Inaugural-Dissertation

zur

Erlangung der Doktorwürde

der

Naturwissenschaftlich-Mathematischen Gesamtfakultät

der

Ruprecht-Karls-Universität
Heidelberg

vorgelegt von	Diplom-Physiker
	Jan de Cuveland
	aus Hamburg
Tag der mündlichen Prüfung	17. September 2009

A Track Reconstructing Low-latency Trigger Processor for High-energy Physics

Jan de Cuveland

Gutachter: Prof. Dr. Volker Lindenstruth

Prof. Dr. Ulrich Brüning

A track reconstructing low-latency trigger processor for high-energy physics

The detection and analysis of the large number of particles emerging from high-energy collisions between atomic nuclei is a major challenge in experimental heavy-ion physics. Efficient trigger systems help to focus the analysis on relevant events. A primary objective of the Transition Radiation Detector of the ALICE experiment at the LHC is to trigger on high-momentum electrons. In this thesis, a trigger processor is presented that employs massive parallelism to perform the required online event reconstruction within 2 µs to contribute to the Level-1 trigger decision. Its three-stage hierarchical architecture comprises 109 nodes based on FPGA technology. Ninety processing nodes receive data from the detector front-end at an aggregate net bandwidth of 2.16 Tbit/s via 1080 optical links. Using specifically developed components and interconnections, the system combines high bandwidth with minimum latency. The employed tracking algorithm three-dimensionally reassembles the track segments found in the detector's drift chambers based on explicit value comparisons, calculates the momentum of the originating particles from the course of the reconstructed tracks, and finally leads to a trigger decision. The architecture is capable of processing up to 20 000 track segments in less than 2 µs with high detection efficiency and reconstruction precision for high-momentum particles. As a result, this thesis shows how a trigger processor performing complex online track reconstruction within tight real-time requirements can be realized. The presented hardware has been built and is in continuous data taking operation in the ALICE experiment.

Ein Triggerprozessor zur Online-Spurrekonstruktion in der Hochenergiephysik

Eine große Herausforderung der experimentellen Schwerionenphysik ist der Nachweis und die Auswertung der großen Anzahl an Teilchen, die bei hochenergetischen Stößen zwischen Atomkernen entstehen. Leistungsfähige Triggersysteme helfen dabei, die Auswertung auf relevante Ereignisse zu konzentrieren. Ein Hauptziel des Übergangsstrahlungsdetektors im ALICE-Experiments am LHC ist es, als Trigger auf Elektronen mit hohem Transversalimpuls zu reagieren. In dieser Arbeit wird ein Triggerprozessor vorgestellt, der die notwendige Online-Spurrekonstruktion durch massives Parallelisieren innerhalb von 2 µs durchführt, um zur Level-1-Triggerentscheidung beizutragen. Seine dreistufige hierarchische Architektur umfasst 109 auf FPGA-Technik basierender Knoten. Über 1080 optische Verbindungen empfangen 90 Verarbeitungsknoten die Daten aus dem Detektor-Front-End mit einer gesamten Nettodatenrate von 2,16 Tbit/s. Mit Hilfe speziell entwickelter Baugruppen und Verbindungen kombiniert das System hohe Bandbreite mit minimaler Latenz. Der eingesetzte Trackingalgorithmus setzt die in den Driftkammern des Detektors nachgewiesenen Teilchenspursegmente aufgrund expliziter Wertvergleiche dreidimensional zusammen, ermittelt aus dem Verlauf der so rekonstruierten Spuren den Impuls der erzeugenden Teilchen und führt schließlich zur Triggerentscheidung. Die Architektur kann in weniger als 2 µs bis zu 20 000 Spursegmente verarbeiten und erreicht eine hohe Nachweiseffizienz und Rekonstruktionsgenauigkeit für Teilchen mit hohem Transversalimpuls. Die Ergebnisse dieser Arbeit zeigen, wie ein Triggerprozessor, der aufwändige Online-Spurrekonstruktion innerhalb enger Echtzeitanforderungen ausführt, verwirklicht werden kann. Die dargestellte Hardware wurde real aufgebaut und befindet sich im Einsatz im ALICE-Experiment.

Contents

List of Figures

List of Tables

1 Introduction

The investigation of the fundamental structure of matter has been a subject of physical research for a long time. Continually improving experimental possibilities have allowed to explore the elementary structures at ever smaller scales. Over the past decades, high-energy physics has been very successful. A variety of experiments have confirmed a theory according to which matter is built from two classes of elementary particles, *quarks* and *leptons*.

The forces between these particles can be attributed to four fundamental interactions. They act through exchange particles, which are characteristic of the respective interaction. Leptons interact via the electromagnetic force using photons as exchange particles and also via the *weak interaction*, which is mediated by W and Z bosons. The *electroweak theory* describes the forces between leptons with high precision. Quarks are additionally affected by the *strong interaction* mediated by gluons. The gravitational force with its not yet experimentally proven exchange particles, the gravitons, affects all particles, but it is negligible in this context because of its comparatively low strength. The 12 known elementary particles are listed in Table 1.1. Table 1.2 summarizes the fundamental interactions with their associated exchange particles and the typical ranges and relative strengths.

The strong interaction is described by *quantum chromodynamics (QCD)*, the theory of color forces. In QCD, quarks are assigned an additional property beside their electrical charge, the color charge. It can have the values red, green, blue, and the appropriate anti-colors. The gluons themselves also possess a color charge, enabling them to interact with each other. Unlike the other forces, the strength of the interaction between quarks increases with distance. As a consequence of this characteristic, only color-neutral particles are found in nature. According to the *standard model*, these composite particles are either *baryons* formed from three quarks of different colors, or *mesons* consisting of a quark and

	Quarks	Q/e	Leptons	Q/e
First generation	d (down)	$-1/3$	electron neutrino ν_e	0
	u (up)	$+2/3$	electron e	-1
Second generation	s (strange)	$-1/3$	muon neutrino ν_μ	0
	c (charm)	$+2/3$	muon μ	-1
Third generation	b (bottom)	$-1/3$	tau neutrino ν_τ	0
	t (top)	$+2/3$	tau τ	-1

Table 1.1: The 12 fundamental particles. They are divided into three generations according to ascending mass. These particles and their anti-particles constitute all known matter.

Interaction	Couples to	Exchange particle	Strength (relative)	Range (m)
strong	color charge	gluon (g)	1	$\approx 10^{-15}$
electromagnetic	el. charge	photon (γ)	10^{-2}	∞
weak	weak charge	$W^{\pm}$, Z^0 bosons	10^{-14}	$\approx 2 \times 10^{-18}$
gravitational	mass	graviton?	10^{-38}	∞

Table 1.2: The four fundamental interactions, which fully describe the known world of today [14].

its anti-quark, in which case color and anti-color neutralize themselves. The *confinement*, a characteristic of the strong interaction, inhibits the isolation of individual free quarks.

Since the confinement prevents the dissection of hadrons[1] into quarks, it is advisable to study the characteristics of the strong interaction in a system of many particles that can be described by means of thermodynamics. Approximate QCD calculations for such thermodynamic systems predict that the confinement is eliminated at extremely high temperatures or densities. This state, in which quarks and gluons can move as quasi-free particles, is called *quark-gluon plasma (QGP)*. The theoretically predicted transition causes a dramatic increase in the degrees of freedom, suggesting a phase transition between confinement and deconfinement [48]. The study of this phase transition, as well as the new state of matter, the quark-gluon plasma, plays an important role in understanding strongly interacting matter.

The different phases of strongly interacting matter are summarized in Figure 1.1. Cold atomic nuclei are depicted in the phase diagram at normal nuclear density ρ_0 and temperature $\mathrm{k}T \approx 0$. If such a nucleus is compressed approximately tenfold, the individual nucleons overlap, so that quarks and gluons can move quasi-freely in the nuclear volume. This state of cold plasma may exist in very dense neutron stars [59]. If, on the other hand, the temperature is increased at low baryon density, QGP also arises, starting at a temperature of approximately $\mathrm{k}T_\mathrm{c} \approx 200\,\mathrm{MeV}$. This phase transition is of special significance for cosmology, since according to Big Bang models in an early phase of the universe the hot matter cooled down, undergoing the phase transition from quark-gluon plasma to hadronic gas. If this phase transition is of first order, i. e., both phases coexist, significant inhomogeneities in the baryon density may arise. This *inhomogeneous nucleosynthesis* is a possible explanation for certain aspects of the abundances of elements in the universe [48]. Therefore, experimental investigation of the phase transition can also contribute to better understanding the emergence of elements in Big Bang models.

Ultra-relativistic heavy ion collisions are used in an attempt to create the state of quark-gluon plasma experimentally. To enable investigation of the phenomenon, two heavy atomic nuclei are collided at high energies, producing an area of extreme temperature and density. Evidence suggests that QGP has already been observed in earlier experiments. The ALICE experiment aims to unambiguously prove its existence and allow for detailed examination of the associated phenomena at substantially higher energies.

[1]Baryons and mesons

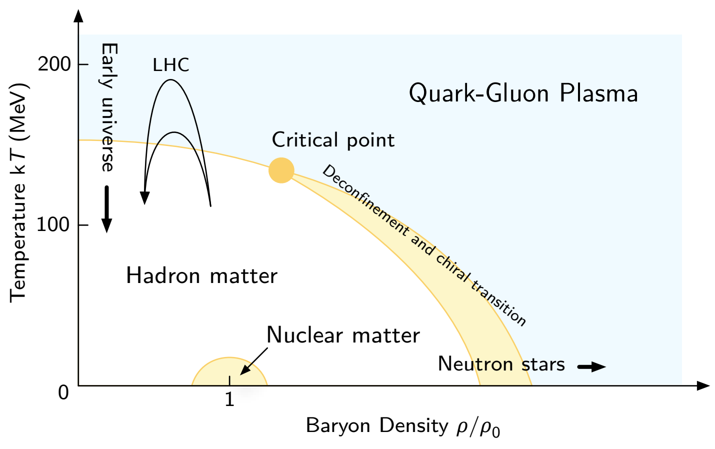


Figure 1.1: A schematic phase diagram of strongly interacting matter. Current investigations of the phase diagram suggest a strong dependence of the properties of the transition on the baryon density. At the LHC, the transition is investigated in the high-temperature range [1, 83].

High-Energy Physics Particle Detection – A Computer Science Problem?

Ultra-relativistic heavy ion reactions are a major experimental challenge. While the experimental techniques are similar to those of particle physics, heavy-ion collisions result in a substantially larger number of generated particles of several thousand per collision. This places significantly higher demands on the resolution of the particle detectors required to analyze the collision events. These detectors have continually evolved over the past decades. Gas-based particle tracking detectors, for example, are read out electronically since the invention of the multi-wire proportional chamber at CERN in 1968. Since then, the improvements of particle detectors have been closely coupled to the advancements of the corresponding signal processing and readout electronics technology, which provides ever faster readout and an increasing number of channels. As modern readout electronics digitize the measured data as early as possible, the striving for further improvements in particle detectors implicates fascinating challenges in the area of computer engineering.

Modern particle detectors produce huge amounts of data within a short time. Since even with today's technology not all data can be read out and saved in real time, a preselection of physically interesting events to read out and store permanently has to be determined during run-time with the help of a trigger system. As events dropped by the trigger system cannot be recovered later, the functionality of the trigger determines the quality of the experiment results. Therefore, trigger logics are also evolving to increasingly complex systems with compelling research topics in the field of computer science and engineering.

To meet the demand for higher data rates and the increasing complexity of trigger algorithms, the features of latest microelectronics and computer technology should be exploited when building new detector and trigger systems. Thus, the design of contemporary particle detectors is affected by the general development of processors and network components, which proceeds mainly towards higher bandwidths, while latencies are not reduced equally. Accordingly, many newer experiments are pipelined as far as possible, so that latencies in analysis electronics are less critical. In many areas, existing up-to-date technology (such as computers and network components) can be employed, for example when filtering the recorded events in a PC cluster. After the data has been received in a PC, a variety of analysis algorithms can be performed using standard hardware and existing network components.

In certain crucial applications, however, the absolute latency is important for physical reasons, for instance when detectors cannot be read out continuously, and there is dead time between event and readout decision. An example of such a detector is the Time Projection Chamber (TPC) of ALICE, which cannot be read continuously, as its gating grid needs to be disabled explicitly after an event to initiate the data acquisition sequence. Here, a trigger decision has to be made with as small a latency as possible in order to retain the efficiency of the overall system and not lose valuable information.

In this case, the system requirements are completely different than in conventional network technology, and dedicated trigger processors are typically used. With the help of special concepts and the employment of application-specific hardware, it is nevertheless possible to utilize today's technology efficiently. Thus, problems of a complexity that could in former times only be handled in offline analysis, which is uncritical in terms of time, can now be solved with very low latency. While trigger processors are often systolic designs, which rely on arithmetics performable in parallel by a systolic array and often a histogramming method, today's technology also allows for more complex macro blocks, and in some cases even high-precision iterative algorithms similar to those used in offline analysis. Because of the tight low-latency requirements, these algorithms require special consideration with respect to parallel processing and taking advantage of intrinsic regularities in the input data. The trigger processor hardware, on the other hand, should also be thoroughly optimized for minimum latency.[2]

This thesis focuses on presenting and exemplifying central concepts involved when designing such a trigger processor. It addresses critical issues of overall system design, low-latency architecture, system management, and adapting a high-precision particle tracking algorithm. The Transition Radiation Detector (TRD) online trigger system of ALICE is used as an example. It is one of the most complex first-level trigger systems used in high-energy physics in the world today. With a trigger processor named *Global Tracking Unit (GTU)* as its central component, the system is capable of performing a complete real-time evaluation of the particle tracks in a transition radiation detector and thus contributing to the first-level trigger decision of the ALICE experiment.

Outline The following chapter briefly presents the LHC accelerator and the ALICE experiment. It contains an introduction to the TRD and its trigger system and explains the demands that this application imposes on a trigger processor. In the third chapter, the hardware architecture of a suitable trigger processor is presented, which is designed to

[2]See [54] for a comparative overview of contemporary trigger systems.

provide low-latency data processing and transfer. The fourth chapter briefly introduces its monitoring and control system. In Chapter 5, the benefit of fast multievent buffering is studied and its implementation in the trigger processor is summarized. Chapter 6 briefly explains the preprocessing required in the detector front-end for efficient trigger processing. A suitable online tracking strategy is developed in Chapters 7 and 8. These chapters also highlight critical aspects of the implementation. While Chapter 7 focuses on recognizing and composing the particle tracks, the eighth chapter concentrates on the reconstruction of the corresponding particle's properties, particularly the transverse momentum. A performance analysis of the design including the results of various simulations is the subject of the ninth chapter. Chapter 10 summarizes the results.

2 The Experiment

At CERN[1], the European Laboratory for Particle Physics located at the border between France and Switzerland near Geneva, physicists study the fundamental constituents of matter and the interactions between them. CERN provides accelerators, which accelerate particles nearly to the speed of light, and supports collaborations building detectors, which make the particles visible. Founded in 1954, CERN today is the world's largest research center for particle physics with 20 member states and some 8000 visiting scientists representing 85 nationalities. The University of Heidelberg is one of 580 participating universities.

2.1 The Large Hadron Collider (LHC)

At present (2008), a new particle accelerator is being commissioned, the Large Hadron Collider (LHC). Fully operational, it will allow interactions of protons at a center-of-mass energy of 14 TeV. In the heavy ion operating mode, heavy atomic nuclei (lead ions) can be accelerated, which collide at an energy of 1150 TeV.

The LHC is built in a tunnel about 100 m underground with a circumference of 27 km. Figure 2.1 shows the structure of the accelerator schematically. The particles are preaccelerated in smaller facilities like the Super Proton Synchrotron (SPS) and injected into the LHC storage ring after reaching a certain minimum energy. Here, the two beams of opposite direction are accelerated further and finally collide at four dedicated points of interaction, each situated at the center of a large experiment setup.

The magnetic fields of more than 8 T necessary to control the protons' trajectories at these energies can only be generated by superconducting magnets. The LHC is operated by more than 1000 of such superconducting magnets (each 13 m in length), which have to be cooled down to a temperature of 1.8 K ($-271.2\,^\circ$C).

The center-of-mass energy for proton collisions at the LHC will be around an order of magnitude higher than the previous maximum of 2 TeV, which is reached at the Tevatron at Fermilab (near Chicago). The luminosity[2] will even be at least 100 times higher than at all other existing experiments. In the heavy ion mode of operation, the center-of-mass

[1]The acronym represents the name of its original founding body, the French *Conseil Européen pour la Recherche Nucléaire* [22].

[2]The luminosity is a measure for the collision rate. Given two particle beams in a storage ring, the luminosity is the collision rate of the particle bunches of the two beams, multiplied by the two numbers of particles per bunch, relative to the beam cross section at the point of interaction [19].

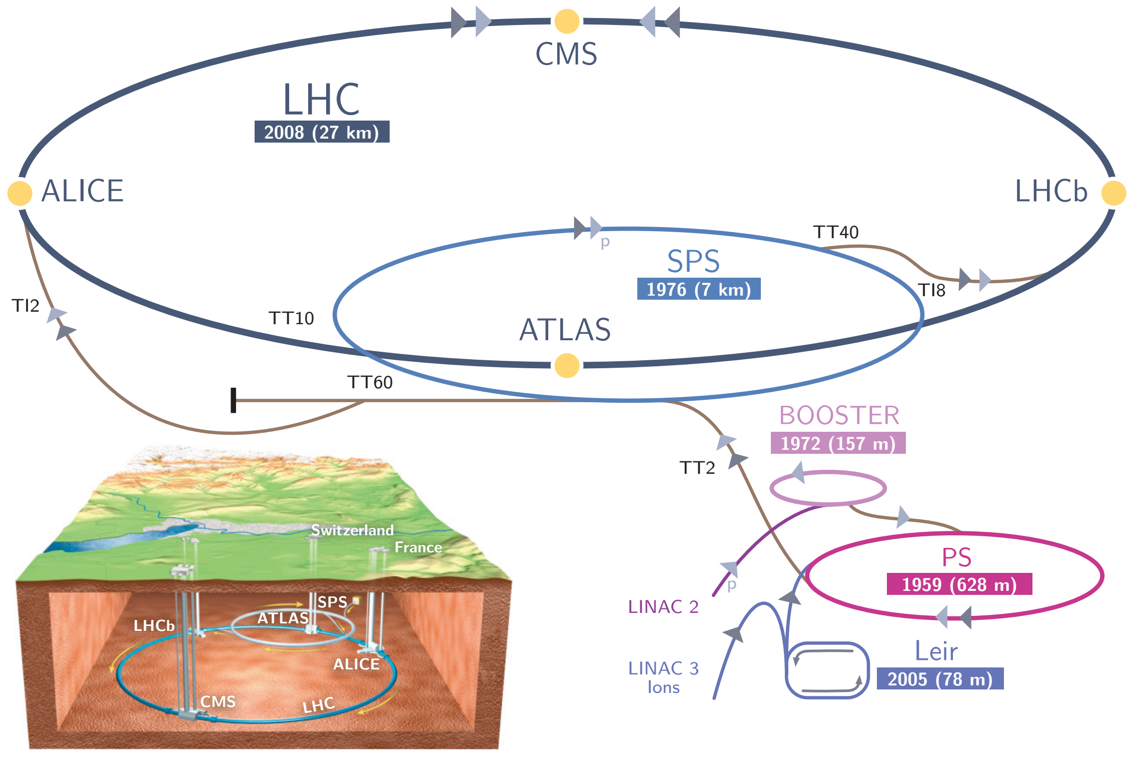


Figure 2.1: The LHC accelerator system with its main experiments and preaccelerators. The LHC ring and the experiments are located approximately 100 m underground. Image: CERN, used with permission.

energy of the Relativistic Heavy Ion Collider (RHIC) at Brookhaven National Laboratory is exceeded thirtyfold. Thanks to the high energy and luminosity, the LHC will enable physical experiments that allow even deeper insights into the structure of matter than preceding experiments. Table 2.1 summarizes some key characteristics of the LHC.

There are six experiments installed at the LHC, five of which are designed primarily for the proton-proton mode of operation. The *ATLAS*[3] experiment is designed as a general-purpose experiment. A major goal is to study the origin of mass in the range of electroweak interactions. The experiment will search for the Higgs boson and aim to determine an upper limit for its mass. Another important goal is the study of physics beyond the standard model. The *CMS*[4] is also designed as a general-purpose detector. With a strong magnetic field of 4 T, it is specifically suited for the analysis of muons. Two smaller experiments are installed near the interaction points used by ATLAS and CMS: The *TOTEM*[5] experiment intends to determine the total cross-section of proton-proton collisions, and *LHCf* will test current models of ultra high-energy cosmic rays. The *LHCb*

[3] *ATLAS*: A Toroidal LHC ApparatuS
[4] *CMS*: Compact Muon Solenoid
[5] *TOTEM*: TOTal Elastic and diffractive cross section Measurement

General	Momentum per proton at injection	450 GeV/c
	Momentum per proton at collision	7 TeV/c
	Revolution frequency	11.245 kHz
	Dipole field at 450 GeV	0.535 T
	Dipole field at 7 TeV	8.33 T
	Machine circumference	26 658.883 m
	Distance between beam axes	194 mm
Lead Ion Operation	Atomic number, lead ions	82
	Mass number, lead ions	208
	Energy per nucleon at injection	0.18 TeV/u
	Energy per nucleon at collision	2.76 TeV/u
	Center-of-mass energy at collision	1148 TeV
	Number of ions per particle bunch	7.0×10^7
	Number of particle bunches	592
	Width of beam at interaction point	15.9 µm
	Luminosity	0.5×10^{27} cm^{-2}s^{-1}
	Total cross section	437 b
	Beam current	6.16 mA

Table 2.1: Key characteristics of the LHC accelerator and the accelerated particles

experiment aims to investigate the characteristics of B-mesons[6], studying in particular CP violation at the decay of these B-mesons.

ALICE[7] is the only one of the six experiments at the LHC which is specifically designed for the heavy ion operating mode. In collisions of lead ions (^{208}Pb) at LHC energies, the formation of a new phase of matter, the quark-gluon plasma (QGP), is expected. A primary goal of ALICE is to identify and study this new state of matter [2].

2.2 The ALICE Experiment

Results of previous heavy ion experiments at CERN's SPS (in the 1980s and 90s) contain indications that quark-gluon plasma (QGP), the predicted state of matter in which quarks and gluons are deconfined and can move quasi-freely, may have already been produced. Recently, measurements at RHIC experiments have also revealed evidence for the existence of a new form of nuclear matter at extremely high density and temperature. However, detailed analyses indicate that this hot, dense medium has surprising properties, which are not yet fully understood [60].

In the ALICE experiment, for the first time, lead ions collide at a center-of-mass energy of 1150 TeV — about a factor 300 higher than at past experiments at CERN. These high

[6]A B-meson is a bound state of a b-quark and a lighter quark, e. g., ($\bar{\text{d}}$b).

[7]*ALICE*: A Large Ion Collider Experiment

energies will allow for a detailed study of the new state of matter and the predictions of quantum chromodynamics (QCD). The nuclear collisions result in final states of unprecedented complexity with several thousand particles produced at each collision. To allow conclusions on existence and characteristics of the QGP, a large fraction of these particles has to be detected and identified.

2.2.1 The Detectors

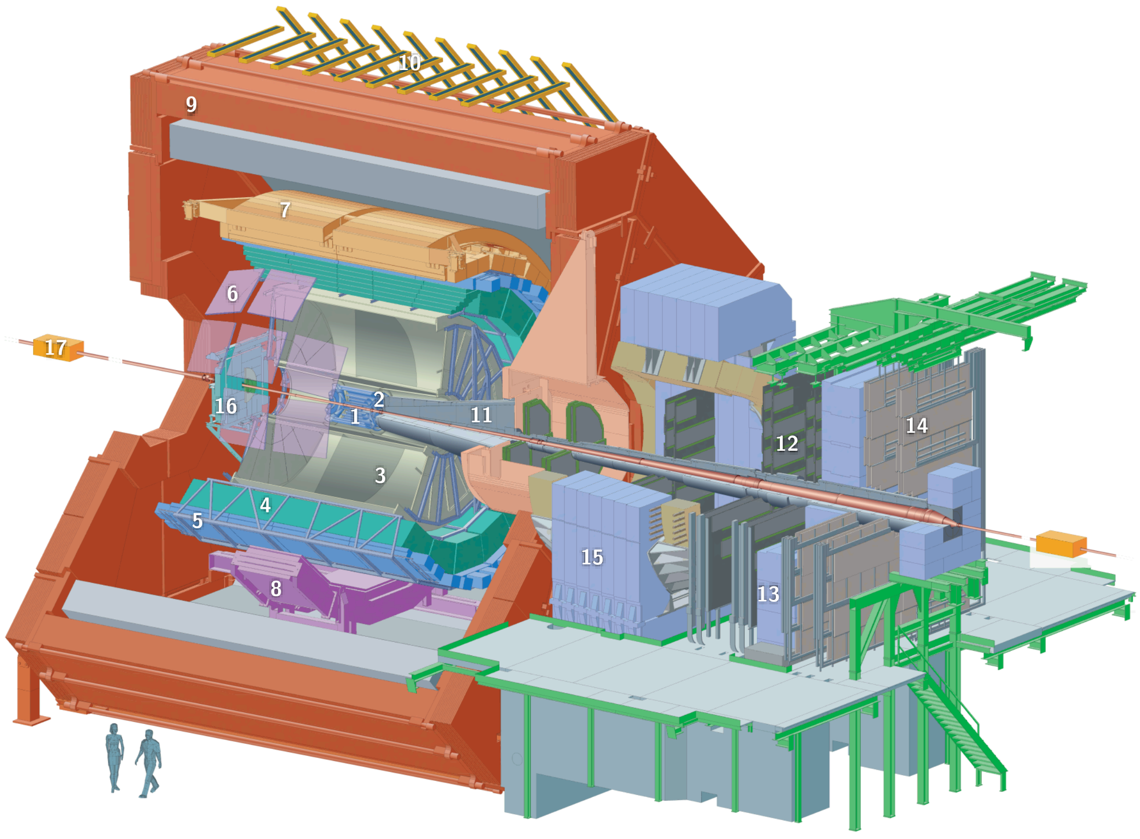
The ALICE experiment uses several complementary detectors with a total size of 25 m in length and 16 m in height and a mass of 10 000 t. An overview of the experimental setup is presented in Figure 2.2. The setup is subdivided into central and forward region. The forward region (right half of the figure) consists of the muon subsystem and supplementary detectors like the Forward Multiplicity Detector (FMD), T0 and V0 [10]. The central region contains most tracking and particle identification detectors, which are described briefly in the following section.

The *Inner Tracking System (ITS)* consists of six cylindrical layers of silicon wafers that directly surround the beam axis in the central region — and thus the point of collision (inner radius about 3 cm, outer radius about 50 cm). With its particularly high resolution of up to 12 µm, it can precisely determine particle positions and points of interaction [4].

Particle tracking is continued outside of the ITS in a larger detector named *Time Projection Chamber (TPC)*. Measuring 5.1 m in length and covering the range up to 2.78 m in radial direction (x direction), the TPC is a large gas-filled volume, to which an electrical field is applied. When charged particles traverse the gas volume, they ionize the gas atoms, and the ionization electrons along the trajectory start drifting in the electrical field. By measuring the arrival time of the charges at electrode pads at the end of the chamber, the TPC reconstructs the trajectory of the original charged particles [7].

Covering the range of 2.94 m to 3.69 m in radial direction and 7.0 m in length, the *Transition Radiation Detector (TRD)* has the form of a hollow cylinder. It consists of six layers of drift chambers, each of which are attached to a layer of a radiator material that causes traversing electrons to generate transition radiation. In the drift chambers, both the generated transition radiation and the track of the particle are detected. A major task of the TRD is to act as a trigger detector deciding on the readout of the TPC [8].

The next detector in outward direction is the *Time of Flight (TOF)* detector, which utilizes 160 000 parallel disc counters to measure the particles' time of flight from the point of collision to the detector cylinder at a distance of 4 m to a precision of 150 ps to determine the mass of high-energy particles [6]. At even higher energies, the smaller *High Momentum Particle Identification (HMPID)* detector (14 m^2) takes over the task of determining the particle mass. Its principle of operation is based on measuring Cherenkov photons that are emitted by particles in a dielectric medium [3]. The *Photon Spectrometer (PHOS)*, located at the perimeter of the central region, is composed of lead tungsten crystals. It is built to record the temperature of collisions by detecting emerging photons [5].



1. ITS (Inner Tracking System)
2. FMD (Forward Multiplicity Detector), T0, V0
3. TPC (Time Projection Chamber)
4. TRD (Transition Radiation Detector)
5. TOF (Time of Flight Detector)
6. HMPID (High Momentum Particle Identification)
7. EMCal (Electromagnetic Calorimeter)
8. PHOS CPV (Photon Spectrometer Charged Particle Veto Detector)
9. L3 Magnet
10. ACORDE (ALICE Cosmic Ray Detector)
11. Absorber
12. Muon Tracking Chambers
13. Muon Filter Wall
14. Muon Trigger Chambers
15. Dipole Magnet
16. PMD (Photon Multiplicity Detector)
17. ZDC (Zero Degree Calorimeter)

Figure 2.2: The detectors of the ALICE experiment. The Transition Radiation Detector (TRD, 4) is the green colored component between TPC (3) and TOF (5). Image: CERN, used with permission.

The entire central region is enclosed by the L3 magnet, which produces a homogeneous magnetic field of 0.4 T parallel to the beam axis. This magnetic field deflects charged particles on their flight path according to their momentum, enabling the detectors to deduce the momentum of a particle from the radius of the particle trajectory.

2.2.2 The Trigger System

The detector systems of ALICE together generate a data rate of up to 20 TB/s. Nevertheless, they are by far not able to record all of the approximately 8000 Pb-Pb collision events per second because of the limited readout rates of most detectors. The TPC for example, the detector producing the largest amount of data, has a readout time of approximately 5 ms — corresponding to a maximum readout rate of 200 Hz.

Trigger	Time after interaction	Expected rate (Pb-Pb)	Remark
Pretrigger	0.3 µs	$\approx$ 5000 Hz	TRD specific wake-up
Level-0	1.2 µs	$\approx$ 5000 Hz	Strobe to sampling electronics
Level-1	6.5 µs	$\approx$ 400 Hz	Major rate reduction
Level-2	$\approx$ 88 µs	$\approx$ 200 Hz	TPC past-future protection
High-level	$>$ 1 ms	$<$ 100 Hz	Software trigger, data compression

Table 2.2: The ALICE trigger system uses a hierarchy of different trigger levels. The respective rates are exemplary estimates for the case of heavy ion interactions with high charged-particle density [9]. The actual values depend strongly on the running scenario.

On the other hand, only few of the collision events are particularly interesting with respect to the physics objective. The aim therefore is to reject the uninteresting events as early as possible and to initiate readout of the slower detectors only for the most relevant events. To this end, trigger detectors are used which supply a preliminary evaluation of the event shortly after the interaction. The TRD as a trigger system, for example, has a decision time of only 6 µs (see Section 2.4).

The trigger system of ALICE is divided into four sequential stages[8] as summarized in Table 2.2. The higher levels include additional data from detectors with larger readout time and more elaborate real-time data analyses. The trigger signals are produced from the information of the individual subsystems and distributed by the Central Trigger Processor (CTP).

The level-0 (L0) trigger, the first trigger stage, is issued 1.2 µs after the collision based primarily on multiplicity[9] information from the FMD. If the L0 is not issued, the other detectors discontinue processing at this time. After 6.5 µs, the CTP decides whether a level-1 (L1) trigger is issued. The decision is derived from results of the Zero-Degree Calorimeter and the Muon Spectrometer concerning the centrality and multiplicity of the event as well as the results of the TRD trigger system, which are available at that time.

The latency of the L1 trigger is constrained primarily by the readout principle of the TPC detector. Data acquisition starts only after L1 has been issued, while the electrons generated in the gas volume are continuously drifting towards the end caps. As data from before the trigger cannot be recovered, a later trigger decision would effectively reduce the active volume of the TPC.

If the L1 is issued, the TPC is read out and data of additional detectors is analyzed to decide on issuing the level-2 (L2) trigger. The final stage of the trigger system is the High-Level Trigger (HLT). At a maximum rate of 1 kHz, it analyzes the complete data from TPC, TRD and other detectors while already compressing it. The HLT triggers at

[8]A fifth, additional trigger stage is introduced by the TRD specific pretrigger signal.

[9]Number of (charged) particles that are produced in an interaction

most 100 times per second, causing the full event data to be stored permanently for offline analysis.

2.3 The Transition Radiation Detector (TRD)

Transition radiation generally occurs when moving charges cross the boundary surface between two media with different refractive indices. An electrically charged particle that moves towards such a boundary surface forms an electrical dipole with its mirror charge, whose strength of field changes while the particle approaches the boundary and disappears at the moment of its entrance. The change in strength of the dipole field leads to the emission of electromagnetic radiation. The transition radiation is emitted into a small conical angular range around the flight direction of the particle. In this experiment, the deviation from the flight direction is negligibly small. The wavelength of the transition radiation is in the range of X-rays.

The total energy of the transition radiation increases according to the Lorentz factor $\gamma = \frac{E}{mc^2}$ of the particle [27]. Because of its dependence on the particle's mass, this effect can be used to identify high-energy charged particles.

The detector modules of the transition radiation detector are constructed in layers. At the inner side, a radiator, a setup of Rohacell HF71 foam and polypropylene fibers, provokes the emission of transition radiation. Thanks to the inhomogeneous structure of the material, each particle crosses many boundary surfaces, increasing the probability for the emission of detectable radiation.

Attached to the radiator is a drift chamber, a volume filled with a special gas mixture (85 % xenon, 15 % CO_2). In the drift chamber, both the traversing particles and the transition radiation photons produce ionization traces. While the photons release their energy almost instantaneously causing charge signals at the inner side of the chamber, the traversing electrons and pions generate continuous ionization traces. In both cases, the applied electrical field causes the ionization electrons to drift towards the cathode wire plane at a constant speed.

The free electrons generated immediately by an ionizing particle crossing the drift chamber are not sufficient to produce a detectable signal without further amplification. Therefore, gas amplification is employed in the outer region of the drift chamber. The electrons that have drifted past the grounded cathode wires are accelerated by a strong electrical field generated by a high voltage (+1.4 kV) at the anode wires. They ionize more gas atoms and thus initiate an avalanche effect, effectively amplifying the total charge by a factor of 4000 [15]. While the electrons vanish rapidly via the anode wires, the gas ions, which drift slowly out of the amplification region, produce induction charges at the cathode pads at the outer surface of the chamber.

From the time profile of the signal at the individual cathode pads, constant drift velocity provided, the course of the original track can be concluded. The time of arrival of the signal

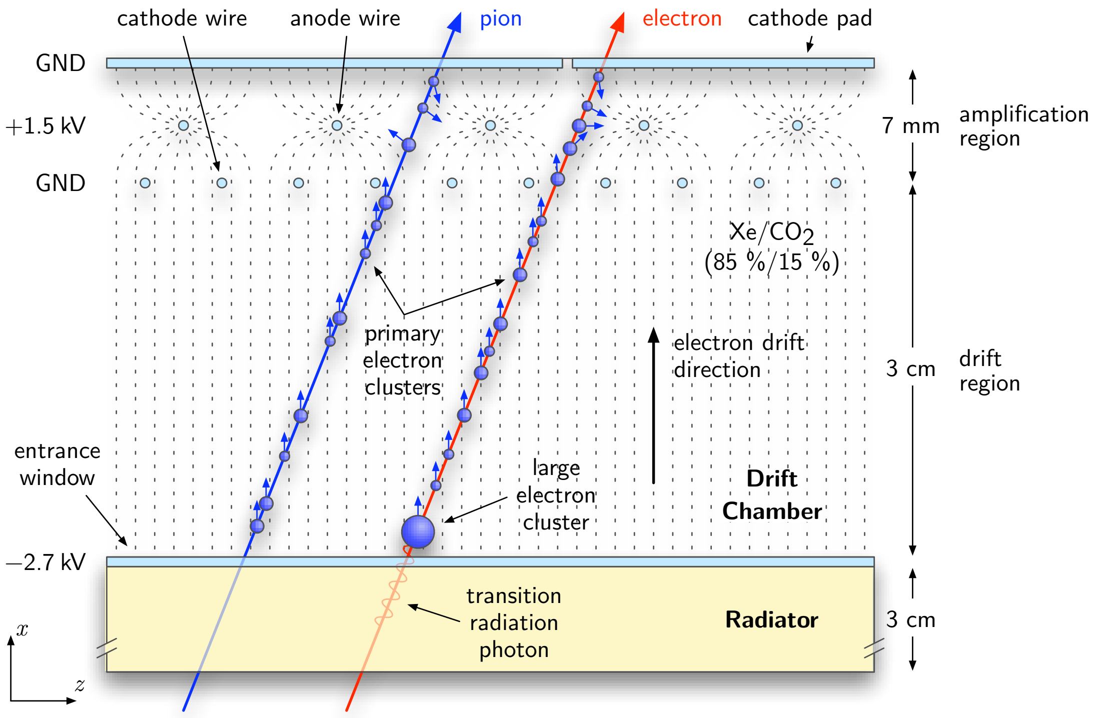


Figure 2.3: The TRD principle of operation, shown in a projection of part of a module onto the x-z plane. High-energy particles like electrons and pions produce traces of ionization in the gas volume. In case of an electron, the transition radiation photon generates a large electron cluster (blue sphere) near the entrance window. The produced electron clusters drift towards the anode wires near the cathode pads.

at the cathode corresponds (with reverse sign) to the original x coordinate of the charge cluster. Figure 2.3 illustrates the fundamental functionality of the detector modules.

The charge induced on the cathode pads is amplified and converted into a voltage signal by charge-sensitive preamplifiers. Analog-to-digital converters (ADCs) sample the signal and convert it into digital information. The first part of the online analysis electronics is located as multi-chip modules (MCMs) on readout boards directly on the chambers.

The TRD is composed of 540 detector modules. A module consists of 12 to 16 pad rows of 144 channels (different cathode pads) each and 96 to 128 MCMs. Each MCM processes the data of 18 neighboring channels within a pad row. The modules are arranged to form a hollow cylinder: 18 in azimuthal direction (φ), five rings in longitudinal direction (z), and six layers in radial direction (x). Figure 2.4 schematically illustrates the structure of the detector. The canonical names of the detector components are specified in [8, p. 243]. A *module* designates the compound structure of radiator and drift chamber, a *layer* consists of 5 modules in longitudinal direction, a *stack* is six modules in radial direction. A *plane* consists of one layer for each of the 18 azimuthal angles φ, i. e., $18{\times}5$ modules.

The TRD is intended both for track reconstruction and particle identification during offline

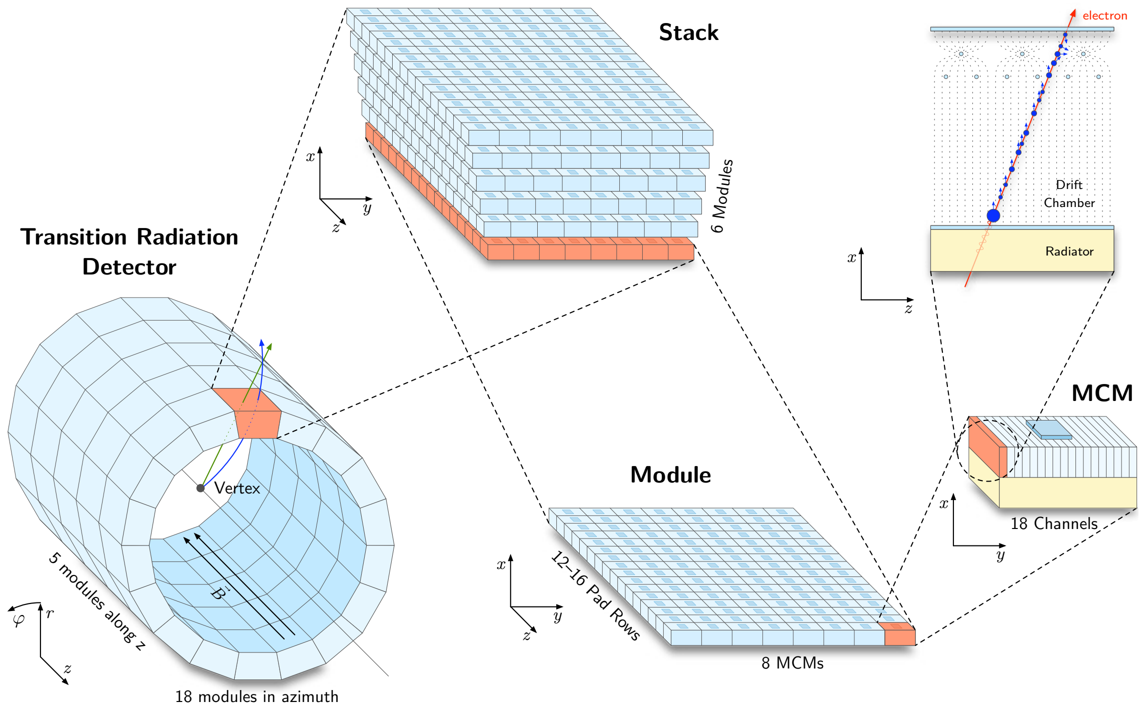


Figure 2.4: The TRD setup is hierarchically structured with detector components named *stack*, *module*, and *MCM*. In total, the TRD is composed of 540 detector modules.

analysis and as an online trigger system. Since the application as a trigger puts far higher demands on the performance of the detector electronics, its design is focused primarily on this task.

2.4 The TRD Trigger System

An important opportunity to study the characteristics of QGP is analyzing the occurrence of heavy vector mesons (J/ψ, Υ) [8, 78]. The dependence of the production probability on the centrality of the collision event is particularly interesting. Since for example Υ resonances are produced only rarely (approximately every 10^5 events), their detection rate has to be increased by a special trigger to gain sufficient statistics. This task is performed by the TRD. Υ particles can be detected via their decay into an electron-positron pair with both particles having a high transverse momentum (typically $p_t > 3\,\mathrm{GeV}/c$). A major objective of the TRD trigger system therefore is to detect electrons and positrons with high transverse momentum p_t.

Particle Identification The transition radiation enables the TRD to differentiate in particular between electrons and pions, which emerge in much larger numbers within the same transverse momentum range. Compared to electrons, pions have a 273 times larger rest

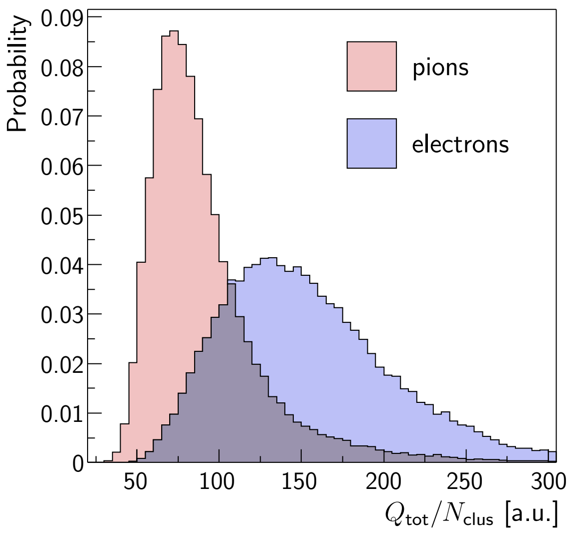


Figure 2.5: Pulse height distribution for electrons and pions with transverse momentum $p_t > 3\,\text{GeV}/c$, integrated over all time samples of a chamber. Source: test beam data, adapted from: [8, p. 102]

mass ($m_{\pi^\pm} = 140\,\text{MeV}/c^2$ compared to $m_{e^\pm} = 0.511\,\text{MeV}/c^2$). Since transition radiation effectively occurs only above a threshold value of $\gamma \approx 1000$, pions at energies below approximately 140 GeV produce no transition radiation photons, but only deposit the energy lost by ionization in the detector [36]. With electrons, the signal of the transition radiation photon is superimposed on the actual ionization track signal. Thus, electrons can be recognized by an on average higher measured charge. The particles' different ionization loss (described by the Bethe-Bloch formula [58]) additionally intensifies this effect.

Figure 2.5 shows the probability distribution for the measurement of a certain total charge (pulse height) in case of electrons or pions. The large overlap region of the two distributions indicates that the total charge value measured in a single chamber does not suffice to unambiguously differentiate between electrons and pions. The selectivity increases considerably if the measurements of all detector layers are combined, because the tracks in the different modules represent statistically independent individual measurements [79]. As the transition radiation photons are absorbed with higher probability close to the radiator, the charge signals differ particularly toward the end of the drift time. Figure 2.6 shows that the average measured charge for electrons is generally slightly higher and further increases toward the end of the drift time, which corresponds to the entry point of the particle into the drift chamber. The signal peak at the beginning of the drift time is an artifact of the amplification region in the chamber.

Momentum Measurement Because of the longitudinal magnetic field, the particles follow circular paths in the observed plane (x-y plane) with radii depending on the transverse momenta of the particles according to $r = \frac{p_t}{e \cdot B}$. Therefore, the curvature of each particle's

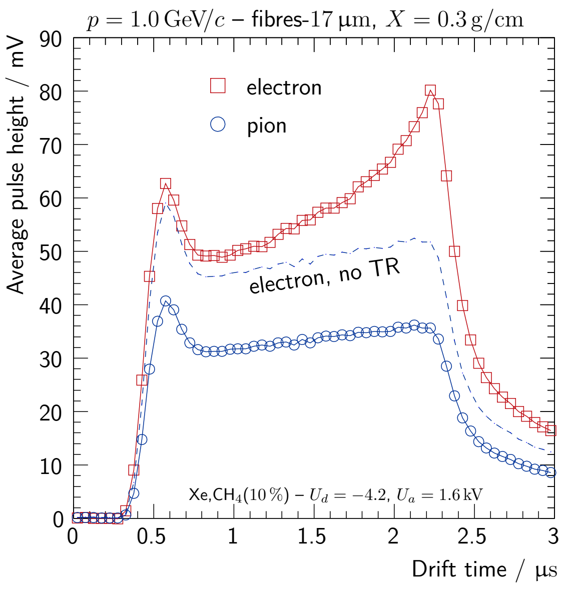


Figure 2.6: Sequence of pulse heights for electrons and pions over the drift time, integrated over many individual measurements. Source: test beam data, adapted from: [8]

trajectory in the x-y plane has to be determined to obtain the transverse momentum.

At the high transverse momenta of the particles that the TRD is designed to identify ($p_\mathrm{t} > 3\,\mathrm{GeV}/c$), the radius r of the circular path is large ($r = 25\,\mathrm{m}$ at $p_\mathrm{t} = 3\,\mathrm{GeV}/c$). Given the large radii, a very precise measurement of the track would be necessary to determine the curvature directly from the difference between the particle track and a straight line. This is not feasible inside the detector chambers, which only cover a range of 3 cm in x direction. In reality, the track segments appear in good approximation as straight line segments.

To nevertheless obtain a first estimation of the transverse momentum from the data of a single detector module, the primary vertex assumption is used, i. e., one assumes that the particle track originates directly from the beam axis. In this case, two additional points suffice to determine the radius of the circle — or, if the track inside the detector module is approximated by a straight line, which is regarded as a secant to the circle, the line parameters axis intercept and slope. A disadvantage of this method is that it allows secondary particles with small transverse momentum to be misidentified as primary particles with high transverse momentum.

To overcome this uncertainty and to increase the reconstruction accuracy, it is necessary to combine and analyze the data from all detector layers. The central unit that performs this task for the entire detector is a trigger processor named *Global Tracking Unit (GTU)*. This thesis focuses on the design of the GTU as a complex low-latency trigger processor capable of performing full online track reconstruction.

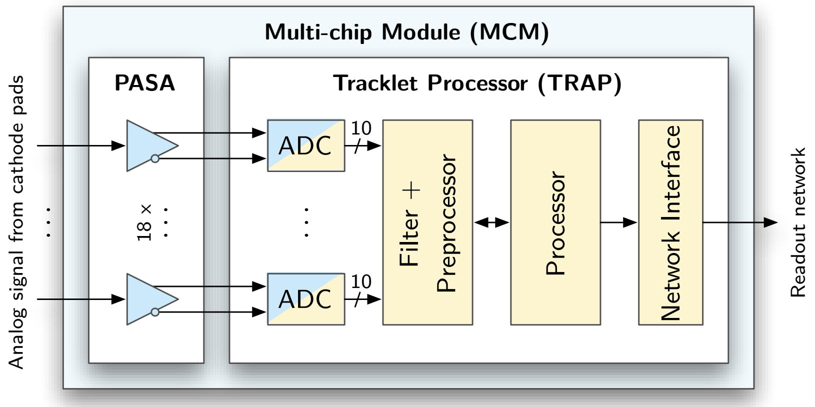


Figure 2.7: Data processing in a multi-chip module (MCM). Shown here are the schematic structure of the detector front-end electronics and the main data flow from the analog input signal to the readout network.

2.4.1 Layout of Trigger System and Detector Readout Chain

The large number of generated particles leads to a high occupancy[10] of up to 34 % in the 1.2 million detector channels. Considering the short time available for the trigger decision, it is necessary to massively parallelize the processing and to reduce the data early. The computations are therefore decentralized and performed locally as far as possible [17].

Electronic signal processing begins in the multi-chip modules (MCMs) directly on the chambers. The MCMs represent the front-end part of the online tracking. These approximately 64 000 MCMs work independently in parallel. As illustrated in Figure 2.7, they contain *Preamplifier/Shaper (PASA)* circuits[11] and the *Tracklet Processor (TRAP)* [53, 52]. Inside the TRAP chip, the signals are first digitized and digitally filtered [37]. The ADCs operate at a frequency of 10 MHz, sampling at about 20 points in time during the drift period of approximately 2 µs. This corresponds to a resolution in x direction (drift direction) of 150 µm.

By comparison with a threshold value, the preprocessor recognizes charge accumulations. To determine the y coordinate of a charge cluster, the preprocessor considers not only the position of the cathode pad carrying the strongest signal, but also the charge pattern on the two neighboring pads, resulting in a precision for the center of a charge cluster of approximately 400 µm — less than a sixteenth of the pad width of approximately 7 mm. In z direction, the resolution of the detector is much lower: in the individual layers, it can distinguish only between different pad rows of 7.3 cm to 10.1 cm in length.

During the drift time of the chamber, the measured y positions are used to already determine parameters of a best fit straight line. After the end of the drift time, the four processors per MCM compute in parallel from the precomputed intermediate results the

[10] Average percentage of detector channels with particle tracks at a time
[11] See [75] for details.

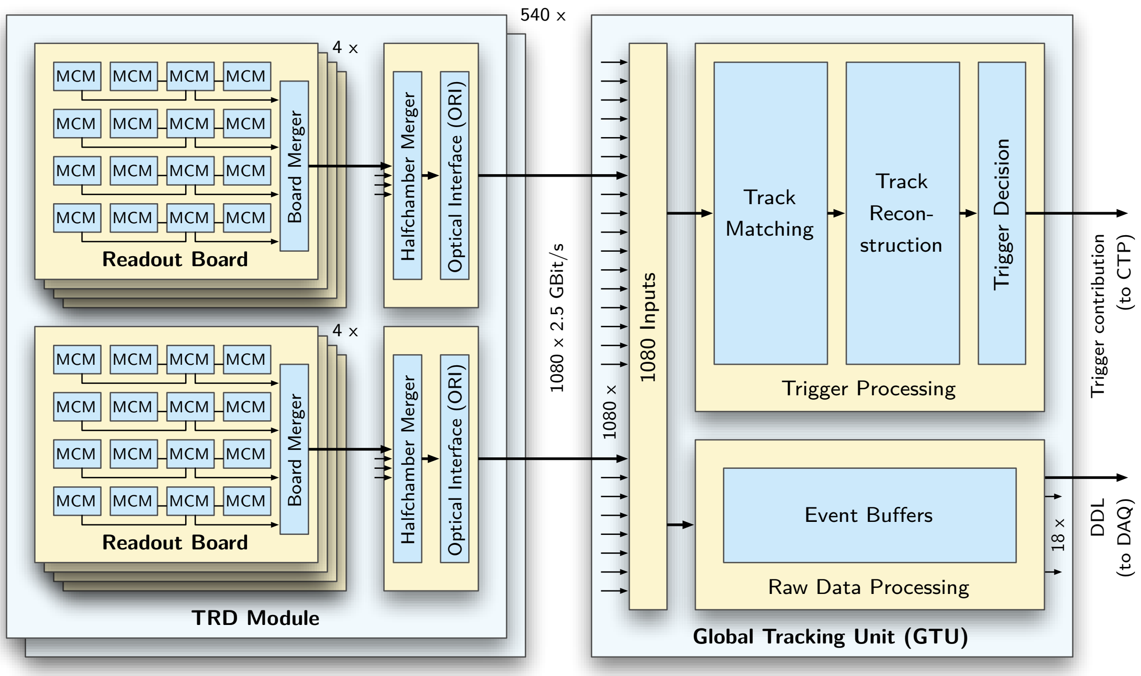


Figure 2.8: The TRD main data path with the GTU as its central component. Trigger and raw event data from the detector's MCMs is collected by the readout network and transmitted to the GTU via 1080 fiber optical links.

parameters axis intercept y_0 and slope $\frac{\mathrm{d}y}{\mathrm{d}x}$ of at most four track segments using linear regression. From these values, an estimate of the transverse momentum is obtained by applying the primary vertex assumption (see Section 6.2). Track segments indicating a transverse momentum below a certain threshold value of approximately $p_\mathrm{t} < 2.3\,\mathrm{GeV}/c$ are rejected at this stage. Parametrized data of the remaining, *stiff* track segments is handed over to a network interface and collected via a readout network [67]. The depicted preprocessing reduces the amount of data by a factor of 1000.

The detector electronics are configured through a specifically designed network [30]. Since the computation in the TRAPs is completely configurable, the exact parametrization of the track segments can be specified to best suit the specific trigger application. Considerations concerning an optimized set of parameters for high-p_t particle tracking are presented in Section 6.1.

