. In Fig. 3.13c, V_{pt} is sufficient that any holes in the floating implant flow to the grounded electrode. More floating implants can be daisy chained together to the edge

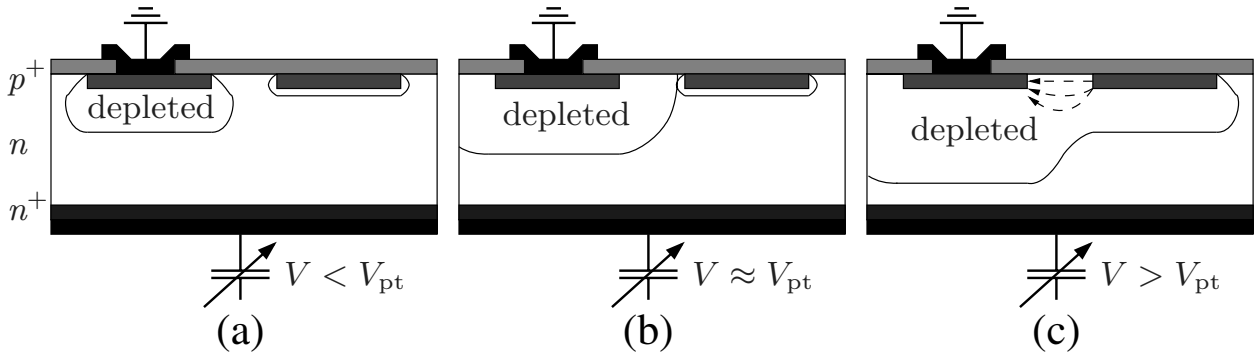


Figure 3.13: The effect of doping concentration and electric field on charge carrier mobility [54].

of the sensor, with a potential drop between each subsequent ring. The spacing of the rings determines their voltage drop and is usually non-uniform over the surface of the sensor. For example, the innermost ATLAS Pixel layer has planar sensors with 13 guard rings. Ramping down from high voltage to ground the ring spacing is: 3 rings $\times$ 25 μm , 50 μm , 5 rings $\times$ 25 μm , 2 rings $\times$ 50 μm , and 100 μm .

A bias grid is present in modern pixel sensors to test the functionality of pixels before the expensive process of bump-bonding them to the front-end readout chip. The bias grid is essentially a net connecting every pixel in the sensor. The pixel implants are not connected to the bias grid directly, instead part of the pixel implant is removed and an isolated implant is introduced, called a bias dot, see Fig. 3.14. Before bump-bonding the bias dot can be coupled to the pixel implant via punch-through, like the guard rings. After bump-bonding the bias grid is either connected to ground via a capacitor or a resistor depending on its design such that noise is sent to ground. The design shown in Fig. 3.14 minimises the charge lost by this space in the pixel implant as any charge that falls into this area will drift towards the surrounding implant.

Pixel and bulk dopant pairings

In the previous parts of this section the basic components of a silicon Pixel sensor have been described including the bulk, ohmic implants, isolation to stop those implants shorting, guard rings to ramp down the bias voltage, and bias grid to test the sensors before bonding them to the front-end readout chip. Whilst these latter features are necessary to test and use the sensor, the majority of its properties - particularly those that are of interest to the topic of radiation damage on which this chapter focuses - stem from the choice of bulk and pixel implant dopant type. In the discussion that follows an implant, at the readout electrode, of type A in a bulk of type B will be referred to as an A-in-B design.

There are four possible combinations of implant and bulk: n^+ -in- n , n^+ -in- p , p^+ -in- p , and p^+ -in- n . Ideally, the depletion region will always grow *from* the pixel implant *towards* the backside of the detector so that even when under-depleted some charge can still be collected. This constraint would also enable single-sided processing. If the pn-junction grows from the back of the sensor, backside processing is needed to produce guard rings. However, if the guard rings are on the front-side they can be produced in the same steps as the pixel implants. Single-sided

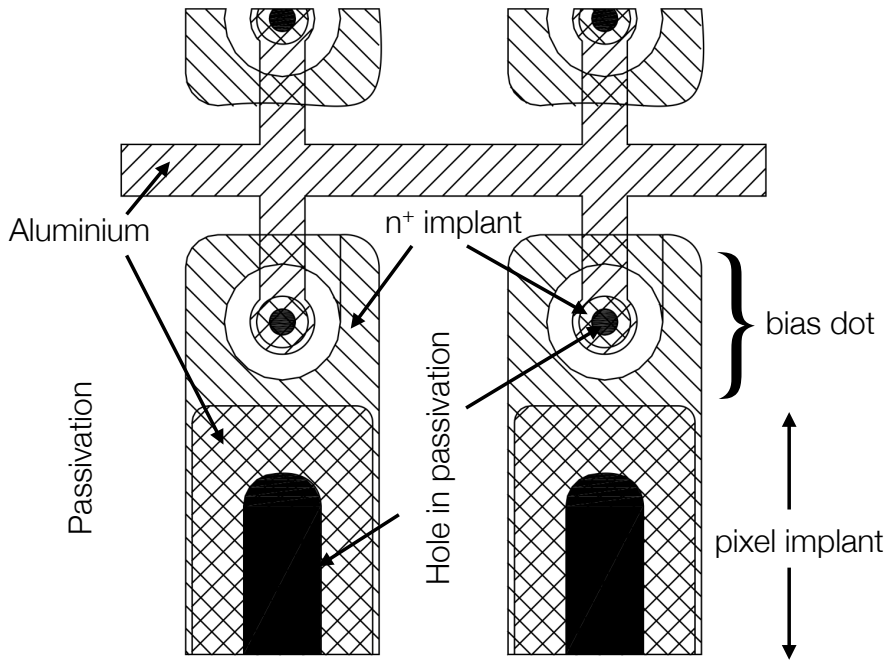


Figure 3.14: The bias grid connecting four pixels. Solid black sections are holes in the passivation, forward hatching is aluminium, and backwards hatching is the pixel implant. Adapted from [54].

processing is not only more cost-effective but allows for thinner sensors. This is shown for p^+ -in-n and n^+ -in-n sensors in Fig. 3.15.

Two bulk and implant pairings with the pn-junction on the front-side are the n^+ -in-p and p^+ -in-n designs. The use of p^+ -in-n sensors in particle physics goes back to the first planar silicon sensors used in the NA11 and NA32 fixed target experiments at the SPS in the early 1980's, and extends through to the current strip sensors in the ATLAS SCT [66]. Additionally, because of the inherently isolated p^+ implants, not only was single-sided processing possible but additionally no p-stop isolation was needed. However, modern Pixel sensors do not use the p^+ -in-n design. The reasons for this stem from the effects of radiation damage on the sensor bulk that causes the net doping concentration in the n-type bulk to change to a p-type bulk, which will be described more completely in section 3.3. Bulk-type inversion causes the depletion region, in a p^+ -in-n sensor, to grow from the back of the sensor. This means the detector has to be fully depleted to avoid signal loss, which requires an increasingly high voltage as the radiation damage continues.

Using an n^+ -in-n sensor instead of p^+ -in-n, in environments which experience a large amount of radiation damage, has the advantage that after bulk-type inversion the pn-junction will grow from the pixel electrode towards the backplane so the detector can be operated below full depletion. However, before type inversion, the pn-junction is located on the backside of the sensor, so the guard rings are also placed on the backside, see Fig. 3.15b. Although placing the guard rings on the backside introduces an additional processing step, having the pn-junction on the backside of the detector reduces the possibility that the high voltage sensor bias could arc to the grounded front-end which is only 10's of microns from the sensor and easily exceeds the critical value in air of 1 kVmm^{-1} [67]. Recent studies however show that introducing a passivation layer of Benzocyclobutane (BCB), which has a critical breakdown value of 300 kVmm^{-1} , allowed early prototypes of n^+ -in-p sensors to be biased to 1 kV for several hours [68]. With this in mind n^+ -in-p sensors combine the benefits of the depletion

⁴P-type silicon does not undergo space-charge sign inversion.

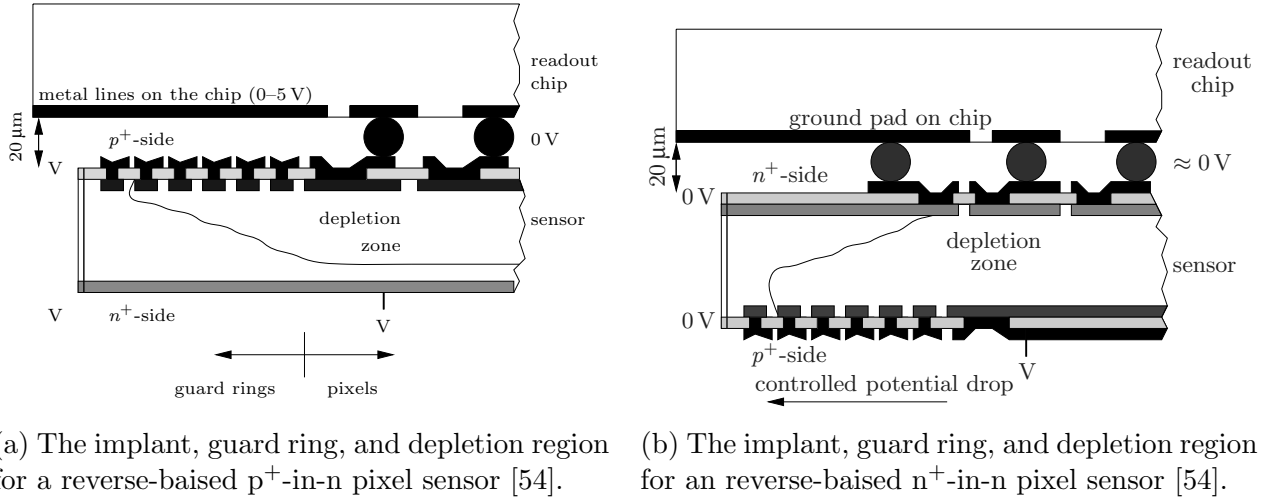
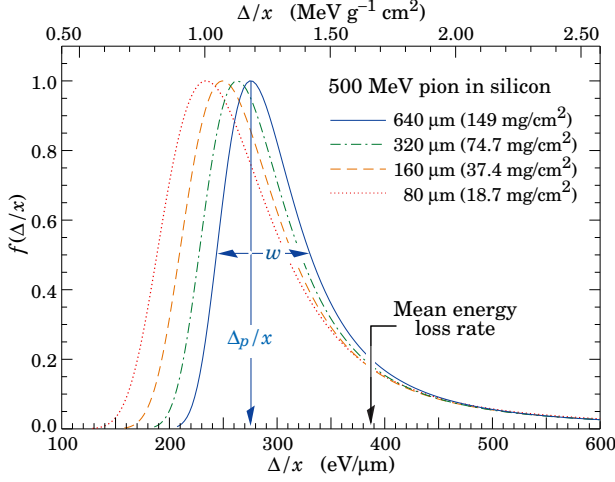


Figure 3.15: Bulk and implant designs

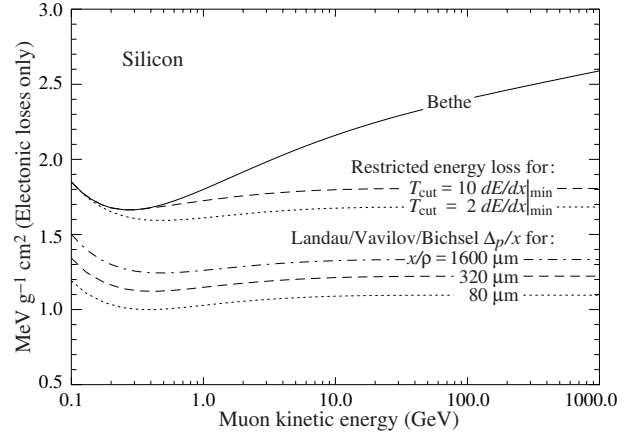
region growing from the frontside of the detector⁴ even after high radiation doses with the single-sided processing seen in p^+ -in- n detectors, see Fig. 3.15a. n^+ -in- p sensors will be used in future silicon detectors, including ITk.

3.2 Energy deposition in silicon detectors

Charged particles traversing a depleted silicon sensor will impart energy to the bulk as they pass through it. This section will describe the mechanism by which this energy is imparted to the bulk and provides a brief discussion on phenomenological models that describe this process. High energy particles traveling through matter lose energy through single-particle interactions. In relatively thin detectors, fewer collisions will take place and there is a high relative variance in energy loss [69]. The Bethe formula, which gives the mean rate of energy loss, is ill suited to describe thin detectors. This is because the energy loss probability distribution function, or straggling function $f(\Delta; \beta\gamma, x)$, is highly skewed with a long tail from single-collision large energy transfers. As such the mean of the distribution is in the tail, and thus the modal value would be a more informed choice. In thin silicon detectors, the most probable energy loss over a thickness, x , is far smaller than the mean energy loss, see Fig. 3.16a. Only when x is around 15 cm thick does the Bethe formula become relevant. This thickness is an order of magnitude greater than the maximum path length a particle could traverse in a single silicon detector used in ATLAS, as is illustrated in Fig. 3.16b. A straggling function that does well to describe the energy loss p.d.f. is the Landau-Vavilov function with Bichsel corrections. Landau's formula derives from the Rutherford cross-section [70] and focuses on the problem of energy loss in finite material thickness with two conditions. The first being that the most probable energy loss would be large relative to the lowest orbital bound electrons, or s -shell electrons, and the second that the energy loss would be very small when compared to the maximum energy loss in a single collision, which was later removed by Valivov [72]. This first condition, however, is not applicable to most MIPs interacting with silicon detectors where, for example, in 250 μm of silicon the most probable energy loss is on the order of 10 keV, whereas the $1s$ orbital electrons have binding energy on the order of 1.8 keV [73]. This condition was, at least partially lifted, by Blunck and Leisegang who convolved the Landau-Vavilov distribution with a Gaussian distribution of width equal to the Rutherford cross-section [70]. Whilst the 'LanGau' is a good approximation to the underlying straggling function, differing by $\sim 5\%$ at 200 μm in silicon, it

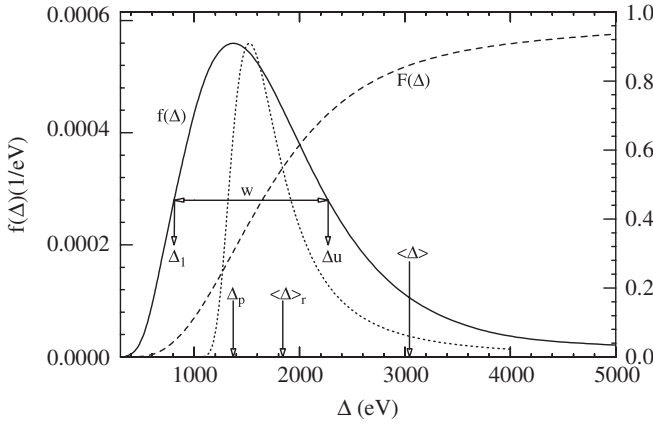


(a) Straggling functions for the energy loss distribution of a 500 MeV pion in various thicknesses of silicon, with each distributions normalised at its modal value. The full width half maximum, w , decreases with increasing thickness.

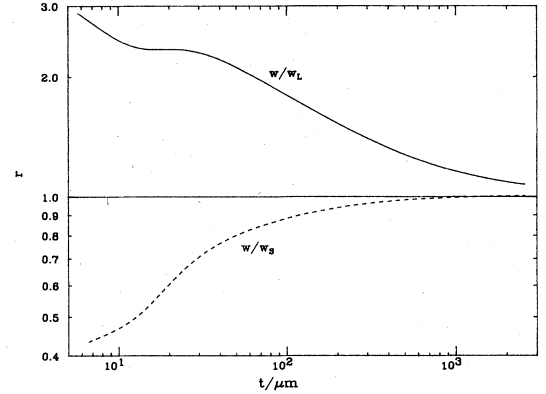


(b) Energy losses in silicon, for muons of various energies. The solid line is the mean energy loss from the Bethe formula. The first set of dashed and dotted lines show the mean loss if $T \leq T_{\text{cut}}$. The final three lines show the most probable energy loss for three different thicknesses.

Figure 3.16: Comparisons of the mean and most probable energy losses for minimum ionising particles in silicon [69].



(a) A comparison of the straggling function (solid line) and the Landau function (dotted line) for a particle of $\beta\gamma = 3.6$ in $10 \mu\text{m}$ of silicon. The dashed line shows the cumulative straggling function. The labels indicate the straggling function m.p.v, Δ_p , the mean energy loss from Bethe function, $\langle\Delta\rangle$, and the mean energy loss calculated from the subset of energies in the range of the plot, $\langle\Delta\rangle_r$, [70].



(b) The ratio, r , of the full width half maximum, w of the straggling functions in a silicon detector compared to w_L in the Landau-Vavilov distribution, for increasing thickness, t (solid line). The dashed line compares w and w_S , the FWHM of the Landau distribution with the Shulek correction. This correction was dropped following its divergence from experimental data [71].

Figure 3.17: Comparisons of the Landau function's modal value and FWHM with detector straggling functions.

breaks down for thicknesses below this.

Bichsel added corrections to the Landau-Vavilov distribution by accounting for the density effect. As a particle's energy increases, its electric field changes shape, increasing the contribution to the energy of distant collisions [69]. Additional corrections, that added resonances related to the electron orbitals in atomic structure, were added which improved upon similar efforts from Blunck, Leisegangs, and Shulek [74].

Bichsel, in his 2006 review of straggling functions for particle tracking and ID [70], makes the case that producing analytical straggling functions from historical first principles is computationally expensive and instead they should be calculated ahead of time with a Monte Carlo simulation. This will be explored further in section 3.5.1.

The straggling functions provide a probability density function (p.d.f.) to sample and extract a collision energy for a given interaction. The Bichsel model also provides a deposition model of how to distribute the charge in simulation. In Section 3.5.1, a charge deposition model based on Bichsel's recommendations was added to the ATLAS simulation framework. This ensures the shape of Pixel clusters is correct and allows these effects to be decoupled from the effect of radiation damage on cluster size.

3.3 Radiation damage in silicon

The ATLAS Pixel and SCT detectors operate on the principle that high energy particles lose energy and ionise the detector volume. However the very particles that these sensors detect can damage the sensing medium. Over time, these damages accumulate and change the microscopic structure of the silicon detector, ultimately changing its macroscopic properties and affecting its performance.

This section will introduce the mechanism of radiation damage, and then explore its effects. Although this section is only concerned with radiation damage in silicon sensors, the front-end readout chip must also be carefully designed to cope with its effects. However, the modifications to make the front-end radiation 'hard' are outside the scope of this document. Discussions on the topic can be found, most recently for the HL-LHC front-end test-chip FE65-p2 [34] and for the current Pixel detector [75].

Understanding the cause of the changes in detector performance is essential to engineering solutions. It also provides a way to generate realistic simulation data for detectors in high-radiation environments. Simulation can then be compared to data collected in the experiment, which is an essential part of data analysis.

3.3.1 Mechanisms of radiation damage

Radiation damage in silicon sensors can broadly be divided up into surface and bulk effects. This is because the thin metal, oxide, nitride layers, see section 3.1.3, at the surface of a silicon sensor exhibit very different behavior following irradiation than the thick layer of low-dopant concentration silicon that forms the bulk of the detector.

Surface effects stem from the Total Ionising Dose, TID, which is the cumulative effect of ionisation in the surface dielectrics. Although the mechanisms that cause bulk damage can occur in the surface layers, the structural damage to the material affects its properties less than ionising radiation losses. Despite the relatively large ionisation energy in SiO_2 of 17 eV [76], high energy particles produced in collider experiments can still ionise the material as they pass through the detector. Many of the charge-carrier pairs produced by ionisation quickly recombine, however

some will not and so drift apart in the electric field. The electrons in charge-carrier pairs created in the oxide layer will quickly be collected at the n^+ side of the detector due to their high mobility, $\approx 20 \text{ cm}^2\text{V}^{-1}\text{s}^{-1}$ [54]. Holes have a much lower mobility than electrons in SiO_2 , $\approx 2 \times 10^{-5} \text{ cm}^2\text{V}^{-1}\text{s}^{-1}$, due to interstitial traps⁵ and are unable to escape, leading to a positive charge accumulating in the oxide layer.

