修 士 学 位 論 文

高輝度 LHC-ATLAS 実験に向けた
初段エンドキャップ部ミューオントリガーシステムの
シミュレーション開発

Development of simulation of Level-0 endcap
muon trigger system for the HL-LHC ATLAS experiment

2025 年 8 月 8 日

専攻名： 物理学専攻
学籍番号： 237S181S
氏名： 張 力

神戸大学大学院理学研究科博士課程前期課程

Abstract

The Large Hadron Collider (LHC), operated by CERN, is the highest-energy proton-proton collider in the world. One of its main detectors, ATLAS, is a general-purpose detector located at one of the collision points and is designed for studying fundamental physics searching for phenomena beyond the Standard Model. The Phase-II upgrade to the High-Luminosity LHC (HL-LHC), starting from autumn 2026, will require an upgrade of the ATLAS trigger system in response to the significantly increased luminosity. Along with this upgrade, the TGC (Thin Gap Chamber) detector, a part of the muon detector for forward-going muons, will have all its electronics renewed in HL-LHC. This study focuses on the development of a software simulator for the TGC Sector Logic, which reconstructs muon tracks from TGC hit information, and on its implementation and testing within the ATLAS software framework, Athena.

The main challenge of the integration on Athena is that directly integrating the existing software algorithm would cause memory usage beyond the limit for distributed computing environment. In this research, an L0TGCSimulator was developed to provide the exact simulation of the Sector Logic behavior, and it has been successfully implemented within the Athena environment. This L0TGCSimulator is based on an existing bitwise simulator, which emulates the bit-level operation of the firmware logic on FPGA. After the implementation, an optimization for memory reduction was performed for the L0TGCSimulator, achieving about a 55% reduction compared to the initial evaluated value. Finally, a Monte Carlo simulation with performance evaluation was carried out for the L0TGCSimulator. In addition, a simple emulator that emulates the muon trigger responses was developed to enable downstream developments in the muon trigger chain.

Contents

1
Introduction

The Standard Model of elementary particle physics (SM) is a remarkably successful theoretical framework, describing the known fundamental particles and their interactions via the electromagnetic, weak, and strong forces. It has been continuously validated through decades of experimental results and provides precise predictions that agree very well with observations.

However, the SM is not a complete theory of nature. It does not incorporate gravity, nor does it provide an explanation for the existence of dark matter or the origin of neutrino masses. These open questions motivate the search for new physics beyond the Standard Model.

To probe these fundamental questions, high-energy particle colliders are indispensable, as they enable us to access shorter distance scales and higher energy regimes where rare physics phenomena are more likely to manifest. The Large Hadron Collider (LHC), located at CERN, is a proton-proton collider with a center-of-mass energy of 14 TeV, designed to investigate fundamental interactions and to search for phenomena beyond the SM. The LHC hosts four major experiments—ATLAS, CMS, ALICE, and LHCb—each designed to investigate different aspects of high-energy physics.

After the discovery of the Higgs boson in 2012 at LHC, the LHC faces more challenges

in probing high energy physics phenomena. To enable the exploration of new physics, including the measurement of the Higgs boson self-coupling or precise determination of the Higgs boson with vector boson and fermion, much larger statistical samples are required. Currently, the LHC is undergoing a comprehensive upgrade to the High-Luminosity LHC (HL-LHC). This upgrade, called Phase-II upgrade, aims to increase the instantaneous luminosity by a factor of five to seven, allowing more collision events to be recorded and improving the precision of physics measurements.

The following sections in this chapter provide an overview of the LHC, the ATLAS experiment, and the upcoming Phase-II upgrade for the High-Luminosity LHC, followed by the motivation of the studies in this thesis.

1.1 The Large Hadron Collider

The Large Hadron Collider (LHC) is a hadron collider of the highest energy in the world, located about 100 m underground at the European Organization for Nuclear Research (CERN) near Geneva, Switzerland. It has a 27-kilometer circular tunnel, crossing the border between Switzerland and France. The LHC was designed to explore the frontiers of high-energy physics by colliding protons and occasionally heavy ions, such as lead nuclei.

Protons are first accelerated using a series of smaller accelerators, like the LINAC, Proton Synchrotron, and Super Proton Synchrotron, before being injected into the LHC ring, where they are further accelerated. The LHC can achieve a center-of-mass energy of up to 13.6 TeV in proton-proton collisions since Run 3, and is capable of delivering instantaneous luminosity of above 2×10^{34} $\mathrm{cm}^{-2}\mathrm{s}^{-1}$. Figure 1.1 provides an overall view of the LHC complex.

One of the most significant achievements of the LHC to date is the discovery of the Higgs boson in 2012 by the ATLAS and CMS collaborations. This long-sought particle completes the Standard Model by confirming the mechanism of electroweak symmetry breaking through the Higgs field, explaining the generation of particle masses.

In the following, a brief overview of the four LHC experiments mentioned above is provided: ATLAS and CMS, which are general-purpose detectors designed to study a wide range of phenomena including Higgs boson production, Supersymmetry, and other physics beyond the SM; ALICE, which specializes in the study of the quark-gluon plasma in heavy-ion collisions; and LHCb, which specializes in rare decays involving b-quarks, for purposes such as measurements of CP violation.

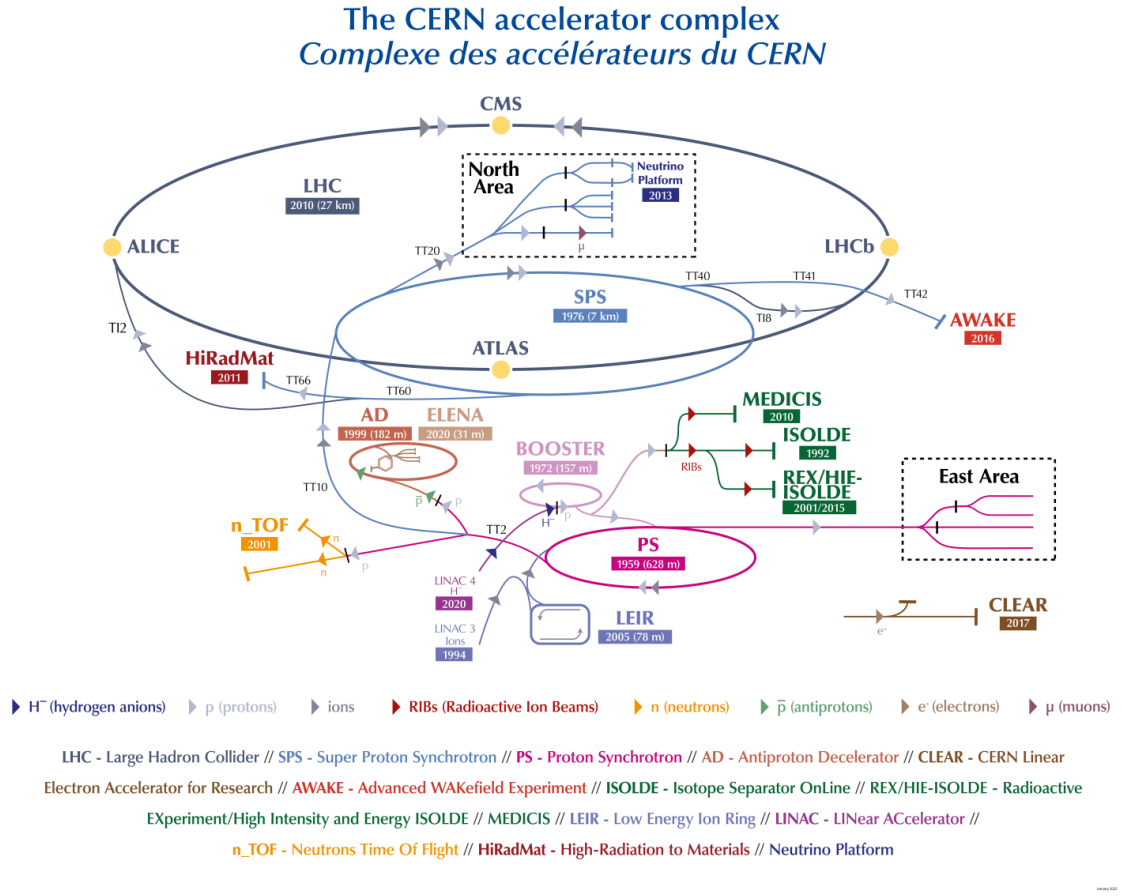


Figure 1.1: Overview of the LHC accelerator complex, illustrating the main accelerator ring and the injector chain responsible for preparing and accelerating the particles prior to collision. [1].

1.2 The LHC-ATLAS Experiment

The ATLAS (A Toroidal LHC ApparatuS) experiment is a general-purpose particle detector experiment located at one of the four main interaction points of the LHC, designed to explore a wide range of physics phenomena. It is composed of several sub-detector systems arranged concentrically around the beam interaction point. Starting from the innermost region, the *Inner Detector*, immersed in a solenoidal magnetic field, is designed to track charged particles and reconstructs their momenta, as well as determining the position of vertices of the hardest scattering, that is, the primary collision with the highest momentum transfer in the event. Surrounding the Inner Detector, the *Calorimeter* system consists of electromagnetic and hadronic calorimeters, which measure the energy of electrons, photons, and hadrons through their interactions with dense absorber materials such as lead and steel. The outermost layer is the *Muon Spectrometer*, which detects muons that penetrate the inner layers, measuring their curvature in toroidal magnetic fields to determine their momenta. The solenoidal magnetic field in the inner region, combined with the toroidal magnetic fields in the outer region, forms the *Magnet System* and allows for precise momentum measurements over a wide energy range. A two-level trigger system is used to select events. In the current LHC Run-3 phase, the first-level trigger is implemented in hardware and uses a subset of the detector information to accept events at a rate below 100 kHz, which is followed by a software-based trigger that reduces the accepted event rate to the order of 1 kHz. A cut-away view of the ATLAS detector is shown in Figure 1.2. A detailed description of the ATLAS muon system, which is most relevant to this study, will be provided in Chapter 2.

1.3 Phase-II Upgrade for HL-LHC

To enhance the sensitivity to rare processes and potential signs of physics beyond the SM, the LHC has experienced a series of upgrades to increase event rate and improve the performance of detectors: upgrades in Long Shutdown 1 (LS1) from 2013 to 2015, and in Long Shutdown 2 (LS2) from 2018 to 2022. Currently, an upcoming *Phase-II upgrade*, a comprehensive upgrade for the LHC to the High-Luminosity LHC (HL-LHC) in LS3, is underway. This Phase-II upgrade, scheduled to start around 2026 and finish at 2030, will bring the LHC from current Run 3 to the Run 4, as shown in Fig 1.3,

The instantaneous luminosity will be enhanced from the current $\sim 2 \times 10^{34}$ cm^{-2}s^{-1} to 5–7.5×10^{34} cm^{-2}s^{-1} by increasing the number of protons per bunch and further focusing the

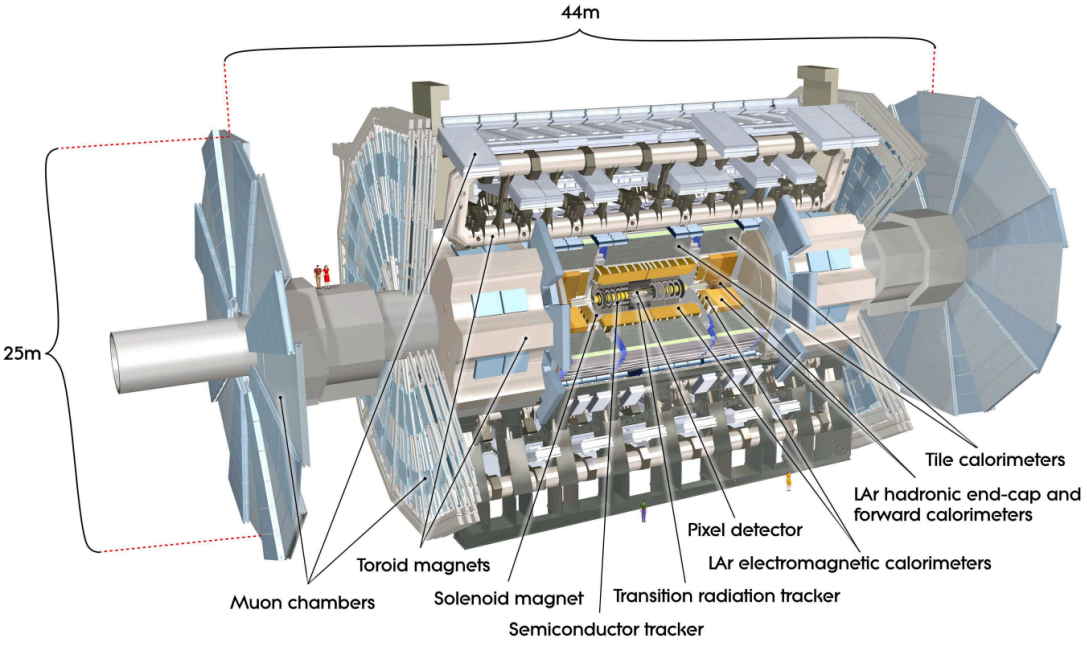


Figure 1.2: Cut-away view of the ATLAS detector [2].

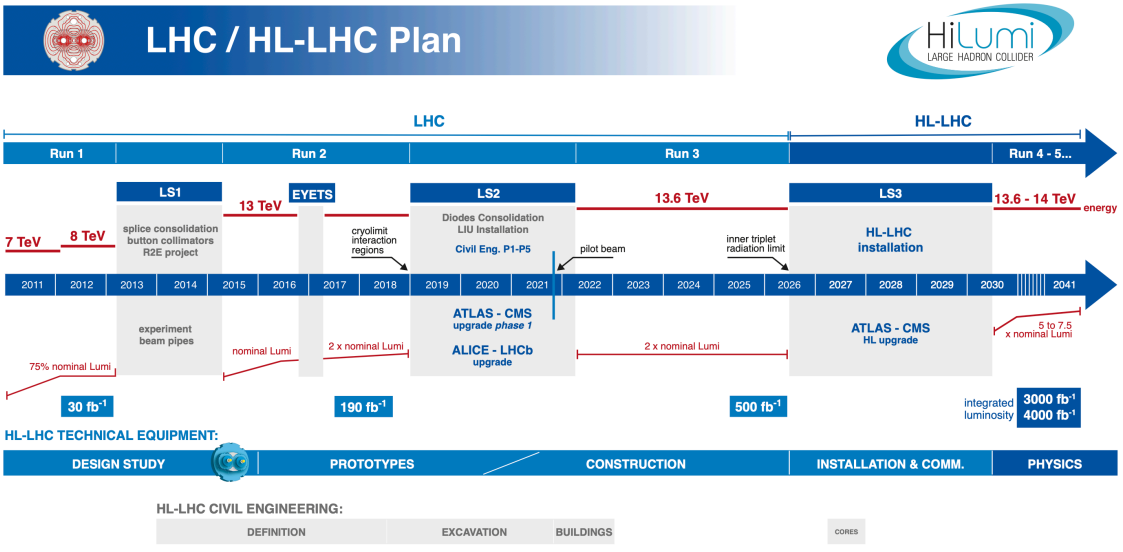


Figure 1.3: HL-LHC schedule (last update January 2025) [3].

beam at the collision points. Over a projected 10-year data-taking period, the HL-LHC is expected to deliver an integrated luminosity of up to 3000–4000 fb^{-1}. This increasing instantaneous luminosity will result in a much higher number of simultaneous proton-proton interactions per bunch crossing, known as *pile-up*, rising from an average of 50–65 in Run 3 to 150–200 during HL-LHC operations (Run 4).

Such a large number of pile-ups poses a challenge to detectors in event reconstruction and background suppression processes. To meet these demands, the ATLAS detector will also undergo an upgrade to nearly all its major subsystems (sub-detectors). In the context of this research, particular attention is given to the relevant upgrades —including improvements to the muon system and the trigger and data acquisition (TDAQ) system, which will be discussed in detail in Chapter 2.

1.4 Motivation and Structure of this Thesis

This thesis focuses on the development of the software simulation for the upgraded muon trigger system in the ATLAS experiment for the HL-LHC. To meet the requirements for the higher luminosity operation, the ATLAS muon trigger system will undergo a major upgrade. As a part of that, the Thin Gap Chamber (TGC), at the endcap region of the muon system, will have its digital electronics and firmware logic entirely replaced and upgraded. As part of this upgrade, software simulation is necessary for development and validation of the firmware, and for estimating the inefficiencies due to the trigger to be corrected in physics data analyses. In order to make the simulation feasible in the muon trigger chain, a corresponding software simulator needs to be implemented in ATLAS software framework, *Athena*. This thesis focuses on the development and the implementation of the Level-0 TGC Sector Logic (SL) software package in simulation for L0 trigger chain.

The main challenge of the implementation is the significant memory usage. There is an existing *bitwise simulator*, which emulates the firmware behavior of the TGC Sector Logic (SL) board, responsible for computing the muon transverse momentum based on the hit pattern combinations. While this approach can reproduce the SL calculation exactly, a single bitwise simulator consumes approximately 296 MB of memory, yet it covers only 1/48 of the TGC region. Consequently, extrapolating this to the full endcap region would result in a total memory usage of

$$48 \times 296 \text{ MB} \simeq 14 \text{ GB}.$$

Since the memory in PCs for distributed tasks of Athena for a software package like the endcap muon trigger simulation should be limited well below 500 MB, the memory usage of 14 GB is far beyond the limitation.

To enable downstream developments in L0 muon trigger chain, a simple emulator was developed as a preliminary step of this research, providing the emulation of outputs of the L0 muon trigger chain. This emulator generates trigger information for the next stage using a Gaussian-based smearing algorithm. Subsequently, a simple simulation for trigger acceptance was implemented to bring the emulator closer to reality. However, the emulator still lacks the ability to reproduce actual physical phenomena, since the actual SL momentum calculation exhibits non-Gaussian characteristics. It confirms, as a result, the necessity for a high-precision simulator that can reproduce the trigger behavior for the L0 TGC Sector Logic on Athena.

Therefore, as the core of this research, the development of simulation of the endcap TGC Sector Logic (SL), was performed on the basis of the bitwise simulator. In this part, a simulator with bit-level behavior was implemented into the Athena framework, simulating consistent logic behavior with the actual SL. A first attempt for optimizing the storage of Look-Up Table (LUT) was also applied to address the memory limitation. The result of the simulation is compared to the stand-alone bitwise simulator.

This thesis is structured as follows: Chapter 2 provides an introduction of ATLAS muon system and TDAQ system, along with the corresponding upgrades for the HL-LHC. Chapter 3 briefly introduces the ATLAS software framework, Athena. Chapter 4 presents the development and the update to the simple emulator for L0 muon trigger system, followed by an assessment of its performance. Chapter 5 describes the implementation and optimization of the simulator for the TGC Sector Logic. Chapter 6 presents the conclusions of this research and discusses future prospects for further development and applications.

2

The ATLAS Experiment

The ATLAS experiment at the LHC uses a multipurpose particle detector with a forward–backward symmetric cylindrical geometry and a near 4π coverage in solid angle, expected to explore physics phenomena from precise measurements of Standard Model parameters to the search for new particles and interactions. In preparation for the HL-LHC, the ATLAS detector is undergoing a series of upgrades. As the instantaneous luminosity increases, resulting in higher pile-up and the background event rates, the detector will be enhanced to suppress backgrounds and maintain high resolution. This chapter first provides background information on the ATLAS coordinate system and magnet system, followed by introductions to the components relevant to this study — the muon detector system and the Trigger and Data Acquisition (TDAQ) system. After that, the Phase-II upgrade strategy on these components is briefly outlined.

2.1 Coordinate System

To describe positions and directions of particles within the ATLAS detector, a right-handed coordinate system is adopted with the origin located at the interaction point (IP). In this

system, the z-axis is along the beam pipe, the x-axis points from the IP to the center of the LHC ring, and the y-axis points upwards.

Cylindrical coordinates (θ, ϕ, z) are also commonly used. Here, θ denotes the polar angle from the z-axis and spans from 0 to π, while ϕ is the azimuthal angle measured around the beam axis, ranging from $-\pi$ to π, measured from the x-axis. The distance from the z-axis is denoted by R. A schematic of the coordinate system is shown in Figure 2.1.

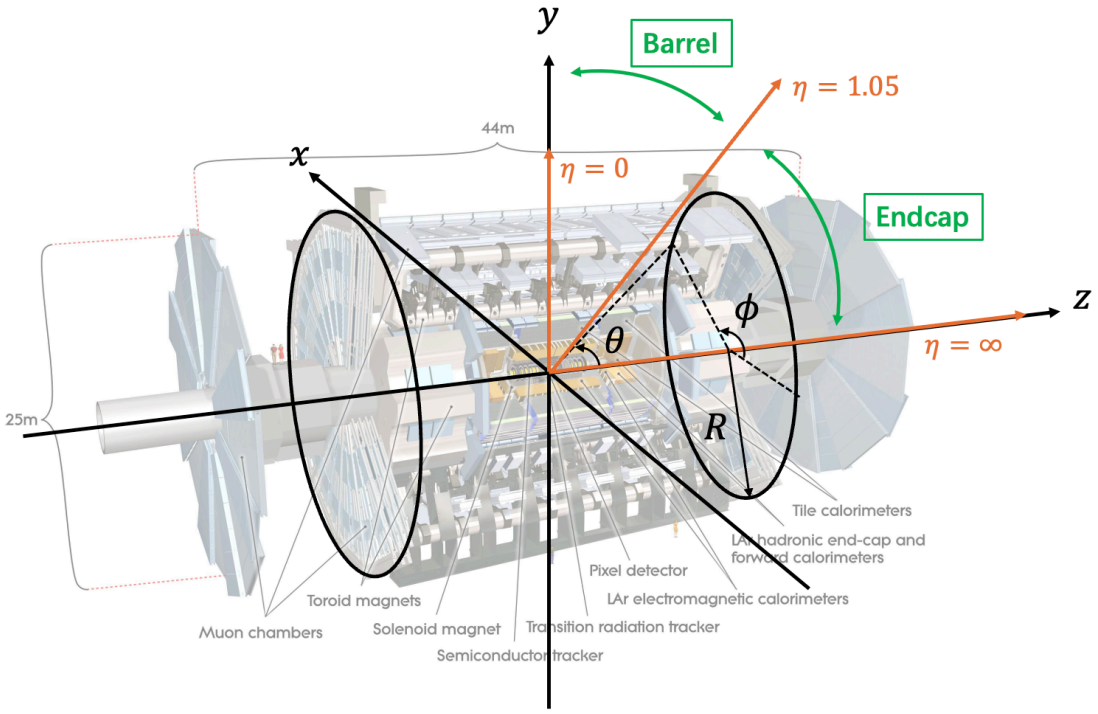


Figure 2.1: The coordinate system used in the ATLAS experiment.