The readout network, organized in a tree structure, collects data from 48 or 64 MCMs on 3 or 4 readout boards (a half module) through board merger and halfchamber merger MCMs. The data collected for each half module is then transferred via an optical link at 2.5 Gbit/s to the GTU, which is situated outside of the L3 magnet. Figure 2.8 shows the schematic structure of the TRD readout with the GTU as its central component.

The GTU, receiving track data from all detector layers of the TRD via 1080 optical links, constitutes the back-end part of the online tracking. At the GTU, data is first converted back into electrical signals by optical receivers. The arithmetic operations of the GTUs

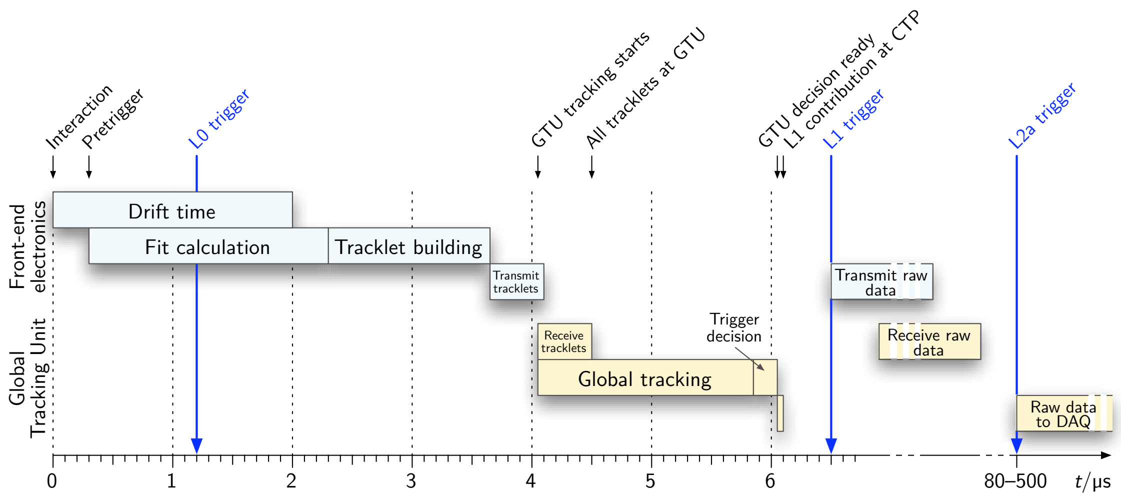


Figure 2.9: Timing of the TRD trigger sequence and raw data readout. Because of the detector drift time as well as data preprocessing and transmission latencies, the GTU has to complete the event reconstruction and analysis in less than 2 µs to contribute to the L1 trigger decision.

are divided into two main sections. In the first section, the trigger processor selects track segments from different layers that are likely to originate from the same particle and merges them to a continuous track. In the second section, the kinds of particles and their transverse momenta are reconstructed more precisely from the compound particle tracks. These results are analyzed by a trigger logic. If certain criteria on number and position of particles with high transverse momentum ($p_t > 3.0\,\mathrm{GeV}/c$) are fulfilled, a positive trigger decision is transmitted to the Central Trigger Processor (CTP).

Following a positive result of the trigger computations, the CTP can send a signal triggering a complete readout of the detector raw data for later analysis. In this case, raw data is transmitted via the same readout network and optical links. The GTU provides fast event buffering capability and eventually forwards requested events to the experiment's data acquisition (DAQ) system.

2.4.2 Timing of the TRD Trigger Computation

The latency of the TRD trigger computation is constrained by the specified L1 time of 6.5 µs after the interaction (see Section 2.2.2). As the CTP requires the trigger contribution at 6.1 µs, the computations have to be concluded before that time. Figure 2.9 shows the projected timing of the trigger readout sequence. The drift time of the electrons in the chamber is 2 µs, during which the calculation of the fit parameters is performed in parallel by the TRAP preprocessors. Processing time required by the processor for track segment (tracklet) building amounts to approximately 1.6 µs. Readout and data transmission generate a latency of 0.5 µs, so that the first track segments arrive at the GTU approximately 4 µs after the interaction.

The GTU immediately starts processing track data that has already arrived. After data transmission is concluded at approximately 4.5 µs after the interaction, 1.5 µs remain for the GTU up to the trigger decision at 6 µs. Within this short amount of time, the data of the 540 detector modules (in each case up to 40 track segments) has to be processed and analyzed completely.

The exceptionally short time interval available for reaching the decision imposes high demands on the design concept of the GTU. The algorithm should be parallelized as far as possible and optimized carefully. Since the computations have to be kept simple, the choice of algorithms is reduced significantly. Above all, the hardware architecture needs to be optimized for minimum latency. To minimize processing time, possible interactions with the other components of the readout chain should also be considered. All of these aspects are studied in the following chapters.

3 GTU Hardware Architecture

The outstanding performance requirements on the TRD trigger system are exemplary for the field of advanced trigger systems in particle detectors: from a large number of measured values (in this case: 24 million charge measurements) that occur distributed in space and time, a single piece of information is to be computed within a short amount of time (in this case: less than 4 µs).

The central part of this system, the GTU, is required to analyze up to 20 000 track segments from 540 detector modules and come to a trigger decision in very short time (approximately 2 µs). Data processing involves reconstructing and analyzing particle tracks across several detector layers. Exploiting the spatial data locality to allow concurrent execution of many computations, processing data from different regions in parallel, is necessary to satisfy the extreme low-latency demands.

In this chapter, a hardware architecture for the trigger processor GTU is presented. Built around the principle of low-latency data processing, it is a powerful, robust, and flexible trigger processor system.[1]

3.1 Structure and Conceptual System Design

During the design process, a system of this size and complexity requires many fundamental decisions based on trade-offs. In the following, the most important of these considerations are summarized.

3.1.1 Selecting Processing Node Technology

The trigger processor has to incorporate a number of processing nodes (whose number is determined by later considerations). There are several fundamentally different options with respect to the technology used for the processing nodes. Applicable alternatives are: standard *commercial off-the-shelf (COTS)* components (i. e., PC technology), PCs extended by a custom peripheral board, custom printed circuit assemblies (PCAs) with microcontroller and digital signal processor (DSP) chips, custom PCAs with programmable logic chips (FPGAs), and finally custom PCAs with dedicated application-specific integrated circuit (ASIC) chips.

[1]A concise overview of the architecture is provided in [24].

A system built solely from standard PC technology would not be able to meet the low-latency requirements simply because of the inevitable latency introduced by standard networking components. PCs equipped with a custom peripheral board could be a viable solution, as all data communication can be handled by the custom PCA with minimum latency. However, because of the optimization for bandwidth rather than latency and the lack of massive parallelism in today's PC CPUs, they are not well suited for the task of low-latency track reconstruction. As the main processing would have to be performed on the custom add-on board, it is doubtful that the additional system complexity and reliability issues introduced by standard PCs are justified. However, in systems involving complex high-level computations without tight low-latency demands, such as the ALICE High-Level Trigger (HLT), this setup is actually preferable. A custom PCA setup with dedicated ASIC chips is not feasible, not only because of the high costs involved, but also because the applied trigger algorithm has to remain configurable as the experiment's objectives and trigger scenarios evolve.

In contrast, FPGAs offer a high level of flexibility and performance. Furthermore, current FPGAs contain DSP blocks, which can be used to emulate much of the conventional DSP functionality. Given these prerequisites, an FPGA-based design is the best solution.

3.1.2 Optimizing the System Structure

Number of Processing Nodes

At first, a suitable granularity for the parallel processing has to be determined, considering the optimum utilization of data locality. There is an intrinsic conflict of goals here, as, on the one hand, plenty of parallel hardware means high parallel speed-up, but also a large communication effort and as a general rule higher overall cost. Few processing units on the other hand reduce the overall system complexity, but may not suffice to fulfill the low-latency requirements.

In the TRD, there are 540 detector modules arranged in 90 stacks. Each halfchamber transmits its data via a distinct link. To reconstruct tracks traversing the detector, data from all six layers is required. This suggests a finest granularity of one processing node per halfstack, involving 180 nodes in total. Other possible choices of granularity include one processing node per stack, per supermodule, or even a single node for the complete system. In this case, the latter two choices are ruled out by the limited number of inputs and logic resources available per FPGA.

The final choice of granularity results from the TRD geometry. As the mechanical structure causes a certain gap between the supermodules, high momentum tracks are unlikely to traverse more than a single supermodule. Because of the mostly projective geometry of the TRD, they are also unlikely to cross more than one module stack. However, the probability of interesting tracks traversing two adjacent halfchambers is considerable. In a 180-node setup, a serious amount of data exchange between computing nodes of adjacent halfchambers would be required, while in a 90-node setup, all nodes can process their data

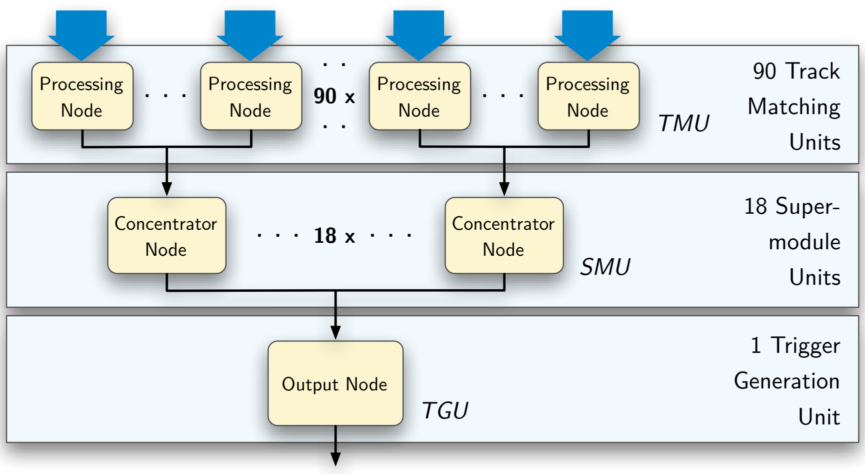


Figure 3.1: Abstract network system view of the TRD GTU. The terms displayed on the right side are the GTU designations of the different node types.

essentially autonomously. As data from 12 links can be handled by modern FPGAs, these considerations lead to the conclusion that the optimum number of processing nodes for the TRD GTU is 90, i. e., one node per stack.

Network Topology

As result of the online track reconstruction and analysis, a single trigger output signal has to be generated at a central location. To compute this signal, input from all processing nodes is required. As minimum latency is a primary concern, a star topology network is the obvious choice.

In case of the GTU, an additional layer of concentrator nodes has to be added, as a simple star topology would lead to the impracticable number of 90 individual input ports at the dedicated output node. The number of concentrator nodes does not affect the signal latency from processing node to output node. It can be chosen to best reflect the TRD structure of 18 supermodules. The resulting structure is a tree network topology with 90 processing nodes, 18 concentrator nodes and 1 central output node (illustrated in Figure 3.1).

System Hardware Structure

An important step in the design process is the decision on how the hierarchical system structure is mapped to a hardware architecture of PCAs and interconnections. This includes the question of the number of nodes per PCA. A high level of integration, that is, multiple nodes per PCA, reduces the effort required for inter-PCA connections and the physical size of the system as well as in some cases the total cost. A more modular design, however, leads to better production yield, improved mechanical stability, easier maintenance (by exchange of defective components), and possibly more relaxed cooling and power supply constraints. Especially the issue of production yield should not be disregarded with respect to today's large and expensive FPGAs with fine-pitched ball grid array (BGA) packages. The GTU utilizes a modular design implementing each node on a separate PCA.

Interconnections and Bus Systems In many applications, the types of interconnections between the PCAs are preordained by the electrical and mechanical standard of the node PCAs. It is generally favorable to follow an industry standard even when implementing a system built exclusively of full custom PCAs. Following an industry standard improves reusability and makes it possible to employ standard components for the mechanical setup, for power supply and distribution, and as backplane systems.

The GTU requirements on the performance of the communication channel between processing node and concentrator node are especially high, as they include both low latency demanded by the trigger feature and high bandwidth to enable full-speed readout of detector raw data. While the latency has to be less than 50 ns for the transmission of the first 32-bit word, the sustained bandwidth must at least amount to 1.6 Gbit/s net data rate to saturate the uplink to the DAQ system.

The parallel buses of current backplane-based standard systems for industrial computers, such as VMEbus and CompactPCI, do not comply with these requirements. Modern variations of these standards, such as the Advanced Telecommunications Computing Architecture (AdvancedTCA), provide high-speed serial point-to-point connections between the boards. However, the serial transmission involves serialization and deserialization of the data words and leads to an additional latency unacceptable for the trigger application.

Under the given circumstances, a custom backplane has to be used that combines high-speed parallel data transmission with point-to-point connections. It is nevertheless beneficial to utilize a standard backplane for all non-critical tasks. A split backplane solution allows to combine the benefits of application specific and standard backplanes. Some standard bus systems, such as CompactPCI, allow to choose from two different physical form factors: a system type occupying 3 rack units (U) and one with 6U. Combining the standard 3U system backplane with 6U printed circuit boards (PCBs) allows for an additional custom backplane of 3U.

The TRD GTU employs this solution by using a standard CompactPCI backplane with three rack units for power distribution, system control, and monitoring, and a dedicated custom backplane for high-speed low-latency trigger and raw data transmission.

In contrast to the interconnections between the processing nodes and the concentrator nodes, the interconnections between concentrator nodes and output node cannot be routed via a backplane because of the physical size of the system. Using standard networking technology is also impracticable because of the tight low-latency requirements of the trigger

Component (PCA)	PCB type	Required	Produced
TMU	XMU 1.2	90	108
SMU	XMU 1.2	18	20
TGU	XMU 1.2	1	2
XMU backplane	XMU-BP 1.0	18	20
TGU backplane	TGU-BP 1.0	1	2
CTP interface	CTP-IF 1.1	1	2

Table 3.1: Summary of custom PCAs for the GTU. A total number of 154 PCAs has been produced using four different types of PCBs.

transmission. The GTU utilizes a custom setup of differential cables and connectors for this purpose.

Custom Hardware Components Each fully customized component requires substantial effort not only in development, but also in production and testing. Thus, an optimized system structure typically minimizes the number of custom components. This includes not only employing standard components as far as possible, but also reducing the number of different custom components by utilizing the same custom components for different specific purposes if practicable.

In case of the GTU, the described system design includes four types of custom PCAs: one for each of the three node types, and the custom backplane connecting processing and concentrator nodes. Of these four types, three require similar components such as a main FPGA and its infrastructure, system memory, and interfaces for configuration and control. It is therefore desirable to unify the design of these three PCAs. But as they also have a considerable amount of diverging characteristics, especially with respect to required interfaces and connections, it is not easily possible to use the exact same type of PCA. The detailed system description in the following sections however demonstrates that it is indeed possible to utilize the same type of PCB for all three types of GTU nodes. This solution requires two additional types of straightforward PCBs to provide the output node with specific interfaces to the concentrator nodes and to the experiment's Central Trigger Processor (CTP).

These structural considerations lead to a total number of six types of custom components (PCAs) with 4 different types of PCBs. Table 3.1 summarizes the types of custom components and their respective number in the complete system. Reflecting their respective primary task or area of responsibility in the TRD GTU, the three node types are named Track Matching Unit (TMU), Supermodule Unit (SMU), and Trigger Generation Unit (TGU).

A block diagram of the full GTU system with its main hardware components is presented in Figure 3.2. In the following sections, the hardware architectures of these system components are discussed.

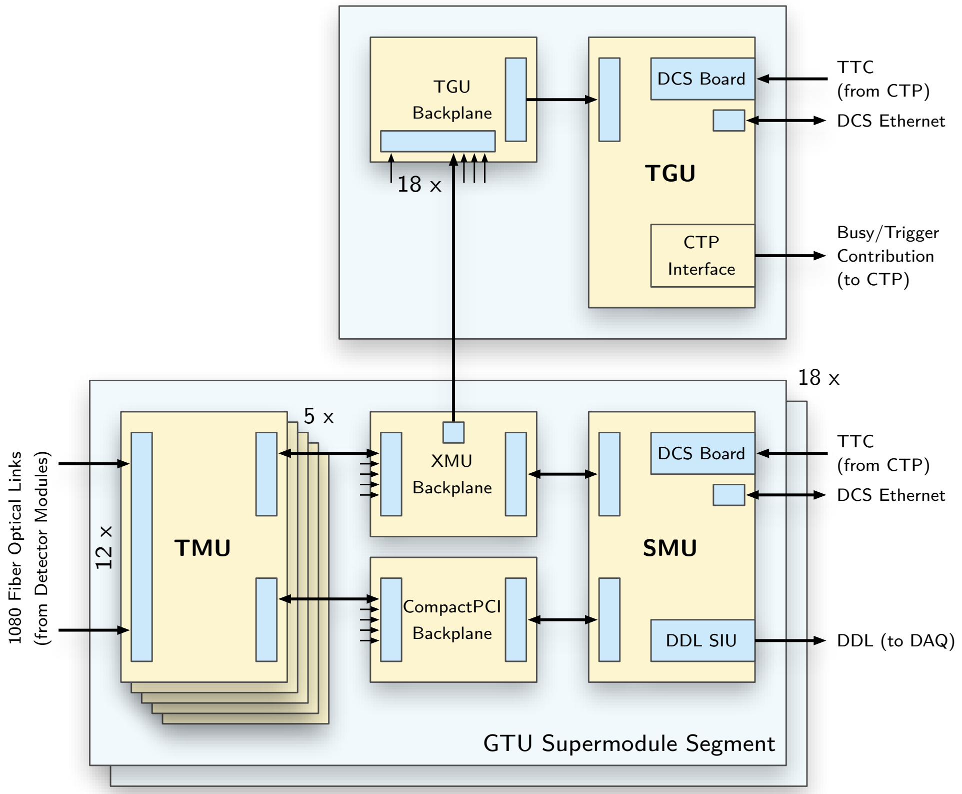


Figure 3.2: Block diagram of the GTU system hardware. Each of the 18 supermodule segments contain an SMU and five TMUs connected by common backplanes.

3.2 The Track Matching Unit (TMU)

The Track Matching Units (TMUs) are the main processing units of the GTU online trigger system. Its major components and main data path are highlighted in Figure 3.3. The TMU is based on a high-density Xilinx Virtex-4 FX FPGA. Its features (summarized in Table 3.2) include embedded multi-gigabit serial transceiver (MGT) and PowerPC processor blocks as well as a large number of logic cells and differential I/O pairs. An FPGA design for the TMU is discussed in detail in Chapters 7 and 8. In the following paragraphs, other major hardware components of the TMU module are introduced.

Optical Transceiver Modules To receive data from the associated stack of TRD modules via the optical readout fibers, 12 industry-standard small form-factor pluggable (SFP) transceiver modules are employed. SFP transceiver modules are highly integrated, pluggable modules widely used in telecommunication applications that provide optical-to-electrical and electrical-to-optical signal conversion at multi-gigabit signaling rates (see [71] for detailed specifications). The utilized modules are specified for signaling rates up to 4.25 Gbit/s.

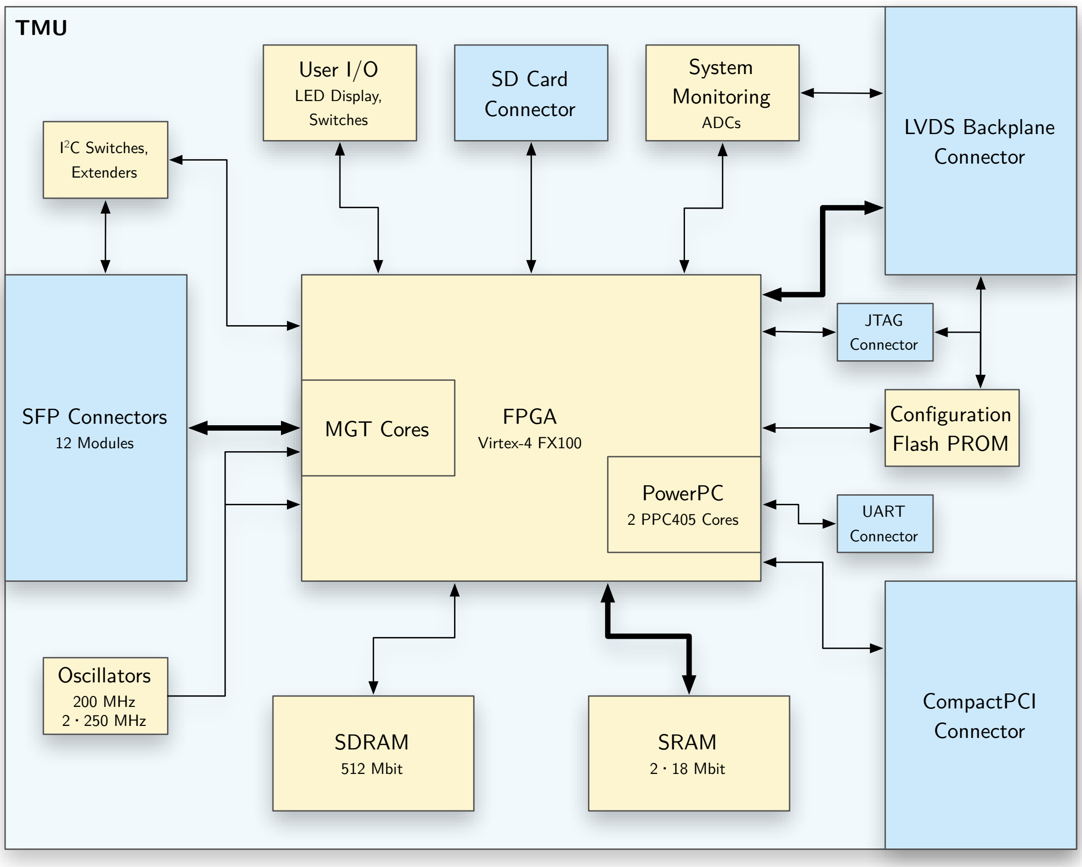


Figure 3.3: Block diagram of the Track Matching Unit (TMU). Bold lines represent the main data flow between multi-gigabit optical inputs, the SRAM, and the LVDS backplane.

In contrast to other optical receiver modules and possible custom solutions, the SFP modules always include a pair of transmitter and receiver circuits. The transmitter parts are not essential for the GTU operation, and the second LC connector occupies a certain additional amount of front panel real estate. However, SFP modules offer a number of important advantages. They are well tested and robust circuits that are provided by a number of different vendors and are competitively priced. Their embedded monitoring interface can provide valuable information like the intensity of the optical signal to the experiment's control system [72]. Finally, the modular design improves production yield and long-term system maintainability. The picture of the TMU PCA (Figure 3.5) shows that using minimum pitch, 12 SFP modules can be arranged on the front panel of the 6U PCA.

Each SFP module provides in addition to the high-speed differential transmitter and receiver lines six control and status signals at comparably low speed, adding up to 72 additional signals that should be connected to the FPGA. In a highly integrated system like the TMU PCA, FPGA I/O pins are a limited resource, which must be assigned primar-

Device family	Xilinx Virtex-4 FX
Part number	XC4VFX100
Speed grade	-11
Slices	42 176
Logic cells	94 896
PowerPC processor blocks	2
RocketIO serial transceivers	20 (12)
Total block RAM (kbit)	6768
Digital Clock Managers (DCM)	12
Phase-matched Clock Dividers (PMCD)	8
Maximum I/O pins	768 (448)
Total I/O banks	15 (11)
Maximum differential I/O pairs	384 (224)
Embedded DSP slices	160
Configuration memory (bit)	33 065 408

Table 3.2: Selected Parameters of the XMU FPGA. Values in parentheses indicate limitations imposed by pin compatibility of the XMU PCA to the smaller FX40 and FX60 devices.

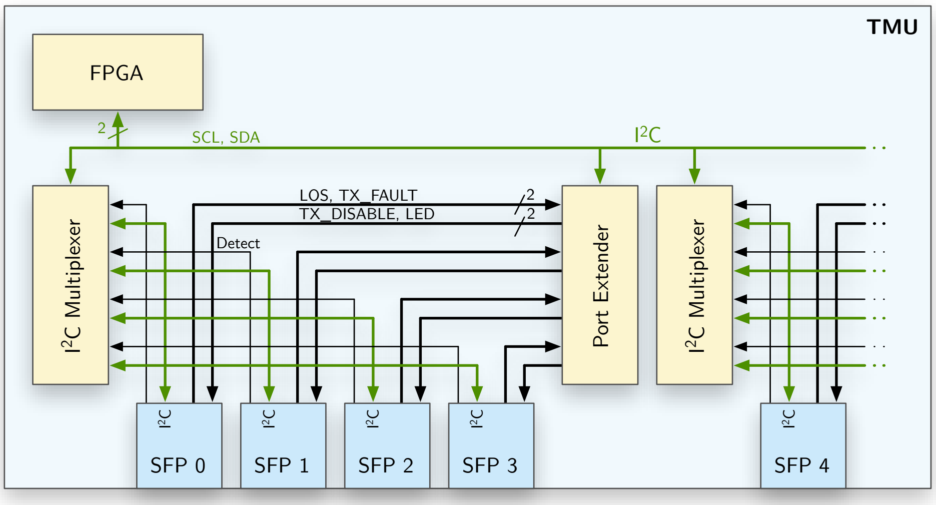


Figure 3.4: The configuration and monitoring interface to the multi-gigabit SFP transceiver modules on a TMU. The FPGA gains access to all 12 modules via a single I^{2}C bus.

ily to high-speed interfaces such as in this case a high-bandwidth memory interface and massive parallel connection to the backplane. By utilizing an array of Inter-Integrated Circuit (I^2C) multiplexers and port extenders, it is possible to reduce the number of pins required at the FPGA for the SFP control and status signals to two (Figure 3.4).

Buffer Memory During raw data readout, event data is received in parallel through all 12 optical links at an aggregate net data rate of 24 Gbit/s. Since the detector readout concept includes no handshake between GTU and the detector modules, the TMU has to be able to sink data at this rate, buffering it for later transmission to the DAQ system. As TRD events can be of considerable size (see Table 5.2), the on-chip block random access memory (RAM) resources available for this purpose do not suffice, and high-bandwidth external buffer memory is required.

The employed DDR2 static random access memory (SRAM) components can meet the bandwidth requirements. They combine a 200 MHz 2-word burst double data rate (DDR) interface based on the high-speed transceiver logic (HSTL) signaling standard with a 36-bit parallel data bus. Offering a capacity of 18 Mbit (including parity) each, they can store several black events. On the TMU PCA, two of these components are installed in parallel, sharing clock and address signals. As the SRAM architecture allows to write every cycle without exception, the full 72 parallel data signals can be written to at a rate of 400 MHz. This corresponds to a data rate of 28.8 Gbit/s.

Handling this data rate in a current FPGA is still demanding, as the interface data rate is near the reasonably attainable maximum operation frequency of the FPGA. As the application demands not only unrestricted write access to the memory, but also interlaced low-priority read access, a sophisticated FPGA design is required. A high-performance SRAM controller fulfills these demands. It is part of the TMU event buffering design described in [61].

The HSTL signaling standard at $V_{\mathrm{CCO}} = 1.8\,\mathrm{V}$ ensures reliable data transmission at high data rates. The signals are single-ended, but each input cell contains a comparator that generates the difference to a constant reference voltage $V_{\mathrm{ref}} = V_{\mathrm{CCO}}/2 = 0.9\,\mathrm{V}$ resulting in minimum uncertainty in the switching region. To avoid signal reflections, it is important to correctly terminate the bidirectional signals at both ends. During PCB layout, the signal termination at the SRAMs requires special attention, as the available space for termination resistors close to the parts is limited, and some signals, such as the read/write address, are a multidrop bus. At the FPGA side, the digitally controlled impedance (DCI) feature of the FPGA can provide internal termination if additional voltages are supplied at dedicated pin locations in each I/O bank. The XMU[2] PCB uses this technique. In addition, all SRAM signals should have matched trace lengths. While the Virtex-4 FPGA provides programmable input delay cells, there is no complement on the SRAM side and there are no easily programmable output delays. On the XMU PCB, the trace lengths are matched to less than 1 cm (equivalent to 50 ps) by meandering the shorter traces.

[2]XMU: TMU or SMU

Purpose	Number of signal pairs
Parallel LVDS I/O	54
Parallel LVDS input only	18
FPGA/PROM configuration (LVDS JTAG)	4
System management I^2C (DCB interface)	1
System management I^2C (XMU interface)	1
VCC_25	1
VCC_33	1

Table 3.3: Allocation of signals on the XMU custom backplane connector. The 80 differential signal pairs are primarily used for high-speed parallel I/O to and from the FPGA.

LVDS Backplane Interface On the XMU PCB, the low-voltage differential signaling (LVDS) standard is used for all data transmission via the custom backplane. The number and interconnection of signals is optimized for maximum bandwidth between TMU and SMU modules. The GTU utilizes differential HM-Zd board-to-board connectors, which are specifically designed for high-speed serial applications and rated for signal rates of up to 6.4 Gbit/s per differential pair. At the highest density available that is still compatible with the form factor predetermined by the 3U CompactPCI backplane, 80 signal pairs can be accommodated per interface. A small number of these pairs must be used for system configuration and management and to supply power to the active components on the custom backplane (see Section 3.5), leaving 72 LVDS pairs available for data transmission.

In the XMU design, given the aim of using the same PCB for both TMU and SMU modules, all of these pairs are routed directly to the FPGA, occupying four of the available I/O banks (see Table 3.4). As most of the FPGA's I/O cell pairs can be configured as either LVDS input or output, the direction of the LVDS signal can be determined by the FPGA designs of SMU and TMU. Special consideration is required when determining the signal interconnections on the custom backplane, since because of FPGA I/O bank limitations, only 54 of the 72 signal pairs can be configured as outputs. Table 3.3 summarizes the signals[3] connecting the XMU PCB to the custom backplane.

As all LVDS signals end at an FPGA where configurable input delay cells can compensate different signal propagation delays, the LVDS traces on the XMU PCA do not have to be matched in length. Experimental results with the final PCB show that at a data rate of 480 Mbit/s, no skew compensation is required for reliable operation.

CompactPCI Interface The CompactPCI interface available on the XMU PCB can be used in combination with a commercially available CompactPCI crate CPU to allow for easy access to the data inside the FPGA from a standard industrial PC system. The medium-performance interface via the CompactPCI backplane can help with problem-solving during the development of the FPGA design. In many applications, the employed

[3]Ground signals are not listed as the connectors feature specific ground contacts dedicated to each differential pair.

Peripheral Component Interconnect (PCI) bus, which provides a maximum throughput of 532 MB/s at 64-bit 66-MHz operation, will suffice as the primary means of data transfer. The CompactPCI interface also offers a number of bused reserved signals, which are routed to all XMU FPGAs on a backplane and are used for additional tasks such as slow control and reset synchronization.

The CompactPCI specification [56] permits two different signaling voltages on the PCI bus, 3.3 V and 5 V. CompactPCI system and peripheral boards may choose to support either or both of these voltages. For maximum flexibility, the XMU PCA is designed to support both voltages. Signal levels on the bus are converted to the FPGA's maximum I/O voltage of 3.3 V by bus switches. These switches also allow to temporarily segregate a system from the PCI bus. The XMU PCA is designed to fully support the CompactPCI hot plugging feature as specified in [55].

In the final GTU system, the custom LVDS backplane makes use of the PCI bus obsolete. The CompactPCI backplane, however, remains in use for industry-standard power distribution and slow communication between the PCAs. As no standard crate CPU is included in the final setup, the PCI bus signals can be used as additional general-purpose bused signals.

System Memory and Mass Storage Although the GTU configuration system outlined in Section 4.3 does not depend on it, the two embedded PowerPC 405 processor cores supporting clock rates of 400 MHz in the FPGA suggest the usage of a full operating system such as Linux for maximum flexibility in system configuration and control. However, such an operating system demands additional components, which are primarily an external working memory of adequate size and a mass storage device.

To allow for this application, the XMU PCA includes both devices. The external operating memory is realized as a single-chip DDR2 synchronous dynamic random access memory (SDRAM) component with a capacity of 512 Mbit (64 MB). For further flexibility, the PCB is designed to support the addressing of SDRAM parts of up to 4 Gbit capacity[4].

Using a data bus width of 16 bit at 200 MHz DDR, the SDRAM can be connected to a single I/O bank of the FPGA. All SDRAM connections use the stub series terminated logic (SSTL) signaling standard at 1.8 V with a reference voltage of 0.9 V. The PCB layout constraints are similar to those of the SRAM components, but since the SDRAM chip provides on-die termination and the FPGA's DCI feature is used, no external termination resistors are required.

As a mass storage interface, the XMU PCB is equipped with a Secure Digital (SD) card connector. By supporting both SD card protocol modes (cf. [70]), the simpler Serial Peripheral Interface Bus (SPI) mode as well as the high speed parallel access, the SD card connector provides a straightforward and efficient interface to inexpensive flash-based mass storage. Currently available Secure Digital High Capacity (SDHC) cards have a capacity of up to 32 GB.

[4]The maximum capacity in currently available compatible components is 2 Gbit (256 MB).

Management and User Interaction Additional peripheral components on the XMU PCB include devices for system management and configuration (see Chapter 4 for details) and direct user interaction. As the GTU system is designed to be monitored and maintained remotely, the direct user interface is limited to the display of status information as well as a small number of buttons and interfaces to support the development process. The only configuration switches on the PCB encode the PCA's serial number enabling each PCA to identify itself. On the front panel, a 5×7 dot matrix light-emitting diode (LED) display can be configured to show any relevant status information. All user interface components are connected through appropriate I^2C circuits to minimize the number of required FPGA user I/O pins.

Figure 3.5 shows a completely mounted TMU PCA. Ninety equally configured modules are installed in the GTU setup.

3.3 The Supermodule Unit (SMU)

The Supermodule Units (SMUs) are designed as middle-layer modules which support and control the TMUs for one supermodule. They act as concentrator nodes in the trigger system and provide interfaces to the experiment's DAQ and Detector Control System (DCS). Figure 3.6 shows the major components of an SMU module. Bold lines highlight the primary data path.

Sharing the same PCB, the SMU PCA also utilizes much of the same components as the TMU. In the following, only prominent differences to the TMU are summarized. The FPGA design for the SMU is discussed in Chapter 5.

Form Factor The most obvious difference to the TMU is the fact that the SMU module is designed for double width (40.64 mm instead of 20.32 mm). The extra space at the bottom side of the PCB is used to accommodate peripheral boards and additional connectors.

Interface to DAQ The interface to the experiment's DAQ system utilizes a specialized mezzanine board named Source Interface Unit (SIU), which is provided by the DAQ collaboration for all ALICE detectors (see [65, 64] for details). While the logical interface to the SIU does not conform to an established standard, the PCB and connector geometry complies with the PCI Mezzanine Card (PMC) standard IEEE 1386.1 [40].

A variety of I/O cards are available for the PMC standard, which is a specialization of the Common Mezzanine Card (CMC) standard IEEE 1386 [39] based on the PCI specification. For improved flexibility with respect to future applications, the XMU PCB is designed to fully support both the PMC standard and the SIU interface specification. To that end, the union of the signals defined in both use cases is connected directly to the FPGA. Of the up to four connectors specified for a PMC board, the SIU uses only one for essential signals. Combined with the fact that an SIU is only half the size of a PMC board, up to two SIUs can be mounted on the XMU's PMC connector setup.

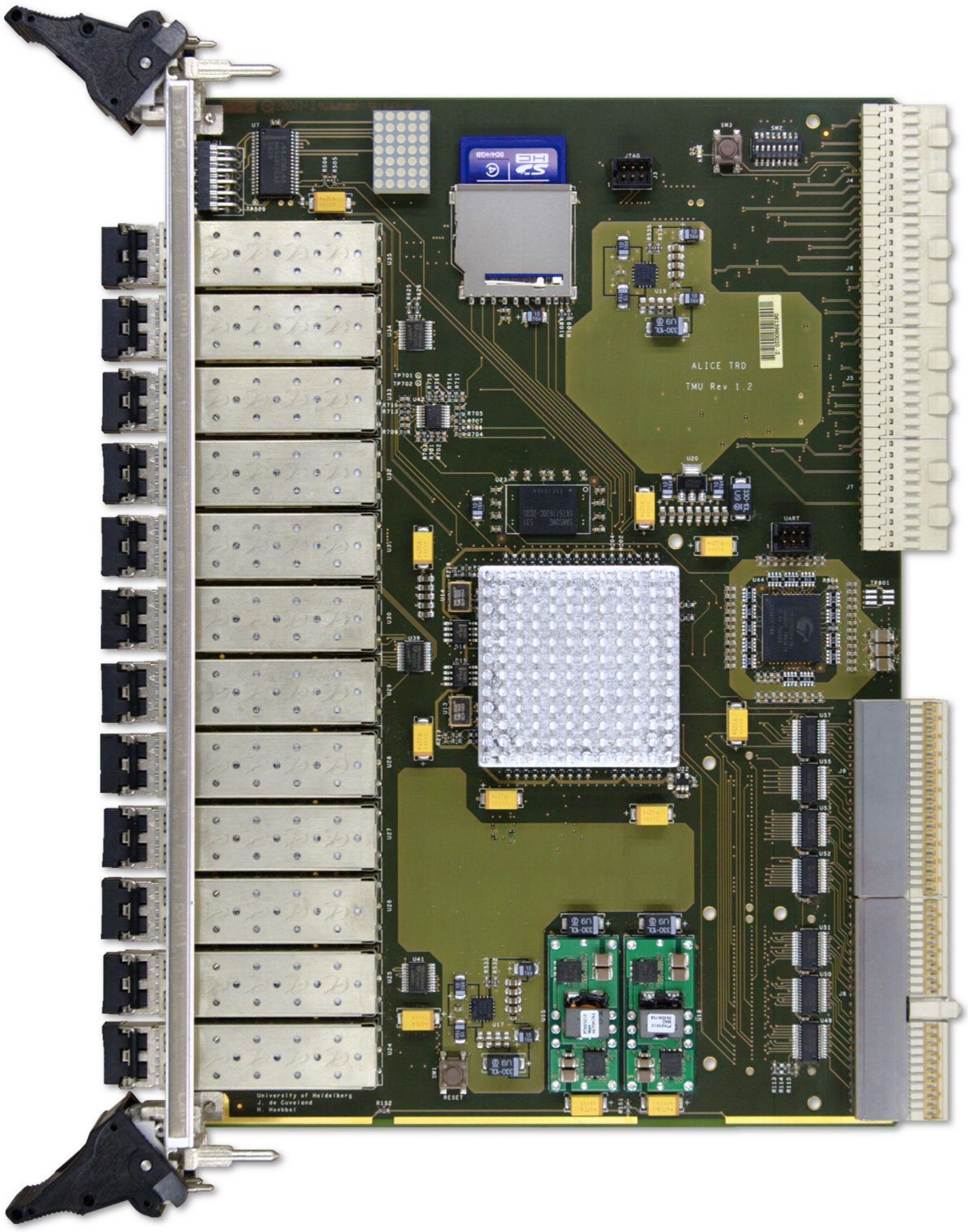


Figure 3.5: Photo of the TMU PCA. The 6U board has interfaces to a CompactPCI backplane (lower half) and a high-speed differential backplane (upper half). The FPGA is protected by a high-performance heat sink.

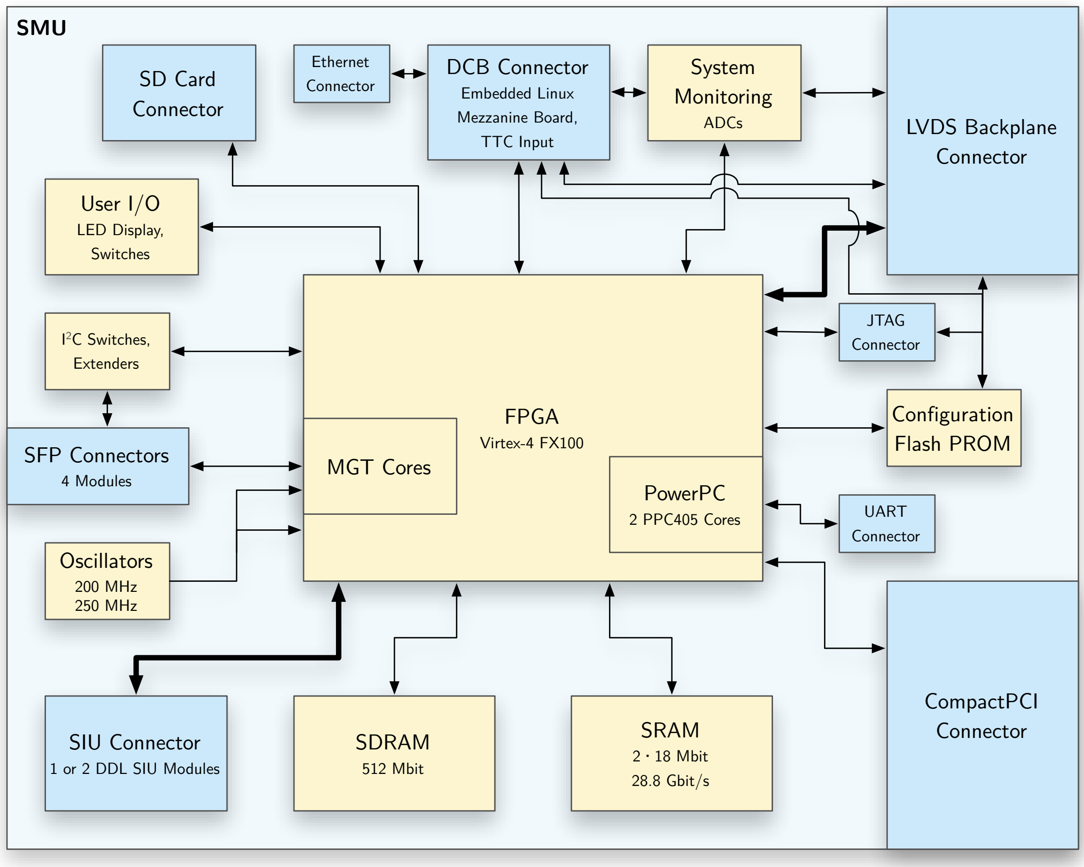


Figure 3.6: Block diagram of the Supermodule Unit (SMU). Bold lines represent the main data flow through the SMU.

The SIU interface specification [65] defines a simple source-synchronous communication protocol at a clock frequency of up to 50 MHz, whose unconventional timing requirements can be satisfied by the FPGA design. The 32 bit wide data bus at 50 MHz limits the data throughput to 1.6 Gbit/s. In contrast to the push architecture of the front-end data transmission, the Detector Data Link (DDL) implements a handshaking mechanism that can slow down the readout in case of DAQ congestion. For a detailed analysis of the TRD readout timing, see Section 5.1.

Fewer Transceiver Modules As the SMU does not receive data directly from the detector modules, the TMU's 12 SFP connectors are not required. Eight of them are omitted to make room for the additional components on the back side of the PCB. Four connectors are kept to allow for future enhancements of the SMU's functionality. Envisioned uses include an optical Gigabit Ethernet interface accessible from the internal PowerPC processor and support of higher DAQ data throughput by implementing the DDL protocol and providing additional links.

Reversed LVDS Signals Regarding the LVDS connections to the backplane, the SMU uses the same traces, but with reversed signal directions to receive data from the five TMUs through dedicated parallel buses and to transmit relevant trigger information to the TGU via twisted pair cables with four differential pairs. As the backplane connector provides 72 LVDS signal pairs in total, each TMU-SMU connection is limited to $\lfloor \frac{72-4}{5} \rfloor = 13$ signals.

DCS and TTC Interfaces Although the GTU is designed not to require constant monitoring, acting in many ways as an autonomous system, a remote access system is required to change running parameters, perform firmware upgrades, and retrieve information on the status of the system and the links to DAQ and detector modules if necessary. To this end, the SMU is interfaced to the ALICE DCS, which is based on Ethernet and TCP/IP. The interface is realized by employing a slightly adapted version[5] of the ALICE DCS board, which is also used by the TRD front-end modules and several other ALICE detectors (see [63] for reference). The DCS board is an FPGA-based embedded Linux system running customized server processes. An overview of the GTU monitoring and configuration system is presented in Chapter 4.

The DCS boards also provide the indispensable interface to the experiment's Timing, Trigger and Control (TTC) system. The TTC system distributes real-time signals to all ALICE detectors. The distributed signals include trigger pulses, additional trigger information, and a common clock reference synchronous to the beam interactions (LHC clock). This information is required by the trigger as well as the raw data readout functionality of the GTU. Because of limited processing and I/O pin resources on the Detector Control System Board (DCB), the serial TTC data streams are forwarded to the SMU for processing. Handling the different trigger sequences and distributing appropriate control signals in the GTU system as well as detecting and reporting trigger error conditions is an involved process. The corresponding FPGA design is described in [47].

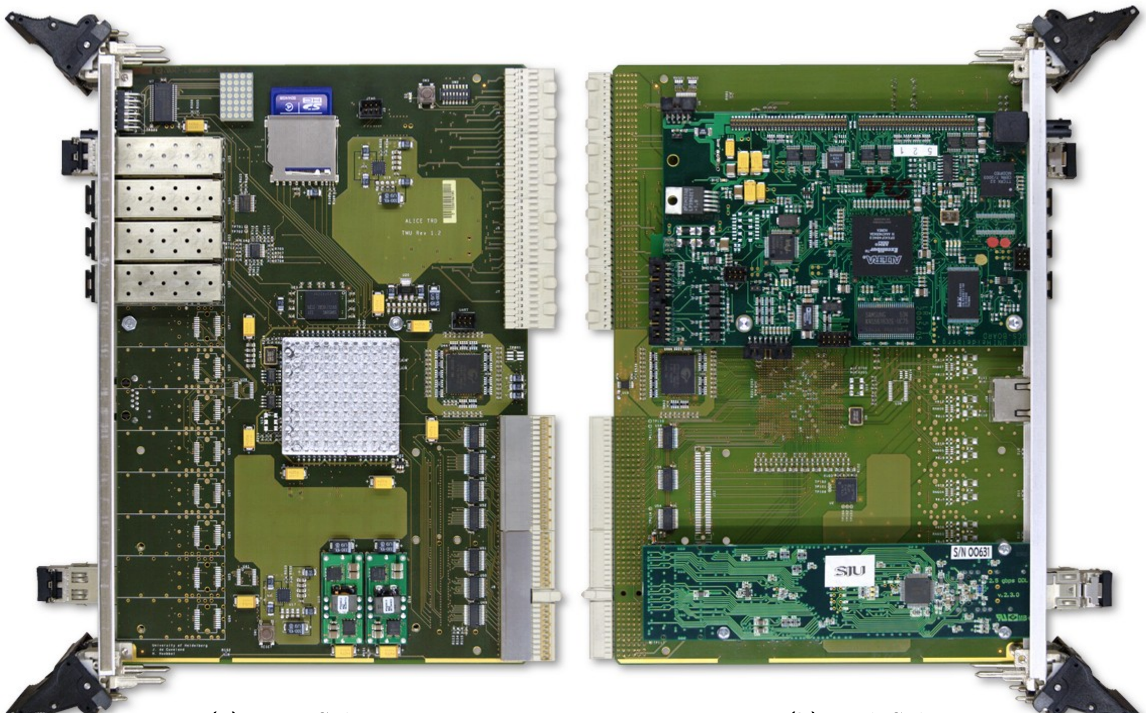
Figure 3.7 displays a fully equipped SMU PCA. The GTU setup comprises 18 of these modules.

3.4 The XMU PCB Design

The XMU PCB is the common basis for the TMU, SMU, and TGU modules. This section summarizes important aspects of the PCB design that are independent of the individual application.

FPGA I/O Assignment Centered around a large Virtex-4 FPGA device, the XMU PCA accommodates a number of essential peripheral devices. As most of these devices should be connected to the main FPGA, sensible usage of the FPGA's user I/O pin resources is indispensable. Figure 3.8 presents an overview of the Virtex-4 FX100's I/O pins.

[5]The DCS boards in the GTU system lack a stand-alone voltage regulator circuit. Instead, they use the 3.3 V GTU system voltage directly to reduce power dissipation.



(a) Front Side **(b)** Back Side

Figure 3.7: Photo of the SMU PCA. On the back side, two additional PCAs are mounted: the DCS board (top) and the SIU (bottom).

Bank	Pins	V_{IO}	I/O Standard	Usage
0	16	3.3 V	LVCMOS	JTAG and configuration PROM Interfaces
1	16	3.3 V	LVCMOS	User Interface, I^2C, SD-Card
2	16	2.5 V	LVDS	Backplane LVDS I/O
3	16	3.3 V	LVCMOS/PCI	UART, DCB, additional CompactPCI
4	16	2.5 V	LVDS	Backplane LVDS I/O, clock inputs, DCB
5	64	3.3 V	PCI	CompactPCI, PMC module
6	64	1.8 V	HSTL	SRAM
7	64	2.5 V	LVDS	Backplane LVDS I/O
8	64	2.5 V	LVDS	Backplane LVDS I/O
9	64	3.3 V	PCI	CompactPCI, PMC module
10	64	1.8 V	HSTL	SRAM
11	64	3.3 V	PCI	Additional PMC modules (FX60/100 only)
12	64	1.8 V	SSTL	SDRAM (FX60/100 only)

Table 3.4: I/O bank usage of the XMU FPGA. In total, five I/O standards at three different I/O voltages are used. Only optional connections are assigned to banks 11 and 12, as these are not available on all FPGA variants.

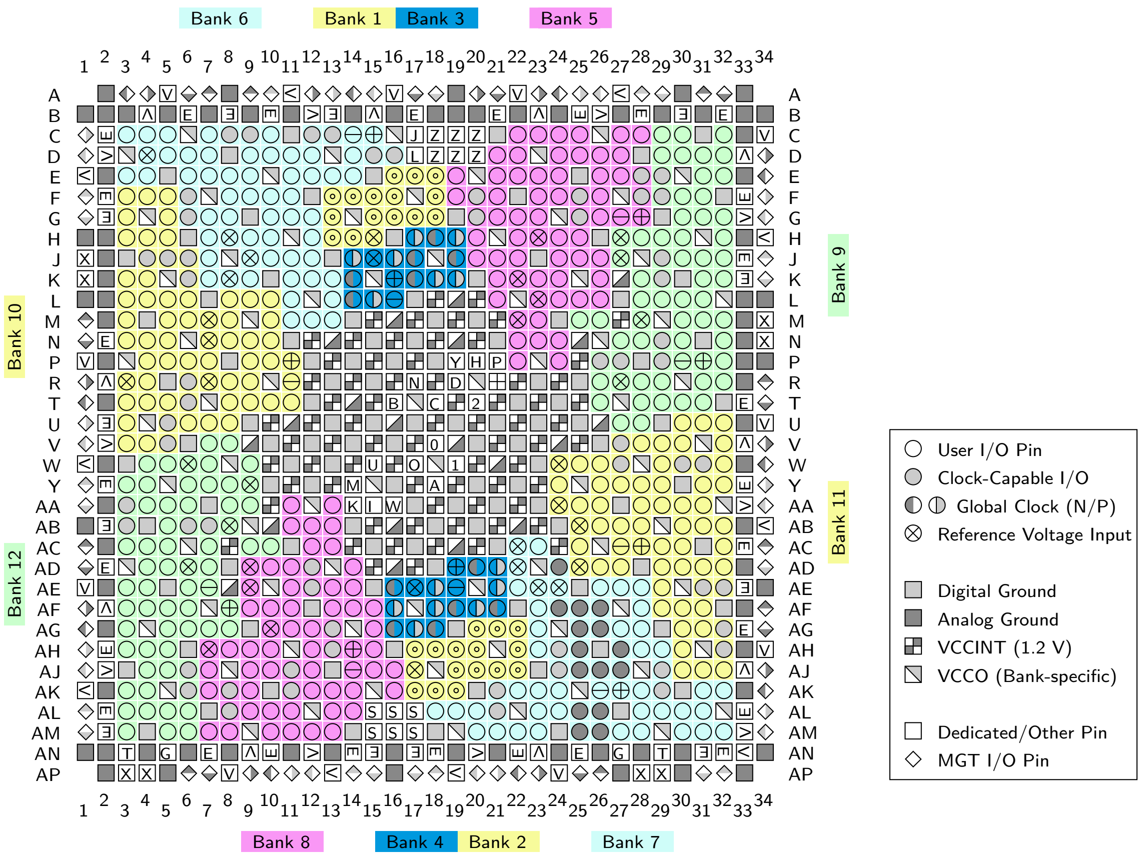


Figure 3.8: Pinout of the Xilinx Virtex-4 FX100 FPGA in the FF1152 package. A detailed list of all pins can be found in the *Virtex-4 Packaging and Pinout Specification* [85].

Assigning the FPGA's I/O resources involves a number of considerations[6]. The related constraints are detailed in the *Virtex-4 FPGA User Guide* [87]. In addition, taking into account pin placement on the FPGA package facilitates signal routing in the PCB layout process. The I/O bank usage of the XMU FPGA is presented in Table 3.4.

The XMU PCB is realized as a 14-layer PCB in a 6U Eurocard form factor. Selected parameters are summarized in Table 3.5. As the design includes a large number of high-speed signals, layout of the PCB is driven by signal integrity considerations, which are discussed in the following section.

3.4.1 Signal Integrity

At all high-speed PCB traces, signal integrity issues are a major concern. This involves not only reflection noise, but also crosstalk noise, and power/ground noise.

[6] I/O, reference and termination voltages have to comply to a set of banking rules specified by the vendor. Fixed clock regions, special pins, and simultaneous switching output (SSO) limitations further confine the assignment.

Number of layers	14
Number of signal layers	8
Insulator material	FR-4
Insulator dielectric constant	4.5
Dimension	233.35 mm × 160 mm
Total thickness	1.717 mm
Minimum trace width	75 µm
Minimum trace spacing	75 µm
Via technology	Through hole
Differential traces	Edge-coupled

Table 3.5: Selected parameters of the XMU PCB

Ground Bounce and Power Supply Noise Precautions taken against ground bounce and power supply noise include several dedicated power and ground planes and dedicated low-resistance connections to all components. The XMU power supply decoupling network includes more than 400 distinct capacitors of different types and values (cf. [84]).

