Multi-Event Buffering and Local Event Building for the ALICE Transition Radiation Detector

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim
Fachbereich Informatik und Mathematik
der Johann Wolfgang Goethe-Universität
in Frankfurt am Main

von
Stefan Kirsch
aus Landau in der Pfalz

Frankfurt (2017)
D(30)

Vom Fachbereich Informatik und Mathematik
der Johann Wolfgang Goethe-Universität
als Dissertation angenommen

Dekan	Prof. Dr. Andreas Bernig
Gutachter	Prof. Dr. Volker Lindenstruth
	Prof. Dr. Uwe Brinkschulte
Datum der Disputation	26. April 2018

Multi-Event Buffering and Local Event Building for the ALICE Transition Radiation Detector

As an integral part of ALICE, the dedicated heavy ion experiment at CERN's Large Hadron Collider, the Transition Radiation Detector (TRD) contributes to the experiment's tracking, triggering and particle identification. Central element in the TRD's processing chain is its trigger and readout processor, the Global Tracking Unit (GTU). The GTU implements fast triggers on various signatures, which rely on the reconstruction of up to 20 000 particle track segments to global tracks, and performs the buffering and processing of event raw data as part of a complex detector readout tree.

The high data rates the system has to handle and its dual use as trigger and readout processor with shared resources and interwoven processing paths require the GTU to be a unique, high-performance parallel processing system. To achieve high data taking efficiency, all elements of the GTU are optimized for high running stability and low dead time.

The solutions presented in this thesis for the handling of readout data in the GTU, from the initial reception to the final assembly and transmission to the High-Level Trigger computer farm, address all these aspects. The presented concepts employ multi-event buffering, in-stream data processing, extensive embedded diagnostics, and advanced features of modern FPGAs to build a robust high-performance system that can conduct the high-bandwidth readout of the TRD with maximum stability and minimized dead time. The work summarized here not only includes the complete process from the conceptual layout of the multi-event data handling and segment control, but also its implementation, simulation, verification, operation and commissioning. It also covers the system upgrade for the second data taking period and presents an analysis of the actual system performance.

The presented design of the GTU's input stage, which is comprised of 90 FPGA-based nodes, is built to support multi-event buffering for the data received from the 18 TRD supermodules on 1080 optical links at the full sender aggregate net bandwidth of 2.16 Tbit/s. With careful design of the control logic and the overall data path, the readout on the 18 concentrator nodes of the supermodule stage can utilize an effective aggregate output bandwidth of initially 3.33 GiB/s, and, after the successful readout bandwidth upgrade, 6.50 GiB/s via 18 optical links. The high possible readout link utilization of more than 99 % and the intermediate buffering of events on the GTU helps to keep the dead time associated with the local event building and readout typically below 10 %. The GTU has been used for production data taking since start-up of the experiment and ever since performs the event buffering, local event building and readout for the TRD in a correct, efficient and highly dependable fashion.

Contents

1 Introduction

Physics research over the past century has yielded detailed insights into the underlying structure of matter, its elementary building blocks and the basic forces governing them. Experiments have become ever more complex, growing from tabletop experiments of the early 1900s to those employing detectors housed in immense underground caverns. They have led to a comprehensive theory, which is not complete, but has been extremely successful in its description of the subatomic world: the Standard Model of particle physics.

The Standard Model describes twelve fundamental fermions organized in two families, quarks and leptons. Four gauge bosons –force-carrying particles interchanged between the elementary particles– explain three of the four fundamental forces. In 2013, a particle then recently discovered at CERN was confirmed to be the Higgs boson, a missing piece in the Standard Model that explains why some particles have mass and whose existence had been predicted since the 1960s.

The strong interaction, one of the three fundamental forces included in the Standard Model, is mediated on a sub-nuclear scale by the exchange of gluons between the constituents of hadrons, the quarks. Quantum chromodynamics as theory behind the strong force attributes a color charge to both quarks and gluons. It has two distinct features: Confinement describes the fact that quarks and gluons are only observed as constituents of color-neutral hadrons. Isolated color-charged particles have never been observed. The second feature, asymptotic freedom, implies that the strong coupling strength vanishes to zero for asymptotically high energies, which allows quarks to move freely in close confinement.

With the energy density exceeding a critical value, quantum chromodynamics calculations predict a phase transition for strongly interacting nuclear matter from a state of hadronic constituents to a deconfined plasma, where quarks and gluons are freed of the effects of strong interaction. This phase, the quark-gluon plasma, whose existence has recently been confirmed, and its characteristics are studied in ultra-relativistic heavy ion collisions [19, 92]. Using a set of complementary detectors grouped around an interaction point, the aim is to track, identify and characterize all emerging particles. By obtaining a complete picture of the final state of the collision, e. g., like illustrated in Figure 1.1, conclusions on the initial state and its evolvement following the interaction may be drawn.

Identifying the remnants of the primordial matter in heavy ion collisions presents a challenging task and imposes a number of special design requirements. Subdetectors with very different characteristics in terms of particle detection technique, intrinsic dead time, readout speed and others, are combined to form a detector system and achieve, together, the desired performance with respect to particle identification and tracking. A large number of particles emerges from a collision. Worst case estimates for the charged particle densities in A Large Ion Collider Experiment (ALICE) were $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}\eta = 8000$ at the time of design [6], corresponding to about 20 000 primary and secondary tracks in the detector. This places a demand for high spatial resolution and leads to highly granular, complex detectors with a large number of detection elements. In combination with the interaction rates delivered by modern particle accelerators, data rates after digitization exceed several tens of TB/s.

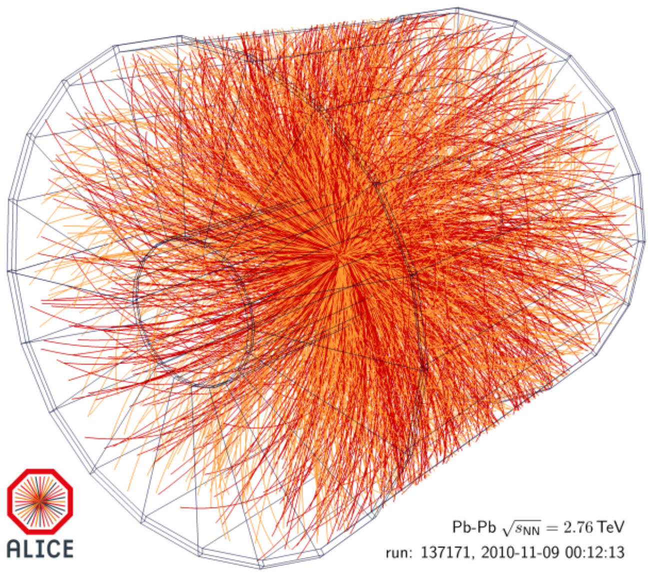

ALICE
Pb-Pb $\sqrt{s_{\mathrm{NN}}} = 2.76$ TeV
run: 137171, 2010-11-09 00:12:13

Figure 1.1: Event display, reconstructed from an ALICE Pb-Pb collision at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$. From [110].

Common, easily accessible phenomena often have already been studied in the past and modern experiments search for rare events exhibiting signatures of interesting, potentially new physics. Therefore, they implement sophisticated mechanisms capable of quickly identifying and selecting these rare events in an effort to reduce the enormous data rate to a level that can be recorded to mass storage, typically few GB/s. These trigger systems are generally organized in multiple hierarchical stages of increasing accuracy, but also latency, and are built using a wide range of technologies from custom developed integrated circuits to re-programmable hardware, e. g., FPGAs, to computer farms [89, 96].

Considering the time required to collect sufficient statistics for a thorough physics analysis –and also the cost of operation for a particle accelerator such as the Large Hadron Collider– it is essential to maximize the availability of each system. For detectors, this includes two aspects: robustness and stability when participating in data taking, as intermissions and restarts typically come with a non-negligible overhead, and minimizing the intrinsic dead time due to the detector's detection principle and/or readout.

The ALICE Transition Radiation Detector (TRD) is a prime example of such a complex, modern detector. It contributes to the trigger stage starting the acquisition of data with the experiment's main, but comparably slow, tracking device. These trigger contributions rely on the fast online reconstruction of data from almost 1.2 million detector channels and identification of electron tracks with high transverse momentum therein, as those prove interesting in the study of quark-gluon plasma. As part of the offline analysis, its data increases the length of the tracking arm, which in turn increases the resolution. To achieve its goals, the TRD uses a massively parallel two-stage approach: More than 250 000 dedicated CPUs process data already on the detector and feed an FPGA-based tracking and readout unit acting as second processing stage [6, 9].

This thesis presents the ALICE TRD Global Tracking Unit (GTU) with focus on event buffering, local event building and its high performance readout system. The GTU is a three-layer hierarchical system comprised of 109 nodes. Using partial track information preprocessed on the detector front-end, it performs the global reconstruction on up to 20 000 track segments and calculates, based on the reconstructed tracks, various Level-1 stage trigger contributions within tight timing requirements of 2 µs [99, 41, 80]. In its role as readout unit, the GTU implements a robust buffering scheme for multiple events to minimize the overall system dead time and is capable of handling input data rates of 2.16 Tbit/s received via up to 1080 optical input links. Steered by the decisions of the experiment's central trigger system, it conducts the building of buffered data into local event fragments and their transmission to the high-level trigger and data acquisition systems via a total of 18 optical output links.

Due to the challenging low latency and high bandwidth requirements originating from its tracking, trigger and readout applications, the GTU is a custom designed, FPGA-based system. Flexibility and versatility, mostly attributable to the use of powerful FPGAs, have allowed to expand the GTU's functionality in the course of the physics campaign and adapt to evolving requirements. Originally conceived primarily as a black box, it has been developed into a system that features extensive integrated monitoring, testing and diagnostics capabilities that prove invaluable in both commissioning phases and day-to-day operation in a complex detector environment.

The GTU has been in constant operation since the initial LHC fills. At the time of writing, it had, with the firmware designs presented here, concluded the first long Run1 data taking campaign and an upgrade of the readout system to meet the new performance requirements of Run2 was successfully completed.

The thesis is organized as follows: An overview of ALICE, its subsystems, and the TRD, as far as relevant to set the scope, as well as an introduction to the GTU hardware is presented in Chapter 2. Chapter 3 focuses on the system's dead time sources and implementation of the GTU's supermodule stage, whose tasks include interfacing of the experiment's trigger system and control of a segment. The buffering of events on nodes of the track matching stage, the building of event fragments, their readout and the Run2 upgrades are presented in Chapter 4. Chapter 5 presents the integration of the GTU into the detector control framework of ALICE, while Chapter 6 highlights the process of commissioning a design to actually participate in data taking with a look at the HDL simulation framework, integrated test infrastructure and diagnostics. Finally, Chapter 7 presents measurements of the GTU system performance and a simulation model that allows to quickly evaluate the performance in a variety of different running scenarios.

2 ALICE and the Transition Radiation Detector

A Large Ion Collider Experiment (ALICE), the dedicated heavy ion experiment at the Large Hadron Collider, aims to study the equilibrium and non-equilibrium physics of strongly interacting matter with a special focus on the creation, dynamical evolution and characteristics of a new state of matter, the quark-gluon plasma. This chapter presents a brief summary of ALICE, and specifically the Transition Radiation Detector (TRD) with its tracking, trigger and readout unit, the Global Tracking Unit (GTU).

2.1 Large Hadron Collider

The Large Hadron Collider (LHC) [48, 95] at CERN is a superconducting, two-ring particle accelerator, installed in a tunnel of almost 27 km circumference underneath the French-Swiss border area close to Geneva. With beam energies of 7 TeV and a peak luminosity in the order of $10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ for proton-proton (p-p) and $10^{27}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ for heavy ion (Pb-Pb) operation, it is, at the time of writing, the world's most powerful particle collider.

After a staged pre-acceleration in CERN's accelerator complex, two counter-rotating particle beams are injected into the LHC at 450 MeV, then accelerated to target energy and brought to collision at the interaction points. Particles in the beams are not distributed homogeneously, but are aggregated to bunches at certain virtual positions, called buckets,[1] on the ring. Of the 3564 potential bunch positions that constitute the LHC orbit, the necessity of an abort gap allows to occupy a total of 2808 per beam. The bunch filling scheme of the two beams determines which bunches are brought to a collision at which interaction points.

Four interaction regions are surrounded by the four major LHC experiments:

- ATLAS is a general purpose detector, whose main goals include investigation and characterization of the Higgs boson, investigation of CP violation and supersymmetry, and precision measurements of the top quark [22].
- CMS is a second general purpose detector, which aims to study the Higgs boson, dark matter and new phenomena in physics. The physics goals of ATLAS and CMS overlap, allowing them to independently cross-check each other's findings [39].

[1]The number of buckets is $h \approx f_{\mathrm{RF}} \cdot 2\pi R/c \approx 35640$, where $f_{\mathrm{RF}} = 400\,\mathrm{MHz}$ is the frequency of the accelerating RF cavities and $2\pi R = 26\,659\,\mathrm{m}$ is the circumference of the LHC. The distance of two neighboring buckets corresponds to 2.49 ns. The bunch spacing is 10 buckets, i. e., 24.95 ns.

- LHCb is a detector targeted at the investigation of electroweak and heavy flavor physics, especially precision measurements of heavy b quarks and CP violation [88].
- ALICE is a dedicated heavy ion detector at the LHC, which focuses on the study of the Quark-Gluon Plasma and strongly interacting matter [8].

Following a successful Run1 campaign from 2009 to 2013 with collisions of protons at center-of-mass energies up to $\sqrt{s} = 3.5\,\mathrm{TeV}$ and heavy ions at up to $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$, the LHC delivers in Run2 p-p collisions at $\sqrt{s} = 7\,\mathrm{TeV}$ and Pb-Pb collisions at $\sqrt{s_{NN}} = 5.5\,\mathrm{TeV}$.

2.2 A Large Ion Collider Experiment

The distinct design of the ALICE detector distinguishes it from the other general purpose detectors at the LHC. Driven by the requirements of heavy ion collisions at LHC energies, i. e., predicted charged particle densities of up to $\mathrm{d}N_{\mathrm{ch}}/\mathrm{d}\eta = 8000$, it is a finely segmented detector that provides good resolution over a wide range of transverse momenta $p_\perp$ and has exceptional particle identification (PID) capabilities. Consequently, it allows resolving the complex collision topology to identify and track emerging particles even in an environment with up to 20 000 tracks per event in the detector acceptance. At LHC design luminosity, interaction rates of up to 8 kHz in Pb-Pb are expected. ALICE is optimized for operation at such primary rates and aimed, in Run1, for readout rates of around 200 Hz in Pb-Pb and up to 1.4 kHz in p-p operation. An in-depth review of the physics objectives, the physical observables and design considerations leading to the final ALICE architecture is given in [11].

The ALICE detector, depicted in Figure 2.1, consists of several subdetectors with complementary characteristics installed in an underground cavern at LHC Point 2. The subdetectors of the barrel section measure leptons, hadrons and photons that are produced in particle interactions in the mid-rapidity region $|\eta| < 0.9$. The solenoid L3 magnet encloses these central barrel detectors and generates a homogeneous, medium-strength magnetic field of 0.5 T in z-direction, i. e., parallel to the beam axis. A forward muon arm, comprised of absorber, dipole magnet and spectrometer, complement the system and allow for detection and identification of muons at large rapidities of $2.4 \leq \eta \leq 4$.

Among the most important subdetectors are:

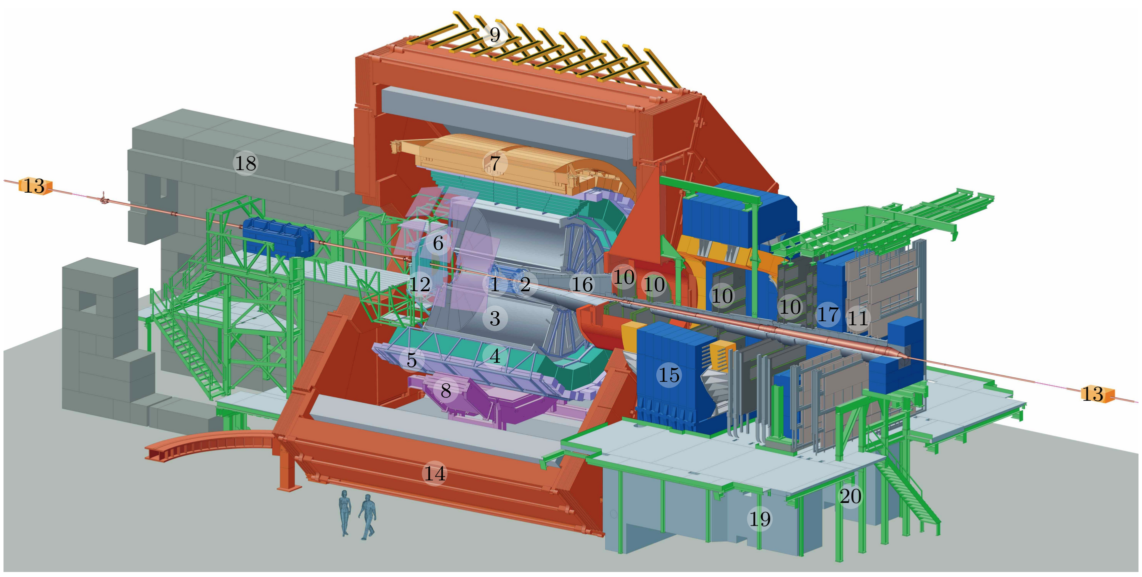
- Inner Tracking System (ITS)
  The ITS consists of six cylindrical layers of silicon pixel, strip and drift detectors. It surrounds the interaction point at radii of 0.04–0.44 m with a rapidity coverage of $|\eta| < 0.9$. The ITS enables precise measurements of primary and secondary vertices and improves, in combination with the TPC, the overall momentum and angle resolution for high-$p_\perp$ particles [3, 11].
- Time Projection Chamber (TPC)
  The TPC, the main tracking detector, is a hollow-cylindrical field cage of $90\,\mathrm{m}^3$ volume filled with a $\mathrm{Ne}/\mathrm{CO}_2$ mixture. It encloses the ITS at a radial position at 0.85–2.50 m and extends ± 2.5 m along the beam axis. Multi-wire proportional

chambers with more than 500 000 readout pads are mounted on the cylinder end caps. Large gaseous detectors are inherently slow due to the drift of electrons and ions. In case of the TPC, the associated insensitivity period is about 280 µs. The TPC provides PID and momentum measurements of charged particles from very low to high $p_\perp$ [16, 11, 91, 100].

- Transition Radiation Detector (TRD)
  The TRD surrounds the TPC at radii of 2.90–3.70 m. It provides electron identification at high-$p_\perp$ and fast tracking and trigger capabilities for Level-1 stage trigger contributions.

- Time-of-Flight Detector (TOF)
  The TOF is a detector based on multi-gap resistive plate chambers installed at a radial position of 3.70–3.99 m. The TOF allows for particle identification in the intermediate momentum range from 0.20–2.50 GeV/c and, in combination with other detectors, enables an event-by-event identification of pions, kaons and protons [5, 11].

- Photon Spectrometer (PHOS)
  The PHOS is an electromagnetic spectrometer comprised of five detector modules of lead-tungstate crystals, which are placed adjacent to TOF at a radius of 4.60 m.



1. ITS (Inner Tracking System)
2. FMD (Forward Multiplicity Detector), V0, T0
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 (Photon Spectrometer)
9. ACORDE (ALICE Cosmic Ray Detector)
10. Muon Tracking Stations
11. Muon Trigger
12. PMD (Photon Multiplicity Detector)
13. ZDC (Zero Degree Calorimeter)
14. L3 Magnet
15. Dipole Magnet
16. Absorber
17. Muon Absorber Wall
18. Concrete Shielding
19. C-Area
20. GTU (Global Tracking Unit)

Figure 2.1: The ALICE detector at CERN LHC in Run1. The red L3 magnet encloses the central barrel detectors. Adapted from [2]

PHOS gives the ability to identify photons and measure the momentum with high spatial and energy resolution [4, 11].

- Electromagnetic Calorimeter (EMCal)
  The EMCal, placed at a radius of approximately 4.50 m adjacent to TOF, enhances the capabilities for jet as well as high-$p_\perp$ photon and electron measurements. It provides contributions to the L0 and L1 trigger stages [1].

Five online systems (Central Trigger Processor (CTP), Data Acquisition (DAQ), High-Level Trigger (HLT), Experiment Control System (ECS) and Detector Control System (DCS)) control and monitor the subdetectors, administer the data taking, and perform the selection of events and archiving of data to mass storage.

2.3 Trigger System

LHC-era detectors produce raw data rates in the order of several ten TB/s, while the sustainable bandwidth to the mass storage systems is in the order of few GB/s. Main task of the trigger system is to select events exhibiting signatures of interesting, typically rare physics, thereby lowering the data rates to levels manageable by mass storage and enriching rare events in the recorded data sample.

The subdetectors that form the ALICE detector have different characteristics in terms of dead time, trigger contributions, achievable readout rates and data volumes produced. Trigger system and ECS therefore support grouping of detectors with matching characteristics (or those, that need to be read out together), into readout partitions. Within each readout partition, one or more trigger clusters, each representing different trigger conditions, are used to trigger and read out the partition's detectors in a synchronized fashion.

2.3.1 ALICE Trigger Levels

The event selection in ALICE is split into levels of increasing complexity. Level-0 (L0) to Level-2 (L2) are implemented by dedicated hardware. The HLT computer farm adds a fourth versatile selection and processing layer. The timing and significance of the different trigger levels, see Table 2.1, reflects the unique design of the ALICE detector with the comparably slow, large-volume TPC as main tracking detector.

The L0 trigger serves as fast start signal for most detectors. It reaches the detector front-ends about 1 µs after the interaction. Main selection criteria at this stage are centrality and multiplicity of the event.

In case of a Level-1 (L1) trigger, which would be issued at a distinct time about 7–8 µs after the interaction, the trigger sequence is continued. The L1 stage, whose increased latency allows for a more detailed analysis of the event than possible at L0, serves as start signal for data acquisition in the TPC. Main contributors to the L1 stage are the trigger algorithms of the TRD, see Section 2.7.3, and EMCal.

The L2 trigger stage represents a past-future protection to avoid an excessive overlap of events within the drift volume of the TPC. Its latency corresponds to the TPC drift time of approximately 90 µs.

Level	System	Distribution	Latency	Rate limit
L0	CTP	TTC A	~ 1 µs	100 kHz
L1	CTP	TTC A+B	~ 8 µs	few kHz
L2	CTP	TTC B	~ 90 µs	1 kHz
HLT	HLT	–	several ms	

Table 2.1: Trigger levels and their latency with respect to the interaction in ALICE. Trigger levels L0 to L2 are generated and processed by dedicated hardware, cf. Section 3.4. The HLT is realized as computer farm.

2.3.2 Trigger System Hardware

The Central Trigger Processor (CTP) [68] receives the input from the trigger detectors and generates trigger sequences to steer, in a synchronized fashion, the operation of a detector partition. It supports a multitude of trigger classes for a given readout partition. Available scalers and the mixing of different triggers allow to adjust the rates of the individual trigger stages to match the corresponding processing capabilities of the detectors in the readout partition.

The CTP consists of a central crate plus several detector TTC partitions, one for each subdetector. The central crate, see Figure 2.2, houses the processing boards for the hardware trigger levels L0 to L2, to which the corresponding contributions from the trigger detectors are connected. The busy processing board evaluates the busy state of the subdetectors in the various readout partitions. Interface boards to the DAQ and fan-out boards interfacing the detector TTC partitions, together with fan-in or switch boards where required, complete the setup. The detector TTC partitions each consist of a Local Trigger Unit (LTU) and a TTCex [108] board. The latter interfaces the Timing, Trigger and Control (TTC) system [36], which provides two channels for the transmission of trigger-related information to the attached subdetector.

The LTU [67] receives detector-specific trigger information from the central CTP logic and generates corresponding trigger sequences for the detector. Furthermore, it relays the busy signal from each detector to the busy processing board. In standalone mode, which is mainly used during testing and commissioning, the LTU is capable of emulating simple trigger sequences for operation independent of the central CTP.[2]

[2]Generation of the trigger sequences is limited, e. g., no conditionally aborted sequences are possible, as the trigger inputs are handled centrally by the CTP and not the LTU.

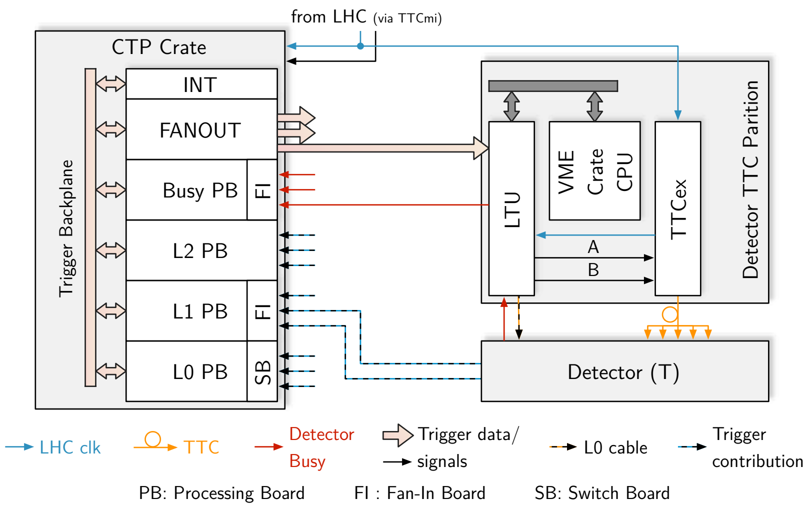


Figure 2.2: Overview of the ALICE trigger system (CTP and detector TTC partition). The CTP itself consists of several types of processing boards interconnected by a custom backplane. Each (sub-)detector is controlled via a TTC partition that is responsible for the encoding and transmission of trigger sequences to the detector. Contributions from detectors with trigger capability are connected directly to the corresponding trigger input processing boards in the central CTP crate.

2.3.3 Event Identification

The LHC distributes a 40.079 MHz clock signal that is synchronous with the crossing of bunches in the interaction points and an orbit signal of 40.079 MHz/3564 = 11.246 kHz to the experiments. Both are forwarded to the subdetectors via the central CTP crate and respective TTC partitions. In case the LHC clock is not available or stable, e. g., in the accelerator ramp-up phase, a stable local 40 MHz clock is transmitted instead. LHC clock and orbit signal form the basis for a synchronized detector operation and event identification on the detector front-end.

2.4 High-Level Trigger and Data Acquisition

The HLT receives the event data and implements the final event selection and/or compression stage. Subject to the decision of the HLT's trigger algorithms, the DAQ then administers the building of event fragments into global events and their archival on permanent data storage [7]. HLT and DAQ are realized as computer farms using commodity servers augmented with custom-developed interface cards.

The connection of detector electronics to HLT/DAQ is established via Detector Data Links (DDLs) [101, 102]. Each DDL is comprised of a Source Interface Unit (SIU), the optical fiber, and a Destination Interface Unit (DIU). For the first generation DDL, DIU and SIU are dedicated add-on cards with FPGA and transceiver, which are mounted directly on the detector front-end and on the Readout Receiver Cards (RORCs), respectively. With the second generation DDL2, SIU and DIU are supplied as HDL modules for direct integration into the respective FPGAs of detector electronics and RORCs.

Local Data Concentrators (LDCs) are servers equipped with RORCs, which function as input nodes to receive event fragments sent via the attached DDLs. At the end of Run1, at total of 480 DDL inputs to 175 LDCs constituted the interface to the subdetectors [33]. On the event building stage, Global Data Concentrators (GDCs) combine event fragments to global events that contain the event data from all participating subdetectors. Event fragments originating from one particular interaction are uniquely identified by the transmitting DDL, period number, orbit number and bunch count. Global events are encapsulated as ROOT [31] objects and stored on a local transient data storage as ROOT files for subsequent migration to off-site permanent data storage at a maximum sustainable rate in the order of 1–2 GB/s. Libraries provided as part of the DAQ's software package, DATE [12], give access to the event data streams on LDC or GDC level, which is used by the GTU to implement tools for online verification.

2.5 Experiment and Detector Control

The mandate of the control system is to facilitate an efficient, reliable and safe way of operating the experiment. The challenge is to integrate a large number and variety of custom-developed (sub-)systems, sensors, devices and interfaces –internal and external–, each with their individual infrastructure and requirements, into a control system that meets the needs of both experts and regular users. The two major building blocks of the ALICE control system are the ECS and the DCS [7].

2.5.1 Experiment Control System

The ECS interfaces the LHC machine control and coordinates the operation of the ALICE online systems. Not all subsystems are always fully operational, nor is a participation of every subsystem required in every scenario. The ECS therefore allows for a partitioning of the detector into subdetector groups that conjointly operate in runs. A run is a certain period of time, in which the detectors of a readout partition take data under a set of well-defined conditions and configurations[3] that reflect the current objective. A fatal error in any of the involved subsystems/subdetectors causes an abort of the run.

A run can be divided into three major phases:

- Start-of-Run phase
  Online systems are started and the data taking conditions at the start of the run are archived.

[3]Conditions and configurations here include the collision scheme delivered by the accelerator, participating subdetectors and active entities thereof, selected trigger scheme, stable ambient temperatures, etc.

- Data taking phase
  Triggering, acquisition, event building and online monitoring are active.
- End-of-Run phase
  Global event building is finalized, the online systems are stopped and the conditions and status at the end of the run are archived and/or logged.

In regular, standard operation, the experiment is operated by a shift crew. The ECS/DAQ operator is in charge of starting and stopping runs with the various readout partitions. During preparation for a run, global run parameter sets are selected via the ALICE Configuration Tool (ACT) [28] and made available to the subsystems, which in turn are configured to reflect the chosen configuration.

2.5.2 Detector Control System

The DCS [93, 7] covers all aspects of remotely controlling and monitoring the ALICE subdetectors, their respective infrastructure as well as other experimental equipment. Due to the large number of subsystems that have to be integrated into the control system, a high degree of scalability and modularity is required. In ALICE, the DCS is implemented as hierarchical structure of Finite State Machines (FSMs), where the top node summarizes a view of the overall experiment. To change the state of the detector, the DCS generates a number of transition commands that are propagated through the FSM hierarchy –or a subset of it– and ultimately effect the desired state change in all currently selected subsystems.

ALICE uses WinCC [106], a commercial SCADA system, in combination with JCOP [35], which is a set of guidelines and software for control systems developed for the LHC experiments, as frameworks for the implementation of the DCS supervisory layer and top-level FSM. Individual subsystems implement, under consideration of the their fundamental architecture and operational principles, subsystem-specific child FSM models, which are included to the FSM hierarchy by an appropriate mapping of states.

2.6 Transition Radiation Detector

The TRD [6, 8, 20] is designed as trigger and tracking detector. It provides electron identification at high $p_\perp$, contributes to the experiment-wide L1 trigger decision and provides event data to extend the tracking arm from ITS through TPC, thus increasing the tracking resolution. A summary of its most important aspects is presented in the following.

In its role as trigger detector, the TRD reconstructs high-$p_\perp$ charged particle tracks to serve as basis for a number of L1 trigger contributions to the CTP. Reconstruction of these tracks within the tight latency requirements of the L1 decision of about 7 µs from data of more than $1.15 \cdot 10^6$ analog detector channels presents a significant challenge. At this stage, the TRD produces data at a rate exceeding 15 TB/s. The TRD hence uses a massively parallelized approach divided in local and global tracking with processing paths optimized for low latency. During local tracking, more than 65 000 Multi-Chip Modules (MCMs) located on the detector chambers perform a parameterization of the particle tracks traversing a single detector chamber. A subsequent global tracking stage reconstructs track segments to tracks that traverse a detector stack. Global tracking, the trigger algorithms processing the reconstructed tracks, and the detector readout are implemented by the TRD's FPGA-based trigger and readout processor, the Global Tracking Unit (GTU).

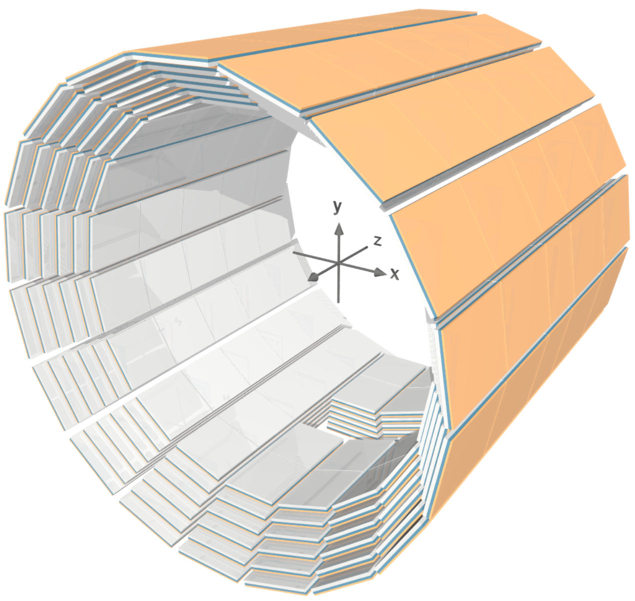
2.6.1 Structure

The TRD is a barrel shaped detector placed in-between TPC and TOF that covers full azimuth and a pseudorapidity range of $|\eta| < 0.9$. Its inner and outer radii are 2.9 m and 3.7 m, respectively. Following the overall segmentation of ALICE, the TRD is segmented into 18 supermodules in the azimuthal angle φ. Along the z-direction, each supermodule typically consists of five stacks, which in turn are each comprised of six layers of detector chambers, see Figure 2.3. To avoid worsening the PHOS detector performance by screening it with additional material, the installation of the central stacks of TRD supermodules 14, 15 and 16 was omitted ("PHOS holes").

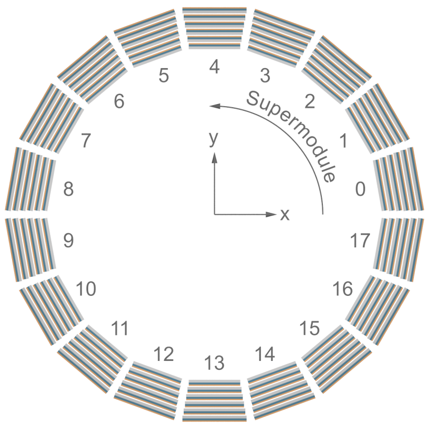
Central element of the on-detector processing electronics are the MCMs, which combine an analog Pre-Amplifier and Shaper (PASA) and a digital Tracklet Processor (TRAP) in one package. A single MCM can process 18 detector channels. A Readout Board (ROB), see Figure 2.4a, features 16 acquisition MCMs in a 4×4 arrangement and an additional MCM as Board Merger (BMM). One ROB per half-chamber is additionally equipped with an Half-Chamber Merger (HCM) to merge data from all ROBs and an Optical Readout Interface (ORI) that constitutes the half-chamber's link to the GTU.

All chambers possess 144 cathode pads (connected to eight MCMs on two ROBs) in y-direction, see Figure 2.4b. In z-direction, the number of pad rows is, depending on the type of readout board, either 12 for the C0 type or 16 for the C1 type, which results in 1728 and 2304 readout channels, respectively. Chambers of the central stack are of type LiC0, while the chambers of the other stacks are of type LiC1, with $i = 0..5$, leading to 12 different chamber types used in the construction of a supermodule [46]. Each chamber features two individual readout trees with an optical transmitter to the GTU to match the requirements for low-latency transmission of up to 40 parameterized 32-bit track segments during trigger readout [104].

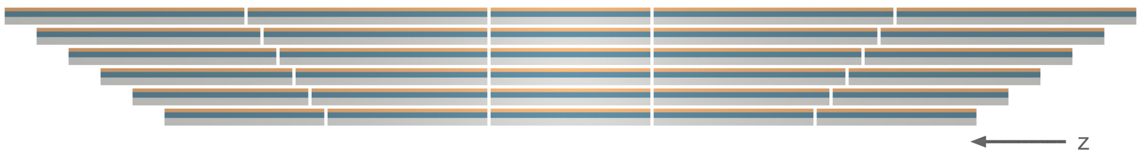
Table 2.2 summarizes the parameters for the full TRD in regular and in the PHOS hole configuration. The following chapters reference, if not denoted otherwise, the numbers for the regular TRD configuration as these are the relevant numbers with respect to the layout of the GTU.



(a) Foreshortened view of the 18 supermodules comprising the TRD, with PHOS holes in sectors 13, 14 and 15.



(b) Orthographic projection of the TRD supermodules onto the xy-plane.



(c) Orthographic projection of readout chambers of one TRD supermodule. Six layers of drift chambers form each of the five detector stacks.

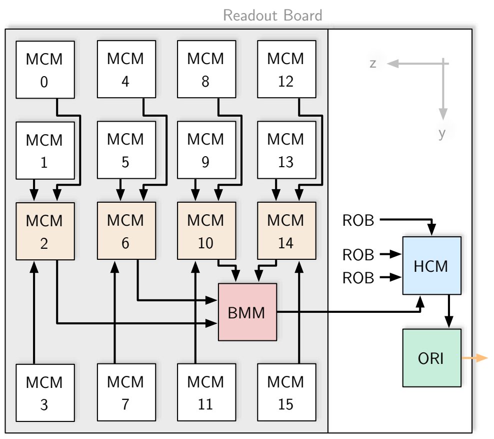
Figure 2.3: Geometry of the ALICE TRD.

2.6.2 Operating Principle

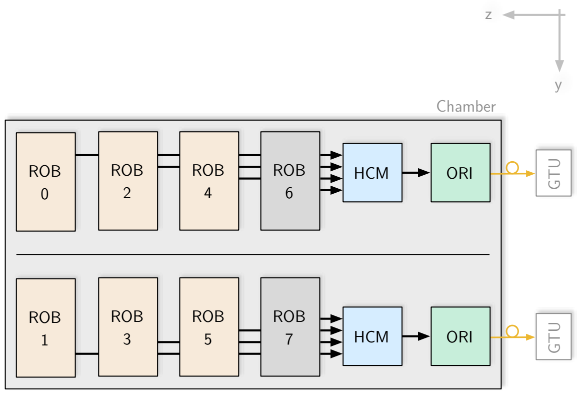
When a charged particle crosses the boundary of two materials with different refractive indices, transition radiation is generated. The energy of the transition radiation photon depends on the Lorentz factor $\gamma = \sqrt{1 - (\frac{v}{c})^2}$, and consequently only electrons or positrons generate transition radiation in the momentum range up to 100 GeV [20]. At comparable transverse momentum $p_\perp$, the TRD utilizes this fact to distinguish electrons from the heavier pions, which occur much more often in Pb-Pb collisions. The tracks of particles crossing the detector and the generated transition radiation photons are measured by a drift chamber.

Figure 2.5 illustrates the structure of a TRD chamber, which is composed of:

- A radiator (48 mm) to ensure a sufficient yield of transition radiation,
- the active volume filled with a Xe/CO_2 gas mixture, divided in drift region (30 mm) and amplification region (7 mm),



(a) Readout Board. One of the MCMs functions as BMM. Certain ROBs are additionally equipped with an MCM functioning as HCM and an ORI.



(b) System view of the readout tree of a C1-type chamber with eight ROBs. Chambers of the C0 type are equipped only with ROBs 0 to 5.

Figure 2.4: Components and structure of the TRD readout tree. Adapted from [104].

- a carbon fiber reinforced honeycomb support structure (23.16 mm) with the pad plane as detecting element on one side and the processing electronics mounted directly on the back.

Charged particles traversing the drift region ionize the gas along their path. For electrons, high-energy transition radiation photons are emitted in the radiator and additionally create large amounts of ionization close to the radiator/drift region border. Under the influence of the electric field, the electron clusters from ionization drift towards the amplification region, where the charge carriers are multiplied by an avalanche effect and subsequently measured as a signal on the cathode pads.

2.6.3 Local Tracking

The TRD Front-End Electronics (FEE) calculates parameterizations of the tracks traversing detector chambers, tracklets, in the timeframe prior to the L1 trigger. Parameterization is started by the arrival of a fast, TRD-specific wake-up signal[4] prior to the L0 trigger. The PASA shapes and amplifies the charge induced on the cathode pads before digitization at 10-bit 10 MHz by the ADCs. After a filter and preprocessor stage, the TRAPs calculate a parameterization of the track segments of high-$p_\perp$ charged particles traversing the chamber via linear regression [60, 104]. The calculation starts already during the drift time, which is in the order of 2 µs. The resulting 32-bit tracklet words, which contain information about

[4] In Run1, the wake-up was generated by the Pretrigger (PT) system, which implemented a fast minimum bias trigger based on input from the V0, T0 and TOF detectors [121, 103, 87, 79]. For Run2, the PT system is succeeded by the Level-Minus (LM) system for better integration with the CTP and reduced wake-up latency.

Detector		
Coverage		$-0.9 < \eta < 0.9$, $0 \leq \varphi \leq 2\pi$
Segmentation	in φ:	18 supermodules
	in z:	5 stacks
	in r:	6 layers
Number of stacks		90 / $87^{\dagger}$
Number of chambers		540 / $522^{\dagger}$
Number of ORIs		1080 / $1044^{\dagger}$
Number of MCMs		70 848 / 68 $976^{\dagger}$
Readout channels		1 181 952 / 1 150 $848^{\dagger}$
Digitization		10 bit at 10 MHz
Time bins		15–30 (configurable)
Chamber		
Segmentation	in y:	144 cathode pads
	in z:	12 / 16 pad rows (C0 / C1)
Readout channels		1728 / 2304 (C0 / C1)

Table 2.2: Summary of parameters for the TRD and a single readout chamber. Parameters marked by † denote the TRD PHOS hole configuration with three central stacks removed.

the PID, the pad row z, the deflection d_y and the intercept with the y-axis, are held in registers of the TRAP.

The tracklet readout mode used by the FEE for shipping tracklet data to the GTU prior to the L1 is optimized for lowest latency, but limited to small amounts of data.[5] The MCM hierarchy forms a tree of depth four, see Figure 2.4a. MCMs of layer 0 (white) function as data sources, while MCMs of layer 1 (yellow) are both data sources and elements of the readout tree. During the tracklet readout, which is implemented as unidirectional push, all MCMs of one layer simultaneously send their tracklets to the MCMs in the next layer, which provides sufficient buffer capacity. The BMM and HCM of layers 2 and 3 combine the data streams on one ROB or the half-chamber, respectively. Tracklets transmitted by the ORI to the GTU are sorted in ascending pad row order, which allows to start global tracking with arrival of the first tracklets.

2.6.4 Half-Chamber Event Data Readout

In case the current trigger sequence is confirmed by the arrival of the L1 trigger, raw event data, which at this point resides in the memories of the MCMs' filters, is moved off detector to event buffers in the GTU. The readout of the raw event data, which can be up to 675 B per MCM for an uncompressed configuration with 30 time bins, uses a handshake protocol.

[5]The tracklet readout mode is optimized for 40 homogeneously distributed 32-bit tracklets per chamber, which corresponds to 84.4 KiB for the full TRD. The absolute limit is four tracklets per MCM. For details, see [104].

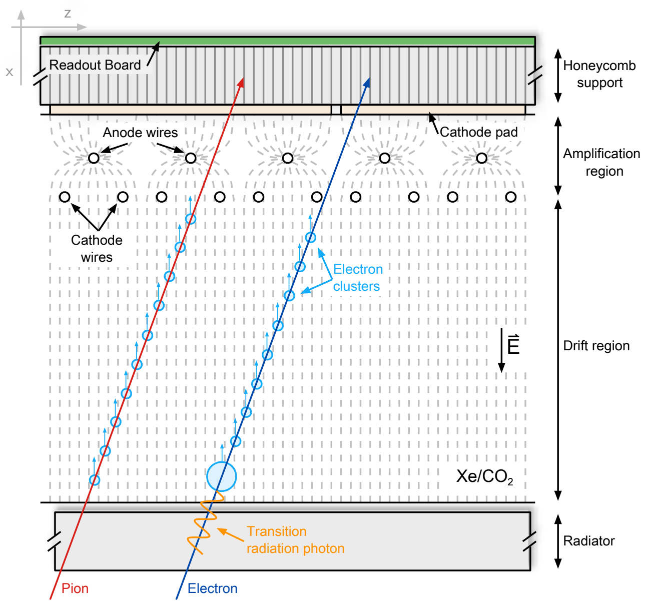


Figure 2.5: Pion and electron traversing a TRD chamber. Transition radiation photons cause an increased amount of ionization in the vicinity of the radiator material for light particles. Picture not to scale, adapted from [104].

In a first step, all MCMs transfer the data to their own, appropriately dimensioned FIFOs. Then the FIFOs of all MCMs in the readout tree are successively emptied in a defined readout order [104]. TRAPs and I/O cells of momentarily unused FIFOs are switched off to save power.

2.6.5 TRD Detector Control System

The TRD DCS ensures safe operation of the detector under all conditions and implements an interface to remotely control the detector in a way that it is usable by the shift crew as well as detector experts.

Figure 2.6 presents a simplified view of the underlying TRD DCS hardware architecture [93]. The operator nodes of the supervisory layer provide graphical interfaces and archiving functionality to the user. The process control layer consists of several worker node computers, which run Scientific Linux CERN (SLC) or WinCC sub-projects, and equipment to interface the various components of the detector. Entities of the field layer are distributed throughout the cavern and implement a suitable interface that enables remote control and monitoring of the respective equipment. Examples are the DCS boards (cf. Section 5.2) on the detector chambers, which implement a FEE server that communicates with the worker nodes via the TCP/IP-based Distributed Information Management (DIM) protocol, or the power supply units, which communicate via OPC or SNMP.

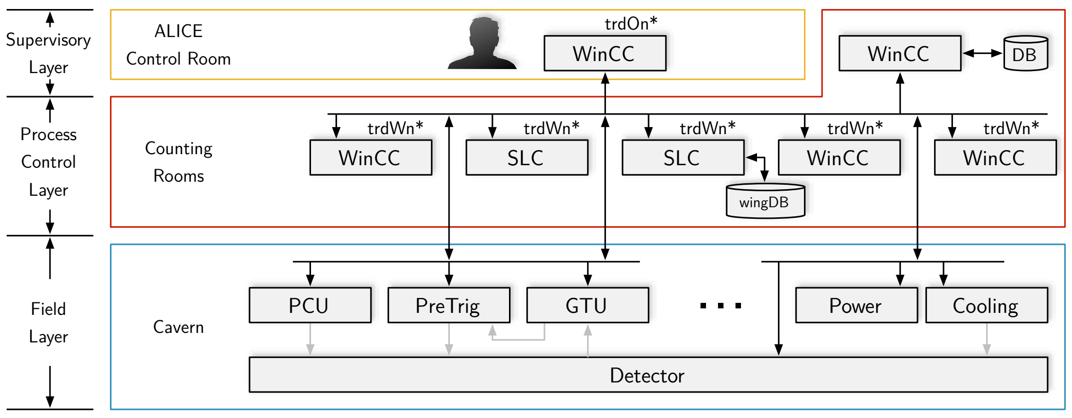


Figure 2.6: Illustration of the principle TRD DCS hardware architecture (simplified, refer to [93] for a detailed, more complete representation).

Following the general experiment guidelines, the operation of the TRD and its subunits is modeled as hierarchical tree of FSM-like objects. The top-level TRD FSM object [32], see Figure 2.7, maps the specific detector conditions (configuring for readout, ramping of the detector HV system, etc.) to FSM states in a way suitable for integration within the global DCS. Subunits are, e. g., the PT system, the gas system, the detector chambers, or the GTU.

2.7 Global Tracking Unit

The GTU is the central element in the trigger and readout chain of the TRD. It performs the global matching of up to 20 000 parameterized track segments, performs a full 3-D reconstruction of the tracks and runs several algorithms yielding the contributions to the experiment-wide L1 trigger within a time budget of approximately 2 µs. As second main task, it equips the detector with event buffering capability, orchestrates the local building of event fragments and facilitates readout of the fragments to the HLT via the DAQ with highest possible performance. This section introduces the GTU hardware architecture, which is presented in more detail in [41], and its components.

2.7.1 GTU Structure

Following the detector geometry, the $90 + 18 + 1$ FPGA nodes of the GTU are divided into 18 segments to handle the tracking, trigger and readout-related tasks of the associated supermodule. Each segment consists of one Supermodule Unit (SMU), which serves as concentrator and control node of the segment, and five Track Matching Units (TMUs).[6] The segments operate independently and in parallel, their operation is synchronized only by

[6]Four TMUs in segments with PHOS hole, where TMU2 is removed.

the trigger sequences issued by the CTP. The top-level output node Trigger Unit (TGU) implements the interface to the ALICE CTP.

The GTU's processing units are arranged in a three-layer hierarchy, see Figure 2.8. Tier 0 consists of 90 TMUs to receive tracklets and raw event data coming from the half-chambers. Twelve links from each of the five detector stacks of a supermodule connect to the five TMUs of a segment at an aggregate bandwidth of 150 Gbit/s.[7] High-bandwidth SRAMs on the TMUs provide buffer capacity for data of several events. A custom LVDS backplane connects the five TMUs to the SMU in the segment in a point-to-point fashion. Auxiliary connections are established using a standard CompactPCI backplane.

Tier 1 consists of eighteen SMUs. Each SMU interfaces the TTC system, generates control signals to steer the operation of the segment and administers the building of event fragments and their readout. The SMUs transmit the segments busy status and supermodule-level trigger information to the TGU.

Tier 2 comprises one processing node, the TGU. It implements the highest trigger stage, concentrates busy information from all segments and implements the output interface to the ALICE CTP.

[7]Segments with PHOS hole: 4 TMUs, 48 half-chambers/input links, 120 Gbit/s input bandwidth. PHOS hole configuration in total: 87 TMUs, 1044 half-chambers/input links, 2.61 Tbit/s input bandwidth.

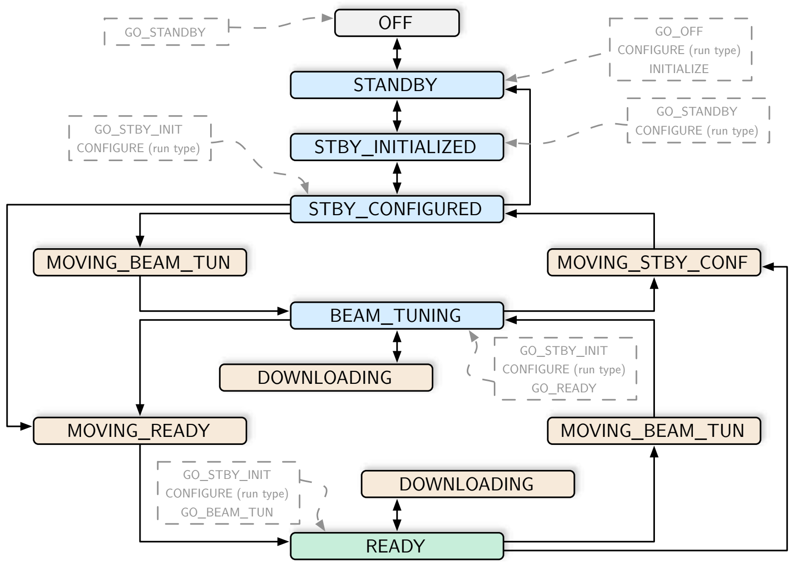


Figure 2.7: The TRD FEE top-level state diagram. Detector operation is facilitated by moving between the states, which each represent a different detector condition. Adapted from [32].

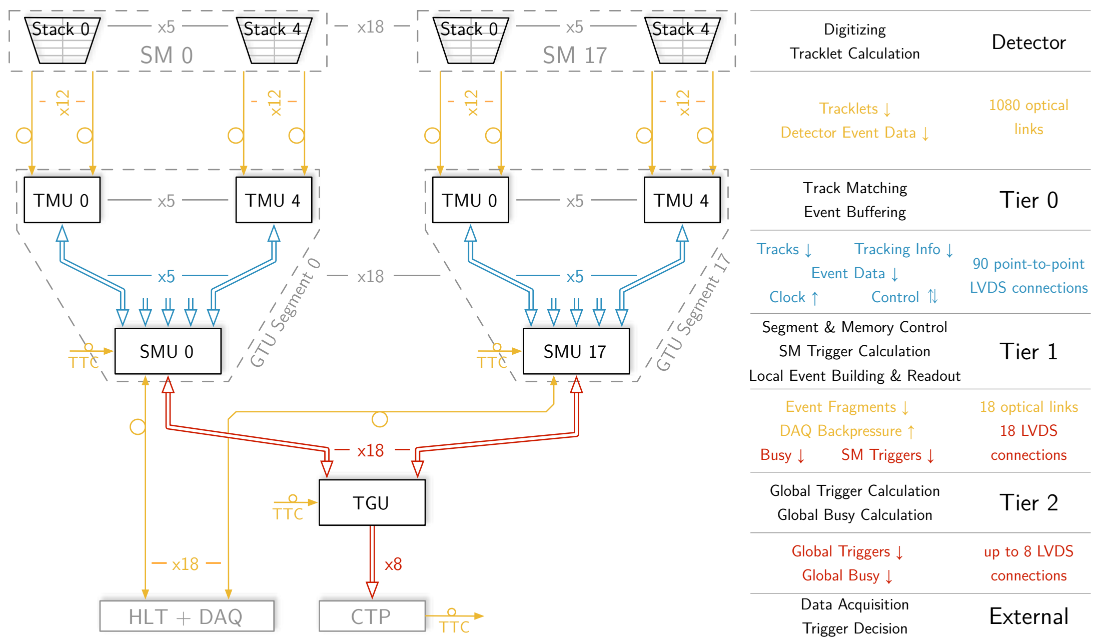


Figure 2.8: Structural overview of the GTU with three levels of hierarchy. Spatial data locality is exploited when reconstructing tracks from the tracklet data on Tier 0. Event data is buffered on Tier 0 at full receiver bandwidth and shipped to the HLT and DAQ via readout links located on Tier 1.

2.7.2 Processing Node Hardware

The different processing node types, TMU, SMU and TGU, are realized as CompactPCI boards. They share a common 14-layer PCB design, but differ in the components and add-on boards equipped.

Common Components Central component of all nodes, see Figure 2.9, is a high-density FPGA with high-speed serial transceivers that provides sufficient I/O, logic and clocking resources to establish system connectivity and implement complex FPGA designs. Important key figures of the utilized Xilinx Virtex-4 FX100 devices are listed in Table A.1. A platform flash PROM (Xilinx XCF32P) provides storage for the FPGA configuration bitstream, which is automatically fetched at power-up. Configuration ports of FPGA and PROM are daisy-chained to a JTAG chain accessible via the DCS board for remote programming. Each FPGA of the Virtex-4 FX series features two embedded PowerPC (PPC) hard-cores.

The nodes are 6 U in height. On the lower 3 U they connect to a standard CompactPCI backplane mounted in the crate for power distribution, system control and auxiliary signaling using the bus reserved signals. On the upper 3 U, the nodes are equipped with HM-Zd LVDS connectors, which enable point-to-point connections across the tiers. The custom-designed LVDS backplane provides sufficient bandwidth to ensure transmission of trigger information at low latency. The RocketIO MGT blocks in the FPGA are connected to SFP cages, which enables the use of robust, industry-standard modules to handle the electrical to optical signal conversion at line rates of up to 4.25 Gbit/s.

The nodes all feature two 36 × 512 kbit DDR SRAM modules. Sharing common clock and address signals, they provide a 72 bit parallel data interface[8] at 200 MHz DDR resulting in a maximum combined read/write bandwidth of 28.8 Gbit/s.

Each board is equipped with 64 MB DDR2 SDRAM and persistent mass storage in form of a 4 GB SDHC card. System health sensors for the temperatures of FPGA and PCB and the voltages are connected via I^{2}C. They can be read out by the FPGA or via the DCS board. The SFPs used in the GTU feature a diagnostics and monitoring controller, which is accessible via I^{2}C and connected to the FPGA using a series of I^{2}C multiplexers and port expanders.

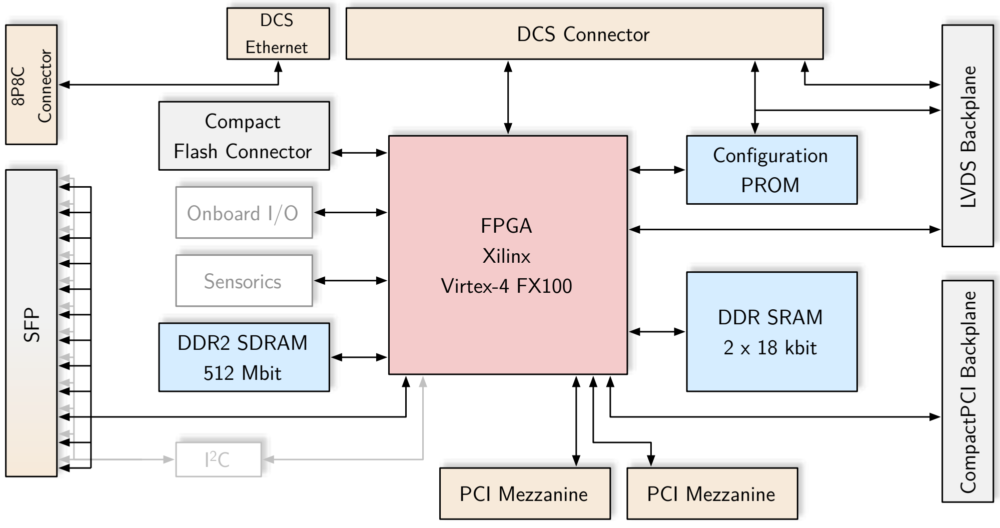


Figure 2.9: Schematic view of the architecture of a GTU processing node with a Xilinx Virtex-4 FPGA as the central component. Depending on the actual node type, different components are equipped, e. g., TMUs carry twelve cages for SFPs, while SMU and TGU boards carry only four. Components marked in yellow are not equipped or used on all types of boards.

Track Matching Unit Characteristic feature of a TMU board, depicted in Figure 2.10, are the 12 cages to mount optical transceiver modules. They connect to the ORI on the detector half-chambers and receive 8b10b-encoded data at a line rate of 2.5 Gbit/s. The SRAM on the TMU serves as a dedicated buffer for event data.

Supermodule Unit In its role as control and concentrator node for a segment, the SMU, depicted in Figure 2.11, houses two daughter boards on the back side, DCS board and SIU. The DCS board, see Section 5.2, is a flash-based standalone embedded Linux

[8]Including parity bits.

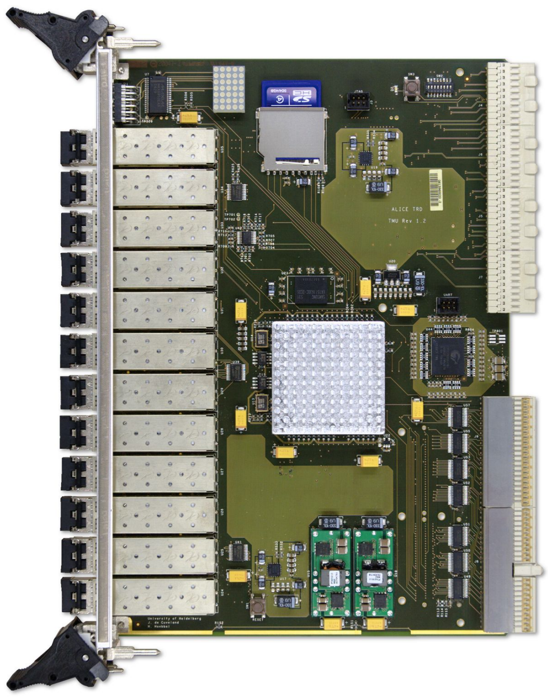


Figure 2.10: Top view of the TMU processing node with 12 optical input links.

system developed at the University of Heidelberg [58] widely used in ALICE detector front-ends for control, monitoring and configuration purposes. It is connected to the SMU via pin headers and implements the connection to the trigger and clock distribution system [36] and to the DCS network. The SIU, see Section 4.6.1, constitutes the FEE side of the DDL in Run1[9] and is mounted via PCI Mezzanine Card connectors. The SMU has point-to-point connections to the segment's five TMUs via the LVDS backplane. The connection to the TGU is established via an 8P8C modular connector, also located on the LVDS backplane. On the SMU, cages for four SFPs are equipped and connected to the Multi-Gigabit Transceivers (MGTs).

Trigger Unit The TGU is equipped almost identically to the SMU, but features a CTP interface board instead of the SIU. Instead of the LVDS backplane, it connects to a custom TGU backplane via the differential HM-Zd board-to-board connectors. Triggers and information about the segment status are received from the SMUs via length matched twisted-pair cables on 18 8P8C connectors located on this dedicated TGU backplane. The CTP interface consists of two[10] PCBs, both connected via the PCI Mezzanine Card connector, which provide eight differential ports of configurable direction that meet the requirements of the CTP [71].

[9] The SIU has become obsolete for Run2, see Section 4.7.

[10] Initially one PCB with five I/Os, the CTP interface was extended with a second PCB of three I/Os during Run1.

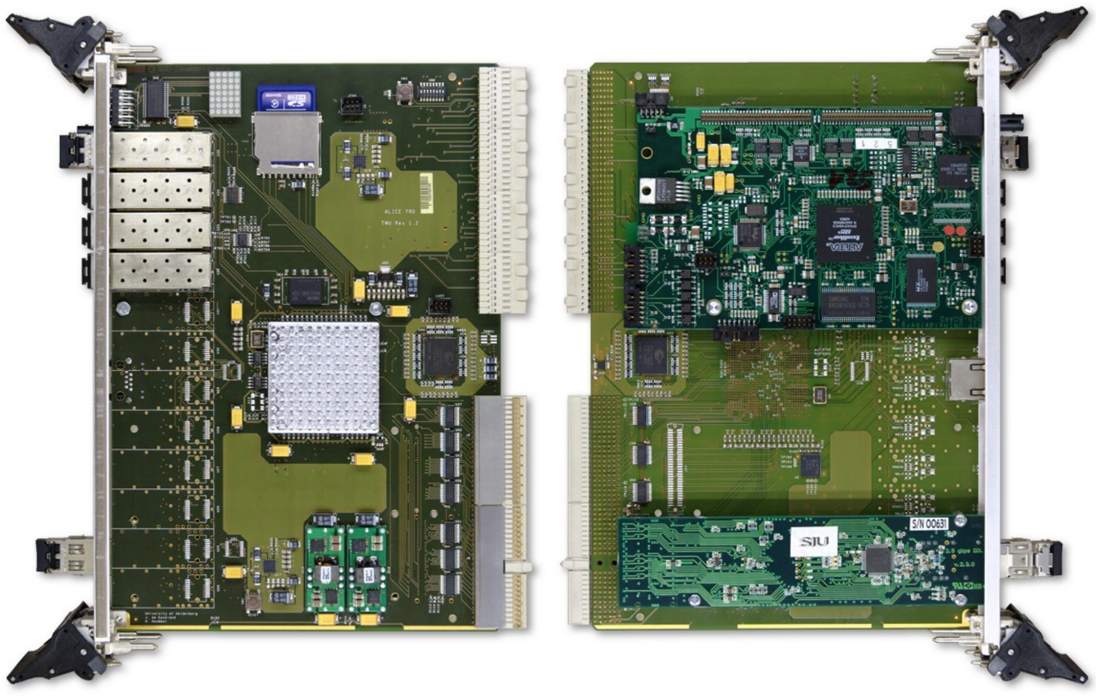


Figure 2.11: SMU processing node with DCS board (top) and SIU (bottom) on the back.

2.7.3 Tracking and Trigger

Reconstructed tracks of charged high-$p_\perp$ particles, which originate in the primary vertex and traverse the layers of a detector stack in a nearly straight line,[11] form the basis of the TRD's trigger contributions at the L1 stage. This section briefly summarizes the global tracking algorithm described in [41] and the triggers presented in [99] to give an overview of the different stages of trigger processing in the GTU, the trigger-related functional units and their location.

Online Track Matching and Reconstruction In order to increase the accuracy of PID and momentum reconstruction with respect to single tracklets, tracklets from individual detector layers are combined to tracks traversing the whole detector stack. The online matching algorithm utilizes that tracklets belonging to a high-$p_\perp$ track show only slight deviations in their coordinates when projected on a center plane. It then applies, see Figure 2.12, a 3-D sliding-window criterion: If the projected coordinates of tracklets from at least four layers are inside the matching window, a high-$p_\perp$ track is considered 'found'.

Each TMU matches tracklets originating at its associated detector stack to exploit spatial data locality, thus enabling parallel tracklet processing across the full TRD. The parameterized track segments from the FEE arrive at the GTU presorted in z-direction, which

[11]The radii of particle trajectories with high $p_\perp$ are large compared to the dimensions of the TRD. With $p_\perp = e \cdot r \cdot B$ follows, e. g., $r \approx 20\,\mathrm{m}$ for $p_\perp = 3\,\mathrm{GeV/c}$, $B = 0.50\,\mathrm{T}$, $e = 0.30\,\frac{\mathrm{GeV}}{\mathrm{c\,m\,T}}$

allows to start the track matching already while the tracklet transmission is still ongoing. Due to the specific detector and pad geometry, only tracklets with certain combinations of layer and pad row can belong together. The corresponding selection into three z-channels is done first and eliminates one dimension in the remaining matching process. For each z-channel, three track finders perform in parallel the window matching with respect to the projected y-position and projected vertex angle α by comparing the stream of tracklets in their layer against all other layers. Tracks that were found multiple times as a result of the parallel processing are subsequently removed by a merging unit.

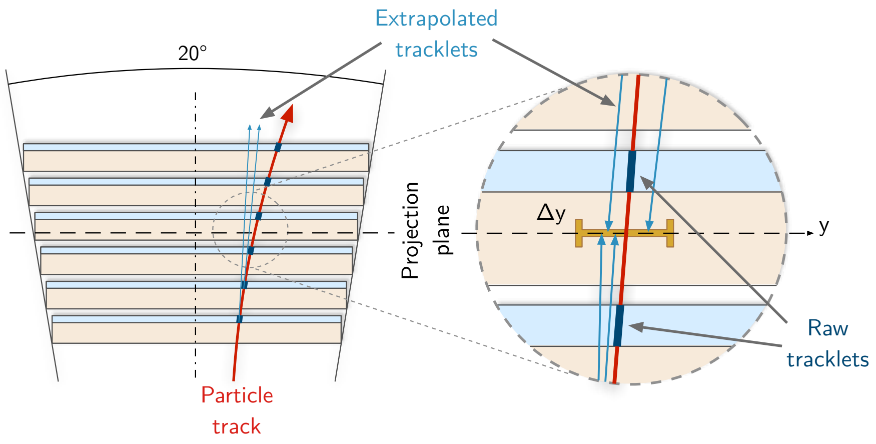


Figure 2.12: Schematic representation of the extrapolation of the tracklets to a virtual reference plane and window criterion used during track matching. Adapted from [41].

Reconstruction of the particle transverse momentum is illustrated in Figure 2.13. The GTU approximates the trajectory using a best-fit straight line. Several simplifications allow to limit the calculation of the line parameter a to few multiplications and table look-ups. The assessment whether a certain threshold $p_{\perp,\mathrm{thr}}$ was exceeded follows directly from a comparison of the product $a \cdot p_{\perp,\mathrm{thr}}$ with a constant c. Actually calculating $p_\perp$ from a is omitted for the trigger application, as it would require a division with high cost in terms of latency. To allow for more complex triggers, the GTU currently implements a comparison to two thresholds, $p_{\perp\mathrm{thr,a}}$ and $p_{\perp\mathrm{thr,b}}$ [99].

