FACULTY OF
PHYSICS AND ASTRONOMY

UNIVERSITY OF HEIDELBERG

DIPLOMA THESIS
IN PHYSICS

SUBMITTED BY
STEFAN KIRSCH
BORN IN
LANDAU, GERMANY

SEPTEMBER 2007

Development of the Supermodule Unit for the ALICE Transition Radiation Detector at the LHC (CERN)

A fault-tolerant, trigger-driven control and readout unit to interface the *TRD* Global Tracking Unit event buffers with the Data Acquisition System

This diploma thesis has been carried out by **Stefan Kirsch** at the

Kirchhoff Institute of Physics

under the supervision of

Prof. Dr. Volker Lindenstruth

Development of the Supermodule Unit for the *ALICE TRD*

Retrieving interesting events from the vast amounts of data produced by the detectors of the *ALICE* experiment at *LHC* during heavy-ion collisions requires the use of a sophisticated, hierarchical trigger system. Designed as fast trigger detector, the Transition Radiation Detector (*TRD*) provides parameterized particle track segments and raw data at rates as high as 2.16 Tb/s. At the Global Tracking Unit (*GTU*), buffers capable of operating with multiple interleaved trigger sequences are used to store arriving event data at full bandwidth.

Like the *TRD*, the *GTU* is divided in 18 segments. As concentrator unit in the time-critical trigger path of a *GTU* segment, the Supermodule Unit administers the buffering of multiple events, according to the associated trigger sequences issued by the *CTP*. For transmission to the *DAQ*, event identification and status information is tagged to the event fragments that are built from the contents of all event buffers.To guarantee integrity of data and identification, the system features complex supervision and error detection for fault-tolerant operation even in the rare event of trigger errors. A suitable FPGA design for the Supermodule Unit and its integration into the *GTU* project is presented in this thesis.

Entwicklung der Supermodule Unit für den *ALICE TRD*

Die Auswahl interessanter physikalischer Ereignisse aus der großen Menge an Daten, die beim Schwerionen–Experiment *ALICE* am *LHC* anfallen, erfordert den Einsatz eines komplexen, hierarchisch gestuften Triggersystems. Ausgelegt als schneller Triggerdetektor produziert der Transition Radiation Detector (*TRD*) sowohl parametrisierte Spursegmente für eine Triggerentscheidung, als auch Rohdaten mit Raten bis zu 2.16 Tb/s. Einlaufende Rohdaten werden in den Speichern der Global Tracking Unit (*GTU*) bei voller Bandbreite zwischengepuffert, wobei die Speicher auf den Betrieb mit verschachtelten Triggersequenzen ausgelegt sind.

Der Geometrie des *TRD* folgend, ist die *GTU* in 18 Segmente unterteilt. Die Supermodule Unit administriert als Konzentrator-Board eines *GTU* Segments das Zwischenspeichern der Rohdaten in Abhängigkeit der ihnen zugeordneten Triggersequenzen. Da sich die *SMU* im zeitkritischen Pfad der Triggerinformationen befindet, werden alle Operationen in Hardware ausgeführt. Zur Übertragung an das Datenaufnahme-System formt die *SMU* Datenpakete aus den Inhalten aller Zwischenspeicher und versieht sie mit Informationen zum Status und zur Identifikation des Events. Um die Synchronisation zwischen Daten und Event-Identifikation sicherzustellen und einen fehlerfreien Betrieb zu gewährleisten, werden die Triggersequenzen überwacht und Fehler abgefangen. Die vorliegende Arbeit präsentiert das FPGA-Hardware-Design der *SMU* und dessen Integration in das Gesamtprojekt.

Contents

List of Figures

1 Introduction

It is assumed that shortly after the big bang the universe was composed of a partonic medium, the Quark-gluon plasma (*QGP*) with 'normal' matter being the result of a phase transition from *QGP* into the hadronic phase. In order to investigate this process, it is inversed in ultra-relativistic heavy-ion collisions. By colliding heavy nuclei at nearly the speed of light, physicists create energy and particle densities as they might have existed in the early phase of the universe, expecting to find evidence for the existence of *QGP*.

In the fascinating search for fundamental particles and the interactions amongst them, physics has arrived at the formulation of the Standard Model, which is constituted primarily of gauge theories of the electroweak and strong interaction. Developed, approved and redefined through many high energy physics experiments, it states the existence of two classes of fermions, *leptons* and *quarks*, as fundamental particles. Forces amongst these particles can be attributed to three fundamental interactions[1], mediated by the gauge bosons as force carriers.

Generation	Leptons	q/e	m	Quarks	q/e	m
first	e	-1	511 keV	u	$2/3$	$\approx 2\,\text{MeV}$
	ν_e	0	$\leq 225\,\text{eV}$	d	$-1/3$	$\approx 5\,\text{MeV}$
second	μ	-1	106 MeV	c	$2/3$	$\approx 95\,\text{MeV}$
	ν_μ	0	$\leq 0.17\,\text{MeV}$	s	$-1/3$	$\approx 1.2\,\text{GeV}$
third	τ	-1	1.78 GeV	t	$2/3$	$\approx 4.2\,\text{GeV}$
	ν_τ	0	$\leq 18.2\,\text{MeV}$	b	$-1/3$	$\approx 174\,\text{GeV}$

Table 1.1: All existing matter is made of 12 fundamental particles. Anti-particles and their corresponding properties are not shown in the table. The mass is given in energy equivalent, while the charge is given as a multiple of the elementary charge. Source: [Y$^+$06]

As shown in table 1.2, all forces, except gravity, couple to a charge. While the weak and the electromagnetic force couple to the weak and the electromagnetic charge, respectively, the coupling of the strong force is explained by introducing a color charge. Contrary to the electromagnetic and the weak force, the strength of the interaction between

[1]The fourth fundamental force, gravitation is not integrated in the Standard Model. It acts through its force carrier, the graviton, on all particles. A theory quantizing quantum gravitation does not yet exist and the graviton is yet undetected.

Force	Range	Strength (rel.)	Couples to	Gauge Boson
Strong	10^{-15} m	1	color charge	Gluon (g)
Weak	10^{-17} m	10^{-2}	weak charge	$W^{\pm}$ Z^0
Electromagnetic	∞	10^{-14}	em charge	Photon (γ)
Gravitation	∞	10^{-38}	mass	undetected

Table 1.2: The four fundamental forces, which can explain all interactions between particles. The forces have effects on very different scales. While weak and strong force are observable inside nuclei only, electromagnetic force and gravitation have macroscopic effects. Source: [Stö00], [Y$^+$06]

quarks increases with growing distance, thus making it impossible to observe solitary quarks. Consulting quantum chromodynamics (*QCD*) yields another way to describe this phenomenon. Six colors are stated by *QCD*: red, green, blue and the corresponding anti-colors. *Confinement* can then be expressed as a particle having to be color neutral.
As a consequence, two kinds of composite quark particles, so-called *hadrons* exist. Since color and anti-color cancel each other out, the resulting particle when combining a quark and its anti-quark is a *meson*. Another way to achieve color neutrality is, in analogy to the classical additive color mixing, to combine three quarks of different colors. Particles consisting of three quarks or anti-quarks, resulting in neutral color, are named *baryons*.

QCD predicts a phase transition from the hadronic into a partonic phase, where *confinement* to *mesons* and *baryons* is abolished and quarks can move quasi-free, the so-called Quark-gluon plasma. Studying the transition and the phase itself might no only yield a better understanding of the properties of the strong interaction, but also give insight into the formation of particles in early cosmological history.
Heavy-ion collisions in *ALICE* are expected to provide very hot *QGP* via hard initial parton scattering. With particle densities of several thousand per unit of rapidity for each collision, the demands to the detector in terms of resolution are high, and massive amounts of data are produced. Since real-time processing or the transfer to off-line storage is impossible even today, a sophisticated hierarchical trigger system preselects, based on several distinct observables, interesting physics events for permanent storage. As part of the *ALICE* trigger system, the Transition Radiation Detector's Global Tracking Unit contributes to the trigger decisions, but is also responsible for the readout and buffering of raw event data.

This thesis presents a design for the Supermodule Unit, a control and concentrator board inside the *GTU*. Developed for fault-tolerant controlling of a *GTU* segment, it interfaces event data buffers and the trigger system, and administers event storage and transmission to the data acquisition system. In the following chapter, an introduction to the *LHC*, the *ALICE* experiment and its detectors are given. Layout and functional principle of the Transition Radiation Detector are explained as well as those of data acquisition and trigger system. Chapter 3 gives an overview over the *GTU* and its components. Design changes to the Detector Control System board, which were necessary to integrate the *SMU* into the *GTU* project, are presented. In chapter 4, the communication with the *ALICE* trigger system and the reception of trigger messages on the *GTU* is addressed. Chapter 5 summarizes the requirements to the *SMU* and gives detailed insight into the implemented entities, which control the operation of the segment. The data path from the event buffers of the Track Matching Units through the SMU to the *DAQ* is the subject of chapter 6. Furthermore, first test results with Supermodule, *GTU* and data acquisition are presented. Chapter 7 summarizes the current status and shows perspectives for future improvements.

2 The ALICE Experiment

At CERN, the European Organisation for Nuclear Research, the nature of matter and the fundamental interactions have been under investigation since its foundation in 1954. With particle accelerators and large computing resources, CERN provides more than 6500 physicists from over 80 countries with an ideal infrastructure to conduct this kind of research.

2.1 Introduction

The Large Hadron Collider Near Geneva, a new particle accelerator, which is currently being installed in the tunnel of its predecessor LEP, is close to completion. At the *Large Hadron Collider (LHC)*, particles will collide at center-of-mass energies of 14 TeV for proton-proton collisions, and 1148 TeV for heavy ion collisions. When operating in heavy ion mode, the center-of-mass energy currently achieved at *RHIC*[1] will be exceeded thirtyfold by the *LHC* and the luminosity by a factor of ten. This will render possible a new generation of physics experiments.

Protons and heavy ions are produced in the linear accelerators *LINAC 2* and *LINAC 3*, respectively, before they are fed into the *Proton Synchrotron* and the *Super Proton Synchrotron*. Having passed these early stages of the CERN accelerator complex (see figure 2.1), two counter rotating beams of particles are injected into the two beam pipes of the *LHC* ring and are accelerated further until they reach the desired energies. A very high magnetic field is necessary to keep a beam current of 0.58 A on track. The required field of 8.4 T can only be produced utilizing superconducting magnets. In the *LHC* tunnel which measures 27 km in circumference, more than 1300 of these magnets were installed and are currently being tested. A selection of *LHC* beam parameters is shown in Tab. 2.1.

The particle beams collide at four interaction points, where the *LHC* experiments are located. One of the main objectives of the *ATLAS*[2] experiment is to investigate the nature of mass and to verify the existence of the Higgs boson in head-on proton-proton collisions. Furthermore, experimental proof for theories beyond the Standard Model, e.g. the detection of supersymmetric particles, shall be provided. The *CMS*[3] experiment pursues the

[1] *RHIC*: Relativistic Heavy Ion Collider, BNL
[2] *ATLAS*: A Toroidal LHC ApparatuS
[3] *CMS*: Compact Muon Solenoid

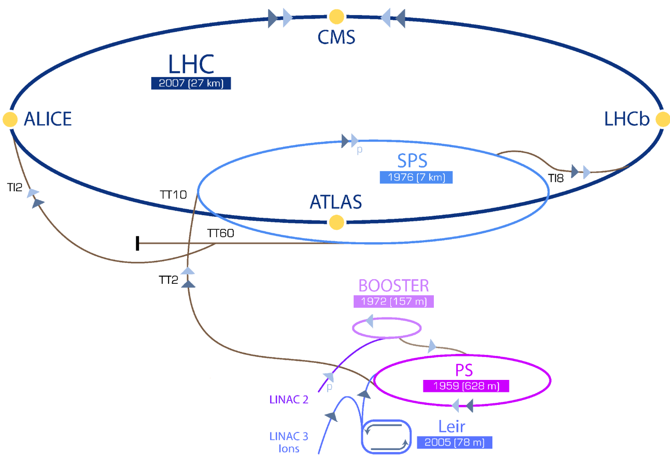


Figure 2.1: Relevant facilities of the CERN accelerator complex

same goals as *ATLAS* and allows mutual verification of the acquired results. *LHCb*[4] is designed to study b-mesons, especially CP violation in interactions of b-mesons. *TOTEM*[5] will share its location with the *CMS* Experiment, and will be used to measure total cross section, elastic scattering and diffraction processes. A smaller astro-particle physics experiment, *LHCf*[6], studies particles in the forward region of collisions in order to compare the data to shower models that are used to estimate the energy of ultra-high-energy cosmic rays. *ALICE*[7] is the only experiment especially designed to operate in heavy-ion mode. The main objective is the detection of quark-gluon plasma (*QGP*) and the study of its properties.

Quark-gluon Plasma According to Quantum Chromo Dynamics (*QCD*), quarks and gluons carry an uncompensated colour charge and can not exist in isolation (*confinement*). Gauge field theories predict a phase transition from hadronic gas to a phase, where the quarks and gluons are not confined to hadrons, but can move *quasi-free*. This state, where confinement is broken and chiral symmetry restored is called quark-gluon plasma (*QGP*). Estimations are that for the transition from normal nucleons to *QGP*, temperatures around 10^{12} K and energy densities of 3–5 GeV fm^{-3} are needed.
In high-energy collisions, the produced system is short-lived, of small latitude and expands rapidly. The initial momentum of a particle along the beam axis is, partially or completely, converted in successional clashes with particles from other nuclei into transverse momenta and the production of new particles, thus resulting in a highly populated interaction zone. While expanding, the system undergoes a series of transitions, starting

[4]*LHCb*: The Large Hadron Collider Beauty Experiment
[5]*TOTEM*: Total Cross Section, Elastic Scattering and Diffraction Dissociation
[6]*LHCf*: LHC forward
[7]*ALICE*: A Large Ion Collider Experiment

General	Ring circumference	26659 m
	Revolution frequency	11.245 kHz
	Main bend field, injection	0.54 T
	Main bend field, collision	8.33 T
	Energy per charge, injection	0.45 TeV/e
	Energy per charge, collision	7 TeV/e
Pb-Pb Operation	Atomic number	82
	Nuclear number	208
	Energy per nucleon, injection	0.18 TeV/u
	Energy per nucleon, collision	2.46 TeV/u
	Center-of-mass energy, collision	1148 TeV
	Number of bunches	592
	Ions per bunch	$7 \cdot 10^{-7}$

Table 2.1: Selection of significant LHC parameters

from a non-equilibrium state (see figure 2.3) via *QGP* to hadronic matter (*hadronization*), where partons are recombining to hadrons and fragmenting into jets. This is followed by a chemical freeze-out[8] and eventually the thermal freeze-out[9], resulting in free hadrons. The creation of hot *QGP* is expected in the compression zone inside the *ALICE* detector.

2.2 A Large Ion Collider Experiment

Studying the non-perturbative aspects of *QCD*, primarily the intensive examination of the creation of *QGP*, its dynamical evolution and phenomena associated with the ensuing phase transitions is the general aim of the *ALICE* experiment.

Detector Layout The *ALICE* detector (figure 2.4) comprises a complex of particle tracking and identification devices, complemented by a muon detection arm in forward rapidity region. The central barrel section which is housed in the solenoidal L3-Magnet, covers a range in pseudo-rapidity of $-0.9 \leq \eta \leq 0.9$ and consists of *ITS*, *TPC*, *TRD*, *TOF*, *HMPID* and *PHOS*. The forward muon spectrometer covers the pseudo-rapidity region of $2.5 \leq \eta \leq 4.0$, and is composed of the Muon Tracking Chambers and Muon Trigger Detectors.

[8]Chemical freeze-out: Due to the stop of all inelastic interactions, particle species are only changed by decays.

[9]Thermal freeze-out: All elastic interactions have stopped.

