ATLAS Trigger: Preparations for Run II

Gorm Galster[*]

June 30, 2015

[*]CERN / The Niels Bohr Institute

Contents

List of Figures

List of Tables

Foreword

Experimental High Energy Physics is the study of the elementary particles that we believe are the essential constituents of matter and energy. The field as we know it has existed for about half a century but has been revolutionized more than a handful of times by theoretical, experimental and technological achievements.

The Standard Model (SM) of particle physics in its current form was finalized in mid-1970 with the experimental verification of the existence of quarks and serves today as the cornerstone of our theoretical understanding. Its description of nature has been thoroughly tested over a broad range of energies, and discoveries, such as the recent discovery of the Higgs boson, have only added credibility to it.

In order to experimentally test the SM – and to look for physics beyond – increasingly sophisticated particle accelerators and particle detectors have been built over the years. The Large Hadron Collider (LHC) is currently the largest and most powerful particle accelerator in the world, situated at the international research center CERN in Geneva. ATLAS is one of four major particle detector experiments situated around the LHC.

My work on ATLAS – and thus the focus of this thesis – revolves around various aspects of the trigger system. In high energy physics experiments, the trigger is the component of the detector system that quickly decides whether a particular collision event should be stored, based on information from the subdetectors. Because of the extremely high rate of collisions and comparatively low rate of signal events, the ATLAS trigger should both reduce the event rate and maximize the signal-to-noise ratio of the recorded data.

Introduction

In February 2013 the Large Hadron Collider (LHC) went in to its planned two year shutdown to allow upgrade and maintenance of the LHC and the detectors around the ring. This marked the end of a successful Run I of the LHC run schedule, which began in late 2009. Not only did it break energy records, first at $\sqrt{s} = 7\,\text{TeV}$ and later $\sqrt{s} = 8\,\text{TeV}$, but the data produced also led to the 2012 discovery of the Higgs Boson. More excitement can be expected as we wait to see what we can learn from Run II, with expected collision energies of $\sqrt{s} = 7\,\text{TeV}$ and much more data.

Between Run I and Run II numerous upgrades were made to the ATLAS detector. In particular, the Trigger and Data Acquisition (TDAQ) system underwent several modifications and upgrades.

The work described in this thesis covers two different projects on the ATLAS trigger carried out before and during the shutdown. Consequently, the thesis is divided in three parts:

Part I provides a contemporary description of high energy physics at ATLAS, covering the Standard Model, the LHC, and ATLAS, with special focus on ATLAS TDAQ ;

Part II covers the work on the upgrade of the Central Trigger Processor (CTP) that provides the Trigger Timing and Control (TTC) signals to all the detectors of ATLAS and makes the initial trigger decision ;

Part III covers the work on data preservation, providing a strategy for how ATLAS can successfully preserve sufficient knowledge of the trigger system, that data from past run periods remain usable even when the trigger system is replaced.

Part I serves as a common foundation for the two other parts. The ordering of Part II and Part III is different from the chronological order in which the two projects were carried out. The description of the trigger and its recent evolution provided in Part I and Part III is complimented by the discussion in Part III of the importance and challange of maintaining an operational knowledge of past trigger systems.

This thesis is part of a "4+4 Ph.D. program" at the University of Copenhagen, in which the Ph.D. is started before the Masters Degree is obtained. In the life cycle of an experiment, analysis and results come from data, and the data must come from the commissioning and construction of the experiment. As this work focuses on the beginning of this life cycle, much of it is technical by nature.

Earlier this month we, the ATLAS collaboration celebrated the first recorded data of the LHC Run II. This data was selected by the upgraded trigger system described here and stored in a manner so that it will remain usable in the future.

Part I

High Energy Physics at ATLAS

Chapter I.1

The Standard Model

I.1.1 Introduction

The Standard Model (SM) is the cornerstone of high energy physics and provides the general framework for understanding the nature of and interactions between what we believe to be the elementary constituents of matter and energy. Formally the SM is a renormalisable Quantum Field Theory (QFT) with an internal gauge symmetry group of $\mathrm{SU}(3) \times \mathrm{SU}(2) \times \mathrm{U}(1)$ – the $\mathrm{SU}(3)$ the strong interaction, the $\mathrm{SU}(2)$ of the weak interaction and the $\mathrm{U}(1)$ of the electromagnetic interaction. This section will describe the SM particles and interactions, followed by a discussions of some of the shortcomings that motivate many beyond the SM searches.

I.1.2 Fundamental Particles

In QFT, both matter and forces are particles described by the excitation of the respective quantum fields they represent. The matter particles of

the SM are *fermions* and come in two classes: *leptons* and *quarks*. There are six flavours of quarks and leptons and these are usually divided into three *generations* of *weak isospin* doublets. Weak isospin, T, is a quantum number relating to the *weak interaction* which – like normal spin – is associated with SU(2) symmetry. Since only one of the generators T_i (corresponding to the Pauli matrices) of SU(2) can be simultaneously diagonalised with weak isospin T, it is customary to only denote the third component, T_3. There are three up-type quarks (u, c, t) with isospin $T_3 = 1/2$ and three down-type quarks (d, s, b) with isospin $T_3 = -1/2$. Similarly, there are three charged leptons (e, μ, τ) with isospin $T_3 = -1/2$ and three neutral leptons, neutrinos, (ν_e, ν_μ, ν_τ) with isospin $T_3 = 1/2$. In the original formulation of the SM, neutrinos were assumed to be massless. Observation of neutrino oscillations indicate that neutrinos do have mass. The nature of neutrinos – if they are Dirac or Majorana particles – is not yet known. The mass, charge and spin of the matter particles are summarised in Figure I.1.1a. For each of the fermions there is an associated flavour quantum number – the six lepton flavours and six quark flavours – which are conserved during strong and electromagnetic interaction but violated by weak interaction. Weak isospin, T_3, is conserved in all interactions. Particles in different generations differ in mass and flavour number but their fundamental interactions are the same.

The particles responsible for any interaction between the matter particles are a set of *bosons* often referred to as *force carriers* or *gauge bosons*. There are 4 such force carriers of the SM: the photon (γ) responsible for electromagnetic interactions, the gluon (g) responsible for the strong in-

teraction and lastly the $W^{\pm}$ and Z responsible for the weak interaction. The force carriers of the SM and their most important properties can be seen in Figure I.1.1b.

The latest confirmed particle of the SM as we know it today is the Higgs boson, which is the excitation of the Higgs field that ensures the mass of the massive particles.

For each particle in the SM, there is a corresponding anti-particle with same mass but opposite quantum charges. Some particles, such as the photon and the Z boson, are invariant under charge conjugation which implies that they are their own anti-particle.

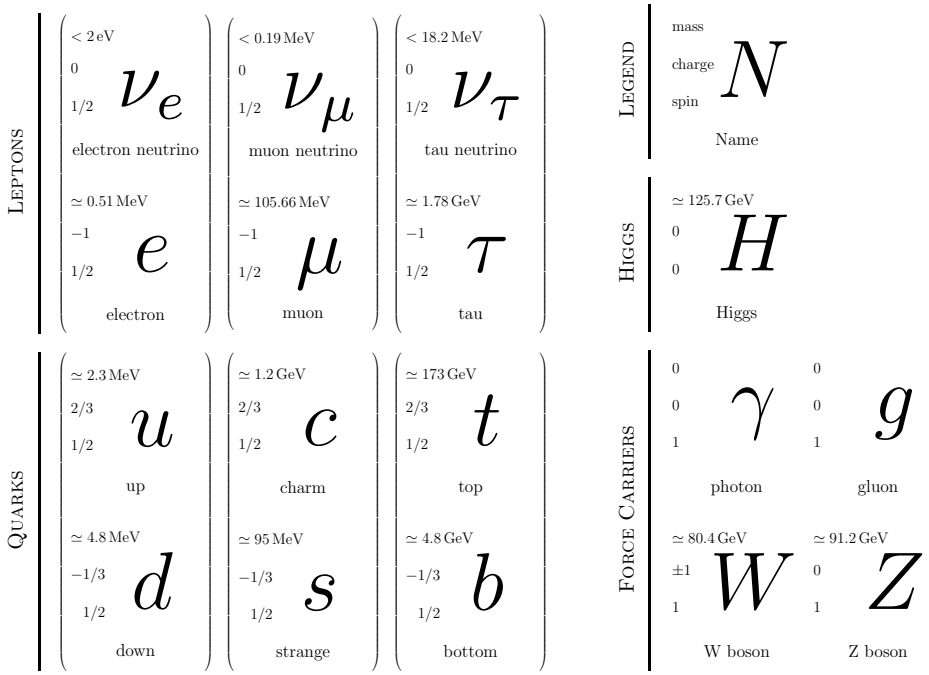
I.1.3 Particle Interactions

The Electroweak Interaction

Electroweak Theory (EWT) is the unification of the electromagnetic interaction and the weak interaction. The internal symmetry group of EWT is a $\mathrm{SU}(2) \times \mathrm{U}(1)$ gauge group. The generator of the $\mathrm{U}(1)$ group, is the *weak hypercharge*, Y_W, which is related to electrical charge, Q, and weak isospin, T_3, through:

$$Y_W = 2\left(Q - T_3\right), \tag{I.1.1}$$

with Q being measured in elementary charge. The generator of the $\mathrm{SU}(2)$ group is weak isospin. The four gauge bosons associated with the symmetry group – three from the $\mathrm{SU}(2)$ group (W^+, W^-, W^0) and one from the $\mathrm{U}(1)$ group (B^0) – are not those we observe: spontaneous symmetry



(a) Standard Model Fermions (b) Standard Model Bosons

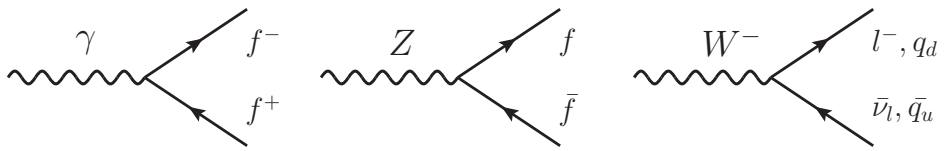
Figure I.1.1: Fundamental particles of the Standard Model, their mass, electric charge and spin. Numbers from [41].

breaking caused by the Higgs mechanism causes the W^0 and the B^0 to mix into the observed Z and the photon, γ:

$$\begin{bmatrix} \gamma \\ Z \end{bmatrix} = \begin{bmatrix} \cos\theta_{\mathrm{W}} & \sin\theta_{\mathrm{W}} \\ -\sin\theta_{\mathrm{W}} & \cos\theta_{\mathrm{W}} \end{bmatrix} \begin{bmatrix} B^0 \\ W^0 \end{bmatrix}, \tag{I.1.2}$$

where θ_{W} is the Weinberg angle, which relates the mass of the $W^\pm$ and the Z boson: $\cos\theta_{\mathrm{W}} = \frac{m_W}{m_Z}$.

The photon, the force carrier of the electromagnetic interaction, couples to all particles with electric charge. The two types of weak interactions, corresponding to the two types of bosons, are called the *Neutral*



(a) γ vertex (b) Z vertex (c) W vertex

Figure I.1.2: Fundamental interactions in EWT

Current for the Z boson and the *Charged Current* for the $W^{\pm}$. Contrary to the photon, they also couple to electrically neutral particles. The general form of these interactions are shown in Figure I.1.2.

The interaction state of the weak interaction is not the mass eigenstate, but rather a flavour eigenstate of the Lagrangian. A mass eigenstate of a given fermion can thus be described as a quantum superposition of flavour eigenstates. For quarks, the basis transformation matrix from flavour states to mass states is the Cabibbo–Kobayashi–Maskawa (CKM) matrix:

$$\begin{bmatrix} d' \\ s' \\ b' \end{bmatrix} = \begin{bmatrix} V_{ud} & V_{us} & V_{ub} \\ V_{cd} & V_{cs} & V_{cb} \\ V_{td} & V_{ts} & V_{tb} \end{bmatrix} \begin{bmatrix} d \\ s \\ b \end{bmatrix}, \tag{I.1.3}$$

where on the left hand side is the flavour states and on the right hand side the CKM matrix and the quark mass states.

Similarly, the Pontecorvo–Maki–Nakagawa–Sakata (PMNS) matrix describe the mixing of lepton mass and flavour states. Had neutrinos indeed been massless, the three corresponding mass states of the neutrinos would have been degenerate and neutrino oscillation – one neutrino

changing to another – would not have been observed. The observance of neutrino oscillations is the primary evidence of neutrino masses.

To complicate matters slightly, both the CKM and the PMNS matrix can be written in an alternative form parameterized by three mixing angles between the quarks (or leptons) of different generation. In addition to these mixing angles, there is one remaining parameter for both of the matrices: an overall complex phase. This phase, usually denoted δ_{CP}, causes *charge parity symmetry*, the symmetry of simultaneous charge and parity conjugation, is violated in weak interactions.

Further, the Z and $W^{\pm}$ bosons are massive, making the weak force short-ranged. The result is that particles decaying via the weak interaction, such as the $\pi^{\pm}$, have unusually long lifetimes

The Strong Force

The strong force is carried by the gluon (g) and couples to a quantum number called colour, carried by both quarks and the gluon itself. It is called colour as the conservation rules vaguely resembles those of additive colour mixing, and the governing theory is also often called Quantum Chromodynamics (QCD). Formally QCD has the symmetry group $SU(3)$ and thus eight gluons exist, each carrying a linear combination of colour and anti-colour. The colour charge comes in three variants: red, green and blue for quarks and correspondingly anti-red, anti-green and anti-blue for anti-quarks. In QCD only colour neutral bound states can exist freely – a phenomenon often referred to as *colour confinement*: when two quarks are separated enough, the energy of the gluon field would be

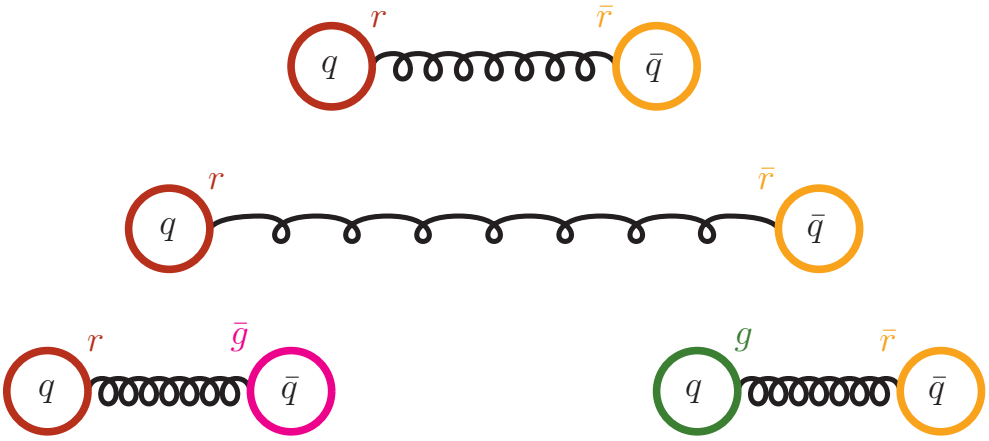
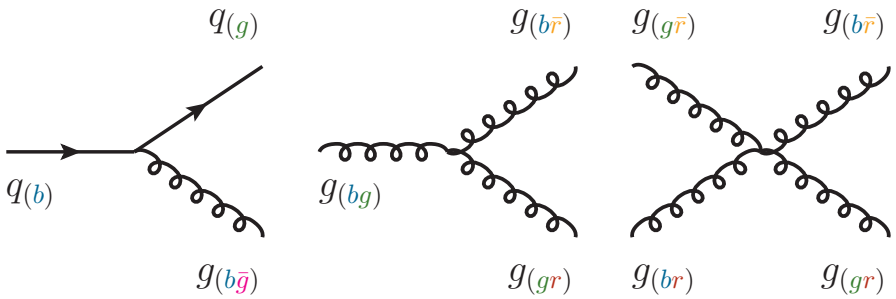


Figure I.1.3: Visualisation of color confinement



(a) qqg vertex (b) ggg vertex (c) $gggg$ vertex

Figure I.1.4: Fundamental interactions in QCD

enough to create a new quark-antiquark pair, as illustrated in Figure I.1.3. The best functional description is named “The Lund string model”, because of its analogy to the stretching of an elastic strings to the point where it snaps in two.

The representation of the gluon colour states are such that they are orthogonal to the colour singlet state, independent of their representation, which implies that colour neutral gluons cannot exist. Since the gluon

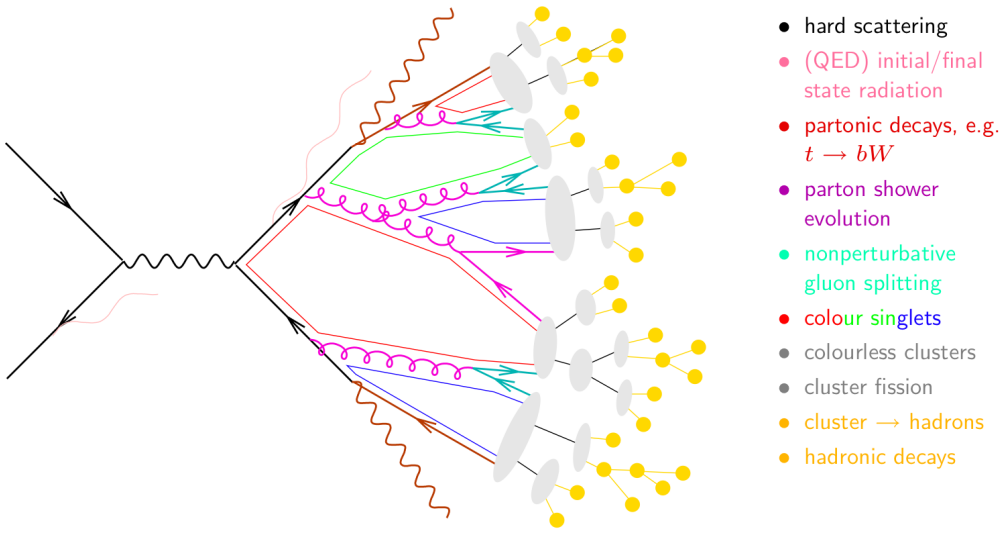


Figure I.1.5: Visualisation of $e^+e^- \to t\bar{t} \to b\bar{b}W^+W^-$. Source: [50]

carries colour, it too can radiate gluons. The fundamental interactions of the strong force are shown in Figure I.1.4.

In a sufficiently energetic process, gluon initiated quark pair production and subsequent gluon radiation can repeat itself multiple times before a colour neutral configuration is found, producing a spray or *jets* of particles. See Figure I.1.5 for visual aid. As will be discussed in more detail later, QCD jet production is the dominant process and thus often the main background at hadron colliders.

Composite Particles and a Sea of Quarks

Because colour confinement prohibits the existance of free quarks, they instead form colour neutral bound states of quarks. These can be either mesons, two-quark states such as the π^+, or baryons, three-quark states such as the proton. Mesons and baryons are both subsets of what we call

(a) $Q^2 = 1\,\mathrm{GeV}^2$ (b) $Q^2 = 10^4\,\mathrm{GeV}^2$

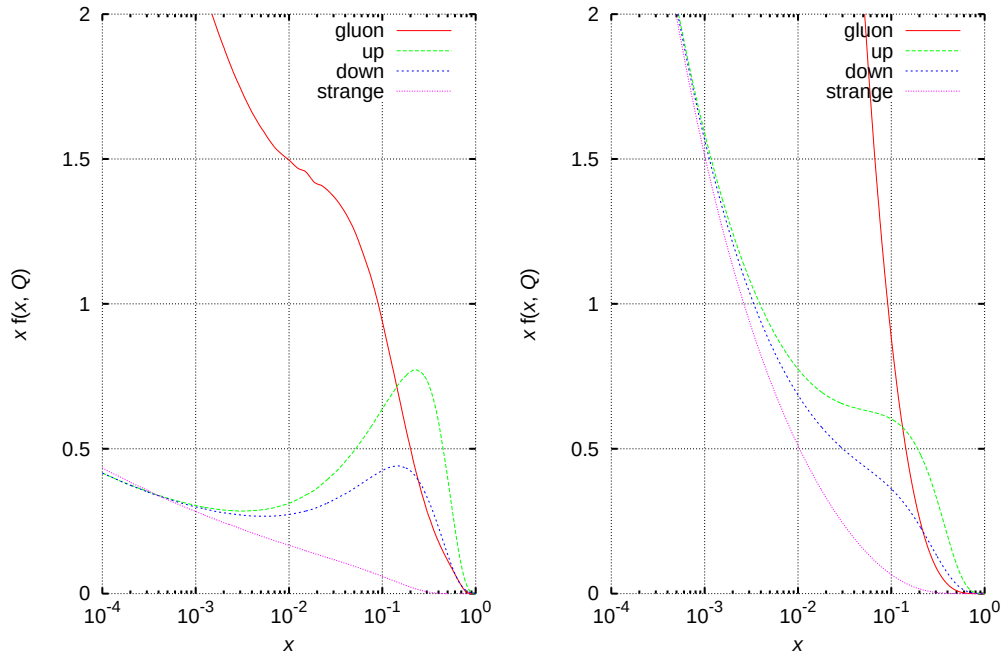


Figure I.1.6: CTEQ10 NNLO pdf for different values of Q^2

hadrons. Searches for bound states with more than three quarks have proved mostly [1] unsuccessful.

Besides the primary quark constituents, the so-called *valence quarks* contributing to the quantum numbers of the hadron, a hadron contains a number of gluons, binding the quarks together and keeping the hadron confined. At high energies both gluons and quarks might radiate additional gluons and the gluons in turn might split to a (virtual) quark-antiquark pair. At high energies, particularly in discussing (hard) scattering of hadrons, the contribution of this "sea" of gluons and virtual quarks

[1] The mass and quantum numbers of X(3872) doesn't fit this two or three pair quark model. One of the possible explanations are a 4 quark bounds state. [1]

becomes increasingly significant. A *parton distribution function* (pdf) gives the probability of finding a parton (quark or gluon) carrying a fraction x of the total momentum of the hadron at a certain energy resolution scale, Q^2. The energy resolution scale is the typical scale at which we observe the hard scattering. In beam-beam collisions this is typically the center of mass energy, $\sqrt{s} = 2\,E_{\text{beam}}$, and one obtains:

$$Q^2 = s = 4\,E_{\text{beam}}^2 \tag{I.1.4}$$

which for Run II of the LHC will mean a Q^2 of $\mathcal{O}\left(10^8\,\text{GeV}^2\right)$. Figure I.1.6 shows the `CTEQ10` [39] pdf at $Q^2 = 1\,\text{GeV}^2$ and $Q^2 = 10^4\,\text{GeV}^2$, corresponding roughly to rest mass of the proton and the electroweak energy scale, where the role of the sea of quarks and gluons becomes more pronounced. Considering first scatterings at roughly the binding energy of the proton, one observes that the naive picture of the quark composition of the quark holds: picking out a parton with one third of the momentum (or more), yields roughly a $2:1$ chance of finding a u quark over a d quark. In general, also at higher Q^2, the valence quarks are those easiest to pick out with a high momentum fraction, while at lower momentum fractions gluons and sea quarks dominate.

As Q^2 increases, the gluon contribution increases, and so does the contribution from the heavier sea quarks, which below the electro weak energy scale is negligible.

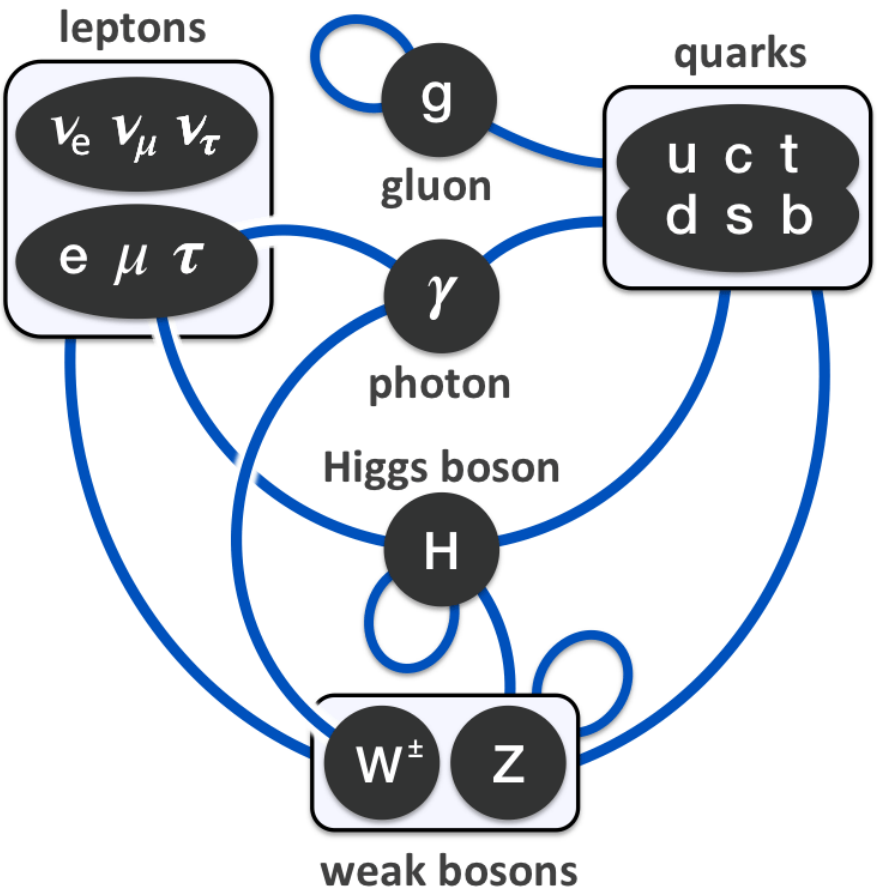


Figure I.1.7: Coupling of the Higgs and Force Carriers of the Standard Model. By Eric Drexler, released under CC0 1.0.

Table I.1.1: Summary of fundamental interactions of the Standard Model

	Fundamental Interaction			
Property	**Electromagnetic**	**Weak**	**Strong**	**Gravity**
Force Carrier	γ	$W^{\pm}$	g	Graviton[a]
Coupling Strength	$1/137$	10^{-6}	1	10^{-39}
Couples to	Electric Charge	Flavour	Colour	Mass

[a] Hypothesised but not observed.

Summary of Interactions

Figure I.1.7 depicts the coupling of the Higgs boson and the force carriers to the particles of the standard model. In Table I.1.1 a quick summary of the forces of the standard model is shown. Gravity is included here, though a both the Graviton and consistent theoretical description of quantum gravity is yet to be found. The extremely weak coupling of $\mathcal{O}\left(10^{-39}\right)$, can be estimated by comparing the magnitude of the electromagnetic force and the (classical) gravitational force between to fundamental particles. The low coupling strength is the reason why gravity can safely be ignored in particle physics.

I.1.4 Shortcomings

While the SM is our best working model it is important to understand that it has its limitations. More and more experimental evidence show

phenomena that is either wrongly described by the SM in its current formulation or not described at all. The following section serve to outline some of the most important shortcomings of the SM.

Neutrinos

By now there is plenty of evidence in support of non-zero neutrino masses [4][6][2]. The inference is based on (observed) neutrino oscillation where flavour and mass eigenstates mix, following a similar scheme as the quark mixing in I.1.3. If the neutrinos were indeed massless, the mass eigenstates would have been degenerate and no oscillation would be observed. There is no exact symmetry that forbids massive neutrinos in the SM, but the mechanism by which massive neutrinos should be introduced depends highly on the deeper nature of the neutrino which, despite active research in the area, remains a mystery. This, in part, as most theoretical models treating massive neutrinos, predicts the existence of one or more inert neutrino-like particles, or particles at mass ranges we have yet been unable to probe[29].

CP asymmetry

The SM has almost perfect symmetry between matter and anti-matter, yet the universe predominantly consists of matter. CP symmetry, implies that the laws of physics should be the same under simultaneous charge conjugation and parity conjugation. While the SM allows for some asymmetry, it is nowhere near enough to explain the observed asymmetry of the universe.

Following the observation of the anomaly of the neutral Kaon decay, CP violation in weak interactions was introduced as a complex phase of the CKM matrix. Similarly in the lepton sector, a complex phase can be ascribed to the PMNS matrix.

Another place where the SM allows for CP violation is in QCD, where an additional term in the Lagrangian, compliant with the internal symmetry group can be added [42]:

$$\mathcal{L}_{\text{QCD,CP}} = \bar{\theta}\,\frac{\alpha_{\text{S}}}{8\,\pi}\,F_a^{\mu\nu}\,\tilde{F}_{a\mu\nu}\,. \tag{I.1.5}$$

The term is a contraction of two terms and for non-zero values of the (pseudo) coupling, $\bar{\theta}$, the term is CP violating. Experimental constraints can be set on the size of $\bar{\theta}$: updating the results of [41] with results from [13] one obtain

$$\bar{\theta} < 10^{-10}\,. \tag{I.1.6}$$

There is no natural explanation for why this value is so small and not of the order of unity. This is a so called *fine tuning* problem often referred to as the *Strong CP Problem*.