The results of these comparisons, the line parameter a, and the matching layers, together with the combined PID information, form the content of the fast tracking word, see Figure 4.11a, which is required for the calculation of triggers. A ‘tracking done’ marker transmitted by each TMU notifies the trigger entities on the SMU of the end of the tracking and enables measurements of the tracking duration. Each tracking word is complemented by an extended tracking word, which contains valuable information for the offline verification, e. g., indices of the associated tracklets and cut flags. These extended words are buffered within the tracking and transmitted only after the time-critical tracking data has been shipped.

Trigger Algorithms The tracking words from the five stacks are received on the SMU, where trigger algorithms in parallel analyze the tracks with respect to certain criteria.

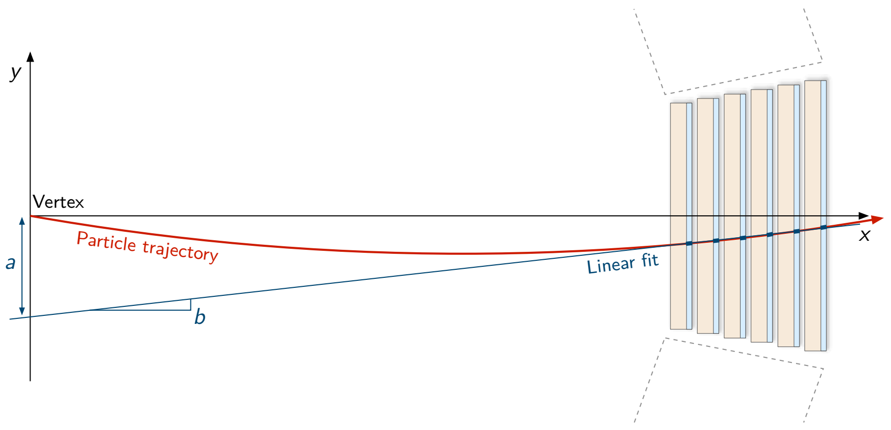


Figure 2.13: High-$p_\perp$ trajectories are approximated using a straight line fit yielding the line parameters a and b. Assuming that the track emits from the primary interaction vertex, its transverse momentum $p_\perp$ can be estimated directly from a. Adapted from [41].

Examples for these criteria are the number of tracks per stack, the number of tracks with $p_\perp \leq p_{\perp,\mathrm{thr}}$, conditions on the contributing layers and others.

Intermediate trigger results from the SMUs are transmitted to the top-level TGU, where they are combined to form a final decision, which is delivered to the CTP at a defined latency[12] with respect to the L0 trigger. Table 2.3 gives an overview of the L1 trigger contributions provided by the GTU in Run1. The current design can support up to 12 trigger algorithms on the supermodule level.

Algorithm	Year	Description	Condition
HCO	2007	Cosmics	$n_{\mathrm{tr}} \geq 1$
HJT	2011	Jets	$(n_{\mathrm{tr,stack}} \geq 3 \wedge p_\perp \geq 3\,\mathrm{GeV/c}) \vee$ $(n_{\mathrm{tr,stack}} \geq 250 \wedge p_\perp \geq 2\,\mathrm{GeV/c})$
HQU	2011	Quarkonia	$n_{\mathrm{tr}} \geq 2\,\mathrm{GeV/c} \wedge \mathrm{PID} > 164 \wedge c_{\mathrm{layer}}$
HSE	2012	Heavy-flavor electrons	$n_{\mathrm{tr}} \geq 3\,\mathrm{GeV/c} \wedge \mathrm{PID} > 144 \wedge c_{\mathrm{layer}}$
HEE	2013	Heavy-flavor electrons plus EMCal	as HSE, but with limit to sectors that overlap with EMCal (sectors 6–8)

Table 2.3: Overview of the L1 trigger contributions in Run1 delivered by the GTU. The layer condition c_{layer} requires at least five tracklets contributing to the track, one of them in layer 0. See [99] for details.

The input track segments, information about the reconstructed tracks, intermediate trigger calculation results, and the resulting trigger contributions are all required to perform a thorough offline evaluation of the TRD online tracking/trigger performance, and are input to a trigger component at the HLT stage. Although the processing related to trigger

[12] The latency of the GTU trigger contributions is 234 LHC cycles in Run1. The latency of the CTP-issued L1 trigger is 260 cycles with respect to the L0.

functions widely independent of the data path, this information has to enter into the event fragment, which is potentially assembled for readout by the SMU only later at the L2 stage. To store this meta data, which is available/valid at different points in time within a trigger sequence,[13] dedicated buffers are provided. These buffers (Section 4.5) are designed to interact with the multi-event capable segment controller (Section 3.2) to attribute the correct meta data set to a given event selected for readout. During the local event building (Section 4.6), the buffers are queried and the content of assembled event fragment facilitates a downstream analysis to ensure optimum performance of the TRD online trigger.

[13]Depending on the source (tracklets/tracking/trigger), meta data is available/valid somewhere in the timeframe between L0 and end of the event data transfer from detector to GTU following a L1.

3 Multi-Event Segment Controller

Modern particle physics experiments search for rare events with very small production cross section. This search is facilitated by accelerators capable of delivering the required high interaction rates and detector systems aiming to inspect, and possibly record, events at the maximum rates achievable. With $\mathcal{L}$ as instantaneous luminosity provided by the accelerator and σ_p as the production cross section, the rate ν_i at which a given process occurs is given by $\nu_i = \sigma_p \mathcal{L}$. The total number of interesting events n_i that can be observed then follows from the integration over the detector live time as

$$n_i = \int_0^T \nu_i = \sigma_p \int_0^T \mathcal{L}(t)\mathrm{d}t = \sigma_p \mathcal{L}_{\mathrm{int}}$$

with $\mathcal{L}_{\mathrm{int}}$ as integrated luminosity. Due to its statistical nature, a prediction of the exact moment of occurrence is not possible. When trying to maximize the number of interesting events observable, it is therefore not only important to employ an efficient trigger scheme for a given signature, but also to minimize the time in which the detector is incapable of recording a new event.

Dead time denotes a period immediately following the acquisition of one event during which the detector is insensitive to another event. Dead time is an intrinsic characteristic of all particle detectors. In cases where the dead time is smaller than the minimum time between events, e. g., minimum time between colliding bunches, the detector can be regarded as dead time free. Typical dead times of modern charged particle detectors range from few ns to several ms per event.[1] Depending on the type of detector, dead time stems from the principle used for particle detection and/or from the subsequent processing and readout of the acquired data.

To counteract dead time related to readout, detectors often employ queues with buffer capacity for several events to smoothen input rate fluctuations and de-randomize the load on the readout system. However, with concurrent handling of multiple events in multi-level trigger systems, the complexity of all elements in the detector's processing chain increases. Consequently, the potential for errors increases, and the overhead connected with a reset/restart of a system again significantly lessens the detector's live time. If the beneficial effect of multiple event buffers is not to be undone, special care has to be taken to arrive at a robust implementation, which ensures stable, uninterrupted detector operation.

This chapter introduces the dead time sources of the TRD readout chain and analyzes the beneficial effect buffers for multiple events have. It furthermore presents both concept and implementation of a robust control logic for a GTU segment, which administers trigger

[1]A short overview on typical detector dead times may be found in [26].

calculation, buffering of events and local event building in accordance with the received trigger sequences.

3.1 Multiple Buffers to Reduce Dead Time

LHC-era particle detectors implement multi-level trigger systems. As the real-time computations contributing to the trigger decision become more complex the higher the level, e. g., because more inputs are factored in, the trigger latency increases. Low trigger levels therefore typically exhibit low latency and coarse selection criteria, while higher levels show increased latency yet efficient rejection of high-rate background signals. The rate of events passing a given trigger stage depends on both the contributing individual trigger algorithms and the mix of algorithms constituting the trigger stage. If the level rates are fine-tuned to match the characteristics of the individual readout detectors and, e. g., slow detectors are only started when a previous trigger stage indicates an interesting event, multi-level trigger systems allow to increase the overall experiment's performance.

Relevant for the physics analysis, e. g., for the determination of cross sections, is the fraction of time the experiment was not able to take data with respect to the total runtime, during which a given data sample was acquired. Dead time is therefore often expressed as relative dead time

$$\tau = \frac{\sum d_i}{t_{\text{tot}}}\,,$$

where d_i is the dead time of an individual event and t_{tot} the total runtime. Because the overall detector efficiency is diminished by the dead time, the collection of a given amount of events requires a prolongation of the experiments runtime by $\tau/(1-\tau)$ compared to a dead time free system.

Collision events in a particle collider occur independently at a constant[2] average rate. The number of events input to the first processing stage per time unit follows a Poisson distribution, thus event interarrival times are distributed exponentially.[3] For subsequent stages, the minimal interarrival times depend on the processing time of the previous stage. Consider a simple processing stage capable of handling one event at a time. For processing an event, it requires a certain time t_p, which shall be constant. The stage is dead whenever an event n_{i+1}, which follows the event n_i, arrives at a time $t_{i+1} \leq t_i + t_p$. The relative dead time is given by

$$\tau = \frac{t_p n}{(\frac{1}{R} + t_p)n} = \frac{x}{1+x}\,, \qquad (3.1)$$

[2]In fact, as the intensity of the beams slowly decreases, the interaction rate decreases. This happens over the course of an hour-long fill. For the timescales considered here, the mean rate can be considered constant.

[3]The collider bunch structure, filling scheme and the resulting short intervals when collisions can occur, of course define a minimum interarrival time and influence the distribution. However, for all practical cases, the Poisson distribution is a suitable approximation.

where n is the total number of processed events, R the average input rate, R^{-1} the average time between events and $x = Rt_p$ a measure for the workload of the stage. When the average input rate R approaches the maximum processing rate of the stage t_p^{-1}, significant dead time is to be expected. When the average time between the arrival of event and the processing time are equal, $x = Rt_p = 1$, the stage exhibits a relative dead time of 50 %. The dead time can be reduced by introducing event buffers at the input to the stage to de-randomize fluctuations of the input rate. Given sufficiently large buffer capacity, the dead time then can be reduced to zero as long as $t_p < R$.

Multiple Event Buffers in Literature The effect on dead time and the fraction of recorded events that multiple buffers preceding a processing stage have, has extensively been studied in [27, 54, 122, 23, 25]. Particularly interesting is [62], where the author considers a scheme in which a first-level trigger inserts events at rate R into a second-level stage with processing time t_p. At its input, the second-level stage provides buffers for N events. The author derives a general formalism for the dead time fraction as function of the buffer depth and then quantifies the dead time for both randomly distributed and constant second-level processing time. Since they are used for comparison in Section 7.5.1, a summary of the results obtained in [62] is given in remainder of this section.

The scheme considered in [62] assumes that new events arrive at the system as long as at least one of the N event buffers is empty. The probability a_j that j new events arrive while another event is being processed follows from the distributions of arrival and processing time. They also define the rates r_m at which the input queue to the stage is observed to have a length m. When inspected immediately after an event has been removed for subsequent second-level processing, the length m of the queue always satisfies $0 \leq m \leq N - 1$. The system idle fraction is characterized by the rate r_0 at which zero events are observed in the input queue, multiplied by probability a_0 of no events arriving and divided by the average input event rate R. The total running time is then expressed as sum of idle time fraction and processing time fraction:

$$1 = \underbrace{\frac{a_0 r_0}{R}}_{\text{Idle fraction}} + \underbrace{\frac{x}{R}\sum_{m=0}^{N-1} r_m}_{\text{Processing fraction}} \tag{3.2}$$

with $x = Rt_p$. When corrected for dead time τ_N, the total processing rate and the input rate have to be equal:

$$R(1 - \tau_N) = \sum_{m=0}^{N-1} r_m \,. \tag{3.3}$$

Combining Equations 3.2 and 3.3 yields the relative dead time expressed in terms of the observed queue length rates r_m as

$$\tau_N = 1 - \frac{\sum_{m=0}^{N-1} r_m}{a_0 r_0 + x \sum_{m=0}^{N-1} r_m} \,. \tag{3.4}$$

Taking into account that an observed queue length of m required the queue to have had a value less than $m+1$ at the time the processing of the previous event started, allows to express the rates r_m as

$$r_m = \sum_{j=0}^{m+1} a_j r_{m-j+1} \,, \qquad m = 1, \dots, N-2 \,. \tag{3.5}$$

The rate r_0 at which an empty queue is observed additionally has to account for periods in which the queue is empty and no new event arrives, i. e., the idle periods. This gives

$$r_0 = a_0 r_1 + a_1 r_0 + \underbrace{a_0 r_0}_{\text{idle}} \,. \tag{3.6}$$

Assuming that the arrival of events follows a Poisson distribution, i. e., the events arrive independently of each other at a fixed average rate R, allows to calculate the probability a_j of j events arriving in the time interval t_p as

$$a_j = \frac{e^{-Rt_p}(Rt_p)^j}{j!} = \frac{e^{-x}(x)^j}{j!} \,. \tag{3.7}$$

Introduction of the sums S_m as

$$S_m = \frac{1}{a_0 r_0} \sum_{j=0}^{m-1} r_j \tag{3.8}$$

allows to express the relative dead time τ_N for the system with N buffers as

$$\tau_N = 1 - \frac{S_N}{1 + x S_N} \,. \tag{3.9}$$

Evaluating Equation 3.8 for a **constant** second-level processing time $t_{p,\text{const}}$, the author of [62] derives

$$S_m^{\text{rnd,const}} = e^{mx} \sum_{j=0} m - 1 \frac{(m-j)^j (-xe^{-x})^j}{j!} \,. \tag{3.10}$$

For the case of **exponentially** distributed processing times $t_{p,\text{rnd}}$, a convolution with the Poisson distribution yields

$$a_j = \frac{1}{1+x}\left(\frac{x}{1+x}\right)^j , \tag{3.11}$$

and consequently, by induction,

$$S_m^{\text{rnd,rnd}} = \frac{1-x^{m+1}}{1-x} . \tag{3.12}$$

With Equation 3.9, the author of [62] calculates the expression found widely throughout literature for the fractional dead time of a system with randomly distributed processing time

$$\tau_N^{\text{rnd,rnd}} = \frac{x^{N+1}(1-x)}{1-x^{N+2}} . \tag{3.13}$$

Figure 3.1 illustrates the results for the relative dead time as a function of the workload parameter $x = Rt_p$ for different numbers of event buffers N at the input to the second-level stage. The dead time of a system with random arrival of events and constant second-level processing time, obtained via Equations 3.10 and 3.9, is shown as solid lines. For comparison, a system with random second-level processing time, see Equation 3.13, is shown as dashed lines. Different colors represent different numbers of buffers.

The beneficial effect multiple event buffers have due to de-randomization of the load on the processing stage is clearly visible. The largest benefits are already gained by a small amount of buffers. For example at $x = 1$ in case of constant second-level processing, $N = 4$ reduces the dead time from 50 % for $N = 0$ (dark blue) to a value of 10 % (orange, solid). When aiming to operate the system very close to the maximum processing rate, a large number of buffers may be beneficial.

The curves for $N = 0$ and $N = 128$ illustrate extreme examples. Without buffers, the system exhibits dead time whenever an event is being processed, which increases to a level of 50 % already at $x = 1$. This corresponds to the single stage example of Equation 3.1. For all systems with finite number of event buffers, short-time fluctuations of the input event rate may flood all buffers for $x < 1$, which results in dead time. However, the probability of all buffers being full diminishes for lower system workload and increased number of buffers. With $N = 128$, dead time practically vanishes completely right up to $x = 1$. For $x > 1$, the system invariably produces dead time, since the input event rate exceeds the processing capabilities and eventually the intake of new events has to be paused for processing to catch up.

The lower plot of Figure 3.1 summarizes the differences $\Delta = d_N^{\text{rnd,rnd}} - d_N^{\text{rnd,const}}$ due to random and constant second-level processing and highlights the importance of the correct processing time distributions. The differences for $N = 0$ and $N = \infty$ are zero, which is intuitively clear as the averages of both constant and random processing time are equal. However, for systems with moderate amount of event buffers, discrepancies are visible.

At maximum they amount to about 7 % for $N = 2$ around $x = 0.95$. The decrease in difference for larger N is, again, intuitively clear: As more event buffers cover a longer time span, the average processing time within that time span approaches, following the law of large numbers, the mean t_p.

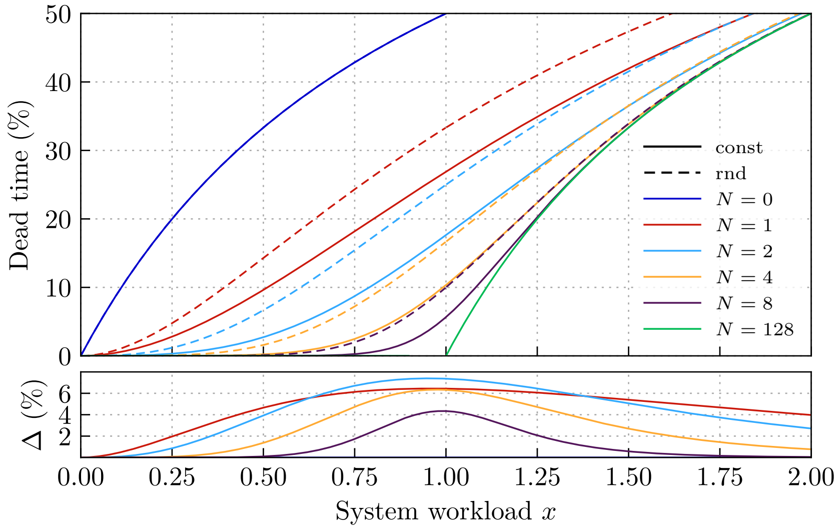


Figure 3.1: Dead time of a processing stage with buffers for N events as a function of the workload x (upper plot). Constant processing times are shown as solid lines, random processing time distribution as dashed lines. Already few buffers significantly reduce the dead time related to the processing stage for values of $x < 1$. The lower plot summarizes the difference resulting from random and constant processing time distributions.

3.2 Segment Controller Overview

The 18 GTU segments, each consisting of $5 + 1$ nodes to handle data and triggering for the associated supermodule, operate independently. They are synchronized only by the trigger sequences received from the CTP. Each segment comprises a unit in the SMU FPGA that decodes triggers received via the TTC system. Based on these triggers, suitable validated signals to steer all activity in the segment, including the buffering of multiple events, are generated by the segment controller.

Figure 3.2 provides a full overview of the flow of event data and trigger information within the GTU hierarchy and of the major firmware building blocks. Details of the SMU design are highlighted in more detail in Figure 3.3. The architecture and entities related to the handling of event data, building of event fragments and readout are presented in Chapter 4. A short overview of the requirements and considerations leading to the chosen control architecture implemented on the SMU, which is presented in depth in Sections 3.4 and 3.5, is given in the following.

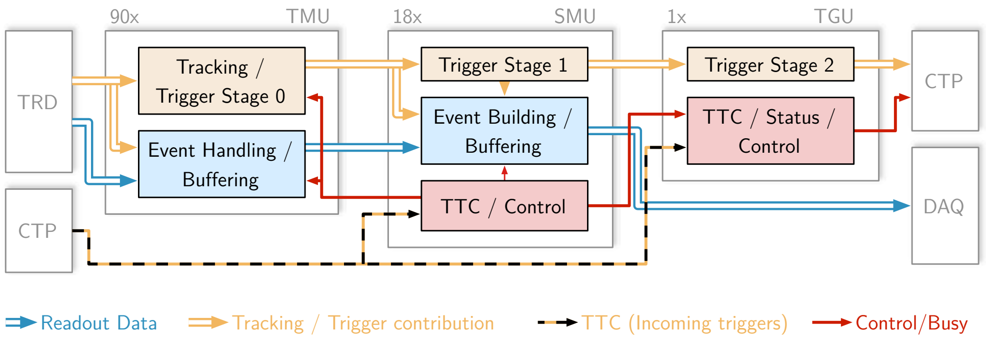


Figure 3.2: Major design blocks of the three types of FPGA designs of the GTU hierarchy. Information related to tracking and trigger calculation enters the data path at various locations and must be buffered for potential readout at a later moment.

Trigger Reception Trigger sequences –and the experiment-wide LHC clock– are distributed to the subdetectors in a partition to steer the acquisition and readout of data in a synchronized manner. Endpoints for the trigger reception on the GTU are the DCS boards, SMU and TGU nodes must therefore implement the reception of triggers.

Trigger Verification From the trigger information, the SMU extracts the event ID and assigns it to the event data acquired by a TRD supermodule. Based on this event ID, the local event fragments are combined to events containing data from all subdetectors on the global event building stage in the DAQ. Undetected errors related to the trigger generation or transmission, e. g., a missing L1 trigger, might cause the assignment of a wrong event ID to data in the fragment and mixing of fragments from different events on the GDC level. The segment control unit therefore must validate the integrity of the trigger sequences.

Internal Control The validated trigger signals of the various levels must be processed and converted to control signals suitable to steer the operation of a GTU segment. The segment control must generate and distribute appropriate control signals, timeframes and references to manage buffering of the event data as well as auxiliary data, reliable trigger calculation and readout in the segment. To allow for an efficient multi-event capable implementation, where the L2-associated readout or reject of events is fully decoupled from the acquisition phase, this requires buffering of partial trigger sequences and their delayed processing.

Busy Generation Each GTU segment monitors the arrival and departure of events in the event buffer and therefore has access to all information required to efficiently generate the supermodule-level busy contributions. These contributions are then combined on level of the TGU to a TRD-global busy. With regard to an optimal system performance, the amount of dead time resulting from factors other than the fundamental characteristics of the detector readout presented in Section 3.3 must be minimized.

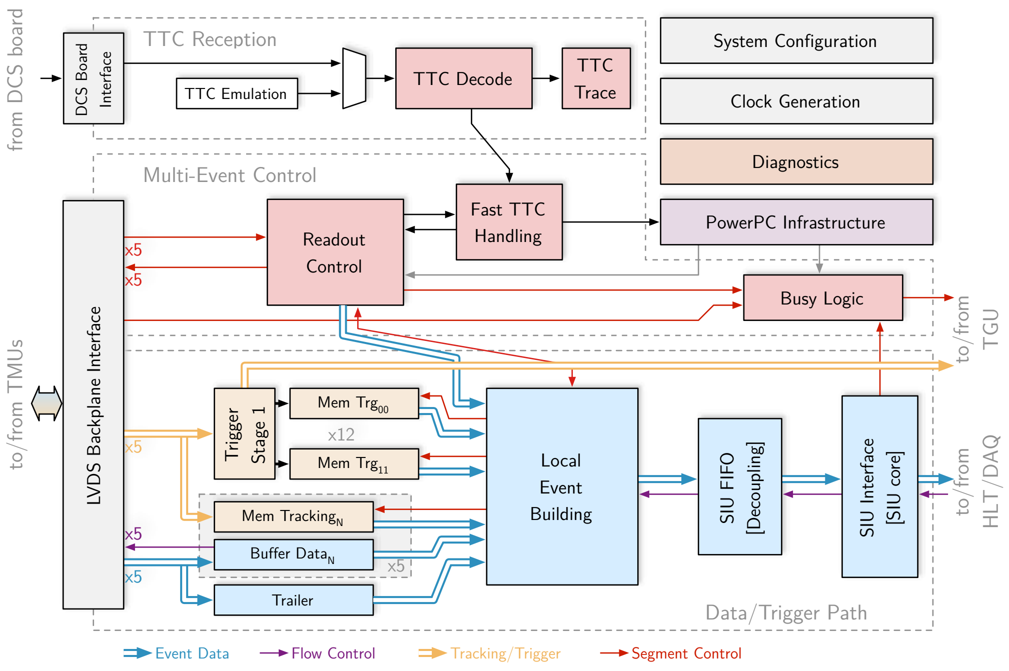


Figure 3.3: Overview of the SMU firmware design and flow of information. The SMU calculates the supermodule-level trigger contributions and administers event reception, buffering, local event building and readout to the DAQ for one GTU segment.

Stable Operation and Post-Mortem Analysis Stable, reliable operation of each individual subsystem is a key requirement when aiming for permanent operation of a complex detector system and high data taking efficiency. For example, lock-ups and the required restarts add to the time the system is not ready for data taking, i. e., dead time. However, the sources for errors are numerous and often not obvious, which makes it difficult to design accordingly. The GTU firmware designs must therefore guarantee correct and stable system operation under normal conditions. Abnormal conditions must be detected and the system should provide the expert with all relevant information to rapidly trace them, allow for a quick recovery, and possibly implement a workaround.

Configurability and Upgradability In a long-running experiment like ALICE, which operates for more than a decade and continuously optimizes its detector for better physics performance, operational parameters and the detector environment itself vary over the runtime of the experiment. For example, the requirements during the commissioning phases differ significantly from those in the production phase. Furthermore, GTU setups are not only installed in the cavern, but are also used, in reduced form, for development, supermodule production and pre-installation testing. These test setups do not feature the full experimental infrastructure with respect to the trigger and control environment. They require extended functionality, e. g., trigger emulation in the case of supermodule production, to operate. All this must be reflected by the firmware design, leading to the support of a variety of different configuration options, e. g. regarding trigger timing and processing, and well-structured designs.

3.3 TRD Operation and Dead Time Sources

The operation of the TRD is divided into different phases, which stem from its dual-use as trigger and readout detector. The transition between the different phases is steered via the trigger sequences provided by the CTP. The dead time exhibited by the TRD is a mix of intrinsic dead time due to detector type (chamber drift time plus digitization) and dead time originating from the processing electronics.

The TRD readout scheme foresees event data to be held on the detector FEE until the L1 decision time, after which it is shipped to the GTU or discarded. The buffering of events and subsequent assembly of event fragments is implemented by the GTU, which is an obvious choice given the availability of many high-speed interconnects to the front-end required by the trigger application and the already high complexity of the FEE ASICs. Figure 3.4 shows a combined timeline for a trigger sequence that leads to the readout of an event and the actions of TRD FEE and GTU. Details of the timeline and aspects related to dead time are discussed in the following paragraphs.

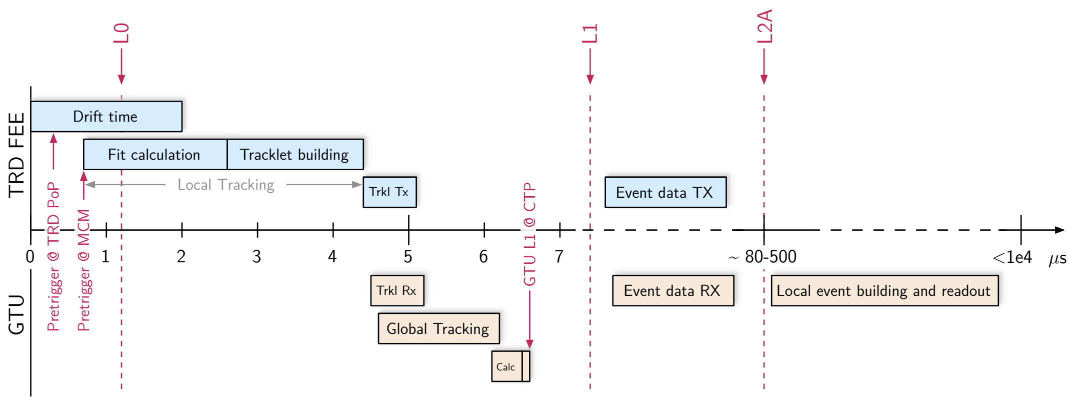
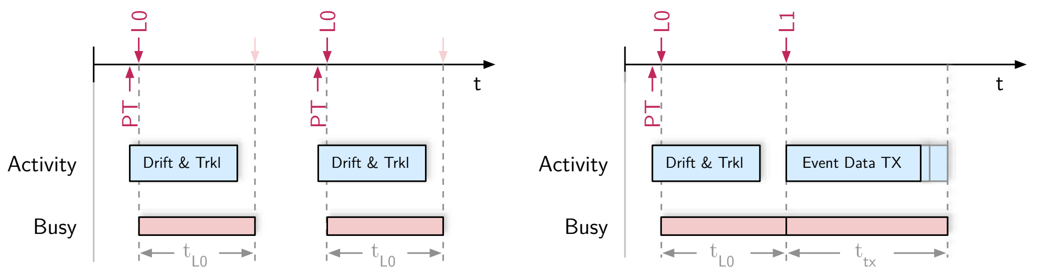


Figure 3.4: Trigger calculation and readout activities on the detector FEE (blue) and in the GTU (yellow) in the course of one L2A trigger sequence leading to the readout of an event (timing is approximate).

Pretrigger/Level-Minus The PT system provides a fast wake-up signal for the detector and implements a protocol conversion to trigger the front-end. The wake-up, which arrives at the detector chambers less than 500 ns after the interaction and thus much earlier than the L0 trigger issued by the CTP at 1.2 µs [121, 84], starts the local tracking on the front-end, see Figure 3.4. For correct operation of the detector, a L0-stage contribution from the PT is a necessary prerequisite to issue a L0 trigger to any readout partition containing the TRD. The PT is succeeded by the LM system, which is more tightly integrated into the CTP, for Run2.

L0-L1 Interval and L1 Accept With an L0 arriving from the CTP, the FEE continues the sequence already started by the pretrigger. The operation of the front-end is not pipelined and gathered data is held in the memories of the filter unit [104, 6]. The system is therefore busy until the data is either discarded or transferred off the detector.



(a) Events rejected at L1 **(b)** Event accepted at L1

Figure 3.5: Dead time generated by the GTU that is associated with front-end operation due to L0/L1 triggers (not to scale). Once a L0 trigger has been received, the detector is busy while waiting for the L1 decision for a fixed amount of time t_{L0}. In case the event is accepted at the L1 stage, the detector is ready again once all data has been moved off.

The GTU interprets the L0 as the start of a new trigger sequence and raises its busy signal to protect the non-pipelined front-end, which possesses no busy connection to the CTP, from any new L0 triggers. In any case, the busy is active throughout the L0-L1 decision window t_{L0}, see Figure 3.5a, and constitutes the dead time contribution τ_{L0}. The L1 trigger, which starts the acquisition with the TPC, is also the signal for the TRD to move data off the front-end to the event buffers in the GTU. Once the last chamber has finalized the transmission after t_{tx}, the front-end is ready for a new event and the GTU releases its corresponding busy contribution, see Figure 3.5b.

With no back-pressure signal going to the detector front-end, the GTU has to guarantee that it is able to admit a new event into its buffers whenever the acquisition with the TRD was started. With multiple event buffers, it must be decided at the end of the FEE-GTU transfer whether another event can be accepted, i. e., another event buffer is free, or not. In the latter case any new trigger sequence has to be prohibited and the GTU raises the busy contribution related to readout for the time t_{ro}, see Figure 3.6.

Settings for the L1 latency are 260 LHC clock cycles (6.49 µs) for Run1 and 280 cycles (6.99 µs) for Run2. The duration of t_{tx} is variable. It primarily depends on the event size and is governed by the half-chamber exhibiting the largest data volume (see Figure 7.1a). Values for t_{tx} observed in Pb-Pb collisions at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$ range from 20 µs for minimum bias to approximately 90–100 µs for central events (see Figure 7.13).

L2 Accept and Reject The L2 trigger decision determines whether event data buffered in the GTU is read out or not. Assuming that the GTU could only handle and buffer the current event, the system is unable to accept a new event before the readout of the data via the DDL is finished. This would result in a dead time of several ms per event in Pb-Pb.

In contrast, Figure 3.6 illustrates the dead time generated for a TRD supermodule when running standalone with multiple buffers and a mix of interlaced L2A and L2R trigger sequences. Shown in blue and yellow are relevant activities of the supermodule and GTU, respectively. Given that events marked for readout are queued in the buffers, the segment is constantly transmitting event fragments via the DDL. Following the arrival of a L2A trigger, a GTU segment builds the event fragment, ships it and frees the corresponding buffer slot. For a L2R, the corresponding event data is discarded and the buffer is freed.

Figure 3.6 highlights the situation in which all buffers are occupied at the end of the FEE-GTU transfer. In this case the GTU must raise the readout-related busy. It is kept active for the time t_{tx} until a buffer is freed. The corresponding dead time contribution τ_{ro} is directly affected by the number of event buffers and the GTU's second stage (L2A/L2R) processing times.

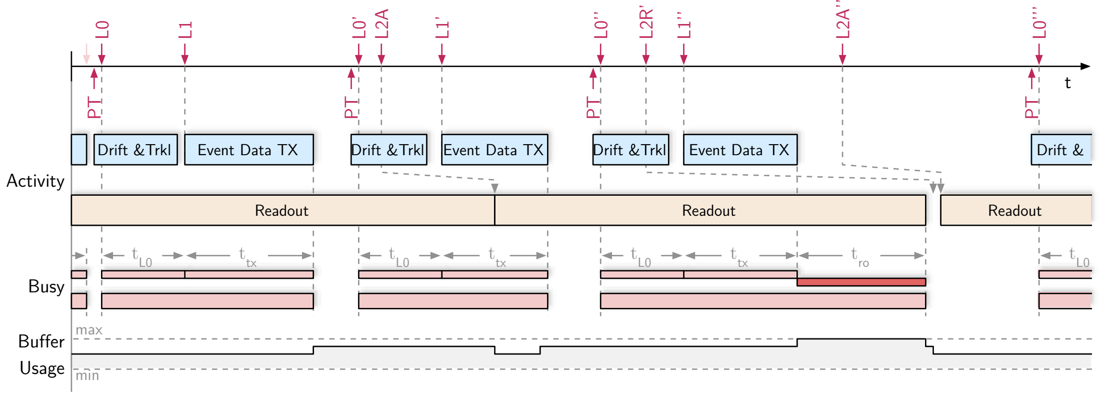


Figure 3.6: Dead time contributions generated by a GTU segment for a supermodule in standalone operation with enabled buffering of multiple events (timing not to scale). The segment generates readout-related dead time t_{ro} only when all available event buffers are occupied, delaying the potential start of new TRD acquisition.

3.4 Trigger Reception

When operating a complex detector system in a synchronous fashion, problems like the distribution of signals with high fan-out over long distances at deterministic latency arise. The ALICE CTP uses the TTC system[4] to distribute trigger information and one of the beam clocks as experiment-wide reference clock to all subdetectors.

In case of the GTU, the optical fiber originating at the TRD TTC partition is connected to a 1:32 optical tree coupler, from where $18 + 1$ fibers establish a unidirectional connection to the inputs of the DCS boards[5] on the GTU nodes. The arrival of an element of a trigger

[4]The TTC system [36] can broadcast signals to multiple endpoints over a passive optical network and has the capability to apply propagation delays or phase shifts.

[5]18 TTC fibers are connected to the SMU, one fiber connects to the TGU. Additional fibers connect the TRD front-end via the PT/LM system.

sequence serves as a reference point to synchronize actions of the detector front-end, the independently operating segments and the TGU. Consequently, custom-developed trigger receiver units are instantiated on both TGU and SMUs.

3.4.1 TTC Communication in ALICE

The TTC system provides two channels, A and B, which are time-division multiplexed and transmitted as a bi-phase mark encoded 80 Mbit/s bitstream. Channel A is intended for the transmission of time-critical 1-bit trigger decisions associated to a certain bunch crossing with deterministic latency. Channel B transmits arbitrary frame-based data to all or selected destinations on the passive optical TTC network. Both channels and the reference clock are recovered on the detector side by dedicated TTCrx ASICs [38].

Table 2.1 lists the different hardware trigger levels, the channels used for distribution and the latency, i. e., the time the CTP starts to transmit the trigger with respect to the interaction. ALICE employs an evolved scheme for channel A, where it is used exclusively to transmit the L0 and L1 trigger signals in encoded form.[6] The complete trigger information sent for L1 includes the encoded signal on channel A and a L1 trigger message transmitted on channel B. The arrival of a L1 trigger (L1A) is expected at a fixed latency with respect to the L0. No encoded channel A signal at that given latency marks a reject of the event at the L1 stage (L1R). L2 triggers are transmitted in form of messages on channel B. Contrary to the L1, two distinct messages explicitly signal an accept (L2A) or reject (L2R) at the L2 stage.

Channel B of the TTC system defines two types of frame formats: Broadcast Command (BRC) and Individually Addressed Command (IAC) frames, see Figure 3.7. A BRC frame carries 8 bit payload and usually targets all entities receiving TTC information, including TTCrx chips and custom decoders. IAC frames provide an increased payload and an addressing feature to target only specific TTC endpoints. Channel B sending equipment allows to inhibit or prioritize the transmission of certain frames [49].

The most prominent use of BRC frames in ALICE is the bunch counter reset command (RST), which is sent to synchronize local event ID counters[7] on the detector electronics. For a comparison of local and global event ID to be meaningful, a fixed offset between them is required, which in turn requires the reset frame to arrive at the FEE once per LHC orbit at an exactly defined bunch crossing. The RST is therefore periodically transmitted towards the end of the LHC abort gap with highest priority. Prior to its transmission, any new traffic on channel B is inhibited for at least 54 clock cycles, i. e., more than the time required to transmit an IAC frame. This ensures immediate transmission, and thus fixed latency, with respect to the LHC orbit.

The trigger messages for L1, L2A and L2R contain information required to build the header of an event fragment. They are transmitted as IAC frames on channel B. Differing from

[6] L0 is encoded as "10" in two successive LHC clock cycles, "11" signals the arrival of a L1. See [70].

[7] At this stage, the event is identified by the number of the colliding bunch within the LHC orbit and the number of the LHC orbit, i. e., bunch counter and orbit counter.

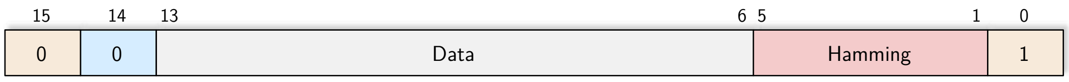
Message	Type	Frames		Payload	Tx duration	Priority
RST	BRC	1		8 bit	0.40 µs	high
L1M	IAC	5	(Run1)	60 bit	5.24 µs	low
		9	(Run2)	108 bit	9.43 µs	low
L2AM	IAC	8	(Run1)	96 bit	8.38 µs	lowest
		13	(Run2)	156 bit	13.62 µs	lowest
L2RM	IAC	1		12 bit	1.05 µs	lowest

Table 3.1: TTC channel B messages relevant for the TRD. The number given as transmission duration is the minimum time required to transmit all frames in case of no congestion on the B-channel.

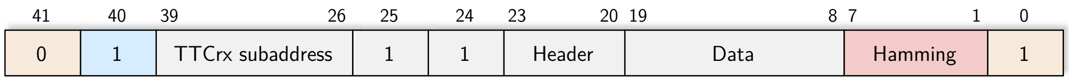
the specification in [38], the payload field in ALICE is interpreted as subaddress[8] (14 bit), header (4 bit) and data (12 bit) [67]. For a L1, the sending of the corresponding L1 trigger message is initiated by the CTP simultaneously to the transmission of the L1 trigger signal at a fixed latency of approximately 7 µs with respect to the L0. The transmission of L2 trigger messages is initiated by the CTP about 90 µs after the corresponding L0 trigger. However, the reception of all trigger messages is non-deterministic with respect to latency. The transmission of a L1 message may be interrupted by the higher-prioritized transmission of a RST command. Furthermore, L1 messages are prioritized over L2 messages and may interrupt ongoing transmissions on a frame basis. Consequently, the arrival at the front-end of the final IAC frame constituting a given message may be delayed. Table 3.1 summarizes the relevant messages transmitted on channel B, their length in IAC frames as well as the minimum transmission duration and priority.

Estimates for the maximum latency of trigger messages are of interest regarding the detection of errors. They are a criterion to detect illegal conditions related to the message transmission, see Section 3.5.4. The L2A message exhibits the maximum latency, as it contains the largest number of IAC frames and is transmitted with lowest priority. Assuming TRD standalone operation with high interaction rate and N event buffers, i. e., potentially interlaced trigger sequences, the prioritization of $N-1$ L1 messages over a

[8]Note that the addressing feature is not used. Also IAC messages are received by all TTC endpoints.



(a) BRC frame, 16 bit



(b) IAC trigger frame, 42 bit

Figure 3.7: TTC frame formats as used in ALICE. The payload of IAC messages is interpreted differently in ALICE than specified in [38].

L2A message represents a worst case scenario. With $N = 4$ buffers, the minimal message transmission duration and trigger latency given in Run2, and assuming the L1 message transmission starts just before the scheduled start of the L2A transmission, a worst case estimate for the L2A message latency with respect to L1 amounts to

$$l_{\mathrm{L2AM,\,max}} = t_{\mathrm{L2}} - t_{\mathrm{L1}} + 3\,t_{\mathrm{L1M}} + t_{\mathrm{L2AM}} \approx 125\,\mu\mathrm{s}\,.$$

3.4.2 Trigger Sequences

In a single-event operation mode, i. e., the GTU raises its busy signal with the reception of a L0 and releases it with a L1R or after a buffer is freed, the trigger events occur in chronological order in three distinct patterns:

- L0 (no trigger at L1 decision interval and therefore implicit reject, 'L0 sequence'),
- L0-L1-L2R (reject at the L2 stage, 'L2R sequence'), or
- L0-L1-L2A (event accepted for readout, 'L2A sequence').

The situation changes with the buffering of multiple events, where the TRD can accept a new event as soon as the event data is stored in the TMU event buffers. The form of the trigger sequences issued by the CTP now depends on the dead time behavior of all detectors in the readout partition. In case the TPC is a member of the partition, the form of the sequences does not differ from the sequences in single-event mode. After the TPC acquisition has been started,[9] the start of a new trigger sequence is inhibited for about 250–300 µs due to restrictions related to the opening of the gating grid. As the L2 message concluding a sequence typically arrives about 100 µs after the interaction, the sequences are not interlaced when running conjointly with the current TPC. However, when the TRD operates in standalone mode or with detectors exhibiting less dead time and this implicit inhibit is not given, a new sequence may be started before the corresponding L2 trigger has arrived. This may lead to interlaced trigger sequences, e. g., of the form $\mathrm{L0}_0$- $\mathrm{L1}_0$- $\mathrm{L0}_1$- $\mathrm{L0}_2$- $\mathrm{L1}_2$- $\mathrm{L2R}_0$- $\mathrm{L2A}_2$ (the index indicates which trigger sequence a trigger event belongs to).

Both TTC receiver and segment controller presented here are designed to handle the more complex, interlaced trigger sequences. This includes non-interlaced sequences as special case, does not impose artificial restrictions on the control logic or performance, and leads to a more robust design.

Software-Initiated Triggers In addition to the physics-related trigger, the CTP may issue software triggers, which are employed to control and synchronize the data flow in the detector and the DAQ. Prominent examples are the Start-of-Run (SoR) and End-of-Run (EoR) sequences [34], which trigger the sending of Start-of-Data and End-of-Data event fragments to unambiguously mark the first and last event in a run, as well as the

[9]Depending on the configuration, the acquisition starts with a L0 or L1, the latter being the default mode.

newly introduced Pause-and-Configure (PAC) and Synchronize (SYNC) triggers. These software-initiated sequences appear as regular L2A sequences, which are identified by a dedicated software class flag in the L1/L2A messages. The readout control bits of the L1 identify the type of software trigger. As regular L2A sequences, they are transparently handled by the hardware.

3.4.3 Trigger Receiver Implementation

With the TTCrx chips on the DCS board as endpoints in the TTC network, a forwarding of trigger and clock from to the FPGAs of the SMU nodes is required. Due to large number of I/O resources occupied by the high-bandwidth LVDS point-to-point backplane interfaces of the SMU, the number of available connections on the PCB is limited. Only two differential pairs are available for the forwarding, therefore a serial protocol is required.

The decision was taken to omit the decoded trigger information presented on the 8-bit parallel interface of the TTCrx. Instead, the serial outputs for channels A and B are utilized and decoding of the TTC protocol is implemented directly in the FPGAs of SMUs and TGU.

Trigger Forwarding from DCS board to TTC Receiver The reception, time-division multiplexing and forwarding of the two channels of the TTC system from DCS board to SMU is depicted in Figure 3.8.

The 40.078 MHz LHC clock is reconstructed by the TTCrx, which also applies corrections related to delay and phase. A 40 MHz quartz on the DCS board provides an alternative backup clock. Switching between LHC and backup clock is handled automatically by the DCS board if it detects the unavailability of the reconstructed clock. The clock is transmitted via one differential pair and feeds the GTU segments' control-related clocking infrastructure.

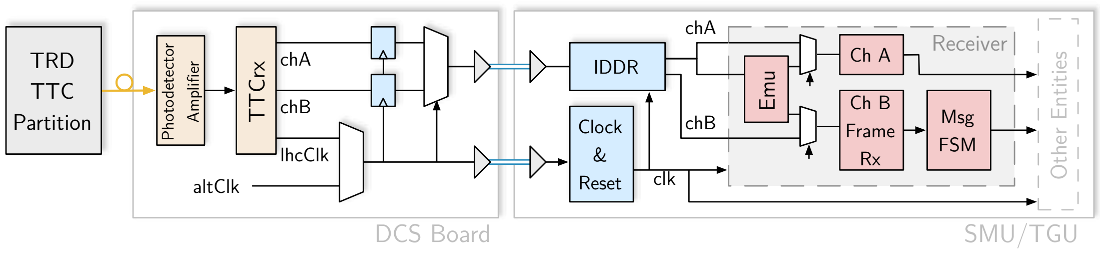


Figure 3.8: Reception and decoding of TTC trigger signals and LHC clock in a GTU segment (or TGU). The serial outputs of the TTCrx, channel A and B, are time-division multiplexed on the DCS board and forwarded to SMU/TGU alongside the LHC clock. The TTC receiver decodes both channels and provides triggers and trigger messages on its output.

The clock is used to time-division multiplex the two TTC channels, which occupies the second differential pair. In the absence of dedicated primitives in the DCS board FPGA, it is realized by replicating DDR functionality in the FPGA fabric. Suitable location constraints place the logic in the vicinity of the LVDS output cells and ensure correct signal timing. On the reception side, the Virtex-4's IDDR primitives are utilized to facilitate the demultiplexing and provide the channels A and B to the decoder logic in serial form.

The chosen solution eliminates the need for a custom protocol transmitting the decoded trigger data between DCS board and SMU/TGU and keeps the resource requirements on the rather resource-limited DCS board FPGA to a minimum. Decoding the TTC channel data directly in the GTU nodes has lowest demands in terms of I/O resources, two differential pairs suffice to transmit both clock and trigger. The decoding is customizable, which allows to detect/monitor TTC transmission errors as well as errors related to the structure of the trigger sequences.

Trigger Sequence Decoding The receiver module implemented for the GTU decodes information transmitted on channels A and B of the TTC system.[10] It supports all relevant characteristics of trigger transmission in ALICE, including the prioritization of L1 over L2 trigger messages, reception of multiple, interlaced sequences, and monitors the decoded trigger information for consistency and errors.

The GTU omits the reception of the L0 signal via a dedicated cable, as latency of the L0 is not critical. Instead, it receives both L0 and L1 triggers in encoded form on channel A [70]. The requirement for the GTU is that the latency of the L0 of about 1.2 µs is less than the arrival of the first tracklets from the FEE, which do not arrive less than 4 µs after the interaction. The optical transmission of the L0 plus decoding adds latency in the order of 100 ns. This leaves sufficient margin to safely distribute all L0-derived control signals within the segment and perform accounting or clean-up operations with the start of each trigger sequence. The receiver module signals the reception of a non-validated L0 or L1 trigger by activating the corresponding output, `trg_l0` or `trg_l1`, for exactly one LHC clock cycle.

For channel B, a frame decoder FSM monitors the bitstream for incoming BRC or IAC frames. The synchronization bits are stripped off and the frame payload data is placed in a shift register. Depending on the frame type, a number of dedicated outputs are presented to the fabric: The reset signals for the local bunch and orbit counter, `reset_bc` and `reset_oc`, are directly derived from the RST frames. L1 and L2 trigger messages, which consist of up to 13 IAC frames, see Table 3.1, are reassembled and monitored for completeness. Strobe signals (`l1m_strobe`, `l2am_strobe` and `l2rm_strobe`) indicate the successful reception of a message with the last IAC frame and validity of the message output. In case of L2A and L2R messages, these strobe signals are presented as trigger signals `trg_l2a` and `trg_l2r` to the control unit.

[10]The module for the channel B message handling is inspired by the single-event trigger receiver module presented in [14, 13].

Detection of Transmission Errors Decoding of the TTC channel data on the SMUs/TGU renders possible monitoring for transmission errors. The decoder signals if an unknown encoded signal, i. e., three successive high cycles, is encountered on channel A. For channel B frames, the included Hamming information allows to correct 1-bit and detect 2-bit transmission errors.[11] The channel B prioritization rules and resulting expected transmission orders are utilized to evaluate arriving frames within the context defined by the previously received frames and assess, to the extend possible, the completeness of individual messages. The occurrences of 1-bit or 2-bit errors, as well as errors related to messages, are signaled to the higher-level control unit for further processing.

Each receiver unit furthermore implements trace memories for all received trigger events, which provide sufficient information to analyze the events leading up to a potential error, see Section 6.3.2.

Internal Sequence Emulation Partial GTU setups are used to facilitate high-rate readout and verify correct supermodule operation during production and pre-installation testing. For example, a trigger on stiff tracks by cosmic particles allows to gather data of relative chamber alignment and ensure compliance with the design parameters already in the supermodule assembly phase [24]. These setups comprise a complete segment (five TMUs, one SMU) and a TGU or equivalent interface to the external trigger logic.[12] Instead of a complete CTP installation, only a minimal, custom trigger system [21] is available at the main site of supermodule production. It initiates the supermodule readout and supplies the GTU with a single pulse sequence start signal.

The TTC receiver features an internal sequence emulator to serve as alternative trigger source. If active, the emulator channel A and B outputs are, in demultiplexed, serialized form, fed into to the standard TTC decoding logic. Pulses received by the emulator on the regular TTC channel A from the DCS board now serve as emulation start signal[13] to produce a trigger sequence. The emulator supports conditional sequence aborts at the L1 and L2 stages, allowing, e. g., to employ the GTU segment-level cosmic trigger contribution. Type, timing and content of the trigger sequences are configurable via the PPC console application, with defaults set to CTP-like operation. With help of the emulator, the GTU can operate in this reduced environment under conditions as close as possible to regular operation without firmware modifications of the core design used in production data taking.

[11] BRC and IAC frames feature (13, 8) and (39, 32) Hamming codes, respectively.

[12] The combining functionality of the TGU is not necessarily required when operating with one segment only, often an interface board with logic level conversion suffices for the forwarding of busy and trigger to the external trigger logic.

[13] Supported as start signal are also pulses input via the auxiliary connector on the LVDS backplane or generated via the PPC.

3.5 Multi-Event Control Unit

The multi-event control unit processes the decoded triggers and derives control signals and timeframes. These are distributed within the segment to steer and synchronize the activities in a segment. The maximum number of trigger sequences that must be handled simultaneously thereby corresponds to the number of event buffer slots supported.

A naive approach to move from handling a single event at a time to a system capable of handling several events would be to instantiate an FSM able to handle a single trigger sequence multiple times. However, the correct arbitration of arriving trigger events to the FSM instances and of the generated control signals to the corresponding processing units seems difficult, especially with respect to the detection of trigger errors. A different approach, where all triggers are handled by a single, large control FSM also seems impractical. Even without the correct handling of the numerous possibilities for trigger-related errors, such an implementation would produce complex logic with a large number of states. It would be difficult to develop, verify, extend and maintain.

Fast arrival of the constituents of the decoded trigger sequences calls for real-time processing with dedicated hardware, which in turn should be kept simple to minimize the potential for errors. An analysis of the different stages of the TRD detector operation, see Figure 3.4, and the fact that trigger events of a given type arrive in chronological order, instead suggests a partitioning of the processing in an acquisition part and a readout part. An overview of the developed control unit, which is an evolution of the works presented in [76, 64] is shown in Figure 3.9. Details of its implementation are discussed in the following.

3.5.1 Acquisition Phase

The event acquisition phase covers the timeframe from the start of a trigger sequence with a L0 trigger to either the L1 decision time, in case of a reject at the L1 stage, or the conclusion of the FEE-GTU event data transfer in case of a L1 accept. During all that time, the front-end is not able to accept another event and, consequently, no new trigger sequence must be started.

In terms of traffic on channel A, this means that a L0 trigger is followed, in error free operation, either by the its corresponding L1 trigger or by the L0 trigger of the next trigger sequence with a minimum trigger spacing equivalent to at least the L1 decision interval. In terms of segment operation, the TMUs must accept input data from the front-end with the L0 and start the tracking entities. Buffering of event data also starts with L0, and requires a flush of the active buffer slot in case of a L1R or due finalization and accounting of the event in case of a L1A.

Acquisition Handler The acquisition handler FSM, see Figure 3.10, derives suitable control signals in the acquisition phase. It receives as inputs the two trigger signals, L0 and L1, and the strobe signal for the L1 message. The reception of a L0 initiates the

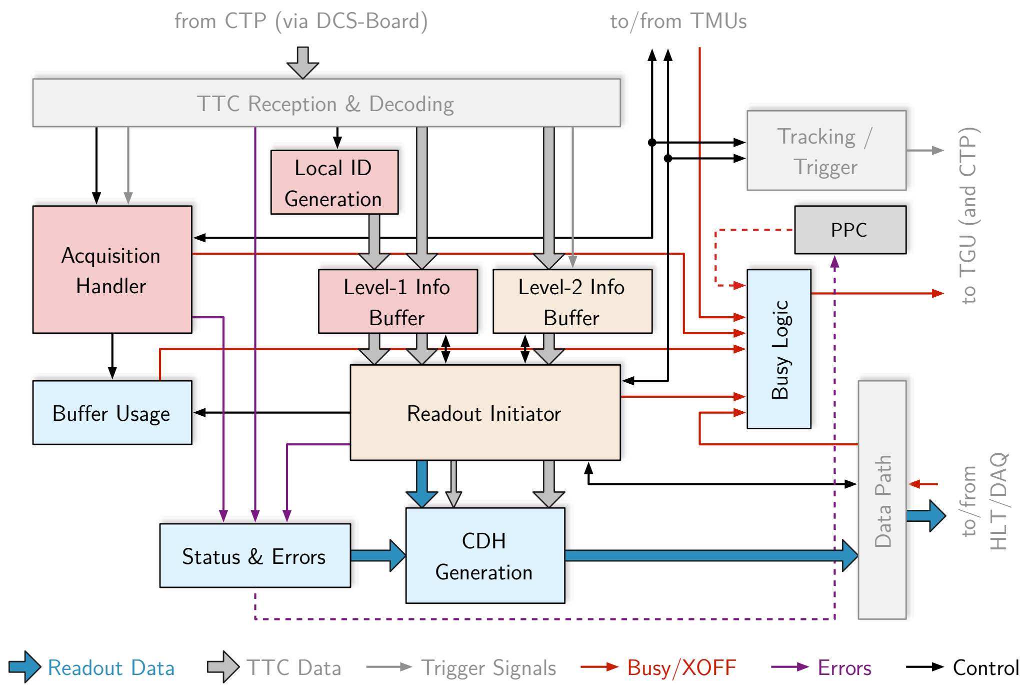


Figure 3.9: Overview of the multi-event capable control logic in the SMU. L0 and L1 related trigger events are processed by the acquisition handler, which administers the recording of required sequence and ID data to the L1 information buffer. Once a L2 trigger arrives, the readout initiator processes this information and either discards the event or initiates the readout process.

transition from IDLE to SEEN_L0 and starts a timer. It also activates the `expect_data` signal, which gates the reception of chamber data at the TMUs, see Section 4.1.3. At a certain, configurable value of the timer, the FSM moves to WINDOW_L1, in which a potential L1 trigger is expected. No L1 within the acceptance window[14] moves the FSM back to IDLE via NO_L1 and releases `busy_l0`. The associated falling edge of `expect_data` without L1 is interpreted as `flush`, which on the TMUs empties the currently active event buffer slot from potentially recorded tracklet data and flushes all pipelines. On the SMU, non-completed entries in the tracking and trigger buffers are flushed.

In case a L1 is received within the acceptance window, the signal `valid_L1` is transmitted to all active TMUs. The FSM starts the L1 message timeout timer and moves, through SEEN_L1, to WINDOW_L1M, where the L1 message is expected. Once it has been received after typically some 10 µs, see Table 3.1, the acquisition handler waits in DATA_RECEIVE until all active TMUs have reported that the reception of event data is finished. By raising this `rx_complete` signal, the memory management of the TMUs guarantees that for each seen rising edge of `valid_l1` an event can be rejected or accepted at the L2 stage without causing a corruption of the event memories. The handler completes the cycle by placing an

[14] A width of 1 suffices for the window in production operation, but it can be configured wider for commissioning and/or testing purposes.

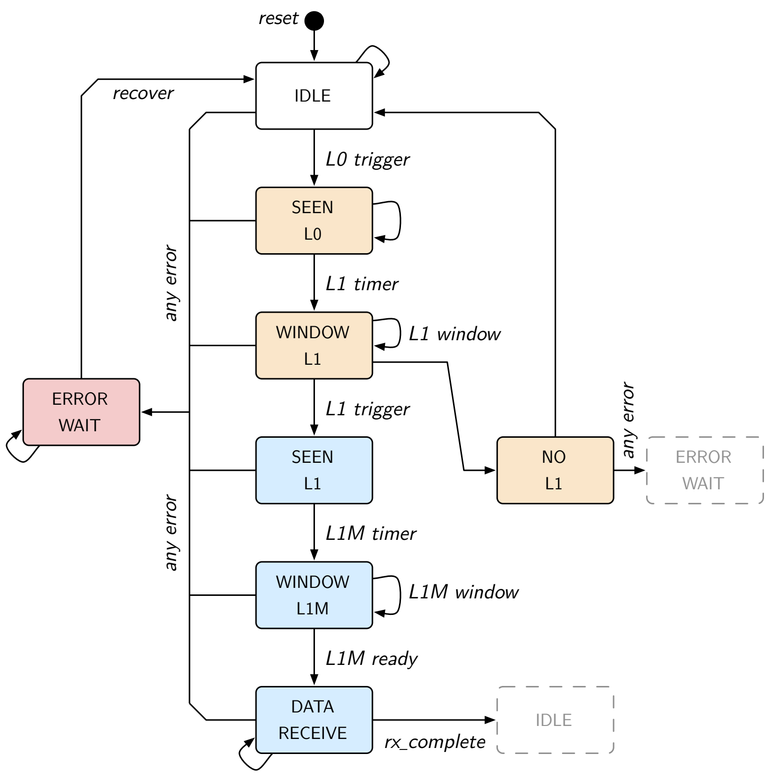


Figure 3.10: Event handler FSM. Colored states highlight the different dead time, i. e., busy contributions, t_{L0} (yellow) and t_{rx} (blue). Also in case of a detected error, the system is halted by raising busy (red).

entry in the L1 info buffer and increases the buffer usage counter before moving to IDLE. Entries of the L1 info buffer, see Figure 3.11a, contain all relevant information extracted from the L1 trigger message and the local event ID sampled at the L0.

Attributing Event Identification to Data Before having passed the L2 trigger stage, events in ALICE are identified using the bunch crossing in which the collision took place and the corresponding orbit number. The 3564 potential bunches in an LHC orbit are identified using a 12-bit number. Together with the 24-bit orbit number, this allows for a unique identification of the event within a timespan about 25 min.[15] This global event identification is distributed by the CTP as part the L2 message. Upon readout at L2, the DAQ extends this event ID with a period number to allow for unique identification over the complete runtime.

In the chosen implementation, the buffering of an event in the acquisition phase is decoupled from the readout or reject upon L2. As crucial element in the readout chain that attributes event ID to event data, the GTU segments implement local event identification counters

[15] Exactly $2^{24} \cdot 3564 \approx 59.79 \cdot 10^9$ LHC clock cycles.

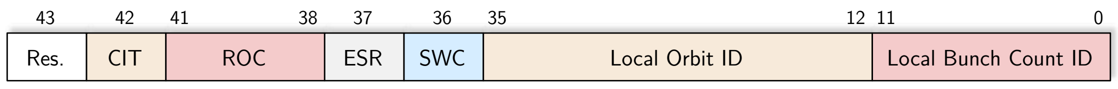
exhibiting the same wrapping characteristics as the global bunch/orbit counters. The local event ID is sampled at the arrival of a L0 and enters, alongside other information, into the L1 info buffer when the end of the data transfer to the segment's event buffers is confirmed. The event buffers on the TMUs and the L1 info buffer are operated in sync and perform identical fetches and drops. In case of correct ID assignment and with deterministic latency with respect to the local bunch counter reset command, local and global ID must always exhibit a constant offset. During preparation of an event fragment for readout, local event ID from the L1 info buffer and global event ID from the L2 info buffer are both written to the CDH, which is prepended to the event data fetched from the corresponding event buffer slot. The DAQ samples, as independent entity, the offset between local and global ID with the first event of the run and monitors it throughout a run. It would raise a critical error if an offset change indicating an incorrect attribution of an event ID to event data is observed.[16]

3.5.2 Readout Phase

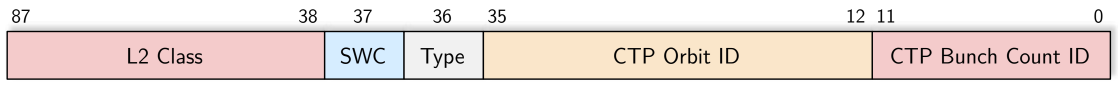
In the readout phase, which lasts for few µs to few ms depending on the type of trigger and event size, each arriving L2 corresponds to one occupied event buffer slot on the TMUs. The readout initiator cyclically processes entries in L2 info buffer, which represent L2 triggers in chronological order, and generates the required control signals for readout or reject.

Readout Initiator L2 triggers may arrive while the readout initiator is occupied with processing of an event or while the event transfer is not yet finished. The parts of the L2 message that are relevant to further processing must therefore be stored temporarily until the readout initiator, shown in Figure 3.12, is ready to process them. Figure 3.11b shows an entry of the corresponding L2 info buffer. ID information, trigger class and software class flag are necessary to build the CDH for the event fragment. The type flag distinguishes L2A and L2R triggers.

[16]No ID mismatch has, so far, been observed with commissioned designs in production data taking runs.



(a) Event info buffer entry



(b) L2 info buffer entry

Figure 3.11: Entries of the event info and L2 info buffers. Entries in the event info buffer contain local event ID as well as information shipped in the L1 trigger message. The L2 info buffer holds relevant parts of the L2 trigger message.

As soon as an entry in the L1 info buffer is available, i. e., all active TMUs have finished reception of the full event, the readout initiator FSM fetches the entry and advances from IDLE to the state WAIT_L2. If (or when) an entry in the L2 info buffer is available, the entry is fetched and processing continues in CHECK_EVENT. The global event ID contained in the L2 trigger message is matched against the local ID stored in the corresponding L1 info entry. A successful match in case of a L2R results in an `event_reject` signal, which is distributed to throughout the segment. A handshake has been introduced to avoid any race conditions in the communication with the TMUs, where the `event_reject` enters several clock domains. The initiator waits in state WAIT_L2R_ACK for the confirmation `reject_ack` that the corresponding buffer slot is freed in all active TMUs. Having received it, it decrements the buffer usage count and moves to IDLE.

A successful ID match in case of a L2A moves the FSM from CHECK_EVENT to DATA_TX and starts the dispatcher (Section 4.6) using `start_readout` to commence the local event building. Once the dispatcher has pushed all words of the event fragment into the DDL transmit buffer, it reports `tx_complete`. The initiator decrements the buffer usage count, moves to IDLE and is ready to process the next L2 trigger.

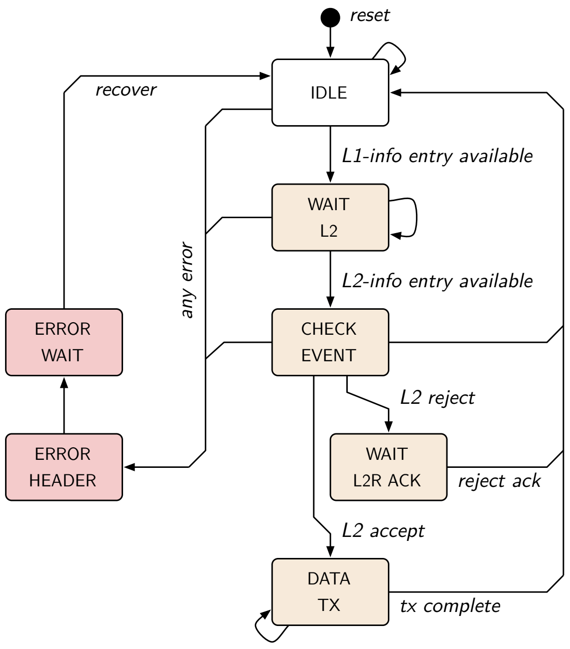


Figure 3.12: Readout initiator FSM

3.5.3 Busy Generation

The GTU generates the busy signal to protect the TRD FEE from new trigger sequences while it is processing or holding data of an event. Contributions are independently generated per supermodule and then merged to a global contribution communicated to the CTP.

Supermodule Busy The busy required to protect a given supermodule is generated by the acquisition handler. For reasons of internal accounting, the busy is split, like indicated in Figure 3.6, into a contribution `busy_l0` of fixed duration for each started trigger sequence and, in case of a L1A, a contribution `busy_rx` that depends on the duration of the FEE-GTU raw data transfer. These individual contributions correspond to the acquisition handler, see Figure 3.10, being in the yellow and blue-colored states, respectively. In terms of absolute generated dead time per event, t_{L0} contributes 6.5 µs in Run1 and 7 µs in Run2. For t_{rx}, values typically range from 20 µs in p-p to 100 µs for central events in Pb-Pb, see Chapter 7.

In regular operation, a third, supermodule-level contribution to dead time t_{ro} originates at the event buffers. If the buffer usage count is incremented, which always coincides with the release of the `busy_rx`, and no buffer slot is free, the system cannot accept new data from the detector and therefore has to inhibit new trigger sequences by raising `busy_ro`. The contribution `busy_ro` is generated from a simple comparison of buffer usage counter with a predefined threshold that represents the number of buffer slots on the TMUs. It always becomes active in the same clock cycle as `busy_ro` is released, which allows for a simple OR of the three contributions to form the segment busy contribution `busy_sm`. Once the readout initiator finishes processing of the current L2 info buffer entry, the buffer usage count is decremented and `busy_ro` is automatically released.

Additional contributions to the segment busy are active only in the run start-up phase, e. g., the LDC signaling its unreadiness to accept events from GTU, or in the error case, e. g., busy contributions from acquisition handler or readout initiator in case a fatal error related to the trigger sequences is detected.

Global Busy Generation The global TRD busy signal is formed on the TGU from the contributions `busy_sm`$_i$ of the individual segments. The detector is busy whenever at least one of the active segments reports busy. The global TRD busy signal communicated to the CTP is therefore a Boolean OR of the segment busy contributions (that have previously been masked by the mask of active segments participating in the run). After issuing a L0, the CTP implements a grace period of 1 µs before the busy from the detectors in the partition is expected to be set [68]. This time is more than sufficient to cover the L0 transmission latency, the propagation from SMU to TGU, and transmission of busy to the CTP.