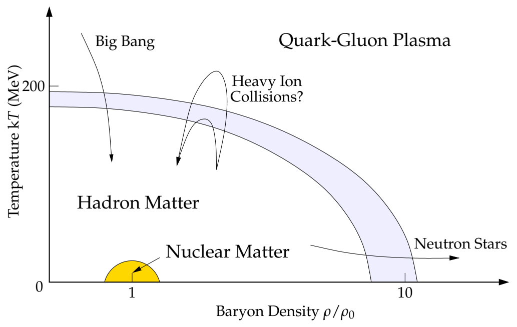


Figure 2.2: The phase diagram of the strong interaction (*QCD*). Adapted from: [PRSZ01] [dC03]

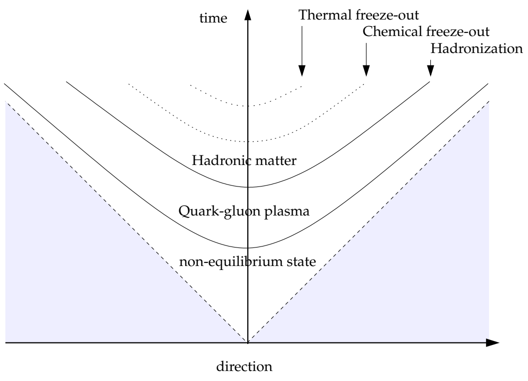
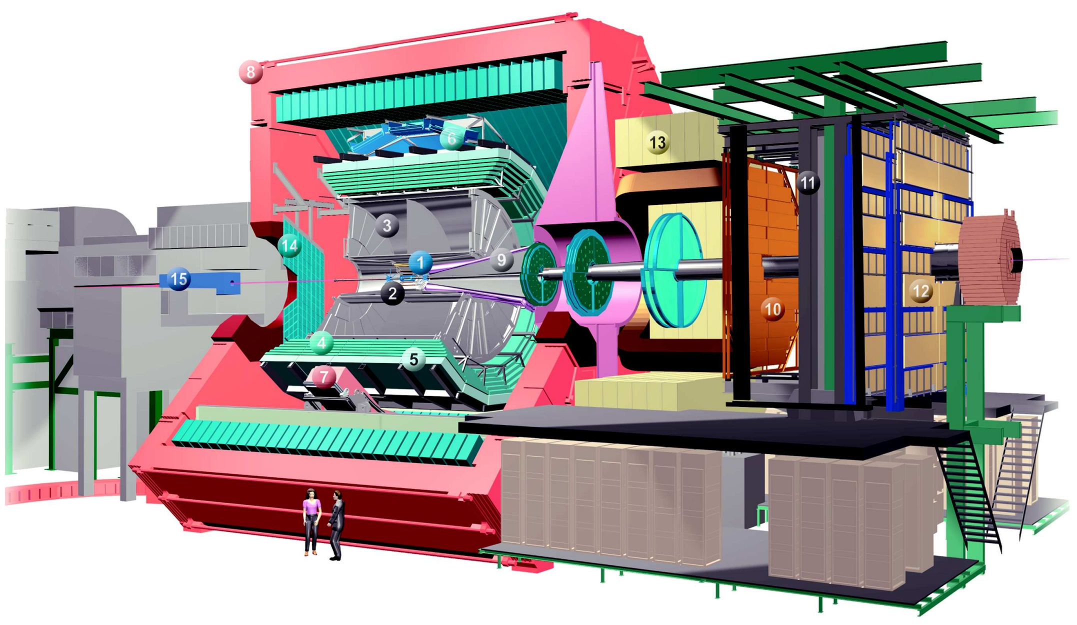


Figure 2.3: Space and time evolution of *QGP* created in a heavy ion collision. Adapted from: [Sat03]

Particle Tracking The Inner Tracking System (*ITS*) is the innermost detector in the central barrel. It consists of six cylindrical layers of silicon detectors. Because of a high track density of 90 cm^{-2} for the inner two layers , high granularity silicon pixel detectors have been used, whereas for the outer layers, silicon drift and silicon stripe detectors are utilized. Objectives of the *ITS* are localization of charged hyperon and muon decays with a resolution better than 100 µm, and the tracking of particles with $p_t \leq 100\,\mathrm{Mev/c}$ which will not reach the *TPC*.
The main particle tracking device in *ALICE*, the Time Projection Chamber (*TPC*), surrounds the *ITS* and expands to an outer radius of approximately 250 cm, measured from the interaction point. When particles travel through its cylindrical gas volume, gas molecules will be ionized along the trajectory. While the z coordinate is determined by the time secondary electrons which are accelerated by a homogeneous electric field parallel to the beam axis need to drift towards the readout chambers, x and y positions can be directly measured. Before reaching the cathode pads, electrons need to pass an amplification region which is shielded by a gating grid. The grid opens only after a Level1 trigger[10] and closes after one drift time interval (typically 90 µs) to prevent the buildup of space charge for non-triggered interactions.
In forward direction, the muon arm measures the decay of quarkonium resonances like J/Ψ, Ψ', Υ and Υ' in the $\mu^+\mu^-$ channel. Still inside the L3-Magnet, a passive front absorber shields the high granularity tracking system which consists of five stations of two tracking chambers each. A second absorber is installed before the muon trigger chambers which cumulates data for a Level0 trigger contribution.

Particle Identification Besides providing event raw data, the primary goal of the Transition Radiation Detector (*TRD*) is to track and identify electrons with $p_t \geq 1\,\mathrm{GeV/c}$, and

[10]For details of the *ALICE* trigger system, please refer to chapter 2.3.



1. ITS (Inner Tracking System)
2. FMD (Forward Multiplicity Detector)
3. TPC (Time Projection Chamber)
4. TRD (Transition Radiation Detector)
5. TOF (Time-of-Flight Detector)
6. HMPID (High-Momentum Particle Identification Detector)
7. PHOS CPV (Photon Spectrometer Charged Particle Veto Detector)
8. L3 Magnet
9. Absorber
10. Tracking Chambers
11. Muon Filter
12. Trigger Chambers
13. Dipole Magnet
14. PMD (Photon Multiplicity Detector)
15. Compensator Magnet

Figure 2.4: Layout of the *ALICE* detector. Adapted from: [CER04]

to provide a trigger contribution to the Level1 trigger. A more detailed description of the *TRD* follows in chapter 2.5.
Adjacent to the *TRD*, the Time-Of-Flight detector (*TOF*), is placed which is optimized for detection of particles in the medium momentum range. *TOF* will identify pions and kaons up to 2.5 GeV/c and protons up to 4 GeV/c with a flight time resolution better than 150 µs. *TOF* is also capable of participating in the Level0 trigger decision.
Further away from the beam axis, the High Momentum Particle Identification Detector (*HMPID*), the Photon Spectrometer (*PHOS*) and the Electromagnetic Calorimeter (*EMCAL*) are installed.
HMPID is a proximity-focusing ring imaging Cerenkov detector, designed for the identification of hadrons with a transverse momentum of $p_t \leq 1\,\mathrm{GeV/c}$.
PHOS, a high resolution electromagnetic calorimeter, contributes to the Level0 and Level1 trigger decisions. Photodiodes, coupled to scintillating lead-tungstate crystals, enable direct measurement of photons as well as measurement of neutral mesons π_0 and η.
EMCAL provides the ability to identify photons directly and offers hadron rejection. Furthermore, it adds a jet-trigger and the capability to study medium-induced modification of jet fragmentation.

Trigger and Forward Detectors Supplementary, several smaller detectors which are located in forward direction ($\eta \geq 4.0$), provide possibilities for fast global event characterization and triggering. Detectors included are *ZDC*[11], *FMD*[12], *PMD*[13], *V0* and *T0*, the latter being used to generate a pre-trigger signal for the *TRD*. An array of scintillators, *ACORDE*, is located on top of the L3 Magnet and can be used for triggering on cosmic rays.

2.3 The *ALICE* Trigger System

Operating in Pb-Pb mode, collision rates of 8 kHz are expected in *ALICE*, whereof central collisions will occur at a rate of 1 kHz. The total data rate produced by all sub-detectors will be 20 TB/s. Readout of the sub-detector with the largest data volume, the *TPC*, takes about 5 ms which corresponds to a maximum readout rate of 200 Hz. Therefore, the detector is not capable of taking data for every event, neither can the full data stream be stored for later analysis. However, the governing principle of the physics under investigation is pure statistics. Many events that will occur quite frequently have already been studied in detail in the past and are of no particular interest. Some events will happen only rarely, and investigation and analysis of these events is of great significance. To address this situation, *ALICE* incorporates a complex trigger system.

Trigger Distribution Together with other *LHC* experiments, the RD-12 Trigger Timing and Control system (*TTC*) is used in *ALICE*. *TTC* is a optical distribution system, which is used to synchronize the detector electronics (i.e. provide the 40.08 MHz bunch-crossing clock, bunch counter reset and event counter reset) to distribute the triggers and a variety of other signals. Two data channels, A and B, are time division multiplexed, bi-phase mark encoded and transmitted via an optical fibre. Channel A is used to transmit Level0 and Level1 triggers, channel B can be used for transmission of arbitrary data (e.g. trigger messages). The *LHC* clock is not transmitted explicitly. Dedicated *TTC* receiver chips (*TTCrx*) at the destination are used to recover it.

Trigger Hierarchy Depending on the input of the trigger sub-detectors and the busy state of the detector system, the Central Trigger Processor (*CTP*) decides whether to start a trigger sequence or not. Chronologically following the progress of a collision and taking into consideration the requirements of various detectors, the trigger system is broken down into four levels of hierarchy. The trigger inputs to the *CTP* are grouped with respect to their latency:

[11]*ZDC*: Zero Degree Calorimeter
[12]*FMD*: Forward Multiplicity Detector
[13]*PMD*: Photon Multiplicity Detector

Input signals which are available 800 ns after the interaction form the basis for issuing a Level0 (L0) trigger. The L0 which is transmitted via fast copper coaxial cables, arrives at the front-end electronics (*FEE*) at 1.2 µs. The L0 trigger is transmitted via channel A of the *TTC* system.
The *CTP*s decision whether to send a Level1 (L1) trigger is based on the value of the trigger inputs at 6.1 µs. The L1 trigger is also sent via *TTC*s A-channel at 6.2 µs and is expected at the *FEE* at 6.5 µs $\pm\Delta t$, where Δt is a programmable margin. If the L1 is not seen within this time interval, it is regarded as aborted trigger sequence, thus the sub-detectors will stop processing the current event. A certain time after the L1 has been issued, a L1 trigger message is transferred via the B channel of the *TTC* system. This message contains information like event identification and trigger details.
After evaluation of the past-future protection condition[14], the *CTP* provokes the transmission of a Level2 accept message (L2a) which starts data transmission to the Data Acquisition (*DAQ*), or a Level2 reject message (L2r). Message arrival is expected by the *FEE* in a window between 80 µs and 500 µs with a rate of approximately 1 kHz.

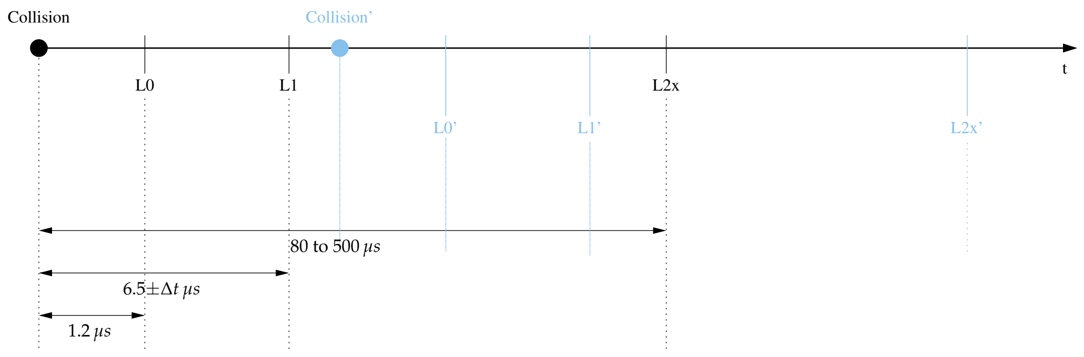


Figure 2.5: Trigger sequence arrival at the *FEE*

As a fourth trigger stage, the High Level Trigger (*HLT*) does an online reconstruction of the event, based on data from several sub-detectors, including *TRD* and *TPC*. Due to the vast data amounts produced by *ALICE* and due to the limited bandwidth from detector to permanent storage, event data has to be filtered and compressed. About 30 events per second will be stored.
As hinted in figure 2.5, overlap of several trigger sequences is possible. While it is illegal to start a new sequence whilst in between the L0-L1 time interval, it is possible to start new sequences in the long gap before the L2x trigger[15], if the desired cluster of sub-detectors is not busy.

[14]Past-future protection: a measure to prevent pile-up and overlapping events
[15]L2x marks any L2 event (L2a, L2r or a timeout for the expected arrival of the L2 message)

2.4 The *ALICE* Data Acquisition System

In *ALICE*, the link between detector front-end electronics (*FEE*) and permanent data storage is established by the Data Acquisition System (*DAQ*). Based on a network of high-speed links, data is transported from the *FEE* to the *DAQ* PCs in the counting room. Event fragments sent via the Detector Data Links (*DDL*) are assembled into full events which are potentially, depending on the decision of the High-Level Trigger, transferred to permanent storage.

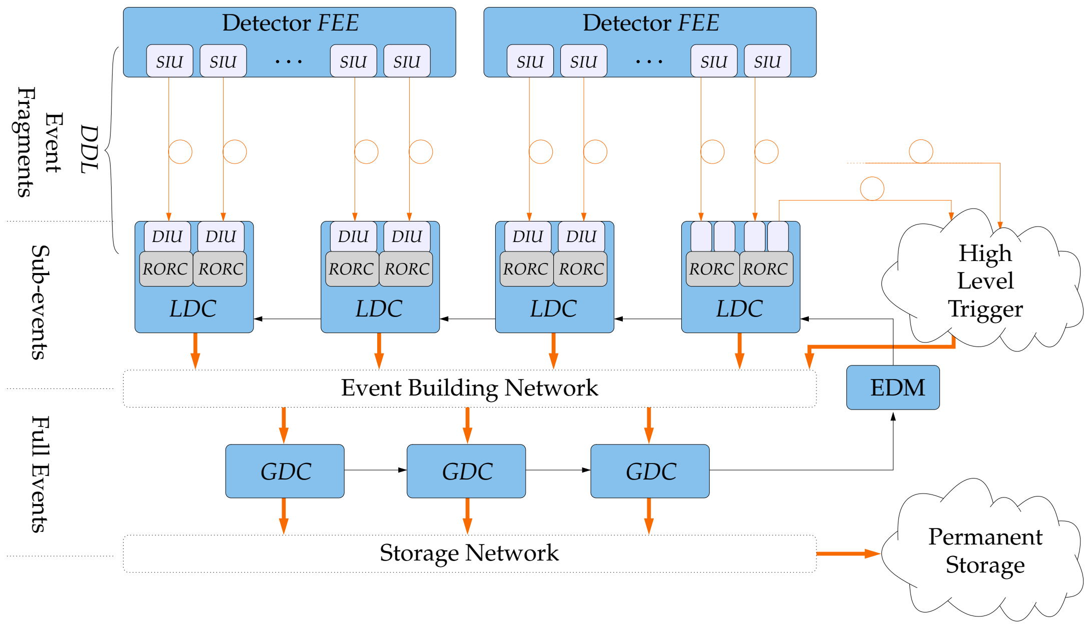


Figure 2.6: Simplified overview of the *ALICE* Data Acquisition system hierarchy

Figure 2.6 shows a simplified overview of the hierarchical *DAQ* structure. Two *DDL* interface units, the Source Interface Unit (*SIU*) and the Destination Interface Unit (*DIU*) are located at the *FEE* and, at the other end, in the Local Data Concentrators (*LDC*). The *SIU* interfaces the detector electronics and is connected by a multi-mode fibre to the *DIU*. In the *LDC*s, PCI-based *D-RORC*[16] modules are used to interface the *DIU*s. The *D-RORC*s, of which an *LDC* can handle one or several, perform transfers into the *LDC* memory where event fragments coming from the detector *SIU*s are logically combined to sub-events.
Identification for every event approved at the L2 trigger stage is supplied by the *CTP* and distributed to the *DAQ* and to the detector electronics. A Common Data Header containing global and local event identification is prepended by the *FEE* to every event fragment transmitted via the *DDL*. For events that belong to the same trigger, i. e. share a matching event ID, the Global Data Collectors (*GDC*) build a full event from sub-events shipped by

[16]*D-RORC*: *DAQ* readout receiver card.

the *LDC*s. The Event Destination Manager (EDM) supplies the *LDC*s with information about the *GDC* loads in order to balance it among the *GDC*s.

2.5 The Transition Radiation Detector

Substantial for the investigation of *QGP* is the analysis of the appearance of heavy vector mesons ($\mathrm{J}/\psi, \Upsilon$) [ALI01, MS86]. Since they are only produced rarely (e.g. every 10^5 collisions for Υ), triggering is inevitable. These resonances can be detected by their decay into electron-positron pairs with high transversal momentum. Therefore, the main objective of the *TRD* is the detection of electrons and positrons with $p_t \geq 3\,\mathrm{GeV/c}$ and the derivation of a trigger contribution to the L1 trigger decision.