Crosstalk Crosstalk, which is noise induced on a signal by another signal, can best be avoided by physically separating traces either vertically by a shielding plane or horizontally by sufficient line spacing. The XMU layer stackup (Figure 3.9) demonstrates how power and ground planes can be utilized to shield the signaling layers. Only two signal layers (Signal 3 and Signal 4) are directly adjacent. For signals routed basically in parallel (as is the case for the wide buses between FPGA and SRAMs), trace density has to be reduced and special caution is required on these two layers to prevent crosstalk.

To ensure a successful implementation, the resulting board layout is analyzed by simulating the analog behavior. Using signal integrity tools, the simulation models of the PCB traces are automatically extracted from the PCB layout. Together with models of the I/O buffer cells of the devices, which are provided by most vendors in the IBIS or SPICE format, the analog system response to a pulse signal can be simulated. If the result is not satisfactory, the layout should be reiterated and analyzed again before it is submitted for PCB production.

Figure 3.10 demonstrates the process. Both diagrams show in the upper part the worst crosstalk induced on any constant signal on the PCB. As the crosstalk signal is different at the transmitter and at the receiver, both signals are shown. The output is driven at a constant high level. In the lower part of the diagram, the aggressor signal with the most influence is shown. For the simulation, the aggressor signals are driven by a clock signal with a frequency of 200 MHz. Figure 3.10a shows the worst crosstalk before layout optimization. The amplitude of the induced crosstalk signal reaches 260 mV peak-to-peak at the receiver. While this is in general not sufficient to toggle the logic level of a static signal, the induced peaks can influence the timing of a signal's transition as perceived

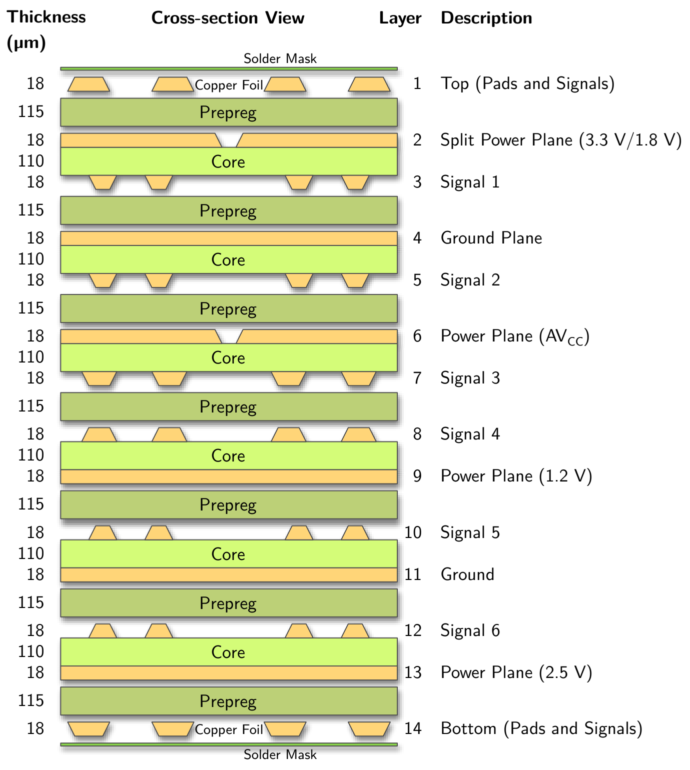
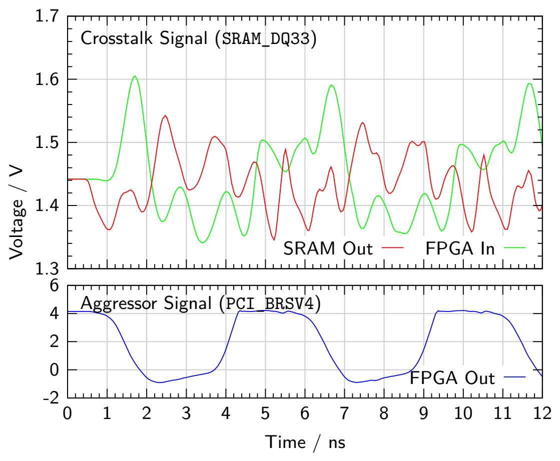
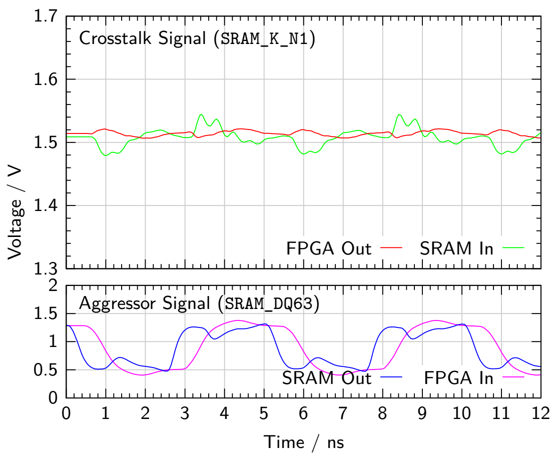


Figure 3.9: The XMU PCB stackup. The 14-layer PCB has a total thickness of 1.7 mm.



(a) Crosstalk before optimization



(b) Crosstalk after optimization

Figure 3.10: Crosstalk affecting SRAM signals (from simulation). The selected signals exhibit the worst crosstalk from any source before (a) and after (b) optimizing PCB trace geometry for minimum crosstalk.

Label	Width (µm)	Spacing (µm)	Imped. Z (Ω)	Diff. Z (Ω)	Layer
Top	190	300	52.4	99.5	1
Signal 1	75	150	51.4	95.7	3
Signal 2	75	150	51.4	95.7	5
Signal 3	100	200	52.3	97.1	7
Signal 4	100	200	52.3	97.1	8
Signal 5	75	150	51.4	95.7	10
Signal 6	75	150	51.4	95.7	12
Bottom	190	300	52.4	99.5	14

Table 3.6: The XMU PCB utilizes 8 signal layers. At the specified trace geometries, single-ended and differential impedance are close to $Z_{\mathrm{se}} = 50\,\Omega$ and $Z_{\mathrm{diff}} = 100\,\Omega$ on all layers.

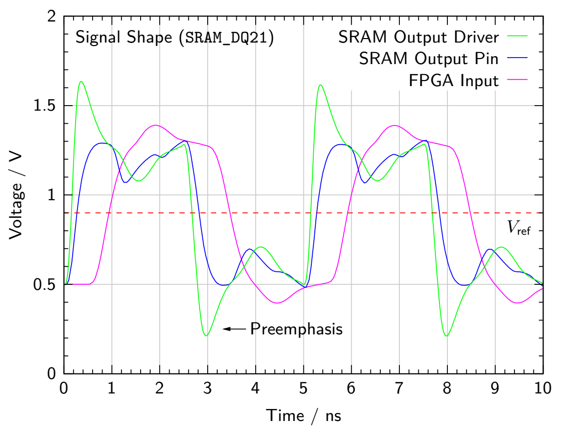
by the receiver, thus reducing the available timing margin for high-speed signals. After careful rerouting (Figure 3.10b), the strongest crosstalk amounts to only 60 mV.

Reflection One of the most demanding parts in a signal-integrity driven design is to avoid reflection noise, which can cause overshoot, undershoot and ringing in high-speed systems. Reflection noise occurs at any impedance discontinuity along the signal transmission path. Besides proper signal termination, important countermeasures against reflection noise include impedance matched PCB traces and a minimum number of vias. The trace impedance depends not only on the geometric dimensions of the copper trace but also on the surrounding material and the distance to the next conducting reference planes. To match the impedance of traces on different PCB layers, distinct trace geometries are required. The single-ended high-speed traces on the XMU are designed for a target impedance of approximately 50 Ω. Table 3.6 shows the trace widths that are required on the different layers to obtain this impedance. The values result from a field solver algorithm that is provided with the stackup geometry and material parameters like the dielectric constant of the insulator material.

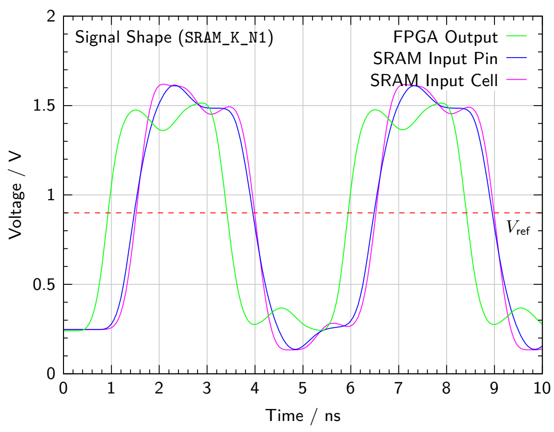
By simulating all high-speed PCB traces after layout and evaluating the worst cases of reflection (Figure 3.11), correct termination and proper layout with respect to impedance continuity is verified.

Differential Signaling

At even higher signal rates and long-distance or board-to-board connections, signal integrity issues become unacceptable for single-ended transmission lines. Differential signaling solves these issues by adding a second line of opposite polarity to each signal and analyzing at the receiver the difference between the two signals. The resulting signal pairs are far less sensitive to noise from distant sources such as crosstalk and power/ground noise, as these sources often affect both signals equally and the ensuing common mode



(a) Data signal driven by SRAM chip



(b) Clock signal driven by FPGA

Figure 3.11: Reflection on SRAM signal traces (from simulation). The selected signals exhibit the worst reflection. In both cases, the signal shape at the receiver is evidently still satisfactory.

noise is filtered by the differential receiver. Reflection noise, however, is still an issue. As the pair of differential signals is closely coupled, impedance discontinuities have to be avoided not only with regard to the single line impedance of the individual traces, but also with respect to the differential impedance between the two lines.

Differential signal pairs can be arranged either horizontally or vertically in the PCB stackup. While vertical alignment on two adjacent planes eases the routing process, manufacturing tolerances in the layer width are generally higher than in the position of traces on a layer. Thus, horizontally aligned (edge-coupled) traces have a more precise matched impedance. The XMU PCB consistently uses edge-coupled layout.

The XMU PCB contains a large number of differential transmission lines, which are matched to a differential impedance of 100 Ω. Table 3.6 shows in the third and fifth column the spacing between the traces that is required to reach the indicated differential impedance.

To achieve close coupling between the differential traces, the spacing between the traces should as a common rule of thumb not be larger than twice the individual trace width. The XMU PCB fulfills this demand.

3.5 The XMU Backplane

The XMU backplane interconnects the TMUs and the SMU of the same segment. It is a custom backplane based on parallel LVDS point-to-point connections that complements the 3U CompactPCI backplane. A block diagram of the XMU backplane is shown in

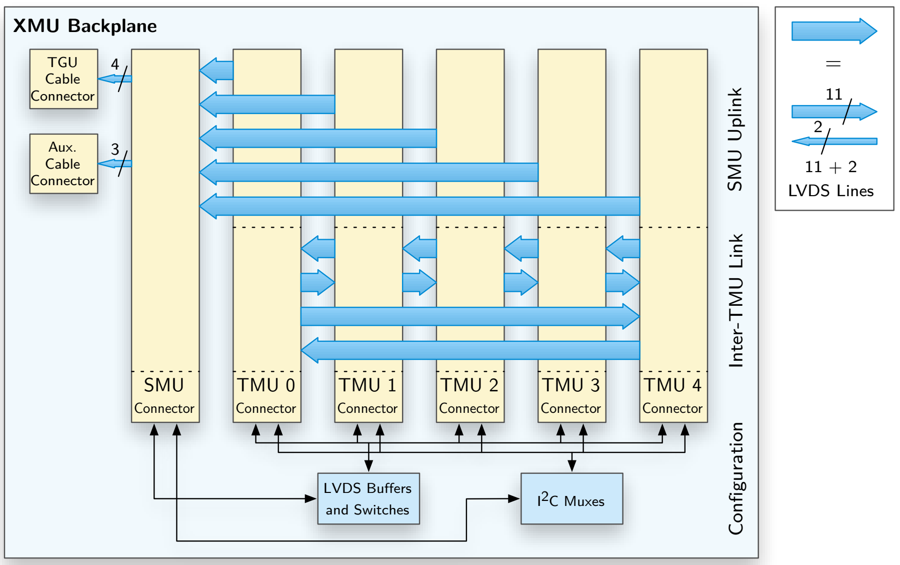


Figure 3.12: Block diagram of the XMU backplane. This backplane complements the CompactPCI backplane with high-speed parallel LVDS point-to-point connections.

Figure 3.12. The backplane is a mainly passive design with few active components for configuration and slow control networks.

The primary objective of this assembly is to provide maximum data throughput from the individual TMUs to the SMU at minimum latency. To this end, the bus width of these SMU uplink connections is maximized to 13 differential signals. The remaining 7 of the SMU's 72 general-purpose LVDS backplane connections are used to transmit trigger information to the TGU and as auxiliary outputs via 8P8C modular connectors.

The TMUs, on the other hand, only need 13 of their general-purpose LVDS backplane connections for the SMU uplink. The remaining signals are used to implement a ring structured communication network between the TMUs. These inter-TMU links can be used to efficiently forward track segment information received in a TMU to the two adjacent TMUs during trigger processing. With data from the surrounding stacks of detector modules, the trigger algorithm can attempt to find tracks across neighboring stacks, which may occur because the TRD's geometry is not perfectly projective.

Finally, the XMU backplane is an essential component of the GTU configuration and monitoring systems, whose hardware aspects are discussed in Chapter 4.

Layout of the XMU backplane requires the same considerations as the XMU PCA with respect to signal integrity. The board is an eight-layer PCB with impedance-matched traces at a differential impedance of 100 Ω. Figure 3.13 shows one of the 18 blue XMU backplane PCAs as used in the GTU setup.

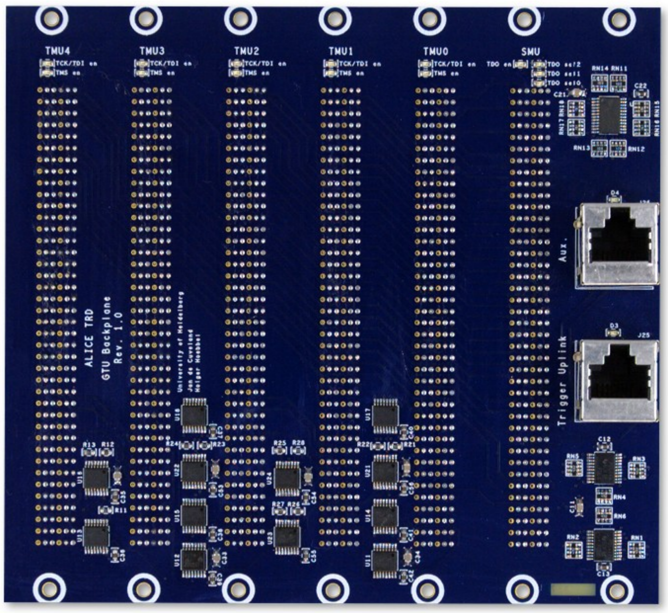


Figure 3.13: Photo of the XMU backplane. The connectors on the right side provide an uplink to the TGU.

3.6 The TGU Backplane

The TGU backplane is a fully passive design. It provides a connection between the 18 separate LVDS cables from the SMUs and the TGU. System management and configuration buses are looped back to the TGU to enable autonomous configuration of the TGU by the TGU's DCS board (see Figure 3.14).

Each SMU is connected via four LVDS signal pairs. Careful consideration of the pin mapping between the modular connectors and the board-to-board connectors ensures that the signals from a single SMU are always routed to the same I/O bank and clock region in the TGU's FPGA. Additionally, the FPGA's input-only LVDS pairs are distributed in such way that one or two of each input's four signals can be configured as outputs serving as a downlink to the SMU, allowing for maximum flexibility with respect to future applications of the GTU system.

Data transmission on the LVDS lines follows a similar concept as for the XMU backplanes of the GTU segments. In contrast to the connection between TMUs and SMUs, this interface can make use of the globally available LHC clock received by the DCS boards' TTC receiver (TTCrx) circuits. As the full four lines are available for data transmission, a bit rate of 8 times the LHC clock (approximately 320 MHz) per line leads to an aggregate bandwidth of 1.28 Gbit/s per segment. The transmitted data words contain the results of the SMU's trigger computations as well as status information like the detector's *busy* state which is available at SMU level and needs to be forwarded to the CTP. The data rate of four bytes per LHC clock cycle allows to transmit precise intermediate results for

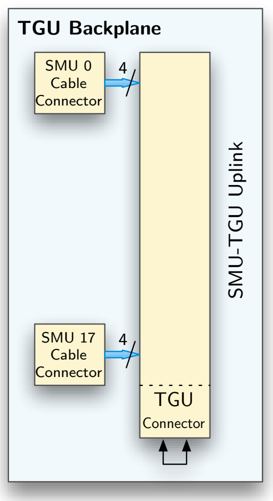


Figure 3.14: The TGU backplane provides a connection between the LVDS cables from the SMUs and the TGU PCA.

final analysis by the TGU according to the trigger scenario.

Figure 3.15 shows the TGU backplane PCA as it is used in the GTU setup. The white board is a specialized four-layer PCB with impedance-matched traces at a differential impedance of 100 Ω.

3.7 The Trigger Generation Unit (TGU) and the CTP Interface

The Trigger Generation Unit (TGU) is the single central point in the GTU system that implements the final trigger decision processing and communication to the experiment's CTP. Its hardware realization is illustrated in Figure 3.16. The PCA is based on the SMU design, using the same PCB layout and major components. Built around a powerful FPGA, the TGU can be configured to perform a variety of fast trigger generation algorithms (cf. Section 8.4).

Via the TGU backplane, the TGU receives trigger and status information from all 18 GTU segments. This high-speed LVDS data transmission utilizes the FPGA's internal input delay cells to shift the phase of the input signals until all internal signals are synchronous to a multiple of the TGU's LHC clock. By using the LHC clock as a common reference, additional resynchronization latency can be avoided.

Instead of the SIU interfaces specific to the SMU, the TGU PCA features a CTP interface connector, which is used to connect to the CTP interface mezzanine card. The CTP interface card is a specific dual-layer PCA that provides five LVDS connections conforming to the parameters demanded by the CTP (see [45] for details). Using its advanced diagnostics features, the card is capable of detecting both non-loaded LVDS outputs and floating LVDS inputs making it possible to constantly monitor the vital connections to the

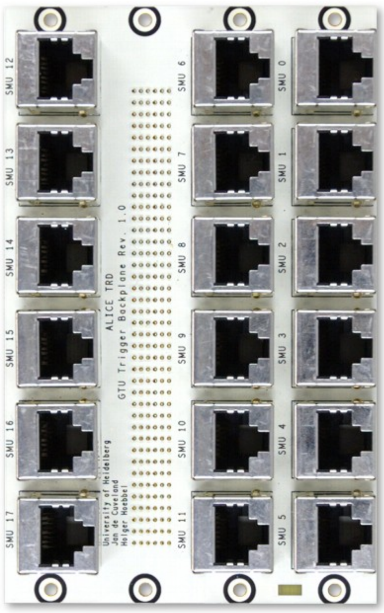


Figure 3.15: Photo of the TGU backplane. The differential signal pairs from the 18 modular connectors on the rear side are routed to a high-speed PCB connector on the front side.

Port	Direction	Description
0	Output	TRD/GTU busy
1	Output	TRD L1 trigger contrib.
2	Output	Additional TRD trigger contrib.
3	Input	Unused
4	Input	L0 input from CTP

Table 3.7: List of LVDS signals between GTU and CTP

CTP. All available ports can be configured individually as either output or input. The default configuration is shown in Table 3.7. Essential signals include the TRD/GTU busy and the L1 trigger output. In addition to the trigger computation logic, the FPGA design contains pattern generators and diagnostics to confirm error-free communication with the CTP system.

3.8 System Integration and Operating Environment

The GTU hardware is mounted to 6U CompactPCI subracks, each housing the modules of two GTU supermodule segments. The layout of these crates is displayed in Figure 3.17. Together with supplementary devices such as remote-controlled power supplies, DCS Ethernet switches, fiber-optical patch panels and cooling devices, the nine GTU

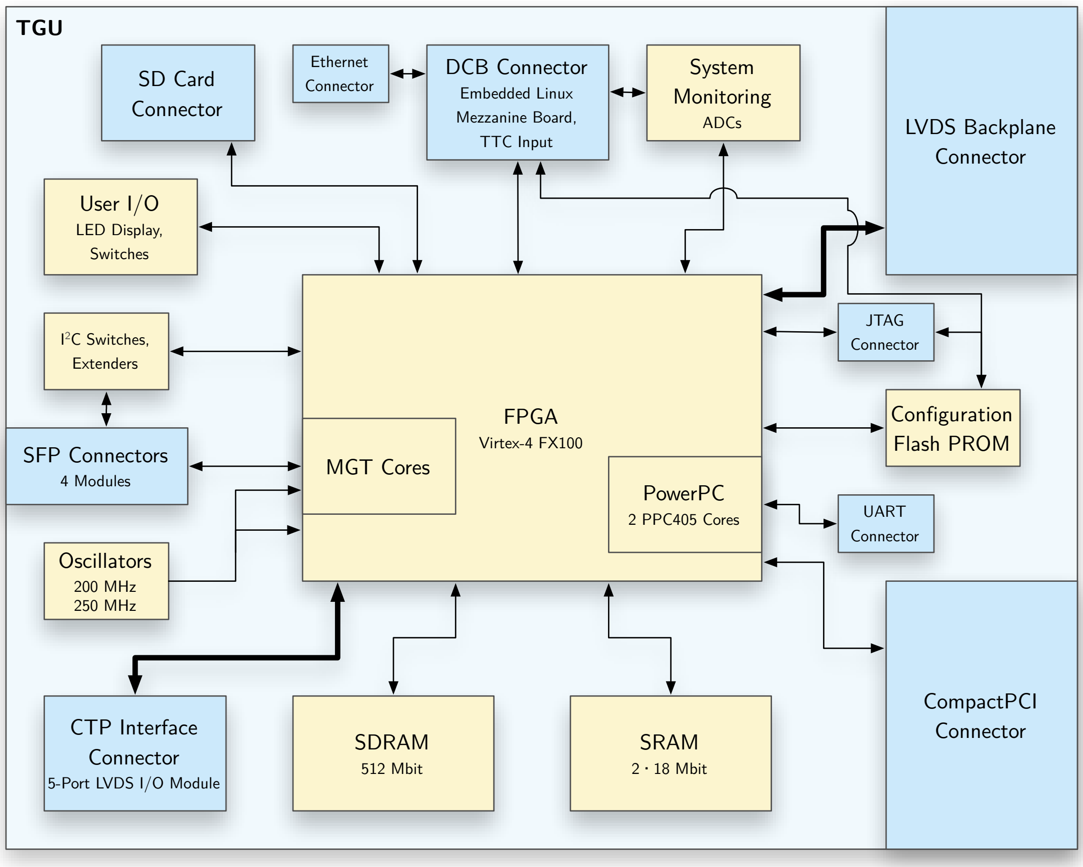


Figure 3.16: Block diagram of the Trigger Generation Unit (TGU). Bold lines highlight the most important data path.

crates are placed in standard 19" racks. The complete GTU system is installed in the experiment cavern UX-25 at CERN LHC. Figure 3.18 illustrates the final GTU rack layout. A picture of the setup is shown in Appendix A.

In a high-energy physics experiment, the special properties of the operating environment have to be considered for all electronic equipment. Apart from general demands of an industrial environment, the undesired effects of ionizing radiation and strong magnetic fields require special attention. In addition, restricted access to the experiment setup and limited periods of accelerator run-time require a strategy to deal with potential hardware failures.

3.8.1 Fault Tolerance Strategy

During experiment runs, radiation levels do not allow persons to enter the area where the ALICE detectors are located. Given the limited amount of run-time every year, especially in the heavy ion collider LHC mode, downtime should be kept to a minimum for all

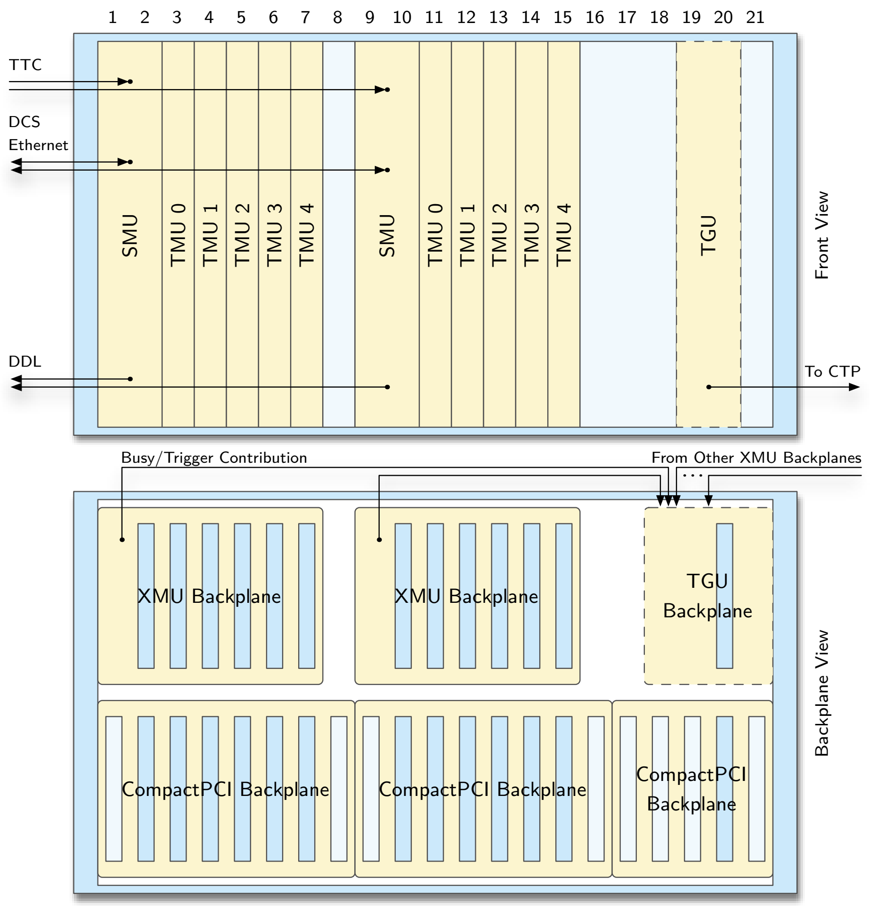


Figure 3.17: Final GTU crate layout. A crate comprises two GTU segments, the specific hardware for two TRD supermodules. Nine of these crates form the complete GTU setup. The TGU and its backplane are only present in the center one.

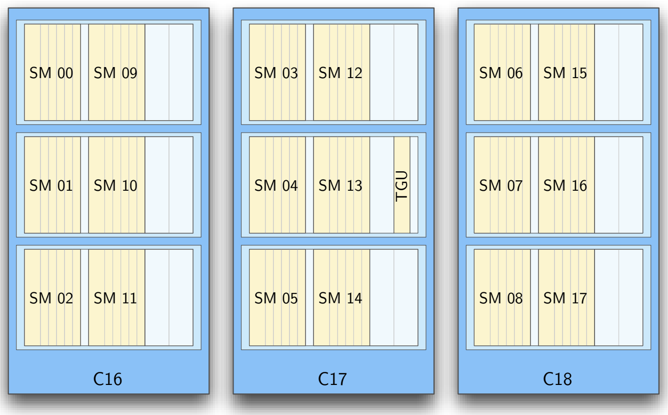


Figure 3.18: Final GTU rack layout. The GTU setup with its 18 supermodule segments is installed in racks C16–C18 in the experiment cavern UX-25 at CERN LHC.

detector systems. In most cases, data from the TRD remains useful for physics analysis even if data from a small fraction of detector modules is missing. The key objective is to prevent a full system failure.

There are two general levels of accessibility: detector components inside the L3 magnet may not be accessible for several consecutive months, while modules outside the L3 magnet can be accessed in a matter of hours or days if the need for immediate maintenance arises (subject to the respective run policy).

The TRD strategy concerning fault tolerance is adjusted according to the level of accessibility. Non-essential systems, i. e., systems whose failure does not cause a complete TRD downtime, are not designed redundantly. Essential systems, i. e., systems that can act as a single point of failure, should be redundant if they are located inside the L3 magnet. Otherwise, they should be easily replaceable. This strategy is consistent with other ALICE detectors and control systems.

The GTU fully complies with this strategy. It can be configured remotely to accept any combination of inoperative detector modules, disregarding the respective optical links. As the SMUs and the TGU can be configured to ignore the uplink from individual TMUs or SMUs, even a failure of GTU components up to a complete GTU segment can be tolerated. The TGU itself and its power supply constitute the only potential single point of failure. However, a redundant design of the TGU and its trigger output would not cause much benefit, as the CTP's trigger input at the other end and the complete CTP system are not designed for redundancy.

All active GTU components are backed up with identical spare parts. The modular design of the GTU combined with the easy access to the crate components assures minimum downtime attributable to hardware failures.

3.8.2 Ionizing Radiation Tolerance

In case of the TRD GTU, the effects of ionizing radiation are considered negligible. The total radiation dose at the GTU location is too low to cause any permanent damage, and even single event upsets are highly unlikely. Even though, if any persisting error in the GTU configuration occurred, it could be handled remotely by reconfiguring the respective subsystem with minimum downtime because of the nature of the GTU configuration system.

3.8.3 Magnetic Field Tolerance

The GTU system is positioned in the UX-25 cavern at rack locations C16–C18. At this site, there is a magnetic stray field primarily from the ALICE muon system dipole magnet, but also from the L3 solenoid magnet. The magnetic field intensity has been measured with both magnets active [76]. While there are no precise measurements at the GTU rack locations, the field intensity can be estimated from the surrounding measurements to

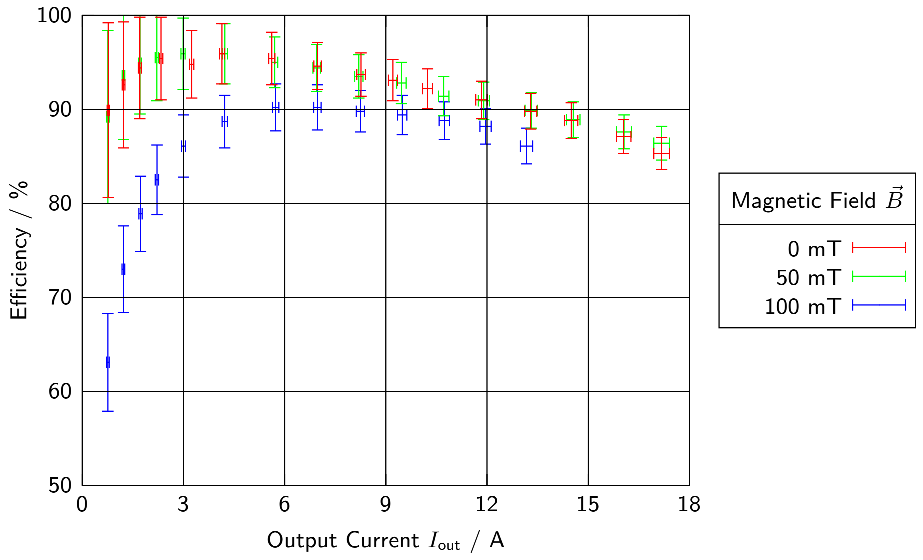


Figure 3.19: Voltage regulator efficiency vs. load current for different values of the surrounding magnetic field. The utilized power module PTH05010W by Texas Instruments is unaffected by a magnetic stray field for values of at least up to 50 mT.

approximately 5 mT. The maximum value measured in any of the C-type locations below the dipole magnet amounts to 20 mT, which can be regarded as an upper limit to the relevant field intensity.

The magnetic stray field can affect all components that incorporate inductors. This includes electronic filters and transformers as well as electromechanical devices such as ventilator motors. The sensitivity of different devices varies considerably. A summary of tolerance measurements for some relevant components can be found in [44]. While the standard rack turbine employed in most ALICE electronics racks tolerates fields of up to 180 mT, a conventional fan tray ceases to operate at 8 mT. The cooling concept of the GTU system therefore relies on the rack turbine in combination with water cooling only. High-efficiency passive heat sinks are utilized to prevent overheating of critical components such as the large FPGAs.

To verify the magnetic field tolerance of the electronics system, it is necessary to test the response to a surrounding magnetic field. Instead of assembling and testing the complete system in an adjustable field, a practical approach is to identify potentially critical parts and to test these components individually. In case of the GTU, the only potentially critical components are the SFP modules that include filters and the power modules based on switching voltage regulators. Tests performed with a large neodymium permanent magnet indicate that at intensities of approximately 100 mT, there is no single-bit error in 4 hours of operation, corresponding to a bit error rate (BER) of $< 10^{-13}$.

Results from tests with the power module PTH05010W by Texas Instruments are summarized in Figure 3.19. For low values of the magnetic field intensity, the measured efficiency $P_{\text{out}}/P_{\text{in}}$ complies with the module's specifications [77]. At $\vec{B} = 50\,\text{mT}$, operation of the circuit is still unaffected. Applying a magnetic field of $\vec{B} = 100\,\mathrm{mT}$, the efficiency decreases significantly. In this case, the regulator output voltage collapses at an output current of 13 A. Up to this limit, the regulator output voltage is kept constant. The results show that all GTU components can operate in a constant magnetic field of at least up to $\vec{B} = 50\,\mathrm{mT}$. This is a factor 10 of the expected field intensity, assuring reliable operation in the ALICE environment.

4 System Monitoring and Control

For a complex trigger and readout system like the GTU in an environment of ever-changing experimental objectives and requirements, intelligent system management is a key issue. Important characteristics of an appropriate monitoring and control system include remote accessibility, durability (the ability to report unexpected conditions and to recover from them), comprehensiveness and extensibility. In an environment such as ALICE, where the system is not physically accessible for prolonged periods of time, it is particularly important that the system provides means of unrestricted remote access to all system parameters.

Besides the unrestricted raw access to all aspects of the system status and configuration, it is necessary to condense the available information to a level that is comprehensible for a non-expert operator. To allow automated control in conjunction with other subsystems, the GTU, like all ALICE detector systems, is integrated into the general ALICE Detector Control System (DCS). This chapter outlines the hardware and software architecture of the GTU monitoring and control system.

On hardware level, all remote monitoring and control access to the GTU is conducted by the employed DCS boards. The DCS board (introduced in Section 3.3, see also [34]) is an FPGA-based embedded GNU/Linux system, which is also used by the TRD front-end modules and several other ALICE detectors. Using the programmable logic resources of the Altera Excalibur EPXA1 FPGA, the ARM922T processor core can be complemented by custom interfaces. The specific DCS board firmware enhancements for the GTU are documented in the diploma theses [47] and [69].

Three separate slow-control bus systems serving different purposes connect the DCS boards to the GTU hardware:

1. The **low-level system configuration** bus provides direct access to the FPGA configurations.
2. The **system health monitoring** (SysMon) bus independently collects diagnostics information concerning supply voltages and temperatures.
3. The **high-level configuration and control** bus based on the GTUcom protocol provides a flexible means of gaining access to run-time status and configuration parameters.

The architecture of the three bus systems is presented in the following sections. The remaining sections of this chapter focus on the software aspects of the monitoring and control system.

4.1 Low-level System Configuration

The most versatile means of accessing the FPGA configuration at a low level involves the incorporated IEEE 1149.1 interface. The IEEE 1149.1 *Standard Test Access Port and Boundary-Scan Architecture* [41] is commonly referred to as JTAG[1]. The JTAG standard describes a four-wire serial bus that can accommodate multiple devices in a daisy chain. Via this interface[2], the FPGA can be reprogrammed partially or at full, the configuration can be read back, and internal monitoring and debugging cores can be accessed. Remote access to this interface is considered recommendable.

However, writing each FPGA's configuration remotely at every system start-up would be opposed to the idea of the GTU as a semi-autonomous system. Both TMU and SMU therefore comprise flash-based configuration PROMs that store the configuration bit stream of the FX100 FPGA and automatically initialize the FPGA during system start-up via a dedicated serial connection. The PROMs themselves are in turn configured via a JTAG interface (see the *Virtex-4 FPGA Configuration User Guide* [86] for details on the different configuration options for Virtex-4 FPGAs).

All changes in configuration are initiated by the DCS board mounted to the SMU, which implements the connection to the DCS system. Each DCS board controls a total of 12 JTAG components. Additional connectors on the XMU PCBs allow to locally connect to the JTAG bus, bypassing the remote configuration system. During firmware development in the laboratory, these ports can be used to gain access to the JTAG chain for development tools such as the real-time verification software Chipscope Pro by Xilinx or the Xilinx Microprocessor Debugger (XMD).

JTAG conventionally uses single-ended signals. Nevertheless, JTAG signals can exceed bit frequencies of 30 MHz, and because of the fast rise/fall time of the utilized driver cells, effective signal frequencies are notably higher. Consequently, signal integrity is to be considered, especially if traversing multiple PCBs. The backplane connectors, however, are optimized for differential signals. Using signal pin pairs for independent single-ended signals can cause excessive crosstalk noise with fast signals. The apparent solution, which is employed by the GTU, is to transmit JTAG signals as differential pairs according to the LVDS standard.

When designing a JTAG-based configuration system for multiple components, a fundamental decision is whether to arrange all components in a single daisy chain or to provide separate JTAG buses for selected or all components. A single daisy chain has the advantage of lower demands on I/O pin and PCB trace resources. Separate buses, on the other hand, provide additional fault tolerance and greater flexibility. In the GTU system, both advantages are required, as only a single JTAG connection can be accommodated on the SMU's backplane connector, while the basic configurability of a GTU segment

[1] JTAG is an acronym for the Joint Test Action Group, the technical subcommittee initially responsible for developing the standard.

[2] Virtex-4 devices support the new IEEE 1532 *Standard for In-System Configuration (ISC) of Programmable Devices* [42], based on the IEEE 1149.1 standard.

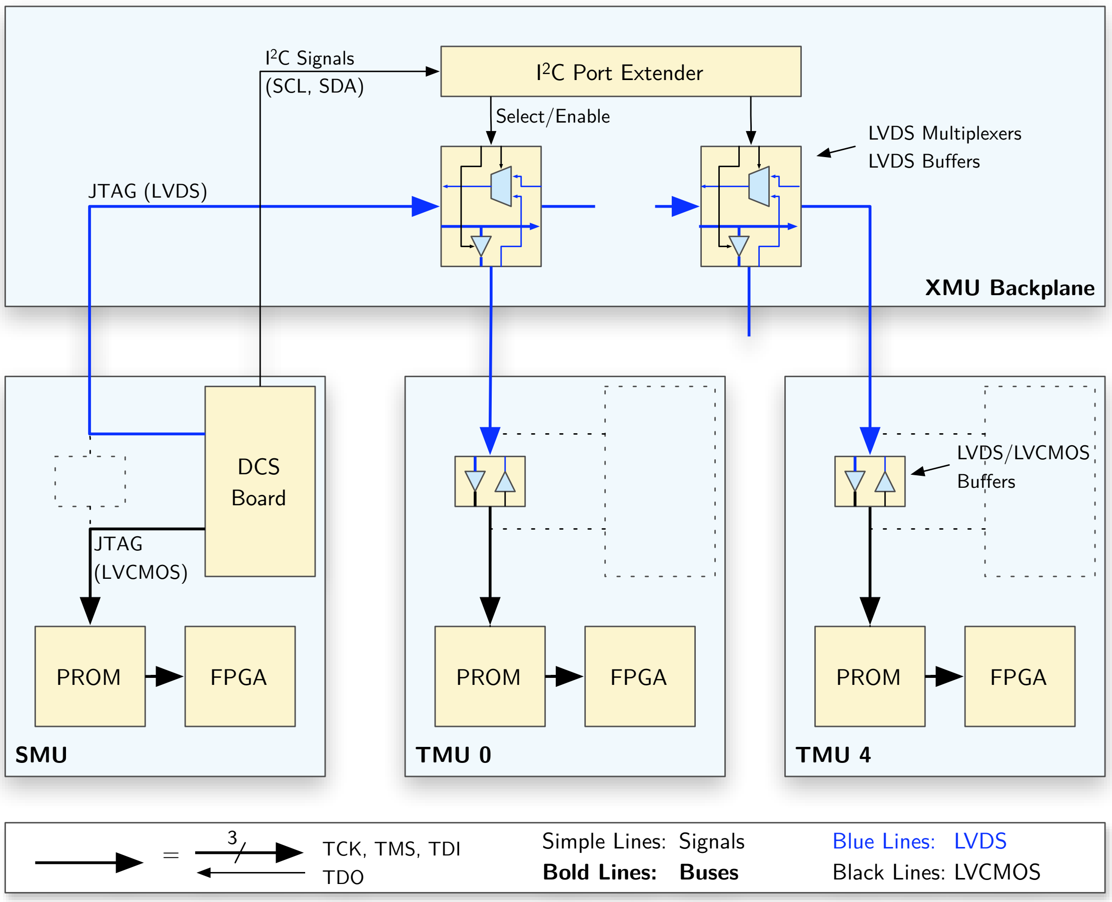


Figure 4.1: The low-level configuration system for one GTU segment. Each PCA implements a separate JTAG chain, which can be accessed selectively by the DCS board.

should evidently not depend on proper installation and functioning of all respective TMU modules.

The solution to this conflict of interests is presented in the GTU configuration system outlined in Figure 4.1. While the SMU components are in a separate JTAG chain directly controlled by the DCS board, a network of active LVDS buffers and multiplexers on the XMU backplane allows to dynamically configure the JTAG chain containing the other devices. LVDS buffers selectively distribute the TCK, TMS, and TDI signals of the JTAG standard to five separate JTAG chains on the five TMUs. Using a tree of multiplexers, the DCS board selects one of the returning TDO signals for reception.

This setup offers several advantages, as all TMU PCAs can be configured independently. If any number of TMUs is missing or malfunctioning, configuration access to all other TMUs is unaffected. As a characteristic of JTAG, the exact bit stream required to configure a device depends on the position of that device in the JTAG chain. In this setup, a minimum number of different configuration files is required, as all chains are equivalent. Finally, by enabling specific LVDS buffers, the DCS board can not only configure each

TMU individually without affecting the others devices, but may also choose to configure several or all TMUs simultaneously. In this case, only one of the TDO signals can be checked. But using the checksum read-back command supported by both PROM and FPGA, correct transmission of the configuration bit stream to the other devices can be verified afterwards with minimum overhead.

The select and enable signals for the LVDS multiplexers and buffers are set by an I^2C port extender circuit, which is controlled via two signals from the DCS board. By mounting either the DCS board or a set of LVDS/LVCMOS driver circuits on the XMU PCB, routing and direction of the JTAG signals are selected. Therefore, SMU and TMU can share exactly the same PCB despite the entirely different roles in the GTU configuration system.

On the DCS board, the FPGA design contains two instances of a JTAG slave module for the on-chip Avalon bus. The JTAG module is complemented by a specific Linux device driver that provides application-level access to the interface. A player application running on the embedded GNU/Linux system reads configuration files in the Xilinx Serial Vector Format (XSVF) format, which contain sequences of high-level IEEE 1149.1 bus operations, and executes these operations. The XSVF file is generated from the device configuration file and the boundary scan description files of the devices in the chain. It contains all necessary information to program the devices and verify the success of the operation.

The enhancements to the DCS board firmware and the necessary adaptations of the player application are documented in [47, p. 35–39]. An XSVF file of 4.2 MB is written to the FPGA in approximately 28 seconds, and the programming speed is limited only by the DCS board's CPU performance. After power-up, the configuration saved in the PROM is transferred to the FPGA in less than two seconds.

Using the configuration system presented, all components of the GTU system can be configured remotely in a convenient and reliable way during development and in case of firmware upgrades, while no external configuration is necessary during regular operation.

4.2 System Health Monitoring

For remotely operated systems, automated health monitoring is an important issue. Given the planned operation time of more than ten years, sporadic hardware failures are to be expected. Malfunctioning generally requires operator intervention. If the cause of the problem is known, the system might be able to rejoin the run in a reduced configuration.

Health monitoring also concerns the safety of operation. The cooling system, for example, depends on constant operation of the rack turbine and cooling water flow in the rack's heat exchanger. In case of unnoticed failure, the GTU's FPGAs would quickly overheat causing permanent damage. While a rapid increase in system temperature will cause an automatic interlock mechanism to interrupt the power supply, all variations are available to the DCS. The GTU system health variables in the DCS are accessible by the operator and can assist in diagnosing the cause for unexpected behavior.

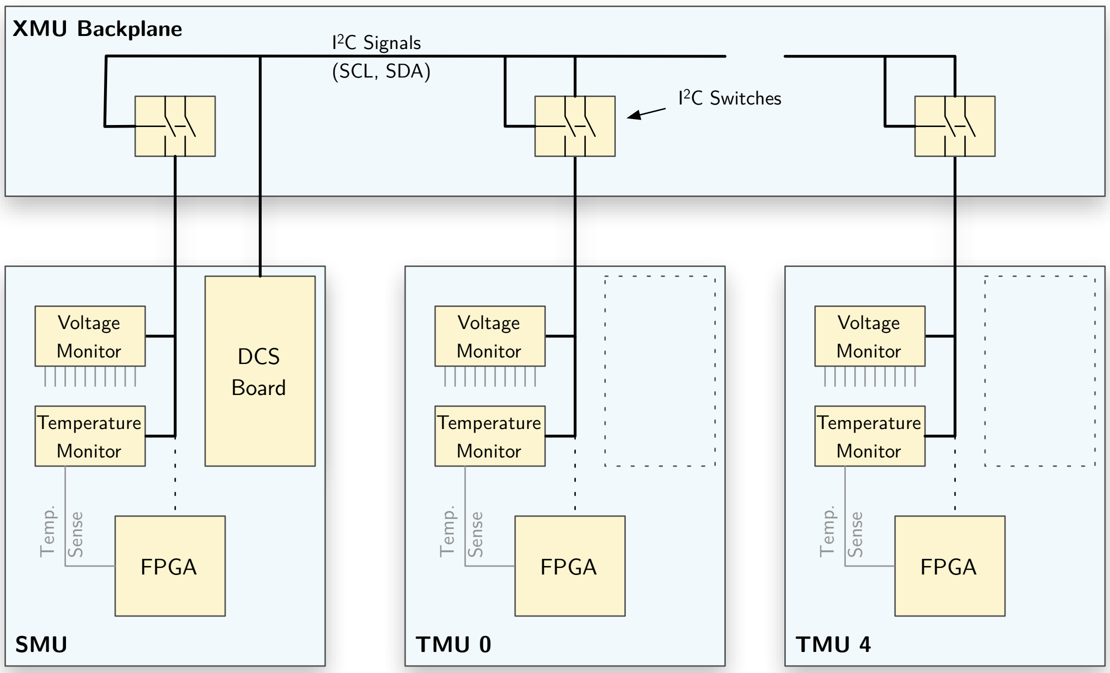


Figure 4.2: The system management bus for one GTU segment. The DCS board selectively accesses the voltage and temperature monitor circuits on the XMU PCAs.

System Health Monitoring is the domain of the second management bus system originating at the DCS board. Communication on this bus is based on the I^2C/SMBus format with sensor devices on all XMU PCBs. Figure 4.2 illustrates the employed bus architecture. In contrast to the other two management buses, this bus is designed to be independent from the individual XMU FPGA, making it possible to detect a wide range of errors, including voltage regulator failures which may cause FPGA malfunction.

For optimum reliability in case of partial power supply failure, the monitoring itself is dependent only on the 3.3 V supply voltage, which is directly provided by an external power supply unit and drives all monitoring circuits and the DCS board. I^2C switches on the XMU backplane allow the DCS board to selectively access the monitor integrated circuits on the different PCAs in a GTU segment, even though they share the same bus address, as they are employed on identical PCAs. As the switches themselves are controlled via I^2C commands, a single two-wire bus from the DCS board is sufficient to monitor a complete GTU segment.

The parameters monitored comprise all relevant XMU supply voltages as listed in Table 4.1 and the temperature at two different positions on each PCA. The first temperature reading reflects the PCB temperature, and the second reading indicates the temperature inside the FPGA. An internal thermal diode is utilized to enable a precise measurement of the die temperature.

Channel	Net name	Voltage / V	Divider
0	`VCC_25`	2.5	2
1	`VCC_18`	1.8	1
2	`VCC_12`	1.2	1
3	`VCC_09`	0.9	1
4	`AVCC_25`	2.5	2
5	`AVCC_15`	1.5	1
6	`AVCCA_12`	1.2	1
7	`AVCCB_12`	1.2	1
8	`VCC_50`	5.0	3
9	`VCC_33`	3.3	2

Table 4.1: On-board supply voltages monitored by the I^2C voltage monitor. The 8 bit ADC utilizes an internally generated reference voltage of $V_{\mathrm{ref}} = 2.048\,\mathrm{V}$. External voltage dividers map the higher supply voltages to the admissible sensing range.

4.3 High-level Configuration and Control

During regular operation of the GTU, some configuration parameters need to be adjusted infrequently. These include values regarding the fundamental running scenario, such as the selection of trigger criteria according to current experimental objectives as well as basic variables referring to the state of the TRD setup, such as the number and position of inoperative TRD modules. It is also desirable to provide the TRD operator with a status display of information concerning the system health and statistical information about data and trigger rates as elementary indicators of the quality of data. Finally, during refinement of the GTU firmware and implementation of future trigger algorithms, the developer needs easy and unobstructed remote access to numerous internal registers and memory areas.

While the JTAG configuration system described in Section 4.1 provides low-level access to the FPGA design, the GTU's high-level configuration and control network serves these three purposes. It consists of several layers. This section focuses on the hardware layer, which provides the physical means of communication between the DCS Board and the FPGAs in the associated GTU segment.

As outlined above, I/O pin resources are particularly limited at the DCS board and the backplane connectors. However, as a slow control system, the interface requires only modest data rates. Under these circumstances, a serial bus-based system is recommended. The GTU implements this system by applying a UART-based multidrop bus[3]. The advantages of using a universal asynchronous receiver/transmitter (UART) protocol are the convenient extensibility to a multidrop bus with a single tristate return line, the availability of compact UART components for all involved FPGAs, and the option of easily connecting to PC EIA-232 interfaces for development.

[3]UART communication parameters can be specified at firmware level. Current settings are: 56 700 bit/s, 8N1.

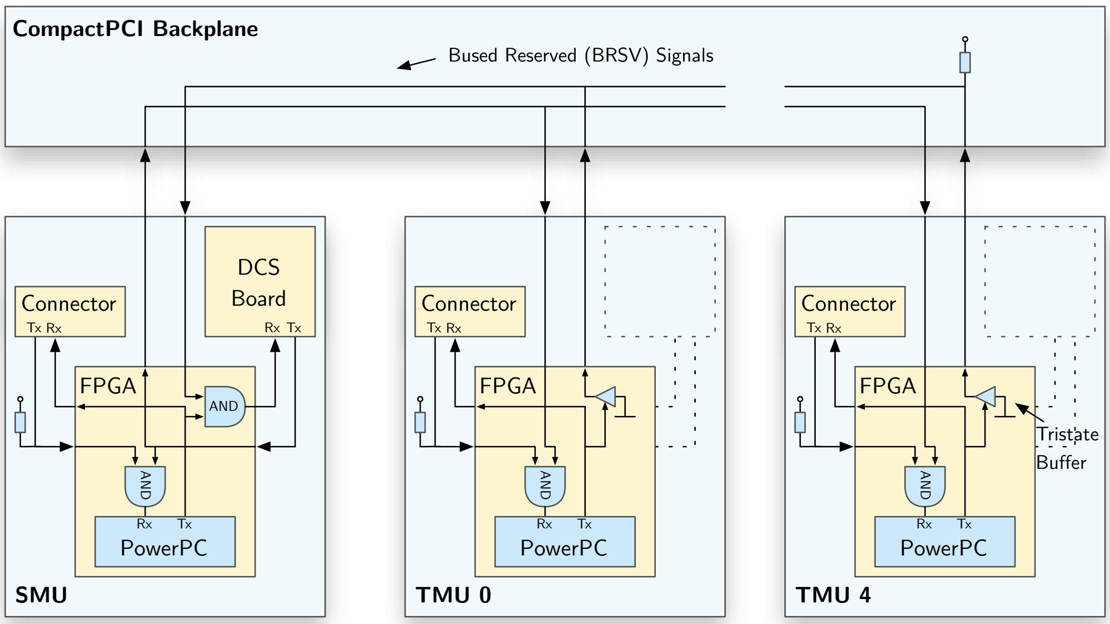


Figure 4.3: The high-level configuration and control network for one GTU segment. The active-low UART-based multidrop bus connects all FPGAs and the DCS board using bused signals on the CompactPCI backplane.

Figure 4.3 shows the monitoring and control network structure of a GTU Segment. The DCS board on the SMU PCA has access to all FPGAs via the multi-drop bus routed on the CompactPCI backplane, while the UART connectors on the PCAs allow local access independent of the DCS board.

A specific protocol named *GTUcom* is used to arbitrate the individual FPGAs on the bus. Device addressing is based on the CompactPCI geographic address (GA) feature, which provides every device on the bus with a unique number corresponding to its position on the backplane. Because of the bus architecture, communication on the slow control network is not compromised by missing or unconfigured TMU FPGAs. Using only few resources, the outlined bus architecture provides the GTU with a robust and flexible slow control system.

4.3.1 The Node Control Processors

In contrast to the common solution of mapping configuration registers to addresses on a dedicated configuration bus and to directly attach this bus to the configuration interface, the GTU employs a higher-level technique. In the FPGA design, all configuration and status registers are connected to the Processor Local Bus (PLB) of one of the PowerPC processors, as is the device's UART interface.