This effect is responsible for the electron accumulation layer that can short n^+ pixel implants in the silicon detector and necessitates the use of a p-spray or p-stop to mitigate its effect, see section 3.1.3.

Non-ionising radiation damage

Ionising radiation loss in the bulk does not change a detector's electrical properties thus there is no concept of TID for the majority of the sensor, instead, Non-Ionising Energy Loss (NIEL) is the dominant cause of radiation damage.

NIEL damage is caused by a particle displacing an atom from its lattice position in the bulk. Often the displaced atom has enough energy to displace others in a 'knock-on' effect, in such cases the subject of the initial NIEL damage is the Primary Knock-on Atom (PKA). The clusters of damage that can form from a PKA consist of several types of lattice defects. A stable defect, consisting of a displaced, interstitial atom (I) and a vacancy (V) in the lattice, is known as a Frenkel pair (FP).

The average number of Frenkel pairs is given by:

$$\langle N_{FP} \rangle = \begin{cases} 1, & \text{for } E_d < E < \frac{2E_d}{\zeta} \\ \zeta \frac{E}{2E_d}, & \text{for } E > \frac{2E_d}{\zeta} \end{cases} \quad (3.15)$$

where E_d is the minimum energy required to displace an atom from its lattice site, E is the energy imparted to the lattice atom, and ζ is the displacement efficiency. This can be used to estimate the average number of Frenkel pairs created from a particle of known E_d and ζ , for example, in neutrons: $E_d = 25 \text{ eV}$ and $\zeta = 0.8$ [77]. Neutrons only transfer energy to silicon detectors in non-ionising interactions and so are used as a standard to measure how many Frenkel pairs and hence how much NIEL damage is done to the bulk. A hardness factor, κ , is used to convert between the a recorded particle fluence and the fluence of 1 MeV neutron per square cm , also referred to as the 'neutron equivalent fluence' or $n_{\text{eq}}\text{cm}^{-2}$.

3.3.2 Effects of NIEL damage on silicon detectors

The specific effects covered in this section are those relevant to the radiation damage modeling undertaken for this work, and are not a complete list. The production of charge carrier traps will first be considered, followed by the multiple effects of space-charge sign inversion including changing the electric field shape, the mobility of the charge carriers, and how annealing can be used to reverse some of the damage.

Charge trapping

If a Frenkel pair does not immediately recombine it can migrate in the lattice and react with other defects to become a trap. A trap causes a charge carrier to become stuck in its electrostatic potential well. In a shallow trap the charge carrier can usually escape simply due to

⁵Traps and structural defects will be discussed further in the next section.

thermal motion, in other words the trap's energy lies close to the respective charge carrier's conduction band. If a trap is deep then the operating conditions of the detector usually have to change for the charge carrier to be freed, see the discussion on annealing that follows. Conversely a deep trap is closer to the respective charge carrier's valence band.

Acceptors denote traps that will become negatively charged when filled and donors are traps

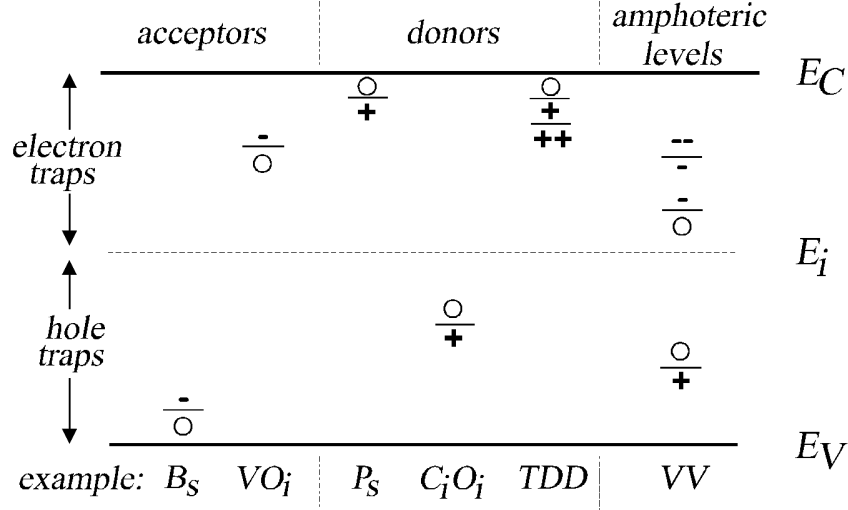


Figure 3.18: Energy levels of a variety of hole and electron traps. The energy levels in the trap are denoted with a negative (-), neutral (o), or positive (+) charge when they are filled with an electron or not [78].

that become positively charged when filled. Amphoteric defects introduce both donor and acceptor levels into the band gap. A common trap in silicon is a di-vacancy, VV . A di-vacancy, amphoteric defect introduces several acceptor states very close to the intrinsic Fermi energy in the band gap. In n-type silicon these are easily filled and contribute a negative space-charge to the bulk. Another common defect is a vacancy capturing an interstitial oxygen atom and forming one or two bonds, generally this trap is referred to as VO_i , which is an acceptor. These and other common traps are shown in Fig. 3.18.

If enough traps are present then free charge carriers traversing the bulk do not make it all the way to the electrode and instead become stuck in the trap.

Space-charge sign inversion

A common reaction in n-bulk silicon is the VP defect, which is formed by the n-dopant phosphorus (P) atom reacting with a lattice vacancy (V). This particular trap differs to the other traps shown and discussed in the previous section in that it does not trap free charge carriers but rather the interstitial defect in the Frenkel pair.

Aside from removing the Frenkel pair's recombination centre, it also removes an electron donor from the bulk, changing the Fermi energy of the material [79]. Donor removal, along with acceptor addition from V_iO formation will result in changing the effective carrier concentration from n-type, with a positive space-charge, to p-type with a negative space-charge. This process is known as space-charge sign inversion, or type inversion, and affects a number of macroscopic detector properties.

A subfield of silicon detector physics attempts to offset space-charge sign inversion using defect engineering, where impurities, such as oxygen, are introduced to the bulk. A popular theory as to why this works is that V_2 defects are less likely to form than VO defects in a higher

oxygen concentration and since VO is a more shallow acceptor trap, an electron is less likely to be trapped for long and contribute to the negative space-charge [80] [81] [82]. In the past decade there has been a particular focus in exploring the use of crystal growth techniques that introduce impurities during growth, for example magnetic Czochralski (MCz) has a homogeneous distribution of oxygen, whereas Diffusion Oxygenated Float Zone (DOFZ) diffuses a layer of oxygen into the surface of the bulk after crystal growth. The DOFZ technique has been used successfully in the most recent iteration of ATLAS Pixel sensors for the innermost barrel layer, the Insertable B-Layer (IBL) [83] and will likely be used in ITk (D. Muenstermann, personal communication, March 2017), ITk prototypes are currently produced in the Float Zone method (FZ) [84].

The change in space-charge can also be viewed as a change in the effective doping concentra-

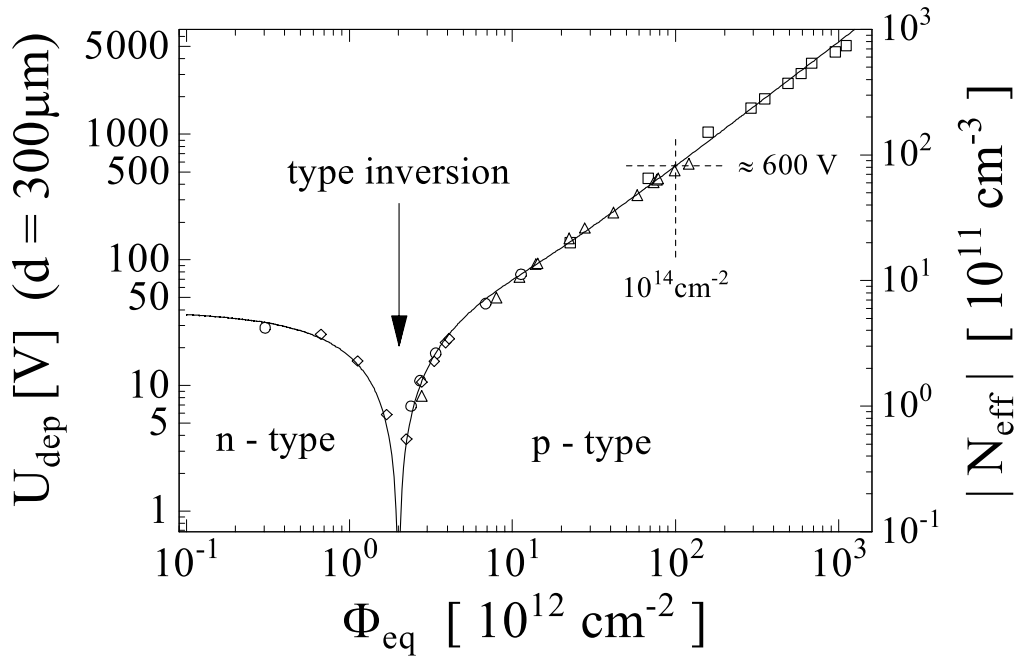


Figure 3.19: The change in effective carrier concentration (N_{eff}) and full depletion voltage (U_{dep}) with neutron equivalent fluence for a 300 μm n-type bulk silicon strip sensor [54].

tion N_{eff} , as seen in Fig. 3.19. In the figure, an n-type material is irradiated with particles of a high hardness factor, leading to a decrease in the effective doping concentration before type inversion, and an increase after it. In p-type materials the combination of donor removal and acceptor introduction means that no type inversion will occur. This fact allows for the n^+ -in-p sensor designs used in ITk, see section 3.1.3.

Changes to electric field shape

Unirradiated sensors have an electric field shape that increases linearly with an increasing bulk depth, reaching a peak at the pn-junction. This shape is due to the relatively uniform space-charge density present in the material. However, once a sensor experiences NIEL damage, traps near both collecting electrodes become filled due to the generation-recombination current near each electrode [85]. The filled traps generate a space-charge near the electrodes: electrons are trapped at the n^+ implant causing a region of negative space-charge and holes are trapped in the p^+ implant, causing a region of positive space-charge. When the detector is fully depleted the n^+ implant usually has a positive space-charge, so the introduction of the space-charge region from trapped electrons creates an additional pn-junction, see Fig. 3.20. The introduction of

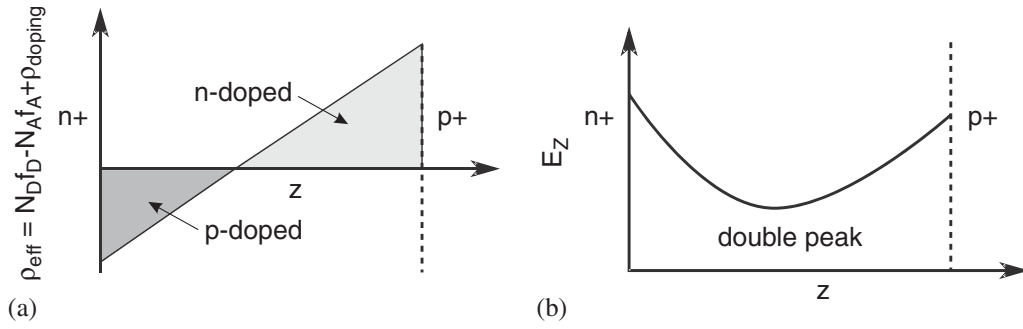
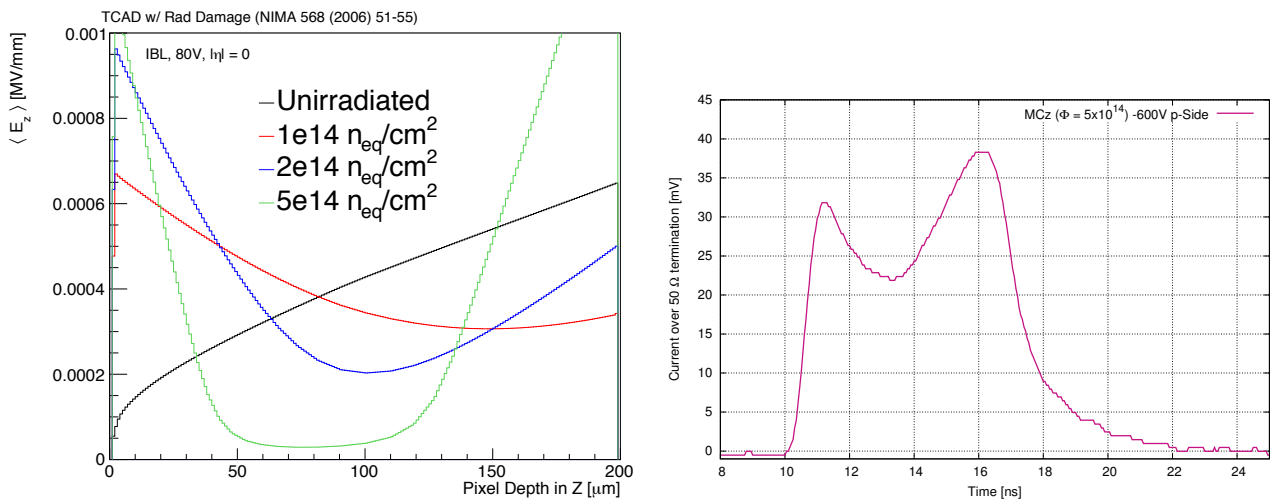


Figure 3.20: The effective charge density (a) and resultant field shape (b) in an irradiated module with trapping [85]



(a) Simulated electric field shape in a 200 μm planar IBL sensor at various fluences as a function of distance from the collecting electrode, where the contributions are averaged over the x-y plane, courtesy of B. Nachman and M. Bomben [86].

(b) The signal produced by electrons traversing an n^+ -in-n diode from backside to collecting electrode. The diode has been irradiated to 5×10^{14} $\text{n}_{\text{eq}}/\text{cm}^2$. The double junction is clearly visible.

Figure 3.21: The double junction phenomenon seen in irradiated diodes and sensors (b) is well modeled in simulation (a) by the Chiochia model.

these space-charge regions also creates a relatively neutral region in the sensor, which is usually skewed from centre. The double pn-junction and low space-charge region create a highly non-linear electric field. This causes the charge carriers to move quickly near the electrodes, and very slowly near the centre of the detector, see Fig. 3.21b. In simulation the non-uniform electric field shapes are modeled offline in a TCAD simulation, utilising the methods outlined in [85], which considers traps with two-defect levels in the band-gap. The number of traps and the number of filled traps changes with fluence, the results of this simulation for an IBL planar sensor irradiated at various fluences is shown in Fig. 3.21a. The simulation shows that even after 1×10^{14} $\text{n}_{\text{eq}}/\text{cm}^2$ the junction position has moved from the front to the back of the sensor. At larger fluences, as more traps become filled near the electrodes, the double junction becomes more prominent, leaving a low-field region in the centre of the sensor.

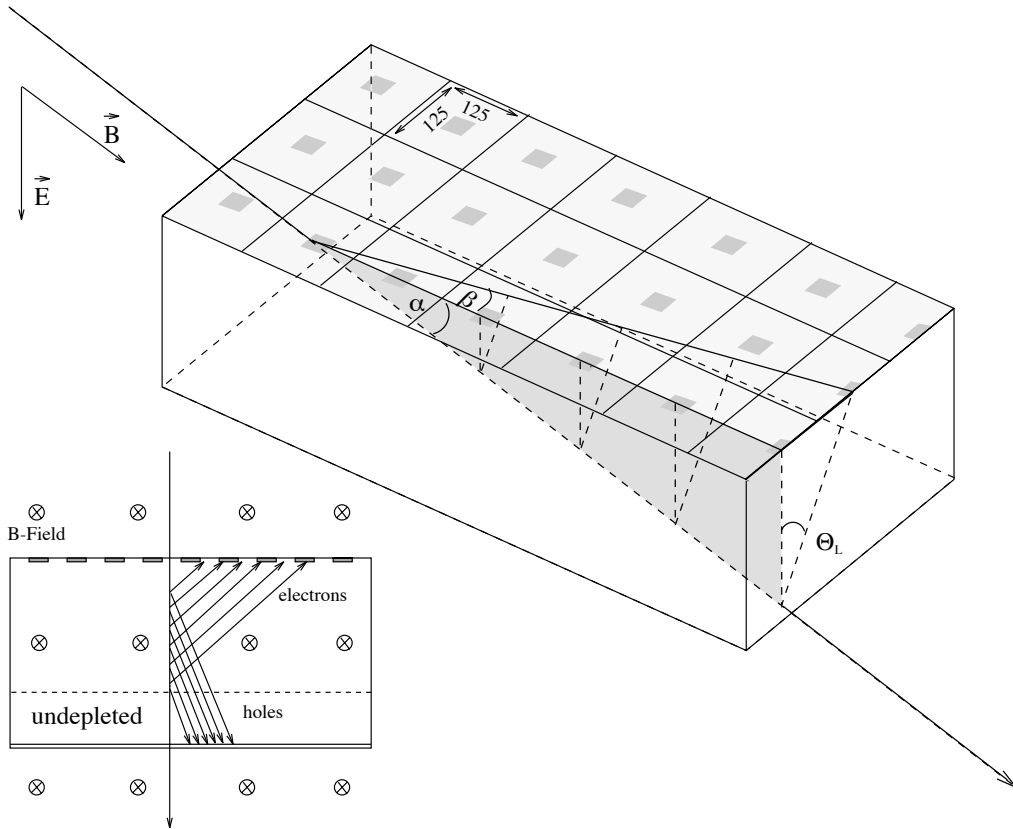


Figure 3.22: A 3D view of the various angles associated with a particle traversing a silicon detector in a perpendicular magnetic field. A slice of the volume, across the front face where the particle exits, is shown in the bottom of the figure and the corresponding drift due to the Lorentz force is shown for both charge carriers. α is the grazing angle of the particle, Θ_L is the Lorentz angle [87].

Drift due to the Lorentz force

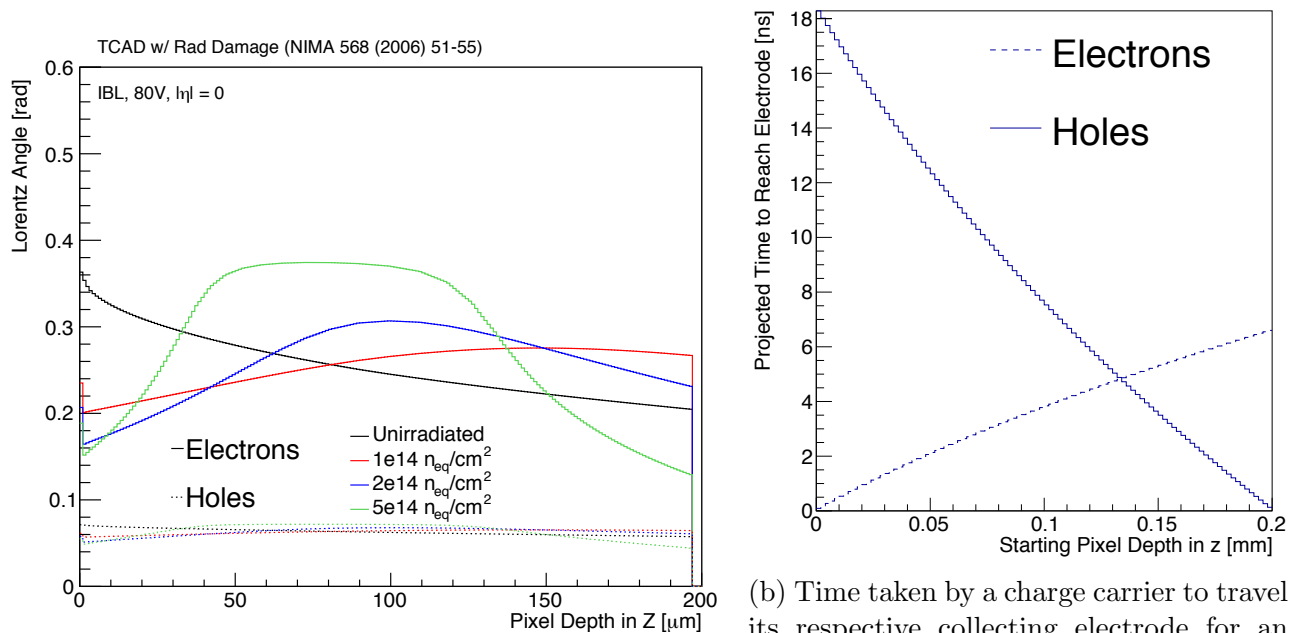
The non-uniform shape of the electric field, including features like the double junction, changes the mobility of the charge carrier, according to the relationship in equation 3.13. This affects the time taken to reach the electrode and recombine which also changes the likelihood of a charge carrier being trapped. The change in carrier mobility affects both the thermal diffusion of the charge carrier and the drift it experiences in a magnetic field, subtended by what is known as the Lorentz angle, see Fig. 3.22. The Lorentz angle is defined in equation 3.16:

$$\tan(\Theta_L) = \mu_H(\vec{E}) \cdot |\vec{B}| = \mu(\vec{E}) \cdot r \cdot |\vec{B}| \quad (3.16)$$

where Θ_L is the Lorentz angle, B is the applied magnetic field⁶, μ_H is the Hall mobility which is related to the drift mobility, μ , by the Hall factor, r . The magnetic field is assumed to be perpendicular to the electric field in the detector volume. The Hall factor, r , accounts for the scattering mechanisms present in silicon and is dependent on temperature [88]. The Lorentz angle for an unirradiated module is effectively uniform but this changes with fluence. The change in Lorentz angle after various fluences at different depths in the sensor is shown in Fig. 3.23a.