Transition radiation is emitted when a highly relativistic charged particle crosses the boundary of two media with different dielectric constants ϵ. It is heavily collimated in forward direction ($\theta \propto \gamma^{-1}$), and the total loss of particle energy, corresponding to soft X-ray radiation, is proportional to the Lorentz factor $\gamma = E/mc^2$. Since $m_{pion} \approx 273 m_{electron}$, transition radiation for pions can be neglected. The effect is used in order to distinguish between electrons and pions, whose production is outnumbering the production of electrons in the targeted momentum range.

Design and Geometry The *TRD* is divided in azimuthal direction in 18 entities (so-called Supermodules) which form a hollow cylinder with an inner radius of 2.9 m and an outer radius of 3.7 m. Each Supermodule consists of five stacks, contiguously placed in z-direction. A stack consists of six modules which themselves are formed by two half-chambers. Each half-chamber comprises 6-8 pad rows with 144 readout channels each and 48 to 64 Multi-Chip Modules (*MCM*s). The 540 *TRD* modules are connected to the Global Tracking Unit[17] using two optical fibres each.

A module[18] is a unit made up of radiator material, drift chamber and readout electronics [ALI01]. A Detector Control System Board (*DCS*), hosted on top of the readout electronics, serves as uplink to the higher level control infrastructure. As radiator, a structure made of Rohacell foam, HF71 and polypropylene fibres is used. Because of the many transitions between media in the foam, the probability for emission of transition radiation is enhanced to a sufficient level. Adjacent to the radiator, gas-filled (85% Xe, 15% CO_2) drift chambers are placed. Unlike electrons and pions which provoke a continuous ionization trail, radiation photons abruptly lose energy resulting in a spatially confined charge signal. Because of the applied homogeneous electrical field, secondary electrons drift towards the amplification region with constant velocity. Given this, the x-coordinate can directly be calculated from the drift time, and the particles track can be reconstructed.

[17] Please refer to chapter 3 for a detailed description.
[18] Please refer to figure 2.7 for nomenclature details.

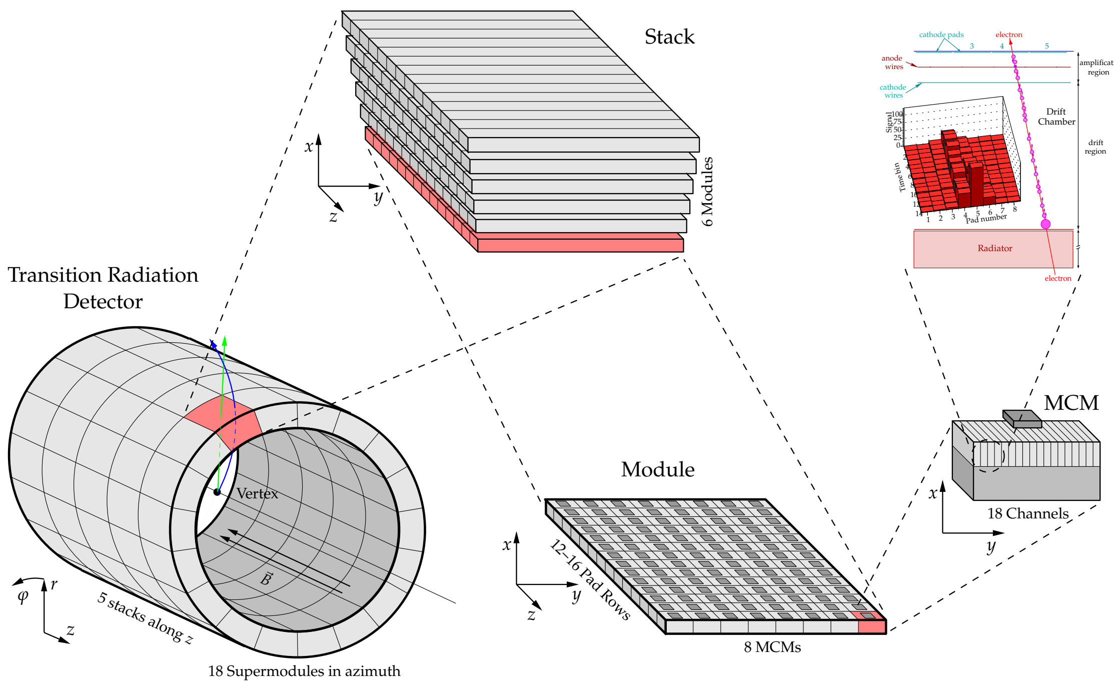


Figure 2.7: The *TRD* comprises 540 modules, each linked to the *GTU* using two optical fibres. Source: [dC03]

The drift time for the maximum distance of 3 cm is 2 µs. In the amplification region, a high voltage (U = 1.4 kV) is applied to anode wires, resulting in a strong, inhomogeneous field which causes avalanche amplification of the primary electrons. Electrons drain off through the anode wire, leaving behind gas ions which themselves induce a measureable charge on the cathode pads. The first part of the detector readout electronics is seated directly on top of the drift chambers. Combined in one Multi Chip Module (*MCM*), which is responsible for 18 pads (channels), is a charge sensitive pre-amplifier and signal shaper unit (*PASA*) and the Tracklet Processor (*TRAP*). In the *TRAP*, data is digitized and filtered [Gut02]. The Analog-Digital-Converters are running at 10 MHz, resulting in a resolution of 0.2 µs for the drift time. Based on the measured coordinates, the *TRAP* parameterizes track segments and prepares them for shipment to the *GTU*. Operation of the *TRAP*s may be supervised and controlled by the *DCS*.

Particle and Momentum Identification In case of electrons, the signal due to transition radiation photons superimposes the signal from the ionization trail. Therefore, electrons are identifiable by a higher average induced charge on the cathode pads compared to pions. Figure 2.10 and figure 2.11 illustrate this. Shown in figure 2.10 is the pulse height distribution for pions and electrons for one chamber, integrated over all time bins. Because of the large overlap of the distribution functions, data from several detector layers has to be combined and analyzed in order to improve electron-pion separation [ALI01].

In figure 2.11, a higher average pulse height for electrons is also clearly visible. The increase in pulse height for electrons towards larger drift time is a result of transition radiation which is absorbed with high probability in close vicinity of the boundary layer of radiator and drift chamber.

As a result of the magnetic field in the L3-Magnet, charged particles pass through the *TRD* on a circular path with a radius[19] in the order of 25 m for momenta of interest. For further calculation, it is assumed that the particle emerged from the primary interaction vertex. When approximating the weakly bent orbit with a straight line, gradient and axis intercept suffice for the characterization of the particle track. In order to increase tracking resolution and filter out erroneous track segments, the *GTU* analyzes parameterized track segment data (so-called *tracklets*) from all supermodules, and assesses a trigger contribution. Detailed information of track matching and particle identification on global scale can be found in [dC03].

TRD Trigger Timing and Contribution Unlike other detectors, the *TRD* expects a pre-trigger signal shortly after the collision for its electronics to wake up and start digitization of event data. Currently, the latency for the arrival of the pre-trigger at the *MCM*s is about 700 ns [Oya06]. If no L0 trigger is seen 1.2 µs after the interaction, data is discarded and the *TRD* returns to its sleep state. If a L0 is issued, fits are calculated and track segments are parameterized. The obtained tracklets are prepared for shipment to the Global Tracking Unit which happens at the latest at 4.3 µs. While still receiving, 90 *TMU*s simultaneously start processing the tracklet data in order to find electron and positron tracks and calculate the *TRD*'s trigger contribution. This tracking is finished at the latest after 1.3 µs. The *CTP* expects the trigger contribution 6.1 µs after the collision[20], and may issue the L1 trigger at 6.2 µs.

If no L1 is issued, the sequence is aborted. Otherwise, the *GTU* will accept and buffer event raw data which is transmitted by the detector at L1. After event shipment is complete, the *TRD* is ready to take new event data. Depending on the *CTP*s L2 decision, the *GTU* discards (L2r) the event, or, after event and error status information have been added to the raw data, forwards the event to the Data Acquisition System (L2a).

Detector Readout Chain In figure 2.13, a survey of the *TRD* readout chain is shown, exemplary for one supermodule. A half-chamber consists of several (3 or 4) Readout Boards and an Optical Readout Interface Board (*ORI*). On each Readout Board, data from 16 *MCM*s is gathered in a tree-like manner in order to minimize latency and merged by a

[19] Orbit radius: $r = \frac{p_t}{e \cdot B}$

[20] Since the L1 serves as pre-trigger for the *TPC*, trigger latency reduces its active volume, thus tight timing requirements have to be applied. Given a *TPC* drift time of 80 µs, a L1 latency of 6.5 µs result a loss of 8%

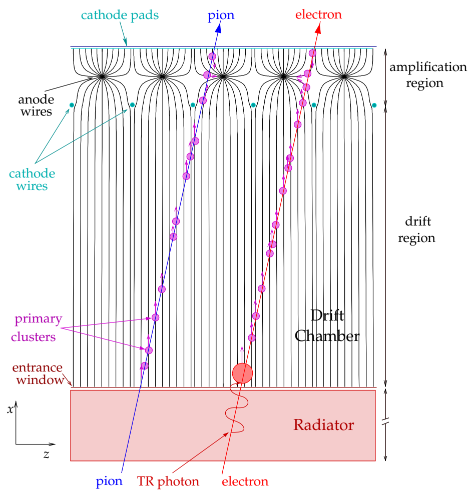


Figure 2.8: Projection of a module segment to the x-z-plane. Source: [ALI01]

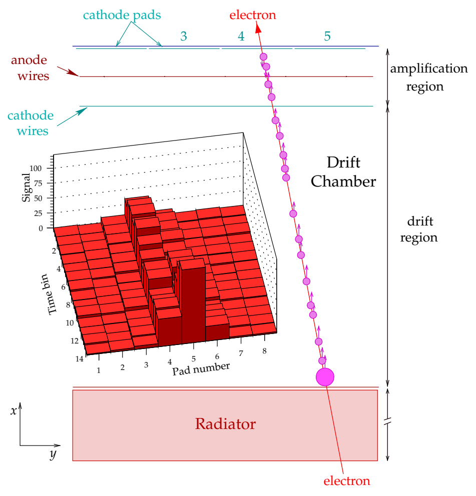


Figure 2.9: Projection of a module segment to the x-y-plane. The histogram shows the pad signal strength distribution for the drift time interval. Source: [ALI01]

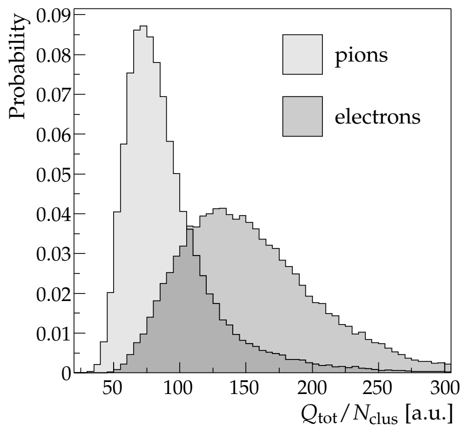


Figure 2.10: Average pulse height distribution for electrons and pions. Source: [ALI01]

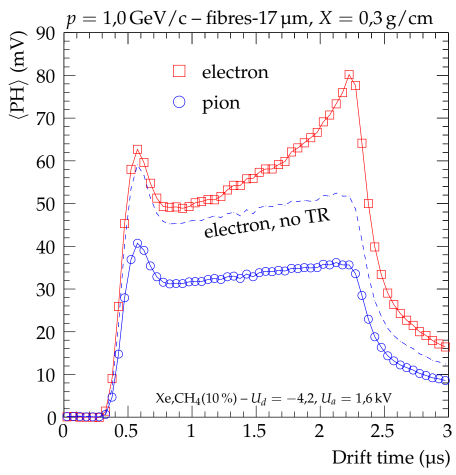


Figure 2.11: Pulse height development as a function of drift time for electrons and pions. Source: [ALI01]

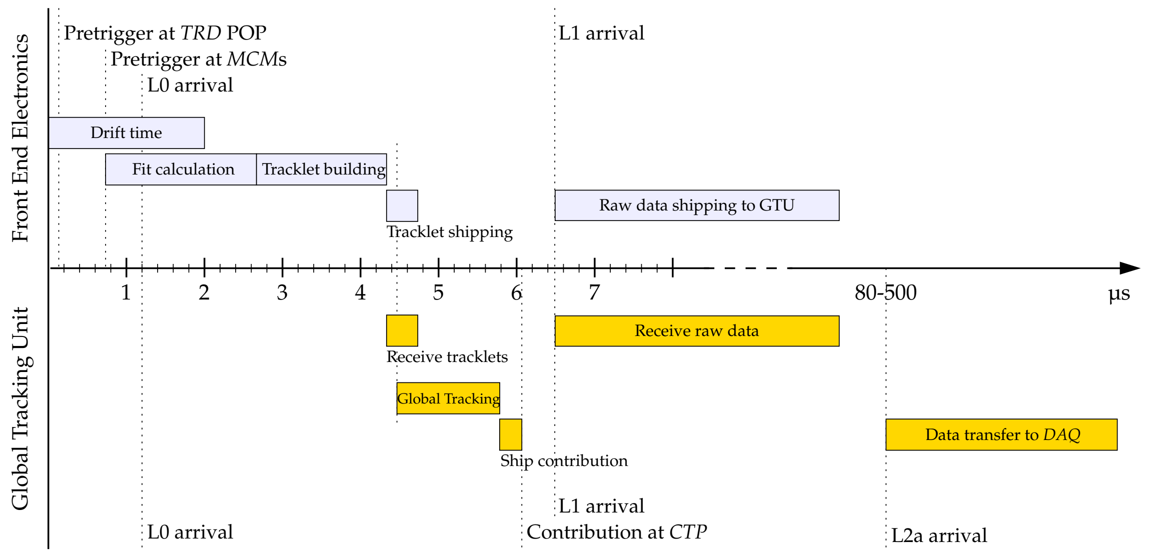


Figure 2.12: Trigger timing for *TRD* and *GTU* in a non-interleaved L0-L1-L2a trigger sequence

board-merger *MCM*. One Readout Board contains an additional *MCM* which merges the data from the board-mergers, called half-chamber merger. The data is then transmitted via an optical transmission line to the *GTU* using one *ORI*.
A *TRD* supermodule is connected to a *GTU* supermodule segment via 60 optical fibres (12 per stack, two per module). Data is received on five Track Matching Unit Boards (*TMU*). Each board receives data on 12 links and contains a unit for track and momentum reconstruction, based on the tracklet data shipped from the detector, and a unit capable of buffering data from multiple events.
The *TMU*s are connected via an LVDS-backplane to a Supermodule Unit (*SMU*) which is responsible for controlling the operation of a *GTU* segment, according to the trigger information received by the *CTP*. Here, event and status data from the *TMU*s is analyzed, updated and transmitted to the data acquisition system upon request. Trigger data is forwarded from all 18 Supermodule Units to the Trigger Unit, from which the L1 trigger contribution is communicated to the *CTP*.

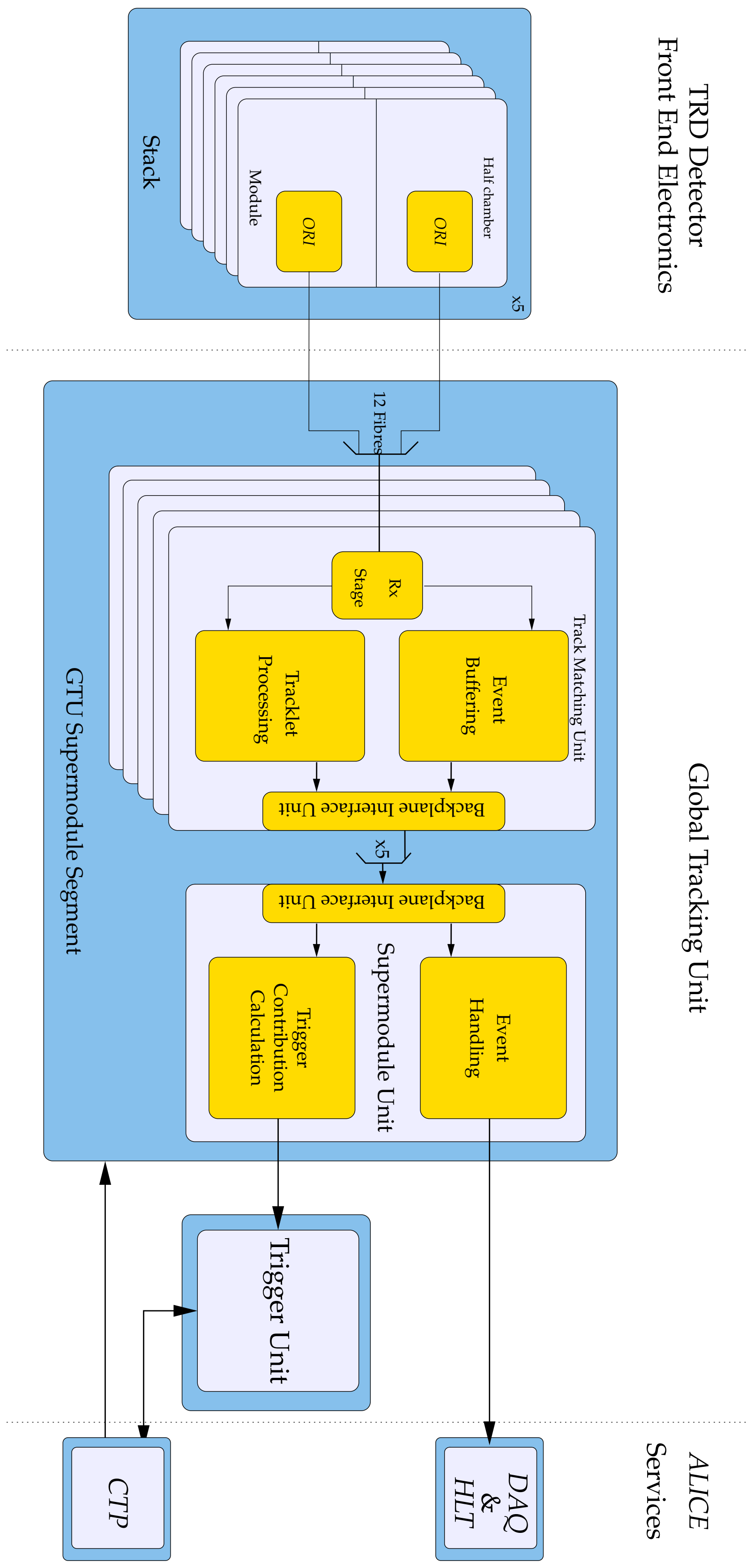


Figure 2.13: The *TRD* data path, shown as an example for one supermodule.