Gravity

Numerous attempts has been made to include gravity in the SM but unfortunately it is far from trivial to include. To account for the space-time metric, the graviton would have to be a spin-2 boson, and because it has infinite range, it must be massless. Unfortunately both direct and indirect

searches for gravitons do not seem experimentally possible because of the extremely weak interaction with matter and the requirements it imposes [44].

Even though gravity is weak and negligible in particle physics, understanding gravity at Planck scale is paramount for understanding black holes or the early conditions in our universe.

Dark Matter and Energy

Dark Matter (DM) was introduced in astrophysics to account for observed gravitational effects related to structure formation, that can not be explained by any particles we know today. These include the galaxy rotation problem[17][43], gravitational lensing[30] and colliding galaxy clusters[43].

From the little we know about DM, we know that it is massive and experimental limits seem to suggest a very low interaction probability with ordinary matter.

As the SM does not have any inert particles and since the SM does not cover gravity, we are left with very little understanding of the nature of dark matter. If *Sterile neutrinos* exist they might be a DM candidate, and if (some) DM is *not completely* inert, but able to interact via a mechanism, that can be no stronger than the weak interaction [2], Weakly Interacting Massive Particles (WIMPs) might be a candidate as well.

The SM can account for $\mathcal{O}(5\,\%)$ of the energy density of the universe, and DM is believed to account for $\mathcal{O}(26\,\%)$. The remainder is unaccounted

[2]To avoid total self-annihilation in the early universe [37]

for, but is needed to explain the accelerating expansion of our universe[5]. The missing contributor to the energy density of the universe is often referred to as Dark Energy (DE) – the nature of which, if it exist, is as elusive as DM. Other theoretical models works with a modification of gravity at cosmological scales, explaining in part or in full the accelerated expansion.

Fine Tuning and Hierarchy

While the SM accurately describes elementary particles, there are several parts of its structure that are not well understood, including

a) things lacking explanation – such as why the electrical charge of the quarks is $1/3\,e$ (or conversely $2/3\,e$)

b) variations by orders of magnitude – such as the difference in strength between the weak interaction and gravity $\mathcal{O}\left(10^{32}\right)$, masses of fermions $\mathcal{O}\left(10^{5}\right)$

c) the need for extremely fine tuning between parameters to cancel each other out exactly in the right way that experimental data can be explained within the context of the SM. One such example is the corrections to the Higgs mass that are magnitudes larger than the Higgs mass itself.

Free Parameters

The SM has a number of free variables, for which arbitrary values will yield a consistent theory, but whose values greatly affect the physics. These

parameters are:

- 3 gauge couplings – one for each of the three gauge symmetries: $\mathrm{U}(1)$, $\mathrm{SU}(2)$ and $\mathrm{SU}(3)$;
- The coupling constant for the Lagrangian term in (I.1.5)
- The Weinberg angle, θ_{W} – that yields the photon and the Z and the γ in electroweak theory;
- 3 mixing angles in the CKM matrix – for the mixing of flavour and mass eigenstates of quarks;
- 1 complex phase of the CKM matrix – the one responsible for the weak CP violation;
- Correspondingly, 3 mixing angles and one CP violating complex phase in the PMNS matrix – for the mixing of flavour and mass eigenstates of leptons;
- 12 Yukawa coupling constants – to give mass to each of the 12 fermions of the SM;
- The vacuum expectation value of the Higgs field;
- The mass of the Higgs boson;

This yields a total of 26 free parameters, which must be determined experimentally. While most of these have been treated in this chapter, the three last bullet points, or correspondingly, 15 of the free parameters have not. The 15 parameters all relate to the Higgs mechanism, which have not

been covered here – see [34] for a fuller description. In brief, each of the 12 fermionic fields couples to the Higgs field with some coupling constant. Through the Higgs mechanism, via spontaneous symmetry breaking, the fermions acquire a mass proportional to the vacuum expectation value, v, of the Higgs field. Further, the Higgs bosons self-coupling is dependant on the mass of the Higgs, m_{H}, as well as the vacuum expectation value:

$$\lambda = \frac{3\, m_{\mathrm{H}}^2}{v}\,. \tag{I.1.7}$$

I.1.5 Searching for the Unknown

Most theories Beyond the Standard Model (BSM) aim at elegantly explaining one or more of the shortcomings outlined in this sections. These theories often predict the existence of one or more particles, many of which we can experimentally look for and study. Until recently the Higgs boson was such a hypothesised particle, the only new particle needed to explain why some particles have mass. The Large Hadron Collider was built in part to search for the Higgs in regions of phase space that had not already been ruled out by either theory or previous experiments – and we found it. More precisely, we found *something* that *seems to be* compatible with a SM Higgs boson. It might also be a Higgs-like particle as described by other theories, so we continue to study it, and the rest of the SM, and hopefully we get closer to understand the nature of this new particle, the other shortcomings of the SM, and the strange but beautiful universe we live in.

Chapter I.2

The Large Hadron Collider

The Large Hadron Collider[31] (LHC) is the most powerful particle collider in the world, located at CERN in Geneva, Switzerland. The LHC is the last acceleration step at the facility before the accelerated protons are collided at the four interaction points, where the detector experiments ATLAS, CMS, LHCb and ALICE study the collision debris.

I.2.1 LHC Run Schedule

Table I.2.1 summarises the LHC run schedule. The schedule alternates between periods of beam collisions, the run periods, and periods for maintenance and upgrades (the shutdown periods).

The goal of Run I was the discovery (or non-discovery) of the Higgs boson. Run II will allow us to study the Higgs boson in more detail – – and perhaps get a glimpse of physics beyond the SM. Should Run II yield a new discovery, Run III will provide us with even more data to explore this new

Time	State Name	Beam Energy
Q4, 2009 – Q1, 2013	Run I	3 TeV, 4 TeV
Q1, 2013 – Q2, 2015	LS I	–
Q2, 2015 – Q2, 2018	Run II	6.5 TeV, 7 TeV
Q2, 2018 – Q4, 2019	LS II	–
Q1, 2020 – Q4, 2022	Run III	7 TeV

LS - Long Shutdown

Table I.2.1: The LHC run schedule. Future dates from [28].

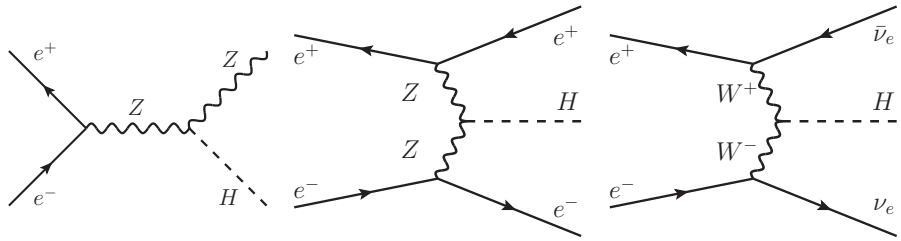
physics. Even if no new particles are discovered, the data of Run II and Run III combined, will still provide valuable input for our understanding of the SM.

The shutdown periods between the run periods allow the LHC and the experiments to do the necessary consolidation and maintainance necessary for the technically demanding conditions.

In Long Shutdown I the upgrades on the LHC included work on magnet alignment and upgrades of the radio frequency systems for increased power. For the experiments, and ATLAS in particular, the primary changes were to the trigger system, but will in the future also involve (partial) replacement of radiation damaged detectors.

I.2.2 On Colliding Hadrons

As any charged particles can be accelerated, it might be worth to quickly go through the conceptual differences between lepton colliders and hadron



(a) Higgs Strahlung

(b) Vector Boson Fusion

Figure I.2.1: Primary diagrams for Higgs production at e^+e^-colliders.

colliders.

Lepton colliders – historically electron-positron colliders – have several advantages but also disadvantages. In a e^+e^-collider the fundamental interaction is the annihilation of a pair of leptons, as in the typical process:

$$e^+ e^- \to \mu^+ \mu^- \,. \tag{I.2.1}$$

This leaves the entire center-of-mass energy of the beam particles available for particle production. The variation in energy available for particle production from collision to collision is effectively limited to the variation in beam energy, making it suitable for production of particles in a narrow mass range. While this is generally true, it should be noted that some particles, like the Higgs boson, are difficult to produce directly at an e^+e^-collider. The Higgs coupling to the electron is small and the production channel $e^+ e^- \to H$ is suppressed by so-called "Higgs strahlung" (Figure I.2.1a), and by vector boson fusion processes (Figure I.2.1b). In these, and similar, processes where the desired particle is not produced directly, each of the resulting particles will carry a fraction of the total energy, resulting in a spread of production energies.

In hadron colliders, since protons are compound particles with a complex composition at high energy, the picture is more complicated. In a deep in-elastic collision of protons, the main particle producing process is between two of the proton constituents – partons carrying a variable fraction of the proton beam momentum. As the primary interaction and particle producing processes are between coloured objects, QCD processes dominate. What is left of the proton in the case of hard scattering, will undergo further strong interactions, leading to QCD jets in the direction of the beam. Because of the complicated nature of the proton structure, the energy available for particle production varies greatly from collision to collision, from the entire CM energy all the way down to zero, as illustrated by the pdf in Figure I.1.6.

In circular colliders, synchrotron radiation loss is proportional to m^{-4}, which limits the acheivable beam energies for circular e^+e^--colliders. For protons, much higher energies can easily be achieved, the limiting factor being the field strength of the dipole magnets.

I.2.3 From Hydrogen to Higgses

Accelerating Steps

CERNs accelerator facility is larger than just the LHC. A depiction of the full accelerator complex is shown in Figure I.2.2. The LHC is the $5^{\underline{\text{th}}}$ and last acceleration step before protons are brought to collide.

It all starts in LINAC2 where hydrogen from a small gas bottle is first ionized then accelerated to $50\,\text{MeV}$. The beam from the LINAC2 feeds

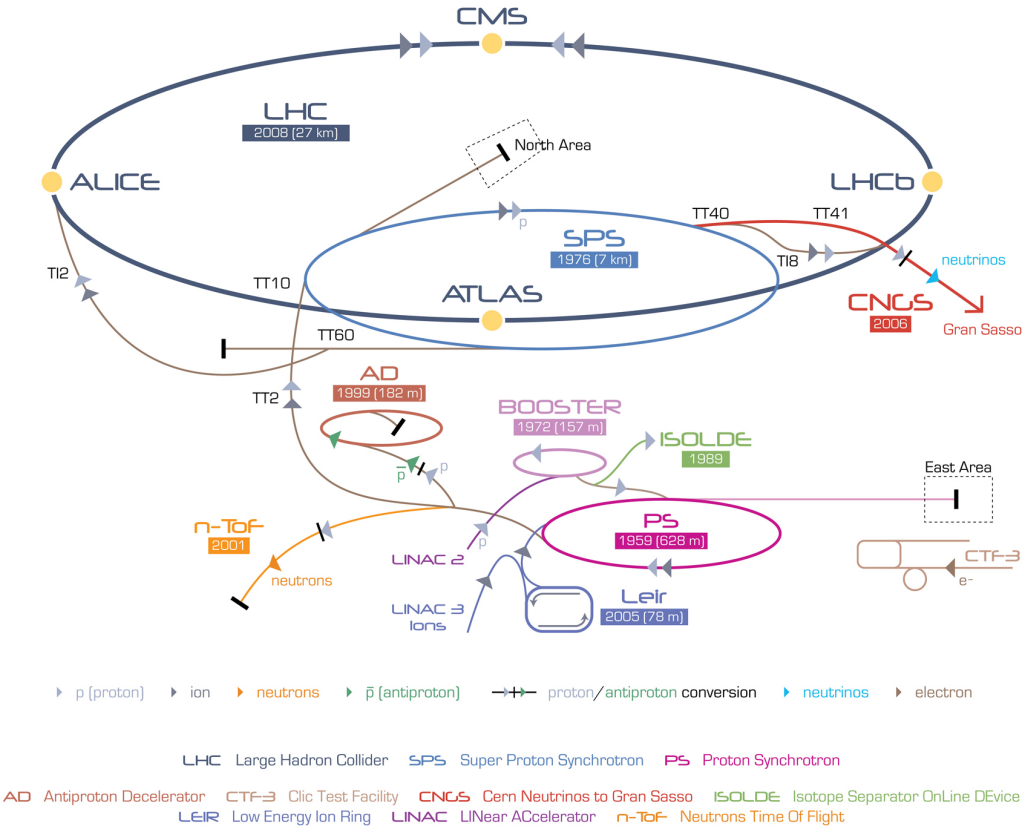


Figure I.2.2: CERN accelerator complex. Source: [21]

the Proton Synchrotron Booster, or *booster* for short. The beam is split into the four superimposed synchrotron rings of the booster where they are accelerated to $1.4\,\mathrm{GeV}$ before the protons are injected into the Proton Synchrotron (PS) as a *bunch* of $\mathcal{O}\left(10^{11}\right)$ protons. The PS accelerates the proton bunches to $25\,\mathrm{GeV}$ before injecting them into the Super Proton Synchrotron (SPS) which again accelerates the protons to a beam energy of $450\,\mathrm{GeV}$, before injecting them into the LHC.

Finally in the LHC the protons are accelerated to collision energies. During Run I beam energy of $4\,\mathrm{TeV}$ were reached and for the start of Run

II the beam energy will reach $6.5\,\text{TeV}$ before collision[18]. The LHC was designed for a maximum beam energy of $7\,\text{TeV}$.

Bunch Structure

The above description might give the idea that one bunch of protons gets filled into the LHC at the time, but the picture is slightly more complicated. The actual bunch structure, as well as bunch to bunch effects, is of importance to the trigger.

The idealised case is as follows: The PS is filled with 72 bunches with $25\,\text{ns}$ spacing. A series of bunches with fixed spacing is often referred to as a *bunch train* or simply a *train*. The rise time of the SPS injection kicker is $\mathcal{O}\,(220\,\text{ns})$ leading to a gap of 8 (empty) bunches, between the 72 filled bunches per PS injection. The SPS is filled with 3 or 4 fills from the PS before the SPS content is transfered to the LHC. The LHC receives 4 SPS fills with a variation in trains following the pattern:

$$334\ 334\ 334\ 333\ . \tag{I.2.2}$$

The rise time of the LHC injection kicker is $\mathcal{O}\,(940\,\text{ns})$ leading to a gap of 38 bunches when injecting 3 trains and a 39 bunch gap when injecting four trains. At the end of the filled bunches a gap of 119 empty bunches is reserved for the rise time of the dump kicker. With the gap from the dump kicker at the end, the pattern of (I.2.2) is approximately 4 fold symmetric, which is both practical and convenient: With four interaction points, it ensures the collision of all the bunches and in the same bunch crossing

each turn at each interaction point. Having all bunches collide further improves the stability of the beam.

Counting filled and empty bunches – here denoted e and f – one obtains for each SPS fill of three and four trains:

$$k_3 = 3\,\underbrace{(72\,f + 8\,e)}_{\text{PS fill}} + \underbrace{30\,e}_{38e-8e}\ ,\ \ k_4 = 4\ (72\,f + 8\,e) + 31\,e\,, \tag{I.2.3}$$

which yields a total number of bunches of

$$k_{\text{LHC}} = 9\,k_3 + 3\,k_4 + \overbrace{81\,e}^{119e-8e-30e} \tag{I.2.4}$$

$$= 2808\,f + 756\,e \tag{I.2.5}$$

$$= 3564 \tag{I.2.6}$$

In reality the fill pattern may vary, for instance when running with a bunch spacing of $50\,\text{ns}$ as in Run I, giving a different relative number of filled and empty bunches. The total number of bunches, 3564, remains constant, and is an important number as it is the length of the orbit in units of bunch crossings:

$$1\,\text{ORB} = 3564\,\text{BC} \tag{I.2.7}$$

The outlined scheme also defines the maximum number of colliding bunches achievable.

Cross Section and Luminosity

Consider the heads-on collision of two bunches with n protons in each and assume that the bunch can be approximated as a cylinder with length, l, and a 2D Gaussian profile with standard deviation, σ_x and σ_y. The probability, dP, of *one* proton from one of the bunches interacting with *any* of the protons in an incident bunch, is proportional to the density of protons in that other bunch, as well as the distance the proton traverses through the bunch:

$$dP \propto \frac{n}{V}\, l = \frac{n}{4\,\pi\,\sigma_x\,\sigma_y}\,. \tag{I.2.8}$$

The factor of proportionality is the *cross section*, also denoted by σ. The cross section is a measure of interaction probability for a certain set of beam parameters and thus, besides the beam parameters discussed here, depends on the process in question and the beam energy, see Figure I.2.5.

The total interaction probability will be n times higher, and as there are k colliding bunches colliding per turn at a revolution frequency of f, the instantaneous interaction rate for a given process will be:

$$\frac{dN_\sigma}{dt} = \mathcal{L}\,\sigma = \frac{f\,k\,n^2\,\sigma}{4\,\pi\,\sigma_x\,\sigma_y}\,, \tag{I.2.9}$$

where $\mathcal{L}$ is the *instantaneous luminosity*, the integral of which, $L = \int \mathcal{L}\,dt$, is called the *integrated luminosity*. The instantaneous luminosity is a measure of the rate of interactions observed to the interaction cross section $\mathcal{L} = \frac{1}{\sigma}\frac{dN}{dt}$, which is only dependant on the beam parameters. During a *fill* the instantaneous luminosity will drop *roughly* exponentially as the

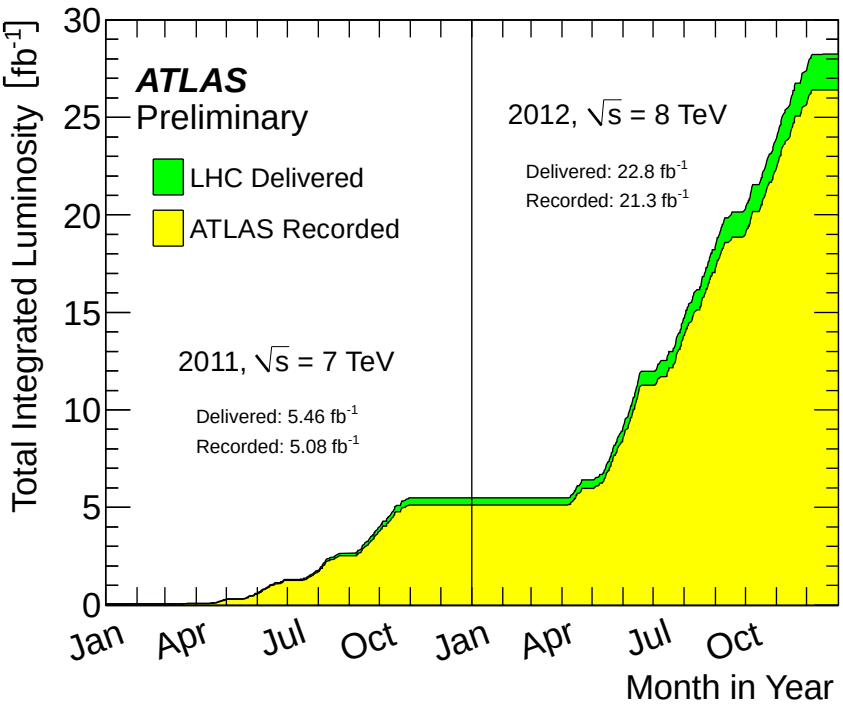


Figure I.2.3: The by LHC delivered and by ATLAS recorded integrated luminosity.

number of protons per bunch drops. Various beam instrumentation effects also lead to a loss of protons and monitoring of interaction rates and measurement of beam current per bunch is an important part of the experimental effort for the experiments.

The integrated luminosity is important as a measure of the number of produced interactions. The peak instantaneous luminosity achieved in Run I was $\sim 7.7 \cdot 10^{33}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ as measured by ATLAS (see Figure A.1) and the total delivered integrated luminosity during Run I was $5.46\,\mathrm{fb}^{-1}$ at $7\,\mathrm{TeV}$ and $22.8\,\mathrm{fb}^{-1}$ at $8\,\mathrm{TeV}$, as can be seen from Figure I.2.3.

The number of interactions per bunch crossing is called the (in-time) *pile-up*. During Run I, the mean number of interactions per bunch cross-

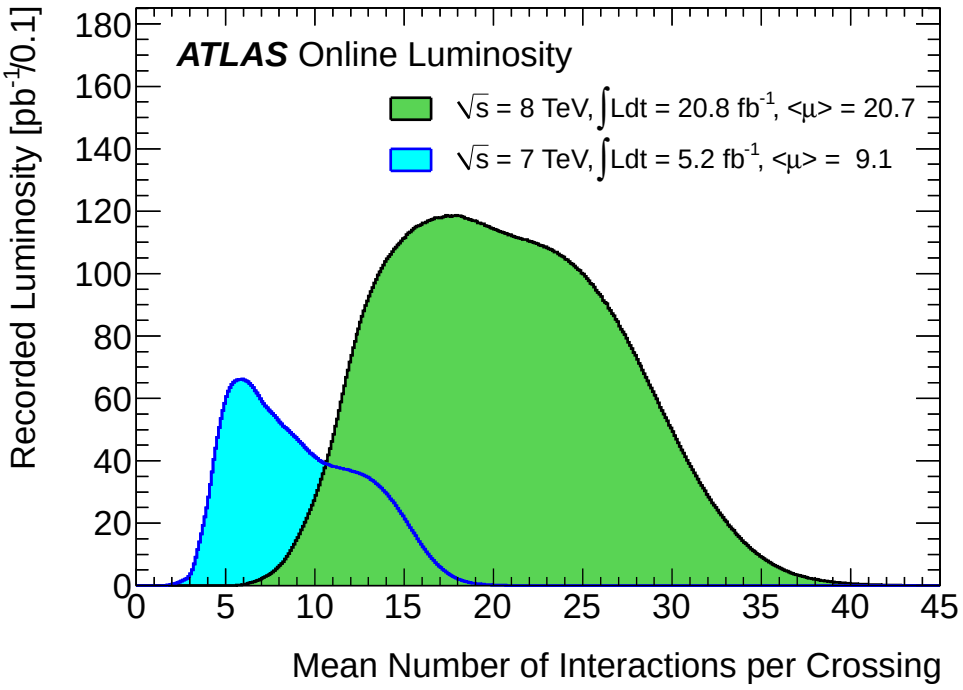


Figure I.2.4: Integrated luminosity as function of average pile-up

ing was $\langle\mu\rangle = 20.7$, ranging from $\langle\mu\rangle < 0.1$ to a peak value of $\langle\mu\rangle > 70$. Figure I.2.4 show the recorded luminosity as function of pile-up. Figure A.3 and Figure A.2 show the peak and average pile-up as function of time.

Figure I.2.5 show cross sections for different processes as function of beam energy. At the peak instantaneous luminosity, using the total cross-section, σ_{tot}, this comes to a total of $\sim$ 700 000 000 collision events per second. The 700 MHz of events is predominantly QCD events. Taking the Higgs production as an example, and using the total Higgs cross section from Figure A.5, one obtains a Higgs production frequency of merely 0.1 Hz. Further, the dominant decay modes of the Higgs – see Figure A.4

for details – makes the decay debris difficult to distinguish from the QCD background. The dominant mode of the Higgs, with a branching fraction of $\mathcal{O}(60\%)$, is the production of a b-quark pair, $H \to b\bar{b}$. This signal is to be compared to the cross section of primary $b\bar{b}$ production which is $\mathcal{O}(10^{10})$ higher. The (b-)jets are then to be separated from the remaining jet background. The inclusive search for $H \to b\bar{b}$ and other modes is not experimentally feasible at hadron colliders and can only effectively be studied, by requiring other observables, such as an associated W or Z with modes that can be used for background suppression and triggering.

The interaction rate cannot be matched by any earthly readout system, while the low signal-to-noise ratio means that stored data should be carefully selected. This implies the need for online data selection, which can both improve signal purity and reduce the event rate.

I.2.4 Onwards to Run II

Several things will change in Run II: bunch spacing will decrease, beam energy will decrease, and beam optics will decrease the size of the beam profile. Consequently, the inst. lum. will increase, leading to higher pileup and harsher experimental conditions.

Table I.2.2 summarises important operational parameters and features of the LHC. It is worth noting that the beam intensity (the number of protons per bunch) has already exceeded the design parameter. As a consequence, the instantaneous luminosity is expected to exceed its design value during Run II. The estimated values are subject to ongoing developments, but the rough reasoning behind this is that the luminosity

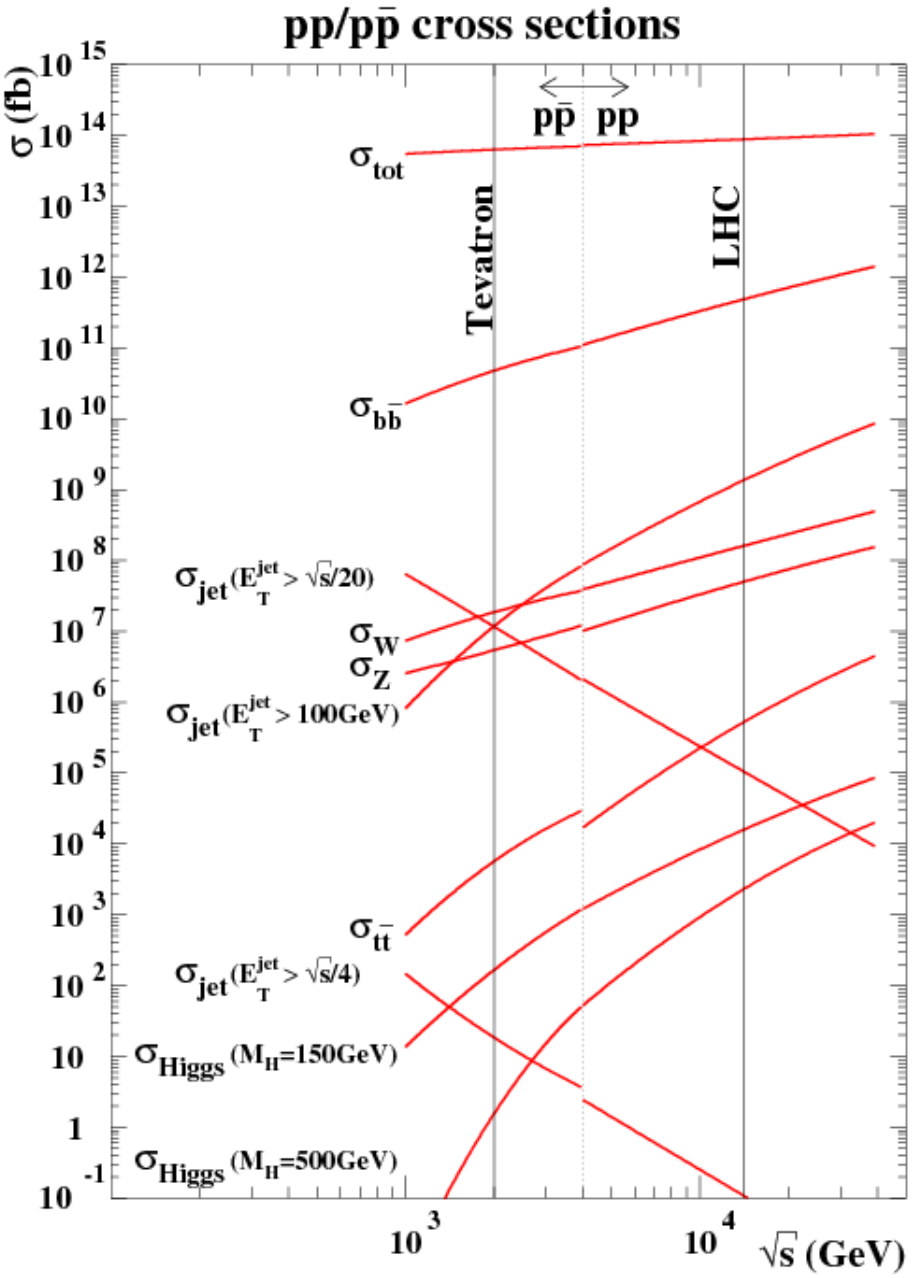


Figure I.2.5: Cross section for different processes as function of center of mass energy. Source: [49]

Parameter	**Run I**	**Run II**	**Design**
Collision Energy [TeV]	8	13	14
Bunch Spacing [ns]	50	25	25
Colliding bunches	1374	2520	2808
Protons per bunch	$1.6 - 1.7 \times 10^{11}$	1.3×10^{11}	1.15×10^{11}
Peak luminosity [$cm^{-1}s^{-1}$]	7.7×10^{33}	1.6×10^{34}	1×10^{34}

Table I.2.2: Operational parameters of the LHC at the end of Run I and estimates for Run II.

grows linearly with the number of bunches, but as the square of the intensity. Reducing the bunch spacing allows for more bunches, but with a high number of high(er than nominal) intensity bunches, various technical difficulties arise. One such might be the power of the radio frequency in the SPS.

The implications for ATLAS and the trigger, will be the topic of Chapter I.4.5.

Chapter I.3

The ATLAS Detector

ATLAS, short for *A Toroidal LHC ApparatuS*, is one of the four large particle physics detectors at the LHC. ATLAS consist of two magnet systems and a number of subdetectors that each extract various features from the particles passing through the detector. The partial information from each subdetector is later combined to form a full picture of what happened during a collision.

This chapter will provide a brief overview of the ATLAS detector. For a detailed description see [27] and [7] which also serve as reference throughout this chapter unless otherwise stated.

