BRUNEL UNIVERSITY

DOCTORAL THESIS

Complexity-reduced Hardware-based Track-Trigger for CMS Upgrade

---

Author:
Maziar GHORBANI

Supervisor:
Dr Joanne COLE

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

CMS Detector Group
Electronic and Electrical Engineering

September 2, 2022

This page was intentionally left blank.

BRUNEL UNIVERSITY LONDON

Abstract

College of Engineering, Design and Physical Sciences

Electronic and Electrical Engineering

Complexity-reduced Hardware-based Track-Trigger

Maziar GHORBANI

The Compact Muon Solenoid (CMS) detector at the Large Hadron Collider (LHC) is designed to study the results of proton-proton collisions. The Tracker sub-detector is designed to detect and reconstruct the trajectories of charged particles produced by the collisions. During the lifetime of the CMS detector, there have been several upgrades aimed at increasing the chance of discovering new physics through increased luminosity levels and instrumentation of advanced technology. The High-Luminosity upgrade optimises the LHC to accelerate high-energy particles with an average of 200 proton-proton interactions per bunch crossing. The Level-1 Trigger system promptly analyses and filters collisions using hardware to reduce the data volume in real-time. For the upgrade, the trigger mechanism will use a particle trajectory estimator that discriminates between particles based on their transverse momentum (p_T). Particles with $p_T \geq 2$ GeV/c will be transmitted to the Level-1 Track-Trigger system for trajectory reconstruction within a fixed 3 μs latency. This thesis presents a novel Hardware-based Multivariate Linear Fitter (MVLF) system focusing on robustness in tracking efficiency and reduction in logic resource usage within the specified latency. The system components are implemented in Field Programmable Gate Arrays (FPGA), targeting 16 nm FinFET UltraScale+ silicon technology. The development was performed using the High-Level Synthesis (HLS) automation tools and the Hardware acceleration platform for Application-Specific Integrated Circuits (ASIC). A firmware demonstrator has been assembled to verify the feasibility and compatibility of the scaled system with the CMS Level-1 Track-Trigger infrastructure. The system's performance is compared to past and current system developments, and the results are presented accordingly.

This page was intentionally left blank.

Colophon

This document was typeset using the TeXLive distribution of LaTeX and the XeLaTeX compiler. References were managed by BibLaTeX with back-end compatibility to biber and IEEE style. The font is set to TeX Gyre Bonum Font Family 12 pt, a fork of the Linux tgbonum typeface by the GUST e-foundry. The same typeset is used for Math Equations, Tables and Captions throughout the document. The hyphenation is disabled to assist Text to Speech Voice Readers such as Speechyfi and Read Aloud algorithm. No Hyphenation has caused spaces between words and sentences as unwanted side effects. The Vectorian ornaments are imitation of pst-vectorian pgfornament CTAN package. For line spacing, singlespacing and onehalfspacing are used interchangeably. The diagrams are edited in diagrams.net online diagram draw.io software and pictures in cross-platform GIMP2 image editor available for GNU/Linux, OS X, Windows.

This page was intentionally left blank.

"*From the point of view of the physicist, a theory of matter is a policy rather than a creed; its object is to connect or co-ordinate apparently diverse phenomena, and above all to suggest, stimulate and direct experiment. It ought to furnish a compass which, if followed, will lead the observer further and further into previously unexplored regions.*"

Sir Joseph John Thomson

This page was intentionally left blank.

Declaration of Authorship

I declare that this thesis titled, "Complexity-reduced Hardware-based Track-Trigger for the CMS Upgrade" and the work presented in it are my own. I confirm that:

- The work in this thesis was conducted wholly or mainly during candidature for a research degree at Brunel University.
- No part of this thesis has been previously submitted to this or any other university as part of the requirement for a higher degree.
- When the published work of others has been consulted, it has been attributed within the text.
- My primary contribution in this thesis is implementing the hardware-based Multivariate Linear Fitter system for the high-luminosity upgrade of the CMS detector and the demonstrator system for the performance analysis and evaluation described in Chapters 6, 7, 8 and 9.
- I contributed to the algorithm optimisation and performance analysis of the Time-Multiplexed Track-Trigger for the development of the Level-1 Track-Trigger system.
- I contributed to the performance analysis and the optimisation of the Hybrid tracking System development for FPGA devices and the high-level synthesis optimisations and hardware performance verification in the demonstrator.
- The results presented in this thesis use simulation samples produced by the CMS experiment but are not officially endorsed by the CMS collaboration in any capacity. An approval procedure seeking this endorsement is under consideration.
- In addition, I contributed to the algorithm designed for the high-voltage biasing of the silicon modules project for the Triggering and Data Acquisition system. I also contributed to the commissioning and calibrating of the CMS silicon-strip Tracker project required for the analogue front-end ASIC and the off-detector board digitisation.

This page was intentionally left blank.

Acknowledgment

I want to take this opportunity to thank those who supported me at every stage of my postgraduate life and research project:

First and foremost, I am incredibly grateful to my supervisor **Dr Joanne Cole** for her advice, support, guidance and patience, especially when I needed it the most.

My most tremendous appreciation goes to **Professor Akram Khan** for his support and oversight throughout my research project.

I express gratitude to **Professor Peter Hobson** who believed in me and allowed me to become a part of his fantastic team.

I express my gratitude to **Dr Hongying Meng** for his guidance and deep insight into my research project.

My sincere gratitude goes to **Dr Ivan Reid** for supporting me throughout my research, even when my ideas were out of reach and felt unreachable.

I want to thank **Dr Ian Tomalin** for unwavering support and teachings, which encouraged me to explore new ideas and learn more through experimenting.

I am incredibly grateful to **Dr Thomas Schuh** for his help, knowledge, tremendous understanding and patience throughout my research.

I extend my deepest gratitude to **Dr Nicholas Martin Craven Brackenbury** for the professional advice, words of encouragement and thoughtful guidance.

I gratefully recognise and express my gratitude to everyone at **Brunel University London**, **UK Liaison Office**, **Science and Technology Facilities Council**, **Rutherford Appleton Laboratory**, **Fermilab**, and **European Council for Nuclear Research** who directly and indirectly has been involved with my project.

There is always a special **THANK YOU** to **my family** and **friends** for believing in me because without their support and help, this great endeavour would not have been possible.

This page was intentionally left blank.

Contents

This page was intentionally left blank.

List of Figures

This page was intentionally left blank.

List of Tables

This page was intentionally left blank.

Chapter 1

Introduction

1.1 Theory and Motivations

Modern research in experimental particle physics continues to reduce the gap between theory and practice [1]. With current advancements in technology, new detectors are developed and made available to help researchers in their quest for exciting discoveries [2]. At Large Hadron Collider (LHC), located at the European Council for Nuclear Research (CERN) [3], technological innovations are explored by scientists in R&D projects. The LHC particle accelerator is an international High-Energy Physics (HEP) organisation with more than 7000 physicists, engineers, and technicians, taking part in an endeavour to understand the world on the smallest scales. The LHC installation comprises circular beam pipes with superconducting magnets and the accelerating mechanism to energise and steer particles in controlled orbits in opposite directions. Strong magnets force the particles to collide at intersections alongside the beam pipe at a velocity close to the speed of light. In proton-proton collisions, building elements of two protons: quarks and gluons, interact. In their interaction, a wide array of low- to high-energy particles are produced and measured by the super-sensitive sensors [4]. In recent years, the LHC has played an essential role in identifying new physics concerning subatomic particles. During the lifetime of the LHC, the accelerator has undergone several upgrades. The optimisations enhance the potential of discoveries by increasing the number of collisions to produce more data [5]. The generated data contain particle's energy and properties at the time of collision and moments after the collision. Scientists can validate hypotheses concerning the matter's origin by measuring these parameters.

More hypotheses are waiting to be tested and confirmed in particle physics. Experimental physicists predict further discoveries are possible under a highly intense environment where more particles interact [6]. The idea inspires scientists to conceptualise and implement compatible instrumentation to increase the chance of finding exciting physics. Shining light over rare physics phenomena and unveiling the universe's secrets continues to present new experimental challenges for researchers. The LHC accelerator and its detectors undergo several significant optimisations to overcome the present limitations and bridge the gap between their conceptual design capabilities and the real-life performance. One of these optimisations occurs during the High-Luminosity upgrade, enabling LHC to increase the number of collisions by order of magnitude greater than the LHC today. With the increase in data volume, a tracking mechanism is required to identify interesting physics against the background noise in the CMS detector [7]. This thesis presents a novel hardware-based trajectory estimator that operates during the data-taking phase of the detector. The custom-designed integrated circuit will act as a track finder to detect and select particles with transverse momentum greater than 2 GeV/c and reject particles with transverse momentum less than 2 GeV/c.

1.2 Large Hadron Collider

The Large Hadron Collider (LHC) was constructed to replace the Large Electron-Positron (LEP) storage ring. The LHC forces the sub-atomic particles to travel in clockwise and anti-clockwise directions as beams (27 km in circumference) [8]. The technologies deployed for the acceleration are ultra-high vacuum tubes, superconducting electromagnetic coils, liquid helium cooler systems, null resistance electrical cables, and various radioactive hardened electrical components. The accelerator carries out its duties in stages. First, protons are bunched and squeezed down in size to increase the possibility of collisions. Then they are injected into the Linear accelerator (Linac 2) in preparation for injection into the accelerator ring. The Synchrotron Booster (SB) and Super Proton Synchrotron (SPS) increase the kinetic energy of protons from 9 MeV to 1.4 GeV in sequence [9]. The LHC main ring consists of two smaller parallel vacuum pipes that the beams

circulate within. Electric fields spaced around the accelerator create radio waves that accelerate particles in bunches, while electromagnets steer and focus the particles in opposite directions to the kinetic energy of 7 TeV per beam. At four interaction points (ATLAS, LHCb, CMS, ALICE) located on the beamline, particles collide in a process known as Bunch Crossing. The LHC detectors measure the effects of collisions (events) in 40 million intervals per second and store the data for further analysis. In Figure 1.1, a simplified illustration of the LHC acceleration beam-pipes and underground installations is shown.

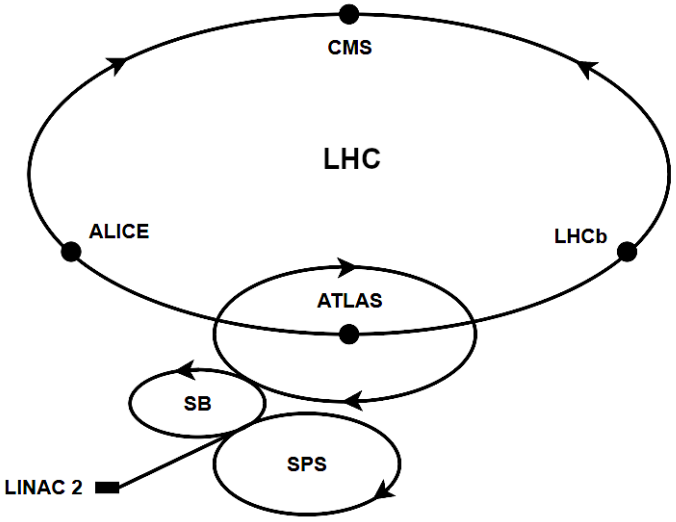

Figure 1.1: CERN's Accelerator Complex [10].

1.3 Compact Muon Solenoid

The Compact Muon Solenoid (CMS) particle detector is one of the LHC detectors. Its purpose is to observe wide ranges of particles produced by high-energy collisions. The detector was constructed in fifteen sections at the ground level before being lowered into an underground cavern near Cessy in France. The individual detectors (sub-detectors) are built surrounding a large magnet in the shape of a cylinder capable of generating a magnetic field of 4 Tesla. Figures 1.2 and 1.3 illustrate the CMS detector from different views; the inner sub-detectors and their structural hierarchy.

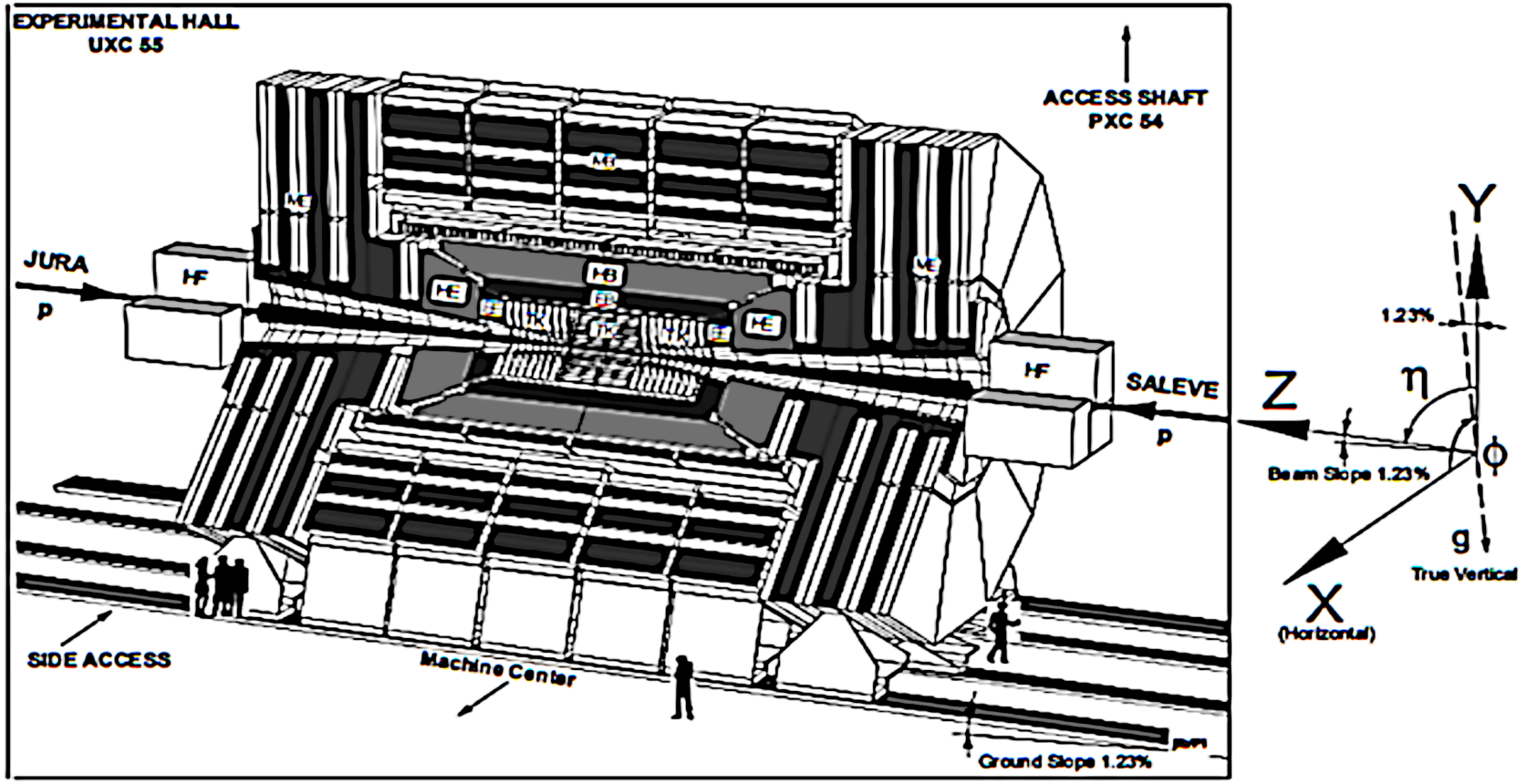

Figure 1.2: Compact Muon Solenoid Particle Detector [11].

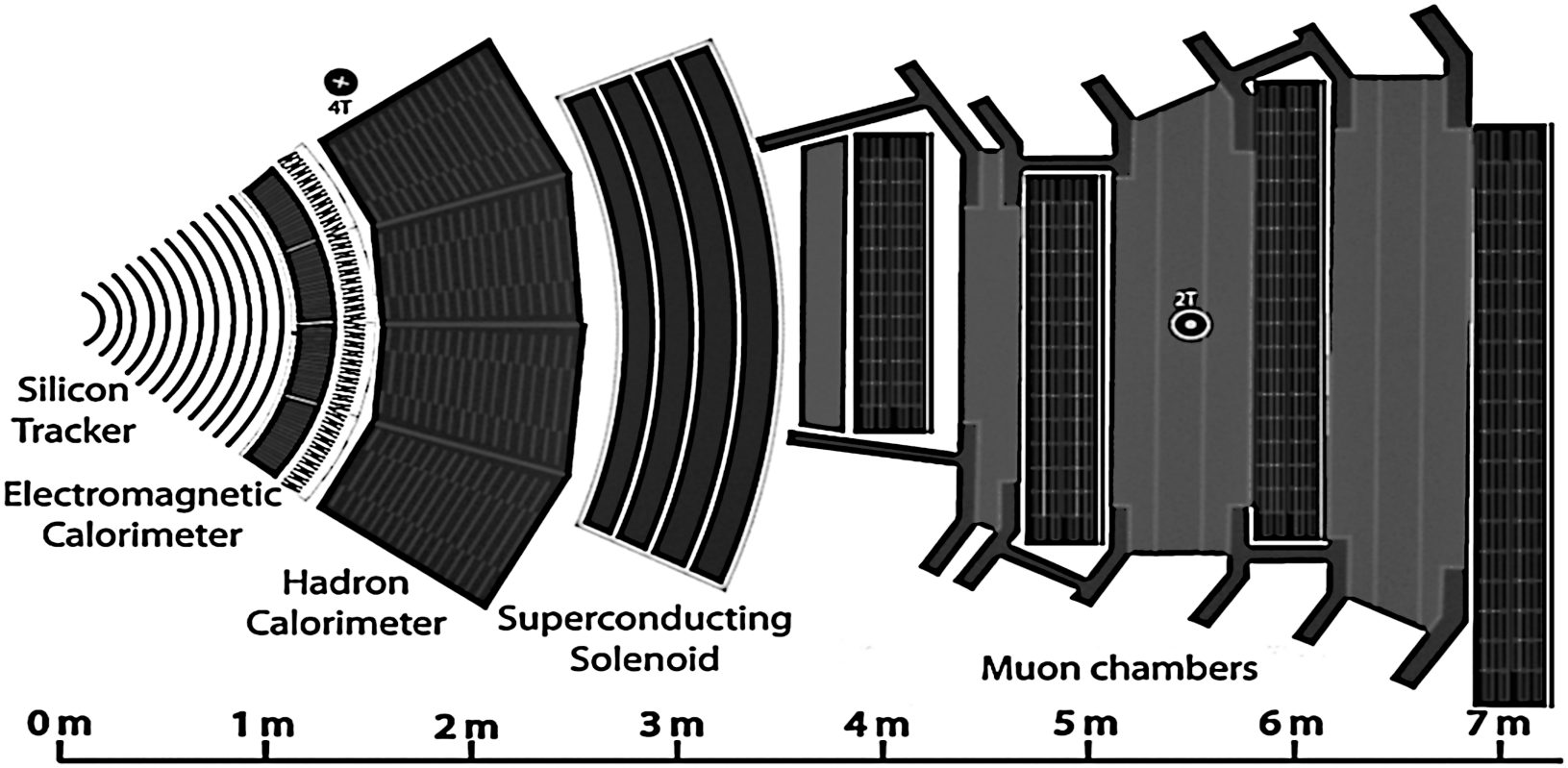

Figure 1.3: Compact Muon Solenoid Detector Regions [12].

The interactions occur at the centre of the CMS detector. The strong magnet bends the charged particle's trajectory back toward the detector's centre (interaction point). The centripetal force sets particles in a circular motion perpendicular to the velocity of the particle along the LHC beampipe. Various measurement sensors and readout electronics rotationally symmetrical to the beampipe capture the particle's trajectories. The detector measures the energy and momentum of photons, electrons, muons, and other products of the collisions. The silicon-based tracker occupies the innermost regions near the interaction point, surrounded by a scintillating crystal electromagnetic calorimeter surrounded by the hadronic detector. The tracker and the calorimetry are compact and fit inside the CMS magnet. The large muon detector consists of small and large sections encircling all other detectors. The functionality of all detectors is to record a particle's path through interpolating and extrapolating its primary vertices.

1.3.1 Silicon Tracker

The Silicon Tracker is the closest to the LHC beam and hence experiences the most significant particle density and multiplicities. The area is also the smallest in radius, and it is made up of silicon devices and readout electronics. The tracker is divided into two inner and outer trackers, each having a barrel and two disks at both ends known as endcaps. Being closest to the particle interaction point means more radiation from collisions and a higher signal-to-noise ratio. Hence, tracking devices in these regions are hardened to radioactivity and have a relatively short life expectancy. In the current Tracker design, the inner tracker deploys pixel sensors near the interaction point surrounded by strip sensors in the outer tracker. A detailed description of the CMS Tracker can be found elsewhere [13].

1.3.2 Electromagnetic Calorimeter

The particles generated in a bunch-crossing have to pass through the Tracker regions to reach the Electromagnetic Calorimeter (ECAL). The hermetic, fine-grained, homogeneous calorimeter comprises photo-detectors, preshower and readout electronics. The ECAL detector comprises a barrel with two disk-like endcaps at both ends. In the Electromagnetic Preshower (ES), lead and silicon

sensors are installed before both endcaps for background noise rejection and better detection resolution for particles with an energy range of 25 GeV to 150 GeV. A total of 75,848 crystal modules, 4,288 sensors, and 137,216 silicon strips are installed within 36 super-modules in between the Barrel (EB) and Endcaps (EE) regions. A detailed description of the ECAL detector can be found elsewhere [14].

1.3.3 Hadron Calorimeter

Surrounding the Electromagnetic Calorimeter is the Hadron Calorimeter (HCAL), where hadrons are detected and measured. The layers of HCAL are formulated in a staggered design to increase particle detection. The HCAL sections consist of closely-spaced absorbers and rectangular plastic scintillators in Hadron Barrel (HB) and Hadron Endcap (HE) calorimeters. The inner section of the HCAL detector is inside the CMS magnet, and the outer area is integrated into the first layer of the Muon chamber outside the superconducting magnet. Approximately 4,300 towers with 17 layers of scintillator tiles are grouped into two light-sampling depths to measure hadronic showers efficiently. The summation of samples on a layer-to-layer basis defines the particle energy in HCAL. A detailed description of the HCAL sub-detector can be found elsewhere [15].

1.3.4 Superconducting Magnets

The magnets are well studied in different fields of particle physics, theoretically and experimentally. The CMS cylindrical coil is large enough to surround the silicon tracker and ECAL with a 12.5m length and 5.9m inner diameter. The magnet's yoke comprises five wheels and two endcaps containing three disks each. The primary characteristic of the solenoid magnet in the CMS detector is creating a uniform magnetic field up to 4 Tesla. The strong magnetic field bends particles back towards the centre of the detector, preventing low-energy particles from reaching the outer sub-detectors. A detailed description of the CMS Superconducting Magnet can be found elsewhere [16].

1.3.5 Muon Chambers

The outermost sub-detector of the CMS is the Muon Detector aligned with the central tracker. The particles that escape the Tracker, ECAL and HCAL layers without interaction with absorbers have an unmistakable signature that identifies them as Muons. The sub-detector comprises a barrel and two endcaps. The muon detector captures particles within its four layers of muon chambers. The particle density and multiplicity in these regions are relatively low compared to the Tracker, ECAL and HCAL which are closer to the interaction point. In the CMS three-dimensional (xyz) geometry, the particles that originate from the interaction point and reach the muon detectors have high-energy levels. Their trajectories resemble almost straight lines in the x-y plane and straight lines in the x-z plane. The muon trajectory signatures make them a good source for simulations in the track reconstruction algorithm. A detailed description of the CMS Muon detector can be found elsewhere [17].

1.3.6 Data Acquisition System

The Data Acquisition (DAQ) system is a compilation of complex software and hardware to process data generated by the CMS sub-detectors. In every collision, the resulting particles interact with various embedded sensors within the detectors to collectively create a picture of each event. The sensors output their data in continuous analogue signals (a combination of noise and data). A series of sensitive electronic devices convert the analogue signals to digital with an acceptable signal-to-noise ratio [18]. The front-end electronic components of DAQ encompass amplifiers, comparators, integrators, quantisers, and convertors [19]. The digitisation encodes the data generated from more than a million channels and thousands of events into binary data streams that logic gates can process. The CMS front-end electronics process raw data captured in events during the data-taking phase. A control mechanism schedules the data processing by generating appropriate signals and flags to middleware components and buffers. The Level-1 Trigger system performs event selection or filtering using fully pipelined custom-designed electronic components. The selected events are sent to buffers, and the rejected events are discarded immediately. The Event Manager subsequently schedules data for distribution to computing services through switches.

In Figure 1.4, the DAQ system with data processing components including Level-1 Trigger is illustrated.

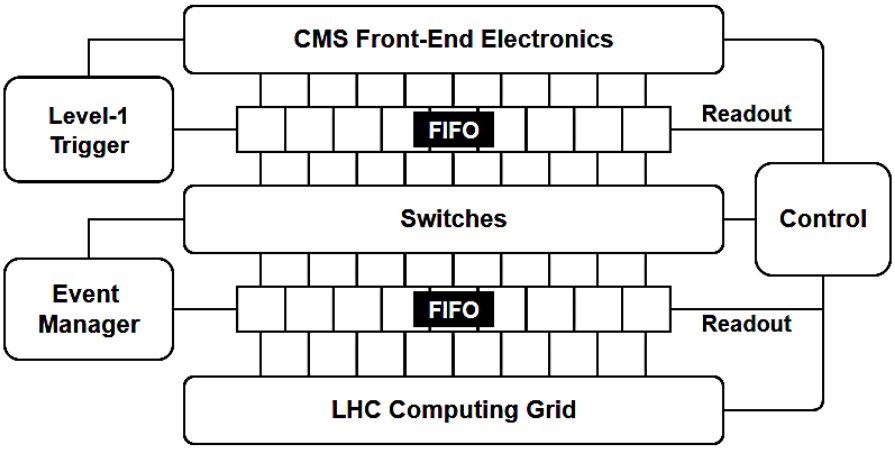

Figure 1.4: CMS Data Acquisition System [20].

This page was intentionally left blank.

Chapter 2

CMS High-Luminosity Upgrade

2.1 High-Luminosity LHC Upgrade

Luminosity (L) in High-Energy Physics is defined as the number of events per second for a given distance and has unit $cm^{-2}s^{-1}$. The High-Luminosity LHC upgrade [18] increases the number of collisions in five consecutive steps (Runs) known as the integrated luminosity period. In integrated luminosity, the performance of the accelerator is evaluated by the measurement of the collected data. The operation commenced in late 2009 and will be finalised in 2040. The research and development life-cycle of the HL-LHC upgrade project follows design study, prototyping, construction, installation and the verification phase in periodic steps. During this time, the functionality of LHC is reduced to a minimum in some periods and a complete stop in others. The latter is known as the Long Shutdown period. The operation takes place in three stages: the first shutdown (LS1), second shutdown (LS2) and third shutdown (LS3). The Particle Physics Project Prioritization Panel (P5) [21] has identified five major targets for the future of the LHC, which can be summarised into two main objectives. The discovery of dark matter through Higgs Boson studies and pioneering instrumentation to explore the unknown. Following the upgrade, LHC will be optimised to accelerate more particles to produce more data. The HL-LHC will rely on several key technological innovations and scientific research to overcome challenges under extreme experimental conditions.

2.2 Planned Upgrade for the CMS Detector

As the collision rate increases, some of the existing detectors that were originally planned for the CMS experiment would not perform well enough. Hence, the CMS detector must be optimised to allow precision measurements of events under increased luminosity. In LS2, the innermost pixel layer of the detector will be replaced with high-luminosity-tolerant and radiation-tolerant components. The pixel sensors will be installed closer to the centre of collisions in a new beam pipe. During the LS3 the new generation pixel devices will replace current pixel detectors entirely. The optimisations for ECAL and HCAL comprise hexagonal silicon sensors and plastic scintillator tiles, distributed over endcaps, providing more information for electrons, photons and hadrons. The outermost muon detectors will be replaced with large Multi-Gas Electron Multiplier (GEM) chambers to increase the precision measurement of scattered muons. These upgrades present new challenges in software and hardware optimisations in preparation for new technologies and advanced physics analysis techniques.

2.3 Motivations for CMS Tracker Upgrade

In Run 1 of the HL-LHC upgrade, optimisations of the CMS tracker delivered successful results of above 95% tracking efficiency for proton-proton collisions with the luminosity of $7.7\text{x}10^{33} cm^{-2} s^{-1}$ which led to the discovery of Higgs Boson. The luminosity increased in Run 2 to $2\text{x}10^{34} cm^{-2} s^{-1}$ and will continue in consecutive Runs until the nominal luminosity of $7\text{x}10^{34} cm^{-2} s^{-1}$ is achieved [22]. Scientists aim to boost the capability to adapt to the higher data rate of superimposed collisions, also known as pileup (PU) [23]. In the pileup, the multiple numbers of individual proton-proton collide per bunch crossing. The mean interaction rate of 140 pileup per collision will create a higher level of particle density and challenges in particle trajectory estimations. The plan to extend the cylindrical section of the tracker toward the beam will extend the coverage of precision tracking and the overall CMS measurement capabilities. The replacement of a large diameter beampipe with a new thinner design will bring sensors closer to the interaction point, introducing new infrastructural challenges in utilising suitable ASIC devices and detection algorithms. Radiation-hardened all-silicon devices will replace

gaseous detectors in the form of system-on-chip integrated circuits. State-of-the-art 50-300 μm n-on-p technology silicon sensors aim to produce robust electronic readout with a faster algorithm for particle detection. All-silicon cost-effective tracker solution will withstand higher irradiation levels with decreased degradation period, infrastructural flexibility for upgrades, and accurate measurements of particle trajectory through high granularity modules. A new pixel detector with 65 million channels in cylindrical dimensions of 4 cm, 7 cm and 11 cm with discs at both ends will receive 10 million particles per square centimetre per second with only 3.5 cm distance from the beam. The pixels are mounted on cooling tubes in rectangular tiles with dimensions 100 μm by 150 μm in several two-dimensional chip layers to create a three-dimensional view of particle trajectories in high occupancy regions [24]. In Figure 2.1, the upgrade view of the pixel modules (red) around the beam pipe (centre of the detector) and surrounding the strip-strip modules (blue and black) is shown.

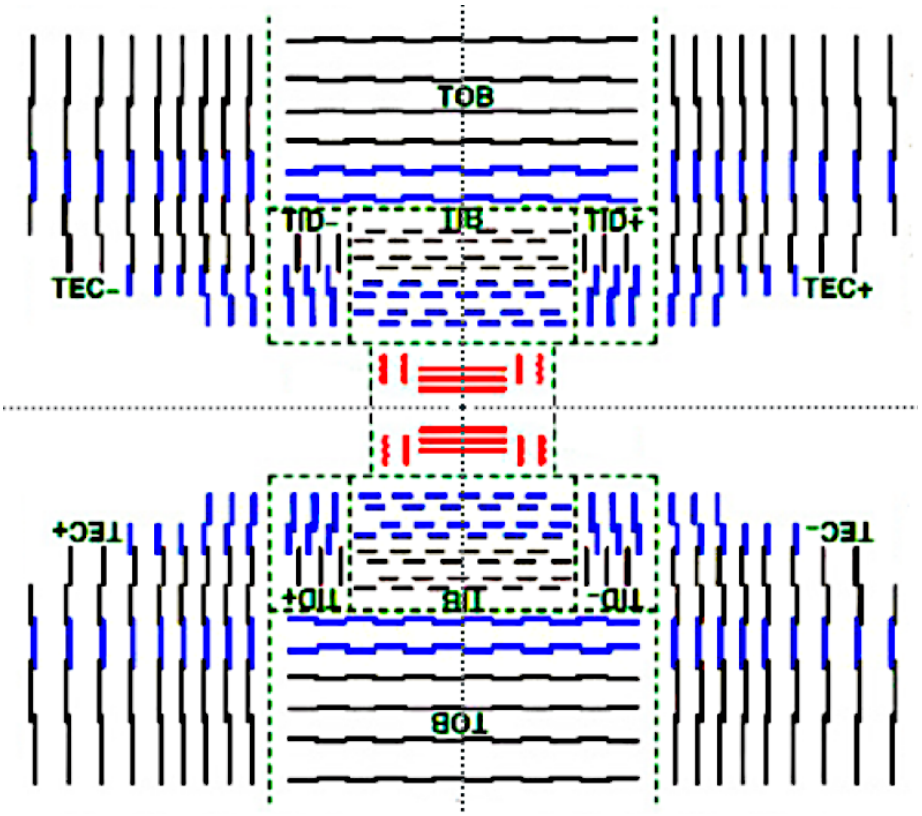

Figure 2.1: Tracker overview with various sensor modules [25].

The barrel region of the tracker wraps around the pixel modules and encompasses four Inner Barrels (TIB), three Inner Endcaps (TID), six Outer Barrels (TOB), and nine outer Endcaps (TEC) or disks on both sides. The colour-coded sensors represent Pixel (red), single strip (blue) and double strip (green) modules on the layers of the tracker. The regions are constructed from concentric layers and small disks with 15,200 sensitive compact modules with 10 million strip sensors and 80,000 readout microelectronic chips. A particle that originated at the centre of the detector may cross more than one region and interact with multiple sensor modules along its path [26].

2.4 Level-1 Trigger in CMS Upgrade

The HL-LHC upgrade and the associated increased pileup will impose challenges in managing higher data rates and limitations in pipelining the processed data efficiently. There are several restrictions on the rate at which the information is processed in trigger systems that primarily fall on low-level circuit implementation and their accumulative latency in the system. The capability to transfer data across a known region is characterised by its bandwidth. The trigger mechanism processes more data within a minimum specified bandwidth by utilising faster electronic devices and paralleled algorithms. If the realisation of faster circuitry proves unfeasible, the system must truncate the data flow to meet the latency requirement of the trigger system, which is not desirable. The trade-offs of truncating data are always between the improved levels of trigger efficiency versus the risks associated with losing data containing useful physics detection. As the triggers are functional during the real-time operation of the CMS, the costs are even higher because once the data is discarded, there is no way of retrieving it. In the Level-1 trigger system, the triggering frequency will be increased from 100 kHz to 750 kHz to achieve the required latency of 12.5 μs imposed by the Level-1 front-end readout electronics. Achieving such latency is made possible through utilising transverse momentum (p_T) modules that are capable of rejecting particles that whose p_T is less than a selected threshold [27].

2.5 Transverse Momentum p_T Modules

The transverse momentum p_T parameter is the first discriminatory factor in reducing data in the real-time data-processing trigger system. A study concerning the CMS tracker indicates that the number of reconstructed tracks is substantially higher in particles with $p_T \geq 2$ GeV/c versus all particles [25]. These particles predominantly produce effects that provide crucial information for understanding the particle's interactions. In Figure 2.2, the tracking efficiency of particles as a function of momentum with p_T range of 0 to 10 GeV/c is illustrated. The study provides a means to differentiate between particles using their p_T as a selection parameter. Identifying and removing particles with $p_T < 2$ will improve the tracking efficiency and results in a reduction in data volume.

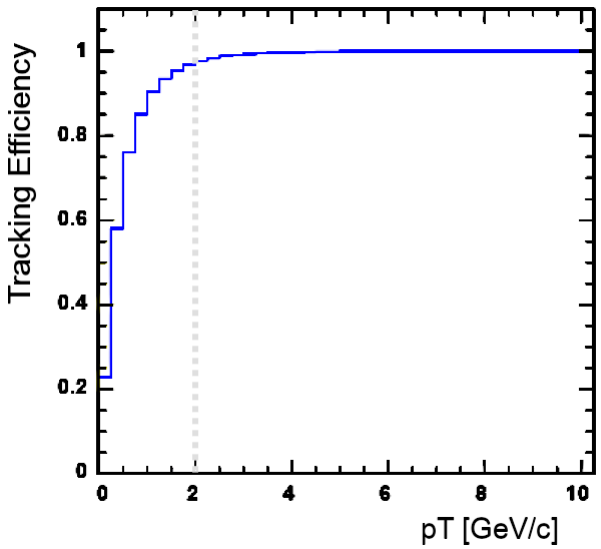

Figure 2.2: Tracking efficiency for $p_T \geq 2$ GeV/c selection [28].

This idea is used to design a real-time track-trigger mechanism to exploit the high p_T tracks relevant to reconstructing exciting physics. The modules use the p_T threshold to reduce the Level-1 outputted data rate to 750 kHz from the initial 40 MHz input. Tracks with lower p_T are immediately discarded. The p_T module design is based on the correlation of particle interactions on two single- and double-sided closely-spaced parallel layers of silicon sensors (strips) and readout front-end ASIC instrumentation [29]. Two hits in the designated area on individual strips create a pair, also known as a

stub. Stubs provide the locations of particle's interactions on a layer of the tracker. The correlation window for the strips varies depending on the location of the modules in the detector and the estimation of the particle's trajectory in the same region. The spacing between strips is greater for modules at larger tracker radii. In Figure 2.3, the concept of low and high p_T with two closely-spaced correlation strips is shown. The arrows represent the particle trajectories with low p_T (left) and high p_T (right) crossing parallel layers of silicon sensors.

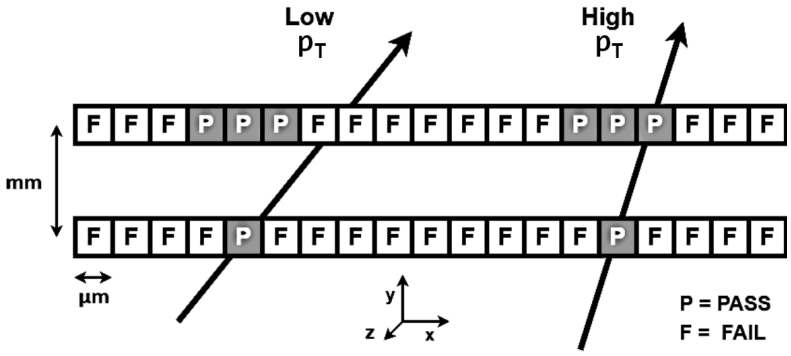

Figure 2.3: Trajectory selection using p_T concept [5].

The particles with high p_T create hits in the pre-configured correlation window. One PASS (P) on the bottom layer and one PASS (P) from the top layer generate a pass for the particle trajectory. If the particle has low p_T, its second hit on the top layer returns FAIL (F), indicating that the particle creating the trajectory must be rejected. Two models of p_T modules are under construction, each suitable for different proximity to the interaction point. Pixel-Strip (PS) in the inner part (at a radius below 60 cm) and Strip-Strip (2S) further outside.

2.6 Tracking Model for Trajectory Reconstruction

The trajectory across prototype p_T modules (PS and 2S) provides initial coordinates of the hit (stub) instance in Cartesian and Polar systems. The global positions are converted into the local points to be used in the track model calculations. In the firmware approach, the segmentation reduces the logic resources and potentially the latency. The significant size of the CMS

detector produces very large global coordinate values that are costly if deployed in hardware directly. The tracker region divisions of $2\pi/9$ in r-ϕ and $2\pi/9/16$ in r-z are illustrated in Figure 2.4.

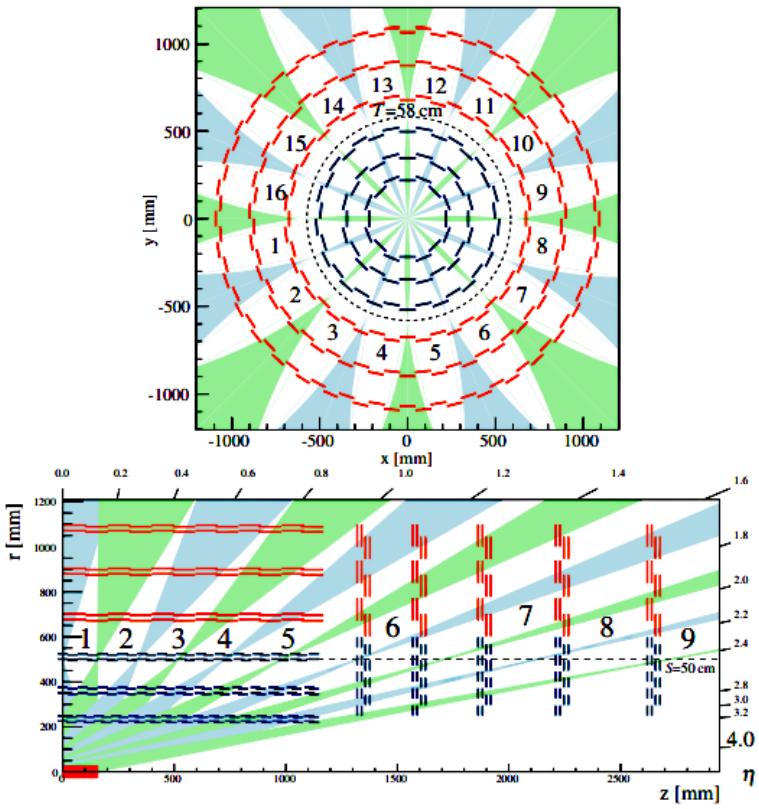

Figure 2.4: Tracker segmentation in r-ϕ (top) and r-z (bottom) [5].

The modules and devices in each segment are addressed through global or local values. To minimise the gap between sectors, *overlapping* with neighbouring sectors is introduced. The particles that interact with more than one sector are often detected separately. The particles that are detected twice become duplicates, and only one is reconstructed following duplicate removal. The chosen values T (58 cm) in r-ϕ and S (50 cm) in r-z in the illustration are offsets to separate Pixel-Strip (blue) and Strip-Strip (red) modules for targeted performance analysis using either of the modules or a combination of two modules. Every segment has all required parameters for track reconstruction with predefined spatial resolution within 1σ of the p_T module position.

2.6.1 Pixel-Strip Modules

The Pixel-Strip (PS) modules are designed from a micro-pixel sensor (pixel-strip) parallel to a silicon strip sensor with a supporting frame and ASICs. A module contains a strip sensor of 2.4 cm in length and a micro-pixel sensor holding 32 rows of 1.5 mm long pixels. The reduced size of the PS modules makes them suitable for the layers closest to the beam, where the particle occupancy and density are at their highest. Different gap-variants of PS modules (1.6 mm, 2.6 mm, 4.0 mm) allow high spatial and p_T resolutions to reconstruct different particle trajectories in the Level-1 Trigger mechanism. The spacing between two top and bottom sensors calibrates to provide sufficient spatial resolution at the right granularity and particle density in different regions of the sub-detector. In Figure 2.5, two parallel top- and bottom-silicon sensor structures that follow the concept of trajectory discrimination are illustrated.

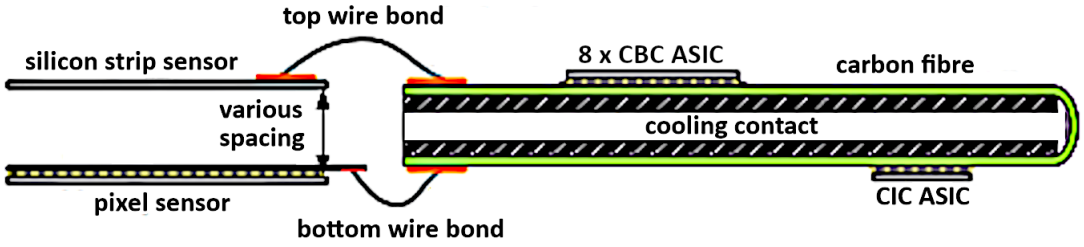

Figure 2.5: Pixel-Strip (PS) module with ASIC device [30].

2.6.2 Strip-Strip Modules

The Strip-Strip (2S) modules are designed of 5 cm silicon strips along with the holding frame and ASIC readout components covering an area of approximately 2,060 cm^2. The internal fabrication precision is 5 μm in the sensor plane and 30 μm in the coordinate plane with an absolute position accuracy of 10 μm. Depending on the position, the geometry of the sensors and the number of readout strips varies. In the barrel region, the sensors are rectangular, and in the endcap, they are structured in a trapezoidal shape to fit on discs. The modules are placed in the Inner Barrel (IB), Outer Barrel (OB), Inner Disks (ID) and Endcap (EC) regions. The realisation of 2S

modules with two different gaps (1.8 mm and 4.0 mm) makes them suitable for the detector regions with lower particle density. In the strip tracker's innermost layer, the occupancy is approximately 5%, decreasing to 2% in the outermost layer. The reason for the approximation is the overlapping placement of adjacent sensors to ensure coverage that can lead to detecting the same hit twice in the same plane. In Figure 2.6, two parallel top- and bottom-silicon sensor structures and ASIC devices are illustrated.

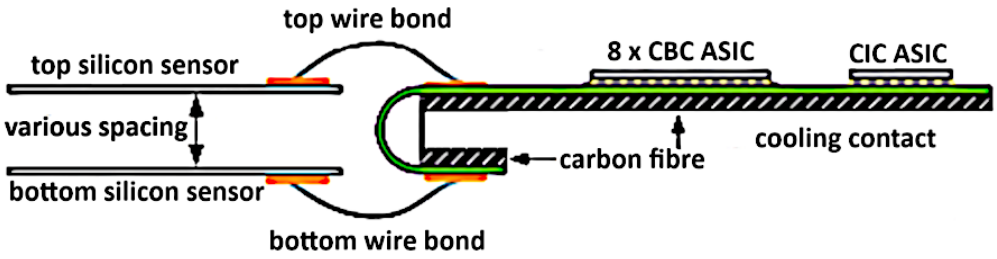

Figure 2.6: Strip-Strip (2S) module with ASIC device [30].

2.7 Front-End and Back-End Electronics

Implementation of ASICs with CMOS technology for tracker front-end electronics is under development in form of CMS Binary Chips (CBCs) design [31]. A CBC will instrument special silicon detectors to identify high p_T particles in real-time that can be used in the Level-1 triggering. The basic functionality of CBC is implemented by 128-channels with an analogue front-end pre-amplifier, and comparator, followed by a binary pipeline RAM. The communications between CBC and the front-end electronics are implemented using the I^2C protocol. The serial communication transfers bits through a single wire synchronously by a clock signal shared between the master and the slave modules. The CBC development aims to meet the final specification of a 254-channel chip designed for the CMOS sensors and the necessary readout logic to identify stubs and transmit them from the module to back-end FPGA devices. The module's output is stored in a 512 kb pipeline memory to handle the trigger latency up to 12.8 μs. The data generated in front-end modules are read and correlated over eight re-configurable CBC boards.

In Figure 2.7, a single CBC with memory, logic components and inputs/outputs is shown.

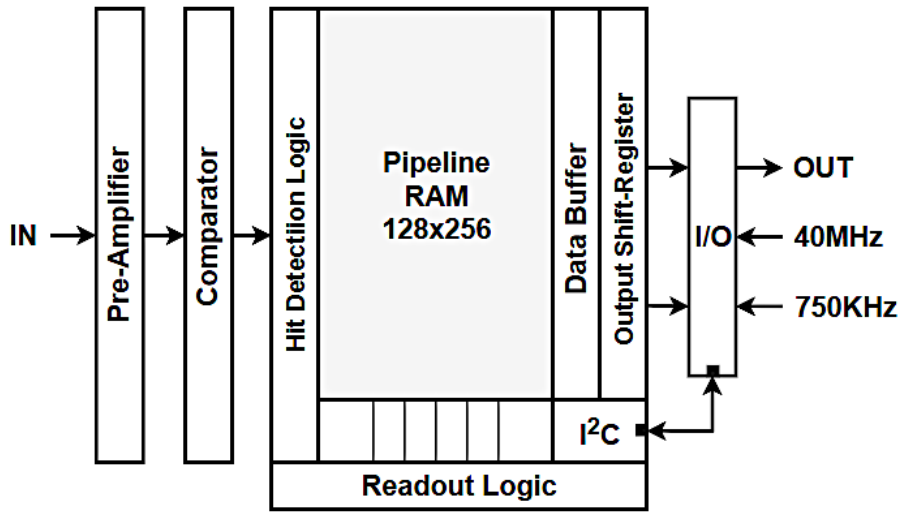

Figure 2.7: CMS Binary Chip and readout electronics [32].

Currently, the trigger system processes data in a pipelined structure at a rate of 40 MHz and latency of 128 bunch crossings with an output rate of 100 kHz. Pipelining data is the most challenging part of data transmission. It requires fast custom-designed electronics with a significant clock control and distribution system through memory partitioning and front-end readout optical links. The 10 Gb/s TCP/IP links interfaced to the FPGA devices function as high-performance switches transmitting on available bandwidth. The total bandwidth is divided between all sub-detectors according to their event allocation within $\pm$10% for the 1 MB event size in the Trigger system. If the size exceeds the nominal total event capacity, the data is scheduled back to the buffers, which can cause a system crash leading to data loss. The application controlling the hardware and data flow is an online software framework known as the cross-DAQ (XDAQ) [33]. The platform is designed specifically to develop distributed data acquisition systems for hardware access, data transfer protocols, and communication services. The Concentrator Integrated Circuits (CICs) [34] are the interconnectors between all CBC modules in the tracker.

In Figure 2.8, a simplified overview of the Level-1 Trigger architecture required for the HL-LHC with its corresponding data rates is shown.

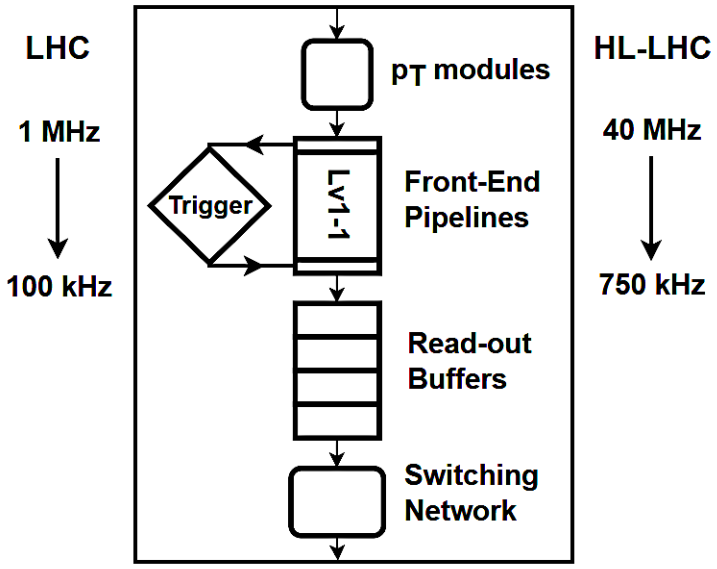

Figure 2.8: Front-end electronic components.

The data generated by the p_T modules and CBCs are grouped into CIC signals and transmitted to Data Trigger Control (DTC) through several Low-Power GigaBit Transceiver (LPGBT) ASIC chipsets. The LPGBT are responsible for assembling the data into packets for transmission and routing. On the generation of the Level-1 accept trigger signal, a frame encapsulates and aggregates the generated hits to off-detector electronics. The average Level-1 data rate will soon increase to 750 kHz per event of 48 hits in PS modules and 54 hits in 2S modules. The capacity of the front-end electronics depends on the number of hits allowed by the CIC aggregation constraints. The data for the trigger is generated in streams of evenly-sized intervals per eight bunch crossing per eight CICs. The overall system capacity per module is 70 stubs per PS module and 34 stubs per 2S module at average rates in interactions containing 200 pileup events. The number of stubs is considerably lower in the layers with larger radii. Full documentation of CBC and CIC developments can be viewed elsewhere [35].

In Figure 2.9, an illustration of Level-1 front-end electronics shows the data transmission through the component's hierarchy.

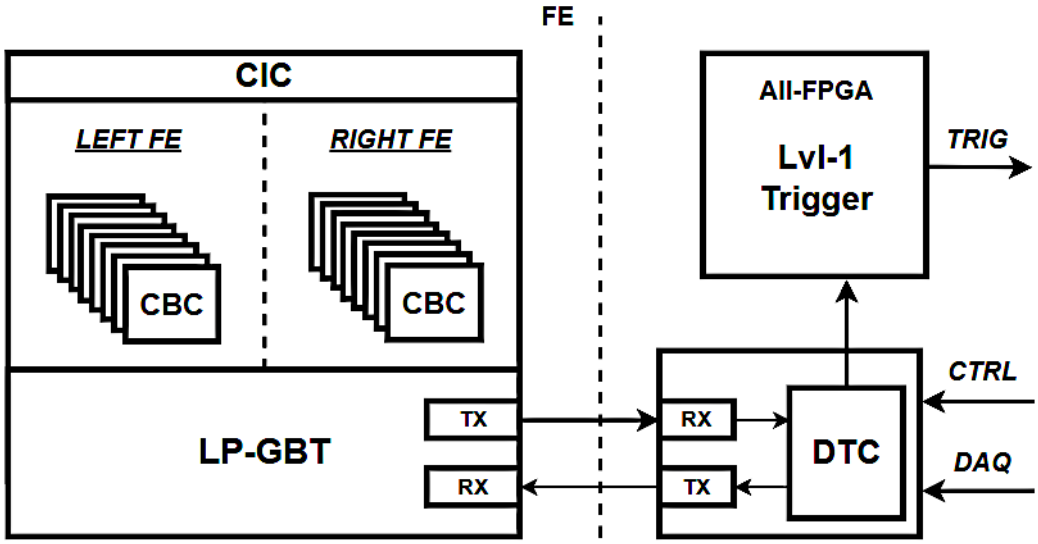

Figure 2.9: Level-1 electronics architecture.

2.8 Tracking Particle Instrumentation

The concept of particle acceleration and tracking is not new. In 1897, Nobel Prize winner Sir Joseph John Thomson experimented with the first cathode-ray glass tube to examine the nature of electric discharge in high-vacuum [36]. The tube consisted of a plate that would emit particles when heated at one end. If the plates in the middle of the tube were electrically charged, the electrodes would accelerate the particles passing through. The accelerated particles would then hit a screen at the other end of the tube, that glowed when heated. Thomson then changed the voltage across the plates and observed that the beam bent as the voltage varied. The experiment led to the discovery of a new subatomic particle known as the Electron.

Figure 2.10 illustrates the instrumentation that led to this discovery.

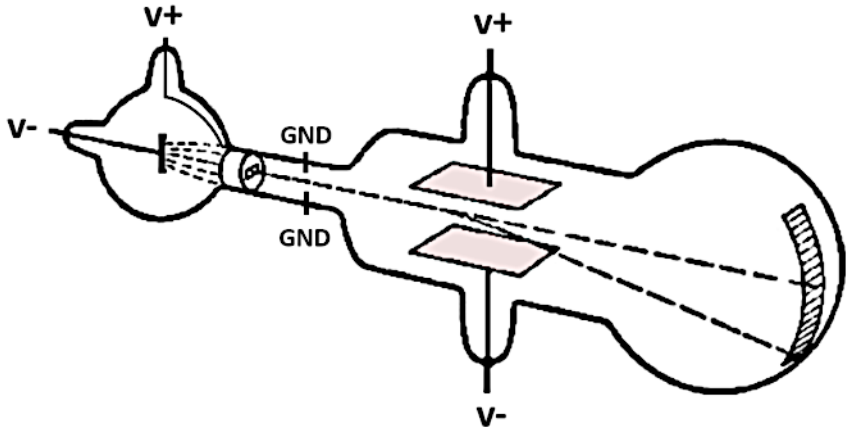

Figure 2.10: Thomson's cathode ray tube [36].

Today's technology deploys a similar approach in the current particle experiment. In the CMS experiment, protons are accelerated in opposite directions in beams before reaching the interaction point at the centre of the detector. The beams are forced to collide at the intersection point creating energetic showers of subatomic particles. The interaction of two particles can result in the destruction of both particles and the formation of new particles. The CMS superconducting magnet is configured to create a constant and uniform magnetic field parallel to the beam line. The motion of a subatomic particle that is affected by the magnetic field follows a curved path. A particle starts its trajectory from the centre of the detector and moves radially outward. The interactions of a particle with different tracking detector layers are recorded by the sensors that make up every layer. Connecting the dots where positive sensor readings are recorded will reveal the particle's trajectory which is known as a *track*. The same rules apply to the particles with sufficient energy to reach the outer regions of the detector. The sub-detector technologies define the type of particles that they can measure and return information to classify the interactions. The main types of subatomic particles that the CMS experiment can detect are Electrons, Photons, Hadrons, and Muons.

2.9 Tracking Parameterisation in CMS

The most efficient design for studying the motion of charged particles in a uniform magnetic field dictates the tracking detector's geometry. The coordinates of the particle's hit and the corresponding sensor's location provide all the parameters required to determine the particle's motion. In reality, the process is a little more complicated than this, bearing in mind that the particle size is smaller than the smallest detection resolution of the electronic sensors. Besides, the closer the detector layer is to the interaction point, the higher the particle occupancy on that layer and the probability that several particles pass through the same sensor is high. For simplicity, this complication is ignored, which means that a hit corresponds to a good reading on one layer of the detector and multiple hits on consecutive layers form a track. The sensor positions are interpolated from one sensor to the next on adjacent layers to reconstruct the first piece of the particle's trajectory. Typically two such hits on two layers are the minimum requirement to define a valid track. The locations of the sensors follow the right-hand CMS coordinate system with the nominal collision point at the centre of the detector as shown in Figure 2.11.