All SMU, TMU, and TGU modules use the same PowerPC environment, which is presented in Figure 4.4. The processor core is supplemented with a local bus structure implemented

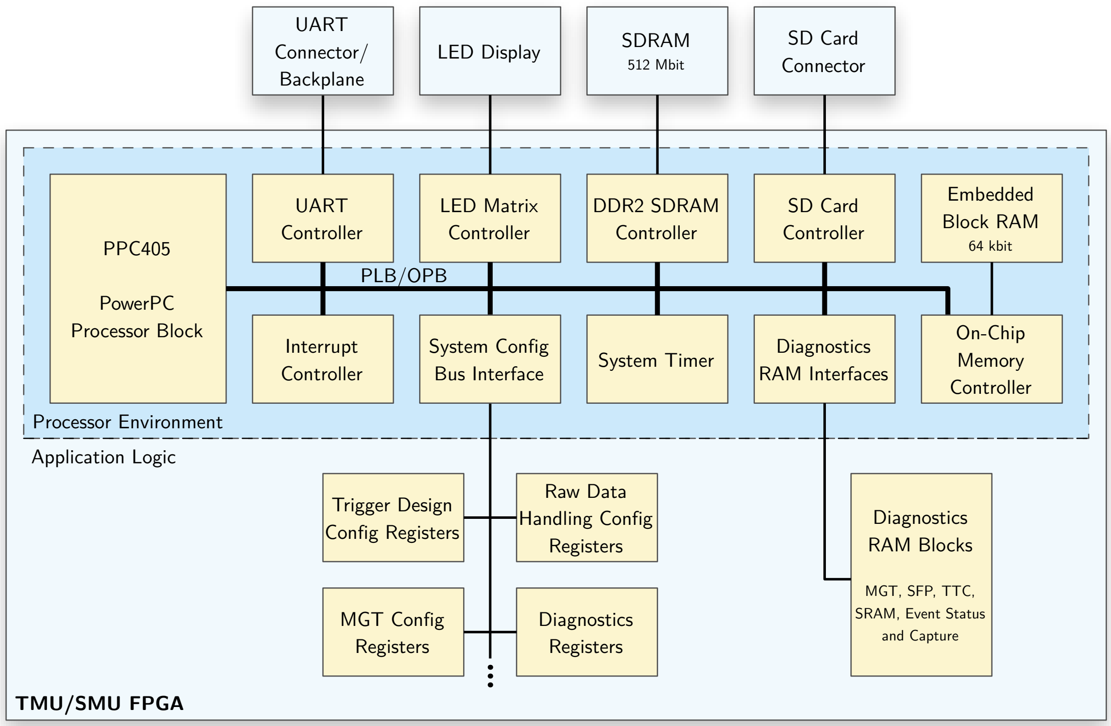

UART Connector/ Backplane
LED Display
SDRAM 512 Mbit
SD Card Connector
PPC405 PowerPC Processor Block
UART Controller
LED Matrix Controller
DDR2 SDRAM Controller
SD Card Controller
Embedded Block RAM 64 kbit
PLB/OPB
Interrupt Controller
System Config Bus Interface
System Timer
Diagnostics RAM Interfaces
On-Chip Memory Controller
Processor Environment
Application Logic
Trigger Design Config Registers
Raw Data Handling Config Registers
Diagnostics RAM Blocks
MGT, SFP, TTC, SRAM, Event Status and Capture
MGT Config Registers
Diagnostics Registers
TMU/SMU FPGA

Figure 4.4: The XMU embedded PowerPC processor environment. The processor core is supplemented with a local bus structure implemented in the FPGA fabric and interfaces to configuration registers and memory blocks in the application logic.

in the FPGA fabric. Peripheral modules are connected to the core through a PLB or On-Chip Peripheral Bus (OPB) interface. These modules include essential processor peripherals like system timer and interrupt controller, memory controllers for both on-chip and external RAM resources, communication interface modules, and dedicated interfaces to configuration registers and memory blocks in the application logic.

The number and purpose of the configuration registers in the application logic differs between SMU, TMU, and TGU designs. In the SMU, most registers deal with primary trigger handling, data formatting and communication with the DAQ system. Configuration in the TMU focuses on trigger computation parameters and variables regarding raw data handling. In the TGU, configuration parameters include the selection of trigger conditions and settings of the output interface to the CTP. Diagnostics RAM blocks are employed where the amount of data exceeds the reasonable limits of status registers, as is the case with trigger and data logging facilities.

4.3.2 The GTUcom Node Control Application

Each node control processor executes a node control application written in the C language, which responds to requests received via the UART interface. The format of the requests is a line-oriented syntax with commands supporting any number of options. This approach provides appreciable adaptability. While simple commands provide fundamental read/write access to all registers and memories, advanced commands allow for more complex actions such as the analysis and interpretation of the contents of larger memory blocks and unit self tests. As only the results have to be transmitted through the slow control link, profound checks and operations can be performed that would otherwise be impracticable. The node control software running on the embedded PowerPC also carries out an automated initialization sequence at system start-up. The application has a modular structure and is fully integrated into the FPGA design synthesis flow. This allows new commands to be added easily and consistently as the FPGA VHDL design is further refined.

4.4 The Detector Control System (DCS)

The ALICE Detector Control System (DCS) is a coherent hierarchical control system for all sub-detectors. It relies on a distributed commercial PVSS II[4] SCADA system complemented with the SMI++[5] framework for modeling the experiment behavior by finite state machines (FSMs) [26]. Both tools are integrated into the experiment independent Joint Controls Project (JCOP) framework, which intends to support the common needs of all LHC experiments by providing an integrated set of tools and guidelines [43] that ease the development of control system applications. An introduction to the JCOP Framework is presented in the *Joint PVSS and JCOP Framework Course* Manuscript [66].

The DCS itself is controlled by the ALICE Experiment Control System (ECS). Figure 4.5 illustrates the hierarchy of controls. The GTU is represented by a top node, corresponding to the GTU worker node computer, and 19 secondary nodes corresponding to the 19 GTU DCS boards.

4.4.1 GTU Monitoring and Control Software

The software structure of the GTU monitoring and control system is summarized in Figure 4.6. The GTU worker node is a dedicated computer running an instance of the SCADA application and an interface to the FSM framework. Via Ethernet TCP/IP connections, the GTU worker node communicates with the 19 GTU DCS boards, using the DIM system[6] as communications layer.

[4]PVSS II is a commercial supervisory control and data acquisition (SCADA) tool from ETM. It is the CERN-recommended SCADA solution.

[5]SMI++ is a multi-platform framework for distributed control systems developed at CERN (see the SMI++ project web site [28] for reference).

[6]The Distributed Information Management (DIM) system is a communication system for distributed environments, which has been developed at CERN. See [32, 31] for reference.

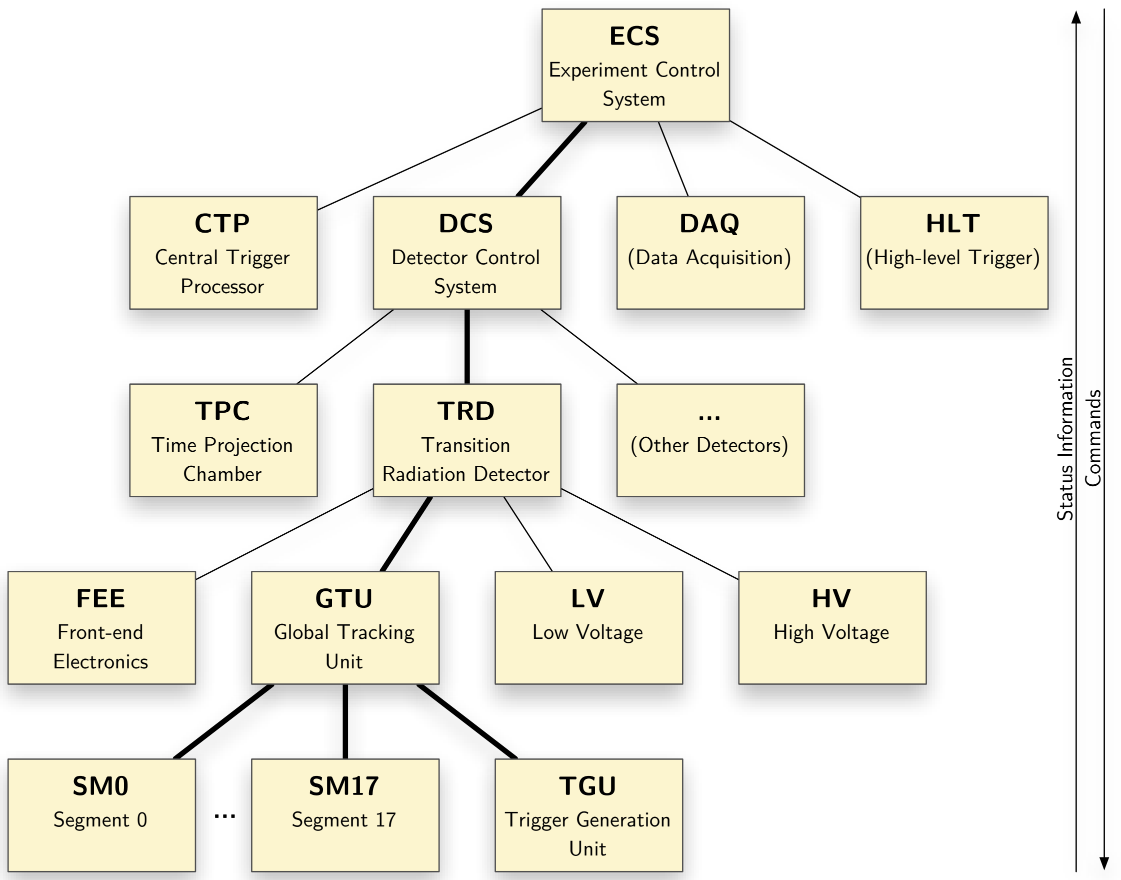


Figure 4.5: The GTU in the controls hierarchy of ALICE. Commands issued by the experiment control system are distributed to the subsystems, while relevant status information from the subsystems is transmitted to the higher-level nodes. Each level of hierarchy represents a layer of abstraction.

Each of the DCS boards executes a GTU control application, which implements a DIM server providing an abstract interface to the GTU configuration. Via a high-level application programming interface (API) and custom utility applications, the GTU control application accesses custom device driver modules. These kernel modules provide access to specific hardware components synthesized into the DCS board's FPGA, which constitute the interfaces to the three different monitoring and control buses. In case of the high-level GTUcom bus, the node control software running on the GTU's FPGAs provides an additional layer of abstraction.

The GTU Control Application The GTU control application registers with a name server and publishes data points according to the DIM protocol. Command channels allow to initiate changes in configuration. While status information from the XMU FPGAs in a compressed format and system health information from the monitoring sensors is requested at regular intervals by the control application, only changes are distributed to

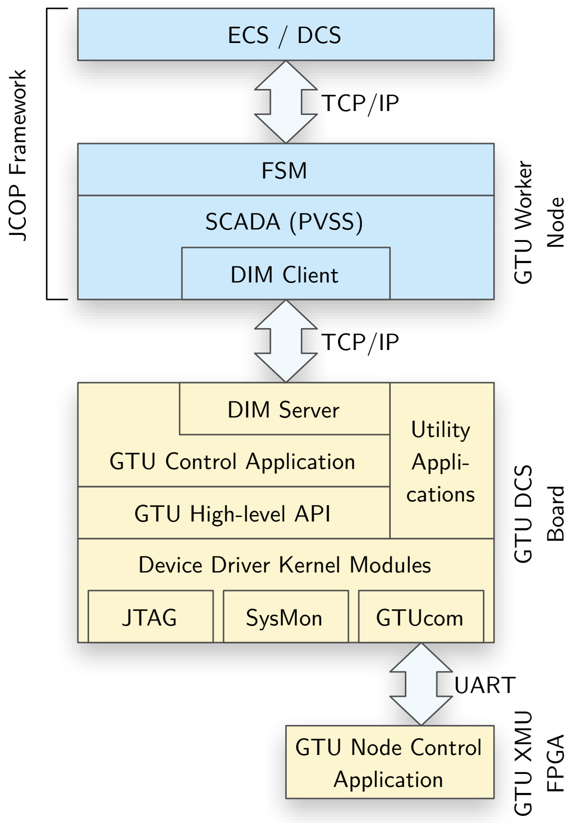
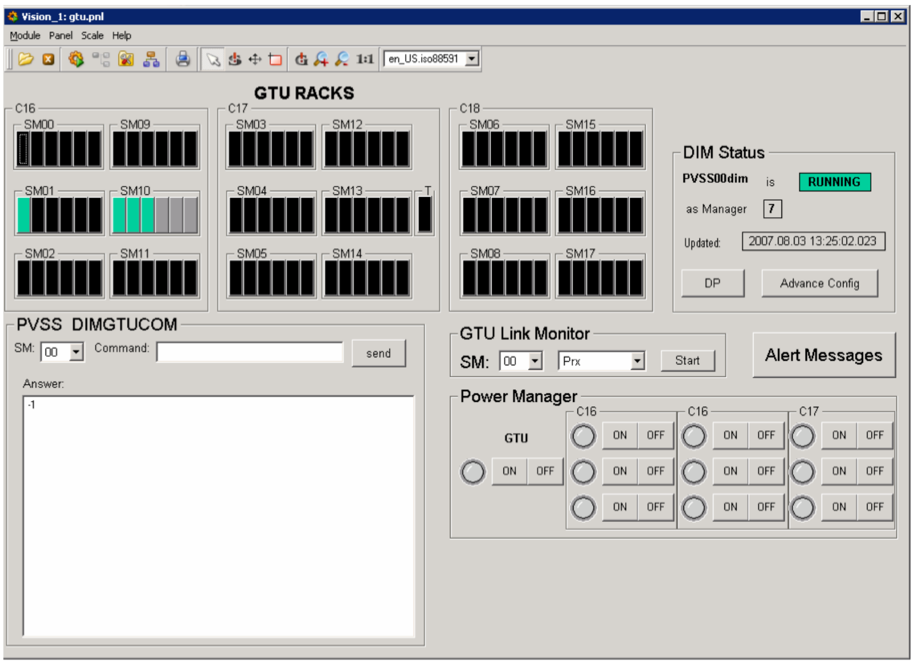


Figure 4.6: The GTU monitoring and control software structure. Several layers of software on the DCS board encapsulate access to the GTU hardware bus systems and provide an interface to the higher levels of the control system.

the subscribed clients via the DIM update mechanism. Configuration requests via DIM command channels are either relayed to the respective FPGAs via the GTUcom system, or, in case of low-level FPGA or PROM configuration changes, converted to JTAG command sequences by an automatically invoked utility application.

The GTU control application is, by design, a multithreaded application, allowing any number of DIM clients to connect simultaneously. Careful locking mechanisms are required to ensure proper arbitration of the different hardware access requests. Together with the high-level API, which encapsulates the access to the Linux device driver kernel modules, the control application provides a solid layer of abstraction from the underlying hardware and the three local monitoring and control bus systems.

While the system is to be controlled by the FSM framework during ordinary running, the operator can obtain high-level status information and control major run parameters using the interface provided by the SCADA system. An example of a high-level PVSS II control panel is presented in Figure 4.7. Prototype versions of the fundamental PowerPC processor environment, of the GTU control application with the underlying GTUcom device driver, and of the SCADA controls are documented in [69].



PVSS00dim is RUNNING

as Manager 7

Updated: 2007.08.03 13:25:02.023

SM: 00 Prx

Answer: -1

Figure 4.7: Prototype SCADA control panel for the GTU. The graphical user interface provides the shift operator with a quick overview of the system status. More detailed information is available in additional control panels linked from the main panel.

4.5 A High-speed Linux-based Control Network

The GTU node control application, which is executed locally on the distributed PowerPC processors and handles incoming monitoring and control requests (cf. Section 4.3.2), is designed to be able to run with or without underlying operating system. Running the GTU nodes without operating system has several advantages: rapid initial system start, reliable real-time performance, straightforward configuration, and reduced overall system complexity.

For some applications of the GTU, however, it is beneficial to have the PowerPC processors execute an operating system such as GNU/Linux. To explicitly allow for this setup, the XMUs provide external SDRAM components to serve as the systems main memory and SD card interfaces as a means of mass storage for the Linux kernel and an associated file system. The operating system provides multitasking capabilities and extends the monitoring and diagnostics facilities by supporting a large set of standard tools. Furthermore, the operating system can enable high-speed configuration access to the FPGA design by using one of the SMU's SFP transceiver modules as optical gigabit Ethernet interface to the embedded system.

In this case, the DCS Ethernet connection is routed to the five TMUs in a GTU segment by the SMU's embedded system, as only the SMU possesses extra SFP module slots to support an additional high-speed network interface. Configuring the PCI bus signals

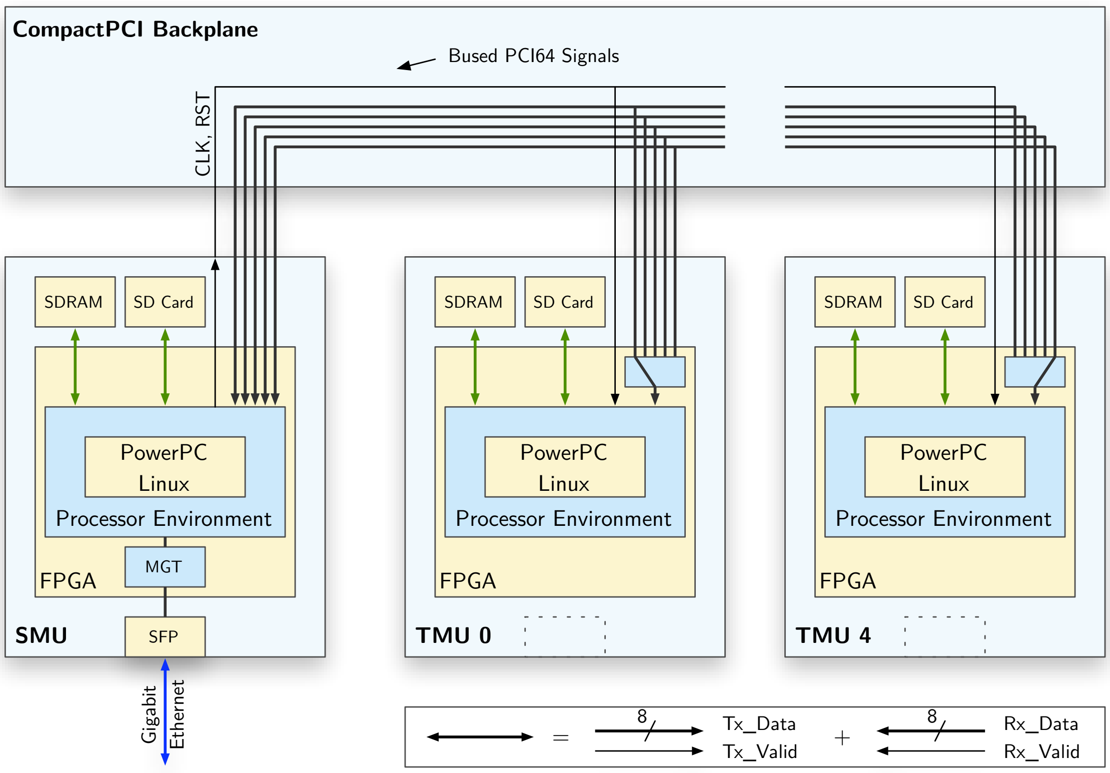


Figure 4.8: Architecture of a high-speed Linux-based control network for one GTU segment. The SMU acts as a router between the DCS network and the TMUs, using the CompactPCI backplane to implement a custom star topology network.

on the CompactPCI backplane to match a custom star topology network with the SMU at its center, the available signals suffice to form five separate full-duplex transmission channels. This architecture is illustrated in Figure 4.8. Each channel is able to transmit 8 bit in parallel at a rate of 66.7 MHz, leading to a maximum raw data rate of 533 Mbit/s between the SMU and each of the TMUs. An efficient implementation of the Point-to-Point Protocol (PPP)[7] enables these channels to be used to attach the TMUs to the IP-based DCS network. The large bandwidth available via these links allows for example to quickly update the file system stored on the SD cards or to upload complex test patterns to the GTU system.

[7]PPP is described by Internet Engineering Task Force (IETF) standard RFC 1661 [74].

5 Event Buffering

To efficiently operate on a large number of input values, a trigger processor has to be connected to the corresponding front-end electronics through low-latency, high bandwidth data links. In case of the ALICE TRD GTU, these data links are 1080 fiber optical links at an aggregate net bandwidth of 2.16 Tbit/s. However, regarding a detector not solely designed as a trigger detector, there is also the need for a raw data readout network, which collects the measured raw data and forwards it to a storage system. With respect to efficient use of resources, it is favorable to use the trigger data links also for raw data transmission. This effectively makes the trigger processor a part of the experiment's raw data readout chain.

Given the resources of a trigger processor such as the TRD GTU, the raw data readout functionality can include not only plain data forwarding, but high-speed event building in dynamically allocated multievent buffers. Temporarily buffering events is a recurring demand at various stages of data acquisition at high-energy physics experiments. In an experiment with a hierarchical system of several trigger levels at different corresponding trigger rates, different stages of event buffering are appropriate. As expected trigger rates translate to required buffer sizes and readout rates, triggering and event buffering designs are inherently related.

In this chapter, the benefits of dynamic multievent buffering are illustrated with the ALICE TRD as an example. Based on Monte Carlo simulations of the readout timing, the consequences of the derandomizing effect are demonstrated. A presentation of key features of the event buffering implementation including trigger handling in the GTU concludes the chapter.

5.1 TRD Raw Data Readout Timing and Requirements

Compared to other ALICE detectors, the TRD has a unique readout structure. Especially the characteristics of the data path used for the detector raw data differ significantly from those of other detectors. The following sections present an analysis of the properties of the TRD readout data path with respect to the important parameters *readout rate* and *dead time*. Multievent buffering is discussed as a concept to improve these measures.

In particle detector systems, the dead time is the time after each event during which the system is not able to record another event. The phrase *dead time* is also used to describe the percentage of time that a system is unavailable relative to the overall run-time. A large dead time is not desirable because it prevents the trigger system from selecting interesting (in particular rare) events for readout. To allow the ALICE Central Trigger Processor (CTP) to efficiently select interesting events, the average dead time should be below 10 %.

As the design of the readout network and the transmission lines connecting the TRD front-end to the GTU is driven primarily by the requirements of the low-latency L1 trigger functionality, this part of the readout chain offers more than one order of magnitude higher bandwidth than would be required for the raw data transmission. On the other hand, the TRD front-end electronics are not pipelined, directly translating the raw data transmission time into front-end dead time (which is generally fully covered up by the larger TPC dead time).

Another important part of the raw data readout chain is the connection between the GTU and the experiment's data acquisition (DAQ) system. As these transmission lines are not used for the trigger functionality, their bandwidth has to be only slightly above the requirements (including a safety margin). While there are 1080 optical links feeding the raw data into the system, there are only 18 *Detector Data Links (DDLs)* to the DAQ allocated running at essentially the same data rate. This is a factor 60 reduction in the available bandwidth in the GTU, compared to an effective average data rate reduction caused by L2 rejects of only about a factor of two.

These numbers indicate that if the GTU readout system was a non-pipelined design using only a single event buffer, the bottleneck introduced by the DDLs would result in a remarkable increase of TRD dead time. Multievent buffering at the interfaces between networks with different bandwidths, in case of the TRD inside the GTU, can solve this problem. If dimensioned correctly, it can smooth the data rate over time, reducing the bandwidth requirements from the peak data rate to the overall average data rate. But at the data rates present in the TRD, implementing multievent buffering requires serious design effort and fast buffer memory. To optimize the implementation, an estimate of the required size of the multievent buffer and the typical timing of the readout process for different running scenarios of the TRD is necessary.

The maximum bandwidth to the DAQ system poses another important constraint. If it is not sufficient, further data compression could be performed by the GTU. Using the additional MGT links on the SMU concentrator board (see Section 3.3), even additional data links to the DAQ are imaginable without a hardware redesign. Whether this is necessary will also be analyzed in the following subsections.

5.1.1 Event Sizes and Absolute Maximum Readout Rates

The general considerations in principle apply independently of the exact values for the TRD. But to gain concrete results with respect to the GTU requirements, it is important to base the analysis on a good estimation of the event sizes and data rates which can be expected in the experiment. In this subsection, the results of these estimations are presented.

Parameter	Symbol	Value
TRD acceptance	$\Delta\eta$	1.8
Secondary particles factor		2
Geometric coverage		90 %
Noise "particles" per event		18
Pads with signal per particle		4
Max. pads per halfchamber (HC)		1152
MCMs with signal per particle		1.12
Max. MCMs per HC		64
HC overhead		24 B
MCM overhead		8 B
Time bins	n_{timebin}	24
Tx delay HC → GTU		1.3 µs
Tx rate HC → GTU		220 MB/s
Pads per layer		196 992
MCMs per layer		10 944
SM overhead		200 B
HCs per SM		60
MCMs per SM		3648
Pads per SM and DCS		65 664
Tx rate GTU → DAQ		200 MB/s

Table 5.1: Parameters used for the trigger timing analysis.

The data sizes and readout times for different types of events can be estimated from the expected multiplicity density[1] $\frac{\mathrm{d}N_{\mathrm{ch}}}{\mathrm{d}\eta}$ and additional parameters reflecting the TRD's geometry and segmentation as well as properties of the front-end electronics and the readout network.

These parameters and their values are summarized in Table 5.1. While the basic geometry parameters have not changed, new information requires to update the estimations compared to the original assumptions of the TRD Technical Design Report [8]. The noise "particles" that have been included in the total number of particles in the TRD account for the chamber and electronics noise that can be expected. This noise can trigger readout of

[1]The *multiplicity density* $\frac{\mathrm{d}N_{\mathrm{ch}}}{\mathrm{d}\eta}$ indicates how many charged particles $\mathrm{d}N_{\mathrm{ch}}$ are produced within a pseudo rapidity interval of $\mathrm{d}\eta$ at a collision. The pseudo rapidity is a function of the production angle θ to the beam axis. It is defined as $\eta = -\ln(\tan(\frac{1}{2}\theta))$ and is a good approximation of the rapidity for relativistic particles [19]. The rapidity is the unitless relativistic velocity measure $\tanh^{-1}\beta$, wherein $\beta = \frac{v}{\mathrm{c}}$ with the particle velocity v and the speed of light c. The angular range covered by the TRD corresponds to a pseudo rapidity range of about $-0.9 \leq \eta \leq 0.9$. Thus, at an average multiplicity density of $\frac{\mathrm{d}N_{\mathrm{ch}}}{\mathrm{d}\eta} = 8000$, approximately 14 400 charged particles emerge into the spatial angle of the detector. This is the maximum value used in the design of the TRD.

Event type		p-p min. bias	Pb-Pb min. bias	Pb-Pb central	black
Multiplicity density	$\frac{\mathrm{d}N_{\mathrm{ch}}}{\mathrm{d}\eta}$	6	400	2000	∞
Particles in TRD (incl. noise)	N_{TRD}	37	1314	6498	∞
Max. number of particles per HC	$N_{\mathrm{HC}}^{\mathrm{max}}$	2.2	15.6	53.4	∞
Max. halfchamber data size	$S_{\mathrm{HC}}^{\mathrm{max}}/\mathrm{B}$	322	2094	6584	37 400
Max. number of GTU event buffers	$n_{\mathrm{buf}}^{\mathrm{max}}$	1067	164	52	9
Front-end readout time (after L1)	$t_{\mathrm{ro,L1}}/\mu\mathrm{s}$	2.76	10.8	31.2	171
Absolute maximum L1 rate	$R_{\mathrm{L1}}^{\mathrm{max}}/\mathrm{kHz}$	107.9	57.7	26.5	5.62
Average pad occupancy	o_{pad}	0.08 %	2.63 %	12.4 %	100 %
Average MCM occupancy	o_{MCM}	0.38 %	12.6 %	48.6 %	100 %
Average supermodule data size	$S_{\mathrm{SM}}/\mathrm{KiB}$	3.27	59.2	269	2082
TRD readout time (after L2)	$t_{\mathrm{ro,L2}}/\mu\mathrm{s}$	16.74	303.2	1378	10 660
Absolute maximum L2 rate	$R_{\mathrm{L2}}^{\mathrm{max}}/\mathrm{Hz}$	59 700	3300	726	94

Table 5.2: TRD occupancy, event sizes, and design readout rates for different types of events. The numbers are computed based on the multiplicity density and the values specified in Table 5.1. The number of particles in the TRD includes secondary particles and noise. Determining the derived numbers requires statistical considerations (see text).

a channel by raising analog-to-digital converter (ADC) values above the zero-suppression threshold. The value of one such case per event and supermodule layer is consistent with experience from test setups. In case of p-p events, this contribution to the event size should not be neglected. In the present TRD raw data format, in addition to the raw ADC values, there is a certain overhead for each multi-chip module (MCM), each halfchamber, and each supermodule contributing to the event data. These numbers are listed individually and considered in the calculation.

The number of activated pads per particle has a strong influence on the total event size. Experience from test runs suggests that (contrary to previous estimations) a number of at least four can be expected. Depending on the particle's angle, one to three pads have a direct hit. Two adjacent channels are transmitted in addition, and there is a probability that a particle causes hits in two adjacent pad rows because the geometry of the drift chambers is not fully projective.

The resulting event sizes and readout times are presented in Table 5.2 for three typical values[2] of $\frac{\mathrm{d}N_{\mathrm{ch}}}{\mathrm{d}\eta}$. Regarding the event sizes for a single halfchamber, the statistical effect of the TRD granularity has to be taken into account. As the TRD must signal its state as *busy* until the last halfchamber has transmitted all of its words, and also the GTU event buffers must be able to accommodate the largest halfchamber subevent, the *average maximum* halfchamber event size $S_{\mathrm{HC}}^{\mathrm{max}}$ is the relevant number. It can be estimated by taking into account the combinatorial probabilities for different spatial distributions of the

[2]The given multiplicity densities reflect current expectations. They are derived from [16].

detected particles. The difference between this number and the *average* halfchamber event size is most striking in case of p-p events, because the small number of particles leads to a particularly uneven distribution.

Estimating the event sizes for a complete supermodule, this effect does not need to be taken into account. The relevant value here is the *average* event size S_{SM}, since all 18 DDLs operate independently and event sizes can be averaged by multiple event buffers. The average pad and MCM occupancies[3] o_{pad} and o_{MCM} determine the data size and readout time. The occupancy numbers are higher than previous estimations (cf. [8]), mostly because latest beam test results suggest that there is a higher number of pads with signal per particle than previously assumed.

The absolute maximum readout rates are determined as the reciprocal of the respective readout time, in case of the L1 rate also taking into account the constant L1 trigger time of 6.5 µs. While the ratio between absolute maximum L1 and L2 rate can be as high as 36.5 in case of Pb-Pb central collisions, it is only 1.8 for p-p minimum bias events.

The maximum number of GTU event buffers is the number of buffers that could theoretically be implemented assuming a constant total buffer memory size of 4 MiB per stack and fully dynamic memory management. Even in case of Pb-Pb central collisions, more than 50 events would fit into this type of buffer memory.

These numbers represent the best possible current estimations, but they are not precise predictions. A number of different features or effects might increase the event data sizes or decrease the effective readout rate:

- One sixth of the ADC channels is digitized redundantly by the front-end electronics. For data integrity studies or calibration purposes, these redundant values could both be transmitted, increasing the event sizes by 17 %.
- The expected multiplicity density $\frac{\mathrm{d}N_{\mathrm{ch}}}{\mathrm{d}\eta}$ for Pb-Pb central events is still under discussion [18]. While results from RHIC suggest values of 1000–3000 [12] and latest estimations predict even lower values [20], the final value might be higher. In this range, event sizes are roughly proportional to the multiplicity density.
- The number of time bins that is stored could be chosen higher than 24, as the front-end electronics support values up to 32. Longer sampling times would however not only lead to a proportional increase in event size, but also prohibit the TRD trigger functionality because of the additional delay. An increase to 30 samples for a special running scenario would increase the event sizes by 25 %.
- The DDL speed is assumed to be at its nominal speed of 200 MB/s. This implies that there is no link congestion and that the DAQ system can sink the full sustained data rate. If the DAQ is for instance only able to operate the link at 140 MB/s, this would tighten the overall timing by 43 %.

[3]The estimation assumes that the front-end readout software sends no data for MCMs without a hit. The effect of the MCM occupancy on the total data size is almost negligible because of the small MCM overhead in the latest implementation.

These reservations should be kept in mind when judging the precise results. Because of these uncertainties, a substantial safety margin is inevitable in the design of the readout system.

5.1.2 Simulation of TRD Readout Timing and Multievent Buffering

The following subsection discusses the results from an event-based Monte Carlo simulation[4] of the trigger and TRD readout timing. The simulation uses a small set of parameters which has been chosen according to Subsection 5.1.1 as best current estimations, representing the latest TRD firmware implementation and incorporating findings from the latest TRD beam test.

The trigger timing is modeled using certain assumptions:

- Every pretrigger is accepted and followed by a L0 if the system is not busy, i. e., the L0 accept probability is 100 %. The dead time contribution caused by L0 rejects will be very small in Pb-Pb mode. In p-p mode, it is highly dependent on the pretrigger/L0 trigger configuration. In many cases, both will utilize the same input signals, thus justifying the simplification.
- The primary pretrigger input signals are evenly distributed in time, the time interval between two signals is distributed exponentially.
- All L0 events have a common constant probability P_{L1} for an L1 (accept), and all L0+L1 events have a common constant probability P_{L2} for an L2 accept. These assumptions imply optimum performance of the CTP. To recover from dead time caused by any detector and keep the average tape rate at its target value, the CTP might for instance sometimes issue L2 accepts at a much higher rate for a short period of time. Thus, the L2 rate would strongly fluctuate. This behavior of the CTP would certainly not be desirable if it happened drastically and regularly, since it could render almost any buffering useless.
- All events have the same size.
- The transmission rate to the DAQ is constant, there is no congestion in the DAQ system.
- The HLT accept ratio is one.
- There are no other detectors generating dead time.

Especially the last assumption will obviously not be fulfilled in the final system. However, since detailed timing models of each of the sub-detectors are not available, the simulation is confined to the TRD contribution. There will be contributions to the dead time by other detectors, both regularly after each L1-accepted event (such as the TPC dead time)

[4]The simulation is written in the C++ language and utilizes OpenMP symmetric multiprocessing. Each of the 1080 data points forming the Figures 5.1 to 5.4 results from a simulation of 5000 s (or, in case of p-p, 500 s) of experimental run-time.

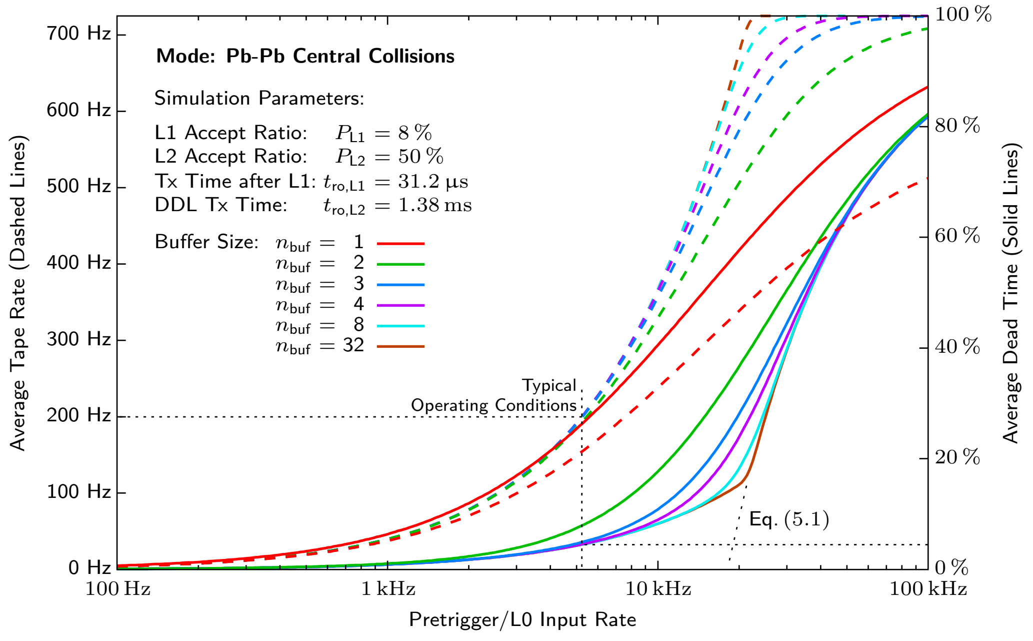


Figure 5.1: TRD trigger rate and dead time vs. L0 input rate for Pb-Pb central collisions (parameter: buffer depth).

and in special cases (such as buffer fill-ups). When running together with the TPC, its dead time will exceed the TRD L1 readout time.

To produce the graphs, the pretrigger/L0 input rate is varied. Results for varying the L1 or L2 accept ratio are comparable. The plots use the number n_{buf} of event buffers that the GTU provides as the parameter of a family of curves, especially illustrating the benefit of a multievent buffer. In the following subsections, the simulation results for two typical LHC running scenarios are discussed.

Pb-Pb Collisions

In the Pb-Pb mode of LHC operation, there will be a combination of minimum bias and central events. While the trigger mix is not fixed at present, the central collisions will determine the data rate limits as a consequence of their much higher number of generated particles. The experiment aims at a tape rate of approximately 200 Hz for these events. The L2 trigger is expected to discard about every second event because of pile-up in the TPC. For several envisioned running modes (such as with a dielectron trigger, cf. [9, p. 12-16]), the final event rate will be limited by the L1 trigger, accepting only a small percentage of events. The simulation assumes an L1 accept ratio of $P_{L1} = 8\,\%$.

Figure 5.1 shows the average TRD dead time and the resulting average tape rate depending on the pretrigger input rate for different GTU event buffer sizes. For an average tape rate of 200 Hz, an average pretrigger input rate slightly above 5 kHz has to be selected (see

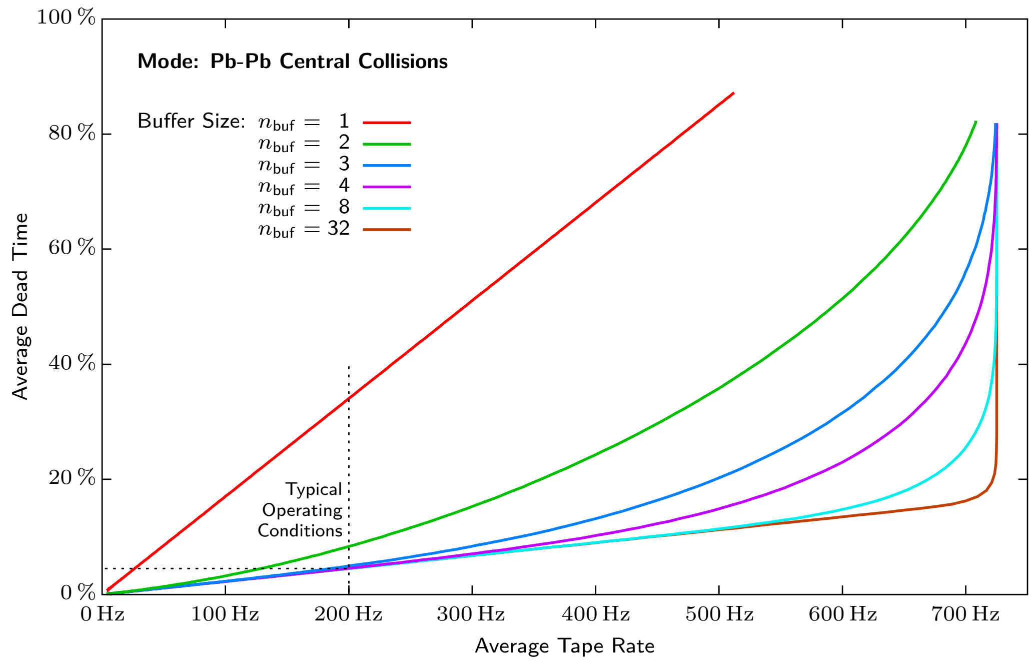


Figure 5.2: TRD dead time vs. accepted trigger rate at event sizes corresponding to Pb-Pb central collisions (parameter: buffer depth).

dotted lines). The solid lines together with the scale on the right show the resulting dead time for a given pretrigger rate. Regardless of the GTU buffer size, the average dead time rises significantly at a certain input rate because of DDL bandwidth saturation. This corresponds to the absolute maximum average tape rate of $R_{\text{L2}}^{\text{max}} = 726\,\text{Hz}$. While multievent buffering cannot overcome this effect, it can decrease dead time and increase the average tape rate so that it more closely approaches the absolute rate limit.

For low L0 input rates, the dead time D is directly proportional to the primary trigger rate f_{L0} with the minimum dead time after L0 ($t_{\text{d,L0}} = 6.5\,\mu\text{s}$) and the average L1 readout time $t_{\text{ro,L1}}$ as parameters. For very high rates, the multievent buffer is constantly saturated, and the maximum average tape rate $R_{\text{L2}}^{\text{max}}$ is the determining factor:

$$D^{\text{low}}(f_{\text{L0}}) = f_{\text{L0}}(t_{\text{d,L0}} + P_{\text{L1}} \cdot t_{\text{ro,L1}}) \qquad D^{\text{high}}(f_{\text{L0}}) = 1 - \frac{R_{\text{L2}}^{\text{max}}}{f_{\text{L0}} \cdot P_{\text{L1}} \cdot P_{\text{L2}}} \tag{5.1}$$

As seen in Figure 5.1, the dead time behavior for typical operating conditions between these extreme cases depends strongly on the number of event buffers.

The direct correlation between average tape rate and average dead time is shown in Figure 5.2. It illustrates the use of a multievent buffer. For low average L2a rates, the dead time is defined by the L1 readout time and the average L1 rate. It is less than 10 %. When approaching the defined maximum average L2a rate ($R_{\text{L2}}^{\text{max}} = 726\,\text{Hz}$), the average dead time does not rise much thanks to the multievent buffering capability of the TRD. At the defined critical rate, there is a step in the function, and it suddenly rises to almost 100 %

dead time when the buffers are saturated. Since this mode of operation is undesirable, the average L2a rate has to be tuned to stay near but below the threshold, which is defined by the reciprocal DDL readout time.

At the assumed operation conditions, the average TRD dead time would be as high as 34 % if the GTU only had a single event buffer (red line). Implementing a second event buffer reduces the dead time to 8.4 %, finally reaching 4.5 % for three or more event buffers.

With only a single event buffer, the TRD has to signal *busy* after each event until the event is rejected by a missing L1 or an L2 reject or until the data has been transmitted to the DAQ. With the minimum dead times after L0 and L1 ($t_{\mathrm{d,L0}} = 6.5\,\mu\mathrm{s}$, $t_{\mathrm{d,L1}} = 80\,\mu\mathrm{s}$), the resulting dead time D_1 can be estimated from the average tape rate f_{tape} as:

$$D_1 = \left(t_{\mathrm{ro,L2}} + \frac{t_{\mathrm{d,L1}}}{P_{\mathrm{L2}}} + \frac{t_{\mathrm{d,L0}}}{P_{\mathrm{L1}}} \cdot P_{\mathrm{L2}} \right) \cdot f_{\mathrm{tape}} = c_1 \cdot f_{\mathrm{tape}} \tag{5.2}$$

Applying the values expected for Pb-Pb operation (see Table 5.2 and Figure 5.1), the factor c_1 amounts to 1.70 ms.

With a larger number of event buffers, on the other hand, the average dead time is dominated by the inevitable TRD front-end dead time after L0 and L1. For a very large number, it approaches the theoretical minimum that is defined by:

$$D_\infty = \begin{cases} \left(\frac{t_{\mathrm{ro,L1}}}{P_{\mathrm{L2}}} + \frac{t_{\mathrm{d,L0}}}{P_{\mathrm{L1}}} \cdot P_{\mathrm{L2}} \right) \cdot f_{\mathrm{tape}} = c_\infty \cdot f_{\mathrm{tape}} & f_{\mathrm{tape}} < R_{\mathrm{L2}}^{\mathrm{max}} \\ 1 & f_{\mathrm{tape}} = R_{\mathrm{L2}}^{\mathrm{max}} \end{cases} \tag{5.3}$$

with in this case $c_\infty = 225\,\mu\mathrm{s}$.

For a practical number of event buffers, results lie between these extremes. Figure 5.2 indicates that at the typical tape rate three event buffers are required for minimum dead time, while at a higher tape rate of 500 Hz, eight event buffers should be utilized. A larger number of event buffers would only be favorable in case of an even higher average tape rate.

p-p Collisions

In the p-p mode of LHC operation, the experiment aims at a tape rate of approximately 1 kHz. As a result of the reduced multiplicity density of the events, data sizes will be about two orders of magnitude smaller than in Pb-Pb operation, increasing the absolute maximum L2 rate by about the same factor. As opposed to this, the absolute maximum L2 rate increases only by a factor of four because granularity effects and absolute latencies limit the speed of the front-end electronics. Therefore, the dead time is dominated by the detector front-end and Figure 5.3 has a slightly different structure. In a typical mode of operation, data taking is controlled mostly by the early trigger levels (pretrigger/L0). For this simulation, the L1 and L2 accept ratios are set to 50 %, also implementing a TPC past/future protection in case of too many p-p events per drift time.

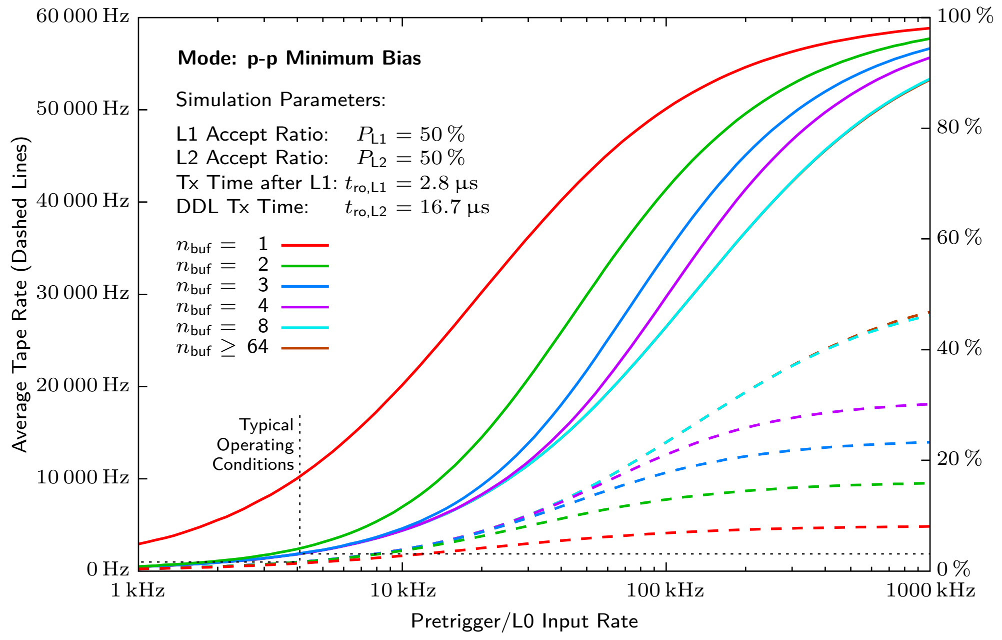


Figure 5.3: TRD trigger rate and dead time vs. L0 input rate for p-p minimum bias collisions (parameter: buffer depth).

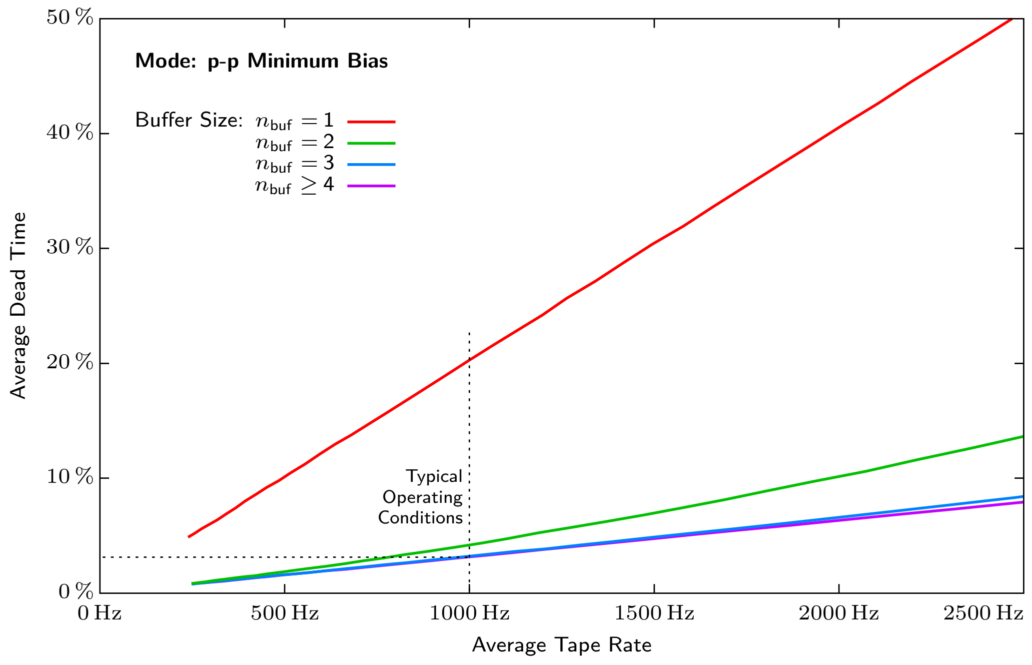


Figure 5.4: TRD dead time vs. accepted trigger rate for p-p minimum bias collisions (parameter: buffer depth).

Estimating the average dead time, the limits are again described by equations (5.2) and (5.3). With the values expected for p-p operation (see Table 5.2 and Figure 5.1), the factors amount to $c_1 = 203\,\mu\text{s}$ and $c_\infty = 31.5\,\mu\text{s}$.

At an average tape rate of 1 kHz, the average dead time in case of a single event buffer would amount to 20 %. As can be seen from Figure 5.4, a second event buffer reduces this value to 4.2 %, and with three event buffers, the average dead time reaches the minimum of 3.2 %. The effect of more than three event buffers is significant only at tape rates beyond 2 kHz. While there would be buffer memory available for much more events than in Pb-Pb mode, the requirements are even more relaxed. Thus, there is no need for special treatment of p-p mode when designing the multievent buffer.

Results

Implementation of multievent buffering in the GTU is required to reduce the TRD dead time to an acceptable level. The number of available event buffers can be small, three event buffers suffice for typical operating conditions.

To reduce the effect of possible issues like additional rate fluctuations or short temporary DAQ failures to a certain extent, it is advisable to further increase the number of GTU event buffers beyond the obtained optimum value of three. A value of eight could be an appropriate trade-off.

With a DDL transmit time of 1.38 ms per event, a three-event buffering TRD running at 200 Hz average L2a rate results in only about 4.5 % dead time, while a single-event buffer would lead to approximately 34 % dead time.

If the absolute numbers for values like data rates and trigger accept rates are varied to account for new developments, the associated graphs are basically shifted. The qualitative effect of multievent buffering stays the same. Thus, even with moderate changes to the readout parameters, the number of required event buffers stays in a similar range.

There is no need for further data compression (entropy coder for instance) in the GTU, although this remains as an option should operating parameters largely deviating from the expectations require an additional speed-up. While zero-suppressed TRD raw data has low entropy because of the nature of the detector, results from the TPC suggest that about a factor 2 could be gained in addition.

With the assumed input values to the estimations, there is a safety margin of factor 2–3 in the TRD readout timing. The average tape rate could be increased by that factor while maintaining a TRD dead time of less than 10 %. This margin can be used to account for effects leading to increased event data sizes or a diminished DAQ transfer rate (see Section 5.1.1). It also provides a certain safety against possible L2 rate fluctuations.

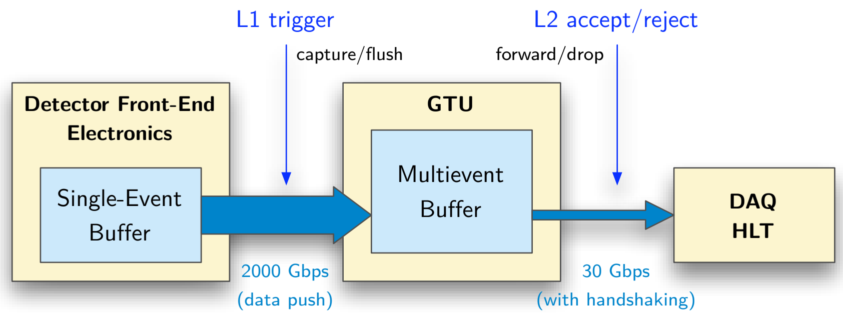


Figure 5.5: The principal event buffering strategy of the TRD. The GTU implements a multievent buffer. While it receives data each time a L1 is issued, data is transferred to DAQ and HLT only after a L2 accept trigger message.

5.2 Event Buffering Implementation

The dead time of a detector system strongly depends on the employed readout and event buffering scheme. Given a fast non-pipelined detector front-end and a slower upstream acquisition system, which requests the event data sets asynchronously and selectively, building an event buffering system in between is important for the performance of the detector. To minimize detector dead time, the system has to support fast data reception and a large number of event buffers. As the common zero-suppressed event data sets vary in size, dynamic memory allocation is needed to maximize the number of events that can be stored.

With wide data paths at high frequencies, implementing such an event buffer based on FPGA technology easily approaches the performance limits of current FPGA architectures. This section outlines crucial parts of the solution developed for the ALICE TRD.

Figure 5.5 shows the principal event buffering strategy of the TRD. Raw data is read out in a two-stage procedure. Following an L1 trigger, data is transmitted from the single-event buffer in the detector front-end to the trigger processor GTU. In the GTU, data is stored in a multievent buffer until the associated asynchronous L2 trigger is received, which either initiates forwarding to the DAQ and HLT systems or causes the event to be dropped from memory in case of a reject. Trigger sequences can be interleaved, i. e., data of a new event can arrive while it is not yet known whether the previous events are to be transmitted to DAQ or not. While data transmission to DAQ utilizes a handshaking protocol, data reception from the detector front-end is designed as a true data push architecture optimized for minimum delay.