In the ATLAS experiment, since proton-proton collisions take place near the nominal IP, angular variables of produced particles are typically defined with respect to this point. While the polar angle θ, measured from the beam (z) axis, provides a direct geometric description, it lacks invariance under Lorentz boosts along the beam direction. Therefore, in a typical inelastic reaction, the distribution of particles in θ is not uniform, making it less suitable for characterizing particle kinematics in high-energy collisions. To overcome this, a variable called rapidity y is introduced. Rapidity y is defined in terms of a particle's energy and the momentum along z-axis, as in Equation 2.1. Rapidity differences Δy between particles are Lorentz-invariant under boosts along the z-axis, making it suitable for comparing particle distributions across different frames:

$$y = \operatorname{arctanh} \beta_z = \frac{1}{2} \ln \left(\frac{1 + \beta_z}{1 - \beta_z} \right) = \frac{1}{2} \ln \left(\frac{E + p_z}{E - p_z} \right), \tag{2.1}$$

where $\beta_z = p_z/E$ is the velocity component along the z-axis, normalized by the speed of light in natural units, E is the energy, and p_z is the momentum along the beam axis.

In most collider events, the final-state particles are highly relativistic, with masses negligible compared to their momenta. In such cases, rapidity y can be approximated by the pseudorapidity η, which depends solely on the polar angle θ and is thus easier to compute from detector measurements:

$$\eta = -\log\left(\tan\frac{\theta}{2}\right). \tag{2.2}$$

This η is widely used in physics analyses as it preserves the boost-invariant properties of rapidity y in the massless limit, which is more appropriate since the masses of particles are not accessible in high-energy collider detectors.

In the ATLAS experiment, the detector is divided into a cylindrical *barrel* and two *endcaps*. The endcap region, corresponding to $1.05 < |\eta| < 2.41$, consists of the "A-side" and "C-side", which point along the positive and negative z-axis, respectively. The barrel region, corresponding to $|\eta| < 1.05$, lies between the two endcaps.

In addition, transverse energy is defined as

$$E_{\mathrm{T}} = E\sin\theta$$

and transverse momentum as

$$p_{\mathrm{T}} = \sqrt{p_x^2 + p_y^2} = p\sin\theta$$

which is the momentum component perpendicular to the beam direction. Since the total transverse momentum remains zero during the collision, the transverse momentum of all the final state particles can be assumed to be zero.

The angular distance ΔR between two particles is a commonly used quantity, and is defined using the pseudorapidity as

$$\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}. \tag{2.3}$$

2.2 Magnet System

The ATLAS detector employs a unique hybrid magnetic system composed of four large superconducting magnets, spanning 24 m in diameter and 45 m in length [2]. This system includes a central solenoid and three toroidal systems (a barrel toroid and two endcap

toroids), enabling high magnetic field coverage across both inner-tracking and muon detection systems.

The solenoid magnet, placed along the beam axis, provides a 2 T axial magnetic field for the Inner Detector (ID) inside the electromagnetic calorimeter. Surrounding the calorimeter system are the toroidal magnets: the barrel toroid, composed of eight superconducting coils, and two endcap toroids, which together generate a toroidal magnetic field of about 0.5 T to 1 T for the muon spectrometer in the central and forward regions. The schematic geometry of the magnet windings is illustrated in Figure 2.2. The solenoid is located inside the calorimeter volume, while the barrel and endcap toroids are interleaved around it.

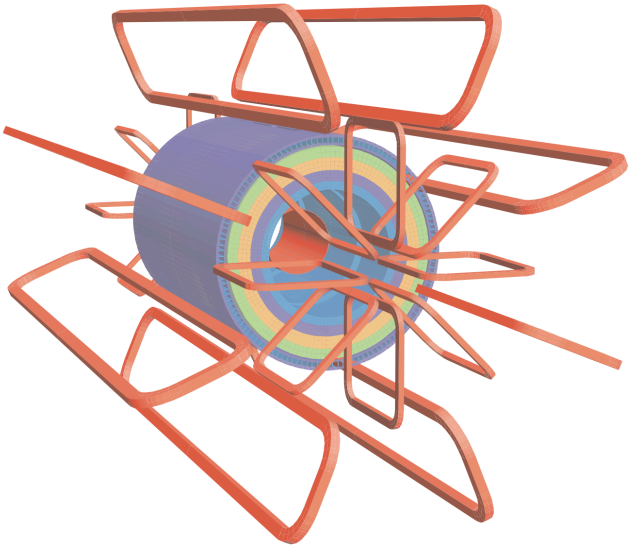


Figure 2.2: Geometry of magnet windings and calorimeter steel [2]. The eight barrel toroid coils and endcap coils are interleaved. The solenoid winding is located inside the calorimeter volume (the light blue cylinder).

2.3 Muon Spectrometer

The ATLAS muon spectrometer is the outermost subsystem of the detector, designed to provide independent momentum measurements for muons. It relies on magnetic deflection of muon trajectories using large superconducting air-core toroidal magnets as introduced in Section 2.2 above.

The magnetic field in the spectrometer is shaped by a central barrel toroid and two endcap toroids with field lines oriented along circles at constant R and z, so that muons are

bent mainly in the direction perpendicular to ϕ. For muons with pseudorapidity $|\eta| < 1.4$, the bending power is provided mainly by the barrel toroid. In the forward regions ($1.6 < |\eta| < 2.7$), deflection comes from the endcap toroids. The intermediate transition region ($1.4 < |\eta| < 1.6$) receives contributions from both barrel and endcap fields. This field configuration creates a predominantly orthogonal bending force relative to the muon trajectory.

Figure 2.3 demonstrates the layout of the ATLAS muon system. For precise track coordinate measurements, Monitored Drift Tubes (MDTs) are used over most of the acceptance. The trigger system covers the pseudorapidity range $|\eta| < 2.4$. In the barrel region, Resistive Plate Chambers (RPCs) are used, while Thin Gap Chambers (TGCs) are used in the endcap. The chambers of RPCs and TGCs perform three main functions: identifying the bunch crossing, providing well-defined p_{T} thresholds as trigger decisions, and determining the muon coordinate. Muon detectors located at the inner endcap region, the New Small Wheels (NSWs), are also integrated into the muon trigger system, working in conjunction with other detectors such as the TGCs to further reduce the background in endcap region.

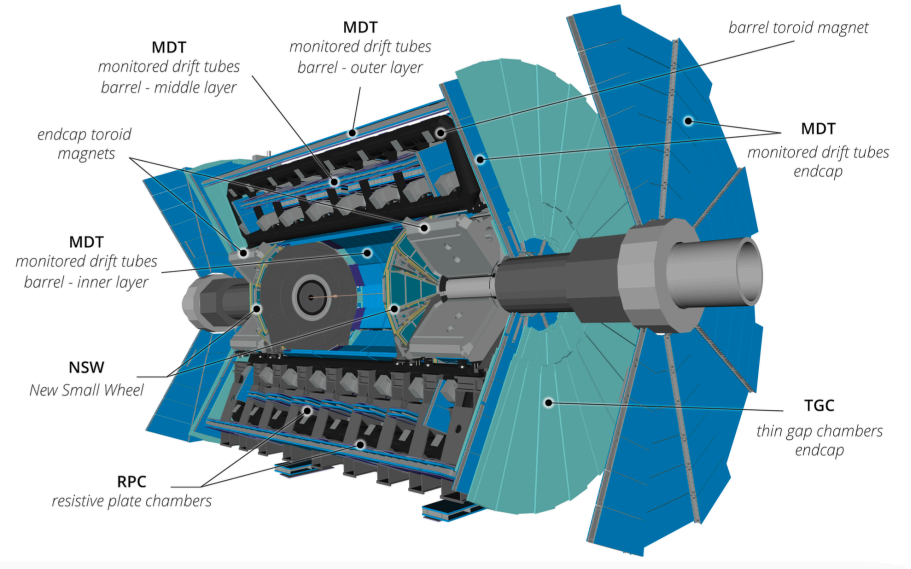


Figure 2.3: Cut-away view of the layout of the ATLAS muon system [4].

2.3.1 Resistive Plate Chamber (RPC)

The Resistive Plate Chambers (RPCs) serve as the barrel trigger detectors in the muon spectrometer system. The RPC is a gaseous detector with two parallel resistive plates and

a 2 mm gas gap. It operates in avalanche mode with fast timing (∼5 ns), using a $C_2H_2F_4$-based gas mixture. High voltage induces avalanches, which are read out via capacitive coupling to external strips. A schematic of RPC chamber is shown in Figure 2.4.

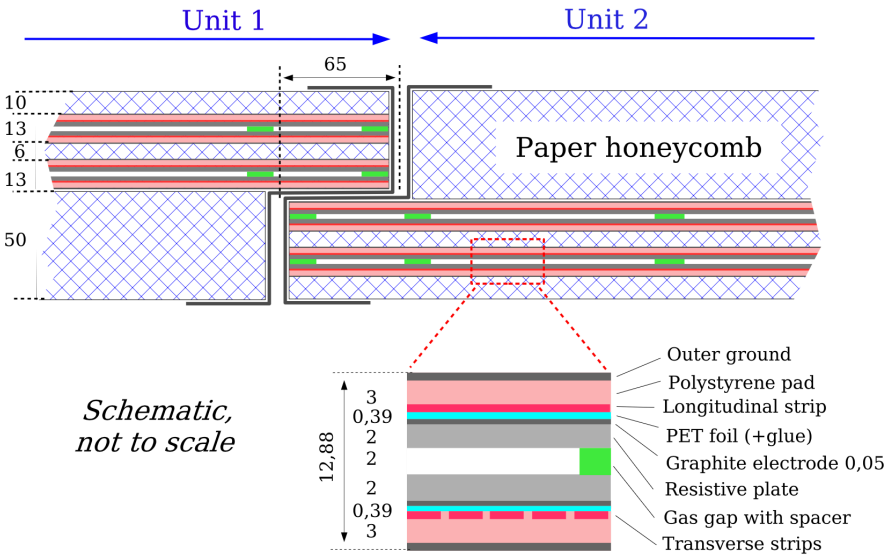


Figure 2.4: A Cross-section view of an RPC chamber [2]. Each chamber is composed of two joined units, with each unit comprising dual gas volumes, four resistive electrodes, and readout planes sensitive to both transverse and longitudinal coordinates. Dimensions are given in mm.

They are currently arranged in three concentric cylindrical layers around the beam axis, referred to as RPC1, RPC2, and RPC3 stations. RPC1 and RPC2 stations are in the Barrel Middle (BM) region and RPC3 station is in the Barrel Outer (BO) region. Trigger decisions are based on spatial coincidences between these stations. Each station comprises two independent detector layers capable of measuring both η and ϕ, providing up to six hit points for a traversing muon. The BM stations (RPC1 and RPC2) are used for low-p_T triggers (6–9 GeV) with a 3-out-of-4 coincidence logic, while the BO (RPC3) station enables high-p_T triggers (9–35 GeV) using a 1-out-of-2 OR logic. A sector of RPC layout in ATLAS is shown in Figure 2.5.

Until LHC Run 3, due to mechanical structures that support the barrel toroid coils and other systems or services in the barrel region, there are some unavoidable "dead zones" in the detectors at the barrel region. The BM region of RPC detector is particularly affected, resulting limitation in muon trigger acceptance and efficiency. To improve the trigger acceptance, a new station, RPC0, will be added in the Barrel Inner (BI) region as a part of the Phase-II upgrade. Figure 2.6 gives the RPC detector layout after the Phase-II upgrade. The BI region, unaffected by mechanical obstructions, offers near-complete

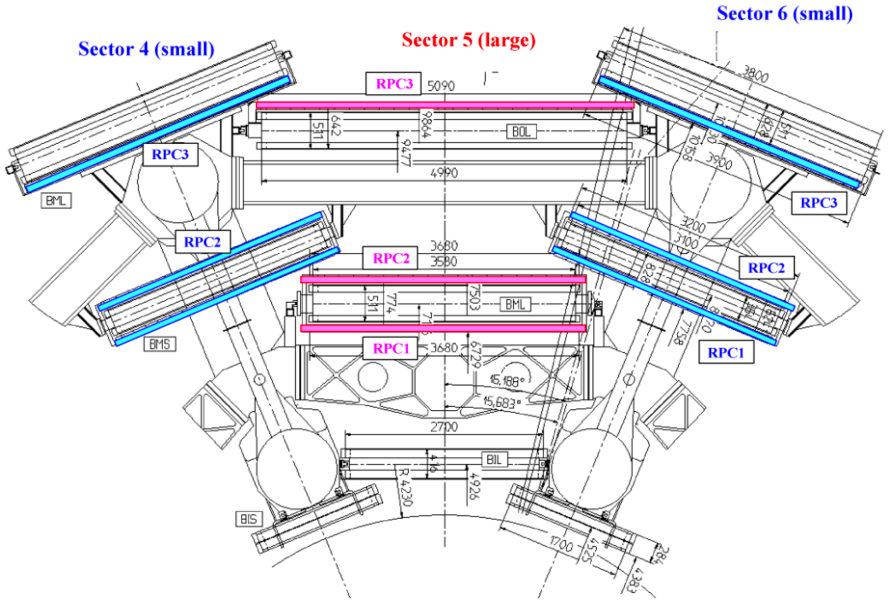


Figure 2.5: A cross-sectional view of the upper barrel region with the RPC chambers highlighted in color [2]. In the middle station, RPC1 and RPC2 are placed below and above the MDT chambers, respectively. In the outer station, RPC3 is located above the MDT in large sectors and below it in small sectors. All dimensions are given in millimeters.

angular coverage, generating more coincidence types combined with other RPC stations, thereby to improve the trigger efficiency.

2.3.2 Monitored Drift Tube (MDT)

The Monitored Drift Tubes (MDTs) are precision tracking detectors covering the region of $|\eta| < 2.7$. They consist of layered arrays of pressurized drift tubes as base elements, each with a diameter of 29.970 mm and a central anode wire of 50 μm in diameter. A cross-section view and a longitudinal cut view of a drift tube of the MDT chamber are shown in Figures 2.7. The tubes are filled with a gas mixture of argon and carbon dioxide ($Ar/CO_2 = 93/7$) at 3 bar. A high voltage of 3080 V is applied between the cathode tube and the central wire [2]. As charged particles traverse the gas, they cause ionisation, and make electrons drift toward the anode wire. By measuring the signal rise time caused by this drift, the radial position of the particle's trajectory can be reconstructed with a spatial resolution of up to 35 μm. The maximum drift time within a tube is about 700 ns.

During Phase-II upgrade, a new type of chamber, sMDT (Small-diameter MDT), with only half the tube diameter of the original MDT chambers, will be used to cope with the

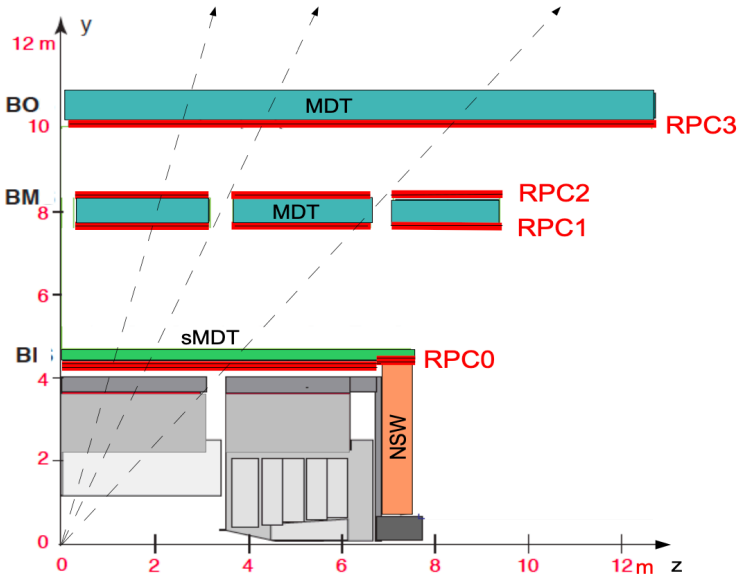
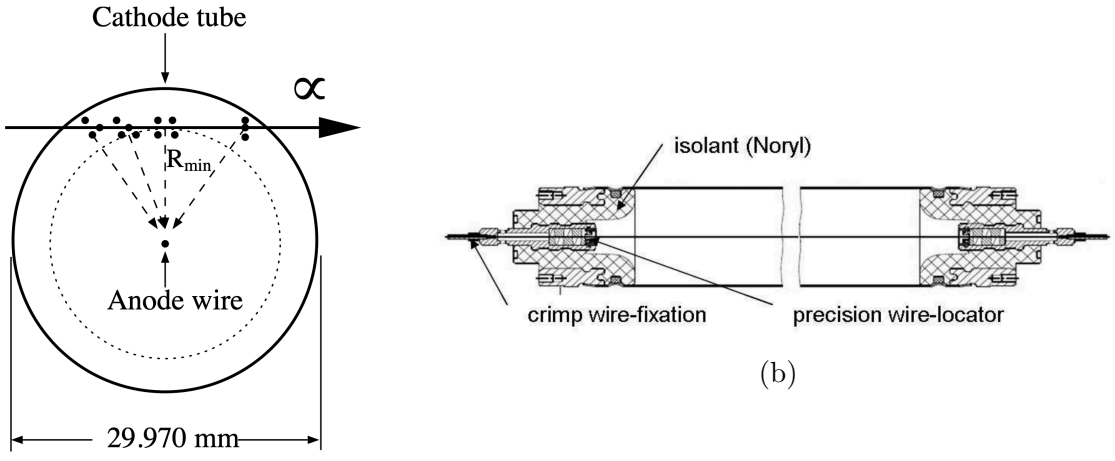


Figure 2.6: RPC detector layout in the Phase-II upgrade [5]. The RPC0 layer is newly added to improve acceptance caused by the holes in the RPC1 and RPC2.



(a) (b)

Figure 2.7: Cross-sectional (a) and longitudinal (b) views of a drift tube in an MDT chamber [2].

higher background rates at the HL-LHC. The maximum drift time of sMDT tubes is only 175 ns, in contrast to about 720 ns for MDT tubes. The sMDT design makes it possible to accommodate additional chambers in the limited available space. Figure 2.8 shows the comparison between MDT and sMDT tubes.

Figure 2.8: Pictures of MDT (left) and sMDT (right) tubes.

2.3.3 Thin Gap Chamber (TGC)

The Thin Gap Chambers (TGCs) are used in the endcap region of the ATLAS muon spectrometer as trigger detectors, covering the pseudorapidity range $1.05 < |\eta| < 2.4$. They are positioned on both sides of the toroidal magnetic field, with the inner detectors located in the Endcap Inner (EI) region and the outer detectors forming the Big Wheel (BW). The picture of TGC BW are given as Figure2.9. The BW TGCs consist of three stations, designated as M1, M2, and M3 from the inner to outer layers.

TGCs are multi-wire proportional chambers (MWPCs), as illustrated in Figure 2.10. The chamber contains *wires* and *strips*, which are arranged orthogonally to enable a two-dimensional readout. As the name suggests, the *wire segments*, which include the sense wires, provide measurement in R direction, while the *strip segments* correspond to the ϕ direction. The chambers are filled with a gas mixture of CO_2 and n-C_5H_{12}, with a high voltage of 2.8 kV applied to the wires. To achieve a high time resolution for bunch crossing identification, the interval between sense wires is designed to be 1.8 mm in order to reduce drift time and signal readout delay.

For higher space resolution during reconstruction, stations are composed of multiple layers. The M1 station contains three layers of wires and two layers of strips (*triplet*), while M2 and M3 consist of two layers each for both wires and strips (*doublet*). The wire and strip signals are read out by grouping them. These readout channels of wires and

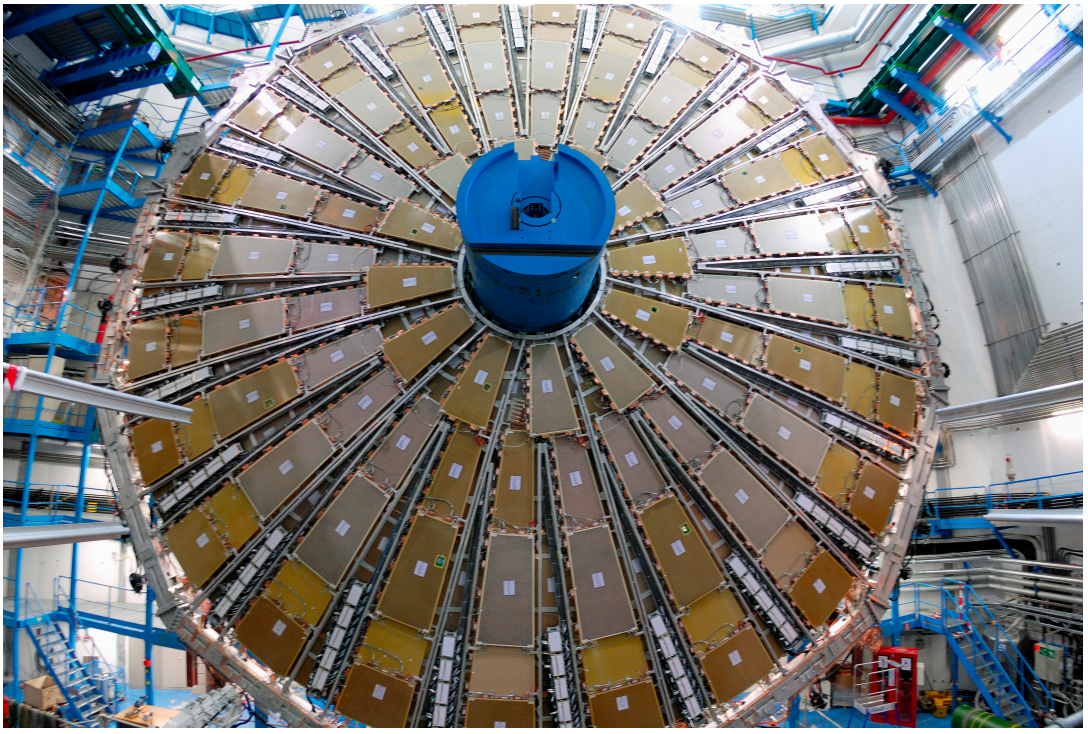


Figure 2.9: The picture of one side of TGC BW, including 24 sectors along ϕ direction.[6].