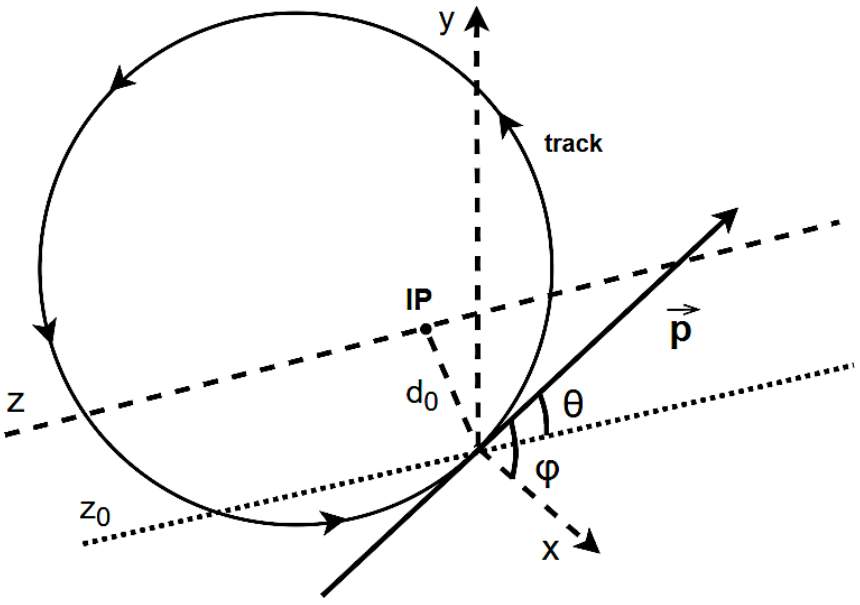

Figure 2.11: Measurement parameters coordinate in CMS.

The structure of the detector is a three-dimensional (3D) cylindrical shape with the beam passing through its z axis, anticlockwise [37]. The parameters are used in the trajectory estimation algorithm throughout the design and verification stages. The relationship defines the association between the CMS coordinate system (x, y, z) and the particle trajectory (track) with polar coordinates θ and φ at the centre of collision with the distance d_0 representing the closest approach of the track to the nominal Interaction Point (IP) in the r-ϕ plane. The vertical axis perpendicular to the z axis is the y axis, and the horizontal axis perpendicular to the z axis is the x axis. All axes have positive and negative values as the interaction point is located at the centre of the detector. If the position and direction of the particles are determined using the CMS global coordinate system, the Cartesian parameters are derived from the momentum $\vec{p} = p_x, p_y, p_z$. The azimuthal angle between the momentum $\vec{p}$ and the x-axis is denoted as φ and the angle between $\vec{p}$ and the z-axis is denoted as θ. The track curvature in the illustration is exaggerated to display the parameters more clearly. In reality, a high-energy track (GeV) only has a small deviation from a straight line in the xy (circumferential) and no deviation in the xz (longitudinal) plane [38].

2.10 Trigger Systems in LHC Experiment

LHC experiment encompasses other projects with general- and special-purpose detectors for investigating physics phenomena and cross-confirmation discoveries. Each detector has a unique triggering mechanism to identify potentially interesting particle interactions, so they can be set aside for further study. During the high luminosity upgrade phase, novel triggering technologies are being developed to ensure detectors can continue taking data with an upgraded LHC. The following sections provide a short overview of operational and under development triggering systems in other main LHC (ATLAS, ALICE and LHCb) experiments.

2.10.1 ATLAS Trigger System

ATLAS (A Toroidal LHC ApparatuS) is one of the large detectors in the LHC experiment, with its sub-detector's layout similar to the CMS detector. ATLAS detector fundamentally had a two-level triggering system. The first-level

trigger is implemented in custom hardware that uses a subset of the detector information from the Level-1 Calorimeter and the Level-1 Muon subsystems to select specified physics signatures. The readout of the accepted events from Level-1 is passed to the High-Level Trigger to be processed synchronously by many CPU cores in the software-based Level-2 trigger system. For the high luminosity upgrade, the detector electronics are being upgraded to increase the Level-1 event acceptance rate by implementing a custom-designed hardware-based Track-Trigger system. The information from newly installed pixel- and silicon-strip modules in the inner tracker are processed by ASIC devices and FPGA technology. The full-silicon application is similar to the CMS detector except for the Associated Memory tracking [21] dedicated for fast track finding in real-time. The Associated Memory algorithm is the central processing element of the tracking system in the ATLAS detector, which detects particle trajectories based on programmable logic devices and dedicated memory infrastructure for pattern matching with a high degree of parallelism. Matched patterns from multiple inner tracker layers are grouped and compared with pre-defined track patterns stored in templates to select trajectories with $p_T > 4$ GeV/c. In the matching process, if a track candidate is selected, linear fitting χ^2 is performed to evaluate track information such as transverse momentum, track direction (φ, η) and impact parameters. The information is combined with other trigger data to accept or reject the track candidates. A detailed description of the ATLAS triggering system can be found elsewhere [22].

2.10.2 ALICE Trigger System

ALICE (A Large Ion Collider Experiment) is one of the general-purpose particle physics detectors designed to exploit the full discovery potential of the LHC. ALICE Central Trigger Processor (CTP) and several Local Trigger Units (LUTs) act as a uniform interface to sub-detector front-end electronics. ALICE has a three-stage hierarchical hardware trigger designed to derive a fast trigger for charged particles with high transverse momentum. The Level-0 trigger sends an early signal to front-end systems to indicate there is an event, followed by the Level-1 trigger that reclassifies the trigger class to characterise the event, followed by the Level-2 trigger to determine whether to accept or reject the event. The CTP has seven VME-based trigger processors [23] that can handle

trigger inputs in parallel, based on multiple custom-designed hardware and independently programmable trigger classes. Information about trigger classes is used at each level of the trigger decision-making process. For the high luminosity upgrade, a novel seven-layer silicon pixel detector will be built close to the interaction point with Fast Interaction Trigger (FIT) to measure beam luminosity, charged-particle multiplicity and azimuthal distribution. FIT trigger and readout electronics are under development as integrated custom-designed processors which allow digital trigger processing and continuous readout for high pileup events. The custom hardware design allows algorithm flexibility for particle identification via the time-of-flight technique as an essential feature of the ALICE detector. A detailed description of the ALICE triggering system can be found elsewhere [24].

2.10.3 LHCb Trigger System

The LHCb (Large Hadron Collider beauty) experiment is one of the LHC particle physics detectors that is specialised for collecting b-physics data. LHCb consists of two-stage triggering system. Level-1 hardware-based trigger reduces that data rate by finding track candidates with high transverse momentum in the muon chambers or high transverse energy in the calorimeter. The selected tracks are sent to the software-based High-Level Trigger to perform a partial track reconstruction to reduce the data rate further. High-Level Trigger is equipped with a software trigger implemented on GPU technology and many CPU processors. A second pass to trigger will select tracks with p_T > 300 MeV/c reducing the data rate even further in the process. For the high luminosity upgrade, the Vertex Locator (VELO) will be replaced by a new silicon pixel detector [25], installed, close to the interaction point with a tracking system that relies on fast pattern recognition and tracks reconstruction in real-time. The development continues to ensure that LHCb trigger and reconstruction algorithms are optimally designed to take advantage of integrated luminosity in the LHC experiment. A detailed description of the LHCb triggering system can be found elsewhere [26].

This page was intentionally left blank.

Chapter 3

Tracking System in Level-1 Trigger

Track reconstruction in CMS refers to the mechanism that utilises software and hardware for estimating the trajectory of a charged particle using the interpolation of its interactions with tracking detector layers. A particle's path begins at the interaction point and passes through the layers of sub-detectors. Reconstructing a trajectory provides crucial information about the particle's characteristics and behaviour. For the High-Luminosity upgrade, reconstructed tracks will be used as part of the decision-making process in the trigger over whether to keep or discard particles based on their transverse momentum. The approach reduces the data volume produced by the upgraded accelerator to maintain efficient storage utilisation. This chapter examines the current and proposed practices for trajectory estimation and track reconstruction in the Level-1 Track-Trigger system.

3.1 Level-1 Track-Trigger Overview

A typical collision event size per bunch-crossing corresponds to approximately 1 MB of data [39]. At 40 MHz, the CMS detector generates approximately 40 TB of data per second. The current technologies do not allow the processing or storing data of this magnitude. An intermediary solution is needed to reconstruct tracks quickly and determine which tracks are worth keeping in real-time data-taking. The hardware-based Level-1 Trigger [40] introduces a mechanism to keep only a fraction of the data and discard the rest. The concept behind the data reduction stipulates that discrimination in transverse momentum leads to the elimination of unwanted

data significantly. Standard CMS event simulations suggest that a high percentage of data produced by collisions are low p_T particles. The focus of HEP is on energetic particles with p_T in the range of GeV/c as they often hold interesting physics signatures [5]. The Track-Trigger at Level-1 aims to deliver track objects with a configurable p_T threshold to the DAQ system within 3 μs latency. Custom-built electronics and front-end readout devices impose timing constraints in building efficient and robust designs in the CMS detector.

The track reconstruction requirements state that the p_T threshold must be kept sufficiently low [41] to maintain high efficiency for potential discoveries. A comprehensive study into the feasibility of all-silicon implementation proposes a pixel-based approach for the inner Tracker and silicon-based strips approach for the outer Tracker in PS and 2S device variations [42]. The prototypes have been tested for durability and resilience at high radiation levels within a ten-year life expectancy [43]. In addition to new sensors, a new layout geometry for the Tracker proposes enhancement in pseudorapidity detection coverage from $|\eta| < 2.4$ to $|\eta| < 4.0$ by installing flat-structured layers very close to the beampipe. The modifications aim to increase the tracking acceptance as a function of η significantly. Further proposals introduce tilted angle modules with supporting conical rings to replace minimum incident-rate flat-structured layers [44]. Rotating the sensor perpendicular to the interaction point (z = 0), or tilted geometry, increases the particle hit coverage and tracking efficiency.

In the area of data transmission, a new architecture for fast readout systems proposes the Telecommunication Computing Architecture (TCA) standards to host a custom-built FPGA-based Advanced Mezzanine Card (AMC) for linking the Tracker Front-End (FE) electronics to μ-TCA crates. The ATCA and μ-TCA devices are extremely fast with a downlink of 2.5 Gb/s and uplink of 10 Gb/s, transmitting data from on-detector to off-detector electronic devices. The technology and its feasibility in the CMS experiment are under study [45]. As discussed in the previous chapter, low-power gigabit transceivers control the readout electronics between p_T modules and the DTC boards. The data is formatted in transmission packets and payloads. The packets are unpacked to extract information concerning the hit coordinates

and regional information. The data is reformatted into stub information associated with a given event and bunch-crossing. The off-detector electronics for the Level-1 Track Finder receive the stubs for particle trajectory reconstruction through a chain of hardware- and software-based modules. Managing around 15,000 stubs [46] per bunch crossing in a sequence of 25 ns intervals requires a fast and efficient data processing system. The following sections discuss the requirements, techniques and approaches to achieve such performance.

3.2 Level-1 Tracking System Requirements

The HL-LHC upgrade will increase the luminosity to 7.5x10^{34}cm$^{2}s^{-1}$ with an average 140 pileup [47]. The individual regions of the CMS Tracker are under development to significantly optimise the data processing mechanism to maximise the exploitation of the produced information. The data generated at the centre of the CMS detector is aggregated through various ASIC and FPGA modules to reach the off-detector data farms. The data is transmitted in binary formats in streams containing the location of the stubs whose transverse momentum is greater than the specified threshold [48]. The Level-1 tracking aims to process stub streams and find tracks. The requirements for an efficient tracking system are:

- Compatibility with front-end and back-end ASICs.
- Ability to identify tracks for the Tracker in the range $|\eta| < 4.0$.
- Tracking efficient for all particles with 2 GeV/c $\leq p_T <$ 100 GeV/c.
- Tracking particles in high pileup events within specified η and p_T ranges.
- Producing valid tracks with a latency of less than 3 μs.
- Robust tracking efficiency in both low pileup and high pileup scenarios.
- Compatibility with floating-point and fixed-point simulations.
- Agreement with Level-1 Track-Trigger Demonstrator for emulations.
- Instantiation in paralleled Tracking system as self-contained IP core.
- Maintaining performance in standalone and integrated development.
- Scalability in software and hardware architecture for the entire system.

The requirements can be extended to be compatible with the Continuous Integration (CI) [49] verification platform and extensible clang-tidy framework for diagnosing and fixing typical programming errors in real-time. Clang-tidy is a clang-based *C++* tool for customisation of sanity and correctness, checking and fixing errors at the early stages of the algorithm development and before version release.

3.3 Level-1 Tracker Layout Modeling

Various modules containing one or two types of single- or double-sided sensitive sensors occupy different radii[1] of the CMS detector layers. Three layers of pixel and strip sensors cover the radius of 4.4 cm < r < 10.2 cm with endcap disks at both ends. The 2S sensor modules cover the radius of 20 cm < r < 110 cm in closed-ends barrel-shaped regions. The Tracker geometry follows the D49 (T15) tkLayout with optimised values for efficient coordinate digitisation. The tkLayout [50] is a software package developed to create three-dimensional models for the design of the CMS Tracker and evaluation of the Tracker performance. The package assists developers to design an architecture that offers the best trade-off among many figures, such as physical positions of hardware modules, the layout of the Tracker layers and the tracking resolution to explore the feasibility of innovative solutions for the tracking detectors. In the layout, r and z coordinates represent the distances of the sensor layers from the interaction point in the barrel and endcap, respectively. The ϕ coordinate gives the azimuth angle of at least one segment of the Tracker ($2\pi/9$) regions. In Figure 3.1, the layout of the Tracker is illustrated. In this geometry, the double-sided strip modules [2] are rotated by a 0.1-radian angle facing the centre of the detector. The Upper-Left plot shows the Tracker tilted geometry with Pixel-Strip (PS) modules in blue and Strip-Strip (2S) modules in red with pseudorapidity $|\eta|$ describing the angle of a particle relative to the beam axis (z) in positive $(r-z)$ plane. The Upper-Right plot shows the Tracker modules in the x, and y plane, Lower-Left shows the number of layers in the $|\eta|$ range, and Lower-Centre presents the number of hits in the η range, and the Lower-Right quantifies the distribution of track objects.

[1] Fixed step-size distance interval from the CMS interaction-point to Tracker layers.
[2] Two single-sided modules mounted back to back to produce the hit information in 3D.

The geometry is configured to cover most instances of a particle trajectory up to $|\eta|$ = 4.0.

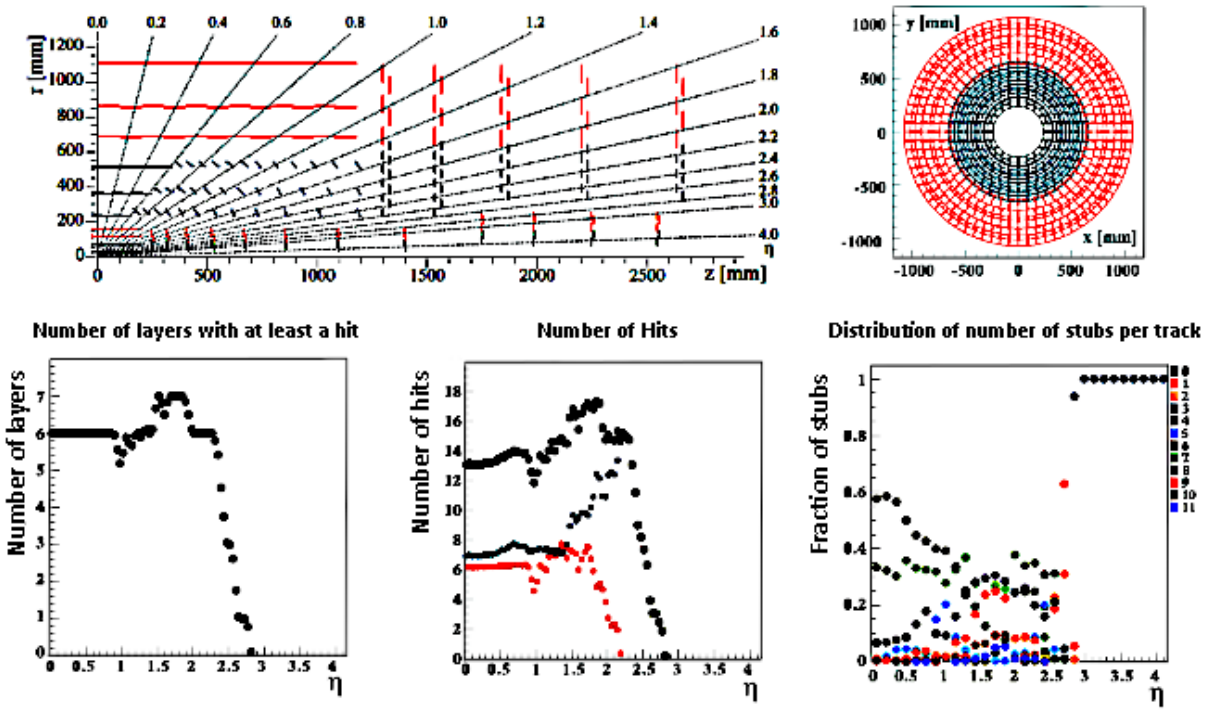

Figure 3.1: Tracker layout in D49 (T15) arrangement [51].

A particle trajectory creates hits on a maximum of seven layers in the selected layout, and the number of hits for a single layer does not exceed four per processing stream. The boundaries of Tracker segments are identified in terms of η and ϕ. The η is used as a spatial coordinate concerning the angle θ and the beam axis. The relation is defined in 3.1:

$$\eta = -ln\left[tan\left(\frac{\theta}{2}\right)\right] \tag{3.1}$$

The segmentation of 0.0 < η < 4.0 values encodes into {0.0, 0.20, 0.41, 0.62, 0.90, 1.26, 1.68, 2.08, 2.50} η-sectors in flat geometry and {0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0} in tilted geometry. The segmentation of ϕ at r > 55.0 cm decodes into {-6.2832, -4.7124, -3.1416, -1.5708, 0.0, 1.5708, 3.1416, 4.7124, 6.2832} radians for ϕ-sectors. These chosen values are important parameters that are used in the tracking systems throughout the software and hardware implementation.

3.3.1 Hit Clustering

Track reconstruction begins with the calculation of the binary information transmitted from all p_T modules. The algorithm groups the local sensor readings, configured threshold, and accumulative charges in the neighbouring pixels and strip sensors according to the identified spatial regions. The grouping is known as Hit Clustering, which identifies the primary vertex and hits produced by particles. The PS modules provide the best spatial resolution because they provide two-dimensional measurements. A cluster defines the particle's transverse momentum p_T, position and direction, calculated through an iterative template-matching and weighted-averaging correction algorithm [25].

3.3.2 Seed Generation

A seed is defined when two particle hits are detected on two adjacent layers or disks. In several track-finding algorithms, the formation of seeds is the first step in calculating the trajectory of a particle originating from the beam spot. A seed contains approximations of all track parameters on individual layers of the Tracker. The extrapolation of detected seeds can reconstruct the entire trajectory of a particle [25].

3.3.3 Track Finder Modeling

Track finding is an essential part of the track reconstruction and is widely used in CMS off-detector algorithm approaches [52]. A reconstructed track encapsulates the character of the particle that produced it. The process is usually an iterative procedure that uses the initial primary vertex and the subsequent detection on a layer-by-layer basis to determine its trajectory. A track may still be considered genuine if the hit positions do not occur in every layer. If the track-finding algorithm determines that a particle has generated both hits, it adds the new candidate to its partially reconstructed trajectory and searches for the next hit. A robust track finder must reduce the number of probabilities and consequently decrease the number of searches. The cost in terms of timing and resources is too high otherwise and must be avoided. In the hardware implementation, the logic utilisation of the search algorithm must be kept to a minimum, possibly to a single iteration. The resource and

latency estimations are obtained in single iteration modules to present a more precise design evaluation. The new modules must be compatible with the track finder for integration in the Track-Trigger firmware chain.

3.3.4 Track Fitter Modeling

The track fitting terminology in conventional track reconstruction refers to the process initiated after the track-finding ends. A track fitter processes all the hit occurrences in the track-finding stage to determine the best possible trajectory. The algorithm used for track fitting must reduce the probability of misidentified hits contributing to the final determination of the particle trajectory and use the initial parameters combined with the geometric information to minimise the duplicated tracks. Duplicates are generated when the algorithms employ a particle hit to find more than one track. In the realisation of a track fitter, one hit belongs to one track only and finding multiple tracks with sharing hits must be avoided. In recent years, the track finding and track fitting terminologies have been used interchangeably in track reconstruction stages [48]. The advancement in technology and hardware acceleration platforms allow both algorithms to be developed in a single module. The track-fitting algorithms are also used in the final goodness of fit estimations and selection of candidate tracks using χ^2 fit calculations in software and hardware.

3.4 Motivation for Hardware-Based Tracking

Since installing the first detector at the LHC, developers have conceptualised desired behaviours of trajectory estimation systems and used algorithms to implement their theories running on commercial Central Processing Units (CPUs) [53]. The approach is solely software-based, orchestrating the fetching and execution of instructions by Arithmetic Logic Units (ALUs). Although the software approach is convenient and relatively fast, it has limitations imposed by the CPUs. One of the main constraints in using only commercially available CPUs is the limitation in programmability in a low-level algorithm abstraction and high-level parallelism. Other constraints are the limitations in designing asynchronous systems and inflexibility in partitioning the system into smaller subsystems if required. In the HL-LHC experiment, the

latency, or the accumulative time that individual blocks consume to produce valid outputs is the fundamental constraint in developing an optimal hardware-based system. Various methods of task scheduling and pipelining architectures have been studied and used in multiple applications for the CMS upgrade to avoid the latency limitations [54]. The pipelining efficiency relies on the techniques that store and execute instructions in order by logistically managing resources. In a pipelined structure, the execution of tasks overlaps to increase the throughput and reduce latency at different levels of the design hierarchy.

FPGA semiconductor technology is mainly designed for flexible pipelining based on the consumer's requirements. Hardware Description Languages such as VHDL and Verilog allow the customisation of desired algorithmic behaviour at Register-Transfer Level (RTL) abstraction. The combination of sequential and combinational data flow in the structural or behavioural levels creates semi- and fully-pipelined designs by effectively controlling logic gates, memory structures, and timing resources. In recent years, the Very High-Speed Integrated Circuit (VHSIC) design has gained popularity as the demands for complex hardware implementation have risen. The CMS Collaboration has announced plans to explore the FPGA technology in ASIC and System-on-Chip designs for the Level-1 Tracker and Trigger systems in preparation for 200 pileup events [55]. These strategies aim to filter tracks to the lowest number possible without compromising the potential discovery of exciting physics. Designing such a robust system using only CPUs is not feasible, leading to costly and redundant subsystems.

3.5 Motivations for High-Level Automation

The computer application development's life-cycle has become exponentially complex in the past decade, particularly in various phases of hardware testing and verification stages. When electronic circuits were not too large, developers coded the entire hardware behaviour in Register-Transfer Level (RTL) or Gate-Level Description Modeling to create a high-level representation of an electronic circuit. After the testing and debugging stages, developers mapped the algorithm into target FPGA devices and sent them for mass

production. At this stage, the probability of failing to achieve the client's specifications was too high. If the design had an undetected flaw, modifying or redesigning the entire hardware was too costly and time-consuming, as often, many areas of the hardware require redeveloping. Computer algorithms and applications have continuously grown in complexity and, as a result, so have hardware development life cycles. The need for automated design tools has become evident since Hardware Description Language (HDL) implementation of the complex algorithm has frequently delayed the design cycle.

The solution for a hardware-based tracking system proposes a composite deployment of Hardware Description Language (HDL) and automated design tools to develop the Level-1 Tracking system. Designing and pipelining hardware-based complex equations in high-energy physics is almost impossible through conventional hardware languages within specified timing and resource requirements. Hence, the algorithms with the lowest complexity are programmed in HDL, and complex algorithms are programmed in High-Level Synthesis (HLS) [56] automation design [57]. The technique supports the development at a higher abstraction level while promoting control and access to a low-level hardware hierarchy. The HLS describes hardware in Object-Oriented Programming (OOP) language *C*, *System-C*, *C++* using Electronic Design Automation (EDA) tools to construct low-level logic structures automatically. The time often spent in the design verification phase is primarily utilised in optimisation by analysing concurrency, critical path, resources usage and latency issues. HLS tools can be integrated into the CMS software framework to evaluate algorithms on both software and hardware platforms. In designing hardware for the trajectory reconstruction of the Level-1 Tracking system, HLS has enabled developers to design, test, and verify the algorithm's feasibility on several FPGA boards simultaneously. In particular, the Level-1 Track-Trigger Demonstrator of the track finder and track fitter, where the physics equations are computationally demanding and complex. Automation practices divide the modules into smaller blocks separated by their data-flow characteristics, arithmetic equations, or top-level design. The modules containing top-level and data flow functionalities are programmed in HDL, and the modules with arithmetic equations are designed using HLS tools on a hardware acceleration platform.

3.6 Level-1 Tracking Architecture

The architecture for the Level-1 Tracking system follows the track reconstruction model in four stages seeding, tracking, fitting and selection. In the Seeding stage, the hits known as *stubs* are detected using clusterisation. The stubs have the primary characteristics of an incomplete trajectory of a particle. In the tracking stage, trajectories are estimated by finding the stubs that share the same trajectory across other layers of the Tracker. In the fitting stage, the candidate tracks are fitted to the nearest trajectory. The duplicate tracks are identified and removed in the selection stage. The duplicate tracks are formed when a hit is used to construct more than one track. In this stage, the fake tracks are also identified and removed. A fake track is formed when a trajectory is reconstructed falsely by the combination of unrelated hits.

3.7 Level-1 Track-Trigger Electronics

This section explains how the tracking modules are integrated into the tracking system architecture for track reconstruction within latency specifications of Level-1 Track-Trigger. The front-end electronics aim to reconstruct particle trajectories with $p_T \geq 2$ GeV/c at intervals of 25 ns, corresponding to the 40 MHz frequency of HL-LHC bunch-crossings. The maximum readout latency across front-end electronics permits 12.5 μs for all sub-modules in the Level-1 Tracker system. The timing includes the complete reconstruction and track formation after the correlator module. From the 12.5 μs time allowance, only t $\leq$ 3 μs is assigned to the Tracking system [58]. This is a fixed latency generated by the hardware components and can not change. In individual building blocks of the Level-1 Tracking architecture, bandwidth, latency, and maximum clock frequencies are predefined by the system. All new modules must be designed in compliance with these parameters. The DTC components are the first reconfigurable blocks in the Tracking system. They are responsible for organising and transmitting stubs from different regions of the Outer Tracker to the rest of the modules down the firmware chain. Figure 3.2 shows the Level-1 Tracking architecture.

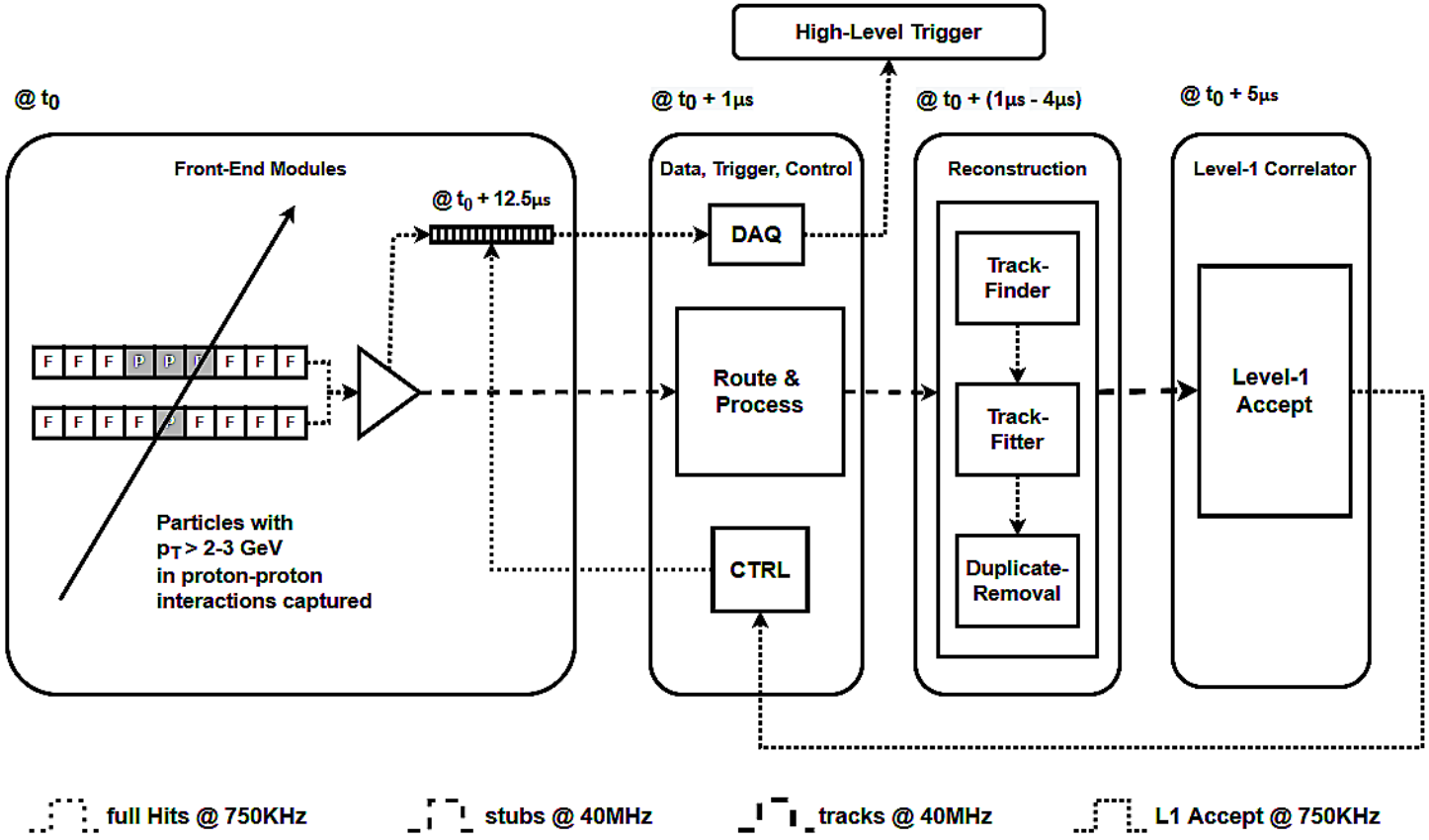

Figure 3.2: Level-1 Tracking architecture.

The data sent from the detector regions to the DTCs are fully pipelined and structured in a parallel configuration of 324 blocks for the entire system. The data transmission between DTC blocks is handled by switching networks and μ-TCA technology [59] with a high data rate. The Track Finder blocks are chained to the DTCs in the architecture. The blocks are responsible for converting stub data to track candidates.

3.8 Data, Trigger and Control

The data generated by the p_T modules are transmitted across LPGBT links to the DTC modules at 25 Gb/s clock frequency [60]. A DTC processes up to 72 links at a time. The number of links follows the load balancing[3] scheme and better data throughput. In the Level-1 Trigger, latency commences from the moment stubs appear at the input of the DTCs until a full track candidate is an output. The structure of a DTC is composed of custom-developed FPGAs with multi-channel optoelectronic transceiver blades in programmable μ-TCA technology for pre-processing (organisation, formatting, and routing) and stub transmission to both Track Finder and DAQ subsystems. The DTC is also responsible for the timing, control and calibration of data and communication protocols. One possible variation in Track-Trigger architecture corresponds to 32 links at 16.3 Gb/s per DTC with a total number of 288 modules for the Outer Tracker [58]. Reducing the number of links per DTC is desirable and preferred. Having fewer links results in fewer peripherals and lowers the associated costs of the interface connectivity between DTCs and track finder modules.

3.9 Time Multiplexing in the Track-Trigger

In trigger time-multiplexing, [61], data is transmitted across a single node to avoid data sharing. The buffers can be configured to hold the data generated from multiple events and route them to one node per event. For a general time-multiplexed system, if the multiplexing factor is configured at n events, n nodes are required at n x 25 ns intervals, one clock phase apart. Nodes are assigned to the processing boards individually with a configurable time-multiplexing factor depending on the FPGA chip resources capability

[3]Load balancing redistributes the workload in a parallelised system to avoid an idle state.

and timing constraints. The challenge for a successful multiplexer is the synchronisation of all data in serial-to-parallel conversions; hence, careful clock distribution and timing management are necessary. In the first clock transition, enabling the read signal allows switches to read the stubs and distribute them to down-chain components. In the second clock cycle, stubs are written and read from the next buffer. The process continues until the last buffer has completed a write and read transition. In Figure 3.3, the demultiplexing, buffering, and switching processes across the Tracker front-end electronics are illustrated.

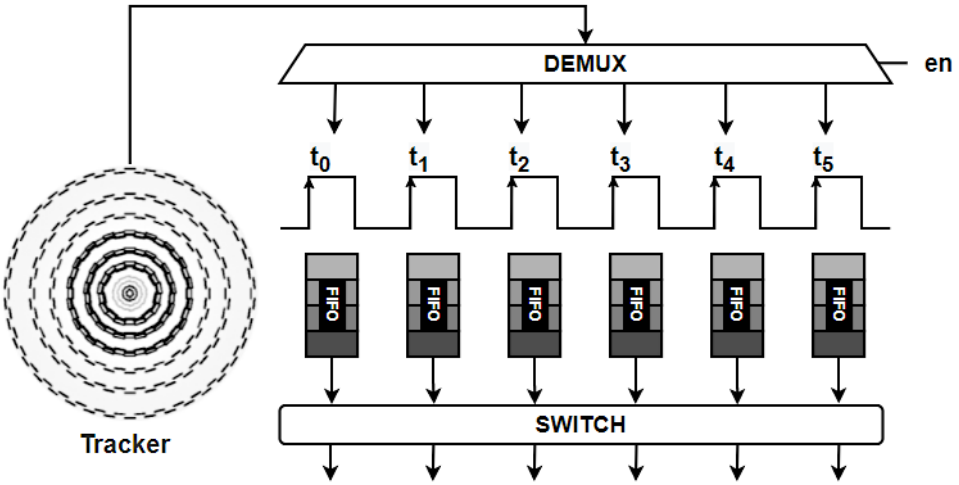

Figure 3.3: Level-1 Time-Multiplexing with a factor of six [61].

The advantage of using time multiplexing is the capability to see a particle's path across a large section of the Tracker if the hits that contributed to the track reconstruction are detected in one event.

3.10 Regional Segmentation

The data stream generated by each collision is time-multiplexed into a fixed number of nodes; however, the limited number of input links on each FPGA board requires additional data segmentation. The regional subdivision of the Tracker separates each of the Tracker regions into symmetrical sections. A solution [58] for the Time-Multiplexed Track-Trigger (TMTT) divides the Tracker into nine π regions over the full range of z as shown in the Figure 3.4.

In this configuration, the regions are called *nonants*, where it is ensured that all stubs required to reconstruct tracks are contained within a single processing section.

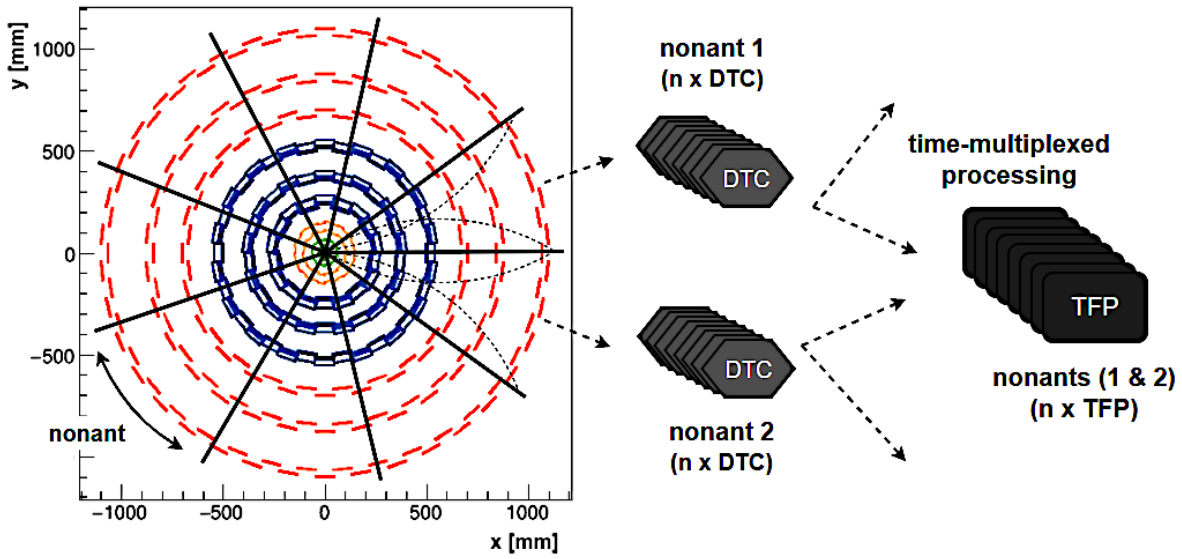

Figure 3.4: Regional segmentation of Outer Tracker [58].

The regional segmentation view presents a two-dimensional perspective of the Tracker regions. In the cylindrical three-dimensional geometry, Each nonant is further divided into ϕ and η regions known as *stub sectors*. The ϕ, η sectors are hard-wired to DTC blocks. A Track Finder Processor (TFP) block can process data from two adjacent nonants. The scheme minimises the number of TFP blocks by half. In the prototyping stage, that building the entire system is costly, only designing one system out of nine can prove the feasibility of the entire system. Once the resource usage and latency requirements are verified, the variables are multiplied by the nine to determine the costs for the system for all nine nonants. Other benefits are parallelisation of individual nonant without data dependency on other nonants, creating efficient testing modules in verification stages for one section instead of all sections, simplification in algorithm debugging and reducing the synthesis period in the development life cycle. There are dotted arcs concerning the area of high efficiency in the Tracker segmentation that is described in Chapter 6.4.

3.11 Track Finding in the TMTT Approach

Up to this point, the stubs generated in the Tracker are time-multiplexed and segmented into smaller regions. The following sections describe the algorithm and hardware components that are used in trajectory estimation and track reconstruction. The TMTT 3.11 approach has been studied and developed for the high-luminosity upgrade of CMS experiment [58]. The TMTT configuration relies on several blocks that are instantiated in the firmware chain as Geometry Processor, Hough-Transform, Kalman Filter, and Duplicate Removal modules to reconstruct tracks from stub streams. The TMTT defines a Track Finder Processor block that contains all the above modules for each processing nonant of the Tracker.

3.11.1 Geometric Processor

The data from DTC blocks are transmitted to the GP blocks for unpacking and pre-processing, where the global coordinates of each stub are extracted, and further regional segmentation is implemented. The GP module defines stub sectors by subdividing the nonants ($2\pi/9$) into 18 x η sectors in the $r-z$ plane and 2 x ϕ sectors in r-ϕ plane [58]. A TMTT Track Finder Processor (TFP) handles 72 (η, ϕ) stub sectors through (2 x 36) DTC blocks. In Figure 3.5, the regions and boundaries for TMTT tracker segmentation are shown.

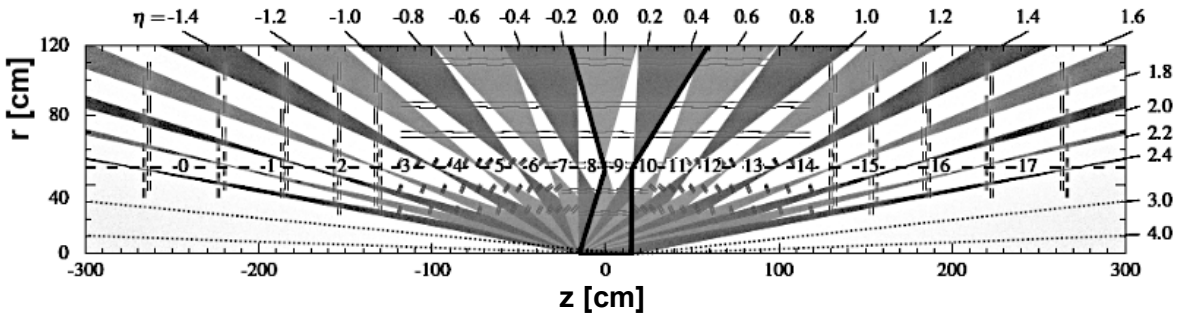

Figure 3.5: Segmentation of r-z into (η, ϕ) sectors in GP [62].

3.11.2 Hough Transform

The stubs from the GP block are transmitted to Hough Transform (HT) block for the primary track reconstruction. In HT, a stub from a stub sector is mapped to a line in the Hough space. Many stubs produce many lines that intersect. The Hough space is divided into n x m cells where the intersection occurs. The intersection returns a coordinate that can be used to reconstruct a trajectory in the Tracker. The probability of identifying duplicate tracks in HT is a disadvantage caused by the smallest resolution in cell granularity. If the intersection of hits belonging to a particle occurs in two adjacent cells, the algorithm assumes that two tracks are detected. Different techniques are used to identify duplicates and remove them in the Duplicate Removal stage of the TMTT algorithm.

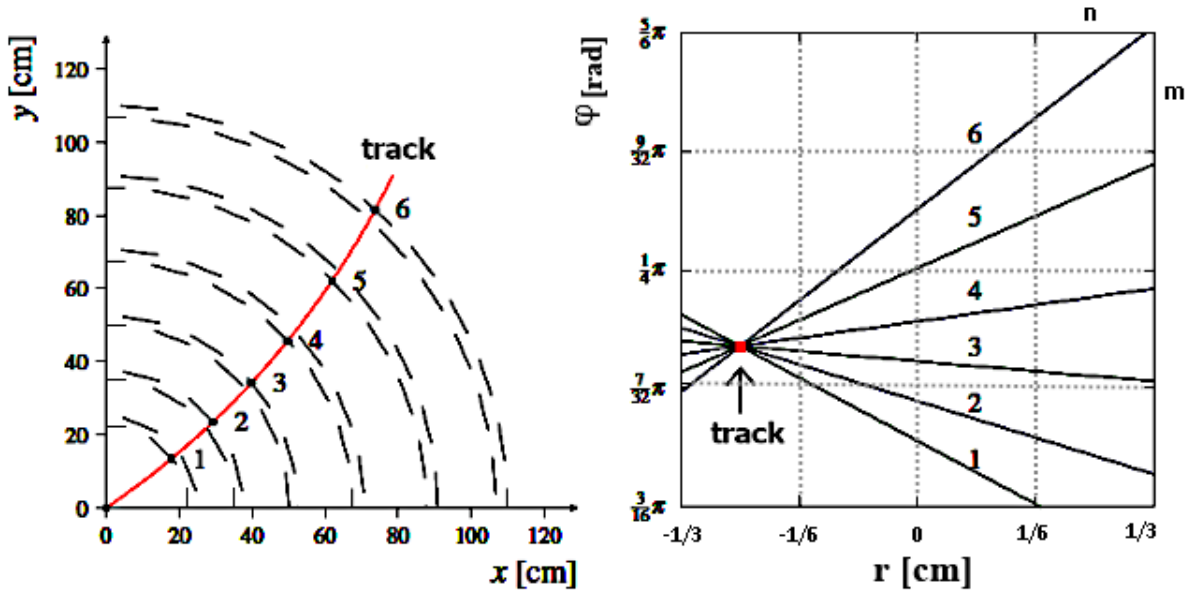

Figure 3.6: The transformation from stubs to a track in HT [63].

The HT module is the first computational block in the Time-Multiplexed Track-Finder firmware chain. The modules that provide input to the HT are DTC and GP. Both modules are responsible for data routing and data segmentation with no arithmetic operations. In software, the application of the HT algorithm has been studied in offline particle trajectory estimation [63], however, in the hardware for the tracking system, the implementation

has never been attempted before.

A compact version of a fully pipelined HT system is implemented in the TMTT Demonstrator targeting Xilinx Virtex-7 FPGA chip running at 240 MHz clock frequency. In the demonstrator Chapter 8 a prototype of two processing nonants is implemented and the evaluation of feasibility in physics and hardware performance are discussed. The selected chip has enough logic resources to accommodate eight single iterations of fully pipelined HT modules. Each iteration receives one stub and produces one line. For a stub stream containing eight stubs, eight iterations are required. An alternative implementation using the Xilinx Kintex Ultrascale+ and increasing clock frequency to 320 MHz reduces resources substantially [64]. The faster clock frequency reduces the time required for processing stubs and decreases the utilisation of multiplexers and buffers in the design leading to less costly hardware implementation.

3.11.3 Kalman Filter

The Kalman Filtering (KF) system is essential to the Level-1 TMTT system as an optimum state estimator. The Kalman System prototype is implemented and tested in the Level-1 Track-Trigger Demonstrator firmware. The conventional Linear Quadratic Estimation (LQE) algorithm [65] is at the core of the Kalman system, which is primarily used in probability measurements and error corrections to the HT trajectory estimation. The track parameters generated by the HT modules are transmitted to the Kalman system for validation and filtering [58]. State generation in the Kalman system combines all Hough-space track coordinates plus uncertainties and noise generation by front-end electronics. The Kalman system links to the HT through the helper mini-HT module responsible for parameterising and routing the outputted parameters from HT to the KF based on a series of vectorised (row x column) parameters. HT simulations indicate that track finding measurements are affected by stub sharing between genuine and fake tracks. The early observations showed that almost 50% of the overall fitted tracks in HT share one or more stubs from other tracks, negatively impacting tracking efficiency in physics performance simulation. The KF aims to detect and eliminate incorrectly reconstructed tracks. The advantage of deploying the Kalman

filtering algorithm for trajectory estimation is arithmetic precision in measurements; however, this comes at a cost in hardware implementation due to the involvement of covariance matrices and complex recursive mathematical operations that are described in Chapter 5.

3.11.4 Duplicate Removal

The Duplicate Removal modules link directly to the Kalman System in the TMTT firmware architecture and possibly the Hybrid system 3.12. Despite the effectiveness of the KF efforts in eliminating unwanted duplicate tracks, the simulations indicate that almost half of the track candidates are copies in the overall track reconstruction. One of the cases occurs when a stub in the Tracker coordinate system transforms into many tracks in Hough space. The discretization of these cells is determined by two factors, rows and columns. The granularity of the cell can be changed trivially in software by choosing larger values leading to more resolution per cell, however, this can lead to additional costs in hardware implementation as the cells are implemented using memories. There is a limited number of memory blocks on an FPGA chip.

The creation of many identical tracks affects the accuracy of physics simulations after the fitting process. An algorithmic solution to remove duplicates has been implemented for the Level-1 Track-Trigger in software and hardware [58]. The hardware components of the module are designed using available logic slices in the HDL IEEE STD-1164 library [66]. The algorithm looks for the track parameters initially identified and generated in the Hough cells. If the track parameters do not originate in a single cell, the track is detected more than once and must be removed. The Logic Block in the DR receives a copy of the event from the HT module before the KF commences its processing. The event contains information concerning all HT reconstructed tracks and their cell locations. Once the KF has reconstructed a track, it is stored in the FIFO block to be used for equality check against all HT tracks. Two tracks with matching coordinates from two different HT cells are duplicates, hence, one copy is stored in the RAM block and the second copy is discarded. Two multiplexers and a demultiplexer ensure that the correct data is selected for processing by the Logic Block.

Figure 3.7 shows the Duplicate Removal architecture.

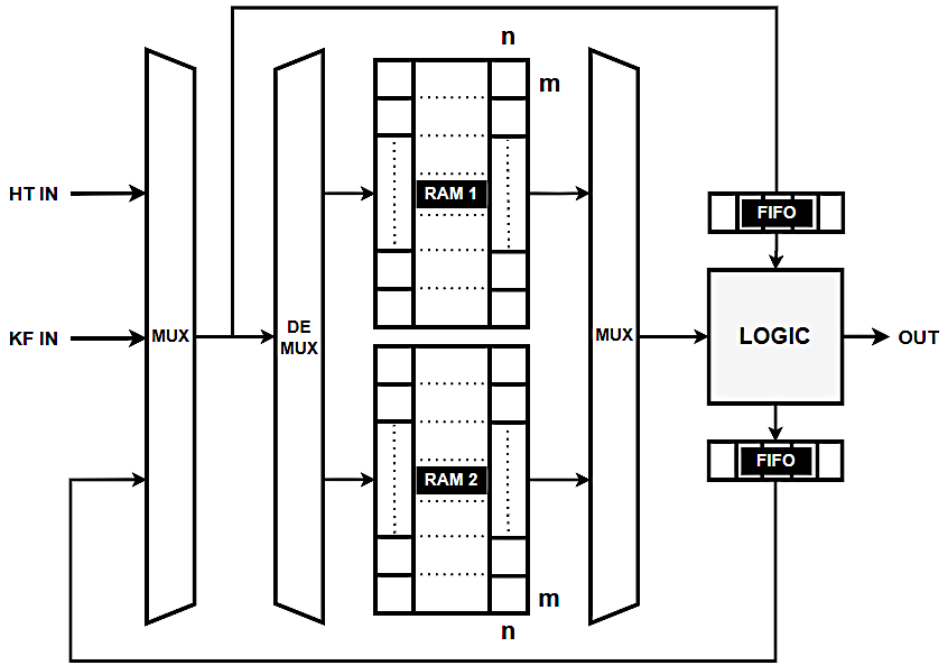

Figure 3.7: Duplicate Removal architecture.

This simple algorithm is faster than comparing all tracks in exhaustive iterations after the KF has processed all tracks; however, a second pass may be required to minimise the incorrect selection of duplicates. The second pass uses the selected duplicates from the first pass to ensure they are not selected incorrectly.

3.12 Hybrid Tracking Approach

The accumulative all-FPGA-based latencies for all modules in the TMTT must comply with specified latency requirements. Any additional module to the system can increase the latency and contribute to the costs associated with the hardware resources. The development of the DR module has led to a decrease in the number of duplicate tracks, however, the overall latency of the TMTT system with HT as the track finder and the KF as the track fitter is a concern. Hence, the Hybrid tracking system is introduced to achieve the desired performance within the latency constraint. In the Hybrid system, the

HT module is replaced by the Tracklet module. The Tracklet algorithm is based on the Road-Search algorithm [67] and template matching techniques at seed-generation levels, followed by the KF module to reconstruct tracks based on a series of complex arithmetic equations. The hardware for the Hybrid system is implemented on a fully FPGA-based system with a mixture of HDL programming and HLS automation algorithm. The hardware latency and resource customisation are discussed in Chapter 8.

3.12.1 Tracklet System for Kalman Filter

In the Tracklet algorithm [67], individual seeds are identified to form pairs or tracklets in adjacent layers in the Tracker barrel and endcaps. A particle trajectory is estimated to start from the interaction point where the tracklet parameters are consistent with $p_T \geq 2$ GeV/c and $|z_0| < 15$ cm for pairs to qualify for the Kalman system. The pairs are then matched to other layers based on pre-calculated residuals between projected tracklets and stubs to estimate a complete trajectory. Every matched stub is added to the track candidate to be corrected by a linearised χ^2 fit algorithm in preparation for Kalman filtering. The calculated parameters are compared to one another to remove duplicate occurrences as the algorithm may identify a track more than once. In practice, following the regional and geometric segmentation of the Tracker, there are regions near the segmented area where sectors overlap. If a track is formed in these regions, it will be detected by two track-finder processors independently. After trajectory reconstruction has taken place if two or more tracks share the same stubs, they are flagged as duplicates, and only one of them qualifies for the Kalman filtering stage. The disadvantage of the tracklet approach is the exhaustive search and the probability of encountering one stub several times in iterations. The advantage of the Tracklet system is that its latency of 3.333 μs is close to the specified latency of 3 μs. Further investigation in HLS implementation can help to reduce the latency further.

3.12.2 Kalman System for Tracking

The Kalman modules containing the KF firmware are linked to the Tracklet to identify and discard invalid tracks. To meet the tracking latency requirement, the Kalman system must process streams of stubs in less than 1 μs. A system with a single iteration consumes approximately 0.3 μs of the latency allowance. The total latency depends on the number of required iterations and the quality of reconstructed tracks. More description of the Kalman system is presented in Chapter 5.

3.13 Proposed Tracking Approach

In this thesis, an alternative approach for trajectory estimation is proposed. The approach is tested in the Level-1 Track-Trigger firmware, the TMTT and the Hybrid architecture. The motivation for the new system is to verify the feasibility of a simpler algorithm to replace the TMTT and Hybrid in software and hardware. The Kalman system's limitations beyond latency and resource usage are the constraints in the operational frequency above 240 MHz. This is the frequency at which the KF operates currently. The reason for the limitation is the complex arithmetic operations involving matrix calculations in recursive iterations. Another constraint in the Kalman system is the algorithm limitation in considering different parametrisations when encoding particle hits in barrel and endcap regions separately. This can lead to high efficiency in the r-ϕ plane and low efficiency in the r-z plane. Selective modifications to the Kalman system have resulted in additional latency and hardware resource utilisation. Further discussions can be found elsewhere [66]. The proposed time-multiplexed Multivariate Linear Fitter (MVLF) system overcomes the KF low-throughput by increasing the operating frequency up to 400 MHz in a fully pipelined architecture based on a simple linearised algorithm. The MVLF considers the layer misalignment in the Tracker by introducing different offsets for PS and 2S modules in both r-ϕ and r-z planes. The latency and resource utilisation of the Linear Fitter is low compared to the KF based on simpler fit calculations and deployment of Digital Signal Processing techniques in hardware implementation. The Linear Fitter algorithm and its architecture are described in Chapter 6 and 7.

3.13.1 Multivariate Linear Fitter in TMTT System

The Time-Multiplexed Multivariate Linear Fitter was initially proposed to replace KF in the TMTT system. The KF module requires additional modules such as HT and mini-HT [4] to pre-process the data. The reason for this is to reduce the data volume generated by the p_T modules transmitted through the CMS front-end electronics. These modules including KF are replaced with the MVLF module and the performance is studied. During the development of the MVLF, the module was integrated into the TMTT system. It was observed that the module can process streams of stubs at high clock frequency with the tendency to overfilter tracks. Overfiltering occurs when the system outputs a low number of reconstructed tracks. The nominal number of tracks is determined by comparing the system under development to other current and past systems. Further investigation revealed that the Linear Fitter can be modified to match the KF in performance if the boundaries of the stub sectors are smaller. More description is provided in Chapter 6.

3.13.2 Multivariate Linear Fitter in Hybrid System

The Tracklet system is used in the proposed MVLF firmware chain instead of TMTT for its efficiency in producing a high number of valid stubs in 200 pileup events. The Linear Fitter system tends to eliminate a higher number of tracks in comparison to the Kalman system. The reason for this is the techniques in the generation of virtual stubs 6.5 on Tracker layers. The Linear Fitter defines virtual stubs within stub sectors by calculating the average of stub populations for minimum and maximum parameters from the edge of stub sectors boundaries.

In the TMTT, the Geometric Processor divides the Tracker sections into 36 (18η x 2ϕ) stub sectors. The segmentation in the Tracklet divides the Tracker octant into 48 (24η x 2ϕ) stub sectors. Defining more sectors decreases the edge-to-edge stub sector boundaries and reduces the probability of discarding tracks. More details and corresponding simulations are presented in the Linear Fitter System, Chapter 6.

[4] HT resolution comes from a rough 32x18 HT array which is increased in mini-HT to 64x36.

3.13.3 Multivariate Linear Fitter System

The Linear Fitter system follows the Multivariate Linear Regression approach [68]. The algorithm based on the MVLR determines the relation between stubs to generate an estimation based on the minimisation of the sum-of-squared residuals between the measured and predicted stub positions. The algorithm expects an almost straight line in r-ϕ and another straight line in the r-z plane for particles with $p_T \geq 2$ GeV/c. The minimisation is achieved analytically by Least-Squares Regression [69]. The approach has never been attempted during the data-taking period of the Level-1 Track-Trigger firmware system. The Linear Fitter architecture allows scalability and configurability, making it a suitable addition to the Level-1 Tracking system. The Linear Fitter defines its geometric boundaries in two r-ϕ and r-z planes by hand-coding the ϕ and z boundaries into the algorithm before estimating stub occurrences in the Tracker sections.

The algorithm is designed to perform efficiently in both planes without the need for additional helper modules in stub routing and stub processing. In parallel operations, the algorithm creates a table of layer populations from a stream of stubs to identify hit occurrences on different layers of the Tracker. The number of stubs and layers are updated according to their corresponding p_T modules for all variations of Pixel- and Strip-Strip modules, independently. In instances where more than one stub is detected per layer, the generation of virtual stub samples hit coordinates based on the mean and variance of the stubs probability distribution. The algorithm continues by constructing a line or a curve that represents the best fit to all virtual stubs on layers of the Tracker. A fit is the best estimation for a straight or curved line through a two-dimensional data set provided the sum-of-squared distances between line and data are minimised. The fit becomes the initial trajectory of a particle using the stubs on every layer of the Tracker.

The Linear Fitter algorithm uses the fit parameters as baseline metrics to calculate the residuals of all stubs to the calculated fit. The algorithm identifies the stub with the largest residual[5] from the estimated fit and flags it for removal, declaring that the stub is too far from the estimated trajectory;

[5]Residual is the difference between the actual and estimated value from the fitted trajectory.

thus, it can not belong to the trajectory. Following the removal, the algorithm validates if the minimum number of stubs for constructing a valid track is achieved by updating the layer population and stub counts. In the current specification of the Linear Fitter algorithm, a minimum of four stubs on a minimum of four layers are required for reconstructing a good particle trajectory. If the requirements are not met, a second iteration will remove the next stub with the largest residual to the estimated trajectory until the number of stubs and layers match the specifications. The algorithm evaluates the exit condition which depends on the minimum number of stubs and layers based on variations of all sensor modules in barrel and endcaps.

The goodness of the fit or χ^2 establishes how well the fitted track matches the best estimation of the trajectory. The calculation uses the remaining stubs to generate the χ^2 values. For this reason, the Linear Fitter system stores the stubs in buffers for final analyses. When the buffers are full, the stubs are transmitted to estimate the trajectory and determine which stubs contributed to the final track reconstruction. The information is used for the final evaluation to differentiate between *genuine* and *fake* stubs. The definition of a genuine stub in Linear Fitter is the stub that is closest to the estimated fit or the last four stubs contributing to the track reconstruction. Any stub that does not belong to the reconstructed track is fake.