For simulation, plots such as these are integrated to produce cumulative distributions, which can be used as look-up tables at runtime. The maps shown here are preliminary versions that will be used to simulate radiation damage. Work is currently ongoing to establish the most accurate

⁶In the ATLAS Inner Detector the magnetic field is applied by a superconducting solenoid, see section 1.2.1.



(a) Simulated electric field shape in a $200 \mu\text{m}$ planar IBL sensor at various fluences as a function of distance from the collecting electrode, where the contributions are averaged over the x - y plane.

(b) Time taken by a charge carrier to travel its respective collecting electrode for an unirradiated sensor. In this plot the electrons are collected at $200 \mu\text{m}$. This plot was produced using a light-weight generator.

Figure 3.23: The electric field is used to calculate the charge carrier mobility and hence the Lorentz angle, simulated using a TCAD model in (a) for various fluences. This is used to determine total time taken to reach the electrode, simulated using a lightweight model for an unirradiated sensor in (b). Both plots were simulated with a $200 \mu\text{m}$ IBL planar sensor, biased at 80 V. Courtesy of B. Nachman and M. Bomben [86].

conditions with which to build the maps. For example, the rate of trap generation, and the capture cross-section of traps are currently being explored to best represent the characteristics of irradiated sensors for a given fluence [86]. The change in carrier mobility also affects how long it takes for a charge carrier to reach the collection electrodes. Since charge carriers can be produced anywhere in the bulk it is again useful to produce a look-up histogram, like the one shown in Fig. 3.23b. These maps are also produced, along with the Lorentz angle maps, for every desired fluence.

Annealing

Traps are composed of defects, some of which can be dissociated by applying thermal energy to the bulk. This can cause defects to travel through the bulk, reacting with other defects, or recombine. Applying thermal energy to the radiation damaged bulk to change its electrical properties is known as annealing.

Annealing is a time and temperature dependent process. Although the exact mechanisms underlying phenomena observed following annealing are still an active topic of investigation, the behaviour is well characterised. The Hamburg model [78] is a phenomenological model that parameterises the effects of annealing. The Hamburg model describes the effective carrier concentration of a material according to five time and temperature dependent processes:

- (a) The removal of electron donors in n-type silicon.
- (b) The constant addition of stable, non-annealable defects that act as acceptors.

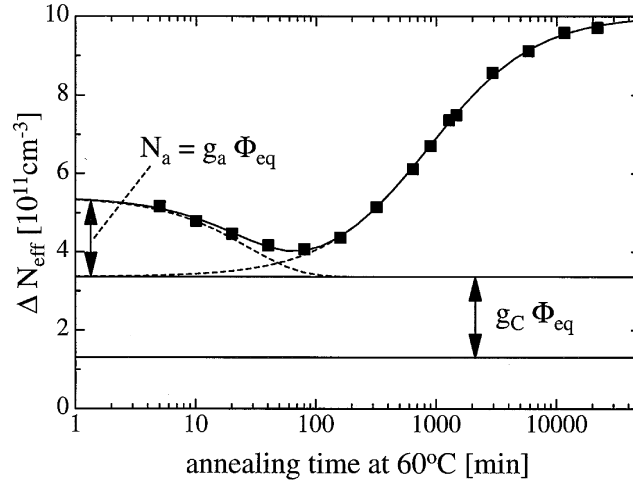


Figure 3.24: Beneficial and reverse annealing in silicon irradiated to $1.4 \times 10^{13} \text{ n}_{\text{eq}}\text{cm}^{-2}$. N_a is the contribution to the effective carrier concentration, N_{eff} , and is parameterised by g_a and the fluence Φ . The fit through the data points is employed using the Hamburg model [89].

- (c) Changing defects that act as acceptors to defects that are neutral.
- (d) The introduction of neutral defects, which have the potential to become acceptor defects, into the bulk.
- (e) Changing neutral defects into acceptors.

Processes (a) and (b) occur during irradiation. Keeping irradiated silicon at 300 K on the order of days results in process (c), this is known as beneficial annealing. Beneficial annealing is so called because it increases the effective carrier concentration in n-type silicon before type inversion, see Fig. 3.24, which appears to reverse some of the damage. However, if the annealing is allowed to continue on the order of weeks, processes (d) and (e) will take over, this is called reverse annealing. The Hamburg model is implemented by fitting to detector data. Beneficial and reverse annealing effectively change the detector properties such that it behaves as if it were irradiated with a different fluence.

To implement this in simulation, the Hamburg model is fitted to experimental data and used to establish the effective doping concentration. The effective doping concentration is then supplied to a TCAD model of the sensor and used to recalculate the electric field map. At this time the change in trapping rate due to annealing is not accounted for.

3.4 Digitization in Athena

The dual purpose of the Athena framework, which was introduced in Section 1.3, is reconstructing raw detector data into physics objects, such as tracks, and simulating detector hits from event generators that incorporate the present understanding of particle physics. Digitization is a simulation stage in Athena that takes HIT information generated by Geant4 and produces ‘digits’, or the corresponding digital response of a detector to the Geant4 stimuli. This is done in several stages: a detailed model of each subdetector is incorporated into the code which allows the subdetector community to fine-tune the analogue signal produced. This analogue signal is then combined with detector specific additions such as thermal noise and subsequently digitized. It is crucial that the digitization code accurately reflect the signal formation and

processing in the actual detector so that the digits produced by the simulation can be compared directly to the data read out of the detector. Ideally, an individual accessing the data container in which digits are stored would not be able to tell whether it was data or simulation. This section will provide an overview of the Athena PixelDigitization package which

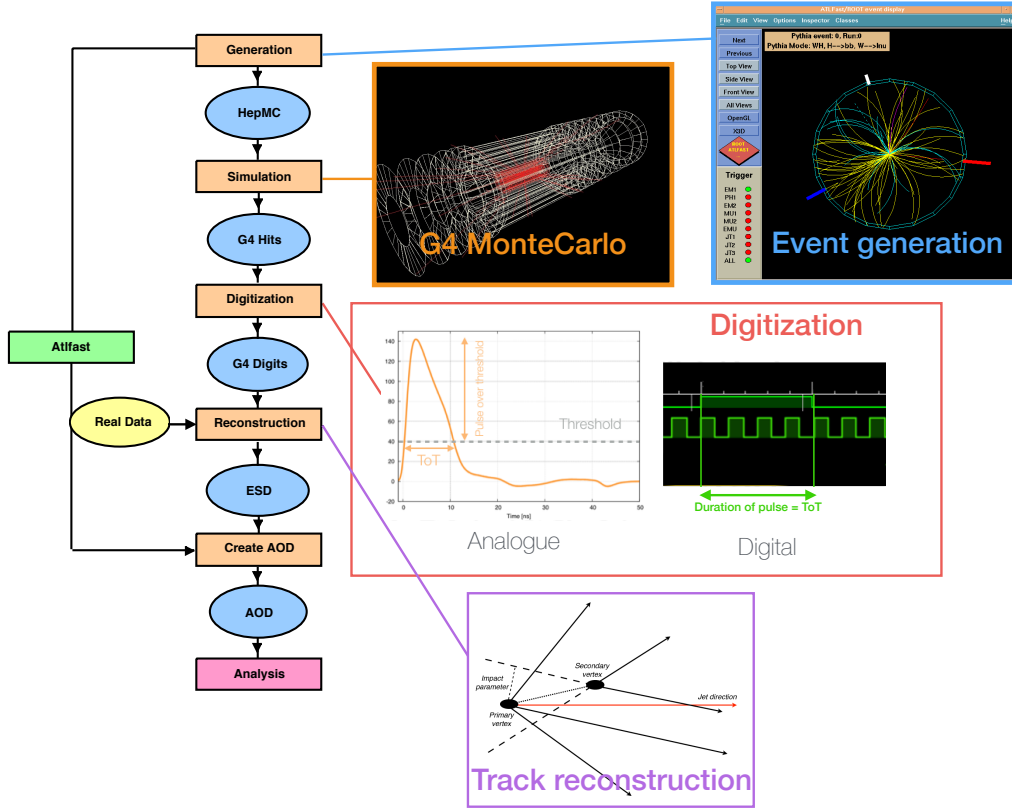


Figure 3.25: The Athena simulation workflow. Each stage of the simulation is shown with a box and the data object produced after each step is shown as an oval. The simulation data is processed until it is in the same format as the incoming detector digits. Thus the real data and the simulation data both enter the workflow at reconstruction.

digitizes all four layers of the ATLAS Pixel detector. Although the SCT detector digitizes energy deposits differently to the Pixel detector they share a common set of classes housed in the package SiDigitization (silicon Digitization) which includes definitions of containers for individual strip/pixel hits. Following this section will be a discussion about changes made to the PixelDigitization package. The package has been restructured for clarity and to reduce repeated code. The energy deposition model has been updated to better represent the spatial distribution of charge carriers in thin sensors using the Bichsel model. The signal formation model has been updated to include radiation damage modeling. This latter feature in particular is important to accurately model the silicon Pixel detector in ITk.

3.4.1 Athena simulation

The Athena framework is an implementation of the Gaudi architecture which was originally designed for LHCb [90]. The Gaudi architecture defines a high-level structure with well-defined interfaces and functionality upon which the software framework is built. Each structure in the architecture is easily decoupled and can be developed independent of the other structures, see Fig. 3.26. The Gaudi architecture produces *DataObjects* and *Algorithms* that operate on those objects. This approach is echoed in the Athena framework, shown in Fig. 3.25, where each

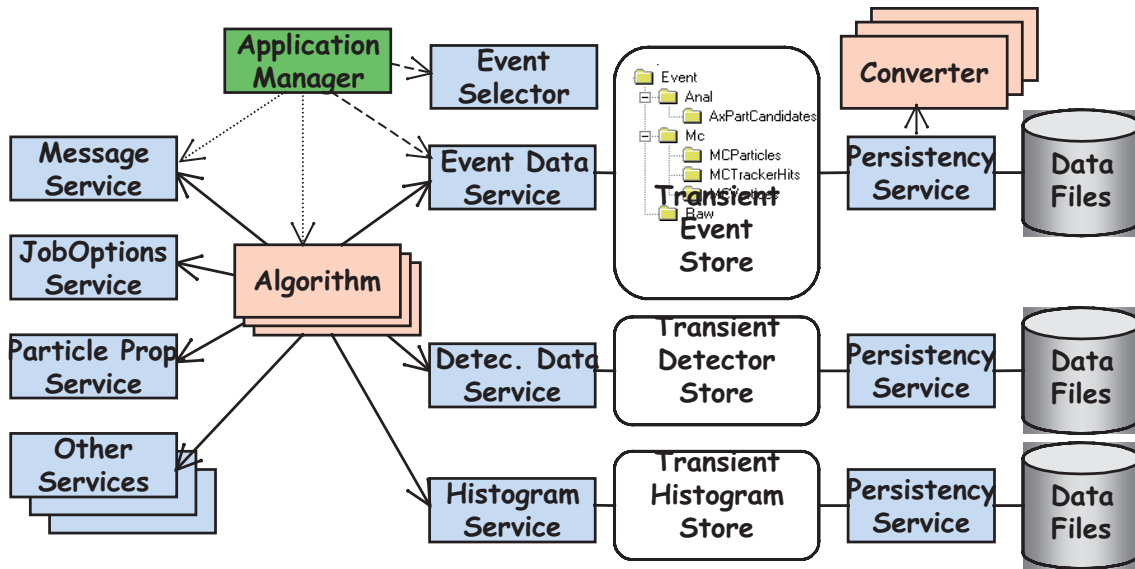


Figure 3.26: The Gaudi architecture. The core concepts of *Algorithms*, *Converters*, and *DataObjects* are shown in pink. The blue boxes are *Services* that supply common functionality to the Algorithms [21].

‘stage’ of simulation produces a new data object. Gaudi also defines a collection of *Services* which provide the Algorithms with a standard way to perform tasks common to the architecture such as filling and reading a data container, processing job options, and event selection.

Athena builds off of this architecture a complex framework that extends and specialises Gaudi. This includes the addition of *Tools* and the *ToolSvc*. Services are setup and initialised once per job by the framework. Moreover, the same instance of a Service is used by many Algorithms which both exist until the end of the job. A Tool acts as a sub-Algorithm, performing operations on specific subsets of a data object or only for part of the Algorithm’s functionality. In the past couple of years, development in Athena has moved away from basing functionality in Algorithms and more in Tools [91]. This is because of the different approaches both classes have to storing events, which become imperative to performance when processing collisions with a large amount of pile-up. Algorithms keep a cache of background events in memory for each event, which requires increasingly large memory usage as pile-up increases, whereas the PileUpTools only load those events that are needed, which increases the CPU consumption dealing with I/O, see Fig. 3.27. Whilst these differences are minimal for the pile-up conditions in LHC Phase-I LHC they will become prohibitive when simulating data for LHC Phase-II. Thus, most functionality in the Digitization package stems from PileUpTools rather than Algorithms.

The Athena development guide provides a loose definition of how a Tool differs from a Service [21] but current development is moving to redefine classes which have a configurable setup as a Tool and those that employ the same functionality to the argument list and need no setup parameters as Services. The Athena release discussed in this thesis does not yet employ this distinction but future modifications to this package may include such a restructure.

The PixelDigitization object derives from an *AthAlgorithm*. It contains several Tools that deal with various subsets of the Pixel detector including separate classes for the DBM⁷ and IBL, both of which derive from the *AthAlgTool* base class, and contains a call to the *PixelCalibrationSvc* which contains the charge-to-ToT conversion.

⁷The Diamond Beam Monitor (DBM) is used to monitor luminosity and beam background and sits in the forward region of the ATLAS detector.

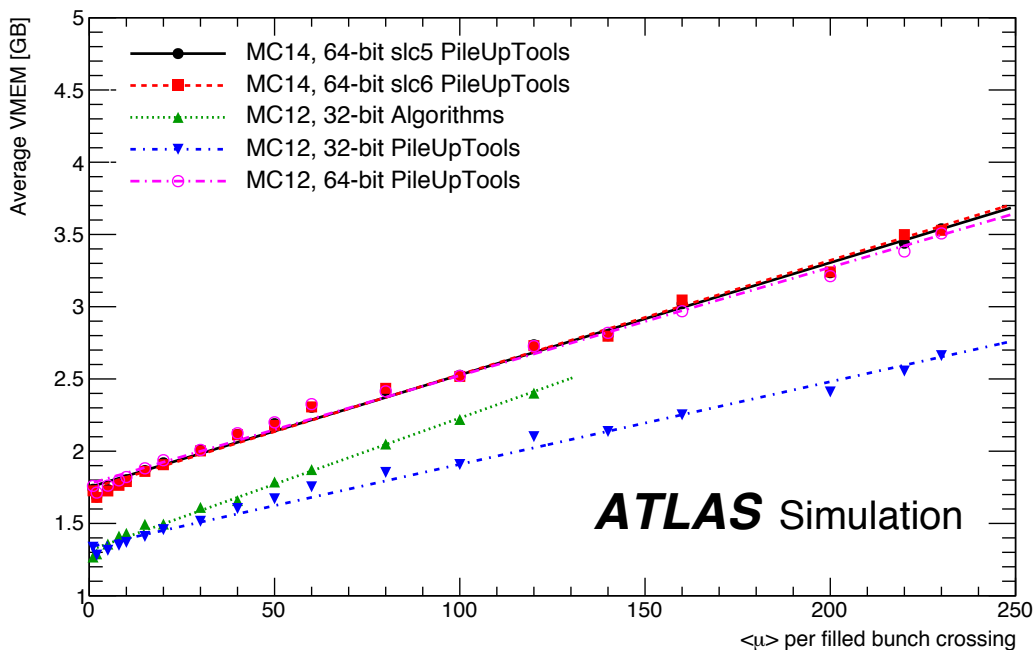


Figure 3.27: The average memory use for several digitization configurations as a function of the pile-up, μ . In comparing the 32-bit Algorithms (green triangles) and PileUpTools (blue inverted triangles) it is clear that the Algorithm approach diverges from the PileUpTools memory usage. The Algorithms data is cut short as it reaches the memory limit of the machines [91].

3.4.2 PixelDigitization structure

The Algorithm is the entry point into the digitization stage for Hits generated in Geant4 that pass through the Pixel detector. There are two possible Algorithms that can be used. First the PixelDigitization Algorithm which has no implemented functionality but simply loads a the PixelDigitizationTool and is mainly used when only subsets of the Digitization should be tested. Secondly there is the DigitizationAlg which collects an array of digitization Tools, including Pixel, SCT, LAr, etc. and is able to provide information to all of them as needed.

The PixelDigitizationTool inherits from the PileUpToolBase, a subclass of the pure abstract class IPileUpTool, and so can be loaded to the Tool array in the DigitizationAlg. The PixelDigitizationTool provides functions to deal separately with pile-up and physics events and is used by multiple Tools in Athena and is effectively the top level class in PixelDigitization. It sets up Services and additional Tools, and contains the functionality to iterate over each hit in an event. A full break-down of the PixelDigitizationTool functionality is not included here, but core functionality will be summarised⁸.

1. The entry point to the core code of Pixel digitization is **DigitizeAllHits()**. When this returns the ‘non-hits’ are digitized, and an RDO is created.
2. **DigitizeAllHits()** loops over the hits in an event with a function called **DigitizeElements()** which creates a collection of pixels and their corresponding collected charges for the event in a container called *chargedDiodes*, which is an instance of the siliconCharged-DiodesCollection. This is passed to a function called **CreateAndStoreRDO()** that applies a charge-to-ToT conversion, along with the operating conditions of the front-end such as

⁸The Athena PixelDigitization package referred to in this document is publicly accessible from: http://acode-browser2.usatlas.bnl.gov/lxr-rel20/source/atlas/InnerDetector/InDetDigitization/PixelDigitization/?%21v=release_20_20_7

the discriminator threshold, and then creates the dataObject, an RDO.

3. **DigitizeElements()** first checks that there are hits to process and uses the SiHit information in the hit pointer to look up the SiDetectorElement being used. In this case an element is a Pixel sensor, which stores information about the location of the module in the detector, such as whether it is a barrel module and what its η and ϕ coordinates are. The detector element is then passed, along with the chargedDiodes collection and the timing information, to the **SurfaceChargesTool::process()** function. The hit collection is looped over until all hits in it have been processed.
4. The SurfaceChargesTool uses the SiDetectorElement information to pick the correct SubChargesTool. Within the Pixel detector there are several different types of modules, including 3D, IBL planar, and the three outermost planar modules. Each have different physical properties, for example sensor thickness and pixel size, and so each are treated differently by the code. Once the correct SubChargesTool has been selected, the function **charge()** is called.
5. The **charge()** function in each SubChargesTool class contains the code that uses the information stored from Geant4 (the total energy deposition, entry point, exit point) to calculate the charge induced on the sensor's collecting electrode. How this works will be described in more detail later in this section.
6. With the chargedDiodes filled by the SubChargesTool::charge() function, the functions return to the PixelDigitizationTool where the collection is processed into an RDO, as described in step 2.