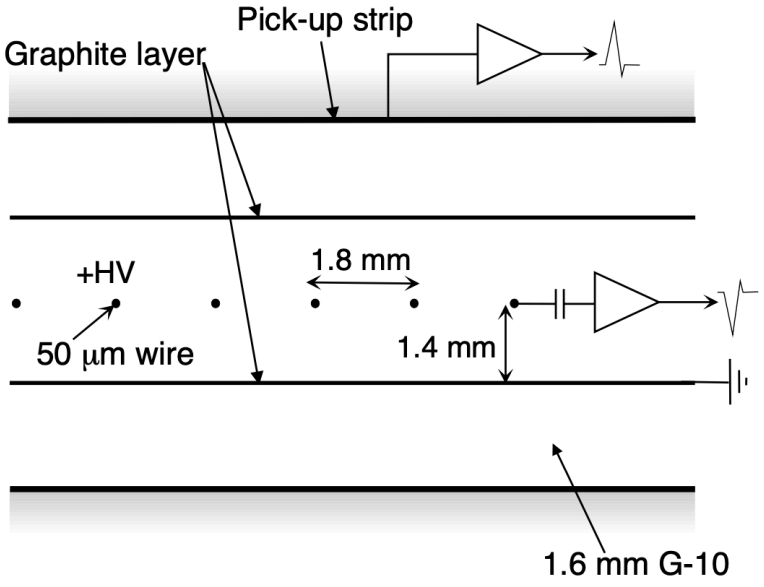


Figure 2.10: The structure of TGC, including anode wires, graphite cathodes, G-10 layers and a pick-up strip, orthogonal to the wires.[2].

strips from these doublet or triplet layers are all "staggered" intentionally with adjacent layers, generating the conception of *staggered ID*. Each staggered ID corresponds to a unique channel combination that spans across offset wire or strip layers, and enables finer position resolution. In M1 with three wire layers staggered by 1/3 each, the staggered ID achieves roughly 1/3 of the layer's intrinsic spatial resolution. In M2 and M3, the two-layer structure results in a resolution gain of roughly 1/2 per ID unit. The staggered IDs are systematically defined and registered in databases, allowing consistent identification across layers during pattern recognition and offline reconstruction.

2.3.4 New Small Wheel (NSW)

The New Small Wheel (NSW) is a muon detector located inside the toroidal magnetic field region, covering the pseudorapidity range of $1.3 < |\eta| < 2.7$. In order to deal with the degradation in tracking efficiency and momentum resolution for muons [7], the NSW replaces the former Cathode Strip Chambers (CSCs) and the inner Multi-Wire Drift Tubes (MDTs) used until LS2 in the endcap region. The NSW has two sides, each consisting of 16 sectors, including 8 large sectors and 8 small sectors overlapping each other in terms of angular coverage. The illustration of NSW sectors layout and the layer structure are shown in Figure 2.11. The NSW consists of 16 layers arranged in a sandwiched structure: four layers of small-strip Thin Gap Chambers (sTGCs), followed by two sets of four-layer Micromegas (MM) chambers, and again four layers of sTGCs. This multilayer configuration enables precise determination of muon position and angle based on the combination of hit positions across layers.

The sTGC, short for "Small-strip Thin Gap Chamber", is named after its much finer strip pitch compared to the TGC chambers introduced above. It consists of a grid of 50 µm gold-plated tungsten wires arranged with a 1.8 mm pitch, sandwiched between two cathode planes positioned 1.4 mm from the wire plane. The MM is a micro-pattern gaseous detector composed of a 5 mm drift gap and a 128 μm amplification region separated by a stainless-steel mesh. The amplified signals are read out via 400 μm pitch strips. The NSW is designed to achieve spatial resolutions of 0.005 in η, 10 mrad in ϕ, and an angular resolution of 1 mrad relative to the beam axis.

During Run 3, the NSW, together with the Tile Calorimeter and outer TGC detectors, participates in "Inner Coincidence" logic of the endcap muon trigger to reject "fake muons" not originating from the interaction point (IP). For the HL-LHC upgrade, additional detectors such as TGC EIL4 and RPC BIS78 will be introduced for the endcap muon trigger system to support this coincidence scheme, further reducing the trigger rate and

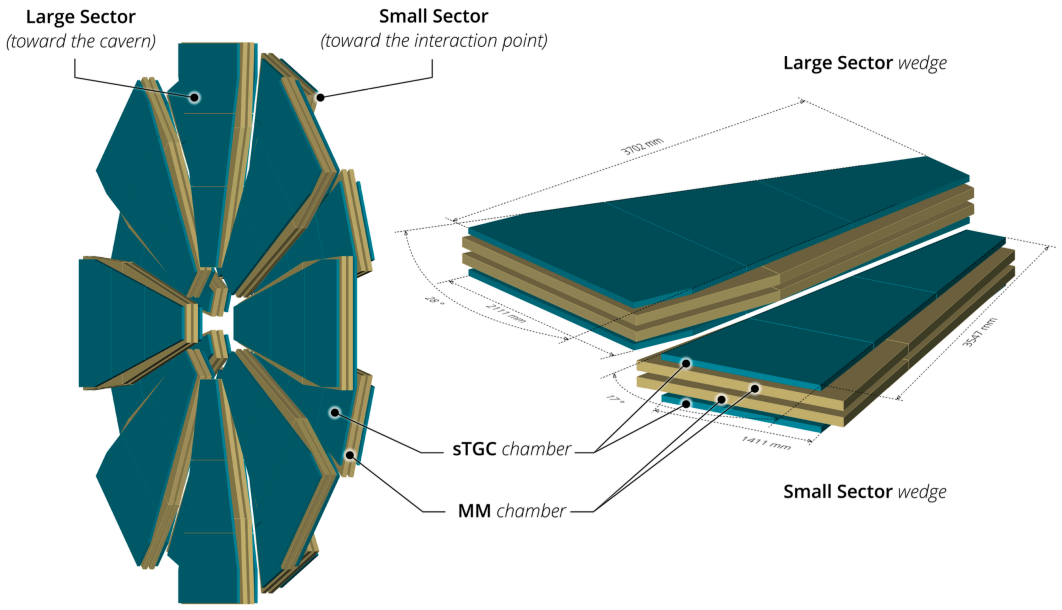


Figure 2.11: Schematic of the NSW layout and layer structure [4].

improving efficiency. Figure 2.12 demonstrates the inner coincidence logic by showing an example of fake muon in HL-LHC.

2.4 TDAQ System

In the ATLAS experiment, the selection and recording of events is handled by the Trigger and Data Acquisition (TDAQ) system, consisting of the Trigger System and the Data Acquisition System. They are introduced as below.

2.4.1 Trigger System

The proton-proton collisions occurring at a frequency of 40 MHz at the LHC, resulting in a total inelastic interaction rate of about 2 GHz in Run 3. Therefore, it is unpractical to record all the events from proton-proton collisions. In addition to the massive data volume, the vast majority of collision events are dominated by interactions with particles with only small p_{T}, regarded as background that do not arouse interests. To address this, ATLAS employs a trigger system to select events of potential interest only for further physics analysis.

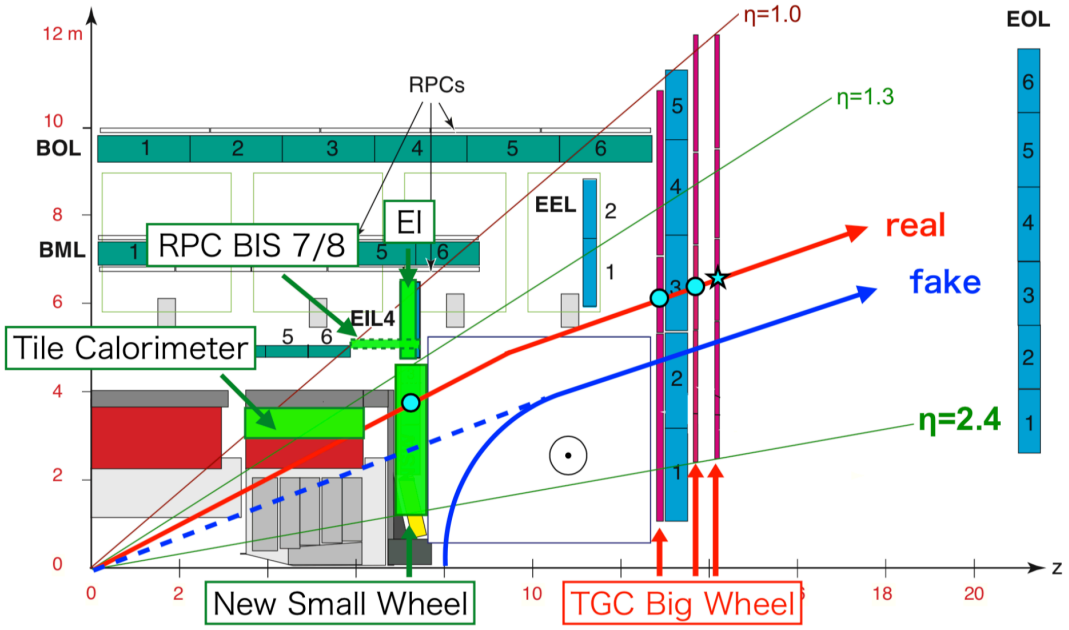


Figure 2.12: An example of a "fake muon", which is a charge particle initially originated from the beam pipe rather than the IP [8].

The trigger system is divided into two sequential stages: a hardware-based Level-1 Trigger (L1 Trigger) that performs a fast initial selection, and a software-based High-Level Trigger (HLT) that provides a further event selection with higher precision as the second step in Run 3. A schematic of ATLAS trigger system in Run 3 is shown in Figure 2.13.

The L1 trigger primarily receives inputs from two independent systems: *L1Muon* and *L1Calo*, which use custom electronics to trigger on reduced-granularity information from the muon detectors and calorimeters, respectively. A third component, the *L1Topo* processor, applies real-time topological selection criteria based on kinematic informations in the L1Muon and L1Calo systems. The trigger decision at L1 is made by the Central Trigger Processor (CTP), which receives input from L1Muon via the Muon-to-Central Trigger Processor Interface (MUCTPI), from L1Calo, L1Topo, and other auxiliary subsystems. Up to 512 different L1 trigger items can be configured in the CTP. The L1 system reduces the event rate from 40 MHz to a maximum of 100 kHz within a latency constraint of less than 2.5 μs.

Once the events are accepted at L1 trigger, they are then sent to a software-based HLT. At this stage, *online* reconstruction algorithms analyze the data at progressively higher levels within restricted Regions-of-Interest (RoIs) identified by L1. These algorithms apply more detailed and computationally intensive reconstruction and selection criteria. The HLT software is incorporated in the ATLAS software framework, Athena, which is also used

in *offline* analysis for recorded data. This Athena framework is introduced in Chapter 3. After HLT processing, the final data output rate is reduced to approximately 3 kHz for Run 3, which will then be written to permanent storage for analysis.

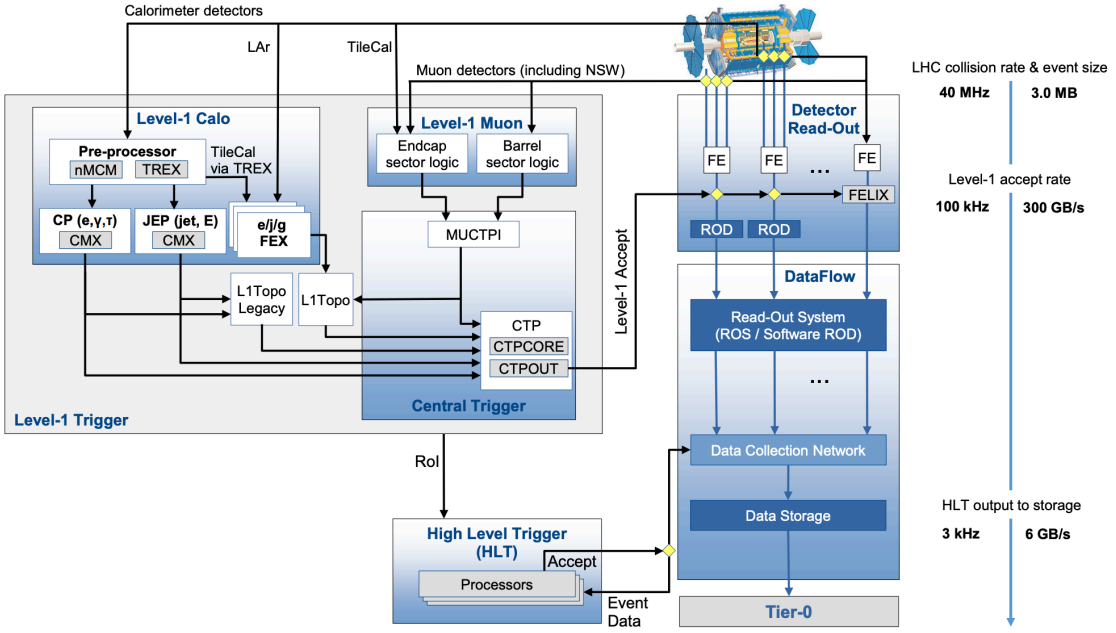


Figure 2.13: Schematic overview of the ATLAS trigger system in Run 3, showing the two-tier architecture of L1 and HLT triggers [9].

2.4.2 Data Acquisition System

The Data Acquisition (DAQ) system is responsible for the transport and assembly of data cooperating with the two-level trigger system. It begins at the detector-specific front-end and off-detector electronics, which perform series of data processing and monitoring features before passing the trigger and other downstream systems. In Run 1 and Run 2 of the LHC, data from the Front-End Electronics (FE) were accepted according to the Level-1 accept (L1A) signal and then transmitted to the Read-Out Drivers (RODs), which subsequently forwarded the accepted data to the common stage Read-Out System (ROS). The ROS, composed of commodity servers equipped with custom-built I/O cards, temporarily stores subdetector data in internal memory buffers, named Read-Out Buffers (ROBs). Data flow from the ROS to the High-Level Trigger (HLT) is orchestrated by the Data Collection Manager (DCM), which handles on-demand data requests from HLT processing nodes (HLTMPPUs). These nodes perform event reconstruction and selection, determining whether events are saved for permanent storage or discarded, and then send

this decision to HLT.

The upgrades for DAQ system in Run 3 are achieved by new modules of the Front-End Link eXchange (FELIX) read-out system and software, running on commodity servers (SW ROD), which are integrated into the read-out path for detector systems with upgraded electronics. L1 Calo, L1 Muon and the Central Trigger all send accepted data to the Read-Out System (or FELIX/Software ROD). Both ROS and SW ROD interfaces present a unified interface to the HLT, enabling data routing in both scenarios. Once the HLT processing has been completed, passed events are forwarded to a dedicated cluster of servers (known as Sub-Farm Outputs (SFOs)) for processes like packing, compression, and transfer to offline storage. The bandwidth to permanent storage in Run 3 are allowed for up to 8 GB/s [9].

2.5 Upgrade for TDAQ System

To cope with the significantly increased event rates at the HL-LHC, the ATLAS TDAQ system is undergoing a comprehensive Phase-II upgrade. The upgraded baseline design is shown schematically in Figure 2.14. which includes enhancements to the Level-0 (L0) Trigger, Event Filter (EF) and Data Acquisition (DAQ) systems.

The upgraded Level-0 Trigger System consists of subsystems of L0Calo, L0Muon, the Global Trigger, and the Central Trigger Processor (CTP). The L0Calo system is mainly based on the former L1Calo system with minor changes. For L0Muon, a thorough upgrade is performed comparing to Run 3, allowing it identify muon candidates by data from all the muon subsystems and a subset of the Tile Calorimeter. To improve the trigger coverage, new RPC inner stations and new function of MDT precise momentum measurements are added into L0Muon system. The Global Trigger system are implemented newly as a subsystem of L0 Trigger, which is responsible for offline-like algorithms on full-granularity calorimeter data. The event rate after L0 Trigger filtering will be less than 1 MHz.

Once the accept signal from the Level-0 trigger (L0A) is issued, event data from detector front-end electronics are sent to the FELIX system, as the first component of the Readout subsystem. From there, data are routed through the Data Handlers and buffered in the Dataflow subsystem. The Dataflow subsystem “adjust” the event data by a series of processes like buffering, transporting, aggregating and compressing, to ensure consistency with the input interface of the Event Filter (EF) system.

The Event Filter (EF) system is based on the Processor Farm [10], a heterogeneous system consisting of CPU cores and accelerators, to perform the EF reconstruction which includes

the compute intensive ITk track reconstruction. By a series of processes, events accepted by the EF are eventually reduced to a rate of 10 kHz. The raw output event size is about 6 MB, and the total bandwidth is 60 GB/s.

2.6 Upgrade for Muon Trigger System

The upgraded Level-0 Muon Trigger System for the HL-LHC is expected to deal with higher trigger rates and improve muon trigger efficiency. The upgrade strategy is illustrated as a block diagram in Figure 2.15. It consists of several main components: the Barrel Sector Logic, Endcap Sector Logic, the NSW Trigger Processor, and the MDT Trigger Processor. Both the Sector Logic and the NSW Trigger Processor components installed during LS2 will be replaced with new hardware to accommodate the higher performance requirements. The Barrel Sector Logic receives hit information from the RPC and energy flags from the Tile Calorimeter. The Endcap Sector Logic, covering the region $1.05 < |\eta| < 1.3$, receives hits from TGC and RPC. At $1.3 < |\eta| < 2.4$, muon trigger rates are reduced by TGC and NSW, retaining the efficiency for the muons with the transverse momentum p_{T} higher than the threshold. The NSW Trigger Processor will be deployed as a separate component from Sector Logic due to large amount of resources consumed by track-segment reconstruction of NSW hits. After initial candidate track reconstruction, the Sector Logic sends these candidates to the newly added MDT Trigger Processor, which offers higher spatial resolution than TGC and RPC, for a refined p_{T} estimation. The filtered candidates are then passed back to the Sector Logic, and the final muon trigger decisions are forwarded to Level-0 MUCTPI.

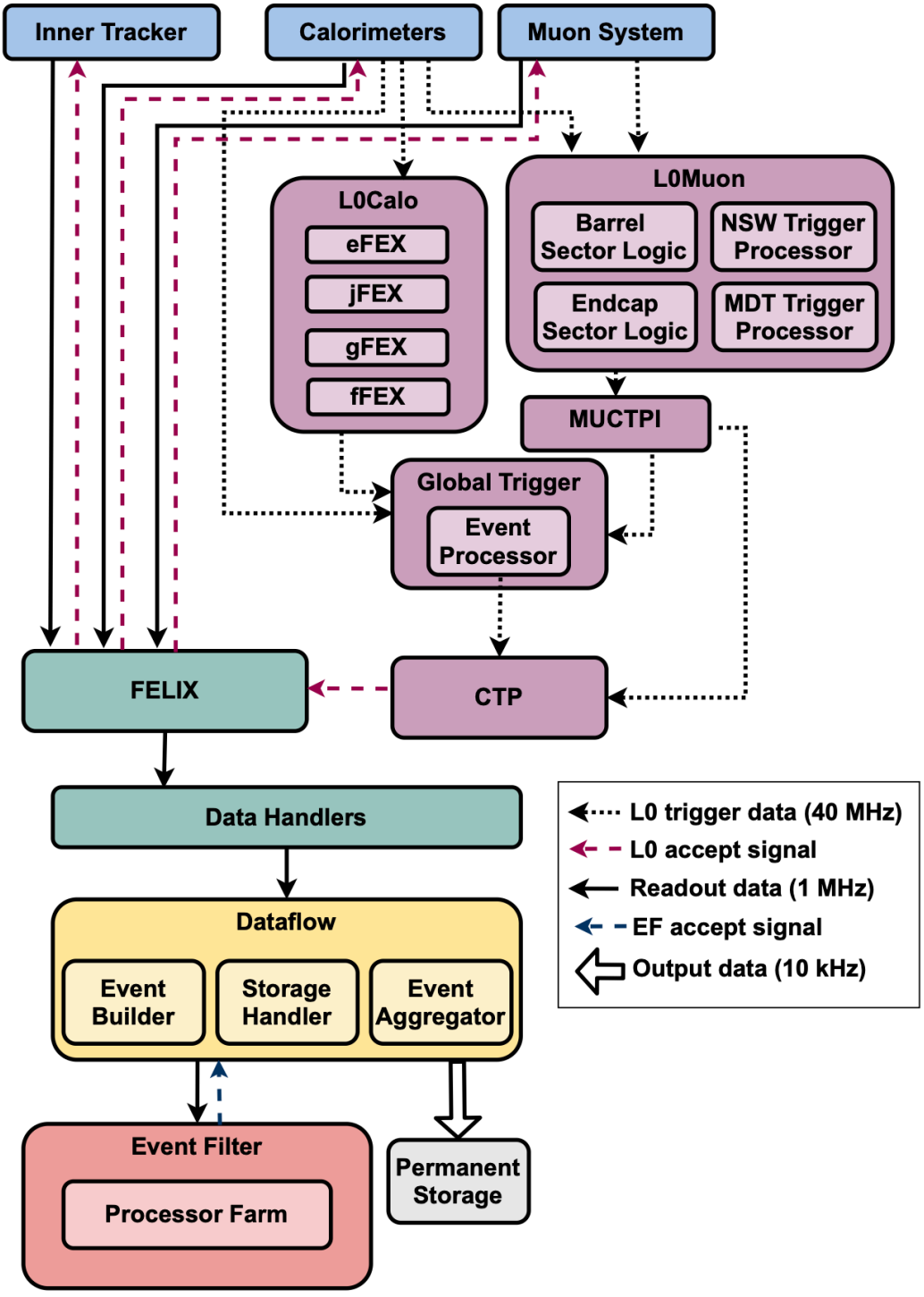


Figure 2.14: Baseline design of the upgraded TDAQ system for HL-LHC [10].