The fit calculations are performed in two planes to reduce arithmetic complexity and increase parallelism. Each plane in the algorithm is processed without data dependency on the other plane, hence making the pipelining possible. The hardware prototype of the Linear Fitter is implemented in all-FPGA technology and EMP hardware framework [70] for Level-1 Track-Trigger system in the demonstrator Section 8.4. The Linear Fitter Chapter 7 describes the algorithm and hardware components in more detail.

3.14 Correlator Trigger

The Correlator Trigger (CT) [5] uses the data from the Level-1 Tracking system to reconstruct events by identifying the primary vertices of generated tracks. The Correlator outputs a list of physics objects such as photons, electrons, muons, hadronic taus, and jets programmed into the system in the form of compact physics classes known as trigger objects. The objects are reconstructed with the highest possible measurable efficiency and purity over the p_T and η range available. The Correlator firmware is implemented on commercially available FPGA boards with custom subsystem programming. The implementation and performance of the Correlator are independent of the Level-1 tracking system development.

This page was intentionally left blank.

Chapter 4

Level-1 Trigger Data Analysis

Since the start of the High Luminosity Upgrade of the CMS detector, the sub-detectors have been in the switch-on mode for periods in-between the scheduled shutdowns. During the CMS operation, data-taking continues typically for months without a pause. The Level-1 Track-Trigger responsibility is to reduce the raw data efficiently to a fraction of its total volume without compromising the integrity of its contents. This chapter focuses on the data for the Level-1 Track-Trigger experiment and data analysis tools for performance validation.

4.1 CMS Offline Data

Offline data is the data that is collected during the data-taking period that is processed and made available for physics analysis.

4.1.1 Locating Data Samples

The collision data and Monte Carlo (MC) samples are located in the Data Aggregation System (DAS) [71] and can be accessed and cached through data services on demand. The DAS-QL [72] is a user-friendly text-based interface with the capability to chain filters or an aggregator function for queries by the data category. A user with a valid certificate can access data through the DAS web page or command-line interface for DAS Client to transfer multiple event files to a desktop or portable device [73]. Event data is distributed throughout local sites by Rucio [74] replacing the PhEDEx [75] service

accessible with both web interface and command-line options. CMS provides data description guidelines for users to identify, verify and use standard production for further analysis. The samples are characterised by their average number of interactions per bunch-crossings (e.g., 0PU, 140PU, 200PU). The events are also categorised into Top Quark pairs ($t\bar{t}$) and single Muons (μ^-), which are used regularly in Level-1 Track-Trigger simulation and Track-Quality analysis [76].

4.1.2 Data Analysis in CMSSW Framework

All the activities concerning simulation, calibration, alignment and reconstruction in the CMS experiment are primarily developed in the CMSSW framework [60]. The CMSSW utility composition is ideal for creating and filling histograms to analyse data. CMSSW provides interactive access to control the sample size by adding and removing information. The CMS software collection encompasses object-oriented programming *C++* and *Python*, Extendable Markup Language (XML), and sample data stored as text files. The CMSSW is primarily used in the software development of the Level-1 Track-Trigger modules and the evaluation of generated data concerning physics performance.

4.2 CMS Online Data

Online data refers to real-time trigger operation when CMS sub-detectors continuously take data and transmit them to the Trigger modules. The analysis is crucial in the early stages of data processing to validate detector readings. Overall, system data integrity relies on individual components from front-end electronics to back-end modules.

4.2.1 Data Analysis in EMP Framework

The infrastructural EMP framework has been developed to control and monitor the track reconstruction algorithm in the Level-1 Track-Trigger experiment. The project uses custom-built hardware-based electronics for various FPGA chips. The modules designed by developers are instantiated as entities inside the Top-Level block individually or as a chain for testing and

simulation. The data samples are fed to the EMP as payloads with pre-configured detector parameters such as LHC clock frequency, LHC bunch count and data-path regions. Data monitoring is captured through TX (transmit) and RX (receive) channels to allow multi-board testing of the firmware systems. The transmissions are controlled with an IP-based protocol known as the IPBus-Builder (IPBB) [77] in the form of data packets. An IPBB packet contains a 32-bit header to define Packet ID and Packet Type in specific byte-ordering (little-endian). The EMP framework generates data in digital waveforms for analysing and validating the performance of hardware modules.

4.2.2 Continuous Integration of Tracking System

GitHub and GitLab have gained popularity in recent years for collaborative development and teamwork. The tracking system modules are placed in CERN GitLab repositories with instructions for other users in the CMS experiment. The Continuous Integration (CI) and Continuous Development (CD) [49] platforms have enabled developers to test ideas in parallel design workflow hierarchies and embed them into the system to meet the targeted design specifications. In the Level-1 Tracking system development, the design schematics, figures, and files are uploaded into GitLab repositories with the added features for creating test vectors and test benches for algorithm quality control at different stages. The process ensures that the latest release of the packages is available for checkout without faults or bugs. In the Level-1 Track-Trigger development, all processes are implemented by the CI/CD automation tools in a pipelined structure to allow additional design plugins at every stage of the development.

4.3 Firmware Tools and Techniques

In Level-1 Track-Trigger hardware development, most modules targeting FPGA devices use Xilinx technology. The Xilinx Virtex-7 and Ultra-Scale [78] platforms have been selected in the development of the tracking system for high performance within the commissioning budgets. The Vitis Unified Software platform [79] provides sets of hardware-accelerated libraries for algorithmic building blocks in custom-design FPGA chips. The platform

permits the development of architecture-aware synthesis over a shorter timescale than hand-coded Register Transfer Level design. The tool also provides a comprehensive report analysis for Quality-of-Result (QoR) in the verification stages of the project. The Xilinx acceleration platform is used in all areas of tracking system hardware developments. However, a less conventional Electronic design automation (EDA) tool, MATLAB [80] is used to investigate and verify the vector-based arithmetic operations. MATLAB is a matrix-based programming platform that allows fast exploration of multiple approaches through a mathematical algorithm or model-based design. A feature of MATLAB, Simulink [81], is used in Linear Fitter development to generate a design hierarchy with defined interfaces and peripherals to link individual IP cores generated in the Xilinx platform in parallel simulations for the verification of mathematical correctness.

4.3.1 Field Programmable Gate Arrays

In digital electronics, the smallest programmable components are semiconductors that, in their basic form, are either powered on (high) or powered off (low). Different combinations of semiconductors in a circuit produce logic gates (NOR, AND, OR), providing basic programmability and decision-making levels. A two-dimensional array of Programmable Logic Devices (PLAs) or Configurable Logic Blocks (CLBs) are interconnected via a series of programmable wiring and millions of logic gates known as FPGAs. The strength of the FPGA circuit comes from a large number of CLBs that can be programmed and reprogrammed at user discretion after manufacturing. In addition, the logic blocks can be programmed and executed simultaneously in parallel, provided there is no data dependency. The primary constraint in creating a data processor that executes instructions in parallel is the contest to access shared variables by the instructions simultaneously. The data dependency is removed by providing the shared resources to the functions at different times at runtime [81].

FPGA devices have improved data processing applications by pipelining the workload and parallelising the operations. FPGA technology has gradually replaced ASIC in fast-evolving experiments, such as back-end electronics and DAQ developments. FPGA circuits are programmed by the

developer in HDL to describe the desired behaviour of the circuits at a low-level abstraction. The process is faster than drawing circuit schematics, and it is flexible for redefining or modifying the structure in large systems. The behaviour of a design is then mapped onto an FPGA chip through place and route, structural analysis and optimisation processes. In recent years, most FPGA devices used in CERN experiments have been designed by Xilinx. The key features of Xilinx devices are DPS blocks, Look Up Tables (LUTs) and Flip Flops (FFs), and Block RAMs (BRAMs), which provide basic operations for arithmetic and Boolean functions synchronous to the generated clock. Each chip is integrated with a limited number of memory blocks categorised by their port numbers and access types for data storage functionalities. Further resources and documentation are available within the different families of Xilinx products.

4.3.2 High-Level Synthesis Hardware Acceleration

Algorithm development and design specification have evolved in size and complexity in recent years. The developers must complete design cycles with the same resource in a shorter time, often moving to the next project or an upgrade phase. The standard computing platforms no longer satisfy the fast-evolving workload, and the need for new tools to improve productivity and efficiency is becoming more evident. Hardware acceleration platforms such as High-Level Synthesis are being used to accelerate RTL debugging and signoff in a shorter time by bringing design and verification teams together. For the High-Luminosity upgrade, the traditional RTL design flow has given way to HLS design flow to decrease verification time and increase design exploration. The development stages of a design life cycle are no longer sequential. The algorithm architecture, RTL test and verification are designed simultaneously with adaptation to last-minute changes in a matter of days rather than months. The entire design is easily migrated between technologies based on the exploration of power performance, resource usage, and latency constraints. In the Track-Trigger projects, the desired functionalities are designed in pure *C++* language. With the help of HLS libraries and built-in support, the algorithm can be retargeted for new designs with minor modifications. It is worth noting that HLS does not translate *C++* code into suitable hardware code. The developer must have

hardware design skills to achieve an optimal RTL and good QoR. The HLS tools aim to empower designers with a faster approach, particularly in the verification and debug stages. One of the advantages of HLS is comparing C-level algorithms to hardware models to verify and solve functionality inconsistencies. The RTL/C verification can be applied to different FPGA devices and ASIC design nodes during the development of the system. The automated technical reports or solutions are generated for each node and are accessible for comparisons. Another advantage is the rapid simulation of proper hardware behaviour by modelling arithmetic precision by measuring and observing the precision loss and overflow effects due to quantisation at runtime. The capability is used to model different Tracker layouts in the Level-1 Tracking algorithm to calculate the goodness of fit by introducing more or fewer bits in the process. More advantages in hardware acceleration through HLS automation, such as loop optimisations, scheduling, and resource sharing, are discussed in Section 7.2.

4.3.3 Automation Firmware Platform

The Xilinx Vivado development kit aims to provide all-in-one required platforms for developing high-level synthesis and hardware acceleration, including compilers, analysers and debuggers. Level-1 Track-Finder firmware chain was initially programmed in HDL and Maxeler technologies [82] targeting the Xilinx Virtex-7 FPGA chip family. In recent years, the Level-1 Trigger development has migrated to Xilinx Vivado and currently Xilinx Vitis design environment targeting Virtex UltraScale+ FPGA chips. The migration to new technologies was necessary as the algorithm grew in complexity, leading to more complicated FPGA-based Intellectual Property (IP) core designs. The approach has provided several advantages in developing tracking systems, such as breaking the gap between software and hardware performance, increasing architectural exploration and creating efficient firmware. Several steps are necessary to generate an IP core for the Tracker, which begins with the cross-conversion between the algorithm written in the CMSSW framework and the synthesisable algorithm in HDL for the hardware framework. The sectionrefchap7:hls describes the life cycle of conversions from CMSSW to Hardware and Hardware back to CMSSW.

All IP cores must be placed physically in the Tracker system firmware chain. Each IP contains many interfaces with the signals necessary for hand-shaking protocols and communications between modules. As the number of signals increases, the design complexity increases too. The Vitis HLS tool provides several features in the interconnection of Level-1 Tracking modules with freedom of choice between several communication protocols. The Vitis Platform is also used in the simulation, and verification stages of the development using Vitis accelerated kernel capability and application acceleration flow with minimal code changes for the migration. The optimisation is implemented through Vitis accelerated libraries to assess standard functionalities such as math, statistics and linear algebra. The interface applications support various connection protocols in the design and verification stages to help developers focus on the unit under development without restricting access and cross-talk communications between inner modules.

4.3.4 Numeric Computing Platform

In the Tracking System project, there have been instances where the results generated from complex mathematical operations could not be verified by observing the system signals as waveforms. Engineering the modules in numeric computing platforms such as MathWorks and Xilinx System Generator for DSP [83] have provided the capability to study the digital signal processing aspects of individual modules, particularly the impact of modelling the Fixed-Point algorithm replacing Floating-Point native variables in the design. The computational intensive modules in the tracking system are verified for the quantisation effects, overflows, and precision loss for numerical accuracy. Different quantisation modellings such as truncation, rounding and convergence with minimum number-of-bit allocation have been explored to develop optimised Kalman Filter and Linear Fitter modules using the MATLAB fixed-point designer [84]. The modules are eventually optimised by an HDL designer [85] to explore alternative hardware structures in evaluating Xilinx Vitis FPGA IP cores achieving full pipelining within specified latency and resources specifications. The System Generator for DSP provides a Model-Based design environment for FPGA module verifications. The custom-designed fixed-point Linear Fitter IP cores have been integrated into

Simulink side-by-side by full precision models created in MATLAB to compare and validate the outputs. The System Generator assists in rapidly evaluating new algorithms, design prototyping, and model analysis in the Level-1 Tracking system. The tool is used to flesh out the algorithm deficiency and investigate the impact of the modifications on hardware. The values in the bit- and cycle-accurate blocks produced in Simulink are compared to their corresponding representations designed by the HLS tool.

This page was intentionally left blank.

Chapter 5

Kalman System for Tracking

The Kalman algorithm is an essential feature of the Tracking system in Level-1 Track Trigger. Although the concept of trajectory optimisation using the Kalman algorithm is not new in the CMS experiment, utilising the Kalman algorithm in online track reconstruction has never been attempted before. The Kalman algorithm estimates particle motion with the help of current and previous measurements in error-correcting feedback loops without knowing how many particle hits make a complete trajectory. The Kalman system is particularly efficient in the joint probability distribution of particle coordinates in the spatial plane and their specified uncertainties related to variations in the parameters. In the HL-LHC upgrade, the KF is deployed in the tracking system's chain after the HT module to validate the output of the Hough trajectory extraction algorithm. This chapter examines the building blocks of the Kalman system in the context of the Level-1 Track-Trigger hierarchy.

5.1 Motivations for using Kalman Filtering

High luminosity upgrade will increase the number of collisions and consequently increasing the data volume significantly. The Level-1 Triggering mechanism is required to extract and deliver the events containing interesting physics to the DAQ system. The p_T modules were introduced to remove unwanted low momentum particles as the first instance of filtering. At this stage, the data volume is still significant and further filtering based on limited tracking information is required. The FPGA-based Associative Memory (AM)

[86] technique was a possible approach that was rejected at the early stages of its development. The AM used pattern recognition and template matching algorithm. In pattern recognition, the algorithm aimed to provide all possible inputs as templates to generate a matching pair through a series of regional searches and exact-matching procedures. The AM approach was practical when all existing patterns were previously identified and converted to templates before runtime. In this approach, the tracker is divided into η, ϕ regions known as Trigger Towers. Data aggregated from the tracker front-end electronics are inputted to towers implemented on FPGA chips. The patterns were extracted and matched with the pre-made templates. If no match was found, the algorithm concluded that the event was not interesting. Approximately one million templates are required per Trigger Tower. In a full-resolution system, 48 (6η x 8ϕ) Trigger-Towers with additional logic were responsible for organising and storing approximately 48 million templates that led to architecturally complex design and costly development.

In the HL-LHC upgrade, the priority is discovering new physics in the newly upgraded detectors. The Kalman algorithm is exceptionally effective in estimating new trajectories with no predetermined knowledge of the paths of the particles with no requirement for storing templates. The hardware implementation of the Kalman filter has been evolving to meet the timing constraints under the Level-1 Track-Finder latency specifications [58]. An advantage of the Kalman system is the possibility of a fully Time-Multiplexed (TM) design. A fully functional model requires fewer nodes to demonstrate the architecture based on the generation of paralleled independent sub-modules, which are easily scalable and reconfigurable. In Chapter 8, an implementation of the Kalman algorithm with only two nonants has been possible. The resource estimates are identical across all paralleled nonants.

5.2 Conventional Kalman Estimator

Before designing the optimisation of the Kalman filter for particle trajectory estimation, a generic version of the algorithm is discussed in this section. As explained in Chapter 3, the Kalman algorithm is a type of state observer that combines measurements and predictions to calculate an estimation [87].

The Kalman algorithm is an iterative process, which is particularly useful when designing a fully pipelined algorithm in the hardware. Other benefits of using this algorithm include:

- The calculation process can begin as soon as the first measurement becomes available in a series of time-sensitive events.
- The algorithm allows placing importance on the measurements or predictions based on Kalman gain.
- The algorithm can be applied to complex problems with high precision in arithmetic operations and mathematical calculations.
- In hardware design, the algorithm can be divided into data flow and calculation blocks in pipelined configuration to reduce design complexity and latency.

Often in motion estimation, there are errors associated with the measurements E_{MEA} and errors associated with the estimates E_{EST} that are taken into account in the calculations. The Kalman algorithm accounts for uncertainties and errors using the following mathematical processes. The Kalman Gain KG gives a measure of how much of the prediction comes from the measurements and how much comes from the estimations by determining the error-in-measurements and error-in-estimations. The Equation 5.1 defines this relationship:

$$\boldsymbol{KG} = \frac{\boldsymbol{E_{EST}}}{\boldsymbol{E_{EST}} - \boldsymbol{E_{MEA}}} \tag{5.1}$$

The Kalman algorithm functions with two-time stamps. The current time defined as t and previous time as $t-1$. The current estimate $\boldsymbol{EST_t}$ is the result of taking into account the previous estimate $\boldsymbol{EST_{t-1}}$, $\boldsymbol{KG}$ and current measurement $\boldsymbol{MEA_t}$ defined in Equation 5.2:

$$\boldsymbol{EST_t} = \boldsymbol{EST_{t-1}} + \boldsymbol{KG}(\boldsymbol{MEA_t} - \boldsymbol{EST_{t-1}}) \tag{5.2}$$

The current error-in-estimate $\boldsymbol{E_{EST_t}}$ is calculated from $\boldsymbol{KG}$ and previous error-in-estimate $\boldsymbol{E_{EST_{t-1}}}$ defined in Equation 5.3:

$$\boldsymbol{E_{EST_t}} = (1 - \boldsymbol{KG})(\boldsymbol{E_{EST_{t-1}}}) \tag{5.3}$$

The procedures in terms of E_{MEA} and E_{EST} are shown in Figure 5.1. In each algorithm iteration, the calculated data are fed back to the system and immediately become available as the previous estimate for the current set of measurements.

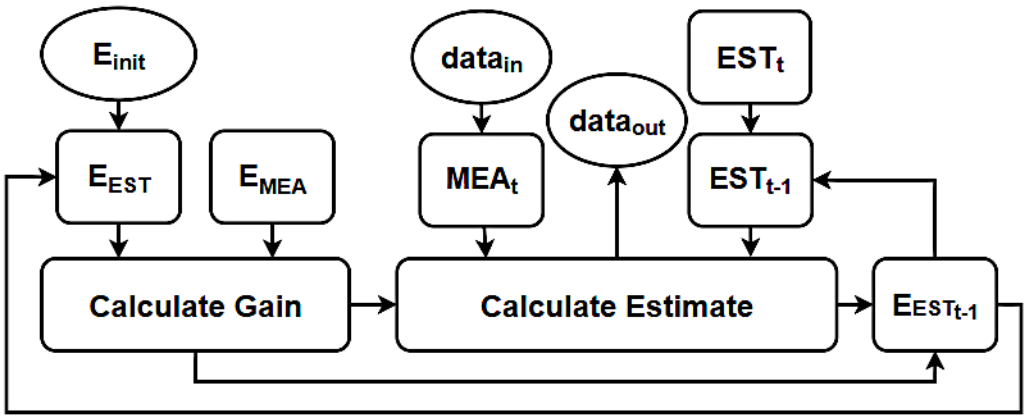

Figure 5.1: Kalman algorithm flow for gain and estimate.

The Kalman equation 5.4 is written as position X_t, previous position AX_{t-1}, and current measurements Bu_t with added noise W_t. The parameters A and B are helper matrices required to perform arithmetic operations as matrix calculations [88]:

$$\boldsymbol{X_t = AX_{t-1} + Bu_t + W_t} \tag{5.4}$$

The measurement equation can be either scalar or a matrix depending on the elements in the observation and is defined as:

$$\boldsymbol{Y_t = CX_t + Z_t} \tag{5.5}$$

Where Y_t and X_t are the measurements, C is the helper matrix and Z_t is the additive noise. Equations 5.2, 5.4 and 5.5 can be combined to give:

$$\boldsymbol{X_t = AX_{t-1} + Bu_t + K(Y_t - C(AX_{t-1} + Bu_t))} \tag{5.6}$$

The previous state estimate is denoted as X_{t-1}, while the predicted state estimate is defined as X_t and the measurement as Y_t. What differentiates X_t and X_{t-1} is the timing interval t when measurements are taken into account from one estimation to the next. The current error covariance P_t is incorporated into the estimates by using:

$$\boldsymbol{P_t = AP_{t-1}A^T + Q_t} \tag{5.7}$$

The Kalman Gain called K henceforth with helper matrix H is calculated by adding the measurement covariance R_t by considering two different scenarios: when $\lim_{\boldsymbol{P_t}\to 0}$ or $\lim_{\boldsymbol{R}\to 0}$ in giving:

$$\boldsymbol{K = \frac{P_t H}{H P_t H^T + R_t}} \tag{5.8}$$

By adding K to Equation 5.9, a new value for the error covariance P_t is obtained. Kalman Gain and measurements produce new parameters for the next iteration:

$$\boldsymbol{P_t = (I - KH)P_t} \tag{5.9}$$

The algorithm uses the estimated state X_{t-1}, error covariance P_{t-1}, and current measurement Y_t, recursively to generate a new prediction in the next time interval. Adding new variables to the Kalman system, such as noise, can increase the complexity of the equations and complicate the design process when considering hardware implementation.

5.3 Kalman Filter for Track Finding

In the HT stage of the Tracker system, the tracking candidates are roughly estimated and presented to the Kalman module as input. A track candidate from the HT usually contains several stubs that are genuine and belong to a single track, however, some stubs do not belong to the track and never originate from the HT-initiated search. In 200 pileup events, almost 50% of the track candidates contain stubs from other tracks or combinations of stubs that do not correspond to a real track [28]. The Kalman filter aims to remove these so-called fake stubs to improve efficiency before the fitting process begins. In

the Time-Multiplexed Track-Trigger, the Kalman algorithm calculates a Kalman state based on the HT stub parameters and the uncertainties associated with the precise location of the hits versus their predicted positions. The process involves pre-estimating track parameters with the uncertainties of individual stubs in a successive step-wise manner, replacing the time dependence with the tracker radius dependence r. The algorithm begins with the stubs closest to the interaction point and then moves layer-to-layer according to where the stubs occur. A robust Kalman filter must be capable of filtering out fake stubs by discarding them and storing genuine stubs for final track reconstruction at the desired efficiency. The Kalman algorithm proposed for stub filtering has some deviations from the conventional Kalman filtering model; however, efforts have been made to keep the alteration to a minimum. The changes are explicitly made in the hardware implementation to help the smooth flow of data and pipeline. The Kalman algorithm initiates the track finding process with the following measurement parameters:

$$\boldsymbol{x} = (\frac{1}{2\boldsymbol{R}},\ \boldsymbol{\phi_0},\ \boldsymbol{cot\theta},\ \boldsymbol{z_0}), \quad \text{where,} \quad \boldsymbol{R} = \frac{\boldsymbol{p_T}}{\boldsymbol{qBc}} \tag{5.10}$$

The parameter ϕ_0 and z_0 denote the azimuthal angle and impact parameter along the z-axis concerning the stub sector position, respectively. The θ coordinate is the polar angle between the track and z-axis. The R parameter defines the distance from the interaction point and corresponding sensor module positions on the layers of the detector. The particle charge is denoted as q, the magnetic field strength as B and the speed of light as c. When a track begins its trajectory with r $\approx$ 0 and transverse through the first layer of the tracker, its global coordinates ϕ and z are mapped to their local coordinates ϕ_0 and z_0 using Equations 5.11 and 5.12, respectively:

$$\boldsymbol{\phi_0} = \frac{\boldsymbol{r}}{2\boldsymbol{R}} - \boldsymbol{\phi} \tag{5.11}$$

$$\boldsymbol{z_0} = \boldsymbol{r cot\theta} - \boldsymbol{z} \tag{5.12}$$

Once these coordinates are defined, the variables must be mapped to their corresponding representations in the Kalman Filter equation. Several discussions concerning trajectory reconstruction concerning the offline

Kalman Filter algorithm can be found here [89]–[91]. The Kalman algorithm uses matrix-based calculations for its stub and state estimations. The helix parameter H_t at radius r is defined by hit position m in (ϕ, z) and hit position covariance matrix x as:

$$\boldsymbol{H_t} = \frac{\boldsymbol{\delta m}}{\boldsymbol{\delta x}} = \begin{bmatrix} -\boldsymbol{r} & 1 & 0 & 0 \\ 0 & 0 & \boldsymbol{r} & 1 \end{bmatrix} \tag{5.13}$$

If the effects of multiple scattering and white noise are negligible (Q_t = 0) and the positions of the particle hits do not change, $\boldsymbol{P_t}$ is defined by Equation 5.14, if (1/2R, ϕ_0) and (cot$\boldsymbol{\theta}$, z_0) are independent linear fits in two r-ϕ and r-z planes:

$$\boldsymbol{xP_t} = (\frac{1}{2\boldsymbol{R}}, \phi_0, \boldsymbol{cot\theta},\ \boldsymbol{z}_0) \begin{bmatrix} \boldsymbol{\sigma_a^2} & \boldsymbol{\sigma_{ab}} & 0 & 0 \\ \boldsymbol{\sigma_{ab}} & \boldsymbol{\sigma_a^2} & 0 & 0 \\ 0 & 0 & \boldsymbol{\sigma_c^2} & \boldsymbol{\sigma_{cd}} \\ 0 & 0 & \boldsymbol{\sigma_{cd}} & \boldsymbol{\sigma_d^2} \end{bmatrix} \tag{5.14}$$

The uncertainties $\boldsymbol{\sigma_a}$, $\boldsymbol{\sigma_b}$, $\boldsymbol{\sigma_c}$, $\boldsymbol{\sigma_d}$, $\boldsymbol{\sigma_{ab}}$, $\boldsymbol{\sigma_{cd}}$ are the estimated locations of the $\boldsymbol{p_T}$ modules in the different tracker regions and are added to the equation. The particle hit uncertainties are initiated in the barrel and endcap regions of the tracker and represented by the parameter R, which is defined as:

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{\sigma_\phi^2} & 0 \\ 0 & \boldsymbol{\sigma_z^2} \end{bmatrix} \tag{5.15}$$

In the barrel region, the uncertainties are defined as follows:

$$\boldsymbol{\sigma_\phi^2} = \left(\frac{1}{\sqrt{12}} \frac{\boldsymbol{StripPitch}}{\boldsymbol{radius}} \right)^2 \tag{5.16}$$

$$\boldsymbol{\sigma_z^2} = \left(\frac{1.5625\ \boldsymbol{StripLength}}{\sqrt{12}} \right)^2 \tag{5.17}$$

For the particle hit in the endcaps, the uncertainties are given by:

$$\boldsymbol{\sigma}_{\phi}^{2} = \left(\frac{1}{\sqrt{12}}\frac{\boldsymbol{StripPitch}}{}\right)^{2} + \left(\frac{1.05}{2\boldsymbol{R}}\right)^{2} \tag{5.18}$$

$$\boldsymbol{\sigma}_{z}^{2} = \left(\frac{1.5625\ \boldsymbol{StripLength}}{\sqrt{12}}\right)^{2} 0.9(\boldsymbol{cot\theta})^{2} \tag{5.19}$$

The measurement and their uncertainties are formatted into a vector of stubs m, covariances of the stubs V, helix parameters x, the covariances of helix parameters C, the predicted stubs for helix parameters H, and goodness of fit χ^2, which are all related as shown in Equation 5.20. The intermediate variable S is defined to store the result of helix parameters and covariances multiplication followed by predicted uncertainties R and the Kalman gain K.

$$\begin{aligned} \boldsymbol{S} &= \boldsymbol{HC} \\ \boldsymbol{R} &= \boldsymbol{VS^{T}} \\ \boldsymbol{K} &= \boldsymbol{SR^{-1}} \end{aligned} \tag{5.20}$$

Stub hit uncertainty $\boldsymbol{Q}$ is defined as the difference between the stub measurement and the helix parameters:

$$\boldsymbol{Q} = \boldsymbol{m} - \boldsymbol{Hx} \tag{5.21}$$

In the second iteration of the algorithm, the measurements and uncertainties are fed back into the process as follows:

$$\begin{aligned} \boldsymbol{x_{updated}} &= \boldsymbol{x} + \boldsymbol{KQ} \\ \boldsymbol{C_{updated}} &= \boldsymbol{C} - \boldsymbol{KS} \end{aligned} \tag{5.22}$$

The algorithm increments the track χ^2 to account for the new stub measurements that have been incorporated. All matrices from the previous iteration are made available to the next iteration:

$$\chi^{2} \mathrel{+}= \boldsymbol{R^{-1}Q^{2}} \tag{5.23}$$

All the states generated from the initial state by adding stubs from subsequent layers are sorted by the value of χ^2 in ascending order meaning there is a high correlation between the measurements and the estimates. The first four states are retained if the tracks corresponding to the states have only one layer. If this condition is not true, the fifth state is considered. In Figure 5.2, the track-fitting process is illustrated, which is based on stub-to-stub processing and assumes the Gaussian distribution of measurements.

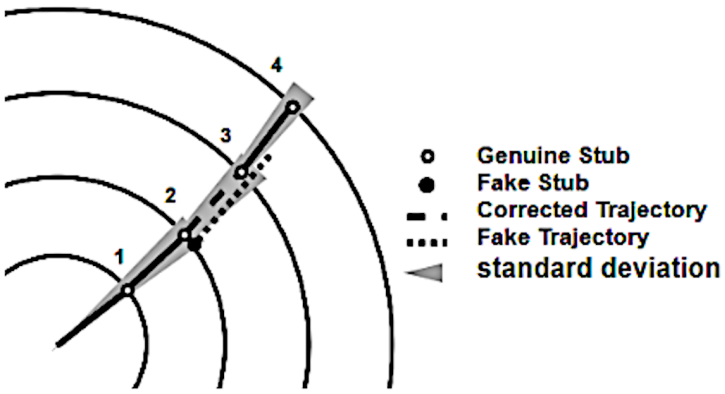

Figure 5.2: Kalman algorithm process in filtering stubs [28].

When the Kalman algorithm receives processed stubs from either HT in the TMTT system or Tracklet in the Hybrid configuration, it assumes that it contains both genuine and fake stubs. Taking a trajectory originated from (r = 0 and z = 0) and interacts with at least four p_T modules on consecutive layers, the first hit coordinate from the first layer become available to the Kalman algorithm. The shaded area surrounding the track in Figure 5.2 presents the standard deviation from the calculated estimates. The first measurement becomes available to the Kalman algorithm when the particle interacts with sensors on the first layer (L1). In this example, two hits are recorded on the second layer (L2), with one yielding a genuine stub and one fake stub. The Kalman algorithm corrects its trajectory estimation once the information on the third layer (L3) becomes available, which results in the stub being rejected. In the process, the trajectory is corrected and the associated data are not propagated to create a state. In a situation, where, only one stub is present in the next layer, if it is within the approximation of the calculated

standard deviation, the data is propagated for state creation. The process continues until all stubs are processed. The Kalman algorithm implementation is divided into two parts; Kalman Flow Control and State Updater to increase throughput. The following section describes the building blocks of Flow Control modules and data transitions between modules.

5.4 Kalman Algorithm Flow Control

The Kalman algorithm must be fully pipelined and capable of processing one stub per clock cycle [58]. The pipelining is achieved by introducing Kalman Workers, each of which consists of a state updater and the flow control blocks required to ensure the steady transmission of stubs and states via transition busses. Flow control blocks in the Kalman algorithm architecture are responsible for formatting and managing the stubs in preparation for the Kalman updater block. All complex mathematical operations are assigned to this block in a pipelined hardware structure. The Kalman worker modules are designed as plugins for the Track-Trigger firmware chain. The modules are added to the firmware following the stub formatting before the workers are integrated into the system and reformatting before linking the workers to other modules in the Level-1 Tracking chain. The process of stub formatting is implemented in input link formatters and output link formatters in HDL. The stubs are received in 64-bit binaries that each represent a set of 72 channels containing the stubs parameters in streams. Eighteen Kalman workers process stubs from two nonant $2\pi/9$ via 72 channels from the DTC blocks. The stubs are formatted into links for the Kalman worker nodes in the top-level design architecture. The 64-bit stubs are repacked into 96-bit to contain information about the states. The Kalman node block deploys link formatting again to extract stub information from each input node. It prepares the stubs for the state creator block that is responsible for extracting Kalman seeds from the previous blocks. The process applies selection criteria while constructing seeds depending on which tracker region the seeds are detected initially. The corresponding slope and intercept of a seed is calculated in Hough-space and converted to what is known as the Kalman state with appended assignments of covariances and state identifier encoded into a 96-bit binary word.

In Figure 5.3 the logic components and inter-connectors are shown with arrows presenting the data movement between modules.

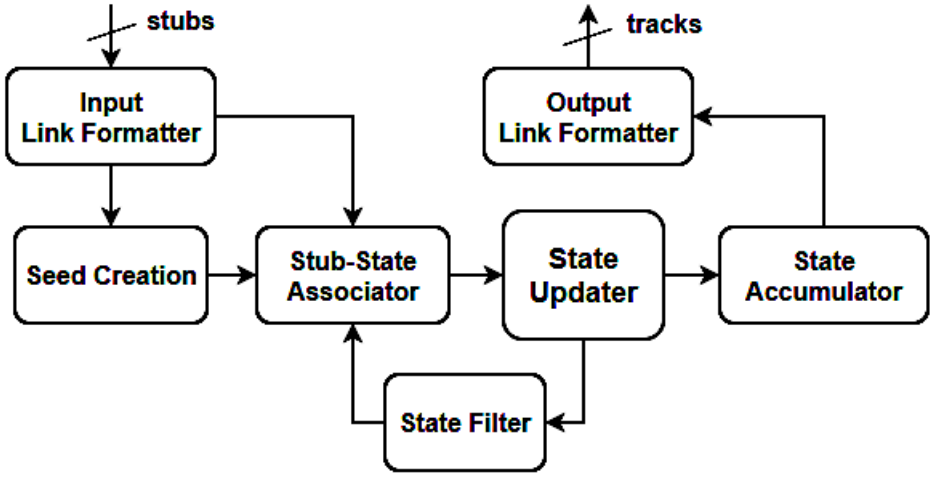

Figure 5.3: Kalman Worker algorithm functions.

The output stubs and states are passed to the next block which is known as the state filter. This block ensures the χ^2 selection requirements are applied correctly to the states to remove those that do not qualify under the selection criteria. At this stage, the selection is based on the states with only one layer from the first four states. The stubs and states are then transmitted to the stub state associator block, where associations are made between stubs and states. A stub and state create a combination that is indexed and stored in different memories according to the region the combination was located. The address book holds the stub state indexing information, always pointing to the newly generated stub state memory location. This block also returns the subsequent stub association with the input state and the new output state, increasing the next layer value.

According to the Kalman filtering arithmetic, at this stage, the stubs and states are associated and transmitted to the computationally intensive Kalman updater. In this block, the algorithm calculates the Kalman gain that is used to differentiate between estimates and hit positions. Many states will not survive this stage of the processing and will be flagged for removal. The outputted states are then transmitted to the state accumulator block for

storage. Further selections based on the number of layers and stub positions are applied in the final filter. The stubs also are formatted for the Duplicate Removal module and transmitted on the output link formatter.

In the top-level design of the Kalman algorithm, the paralleled workers are integrated into each Track Finder processor. The pipelining of workers is enforced across top-level sections by First-In-First-Out (FIFO) memory blocks. A stub received by a worker is transmitted on two 64-bit buses. The first bus is stored in memory and passed through to the stub state associator for indexing. The second bus is inputted to the seed creator block to create a new state. The stub with a unique identifier is passed to the state updater, where Kalman computations are implemented to apply selection requirements based on p_T, χ^2 and z parameters. Two states (State_t and State_{t-1}) are presented to the state control block to decide which direction the trajectory must take. If the states have more than one stub on each tracker layer, the stub state association is removed and the stub is fed back to the algorithm for the second pass. If the stubs are not found on the next layer of the tracker, it is permitted to search and use stubs from the layer after next instead. After the process, a comparison between candidate tracks reveals the number of missed layers per track and their χ^2 parameters. Then only a track with the fewest missing layers and the smallest χ^2 is retained. A complete track is an output following the conclusion of four consecutive iterations. Further iterations result in more hardware resources under the tracking system latency specifications. As a consequence, only four iterations are permitted. The number of iterations is primarily decided by comparing the tracking efficiency for the different data sets. In the Kalman filter, a stream of stubs contains four stubs, therefore four iterations are necessary.

5.5 Kalman Firmware Implementation

The implementation of the Kalman system in tracker firmware follows the hardware design strategy of top-down design and bottom-up implementation approach. The process allows individual blocks to be tested inside Kalman workers by introducing a null interface 8.9 in all blocks except the block under test. The strategy also helps with the overall design process in

scenarios where implementing a complex block has proven difficult and requires more time for completion. Following the implementation of the top-level design, as soon as a block becomes available will be plugged into the Track Finder Processor and firmware chain. The Kalman system was placed initially in the Track Fitter architecture and interfaced as a null block. The inner blocks were added gradually following stand-alone implementation and testing stages.

Implementing the Kalman algorithm in hardware is computationally demanding due to matrices and data flow control in a time-sensitive manner due to recursive iterations of the algorithm. These properties increase the complexity of the hardware surpassing the latency and resource specifications of the Level-1 Track Finder system in preliminary simulations. The solution is to divide the algorithm into smaller manageable functions that can be pipelined using added registers for temporarily storing data. In Figure 5.4, an illustration of Kalman worker data flow using an integrated memory structure is presented.

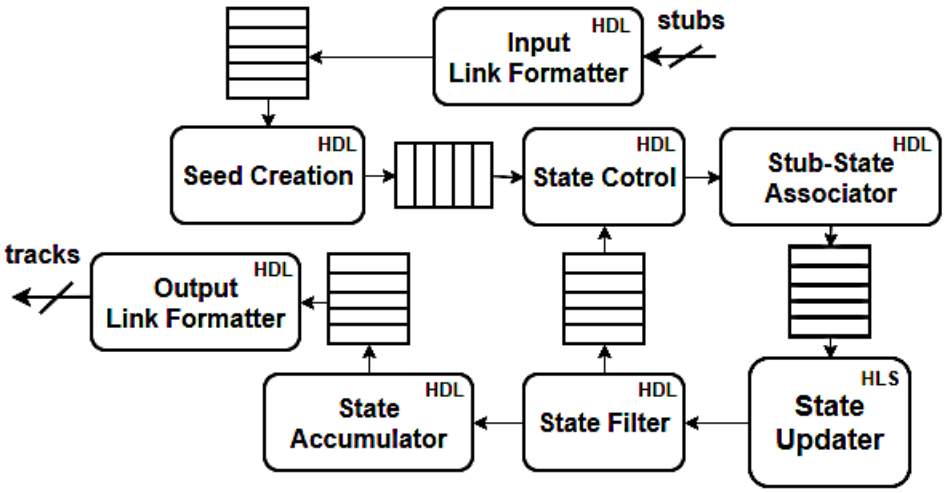

Figure 5.4: Kalman Worker data flow structure in hardware.

The modules are defined in terms of data flow to transmit the results of the matrix arithmetic operations, updating bookkeeping parameters, calculating the covariance matrices, estimating the χ^2 value, and finally controlling the flow of stubs and state association in preparation for the state

updater. Random Access Memories (RAMs) are used inside the blocks to store intermediate results ranging from 16 kb to 512 kb storage capacity. The RAMs are implemented in VHDL with one read and write capability per clock cycle. The structure is particularly useful in pipelining the data flow between blocks. Access optimisation directly improves the system's performance at the frequency of RAM's operational capability. The implementation also limits the data flow rate at a higher frequency due to read/write limitations of the RAMs.

The state updater is entirely designed using the HLS automation tool. The rest of the blocks are implemented using HDL. The method separates data flow blocks that are trivially programmable in HDL from computational blocks that are complex. Each block is separated by intermediate memory blocks controlled by the user-defined clock and reset capability, except the stub state associator, which is implemented using dual-port RAM for storing associations. The HLS acceleration with supported directives for built-in fixed-point design and pipeline scheduling has helped the overall hardware development overcome the latency and resource constraints in implementing the Kalman system in the final assembly. The primary issue in the development of the state updater is the use of the division operator while finding the determinant of the matrix used in the calculation of the track helix parameters. A custom division algorithm has been devised using lookup tables and a multiplier for faster matrix inversion. Division operations are the most costly in terms of logic resources and latency. In hardware design, the divide expression (a/b) is broken down into two expressions (a x 1/b) and implemented using custom logic and lookup tables. The diagonal matrix operations have benefited from implementing inverse operations that naturally require large RAMs for a variable with a word length exceeding 18-bits. A similar technique is adopted for the calculation of fit parameters in the Linear Fitter module. The inverse model is described in more detail in Linear Fitter Chapter 6 and Appendix A2. The implementation of the Kalman filter in the hardware has made use of several other design alternatives and techniques to meet the latency and resource usage requirements, which are discussed in Chapter 7.

5.6 Kalman Algorithm Performance

Implementing the Kalman algorithm is an ongoing process that requires further studies to determine the optimal number of Kalman workers per Track Finder processor. Also transferring the χ^2 calculations to a separate module due to the complexity of the algorithm. The physics simulations in this section are produced using both old and new implementations of the Kalman algorithm. In the new Kalman algorithm, the dependency on the maximum number of stubs (4) is removed. Instead, the algorithm can process up to six stubs resulting in Kalman performance increasing across the p_T range and a reduction in duplicate tracks. The simulations use samples containing top quark pair production ($t\bar{t}$) that also include 200 pileup events. The samples are analysed in the CMSSW framework using the firmware demonstrator. The new Kalman algorithm improves physics performance and efficiency by fine-tuning the Kalman filter variables and modifying data flow transmission in hardware. The Kalman algorithm reconstructs tracks with hits that are not generated from a real particle. The reconstruction algorithm allows incorrect stubs in the trajectory estimation if only part of the trajectory matches a genuine track.

The algorithm combines track-finding and track-fitting in identifying matched and unmatched tracks to improve performance leading to high purities and high efficiencies. With the help of the HT algorithm, the two-stage track reconstruction techniques coarsely identify particles with high p_T reconstructing many candidate tracks. The number of fake tracks at high p_T negatively impacts efficiency due to the Kalman algorithm's inability to reconstruct tracks without helical parameters. High energy particles produce a less helical trajectory through the tracker layers. The trajectory of high-energy particles is closer to a straight line with almost no bend caused by the magnetic field. Also, as most particles in the simulated events are hadrons, the deterioration of the performance can be related to the production of secondary particles. These particles are often found very close to the parent particles and can mislead the algorithm. The loss in efficiency is evident in particles with $p_T > 70$ GeV/c.

The tracking efficiency as a function of p_T and η are shown in Figures 5.5 and 5.6.

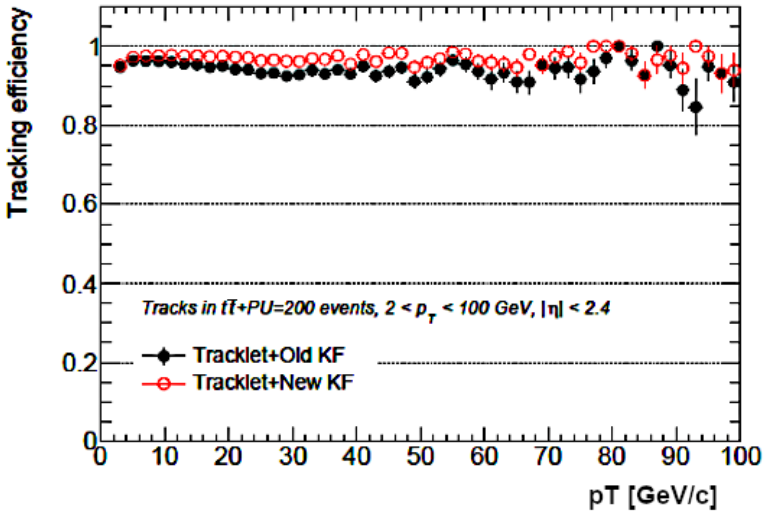

Figure 5.5: Tracking efficiency of Kalman filter as a function of p_T.

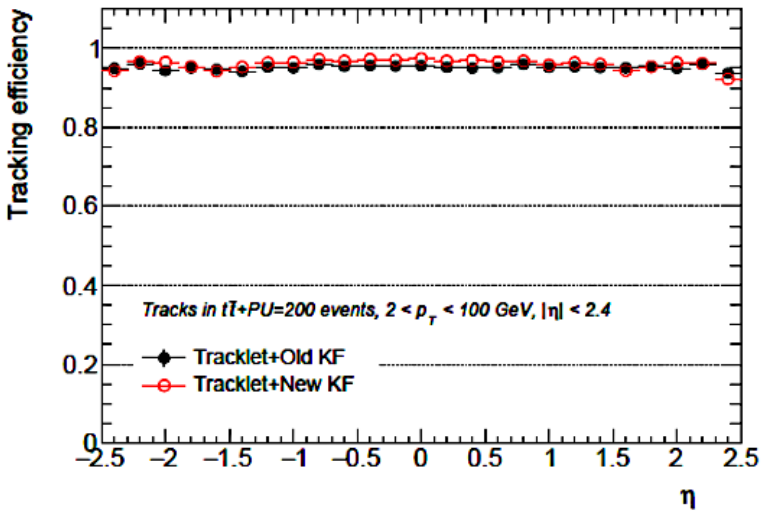

Figure 5.6: Tracking efficiency of Kalman filter as a function of η.

5.7 Resource Usage and Latency

The tracker segmentation approach allows paralleled implementation of the TFP containing Kalman workers. The regional segmentation divides the r-ϕ plane into two regions and the r-z plane into eighteen regions with a time-multiplexing factor of eighteen. In this configuration, there are eighteen Kalman workers assigned to a TFP, receiving stubs from two nonants. The number of workers is calculated by a trade-off between efficiency and resource estimations for a minimum number of workers based on the $t\bar{t}$+200PU events. This model of the tracking system is proposed for demonstration purposes to investigate the feasibility of a fully functional Kalman system, and to estimate resource usage and latency for the entire system. The results presented for latency and logic resources are identical for every TFP and nonant. In Table 5.1, the logic resources and latency for Xilinx Virtex-7 (690T) FPGA chip are shown. The acronyms that are used are Lookup Table (LUT), FIFO (FF), Block RAM (BRAM) and Digital Signal Processing slices (DSP) in logic quantity (slices) and percentage of used resources in the FPGA chip.

resources	utilisation	available	utilisation (%)	time (ns)
LUT	305,406	433,200	70.5	-
FF	687,921.6	866,400	79.4	-
BRAM	1,858.1	1,470	126.4	-
DSP	597.6	3,600	16.6	-
LATENCY	-	-	-	1,243

Table 5.1: Resource and latency for the Kalman system.

As can be seen in the Table 5.1, the number of BRAMs (1,858.1) exceeds the maximum available resources (1,470) on the Xilinx Virtex-7 FPGA chip. Currently, the memory constraints require the use of two chips. The Kalman system needs memory for storing stubs, states, and their associations. The number of DSP blocks also introduces constraints for the scalability of the design. Reducing the total number of DSP blocks is desirable and may be necessary for a complete system prototype.

The tracking efficiency, measured for both software (SW) and the hardware (HW), for samples of $t\bar{t}$+200PU events as a function of p_T and η shown in Figures 5.7 and 5.8.

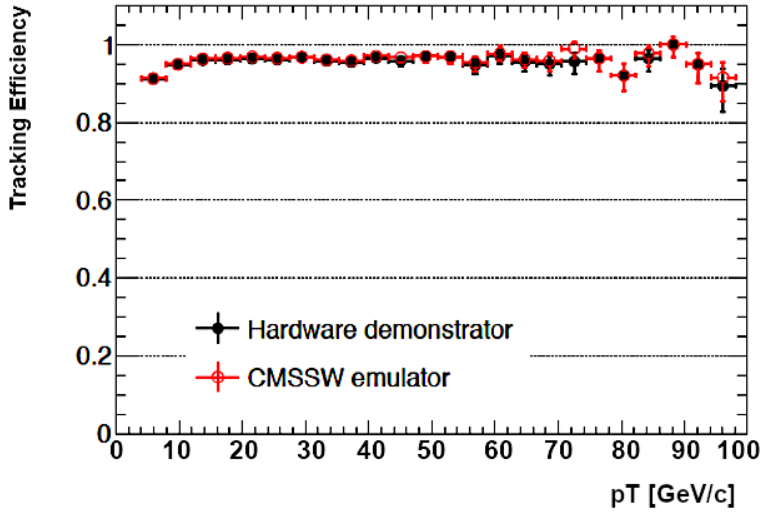

Figure 5.7: Tracking efficiency (SW+HW) as a function of p_T

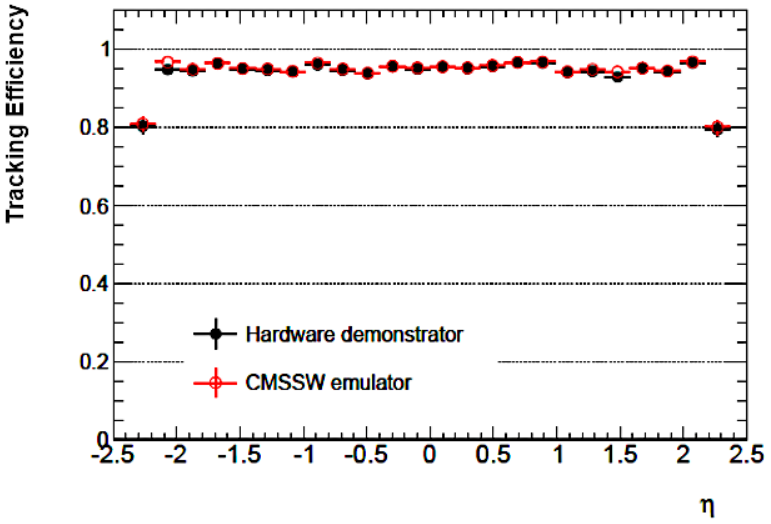

Figure 5.8: Tracking efficiency (SW+HW) as a function of η

This page was intentionally left blank.

Chapter 6

Linear Fitter for Tracking

This chapter presents an alternative software- and hardware-based approach using the Multivariate Linear Regression algorithm and Linear Track Fitting for the Level-1 Track-Trigger system. Two additional architectures, the TMTT and the Hybrid system, both using the Kalman algorithm for track reconstruction, have been developed and investigated in terms of performance, hardware resources and latency requirements. The baseline metrics defined under the Level-1 Tracking system specifications [92] are used for comparison. A Multivariate Linear Fitter system prototype is implemented in software using the CMSSW framework and in the hardware using the EMP framework, specifically designed for the Level-1 Tracking system firmware platform. Simulation and emulation results are presented for the standalone[1] version in this chapter and implementation on the broader system is discussed in the Chapter 8.

6.1 Linear Fitter Model

The Linear Fitter algorithm is based on the Multivariate Linear Regression approach, perhaps the most well-understood algorithm in statistics and estimation applications. Although the Linear-Regression-based algorithm has gained popularity in offline CMS data analysis, the algorithm has never been used for online track reconstruction and track quality estimation. The Linear Regression algorithm is simple in terms of the arithmetic operations involved. The algorithm's simplicity makes the model a good candidate for real-time

[1] In standalone development, the dependencies on other systems are manually defined.

stub processing. There are challenges for modules operating in a real-time environment, such as constraints in using logic resources and latency imposed by the Level-1 Track-Trigger system, which are discussed in this chapter. In the classic Multivariate Linear Regression approach [93]–[96], the model predicts a relationship between the independent (predictor) and dependent (outcome) variables through correlation analysis within the generated data-set. In a two-variate Linear Regression, the process begins with placing the data (blue points) in a two-dimensional plane with independent and dependent axes, as shown in Figure 6.1. A horizontal line with gradient zero passing through the centre of the data points is drawn, such as the black dotted line shown in the top plot. The centre is determined by finding the mean of independent and dependent variables. The distances from the data points to the line are measured and squared to avoid negative values. The summation is known as the Sum of Squared Distances (SSD). The calculation results in an SSD value with its corresponding gradient plotted in the bottom plot.

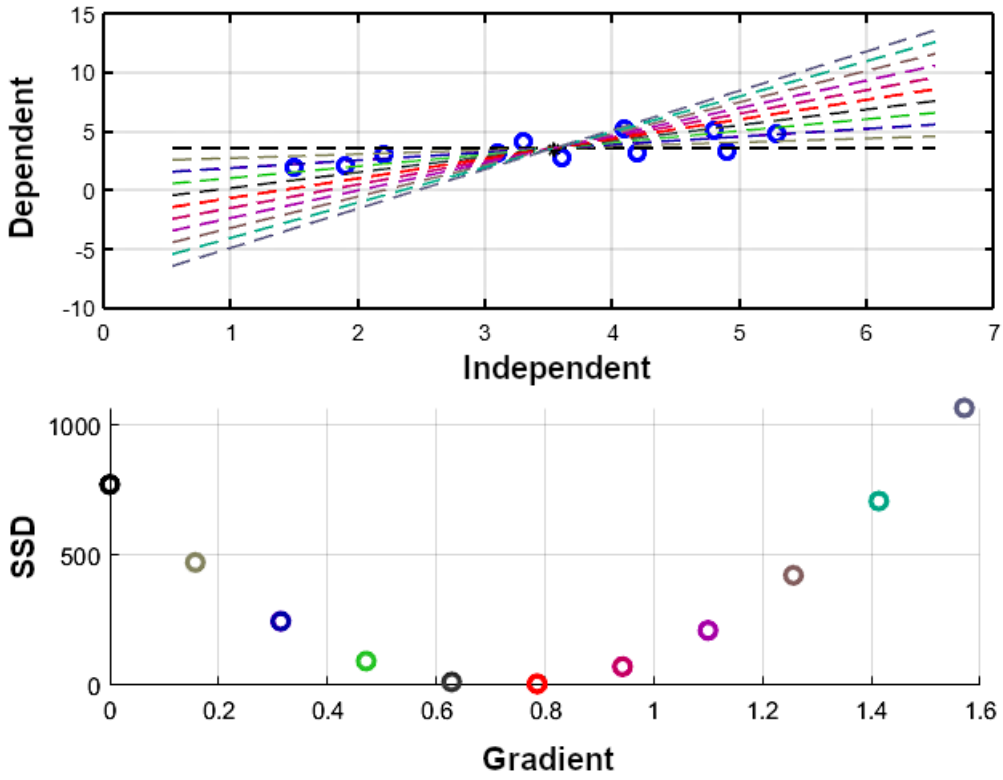

Figure 6.1: Fitting a line to data with the least SSD.

The process is repeated by rotating the line by a small degree in one plot, calculating the SSD and plotting both values in the other plot. Repeating the operation several times generates two new sets of values; gradients and SSDs. The line with the least value for SSD defines the best fit. This line minimises the distance between the estimated line and data points, such as the red line in the top plot and its corresponding red point in the bottom plot. While the value of the SSD approaches zero the estimated line becomes closer to a good fit for the data.

The Goodness-of-Fit χ^2 determines if observed data (O) aligns with estimated data (E) based on the differences between observations and estimations, depending on the degrees of freedom (c) and the sample size (n). The degree of freedom is the number of variables in the data-set minus one. For the two variates linear regression, the degree of freedom is one. The χ^2 formula is shown in Equation 6.1;

$$\chi_c^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{6.1}$$

A low value for χ^2 describes a high correlation between observed data and estimated fit. The calculated χ^2 can be compared to a predefined χ^2 table with a critical χ^2 value for various degrees of freedom. If the χ^2 is greater than the critical value, there is a significant difference between the observations and the estimated fit. This chapter proposes a Linear Regression model to estimate particle trajectories according to the Level-1 track reconstruction specifications for the High Luminosity upgrade of the CMS experiment. The algorithm will be described and discussed in terms of synthesisable algorithms and compatibility with front-end hardware developments in the Level-1 Tracking system.

6.2 Motivations for using Linear Fitter

The previous chapter described the use of the Kalman algorithm for particle trajectory estimation and fit calculations for the hardware-based track reconstruction. The evolution of the hardware-based Kalman algorithm for

use in the Level-1 Track-Trigger system [95] has resulted in an in-depth study and analysis of track reconstruction using an estimation algorithm for particle trajectories with transverse momentum exceeding 2 GeV/c. The Kalman system was initially conceptualised and developed for the TMTT firmware alongside the Hough-Transform and Duplicate Removal modules. A fully pipelined Kalman system can process stubs at 240 MHz within latency and hardware resources. Hardware simulations have shown that the Kalman system can achieve high-efficiency levels in physics performance at the expense of additional hardware resources [97]. To reduce resources and maintain high efficiency, an additional module known as the mini Hough Transform (mHT) was implemented to handle data formatting outside the Kalman system. Any added module to the tracking system results in larger overall latency and resources. Further examination of the Kalman system revealed that its complexity is driven by matrix-based arithmetic concerning covariances and error estimations in the feedback iterations [98].

The efficiency of the Kalman filter relies on using the full dynamic range of double-precision floating-point variables for the measurements of parameters to calculate the Kalman gain efficiently. In FPGA design, the representation of native parameters counterparts in integer-based or fixed-point values resulted in excessive utilisation of logic resources for equal efficiency levels. Implementing a full integer-based model is not feasible, considering multiple copies of the Kalman system will be required in a full tracking system architecture. Additional techniques are deployed to reduce resource usage in multiply-accumulate operations with a minimised number of bits without considering the effects of precision loss and overflow, which otherwise would introduce extra bits in guard bits declaration. Consequently, the efficiency deteriorated, and the effects being observed in the final fit efficiency simulation [99]. In physics simulation, the Kalman filter algorithm is used for both the r-ϕ and r-z planes without taking into account the p_T module's misalignment in r-z, resulting in efficiency loss in trajectory estimation for particles crossing from one plane into the other.