3.4.3 SubChargesTool

The modifications made to the PixelDigitization package discussed in this thesis are concerned with the step that takes the Geant4 output and outputs a siliconChargedDiodesCollection, in other words the part of digitization that forms the integrated charge in each pixel. This comprises two steps: energy deposition and signal formation. In the description of the class below, variable names may differ from those used in the original digitizer code, for clarity, these are explained in Appendix B.1. The names used here are those used in the modified code.

Energy deposition

The Geant4 step of the simulation provides three pieces of information which can all be accessed by the TimedHitPointer:

- Energy lost by a particle in the detector volume, accessed via the HIT pointer: `phit->energyLoss()`
- Particle entry point, which has entries in the *local* η , ϕ , and depth direction, accessed via `startPosition = phit->localStartPosition()`
- Particle exit point, also has entries in three dimensions as above, and accessed via `endPosition = phit->localEndPosition()`

The latter two values can be used to calculate the path length of the particle:

$$\text{pathLength} = \sqrt{(\Delta\phi)^2 + (\Delta\eta)^2 + (\Delta\text{depth})^2} \quad (3.17)$$

⁹The local geometry is the geometry defined with respect to a coordinate basis in the detector geometry.

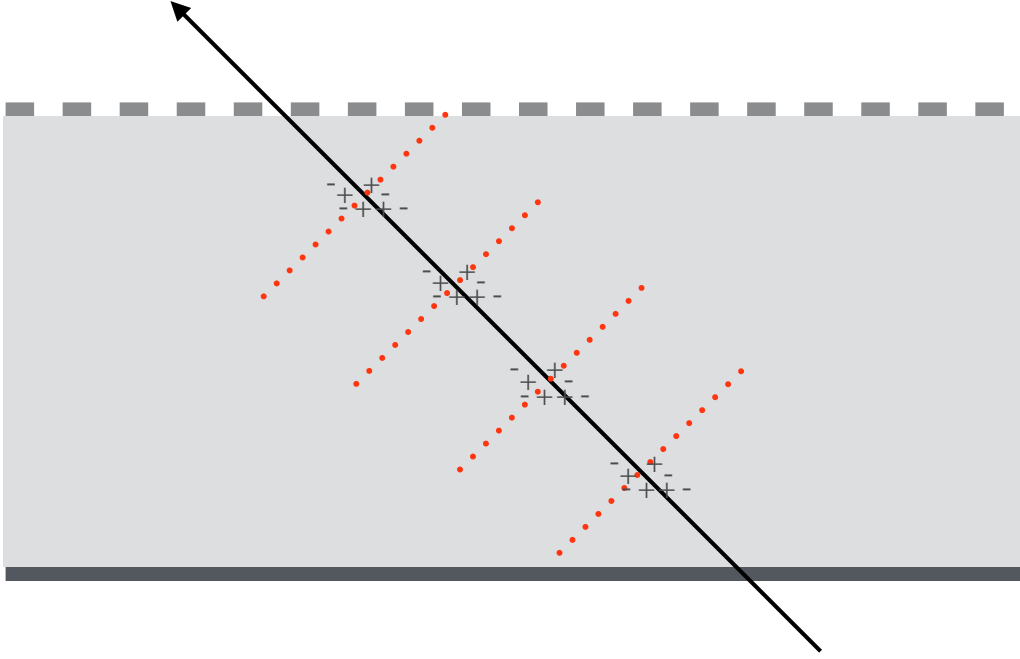


Figure 3.28: The uniform deposition of energy lost by the traversing particle in the sensor bulk. The red lines show the even divisions of the path length with equal amounts of charge-carrier pairs created per division.

where the coordinate variables are defined for the ‘local’⁹ sensor geometry and not with respect to the ATLAS detector. The energy loss is then deposited uniformly across the path length in steps, see Fig. 3.28. The stepsize is defined by the sensor thickness and an initial number of steps. The number of steps to process the energy deposition is then scaled by the path length:

$$\begin{aligned} \text{stepsize} &= \frac{\text{sensor thickness}}{\text{initial nSteps}} \\ \text{scaled nSteps} &= \text{int} \left(\frac{\text{path length}}{\text{stepsize}} \right) + 1 \end{aligned} \quad (3.18)$$

This energy deposition model is designed to be uniform, which means each step will contain the same amount of energy loss. The energy deposited in each step is further divided into ‘chunks’. To keep the processing time the same regardless of path length, the initial number chunks is scaled by path length. If the path length is large relative to the sensor thickness, the number of chunks per step will be smaller, in other words each chunk will represent a larger amount of energy loss than if the path length were smaller. This will become an important consideration in the radiation damage simulation. Equation 3.19 defines the scaled number of charge chunks in each step.

$$\text{scaled nChunks} = \text{initial nChunks} \cdot \left(\frac{\text{initial nSteps}}{\text{scaled nSteps}} \right) + 1 \quad (3.19)$$

The amount of energy deposited by the particle into the bulk is then equally split between all the charge chunks along the path:

$$\text{energyLoss per chunk} = \frac{\text{total energy loss}}{(\text{scaled nChunks}) \cdot (\text{scaled nSteps})} \quad (3.20)$$

Which is then converted to charge by dividing by the average energy needed to ionise silicon: 3.6 eV.

To summarise, this portion of the simulation places a scaled number of charge chunks, with

each chunk representing an integer number of fundamental charge carriers, at intervals along the path known as steps. Whilst this energy deposition is easy to simulate and understand it does not simulate the realistic distribution of energy along the path length. The most recent version of PixelDigitization includes a more accurate deposition model that is discussed in section 3.2.

Signal formation

After energy has been deposited uniformly in the bulk, the subCharges tool applies drift and diffusion to assess which electrode will collect the charge from the chunks and be added to the chargedDiodesCollection. The charge carriers drift towards their respective electrodes,

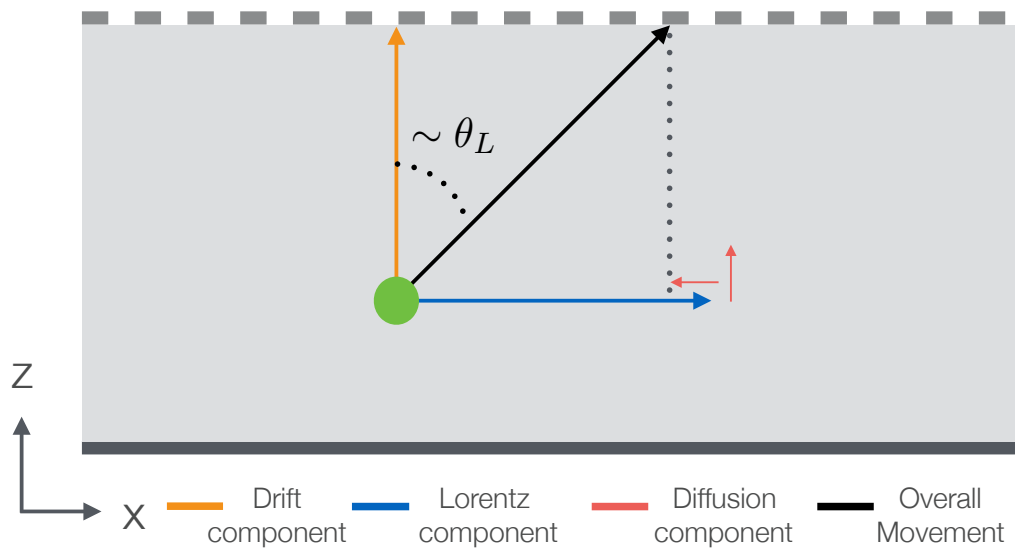


Figure 3.29: For each cluster of charge carriers, drift and diffusion are applied to identify the collecting electrode. The cartoon shows how these contributions are broken down in the code, the contributions are not to scale.

perpendicular to the sensor surface, due to the applied bias. In a magnetic field the Lorentz force will also cause the charge carriers to drift, in barrel modules in the ATLAS detector this motion is parallel to the pixel surface. The charge carriers will not follow the electric field lines exactly but exhibit some thermal motion around the path, known as diffusion. These contributions to the final position of the electron in the charge carrier pair are shown in Fig. 3.29.

Signal formation is simulated by two loops in the SubChargesTools. The first runs over each step along the path, the second gets the final position of each charge chunk in the step by calculating the lateral displacement due to the Lorentz angle from the charge chunk's distance from the electrode.

It also calculates the maximum displacement due to diffusion based on the distance from the electrode and makes a random number call to decide on what that diffusion actually is.

The signal formation logic is summarised in the pseudocode labelled Algorithm 1.

Once all the charge chunks have been ‘collected’, the final position of each is converted to a pixel index and stored for digitization.

Algorithm 1 Original Athena signal formation in SubChargesTool, where x is a 3D position vector, and θ_L is the Lorentz angle.

```

1: diffConst := 0.007 [mm]
2: for  $i < \text{scaled nSteps}$  do
3:    $x_i = x_0 + \Delta x \left(i + \frac{1}{2}\right)$ 
4:    $\text{distElectrode} = \frac{1}{2} \cdot \text{sensorThickness} - \left(\text{module.readoutSide}() \cdot x_i[\text{depth}]\right)$ 
5:   for  $j < \text{scaled nCharges}$  do
6:      $\text{rdif} = \text{diffConst} \cdot \sqrt{\text{pathLength}/300} \mu\text{m}$ 
7:      $x_f = x_i + \text{distElectrode} \cdot \tan(\theta_L) + \text{rdif} \cdot \text{rand}(\text{gaussian})$ 
8:   end for
9: end for

```

Discussion

The SubChargesTools are relatively short pieces of code which nevertheless simulates a lot of functionality. A few of the assumptions made, however, affect the accuracy of the simulation and will be discussed here.

The energy deposition part of the code assumes a uniform spatial deposition, whereas the spatial deposition is actually governed by a Poisson process, as described in section 3.2. The Geant4 simulation prior to digitization samples the straggling function for the detector volume to obtain the energy loss but does not provide a spatial distribution. This has been corrected by rerunning the energy distribution as Monte Carlo in the latest Athena release, which is described in section 3.5.1.

The signal formation portion of the code only drifts and diffuses the carrier collected by the readout-side electrode, which in the Pixel detector is electrons. The holes are essentially ignored but the implied behaviour is that they drift by the same amount as electrons and are collected on the backside. Although both holes and electrons are generated in equal amounts during ionisation, electrons make up the dominant portion of the signal, as will be explained in Section 3.6.2. Since holes have a smaller mobility than electrons, they drift less under the Lorentz force and so could end up underneath a different pixel than their partner electron. At maximum the two could differ by one pixel, so it is unclear if this has a significant effect on the cluster shape. However, when modeling radiation damage holes and electrons must be treated separately as they have different trapping rates.

The only difference between the treatment of each charge chunk in a given step (i.e. the difference between each j in Algorithm 1) is the diffusion applied, which in this implementation is nominally less than $7 \mu\text{m}$. The diffusion in this implementation is scaled according to a pre-calculated value that assigns an average diffusion length of $7 \mu\text{m}$ for a $300 \mu\text{m}$ piece of silicon and scales it according to the actual path length. The diffusion, as was seen in equation 3.7b, is dependent on both temperature and charge carrier mobility (and hence electric field) and is not constant. The newer versions of the signal formation model thus implements the Einstein diffusion relation via:

$$\sigma = \sqrt{2Dt} \quad (3.21)$$

where σ is the spread of the arrival position, D is the diffusion constant, and t is the time the particle is in motion for. This, however, does not account for effects at large bias voltages which affect the mean free energy of the electron [59] and hence the mean diffusion. Studies are currently underway by the PixelOffline subgroup to decide how best to model this specific dependence.

3.5 Digitization restructure

This section describes efforts to reduce the number of classes and duplicated code in the PixelDigitization package and simplify the workflow. This discussion is preceded by a brief summary of recent modifications, by another student, to the energy deposition portion of the code, included here to motivate the restructure.

3.5.1 New energy deposition model

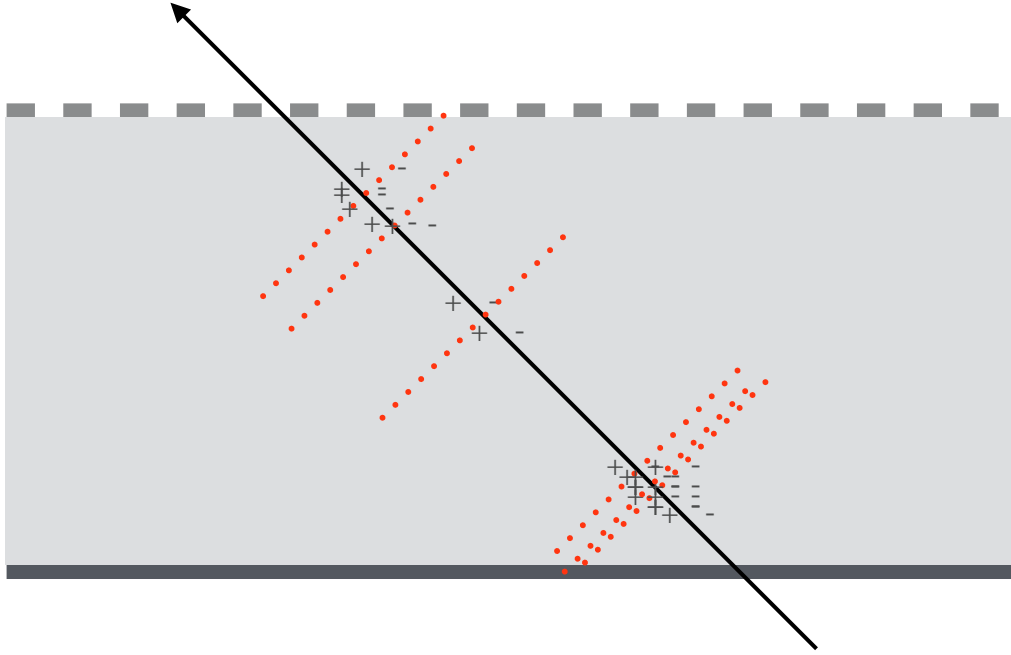


Figure 3.30: A Poisson-distributed spatial distribution of ionisation. The energy lost by the traversing particle in the sensor bulk is sampled from a distribution that includes the Landau-Vavilov-Bichsel corrections. The red lines show the collision points of the particle with the bulk, note that the amount of energy, and hence the number of charge carriers, deposited per collision is not the same from collision to collision.

An energy deposition model which is more representative of the underlying physics in ionisation was implemented prior to the author's work on the project. Solid state ionisation does not spread the deposited energy uniformly throughout the bulk, instead the particle traversing the bulk undergoes random collisions with lattice electrons and can deposit a range of energies in each collision, as was described in section 3.2. Bichsel suggests a good way to simulate the spatial deposition of energy in the 2006 review [70], which is the approach followed in the Athena implementation, described below, so the tools that implement this model have Bichsel in the name, for example, PixelBarrelBichselChargeTool.

The new energy deposition model, referred to as the Bichsel model hereafter, does not start off with a pre-defined number of steps or charge chunks but first runs a Monte Carlo over the path length of the particle in the detector volume. This Monte Carlo proceeds by simulating each collision in the bulk and then deciding on the energy loss of that collision. This is done with a Poisson distribution describing the probability, $P(n)$, of a particle undergoing n collisions with the detector material in a segment, x :

$$P(n) = \frac{m_c}{n!} e^{-m_c} \quad (3.22)$$

where m_c is the average number of collisions for all particles in x :

$$m_c = x\Sigma_t = \frac{x}{\lambda} \quad (3.23)$$

where Σ_t is the integral of the the differential cross-section multiplied by the density of the material and λ is the mean free path between collisions. The differential cross-section is usually computed from the Fermi Virtual Photon (FVP) approach, which requires the $\beta\gamma$ of the incident particle, and various properties of the detector, such as the dielectric function and the atomic number. As such, look-up tables for 6 particles including the electron, muon, proton, pion, and kaon, produced in the event generation have been added to the package. If a particle produced in the event generation is not in that list then the default uniform deposition model is used. The look-up tables contain entries for a particle's $\beta\gamma$, Σ_t , and the most probable collision energy $ColE$. The procedure for the Bichsel deposition model then goes as follows:

1. Obtain the traversing particle's type (pdgID), e.g. electron, proton, etc, and $\beta\gamma$ from the **genPart_4V** handle, which contains the four-vector of the particle from the event generation.
2. Using this information, access the look-up tables and determine the integrated cross-section, Σ_t (and hence the mean-free path before a collision occurs, for a particle of this type, this energy, in this material).
3. Enter a simulation loop. For each step in the loop throw a random number for the distance travelled by the particle before a collision happens.
4. Check to see if the particle has travelled outside the detector volume.
5. If not, throw another random number, between 0 and Σ_t and interpolate the collision energy based on this value and $\beta\gamma$. Check that the energy loss is less than the amount needed to produce a delta ray. If not, ignore it as delta rays are already simulated by Geant4.
6. Record the position and energy loss and save to the running total.

This is also summarised in Algorithm 2. A couple of things to note about this method are that it no longer uses the total energy loss provided by Geant4, however since most of the particles traveling through the detector are minimum ionising, the differences between what Geant4 propagates to the calorimeters and the particle energy loss simulated in the pixel layers is negligible. This simulation also always uses the particle's starting energy from the event generator, and so may overestimate $\beta\gamma$, however for the same reasons above this discrepancy is likely negligible.

Although not shown explicitly here, it's worth noting that the `initial_num_chunks` and `initial_num_steps` are still used to cluster the energy deposition via a weighted mean such that the signal formation part of the code still has the same number of steps.

This model has recently been validated against data showing it to more accurately represent cluster shapes seen in data¹⁰. More verification will be made with recent data once the radiation damage model, which also changes cluster shape, is implemented in Athena.

¹⁰See doctoral thesis of Qi Zeng of Stanford University (2017), unpublished at time of writing.

Algorithm 2 New Bichsel energy deposition simulation, behaviour covers functions in ChargeTools and BichselSimTool.

```

1: pdgID := genPart → pdg_id()
2: if deltaRay then
3:   break
4: else
5:    $\beta\gamma := \text{getBetaGamma}(\text{genPart})$ 
6: end if
7:  $\Sigma_t := \text{lookupMaxIntXsection}(\text{pdgID}, \beta\gamma)$ 
8:  $\lambda := \frac{1}{\Sigma_t}$ 
9: while 1 do
10:   $x_i = -\lambda \cdot \ln(\text{rand}[0, 1])$ 
11:  if ( $x_{total} + x_i$ ) > pathLength then
12:    break
13:  end if
14:  index := rand( 0,  $\Sigma_t$  )
15:   $e_i := \text{lookupEnergy}(\beta\gamma, \text{index})$ 
16:  if ( $e_{total} + e_i$ ) > deltaRayCut then
17:    break
18:  else if ( $e_{total} + e_i$ ) > genPart → getEnergy() then
19:     $e_{total} += (\text{genPart} \rightarrow \text{getEnergy}() - e_{total})$ 
20:    break
21:  else
22:     $x_{total} += x_i$ 
23:     $e_{total} += e_i$ 
24:  end if
25: end while

```

3.5.2 Restructure of subCharge tools

At the time of writing the PixelDigitization package contains nine, near-identical classes which derive from the SubChargesTool: DBM, IBLPlanar, IBL3D, PixelBarrel¹¹, PixelEC, IBLPlanarBichsel, IBL3DBichsel, PixelBarrelBichsel, and PixelECBichsel. The differences between most of the classes are minimal:

- The PixelBarrel and PixelEC classes are identical.
- The IBLPlanar and PixelBarrel/EC tools are identical except for an additional 20 lines of code in the IBL tool that accounts for the reduced collection efficiency in IBL planar sensors due to their slim-edge design.
- The DBM tool differs from the PixelBarrel/EC tools only by its hardcoded physical constants (e.g. diffusion constant, number of charge-carrier pairs produced per eV) and a rudimentary treatment of trapping to change the charge collection efficiency. The code is not commented, so it's not clear where the hard-coded values come from. Since they are not dependent on any parameters drawn from the conditionsDB it is assumed this trapping treatment is not related to radiation damage. The DBM code was not within the scope of this project and will not be discussed further.
- The signal formation portion of the 3D charge tools differ quite extensively from its planar equivalent but shares the same energy deposition model. The 3D tool is not studied in this thesis¹².
- All the comparisons above, excluding DBM, extend to the Bichsel equivalent tools.