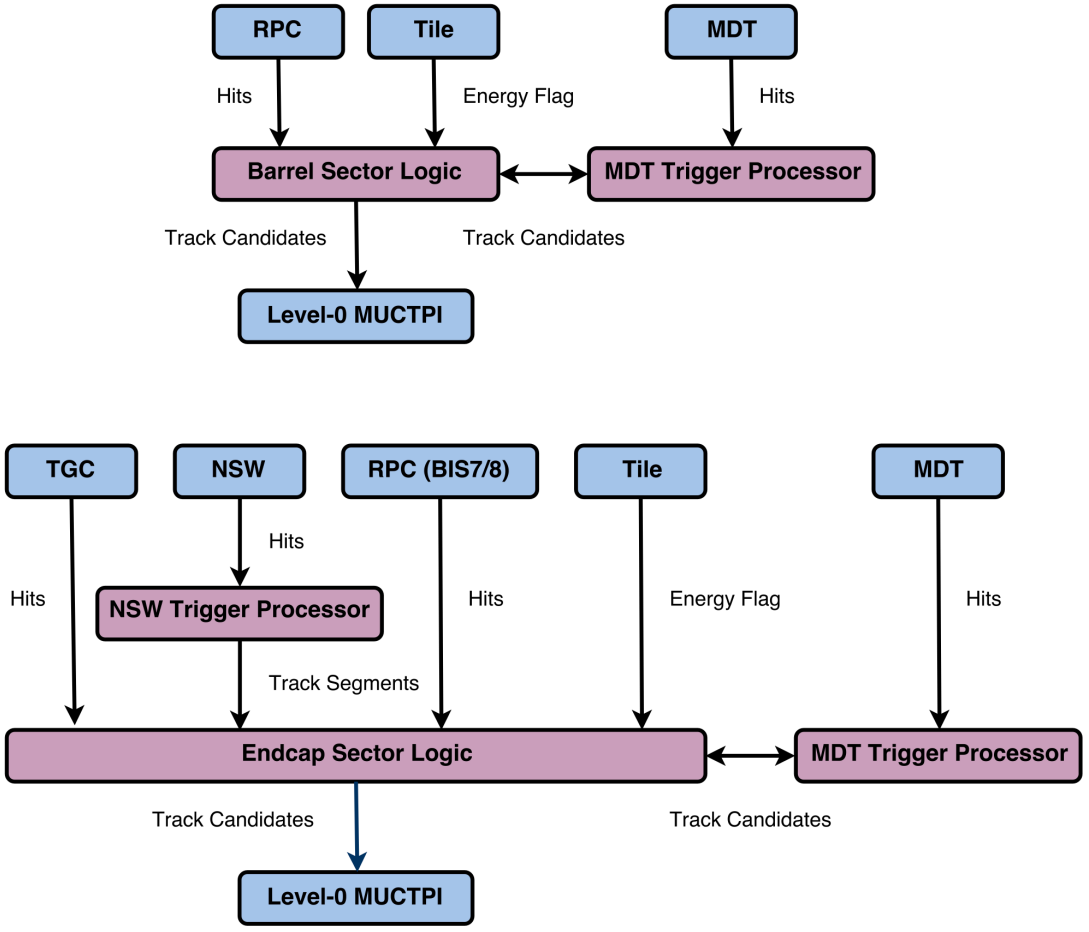


Figure 2.15: Block diagram of the upgrade strategy of Level-0 muon trigger system at HL-LHC [5].

3

The Athena Framework

The ATLAS software framework supports experimental data transformation, Monte Carlo generation and simulation, and downstream analysis of the ATLAS detector data. The framework includes several projects, the most comprehensive one among them forms the basis of Athena, a general-purpose offline software framework based on Gaudi architecture [11]. This chapter briefly introduces the Gaudi architecture, followed by an overview of the Athena framework built on it. Besides, new development to multi-threaded Athena are also presented.

3.1 The Gaudi Architecture

The Large Hadron Collider (LHC) generates petabytes of raw data each year. To extract meaningful physics insights, these data must be processed through multiple stages such as reconstruction of kinematical variables by detector signals, filtering and selection of data of interest, and offline analysis for physics purposes. Since the experiments are planned to continue for many years, it is crucial to anticipate evolving software requirements and advancements in the underlying technologies. Therefore, the software must be designed

with enough flexibility and adaptability, allowing it to accommodate these changes and remain maintainable over extended periods of operation.

To address that, a new object-oriented software framework for High Energy Physics, Gaudi, was developed [12]. Originally designed for the LHCb experiment, Gaudi provides a modular and flexible environment for building various data-processing applications across various computing platforms. As the core of the framework, Gaudi is based on a well-defined architecture that specifies the main components and their interactions. Figure 3.1 shows the major components of the Gaudi software architecture. The architecture of the Gaudi framework is introduced in the following.

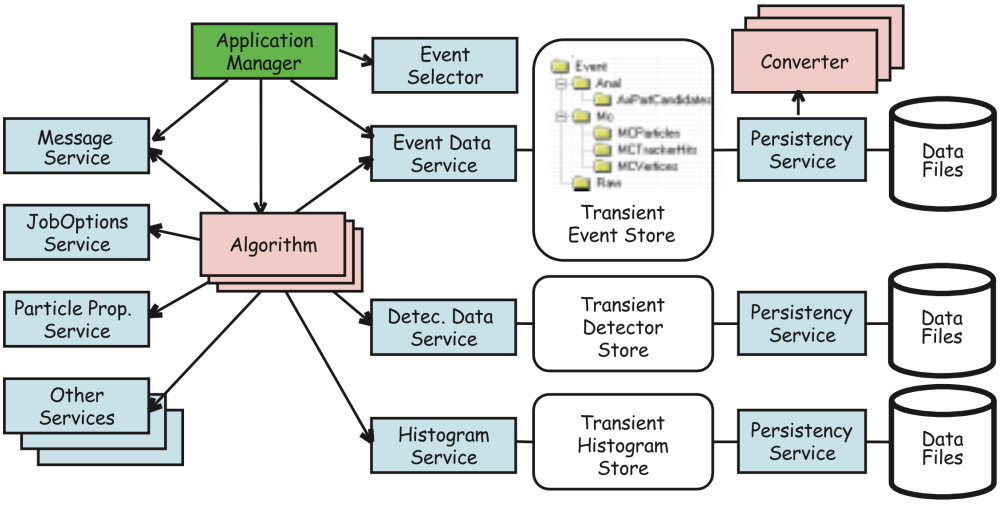


Figure 3.1: Object diagram of the Gaudi architecture [12].

3.1.1 Algorithms and Application Manager

In event data processing, the core functionality is realized through *physics algorithms*, which are encapsulated as modular components referred to as algorithms. The algorithms implement a standard set of interfaces, allowing them to be invoked (called) without requiring knowledge of their internal workings. More complex functionalities can be constructed by composing simpler algorithms. Overseeing the algorithm execution flow is the *application manager*, responsible for instantiating and orchestrating algorithms as needed.

The execution of algorithms follows an explicit scheduling model. A complex algorithm manages the order of their sub-algorithms to ensure correct results. If a particular algorithm relies on data produced by another, it becomes necessary to explicitly define the

execution order to maintain consistency. Figure 3.2 shows how proper sequencing and the use of a transient data store (see below) enables coherent data flow across different algorithmic stages.

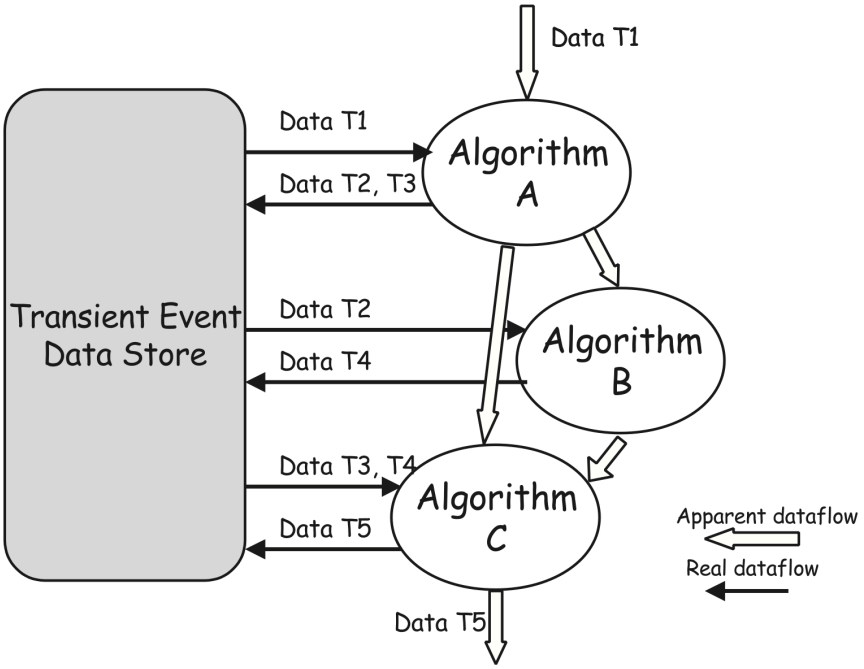


Figure 3.2: A demonstration of achieving the intended dataflow via structured scheduling of algorithmic components [12].

3.1.2 Transient data stores

The Gaudi framework uses several *transient data stores* to manage the exchange and lifecycle of data between algorithms. They are applied depending on the nature and lifetime of the data:

- **Transient Event Store** handles event data that are valid only during the processing of a single event.
- **Transient Detector Store** contains data that describe various aspects of the behavior of the detector, which typically persist across many events.
- **Transient Histogram Store** holds statistical data, and generally lasts across the processing of a complete job.

The purpose of the Transient Store is to minimize the coupling between algorithms and data objects. Algorithms can store intermediate results into the transient store, and other algorithms can access this data later without needing to know how it was produced. In this way, different algorithms communicate indirectly via a shared data space. The Transient Store serves as an intermediate buffer between different data representations, responsible for conversion from transient data to persistent or graphical formats.

3.1.3 Services

Services in the Gaudi framework are a category of components that provide all the services and functionalities required by the algorithms, either directly or indirectly. This architectural approach releases many software routine tasks from the algorithm developer, enabling them to focus on physics data processing logic. These services are briefly introduced as below, some of which could be seen in Figure 3.1.

Some services are responsible for managing transient data stores, including the event data service and detector data service, etc. These services simplify data access and ensure efficient communication between different components of the framework. In addition, the different persistency services provide the functionalities in managing the transformation of data between transient and persistent representations. These transformations rely on specific *converters*, which are capable of converting a given data object into its appropriate format. Auxiliary services are also provided by Gaudi, such as the job options service, the message service, particle properties service and other services such as visualisation and event selectors.

3.2 Athena

Although Gaudi was originally developed within the context of the LHCb experiment [13], it was designed to be highly customizable and adaptable to various tasks, making it suitable for integration into the software environments of other experiments. The Gaudi framework is now shared by many particle physics experiments, in which the ATLAS is included [14]. Here, a brief overview of ATLAS control framework based on Gaudi architecture, *Athena*, is presented.

Athena is the object oriented control framework used by the ATLAS experiment at CERN. It is developed in C++, and is designed with a modular component architecture, consisting of a series of packages covering all the main processes along the data flow. It is also

supplemented by external libraries such as POOL, a data storing service, and Geant, a software package for MC simulation. The framework enforces a clear separation between transient and persistent data. Components access data through abstract interfaces rather than direct access to the implementation. This allows individual components to be easily replaced or updated as technologies advance. Athena comprises the ATLAS specific extensions to Gaudi, which includes:

- **StoreGate** - a transient data store used for exchanging information between algorithms during processing [15].
- **Interval of Validity Service (IOVSvc)** - handles time-dependent conditions and detector data.
- **Pileup** - supports the simulation of multiple interactions within a single bunch crossing to approach realistic experimental conditions.
- **History Service** - maintains a multi-level record of data provenance, enabling traceability and reproducibility of reconstructed data.
- **Python Scripting** - provides configuration and interactive control of Athena components based on Python.

In a data processing flow on Athena, dynamically loadable components are employed, leading to the concepts of Algorithms, Services, and Tools introduced in Section 3.1. The processing flow is illustrated as Figure 3.3.

Algorithms operate on data reside in a shared event store, where they retrieve and store objects identified by type and a string key. In principle, each Algorithm is stateless with respect to event data and communicates with others solely through the event store. Services are shared resources accessed by multiple components, such as the event store itself, error logging, and random number generation. Tools act as auxiliary components and can be uniquely owned by Algorithms, Services, or even other Tools. Each of the three component types, namely the Algorithms, Services and Tools, supports the declaration of configurable properties, allowing consistent initialization as part of the job setup phase.

3.3 Multi-threaded Developments for Athena

The Athena framework was initially developed in the early 2000s, when the processing speed is determined mainly by that of single-core CPU clock speeds. As such, it was fundamentally designed for serial event processing. During LHC Run 2, 2015 to 2018,

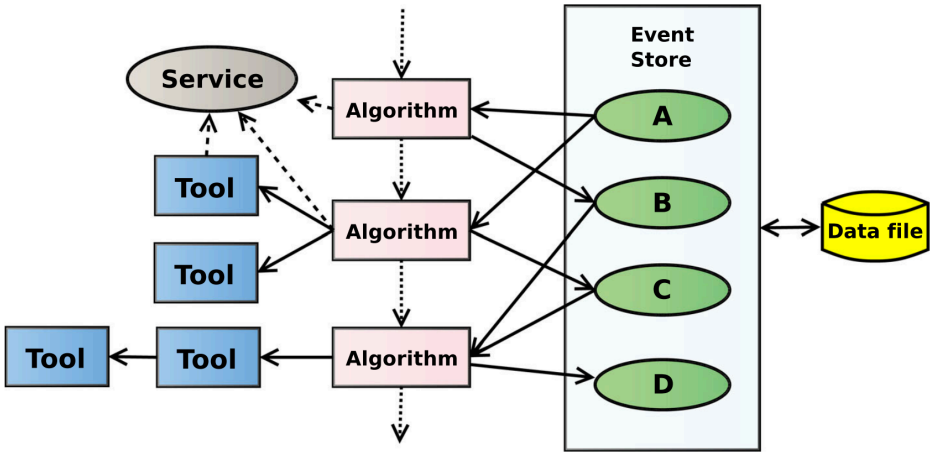


Figure 3.3: Schematic of Athena processing flow. The solid lines on the right indicate data flow, on the left they indicate ownership. The dotted line indicates the application control flow. The dashed lines indicate a non-owning reference between components [11].

increasingly demanding computing conditions were addressed through the development of *AthenaMP* [16], a multi-process version of Athena. AthenaMP operates by forking multiple worker processes from a primary process after the initialization phase. These workers run the event loop in parallel, allowing large static memory structures, such as detector geometry and magnetic field maps, to be shared via the Linux kernel's copy-on-write mechanism.

However, this approach also presents limitations. Minor changes in memory, such as modifying a single bit, can cause entire memory pages to become unshared, negating the benefits of memory sharing. Furthermore, the C++ memory model does not allow for fine control over which data are assigned to which physical pages. In addition, with the further evolution of the LHC, event complexity and data acquisition rates are expected to increase significantly. These challenges motivated the development of *AthenaMT*, a multi-threaded version of the Athena framework in LHC Run 3 phase. In multi-process (MP) parallelism, worker processes are forked from a primary process a pre-configured stage in execution (e.g., before or after the first event is processed). After forking, workers share memory pages allocated in the primary process but otherwise execute independently in parallel. Each worker has its own private memory region and produces output separately, requiring a dedicated post-processing step for output merging. The event throughput and memory usage comparison between AthenaMP and AthenaMT are shown in Figure 3.4 and Figure 3.5, respectively.

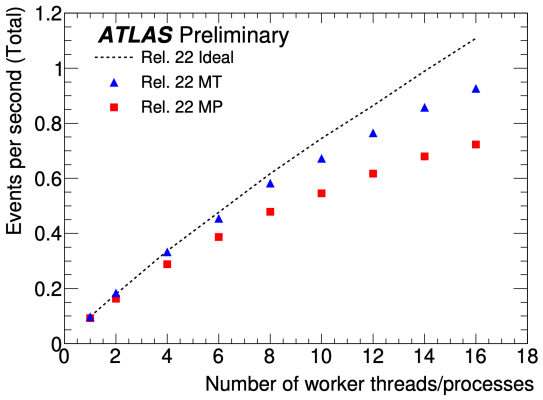


Figure 3.4: Comparison of event throughput of the ATLAS reconstruction as a function of number of threads/processes in Athena release 22 [17].

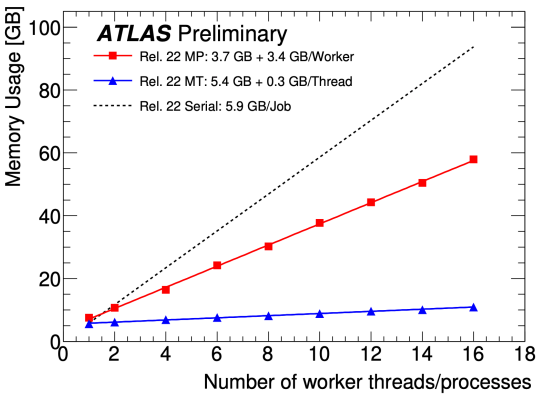


Figure 3.5: Comparison of memory usage of the ATLAS reconstruction as a function of number of threads/processes in Athena release 22 [17].

The multi-threaded framework adopted in Run 3 sets some new requirements for software development and code design [11]. In the Athena implementation of muon trigger simulation for Phase-II upgrade, which will be discussed in Chapter 5, several adaptations were made compared to non-threaded code in order to ensure safe and consistent behavior under multi-threaded environment:

1. **Event and conditions data are accessed via handles.**
   All data access is performed through *handles*, avoiding direct use of non-thread-safe caching or back-channel communication. For example, `SG::ReadHandleKey` and `SG::WriteHandleKey` defined in the `StoreGate` class, are used for retrieving and writing data in StoreGate by using "Keys" corresponding to the specific data being accessed, preventing race conditions when data is accessed concurrently by different threads.

2. **Unique object for each process are applied.**
   Each algorithm instance writes to a separate data object. In a multi-threaded environment, event data must only be modified by the processing thread. Therefore, this implementation uses unique objects for each process, avoiding appending to or modifying existing data objects.

3. **Non-const data structure are avoided.**
   The non-const data structures, including functions and variables, are considered unsafe in multi-threaded processes, since it may cause data conflict if a non-const data structure was shared in several threads. Shared non-const static data can

introduce data conflicts and non-determinism across threads.

4. **Thread-safe Services are used.**
   All Services used in this implementation are explicitly thread-safe. Since Services are global components accessed by multiple algorithms, they are either stateless, properly synchronized (e.g., via status locking), or designed to operate with thread-local data.

Through these measures, the implementation can be operated with thread safety on the Athena multi-threaded environment.

4

Development of a Simple Emulator for L0 Muon Trigger System

This chapter introduces the development of a simple emulator, emulating the L0 muon trigger behavior on both barrel and endcap region, called L0MuonEmulator. The L0MuonEmulator is a C++ based software package on Athena that emulates the trigger decision for muons as provided by the L0 trigger system for HL-LHC. The L0MuonEmulator makes decisions of whether a muon candidate should be accepted or rejected based on a pre-set step function of transverse momentum p_{T}, and then outputs smeared p_{T} as the transverse momentum information in ROI (Region Of Interest). In this chapter, an overview of the emulator's design and logic is provided. Since the simulation based solely on smearing was insufficient to replicate the actual trigger logic, a masking procedure to make the trigger performance is also introduced to reproduce the angular acceptance of the L0 muon trigger system. The result of the emulation is presented along with a discussion about the need for a more precise simulation approach.

4.1 Purpose and Function of the Simple Emulator

The purpose of the L0MuonEmulator is to provide a temporary input interface for downstream development, the Event Filter (EF), which we introduced in Chapter 2. This emulator provides p_{T} and the ROI information of muons for further selection by EF. At present, the emulator takes the truth values of the four-momentum for all generated particles in a simulated event as an input.

In practice, only final state muons and its anti-particles are used. These selected muons and anti-particles are then passed through a smearing and filtering procedure that models the detector resolution and trigger efficiency, providing a collection of RoI information based on the emulated L0 muon trigger. Each RoI contains information on transverse momentum (p_{T}), pseudorapidity (η), azimuthal angle (ϕ), and charge.

The filtering function in the L0MuonEmulator simulates the trigger efficiency as a function of muon transverse momentum. A simplified step-function model is implemented, in which muons below a certain threshold are discarded, while muons above that threshold are accepted with an efficiency of 95%. This 95% efficiency is approximately determined based on the efficiency of endcap in L1 muon trigger in Run 3 and the efficiency of barrel simulated for Phase-II upgrade, as shown in Section 4.4. In addition, the emulator includes a basic geometric masking procedure to exclude regions outside the muon detector acceptance. Currently, this is implemented by discarding all truth muons with pseudorapidity $|\eta| > 2.41$, which roughly corresponds to the coverage limit of the L0 muon trigger chambers in ATLAS. Truth muons outside this region are automatically rejected.

The smearing algorithm emulates the detector resolution by applying a smearing function to the inverse transverse momentum (q/p_{T}) of the selected muons. Currently, a Gaussian distribution is used as the smearing function, tentatively with a width of 5% of the input q/p_{T}. The η and ϕ positions are not smeared at this stage, as the RoI granularity is coarser than the smearing scale, which thereby dominates the uncertainty. Once smeared, the kinematic variables are encoded into a bit-wise RoI word format compatible with the hardware-level representation used in the ATLAS trigger system.

In addition, the L0MuonEmulator provides a monitoring function based on the Athena Monitoring Framework. A series of histograms are produced during the execution to validate the emulator output with the input truth muon properties. These include distributions of η, ϕ, p_{T}, and q/p_{T} before and after smearing, as well as resolution plots such as $\Delta\eta$, $\Delta\phi$, and relative p_{T} deviation.

4.2 Muon Trigger Acceptance after Phase-II Upgrade

As discussed in Chapter 2, at the barrel region of the ATLAS detector, there are some dead zones in the RPC detector, affecting the trigger acceptance and efficiency. Therefore, a new station of RPC detector in the BI region will be added in Phase-II upgrade. This additional RPC0 enables a four-layer RPC configuration and allows for more flexible coincidence schemes. A summary of the logic combinations is provided in Table 4.1. Three coincidence schemes are considered "3/3 chambers", "3/4 chambers", and "3/4 chambers + BI-BO", which are defined by the number of required stations to register hits of the barrel detector.

The performance of muon trigger acceptance in the barrel region for "3/3 chambers" in Run 3 are shown in Figure 4.1a. Figure 4.1b and Figure 4.1c shows the acceptance for "3/4 chambers", and "3/4 chambers + BI-BO" schemes in Run 4, respectively. From these pictures, we can see that the trigger acceptance gradually improves through the Phase-II upgrade, with the highest coverage achieved in the "3/4 chambers + BI-BO" pattern.

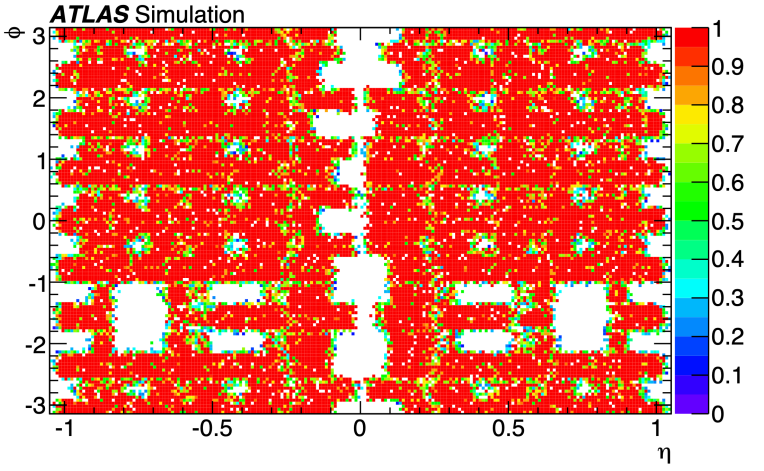
Table 4.1: Hit requirements for different RPC trigger coincidence modes.