All the Kalman system advantages and disadvantages have been considered while developing the linear fitter trajectory estimator that uses a simple multivariate Linear-Regression approach. In particular, the concept of complexity reduction in hardware implementations by considering the effects of deploying simple arithmetic equations in the track reconstruction architecture. Once the hardware specifications have been met in the full system, additional features can easily be integrated into the architecture with room for improvement (e.g., curved track reconstruction plugin). The Multivariate Linear Fitter algorithm is optimised for a higher frequency of 400 MHz, reducing the latency in compliance with the Level-1 Tracking specifications. The Linear algorithm estimates a fit immediately after the geometry processor block without requiring the Hough Transform algorithm and data conversion block in the r-ϕ and r-z planes. The effects of the detector's layer alignments in Pixel-Strip and Strip-Strip modules in the barrel and endcap regions are encoded into the algorithm, which provides similar efficiency in both planes. The Linear Fitter system has been developing in parallel with the Kalman system; hence, it is compatible with TMTT and Hybrid architectures.

6.3 Linear Fitter for Track Finding

The tracker geometry of the HL-LHC upgrade in the tracker barrel and endcaps follows the p_T module concept with tilted modules for maximised efficiency in trajectory reconstruction. The Linear Fitter algorithm uses both Pixel-Strip and Strip-Strip modules within the specified ranges of the CMS detector geometry. The algorithm uses the CMS coordinate system, as defined in the Section 2.9.

In the conventional Multivariate Linear Regression approach, the equation of a line in terms of the independent variable x and dependent variable y with gradient m and intercept b is shown in the Equation 6.2:

$$y = mx + b \tag{6.2}$$

The Equation 6.2 is derived from the summation of all distances E to the fitted line and each data point as given in Equation 6.3:

$$\boldsymbol{E} = \sum_{i=0}^{n-1} (\boldsymbol{y_i} - (\boldsymbol{m x_i} + \boldsymbol{b}))^2 \tag{6.3}$$

Through a series of algebraic operations, the summation produces the gradient and intercept as described in more detail in Appendix A5, The final equation for these quantities are:

$$\boldsymbol{m} = \frac{\bar{\boldsymbol{x}}\bar{\boldsymbol{y}} - \overline{\boldsymbol{x}\boldsymbol{y}}}{\bar{\boldsymbol{x}}^2 - \overline{\boldsymbol{x}^2}} \tag{6.4}$$

$$\boldsymbol{b} = \bar{\boldsymbol{y}} - \boldsymbol{m}\bar{\boldsymbol{x}} \tag{6.5}$$

The values for $\overline{x}$ and $\overline{y}$ are the mean of x and y variables, respectively. The conversion from (x, y, z) to (r, ϕ, z) is defined in the r-ϕ and r-z planes independently as shown in Equations 6.6 and 6.7:

$$\begin{aligned} \boldsymbol{m} &= \frac{\bar{\boldsymbol{r}}\bar{\boldsymbol{\phi}} - \overline{\boldsymbol{r}\boldsymbol{\phi}}}{\bar{\boldsymbol{r}}^2 - \overline{\boldsymbol{r}^2}} \\ \boldsymbol{b} &= \bar{\boldsymbol{\phi}} - \boldsymbol{m}\bar{\boldsymbol{r}} \end{aligned} \tag{6.6}$$

$$\begin{aligned} \boldsymbol{m} &= \frac{\bar{\boldsymbol{r}}\bar{\boldsymbol{z}} - \overline{\boldsymbol{r}\boldsymbol{z}}}{\bar{\boldsymbol{r}}^2 - \overline{\boldsymbol{r}^2}} \\ \boldsymbol{b} &= \bar{\boldsymbol{z}} - \boldsymbol{m}\bar{\boldsymbol{r}} \end{aligned} \tag{6.7}$$

The Linear Regression model finds the relationship between two variables r and ϕ in the r-ϕ plane, independent of r and z in the r-z plane. Even though the straight lines are calculated and fitted in two planes with the stub radius as their common shared parameter, minimising the squared distances is determined in one dimension.

Once the gradient and slope are calculated, they are incorporated into the equation of a line 6.2 to calculate the estimated ϕ values. The process is repeated to calculate the estimated z values. The fit generated by these values in each plane must minimise the distances between the fit and data points,

resulting in the least sum of squared distances in that plane. Additionally, the summation of these values obtained from each plane must return the least squared distance to data points in the three-dimensional plane.

The feasibility of converting a three-dimensional data-set into two separate fits is examined in Figure 6.2. The figure displays four plots with Subplot 1 showing the data points (stubs) density in the (r, ϕ, z) plane. In Subplot 2, a linear fit is generated in the three-dimensional plane. In Subplot 3 and Subplot 4, the fits in r-ϕ and r-z superimposed on data are displayed in two-dimensional planes separately. The advantage of this approach is the flexibility of encoding the tracker misalignment in the r-z plane separately from the r-ϕ plane when calculating the stub distances from the generated fit, hereafter called the residuals.

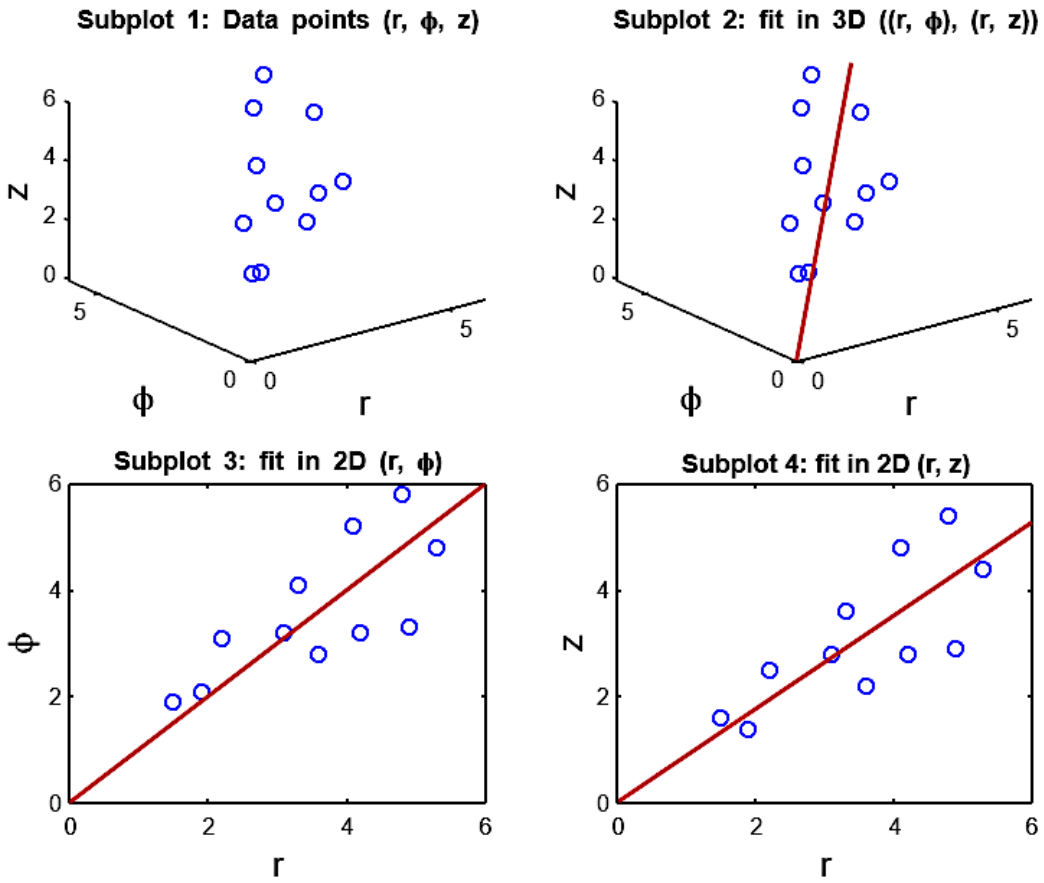

Figure 6.2: Converting a 3D fit into two 2D fits.

The gradient and the intercept of the fit in r-ϕ are used to calculate the sum of squared residuals in the r-ϕ plane as seen in Equation 6.8:

$$\boldsymbol{E_\phi} = \sum_{i=0}^{n-1} (\boldsymbol{\phi_i} - (\boldsymbol{m_{r\phi} r_i} + \boldsymbol{b_{r\phi}}))^2 \tag{6.8}$$

The gradient and intercept of the fit in the r-z plane are used, taking into account the stub radius offsets as $r_{r\phi}$ and r_{rz} to increase efficiency in the trajectory estimation as shown in Equation 6.9:

$$\boldsymbol{E_z} = \sum_{i=0}^{n-1} (\boldsymbol{z_i} - (\boldsymbol{m_{rz}}(\boldsymbol{r_{r\phi} r_i} + \boldsymbol{b_{rz}}) - \boldsymbol{r_{rz}}))^2 \tag{6.9}$$

The residuals from the estimated trajectory to the individual stubs in two planes are calculated for all stubs in two planes as shown in Equation 6.10:

$$\boldsymbol{E_{(\phi z)_i}} = \boldsymbol{E_{\phi_i}} + \boldsymbol{E_{z_i}} \tag{6.10}$$

If the stubs are detected in the tracker endcaps, additional parameters are introduced in the calculation of their residuals depending on where the $\boldsymbol{p_T}$ modules associated with their original hit are located. The parameters are chosen to increase tracking efficiency and hand-coded into the algorithm. For instance, if a stub originated in the endcap and its hit information indicates that it is from a Pixel-Strip module, its residual E_z is divided by an offset to compensate for shorter distances between its $\boldsymbol{p_T}$ modules in the endcap regions. If the stub is in the barrel and it originated from a Strip-Strip module, its residual E_z is divided by an offset to compensate for the larger distance between $\boldsymbol{p_T}$ modules in the barrel. Other classifications are introduced based on individual layer alignments in barrel and endcap regions to increase efficiency. These include finding the maximum $\boldsymbol{p_T}$ module population areas in the tracker and introducing offsets to concentrate the computation intensity on these regions. The sum of squared residuals is calculated by using three models, Sum of Squares Total (SST), Sum of Squares due to Error (SSE) and Sum of Squares due to Regression (SSR). The SST calculates the ϕ residuals from the mean $\bar{\phi}$. The SSE calculates the ϕ residuals from estimated $\hat{\phi}$. The SSR is the residuals in estimated $\hat{\phi}$ from the mean $\bar{\phi}$.

Figure 6.3 shows these models with their equations concerning the stub location, estimated fit and mean.

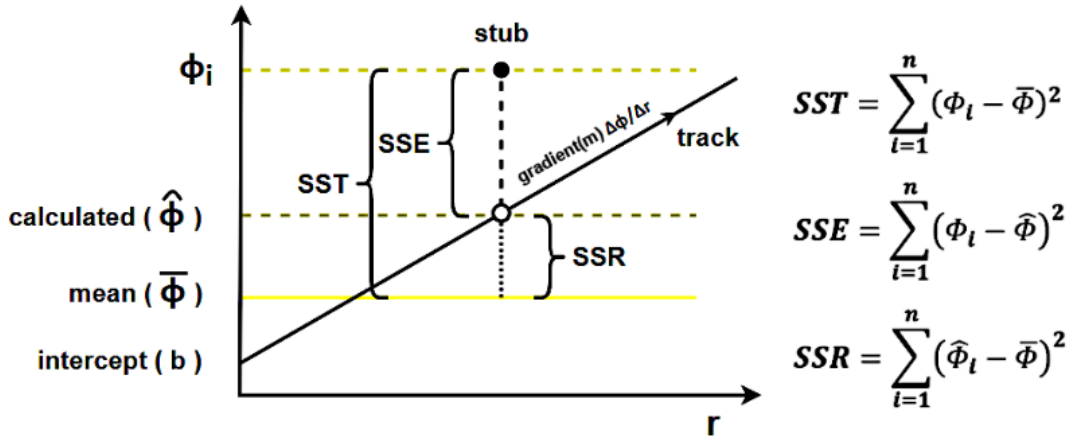

Figure 6.3: Models for calculating Sum of Squares in the r-ϕ plane.

Similarly, in the r-z plane, SST, SSE, and SSR determine how well the estimate fits the data. The r coordinate or the distance from the interaction point is the shared parameter between the r-ϕ and r-z planes, which becomes the relating factor in calculating residuals in both planes. It is only logical to use the same sum of squares model in calculations of residuals in the r-z plane for transparency, as shown in Figure 6.4.

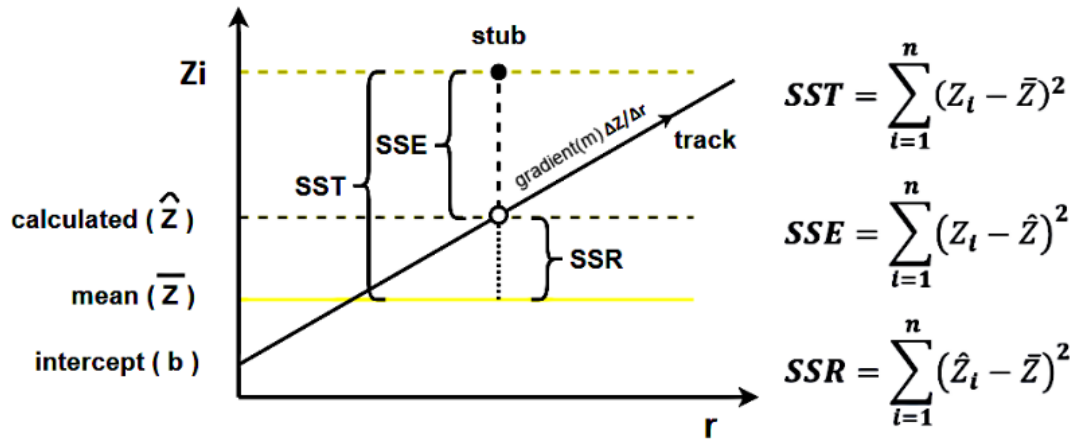

Figure 6.4: Models for calculating Sum of Squares in the r-z plane.

Generally, just one of these arrangements will be sufficient to establish the relationship between the data. However, both are typically calculated, and the best is selected. The SSE model calculates the errors or possible corrections to the fit without storing the mean in a loop, resulting in fewer logic resources in the hardware implementation.

Finally, the goodness of the fit χ^2 is calculated using the Equation 6.11. If χ^2 > significance level, the estimated trajectory is rejected; otherwise, the trajectory is flagged as a valid candidate track and stored in memory for further examination.

$$\chi^2 = \frac{\boldsymbol{SSR}}{\boldsymbol{SST}}$$
$$\chi^2 = 1 - \frac{\boldsymbol{SSE}}{\boldsymbol{SST}} \tag{6.11}$$

The χ^2 is also used in finding the efficiency of the tracking system by taking into account the total number of tracks before and after the estimation for many events as shown in Equation 6.12.

$$\chi^2 = \sum_{\boldsymbol{i}=1}^{\boldsymbol{tracks}} \frac{(\boldsymbol{observed_i} - \boldsymbol{estimated_i})^2}{\boldsymbol{estimated_i}} \tag{6.12}$$

6.4 Tracker Geometry for Linear Fitter

In the Time-Multiplexed Linear Fitter, the tracker is divided into $2\pi/9$ regions known as nonants. The trajectory of particles with p_T > 2 GeV/c is observed as a straight line in the r-z plane; however, the trajectory may slightly bend in the $r-\phi$ plane, entering the neighbouring regions. Hence, the tracker is divided into two areas of high and low efficiency. A trajectory in overlapping areas contributes to the reconstruction of two candidate tracks and consequently becomes duplicates. The track finding algorithm can be modified to differentiate between overlapping and non-overlapping areas as low efficiency and high efficiency, respectively.

The overlapping areas are calculated by the Equation 6.13 incorporating a track radius r_T with transverse momentum p_T originating from the centre of the detector and interacting with a PS or 2S module in its path through the magnetic field B in the CMS. For a 2 GeV/c particle, the r_T is approximately 1.7 m, which is larger than the radius of a p_T modules on the layers of the tracker.

$$r_T[m] \approx \frac{p_T[GeV/c]}{0.3B[T]} \tag{6.13}$$

As shown in the Figure 6.5, the tracker is divided into nine nonants with circles representing tracks with p_T = 2 GeV/c. The darker regions represent the overlapping areas superimposed on the tracker layers: Layer 1 (red) to Layer 7 (blue).

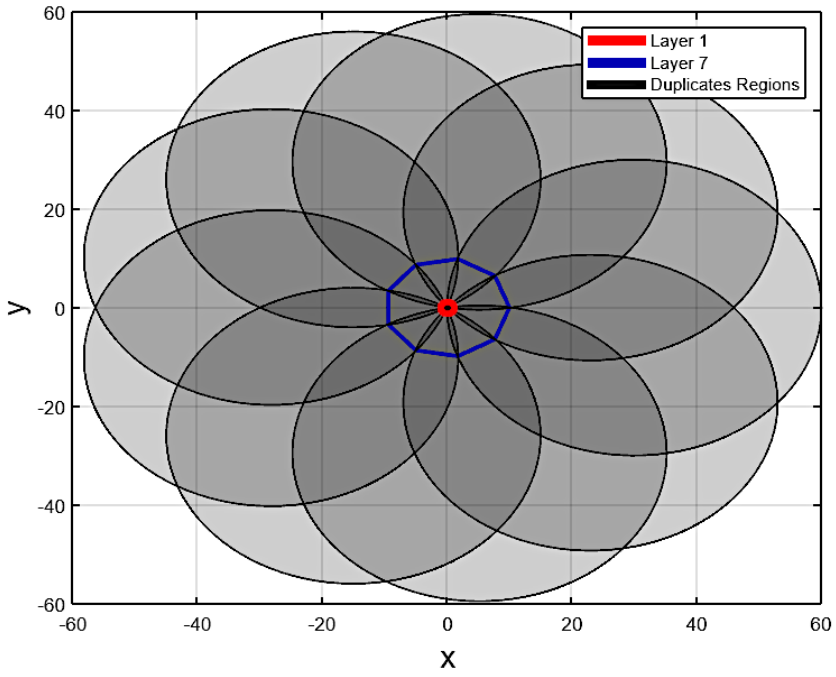

Figure 6.5: Tracker segmentation for increased efficiency.

The overlapping areas will be minimal as a particle trajectory approaches a straight line. If the bend is small enough to be neglected, the tracking process can be implemented without cross-regional communication between nonants, significantly reducing the algorithm complexity in software and hardware implementation.

6.5 Virtual Stubs in Linear Fitter

The Linear Fitter algorithm is optimised to process high occupancy 200 pileup events in layers closer to the beam pipe. Following the regional segmentation of the tracker into stub sectors, calculating virtual stubs is a necessary step in the track reconstruction algorithm. Virtual Stub creation reduces the stub occurrences on each tracker layer within stub sectors by merging all existing hits to one per layer. The model for the virtual stub module is chosen carefully as the calculated stub per layer must be a good presentation of all stubs on the processing layer. Figure 6.6 shows an illustration of possible stub occupancy on four layers in r-ϕ and r-z planes. The dotted lines define the stub sector boundaries with the minimum and maximum values configured by the Geometry Processor and top-level modules. These values are passed to the individual instances of the Linear Fitter. Blue dots represent particle hits on each tracker layer, randomly selected in this instance. The algorithm must differentiate between inner and outer boundary particles.

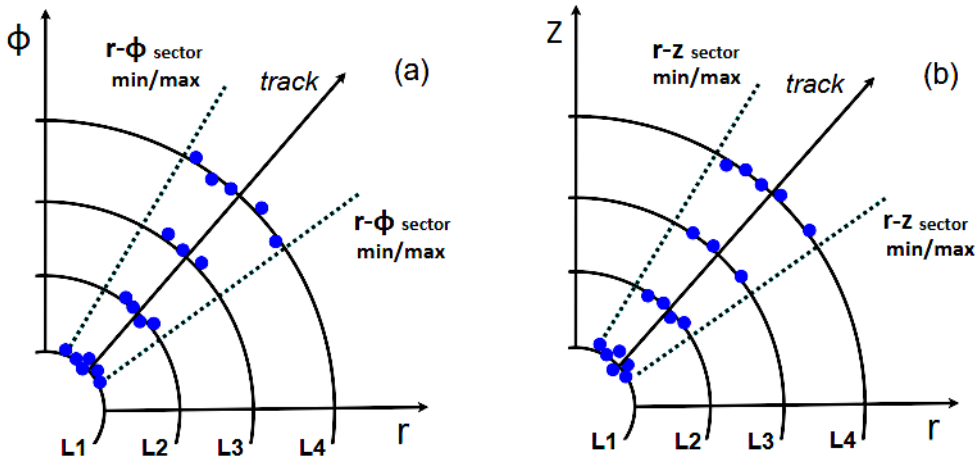

Figure 6.6: Hit population in a: r-ϕ and b: r-z plane.

In conventional Descriptive Statistics practices, the popular models to calculate an average of random samples are the mean, median, and mode [96].

If the total number of particle hit is denoted as n and the number of stub occurrences as (x = x_0, x_1, ..., x_{n-1}), the mean or the Simple Average Model as shown in Equation 6.14:

$$\boldsymbol{X_{mean}} = \frac{1}{\boldsymbol{n}} \sum_{\boldsymbol{i}=0}^{\boldsymbol{n}-1} \boldsymbol{x_i} \tag{6.14}$$

The equation is most commonly used in statistics when the total number of samples is known at the runtime[2] of the system. In a fully pipelined Linear Fitter, the stubs are processed as they arrive at module's inputs and are processed immediately, meaning the total number of stubs is unknown at runtime. As a result, calculating virtual stubs using the mean model is not practical. Therefore, an alternative method has been developed and implemented to modify the mean algorithm for pipelining to solve the runtime problem and the unknown number of hits. In this model, the first iteration only sees the first stub at its input and stores it in an intermediate variable $x_{iter(1)}$. In the second iteration $x_{iter(2)}$, the second stub arrives at the input, which is used in the calculation of the mean of two variables x_1 and x_2. In the third iteration, the next stub $x_{iter(3)}$ arrives and the mean value is calculated in the second iteration $x_{iter(2)}$ is used to calculate the mean of x_1, x_2 and x_3. The process continues with the result from the previous iteration used to calculate the mean in the current iteration until the last stub is cleared. In Equation 6.15 the arithmetic operations in the model with a multiplication, an addition and a division operator are shown.

$$\begin{aligned}
\boldsymbol{x_{iter(1)}} &= \boldsymbol{x_1} \\
\boldsymbol{x_{iter(2)}} &= \frac{1(\boldsymbol{x_{iter(1)}}) + \boldsymbol{x_2}}{2} \\
\boldsymbol{x_{iter(3)}} &= \frac{2(\boldsymbol{x_{iter(2)}}) + \boldsymbol{x_3}}{3} \\
\boldsymbol{x_{iter(i)}} &= \frac{\boldsymbol{i}(\boldsymbol{x_{iter(i-1)}}) + \boldsymbol{x_{i+1}}}{\boldsymbol{i}+1}
\end{aligned} \tag{6.15}$$

[2]The instance the algorithm commences the stub processing in real-time.

The software implementation of this model produces the correct results mathematically. The hardware implementation of the model also produces reliable synthesis results in a fully pipelined structure. However, it fails to meet the minimum requirements in terms of logic resources. The reason is the division operation with the divisor ranging from integer 2 to n, depending on the number of samples. The division is a costly operation in hardware implementation for its use of logic resources and the latency associated with them. In Equation 6.16, the cross-conversion model for replacing the division with a shift operation is shown provided that the divisor is a radix$_2$ binary number. Shifting bits in hardware can be implemented by only hard-wiring bit locations without any use of logic gates. If log2(n) returns a decimal number, it will be rounded to the nearest value that can be represented in binary. This will cause precision lost which can affect the correctness of the results. In the mean model, the cost is significant and therefore its use in the Linear Fitter design was avoided.

$$x/n = x >> \log_2(n) \qquad \textit{where,} \qquad n = (1, 2, 4, 8, 16, ...) \tag{6.16}$$

Other approaches such as the median[3] also result in pipelining issues in hardware. In the median model, all stubs must be present in ascending or descending order to find the exact middle values in the data set. The implementation requires all stubs to be stored and then sorted before using equality logic to find the midpoint. If the number of stubs is even, an additional division or shifting operation is required to find the mean of two midpoint values. In this instance, additional operations excessively utilise logic resources and latency.

An alternative mean approach produces accurate results in a pipelined structure and uses minimum hardware resources due to utilising a shift operator instead of division. Following the geometric segmentation of the tracker regions, the minimum and maximum values for the stub sector boundaries are retrieved and stored in variables (SectorMin and SectorMax) for each parameter ϕ, $r\phi$, z and rz coordinates. For initialisation, the maximum value of the sector boundary is assigned to the sectorMax and the minimum value of the sector boundary is assigned to SectorMax. The

[3]Midpoint of the stubs distribution.

arrangement helps to pinpoint the stub position within a stub sector. In the next step, the minimum and maximum values for the sector boundary and the first stub position on its corresponding layer are found by min/max conditional operators. The minimum value is added to the maximum value followed by the shift operator to find the virtual stub position on the layer. If there are more stubs present on the layer, a second iteration is used to find the mean of stubs to create only one virtual stub per layer. The arithmetic blocks used in the calculation are an adder and a shift operator. This reduces the cost of hardware and consequently the latency associated with the logic resources. The design can be fully pipelined in a parallel logic architecture proportional to the number of stubs on that layer. The process for the virtual stub (VStub) creation for the ϕ parameter is shown below;

Initialisation only:

SectorMin_ϕ	= Max value of stub-sector boundary in ϕ
SectorMax_ϕ	= Min value of stub-sector boundary in ϕ

Loop over stubs on a layer:

SectorMin_ϕ	= min(SectorMin_ϕ, StubPos_ϕ)
SectorMax_ϕ	= max(SectorMax_ϕ, StubPos_ϕ)
Stub_ϕ	= SectorMin_ϕ + SectorMax_ϕ
VStub_ϕ	= $\text{Stub}_\phi >> 1$

6.6 Linear Fitter Flow Control

A fully pipelined implementation of the Linear Fitter divides the module into smaller and better manageable blocks. The approach deploys the null algorithm strategy 8.9 in the Track Finder Processor top-level firmware design for faster integration of the individual blocks. Time-Multiplexed Linear Fitter modules are instantiated in Track Finder Processors in parallel configurations. A single module of the Linear Fitter contains two types of blocks for data processing and data calculations. The data processing blocks are responsible for routing the streams of stubs between blocks, updating the runtime values for stubs and layers for bookkeeping, and determining the exit strategy. The calculation blocks contain arithmetic operators for fit and residuals estimation. In Figure 6.7, the components of the Linear Fitter are shown. The arrows show the data flow direction, and the partitioned stacks

display the memory structures for stubs and reconstructed tracks. In this configuration, the stubs are stored in memories before and after link formatting to control the I/O data flow. N number of tracks are produced and stored for the final track reconstruction algorithm to select the best estimate. In the simulated 200 pileup events, the algorithm expects high layer populations[4] in all tracker layers. The number of stubs on the layers closer to the interaction point is higher than those further away. In the Linear Fitter, a module is designed to keep the count on the tracker layers. The layer population module holds information about the hit patterns as they are detected. The information from the layer population is used in the check validity block for hit validations based on association with the $p_T \geq 2$ GeV/c particles.

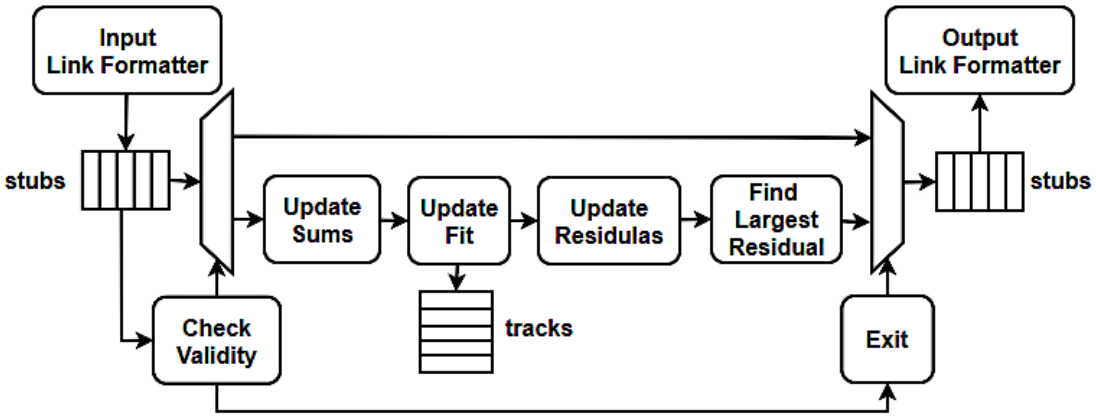

Figure 6.7: Linear Fitter in the pipeline configuration.

In fitting a straight line, at least two hits (stubs) on two different layers of the tracker are required to reconstruct a partial path. The check validity block updates the total number of stubs and layers counts according to whether the hit was obtained from a PS or 2S module. Considering the data from the p_T modules independently increases the track quality efficiency as the corresponding module alignments can be incorporated into the algorithm. The check validity block also determines an exit strategy for terminating the iterations when the desired number of stubs are filtered. The exit strategy also considers the number of stubs in PS and 2S layers to ensure two stubs remain on two layers of the tracker at runtime. For a

[4]Many hits or stubs are expected on each layer of the tracker.

stream of sixteen stubs, if a maximum of sixteen iterations are executed, and a valid candidate track is not reconstructed, the algorithm flags all calculated fits as invalid. The bookkeeping functions in the check validity block are fully pipelined, ensuring a steady transmission of one stub per clock cycle until all stubs in the stream have been processed. In the calculation blocks, the Linear Fitter arithmetic equations are implemented. The update sums block receives the stubs in streams and verifies if the coordinates are within the stub sector boundaries. It also applies corrections to the r coordinate in the r-ϕ and r-z planes by chosen offsets; $r_{r\phi}$ and r_{rz} to compensate for track inefficiencies arising from the p_T modules and the gaps between layers. The summation and mean calculations use individual hit coordinates on a layer-to-layer basis to create a virtual stub per layer. The values of counters are also incremented, indicating how many virtual stubs are generated at runtime. The calculated values for gradient and intercept are prepared for the next block in both planes. The calculate fit block is the most computationally intensive as it performs the fit arithmetic. In this block, the summation values from the update sum block are used to generate fit parameters in the form of $\text{gradient}_{r\phi}$, $\text{intercept}_{r\phi}$, gradient_{rz} and intercept_{rz} accounting for the p_T modules offsets in both planes. Find the largest residual block proceeds with the fit parameters to calculate the sum of squares in ϕ and z independently. Further corrections are made to the residuals to align PS and 2S modules in the barrel and endcaps. The sum of squared residuals is calculated for all stubs by calculating the distance between the observed and estimated values as the summation of residuals in the r-ϕ and the r-z.

In the remove largest residual block, the individual stub residuals are compared, and the largest value is selected, which pinpoints the stub with the greater distance from the fit. The stub with the largest residual is removed from the stream. The check validity block records the stub and updates its counters accordingly. The exit conditions decide if the process must terminate or continue based on the remaining stubs in the stream. If the block determines that the next iteration is necessary, the remaining stubs are considered in the next processing iteration until the minimum number of stubs is reached, and the algorithm terminates accordingly. The output is a stream of valid/invalid stubs and genuine/fake tracks. The stubs that survived the filtering are flagged as valid stubs, and their corresponding track

is flagged as genuine. The final track parameters are updated and stored in memory for track quality check. The Linear Fitter is interfaced to the Track-Trigger firmware chain via the input link formatter and output link formatter modules. The demultiplexer and multiplexers operate as switches for stub routing between firmware modules in the tracking chain. When a track reconstruction is validated by the previous modules and the valid bit of the track word is set, the check validity block recognises and passes the reconstructed track to down-chain modules. As a result, the track will skip the processing operation by default and will transmit instead to the output link formatter. The building stubs of the reconstructed track can be accessed through the stub memories in the succeeding clocks.

6.7 Performing a Fit in a Pipelined Structure

The virtual stubs created for each tracker layer with stub occupancy > 1, return the (r, ϕ, z) coordinates per layer calculated at runtime. If there is only one hit per layer, the hit is flagged as the virtual stub. The next step is to use the parameters in the fit calculations in the two planes. The equation for the fit contains multiply/accumulate at its core, with multiplication as the major contributor to logic resources and latency. An asynchronous multiplier is designed and implemented with only AND and XOR logic gates to control the hardware at the multiplication stages. Designing an asynchronous multiplier [100] has advantages in Hardware Description Language (HDL) and High-Level Synthesis (HLS) implementations. In HDL standard library packages, the multipliers often have additional multi-purpose logic gates to make the entity suitable for all designs. The added logic capability adds unnecessary resources to the overall system design. The standard library multipliers also limit the user's access to the data path, limiting the use of rounders and limiters to quantize the word length at runtime. The custom-designed asynchronous multiplier minimises additional control logic resources and grants access to the data path, which is critical to managing the bit growth. An implementation of a 4-bit multiplicand a and 4-bit multiplier b with an 8-bit product p is shown in Figure 6.8. The partial additions using AND gates are denoted as (p_{0-7}), and the parity checks are denoted as (x_{1-5}). In the bitwise configuration, multiplicand and multiplier

can be either '0' or '1'; however, the only multiplier with value '1' can change the result of partial addition in locations $1 \rightarrow 8$. The intermediate locations in the multiplier are hardwired with the critical path not exceeding the 1/sc (system clock) frequency at 400 MHz. The combinational structure produces the results as long as the values are present at the multiplier input (a, b). The implementation strategy allows new transitions to occur earlier than the next clock cycle, reducing the overall system latency. Concerning the hardware resources, the building blocks of the multiplier only contain primitive logic gates.

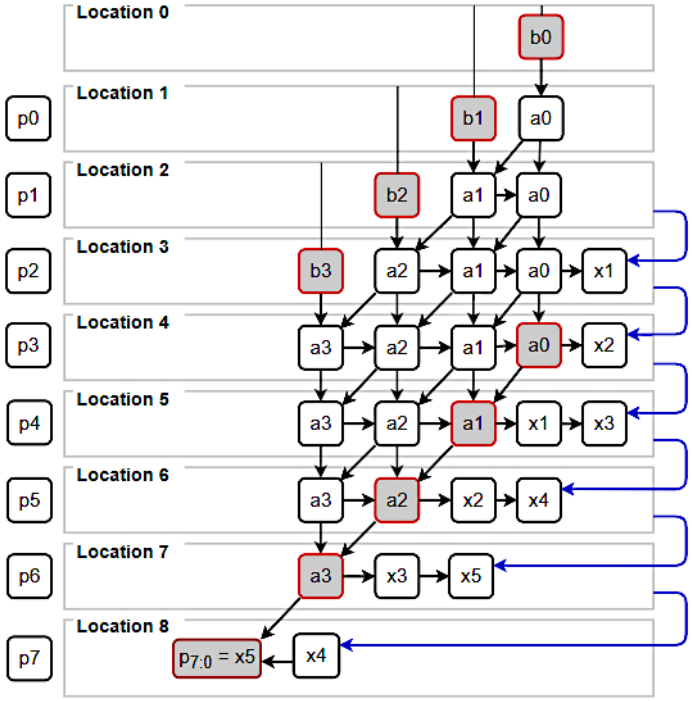

Figure 6.8: 4-bit asynchronous multiplier with primitive logic.

The multiplication operations in Xilinx FPGA chips are passed to the DSP48E slices. These blocks are designed to perform multiply/accumulate operations with added registers for pipelining the data path. The DSP48E blocks are efficient and effective for arithmetic operations; however, there are limited slices available on an FPGA chip depending on the product's series and costs. In large projects such as Kalman Filter or Linear Fitter, there are not enough arithmetic units for instantiation of 16-18 parallelised modules on the most expensive chip; hence, utilisation of DSP48E slices is avoided unless necessary. A fully pipelined and scaled 18-bit version of the

asynchronous multiplier is implemented in HDL to carry out multiply/accumulate in the Linear Fitter fit calculations. The Multiply/Accumulate (MAC) operation is a reoccurring operation in DSP projects. Designing an efficient MAC slice is crucial in providing accurate results from Linear Fitter calculations, considering the overflow and the precision loss can completely change the arithmetic operation's outcome. Hence, a custom-designed fully pipelined MAC architecture is implemented containing the asynchronous multiplier to access the data path within the structure and to remove dependency on the global system clock. The characteristics of MAC benefit the design in minimising the latency, minimising logic resource usage and reducing power consumption. The architecture of the MAC is shown in Figure 6.9.

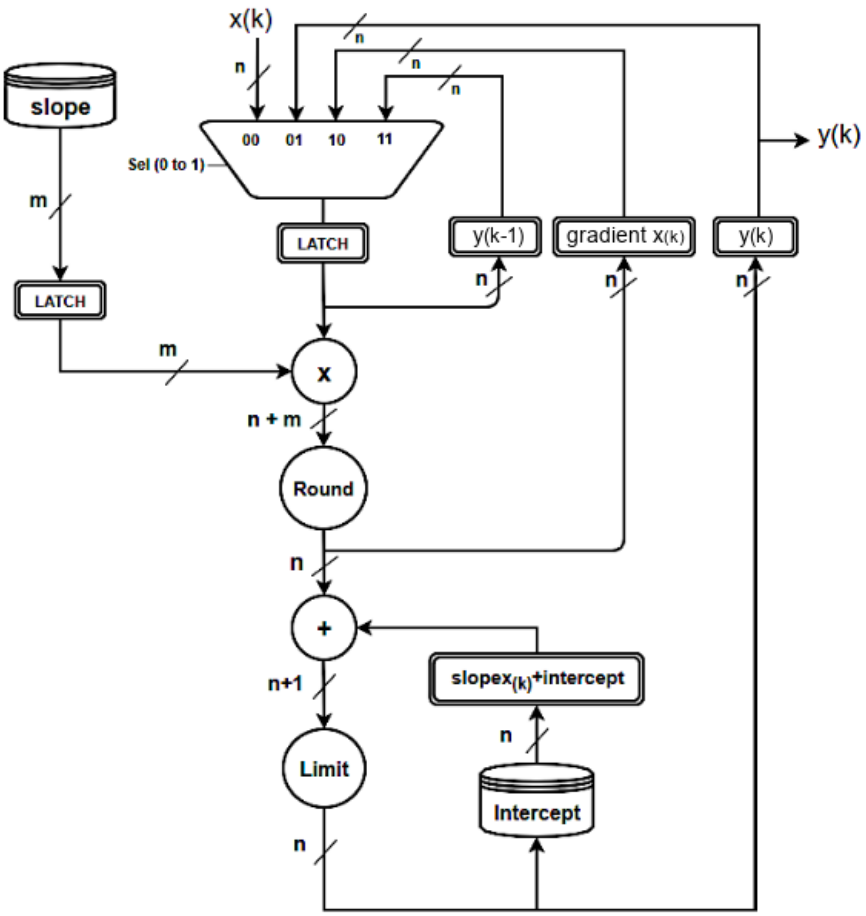

Figure 6.9: Fully pipelined MAC for fit architecture.

In this architecture, the fit is calculated and pipelined in four clock cycles producing the result for the Equation 6.17:

$$y_k = gradient(x_k) + intercept \tag{6.17}$$

The gradient and intercept are available to the block at every rising clock edge. The multiplexer (4:1) permits only one input to go through the data path, controlled by a latch. After the latch, the asynchronous multiplier produces the result for the first part of the equation. A custom-designed Rounder block is introduced after the multiplier quantizes the data to the desired word length. The multiplication of n-bit multiplicand and m-bit multiplier produces an (n+m)-bit product that can grow in length at every iteration if not cut back to n+m. After multiplication, the data path holds the result for gradient(x_k), quantized.

A custom-designed limiter block is placed after the add/accumulate operator stops the word length from growing, otherwise resulting in overflow and the incorrect output. The accumulator using an n-bit adder can produce an (n+1)-bit result that can contribute to the word length becoming larger during iterations. The intercept value is added to the data path through a clocked latch. The path after the addition operator holds the value of y_k for the first iteration, which is fed back to the adder in the next iteration. The y value is read at the final clock cycle of the fourth rising edge of the clock. The y_k can be read at every clock cycle, which yields a partial fit calculation. The complete fit is driven after the last stub has entered and cleared from the last (y_k) latch. The rounder and limiter are essential features of the MAC data processor described in the following sections.

6.8 Integer-based Division for Linear Fitter

In calculating the gradient, the division operation (a/b) is separated into multiplication and reciprocal operators (a x 1/b). For divisors, with less than 10 bits, the lookup tables store all possible division results in tabulated form with the precision of 0.0625 and made available at runtime. Even though the implementation is hardware friendly, it can not be used for operations where

the divisor has bits more than 10 bits, leading to excessive memory usage. Hence, an alternative custom-designed hardware structure is implemented and used to calculate reciprocal operations with minimum logic resources. The process begins by declaring n-bit registers to hold the values for dividend, divisor, quotient and remainder with the initiation of all bits to zeros except the most significant bit of the dividend, which is set to '1'. In the first iteration (i = 0), the binary values for the remainder and quotient are shifted to the left by one bit. A full custom-designed asynchronous adder/subtractor subtracts the divisor from the remainder, with the result stored back in the remainder. For example, if the most significant bit in the remainder is '0', the quotient i^{th}-bit becomes '1'; otherwise, the subtraction is reversed, and '0' is assigned to the quotient i^{th}-bit. The process continues until all bits in the divisor are shifted out of the register. The model follows the implementation of Restoring Division in ASIC design [101]. The architecture of the reciprocal slice is shown in Figure 6.10:

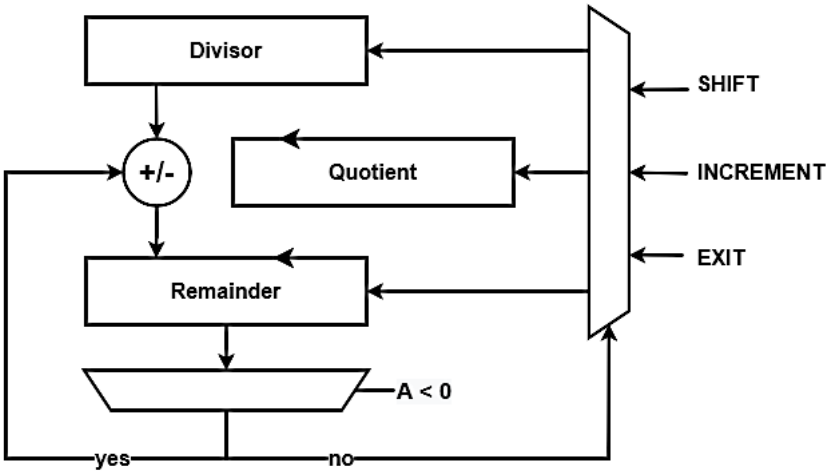

Figure 6.10: Pipelined reciprocal hardware architecture.

A low-power asynchronous Adder/Subtractor in the fully pipelined architecture is implemented using custom-designed logic. The HDL Standard Cell Library does not provide Adders with serial data flow characteristics. The inspiration for the design comes from microelectronics system design practices [102], [103]. The modified logic uses only one n-bit full-adder in a serial configuration using only AND, XOR and OR gates. To convert the adder

into a subtractor, the simplest approach is to negate[5] b in a + b n-bit addition to form a + -b subtraction logic in hardware. An XOR logic gate controls the add/subtract operations in the architecture. The latches hold the operation result and can be controlled by the rising/falling edge of the clock 2^n times faster than the master or global system clock. The n value is selected to compensate for the 1/sc system clock idle period required to calculate the full result of addition or subtraction. The period that the logic is forced to stay idle is approximately 75% of the system clock period at 40 MHz frequency. The critical path in this architecture is the carry bit which limits the speed of data flow and consequently dictates the overall latency of the structure. A gate view of the architecture is shown in Figure 6.11;

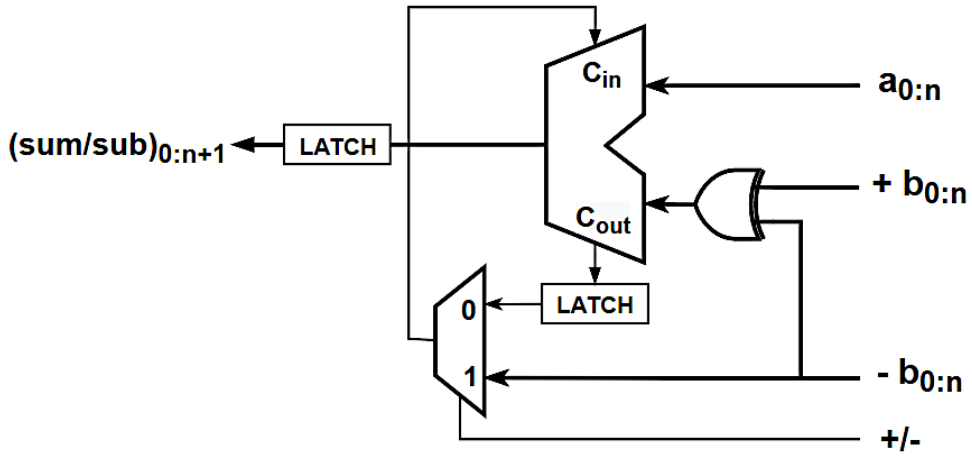

Figure 6.11: Adder/Subtractor for the reciprocal architecture.

6.9 Regression Fit with Fixed Latency

Finally, the logic blocks (Multiplier, Adder/Subtractor, Multiply/Accumulate, Reciprocal) are placed in the fixed latency Linear Fitter structure in a fully pipelined architecture as shown in Figure 6.12. The rising edge of the clock shows the timing for every stage of the architecture. A stub in the Level-1 Track-Trigger system has at least three parameters (r, ϕ, z). As the Linear Fitter conducts its operations in two planes, the parameters are grouped into

[5]Negation in two's complement is the inversion of bits followed by adding 1 to the result.

(r, ϕ) and (r, z), which helps to compare the hits and stub sector boundaries for the generation of virtual stubs. The information is divided into two (numerator, denominator) for the r-ϕ and the r-z plane, followed by the reciprocal operator. The values for gradient and intercept are calculated to determine track parameters in both planes. The total latency for the architecture is fixed at sixteen clock cycles. As the latency and number of stubs in the streams are the same, the processing interval of one stub per clock cycle is achieved.

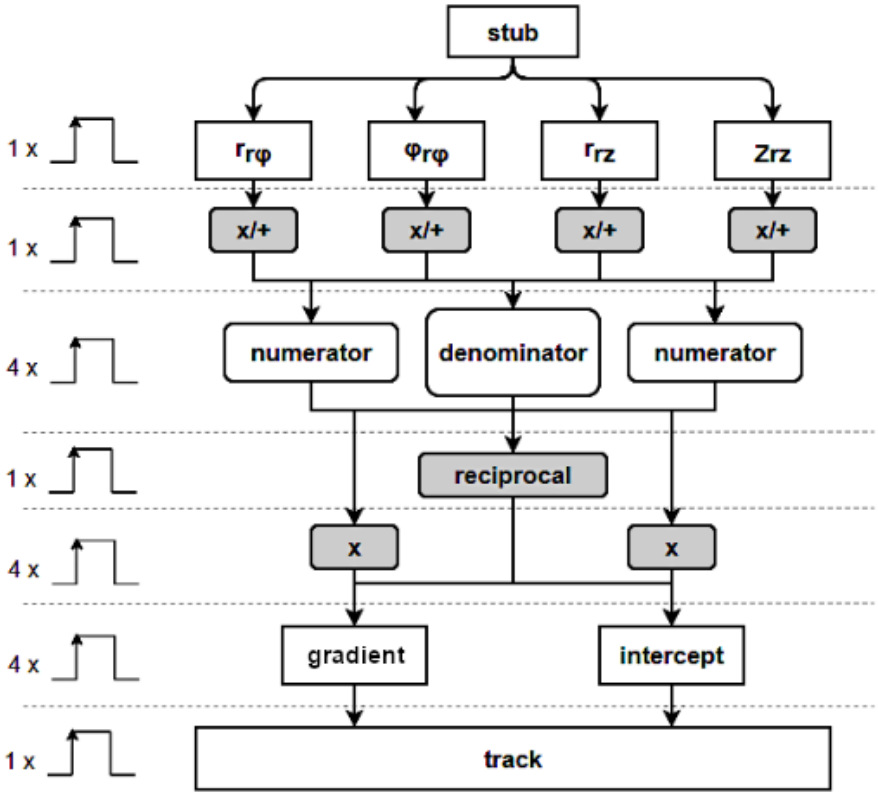

Figure 6.12: Logic architecture (16 clocks) for driving a fit.

At this stage, about 95% of the reconstructed tracks are invalid. The module creates a track in every clock cycle without determining if the track is genuine. To separate the genuine tracks from all tracks, invalid stubs that are wrongly associated with the track must be identified and eliminated up to only two stubs of PS or two stubs of 2S remaining on two PS or 2S layers. The removal of stubs is implemented in the next section.

6.10 Residuals in a Pipelined Structure

The general definition of a residual in the Linear Regression algorithm is the distance between the individual measurement and the estimation. In the Linear Fitter, a residual defines the distance between each stub and the fit, calculated for all stubs in the stream. If the fit is performed correctly, the residuals appear to have random approximation from the fit and are scattered around the fit. If the residuals appear to have systematic patterns, then the generated fit has not been performed correctly. Analysing the residuals at this stage provides insight into the algorithm performance, ensuring that the arithmetic operations in the previous stages have not encountered large precision loss or overflow. Calculating residuals permit stub identification with the largest distance from the estimated fit. The stubs with the largest residuals indicate that the hit was wrongly associated with the trajectory and did not belong to the reconstructed track. These stubs are flagged for elimination and will eventually be removed from the stub stream. Only stubs closest to the fit will remain to generate the final track.

In hardware, all the custom-designed arithmetic blocks are reused in finding the stubs with the largest residuals in a pipeline structure. The algorithm specifically calculates ϕ_r in the r-ϕ and z_r in the r-z plane. The equations for the arithmetic are shown in the Equation 6.18. The asynchronous adder/subtractor is utilised in both additions and subtractions in arithmetic operations.

$$\begin{aligned} \phi_r &= \phi_{r\phi} - (\boldsymbol{intercept}_{r\phi} + \boldsymbol{gradient}_{r\phi}((\boldsymbol{r}_{r\phi} + \boldsymbol{r}_{r\phi}(\boldsymbol{offset})) - \boldsymbol{r}_{rz}(\boldsymbol{offset})) \\ z_r &= z_{rz} - (\boldsymbol{intercept}_{rz} + \boldsymbol{gradient}_{rz}((\boldsymbol{r}_{rz} + \boldsymbol{r}_{rz}(\boldsymbol{offset})) - \boldsymbol{r}_{r\phi}(\boldsymbol{offset})) \end{aligned} \tag{6.18}$$

The offsets are added to maximise the trajectory efficiency in tracker regions. The values for the offsets are selected according to the physical stub locations concerning high-occupancy areas. Once the results of residuals are obtained, the values must be squared to remove negative values and to prevent the sum from adding up to zero. As taking the square is costly in terms of logic resources in hardware, taking the absolute value of the

residuals is preferred. The offsets for p_T modules are calculated after taking the absolute values for ϕ_r and z_r as residual$_{PS}$ for PS modules and residual$_{2S}$ for 2S modules in barrel and endcaps. The selected offsets differentiate between PS and 2S modules according to the distance between them in different regions of the tracker. The thresholds are often hardwired between modules as constants or directly obtained from the tracker geometry, DTC, or track reconstruction utility modules. Once the residuals are recalculated with added offsets for ϕ_r and z_r, they are summed to describe the residuals in (r, ϕ, z). As the values are being processed, an inequality operator compares the values to find the largest. The stub with the largest residual is removed from the stream. All operations are conducted concurrently using parallelised logic gates to eliminate the use of memory blocks for storing the results of calculations.

6.11 Precision loss and Overflow

Using fewer binary bits in computations results in a major reduction in logic resource usage and decreases the total latency when implementing hardware. Choosing the correct number of bits to present a variable is critical to ensuring an accurate result mathematically. Standard library packages support finite word length or fixed-point data representation in HDL and HLS. However, the effects of quantisation and overflow are not automatically included in the toolsets. For this reason, the fixed-point implementation of Linear Fitter is designed in MATLAB. For arithmetic operations using binary numbers, two's complement[6] is a good choice. The binary representation supports both unsigned and signed variables with control over the binary point for declaring integer and fraction bits. The general presentation of a fixed-point variable x_{fp} in binary b with the number of integer bits n and the number of fraction bits f is defined by the following relation:

$$x_{fp} = \sum_{i=0}^{n-1} b_i 2^i + \sum_{i=1}^{f} b_i 2^{-i} \tag{6.19}$$

[6]Bit ranges $(0 \rightarrow 2^i - 1)$ for unsigned, and $(-2^{(i-1)} \rightarrow 2^{(i-1)} - 1)$ for signed numbers.

If guard bits are required, they are added to the Most Significant Bit (MSB) to facilitate larger integer bits. The quantisation model considers the minimum and maximum values that a number can present, divided by the number of bits (2^n-1) available for discretisation. The roundoff errors or the noise generated by quantisation is defined by $\pm Q/2$ that is not in any way proportional to the represented quantity. The quantisation model is shown in the Figure 6.13.

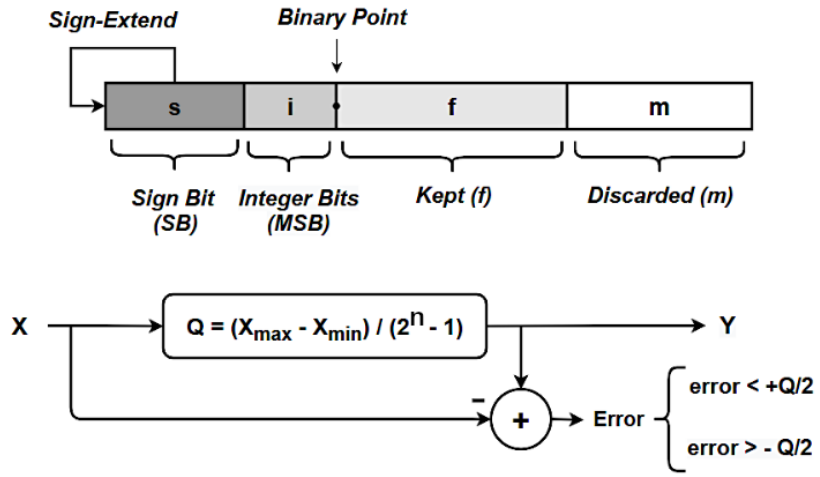

Figure 6.13: Binary rounding model for a finite word length.

As is shown in the illustration, the representation of a double-precision floating-point variable in a fixed-point value can result in accumulative noise depending on the number of assigned bits. The Linear Fitter receives its input from the preceding and succeeding modules in the Level-1 Tracking firmware chain. The parameters are provided in fixed-point representation with a finite number of bits. The output parameters of the Linear Fitter are also provided in the fixed-point binary format in compliance with the requirement for the Level-1 Track-Trigger system. The mathematical blocks in the Linear Fitter include addition, subtraction, multiplication, and reciprocal operations. These operations permit bit growth in their integer bits except for the reciprocal operator where growth occurs in the fraction bits. The most bit growth occurs in multiply/accumulate at runtime during the calculation of the regression fit. A larger fixed-point variable must be declared to accommodate the extended bits, negatively affecting the logic resource usage in the hardware implementation. To address this problem, the possible ranges of variables in both positive and negative directions before, during,

and after runtime are considered. The effects of defining the Linear Fitter signals with a finite number of bits are observed as quantisation noise that often leads to incorrect system output. More specifically, data path quantisation is the most challenging task in the Linear Fitter design. There are two new modules introduced in the Linear Fitter architecture. A rounder module to control the data path bit growth after the multiplication blocks, and a limiter module to saturate the data path if overflow is detected. The logic resources and latency in rounders and limiters must be kept to a minimum as their use in Linear Fitter design is frequent. Four types of round, truncate, convergent, and round-to-zero have been investigated for quantisation, and the effects of precision loss for roundoff errors have been examined. In the limiter module, a simple algorithm saturates the data path to maximum and minimum values permitted by the fixed point declaration. The characteristics of quantisation noise are presented in Table 6.1;

type	condition	r < Q/2	r = Q/2	r > Q/2
Round	-	round down	round up	round up
Truncation	-	round down	round down	round down
Convergent	s even s odd	round down round down	round down round up	round up round up
Round to Zero	$s \geq 0$ s < 0	round down round up	round down round up	round down round up

Table 6.1: Effects of rounding models in quantisation.

In Figure 6.14 the continuous line represents a portion of the reconstructed track, and the step-wise lines represent the new quantized track. As can be seen in the figure, the effects of the round and convergent methods are very similar with signal to noise ratio (SNR) of 1.8202. Also, the effects of the truncate and round to zero methods are similar with an SNR of -0.1315. A negative SNR means that signal power is lower than the noise power.

These different rounding approaches indicate that either round or convergent are good choices for hardware implementation in the Linear Fitter; however, the convergent use more logic gates. The method distinguishes between even and odd numbers using absolute and remainder operators. The rounding model uses fewer logic gates and is therefore selected for use in the Linear Fitter in hardware.