As illustrated in Figure 5.6, the event buffering design is implemented in the GTU in parallel to the trigger computation, partly sharing the same data path. As only the TMUs can sink the high input data rate, and the SMUs provide the interface to the DAQ system, raw data handling is divided into multievent buffering in the TMUs and readout control including event building in the SMUs.

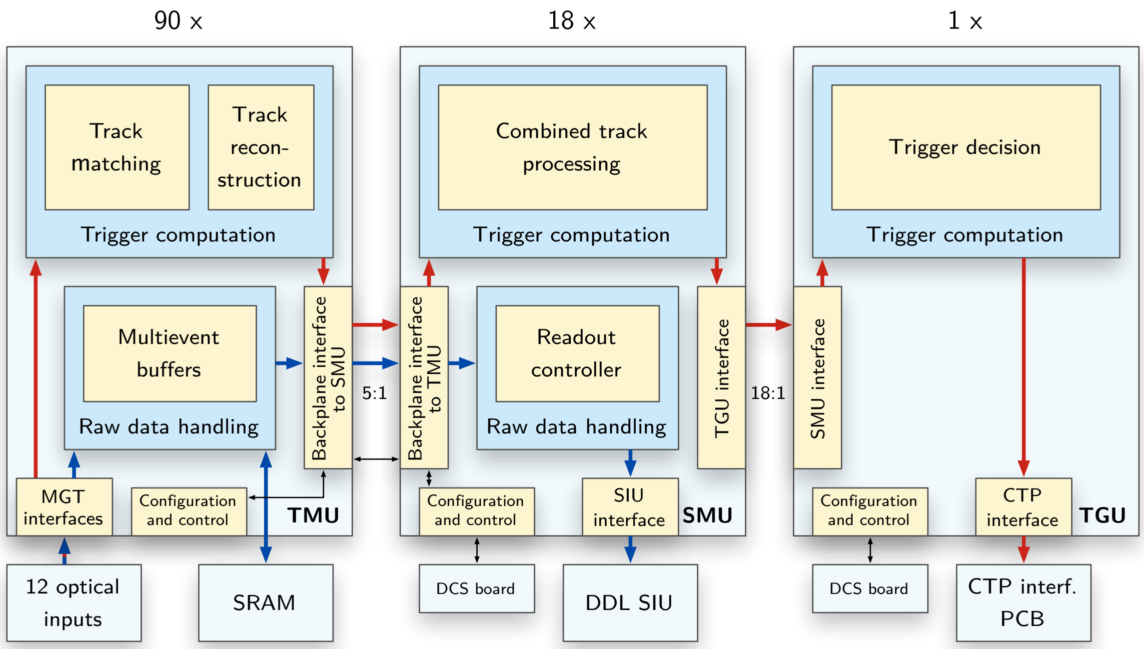


Figure 5.6: The overall architecture of the GTU design. Trigger computation and raw data handling designs share the resources of the TMU and SMU FPGAs.

5.2.1 High-bandwidth Multievent Buffering

Variable event sizes and interleaved trigger sequences demand a complex buffering scheme in the TMU. The corresponding design is presented in detail in [61]. It utilizes the single 4-Mbit SRAM as 12 logically independent dynamic ring buffers as shown in Figure 5.7).

A major challenge is constituted by the requirement to merge 12 independent 16-bit data streams at 2.5 Gbit/s with a total net data rate per TMU of 23 Gbit/s in real time (see Figure 5.8). As the FPGA-internal memory blocks do not provide sufficient capacity to store even a single event, the achieved bandwidth to the external SRAM has to match the incoming data bandwidth. Because the SRAM is attached to the FPGA via a 128 bit wide interface at 200 MHz, a highly pipelined, 128 bit wide data push architecture is needed. While receiving data from the front-end, a write transaction of a full 128-bit word is required in at least 94 % of the clock cycles. The less time-critical read accesses are handled by interlacing the requests in between the write cycles.

At the required width of the data path, even uninvolved combinational designs such as multiplexers can complicate routing significantly so that the target design speed of 200 MHz cannot be reached. However, by heavily utilizing the FPGA's embedded true dual-ported BRAM blocks, the timing requirements can be met. Figure 5.9 shows as an example the data aligning buffer for a single link. Via asymmetric access to the memory matrix, both 16/32-bit data collation and clock domain crossing can be accomplished using the same hardware primitive.

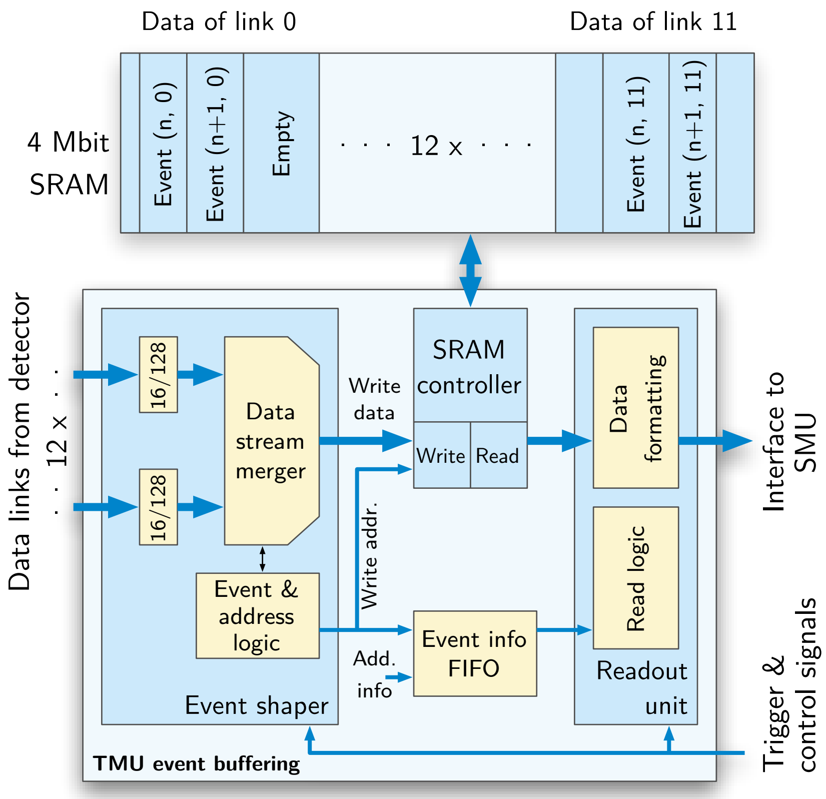


Figure 5.7: The event buffering architecture of the TMU. Event data received through 12 optical links is stored in a dynamically allocated multievent buffer. Depending on trigger and control signals, the data is forwarded to the SMU for readout. Source: based on [61]

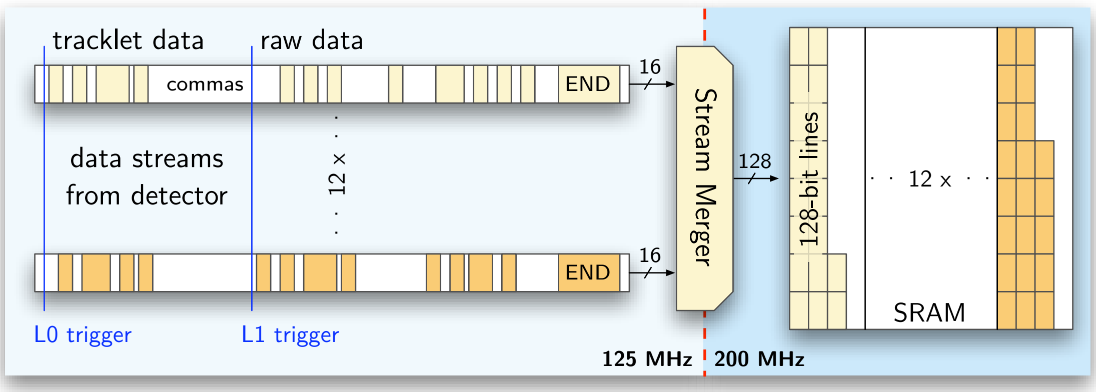


Figure 5.8: Both preprocessed track segment (tracklet) information and raw ADC data is transmitted through the same 12 links per TMU. The unsynchronized data streams are merged to generate a continuous stream of 128-bit words at 200 MHz to fill the SRAM event buffers. Source: adapted from [61]

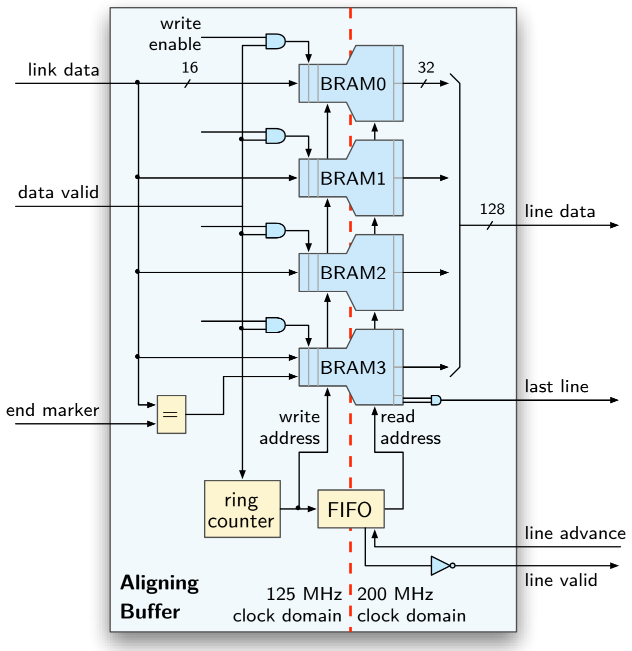


Figure 5.9: The data aligning buffer for one of the 12 links. Reformatting a stream of 16-bit words with gaps to a stream of 128-bit words at 200 MHz can be accomplished in an FPGA by carefully optimizing the design around the embedded BRAM blocks. Source: adapted from [61]

5.2.2 Handling Readout Requests

An important feature of a multievent buffer as a means of derandomizing the data flow is the ability to forward events upon request selectively and without interruption. In the ALICE experiment, the readout requests are distributed as trigger messages via the TTC network and received in the TRD by the GTU's SMU modules.

Figure 5.10 shows an outline of the SMU architecture. A specialized controller processes the incoming trigger messages and administrates the event buffers of the associated TMUs. It also controls a data path design merging the data from the stacks of the same supermodule as a first step of event building and encapsulating the resulting data stream with status information for transmission via the DDL.[5] As the SMU does not offer sufficient memory resources to buffer a full event, a back pressure mechanism is employed while receiving from the TMUs. The PowerPC interface allows to easily configure special modes of operation such as the CTP simulator mode, which allows the GTU to run the TRD or single supermodules without external infrastructure, and to retrieve a record of past trigger events and errors for analysis.

[5]Details concerning the structure of the resulting data stream are provided in the ALICE TRD DAQ data format specification [25]

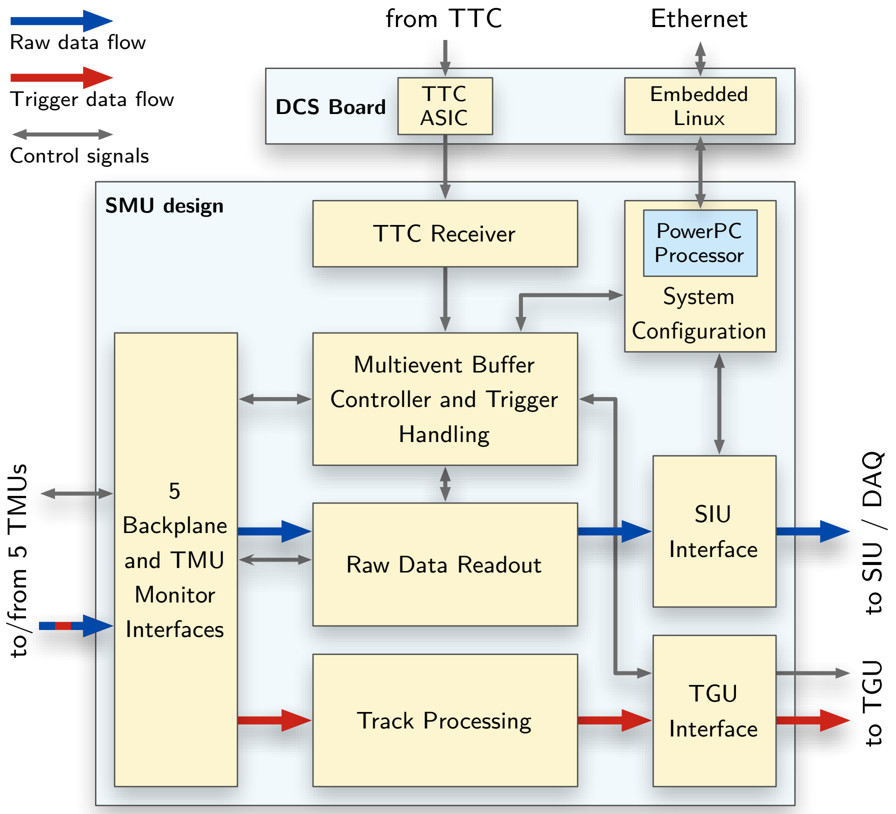


Figure 5.10: The architecture of the SMU. Based on trigger signals received via the TTC, a trigger controller logic coordinates raw data readout on the associated TMUs as well as preprocessing and merging of the raw data streams on the SMU. Source: based on [47]

The considerable complexity of the controller results from the support for interleaved trigger sequences, which make the detector performance independent of the timing of the L2 trigger message, and the consequent large number of possible trigger input errors, which have to be handled gracefully to ensure reliable operation of the system. A sketch of the corresponding design is presented in detail in [47].

Figure 5.11 summarizes the timing of both trigger computation and raw data readout. The presented case is an example for a full trigger sequence as seen by the GTU. The expected values for the indicated readout times at different running scenarios are presented in Section 5.1.1.

Conclusion Multievent buffering is an important feature for the performance of the ALICE TRD. Its implementation is challenging because of the high bandwidths involved in the ALICE experiment, dynamic event sizes, and the complex trigger system supporting interleaved sequences. As presented in this chapter, it can be accomplished in the FPGA-based trigger processor GTU.

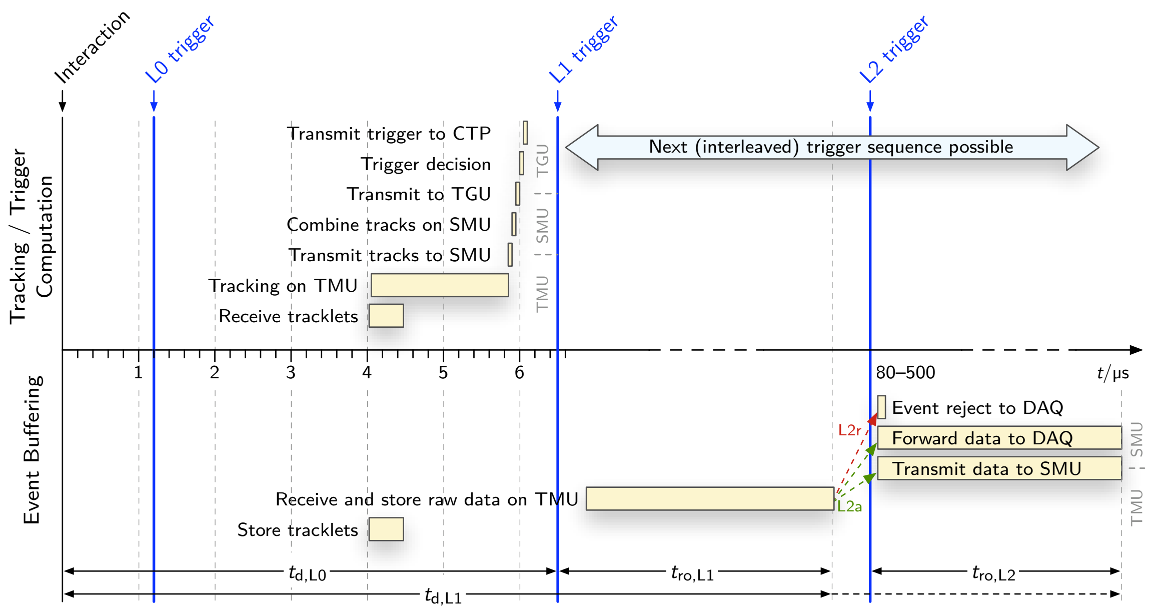


Figure 5.11: The TRD readout timing for a typical sequence of L0, L1 and L2 triggers. Between L1 and L2 triggers, another L0 trigger of an interleaved trigger sequence is possible.

6 Front-end Data Preprocessing

Extensive data preprocessing in the front-end electronics[1] is an important part of the ALICE TRD trigger concept. The computations are performed by freely programmable microprocessors. Controlled by an assembler program, they can implement arbitrary calculations within the available processing time. Each of the approximately 250 000 processing units (CPUs) can be assigned a particular, independent program via the detector's configuration network. In addition, registers are available in the CPUs for numeric constants, which are also configurable separately for each CPU.

In trigger readout mode, the program performs the final steps of fitting a straight line to the position of the detected charge clusters. It reads the sum values precomputed by the preprocessor during drift time and calculates from them the axis intercept and slope of the best fit straight line (see [38] for reference). Subsequently, these track segment parameters have to be corrected for certain effects (see Section 6.2). Finally, a first estimate of the particle's transverse momentum is derived from the slope of the track segment. For track segments whose estimated transverse momentum exceeds a configurable threshold, the parameters are finally compiled into a data word, which is transferred to the readout network.

While this thesis focuses on the design of the central part of the trigger system, the Global Tracking Unit (GTU), the online tracking strategy created in this context covers also parts of the computations in the front-end electronics. In the first section of this chapter, the format of trigger data communication from the front-end to the GTU is specified. The second section summarizes additional data preprocessing to be performed in the front-end. The presented computations are optimized particularly for efficient implementation in the front-end processors.

6.1 Contents of the Transmitted Data Words

As a result of the tight latency requirements, only a single 32-bit word can be transmitted to the GTU per track segment. This data word has to include all information relevant to the global track reconstruction with the necessary precision. A careful specification of the contents of this data word is crucial to the performance of the trigger system.

Considerations regarding the kind of parametrization and the required precisions at which the parameters are transferred are discussed in detail in [23]. Table 6.1 summarizes the

[1]For the TRD front-end processors, the term *Local Tracking Units* (LTUs) has been used in previous publications [8]. It is avoided here because of its ambiguity with respect to other ALICE components.

Name	Symbol	Granularity	Range of values	Bits
Axis intercept	y	160 µm	$-643.2\,\mathrm{mm} \ldots 643.2\,\mathrm{mm}$	13
Deflection	d_y	140 µm	$-8.8\,\mathrm{mm} \ldots 8.8\,\mathrm{mm}$	7
Pad row	z	1	$0 \ldots 15$	4
e^- probability	P	0.39 %	$0 \ldots 1$	8

Table 6.1: Contents of the data word which is transmitted to the GTU for each track segment.

resulting contents of the transmitted data words. Each word contains the following four values:

- The y position of the intercept point between track and outer chamber surface, as a signed distance from the chamber center,
- The absolute deflection of the track inside the chamber, i. e., the difference between the y positions at the outer and the inner chamber surface,
- The z position as the number of the pad row (0–15),
- The probability (between 0.0 and 1.0, deduced from the measured charge), that the corresponding particle is an electron.

For efficiency reasons, floating-point numbers are unsuitable for the transmission. In order to optimally use the available range of values, the values are instead multiplied with an individual precision factor and rounded to the nearest integer before transmission.

Axis Intercept and Deflection The linear regression computed in the front-end CPUs immediately delivers two values for each track segment: the y coordinate of the intercept point between straight line and outer chamber surface relative to the position of the front-end multi-chip module (MCM) and the slope $\frac{\mathrm{d}y}{\mathrm{d}x}$ of the straight line [38]. Both values are given in units of the pad width in the respective chamber.

As the information relevant to the GTU is the absolute position of a track segment with respect to the stack of detector modules, the local y values have to be translated into global coordinates[2] before transmission. This requires scaling according to the pad width and addition of a constant offset based on the position of the individual front-end MCM. As an advantage of this strategy, minor chamber misalignment can be easily accounted for by modifying the respective position offset values when configuring the front-end electronics.

Instead of the slope $\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{d_y}{d_x}$, in the following the *deflection* d_y within the drift chamber (see Figure 6.1) is regarded, which is equivalent because the thickness $d_x = 3\,\mathrm{cm}$ of the drift chambers is constant. Scaling the measured deflection with the constant pad width gives the absolute deflection, which is an important result of the front-end processing. Based on the deflection, the front-end decides whether a detected track segment is to be transmitted

[2]The term *global coordinate* here refers not to the entire detector, but to a module stack.

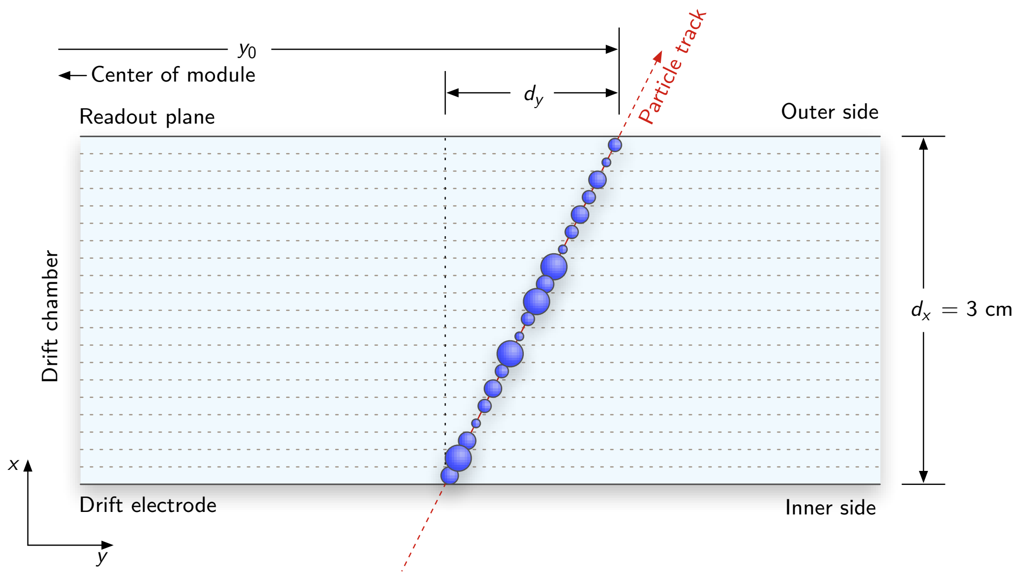


Figure 6.1: Definition of y position and deflection within the drift chamber.

to the GTU, thus contributing to the early data reduction required for an efficient trigger implementation. The decision is based on the comparison of the angle between particle track and vertex direction[3] with a threshold, which effectively implements a local cut on the estimated transverse momentum (see Section 6.2.3). To be used in the corresponding calculations, the deflection first has to be corrected for two effects that affect its measured value: the change of drift direction in a magnetic field (see Section 6.2.1) and a value shift introduced by the TRD's pad geometry (see Section 6.2.2). Both corrections are also included in the absolute deflection value transmitted to the GTU.

Electron Probability The electron probability of a track segment, i. e., the probability that the detected track segment originates from an electron or positron as opposed to a pion, is encoded in 256 steps using 8 bit. Transforming the measured charge into an electron probability essentially involves checking a constant lookup table in which the standardized values of the probability distribution (Figure 2.5) are stored. By implementing this strongly nonlinear operation in the front-end processors instead of transmitting raw charge values to the GTU, the available resolution of the transmitted values can be used to full capacity. In addition, it allows for changes of the particle identification strategy without the need to modify the track segment data format. Figure 2.6 shows that the deviation between the average pulse heights for electrons and pions is larger in the second half of the drift time. It may therefore be beneficial to consider the total charges in different subranges of the drift time separately. The different possibilities are still being

[3] *Vertex direction* is the direction from a given point to the primary interaction point at the center of the detector. It is the direction of a track produced by an imaginary particle with infinite transverse momentum.

examined at present. Several strategies of electron/pion separation in the TRD as well as first test results are discussed in [82]. To support procedures incorporating table lookups, the front-end CPUs have configurable memory available that can be initialized during detector configuration.

6.2 Computations in the Detector Front-End

The front-end processors execute a local tracking algorithm that provides the point of intercept y_0 with the chamber outer surface and the measured deflection in y direction $d_{y,\,\mathrm{raw}}$. However, additional calculations need to be performed by the front-end electronics to cooperate with the GTU and provide it with a track segment parametrization as specified above. These computations are an important prerequisite for the tracking calculations performed in the GTU. As shown in the following sections, they can be implemented efficiently in the front-end processors.

6.2.1 Lorentz Angle Correction

The homogeneous $\vec{B}$ field which permeates the detector in longitudinal direction also affects the drift of the ionization electrons inside the chamber.

When a $\vec{B}$ field is present, the drift direction is no longer parallel to the electric field, but the electrons are deflected in y direction on their way to the pads. The drift velocity of the electrons is reduced. Figure 6.2 illustrates the effect. The resulting drift angle is the *Lorentz angle* Ψ_L. If $\vec{E}$ and $\vec{B}$ are perpendicular, it is given by

$$\tan(\Psi_\mathrm{L}) = \frac{\mathrm{e}\tau B}{m}$$

(cf. [50]). In this equation, τ is the average time between two collisions with gas molecules. Via τ, the Lorentz angle depends on the composition of the gas mixture and the strength of the $\vec{E}$ field. In the experiment, it will presumably amount to $\Psi_\mathrm{L} \approx 7°$.

While the measured y position at the chamber outer surface is hardly impaired by this effect, the measured deflection d_y has to be corrected. A charge cluster from the chamber inner surface reaches the pad row shifted by an offset of

$$d_\mathrm{Lorentz} = -\tan(\Psi_\mathrm{L}) \cdot d_x = -3.68\,\mathrm{mm}\,, \tag{6.1}$$

while a charge cluster at the chamber outer surface is not deflected. To correct for this effect, the constant value d_Lorentz has to be added to each measured deflection:

$$d_y = d_{y,\,\mathrm{raw}} + d_\mathrm{Lorentz}\,.$$

Thus, the Lorentz angle correction is a correction of the deflection or the slope of the track segment.

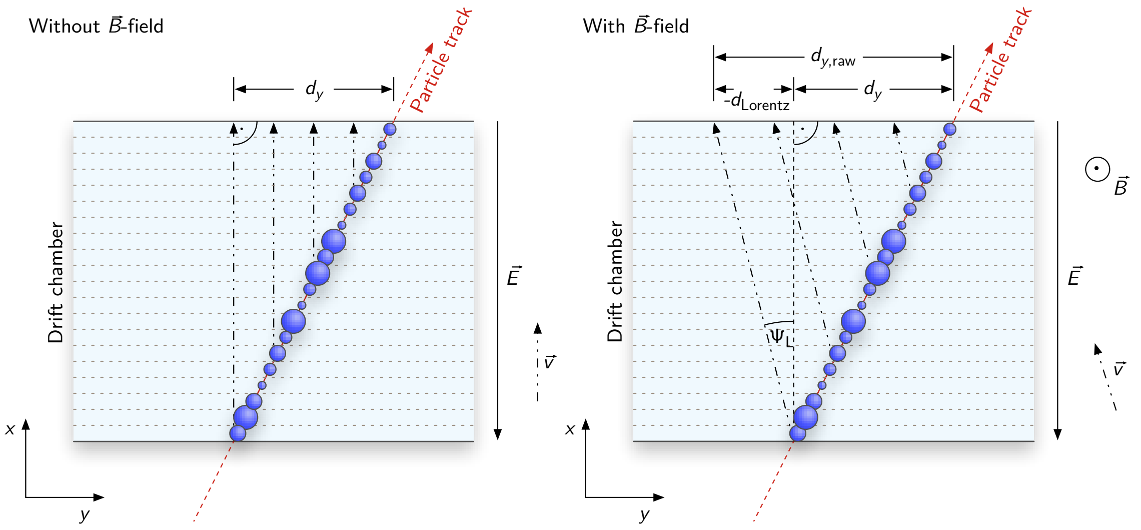


Figure 6.2: Drift direction of the electrons in the chamber without (left) and with (right) $\vec{B}$ field. The additional field changes the direction by the Lorentz angle Ψ_L and thus the measured deflection by the distance $|d_\text{Lorentz}|$.

The Lorentz force reduces the electron drift velocity and increases the drift length. This effect is accounted for by setting the drift time, which is a scaling factor of the slope, accordingly in the front-end processors.

6.2.2 Tilted Pads Correction

The cathode pads at the chamber outer surface are not accurately rectangular, but deliberately slightly shifted to parallelograms, using opposite directions on adjacent detector layers (*tilted pads*). The tilting angle amounts to $\beta_\text{tilt} = 2°$ (see Figure 6.3). This configuration allows the offline analysis to improve the resolution in z direction for traversing tracks by combining data from several layers. Without tilted pads, the course of the particle track in z direction — assuming ideally projective detector geometry — could not be determined more precisely than the width of a pad row ($d \approx 10\,\text{cm}$).

The trigger computation, however, does not benefit from the tilted pads because it does not require a high precision z coordinate. Nevertheless, the pad geometry has to be taken into account even in the online analysis since it considerably adulterates the measured y position and deflection.

The y position is distorted because charges with the same actual y position in the frontier region between two adjacent pads are projected to one or the other pad depending on their z position. The error in y amounts to $\pm\frac{d}{2} \cdot \tan(\beta_\text{tilt}) \approx 1.8\,\text{mm}$ at most and cannot be corrected in the front-end, since the exact z position is unknown at chamber level.

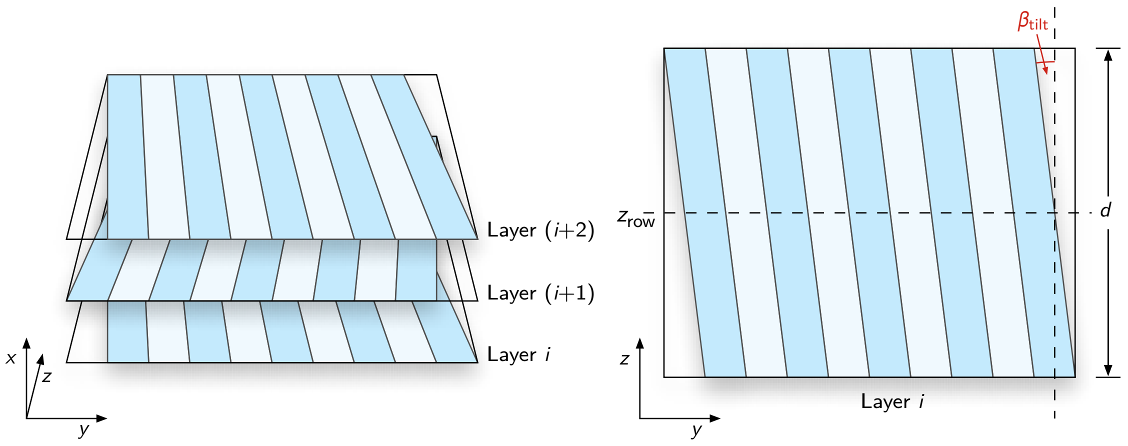


Figure 6.3: Schematic representation of the cathode pads of three pad rows lying on top of each other. To increase the position resolution in z direction, the pads in the different layers are tilted alternatingly by an angle of $\pm 2°$ (*tilted pads*).

Because with tilted pads the measured y position is no longer independent of the z position, also the measured slope $\frac{\mathrm{d}y}{\mathrm{d}x}$ changes. This effect can however be corrected in the front-end for primary particles with the aid of the primary vertex assumption [80].

Instead of the real y coordinate, with tilted pads the measured value is

$$y' = y \pm (z - z_{\text{row}}) \cdot \tan(\beta_{\text{tilt}}) \,. \tag{6.2}$$

with z_{row} being the z coordinate at the center of the pad row. Instead of $d_y = y_{\text{outer}} - y_{\text{inner}}$, the measured deflection d'_y accordingly follows as:

$$\begin{aligned} d'_y &= y'_{\text{outer}} - y'_{\text{inner}} \\ &= d_y \pm (z_{\text{outer}} - z_{\text{inner}}) \cdot \tan(\beta_{\text{tilt}}) \\ &= d_y \pm \Delta z \cdot \tan(\beta_{\text{tilt}}) \,. \end{aligned} \tag{6.3}$$

Using the primary vertex assumption, according to Figure 6.4 Δz can be derived in good approximation from the position of the pad row as

$$\Delta z = \frac{z_{\text{row}} \cdot d_x}{x_0} \,.$$

Hence, the error in the local deflection caused by the tilted pads can be corrected by adding the value

$$\begin{aligned} d_{\text{tilt}} &= \pm \frac{z_{\text{row}} \cdot d_x}{x_0} \cdot \tan(\beta_{\text{tilt}}) \\ &= \pm 1.048\,\text{mm} \cdot \frac{z_{\text{row}}}{x_0} \,, \end{aligned} \tag{6.4}$$

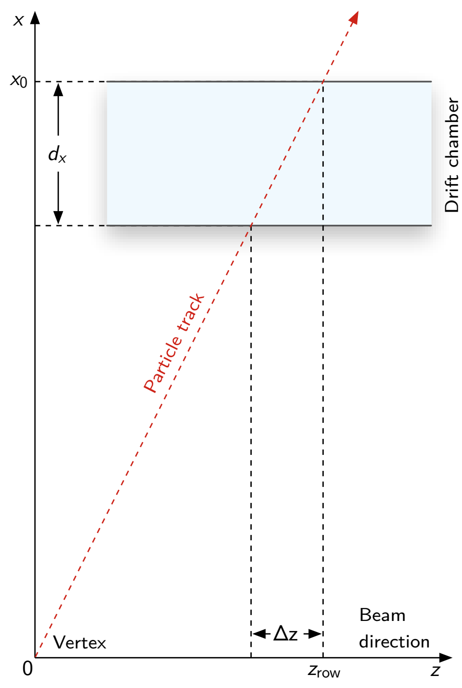


Figure 6.4: In the x-z-plane, the particle tracks are straight. For particles originating from the primary vertex, the deflection Δz in the chamber can be computed directly from the coordinates x_0, z_{row} of the pad row.

where

$$\begin{aligned} x_0, z_{\text{row}} &: \text{Coordinates of the pad row}\,, \\ d_x &: \text{Width of a chamber (3.0 cm)}\,, \\ \beta_{\text{tilt}} &: \text{Pad angle with respect to the } z \text{ axis}\,. \end{aligned}$$

Together with the Lorentz angle correction, two (locally constant) correction terms result:

$$d_y = d_{y,\,\text{raw}} + \underbrace{d_{\text{Lorentz}} + d_{\text{tilt}}}_{d_{\text{corr}}} \,. \tag{6.5}$$

The sum d_{corr} of both values can be precomputed for each pad row of the detector and stored in the front-end processors of this row to be added to every measured deflection. In that way, both corrections are reduced to the addition or subtraction of a single suitable constant.

6.2.3 Local Cut on the Transverse Momentum

For efficiency reasons, only track segments with sufficiently large transverse momentum or, equivalently, sufficiently small deflection against the vertex direction should be transmitted to the GTU. To achieve this, it is not necessary to calculate the angle against the vertex direction directly. Instead, the maximum deflection angle $\alpha_{\max}$ can be translated into a minimum and maximum acceptable deflection $d_{y,\,\min}$ and $d_{y,\,\max}$.

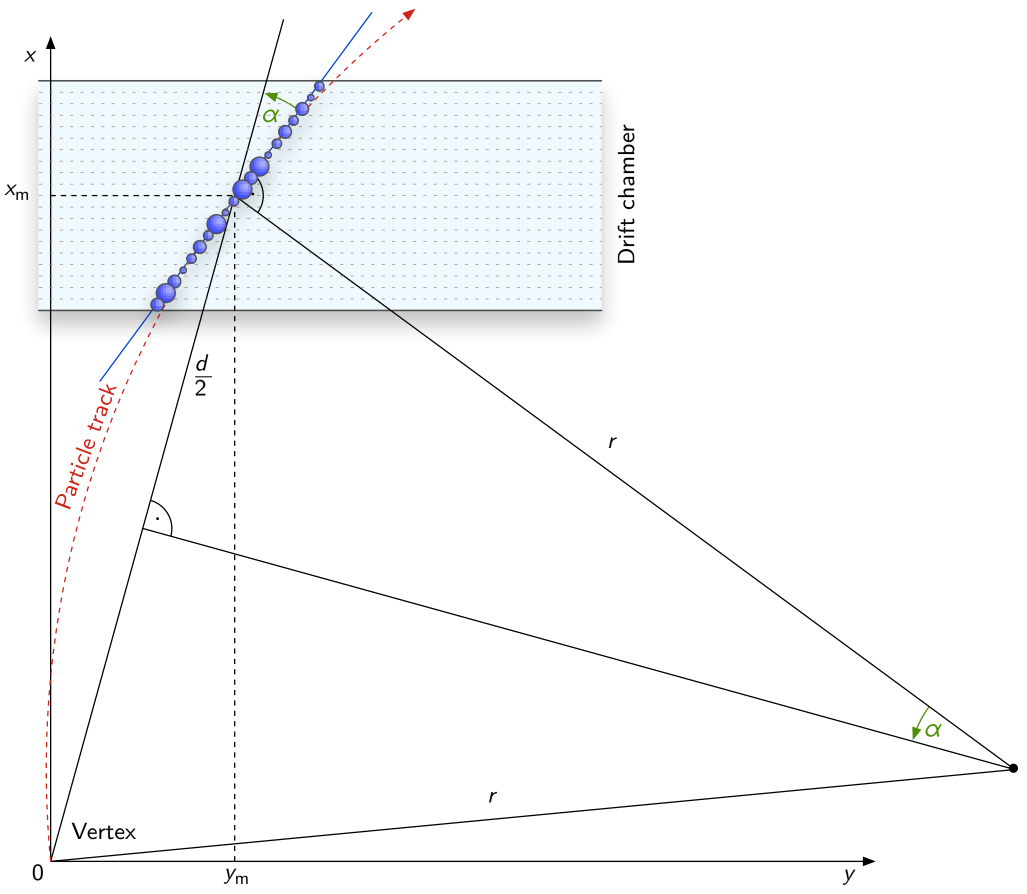


Figure 6.5: Schematic representation of the circular particle track in relation to a detector module. From the angle α of the track segment to the vertex direction, the radius r of the particle track can be deduced. For illustration, the curvature of the particle is strongly exaggerated.

Figure 6.5 illustrates the correlation between the angle α of the track segment to the vertex direction and the radius r of the circular particle track. In the upper right-angled triangle, the radius r is given by

$$r = \frac{d/2}{\sin(\alpha)}\,, \tag{6.6}$$

where $d = \sqrt{x_\text{m}^2 + y_\text{m}^2}$ is the distance between track segment and beam. In the case of well-known magnetic induction B, the transverse momentum p_t of the particle results from the radius r of the particle trajectory in the x-y-plane as

$$p_\text{t} = \text{e} \cdot r \cdot B = 0.30\,\text{GeV}/c \cdot \frac{r}{\text{m}} \cdot \frac{B}{\text{T}}\,, \tag{6.7}$$

where the elementary charge e is written as $\text{e} = 0.30\,\frac{\text{GeV}/c}{\text{m}\cdot\text{T}}$ [35]. Hence, the maximum deflection angle α_max can be calculated from the minimum transverse momentum $p_\text{t, min}^\text{FE}$

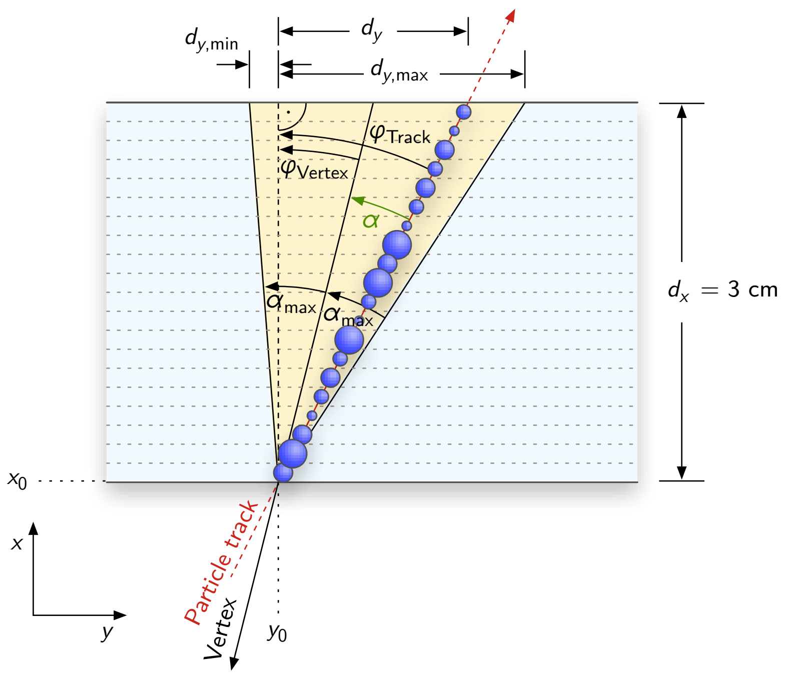


Figure 6.6: Schematic representation of the particle trajectory through a detector module and the thresholds for minimum and maximum deflection. From the deflection d_y, the angle α between track segment and vertex direction can be concluded. Instead of comparing α with α_{max}, d_y can be compared directly with $d_{y,\,\mathrm{min}}$ and $d_{y,\,\mathrm{max}}$.

as

$$\alpha_{\mathrm{max}} = \arcsin\left(\frac{\sqrt{x_{\mathrm{m}}^2 + y_{\mathrm{m}}^2} \cdot eB}{2 \cdot p_{\mathrm{t,\,min}}^{\mathrm{FE}}}\right) . \tag{6.8}$$

The currently implemented value of $p_{\mathrm{t,\,min}}^{\mathrm{FE}} = 2.3\,\mathrm{GeV}/c$ results in a maximum deflection angle of $\alpha_{\mathrm{max}} = 5.5°$ for outer pads.

To compute the angle α to the vertex direction from the deflection d_y, the x position of the chamber and the y position of the track segment are required (see Figure 6.6). The angle results as:

$$\tan(\varphi_{\mathrm{Vertex}}) = \frac{y_0}{x_0} , \qquad \tan(\varphi_{\mathrm{Track}}) = \frac{d_y}{d_x} ,$$
$$\alpha = \varphi_{\mathrm{Track}} - \varphi_{\mathrm{Vertex}} = \arctan\left(\frac{d_y}{d_x}\right) - \arctan\left(\frac{y_0}{x_0}\right) . \tag{6.9}$$

The quantities x_0 and d_x are constant for each chamber, and y_0 can be regarded as constant in sections within the necessary resolution. Using equations (6.6), (6.7), and (6.9), the transverse momentum p_{t} can be written as a bijective function of the deflection d_y. Thus, instead of using the condition $|p_{\mathrm{t}}| \geq p_{\mathrm{t,\,min}}^{\mathrm{FE}}$ directly, the condition

$$d_{y,\,\mathrm{min}} \leq d_y \leq d_{y,\,\mathrm{max}}$$

can be checked with suitably chosen thresholds $d_{y,\,\mathrm{min}}, d_{y,\,\mathrm{max}}$ (see Figure 6.6).

The minimum and maximum deflection $d_{y,\,\mathrm{min}}$ and $d_{y,\,\mathrm{max}}$ result from the maximum deflection angle α_{max} as

$$\begin{aligned} d_{y,\,\mathrm{min}} &= d_x \cdot \tan\left(\arctan\left(\frac{y_0}{x_0}\right) - \alpha_{\mathrm{max}}\right) \,, \\ d_{y,\,\mathrm{max}} &= d_x \cdot \tan\left(\arctan\left(\frac{y_0}{x_0}\right) + \alpha_{\mathrm{max}}\right) \,. \end{aligned} \tag{6.10}$$

The thresholds $d_{y,\,\mathrm{min}}, d_{y,\,\mathrm{max}}$ can be precomputed individually for each front-end processor according to the equations (6.8) and (6.10) using the MCM's position in the supermodule and several different y_0 values from the its y range. By storing them in a lookup table in the processor's memory, the front-end processor is enabled to retrieve the appropriate thresholds for the segment's y position from the lookup table while processing a track segment. Comparisons with the measured deflection d_y result in a decision whether the track segment is rejected or handed over to the readout network.

The two presented corrections and the selection procedure can be implemented efficiently in the front-end CPUs, since the computations consist in total only of three additions or comparisons. The location-dependent correction and threshold constants are generated by a specific software individually for each front-end processor. As both the general scaling of the deflection and the Lorentz angle depend on the drift velocity of charges in the gas mixture, the configuration has to be regenerated dynamically during run-time if gas or high voltages parameters change.

7 Online Track Recognition

As a trigger processor, the GTU contributes to the level-1 (L1) trigger decision of the ALICE experiment. As input values, it receives data from the six layers of TRD detector modules. Because the data of the individual modules is not sufficiently accurate to directly allow for reliable particle identification or transverse momentum determination, the trigger processor has to combine and analyze the data of all detector layers to recognize and reconstruct continuous particle tracks.

The input data received from the TRD front-end electronics is specifically optimized for this task. It consists of preprocessed and parametrized *track segments* that are detected on the individual layers. Figure 7.1 displays the structure of the 32-bit data word that is transmitted for each track segment. Up to 240 such track segments are received per event by each of the 90 processing nodes of the GTU.

Time constraints of the ALICE experiment limit the total processing time for the trigger computation in the GTU to less than 2 µs. This reduces the choice of applicable algorithms remarkably. Especially sophisticated iterative tracking techniques are generally unsuitable for the low-latency trigger computation. Instead, the procedures have to be selected in such a way that they can be realized efficiently in hardware.

A common approach to online tracking in high-energy physics is to use a histogramming method[1] based on a Hough transform. In case of the ALICE TRD, this would require massive parallel histogramming even for a coarse representation of the source data. As an advantage, histogramming commonly delivers an execution time increasing only linearly with the occupancy. However, finding the accumulation points can be expensive if effects of the cell boundaries are to be considered. In addition, the coarse resolution is a major

[1]In a histogramming procedure, the track segments are entered into a multi-dimensional histogram which is then searched for accumulation points.

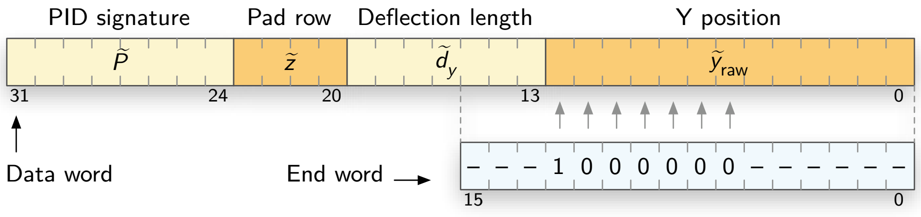


Figure 7.1: Composition of the 32-bit data words that are transferred to the GTU. The end word contains only 16 bit and is identified by the indicated bits.

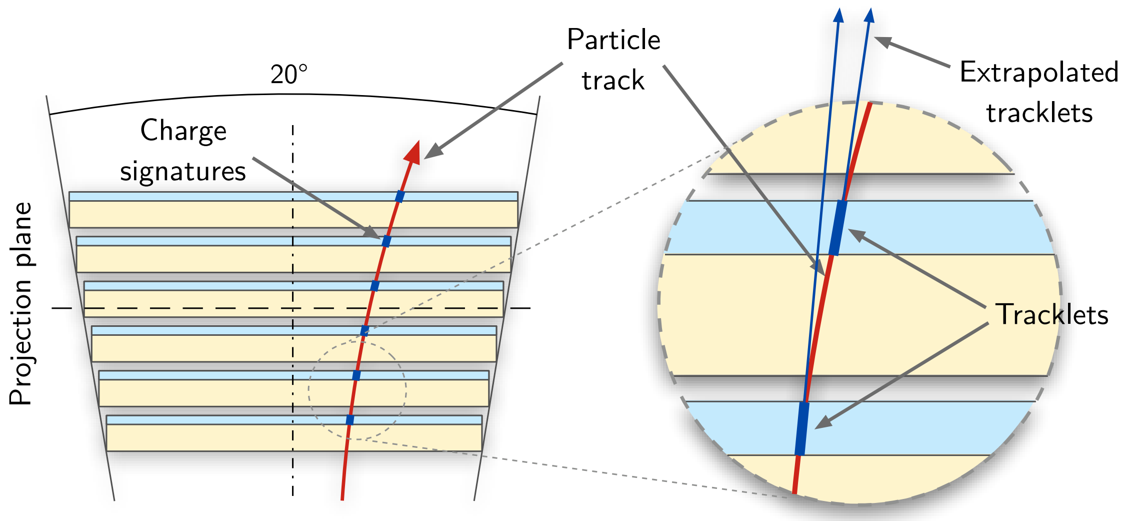


Figure 7.2: The track segments (tracklets) are projected to a virtual center plane. Source: adapted from [68]

disadvantage of this method, as it can result in tracks that are tainted by non-matching track segments.

While parallel histogramming is the standard approach, the potential of contemporary FPGA technology suggests the application of more sophisticated tracking techniques. This chapter demonstrates that within certain parameters and with careful optimization, an approach based on direct value comparisons is feasible even within tight low-latency requirements.

7.1 Overview of the GTU Trigger Computations

7.1.1 Tracking Algorithm

An online tracking algorithm based on direct value comparisons requires an easy verifiable criterion that indicates on the basis of the data of several track segments whether these most likely belong to the same track or not. In case of the ALICE TRD, different track segments produced by the same particle exhibit a similar deflection angle α to the vertex direction. Their z positions projected to the x-z plane form a straight line together with the collision point. Finally, assuming only slightly curved tracks, their y positions projected to the x-y plane approach a straight line with a slope $\frac{\Delta y}{\Delta x}$ similar to the the slope of the individual track segments.

To examine these conditions, a straightforward approach is to project the track segments to a conceived common plane perpendicular to the x axis. The track segment's slope is used for the projection in y direction; in z direction, it is projected towards the primary

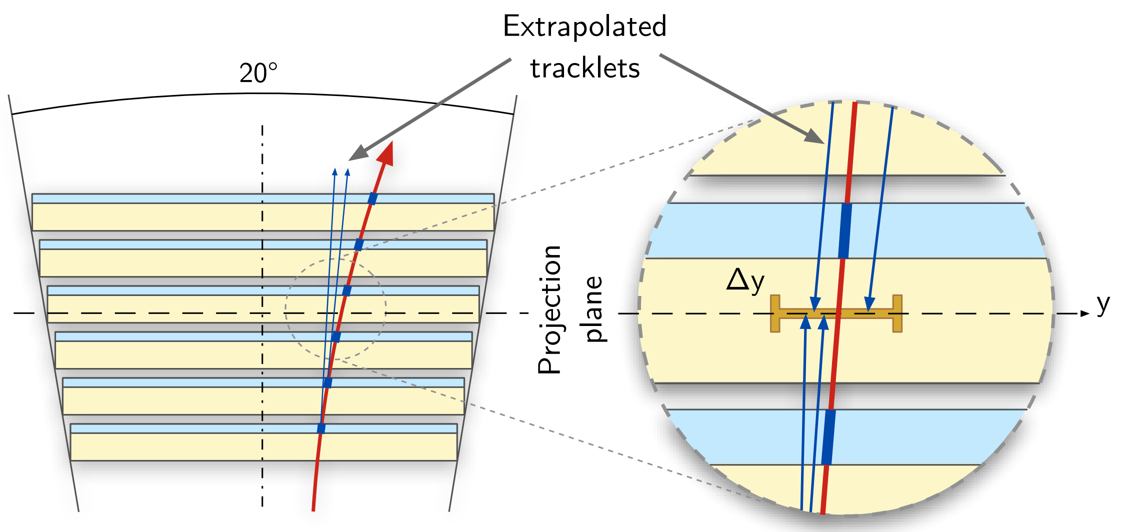


Figure 7.3: Schematic representation of the window criterion for matching track segments in the GTU.

vertex. In order to keep the errors in the projected position as small as possible and also as similar as possible for track segments from all layers, the center plane of a stack of modules is used as projection plane. Figure 7.2 illustrates the prolongation of the track segments to the virtual center plane.

Track segments belonging together are characterized by the facts that their projections on the common plane are located closely together and that their angles α almost agree. This suggests the application of a window criterion for matching track segments as it is shown in Figure 7.3. Track segments that differ only slightly in the three computed measures should be combined to a track. Thus, track matching corresponds to the search for window positions in the projection plane that contain more track segments than a given threshold. The GTU assumes a found particle track if there are at least four track segments from different layers within the three-dimensional window.

Aiming for an algorithm based on direct value comparisons, the straightforward approach of comparing each of the up to 40 track segments per module [67] successively to all track segments of the other layers is impractical. Each TMU would have to perform sequentially up to $\frac{40 \cdot 5 \cdot 40 \cdot 6}{2} = 24\,000$ paired comparisons, resulting in a processing time of 0.4 ms assuming a design clock rate of 60 MHz. This exceeds the available time by more than two orders of magnitude.

The process of finding segments belonging to one track is essentially a task of three-dimensional matching. If, however, the track segments are sorted according to one of the three variables, then not all but only sequential track segments have to be considered to evaluate the matching criterion in this variable. However, this one-dimensional solution is not transferred easily to the three-dimensional problem.