Requirement	RPC0 (BI)	RPC1+2 (BM)	RPC3 (BO)
3/3 chambers	–	3 out of 4	1 out of 2
3/4 chambers	–	3 out of 4	1 out of 2
	2 out of 3	3 out of 6	
3/4 chambers + BI-BO	–	3 out of 4	1 out of 2
	2 out of 3	3 out of 6	
	2 out of 3	–	1 out of 2

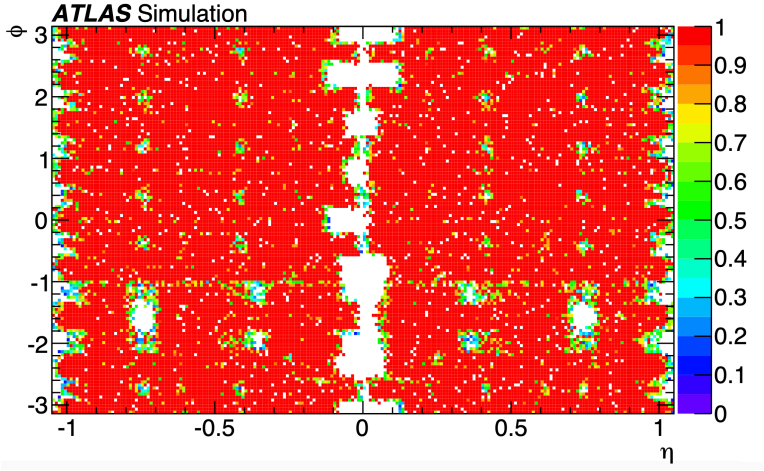
4.3 Simulation for the trigger acceptance

Section 4.2 above shows that even applying the coincidence scheme with highest acceptance, some unavoidable holes remain due to mechanical limitations. In this section, a simple simulation that emulates the barrel muon trigger acceptance in Run 4 are implemented. This emulates the "holes" caused by mechanical structures based on the "3/4 chambers + BI-BO" scheme, which shows the highest coverage.

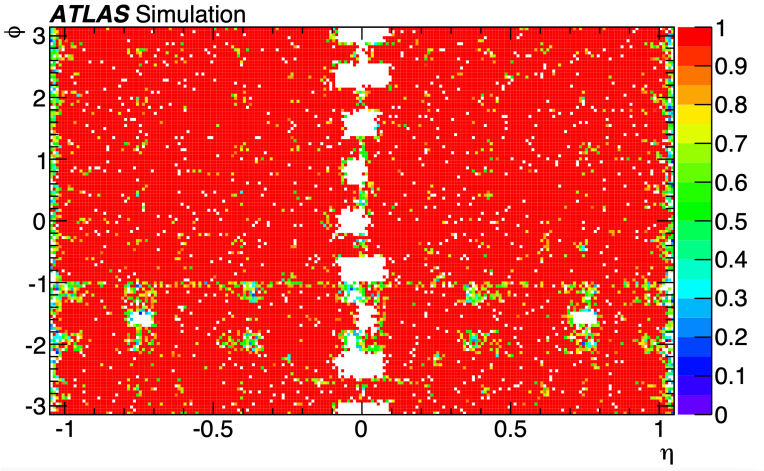
As described in Section 4.1, the original implementation adopted a simplified geometric acceptance cut by discarding all truth muons with pseudorapidity $|\eta| > 2.41$, roughly



(a) 3/3 chambers



(b) 3/4 chambers



(c) 3/4 chambers + BI-BO

Figure 4.1: Trigger acceptance maps in η-ϕ space for different coincidence logics [5].

corresponding to the nominal detector coverage of the L0 muon chambers. To further filter those "holes", specific (η, ϕ) masking scheme corresponding to uncovered regions are manually excluded within the smearing algorithm. Figure 4.2 shows those excluded regions, which are enclosed by blue dashed lines.

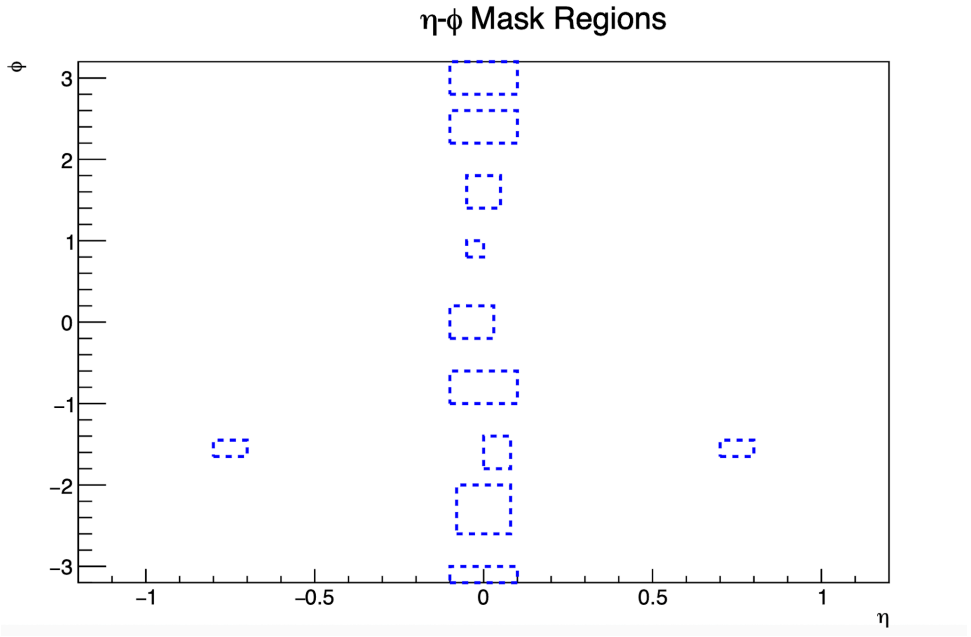


Figure 4.2: Masked region for simulation of barrel muon trigger acceptance.

4.4 Performance of the L0MuonEmulator

In this section, the efficiency of the L0MuonEmulator is compared to the references of the trigger efficiency in Run 3 for endcap, and Run 4 for barrel with higher efficiency being performed. The data file used for the efficiency calculation is a Monte Carlo sample of $Z \to \mu^+\mu^-$ events at a center-of-mass energy of 13.6 TeV, with a total of 60,000 events.

Figure 4.3 shows the muon trigger efficiencies in the endcap region with several p_{T} thresholds, using experimental data from LHC Run 3. Figure 4.4 gives the barrel muon trigger efficiencies with four steps of p_{T} thresholds based on simulation for Phase-II upgrade. Figure 4.5 and Figure 4.6 show the efficiencies of the L0MuonEmulator in endcap and barrel regions with five thresholds, respectively. The efficiency is defined in Equation 4.1.

$$\text{Efficiency} = \frac{\text{Muons selected by the filtering function}}{\text{All the truth muons}} \tag{4.1}$$

As a result, the turn-on curves are reproduced by the L0MuonEmulator, showing a fair agreement with the references.

Figure 4.7 gives the two-dimensional (η, ϕ) acceptance maps: the reference map from Phase-II trigger studies (top) and the efficiency map generated by L0MuonEmulator (bottom). Dashed blue boxes in the panel below indicate the manually masked regions like Figure 4.2. Only obvious region around $\eta = 0$ and $|\eta| = 0.7$ are qualitatively reproduced, while other smaller inefficiency regions are not reproduced. As a result, obvious "holes" are simulated by the masking scheme, especially those areas around $\eta = 0$.

In conclusion, according to the efficiencies on both p_{T} and (η, ϕ) plane, the behavior of the efficiency is not well reproduced, since it merely parameterize the efficiency, assuming that it is uniform across the angular range except for the holes implemented. Therefore, a complete simulator of high precision that simulates the trigger logic in full detail is still necessary for the Athena implementation. This motivates the implementation of the simulator for TGC Sector Logic, which will be introduced in the next chapter.

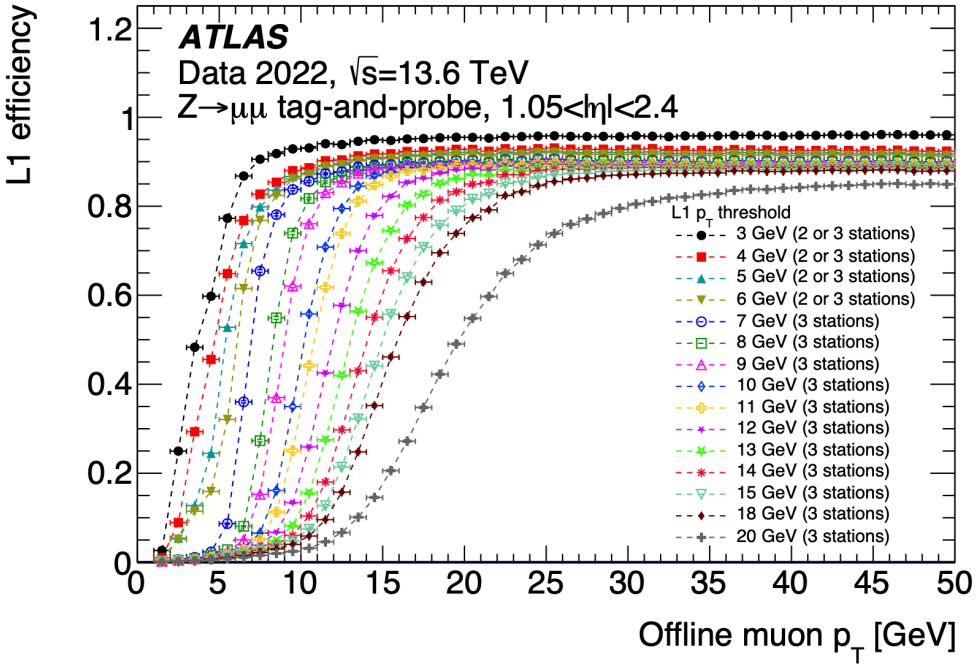


Figure 4.3: Efficiency of L1 muon triggers in the endcap region for several p_{T} thresholds [9].

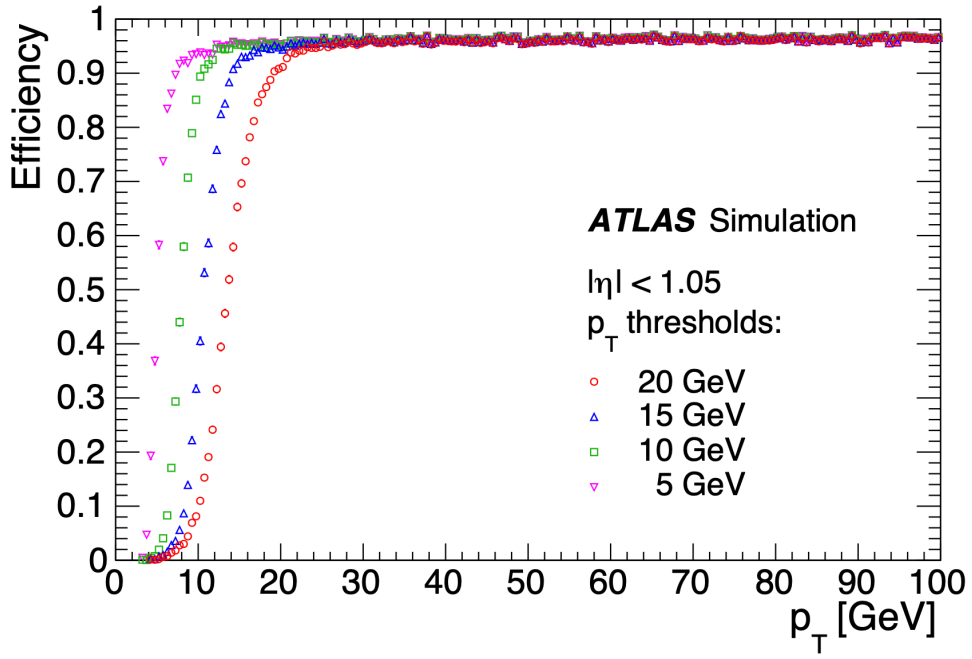


Figure 4.4: Efficiency of RPC L0 muon triggers in the barrel region for different p_{T} thresholds using "3/4 chambers + BI-BO" scheme [5].

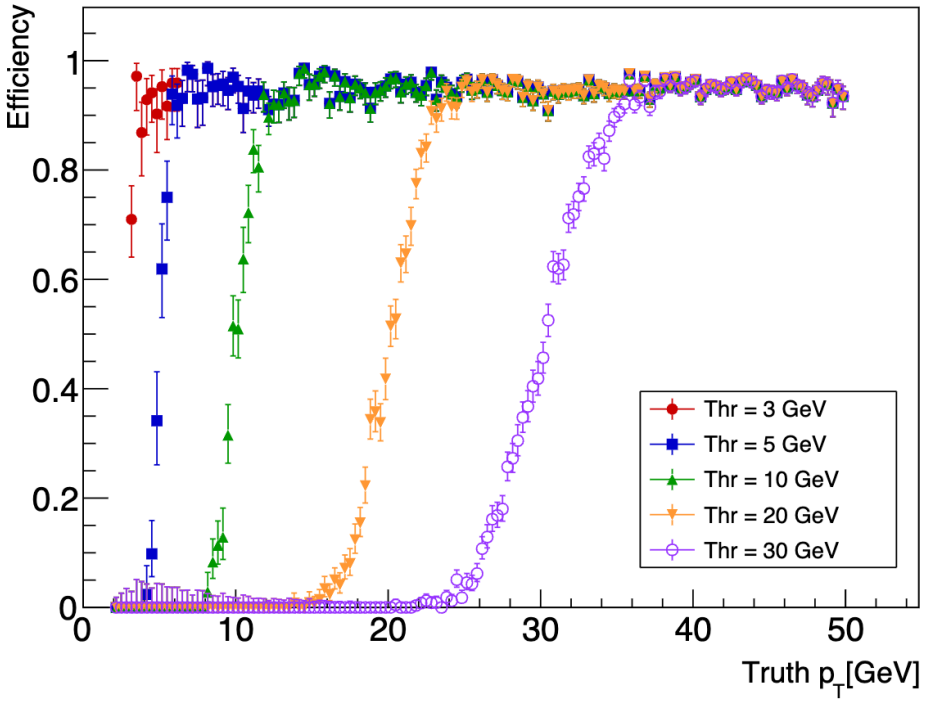


Figure 4.5: Efficiency of L0MuonEmulator in the endcap region for five p_{T} thresholds: 3 GeV, 5 GeV, 10 GeV, 20 GeV, 30 GeV.

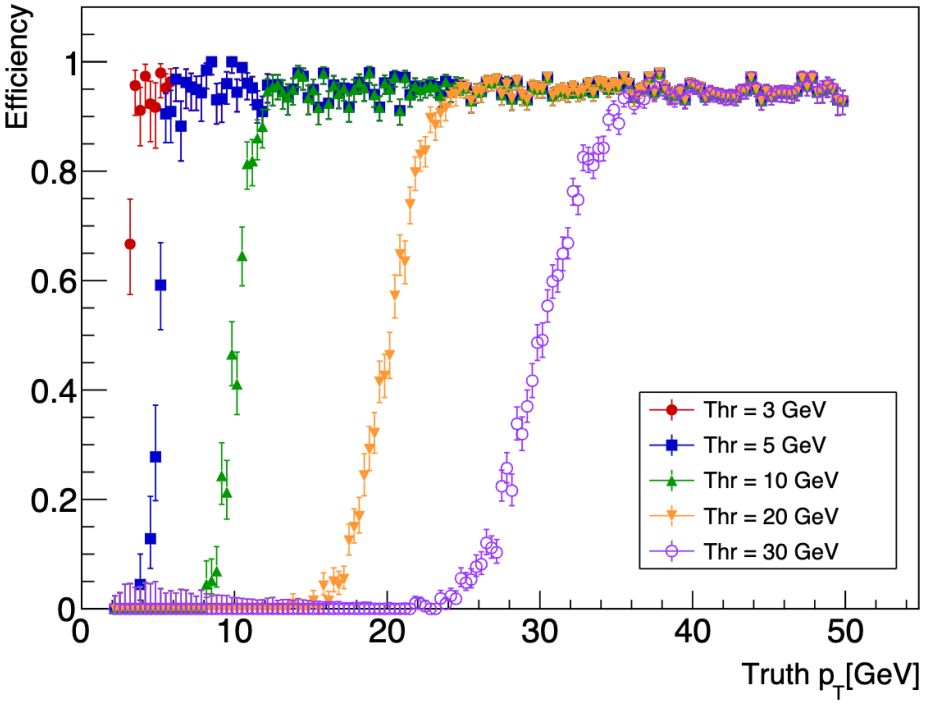


Figure 4.6: Efficiency of L0MuonEmulator in the barrel region for five p_{T} thresholds: 3 GeV, 5 GeV, 10 GeV, 20 GeV, 30 GeV.

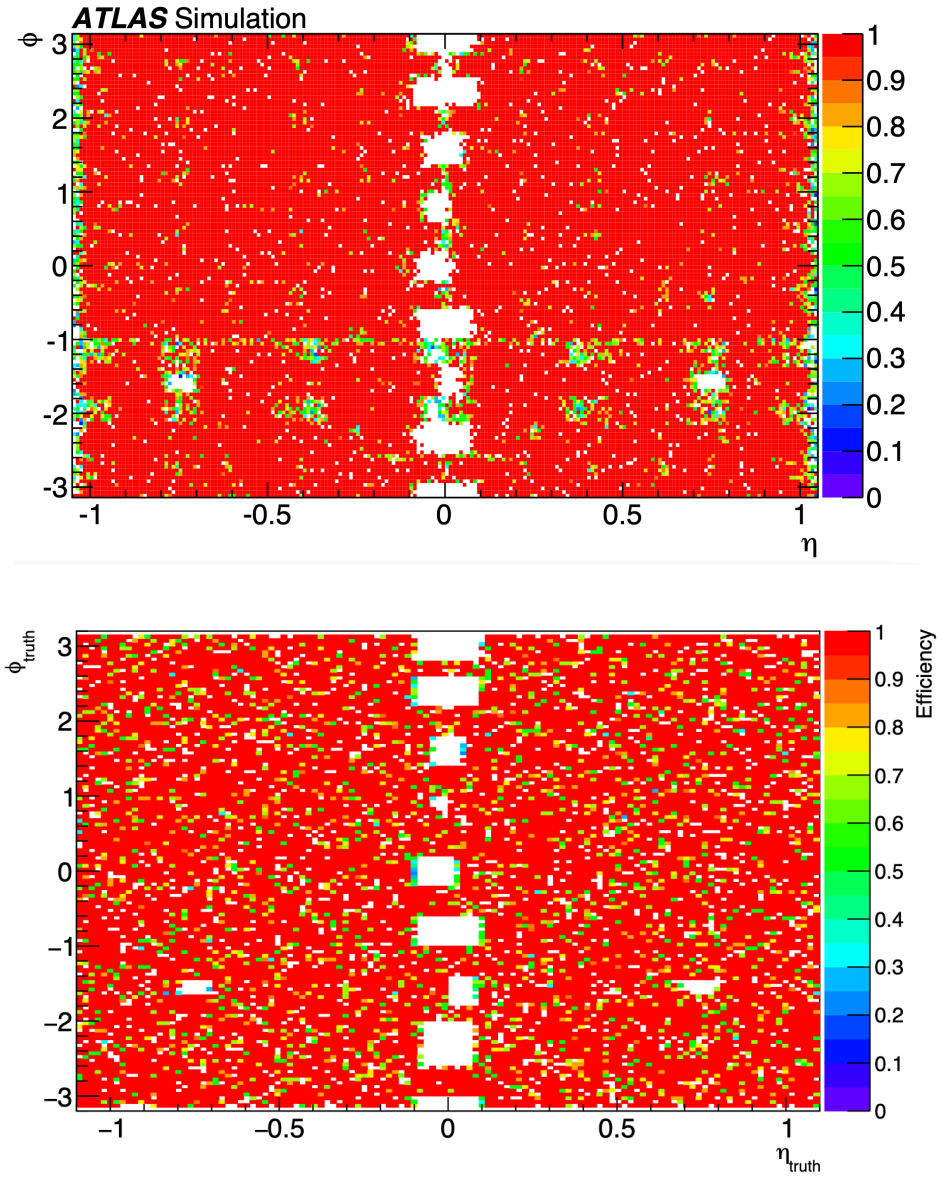


Figure 4.7: Comparison of trigger acceptance maps in the (η, ϕ) plane of barrel region. Top: Reference acceptance from Phase-II studies. Bottom: Emulator efficiency map including manually masked regions with p_{T} threshold of 3 GeV.

5

Implementation of a Simulator for L0 Endcap Muon Trigger System

This chapter presents the implementation of the simulator for the TGC Sector Logic (hereafter referred to as L0TGCSimulator), which performs the fast reconstruction of muon tracks and their momenta in the endcap region of L0 muon trigger system. This L0TGCSimulator is for a precise simulation, based on an existing bit-level simulator called "bitwise simulator", which emulates the firmware logic of the TGC SL. The chapter begins with the description of TGC SL firmware behavior, including its hit processing and reconstruction logic. It then brief summarizes the implementation of the bitwise simulator into Athena environment, followed by a memory optimization technique applied. Finally, the trigger efficiency performance of the implemented simulator is evaluated.

5.1 Firmware Behavior in TGC Sector Logic

Figure 5.1 shows the block diagram of the trigger logic of the endcap SL firmware. The firmware consists of four Super Logic Regions (SLRs) on FPGAs. Among them, SLR0, SLR2, and SLR3 are for simulating muon candidates in the Endcap 1, Endcap 2, and

Forward regions, respectively, as illustrated in Figure 5.2. SLR1 functions as a post-processing stage for the other three SLRs, forwarding the reconstructed results to the Inner Coincidence module in order to suppress muons that do not originate from the initial proton-proton collision point. TrackSelector selects muon candidates with higher p_{T} and forwards them to the MUCTPI for trigger processing.

5.1.1 Channel Mapping

Each SL corresponds to one of the 1/24 repetitive segments of the TGC Big Wheel (BW). Each SL board receives hit information from 29 Primary Processor boards (PS boards) via two optical links. Each link transmits 128 bits of data every 25 ns, resulting in a complete input to the Sector Logic in the form of a 128-bit $\times$ 58-link bitmap.