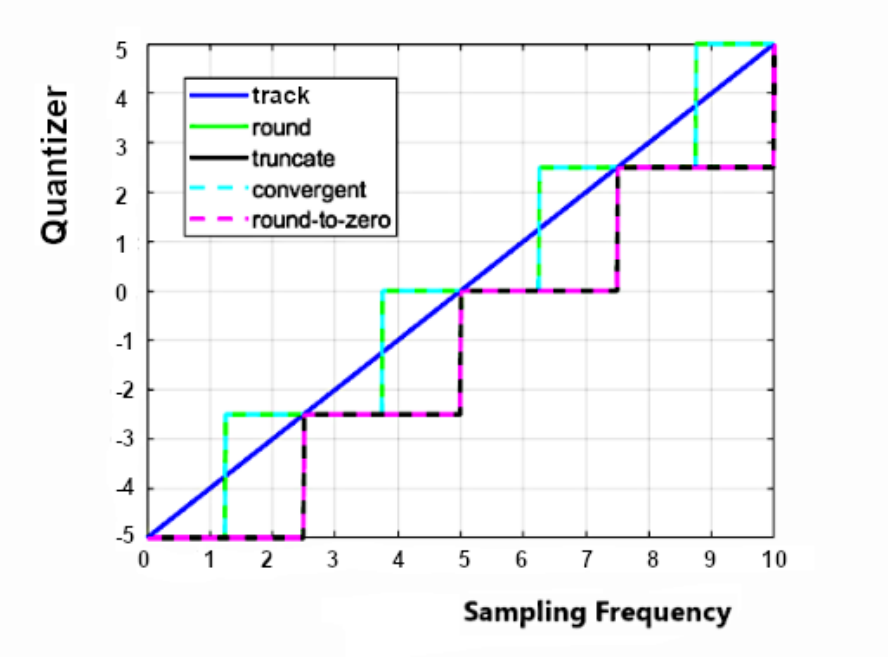

Figure 6.14: Quantisation error in a finite word length.

Investigating finite word length continues with developing a strategy for overflow. The effects of overflow in the representation of signals with a fixed number of bits can exceptionally force the value from positive to negative or vice versa. For example, a simple binary addition of 3 (b'011) + 1 (b'001) can result in -4 (b'1111) if there are only four bits are available. The most significant bit is reserved for the sign bit in fixed-point representation. The logic for overflow detection in limiter uses only NAND and NOR gates to clip the signal if going over the permitted range. Additional bits are required after the Most Significant Bit (MSB) as Guard bits. For positive numbers, the MSB ('0') is extended by n bits, while for negative numbers, the MSB ('1') is extended depending on the polarity of the output expectations. When the MSB bit and guard bits are not the same, the overflow is detected.

Input scaling also reduces the potential for bit growth, ensuring the dynamic range of signals is contained within their minimum and maximum binary presentations. In Linear Fitter, scaling effects are considered at the input and the intermediate nodes between inner modules. Several methods for signal level estimations are considered with the available data set. However, as the implementation of the Level-1 Tracking system is in its initial hardware implementation stage, the worst-case scenario is assumed in the simulation, and the effects of choosing the best scaling factor are studied. If the scaling factor is too small, the quantisation noise will become the dominant factor overpowering all signals. If the scaling factor is too large, it leads to an overflow in the accumulators. Every computational module in the Linear Fitter has a scaling factor b for each input signal.

For calculating a scaling factor for the worst-case scenario, all variables are replaced with their minimum and maximum in double-precision floating-point to carry out arithmetic operations in turn. The same is done for the variables using the fixed-point representations. The absolute difference between these two is obtained for all locations in the design. The scaling factor is used to minimise the difference for every stage or as the summation of all factors per signal at the input as shown in Equation 6.20:

$$\boldsymbol{b} = \sum_{\boldsymbol{i}=0}^{\boldsymbol{n}} \left| (\boldsymbol{Min/Max})_{\boldsymbol{node,\ stage_{(i)}}} \right| \tag{6.20}$$

Figure 6.15 shows the impact of the scaling factors in the Linear Fitter for signals r, ϕ and z. The unbroken lines represent the input signals at runtime, where dotted lines represent the signals after applied rounders, limiters and scaling factors. If the signal is deemed as too large for its corresponding fixed-point number after the first arithmetic operation, the algorithm will detect it as an overflow and clip the signal. As it can be seen in the first iteration, saturation logic detects an overflow, observed as a long drop in two signals, and forces the values to be within their declared fixed-point range.

The overflow detection algorithm prevents further overflow in the second and third iterations.

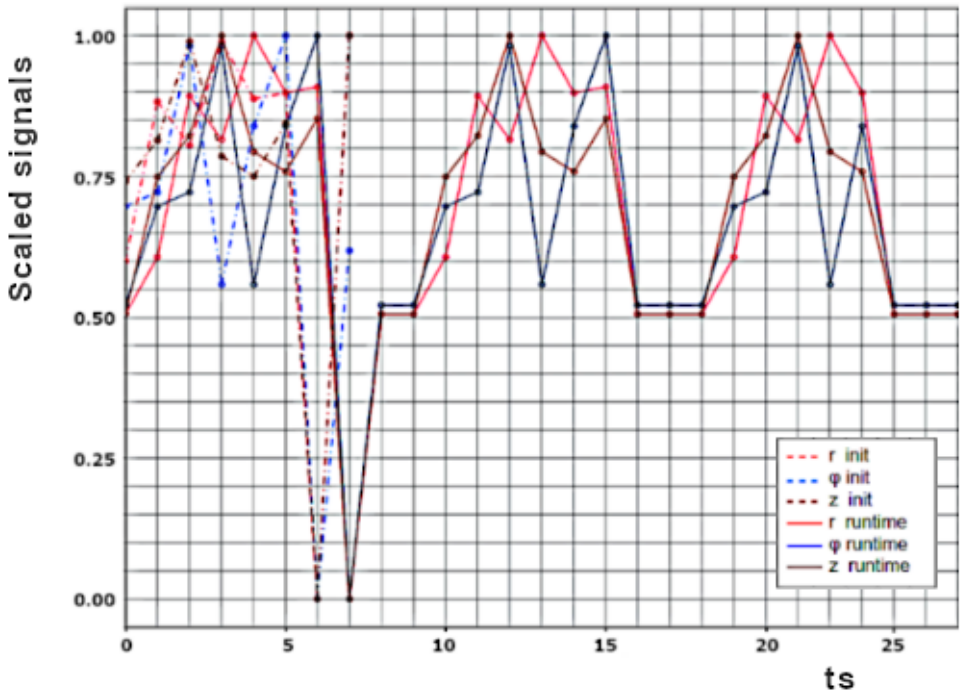

Figure 6.15: Signal scaling in the Linear Fitter.

6.12 Data Format in Linear Fitter

The Tracker segmentation defines the Linear Fitter data format in the DTC, GP and HT in TMTT implementation. The scheme also applies to the Tracklet module in the Hybrid configuration. Knowing the ranges of the parameters is crucial for the hardware implementation of the Linear Fitter, particularly in the fixed-point implementation of the design. The aim is to identify the absolute values seen by each module to develop strategies for handling roundoff errors and overflow.

In Table 6.2, the ranges for r (barrel) and z (endcaps) are given. There are Endcaps disks on both sides of the Barrel, resulting in positive and negative ranges.

Barrel	r (cm) range	Endcap	|z|(cm) range
Layer 1	21.8947 < r < 28.3838	Disk 1	128.9539 < |z| < 133.4292
Layer 2	34.7610 < r < 40.3326	Disk 2	152.7622 < |z| < 157.2374
Layer 3	49.9320 < r < 55.3471	Disk 3	183.1043 < |z| < 187.5795
Layer 4	66.9827 < r < 70.5633	Disk 4	219.3985 < |z| < 223.8289
Layer 5	84.2795 < r < 87.8377	Disk 5	262.7635 < |z| < 267.2387
Layer 6	106.5885 < r < 110.1020	-	-

Table 6.2: Ranges of r and |z| in Tracker barrel and endcaps.

The ranges for ϕ are determined by the tracker segmentation in the DTC module firmware configuration according to the centre of the stub sectors and their boundaries in the GP module. The stub information for the GP module is assigned to a 64-bit stub word containing coordinates (r, ϕ, z) and (ϕ, η) sectors that the stub originated from. The stub word contains minimum and maximum ranges for HT cells along with the layer identifier where the hit was initially detected. The MSB of the stub word customarily is reserved for hit validation. The MSB bit set to '1' indicates that the stub word contains information that is useful for the track reconstruction. As the reserved bits are declared for new variables or extending the ranges of the current variables. Tracker parameters r and z are represented by their integer-based values using custom-designed utility classes in the CMSSW by the Equation 6.21:

$$x \rightarrow x(1 << scale) \tag{6.21}$$

The *scale* is the number of fraction bits present in the word length. Equation 6.22 converts back the integer-based values to real-world values after processing:

$$x \rightarrow x/(1 << scale) \tag{6.22}$$

Table 6.3 presents the data format for the GP module in η and ϕ sectors.

parameter	unit	resolution	range min	range max	bits	signed	bit range
ϕ	rad	$\Delta(\phi)$	$-\Delta(\phi) * 2^{13}$	$\Delta(\phi) * (2^{13} - 1)$	14	y	[0:13]
reserved	-	-	-	-	2	-	[14:15]
r	cm	$\Delta(r)$	$-\Delta(r) * 2^{11}$	$\Delta(r) * (2^{11} - 1)$	12	y	[16:27]
layer$_{ID}$	layer	1	0	1	3	n	[28:30]
reserved	-	-	-	-	1	-	[31:31]
z	cm	$\Delta(z)$	$-\Delta(z) * 2^{13}$	$\Delta(z) * (2^{13} - 1)$	14	y	[32:45]
reserved	-	-	-	-	2	-	[46:47]
mbin min	bin	1	-16	15	5	y	[48:52]
mbin max	bin	1	-16	15	5	y	[53:57]
η sector$_{ID}$	sub-sector	1	0	1	1	bool	[58:58]
ϕ sector$_{ID}$	sub-sector	1	0	1	1	bool	[59:59]
reserved	-	-	-	-	3	-	[60:62]
$stub_{valid}$	-	1	0	1	1	bool	[63:63]
total	-	-	-	-	64	-	-

Table 6.3: The data format is defined by the GP module.

Using local variables instead of global variables reduces the parameter ranges and their finite word length of the presentations significantly. If the Linear Fitter is linked to the HT module in TMTT firmware, the ϕ ranges are defined by Hough cells configuration. Table 6.4 shows the data format in the HT module.

parameter	unit	resolution	range min	range max	bits	signed	bit range
layer$_{ID}$	layer	$\Delta(\phi)$	0	7	3	n	[0:2]
η sector$_{ID}$	sector	1	0	17	5	n	[3:7]
cBin	bin	1	-16	15	5	y	[8:12]
z	cm	$\Delta(r)$	$-\Delta(z) * 2^{11}$	$\Delta(z) * (2^{11} - 1)$	12	y	[13:24]
ϕ	rad	$\Delta(\phi)$	$-\Delta(z) * 2^{8}$	$\Delta(z) * (2^{8} - 1)$	8	y	[25:33]
r	cm	$\Delta(z)$	$-\Delta(z) * 2^{11}$	$\Delta(z) * (2^{11} - 1)$	12	y	[34:45]
mSel	bin	1	0	1	1	bool	[46:46]
reserved	-	-	-	-	2	-	[47:62]
stub$_{valid}$	-	-	0	1	1	bool	[63:63]
total	-	-	-	-	64	-	-

Table 6.4: The data format is defined for the HT module.

In the TMTT configuration, the stubs are formatted into a 64-bit stub word with additional parameter definitions for HT cells. The ϕ sector is calculated from the ϕ ranges from the GP module. Similar to the GP data format, the stub coordinate (r, ϕ, z) and its corresponding η sector with a uniquely defined layer identifier are assigned to the stub word. The cBin holds information for the HT cell with the MSB bit assigned for the stub validation. The reserved bits can be used to extend the ranges of the current parameters.

The Linear Fitter operates on stubs processed by the GP module and additional information from the DTC module. The stub information required by the Linear Fitter contains ϕ, r, z, PS, 2S, layer$_{ID}$, valid parameters. The GP block provides ϕ, r, z, layer$_{ID}$ and valid parameters. However, the information for PS and 2S are obtained from the DTC separately. The track parameters q/p_T (r$_{r\phi}$), ϕ_T ($\phi_{r\phi}$), cotθ_T (r$_{rz}$), and z$_T$ (z$_{rz}$) are calculated at the runtime along with the gradient and the intercept in the r-ϕ and r-z plane. All the parameters are concatenated into a 96-bit track word. The validity of the track is tested against the MSB. The variable naming depends on the module arrangements and deployment of the HT module in the chain. In HT, the gradient in r-ϕ is defined as q/p_T and the intercept as ϕ_T concerning the chosen offset. The gradient in r-z is defined as cotθ_T and its corresponding intercept as z$_T$ concerning the centre of the stub sector. The data format presented in the Table 6.5 is used to unpack the stub information into individual parameters in preparation for Linear Fitter arithmetic operations.

parameter	unit	resolution	range min	range max	bits	signed	bit range
r	cm	$\Delta(r)$	$-\Delta(r) * 2^{11}$	$\Delta(r) * (2^{11} - 1)$	12	y	[0:11]
ϕ	rad	$\Delta(\phi)$	$-\Delta(\phi) * 2^{13}$	$\Delta(\phi) * (2^{13} - 1)$	14	y	[12:25]
z	cm	$\Delta(z)$	$-\Delta(z) * 2^{13}$	$\Delta(z) * (2^{13} - 1)$	14	y	[26:39]
layer$_{ID}$	layer	1	0	7	3	n	[40:42]
PS module	-	1	0	1	1	bool	[43:43]
2S module	-	1	0	1	1	bool	[44:44]

Table 6.5: Stub word 45-bit format in the Linear Fitter module.

The data format presented in the Table 6.6 is used to unpack the track information into individual parameters before Linear Fitter arithmetic operations.

parameter	unit	resolution	range min	range max	bits	signed	bit range
q/p_T	cm	$\Delta(r)$	$-\Delta(rT) * 2^{11}$	$\Delta(r) * (2^{11} - 1)$	12	y	[45:56]
ϕ_T	rad	$\Delta(\phi)$	$-\Delta(\phi T) * 2^{13}$	$\Delta(\phi T) * (2^{13} - 1)$	14	y	[57:70]
cotθ_T	-	$\Delta(tan\lambda)$	$-\Delta(tan\lambda) * 2^{13}$	$\Delta(tan\lambda) * (2^{13} - 1)$	14	y	[71:84]
z_T	cm	$\Delta(zT)$	$-\Delta(zT) * 2^{9}$	$\Delta(zT) * (2^{9} - 1)$	10	y	[85:94]
$stub_{valid}$	-	1	0	1	1	bool	[95:95]
total	-	-	-	-	96	-	-

Table 6.6: Track word 96-bit format in the Linear Fitter module.

The data format for the parameters at the runtime are $r_{r\phi}$, $\phi_{r\phi}$, $gradient_{r\phi}$, $intercept_{r\phi}$ in the $r-\phi$ plane and r_{rz}, ϕ_{rz}, $gradient_{rz}$, $intercept_{rz}$ in the r-z plane with defined ranges and word lengths as shown in Table 6.7 and 6.8. The resolution is defined by the number of bits present in the fractional part of the word length. The extra number of bits assigned for the r-z plane is necessary to encode the whole z range and to utilise the chosen offsets. The values for the offsets are converted to their fixed-point representation in advance and stored in intermediate lookup tables acting as memory blocks. There are always trade-offs between bit assignments and efficiency, resulting in the trade-off between logic resource usage and performance in the physics simulation.

parameter	unit	range min	range max	bits	resolution
$r_{r\phi}$	cm	$-\Delta(r) * 2^{9}$	$\Delta(\phi) * (2^{9} - 1)$	10	0.1195
$\phi_{r\phi}$	rad	$-\Delta(r) * 2^{7}$	$\Delta(\phi) * (2^{7} - 1)$	8	0.0391
$gradientr\phi$	rad/cm	$-\Delta(r) * 2^{5}$	$\Delta(r) * (2^{5} - 1)$	6	0.0156
$intercept_{r\phi}$	rad	$-\Delta(r) * 2^{6}$	$\Delta(\phi) * (2^{6} - 1)$	7	0.0078
$residual_{r\phi}$	rad	$-\Delta(r) * 2^{5}$	$\Delta(\phi) * (2^{5} - 1)$	6	0.0156

Table 6.7: Runtime variables and ranges are declared for the r-ϕ plane.

The information is accessible during processing and immediately after the last stub in the stub stream is cleared.

parameter	unit	range min	range max	bits	resolution
r_{rz}	cm	$-\Delta(r) * 2^9$	$\Delta(\phi) * (2^9 - 1)$	10	0.1195
z_{rz}	cm	$-\Delta(z) * 2^{10}$	$\Delta(z) * (2^{10} - 1)$	11	0.00048
$gradient_{rz}$	1	$-\Delta(r) * 2^{11}$	$\Delta(\phi) * (2^{11} - 1)$	12	0.00024
$intercept_{rz}$	cm	$-\Delta(r) * 2^{11}$	$\Delta(\phi) * (2^{11} - 1)$	12	0.00024
$residual_{rz}$	cm	$-\Delta(r) * 2^6$	$\Delta(\phi) * (2^6 - 1)$	7	0.0078

Table 6.8: Runtime variables and ranges are declared for the r-z plane.

In the following clock transition, the parameters are passed to external memory blocks and are no longer visible to the Linear Fitter module. The track parameters for this stage are assigned to a 64-bit track word and shown in Table 6.9.

parameter	unit	range min	range max	bits	signed	bit range
$gradient_{r\phi}$	1/cm	$-\Delta(\phi) * 2^{13}$	$\Delta(\phi) * (2^{13} - 1)$	12	y	[0:11]
reserved	-	-	-	2	-	[12:13]
$intercept_{r\phi}$	rad	$-\Delta(r) * 2^{17}$	$\Delta(r) * (2^{17} - 1)$	16	y	[14:29]
reserved	-	-	-	2	-	[30:31]
$gradient_{rz}$	1	$-\Delta(r) * 2^{13}$	$\Delta(r) * (2^{13} - 1)$	8	y	[32:39]
reserved	-	-	-	2	-	[40:41]
$intercept_{rz}$	cm	$-\Delta(r) * 2^9$	$\Delta(r) * (2^9 - 1)$	12	y	[42:53]
reserved	-	-	-	9	-	[54:62]
$track_{valid}$	-	1	0	1	bool	[63:63]
total	-	-	-	64	-	-

Table 6.9: Data-path packing in Linear Fitter runtime.

The reserved bits are pre-allocated for future modifications in tracker geometry. The extra bit allows the extension of the variable length by two bits. The added length used in integer or fraction bits increase the resilience to bit overflow or improves fractional precision.

This page was intentionally left blank.

Chapter 7

Linear Fitter Firmware

7.1 Linear Fitter Firmware Implementation

In the hardware implementation of the Linear Fitter module, the focus is on the optimisation of the resource usage and latency by simplification of the track reconstruction algorithm and custom-designed hardware. The key parameters for hardware optimisation include the number of components utilised, execution time, memory requirements, power consumption and resource allocation on an FPGA chip. There are several techniques used to achieve an optimal design such as simplification of arithmetic operations in the track reconstruction phase by studying alternative solutions to break down the necessary equations and replacing them with primitive logic components. In higher level abstraction, all application-specific data flow and digital signal processing blocks are carefully custom-designed. The CMS experiment has commissioned the purchase of the 16 nm FinFET, Xilinx Virtex Master Processor, and FPGA chip *(XC7VX690T)* for its capability with computationally intensive algorithm and high-performance high-throughput characteristics [103]. An alternative FPGA chip Kintex UltraScale from Xilinx is also being tested for compatibility with the Linear Fitter for future scalability. The Kintex family are relatively new in comparison to the Virtex devices with more on-chip memory and logic slices. In a fully pipelined implementation, the latency is defined as the time consumed until the last valid output has exited the last block of the design. An important metric in latency management is the Initiation Interval (II), which defines a module's timing before accepting new input [104]. Typically, in DSP modules three

actions read, process, and write occur repeatably which is described as reading data from memory, processing the data and writing the data back to the memory. If there is only one copy of the data processor available, the data has to be fed to it sequentially, however, with more copies of the same processor, the data can be scheduled for processing in parallel. Multiple read, process, and write are sent to multiple copies, one clock phase apart. In Figure 7.1, the timing definitions are shown for three copies of the processing units. The top-level design can accept a new input at every clock cycle or the initiation interval of one. Other quantities such as loop latency, loop Initiation Interval, and trip counts are also shown. FPGA are effectively efficient in providing logic slices for custom-designed data processors.

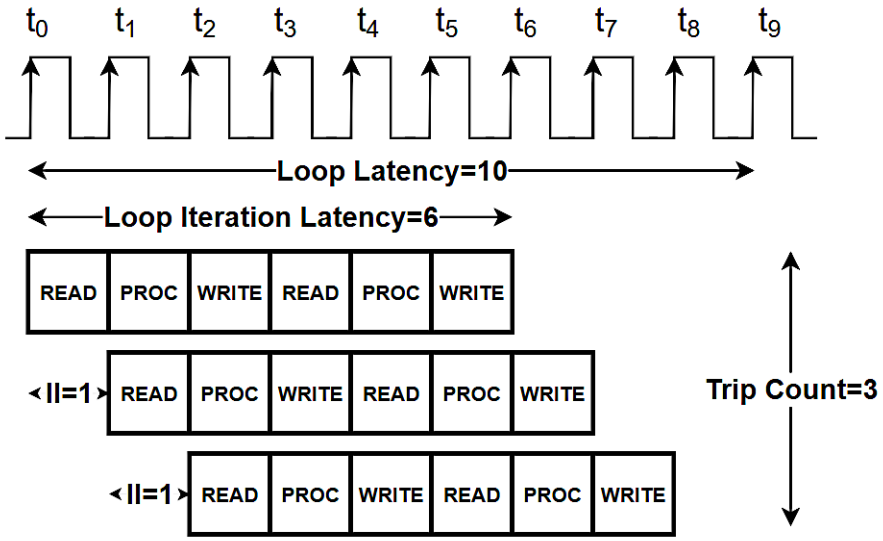

Figure 7.1: Timing definitions in a fully pipelined design.

In the Linear Fitter design, the modules are divided into sub-modules with the strategy of only transferring the required data to different sub-modules at the appropriate time. If data is not needed in a given clock cycle, the corresponding logic components are kept idle to save resources and processing power. The data dependencies in the sub-modules are carefully considered to eliminate critical paths and increase throughput. The critical path is the path with the longest latency. Vivado Design Suite [105] provides tools for measuring the longest path. Once the path is identified, alternative logic mapping can be used to change the flow of data. In uncomplicated

scenarios, often delaying the path with the use of clocked latches can divide the longest path into two clock cycles. In complicated scenarios, the longest path contributes to the total latency in the system, if not modified. In Figure 7.2, the pipelined architecture with paralleled structure is shown. In section a of the diagram, the Linear Fitter is displayed as a filter module with two memory blocks for storing stubs and tracks, and feedback to the stub RAM to facilitate the iterations if more than one is needed.

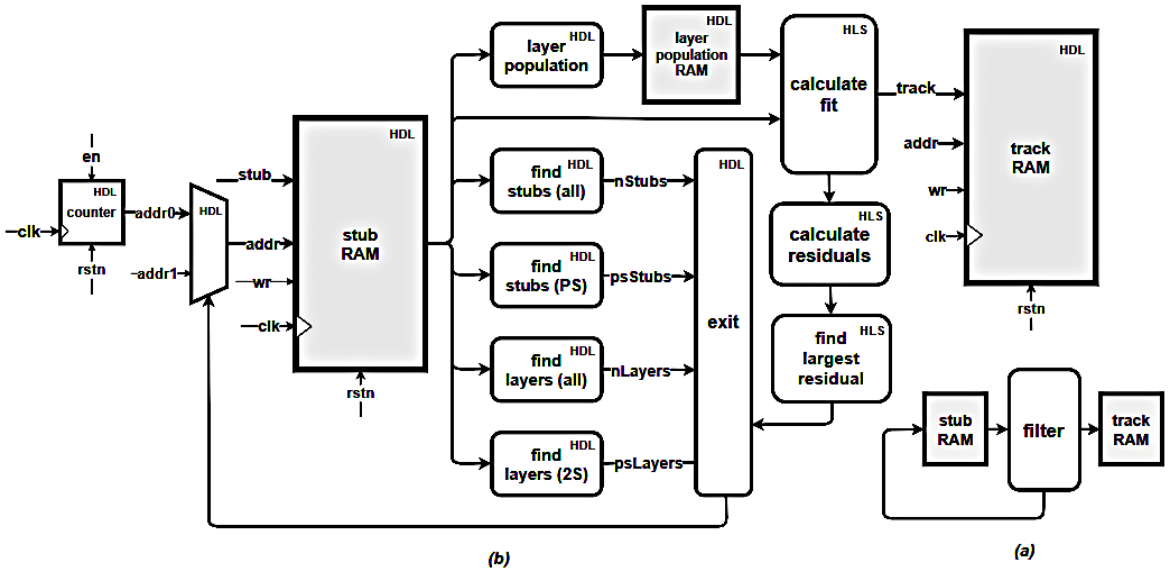

Figure 7.2: Linear Fitter in a fully pipelined recursive design.

The expanded overview of the hardware structure is shown in section b of the diagram with paralleled sub-modules. A program counter is used to increment the memory addresses in the stub RAM and track RAM with multiplexing logic to either write to a new location or read from an existing location in stub RAM. Once a stub with the largest residual is found, it is removed from memory by inverting the stub word valid signal to invalidate it rather than clearing the memory location, which is costly in hardware. The structure also permits recursive realisation of a fully pipelined logic rather than placing n number of copies in the hardware chain. A slice of the lookup table stores the current layer populations at runtime instead of RAM. The exit block decides when the criteria for a genuine track candidate have been satisfied and accordingly terminates all processes.

In the first design iteration, all Linear Fitter modules are implemented in HDL and HLS. However, in the final version, sub-modules are categorised into two types: data flow and arithmetic blocks. The data flow modules, including layer population, find stubs (PS), find layers (PS), find stubs (2S), find layers (2S), and exit, are implemented in HDL. The arithmetic blocks including calculate fit, calculate residual and find largest residual are implemented in HLS hardware accelerators and integrated into the architecture as HLS IP cores. In this configuration, designing in HDL has provided wider control over the logic structure and behaviour of the system in the RTL architecture. The hardware acceleration tool has permitted pipelining of the design regions where HDL-only development did not. The compatibility of HLS with the CMS software framework has also helped with the verification and debugging stages in the physics performance evaluations.

An alternative fully pipelined implementation of the Linear Fitter algorithm has also been designed in HDL to allow comparison with the development in HLS. The logic level implementation of the Linear Fitter explicitly describes the behaviour of the circuits for one iteration only. The process begins with streaming the stub parameters in buffers with a timing interval of one stub in, and one stub out per clock. The layer information is used to identify stubs on the different tracker layers to generate virtual stubs. All the arithmetic blocks such as multiply/accumulate, multipliers, adders, and reciprocal slices are custom designed and tested in preparation for the system. The structure is concatenated to its copies to model the recursive behaviour of HLS design as there is no feedback loop in this architecture. Therefore, one iteration can only find and eliminate one stub.

The Linear Fitter arithmetic operations are divided into four blocks: one block (top-level) is the wrapper for the other three blocks Update Sums, Update Fit and Calculate Residuals required to compute the trajectory estimations. Four BRAM of (24,1) dimensions are dedicated to holding stub information (ϕ, r, z, layer$_{ID}$) and mapping them according to the layer in which they were first detected. The implementation follows the configuration of streams of sixteen stubs per processing nonant, covering all seven layers with a maximum of four stubs per layer. This configuration covers all scenarios in the $t\bar{t}$ pileup events.

In Figure 7.3, the distribution of average stub count (ranging from 0 to 280) per tracker layer is shown for 100 $t\bar{t}$ pileup events.

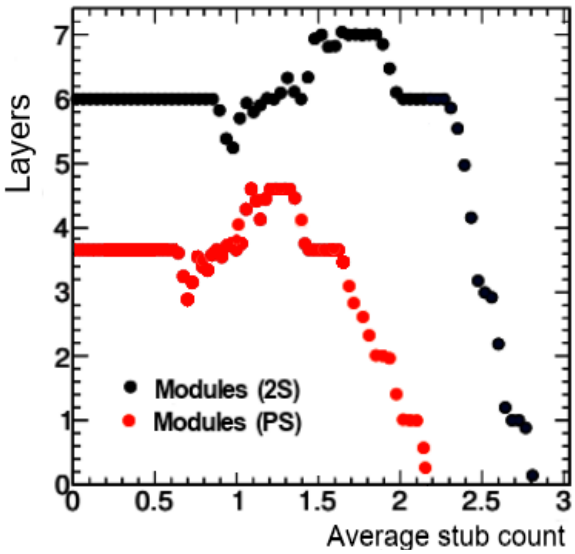

Figure 7.3: Average stub count per tracker layers per event.

The particles are divided into two categories according to the p_T modules (2S and PS) appearing on seven layers of the tracker with a maximum of three stubs per layer. The stub mapping can be presented by a two-dimensional array of seven rows and three columns. The stub map configuration in Linear Fitter allows for higher stub occupancy in the future, hence an extra column is added to the stub map array to accommodate a possible fourth stub on each layer. To create a virtual stub per layer, the algorithm must know the correct mapping of stub occurrences on all layers. The information is particularly used in layers with no stub occupancy to avoid divide-by-zero in the fit arithmetic operations when calculating the average stubs on each layer. The mapping uses a custom-designed address generator to store sixteen stubs according to their corresponding layers. The implementation avoids nested loops and complicated logic for the cross-conversion from one-dimensional to two-dimensional array and vice versa. In this configuration, the rows hold information for the layers and the columns hold the information for stub occurrences on each layer. The indexes are calculated by a multiplier, an adder and two simple up-counters.

The mapping model is shown in Figure 7.4.

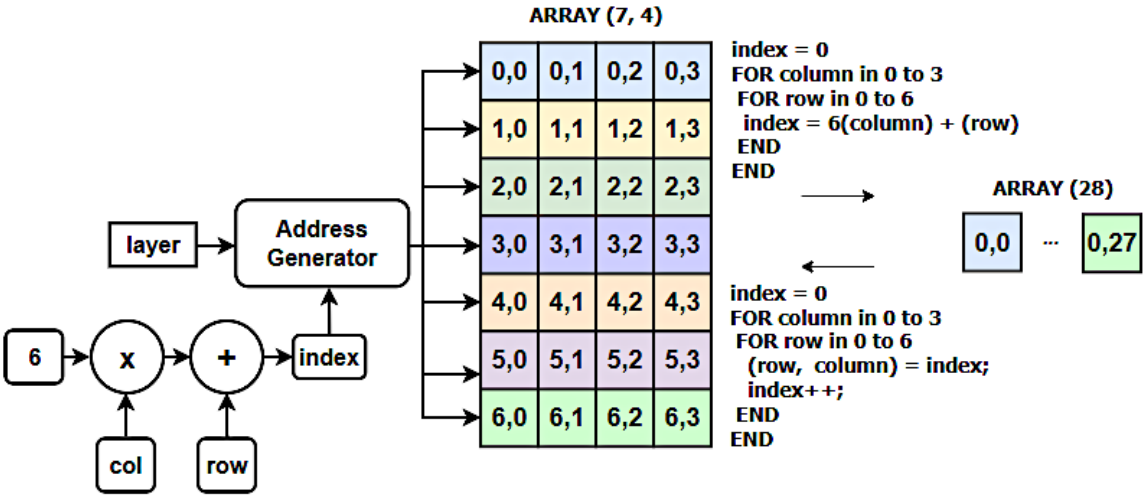

Figure 7.4: Converting 2D layer mapping to 1D mapping and vice versa.

The cost of array conversion producing a 1 x 28 array increases as a result of the additional logic resources. Yet, the advantages are more significant in removing synthesis issues for generating two-dimensional mapping algorithms using ready-built libraries such as std::map class in *C++ STL* [106]. The FIFO blocks used in the single iteration design operate on ϕ, r, z and layer identifiers. The configuration enforces a fully pipelined design for one stub per clock cycle. At the rising edge of the clock, the stubs are moved to ϕ, r, z and layer FIFO blocks and shifted forward one location per clock cycle. The layer population mapping is implemented in a LUT block to determine the correct exit condition when the desired filtering is achieved. The variable means are calculated with primitive logic gates and adder slices for the $r-\phi$ and $r-z$ planes separately. The gradient and intercept are computed with the MAC and reciprocal slices, eliminating the use of DSP48E1 blocks [107] to save resources. The offsets for ϕ and z are incorporated into the structure as constants, and the total residual for the elements (ϕ_{resid} + z_{resid}) are calculated to return the index to the stub memory with the largest distance from the fit. The valid bit of the stub word in memory is set to zero to indicate the stub is not part of the reconstructed track any longer.

In the Figure 7.5, an overview of the Linear Fitter single iteration architecture is shown.

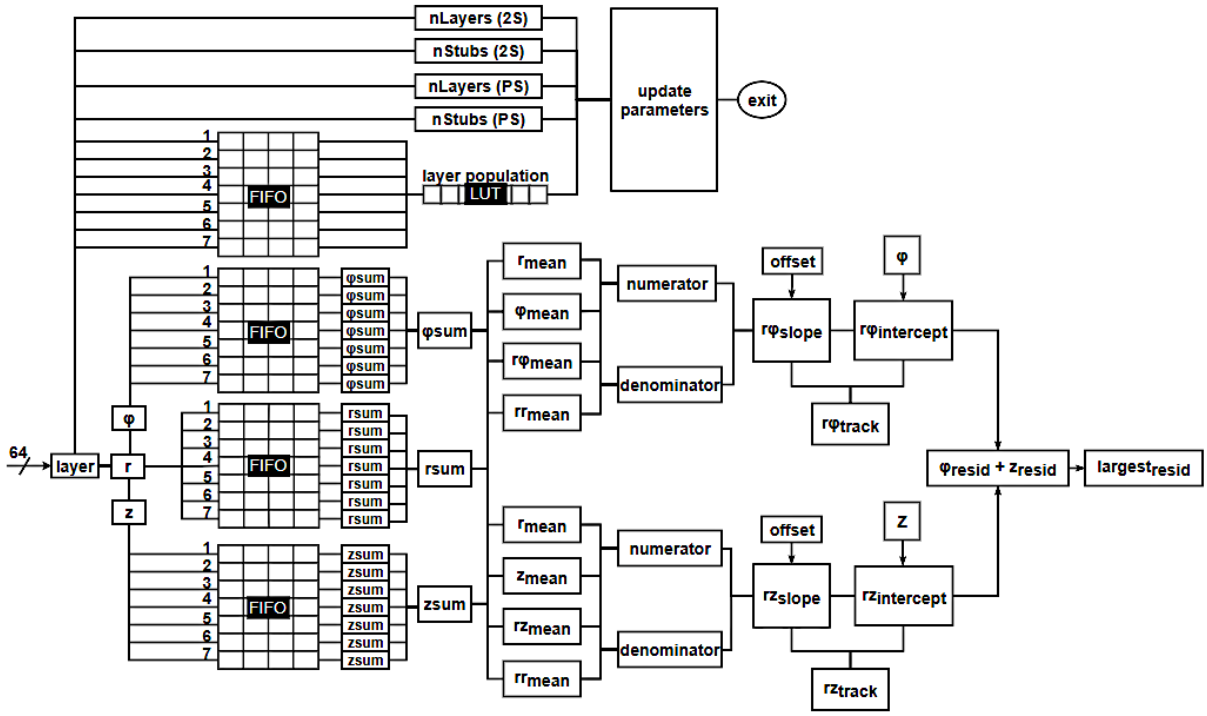

Figure 7.5: Single iteration RTL implementation of Linear Fitter.

The communication between logic components and memory modules is implemented using simple (start, done) signals in the top-level module. The clock signals ($\varphi 1$ and $\varphi 2$) are generated from half the system clock, each one phase apart. The logic and arithmetic processor is clocked using $\varphi 1$ and the dual-port RAM block is clocked on $\varphi 2$ to allow one read and write per clock cycle. When the data become available, the start signal is raised which is the indication for the arithmetic processor to commence transition. Once the processor has completed its operation, it will raise the done signal, which is an indication to the top level to reset the start signal and prepare for the next operation. In addition to two-phase clocking, extra registers are used to enforce pipelining. The communication protocol is necessary to ensure that one processing module completes its operations before accepting new input.

Figure 7.6 shows the arithmetic block with the communication protocol and clock transitions for stub transmission.

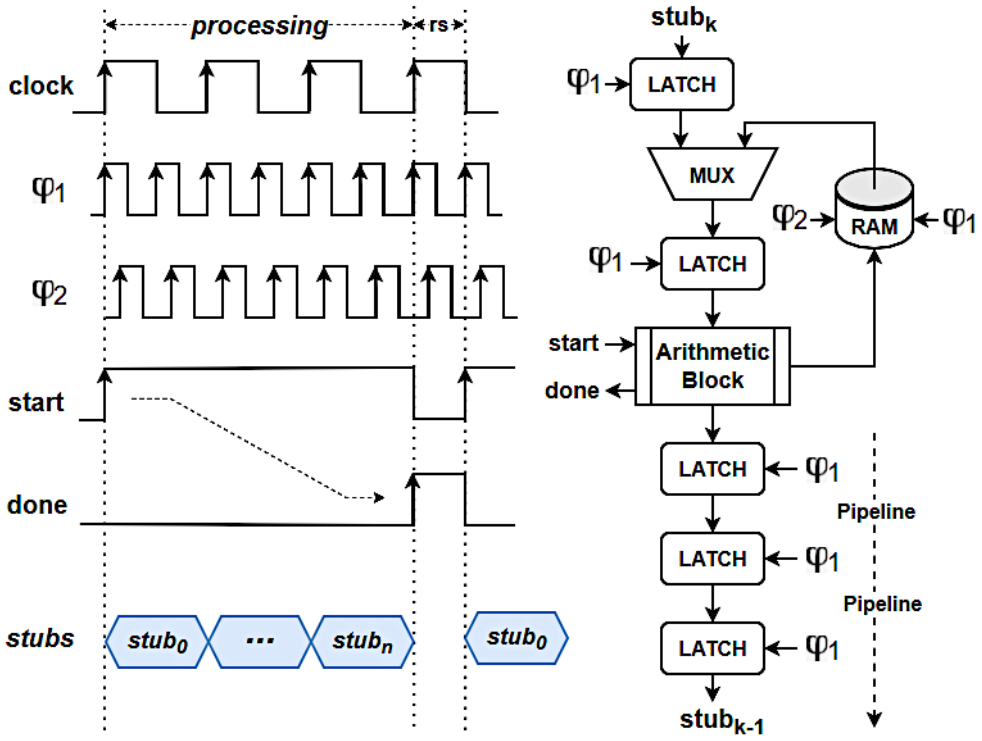

Figure 7.6: Simple communication protocol for Linear Fitter.

The next step in the hardware implementation is to study the input/output parameter ranges for fixed-point instrumentation. When converting a double-precision floating-point value to a fixed-point, the optimal fixed-point data types must meet the instrumentation and acceleration constraints to satisfy system numerical accuracy requirements. Closely monitored parameters are ϕ, r, z for input/output stub parameters along with generated output track variables gradient$_{r\phi}$ (q/p_T), intercept$_{r\phi}$ (ϕ_T), gradient$_{rz}$ ($cot\theta_T$) and intercept$_{rz}$ (z_T). In addition, the intermediate fit parameters and effects of finite word length representation at runtime are considered carefully. The variable ranges are determined by using the simulated data from $t\bar{t}$+200PU generated events for the D49 tracker geometry. Table 7.1 shows the variable ranges seen by the top-level Linear Fitter with word length, fractional length, and proposed fractional length. In scenarios where the fraction length is less than or equal to the proposed length, no action is required as the effects of precision loss and overflow have been

accounted for. When the proposed fraction length is greater than the current fraction length, additional consideration must be made to reach an optimal value that minimises the error between planned HLS fixed-point variables and HDL implementation. The component class determines if the parameter is a scalar (1 x 1) sample-based or a vector (1 x 16) stream-based. The declared quantities including sim min/max define the parameter ranges assigned to the variables during the simulations.

parameter	type	size	WL	FL	PFL	sim min	sim max
ϕ_i	input	16x1	14	11	11	$-0.7250(2^2)$	$+0.6000(2^2)$
r_i	input	16x1	12	4	4	$-0.8602(2^7)$	$+8577(2^7)$
z_i	input	16x1	14	4	4	$-0.5220(2^9)$	$+0.9796(2^8)$
gradient$_{r\phi}$	output	1x1	12	8	16	$+0.8579(2^{-5})$	$+0.9359(2^{-5})$
intercept$_{r\phi}$	output	1x1	14	8	20	$-0.9575(2^{-9})$	$+0.9378(2^{-7})$
gradient$_{rz}$	output	1x1	14	8	11	$+0.6514(2^2)$	$+0.7336(2^2)$
intercept$_{rz}$	output	1x1	12	10	11	$+0.6087(2^{-2})$	$+0.8269(2^0)$
ϕ_o	output	16x1	14	11	11	$-0.7250(2^2)$	$+0.6000(2^2)$
r_o	output	16x1	12	4	4	$-0.8602(2^7)$	$+8577(2^7)$
z_o	output	16x1	14	4	4	$-0.5220(2^9)$	$+0.9796(2^8)$

Table 7.1: Linear Fitter top-level fixed-point variables.

In Table 7.2, the run-time parameters are shown with their word length declarations of integer bits and fraction bits including their proposed fraction bits based on the observed ranges during the simulation of $t\bar{t}$+200PU events.

parameter	type	size	WL	FL	Proposed FL	sim min	sim max
b_{scale}	local	1x1	5	0	0	$+0.0000(2^0)$	$+1.0000(2^5)$
$x_{r\phi}$	local	1x1	18	8	8	$+0.0000(2^0)$	$+0.7221(2^9)$
$y_{r\phi}$	local	1x1	18	13	13	$-0.8000(2^{-3})$	$+0.7375(2^4)$
$xx_{r\phi}$	local	1x1	18	1	1	$+0.0000(2^0)$	$+0.5848(2^16)$
$xy_{r\phi}$	local	1x1	18	7	6	$-0.6489(2^{-1})$	$+0.5306(2^11)$
x_{rz}	local	1x1	18	8	8	$+0.0000(2^0)$	$+0.7221(2^9)$
y_{rz}	local	1x1	18	6	6	$+0.0000(2^0)$	$+0.5669(2^11)$
xx_{rz}	local	1x1	18	1	1	$+0.0000(2^0)$	$+0.5848(2^16)$
xy_{rz}	local	1x1	18	0	0	$+0.0000(2^0)$	$+0.8111(2^17)$

Table 7.2: Proposed fixed-point variables with a scaling factor.

In calculating the mean of variables, the logic is developed to receive stubs within the stub sector boundaries to create virtual stubs per layer. According to the fit equations given in the previous Chapter 6.3, the parameters are placed in local variables declarations in the fit calculations. As the fit is generated in two planes, two sets of local variables are declared to hold the parameters x, y, xx and xy in both planes. The xx variable holds the result of the multiplication of the stub radius, and xy holds the multiplication of the stub radius r with the local angle ϕ. In the r-z plane, the xy holds the multiplication result of stub radius r and the longitudinal stub parameter z. In the update fit block, the algorithm expects the largest bit growth as the parameter summations in the sum update block are reused in the fit calculations. Hence, an additional parameter is added as the scaling factor to scale down the variables to contain them within a finite word length with fewer integer bits.

The effects of using different values for the scaling factor are considered, and the optimal value is obtained for a maximum of an 18-bit word length in fit calculation and a 20-bit in the residual arithmetic operations. The radix_2 classification of the scaling factor is significant in replacing the division operator with the bitwise shifting operations, given in Equation 7.1.

$$\frac{x}{b} = \overbrace{x \gg log_2(b)}^{SHIFT\ X} = \begin{cases} b & : b \geq 2 \\ b & : (b)_2 \end{cases} \tag{7.1}$$

In the update fit section of the design, the scaled-down variables calculate the gradient and intercept in two planes. The reciprocal hardware slice is reused to implement the division operator in two stages of multiplication and reciprocal operations. The accumulator in 18-bit reciprocal needs to be two bits wider because the remainder becomes unfitting. For example, when we divide eight by nine, the remainder should be eight 4'b1000, but without the wider accumulator the left-most digit would be lost, and the remainder would appear to be 4'b0000.

Table 7.3 shows the local variables and their fixed-point integer and fraction bits with their proposed fraction bit counterparts.

parameter	type	size	WL	FL	Proposed FL	sim min	sim max
$gradient_{r\phi}$	local	1x1	12	8	16	$+0.0000(2^{0})$	$+0.9359(2^{-5})$
$intercept_{r\phi}$	local	1x1	14	8	20	$-0.9575(2^{-9})$	$+0.9378(2^{-7})$
$1/denom_{r\phi}$	local	1x1	18	1	3	$+0.0000(2^{0})$	$+0.5077(2^{14})$
$gradient_{rz}$	local	1x1	14	8	11	$+0.0000(2^{0})$	$+0.7336(1^{2^{2}})$
$intercept_{rz}$	local	1x1	12	10	11	$+0.0000(2^{0})$	$+0.8269(1^{2^{0}})$
$1/denom_{rz}$	local	1x1	18	1	3	$+0.0000(2^{0})$	$+0.5077(1^{2^{14}})$

Table 7.3: Proposed fixed-point variables for fit.

The sim max and min values are the errors expected with the current fixed-point instrumentation. As observed from the proposed fractional bits, the precision of the variables must be increased for both gradients in both planes. A similar approach is adopted in the largest residual block. The generated update fit block parameters are used to calculate all stub residuals and identify the stub with the largest residual in ϕ and z. It can be seen in Table 7.4, that except for the residual$_{\phi}$, there are disagreements between fraction length and proposed length for defined variables. The effects of precision loss in these variables are considered, along with adjusting the scaling factor to minimise the effects. It is worth noting that some disagreement between these variables is expected in all stages of the hardware implementation due to the fixed-point representation of double-precision native variables.

parameter	type	size	WL	FL	Proposed FL	sim min	sim max
residual$_{idx}$	local	1x1	5	0	0	$+0.0000(2^{0})$	$+1.0000(2^{5})$
residual$_{\phi}$	local	1x1	18	17	17	$+0.0000(2^{0})$	$+0.8563(2^{1})$
residual$_{z}$	local	1x1	18	18	11	$+0.0000(2^{0})$	$+0.5605(2^{7})$
residual$_{\phi z}$	local	1x1	18	15	11	$+0.0000(2^{0})$	$+0.5739(2^{7})$
residual$_{\phi}$ >	local	1x1	18	13	17	$-0.5000(2^{1})$	$+0.8563(2^{1})$
residual$_{z}$ >	local	1x1	18	13	11	$-0.5000(2^{1})$	$+0.5605(2^{7})$

Table 7.4: Proposed fixed-point variables for residuals.

The output of the largest residual block is an index to the stub location in the RAM. Should the find largest residual block provides a valid index, the valid signal of the stub word in memory is negated to indicate the stub is invalid. The bookkeeping block prioritises reading the valid bit of the stub word. If the test returns an invalid bit, it is an indication that the tracking process must stop. The bookkeeping block raises a flag to allow valid stubs to be read from the memory. All valid stub information in the stub RAM is considered as genuine hit candidates generated by a single particle, hence returning a genuine track.

7.2 All High-Level Synthesis Tracking System

Until recently, all firmware applications for LHC trigger and tracking systems were implemented in VHDL [108] and Verilog [109]. The algorithm was initially designed and tested in CMSSW using simulated data generated for the high luminosity upgrade of the CMS detector. At this stage, compatibility, robustness, and algorithm performance for validation are obtained using physics simulations. The next stage involves designing a standalone version of the design in preparation for hardware implementation, which involves eliminating dynamic data allocations and libraries that are not synthesizable in FPGAs. A standalone version of the design is implemented in HLS with several testbench entities verifying the accuracy of the outputs.

The verification also invokes *C++/RTL* co-simulation [56] to ensure the behaviour of the software and the generated hardware are identical. Once the design has passed the desired latency and resource usage tests, a logic synthesis of the implementation is exported as a self-contained IP core for integration into the Level-1 Tracking firmware architecture. The integration processes are described in Chapter 8. The HLS libraries allow automatic transformation from software to hardware synthesisable code that is programmed into FPGAs. Even though the task of programming logic gates using low-level languages has been reduced in coding complexity, the HLS language is somewhat different to *C++* language when targeting specific latency and resource usage.

In Figure 7.7, the process involved in the high-level design of the Linear Fitter HLS is shown.

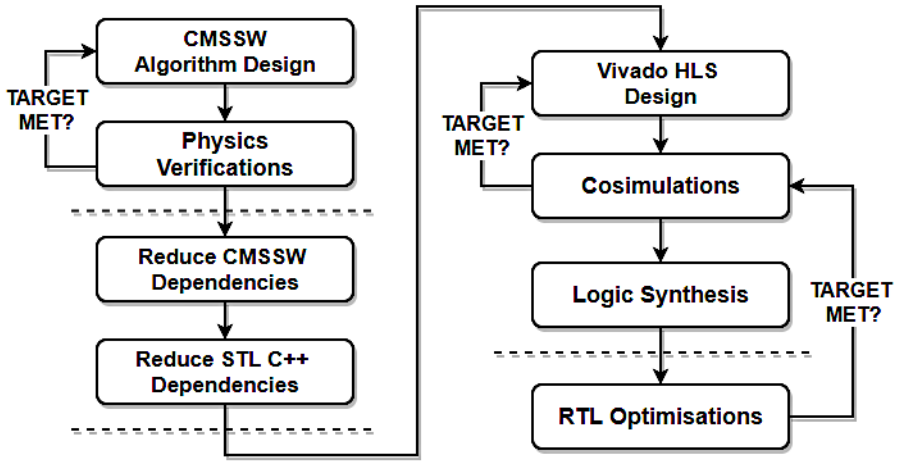

Figure 7.7: High-Level Synthesise design flow in CMSSW.

Code written in CMSSW is not compatible with HLS. Even if the *C++ STL* and ROOT [110] dependencies are removed, the resource utilisation may still surpass the logic resources available on an FPGA board for a simple design. In the case of the Linear Fitter, the initial synthesis results indicated that the logic utilisation is about 500% higher than all logic resources available in the Xilinx Virtex UltraScale board. Considering that sixteen copies of the Linear Fitter are required for just one nonant, the design was not feasible. The primary consideration had to be made in reducing the memory usage in a fully pipelined architecture. The conventional memory blocks (ROM, RAM, BRAM, ultraRAM) on an FPGA chip have one read or write command execution constraint per clock cycle in single-port memory and two read or write command execution in dual port memory per clock cycle. This limitation introduces a data dependency issue or bottleneck in pipelining stage where more memory access per clock cycle is required. The Xilinx Vitis-HLS synthesis tool [111] provides access to hardware reports and control logic to identify the problem area. The tool is particularly useful in identifying performance bottlenecks by looking at profiling reports. The synthesis tool also provides directives to customise the synthesis results across different solutions by highlighting the aspects of data access that limit the hardware performance. A few examples of applying these directives in

Linear the Fitter design architecture are discussed here. The HLS optimisation directives (pipeline, array-partitioning, loop unrolling, function inlining) have been applied to different data flow algorithm locations to achieve the required performance using minimum resources. In terms of managing input/output, a block-level interface protocol is added to the design to control the communications between modules. The stub stream data flow between individual modules at the top-level hierarchy is modelled using the *hls::stream* library and *ap-fifo* of infinite depth. The library creates data containers similar to *std::vector* in the *C++* Standard Template Library [106] for instances when the total number of stubs is unknown at runtime. The double-precision native variables in the initial CMSSW version of the Linear Fitter are converted to arbitrary precision fixed-point types using Vivado-HLS and the Vitis-HLS *ap-fixed* library to ensure the decimal point is correctly positioned and aligned in arithmetic operations. The library also provides quantisation models to represent variables in a finite word length.

7.3 Linear Fitter Firmware Configuration

The Linear Fitter hardware design has been developed in three variations based on the evolution of the Track-Trigger and Tracking system in Level-1 Demonstrator. In the early stages of the Time-Multiplexed Track-Trigger and Hybrid project, the data flow was specified as either sample-based [112] or frame-based [113] for hardware pipeline strategies. Although the movement of data in these two configurations is different, in performance, they are similar. The concept is to create a design with the highest possible performance, processing one sample of new input stubs every clock cycle, and addressing the optimizations before reducing latency and resources. The optimisation stage aims to reduce the initiation interval by allowing the concurrent execution of operations by classifying the available data flow. The primary characteristic of the sample-based data flow is how to read and write operations for a single stub. Often, the style uses static variables such as counters or accumulators to retain the values of the variables between transitions, otherwise, the values will be lost or set to zero. The system must process one input per clock cycle to achieve the required initiation interval. The design is then considered to be fully pipelined. In the frame-based

strategy, the primary characteristic is to process more than one sample in a transition. In this form, the data movement is often in memory blocks with pointer declarations to their first element. A transition is defined by complete read and write operations of all samples. This style produces maximum throughput in processing a stream of data, however, the pipelining to achieve the required initiation interval is not trivial and needs careful consideration to solve dependencies to eliminate bottlenecks. In a hardware implementation, there are strict rules in the declaration of memory size before runtime for compilers to allocate adequate storage. When compilers can not determine the exact size of a block memory, they throw exceptions that can lead to the termination of the processes. If there is no good strategy for dynamic memory allocation, pipelining the design is also impossible. In the following sections, three designs of the Linear Fitter using sample- and frame-based models are described and a hybrid of these is introduced.

7.3.1 Linear Fitter Single Iteration

The Linear Fitter single iteration architecture follows sample-based design development. The stubs are stored in shift registers or line buffers fed to the first Linear Fitter modules through the top-level design. Each Linear Fitter single iteration block receives and processes only one stub per clock cycle. In this configuration n number of blocks are integrated into the firmware chain with n number of bookkeeping blocks to process and update the variable counts. The bookkeeping blocks are also responsible for terminating the algorithm when appropriate. The number of iterations is in sync with the fixed number of samples. The latency is defined by the ratio of clock cycles and the number of blocks in the chain. The implementation emulates the dynamic configuration as the total number of stubs can be unknown at runtime. The disadvantage of the implementation is excessive resource utilisation in some cases, the number of stubs is less than the number of blocks. The inefficiency become more evident in the later development of Time-Multiplexed Track-Trigger when a reconstructed track candidate reduced the number of processing stubs. This resulted in several down-chain blocks staying idle throughout the track fitting process. The advantage of the single iteration model is its flexibility for integrating various tracking system architectures to estimate latency and resources accurately. A single iteration

block with an initiation interval of one is often sufficient for assessing and evaluating latency and logic resources. Figure 7.8 shows the single iteration block with the Linear Fitter and the bookkeeping modules.

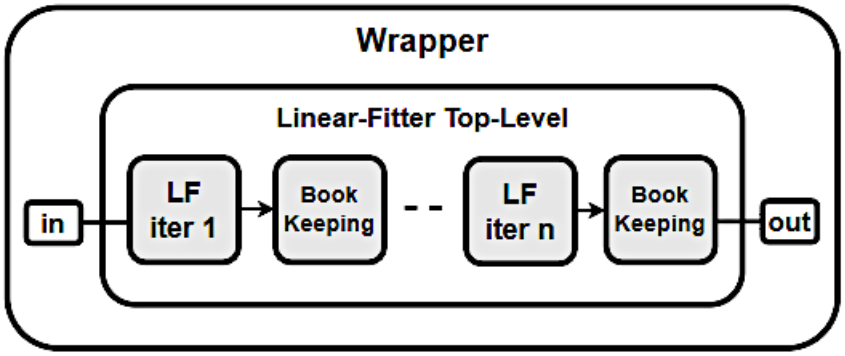

Figure 7.8: Linear Fitter Single Iteration model.

7.3.2 Linear Fitter Recursive Architecture

The Linear Fitter Recursive architecture follows the frame-based design development. The process aims to maximise throughput by increasing the number of stubs processed per clock cycle. In frame-based development, the Top-Level Linear Fitter module sees an array of stubs at its input and is capable of processing more stubs concurrently. The input is stored in a block RAM implemented as a dual-port interface supplying two samples per clock. Single-port or Dual-port memories can only supply a limited number of stubs per clock, resulting in bottlenecks and highly paralleled hardware design often unable to process all the data within the required initiation interval. In cases where higher concurrency and throughput are required, the RAM is partitioned, resulting in multiple small memories or multiple registers instead of one large memory, increasing the number of read and write ports significantly. In addition, the number of addresses in memory must be declared as a constant variable for compilers to allocate the necessary resources for RAM implementation. In the Linear Fitter, a block RAM to store 16 x 64-bit stub words[1] utilises an 18 kb memory slice and a block RAM to

[1]Maximum number of stubs in a stub stream.

store 16 x 96-bit track work utilises a 36 kb slice. In the recursive configuration shown in Figure 7.9, the Linear Fitter processes all sixteen stubs in a fixed-sized loop to efficiently assist compilers in applying pipelining optimisations. The advantage of the frame-based implementation is high throughput as more samples are processed per clock cycle. The disadvantage is the excessive memory usage and array optimisations for a fully pipelined design.