3 The Global Tracking Unit

Development and integration of the Supermodule Unit into the Global Tracking Unit (*GTU*) project are the aim of this thesis. In this chapter, a survey of the *GTU*, its structure, tasks, modules and interfaces, as far as concerning this thesis, is given.

3.1 Functional Principle and Objectives

The Global Tracking Unit has two main objectives. The first is to contribute to the Level1 trigger decision. Based on parameterized track segments of identified particles from the drift chambers, their trail through the detector stack is reconstructed. Fig. 3.1 illustrates the tracking approach: Track segments of the six layers are regarded as a track, if their straight extensions projected onto the y-z-plane are in close vicinity. As illustrated in fig. 2.12, the timing requirements for global tracking are very tight. In less than 2 µs, the *GTU* has to reconstruct tracks, process the reconstructed data from all Supermodules, and communicate its L1 contribution to the *CTP*. Therefore, tracking for the *TRD* stacks is performed in parallel on 90 Track Matching Units (*TMU*) and related operations are carried out with highest priority. Based on the identified tracks and their parameters, a trigger decision is communicated to the *CTP*.

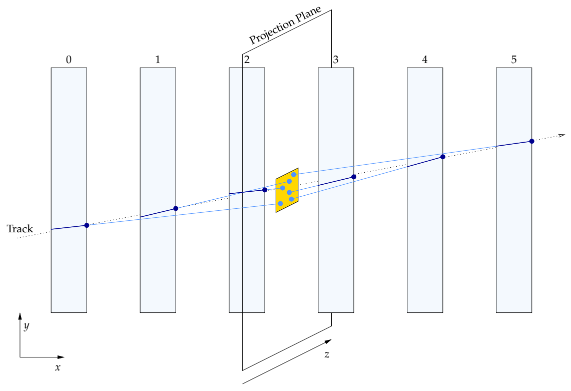


Figure 3.1: Matching of track segments in the *GTU*. Source: [dC03]

The second objective concerns the event raw data recorded by the *TRD*. As part of its readout chain, the *GTU* is also responsible for buffering and forwarding raw data to the Data Acquisition System (*DAQ*). Following the considerations elaborated in section 2.3, the *TRD* dead time has a significant impact on the performance of the whole *ALICE* experiment. In order to minimize dead time, the *GTU* is designed to store and administer raw data from several events prior to the transmission to the *DAQ*. Thus, the *TRD* detector is ready for a new event as soon as the data has been shipped to the *GTU*, which is especially important when operating with multiple interlaced trigger sequences.

3.2 Layout

Following the geometry of the *TRD*, the *GTU* consists of 18 sub-entities, each responsible for one Supermodule (see fig. 3.2). Such a segment is formed by five *TMU* and one *SMU* boards. Communication between *TMU*s and *SMU* is established by a high speed LVDS backplane, and a CompactPCI backplane, which is used for administration and control purposes. Every two of those sub-entities are placed in a 19″ crate. The resulting nine crates are situated outside the magnetic field of the L3-Magnet, one floor below the muon detection arm. One crate additionally houses a Trigger Unit (*TGU*) which serves as sending interface for communication with the *CTP*.

Board Hardware The three board types are realized as 6U-CompactPCI© boards, developed by Jan de Cuveland [dC]. They share an identical 14-layer PCB[2] design, but vary in the components equipped. The Configuration for a Supermodule Unit is shown schematically in fig. 3.3.
Common to all boards is a high-end FPGA[3] of the Virtex™-4 family, namely a XC4VFX100. This large device provides 94896 configurable logical blocks and 768 freely usable I/O pins along with a variety of specialized functional blocks. Amongst these are 20 Multi-Gigabit Transceivers (*MGT*) that can be used for serial data processing with gigabit rates and two PowerPC® processor blocks, in this case utilized for configuration and monitoring purposes. Several Digital Clock Managers (*DCM*) allow the manipulation and derivation of clock signals with a constant phase relationship to the reference clock.

The XCF32P PROM holds the FPGA configuration, which is loaded upon either power-up or request. Both, FPGA and PROM, can be programmed in system via *JTAG*[4]. As described in section 3.5.1, a suitable programming solution has been developed in this thesis, which allows to remotely program the devices. A DRAM module is intended to extend the memory of the PowerPC cores, and all boards provide the capability to

[1]Power supplies, switches, patch panels, etc. Not shown in fig. 3.2
[2]PCB: Printed Circuit Board
[3]FPGA: Field Programmable Gate Array. A device providing a large number of configurable logic blocks, connected by a programmable switch matrix
[4]*JTAG*: Joint Test Action Group. Standardized architecture for test access ports and PCB testing (IEEE 1149.1)

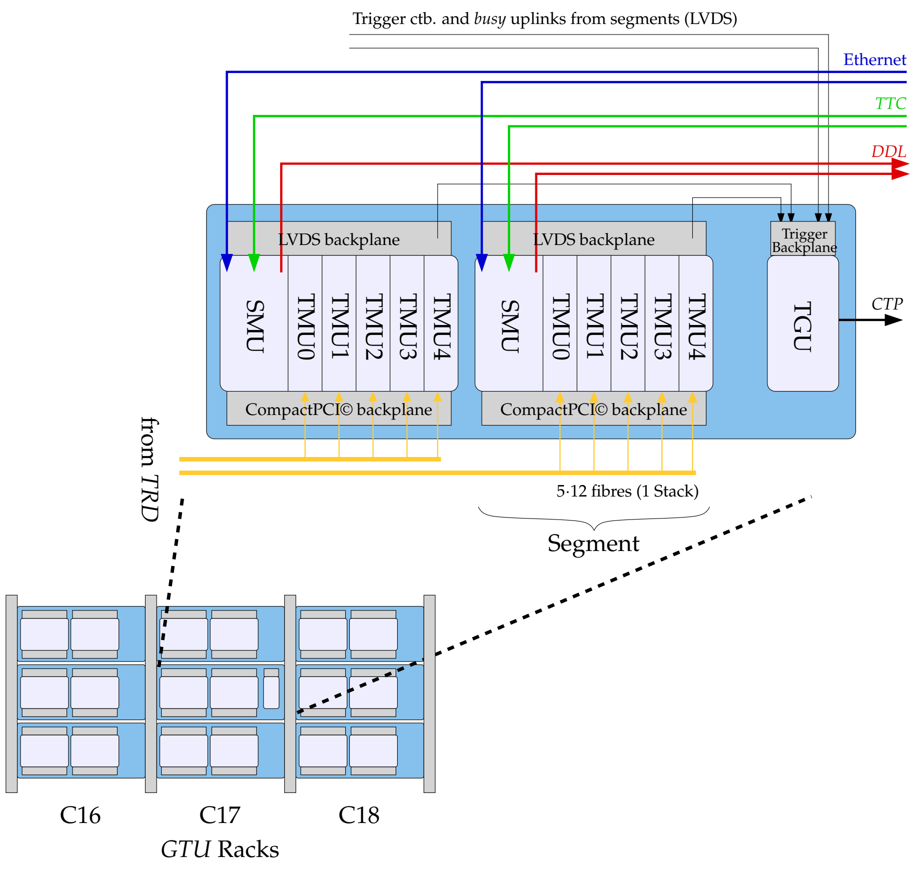


Figure 3.2: Layout of the *GTU*. Details and signal connections are shown as an example for the crate housing the Trigger Unit. The *GTU* and the necessary infrastructure [1] is situated in racks C16, C17 and C18 below the muon detection arm.

monitor temperatures (for PCB and FPGA) and operating voltages. Utilizing PowerPC and *DCS*, these working parameters can be communicated to the higher level control systems. A 7×5 dot-matrix display also allows for their instant visual indication. The reference clock of 200 MHz for the FPGA is generated by a crystal oscillator, whereas high-precision, low-jitter crystal oscillators produce the 250 MHz operating clock for the *MGT* blocks.

Two daughter boards are placed on the back of an *SMU*. The Source Interface Unit (*SIU*) establishes the optical link to the *DAQ* and the High Level Trigger. The second is a Detector Control System board, which runs Linux on an embedded CPU and is connected via Ethernet to the *TRD* network. Since there is no Ethernet jack on the *DCS* board, the one sited on the front of each *SMU* is used to establish the physical connection.
The *TGU* also holds a *DCS* board, but instead of being equipped with a *SIU* it carries an LVDS interface board to the Central Trigger Processor with two inputs and three outputs.

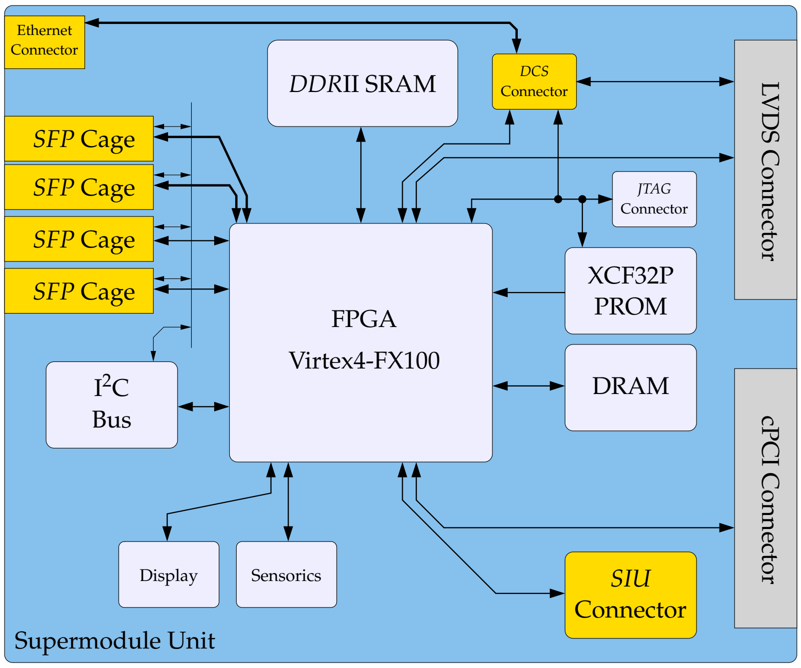


Figure 3.3: Schematic representation of a *SMU* board. Components marked in orange are not common to all board types.

Currently, two of the outputs are used for the transmission of the *busy* signal and the trigger contribution. One output and the two inputs are spare. In contrast to *TMU* boards, which do not hold any additional boards, *SMU* and *TGU* occupy two units in width.
The *DDR*II SRAM on the *TMU* boards serves as buffer for event and tracklet data. Two $512\,\mathrm{k}\times72$ modules are combined, resulting in a memory large enough to hold uncompressed data from 5 events.
*TMU*s are equipped with 12 cages for *SFP*[5] modules, to which the fibres from one stack are attached. The 4 cages situated on *SMU* and *TGU* are currently unused, but offer the possibility to implement direct gigabit ethernet connections to the embedded processors.

3.3 The Track Matching Unit

The Track Matching Unit boards each receive the data from one detector stack via 12 optical transmission lines. The fibres are plugged into *SFP* modules, which are used to convert the optical signals into differential electrical signals. Dedicated FPGA blocks for the handling of fast serial data streams, Multi-Gigabit Transceivers (*MGT*s), make the

[5]*SFP*: Small Form-factor Pluggable, standardized modules for optical transmissions. Each module has a sender and receiver unit.

electrical signals available in the FPGA fabric.
In order to minimize overall latency for the trigger contribution, the reception stage of the *TMU* hardware design is built as data-push architecture that can accept all data coming from the *TRD* detector without reverse flow control. According to its tasks, the FPGA design for the Track Matching Unit (see fig. 2.13) can be divided in two main parts, Event Buffering and Tracklet Processing.

Tracklet Processing After tracklet data has been received, processing of tracklet data from the 6 layers of a detector stack is done in the tracklet processing unit, which performs the track matching described in section 3.1. The calculated tracks are then transmitted to the *SMU* via the LVDS backplane on a dedicated channel.

A detailed description of the concept, hardware implementation and simulations can be found in [dC03]. This entity is currently being ported from Altera to Xilinx hardware and will soon be merged with the currently existing *TMU* design.

Event Buffering and Readout The Event Buffering Unit stores event data from one detector stack for several events. The Supermodule Unit receives and interprets the trigger sequences and issues corresponding control signals to the five *TMU*s in the segment in order to initiate the transmission of event data through the *SMU* to the *DAQ* (L2a) or to discard the event (L2r).

A survey of the hardware design developed by Felix Rettig ([Ret07]) is shown in fig. 3.4. The event shaper unit receives 16 bit raw data words in the 125 MHz clock domain. This data is then aligned to blocks of 128 bit, which is the size of one SRAM line. If the number of data words is not a multiple of 8, the data is padded with end markers. Four dual-port Block RAMs with asymmetric write (16 bit) and read side (32 bit) width and a ring counter are used to align and pad the data and implement the clock domain crossing to the 200 MHz SRAM clock domain.

The *data_capture*-signal issued by the *SMU* defines a frame in which data from the *TRD* will be recorded. If desired, tracklet data sent by the *MCM*s can also be recorded in the *TMU*s. In this case, the recording starts with the L0. In the resulting data stream for the event (see section 6.1), the presence of tracklet data is marked in the Supermodule Index Word. The payload for each link from each stack then consists of tracklet data and event raw data, separated by tracklet end markers (`0x1000`). For raw event data, recording starts with a L1 and the payload consists of raw data only.

The SRAM is divided into 12 logically independent blocks, each organized as a ring buffer structure. For each block, which holds the data for one link, pointers for the current write position, the current read position and the start position of the current event

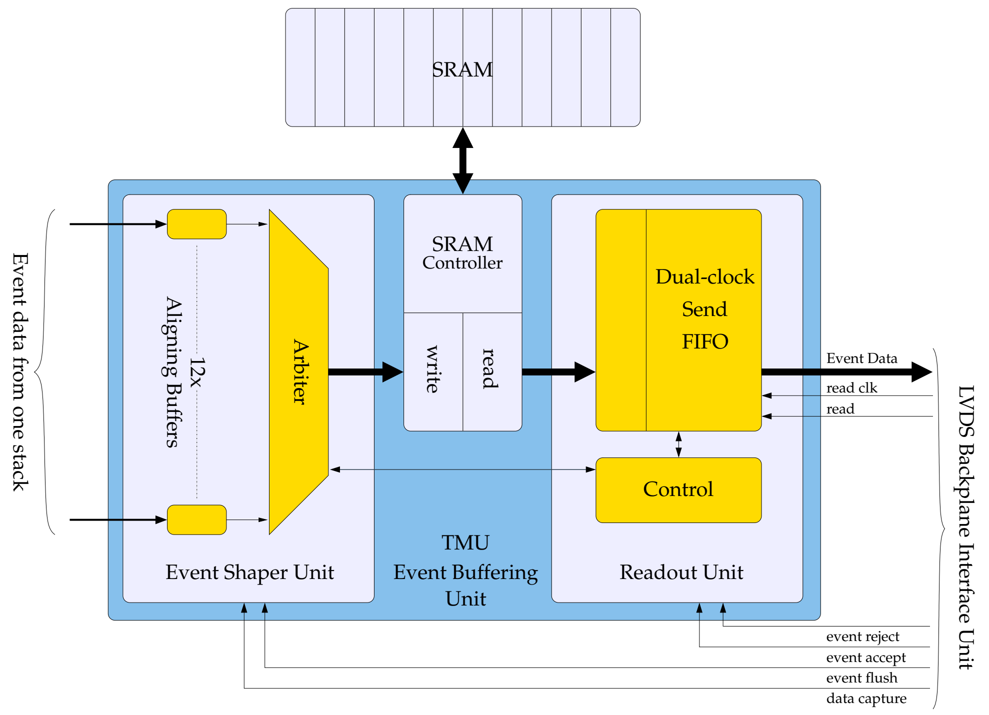


Figure 3.4: An overview of the *TMU* Event Buffering design. The design is structured in Event Shaper, SRAM Controller and Readout Unit.

are embedded. The latter is necessary to support the rejection of an event, which is currently being recorded (*event_flush*[6]). Address management for the SRAM blocks is done in the Event Shaper Unit.
The SRAM controller interfaces the external *DDR*II SRAM memory. It accepts 128 bit lines of write data from the Event Shaper and outputs equally sized read data to the Readout Unit.