Avoiding Implicit Dead Time The PT/LM system generates wake-up triggers for the FEE to start the acquisition prior to a L0 arriving from the CTP. To not disturb the FEE, new wake-up signals must be suppressed if the acquisition or processing of an event is not yet completed. In a first implementation of the PT system, new wake-ups were inhibited for a fixed period of time after a L0. The length of the inhibit interval is determined using a worst-case assumption for the FEE-GTU transfer time of approximately 300 µs. As L0 triggers are rejected in a readout partition with the TRD if there is no L0 contribution from the PT system, this approach causes implicit dead time. To eliminate it, the GTU was extended to additionally feed its busy signal to the PT, where it is used to gate wake-up triggers for the exact right amount of time.

3.5.4 Trigger Sequence Verification

Since the SMU attributes event ID and event data, it is essential to detect errors that could lead to an incorrect attribution. An undetected error could have severe consequences and render several hours of run useless. To guarantee the integrity of the event data and ensure stable operation of each segment, all trigger sequences are verified before the corresponding control signals are generated and distributed within the segments.

Error Detection While the TTC receiver monitors the transmission of trigger signals and messages on both channels of the TTC system, see Section 3.4.3, the acquisition handler and readout initiator FSMs feature built-in detection capabilities related to timing, succession and content of individual trigger events of a sequence. The required additional logic adds only little to the complexity of the acquisition handler and readout initiator FSMs and allows for an efficient implementation. Erroneous conditions are evaluated with high priority and supersede potentially simultaneously arriving trigger events.

Once the acquisition handler leaves the IDLE-state, no further L0 trigger is allowed and an occurrence would send the FSM to ERROR. The handler oversees the observance of time limits for the potentially following L1 trigger and arrival of the L1 message. An occurrence of these events outside the specified, valid time intervals also moves the FSM to the ERROR state. Furthermore, the acquisition handler features an input to signal a significant problem with the reception of a trigger. For example an unknown signal encountered on channel A and signaled by the TTC receiver would force the handler to ERROR.

The readout initiator checks that the global event ID transmitted as part of a L2 trigger message matches an entry in the event buffers. The buffered event is identified by the local event ID, which is contained in an entry of the event info buffer. Due to the chronological arrival of L2 trigger messages and the sequential processing of buffered events, the order of corresponding entries in the event info buffer and L2 info buffer matches during regular operation. An ID mismatch signals a severe inconsistency of the received trigger information and therefore stalls further processing by moving the readout initiator to ERROR. An input to the readout initiator can be used to signal an external error, e. g., an error related to the data transmission to the DAQ.

Table 3.2 lists the error conditions that can be detected and have the potential to corrupt the GTU readout as well as their type and severity. A single-bit transmission error on channel B of the TTC system is the only condition that is not critical. It can be restored transparently using the Hamming code, but is reported in a dedicated field of the CDH in order to detect, e. g., a deteriorating optical connection.

Error Condition	Description / Detected by	Critical
ChA encoding	Illegal encoding detected on TTC channel A TTC receiver	yes
ChB single bit	1-bit transmission error on TTC channel B, recoverable TTC receiver	no
ChB double bit	2-bit transmission error on channel B TTC receiver	yes
ChB word	Trigger message incomplete, words missing TTC receiver	yes
L0 spurious	L0 trigger received while busy Event handler	yes
L1 spurious	L1 trigger received outside L1 decision interval Event handler	yes
L1M spurious	L1 message received without L1 trigger Event handler	yes
L1M timeout	L1 message not received within expected interval Event handler	yes
ID mismatch	Mismatch between local and global bunch count ID Readout initiator	yes

Table 3.2: Errors related to received trigger sequences that can be detected by the multi-event control unit of each GTU segment. The processing is stopped immediately on encountering a critical error condition, giving the expert the opportunity to conduct a detailed error analysis.

Error Handling and Recovery Contrary to errors related to event data, which do not have a direct influence on the control flow in the GTU and where an indulgent handling offers certain benefits, see Section 4.1.3, trigger errors are severe and their occurrence generally disturbs the control logic.

At design time, it was not entirely clear whether advanced strategies to cope with errors related to illegal triggering would be required. The existence of a fully automatic recovery and reporting procedure for detectors is suggested in [51]. The architecture shown in Figure 3.9 therefore was developed with a hardware/software co-design in mind. The hardware part, which is intentionally kept as simple as possible to minimize the potential for bugs, handles standard operation in quasi real time. In case a trigger-related error is detected, the idea then was to transfer control to the PPC system, which offers the possibility to implement a complicated recovery procedure in software. It would halt the system (by raising the busy signal), conduct, with the help of the TTC trace and information on the system state, a sophisticated analysis of the events leading up to the error, report it, recover and then resume standard operation within few ms.

The approach taken was to implement the hardware part first and then potentially augment it with the PPC-based error handling. Experience however showed that while it is essential to equip the GTU with powerful error detection and debug capabilities allowing for manual inspection, a fully automatic handling of and recovery from trigger-related errors is not required. In the presented implementation, an error detected by the acquisition handler or readout initiator intentionally moves them to the `ERROR` state, where they remain. In the error state the busy signal is permanently set, which is immediately noticed by the shift crew. Using the full integrated trace and debugging functionality, see Section 6.3.2, the notified on-call expert can trace the problem and assess whether it is within the GTU or originates further upstream. All information to narrow down the cause of the error is available via the system console. The recovery, which needs a reconfiguration and restart of the system, is initiated by the expert via the DCS once the error is understood.

Those errors present in the early phase of the experiment, e. g., spurious L1 triggers sent by the CTP causing readout partitions to malfunction and stall, were detected, successfully identified by the GTU on-call expert with help its debugging features – especially the detailed TTC trace – and, as a consequence, remedied at the origin. Hence the stability of the trigger system was significantly improved and did not justify or require the extension of the GTU segment control logic with PPC-based automated recovery. Although the error detection logic is in place and active, since then no further occurrences of severe errors related to the CTP/TTC complex have been observed.

3.5.5 Number of Supported Event Buffers

The number of event buffer slots the presented logic can manage is only limited by the depth of the L1 and L2 info buffers. Implemented as BRAM-based FIFOs, they already allow to store more than 100 entries in the current implementation.

Furthermore, although the control logic handles the buffer slot accounting, it is also compatible with a scheme in which the TMUs independently manage their buffers. This would allow to move from a slot-based event buffer management to a more space-efficient circular-buffer type management, in which each TMU provides a `full` to signal if the buffering of one more event cannot be guaranteed. The buffer threshold on the SMU would be set to a large number and serve as an absolute upper limit of events simultaneously in the buffers, while the immediate contributions to the `busy` are the `full` signals from the TMUs.

Summary

Low dead time and robust operation are important aspects of readout systems for modern particle detectors. The buffering of multiple events is a way to de-randomize input rate fluctuations and, as a consequence, lessen detector dead time for given average input rates. An analysis of the specific dead time sources of the TRD and the trigger operation in ALICE, leads to the presented architecture for the trigger reception and segment controller, which are both capable of handling multiple interlaced trigger sequences. The segment controller, whose implementation is presented in detail, cyclically processes received triggers to administer the buffering of multiple events. Furthermore, it features extensive verification capabilities to ensure correct operation and data integrity.

4 Buffering and Local Event Building

When the TRD starts the acquisition processing of an event with a pretrigger, more than 65 000 MCMs generate data at a rate in excess of 15 TB/s. While tracklets are immediately transmitted when calculated, the acquired raw event data is held in the filter memories of the dedicated CPUs on the detector front-end until a positive trigger decision is taken at the L1 stage. During the tracklet or raw event data transmission, 540 detector chambers[1] push event data to the GTU via 1080 unidirectional optical links operating at a line rate of 2.5 Gbit/s. With no means of flow control to the front-end, the GTU receives this data at full sender bandwidth and implements buffering and local event building for the TRD.

The bandwidths of the interfaces and the relevant trigger stages in the TRD event data flow are summarized in Figure 4.1. Data is received at the GTU at an aggregate rate of 270 GB/s. For readout to the DAQ an aggregate output bandwidth of 3.43 GB/s, realized by 18 optical links at a line rate of 2.125 Gbit/s, is available in Run1.

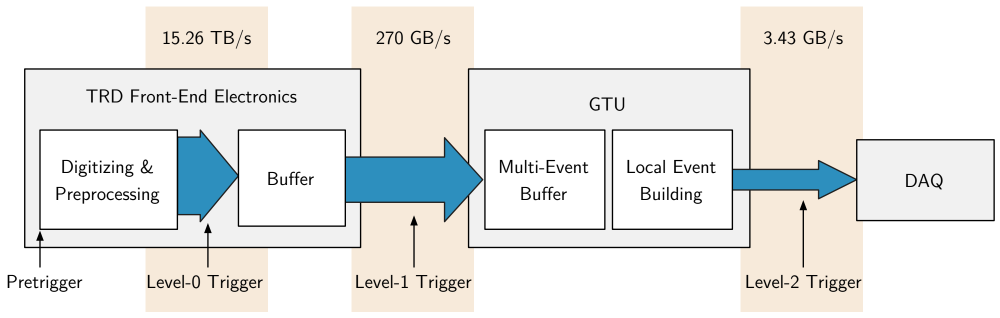


Figure 4.1: TRD event data flow, trigger events and bandwidths (Run1). After a sequence is confirmed by a L1 trigger, event data is moved off detector to the GTU for buffering and potential readout, if a L2A is received.

The handling of raw event data in the GTU, which is steered by the control unit presented in Chapter 3, is non-trivial not only because it involves large data rates. It must also implement the reception and buffering of multiple events in a robust way, which can cope with occasional misbehavior as exhibited by detector chambers operating in a high-radiation environment. An efficient and reliable intra-segment communication is required, which meets the latency requirement of the trigger application, but also provides sufficient bandwidth to saturate the readout link in order to minimize the readout time. Challenges with respect to the local event building arise from the fact that data entering the event

[1]In PHOS-hole configuration, the central stacks of three supermodules are missing, giving a total of 522 installed chambers.

fragment is generated by multiple entities in different levels of the design hierarchy. These produce their results at differing points in time in a trigger sequence, which calls for suitable multi-event capable buffering structures to include their data, e. g., intermediate trigger results required for offline analysis, in the event fragment stream. This chapter presents the firmware designs of TMU and SMU with focus on the handling of raw data and the local event building.

4.1 Data Reception

Each TMU processes and buffers the data arriving via the 12 optical fibers from its associated detector stack. After the optical-to-electrical conversion in the SFPs, the signals are routed to dedicated transceiver blocks in the FPGAs. An overview of the event buffering architecture of a TMU node, which is discussed in the following sections, is presented in Figure 4.2.

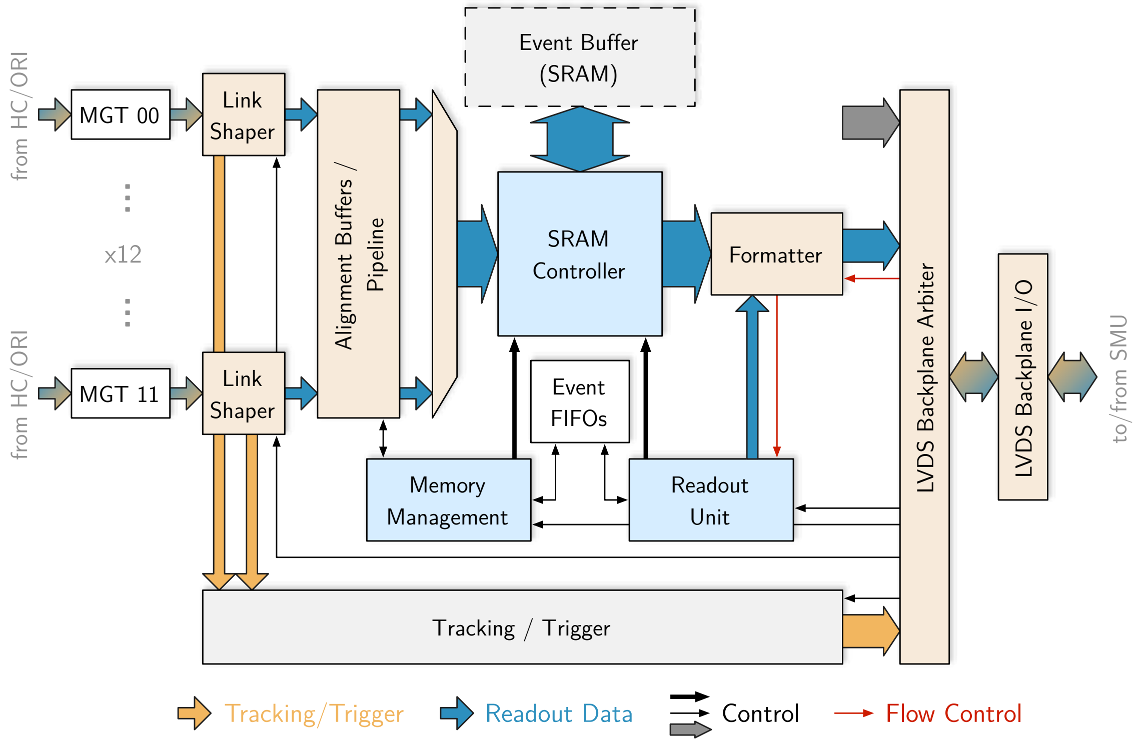


Figure 4.2: Overview of the data reception and event buffering architecture on a TMU node. Link data shapers ensure that link data is correctly formatted for processing in the subsequent units, irrespective of the half-chamber behavior.

4.1.1 High-Speed Serial Input Interface

FPGAs of the Virtex-4 FX series provide dedicated Multi-Gigabit Transceiver (MGT) blocks for transmission and reception of high-speed serial signals. The 12 MGTs used on a TMU are organized in two columns, each with three tiles. The two MGTs in a tile share a Phase-Locked Loop (PLL) to provide a common sender reference clock. Receiver-wise, they operate independently and two PLLs, one for each MGT, provide the clock reconstructed from the received data.

The MGTs consist of two major functional blocks: The Physical Media Attachment (PMA) block generates or receives the ultra high-frequency serial signals, while the configurable Physical Coding Sublayer (PCS) block handles, e. g., encoding or decoding of the parallel sender or receiver data. For detailed information about the configuration of the MGT parameters in this application, the reader may refer to [98]. Each MGT exposes a 16-bit wide interface at the reconstructed receiver clock of 125 MHz to the FPGA fabric. With 12 links per stack this results in an incoming peak data rate of 24 Gbit/s at each TMU.

4.1.2 Data Transmission from FEE to GTU

The phase of the current trigger sequence defines whether the data word sent by the half-chamber is a tracklet or a raw data word. Tracklets are only transmitted in the L0-L1 timeframe, while the raw data transfer commences only after a L1A trigger has been received. Transmission on the FEE-GTU links follow a basic format, see [94], with 8b10b-encoded 16-bit words and no forward error correction. Tracklet and raw data endmarkers notify the TMUs of the end of the transmission on a given link. They are of form `0x1000` and `0x0000` for tracklet and raw event data endmarkers, respectively. As illustrated in Figure 4.3, the 16-bit endmarkers are sent four times for reasons of increased reliability in case of bit errors. Even if no tracklets or raw event data are to be transmitted, the merger MCM of the half-chamber is required to produce appropriate endmarkers. Tracklets and raw event data are 32-bit on the FEE, but split for the transmission into two 16-bit words. Consequently, the sets of four 16-bit endmarkers are aligned to a 32-bit word boundary if the link operates correctly.

Endmarker Evaluation With respect to the tracking algorithm, a missing endmarker has no direct effect. Operation of the tracking commences with the arrival of the first tracklet words and proceeds normally, endmarkers characterize only the end of the tracklet arrival. With endmarkers not detected, the tracking does not reach a TRACKING_DONE state for the given event, but is fully operational again for the next. Information about the tracklet timing and whether the TRACKING_DONE state is reached is stored and contained in the readout data for subsequent offline analysis.

Concerning the raw data transmission from the detector, the arrival of endmarkers directly affects the dead time. An upper limit for the maximum transmission time on a link follows from considerations of maximum half-chamber event size and FEE readout tree latency.

However, since a chamber with full channel occupancy, and thus maximum transmission time, is never encountered in real operation, the use of a timer-based approach with maximum transmission time to signal the end of the raw data transmission would generate excess dead time. To determine the end of the transmission, detection of the endmarkers is crucial in order to minimize the corresponding dead time t_{tx}.

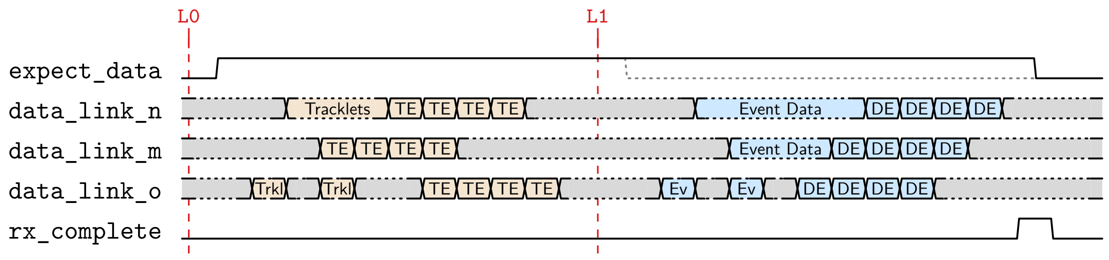


Figure 4.3: Transmission of tracklet (orange) and raw data (blue) words from FEE to GTU, each completed by four dedicated endmarkers (`TE`/`DE`). The link with the longest raw data transmission determines the time at which the TMU signals the availability of an event (`rx_complete`) to the MEB control unit on the SMU. Link data shapers monitor the link data, forward only data received within the gating defined by `expect_data` and reshape the event if required.

4.1.3 Handling of Input Data

For each event accepted at the L1 stage, the MEB control unit places an entry in its L1 info buffer. The entry is placed once all TMU event handling units have reported the buffering of the event to be finished. In regular operation, this corresponds to the successful detection of raw data endmarkers of the active link with the longest transmission time. In order to limit the complexity of the MEB control logic, the TMUs guarantee that for each validated L1 trigger a buffered event exits, which can be accepted or discarded at the L2 stage.

Link Format Violations Although individual detector chambers are reliable, their large number, the complexity of their readout-tree and the fact that they operate in a harsh radiation environment results in a non-negligible probability of a malformed transmission of half-chamber data. In such a case, the data does not exhibit the expected timing and endmarker behavior as illustrated in Figure 4.3. Faulty chamber readout behavior has been observed with symptoms ranging from no data being sent at all to continuous transmission of illegal data outside of the defined windows. The reasons for illegal chamber behavior are under investigation by TRD FEE experts. Various error sources have been identified, among them single-event upsets in the FEE, problems with trigger distribution within the pretrigger system, issues with voltage regulators and others. Depending on the exact cause of the underlying error, the occurrence frequency observed ranges from isolated incidents, which only affect a single link once, to several chambers constantly exhibiting illegal behavior from a given point in the run onward.

Indulgent Handling of Link Format Violations Singular occurrences of incorrect link behavior mostly occur on a half-chamber granularity. Unlike an error related to the CTP/TTC complex, the errors related to the link format at most corrupt the data of the given links, which generally does not justify an abort of the run. Events containing erroneous link data may – if the incident is detected correctly – be (fully or partially) ignored during the offline processing. When dealing with isolated occurrences of erratic link behavior, the penalty of a lost event is negligible when compared to the time in the order of minutes required for a run stop-reconfigure-restart cycle. In order to allow for stable running, the TMUs therefore must be able to detect and cope with all varieties of incorrectly formed data coming from the detector.

For this purpose, the TMUs implement shaping units to analyze the incoming data stream on all input links from the detector. The rising edge of the `expect_data` signal, which is generated by the segment controller, starts the shaper operation. With it begins a gating interval in which tracklets, raw data words and the appropriate endmarkers finalizing the link transmission are expected. Incoming data words are monitored by the shapers for the presence of the valid raw data endmarker pattern, i. e., four successive 16-bit endmarkers and alignment to 32-bit word boundaries. Only within the interval defined by `expect_data` received data words are passed through to the tracking units, the adjacent alignment buffers and CRC stages. In standard operation this interval is closed by the segment controller once correctly aligned raw data endmarkers were encountered on all active links, or if the event is rejected at the L1 trigger stage.

In case of a L1A and a misbehaving half-chamber, i. e., no valid endmarker pattern is detected by the corresponding link data shaper, the end-of-transmission time-out fires after approximately 265 µs.[2] The shaper then discards all potentially arriving data of its link and flags the link data as erroneous. In case misalignment is detected, the link data is padded to 32 bit. In all error cases, the valid raw data endmarker pattern is restored or appended to bring the link data into a form eligible for downstream processing. Furthermore, the SMU is notified of the end of the transmission.

The flag fields, each covering all 12 links of a stack, are available in the generated event data stream as part of the stack header section. They are accessible by monitoring the output data stream, e. g., via the AMORE [61] monitoring framework, and provide valuable real-time information about the state of the FEE. In case of a high occurrence rate, an increasing number of link format errors or failure of several links, the shift team can decide on basis of the dead time associated with the FEE-GTU data transmission and link shaper flags in AMORE whether a chamber reconfiguration and restart of a run is justified or not.

The reshaping and restoration of structural integrity of malformed link data defines a fixed worst case event size. More importantly, it guarantees a defined link data format even with malfunctioning links. This makes it trivial to guarantee, that for each received, valid L1 trigger an event is stored in memory and enables a uniform processing of data in

[2]This time is defined by the transmission time of the largest event encountered during operation, namely a dump of the detector configuration memory, plus a small safety margin, see Section 4.2.2.

subsequent stages of the event handling. This significantly reduces the complexity of all downstream entities and also the generation of control signals.

Dynamic Detection of Active Links For an efficient implementation of the input data shaping and consistent handling of the link data in the event buffering, the GTU requires information which input links are supposed to deliver data. Therefore, each TMU captures the mask representing the currently active links at the start of each new trigger sequence. It is propagated together with the link data for consistent processing in all following stages. This link mask is generated like

```
link_mask <= sfp_con and (not sfp_loss) when mode = '0' else
             ppc_link_mask;
```

with information from the SFP status monitoring. Links without connected fiber or no optical power are automatically disabled, no explicit configuration has to be applied to the GTU.[3] The TRD operator can disable malfunctioning chambers simply by sending them to the STANDBY state.[4] Due to the capturing and propagation of the link mask within the GTU, a half-chamber may be disabled anytime without disturbing a potentially ongoing run.

4.1.4 Input Stream Compactification

Words arriving on a link are not valid data words in every clock cycle, but are interspersed with idle characters. For each input link, an alignment buffer compactifies 16-bit words received from the link shapers to dense 128-bit lines. The design of the alignment buffers was carried over from their initial implementation presented in [98]. Main building blocks of the alignment buffers are dual-port 18 kbit BRAMs, which also handle the crossing from the MGT to the SRAM clock domain. When the end of the transmission is signaled, partially filled lines are zero-padded to 128 bit. Completed lines are fetched from the 12 alignment buffers in a round-robin scheme for input to the SRAM controller write side data FIFO. At 125 MHz, the alignment buffers construct a completed line in no less than 64 ns. A round-robin fetch of completed lines in the 200 MHz SRAM domain corresponds to a potential line read every 60 ns. Even when receiving at full sender rate, the alignment buffers are generally drained and do not overflow.

4.2 Event Buffering

The implementation of the push data path capable to absorb and buffer the 12 independent link data streams at full rate is one of the key requirements to the TMU event buffering

[3]The contribution `ppc_link_mask` can be controlled by the user via console application. It serves as override required in certain debug or test setup scenarios only.
[4]If a half-chamber is in STANDBY, the optical power of the ORI is disabled.

design. A second requirement is the support for buffering of multiple events with accepts and rejects at the L1 and L2 trigger stages.

4.2.1 Prioritizing SRAM Interface

Each TMU is equipped with two 512 × 36 kbit DDR2 SRAM chips,[5] which feature a 2-word burst architecture, to serve as buffer for event data. In one cycle, they allow for read or write access to two subsequent 36-bit dates at a latency of two and one clock cycles, respectively. The two memories are arranged in a parallel configuration and present a 72-bit wide data path at 200 MHz DDR to the fabric.

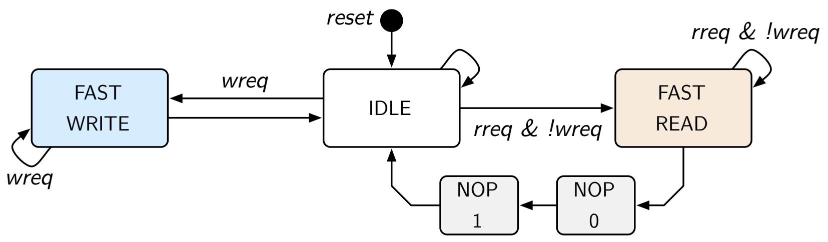


Figure 4.4: SRAM controller FSM. A write request for a line immediately aborts a potentially ongoing read. Leaving a read, two no-operation cycles are required before the write can be issued. The slow interface, which is connected to the PPC system for in-system debugging purposes, is not shown.

The controller encapsulates the 72-bit DDR output logic and implements the interface to adjacent entities via two sets of 128-bit wide data[6] and 18-bit wide address FIFOs at 200 MHz for read and write, respectively. Data lines collected from the alignment buffers are placed in the data input FIFO. The corresponding link write addresses, which are calculated by the memory management, are simultaneously placed in the input address FIFO. The `not_empty` signal of the input address FIFO indicates a write request `wreq` and causes the controller FSM, see Figure 4.4 for a state diagram, to immediately abort a potentially ongoing read. To satisfy the maximum push bandwidth requirements of 24 Gbit/s during detector readout, write requests are strictly prioritized over potentially pending reads. While consecutive reads or writes to arbitrary addresses may be executed every clock cycle, abort transitions from read to write introduce a penalty of two no-operation cycles.

The maximum combined r/w bandwidth available is 25.6 Gbit/s, which compares to a peak rate of incoming data of 24 Gbit/s, However, the link data streams are not dense and exhibit transmission pauses caused by handshakes and buffering in the hierarchical FEE readout tree [104]. This generally leaves sufficient resources to retrieve data for readout while a transmission from the FEE is ongoing and saturate the effective output bandwidth to the DAQ of 1.6 Gbit/s per supermodule in Run1, see Section 7.1.3.

[5]Cypress CY7C1320BV18-200BZXC 18 Mbit DDR2 SRAM
[6]One bit per byte is reserved for parity.

4.2.2 Memory Management

Due to the independent nature of data arrival on the input links, the 4 MiB SRAM of each TMU is partitioned into 12 independent memory regions, one for each input link. The memory management unit performs the calculation of SRAM write addresses for the given regions. Its support for accepts or rejects at the L1 trigger stage is steered by the control signals originating at the segment controller.

Event Sizes and Memory Partitioning To implement the support for multiple events in the individual link memory regions, two possibilities exist: a static layout, which supports a fixed number events, or a dynamic layout, where the individual link memory regions are organized as circular buffers and the number of storable events depends on the respective event sizes.

In the static layout, each link memory region is divided into N equally-sized slots. The `buffer_full` signal and, consequently, readout-related busy contribution `busy_ro` are raised when all slots are occupied. The number of buffers N is determined by the maximum expected data volume and overall buffer capacity. Typically, the largest data volume is produced when the fully illuminated detector sends uncompressed raw event data (black events). In case of the TRD, the largest black event link data volume is exhibited in runs with a 30-timebin configuration. Table 4.1 lists the data volume for back events at different granularity for 22 and 30 time bins. However, even larger link event sizes are produced by configuration dump events. These are sent with the first L2A trigger following the configuration of a half-chamber and contain a full snapshot of the chamber state and its configuration. With about 62 KiB per link, transmission duration for these events exceeds those of the largest expected black events and thus determines the time-out threshold of the input shapers. Adding a small margin, it is correspondingly set to approximately 265 µs. Without flow control on the links to the front-end, the link shaper cut-off time then defines the maximum data volume each buffer slot must be able to hold. Assuming a misbehaving chamber constantly produces 16-bit data for the entire length of the interval gated by the link shapers, the number of slots each link memory region can be divided into in a static layout is

$$\left\lfloor \frac{4\,\text{MiB}}{12 \cdot 265\,\mu\text{s} \cdot 125\,\text{MHz} \cdot 16\,\text{bit}} \right\rfloor = \left\lfloor \frac{4\,\text{MiB}}{12 \cdot 64.7\,\text{KiB}} \right\rfloor = 5\,.$$

A dynamic layout with a circular-type buffer only requires that the TMUs signal if the available memory for any link is less than 65 KiB. Not imposing this worst case requirement on every slot, but only on the last one, would consequently allow for a more efficient buffer utilization, especially with link event sizes in central Pb-Pb in the order of 15 KiB. However, the more complicated address logic required in the fast 200 MHz domain imposes tight timing constraints that prove challenging in an FPGA of this generation. Considering the already high resource utilization in the TMUs due to the tracking algorithm, see Appendix A, this would substantially increase the design complexity.

	22 Time bins		30 Time bins		CfgDump	
TRD	33.1	32.2†	45.1	43.9†	62.1	[MiB]
Supermodule	1.8	1.5‡	2.5	2.1‡	3.5	[MiB]
Stack	396.0	297.0*	540.0	405.0*	744.0	[KiB]
Link	33.0	24.8^{+}	45.0	33.8^{+}	62.0	[KiB]

Table 4.1: Event sizes for 22-timebin, 30-timebin and configuration dump events. TRD and supermodule sizes in the highlighted columns are calculated with the regular 5-stack supermodule configuration. Stack and link sizes are for stacks of type C1 (16 pad rows) and links originating in these stacks. Non-highlighted columns show the values for: TRD with PHOS holes (†), supermodule with missing central ($z = 0$) stack (‡), central C0-type stacks with 12 pad rows (*), links originating in central stack ($^{+}$).

Although a large number of buffer slots may be –depending on the system workload– beneficial with respect to readout related dead time, simulations of a common readout partition with TRD in typical operating conditions, see Section 7.4, suggest that already a comparably low number of buffers is sufficient to match the performance of other detectors in a common partition. Therefore, a static layout is chosen for the actual implementation. The complexity of the address logic in the 200 MHz domain, which turns out the be the critical path in the implementation, is significantly reduced when $n = 2^x, x \in \mathbb{Z}^+$. Consequently, $N = 4$ buffer slots are implemented with the result of an intentionally simple, yet very reliable address logic. However, in the course of the experiment detector requirements and operating conditions may change. The memory management is therefore encapsulated as separate module to support future upgrades to advanced buffer management schemes, such as a dynamic layout. The segment control logic presented in Section 3.5.5 already supports such schemes. Furthermore, alternative possibilities for further reduction of the readout dead time τ_{ro} are discussed in Section 7.6.

Write Address Calculation and L1 Stage Rejects/Accepts The pipelined structure of the data path allows for a latency of two 200 MHz cycles for calculation of an address associated with a given link. The memory management unit receives as input the `link_index`, which identifies the alignment buffer sourcing the current SRAM line, and a `line_valid` signal. The current write positions for the 12 individual links are internally represented as a register bank. These write pointers are relative to the offset of the link memory region plus current buffer slot offset and start with zero for the first word of an event. Figure 4.5 illustrates the partitioning of the event memory. A `line_valid` increments the write pointer of the currently active link signaled by `link_index`. The start address of the currently selected buffers slot and the start address of the current link memory region are both added to the relative write pointer to obtain the resulting global SRAM `line_address`.

The memory management implements the support for accepts and rejects at the L1 trigger stage. Depending on whether the option to record tracklets along with event raw data is enabled, data enters the SRAM already in the L0-L1 timeframe (tracklets) or only after a

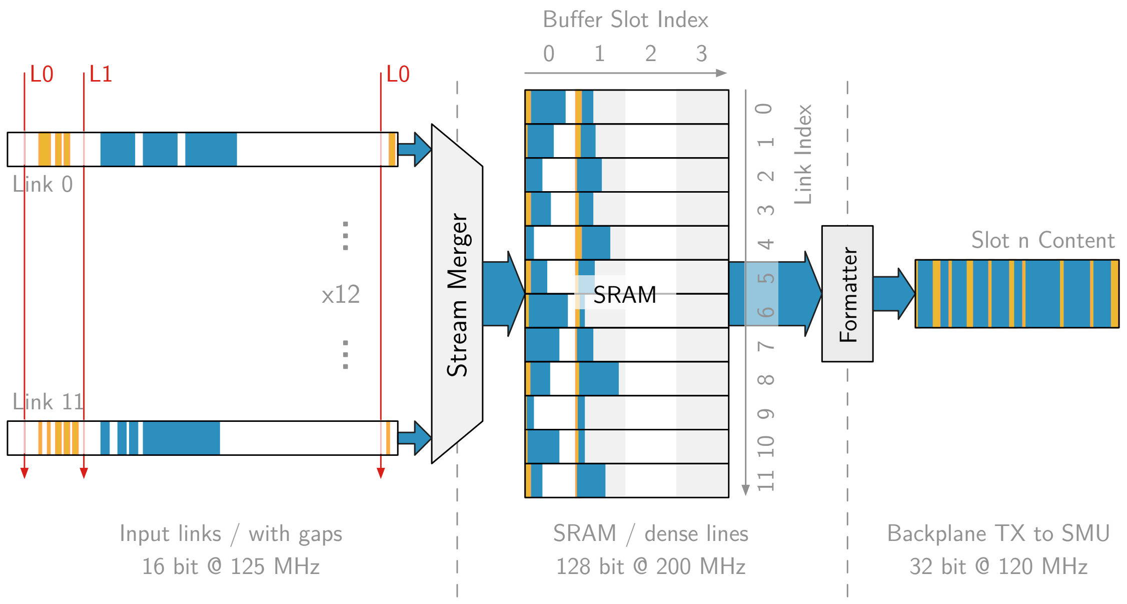


Figure 4.5: Reception and buffering of raw data and tracklets in different clock domains on a TMU (simplified). Alignment buffers compactify the link data stream for subsequent storage in the SRAM. The memory is partitioned into 12 independent link regions. In the current implementation with static layout, they provide buffering capacity for 4 events. The stack data is augmented with a header and transmitted to the SMU in ascending link order.

L1 (event raw data). With relative write pointers, the reject of an event at the L1 stage is implemented by simply clearing the link write pointers whilst retaining the current value of the buffer slot index. In case of a L1 accept, the notification that an active link has seen a valid raw data endmarker marks the end of the recording of data. Data lines for finalized links are retrieved from the alignment buffers and stored in the SRAM. Once the end of the data transmission is signaled, all final word link write positions are placed in the TMU event FIFOs alongside the current buffer index and the mask of active links. The recording of the event is finalized by incrementing the 2-bit buffer index and clearing the link write pointers. After that, the TMU event handling is immediately ready to receive the next event. For both L1A and L1R, the arrival of new data words is prohibited by the gating of incoming data with `expect_data`. In the actual implementation, the relative pointers are cleared with the start of each new trigger sequence, thus no explicit action is required for a L1R.

L2 Stage Rejects/Accepts and Data Retrieval On the SRAM read side, a FSM implements the retrieval of an event upon a L2A or discarding upon a L2R. The retriever operates on the entries stored in the TMU event FIFO.

In case of a L2A, the retriever fetches the relative link end positions, buffer index and link mask from FIFO. Absolute line read addresses for link data are calculated knowing the buffer offset and link memory region start address. Requests for all lines associated to active links are then posted to the read address FIFO of the SRAM controller in ascending link order. In preparation for the transport via the LVDS backplane, retrieved 128-bit lines are split into 32-bit words and stored in a dual-clock FIFO, which connects the retriever unit to the prioritizing backplane interface and handles the crossing into the readout clock domain. As the last line of each link was potentially padded, the link data is shaped to conclude with exactly two 32-bit raw data endmarkers. For a L2R, the retriever simply removes the corresponding write position entry from the TMU event FIFO to execute the reject of the event.

To avoid potential race conditions in the communication with the segment control on the SMU, and thus preserve buffer consistency, both removal and retrieval of an entry from the event FIFO are acknowledged. The segment controller waits for the acknowledgment from all TMUs before a new validated L2 accept or reject is distributed within the segment.

4.3 Intra-Segment Communication

As the readout interface is located/implemented on the SMU, event raw data held in the event buffers on the TMUs needs to be transferred within the segment in the process of local event building and readout. For communication within each segment, SMU and TMUs are connected by a custom LVDS backplane and a CompactPCI backplane.

The CompactPCI backplane is an off-the-shelf component, which is used to distribute power to the boards of the segment and facilitate the transmission of auxiliary control and status signals. The majority of high-speed intra-segment communication uses the LVDS backplane. It is a mostly passive assembly custom-designed to satisfy requirements for high bandwidth and low latency of the tracking and trigger application [41].

These requirements and the demand for resource sharing between trigger and readout application must be met by the firmware implementation. Presented in the following is a parallel LVDS backplane interface that employs a direct-clocking technique to implement highly reliable, prioritized backplane communication with low latency.

4.3.1 LVDS Backplane Physical Layer

To avoid unnecessary latency due to synchronization related to the intra-segment transmission, a direct-clocking technique is employed to implement LVDS backplane transmissions. The backplane physical layer, which encapsulates the I/O entity instances and the required calibration logic, and the backplane interface therefore operate on multiples of a common clock. A derived clock is available to drive the tracking entities on the TMUs and thus eliminate the need for backplane-related synchronization.

I/O Resources The availability of differential I/O pairs on the LVDS connectors of the SMU, which has to establish five point-to-point connections with the TMUs, limits the number of differential lines available to implement each connection [41]. For each connection, ten differential pairs are available in uplink[7] and three differential pairs in downlink direction. This asymmetric assignment of lines was chosen to retain parallel uplinks with high bandwidth for the transmission of tracking data and event data.

Clock-to-Data Phase Alignment Traditional approaches for the implementation of source-synchronous interfaces employed PLLs to insert a phase shift between source clock and capturing clock to move the rising edge of the capture clock into the center of the data valid window. The phase shift is determined by the designer to match the estimated propagation delays. Being fixed, it does not account for delay variations due to manufacturing or ambient conditions, which limits application of this approach to low-frequency interfaces only. Since the correct alignment of data to clock is a common problem, e. g., with memory interfaces at frequencies exceeding 200 MHz [86, 119], modern FPGAs possess configurable I/O delay elements.[8] These allow to implement fast interfaces with a direct-clocking technique, where source-synchronous data is delayed with respect to the capturing clock. The traces that constitute a given differential pair are length-matched on the node PCBs and on the LVDS backplane. However, the pairs that together form a parallel uplink or downlink are not. Thus, in order to implement direct-clocking and present a high-speed parallel interface for the LVDS backplane, a bit-wise calibration mechanism is required, which adjusts the individual IDELAY cells for an ideal clock-to-data alignment.

Figure 4.6 illustrates the physical layer architecture. The common clock, which serves as reference clock for the source-synchronous transmission, is supplied by the SMU and distributed via one differential downlink pair per TMU. Consequently, two differential downlink pairs per connection remain, sufficient to implement the transmission of control signals. On each of the TMUs, a dedicated Digital Clock Manager (DCM) generates 0.5x, 1x and 2x multiples of this reference clock with a constant phase relationship. The transmitting OSERDES and receiving ISERDES blocks are encapsulated by calibration entities. To not supply invalid data to adjacent units, they decouple the physical layer during the calibration process and contain entities that support or speed up the IDELAY calibration. Auxiliary signals, i. e., the calibration requests, are routed as bus-reserved signal lines on the CompactPCI backplane. To limit the number of auxiliary I/Os required on all nodes to two, the downlink calibration request is shared between all TMUs.[9] All nodes feature a port for receiving a calibration request and activating their request.

The clock supplied by the SMU is the 120 MHz data path clock. In order to avoid the complexities associated with the use of a fast clock in the FPGA fabric and achieve timing closure in a device of this generation, the design employs the dedicated fast I/O SERDES

[7]Uplink is defined as TMU $\rightarrow$ SMU, downlink denotes SMU $\rightarrow$ TMU.

[8]Xilinx introduced input delay elements with Virtex-4, output delay elements were introduced with Virtex-5.

[9]The connection scheme for the downlink calibration request is similar to the one used to communicate the lock status of the TMU backplane DCMs to the SMU, see Figure 5.11.

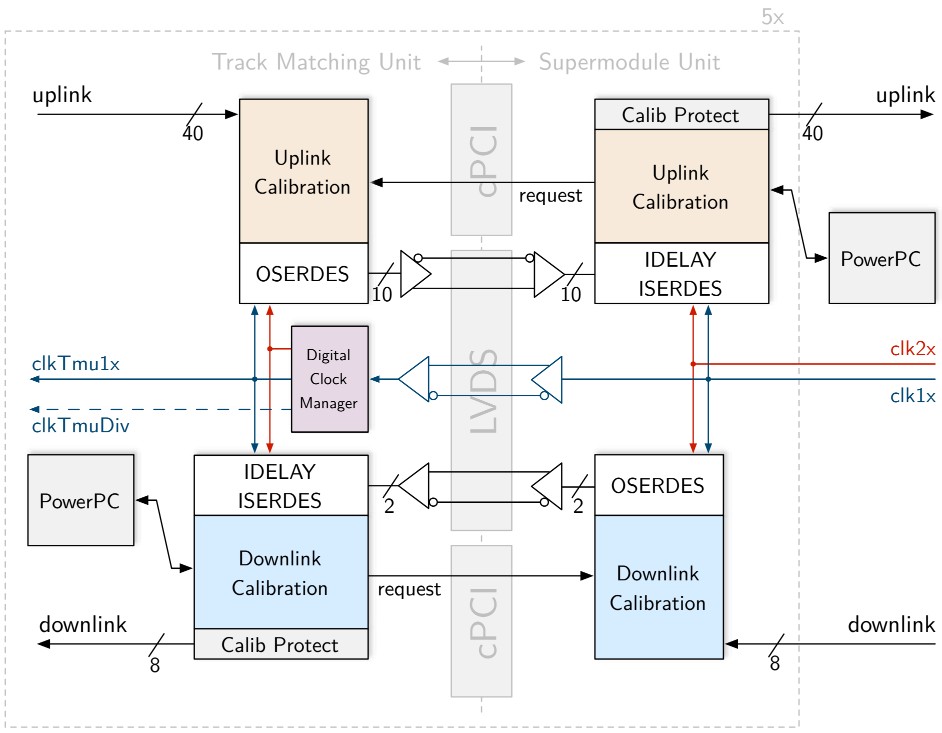


Figure 4.6: Physical layer of the uplink connection and downlink connection between an SMU and one TMU on the LVDS backplane. Dedicated calibration blocks, which encapsulate the fast serial I/O blocks, implement hardware to assist with the adjustment of the clock-to-data delay to allow for source-synchronous transmission via the LVDS backplane.

blocks in a 4:1 configuration and limits the use of the 240 MHz clock to these blocks only. The derived 60 MHz clock is available to drive the tracking on each TMU, while SRAM data retrieval logic and backplane interface are operated by the 120 MHz clock.

With I/O cells at 240 MHz DDR, the physical layer presents, for each connection, an 8-bit downlink and 40-bit uplink interface[10] at 120 MHz SDR to the adjacent entities on SMU and TMU.

Dynamic Input Delay Calibration The calibration mechanism determines an ideal setting of the IDELAY elements, such that transitions on the data lines occur in-between the sampling clock edges with maximum margins to satisfy setup and hold time requirements.

To do that, the mechanism applies a training pattern and sweeps the delay settings to shift data with respect to the clock. For each individual LVDS pair,[11] both the start position s and the width w of the window, in which a valid training pattern is received, is measured

[10] Two differential pairs are available for data in downlink direction, the third pair is used for the clock. Ten pairs constitute the physical uplink.
[11] In the following, the term line is used as synonym for one differential pair.

with large statistics. The best IDELAY tap setting for each line is then calculated as $s_i + w_i/2$, which places the data transition in-between two sampling clock edges. The calculated position might be off center with respect to the clock, but is at an ideal position regarding a maximum margin for setup and hold times.[12]

The calibration is implemented as hardware/software co-design employing one of the embedded PPC cores and support logic in the LVDS physical layer. The core calibration procedure is very similar for both uplink and downlink connections. Therefore care was taken to use a common calibration software code base and choose the amount of auxiliary logic such that size and execution speed demands of SMU as well as TMU are met. The calibration for the lines input to a node is initiated by the node's high-level PPC, typically as part of the start-up procedure or DCS configuration. To start the calibration of a downlink or uplink connection, the PPC on the receiving side activates the calibration `request` signal. Upon detecting the request, the sender node commences the transmission of a uniquely identifiable calibration training pattern.[13]

IDELAYs on the Virtex-4 allow to delay with a granularity of 75 ps in 64 steps (taps), and thus enable a maximum data retardation of 4.8 ns using taps alone. The primitive also gives control of the reordering of received serial data to word boundaries on the parallel output via its bitslip submodule. One reordering of the parallel output corresponds to an effective shift by one transmitted bit, i. e., 2.08 ns at 240 MHz DDR.

Figure 4.7 depicts a flow chart of the sequential calibration of all five uplink connections as executed by the high-level PPC of a SMU. The dashed area therein denotes the actions required for the calibration of an individual connection, such as executed by the TMUs for the downlink calibration. The calibrate one connection, the software sweeps the tap range beginning with tap 0. For each tap, the software initiates a measurement, i. e., reference pattern and received pattern are compared in parallel for all lines constituting a given uplink or downlink connection. If a promising window candidate[14] is found, the bitslip is kept constant and the taps are incremented to find windows of successful measurements for the individual lines. In case no window candidate is found, the tap setting is restored and the measurement is repeated for all possible bitslips before advancing to the next tap. In case a sufficiently wide window with successful measurements is found for every line for a given bitslip, the optimum delay settings for each line are calculated and applied. The calibration is then finished. In case the window is not good, i. e., the window for at least one line is not wide enough, the bitslip is marked as bad and is not considered anymore in the remaining search. If the board is present, eventually a sufficiently wide window is found by the calibration and the ideal tap settings can be applied. Figure 4.8 exemplarily visualizes the measured window positions for all uplink lines of a segment (here Segment 0) for different bitslip settings.

[12]Setup and hold values for the employed configuration are not available. However, values for a similar configuration of the ISERDES blocks given in [56] are $t_{\text{setup}} = 0.87$ ns and $t_{\text{hold}} = 0.68$ ns.

[13]The default pattern used for calibration of the de-/serializers in the employed 4:1 mode is `0x2`.

[14]A window candidate is characterized by a successful measurement for at least one line at the current tap and bitslip setting, and, if tap $\neq 0$, an unsuccessful measurement on all lines in the previous tap for the given bitslip.

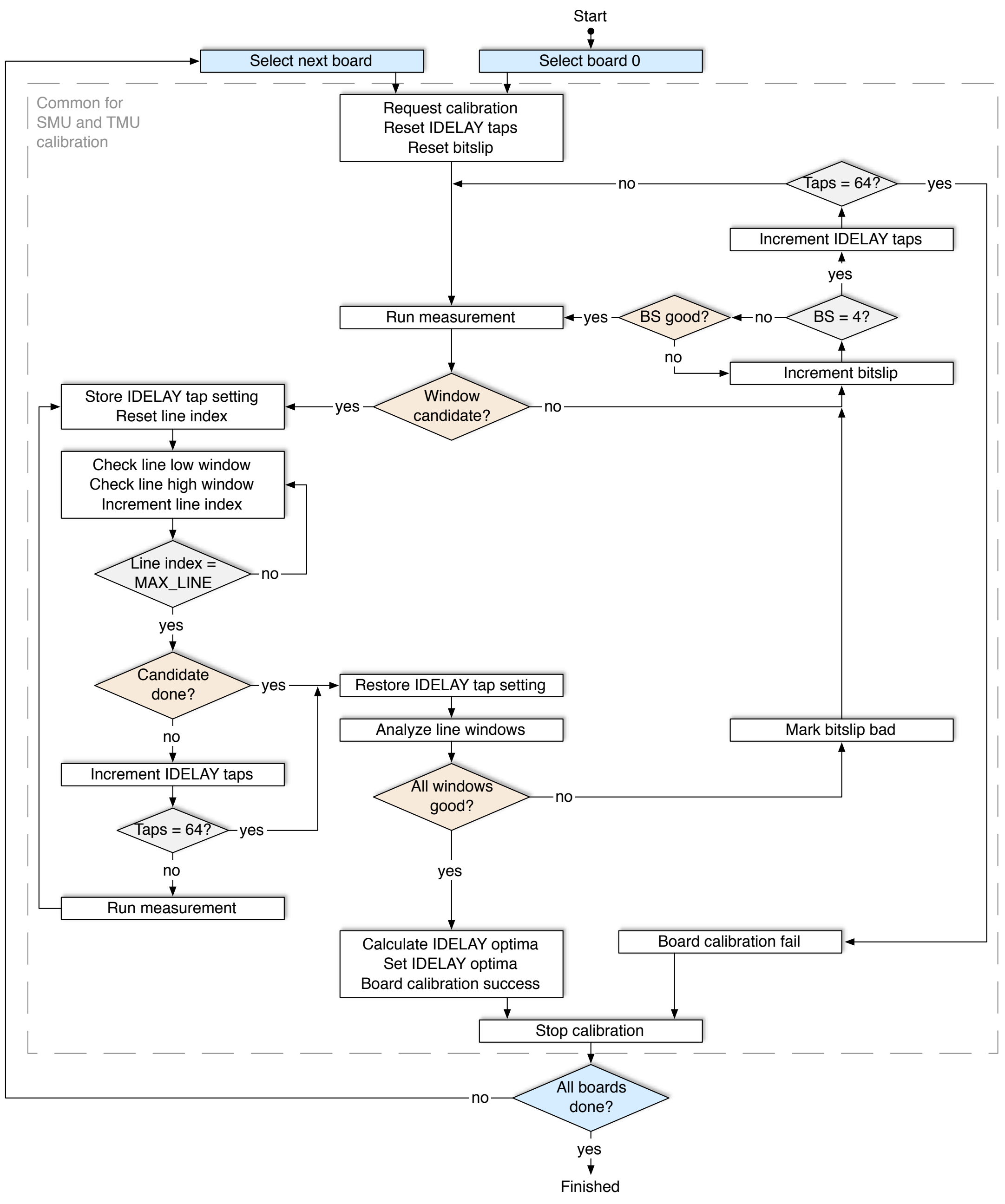


Figure 4.7: Flow chart depicting the core functionality of the IDELAY uplink calibration software executed by the high-level PPC. The dashed area marks the common actions required to calibrate one connection, which is executed by the SMU multiple times to calibrate all uplinks and by a TMUs for calibration of its downlink. The software sweeps over the available tap/bitslip parameter range and identifies the first tap/bitslip combination that ensures correct data-to-clock alignment.

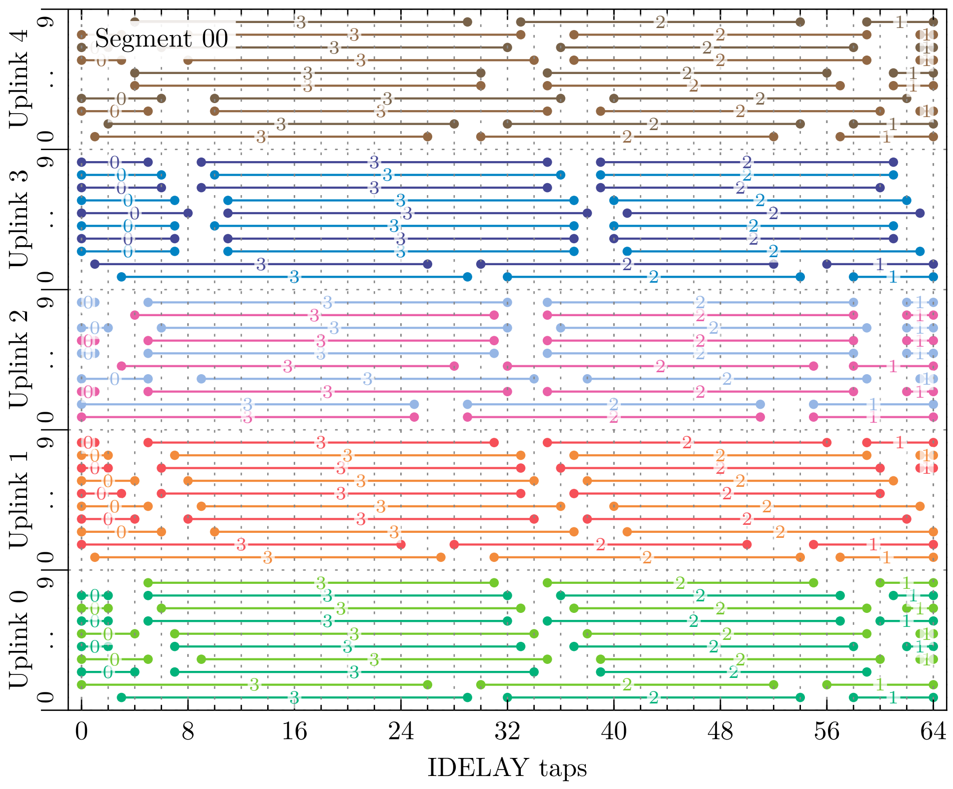


Figure 4.8: Exemplary scan over the IDELAY range for all 10 lines of the 5 uplink connections arriving at the SMU of Segment 0. The lines mark the IDELAY tap windows found by the calibration algorithm, the numbers on the line mark the corresponding bitslip. Due to the large differences in window positions, e. g., for lines 1 and 3, an individual calibration of each line is required.

The PPC controls IDELAY taps and bitslip via the `idly_ce`, `idly_rst` and `bitslip_ce` signals. The 6-bit IDELAY tap counters, of which 50 are needed on the SMU and two on each TMU, are realized in software. The trace length variations within lines constituting one connection are less than 20 cm, thus they can be compensated using the tap delay line only. Consequently, one common bitslip setting for all lines of an uplink or downlink connection can be found and one 3-bit bitslip counter per connection is implemented in hardware.[15]

An individual measurement for a given tap/bitslip combination is started by the PPC via `cycle_start` and `cycle_complete` indicates the availability of the result. A measurement cycle is considered as match (or successful) if received and reference pattern match for at least 1024 consecutive cycles. As tradeoff between calibration speed and resource usage, the

[15]Contrary to the delay line, the bitslip module is missing a dedicated reset port. It cannot be reset from software, it is reset only with the ISERDES reset. An implementation of the bitslip counter in hardware therefore greatly simplifies the synchronization of the bitslip counter and actual setting of the bitslip submodule.

hardware features measurements of all lines of an uplink or downlink connection in parallel. The ten-fold replication of the corresponding counters and comparators on the SMU is acceptable, since the small amount of extra resources gives the benefit of a significant calibration speed-up and simpler calibration software (and resource usage is not a pressing issue on the SMU).

Taking into account jitter is crucial when searching for the matching window, because it can distort measurements of the window edge positions and, as a consequence, lead to a suboptimal IDELAY tap setting for the given line. In this application jitter is countered, to a certain degree, at the hardware level by the requirement of n consecutive match cycles for a successful measurement. Furthermore, when determining the width of the match window, the software takes into account only consecutive successful measurement in adjacent taps. In case an unsuccessful measurement cycle is encountered, the match window is shrunk accordingly.

Figure 4.9 exemplary summarizes the result of the calibration of LVDS backplane connections for Segment 15 (colored) and Segment 16 (gray). Shown are the average values of calculated IDELAY tap settings, the sizes of the underlying tap windows and the corresponding bitslips that were found ideal in 250 calibration cycles. The errors in the calculation of ideal tap values are small, which indicates that the calibration algorithm is stable and that the jitter issues are properly addressed. For each line, the algorithm reliably finds the lowest working delay setting (and the associated bitslip). Although the effect is very small, this is favorable when aiming for low latency in the trigger application and also when taking into account signal degradation that is added within each tap of the delay line [43].

Although the hardware of the two shown segments is identical with respect to PCB layout of the involved nodes and backplane, corresponding lines on the two segments nevertheless show quite some variation in tap settings, e. g., line 1 of uplink 0. The differences may be attributed to a number of factors, among them variations in capacitances, resistances and operating temperatures. Considering the PCB trace lengths differences, the node-wise variations and the large number of involved nodes (108), an approach using pre-determined delays would have required a configuration with a complex static delay map. The dynamic phase alignment approach implemented not only ensures at least an equivalent transmission quality, but also replaces the need to manage and apply such configuration with a PPC startup procedure, which fits in well with the overall effort to minimize GTU configuration items. In the test setup with one full segment, a bit-wise checking of data produced by the link-level pattern generators was performed on an LDC, see Section 6.2. No errors were observed in >150 TB of data transported through a segment to the readout system, which indicates a bit error rate of $<10^{-15}$ for the data handling of a full segment.

TMU Auto-Detection For the supermodules 13, 14 and 15, the central stack is removed in order to not increase the material budget in front of selected PHOS modules. Consequently, the corresponding GTU segments miss the central TMUs, which are removed and serve as cold spare.

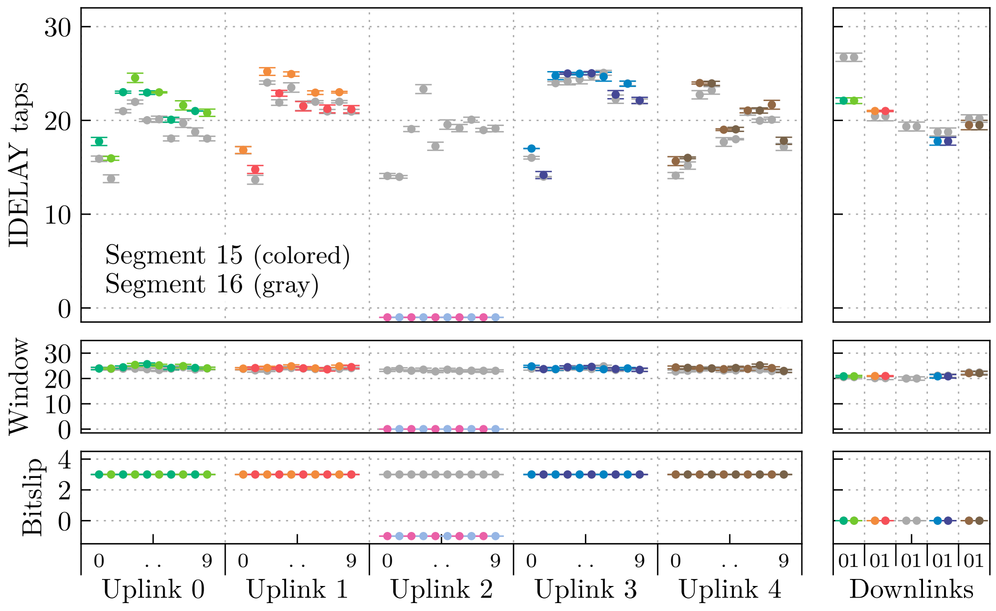


Figure 4.9: Average optimum IDELAY tap settings, average tap window sizes and associated bitslip measured in 250 calibration cycles for all lines of a segment. Segment 15, shown here in colors, misses the center TMU (PHOS hole). This is automatically detected by the calibration mechanism, which sets the taps and bitslip setting to a negative value. Segment 16, in gray, is fully equipped.

The calibration mechanism allows to automatically detect the presence of TMUs in a segment. Segment 15 in Figure 4.9 misses TMU2, which is reliably detected and indicated by the calibration as tap setting −1 and window size 0. The automatic TMU detection is exploited to keep the number of configuration parameters low, segments without central stack will set the TMU mask accordingly, thus excluding this TMU from readout.[16]

4.3.2 LVDS Backplane Interface

For the transmission of trigger, status, and event data in uplink direction and control data in downlink direction, a sensible arbitration of the backplane resources is required. The backplane interface presented here solves this by providing prioritizing access to the backplane.

TMU-SMU Uplink Data output by the global tracking entities on the TMUs requires immediate transmission to the trigger stage on the SMU. Since the tracking entities produce result data in multiples of 32 bit, and also event raw data is stored in multiples of 32 bit, the 40-bit uplink provided by the physical layer is divided into 32 bit for payload plus 8 bit for status and type information, as depicted in Figure 4.10.

[16]A manual override for testing purposes is available.

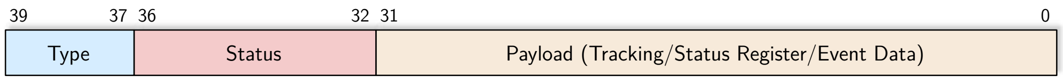


Figure 4.10: Format used during transmission via the LVDS backplane in uplink direction (120 MHz SDR).

Initial implementations of the tracking produced a dedicated 32-bit decision word used for the trigger calculation, followed by 32-bit words with details about the reconstructed tracks. With 32-bit words output from the tracking at 60 MHz and a maximum readout bandwidth of 1.6 Gbit/s (in Run1), the 3.84 Gbit/s available for payload may be simply multiplexed between trigger and event data transmission without restricting the performance. However, in the course of the Run1 trigger development, the tracking output format was changed to produce 64-bit words containing information for sophisticated triggering, see Figure 4.11. For each found track, a track word plus an extended track word is produced. With regard to the fact that the number of tracks is very limited,[17] and therefore a lot of bandwidth is wasted when using a time-division multiplexing approach, a design is chosen for the backplane interface that strictly prioritizes tracking-related data over status information and event data.

Table 4.2 lists the different payloads, which are identified via the `type` field, and their transmission priorities. A net bandwidth of 3.84 Gbit/s is instantly available for the transmission of tracking data and event data when required. An ongoing transmission of event data is potentially interrupted by the transmission of tracking words for brief periods of time, but immediately resumes at full bandwidth. The `status` field allows to transmit 5 bit of status information irrespective of other activities with deterministic latency and is used, e. g., for the L2 handshake signal. A second mechanism extends the possibilities for the transmission of status information. A request to transmit a 32-bit status register, whose content is supplied by the TMU tracking or event handling entities, as payload of type `status` is generated internally whenever any bit in the register changes its value. The mechanism is suitable to transmit non-critical information or values that are guaranteed not to change in the time window when tracking data is sent, e. g., `tmu_full` or `tmu_fee_rx_done`, as their transmission may be delayed by the transmission of tracking words.

SMU-TMU Downlink In downlink direction, few control signals and the fill state of the SMU data input FIFOs have to be transmitted. For that, the physical layer offers 8 bit at 120 MHz, which allows for a straightforward implementation with deterministic latency. The control signals, which are common for all TMUs of the segment, emerge at the multi-event segment controller and include `expect_data` plus the validated triggers

[17] The number of reconstructed GTU tracks in central Pb-Pb events at $\sqrt{s_{NN}} = 2.76$ TeV is in the range of 50–100 per supermodule, i. e., about 10–20 tracks per TMU. With a 64-bit track word plus 64-bit extended track word per track, this corresponds to about 160–320 B of track-related data on each uplink per L0. For a detailed analysis of the tracking see [99].

`l1_accept`, `l2_accept` and `l2_reject`. The signals are masked with the mask of active TMUs before transmission.

4.4 Communication between SMUs and TGU

The TGU acts as concentrator node for the `busy` signal communicated to the CTP and as global trigger calculation node. The connection between the SMUs and top-level TGU is established via Cat 6e cables, which each connect to the 8P8C connector on the segment's LVDS backplane and on the dedicated TGU backplane. Each of the 18 cables, which are length-matched to 5 m, provide four differential pairs to transmit data between the two nodes.

Physical layer implementation and the calibration of the clock-to-data delay for the SMU-TGU connections is modeled after that of the LVDS backplane physical layer presented in Section 4.3.1. The 40 MHz LHC clock, which is reconstructed on both ends by the TTCrx on the DCS board, serves as base clock for the connections at 240 MHz DDR. Table 4.3 summarizes the available transmission lines per connection and their usage. For the slow-changing busy status, transmission as a level signal is sufficient. For details on the SMU-TGU link, especially the transmission of trigger-related information and transmission reliability, see [99].

4.5 Collection of Auxiliary Event Data

Tracking, trigger and raw data handling on the GTU provide valuable real-time information. This data is generated by a number of entities within a segment, e. g., by link data shapers at the TMU inputs and trigger entities on the SMU, and is valid only at certain points in time of a trigger sequence. Integrated into the Data Quality Monitoring (DQM) system, this data aids, e. g., in the online supervision of the current detector behavior. As it is specific to a given event and used both online and offline during the analysis, this additional

Type	Payload	Priority
`"000"`	Invalid data	-
`"001"`	Tracking/Trigger	high
`"010"`	Status	medium
`"011"`	Stack event data	low
`"100"`	Stack event header	low
`"101"`	Stack event trailer	low
`"110"` / `"111"`	Reserved	-

Table 4.2: Payload types and priorities for the LVDS backplane transmission in uplink direction.

data is read out as part of the event fragment. The following sections present an overview of the types of data collected and the required buffering structures to store the data and attribute it to the correct event during readout.

4.5.1 Link-level Information

The mask that indicates the currently active detector half-chambers is acquired on an event basis by the TMUs by reading the monitored SFP RX power. This offers the possibility to remove chambers, for example those exhibiting erratic behavior, from an ongoing run by switching them off. This `link_mask` is not only required to control the endmarker detection during event buffering, but also offline by the reader software. Further link-level information originating at the TMU input stages includes the link data shaper flags to indicate correctly formatted link data, information about the data source and calculated CRCs.

As the amount of data is fixed, it is stored in a FIFO that receives control signals identical to those of the TMU event FIFO holding the link data end positions. Generated in the L0-L1 interval, it has to be stored for each event accepted at the L1 stage and must be discarded at a L2R. In case of a L2A, the information is retrieved and placed in the stack header assembled by each TMU.

4.5.2 Tracking Information

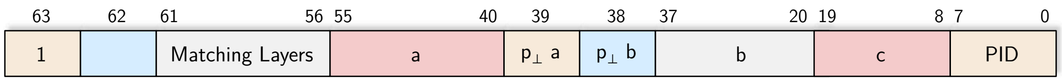
Results from the tracking are collected to enable offline verification of the tracking algorithm and recalculation of the trigger. Furthermore, the reconstructed high-$p_\perp$ TRD tracks are used in HLT monitoring components and are available for further matching to other detectors.

An instance of the tracking entity runs on each TMU. Shown in Figure 4.11 are the three types of result words output by the latest version of the tracking in order of instancy. Tracking words form the basis for the successive trigger algorithms, thus they are generated first and handled with highest priority. For each track found, such a tracking word contains matching detector layers, $p_\perp$ threshold flags, fit parameters and PID. A 'tracking done' marker is produced once per event when the corresponding tracking instance has concluded reconstruction of all tracks. The done marker contains measurements of the tracklet arrival

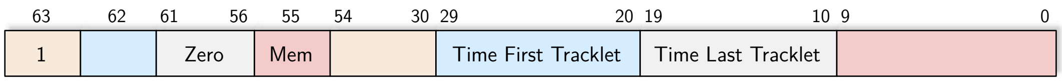
Pair	Transmitted	Type	Direction	Encoding
0	Reserved	-	up	Hamming
1	Trigger information	Message	up	Hamming
2	Segment busy	Level	up	-
3	Calibration request	-	down	-

Table 4.3: Transmission lines between the SMU of a segment and the TGU.