I.3.1 ATLAS Detector overview

Design Philosophy

A cut-away picture of the ATLAS detector is shown in Figure I.3.1. The detector has a barrel geometry and almost perfect forward-backwards sym-

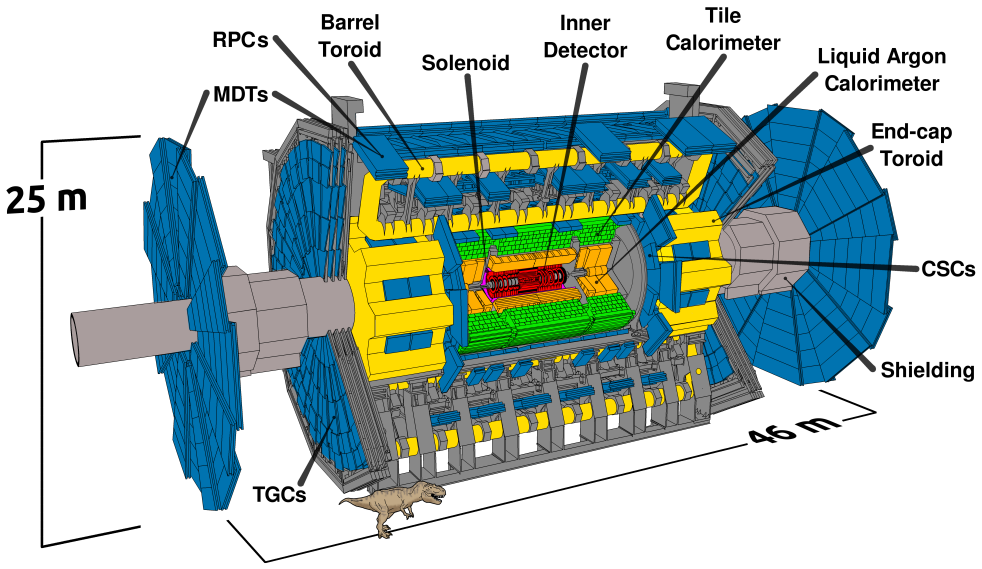


Figure I.3.1: Cut-away depiction of the ATLAS detector with a T-Rex for scale. Source: [35]

metry. In the beam collisions, a variety of particles are produced with a broad range of energies. ATLAS is designed as a *general purpose detector*, able to detect and measure properties of any (new) physics processes that might take place. The challenge at the LHC is the high energies and the high rate of collisions, which requires ATLAS to be larger and more complex than previous build detectors. ATLAS is optimised for reconstruction of high energy charged leptons (here electrons and muons), as the detector signature is clean and easy to distinguish from the QCD background. Consequently, it has precise electromagnetic calorimeter and a huge and elaborate muon system.

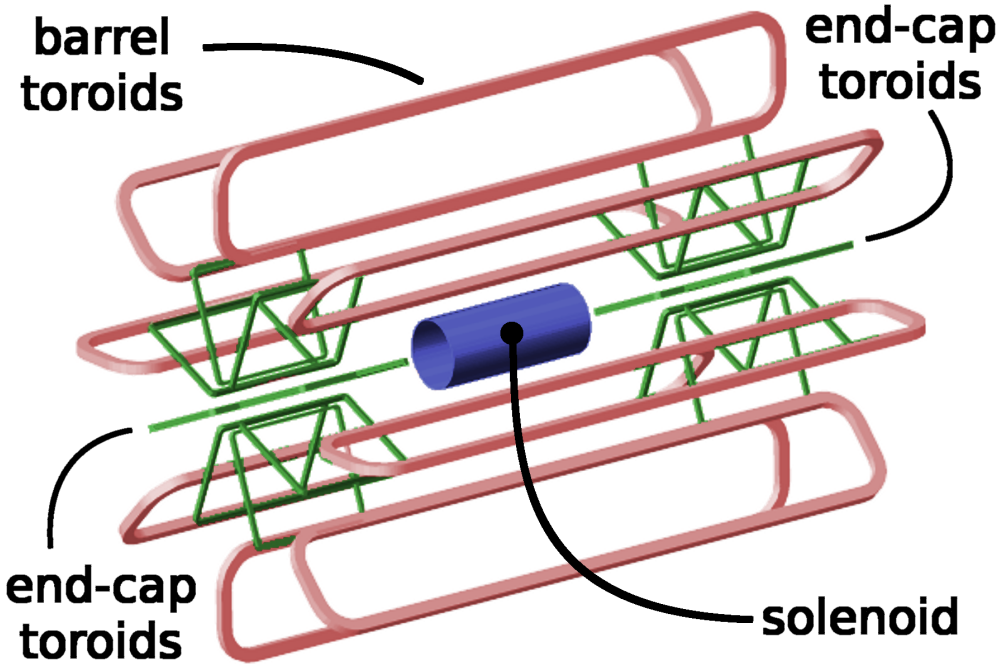


Figure I.3.2: Depiction of the ATLAS magnet coils. Source: [35]

ATLAS Magnet System

The role of ATLAS magnet system, is to bend the trajectory of the charged particles. A curved trajectory reveals information about the particles momentum and also about the sign of the electrical charge. The magnet coils of ATLAS are depicted in Figure I.3.2. A central solenoid in the inner part of ATLAS produces a $2\,\mathrm{T}$ field parallel to the beam line, bending the trajectory of charged particles in the plane transverse to the beam line. A toroidal field outside the calorimeter region, produced by the barrel toroid and two end-cap toroid magnets, bends the trajectory of charged particles, primarily muons, in the direction of the beam line. The strength of the toroid field varies between $0.5\,\mathrm{T}$ and $1\,\mathrm{T}$ in the barrel region up to $2\,\mathrm{T}$ in the end-caps.

The entire magnet system measures 26 m in length and 20 m in diameter.

Particle Detection in ATLAS

Particle detection in ATLAS is done by combining the partial information provided by the different sub-detectors into more sophisticated particle hypothesis. e.g., a particle that deposits all its energy in the electromagnetic calorimeter is electromagnetic by nature. By combining this with the tracking information one can distinguish between a photon (no track), an electron (calorimeter deposit and associated track) or a positron (same as electron but opposite curvature). Figure I.3.3 show some typical signatures of particles in ATLAS.

The only SM particle that can not be observed directly is the neutrino. Instead, a quantity called *missing energy* is used, to describe the residue energy which can be inferred from conservation of energy but which escapes detection. In proton-proton collisions however, the total energy can not be obtained, as the momentum (transfer) of the interacting partons along the beam direction is not known from collision to collision. However, in the plane transverse to the beam, the initial momentum is zero. Consequently, the *missing transverse energy* (MET) – usually denoted, $\not{E}_{\mathrm{T}}$ – is used.

Particle Reconstruction in ATLAS

The event reconstruction of data in ATLAS uses *physics objects* to describe and classify the observed final states: These are:

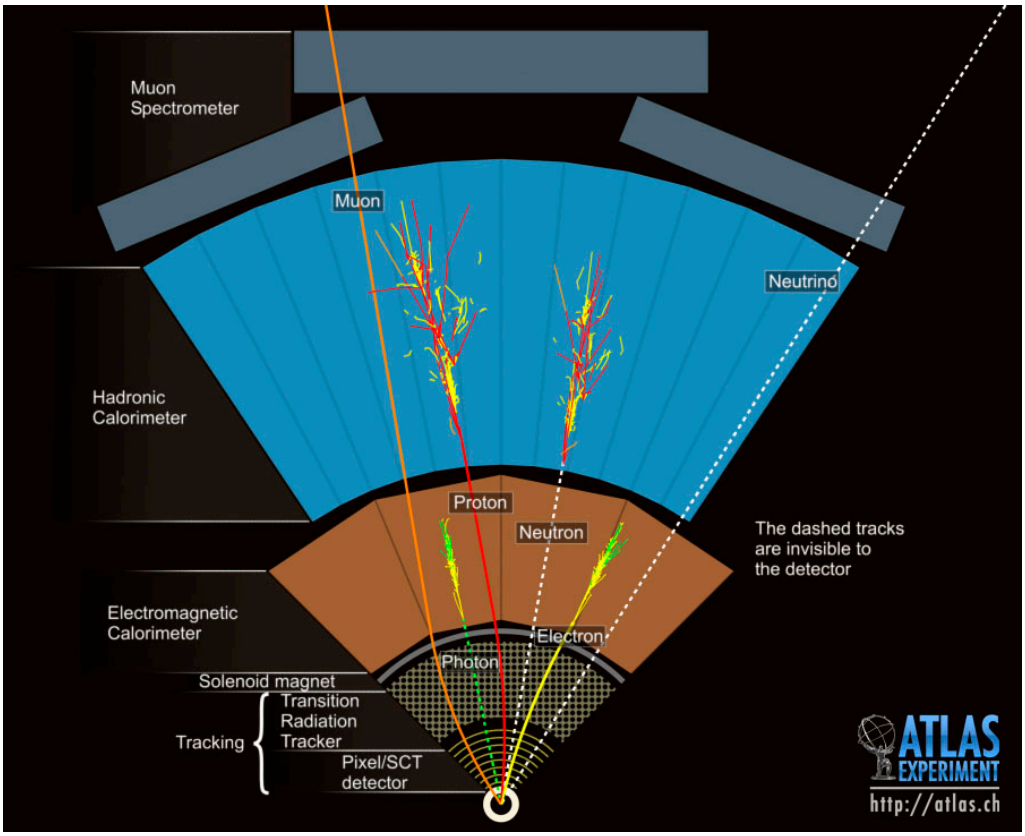


Figure I.3.3: Particle Detection in ATLAS

- electrons,
- photons,
- muons,
- (hadronically decaying) taus and,
- jets.

The short lifetime of the tau lepton, causes it to decay before detection. Either it decays leptonically (to an electron plus neutrinos or a muon plus

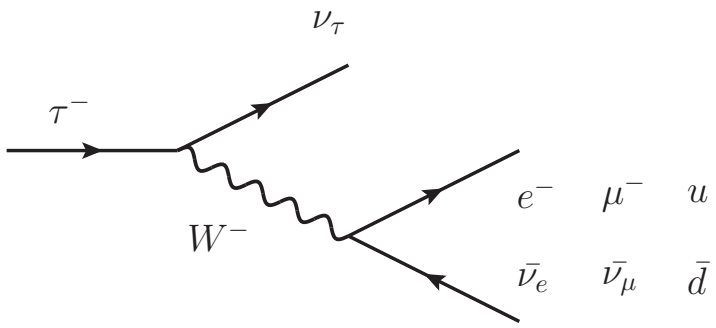


Figure I.3.4: Diagram for the decay of a τ lepton.

neutrinos), or hadronically to a quark-anti-quark pair, as depicted in Figure I.3.4. In order to provide discrimination for the hadronic tau decay against other jet-type objects, its kinematic and geometric properties (low invariant mass, collimation, etc.) can be exploited. In the leptonic case, it will result in a (fake) electron or muon as well as add to the missing energy.

These basic physics objects are used in analysis to deduce more elaborate quantities and create more complex objects. A simple example is a Z, which could be constructed by requiring two muons of opposite sign and an invariant mass close to the mass of the Z. Other deduced quantities that will resurface in later discussion, are:

Transverse mass, m_{T} – a Lorentz invariant quantity based on invariant mass, typically used when one or more particles in a decay avoid detection and only the $\not{E}_{\mathrm{T}}$ can be used to infer the energy.

H_T – the scalar sum of the transverse energy of all jets in an event, used as measure of jet activity in an event and often in searches for physics beyond the SM.

As will discussed later, the information available at early stages of the trigger system is not complete, leading to the notion of *candidates* and *trigger objects* rather than physics objects.

Coordinate System

The origin of the ATLAS coordinate system is chosen at the center of ATLAS, where the beams cross. The right handed Cartesian coordinate system is laid out such that the x axis points from the interaction point to the center of the LHC, the y axis points upwards towards the surface, and the z axis points in the direction of the beam, tangentially to the LHC. Often, a representation in pseudo-rapidity, η, and azimuthal angle, ϕ, falls more naturally than the Cartesian or a similar spherical representation, as particle production and distribution is more naturally described in (pseudo) rapidity. The pseudo-rapidity is defined as

$$\eta = -\ln\left[\tan\frac{\theta}{2}\right] , \tag{I.3.1}$$

where θ is the polar angle between a particles three-momentum and the beam.

I.3.2 The Inner Detector

Overview

The inner detector consist of three different types of tracking detectors with varying spacial resolution. The detectors are contained within a cylinder measuring $\mathcal{O}(7m)$ in length and $\mathcal{O}(1m)$ in radius. A charged particle

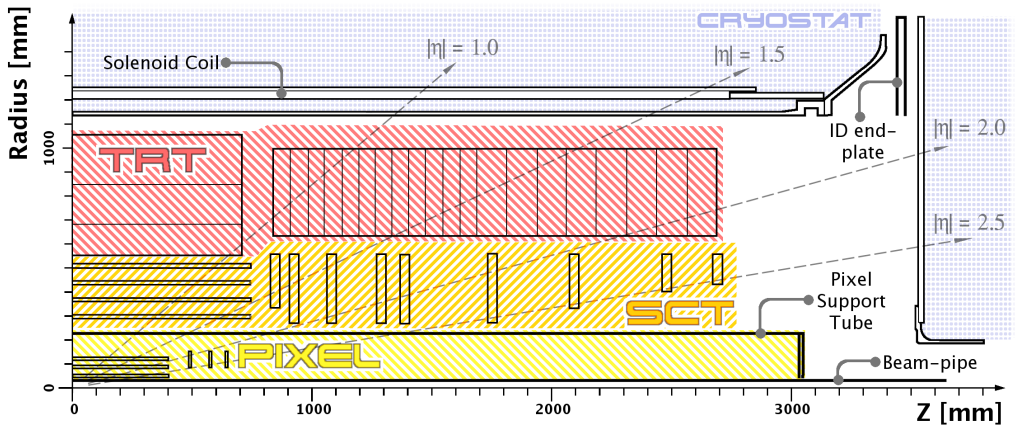


Figure I.3.5: Depiction of the inner detectors of ATLAS. Source: [35]

passing through these detectors will leave a series of localised energy deposits, *hits*, from which the trajectory of the individual particles can later be deduced. From the curvature of the tracks, the charge to transverse momentum can be extracted. The tracking provided by the inner detectors is also important for the correct determination of the vertex position and of the impact parameter. This is required for correctly identifying particles with a long lifetime – in particular particles, such as pions and b-quarks that rely on the weak interaction for their decay.

Subdetectors

The size and location of the tracking detectors of ATLAS is shown in Figure I.3.5.

Closest to the beam pipe we have the pixel detectors: The Pixel Detector consist of three barrels and three disks at each side holding a total of $\mathcal{O}(1700)$ pixel detector modules with an intrinsic accuracy per pixel of $10\,\mu\text{m} \times 115\,\mu\text{m}$. The design and placement are such that any particle in

$|\eta| < 2.5$ will pass through 3 layers. During the shutdown between Run I and Run II, a new subdetector, the Insertable B-Layer (IBL) [20], was installed, adding a fourth pixel layer closer to the interaction point and thus improving the resolution of both tracking and impact parameter estimation. The IBL is unfortunately not depicted on Figure I.3.5, but it is installed between the beam pipe and the innermost layer of the original Pixel Detector. The installation of the IBL, included the replacement of a section of the beam pipe with thinner one to make room the new detector. The new beam pipe extends to $R = 29$ mm. The IBL is 64 cm long and extends from $R = 31$ mm to $R = 40$ mm. The primary motivator for the IBL is the harsher conditions of Run II. The better resolution, on tracking and vertexing provided by the IBL will result in better momentum resolution which is important for triggering and background reduction.

Outside the Pixel Detector is the Semiconductor Tracker (SCT). The SCT consists of four barrels with silicon microstrip detectors on both sides, providing a total of eight hits per track. The microstrips are aligned with a typical tilt of ± 20 mrad relative to the beam direction, resulting in different resolutions in η and ϕ of roughly $20\,\mu$m $\times$ $580\,\mu$m. This so-called stereo view is depicted in Figure I.3.6. The two SCT end-cap detectors each consist of nine discs with silicon detectors on either side.

The Pixel and IBL have $\mathcal{O}\,(86.4\,\text{mil})$ readout channels[20] and SCT has $\mathcal{O}\,(6.3\,\text{mil})$ readout channels. The two detector technologies provide high resolution tracking information in the range $|\eta| < 2.5$.

The outermost part of the inner detector, is the Transition Radiation Tracker (TRT). The TRT consists of a total of $\mathcal{O}\,(300\,000)$ thin-walled pro-

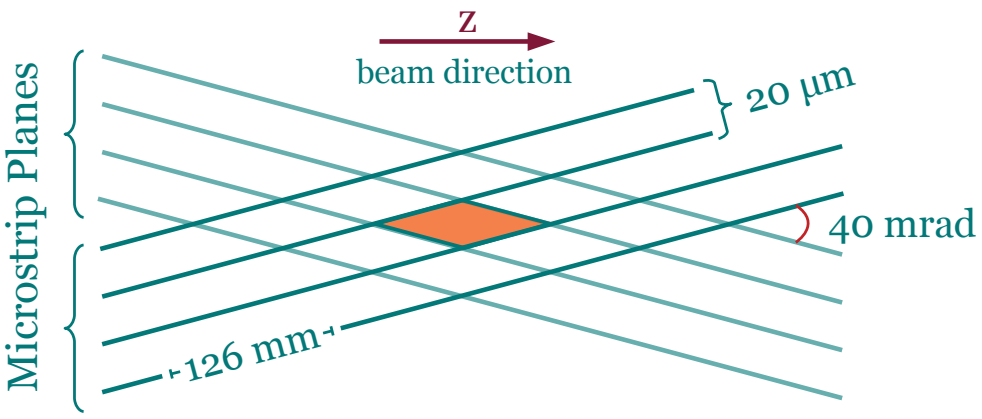
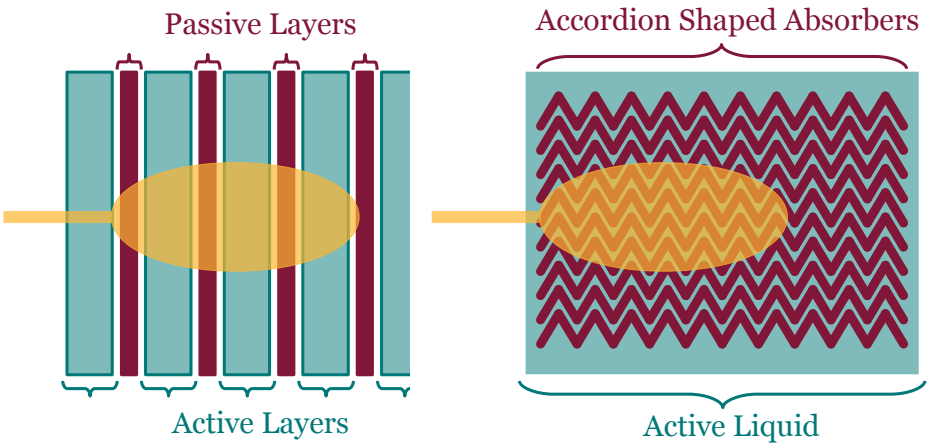


Figure I.3.6: Conceptual drawing of stereo angle between SCT active planes. Numbers from [48].

portional drift tubes, or *straws*, of 4 mm in diameter. By measuring drift time, or rather time-over-threshold, the resolution per straw is $\mathcal{O}\,(100\,\mu\text{m})$. The straws in the barrel, $\mathcal{O}\,(53\,000)$, are aligned parallel to the beam direction, while the straws in the two endcaps, $\mathcal{O}\,(123\,000)$ on each side, are radially aligned to the beam axis. The detector covers $|\eta| < 2.0$ and the geometry ensures that an incident particle will pass through $\mathcal{O}\,(40)$ straws[26]. The high number of hits per track compensates for the lower resolution.

Further, the TRT provides discrimination between electrons and other high energy particles through "high threshold" hits caused by transition radiation (thus the name of the detector): Transition radiation is produced at the boundary of the straw when a highly relativistic particle enters. Being radiated in a forward cone, the transition radiation photons enter with the particle and cause additional ionisation in the gas. As electrons typically have a gamma factor much higher than any other particles,



(a) Typical (b) Accordion

Figure I.3.7: Conceptual depiction of two types of sampling calorimeters.

this provides efficient discrimination.

I.3.3 The Calorimeters

ATLAS uses sampling calorimeters for determining the energy of incident particles. In a sampling calorimeter this is done by alternating layers of

passive absorber material – characterised by a short *radiation length*, X_0, and similarly short *nuclear interaction length*, λ_0,

active material – for detection and measurement of produced radiation and ionization.

A conceptual depiction of a typical sampling calorimeter is shown in Figure I.3.7a. Sampling calorimeters are cheap, and both active and passive materials can be optimized for different uses. The downside is that only a fraction of the deposited energy is detected in the active layers.

There are different dominant stopping processes for electrons and photons. For electrons, the dominant process is bremsstrahlung,

$$e^{\pm}\, N \longrightarrow e^{\pm}\, \gamma\, N\ , \tag{I.3.2}$$

while for photons, pair production dominates,

$$\gamma\, N \longrightarrow e^{+}\, e^{-}\, N\ , \tag{I.3.3}$$

where N is a nucleus of the absorber material. In one radiation length, X_0 an electron will have lost $\Delta E_{X_0} = 1/e$ of its energy to bremsstrahlung photons. A photon will typically produce an $e^{+}e^{-}$ pair after $9/7 X_0 \approx X_0$.

For incident hadronic particles, the interaction processes are of a lot more complicated nature, as they primarily involve strong interaction between the incident hadron and a nucleus of the absorber material. Typically a number of lighter secondary hadrons will be produced, some of which might decay to electromagnetic particles before interacting with the absorber material. The mean free path of the hadronic particles is the nuclear interaction length, λ_0.

For electromagnetic calorimetry, a high ratio between X_0 and λ_0 is preferred, to ensure as many electromagnetic interactions (or conversely as few hadronic interactions) as possible. For hadronic calorimetry, the ratio is of less importance. In both cases, relatively small values for at least one of the two parameters, X_0 and λ_0, are preferred, to minimise the effect of "punch through particles" in the muon system. – particles energetic enough to penetrate through the calorimeter and enter the muon detectors.

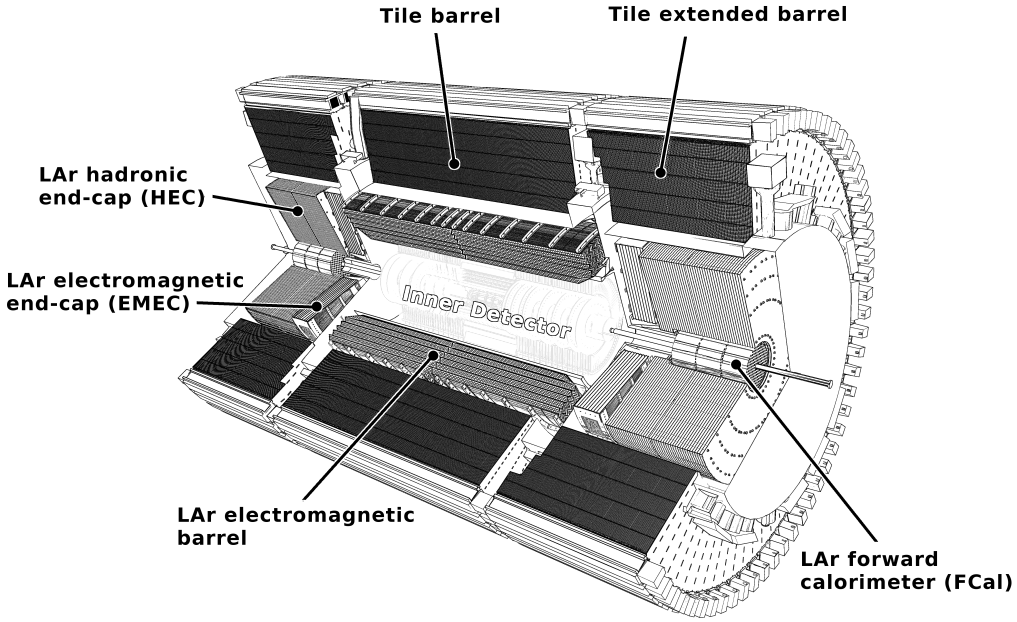


Figure I.3.8: The ATLAS Calorimeter System. Source: [35]

ATLAS calorimeter system is depicted in Figure I.3.8. The calorimeter system consists of five different calorimeters, which covers the range $|\eta| < 4.9$. The innermost four use *Liquid Argon* (LAr) as the active material but differ in material and design for the absorber. The electromagnetic calorimeters use lead as the absorber and use an accordion geometry for optimal azimuthal coverage without cracks, see Figure I.3.7b. The barrel calorimeter has three regions of varying radial length, resolution, and density, to provide discrimination between incident particles based on the shower development. The hadronic LAr calorimeters use a simpler geometry and use copper disks as absorber. The forward calorimeter also uses tungsten. The last calorimeter, the tile calorimeter, uses steel as absorber and scintillators as active material.

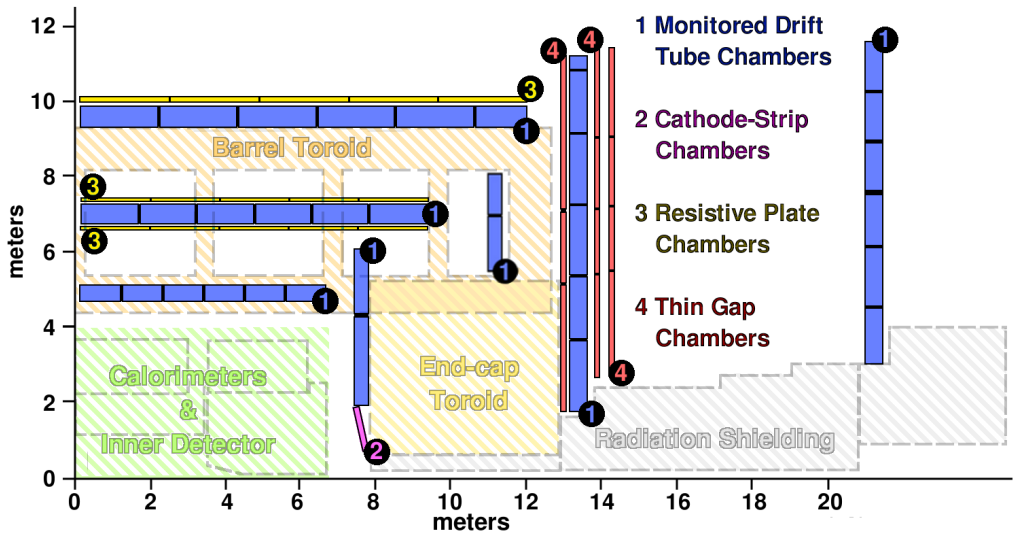


Figure I.3.9: Cross section view of ATLAS Muon Spectrometer. Source: [35]

Calorimeters typically allow for very fast readout typically $\mathcal{O}(5\,\text{ns})$ and are thus very well suited as trigger detectors. The entire calorimeter system has $\mathcal{O}(200\,000)$ readout channels.

I.3.4 The Muon Detectors

Further out, embedded in the toroidal magnetic field, are the various muon detectors, responsible for detecting and measuring the momentum of the muons, as these are not stopped by the calorimeters.

The ATLAS muon system consists of a broad range of detector technologies, some providing detailed track information on behalf of time-consuming readout, others providing coarse resolutions, but with a response time suitable for triggering. A cross section of the ATLAS muon

spectrometer can be seen in Figure I.3.9. In most of the pseudo-rapidity range, several layers of Monitored Drift Tubes (MDTs) – arrays of drift tube detectors – provide precise coordinate information of each track. In the very forward direction, where the background is much larger, Cathode Strip Chambers (CSCs) – a multi-wire proportional chamber – provide additional measurements on each track. Each CSC module contains four wire planes providing four η, ϕ measurements per track.

The triggering functionality is provided by Resistive Plate Chambers (RPCs) in the barrel region and by Thin Gap Chambers (TGCs) in the endcap region. In a RPC two resistive plates are seperated by a thin layer of ioniseable gas. High voltage is applied and the signal induced by an ionising muon passing through the gas is read out via capacitive coupling to metallic "pick-up" strips mounted on the outsides of the resistive plates. The spacial resolution provided is $\mathcal{O}\,(10\,\mathrm{mm})$, but timing is fast, $\mathcal{O}\,(5\,\mathrm{ns})$.

The TGCs are, like the CSCs, multiwire proportional chambers, but designed for high timing resolution, which (amonst other things) means shorter distance between wires and higher applied voltage. Besides providing muon trigger input, the TGCs complement the endcap MDTs by providing a measurement of the azimuthal component of the track coordinate.

Altogether the muons system has $\mathcal{O}\,(1\,\mathrm{mil})$ readout channels with roughly $300\,000$ each from MDT, RPC and TGC.

Between Run I and Run II, four Micromegas detector modules were installed into the same space as the CSCs are occupying. Micromegas is a relatively new gas detector technology that uses a micro-mesh to divide

the gas volume into two. Micromegas detectors can be operated fast and with very high gain. After Run II these are foreseen to be replaced the MDTs. For Run II they are not used as trigger detectors, but only for vertexing.