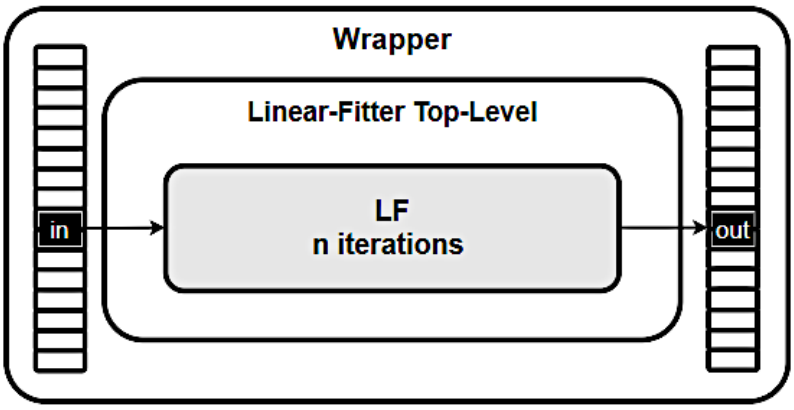

Figure 7.9: Linear Fitter Recursive model.

7.3.3 Linear Fitter Combined Model

In the Linear Fitter Combined configuration, the advantages and disadvantages of both single and recursive models are investigated to create a design compatible with different architectures of the Level-1 tracking systems for various real event samples. The model overcomes the variable-sized loop constraint by studying the average number of stubs using simulated $t\bar{t}$ + (0, 140, 200, 250) pileup events. A Linear Fitter recursive module is integrated into combined architecture which permits the hardware assembly for any number of stubs in a track candidate with the cost of adding or removing a single iteration block from a design. For example, in $t\bar{t}$+200PU events, the minimum number of stubs is known to be six and the maximum number is eleven. The recursive fixed-sized loop is set to six, and one (11 - 6 - 4 = 1) single iteration block is instantiated if four stubs are required in the reconstructed track. in the Top-Level architecture. The hardware model in

Figure 7.10, imitates the dynamic loop with the cost of adding a few logic gates for the bookkeeping process and FIFO blocks for storing incoming stubs. The combined design is used in Time-Multiplexed Track-Trigger and Hybrid architectures for its dynamic characteristic of handling the variable-sized stub streams.

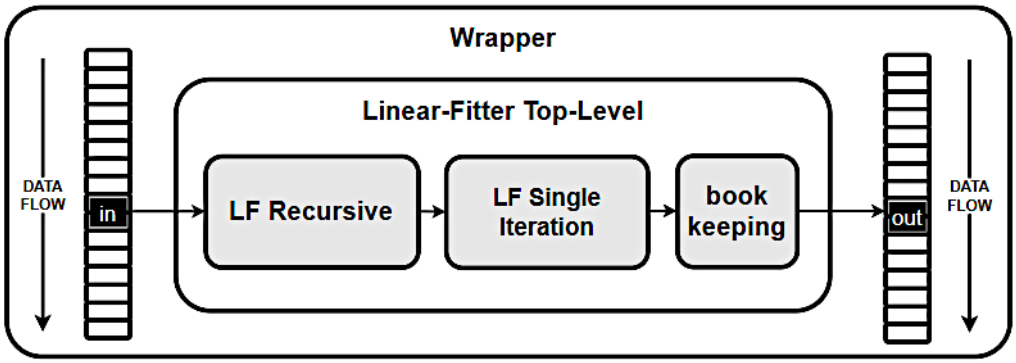

Figure 7.10: Linear Fitter Combined model.

7.4 Linear Fitter Performance

This section presents preliminary results of Linear Fitter against the applied trigger algorithm offline with standalone modules. As the Trigger system is not available yet, there is currently no real data from which to build samples. Therefore the required data is generated offline by particle physics event generators and Monte Carlo (MC) simulation toolkit. The data for simulations is usually produced by the CMS event generator group which is responsible for MC production and incorporating the required and expected parameters defined by the experts. The first stage of generating samples involves the MC event generator containing packages usually written by the theorists/phenomenologists. There are many packages, but basically, all work in the same way, using a theoretical model to simulate the produced particles in a specific type of collision. The theory is most commonly the Standard Model of Particle Physics or a new theory that the simulation author (programmer) likes to test. For top and anti-top quark pairs in proton-proton collisions at 7 GeV, the generator is instructed to produce

processes with these definitions in every event. If a variety of collisions is required, a random number generator will produce a sample of events, where each event contains a complete list of particles produced in that event with their physics properties. In generated events with top quark pair, the particle listing contains quark and gluon from the interaction of two protons producing t, $\bar{t}$ and the top pair itself. A comprehensive list of the behaviour of different fundamental particles can be found elsewhere [114].

For producing pileup events, the generator can be used to generate samples of typical pileup in proton-proton collisions. The samples are mixed with the top quark pair event produced in the first stage, overlaid. The particle listing for each event contains all the produced particles in both types of the pileup and $t\bar{t}$ events.

In the final stage, the generated events are fed into software packages that simulate the CMS detector. The detector simulation is based on the GEANT4 [115] software package, designed by CERN to specifically simulate the behaviour of the detector. The samples are fed to the detector as input to determine what the output looks like for each event. The ramification of new optimisations and the detector material are studied from these simulations. If required, the existing particle listings are modified to reproduce samples to study a particular behaviour in a sub-detector (e.g., Tracker). With the benefit of knowing how the collision that created the output looked, using the simulated samples rather than the raw data in the simulations is recommended during the analysis stage of the detector's development.

The MC truth particles are the particles from the event listing that are used for efficiency studies. Only particles from the event that could realistically have produced a track in the tracker will be included in the truth particle collection. The *track matching* process associates the truth particles to the particles from a reconstructed track. The event listing contains values η and p_T for all truth particles. For the new module's testing, these values are compared to the values from the reconstructed tracks for one or many events.

In Figure 7.11 and 7.12 the tracking efficiency of Linear Fitter as a function of η and p_T for *Single*, *Recursive* and *Combined* modules are shown.

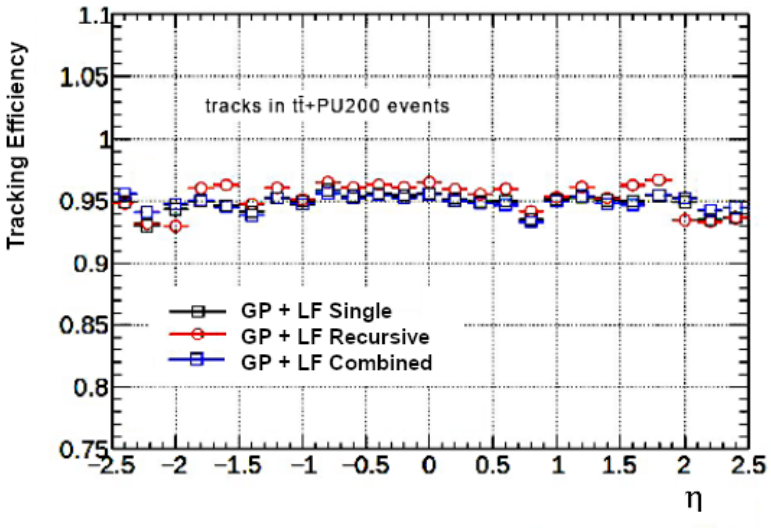

Figure 7.11: Tracking efficiency as a function of η.

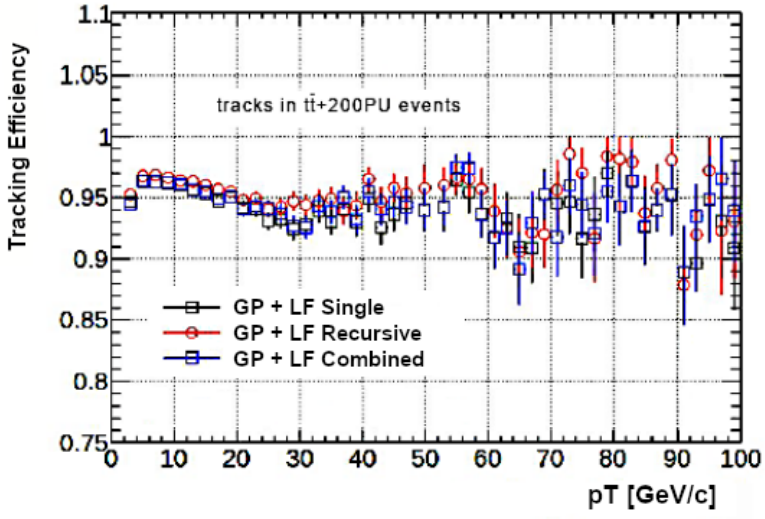

Figure 7.12: Tracking efficiency as a function of p_T.

The tracking efficiency (eff_{track}) is calculated by the number of matched tracks (k) over the simulated tracks (n) for 1000 $t\bar{t}$ pileup events given in the Equation 7.2 [27].

$$eff_{track} = \left(\frac{k}{n}\right)100 \tag{7.2}$$

Linear Fitter track reconstruction in standalone configuration for three Single, Recursive and Combined modules for the p_T ranges is shown in Table 7.5. Tracking performance is characterised separately in low-p_T ($8 > p_T \geq 2$), mid-p_T ($100 > p_T \geq 8$) and high-p_T ($p_T \geq 100$) ranges. The motivation for this is to observe the tracking efficiency under the challenges of calculating a particle trajectory in different stub occupancy regions. The low-p_T region corresponds to the highest number of stub occupancy making the tracking more challenging, and the high-p_T region corresponds to the lowest number of stub occupancy making the tracking less challenging for all Level-1 Track Finding modules.

p_T[GeV/c]	$8 > p_T \geq 2$	$100 > p_T \geq 8$	$p_T \geq 100$
Simulated	896	414	56
Single	849	397	54
Recursive	854	399	54
Combined	861	405	55

Table 7.5: Number of matched and simulated tracks for p_T ranges.

The tracking efficiency for reconstructed tracks is shown in the Table 7.6 in percentages. The Complexity represents the module characteristics concerning the overall algorithm and hardware implementation for producing the given efficiency and performance. The track-finding modules can be selected according to the different levels of required efficiency and available resources on an FPGA chip. The single iteration module has the lowest overall complexity and dependency on memory usage which is suitable for chips with fewer Block RAMs and FIFOs. The Recursive module uses memory for storing intermediate results between iterations which are implemented in Block RAMs or FIFOs. The number of Block RAM slices depends on the number of bits required for storing the stubs and track parameters. The Combined

module is a mixture of Single and Recursive modules specifically designed for future scalability or downsizing of the Level-1 Track-Trigger system which will affect the number of stubs in the stub streams. This value is currently set to sixteen, however, in future system developments, the value might be decreased or increased depending on the integrated luminosity upgrades.

p_T[GeV/c]	$8 > p_T \geq 2$	$100 > p_T \geq 8$	$p_T \geq 100$	Complexity
Single	94.7%	95.8%	96.4%	low
Recursive	95.3%	96.3%	96.4%	medium
Combined	96.1%	97.8%	98.2%	high

Table 7.6: The ratio of matched and simulated tracks for p_T ranges.

The Linear Fitter tracking efficiency increases as the p_T range increases. For particles with p_T > 200 GeV, almost all tracks are reconstructed. With 99.7% efficiency, only tracks that their trajectories are partially in the barrel and partially in the endcap suffer from one or two missed layers due to p_T modules alignment in the tracker. The layers encoding schemes can help the Linear Fitter algorithm to identify such tracks and reconstruct them differently by connecting two partially reconstructed tracks into one. This introduces additional costs in hardware design and changes with other tracker layouts.

7.5 Track Bend Analysis

Linear Fitter is designed based on p_T module development that selects particles with high transverse momentum $p_T \geq 2$ GeV/c. A correlation window of the strips in $r - \phi$ can discriminate against the particles whose trajectory bends significantly in the 4 T magnetic field. Correlations using stacked sensors can identify hits from straight tracks interacting within the coincidence window at high efficiency. This is because the ratio of a p_T module's radius installed close to the interaction point is considerably smaller than the radius of a high transverse momentum track. The Linear Fitter algorithm is particularly effective in reconstructing a track where its corresponding particle trajectory approaches a straight line in the $r - \phi$ plane. As the particle trajectories are bent by the magnetic field, there might

be some curve to data that a straight line provides a reasonable enough fit to make a prediction. However, for a precise fit, this is not enough. In contrast to linear relationships, curved relationships between variables are more difficult to fit and evaluate. In Linear Fitter design two scenarios have been considered for adding curve fitting feature for particle trajectories in the lower p_T range. The first scenario is the development of polynomial regression in the hardware implementation which resulted in excessive use of logic resources (up to 200%) in calculating the gradient, intercept and fitting process. The second scenario is the studying of the relationship between the transverse momentum and the track bend in an extended version of the Linear Fitter, which is discussed in more detail. One important metric for curved tracks is the stub reconstruction efficiency, and its dependence on angle α, which can be used to emulate the bending of a track in the CMS detector. The parameters involved in the calculation are shown in Figure 7.13.

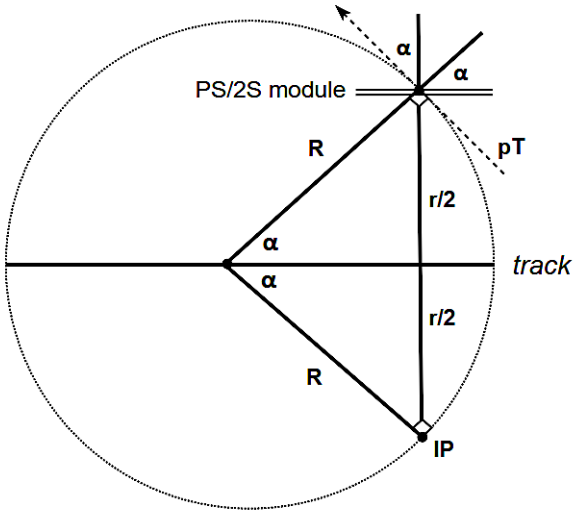

Figure 7.13: Relationship between angle α and p_T.

The radius of curvature R (in m) of a charged particle with transverse momentum p_T (in eV), and charge q (in electron units), bent in the transverse plane by a homogeneous magnetic field of strength B (in Tesla) is given by Equation 7.3.

$$R = \frac{p_T}{qBc} \tag{7.3}$$

With the information from Figure 7.13, the angle α can be denoted as:

$$sin(\alpha) = \frac{r}{2R} \tag{7.4}$$

Combining with the Equation 7.3, the relationship between the angle α, and the p_T of a traversing particle at radius r is derived within the CMS magnetic field by Equation 7.5 [96].

$$p_T[eV/c] = \left(\frac{qBc}{2}\right)\frac{r}{sin(\alpha)}, \quad p_T[GeV/c] = \frac{0.57r}{sin(\alpha)} \tag{7.5}$$

The radius of curvature of traversing particle from the interaction point (IP) for a module at a given radius r and the angle α concerning the perpendicular of a p_T, α decreases with an increased radius of curvature R which corresponds to an increase in p_T. Figure 7.14 shows the relationship between α, ϕ_0 and ϕ of a track in the transverse plane generating at the interaction point.

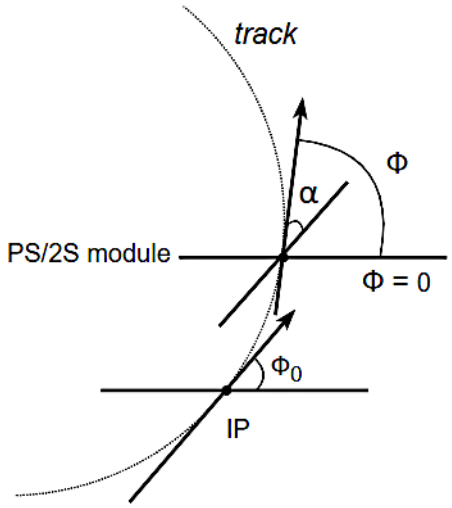

Figure 7.14: Relationship between angle α and ϕ_0 and ϕ.

Using the angle relationship between parallel lines in Figure 7.13, the Equation 7.6 can be written as;

$$\frac{r}{2R} = sin(\phi - \phi_0) \tag{7.6}$$

For tracks with p_T > 2 GeV, the radius of curvature is large (at least 1.75 m), so the small angle approximation can be used, which gives:

$$\frac{r}{2R} \approx (\phi - \phi_0) \tag{7.7}$$

Combining Equation 7.3 and 7.7, one obtains the relation between the (r, ϕ) coordinates of a single stub, and the corresponding straight line in the (q/p_T), ϕ_0 as:

$$\phi_0 = \phi - \left(\frac{Bcr}{2}\right)\frac{q}{p_T} \tag{7.8}$$

It is now evident how one can transform a stub with coordinates (r, ϕ) to a line with intercepting ϕ_0 and a gradient q/p_T proportional to r. The Equation 7.8 is used to convert Cartesian stubs coordinates (x, y) equivalent to (r, ϕ) into the line in (q/p_T, ϕ_0). The gradient of each line is proportional to the radius r of the stub, so is always positive. To make sure that a given track has stubs with an appropriate range of positive and negative gradients in (q/p_T, ϕ_0), it is preferable to measure the radius of the stub using the variable $r_T = r + T$, where T is a selected offset.

$$\phi_T = \phi - \left(\frac{Bcr_T}{2}\right)\frac{q}{p_T} \tag{7.9}$$

Where ϕ_T is the ϕ of the track at a chosen radius T. The technique is used to measure Linear Fitter sensitivity of the assumption that a high transverse momentum p_T > 2 GeV/c approaches a straight line in the tracker r-ϕ plane. As the result, an additional feature is added to the Linear Fitter as a plugin module to compare the tracking efficiency in an extended version. To implement the module in FPGA logic, it is necessary to create an array cell with a range of $|q/p_T| < q/p_T^{min}$ where p_T^{min} = 2 GeV/c and $\phi_T^{min} < \phi_T < \phi_T^{max}$. A track candidate is identified if its stubs from the tracker barrel layers or endcaps accumulate in one of these cells. The coordinates of the cell are used in the Linear Fitter as the extended parameters instead of retrieving the inputs directly from the Geometric Processor.

In Figure 7.15 the simulation output is compared to the original Linear Fitter results to observe the tracking efficiency in transverse plane $r-\phi$.

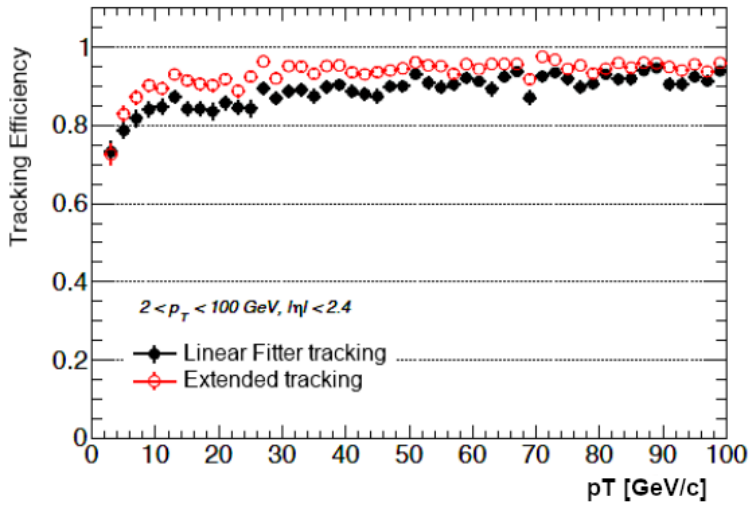

Figure 7.15: Tracking efficiency as a function of p_T.

Figure 7.16 shows the tracking efficiency of the Linear Fitter against the extended tracking in the $r-z$ plane. The tracking efficiency is marginally affected in η due to non-existing overlapping regions 6.4 for curved tracks.

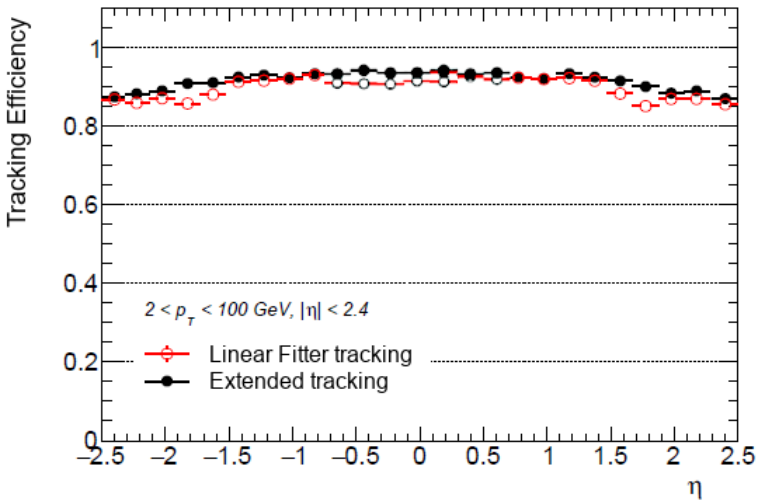

Figure 7.16: Tracking efficiency as a function of η.

According to the simulation, the tracking efficiency is improved by considering the track bend in the algorithm, however the associated hardware cost is greater for added capability. The overall tracking is generally more effective with finer array cell granularity. For a 32x32 cell layout, it is estimated that an additional cost of approximately 10% or the utilisation of 146 block RAMs is required. The cost will increase if finer granularity, such as 64x32 array cells, is desired. In addition, a cost for arithmetic logic will occur in the multiplication/division stage of the implementation which is discussed in the hardware development section in the original Linear Fitter Section 6.3.

7.6 Potential for Improvement

The Linear Fitter is linked to the DTC and GP modules in the tracking system. The geometric processor parameters have been primarily tuned for the Kalman filter algorithm, dividing the tracker into $2\pi/9$ nonants, with each segment further divided into 36 (18η, 2ϕ) stub sectors. The segmentation creates large regions that impact the efficiency of the Linear Fitter due to virtual stub creation. The algorithm for calculating a virtual stub per layer of the tracker uses the boundaries of the stub sector to generate one stub from many detected stubs by the mean algorithm. If two tracks are formed in a stub sector, they will be merged into one track. Despite Linear Fitter reconstructing a track in every clock cycle, the probability of filtering a genuine track is high. Therefore, an alternative geometry segmentation is under investigation that divides the nonants into 48 (24η, 2ϕ) stub sectors, reducing the area between boundaries even further in the process. The segmentation is implemented in the GP module by taking into account the stub distribution over all sectors and the consistency with tracker regions. Each sector must receive a similar number of stubs to reconstruct tracks. In addition, the feasibility of the segmentation must be studied in the hardware implementation to ensure the load balancing can be divided across the hardware links. As the stub sectors are processed in parallel, it is expected that the latency will not be affected, however, the number of added links can present challenges in hardware implementation due to the limited availability of peripherals on the FPGA devices.

This page was intentionally left blank.

Chapter 8

Track-Trigger Demonstrator

The demonstrator system encompasses a collection of software and firmware modules running in parallel to provide simulation results for testing and verifying the firmware modules. In the Level-1 upgrade, individual modules were initially created as standalone devices independent of other software and hardware modules. After successfully verifying their functionalities in standalone mode, the modules with both software and hardware counterparts are transferred to the demonstrator for further debugging. All modules in the demonstrator must comply with the integration rules and design optimisations under the Level-1 Tracking system specifications, which are described in detail in this chapter.

8.1 Demonstrator Overview

The Level-1 Track-Trigger demonstrator is a replica of the software and firmware necessary for prototyping the entire system. The prototypes and the demonstrator firmware slices are located at Tracker Integration Facility (TIF) at CERN and the Rutherford Appleton Laboratory in the UK. Progress is regularly reviewed by a CMS internal expert committee [116]. Figure 8.1 shows the μ-TCA demonstrator located at the TIF, which comprises several general-purpose FPGA Master Processors in a Carrier Hub (MCH). The modules in the Tracker system are assigned to MP7 blades in a chained configuration based on their functionality and location in the track finder and track fitter hardware chain. The modules are exported as IP cores, including unique, verified software and hardware identifiers with custom

configurations. The IP cores are programmed onto the FPGA chips and deployed in the demonstrator architecture. The input to the demonstrator is converted to pattern files to create test vectors for both software and hardware simultaneously. There are several modules in the demonstrator responsible for preparing data for testing and verification.

First, a producer module processes stubs and converts them into digital tracks, followed by a writer module that generates pattern files from tracks for hardware. After the data has been interfaced and processed by the Unit Under Test (UUT), a reader module captures the patterns and converts them back into digital tracks. Finally, a comparator module compares the patterns generated from both producer and hardware to create simulation files. The performance of the system in software and hardware is superimposed in a plot. If there is an inconsistency between tracks generated by the software and tracks generated by the hardware, it can be seen at this stage.

Figure 8.1: The μ-TCA hub at the Tracker Integration Facility [117].

The verification process relies on simulation tools and ROOT data analysis software to produce plots or histograms. The demonstrator architecture is complex and written in several programming languages, it therefore has a user-friendly interface to facilitate integration with all parts of the system. The processes have been automated using executable scripts and source files, so the system can be used by people with little or no knowledge of the demonstrator's components.

8.2 Demonstrator Scope

The CMS Level-1 upgrade [118] specifies a Track-Trigger system that produces efficient physics results under intensive conditions expected at the HL-LHC. This specification has been applied to the demonstrator system as it is intended to be a replica of the tracking system planned for the experiment. The timeline for the Level-1 Track-Trigger is laid out in Appendix A7. Successful module integration must comply with the required latency and resource allocations for each module.

The goal of the demonstrator is to confirm the creation of Level-1 Track-Trigger firmware that can operate within a 3 μs latency for events containing a large numbers of particles. The system must particularly evaluate the performance of fixed-point hardware development replacing the floating-point software design. The demonstrator must represent the minimum feasible segment of the detector to minimise the hardware necessary for effective demonstration. The software and hardware modules in the demonstrator must produce robust physics performance results within the specified hardware resource usage and latency. The architecture of the demonstrator must be scalable without requiring additional sub-modules and external memory.

The consistent use of a specified set of simulated data samples produced by CMS in both simulation and emulation analysis. For example, a sample of $t\bar{t}$+250PU events to examine the performance under extreme conditions to determine the impact of latency. The performance is quantified as the efficiency in p_T and η ranges to ensure robust tracking for overall matched tracks. The number of fitted tracks per bunch-crossing as a function of genuine tracks and duplicate tracks must be studied to estimate fake rates. When low efficiency is obtained, the simulation results must be reviewed to investigate the cause and to attempt to improve the performance.

8.3 Hardware Platforms

The high luminosity upgrade requires new custom-designed FPGA development boards with backward compatibility with the current CMS Trigger and DAQ system. The specifications for hardware-based platforms require the use of Xilinx technology with modular capability, support for Samtec FireFly optics up to 25 Gb/s and high versatility for the Level-1 hybrid tracking algorithm [119]. High-performance connectivity is required for stub and track transmission between modules (FPGA-to-FPGA) using high-speed multi-gigabit transceivers. Currently, three separate development strands are being pursued simultaneously to customise FPGA platforms for the Level-1 tracking system. A short description of each is given in the following sections as one of them will be used in the planned Track-Trigger system.

8.3.1 Master Processor

The use of all-FPGAs systems allows fast data processing at a substantial rate of 10 Tbps. The Imperial Master Processor (MP) board [120] was developed and upgraded for the Track-Trigger to meet the latency requirement and to handle complex algorithms with minimum resource usage. A combination of a Xilinx Virtex-7 (XC7VX690T) FPGA chip and optical interfaces on a single board has simplified the chaining process for different modules. The chip is mounted on an MP7 board for portability and future upgrades. Each board is customised to endure harsh heat environments with a mounted heat sink. The generated heat by many FPGA boards will be a concern in the all-FPGA system. The board has 72 peripherals that facilitate 72 output links to DTC modules with a 10 Gb/s data rate per link. The board is compact and smaller compared to other boards under development. The dimensions of the board allow installation in tight spaces in CMS experimental testing facility and tracker integration lab. The power consumption of each board is estimated from the total number of active pins and firmware configurations at around 30-40 W [121]. Power is a concern for usability in μ-TCA crates with many connections simultaneously. Therefore, other proposals are being considered for the cooling systems or alternative master processors. Despite this, the MP7 board is widely used for various CMS upgrade projects.

The MP7 board can be seen in Figure 8.2.

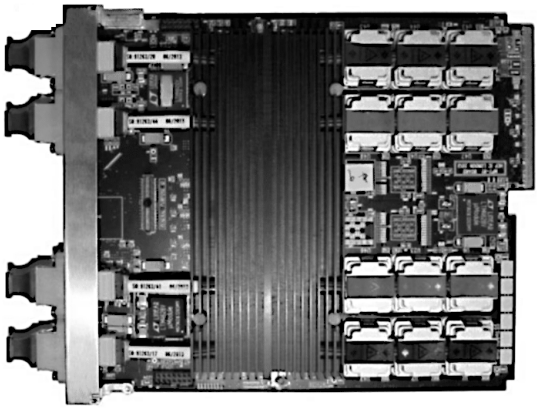

Figure 8.2: A Master Processor platform with a heat sink [122].

8.3.2 Apollo Platform

Apollo is a generic open-source development platform that uses ATCA blades designed for high throughput data processing applications [123]. Two programmable high-speed FPGA chips provide a transmission rate of up to 28 Gb/s with onboard memory and optical fibre interfaces operating jointly or independently. Apollo provides a service module for ATCA control for board-to-board communication and a command module for application-specific data processing algorithms. Various communication protocols have been considered for the Trigger readout electronics from the detector's front-end modules to FPGA pins through I2C, and AXI peripherals. The board uses Xilinx Kintex XCKU15P and Xilinx Virtex UltraScale+ XCVU7P chips to accommodate track finding modules [124] required in the Hybrid configuration. An entire nonant can be placed on one Apollo board for demonstration and verification with an efficient floorplanning [1] scheme. There is a large heat sink covering both FPGA chips connected to an extended frame to help heat dissipation across the hub. The board has more optical links peripherals to increase connectivity to sub-detector front- and back-end electronics and board-to-board interfacing.

[1] Choosing the best layout for placing synthesised modules on an FPGA chip.

The Apollo development board can be seen in Figure 8.3.

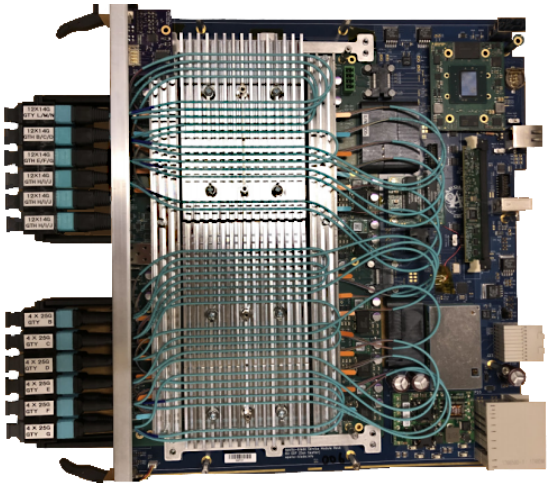

Figure 8.3: Apollo Platform with the heat sink [123].

8.3.3 Serenity platform

Serenity aims to simplify board-to-board communications by separating software from hardware using onboard Central Processing Units and standardised board-level control [125]. ATCA is used for intercommunication between FPGA chips and the operating system that controls the board with Daughter Cards for the data processing applications. The UK collaboration is currently designing Serenity boards for use in the various high luminosity upgrade and Trigger subsystems. The board accommodates two replaceable Xilinx Kintex UltraScale XCKU115 FPGA chips with 144 links at 25 Gb/s each. The Serenity board has been tested in the demonstrator at 320 MHz clock frequency. If general concerns about power management and noise reduction can be addressed, the board will be the preferred platform for the Track-Trigger system.

Figure 8.4 shows a prototype Serenity board including daughter cards.

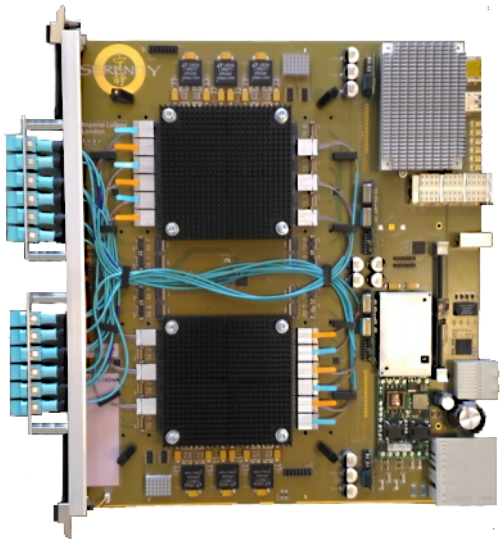

Figure 8.4: Serenity platform with the Daughter Cards [126].

8.4 EMP Infrastructural Firmware

The EMP framework is designed for trajectory reconstruction using the prototype Level-1 Track-Trigger FPGA boards. The reconstruction algorithms are verified and exported as self-contained IP cores which are chained in line with the EMP architecture for testing and monitoring. The interface to the EMP framework is through a user-defined data module known as the payload. Transmission between modules is managed through signalling protocols and on-chip IP Bus Builder (IPBB) [127] connections. IPBB supports both Vivado and Mentor Graphic emulation tools [128], providing a simple command-line interface for project creation, synthesis, implementation and bit file generation for programming FPGA chips. The input data is loaded into the channel buffers and injected into the payload. The processed data are then captured from the specified payload output channels and downloaded into output buffers to be read. The Hardware Access Library (uHAL) [129] provides complete access to memory blocks in the design to either write or read in RAM or FIFO configuration. The IPBB protocol supports User Datagram Protocol (UDP) and Peripheral Component Interconnect (PCI) for communication between all-FPGA IP cores. The top-level function of the EMP

framework provides access to and configuration for hardware components with user-defined parameters. In Figure 8.5, the architecture of the EMP framework is illustrated along with the Linear Fitter top-level module in the firmware chain. The EMP framework allows Tracking system modules to be chained together in sequential configuration.

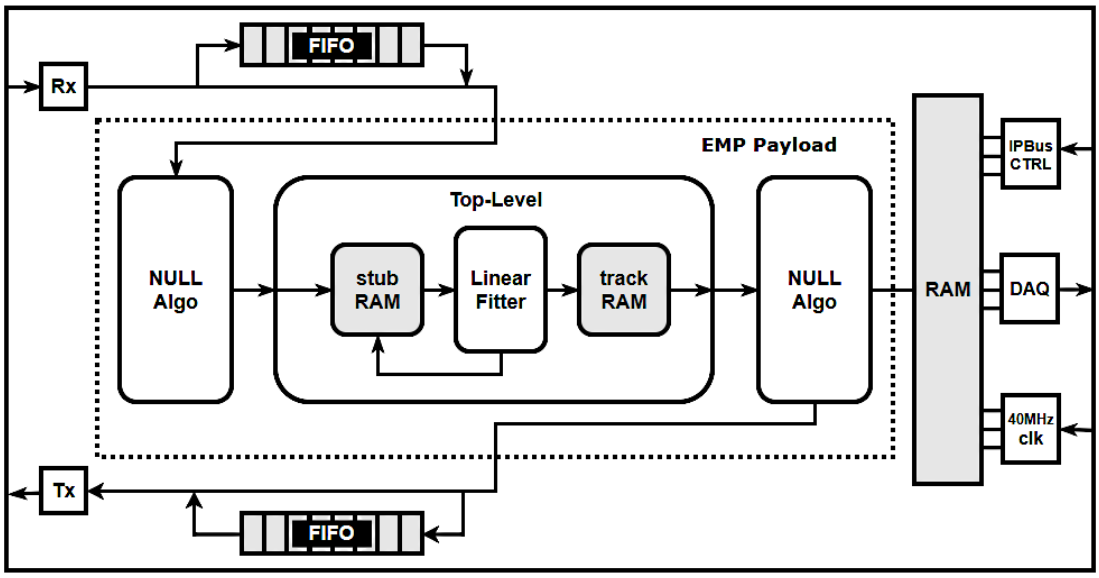

Figure 8.5: Linear Fitter integration in EMP framework.

The additional modules are labelled as null algorithm 8.9 interfacing before and after the Linear Fitter module. These locations are reserved for other modules depending on which version of the system is being used. The parameters in the payload are specific to the desired hardware behaviour, such as the number of bits per channel, LHC bunch-count, data-path regions, and clock ratio. The data is transmitted in frames specific to regional and geometric segmentation. One frame contains a sequence of 64-bit hexadecimal format words containing the stub information per clock per link per stub sector. The frames are used in prototype testing and verification by capturing the output data in text files or electronic waveforms.

8.5 Latency Requirements in Demonstrator

The trigger must reconstruct tracks with $p_T \geq 2$ GeV/c from stub streams during live data taken by the CMS detector. The stubs are produced at a 40 MHz clock frequency. The output rate of the Level-1 Trigger is fixed at a 750 kHz clock frequency. A reduction from 40 MHz to 750 kHz is achieved through the application of selection criteria and filtering. In terms of input/output, the difference between the time the first input is injected into the module and the time the module generates the first computed output defines the latency. The time allocation for individual modules is provisionally calculated by dividing the total latency by the number of modules in the system. This is only an estimate as some modules in the system require more time than others. More modules in the system mean shorter latency assignments for individual modules in the chain.

To create a system with good latency, parallel processing rather than the use of additional resources is inevitably the way forward. The modules in the Level-1 Tracking system aim to reduce the data rate according to a fixed latency specification by using n copies of the TFPs to cover the entire tracker. Every module is designed with a specific algorithm to be used at different stages and to do so with limited resource usage and within its latency allowance. Once the results are within acceptable limits, the modules are added to the demonstrator for further evaluation and validation.

8.6 Data Format in the Demonstrator

Stubs from 18 consecutive bunch-crossings are fed to the demonstrator in text file format using 72 channels [70]. A channel is an optical link to/from the tracking system. A text file is formatted into rows known as frames which contain packets from all channels for one clock cycle. A packet contains a 64-bit binary encoded with the hit coordinates, stub sector identifier and a valid signal. A valid stub generated from the first stub sector in the tracker transmitting from the first nonant on the first channel is represented as *Frame 0000: 1v00123456789abcef*. If empty frames are required between frames, they are added using empty packets *0v0000000000000000*.

Figure 8.6 shows a text file with eighteen frames for three channels.

Chan/Sector	0000/0000	0001/0000	0002/0000
Frame 000:	1v00123456789abcdf	0v0000000000000000	0v0000000000000000
Frame 001:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 002:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 003:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 004:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 005:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 006:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 007:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 008:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 009:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 010:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 011:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 012:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 013:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 014:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 015:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 016:	0v0000000000000000	0v0000000000000000	0v0000000000000000
Frame 017:	0v0000000000000000	0v0000000000000000	0v0000000000000000

Figure 8.6: Event data packet for injection into the EMP firmware [70].

8.7 Modules in the Demonstrator

A top-level design is created to wrap around the modules and grant access to their ports. During testing, a testbench is created to verify the correct functionality of the module through the use of test vectors and reporting utilities. Once validated, a fixed-point implementation of the module is created and utilisation reports are obtained. The design is optimised to produce the same outputs in software and hardware, correcting the loss of precision and overflow. Once the design has been optimised, an FPGA synthesis is created in HDL or as an IP core. The result is added to the demonstrator as a UUT to ensure it meets the Level-1 design specifications.

8.8 Efficiency and Performance Metrics

Dedicated upgrade documentation [101] outlines a series of baseline metrics using analysis tools and techniques developed by CMS. The metrics include tracking performance, robustness, and efficiency for the Multivariate Linear Fitter, Time-Multiplexed Track-Trigger and Hybrid systems. This approach

helps developers to evaluate new modules for the upgrade using standardised performance plots and efficiency measurements. The collaboration also specifies standard test conditions and data samples to facilitate performance comparisons between candidate systems.

8.9 Null Interface in the Demonstrator

The null interface creates a templated blueprint with plugin capability in the EMP architecture. For the hardware prototype, the aim is to configure the demonstrator to generate data flow similar to the FIFO back-end CIC electronics. As shown in Figure 8.5, the interconnections between the Linear Fitter module and EMP structure are through two memory blocks, stub RAM at the input and track RAM at the output. In Figure 8.9, the Top-Level Linear Fitter architecture in the EMP structure with the null interface and memory module is shown.

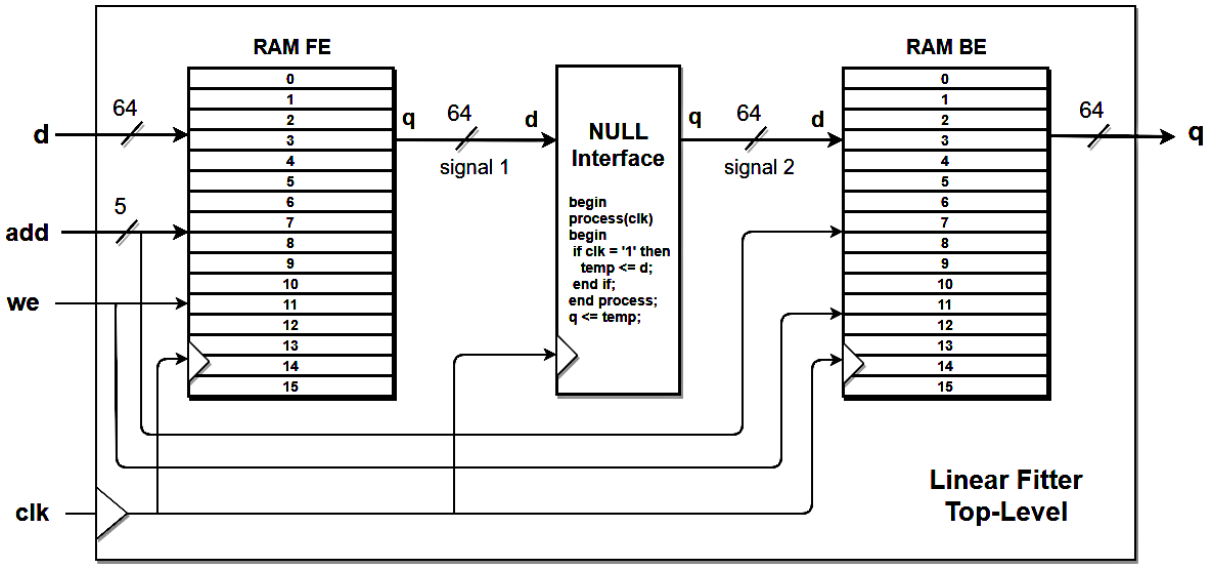

Figure 8.7: Null interface development for EMP framework.

The memories are only required in the EMP firmware demonstrator for storing inputs/outputs at the rate of the LHC system clock. Synchronising the data is particularly significant when comparing different approaches. The

stubs entering the EMP firmware are buffered at both ends to create the required initiation interval in a pipelined structure. The null interface is used to accurately calculate the latency in the demonstrator through time-zero marking. The timing coincides with draining the FIFOs before initiating a new process or clearing the block RAM counters before every run. In practice, If pre-processing of stubs is not required, time-zero corresponds to the transmission of the first stub for a given packet from the DTC-equivalent output. The null interface is particularly helpful in measuring the latency of the top-level subsystems before adding new modules to the demonstrator. The input and output storage are block RAMs programmed to function as FIFOs by shifting the stubs in and out of the null interface block on every rising edge of the clock. The implementation is also used to ensure the memory read/write instructions are efficiently executed. After verification, the Linear Fitter replaces the null interface module. An implementation of the null interface module in the Vivado Design Suite with the stub RAM as RAM_{FE} and the track RAM as RAM_{BE} on each side as shown in Figure 8.8.

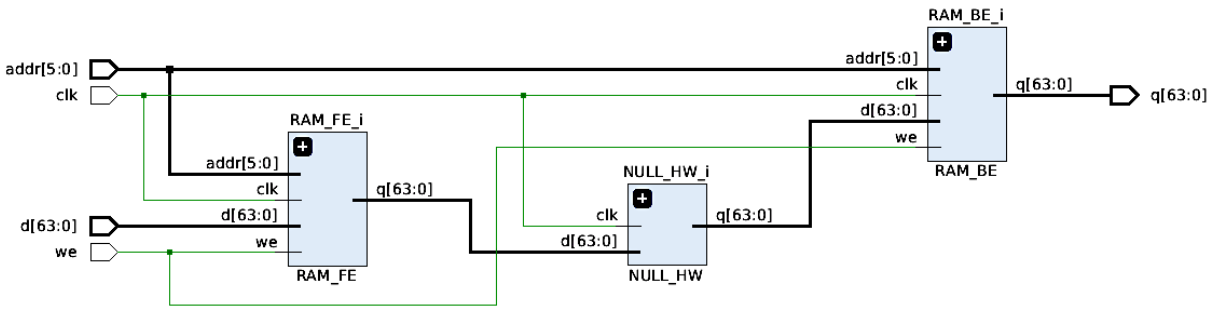

Figure 8.8: Null interface in Vivado Design Suite.

The input to the module is passed to the output on every clock transition. The Vivado implementation is based on the Virtex 7 Master Processor FPGA chip (XC7VX690T) at 40 MHz clock frequency. The RAM blocks are synthesised to Distributed RAM (LUTRAM), LUT for I/O interfaces, and FIFO for buffering and transmitting stubs per clock cycle. In addition, a global buffer (BUFG) [49] to distribute high fanout signals and the latency for the longest critical path is shown in the Table 8.1. The null interface effectively estimates the Linear Fitter resources and latency without considering the cost of utilising memory and interfaces exclusively. In the Linear Fitter

demonstrator, the depth of the block RAMs is equal to the number of stubs in the stream. The number varies in the different tracking systems approaches, leading to different latency and logic resource usage.

resources	utilisation	available	utilisation (%)	time (ns)
LUT	64	433,200	0.01	-
FF	4	866,400	0.03	-
BRAM	2	1,470	0.01	-
IO	2	600	5.33	-
DSP	0	3,600	0	-
BUFG	2	32	6.25	-
LATENCY	-	-	-	400

Table 8.1: Logic resource usage estimates in the null interface.

8.10 Modules in the Linear Fitter Demonstrator

The Linear Fitter is instantiated in the TFP hardware structure following regional segmentation ($2\pi/9$, $\pm z$) and geometric (24η, 2ϕ) segmentation of the tracker with a time-multiplexing factor of sixteen. The TFP receives a stream of stubs from two nonants with a maximum depth of sixteen. The firmware configuration is the simplest tracking system, placing only two modules (GP + LF) in the firmware chain. The DTC receives the stream of stubs from the front-end tracker electronics generated by the p_T modules and distributes them to the FPGA boards based on the regional segmentation scheme. The DTC first merges all stubs from all modules to one stream using routing blocks. Then, it organises the stubs according to stub streams per region. Next, the conversion from DTC stubs to stub streams is implemented for distribution to GP modules. The GP modules receive the stub streams and extract the η and ϕ components to produce stub sectors. If the segmentation follows 48 (24η, 2ϕ) segmentation per $2\pi/9$ it will produce 432 (η, ϕ) sectors that covers all the tracker regions. The number of TFP depends on the processing time of the Linear Fitter in the system. The Linear Fitter *Combined* module processes stub at 400 MHz. The algorithm in the Linear Fitter recognises the stub sectors boundaries and begins the extraction of stub

components r, ϕ, z, PS, 2S, layer$_{ID}$ and a valid bit from the stub stream for processing. The result would be one stub per layer if any stub occurrences were detected on the processed layer. The fit is calculated in the r-ϕ and r-z planes. The distance from the fit to the stub positions is calculated and the stub with the largest residual from the fit is removed from the stub stream. In the following iterations, more fits are calculated with fewer stubs until the minimum number of stubs that qualify for a valid track remains. There is no need for duplicate removal in this approach as the stubs with $p_T \geq 2$ GeV/c near the stub-sector boundaries have the most significant residuals and will be eliminated. The Linear Fitter is synthesised for deployment in the Master Processor Xilinx Virtex-7 (690T). An illustration of Time-Multiplexed Linear Fitter architecture in the demonstrator is shown in Figure 8.9 with Source and Sink modules to store stubs before and after the processing.

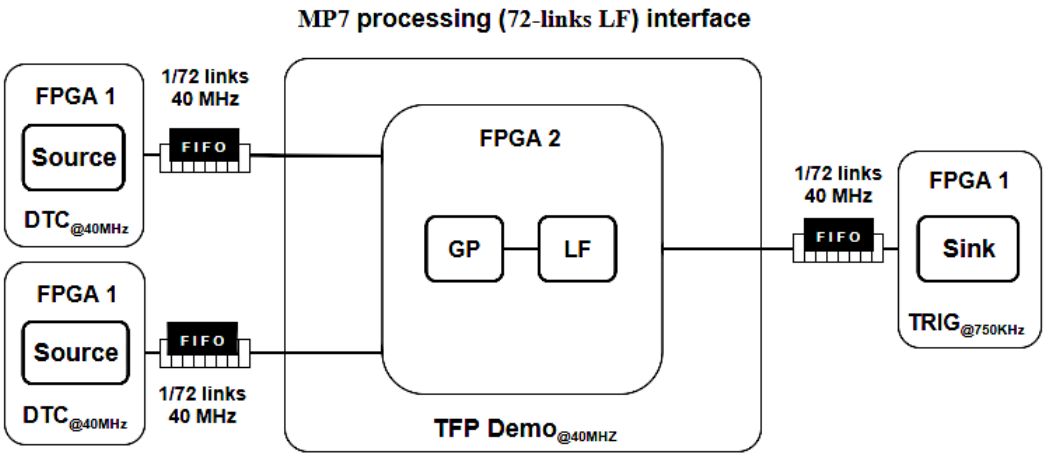

Figure 8.9: MVLF configuration of the TFP demonstrator.

8.10.1 Latency and Resource usage

This section presents the latency and resources estimates for the MP7 development board using 48 links over 72 available links and a mounted FPGA chip (XC7VX690T). The power estimation is discussed in Chapter 9.

The latency and resources for a TFP correspond to the sum of all latency and resources required for individual modules presented in Table 8.2.

board	resources	utilisation	available	utilisation (%)	time (ns)
FPGA 2 8.9	LUT	343,528	433,200	79.3	-
	FF	752,901	866,400	86.9	-
	BRAM	693.8	1,470	48.3	-
	IO	112	600	18.8	-
	DSP	288	3,600	8	-
	BUFG	5	32	15.6	-
	LATENCY	-	-	-	1,033.33

Table 8.2: Latency and resource in MVLF Demonstrator.

The utilisation is calculated as a percentage of the total available resources on the FPGA chip (690T) used in the demonstrator. The values for the source and sink modules are not presented as these modules are redundant in the complete system assembly. In instances where the estimates are capped to 100%, the suitability of the design is rejected, considering the null interface requires at least 2% logic availability for buffering and transmitting stubs between modules in the system. The input stubs are transmitted upstream over 48 x 25-Gb/s serialiser/deserialiser links out of the available 72 links to provide nearest-neighbour communications. The output stubs (reconstructed tracks) are transmitted downstream over a 25 Gb/s link to the correlator system. The information from the tracking detector is combined with information from other sub-detectors to identify physics quantities.

8.10.2 Physics Performance

The track reconstruction efficiency is measured relative to all generated charged particles from the primary interaction producing at least four hits on four layers of the tracker. A trajectory is considered to be successfully reconstructed if it is fitted from a track candidate consisting of stubs produced by a single charged particle with the $p_T \geq 2$ GeV/c and the $|\eta| < 2.4$.

The baseline performance metrics are shown in Figures 8.10 to 8.13.

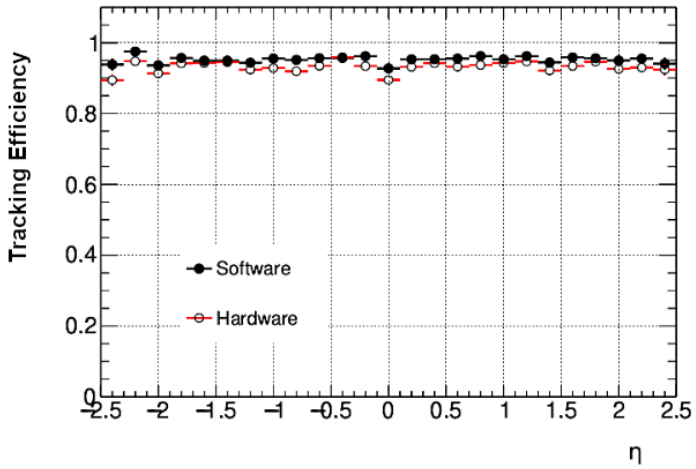

Figure 8.10: MVLF tracking efficiency as a function of η.

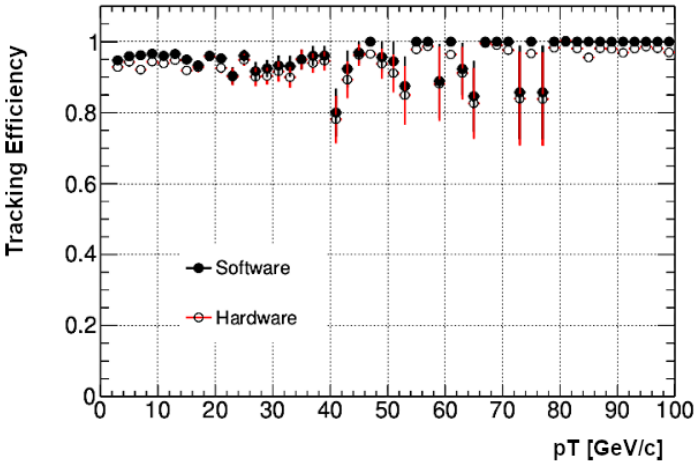

Figure 8.11: MVLF tracking efficiency as a function of p_T.

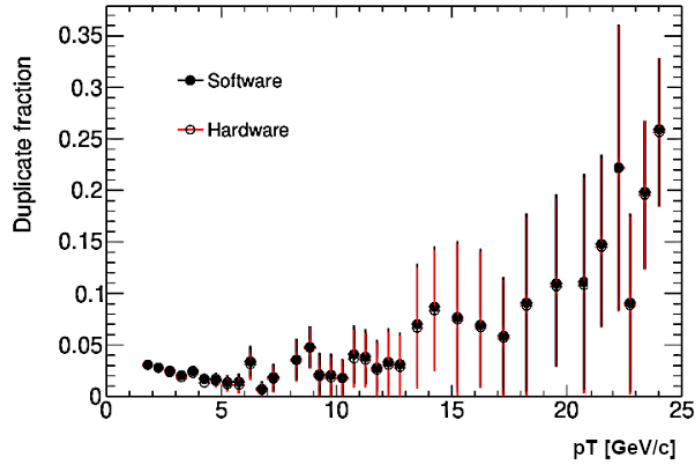

Figure 8.12: MVLF duplicate fractions for $0 < p_T < 25$ GeV/c.

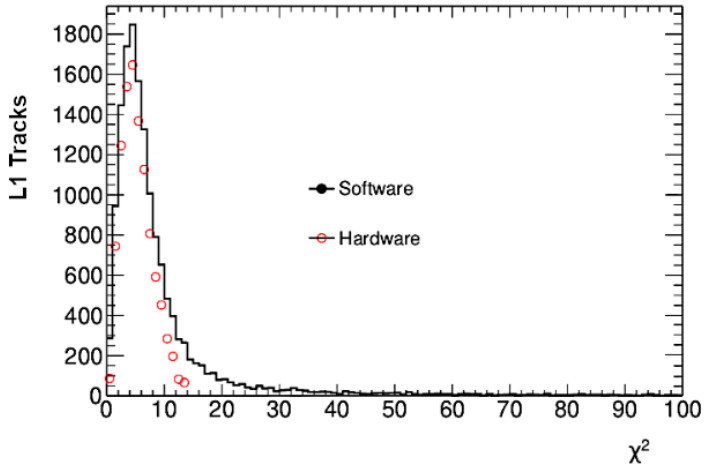

Figure 8.13: MVLF goodness of the fit χ^2 for matched tracks.

8.11 Modules in TMTT Demonstrator

Following the Linear Fitter integration in the demonstrator, the module is instantiated in the TMTT, replacing the Kalman system as the tracking algorithm. The TMTT architecture is more complex than the Linear Fitter design as it uses more modules in the firmware chain. The concern in TMTT development is achieving the required latency. The Linear Fitter aims to reduce the latency of the TMTT through a simplified algorithm and increased throughput. In addition, the Linear Fitter aims to achieve similar or better

efficiency through a precision fit algorithm in an integer-based design process. The TMTT system contains the DTC, GP, HT, mini HT, KF and DR modules. The additional modules are intended to increase efficiency and remove fake tracks. The functionality of DTC and GP modules has already been described. The HT module detects tracks in the r-ϕ plane and the value for z is determined from calculated values. Therefore, in Linear Fitter the value for z is obtained from the input of the HT instead of the output. This has eliminated the use of mini HT from the configuration. As a trajectory is reconstructed in every clock cycle, there will be sixteen tracks, however, the final track with the minimum number of stubs is selected. In cases, duplicates are present due to regional overlapping where one track crosses one region to the neighbouring region, they are removed by the χ^2 algorithm. In the demonstrator, the modules are chained using IP Bus Builder in the EMP framework at the system clock frequency of 240 MHz. The implementation is optimised for MP7 for the $t\bar{t}$+200PU events for consistency. The figreffig:fig814 shows an assembly of the TMTT. The goal of the demonstration is to use a minimum number of FPGA boards for practicality and cost reasons.

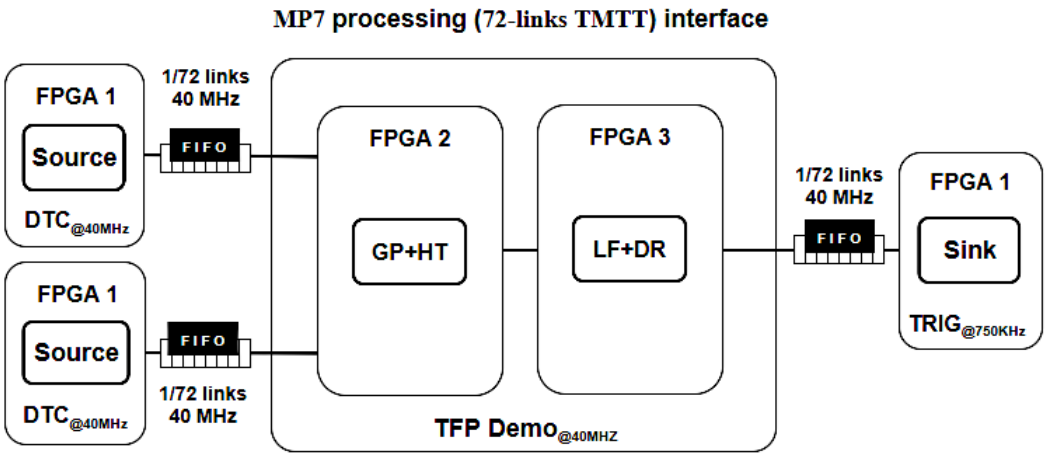

Figure 8.14: TFP demonstrator in the TMTT firmware chain.