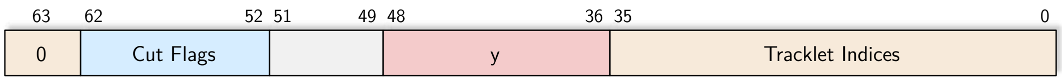
times and its arrival on the SMU is used to evaluate whether the tracking was capable of processing all tracklets to potential tracks in time, i. e., sufficiently long before the L1 decision time. Finally, an extended tracking word is produced for each reconstructed track. Its content is not required for online triggering, but provides the data required to re-simulate and verify the tracking offline.



(a) Tracking word



(b) Tracking done marker



(c) Extended tracking word

Figure 4.11: Formats of the data words produced by the tracking. Tracking words contain information utilized by the trigger algorithms, while tracking done marker and extended tracking word contain information needed for the offline analysis of the tracking and its performance. Empty fields are currently reserved.

Tracking Result Buffers Contrary to the fixed amount of link-level information, the amount of memory required to store the tracking results is not known up-front, but varies with the amount of high-$p_\perp$ tracks in the event. The SMUs implement a set of five dedicated tracking buffers. Figure 4.12 shows the location of the tracking buffers and their associated readout unit in the SMU data path between the last tracking stage and the event dispatcher. The buffers support two clock domains. They offer a 64-bit write interface in the tracking clock domain and a 32-bit read interface in the readout domain. Each buffer encapsulates dual-port BRAMs along with the required write and read logic. The number of BRAMs used is configurable to allow for easy extension, if required. For each event, a record consisting of a mandatory 32-bit index word, see [42], plus a variable number of 64-bit tracking output words is stored.

Recording Window The tracking buffers define a window in which tracking output words are accepted. The window opens with a validated L0 issued by the segment control logic. As the generation of track words is generally finished well before the arrival of the L1 trigger, the recording window could be closed with the L1 decision window. However, for a L1A it is beneficial to extend the recording until the corresponding TMU signals the end of the FEE data transmission. This grants additional time to the tracking for the shipping of the low-priority extended track words. Furthermore, in cases where the tracking does not complete before the L1,[18] potentially reconstructed tracks and tracklet timing measurements are still recorded. This provides valuable offline debug information as to why the tracking did not complete.

To ensure efficient memory usage and flexibility, the buffers are implemented as circular buffers. A maximum allowed number of tracking output words per event record[19] is defined. This allows to calculate a `full` signal to contribute to the segment `busy`, and therefore effectively implements buffering with a variable number of slots. This threshold is also used to implement a protection against potential buffer corruption and the subsequent stall of the segment operation due to an excess of produced tracking words. Once the threshold is reached, no more words are stored for a given event and an overflow flag is set in the index word.

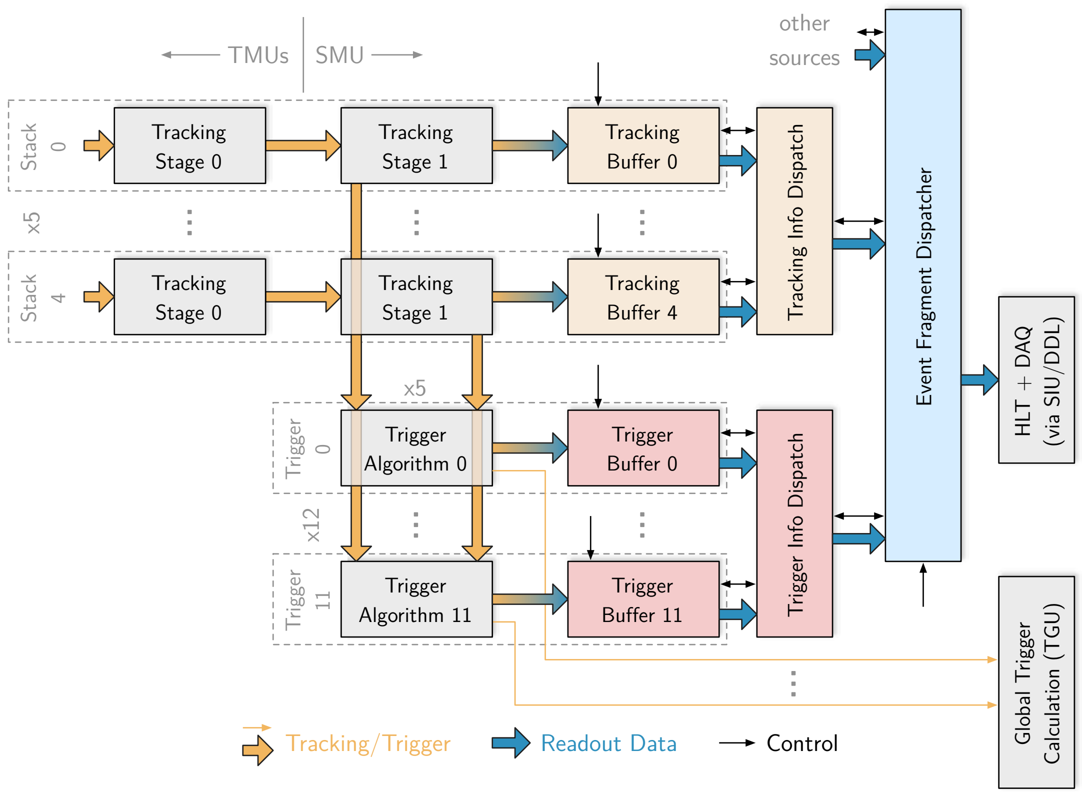


Figure 4.12: Collection of information related to the trigger application at various calculation stages. The data that is relevant to the offline trigger verification and performance analysis is collected in dedicated multi-event capable buffers and inserted into the event fragment during local event building.

[18]The tracking algorithm may not finish within the L0-L1 timeframe for a variety of reasons: an exceptionally large number of tracklets in a high-multiplicity event, late arrival of tracklets due to detector configuration/faulty chamber, etc. See [99].

[19]The current threshold is 60 words, i. e., a factor of 2 compared to the number of words observed in a central Pb-Pb event at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$.

Accepts/Rejects Analog to the event buffering on the TMU, a `flush` signal, corresponding to a reject at the L1 stage, resets the write pointers to their previously stored sequence start value to discard already recorded tracking words. In case of a L1A, the arrival of the `rx_done` signal from the associated TMU finalizes the record. Before positioning the write pointer to the beginning of the next event, the write controller updates the index word with record size information, flags and the duration of the overall tracking time measured on the SMU.

Control signals reflecting the L2 stage triggers are received by the tracking dispatch unit. In case of a L2R, it forwards the reject command to all tracking buffers associated with enabled stacks. The reject of a record is executed within a single cycle by repositioning the read pointer to the address of the next event. If an event has been accepted for readout with a L2A, at a given point during the local event building the event fragment dispatcher passes control to the tracking dispatcher. It then waits until it has finished fetching the individual tracking records from the buffers and moving them into the FIFO interfacing the SIU. Control is then transferred back to the event fragment dispatcher.

4.5.3 Trigger Information

The SMU offers the possibility to record intermediate or final results of the different trigger algorithms as part of the event fragments. This is especially helpful during the development and verification of new trigger algorithms.

For each of the up to 12 supported trigger algorithms, a trigger buffer instance is foreseen. Trigger buffers are modeled after the tracking buffers and provide identical functionality with respect to accepts and rejects at the various trigger stages. However, it is assumed that the amount of data produced by a trigger algorithm is small and its format fixed. Located between algorithm and event dispatcher, see Figure 4.12, they present a 32-bit interface in both the trigger and readout clock domain. It is up to the algorithm to define the maximum record size and guarantee the availability of the mandatory index word with record size information alongside the record content. The fixed record format leads to a greatly simplified write logic compared to the tracking buffers.

In order not to excessively bloat the produced event fragment, several options are available to control the readout of trigger headers. For each algorithm, the operator may select whether to include trigger headers in the readout

- never,
- when the trigger is enabled, or
- only when the trigger is enabled and has fired.

4.5.4 Cyclic Redundancy Checks on Event Data

If a bit in an event fragment is altered, chances are that it goes unnoticed during the offline analysis if, e. g., only the lower bits of an ADC value are affected. Even if the error is detected, e. g., if one of the half-chamber headers are affected, it is still unclear what caused the error and where in the readout chain it occurred — a common problem in large, complex readout systems.

Main purpose of the CRC calculation in the GTU is to detect errors related to the data transport within a GTU segment, but also to distinguish between errors introduced by the FEE and those introduced by the GTU. The calculation entities are placed in a way that makes it possible to pin down the error's origin to distinct regions of the readout tree. Figure 4.13 shows the different calculation stages and their location in the data path of a segment.

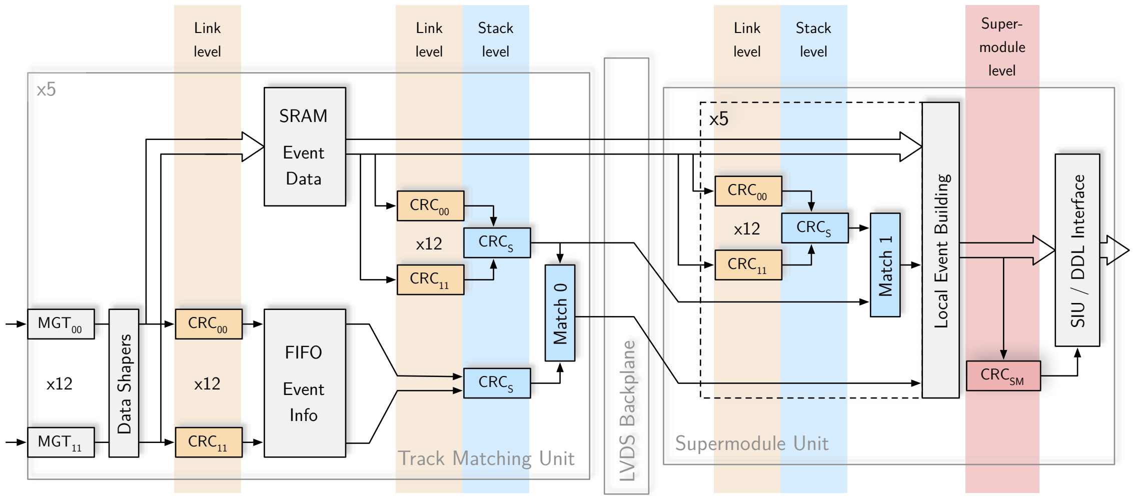


Figure 4.13: Generation of CRC checksums and match flags at various stages in the data path of a GTU segment. The match flags produced by the comparison stages help to narrow down the location of a potentially malfunctioning entity.

Stack-wise CRC As reference, 12 link-level CRCs are calculated on data received from the FEE. The calculation unit is located directly behind the input data shapers, but before the alignment buffers and SRAM. Raw data endmarkers potentially introduced by the input shapers to emend the event format also enter the CRC calculation, which enables comparisons to the reference CRCs even in the case of malfunctioning chambers. The calculation uses the CRC-16-ANSI polynomial.[20] Link-level reference CRCs are buffered in the TMU event FIFO along with other link-related data, e. g., link shaper flags. The 16-bit link-level reference CRCs are combined to a reference CRC representing the full stack in case the event is read out. This reference stack CRC is used in subsequent comparisons.

[20]CRC-16-ANSI: $g(x) = $ 0x8005

The reference stack CRC is compared against recalculated CRCs at two locations in the data path. A stage after the SRAM on each TMU calculates first link-level CRCs, then stack CRC and compares it to the reference. Another stage on the SMU calculates and compares, for each TMU, the recalculated CRCs to the reference after the intra-segment transmission over the LVDS backplane. The two sets of accompanying match flags, each containing the five comparison results of stack CRCs, reflect the data integrity at these key locations within a segment's data path.

Supermodule CRC On the SMU, the common IEEE802.3 CRC-32[21] is used to calculate a supermodule-level CRC on the full event fragment excluding the event trailer. It allows to detect errors related to the SIU interface in the FPGA fabric, the SIU add-on card (or SIU core in Run2) and the DDL transmission.

The calculated CRCs and match flags, when appended to the event data during readout, allow to pinpoint errors to three distinct regions of the data path: The TMU event handling including the external SRAM, the transmission from TMU via the trigger-prioritizing LVDS backplane interface to the SMU input buffers, and building of the local event fragment. Modern CPUs facilitate efficient computation of CRCs using, e. g., dedicated vector instructions combined with a parallel folding technique [57]. Employed on a monitoring computer, the recalculated CRCs provide a fast and reliable tool to locate problems with the segment's data transport in quasi real time.

4.6 Local Event Building and Readout

The local event building assembles the supermodule event fragments, which are later combined, using the event ID in the fragment's header, to global ALICE events by the DAQ event building network. The event fragment dispatcher in charge of building the fragment and the readout interface are located on the SMU.

4.6.1 Interface to the Data Acquisition (Run1)

A segment's link to the DAQ/HLT is established via the SIU, which, together with the optical medium and the DIU, constitutes the DDL. In Run1, the SIU is a daughter board in PCI mezzanine form factor mounted on the back of the SMU. Its transceiver operates at a line rate of 2.125 Gbit/s (see Section 4.7 for the upgraded readout interface in Run2).

The source-synchronous communication with the SIU card, specified in [102] and [101], is encapsulated by an interface core. The core is detached from the data path by using a 512 word deep dual-clock SIU output FIFO. The core clock, which is also provided to the SIU card, is supplied by a dedicated DCM and can be adjusted to meet the interface

[21]CRC-32-IEEE [52]: $g(x) = \text{0x04C11DB7}$

frequency requirements.[22] Words placed in the output buffer are then transmitted by the core subject to the DDL busy status and potentially required transmission gaps. Careful adjustment of the clock's frequency and phase allowed to match the large SIU interface setup time requirements of 8 ns at a frequency of 50 MHz. With a 32-bit data interface, the available output bandwidth is 1.6 Gbit/s. The reader may refer to [76] for a more detailed description of the core and interface timing.

The interface core exposes `siu_channel_active`, reflecting the DDL's basic readiness to accept data, and `siu_buffer_ready`, a flow control signal, to the adjacent entities. The core generates and sends the required end-of-block transmission status word once the last data word of the event fragment is transmitted. This last word is indicated by an `eoe` flag, which is propagated through the buffer FIFO alongside the event fragment data.

4.6.2 Generation of the Event Fragment

Each event fragment is assembled from the buffered event data of the supermodule's half-chambers and the auxiliary data generated in the GTU segment. The signal to commence the readout of an event is given by the MEB control logic, which had previously fetched entries from the SMU event info and L2 info buffer. It is distributed to the event fragment dispatcher on the SMU and to the readout units of the TMUs in the segment.

Fragment Structure With the experiment in constant development over its runtime, certain changes to the data format are inevitable. When changing the data format, it is mandatory that with any upgrade any previous version is still readable by the offline analysis. The GTU structures the event data originating at the supermodule and the collected auxiliary data using a series of headers. Mandatory part of these headers is a header index word containing the header's size and a version field. In combination they provide sufficient flexibility for format updates, e. g., the introduction of new header blocks, and offer a way to retain backward compatibility. A brief overview of the event fragment structure is given in Appendix B, the complete format specification can be found in [42].

Depending on the readout scenario, certain data, especially auxiliary data, might not be needed. For example, full readout of the trigger information even in cases where the trigger did not fire is essential for trigger development, but can be omitted once the trigger is characterized, verified and in production. To reduce excess data volume and long term storage cost, the GTU makes the content of the event fragment highly configurable. The availability of certain blocks of information, e. g., tracking headers, is indicated by flags in the mandatory supermodule header.

[22] The maximum allowed operating frequency of the SIU interface changed over several versions of [101]. In the latest version it is specified as 50 MHz.

Stack Readout and Transfer to SMU

With the corresponding signal sent by the segment controller, the TMU readout units fetch an entry from the TMU event FIFO. Knowing the link end positions and buffer slot index, they post SRAM read requests for data of active links in ascending link order. The retrieved 128-bit lines are split into 32-bit words for transfer via the backplane. Simultaneously, each readout unit prepares the stack header using captured configuration data, e. g., the mask of active links, and stack-related error information retrieved from the event information FIFOs. Stack header words are analyzed on the SMU in order to summarize the segment status and set the appropriate flag in the `status_and_error` field of the main event fragment header, the Common Data Header (CDH). Each stack header is followed by the retrieved link data content and the stack trailer, which contains the stack-level CRC. Figure 4.14 shows an example of the transmission of event data over the backplane, in this case interrupted by higher-prioritized tracking words. Input FIFOs on the SMU, which receive the raw data from the stacks, compensate for short interruptions caused by the transmission of tracking words. They also provide a flow control signal, `smu_ready`, which is raised sufficiently in advance to account for the backplane latency.

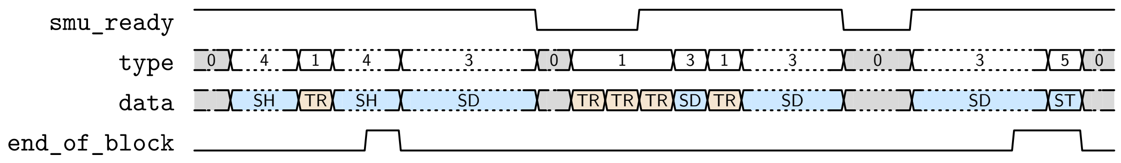


Figure 4.14: Stack data transmission (blue) from a TMU to SMU. Prioritized track-related data (TR, yellow) potentially interrupts the ongoing transmission, which is organized in three blocks: Stack header (SH), stack data (SD) and stack trailer (ST). The final word of each block is marked by the `end_of_block` signal.

Local Event Building

When the start signal is received, the SMU event dispatcher commences the assembly of the event fragment. Dispatcher and data path are designed, in Run1, to deliver a dense stream of data words to the SIU at a rate of 60 MHz.

Common Data Header Each fragment transmitted to the ALICE DAQ must begin with the CDH [45] as first data block. In Run1, it comprises eight 32-bit words that contain the CDH version, the event IDs and trigger information distributed by the CTP. The segment control assembles the corresponding CDH words from the contents of the L1 and L2A message and adds the local event ID. The header's `status_and_error` field is updated with information from the trigger receiver, the segment controller, and, after inspection of the arriving relevant stack header words, a contribution from the TMUs.

Supermodule Header The CDH is followed by the supermodule header, which defines the internal structure of the event fragment. It contains all configuration information required by the offline reader software to successfully parse the event fragment. Apart from the header version and size, main contents are the mask of active detector stacks and availability flags for trigger headers, tracking headers and the event trailer.

Tracking and Trigger Headers Tracking headers follow the transmission of the supermodule header. The event dispatcher passes control to the tracking info dispatcher, which in turn facilitates the readout of tracking headers associated with enabled detector stacks in ascending order, if their readout is enabled. If not, the readout of the tracking information is skipped and control is returned immediately to the event dispatcher.

Readout of the trigger headers follows next in a similar fashion. The trigger info dispatcher evaluates trigger readout mode, trigger mask and firing flags to decide whether a certain header is to be written to the SIU output buffer.

Stack Headers and Stack Data Readout of the stack headers and the stack data follows next. The dispatcher first fetches the individual headers of all enabled stacks from the SMU input buffers in ascending stack order. Next, the stack data is fetched from the input buffers until the last data word is seen and placed in the SIU output FIFO. Again, stack data is fetched in ascending stack order. The stack trailer with CRC information is not transmitted with the data, but is available at the output register of the SMU input FIFO once all stack data has been transferred.

The latest SMU firmware version appends a marker[23] to clearly separate data content of different detector stacks. This renders possible the processing of the remaining event in case the data of one link is corrupted by, e. g., a faulty half-chamber header, and simplifies debugging.

Event Trailer The calculated supermodule-level CRC, the five stack-level CRCs and the match flags are appended to the event as trailer, if the trailer readout is enabled. Once the last word is placed in the SIU output buffer, the dispatcher notifies the segment controller of the successful readout, which in turn then processes the next L2.

4.7 Upgrade of the TRD Readout for Run2

After the successful Run1 period with Pb-Pb collisions at $\sqrt{s_{NN}} = 2.76\,\text{TeV}$ and an instantaneous luminosity exceeding $10^{26}\,\text{cm}^{-2}\,\text{s}^{-1}$, ALICE, at the time of writing, is preparing for the next data taking period. Run2 will see collision energies of $\sqrt{s_{NN}} = 5.5\,\text{TeV}$ and an

[23]A stack separation marker is a 64-bit marker, split into two successive 32-bit words, `0xe0d10000` and `0xe0d10000`. The words are chosen to be not producible by the detector FEE [78].

increased instantaneous luminosity of up to $4 \cdot 10^{27}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, corresponding to interaction rates in the range of 8–30 kHz [123, 72].

Readout Bandwidth Increase To benefit from the increase in luminosity, ALICE plans to operate at higher data taking rates in Run2. In the most common running scenario one readout partition is formed by Time Projection Chamber (TPC), TRD and a number of other detectors. The readout rate achievable is governed by the performance of the slowest detector. In Run1, the maximum readout rate of the TPC for central Pb-Pb collision was 250 Hz, mainly limited by the bandwidth of the GTL buses between TPC Readout Control Unit (RCU) and Front-End Cards. The TPC relaxes this limitation for Run2 with an upgraded version of the RCU, where a higher granularity of the GTL bus readout and faster DDL facilitates higher readout rates [15, 109]. An increase in readout bandwidth of approximately 80 % is expected [120].

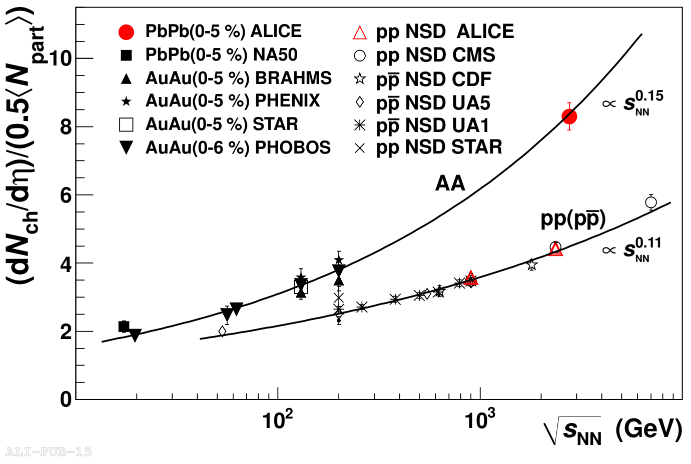


Figure 4.15: Charged particle pseudo-rapidity density per participant pair for central nucleus-nucleus and non-single diffractive p-p/p-$\bar{p}$ collisions as a function of $\sqrt{s_{NN}}$. Source: [10].

The increased collision energy in Run2 implicates an increased number of charged particle tracks per interaction, i. e., an increased average event size, and further challenges the subsystem readout electronics. Using the charged particle pseudo-rapidity density for nucleus-nucleus collisions at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$ as baseline, see Figure 4.15, an average increase in event size of around 25 % can be estimated for Pb-Pb collisions at $\sqrt{s_{NN}} = 5.5\,\mathrm{TeV}$.

Considering the increase in event sizes as well as the increased readout bandwidth of the TPC in Run2, readout rates of 400–550 Hz, depending on the trigger mix composition of central and semi-central Pb-Pb triggers, must be expected in common readout scenarios for the central barrel detectors. As shown in Chapter 7, the Run1 TRD readout electronics is capable of operation up to 250 Hz for central Pb-Pb at an acceptable dead time below 20 %. To reach the envisaged Run2 rates, an update of the TRD readout electronics is required.

Trigger Class Extensions and CDH Update The trigger classes available have been used up by the variety of different trigger conditions in the course of Run1. To remove this limitation, the number of classes is extended for Run2 from 50 to 100 classes [83]. Similarly, the number of clusters is extended from 6 to 8. The trigger class and cluster information is transmitted to the detector FEE as part of the L1 and L2A trigger messages, which in turn uses the information to construct the CDH. Their extension therefore has consequences to trigger reception and local event building. Both have been adapted for Run2 to be compatible with the upgraded CTP and DAQ systems.

4.7.1 Options for a Readout Bandwidth Increase

A look at the bandwidths of components and interfaces involved in the handling of raw data in Run1, see Figure 4.16, reveals that during local event building the SMU can access data in each TMU event buffer at a rate of $4.8\,\text{Gbit/s} \cdot 40/32 = 3.84\,\text{Gbit/s}$ given the current LVDS uplink transmission format. This rate may decrease only for short periods of time during FEE data transfers, see Figure 7.2. Fully concurrent access, as currently exercised at the start of each event readout when filling the SMU input FIFOs, even offers a total aggregate net bandwidth of 19.2 Gbit/s. The DDL line rate of 2.125 Gbit/s and the SMU-SIU interface limitations resulting from exceptionally tight SIU timing requirements can clearly be identified as bottleneck of the Run1 readout. Readout performance simulations, see Section 7.4, show that an increase of about a factor two in readout bandwidth is sufficient to meet the target readout rates envisaged for Run2 and match the performance of the upgraded TPC electronics.

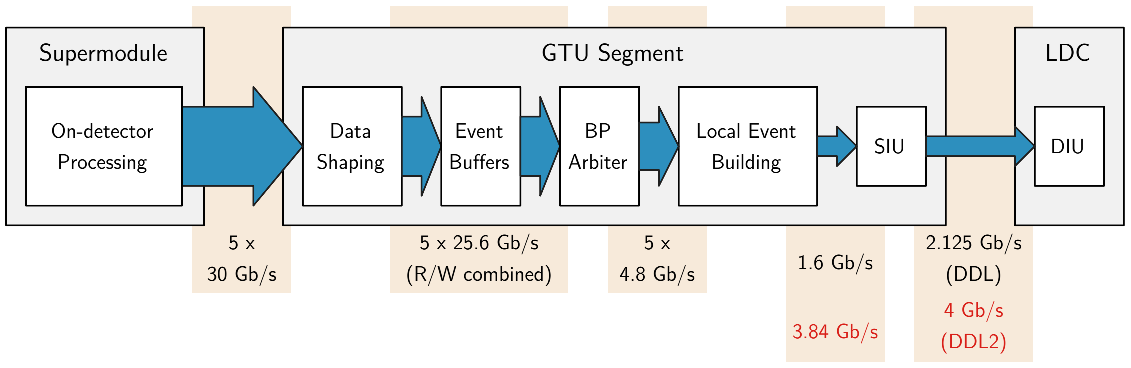


Figure 4.16: Uncoded transmission rates of components and interfaces involved in the raw data handling for one TRD supermodule in Run1 (black) and Run2 (red).

In Run2 the newly developed Common Readout Receiver Card (CRORC) is available, which is used as interface card for ALICE DAQ and HLT [47, 29]. The CRORC features a configurable high-precision oscillator, which provides the transceiver reference clock. This gives the flexibility to select from a wide range of line rates and supports legacy first generation DDL hardware as well as second generation DDL2 hardware at line rates of up to 6.6 Gbit/s.

To increase the total TRD readout bandwidth, several upgrade options were identified:

- SIU add-on board redesign
  A redesign of the SIU add-on board with optical transceivers at an increased line rate could constitute the source side of the DDL2 and remove the tight timing constraints of the current SMU-SIU interface.

- Implementation of the SIU functionality within the SMU FPGA
  Due to the common PCB layout of the three GTU node variations, the SMU board possesses four spare gigabit transceivers and sufficient resources to integrate the functionality of the SIU core into the FPGA. One of the spare transceivers could be utilized to implement the physical DDL2 transmission layer and interface the CRORC at an increased line rate.

- Increase the number of DDLs per TRD supermodule
  An increase of the number of DDLs per supermodule could be facilitated by mounting a second SIU on a currently unused PMC connector on the SMU. Another option to increase the number of DDLs would be an extension of the previous option, namely the instantiation of several SIU cores and transceivers within the FPGA.

All of the presented options are suitable to achieve a doubling of the readout bandwidth. However, the integration of SIU functionality into the SMU and transmission over one DDL2 at an increased line rate is most favorable. It does not require the development of new hardware or major modifications of the existing setup. At a target line rate of 4 Gbit/s, the only required modifications to the installed system are reconnection of DDL patch fibers to the SMU SFPs and installation of CRORCs. The use of a single DDL2 per supermodule furthermore avoids the complexities related to the distribution of data from a single event fragment over several links. The global event building concept relies on the availability of an event fragment with valid CDH on each detector link for each event read out. A simple scheme, where event n would be send via link 0 and event $n+1$ would be sent via link 1, would therefore require the generation of small event fragments on the unused link and potentially some advanced pre-fetching/buffering stage on the SMU to ensure that both links can be saturated simultaneously. Another scheme, where data of one event from one supermodule would be distributed over several links, e. g., stacks 0 and 1 via link 0 and stacks 2, 3 and 4 via link 1, would lead to unbalanced link utilization. More importantly, this splitting of event data would require significant changes to the TRD DDL format and, subsequently, in the offline processing software. The solution with a single, yet faster detector link per supermodule instead preserves the data format and ensures full backwards compatibility.

The two major tasks required to implement the selected solution and ensure an efficient use of the available DDL2 bandwidth, integration of the SIU core and an upgrade of the TMU-SMU data path, are presented in the following.

4.7.2 Integration of the SIU Core into the SMU

The implementation of the detector-side DDL2 interface is presented in Figure 4.17. The Virtex-4 MGT operates at 4 Gbit/s line rate, which is the maximum achievable line rate with the equipped on-board 250 MHz reference oscillator and the permitted clock multiplier/divisor settings of the MGT core. DDL data is 8b10b encoded, therefore the core presents a 16-bit interface at 200 MHz to the FPGA fabric.

All components of the SIU core[24] are adapted to the Virtex-4 architecture and integrated into the SMU FPGA design. The interface to the SIU daughter board existing in Run1 is ported to directly interface the SIU core with a 32-bit data path and a number of control signals. Together with the framing entities, the core implements the logical layer of the DDL protocol [101].

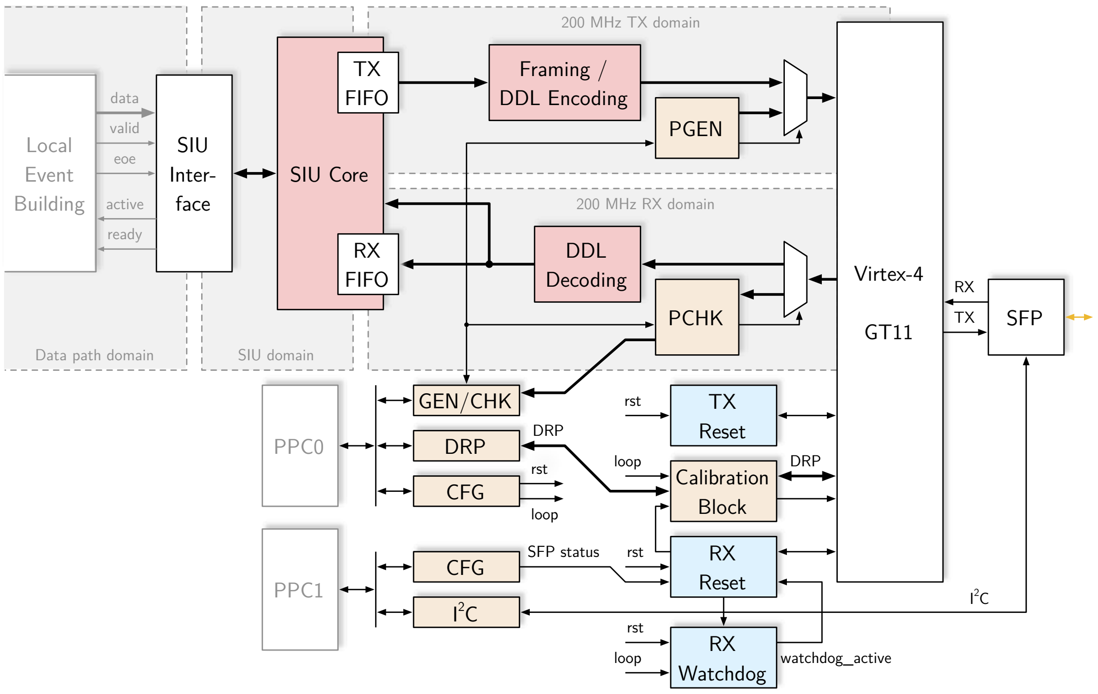


Figure 4.17: SIU core instance and MGT infrastructure implementing the detector side DDL2 interface on the SMU. SFP power monitoring, calibration block and RX watchdog are required to ensure safe operation of the MGT independent of the status of the far side transceiver.

A pattern generator and corresponding checker have been implemented to run full-rate transceiver tests. The operation of generator, checker, data path, but also the status of the MGT can be monitored and/or controlled by the user at runtime via the high-level PPC. In combination with the far-end loopback mode available in the Virtex-6 GTXs of the CRORC, the complete DDL2 physical transmission layer, comprised of GTU transceivers,

[24] A reference implementation for Virtex-6 is provided by the ALICE DAQ team.

patch panels, connectors, DDL fibers exceeding 100 m in length and CRORC transceivers, can be efficiently be tested for correct operation in situ.

Safe Operation of the MGTs Erratas published by the FPGA vendor indicate problems related to static operation behavior of the Virtex-4 transceivers. They state that the transceivers might permanently cease to transmit or receive data after a given number of cumulative hours if the following conditions are met:[25] The FPGA is powered and, depending on whether the device is a production or a late engineering sample, the transceiver is not instantiated or no transitions are occurring in the transmit or receive direction [117, 111]. The time these conditions have to apply before permanent damage occurs varies depending on temperature, stepping and sample type from few hundred to few thousand hours. The recommended workaround is to detect the aforementioned conditions and bring, if met, the transceiver into a safe state by configuring the MGT to operate in the serial pre-driver loopback mode. The vendor-supplied calibration block generates, on request, the corresponding DRP commands to reconfigure the MGT. Considering the typical core operation temperatures of the GTU FPGAs in the cavern of around 65 °C, transceiver problems are not expected to manifest before 1000 h cumulative. However, this erratic transceiver behavior places certain demands on the system architecture[26] and interplay of GTU and CRORC.

The reset entities, which are modeled in due consideration of the guidelines specified in [115] and the special requirements arising from the Virtex-4 errata, bring PMA and PCS of both MGT RX and TX directions to an operational state. The RX reset FSM, see Figure 4.18, additionally implements a safe state. It has inputs for

- a manual reset request,
- the lock signal of the RX PLL `rx_lock`,
- the SFP loss signal `sfp_loss`,
- the SFP connected signal `sfp_con` and
- an inhibit signal from the RX watchdog `watchdog_active`.

The signal `rx_signal_ok` is generated as

```
rx_signal_ok <= sfp_con and
                (not sfp_loss) and
                (not watchdog_active).
```

An inactive `rx_signal_ok` ultimately brings the FSM to the CHECK_POWER state, in which the activated calibration block forces the MGT into the safe pre-driver serial loopback mode.

Two scenarios have been identified that require special attention to ensure transceiver survival throughout the experiment's lifetime. In the first, a QSFP channel on the CRORC

[25] The special case of an unconfigured FPGA can be neglected in the GTU production setup.

[26] Active MGTs on all node types are protected in a similar fashion. For MGTs that are not in use, but whose RX and TX lines are route to SFP cages, corresponding vendor-supplied dummy logic templates are instantiated.

is powered off or disabled, e. g., during DAQ maintenance periods or scheduled shutdowns, but the GTU is not. As a consequence, no transitions occur on the MGT RX line on the GTU side. The required signal to switch to the safe mode is provided by the SFPs employed on the SMUs, which allow for pseudo real-time access to the operating parameters and diagnostics [50, 105]. The link status and power information is constantly monitored by the low-level PPC via the I^2C interface and is made available in the FPGA fabric. In case a lack or loss of RX power is detected, the RX FSM moves to the safe CHECK_POWER state.

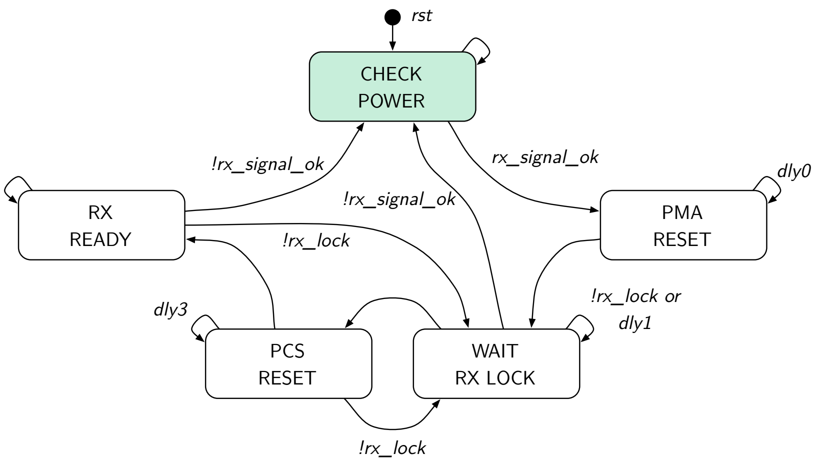


Figure 4.18: Simplified view of the MGT RX reset FSM. In the CHECK_POWER state, the calibration block is enabled, which configures the MGT to the safe serial pre-driver loopback mode.

The second scenario relates to the GTU MGT PLL not being able to lock onto the recovered RX clock. By default the configurable oscillator providing the GTX reference clock on the CRORC operates at a frequency corresponding to 2.125 Gbit/s. To operate at 4 Gbit/s, as required by the GTU, a reprogramming of the oscillator via the host LDC to 200 MHz is required after powering up the CRORC. It is unclear whether the resulting incapability of achieving RX lock might have similar implications on the long-term stability transceiver performance as the described static operation behavior. To rule out any possibility of damage to the SMU transceivers, a procedure implemented on the DAQ-side turns off optical power on the QSFPs if an incorrect oscillator frequency is detected. In turn, the corresponding MGT is forced into the safe state. Only after the correct oscillator frequency is set, the optical power in DDL downlink direction is turned on and the MGT attempts a new RX lock cycle.

In addition, a watchdog on the SMU side monitors the time span required to reach the RX_READY state after leaving the CHECK_POWER, which typically takes less than 350 µs. If the ready state is not reached within 15 s, the watchdog causes the RX reset FSM to move to, and remain in, the safe state. Configuration of the oscillator and powering on the QSFP transmitter diodes is implemented as startup service on the TRD LDCs. Therefore, the watchdog should never fire during normal operation. If it does, an RX reset request (either manually or via the DCS system) is required to acknowledge the abnormal situation and trigger a new lock cycle.

4.7.3 Data Path Upgrade

To ensure maximum utilization of the available DDL2 bandwidth, the data path in the TMUs and SMU is upgraded. With the inefficiencies introduced by the 8b10b encoding and SIU framing, see Section 7.2, the 4 Gbit/s line rate corresponds to an effective output rate of approximately 3.1 Gbit/s. Consequently, the operation frequency of the event fragment dispatcher and all other entities of the data path domain, e. g., trigger and tracking buffers, must be increased.

Figure 4.19 summarizes the changes. The frequency of the data path domain is doubled to 120 MHz, so that it can fill the SIU decoupling FIFO at 3.84 Gbit/s. This FIFO, which handled the clock domain crossing from data path to slower domain of the SIU add-on card in Run1, is obsolete as the SIU TX FIFO, which is part of the SIU core, now interfaces the serializer domain. However, it is kept and even increased to 256 Kibit to compensate potential short drops in the available SRAM read bandwidth due to the write prioritization (see Section 7.1.3). Sizes of other FIFOs in the data path are increased for the same reason.

To minimize the occurrence and effect of SRAM read rate drops, the SRAM interface, see Section 4.2.1, is revised. The upgraded version weakens the strict write prioritization by introducing a threshold on the number of pending writes upon which ongoing reads are aborted. This mitigates the negative effect of the required no-operation cycles when switching from read to write. At the current setting, eight consecutive reads can be guaranteed. Special read abort requirements, e. g., forced SRAM write at the end of the FEE transfer to minimize the corresponding dead time contribution d_{tx}, are handled by the implementation. Optimized and non-optimized interfaces are compared in Section 7.1.3.

A similar optimization is performed concerning the release of the SIU backpressure, which is generated contingent on the fill level of the SIU TX FIFO. With proper optimization of the corresponding threshold, unnecessary cycles without transmission of data words are eliminated when the backpressure is released and the transmission is restarted.

4.7.4 Trigger Reception for Run2

The extension of trigger class information to 100 bits and clusters to 8 requires updates of the TTC receiver, multi-event control and local event building. The implementation is straightforward due to the modular architecture of the receiver. Although not a primary design goal since trigger and DAQ systems of all existing test setups are moved to the Run2 formats, backwards compatibility is retained via a generic. The TTC receiver is updated to handle the new L1 and L2A trigger messages, which now consist of 9 and 13 IAC frames, respectively, and to detect errors within their transmission. Interpretation of the new messages formats as specified in [75], extended buffer capacity for header data and

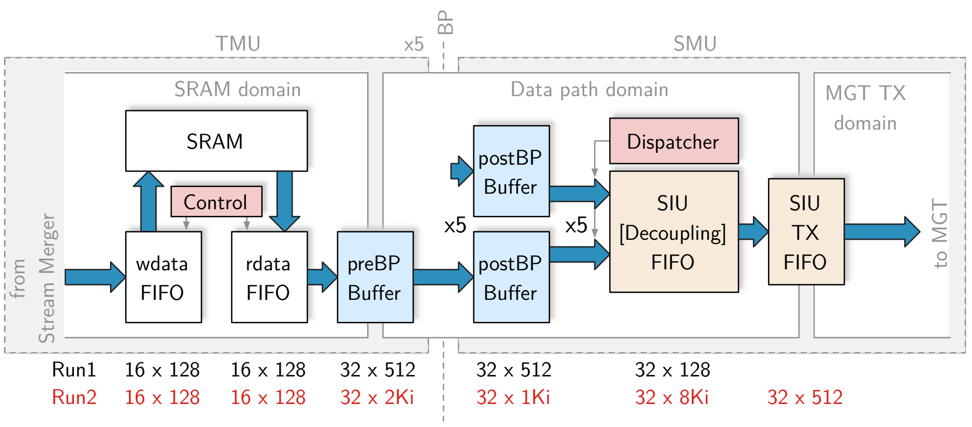


Figure 4.19: The locations of buffers, their respective sizes and relevant control elements in the TMU-SMU data path in Run2. Colored items have been optimized to allow for high possible DDL2 utilization. Noted in black/red are the sizes of data path buffers for Run1 and Run2, respectively.

the generation of the CDH version 3 as specified in [44] are implemented in the multi-event control unit.

Summary

The presented data path performs the buffering of event data, assembly of event fragments and their efficient readout. It provides stable, reliable and correct handling of event data that originates from detector chambers operating in a harsh radiation environment.

The implemented real-time supervision of the 1080 incoming 2 Gbit/s link data streams by dedicated shaper units, combined with potential reshaping to a defined format if necessary, has the beneficial effect that each GTU segment is agnostic to misbehaving detector chambers with respect to running stability. Detector chambers that malfunction, e. g., due to radiation-induced errors, do not stall or end an ongoing run. Corresponding corrupted link data streams are flagged by the GTU to enable efficient online chamber health assessment by higher-level data quality monitoring. As a second beneficial effect, the complexity of all downstream data path entities is reduced significantly, especially with regard to handling buffered data of multiple events. This makes their design simpler, and thus much more reliable.

A segment's data path is constituted by numerous entities on the six FPGA nodes of the segment. The TMUs buffer the event raw data until potential readout in accordance with the control signals issued by the segment control presented in Chapter 3. Requested for readout, the event raw data is moved to the SMU via a custom LVDS backplane, a resource shared with the trigger application. The interface to this backplane employs a direct-clocking technique and prioritizes trigger and control information to guarantee the required low latency, yet provides all remaining bandwidth to the event data transfer. Line-individual calibration of the corresponding data-to-clock delays contributes to low bit error rates of $< 10^{-15}$ for the transport through the full segment.

The readout unit on the SMU assembles a structured event fragment for readout that contains raw event data, but is also augmented with meta data allowing for a full offline evaluation of the GTU's tracking, trigger and readout functionality. CRCs calculated at distinct locations in the design are part of the meta data and provide references to assess the data transport quality through a GTU segment live. To minimize the processing time of the readout stage, and therefore the readout dead time τ_{ro}, the data path is designed to fully utilize the output bandwidth provided by the DDL interfaces of Run1 and Run2.

5 Control System Integration

The primary objective of a particle detector's control system is to achieve the concerted operation of the detector's subentities in a safe and efficient manner. If well designed, the control system helps to keep downtime to a minimum, which increases the running efficiency and amount of high-quality data taken during operation with beam. Since an experiment is typically run by a crew of non-expert operators, its control system has to provide an interface of manageable complexity that supports the tasks recurring during regular data taking operation as well as those arising in standby and maintenance periods.

The Experiment Control System (ECS) of the ALICE detector is a control layer that interconnects the activity domains represented by the four online systems DAQ, HLT, Detector Control System (DCS) and CTP, and ensures their synchronized operation. It supports grouping of detector resources into partitions that function independently and provides the primary interface to control the operation of the ALICE experiment. The DCS interfaces the detector subsystems and associated infrastructure systems. It models their operation as a hierarchical tree of state machine-like objects. The state of the top node object, which represents the overall state of the subsystem, is communicated to, and evaluated by, the ECS.

Its crucial roles as trigger and readout processor require a seamless integration of the Global Tracking Unit (GTU) into the overall control system architecture. This chapter presents the concepts to integrate the GTU into the control system of the Transition Radiation Detector (TRD), and summarizes development and functionality of the related services and tools.

5.1 Overview

Centrally steered operation as part of the ALICE detector, inaccessibility of the hardware due to the installation in the C-Area, and use also in test or production setups imposes a number of requirements on the GTU control system.

For automated, ECS-controlled operation, a suitable description in terms of a state model and the integration into the superordinate TRD Finite State Machine (FSM) unit [93, 32] is required. Transitions between the states must result in corresponding initialization and configuration actions being executed on the different types of FPGA nodes within reasonably low time, typically in the order of few seconds. This facilitates a fast recovery in case of an error. In such a case, the system should foresee means to gather the relevant debug information. Since the GTU is not accessible during accelerator operation, ways to control, monitor and debug the system fully remotely have to be provided. The corresponding utilities must integrate well with the automated FSM operation and also support manual inspection and intervention by the expert. Contrary to the production environment, the ECS/WinCC infrastructure is not necessarily available in the locations where reduced GTU setups are used for testing and supermodule production. It is therefore desirable to develop the system and required tools independently with only minimal dependency on WinCC. Last but not least, the control system must ensure safe operation of the hardware.

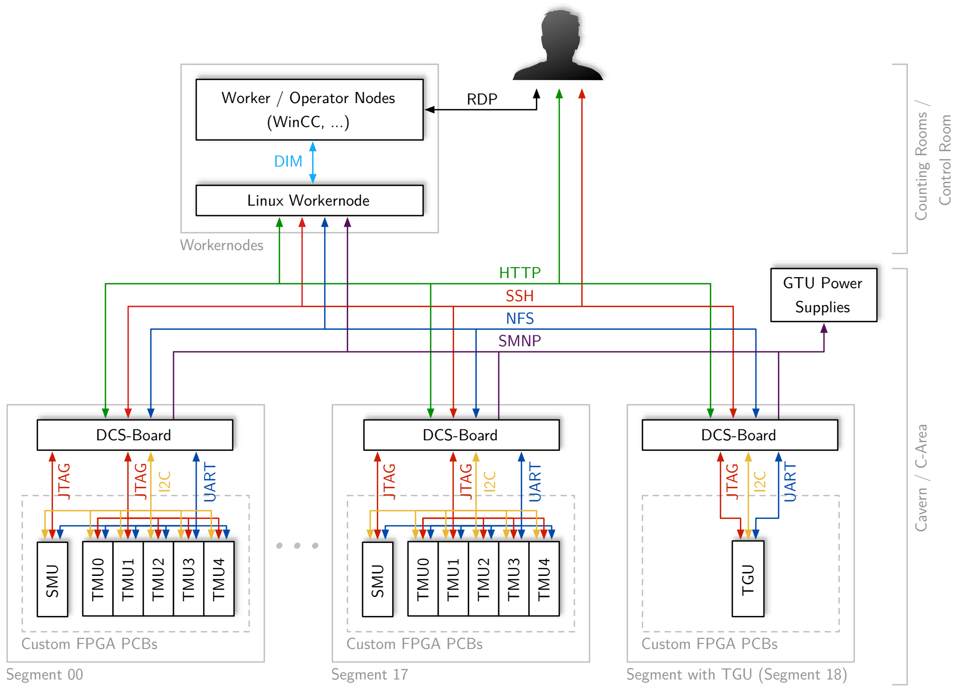


Figure 5.1: Hardware components of the GTU control system. The systems resorts to standard protocols for intercommunication and data exchange as far as possible.

These requirements lead to the layered control system architecture depicted in Figure 5.1, whose details are discussed in the remainder of this chapter. A summary of key characteristics of the hardware used can be found in Table 5.1. Services running on the worker node, which is a regular Linux server in the DCS network, implement the FSM and interface to the higher levels of DCS/ECS on the one hand, and serve as sources/aggregators for data to/from all FPGA nodes in the individual segments on the other. A DCS board per segment, see Section 5.2, establishes the connectivity to the DCS network. It serves as a platform to program and monitor the FPGA nodes, and run segment-related services. On the node level, a custom console application is implemented, which runs on one of the PPC cores in each FPGA. It provides the required routines for arbitrary node register access, configuration and information retrieval.

	Worker node	DCS board	GTU FPGA node
Scope	Global	Segment	Node
Number	1	18 + 1	90 + 18 + 1
Hostname(s)	alidcswn008	alidcsdcb06[00..18]	–
Location	Counting room	Cavern (C-Area)	Cavern (C-Area)
Platform	x86 (Linux)	ARM (Linux)	PowerPC (custom)

Table 5.1: GTU hardware running DCS-related services.

5.2 DCS Board for the GTU

The DCS board is a flash-based, standalone single-board Linux computer developed at the University of Heidelberg [58]. It features an ARM922T CPU complemented with 8 MiB flash memory as non-volatile storage, 64 MiB SDRAM as volatile memory, Excalibur Programmable Logic Device (PLD) and an Ethernet interface with customized, magnetic-field-tolerant physical layer.

On each of the SMU and TGU nodes, a DCS board is mounted as daughter board to connect the segment to the DCS network via Ethernet. The combination of RISC CPU, Linux operating system and PLD provides a flexible platform to implement low-level monitoring and segment control. The firmware used in the GTU bases on the standard variant for the TRD half-chamber DCS boards [82], but is heavily modified and extended with custom hardware modules and software.

Extensions of the PLD firmware, see Figure 5.2, include

- a custom JTAG module to program FPGA and PROM on the GTU nodes,
- a UART16550 core to communicate with the PPC console application on nodes in the segment, and
- a multiplexing module for forwarding the TTC channels (see Section 3.4.3).

The GTU-specific drivers and software, most notably

- the service daemon gtudcsd,
- the temperature interlock swinterlk,
- the programming application playxsvf, and
- the communication utility gtucom

are packaged as .ipk packages[1] to simplify DCS board software deployment and version updates in the field.

All DCS boards mount a network file system exported by the Linux worker node as mass storage and run a ssh daemon for local shell access to a GTU segment. System log messages are relayed to a central TRD syslog node.

[1]The Itsy Package Management System is a package management system targeted at embedded devices. It is similar to Debian's dpkg.

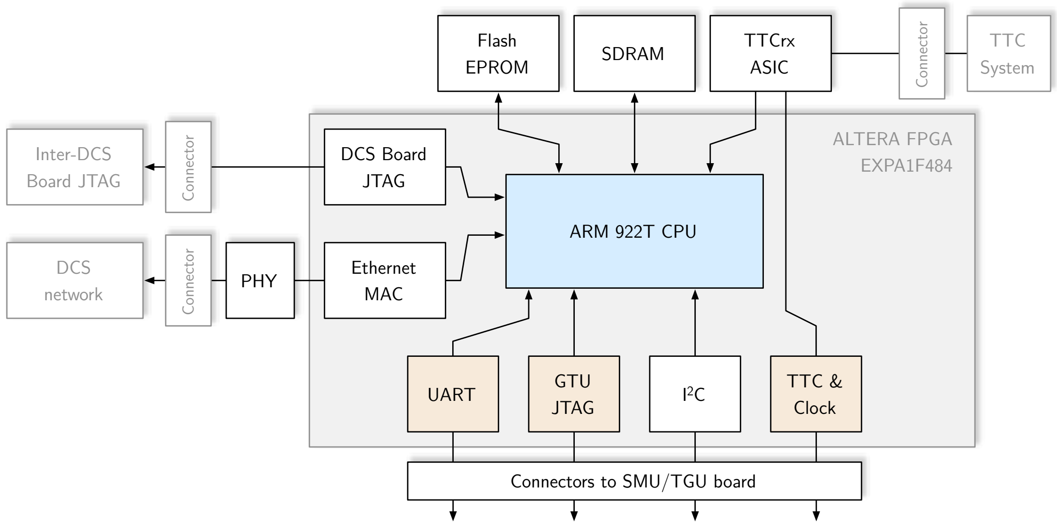


Figure 5.2: The Excalibur PLD firmware of the DCS board is extended by GTU-specific components (yellow). The DCS board connects to the GTU FPGA nodes, the TTC system, the neighboring DCS board, and the DCS network.

5.3 Firmware Updates

Inaccessibility of the hardware in the C-area requires solutions to remotely update the firmware of both the DCS boards and GTU FPGA nodes.

5.3.1 GTU Node Firmware Update

To facilitate firmware updates on the 109 GTU nodes, a JTAG[2] programming solution, consisting of hardware module, kernel module and accompanying application for programming, is implemented on all DCS boards.

Associated devices in a segment and their interconnections are displayed in Figure 5.3. The SMU connects two JTAG chains to its DCS board. The first daisy-chains PROM and FPGA of the SMU itself using single-ended signaling. The second chain connects the programmable devices of the TMUs in the segment in a parallel fashion via the LVDS backplane. Buffers with output enable and demultiplexers, all controllable from the DCS board via I^{2}C, allow both simultaneous or individual programming of the devices on the TMU nodes. In the case of simultaneous programming, the `TDO` signal of one selected TMU is read back. JTAG programming on the TGU is identical to the SMU, with the exception that the second chain is not connected.

[2] The Standard Test Access Port and Boundary Scan Architecture, commonly designated as JTAG, is defined in IEEE Standard 1149.1 [66]. It describes a 4-wire test access port, where the signals `TMS`, `TDI`, `TDO` and the free-running clock `TCK` are used to control, test and debug JTAG-compatible devices.

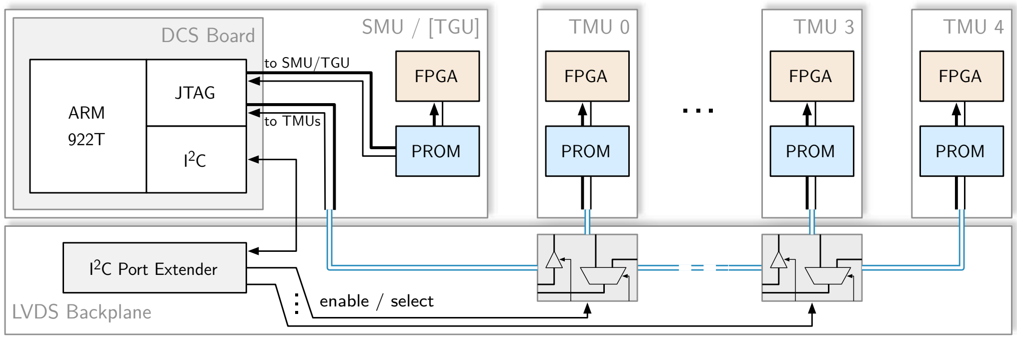


Figure 5.3: JTAG configuration chains of a GTU segment. Two JTAG chains are used to control the up to 12 devices in a segment. The signals of the TMU JTAG chain are routed from DCS board via the SMU PCB and then differentially via the LVDS backplane to the TMUs, which can be configured individually or simultaneously.

On the DCS board, a custom-developed, two-port Avalon slave JTAG module connects the two chains to the processor bus, see Figure 5.4. A kernel module maps the memory corresponding to the module's control and status registers to user space. A utility, jtag-ctrl, allows to control programming speed, select the port and retrieves status debug and status information from the hardware module. The programming application playxsvf is based on an example given in [85]. It is adapted to the DCS board and exploits the hardware acceleration features of the JTAG module to replay .xsvf files,[3] which contain instructions to program a device, initiate loading of the firmware from PROM, read JTAG user registers, and more. The scripted firmware build flow of the GTU nightly build system is aware of the JTAG chain layout on the nodes and automatically builds the .xsvf files required for in-system programming of the devices from the corresponding .bit or .mcs files.

Offload to Increase Programming Speed Regular practice is to keep the current stable firmware version for the given node type in the flash memory, from where it is automatically fetched at power-up or on request in an instant. However, for testing of new features and during development, the need arises to directly program FPGAs and not alter the stable default configuration in the PROM guaranteeing a fast roll back. Without optimizations to neither JTAG module nor programming software, even programming of the FPGA takes several minutes. This is not only an impediment for development, but also significantly slows down the initialization phase in automated FSM operation in case direct FPGA programming is used.

To lower the programming time to an acceptable level, the presented solution utilizes the software optimizations presented in [76]. Additionally, an analysis of the .xsvf files identifies two very common instructions. The first, which constitutes the vast majority when targeting the FPGA, moves the TAP controller to the shift data register state and handles shifting

[3] SVF files [30] describe high-level IEEE 1149.1 bus operations as a series of movements between states of the TAP controller FSM and scan operations. XSVF is a Xilinx-specific binary version of SVF targeted at embedded devices.

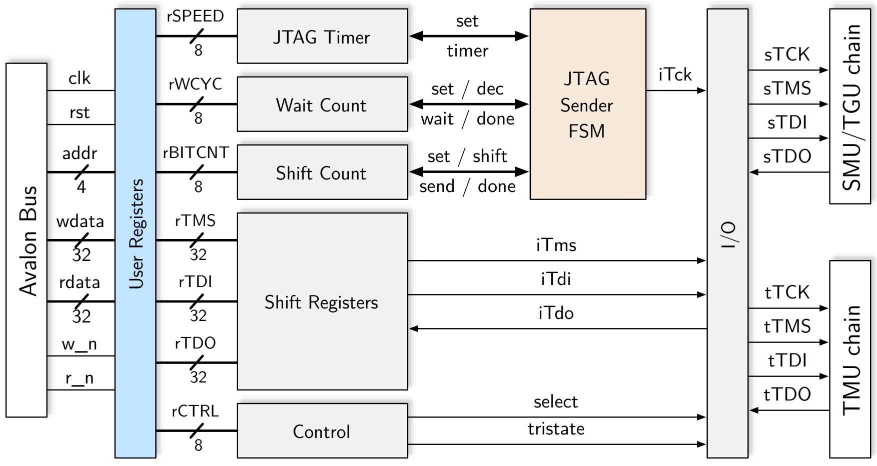


Figure 5.4: Block diagram of the Avalon bus slave module for programming of the GTU nodes. The control logic supports offloading of the two most common instructions encountered in the .xsvf progamming files of FPGA and PROM for significant performance gains. In the presented setup, it can drive JTAG data at configurable speeds up to 20 MHz.

of the data. The second instruction is very common in .xsvf files targeted at the PROM. It causes the controller to idle, but toggle the free-running JTAG clock line. By implementing dedicated support to offload these two most common types of instructions, the programming performance can be significantly improved. The corresponding hardware is built into the JTAG module and is used by the software. For further speedup, the update frequency of the temperature interlock, which emulates I^{2}C for sensor readout in software, is reduced to minimize concurrent CPU load during device programming. With all optimizations, the time to program a Virtex-4 FX100 device via the DCS board can be shortened from initially 4–5 min to about 14 s. This notably speeds up the development process and even makes it possible to program FPGAs during the FSM initialization phase.

5.3.2 DCS board Firmware Update

Due to the deployment of DCS board software as .ipk packages, the DCS board firmware requires updates very infrequently. However, a full firmware update becomes necessary when, e. g., the PLD design is modified. The standard self-update procedure [82] accesses the flash memory via the memory device subsystem of the Linux operating system. It allows to update all flash partitions (boot loader, boot environment, kernel image and root file system) as well as the PLD design while the system is running, and runs the updated design after a reboot. However, unforeseen problems, such as a power cut, appearing during the upgrade process may break the stored firmware image.

The recover the system in such a case, the DCS board design provides a fallback update method [82] that updates the target board from a functioning neighboring board via JTAG. This feature is employed by the GTU to implement a failover during DCS board system upgrades. Neighboring DCS boards of a crate are connected in a daisy chain, with JTAG cables as shown in Figure 5.5, to enable the neighbor update procedure. A split update of left-only DCS boards first followed by a verification of the functionality, and only then an update of the right-only DCS boards, allows to recover neighboring boards remotely even if a self-update or neighbor update fails.

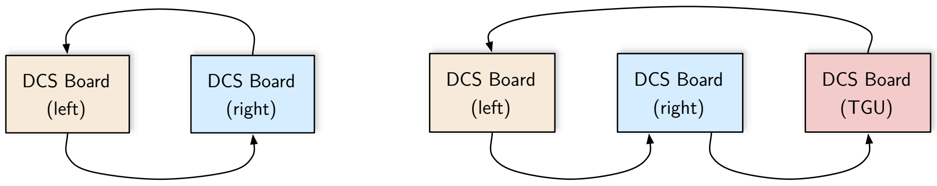


Figure 5.5: JTAG daisy chain loop within a crate (left: regular crate, right: crate with TGU) to provide failover during DCS board system updates.

5.4 Console for Node Configuration and Diagnostics

All GTU nodes implement an interconnected processing system comprised of the two PPC hard cores embedded in each FPGA. One of them is set up to perform low-level, quasi real-time monitoring tasks. The other, PPC0 or high-level PPC, runs a console application that implements control, configuration and diagnostics for the node. Program and data for this application reside in FPGA-internal block memories for reasons of simplicity. The two PPCs of a node can exchange data using a dedicated block memory.

The communication between DCS board and console applications on the high-level PPCs of a segment is established by a UART, currently configured to 57 600 baud. Due to strict I/O resource limitations, a single UART core on the DCS board is used to connect to the up to 6 PPCs in the segment. The TX line is connected in a parallel fashion, hence all PPCs receive identical TX data. The DCS board RX line is shared between all high-level PPCs in the segment as illustrated in [76, 41]. The scheme enables point-to-point communication initiated by the DCS board with individual high-level PPCs. The addressing is based on PCI geographical addresses. All PPCs monitor the UART for incoming commands. The PPC whose geographical address matches that of the command processes it and outputs the result on the RX line, while all other PPCs keep their output tri-stated. The scheme additionally facilitates broadcast commands, i. e., commands without address identifier. They are received and processed by all PPCs, but come with the restriction that responses to broadcast commands potentially cause collisions due to concurrent access to the RX line and should therefore be considered unusable.

A custom device driver module implements blocking open access to the UART resource on the DCS board and exposes a character device file to the application level. The utility `gtucom` is provided to access the console applications on the PPCs from the DCS board Linux console. Additionally, the daemon `gtudcsd` can relay received commands to the console, which is the typical communication method when running in automatic, FSM-driven mode.

The console application implements basic commands, such as raw register access to the configuration space attached to the PPC and certain low-level system monitoring, on all node types. More advanced functionality, such as calibration subroutines or diagnostics routines, see Table 6.1, is specific to a given node type. An integrated help function assists with an overview of the commands and options available on a certain node type.

5.5 Temperature Interlock

With respect to safe operation of the GTU hardware, the most perilous condition[4] is a failure of the rack cooling turbines and potential subsequent damage caused by overheating of individual FPGA nodes. An interlock system prevents that.

Due to the high device utilization and number of concurrently active MGTs, TMU nodes exhibit the highest operating temperatures[5] and reach a critical temperature about 35 s after a turbine failure. Since the turbines themselves are monitored by central DCS services, an interlock could be implemented on the WinCC supervisory layer. However, for this critical system an approach directly sensitive to the measured FPGA temperature is preferred as it involves fewer external and much less complex systems, and is therefore less prone to potential errors.

The software interlocks are implemented as daemons swinterlk that run locally on the DCS boards. Each daemon queries periodically the temperature and voltage sensors of all nodes of its segment via I^2C.[6] Gathered readings are stored in a shared memory segment to be also accessible by other processes. The utility swinterlk_ctl allows to adjust the sensor query frequency within valid limits and display readings of the last query cycle. The interlock defines configurable thresholds of severity 'warning' and 'fatal' for both FPGA and PCB temperatures. Exceeding such a threshold triggers a corresponding message to the system log, which is also relayed to the TRD syslog node. In case a fatal threshold is exceeded, the interlock then issues a command[7] via SNMP to the rack's power supply unit to power down all voltage groups associated with the segment.

5.6 Operation Controlled by the DCS

Control and supervision tasks related to the detector hardware are handled by the DCS. Within the DCS, the detector with all its infrastructure and subsystems is modeled as hierarchical tree of FSM-like objects. An object's state represents the state of a given

[4] Protection of the MGTs against static operation is implemented by the hardware in interaction with the low-level PPC. An example is presented in Section 4.7.2.
[5] Cold air enters the racks at the bottom and streams upwards through three GTU segments to the heat exchanger mounted in the top. Consequently, TMU FPGAs in the topmost segments are hottest with observed FPGA core temperatures of up to 65 °C.
[6] See [41] for documentation on the underlying I^2C monitoring infrastructure.
[7] The command is issued with a short delay to allow for the transmission of the log message to the syslog node.

functional entity, e. g., the readout system of a subdetector. The object is integrated into the hierarchical tree using an appropriate mapping of the object's states and those of its superordinate FSM object. The tree's root represents the overall state of the ALICE detector. Transitions between the states encapsulate necessary procedures needed to correctly and safely operate the entity. Top-level transition commands are propagated down the tree hierarchy where they trigger, after potentially being suitably converted by a respective mapping layer, the execution of certain entity-specific actions, e. g., programming a device and applying a basic configuration, on the tree vertices.