I.3.5 Other Detectors

The detectors covered in the previous sections are the main sub-detectors of ATLAS. A few other detectors deserve mentioning as they provide valuable input for the trigger system.

Forward Physics

Located roughly 240 m from the interaction point, along the beam line, four Roman Pots are installed – two on each side of ATLAS. These are the detectors of ALFA, a subdetector for luminosity measurement and for forward physics. During operations, the pots are moved close to the beam to detect and measure protons at low angles originating from elastic scattering in the interaction point. The detector deserves mentioning as it delivers a trigger input to the Central Trigger Processor and because the distance from the remainder of ATLAS means that this input is received so late that it needs to be treated specially to stay within the latency window of the trigger system.

Beam Monitoring

For luminosity and timing measurements on less than 25 ns level, ATLAS, and in particular the Central Trigger Processor, utilizes the Beam Conditions Monitor[25] (BCM) and the beam pickup stations.

The BCM is located close to the beam pipe on each side of the pixel detector. Four diamond pad detectors provides a precise Time-of-Flight measurement. Between Run I and Run II the Diamond Beam Monitor[24] (DBM) was developed and installed along with the IBL. The DBM compliments the BCM by additionally providing tracking information and discrimination against secondary particles.

The BPTX stations are provided by the LHC, but are operated by the experiments. The $\mathcal{O}\left(1ns\right)$ timing resolution on bunches makes the BPTX useful for reading the fill pattern. This capability is used by the CTP to classify and logically group different bunch crossings and for monitoring purposes.

Chapter I.4

ATLAS Trigger and Data Acquisition

With a nominal time between bunch crossings of $25\,\text{ns}$ and $\mathcal{O}\left(100\,\text{mil}\right)$ readout channels, the produced raw data volume far exceeds what can stored. The event size can be reduced to $\mathcal{O}\left(1.7\,\text{MB}\right)$ using online data formatting, zero suppression, and compression done by the detector readout. At $40\,\text{MHz}$ of collissions, this implies data rates of $\mathcal{O}\left(70\,\text{TB/s}\right)$. Such data rates can not be handled, stored, or later accessed in any meaningful way, and calls for a means of online data reduction. An overwhelming QCD background in the data, typically of $\mathcal{O}\left(10^6\right)$ above the physics processes of interest, further motivates that data rates should not only be reduced, but that ideally the signal purity of the recorded sample should be improved.

Providing a fast, efficient, physics motivated selection of what collision events to store is the primary role of ATLAS trigger.

This chapter will first outline the role of and interplay between the different parts of the trigger and the DAQ, followed by a discussion of how the trigger is operated to ensure an optimal event selection physics analysis.

Between Run I and Run II, the trigger received several upgrades, and the DAQ infrastructure was completely redesigned. This chapter will focus on the configuration used for Run II. A brief description of the primary differences between the Run I and Run II systems will be provided as context for the discussion in Part III. The trigger upgrades and their implications will be treated in Part II.

I.4.1 ATLAS Trigger at a Glance

Figure I.4.1 shows a schematic representation of the ATLAS Trigger and Acquisition (TDAQ) system. The data flow between detectors and DAQ components is seen on the right, and the different trigger parts are on the left.

ATLAS TDAQ consists of a two layered trigger system: a number of custom-built hardware systems close to the detector hardware for making the first selection, and a farm of conventional computers for making the final decision based on the full detector data. The first level is called *Level 1* and the second level is the *High Level Trigger* (HLT).

In between the Level 1 trigger and the HLT is the Region of Interest Builder (ROIB) – named so for historical reasons – which serves as the interface between the two layers.

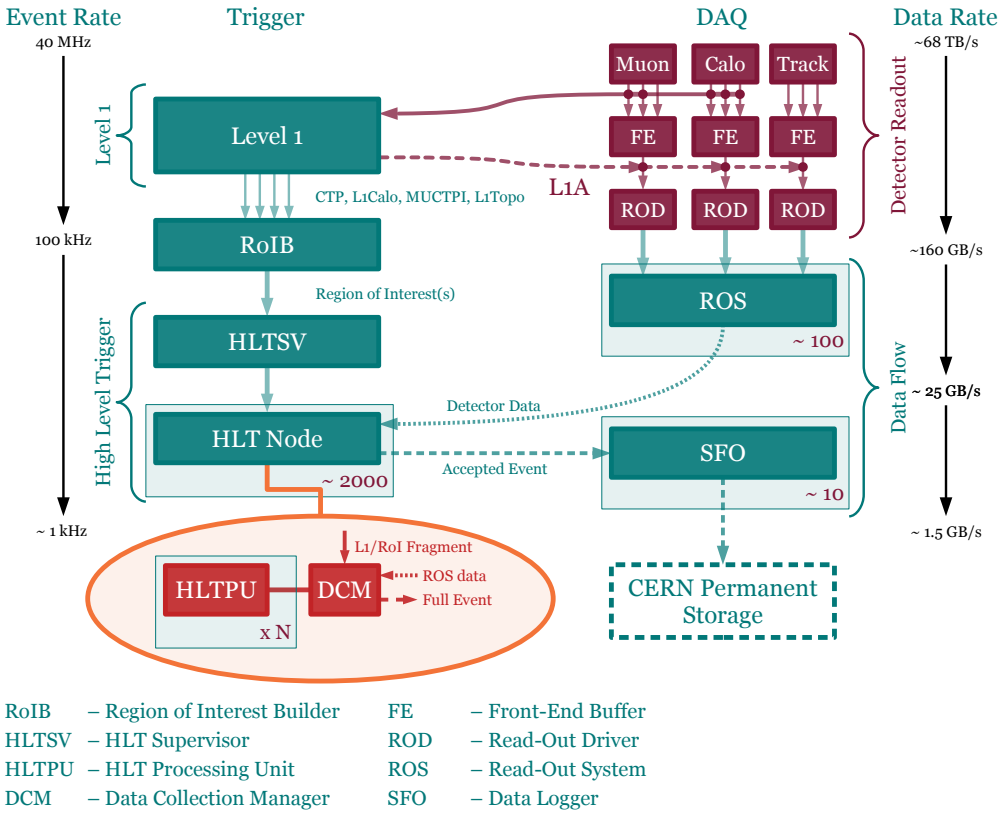


Figure I.4.1: Schematic of ATLAS Trigger and Data Acquisition for Run II.

The Level 1 trigger operates in a highly synchronous manner: data from the relevant subdetectors is received and processed at a constant rate $40\,\text{MHz}$. The operational constraints on the Level 1 trigger is a maximum accept rate of $100\,\text{kHz}$ with a maximum latency of $2.5\,\mu\text{s}$. The HLT operates in a highly parallel manner: The HLT Supervisor (HLTSV) distributes work to the HLT Nodes on a one-event-per-node basis and even within the HLT Node the processing is distributed on as many cores as the node has available. The operational parameters of the HLT is an accept rate of $1\,\text{kHz}$ at an average latency of $\mathcal{O}\,(200\,\text{ms})$.

I.4.2 From Collision to Storage

The maximum latency of the Level 1 trigger is $2.5\,\mu$s, corresponding to 100 collisions at the nominal collision rate of 40 MHz. During this time, event data is kept in pipeline memories on the detectors front-end (FE), awaiting response from the Level 1 trigger. The *trigger detectors*, including some of the muon detectors (TGC and RPC), and a few other detectors, send their data to the Level 1 trigger in addition to storing it in pipeline memories. The Level 1 trigger processes the received data in a synchronous manner and uses it to form an initial trigger decision. In case of accept, two things happen:

- The *Level 1 Accept* (L1A) signal, is sent to all the subdetectors;
- Trigger information from all parts of the Level 1 trigger is sent to the RoIB.

Upon receiving the L1A, a simple electric pulse that carries no additional information, the subdetector will transfer data from its FE pipelines to a Read-Out Driver (ROD). The ROD collects and formats the detector data before pushing it to a Read-Out System (ROS) – typically a computer with a network interface and an optical connection to one or more RODs. The formatted data can then later be requested or cleared by the HLT.

The RoIB, upon receiving detailed trigger information from the various parts of the Level 1 trigger, will compile a detailed trigger summary and send this to the HLT Supervisor (HLTSV). For historical reasons, the fragment produced by the RoIB is called a Region of Interest (RoI).

The HLTSV assigns the RoI fragment to an HLT Node which is then in charge of processing the event. The Data Collection Manager on the HLT Node will inspect the RoI fragment and request the full event data from the ROSes if needed. The event is then reconstructed using algorithms as close to those used offline for analysis as possible. The HLT Processing Units (HLTPU) running on the node will, based on the trigger information in the RoI fragment, execute a number of selection algorithms to determine if the event should be stored. In case of accept, the DCM sends the HLT result and the event data to one of the data loggers (SFO).

The SFO then transfers the full event data to a temporary network storage from which it is written to permanent storage and replicated on the World Wide LHC Computing Grid[22].

I.4.3 Constituents of the Trigger

Level 1 Trigger

A schematic representation of the Level 1 trigger is shown in Figure I.4.2.

The Level 1 trigger is composed of a number of dedicated trigger processors, each receiving data from various subdetectors. These trigger processors provide input to the CTP that forms the L1A from a set of logical conditions on the input.

The CTP receives most of its input from the calorimeter trigger processor (L1Calo) [3] and the Muon CTP Interface (MUCTPI) [10]. These in turn receive input from the respective trigger detectors of the calorimeter and muon systems at a constant rate of 40 MHz.

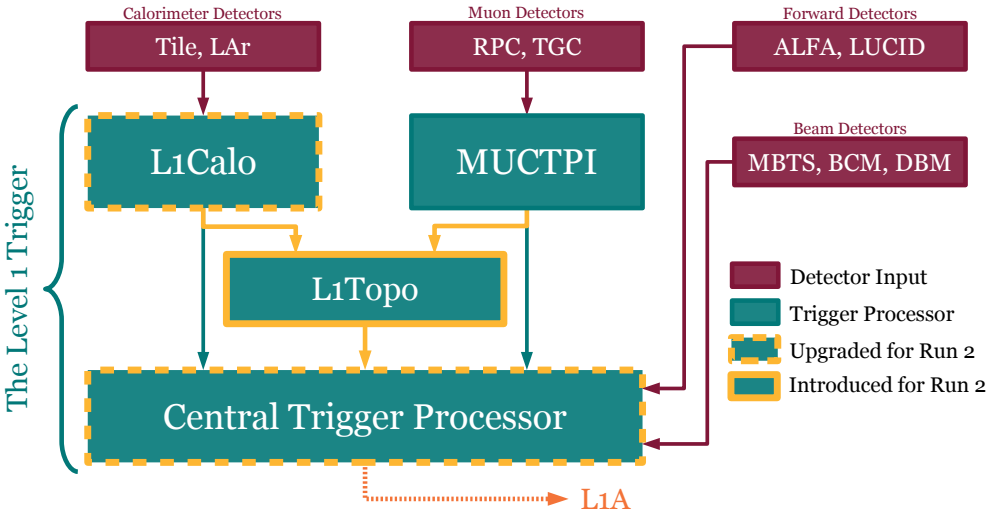


Figure I.4.2: Schematic of ATLAS Level 1 Trigger.

From the calorimeter information L1Calo identifies various trigger objects: electron, photon and tau/hadron candidates with transverse energy, E_{T}, above programmable thresholds as well as isolation in (η, ϕ) if so desired. L1Calo further identifies generic jet type objects, calculates the total, missing and jet-sum E_{T}. These quantities (type, thresholds, and multiplicity where applicable) are sent to the CTP every bunch crossing.

The MUCTPI, operating with six programmable p_{T} thresholds, counts the multiplicity of muon candidates above threshold. Unlike the L1Calo, the MUCTPI doesn't construct the trigger objects, but gets them directly from the RPC and the TGC. The multiplicity of each threshold is passed on to the CTP corrected for potential double counting.

No geometric information about the various candidates is provided by L1Calo or the MUCTPI. Between Run I and Run II a topological trigger processor (L1Topo) [46] has been installed. L1Topo uses information from both L1Calo and the MUCTPI to form trigger inputs to the CTP

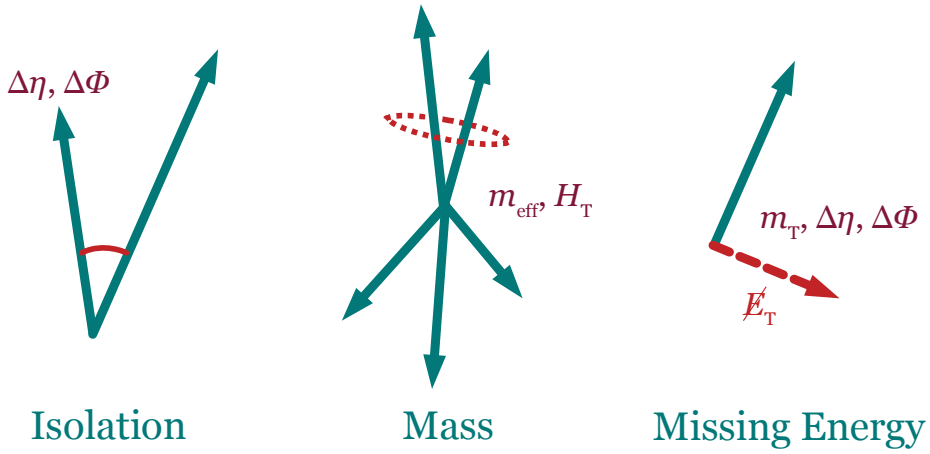


Figure I.4.3: Example topologies for use with L1Topo and typical kinematic variables

that are based on geometric properties of the event rather than on sole multiplicities. Example of such topologies can be seen in Figure I.4.3. The ability to exploit event geometry and topology can increase the efficiency for physics processes where unstable (intermediate) particles are produced. Typical examples are SM $H \rightarrow \tau\tau$ where one or both taus subsequently decay hadronically. By requiring a maximum separation in $(\eta,\ \phi)$ between the two τ candidates, the trigger rate from di-τ can be reduced without loss in signal efficiency. For B-physics, where a di-μ trigger is commonly used, geometrical constraints can dramatically increase the efficiency. Many searches for physics beyond the SM look at events with multiple energetic jets, and constraints on the transverse/invariant mass, spatial separation, and $\not{E}_{\mathrm{T}}$ can be used to greatly reduce the background. The latency constraints set by the remainder of the L1 systems and maximum latency of $2.5\ \mu$s leaves $\mathcal{O}\left(200\ \mathrm{ns}\right)$ – or roughly 10 BCs – for

L1Topo to receive, process the data from L1Calo and the MUCTPI and to transmit the constructed trigger objects to the CTP [46].

The above accounts for the primary trigger inputs to the CTP used during normal operations to trigger on physics events. In addition a few other detectors provide trigger inputs for the CTP, in particular:

TRT: a high occupancy trigger used during cosmics.

ALFA: numerous inputs for their forward physics program.

Tile: input for a calibration request trigger as the calorimeter is periodically calibrated using a laser pulse.

BCM, BPTX: inputs for minimum bias triggers but also for various timing purposes – sub-BC timing resolution and sensitivity at low bunch currents.

The CTP supports up to 512 logical combinations – *trigger items*, or simply *items* – to be formed from the received input. The L1A is formed as the logical OR of all defined items. Examples of L1 items from Run I[1] are shown in Figure I.4.1. The conditions are inclusive, and thus e.g., `L1_MU20` implies an event with *at least* one muon over the $20\,\text{GeV}$ p_T threshold. The operation of the trigger system will be discussed in more detail in Chapter I.4.4.

It should be stressed that L1Topo as well as the MUCTPI and L1Calo only have partial detector information available, and that the reconstructed candidates at L1 have a much higher uncertainty than the corresponding

[1]The triggers used for Run II is not yet made public.

Item Name	Level 1 selection
L1_MU20	Single muon, $p_{\mathrm{T}} > 20\,\mathrm{GeV}$
L1_2MU6	Two muons, each w. $p_{\mathrm{T}} > 6\,\mathrm{GeV}$
L1_J50xE60	One jet w. $E_{\mathrm{T}} > 50\,\mathrm{GeV}$ AND $\not{E}_{\mathrm{T}} >$ GeV
L1_JE1000	Jet transverse energy sum of $\sum E_{\mathrm{T,jet}} > 1000\,\mathrm{GeV}$

Table I.4.1: Example of L1 trigger items from Run I.

trigger objects at HLT or physics objects used later for analysis. Usually a tighter selection is performed by the HLT.

Following a Level 1 Accept, all the trigger processors of the Level 1 trigger transfer an event fragment to their respective ROS, containing a full trigger summary of the event. For L1Topo, L1Calo, and the MUCTPI, the following information about all candidates is provided: type, energy/momentum thresholds, isolation requirements, associated multiplicities (also provided to the CTP), and geometrical information (which is not provided to the CTP). The CTP event fragment contains a complete record of which of the 512 trigger conditions were fullfilled as well timing and synchronisation related information such as the unique event number and the bunch crossing identifier the accepted event belong to. These fragments are also sent via optical fibers to the RoIB when the L1A is issued.

Region of Interest Builder

As the Region of Interest Builder (RoIB) is exposed to the full Level 1 Accept rate, it too is implemented with custom made dedicated electrical

boards, as no commodity hardware exists that can gracefully handle the high data rates. The RoIB creates the Region of Interest (RoI) fragment by combining the trigger relevant information from the event fragments received from the Level 1 trigger processors into one record. The RoI fragment is then sent to the HLTSV over network.

High Level Trigger

The High Level Trigger (HLT) consist of server computers. One of these[2] serves as HLT Supervisor (HLTSV) for the HLT Nodes.

The HLTSV receives the RoI fragments from the RoIB and schedules events to the HLT farm, by assigning one HLT Node per event. The HLT Node runs a Data Collection Manager (DCM) process and a number of HLT Processing Units (HLTPU) processes depending on the number of cores available on the given node.

The DCM unpacks the RoI fragment and inspect the list of L1 conditions that resulted in the L1A. From the list of L1 items, a list of HLT algorithms is compiled based on a global configuration of the HLT. If the list is empty, the event is rejected immediately and a clear request is sent to the ROSes, to release the data for that event. If it is non-empty, the entire detector data is requested from the ROSes by the DCM and reconstructed. Sharing the reconstructed event data, the HLTPUs execute the scheduled selection algorithm.

In case of accept, the full event data, including the RoI, and a similar summary fragment produced by the HLT, is sent to the data logger.

[2] Up to two is supported for redundancy.

Evolution of the HLT

The HLT of Run I consisted of two parts: Level 2 (L2) and Event Filter (EF), both based on commodity computers. The L2 received the RoI fragments from the RoIB, and used the geometric information to request a subset of the detector data. A number of decision algorithms were then be executed on the requested detector data without event building.

Upon accept, a Level 2 Accept (L2 Accept) was then issued, in which case a centralised Event Builder (EB) requested the full detector data and constructed an event object that was then sent to the EF along with the RoI and the L2 Result fragment. Based on the event object and the trigger results from L1 and L2, the EF scheduled and executed a set of more sophisticated selection algorithms.

The idea behind L2 was to achieve high rejection by only requesting fractions of the data. But as the number of RoIs increased with pile-up, so did the amount of data requested by L2. Since the EB needed to request the same data again, a lot of strain was put on the ROS computers.

To accomodate this, for Run II, L2 and EF were merged, and the Dataflow network was redesigned accordingly. The redesign further reduces the overhead of having to pack, transfer and unpack data between computers, as most tasks can be handled within the same HLT Node.

I.4.4 Triggering for Physics

This section covers topics important to how the trigger is normally operated and how the available bandwidth is being utilised.

Operating the Trigger

Trigger Menu

The trigger menu (or simply menu) is primarily composed of the definition of the 512 trigger items and the L1 items used to seed the various HLT algorithms. The menu is usually divided into a L1 specific part and a HLT specific part, as most of the additional configuration applies specifically to either of these. Examples include frequencies of random generators used by the CTP and the maximum execution time for an algorithm at HLT. The menu is considered static and can not be changed during data taking.

Prescales

During data taking however, the instantaneous luminosity will drop and so will the rate of all physics motivated triggers. To account for this and to generally allow fine grained control of the trigger rates, *prescaling* is applied to all items at Level 1 and all algorithms at the HLT.

Prescaling is the means by which the rate of triggering, and thus data recording, can be throttled by only accepting every $N^{\underline{\text{th}}}$ trigger item/algorithm. This provides an efficient means of reducing the overall throughput, and the per item/algorithm prescaling allows for a physics motivated selectivity in what physics signatures are stored and at what rate. During Run I *deterministic prescaling* was applied in the CTP, meaning that it was exactly every $N^{\underline{\text{th}}}$ trigger that was accepted. In Run II, with the upgraded CTP, *probabilistic prescaling* is applied: an item is accepted with a prob-

ability of $P_{\text{item}} = 1/N$. For normal operation, the discussion of deterministic versus probabilistic prescaling is unimportant – the rate is throttled to one $N^{\underline{\text{th}}}$ in both cases. In some corner cases of operation, however, as well as subsequently, during analysis, when estimating and correcting for trigger (in)efficiencies, it becomes important.

The HLT applies:w probabilistic prescaling to each algorithm, but *before* its execution. This is was also the case for Run I.

Tables of prescale values for all L1 items and all HLT algorithms are often referred to as either *prescales sets* or as *prescale keys* (as the tables are stored in a database and later queried using an identifying key).

The prescale sets can be – and are – changed during data taking, and doing so is *the way* of controlling the trigger rates during operations. Typically a number of such prescale sets are generated targeting different ranges of instantaneous luminosities. These can then be reused unless the menu or the desired relative bandwidth between trigger items change.

Dead-time

Attempting to operate the trigger at too high rate will cause trouble; typically either a detector can't read out fast enough or the HLT can't process events fast enough. In this case the affected system becomes *busy*. This information is propagated back to the CTP by a system-dependent method and used to veto new triggers until all systems have recovered. Unless the trigger rate is regulated, this will display an oscillatory behaviour.

The fraction of time in which no events can be accepted is called *dead-time*. The dead-time should be minimised, as it translates directly to a measure of how much of the provided LHC luminosity that is lost. Another reason to attempt to minimize the dead time, is that it directly affects the efficiency of the physics motivated triggers by acting as a democratic prescale on all items.

There are other causes of dead-time, in particular *preventive dead-time* imposed by the CTP to protect detector readout and prevent data loss or corruption. Dead-time is also introduced when the trigger is being held, e.g., during configuration changes, to avoid undefined behaviour.

Applying and monitoring dead-time is an important task of the CTP and is discussed in Chapter I.5.

Lumiblock time

Internally, ATLAS divides periods of data taking into *luminosity blocks*, or *lumiblocks*, during which all operational parameters – such as the instantaneous luminosity and the various configurations – are considered stable. The typical length of a lumiblock is set to 1 min for Run II but it could be as short as 10 s if a configuration change is made. The typical length of a lumiblock was changed during Run I from 2 min to 1 min, motivated in part by the high peak luminosity delivered.

Balancing the Bandwidth

The trigger reduces the overall event rate by a factor of $40\,000$, and the remaining 1 kHz bandwidth needs to be carefully balanced between the

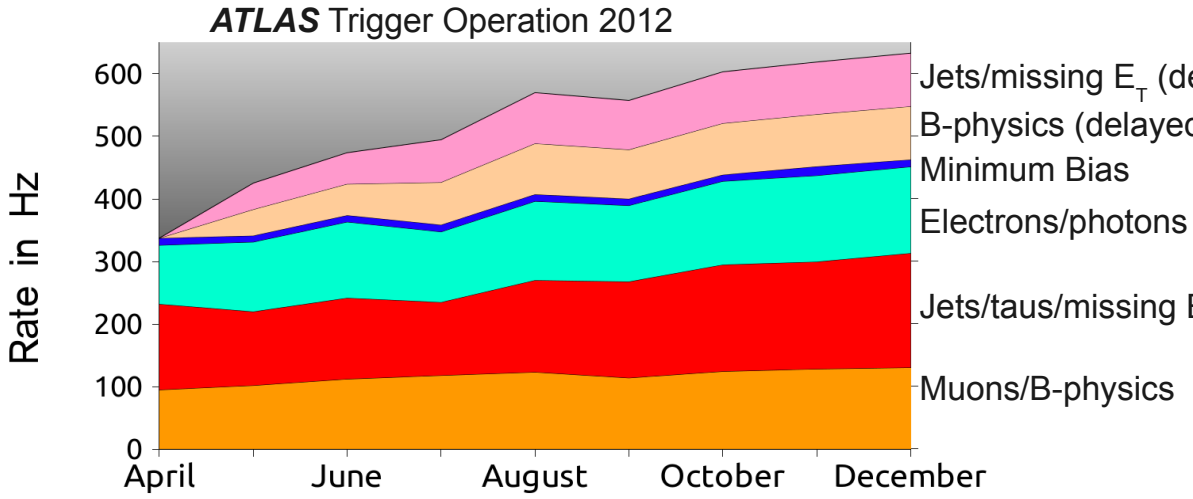


Figure I.4.4: The bandwidth allocation per slice during 2012 data taking. Source: [11]

various physics analyses.

For physics analysis a data sample in a particular Signal Region of phase space is required. Most analyses also require one or more Control Region samples, which are orthogonal to the signal region. What classifies as Signal Region, and what is suitable for Control Region depends on the analysis in question. In some cases, Control Regions can be effectively obtained by inverting the selection of the Signal Region – e.g., Signal Region defined my invariant mass within a certain window, control region as everything else. In other cases the Control Region can be obtained through a statistically independent set of observables – e.g., Signal Region defined by muon trigger, Control Region obtained by jet trigger.

The various trigger items are divided into "slices" corresponding to the respective observables, namely: Electron/Photon, Muon, Tau, Jet,

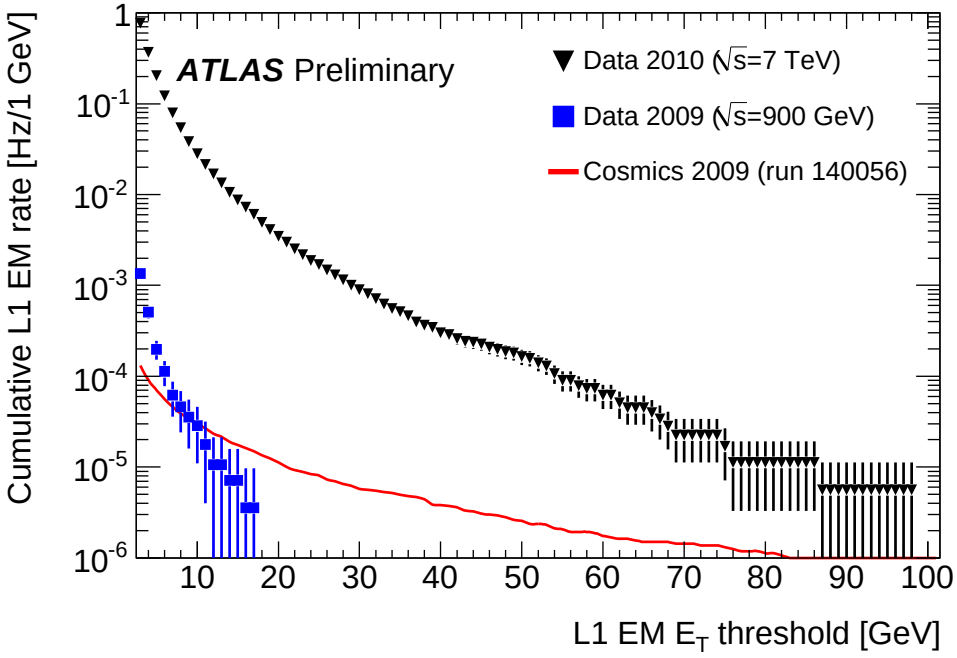


Figure I.4.5: Trigger rate of electromagnetic objects as function of threshold Source: [12]

Missing E_{T}, b-jets, B-physics and MinBias/Forward Physics. Figure I.4.4 shows the bandwidth allocation per slice as function of time during 2012 data taking.

Most of the available bandwidth is taken up by single lepton triggers, as they have broad applicability and a clear signal, leading to high efficiencies. The thresholds are generally kept as low as possible at trigger level to keep as much of the phase space as possible for analysis, but increasing the threshold proves an efficient means of reducing the rate. As an example, the Level 1 rate for electromagnetic items (primarily e and γ) is shown in Figure I.4.5 as function of threshold energy threshold. Another means for controlling the rate, is requiring isolation in (η, ϕ). This

became necessary by the end of Run I to reduce the rate without reducing phase space too much. The spacial resolution for isolation requirement during Run I was quite coarse. Improving the resolution was one of the main motivations for the upgrade of L1Calo (though, the introduction of L1Topo was probably *the* primary motivation).

For jet triggers, multi object trigger items are the most common, and rates are balanced between multiplicity and threshold. For the missing energy triggers, the threshold is simply kept as low as possible.

It is desired to keep the primary triggers unprescaled and balance the threshold instead, as this gives finer control with the rate than can be offered with integer prescales.

For both low threshold and control sample triggers, prescales are often applied, to either manage the rate or reduce it proportionally to the primary triggers.