As the first stage of the TGC SL, the Channel Mapping module is responsible for converting this 128 $\times$ 58 bitmap input, originated from the PS boards, into a format suitable for the next processing stage, the Intra-Station Coincidence. This conversion process is referred to as *mapping*. During this step, the original bitmap is transformed into logical channel numbers that reflect the geometrical position on the TGC detector. Each hit is represented by a tuple of subsector, layer, and channel, uniquely identifying its position within the detector. This structured representation enables coincidence logic across different layers within the same station in the subsequent logic stage.

5.1.2 Intra-Station Coincidence

As described in Section 2.3.3, in order to achieve higher position resolution, the detector channels within each of the three TGC stations are intentionally staggered relative to the adjacent layers. The Intra-Station Coincidence step reorganizes the hits detected by each layer within a station according to predefined coincidence types. This process yields a Position ID (also referred to as a staggered ID), which serves as the input for Segment Reconstruction. For the Phase-II upgrade, a more flexible coincidence logic has been adopted compared to that of Run 3 (see [19]). Figure 5.3 illustrates an example of the reconstruction method for Intra-Station Coincidence in the M1 station of a TGC Wire segment.

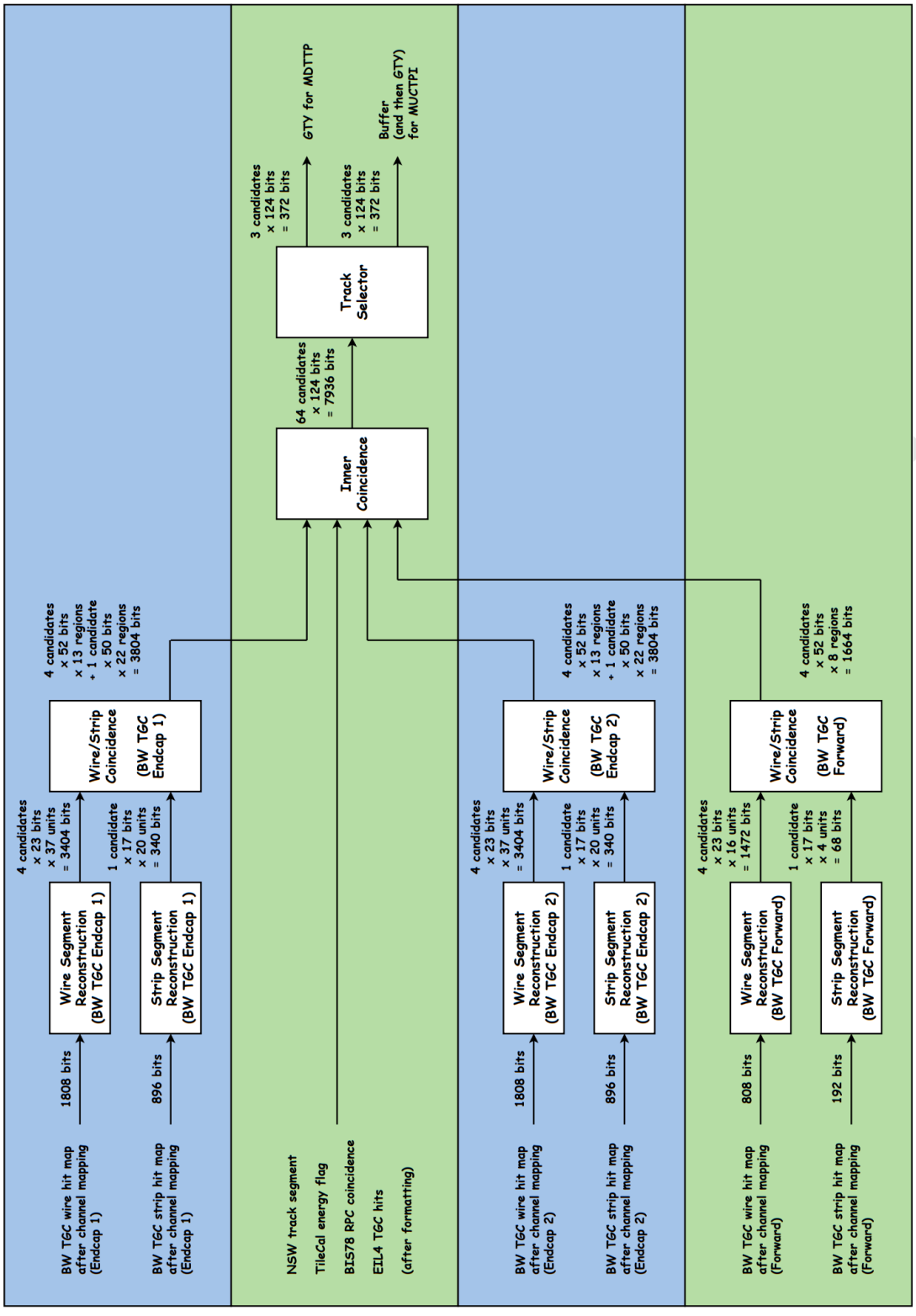


Figure 5.1: Block diagram of the Trigger Logic of the Endcap SL firmware [18].

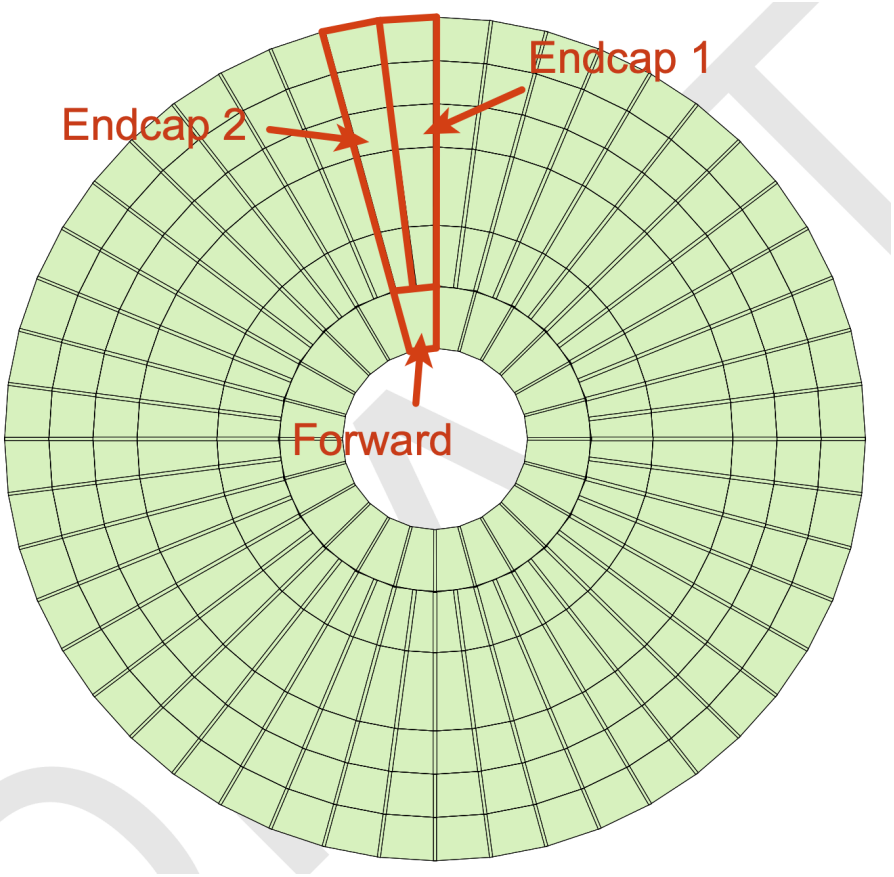


Figure 5.2: Schematic view of Big Wheel (BW) TGC of an Endcap SL blade, which is segmented into the "Endcap 1", "Endcap 2", and Forward trigger sectors [18].

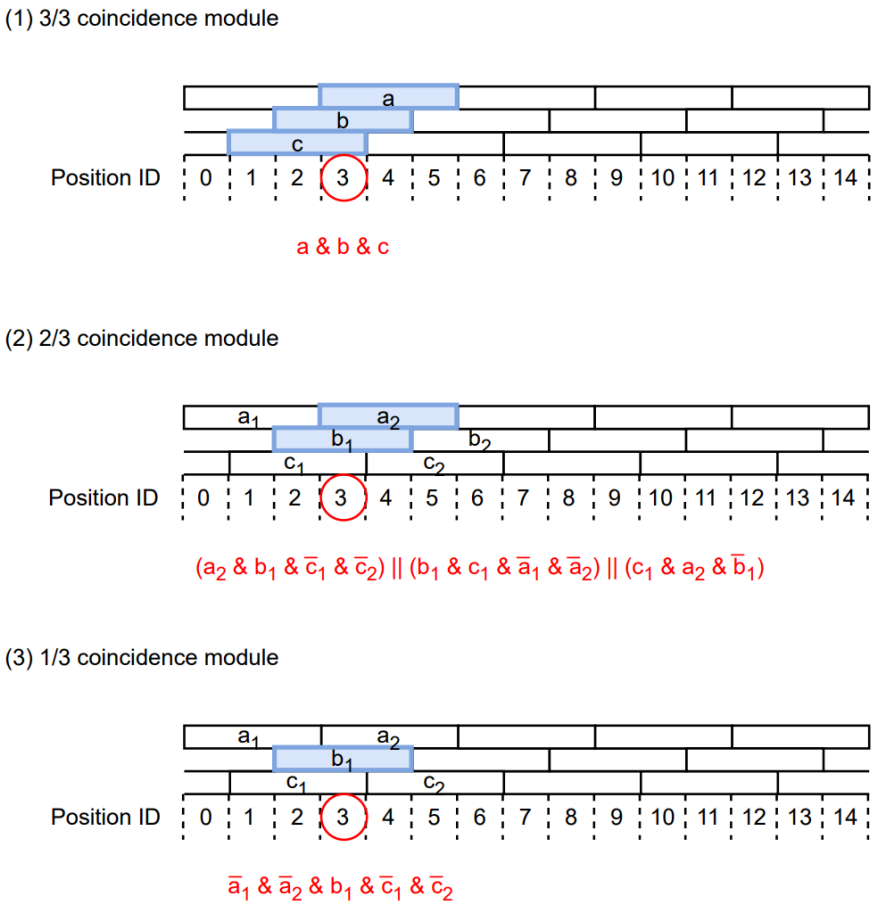


Figure 5.3: Concept of the station coincidence for TGC wire channels in M1 (triplet) [18].

5.1.3 Segment Reconstruction

Using the combination of Position IDs calculated from the Intra-Station Coincidence of the three stations M1, M2 and M3 in the previous step, a pre-calculated Look-Up Table (LUT) is queried to obtain the corresponding deviations of the angle of the segment connecting the three points from an infinite-momentum trajectory, denoted as $\Delta\theta$ (or $\Delta\eta$) for wire and $\Delta\phi$ for strips, as illustrated in Figure 5.4. These values, $\Delta\theta$ and $\Delta\phi$, represent the curvature of the muon's trajectory under the influence of the toroidal magnetic field within the detector, with the position information of hits, are passed to the subsequent Wire-Strip Coincidence process for momentum calculation.

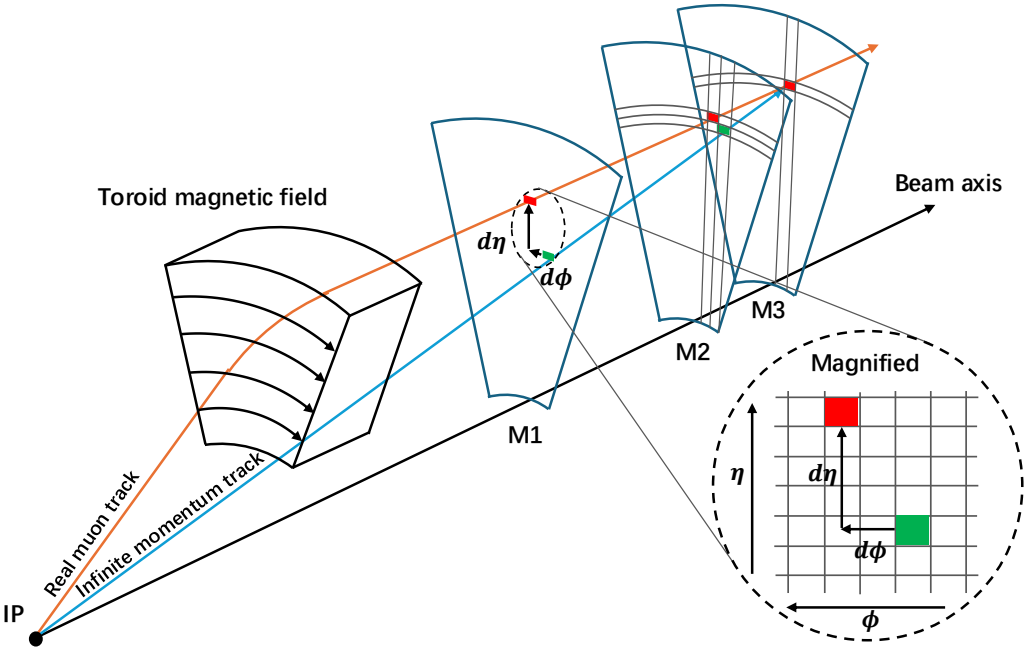


Figure 5.4: Schematic of the reconstruction logic of the endcap muon trigger.

The LUT is preloaded with possible combinations of Position IDs and their corresponding $\Delta\theta$ and $\Delta\phi$ values to eliminate possible redundant computations from iterative calculation and accelerate the reconstruction process. In the firmware, the LUT is accessed via UltraRAM (URAM), a form of high-speed random-access memory.

The Segment Reconstruction step is executed independently for the wire and strip segments. The details of each are described as follows.

Wire Segment Reconstruction

For the wire segment, the three stations, M1 (triplet), M2 (doublet), and M3 (doublet), have different number of gas layers. Therefore, the reconstruction logic must distinguish between triplet and doublet geometries. Moreover, not all possible combinations of Position IDs are included in the LUT, since some of the combinations would correspond to unrealistically large bending angles, which are unlikely to result from real muon trajectories. To constrain the valid combinations, the concept of *Units* and *Subunits* is introduced.

The different coincidence pattern are used for the wire segment reconstruction depending on the positions of the Intra Station Coincidence. They are subdivided into 90 parts called *Units*. Each Unit covers:

- 8 wire channels per layer in M3,
- 16 wire channels per layer in M2,
- 32 wire channels per layer in M1.

These Units are designed to capture muon trajectories (both μ^+ and μ^-) with transverse momenta down to 4 GeV. Each Unit is further subdivided into four *Subunits*. The coverage of a typical Unit is illustrated in Figure 5.5. The summary of coincidence patterns used in wire segment reconstrutction is shown in Table 5.1.

Table 5.1: Coincidence patterns for 7-layer wire segment reconstruction. The indices A, B, C, D denote different patterns that share the same number of hit layers.

Coincidence pattern	M1	M2	M3
7/7	3/3	2/2	2/2
6/7 A	2/3	2/2	2/2
6/7 B	3/3	1/2	2/2
6/7 C	3/3	2/2	1/2
5/7 A	2/3	2/2	2/2
5/7 B	2/3	1/2	1/2
5/7 C	3/3	2/2	1/2
5/7 D	1/3	2/2	2/2

Strip Segment Reconstruction

Unlike the wire segment, all three stations (M1, M2, and M3) for the strip segment are doublets and share symmetric geometry for the strip segment. A single unified function is

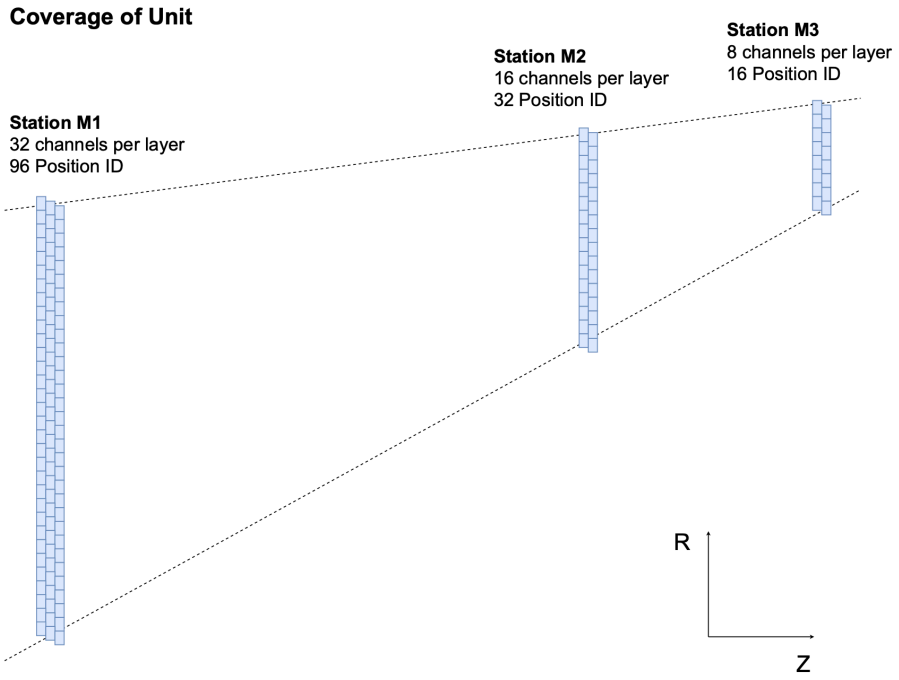


Figure 5.5: Schematic of the coverage of a "Unit" region in three stations of wire segment reconstruction [18].

processing across all stations, therefore. Similar to the wire segments, *Units* and *Subunits* are also defined to structure the reconstruction process.

Similar to the wire segment, The strip segment reconstruction is divided into: 20, 20 and 4 Units in the Endcap 1, Endcap 2 and Forward region, respectively. Each Unit covers:

- 8 strip channels per layer in M3,
- 12 strip channels per layer in M2,
- 20 strip channels per layer in M1.

Each strip Unit is subdivided into two *Subunits*. Figure 5.6 shows the coverage of a typical strip Unit. The summary of coincidence type used in strip segment reconstrutction is shown as Table 5.2.

5.1.4 Wire-Strip Coincidence

The Wire-Strip Coincidence is the final stage of the trigger calculation in the Sector Logic. In this stage, the $\Delta\theta$ and $\Delta\phi$ values, along with the position information θ and ϕ from segment reconstruction are used to determine the transverse momentum (p_{T}) of muons.

Table 5.2: Coincidence patterns for 6-layer strip segment reconstruction. The indices A, B, C denote different patterns that share the same number of hit layers.

Coincidence pattern	M1	M2	M3
6/6	2/2	2/2	2/2
5/6 A	2/2	1/2	2/2
5/6 B	1/2	2/2	2/2
5/6 C	2/2	2/2	1/2
4/6 A	1/2	1/2	2/2
4/6 B	2/2	1/2	1/2
4/6 C	1/2	2/2	1/2

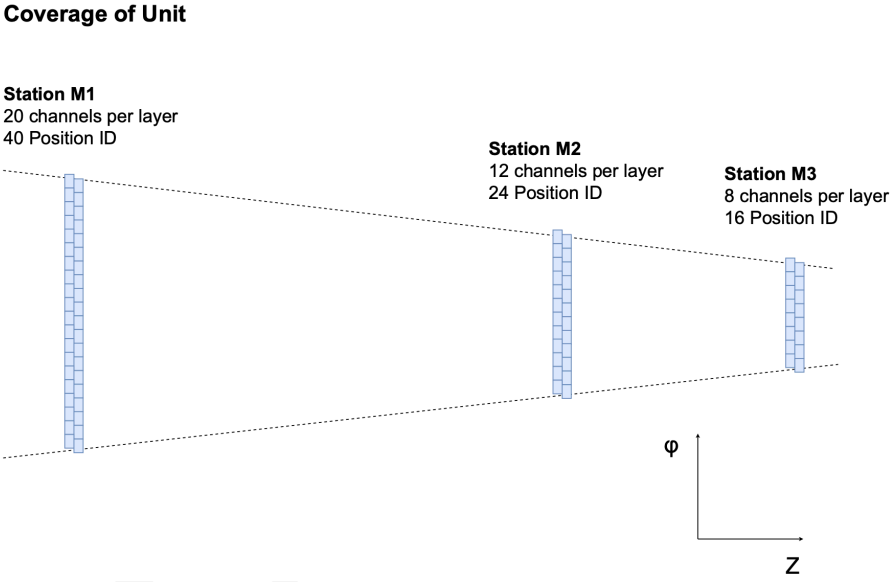


Figure 5.6: Schematic of the coverage of a "Unit" region in three stations of strip segment reconstruction [18].

In the firmware, Wire-Strip Coincidence are divided into three parts: p_{T} Calculator, Wire Position Corrector and Block Seletor, all of which are processed in parallel. These components are all simulated in this simulator, as described below.

p_{T} Calculator

The previously obtained $\Delta\theta$ and $\Delta\phi$ are used to query a LUT determined by θ and ϕ to obtain the p_{T}. The LUT used in this process is also referred as the *Coincidence Window*, where $\Delta\theta$ is represented by the horizontal axis and $\Delta\phi$ by the vertical axis, and the value of p_{T} is encoded as a color corresponding to its magnitude. An schematic of this process is shown in Figure 5.7.

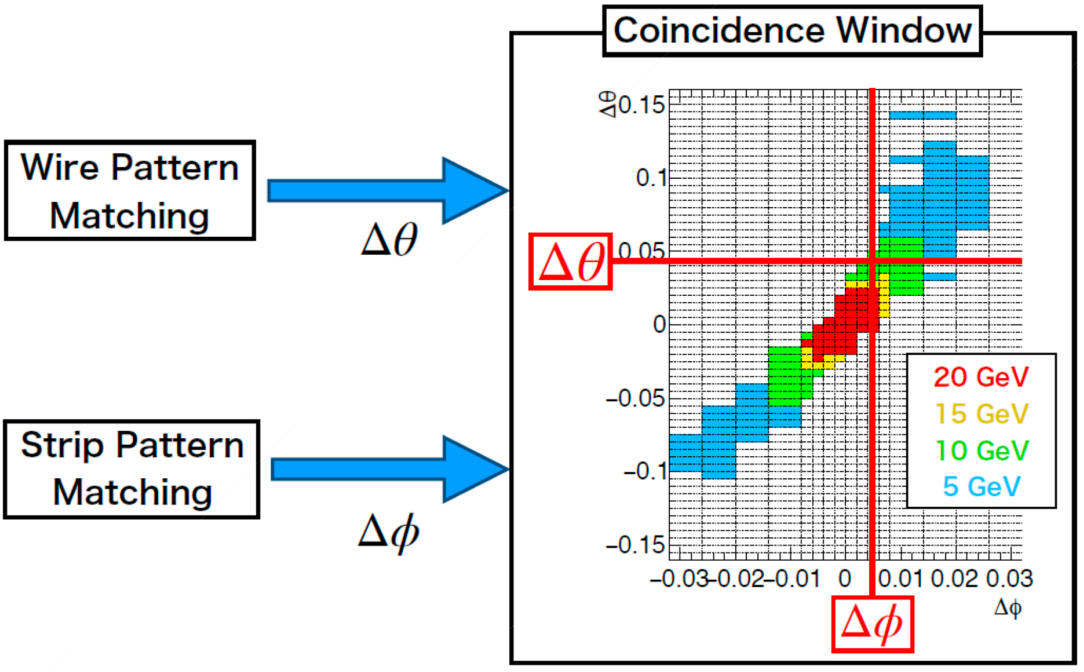


Figure 5.7: Schematic of p_{T} look-up process using Coincidence Window [18].