Upon request by the *SMU* (*event_accept*), the Readout Unit initiates the readout of the oldest event in the external memory. It is designed to minimize latencies when accessing the SRAM in order to be able to saturate the adjacent transmission stages. Upon a L2r (*event_reject*), the oldest event is discarded and the associated memory cleared.
In the former case, the data is buffered in a dual-clock FIFO used to implement the clock-domain crossing between the 200 MHz SRAM domain and an the 120 MHz backplane domain. All readout operations are subject to a reverse flow control, ultimately caused by a saturation of the Detector Data Link (*DDL*).
The LVDS Backplane Interface which manages communication with the *SMU* on the *TMU* side is discussed in detail later on in this thesis (see section 6.2).

[6]This will occur frequently if the option to record tracklets is chosen. An event whose tracklet information was recorded and stored in the SRAM will be dropped in case of a missing L1. Furthermore, trigger errors can lead to event drops.

3.4 The Supermodule Unit

The Supermodule Unit serves as control and concentrator board for a *GTU* segment. Being the segments link to the *CTP*, the *DAQ*, the *TRD* network and the Trigger Unit, its tasks are manifold. Requirements to the FPGA design and the implemented solutions are presented in detail in the following chapters. Photographies of the *SMU* and *TMU* boards can be found in appendix A.

3.5 The Detector Control System Board for the *GTU*

The Detector Control System (*DCS*) board is used for control and monitoring purposes throughout *ALICE*. Equipped with network hardware, SDRAM and an Altera Excalibur FPGA featuring an ARM RISC processor interfaced with a programmable logic architecture, the system is capable of running Linux, thus enabling access via standard Ethernet connections. It is also equipped with a *TTCrx* chip that interfaces the *TTC* system and recovers the experiment-wide bunch-crossing clock.
As described in the following sections, the standard hard- and software of the *DCS* board were successfully extended to fulfill the requirements of the *GTU*.

3.5.1 Remote FPGA Configuration

Since the room below the muon detection arm will not be accessible once the experiment is running, FPGA and PROM configurations of the *GTU* boards need to be updatable in situ. The Xilinx devices used support the IEEE 1532 standard for In-System Configuration (ISC), which is based on the IEEE 1149.1 Test Access Port and Boundary-Scan Configuration standard [Xil06b].

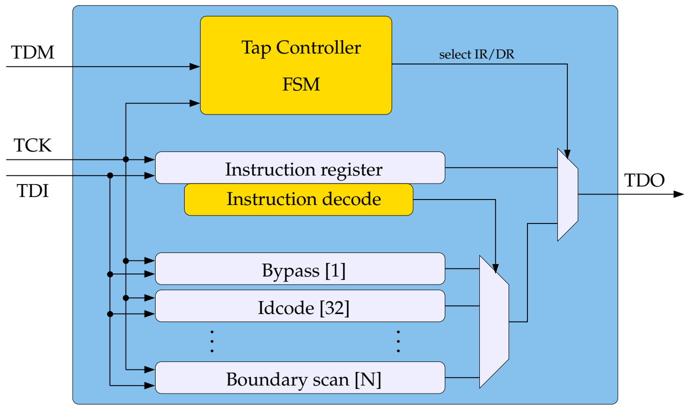


Figure 3.5: Typical IEEE 1149.1 *JTAG* architecture. Adapted from: [Xil06b]

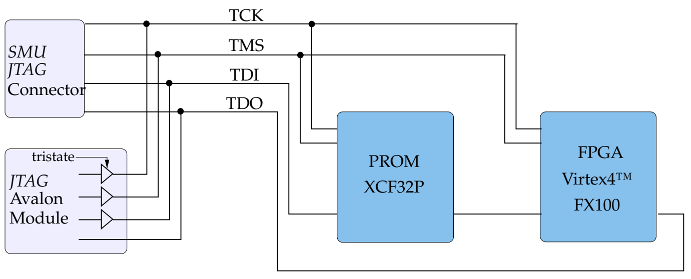


Figure 3.6: *JTAG* wiring on the *SMU* boards. An external *JTAG* adapter can be used without hardware changes to the design.

In application note xapp058, Xilinx presents an ISC solution which uses a microcontroller to program a chain of *JTAG* devices. A programming file, which can be created from *bit*-files and contains all the information needed to program the system, is replayed by a special player software in order to program the devices. Since the concept was compatible and it was possible the integrate the automatic creation of programming files into the *GTU* team build flow, it was chosen to adapt this solution. The necessary modifications in hard- and software are described in the following paragraphs.

In-System *JTAG* Configuration If the devices are daisy-chained together, a simple 4-wire interface suffices to program all the devices. The four wires are:

- TCK: Test Access Port Clock
- TMS: Test Mode Select
- TDI: Test Data In
- TDO: Test Data Out

TCK is a free running clock and used to clock the access port logic. Based on the value of TMS at the rising edge of TCK, the sequence of states of the Tap controller state machine are chosen. In principle, two sequences of states exist: Shift Instruction Register (SIR) and Shift Data Register (SDR). They determine which register is to be loaded with TDI values and which register is output on the TDO line. Pulsing TCK at least 6 times with TMS $= 1$ guarantees that the Tap controller is in its idle state.
A typical *JTAG* architecture is shown in fig. 3.5. Standard compliant implementations feature a series of data registers, such as the Bypass or the Idcode register. The register used to input the device configuration data is not shown.
Fig. 3.6 shows the wiring on the *SMU* boards. Programming can be done without changes to the board hardware using either an external adaptor, or using the *DCS* board solution with the Avalon *JTAG* module, thus preserving the possibility to use debugging tools like ChipScope.

The *JTAG* Avalon Slave The Altera Excalibur device on the *DCS* board combines a RISC processor system with a programmable logic device. The processor stripe of the device (EXPA1) contains an ARM922T 32 bit RISC processor core, peripherals and the memory subsystem with 32 k single and 16 k dual-port memory. The processor stripes advanced high-performance bus (AHB) is used to connect to the PLD, which consists of 4160 logic cells and 186 user I/O pins.
The Avalon Bus is a simple synchronous bus architecture designed to connect on-chip processors and peripherals to a system-on-a-programmable-chip, which utilizes separate address, data, and control lines. It specifies the port connections between master and slave components and the timing of bus communications. A master component is capable of initiating a bus transfer, while an Avalon slave component can only respond to requests. On the *DCS* board, all user logic is implemented as Avalon slave module. Further details on the Avalon Bus Architecture and the Excalibur devices can be found in [Alt02a] and [Alt02b], respectively.
In fig. 3.7, the bus connection between processor stripe and the *JTAG* Avalon slave module is shown. The AHB-to-Avalon-Bridge connects stripe with the PLD. The master port on the bridge, the Avalon Bus Module which is automatically generated by the SOPC-Builder, and the *JTAG* slave form a master-slave-pair.

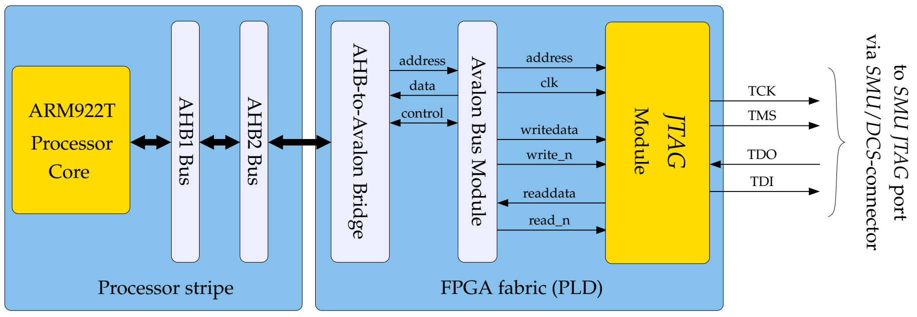


Figure 3.7: Bus connections between ARM processor and *JTAG* user logic.

The *JTAG* entity is simple in design. It consists of output registers for TCK, TMS and TDI and corresponding tristate buffers. The output registers are placed at write address 0x0 . To preserve the functionality mentioned earlier, the buffer is tristated at power-up and can be enabled by the lowest bit of the control register (0x3). Reading from addresses 0x0 and 0x3 gives the value of the TDO line and the content of the control register, respectively.
Two instances of the entity exist in the *DCS* hardware design. The first is connected to the *JTAG* wiring on the *SMU* board. The second is connected to the LVDS backplane. I^{2}C-controllable buffers and multiplexers are used here to choose which *TMU* to program. Each or all *TMU*s can be selected. In the latter case, the TDO line from *TMU*0 is selected for read back.

Linux Device Driver and Programming Software The driver to access the module in Linux is implemented as character device driver and supports the basic operations `open`, `release`, `read` and `write` and the functions `init_module` and `cleanup_module`.
When the module is loaded, `init_module` is called. It requests a file system handle by which the new hardware can be accessed and derives the virtual base address from the physical address, which is defined by the SOPC-Builder at the time the module is integrated into the system. It also sets the output tristates to active.
Since `read` and `write` are implemented as file system operations, the hardware can be addressed like a file. Two device nodes, */dev/jtag_smu* and */dev/jtag_tmu*, are created and give access to the *JTAG* port of *SMU* and *TMU*s, respectively.

The source code for the player application was freely available and was successfully ported as-is to the *DCS* hardware. In order to do that, the I/O functions and target-specific device functions had to be modified.
The Xilinx Serial Vector Format (xsvf) is a file format that contains sequences of high level IEEE 1149.1 bus operations. With a small set of instructions it is possible to move between the stable states of the Tap controller and perform scan operations. A major advantage over the Serial Vector Format is the much better data compression. A small example for xsvf-instructions shall be given:

- `XSDRB <TDIvalue>`
  Brings the Tap controller in the Shift-DR state and shifts in TDIvalue. The Tap controller remains in the Shift-DR state.
- `XSDRC <TDIvalue>`
  Assumes that the Tap controller is in Shift-DR state, shifts in TDIvalue. The Tap controller remains in Shift-DR state.
- `XSDRTDOE <TDIvalue> TDOexpected <TDOexpectedValue>`
  Assumes that the Tap controller is in Shift-DR state, shifts in TDIvalue and compares the TDO value shifted in to TDOexpectedValue. When finished, the Tap controller moves to state Run-Test/Idle.

The complete xsvf instruction set can can be found in [Xil04]. For a given, known *JTAG* chain of devices, the xsvf-file, generated from *bit*-file and the boundary scan description files of the devices in the chain, contains all the information to program the devices and verify the success of the operation.

Using the xsvf-player without major modifications in this setup worked, but resulted in an unacceptably slow programming speed[7]. Too many context switches between user and kernel mode significantly slowed down the program. A closer analysis of the programming files revealed a structure with several scan instructions at the beginning, followed by a large number of uniform data shift instructions (`XSDRTDOC`), where the actual

[7]Programming of a XCV4FX60 device took approx. 3min.

FPGA configuration was shifted in. To speed up programming, a block mode was implemented for the `XSDRTDOBCE` instructions. Here, the player passes the TDI data in blocks[8] to the kernel module, which then executes the sending of the correct TCK, TMS and TDI values. To avoid huge overhead when reading data which is accessible on network shares (as it will be the case in the final system), block–by–block read of the xsvf-file was added. This eventually resulting in a programming time of approx. 18 s for the FX60 device, compared to approx. 3 min compared to the inital implementation.

3.5.2 *GTU* Control Access

During development, the need for an interface to remotely control and monitor the *GTU* became obvious. Providing standard tools and access, the *DCS* board is an ideal host for such an interface. Due to the limited number of FPGA user I/O pins, only two data lines are available with the boards of *GTU* segment. Thus, a serial communication protocol (RS-232) was chosen and the necessary infrastructure built.

Since the fixed UART peripheral of the Excalibur device was already in use, a second UART core from opencores was added to the *DCS* board hardware design. This core ([JGM]) had a WISHBONE(WB)-compliant interface. Avalon, unlike WB, does not provide a frame signal for a valid bus transfer cycle (`wb_cyc`) and a signal to mark a valid data transfer (`wb_stb`). However, `chipselect`, which is asserted by the bus when the address was decoded, can be used as a replacement and is connected to both `wb_cyc` and `wb_stb`. The WB acknowledge (`wb_ack`) has the role of an inverted `waitrequest`. Other signal mappings are trivial. The necessary kernel driver to use the UART core and the software to communicate with the PowerPCs was written by Marcel Schuh ([Sch07]).

Fig. 3.9 shows the communication structure. The functionality of the onboard UART connector was preserved. Logged onto the *DCS* board, the user can communicate with the PowerPCs on all boards. A large number of the parameters of the *SMU* and *TMU* hardware designs can be monitored and set at runtime using one of the built-in PowerPCs, and this method enables the *GTU* to be monitored and controlled in detail from anywhere on the *TRD* network.

[8] 128 bit was an appropriate size, since this is the amount of TDI values in one shift instruction.

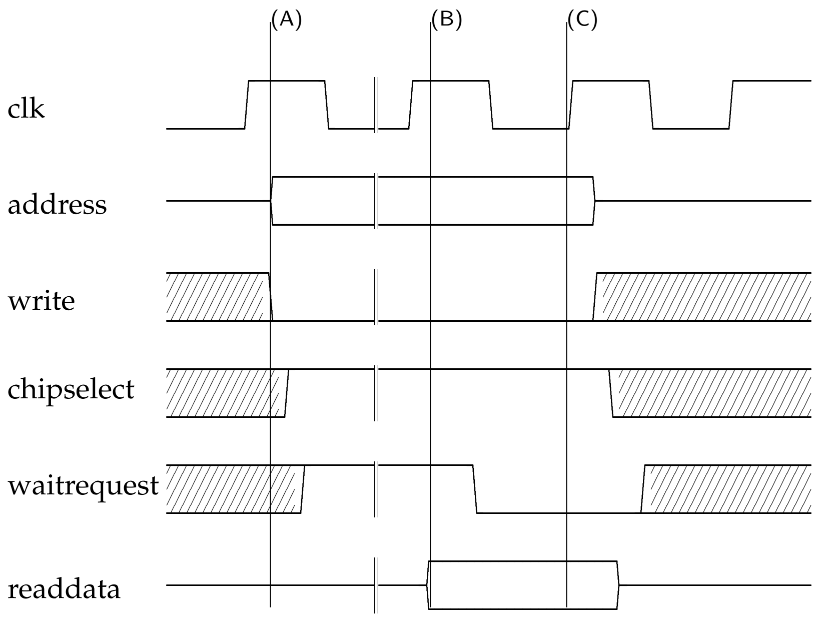


Figure 3.8: Single read transfer on the Avalon bus [Alt02a].
(A) The address and write (read_n) are valid. The bus decodes address and asserts chipselect. The slave asserts waitrequest.
(B) Readdata is valid. The slave sets waitrequest low. Several waitcycles can occure between A and B.
(C) Readdata is captured at the next rising edge of clk by the Avalon Bus Module. The bus transfer ends.

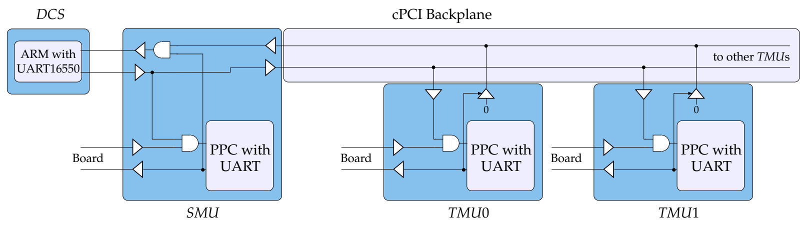


Figure 3.9: Serial communication between the boards of a *GTU* segment and the *DCS* board. Pull-ups at the FPGA output buffers are not shown.