I.4.5 Challenges for Run II

The increase in peak luminosity expected in Run II, as well as the increase in beam energy, is expected to lead to a factor of 5 increase in interaction rate (a factor of 2 from the luminosity and a rough factor of 2.5 from the increase in total cross-section). The higher interaction rates will directly impact the operation of the trigger, and consequently the bandwidth for Run II has been increased: The Level 1 rate has been increased from $75\,\text{kHz}$ to $100\,\text{kHz}$, and the HLT accept rate has been increased from $\sim 500\,\text{Hz}$ to $1\,\text{kHz}$. The mismatch between expected rate increase and bandwidth

increase implies that higher rejection must be achieved in Run II. If unprescaled single lepton triggers are to be used exclusively, this will require tighter isolation and higher thresholds. In particular for tau triggers, this means cutting into the regions of phase space most relevant for analyses. Single tau triggers already suffer from high energy thresholds to distinguish the hadronic tau candidates from the dominant QCD background.

For Run II, L1Topo will provide better capabilities for triggering on multi particle objects, such as jets and (hadronically decaying) taus, which should improve the trigger performance for such trigger objects.

Another challenge for Run II is *out-of-time pile-up*, which is a detector phenomena that occurs when the time between collisions becomes lower than the readout-time of the detector. For a $25\,\text{ns}$ bunch spacing, this will be a concern for many of the subdetectors in ATLAS, particularly the calorimeters and the detectors relying on drift times.

Chapter I.5

The Central Trigger Processor

The Central Trigger Processor (CTP) is the autonomic nervous system of ATLAS. The CTP both provides adequate response to sensory input, forming and sending the Level 1 Accept (L1A) to all subdetectors, and also controls the clock distribution that synchronises all parts of the system. With a diameter of $\mathcal{O}(25\,\mathrm{m})$ the ATLAS detector is *huge* compared to the interaction frequency, and a precise timing distribution is paramount to avoid data corruption. For example, a muon produced in a given collision will only reach the outer parts of the detector one to two bunch crossings later. The CTP is also the system responsible for applying dead-time in ATLAS. There are two types of dead-time in ATLAS: preventive dead-time applied by the CTP to protect the subdetectors and dead-time due to a throttling signal – a busy-signal – being sent to the CTP, typically from a subdetector or from DAQ. As these are vital functions for the experiment and for data taking they are all carefully monitored by the CTP.

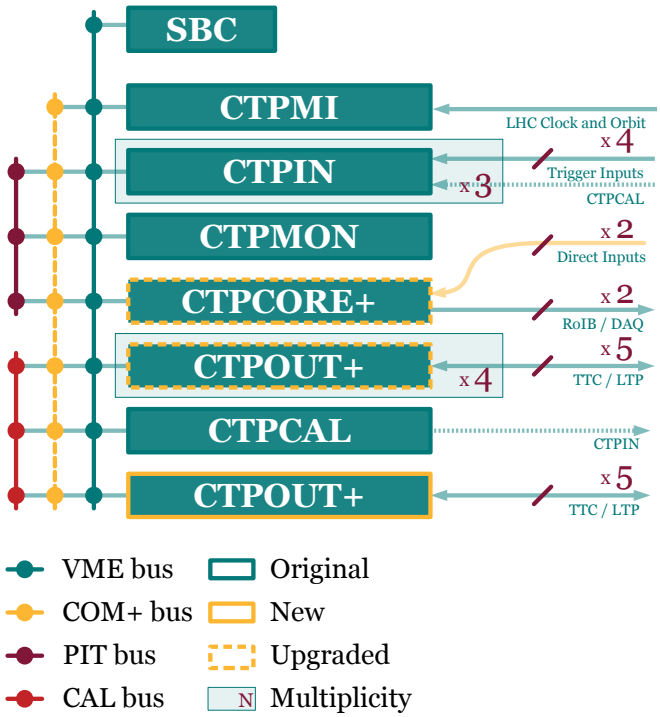


Figure I.5.1: Board, bus and I/O layout of the CTP

The topic of this chapter will be the anatomy and functionality of the CTP. The primary focus will be the trigger and dead-time generation as well as the CTPs monitoring capabilities. The following discussion holds for the CTP of Run I and for the primary uses of the CTP for Run II. No distinction will be made between the two systems, unless explicitly stated. A treatment of the CTP upgrade and its implications is the topic of Part II.

I.5.1 Anatomy of the CTP

Boards of the CTP

Figure I.5.1 shows a schematic overview of the CTP and its constituents. The CTP is hosted in a standard 9U VME crate and consists of 12 custom made boards and a single board computer (SBC) with a VME interface for controlling the boards. The SBC communicates with the boards via the VME bus. The boards communicate amongst themselves via the three other buses: the *COM bus* for common trigger and timing signals, the *PIT bus* for the synchronized trigger inputs and the *CAL bus* for handling calibration requests from the subdetectors.

The CTPMI is the Machine Interface board of the CTP and is used to interface the LHC machine. From the LHC, the CTPMI receives the 40 MHz bunch clock synchronised to the beam, corresponding to one tick per 25 ns, as well as the orbit signal that defines one revolution, or correspondingly 3564 clock ticks or bunch crossings.

The CTPINs receive the trigger input from L1Calo, the MUCTPI, and most of the trigger detectors. L1Topo and ALFA provide their input directly to the CTPCORE. The role of the CTPINs is to align the trigger inputs in time.

The CTPMON monitors the PIT bus on a per bunch crossing level.

The CTPCORE generates the L1A, compiles the full trigger summary, and sends this to the RoIB and the ROS. The CTPCORE is also where dead-time is generated, received, applied, and monitored.

The CTPCAL tackles calibration requests from the subdetectors and produces a trigger input to the CTPIN, via a cable on the front, in case the request is accepted.

The CTPOUTs route and monitor Trigger, Timing, and Control (TTC) signals between the CTP and the various subdetectors and trigger processors.

Signals on the COM Bus

As the COM bus plays a central role in the CTP and in its upgrade, and as the signals on the COM bus covers most of the Trigger Timing and Control (TTC) signals distributed to the subdetectors, their meaning should be covered. A bit for bit schematic of the signals on the COM bus is shown in Figure I.5.2.

The use and generation of most of these signals will be discussed in the following sections, with exception of the SYN signal, which is an internal test signal not used during normal operations, but included on the schematic for completeness. The multiplicity of signals affected by the upgrade as well as the Trigger Type words, relates to the support for multiple users of the upgraded CTP. This will be discussed in greater detail in Part II.

The TTC signals on the COM bus are:

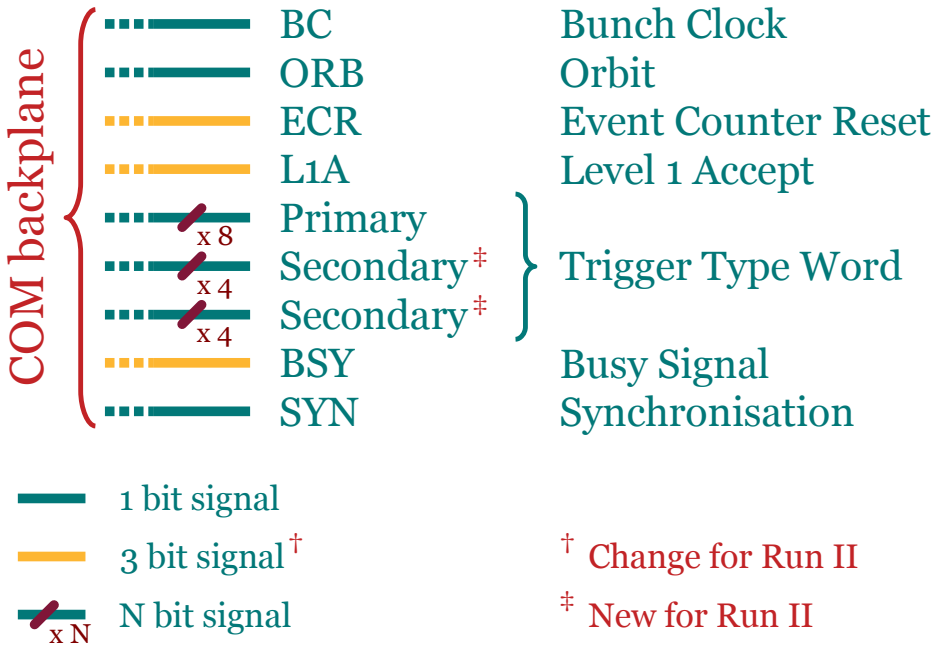


Figure I.5.2: Schematic of the signals on the COM bus.

BC, ORB are the timing signals used internally by the CTP and distributed to the remainder of ATLAS.

ECR, L1A are both part of the trigger signals. The L1A is indeed *the* Level 1 Accept signal. The ECR is periodically issued and distributed to all systems to reset the L1A counters[1]. The combination of ECR counter and L1A counter uniquely identifies a L1A.

Trigger Type Word contains a condensed version of what triggers caused the L1A to be issued. With at most 8 bit available only the type of trigger (muon, calorimeter, physics, calibration, etc.) can be encoded.

[1] At 40 MHz a 32 bit counter will overflow in less than 2 min.

BSY is the logic OR of busy signals received by the CTPOUTs from the subdetectors.

I.5.2 TTC Distribution

On the CTP site, the CTPOUT manages the routing of the various TTC signal on the COM backplane to and from the subdetectors of ATLAS. In addition to the TTC signals on the COM bus, a calibration request signal from the subdetectors is also handled by the CTPOUTs

The general direction of transmission is from the CTP to the subdetectors, the only exception being the busy signal and the calibration request. In response to a busy signal, the CTPOUT will raise the corresponding busy signal on the COM bus where it is later picked up by the CTPCORE to veto triggering. Similarly, a calibration request received by the CTPOUT is made available to the CTPCAL on the CAL bus. If the calibration request is accepted, the CTPCAL sends the trigger to one of the CTPIN boards, via a cable on the front panel of the board.

The CTPMI receives the bunch clock and orbit signal from the LHC and is responsible for the periodic generation of the Event Counter Reset (ECR) used by all subdetectors to synchronise event data. The CTPMI makes these signals available to the remainder of the CTP boards on the COM bus.

The L1A and Trigger Type word is made available on the bus by the CTPCORE when a L1A is issued.

A CTPOUT is highly configurable and allow detailed routing of the individual signals.

I.5.3 Applying Dead-time

Subsystem dead-time

The dead-time applied for the subsystems can be divided in two:

- busy-signals from subdetectors – either because of trigger rate limitations or as part of a recovery procedure;
- "back pressure" from the DAQ/HLT as event data can not be requested or processed fast enough.

While both result in a throttling of the trigger rate, the signal propagation differs.

Sub-detector busy is communicated via the TTC system and received by the CTPOUTs. The CTPOUTs raises the busy flag on the COM bus where it is read by the CTPCORE. The CTPOUT can be configured to ignore the busy signal from one or more subdetectors.

The back pressure from DAQ/HLT occurs when either DAQ or HLT can not keep up with the rate of events accepted by the Level 1 trigger. This causes the readout buffers of the trigger processors of the Level 1 trigger – including the CTP – to fill up, as accepted events can not be sent to the RoIB fast enough. In case the problem is with the HLT, this happens because the RoIBs buffers fill up as the HLT is not accepting the RoIs. When one of the readout buffers on one of the trigger processors becomes full, the corresponding system will raise a subdetector busy. In case of the CTP this can be handled internally in the CTPCORE.

Preventive Dead-time

The CTPCORE introduces two types of preventive dead-time, usually referred to as simple and complex dead-time.

The simple dead-time imposes a veto of a number of bunch crossings after each L1A, typically 4 bunch crossings. This protects the subdetector front end electronics against receiving triggers while an event is being read out, which potentially could lead to data corruption.

The complex dead-time is implemented using four *leaky bucket* algorithms, each characterised by two numbers: an integer size of the bucket, s, and an (inverse) leak rate, r, in units of bunch crossings. At every L1A, one token is added to the bucket, and every r bunch crossings, one is removed. If the number of tokens in the bucket reaches s, the bucket is full and a veto signal is formed. The setup emulates the detector front-end derandomizer buffers, where event data is temporarily stored after the L1A is received. Events are accepted in a random manner (following Poisson statistics), and read out (or cleared) from the buffer at more or less a constant rate. In case of a burst of triggers, the buffer would overflow and data would be lost. The CTPCORE implements four such leaky buckets. The two complex dead-times, corresponding to the two veto groups, is formed by a logical OR of the bucket-full flag from a programmable subset of the four buckets. This is in contrast to Run I where two buckets were implemented – one for each of the two complex dead-times. The simplest argument for this change is that owing to differences in implementation of the readout, size of event fragments, etc., each detector requires a different value for the size and the leak rate of the leaky

bucket, In the setup of Run I, one would compromise between the two most restrictive bucket configurations, obtaining a functional but not optimal solution: using e.g., the smallest bucket size and the lowest leak rate will likely satisfy the requirements of protecting the buffers but is less optimal – provides more dead time – than an OR between several individual buckets, each representing an individual buffer. The result of the Run II setup is a reduced complex dead-time and a configuration closer to the buffers that the complex dead-time aim at protecting.

The preventive dead time, being configurable, is an interesting quantity as it, along with the filling scheme, limits the maximum accept rate of the system as a whole. While the simple dead time will increase linearly, as the time between triggers decreases, the complex dead-time will increase quite steeply as the rate approaches s triggers per r bunch crossings.

Other Dead-Time Sources

In addition to these common causes of dead-time, it should be mentioned that triggering can also be held manually by using the CTPMI or the CTPCORE to artificially raise the busy on the COM bus. For $100\,\mu$s before and after the ECR generation, the CTPMI raises the COM bus busy to avoid spurious triggers while the respective counters throughout ATLAS are being reset.

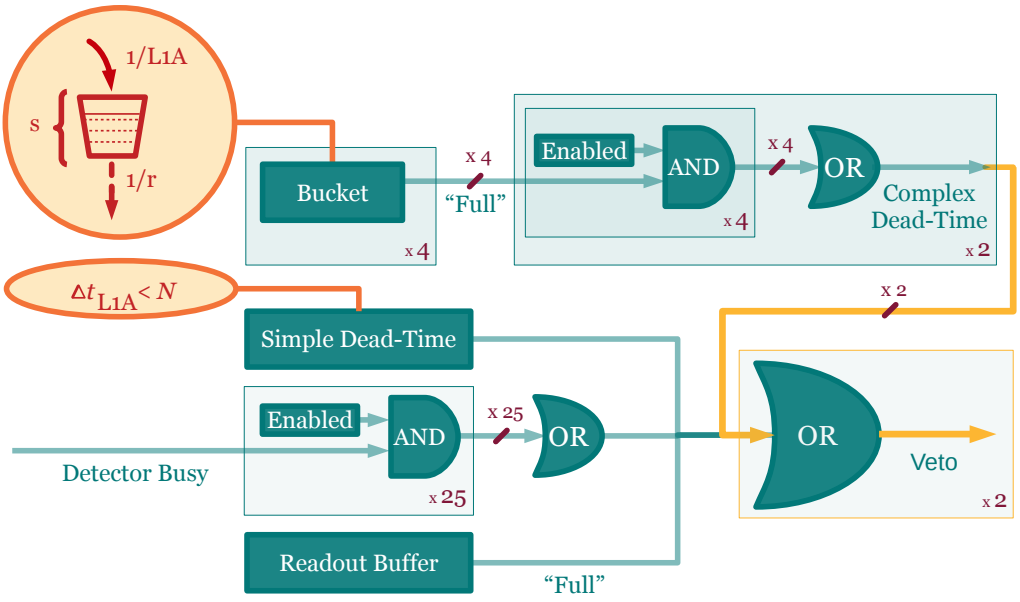


Figure I.5.3: Schematic of the CTP veto logic for the two veto groups.

Veto Groups

The different sources of dead-time are combined to form two veto signals, following the logic of Figure I.5.3.

Any trigger item belongs to a *veto group* decided by which of the two veto signals are used. The second veto group has never been used in operation, and with the new TDAQ system the gain would be little to none. Consequently, all trigger items belong to the same veto group.

The motivation and envisaged use was to allow some trigger items – typically associated with detectors with fast readout – more room for bursts, by applying a more relaxed complex dead-time. With the L2 trigger of Run I, where only data from a subset of detectors was requested and used by the algorithms, it would be possible to start the L2 selection while the remaining detectors finished readout, and thereby save a couple

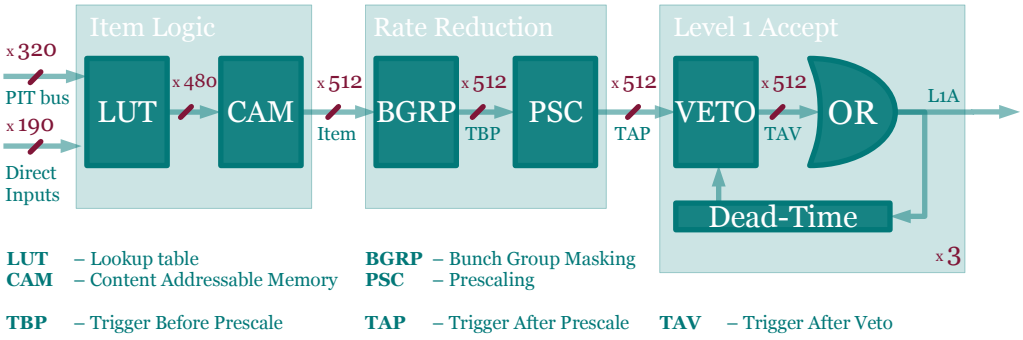


Figure I.5.4: Forming the Level 1 Accept in the CTP.

of milliseconds. The implications for the dead-time corrections and the low complex dead-time during Run I left this feature unused.

I.5.4 Forming the L1A

Outline of the Trigger Path

Figure I.5.4 shows the Level 1 trigger path in the CTPCORE. The L1A is formed in the CTPCORE by logically combining the trigger inputs it receives – either from the PIT bus or directly – into 512 trigger items. The L1A is formed as the OR of all items after prescaling and veto has been applied.

Bunch Groups

So far, a Level 1 item has been treated as solely consisting of a logical combination of the trigger inputs, but there is an additional requirement being imposed, namely in what bunch crossings a trigger is allowed. In many cases this is trivially the bunch crossings with colliding bunches.

The CTP operates with a concept of *bunch groups* Bunch groups are a data abstraction used to group each of the 3564 bunches per turn into more meaningful groups. The bunch groups often reflects the bunch structure, and commonly defined groups include the set of bunch crossings with colliding bunches as well as the set of bunch crossings with no colliding bunches. Most physics related trigger items are coupled to the bunch group of the colliding bunches. While the two mentioned groups are completely disjoint, a bunch crossing can belong to several groups at the same time. Another commonly defined group is used with calibration request triggers. This group thus defines when calibration request triggers can be issued but is independent of the bunch structure. The CTPCORE currently supports up to 16 bunch groups.

For items belonging to other bunch groups than the group of colliding bunch, is usually named accordingly: the Level 1 item `L1_TAU8_EMPTY` is based on the same logical condition as `L1_TAU8_EMPTY`, namely a τ candidate with E_{T} above 8 GeV. While for physics this item is uninteresting it gives a good handle for monitoring unexpected behaviour, such as a hot cell in the calorimeter that leads to a spurious trigger rate.

Trigger Inputs

The trigger inputs from L1Calo and the MUCTPI – the thresholds and multiplicities of trigger objects as described in Chapter I.5 – are received by the CTPIN boards on each bunch crossing. The CTPINs internally align the inputs that might arrive out of time and synchronize it to the bunch clock. The time aligned *trigger inputs* (TIPs) – historically called “Pattern

In Time (PIT)" – are put on the PIT bus where they are picked up by the CTPCORE.

The trigger input from ALFA and L1Topo is provided directly to the CTPCORE via the Direct Inputs. This saves some latency compared to going via the CTPINs and over the PIT bus but require the sender of the input to format and time align the trigger inputs – the job normally carried out by the CTPINs – before sending it. For ALFA and L1Topo this is necessary to stay within the latency window.

Random Triggers

In addition to the ordinary TIPs, the CTPCORE is equipped with four random generators with programmable frequency that each provide a TIP signal that can be used to seed a trigger item. Random triggers are one way of obtaining an unbiased data sample, usually referred to as a Minimum Bias or Zero Bias sample. In some operational setups the random triggers are used to stress test one or more systems or simply to ensure a minimal throughput.

Trigger Items

The TIPs are then combined logically in the CTPCORE – to require, say, a lepton above a certain threshold as well as missing energy. This happens in the LUT and the CAM.

Bunch group masking is then applied by comparing the current bunch crossing identifier to the 16 bunch groups and determining which ones are

active. A logical AND between the input conditions and their respective bunch group is then made to form the final item.

The Level 1 Accept

Probabilistic prescaling is applied per item with the possibility of configuring an item to have an infinite prescale or conversely zero chance for causing an L1A to be issued. This would typically be the case for an item such as the `L1_TAU8_EMPTY` discussed above, but is also used as a convenient way of toggling individual items on and off. After prescaling dead-time vetoing is applied and the L1A condition is formed as the OR of all items and the preventive dead-time algorithms are updated accordingly.

In case of accept, the Trigger Type word is formed and made available on the COM bus along with the L1A itself. The full trigger record is compiled, formatted, and transfered to both ROS and RoIB.

I.5.5 Monitoring

The Need for Monitoring

Monitoring is an important part of the responsibilities of the CTP as it provides crucial information, both about data taking and about the state and condition of ATLAS as a whole.

Going back to the analogy of the CTP as the nervous system of ATLAS, it might be evident that monitoring of vital functions – such as the clock distribution, the number of accepted events, the total dead-time, etc –

is of importance to ensure proper functioning of the trigger and the integrity of the recorded data. What might be less evident is that almost any perturbation or hick-up in any part of ATLAS reflects immediately on the CTP. Two non-trivial examples are:

- A wrongly adjusted (amplifier) gain on one side of one of the calorimeters was discovered by an unusual high rate for $\not{E}_{\mathrm{T}}$ triggers.
- A problem with a power converter for one of the magnets resulted in an increased number of high p_{T} muons.

In addition to the total dead-time, per subsystem monitoring of the busy signal proves an efficient way of identifying problems and for placing responsibility.

Types of Monitoring

The primary things that are monitored by the CTP are:

Configuration throughout the system, as sanity check, checking that the applied configuration matches the desired configuration.

Rates and counts of the TIPs and the trigger items For PITs, monitoring with per orbit resolution is done by the CTPINs, while monitoring with per bunch monitoring is done by the CTPMON for a programmable subset of PITs. Similar monitoring capabilities are provided by the CTPCORE for the trigger items at all stages (before prescale, after prescale and after veto). The rate and total number of L1As is also measured in the CTPCORE.

Dead-time and busy fraction is monitored by the CTPOUTs per subsystem with per orbit resolution. The total dead-time as well as the individual fractions contributed by the simple, complex, and, total subdetector dead-time is monitored at both per turn and per bunch resolution in the CTPCORE.

Part II

Upgrade of the CTP for Run 2

Introduction

The upgrade of the Central Trigger Processor (CTP) for Run II reflects the collaborative work of numerous people over a timescale that far exceeds that of my own involvement in the CTP, ATLAS trigger, or high energy physics in general. The following discussion of the CTP upgrade will inevitably touch upon the work of others, though the focus and the topics covered in the discussion are those most relevant for my own work.

For the sake of clarity, the extent of my involvement and responsibilities should be clearly stated. My areas of responsibility and my contribution to the upgrade of the CTP can be summarised as:

- Design and development of all software relating directly to the CTPOUT+ – including the low level libraries, test suits, and capture programs ;
- The design of the new software architecture for Run II and the development of shared components ;
- All aspects of monitoring the CTP – starting at board level, going through all the steps that allow the information to be displayed as well as stored for later use.

In particular the latter, the monitoring, has been the main focus. Through my work, I have learned the often underestimated importance of monitoring: It provides the eyes through which we view our experiment, and without it, we are fumbling in the dark with no means of understanding the behaviour of the experiment or the data it provides.

Chapter II.1

The CTP+

The upgrade of the CTP took place between Run I and Run II, along with other upgrades to ATLAS. For the discussion of the CTP upgrade, the most notable changes are the installation and integration of new detectors (IBL, DBM and Micromegas) and the addition of a topological trigger processor (L1Topo) in the Central Trigger. Other parts of the Central Trigger underwent upgrades in the same period, but none of these has a direct impact on the discussion of the CTP upgrade.

II.1.1 Motivation

The primary motivator for the upgrade between Run I and Run II is the increase in beam energy and instantaneous luminosity scheduled for Run II of the LHC. These changes will result in higher in-time and out-of-time pileup, leading to harsher experimental conditions on all fronts. For Run II the accept rate of the CTP has been increased from 75 kHz to 100 kHz,

Parameter	**Used**[1]	**Available**	**Upgrade**
PIT-bus Lines [a,b,c]	160	160	320
Distributed TTC Partitions [c,d,e]	1	1	3
Input Cables [a]	9	12	
Output Cables [d]	20	20	25
Trigger Items [c]	241	256	512
Bunch Groups [c]	8	8	16
Max Bits per OR Term for Items [c]	6	12	15
Per Bunch Item Counters [c]	12	12	256

[1] as of 2012

Component

[a] CTPIN
[b] CTPMON
[c] CTPCORE
[d] CTPOUT
[e] COM backplane

Table II.1.1: Resource limitations and usage of the CTP. Source: [8]

while the total latency of $2.5\,\mu$s is kept the same[1][15]. While the interaction rate is higher, it is necessary to reduce it to roughly the same accept rate while maintaining physics-motivated selectivity. To do this, a number of upgrades to the central trigger have been carried out, including the upgrade of the L1Calo as well as the introduction of the topological trigger processor L1Topo. The upgraded L1Calo gives higher spatial resolution in the calorimeter clusters, while L1Topo provides trigger inputs to the CTP based on event geometry. As seen in Table II.1.1, the CTP was operated very close to the design limit in Run I. The main motivation for the upgrade of the CTP is to remove these limitations so as to accommodate the newly introduced subdetectors and the requirement of more (and more complex) trigger items.

[1]The limitation is not the CTP but a design limit on the (detector) readout[15].

To effectively commission the other newly installed detectors without affecting data taking, the upgraded CTP can provide an additional two TTC partitions, that can serve in parallel to the primary one.

II.1.2 The Upgrade

Table II.1.1 shows the parts of the CTP affected by the individual upgrade requirements. The implementation of the upgrade goals and their implications will the subject of the remainder of this section.

Firmware Upgrade

The number of PIT bus lines effectively limits the number of trigger inputs the CTP can receive and use to form trigger items. By using double data rate (DDR) signalling on the (existing) PIT backplane, the number of lines is effectively doubled at the cost of an estimated latency penalty of $\mathcal{O}(2\,\text{BCs})$[2] at each end. This solution required a firmware update of the CTPINs and the CTPMON, and it was preferred as it require minimal change and the introduced penalty, where important, can be accounted for by CTPCORE+.

COM+

Providing three TTC partitions, with replication of trigger and control signals[3] for each of the partitions, implies the distribution of said signals on

[2] Multiplexing and de-multiplexing can be estimated to $\mathcal{O}(1)$ clock-tick

[3] Timing signals, as provided by the CTPMI, remains shared between partitions.

the COM bus. As these signals are timing critical, and as the COM backplane was not designed with this use in mind, an upgrade is called for.

CTPOUT+

The task of routing the TTC signals between the COM bus and the subdetectors increases in complexity with the introduction of multiple TTC partitions. Not only does the upgraded board allow routing on a per-signal level, but it also ships with a number of additional features. In contrast to the original boards, the busy signal from the subdetectors can be monitored per bunch crossing. The CTPOUT+ is further equipped with a 1 GB memory that can be used either for recording signals from the COM bus and subdetectors or for playing back a desired signal sequence to the subdetectors. [4] or for playing back a desired signal sequence to the subdetectors. This provides a new and powerful means for controlling, monitoring and debugging the TTC distributions.

CTPCORE+

As can be seen from Table II.1.1 the CTPCORE module is affected by nearly all upgrade goals. While employing the functionality of the original module, the CTPCORE+ implements several new features and a different architecture which distinguishes between latency critical and noncrititical functionality. The functionalities of the CTPCORE+ that are furthest from the original are the direct inputs for trigger items and the partitioning of resources for multi-partition operation.

[4] Time-wise, this corresponds to roughly Θ (10 s)

Direct Inputs

The solution to the number of PIT bus lines introduces a latency penalty, problematic for the use of certain trigger detectors. Upon receiving data from L1Calo and the MUCTPI, L1Topo has $\mathcal{O}(100\,\text{ns})$ to make a decision [47]. The added latency from the CTPIN to the CTPCORE(+) must be taken from this budget, with devastating consequences for the performance. The added latency would render ALFA useless as a trigger detector, since it is far from the interaction point and already struggles to stay inside the required time window. The 192 direct inputs provided by the CTPCORE+ avoid the additional latency, but require the sender to align the trigger inputs in time, a job normally undertaken by the CTPINs. When received by the CTPCORE+, the direct inputs are treated on equal footing as the inputs received on the PIT bus.