The DTC in the demonstrator processes a packet of 72 events through 25 Gb/s μ-TCA blades. For emulation of two adjacent nonants, one FPGA board is used for both sourcing and sinking stubs to/from the TFP. The source in the FPGA board is responsible for storing two incoming text vectors and transmitting them to the TFP board. The sink FPGA board is responsible for receiving stubs from TFP and storing them in onboard block RAMs. The stubs are buffered in FIFOs before and after the TFP with the transmission interval of one stub per clock cycle. With efficient floorplanning, the GP and HT are programmed onto one FPGA board, and Linear Fitter and DR are programmed onto one FPGA board. In total, three boards are used in the emulator to verify the feasibility of the TMTT in the demonstrator. In the following section, the latency and resource usage of the TMTT implementation are examined.

8.11.1 Latency and Resources

The latency and resource usage in the TMTT configuration are shown in Table 8.3 for the FPGA boards marked *2* and *3*. For resource usage estimates for the entire system, the logic utilisation is multiplied by the total number of TFP blocks.

board	resources	utilisation	available	utilisation (%)	latency (ns)
FPGA 2 8.14	LUT	299,774.4	433,200	69.2	-
	FF	750,302.4	866,400	86.6	-
	BRAM	2,456.4	1,470	167.1	-
	IO	134	600	22.4	-
	DSP	1,440	3,600	40	-
	BUFG	3	32	3.13	-
	LATENCY	-	-	-	2,479
FPGA 3 8.14	LUT	389,446.8	433,200	89.9	-
	FF	855,136.8	866,400	98.7	-
	BRAM	1,026.06	1,470	69.8	-
	IO	14	600	19.3	-
	DSP	288	3,600	8	-
	BUFG	4.8	32	15.3	-
	LATENCY	-	-	-	1,641

Table 8.3: Latency and resource usage in the TMTT demonstrator.

8.11.2 Physics Performance

The TMTT physics performance in the demonstrator is evaluated against the metrics used in the Linear Fitter emulation for consistency. The aim of obtaining efficiency results in the TMTT is to compare them to other systems under development. Results are shown in the 8.15 to 8.18.

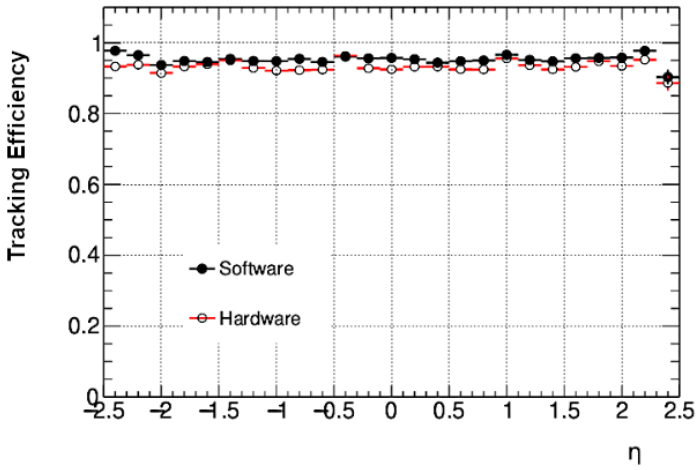

Figure 8.15: TMTT tracking efficiency as a function of η.

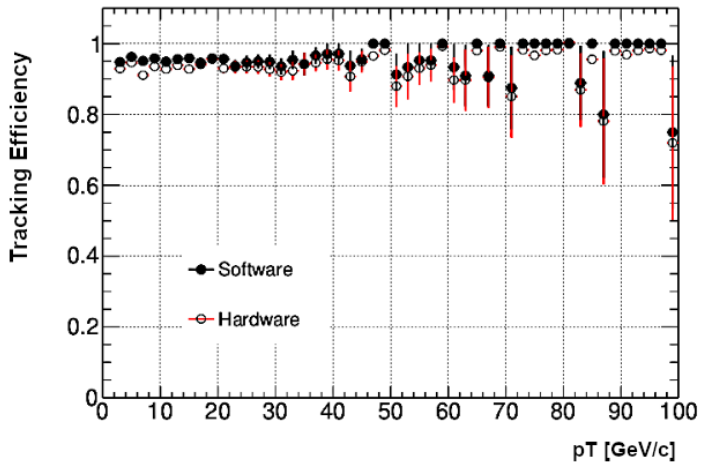

Figure 8.16: TMTT tracking efficiency as a function of p_T.

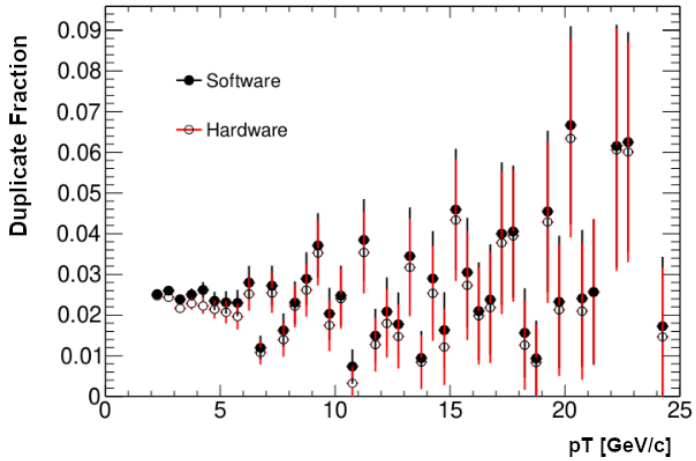

Figure 8.17: TMTT duplicate fractions for $0 < p_T < 25$ GeV/c.

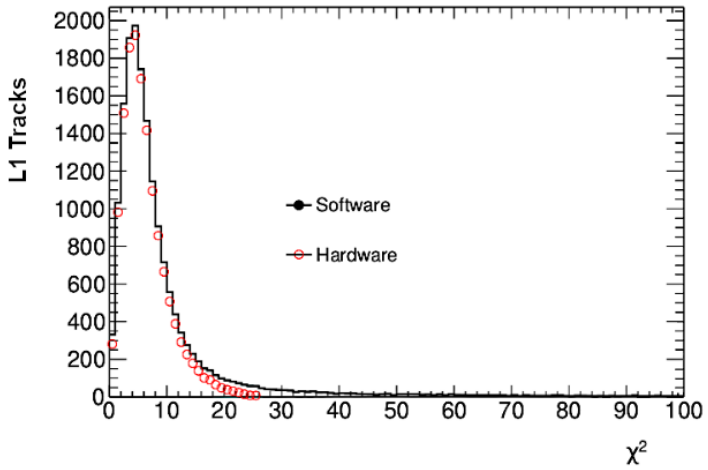

Figure 8.18: TMTT goodness of the fit χ^2 for matched tracks.

Comparing the plots generated in the TMTT to the Linear Fitter reveals that the efficiency in p_T and η are similar, however, in the goodness of fit plot, the hardware stops processing the stubs that produce χ^2 with a value under the precision range of its corresponding fixed-point representation. This would not present problems in determining the genuine tracks as the critical value for the χ^2 for a high degree of freedom is much larger than the plotted values.

8.12 Linear Fitter in the Hybrid Demonstrator

Following the integration of the Linear Fitter in the TMTT demonstrator, the module is instantiated in the Hybrid system, replacing the Kalman filter. The Hybrid architecture is more complex than the Linear Fitter and the TMTT as it uses more modules in the firmware chain. These modules have been required to provide a minimum number of candidate tracks to the KF module to reduce the fake rate. The concern in the Hybrid development is achieving the required latency. The Linear Fitter aims to reduce latency in Hybrid architecture by replacing the Kalman filter with Linear Fitter and removing the DR module. In addition, the Linear Fitter aims to achieve similar or better efficiency through the precision fit algorithm through custom-designed arithmetic slices to minimise DSP block usage. In the following section, the modules in the Hybrid system and the hardware performance in the demonstrator are discussed.

8.12.1 Modules in Hybrid Demonstrator

The Hybrid system is divided into Tracklet, Purge Duplicates (DP) and the Linear Fitter design. In the Tracklet design [67], several modules are chained in sequence to find track seeds from pairs of stubs in adjacent layers. Once the pairs or tracklets are identified, a road search algorithm defines their relationship to produce loose tracks. The tracks are interfaced to the DP module to identify reoccurring tracks in overlapping regions and remove them. The output from DP is interfaced with the Linear Fitter module to eliminate invalid tracks. A novel Track Quality algorithm replaces the χ^2 algorithm in the tracking efficiency and goodness of the fit stage. The use of the Track Quality algorithm as the final module in the Hybrid architecture is currently being studied. The geometry segmentation in Hybrid development is different from the Linear Fitter and the TMTT, as the tracker is divided into 28 x ϕ sectors with a time-multiplexing factor of six. The first module in the tracklet system [29] is the input router which maps and distributes the stubs to BRAMs according to the layers in which the stub vertices are initially detected. The virtual router divides the stubs into ϕ and z units known as virtual modules. The tracklet engine creates pairs from particle hit positions, and their parameters are calculated in the tracklet calculator module. The

projection transceiver and the projection router modules find the association between tracklets in adjacent layers and store them according to their corresponding modules. The match engine and the match calculator modules compare stub positions and transmit matches to their relative sectors. At this point, candidate tracks are formed with a high data rate and are accumulated in the full match module. The purge duplicate module is necessary to reduce the data rate for transmission to the next module. In the Track Fitter section, the Linear Fitter is used as a track fitter and added to the Hybrid firmware chain. In Figure 8.19, an overview of the Hybrid hardware architecture is shown, divided into four processing steps.

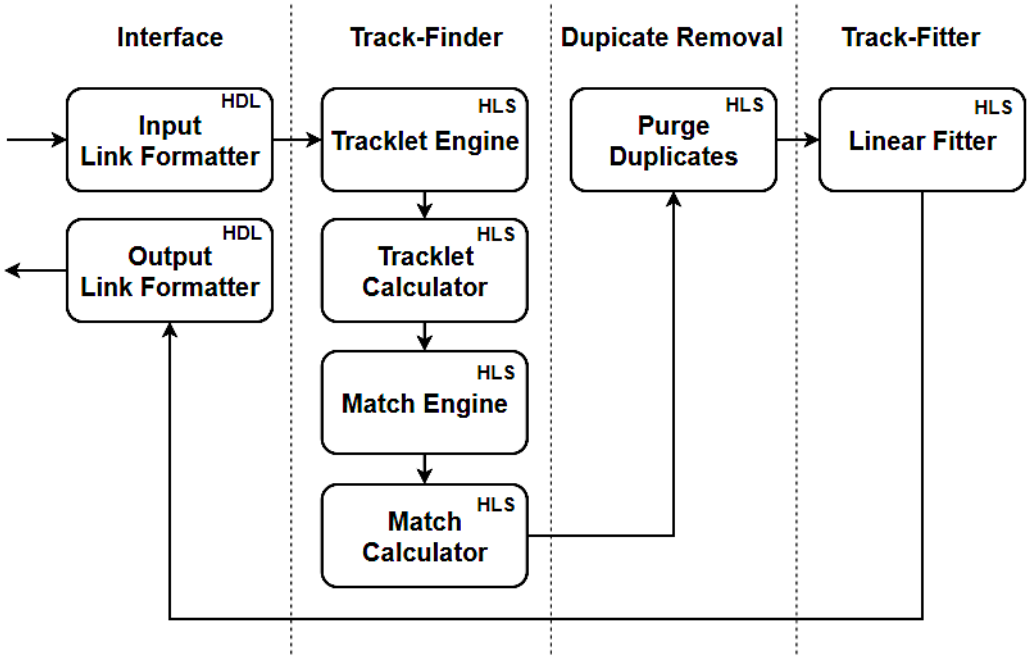

Figure 8.19: Modules in the Hybrid architecture.

Currently, the HLS modules are instantiated in the firmware chain with one wrapper per module. The use of one wrapper for all modules can reduce the algorithm complexity by removing redundant memory blocks between modules, providing the number of inputs/outputs are less than the number of available I/O pins in the MP7. The processing sectors receive a new stream of stubs at 150 ns intervals instead of 450 ns in the TMTT.

Additional parallelisation is achieved in both Track-Finder and Track-Fitter algorithms with the help of the HLS pipeline and loop unrolling directives. Figure 8.20 shows the Linear Fitter in the Hybrid configuration.

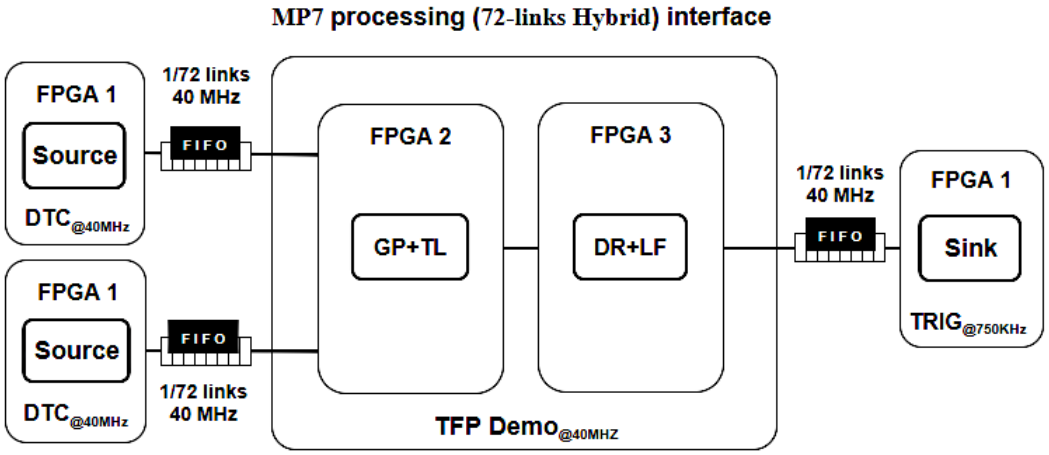

Figure 8.20: TFP demonstrator in the Hybrid configuration.

The Hybrid system has a different data format compared to the Linear Fitter and TMTT systems. More parameters such as χ^2_{rphi}, χ^2_{rz} are concatenated in a 96-bit binary track word for trajectory estimations. The coordinate parametrisation follows the trajectory of charged particles in a right-hand Cartesian coordinate system, creating a helical path rotating in both negative and positive directions. However, for high p_T particles, the trajectory resembles a straight line. All parameters in the track word with their bit assignments and ranges have been assigned. A valid bit confirms the validity of the track in the LSB location. The r coordinate is replaced with the signed inverse radius 1/R, the tanλ represents the relation between dip angle θ and the hit latitude λ. The azimuthal angle ϕ_0 and the longitude z_0 are calculated from the centre of the stub sectors. The d_0 is the distance of the closest approach of the track to the centre in the r-ϕ plane. The number of bits for each parameter considers the minimum and maximum variables in conversion from floating-point to fixed-point implementation. Stub transmission in the Hybrid architecture is similar to Linear Fitter and TMTT through 72 links with 25 Gb/s μ-TCA compatibility. The stubs are buffered in FIFOs before and after the TFP with the transmission interval of one stub per clock cycle.

With efficient floorplanning, the GP and Tracklet are synthesised on one FPGA board and the DP and the Linear Fitter on another FPGA board. Table 8.4 shows the bit assignments to the parameters.

parameter	unit	resolution	range min	range max	bits	signed	bit range
valid	-	-	0	1	1	no	[0:0]
1/R	1/cm	0.00000052	-0.00852662681	$+0.00852662681$	15	yes	[1:15]
ϕ_0	rad	0.0003	-0.69823248	$+0.69813248$	12	yes	[16:27]
tanλ	rad	0.0002	-8	$+8$	16	yes	[28:43]
z_0	cm	0.0099	-20.46922512	-20.46912512	12	yes	[44:57]
reserved	-	-	-	-	-	-	[58:70]
$\chi^2_{r\phi}$	-	-	0	∞	4	-	[71:74]
χ^2_{rz}	-	-	0	∞	4	-	[75:78]
reserved	-	-	0	∞	3	-	[79:81]
reserved	-	-	-	-	13	-	[82:94]
total	-	-	-	-	96	-	[0:95]

Table 8.4: Data format for the Hybrid demonstrator.

With 150 ns for six events and a clock speed of 240 MHz, a maximum of 36 stub pairs is transmitted from DTC-equivalent to the TFP through 28 links instead of 36 links. The stubs are processed and transmitted on 28 links to the sink module. If the number of stubs in-stream exceeds 36, the stream is truncated, resulting in a slight degradation of the efficiency if the effects are not significant.

8.12.2 Latency and Resource usage

In this section, the latency and resource usage of the Hybrid demonstrator is presented. A firmware implementation of the Hybrid demonstrator is assembled to emulate $t\bar{t}$+200PU events. The demonstrator system is optimised for data transmission and cross-module communications using the MP7 board with a mounted Xilinx Virtex-7 (690T) FPGA chip. The Hybrid demonstrator is also tested with the $t\bar{t}$+250PU. The total latency of the Tracklet with Linear Fitter module is found to be 3.3 μs, which can be optimised to meet the required latency of 3 μs. Hardware acceleration tools and HLS have been beneficial in achieving shorter latency through pipelining directives and array partitioning in a Hybrid system. The Linear Fitter is implemented in HLS for compatibility with the Tracklet firmware as part of an

additional module for the Hybrid system. The Latency and resource usage is shown in Table 8.5 are estimated by instantiating the Linear Fitter IP core in the Xilinx Vivado platform and the Virtex-7 690T FPGA chip through Vivado Quality By Design model [130] to ensure the design operates reliably under the Level-1 Tracking requirements. The reports from the source and the sink modules are not included as these modules are redundant in the complete system. It is estimated that the entire system requires 182 MP7 μ-TCA blades to achieve latency below 3 μs. The validation for the total number of FPGA boards is obtained in Chapter 9.

board	resources	utilisation	available	utilisation(%)	time (ns)
FPGA 2 8.20	LUT	387,714	433,200	89.5	-
	FF	998,959.2	866,400	115.3	-
	BRAM	1,946.2	1,470	132.4	-
	IO	459	600	76.5	-
	DSP	576	3,600	16	-
	BUFG	4	32	14	-
	LATENCY	-	-	-	3,133.3
FPGA 3 8.20	LUT	296,742	433,200	68.5	-
	FF	756,370.1	866,400	87.3	-
	BRAM	1,005.4	1,470	68.4	-
	IO	137.4	600	22.9	-
	DSP	1,476	3,600	41	-
	BUFG	6	32	18.8	-
	LATENCY	-	-	-	1,136.5

Table 8.5: Resource and latency estimates for the Hybrid system.

The total latency exceeds the timing requirements. Another area of concern in the firmware configuration is the number of FIFO and Block RAM slices in the FPGA-2 board that are greater than the number of available resources in the MP7 chip. It is possible to use LUT slices to balance the load and divert the memory requirements to other available resources on the chip. For example, the layer information in the Tracklet can be stored in lookup tables and accessed when required. The modifications can be made trivially made in the HLS implementation through directives to optimise the design and reduce storage dependencies of the resulting synthesis.

8.12.3 Physics Performance

The Hybrid demonstrator is examined for its ability to perform tracking efficiently in $t\bar{t}$+200PU events. The same samples are used throughout the performance evaluation for consistency with other developments. The Hybrid design achieves high efficiency and good resolution in software and hardware with relatively low duplicate fractions and χ^2 statistic as shown in Figure 8.21 to 8.24.

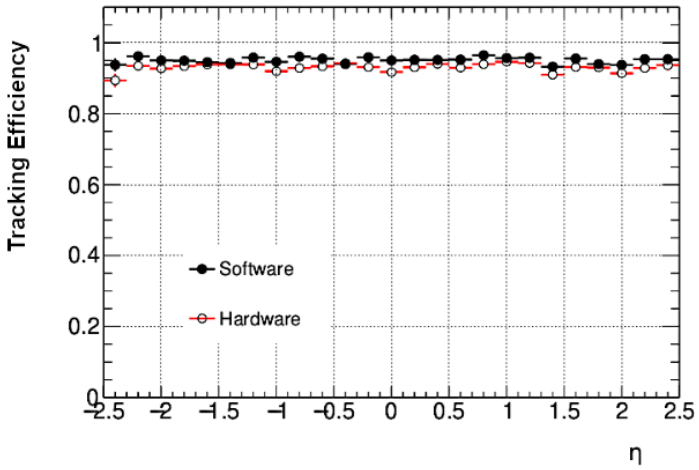

Figure 8.21: Hybrid tracking efficiency as a function of η.

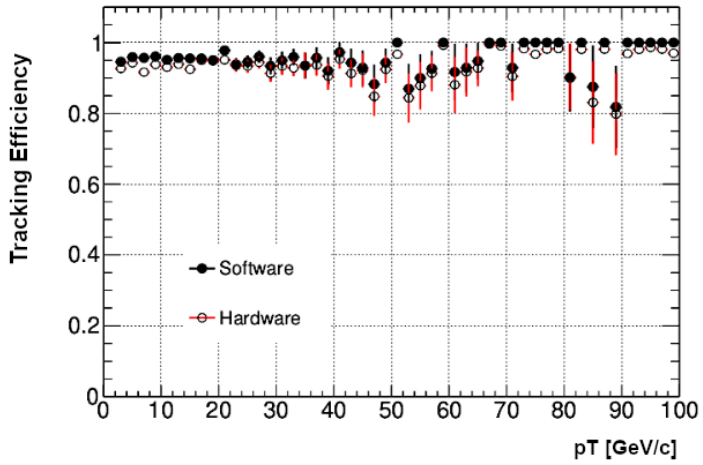

Figure 8.22: Hybrid tracking efficiency as a function of p_T.

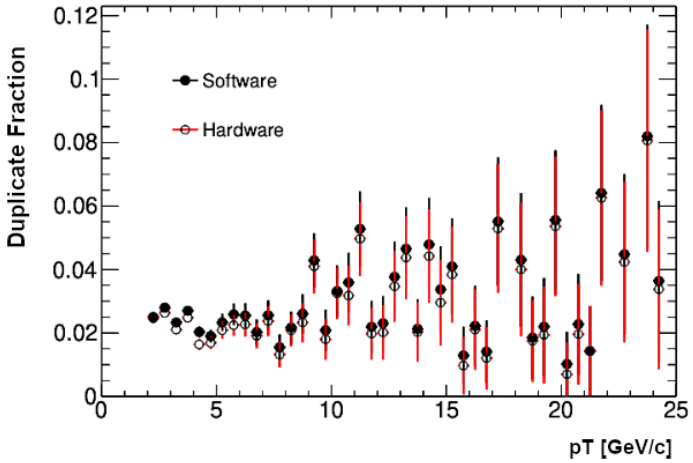

Figure 8.23: Hybrid duplicate fractions for $0 < p_T < 25$ GeV/c.

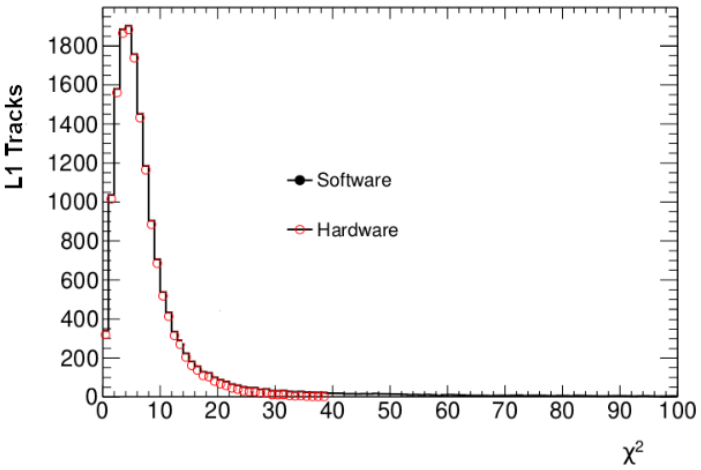

Figure 8.24: Hybrid χ^2 for matched tracks.

The efficiency in the η and the p_T are similar to other systems, however, the use of the Tracklet has resulted in fewer duplicate tracks and better agreements between the software and hardware in the χ^2 tests. As mentioned previously slight degradation is expected when representing native double precision values in software with their corresponding fixed-point values in hardware. Reducing the discrepancies requires further tuning of fixed-point variables at the runtime.

8.13 Latency and Resource Usage for All Modules

In the previous sections, the hardware resources and physics performance of MVLF, TMTT and Hybrid were considered separately. In this section, the latency and hardware resources are compared side by side for LUTs, FIFOs, BRAMs, DSPs, IOs and the Latency shown in Figure 8.25.

Figure 8.25: Latency and resource usage of MVLF, TMTT and Hybrid.

Deployment of linear regression algorithm in the Level-1 Track-Trigger design has led to a significant reduction in the utilisation of logic resources. As a result, the latency has decreased to comply with other readout devices for the CMS high luminosity upgrade. The main area of reduction belongs to the on-chip BRAM and IO slices. The data flow management and pipelining in MVLF have revamped the storage necessity for RAM usage to asynchronous registers by transmitting data to the components where it is needed, otherwise held in the latches for one clock cycle only. The use of AXI interfaces in block design has replaced single pin assignment to buses with standard communication protocol providing a register-like structure with reduced complexity. The feature can be used as a single-bit data transfer or burst mode which is suitable for high-throughput data processing of stub

streams. Other reductions are in DSP slice usage as a result of the custom-designed arithmetic logic unit with multiply/accumulate and asynchronous multipliers. Overall on-chip resources are reduced by floorplanning and optimising the area that modules use on the FPGA chip with minimised routing resources [2]. In Figure 8.26, total hardware resources for modules MVLF, TMTT and Hybrid is shown.

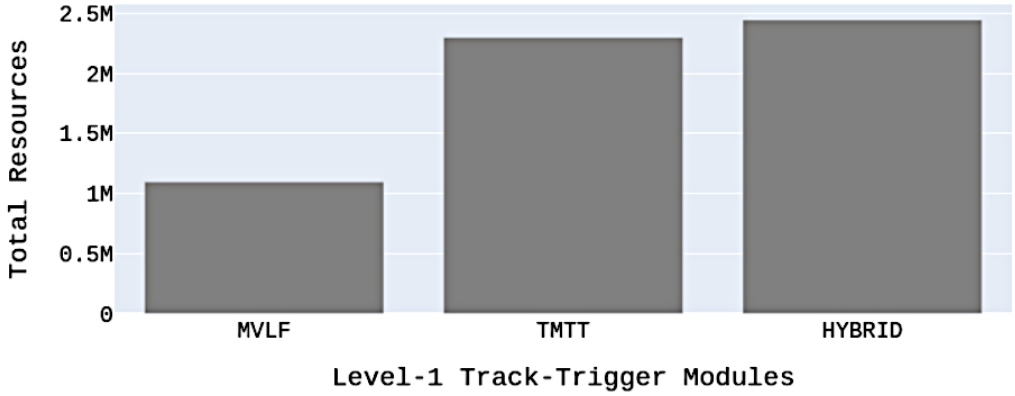

Figure 8.26: Total hardware resources for MVLF, TMTT and Hybrid.

8.14 Demonstrator Results

In the demonstrator, the software and hardware components of the modules that made up the Level-1 Track-Trigger are placed in a system to validate their performance. Several activities are conducted in the demonstrator that involves integrating tracking modules in the EMP framework, building and uploading firmware, simulation and emulation, and finally comparing the extensive results between different modules and prototypes. If a module under test passes the Level-1 Track-Trigger requirements, it is considered feasible for use in the upgraded CMS detector. Currently available modules (MVLF, TMTT, Hybrid) for a viable and cost-effective p_T trigger system have been considered. The aim is to assemble a Track-Trigger that performs well under extreme tracking conditions of high pileup within a latency of 3 μs and practical hardware resource usage.

[2]The more routing resources that are used, the slower the design will operate.

All combinations are examined with the same simulated event samples and firmware setup, minimising the external interference for achieving robust results. The effects of achieving lower latency are proportional to hardware resource usage in creating parallel structures or generating faster components that can be clocked at a higher frequency. The proposed Linear Fitter achieves this by dividing the tracker into π regions and further segmentation in the η and the ϕ sectors, time-multiplexed to process all regions in parallel. In addition, designing asynchronous application-specific custom-built arithmetic slices that can process stubs faster, combined with extremely fast μ-TCA technology to overcome the data flow timing constraints. The Linear Fitter is integrated with the demonstrator as the module for the track finding/fitting, and the results are presented accordingly.

The preliminary simulations indicated that the Linear Fitter demonstrator produces fewer genuine tracks than other systems. The initial prognostic was that the module is not sufficient for single module implementation in the tracking system. Therefore, the TMTT demonstrator was assembled to validate the results with help of Hough Transform. During two years of development, the Hybrid system gradually replaced the TMTT system. At this time, the Hybrid demonstrator was assembled. The Linear Fitter continued to produce fewer tracks in both TMTT and Hybrid systems. The simulations concluded that the Linear Fitter tends to over-filter genuine tracks. Overfiltering is the process of merging two candidate tracks that both are genuine into one track and as a result reducing the number of output tracks. As discussed in Section 6.5, the investigation into the cause pointed to the generation of virtual stubs within the stub sector boundaries. A feasible solution to this problem was the segmentation of π regions into smaller segments. The concluded simulation results indicated that many filtered tracks were recovered in final track reconstruction at additional costs of more I/O links and their associated logic resources. This problem was mitigated by introducing an Advanced eXtensible Interface (AXI) communication protocol [131] that is discussed in the Chapter 9.

As a result, the Linear Fitter performs within a maximum of ± 3 deviations from other systems in the physics simulation. In the hardware simulation, a significant reduction in logic resources is achieved.

The number of FPGA boards required to process two neighbouring nonants with the total latency for MVLF, TMTT and Hybrid modules is displayed in Table 8.6. The latency is somewhat different to the standalone emulations, as the EMP framework has difficulty scaling to system-level performance while maintaining the capability to run in real-time for FPGA simulations. In comparison to standalone simulations, the latency is slightly different as the EMP framework struggles to scale to system-level speed while maintaining the ability to run FPGA emulation in real-time.

architecture	FPGA boards	efficiency η (%)	efficiency p_T (%)	latency (ns)
MVLF	7	96.7	94.2	1472.86
TMTT	16	97.1	94.8	3263.06
HYBRID	18	96.5	95.1	3342.43

Table 8.6: Comparing tracking systems for a nonant.

In the next chapter, the scalability of Linear Fitter for the entire Track-Trigger system is discussed.

This page was intentionally left blank.

Chapter 9

Project Scaling and Summary

9.1 Demonstrator Migration

New hardware modules are integrated into the EMP demonstrator for testing and verification. Even though the EMP framework is designed with backward compatibility with the Xilinx Vivado, it has limited functionalities such as constraints in pipelining the modules and compatibility with Xilinx's new tool features for estimating power and efficient floorplanning. In addition, in the EMP demonstrator, the number of I/O links exceeds the maximum number of I/O pins on FPGA chips required for the more refined segmentation of the nonants and array partitioning for pipelining the design. The Xilinx platform extends the Vivado capability to AXI to consolidate an array of interfaces into one. The AXI is defined by ARM as part of the Advanced Microcontroller Bus Architecture standard [132] acting as a single bus that groups single data links into data buses. AXI-based system design [133] defines the key concept of the communication protocol for three types of interfaces: AXI4, AXI-Lite, and AXI4-Stream. The AXI4-Stream can transmit high-speed data streams through master/slave interfaces, with the master initiating the transactions and the slave responding to them. The main feature beneficial to TFP development is the AXI4-Stream FIFO core for memory mapping through a single bus architecture. The feature also reduces memory utilisation for storing input parameters in BRAMs for all I/Os. In addition, the optimisation of a TFP allows the development to be completely independent of the IP Bus-Builder protocol in the EMP framework. Hence the implementation is transferable to Vivado Design Suite or Vitis platform for new modules. The

components required for performance in Vivado Design Suite are shown in Figure 9.1. The CPU provides flexibility to select the combination of peripheral and interfaces to ensure smooth data flow from the Block RAM to the modules under test at the lowest cost possible.

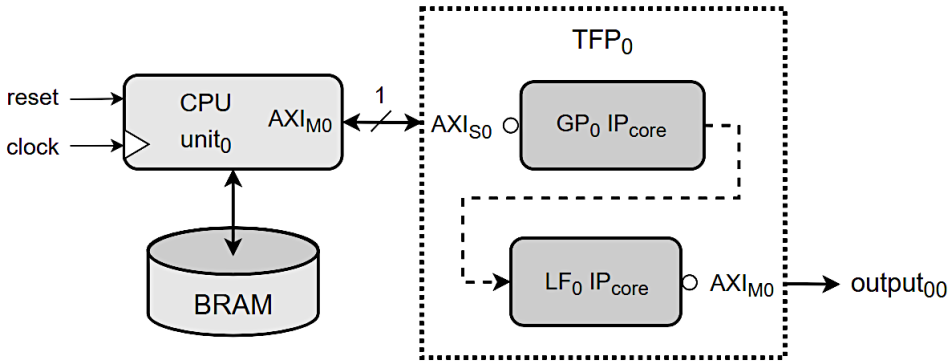

Figure 9.1: Module instantiation in the Xilinx Vivado design suite.

The AXI4-Stream implementation of the TFP blocks reduces the number of I/O links resulting in a decrease in the number of LUT slices by approximately 50% and BRAM slices by 50%; however, the number of FIFO is increased by about 25%. The optimisation benefits the design in two areas: choosing the larger value for the time-multiplexing factor and implementing fewer processors per nonant and eventually the entire system. In addition, redirecting the memory resources to the FIFOs makes the design suitable for the proposed Kintex KU15P device. A view of the AXI4-Stream implementation of the TFP is shown in Figure 9.2 with added components for managing the module reset and clock ports for one slice only. The GP and Linear Fitter modules are integrated into a block design sequentially. The input to the first module contains twelve interfaces or AXI bus-in ($0 \rightarrow 11$) generated by the first twelve stub sectors out of 48. The TFP block processes the stubs at 360 MHz frequency producing a track in 2.77 ns intervals. The output generated by the TFP is transmitted to the next Block Design through an AXI bus through the link ($0 \rightarrow 0$).

The maximum clock frequency is 480 MHz with the current configuration before timing violations are reported in the synthesis phase. In areas where the timing violations are too severe, a review of the algorithm and techniques in post-implementation needs to be evaluated before considering a faster target technology. The results from single iteration Linear Fitter indicate that the module can run at a higher frequency if the dependencies on BRAMs for storing stub/track parameters are removed.

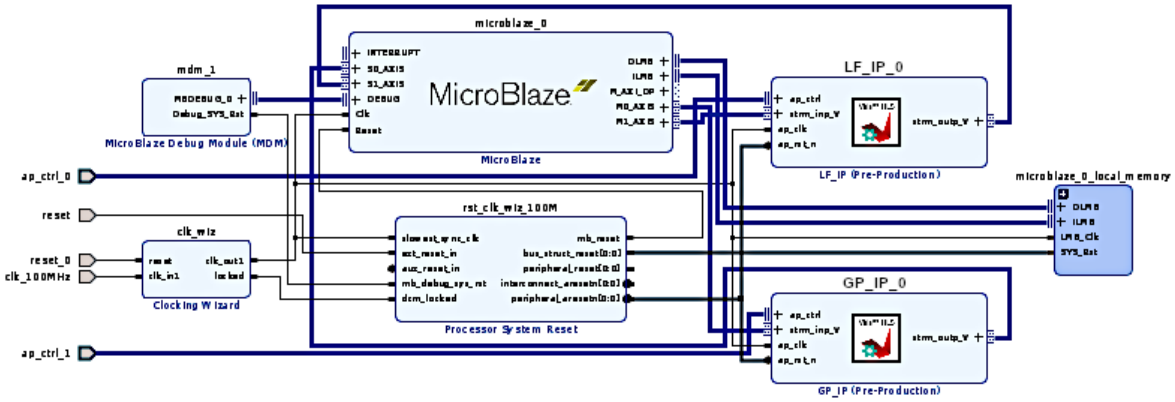

Figure 9.2: Module migration from EMP to Xilinx Vivado design suite.

The concept follows combinational sample-based modelling with fixed latency and centralised shared memory between each module to establish read/write operations. As the data are fed to the design as bursts, the AXI4-Lite can provide similar performance at a frequency higher than 480 MHz. An issue in this configuration is the memory bandwidth in the demonstrator, which must handle all I/O transitions at a higher frequency. Achieving higher speed is possible through the AXI Interconnect and the AXI BRAM Controller by providing multiple masters access to a local BRAM. The Xilinx Kintex UltraScale devices provide Double Data Rate 4 Synchronous Dynamic Random-Access Memory (DDR4 SDRAM) with support for up to 2666 Mb/s [134].

The components required for the implementation in Vivado are shown in Figure 9.3. Up to 64 TFP blocks can be added to this configuration trivially.

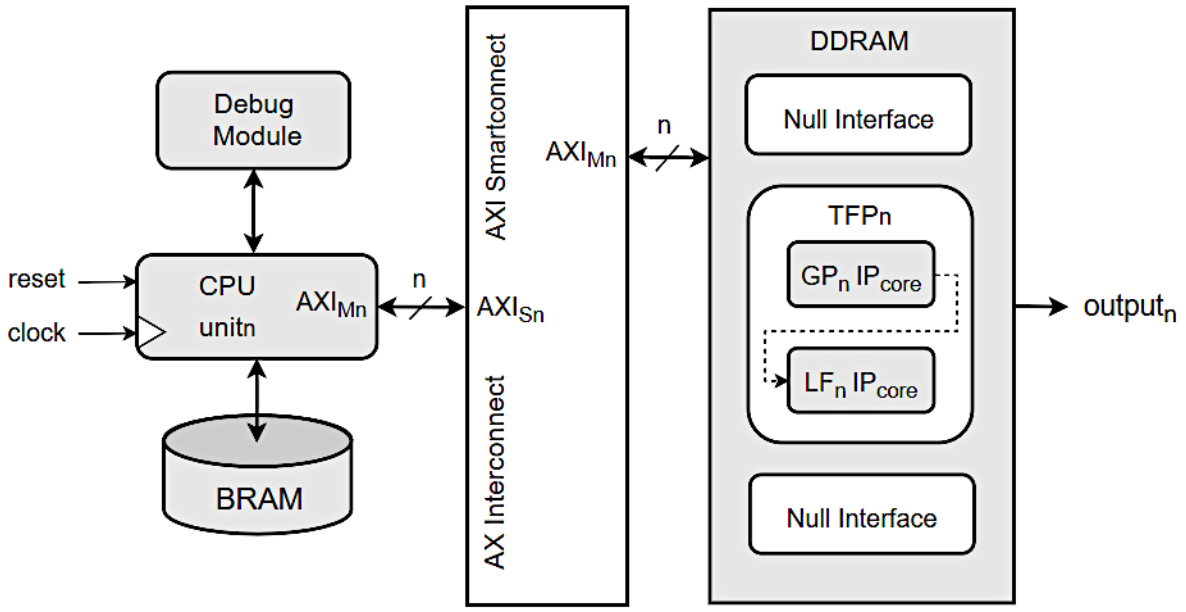

Figure 9.3: TFP implementation in Vivado Design Suite.

An implementation of the design indicates the Linear Fitter Single Iteration IP can run at a maximum frequency of 649 MHz with the longest path of 1.49 ns at an extra cost of 1024 kb DDRAM. The realisation is validated via Microblaze CPU [135] soft IP core and Xilinx Software Development Kit (XSDK) [136]. The TFP blocks are exported as self-contained IP cores and added to the IP Integrated block with general-purpose memories and logic fabrics. Additional components are required for integration into the design to manage the AXI interfaces, memory access and tracking modules through the Address Editor. In the address editor, each module is mapped to a portion of the memory with start and end addresses. The data flow is through the DDRAM with one read/write with an initiation interval of two. The initiation interval becomes less significant in this configuration as the blocks communicate through sophisticated handshake protocols. Once the master initiates the first transaction, the process continues until the slave returns the "done" signal. The master then takes control and moves to the next address block assigned to the next module.

The components required for the implementation in Vivado Design Suite are shown in Figure 9.3.

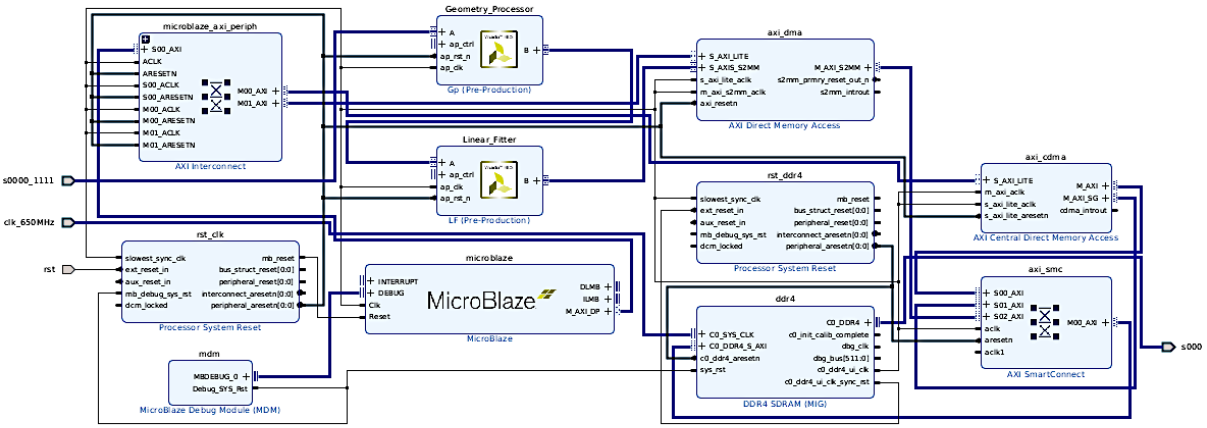

Figure 9.4: TFP implementation with DDRAM shared memory.

If additional modules are required, they are added to the system as slave interfaces by modifying the number of ports in the AXI interconnect and address editor for the automation tool to map all the necessary connections. The IP Integrator tool (industry-standard IP-XACT data formatter) [137] captures memory requirements and endpoint master/slave mapping to assign address segments to available memory spaces. For the testing phase of the design, a Microblaze can manage up to 32 modules as a deterministic real-time processing unit. The test vectors are written to textfiles and temporarily stored in volatile arrays to be interfaced to the Track Finder top-level block. The custom-designed subsystems such as the GP and the Linear Fitter are added hierarchically as pre-production IP packages containing the inner modules and a testbench to verify the basic functionality of AXI masters and AXI slaves. The system response is captured in the form of text files to allow comparisons to be made.

9.2 Scaling Projection

In theory, one FPGA board must handle all incoming transmissions from all tracker regions. In practice, the number of modules and the I/O links in the Linear Fitter system exceeds the available resources on one FPGA board. The latest regional and geometry segmentation produces 48 stub sectors per nonant. For a stub with seven parameters, 6,048 I/O pins are required with additional pins necessary for the implementation of the communication protocols. In the Linear Fitter system, the number of boards in a nonant is estimated at seven, with each handling maximum of twelve stub sectors. The description of the segmentation is shown in more detail in Appendix A3. In the design life cycle, following the software evaluation and hardware verification, the transition from Xilinx Virtex-7 to the relatively new Xilinx UltraScale+ FPGA chip has provided an increased link bandwidth up to 58 Gb/s transceivers with more peripherals. The number of available logic slices has been also considered to suit the TFP hardware needs. Table 9.1, shows the logic resource usage for the under development FPGA boards, including the estimated resources required for a single TFP block. The last row refers to the proposed FPGA device being utilised and tested in the Level-1 Track-Trigger demonstrator. The number of available LUT, RAM, FIFO and DSP slices makes the device suitable for high-performance computing and burst readout operations within the baseline budgeting.

Board	Chip	LUT	FIFO	BRAM	DSP	IO
Dev	A TFP Unit	514,954	343,792	694	228	224
MP7	Virtex X690T	633,200	866,200	1,470	3,600	600
Apollo	Virtex VU7P	788,160	176,320	1,440	4,560	832
Serenity	Kintex KU115	663,360	1,326,729	2,160	5,520	832
Proposed	Kintex KU15P	522,720	1,045,440	984	1,968	668

Table 9.1: Resources compared for a TFP unit and FPGA chips.

The number of LUT in the current development is very close to the maximum available resources on the proposed Kintex KU15P device. LUT slices are the main resources for implementing logic functionalities in FPGA

hardware development that are frequently used as static random-access memory cells. In the Linear Fitter design, a TFP is linked to twelve channels, each streaming 64-bit stub information from the DTC units. The number follows the time-multiplexing factor and the processing speed of a single TFP block. The channels are decoded into stub parameters from streams at the input. Before output, they are encoded back into streams utilising 229,379 or 43.88% of available LUT slices in the process. The number of FIFO slices in the TFP uses 32.88% of the available resources. Hence the design has been modified to account for SoC buses as the communication medium between modules redirecting the LUT resources to FIFO slices by replacing the twelve links with one channel per TFP.

9.3 Power Estimation

In Table 9.2, the static and dynamic power estimates for the MP7 chip at junction temperature[1] 60.0° C are shown. The MP7 requires 30–40 W, which is a significant amount of power to supply and heat to dissipate [138]. The generated temperature from power dissipation must ideally be kept below 50.0° C [139], which becomes extremely challenging for high-speed devices installed in a dense environment. The total power consumption increases with a faster clock cycle, more BRAM and input/output pins per TFP block and as a result temperature increases too. It will be beneficial to design software processors and cooling systems to monitor power and temperature periodically if a long life expectancy is desired.

frequency	static	clock	signals	logic	BRAM	DSP	I/O	total
240 MHz	340	807	629	297	470	301	1,290	4,134 (mW)
360 MHz	366	992	856	502	542	411	1,801	5,470 (mW)
480 MHz	391	1,342	1,145	782	876	581	2,234	7,351 (mW)
650 MHz	462	1,895	1,567	1,011	2,519	862	2,903	11,219 (mW)

Table 9.2: Power consumption for TFP blocks at different frequencies.

[1]Junction temperature includes the heat-sink temperature obtained from the MP7 datasheet.

9.4 Scaled Multistage Linear Fitter

In a Multivariate Linear Fitter design with 48 (η, ϕ) sectors and a time-multiplexing factor of twelve, a nonant is processed by seven FPGA chips. For the entire system of nine nonants, 64 (63 + 1) chips will be sufficient to process the stubs from all tracker regions. The processors are labelled 0→62, beginning with the track finder processing 0→2π with the 64^{th} device processing output from all nonants, if required. The connectivity between the devices will be via chip-to-chip mid-board optical cables at a rate up to 28 Gb/s over a maximum 100 m distance. The transmitters TX and RX are configured as 12-channel unidirectional optical channels through optical Firefly modules. Figure 9.4 shows nonants (1→9) with 64 (0→63) TFP blocks and identification labels.

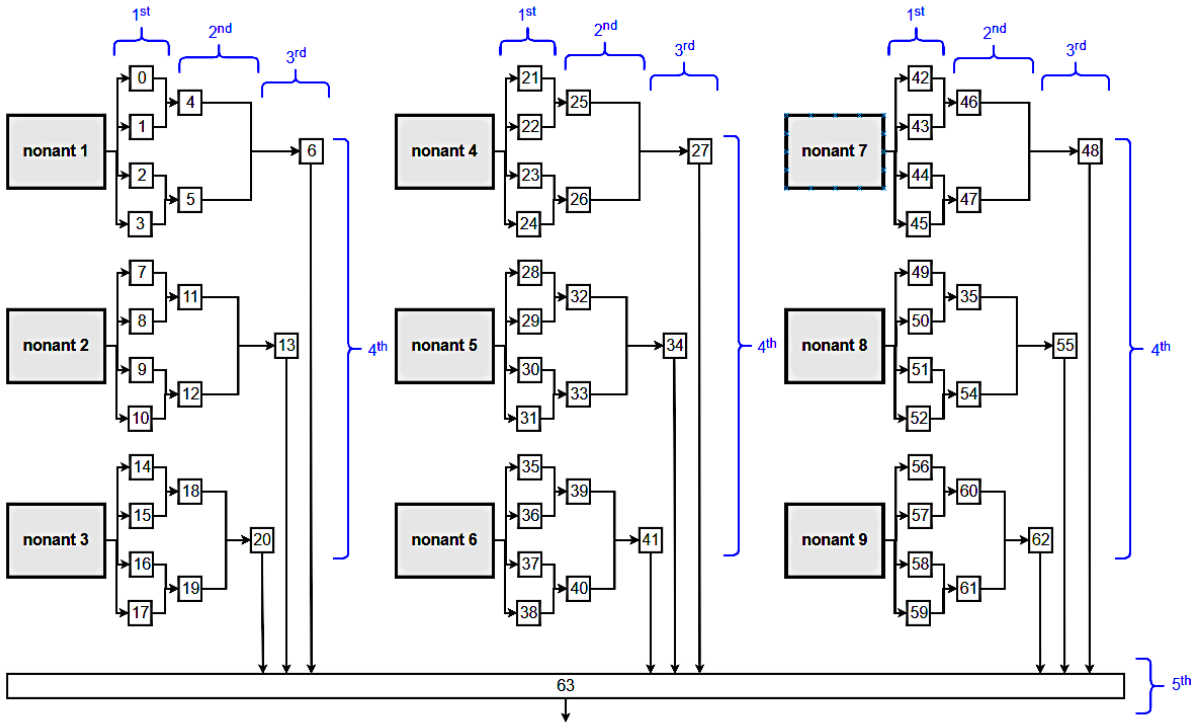

Figure 9.5: The 5-stage Linear-Fitter for the entire Tracker.

The architecture is fully reconfigurable in operational frequency settings up to 480 MHz and the number of desired TFP blocks. In this configuration, the processing is done in five stages with the first stage, the FPGA chips receive a maximum of twelve links as I/O, channel one. In the second, third

and fourth stages, the chips receive two channels left and right or up and down and produce one output. In the final stage, the last chip receives nine inputs and produces one output. All chips are connected to the global clock at a 40 MHz frequency through the local clock dividers to generate transitions with phase offset. The data transition takes place at 25 ns intervals, with the longest path of 1,333.25 ns. This is the time required for at least one fitted track per nonant to arrive at the input of the last TFP. The final processor selects the track with the χ^2 smaller than the pre-defined significance level from nine nonants at 750 kHz frequency. In the final system, the latency of the final TFP is added to the total latency which will be approximately 2,666.5 ns.

The templated algorithm can be trivially modified to incorporate new devices at the desired frequency in the Vivado demonstrator without dependency on the EMP framework. The effects of modifications can be automatically cross-referenced between hardware and software as the algorithm is directly linked to the CMSSW framework. At this stage, it is beneficial to investigate various tunings of the residual weights in new kinematic cut regions and to study the effect of using fewer bits to represent the modified parameters in hardware.

9.5 Final System Estimates

The floor planning has improved the performance of the TFP by grouping connectivity and manually placing them on the chip to meet the timing. In TFP design, two floorplanning techniques [140], Cell-Levels and Hierarchical, are used to investigate better logic placement to reduce route delays. In Cell-Level, the critical path in the architecture is identified, and the corresponding logic elements are hand-placed on a specific site in the chip. In Hierarchical, the hierarchy levels are placed on different chip regions to achieve the best timing result. Cell-Level placement becomes increasingly tricky in complex systems and does not guarantee solving timing issues even after modifying the RTL. In the Hierarchical approach, the modules are placed very close to one another on a small chip region. The process is iterative and often provides good results if the critical path does not span more than one hierarchical module. The Vivado interactive design

environment [141] provides floorplanning assistance by visualisation of I/O pins and clocked logic resources by simultaneously displaying the available packages next to the device view. This capability is particularly helpful in designing a null interface by assigning I/O banks and clock regions to both the physical package and internal die before adding the subsystem to the chip. Other benefits include examining the chip resources in various ways to optimise the *place&route* phase, differentiating between external and internal connectivity, and building relationships based on proximity to internal resources. In the early stages of the development, if the device for the end product is unknown for pin planning, the assignment is done using alternate devices by allocating I/O across multiple viable chips. The technique has been beneficial in removing obstructions in the Level-1 Track-Trigger module development as the optimisations aim to reduce the number of FPGA chips over time. In Figure 9.4, the package and device view of the MP7 (xc7vx690tffg1157-1) after pin configuration and floorplanning for a TFP is shown.

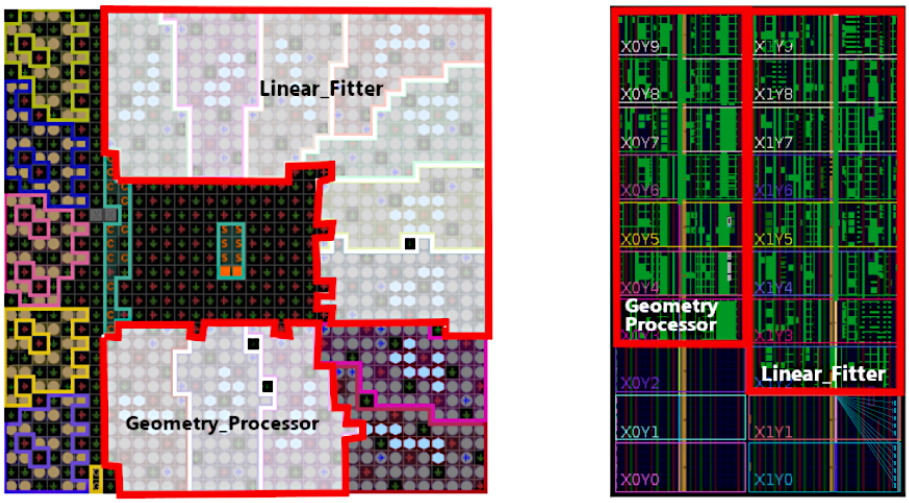

Figure 9.6: The MP7 package (left) and the device view (right).

The optimisations have not achieved the ultimate goal of placing two TFPs on a chip, however, it has made a total of 13,184 slices (I/O, DSP, LUT, RAM) available for additional hardware implementation, if required. At this stage, further investigation of the design can determine which hierarchy is most suitable for floorplanning by identifying the timing issues of each block separately.

Total on-chip power for a single device is estimated from the implemented netlist and activity derived from the constraints files, simulation files or vector less analysis [141] at two stages. The static power is usually estimated by programming the FPGA chip with an empty bitstream to record the response of the transistor leakage in various voltages and temperatures. The dynamic power is calculated by the summation of all static powers with I/O activities and levels of logic routing in the design. Off-chip power consumption is also generated from external power sources and interlinks to FPGA boards through I/O pins buffers and LEDs, independent of the chip. Essentially the toggle rate at which the clocked elements change status from on to off and vice versa determines the dynamic power consumption at the device level. The Xilinx Power Estimator (XPE) [142] allows an individual or group power estimation by accounting for slices utilisation and the probability of switch rates at various clocking frequencies. Faster clock activities result in higher dynamic power consumption. It is worth reiterating that if FPGA power consumption is not measured accurately and managed effectively, it can result in very high power dissipation, which correlates with high power temperature.

9.6 Summary

A firmware demonstrator has been assembled to prove the feasibility of a complexity-reduced hardware-based Track-Trigger for the CMS High-Luminosity LHC upgrade. The system will reconstruct tracks within a 3 μs latency to comply with electronic readout devices developed for 40 MHz event generation. Track reconstruction is an important addition to the future of the Track-Trigger system to select only particles with high transverse momentum in high pileup events. This not only increases the probability of exciting discoveries, but also reduces the data volume significantly. The Track-Trigger was implemented for FPGA-based devices using custom-designed hardware modules for real-time data transmission and data processing during CMS data-tacking phase. The demonstrator implements a Geometric Processor module for coarsely segmenting the Tracker into regions, followed by a Linear Fitter module to reconstruct charged particle trajectories using logic design methodologies and a hardware acceleration platform. The development was successfully demonstrated in the simulation of track

finding and track fitting for the particles with $p_T \geq 2$ GeV/c within specified latency and hardware resource usage under extreme conditions of data containing high pileup events. The candidate tracks are used to trigger a capture signal to the trigger system to keep or discard the corresponding event. For the evaluation stage, past and current Track-Trigger systems have been considered as the baseline metrics for comparing the system performance. In physics simulation, the system performs similarly in most aspects of track reconstruction with an efficiency $\pm 2\%$ of precision, whereas, a significant reduction in latency and the number of FPGA devices required for the entire system is observed in hardware emulation. The achievement is accomplished by designing several custom-designed application-specific logic slices, algorithm optimisation at Register Transfer-Level architecture and High-Level Synthesis acceleration techniques in FPGA technology. A significant reduction in latency and hardware resource usage is expected with further optimisation of the planned system and foreseen hardware accelerator technologies.

This page was intentionally left blank.

Bibliography

[1] Zyla et al. “Review of Particle Physics”. In: *PTEP* 2020.8 (2020), pp. 186–190. DOI: 10.1093/ptep/ptaa104.

[2] Demarteau et al. “Particle and nuclear physics instrumentation and its broad connections”. In: *Reviews of Modern Physics* 88 (Dec. 2016), pp. 20–34. DOI: 10.1103/RevModPhys.88.045007.