The address for accessing the Coincidence Window (CW) in RAM consists of three components: 7 bits for the wire-side $\Delta\theta$, 4 bits for the strip-side $\Delta\phi$, and 2 bits indicating the URAM region number, making a total of 13 bits. Although the $\Delta\theta$ and $\Delta\phi$ values provided by the Segment Reconstruction module have widths of 8 bits and 9 bits, respectively, these are reduced to limit the size of the CW. For $\Delta\theta$ of wire segment, the least significant bit is simply discarded, resulting in a 7-bit address input. For the strip side, $\Delta\phi$, consisting of a 1-bit sign and an 8-bit absolute value, is nonlinearly compressed to 4 bits using the scheme in Table 5.3 to retain resolution near zero.

Table 5.3: Mapping between 8-bit absolute value of input $\Delta\phi$ and 3-bit compressed address value used in the p_{T} Calculator.

Input $\Delta\phi$ Absolute Value (8-bit)	Compressed Address Value (3-bit)
0–3	Same as input (0–3)
4–6	4
7–9	5
10–12	6
13 or more	7

As the output, the CW offers a 4-bit transverse momentum value. In the current implementation, five discrete output levels are defined: 0 (invalid), and 1–4 corresponding to transverse momentum thresholds of 5, 10, 15, and 20 GeV, respectively. To further improve precision, the development of a 16-stage CW is currently underway by ATLAS Japan group, as well as the application of machine learning techniques for CW generation.

Block Selector

The inputs to the Wire-Strip Coincidence consist of one wire information per *subunit* and one strip information per *unit*. The combination of the wire subunit and the strip unit is referred to as a *block*. The Block Selector module evaluates these blocks within a specific angular area, called *region*, using the angular information $(\Delta\phi, \Delta\theta)$ to determine which block within the region should be selected among the blocks in that region to give the most appropriate p_{T} value. The Block Selector operates independently of the p_{T} Calculator and Wire Position Corrector (see below), and thus performs a dedicated track segment selection that does not rely on the calculated p_{T} value.

Two types of *regions* are defined here: the *8 Unit Region* and the *32 Unit Region*. An 8 Unit Region consists of 2 wire subunits and 4 strip units, resulting in 8 block combinations. A 32 Unit Region, on the other hand, is composed of 8 wire subunits and 4 strip units, forming 32 possible blocks. As shown in Figure 5.8, in the endcap sectors $\phi 0$ and $\phi 1$, the region $|\eta| < 1.3$ is covered by 22 instances of 8 Unit Regions, while the region $|\eta| > 1.3$ is covered by 13 instances of 32 Unit Regions. In the forward region, 8 instances of 32 Unit Regions are used.

Within the Block Selector, wire and strip segments are reconstructed independently according to the following priority rules:

1. Select the candidate with the largest number of coincidence layers.

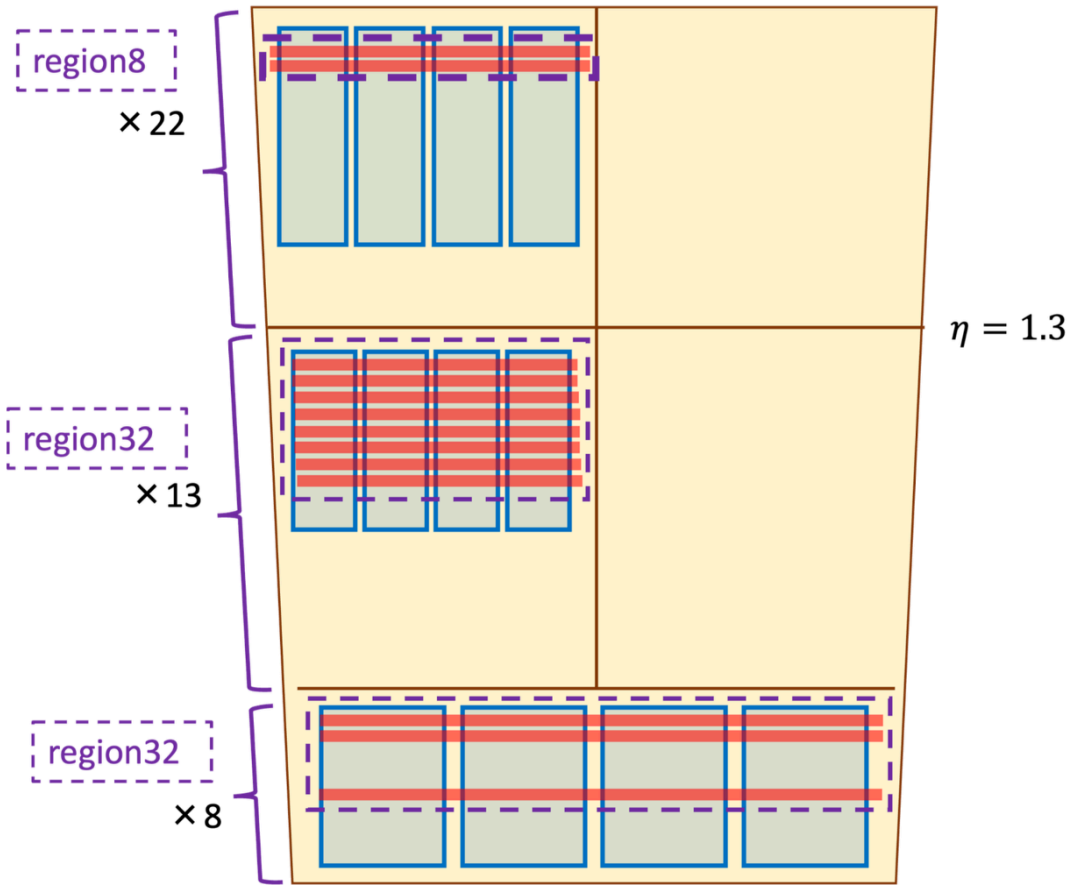


Figure 5.8: Structure of 8 Unit Region (as "region8") and 32 Unit Region (as "region32") in Wire-Strip Coincidence [19].

2. If multiple candidates have the same number of matched layers, choose the one with the smaller absolute value of $(\Delta\phi, \Delta\theta)$.

In an 8 Unit Region: The region can contain up to 4 strip segments and 2 wire segments, yielding $4 \times 2 = 8$ possible block candidates. Based on the priority logic, one strip segment and one wire segment are selected and combined to form a single output block candidate.

In a 32 Unit Region: This region with larger η may contain up to 4 strip segments and 8 wire segments, resulting in 32 possible block candidates. The Block Selector selects segments as below:

- One wire segment with a positive η angle, and one with a negative η angle;
- Two strip segments.

As a result, $2 \times 2 = 4$ block candidates are selected in a 32 Unit Region.

Wire Position Corrector

The Wire Position Corrector is introduced to compensate the geometric deviation in the reconstructed η coordinate caused by the linear sense wire structure of the wire segment. As is shown in Figure 5.9, the sense wire (anode wire) in the detector is arranged in straight lines, which do not align with the constant-η curves. Consequently, even for the same wire Position ID, the η value can vary depending on the associated reconstruction result in strip segment.

To correct this discrepancy, a dedicated LUT that encodes the correspondence between the combination of wire and strip Position IDs and the true η coordinate is applied. The LUT takes as input an 8-bit address, composed of the 2 bits from the wire Position ID and 6 bits from the strip Position ID. The output is a corrected 8-bit η value, which is then attached to the reconstructed track.

5.2 Implementation on Athena

This section describes the implementation of the bitwise simulator into Athena framework. The bitwise simulator is a C++-based software tool developed for the Phase-II upgrade that emulates the TGC SL by replicating the bit-level operations performed by

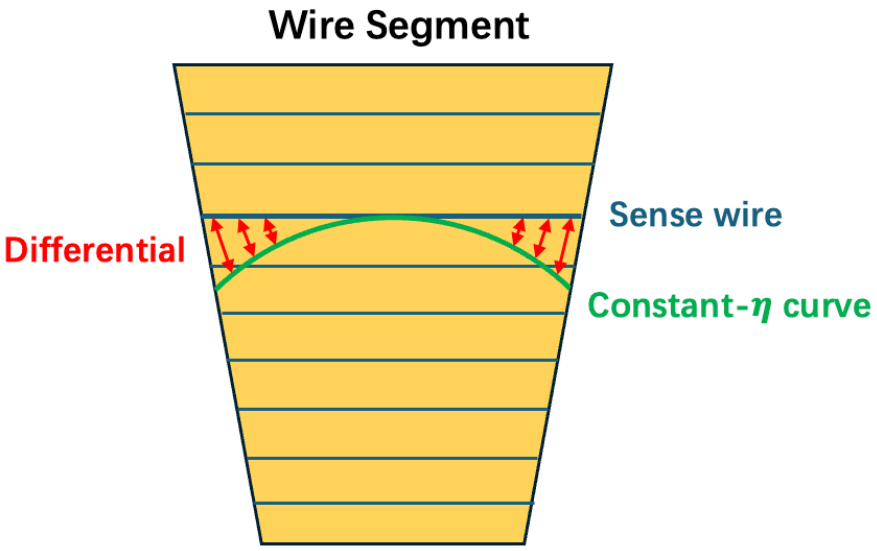


Figure 5.9: Deviation between sense wire and constant-η curve.

the firmware. It receives a hit map of wire segment and strip segment as an input, then reconstructs the muon tracks and calculate the position information (η, ϕ) and the transverse momentum (p_{T}). This implementation of the bitwise simulator has been adapted for multi-threaded execution to adapt the AthenaMT development. The implementation work in this section is divided into two parts: first, the data input interface of the bitwise simulator was modified to conform to Athena's data flow and processing model; and second, the simulator, which originally handled only a single 1/48 TGC BW region, was extended to cover all sectors, including both A-side and C-side. This extension provides a testing environment for the Monte Carlo simulations described in the next chapter.

5.2.1 Input Adaptation for Athena

The original bitwise simulator adopts bitmaps as its input format as it was designed to emulate the firmware behavior. On the other hand, the Athena-based implementation aims to simulate the trigger logic by Monte Carlo simulation samples as input. This section describes the unification of the input formats between the two simulators, allowing consistent application of the same trigger logic on Athena.

Figure 5.10 illustrates the forepart of data preparation and transformation steps for both the bitwise simulator and the Athena-based L0TGCSimulator. In the bitwise simulator,

ROOT tree files[1] are used as input, which stores bitmap as an imitation of the signals in optical links from the PS boards. The *hitpattern operator*, as a dedicated tool, is used to convert the bitmap format into vectors of boolean values representing the presence or absence of hits. These vectors are subsequently passed to the channel mapping module, as described in Section 5.1.1, which converts them into structured vectors that encode hit information in terms of *subsector*, *layer*, and *channel*.

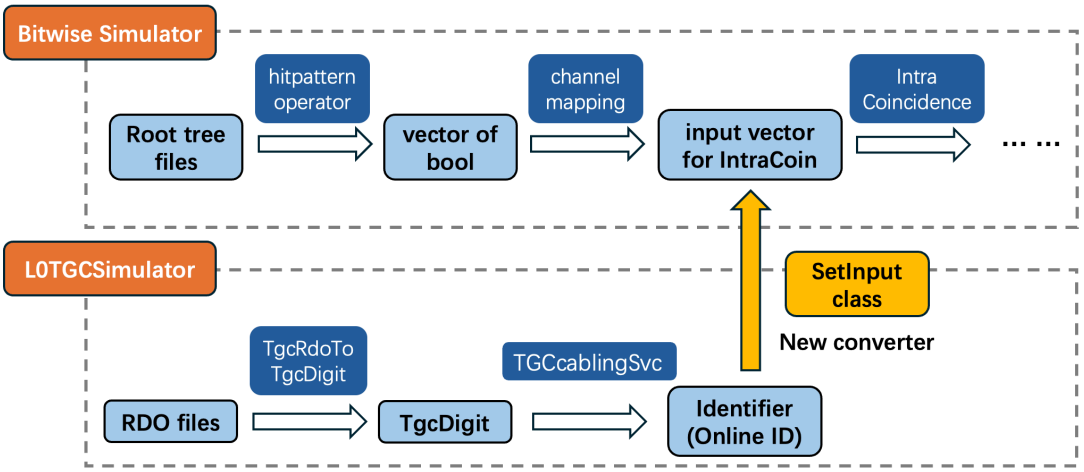


Figure 5.10: Illustration of the input adaptation process from the bitwise simulator to the L0TGCSimulator, enabled by the newly developed SetInput converter.

The L0TGCSimulator receives a RDO (Raw Data Object) file as input, which is a bit-packed format read out from the detector. First, the `TgcRdoToTgcDigitCfg` tool on Athena is used to generate the `TgcDigits`, which represents digitalized hits position information, from the RDO file. Then, a data format converter from `TGCcablingSvc` maps these `TgcDigits` into the *OnlineID*, which describes the specific geometric position on the TGC detector. A summary of the structure and meaning of the OnlineID is provided in Table 5.4.

To interface this with the trigger logic, a class named *SetInput* is developed for this study, as shown in Figure 5.10. This class receives the decoded OnlineIDs and converts them into the boolean input vectors required for the Intra Station Coincidence stage. The mapping rules between the OnlineID values and the corresponding indices of the input vector are described in Table 5.5. Currently, the offset* for wire in the table means the differential of channels caused by the overlaps between adjacent chambers in wire endcap, which

1. a format of data storage, which stores data in a tree-like structure, commonly used in data analysis of high-energy physics.

Table 5.4: List of variables in OnlineID

Variable Name	**Type**	**Value Range**	**Meaning**
subsystemNumber	int	A/C : +1/-1	Indicates Aside or Cside
octantNumber	int	0–7	Index when dividing the wheel into 8 segments(3 sectors for each segments) in the increasing ϕ direction
moduleNumber	int	0–12	Identifier of each segment in increasing ϕ direction when dividing Endcap/Forward wheels into 8 parts (refer to Appendix)
layerNumber	int	0–6: M1–M3; 7,8: EI/FI	Index assigned to each layer in the increasing z direction
rNumber	int	M1: 1–4; M2,M3: 0–4; Forward: 0	Index assigned to each chamber in the increasing η direction
wireOrStrip	bool	T/F: Strip/Wire	Distinction between Strip and Wire
channelNumber	int	0–n	Logical channel number assigned to each channel Wire: increases with η Strip: increases with ϕ (AsideForward, CsideBackward) decreases with ϕ (AsideBackward, CsideForward)

comes from the different definitions of channel numbers between input vector indices and OnlineID; the offset** for strip indicates the channel numbers of the chamber corresponding to the rNumber of OnlineID, because the definitions of channel numbers is the "local id" within a chamber in the OnlineID, and the "global id" across all chambers in the vector.

Table 5.5: Mapping rule between the input vector indices and OnlineID

Indices of Input Vector	**OnlineID**	**Mapping Rule**
side	subsystemNumber	subsystemNumber = 1: side = 0 (A-side) subsystemNumber = -1: side = 1 (C-side)
sl	octantNumber, moduleNumber	sl = 3 × octantNumber + (moduleNumber % 3)
subsec	moduleNumber	subsec = moduleNumber % 3
layer	layerNumber, isStrip	Wire: layer = layerNumber Strip: layer = layerNumber + 6 (no layerNumber = 1 for Strip)
ch	channelNumber	Wire Endcap: ch = channelNumber + offset* Strip Endcap: ch = channelNumber + offset** Wire Forward: ch = channelNumber Strip Forward: ch = channelNumber

5.2.2 Extension to All TGC Sectors

Since the bitwise simulator was originally designed to simulate only 1/48 of the TGC Big Wheel (hereafter referred to as "one TGC sector"), in order to cover the complete endcap trigger logic and simulate muons across the entire endcap region, it is necessary to extend the bitwise simulator to all 48 TGC sectors of the BW (24 sectors on each of the A and C sides).

In the previous section, we described the adaptation of the `TgcDigit` input into the for-

mat used by the bitwise simulator for the Intra-Station Coincidence logic. Since the Intra-Station Coincidence operates independently in each sector and follows the identical coincidence logic, this part is directly extended and implemented by independently processing each TGC sector in parallel. In the following, we describe the extension of the subsequent processes: Wire/Strip Segment Reconstruction and Wire-Strip Coincidence (for convenience, we refer to these modules as *IntraCoin*, *SegRec* (*WireRec* and *StripRec*), and *WSCoin* hereafter, respectively.)

The modules *SegRec* and *WSCoin* rely on Look-Up Tables (LUTs) for the reconstruction process. Before performing the sector-wide extension of these modules, we first analyze the correspondence between LUTs and TGC sectors.

For *SegRec*, one set of LUT are used for *WireRec* and *StripRec*, respectively. In the *WSCoin* process, due to the presence of the Wire Segment Corrector, two set of LUTs are required: one for transverse momentum (p_{T}) calculation, and another for η correction of the wire segment. Taking into account the ϕ-dependent geometry of each sector and the inherent symmetry across sectors, it is determined that a single shared LUT can be used for all 24 sectors on one side (Aside or Cside) for *WireRec*, *StripRec*, and the Wire Segment Corrector. However, the p_{T} LUT (referred to as WSPattern) follows a cyclical pattern across sectors: every three adjacent sectors use distinct LUTs, and this pattern repeats eight times within each side.

The correspondence between LUT types and TGC sectors is summarized in Table 5.6.

Table 5.6: LUT types and their sector-wise application strategy

LUT Type	**LUT-Sectors Mapping Strategy**
WireRec LUT	Shared across all sectors on one side
StripRec LUT	Shared across all sectors on one side
WSPattern LUT	3-sector cyclic pattern repeated 8 times on one side (see Figure 5.11)
WCorrPattern LUT	Shared across all sectors on one side

Following a similar approach to the *IntraCoin* extension, we defined one object per sector for *WireRec*, *StripRec* and *WSCoin*, and each object is assigned the corresponding LUT according to its sector position. The output results confirm that the reconstruction is processing successfully across all 48 TGC sectors. Further evaluations are performed in Section 5.4.

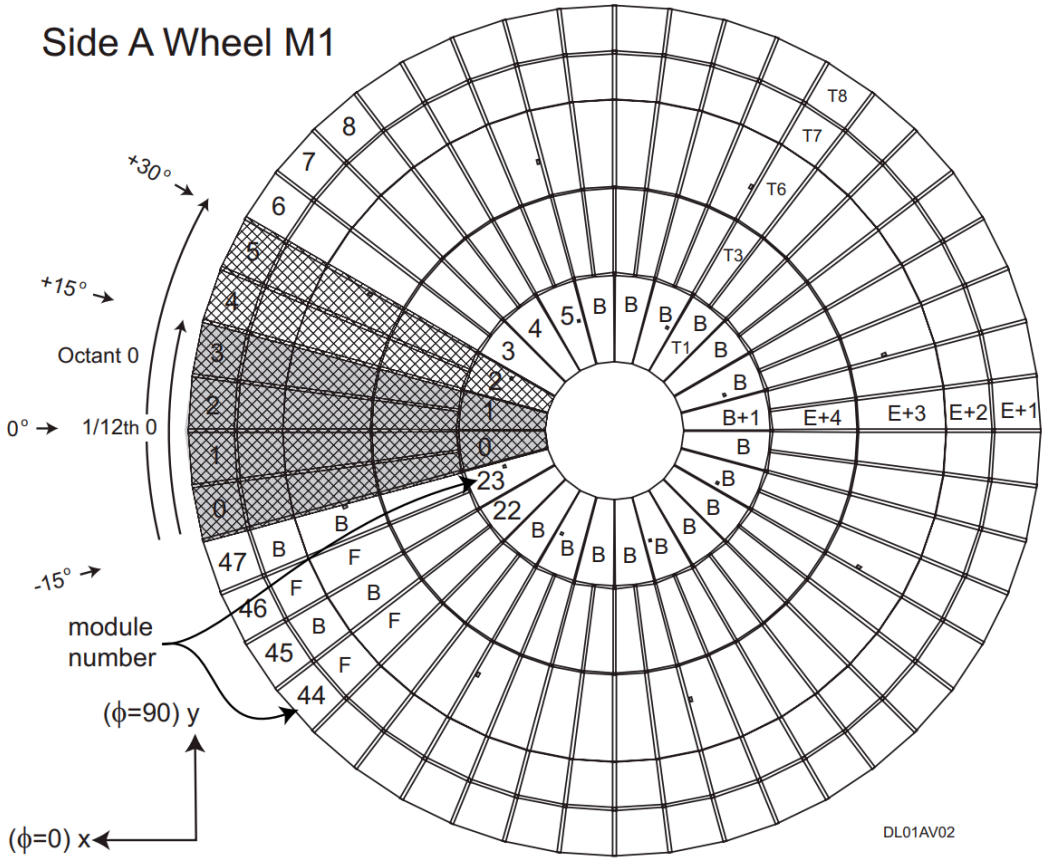


Figure 5.11: Illustration of TGC BW M1 on A-side. The WSPattern LUT are sharing every 3 adjacent sectors, marked in shadows [20].

5.3 Memory Reduction of LUT

In the previous section, the bitwise simulator was successfully integrated into Athena and verified to run locally. However, to support future integration into the real Athena environment, it is essential to reduce the simulator's memory usage to meet the framework's memory constraints, which is around 500 MB for the simulation of TGC detector. This section presents an attempt of memory reduction optimization by a LUT storage simplification.

A previous simple calculation in Chapter 1 shows that, if directly scaled the bitwise simulator to cover all 48 TGC sectors, the total memory usage would reach about 14 GB, which significantly exceeds the permissible limit. Through the optimisation strategies described in this section, the extended L0TGCSimulator reduces total memory usage to approximately 6.3 GB.