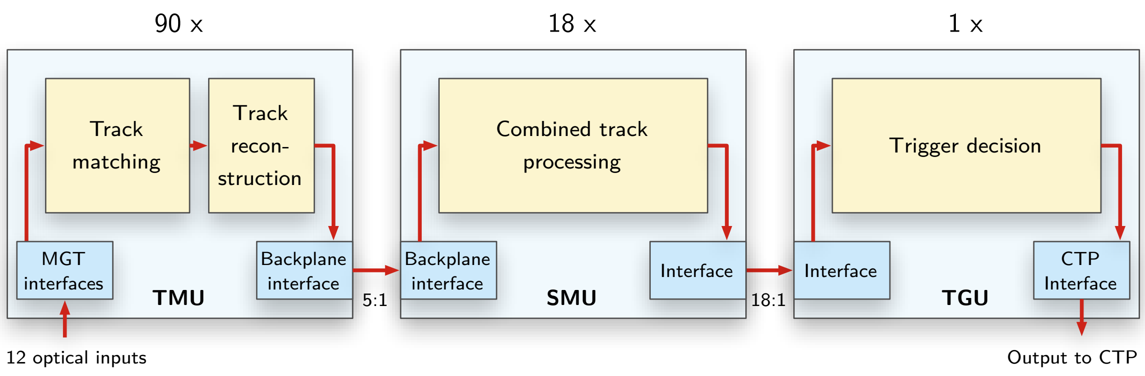


Figure 7.4: The architecture of the GTU trigger design

The intelligent sliding window algorithm presented here takes advantage of the fact that the three comparison values are asymmetric in their properties. As a result of the TRD's geometry, the dimensions differ significantly in terms of accuracy and range of values. The *distinctiveness* of a variable with respect to track matching can be determined by comparing the expected width of the matching window with the range of possible values for this variable. The (projected) y coordinate is by far the most distinctive variable. The used window width amounts to only about $\frac{1}{110}$ of the range of values. With approximately $\frac{1}{5}$ of the range of values, the deflection angle α is the least distinctive value. In the z position, the fraction amounts to about $\frac{1}{16}$. To optimize the implementation, the variables should be considered in the order determined by their distinctiveness.

Because of the special TRD readout system, the z coordinate is an exception to this rule. The presented algorithm makes use of the configurable readout order of the TRD front-end electronics, allowing for processing to already begin during the transmission period. The track segments are read out from each half module sorted according to the pad row number $\tilde{z}$ in ascending order. To utilize this natural order, the z values of the track segments are compared first. As a specific characteristic, the range of the number z of the pad row covers only 16 possible values. The explicit comparisons of the (projected) z coordinates can thus be realized implicitly by considering all possibilities in parallel. This procedure is explained in Section 7.3.

Since the (projected) y coordinate is the most distinctive variable, its deviation between different track segments subsequently forms the main track matching criterion. If track segments are found that agree sufficiently in z and y, the angle α of the track segments is finally compared as well.

7.1.2 Architecture Outline

Following the architecture of the GTU, the trigger design is divided into three components representing the hierarchical stages. It is outlined in Figure 7.4. The first component, implemented in the 90 TMU modules, performs online track matching and track reconstruction, which is the most expensive part of the trigger computation. As a result,

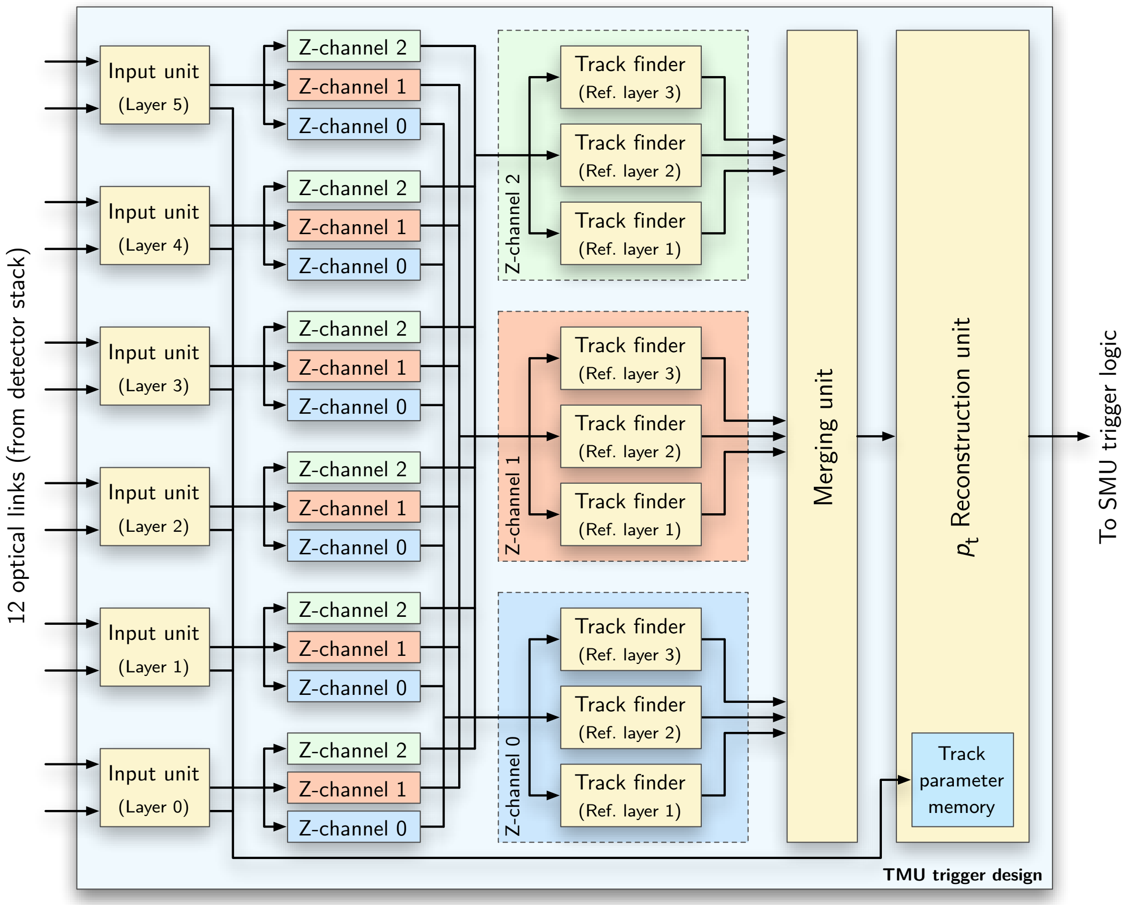


Figure 7.5: The architecture of the TMU trigger design. The TMU receives the track segment data from a module stack, combines it to tracks using several processing stages and reconstructs the transverse momentum of the original particles. The data of the found tracks are transmitted to a superordinate trigger logic that combines the results of the different TMUs.

parametrized tracks are transmitted to the 18 SMUs. In the SMU, tracks from the associated TMUs are combined. The resulting information, which is transmitted to the TGU, as well as the trigger decision algorithm in the TGU itself depend on the chosen trigger scenario.

Each of the GTU's 90 identical TMUs independently processes the data of one detector stack. Since the distance between the stacks is relatively large compared to the curvature of the target particle tracks, and as the geometry of the detector modules is designed to be projective to the point of interaction, looking for tracks across several stacks of detector modules is not necessary. Figure 7.5 gives an overview of the architecture of the TMU trigger design.[2] In the input units, the parametrized track segments from the front-end

[2]See [23] for a detailed discussion of a corresponding hardware implementation. The implementation of

electronics of the six layers are received and further parameters concerning track matching are precalculated. The z-channel units then implicitly check the agreement of the track segments in the z direction and sort them according to their y coordinate. In the 9 track finder units running in parallel, the track segments of the different layers are combined to tracks by comparing their y coordinates and deflection angles. The results of the track finder units are combined into a data stream of found tracks. The reconstruction unit finally calculates for each track the transverse momentum p_t of the producing particle.

The design of the units adheres to several common principles that are important for an efficient low-latency implementation. The first principle is massive parallelization at various stages: Computations that can be performed independently at the same time are processed in parallel by duplicating the relevant execution unit. Another major concept is the use of a fully pipelined data push architecture. When intermediate results are determined, they are passed on immediately to the next unit. There is no handshaking between the individual design units, i. e., a unit may pass on data to the next at any time and also has to be able to accept data at every clock cycle.[3]

Another important principle is to use only bit-optimized signals and fast integer arithmetics. The data signals that represent physical quantities are implemented as integers with minimum bit widths by determining the necessary resolution and the range of values for each signal. The numbers are generally signed and realized in two's complement representation. The input values to the GTU, e. g., use specifically optimized resolutions (see also Table 6.1):

$$y_\mathrm{raw}^\mathrm{res} := \frac{1}{160\,\mu\mathrm{m}}\,, \qquad d_y^\mathrm{res} := \frac{1}{140\,\mu\mathrm{m}}\,, \tag{7.1}$$

$$\tilde{y}_\mathrm{raw} := y_\mathrm{raw} \cdot y_\mathrm{raw}^\mathrm{res}\,, \qquad \tilde{d}_y := d_y \cdot d_y^\mathrm{res}\,. \tag{7.2}$$

In this chapter, the tilde symbol denotes integer variables as used in the hardware design. If possible, complex operations are substituted by precomputed lookup tables. And finally, a key principle is the utilization of the transmission time: The processing of the track segments begins already while data is still being received from the front-end electronics.

7.2 Computations for Each Track Segment

Three important parameters can be computed for each track segment individually using only a small amount of processing time before the actual track matching begins: the projected y coordinate, the angle to the vertex direction in the transverse plane, and the tilted-pad aware coefficient y'.

the tracking strategy and architecture summarized here is optimized with respect to the GTU hardware as presented in Chapter 3 and adapted to the final geometry of the ALICE TRD.

[3]Where applicable, this has to be ensured by fast buffer memories.

7.2.1 Projecting the Y Coordinate

To provide a straightforward means of comparing track segments from different detector layers, the measured y coordinate y_{raw} of each segment can be projected onto a common virtual center plane using the track segment's deflection:

$$y_{\text{proj}} = y_{\text{raw}} - d_y \cdot \underbrace{\frac{x_i - x_{\text{mid}}}{d_x}}_{-C_i^{y_{\text{proj}}}} \tag{7.3}$$

with:

$$\begin{aligned} x_i &: x \text{ coordinate at the layer } i \text{ drift chamber exterior surface} \\ x_{\text{mid}} &: x \text{ coordinate of the target projection plane } ((x_0 + x_5)/2 = 3.215\,\text{m}) \\ i &: \text{original layer of the track segment} \\ d_x &: \text{thickness of a drift chamber } (30\,\text{mm})\,. \end{aligned}$$

As the constant $C_i^{y_{\text{proj}}}$ depends only on the detector layer i and fixed geometry values, it can be statically precalculated for each of the six layers.

Converting Equation (7.3) to integer arithmetics in a form suitable for hardware implementation yields[4]

$$\tilde{y}_{\text{proj}} = \left\lfloor \frac{1}{2} \left\lfloor \frac{1}{2^2} \left(\tilde{y}_{\text{raw}} + \left\lfloor \frac{\tilde{C}_i^{y_{\text{proj}}}}{2^e} \tilde{d}_y \right\rfloor \right) \right\rfloor + \frac{1}{2} \right\rfloor \approx \frac{1}{2^3} \cdot \tilde{y}_{\text{raw}} + \frac{\tilde{C}_i^{y_{\text{proj}}}}{2^{3+e}} \cdot \tilde{d}_y \tag{7.4}$$

with the number e of additional bits used for the multiplication and the constant

$$\tilde{C}_i^{y_{\text{proj}}} = -\frac{x_i - x_{\text{mid}}}{d_x} \cdot \frac{y_{\text{raw}}^{\text{res}}}{d_y^{\text{res}}} \cdot 2^2. \tag{7.5}$$

depending only on the layer i. Thus, a suitable multiple of the deflection — depending on the detector layer — is added to the raw y coordinate. Cutting off the least significant bits in combination with the second addition achieves a rounding of the projected y value. The result is the projected y coordinate

$$\tilde{y}_{\text{proj}} = y_{\text{proj}} \cdot \frac{y_{\text{raw}}^{\text{res}}}{2^3} = \frac{y_{\text{proj}}}{1.28\,\text{mm}}\,. \tag{7.6}$$

The resolution of the projected y position is chosen to be lower because of the limited precision of the scaled deflection length.

[4]The square brackets $\lfloor\,\rfloor$ denote the floor function, which maps a real number to the next lower integer. Dividing a value by a power of two and consequently applying the floor function effectively truncates bits from the right side of the value's binary representation. Adding $\frac{1}{2}$ before truncating means that the value is rounded arithmetically.

7.2.2 Computing the Deflection Angle

Another criterion for identifying track segments from the same track is the track segment's deflection angle α to the vertex direction. It is given by:

$$\alpha = \arctan\left(\frac{d_y}{d_x}\right) - \arctan\left(\frac{y_\text{raw}}{x_i}\right) \approx \frac{d_y}{d_x} - \frac{sY}{x_i}\,. \tag{7.7}$$

The error in the approximation is negligible because the values for the arc tangent's arguments are sufficiently small.[5] Accordingly, the natural unit for this angle is given by $\alpha^\text{res} = d_x \cdot d_y^\text{res}$:

$$\tilde{\alpha} = \alpha \cdot \underbrace{d_x \cdot d_y^\text{res}}_{\alpha^\text{res}} = \frac{\alpha}{4.\bar{6}\,\text{mrad}} \approx \frac{\alpha}{0.27^\circ}\,. \tag{7.8}$$

Equation (7.7) is expressed using integer arithmetics as

$$\tilde{\alpha} = \left\lfloor \tilde{d}_y + \frac{1}{2} \cdot \left\lfloor \left\lfloor \frac{\tilde{y}_\text{raw}}{2^4} \right\rfloor \cdot \frac{\tilde{C}_i^\alpha}{2^{10}} \right\rfloor + \frac{1}{2} \right\rfloor \approx \tilde{d}_y + \frac{\tilde{C}_i^\alpha}{2^{15}} \cdot \tilde{y}_\text{raw}\,, \tag{7.9}$$

with the constant

$$\tilde{C}_i^\alpha := -\frac{d_x}{x_i} \cdot \frac{d_y^\text{res}}{y_\text{raw}^\text{res}} \cdot 2^{15} \tag{7.10}$$

depending only on the layer i of the detector module and fixed geometry parameters, which can be precomputed statically for each layer. By not scaling the deflection $\tilde{d}_y$, an addition and a multiplication with a constant suffice to compute the deflection angle of a track segment in hardware.

7.2.3 Precomputing Reconstruction Parameters

To precisely reconstruct tracks in consideration of the TRD's tilted pad geometry requires the computation of tilted-pad-aware y positions y' for each track segment as explained in Section 8.1. The value y' is computed using Equation (8.4):

$$y' = y_\text{raw} - \underbrace{(-1)^i \cdot z_{i,\,\text{row}} \cdot \tan(\beta_\text{tilt})}_{-C_{i,z}^{y'}}$$

where β_tilt is the tilting angle of the pads and $z_{i,\,\text{row}}$ is the z coordinate at the center of the corresponding pad row. As the values for $C_{i,z}^{y'}$ depend only on the layer i and the number of the pad row $\tilde{z}$, the values are again precomputed and the online computation of $\tilde{y}'$ is reduced to a table lookup and an addition:

$$\tilde{y}' = \tilde{y}_\text{raw} + \tilde{C}_{i,\,\tilde{z}}^{y'}\,.$$

As the $\tilde{y}'$ values are not required for the initial combination of the track segments to tracks, they together with the electron probabilities $\tilde{P}$ are stored in memories for later use in the reconstruction algorithm.

[5] At a resolution of 0.005 rad, the simplified formula results in 81 % of the cases in the exact same value, in the other cases the difference amounts to ±1 bit.

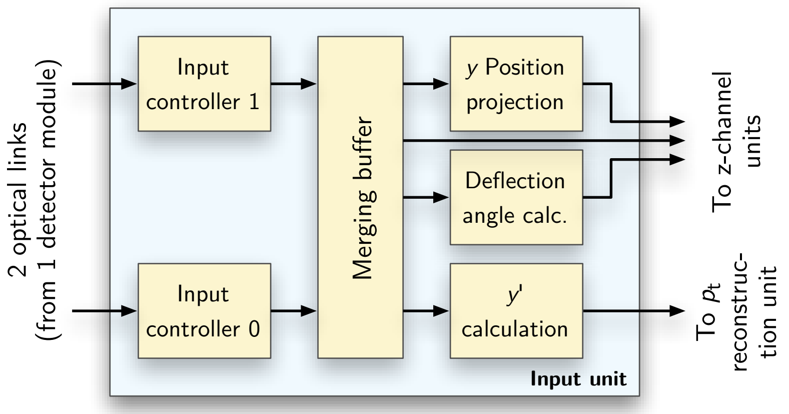


Figure 7.6: The architecture of an input unit as block diagram. The input unit receives the track segment data from a detector module and performs the required calculations that can be carried out independently for each segment.

7.2.4 Architecture of an Input Unit

The computations that are required for each track segment can be performed in an input unit as shown in Figure 7.6. The input controllers reassemble the data from the readout network to 32-bit words. They also check for the 16-bit end word 0x1000 (see Figure 7.1) indicating the end of the transmission and ignore spurious data words.[6] In the merging buffer, the 32-bit words of the two input controllers are buffered in FIFO[7] memories and merged into a single sorted data stream. Finally, dedicated units perform the presented arithmetic operations.

7.3 Matching by Selecting: The Z-Channels

Similar to the y coordinates of the track segments, the z coordinates could be projected onto the virtual center plane, finding matches by regarding the distances between the projected z coordinates and comparing them to the width of the matching window. As all segments would have to be compared in pairs, this would result in considerable processing time. However, as only 16 different possibilities exist for the number of the pad row, computations can be parallelized at this point. The technique explained in this section reduces the complexity of matching the track segments by one degree of freedom at a low cost in terms of latency.

[6]The network interface internally treats the data words in each case as two 16-bit words, transmitting the less significant half-word first. The completion of the transfer is marked by the network interface by a 16 bit wide final word, which is freely configurable. It should be defined in such a way that it corresponds to a value impossible in the application if interpreted as the lower half of a data word. The GTU examines the indicated bits of the final word only.

[7]FIFO: First In, First Out

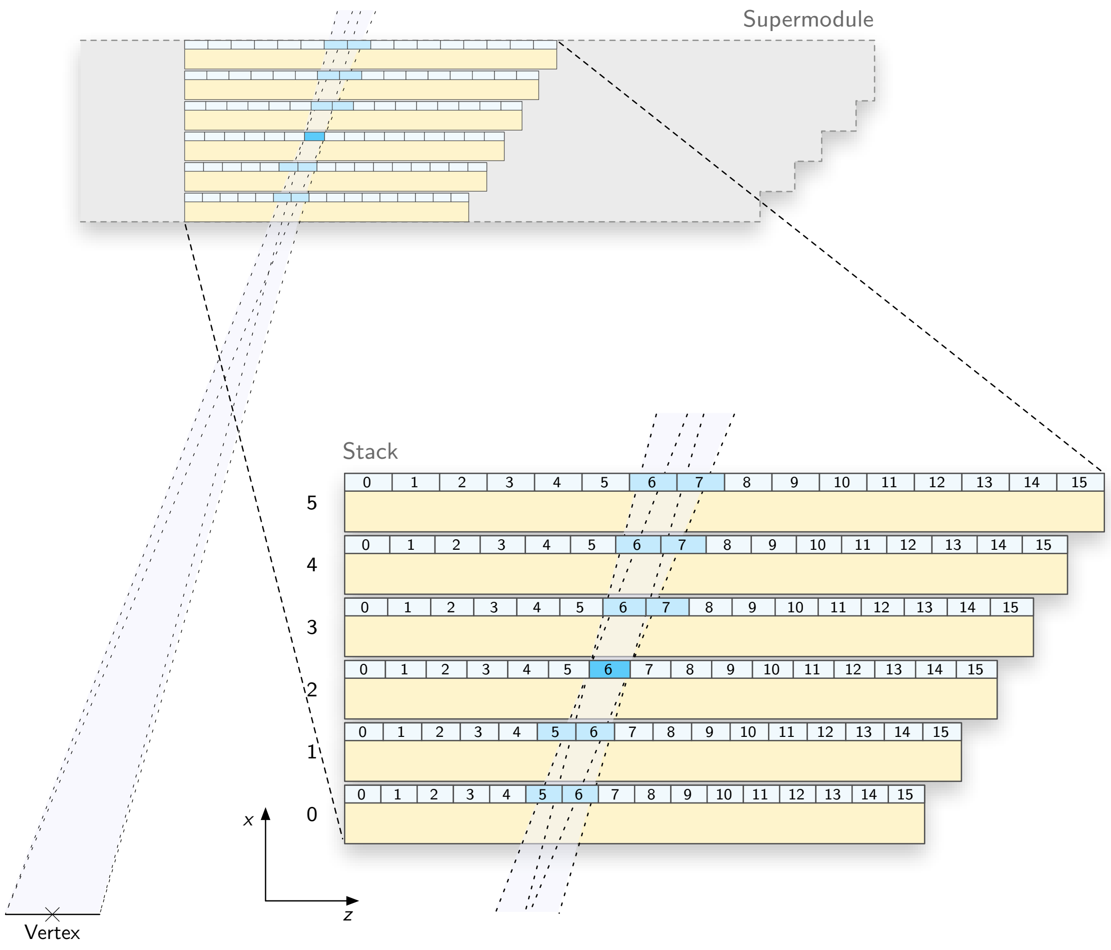


Figure 7.7: Cross section of a module stack. A particle track can traverse the detector modules of different layers in different pad rows. The marked area indicates exemplarily the pad rows that a particle from the vertex crossing layer 2 in pad row 6 can reach.

Instead of projecting the z coordinates, the window itself is projected onto the six layers of the detector modules and compared directly to the z coordinates. As originating plane for the projection not the exact virtual center plane is used, but one of the existing detector layers. Selecting layer 2 results in an efficient implementation.[8]

Figure 7.7 illustrates the principle of backward projection using an example. A particle crossing layer 2 in pad row 6 (marked in blue) can — if it originates from the primary vertex — only cross certain pad rows in the other layers because of the detector geometry. The projection respects an uncertainty of $\pm 20\,\text{cm}$ in the position of the primary vertex (point of collision) in z direction. By considering only those track segments that are located in the pad rows marked blue in the figure, comparing the z coordinates of these

[8]The selection of the layer is basically arbitrary. However, using one of the two middle layers as origin layer is advisable since this minimizes the overlap of the projections. As the following will show, this allows for a particularly efficient implementation.

segments is no longer necessary. It is already clear by the selection of the track segments that they sufficiently agree in their projected z coordinates. As only tracks crossing layer 2 in this particular pad row[9] are found by this selection, the selection procedure is performed in parallel for all 16 pad rows of the origin layer to find all tracks.

It is, however, not necessary to implement 16 completely independent units. Considering the detector geometry and the assumed uncertainty of the vertex position, the projection range in the other layers does not exceed three pad rows, and the projection range of pad rows three units apart does not overlap. Since the track segments arrive sorted in z direction at the GTU, the same unit can process at first the projection of the first pad row, and then the fourth, seventh (etc.). Instead of 16 independent units, it is sufficient to use three units, each dealing with the data of every third projection range.

The data of the three units are hereinafter referred to as *z-channels* 0, 1 and 2. The z-channel j contains the track segments from the projection range of the pad rows k with $j \equiv k \pmod 3$. Within a z-channel, the *index* identifies the number of the projection (subchannel).

As the data of the three z-channels is processed completely independently in parallel and within a z-channel, only track segments with the same index can be merged to tracks, the agreement with respect to the z position is guaranteed and does not need to be considered in the remaining computations. While this approach triples the remaining merging logic, it significantly reduces the complexity of the problem, as the track segments have to be compared directly only in two dimensions.

7.3.1 Architecture of a Z-Channel Unit

Transforming this selection principle into a hardware architecture, the mapping of segments to z-channels and indices in a stack can be precomputed and stored in lookup tables (LUTs), as it depends only on the detector geometry. The order of track segments in the data stream requires additional attention. As a consequence of the projection onto the center plane and the merging of segments from up to three pad rows, the stream of track segments, which is originally sorted according to the raw y coordinate within a pad row, is no longer sorted correctly. The employed matching algorithm, however, requires sorted track segments to avoid having to compare all segments with each other.

The architecture of a z-channel unit is shown in Figure 7.8. The altogether $6 \cdot 3 = 18$ z-channel units each process one of the three z-channels for one detector layer. By comparison of the $\tilde{z}$-value with a LUT, it is first decided for each track segment whether and — if so — with which subchannel index it belongs to this z-channel. Track segments with the same index are sorted afterwards in a parallel sorter according to their projected y coordinate.

The sorter is specifically optimized for minimum latency. After the last track segment belonging to an index has arrived, output of the smallest element starts already in the

[9]However, it is insignificant whether a contributing track segment is actually found in this layer.

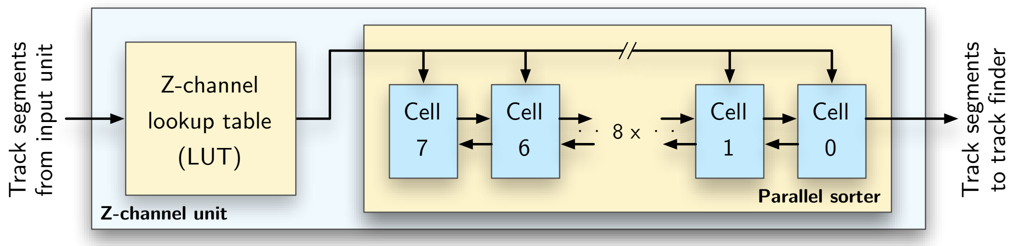


Figure 7.8: The architecture of a z-channel unit as a block diagram. The unit receives all track segment data of one layer. Using a lookup table (LUT), the segments for this z-channel are selected and assigned to the subchannels via an index. The segments of each subchannel are sorted according to their projected y coordinate and passed on to the track finder units.

following clock cycle. In order to reach this, it is built according to Figure 7.8 from parallel flip-flop-based sorter cells. Each cell stores one track segment data word. Depending on the constellation, the individual values in the sorter are shifted an adjacent cell or keep their position, so that the segments of the cells are always sorted in ascending order. Independently of its current state, the sorter can accept a data word each clock cycle. New entries are compared simultaneously in parallel to all existing entries and inserted directly at the correct position.

7.4 Finding Tracks in Streams of Track Segments

For efficiency reasons, it is important not to compare all available track segments in pairs when trying to find matching combinations. The solution presented here drastically limits the number of pair comparisons by regarding in a sorted stream of track segments only a small number of most relevant segments at a time. While finding tracks in the presented algorithm is still a sequential procedure, its run-time is in most practical cases mainly linearly dependent on the total number of track segments (see Chapter 9 for a detailed analysis) and thus comparable to other pipelined parts of the trigger design.

A track finder unit handles the track segments of one z-channel over all layers and merges them to tracks. To this end, it first compares the $\tilde{y}_{\text{proj}}$ value and, in case of agreement, additionally the value of the angle $\tilde{\alpha}$.

Using a Reference Layer

The principle of efficient sequential track finding is that not all track segments are compared in pairs, but from each layer, only a small part of the values sorted according to the projected y coordinate is regarded, which is successively shifted to larger values. The complexity of the procedure is reduced considerably if one of the detector layers is defined as a reference layer with whose track segments the segments of the other layers are compared. However, this limits track finding to tracks that contain a contributing track segment in the reference layer. The restriction can be circumvented by

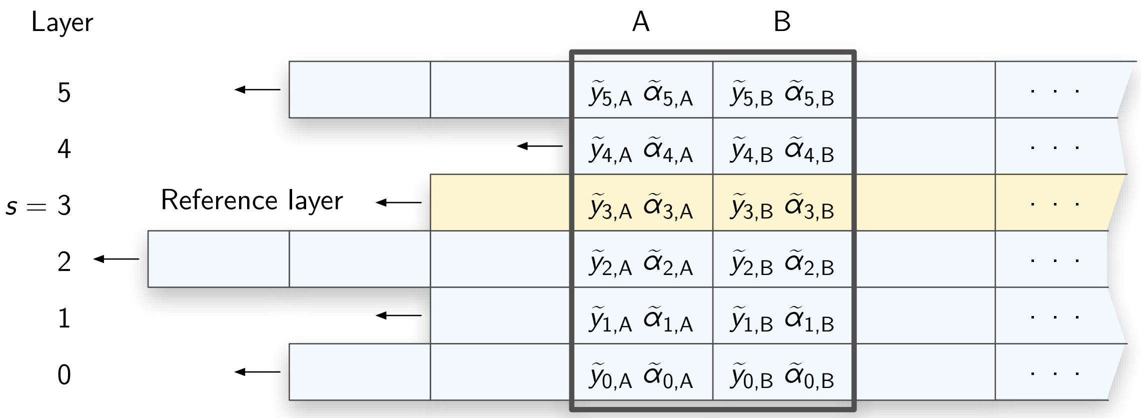


Figure 7.9: The track finder unit regards two successive track segments from each layer. It aligns the data of the different layers in such a way that track segments belonging together are positioned together in the field of vision.

instantiating the entire track finder unit triply in parallel, each using a different reference layer. This ensures that tracks with segments in at least four of the six layers are found in at least one of the units.

Size of the Comparison Range Two tracks coincidentally intersecting the projection plane under different angles at the same point may overlap in the track segment data stream, which is sorted only according to y_{proj}. Thus, it is not sufficient to regard in each layer only the track segment that best fits in y_{proj} when compared to the reference segment on the reference layer. To correctly assign a track segment through its angle even if another segment is coincidentally also matching in y_{proj}, several consecutive track segments have to be compared simultaneously. However, as there are rarely more than two segments inside the y_{proj} window (see [23] for a detailed analysis) it is sufficient in case of the TRD to limit the search for matching track segments to examining only the first two segments matching in y_{proj} with respect to agreement in the deflection angle.

Figure 7.9 illustrates the track finding algorithm. The rows containing the sorted data sets of the six layers are aligned in such a way that track segments belonging together are placed together in the field of vision. The reference layer s is marked in color.

7.4.1 Comparing Track Segments

At this stage, track matching is based exclusively on the integer values of the projected y coordinates $\tilde{y}_{\text{proj}}$ and the deflection angles $\tilde{\alpha}$ of the track segments at the two positions A and B of the viewing area in each of the six layers, which are denoted by:[10]

$$\tilde{y}_{i,\text{A}}, \tilde{y}_{i,\text{B}} \quad \text{and} \quad \tilde{\alpha}_{i,\text{A}}, \tilde{\alpha}_{i,\text{B}} \qquad (\text{for } 0 \leq i < 6)\,.$$

[10]The z-channel indices are not explicitly mentioned in the following. A difference in the subchannel index is handled like a large difference in $\tilde{y}_{\text{proj}}$.

Quantity	Resolution	Max. deviation	Factor	Notation
y_{proj}	$1/1.28\,\text{mm}$	$11.625\,\text{mm}$	9	$\Delta\tilde{y}_{\text{proj}}$
α	$1/4.\bar{6}\,\text{mrad}$	$0.05\,\text{rad}$	11	$\Delta\tilde{\alpha}$

Table 7.1: The numerical values for the GTU window criterion. The resolution of the values is a result of the optimization for the hardware implementation.

To efficiently find matching track segments, the limits of the matching window are computed based on the segments on the reference layer and compared in parallel to the segment positions on the other layers. From a track segment A on the reference layer s, using the configurable window sizes $\Delta\tilde{y}_{\text{proj}}$ and $\Delta\tilde{\alpha}$ (numerical values shown in Table 7.1[11]), the window range follows as:

$$\tilde{y}^{+} = \tilde{y}_{s,\text{A}} + \Delta\tilde{y}_{\text{proj}}\,, \qquad \tilde{\alpha}^{+} = \tilde{\alpha}_{s,\text{A}} + \Delta\tilde{\alpha}\,,$$
$$\tilde{y}^{-} = \tilde{y}_{s,\text{A}} - \Delta\tilde{y}_{\text{proj}}\,, \qquad \tilde{\alpha}^{-} = \tilde{\alpha}_{s,\text{A}} - \Delta\tilde{\alpha}\,.$$

From this, a set of Boolean variables can be derived, indicating for each layer i whether the track segment A or B can be combined to a track with the reference segment:

$$b_{\text{hit A},\,i} := (\tilde{y}^{-} < \tilde{y}_{i,\text{A}} < \tilde{y}^{+}) \wedge (\tilde{\alpha}^{-} < \tilde{\alpha}_{i,\text{A}} < \tilde{\alpha}^{+})\,,$$
$$b_{\text{hit B},\,i} := (\tilde{y}^{-} < \tilde{y}_{i,\text{B}} < \tilde{y}^{+}) \wedge (\tilde{\alpha}^{-} < \tilde{\alpha}_{i,\text{B}} < \tilde{\alpha}^{+})\,.$$

If both variables are false for a given layer, this does not automatically mean that no suitable segment exists in this layer, because the wrong section of the layer could have been considered. To decide whether a layer i is *aligned* relative to the reference layer, meaning the viewing area is already sufficiently advanced for an assessment, the Boolean variable

$$b_{\text{aligned},\,i} := (\tilde{y}^{-} < \tilde{y}_{i,\text{A}}) \vee (\tilde{y}^{+} < \tilde{y}_{i,\text{B}})$$

is regarded. It is true if the projected y coordinate of segment A is not below the window range or if the projected y coordinate of segment B is already above the window range, as in this case shifting the layer cannot move matching track segments into the field of vision.

A track is considered to be found if:

$$N_{\text{hits}} \geq 4 \wedge N_{\text{uncertain}} = 0$$

where N_{hits} indicates the number of definite agreements and $N_{\text{uncertain}}$ counts the layers for which a reliable statement is not yet possible[12]:

$$N_{\text{hits}} = \sum_{0 \leq i < 6} \mathbb{1}_{\{b_{\text{aligned},\,i} \wedge (b_{\text{hit A},\,i} \vee b_{\text{hit B},\,i})\}}$$
$$N_{\text{uncertain}} = \sum_{0 \leq i < 6} \mathbb{1}_{\{\neg b_{\text{aligned},\,i}\}}\,.$$

[11]These values are somewhat larger than those in [8, p. 100]. They are chosen on the basis of detector simulations [81] and have proven appropriate using data from beam tests.

[12]The symbol $\mathbb{1}$ denotes the indicator function, which is one if its argument is true, and zero otherwise.

7.4.2 Incrementing the Positions in the Data Streams

In each clock cycle, it has to be decided for each layer how far this data row may be shifted with respect to the field of vision, i. e., by what number its position may be incremented. Herein, the reference layer takes a special role. Its data row is usually shifted by one position per clock cycle. However, it should not be shifted if it is unclear whether a track including the current segment at position A can be found or not. Thus, the position in the reference row is kept if one of the other rows is not yet aligned accordingly and at the same time a possibility of finding a suitable track still exists, i. e.:

$$(N_{\text{uncertain}} \neq 0) \wedge (N_{\text{uncertain}} + N_{\text{hits}} \geq 4)\,. \tag{7.11}$$

Otherwise, the reference row can be shifted by one position[13], so that in the next clock cycle tracks matching the following segment can be found.

Depending on the behavior of the reference row and the results of the coordinate value comparisons, a combinational logic decides on the shifting of the other rows. If the reference layer is shifted, the increment of the other rows results from a comparison with segment B of the reference layer, which becomes the reference segment in the following cycle. The increment is selected as high as possible (with a maximum value of two) without losing possible matches. If the reference row is not moved, the remaining rows are shifted until they are aligned to match the current reference segment, or even further, if they contain no matching segment. This procedure continues until enough layers reach their last segment, rendering another match impossible.

In each clock cycle and for each layer, the selection process aims to select the maximum possible increment of the window position to minimize the total latency of the unit. Thanks to the possibility of shifting each row by two positions, the latency of the track matching is low (see Section 9.3.1 for an analysis of the timing performance).

7.4.3 Architecture of a Track Finder

A track finder unit implementing the presented algorithm essentially consists of memories for the data rows, read address registers, and a purely combinational logic, which compares the values read from memory. The logic determines whether the track segments together form a track and specifies the increment of the address counter registers in each clock cycle. Track segment data is accepted in parallel for all six layers directly from the sorters of the corresponding z-channel units. It is stored in two memories per layer to allow for regarding two successive data items from each layer at the same time. With the exception of the more complicated start and end conditions, the memory read addresses correspond to the position of the regarded range (see Figure 7.9). In each clock cycle, the viewing area can be shifted by one or even two positions or stay unchanged. If a track is found,

[13] In special cases, the reference row can even be shifted by two positions. These cases are determined by additional logic comparing the segments of the other layers with segment B of the reference layer.

the addresses of the contributing track segments together with the mask of contributing layers are passed on to the next unit.

The combination logic is a complex combinational part of the tracking design. For the parallel comparisons, each unit requires a total of 39 10-bit adders, 22 7-bit adders and 35 3-bit adders. In terms of timing, it represents the critical path of the design, as the next memory address is computed based on the current memory output by several stages of logic containing arithmetics. However, in the utilized FPGA components, it operates at a clock rate of 60 MHz.

7.5 Removing Duplicate Tracks

The distribution of the data into z-channels and the parallel track search using different reference layers can cause the same track to be detected multiple times in the different parallel units. To fix this disadvantage of the parallelization, the data streams of the nine track finder units have to be merged, guaranteeing that each track is passed on only once. In the hardware implementation, the merging unit fulfills this task.

In order to not have to compare all tracks in pairs, the principle of presorting is applied once again. In several merging stages, the arriving data streams are buffered and merged in such a way that identical tracks follow one another. The so-prepared data streams pass uniqueness modules, which recognize identical tracks in the sequence and reject duplicate occurrences. First, the tracks that are found based on different reference layers are united within the individual z-channels. The next step is to merge the tracks from the three z-channels. Here, a two-stage procedure is necessary, since by simply merging the streams no order can be accomplished in which identical tracks always follow one another. Finally, the resulting data stream of unique tracks is forwarded to the reconstruction unit for analysis.

8 Transverse Momentum Reconstruction and Trigger Decision

After the track segments have been combined to tracks in the GTU, the transverse momentum p_t of the producing particle is to be reconstructed from the trajectory. The necessary calculations and their hardware implementation are presented in this chapter.

In the x-y plane, the track segments produced by a particle are positioned on a circular arc together with the primary interaction point. From the radius of the arc, the transverse momentum can be computed. Of the various methods of fitting a circular path to a set of data points (see for example [29]), most are not suitable for use in a low-latency trigger calculation because of their high cost of computation.

A method inexpensive enough in principle to be used in a trigger application is *conformal mapping*. Via inversion in the complex plane ($u = \frac{x}{x^2+y^2}, v = \frac{y}{x^2+y^2}$), a circle through the origin is mapped to a straight line. The impact parameter of the line is then inversely proportional to the radius of the circle. However, even this method is comparatively expensive to compute because of the necessary divisions.

On the other hand, the radii of the circular paths of the target particles are so large that even on GTU level the tracks appear approximately as straight lines. As simulations show, it is indeed sufficient to fit a straight line to the track segments as illustrated in Figure 8.1 and regard the line as a secant of the circular arc. From the resulting line parameters a and b, the transverse momentum of the original particle can be calculated. The following sections discuss the required computations and their optimization for a low-latency hardware implementation.

8.1 Fitting a Straight Line to the Track Segments

To fit a straight line to the track segments, a suitable parametrization of the track is required. Without *tilted pads* (see Section 6.2.2) the deflection in z direction would not matter, and the track could be parametrized by

$$y_i = a + b \cdot x_i \tag{8.1}$$

with

y_i : y coordinate of the track segment in layer i

x_i : x coordinate at the outer side of the drift chamber in layer i

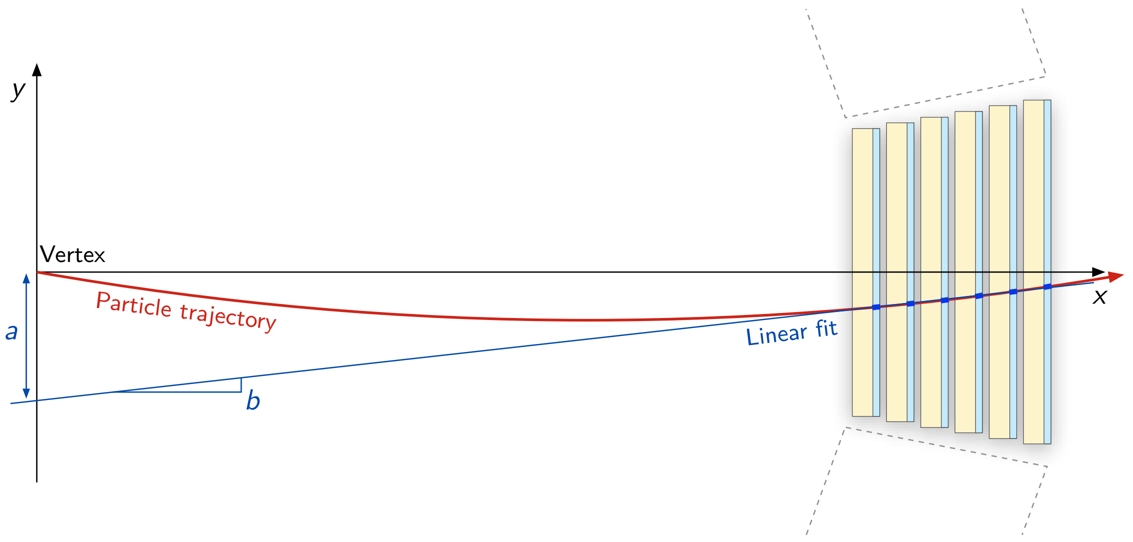


Figure 8.1: Fitting a straight line through the track segments. In this drawing, the relevant distances are true to scale.

and the line parameters a and b.

With *tilted pads*, however, the true value y_i is not known, as it is no longer equal to the measured value $y_{\mathrm{raw},\,i}$. Instead, it depends according to Equation (6.2) in Section 6.2.2 slightly on the z position:

$$y_i = y_{\mathrm{raw},\,i} + (-1)^i \cdot (z_i - z_{i,\,\mathrm{row}}) \cdot \tan(\beta_{\mathrm{tilt}}) \tag{8.2}$$

Here, z_i is the z position, which is also not directly measurable, and $z_{i,\,\mathrm{row}}$ is the (known) z coordinate of the center of the pad row. This results in the parametrization

$$y_{\mathrm{raw},\,i} = a + b \cdot x_i - (-1)^i \cdot (z_i - z_{i,\,\mathrm{row}}) \cdot \tan(\beta_{\mathrm{tilt}}). \tag{8.3}$$

Assuming that, in the x-z plane, the particles come from the origin, the following applies for a constant c:

$$z_i = c \cdot x_i$$

Defining the variables

$$\begin{aligned} y'_i &:= y_{\mathrm{raw},\,i} - (-1)^i \cdot z_{i,\,\mathrm{row}} \cdot \tan(\beta_{\mathrm{tilt}}) \\ c' &:= c \cdot \tan(\beta_{\mathrm{tilt}}), \end{aligned} \tag{8.4}$$

where the y'_i values are already computed in the input units for all incoming track segments, results in a convenient form of the track parametrization:

$$y'_i = a + b \cdot x_i - c' \cdot (-1)^i \cdot x_i \tag{8.5}$$

The corresponding least squares optimization problem is to find

$$\min_{(a,b,c')\in\mathbb{R}^3} \sum_{i\in\mathfrak{I}} \Big(y_{\mathrm{raw},\,i} - (a + b \cdot x_i - c' \cdot (-1)^i \cdot x_i)\Big)^2 \tag{8.6}$$

The set $\mathfrak{I}$ herein contains the indices $0 \leq i < 6$ of the detector layers from which a track segment contributes to the track. As at least four of the six layers are required, it applies $4 \leq |\mathfrak{I}| \leq 6$. The composition of the index set $\mathfrak{I}$ can be coded by a number k in such a way that a 1 at the nth position of the binary notation of k means that a track segment is present at layer n. Thus, for $k = 23 = 010111_2$, for example, it follows $\mathfrak{I}_{23} = (0, 1, 2, 4)$. For a valid track candidate, at least four of the six track segments are demanded. It follows by combinatorial considerations that there are 22 valid values for k or possible sets $\mathfrak{I}_k$.

Expression (8.6) shows that as a consequence of the *tilted pads*, a three-dimensional linear regression has to be calculated instead of a two-dimensional regression. The parameter c' is needed to be able to determine a and b exactly; however, its value is not used in further computations. Rewriting the optimization problem in terms of column vectors results in:

$$\min_{\mathbf{p}\in\mathbb{R}^3} \|\mathbf{y}' - \mathbf{F}_{\mathfrak{I}}\mathbf{p}\|_2 \tag{8.7}$$

where the vector $\mathbf{y}'$ contains the y'_i of all six layers and $\mathbf{p}$ contains the three fit parameters:

$$\mathbf{y}' = \begin{pmatrix} y'_0 \\ y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \\ y'_5 \end{pmatrix}, \qquad \mathbf{p} = \begin{pmatrix} a \\ b \\ c' \end{pmatrix}. \tag{8.8}$$

$\mathbf{F}_{\mathfrak{I}}$ is a matrix whose columns contain the basis functions. If segments from all layers contribute to the track, it is given by

$$\mathbf{F}_{\{0,\ldots,5\}} = \begin{pmatrix} 1 & x_0 & -x_0 \\ 1 & x_1 & x_1 \\ 1 & x_2 & -x_2 \\ 1 & x_3 & x_3 \\ 1 & x_4 & -x_4 \\ 1 & x_5 & x_5 \end{pmatrix}. \tag{8.9}$$

In the other cases, $\mathbf{F}_{\mathfrak{I}}$ is obtained from $\mathbf{F}_{\{0,\ldots,5\}}$ by replacing the values in rows corresponding to a nonexistent index in $\mathfrak{I}$ by zeroes.

The optimization problem (8.7) is solved by applying the orthogonality principle, which states that the error vector for optimal $\mathbf{p}$ should be perpendicular to all of the basis vectors [73]. This can be expressed directly in terms of the matrix $\mathbf{F}_{\mathfrak{I}}$:

$$\mathbf{F}_{\mathfrak{I}}^{\mathrm{T}} * (\mathbf{y}' - \mathbf{F}_{\mathfrak{I}}\mathbf{p}) = 0 \tag{8.10}$$

Solving for $\mathbf{p}$ gives:

$$\mathbf{p}_{\text{opt}} = \underbrace{(\mathbf{F}_{\mathfrak{I}}^{\mathrm{T}}\mathbf{F}_{\mathfrak{I}})^{-1}\mathbf{F}_{\mathfrak{I}}^{\mathrm{T}}}_{=:\mathbf{F}_{\mathfrak{I}}^{\dagger}}\mathbf{y}' = \mathbf{F}_{\mathfrak{I}}^{\dagger}\mathbf{y}' \tag{8.11}$$

The square matrix $(\mathbf{F}_{\mathfrak{I}}^{\mathrm{T}}\mathbf{F}_{\mathfrak{I}})$ is invertible if (and only if) $\mathbf{F}_{\mathfrak{I}}$ is full-rank, which is equivalent to saying that the basis vectors are linearly independent. For the given $\mathbf{F}_{\mathfrak{I}}$ with $|\mathfrak{I}| \geq 4$, this is always the case.

The matrix $\mathbf{F}_{\mathfrak{I}}^{\dagger} = (\mathbf{F}_{\mathfrak{I}}^{\mathrm{T}}\mathbf{F}_{\mathfrak{I}})^{-1}\mathbf{F}_{\mathfrak{I}}^{\mathrm{T}}$ is known as the pseudo-inverse of the matrix $\mathbf{F}_{\mathfrak{I}}$. According to Equation (8.9), it depends solely on the x coordinates of the contributing detector layers, which are fixed by the detector geometry. Thus, the entries

$$\mathbf{F}_{\mathfrak{I}_k}^{\dagger} = \begin{pmatrix} a_{k0} & a_{k1} & a_{k2} & a_{k3} & a_{k4} & a_{k5} \\ b_{k0} & b_{k1} & b_{k2} & b_{k3} & b_{k4} & b_{k5} \\ c'_{k0} & c'_{k1} & c'_{k2} & c'_{k3} & c'_{k4} & c'_{k5} \end{pmatrix} \tag{8.12}$$

of $\mathbf{F}_{\mathfrak{I}_k}^{\dagger}$ are dependent only on the set of indices $\mathfrak{I}_k$ and can therefore be precalculated and tabulated for all values of k.

Substituting (8.12) into (8.11) results in

$$a = \sum_{i=0}^{5} a_{ki} y'_i, \qquad b = \sum_{i=0}^{5} b_{ki} y'_i. \tag{8.13}$$

For the 22 valid values of k, the coefficients a_{ki} and b_{ki} can be stored in lookup tables in the hardware design. This way, the values for the coefficients a and b can both be computed from y'_i according to (8.13) using only six multiplications and five additions.

8.1.1 Intersections of Straight Line and Circular Path

A source of inaccuracy in this procedure is that the fitted straight line is neither accurately a tangent to the circular particle track nor a secant with known points of intersection. Rather, it lies between these two lines (see Figure 8.2). To precisely reconstruct the circle it would be necessary to know the x coordinates of the intersections between best fit straight line and circular path. The correct choice of these points depends not only on the positions of the layers with track segments contributing to the calculation but also on the radius of the circular path. The points can therefore not be determined trivially. In simulation, good results are achieved using the two approximations

$$x_{\mathrm{A}} = x_{i_{\mathrm{inner}}} + x_{\mathrm{d}} \cdot \frac{n_{\mathrm{Hits}} - 1}{6} \qquad x_{\mathrm{B}} = x_{i_{\mathrm{outer}}} - x_{\mathrm{d}} \cdot \frac{n_{\mathrm{Hits}} - 1}{6} \tag{8.14}$$

with

i_{inner} : detector layer of the innermost contributing track segment
i_{outer} : detector layer of the outermost contributing track segment
n_{Hits} : number of contributing track segments
x_i : x coordinate at the outer side of the drift chamber in layer i
x_{d} : distance between the outer sides of two drift chambers

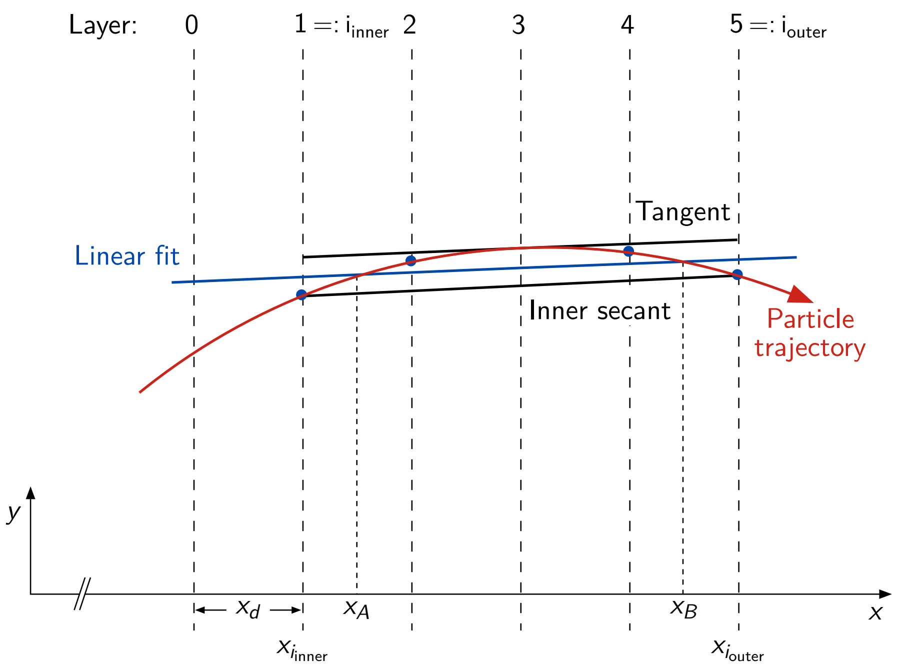


Figure 8.2: The best fit straight line to the track segments lies between tangent and secant. The correct intersections x_A, x_B cannot be determined in a trivial way. For illustration, the curvature of the particle is strongly exaggerated.

Since the radius of the particle trajectory is large, the quantitative influence of the choice of x_A and x_B is limited. By using the specified approximation instead of the even simpler estimation $x_\mathrm{A} = x_{i_\mathrm{inner}}, x_\mathrm{B} = x_{i_\mathrm{outer}}$, the resolution in p_t is improved by 1.0 %, and the systematic deviation of the average value of the error distribution from zero is reduced by 0.022 GeV/c (27 %).

8.2 Computation of the Track Radius

From the three points $(0, 0)$, $(x_\mathrm{A}, y_\mathrm{A})$, and $(x_\mathrm{B}, y_\mathrm{B})$ the radius of the circular path can be determined. From Figure 8.3 (upper right-angled triangle) it follows that:

$$r = \frac{d_\mathrm{AB}/2}{\sin(\alpha)} \tag{8.15}$$

with

$$\alpha = \varphi_\mathrm{B} - \varphi_\mathrm{A} = \arctan\left(\frac{y_\mathrm{B}}{x_\mathrm{B}}\right) - \arctan\left(\frac{y_\mathrm{A}}{x_\mathrm{A}}\right) \tag{8.16}$$

and

$$\begin{aligned} d_\mathrm{AB} &= \sqrt{(x_\mathrm{B} - x_\mathrm{A})^2 + (y_\mathrm{B} - y_\mathrm{A})^2} \\ y_\mathrm{A} = a + b \cdot x_\mathrm{A} &\qquad y_\mathrm{B} = a + b \cdot x_\mathrm{B} \end{aligned} \tag{8.17}$$

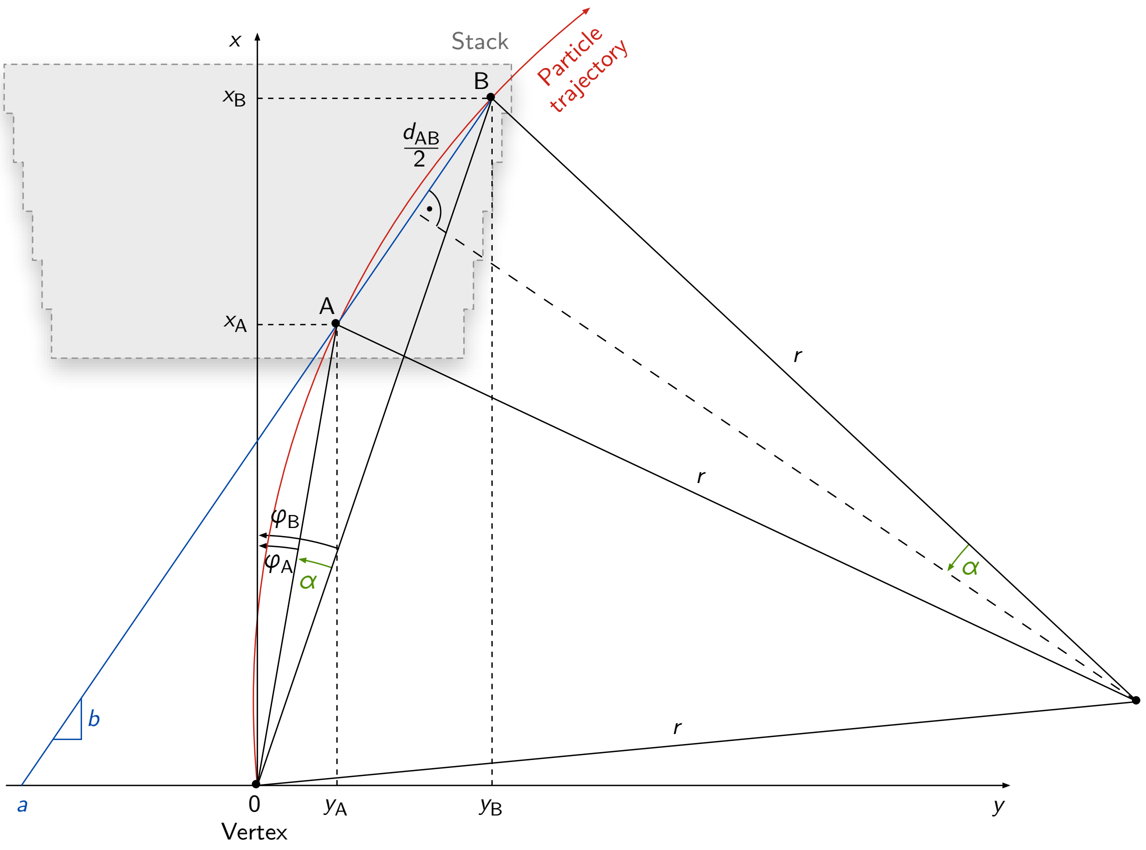


Figure 8.3: Illustration of the computation of the transverse momentum on GTU level.