4 Communication with the Trigger System

The *ALICE* trigger system controls the operation and the readout of all detector front-end electronics (*FEE*). In order to control a *GTU* segment, the Supermodule Unit receives information sent by the Central Trigger Processor and issues appropriate commands to the Track Matching Units. For each trigger sequence that is concluded with a L2a trigger event data from the participating detectors is sent to the *DAQ*. Correlation between these data fragments and the commands given by the trigger system can be accomplished by analysis of the Common Data Header, which contains local and global event identification.
This chapter addresses the communication between the *GTU* and the trigger system. After an introduction to the layout and functional principle, the *DCS* and *SMU* implementations responsible for receiving the *TTC* protocol are presented.

4.1 Trigger Sequence Generation and Distribution

The layout of the *ALICE TRD* trigger system is pictured in fig. 4.1. The *LHC* clock (40.079 MHz) and the *LHC* orbit[1] signal are distributed via optical fibers from the CERN Prévessin site. Received by the *TTC* machine interface[2] (*TTC*mi), clock and orbit are distributed to the detector *TTC* partitions and the Central Trigger Processor.

Central Trigger Processor The Central Trigger Processor receives data from all trigger detectors and issues a trigger decision based on parameters such as the level of centrality of the collision or the existence of high p_t electrons or hadrons. More complex trigger decisions, e.g. the geometrical matching of between several sub-detectors, are relocated to the High Level Trigger due to the short time scale on which the decisions have to be reached.
Input to the *CTP* are 24 L0 inputs with a latency l less than 800 ns, 20 L1 inputs with $l \leq 6.1\,\mu s$ and 6 L2 inputs with $l \leq 87.6\,\mu s$.

The *CTP* comprises several entities. Those which are relevant are shown in fig. 4.1. The Fan-Out boards connect to the sub-detector *TTC* partitions and transmit the trigger

[1]The *LHC* orbit signal is a square wave signal with a period of 88.924 µs.

[2]The *TTC*mi incorporates a *TTC*rx receiver ASIC and a voltage controlled oscillator with a PLL for jitter reduction from 80 ps to 7 ps.

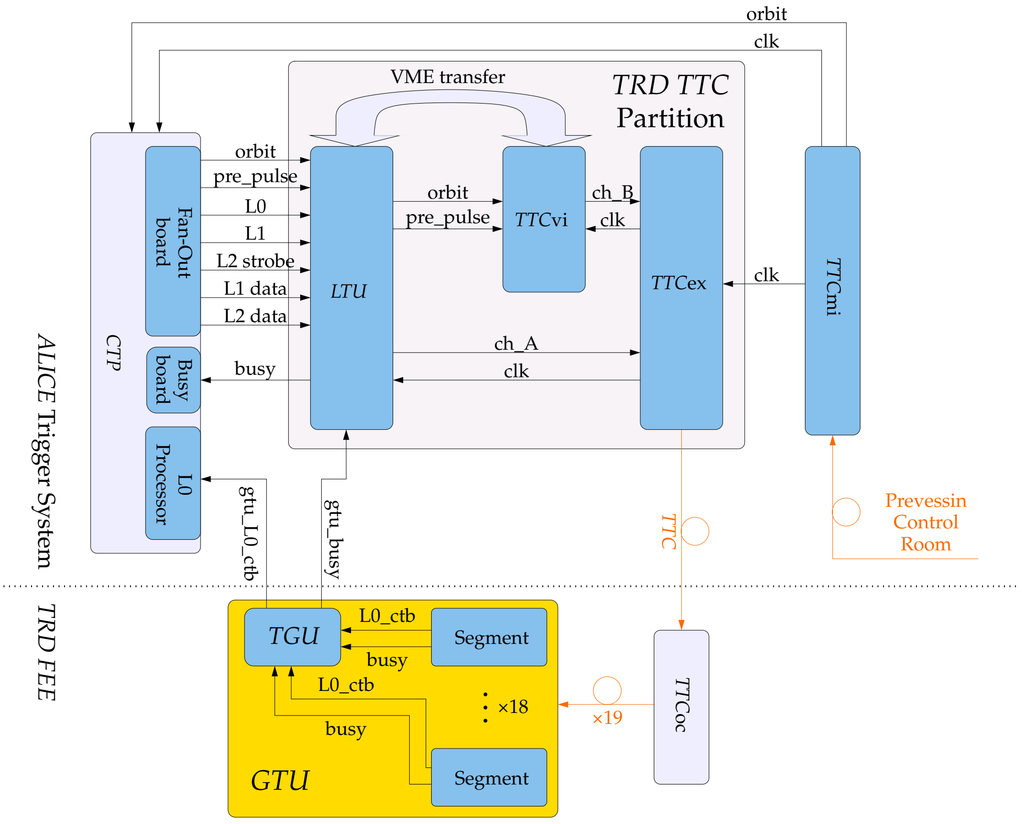


Figure 4.1: Survey of the *ALICE* trigger system and the *GTU*. Each *SMU* of a segment and the *TGU* is connected to the trigger system. Local trigger contributions and *busy* are concentrated by the *TGU*, which communicates the global signals to the trigger system.

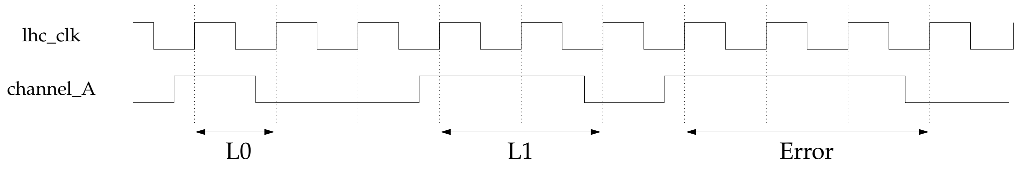
signals (L0, L1, L2 strobe and the trigger messages content) to the Local Trigger Unit (*LTU*). The Busy board and the L0 Processor are input boards which provide ports for *busy*-signals and the L0 trigger contributions, respectively.

Sub-detector *TTC* Partition Three VME system boards comprise a *TTC* partition. The *LTU* is the interface board between *CTP* and sub-detector electronics. Depending on the operation mode, global or standalone, the *LTU* receives data to be transmitted via the *TTC* system from the *CTP* or does a full emulation of the *CTP* functionality. The messages are deserialized and the 4 bit header is prepended to every 12 bit data word[3] that is received or generated. The 16 bit words are transferred via the VME bus to the *TTC*vi. The *TTC* VMEbus interface (*TTC*vi) delivers the B-channel data to the *TTC* laser drivers (*TTC*ex), which handle channel multiplexing, encoding and optical transmission. Chan-

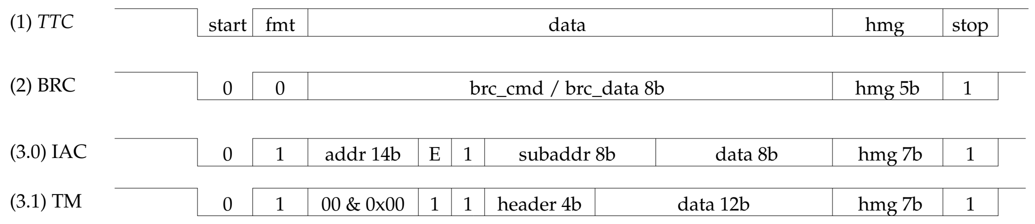
[3]TTC message format: See fig. 4.3.

nel A data is transmitted from *LTU* to *TTC*ex directly. Using a *TTC* fiber[4] the system is connected to a 1:32 optical tree coupler (*TTC*oc), which fans out 19 fibers connected to the *GTU*.

***TTC* Transmission** The *LHC* clock, the trigger signals and the trigger messages are distributed via the *TTC* system. The clock is recovered at the target destination by receiver ASICs (*TTC*rx), which also decode the data content. Channel A is used exclusively for the transmission of the trigger signals, L0 and L1, while channel B can deliver various types of data to the front-end electronics. As shown in fig. 4.2(a), data on channel A is encoded by pulse width. A L0 trigger is sent as one pulse, while a L1 is denoted by two consecutive pulses. Three consecutive pulses reflect an error condition, since the minimum time between a L1 and the next L0 trigger is two *LHC* clock cycles ([JJ06]).



(a) Channel A.
L0 and L1 trigger are encoded by one and two high cycles, respectively.



(b) Channel B.
(1) General *TTC* frame format,
(2) Broadcast frame (BRC),
(3.0) Individually addressed frame (IAC) and
(3.1) Trigger message, a special form of an IAC.
Adapted from [CMMT04].

Figure 4.2: Signal transmission via the *TTC* system in *ALICE*.

Fig. 4.2(b) shows the format sent via channel B. Each message begins with the *start* bit (0) and is followed by the *fmt* bit, which indicates whether the message is a broadcast com-

[4]50/125 µm graded index multi-mode fiber, 1310 nm.

mand (BRC)[5] or an individually addressed command (IAC). For the latter, the 14 bit address selects a specific *TTC*rx in the system[6] and the external bit *E* determines the address space[7].
In case of trigger messages the *fmt* bit, *address* and the *E* bit are set to 0x8001 , meaning that they are IACs distributed to all *TTC*rx in the system whose data content is made available to the *FEE*. Furthermore, the 16 bit of *subaddr* and *data* are renamed to give a *data* width of 12 bits, leaving 4 bit for the *header*. This message payload is provided by the *CTP*.
The message concludes with a Hamming error code, which is checked by the front-end electronics to detect double-bit and correct single-bit errors, and the *stop* bit (1).

4.2 Trigger Messages and Common Data Header

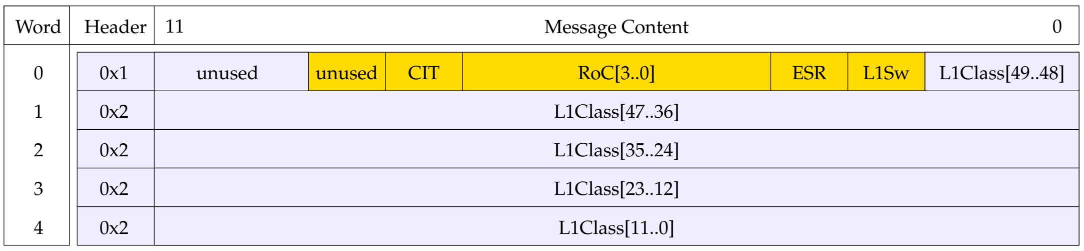
Trigger Messages The *CTP* issues three types of *TTC* messages depending on the current state of the trigger sequence. The L1 message is sent simultaneously with or after issuing the L1 trigger pulse. Due to the absence of a pulse for an L2a or L2r, the successful arrival of the messages reflects and updates the state of the trigger sequence at the front-end electronics. Arrival of the messages at the *SMU* is expected 500 µs after the L1 trigger at the latest.

The contents of all trigger messages are illustrated in fig. 4.3. A message word consists of 12 bit message content and 4 bit header.
The leading word of the L1 message (header ID 0x1) contains data used to characterize the event. The *L1Sw* status bit specifies the type of trigger (software or physics), while the readout control bits *RoC*[3..0] define special readout options, e.g. the generation of pseudo-events in case of a software trigger. The Common Data Header field *L1Message* is formed by bits [9..2] of word 0.

Information for the identification of the current event is found in the L2 messages. In Pb-Pb runs, every bunch of the foreseen 3564 bunches in one *LHC* orbit ([ALI05]) is marked by the BunchCountID (BCID). The orbits are counted by the trigger system. The OrbitID (OID) gives, together with the BCID, a unique identification of the event in a time-frame of approx. 24 minutes[8], which is sufficient for this application. BCID and OID are loaded into the *EventID1* and *EventID2* fields of the CDH, respectively.

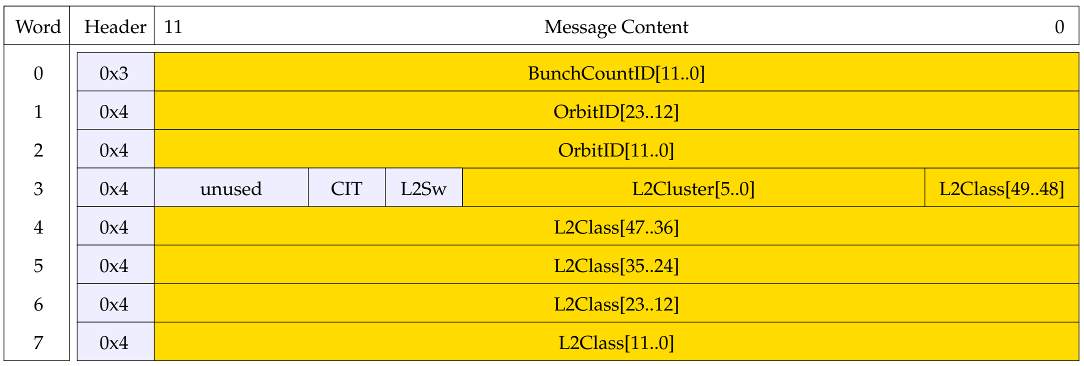
For physics triggers, the message fields *L1Class* and *L2Class* reflect the state of the 50 independent trigger classes, which are a basic processing structure of the *CTP*. In case

[5]The reset signal to the bunch counters in the *FEE* is transmitted as BRC.
[6]Address 0x0 denotes a broadcast to all *TTC*rx.
[7]The internal address space gives access to registers of the *TTC*rx ASIC, while the external address space is used to transmit data to the FEE.
[8]Unique ID: $2^{24} \cdot 88.9\,\mu s = 1492\,s$.



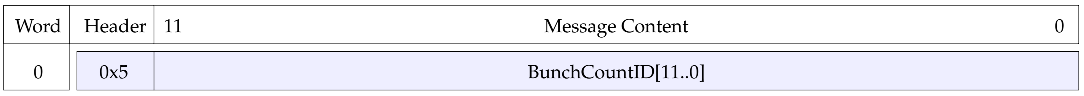
Word	Header	11			Message Content			0
0	0x1	unused	unused	CIT	RoC[3..0]	ESR	L1Sw	L1Class[49..48]
1	0x2				L1Class[47..36]			
2	0x2				L1Class[35..24]			
3	0x2				L1Class[23..12]			
4	0x2				L1Class[11..0]			

(a) L1 message



Word	Header	11		Message Content		0
0	0x3			BunchCountID[11..0]		
1	0x4			OrbitID[23..12]		
2	0x4			OrbitID[11..0]		
3	0x4	unused	CIT	L2Sw	L2Cluster[5..0]	L2Class[49..48]
4	0x4			L2Class[47..36]		
5	0x4			L2Class[35..24]		
6	0x4			L2Class[23..12]		
7	0x4			L2Class[11..0]		

(b) L2 accept message



Word	Header	11	Message Content	0
0	0x5		BunchCountID[11..0]	

(c) L2 reject message

Figure 4.3: Trigger messages arriving at the *GTU*. A message word comprises of a 4 bit header used to define the type of word and 12 bit data payload. The content of the fields marked yellow is used to build the Common Data Header. Adapted from [CMMT04].

of software trigger sequences[9], the *L2Class* field is used to transmit information on the participating sub-detectors. Word 4 and 5 of the L2a message are supplied to the CDH as *Participating Subdetectors*. Further details about the trigger messages and their generation can be found in [Jov04].

Common Data Header The Common Data Header (CDH) contains information about the type of event and the trigger details as well as unique event identification. It is generated by the *SMU* from the content of the trigger messages and *GTU* segment-specific information and prepended to every event fragment sent via the *DDL*.
The structure of the CDH is shown in fig. 4.4. Marked in yellow is the header content which is a direct copy of the L1 and L2a message data, while white signals unused or zero-filled content. Data supplied by the *SMU* is indicated in blue.

The *Format Version* reflects the version, i.e. the structure, of the CDH. In the *SMU*, the *Mini-EventID* is given by the value of the local bunch counter which is sampled at the arrival of the L1 trigger. If the trigger system and *GTU* are running synchronously, the difference of the L2a-supplied *BunchCountID* and the *Mini-EventID* is constant. The *DAQ* catches this value at the beginning of a run, checks the following incoming events and aborts the run on a change of value.