Within the planar-sensor classes, the differences between the tools are either non-existent or minimal. All tools, regardless of material or sensor dimensions, share the same energy deposition model. Moreover, the Bichsel tools incorporate the uniform treatment of energy deposition as a backup in case the Bichsel deposition fails, so the energy deposition model is also duplicated in the Bichsel tools. The situation is summarised in the left cartoon of Figure 3.31, where the SurfaceChargesTool picks one of these ‘technology’ classes based on what part of the detector the module resides. To increase code comprehension and reduce the number of classes in the PixelDigitization package, the class-structure on the right side of Figure 3.31 was implemented. All subChargesTools implement energy deposition and signal formation somewhat separately, moreover all share the same energy deposition model which is independent of sensor geometry. So this portion of the code is extracted into a separate class, the EnergyDepositionTool and is given flags depending on whether the energy deposition occurs in silicon or diamond.

The signal formation models then can be divided into planar and 3D sensor geometries. The energy deposition tool can decide which one to use in the same getTechnology() function call that determines the sensor material. Then, only two signal formation tools are needed: one for 3D implants and one for planar implants.

The way the signal formation tools work means that pixel geometry, sensor thickness, and sensor size are stored in a database of sensor geometries currently obtained by the SiDetectorElement class. So the SignalFormationPlanarTool can be used for IBL, the three outer layers, and the DBM. To deal with the additional constraints on IBL and DBM, a flag is passed into the tool to assert the relevant code blocks. In this way nine classes is reduced to three. This is also useful for extending to future technologies such as non-hybrid CMOS modules, modifying the existing structure.

¹¹The naming convention here is to refer to the pixel layer closest to the beam-pipe as IBL and the outer 3 layers as ‘Pixel’.

¹²Please see the summary note [86] for the full treatment of 3D and planar sensors.

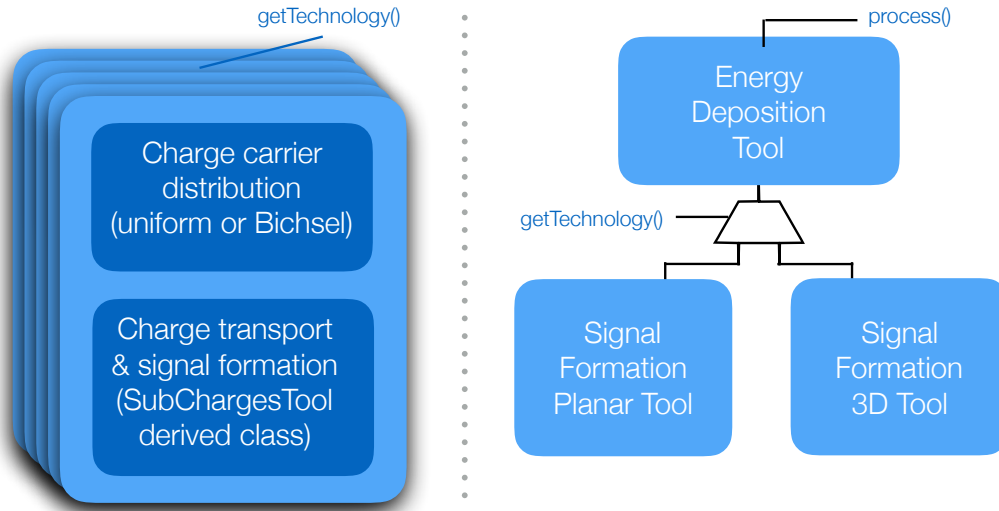


Figure 3.31: A graphical representation of how different types of sensor geometry are handled in the current PixelDigitization chargeTools release (left) and how the restructured tools handle different sensor geometry (right).

It should be noted that this version of the restructured chargeTools was not selected for inclusion in the next release of the Athena PixelDigitization package and instead a different contributor's restructure, that keeps the energy deposition and signal formation in the same class, was chosen. The selected approach still reduces the number of subChargesTools from nine to three.

3.6 Radiation damage simulation

This section will describe how specific features of radiation damage are modeled in simulation and how this simulation was integrated into the Athena chargeTools. Radiation damage is introduced to the signal formation simulation by accounting for trapping and the change in the electric field shape. Surface charge effects are considered indirectly by the change in electric field shape near the electrode implants.

3.6.1 Trapping time and position

Charge trapping can be simulated according to a characteristic lifetime, which is dependent on fluence. The fraction of charge carriers that are *not* trapped after some time, $N(t)$, is given by equation 3.24.

$$\frac{N(t)}{N(0)} = e^{-\frac{t}{\tau}} = e^{-\beta_d \Phi t} \quad (3.24)$$

where $N(0)$ is the initial number of free charge carriers, τ is the characteristic lifetime which is given by the damage parameter, β_d , and the fluence, ϕ , [78]. Trapping is simulated by throwing a random number for the fraction carriers not trapped and deducing for how long the chunk of charge carriers would have to move in the bulk for that to happen.

$$t_{\text{trap}} = -\frac{1}{\beta_d \Phi} \ln(u) \quad (3.25)$$

where $u \in [0, 1]$ is a random number drawn from a uniform distribution, and t_{trap} is the time the charge carriers move for until they are trapped, also known as the trapping time. To determine

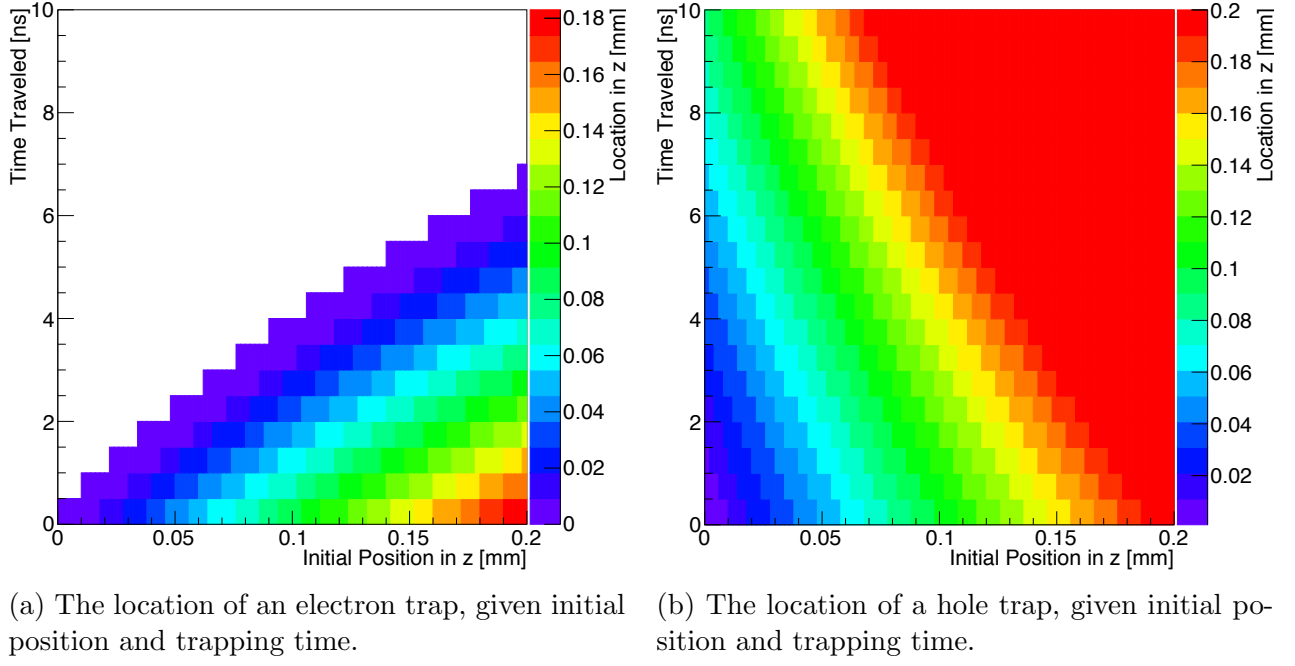


Figure 3.32: The location of a trap given the charge carrier starting position and trapping time. The electrons are collected at 0 and the holes at $200\ \mu\text{m}$ in this figure. The plots were produced using the simplified model for unirradiated IBL planar sensors, courtesy of B. Nachman [86].

if a chunk of charge carriers is trapped, the trapping time is compared to the time-to-electrode, which is looked up in the stored values shown in the previous section. If the trapping time is less than the time to electrode, the chunk is said to be trapped. Once the charge chunk is trapped, the trapping time can be used, along with the Lorentz angle and carrier mobility, to find the position of the trap. It is assumed that charge carriers trapped in the signal formation process do not escape and so parts of the signal from ionisation will be lost. The position of the trap in the bulk is used to calculate the charge induced on the collecting electrode, which is explained in the following section. The trap-location calculations are performed offline and stored in look-up tables, an example for an unirradiated IBL planar sensor is shown in Fig. 3.32.

3.6.2 Signal formation in a damaged bulk

Charge-carrier pairs, created by a high energy particle ionising the bulk of a silicon detector, when coupled with a charge sensitive amplifier, induce a signal as they move in the bulk. Signal formation does not begin when the charge carriers recombine at their respective electrodes but instead from when they begin to move in the electric field, essentially immediately after ionisation. Initially, it seems that modeling charge collection only as recombination would not affect the simulation. The integrated induced charge is the same if the charge is drifted in the electric field as if it is just assumed to recombine at the electrode. There would be some offset in the timing: if an electron-hole pair was formed near the backside of an n^+ -in- n detector, the free electron would have to travel the entire depth of the bulk before it recombined, which takes on the order of 10 ns. If instead a majority-holes sensor was used, like the SCT p^+ -in- n , a hole travelling the full width of the bulk could take on the order of 50 ns or 2×25 ns bunch-crossings to arrive. However, since the shape of an induced signal is not simulated, this would not affect the simulated charge in the absence of trapping.

In Section 3.6.1 charge carrier traps were introduced. If a charge carrier is trapped in the bulk,

it will not be collected by its respective electrode. Nevertheless, a charge carrier that is trapped will induce a non-zero signal on both electrodes up to when it is captured. In this case, the induced charge on the electrodes is not equal to the collected charge and has to be calculated separately. This section will discuss different ways to calculate the induced charge and how they have been optimised for simulation.

Prior to the 1930's, induced charge was calculated by moving a charge along the electric field lines of the detector in small increments and calculating the corresponding change of voltage on the detector surfaces by assuming the total energy of the system had not changed. This form of iterative 'energy balancing' did not account for defects in the lattice nor space-charge [92]. Another way of calculating the charge induced by a moving charge carrier on an electrode is to solve Gauss's law. Doing so requires a detailed Electric field map that takes into account the bias voltages, space-charge, and moving charges at every time instance. Whilst this may be possible for simple geometries such as parallel plate capacitors it becomes a lengthy and impractical calculation for larger, more complex geometries.

Ramo-Shockley weighting potential

In the late 1930's Shockley and Ramo independently published a means to calculate the induced charge on an electrode [93] [94]. The resulting 'weighting field' provides a means of calculating the charge induced on an electrode simply from knowing the geometry of the system, and the displacement of the charge carrier over its full range of motion. The weighting field can be derived from setting up the following two cases. First a single charge carrier is surrounded by some complex electrode geometry, see Fig. 3.33. If all the electrodes are grounded, Gauss's law can be used to calculate the flux of the carrier through a Gaussian surface, shown as a sphere of potential V_e in Fig. 3.33a. Starting from Green's theorem [95]:

$$\int_{\text{vol.}} (V' \nabla^2 V - V \nabla^2 V') dv = - \int_{\text{surf.}} \left(V' \frac{\delta V}{\delta n} - V \frac{\delta V'}{\delta n} \right) ds \quad (3.26)$$

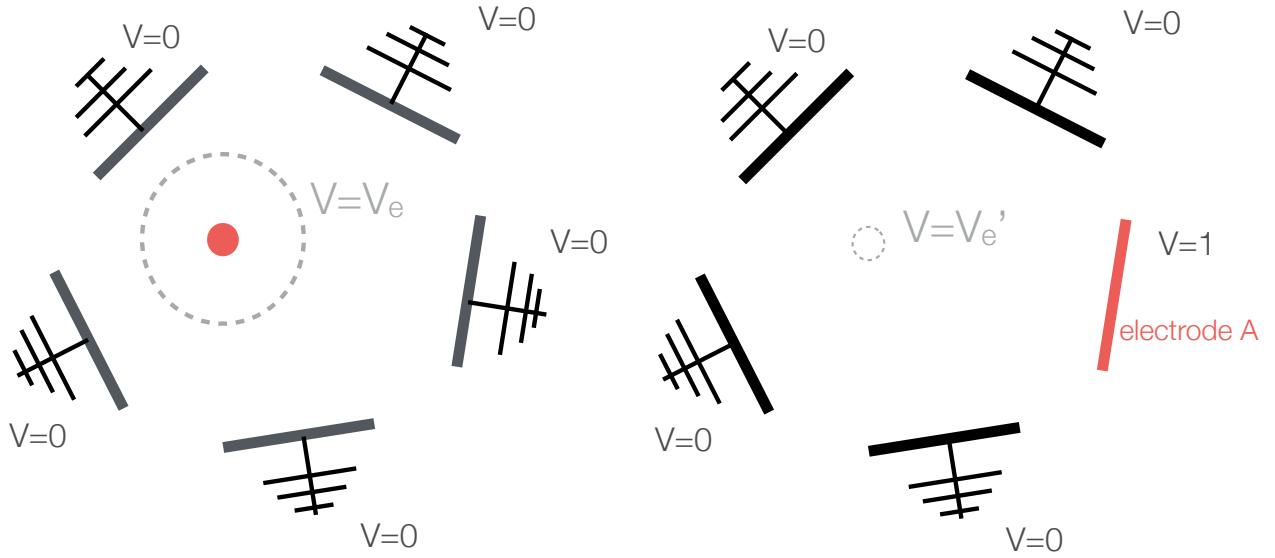
where V is the potential of the electric field between the conductors, $\frac{\delta}{\delta n}$ is the differential with respect to the normal of a surface. If the test charge is removed from the geometry and one of the electrodes, historically called electrode A , is raised to have a unit potential, the potential of the field is then V' , see Fig. 3.33b.

Green's theorem is then evaluated for these cases. If the volume evaluated on the left hand side is bounded by all conductors, both the Laplacians evaluate to zero as there is no net charge. The right hand side of the equation can then be evaluated for three cases, where a surface integral is performed over:

1. All electrodes, except A , will, by definition, evaluate to 0.
2. Electrode A will result in the first term in equation 3.27.
3. The equipotential sphere will result in the second term in equation 3.27.

Adding the results of these three cases together to form the complete picture results in equation 3.27. From equation 3.26, the induced charge, Q_A , on electrode A can be derived:

$$\begin{aligned} 0 &= - \int_A \frac{\delta V}{\delta n} ds - V_e' \int_{\text{sphere}} \frac{\delta V}{\delta n} ds \\ &= 4\pi Q_A + 4\pi e V_e' \\ \therefore Q_A &= -e V_e' \end{aligned} \quad (3.27)$$



(a) A test charge in a complex geometry of grounded electrodes.

(b) The test charge in figure (a) is removed and an electrode is raised to unity potential.

Figure 3.33: A graphical representation of the derivation of the weighting field from Green's theorem.

where V_e' is the weighting potential [93]. The weighting potential, unlike the electric potential, depends only on the geometry of the electrodes and not operational conditions such as bias voltage or temperature. This is a powerful tool as the weighting potential can be calculated and stored ahead of time, reducing the need to perform calculations during simulation.

For radiation damage modeling the weighting potential is used to find the induced charge on an electrode if the charge carrier is trapped. This is done by looking up the value of the weighting potential at the starting position of the charge carrier and the value of the weighting field at the position where the charge carrier is trapped:

$$Q_{\text{induced}} = e \left(\phi(\vec{s}_{\text{end}}) - \phi(\vec{s}_{\text{start}}) \right) \quad (3.28)$$

where $\vec{s} = (x, y, z)$ is the position of the charge carrier in the bulk, ϕ is the weighting field, and e is the charge of an electron. An example of the weighting potential for a single pixel at increasing sensor depth is shown in Fig. 3.34, the maps shown in the figure are those used in the simulation of IBL planar sensors.

Induced charge on neighbouring electrodes

The weighting potential can also be used to easily calculate the induced charge on adjacent electrodes. In Strip and Pixel detectors, collecting electrodes are closely packed and it is feasible that a free charge carrier could pass close to more than one before either recombining or being trapped. Without trapping, the charge carrier travels to an electrode and recombine and any signal induced on an adjacent electrode integrates to zero [96]. The current pulse on the adjacent electrodes is bipolar, see Fig. 3.35, which means that if the charge carrier is trapped a non-zero charge will be integrated on that electrode. This effect is accounted for by looping over all nearest neighbour pixels, see Fig. 3.36. A weighting potential calculation on the geometry of ~ 2.5 pixels is made beforehand and stored in a Root 3D histogram, which is essentially used as a look-up table at runtime. By storing the weighting field for more than one pixel, the charge induced on neighbouring electrodes can be checked easily.

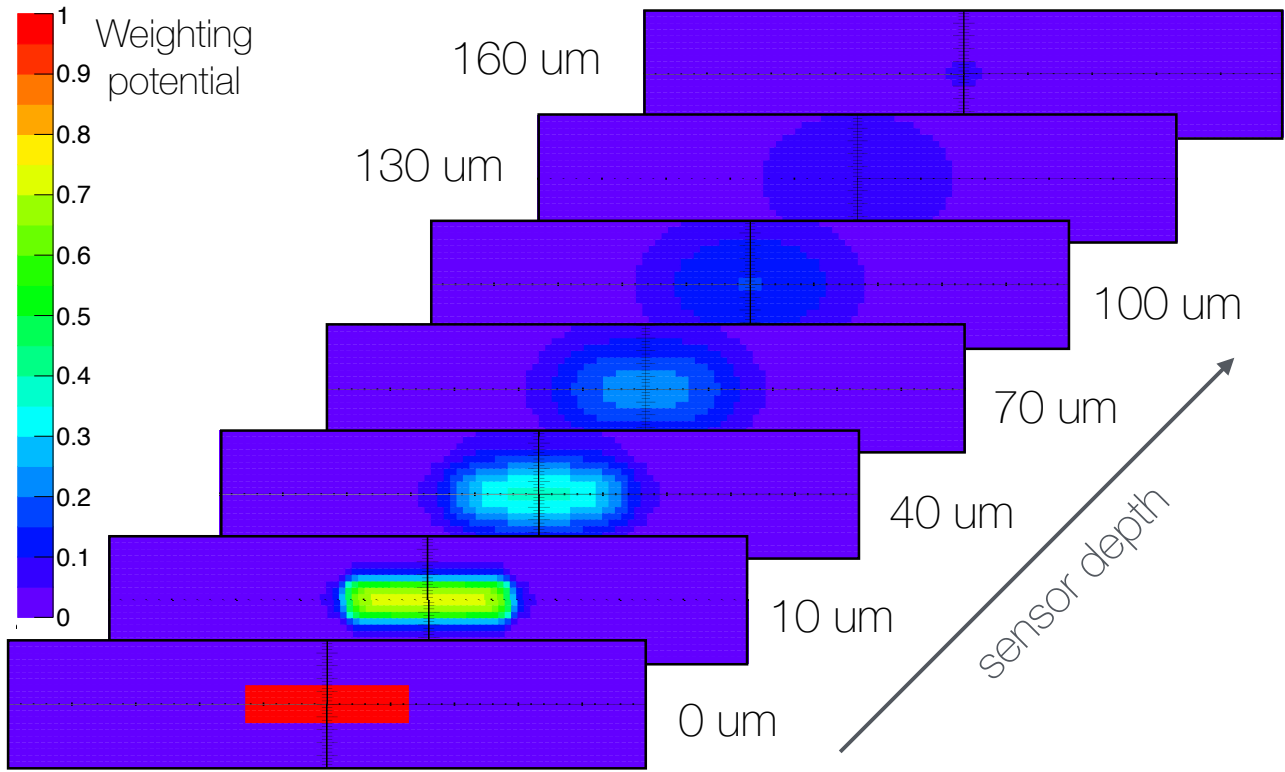


Figure 3.34: Slices of the generated 3D weighting potential map shown at different depths in the sensor. The weighting potential maps is of a quarter of an IBL pixel ($50 \times 250 \mu m$) which extends for 2.5 pixels but is shown here for the full pixel.