Using these equations, the radius of the particle track can be reconstructed from the two coefficients a and b. Once the track radius r is reconstructed, the transverse momentum p_t of the particle results according to (6.7) from multiplying by a constant.

8.2.1 Optimizing the Computation for Online Processing

The computation of the track radius using the presented equations is still expensive for a hardware implementation with the existing time limits. It is therefore necessary to investigate the extent to which the calculations can be simplified by using approximations without causing the result to deteriorate significantly.

Using (8.17), the distance d_{AB} can be written as

$$d_{AB} = \sqrt{1+b^2} \cdot (x_B - x_A)$$

There is almost no loss of accuracy if the root is expanded according to

$$\sqrt{1+x} \approx 1 + \frac{1}{2}x \qquad \text{(for small } x\text{).}$$

Since the angle α is small in the experiment, the estimation

$$\sin(\alpha) \approx \alpha \qquad \text{(for small } \alpha\text{)}$$

also causes only minor loss. The calculation now reads:

$$d'_{\mathrm{AB}} = (x_{\mathrm{B}} - x_{\mathrm{A}}) \left(1 + \frac{b^2}{2}\right) \tag{8.18}$$

$$\varphi_{\mathrm{A}} = \arctan\left(\frac{a}{x_{\mathrm{A}}} + b\right) \qquad \varphi_{\mathrm{B}} = \arctan\left(\frac{a}{x_{\mathrm{B}}} + b\right) \tag{8.19}$$

$$r' = \frac{d'_{\mathrm{AB}}}{2 \cdot (\varphi_{\mathrm{B}} - \varphi_{\mathrm{A}})} \tag{8.20}$$

Up to this point no deterioration of the results is recognized in a simulation based on randomly generated events.

The estimation $\arctan(x) \approx x$, which is valid for small x, would lead to a radical simplification; however, it produces significantly poorer results. Expanding instead $\alpha = \varphi_{\mathrm{B}} - \varphi_{\mathrm{A}}$ as a Taylor series leads to:

$$\alpha = \varphi_{\mathrm{B}} - \varphi_{\mathrm{A}} = a(b^2 - 1) \cdot \frac{x_{\mathrm{B}} - x_{\mathrm{A}}}{x_{\mathrm{A}} x_{\mathrm{B}}} + a^2 b \cdot \frac{x_{\mathrm{B}}^2 - x_{\mathrm{A}}^2}{x_{\mathrm{A}}^2 x_{\mathrm{B}}^2} + \ldots$$

Rearranging results in a simplified calculation method:

$$c_1 = \frac{x_{\mathrm{A}} x_{\mathrm{B}}}{2} \qquad c_2 = \frac{x_{\mathrm{A}} + x_{\mathrm{B}}}{x_{\mathrm{A}} x_{\mathrm{B}}} \tag{8.21}$$

$$r'' = c_1 \cdot \frac{\frac{b^2}{2} + 1}{a(b^2 - 1) + a^2 b c_2} \tag{8.22}$$

The constants c_1 and c_2 are dependent only on the distribution of segments on the layers, allowing the 22 possible values c_{1k} and c_{2k} to be tabulated. Thus, Equation (8.22) reduces the computation of the track radius to the basic arithmetic operations addition and multiplication as well as a single division.

Nevertheless, as the values of the slope b are small, an even more far-reaching simplification is justified. Neglecting the slope completely, a simple relation results:

$$r''' = \lim_{b \to 0} r'' = -\frac{c_1}{a} \tag{8.23}$$

This allows the value tables for b_{ki} and c_{2k} to be completely omitted; the complete calculation is reduced to few table operations beside the multiplications for the determination of a from the y_i values. If the radius is only to be compared with a threshold value, then even the division can be omitted.

A hardware-oriented simulation using randomly generated events shows that even this drastic simplification worsens the p_{t} resolution on average only by 5 % from 0.132 GeV/c to 0.138 GeV/c. There is, however, an additional systematic bias of the average error from −0.073 GeV/c to −0.125 GeV/c, which can be compensated on the average by adding a constant. In view of the trigger application of the GTU, this result is perfectly acceptable. Therefore, this algorithm is selected for the actual hardware implementation.

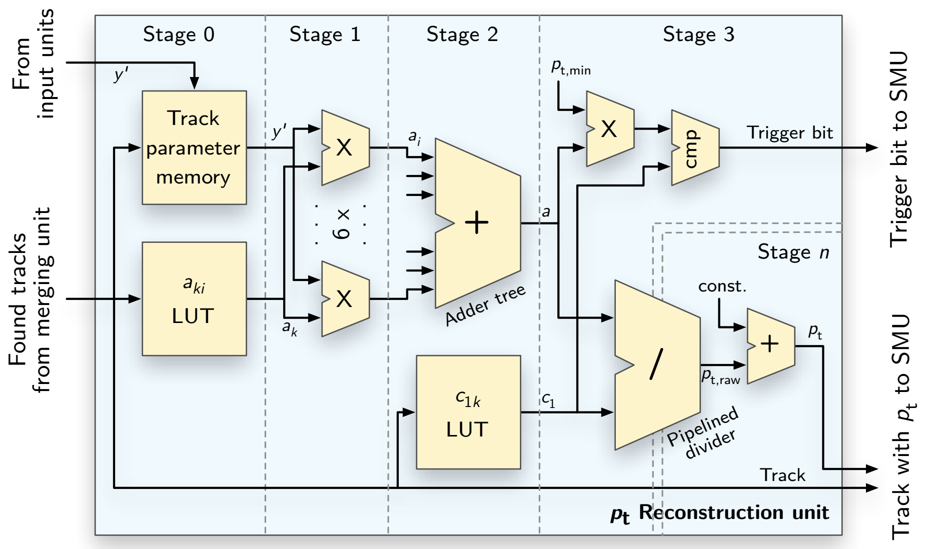


Figure 8.4: The architecture of the p_t reconstruction unit. For a track data set, the appropriate reconstruction parameters are retrieved from the input units. The parameter a is calculated, which corresponds to the axis intercept of a best fit straight line to the track segments. From the values c_1 and a, the transverse momentum is computed by division. If the exact numerical value is not required, a fast trigger signal bypassing the time-consuming division is available in pipeline stage 3.

8.3 Transverse Momentum Reconstruction Architecture

In the p_t reconstruction unit, the computations presented in the previous sections are implemented as a combination of basic arithmetic operations and table accesses. The accuracies and bit widths of the signals are selected such that according to simulations no deterioration of the results is detectable. Details of the hardware implementation are presented in [23].

The architecture of the p_t reconstruction unit is illustrated in Figure 8.4. It is a fully pipelined data push architecture optimized for low latency, allowing for a new track to be accepted in every clock cycle. The unit is operating in the TMU's FPGA at a clock rate of 60 MHz. In the first three pipeline stages, the line parameter a is calculated, which corresponds to the axis intercept of a best fit straight line to the track segments. Together with the value of the constant c_1, which depends exclusively on the combination of contributing detector layers, this is sufficient to determine p_t according to the Equations (8.23) and (6.7).[1]

The number of remaining pipeline stages is configurable and depends on the form of the

[1]The electron probability P of the segments is not yet fully processed in the presented design because the details of the approach to particle identification are currently still under investigation. Several strategies of electron/pion separation in the TRD as well as first test results are discussed in [82]. A straightforward solution, which is implemented in the present design, is to add the probabilities of the contributing segments and to compare the result to a predetermined threshold value.

desired result. If the resulting p_t of the reconstructed track is only to be compared to a configurable threshold value, the expensive division in Equation (8.23) is expendable. Instead, the value of a can be multiplied by the threshold value and the result can be compared to c_1 directly. In this case, an intermediate trigger signal depending on the sign of the track curvature is transmitted to the trigger logic in the upstream SMUs and TGU, which then can decide on basic coincidence conditions such as the presence of at least one high-p_t electron and one high-p_t positron or the presence of a minimum number of high-p_t particles in a certain angular region.

However, if further computations are to be accomplished like the reconstruction of the invariant mass of the originating particle from a pair of electron/positron tracks, the precise transverse momentum value is required and the division has to be performed. Depending on the necessary precision of the result, a divider with several internal pipeline stages[2] is needed. The final correction[3] in stage n reduces the systematic bias introduced on the average by the approximation (8.23). The precision of the results is analyzed in Chapter 9.

8.4 Trigger Decision

By providing not only multiplicity and precise location information of high-p_t particle tracks, but particle identification and the individual transverse momenta, the GTU as a tracking trigger processor can accommodate a variety of trigger schemes.

To reach a trigger decision, the reconstructed tracks have to be analyzed with regard to the physics trigger objective. In the GTU, this is accomplished by applying basic track selection criteria in the TMUs, combining the information in the SMUs, and eventually computing the trigger decision in the TGU performing high-level calculations if required.

The amount of final processing required in the TGU module depends on the selected trigger strategy. A basic high-p_t electron/positron trigger, for example, needs only a disjunction of the signals indicating the presence of the respective particle. A basic jet trigger can be implemented in a similar way.

If latency conditions permit, however, the TGU can analyze the reconstructed high-p_t particle tracks globally. By evaluating pairs of tracks, more intricate trigger scenarios can be realized. The computations can include complex operations like the reconstruction of the invariant mass m_inv of the generating particle from the properties of an electron/positron pair. For this mode of operation, the data words transmitted by the SMUs have to include position and momentum information.

[2]The current implementation uses a divider with 11 internal pipeline stages to compute the result to a precision of 17 bit. The resulting reconstructed transverse momentum p_t is represented by a fixed-point number with seven binary positions after the decimal point in units of GeV/c. This high-precision design variant allows for direct and precise verification of the hardware results with simulations.

[3]Depending on the sign of the intermediate result $p_{\mathrm{t,\,raw}}$, one of two different correction constants is chosen to compensate for the sign-dependent average value shift that results from cutting off positions after the decimal point.

With the relativistic energy $E = \sqrt{m^2c^4 + \vec{p}^2c^2}$ for each particle, the invariant mass

$$m_{\text{inv}} = \frac{1}{c^2}\sqrt{(E_1+E_2)^2 - c^2(\vec{p}_1+\vec{p}_2)^2} \tag{8.24}$$

$$\overset{E_i \gg m_e c^2}{\approx} \frac{1}{c}\sqrt{2|\vec{p}_1||\vec{p}_2| - 2\vec{p}_1\vec{p}_2} \tag{8.25}$$

can be derived from the reconstructed electron/positron momenta

$$\vec{p}_i = p_{\text{t},i}\begin{pmatrix}\cos\varphi_i\\ \sin\varphi_i\\ \cot\vartheta_i\end{pmatrix}, \qquad |\vec{p}_i| = p_{\text{t},i}\frac{1}{\sin\vartheta_i}, \tag{8.26}$$

which depend only on the observed azimuth and zenith angles φ_i and ϑ_i of the particle tracks and the reconstructed absolute transverse momenta $p_{\text{t},i}$, which are all available in the GTU. A cut on the reconstructed invariant mass could notably increase the trigger selectivity for specific dilepton decays.

The current implementation includes a basic coincidence-based high-p_t dielectron trigger. The analysis and implementation of different trigger strategies for the ALICE TRD is an area of future research and development.

Conclusion For high-p_t tracks, circle fitting in the ALICE TRD can be replaced by linear regression. Because in a layered detector one of the spatial coordinates is discrete with a limited number of possible values, the computation of the regression can be simplified remarkably. Even taking into account the additional complexity introduced by the TRD's special readout pad geometry, the line parameter a can be calculated using only precomputed lookup tables and few additions and multiplications. Assuming a track origin at the primary interaction point, the transverse momentum can be estimated directly from the line parameter a. Comparison with a p_t threshold delivers a fast trigger signal. All computations can be implemented in a fully pipelined data push architecture optimized for low latency. The results provided by the online track reconstruction allow for a variety of sophisticated trigger strategies.

9 Performance Analysis

This chapter summarizes the results of various analyses regarding the performance of the GTU system as a trigger processor. The first part focuses on how well the GTU system together with the front-end electronics meets the requirements regarding efficiency and accuracy. The timing performance of the hardware implementation is analyzed in the second part.

While the final GTU system is fully installed and commissioned at the CERN LHC (see Appendix A for illustration) and experience from long-term continuous cosmic data taking using this setup is available, no actual physics data from LHC collisions is obtainable up to now. Therefore, the analyses in this chapter are based on simulations and results from a test beam setup.

Test Setup at CERN Proton Synchrotron (PS)

The analyses presented in this chapter include data from a test setup at the CERN PS. The setup is outlined in Figure 9.1. It includes a TRD supermodule representing an 18th of the full detector and additional detectors used for triggering and as references for optimizing particle identification. The particle beam consists of a mixture of electrons and pions at energies of up to 6 GeV. It crosses a fixed stack of the supermodule under constant angles $\varphi = 10°$ and $\vartheta = 16°$. The width of the beam as selected by the trigger detectors is approximately 5 cm. The setup also includes a GTU segment consisting of 5 TMUs and an SMU module as part of the readout chain. A picture of the setup is shown in Appendix A.

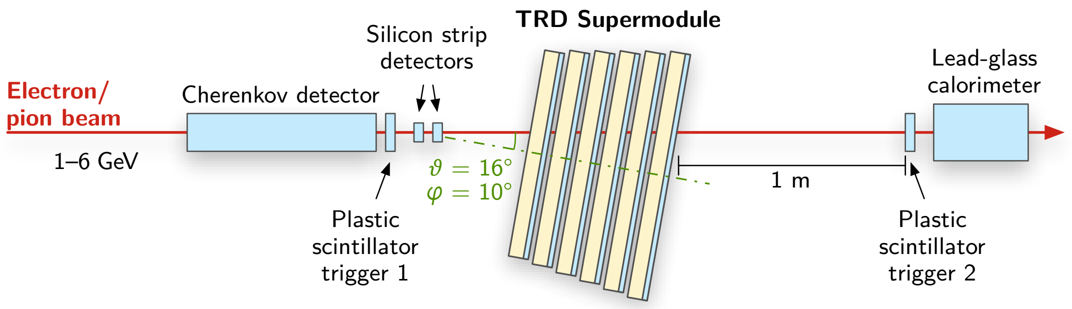


Figure 9.1: The setup of the November 2007 test beam at CERN PS with a TRD supermodule and additional detectors

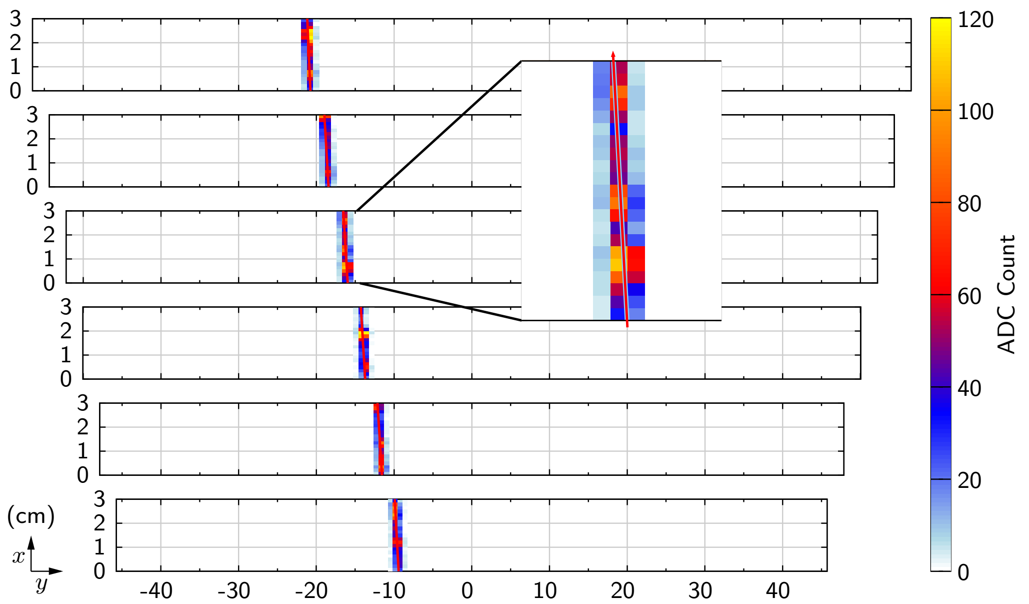


Figure 9.2: Display of a single event in a TRD stack. The color represents the measured ADC count for a given position. The red lines depict the parametrized track segments as received by the GTU for the same event. The width of the drift chambers is exaggerated to illustrate the charge measurements.

While the test beam setup provides no realistic environment to evaluate the tracking efficiency — the isolated, clean tracks crossing all six detector layers are always found in the GTU, resulting in an efficiency of one — the performance of the local tracking in the detector front-end and the precision of the track reconstruction in the GTU can be investigated. The test beam conditions also verify the functioning of the system under real-world conditions of continuous data-taking and provide data to analyze the timing behavior of the online tracking implementation.

Online Tracking Parameters

The front-end computations outlined in Chapter 6, employed in this setup for the first time together with the existing local tracking algorithm, deliver correct results. Figure 9.2 shows an exemplary event[1] as an overlay of raw ADC count and parametrized track segments as received by the GTU. In the magnified view of all six detector modules shown in Figure 9.3, the close match of the parametrized track segments generated online in the front-end processors (depicted as red arrows) and the raw ADC values can be seen.

[1]Data presented in this chapter results from analyzing run number 454. The single event depicted here is event number 1 of this run.

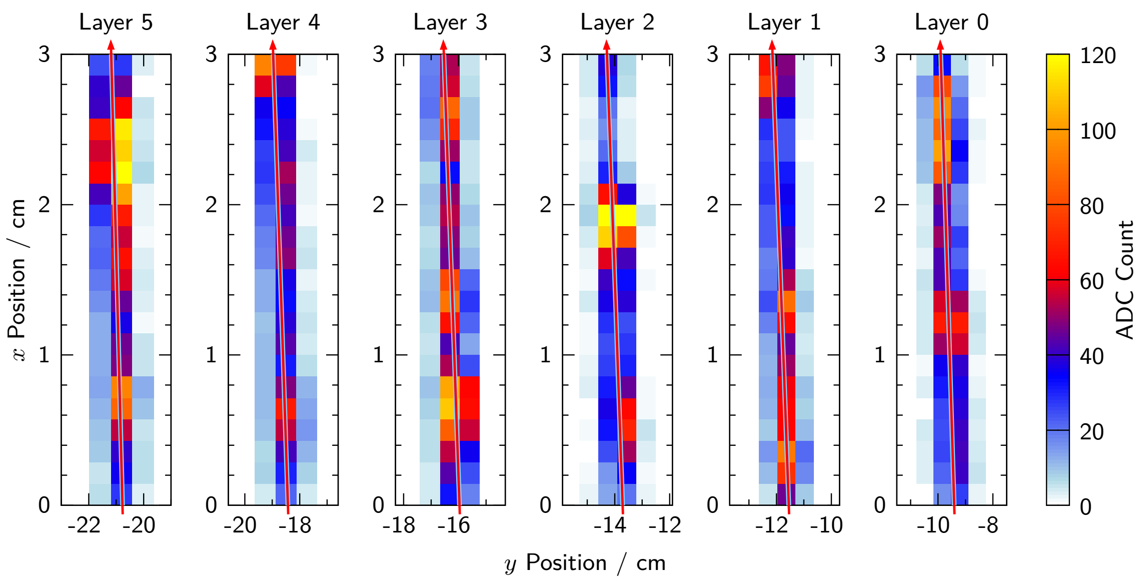


Figure 9.3: In the magnified view of all six modules, the close match of front-end-generated track segments (red arrows) and raw ADC values is visible.

9.1 Front-End Tracking Performance

The most important and most sensitive parameter of the front-end tracking procedure is the slope of the track segment, which is expressed as the local deflection per chamber. The local deflection is used for two purposes: track segment data reduction on front-end level and track matching in the GTU. It does, however, not contribute to track reconstruction in the GTU. Therefore, the deflection resolution does not influence the precision of the GTU reconstructed transverse momentum.

Figure 9.4 shows the error in the locally measured deflection. It results from analyzing actual track segments from the 2007 test beam setup. Since the test beam events consist of straight tracks, the error in the deflection can be determined by comparing the measured local deflections to the slope of a straight line fit to the track segment positions of each event. The standard deviation of a Gaussian fitted to the error distribution amounts to 1.0 mm. This is somewhat larger than the value of 0.5 mm estimated theoretically for the design of the TRD [67, p. 42]. In this setup, the angle $\varphi = 10°$ of the supermodule to the beam direction leads to a comparatively large average deflection of 5.3 mm. Previous results of an analysis of cosmic radiation events taken using a preproduction setup [38, p. 113] do not cover this deflection value. However, extrapolating the results for deviation and resolution of the online deflection to this deflection leads to similar results (approximately 1 mm). This source also indicates that results can be about a factor of two better if the online tail cancellation filter is activated. Thus, the larger error in the locally measured deflection when compared to the theoretical estimate can be attributed to a suboptimal configuration of the digital filter parameters, primarily the settings for the tail cancellation filter, during the test beam.

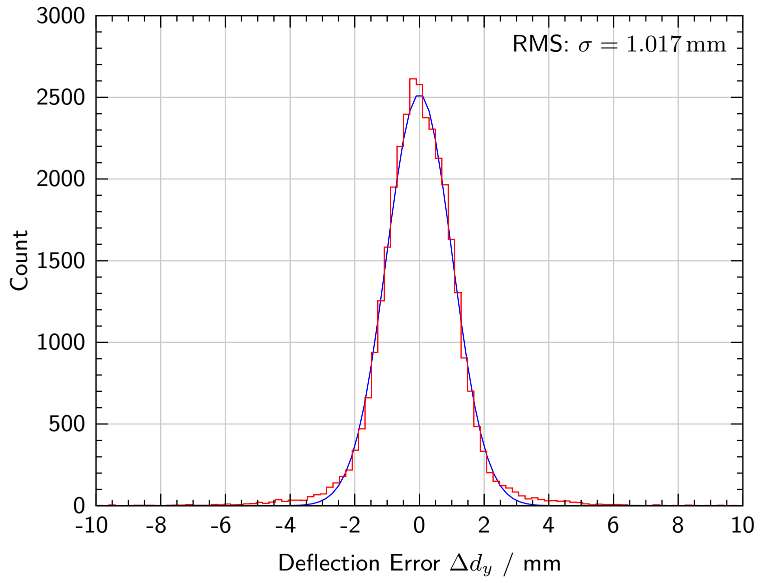


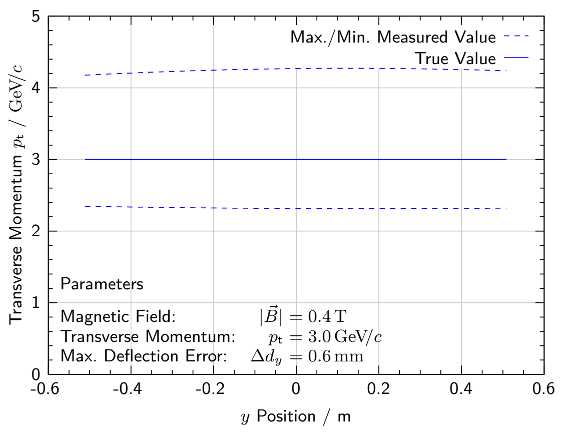
Figure 9.4: Distribution of the error in the measured local deflection. The analysis is based on actual track segments data from the 2007 test beam setup (run number 454), the first measurement of track segments in a production setup. The larger error when compared to previous estimations can be attributed to a suboptimal configuration of the digital filter parameters (see text).

The first drawback of a limited deflection precision is that the deflection cut in the MCMs, which reduces the number of track segments transmitted to the GTU, has to be shifted to lower values of the transverse momentum, thus increasing the background of transmitted irrelevant track segments. Otherwise, valid high-p_t track segments might be lost. This elementary relation is illustrated in Figure 9.5.[2] Because of the error in the measured deflection, the locally reconstructed transverse momentum deviates from the true value. Thus, the transverse momentum cut has to be set to a lower value. At a local deflection error of $\Delta d_y = 1\,\mathrm{mm}$, aiming for particles above 3.0 GeV requires a local transverse momentum threshold below 2.0 GeV.

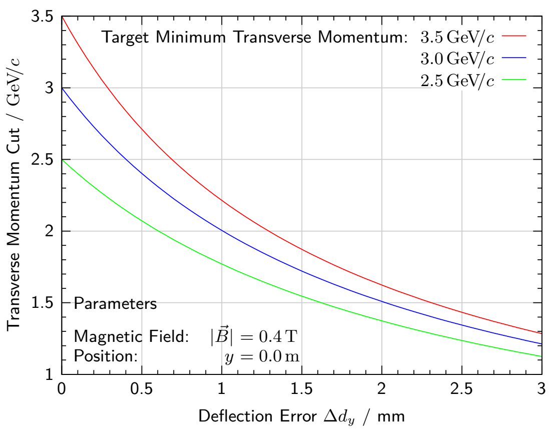
In the GTU, the track segment deflection is used to project the track segments to a common plane. Therefore, an error in the deflection causes an error in the track segment's y position on the projection plane. For the innermost and outermost layers, the projection factor is 10.5. The windows size in y has to be increased by $2 \cdot 10.5 \cdot \Delta d_y$ to account for the local deflection error.

At this stage, the error in the y position is less critical. Fitting straight lines through the y positions results on average in a root mean square deviation of each position of 0.8 mm. This is about the same size as the average error expected to be introduced by the tilted pad geometry (see Chapter 6), not taking into account the imperfect alignment of the chambers.

[2]The slight asymmetry in x in Figure 9.5a originates from the assumed charge of the particle. Values are mirrored for opposite charge.



(a) Because of the deflection error, the locally estimated transverse momentum deviates from the true value. The range of measured values for a given deflection error depends only slightly on the track position in the chamber.



(b) The transverse momentum cut is selected depending on the target minimum transverse momentum and the deflection error. A large value for the deflection error requires a reduced threshold setting.

Figure 9.5: Effect of the track segment deflection error on the required transverse momentum threshold on MCM level.

9.2 GTU Tracking Performance

Figure 9.6 shows the superposition of a test beam particle track reconstructed in the GTU with corresponding raw ADC values and front-end generated track segments in a TRD stack. In the following, the precision of the track reconstruction is analyzed quantitatively.

9.2.1 Resolution of the Reconstructed Transverse Momentum

As a criterion for the quality of the reconstruction, the mean error of the reconstructed transverse momentum is considered. A precisely reconstructed transverse momentum in the GTU is an important prerequisite for an efficient trigger decision as it allows for accurate cuts on the particle momentum itself or derived quantities such as the originating particle's invariant mass.

The precision of the reconstructed momentum can be analyzed by comparisons at several stages. Figure 9.7 summarizes the major sources of error in the reconstructed transverse momentum. Using actual data from the test beam, the results of the online p_t reconstruction in the GTU can be compared to precise software computations[3] on the same data.[4] This corresponds to the difference between the stages labeled *quasi-offline* (p_t^sw) and *track*

[3]The employed circle fitting algorithm is described in [29].

[4]To allow for the analysis of the reconstructed transverse momentum, the event data sets are considered as circular high-p_t particle tracks originating from a virtual primary vertex.

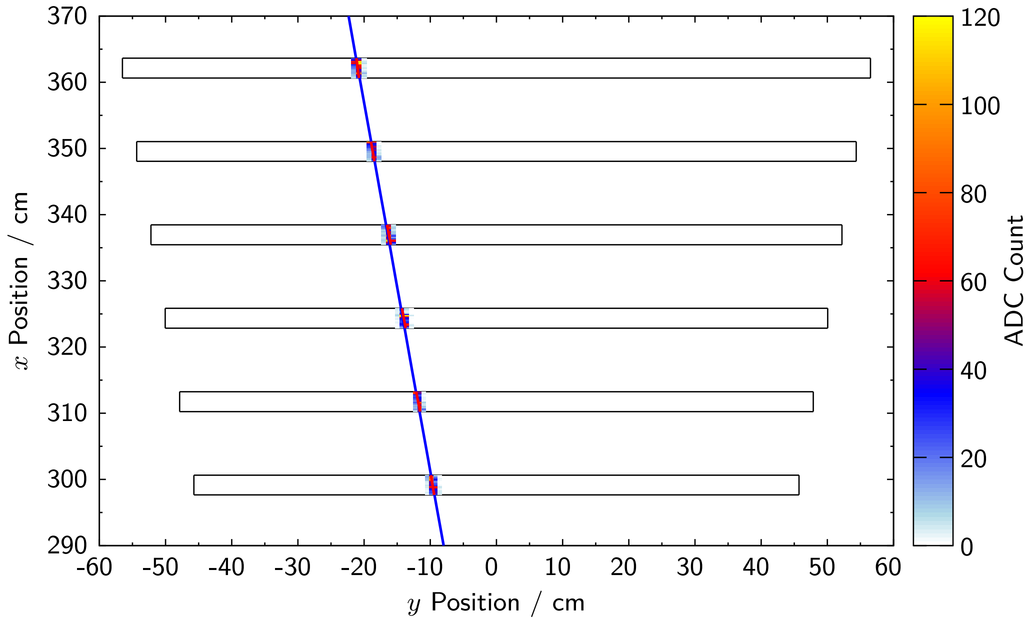


Figure 9.6: TRD stack with GTU-reconstructed track. The parametrized GTU track (blue line) is overlaid with front-end generated track segments (red lines) and corresponding raw ADC values. The illustration is true to scale.

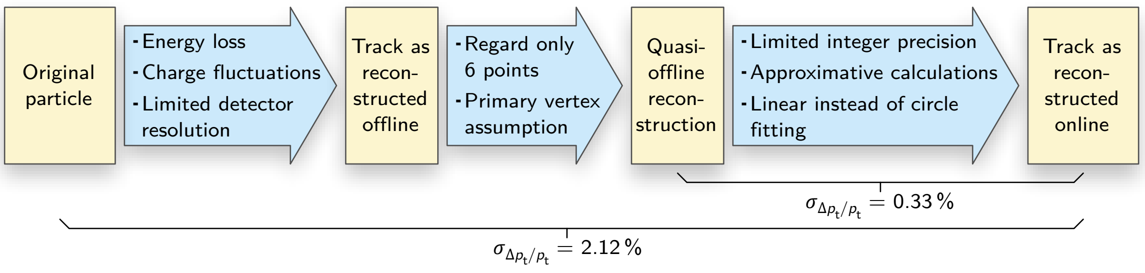

Original particle
· Energy loss
· Charge fluctuations
· Limited detector resolution
Track as recon-structed offline
· Regard only 6 points
· Primary vertex assumption
Quasi-offline recon-struction
· Limited integer precision
· Approximative calculations
· Linear instead of circle fitting
Track as recon-structed online
$\sigma_{\Delta p_t/p_t} = 0.33\,\%$
$\sigma_{\Delta p_t/p_t} = 2.12\,\%$

Figure 9.7: The major sources of error in the reconstructed transverse momentum

as reconstructed online (p_t^{rec}) in the diagram. It makes sense to regard the relative error since the error increases for large transverse momenta (follows from Equation (8.23) in the case of constant error in the axis intercept a). The results of this comparison are shown in Figure 9.8. The errors approximately follow a Gaussian distribution with a standard deviation of $\sigma_{\Delta p_t/p_t} = 0.329\,\%$, which is an estimate of the relative resolution of the reconstructed transverse momentum. This result represents the accuracy of track reconstruction in the GTU. It includes all errors introduced by the approximate calculations, the limited precision of the utilized integer representation, and the simplified linear fitting algorithm as presented in Chapter 8.

For comparison, Figure 9.9 shows the distribution of the relative error of the result of the track reconstruction performed by the GTU related to the real transverse momentum of high-p_t electrons in a Monte Carlo event simulation (see [23] for details). This distribution

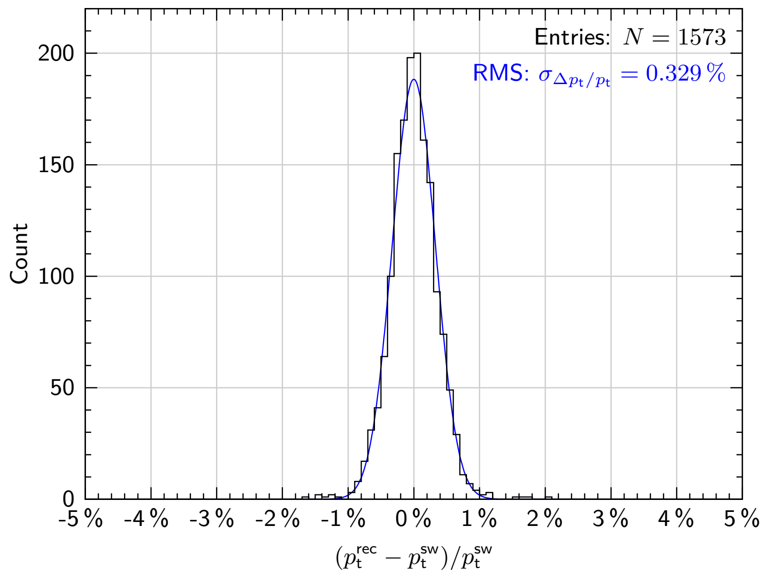


Figure 9.8: Resolution of the p_t reconstruction algorithm in the GTU. A Gaussian distribution (rendered in blue) is fitted to the error distribution. The width of the distribution is a measure of the accuracy of the online reconstruction procedure in the GTU.

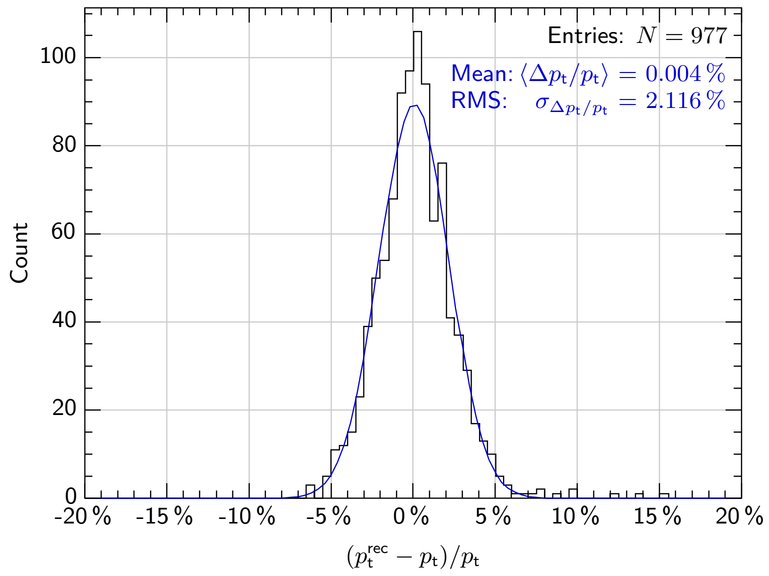


Figure 9.9: Resolution of the transverse momentum reconstructed in the GTU. A Gaussian distribution (rendered in blue) is fitted to the error distribution. Mean value and width of the distribution are measures of the accuracy of particle tracking in the TRD including the online reconstruction process in the GTU.

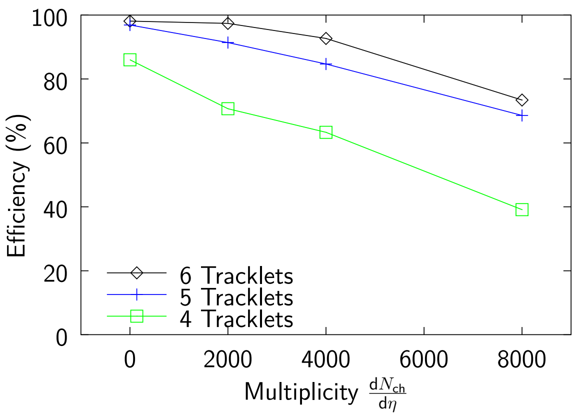


Figure 9.10: Tracking efficiency of the GTU for tracks with four, five, and six segments (tracklets) at different multiplicity densities (from simulation). Source: [23]

additionally includes the errors introduced by the detector and the experiment on the whole, such as energy loss of the particle before and inside the detector and the limited resolution of the detector.[5] The contribution of the second set of errors summarized in Figure 9.7, the limitation to only 6 data points instead of the clusterized raw ADC values and the primary vertex assumption, is comparatively small for high-p_t particles. The standard deviation of the error distribution amounts to $\sigma_{\Delta p_t / p_t} = 2.116\,\%$.

Comparison of the two results shows that most of the error in the reconstructed transverse momentum originates from detector limitations. The error introduced by the simplified computations is small, and the online reconstruction algorithm implemented in hardware in the GTU reaches a high level of precision. As the errors add up quadratically, further increasing the precision of the GTU computations would have only negligible effect on the total online reconstruction precision.

9.2.2 Tracking Efficiency

Results regarding the tracking efficiency and background rate of the GTU are discussed in detail in [23]. The detection efficiencies shown in Figure 9.10 result from a simulation of electrons and positrons that have been produced in a Υ-particle decay with a transverse momentum of $p_t > 3.0\,\text{GeV}/c$, pass the detector mostly unaffected[6], and create a track segment in at least four of the six layers. These results do not reflect the performance of the isolated GTU under optimal conditions, but its behavior regarding the application in the TRD. The efficiency of the online tracking decreases with rising multiplicity density $\frac{dN_{ch}}{d\eta}$ of the events. Tracks with fewer segments are more affected by this effect.

The decrease in detection probability at higher multiplicity has two main reasons. On the one hand the probability that an irrelevant track segment is mistakenly added to a searched

[5] The simulation regards only electrons and positrons with a transverse momentum $p_t > 3.0\,\text{GeV}/c$ that lose at most $1\,\%$ of their transverse momentum each before and inside the detector.

[6] The analysis is limited to particles that still possess at least $90\,\%$ of their original transverse momentum when entering the detector and beyond that lose at most $10\,\%$ of their transverse momentum while traversing the detector.

track because it coincidentally meets the window criterion increases with the multiplicity. In this case, there is the possibility that the extra segment distorts the reconstructed transverse momentum $p_{\mathrm{t}}^{\mathrm{rec}}$ in such a way that it is shifted below the threshold value of $p_{\mathrm{t,\,min}}^{\mathrm{GTU}} = 2.7\,\mathrm{GeV}/c$ and the respective electron is not found.

With larger multiplicity on the other hand the probability increases that already on the level of the front-end electronics a single track segment consists of the superposition of two particle tracks. Through the influence of the other particle, especially the slope of the track segment can be distorted such that it is not added to the track by the GTU. Tracks with only four segments are already rejected if only a single segment is missing, so that in this case the impact of this effect on the efficiency is particularly evident.

At a multiplicity density of $\frac{\mathrm{d}N_{\mathrm{ch}}}{\mathrm{d}\eta} = 2000$, which is slightly below current expectations, 97.4 % of the complete particle tracks (6 segments) are found, for tracks with only 5 or 4 segments generated in the front-end the percentage drops to 91.4 % or 70.7 %.

9.3 GTU Timing Performance

The trigger decision has to be delivered at a specific point in time (see Section 2.4.2). The tracking procedure presented in Chapters 7 and 8, however, requires by its nature a variable processing time, which generally depends not only on the amount of input data but also on the specific contents of the input data words. It is designed such that each unit passes on its data on average with as small a latency as possible. To a certain extend, delays are balanced out between the units. Moreover, it can be assumed that the at most 40 track segments per module at highest multiplicity occur distributed over the chamber. Since the GTU trigger design is optimized for a low total latency under these assumptions, it does not make sense to investigate the theoretical maximum processing time. Instead, the timing behavior is studied by simulations.

The timing analyses presented in this section result from functional simulations of the GTU hardware model. The input data streams consist of track segments from the test beam setup complemented by randomly generated track segments to allow for the analysis of different track segment counts. In the experiment, the number of track segments per stack depends on the multiplicity of the event and on the deflection cut that is applied in the front-end. Allowing a maximum deflection of 6 mm (corresponding to a minimum transverse momentum of approximately 1.0 GeV/c), approximately 18 track segments are expected on average per module at full multiplicity density of $\frac{\mathrm{d}N_{\mathrm{ch}}}{\mathrm{d}\eta} = 8000$ [67].

9.3.1 TMU Trigger Processing Time

Figure 9.11 shows the distribution of the total trigger computation time in an individual TMU. The processing time depends heavily on the number of track segments. The minimum latency in case of a single clean track (6 segments) amounts to 550 ns. The

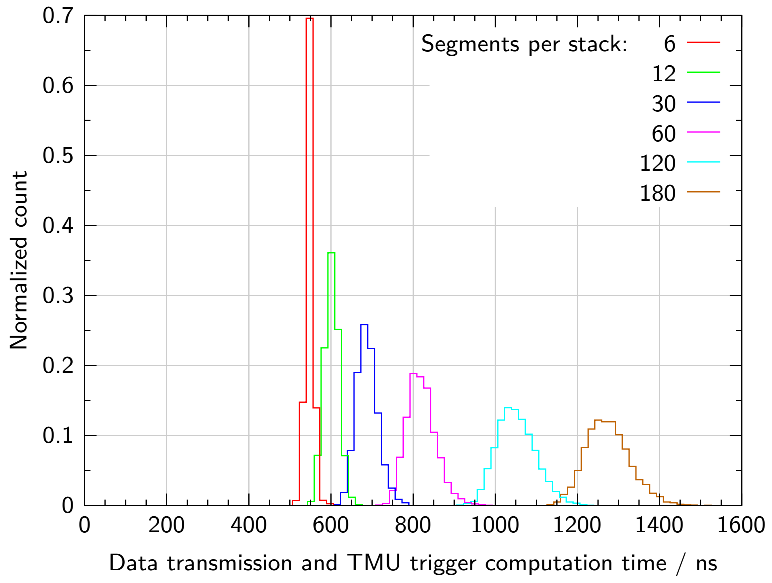


Figure 9.11: The total trigger processing time in the TMU is not constant but depends heavily on the number of received track segments — and thus on average on the multiplicity density. The time zero designates the time of the arrival of the first track segment, hence, the transmission duration is included in the indicated time. Each histogram contains data of approximately 7000 events.

distribution widens with increasing number of track segments per stack and the mean value reaches 1.3 µs for 240 segments per stack, i. e., on average 15 segments per link.

This time includes the track segment transmission duration and all trigger processing in the TMU. It starts when the first track segment arrives at the TMU trigger design and ends when the trigger contribution is transmitted to the SMU. This corresponds to the time period labeled *Tracking on TMU* in Figure 5.11, for which approximately 1.8 µs are allocated.[7] The analysis assumes that the track segments are received as a continuous stream without gaps caused by the detector readout tree. This is acceptable because merging in the tracking design's input buffers approximately halves the data rate, balancing out potential gaps in the data stream.

The total trigger processing time in the TMU can be divided into three major parts: track segment transmission, segment buffering and synchronization, and the actual trigger calculation including track reconstruction. The partitioning of the total processing time is shown in Figure 9.12. The track segment transmission time is with minor statistical fluctuations proportional to the number of track segments per stack. Synchronizing the track segments between the two links per layer is accompanied by a data rate reduction, which explains its large contribution that is also proportional to the number of track

[7]In this chapter, the TMU trigger decision time is the time at which the information whether an electron or positron with a transverse momentum above a certain threshold value is found or not is available in the TMU. It assumes that the decision is only forwarded on SMUs and TGU and does not account for latencies introduced by additional trigger logic and data transmission to SMU and TGU.

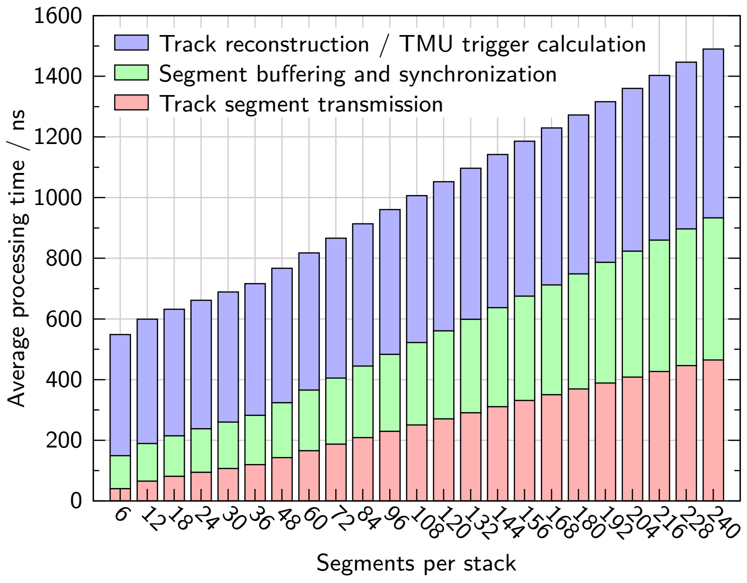


Figure 9.12: The total trigger processing time in the TMU can be divided into three major parts with different dependencies on the number of segments per stack.

segments for large track segment counts. For small counts, the synchronizing itself takes precedence. The remaining time is spent on track reconstruction and trigger calculation, which has a significant offset caused by the total depth of the pipeline, but on average increases only slightly with the number of track segments.

To statistically analyze the timing behavior with respect to the maximum allowable latency, the average value of the distribution is not sufficient. Instead, the complete distribution, in particular the high-end region, has to be considered. Figure 9.13 combines the histograms shown in Figure 9.11 together with data of additional segment counts. While the histograms are rendered as colors in the background, the graphed values represent a five figure summary of the respective distribution. While the red bars depict mean value and standard deviation of the distributions, the three additional points represent three different quantiles[8]. The quantiles represent the extreme probabilities that are required if the computation is to be completed in time in the vast majority of cases. At 180 segments per stack, for example, the computations are completed after 1.5 µs in 99.9 % of the cases.[9]

In Figure 9.14, the time required for track reconstruction and trigger computation in the TMU is displayed in the same manner. While the TRD is designed to deliver less than 240 track segments per stack, the algorithmic efficiency, i. e., the dependence of the run-time on the number of input data words (track segments), is interesting from a fundamental

[8]In this terminology, the p-quantile is the value x such that the probability that a random variable will be less than x is p.

[9]The fluctuations in single data points for high quantiles result from the low statistics in these extreme data points. However, the fitted smooth curves approximate the true values well as they implicitly improve statistics by combining adjacent values.

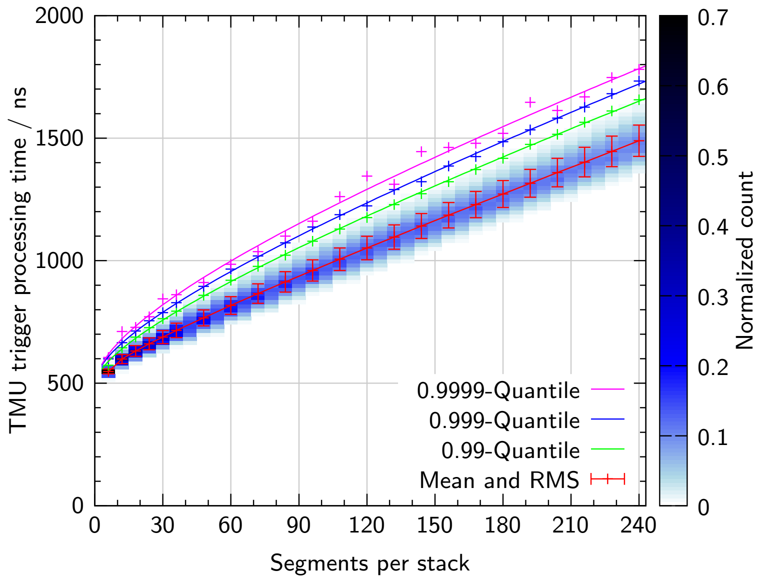


Figure 9.13: Summary of the total trigger processing time in the TMU. The graphed values represent a five figure summary of the respective distribution (see text). Each data point results from processing approximately 7000 events.

perspective. A straightforward sequential algorithm comparing all track segments of one layer with all segments of the other layers would result in a quadratic dependence on the number of input words. A histogramming procedure would commonly result in a linear dependence.

In this case, the pattern of the values suggests that the average processing time t_{rec} can be approximated by

$$t_{\mathrm{rec}} = p_1 \cdot \sqrt{n_{\mathrm{sec}}} + p_0 \tag{9.1}$$

as the sum of the absolute pipeline latency ($p_0 \approx 380\,\mathrm{ns}$) and an additional time for combining the track segments that is proportional to the square root of the segment count ($p_1 \approx 12\,\mathrm{ns}$).[10] While this is only valid within the given range of values and for proper input data, it indicates that under these conditions the employed tracking algorithm performs its computations at a low latency that depends only slightly on the number of track segments. As a result, the total TMU trigger processing time including track segment transmission and buffering depends only linearly on the track segment count.

9.3.2 GTU System Trigger Processing Time

Caused by the principle of the tracking algorithm, rare cases in which the track reconstruction or the trigger computation does not conclude in time for the trigger contribution cannot be excluded. These cases require a fallback strategy, which is to always issue a

[10]The time values in Figure 9.13 are fitted by a similar function with an additional summand $p_2 \cdot n_{\mathrm{sec}}$ proportional to the track segment count n_{sec}.

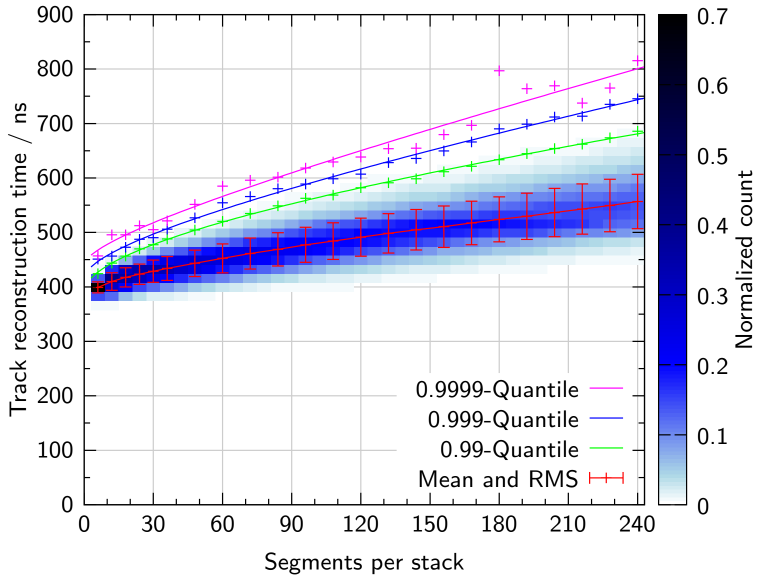


Figure 9.14: Summary of the track reconstruction time in the TMU. Starting at an offset of approximately 400 ns, the track reconstruction time rises only slightly with increasing number of segments per stack.

trigger.[11] The impact of this fallback behavior on the overall data rate is small if the frequency of these cases is adequately low, which has to be guaranteed by the trigger algorithm.

To evaluate the processing time required for the full TRD, the presented stack data has to be extrapolated. As the overall GTU trigger computation can only finish after all 90 TMU have completed their computations, the longest TMU processing time determines when a decision is available. To deduce the probability of the GTU completing the processing in a given time, the respective probabilities of the contributing TMUs have to be multiplied. Thus, the 0.999-quantile in Figure 9.13 for a single stack corresponds only to a $0.999^{90} = 91.4\,\%$ completion probability for the GTU.

What rate of incomplete trigger computations is considered acceptable in the experiment depends on the running scenario. The required time for a completion probability of 99 %, i. e., less than one erroneous trigger caused by incomplete computations in every 100 events, is displayed in Figure 9.15. At 10 000 track segments in the detector per event, 1.25 µs are required to reach this level of confidence. At the maximum number of 20 000 track segments per event, this time amounts to 1.7 µs. As before, the transmission duration is included in these values, while the time required for data transmission from the TMUs to the trigger logic in SMUs and TGU and their computations remains to be added.

[11]Besides the common correlation of high multiplicity events causing extensive trigger computations and interesting events, this strategy is required for bias-free operation of the trigger. By recomputing the trigger decisions to the end in an offline algorithm for the specific cases in which the GTU online processing did not finish, these events can be decided on later.