Word	31	Message Content			0
0	Block Length[31..0]				
1	Format Version[31..24]	MBZ[23..22]	L1Message[21..14]	MBZ[13..12]	EventID1[11..0]
2	MBZ[31..24]	EventID2[23..0]			
3	Block Attr.[31..24]	Participating Subdetectors[23..0]			
4	MBZ[31..28]	Status and Error field[27..12]	Mini-EventID[11..0]		
5	Trigger Classes low[31..0]				
6	RoI low[31..28]	MBZ[27..18]	Trigger Classes high[17..0]		
7	RoI high[31..0]				

Figure 4.4: The structure of the Common Data Header as specified in [DJVdV07]. A CDH is prepended to every sub-event sent from a *GTU* segment to the *DAQ*. Fields marked in blue provide local event identification and error status and are supplied by the *SMU*.

The *Status and Error* field holds information on the actual condition of the sub-event, especially on errors related to the trigger system or transmission, but also on the general state of the front-end electronics.

[9]Software triggers: the software class bits *L1Sw* and *L2Sw* are set.

SoR and EoR Trigger Sequences At the start and end of a run, the Experiment Control System executes a Start-of-Run (SoR) and End-of-Run (EoR) sequence. In the course of each of those sequences the trigger system is requested to generate and send a software trigger once all participating sub-detectors are ready to receive.
Set software class bits *L1Sw* and *L2Sw* and *RoC*[3..0] values of 0xE and 0xF, for SoR and EoR, respectively, allow the distinction from other trigger sequences. These special sequences do not trigger the readout of an event from the *TMU*s and its transmission to the *DAQ*, but require the sending of so-called SoD and EoD events. These consist of a CDH only, with software class and readout bits set according to the trigger messages.
When a SoD event is received at the *DAQ*, all readout programs involved in the data flow become ready to take physics event data. The EoD event serves as marker that indicates the end of the data stream for the current run. Upon reception, the readout programs start their end-of-run procedure.

4.3 Trigger Reception at the *GTU*

4.3.1 Transmission from *DCS* Board to *SMU*

The fibers leaving the *TTC*oc establish the unidirectional connection between the 18 Supermodule Units (and the Trigger Unit) and the trigger system. On each *DCS* board, a fiber plugs into a jack hosting a photodiode, from where a differential pair is routed to the input of the onboard *TTC*rx ASIC.

***TTC*rx** Main purpose of the *TTC*rx is the reconstruction of the *LHC* clock and the decoding of the 80 Mbit/s *TTC* data stream, which is separated in two channels A and B. The output *L1Accept* is driven directly by the *TTC* signal of channel A and is, like all signals described in this paragraph, routed to the FPGA on the *DCS* board. Data transmitted via channel B is decoded, hamming–checked and corrected if necessary. The *TTC*rx provides a parallel interface which is accessible by the *DCS* FPGA. Corresponding to fig. 4.2(b), the data interface provides the *subaddr*[7..0], *data*[7..0] and the appropriate strobes, whereas the broadcast interface provides *brc_data*[7..0][10] plus strobe. The *TTC*rx also publishes the values of its internal bunch and event counters by *b_cnt*[11..0]. Three strobes mark the bunch count and the low and high half of the event counter.
Furthermore, it is possible to receive the raw *TTC* channel B data stream on the *serBchan* output pin by enabling the corresponding control register. The control register can be written via I^{2}C from the *DCS* board or by sending an IAC with $E = 0$ to access the inter-

[10]The two least significant bits of a broadcast command are reserved for bunch and event counter reset.

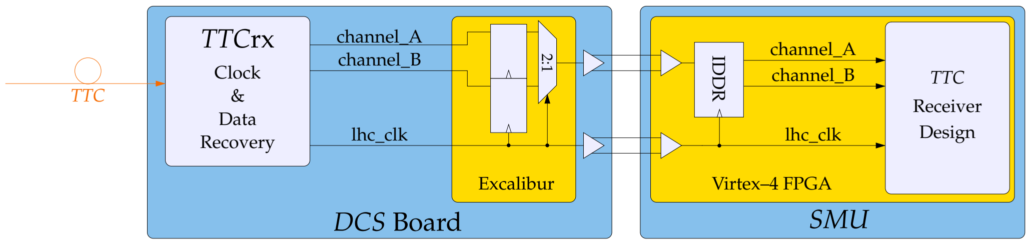


Figure 4.5: Transmission of the trigger signals *lhc_clk*, *channel_A* and *channel_B* inside the *GTU*.

nal address space. The latter is done during the initialization sequence of the *LTU*[11]. This method provides a serial interface to channel B.

Transmission of *TTC* Data to the *SMU* Due to the very limited number of interconnections between the *DCS* board FPGA and *SMU* it is not possible to access the *TTC*rx outputs through the parallel interface directly. Instead, some sort of serial transmission protocol is required. Since access to the channels is already given in serial form, it is more suitable to omit the decoding and correction capabilities of the *TTC*rx and transmit raw *TTC* data to the *SMU* directly.

Decoding the *TTC* signal in the *SMU* has several advantages. Checking the Hamming code and channel A signals in the *SMU* allows close monitoring of the transmission and possible errors can be reported to the *DAQ*. Furthermore, only minor modifications of the already crowded *DCS* FPGA design are necessary.
Fig. 4.5 illustrates the distribution of the trigger signals from *DCS* board to the *SMU*. In order to profit from the noise resilience of LVDS, clock and serial data are transmitted on differential lines. Since only two line pairs are available in downlink[12] direction, the serial *TTC* signals, *channel_A* and *channel_B*, are time-division multiplexed to allow the transmission on one line. Contrary to the Virtex-4, the Excalibur FPGA does not feature dedicated ODDR[13] primitives which could be utilized for this purpose. Therefore, a DDR stage is implemented with two registers and a 2:1 multiplexer, switching the data lines with the falling and rising edge of *lhc_clk*. Constraining the location to the Logic Array Block adjacent to the output pin ensures proper signal timing.
On the *SMU* side an IDDR primitive is used for reception. Its outputs, *channel_A* and *channel_B*, are input to the *TTC* Receiver Unit which rebuilds and extends the functionality of the *TTC*rx.

[11]During development it emerged that a not properly configured *TTC*rx is a common source of error. Therefore, the initialization sequence of the *LTU* should always be triggered after power-up of a *DCS* board. Alternatively, the control register could be set via I^2C at startup of the *DCS* board.
[12]Downlink: from *DCS* board to *SMU*
[13]ODDR: Output DDR register stages

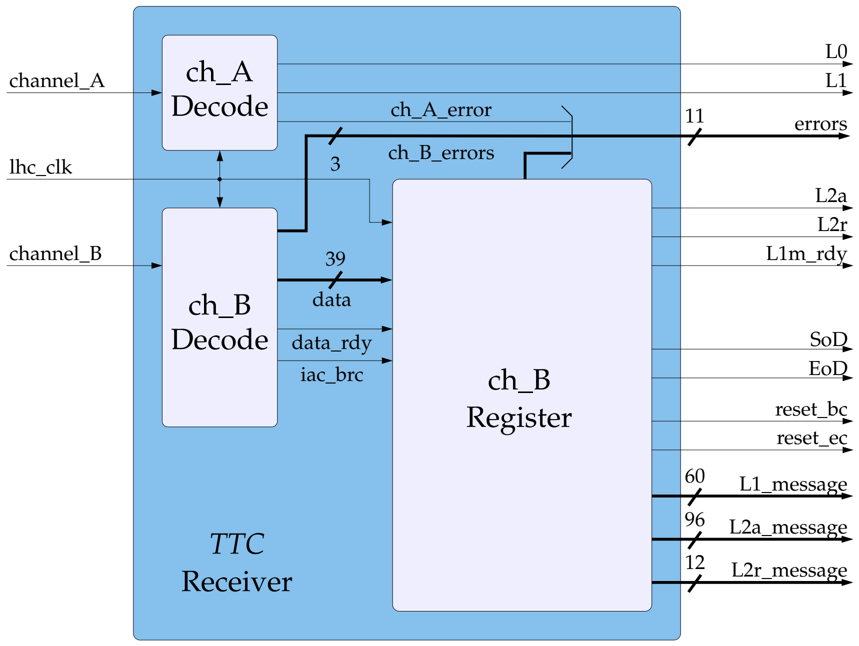


Figure 4.6: The interface to the *TTC* Receiver Unit on the *SMU* and its building blocks.

4.3.2 *TTC* Receiver Unit

The interface and building blocks of the *TTC* Receiver Unit is shown in fig. 4.6. It is based on the design by J. Alme ([Alm05]) and has been adapted and extended to meet the requirements of the *GTU*.
The decoding of channel A, which gives the signals for the L0 and L1 trigger, induces a latency of one bunch count cycle to the signal. However, since tracklets and raw data from the Supermodule arrive at the *GTU* only several microseconds after the triggers, this is not critical for this purpose. The unit also monitors the transmission of any unexpected data, which would give a *ch_A_error*.

The state machine of the ch_B Decode block, which receives and checks the data on channel B, is shown in fig 4.7. A start bit (zero) indicates the start of a message and moves the state machine from the *idle* to the *fmt*-state. The format bit indicates whether the type of message, which is shifted in in the *get_brc* or *get_iac*-states, is a broadcast or individually addressed message. Once the expected number of bits is received, the state machine enters the stop state, in which the Hamming code is checked. If necessary and possible, the data is corrected and its readiness signaled to the register unit. If either a recoverable single-bit error or an uncorrectable double-bit error is detected, the *error* state is entered. Here, flags are set to indicate a channel B error and the exact type of the error. Otherwise, the machine returns to *idle* and is ready for the next data transmission.

The ch_B Register block accepts data from the decode block and evaluates its content. It receives *data*[39..0] of the full length of an IAC message, excluding *start*, *fmt* and *stop* bit. *Iac_brc* indicates the message type, while *data_rdy* serves as strobe.

For an IAC message several checks are performed. The header of the first word gives information on the type of the message and how many data words to expect before the next header. E.g., $header = 0x3$ marks the first word of a L2a message, and exactly seven subsequent words with $header = 0x4$ should follow. If any other header is encountered, a L2a message error is reported. Analogous monitoring is performed for the L1 message. Arrival of an unknown header is signaled by setting an error flag, as are inconsistencies in the message content (e.g. $\mathrm{CIT}_{\mathrm{L1}} \neq \mathrm{CIT}_{\mathrm{L2a}}$).
The message strobes (*L1m_rdy*, *L2a* and *L2r*) indicate the successful arrival of the complete trigger message (*L1_message*, *L2a_message* and *L2r_message*). In the case of an L2a or L2r, they additionally serve as trigger pulse, which is forwarded to the control unit.
SoD and *EoD* signalize the reception of a SoR or EoR trigger sequence (L0, L1, L2a) one cycle after the corresponding *L2a*.

For local event identification, the *SMU* incorporates a bunch counter, which is incremented with *lhc_clk* and synchronized to the one of the trigger system by *reset_bc*. The reset for the local bunch counter and a local orbit counter, *reset_bc* and *reset_ec*, originate from the two least significant bits of a broadcast command.

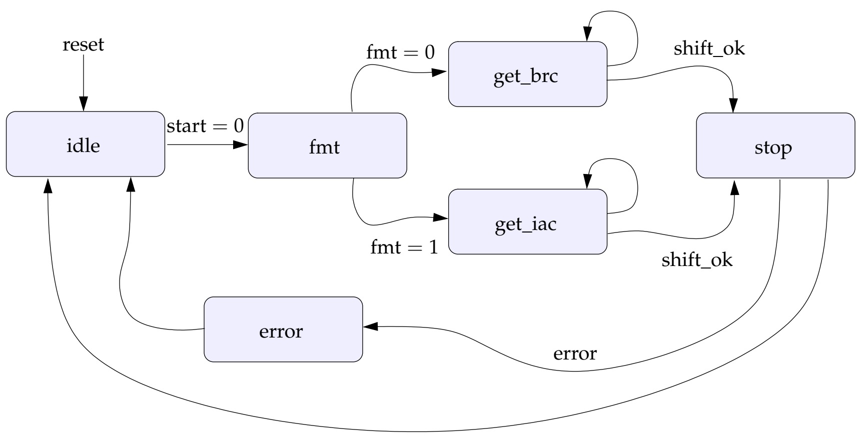


Figure 4.7: The finite state machine, which receives the *TTC* messages pictured in fig. 4.2.

5 Operating a *GTU* segment

In its function as control and concentrator board for a *GTU* segment, the Supermodule Unit operates itself and the five Track Matching Units according to the trigger sequences. It interfaces the *TTC* system for reception of trigger data. Via a mounted *SIU* card, the *SMU* establishes the link the *DAQ*, to which either event raw data or experiment control and status information is transmitted. The *TGU* interface is used to announce the *busy* state and L1 contribution of the segment to the Trigger Unit with minimal latency for subsequent forwarding to the *CTP*. A control and monitoring interface is implemented utilizing one of the local PowerPC cores, which can be accessed from anywhere in the *TRD* network through the *DCS* board.
As hinted by the variety of interfaces, the tasks the *SMU* has to fulfill are manifold. This chapter summarizes the requirements and gives an overview of the layout of the *SMU* design. The implemented unit which monitors the *TTC* transfers and generates the control signals is discussed in detail. At the end of the chapter, a possible implementation for a multi-event buffering capable design is sketched.

5.1 Design Requirements

The *SMU* administers the processing of tracklets, the recording and storing of event data and potentially initiates and controls the transmission of an event fragment[1] to the *DAQ*. In order to do so, the following requirements must be met:

- Error detection on arriving trigger signals:
  On level of the Track Matching Units, events, which are stored in the event buffer upon a L1, cannot be tagged with the *eventID* contained in the associated trigger sequence, since the *TRD* does not provide any form of event ID. The only way to identify the events in the *TMU*'s buffer is the chronological order in which they are stored. In such a scenario an arbitrarily formed error in the trigger sequence or its transmission can have great implications to the system. E.g., the consequences of an unrecognized L2 reject would be a displacement between event and associated *eventID* by one for the rest of the run.
  It is therefore essential to monitor the incoming trigger sequences and check their

[1]Data sent by a *SMU* via the *DDL* to the *DAQ* is correctly referred to as event fragment. However, on *GTU* level, event fragments might also be called event data or just event.

integrity. Errors, caused either by the *CTP* itself, by the transmission via the *TTC* system or during the reception by the *SMU*, have to be identified and reported and proper clean-up actions must be taken. In order to trace errors and speed up debugging, appropriate monitoring resources must be implemented.

- Issue control signals to the *TMU*s:
  Depending on the state of the currently active trigger sequence, the *SMU* must issue corresponding control signals to the *TMU*s. As introduced in section 3.3, the *TMU*'s event buffering design is interfaced by the signals *data_capture*, *event_flush*, *event_accept* and *event_reject*. *Expect_tracklets* signals the track matching entity that incoming tracklet data must be expected.

- Readout of event data:
  The *SMU* must administer event data readout for the segment. From the data transmitted by the *TMU*s and the trigger system, an event fragment has to be formed that complies with the requirements for transfer via the *DDL*. To all blocks of data sent via the *DDL*, a Common Data Header which contains the minimum information to identify the data in the *DAQ* system has to be prepended. The CDH (see section 4.2) is assembled from data transmitted by the trigger system and *GTU* segment-specific information.

- Support of SoD and EoD events:
  The sending of SoD and EoD event fragments to the *DAQ* is requested by a software trigger during the SoR and EoR sequences and must be supported (see section 4.2).

- Multi-event buffering:
  After an L1, the *GTU* accepts and stores event data transmitted by the *TRD*. In order to minimize the *TRD* dead time, the *GTU* is designed to buffer data from several events prior to their readout in case of an L2 accept. Thus, the operation with multiple interlaced trigger sequences (see fig. 2.5) must be supported.

- Transport and process track data with minimal latency:
  The track data, sent after the track matching for the detector stack has been completed, must be collected from all *TMU*s and forwarded to the *TGU*. Due to the tight timing requirements for the L1 contribution, it is essential to ensure minimal latency.

- Support for 'simple' trigger schemes:
  Since the trigger system is not always available during the development, production and commissioning phases, it must be possible to trigger event readout in a simple manner (e.g. by an externally provided pulse) and send events with valid data header to the *DAQ*.

5.2 Trigger Errors

Of the sub-events sent from the *GTU* to the *DAQ*, those with matching ID fields in the Common Data Header are selected and combined to build an event by the Global Data Concentrators. Since part of the event identification is supplied by the trigger messages, it must be guaranteed that CDH and corresponding event data are assembled correctly in order to retain data consistency. It is therefore mandatory to monitor the input from the trigger system to detect and report all errors.

For a normal sequence (see fig. 5.2), an L1 is expected to arrive in a time-frame of very few clock cycles, $\Delta t_{\text{L1\_Window}}$, centered around the fixed time $t_{\text{L0,L1}}$ after the L0 trigger. The presence or absence of an L1 in this window is interpreted as L1 accept or L1 reject, respectively, and the current trigger sequence is aborted or continued.
If an L1 is received, the arrival of the corresponding L2 message is expected in a time-frame of $\Delta t_{\text{L2\_window}}$, starting at $t_{\text{L1,L2\_low}}$.