Resource Partitioning

For each of the three TTC partitions, the CTPCORE+ implements a separate set of preventive dead-time settings, both the simple and the complex, to best accommodate the needs of the subdetectors associated with a given partition. The 512 trigger items are shared non-exclusively between the three partitions, allowing each partition to use an item with a veto group of choice. Each of the partitions can choose to completely ignore some or all items. As the prescale of an item is not shared, the non-exclusiveness of items pose an operational challenge and is mainly useful for running a TTC partition parasitically. The readout machinery sparked by a L1A is not replicated for each of the partitions, and only an L1A from

the *primary partition* (or *physics partition*) will cause an event header and trigger record to be formed and shipped to the RoIB[5]. This implies that data taking can be done with at most one partition, the primary one. In essence 3 operational modes can be for seen:

1. Physics – All resources, including readout, used by primary partition

2. Parasitic – Most resources, including readout, used by primary partition, one or two secondary partitions use remaining resources.

3. Comissioning – Three partitions running, each using a subset of resources. No readout.

[5]Event counter and other such measures in the event header will also correspond to that of the primary partition.

Chapter II.2

Software Architecture

Besides ensuring the correct operation of the CTP, an important responsibility of the software is the monitoring and recording of trigger-related quantities. These quantities are crucial for characterizing the integrity of the recorded data, and are also used in real-time as a means to identify problems affecting the trigger and ATLAS data taking. The monitoring data are also used in real-time during operation as a means to identify problems that affects the trigger, and thus ATLAS data taking.

The CTP software for both Run I and Run II can conceptually be divided into a low level part, responsible for talking to the electronics boards, and a high level part, responsible for talking to ATLAS Online Software. Because Run II drops the Run I requirement of concurrent usage of the CTP, the software architecture was consequently redesigned.

ATLAS Online Software is a highly distributed multi-computer multi-process environment, and relevant topics will be covered before the discussion of the software architectures of Run I and Run II.

II.2.1 Importance of Monitoring

Dead-time Correction

Among the responsibilities of the CTP software is the precise monitoring of the exact number of triggers and of dead-time, which is crucial for later use of recorded data.

In a typical cross-section measurement, the cross-section times branching fraction[1], $\sigma \times \mathrm{Br}$, is calculated as the number of background subtracted signal events, N, divided by the integrated luminosity, L:

$$\sigma \times \mathrm{Br} = \frac{N}{\epsilon} \frac{1}{L} \tag{II.2.1}$$

where ϵ is the intrinsic efficiency of the measurement.

For determining the integrated luminosity L, corrections for dead-time and prescales are needed:

$$L = \int p \, \ell \, \mathcal{L} \, dt \, , \tag{II.2.2}$$

where p and ℓ are the time-dependent correction factors for prescales and dead-time respectively. The time integral can be replaced by a sum over lumi-blocks, which implies a per lumiblock resolution on the dead-time.

Even higher precision on the dead-time correction can be achieved through the per-bunch monitoring capabilities of the CTP: In (I.2.9) it was assumed that the number of protons in each bunch and in each beam

[1]The branching fraction is the probability of decay via a particular mode

is the same. In reality this not the case, and the instantaneous luminosity is better described as a sum over the colliding bunches:

$$\mathcal{L} = \frac{f \sum_k n^k_{\mathrm{B1}} n^k_{\mathrm{B2}}}{4 \pi_x \pi_y} , \tag{II.2.3}$$

where n^k_{B1} and n^k_{B2} denote the number of protons per bunch in beam 1 and beam 2 at the $k^{\underline{\mathrm{th}}}$ collision. This exploits the symmetry of the LHC fill pattern, which ensures collision of the same two bunches at the same bunch crossing each turn. Substituting the previous expression into (II.2.2) and replacing the integral to a sum over lumi-blocks one obtains:

$$L = \sum_l p^l \, \frac{f \sum_k \mathscr{d}^{lk} n^k_{\mathrm{B1}} n^k_{\mathrm{B2}}}{4 \pi_x \pi_y} \tag{II.2.4}$$

where $\mathscr{d}^{lk}_{\mathrm{dt}}$ is the dead-time correction factor for the $k^{\underline{\mathrm{th}}}$ colliding bunch in the $l^{\underline{\mathrm{th}}}$ lumi-block.

Various measures for the dead-time can be obtained from the monitoring data, but most importantly from the CTPCORE+ where the total dead-time can be monitored at both lumi-block and per bunch level. The latter was not supported by the old CTPCORE, but the dead-time could still be estimated by determining the per-item dead-time for a number of representative trigger items – typically low threshold and minimum-bias items. As the various item rates are monitored, an estimate of the dead-time per item can be obtained as:

$$f_{\mathrm{dt}} = 1 - \frac{N_{\mathrm{TAV}}}{N_{\mathrm{TAP}}} , \tag{II.2.5}$$

where N_{TAV} and N_{TAP} is the triggers after veto and after prescale respectively. The triggers items can be monitored at lumi-block resolution for all items and at per-bunch level for a selected subset.

Online Monitoring

The monitoring data provided by the CTP during operation are used for more than ensuring that the CTP is functioning correctly.

As discussed in Chapter I.5.5, the trigger (starting with the CTP) is usually affected by perturbations in other parts of ATLAS. Consequently, the applications for displaying the monitoring data of the CTP is amongst the most used detector-specific software.

Another related use of the monitoring data is an automatic feedback system that gathers most of the published monitoring data from the CTP and uses it to detect external problems, and adapts the CTP configuration accordingly to protect the trigger system and the data taking.

II.2.2 Groups of Software

Before diving into technical aspects of the software it is worth taking a second to outline the tasks that the different parts of the software tries to solve.

The CTP software can be grouped into three parts:

Low Level Libraries that provide access to the electronic boards of the CTP;

High Level Software used during data taking and implemented as part of ATLAS Online Software.

Auxilary Software that uses a broader range of the functionalities of the CTP to conduct various tests on the electronics boards and on the CTP system as a whole.

Both High Level Software and the Auxiliary Software makes heavy use of the various Low Level Libraries. The Auxilary Software does not make use of the ATLAS Online Software and are not used during data taking under normal circumstances.

Any software application that makes use of the Low Level Library must be run on the Single Board Computer (SBC) in order to reach the boards over VME. See Figure I.5.1 for layout of the CTP. Software that does not need direct access to the CTP over VME does not need to be run on the SBC but can be run on any computer.

II.2.3 ATLAS Online Software

In order to ensure the correct functioning and configuration for the entirety of ATLAS, from subdetectors to readout systems, a huge number of computers and processes must be orchestrated. The hightly distributed and parallel nature of this problem requires an effective means of both inter-process communication (ipc), often between processes running on different computers, as well as procedures for synchronising and recovering in case of errors. ATLAS online software defines a *software partition*

(not to be confused with the TTC partitions discussed so far) that defines the computer and detector resources to be used, the applications to be run, and their life time. The partition further defines a set of infrastructure applications and services, offering amongst other things the necessary ipc, a finite state machine for communication, and synchronisation throughout the partition.

It is important to note that several software partitions *can* and typically *do* co-exist. There are typically two motivations for this. The first is when the lifetime of one or more applications needs to exceed that of the partition, and the second is that a more natural separation of resources, applications, or the state exists. The normal operation of ATLAS is done within this framework of a software partition. There is normally one central partition, *the ATLAS partition*, and a number of indirectly related partitions, which run in parallel and may be related to subdetector specific operation.

Software Partition Constituents

A software partition is given its own

1. name space – which is used for all ipc as the top level identifier. (In case of the ATLAS partition, “ATLAS” is often used.)

2. state machine – which establishes procedures for recovery and internal synchronisation

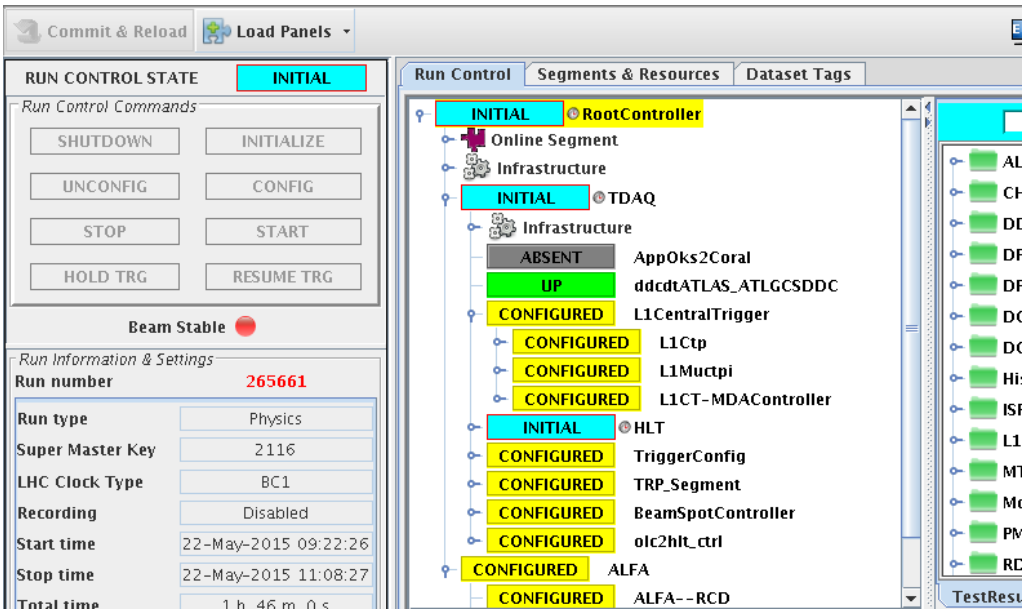


Figure II.2.1: ATLAS Run Control application.

3. infrastructure applications – which routes ipc and provides common services, the most important of which will be the topic of this section.

4. definition of resources and configuration

A partition additionally contains a minimum of one *segment*, which in turn each may contain one or more *resources*. The segment is a specialisation of a resource, and thus one segment might contain more segments. On a less abstract level, a resource is any part of the system, an application or even a detector part, such as an HLT Node, that can be individually modified without affecting the data taking[2]. Similarly, segments are used to group resources and segments of a particular kind. The nested

[2]This is a truth with modifications, but it suffice for here

structure of segments usually implies a degree of hierarchy. For example, the TDAQ segment usually includes a Level 1 segment and a HLT segment, and the HLT segment contains description of racks of computers, etc. The state transitions are forward propagated from the top-level partition to the segments and from the segments to their resources, while the state itself is backwards propagated from the leaf resources and back to the top-level partition. The TDAQ is not considered "Configured" until the HLT and the remainder constituents have reached "Configured", and the the HLT is not considered "Configured" until all computers of the HLT are "Configured". [3]. Figure II.2.1 shows the Run Control application used to steer and monitor the state of the partition. The state control of the entire partition is in the top left, and the tree in the center of the applications shows the different segments and their current state: TDAQ and ALFA are two top-level segments and TDAQ contains an HLT and a segment related to the Central Trigger. The Central Trigger segment further contains a segment for the CTP, which (finally) holds the application(s) needed for the ATLAS partition to utilize the resources of the CTP. These processes and applications will be the topic of Chapter II.2.4. Other detectors or sub-systems of ATLAS are described in a similar hierarchal manner within the partition.

[3]The framework extends to recovery from failed transtitions and mixed states states, but this is beyond the scope of this discussion.

Configuration Description

The configuration of a software partition is done via OKS[36], an in-house developed object database that uses XML for both class definitions and data instances. The OKS description is incredibly rich and contains every bit of information needed for running and configuring any resource in the partition. This includes a detailed description of the software to be run, the environment and arguments to start the software, what computer to start it on, the CPU architecture of the computer, and the actions that should be taken should the application crash or hang. It also contains a full description of the detector resources and their use within the partition. This again is with a level of detail that includes length of pipelines, descriptions of each cable connection, and in some cases even the description of the usage of individual signals. The full OKS description is made available to any application within the partition by one of the infrastructure applications. This allows an application to query relevant details of the detector description and even deduce its proper configuration from whether a resource, such as a cable or another application, is being used or requested somewhere by someone. The OKS configuration of another partition can not be accessed from within the partition, as there are no guarantees of the existence or potential lifetime of a software partition. The configuration of a partition is considered frozen when the partition is started. Thus, changes made to the (OKS) configuration while the partition is running will not be propagated to the partition. This implies that you can not request more resources on the fly but rather must request your resources before starting the partition. It further implies that

run-time configuration, such as prescale changes, are not fully described by OKS. For these types of configuration parameters OKS holds an initial configuration, i.e., the prescales that are loaded when the partition is started.

Databases and Traceability

Operating an experiment as large as ATLAS, requires the effort of a lot of people. It also require that changes made to the system by various experts get synchronised, and that the configuration being used is known at all times. In order to provide adequate traceability, the entire OKS description is kept under version control. As the OKS contains a complete description of the resources used, from detectors and cables to software and firmware versions, the version control provides a traceable description of every partition started. From the name and start-time of the partition, a complete description of used resources and their configurations can be obtained. Run-time changes to configuration, such as prescale key changes, are written to one or more databases using the internal luminosity-block time of the partition, in which all experimental and configurable parameters are assumed constant. Only one of these databases will be briefly touched upon, since they are generally beyond the scope of discussion. However, it should be noted that these databases usually live outside the partition as the y need to be available to any partition at any time and because they do not benefit from the infrastructure provided.[4], as the service and data they provide need to be available to

[4]In most cases, outside *any* software partition.

any partition at *any time* and because they don't benefit from the infrastructure provided.

Trigger Database

ATLAS uses a relational Oracle database to store relevant trigger information, including

- mappings of bunch crossing identifiers to bunch groups,
- trigger item logic definitions (conditions on TIPs and bunch group mask)
- sets of prescales to be used with trigger items at L1 and HLT

and much more. Each of these can be referred to and retrieved by using a database key.

The applied keys, along with partition name and time stamp of when the configuration was applied, are written to a database for traceability.

Inter-Process Communication

The need for ipc has been mentioned a number of times in this section already, but without much justification. Several cases require an efficient communication between processes, which include

1. communication with infrastructure applications. For example, obtaining the served OKS configuration requires communication between the requesting application and the service providing OKS.

2. run-time configuration that can not be done via OKS – such as prescale changes

3. propagating state machine transition and errors – from the Run-Control application in Figure II.2.1 to the segments, and from the segments to (sub)-segments and resources, and so forth.

4. local synchronisation between processes – general client-server or server-server communication implemented locally in the software between selected applications.

ATLAS uses CORBA[5] as the subsystem for routing ipc. An interface for communication is established through an interface description language (idl) from which appropriate headers and libraries are generated. An application is usually addressed by the partition name and the application name – both of which are available at runtime via OKS. While CORBA also facilitates communication between infrastructure applications and partition-defined processes, this is usually hidden under another layer of abstraction. The ipc implementation of ATLAS allows for communication between software partitions as follows: an application in partition *A* can address an application in partition *B* if it knows the name of the application running in partition *B*. The application in partition *A* is limited to reading its own (OKS) configuration and thus obtaining the name of the application in partition *B* may be difficult, without making assumptions.

[5]Amorim:1998ue

Information Service and Online Histogramming

Another central part of the online software infrastructure of ATLAS is the Information Service (IS)[45]. IS is another in-house development, which, like OKS, uses XML for class definitions, but unlike OKS, uses online servers to host the actual object instances. Any application can publish and object to an IS server, and any application may subscribe to an IS server to receive changes to a publication of a certain type or name. IS does not provide any ownership or access control, and information is in principal available on equal footing to everyone for as long as the server is alive. Information published on an IS server has the typical lifetime of the IS server which implies the lifetime of the partition. To avoid cluttering, a separate IS server is typically requested per sub-system, and addressing is usually done using the partition, server, and publication names. Examples are the RunParams server that hosts run parameters of the partition, and the L1CT server that hosts information related to the CTP and so on.

IS is the most used way in ATLAS software to broadcast information that might be relevant to other applications. An extension to IS is the Online Histogramming (OH) service, which uses IS as the protocol for publishing histograms. In the following, where IS is discussed in context of the CTP software, there will not be made a distinction between IS and OH, though histograms are an important part of the publications provided by the online CTP software.

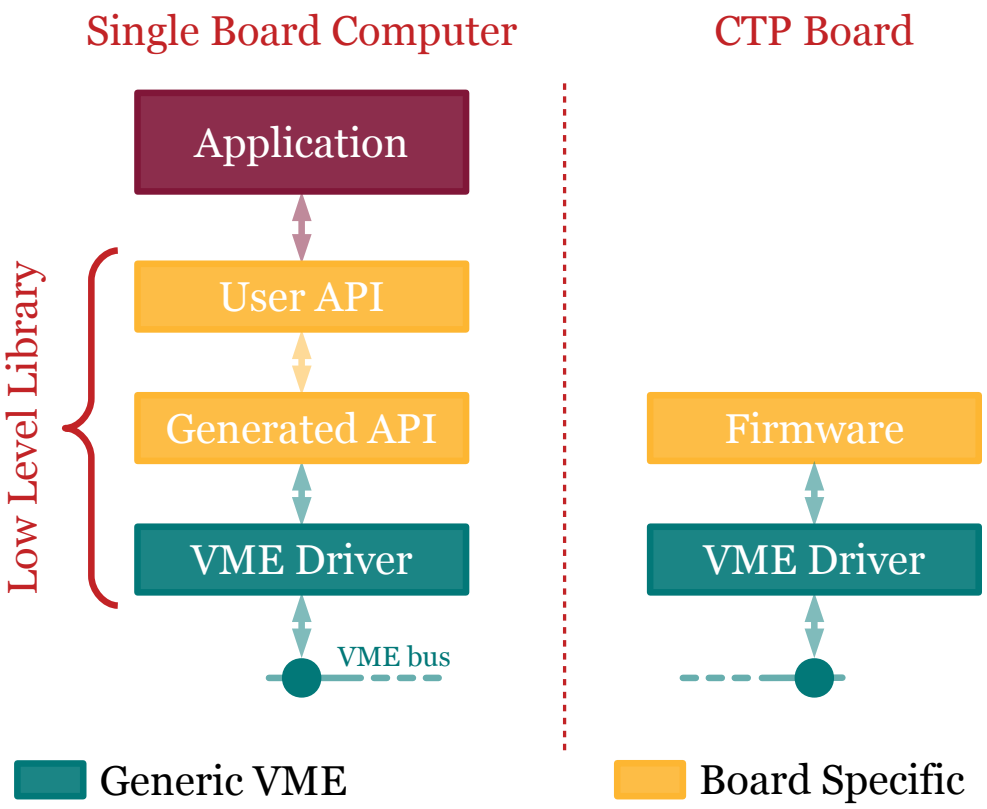


Figure II.2.2: Data flow between user application and CTP board.

II.2.4 CTP Software

Low Level Libraries

The purpose of the low level libraries is to provide an Application Programming Interface (API) for an application to talk to the electronics boards of the CTP. The low level libraries can thus be seen as the device drivers for the boards. The communication between a program and board goes via the low level library to the VME driver, then on to the VME bus, where the firmware on the board obtains the communcations data, and then all the way back up the application. Figure II.2.2 depicts the communica-

tion paths from an application through the different parts of the low level library. Based on an XML description of the registers accessible by the firmware, a simple library is generated around the VME driver to provide basic access to a board. The generated library is too simple for direct use but serves as the foundation for the development of the actual low level library. The low level library implements user-level data structures and methods for interfacing the functionality of the board in a safe and convenient way. It also ensures the order and integrity of execution on the board when changing configuration or state by adhering to board-level protocols. The low level libraries serve as the device drivers and are per design tightly tied to the firmware of the boards; firmware changes often bring with them changes to the low level libraries.

The introduction of the two new boards, the CTPOUT+ and the CTPCORE+, meant the development of two new low level libraries as well as adaption in code for existing applications not part of the high level software that got replaced.

High Level Software

The purpose of the high level software is to provide a way to utilize the CTP resources within a software partition, typically in the ATLAS partition with the rest of the sub-detectors and sub-systems. The tasks of the software, new as well as old, is thus to facilitate the configuration, monitoring and other utilization needed for ATLAS data-taking.

The high level software of the CTP was completely rewritten between Run I and Run II, since the software architecture was redesigned. The

architecture for Run I was simpler and took a board-centric approach. This approach proved to be too simple to accommodate for the potential concurrent use of the new CTP. The architecture for Run II takes a more user-centric approach, but requires a more complicated layout to manage the resources. A short overview of the Run I architecture will be given, followed by a slightly more detailed description of the architecture of Run II.

Run I Architecture

During Run I, the high level software served as a direct extension of the low level software, bridging the gap between the boards and driver, as well as between the desired configuration and functionality. This is outlined in Figure II.2.3. The board-centric nature has advantages and disadvantages – the main advantage being simplicity, on several levels:

1. It is conceptually simple – The high level library takes care of one or more boards all of the same type and is responsible for only that.

2. Implementation, maintenance and debugging is simple – Since the information flow is well defined between the partition and the board, errors and unexpected behaviours are easily traceable.

3. Relatively little ipc and parallel computing is required. This is not immediately obvious, but by staying within *one* application, even with multiple sub-processes, the different parts of the software can use simple synchronisation primitives (mutexes, semaphors and locks) to coordinate and ensure correct functioning of each board.

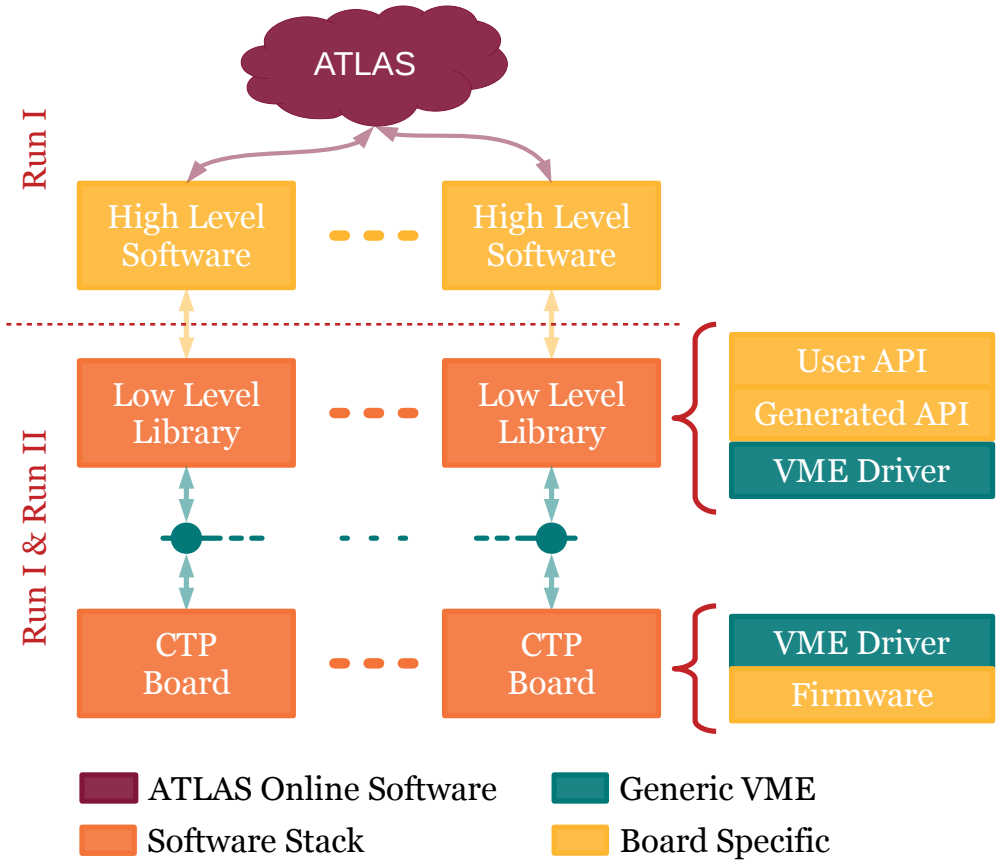


Figure II.2.3: Software layout of the CTP during Run I

The disadvantage of the Run I system is that it keeps all the eggs in one basket: all tasks need to be performed on all boards in one application on one computer. Thus, critical and non-critical tasks are mixed, and should a non-critical task fail, the entire CTP software would crash, interrupting the data taking.

As the Run I approach provides no means of coordination between the users, it cannot be realised with the new CTP, which has multiple users. These users cannot know of each other, as they live in different partitions that might be started and stopped at will, and thus the coordination, allo-

cation and management of the CTP resources can not be handled safely by starting three independent and isolated instances of the same program. While this might not be immediately clear, it turns out that the nail in the coffin for the old architecture is its failure to provide safe concurrent access to the CTP resources. Additional non-trivial constraints on concurrency are

the VME bus which is concurrent in nature;

the layout of the low-level libraries which, in order to adhere to the board-level protocols, assume that they are exclusive in communicating with a given board;

the trigger path , including the readout: There can only be one user of the readout system and the trigger item definition needs to be shared between users.

Lastly, the upgraded CTP boasts more resources on the new boards but the SBC remains the same and was already during Run I considered resource limited. With more data to be read out, processed, and sent around, and with (more) concurrent users, moving computational tasks away from the SBC would be advantageous, if not necessary.

Run II Architecture

Overview: The architecture for Run II cannot be easily depicted in the same way as the Run I architecture, as it is oriented around the functionality rather than the boards.

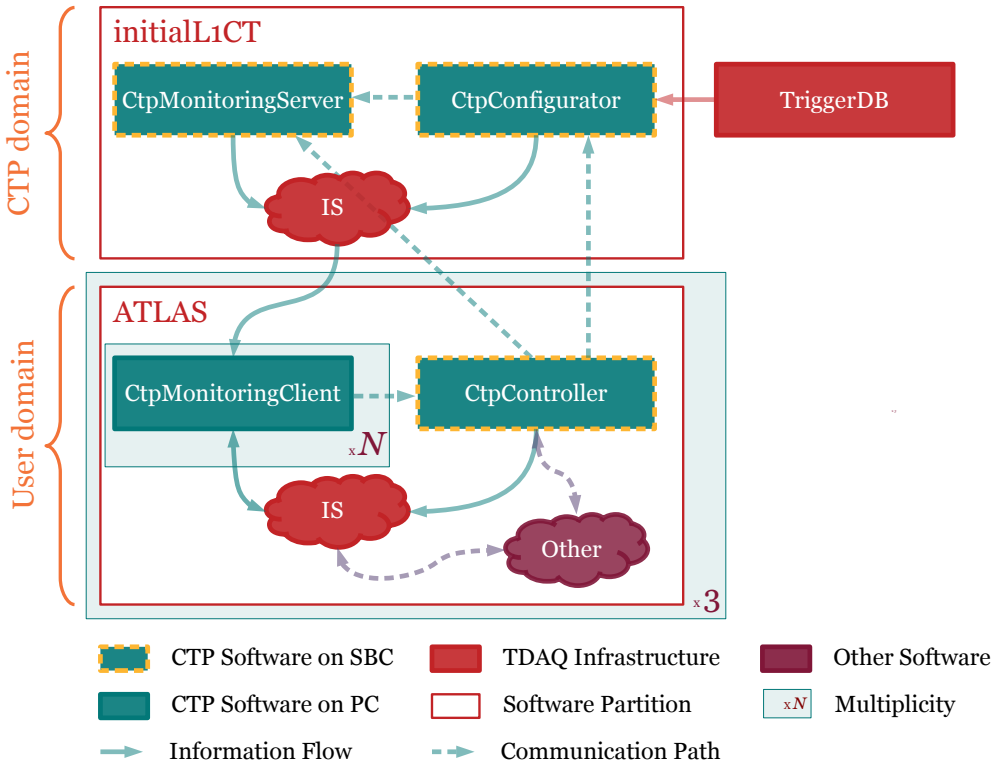


Figure II.2.4: Software layout of the upgraded CTP for Run II

The new architecture uses a partition, which is independent of the rest of ATLAS to perform tasks that can and should be completely decoupled from the user. This includes basic monitoring as well as services for users to request, reserve and configure the CTPs resources. This partition, in principle, has an infinite life time.

Up to three ATLAS-like partitions may use the CTP resources concurrently to facilitate whatever subset of subdetectors or subsystems is included in the respective partition[6]. A depiction of the new CTP software architecture is shown in Figure II.2.4. The picture shows the software

[6]Note that the CTP can provide three partitions, but only one readout, making the readout a resource that requires exclusive ownership.

relevant to the CTP. The software of other systems is included as a purple blob of "Other" software[7].

The new architecture splits what used to be one program into several smaller programs, and it uses IS and the ipc as means of communication between programs and between the CTP partition and the user partitions.

Configuration and Resource Management: The `CtpConfigurator` serves as a resource manager for the user. The user requests a partition, sub-detectors to be included in the partition, trigger inputs, and items to be used. Then, the `CtpConfigurator` handles the reservation and application of necessary configuration which it reads from the trigger database. The `CtpConfigurator` publishes the configuration information to IS where it is available to anyone.

Monitoring: The `CtpMonitoringServer` monitors the CTP hardware and makes the raw data available on IS. The data includes occupancies of buffers, busy-rate counters throughout the system, dead time, trigger item and rates, and the configuration applied to the board. In the user world `CtpMonitoringClients` are responsible for gathering the publications from the `CtpConfigurator` and the `CtpMonitoringServer` and publishing them in a way more meaningful to the user. In this way the user can also reduce the local data quantity by only requesting data that is relevant for the user. Another advantage of this is that the processing of the data is moved away from the SBC, which only has limited resources avail-

[7]This blob usually makes up most of the software in the partition.

able. All publications by the `CtpMonitoringClients` are being stored for later, including

- Prescale configurations;
- Busy rates and dead-time;
- Item rates and counts.