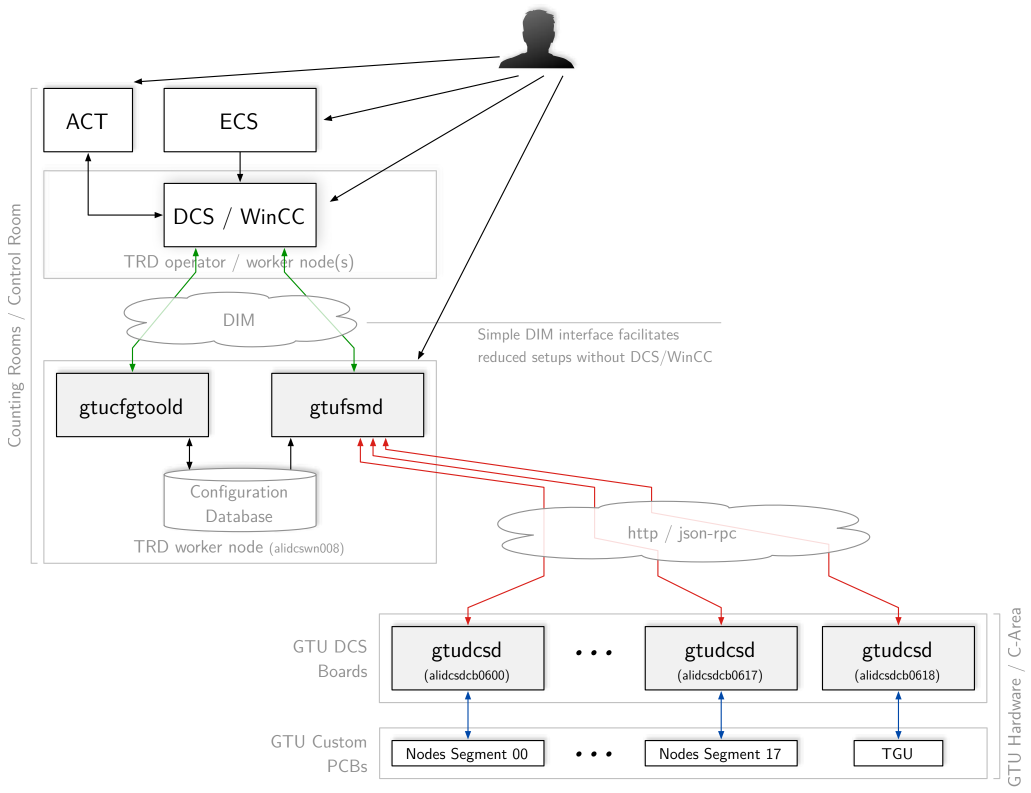


Figure 5.6: Overview of the system for automated operation of the GTU in the context of ECS. FSM service gtufsmd and configuration utility gtucfgtoold serve as interface to the higher-level DCS. The FSM service communicates with a service daemon on the DCS boards, gtudcsd, to program, configure and monitor the GTU segments.

Figure 5.6 presents an overview of the entities related to the automated operation of the GTU. The services running on the Linux worker node, gtufsmd and gtucfgtoold establish the interface to the higher-level control implemented in WinCC via DIM.

5.6.1 GTU FSM Service

The FSM service gtudcsd translates the received transition commands into actions executed on the GTU segments located in the cavern. It implements a state machine that manages the standard operation procedures. Figure 5.7 depicts the state diagram. The service is implemented in Python for reasons of portability and runs in daemon mode on the TRD Linux worker selected to run GTU-related processes. It is integrated into the control tree by an eligible mapping of states/transitions to those of the parent, the TRD control FSM, with help of a mapping layer in WinCC. Due to the similar state layout, most mappings are straightforward.

For control, gtufsmd provides the DIM command channel `gtufsmwn/Command` and publishes its current state as DIM data point `gtufsmwn/State`. To reach a different state, the mapping layer issues the desired sequence of transition commands to the command channel. Multiple commands can be issued in quick succession, as they are queued by gtufsmd and subsequently processed in receive order. The mapping layer evaluates the state data point to determine the state of the superordinate entity. The simple interface with command channel and state data point provides a natural way to decouple the operation of the GTU FSM service from WinCC, which facilitates its use also in the reduced setups for supermodule production and testing.[8]

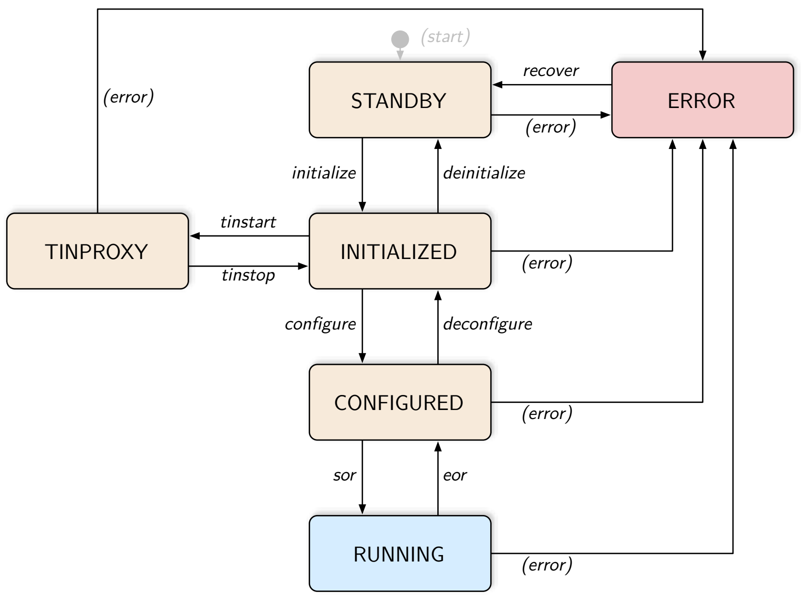


Figure 5.7: State diagram of the GTU FSM. The system remains in the current state in absence of a transition command or detected error.

The states defined by gtufsmd and their meanings are:

- STANDBY
  The segments are powered and each FPGA is configured with a safe default firmware.
- INITIALIZED
  FPGA nodes of the segments that are to participate in data taking are programmed with the desired device firmware.

[8]DIM provides command line utilities to send commands and retrieve data points in these cases.

- CONFIGURED
  The run type information, provided by the DCS, was received and any configuration depending on the run type was applied. The system is ready to commence data taking.
- RUNNING
  The Start-of-Run (SoR) command, sent by DCS, was received and the system is running.
- TINPROXY
  The outputs of the CTP interface add-on board are driven with test patterns to facilitate the adjustment of CTP input timings [74, 71].
- ERROR
  The system detected an erroneous condition and was moved to the error state. It remains in this state until the command to recover is given.

State Transition The validity of a received transition command is evaluated in context of the state. Any unexpected command is interpreted as error and forces the FSM to the corresponding ERROR state. If valid, the FSM initiates the required actions to reach the new state.

To best meet the demands of both production system with continuous updates and test system, these actions are defined in scripts executed by gtufsmd. A table in the configuration database maps a given script to the received transition command or a combination of transition command and run type.[9] This allows to support various running scenarios, the most important ones are listed in Table 5.2. The definition of available run types and existing scripts, the mapping of scripts to commands and run types and certain parameters passed to the scripts, are organized in a SQLite database residing on the TRD worker node.[10] As the GTU is designed to keep the number of configuration parameters low and encapsulate common procedures for direct execution on the high-level PPCs, the scripted approach introduces only a small overhead, yet offers a high degree of flexibility.

In case multiple scripts are to be executed for a given combination of transition and run type, an execution order is defined. Timeouts and a critical flag attributed to each script allow to detect and react to illegal conditions. The scripts may contain arbitrary commands, which can, e. g., target the local host, the console application on the FPGA node or the shell on the DCS boards. The FSM exports a number of environment variables for use in the scripts, such as the run number or mask of active segments. The utility gtucomwn simplifies the communication with the DCS boards and their gtudcsd service daemon. It allows to, e. g., quickly distribute commands to only the active nodes participating in the run or gather data from a selected set of nodes.

[9]The combination of run type and transition command is used as soon as it is provided by the DCS, i. e., for CONFIGURE, START-OF-RUN and their counterpart transitions.

[10]Utilities to easily modify the contents of this database, and therefore the configuration of the GTU, are provided.

Run Type	Description
PHYSICS	Standard physics operation
TECHNICAL	Technical (test) runs with the detector
COSMICS	Cosmic trigger configuration
PATTERN	Supermodule test pattern
GTUPATTERN	GTU test pattern generators

Table 5.2: Important GTU run types.

Operational Parameters from ECS In certain phases of the start-up process, the FSM relies on information on the from the ECS or the ALICE Configuration Tool (ACT), which are conveyed via the DCS. Examples are, among others, the mask reflecting the TRD supermodules participating in the run, the run type or the run number.[11]

To support operation with WinCC and without, the configuration utility daemon gtucfgtoold allows to modify relevant operational parameters stored in the SQLite database. The FSM queries the database for these parameters at given points in the start-up process. It makes them, for the period of their validity, available for use within the FSM or in the configuration scripts. The utility provides DIM command channels to alter the parameters to enable a straightforward integration into the DCS as well as standalone use in the reduced setups.

5.6.2 Service Daemon on the GTU DCS boards

Each DCS board runs a service daemon, gtudcsd, to support remote retrieval of requested information and execution of received commands. The daemon is based on Mongoose [90], a small footprint web server targeted at embedded devices.

The daemon is designed to work primarily, but not exclusively, with the FSM service on the worker node or the scripts executed by it. Retrieval of information and reception of commands is implemented as JSON-RPC [40, 73], a lightweight remote procedure call protocol. The transferred request and response objects are serialized using JSON. As the overall expected data volume for communication is low, the advantages of JSON, which brings implicit cross-node compatibility, wide support in modern browsers and programming languages, simplicity, and (human) readability, outweigh the disadvantage of a slightly increased data volume per message when compared to the transmission of binary data.

JSON-RPC defines the following three request object properties:

- A request `id` to match request and corresponding response,
- the `method` to be invoked,
- and array of objects in JSON array notation, `params`, that are to be passed to the method.

[11]Run type information is distributed by the ECS prior to the CONFIGURE command, the run number is updated just prior to the START-OF-RUN.

The corresponding response object implements the following properties:

- The response `id`,
- an `error` object, which contains detailed information in case of an error,
- and the `result` object, containing the response returned by the invoked method.

The current `gtudcsd` implementation supports the following methods:

- `shell` for the remote execution of shell commands on the DCS boards,
- `gtucom` to interface the PPC console application for command forwarding and information retrieval to/from the nodes in the segment, and
- `tv` for a quick summary of the segment's temperature and voltage data.

5.6.3 PPC Console Routines for Fast Initialization and Configuration

To speed up and simplify configuring the FPGAs, the PPC console application implements dedicated commands, which encapsulate common actions that have to be performed during certain state transitions.

Table 5.3 summarizes the actions performed by these commands, whose exact functional range and implementation varies depending on the given node type. Typically executed after a power-up of the nodes or programming of an FPGA, the command `dcs init` performs basic calibration of selected interfaces and applies a common, basic configuration to all FPGAs. Once executed, the system is in principle ready for data taking with a default configuration. The command `dcs sor` clears selected entities in-between two runs.

The corresponding routines are executed directly by the high-level PPC, which minimizes communication overhead over the network and, especially, the comparably slow UART. Certain features, see Section 5.7, are supported in hardware to guarantee fast execution and simplify the software implementation. With all node types supporting the same commands, it is furthermore possible to broadcast the commands on the UART to further reduces the configuration time (the response, however, cannot be evaluated).

With the aforementioned tools/services on the worker node and DCS boards, bringing all FPGA nodes in the system that participate in the readout to a basic operational state is simplified to a script which essentially contains

```
gtucomwn -t dcs-all-readout -g "dcs init"
```

and is executed by the FSM upon reception of a CONFIGURE. Deviating configuration options, e. g., enabling of the tracking's z-channel merging for cosmic triggering, can then be applied on top of the default as part of the specialized configuration scripts for a given run type.

Command	Action	Node type
`dcs init`	SRAM interface calibration	TGU, SMU, TMU
	TMU autodetection	SMU
	LVDS uplink calibration	SMU
	TGU uplink calibration	SMU
	Configure TTC logging	SMU
	LVDS downlink calibration	TMU
`dcs sor`	Initiate SoR reset/clear	TGU, SMU, TMU
	Reset statistics and counters	TGU, SMU, TMU
	Clear TTC trace memory	TGU, SMU

Table 5.3: The `dcs init` and `dcs sor` commands encapsulate several actions to bring the GTU to an operational state and feature differing implementations for the given node types.

5.6.4 Interface to the File Exchange Server

In ALICE, the extraction of conditions data necessary for offline analysis is steered by a system called Shuttle. It performs detector-specific merging, consolidation and reformatting to store conditions data in the Offline Conditions Database (OCDB) [59] on the grid. In case of the TRD, the process fxsproxy [81] receives a command from ECS and initiates the collection of conditions data from detector chambers, pretrigger system and GTU. Conditions data is assembled and placed on the File Exchange Server (FXS), a temporary storage, from where it is picked up by the Shuttle for subsequent processing at the end of each run. Shuttle and fxsproxy infrastructure with focus on the interplay with the GTU are depicted in Figure 5.8.[12]

The fxsproxy requests the transfer of conditions data from the GTU using a DIM command channel of gtufxsd. The latter serves as relay to retrieve FXS information via the DCS service daemon gtudcsd from the individual GTU nodes participating in the run. The PPC console supports this with the dedicated command `dfxs`, which returns the relevant data as JSON-formatted string and is implemented on all node types. Individual contributions to a GTU-wide response are merged by gtufxsd, reformatted to .xml as required, and published on the DIM result service.

The run-related conditions data of interest is primarily related to the design revisions used as well as to the configuration of the tracking and trigger algorithms, whose number and design was subject to constant development and optimizations during Run1, see [99]. Local extraction of FXS data in JSON format implemented in the console application and gtufxsd as interface layer to the fxsproxy are well suited to support ongoing developments.

[12] A revised implementation of gtufxsd, which removes DIM from communication with the DCS boards for reasons of stability, is under development at the time of writing.

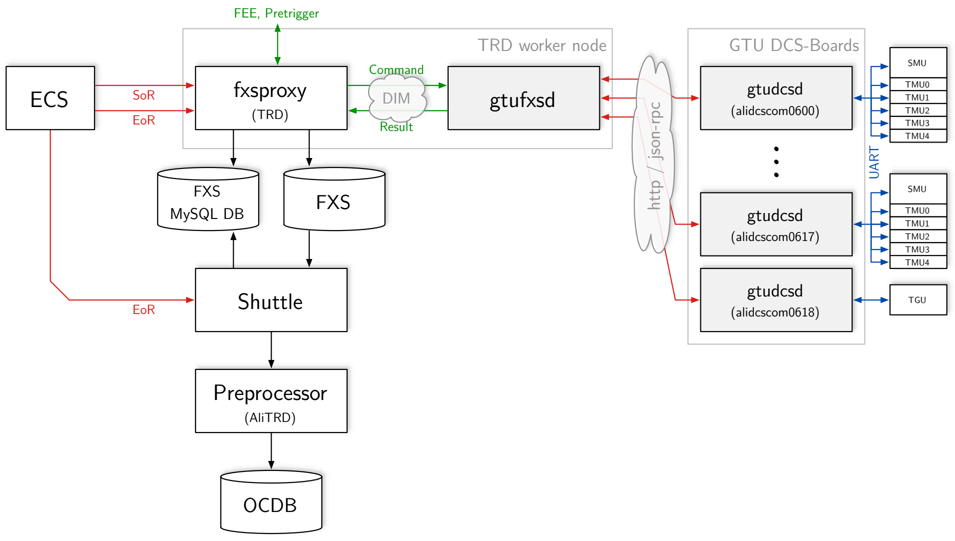


Figure 5.8: Retrieval of conditions data for the offline analysis. The fxsproxy requests conditions data from the gtufxsd, which in turn gathers respective data from the nodes and publishes the result. After the end of a run, the Shuttle collects the conditions data and processes it for subsequent storage in the grid OCDB.

5.7 System Startup, Clear and Recovery

The automated operation in context of the FSM requires the GTU to reliably start up to a defined initial state after device programming. Furthermore, fast clear procedures for certain entities and partial design resets are needed to facilitate quick restarts of runs with identical configuration. Figure 5.9 presents the distribution of reset and clear signals in a GTU segment. Details of the system reset generation, the inter-node reset, and partial resets supporting fast FSM state transitions are discussed in the following.

5.7.1 Node Reset Generation

A typical reset signal is a slow signal asserted for some period of time. At a certain point it is released and connected entities commence operation with their reset value. Although the release of the reset generally happens asynchronously, it has to be treated as time-critical event. Flips-flops in the same clock domain could resume normal operation in different clock cycles if the setup time of the reset signal is violated at one of the flip-flops due to routing delay differences. If a given set of flip-flops then represents the state vector of an FSM, the asynchronous de-assertion of the reset could lead to the FSM being locked in a non-existing state. Another important aspect to consider when implementing resets is the usage of FPGA routing and slice resources. Practices that come with resource utilization penalties in FPGAs are, e. g., the use of asynchronous resets, the distribution of global

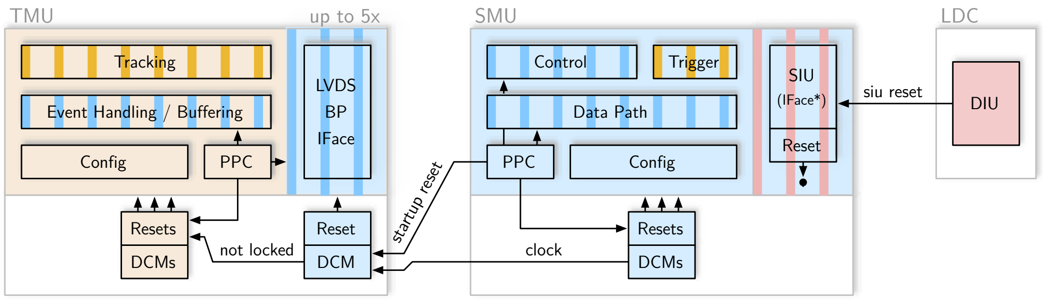


Figure 5.9: Reset and clear scheme of a GTU segment. Blue and yellow areas mark the scopes of the system resets of the TMU and SMU nodes. Design blocks marked with stripes support a partial reset, which brings the entities to an operational state without additional reconfiguration. As of Run2, the integrated SIU core and interfacing logic allow for external reset by the DAQ via the DDL.

resets instead of localized resets, and low-active resets. They therefore increase the routing complexity and potentially reduce the maximum operation frequency [37, 53, 107].

Figure 5.10 illustrates the functional principle of the generation of the node-wide reset as utilized in all 109 GTU nodes, which addresses the synchronous reset release also in case of multiple clock domains. Shown is a simplified case with two unrelated clock domains, slow and fast. A reset cycle is initiated when the lock output of at least one Digital Clock Manager (DCM) is low, such as after power-up or FPGA reconfiguration, or if requested via the PPC system. The corresponding signal (red) causes the negated output of the D Flip-Flop (FDCE) primitive to asynchronously activate the reset in the slow clock domain and also clear the master reset counter. Once all DCMs report stable clocks, the master reset is kept active for a sufficient amount of cycles to facilitate the synchronization of the master reset from the slow domain to the target clock domains, the distribution to the attached design elements and the clearing of all data pipelines and buffers. A description of the metastability resolution circuit used for re-synchronization can be found in [18]. The use of localized, synchronous reset signals enables the EDA tools to evaluate the respective timing constraints and choose an adequate placing and routing. If timing is met, de-assertion of the reset at all elements in a respective clock domain is guaranteed to happen at the same rising edge of the clock.

5.7.2 Reset of TMU Backplane Interface

The direct-clocking technique implemented on the LVDS backplane requires a reference clock, which is supplied by the SMU and distributed within the segment to a DCM on each of the TMUs. To achieve a consistent locking behavior and low jitter operation, Virtex-4 DCMs depend on stable clocks at their input and feedback ports at the time the locking process begins [116]. As an unstable or lost reference clock can lead to unexpected behavior in the segment, it is necessary to reset the backplane DCMs on the TMUs once the reference clock is stable (again) [112].

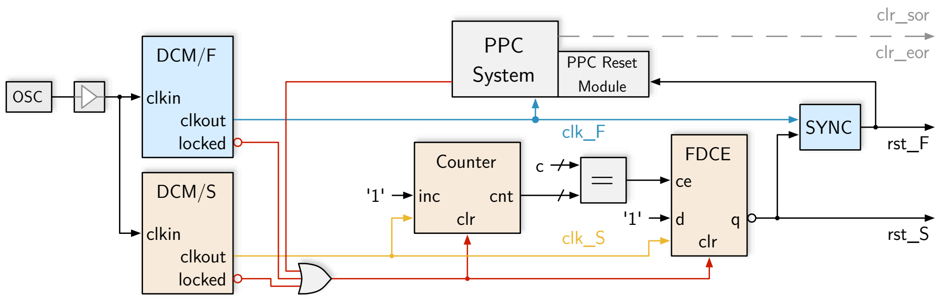


Figure 5.10: Functional principle of the generation of synchronous resets on the GTU nodes. An asynchronous signal (red), generated from the locked signal of the DCMs of the fast (F) and slow (S) clock domain and potentially via PPC, initiates the reset. The reset is released after a sufficient number of cycles in clock domain S have passed. Synchronization ensures a synchronous reset signal also in the fast domain.

To be agnostic to the programming or start-up order of the nodes in a segment, a reset of the TMU backplane DCMs is initiated as part of the SMU start-up sequence once the reference clock is stable. Figure 5.11 illustrates the clock distribution to the TMUs as well as the generation and distribution of the DCM reset signal and locked status. The reset of the TMU backplane DCMs is derived from the node reset signal of the SMU or initiated in software. It is prolonged to meet DCM requirements[13] and routed as bus reserved signal on the CompactPCI backplane. With the reference clock stable when the prolonged reset is released, each backplane DCM on the TMUs can acquire a new lock. As the negated backplane DCM locked output serves as input to the TMU node reset, reprogramming or resetting the SMU segment controller triggers a subsequent reset of all attached TMU nodes.

The locked status is communicated back to the SMU PPC via a CompactPCI shared line. Pull-up resistors set the line's default level to high, so that a single DCM unable to acquire a lock suffices to signal an error by pulling the line low. Special segment configurations, such as those with missing central TMU, are then supported transparently. The combined locked status is accessible from the SMU PPC system, so that the bidirectional delay adjustment process described in Section 4.3.1, can be automatically timed and performed as part of the PPC start-up sequence. This scheme guarantees that all segment nodes are in a well-defined, default operational state following a reset, reprogramming or power-up of the system.

5.7.3 Partial Resets and Support for Run-over-Error

To facilitate a fast recovery and quickly bring the system into a clean state, the nodes implement a 'clear' procedure for certain entities that neither affects the configuration register file nor the PPC subsystem. A software-controllable clear signal is connected to

[13]DCMs in Virtex-4 devices initiate a new cycle to acquire a lock after their reset input was active for at least 200 ms [113].

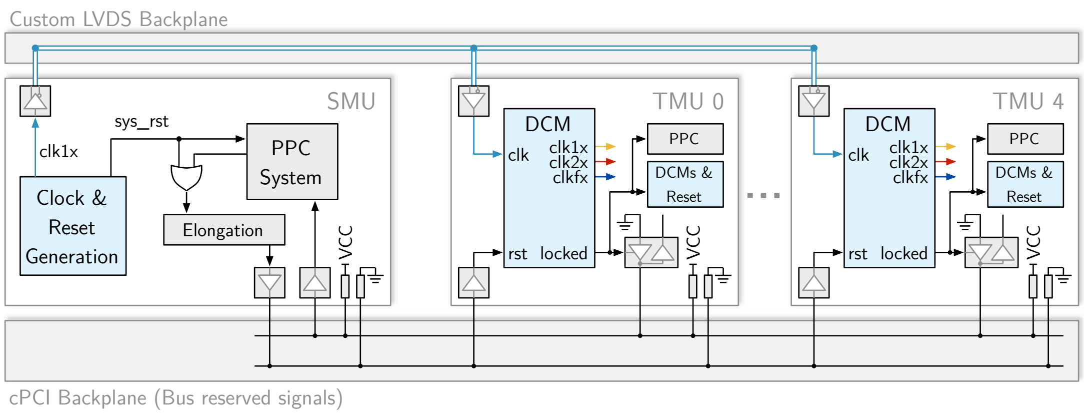


Figure 5.11: Clock distribution from SMU to TMU on the LVDS backplane and reset control lines on the CompactPCI backplane. Following a power-up or reset of the SMU, the DCMs on the TMUs acquire a new lock once the SMU-distributed clock is stable and initiate a reset of the entities in the associated clock domain.

selected entities of the data path and of the segment control unit on the SMU and TMUs, as marked in Figure 5.9.

If activated, these entities are reset to their initial state and associated statistics counters are cleared. Data path buffers and control signal pipelines are drained. In Run1, also the SIU interface and corresponding buffers are cleared. Entities related to tracking and trigger operate only in the timespan from L0 to L1. They are reset with the arrival of each L0 trigger and therefore do not require an explicit software-initiated clear. Activating the clear signal is part of the `dcs sor` command, which is typically sent to the nodes by the FSM in the SoR phase. As an applied configuration is retained, the procedure facilitates quick restarts of runs, if the configuration does not have to be changed.[14]

Based on experience from Run1, ALICE decided to implement a run-over-error strategy for Run2. It enables detectors to do certain adjustments, e. g., recover singular HV trips, without the quite significant overhead of a full start or stop of a run. The strategy foresees a fast stop and subsequent start of a new run without change of the fundamental configuration of the detectors. DDLs are kept active during the fast restart, i. e., no `RDYRX` status word is transmitted from DAQ to GTU at the start of the new run. As of Run2, the GTU, and therefore the TRD, complies with the requirements for this strategy and consequently excludes a reset of SIU core and associated entities from the clear procedure to retain the DDL status. The reset of the SIU core is now handled externally by the DAQ, which transmits a `siu_reset` command during a regular start or omits the command during a fast start.

[14]Quite commonly, errors in other subsystems may end a run and cause the DAQ to shut down the DDLs while not yet all data has flown off the GTU, resulting in stale data in the data path pipeline. The clear procedure also enables successful restarts in such a scenario, since a complete reset of the data path is implemented.

Summary

The GTU requires a sophisticated DCS architecture for flexible, robust operation and proper integration into the experiment. The chosen implementation with FSM-related processes running on a Linux server in the DCS network and segment-level control with a DCS board as underlying platform provides a flexible interface to the higher-levels of the control system and enables parallel, independent execution of control tasks on the individual segments.

By extending the firmware of the DCS board platform with custom-developed hardware modules and software, the DCS board can be adapted to the needs of the GTU and allows to realize the full remote administration and safety monitoring of the segment and its FPGA nodes. To address the restrictions coming from limited I/O resources, a lightweight multi-drop configuration bus connects the DCS board with the PPCs of each FPGA node. The console application on each node provides full access to the configuration registers, status and monitoring information, and facilitates direct command entry for maximum flexibility. To minimize the time to initialize/configure the system and avoid communication overhead, more complex, node-specific calibration and start-up routines are integrated in the console application for local, fast execution on the node's PPCs. By encapsulating the console application as integral part of the node's FPGA firmware, hardware and low-level software can be developed and maintained side by side. This guarantees their compatibility and greatly simplifies, together with the dedicated start-up routines that define a clear interface between segment-level and higher-level control, updates in the field.

The FSM service and a number of utilities for adjustment of the configuration integrate the GTU into the control system of the experiment, yet can also be used standalone without the higher-levels of the control system in reduced setups. Modeling the recurring tasks required to operate the GTU using a simple state diagram allows for a transparent integration into the hierarchical control tree and fully automated operation in context of the ECS controlled by the shift crew. With the FSM implemented as script execution engine that depends on run type and current state, a high degree of flexibility can be retained and various operational scenarios ranging from real physics runs to commissioning runs can be supported.

6 Verification and Diagnostics

Ever since the start of the ALICE experiment, the GTU successfully performed the local event building and readout of event fragments for the TRD. Over the course of the experiment, the node firmware designs have been advanced to include, amongst others, significant improvements regarding the readout performance and trigger capability. However, continuously upgrading a system used in production data taking comes at the risk of introducing new bugs, thus causing irregular system behavior. The preciousness of beam time and the fact that malfunctions may not only disturb the TRD, but the operation of the full ALICE detector, call for a thorough testing and verification procedure, before and when the designs are actually deployed in the field.

An overview of the simulation framework used to initially assess the correct operation, the testing setup, the procedures in the commissioning process, and the in-system debugging capabilities is presented in this chapter.

6.1 Firmware Simulation

Functional simulation allows to verify the functionality of the majority of the GTU firmware design and ensures that the design entities under test comply with their specifications. During development of the firmware designs described in Chapters 4 and 5, a simulation framework for a system simulation has been built, which allows to simulate the data handling and trigger aspects of the GTU.[1] To maximize the validity of the simulation, special care has been taken to provide realistic, timing-accurate stimuli for the link input data and TTC trigger sequences.

6.1.1 Framework Overview

For verification of the node firmware designs, they are embedded in a test environment that aims to mimic the production setup. An overview of this environment is given in Figure 6.1. The main test bench consists of board models of the various nodes, which in turn consist of the developed custom FPGA firmware designs, vendor-supplied IP cores, and models for the peripheral components on each board. All simulation models and wrappers are written in VHDL. Transport models for the SMU-TGU interconnect, the LVDS backplane and the CompactPCI backplane connect the different types of nodes to

[1]The framework has been maintained and continually extended in an collaborative effort with Felix Rettig, with contributions by Dirk Hutter.

form a full system. A hardware emulator for the trigger partition replays user-defined trigger sequences with realistic timing and takes into account the busy feedback loop. Regarding the link input data, numerous pattern generators are available to provide stimuli with different characteristics for a variety of scenarios. The setup allows to simulate the event buffering, trigger tasks and data transport through all levels of the GTU hierarchy.

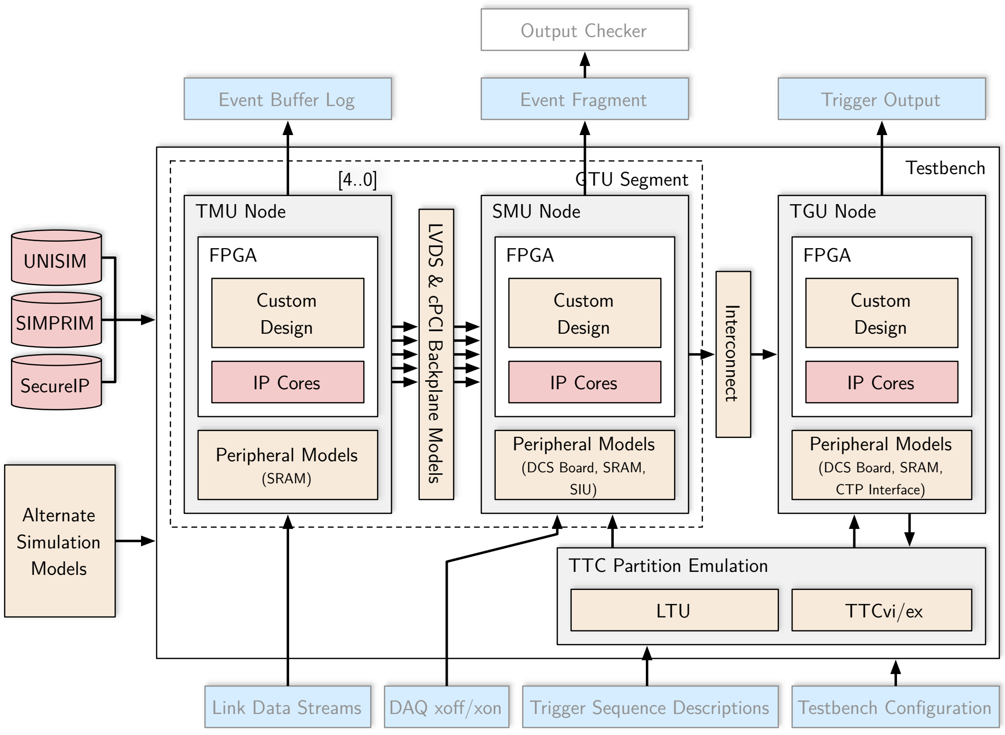


Figure 6.1: Overview of the GTU simulation environment.

Simulation Flow The user customizes the simulation[2] and selects the stimuli for the input data source and trigger emulation. The simulation is scripted around ModelSim as simulation tool and controlled through .tcl scripts. The main simulation script then executes the necessary initialization and setup routines across all node firmwares under test to produce an accordingly configured GTU setup. A makefile assists with the incremental recompilation of modified hardware entities, the initial setup of the simulation and the generation of the stimuli.

The user may then run the simulation for a defined time and inspect the results. Waveform scripts targeted at specific design blocks (event handling, backplane interfaces, trigger, etc.) are available to generate waveform views of important, preselected signals. Especially in the case of simulations targeting data transport and event buffering, manual inspection is insufficient as the amount of produced data is large. Here, data sink components integrated

[2]The full configuration space of the nodes, such as data path multiplexers, trigger timings, etc., is controllable.

into the SIU and SRAM models produce text output files with intermediate and final results. Integrity of the resulting output can thus automatically be checked using the tools developed for the online verification, see Section 6.2.2.

The time to simulate large designs like a full GTU segment is significant and can easily exceed several minutes for a single trigger sequence or event. To reduce the simulation effort, the framework allows to adjust the segment layout.[3] Initialization tasks that are normally handled by the PPCs, e. g., the IDELAY calibration at startup, are scripted in tcl. Mock-up versions of these scripts are available to shorten the simulation time in case the associated hardware is not under test.

Integrated Tests Individual test benches for components are used where suitable, especially in the initial stages of development. In addition, testing of the components in a full system simulation gives a number of benefits:

- As the system simulation wraps the board top-level designs, the integration of new components comes for free and components can be simulated with little additional effort.
- The focus is shifted from individual tests with hand-crafted stimuli for each module to simulations that rely on realistic variants of the most basic stimuli, which are detector data, trigger sequences and readout backpressure patterns.
- These stimuli are, if necessary, upgraded or extended to reflect the behavior of the real system and already existing components are automatically re-tested.

Although a full system simulation has rather high demands with respect to invested simulation time, the fact that inconsistencies and bugs in the components of individual firmware designs and their interplay across the hierarchy can often already be detected at this stage justifies the effort.

6.1.2 Board Models for Simulation

TMU Board Model The TMU board simulation entity consists of the custom FPGA design, a vendor-supplied model for the two SRAM chips and a log entity, which provides an intermediate snapshot of the SRAM content. Stimuli for the link data streams enter the GTU simulation via the TMU models. The TMU firmware incorporates hardware event generators, which can be utilized also in simulation to supply input data. Figure 6.2a shows the configuration options for the input stage with MGT entity and the two types of available hardware event generators, PRNG and DCR-attached memory event generators (see Section 6.2.1).

Running the MGT SmartModels is expensive with respect to simulation time. For most aspects regarding the simulation of the data path and trigger path, it does not give

[3]For tests regarding, e. g., the segment control logic, a single TMU in a segment may suffice, while simulations of the tracking may require presence of all TMUs in a segment.

additional benefit. The entity containing the already well-tested, customized instances of the MGTs for data reception at 2.5 Gbit/s is therefore replaced[4] with a mock-up. In addition to the generators in the fabric, the entity replacing the MGTs, see Figure 6.2b, offers two link-level data sources specifically adjusted to simulation purposes:

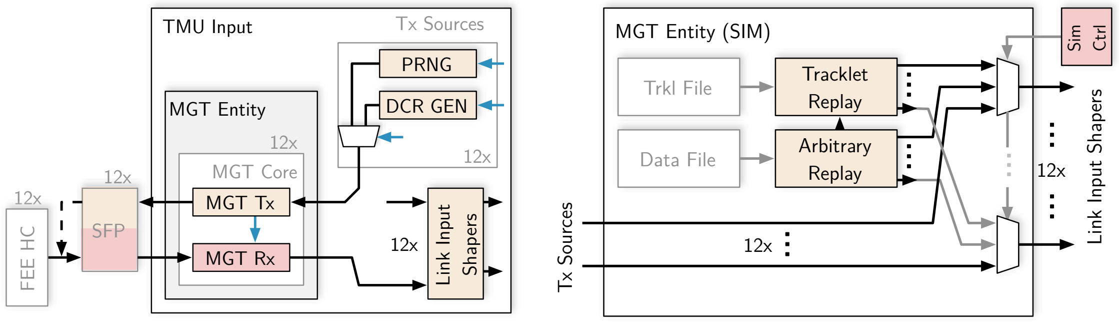
- Tracklet Generator
  The tracklet generator replays tracklet sets from file (extracted from recorded events).
- Arbitrary Event Generator
  The arbitrary event generator is a versatile generator to replay 12 link data streams of arbitrary length, whose content and exact timing is specified in a text file. Cycle-accurate, user-specified streams are particularly useful to replicate irregular half-chamber behavior and harden the GTU data handling.

All generators operate on an event basis and are steered via the trigger sequences issued by the segment control unit. Selection of a generator for the simulation and the corresponding generator parameters are specified in the simulation configuration file.

SMU Board Model Alongside the custom FPGA design, SRAM model and important on-board components such as oscillators, the SMU board model contains mock-ups providing the relevant functionality of the DCS board and, in Run1, the SIU.

The DCS board mock-up implements the receive-side interface to the trigger system as well as the serial output and LHC clock recovery functionality of the TTC receiver circuit. It mimics the time-division multiplexing of the two TTC channels to provide the input expected by the FPGA designs.

[4]The process of replacing is scripted and does not require changes to the TMU source code. It is achieved by loading an alternate implementation of the MGT entity within the simulation tool.



(a) TMU input stage **(b)** Simulation MGT entity

Figure 6.2: Each TMU input stage (left) is equipped with a PRNG and DCR-attached event generator. Their operation can be controlled via PPC. For the simulation, the entity containing the MGT instances is replaced with an alternate entity (right), that additionally allows to select file-based replay generators as link data input source.

The SIU mock-up receives data from the SMU by implementing the communication interface described in [101]. Received data is fed to a log entity to write the event fragment data stream. The mock-up exposes the DDL backpressure signal. Several backpressure patterns[5] can be applied to `siu_ddl_backpressure`, allowing to test the full flow control within a segment up to the suppression of new trigger sequences in different load scenarios.

TGU Board Model The TGU board model includes the custom FPGA design and models for the DCS board, SRAM and the CTP interface board. The global GTU `busy` signal is connected to the trigger emulator to establish the busy feedback loop.

6.1.3 Trigger Stimuli with Feedback Loop

In order to efficiently develop the event handling with buffering of multiple events and verify the system behavior under both normal and abnormal operating conditions, suitable trigger stimuli are required. At first glance, a solution would be to define encoded, serial TTC channel A and B traffic to represent a range of normal and erroneous running scenarios, replay it and feed it into the SMU and TGU board models. However, this approach is too simplistic as the `busy` signal feeds back and potentially delays the start of a new trigger sequence. The `busy` in turn depends on the transmission time of the FEE link with the largest data volume and the current fill state of the event buffers. As a consequence, different input data sizes or trigger timing would require trigger stimuli to be tuned to the specific test. In other words, while static trigger stimuli might represent correct trigger sequences for some data sizes, they might be illegal for others since the start of new sequences is not allowed when the system is `busy`.

As the approach with static trigger stimuli introduces new error sources and is not suited to reproduce realistic system behavior combined with sufficient flexibility for an easy-to-use simulation environment, the GTU test bench features a dedicated trigger emulation entity.

TTC Partition Emulation The emulation of the TTC partition mimics interface and functionality of the ALICE trigger system. This includes trigger timing, busy feedback, trigger sequence structure and encoding. It allows to verify not only the correct functioning of the trigger reception, but also the correct reaction of the full GTU setup to both legal and illegal trigger sequences.

The synthesizable emulator design comprises two major blocks: An emulator core with LTU-like functionality and an entity that implements the combined, required functionality of TTCvi and TTCex, in particular inhibit intervals and TTC frame prioritization on channel B. All communication mechanisms inside the partition are intentionally modeled after the existing TRD TTC partition to guarantee a most realistic trigger timing. The trigger sequences to replay are described in `.xml` format. A utility converts the sequence

[5]The patterns currently implemented are: ‘no backpressure’, ‘light’, ‘toggling’ and ‘heavy’.

description to a binary format used to initialize the emulator core sequence memory at the start of the simulation.[6]

Emulator Operation The emulator core, depicted in Figure 6.3, uses a 48-bit counter as timing reference with respect to the start of the emulation and operates with absolute time stamps. A record stored in the sequence memory contains the sequence type, its start time and, depending on the type, timings, content and modifier flags of the individual events within the sequence. Having fetched such a record from memory, the emulator attempts to start the sequence at the given start time. If at that point `busy` is active, the start of the current and all following sequences is delayed until `busy` becomes inactive. With the start of a sequence, a L0 trigger is issued and entries for remaining events in this trigger sequence, which contain adjusted timing information to retain the sequence-internal timing specified by the user, are pushed to the individual event queues. The transmission of corresponding events as triggers or trigger messages is initiated once the specified simulation time is reached. The sequence memory can hold up to 128 sequence records. The automatic restart of an emulation cycle after the last sequence and the possibility to randomly skip the execution of messages enable extensive tests to cover a wide variety of potential trigger sequence successions and their respective timings.

The TTCvi/TTCex mock-up produces IAC message frames with hamming encoding from L1 and L2 trigger message words transmitted by the emulator core with due regard to L1-over-L2 message prioritization. The mock-up furthermore implements inhibited message transmissions and prioritized transmissions of the BRC bunch counter reset frames. Bandwidth limitations and flow control mechanisms for transfers within the trigger partition's VME crate are accounted for to reproduce real-life timing behavior. The emulator core and TTCvi/TTCex mock-up output the two TTC channels A and B in serial form for direct connection to the simulation's DCS board model.

Stimuli Specification and Error Injection The `.xml` format used to specify the trigger stimuli, see Appendix D.1 for an example, specifies the timing, structure and content of a regular trigger sequence in its default section. If not overridden, corresponding values are applied to all sequences. The sequence list describes the actual sequential arrangement of trigger sequences to produce.

Although the LTU offers an error-on-demand feature capable of provoking certain trigger-related errors [69], its efficient use during development proves complicated. Long design compilation times, mostly ILA-based debugging and the overhead of moving to the physical test system significantly slow down development. To allow for extensive, low-overhead testing against trigger-related errors already in simulation, the emulator is capable of manipulating sequences in a controlled fashion. These manipulations include the incidence and relative timing of trigger events within the given sequence, the disregard of the `busy`

[6] An Altera version of the emulator is available, which can be synthesized and replace the original firmware on the LTU board for system debugging. In this case, the content of the sequence memory is loaded via the VME crate controller board.

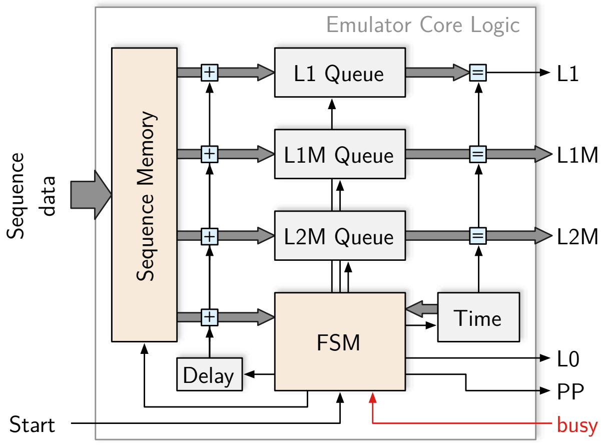


Figure 6.3: The logic of the emulator core. Depending on the settings for a given sequence, the emulator may or may not take into account the GTU feedback via the `busy` signal to potentially delay the start of this new trigger sequences.

signal status at sequence start, and more advanced features like the suppression of individual message words or manipulation of the word content for trigger messages. They have been used to successfully reproduce illegal LTU behavior observed in the field.

The event ID fields of a trigger message, which provide a 36-bit bit identification of the sequence,[7] require special attention due to the sequence delay mechanism. The default behavior is to automatically insert the current values of the emulator's internal bunch and orbit counters as global, unique event ID. The behavior can be overridden by explicitly specifying bunch and orbit count values to test the detection of mismatches between the global and local event IDs.

Stimuli files representing normal operating conditions at different input rates or reject ratios, as well as stimuli files producing the error conditions specified in Table 3.2, are available in the GTU repository.

6.2 Firmware Testing and Commissioning

A common problem when testing complex readout systems is the question how to verify efficiently and on a large scale that a system is working as expected. In case of the GTU, constant production data taking and therefore extremely limited availability for full scale testing further complicates the situation. In the past, firmware updates, which implemented fixes or brought new trigger functionality with accompanying changes to the data path, were frequent even during data taking periods. The timeframe available for deployment of new designs typically is in the order of few hours.

[7]The event ID is a concatenation of 24-bit orbit count and 12-bit bunch count.

To address these issues, the GTU uses a staged commissioning approach. A successful system simulation is followed first by extensive tests with a test setup, before moving to the final commissioning on the ALICE production setup. An overview of the setups and their capabilities is shown in Figure 6.4.

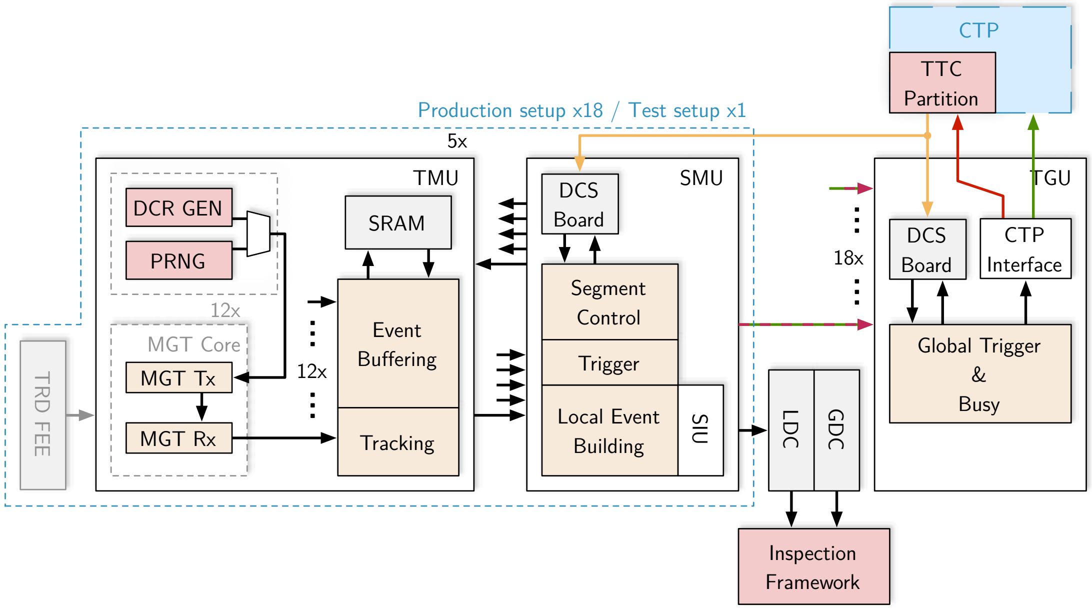


Figure 6.4: Production setup and test setup (no CTP, one LDC/GDC) of the GTU. Each TMU features 12 PRNGs, which can be configured as input source via an electrical loopback in the MGTs. In combination with the inspection framework they allow for efficient, large scale verification of the GTU data handling. The test setup comprises one complete segment plus TGU and uses a trigger generator provided by the LTU. Global runs in the production setup are required for evaluation of the trigger contributions.

The test setup is used to verify GTU firmware or software upgrades and for readout of the supermodules on the surface prior to their installation. It consists of one fully equipped GTU segment plus TGU. Trigger sequences generated using the LTU's emulator are transmitted optically via the TTC system. A computer with integrated DDL receiver card[8] running DATE receives the data and can take the roles of both LDC and GDC. Received data is verified using the inspection framework, see Section 6.2.2. The setup allows to perform standalone runs and test both event handling and trigger calculation, the latter with the restriction that sequences may not be aborted irrespective of the L1 trigger contributions.

6.2.1 Integrated Event Generators

Each TMU is equipped, as indicated in Figure 6.2a, with event generators to provide alternate data streams for each of the 12 individual input links. These integrated generators

[8] A DRORC is used for the Run1 designs with dedicated SIU. As of Run2, a CRORC is used.

facilitate testing of the GTU data path independently of an attached TRD detector module and serve as reliable, easy-to-control data source. Two types of generators, which address different demands, are available:

- DCR-attached Memory Event Generator
  The generator is attached to the PPC via DCR and replays user-supplied data stored in BRAMs. Restrictions regarding the available BRAMs in the TMU limit the memory per link to 8 kbit. The generator is primarily used to deterministically inject meaningful sets of tracklets into the system in order to verify tracking and trigger calculations in the test setup without detector and collider-provided beam.

- Pseudorandom Number Generator
  The PRNG delivers pseudo-random link data generated by a linear feedback shift register.[9] A prepended header includes the generator start count as sequence number and the initial shift register seed. In terms of data word output timing and endmarkers, the generator reproduces the expected half-chamber chamber behavior. The amount of data words generated is configurable on a link basis to link data sizes of up to approximately 2 MB per event, which covers all relevant sizes. The PRNG is complemented by an automated real-time inspection on the receiving computer, see Section 6.2.2.

Both types of generators operate on an event basis and can deliver data at the maximum effective link input bandwidth of 2 Gbit/s. They are started when the segment control unit signals the start of a new trigger sequence. Sequence aborts, i. e. L1 rejects, move the generators to their idle state. To test as much of the GTU-internal data handling as possible, the generators feed the transmit side on the MGT blocks. Data then passes the PCS and PMA sublayers of the MGTs and is, after serialization, fed back to the receivers in a serial loopback [115] just before the MGT output drivers. On the receive side, data is deserialized and handled in exactly the same way as data received from the detector.

The generators can be selected as alternative input to the data path in an ongoing run without restart, i. e., potential reconfiguration. Finding out that certain readout errors suddenly do not occur anymore when generators are enabled as data source often hints to problems that are not caused by the GTU.

6.2.2 Data Inspection

With data rates of several hundred MB/s from a single segment, it is evident that manual inspection and verification does not suffice. For verification with large quantities of event data at high rates, a framework enables the inspection of event fragments or fully built events, see Figure 6.5. It is written in C++ and extensible via light-weight modules.

[9]See [55]. Taps used for maximum length sequences are 16, 15, 13 and 4.

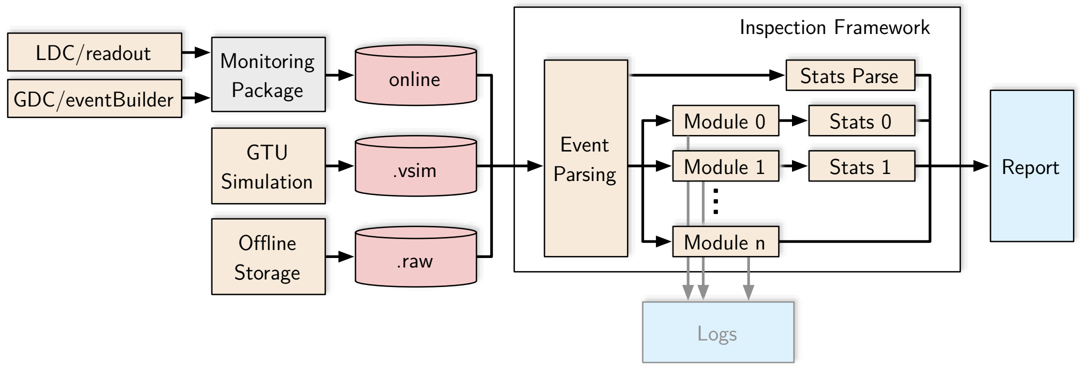


Figure 6.5: The inspection framework, which is extensible via light-weight modules, analyzes the event fragments contained in an event and enables automated checking for errors and data inspection online as well as offline.

Input Sources The framework is designed to work with all input data formats encountered in the design process. During the simulation `.vsim` files are generated, which contain data sent by one segment via a single DDL. Data recorded by DATE for offline analysis is organized chunks of `.raw` files, which contain the events in event or super-event format [12]. The framework can also attach to a DAQ online data producer, which is either the readout process on an LDC or the event builder on a GDC, by using the DATE monitoring package and retrieve raw TRD events via the local network. Several options, like event selection and blocking monitoring may be used to adjust the retrieval process.

Framework Operation The events in the selected input data stream are processed by the framework on basis of individual event fragments assembled by the SMUs. Equipment IDs attributed to the fragments facilitate the selection of data originating at given TRD supermodules. In a first step, the framework attempts to parse selected fragments, i. e., evaluate versions and size fields of all GTU-generated headers and identify all individual link data blocks. A number of sanity checks are performed and give a first assessment of the integrity of an event. Incorrect header sizes or missing words, for example, lead to a failure in parsing of the fragment that is immediately reported to the user. In a second step, modules perform a more detailed inspection or analysis of the data.

Extension via Modules The event parsing stage provides the modules with information about the event structure and access to the event fragment data itself. Using this, the user can craft modules to perform specific tasks, such as the recalculation and comparison of the CRC or inspection of trigger flags. At start-up of the inspection framework, the user selects which modules to activate. The modules optionally feed corresponding statistics modules to summarize the results of the fragment inspection and detected errors over long runs. Important modules used regularly during commissioning are:

- GTU link-level PRNG module
  Counterpart to the integrated link-level PRNG generators. It recalculates the expected data words and performs a bit-level comparison of the link data content.

- Detector module[10]
  Determines the type of transmitted detector data and subsequently operates in one of two modes:

  - Regular mode
    Performs a number of sanity checks on the received data, e. g., checking for half-chamber headers at the expected position.

  - Supermodule PRNG mode
    Performs bit-level checking of supermodule link data content, which allows to, e. g., identify faulty chip interconnects on the ROBs to develop alternate FEE configurations.

- CRC module
  Calculates the stack-level and supermodule-level CRCs and compares them to the hardware CRCs shipped in the event trailer.

- Input event shaper module
  Monitors the data stream for input data shaper activity, i. e., irregular half-chamber behavior.

- Event sequence module[11]
  Relates GTU PRNG sequence numbers, received trigger counts and validated trigger counts to signal incorrect accepts or rejects at the different trigger stages.

6.2.3 Commissioning of the TRD Readout Chain

An efficient commissioning process is important when considering the limited availability of the production system. The inspection framework, in combination with the integrated pattern generators at the GTU input stage and on the FEE itself, allow to verify the correct data flow in the full readout chain on a bit level with high statistics in comparably short time.

First stage tests performed with the test setup use the TMU link-level PRNGs and the corresponding checker module. With the PRNG seed shipped as part of a generator header, the expected link data content is recalculated and compared to the actually shipped data. In parallel the CRC module and, during MEB development, the event sequences module are used. Triggers are supplied by the trigger generator integrated in the LTU and include include L0, L2A and L2R sequences in different ratios and varying chronological succession. Important parameters varied during the tests are:

[10]The module is contributed and maintained by Jochen Klein.
[11]Specifically developed for MEB commissioning, required hardware is now partly omitted.

- Trigger rates
  The rate at which the LTU emulator tries to start a new emulation cycle is varied from few Hz to more than 10 MHz. The latter leads to a start of the next sequence practically immediately after the GTU `busy` is released.

- Event sizes
  Event sizes are varied on link granularity during ongoing tests to check for potential overflow of buffers, illegal memory management, etc. Tested link data sizes range from very small to sizes exceeding black events that trigger timeouts of the input data shapers.

- DAQ backpressure
  The readout process on the LDC is artificially slowed down to make the DDL flow control send a `link_full` to the SIU. In turn, this tests the flow control in the GTU segment up to complete stalls and subsequent resumption of the event readout.

In the second commissioning stage these tests are repeated in the production setup with bit-level checking at the event fragment level on the LDCs, or on a super-event level, i. e., fragments from multiple DDLs, on the GDCs. If no errors are encountered, commissioning continues with the available TRD supermodules as data source. Supermodule PRNG tests are performed prior to tests with regular detector data. The detector module, CRC module and input shaper module of the inspection framework give a quick and direct indication regarding the correct operation of the readout chain.

6.3 Integrated Diagnostics

Although all firmware undergoes the presented thorough testing process before deployment, it is virtually impossible to anticipate and cover all error cases in these tests. The complexity of the system with multiple, interconnected FPGA designs and large number of components makes errors furthermore extremely hard to trace, especially if they only manifest in the full-scale production system or originate in a system the GTU interfaces. As a countermeasure, extensive diagnostics have been developed and embedded in the designs to enable in-system debugging and monitoring.

The diagnostics is customized to the different types of GTU FPGA nodes to provide the relevant information that characterizes both the internal state of a node as well as the interfaces to the external systems in sufficient detail. All diagnostic features are built around the console application. Thus, they can be queried remotely or used in scripts executed by the FSM engine, e. g., to dump a snapshot of the system status at the end of a run. Table 6.1 summarizes the most important diagnostic commands.

Three characteristic examples, which illustrate the different methods for the acquisition of diagnostic values employed in the GTU, are given in the following.

Command	Available on	Description
`smu`	SMU	Status of SMU control logic and data path configuration
`ttc_trace`	SMU, TGU	Chronological history of TTC traffic
`meb`	SMU, TMU	MEB control status: FSM states, buffer occupancy and bandwidth histograms
`busy`	SMU	Busy status, dead time contributions and chronological history
`ddl`	SMU	DDL usage statistics and histograms
`lvds`	SMU, TMU	LVDS backplane status and calibration
`sfp`	SMU, TMU	SFP diagnostics
`mgt`	SMU, TMU	MGT status (rx/tx error counts, locking, etc.)
`ctp`	TGU	CTP interface status and configuration

Table 6.1: Selection of important embedded, top-level diagnostics and monitoring commands. The commands provide an integrated help function showing further options.

6.3.1 Hardware Status Snapshots: Status of FSMs

Certain information is of interest only when the system is in a static state, e. g., when the readout is stalled or no (new) triggers are currently arriving. In these cases, periodically updated snapshots of values are sufficient.

In case of the GTU, relevant information is scattered over the multiple clock domains in the individual node designs. A resource-efficient way to collect multiple values of interest emerging in a certain acquisition clock domain and conveniently handle the crossing into the PPC read clock domain are BRAM-based snapshot memories. With the help of a trivial write logic, essentially consisting of an address counter and a multiplexer, the quantities in memory are updated in a round-robin fashion, which results in a snapshot of a quantity every n clock cycles. The BRAM read side is attached to the PPC system via the PLB BRAM interface for extraction and processing of the values.

One of the many examples in which these snapshots provide invaluable information about the state of the system are the state vectors of the FSMs that control the segment operation and readout-related tasks, see Chapters 3 and 4. In case of an irregularity, at least one of these FSMs typically does not advance to the next state as one would expect. Thus, the current state of each of these FSMs, which are distributed over all six FPGA nodes in a segment, combined with their current input vectors (often) gives a clear indication as to which part of the design or external entity is causing the problem. A review of the state of the most relevant FSMs related to segment control and readout, listed in Table 6.2, is often the starting point for a thorough error analysis.

FSM	Location	Clock domain
Acquisition handler	SMU	LHC (40.08 MHz)
▷ Event shaping	TMUs	Input (125 MHz)
Readout initiator	SMU	LHC
▷ Main event dispatcher	SMU	Readout (60 MHz Run1, 120 MHz Run2)
▷ Tracking readout	SMU	Readout
▷ Trigger readout	SMU	Readout
▷ Trailer readout	SMU	Readout
▷ Stack readout initiator	TMUs	SRAM (200 MHz)
▷ Stack readout acceptor	TMUs	SRAM

Table 6.2: Most relevant FSMs related to segment control and readout, their interdependencies and location. Corresponding states and input vectors are accessible via the PPC console.

6.3.2 Guaranteed Real-Time Value Extraction: Logging of TTC Traffic

Error-free reception of valid trigger sequences issued by the CTP is crucial for synchronized operation of ALICE and correct functioning of the hardware. In order to clearly distinguish between errors of the GTU control logic itself and externally induced errors, e. g., due to illegal triggering, all segments log the TTC traffic.

The TTC trace hardware is an example for the real-time extraction of diagnostic values that exploits certain characteristics, in this case those of the trigger transmission protocol, to guarantee value extraction without loss. As the chronological sequence of received triggers leading up to an erroneous condition is of interest, it is built around a dual-port BRAM organized as circular buffer that provides an option to halt the logging in case an error is detected, e. g., by the segment controller. Figure 6.6 illustrates the architecture of the trace entity and its connection to the PPC system. The read side of the block memory is attached to the processor bus of the high-level PPC via the PLB BRAM interface component.

The trace write logic ensures logging of all events even in case of simultaneous arrival of events on the two TTC channels. To maximize the logged trigger history with the given amount of memory, the entity does not log continuously, but operates on basis of received individual log events. Three categories are defined: Channel A events (L0, L1 or illegal channel A data), channel B events (received BRC or IAC frames) and information events (complete trigger message detected, etc.). Type, time stamp, local bunch count ID and, if applicable, the content of the decoded TTC channel data are stored. Filters on the write side can be configured to ignore certain message types, such as the periodic BC reset message. In the current implementation employing a 18 Kibit memory, the trace is capable of storing at least[12] 14 complete L2A sequences or up to 512 L0 sequences.

[12]L2A sequences occupy, primarily due to the number of IAC frames, the largest amount of memory. Therefore, they determine the minimum number of complete sequences the trace buffer can hold to 22 with Run1, or 14 with Run2 message formats, respectively. With trigger rejects at the L1 and L2 stages, the number of sequences visible in the trace history is much larger in practice.

Listing 1 shows a trace extracted via the console application. First and second column identify the origin of the event and its relative arrival time with respect to the previous log entry (or trace start) in LHC cycles up to a maximum spacing of $2^{17} \cdot 24.95\,\text{ns} = 3.27\,\text{ms}$. Larger times between trigger events are of no importance considering the given trigger timings. The remaining columns contain the local bunch count ID at the time the log entry was stored and the identified event type. In case of channel B transmissions, the payload is analyzed to identify known trigger message words (L1 message header word, etc.) and the word data content.

6.3.3 Occurrence-Driven Statistics Acquisition: Dead Time Accounting

The GTU has exclusive access to certain quantities like the duration of the FEE-GTU transmission or the readout duration of an event. With a measurement and synchronized extraction of matching values on an event-by-event basis, their accumulation and post-processing, it is possible to implement an in-system characterization of the current and longer-term performance that helps to spot potential bottlenecks live.

Due to the trigger-driven readout of the system, the values of interest, which often are counters that are reset for the next sequence of processing cycle, are valid at certain distinct points in time in a trigger sequence. A these points in time, a matching set of characterizing values needs to be extracted. To analyze the performance over macroscopic periods, the event-by-event quantities furthermore need to be accumulated and post-processed to calculate the desired (period) averages. As the acquisition and processing of these values solely in hardware is resource-intensive and offers only very limited flexibility, the GTU node designs employ a combination of occurrence-driven, synchronized acquisition in hardware and real-time PPC-based accumulation and post-processing.

The general layout of a hardware unit handling the acquisition for a given domain and its connection to the PPC system is depicted in Figure 6.7. At a defined moment, quantities are captured in shadow registers and subsequently pushed, as a matching set, to a statistics buffer. The buffer can hold a configurable number of sets and is connected on the read side to the low-level PPC by a 32-bit data-side memory bus at 100 MHz. This PPC core is specifically set up to handle interrupt requests with deterministic timing in quasi real time.[13] The buffer requests processing of all available sets via a priority interrupt once the number of sets reaches a certain configurable threshold or, in case of very low occurrence rates, when a timer fires. An overflow flag indicates exceptional cases, e. g., where the PPC does not manage to cope with the acquisition rate. The interrupt routines are tailored to generally meet the processing time requirements determined by the occurrence rates of the associated events, which typically range from some 10 kHz for measurements related to the L0-L1 interval to few kHz for readout-related measurements. The routines process all

[13]The compact program code of the low-level PPC and interrupt vector table reside in a BRAM connected via a 64-bit instruction-side memory bus at 200 MHz. Two instructions are fetched per cycle, which, in combination with a linear program flow, minimizes wait cycles due to instruction fetches. Highly frequented data memory sections, especially uninitialized data, stack and heap, are limited to and located in the 16 kB data-side cache to allow for full-speed access. See [99, 77] for further details.

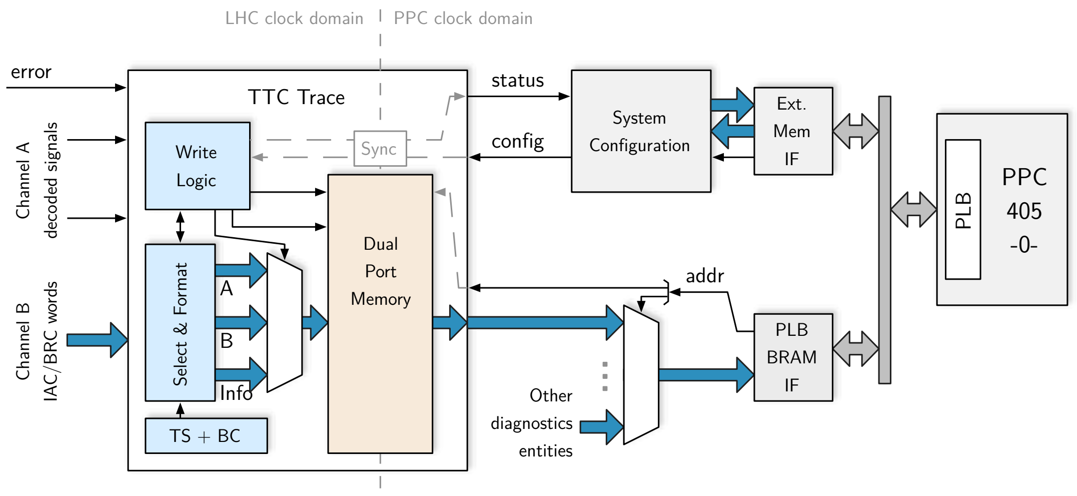


Figure 6.6: TTC trace entity on the SMU and TGU nodes, and its attachment to the PPC system.

```
smu00:/ > gtucom 2 ttc_trace show 100
ChA  124555 0x4AD   L0
ChA  000259 0x5B1   L1
ChB  000055 0x5E9 f(IAC) h(L1H ) d(0x1C0)
ChB  000043 0x615 f(IAC) h(L1D ) d(0x000)
ChB  000043 0x641 f(IAC) h(L1D ) d(0x000)
ChB  000043 0x66D f(IAC) h(L1D ) d(0x000)
ChB  000043 0x699 f(IAC) h(L1D ) d(0x000)
Info        0x69A   L1 message
ChB  003772 0x76A f(IAC) h(L2AH) d(0x58D)
ChB  000043 0x796 f(IAC) h(L2AD) d(0xC13)
ChB  000043 0x7C2 f(IAC) h(L2AD) d(0x149)
ChB  000043 0x7EE f(IAC) h(L2AD) d(0x000)
ChA  000010 0x7F9   L0
ChB  000032 0x81A f(IAC) h(L2AD) d(0x800)
ChB  000043 0x846 f(IAC) h(L2AD) d(0x000)
ChB  000043 0x872 f(IAC) h(L2AD) d(0x000)
ChB  000043 0x89E f(IAC) h(L2AD) d(0x000)
Info        0x89F   L2a message
ChA  131072 0x22D   L0
```

Listing 1: Snapshot of a L2A trigger sequence and two L0 as seen by the built-in TTC trace. The trace history allows to identify the transmission channel, relative event timing, local event ID, type and potentially the message content of TTC events.