3.6.3 Charge chunks

Aside from dealing with electrons and holes separately, an additional consideration in radiation damage simulation that wasn't necessary in the original signal formation simulation is that charge carriers are not propagated individually but in chunks. A particle traveling perpendicular to the sensor plane will produce $\approx 20,000$ charge carriers and, for the default chargeTool settings of 10 steps and 50 chunks per step, each chunk of charge will be composed of ≈ 40 charge carriers. In the simulation a random number is drawn for each charge chunk to decide whether it is trapped or not and a trapping time is calculated, as was shown in equation 3.25. However equation 3.25 is only valid for single charge carriers and not chunks of charge. Although, on average, the amount of induced charge will be the same when using chunks rather than individual charge carriers, the variance of the induced charge will be overestimated if there are many fundamental charge carriers in each chunk.

To deal with this the final step applied before the induced charge is added to the chargedDiodeCollection is to 'unsmear' or apply a correction that ensures the variance of the trapped charge carriers is correct. This is done as shown in equation 3.29, by replacing the induced charge with:

$$Q \rightarrow \langle Q \rangle + \frac{1}{\sqrt{N}} (Q - \langle Q \rangle) \quad (3.29)$$

which clearly maintains the same average but reduces the individual contribution by a factor of $N^{-\frac{1}{2}}$. This correction is applied to both trapped and not-trapped charge in the simulation. The effects of charge chunk unsmearing can be seen in Fig. 3.37, which applies the same corrections in a 3D sensor.

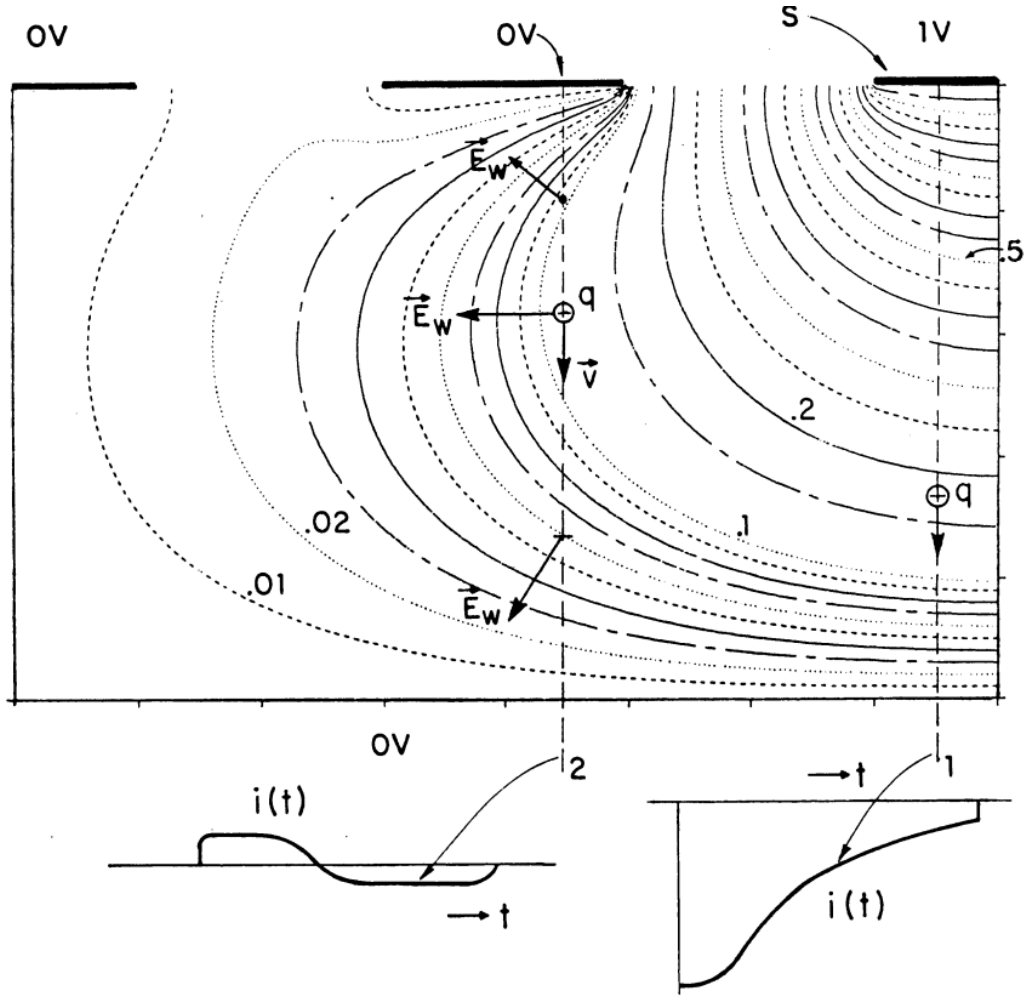


Figure 3.35: A charge carrier moving in the weighting potential, with the field norm. labelled $\vec{E}_w$ (main image) and resulting current flow in strip detector electrodes (bottom). The weighting potential shown here is calculated for the collecting electrode, labelled '1'. Adjacent electrodes are labelled '2' [96].

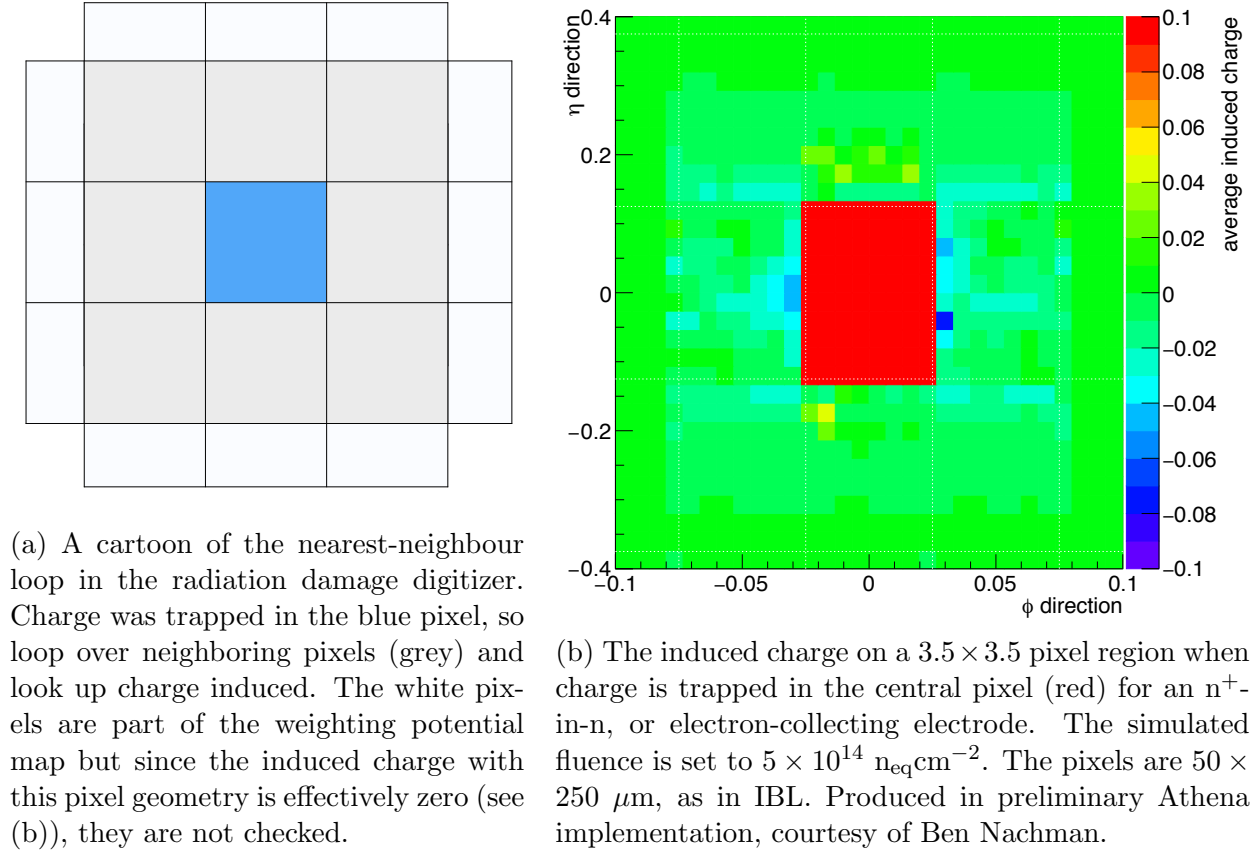


Figure 3.36: The effect of trapping on induced charge on neighbouring pixels.

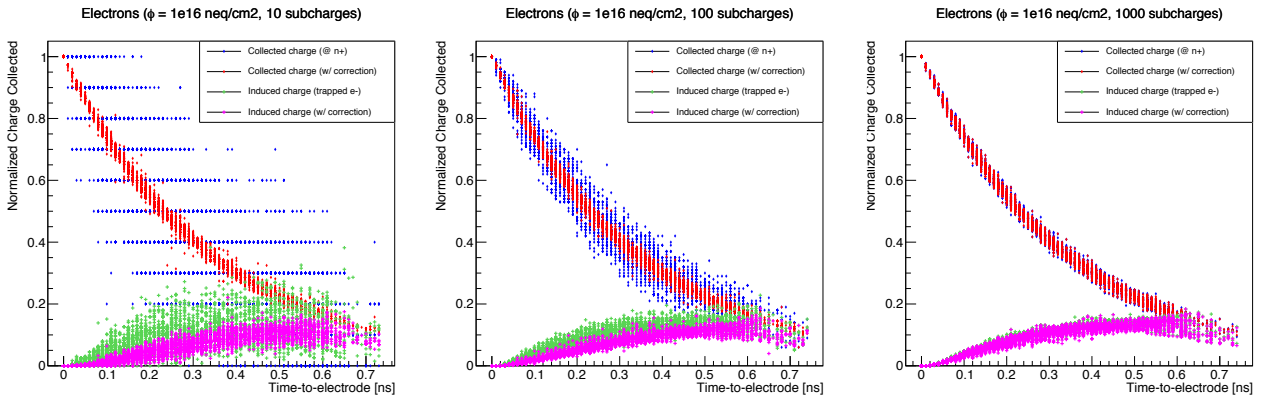


Figure 3.37: Unsmearing of the distribution of collected charge when using chunks of charge as opposed to individual charge carriers when simulating trapping. The plots were made from data in the Allpix 3D IBL simulation, which uses the same correction formula applied to the planar sensors. The spread of induced charge decreases from left to right, without unsmearing applied, as the amount of charge per chunk (or the total number of charge chunks) decreases. Plots courtesy of V. Wallangen, [86].

3.6.4 Athena implementation

The entry point to radiation damage modeling occurs after energy deposition. As before, there is a loop over each step, where energy has been deposited, and a loop over the charge chunks in each step. In the radiation damage signal formation, there is an additional loop that runs twice over the signal formation code, once for electrons and once for holes which is necessary as they experience trapping at different rates. The various contributions to the motion of the charge carrier before being trapped are shown in Fig. 3.38. The radiation damage simulation then

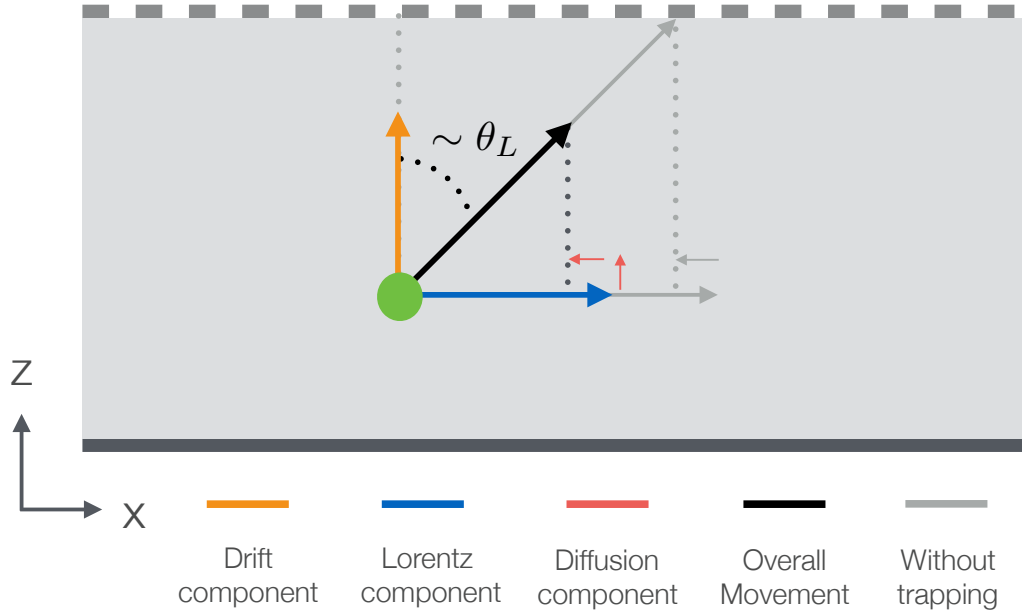


Figure 3.38: The contributions to signal formation from a trapped chunk.

is then comprised of a few steps that include all the considerations mentioned in the previous section and is also summarised in Algorithm 3.

1. Based on the current position in the bulk, look-up the time it would take the charge carrier to reach its respective electrode using the maps that take into account the modified electric field shape and Lorentz angle for the given operating conditions and fluence, which are both taken from the conditions database for this Run and module.
2. Throw a random number and calculate the trapping time, as was detailed in section 3.6.1.
3. If the trapping time is less than the time taken to reach the electrode, the charge carrier will be trapped and so enter the trapping loop. Else, drift and diffuse the charge carrier to its respective electrode, as was done in the original chargeTool. Next apply the unsmearing, using a look-up to determine the correct $\langle Q \rangle$.
4. For the trapped charge the next step is to look up the trap depth based on the charge carrier's initial depth in the bulk and the trapping time. The trap depth does is independent of the lateral drift from the Lorentz angle which is also looked-up.
5. A random number is thrown to apply diffusion to the drifted charge carrier.
6. The final charge position is then used to find the weighting potential value at this position. The weighting potential at the initial position in the bulk is also found.
7. Next the nearest neighbour pixels are looped over and the weighting potential value at the centroid of the neighbor pixel is looked up for the start and end position of the charge

carrier in each.

8. The induced charge is then calculated for the 3×3 pixel region and the unsmeared factor is again applied before adding the induced charge to the chargedDiodesCollection.
9. Any areas with a low electric field, such as the IBL slim edges can be applied as a scaling factor to the induced charge in the same way it was applied in the original chargeTools.

These steps are shown in Algorithm 3. Although there are many additional steps in simulating

Algorithm 3 The signal formation simulation with radiation damage

```

1: [fluence, sensorBias, temperature,  $\langle Q \rangle$ ]  $\leftarrow$  getConditions( runNumber)
2: carriersPerChunk := totalEnergyDeposited  $\div$  (3.6  $\cdot$  nChunks  $\cdot$  nSteps)
3: for ( $i < \text{nSteps}$ ) do
4:   energyDeposited in this step := 0.5  $\times$  hitRecord[i].energy()
5:   initialPosition := hitRecord[i].position()
6:    $\phi_i \leftarrow$  lookupWeightingPotential( initialPosition )
7:   for ( $j < \text{nChunks}$ ) do
8:     if doRadDamage then
9:       for ( $k < 2$ ) do {Holes and electrons loop}
10:        timeToElectrode  $\leftarrow$  lookupTimeToElectrode( isHole, sensorThickness, depth $i$ )
11:        trappingTime  $\leftarrow -(\beta_d \Phi)^{-1} \ln(\text{rand})$ 
12:        if (trappingTime < timeToElectrode) then {charge is trapped}
13:          trapLocation[ z ]  $\leftarrow$  lookupTrapLocation( trappingTime, isHole, depth $i$ )
14:          trapLocation[ x ] += lorentzDrift( depth $i$ , trapLocation[z], isHole )
15:          trapLocation += diffusion( rand )
16:          for (x_index = -1 to 1) do {nearest neighbour loop}
17:            xPixel := trapLocation[x] + (x_index  $\cdot$  x_pixelPitch)
18:            for (y_index = -1 to 1) do {nearest neighbour loop}
19:              yPixel := trapLocation[y] + (y_index  $\cdot$  y_pixelPitch)
20:               $\phi_f \leftarrow$  lookupWeightingPotential( xPixel, yPixel, trapLocation[z] )
21:              smearedCharge := energyDeposited  $\cdot$  (3.6 eV)-1  $\cdot$  ( $\phi_f - \phi_i$ )
22:              inducedCharge =  $\langle Q \rangle - (\text{carriersPerChunk})^{-1/2} (\text{smearedCharge} - \langle Q \rangle)$ 
23:            end for
24:          end for
25:        end if
26:      end for
27:    else { ! doRadDamage }
28:      do original signal formation code in Algorithm 1
29:    end if
30:  end for
31: end for

```

the signal formation, many of them involve looking up values in pre-calculated maps, for example the weighting potential. However there are also two additional loops: one to treat electrons and holes separately and one to find the nearest neighbour induced charge. As this code is still in development, performance tests within Athena have not yet been run but care is being taken to minimise the number of calculations needed, for example by incorporating the drift due to the Lorentz angle into the distance maps.

The radiation damage code can be included in any of the current signal formation tools by adding a flag that allows the radiation damage simulation portion to be accessed. Maps of

stored values should also be added somewhere so as to be accessible to the code. There has been some discussion as to whether these maps should be stored locally in the PixelDigitization share folder or in the conditionsDB. The majority of the maps should be stored in conditions as their values rely heavily on the operating conditions of the detector and how much fluence it has been exposed to. The chargeTool or SignalFormation tool should not decide which fluence and map to access, this should be decided by some external service given the module ID and Run number associated with the simulated hit. The weighting potential is only dependent on sensor geometry and so should not be kept in the conditions database but rather the service or tool that picks the module technology, for example planar or 3D, IBL sensor or outer layer sensor. The setup of these support services that interact with the conditions/geometry databases has not yet been tested but discussions have been started between the radiation damage working group and the PixelOffline group with regards to this specific issue.

3.7 Outlook

The author's task in this project was to take the radiation damage simulation under development in the standalone Geant4 simulation framework, Allpix, and migrate that code to be compatible with code in Athena. This involved identifying the correct stage of Athena simulation (digitization), where the simulation should be placed in the flow of digitization (subChargesTool/signalFormation), and implementing the model. In doing so, various physical and technical aspects of the simulation were also improved. However this is only the first step in the Allpix-to-Athena migration. Aside from setting up the appropriate Services/Tools to handle the database tools, the code now needs to be validated. This will include running unit tests on the code if zero fluence is applied, comparing the cluster shapes produced with a particle gun at a specific angle to those found in modules irradiated in testbeams at the same angle, comparing the ToT in simulation and irradiated module data, and checking how well the simulation matches reconstructed data from LHC Run 2 and 3. Alongside these efforts the radiation damage working group is continuing to validate the individual physical parameters against data from the detector, for example the Lorentz angle modeling, fluence, and annealing.

Chapter 4

Conclusion

The HL-LHC presents an opportunity to not only extend our understanding of particle physics but also augment our instrumentation. Development for HL-LHC Pixel detectors are at the cutting edge of what is physically achievable with the energy budget available and data rates that can be physically read off the detector. How then does the instrumentation adapt to deal with future colliders?

There is already a finite amount of data processing done within the chip itself, for example digitizing an analogue integrated charge measurement to ToT. It might be possible in the future to extend this further and perform additional processing on-chip. For example, on-chip clustering that can discard noise hits at shallow angles in the forward region of the detector. Or, perhaps, inter-chip communication, to highlight regions of interest or even perform rudimentary track building. These latter two points specifically, would require some tracking-based, on-chip processing which could derive from a Kalman filter.