The ipc between the `CtpMonitoringServer` and the `CtpConfigurator` is introduced, since the `CtpConfigurator` needs to reset one or more of the boards for some configuration tasks. To avoid spurious monitoring results, the `CtpConfigurator` notifies the `CtpMonitoringServer` before and after a board reset.

The ipc between `CtpController` and `CtpMonitoringServer` is introduced to steer the different monitoring tasks in favour of the user partition controlling the readout: during operation with full readout, publications get timestamped. In addition, some publications of the `CtpMonitoringServer`, such as the total number of L1As, are important for determining the recorded integrated luminosity correctly. During lumi-block transition (changing from one lumi-block to the next) the triggers are held in order to allow these numbers to be read out exactly. As the `CtpMonitoringServer` is by design unaware of what is ongoing in the users world, the `CtpController` of the primary partition is allowed to temporarily take over the steering of the `CtpMonitoringServer` in order to reduce the data-loss to sub-second level. The override of the normal steering of the `CtpMonitoringServer` also ensures that the VME bandwidth is not being used

to perform expensive monitoring tasks during these time critical transitions.

Utilisation: The request for resources is done by the `CtpController`. When granted by the `CtpConfigurator`, the `CtpController` is further responsible for utilising these and making these available to the rest of the partition. Within the users partition, the `CtpController` is responsible for issuing new lumi-blocks and for enforcing the maximum length of a lumi-block by issuing a new one. The `CtpController` also serves as the interface for applications to request a new lumiblock in the case of a pending configuration change and for requesting changes that in other ways affects the trigger. This could be a prescale changes, which would then be forwarded to the `CtpConfigurator`, but it could also be requests from subsystems to hold the trigger (e.g., while a detector segment is being reset or resynchronised). For holding the trigger either per request, during lumi-block transition, or even during the state transitions of the user partition – the `CtpController` needs access to the CTP hardware. This is strictly not true, as it could technically be elegantly handled via ipc, but as in particular the lumi-block transitions are considered time critical the faster direct access was preferred.

Autonomous feedback: One of the `CtpMonitoringClients` that deserves special mentioning is the `Auto Prescaler`.

In line with the discussion in Chapter I.5.5, almost any subdetector or subsystem problem will affect the trigger rates. By aggregating a large subset of the published monitoring data, it is possible to quickly detect

and subsequently react to such a disturbance by requesting a set of prescales. When the symptomatic behavior disappears, the original prescales will then again be requested.

One example is the electromagnetic calorimeter, which struggles with hot cells at times. When this happens, the trigger rate (before prescale) of both `L1_TAU8_EMPTY` and `L1_TAU8_EMPTY` increase to anywhere between $\mathcal{O}(100\,\text{kHz})$ and $\mathcal{O}(40\,\text{MHz})$. As the expected rate of `L1_TAU8_EMPTY` is $\mathcal{O}(0\,\text{Hz})$ this proves a useful way of detecting hot cells. Upon detection, all affected trigger items can be prescaled accordingly.

The functionality of the `Auto Prescaler` does not currently extend far beyond correcting for such behaviours, but it would be trivial to extend the framework to correct the applied prescales for luminosity drops and to have automatically-generated full prescale sets based on targeted trigger rates. The functionality could be extended outside the domain of the CTP and also used to generate or adapt the HLT prescales in a similar manner. The result would be less error prone by reducing the amount of human involvement and would give rise to more efficient data taking as the turn around time for generating and applying new prescales would be greatly reduced.

Auxiliary Software

In addition to the software infrastructure described above for operating the CTP with ATLAS a number of auxiliary programs exist to monitor and control the CTP. These serve as useful tools for monitoring the CTP during

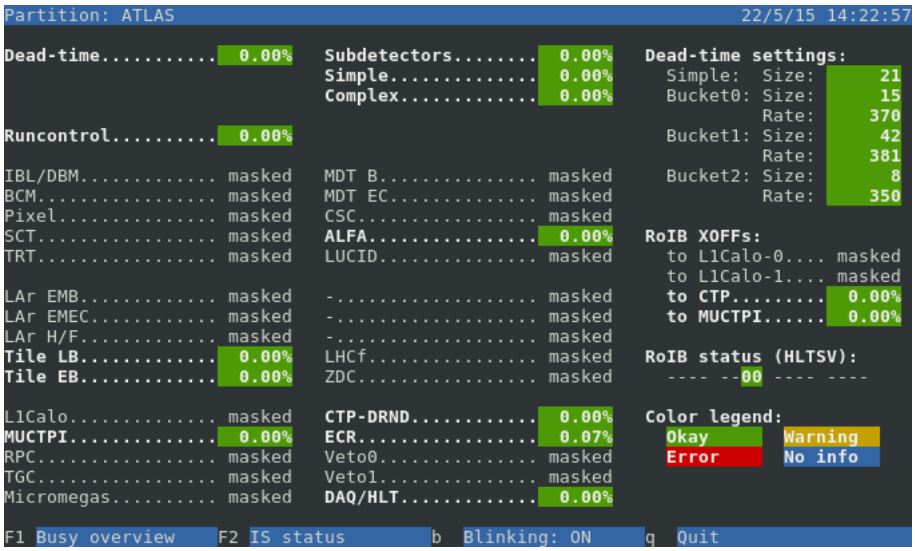


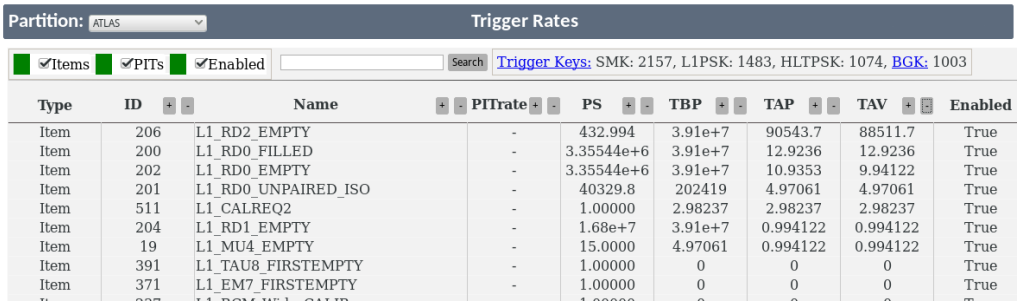
Figure II.2.5: Screen-shot of the terminal application for displaying the busy state of ATLAS.

operation and for making custom configurations of the CTP hardware, either for special runs or for testing the CTP system.

Online Monitoring

The most-used programs and services are those used to present the IS information in an easily interpretable way. Web and terminal applications have been developed for displaying trigger rates as well as for the tracing the dead time and its cause. In the ATLAS Control Room (ACR), the web display of the rates and the terminal display of the busy take up two of the 8 main screens projected onto the wall.

A screen-shot of the terminal application is shown in Figure II.2.5. The raw data from the CTP provided by the `CtpMonitoringServer` is picked up from IS along with useful names for each of the CTPOUT+ cables,



Type	ID	Name	PITrate	PS	TBP	TAP	TAV	Enabled
Item	206	L1_RD2_EMPTY	-	432.994	3.91e+7	90543.7	88511.7	True
Item	200	L1_RD0_FILLED	-	3.35544e+6	3.91e+7	12.9236	12.9236	True
Item	202	L1_RD0_EMPTY	-	3.35544e+6	3.91e+7	10.9353	9.94122	True
Item	201	L1_RD0_UNPAIRED_ISO	-	40329.8	202419	4.97061	4.97061	True
Item	511	L1_CALREQ2	-	1.00000	2.98237	2.98237	2.98237	True
Item	204	L1_RD1_EMPTY	-	1.68e+7	3.91e+7	0.994122	0.994122	True
Item	19	L1_MU4_EMPTY	-	15.0000	4.97061	0.994122	0.994122	True
Item	391	L1_TAU8_FIRSTEMPTY	-	1.00000	0	0	0	True
Item	371	L1_EM7_FIRSTEMPTY	-	1.00000	0	0	0	True

Figure II.2.6: Screen-shot of the terminal application for displaying the busy state of ATLAS.

which are obtained from OKS and published to IS by the `CtpController`. The information is combined to provide real-time depiction of the total dead-time applied (top), preventive dead-time settings (top right), status of RoIB/HLT (middle right) as well as busy coming directly from the various subdetectors or the Run Control application holding the trigger. This provides an efficient and powerful way of identifying the source of any dead-time.

Similarly Figure II.2.6 shows a screen-shot of the web-app for monitoring trigger input and item rates. A monitoring client in the user partition gathers and combines relevant publications – the trigger rates from one place, TIP rates from a second (though both provided by the `CtpMonitoringServer`) and the names from a third – and republishes it to IS in a condensed format. The web application gets access to the IS information via an infrastructure application that provides an HTTP interface for accessing the IS information. The web interface provides quick access to all rates of all items rates before prescale, after prescale, and after veto as well as the PIT rates. Combined with the possibilities to filter,

sort and search this provides a powerful interface for investigating the L1 bandwidth usage and identifying the source of spurious rate or simply ensuring that the targeted rate is reached.

The Menu Programs

The menu programs are small-terminal based applications that serve as a front-end for the low level libraries. Thus these provide direct access to the boards functionality on a low level. As most applications only use a subset of the functionalities of each board, usually in a very application specific manner, the menu programs provides a convenient way of interfacing the full functionality of each board without requiring the development of a new application. The menu programs are useful for testing the boards and verifying their functionality, but also prove useful for cross checking and debugging software that makes use of the low level routines. Most of the CTP boards provide functionality beyond what is needed for normal operation. This includes the ability to monitor signals at various points as well as artificially generate, inject or inhibit signals in different places. As an example, the CTPOUT+ can generate its own orbit signal and it can, per subdetector, route individual incoming and outgoing signals differently and thereby collide two (or three) user partitions. It is also possible to generate single pulses, replay series of signal patterns, or similarly record them. While the above functionality is clearly not for normal operation it provides an incredible flexibility to test the various parts of the CTP using the CTP itself and for performing these kind of manual overrides the menu programs is a powerful tool.

Development of the menu programs for the CTPOUT+ and the CTPCORE+ went hand in hand with the development of their respective low level libraries.

Test Suites

Many of the tests of the CTP system and of its connections to sub detectors can be automated, and a series of programs have been created that apply a specific configuration to the relevant part of the systems, and systematically tests that any combination of generated signals produce the desired output. These programs are most often used to test new firmware but are also useful for identifying possible electrical problems such as small hardware faults on the boards, a loose connection in a cable, or a damaged connector. For the CTPOUT+ such a test suite was created to verify the integrity of each board as they arrived from production.

Most of these tests suites don't require a cabling different from the one used in the ATLAS cavern and thus allow for in-situ testing. A few of these suites can, at least theoretically, be run parasitically without affecting data taking.

The Capture Programs

Several of the boards include monitoring capabilities that extend beyond what is needed during normal operations. These include the CTPOUT+ boards that have the capability to use an on-board memory to record all the received signals as well as the CTPCORE+ with its extended capabilities for event monitoring. The CTPINs too have capabilities for recording

the incoming signals. This has lead to the development of a handful of programs that can be used for recording and dumping such data as well as for comparing the recorded data against a reference.

Chapter II.3

Concluding Part II

The CTP was upgraded to remove some of the resource limitations and to accommodate new sub-detectors, in particular the L1Topo trigger detector. This required the complete replacement of the CTPCORE, the COM backplane and all the CTPOUT modules. The CTPINs and the CTPMON only required a firmware update.

The new requirement for sharing the resources of the CTP required a change of the CTP software architecture.

My contribution to the upgrade of the CTP has been the development of the low level library and software for controlling, testing and operating the CTPOUT+ modules. I have further contributed actively to the redesign and implementation of CTP software, both through direct involvement and through supervision.

With the successful commissioning, testing, and installation of the upgraded CTP, and with the other upgrades not discussed here, ATLAS is ready for Run II data-taking.

Part III

The Data Preservation Problem

Introduction

The trigger system is evolving constantly: algorithms and selection criteria change over time and several major changes, such as those made between Run I and Run II, can be foreseen. Due to the tight coupling between the recorded data and the trigger selection used, the effective lifetime of the recorded data is determined by the ability to preserve knowledge of past trigger systems, and the ability to precisely simulate their selection. Having a viable strategy for how to preserve this information on timescales of more than 20 years is important for the ATLAS experiment.

The aim of the study presented here was to formulate a suitable strategy and determine the immediate challenges for its implementation. The results of the study were presented at the two conferences Computing in High Energy Physics (CHEP) and Advanced Computing and Analysis Techniques in Physics Research (ACAT), and they have since been adopted by ATLAS. The discussion here parallels the ones in the two corresponding conference proceedings [32] and [33]. These will be used as source without citation.

The study in its entirety reflects my own work.

Chapter III.1

Motivation

III.1.1 A Time of Change

Changes to the trigger system affect how data is selected for recording. Analysing and understanding the recorded data imply analysing and understanding the recorded trigger selection. The ability to accurately simulate and study the trigger selection used for recording a data sample is paramount to its usability. This implies that the understanding of the trigger system as well as its selection must be kept operational for the lifetime of the recorded data.

The trigger system is expected to evolve over the lifetime of the experiment, both continuously and in steps. During Run I, new trigger selection algorithms were introduced, some where modified and others again where removed. Between Run I and Run II several upgrades were made to ATLAS and its Trigger and Data Acquisition System, most significantly:

- The merging of the Level 2 trigger and the Event Filter into a common High Level Trigger (HLT) ;
- The upgrade of the Central Trigger Processor ;
- The introduction of the topological trigger processor, L1Topo ;
- The introduction of new detectors.

The trigger is expected to evolve throughout Run II and to receive another major upgrade between Run II and Run III.

The amount of knowledge about old and obsoleted trigger systems that needs to be preserved and kept operational will grow over time. Having a strategy for preserving this information without requiring an increasing amount of effort is essential.

The ATLAS trigger is more than the parts it is made of: Over timescales of decades computers, operating systems and compilers we rely on in ATLAS is expected to change too. Over the last couple of years there was a transition from 32 bit to 64 bit computers and a change of operating system to Scientific Linux 6. The future is expected to bring more change.

A strategy for preserving the knowledge of the trigger and the ability to simulate its selection must also take into account such changes to the environment. Throughout Run I it became increasingly evident that the chosen strategy was inadequate and that a new strategy was needed for future data taking and if the knowledge of the Run I trigger system was to be preserved.

Figure III.1.1: Depiction of ATLAS simulation chain.

III.1.2 Monte Carlo and Data Analysis

The need for preserving methods for precise trigger simulation in the future is motivated by the need of many analyses to have data accompanied with corresponding amounts of simulated Monte Carlo (MC) data. Normally MC is generated alongside the data taking using the same software as is used for data taking for trigger selection and reconstruction.

There are four steps to the process of generating MC data for analysis, in analogy with the steps of data taking, namely:

1. Event Generation – simulation of the collisions and underlying physics processes ;
2. Detector Response – simulation of ATLAS response to the generated events ;
3. Trigger Response – simulation of ATLAS trigger decision ;
4. Reconstruction – events that pass trigger selection get stored and reconstructed.

For MC production ATLAS uses non-ATLAS software for the event generation step. The remaining three steps – depicted in Figure III.1.1

– are referred to as *the ATLAS simulation chain*, or simply *the simulation chain*. Each step is executed sequentially, and the output of one step is used as the input for the next step. The output of the simulation chain is the input of the detector simulation and this output is the output of the reconstruction. Other intermediate files are deleted.

The software of the simulation chain is packed in to special MC production releases. The actual MC production is usually carried out on the LHC Computing Grid[22], which will be referred to as simply the grid from here on out. The actual execution of the simulation chain is often split in two jobs: detector simulation as one job, and trigger simulation and reconstruction in another.

Chapter III.2

Re-running the Trigger Simulation

III.2.1 Motivation

Improvements to any of the four steps of generating MC will require the trigger response to be (re-)simulated. Examples are:

1. Introduction of new or better event generators – new physics processes might need to be simulated for the entire ATLAS data set ;
2. Improved description of the detector geometry or an improved detector response simulation[1] – implies a different input for the trigger and requires the trigger response simulation to be rerun ;

[1]There is a subtle but important difference between the actual simulation of the response and in the description of the detector used for the simulation

3. New or improved trigger selection algorithms – for trigger specific studies or for new physics processes, the trigger response might need to be (re-)simulated ;

4. Improved off-line reconstruction – could motivate a reprocessing of the entire data set (potentially spanning several years and run periods) with a homogeneous reconstruction setup. Improvements to the detector simulations would likely require the re-simulation of the detector response for the corresponding MC sample and with that a re-simulation of the trigger response.

As the understanding of detector response and of systematics improve over time, more precise and detailed simulation of the trigger response will be needed for the different analyses. This could motivate increasing the size of the MC samples for each periods to reduce uncertainties from limited statistics.

III.2.2 Assessment of Possible Strategies

A number of strategies were considered for retrospective trigger response simulation, based on the precision offered and the amount of effort required over time.

Porting Old Trigger Code

During Run I old trigger selection algorithms was continuously ported (i.e., updated and adapted) to comply with changes in the trigger simulation software. Old selection algorithms and criteria were kept operational

along with the new ones. This introduced additional code and configuration complexity in order to resolve name clashes and other ambiguities.

On top of the immense and ever growing effort needed to update and maintain old and unused selection code, this strategy requires a framework to test and ensure that the selection of the ported selection algorithms remains the same. Maintaining this framework also requires a continued effort.

Run II makes the Level 2 and the Event Filter of Run I obsolete. There is no trivial way of porting the old Level 2 and Event Filter algorithms of Run I so that they can co-exist with the HLT algorithms of Run II. Doing so would require an immense and ever growing effort.

By the end of 2013 this strategy was abandoned as it had become too difficult to maintain the legacy selection code.

Parameterisation of the Trigger Response

A strategy requiring less effort would be to approximate the trigger selection, by parameterizing the turn-on curves that describe the triggers efficiency against different variables, typically p_{T} threshold and η. This method would suffice in most cases but for rare or new physics processes it would not suffice. Maintaining a description of correlations between trigger selection algorithms would also be difficult.

This strategy was not pursued as the required precision could not be achieved.

Running Legacy Trigger Code

Running the legacy trigger code “as is” using an old simulation release would allow for an exact re-simulation of the trigger response with close to no maintenance of code required, and no need for resolving conflicts between old and new systems. This strategy shifts the effort of preserving the trigger simulation from maintenance of selection algorithms to a more technical problem of preserving the old releases and the necessary execution environment (computer architecture, operating system, etc.).

III.2.3 Immediate Challenges

Running legacy trigger selection code within the simulation chain poses a number of challenges: The simulation chain uses a homogeneous version between all steps and a volatile data format for intermediate files. While it is paramount that the trigger selection matches the selection used during data taking, it is often advantageous to use the latest or best versions available for the detector response simulation and for the reconstruction step. This breaks the design of the existing simulation chain and is likely to result in issues with data format compatibility due to the rapidly changing data format used as for the intermediate files. Changing release implies changing the execution environment. Changing the execution environment between two steps might mean changing operating system, or perhaps in the future, even computer architecture. These legacy environments needs to be preserved and kept operational too.

The mentioned challenges needs to be handled gracefully by the proposed strategy, and preferably with a minimal change to existing infrastructure.

Chapter III.3

Running Legacy Trigger Selection Code

The immediate problems relating to data format compatibilities can be summarised as:

1. A forward compatibility issue: The *old* trigger simulation needs to read input files produced by the *new* detector simulation ;
2. A backwards compatibility issue: The *new* reconstruction step needs to read input files produced by the *old* trigger simulation.

The immediate problems relating preservation of the execution environment can be summarised as:

1. Preserving Context: The long term preservation of everything needed to run legacy trigger selection code ;

2. Switching Context: The challenge of changing between versions of software (and operating systems) between different steps of in the simulation chain.

III.3.1 On Data Formats

The Raw Data Object Format

The data format used through out the simulation chain for intermediate files is the Raw Data Object (RDO) format[23]. The format is a container format based on the ROOT[9] – the de-facto analysis and histogramming framework used in High Energy Physics. The RDO contains serialised objects, instances of classes, and meta-data. At each step of the simulation chain relevant objects and meta data is read and analysed, and new object are added to the RDO.

The trigger simulation step receives an RDO file from the detector simulation containing:

- Data objects describing the detector response ;
- Data objects describing the “true” particles from the event generation step ;
- Meta-data covering key parameters for the simulation at each step.

The trigger simulation only needs the detector data to run as well as a few trigger specific parameters from the meta data used to determine the

selection algorithms to run and their prescales. The MC truth information is only used at the reconstruction step where both the truth particles and the events passing the trigger simulation are reconstructed from the detector information.

The format guarantees neither forwards nor backwards compatibility. The complex nature of the data format, the rapid evolution of the structure of data inside the container, and the evolution of the serialised classes that is the actual container payload, makes it difficult to provide conversion between versions of RDO files. An attempt at reading and writing RDO using different software releases showed incompatibilities between releases separated by no more than $\mathcal{O}(1/2\,\mathrm{y})$. Attempts at providing conversion steps proved unsuccessful.

ATLAS Detector Data Format

The ATLAS detector has a native data format for detector data: the *byte stream format*[16]. The byte stream format is a chunk based data format that allow versioning of both the format itself and of the various payload chunks. Figure III.3.1 show a simplified version of the format: The file contains a header specifying the format version and the size of the container. Knowing the format version, the header of each chunk and be deciphered. The header of each chunk contains as a minimum information of the type of chunk (unique for each subdetector), as well as the size and version of the payload of the chunk. The payload, containing detector data, can then be decoded knowing the version.

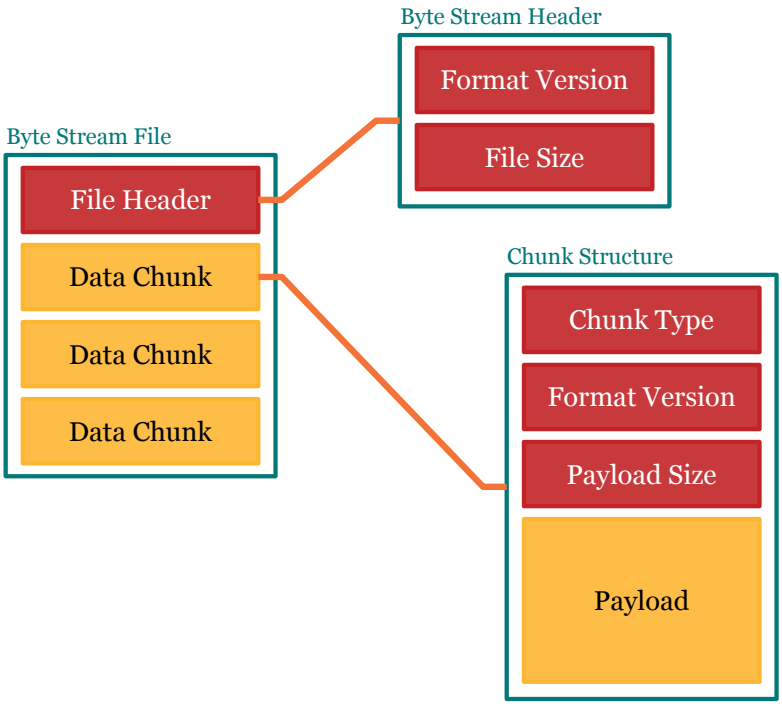


Figure III.3.1: Simplified data structure of the byte stream format

It is a requirement from the ATLAS side that all byte stream data remain readable, ensuring backwards compatibility. No forwards compatibility is guaranteed though, but the format can be expected to evolve slowly due to its ties to the detector readout. The code for writing older versions of the byte steam format already exist, though it is not necessarily kept operational. Some detectors including the L1Calo already support writing payload of different versions. Imposing forwards compatibility as a requirement in the future would be a possible option and require a minimal effort as both the data format as well as the data payload is usually quite simple.

For the trigger simulation step, the byte stream format is both supported and in many ways preferred: upon receiving an RDO, the objects

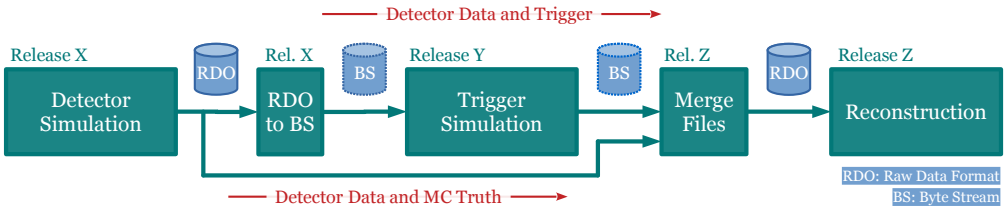


Figure III.3.2: Depiction of the simulation chain, modified to accommodate use of a different software release at each step.

describing the detector data are first converted to byte stream before the trigger simulation is run, properly simulating the conditions during data taking.

The downside to the byte stream format is that it does not support either truth information or meta data. This information still needs to be passed on to the reconstruction step. While one could tamper with the format and force support for truth information and payload – for instance by adding it to a data chunk with an undefined subdetector type – it was discouraged for a number of reasons and, as it turns out, not necessary.

Altering the Simulation Chain

To accommodate the issues with compatibilities between data formats the simulation chain was modified. The proposed simulation chain is shown in figure in Figure III.3.2.

The trigger simulation step only needs the detector data from the input RDO file. The relevant trigger specific parameters can be extracted from the file before the execution of the trigger simulation and passed to the simulation step by other means, e.g., as command line arguments or in a

separate file with a data format more stable than RDO. In the proposed simulation chain, the output of the detector simulation is converted to byte stream and used as input for the trigger simulation. The original RDO file from the detector simulation is kept while the trigger simulation step is executed.

After the trigger simulation, an output file will have been created containing the detector data as well as trigger response record, but without the truth information or the meta data. The information contained in the two files would have to be merged for the reconstruction to work. The trigger simulation step can however be modified to produce a byte stream file containing the trigger record as the output. From the requirement of backwards compatibility this file would be readable in any future release.

The merging step reads the trigger response record from the byte stream file and writes it the RDO from the detector simulation, and the resulting RDO is used for the reconstruction.

III.3.2 Virtualization and Context Preservation

Introduction to Virtualization

Virtualization is an area of computing where software is used to emulate hardware. Modern technology allow for entire computers to be simulated – even several at a time – on computers with resources comparable to a modern laptop. This is done by exploiting similarities in the architecture

of the *host* computer and the *guest* systems and thus reducing overhead. Simulated computer systems are often referred to as *Virtual Computers* or as *Virtual Machines* (VMs).

The software on the host system that is used for defining, simulating and managing the VMs is called a *hypervisor*. The Kernel Virtual Machine[38] (KVM) is a part of the Linux kernel and virtualization is natively supported by modern Linux computers.

A VM is first defined within the hypervisor, including the architecture, available memory, storage, and network resources, etc. Subsequently the VM can be started, and the following steps resemble those of a normal computer: An operating system is installed to a (virtual) hard drive using a (virtual) USB device or a (virtual) CD. Additional software can be installed and configured in ways dictated by the operating system used. The VM can generally be used, shutdown, rebooted, or even hibernated in ways similar to a normal computer.

The process of installing and configuring software on a VM is called *contextualization*. A handful of methods for automatic contextualization have been developed, but support for most methods relies on support from the operating system used on the VM.

CERN Virtual Machine Project

CERN has a virtual machine project called *CernVM*[19]. The project is actively maintained, and it aims at aiding the experiments in their data preservation effort. The various grid sites use pre-contextualized CernVM

instances as computing nodes, providing a homogeneous setup across all data centers.

As part of the CernVM project, a HTTP based file format called *CernVM File System*[19] (cvmfs) has been developed. The file system exerts a number of features important for data preservation: the file system has strict global version control, all files are read only (and write-once[1]). The file system is easily distributed and can be locally cached, as no changes can be made to existing files.

The CernVM disk images has a size of $\mathcal{O}(100\,\mathrm{MB})$ which is small compared to normal VM disk images of $\mathcal{O}(10\,\mathrm{GB})$. The CernVM image contains a very small Linux installation that provides an interface for contextualization of the VM from software stored on cvmfs. During this process, the initial disk image is expanded and often the original content of the disk image is wiped. The CernVM group already offers all versions of Scientific Linux as well as relevant libraries and compilers on cvmfs. The ATLAS Experiment has started publishing selected releases on cvmfs as part of the effort for long term data preservation. A full contextualization of a CernVM image to the point where the trigger simulation software can be executed is thus technically possible.

Preserving the Context

Besides information about the release software to use the information about the CernVM itself, its definition as well as its contextualization, need to be preserved.

[1] Over-writing is done creating a new version.