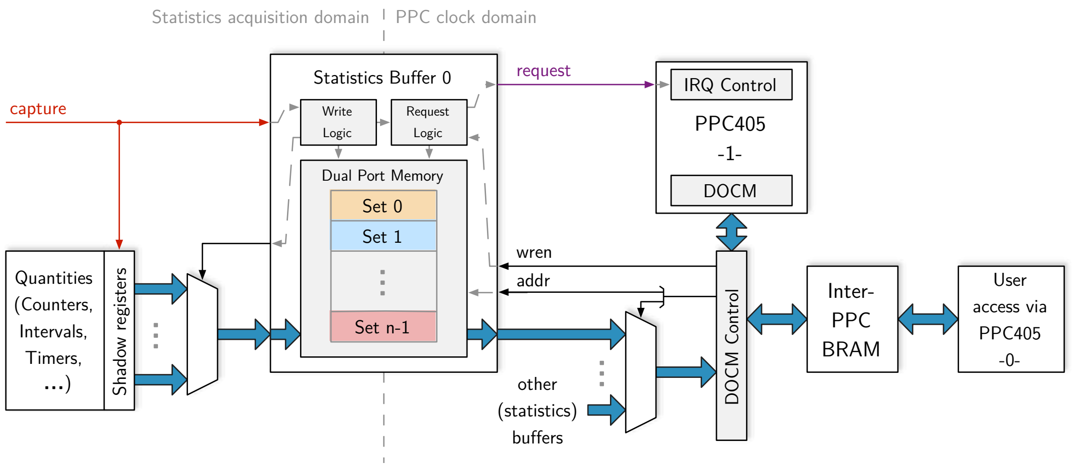


Figure 6.7: Buffer for the occurrence-driven collection of interesting quantities. A set of values is captured at a given time and subsequently pushed to the buffer. Based on fill level or time, the buffer requests readout of its sets via priority interrupt to the low-level PPC, where long-time accumulation and post-processing of the measured values is performed.

available sets in the buffer and accumulate numbers over longer periods, e. g., the entire duration of a run. Less-prioritized timer interrupt routines are employed to calculate period averages or long-term statistics. The results are placed in the memory connecting both PPC cores, where they are externally accessible.

The combination of event-based statistics acquisition in hardware with post-processing in software offers a high degree of flexibility at reasonable resource usage.[14] Examples where the occurrence-driven statistics acquisition is used are the monitoring of the DDL, see Appendix C.2, as well as the dead time monitoring in the SMU. The latter implicitly characterizes the raw data transmission from the FEE and the current load on the readout system.

Listing 2 presents an example for the dead time statistics taken in a run with the test setup and pattern generators as the data source. The individual contributions to the resulting dead time are measured for each trigger sequence. A detailed account for the last eight sequences is presented to quickly identify recurring problems.[15] By breaking down the FEE-GTU transmission duration into contributions of the individual stacks, increased stack noise or problematic detector half-chambers can immediately be identified. The longer-term measurements immediately show the system performance in the context of the current readout partition and enable the identification of bottlenecks. Histograms of event

[14] Although some fabric resources are required as shadow registers, it is now possible to tailor the width of the basic measurement counters (and thus shadow registers) to the short time interval of interest. The 64-bit types in the PPC allow to conveniently accumulate and process values over large time spans at the tolerable expense of some block memory.

[15] These measurements exhibit minor errors due to necessary clock domain crossings and also due to the restriction to integer arithmetic instead of floating-point emulation in the PPC for size/resource reasons. For the purpose of online sanity evaluation, these errors are negligible.

buffer usage and dead time contributions, such as shown in Listing 3 for the dead time due to the FEE-GTU transmission, complete the measurements.

Summary

To support ongoing development of the complex GTU designs and facilitate commissioning of new firmware versions even on short timeframes like LHC fill pauses, a thorough verification process is needed.

The developed test and commissioning process consists of several stages. Initially, new developments are included in a system-level functional simulation with automated output verification. For external components of the simulation framework, e. g., the input links or trigger input, simulation models have been developed and refined, so that they realistically reflect all relevant aspects of their real-life counterparts. Having successfully passed the simulation stage, the designs are exercised in the test setup under varying conditions in terms of trigger sequences, backpressure and rates. Bit-level data stream verification is performed online on large amounts of data, typically several TB, by using the developed validation framework. Successful tests with the test setup are scaled up to the production system and repeated in the final commissioning phase. As a result of the through testing process, although firmware updates were frequent, only very few occurrences of newly deployed designs exhibiting problems were encountered in production data taking over the lifetime of the GTU.

As the central part of a complex detector system, the GTU implements diagnostics and monitoring of all relevant internal status information and of the interfaces to external systems. It uses the PPC system not only to access diagnostics data, but also extensively in the process of acquiring values, as this provides the possibility of flexible post-processing and calculation of informative quantities that characterize the system. The extensive embedded diagnostics and live monitoring capabilities of the GTU, especially those of the interfaces, have proven invaluable on many occasions not only for the GTU, but for the full detector system. With the possibility to conduct a rapid, yet detailed in-system analysis of complex irregular situations and gain an understanding of the underlying problem, it was possible to develop the system into the stably operating detector readout and trigger chain it is today.

```
smu00:/ > gtucom 2 busy stats
Dead time for last 8 seq
  Seq    t(L0-L1) [us]      t(L1-RxDone) [us]     t(MEB) [ms]
   0:        6.49              47.932               0.000
  -1:        6.50               0.000               0.000
  -2:        6.50               0.000               0.000
  -3:        6.49              47.932               0.311
  -4:        6.50               0.000               0.000
  -5:        6.49              48.007               0.068
  -6:        6.50               0.000               0.000
  -7:        6.48              47.940               0.000

FEE Rx per stack for last 8 seq [us]
   0:     2.365 (S4)      47.716 (S3)      2.357 (S2)      2.373 (S1)      47.783 (S0)
  -1:     0.000 (S4)       0.000 (S3)      0.000 (S2)      0.000 (S1)       0.000 (S0)
  -2:     0.000 (S4)       0.000 (S3)      0.000 (S2)      0.000 (S1)       0.000 (S0)
  -3:     2.373 (S4)      47.716 (S3)      2.307 (S2)      2.365 (S1)      47.783 (S0)
  -4:     0.000 (S4)       0.000 (S3)      0.000 (S2)      0.000 (S1)       0.000 (S0)
  -5:     2.365 (S4)      47.716 (S3)      2.307 (S2)      2.365 (S1)      47.841 (S0)
  -6:     0.000 (S4)       0.000 (S3)      0.000 (S2)      0.000 (S1)       0.000 (S0)
  -7:     2.373 (S4)      47.675 (S3)      2.357 (S2)      2.382 (S1)      47.791 (S0)

Avg dead time   per L0      per L1a
 L0-L1 [us]:     6.490
 FEERx [us]:    23.962      47.940
 Rdout [ms]:     0.008       0.017

Avg dead time
 Averaged over last 5.14 s
 L0-L1     1.6 %
 FEERx     5.9 %
 Rdout     2.1 %
 Total     9.7 %
```

Listing 2: Snapshot of the integrated measurement of the contributions to a segment's dead time and long-term accumulation of values. This snapshot shows data taken in the test setup with pattern generators, in this case configured to deliver large and very small events per stack, as the data source.

```
alidcsdcb0610:/ > gtucom 2 busy stats -hrx
Dead-time by Tx FEE->GTU [us]
     0:       268  | X
    10:         0  |
    20:     53019  | XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    30:      6199  | XXXXXX
    40:      2604  | XXX
    50:      1935  | XX
    60:       913  | X
[...]
   180:         2  | X
   190:         0  |
```

Listing 3: Excerpt of the PPC-based live histogramming of the FEE-GTU transmission duration taken from a p-p run. Entries in the lowest bin typically point to rejects at the L1 stage, entries in the highest bins (may) point to irregular half-chamber behavior.

7 System Characterization and Performance

When several ALICE subdetectors take data in unison as part of a readout partition, the system exhibiting the largest dead time typically governs the overall performance. To minimize the dead time and maximize the physics output, a thorough understanding of the dead time generated by processing stages of individual subdetectors is required.

To assess the dead time behavior of the GTU, the following chapter first characterizes the input from the FEE and the implications for the processing time of the L0 stage. A detailed look on the data transport through the system and an analysis of the readout interface reveals the L2 stage processing times. These results form the basis for a modular TRD readout model, which can simulate the behavior of individual detector segments as well as the full TRD. The model allows to evaluate the TRD readout performance standalone and, with the help of additional detector models, also in the context with other detectors as part of a global readout partition. The chapter concludes with a review of the GTU operation in Run1 and the prospects for Run2.

7.1 Data Transfers from FEE to GTU

From the start of a trigger sequence until the end of the L1 trigger processing, a supermodule is busy with the calculation of tracklets and potential transfer of raw data to the GTU. Potential influences of tracklet and raw data transmissions on dead time are quantified in the following.

7.1.1 Tracklet Transmission

Parameterized track segments calculated on the FEE are shipped to the GTU prior to the L1 decision interval within the timespan t_{L0}, see Section 3.3. In this interval the SMU anyway raises its busy signal to stop new trigger sequences, therefore no special treatment of tracklet-induced dead time in the L0-L1 timeframe is required.

If data is written to the SRAM for subsequent readout, the available SRAM read bandwidth is reduced. The effect, which is a consequence of the push architecture and strict write prioritization, is discussed in Section 7.1.3 for the transmission of raw data. In principle, this could prolong an ongoing event readout and produce additional dead time. However, due to the small transmission time for tracklets,[1] the write hysteresis in the SRAM controller and the depth of data path buffers in TMU and SMU cancel out any effect.

[1]Detailed studies related to the number of expected tracklets and arrival timing may be found in [99].

7.1.2 Raw data Transmission

On reception of a L1A, raw event data is transferred from filter memories on the FEE to the event buffers in the TMUs. The raw data transfer time t_{tx} directly effects the dead time as `busy` is released only after all event data was transferred to the GTU.

To characterize the data transfer behavior of the FEE and gain credible numbers for the transmission bandwidths and transfer time as function of the link data volume, series of tests runs were conducted, see Figure 7.1. A short description of these runs may be found in Appendix C.1. The underlying timing information for the plots shown was acquired with GTU firmware designs modified to inject time stamps for certain system events[2] into the data stream for subsequent analysis offline. Event fragments recorded with standard production GTU firmware designs do not contain these time stamps for reasons of event size and long-term storage cost.[3]

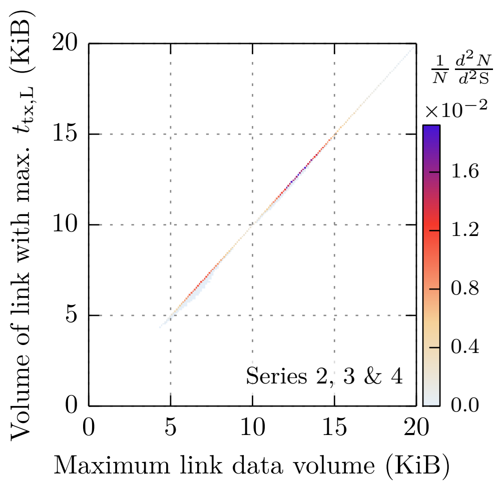
Figure 7.1a shows the data volume of the link with maximum transmission duration per supermodule as function of the maximum link data volume of all links of the supermodule for all events in the corresponding runs. In the presented plot, 96.7 % of the entries are on the diagonal. This means that the link with maximum volume typically also exhibits the largest transmission time, and therefore dominates the segment's contribution to the dead time associated with the raw data transfer t_{tx}. The few entries off the diagonal correspond to links that exhibit slightly less than maximum volume, but show the longest transmission times. This is caused by larger transport delays due to the distribution of data on the MCMs. As one would expect, this effect is more distinct for small link data volumes. In heavy-ion-like, high-occupancy events with homogeneous distribution of data across the half-chamber, the effect hardly visible.

Figure 7.1b shows the arrival of link data endmarkers with respect to the L1A as function of the data volume transmitted on the link. The two distinct bands visible correspond to links that originate at chambers of either the C0 or C1 type. The latter exhibits an increased transmission time due to the additional ROB branch, see Figure 2.4. At given link volumes, variations of the transmission time within the bands can be attributed to the distribution of data on the MCMs in the half-chamber and associated delays introduced by the handshake readout protocol. The dashed blue line represents a fit for the average transmission duration of C1-type chambers distribution across a half-chamber. The slope m corresponds to a data rate of 1.923 Gbit/s and agrees well with the rate at which the HCMs feed data to the ORIs, i. e., 16 bit at 120.23 MHz. It also resembles the peak data rate to expect from a half-chamber. The offset c amounts to the time until arrival of the first data word plus potential idle cycles in the transmission.

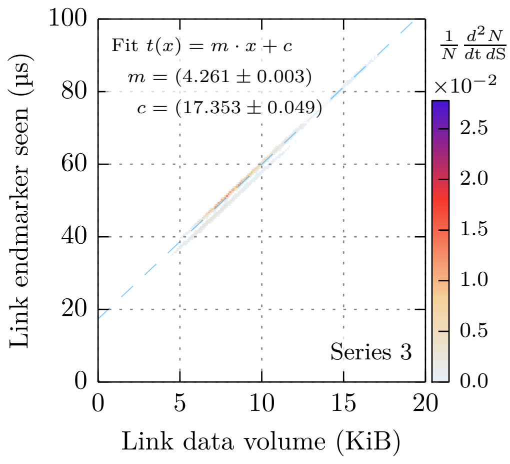
Figures 7.1c and 7.1d show details of the first word arrival and transmission idle cycles. The first words from the detector typically arrive at the GTU around 1.7–1.8 µs after the

[2]These events are, for example, arrival of the endmarker on link, setting and releasing the segment busy, start of readout to the DAQ or its finalization.

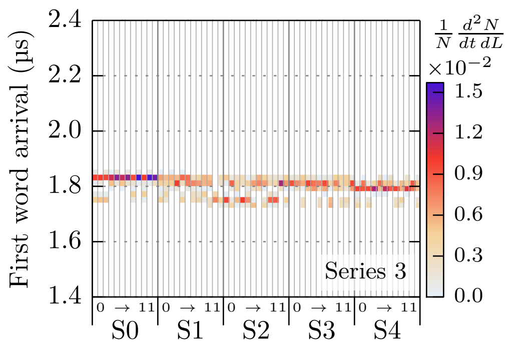
[3]However, certain important measurements, e. g., the transmission duration of a stack, are also accessible, with reduced statistics, in-system via the PPC console and the previously presented statistics measurements. See Section 6.3.3.



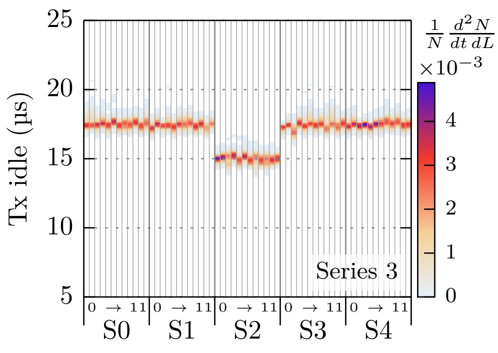
(a) Dominating contribution to the raw data transfer time t_{tx}



(b) FEE transfer duration



(c) Time until first data word (grouped by stack and link index)



(d) Accumulated idle cycles (grouped by stack and link index)

Figure 7.1: Characterization of the raw data transfer from FEE to GTU, measured in series of test runs. All times are measured with respect to the arrival of the L1 trigger on the TMUs. See text for a detailed description.

L1. The accumulated number of cycles without a valid data word transmitted between first word and endmarker arrival amounts to typically 17–18 µs for C1-type, and 15–16 µs for C0-type chambers. Idle cycles can be attributed to the preparation of data for readout, the handshaked MCM raw data protocol [104], and latency for (de-)serialization and optical transmission.

7.1.3 SRAM Read/Write Bandwidth

The SRAM controller on each TMU presents a 128-bit interface at 200 MHz to the event handling design, which results in a maximum bandwidth of 25.6 Gbit/s shared between reads and writes. The write prioritization reduces the available read bandwidth during an ongoing transfer of data from FEE to the event buffers. If reduced below the level required to saturate the DDL, it could prolong an ongoing readout of an event and potentially contribute additional readout-related dead time in cases where the system operates at a workload close to $x = 1$.

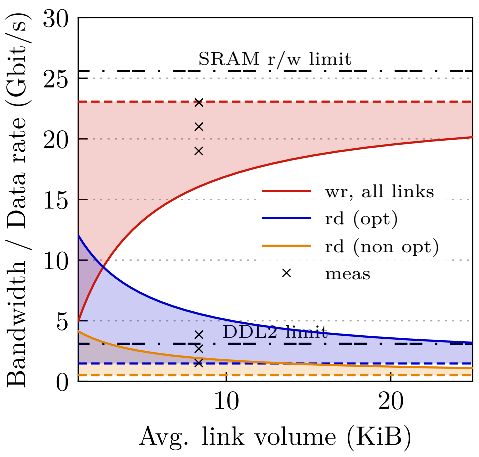


Figure 7.2: Bandwidths for SRAM write and read, with a non-optimized (Run1) and optimized (Run2) SRAM controller.

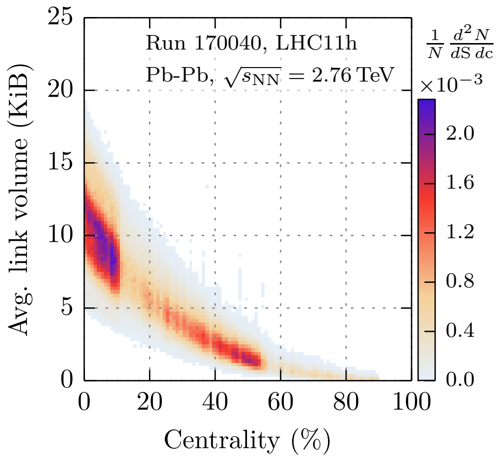


Figure 7.3: Link data volume averaged per detector stack as function of the event centrality in Pb-Pb (2011).

Based on the observations of the FEE transmission behavior presented in Section 7.1.2, best and worst case assumptions for the SRAM write data rate and the resulting read data rate are summarized in Figure 7.2. Assuming that all transmission idles essentially occur in a large block and all links of a stack deliver data at the maximum rate, a total input rate of 23 Gbit/s per stack has to be expected. This maximum SRAM write rate is shown as dashed red line. The minimum write rate, which assumes that idle cycles are homogeneously interspersed with data words, is represented by the red solid line. The real write rate, which depends on the number and distribution of idle cycles in the event data transmission, lies somewhere in the shaded area in-between these two lines.

The structure and switching of a half-chamber readout tree implies that idle cycles, or smaller idle periods, are spread out between first word and endmarker, thus hinting at an actual write rate that is considerably lower than the maximum. Figure 7.3 shows the link data volume averaged per detector stack as function of the event centrality, measured in a Pb-Pb run at $\sqrt{s_{NN}} = 2.76\,\text{TeV}$. For central (0–20 %) heavy ion events, the majority of average link volumes observed is in the range 5–15 KiB. Consulting Figure 7.2, this translates to write rates per stack in the range of 15–21 Gbit/s that are seen in practice.

The available read bandwidths corresponding to worst and best case are shown as dashed and solid orange lines for the basic SRAM controller used throughout Run1, and as dashed and solid blue lines for the optimized controller of Run2. It is assumed that whenever the controller enters the `read` state, write requests immediately arrive. The optimized controller version with write hysteresis eliminates controller inefficiencies, see Section 4.7.3, and shows significantly improved read data rates. It delivers at least 1.48 Gbit/s, even under worst case conditions.

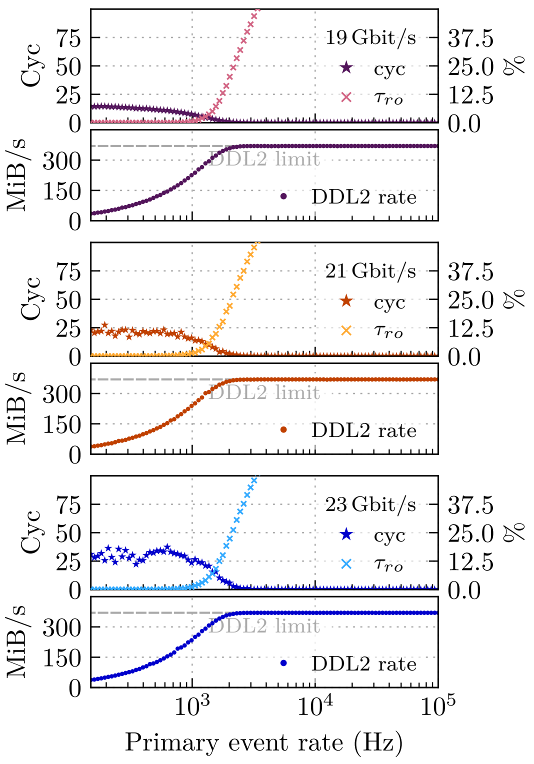
To bridge periods in which the read rate potentially drops below the rate needed to fully saturate the DDL2, buffers are placed in the data path. On the SMU a 256 Kibit buffer decouples the readout logic from the SIU interface. At the maximum rate difference between DDL2 and SRAM read of approximately 1.63 Gbit/s, it ensures that for a period of 160 µs the full effective DDL2 output rate can be fed into the SIU, even if continuously disturbed by ongoing transfers from the FEE.[4] If fully depleted, it replenishes within 40 µs of normal readout. Buffers on the TMUs may contribute up to another 60 µs of coverage depending on their fill state.

Figure 7.4 visualizes data path inefficiencies for Run2 designs. It shows the average number of cycles per event in which no data is input to the SIU, although readout of an event has been started and the SIU is ready to send. A small number of such 'unused' cycles, typically 16–18 per event, are attributed to the initiation of a readout and the time required to fetch event data from the TMUs. Any excess cycles point to inefficiencies in the data path, e. g., insufficient SRAM read bandwidth. The measurements, shown for different trigger ratios and different FEE input data rates,[5] are acquired with the test setup, see Appendix C.2 for details.

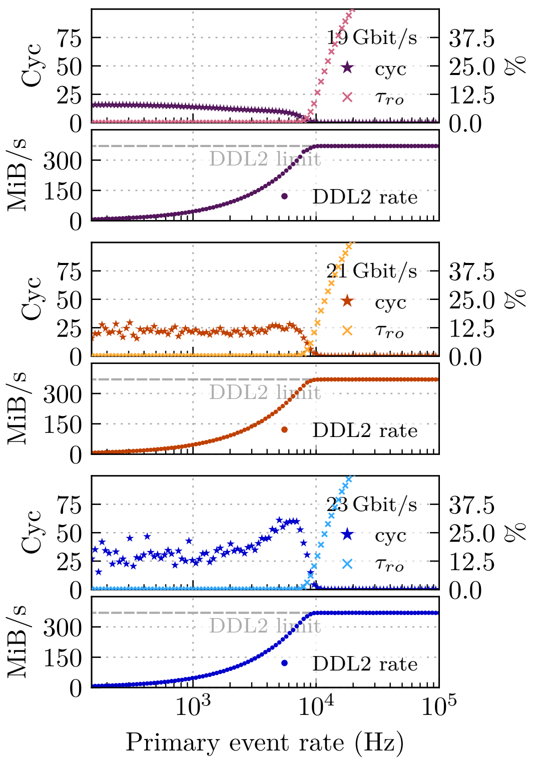
Interruption of SRAM reads by data transfers from the FEE and an increase in unused cycles, i. e., prolongation of the event readout time, is observable for input rates around 20 Gbit/s and up. For low primary event rates, there is a high chance that the decoupling FIFO does not compensate for the drop in SRAM read bandwidth, because it is empty at the time of the FEE transfer. In such a case, the ongoing readout of an event is slightly prolonged, however, it does not affect the dead time. As soon as there are almost constantly events being transferred to the DAQ and the readout-related

[4]These values compare to the duration of the FEE transmission of approximately 100 µs for a zero-suppressed central Pb-Pb event.

[5]Markers 'meas' in Figure 7.2 denote the target write and corresponding SRAM read bandwidths of the measurement presented in Figure 7.4.



(a) Trigger ratios:
$a_{L1} = 0.5$, $a_{L2} = 1.0$
Event fragment size ≈ 500 KiB



(b) Trigger ratios:
$a_{L1} = 0.1$, $a_{L2} = 0.75$
Event fragment size ≈ 500 KiB

Figure 7.4: Average number of 'unused' cycles into the SIU interface per event in relation to the readout-related dead time τ_{ro} and the outgoing data rate to the DAQ. The measurements are shown for two different trigger scenarios (left, right) and different input rates per detector stack of 19 (violet), 21 (orange) and 23 Gbit/s (blue).

dead time τ_{ro} rises due to increased system load, the FIFO compensates for the difference in DDL2 and SRAM read bandwidth and keeps the utilization of the link at maximum. The combination of improved controller and buffers in the data path suffices to eliminate any side effect of SRAM writes on the readout-related dead time τ_{ro} in this system.

7.2 DDL Readout Interface

If a segment's data path is capable of completely saturating the input to the SIU, effective output bandwidth and event fragment size directly translate to the time required to process an event at the L2 stage. The effective bandwidths of the DDL interfaces for Run1 and Run2 are therefore quantified in the following.

Bandwidth Limits The bandwidth limits of a segment's readout link are summarized in Figure 7.5 in terms of transceiver line rate and input interface bandwidth for Run1 and Run2. The SIU add-on card used in Run1 utilizes a transceiver at 2.125 Gbit/s. Taking into account the overhead for 8b10b encoding and SIU framing yields an maximum DDL rate of 196.52 MiB/s (blue, dashed).[6] The add-on card is interfaced by the SMU FPGA via a 32-bit wide data bus at a maximum operation frequency of 50 MHz, which limits the usable rate to 190.73 MiB/s (light blue, dashed). Requirements of the DDL event transmission transaction protocol, such as a gap of at least 16 SIU clock cycles in-between events and sending of the front-end status word, see [101], are encapsulated within the SIU interface logic. The total overhead per event of the GTU implementation determined in simulation is 19 cycles in the SIU clock domain. It limits the event transaction bandwidth (blue, solid) only for very small event fragment sizes. Considering that already p-p event fragments are typically larger than 5 KiB, the full SIU input bandwidth of 190.73 MiB/s is available for all practical applications.

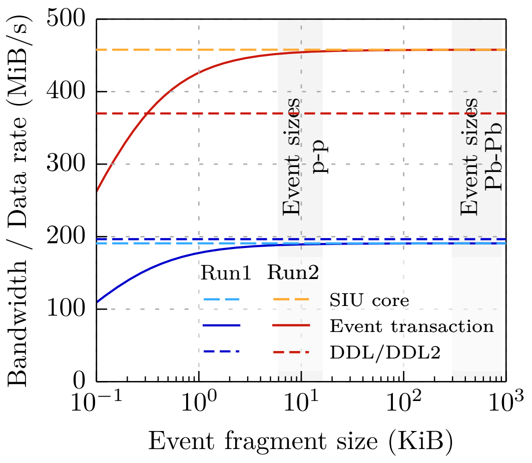


Figure 7.5: Bandwidth limits of the SIU (core), the event transfer and the transceivers function of the event fragment size for Run1 (blue) and Run2 (red/orange).

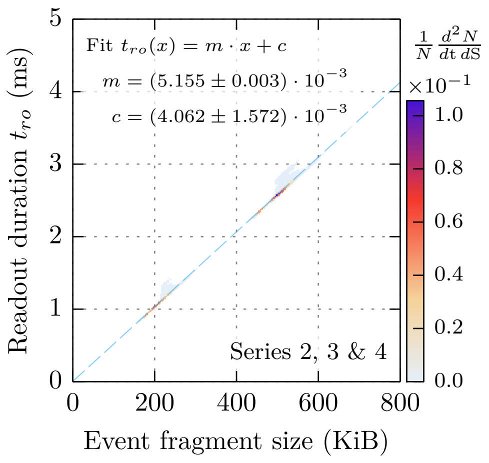


Figure 7.6: Readout duration as function of the event fragment size in Run1. See text for detailed description.

With the integration of the SIU functionality into the SMU FPGA for Run2, the frequency of the SIU core could be raised to 120 MHz. This results in an increase of the core input bandwidth to 457.76 MiB/s (orange, dashed). Again, the effect of the event transaction overhead (red, solid) can be ignored for event fragment sizes seen in the experiment. The transceiver integrated in the SMU operates at 4 Gbit/s, which limits the maximum DDL2 output rate to 369.91 MiB/s (red, dashed), again taking into account inefficiencies due to 8b10b encoding and SIU framing. The backpressure fraction of 19.19 %, which is

[6] $R_{\text{DDL,eff}} = 2.125\,\text{Gbit/s} \cdot \underbrace{8/10}_{\text{8b10b}} \cdot \underbrace{256/264}_{\text{SIU framing}} \approx 196.52\,\text{MiB/s}$

consequently seen by the SIU core in regular operation, is shadowed by the event buffering and contributes to dead time only indirectly by way of potentially filled event buffers.

DAQ Interface Performance Run1 For Run1, the readout performance is measured by analyzing time stamps that were inserted into the event data streams in the previously mentioned series of test runs, see Appendix C.1. Figure 7.6 shows the readout duration as a function of the event fragment size. Three supermodules, SMs 06, 09 and 10, participated in the test runs. The occasionally observed prolonged readout duration is not caused by the GTU, but has been verified to be caused by throttling of the readout by the LDC of SM 06. Data from SM 06 has therefore been excluded from the fit calculation. The slope m of 5.155 µs/KiB resembles, when ignoring DAQ backpressure, the processing time of the L2 stage.[7] It corresponds to an effective output data rate of 189.44 MiB/s per SM, which signifies a 99.3 % use of the available output bandwidth.

DAQ Interface Performance after Run2 Upgrade With the Run2 upgrade, DDL2 characteristics and performance data are measured by the PPCs in-system. The data rate into the SIU core (red) and the effective data rate over the DDL2 (blue) are shown in Figure 7.7. Additionally, the backpressure fraction is plotted (orange) on a detailed (upper plot) and full scale (lower plot). Each data point is obtained by averaging event-by-event measurements over a period of several seconds, see Appendix C.2. The rate into the SIU core is measured before the decoupling FIFO and the TX FIFO (see Section 4.7.3). At low primary event rates, it is higher than the output data rate over the DDL2. The average fill level of these FIFOs increases with the primary event rate, which explains the settling of the SIU input data rate to the level of the output data rate once the system's workload increases. The system is capable of supplying the maximum data rate to saturate the DDL2 irrespective of the primary event rate. The minor rate drops visible are a direct consequence of short periods of backpressure originating at the LDC.

Figure 7.8 shows the measured readout duration as function of the event fragment sizes. The L2 stage processing time of 2.641 µs/KiB translates to an effective output data rate of 369.79 MiB/s when shipping an event and corresponds to a DDL2 utilization of 99.97 %.

7.3 Event Fragment Sizes

The event fragments built by each GTU segment consist of tracklets and raw event data coming from the detector, and additional data, such as header structures, injected by the GTU. The total fragment size determines, together with the output bandwidth, the L2 stage processing time per event.

[7]The offset c of 4 µs can be ignored for a typical readout duration in the order of ms.

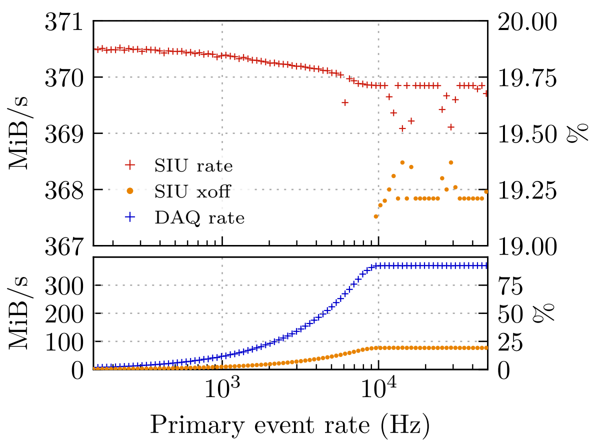


Figure 7.7: Input data rate into the SIU, effective data rate over the DDL2 and back-pressure measured at the interface between data path and SIU core with $r_{\mathrm{L1/L0}} = 0.1$, $r_{\mathrm{L2/L1}} = 0.75$ and approximately 500 KiB event size.

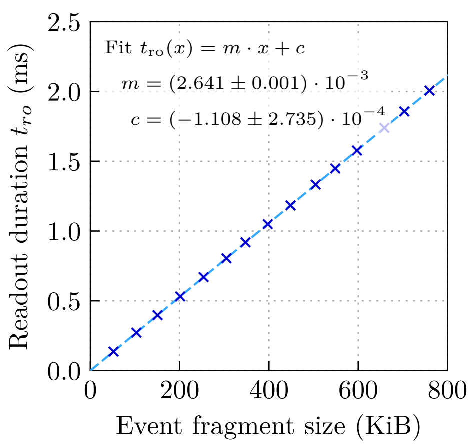


Figure 7.8: Measurement of the readout duration as function of the event fragment size in Run2.

Pb-Pb Figure 7.9a shows the size of event fragments transmitted to the DAQ as function of the event centrality in a Pb-Pb sample at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$.[8] Going from peripheral (100 % centrality) to central (0 %) collision events, the multiplicity, and thus channel occupancy, increases. This is reflected in an increase of event fragment sizes from few KiB to sizes of 700 KiB for collisions with low impact parameter. In extreme case fragments of up to 870 KiB are observed. Assuming sufficient buffers in the data path and no backpressure, the average L2 stage processing time is determined by the readout duration of an event fragment of average size, which in turn largely depends on the selected trigger mix. In the inspected run, a mix of triggers on central and semi-central events was active, see Figure 7.9b for the exact centrality distribution. This resulted in an average event fragment size of 400–425 KiB, which corresponds in case of no backpressure to a readout duration of 2.06–2.19 ms in Run1 and 1.06–1.12 ms in Run2.

Figure 7.10 shows, in absolute and relative numbers, the data volume added to a fragment by the SMU in the process of local event building. The offset includes static elements added to the fragment, such as the supermodule header, the stack headers, the event trailer with CRCs or the stack separation markers. The reconstructed tracks and information on the resulting triggers, which are also added to the fragment, contribute the bulk of generated data for central collisions. Fragments transmitted to the DAQ consist largely of raw event data sent by the TRD FEE, the percentage of generated data for Pb-Pb is in the order of 0.1–1.5 %.

[8]Corrections for inactive detector chambers were applied. The corresponding event fragment sizes were scaled up to resemble those of fully equipped, operational supermodules.

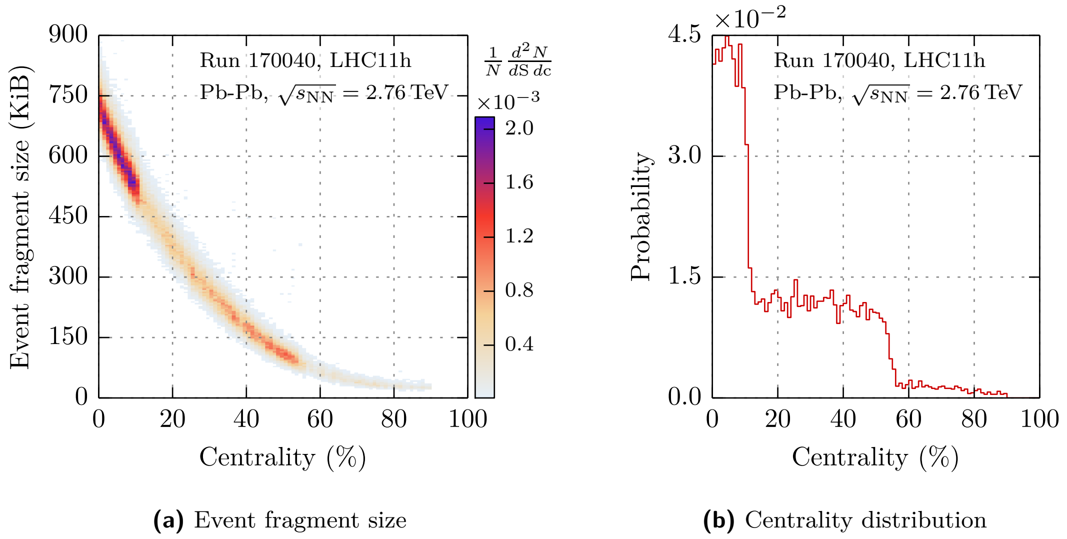


Figure 7.9: Event fragment size as function of the event centrality in Pb-Pb (Run1) and the centrality distribution of the selected run.

p-p, p-Pb Event fragment sizes in other running scenarios differ substantially from the ones observed in Pb-Pb. In p-p for example, average event fragment sizes are in the range of 7–12 KiB, while p-Pb on average exhibits fragments of 20–30 KiB. With the firmware designs for Run1, this corresponds to a readout duration of about 35–60 µs and 100–150 µs, respectively. With the GTU designed to support concurrent reception and readout of events, readout of such a low volume event typically finishes while other detectors in the partition still exhibit dead time. Consequently, the GTU event buffers do not run full and no readout-related contribution to dead time is present in these scenarios.

7.4 TRD Readout Performance Simulation

When deciding on the mix of active triggers for a physics run, it is indispensable to evaluate the dead time behavior of the detectors in a partition and the readout data rates that have to be handled. Knowing how the generated dead time varies as function of the primary event rate and accept ratios for the trigger stages allows to adjust the load on different processing stages such that the maximum amount of statistics for physics analyses can be acquired in a given time.

As pointed out in [62], an analytic treatment of the readout chain of a single detector subsystem with multiple trigger levels already is extremely difficult. Running multiple subdetectors together in a readout partition introduces correlations regarding the individual event arrival or processing time distributions, which further complicates an analytical approach. To determine the expected dead time behavior of the TRD, standalone as well as in common operation with ALICE's main tracking detector, the TPC, an extensible, event-based Monte Carlo simulation was developed.

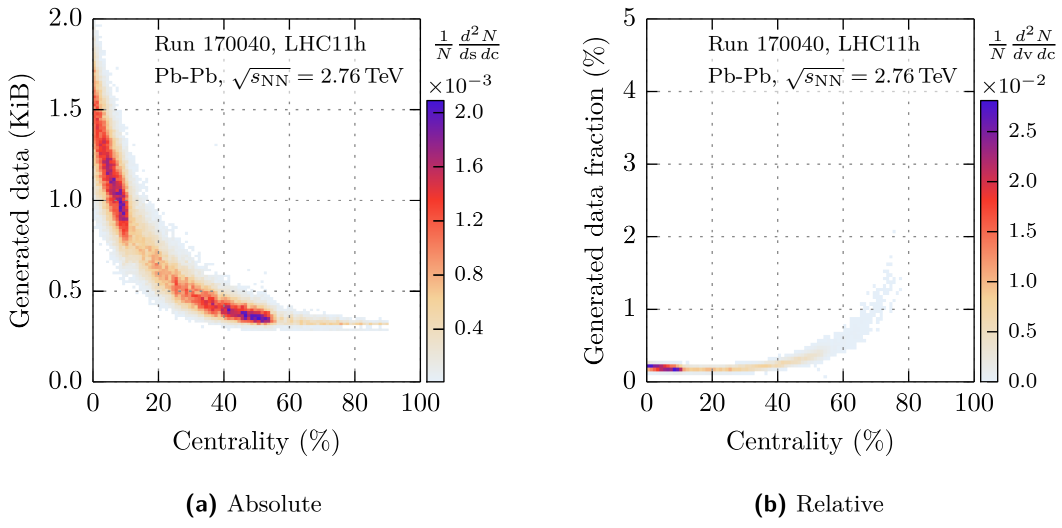


Figure 7.10: Data volume generated per event fragment by a GTU in Pb-Pb (Run1). This includes inserted header structures as well as calculated tracking and trigger information.

7.4.1 Simulation Overview

The simulation, depicted in Figure 7.11, is built in a modular fashion and comprises a trigger processor module as basis element, plus several detector subsystem modules. Participation of a detector module in the simulation is configurable to give the flexibility to study the readout behavior of a single detector, but also that of a readout partition comprised of several subdetectors. Complexities of a subdetector's readout process as well as parameters specific to the subdetector are part of the corresponding subsystem module to keep the simulation code structured and maintainable. A typical simulation run sweeps the primary event rate (collision rate) in the range from 0.2–200 kHz to calculate dead time and expected readout rates for a given set of simulation parameters.

The simulation utilizes the SystemC [65], a C++ library targeted at the modeling of complex hardware/software systems, to implement functional models for the CTP and the detector subsystem modules. Calculation of the individual simulation steps is parallelized using Python's multiprocessing package [63].

7.4.2 Trigger Mix and Processing Times

Correlations between the processing times of individual subdetectors' trigger stages exist in real operation. Looking at TRD and TPC, an increased detector occupancy, e. g., due to a central Pb-Pb event, causes higher readout times in both detectors than low-occupancy events. To enable the simulation of a variety of trigger mixes and to model the correlation of processing times between detectors, a `cent` distribution is provided as input to the simulation. By forwarding the `cent` information of the current event to the modules, the subsystems can select appropriate processing times, which often correspond to event sizes,

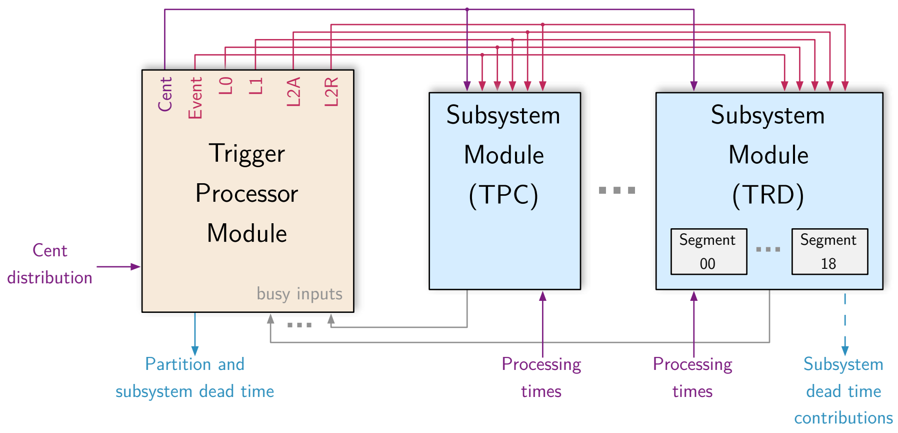


Figure 7.11: Modular event-based Monte Carlo simulation of the partition readout behavior implemented using SystemC. Using centrality and processing time distributions for the different trigger stages as input, the simulation allows to calculate the dead time behavior of the partition and of individual subdetectors.

for injection into their processing stages. In case of Pb-Pb simulations, cent may be interpreted as event centrality.

The simulation offers a number of running scenarios, whose detector-specific details are implemented by the corresponding submodules. They differ with respect to the amount and variation of processing time of the various stages:

- const is used to simulate constant processing times. The processing time is user-defined for each processing stage of a subdetector, the cent parameter has no effect.
- normal simulates processing times following a normal distribution $f(\mu, \sigma)$ with user-defined mean μ and $\sigma = 0.1\,\mu$. The cent parameter again has no effect.
- pbpb is targeted at reproducing the detector operation in heavy ion runs. It allows to inject processing times selected on the basis of the cent parameter. The underlying processing time distributions are those observed in Pb-Pb runs at $\sqrt{s_{NN}} = 2.76\,\mathrm{TeV}$.

7.4.3 Component Modules

Central Trigger Processor Module The CTP module implements the event loop and steers the simulation by generating triggers and auxiliary signals. The event port notifies the connected subsystems of a collision event, while the L0, L1 and L2 output ports represent the ALICE hardware trigger stages. The cent port is updated with a value in the range of 0–100 simultaneously with the event port and remains valid until the L1 decision interval.

Figure 7.12 shows three exemplary centrality distributions used in the simulation and the resulting average centralities c_{avg}. The distribution central-very simulates a worst case scenario with strong focus on triggers the most central collisions, i. e., high multiplicity, while central-semi resembles an average trigger mix encountered in Pb-Pb, see Figure 7.9b.

Three parameters provide control over the accept ratios for the different trigger stages:

- a_{L0}: the fraction of primary events passing the L0 stage,
- a_{L1}: the fraction of L0 events passing the L1 stage and
- a_{L2}: the fraction of L1 events passing the L2 stage.

Using the busy information reported back from the subsystems, the CTP module handles accounting of dead time contributions from individual subsystems and of the overall dead time of the readout partition.

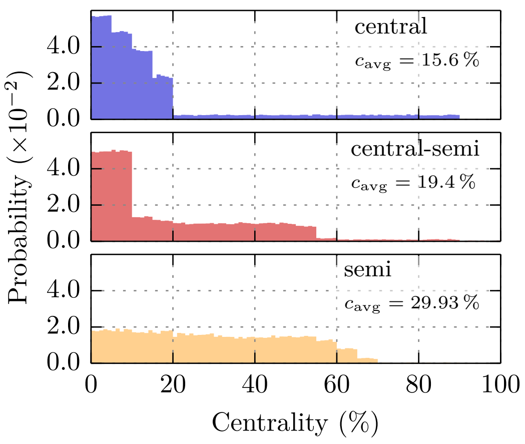


Figure 7.12: Exemplary centrality distributions and resulting centrality averages used as input to the performance simulation.

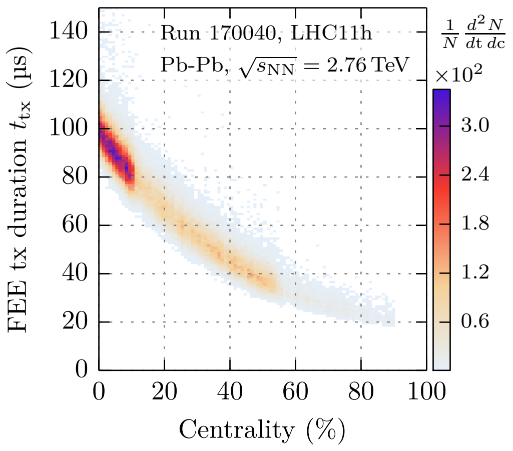


Figure 7.13: Duration of the data transmission from FEE to GTU as function of the event centrality in Pb-Pb (2011).

TPC Subsystem Module The module implements an abstract readout model for the TPC following the implementation of the TPC readout system for Run1 presented, for example, in [91, 97]. The TPC possesses 216 independently operating sectors with one DDL per sector and four event buffers on its FEE. It cannot accept a new event if one or more of the sectors has all four event buffer slots occupied. The acquisition of a new event is typically started with the arrival of a L1 trigger at the TPC FEE and ends after approximately 100 µs. At this time, a new event is available for readout in the TPC event buffers. The acquisition phase is followed by an ion feedback protection interval, during which the gating grid is closed for about 180 µs. Only afterwards, i. e., approximately 280 µs after the L1, the TPC lowers its `busy` and can record a new event, given that the event buffers are not full. Processing of a queued event starts immediately once the corresponding L2 trigger was received and leads either to a drop in case of a L2R or readout to the DAQ in case of a L2A. The interruption of a potentially ongoing readout for the duration of the acquisition phase, which is done to minimize the detector noise, is also reflected in the model.

To delimit the simulation effort, the abstract model does not implement 216 independent sectors with individual readout duration. Instead, a single queue is used in combination with the distribution of the readout duration measured for the complete TPC in Run1. The readout duration is modeled as function of the event centrality for Pb-Pb at $\sqrt{s_{NN}} = 2.76\,\text{TeV}$ based on the data provided for sparse readout in Figure 4.23 of [97]. The resulting readout time averages with the aforementioned centrality distributions are listed in Table 7.1.

At the time of writing, final numbers for the Run2 readout performance with the upgraded TPC readout electronics [120] are not yet available. However, the average readout duration is assumed to decrease by a factor of 0.7 down to a minimum time of 100 µs that is required to scan channels for hits in sparse readout mode. This decrease in readout duration is motivated by an expected increase of event sizes by 25 % and an increase of the TPC DDL bandwidth by approximately 80 % compared to Run1 [17].

TRD Subsystem Module The TRD module is a wrapper module for up to 18 segment submodules, which are modeled closely after the actual hardware. Each submodule essentially implements the equivalents of event handler and readout initiator outlined in Section 3.5. The event handler generates the L0/L1-related dead time contributions and pushes, in case of a L1A, an entry for the event into the submodule event queue once the FEE transfer is completed at t_{tx}. Depending on the fill state of the event queue, readout dead time is generated. Once an entry present in the L2 trigger queue and in the event queue, both entries are consumed by the readout initiator in t_{ro} for a L2A or a fixed time of 100 ns for a L2R. With independent queues, all segment submodules operate independently in terms of readout and are only synchronized by the received trigger sequences. The number of segments participating in the simulation and the depth of their event buffer queues is configurable. The TRD-wide busy and dead time contributions τ_{L0}, τ_{tx} and τ_{ro} are calculated based on the contributions from the participating segments and communicated to the trigger processor module.

In the pbpb simulation scenario, t_{tx} and t_{ro} distributions provide the TRD L1 and L2 stage processing times, respectively. They are extracted as function of the event centrality from heavy ion run data at $\sqrt{s_{NN}} = 2.76\,\text{TeV}$, see Figures 7.9a and 7.13.[9] For different input centrality distributions in the pbpb scenario, Table 7.1 summarizes the average values for t_{tx}, t_{ro} and the corresponding event fragment size. Simulations for Run2 take into account the increase in readout bandwidth and assume increases for the FEE transfer duration and event fragment size of 5 % and 25 %, respectively.

Figure 7.14 shows the correlation of the FEE transfer duration t_{tx} and the readout duration t_{ro}. The strong correlation visible for Pb-Pb is modeled in the submodule. The cent parameter of an event propagates through the L1 stage (event handler) to the L2 stage (readout initiator) and each stage selects an appropriate processing time. Albeit optimized for Pb-Pb, the current dead time simulation can also be used to assess the behavior in p-p runs. As the total variation of both transfer and readout duration is small, simulations

[9]Centrality bins of 5 % width are projected onto the Y-axis to provide the input to the simulation. Results from Sections 7.1.2 and 7.2 are used to convert link data volumes and event fragment sizes to transfer/readout times.

	Centrality	TRD			TPC
	c_{avg}	$t_{\text{tx,avg}}$	$t_{\text{ro,avg}}$	$s_{\text{frag,avg}}$	$t_{\text{ro,avg}}$
Run1, $\sqrt{s_{NN}} = 2.76\,\text{TeV}$					
central	15.6 %	77 µs	2.54 ms	493 KiB	2.66 ms
central-semi	19.4 %	72 µs	2.26 ms	439 KiB	2.50 ms
semi	29.9 %	59 µs	1.60 ms	310 KiB	2.02 ms
uniform	49.5 %	45 µs	1.01 ms	196 KiB	1.50 ms
Run2, $\sqrt{s_{NN}} = 5.5\,\text{TeV}$					
central	15.6 %	81 µs	1.62 ms	615 KiB	1.85 ms
central-semi	19.4 %	75 µs	1.44 ms	546 KiB	1.73 ms
semi	29.9 %	62 µs	1.02 ms	388 KiB	1.41 ms
uniform	49.5 %	47 µs	0.65 ms	245 KiB	1.03 ms

Table 7.1: Resulting averages for the TRD FEE transfer time t_{tx}, the TRD readout time t_{ro}, its corresponding event fragment size $s_{\text{frag,avg}}$ and the TPC readout time in the pbpb simulation scenario for different input centrality distributions. With respect to Run1, an increase of 25 % in TRD event fragment sizes and 5 % for the FEE transfer duration is assumed for Run2.

of the const or normal scenario with mean values for $t_{\text{tx}}/t_{\text{ro}}$ selected to reflect the desired trigger mix will yield accurate results.

7.5 Readout Performance and Dead Time

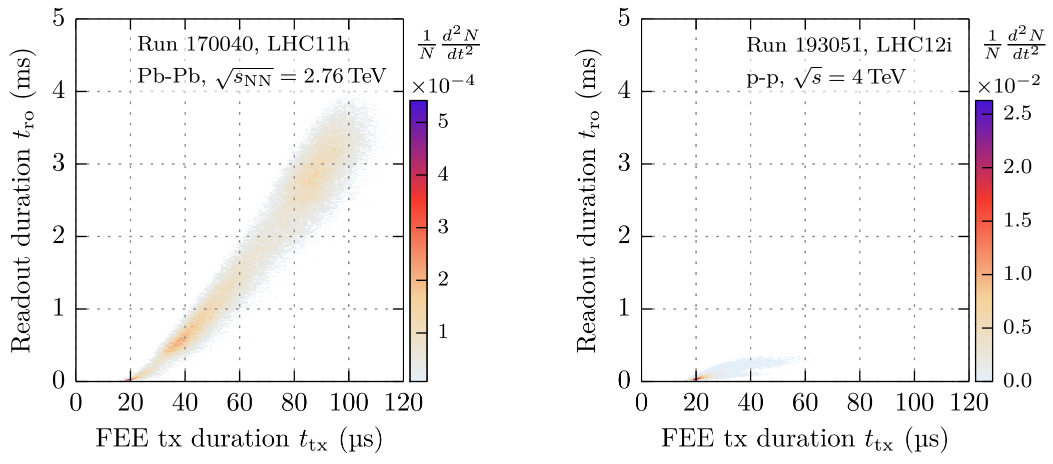
Using the measurements from test setup and production system, the simulation for the readout performance presented in Section 7.4 is verified. It is then used in Section 7.6 to determine the readout performance and dead time for common running scenarios.

7.5.1 Simulation Compared to Theory of Two Independent Stages

Although interconnected and therefore not independent, the simulated behavior of the two successive processing stages of a TRD supermodule is compared to theory-predicted results of two independent stages with given characteristics.

This comparison is shown in Figure 7.15 for different numbers of event buffers N.[10] The simulation of a single TRD supermodule plus GTU segment is run with the trigger accept ratios $a_{\text{L0}} = 1.0$, $a_{\text{L1}} = 0.075$ and $a_{\text{L2}} = 0.9$. A constant processing time model is used and the FEE transfer time is set to constant $t_{\text{tx}} = 72\,\mu\text{s}$. Because of the potential abort of the trigger sequence at L1, the average processing time of the first stage then

[10]Note the different definitions for the number of event buffers N: The author of [62] defines N_{Heath} as "buffer space for N events, in addition to the event currently being processed." Here, N_{GTU} is defined as the total number of available buffers slots, i. e., $N_{\text{Heath}} = N_{\text{GTU}} - 1$.



(a) Pb-Pb, LHC11h **(b)** p-p, LHC12i

Figure 7.14: Duration of the data transmission t_{tx} from FEE to GTU and the time required to read out the event fragment t_{ro} to the DAQ in Run1.

is $T_{p,\mathrm{1st}} = t_{\mathrm{L0}} + a_{\mathrm{L1}} \cdot t_{\mathrm{tx}} = 11.90\,\mathrm{µs}$ with $t_{\mathrm{L0}} = 6.5\,\mathrm{µs}$. For the readout stage a constant processing time of $T_{p,\mathrm{2nd}} = t_{\mathrm{ro}} = 2.27\,\mathrm{ms}$ for a L2A is simulated, which corresponds to an event size of 440 KiB in Run1. The simulated total supermodule dead time is plotted in solid red in Figure 7.15. Individual contributions to it from the first and second stage are shown in orange (dashed and solid, respectively).

The first stage is compared to the calculated dead time of a system with random arrival/constant (rnd-const) processing time, see Equation 3.10, which can handle one event at a time in $T_{p,\mathrm{1st}} = 11.90\,\mathrm{µs}$ (light blue, dashed). In terms of primary event rate this corresponds to $R_p = 84.0\,\mathrm{kHz}$ for a workload of $x_{\mathrm{1st}} = RT = 1$ and $a_{\mathrm{L0}} = 1$. The second stage is compared a rnd-const system with $R_p = 6.5\,\mathrm{kHz}$ at $x_{\mathrm{2nd}} = 1$ in terms of primary event rate (dark blue, dashed). R follows from the simulated processing time $T_{p,\mathrm{2nd}}$ plus corrections for the accept ratios of the trigger stages. The four plots show the corresponding curves for $N = 1, 2, 4$ and 8 event buffers. Shown additionally for comparison in all plots is a second stage with $N = 128$ as dotted green line. It practically represents the best-case second-stage behavior, as dead time only starts to increase at the primary event rate R_p where the workload is $x_{\mathrm{2nd}} = 1$.

Considering the discrepancies in the processing time distributions of simulated and calculated case, the event-based simulation and the calculation show very good agreement. For $N = 1$, the readout is the major contributor to the system dead time and the simulated TRD dead time (red) essentially matches the calculated second stage curve (dark blue). With increasing N, dead time contributions from the readout are diminished and two domains are recognizable: For low rates, at minimum up to the point where $x_{\mathrm{2nd}} = 1$, the total dead time is dominated by the first stage contribution. For higher rates, starting around $x_{\mathrm{2nd}} = 1$ at the latest, the second stage becomes the dominating factor.

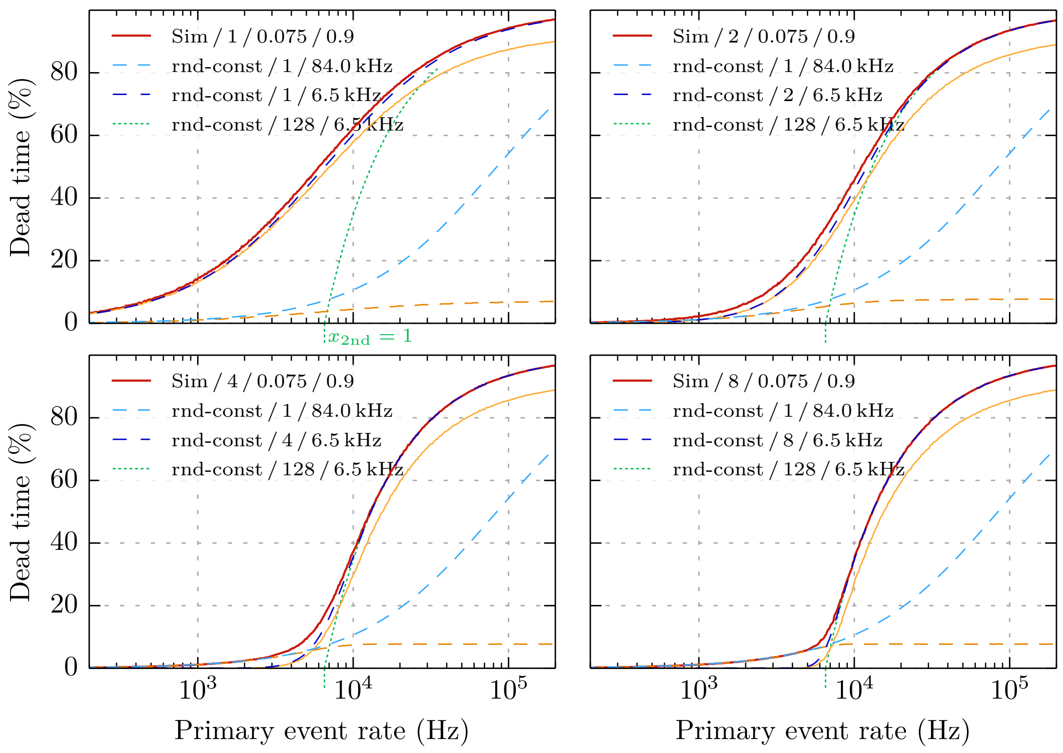


Figure 7.15: Comparison of the simulated TRD dead time (red, solid) and the calculated dead time of two independent processing stages (blue and light blue, dashed) with random arrival/constant processing time for different number of event buffers $N = 1, 2, 4, 8$. Shown (in orange) are the contribution to dead time of the TRD L0/L1 (dashed) and L2 (solid) processing stages. Legend parameters denote N / a_{L1} / a_{L2} for the simulation, N / R_p for the calculations.

While the reduction of dead time that can be achieved with buffering of several events is comparably large up to $N = 4$, a further increase of N yields only minor improvements for events of the investigated size.

7.5.2 Measured Performance Compared to Simulation

Comparison for Run1 Designs From time stamps written to the event fragments in test runs with dedicated firmware designs, the dead time generated by the GTU for the TRD and the achieved readout rates can be extracted. These test runs, see Appendix C.1, are carried out with the production setup. For a realistic test, the participating TRD supermodules run in a global partition with full CTP trigger logic and are configured to produce noise data events of roughly a given size.

Each run within a series corresponds to a selected primary event rate. The four recorded series of runs differ with respect to their parameters:

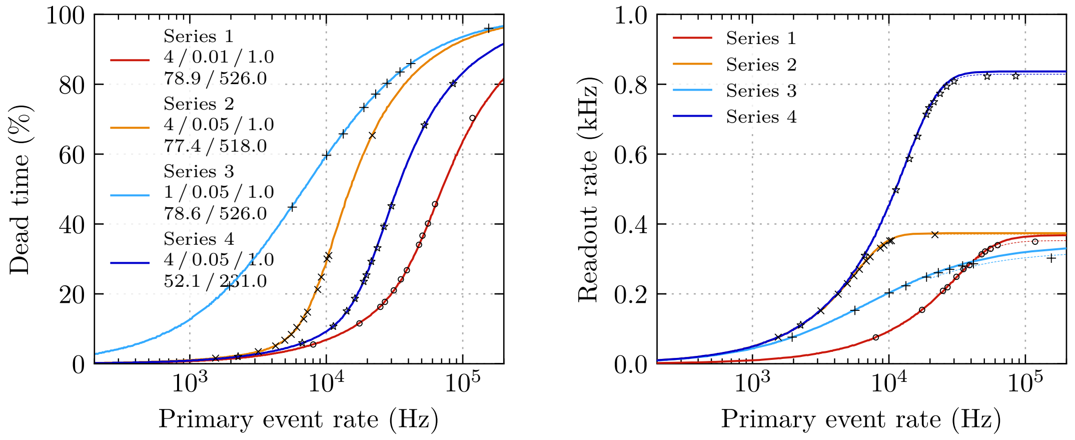


Figure 7.16: TRD dead time (left) and corresponding readout rates (right) measured (points) in Run1 compared to the performance simulation (colored lines). Simulation parameters are N / a_{L1} / a_{L2} (1st row) and t_{tx} [µs] / s_{frag} [KiB] (2nd row). FEE transfer duration and event fragment sizes used in simulation are mean values of a series.

- The L1/L0 trigger ratio a_{L1} (Series 1: $a_{\mathrm{L1}} = 0.01$, others $a_{\mathrm{L1}} = 0.05$),
- the number of event buffers N (Series 3: $N = 1$, others $N = 4$), and
- the event fragment size s_{frag} (Series 4: about 230 KiB, others about 525 KiB).

Figure 7.16 compares the Run1 measurement results to simulation. The simulation is run with one active segment only, although three supermodules and their GTU segments participate in the measurements. This is justified by the observation that, although configured identically, one supermodule produces on average larger events due to its specific noise behavior. Its increased FEE transfer duration t_{tx} and the increased readout duration t_{ro} for that segment consequently dominate the global TRD dead time.

The simulation parameters t_{tx} and t_{ro} are the average values extracted in an analysis of all runs of a given series. The simulation reliably reproduces the measurements for all shown parameters sets, which indicates its correctness. Minor deviations between simulation and measurement, mostly observable for data points at large primary event rates, can be attributed to two effects both related to limitations of the measurement setup. The first effect is light backpressure originating at one of the involved LDCs. This external throttling of the readout, e. g., seen for the highest rate data points of Series 3 and 4, increases the effective readout time. The second is an increase in the event fragment size for larger primary event rates due to increased detector noise. This unintended behavior spawns from limited possibilities to produce event sizes fluctuating around a desired mean size with the FEE. The run corresponding to the data point at 118 kHz of Series 1 (red) exhibits an average event size of 559 KiB, while the series average size used to simulate is 526.8 KiB. In the readout rate plot, dashed lines show the simulation results when using the largest average event size found in the series as simulation parameter. This usually corresponds the most-right data point of a series, and again illustrates good agreement of simulation (with correct input parameters) and measurement.

Comparison for Run2 Designs The revised internal dead time accounting implemented alongside the readout upgrade for Run2 facilitates a direct measurement of the segment's individual dead time contributions and readout bandwidths. Due to unavailability of the production setup, the measurements are carried out with the test setup. The accept ratios of all trigger stages are fixed to 1.0.[11] The test again sweeps over the primary event rate and the relevant PPC-measured quantities are recorded for each data point, see Appendix C.2 for further details. The GTU link-level pattern generators are set to produce events of a constant size and the number of event buffers is fixed to $N = 4$. The measurements (data points), shown in Figure 7.17, are carried out for different event sizes and compared to a simulation (colored lines) with one participating segment using the same parameters.

The TRD segment simulation model has been adapted to reflect the reduced processing time of the L2 stage due to the increased DDL bandwidth in Run2. Event sizes are converted to processing times in accordance to the measurements shown in Figure 7.8. The difference of SIU decoupling FIFO input bandwidth and effective output bandwidth of the SIU TX FIFO, see Section 4.7.3, can cause the event fragment dispatcher to see an increased readout bandwidth. Assuming no DDL backpressure and empty FIFOs at the start of an event transmission, the combined FIFO size of 34 KiB allows the first 377 µs of each transmission to occur at 3.84 Gbit/s before throttling to the effective DDL output rate of 3.1 Gbit/s. The corresponding decrease of the L2 stage processing time around $x_{\text{2nd}} = 1$, where the amount of reduction depends on the current fill state of the FIFOs, has been modeled in the simulation. However, at the given rate difference and FIFO sizes, it is comparably small and only slightly affects the shape of the dead time curve.

The slight deviation of measurement and simulation observable for the lowest event size (green) at large primary event rates has been traced to be a limitation of the measurement setup. It originates at the LDC, which produces increased backpressure when handling events at readout rates above 7.5 kHz and therefore limits the available average output bandwidth to a value lower than presumed in the simulation. Under the reasonable condition that the readout duration used to simulate corresponds to actual output bandwidth offered by the LDC, the simulation shows perfect agreement with the measurement. It can reliably reproduce and predict both dead time exhibited by the system and achievable readout rates with the Run2 hardware.

7.6 Performance in Physics Runs

With help of the verified TRD simulation, the performance of the GTU with respect to dead time and readout rates can be evaluated for different running scenarios. The following simulations show the behavior of 18 TRD supermodules in standalone operation.

[11] The test setup does not feature the full CTP with the option of conditional L1 or L2 accepts. Instead, the LTU's internal emulator supplies a defined, limited set of trigger sequences replayed with a random signal acting as start signal for a sequence. The repetitive replay of a set with limited number of sequences does not allow to imitate sufficient randomness regarding the L1 and L2 decisions. This leads to certain event buffer fill/drop patterns and thus prevents a clean comparison of measurement and simulation with trigger ratios other than 1.0 with this setup.