On-chip clustering and tracking would not only have to be done to a high degree of accuracy, but would also have to be performed within the trigger latency. A feature of TrueNorth that could be exploited to provide the speedup necessary to perform these computations is lossy compression for approximate computing. For example, one of the tests performed in TrueNorth was the resilience of the neuromorphic Kalman filter implementation to truncated data, for speedup. It was found that the Kalman filter did not diverge from the mean value of the state being tracked, but produced a state estimate with a higher noise. Perhaps truncating data inside the chip, before performing computations would make online processing more feasible to complete within the trigger latency.

If inter-chip communication for online track reconstruction seems unlikely, then processing off-detector, but within time to be useful to the trigger system, can be considered. Whilst track triggering is already being pursued in the form of pattern matching in associative memories in the ATLAS detector FTK for HL-LHC, active track reconstruction during data taking is not. Part of the difficulty in processing data in a high-radiation environment, such as inside or close to the ATLAS detector is the possibility of bit-flips. Bit flips can occur due to the high fluence of particles passing through the electronics, so fault tolerance is useful in high multiplicity environments. Although spike encoding is highly inefficient, it is very fault tolerant: a bit flip in a spike train will only change the encoded value by ± 1 whereas in binary a single bit flip can change the encoded value by $\pm$ the flipped bit. Although the low power offered by processing sparse spike trains might not be useful off-detector, on detector it might also be a useful feature as it could lessen the energy budget by reducing the amount of cooling needed. However, if spikes were used they would have to be transduced or a native sensor used, both of which could offset any benefits from cooling and introduce more latency into the

system.

These question of how exactly these features could be added to the computing infrastructure do not necessarily need to be answered for the detectors in the HL-LHC but could be of interest to future collider experiments. In a post-Moore's law field of computation, if technology does not grow at the rates that are needed to meet the requirements of future multiplicities, alternative data processing methods drawing from technology such as neuromorphic computing may be considered.

Whilst the three outermost layers of the current Pixel detector have an 8 bit ToT resolution, the Pixel detectors produced for HL-LHC will likely have a 4 bit resolution. The quantisation error, from having a smaller ToT register, will have to be decoupled from the effects on cluster size and shape of radiation damage in future detector's signal formation. As such, carefully modeling microscopic defects in silicon ensure that the behaviour of the detector is well understood and allows us to decouple simulation error from real anomalies in the signal that might be evidence for new physics. For example, highly-ionising particles may leave very different cluster shapes in the detector to those from Standard Model particles. Decoupling the cluster shape from radiation damage effects and quantisation errors is an important first step in making that distinction.

Whilst top-level data taking schemes are necessary to ensure that data can still be physically recorded at future colliders, understanding our detectors at the fundamental level is also essential for exploring data for new physics.

Appendices

Appendix A

TrueNorth

A.1 Complete neuron description

The full neuron equation in TrueNorth contains many parameters that were not used in the final implementation of the Kalman filter. They are included here for completeness. Equation A.1 describes the complete neuron update equation, including the concepts of both leak and stochasticity that were not described in the main text. The symbols in equation A.1 are described in table A.1.

$$\begin{aligned}
 V_j(t) &= V_j(t-1) + \sum_{i=1}^{256} A_i(t) \times w_{i,j} \times \text{sgn}(s_j^{G_i}) \times \text{weight} \\
 \text{weight} &= \begin{cases} |s_j^{G_i}|, & \text{if } b_j^{G_i} = 0 \\ 1, & \text{if } (b_j^{G_i} = 1) \wedge (s_j^{G_i} \geq \rho_j^{G_i}) \\ 0, & \text{otherwise} \end{cases} \quad (\text{A.1a})
 \end{aligned}$$

Leak update:

$$\begin{aligned}
 V_j(t) &+= \Omega_j \times \text{leak} \\
 \Omega_j &= \begin{cases} \text{sgn}(\lambda_j), & \text{if } \varepsilon_j = 0 \\ \text{sgn}(\lambda_j \times V_j(t)), & \text{if } \varepsilon_j = 1 \end{cases} \\
 \text{leak} &= \begin{cases} \lambda_j, & \text{if } c_j = 0 \\ 1, & \text{if } (c_j = 1) \wedge (\lambda_j > \rho_j^\lambda) \\ 0, & \text{otherwise} \end{cases} \quad (\text{A.1b})
 \end{aligned}$$

The potential, V_j is capped at a ceiling and floor defined as:

$$\begin{aligned}
 \text{ceiling} &= \begin{cases} \alpha_j + M_j, & \text{if } \gamma_j = 2 \\ 393216, & \text{otherwise} \end{cases} \\
 \text{floor} &= \begin{cases} -\beta_j - M_j, & \text{if } \gamma_j = 2 \\ -393216, & \text{otherwise} \end{cases} \quad (\text{A.1c})
 \end{aligned}$$

The neuron will cross the positive threshold and fire if:

$$V_j > (\alpha_j + \eta_j) \quad (\text{A.1d})$$

The neuron will cross the negative threshold if:

$$V_j(t) < \begin{cases} -\beta_j, & \text{if } \kappa_j = 1 \\ -\beta_j - \eta_j, & \text{if } \kappa_j = 0 \end{cases} \quad (\text{A.1e})$$

If V_j crosses the positive threshold it will be reset according to:

$$V_j(t) = \begin{cases} Vrst_j, & \text{if } \gamma_j = 0 \\ V_j(t) - \alpha_j - \eta_j, & \text{if } \gamma_j = 1 \\ V_j(t), & \text{if } \gamma_j = 2 \end{cases} \quad (\text{A.1f})$$

If V_j crosses the negative threshold it will be reset according to:

$$V_j(t) = \begin{cases} -Vrst_j, & \text{if } \kappa_j = 0, \gamma_j = 0 \\ V_j(t) + \beta_j + \eta_j, & \text{if } \kappa_j = 0, \gamma_j = 1 \\ V_j(t), & \text{if } \kappa_j = 0, \gamma_j = 2 \\ -\beta_j, & \text{if } \kappa_j = 1 \end{cases} \quad (\text{A.1g})$$

where sgn denotes the signum function.

scope	symbol	range	description
Basic neuron update	i, j	$[1, 256]$	Axon and neuron indices, respectively
	$V_j(t)$	$[-393216, 393216]$	Neuron potential at tick t
	$A_i(t)$	$[0, 1]$	Presence of spike in axon i at tick t
	$w_{i,j}$	$[0, 1]$	Presence of synapse connecting axon i and neuron j
	G_i	$[1, 4]$	Axon label
	$s_j^{G_i}$	$[0, 2^8 - 1]$	The weight assigned spikes incident on neuron j from axons with label G_i
Stochastic weights	$b_j^{G_i}$	$[0, 1]$	Selects whether to use a static neuron weight or a stochastic weight
	$\rho_j^{G_i}$	$[0, 1]$	Output of a pseudorandom number generator to decide whether neuron will spike in stochastic mode
Basic leak update	Ω_j	$[-1, 1]$	Controls the behaviour and direction of the leak
	λ_j	$[0, 2^8 - 1]$	Leak weight
	ε_j	$[0, 1]$	Changes the sign of the leak depending on the sign of the neuron potential
Stochastic leak	c_j	$[0, 1]$	Selects whether to use a static leak or a stochastic leak
	ρ_j^λ	$[0, 255]$	Output of a pseudorandom number generator to decide whether neuron will leak in stochastic mode
Threshold and reset	α_j	$[0, 2^{18} - 1]$	Positive threshold. If V_j exceeds this the neuron will fire
	β_j	$[0, 2^{18} - 1]$	Negative threshold. If V_j drops below this it will not fire
	M_j	$[0, 2^{18} - 1]$	Acts as a mask for the reset stochasticity and also extends the ceiling for the special reset case where $\gamma_j = 2$
	γ_j	$[0, 2]$	Membrane potential reset mode
	κ_j	$[0, 1]$	Negative V_j threshold saturation or normal reset
	η_j	$[0, 2^{18} - 1]$	Masked stochastic threshold. This is a bitwise AND of $\rho_j^{TM} \& M_j$
	$Vrst_j$	$[0, 2^{15} - 1]$	Reset voltage in special reset mode

Table A.1: Complete neuron parameter list.

A.2 Neuron parameter search

The TrueNorth implementation of a Kalman filter involves finding an accurate representation of weights, for example the elements in the Kalman gain matrix, in the form of the neuron weight and threshold. The general procedure for doing this is finding a rational representation of the matrix element of the form:

$$W = \frac{s_j^{G_i}}{\alpha_j} \quad (\text{A.2})$$

Whilst $s_j^{G_i}$ is only an 8 bit signed register, its range can be extending by duplicating the associated axon input q times. For example, to extend the range of weights from $[-255, 255]$ to $[-1275, 1275]$, $q = 5$. So if it was required that $s_j^{G_i} = 1,200$, four axons could be assigned a weight affiliated with their label of 255 and one axon would be assigned the weight 180 as $4(255) + 180 = 1200$. In the case of the serial Kalman filter implementation, an entire core can be dedicated to optimising only the weight representation, so there are 256 total axons that can be used to increase the range of the neuron weight register. In this case the rational representation search consists of using the Matlab inbuilt continued fraction estimator *rat*¹. This is an iterative process the precision of the number that should be represented as an argument. For example, consider a weight, $W = 0.9182991$, if the precision is set to $1e - 3$:

$$\text{rat}(W, 1 \times 10^{-3}) = \frac{45}{49} = 0.918(37)$$

Since $45 < 255$, $q = 1$ so no additional axons are needed. If, instead, we want W to 4.s.f.:

$$\text{rat}(W, 1 \times 10^{-4}) = \frac{281}{306} = 0.9183(00)$$

Since $281 < (255 \times 2)$, $q = 2$ so one additional axon is needed. If, 7.s.f. are needed:

$$\text{rat}(W, 1 \times 10^{-4}) = \frac{1,922}{2,093} = 0.9182990(92)$$

As $1922 < (255 \times 8)$, $q = 8$ so seven additional axons are needed.

However, in the parallel implementation there are not 256 axons available to increase the range of the neuron weight and using the full range of the denominator in the ration representation of W , the threshold α_j , is not possible (see section 2.4.4) because of the reset scheme used and the number of axon-neuron pairs dedicated to parallelism. There is also the additional constraint that because each neuron can only store up to four weights, the threshold, or denominator, must be a multiple of 255. In this case an adapted rational representation search is used.

The aim is to maximise the precision to which W is represented for a given blocksize, b . Although it is possible to use the desired precision to set the maximum speedup through parallelism attainable, in practice all corelets in the parallel implementation must have the same blocksize. So it is better to pick b based on either latency constraints or the minimum acceptable precision for weight representation and then choose $p, q, s_j^{G_i}, \alpha_j$.

In the parallel implementation, the minimum stepsize (and hence resolution) attainable is based

¹The *rat* function man page: <https://www.mathworks.com/help/matlab/ref/rat.html>.

on the value of the denominator. The larger the denominator, the smaller the stepsize for weight representation, and the more precisely W can be represented. However, if more of the available axon-neuron pairs not taken up in parallelism are used to extend the threshold, this number is denoted by p , then less are available to extend the range of the neuron weight q . Using equation 2.24, repeated here for clarity: a look-up table of p and q values for a given blocksize, b , is created. This is the maximum possible value of both p and q as a pair, so a smaller value of either could always be used:

$$W \leq \frac{s_{max}}{\alpha} \quad (\text{A.3})$$

where

$$s_{max} = 255 \cdot q \quad (\text{A.4})$$

With this in mind, the search goes as follows:

1. A look-up table of p and q values for a given blocksize, b , is created.
2. Since alpha must be a multiple of 255:

$$\alpha = 255 \cdot p \quad (\text{A.5})$$

Then:

$$s = \text{round}(255 \cdot p \cdot W) \quad (\text{A.6})$$

As α gives the precision, we want to maximise p .

3. However, there is a relationship between s_{max} and s :

$$\begin{aligned} s_{max} &\geq s \\ s_{max} &\geq 255 \cdot p \cdot W \\ \frac{255 \cdot q}{255 \cdot p} &\geq W \\ \therefore \frac{q}{p} &\geq W \end{aligned} \quad (\text{A.7})$$

So p is maximised up to the point where this relationship is no longer true.

4. With p and q assigned, s and α are calculated according to equations A.6 and A.5.

So the optimal weight representation available for a given b is found by maximising p , whilst still obeying the inequality A.7.

Appendix B

Radiation Damage modelling

Supplementary material for Chapter 3, which describes Non-ionising energy loss damage modelling in silicon for the Athena framework.

B.1 List of variable name changes

For clarity, the variable names in the new digitizer code discussed in this thesis have been changed from the original source code. The table below provides a look-up to convert between the variable names of the legacy code and the variable names used in more recent additions to the PixelDigitization package, for example the radiation damage tools.

Old names	New names	Description
pos	startPosition	Particle entry point into detector volume, supplied from Geant4 (vector)
cs	endPosition	Particle exit point from detector volume, supplied from Geant4 (vector)
xEta / xPhi / xDep	eta_0 / phi_0 / depth_0	Individual components of startPosition
xEtaf / xPhif / xDepf	eta_f / phi_f / depth_f	Individual components of endPosition
xEta1 / xPhi1 / xDep1	eta_i / phi_i / depth_i	Components of position at step i in simulation
xEtaD / xPhiD	eta_drifted / phi_drifted	Components of position after drift has been applied
cEta / cPhi / cDep	totalChangeEta / totalChangePhi / totalChangeDepth	Components of the total path length
length	pathLength	
nsteps	scaledNumberOfSteps	Number of steps used in simulation loop, scaled by the path length.
ncharges	scaledNumberOfCharges	Number of charge chunks used in simulation loop, scaled by the path length.
e1	stepSizeEnergy	Amount of energy deposited by the particle in each step. This effectively determines how many charge carriers each chunk of charge represents.
spess	dist_electrode	Distance between charge carrier and electrode.
istep	i	Outer loop iterator.
i	j	Inner loop iterator.

Table B.1: Summary of variable name changes in PixelDigitization subChargesTool classes.

Bibliography

- [1] G. Aad et al. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC”. In: *Physics Letters B* 716.1 (2012), pp. 1–29. ISSN: 0370-2693. DOI: <https://doi.org/10.1016/j.physletb.2012.08.020>. URL: <http://www.sciencedirect.com/science/article/pii/S037026931200857X>.
- [2] David Galbraith. “Standard model of the Standard Model”. (Image). URL: <http://davidgalbraith.org/portfolio/ux-standard-model-of-the-standard-model/>.
- [3] ATLAS Collaboration public plots. “ATLAS_e_SMSummary_FiducialRatio”. (Image). URL: https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/CombinedSummaryPlots/SM/index.html#ATLAS_e_SMSummary_FiducialRatio.
- [4] J.J. Goodson. “Search for Supersymmetry in States with Large Missing Transverse Momentum and Three Leptons including a Z-Boson”. Presented 17 Apr 2012. PhD thesis. Stony Brook University, 2012.
- [5] “Luminosity Public Results Run2”. ATLAS Experiment: public plots. URL: https://atlas.web.cern.ch/Atlas/GROUPS/DATAPREPARATION/PublicPlots/2016/DataSummary/figs/mu_2016.png.
- [6] Claudia Gemme. “Latest ATLAS results from Run 2”. In: *9th International Workshop on Top Quark Physics (TOP 2016) Olomouc, Czech Republic, September 19-23, 2016*. 2016. arXiv: 1612.01987 [hep-ex]. URL: <http://inspirehep.net/record/1501935/files/arXiv:1612.01987.pdf>.
- [7] Mark Thomson. *Modern particle physics*. New York: Cambridge University Press, 2013. ISBN: 9781107034266. URL: <http://www-spires.fnal.gov/spires/find/books/www?cl=QC793.2.T46::2013>.
- [8] Thomas Lenzi. “Development and Study of Different Muon Track Reconstruction Algorithms for the Level-1 Trigger for the CMS Muon Upgrade with GEM Detectors”. MA thesis. U. Brussels (main), 2013. arXiv: 1306.0858 [physics.ins-det]. URL: <https://inspirehep.net/record/1236817/files/arXiv:1306.0858.pdf>.
- [9] Joao Pequeno. Computer generated image of the ATLAS inner detector. 2008.
- [10] M. Backhaus. “The upgraded Pixel Detector of the ATLAS Experiment for Run 2 at the Large Hadron Collider”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 831 (2016). Proceedings of the 10th International “Hiroshima” Symposium on the Development and Application of Semiconductor Tracking Detectors, pp. 65–70. ISSN: 0168-9002. DOI:

- <https://doi.org/10.1016/j.nima.2016.05.018>. URL: <http://www.sciencedirect.com/science/article/pii/S0168900216303837>.
- [11] Daniel Adam Dobos, Claus Gossling, and Reiner Klingenberg. “Commissioning Perspectives for the ATLAS Pixel Detector”. Presented on 1 Sep 2007. PhD thesis. Dortmund: Dortmund U., 2007. URL: <https://cds.cern.ch/record/1092108>.
- [12] E. Hines. “Performance of Particle Identification with the ATLAS Transition Radiation Tracker”. In: *Particles and fields. Proceedings, Meeting of the Division of the American Physical Society, DPF 2011, Providence, USA, August 9-13, 2011*. 2011. arXiv: 1109.5925 [physics.ins-det]. URL: <https://inspirehep.net/record/929696/files/arXiv:1109.5925.pdf>.
- [13] G. Aad et al. “The ATLAS Experiment at the CERN Large Hadron Collider”. In: *JINST* 3 (2008), S08003. DOI: 10.1088/1748-0221/3/08/S08003.
- [14] Giordano Cattani and the RPC group. “The Resistive Plate Chambers of the ATLAS experiment: performance studies”. In: *Journal of Physics: Conference Series* 280.1 (2011), p. 012001. URL: <http://stacks.iop.org/1742-6596/280/i=1/a=012001>.
- [15] G. Aad et al. “Commissioning of the ATLAS Muon Spectrometer with Cosmic Rays”. In: *Eur. Phys. J. C* 70 (2010), pp. 875–916. DOI: 10.1140/epjc/s10052-010-1415-2. arXiv: 1006.4384 [physics.ins-det].
- [16] S. Aefsky et al. “The optical alignment system of the ATLAS muon spectrometer end-caps”. In: *JINST* 3 (2008), P11005. DOI: 10.1088/1748-0221/3/11/P11005.
- [17] Theodoros Argyropoulos et al. “Cathode strip chambers in ATLAS: Installation, commissioning and in situ performance”. In: *IEEE Trans. Nucl. Sci.* 56 (2009), pp. 1568–1574. DOI: 10.1109/TNS.2009.2020861.
- [18] ATLAS Approved Plots: DAQ/HLT. URL: <https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ApprovedPlotsDAQ>.
- [19] Mark S. Neubauer. “A Fast Hardware Tracker for the ATLAS Trigger System”. In: *Particles and fields. Proceedings, Meeting of the Division of the American Physical Society, DPF 2011, Providence, USA, August 9-13, 2011*. 2011. arXiv: 1110.1910 [hep-ex]. URL: <http://inspirehep.net/record/931205/files/arXiv:1110.1910.pdf>.
- [20] Andreas Salzburger. “Track Reconstruction Challenges for FCC-(hh) and HL-LHC”. Connecting the Dots 2016, slides. URL: <https://indico.hephy.oeaw.ac.at/event/86/session/2/contribution/4/material/slides/0.pdf>.
- [21] “Athena The ATLAS Common Framework”. Developer Guide. URL: <http://www.lnf.infn.it/~gatti/MANUALI/AthenaDeveloperGuide-8.0.0-draft.pdf>.
- [22] Andreas Salzburger, Dietmar Kuh, and Markus Elsing. “Track Simulation and Reconstruction in the ATLAS experiment”. PhD thesis. Innsbruck U., 2008. URL: <https://cds.cern.ch/record/2224514>.
- [23] The ATLAS collaboration. “A neural network clustering algorithm for the ATLAS silicon pixel detector”. In: *Journal of Instrumentation* 9.09 (2014), P09009. URL: <http://stacks.iop.org/1748-0221/9/i=09/a=P09009>.