After an investigation of the memory structure of the bitwise simulator, it is revealed that the LUTs consume about 50% of the whole memory usage of bitwise simulator, while the remaining memory is occupied by vectors storing hit information for processing, as well as by the database, cache, and other. A brief summary of the memory usage of the original bitwise simulator is given by Figure 5.12.

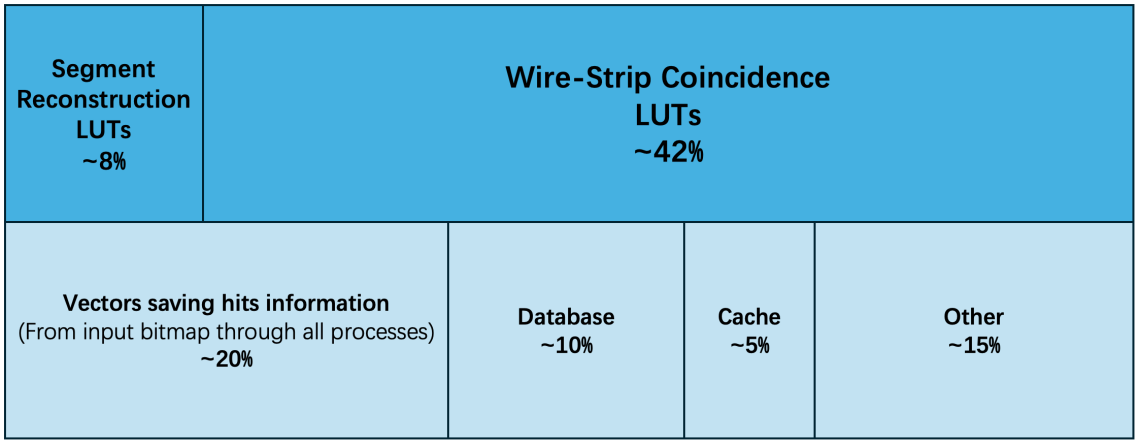


Figure 5.12: Summary of the memory usage of the bitwise simulator before implementation

The reason of the huge memory of LUT is that, the LUTs contain a great number of possible hit patterns for searching, especially in *WSCoin* process, where more than 900k patterns of $(\Delta\theta, \Delta\phi, p_{\mathrm{T}})$ are pre-defined. Furthermore, unlike the firmware where URAM is used for saving LUTs, the LUTs in bitwise simulator are decoded from text files and then stored into high-dimensional vectors of *map* variables. Each vector dimension corresponds

to the pivot hit position information, which are used to determine which LUT to be referred to. This position information indicates the corresponding *subunit* position of hit on M3 station in *SegRec*, and the η value of the M3 hit in *WSCoin*. This storage structure is applied for both the *StripRec* and *WSCoin* processes.

In the original bitwise simulator, any index as dimension only uses lower than 6 bits in value, however stored within a 32 bit integer, resulting in vast amounts of unused address space. This inefficiency is visualised on the left side of Fig. 5.13, which abstracts the LUT container into a three-dimensional cube.

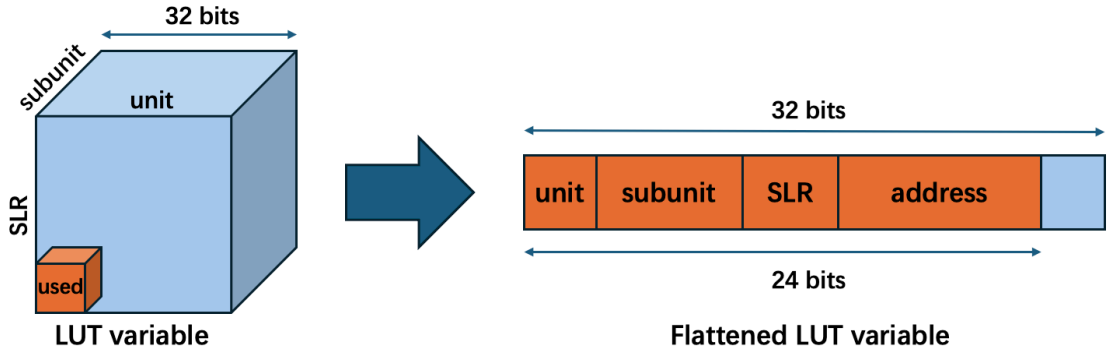


Figure 5.13: Illustration of the LUT optimization strategy. The left part shows the high-dimensional vectors storing way, while the right one demonstrates the storing after flattening technique.

To address this, a new scheme was applied that flattens the multi-dimensional container into a one-dimensional map by packing the individual coordinate indices into a single integer key as `uint`. Each field uses only as many bits as required by its range, and the final key is constructed through bit-shifting. The result is a more compact and efficient memory structure, as shown in the right part of Fig. 5.13. An illustrative overview of the new LUT structure is shown in Table 5.7. All four LUT types, *WireRec*, *StripRec*, *WSPattern*, and *WCorr*, were migrated to this representation.

To further optimize both memory and access time, the standard `std::map` container was replaced by `std::unordered_map`. A `std::map` in C++ is an associative container that stores key-value pairs in a sorted order according to the key. Internally, it uses a self-balancing binary search tree (BST), like a Red-Black tree. This guarantees logarithmic time complexity $\mathcal{O}(\log n)$ for insertion, deletion, and search operations. On the other hand, `std::unordered_map` is implemented using a *hash table*. This allows average constant-time complexity $\mathcal{O}(1)$ for insertion, deletion, and lookup, although the element order is not maintained. Because it avoids tree-based structures, it also consumes less memory on

Table 5.7: Bit-level structure of key and value for each LUT in the simulator

m_map_SegRecWire_LUTs	SLR	unit	subunit	i	key	value
bits	2	6	2	2	12	18
sum	24-bit key, 18-bit value					

m_map_SegRecStrip_LUTs	SLR	ichamber	iuram	isUramB	key	value
bits	2	3	2	1	16	9
sum	24-bit key, 9-bit value					

m_WSPattern_map	eta_ID	address	value
bits	16	16	4
sum	32-bit key, 4-bit value		

m_WCorrPattern_map	eta_ID	address	value
bits	16	8	1
sum	24-bit key, 1-bit value		

average. In the present case, since the keys used to represent hit patterns stored in the LUT are typically sparse and discontinuous, using an unordered_map is suited to this purpose and does not cause problems. The main differences between map and unordered_map are listed in Table 5.8.

Table 5.8: Comparison between `std::map` and `std::unordered_map` in C++

Feature	`std::map`	`std::unordered_map`
Data Structure	Self-balancing BST	Hash Table
Order	Sorted order	No specific order
Time Complexity	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$ on average
Memory Usage	Higher (due to tree nodes)	Lower (uses hashing)
Use Case	When sorted order is required	When order is unimportant but speed is crucial

The final data structures used are as follows: the sizes of "uint" variables are adjusted by the numbers of necessary bits of address and position information.

- `std::unordered_map<uint32_t, uint32_t>` for *WireRec* LUTs;
- `std::unordered_map<uint32_t, uint32_t>` for *StripRec* LUTs;
- `std::unordered_map<uint32_t, uint8_t>` for *WSPattern* LUTs;

- `std::unordered_map<uint32_t, uint8_t>` for *WCorr* LUTs.

As a result of this optimization, combined with the simplification of input data chain discussed in Section 5.2.1 and the LUT sharing scheme, the total memory usage of implemented L0TGCSimulator was reduced from the originally estimated 14 GB down to approximately 6.3 GB, corresponding to a 55% decrease.

5.4 Performance of the L0TGCSimulator

In this section, we will investigate the behavior of the L0TGCSimulator by means of the efficiency of the segment reconstruction. The efficiency is then compared to that from the stand-alone bitwise simulator performed by previous studies as reference. The simulation for the L0TGCSimulator uses Monte Carlo sample of single muon events at a center-of-mass energy of 13.6 TeV, from $2 < p_{\mathrm{T}} < 50$ GeV, with a total of 5,000 events.

Figure 5.14 and Figure 5.15 give the efficiencies on η in wire segment of stand-alone bitwise simulator [19] and the L0TGCSimulator, respectively. Figure 5.16 and Figure 5.17 show the efficiencies on η in strip segment of stand-alone bitwise simulator [19] and the L0TGCSimulator, respectively. The efficiencies are defined in Equation 5.1.

$$\text{Efficiency} = \frac{\text{Muons reconstructed successfully by the wire/strip segment}}{\text{All the truth muons in endcaps}} \tag{5.1}$$

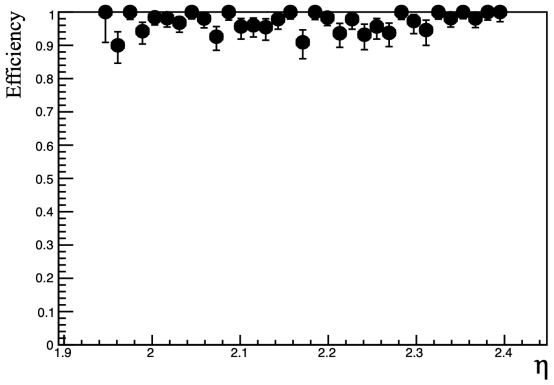


Figure 5.14: Efficiency on η in forward region of wire segment of the stand-alone bitwise simulator. The horizontal axis is η [19].

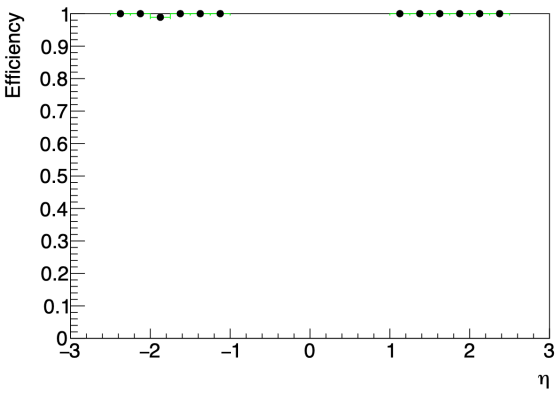


Figure 5.15: Efficiency on η in wire segment of the L0TGCSimulator. With muons of $1.05 < |\eta| < 2.41$

Figure 5.18 and Figure 5.19 gives the efficiencies on p_{T} in wire and strip segment of the L0TGCSimulator, respectively.

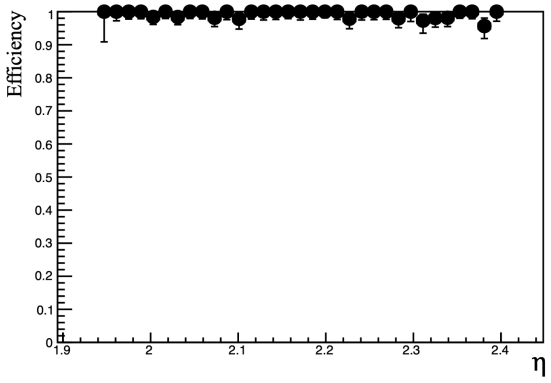


Figure 5.16: Efficiency on η in forward region of strip segment of the stand-alone bitwise simulator. The horizontal axis is η [19].

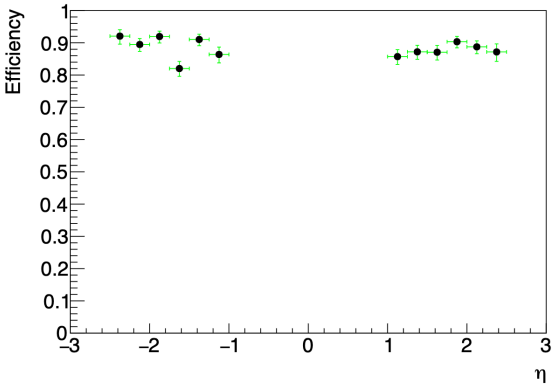


Figure 5.17: Efficiency on η in strip segment of the L0TGCSimulator. With muons of $1.05 < |\eta| < 2.41$

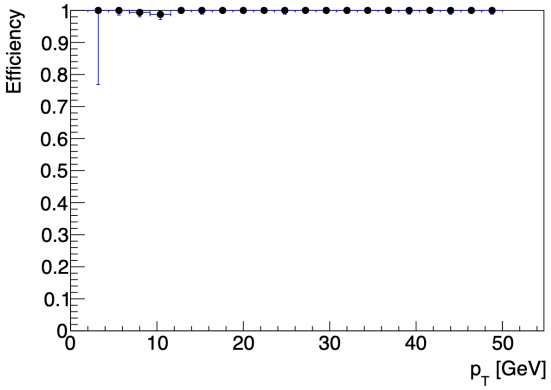


Figure 5.18: Efficiency on p_{T} in wire segment of the L0TGCSimulator.

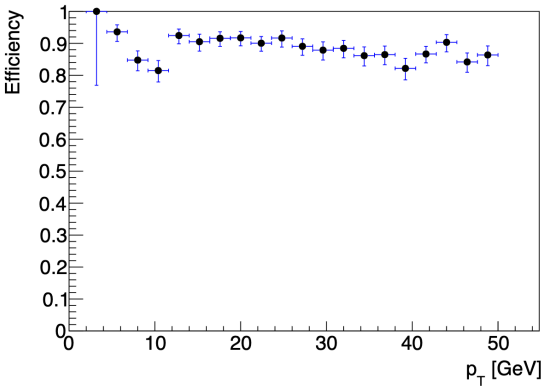


Figure 5.19: Efficiency on p_{T} in strip segment of the L0TGCSimulator.

As a result, the L0TGCSimulator reproduced the efficiency fairly in wire segment, with efficiencies higher than 95% over the $1.05 < |\eta| < 2.41$ and p_{T} over the given range of $2 < p_{\mathrm{T}} < 50$ GeV. In the strip segment, however, the efficiency is not as good as the reference, with efficiencies at about 85% on average. After examining the intermediate outputs, this 15% inefficiency in strip segment may due to a mismatch in the mapping scheme between the OnlineID and the input vector indices, as introduced in Section 5.2.1. The specific reason is under investigation.

6

Conclusion and Outlook

The ongoing upgrade of the ATLAS detector for the High-Luminosity LHC operation will involve a comprehensive upgrade of the electronics for the endcap muon trigger system, including the TGC Sector Logic, which is responsible for reconstructing muon tracks for further selection. In this upgrade, software simulation is required both for firmware development and for assessing trigger inefficiencies to be corrected in physics data analyses. In order to simulate the TGC Sector Logic within the whole muon trigger chain, a corresponding software simulator has to be implemented in ATLAS software framework, Athena.

This study developed two kinds of simulation software for the Athena implementation. The first simulation software is a simple emulator, the L0MuonEmulator, to support downstream development for the simulation of the trigger chain. The emulator smears the p_{T} of muons and assumes a flat efficiency for muons above a p_{T} threshold across angular coverage of muon detector. Some "holes" in the detector acceptance, caused by mechanical structures, are excluded in the coverage, where the efficiency is set as zero. The efficiency of the L0MuonEmulator shows similar behavior to the efficiency of the endcap muon trigger evaluated by the LHC Run 3 data. The emulator also gives similar plateau efficiency to the efficiency for the HL-LHC configuration of the barrel muon trigger es-

timated earlier. This L0MuonEmulator provides a useful framework to study the muon trigger spatial dependence and efficiency.

The second software simulation is a simulator running on the Athena environment, the L0TGCSimulator, which reproduces the firmware behavior of the hardware L0 trigger for the endcap muon system. The firmware simulation is based on an existing simulator, the bitwise simulator, emulating the bit-level operation of the TGC Sector Logic firmware. In order to reduce the memory usage, which is constrained by the offline computing environment, an optimization was performed through the simplification of the LUT storage structure, which reduced the memory usage from the expected 14 GB to approximately 6.3 GB, corresponding to a reduction of about 55%. The efficiency of the implemented L0TGSimulator showed that the efficiency in wire segment was consistent with that of the stand-alone bitwise simulator. The strip segment, however, has an efficiency about 15% lower than that in the reference. Investigation suggests that this discrepancy originates from mismatches in input adaptation between the bitwise simulator and Athena. The exact reason is under investigation.

As an outlook of this study, the inefficiency in the strip segment should be solved. For further memory reduction of the L0TGCSimulator, reduction of the number of LUT patterns is also considered. By removing some impossible patterns, or “merging” several close patterns as a single pattern (lossy compression), the memory of LUTs could be reduced further.

Bibliography

[1] E. Lopienska, The CERN accelerator complex layout in 2022, 2022. https://cds.cern.ch/record/2800984. (cited on page 9)

[2] ATLAS collaboration, The ATLAS Experiment at the CERN Large Hadron Collider, *JINST* **3** (2008) S08003. (cited on pages 11, 17, 18, 20, 21, 22, and 24)

[3] CERN, High-Luminosity LHC Project, 2025. https://hilumilhc.web.cern.ch/content/hl-lhc-project. (cited on page 11)

[4] ATLAS collaboration, The ATLAS experiment at the CERN Large Hadron Collider: a description of the detector configuration for Run 3, *JINST* **19** (2024) P05063. (cited on pages 19 and 26)

[5] ATLAS Collaboration, Technical Design Report for the Phase-II Upgrade of the ATLAS Trigger and Data Acquisition System, Tech. Rep. CERN-LHCC-2017-020, CERN (2018). (cited on pages 22, 32, 44, and 47)

[6] C. Marcelloni, Installation of the first of the big wheels of the ATLAS muon spectrometer, a thin gap chamber (TGC) wheel, 2006. https://cds.cern.ch/record/986163. (cited on page 24)

[7] ATLAS Collaboration, Technical Design Report - New Small Wheel, Tech. Rep. CERN-LHCC-2013-006, CERN (2013). (cited on page 25)

[8] Antonio Policicchio, ATLAS Muon Trigger performance, 2019. https://cds.cern.ch/record/2692868/files/ATL-DAQ-SLIDE-2019-740.pdf. (cited on page 27)

[9] ATLAS collaboration, The ATLAS Trigger System for LHC Run 3 and Trigger performance in 2022, *JINST* **19** (2024) P06029. (cited on pages 28, 29, and 47)

[10] ATLAS Collaboration, Technical Design Report for the Phase-II Upgrade of the ATLAS Trigger and Data Acquisition System - EF Tracking Amendment, Tech. Rep. CERN-LHCC-2022-004, CERN (2022). (cited on pages 29 and 31)

[11] ATLAS Collaboration, Software and computing for run 3 of the atlas experiment at the lhc, *Eur. Phys. J. C* **85** (2025) 234. (cited on pages 33, 38, and 39)

[12] G. Barrand, P. Binko, M. Frank, C. Gaspar, F. Govoni, M.D. Joos *et al.*, Gaudi —a software architecture and framework for building hep data processing applications, *Computer Physics Communications* **140** (2001) 45. (cited on pages 34 and 35)

[13] CERN. Geneva. LHC Experiments Committee, A Large Hadron Collider Beauty Experiment for Precision Measurements of CP Violation and Rare Decays, Tech. Rep. CERN-LHCC-98-004, CERN (1998). (cited on page 36)

[14] CERN, Computing in High Energy and Nuclear Physics 2004 (CHEP'04), Tech. Rep. CERN-2005-002, CERN (April, 2005). (cited on page 36)

[15] P. Calafiura, C. Leggett, D. Quarrie and S.R. H. Ma, The StoreGate: a Data Model for the Atlas Software Architecture, . (cited on page 37)

[16] S. Binet, P. Calafiura, M.K. Jha, W. Lavrijsen, C. Leggett, D. Lesny *et al.*, Multicore in production: advantages and limits of the multiprocess approach in the atlas experiment, . (cited on page 38)

[17] ATLAS Collaboration, Performance of Multi-threaded Reconstruction in ATLAS, Tech. Rep. ATL-SOFT-PUB-2021-002, CERN (2021). (cited on page 39)

[18] ATLAS Collaboration, Endcap Sector Logic: PBS/WBS 1.1.5.1 (Preliminary Design Report), Tech. Rep. CERN (2021). (cited on pages 53, 54, 55, 58, 59, and 60)

[19] E. Yamashita, Development and verification of level-0 muon trigger for high-luminosity LHC-ATLAS Experiment, Master's thesis, University of Tokyo, 2022. (cited on pages 52, 62, 73, and 74)

[20] D. Lellouch, L. Levinson, K. Hasuko and A.L.-.M.T. Group, TGC naming and numbering scheme for the Endcap muon trigger system, Tech. Rep. ATL-MUON-2001-002, ATLAS Collaboration (February, 2005). (cited on page 69)

Acknowledgement

I would like to express my sincere gratitude to Prof. Junpei Maeda and Prof. Yuji Yamazaki, who have continuously guided and supported my research throughout my master' s program. Their patience, encouragement, and insightful advice were crucial to the completion of this thesis. Prof. Junpei Maeda provided continuous technical guidance and encouragement during the most challenging phases of my research. Prof. Yuji Yamazaki, as my supervisor, guided the overall direction of my study and provided detailed feedback on this thesis. Without their support, the completion of this thesis would have been impossible.

I would like to thank Prof. Hisaya Kurashige from Kobe ATLAS group for his guidance during weekly meetings and for his kind assistance in discussions related to physics. I would also like to express my gratitude to other teachers in our research room: Prof. Yasuo Takeuchi, Prof. Kentaro Miuchi, Prof. Atsumu Suzuki, Prof. Hiroshi Ito, Dr. Satoshi Higashino, for their assistance during my presentations and colloquiums.

I am grateful to Prof. Susumu Okubo and Prof. Junpei Maeda for taking the time to review my thesis as secondary examiners, despite their busy schedules.

I would like to extend my appreciation to the former members of Kobe ATLAS group during my master's studies: Koya Toji, Yui Murata, Ryugo Mizuhiki, Shota Nishi, Rukumo Higuchi, as well as the current members: Ryosuke Tanaka, Rintaro Yamaguchi, Yuma Sano, Yuki Asami, Keita Sakura, and Masahiro Sasada, for their company and help

throughout my master's research journey. Also, I would like to thank members of the other groups in the research room, whose presence added cheerfulness to my research life.

I would like to acknowledge the staff members of the ATLAS Japan group: Yasuyuki Okumura, Yasuyuki Horii, Shota Izumiyama, Yuki Sue, and Masato Aoki, for their valuable help and advice. I would also like to thank students of the ATLAS Japan group in ICEPP: Erika Yamashita, Airu Makita, Towa Mizuochi, and Kazuma Maki, who continuously shared their research progress.

Finally, I would like to express my deepest gratitude to my parents, who have always supported and encouraged me unconditionally. They provided me with the opportunity to study and communicate overseas. I offer them my highest respect and heartfelt thanks.

Li Zhang
Kobe, the 8th of August 2025