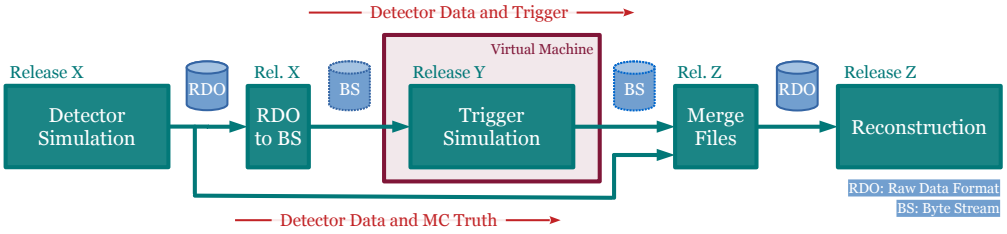


Figure III.3.3: Depiction of a simulation chain that utilizes virtualization technologies for execution of the trigger simulation step.

Over time, the host architecture and hypervisor technology might change, and thus a hypervisor-independent way format for storing VM definitions is needed. The libvirt[40] project provides a common Application Programming Interface (API) for most mainstream hypervisors and uses an XML based description of VMs.

The CernVM provides several methods for describing the contextualization of a CernVM instance, ranging from simple configuration files to a web-interface that uses cvmfs to create, store and retrieve a certain CernVM contextualization.

The amount of additional information that ATLAS needs to be preserve amounts to a couple of text of at most $\mathcal{O}\,(100\,\mathrm{kB})$ containing the VM definition and the CernVM contextualization.

Altering the Simulation Chain

A depiction of a modified simulation chain that could accommodate both the changes in data format evolution and the use of virtualization is shown in Figure III.3.3. In this setup, following the conversion of the RDO to

byte stream, a contextualization to support the desired version the trigger simulation would be started or created. The trigger simulation would then be executed within the VM to produce the byte stream file containing the trigger response. After this, the VM is no longer needed and can be disposed off. The remaining merge of data files and subsequent execution of the reconstruction step can then be realised using the desired release.

However, this depiction is deceptive, as a VM doesn't act as script or as an executable but as a computer. While the desired scenario is exactly as depicted, it does not answer simple questions such as how the input files gets into the VM, how the trigger simulation gets started within the VM, or how the output file gets out of the VM.

Moving files in and out of the VM turns out to be a smaller problem. Linux has a near native support for various network files systems that could be used for temporary storage of the various files used along the simulation chain. Most hypervisors also support bridging between the host file systems and the quests running on the host.

The means for operating a VM from the outside is limited and the naive approach of using a VM as a (disposable) wrapper for the execution of a program is potentially a novel use case of virtualization technology. As will be discussed later there are several reasons why are further investigation into the implementation of virtualization should be studied further.

III.3.3 Proof of Concept Solution

A proof of concept (poc) setup was created for running the modified simulation chain in Figure III.3.3 using an on-demand created VM for exe-

cution of the trigger simulation step.

In this setup the original trigger simulation step was replaced by a script that would define and contextualize an appropriate VM, execute the trigger simulation and afterwards dispose of the VM. The implementation uses KVM through libvirt for creating a CernVM instance and a method for contextualisation called HEPPIX. This contextualization method uses a CD image to store the CernVM contextualization information and additionally allows for files on the CD to be installed to the VM as part of the contextualization. The intended use of the later feature, is for the grid sites to adapt the CernVM images to local site policies.

The CD image used for contextualization was generated on the fly by the script and was used to hijack the execution flow of the VM after contextualization by overwriting relevant system files. Parameters about the software release to use for the trigger simulation, name and location of input files, and the desired trigger configuration were passed as command line options to the script. From the script it was passed on to the VM through the generated CD image.

A network file system available to the VM was used as storage area for the tests.

Chapter III.4

Discussion and Future Perspectives

III.4.1 The Russian Doll

The MC production jobs are normally executed on GRID sites, which use VM instances as computing nodes. The proof of concept setup would require a virtual machine to be started within a virtual machine leading to a highly undesirable “Russian Doll” scenario. Until recently, nested virtualization was not possible, but experimental support is slowly emerging[14]. Whether this will be a viable solution in the future is unclear, as virtualization (at each layer), no matter how well supported by the architecture, introduces an overhead.

From the point of view of the trigger simulation it is unimportant where the created VM is actually located as long as it can be controlled by the virtual machine. If, for instance, a method existed for talking to the host

systems hypervisor from within the guest system, it would be possible to request create a VM in parallel to the primary one. While this is not possible for a number of reasons it might be possible to realise a similar setup by other means and thereby avoid the Russian Doll.

III.4.2 Contextualization of CernVM

A more graceful way of getting the VM to execute the trigger simulation should be found.

Creating and contextualizing the VM to serve as a worker for a job queue might a viable solution for the future. The VM could be created before execution of the simulation chain. Then, the information about what release to use, the location of data files and the desired trigger configuration could then be passed to the VM as part of a job script submitted to the queue. This could be generalised to all steps in the simulation chain: Three different VMs with three different job queues and access to a common storage area for intermediate file. The user would then submit a description of how to run the detector simulation to the first queue. The corresponding VM would receive the job, execute it, and submit a job to the next queue of the trigger simulation, and so on. This would additionally offer a better distribution of workload. The reconstruction step is by far the most time consuming step in the simulation chain, thus allocating more reconstruction workers would be advantageous.

If grid sites offered the possibility of requesting a job to be executed on a CernVM with a user defined contextualization, it could work. For valid concerns about security, user (re-)contextualization is not supported.

III.4.3 Moving Forward

The proposed strategy for a retrospective trigger simulation with the potential use of virtualization technology for the trigger simulation step has been accepted as the current viable strategy for trigger simulation in ATLAS. The necessary hooks to allow change of release between steps have been implemented, and the conversion steps have been integrated into the simulation chain.

A viable implementation using virtualization for running the trigger simulation step is still under investigation. So far virtualization is not needed as even the oldest releases used for data taking can still be executed on the same platform as our newest release. However, this is bound to change.

Chapter III.5

Concluding Part III

A long term strategy for simulation of ATLAS trigger using legacy trigger selection code has been proposed. Different methods for the implementation of the strategy using existing technologies were studied, and a proof of concept implementation was created. The proof of concept implementation demonstrates that the proposed strategy can be realised, and it highlights a couple of challenges and pitfalls of using virtualization. The use of virtualization technologies will first be needed in the future when old code can no longer be natively supported. The proposed strategy has been accepted by ATLAS and is currently implemented without virtualization. Investigation into the integration of virtualization is ongoing.

Bibliography

[1] R. Aaij et al. “Determination of the $X(3872)$ Meson Quantum Numbers”. In: *Phys. Rev. Lett.* 110 (22 May 2013), p. 222001. doi: 10.1103/PhysRevLett.110.222001. url: http://link.aps.org/doi/10.1103/PhysRevLett.110.222001.

[2] M.G. Aartsen et al. “Measurement of Atmospheric Neutrino Oscillations with IceCube”. In: *Phys.Rev.Lett.* 111.8 (2013), p. 081801. doi: 10.1103/PhysRevLett.111.081801. arXiv: 1305.3909 [hep-ex].

[3] R Achenbach et al. “The ATLAS Level-1 Calorimeter Trigger”. In: *Journal of Instrumentation* 3.03 (2008), P03001. url: http://stacks.iop.org/1748-0221/3/i=03/a=P03001.

[4] P.A.R. Ade et al. “Planck 2015 results. XIII. Cosmological parameters”. In: (2015). arXiv: 1502.01589 [astro-ph.CO].

[5] P.A.R. Ade et al. “Planck 2015 results. XIV. Dark energy and modified gravity”. In: (2015). arXiv: 1502.01590 [astro-ph.CO].

[6] N. Agafonova et al. “Observation of a first ν_τ candidate in the OPERA experiment in the CNGS beam”. In: *Phys.Lett.* B691 (2010), pp. 138–145. doi: 10.1016/j.physletb.2010.06.022. arXiv: 1006.1623 [hep-ex].

[7] A. Airapetian et al. *ATLAS detector and physics performance: Technical Design Report, 1*. Technical Design Report ATLAS. Electronic version not available. Geneva: CERN, 1999. url: https://cds.cern.ch/record/391176.

[8] G Anders et al. *The Upgrade of the ATLAS Level-1 Central Trigger Processor*. Tech. rep. ATL-DAQ-PROC-2012-054. Geneva: CERN, Oct. 2012. url: https://cds.cern.ch/record/1490386.

[9] I. Antcheva et al. "ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization". In: *Comput.Phys.Commun.* 180 (2009), pp. 2499–2512. doi: 10.1016/j.cpc.2009.08.005.

[10] S Ask et al. "The ATLAS central level-1 trigger logic and TTC system". In: *Journal of Instrumentation* 3.08 (2008), P08002. url: http://stacks.iop.org/1748-0221/3/i=08/a=P08002.

[11] ATLAS. *ATLAS Experiment Public Results - Trigger Operations Public Results*. url: https://twiki.cern.ch/twiki/bin/view/AtlasPublic/TriggerOperationPublicResults.

[12] ATLAS. *ATLAS Experiment Public Results - Trigger Public Results*. url: https://twiki.cern.ch/twiki/bin/view/AtlasPublic/TriggerPublicResults.

[13] Varouzhan Baluni. "CP-nonconserving effects in quantum chromodynamics". In: *Phys. Rev. D* 19 (7 Apr. 1979), pp. 2227–2230. doi: 10.1103/PhysRevD.19.2227. url: http://link.aps.org/doi/10.1103/PhysRevD.19.2227.

[14] Yang Z Zhang Bandan Das and Jan Kiszka. "Nested Virtualization – State of the art and future directions". url: http://www.linux-kvm.org/images/3/33/02x03-NestedVirtualization.pdf.

[15] R (SLAC) Bartoldus et al. *Technical Design Report for the Phase-I Upgrade of the ATLAS TDAQ System*. Tech. rep. CERN-LHCC-2013-018. ATLAS-TDR-023. Final version presented to December 2013 LHCC. Geneva: CERN, Sept. 2013. url: https://cds.cern.ch/record/1602235.

[16] C.P. Bee et al. "The raw event format in the ATLAS Trigger & DAQ". In: (1998).

[17] Gianfranco Bertone, Dan Hooper, and Joseph Silk. "Particle dark matter: Evidence, candidates and constraints". In: *Phys.Rept.* 405 (2005), pp. 279–390. doi: 10.1016/j.physrep.2004.08.031. arXiv: hep-ph/0404175 [hep-ph].

[18] R. Bruce et al. "Baseline LHC machine parameters and configuration of the 2015 proton run". In: (2014). arXiv: 1410.5990 [physics.acc-ph].

[19] P Buncic et al. "CernVM – a virtual software appliance for LHC applications". In: *Journal of Physics: Conference Series* 219.4 (2010), p. 042003. url: http://stacks.iop.org/1742-6596/219/i=4/a=042003.

[20] M Capeans et al. *ATLAS Insertable B-Layer Technical Design Report*. Tech. rep. CERN-LHCC-2010-013. ATLAS-TDR-19. Geneva: CERN, Sept. 2010. url: http://cds.cern.ch/record/1291633.

[21] CERN. *The CERN accelerator complex*. url: http://te-epc-lpc.web.cern.ch/te-epc-lpc/machines/pagesources/Cern-Accelerator-Complex.jpg.

[22] CERN. *The Worldwide LHC Computing Grid*. url: http://home.web.cern.ch/about/computing/worldwide-lhc-computing-grid.

[23] CERN/ATLAS. *POOL - Persistency Framework*. url: http://pool.cern.ch.

[24] M Červ. "The ATLAS Diamond Beam Monitor". In: *Journal of Instrumentation* 9.02 (2014), p. C02026. url: http://stacks.iop.org/1748-0221/9/i=02/a=C02026.

[25] V Cindro et al. "The ATLAS Beam Conditions Monitor". In: *Journal of Instrumentation* 3.02 (2008), P02004. url: http://stacks.iop.org/1748-0221/3/i=02/a=P02004.

[26] The ATLAS TRT collaboration et al. "The ATLAS Transition Radiation Tracker (TRT) proportional drift tube: design and performance". In: *Journal of Instrumentation* 3.02 (2008), P02013. url: http://stacks.iop.org/1748-0221/3/i=02/a=P02013.

[27] The ATLAS Collaboration et al. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3.08 (2008), S08003. url: http://stacks.iop.org/1748-0221/3/i=08/a=S08003.

[28] CERN Beam Department. “Longer term LHC schedule”. The outline LHC schedule out to 2035 presented by Frederick Bordry and approved by CERN management and LHC experiments spokespersons and technical coordinators on Monday 2nd December 2013. url: http://lhc-commissioning.web.cern.ch/lhc-commissioning/schedule/LHC-long-term.htm.

[29] Marco Drewes. “The Phenomenology of Right Handed Neutrinos”. In: *Int.J.Mod.Phys.* E22 (2013), p. 1330019. doi: 10.1142/S0218301313300191. arXiv: 1303.6912 [hep-ph].

[30] Albert Einstein. “LENS-LIKE ACTION OF A STAR BY THE DEVIATION OF LIGHT IN THE GRAVITATIONAL FIELD”. In: *Science* 84.2188 (1936), pp. 506–507. doi: 10.1126/science.84.2188.506. eprint: http://www.sciencemag.org/content/84/2188/506.full.pdf. url: http://www.sciencemag.org/content/84/2188/506.short.

[31] Lyndon Evans and Philip Bryant. “LHC Machine”. In: *Journal of Instrumentation* 3.08 (2008), S08001. url: http://stacks.iop.org/1748-0221/3/i=08/a=S08001.

[32] Gorm Galster, Joerg Stelzer, and Werner Wiedenmann. “A new Scheme for ATLAS Trigger Simulation using Legacy Code”. In: *Journal of Physics: Conference Series* 513.2 (2014), p. 022039. url: http://stacks.iop.org/1742-6596/513/i=2/a=022039.

[33] Gorm Galster, Joerg Stelzer, and Werner Wiedenmann. "ATLAS trigger simulation with legacy code using virtualization techniques". In: *Journal of Physics: Conference Series* 523.1 (2014), p. 012031. url: http://stacks.iop.org/1742-6596/523/i=1/a=012031.

[34] David Griffiths. *Introduction to Elementary Particles*. Wiley, 2008.

[35] 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, May 2012.

[36] R. Jones et al. "The OKS persistent in-memory object manager". In: *IEEE Trans.Nucl.Sci.* 45 (1998), pp. 1958–1964. doi: 10.1109/23.710971.

[37] M. Kamionkowski. "WIMP and axion dark matter". In: (1997). arXiv: hep-ph/9710467 [hep-ph].

[38] *KVM – Kernel Virtual Machine*. url: http://www.linux-kvm.org/page/Main_Page.

[39] Hung-Liang Lai et al. "New parton distributions for collider physics". In: *Phys.Rev.* D82 (2010), p. 074024. doi: 10.1103/PhysRevD.82.074024. arXiv: 1007.2241 [hep-ph].

[40] *libvirt: The virtualization API*. url: https://libvirt.org/.

[41] K.A. Olive et al. "Review of Particle Physics". In: *Chin.Phys.* C38 (2014), p. 090001. doi: 10.1088/1674-1137/38/9/090001.

[42] Roberto D Peccei. *CP violation: a theoretical review*. Tech. rep. hep-ph/9508389. UCLA-95-TEP-27. Los Angeles, CA: Calif. Univ. Los Angeles, Aug. 1995.

[43] Matts Roos. "Dark Matter: The evidence from astronomy, astrophysics and cosmology". In: *arXiv preprint arXiv:1001.0316* (2010).

[44] Tony Rothman and Stephen Boughn. "Can gravitons be detected?" In: *Found.Phys.* 36 (2006), pp. 1801–1825. doi: 10.1007/s10701-006-9081-9. arXiv: gr-qc/0601043 [gr-qc].

[45] Alexandru D Sicoe et al. "A persistent back-end for the ATLAS TDAQ online information service (P-BEAST)". In: *Journal of Physics: Conference Series* 368.1 (2012), p. 012002. url: http://stacks.iop.org/1742-6596/368/i=1/a=012002.

[46] Eduard Simioni. "The Topological Processor for the future ATLAS Level-1 Trigger: from design to commissioning". In: (2014). arXiv: 1406.4316 [physics.ins-det].

[47] E Simioni et al. *Topological and Central Trigger Processor for 2014 LHC luminosities*. Tech. rep. ATL-DAQ-PROC-2012-041. Geneva: CERN, July 2012. url: https://cds.cern.ch/record/1459209.

[48] Y. Unno. "ATLAS silicon microstrip detector system (SCT)". In: *Nucl.Instrum.Meth.* A511 (2003), pp. 58–63. doi: 10.1016/S0168-9002(03)01751-0.

[49] G. Weiglein et al. "Physics interplay of the LHC and the ILC". In: *Phys.Rept.* 426 (2006), pp. 47–358. doi: 10.1016/j.physrep.2005.12.003. arXiv: hep-ph/0410364 [hep-ph].

[50] Dieter Zeppenfeld. “Event generation and partonshower”. PITP 2005. url: https://www.sns.ias.edu/pitp2/2005files/Lecture%20notes/Zeppenfeld-3.pdf.

Appendix A

Appendix

A.1 Additional Luminosity Plots

A.2 Higgs at LHC

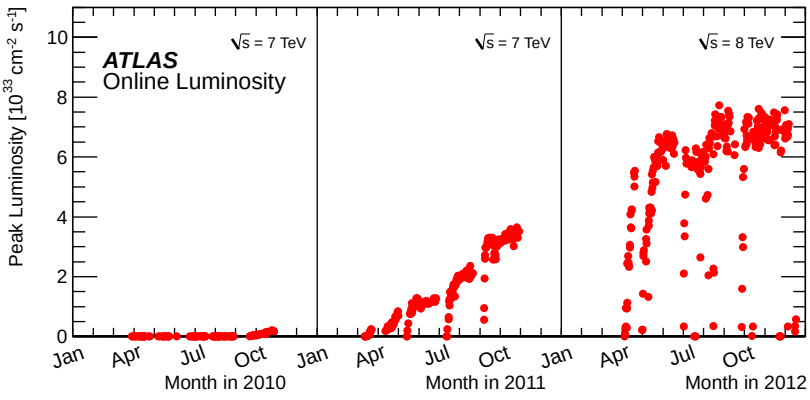


Figure A.1: Peak instantaneous luminosity as measured by ATLAS.

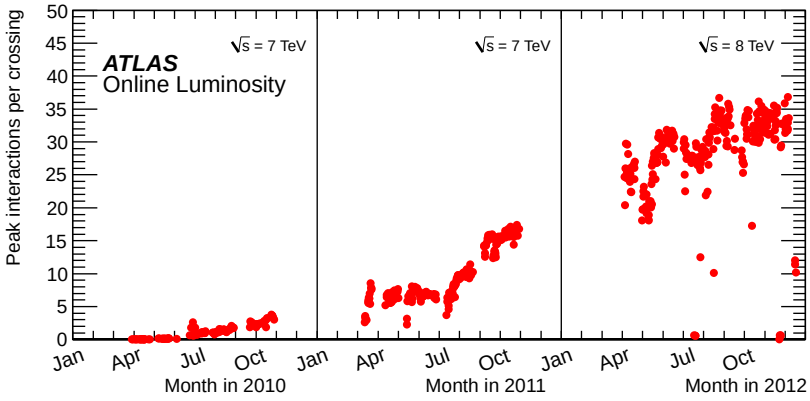


Figure A.2: Peak interactions per bunch crossing in ATLAS as function of time.

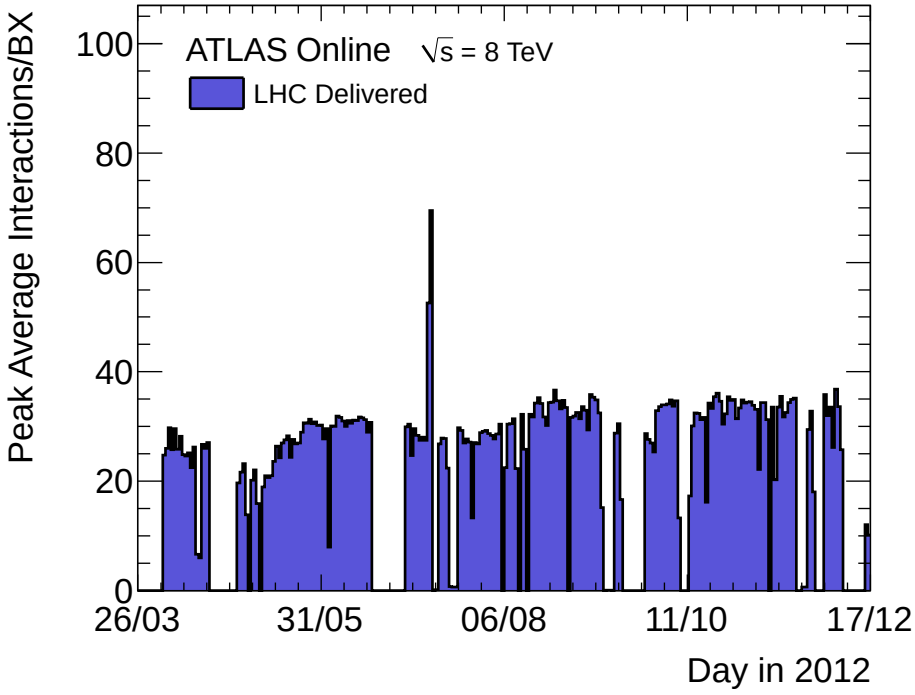


Figure A.3: Peak average pile-up as function of day.

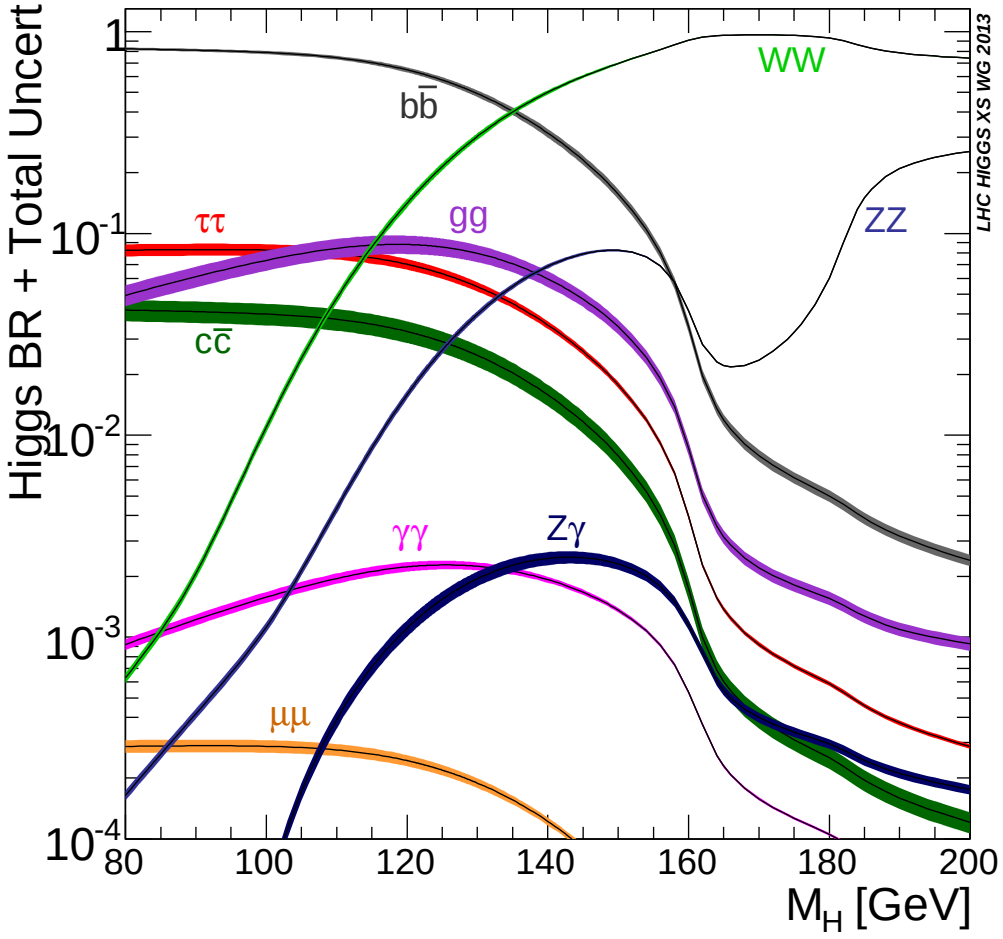
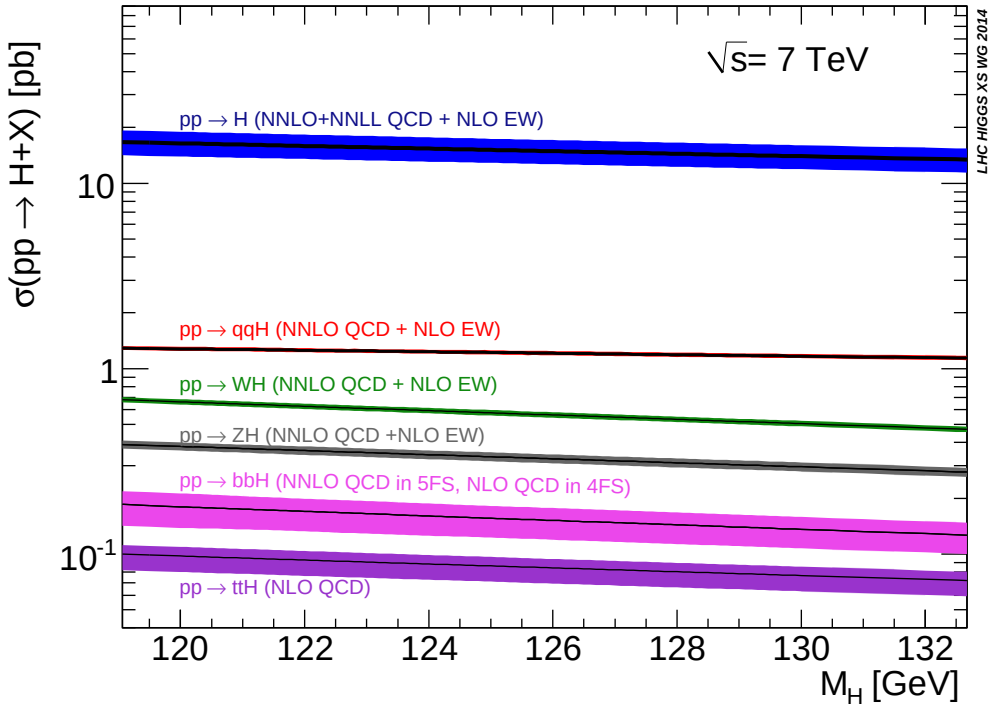
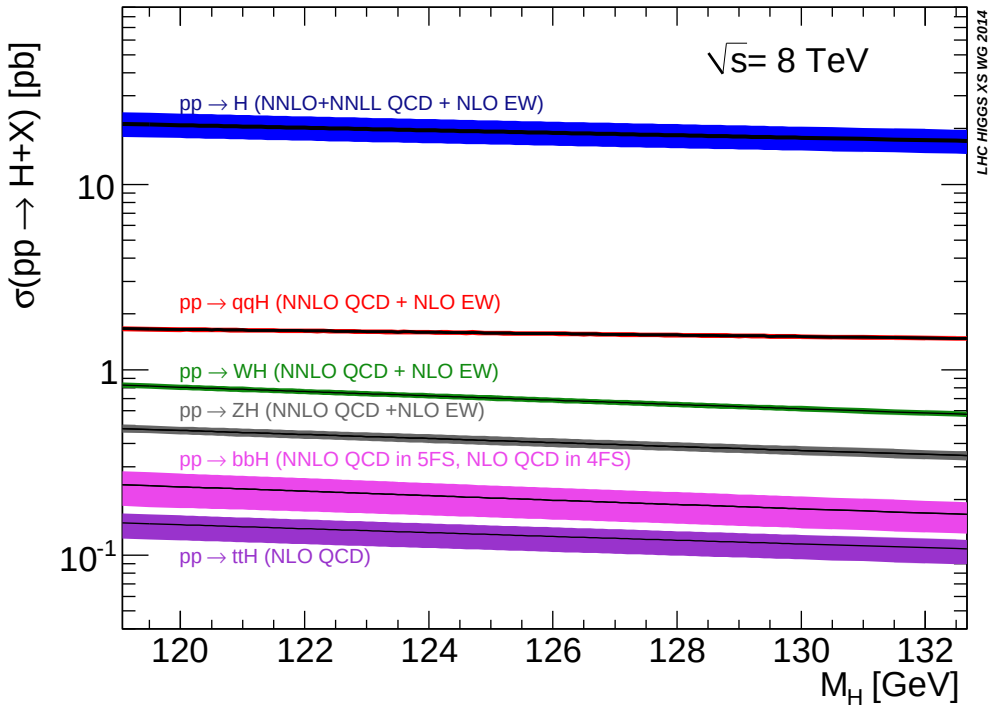


Figure A.4: Branching ratio of Standard Model Higgs. Credit LHC Higgs Cross Section Working Group.



(a) $\sqrt{s} = 7\,\text{TeV}$ – conditions of 2011 data taking



(b) $\sqrt{s} = 8\,\text{TeV}$ – conditions of 2012 data taking

Figure A.5: Total Higgs Cross Section at center of mass energies of $\sqrt{s} = 7\,\text{TeV}$ and $\sqrt{s} = 8\,\text{TeV}$. Credit LHC Higgs Cross Section Working Group.