- [24] Jovan Mitrevski. “Preparing ATLAS reconstruction software for LHC’s Run 2”. In: *Journal of Physics: Conference Series* 664.7 (2015), p. 072034. URL: <http://stacks.iop.org/1742-6596/664/i=7/a=072034>.
- [25] R. Frühwirth. “Application of Kalman filtering to track and vertex fitting”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 262.2 (1987), pp. 444–450. ISSN: 0168-9002. DOI: [http://dx.doi.org/10.1016/0168-9002\(87\)90887-4](http://dx.doi.org/10.1016/0168-9002(87)90887-4). URL: <http://www.sciencedirect.com/science/article/pii/0168900287908874>.
- [26] P. V. C. Hough. “Machine Analysis Of Bubble Chamber Pictures”. In: *Proceedings, 2nd International Conference on High-Energy Accelerators and Instrumentation, HEACC 1959: CERN, Geneva, Switzerland, September 14-19, 1959*. Vol. C590914. 1959, pp. 554–558. URL: http://inspirehep.net/record/919922/files/HEACC59_598-602.pdf.
- [27] “The HL-LHC Project”. (Image). URL: <http://hilumilhc.web.cern.ch/about/hl-lhc-project>.
- [28] *ATLAS Phase-II Upgrade Scoping Document*. Tech. rep. CERN-LHCC-2015-020. LHCC-G-166. Geneva: CERN, 2015. URL: <https://cds.cern.ch/record/2055248>.
- [29] Noemi Calace. “ATLAS ITk Layout Design and Optimisation”. In: (2016). URL: <https://cds.cern.ch/record/2234847>.
- [30] Simon Viel et al. *Performance of Silicon Pixel Detectors at Small Track Incidence Angles for the ATLAS Inner Tracker Upgrade*. Tech. rep. ATL-INDET-PROC-2015-011. Geneva: CERN, 2015. URL: <https://cds.cern.ch/record/2065104>.
- [31] Mauricio Garcia-Sciveres. *RD53A Integrated Circuit Specifications*. Tech. rep. CERN-RD53-PUB-15-001. Geneva: CERN, 2015. URL: <https://cds.cern.ch/record/2113263>.
- [32] M Backhaus. “Characterization of the FE-I4B pixel readout chip production run for the ATLAS Insertable B-layer upgrade”. In: *Journal of Instrumentation* 8.03 (2013), p. C03013. URL: <http://stacks.iop.org/1748-0221/8/i=03/a=C03013>.
- [33] “Fluence distributions for ITk extended@4 layout”. ATLAS Approved Plots. URL: <https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PLOTS/ITK-2016-002/>.
- [34] Mauricio Garcia-Sciveres. “Results of FE65-P2 Pixel Readout Test Chip for High Luminosity LHC Upgrades”. In: (2016). URL: <http://cds.cern.ch/record/2231609>.
- [35] Sioni Summers. “Online track reconstruction using Kalman Filters on FPGAs”. In: *Connecting the Dots and Workshop on Intelligent Trackers*. 2017.
- [36] Giuseppe Cerati et al. “Kalman filter tracking on parallel architectures”. In: *22nd International Conference on Computing in High Energy and Nuclear Physics (CHEP 2016) San Francisco, CA, October 14-16, 2016*. 2017. arXiv: 1702.06359. URL: <https://inspirehep.net/record/1514547/files/arXiv:1702.06359.pdf>.
- [37] Karlheinz Meier. “Brain inspired computimng in the EU Human Brain Project”. Discussion. 2017.
- [38] Bartosz Paprocki and Janusz Szczepanski. “Efficiency of neural transmission as a function of synaptic noise, threshold, and source characteristics”. In: *Biosystems* 105.1 (2011), pp. 62–72. ISSN: 0303-2647. DOI: <https://doi.org/10.1016/j.biosystems>.

- 2011.03.005. URL: <http://www.sciencedirect.com/science/article/pii/S030326471100058X>.
- [39] Dmitri B. Chklovskii, Thomas Schikorski, and Charles F. Stevens. “Wiring Optimization in Cortical Circuits”. In: *Neuron* 34.3 (2002), pp. 341–347. ISSN: 0896-6273. DOI: [https://doi.org/10.1016/S0896-6273\(02\)00679-7](https://doi.org/10.1016/S0896-6273(02)00679-7). URL: <http://www.sciencedirect.com/science/article/pii/S0896627302006797>.
- [40] Larry Squire, ed. *Fundamental Neuroscience*. 4th ed. Academic Press, 2012.
- [41] Khan Academy. “Overview of neuron structure and function”. (Images).
- [42] A. L. Hodgkin and A. F. Huxley. “A quantitative description of membrane current and its application to conduction and excitation in nerve”. In: *The Journal of Physiology* 117.4 (1952), pp. 500–544. ISSN: 1469-7793. DOI: 10.1113/jphysiol.1952.sp004764. URL: <http://dx.doi.org/10.1113/jphysiol.1952.sp004764>.
- [43] Peter Diehl and Matthew Cook. “Unsupervised learning of digit recognition using spike-timing-dependent plasticity”. In: *Frontiers in Computational Neuroscience* 9 (2015), p. 99. ISSN: 1662-5188. DOI: 10.3389/fncom.2015.00099. URL: <http://journal.frontiersin.org/article/10.3389/fncom.2015.00099>.
- [44] Todd Hylton. “Systems of Neuromorphic Adaptive Plastic Scalable Electronics”. (Slides). URL: <https://aeromotores.files.wordpress.com/2011/03/darpa-synapse-program.ppt>.
- [45] Steven K. Esser et al. “Convolutional networks for fast, energy-efficient neuromorphic computing”. In: *Proceedings of the National Academy of Sciences* 113.41 (2016), pp. 11441–11446. DOI: 10.1073/pnas.1604850113. eprint: <http://www.pnas.org/content/113/41/11441.full.pdf>. URL: <http://www.pnas.org/content/113/41/11441.abstract>.
- [46] Rudolph Emil Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.
- [47] Jeroen Schouwenberg. “Kalman Filter Trained Look-Up Tables for the ATLAS Level-1 Topological Missing Transverse Energy Trigger”. MA thesis. Radboud University, 2014.
- [48] O. Welch and G. Bishop. *An introduction to the Kalman filter*. Tech. rep. TR 95-041. UNC-CH, 1995. URL: <http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html>.
- [49] Simon Hayik. *Kalman Filtering and Neural Networks*. Wiley, 2001. ISBN: 9780471369981.
- [50] Chris Eliasmith. *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological systems*. A Bradford Book, 2004. ISBN: 9780262050715.
- [51] A. Andreopoulos et al. “A low-power neurosynaptic implementation of Local Binary Patterns for texture analysis”. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016, pp. 4308–4316. DOI: 10.1109/IJCNN.2016.7727762.
- [52] Chetvorno. “Solid-state electronics band structure”. (Image). 2017. URL: https://commons.wikimedia.org/wiki/File:Solid_state_electronic_band_structure.svg.

- [53] “Basic Electrical, Electronics Insulators, Conductors and Semiconductors”. (Image). 2017. URL: <http://www.studyelectrical.com/2015/05/insulators-conductors-and.html>.
- [54] Leonardo Rossi et al. *Pixel Detectors*. Ed. by Alexander Chao et al. Particle Acceleration and Detection. Springer, 2006. ISBN: 9783540283324.
- [55] N. Dimu. “Instrumentation on silicon detectors: from properties characterization to applications”. Habilitation à diriger des recherches. Université Paris Sud - Paris XI, Oct. 2013. URL: <https://tel.archives-ouvertes.fr/tel-00872318>.
- [56] “Basic notions about solids”. (Image). URL: <http://www.porous-35.com/electrochemistry-semiconductors-3.html>.
- [57] TheNoise at English Wikipedia. “Pn-junction in thermal equilibrium”. (Image). URL: <https://commons.wikimedia.org/wiki/File:Pn-junction-equilibrium.png> (visited on 2007).
- [58] Tony Kuphaldt. “Lessons In Electric Circuits”. In: vol. III. 2001. Chap. 2. URL: https://www.ibiblio.org/kuphaldt/electricCircuits/Semi/SEMI_2.html.
- [59] C. Jacoboni et al. “A review of some charge transport properties of silicon”. In: *Solid-State Electronics* 20.2 (1977), pp. 77–89. ISSN: 0038-1101. DOI: [http://dx.doi.org/10.1016/0038-1101\(77\)90054-5](http://dx.doi.org/10.1016/0038-1101(77)90054-5). URL: <http://www.sciencedirect.com/science/article/pii/0038110177900545>.
- [60] Charles Sodini. “Microelectronic devices and circuits”. In: 2007. Chap. 3 - Semiconductor Physics II. URL: <http://web.mit.edu/6.012/>.
- [61] D. M. Caughey and R. E. Thomas. “Carrier mobilities in silicon empirically related to doping and field”. In: *Proceedings of the IEEE* 55.12 (1967), pp. 2192–2193. ISSN: 0018-9219. DOI: [10.1109/PROC.1967.6123](https://doi.org/10.1109/PROC.1967.6123).
- [62] A. S. Grove. *Physics and technology of semiconductor devices*. John Wiley & Sons, 1967. ISBN: 9780471329983.
- [63] Warwick University. *Semiconductors: The Ideal Nonrectifying Barrier*. URL: <https://www2.warwick.ac.uk/fac/sci/physics/current/postgraduate/regs/mpags/ex5/devices/hetrojunction/ohmic/> (visited on 2011).
- [64] R.H. Richter et al. “Strip detector design for ATLAS and HERA-B using two-dimensional device simulation”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 377.2 (1996), pp. 412–421. ISSN: 0168-9002. DOI: [http://dx.doi.org/10.1016/0168-9002\(96\)00257-4](http://dx.doi.org/10.1016/0168-9002(96)00257-4). URL: <http://www.sciencedirect.com/science/article/pii/0168900296002574>.
- [65] G. Pellegrini et al. “Technology development of p-type microstrip detectors with radiation hard p-spray isolation”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 566.2 (2006), pp. 360–365. ISSN: 0168-9002. DOI: <http://dx.doi.org/10.1016/j.nima.2006.07.005>. URL: <http://www.sciencedirect.com/science/article/pii/S0168900206012587>.
- [66] Frank Hartmann. “Evolution of Silicon Sensor Technology in Particle Physics”. In: *Springer Tracts Mod. Phys.* 231 (2009), pp. 1–204.

- [67] T. Rohe et al. “Sensor design for the ATLAS-pixel detector”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 409.1–3 (1998), pp. 224–228. ISSN: 0168-9002. DOI: [http://dx.doi.org/10.1016/S0168-9002\(97\)01266-7](http://dx.doi.org/10.1016/S0168-9002(97)01266-7). URL: <http://www.sciencedirect.com/science/article/pii/S0168900297012667>.
- [68] A. La Rosa et al. “Novel silicon n-in-p pixel sensors for the future ATLAS upgrades”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 718 (2013). Proceedings of the 12th Pisa Meeting on Advanced Detectors La Biodola, Isola d’Elba, Italy, May 20–26, 2012, pp. 329–330. ISSN: 0168-9002. DOI: <http://dx.doi.org/10.1016/j.nima.2012.10.091>. URL: <http://www.sciencedirect.com/science/article/pii/S0168900212012624>.
- [69] C. Patrignani et al. “Review of Particle Physics”. In: *Chin. Phys.* C40.10 (2016), p. 100001. DOI: 10.1088/1674-1137/40/10/100001.
- [70] Hans Bichsel. “A method to improve tracking and particle identification in TPCs and silicon detectors”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 562.1 (2006), pp. 154–197. ISSN: 0168-9002. DOI: <http://doi.org/10.1016/j.nima.2006.03.009>.
- [71] Hans Bichsel. “Straggling in thin silicon detectors”. In: *Rev. Mod. Phys.* 60 (3 1988), pp. 663–699. DOI: 10.1103/RevModPhys.60.663. URL: <https://link.aps.org/doi/10.1103/RevModPhys.60.663>.
- [72] Richard Talman. “On the statistics of particle identification using ionization”. In: *Nuclear Instruments and Methods* 159.1 (1979), pp. 189–211. ISSN: 0029-554X. DOI: [http://dx.doi.org/10.1016/0029-554X\(79\)90348-3](http://dx.doi.org/10.1016/0029-554X(79)90348-3). URL: <http://www.sciencedirect.com/science/article/pii/0029554X79903483>.
- [73] “X-ray data booklet”. In: ed. by Albert Thompson and Douglas Vaughan. Lawrence Berkeley National Lab., 2009. Chap. 1.
- [74] Hans Bichsel. “Straggling of Heavy Charged Particles: Comparison of Born Hydrogenic-Wave-Function Approximation with Free-Electron Approximation”. In: *Phys. Rev. B* 1 (7 1970), pp. 2854–2862. DOI: 10.1103/PhysRevB.1.2854. URL: <https://link.aps.org/doi/10.1103/PhysRevB.1.2854>.
- [75] I. Peric et al. “The FEI3 readout chip for the ATLAS pixel detector”. In: *Nucl. Instrum. Meth.* A565 (2006), pp. 178–187. DOI: 10.1016/j.nima.2006.05.032.
- [76] George Ausman and Flynn Mclean. *Electron-hole Pair-creation energy in SiO2*. Tech. rep. HDL-TR-1720. NTIS, 1975.
- [77] Claude Leroy and Pier-Giorgio Rancoita. “Particle interaction and displacement damage in silicon devices operated in radiation environments”. In: *Reports on Progress in Physics* 70.4 (2007), p. 493. URL: <http://stacks.iop.org/0034-4885/70/i=4/a=R01>.
- [78] Michael Moll. “Radiation damage in silicon particle detectors: Microscopic defects and macroscopic properties”. PhD thesis. Hamburg U., 1999. URL: <http://www-library.desy.de/cgi-bin/showprep.pl?desy-thesis99-040>.
- [79] Z. Li and H. W. Kraner. “Studies of the dependence on oxidation thermal processes of effects on the electrical properties of silicon detectors by fast neutron radiation”. In:

- Conference Record of the 1991 IEEE Nuclear Science Symposium and Medical Imaging Conference*. 1991, 239–245 vol.1. DOI: 10.1109/NSSMIC.1991.258884.
- [80] Michael Moll. “Development of Radiation Hard Sensors for very High Luminosity Colliders”. Slides. 2002. URL: <http://www.phys.hawaii.edu/vertex2002/vprivate/Moll.pdf>.
 - [81] Magdalena Ciurea. “Defects in silicon: From bulk crystals to nanostructures”. In: *Romanian Reports in Physics* 60.3 (2008), pp. 735–478.
 - [82] C. A. Londos et al. “Vacancy-oxygen defects in silicon: the impact of isovalent doping”. In: *Journal of Materials Science: Materials in Electronics* 25.6 (2014), pp. 2395–2410. ISSN: 1573-482X. DOI: 10.1007/s10854-014-1947-6. URL: <http://dx.doi.org/10.1007/s10854-014-1947-6>.
 - [83] J. Weingarten. “ATLAS IBL sensor qualification”. In: *JINST* 7 (2012), p. C01039. DOI: 10.1088/1748-0221/7/01/C01039.
 - [84] Stefano Terzo et al. “Thin n-in-p planar pixel sensors and active edge sensors for the ATLAS upgrade at HL-LHC”. In: *JINST* 9.12 (2014), p. C12029. DOI: 10.1088/1748-0221/9/12/C12029. arXiv: 1409.8579 [physics.ins-det].
 - [85] V. Chiochia et al. “A double junction model of irradiated silicon pixel sensors for {LHC}”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 568.1 (2006). New Developments in Radiation Detectors Proceedings of the 10th European Symposium on Semiconductor Detectors, pp. 51–55. ISSN: 0168-9002. DOI: <http://doi.org/10.1016/j.nima.2006.05.199>. URL: <http://www.sciencedirect.com/science/article/pii/S0168900206010850>.
 - [86] Mathieu Benoit et al. *Modeling Radiation Damage Effects for Pixel Sensors in the ATLAS Detector*. Tech. rep. ATL-COM-INDET-2017-011. Geneva: CERN, 2017. URL: <https://cds.cern.ch/record/2255825>.
 - [87] B Henrich and R Kaufmann. “Lorentz-angle in irradiated silicon”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 477.1–3 (2002). 5th Int. Conf. on Position-Sensitive Detectors, pp. 304–307. ISSN: 0168-9002. DOI: [https://doi.org/10.1016/S0168-9002\(01\)01865-4](https://doi.org/10.1016/S0168-9002(01)01865-4). URL: <http://www.sciencedirect.com/science/article/pii/S0168900201018654>.
 - [88] Tommaso Lari. *Lorentz Angle variation with electric field for ATLAS silicon detectors*. Tech. rep. ATL-INDET-2001-004. INFN, 2001. URL: <http://cds.cern.ch/record/684187/files/indet-2001-004.pdf>.
 - [89] M. Moll, E. Fretwurst, and G. Lindström. “Investigation on the improved radiation hardness of silicon detectors with high oxygen concentration”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 439.2–3 (2000), pp. 282–292. ISSN: 0168-9002. DOI: [https://doi.org/10.1016/S0168-9002\(99\)00842-6](https://doi.org/10.1016/S0168-9002(99)00842-6). URL: <http://www.sciencedirect.com/science/article/pii/S0168900299008426>.
 - [90] LHCb collaboration. “LHCb Data Processing Applications Framework”. Gaudi User Guide.

- [91] Zachary Marshall and the Atlas Collaboration. “Simulation of Pile-up in the ATLAS Experiment”. In: *Journal of Physics: Conference Series* 513.2 (2014), p. 022024. URL: <http://stacks.iop.org/1742-6596/513/i=2/a=022024>.
- [92] Helmuth Spieler. *Semiconductor Detector Systems*. Vol. v.12. Semiconductor Science and Technology. Oxford: Oxford University Press, 2005. ISBN: 9780198527848. URL: <http://www.oup.co.uk/isbn/0-19-852784-5>.
- [93] Simon Ramo. “Currents induced by electron motion”. In: *Proc. Ire.* 27 (1939), pp. 584–585. DOI: 10.1109/JRPROC.1939.228757.
- [94] W. Shockley. “Currents to Conductors Induced by a Moving Point Charge”. In: *Journal of Applied Physics* 9.635 (1938).
- [95] James Jeans. “The mathematical theory of electricity and magnetism”. In: 2nd ed. Cambridge University Press, 1908. Chap. 7, p. 160.
- [96] Helmuth Spieler. *Semiconductor detector systems*. Semiconductor Science and Technology. Oxford: Oxford Univ. Press, 2005. URL: <https://cds.cern.ch/record/1010490>.

Acknowledgements

I am grateful for the support of the Lawrence Berkeley National Laboratory ATLAS group, who have hosted me for three fantastic years. Thanks are also due to Stockholm University for hosting this joint programme and for being so flexible given the unique conditions under which this degree is being conducted.

I am incredibly fortunate to have been supervised by not one but three nurturing and compassionate people: Sam Silverstein, Sara Strandberg, and Maurice Garcia-Sciveres. Sam and Sara, despite the fact that we are separated by an entire ocean, your intuition with regards to my work and the practical organisation of my studies has never felt distant.

Thanks also to my day-to-day supervisor, Maurice Garcia-Sciveres, for not only providing me with fantastic opportunities but also for always being willing to sit down and brainstorm the details.

I expected that having an official ‘mentor’ in this programme would be a formality, but Marjorie Shapiro has been an absolute rock in this process and I am so grateful for the time we’ve spent together, however brief.

During my degree I have enjoyed extensive debugging sessions with Simone Pagan-Griso, Zach Marshall, and Simon Viel. Thank you for the time and genuine care you put into responding to my questions and following up our discussions.

Many thanks to Paolo Calafiura for giving me the freedom to explore TrueNorth and for supporting me throughout the project. And to Ben Nachman, thank you for being so quick to follow-up with my Allpix queries and so enthusiastic to follow-through with my suggestions. Finally, many thanks to Su Dong and Carsten Hast at SLAC for providing such wonderful support in the testbeams we’ve worked on together so far.

Working at LBNL has made me appreciate the benefits of working in a large collaboration. Every person here has taught me so much, not only about how to be a good scientist but how to be a good scientist *in a community*. With those I have worked with directly, thank you, and I look forward to working with the rest of you in the near future.

And to Timon, you ground me and shake me up every single day and I can’t imagine my life without you.