[3] European Council for Nuclear Research. *CERN*. Dec. 2021. URL: https://home.cern/.

[4] Lyndon Evans and Philip Bryant. “LHC Machine”. In: *JINST* 3 (2008), pp. 164–166. DOI: 10.1088/1748-0221/3/08/S08001. URL: https://cds.cern.ch/record/1129806.

[5] CMS Collaboration. *The Phase-2 Upgrade of the CMS Tracker*. Tech. rep. Geneva: CERN, June 2017. DOI: 10.17181/CERN.QZ28.FLHW. URL: http://cds.cern.ch/record/2272264.

[6] Contardo et al. *Technical Proposal for the Phase-II Upgrade of the CMS Detector*. Tech. rep. Geneva, June 2015. DOI: 10.17181/CERN.VU8I.D59J. URL: https://cds.cern.ch/record/2020886.

[7] Chatrchyan et al. “The CMS experiment at the CERN LHC. The Compact Muon Solenoid experiment”. In: *JINST* 3 (2008), pp. 11–32. DOI: 10.1088/1748-0221/3/08/S08004. URL: https://cds.cern.ch/record/1129810.

[8] The German Tunnelling Committee (DAUB). *CERN’s LEP - Large Electron Positron Storage Ring*. 2021. URL: https://www.daub-ita.de/en/tunnel-projects/schweiz/cerns-lep-large-electron-positron-storage-ring/.

[9] Benedikt et al. *LHC Design Report*. CERN Yellow Reports: Monographs. Geneva: CERN, 2004. DOI: 10.5170/CERN-2004-003-V-3. URL: https://cds.cern.ch/record/823808.

[10] Fabienne Marcastel. “CERN’s Accelerator Complex. La chaîne des accélérateurs du CERN”. In: (Oct. 2013). General Photo. URL: https://cds.cern.ch/record/1621583.

[11] Tai Sakuma. “Cutaway diagrams of CMS detector”. In: (May 2019). General Photo. URL: https://cds.cern.ch/record/2665537.

[12] Cittolin et al. “CMS Photo Book "The Compact Muon Solenoid Experiment at the LHC"”. In: (Feb. 2012). General Photo. URL: https://cds.cern.ch/record/2629368.

[13] Karimäki et al. *The CMS tracker system project: Technical Design Report*. Technical design report. CMS. Geneva: CERN, 1997. URL: https://cds.cern.ch/record/368412.

[14] CMS Collaboration. *The CMS electromagnetic calorimeter project: Technical Design Report*. Technical design report. CMS. Geneva: CERN, 1997. URL: https://cds.cern.ch/record/349375.

[15] CMS Collaboration. *The CMS hadron calorimeter project: Technical Design Report*. Technical design report. CMS. Geneva: CERN, 1997. URL: https://cds.cern.ch/record/357153.

[16] CMS Collaboration. *The CMS magnet project: Technical Design Report*. Technical design report. CMS. Geneva: CERN, 1997. DOI: 10.17181/CERN.6ZU0.V4T9. URL: https://cds.cern.ch/record/331056.

[17] J. G. Layter. *The CMS muon project: Technical Design Report*. Technical design report. CMS. Geneva: CERN, 1997. URL: https://cds.cern.ch/record/343814.

[18] G Apollinari, I Béjar Alonso, O Brüning, et al. *High-Luminosity Large Hadron Collider (HL-LHC): Preliminary Design Report*. CERN Yellow Reports: Monographs. Geneva: CERN, 2015. DOI: 10.5170/CERN-2015-005. URL: https://cds.cern.ch/record/2116337.

[19] CMS Collaboration. *The Phase-2 Upgrade of the CMS DAQ Interim Technical Design Report*. Tech. rep. Geneva: CERN, Sept. 2017. URL: https://cds.cern.ch/record/2283193.

[20] "Arcidiacono et al." “The 2 Tbps data to surface system of the CMS data acquisition”. In: July 2005, pp. 5–8. ISBN: 0-7803-9183-7. DOI: 10.1109/RTC.2005.1547433.

[21] Sotiropoulou et al. “The Associative Memory System Infrastructures for the ATLAS Fast Tracker”. In: *IEEE Transactions on Nuclear Science* 64.6 (2017), pp. 10–19. DOI: 10.1109/TNS.2017.2703908.

[22] Collaboration ATLAS. *Technical Design Report for the Phase-II Upgrade of the ATLAS Trigger and Data Acquisition System - Event Filter Tracking Amendment*. Tech. rep. 2022. DOI: 10.17181/CERN.ZK85.5TDL. URL: http://cds.cern.ch/record/2802799.

[23] Kluge et al. “Design, production and first operation of the ALICE Silicon Pixel Detector system”. In: (2008). DOI: 10.5170/CERN-2008-008.205. URL: http://cds.cern.ch/record/1158616.

[24] I Ravasengaon. “The upgrade of the ALICE Inner Tracking System at the CERN LHC”. In: *Nuovo Cimento C* 42 (2019), pp. 55–59. DOI: 10.1393/ncc/i2019-19068-y. URL: https://cds.cern.ch/record/2729174.

[25] LHCb Collaboration. *LHCb VELO Upgrade Technical Design Report*. Tech. rep. Nov. 2013. URL: http://cds.cern.ch/record/1624070.

[26] LHCb Collaboration. *LHCb PID Upgrade Technical Design Report*. Tech. rep. Nov. 2013. URL: http://cds.cern.ch/record/1624074.

[27] The CMS Collaboration. “CMS Physics Technical Design Report, Volume II: Physics Performance”. In: *Journal of Physics G: Nuclear and Particle Physics* 34.6 (Apr. 2007), pp. 3–45. DOI: 10.1088/0954-3899/34/6/s01. URL: https://doi.org/10.1088/0954-3899/34/6/s01.

[28] Andrew Boutros and Vaughn Betz. “FPGA Architecture: Principles and Progression”. In: *IEEE Circuits and Systems Magazine* 21 (2021), pp. 4–29. DOI: 10.1109/MCAS.2021.3071607.

[29] Sirunyan et al. In: 15.09 (Sept. 2020), pp. 23–41. DOI: 10.1088/1748-0221/15/09/p09018. URL: https://doi.org/10.1088/1748-0221/15/09/p09018.

[30] Alessandro La Rosa. *The upgrade of the CMS Tracker at HL-LHC*. Tech. rep. Geneva: CERN, Oct. 2020. DOI: 10.7566/JPSCP.34.010006. arXiv: 2102.06074. URL: http://cds.cern.ch/record/2776511.

[31] Raymond et al. “The CMS binary chip for microstrip tracker readout at the SLHC”. In: *Journal of Instrumentation* 7 (Jan. 2012), p. C01033. DOI: 10.1088/1748-0221/7/01/C01033.

[32] The CMS Collaboration. "Description and performance of track and primary-vertex reconstruction with the CMS tracker". In: *Journal of Instrumentation* 9.10 (Oct. 2014), pp. 16–19. DOI: 10.1088/1748-0221/9/10/p10009. URL: https://doi.org/10.1088/1748-0221/9/10/p10009.

[33] Sirunyan et al. "Performance of the CMS Level-1 trigger in proton-proton collisions at $\sqrt{s} = 13$,TeV". In: 15 (2020), pp. 117–221. DOI: 10.1088/1748-0221/15/10/P10017.

[34] Khachatryan et al. "CMS tracking performance results from early LHC operation". In: *European Physical Journal C* 70 (Apr. 2010), pp. 5–19. DOI: 10.1140/epjc/s10052-010-1491-3.

[35] Louise Skinnari. *L1 Track Triggering at CMS for High Luminosity LHC*. Tech. rep. CERN, 2014. DOI: 10.1088/1748-0221/9/10/C10035. URL: https://cds.cern.ch/record/1746267.

[36] CMS Collaboration. *The Phase-2 Upgrade of the CMS Tracker*. Tech. rep. Geneva: CERN, June 2017. DOI: 10.17181/CERN.QZ28.FLHW. URL: https://cds.cern.ch/record/2272264.

[37] CMS Collaboration. *CMS Binary Chip Documentation*. 2021. URL: https://www.hep.ph.ic.ac.uk/ASIC/CBC_documentation/.

[38] G Hall, M Raymond, and A Rose. "2-D PT module concept for the SLHC CMS tracker". In: 5 (2010). DOI: 10.1088/1748-0221/5/07/c07012. URL: https://doi.org/10.1088/1748-0221/5/07/c07012.

[39] Data Acquisition System Collaboration. *XDAQ CMS Online Software*. 2021. URL: https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOS.

[40] Nodari et al. *First results from the CIC data aggregation ASIC for the Phase 2 CMS Outer Tracker*. Tech. rep. Geneva: CERN, Oct. 2019. URL: https://cds.cern.ch/record/2780281.

[41] Lutz Werner Feld and Gadekl. *First Implementation of a Two-Stage DC-DC Conversion Powering Scheme for the CMS Phase-2 Outer Tracker*. Tech. rep. 2016, pp. 1–64. DOI: 10.1088/1748-0221/12/03/C03090. URL: https://cds.cern.ch/record/2233163.

[42] Isobel Falconer. "J J Thomson and the discovery of the electron". In: 32 (1997), pp. 226–231. DOI: 10.1088/0031-9120/32/4/015. URL: https://doi.org/10.1088/0031-9120/32/4/015.

[43] The CMS collaboration. “Alignment of the CMS tracker with LHC and cosmic ray data”. In: *Journal of Instrumentation* 9.06 (June 2014), pp. 32–57. DOI: 10.1088/1748-0221/9/06/p06009. URL: https://doi.org/10.1088/1748-0221/9/06/p06009.

[44] Bauerdick et al. “Multiple-view, Multiple-selection Visualization of Simulation Geometry in CMS”. In: *Journal of Physics: Conference Series* 396.2 (Dec. 2012), pp. 11–12. DOI: 10.1088/1742-6596/396/2/022052. URL: https://doi.org/10.1088/1742-6596/396/2/022052.

[45] M. Jeitler. “The Phase-2 Upgrade of the Hardware Trigger of CMS at the LHC”. In: *Journal of Instrumentation* 15.09 (Sept. 2020), p76–121. DOI: 10.1088/1748-0221/15/09/c09009. URL: https://doi.org/10.1088/1748-0221/15/09/c09009.

[46] Sirunyan et al. “Performance of missing transverse momentum reconstruction in proton-proton collisions at $\sqrt{s = 13}$ TeV using the CMS detector”. In: *Journal of Instrumentation* 14 (July 2019), p23–24. DOI: 10.1088/1748-0221/14/07/P07004.

[47] Butler et. al. *CMS Phase II Upgrade Scope Document*. Tech. rep. Geneva: CERN, Sept. 2015. URL: https://cds.cern.ch/record/2055167.

[48] Bartz et al. *FPGA-Based Approach to Level-1 Track Finding at CMS for the HL-LHC*. Tech. rep. Geneva: CERN, Apr. 2017. URL: https://cds.cern.ch/record/2262935.

[49] Valentin Kuznetsov. *Data Aggregation System Documentation, Release development*. Feb. 2014. URL: https://readthedocs.org/projects/dasdev-vidma/downloads/pdf/latest/.

[50] CMS Collaboration. *The Phase-2 Upgrade of the CMS Muon Detectors*. Tech. rep. Geneva: CERN, Sept. 2017. URL: https://cds.cern.ch/record/2283189.

[51] Duccio Abbaneo. *Performance requirements for the Phase-2 Tracker Upgrades for ATLAS and CMS*. Tech. rep. CERN, 2016. DOI: 10.1051/epjconf/201612700002. URL: https://cds.cern.ch/record/2208302.

[52] CMS Collaboration. *The Phase-2 Upgrade of the CMS Muon Detectors*. Tech. rep. Geneva: CERN, Sept. 2017. URL: https://cds.cern.ch/record/2283189.

[53] Nodari et al. “First results from the CIC data aggregation ASIC for the Phase 2 CMS Outer Tracker”. In: 2019 (2020), p102–103. DOI: 10.22323/1.370.0102. URL: https://cds.cern.ch/record/2725207.

[54] A Tapper and Darin Acosta. *CMS Technical Design Report for the Level-1 Trigger Upgrade*. Tech. rep. June 2013. URL: https://cds.cern.ch/record/1556311.

[55] Zhengcheng Tao. *Level-1 Track Finding with an all-FPGA system at CMS for the HL-LHC*. Tech. rep. Geneva: CERN, Nov. 2018. URL: https://cds.cern.ch/record/2648960.

[56] Giovanni Bianchi. *tkLayout: a Design Tool for Innovative Silicon Tracking Detectors*. Tech. rep. CERN, 2013. DOI: 10.1088/1748-0221/9/03/C03054. URL: https://cds.cern.ch/record/1640926.

[57] Nicoletta De Maio Giovanni Bianchi. 2020. URL: https://cms-tklayout.web.cern.ch/cms-tklayout/layouts/cmssw-models/Q_OT616_200_IT613_MB_2019_04/layout.html.

[58] L. Skinnari. “L1 track triggering at CMS for High Luminosity LHC”. In: *Journal of Instrumentation* 9.10 (Oct. 2014), p123–132. DOI: 10.1088/1748-0221/9/10/c10035. URL: https://doi.org/10.1088/1748-0221/9/10/c10035.

[59] G Benelli, the CMS Offline, and Computing Projects. “The CMSSW benchmarking suite: Using HEP code to measure CPU performance”. In: 219.5 (2010), p166–211. DOI: 10.1088/1742-6596/219/5/052016. URL: https://doi.org/10.1088/1742-6596/219/5/052016.

[60] Mostafa Abd-El-Barr and Hesham El-Rewini. “Pipelining Design Techniques”. In: Jan. 2005, p185–216. ISBN: 9780471467410. DOI: 10.1002/0471478326.ch9.

[61] M Jeitler. “The upgrade of the CMS trigger system”. In: 9 (2014), p143–165. DOI: 10.1088/1748-0221/9/08/c08002. URL: https://doi.org/10.1088/1748-0221/9/08/c08002.

[62] Xilinx Inc. *Vivado Design Suite User Guide*. May 2021. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug902-vivado-high-level-synthesis.pdf.

[63] N.P. Ghanathe, A. Madorsky, H. Lam, et al. “Software and firmware co-development using high-level synthesis”. In: *Journal of Instrumentation* 12.01 (Jan. 2017), p65–94. DOI: 10.1088/1748-0221/12/01/c01083. URL: https://doi.org/10.1088/1748-0221/12/01/c01083.

[64] Tomalin et al. *An FPGA based track finder for the L1 trigger of the CMS experiment at the High Luminosity LHC*. Tech. rep. Geneva: CERN, Jan. 2017. DOI: 10.1088/1748-0221/12/12/P12019. URL: https://cds.cern.ch/record/2287640.

[65] Cosmo et al. “MicroTCA and AdvancedTCA equipment evaluation and customization for LHC experiments”. In: 10 (2015), pp. 34–45. DOI: 10.1088/1748-0221/10/01/C01008.

[66] Xavi Fernandez-Tejero. “Design and Optimization of Advanced Silicon Strip Detectors for High Energy Physics Experiments”. 2020. URL: https://cds.cern.ch/record/2722118.

[67] G. Hall. “A time-multiplexed track-trigger for the CMS HL-LHC upgrade”. In: 824 (2016), pp. 292–295. DOI: https://doi.org/10.1016/j.nima.2015.09.075. URL: https://www.sciencedirect.com/science/article/pii/S0168900215011304.

[68] Thomas Schuh. “Entwicklung des CMS-Spurtriggers für den Hochluminositätsbetrieb des Large Hadron Colliders”. German. PhD thesis. 2018. DOI: 10.5445/IR/1000079109.

[69] Xilinx Inc. *Kintex UltraScale FPGAs Data Sheet: DC-AC Switching Characteristics*. 2021. URL: https://www.xilinx.com/support/documentation/data_sheets/ds892-kintex-ultrascale-data-sheet.pdf.

[70] R. Frühwirth. “Application of Kalman filtering to track and vertex fitting”. In: 262 (1987), pp. 44–50. DOI: https://doi.org/10.1016/0168-9002(87)90887-4. URL: https://www.sciencedirect.com/science/article/pii/0168900287908874.

[71] The Institute of Electrical and Inc. Electronics Engineers. *IEEE Standard Multi-value Logic System for VHDL Model Interoperability (Std-logic-1164)*. 2021. URL: https://perso.telecom-paristech.fr/guilley/ENS/20171205/TP/tp_syn/doc/IEEE_VHDL_1164-1993.pdf.

[72] Bartz et al. “FPGA-based tracking for the CMS Level-1 trigger using the tracklet algorithm”. In: *JINST* 15 (Oct. 2019), pp. 11–19. DOI: 10.1088/1748-0221/15/06/P06024. arXiv: 1910.09970. URL: https://cds.cern.ch/record/2696252.

[73] Luigi Calligaris. *Trigger level track reconstruction in CMS with a fully time-multiplexed architecture using a Hough transform implemented in an FPGA*. Tech. rep. Geneva: CERN, Nov. 2016. DOI: 10.22323/1.282.1000. URL: https://cds.cern.ch/record/2233165.

[74] LiLonghai Y.Weixin. “A New Regression Model: Modal Linear Regression”. In: *Scandinavian journal of statistics* 41 (2014), pp. 6–23. DOI: https://doi.org/10.1111/sjos.12054.

[75] Ifan Hughes and Thomas P. A. Hase. *Measurements and their uncertainties: a practical guide to modern error analysis*. New York, 2010. URL: https://ebookcentral.proquest.com/lib/brunelu/detail.action?docID=584562.

[76] EMP Collaboration. *The EMP framework provides common infrastructural firmware*. 2019. URL: https://serenity.web.cern.ch/serenity/emp-fwk/Emp-FwkGuide.pdf.

[77] CMS Collaboration. *Data Aggregation System*. 2021. URL: https://cmsweb.cern.ch/das/.

[78] Valentin Kuznetsov. *DAS, version development DAS QUERY LANGUAGE*. 2021. URL: https://das.readthedocs.io/en/docs_reorganize/das_ql.html.

[79] CERN Collaboration. *Scientific Data Management System Rucio*. 2021. URL: https://rucio.cern.ch/.

[80] Physics Experiment Data Export Collaboration. *PhEDEx*. 2021. URL: https://twiki.cern.ch/twiki/bin/view/CMSPublic/PhEDEx.

[81] Claire Savard for CMS collaboration. *Track quality machine learning models on FPGAs for the CMS Phase 2 Level 1 trigger*. 2020. URL: https://indico.cern.ch/event/924283/contributions/4105200/attachments/2153984/3632646/Fast_ML_v2.pdf.

[82] Ghabrous et al. “IPbus: a flexible Ethernet-based control system for xTCA hardware”. In: *Journal of Instrumentation* 10.02 (Feb. 2015), pp. 26–35. DOI: 10.1088/1748-0221/10/02/c02019. URL: https://doi.org/10.1088/1748-0221/10/02/c02019.

[83] GitLab Inc. *Continuous Integration*. 2021. URL: https://docs.gitlab.com/ee/ci/pipelines/.

[84] *Xilinx Inc, 7 Series FPGAs Data Sheet: Overview*. 2020. URL: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf.

[85] *Xilinx Inc, UltraScale Architecture and Product Data Sheet: Overview*. 2020. URL: https://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf.

[86] Xilinx Inc. *Vitis Unified Software Platform*. 2021. URL: https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html.

[87] *MATLAB Inc, Math. Graphics. Programming.* Aug. 2020. URL: https://uk.mathworks.com/products/matlab.html.

[88] *Simulink is for Model-Based Design*. Aug. 2020. URL: https://uk.mathworks.com/products/simulink.html.

[89] S. Summers, A. Rose, and P. Sanders. “Using MaxCompiler for the high level synthesis of trigger algorithms”. In: 12 (2017). DOI: 10.1088/1748-0221/12/02/c02015. URL: https://doi.org/10.1088/1748-0221/12/02/c02015.

[90] Xilinx Inc. *System Generator for DSP*. 2021. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/sysgen_user.pdf.

[91] MathWorks Inc. *Fixed-Point Designer*. 2021. URL: https://uk.mathworks.com/products/fixed-point-designer.html.

[92] MathWorks Inc. *HDL Designer*. 2021. URL: https://uk.mathworks.com/products/hdl-coder.html.

[93] David Sabes. “L1 track triggering with associative memory for the CMS HL-LHC tracker”. In: *Journal of Instrumentation* 9 (May 2014), pp. 9–17. DOI: 10.1088/1748-0221/9/11/C11014.

[94] R E Kalman. “A new approach to linear filtering and prediction problems”. In: *Trans. ASME, D* 82 (1960), pp. 35–44. URL: https://cds.cern.ch/record/434680.

[95] Mahshad Valipour and Luis A. Ricardez-Sandoval. “Assessing the Impact of EKF as the Arrival Cost in the Moving Horizon Estimation under Nonlinear Model Predictive Control”. In: *Industrial & Engineering Chemistry Research* 60.7 (2021), pp. 2994–3012. DOI: 10.1021/acs.iecr.0c06095. eprint: https://doi.org/10.1021/acs.iecr.0c06095. URL: https://doi.org/10.1021/acs.iecr.0c06095.

[96] Tom James. “A hardware track-trigger for CMS at the High Luminosity LHC”. PhD thesis. 2018. DOI: https://doi.org/10.25560/60593. URL: http://hdl.handle.net/10044/1/60593.

[97] Yaakov bar-shalom, X.-Rong Li, and Thia Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*. Jan. 2004. ISBN: 047141655X. DOI: 10.1002/0471221279.ch11.

[98] Richard F. Gunst and Robert L. Mason. *Regression analysis and its application*. Vol. 34. Dekker, 1980. URL: https://doi.org/10.1201/9780203741054.

[99] Cieri et al. *L1 Track Finding for a Time Multiplexed Trigger*. Tech. rep. Geneva: CERN, June 2015. DOI: 10.1016/j.nima.2015.09.117. URL: http://cds.cern.ch/record/2027711.

[100] Demirsoy, Suleyman Sirri, Kale, Izzet. “Reconfigurable Multiplier Blocks: Structures, Algorithm and Applications”. In: (2007). DOI: 10.1007/s00034-007-9005-8. URL: https://rdcu.be/chsTy.

[101] Götz Trenkler. “Methods and Applications of Linear Models — Regression and the Analysis of Variance”. In: *Computational Statistics & Data Analysis - CS&DA* 26 (Jan. 1998), pp. 378–379. DOI: 10.1016/S0167-9473(97)81758-6.

[102] A. Coskun, I. Kale, and Y. Eminaga. “Multiplier Free Implementation of 8-Tap Daubechies Wavelet Filters for Biomedical Applications”. In: *2017 New Generation of CAS (NGCAS)*. 2017, pp. 129–132. DOI: 10.1109/NGCAS.2017.63.

[103] Jesús Lázaro, Jagoba Arias, Armando Astarloa, et al. "Hardware architecture for a general regression neural network coprocessor". In: *Neurocomputing* 71.1 (2007), pp. 78–87. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2007.01.012. URL: https://www.sciencedirect.com/science/article/pii/S0925231207002160.

[104] Sioni Paris Summers. "Application of FPGAs to Triggering in High Energy Physics". 2018. URL: https://cds.cern.ch/record/2647951.

[105] Ian Tomalin. *Level-1 Tracking Firmware*. 2021. URL: https://twiki.cern.ch/twiki/bin/view/CMS/L1TrackFirmware.

[106] Standard C++ Foundation. *ISO/IEC. (2014). ISO International Standard ISO/IEC 14882:2014(E) – Programming Language C++. [Working draft]*. 2021. URL: https://isocpp.org/std/the-standard.

[107] Xilinx Inc. *7 Series DSP48E1 Slice User Guide*. 2018. URL: https://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf.

[108] IEEE technical professional organization. *IEEE 1076-2019 - IEEE Standard for VHDL Language Reference Manual*. 2021. URL: https://standards.ieee.org/standard/1076-2019.html.

[109] IEEE Computer Society. *IEEE Standard for Verilog Hardware Description Language*. 2021. URL: https://www.eg.bucknell.edu/~csci320/2016-fall/wp-content/uploads/2015/08/verilog-std-1364-2005.pdf.

[110] I. Antcheva, M. Ballintijn, and Bellenot. "ROOT — A C++ framework for petabyte data storage, statistical analysis and visualization". In: 180 (2015), pp. 213–304. DOI: 10.1016/j.cpc.2009.08.005. URL: http://cds.cern.ch/record/1262045.

[111] Xilinx Inc. *Vitis High-Level Synthesis User Guide*. 2021. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_2/ug1399-vitis-hls.pdf.

[112] Xilinx Inc. *Sample-Based C Code*. 2021. URL: https://www.xilinx.com/html_docs/xilinx2017_4/sdaccel_doc/qms1504034431432.html.

[113] Xilinx Inc. *Frame-Based C Code*. 2021. URL: https://www.xilinx.com/html_docs/xilinx2017_4/sdaccel_doc/iwp1504034431266.html.

[114] Particle Data Group. *2020 Review of Particle Physics*. 2021. URL: http://pdg.web.cern.ch/pdg/2020/listings/contents_listings.html.

[115] S. Agostinelli, J. Allison, and K. "Geant4—a simulation toolkit". In: 506 (2003), pp. 250–303. DOI: https://doi.org/10.1016/S0168-9002(03)01368-8. URL: https://www.sciencedirect.com/science/article/pii/S0168900203013688.

[116] M. Jeitler. "The Phase-2 Upgrade of the Hardware Trigger of CMS at the LHC". In: (2020). DOI: 10.1088/1748-0221/15/09/c09009. URL: https://doi.org/10.1088/1748-0221/15/09/c09009.

[117] Iles G. *Decisions down to the wire*. 2021. URL: https://cerncourier.com/a/cms-gears-up-for-the-lhc-data-deluge/.

[118] J Chaves. "Implementation of FPGA-based level-1 tracking at CMS for the HL-LHC". In: 9.p10-11 (2014). DOI: 10.1088/1748-0221/9/10/c10038. URL: https://doi.org/10.1088/1748-0221/9/10/c10038.

[119] Samtec Inc. *25/28 GBPS FIREFLYTM FMC+ DEVELOPMENT KIT*. 2021. URL: https://www.samtec.com/kits/optics-fpga/25g-28g-firefly-fmcp.

[120] Andrew et al. *User guide for the Master Processor, Virtex-7, Kintex-7*. 2021. URL: http://www.hep.ph.ic.ac.uk/mp7/documents/UserGuideForTheMP7.latest.pdf.

[121] Imperial College London. *"The Imperial Master Processor, Virtex-7 (MP7)"*. 2021. URL: http://www.hep.ph.ic.ac.uk/mp7/.

[122] Baber et al. "Development and testing of an upgrade to the CMS level-1 calorimeter trigger". In: *Journal of Instrumentation* 9 (Dec. 2013), pp. 18–19. DOI: 10.1088/1748-0221/9/01/C01006.

[123] Albert et al. *The Apollo ATCA Design for the CMS Track Finder and the Pixel Readout at the HL-LHC*. Tech. rep. Geneva: CERN, Oct. 2021. URL: https://cds.cern.ch/record/2797673.

[124] Hazen et al. "The APOLLO ATCA Platform". In: Mar. 2020, pp. 120–121. DOI: 10.22323/1.370.0120.

[125] Andrew Rose. *Serenity-An ATCA prototyping platform for CMS Phase-2*. Tech. rep. CERN, 2018. DOI: 10.22323/1.343.0115. URL: https://cds.cern.ch/record/2646388.

[126] Andrew Rose. *What is Serenity*. 2018. URL: https://indico.cern.ch/event/875862/attachments/1985032/3307290/Seminarenity.pdf.

[127] Thomas Stephen Williams. *IPbus A flexible Ethernet-based control system for xTCA hardware*. Tech. rep. CERN, 2014. URL: https://cds.cern.ch/record/2020872.

[128] Intel Corporation. *ModelSim-Intel FPGA Edition Software*. 2021. URL: https://www.intel.co.uk/content/www/uk/en/software/programmable/quartus-prime/model-sim.html.

[129] C. Ghabrous Larrea, K. Harder, D. Newbold, et al. "IPbus: a flexible Ethernet-based control system for xTCA hardware". In: 10 (2015), pp. 2–16. DOI: 10.1088/1748-0221/10/02/c02019. URL: https://doi.org/10.1088/1748-0221/10/02/c02019.

[130] Xilinx Inc. *Quality by Design*. 2021. URL: https://www.xilinx.com/support/quality/quality-design.html.

[131] Xilinx Inc. *AXI Reference Guide*. 2011. URL: https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf.

[132] ARM LTD. *AXI protocol overview*. 2020. URL: https://developer.arm.com/documentation/102202/0200/AXI-protocol-overview.

[133] Xilinx Inc. *AXI Reference Guide*. 2011. URL: https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf.

[134] Xilinx Inc. *UltraScale Architecture-Based FPGAs Memory IP v1.4*. 2021. URL: https://www.xilinx.com/support/documentation/ip_documentation/ultrascale_memory_ip/v1_4/pg150-ultrascale-memory-ip.pdf.

[135] Xilinx Software Inc. *Getting Started with Xilinx SDK*. 2021. URL: https://www.xilinx.com/html_docs/xilinx2015_1/SDK_Doc/index.html.

[136] Xilinx Inc. *MicroBlaze Processor Reference Guide*. 2018. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_2/ug984-vivado-microblaze-ref.pdf.

[137] Xilinx Inc. *Designing IP Subsystems Using IP Integrator*. 2021. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug994-vivado-ip-subsystems.pdf.

[138] Xilinx Inc. *Power Analysis and Optimization*. July 2012. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug907-vivado-power-analysis-optimization.pdf.

[139] G ILES, J Jones, and A Rose. "Experience powering Xilinx Virtex-7 FPGAs". In: 8 (2013). DOI: 10.1088/1748-0221/8/12/c12037. URL: https://doi.org/10.1088/1748-0221/8/12/c12037.

[140] Xilinx Inc. *Floorplanning Methodology Guide*. Ultra-Scale Chipset. Apr. 2013. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/Floorplanning_Methodology_Guide.pdf.

[141] Xilinx Inc. *Using the Vivado IDE*. 2018. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug893-vivado-ide.pdf.

[142] Xilinx Inc. *Xilinx Power Estimator User Guide*. 2021. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug907-vivado-power-analysis-optimization.pdf.

Appendices

A1 Arithmetic and Bit Growth

In the Xilinx 7-Series DSP48E1 Slice User Guide [107], the allowed word length is given as 18-bit for A and 25-bit for B in a (A x B) operations using one DSP48E1 slice. A study of word length growth using the Xilinx 7 Series DSP48E1 arithmetic block the HLS is displayed in Table, A1. The word length in fixed-point implementation accommodates multiplication and addition within the specified ranges. The maximum word length allowance in the multiplications, additions and accumulations using only one DSP48E1 block is set to 26-bits, however, the number of DSP slices increases almost exponentially if the word length exceeds the maximum value, as can be seen in the table. If the maximum word length has been reached, the synthesis tool automatically increases the number of DSP blocks to produce an accurate result. This will negatively impact the hardware implementation and resource usage if a suitable strategy is not put in place.

operation	min range	max range	DSP(s)
A x B = C	1	26	1
A x B = C	27	28	2
A x B = C	29	39	4
A x B = C	40	44	5
A x B = C	45	56	9
A x B = C	57	67	10
A x B = C	68	68+	11+
C = C + D	1	48	1
C = C + D	49	49+	overflow

Table A1: Bit growth in arithmetic operations Xilinx 7-Series.

A2 Finite Wordlength Division

In Kalman filter arithmetic, matrix inversion is one of the operations that contribute to resource usage in the hardware implementation. As shown in Equation A2, there are two multiplications, where, the word length will grow at runtime, resulting in an overflow and incorrect arithmetic results.

$$x^{-1} = \begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix}^{-1} = \frac{1}{\mathbf{d}} \begin{bmatrix} x_{11} & -x_{01} \\ -x_{10} & x_{00} \end{bmatrix}, \; where, \; \mathbf{d} = (x_{00}x_{11} - x_{01}x_{10}) \tag{A2}$$

The word length is redefined as the summation of powers of two in this scenario. The summation term is divided into parts of Most Significant Bit (MSB) and Least Significant Bit (LSB). The simplification has allowed MSB to use an 11-bit word length implementation with a 36 kb block ROM as a lookup table with maximum precision when the MSB is one. If the MSB is not one, then a bitwise shift operation is performed to push the first '1' occurrence to the MSB position. The values generated from the subtraction of x_{MSB} and x_{LSB} bits are multiplied by the value representing $1/x_{MSB}$ stored in the LUT, using only one DSP48E block. To analyse the quantisation errors, a utility library is designed to compare double-precision floating-point and fixed-point variables to provide estimated word length at runtime. The utility also calculates the precision loss and flags overflows in all multiplication and subtraction operations. In hardware implementation. the structure allows access to intermediate results for the deployment of a rounder and limiters as shown in Figure A2.

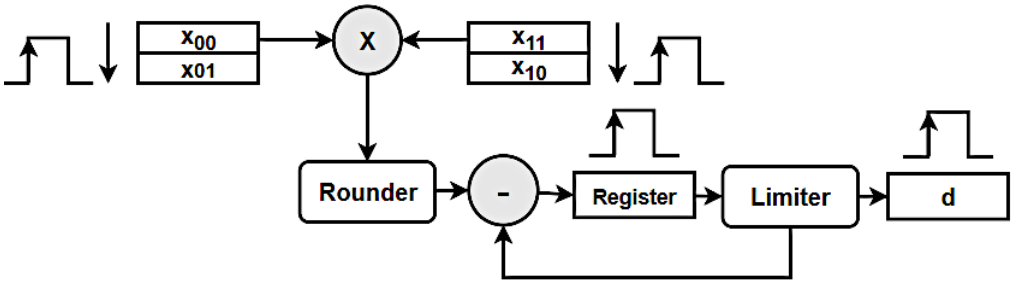

Figure A2: Inverse matrix custom hardware architecture.

A3 Time and Regional Multiplexing

Track-Trigger system with the minimum logic resource usage, the tracker is divided into regions of $1/2\pi/9$, each segmented into 48 (24η, 2ϕ) stub sectors with a time-multiplexing factor of 16. A TPF processes 16 stubs from 12 stub sectors at 40 MHz. Figure A3, shows a nonant $1/2\pi/9$ containing 48 stub sectors (columns) with the linked TPF blocks. HL-LHC creates an event at 40 MHz or 25 ns intervals. It takes 400 ns (16 x 25 ns) for all 48 column buffers to be filled, one clock phase apart. Four TFP blocks process stubs at 480 MHz with 25 ns intervals (1/480MHz x 12) at the first stage. In the second stage, stubs that survived from the first stage are processed by the next TFP blocks within 25 ns. In the third stage, the process continues with the remaining stubs from the second stage being processed in 25 ns. If a final track for the entire tracker is required, nine candidate tracks from nine nonants are fed into one TFP block to generate one track. The latency of the system is 475 (400 + 25 x 3) ns for a nonant or 500 ns for the entire tracker, whereas, in emulation, the total latency is dictated by the longest critical path from the first stage to the final stage estimated to 1,333.03 ns.

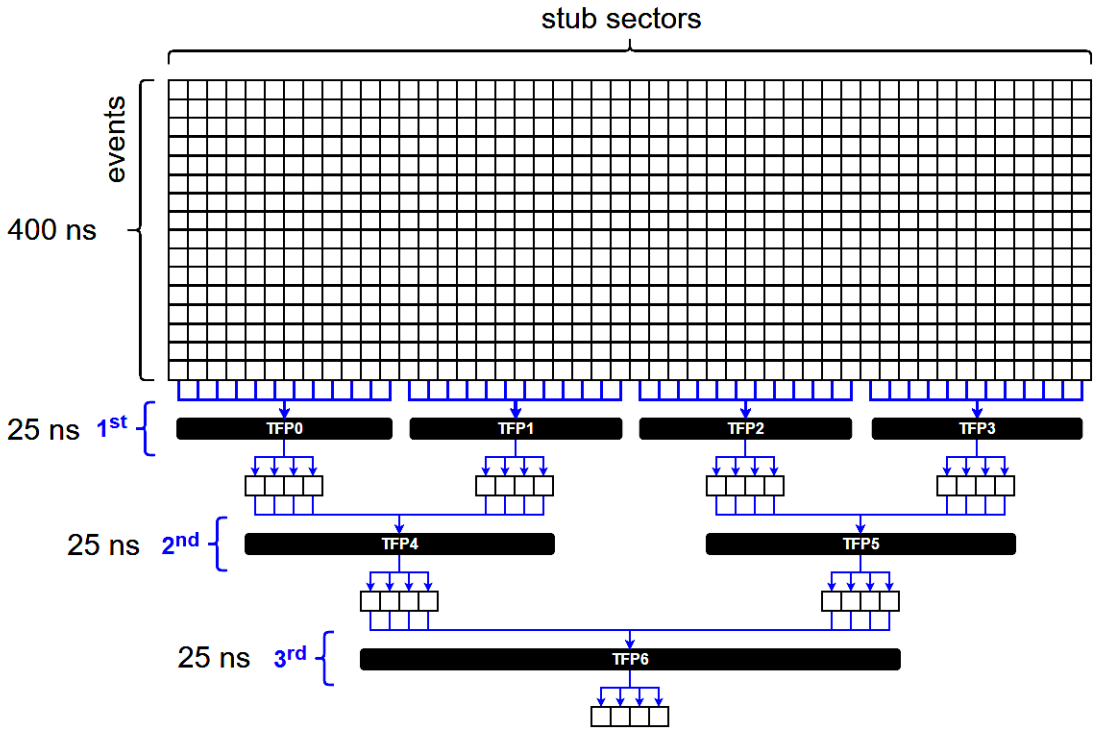

Figure A3: A nonant in Time-Multiplexed Linear Fitter.

A4 DAQ Sub-Systems FED Architecture

Figure A4 shows the central Triggering and Data Acquisition system with all sub-systems. The DAQ monitors and controls all the sub-detectors interlinked through readout electronics via newly upgraded μ-TCA, full-size, double-width Advanced Mezzanine Cards, holding the Xilinx Kintex-7 FPGA devices and optical SLINK-express 10 Gb/s transmitters. Triggering in the CMS DAQ operates at two levels: the Level-1 and the High-Level trigger systems, which are very different in how they filter the data. The High-Level Trigger is a 13,000 CPU core farm with a ~500 Hz accept rate. The Level-1 Trigger uses FPGA custom-built devices with an acceptable rate up to 750 kHz. The Level-1 Trigger is being upgraded with a Track-Trigger system. For Tracker, this means using p_T modules as the filtering mechanism for data reduction. As is shown in Figure A4, the Tracker occupy a significant portion of the CMS DAQ system with an event size of up to 1 MB. Each rectangular green box is a readout device connected to the TEC-, TIB, TOB, and TEC+. The figure is a screenshot from Level-1 DAQ online system during cruzet-2021 data-taking period.

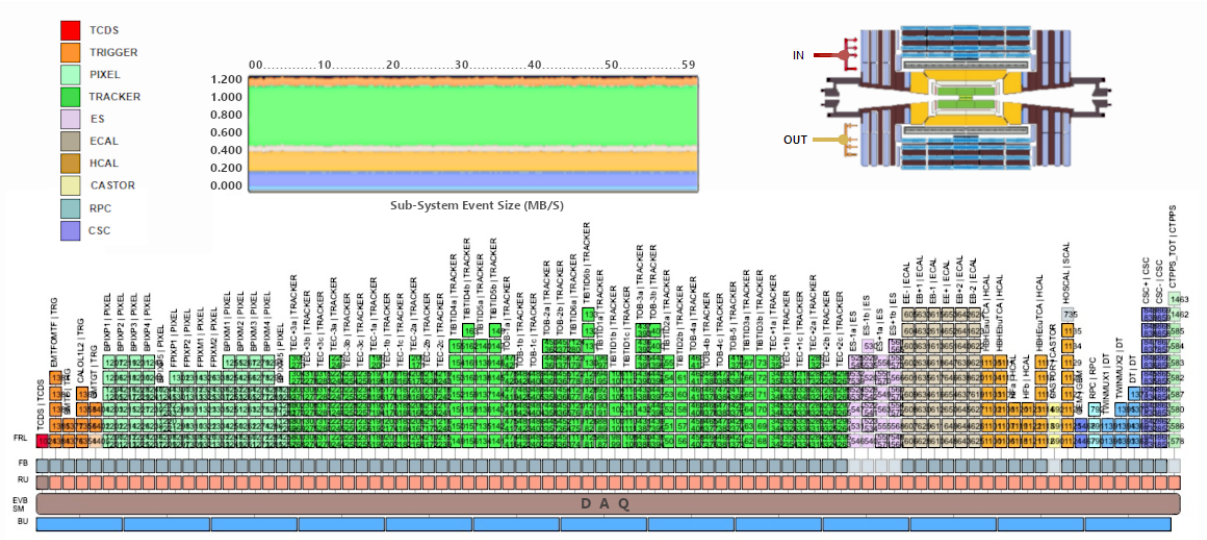

Figure A4: Central Triggering and Data Acquisition system.

A5 Driving the Linear Fit Algebraically

The gradient (m) and intercept (b) are parameters in a linear fit ($mx_n + b$) with n data points, and the sum of Squared Errors $SE_{fit} = (y_n - (mx_n + b))^2$ are the distances from the data points to the fit in y, given:

$$
\begin{aligned}
SE_{fit} &= \sum_{i=1}^{n} (y_i - (mx_i + b))^2 \\
&= (y_1 - (mx_1 + b))^2 + (y_2 - (mx_2 + b))^2 + \ldots + (y_n - (mx_n + b))^2 \\
&= (y_1^2 - 2y_1(mx_1 + b) + (mx_1 + b)^2) + \\
&\quad (y_2^2 - 2y_1(mx_2 + b) + (mx_2 + b)^2) + \ldots \\
&\quad y_n^2 - 2y_1(mx_n + b) + (mx_n + b)^2 \\
&= y_1^2 - 2y_1 m x_1 - 2y_1 b + m^2 x_1^2 + 2mx_1 b + b^2 + \\
&\quad y_2^2 - 2y_2 m x_2 - 2y_2 b + m^2 x_2^2 + 2mx_2 b + b^2 + \ldots \\
&\quad y_n^2 - 2y_n m x_n - 2y_n b + m^2 x_n^2 + 2mx_n b + b^2 \\
&= (y_1^2 + y_2^2 + \ldots + y_n^2) - \\
&\quad 2m(x_1 y_1 + x_2 y_2 + \ldots + x_n y_n) - \\
&\quad 2b(y_1 + y_2 + \ldots + y_n) - \\
&\quad m^2(x_1^1 + x_2^2 + \ldots + x_n^2) + \\
&\quad 2mb(x_1 + x_2 + \ldots + x_n) + nb^2 \\
&= n\overline{y^2} - 2mn\overline{xy} - 2bn\overline{y} + m^2 n\overline{x^2} + 2mbn\overline{x} + nb^2
\end{aligned}
$$

$$
\left(\frac{\delta SE_{fit}}{\delta m} = 0\right) \; + \; \left(\frac{\delta SE_{fit}}{\delta b} = 0\right)
$$

$$
\frac{\delta SE_{fit}}{\delta m} = -2n\overline{xy} + 2n\overline{x^2}m + 2bn\overline{x} = 0 \;\rightarrow\; -\overline{xy} + m\overline{x^2} + b\overline{x} = 0
$$

$$
\rightarrow\; m\overline{x^2} + b\overline{x} = \overline{xy} \;\rightarrow\; m\frac{\overline{x^2}}{\overline{x}} + b = \frac{\overline{xy}}{\overline{x}} \;\rightarrow\; (m\overline{x^2} + b\overline{x} = \overline{xy}) - (m\frac{\overline{x^2}}{\overline{x}} + b = \frac{\overline{xy}}{\overline{x}})
$$

$$
\rightarrow\; m\left(\overline{x} - \frac{\overline{x^2}}{\overline{x}}\right) = \overline{y} - \frac{\overline{xy}}{\overline{x}} \;\rightarrow\; m = \frac{\overline{y} - \frac{\overline{xy}}{\overline{x}}}{\overline{x} - \frac{\overline{x^2}}{\overline{x}}} \;\rightarrow\; m = \frac{\bar{x}\bar{y} - \overline{xy}}{\overline{x}^2 - \overline{x^2}}
$$

$$
\rightarrow\; m = \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - \overline{x}^2}
$$

$$
\frac{\delta SE_{fit}}{\delta b} = -2n\overline{y} + 2mn\overline{x} + 2bn = 0 \;\rightarrow\; -\overline{y} + m\overline{x} + b = 0 \;\rightarrow\; m\overline{x} + b = \overline{y}
$$

$$
\rightarrow\; b = \overline{y} - m\overline{x}
$$

$$
\therefore\; y = mx + b \;\rightarrow\; y = \left(\frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - \overline{x}^2}\right)x + (\overline{y} - m\overline{x})
$$

A6 High Luminosity LHC Plan

Figure A6 shows a ten years timescale encompassing the Long Shutdown 1 (2013), Long Shutdown 2 (2019) and Long Shutdown 3 (2025). The LHC/HL-LHC plan constitutes Tracker, DAQ, Level-1 Trigger development and Technical Design Report (TDR) documentation in high luminosity upgrade. This plan has been obtained in 2018 and updated regularly by the CMS Collaboration.

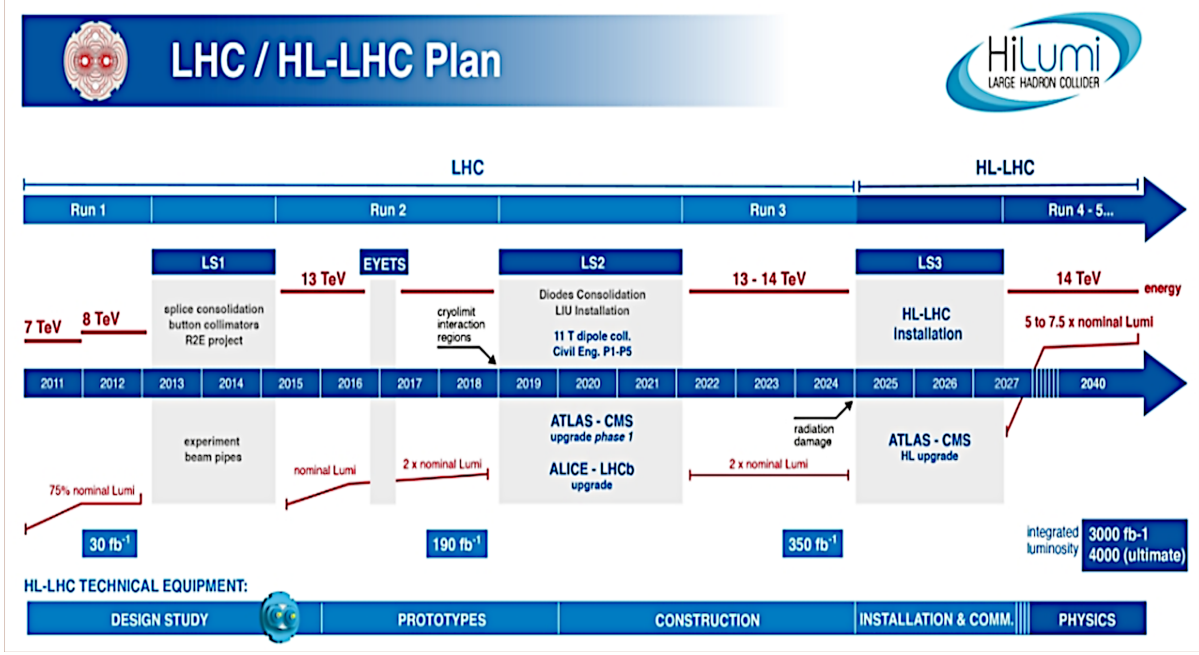

Figure A6: Design, development and prototype timescale [22].

A7 Project Milestone

The current HL-LHC upgrade production plan for the Level-1 Track-Trigger modules for the CMS detector is shown in Figure A7. The information concerning the Kalman Filter, Linear Fitter and the Demonstrator is added for the period 2018 to 2021. The Level-1 Trigger project was in the pre-production stage in the year 2021. The milestone was last updated at the end of 2021.

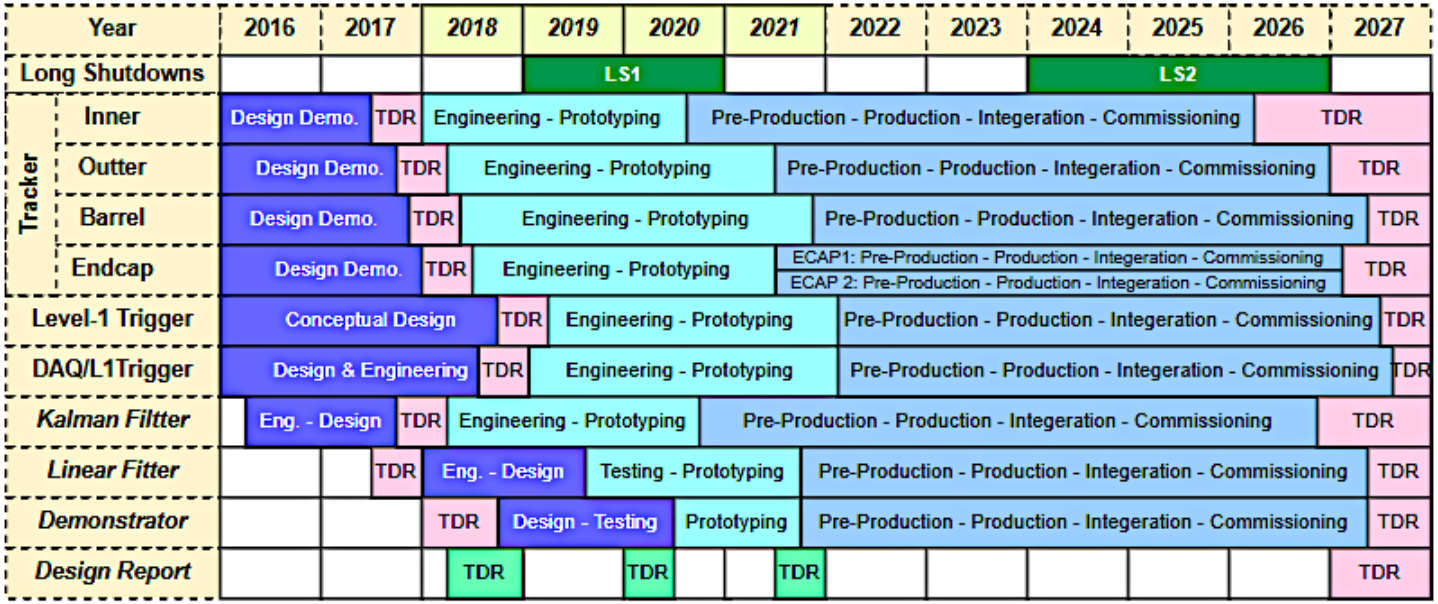

Figure A7: Level-1 Track-Trigger scheduling and timing.

This page was intentionally left blank.

Acronyms

A

ALICE **A** **L**arge **I**on **C**ollider **E**xperiment
ASIC **A**pplication **S**pecific **I**ntegrated **C**ircuit
ATCA **A**dvanced **T**elecommunications **C**omputing **A**rchitecture
AMC **A**dvanced **M**ezzanine **C**ard
ALU **A**rithmetic **L**ogic **U**nit
AOD **A**nalysis **O**bjects **D**Data
AM **A**ssociative **M**emory
AXI **A**dvanced **eX**tensible **I**nterface

B

BX **B**unch **C**rossing in **A**ccelerator **E**xperiment
BE **B**ack-**E**nd **E**lectronics
BRAM **B**lock **R**andom **A**ccess **M**emory

C

CMS **C**ompact **M**uon **S**olenoid
CMSSW **CMS** **S**oftware **F**ramework
CMOS **C**omplementary **M**etal **O**xide **S**emiconductor
CC **C**harge **C**ollection in **S**ensors
CIC **C**oncentrator **I**ntegrated **C**ircuit
CBC **CMS** **B**inary **C**hip
CI **C**ontinues **I**ntegration
CD **C**ontinues **D**evelopment

CPU **C**entral **P**rocessing **U**nit
CKF **C**ombinatorial **K**alman **F**ilter
CT **C**orrelator **T**rigger

D

DAQ **D**ata **A**cquisition
DTC **D**ata **T**rigger and **C**ontrol
DR **D**uplicate **R**emoval
DAS **D**ata **A**ggregation **S**ervices
DQM **D**ata **Q**uality **M**onitoring
DAQ **T**riggering **D**ata **A**cquisition
DSP **D**igital **S**ignal **P**rocessing
DOF **D**egree of **F**reedom
DDR **D**ouble **D**ata **R**ate **M**emory

E

ECAL **E**lectromagnetic **C**alorimeter
ES **E**lectromagnetic **P**reshower
EP **E**xperimental **P**hysics
EDA **E**lectronic **D**esign **A**utomation
EMP **C**ommon **I**nfrastructural **F**irmware **F**ramework
EDM **E**vent **D**ata **M**odel
EDA **E**lectronic **D**esign **A**utomation

F

FinFET **F**in **F**ield-effect CMOS **T**ransistor
FPGA **F**ield **P**rogrammable **G**ate **A**rray
FE **F**ront-**E**nd **E**lectronics
FED **F**ront-**E**nd **D**evices
FEC **F**ront-**E**nd **C**ontrollers
FF **F**lip **F**lope
FIFO **F**irst in **F**irst **O**ut
FET **F**ield-**E**ffect **T**ransistor
FL **F**ractional **L**ength

G

GPU **G**raphical **P**rocessing **U**nit
GP **G**eometric **P**rocessor
GUI **G**rphical **U**ser **I**nterface
GLD **C**onfigurable **L**ogic **B**locks
GBDT **G**radient **B**oosted **D**ecision **T**ree

H

HL-LHC **H**igh **L**uminosity **L**arge **H**adron **C**ollider
HL **H**igh **L**uminosity
HCAL **H**adron **C**alorimeter
HEP **H**igh **E**nergy **P**hysics
HLT **H**igh **L**evel **T**rigger
HDL **H**ardware **D**escription **L**anguage
HLS **H**igh **L**evel **S**ynthesis
HT **H**ough **T**ransform
HTTP **H**ypertext **T**ransfer **P**rotocol

I

IP **I**ntellectual **P**roperty
IPBB **IP** **B**us **B**uilder
II **I**nitiation **I**nterval

K

KF **K**alman **F**ilter
KG **K**alman **G**ain

L

LHC **L**arge **H**adron **C**ollider
LEP **L**arge **E**lectron-**P**ositron
LpGBT **L**ow-Power **G**iga-Bit **T**ransceiver
LF **L**inear **F**itter
LR **L**inear **R**egression
LUT **L**ook**up** **T**able

LSB **L**east **S**ignificant **B**it

M

MSGC **M**icro**S**trip **G**as **C**hambers
MVLF **M**ulti**v**ariate **L**inear **F**itter
MC **M**onte **C**arlo **S**imulations
ML **M**achine **L**earning
MAC **M**ultiply **A**ccumulate
MSB **M**ost **S**ignificant **B**it
MP7 **G**eneral **P**urpose **P**rocessor 7
MCH **M**icro-TCA **C**arrier **H**ub

O

OOP **O**bject **O**riented **P**rogramming
OOM **O**bject **O**riented **M**odelling

P

PS **P**ixel **S**trip **M**odule in **T**racker
PAT **P**hysics **A**nalysis **T**oolkit
PU **E**vent **P**ile**u**p
PP **P**roton **P**roton
PLD **P**rogrammable **L**ogic **D**evices
PL **P**roposed **L**ength
PCI **P**eripheral **C**omponent **I**nterconnect

R

R&D **R**ecearch and **D**evelopment
RTL **R**egister **T**ransfer **L**evel
RAW **R**aw **D**ata **P**roduction
RECO **R**econstructed **T**rackes
RAM **R**andom **A**ccess **M**emory
R^2 **G**oodness of **F**it
RAL **R**utherford **A**ppleton **L**aboratory

S

SM **S**tandard **M**odel
SW **S**oftware **D**evelopment
SB **S**ynchrotron **B**ooster
SoC **S**ystem on **C**hip
SS **S**um-of-**S**quared
SST **T**otal **S**um-of-**S**quares
SSE **S**um-of-**S**quare **E**rrors
SST **S**um-of-**S**quares **R**egression
SNR **N**oise to **S**ignal **R**atio
STL **S**tandard **T**emplate **L**ibrary

T

TDR **T**echnical **D**esign **R**eport
TB **T**racker **B**arrel
TE **T**racker **E**ndcap
TIB **I**nner **B**arrel **R**egions
TOB **O**uter **B**arrel **R**egions
TIB **I**nner **B**arrel **R**egions
TID **I**nner **B**arrel **E**ndcap **R**egions
TEC **T**racker **E**ndcap **R**egions
TRG **L**evel-1 **T**rigger **M**echanism
TCA **T**elecommunication **C**omputing **A**rchitecture
TT **T**rack **T**rigger
TM **T**ime **M**ultiplexing
TFP **T**rack **F**inder **P**rocessor
TMTT **T**ime-**M**ultiplex **T**rack-**T**rigger
TL **T**raclet **S**ystem
TMLF **T**ime-**M**ultiplexed **L**inear **F**itter
TXRX **T**ransmit and **R**eceive
TP **T**racking **P**article
TIF **T**racker **I**ntegration **F**acility

U

UUT **U**nit **U**nder **T**est
UDP **U**ser **D**atagram **P**rotocol

V

VHSIC **V**ery **H**igh-**S**peed **I**ntegrated **C**ircuit

W

WLCG **W**orldwide **LHC** **C**omputing **G**rid

X

XML **E**xtendable **M**arkup **L**anguage

XSDK **X**ilinx **S**oftware **D**evelopment **K**it

This page was intentionally left blank.