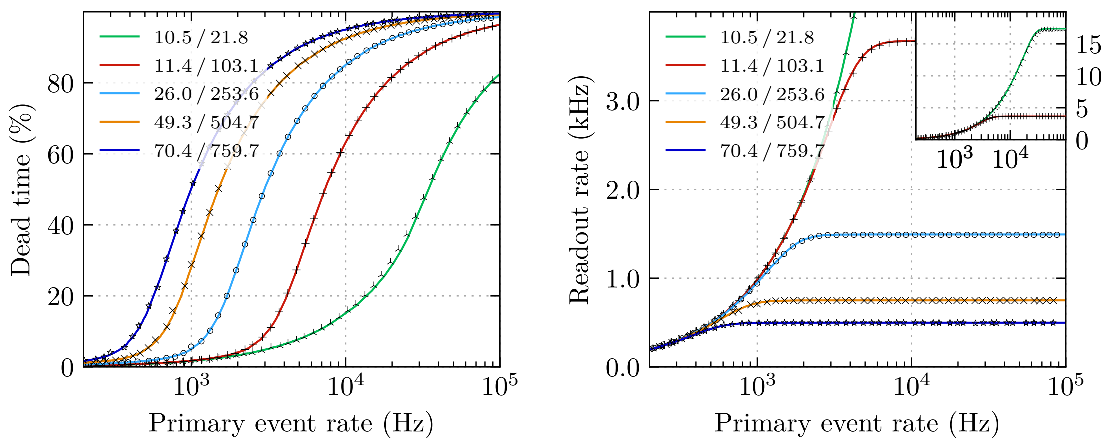


Figure 7.17: Measurement of dead time (left) and corresponding readout rates (right) with the upgraded GTU designs in Run2. Data from the test setup (data points) is compared to the performance simulation (colored lines). Simulation parameters are t_{tx} [µs] / s_{frag} [KiB]. The number of buffers N and all accept ratios are fixed to 4 and 1.0, respectively. The inset shows the achievable readout rates for the two lowest event sizes at full scale.

Additionally, the behavior of a common readout partition with the TRD and TPC, which is a typical use case in physics runs, is shown for comparison.

Processing times in the simulation are modeled after those found in Pb-Pb runs (pbpb scenario, see Table 7.1). The input centrality distribution, central-semi,[12] comprises triggers on central and semi-central collisions, resulting in an average centrality of about 20 %. For the TRD, all 18 supermodules with individual event queues are simulated, either in standalone operation (TRD) or in common readout with the TPC (TRD+TPC). The accept ratios for the L1 and L2 stages are varied, simulated are $a_{\mathrm{L1}} = [0.01, 0.05, 0.1]$ and $a_{\mathrm{L2}} = [0.5, 1.0]$.

7.6.1 Performance in Pb-Pb, Run1

The simulation results with event sizes and readout interface as in Run1 are shown Figure 7.18. The two top graphs show the dead time as function of the primary event rate, i. e., interaction rate. Solid lines mark the dead time of the complete TRD when running alone, while the dashed lines show the system dead time of a common TRD+TPC partition. Additionally shown as dotted lines is the dead time $\tau_{\text{ro-trd}}$, which is the readout-related TRD dead time when running in the common partition. Colors denote the different L1 accept ratios. The discrepancy in dead time between TRD and TRD+TPC for low event rates is due to the 280 µs dead time of the TPC upon each opening of its gating grid [91]. This compares, at this average centrality, to 72 µs for the TRD FEE transfer upon a L1.

[12]See Appendix C.3 for simulations with other input centrality distributions.

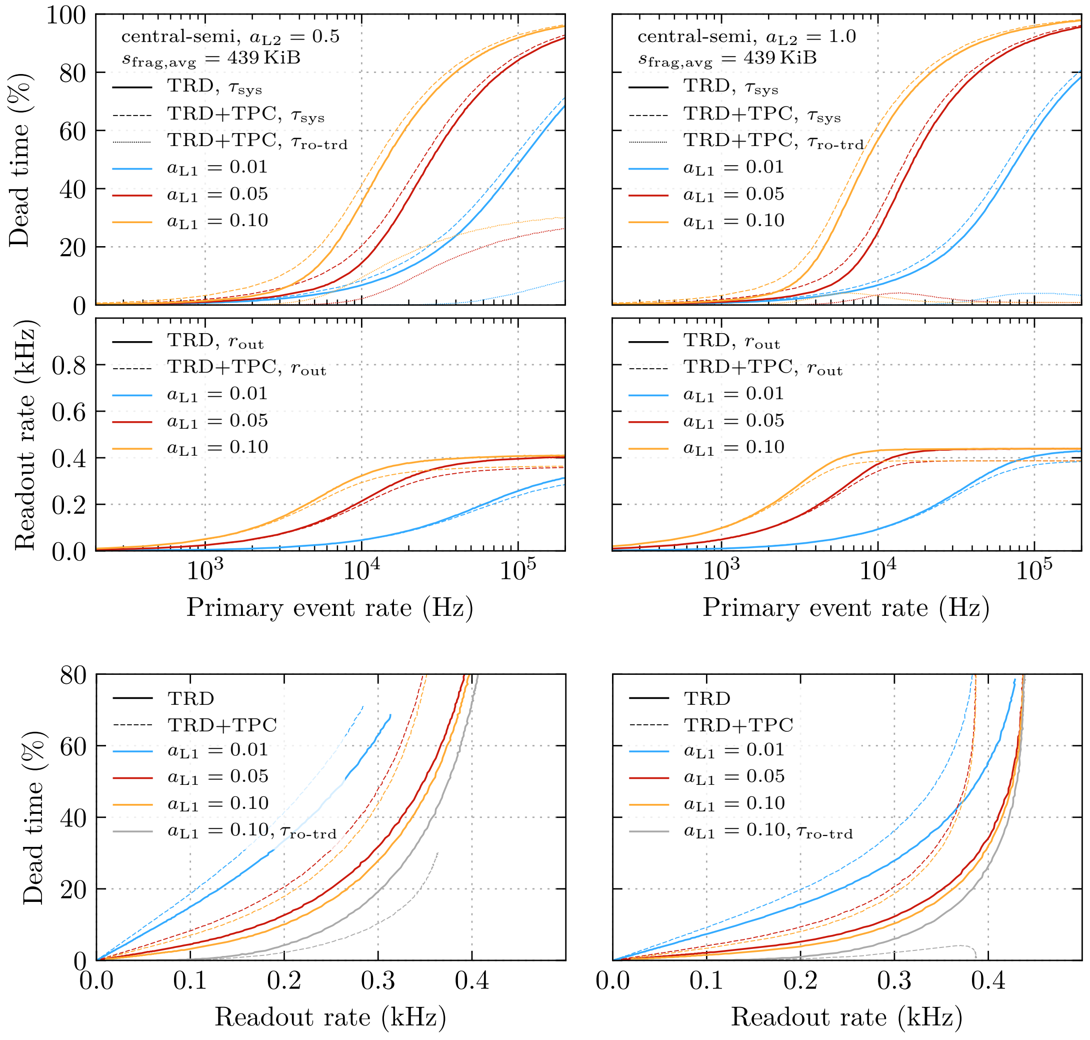


Figure 7.18: Run1 results from the system simulation for two L2 accept ratios, $a_{L2} = 0.5$ (left column) and $a_{L2} = 1.0$ (right column). The pbpb scenario with input centrality distribution central-semi results in average TRD event fragment sizes of $s_{frag,avg} = 439$ KiB. Colors denote different L1 accept ratios, event buffers are fixed to $N = 4$. Simulated are 18 TRD supermodules, both in standalone operation (solid) as well as in a common readout partition with the TPC (dashed/dotted).

At higher event rates, the dead time is dominated by the readout capabilities. Table 7.1 shows that the processing time of the TRD readout stage does not exceed that of the TPC readout, which explains the comparably low TRD readout dead time $\tau_{\text{ro-trd}}$ for large event rates in the common partition. At the given accept ratios, the TPC dominates the partition dead time also at high rates.

The two center graphs shows the readout rate, i. e., L2A rate, as function of the primary event rate. If dead time is not an issue, e. g., in minimum bias data taking, the TRD achieves readout rates of 400–425 Hz, depending on the L2 accept ratio, at the given event fragment size of 439 KiB. The combined partition allows for maximum readout rates of 360–390 Hz.

The two bottom plots, which show the system dead time as function of the readout rate, enable a performance comparison to the target for Run1, which is recording central events at 200 Hz at reasonably low dead time. With the TRD in standalone operation, readout rates up to 250 Hz with less than 20 % dead time are feasible for $a_{\text{L1}} > 0.05$ and $a_{\text{L2}} = 0.5$. For $a_{\text{L2}} = 1$ the dead time is less than 10 %, as a larger L2A ratio corresponds to less stress on the event buffers per event read out. For comparison, the dead time contribution of the TRD readout $\tau_{\text{ro-trd}}$ is shown in gray for $a_{\text{L1}} = 0.1$.[13] This contribution is shadowed most often by other dead time contributions of the partition.[14] Up to the target readout rates, τ_{L0} and τ_{tx} constitute the majority of the TRD dead time. The contribution $\tau_{\text{ro-trd}}$ is small and almost negligible in case of the common partition and target readout rates. For readout rates very close to maximum partition workload, the `busy` of the partition gradually starves the TRD, so that it manages to slowly empty its buffers, as indicated by a decreasing $\tau_{\text{ro-trd}}$.

7.6.2 Performance in Pb-Pb, Run2

For Run2, the simulations assume an increase in event size in Pb-Pb runs at $\sqrt{s_{NN}} = 5.5\,\text{TeV}$ of 25 % and upgraded DAQ interfaces for the TRD and TPC, see Table 7.1. Again, 18 TRD supermodules are simulated standalone and in common operation with the TPC. Figure 7.19 summarizes the results.

At the same average centrality, but increased event fragment size, the upgrade facilitates a maximum readout rate of the TRD in standalone operation of 620–680 Hz, depending on a_{L2}. Target readout rates for Run2 are 450–500 Hz for central events. Again, the overall dead time is dominated by the TPC for both low and high primary event rates. Although the final performance numbers of the TPC are not yet known, the simulation does not hint at a signification dead time contribution originating at the TRD. The readout capabilities of the TRD are expected to at least match those of the TPC.

[13]The simulation was performed for primary event rates in the range 0.2–200 kHz. Values for higher primary event rates are not plotted. Curves for $a_{\text{L1}} = [0.01, 0.05]$ are almost identical.

[14]The simulation also counts partition dead time caused exclusively by the TRD readout stage. These values, however, are very close to zero for the presented simulations and are therefore not plotted.

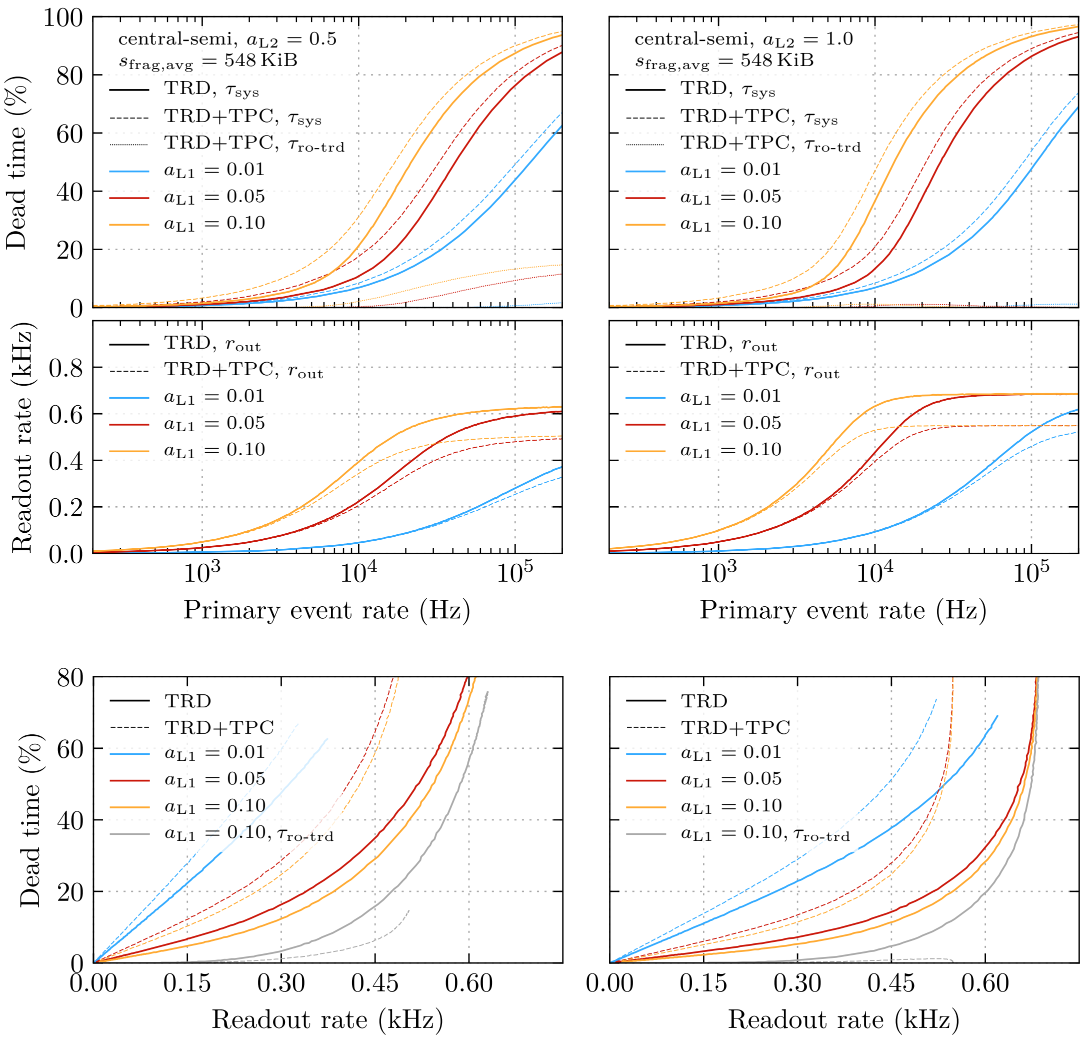


Figure 7.19: Run2 results from the system simulation for two L2 accept ratios, $a_{L2} = 0.5$ (left column) and $a_{L2} = 1.0$ (right column). Assuming an increase in event size of 25 % compared to Run1, the pbpb scenario with input centrality distribution central-semi results in average TRD event fragment sizes of $s_{frag,avg} = 548$ KiB. Colors denote different L1 accept ratios, event buffers are fixed to $N = 4$. Simulated are 18 TRD supermodules, both in standalone operation (solid) as well as in a common readout partition with the TPC (dashed/dotted).

7.7 Potential for Further Performance Improvements

In case the TPC readout exceeds the expected performance and readout-related TRD dead time presents a limit to the partition readout, the GTU offers the possibility to implement another stage of event buffering. The current width of the valid IDELAY tap windows in line-wise clock-to-data delay calibration of more than 1.5 ns at 240 MHz DDR, see Figure 4.8, furthermore suggests that an increase of frequency for the backplane transmission from 240 MHz to 320 MHz is feasible. Sufficient FPGA fabric resources are available on the SMU, see Appendix A, to implement potentially required pipeline stages and increase the data path frequency accordingly from 120 MHz to 160 MHz. The 4 MiB SRAM, used for event buffering on the TMU nodes, is currently unused on the SMUs. It can be used to implement a large FIFO that replaces the BRAM-based, limited-size SIU decoupling FIFO. At typical event sizes of less than 750 KiB and an input data rate of 5.12 Gbit/s, the FIFO could quickly sink several events and significantly reduce the L2 stage processing time.[15]

The effect of a 4 MiB output FIFO, an increased frequency of the data path of 160 MHz and the resulting L2 processing time reduction is simulated and compared to the current implementation in Figure 7.20. As before, the TRD is simulated in standalone operation in Pb-Pb runs (pbpb, central-semi). The enhanced system with the large buffer (solid lines) shows a significant improvement under increasing load on the readout when approaching $x_{\mathrm{2nd}} = 1$. The de-randomizing effect of the large buffer is clearly visible. With increasing readout rate the dead time shows only the inevitable linear increase due to the intrinsic τ_{L0} and τ_{tx} contributions. A significant contribution from the readout τ_{ro} is only observed just before approaching the maximum readout rate that can be handled by the DDL2.

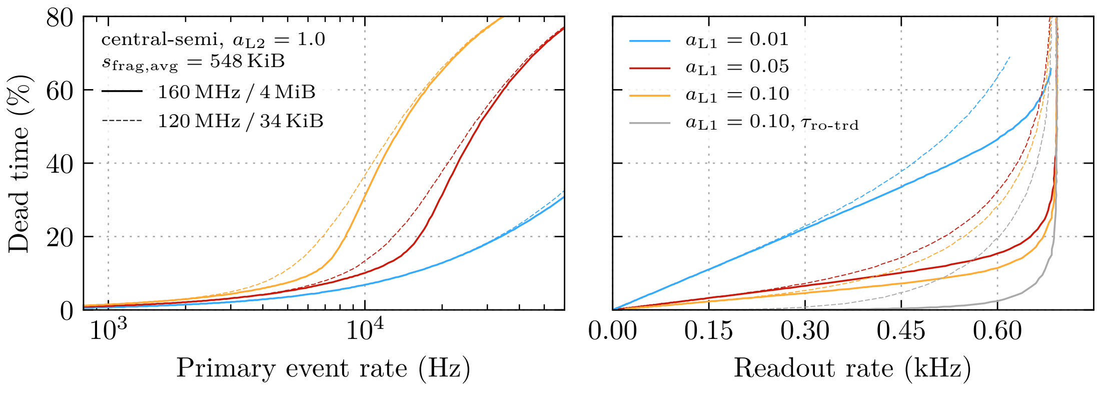


Figure 7.20: Comparison of dead time and readout rate of the current implementation with 120 MHz data path and small SIU output FIFO (dashed) to an enhanced data path at 160 MHz and large, SRAM-based output FIFO (solid lines). Simulated is the TRD in standalone operation with $a_{\mathrm{L2}} = 1.0$. Different colors denote different L1 accept ratios.

[15] A buffer slot can be freed on the TMUs as soon as the last word of the event has entered the FIFO after the event dispatcher.

Summary

The transfer of tracklets and event raw data from detector half-chambers and the associated processing times of the GTU's first stage provide valuable input to the dead time simulation performed in this chapter and are therefore characterized. An analysis of the internal buffering and data handling furthermore shows that the processing times for the second/readout stage in practice only depend on the performance of the interface. The sophisticated design and the data path optimizations presented in the previous chapters allow to achieve corresponding DDL output bandwidths for Run1 and Run2 of 190.7 MiB/s and 369.8 MiB/s, respectively, which correspond to a DDL utilization of 99.30 % and 99.97 %.

The stage processing times provide, together with the presented analysis of event fragment sizes distributions in the common run scenarios, the input to a custom-developed Monte Carlo simulation of the readout. The simulation allows to study the effects of different input parameter sets, such as event size distributions or trigger stage reject ratios, on readout performance and dead time generation for individual subdetectors, but also for jointly operating readout partitions. The simulation model for the TRD readout chain has been verified by hardware measurements to show perfect agreement, and clearly illustrates the beneficial effect of buffering multiple events on the GTU.

The verified simulation allows to study the performance of the TRD readout chain for various running scenarios, and in particular to compare standalone operation with conjunct operation in a readout partition with the TPC. The simulation results illustrate that both the designs for Run1 and the upgraded designs for Run2 are capable of handling the target readout rates of 200 Hz and 450–500 Hz, respectively. Dead time originating at the GTU readout stage itself is less than 10 % in these cases. This reasonably low dead time proves that the TRD readout stage is not a limiting factor in a common readout partition, and indicates ample data handling and readout performance for successful participation in efficient physics data taking.

8 Summary

The Transition Radiation Detector (TRD) of the ALICE experiment at the Large Hadron Collider provides fast, track-based triggers for the high-resolution main tracking detector of the experiment and raw event data that increases the tracking resolution in the offline analysis by extending the experiment's tracking arm. The Global Tracking Unit (GTU) is the central element of its processing chain and performs the online matching of track segments to global tracks, the subsequent trigger calculation and the high-speed local event building for the TRD. The requirements to supply track-based triggers within 7 µs after the interaction and to realize a fast, high-bandwidth readout of the TRD chambers lead to a GTU topology that organizes 109 FPGA-based processing nodes in a three layer hierarchy [41].

The high data rates the system has to handle and its dual use as trigger and readout processor with shared resources and interwoven processing paths require the GTU to be a unique, high-performance parallel processing system. Taking furthermore into account the complexity of a detector assembly like ALICE and the significant cost of operating both experiment and accelerator, it is essential to design all individual entities of such a system for optimum data taking efficiency, i. e. high running stability and low dead time.

The solutions presented in this thesis for the handling of readout data in the GTU, from the initial reception to the final assembly and transmission to the HLT computer farm, address all these aspects. The presented concepts for the trigger reception, control logic and readout data handling employ multi-event buffering, in-stream data processing, extensive diagnostics, and advanced features of modern FPGAs to build a robust, high-performance system that can conduct the high-bandwidth readout of the TRD with maximum stability and minimized dead time. The work summarized here not only includes the complete process from the conceptual layout of the multi-event data handling and segment control, but also its implementation, simulation, verification, integration alongside the trigger-related works presented in [99], and commissioning. It furthermore covers the system upgrade for the second data taking period and presents an analysis of the system performance in Run1 and Run2.

During a run, the operation of the ALICE detector is synchronized by trigger sequences issued by the central trigger processor. Consequently, all 18 GTU segments require a control logic that independently steers the readout and trigger-related activity in the segment according to these trigger sequences. With a study of the individual sources of dead time in the TRD processing chain and by exploiting characteristics of the triggering scheme in ALICE, a robust two-stage segment control unit is implemented that generates all needed steering signals and timeframes. The control unit and the preceding trigger reception unit are built to support multiple interlaced trigger sequences, and therefore the simultaneous buffering and handling of data from multiple events. As a result, fluctuations of the input event rate can be derandomized, thus mitigating the negative effects on dead time that originate from the readout stage. By implementing the trigger reception directly on the GTU FPGA nodes, the capability to detect and trace illegal elements in the received sequences can be built in. Consequently, any unnoticed recording of data with potentially corrupted event identification can be prevented and the analysis of real, erroneous situations in the production system provides the required input to harden the design of all entities involved.

The segments implement the reception of data pushed by the detector front-end on 1080 optical links at the full aggregate net sender bandwidth of approximately 2.16 Tbit/s and its buffering in multiple event-associated slots in the SRAMs. Despite the high bandwidth, the handling of event data in a segment is not limited to a simple aggregation and forwarding during readout. By processing the data originating from detector chambers that operate in a harsh radiation environment in-stream and by potentially reshaping it, the system is made tolerant towards violations of the input link transmission format and the downstream design complexity is reduced. This leads, directly and indirectly, to a significantly improved running stability. By providing appropriate buffering structures to internally gather event meta data and augmenting the detector data with it during the local event building process, valuable information can be provided already online to the earliest stages of subsequent processing. Examples here are status flags and CRCs, which allow quick assessments of the chamber operation and data transport, or the calculated global tracks, which form the basis for a HLT trigger component and the offline evaluation of the trigger.

Data path resources shared with the time-critical trigger application realize the high-priority transmission of trigger-related data at low latency. The latter is achieved by operating the backplane in a parallel, source-synchronous fashion and employing a line-wise dynamic delay calibration mechanism to ensure optimal clock-to-data alignment and automatically account for delay variations over all node pairs within the system. As a result, bit error rates for the data transport through a segment and over the DDL have been measured to be less than 10^{-15}. Despite the shared resources and concurrent activity of trigger and readout, the design of the data path and its readout interface allow to utilize 99.30 % (99.97 % for Run2) of the available output bandwidth. Consequently, the readout time per event, and thus the readout-related dead time, is minimized.

As part of a large, data-driven experiment, whose requirements change over its runtime, substantial parts of the firmware are continuously under development in parallel to the system being used in production data taking. With comparably short timeframes available to deploy upgrades and high demands with respect to the correctness and stability of operation, an efficient development and verification process is indispensable. By using a scripted system-level firmware simulation that wraps the top-level entities of the nodes and provides realistic, dynamic stimuli for all external systems, the regular data flow over a wide range of input parameters, but also the behavior of the GTU with respect to erroneous conditions can be tested. Building in extensive, configurable test/verification features and combining them with suitable verification software enables a rapid, semi-automated verification of the designs with high statistics at large rates in-system as well as offline. A staged approach with system-level simulation and thorough verification first in the test system, then in the full-scale production system with all nodes, ensures quick design cycles and leads to carefully tested designs going into production. The extensive integrated diagnostics, which is implemented in each node in the context of the embedded PPC system, and the configurable built-in tests have proven invaluable on multiple occasions to review and understand, on a full system scale, problems such as illegal trigger sequences or diminished readout bandwidths. As illustrated by Figure 8.1, the GTU has been in continuous operation over the entire runtime of the LHC. With a thorough verification in place, it was possible to gradually upgrade from initial designs offering only basic functionality to the optimized Run2 designs featuring multi-event buffering, various trigger algorithms and increased readout capabilities without obstruction to the activities of the experiment.

In order to operate the GTU in the context of the control system of the experiment, a system is required that is capable of initializing, configuring and safely operating a heterogeneous system with more than hundred configuration endpoints. The presented layered architecture of the GTU DCS with worker node FSM service, segment-oriented controls on the DCS board and PPC-supported configuration on the FPGA nodes themselves meets these demands, yet furthermore provides sufficient flexibility to incorporate ongoing developments and gives full low-level access to the developer. It not only integrates all elements of the GTU into the higher-level control system by presenting a simple, abstracted FSM interface for automated operation, but also enables run-type-specific configurations with full access to all low-level control networks in the system and is therefore highly adaptable.

By modeling the operation of the GTU in an event-based, verified Monte Carlo simulation, the performance of the TRD readout system can be analyzed with respect to dead time, achievable, and maximum readout rates for different running scenarios. In typical Pb-Pb runs with a mix of semi-central and central events in Run1, the aggregate DDL output bandwidth of 3.33 GiB/s enables absolute maximum readout rates of approximately 425 Hz. With multi-event buffering and a careful design of the data path, the TRD readout dead time contribution could be reduced to be typically below 5 % at the target readout rate for Run1 of around 200 Hz. The effective aggregate DDL2 output bandwidth of 6.50 GiB/s is the result of an initially unforeseen, yet very successful system upgrade. Despite the expected increase in multiplicity and event size for Run2, it increases the absolute maximum readout rate in a typical Pb-Pb run to about 670 Hz. The low readout-induced dead time, which typically stays below 10 % and usually coincides with dead time contributions of other detectors in conjoint operation at the target readout rates of 450–500 Hz indicates ample system performance for a successful participation in Run2.

The solutions presented here are implemented in the firmware designs of the GTU. In Run1, see Figures 8.2 and 8.1, the GTU accumulated a total of 572 days of actual stable commissioning and physics data taking. During that time, $2.3 \cdot 10^9$ events in various collision schemes with an aggregate data volume of approximately 0.55 PB were recorded to mass storage. By beginning of Run2 in 2015, installation of the TRD front-end electronics was finalized to achieved full azimuthal coverage and the Run2 firmware designs of the GTU were commissioned. Since that time, the GTU again successfully conducts buffering and local event building for the TRD in a correct, fast and highly dependable fashion, and therefore renders possible an efficient data taking delivering particle collision data for physics analyses.

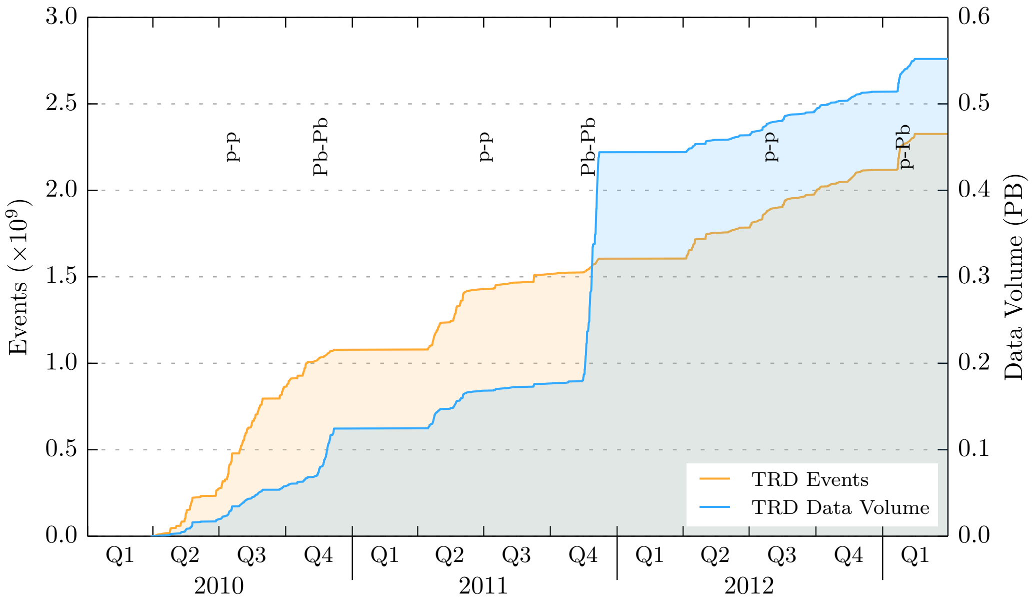


Figure 8.1: The accumulated number of physics events (orange) and the corresponding data volume (blue) recorded by TRD in runs with beam from LHC in the time period between Jan. 1, 2010 and Mar. 31, 2013 with TRD participating as readout detector. Periods of LHC operation (gray background) with different types of beams (p-p, Pb-Pb, p-Pb) are interrupted by shutdown/maintenance periods with no LHC beam production (white background). About one-third of the recorded data volume results from p-p runs.

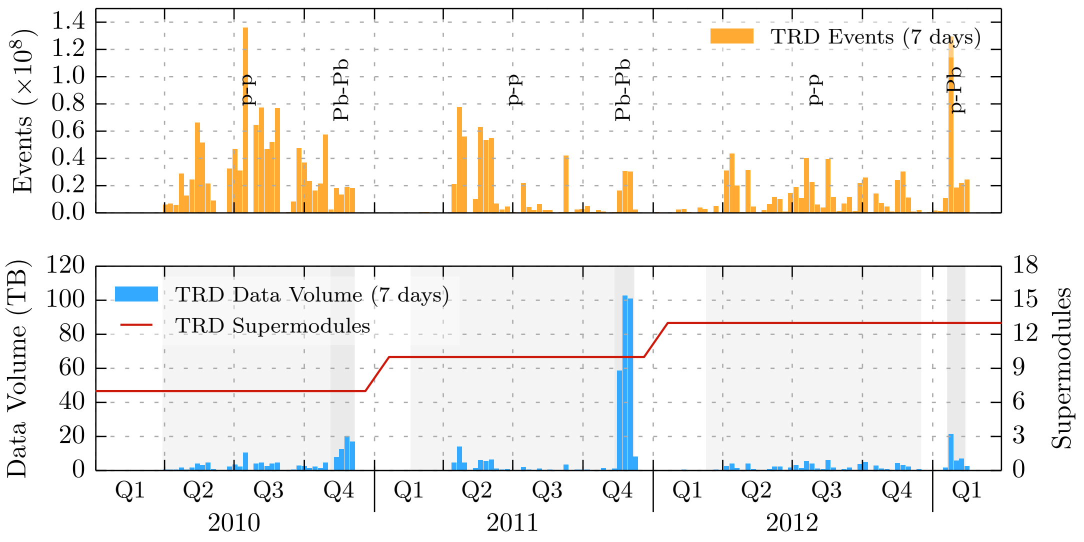


Figure 8.2: Number of events recorded by TRD and corresponding data volume integrated over a period of seven days. Run1 saw the installation of three new TRD supermodules during each of the 2010 and 2011 winter shutdowns. GTU segments without associated supermodule were powered off to prevent unnecessary aging of the electronics.

Acknowledgements

Working in the community of the ALICE Collaboration in anticipation of the LHC start-up, and then seeing the detector and the accelerator commence standard operation, has both been a pleasure and a privilege. With conclusion of this thesis, there are many I am indebted to for encouragement, support and opportunities given.
First of all, I want to thank my supervisor, Prof. Dr. Volker Lindenstruth, for the opportunity to join his group and pursue my doctoral thesis. With his continuous trust and support, and with a lot of freedom, working on this project has been a unique, sometimes challenging, but very exciting, rewarding and instructive experience.
Special thanks go to Felix Rettig, who focused on the development of the GTU triggers, for the many discussions, the seemingly endless, yet quite fruitful, debugging sessions and the close collaboration over all these years. The same is true for Dr. Jochen Klein, who brought invaluable expertise on the detector front-end and excellent ideas to commissioning and discussions. Getting it all to work resulted in quite some memorable moments.
I am indebted to Dr. Jan de Cuveland and Dirk Hutter not only for proof-reading this thesis and their contributions to the GTU, but also for the ongoing support, for sharing their knowledge, for the side projects, and for becoming friends over many coffees.
Thanks go to everybody who contributed to the TRD, to the colleagues at Point 2 and to past and current members of the group at FIAS. I have always enjoyed the constructive atmosphere and collaborative spirit.
To the many people I spent much time with in and around CERN, Dr. Christian Lippmann, Dr. Chiara Zampolli, Torsten Alt, Dr. Jochen Klein, Dr. Øystein Haaland, Johannes Lehrbach, Dr. Magnus Mager, Jens Steckert, Dr. David Rohr, Dr. Jochen Thäder, Timo Breitner, Heiko Engel, Artur Szostak, Dr. Indranil Das, Dr. Yvonne Pachmayer, Dr. Jorge Mercado, I would like to say thank you for plenty of great lunches, discussions, hikes, evenings, parties, drinks, jokes, and fun. That also extends to Dr. Sebastian Kalcher, Dr. Sebastian Manz, Dr. Nicolai Schroer, and Katja Regelmann. You all have made this a time with many good memories I happily look back to.
At last, I would like to thank my family for the love, encouragement and continuous support.

This work has been sponsored by the Wolfgang Gentner Programme of the German Federal Ministry of Education and Research (grant no. 05E12CHA).

Appendix A

FPGA Resource Usage

A.1 Available Resources

Table A.1 summarizes important key figures for Xilinx Virtex-4 FX100 devices.

Device	Xilinx Virtex-4 XC4VFX100-11FFG1152C
Slices	42 716
Logic Cells	94 896
Block RAMs	376 × 18 kbit
Distributed RAMs	659 kbit
RocketIO serial transceivers	20
PowerPC cores	2
User I/O pins	768
DCMs	12
PMCDs	8

Table A.1: Xilinx Virtex-4 FX100 key figures. Excerpt from [118].

A.2 Used Resources

An overview of the resources used in the SMU and TMU FPGA firmware designs is given in Table A.2. Figures A.1 and A.2 give a graphical representation of the FPGA structure and the resources used for the various design parts of TMU and SMU, respectively.

The number of occupied slices[1] in the TMU is at a relatively high level of about 83 %, mostly due to the implementation of the tracking algorithm and the 128-bit wide, pipelined data path. Slice occupation in the SMU is at 60 %, which would enable even further upgrades. Although only one MGT (GT11) is used on the SMU for the DDL2, all four MGTs whose I/Os are routed on the PCB are instantiated to keep them operational [114, 111].

[1] An occupied slice does not necessarily require all of its resources to be used, e. g. only one of the two LUTs of a slice might be used.

TMU FPGA, Run2

Resource	avail.	used		Resource	avail.	used	
Slices	42176	35150	83 %	PowerPCs	2	2	100 %
as SLICEM	21088	2390	11 %	GT11	20	12	60 %
DCM	12	4	33 %	PMCD	8	1	13 %
RAMB16	376	322	85 %	DSP48	160	20	13 %

SMU FPGA, Run2

Resource	avail.	used		Resource	avail.	used	
Slices	42176	25581	60 %	PowerPCs	2	2	100 %
as SLICEM	21088	1400	6 %	GT11	20	4	60 %
DCM	12	4	33 %	PMCD	8	3	38 %
RAMB16	376	180	47 %				

Table A.2: Overview of logic, memory and clocking resources used by the TMU and SMU FPGA designs as used in Run2.

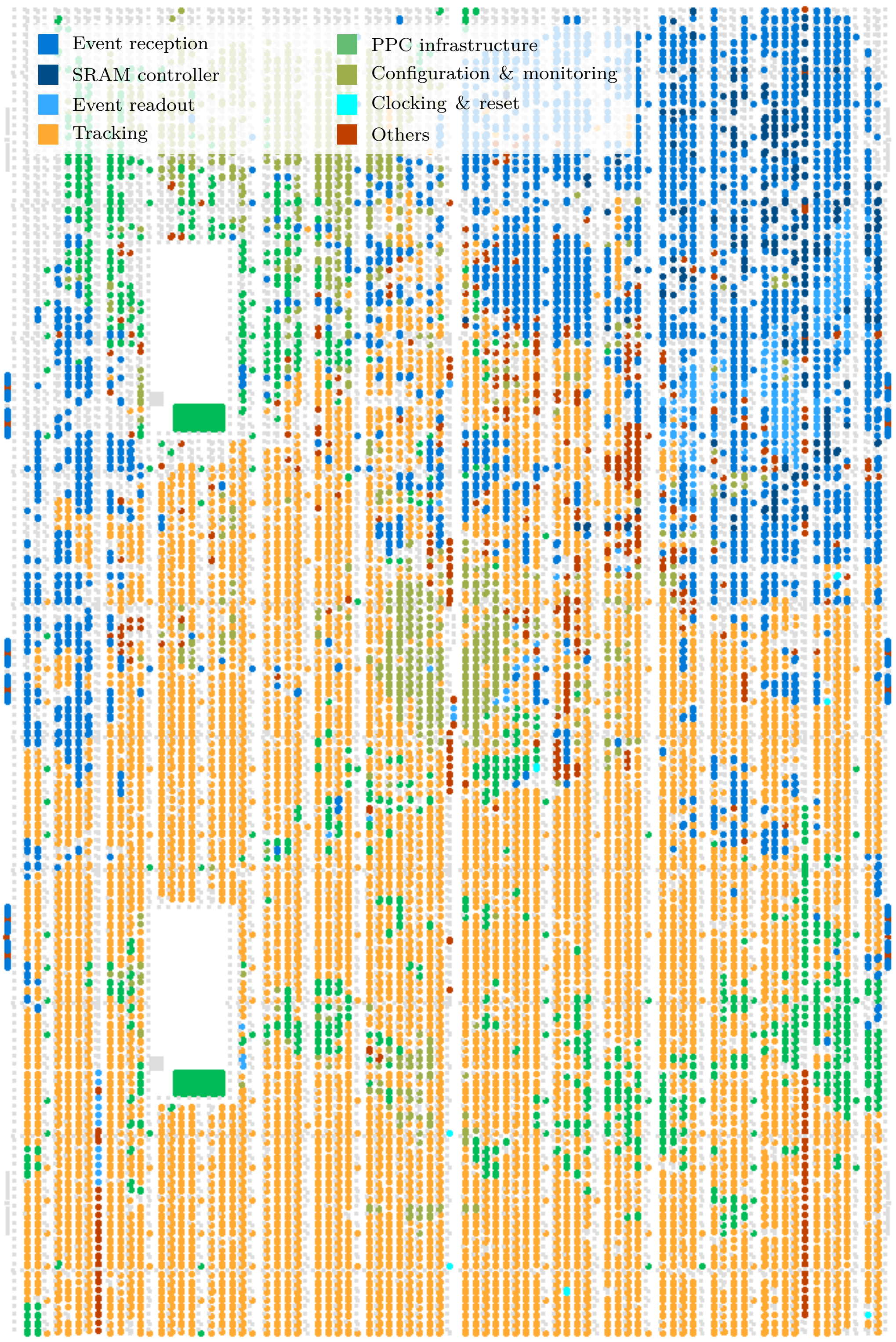


Figure A.1: Resources used in the Xilinx Virtex-4 FX-100 of the TMU after the Run2 upgrade. Slice usage is about 83 %.

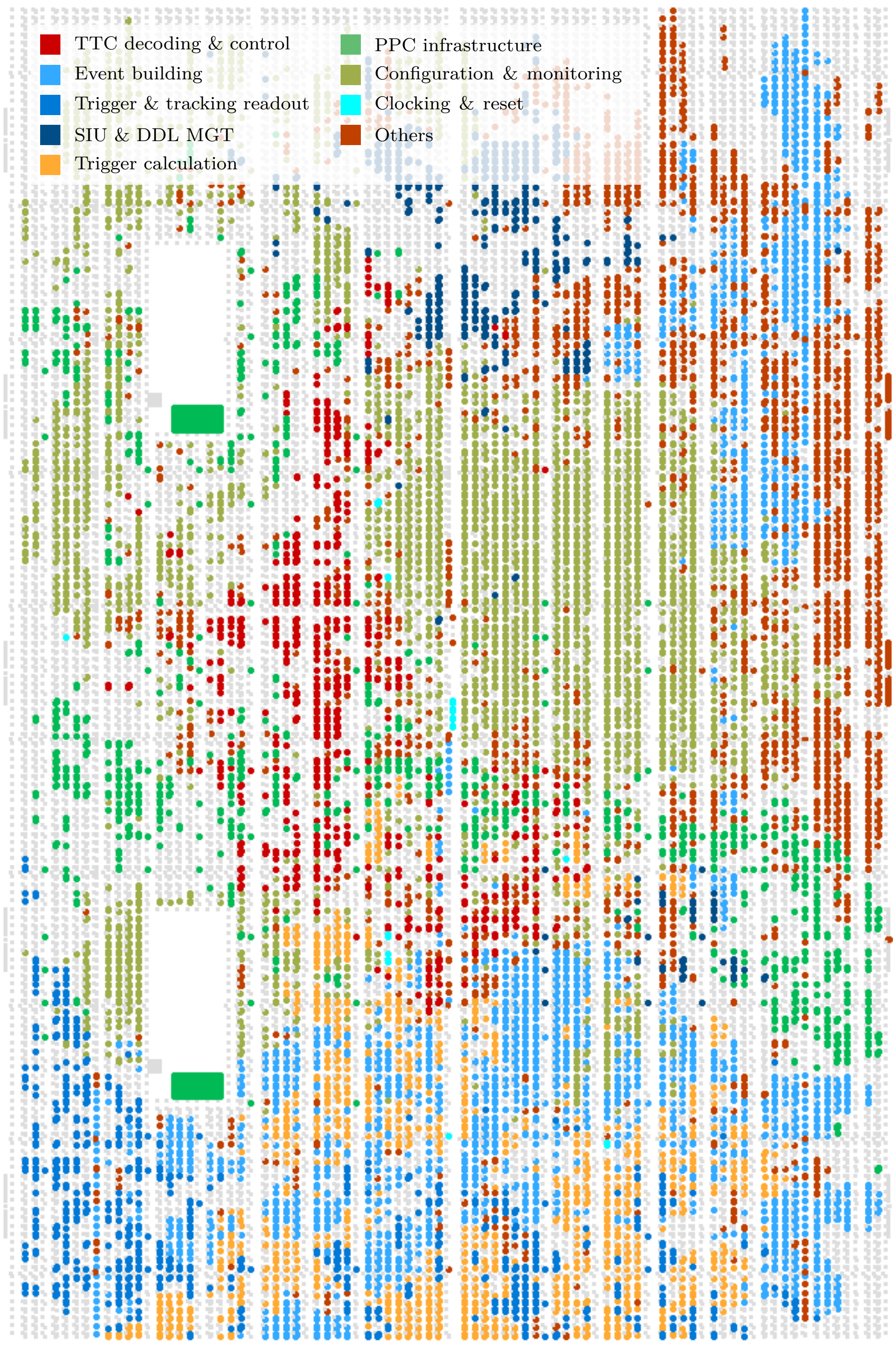


Figure A.2: Resources used in the Xilinx Virtex-4 FX-100 of the SMU node after the Run2 upgrade. Approximately 60 % of the slices and 47 % of the BRAMs are occupied.

Appendix B

TRD Event Fragment Data Format

The event fragment that contains the event data of a TRD supermodule and is generated by the associated GTU segment for transmission to the DAQ, has the following structure:

Size [words]	**Content**	
8	Common Data Header	
1	Supermodule Index Word	
variable ($\mathtt{s}$)	Supermodule Header	
1	Stack 0 Tracking Index Word	only present if (1), $i = 0$
variable ($\mathtt{s}_{\mathrm{tk},0}$)	Stack 0 Tracking Header	
	. . .	
1	Stack 4 Tracking Index Word	only present if (1), $i = 4$
variable ($\mathtt{s}_{\mathrm{tk},4}$)	Stack 4 Tracking Header	
1	Trigger 0 Index Word	only present if (2) || ((3), $i = 0$
variable ($\mathtt{s}_{\mathrm{tg},0}$)	Trigger 0 Header	
	. . .	
1	Trigger 11 Index Word	only present if (2) || ((3), $i = 11$
variable ($\mathtt{s}_{\mathrm{tg},11}$)	Trigger 11 Header	
1	Stack 0 Index Word	only present if (4), $i = 0$
variable ($\mathtt{s}_0$)	Stack 0 Header	
	. . .	
1	Stack 4 Index Word	only present if (4), $i = 4$
variable ($\mathtt{s}_4$)	Stack 4 Header	
variable	Stack 0 Data Content	only present if (4), $i = 0$
2	Stack Separation Marker	only present if (5), $i = 0$
	. . .	
variable	Stack 4 Data Content	only present if (4), $i = 4$
2	Stack Separation Marker	only present if (5), $i = 4$
1	Trailer Index Word	only present if (6)
variable ($\mathtt{s}_{\mathrm{trl}}$)	Trailer Data Content	

Each word is 32 bit.

	Condition
(1)	Version of the supermodule header $\geq$ `0xc` and readout of tracking information enabled and readout of stack i enabled
(2)	Version of the supermodule header $\geq$ `0xc` and readout of trigger information always enabled and trigger algorithm i enabled
(3)	Version of the supermodule header $\geq$ `0xc` and readout of the trigger information only when fired and trigger algorithm i enabled and trigger algorithm i fired
(4)	Readout of stack i enabled
(5)	Version of the supermodule header $\geq$ `0xd` and readout of stack i enabled
(6)	Version of the supermodule header $\geq$ `0xc` and readout of the trailer enabled

The presented data format is an excerpt from [42].

Appendix C

Performance Measurements

C.1 Characterization of FEE Transmission and Run1 Readout

To acquire credible measurements regarding the transmission of data from detector to GTU, the readout interface and associated processing times, a series of test runs with modified, non-standard GTU designs were conducted. These designs inject time stamps for selected events in the system into the data stream, so that a chronological sequence of these events can be reproduced offline.

Goal of the subsequent analysis was to measure/characterize

- the transmission behavior of the FEE, especially with respect to
  - transmission idles/pauses,
  - transmission duration as function of the link data volume, and
  - the effective FEE link input bandwidth,
- the readout duration as function of the event fragment size,
- dead time related to the L0-L1 timeframe, and
- dead time caused by the event readout.

From measurements of the bandwidths and transmission behavior ultimately follow the processing times for the L0-L1 stage and readout stage (in Run1), which serve as input to the event-based Monte Carlo dead time simulation.

The required modifications of the firmware designs included addition of

- time stamps on the SMU for
  - the arrival of L0 triggers,
  - the end of FEE data transfer,
  - the raise and release of the busy signal,
  - the start and stop of the readout of a given event, and
- time stamps on the TMU for the arrival of the first and last data word on each link,

plus the required support infrastructure and buffers to manage time stamps of multiple events and inject them into the respective event fragment for readout. These time stamps were extracted from the resulting event data stream[1] and converted into a chronological list of events that includes all system events on all GTU segments participating in a given run. Combined with an analysis of the recorded event data itself, this list enables an evaluation of bandwidths, processing times and dead time generated by individual system components.

Several series of test runs were recorded with the production setup in a global readout partition using full CTP trigger logic. In the runs, the TRD pretrigger system is used to supply a start signal at a controllable rate to the CTP, and thereby conduct a sweep of the primary event rate. Rejects at the L1 trigger stage are realized by supplying a random signal from GTU to the CTP that exhibits the desired accept/reject ratio. Three TRD supermodules were available for testing and participated in the runs. To mimic realistic conditions, they produce event fragments of random sizes, which fluctuate roughly around an average fragment size s_{frag}. This is achieved by loading a FEE configuration that has an intentionally lowered noise threshold adjusted to generate roughly the desired fragment size.

Series			1	2	3	4
Runs		first	198 494	198 526	198 546	198 584
		last	198 510	198 542	198 557	198 596
N			4	4	1	4
a_{L1}			0.01	0.05	0.05	0.05
a_{L2}			1.00	1.00	1.00	1.00
s_{frag}	[KiB]	min	516	517	518	230
	[KiB]	max	550	523	559	234
$t_{\text{rx, avg}}$	[µs]	min	79.80	79.59	76.19	51.13
	[µs]	max	88.56	81.55	90.45	55.44
$t_{\text{ro, avg}}$	[ms]	min	2.65	2.66	2.66	1.19
	[ms]	max	2.85	2.71	2.88	1.21

Table C.1: Overview of the series of runs used to evaluate FEE behavior, event buffering performance, and readout in Run1. The series vary with respect to the configured L1/L0 ratio, the number of active event buffers N, and the average event fragment size s_{frag}. Min and max show the minimum and maximum average values of a run for s_{frag}, $t_{\text{tx,avg}}$ and $t_{\text{ro,avg}}$ encountered in the runs of a series.

[1]Following the concept of event identification in the experiment, a combination of local orbit and bunch counter is used as time stamp. Furthermore, events in the individual chunks recorded by the DAQ are not chronologically ordered. Consequently, the time stamps were extracted and, after calibration using global information from the CDH, sorting and corrections (e. g., for potential wrapping of the limited width hardware counters) converted into a chronological list of events that are attributed an absolute 64-bit time value.

Four series of runs were recorded, see Table C.1. Each run within a series corresponds to a selected primary event rate. Series of runs differ with respect to certain parameters that have been varied:

- The L1/L0 trigger ratio a_{L1} (Series 1: $a_{\mathrm{L1}} = 0.01$, others $a_{\mathrm{L1}} = 0.05$)
- the number of event buffers N (Series 3: $N = 1$, others $N = 4$), and
- the event fragment size s_{frag} (Series 4: about 230 KiB, others about 525 KiB).

Unfortunately, the half-chamber noise slightly increases with higher primary event rates and leads to non-constant average event sizes and processing times within a series. The dead time simulation for a series therefore uses the series average values for t_{rx} and t_{ro}, see Figure 7.16.

C.2 Run2 DDL2 Measurements

Due to unavailability of the production setup for longer measurements, the performance measurements for the upgraded Run2 designs are carried out using the test setup. The test aims at measuring the effective DDL2 output bandwidth, the various dead time contributions and, implicitly, the performance of the data path improvements. Measured results are shown in Figure 7.17.

The test setup provides a more controllable environment, yet comes with certain restrictions concerning the triggering. These measurements are automatized with by a script controlling

- the setup of the TMU link-level PRNGs,
- the setup of the LTU hardware trigger emulation,
- the sweep over the primary event rate (by altering the LTU emulator start signal rate threshold), and
- the actual measurement and extraction of the desired quantities from the PPC system.

The TMU PRNGs deliver event data of a fixed size, since a change of the delivered data volume is not foreseen on an event-by-event basis at high rates. The emulator continually replays a set of predefined trigger sequences stored in memory. These sets are manually specified to produce the desired trigger ratios. A pseudo random signal, generated internally by the LTU, is selected as start signal source. Although the emulator memory can be loaded with a set containing a mixture of L0, L2A and L2R sequences at the desired ratios, the repetitive replay of a limited number of sequences does not yield sufficient randomness to produced undistorted measurements with trigger accept/reject ratios $0 < a_i < 1$, with $i = [\mathrm{L1}, \mathrm{L2}]$. When combined with a deterministic behavior of the readout interface and link volumes of constant size (and thus, processing time) delivered by the PRNGs, the limited randomness will lead to certain event buffer fill/reject patterns. The resulting measurements show good agreement with the results of the event-based Monte Carlo simulation in the regions where the workload is close to $x_{\mathrm{2nd}} = 0$ (buffers mostly empty) and $x_{\mathrm{2nd}} = 1$ (buffers mostly filled). However, although also the general dead time shape of the curve is similar, the measurements are not comparable with the simulation for $0 \ll x_{2\text{nd}} \ll 1$, since the simulation assumes random accepts/rejects and not repetitive buffer fill patterns. Due to this limitation of the measurement setup, the trigger ratios used for comparison with the simulation are fixed to $a_i = 1$.

The desired quantities are gathered internally with help of the integrated DDL2 performance measurement entities over a period of approximately 12 s for each data point. All values related to DDL2 are measured at the crossing from SMU data path to integrated SIU core, see Figure 4.17. Thus, an output bandwidth higher than the effective DDL2 bandwidth might be observed when the SIU decoupling buffer and output FIFO are being filled.

DDL2 Performance Measurement Example Listing 4 presents a concise overview of the DDL2 performance that allows to quickly pinpoint readout-related issues, such as periods of increased backpressure originating downstream at the DAQ or HLT. It is available live via the SMU console application.

The performance measurement is implemented in the SMU using the PPC-assisted statistics acquisition presented in Section 6.3.3. Each measuring interval ends with the last word of an event written to the SIU decoupling buffer. The corresponding set of values, which contains, e. g., the interval duration and the number of words transmitted, is immediately pushed in the following cycle and a new measuring interval begins.

C.3 Dead Time Simulation with Lower Average Event Fragment Size

The dead time simulation of Section 7.6 is repeated for the less demanding case of the semi centrality distribution. The distribution, see Figure 7.12, exhibits an average event centrality of $c_{\text{avg}} = 29.93\,\%$, which results in average event fragment sizes of 310 KiB for Run1, and 387 KiB for Run2.

Figures C.1 and C.2 show the results of the performance simulation for the semi input centrality distribution. For general remarks on the simulation an plots see Section 7.6. For an overview of the simulation parameters refer to Table 7.1.

```
smu00:/ > gtucom 2 ddl
Stats last event@SIU
 EvSize   597 kiB/153059 words
 TxTime  1572 us
 EffBw    369.8 MiB/s
 Xoff      19 %
 Ineff      0 %
 NUsed      0 %

Stats over last 16.0s @ SIU
Ival - Rate  - Events - Avg size - Eff DDL - Avg DDL - Avg XOFF - Avg TxTime - Ineff
       [Hz]              [kiB]    [MiB/s]   [MiB/s]      [%]        [ms]         [%]
  0    631.5     1263     597.8     369.8     369.7      19.2       1.572        0.0
 -1    630.0     1260     597.8     369.8     369.5      19.2       1.572        0.0
 -2    631.0     1262     597.8     369.8     369.7      19.2       1.572        0.0
 -3    630.5     1261     597.8     369.8     369.5      19.2       1.572        0.0
 -4    630.0     1260     597.8     369.8     369.7      19.2       1.572        0.0
 -5    631.5     1263     597.8     369.8     369.7      19.2       1.572        0.0
 -6    300.0      600     597.8     369.8     369.7      19.2       1.572        0.0
 -7      0.0        0

Stats (total):
 Interval       : 13.3 s
 Events         : 8479
 IntervalTxVol  : 4.8 GiB
 IntervalTxTime : 13332.349 ms

Stats (avg per event):
 TxTime  : 1.572 ms
 AvgSize : 597.2 kiB
 EffBW   : 369.8 MiB/s
 AvgBW   : 369.7 MiB/s
 AvgXOFF : 19.20 %
 AvgIneff: 0.00 %
```

Listing 4: DDL2 performance acquired in the first approx. 13 s after the start of a run (test system, integrated pattern generators at fixed event size). Listed are the details for the last event, measurements of the last eight two-second intervals and the total statistics in absolute and relative numbers.

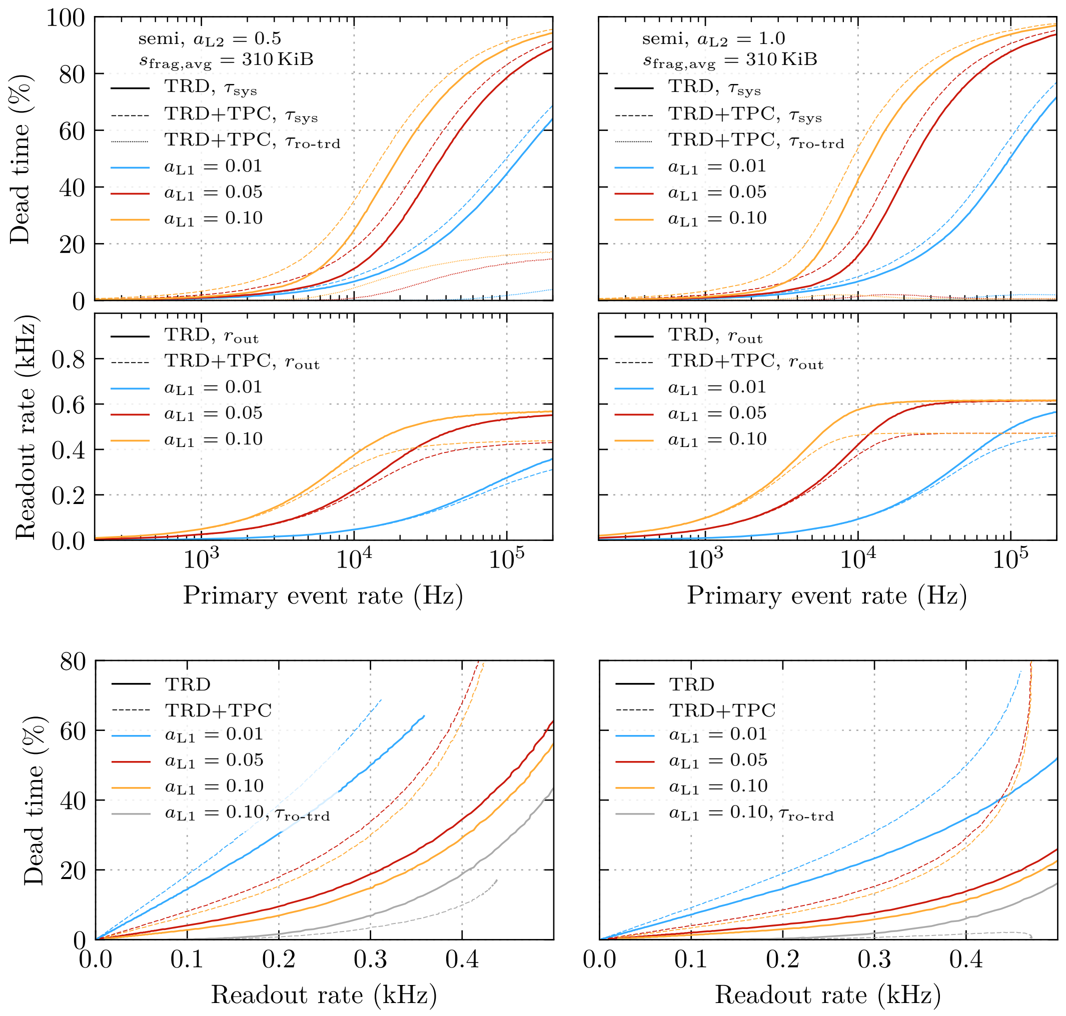


Figure C.1: Run1 simulation with pbpb and semi input distribution for two L2 accept ratios, $a_{L2} = 0.5$ (left column) and $a_{L2} = 1.0$ (right column). The average TRD event fragment size is 439 KiB. Colors denote different L1 accept ratios, event buffers are fixed to $N = 4$. Simulated are 18 TRD supermodules, both in standalone operation (solid) as well as in a common readout partition with the TPC (dashed/dotted).

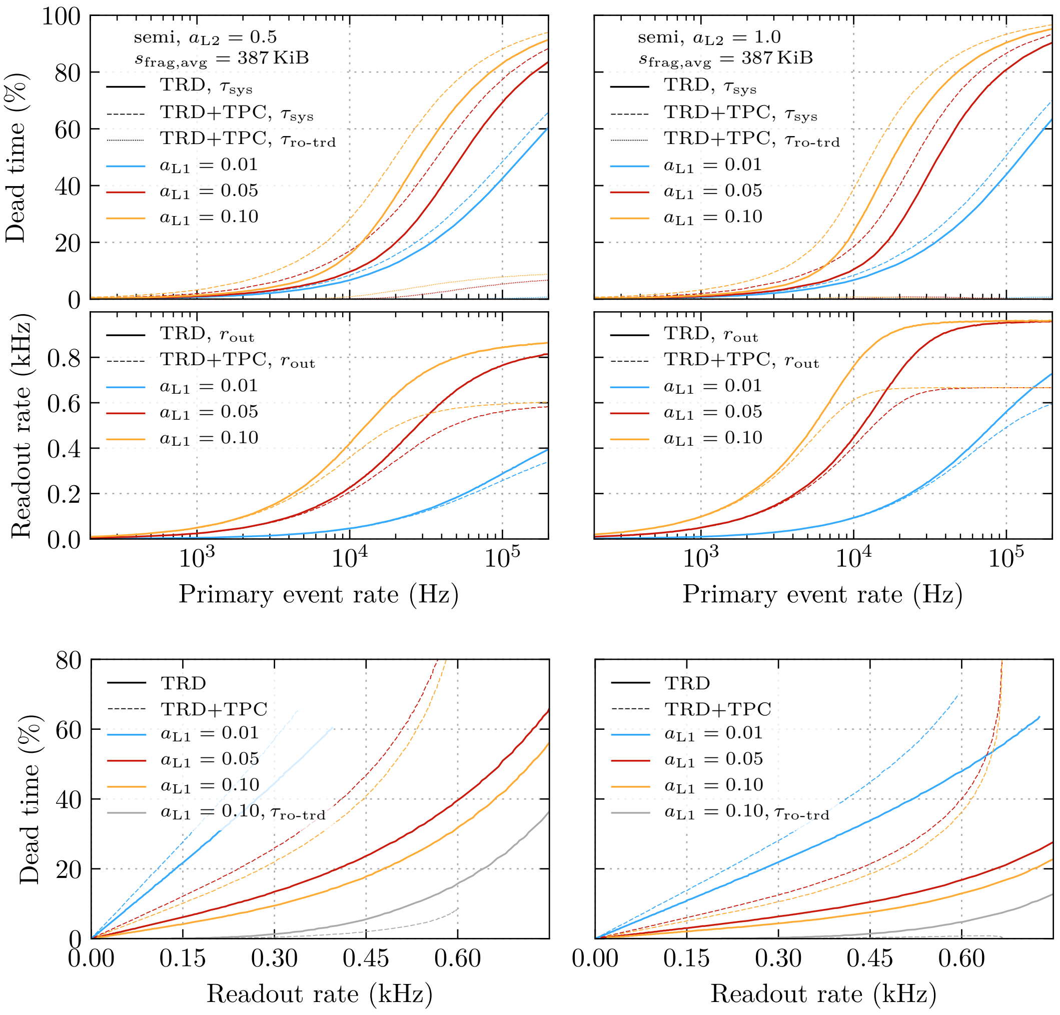


Figure C.2: Run2 performance simulation with pbpb and semi input distribution for two L2 accept ratios, $a_{L2} = 0.5$ (left column) and $a_{L2} = 1.0$ (right column). TRD fragment sizes are 387 KiB. Colors denote different L1 accept ratios, event buffers are fixed to $N = 4$. Simulated are 18 TRD supermodules, both in standalone operation (solid) as well as in a common readout partition with the TPC (dashed/dotted).

Appendix D

Trigger Stimuli in System Simulation

The GTU system simulation framework utilizes a custom, synthesizable trigger emulator, capable of simulating correct as well as erroneous sequences, to provide realistic trigger stimuli. They are suitable for direct input to the top-level firmware SMU and TGU board models. This trigger system has an option to take into account the busy signal to simulate the dynamic behavior of the system with respect to readout throttling.

Stimuli for various scenarios are provided as .xml files, which are converted to a binary form and loaded to the emulator at the start of the simulation. Below is shown an example of such a stimulus file and the corresponding document type definition.

D.1 Example File

```
<?xml version="1.0"?>
<!DOCTYPE root SYSTEM "sequence_def.dtd">

<root>
  <defaults>
    <def_seq repeat="1" skip="0" busy_ignore="0">
      <def_l0 enable="1"/>
      <def_l1 enable="1" time_rel="260"/>
      <def_l1_message enable="1" time_rel="260" wordmask="0x1f">
        <def_l1m_content swc="0" cit="0" spare="0" class_low="0x00c01001"
                         class_high="0x04008" roc="0x0" esr = "0"/>
      </def_l1_message>
      <def_l2_message enable="1" time_rel="550" wordmask="0xff">
        <def_l2m_content swc="0" cit="0" spare="3" class_low="0x0"
                         class_high="0x0" cluster="0x0" bcid="0x0"
                         orbitid="0x0"/>
      </def_l2_message>
    </def_seq>
  </defaults>

  <sequence_list>
    <seq type="L2a" repeat="1"><start time="0x20"/></seq>
    <seq type="L2a"><start time="0x20"/></seq>
    <seq type="L2r"><start time="0x20"/></seq>
    <seq type="L2r"><start time="0x20"/></seq>
    <seq type="L2a" busy_ignore="1"> <!-- ERROR: produces spurious L1 trigger -->
      <start time="0x6"/>
      <l0 enable="0"/>
      <l1_message enable="0"/>
      <l2_message enable="0"/>
      </seq>
      <seq type="L0"><start time="0x10"/></seq>
      <seq type="L0"><start time="0x10"/></seq>
      <seq type="L2a">                <!-- ERROR: produces L2 timeout -->
        <start time="0x20"/>
        <l2_message enable="0"/>
      </seq>
    </sequence_list>
  </root>
```

In the `default` section of the example file, the defaults for all sequences specified in the sequence list are defined. The defaults include the relative timing of individual trigger events within the sequence, trigger message contents, etc., but also control flags for the emulator. The actual sequences to replay are listed in the sequence list. Apart from several ordinary L0, L2A and L2R sequences, the example illustrates two different cases of erroneous trigger sequences. A L2A message with all events but the L1 trigger suppressed produces a spurious L1 trigger, that is sent irrespective of the state of the `busy` signal. It recreates a pattern emitted by the real CTP in the production system during the commissioning phase. Another erroneous L2A trigger sequence suppresses the sending of the complete L2A trigger message and will therefore produce a timeout error. The start time of new sequences is given relative to the start of the previous sequence or emulator start. Individual events within a trigger sequence are timed using the start of the sequence as reference point. All times are converted to absolute times by the converter utility during the `.xml` to `.bin` conversion. The utility furthermore checks that the specified sequence of trigger events is consistent.