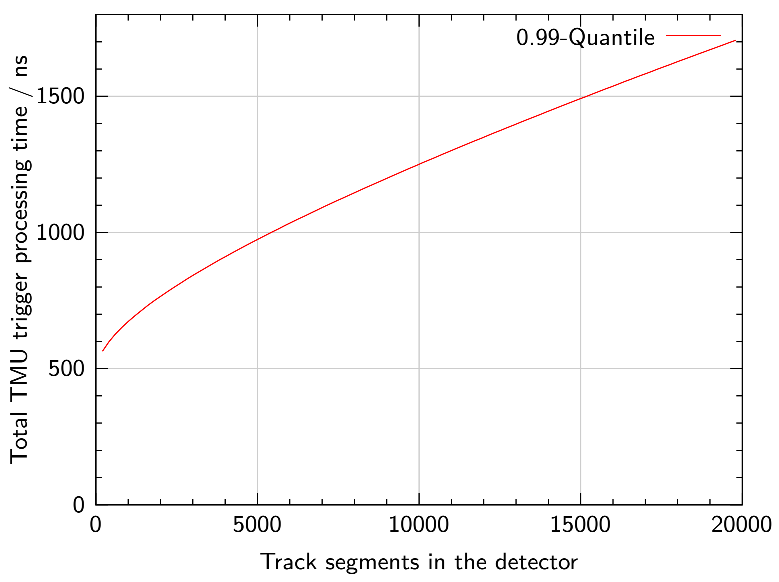


Figure 9.15: The total TMU trigger processing time with respect to the complete GTU results for each event from the maximum processing time of the individual TMUs. The graph shows the overall processing time that is required for a completion probability of 99 %.

In conclusion, the trigger design in the TMU finishes its computations within the envisaged time frame up to the highest intended number of 20 000 track segments in the TRD. At this extreme occupancy, however, only a small amount of time remains for future, more sophisticated trigger decision logics in the SMU and TGU. Nevertheless, depending on the multiplicity of the events and the setting of the deflection cut applied in the front-end processors, currently expected values for the track segment count are considerably lower. Scaling the number of track segments with the expected multiplicity density from $\frac{\mathrm{d}N_{\mathrm{ch}}}{\mathrm{d}\eta} = 8000$ to $\frac{\mathrm{d}N_{\mathrm{ch}}}{\mathrm{d}\eta} = 2000$ according to current expectations results in a significantly reduced total TMU trigger processing time of less than 1 µs.

10 Summary and Outlook

The Transition Radiation Detector (TRD) of the ALICE experiment provides spatially resolved measurements of particle tracks, momentum measurements, and particle identification. It serves in particular as a trigger for slower, high-resolution detectors of the experiment. The hierarchical trigger concept of the TRD incorporates distributed, massively parallel data processing and reduction directly on the detector chambers, which are arranged in cylindrical layers, in conjunction with central data processing and analysis in a dedicated trigger processor named Global Tracking Unit (GTU). The track segments observed in the chambers are parametrized locally as line segments. The trigger processor's major task is to reassemble the related segments from the different detector layers to tracks and to reconstruct from the course of these tracks the transverse momentum of the producing particles, which represents an important trigger criterion.

In this thesis, a trigger processor is presented that combines existing and novel techniques for low-latency online tracking aiming at achieving results in some aspects similar to offline analysis. The work summarized in this thesis covers the complete process from initial simulations and design studies via specifying and implementing tracking algorithms as well as developing a suitable hardware architecture to constructing and building the full trigger processor system as it is now in active operation in the first stages of the ALICE experiment.

Because of requirements of the experiment, the GTU has less than 2 µs available for the computations on the up to 20 000 track segments until a trigger decision has to be reached. This constraint dominates every aspect of the system structure as it demands not only efficient data processing, but also optimization for low latency of every relevant communication path in the system.

The resulting hardware architecture is a three-stage hierarchical structure connected in a tree network topology. The system comprises 90 processing nodes, 18 concentrator nodes, and 1 output node. The altogether 109 nodes are based on FPGA technology. Each node is realized as a custom printed circuit assembly using a CompactPCI form factor. The system is designed to follow industry standards where applicable, but employ custom developments where necessary. The connections between the nodes are a critical point in the design. Commercially available bus systems and network components do not meet the low-latency requirements. A custom backplane implementing a star plus ring topology based on source synchronous LVDS signals combines high bandwidth with minimum latency. The processing nodes receive data from the detector modules at an aggregate net bandwidth of 2.16 Tbit/s via 1080 optical receiver modules. Because the system architecture reflects the TRD setup of 90 module stacks, the nodes can perform most data processing independently in parallel.

Each of the GTU's 109 FPGAs is configured and monitored by an embedded PowerPC processor. Three distinct configuration and control networks at different layers provide flexible and reliable means of remote system management and allow the trigger processor to be integrated into the general ALICE control systems.

With respect to efficiency, it is favorable to use the trigger data links also for raw data transmission, effectively making the trigger processor a part of the experiment's raw data readout chain. Given the resources of a trigger processor such as the TRD GTU, the raw data readout functionality can include not only plain data forwarding, but high-speed event building in dynamically allocated multievent buffers. Multievent buffering is an important feature for the performance of the ALICE TRD with respect to detector dead time and readout rate. Despite the high bandwidths involved, it can be accomplished in the FPGA-based trigger processor GTU, resulting in a significantly reduced dead time.

Corresponding to the hardware structure, the trigger computation is a three-stage process. The online tracking as its first and major part is distributed among the processing nodes, which handle different angular regions in parallel. By first projecting the line segments from the six detector layers onto a common center plane, it is possible to decide whether two given segments belong to the same track. Using a parallel selection method, track matching in one of the three coordinates can be resolved implicitly. In total 810 parallel track finder units are the most complex part of the design and in terms of maximum clock rate the critical point of the architecture. By iterating the projected track segments of the different layers, they find continuous tracks based on a reference layer. As the positions of the track segments are compared directly in the remaining two coordinates, the thresholds for detecting an agreement can be specified precisely.

To reconstruct the transverse momentum p_t of a recognized particle, the curvature of its track has to be analyzed. For high-p_t particles emerging from the primary vertex, circle fitting in the ALICE TRD can be replaced by linear regression. Because in a layered detector one of the spatial coordinates is discrete with a limited number of possible values, the computation of the regression can be simplified remarkably. Even taking into account the additional complexity introduced by the TRD 's special readout pad geometry, the transverse momentum can be calculated using only precomputed lookup tables and few inexpensive operations such as additions and multiplications. All computations can be implemented in a fully pipelined data push architecture optimized for low latency.

To reach a trigger decision, the reconstructed tracks have to be analyzed with respect to the physics trigger objective. While comparison with a p_t threshold delivers a fast trigger signal, the results provided by the online track reconstruction allow for more sophisticated trigger strategies. By providing not only the number and precise location of high-p_t particle tracks, but also particle identification and the individual transverse momenta, the GTU as a tracking trigger processor can accommodate a wide variety of possible trigger schemes including the selection of specific dilepton decays and jets. The flexibility provided by the FPGA-based trigger processor architecture allows to efficiently adapt the system to these signatures and to configure it to changing luminosities or unexpected rates in the future. The analysis and implementation of different trigger strategies for the ALICE TRD as well as optimizing particle identification in the online application are areas of future research and development.

The statistical analysis of simulated events shows that the behavior of the tracking design fulfills the requirements regarding tracking efficiency and reconstruction precision. Detailed timing analyses indicate that particle tracking is accomplished in the required time even up to the original maximum design multiplicity of the TRD. As the presented design includes only a basic trigger logic, future trigger algorithms may require more processing time. The presented architecture, however, provides possibilities of reducing the necessary number of clock cycles even further by replacing generalized and powerful design components such as the logic for rejecting duplicate track findings by simpler or more optimized units adapted to a specific trigger scenario. Furthermore, the currently expected multiplicity is considerably lower than the original maximum multiplicity, allowing for even more sophisticated algorithms. Besides the development of further refined trigger strategies, next major steps include the complete integration of the trigger processor into the general ALICE detector control environment and additional performance tuning, particularly with regard to raw data readout and the multievent buffering implementation.

As a result, this thesis shows that optimizing the hardware architecture for low latency, online track reconstruction is possible using an essentially sequential and precise algorithm based on explicit value comparisons even within tight timing constraints. By carefully analyzing the structure of the input data with regard to the detector's geometry, multidimensional matching is accomplished in a time that in typical cases depends only linearly on the occupancy. The concept of the GTU is put into practice at the ALICE experiment at CERN, where it is continuously in operation as a vital part of the TRD's readout and trigger chain.

Acknowledgments I would like to express my deepest gratitude to my supervising professor, Prof. Dr. Volker Lindenstruth, whose expertise, enthusiasm, and dedication have been an inspiring example. He provided me with the exciting challenge of working on this multifaceted topic including a stimulating six months research period at CERN. I am grateful for his strong support of my own ideas throughout the progress of this project, and his great confidence in my abilities. I would also like to thank Prof. Dr. Ulrich Brüning for taking time from his busy schedule to serve as my second referee. A special thanks goes out to Felix Rettig and Stefan Kirsch. I enjoyed our close collaboration in many occasions both in Heidelberg and at CERN. Thanks for the fun and support. I wish to express my gratitude to Dr. Venelin Angelov and Dr. Rolf Schneider for lots of helpful suggestions and enlightening discussions. Thanks also goes out to numerous people of our group and the ALICE collaboration for all the instances in which their support helped me along the way. I would like to thank Holger Höbbel and Dirk Hutter for their technical efforts and suggestions, without which this project would never have reached fruition. I would also like to thank Rebecca Neumann for the support she provided me. Your love and encouragement have upheld me. In conclusion, I would like to thank my parents for the constant support and encouragement they provided me through my entire life and in my attainment of this goal.

Appendix A

The GTU Setup at CERN

Figure A.1 illustrates the setup of the November 2007 test beam at the CERN Proton Synchrotron (PS). The beam of electrons and pions enters the setup at the upper right corner of the image. The arrangement of ALICE Transition Radiation Detector (TRD) supermodule and additional reference detectors in the center of the picture corresponds to the schematic drawing shown in Figure 9.1. The rack in the background contains a Global Tracking Unit (GTU) segment used for all data taking. Data analyzed in Chapter 9 results from this setup.

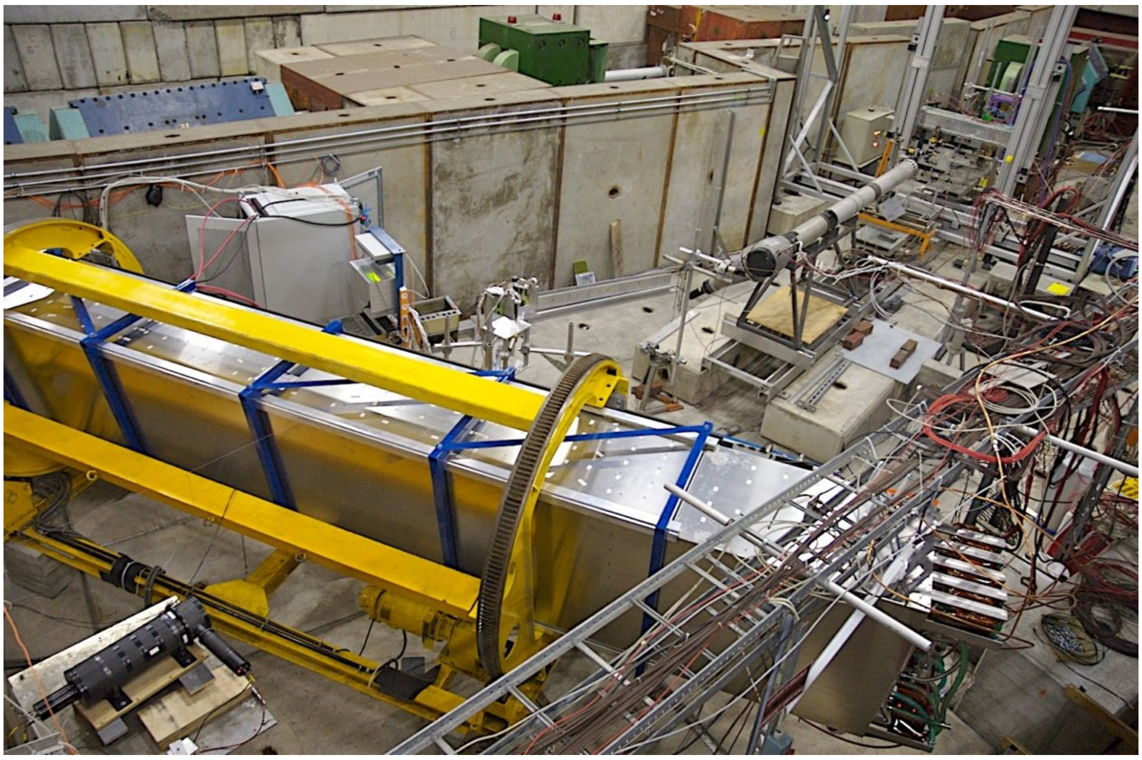


Figure A.1: Photo of the November 2007 test beam setup at CERN PS with a complete TRD supermodule shown inside the yellow rotating tool in the foreground. The rack in the background contains a GTU segment.

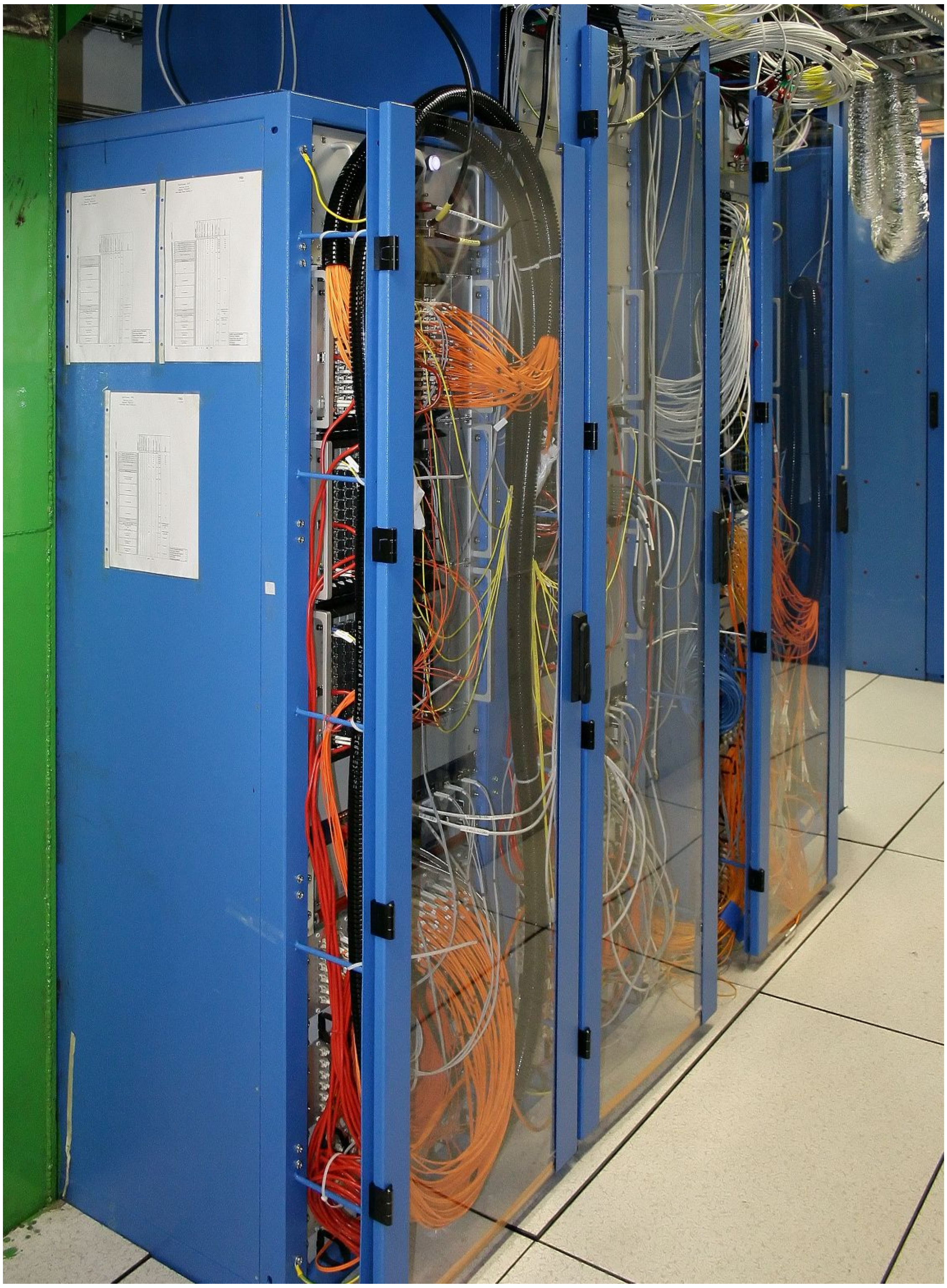


Figure A.2: Photo of the GTU setup. The trigger processor is installed in racks C16–C18 in the experiment cavern UX-25 at CERN LHC.

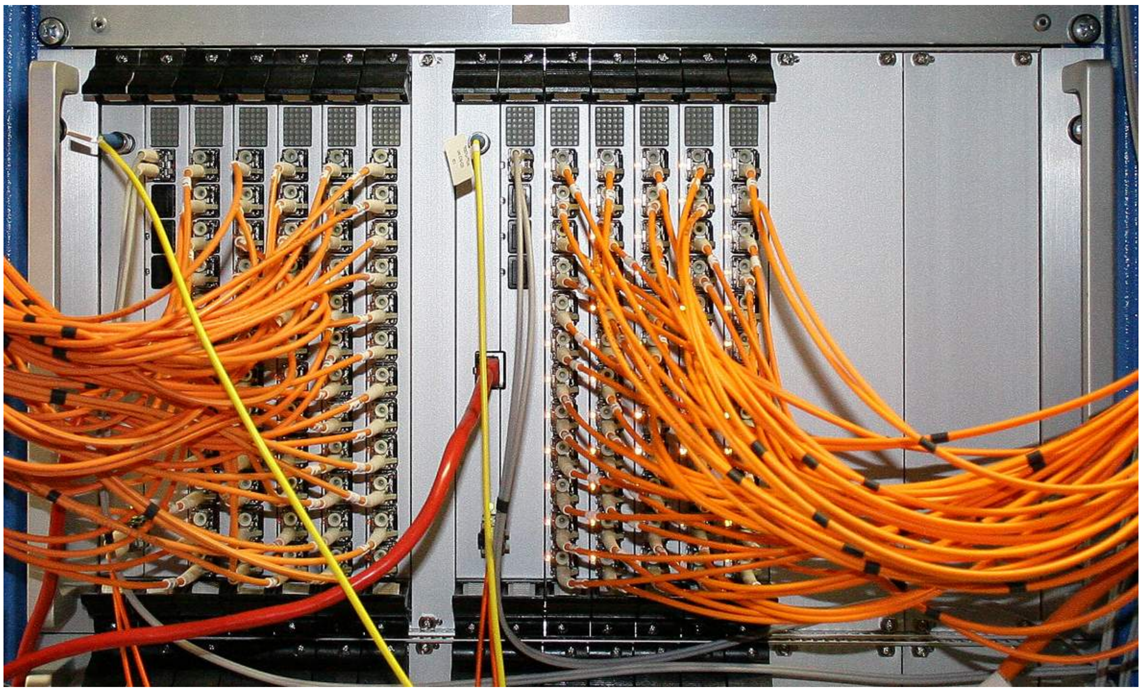


Figure A.3: Photo of a GTU subrack containing 2 supermodule segments. The complete trigger processor contains 18 of these segments.

Figure A.2 shows the full GTU setup in the underground experiment cavern UX-25 at CERN LHC. It contains 18 segments, each processing the data of a TRD supermodule. As a detail of the picture, a single subrack containing two supermodule segments fully equipped with more than 120 optical fibers can be seen in Figure A.3. The orange fibers connect to the detector front-end and the data acquisition system, while the yellow fibers deliver trigger messages from the CTP. The gray and red cables are optical and electrical connections to the Ethernet-based Detector Control System network.

The trigger processor is placed at the lowest level of the cavern in direct vicinity of the L3 magnet and the CTP to ensure low signal propagation delays. As the racks are located in the magnetic stray field of the muon system dipole magnet, conventional cooling fans cannot be used. Instead, each rack is equipped with a powerful turbine and a water-cooled heat exchanger at the top. Carefully guided airflow in the racks is required to cool the GTU, particularly its 109 FPGAs.

The operating voltage for the GTU is provided by water-cooled power supply units capable of delivering 110 A at 3.3 V and 55 A at 5.0 V to each of the nine subracks, resulting in a total maximum power of 5.7 kW. The actual power input of the system depends significantly on the processing performed in the FPGAs. Typical values amount to approximately 50–60 % of the maximum power.

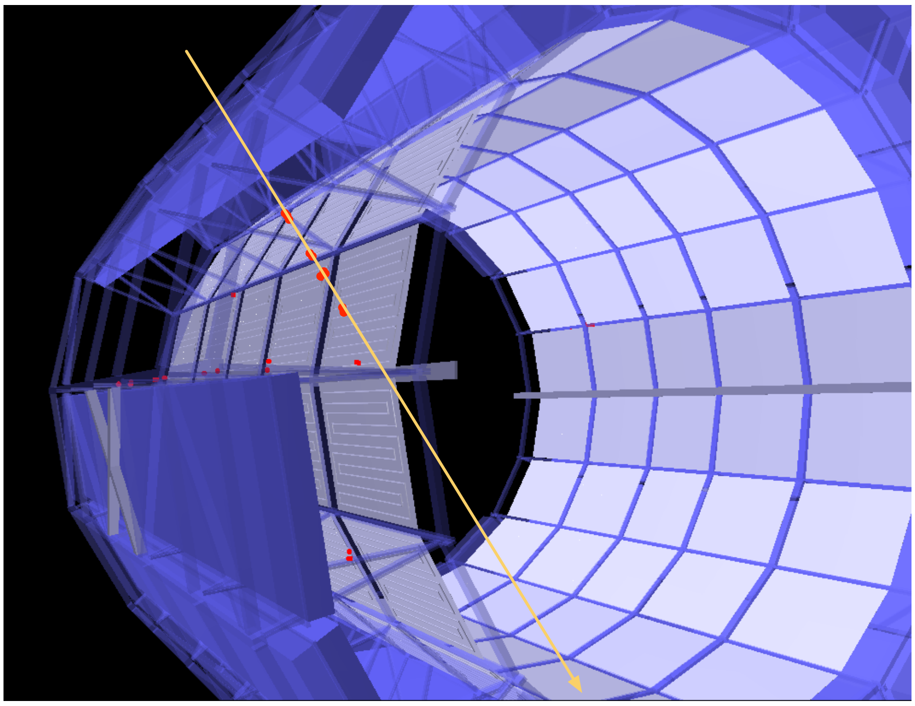


Figure A.4: TRD event display showing a cosmic particle event triggered and acquired by the GTU. The red markings represent the charge clusters detected in the different layers of the TRD. Source: [62]

Since its installation and commissioning, the TRD GTU has participated in numerous runs of cosmic particle data taking. In recent months, an L1 trigger on cosmic particle events has been developed [62] based on the hardware presented in this thesis. It illustrates the flexibility of the GTU system. Figure A.4 shows the visualization of a cosmic particle event in the ALICE TRD triggered by the GTU.

Appendix B

Technical Aspects of the GTU Implementation

B.1 Hardware Description in VHDL

The trigger processor GTU is based on programmable logic components. Derived from the algorithms and architectures outlined in Chapters 5, 7, and 8, the configuration of the 109 utilized FPGAs (see Chapter 3) determines its functionality and performance. To define the architectures implemented in the FPGAs, the hardware description language VHDL[1] is used. It allows to unambiguously specify the individual functional modules of a design at different levels of abstraction. A comprehensive introduction to VHDL as well as a reference guide can be found in [51]. Valuable optimization techniques are described in [46].

While the GTU configuration system enables each FPGA in the system to have an individual hardware design, a small number of different designs is desirable to allow for careful optimization and verification of each design. Corresponding to the partitioning illustrated in Figure 5.6, three different FPGA designs are required:

- The TMU design performs all processing that can be done on the data of a single detector stack. This includes the majority of the trigger computations as well as high-bandwidth event buffering.
- The SMU design handles trigger and raw data on the level of a supermodule and implements the interface to DAQ and DCS. It controls the associated TMUs according to incoming trigger messages.
- The TGU design executes all TRD-global trigger processing, which can include high-level track analysis. It implements the interface to the CTP.

The VHDL source code of the GTU designs is stored in a central repository managed by a version control system (SVN). At present, the VHDL sources comprise 67 000 lines of custom code. A large fraction of the sources (22 k lines) is common to all three designs, the remainder is specific to TMU (25 k), SMU (14 k), or TGU (6 k). In addition, test benches and support files for the verification of the individual modules add up to another 20 000

[1] VHDL: VHSIC (Very-High-Speed Integrated Circuits) hardware description language

Design	Slices	BRAMs	Description
TMU	38 878 (92 %)	173 (46 %)	Tracking and event buffering
SMU	21 099 (50 %)	92 (24 %)	Readout and trigger handling
TGU	10 950 (25 %)	84 (22 %)	Trigger interface to CTP

Table B.1: Resource usage of the GTU FPGA designs

lines of source code. The TMU trigger design including track recognition (cf. Chapter 7) and transverse momentum reconstruction (cf. Chapter 8) consists of 11 k lines of VHDL code.

To reliably handle a design of this complexity, careful verification of each subunit is mandatory. For this purpose, the VHDL simulator *ModelSim SE 6.2c* of *Model Technology* is used. It is able to simulate the behavior of individual functional blocks of the designs as well as the complete top-level models and their interactions. Because the simulator provides a convenient way to study the statistic behavior of the trigger computations, it is also used to analyze the timing performance of the TMU design as presented in Section 9.3.

B.2 FPGA Synthesis Results

To produce the configuration data for the FPGAs, the hardware description in VHDL is *synthesized* and mapped to the utilized FPGA components by the *Xilinx ISE/EDK 9.2* suite of applications in a custom Makefile-based build flow.[2] The resulting FPGA configuration data files can be used to remotely program all FPGAs in the GTU as well as the configuration flash PROMs.

The Makefile-based build flow allows for a fully automated build system. By periodically synthesizing the designs based on the latest revision of the VHDL source code in a nightly build system, the validity of the source code in the GTU code repository is monitored and the availability of current configuration files is guaranteed.[3]

When mapping a VHDL description to an FPGA, the utilization of resources is a measure of the size of the design. In this application, the most important measures are the number of occupied general logic resources (*slices*) and embedded memory blocks (*BRAMs*). Table B.1 lists the resource usage information resulting from a synthesis for the GTU's Xilinx Virtex-4 FX FPGAs.[4] While the TMU design utilizes a large fraction of the available resources, the SMU and TGU FPGAs have resources remaining, which allows for future developments such as an invariant mass selection implemented in the TGU.

[2] An overview of the build flow is given in [33]

[3] At present (SVN revision `r1232`), building the three designs takes approximately three hours on a dual-core Intel Pentium D at 3.0 GHz running Linux 2.6.17.

[4] All numbers correspond to the GTU designs at SVN revision `r1232`.

TMU design component	Slices	BRAMs
Online tracking	20 216 (42 %)	78 (21 %)
Event buffering	11 327 (22 %)	14 (4 %)
Embedded PowerPC system	6277 (12 %)	69 (18 %)
Infrastructure	1058 (3 %)	12 (3 %)
Total	38 878 (92 %)	173 (46 %)

Table B.2: Resource usage of the individual TMU FPGA design components. The design utilizes 92 % of the logic slices in a Xilinx Virtex-4 FX100. In addition, 16 global clocks and the following special resources are used: 4 DCMs, 2 PMCDs, 12 MGTs (cf. [87]).

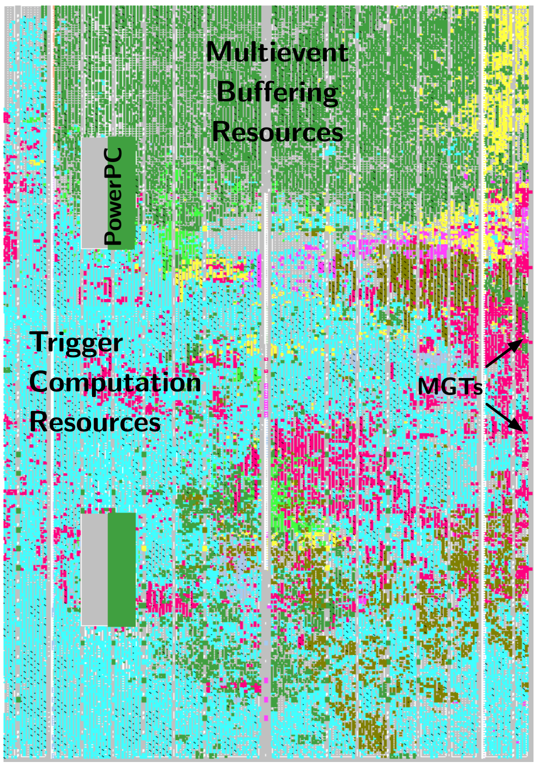


Figure B.1: Visualization of the resource usage of the TMU design mapped to the Xilinx Virtex-4 FX-100. The design in total utilizes 92 % of the device's slice resources.

The size of the individual TMU FPGA design components is indicated in Table B.2. The trigger design performing track matching and reconstruction as discussed in Chapters 7 and 8 is the largest design component utilizing 42 % of the available slices. The size of the event buffering component results mainly from the large number of pipeline flip-flops required by the wide data path. Figure B.1 shows a snapshot of the TMU design mapped to the Xilinx Virtex-4 FX-100.

B.3 Detector Simulation Environment

The detector systems of the ALICE experiment use a programming environment named *AliRoot* for simulation, reconstruction and analysis [11]. AliRoot utilizes the *Root* system[5] as the basis for all applications by mapping the functionality and geometry of the ALICE detectors to Root classes. Root is a C++-based object-oriented programming environment for data analysis and simulation. It is specifically geared towards the large amounts of data generated at the LHC. With the help of the built-in C++ interpreter, C++ can not only be used as the programming language for own classes, but also as a fast scripting language. The numerous Root classes are documented in [21].

The analyses presented in Chapter 9 are all based on custom classes and scripts implemented in AliRoot. The specific C++ code to analyze the test beam data comprises approximately 12 000 lines. In addition, Sections 9.2.1 and 9.2.2 include results from a conceptual simulation of the tracking design using synthetically generated random collision events from a HIJING[6] event generator enriched by Υ particles, which decay into a pair of high-energy electron and positron.

[5]Root is available as free software. It can be downloaded at `http://root.cern.ch` as source code and as precompiled binary for various architectures.
[6]HIJING: Heavy Ion Jet Interaction Generator

List of Acronyms

8P8C	8 Position 8 Contact *A type of communications connector commonly used in Ethernet.*
ACORDE	ALICE Cosmic Ray Detector
ADC	Analog-to-digital converter
AdvancedTCA	Advanced Telecommunications Computing Architecture
ALICE	A Large Ion Collider Experiment
API	Application programming interface
ASIC	Application-specific integrated circuit
ATLAS	A Toroidal LHC ApparatuS
BER	Bit error rate
BGA	Ball grid array
BRAM	Block RAM
CERN	European Organization for Nuclear Research *The European Organization for Nuclear Research is the world's largest particle physics laboratory. The acronym CERN originally stood for* Conseil Européen pour la Recherche Nucléaire *(European Council for Nuclear Research)*
CMC	Common Mezzanine Card
CMS	Compact Muon Solenoid
CompactPCI	Compact Peripheral Component Interconnect *An industrial computer bus standard, utilizing the Eurocard form factor.*
COTS	Commercial off-the-shelf *Computer products that are ready-made and available for sale to the general public.*
CPU	Central processing unit
CTP	Central Trigger Processor
DAQ	Data acquisition

DCB	Detector Control System Board
DCI	Digitally controlled impedance
DCM	Digital clock manager
DCS	Detector Control System
DDL	Detector Data Link *Data transmission system based on fiber-optical links, developed for readout of the ALICE detectors.*
DDR	Double data rate *A signaling mode for data on a computer bus. Data is transferred on both rising and falling edges of the clock signal.*
DDR2	Double data rate two *A technology used for high speed memory access. DDR2 is based on DDR, but typically operated at higher speeds.*
DIM	Distributed Information Management
DSP	Digital signal processor
ECS	Experiment Control System
EIA	Electronic Industries Alliance
EMCal	Electromagnetic Calorimeter
FIFO	First In, First Out
FMD	Forward Multiplicity Detector
FPGA	Field-programmable gate array
FSM	Finite state machine
GA	Geographic address
GTU	Global Tracking Unit
HIJING	Heavy Ion Jet Interaction Generator
HLT	High-Level Trigger
HMPID	High Momentum Particle Identification
HSTL	High-speed transceiver logic
IBIS	Input Output Buffer Information Specification
IC	Integrated circuit
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force

I²C	Inter-Integrated Circuit *A multi-master serial computer bus used to attach low-speed peripherals to a system.*
I/O	Input/output
IP	Internet Protocol
ISC	In-System Configuration
ITS	Inner Tracking System
JCOP	Joint Controls Project
JTAG	Joint Test Action Group *Common name for the IEEE 1149.1 standard (Standard Test Access Port and Boundary-Scan Architecture).*
L0	Level-0 *Primary trigger level of ALICE, issued* 1.4 µs *after the interaction.*
L1	Level-1 *Secondary trigger level of ALICE, issued* 6.5 µs *after the interaction.*
L2	Level-2 *Tertiary trigger level of ALICE, issued approximately* 80 µs *after the interaction.*
LED	Light-emitting diode
LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty
LHCf	Large Hadron Collider forward
LTU	Local Tracking Unit
LUT	Lookup table
LVCMOS	Low-voltage CMOS
LVDS	Low-voltage differential signaling *An electrical signaling system that transmits two different voltages which are compared at the receiver.*
MCM	Multi-chip module
MGT	Multi-gigabit transceiver
OPB	On-Chip Peripheral Bus
OS	Operating system
PASA	Preamplifier/Shaper

PCA	Printed circuit assembly *A PCB populated with electronic components.*
PCB	Printed circuit board
PCI	Peripheral Component Interconnect *A computer bus standard.*
PHOS	Photon Spectrometer
PLB	Processor Local Bus *A parallel bus system used for processors.*
PMC	PCI Mezzanine Card
PMCD	Phase-matched clock divider
PMD	Photon Multiplicity Detector
PowerPC	Power Performance Computing *A RISC microprocessor architecture.*
PPP	Point-to-Point Protocol
PROM	Programmable read-only memory
PS	Proton Synchrotron
QCD	Quantum chromodynamics
QGP	Quark-gluon plasma
RAM	Random access memory
RFC	Request for Comments *Document published by the Internet Engineering Task Force (IETF) addressing Internet standards*
RHIC	Relativistic Heavy Ion Collider *A heavy-ion collider located at Brookhaven National Laboratory.*
SCADA	Supervisory control and data acquisition
SD	Secure Digital *A format of flash-based memory cards.*
SDHC	Secure Digital High Capacity
SDRAM	Synchronous dynamic random access memory
SFP	Small form-factor pluggable
SIU	Source Interface Unit *Transmitter module for an ALICE DDL*
SMBus	System Management Bus

SMI++	State Management Interface *A multi-platform framework for distributed control systems developed at CERN.*
SMU	Supermodule Unit
SPI	Serial Peripheral Interface Bus
SPICE	Simulation Program with Integrated Circuit Emphasis
SPS	Super Proton Synchrotron
SRAM	Static random access memory
SSO	Simultaneous switching output
SSTL	Stub series terminated logic
TCP	Transmission Control Protocol *A communications protocol that is part of the Internet protocol suite.*
TDR	Technical Design Report
TGU	Trigger Generation Unit
TMU	Track Matching Unit
TOF	Time of Flight
TOTEM	TOTal Elastic and diffractive cross section Measurement
TPC	Time Projection Chamber
TRAP	Tracklet Processor
TRD	Transition Radiation Detector
TTC	Timing, Trigger and Control *A distribution system for timing, trigger and control signals developed for particle physics experiments at the LHC.*
TTCrx	TTC receiver
U	Rack unit *Measurement of height within racks. One rack unit corresponds to a height of* 44.45 mm.
UART	Universal asynchronous receiver/transmitter
VHDL	VHSIC (Very-High-Speed Integrated Circuits) hardware description language
VMEbus	VERSAmodule Eurocard bus *A computer bus standard, physically based on Eurocard sizes.*
XMD	Xilinx Microprocessor Debugger

XMU	TMU or SMU
XSVF	Xilinx Serial Vector Format
ZDC	Zero Degree Calorimeter

Bibliography

[1] A Large Ion Collider Experiment (ALICE) at CERN LHC. Web Site. URL http://aliceinfo.cern.ch/

[2] ALICE Collaboration. *Technical Proposal for A Large Ion Collider Experiment at the CERN LHC*. CERN/LHCC 95-71. CERN/LHCC, Geneva, December 1995. ISBN 92-9083-077-8.

[3] ALICE Collaboration. *Technical Design Report of the High Momentum Particle Identification Detector*. CERN/LHCC 98-19. CERN/LHCC, Geneva, August 1998. ISBN 92-9083-134-0.

[4] ALICE Collaboration. *Technical Design Report of the Inner Tracking System (ITS)*. CERN/LHCC 99-12. CERN/LHCC, Geneva, June 1999. ISBN 92-9083-144-8.

[5] ALICE Collaboration. *Technical Design Report of the Photon Spectrometer (PHOS)*. CERN/LHCC 99-4. CERN/LHCC, Geneva, March 1999. ISBN 92-9083-138-3.

[6] ALICE Collaboration. *Technical Design Report of the Time of Flight System (TOF)*. CERN/LHCC 2000-12. CERN/LHCC, Geneva, February 2000.

[7] ALICE Collaboration. *Technical Design Report of the Time Projection Chamber*. CERN/LHCC 2000-001. CERN/LHCC, Geneva, January 2000. ISBN 92-9083-155-3.

[8] ALICE Collaboration. *Technical Design Report of the Transition Radiation Detector*. CERN/LHCC 2001-021. CERN/LHCC, Geneva, 2001. ISBN 92-9083-184-7.

[9] ALICE Collaboration. *Technical Design Report of the Trigger, Data Acquisition, High-level Trigger and Control System*. CERN/LHCC 2003-062. CERN/LHCC, Geneva, 2004. ISBN 92-9083-217-7.

[10] ALICE Collaboration. *Technical Design Report on Forward Detectors: FMD, T0 and V0*. CERN/LHCC 2004-025. CERN/LHCC, Geneva, September 2004. ISBN 92-9083-229-0.

[11] ALICE Collaboration. *Technical Design Report of the Computing*. CERN/LHCC 2005-018. CERN/LHCC, Geneva, 2005. ISBN 92-9083-247-9.

[12] ALICE Collaboration, Alessandro, B., Antinori, F., Belikov, J. A., Blume, C., Dainese, A., Foka, P., Giubellino, P., Hippolyte, B., Kuhn, C., et al. ALICE: Physics Performance Report, Volume II. *Journal of Physics G: Nuclear and Particle Physics*, 32(10):1295–2040, 2006.
URL `http://stacks.iop.org/0954-3899/32/1295`

[13] ALICE Collaboration, Carminati, F., Foka, P., Giubellino, P., Morsch, A., Paic, G., Revol, J.-P., Safarík, K., Schutz, Y., and Wiedemann, U. A. ALICE: Physics Performance Report, Volume I. *Journal of Physics G: Nuclear and Particle Physics*, 30(11):1517–1763, 2004.
URL `http://stacks.iop.org/0954-3899/30/1517`

[14] Amsler, C., et al. (Particle Data Group). Review of Particle Physics. *Physics Letters*, B667:1+, 2008.
URL `http://pdg.lbl.gov`

[15] Andronic, A. Space Charge in Drift Chambers Operated with the Xe,CO2(15%) Mixture. *Nucl. Instrum. Meth.*, A525:447, 2004.

[16] Andronic, A. Updated Multiplicity Expectations in the ALICE Experiment, 2008. Private communication.

[17] Angelov, V., Catanescu, V., Ciobanu, M., Lesser, F., Lindenstruth, V., and Schneider, R. ALICE TRD Trigger Architecture. In *TRDs for the 3rd millennium – Workshop on advanced Transition Radiation Detectors for accelerator and space applications*. Bari (Italy), September 2001.

[18] Armesto, N. Review of Monte Carlo Methods for Particle Multiplicity Evaluation. *J. Phys. Conf. Ser.*, 5:219–229, 2005. doi:10.1088/1742-6596/5/1/020.

[19] Bock, R. K. and Vasilescu, A. *The Particle Detector BriefBook*. Springer, Berlin, Heidelberg, 1998. ISBN 3-540-64120-3.
URL `http://www.cern.ch/Physics/ParticleDetector/BriefBook/`

[20] Borghini, N. and Wiedemann, U. A. Predictions for the LHC Heavy-ion Program. *Journal of Physics G: Nuclear and Particle Physics*, 35(2):023001 (31pp), 2008.
URL `http://stacks.iop.org/0954-3899/35/023001`

[21] Brun, R. and Rademakers, F. ROOT Reference Guide, 2003.
URL `http://root.cern.ch/root/Reference.html`

[22] CERN. Web Site.
URL `http://www.cern.ch`

[23] de Cuveland, J. *Entwicklung der globalen Spurrekonstruktionseinheit für den ALICE-Übergangsstrahlungsdetektor am LHC (CERN)*. Diploma thesis, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg, 2003.

[24] de Cuveland, J., Rettig, F., Angelov, V., and Lindenstruth, V. An FPGA-based High-speed, Low-latency Trigger Processor for High-energy Physics. *International Conference on Field Programmable Logic and Applications (FPL) 2008*, pages 293–298, September 2008.

[25] de Cuveland, J., Rettig, F., and Kirsch, S. ALICE TRD DAQ Data Format. Internal Note, 2008.

[26] De Cataldo, G., Augustinus, A., Boccioli, M., Chochula, P., and Jirdén, L. S. Finite State Machines for Integration and Control in ALICE. In *Proceedings of the 2007 International Conference on Accelerator and Large Experimental Physics Control Systems*, pages 650–652. Knoxville, Tennessee (USA), October 2007.

[27] Dolgoshein, B. Transition Radiation Detectors. *Nucl. Instrum. Meth.*, A326:434–469, 1993.

[28] Franek, B. and Gaspar, C. SMI++ – State Machine Interface.
URL `http://cern.ch/smi`

[29] Frühwirth, R., Strandlie, A., Waltenberger, W., and Wroldsen, J. A Review of Fast Circle and Helix Fitting. *Nucl. Instrum. Meth.*, A502:705–707, April 2003.

[30] Gareus, R. *Slow Control Serial Network and its Implementation for the Transition Radiation Detector*. Diploma thesis, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg, October 2002.

[31] Gaspar, C. Introduction to DIM.
URL `http://cern.ch/dim/dim_intro.html`

[32] Gaspar, C., Dönszelmann, M., and Charpentier, P. DIM, a Portable, Light Weight Package for Information Publishing, Data Transfer and Inter-process Communication. In *Proceedings of the 2000 International Conference on Computing in High Energy and Nuclear Physics (CHEP)*. Padova, Italy, February 2000.

[33] Gerlach, T. *Development of an Embedded Linux System for the Global Tracking Unit of the ALICE TRD at the LHC (CERN)*. Diploma thesis, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg, September 2008.

[34] Gottschalk, D. DCS Board Documentation Page, May 2006.
URL `http://www.kip.uni-heidelberg.de/ti/DCS-Board/current/`

[35] Green, D. *The Physics of Particle Detectors*. Cambridge University Press, Cambridge, 2000. ISBN 0-521-66226-5.

[36] Grupen, C. *Teilchendetektoren*. BI-Wiss.-Verl., Mannheim, Leipzig, Wien, Zürich, 1993. ISBN 3-411-16571-5.

[37] Gutfleisch, M. *Digitales Frontend und Preprozessor im TRAP1-Chip des TRD-Triggers für das ALICE-Experiment am LHC (CERN)*. Diploma thesis, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg, 2002.

[38] Gutfleisch, M. *Local Signal Processing of the ALICE Transition Radiation Detector at LHC (CERN)*. Ph.D. thesis, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg, 2006.

[39] IEEE. IEEE Standard for a Common Mezzanine Card (CMC) Family. *IEEE Std 1386-2001*, 2001.

[40] IEEE. IEEE Standard Physical and Environmental Layers for PCI Mezzanine Cards (PMC). *IEEE Std 1386.1-2001*, 2001.

[41] IEEE. IEEE Standard Test Access Port and Boundary-Scan Architecture. *IEEE Std 1149.1-2001*, pages i–200, 2001. doi:10.1109/IEEESTD.2001.92950.

[42] IEEE. IEEE Standard for In-System Configuration of Programmable Devices. *IEEE Std 1532-2002 (Revision of IEEE Std 1532-2001)*, pages i–141, 2003.

[43] JCOP Framework Working Group. *Joint Controls Project (JCOP) Framework Sub-Project – Guidelines and Conventions*. CERN-JCOP-2000-008. CERN IT/CO, Geneva, July 2007.
URL `http://www.cern.ch/itcobe/Projects/Framework/Documentation/`

[44] Jirdén, L. S. B-Field Sensitive Components. ALICE Technical Board Presentation, February 2004.

[45] Jovanovic, P. Trigger Output Logic – Hardware Guide for the Front-end Designers, March 2007.
URL `http://epweb2.ph.bham.ac.uk/user/pedja/alice/ctp/`

[46] Kilts, S. *Advanced FPGA Design – Architecture, Implementation, and Optimization*. Wiley-Interscience, Hoboken, NJ, 2007. ISBN 0-470-05437-9.

[47] Kirsch, S. *Development of the Supermodule Unit for the ALICE Transition Radiation Detector at the LHC (CERN)*. Diploma thesis, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg, 2007.

[48] Klapdor-Kleingrothaus, H. V. and Zuber, K. *Teilchenastrophysik*. Teubner, Stuttgart, 1997. ISBN 3-519-03094-2.

[49] Klein, J. *Commissioning of and Preparations for Physics with the Transition Radiation Detector in A Large Ion Collider Experiment at CERN*. Diploma thesis, Universität Heidelberg, Physikalisches Institut, Heidelberg, 2008.

[50] Kleinknecht, K. *Detektoren für Teilchenstrahlung*. Teubner, Stuttgart, third edition, 1992. ISBN 3-519-23058-5.

[51] Lehmann, G., Wunder, B., and Selz, M. *Schaltungsdesign mit VHDL*. Franzis, Poing, 1994. ISBN 3-7723-6163-3.

[52] Lesser, F. *Entwurf und Realisierung eines vierfach MIMD Prozessor Mikrochips für eine Anwendung in der Hochenergiephysik*. Ph.D. thesis, Universität Mannheim, Kirchhoff-Institut für Physik, Heidelberg, 2002.

[53] Lesser, F., Angelov, V., de Cuveland, J., Lindenstruth, V., Reichling, C., Schneider, R., and Schulz, M. W. A MIMD Multi Threaded Processor. In *Proceedings of the 13th IEEE Hot Chips Conference*. Palo Alto (USA), August 2001.

[54] Lindenstruth, V. and Kisel, I. Overview of Trigger Systems. *Nucl. Instrum. Meth.*, A535:48–56, August 2004.

[55] PCI Industrial Computer Manufacturers Group. *CompactPCI Hot Swap Specification*. PCI Industrial Computer Manufacturers Group, Wakefield, Mass., January 2001.

[56] PCI Industrial Computer Manufacturers Group. *CompactPCI Specification PICMG 2.0 R3.0*. PCI Industrial Computer Manufacturers Group, Wakefield, Mass., 2002.

[57] Peitzmann, T. *Kernmaterie unter extremen Bedingungen – Die experimentelle Suche nach dem Quark-Gluon-Plasma*. Universität Münster, 1997. Habilitationsschrift.

[58] Perkins, D. H. *Introduction to High Energy Physics*. Cambridge University Press, Cambridge, fourth edition, 2000. ISBN 0-521-62196-8.

[59] Povh, B., Rith, K., Scholz, C., and Zetsche, F. *Teilchen und Kerne: eine Einführung in die physikalischen Konzepte*. Springer, Berlin, Heidelberg, fifth edition, 2001. ISBN 3-540-65928-5.

[60] Relativistic Heavy Ion Collider (RHIC) Experimental Collaborations. *Hunting the Quark Gluon Plasma – Results from the First 3 Years at RHIC*. BNL-73847-2005. Brookhaven National Laboratory, Upton, NY, April 2005. Formal Report.

[61] Rettig, F. *Entwicklung der optischen Auslesekette für den ALICE-Übergangsstrahlungsdetektor am LHC (CERN)*. Diploma thesis, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg, 2007.

[62] Rettig, F. First TRD-triggered Cosmics, July 2008. Email to alice-trd@cern.ch.

[63] Richter, M., Alme, J., Alt, T., Bablok, S., Campagnolo, R., Frankenfeld, U., Gutierrez, C., Keidel, R., Kofler, C., Krawutschke, T., et al. The Control System for the Front-end Electronics of the ALICE Time Projection Chamber. *Nuclear Science, IEEE Transactions on*, 53(3):980–985, 2006. ISSN 0018-9499. doi:10.1109/TNS.2006.874726.

[64] Rubin, G. and Soós, C. *ALICE Detector Data Link – Hardware Guide for the Front-end Designers*. Internal note, European Organization for Nuclear Research, September 2007.

[65] Rubin, G., Vande Vyvre, P., and Soós, C. *ALICE Detector Data Link – Interface Control Document*. Internal Note ALICE-INT-2004-018, European Organization for Nuclear Research, July 2004.

[66] Schmeling, S. M. *Joint PVSS and JCOP Framework Course – Course Manuscript.* CERN-JCOP-2004-016. CERN IT/CO, Geneva, March 2004.
URL `http://www.cern.ch/itcobe/Projects/Framework/Documentation/`

[67] Schneider, R. *Entwicklung des Triggerkonzeptes und die entsprechende Implementierung eines 200 GB/s Auslesenetzwerks für den ALICE-Übergangsstrahlungsdetektor.* Ph.D. thesis, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg, 2003.

[68] Schneider, R., Angelov, V., Gutfleisch, M., Gareus, R., Lesser, F., Lindenstruth, V., Reichling, C., and Torralba, G. 64k Networked Multi-Treaded Processors and their Real-Time Application in High-Energy Physics. In *The 6th World Multiconference on Systemics, Cybernetics and Informatics.* Orlando, Florida (USA), 2002.

[69] Schuh, M. *Entwicklung und Implementierung eines Steuersystems für die Trigger- und Datenausleselogik des ALICE-Übergangsstrahlungsdetektors am LHC (CERN).* Diploma thesis, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg, 2007.

[70] SD Card Association, Technical Committee. *SD Specifications, Part 1, Physical Layer, Simplified Specification, Version 2.00.* SD Card Association, San Ramon, California (USA), 2006.
URL `http://sdcard.org`

[71] SFF Committee. INF-8074i Specification for SFP (Small Formfactor Pluggable) Transceiver, May 2001.
URL `http://sffcommittee.com/ie/Specifications.html`

[72] SFF Committee. SFF-8472 Specification for Diagnostic Monitoring Interface for Optical Transceivers, June 2007.
URL `http://sffcommittee.com/ie/Specifications.html`

[73] Simoncelli, E. Least Squares Optimization, July 2003.

[74] Simpson, W. The Point-to-Point Protocol (PPP). RFC 1661 (Standard), July 1994. Updated by RFC 2153.
URL `http://www.ietf.org/rfc/rfc1661.txt`

[75] Soltveit, H.-K. Development of a Fast TRD Pre-Amplifier Shaper. In *GSI Scientific Report.* 2004.

[76] Swoboda, D. and Taureg, H. *Measurements of the Magnetic Stray Field for the ALICE Muon Dipole and L3 Solenoid.* Internal Note ALICE-INT-2006-001, European Organization for Nuclear Research, March 2008.

[77] Texas Instruments. *PTH05010W: 15-A, 5-V Input Non-Isolated Wide-Output Adjust Power Module.* POLA Point-of-Load Alliance, Texas Instruments, Post Office Box 655303, Dallas, Texas 75265, December 2003.

[78] Vulpescu, B. Physics with the ALICE TRD. In *TRDs for the 3rd millennium – Workshop on advanced Transition Radiation Detectors for accelerator and space applications*. Bari (Italy), September 2001.

[79] Vulpescu, B. Online Pion Rejection (GTU), 2003.
URL `http://www-aix.gsi.de/~vulpescu/AliPw/Lhood.ps`

[80] Vulpescu, B. Online Tracking: First Step to LTU and GTU Algorithms, 2003.
URL `http://www-aix.gsi.de/~vulpescu/AliPw/Drawings.ps`

[81] Vulpescu, B. Simulation of the ALICE TRD Online Tracking Performance, 2003. Private communication.

[82] Wilk, A. *Elektronen-Pionen-Separation im ALICE-TRD*. Diploma thesis, Universität Münster, Institut für Kernphysik, Münster, 2004.

[83] Wong, C.-Y. *Introduction to High-energy Heavy-ion Collisions*. World Scientific, Singapore, London, 1994. ISBN 981-02-0264-4.

[84] XILINX. *Virtex-4 PCB Designer's Guide*, ug072 (1.1) edition, September 2004.
URL `http://www.xilinx.com/support/documentation/user_guides/ug072.pdf`

[85] XILINX. *Virtex-4 Packaging and Pinout Specification*, ug075 (v3.1) edition, June 2007.
URL `http://www.xilinx.com/support/documentation/user_guides/ug075.pdf`

[86] XILINX. *Virtex-4 FPGA Configuration User Guide*, ug071 (v1.10) edition, April 2008.
URL `http://www.xilinx.com/support/documentation/user_guides/ug071.pdf`

[87] XILINX. *Virtex-4 FPGA User Guide*, ug070 (v2.4) edition, April 2008.
URL `http://www.xilinx.com/support/documentation/user_guides/ug070.pdf`