D.2 Document Type Definition for Trigger Stimuli Files

```
  <!ELEMENT root (defaults, sequence_list)>

  <!ELEMENT defaults (def_seq)>
  <!ELEMENT def_seq (def_l0, def_l1, def_l1_message, def_l2m_message)>
  <!ELEMENT def_l0 EMPTY>
  <!ELEMENT def_l1 EMPTY>
  <!ELEMENT def_l1_message (def_l1m_content)>
  <!ELEMENT def_l1m_content EMPTY>
  <!ELEMENT def_l2_message (def_l2m_content)>
  <!ELEMENT def_l2m_content EMPTY>

  <!ATTLIST def_seq         repeat NMTOKEN #REQUIRED
                            skip ( 0 | 1 ) #REQUIRED
                            busy_ignore ( 0 | 1 ) #REQUIRED>
  <!ATTLIST def_l0          enable ( 0 | 1 ) #REQUIRED>
  <!ATTLIST def_l1          enable ( 0 | 1 ) #REQUIRED
                            time_rel NMTOKEN #REQUIRED>
  <!ATTLIST def_l1m         enable ( 0 | 1 ) #REQUIRED
                            time_rel NMTOKEN #REQUIRED
                            wordmask NMTOKEN #REQUIRED>
  <!ATTLIST def_l2m         enable ( 0 | 1 ) #REQUIRED
                            time_rel NMTOKEN #REQUIRED
                            wordmask NMTOKEN #REQUIRED>
  <!ATTLIST def_l1m_content swc ( 0 | 1 ) #REQUIRED
                            cit ( 0 | 1 ) #REQUIRED
                            spare ( 0 | 1 ) #REQUIRED
```

```
                                class_low NMTOKEN #REQUIRED
                                class_high NMTOKEN #REQUIRED
                                roc NMTOKEN #REQUIRED
                                esr ( 0 | 1 ) #REQUIRED>
<!ATTLIST def_l2m_content swc ( 0 | 1 ) #REQUIRED
                                cit ( 0 | 1 ) #REQUIRED
                                spare NMTOKEN #REQUIRED
                                class_low NMTOKEN #REQUIRED
                                class_high NMTOKEN #REQUIRED
                                cluster NMTOKEN #REQUIRED
                                bcid NMTOKEN #REQUIRED
                                orbitid NMTOKEN #REQUIRED>

<!ELEMENT sequence_list (seq+)>
<!ELEMENT seq (start, l0?, l1?, l1_message?, l2_message?)>
<!ELEMENT start EMPTY>
<!ELEMENT l0 EMPTY>
<!ELEMENT l1 EMPTY>
<!ELEMENT l1_message (l1m_content?)>
<!ELEMENT l1m_content EMPTY>
<!ELEMENT l2_message (l2m_content?)>
<!ELEMENT l2m_content EMPTY>

<!ATTLIST seq         type ( L0 | L2a | L2r | PP | CL0 | CL2a | CL2r ) #REQUIRED
                      repeat NMTOKEN #IMPLIED
                      skip ( 0 | 1 ) #IMPLIED
                      busy_ignore ( 0 | 1 ) #IMPLIED >
<!ATTLIST start       time NMTOKEN #REQUIRED>
<!ATTLIST l0          enable ( 0 | 1 ) #IMPLIED>
<!ATTLIST l1          enable ( 0 | 1 ) #IMPLIED
                      time_rel NMTOKEN #IMPLIED>
<!ATTLIST l1_message  enable ( 0 | 1 ) #IMPLIED
                      time_rel NMTOKEN #IMPLIED
                      wordmask NMTOKEN #IMPLIED>
<!ATTLIST l2_message  enable ( 0 | 1 ) #IMPLIED
                      time_rel NMTOKEN #IMPLIED
                      wordmask NMTOKEN #IMPLIED>
<!ATTLIST l1m_content swc ( 0 | 1 ) #IMPLIED
                      cit ( 0 | 1 ) #IMPLIED
                      spare ( 0 | 1 ) #IMPLIED
                      class_low NMTOKEN #IMPLIED
                      class_high NMTOKEN #IMPLIED
                      roc NMTOKEN #IMPLIED
                      esr ( 0 | 1 ) #IMPLIED>
<!ATTLIST l2m_content swc ( 0 | 1 ) #IMPLIED
                      cit ( 0 | 1 ) #IMPLIED
                      spare NMTOKEN #IMPLIED
                      class_low NMTOKEN #IMPLIED
                      class_high NMTOKEN #IMPLIED
                      cluster NMTOKEN #IMPLIED
                      bcid NMTOKEN #IMPLIED
                      orbitid NMTOKEN #IMPLIED>
```

Appendix E

GTU Installation

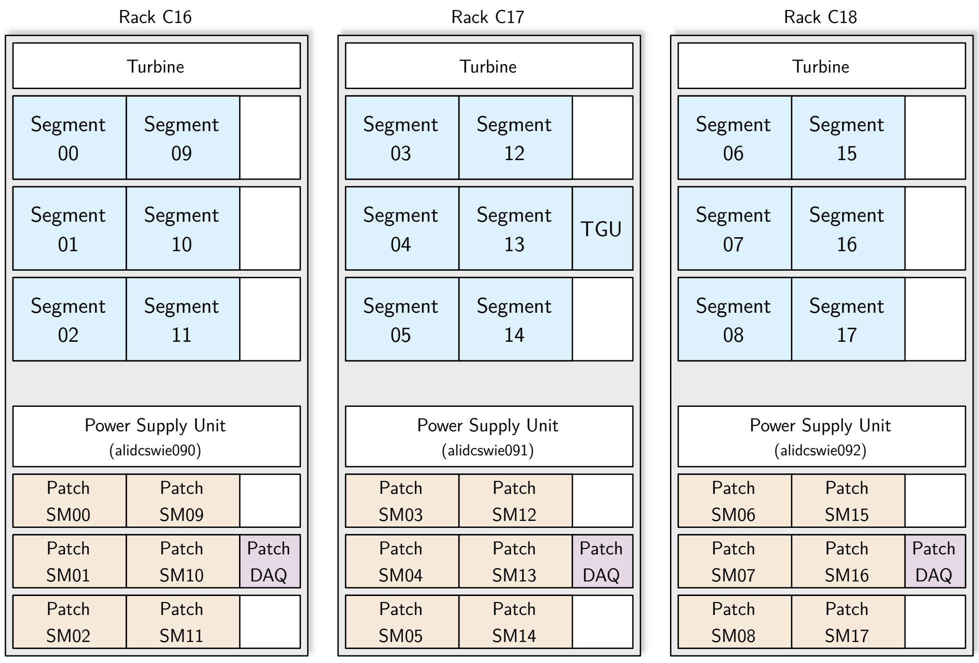


Figure E.1: Schematic overview of the GTU installation (blue) in the C-area of the ALICE cavern (non-GTU related entites in racks are not shown).

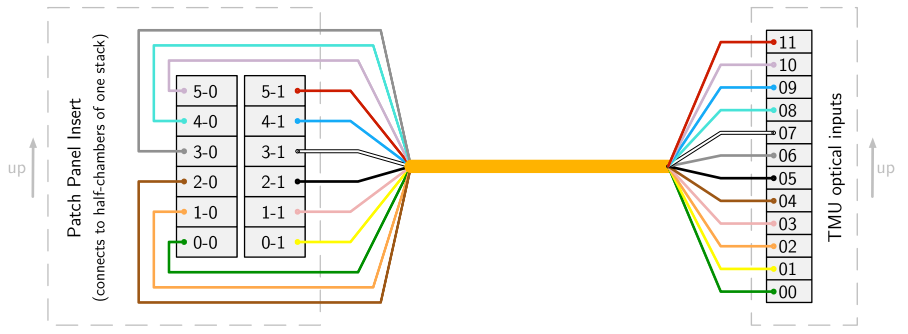


Figure E.2: Patching of the 12 fibers of a stack to the optical receivers on a TMU.

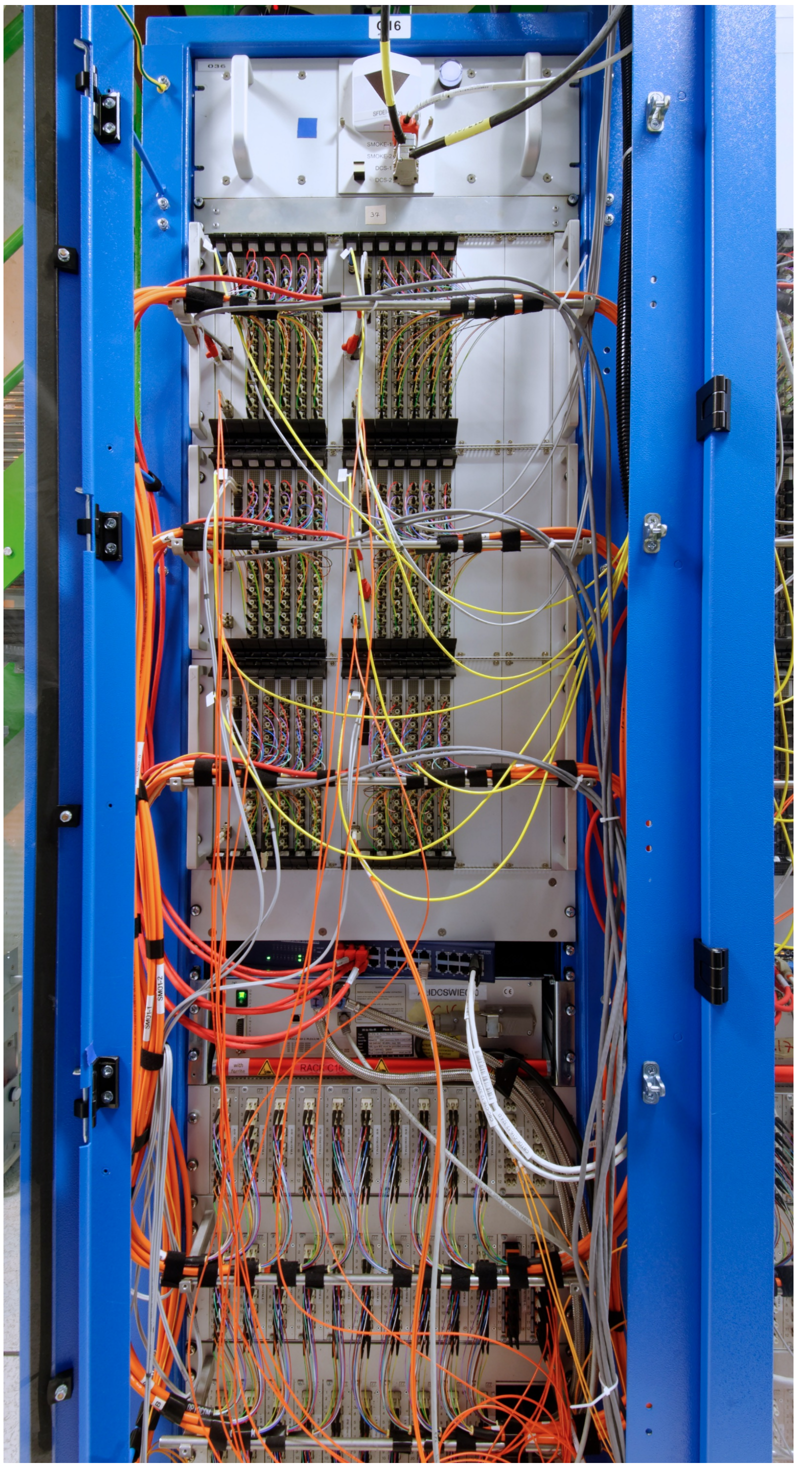


Figure E.3: Rack C16 front view with (from top to bottom): Turbine for rack cooling, six fully installed GTU segments, power supply, and optical patch panels.

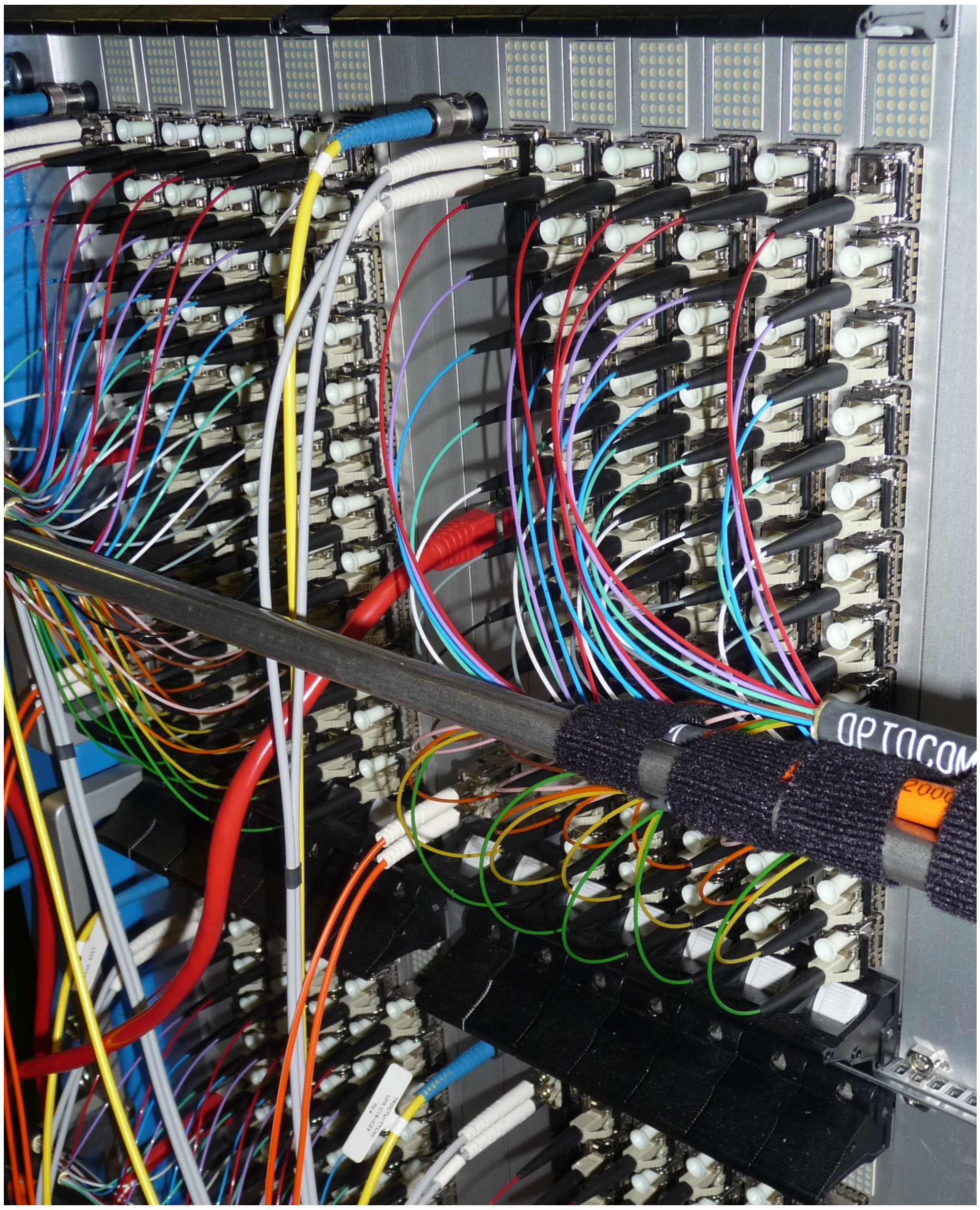


Figure E.4: Close-up of two installed GTU segments. SMU and five TMUs of a segment are connected by the rear-side LVDS and CompactPCI backplanes. The TMUs connect to five detector stacks with a total of 60 optical links (thin, colored fibers). The SMU has connections to the HLT/DAQ, DCS network, and TTC system.

List of Figures

List of Tables

List of Acronyms

8P8C	8 Position 8 Contact
ACORDE	ALICE Cosmic Ray Detector (ALICE subdetector)
ACT	ALICE Configuration Tool
ADC	Analog-to-Digital Converter
ALICE	A Large Ion Collider Experiment (LHC experiment)
AMORE	Automatic Monitoring Environment
ASIC	Application-Specific Integrated Circuit
ATLAS	A Toroidal LHC Apparatus (LHC experiment)
BC	Bunch Crossing
BMM	Board Merger
BRAM	Block RAM
BRC	Broadcast Command (TTC frame format)
CDH	Common Data Header
CERN	European Organization for Nuclear Research
CMS	Compact Muon Selenoid (LHC experiment)
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CRORC	Common Readout Receiver Card Readout card in use for DAQ and HLT for Run2 [PCIe Gen2 x8]
CTP	Central Trigger Processor
DAQ	Data Acquisition
DATE	Data Acquisition and Test Environment ALICE DAQ software framework
DCM	Digital Clock Manager
DCR	Device Control Register Bus

DCS	Detector Control System
DDL	Detector Data Link (Multi-purpose link between FEE and RORC)
DDL2	Detector Data Link 2 (Upgraded DDL used in Run2)
DDR	Double Data Rate
DDR2	Double Data Rate 2
DIM	Distributed Information Management
DIU	Destination Interface Unit (DDL endpoint)
DQM	Data Quality Monitoring
DRORC	DAQ Readout Receiver Card DAQ Readout receiver card for Run1 [PCI-X, PCIe]
DRP	Dynamic Reconfiguration Port
ECS	Experiment Control System
EDA	Electronic Design Automation
EMCal	Electromagnetic Calorimeter (ALICE subdetector)
EoD	End-of-Data
EoR	End-of-Run
FDCE	D Flip-Flop
FEC	Front-End Card
FEE	Front-End Electronics
FIFO	First-in, First-out
FMD	Forward Multiplicity Detector (ALICE subdetector)
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
FXS	File Exchange Server
GDC	Global Data Concentrator
GTL	Gunning Transceiver Logic
GTU	Global Tracking Unit
GTX	Gigabit Transceiver
HCM	Half-Chamber Merger
HDL	Hardware Description Language

HLT	High-Level Trigger
HMPID	High Momentum Particle Identification
HV	High-voltage
I^{2}C	Inter-Integrated Circuit
IAC	Individually Addressed Command (TTC frame format)
ID	Identification
IDELAY	Input Delay Element
IEEE	Institute of Electrical and Electronics Engineers
ILA	Integrated Logic Analyzer
IP	Intellectual Property
ISERDES	Input Serial-to-Parallel Converter
ITS	Inner Tracking System (ALICE subdetector)
JCOP	Joint Controls Project
JSON	Java Script Object Notation
JTAG	Joint Test Action Group
L0	Level-0
L1	Level-1
L1A	L1 accept
L1M	L1 message
L1R	L1 reject
L2	Level-2
L2A	L2 accept
L2R	L2 reject
L2AM	L2 accept message
L2RM	L2 reject message
LDC	Local Data Concentrator
LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty (LHC experiment)
LM	Level-Minus

LTU	Local Trigger Unit CTP building block attributed to a subdetector
LUT	Look-up Table
LVDS	Low-Voltage Differential Signalling
MCM	Multi-Chip Module
MEB	Multi-Event Buffering
MGT	Multi-Gigabit Transceiver
OCDB	Offline Conditions Database
OPC	Object Linking and Embedding for Process Control
ORI	Optical Readout Interface
OSERDES	Output Parallel-to-Serial Converter
PAC	Pause-and-Configure
PASA	Pre-Amplifier and Shaper
PCB	Printed Circuit Board
PCI	Peripheral Component Interconnect
PCS	Physical Coding Sublayer
PHOS	Photon Spectrometer (ALICE subdetector)
PID	Particle identification
PLB	Processor Local Bus
PLD	Programmable Logic Device
PLL	Phase-Locked Loop
PMA	Physical Media Attachment
PMC	PCI Mezzanine Card
PMCD	Phase-Matched Clock Divider
PMD	Photon Multiplicity Detector (ALICE subdetector)
PPC	PowerPC
PRNG	Pseudorandom Number Generator
PROM	Programmable Read-Only Memory
PT	Pretrigger
QGP	Quark-Gluon Plasma

QSFP	Quad SFP
RAM	Random-Access Memory
RCU	Readout Control Unit
RISC	Reduced Instruction Set Computing
ROB	Readout Board
RORC	Readout Receiver Card
RPC	Remote Procedure Call
RST	Reset Broadcast Message (Specialized TTC broadcast message frame)
SCADA	Supervisory Control and Data Acquisition
SDHC	Secure Digital High-Capacity
SDR	Single Data Rate
SDRAM	Synchronous Static Random-Access Memory
SERDES	Serializer/Deserializer
SFP	Small Form-factor Pluggable
SIU	Source Interface Unit (DDL endpoint)
SLC	Scientific Linux CERN
SM	Supermodule (TRD is segmented into 18 supermodules)
SMU	Supermodule Unit (GTU, 2nd stage)
SNMP	Simple Network Management Protocol
SoD	Start-of-Data
SoR	Start-of-Run
SRAM	Static Random-Access Memory
SVF	Serial Vector Format
SYNC	Synchronize
T0	TZERO Detector (ALICE subdetector)
TAP	Test Access Port
TGU	Trigger Unit (GTU, 3rd stage)
TMU	Track Matching Unit (GTU, 1st stage)
TOF	Time-of-Flight Detector (ALICE subdetector)
TPC	Time Projection Chamber (ALICE subdetector)

TRAP	Tracklet Processor
TRD	Transition Radiation Detector (ALICE subdetector)
TTC	Timing, Trigger and Control System to distribute synchronous timing and trigger information
UART	Universal Asynchronous Receiver/Transmitter
V0	VZERO Detector
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VME	Versa Module Europa Bus
ZDC	Zero Degree Calorimeter

Bibliography

[1] Abeysekara, U. et al. ALICE EMCal Physics Performance Report. 2010.

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

[3] ALICE Collaboration. *ALICE Inner Tracking System (ITS): Technical Design Report*. Technical report, CERN, Geneva, 1999.
URL http://cds.cern.ch/record/391175

[4] ALICE Collaboration. *ALICE Technical Design Report of the Photon Spectrometer (PHOS)*. Technical report, Geneva, 1999.
URL https://edms.cern.ch/document/398934/1

[5] ALICE Collaboration. *ALICE Time-Of-Flight system (TOF): Technical Design Report*. Technical report, Geneva, 2000.
URL http://cds.cern.ch/record/430132

[6] ALICE Collaboration. *ALICE Technical Design Report of the Transition Radiation Detector*. Technical report, CERN, Geneva, 2001.

[7] ALICE Collaboration. *ALICE Technical Design Report of the Trigger, Data Acquisition, High-Level Trigger and Control System*. Technical report, CERN, Geneva, 2004.

[8] ALICE Collaboration. The ALICE experiment at the CERN LHC. *JINST*, page S08002, August 2008.
URL http://dx.doi.org/10.1088/1748-0221/3/08/S08002

[9] ALICE Collaboration. The ALICE Transition Radiation Detector: construction, operation, and performance. Publication planned.

[10] ALICE Collaboration, Aamodt, K., et al. Charged-Particle Multiplicity Density at Midrapidity in Central Pb-Pb Collisions at $\sqrt{s_{NN}} = 2.76$ TeV. *Phys. Rev. Lett.*, 105:252301, Dec 2010.
URL http://dx.doi.org/10.1103/PhysRevLett.105.252301

[11] ALICE Collaboration, Carminati, F., Foka, P., Giubellino, P., Morsch, A., Paić, G., Revol, J.-P., Šafařík, K., Schutz, Y. and Wiedemann, U. A. (editors). ALICE: Physics Performance Report, Volume I. *Journal of Physics G: Nuclear and Particle Physics*, 30(11):1517, 2004.
URL http://dx.doi.org/10.1088/0954-3899/30/11/001

[12] ALICE DAQ Project. DATE Reference Manual (v7). ALICE-INT-2010-001, December 2010.

[13] Alme, J. TTC receiver requirement specification. Revision 0.3, 2005.

[14] Alme, J. *Firmware Development and Integration for ALICE TPC and PHOS Front-end Electronics*. Ph.D. thesis, University of Bergen, Bergen, August 2008.

[15] Alme, J., Alt, T., Bratrud, L., Christiansen, P., Costa, F., David, E., Gunji, T., Kiss, T., Langøy, R., Lien, J., et al. RCU2 — The ALICE TPC readout electronics consolidation for Run2. *JINST*, 8(12):C12032, 2013.
URL http://dx.doi.org/10.1088/1748-0221/8/12/C12032

[16] Alme, J., Andres, Y., Appelshäuser, H., Bablok, S., Bialas, N., Bolgen, R., Bonnes, U., Bramm, R., Braun-Munzinger, P., Campagnolo, R., et al. The ALICE TPC, a large 3-dimensional tracking device with fast readout for ultra-high multiplicity events. *Nuclear Instruments and Methods in Physics Research A*, 622:316–367, October 2010.
URL http://dx.doi.org/10.1016/j.nima.2010.04.042

[17] Alt, T. Personal communication.

[18] Altera Corporation. *Understanding Metastability in FPGAs*, 2009.

[19] Andronic, A. and Braun-Munzinger, P. *Ultrarelativistic Nucleus–Nucleus Collisions and the Quark–Gluon Plasma*, pages 35–67. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-44504-3.
URL http://dx.doi.org/10.1007/978-3-540-44504-3_2

[20] Andronic, A. and Wessels, J. P. Transition Radiation Detectors. *Nuclear Instruments and Methods in Physics Research A*, 666:130–147, 2012.
URL http://dx.doi.org/doi:10.1016/j.nima.2011.09.041

[21] Anielski, J. *Entwicklung eines Triggersystems zum Testen und Kablibrieren der Supermodule des ALICE–TRD*. Diplomarbeit, Universität Münster, Institut für Kernphysik, Münster, February 2011.

[22] ATLAS Collaboration. The ATLAS Experiment at the CERN Large Hadron Collider. *JINST*, page S08003, August 2008.
URL http://dx.doi.org/10.1088/1748-0221/3/08/S08003

[23] Barton, J. C. Effect of a data buffer on the recorded distribution of time intervals for random events. *Nucl. Instrum. Methods Phys. Res., A*, 133, 1976.
URL http://dx.doi.org/10.1016/0029-554X(76)90443-2

[24] Bathen, B. *Aufbau einen Triggers für Tests der ALICE-TRD-Supermodule mit kosmischer Strahlung*. Diplomarbeit, Universität Münster, Institut für Kernphysik, Münster, 2007.

[25] Bell, R. E. The Resolver, a Circuit for Reducing the Counting Losses of a Scaler. *Canadian Journal of Physics*, 34(6):563–576, 1956.
URL http://dx.doi.org/10.1139/p56-064

[26] Beringer, J. et al. The Review of Particle Physics. *Phys. Rev.*, D86:010001, 2012 and 2013 partial update for the 2014 edition.
URL http://pdg.lbl.gov

[27] Blanco, F. and Arqueros, F. Dead-time losses in buffer systems. *Nucl. Instrum. Methods Phys. Res., A*, A414, 1998.
URL http://dx.doi.org/10.1016/S0168-9002(98)00686-X

[28] Bocchioli, M., Carena, F., Chapeland, S., Chibante Barroso, V., Lechman, M., Jusko, A., and Pinazza, O. The ALICE Configuration Tool. *J. Phys.: Conf. Ser.*, 331:022034, September 2011.
URL http://dx.doi.org/10.1088/1742-6596/331/2/022034

[29] Borga, A., Costa, F., Crone, G., Engel, H., Eschweiler, D., Francis, D., Green, B., Joos, M., Kebschull, U., Kiss, T., et al. The C-RORC PCIe card and its application in the ALICE and ATLAS experiments. *JINST*, 10(02):C02022, 2015.
URL http://dx.doi.org/10.1088/1748-0221/10/02/C02022

[30] Bridgford, B. and Cammon, J. SVF and XSVF File Formats for Xilinx Devices. XAPP503 (v1.0), April 2002.

[31] Brun, R. and Rademakers, F. ROOT — An object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1–2):81 – 86, 1997.
URL http://dx.doi.org/http://dx.doi.org/10.1016/S0168-9002(97)00048-X

[32] Busch, O. (for the ALICE Collaboration). The detector control system of ALICE TRD. *Nucl. Instrum. Methods Phys. Res., A*, 706(0):86 – 89, 2013.
URL http://dx.doi.org/10.1016/j.nima.2012.05.007

[33] Carena, F., Carena, W., Chapeland, S., Barroso, V. C., Costa, F., Dénes, E., Divià, R., Fuchs, U., Grigore, A., Kiss, T., et al. The ALICE data acquisition system. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 741(0):130 – 162, 2014.
URL http://dx.doi.org/http://dx.doi.org/10.1016/j.nima.2013.12.015

[34] Carena, F., Carena, W., Evans, D., Jovanovic, P., Jusko, A., Lietava, R., Marin, J. C., Schossmaier, K., Vande Vyvre, P., and Villalobos Baillie, O. Start of data and end of data events in ALICE. ALICE-INT-2005-019 v2, September 2005.

[35] CERN. Joint Controls Project. Web page.
URL https://wikis.web.cern.ch/wikis/display/EN/JCOP+Framework

[36] CERN. Timing, Trigger and Control Systems for the LHC. Web page.
URL http://ttc.web.cern.ch/TTC/

[37] Chapman, K. *Get Smart About Reset: Think Local, Not Global.* Xilinx Inc., 2008.

[38] Christiansen, J., Marchioro, A., Moreira, P., and Toifl, T. TTCrx Reference Manual. Version 3.9, 2004.

[39] CMS Collaboration. The CMS experiment at the CERN LHC. *JINST*, page S08004, August 2008.
URL http://dx.doi.org/10.1088/1748-0221/3/08/S08004

[40] Crockford, D. The application/json Media Type for JavaScript Object Notation (JSON), 2006.
URL http://www.ietf.org/rfc/rfc4627.txt

[41] de Cuveland, J. *A Track Reconstructing Low-latency Trigger Processor for High-energy Physics.* Ph.D. thesis, University of Heidelberg, Kirchhoff-Institut of Physics, Heidelberg, September 2009.

[42] de Cuveland, J., Kirsch, S., and Rettig, F. ALICE TRD DAQ Data Format. 2011.

[43] Day, B. *16-Channel, DDR LVDS Interface with Real-Time Window Monitoring.* Xilinx Inc., July 2008.

[44] Divià, R. ALICE Common Data Header specifications. ALICE-INT-2013-XXX V 1.0, July 2014.

[45] Divià, R., Jovanovic, P., and Vande Vyvre, P. Data Format over the ALICE DDL. CERN-ALICE-INT-2002-10 v12.0, October 2011.

[46] Emschermann, D. *Construction and Performance of the ALICE Transition Radiation Detector.* Ph.D. thesis, University of Heidelberg, Heidelberg, 2010.

[47] Engel, H. and Kebschull, U. Common read-out receiver card for ALICE Run2. *JINST*, 8(12), 2013.
URL http://dx.doi.org/10.1088/1748-0221/8/12/C12016

[48] Evans, L. and Bryant, P. e. LHC Machine. *JINST*, page S08001, August 2008.
URL http://dx.doi.org/10.1088/1748-0221/3/08/S08001

[49] Farthouat, P. and Gällnö, P. TTC-VMEbus Interface TTCvi - MkII. Rev 1.6, 2000.

[50] Finisar Corporation. *FTLF8524P2xNy 4.25 Gb/s RoHS Compliant Short-Wavelength SFP Transceiver.* Product specification, March 2005.

[51] Formenti, F., Kluge, A., and Vande Vyvre, P. Trigger error handling and reporting. (ALICE-INT-2006-028 V 1.0), 2006.

[52] Fujiwara, T., Kasami, T., and Lin, S. Error detecting capabilities of the shortened Hamming codes adopted for error detection in IEEE Standard 802.3. *Communications, IEEE Transactions on*, 37(9):986–989, Sep 1989.
URL http://dx.doi.org/10.1109/26.35380

[53] Garrault, P. and Philofsky, B. *HDL Coding Practices to Accelerate Design Performance*. Xilinx Inc., January 2006.

[54] Gavillet, P. Trends in new collider experiment data acquisition systems. *Nucl. Instrum. Methods Phys. Res., A*, 235(CERN-EF-84-012. CERN-EF-84-12):363–379. 40 p, Aug 1984.

[55] George, M. and Alfke, P. Linear Feedback Shift Registers in Virtex Devices. XAPP201 (v1.3), April 2007.

[56] George, M. and Alfke, P. *Virtex-4 FPGA Data Sheet: DC and Switching Characteristics*. Xilinx Inc., September 2009.

[57] Gopal, Vinodh and Guilford, Jim and Ozturk, Erdinc and Wolrich, Gil and Feghali, Wajdi and Dixon, Martin and Karakoyunlu, Deniz. *Fast CRC Computation for Generic Polynomials Using PCLMULQDQ Instruction*. Intel Corporation, December 2009.

[58] Gottschalk, D. DCS Board Documentation Page. Web Site.
URL http://www.kip.uni-heidelberg.de/DCS-Board/

[59] Grosse-Oetringhaus, J.-F., Zampolli, C., Colla, A., Carminati, F., et al. The ALICE Online-Offline Framework for the Extraction of Conditions Data. In *17th International Conference on Computing in High Energy and Nuclear Physics*. 2009.
URL http://dx.doi.org/10.1088/1742-6596/219/2/022010

[60] Gutfleisch, M. *Local Signal Processing of the ALICE Transition Radiation Detector at LHC (CERN)*. Ph.D. thesis, University of Heidelberg, Heidelberg, March 2006.

[61] von Haller, B. et al. The ALICE data quality monitoring system. *J. Phys.: Conf.Ser.*, 331:022030, 2011.
URL http://dx.doi.org/10.1088/1742-6596/331/2/022030

[62] Heath, G. P. Dead Time Due to Trigger Processing in a Data Acquisition System With Multiple Event Buffering. *Nucl. Instrum. Methods Phys. Res., A*, 278:431–435, 1989.
URL http://dx.doi.org/10.1016/0168-9002(89)90861-9

[63] Hellmann, D. *The Python Standard Library by Example*. Developer's Library. Pearson Education, 2011. ISBN 9780132778619.

[64] Hutter, D. *Multi-Event Buffering und HDL-Simulationsframework für die Global Tracking Unit des ALICE-Übergangsstrahlungsdetektors am LHC (CERN)*. Diplomarbeit, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg, 2010.

[65] IEEE Computer Society. IEEE Std 1666™-2011: IEEE Standard for Standard SystemC® Language Reference Manual, January 2012.
URL http://www.accellera.org/downloads/standards/systemc

[66] Institute of Electrical and Electronics Engineers. *IEEE Standard Test Access Port and Boundary-Scan Architecture*. IEEE Std 1149.1-2001, 2001.
URL http://dx.doi.org/10.1109/IEEESTD.2001.92950

[67] Jovanovic, P. Local Trigger Unit - Preliminary Design Review. Revision 1.0, October 2002.
URL http://www.ep.ph.bham.ac.uk/user/krivda/alice/ltu_preliminary10.pdf

[68] Jovanovic, P. Central Trigger Processor - Preliminary Design Review. Revision 0.2, June 2004.
URL http://www.ep.ph.bham.ac.uk/user/krivda/alice/ctp_preliminary_02.pdf

[69] Jovanovic, P. Local Trigger Unit Software Model. Revision 0.1, August 2004.
URL http://alicetrigger.web.cern.ch/alicetrigger/ltu_model01.pdf

[70] Jovanovic, P. Proposal to transmit the L0 signal over the TTC optical link. June 2005.
URL http://www.ep.ph.bham.ac.uk/user/krivda/alice/l0_ttc.pdf

[71] Jovanovic, P. Trigger output logic - hardware guide for the front-end designers. March 2007.
URL http://epweb2.ph.bham.ac.uk/user/krivda/alice/ctp/trigger_inputs.pdf

[72] Jowett, J. M., Schaumann, M., and Versteegen, R. Heavy Ion Operation from Run 2 to HL-LHC. *RLIUP: Review of LHC and Injector Upgrade Plans*.
URL http://dx.doi.org/10.5170/CERN-2014-006.167

[73] JSON-RPC.ORG. *JSON-RPC 1.0 Specification*.
URL http://json-rpc.org/wiki/specification

[74] Jusko, A. An example of DIM server for CTP inputs synchronisation.
URL http://epweb2.ph.bham.ac.uk/user/jusko/ctpinputs/index.html

[75] Jusko, A. L1 and L2 message format, 2013.
URL http://alicetrigger.web.cern.ch/alicetrigger/ls1_upgrade/data_format_run2.pdf

[76] Kirsch, S. *Development of the Supermodule Unit for the ALICE Transition Radiation Detector at the LHC (CERN)*. Diploma thesis, University of Heidelberg, Kirchhoff-Institut of Physics, Heidelberg, September 2007.

[77] Kirsch, S., Rettig, F., Hutter, D., De Cuveland, J., Angelov, V., and Lindenstruth, V. An FPGA-based High-Speed, Low-Latency Processing System for High-Energy Physics. In *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, pages 562–567. Aug 2010. ISSN 1946-1488.
URL http://dx.doi.org/10.1109/FPL.2010.110

[78] Klein, J. personal communication.

[79] Klein, J. *Commissioning of and Preparations for Physics with the Transition Radiation Detector in A Large Ion Collider Experiment at CERN*. Diplomarbeit, Universität Heidelberg, Physikalisches Institut, Heidelberg, 2008.

[80] Klein, J. Triggering with the ALICE TRD. *Nucl. Instrum. Meth.*, A706:23–28, 2013.
URL `http://dx.doi.org/10.1016/j.nima.2012.05.011`

[81] Kramer, F., Blume, C., Dietel, T., and Oyama, K. A DCS-Offline Communication Framework for the ALICE TRD. *GSI Scientific Report*, 2008.

[82] Krawutschke, T. *Reliability and Redundancy of an Embedded System used in the Detector Control System of the ALICE Experiment*. Ph.D. thesis, University of Mannheim, Mannheim, 2009.

[83] Krivda, M. ALICE CTP upgrade, September 2013. Topical Workshop on Electronics for Particle Physics 2013.

[84] Krivda, M., Barnby, L., Bombara, M., Evans, D., Jones, P. G., Jusko, A., Králik, I., Lazzeroni, C., Lietava, R., Matthews, Z. L., et al. The ALICE trigger system performance for p-p and Pb-Pb collisions. *JINST*, 7(01):C01057, 2012.
URL `http://dx.doi.org/10.1088/1748-0221/7/01/C01057`

[85] Kuramoto, R. Xilinx In-System Configuration Using an Embedded Microcontroller. XAPP058 (v4.1), March 2009.

[86] Lattice Semiconductor. DDR Interface Design Implementation. December 2004.

[87] Lehnert, J., Dönigus, B., Klein, J., Krawutschke, T., Oyama, K., Schicker, R., Schmiederer, S., Schweda, K., and Stachel, J. Status of the Pretrigger System for the ALICE Transition Radiation Detector, 2009. GSI Scientific Report.

[88] LHCb Collaboration. The LHCb detector at the LHC. *JINST*, page S08005, August 2008.
URL `http://dx.doi.org/10.1088/1748-0221/3/08/S08005`

[89] Lindenstruth, V. and Kisel, I. Overview of trigger systems. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 535(1–2):48 – 56, 2004. ISSN 0168-9002. Proceedings of the 10th International Vienna Conference on Instrumentation.
URL `http://dx.doi.org/http://dx.doi.org/10.1016/j.nima.2004.07.267`

[90] Lyubka, S. Mongoose - easy to use web server. Web page.
URL `http://code.google.com/p/mongoose/`

[91] Mager, M. *Studies on the upgrade of the ALICE central tracker*. Ph.D. thesis, Technical University of Darmstadt, Darmstadt, 2012.

[92] Martinez, G. Advances in Quark Gluon Plasma. 2013.

[93] Mercado Pérez, J. *Development of the control system of the ALICE Transition Radiation Detector and of a test environment for quality-assurance of its front-end electronics*. Ph.D. thesis, Universität Heidelberg, Physikalisches Institut, Heidelberg, September 2008.

[94] Oyama, K., Adler, C., Angelov, V., Emschermann, D., and Lippmann, C. ALICE TRD Raw Data Format Specification. 2007.

[95] Pettersson, T. S. and Lefèvre, P. *The Large Hadron Collider: conceptual design*. Technical Report CERN-AC-95-05-LHC, CERN, Geneva, Oct 1995.
URL http://cds.cern.ch/record/291782

[96] Pozniak, K. T. FPGA-based, specialized trigger and data acquisition systems for high-energy physics experiments. *Measurement Science and Technology*, 21(6):062002, 2010.
URL http://stacks.iop.org/0957-0233/21/i=6/a=062002

[97] ur Rehman, A. *The ALICE TPC Readout Electronics - Design, performance optimization and verification of the DAQ circuit*. Ph.D. thesis, University of Bergen, Department of Physics and Technology, Bergen, October 2012.

[98] Rettig, F. *Entwicklung der optischen Auslesekette für den ALICE-Übergangsstrahlungsdetektor am LHC (CERN)*. Diplomarbeit, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg, 2007.

[99] Rettig, F. *Doctoral thesis*. Ph.D. thesis, University of Frankfurt, Frankfurt Institute for Advanced Science, Frankfurt, Publication planned.

[100] Rossegger, S., Schnizer, B., Riegler, W., and Betev, L. *Simulation and Calibration of the ALICE TPC including innovative Space Charge Calculations*. Ph.D. thesis, Graz, Tech. U., Graz, 2009. Presented on 30 Oct 2009.
URL http://cds.cern.ch/record/1217595

[101] Rubin, G. and Soós, C. ALICE DETECTOR DATA LINK: Hardware Guide for the Front-end Designers. v2.4, September 2007.

[102] Rubin, G., Vande Vyvre, P., and Soós, C. ALICE DETECTOR DATA LINK: Interface Control Document. ALICE-INT-2004-018 v1.0, July 2004.

[103] Schmiederer, S. *Development and Implementation of the Control System for the Pre-Trigger System of the Transition Radiation Detector at ALICE*. Diplomarbeit, Universität Heidelberg, Physikalisches Institut, Heidelberg, October 2009.

[104] Schneider, R. *Entwicklung des Triggerkonzepts und die entsprechende Implementierung eines 200-GBs-Auslesenetzwerks für den ALICE-Übergangsstrahlungsdetektor*. Ph.D. thesis, University of Heidelberg, Heidelberg, May 2008.

[105] SFF Committee. *Specification for Diagnostic Monitoring Interface for Optical Transceivers*. SFF-8472.
URL ftp://ftp.seagate.com/sff

[106] Siemens. Simatic WinCC. Web page.
URL http://w3.siemens.com/mcms/automation/en/Pages/automation-technology.aspx

[107] Srikanth, E. *How do I reset my FPGA?* Xilinx Inc., 2011.

[108] Taylor, B. G. TTC laser transmitter (TTCex, TTCtx, TTCmx) User Manual. Rev 2.0.

[109] Velure, A. Upgrades of the ALICE TPC Front-End Electronics for Long Shutdown 1 and 2. *IEEE Transactions on Nuclear Science*, PP(99):1–1, 2015.
URL http://dx.doi.org/10.1109/TNS.2014.2382332

[110] Weber, S. G. and Andronic, A. ALICE event display of a Pb-Pb collision at 2.76A TeV, Jul 2015. General Photo.
URL http://cds.cern.ch/record/2032743

[111] Xilinx Inc. Virtex-4 XC4VFX20CES4, XC4VFX60CES4, XC4VFX100CES4, and XC4VFX140CES4 Errata. EN042 (v1.2), October 2006.

[112] Xilinx Inc. Answer Record 14425. August 2007.
URL http://www.xilinx.com/support/answers/14425.html

[113] Xilinx Inc. Answer Record 21435. November 2007.
URL http://www.xilinx.com/support/answers/21435.htm

[114] Xilinx Inc. Inactive Transceiver Behavior Work-Arounds for Virtex-4 FX RocketIO MGT. XAPP732 (v1.1), September 2007.

[115] Xilinx Inc. Virtex-4 RocketIO Multi-Gigabit Transceiver. UG076, Version 4.1, November 2008.

[116] Xilinx Inc. Virtex-4 User Guide. UG070, Version 2.6, December 2008.

[117] Xilinx Inc. Virtex-4 XC4VFX20, XC4VFX40, XC4VFX60, XC4VFX100, and XC4VFX140 Production Errata. EN070 (v1.8), May 2008.

[118] Xilinx Inc. Virtex-4 Family Overview. DS112, Version 3.1, August 2010.

[119] Yeoh, T. Y. DDR2 SDRAM Physical Layer Using Direct-Clocking Technique. XAPP701 (v2.0), March 2007.

[120] Zhao, C., Alme, J., Alt, T., Appelshäuser, H., Bratrud, L., Castro, A., Costa, F., David, E., Gunji, T., Kirsch, S., et al. First performance results of the ALICE TPC Readout Control Unit 2. *JINST*, 2016. Topical Workshop on Electronics for Particle Physics 2015.
URL http://dx.doi.org/10.1088/1748-0221/11/01/C01024

[121] Zimmer, S. *Design, Implementation and Commissioning of the Pretrigger System for the Transition Radiation Detector at the ALICE experiment of CERN.* Diplomarbeit, Universität Heidelberg, Physikalisches Institut, Heidelberg, October 2009.

[122] Øverås, H. Dead-time losses in a buffered data recording system. *Nucl. Instrum. Methods Phys. Res., A*, 104, 1972.
URL http://dx.doi.org/10.1016/0029-554X(72)90300-X

[123] Šafařík, K. (for the ALICE Collaboration). Overview of Recent ALICE Results. *Nuclear Physics A*, 904–905(0):27c–34c, 2013.
URL http://dx.doi.org/10.1016/j.nuclphysa.2013.01.041

Zusammenfassung

Zwischenspeicherung von Ereignisdaten zur Minimierung auslesebedingter Totzeit und lokale Ereigniszusammenstellung für den ALICE-Übergangsstrahlungsdetektor

Der Übergangsstrahlungsdetektor (TRD) des ALICE-Experiments am Large Hadron Collider (LHC) des CERN ist als Trigger- und Auslesedetektor konzipiert [6]. Basierend auf einer Spurrekonstruktion von Teilchen mit hohem Transversalimpuls und Betrachtungen bezüglich Impuls, Anzahl und räumlicher Verteilung liefert er Beiträge zur L1-Triggerentscheidung des Experiments [99]. Weiterhin liefert der TRD, sofern ein Ereignis zur Auslese selektiert wurde, die kompletten Ereignisrohdaten zur späteren Offline-Analyse. Das zentrale Element in der Trigger- und Auslesekette des TRD ist hierbei die Globale Spurfindungseinheit (GTU), die den wesentlichen Teil der Triggeraufgaben sowie die Auslese der Ereignisrohdaten übernimmt. Die GTU ist als Trigger- und Ausleseprozessor mit 109 FPGA-basierten Verarbeitungseinheiten in drei Hierarchiestufen konzipiert [41].

Die hohen Eingangsdatenraten im Bereich mehrerer Tbit/s und die zweiteilige Funktion als Auslese- und Triggersystem, an das strenge Anforderungen bezüglich Latenz gestellt sind, machen die GTU zu einem einzigartigen Hochleistungssystem zur Datenverarbeitung. Betrachtet man die hohe Komplexität eines Detektorverbands wie ALICE und die hohen Kosten, die der Betrieb von Beschleuniger und Experiment verursachen, ist es offensichtlich, dass ein solches System auf maximale Effizienz bei der Datennahme optimiert sein muss. Wesentliche Punkte sind hierbei eine hohe Stabilität im Betrieb sowie eine niedrigstmögliche Totzeit.

Die vorliegende Arbeit stellt die Lösungen vor, die zur Verarbeitung von zur Auslese bestimmten Daten in der GTU erarbeitet wurden, vom initialen Empfang hin zur lokalen Ereigniszusammenstellung und Übertragung an die Rechnerfarm des High-Level Trigger. Die in diesem Rahmen konzipierten Verarbeitungseinheiten nutzen unter anderem das Zwischenspeichern mehrerer Ereignisse, eine Inspektion und eventuelle Aufbereitung einlaufender Datenströme in Echtzeit, umfangreiche Diagnostik, sowie charakteristische Funktionen moderner FPGAs, um ein robustes Hochleistungs-Auslesesystem zu schaffen, das die Auslese des Detektors bei hoher Bandbreite mit der erforderlichen hohen Stabilität und gleichzeitig minimierter Totzeit bewältigen kann. Die vorliegende Arbeit umfasst hierbei nicht nur die konzeptionelle Auslegung der Zwischenspeicherung und Auslese von Daten sowie einer geeigneten Segment-Kontrolleinheit. Bestandteil sind ebenso deren Implementierung, Simulation, und Verifikation, deren Integration in das Gesamtsystem einhergehend mit den in [99] vorgestellten Arbeiten bezüglich der GTU-Trigger, sowie deren Inbetriebnahme im Experiment. Die für eine komplette Fernwartung und -steuerung entwickelte Infrastruktur und die Verifikations- und Testverfahren, die es erlauben, Upgrades des Systems auf kurzen Zeitskalen sicher durchzuführen, werden ebenfalls vorgestellt. Abschließend wird das entwickelte Auslesesystem vermessen, dessen Leistung und Totzeit im alleinigen Betrieb, aber auch im Zusammenspiel mit dem Haupt-Spurverfolgungsdetektor des Experiments, evaluiert und eine kurze Zusammenfassung des Betriebs in der ersten, bereits abgeschlossenen Datennahmekampagne am LHC gegeben.

Die FPGAs der GTU sind das Bindeglied zwischen speziell gefertigten, angepassten integrierten Schaltkreisen auf dem Detektor selbst und den zur Weiterverarbeitung der Daten verwendeten Rechnerfarmen. Eine mit jeder FPGA-Generation steigende Anzahl an Ressourcen erlaubt es, zahlreiche Übertragungsprotokolle zu implementieren und die Firmware der FPGAs fortwährend an aktuelle Gegebenheiten des Experimentes anzupassen, wodurch FPGAs in den Auslesesystemen aller großer LHC-Experimente der aktuellen und kommenden Detektorgeneration weite Verbreitung finden. Zusätzlich ermöglichen optimierte, datenflussorienterte Algorithmen in FPGAs die Verarbeitung von Daten bei großen Datenraten quasi in Echtzeit. Im komplexen Trigger- und Datenpfad der GTU finden sich, unter anderem mit dem Spurfindungsalgorithmus, dem Empfang und der Inspektion eingehender Ereignisdaten in Echtzeit, sowie der Verarbeitung eingehender Triggersequenzen, hervorragende Beispiele für diese Aspekte.

Vorab bietet Abbildung 1 eine Übersicht über den grundlegenden Aufbau und eine grobe Verteilung der Aufgaben in der GTU. Die erste Stufe bilden insgesamt 90 TMU-Einheiten,[1] welche die von den Detektorkammern gesendeten Daten bei einer aggregierten Rate von bis zu 2,16 Tbit/s annehmen und zwischenspeichern. Je 5 TMUs, jede davon einem Detektor-Stapel mit 12 optischen Links zugeordnet, bilden zusammen mit einer SMU, einer Verarbeitungseinheit der zweiten Stufe, ein GTU-Segment. Die SMU übernimmt hierbei die Rolle einer Segmentsteuerung, die trigger- und auslesebezogene Aktivitäten in den TMUs koordiniert und auf Supermodul-Ebene selbst ausführt. In jedem der 18 Segmente findet parallel die komplette Verarbeitung der Daten des zugeordneten TRD-Supermoduls statt. Die von den Segmenten berechneten Trigger-Informationen sowie das Totzeitsignal werden auf der TGU zu globalen Signalen zusammengeführt und an den zentralen Triggerprozessor des Experiments übermittelt.

Beim Betrieb des ALICE-Experimentes zur Datennahme werden alle Detektoren vom zentralen Triggerprozessor durch mehrstufige Trigger synchronisiert und gesteuert. Dementsprechend benötigen alle 18 GTU-Segmente eine geeignete Steuerungseinheit, die darauf basierend alle trigger- und auslesebezogenen Aktivitäten im Segment koordiniert. Durch eine Analyse der im Gesamtsystem vorhanden Totzeitquellen und der Struktur der ankommenden Triggersequenzen war es möglich, eine robuste, zweistufige Kontrolleinheit zu konzipieren, die die erforderlichen Steuersignale und Referenzpunkte bereitstellt. Multiple

[1] Da sich die Auslegung der GTU, insbesondere ihrer Segmente, an voll ausgestatteten TRD Supermodulen mit je 5 Detektor-Stapeln orientieren muss, wird den hier angegebenen Zahlen ein Vollausbau des TRD mit 18 kompletten Supermodulen zugrunde gelegt. Die drei speziell entfernten, zentralen Stapel der Supermodule 13, 14 und 15 werden nicht berücksichtigt.

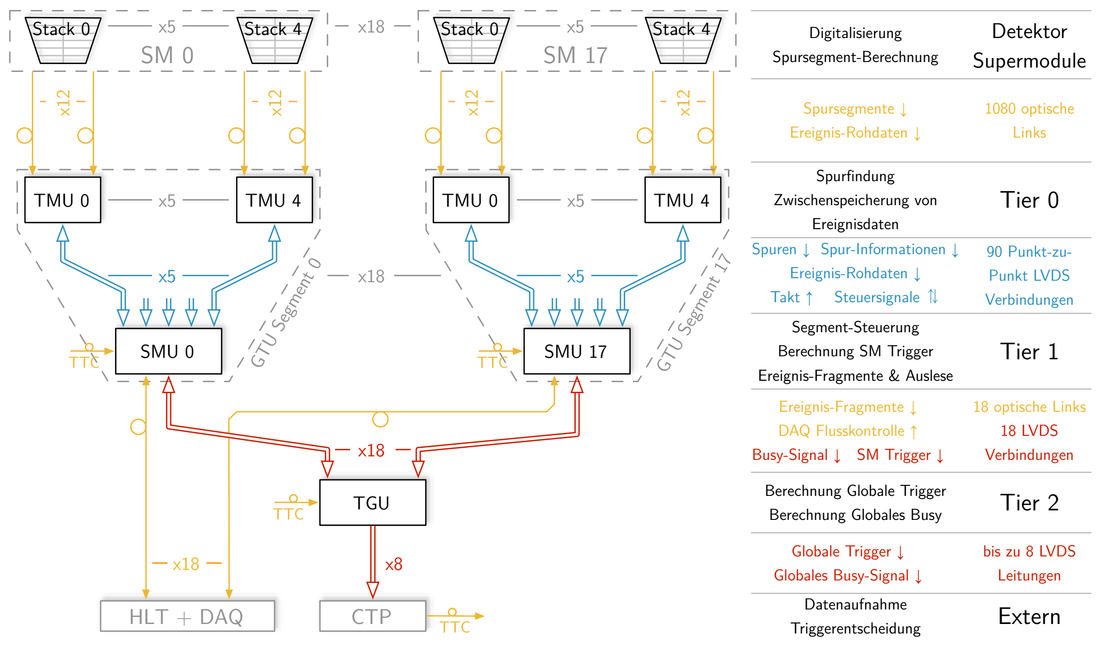


Abbildung 1: Aufbau der GTU mit drei Hierarchiestufen, sowie die Verbindungen zum Detektor und externen Systemen.

Ereignisspeicher dienen bei einem mehrstufigen Triggersystem dazu, kurzzeitige Fluktuationen der eingehenden Ereignisrate abzufedern und damit durch die Auslese bedingte, negative Totzeit-Effekte abzuschwächen. Die grundlegenden Prinzipien, insbesondere der Einfluss der Anzahl vorhandener Speicherplätze, werden in der Arbeit zusammengefasst. Die Kontrolleinheit und die vorgeschaltete Einheit zum Triggerempfang wurden entsprechend so gestaltet, dass sie die Verarbeitung mehrerer verschachtelter Triggersequenzen unterstützen, da dies wiederum Grundvoraussetzung für gleichzeitige Speicherung und Verwaltung von Daten mehrerer Kollisionsereignisse ist. Durch das Dekodieren der Triggersequenzen direkt in der GTU und entsprechende Erweiterungen der Kontrolleinheit war es weiterhin möglich, illegale Sequenzen zu erkennen und somit die unbemerkte Aufzeichnung potenziell fehlerhaft zugeordneter Daten zu verhindern. Die in das System integrierte, umfangreiche Diagnostik erlaubt in einem solch gravierenden Fehlerfall eine genaue Analyse der Ursachen des Problems, um diese zukünftig zu beheben und trägt somit zu einer signifikanten Steigerung der Stabilität des Gesamtsystems bei.

Die Übertragung von Detektordaten zur GTU erfolgt unidirektional ohne Möglichkeit zur Flusskontrolle zu bestimmten Zeiten innerhalb einer Triggersequenz. Die Eingangseinheiten und Ereignisspeicher, die durch eine Unterteilung der TMU-SRAMs in mehrere Bereiche realisiert sind, wurden dementsprechend so ausgelegt, dass sie die Daten eines Ereignisses bei der vollen aggregierten maximalen Sendebandbreite von etwa 2,16 Tbit/s annehmen und zwischenspeichern können. Trotz der hohen Bandbreiten beschränkt sich die GTU aber nicht auf ein simples Aggregieren und Weiterleiten der Daten. Da bei in einem Umfeld mit erhöhter Strahlung agierender Elektronik mit gelegentlichen Fehlern gerechnet werden muss, durchlaufen eingehende Detektordaten spezielle Link-Former-Einheiten, die sicherstellen, dass alle Daten einem vordefinierten Format entsprechen. Im Falle einer Verletzung des Formats können sie Fehler markieren und die Daten in einen Zustand bringen, der problemlose Weiterverarbeitung erlaubt. Diese Echtzeit-Inspektion der Daten trägt direkt zur einer Erhöhung der Stabilität des Betriebs bei, da nun Ausfälle einzelner Kammern nicht zu einem Stillstand der Auslese führen. Zusammen mit der Garantie, dass von der Kontrolleinheit ausgehende Signale verifiziert und gültig sind, reduziert dies maßgeblich die Komplexität aller nachfolgenden Einheiten des Datenpfades und erlaubt einfache Schnittstellen zur Segmentsteuerung. Durch die Bereitstellung geeigneter Pufferstrukturen können die eigentlichen Ereignisrohdaten um wichtige, in der GTU berechnete Informationen erweitert werden, die dann wiederum direkt online weiteren Verarbeitungsstufen zur Verfügung stehen. Beispiele sind hier die Status-Informationen der Link-Former-Einheiten oder von berechneten CRCs, die direkt einen schnellen Überblick über das Verhalten der Detektorkammern und den Zustand des Datentransports erlauben, oder Zwischenergebnisse aus der Triggerberechnung, die als Grundlage einer HLT-Triggerkomponente sowie offline zur Verifikation des Triggers dienen.

Der Datenpfad der GTU teilt sich bestimmte Ressourcen mit der extrem zeitkritischen Triggerberechnung. Um die Latenz des Triggers zu minimieren, werden folglich triggerbezogene Daten mit höchster Priorität behandelt. Weiterhin konnte ein paralleles, source-synchrones Übertragungsschema für die Busplatine zwischen TMU und SMU realisiert werden, das eine hohe Bandbreite bei niedriger Latenz von nur wenigen Taktzyklen zur Verfügung stellt. Die entsprechende Schnittstelle nimmt außerdem eine automatisierte, individuelle Kalibrierung des Abtast-Zeitpunktes jeder Übertragungsleitung vor, um die bestmögliche Datenübertragung für alle Einzelverbindungen zwischen TMUs und SMUs im System sicherzustellen. Das Resultat zeigt sich in der Bitfehlerrate für den Datenpfad eines kompletten Segments, die bei Langzeit-Versuchen auf weniger als 10^{-15} bestimmt werden konnte.

Die übliche Nutzung der installierten, voll ausgebauten GTU als Produktivsystem und die nur sehr kurzen Zeitfenster, die sich während des regulären Beschleunigerbetriebes für Updates der Firmware bieten, lassen umfangreiche Tests mit diesem System im Allgemeinen nicht zu. Um die parallel zum Produktivbetrieb laufende Weiterentwicklung der GTU, deren Anpassung an aktuelle Anforderungen des Experiments und eine schnelle Verifikation im Produktivsystem zu ermöglichen, sind ausgereifte Simulations- und Verifikationsverfahren unabdingbar. Die erste Stufe bildet hierbei eine umfangreiche Simulation des Gesamtsystems, in welcher die drei Typen von GTU-FPGA-Designs auf Top-Level-Ebene unverändert mit Modellen aller relevanten, externen Komponenten verbunden sind. Um möglichst realistisch und aussagekräftig simulieren zu können, wurde eine spezielle Triggeremulation entwickelt. Deren Stimuli sind in Bezug auf Kodierung, Struktur, rückgekoppeltem Zeitverhalten und Inhalt realen Triggersequenzen in ALICE nachempfunden und können kontrolliert manipuliert werden, um auch das Verhalten der GTU im Fehlerfall zu testen. Da sie in Hinblick auf Anzahl der Knoten, Datenquellen und das Verhalten der Datensenken vielseitig konfigurierbar ist, erlaubt die Systemsimulation eine große Zahl von Szenarien unter realistischen Bedingungen mit vergleichsweise geringem Zeitaufwand zu testen. Um den Datentransport auch bei hohen Raten und mit hoher Statistik automatisiert zu verifizieren, sind pseudo-zufällige Ereignisgeneratoren in die Firmware integriert. Diese können im Betrieb dynamisch an vorderster Stelle in den TMU-Eingangspfad geschaltet werden, und erlauben in Kombination mit eigens entwickelter Verifikationssoftware auf dem Zielrechner eine bitgenaue Verifikation des kompletten Datentransports durch die GTU-Segmente bis hin zum Datenaufnahmesystem. Die korrekte Funktion zur Verwendung im Produktivbetrieb bestimmter Firmwares lässt sich somit zuerst im Testsystem, dann auf großer Skala im Produktivsystem, in kurzer Zeit mit hoher Statistik verifizieren. Das hier entwickelte, mehrstufige Simulations- und Verifikationsverfahren ermöglicht schnelle Design-Zyklen und stellt gleichzeitig sicher, dass nur hinreichend getestete FPGA-Designs im Produktivbetrieb eingesetzt werden.

Für den regulären automatisierten, ferngesteuerten Betrieb ist eine Integration der GTU in das Kontrollsystem des Experiments notwendig. Hierbei müssen mehr als hundert heterogene Endpunkte von einem zentralen System aus initialisiert, konfiguriert und sicher betrieben werden. Die vorliegende, mehrlagige Architektur des GTU-DCS mit zentralem Hintergrunddienst, segment-orientierten Kontrollprogrammen auf den DCS-Boards, und damit vollem Zugriff auf die elementaren Konfigurationsnetze aller 109 FPGA-Knoten, erfüllt diese Anforderungen. Der als Zustandsautomat implementierte Hintergrunddienst integriert die GTU in das übergeordnete, vom Experiment verwendete Steuerungssystem. Er erlaubt eine flexible, run-typ-abhängige Konfiguration und den Betrieb der GTU über eine vereinfachte Schnittstelle. Um die Initialisierung und Konfiguration der GTU zu beschleunigen, wurde das Programmieren der Devices via JTAG stark optimiert und gängige, aufwendige Kalibrierungs- und Konfigurationsabläufe in dedizierte Sequenzen gekapselt, die direkt auf den eingebetteten CPUs der FPGA-Knoten schnell ausgeführt werden. Zur genauen Analyse des internen Zustands der GTU bei eventuell auftretenden Problemen sind umfangreiche Statusinformationen und Diagnostikfunktionen, die insbesondere alle externen Schnittstellen beinhalten, durch das Konsolenprogramm zugänglich und automatisiert extrahierbar. Dies trug erheblich zur Fehlerfindung in der kompletten Auslesekette und damit maßgeblich zu einer Erhöhung der Stabilität und einer Optimierung der Totzeit bei.

Die effektive Auslesebandbreite hat, da sie die Bearbeitungszeit der GTU-Auslesestufe bestimmt, direkten Einfluss auf die auslesebedingte Totzeit. Durch sorgfältige Optimierung des Datenpfades und der Schnittstelle zum Auslesesystem konnte in Run1, trotz gleichzeitiger Aktivität von Trigger und Auslese, eine aggregierte effektive Auslesebandbreite von 3,33 GiB/s erreicht werden. Dies bedeutet, dass 99,30 % der verfügbaren Taktzyklen bei der Übertragung an das Datenauslesesystem nutzbar sind. Obwohl initial nicht vorgesehen, konnte die Auslesebandbreite für Run2 ohne signifikante Hardware-Änderungen, aber durch Verlagerung von Funktionalität in die GTU, Benutzung GTU-eigener Übertrager bei höheren Raten und weiterer Optimierung der Speicherschnittstelle und des Datenpfades auf insgesamt 6,50 GiB/s gesteigert werden, was einer möglichen Linkauslastung von 99,97 % entspricht.

Mit Kenntnis der Bearbeitungszeit der verschiedenen Verarbeitungsstufen lassen sich Aussagen über das Totzeitverhalten des TRD im alleinigen, als auch im Betrieb mit anderen Detektoren in einer gemeinsamen Auslesepartition untersuchen. Hierzu dient eine ereignisbasierte Monte-Carlo-Simulation, die das Detektorverhalten modelliert und deren Korrektheit für den TRD verifiziert wurde. Aus der Simulation ergibt sich, dass die Auslesebandbreite in Run1 bei einem typischen Pb-Pb-Run mit einer Mischung aus zentralen und einigen semi-zentralen Ereignissen einer maximalen Ausleserate von etwa 425 Hz entspricht. Durch Einsatz der multiplen Ereignisspeicher und der hohen möglichen Linkauslastung zum Datenaufnahmesystem kann die auslesebedingte Totzeit bei den vorgesehenen Raten für Run1 von etwa 200 Hz auf unter etwa 5 % reduziert werden. Die erwartete Steigerung von Ereignisgrößen aufgrund höherer Multiplizität in Run2 mit eingerechnet, entspricht die effektive Auslesebandbreite in Run2 im vorherigen Szenario einer maximalen Ausleserate von etwa 670 Hz. Im Verbund mit anderen Detektoren ergibt sich bei Betrieb im Bereich der vorgesehenen Rate von 450–500 Hz ein auslesebedingter Beitrag zur Totzeit von unter 10 %, der außerdem weitgehend von Beiträgen anderer Detektoren überlagert wird. Dieser niedrige Totzeitbeitrag durch die Auslese lässt einen negativen Einfluss auf die Totzeit des Gesamtdetektors von Seiten der TRD-Auslese nicht erwarten und qualifiziert das System somit zur effizienten Datennahme in Run2. Möglichkeiten, diesen auslesebedingten Beitrag im vorgestellten System weiter zu minimieren, wurden identifiziert und quantifiziert. Sie könnten, sofern dies eine unerwartete Leistungssteigerung des Gesamtdetektors nötig macht, umgesetzt werden.

Die hier vorgestellten Lösungen sind in den FPGA-Designs der GTU implementiert. Die GTU befindet sich seit Beginn der Datennahme mit dem LHC im kontinuierlichen Produktivbetrieb und zeichnet sich dabei als robuste, zuverlässige und performante Trigger- und Ausleseeinheit des TRD aus. Die Laufzeiten aller Inbetriebnahme- und Produktionsruns zusammengerechnet, kam die GTU in der ersten, langen LHC-Kampagne auf 572 Tage effektiver Datennahme. Dabei wurden etwa $2{,}3 \cdot 10^9$ Ereignisse mit einem Gesamtdatenvolumen von circa 0,55 PB aufgezeichnet. Mit dem Beginn der zweiten LHC-Kampagne 2015 war die Installation des TRD mit 18 Supermodulen abgeschlossen. Die GTU führt seitdem mit erhöhter Auslesebandbreite die Zwischenspeicherung von Ereignisdaten, die lokale Ereigniszusammenstellung und die Auslese für den TRD durch. Durch die erreichte sehr hohe Zuverlässigkeit und hohe Ausleseleistung hat sie signifikanten Anteil an einer effizienten Datennahme mit dem TRD und den daraus resultierenden Kollisionsdaten für die Physikanalyse.