Fig. 5.2 gives an overview of the activities of the *GTU* during a normal trigger sequence. Concerning the event buffering design, the current event can be flushed as long as it is not completely transmitted by the Supermodule. After that, the event is accessible through the *event_reject* and *event_accept* signals. With several events in the buffer which are ordered chronologically only the oldest event can be accept or rejected.

Errors can be divided in two groups: Those that can be detected after an L1 and those which are not detectable until the arrival of an L2 or an L2 timeout. If detected directly after an L1, the reception and storing of event data in the *TMU* is not completed and can be aborted by an *event_flush*[2]. The allocated buffer partition is cleared and immediately available for the next event.
Errors concerning missing or spurious L2 triggers are more critical, since event data might already have been fully recorded and stored in the event buffer and is accessible in chronological order, only.

An illustration of errors in trigger sequences which are related to false timing or missing triggers is found in fig. 5.2.

- L0 overlap error:
  If an L0 was issued, no other L0 may be received until $t_{\text{L0,L1}} + \Delta t_{\text{L1\_Window}}/2$.
  In the case of an L0, the possible arrival of tracklet data is signaled to the *SMU* and the event buffering designs start to accept data if the *get_tracklet* option is enabled. Therefore, a detected L0 overlap error requires a reset of the track matching units and a flush signal to the *TMU*.

[2]Latest measurements (30 time bins, no compression) with the test setup described in section 6.4 give approx. 240 µs for the time from an L1 to the last data word arriving from the Supermodule at the *TMU*, upon which the storage of the event in the event buffer is completed. This gives sufficient time for the proper handling of errors.

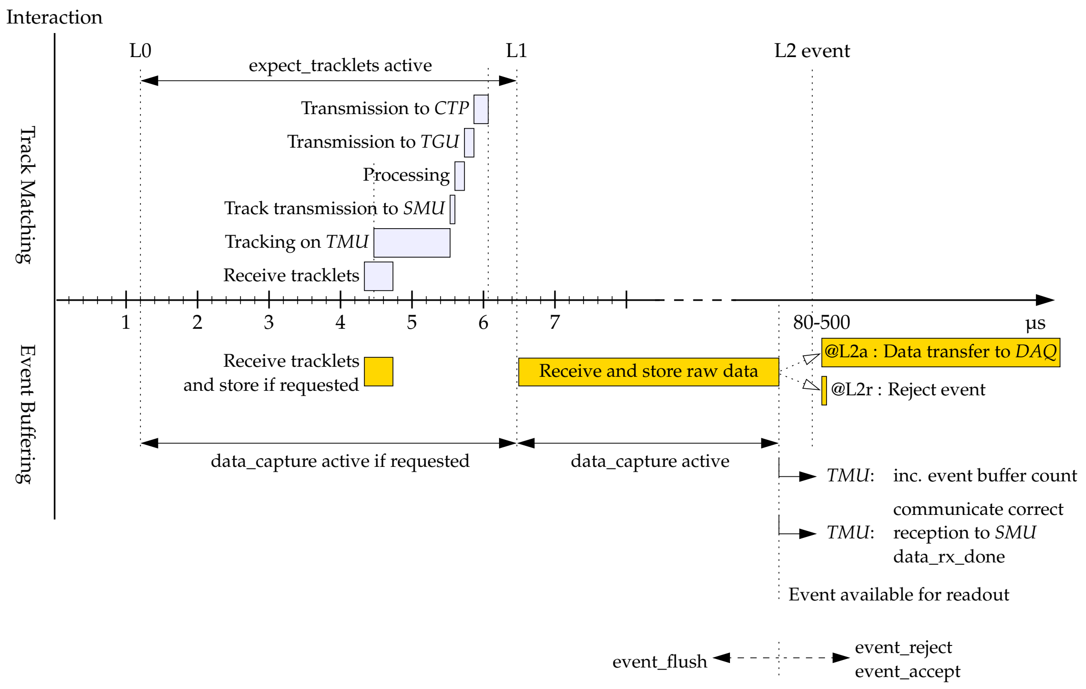

Interaction
L0
L1
L2 event
expect_tracklets active
Transmission to CTP
Transmission to TGU
Processing
Track transmission to SMU
Tracking on TMU
Receive tracklets
Track Matching
Event Buffering
1
2
3
4
5
6
7
80-500
µs
@L2a : Data transfer to DAQ
@L2r : Reject event
Receive tracklets and store if requested
Receive and store raw data
data_capture active if requested
data_capture active
TMU: inc. event buffer count
TMU: communicate correct reception to SMU data_rx_done
Event available for readout
event_flush
event_reject
event_accept

Figure 5.1: Activities of the *GTU* during a normal trigger sequence. The upper half addresses track(-let) data, which is received and processed between an L0 and L1 trigger, whereas the lower half deals with event buffering. As soon as the event is completely shipped by the Supermodule, the event buffer counter is increased and the new event is announced to the *SMU* by the *data_rx_done*-signal.

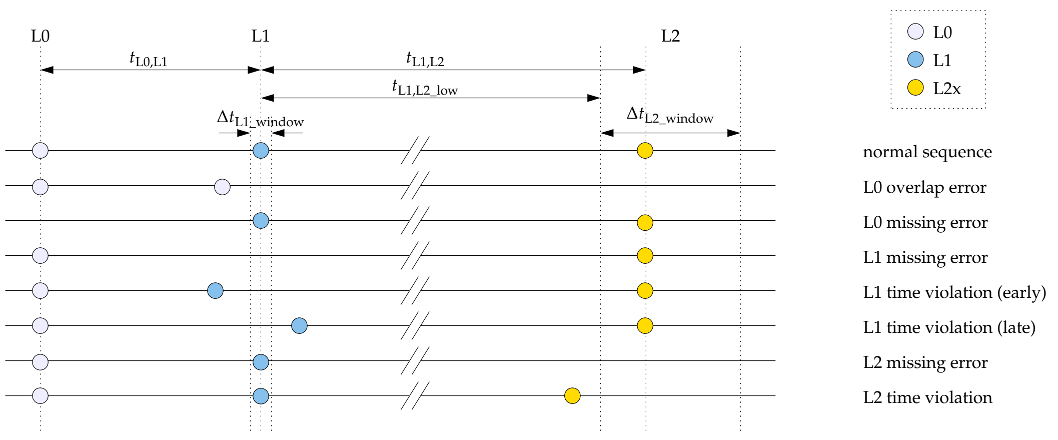


Figure 5.2: Listing of trigger errors due to missing triggers or wrong timing

- L0 missing error:
  A missing L0 trigger would cause the *TRD* Supermodules to abort the tracklet processing, which was started with the pretrigger, and return to the idle state. The error can be detected on the *SMU* before commands to the *TMU*s are issued.
- L1 time violation error:
  An L1 time violation is the reception of an L1 trigger outside the L1 valid window. In any case, the corresponding L2 trigger seems spurious, since the sequence started with the L0 was aborted. An *event_flush* to the *TMU*s clears the currently processed event.
- L1 missing error:
  The L1 missing error is difficult to detect. Since the *ALICE* trigger system does not incorporate an explicit L1 reject signal, the sequence appears to be a normal sequence in which the *CTP* has decided not to issue an L1 trigger. Assuming an aborted trigger sequence, the arrival of the L2 seems to be a spurious trigger. Only by comparison of the event ID in the L2 message against the local bunch count value at the L0 arrival[3], this error can be identified.
- L2 missing error:
  A valid trigger sequence that has passed the stage of the L1 trigger must always be concluded by an L2 event. If no L2a or L2r is received in the L2 valid window between 80 µs and 500 µs after the corresponding L1, an L2 missing error must be reported and the corresponding event in the *TMU* buffer discarded.
- L2 time violation error:
  Transmission of an L2 trigger outside the L2 valid window is a time violation error. The corresponding event must be discarded and the error reported.

5.3 Implementation

5.3.1 *SMU* Layout

For reasons of maintainability and to provide maximum flexibility, the design is divided into several functional blocks, which are shown in fig. 5.3. The *TTC* receiver unit already presented in section 4.3.2 handles the reception of trigger sequences. Decoded triggers and trigger messages are passed to the control and monitor unit, where the message content is stored and processed.

The control and monitor unit analyzes the *TTC* transmission for errors and stores information about its status in the status registers. Together with the local bunch count and the

[3]Identification of the L1 missing trigger:
$\mathrm{EventID}_{\mathrm{L2}} \in \left[\mathrm{BC}_{\mathrm{L1,low}}, \mathrm{BC}_{\mathrm{L1,high}}\right]$ with $\mathrm{BC}_{\mathrm{L1,low/L1,high}} = \mathrm{BC}_{\mathrm{L0}} + t_{\mathrm{L0,L1}} \cdot f_{lhc_c lk} + \mathrm{Diff} \pm \Delta t_{\mathrm{L1\_Window}}/2$.
Diff is the constant difference between the local bunch counter and the *CTP* bunch counter, due to the latency of the *bc_reset* signal.

trigger message data, its content is used to assemble the CDH. If no *TTC* data is available during standalone tests, a valid CDH can be generated internally and prepended to the event fragment. Except for *event*_accept, the control unit and monitor unit also generates the control signals for the *TMU* boards according to the state of the trigger sequence.

The readout unit is capable of independently managing the event readout and the transfer to the *DAQ*. Once started, it causes the *TMU*s to send the requested event and supervises the transmission to the *DAQ* subject to the reverse flow control originating at the *SIU*.

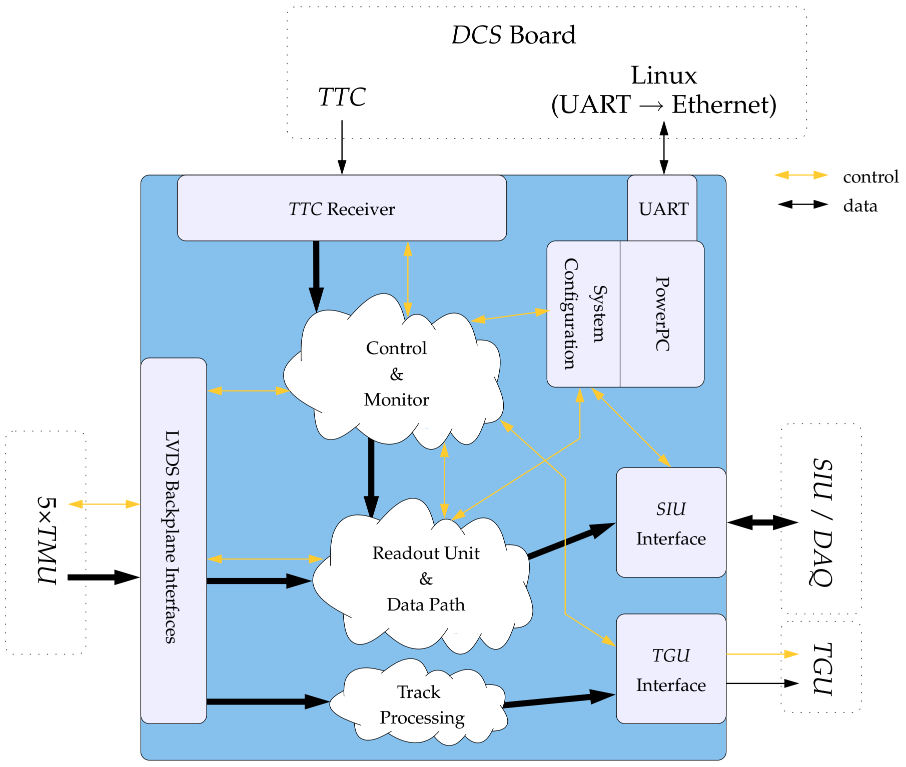


Figure 5.3: The functional blocks of the *SMU* hardware design and a survey of the interfaces

The readout unit is designed to allow the starting of a readout cycle with a single pulse, which can be provided externally or issued by the control unit or by PowerPC. The readout unit and the path of event and track data through the *SMU* are discussed in depth in chapter 6.

To comply with the demands of the *TRD* group for a functional *GTU* segment for Supermodule testing and to gain experience with the behavior of the *TMU* designs, development was focused on a design that supports recording and readout of a single event for the time being. The design is upgradable to support multi-event buffering design with partial modifications in the control and monitoring unit.

5.3.2 Layout of the Control and Monitoring Unit

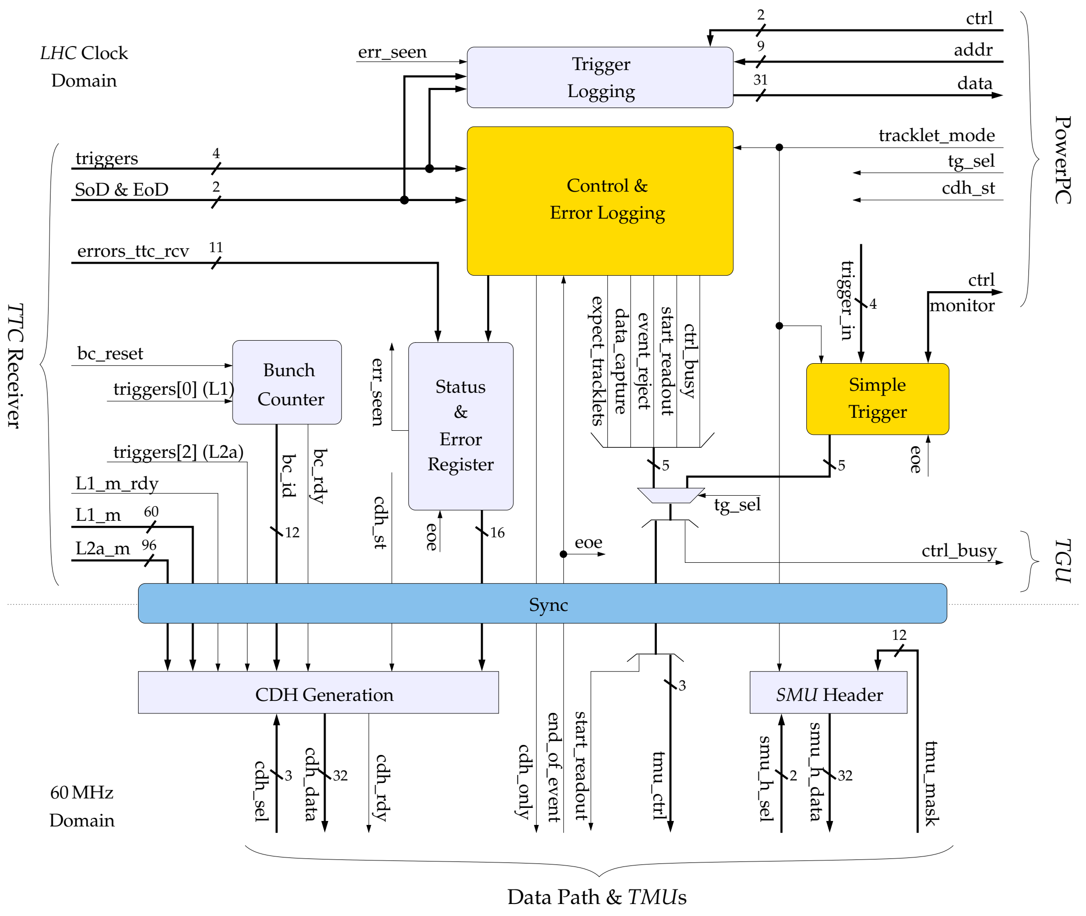


Figure 5.4: The control and monitoring unit can either operate with *CTP* provided triggers or 'simple triggers', enabling the *GTU* to be used for Supermodule testing without *TTC* system.

The unit receives trigger signals and generates appropriate control signals for the *TMU*s and the data path. If running with the *ALICE* trigger system, the incoming trigger sequences are monitored and errors are reported to the *DAQ*. Furthermore, together with a time stamp, all incoming trigger events are logged and stored in the trigger logging unit[4]. The Common Data Header is assembled using the error information from the status register, the local bunch count identification (*bc_id*) and the trigger message content. For development and testing purposes, the *SMU* also features a 'simple trigger'-mode. In this mode, a simple trigger pulse is used to trigger the readout of an event. The trigger pulse can be provided by PowerPC, by an external LVDS signal or optically, using channel A of the *TTC* system.

[4]Trigger logging: Up to 512 events can be stored

Fig. 5.4 shows a schematic representation. The CDH's *Mini-EventID* is given by the local bunch count at the arrival of the L1 trigger. It is synchronized to the *CTP* bunch counter by the *bc_reset* signal, which is received as broadcast command via the *TTC* system. Since the *DAQ* accepts a constant difference between the *Mini-EventID* and the *EventID1*, latency in the transmission of *bc_reset* is of no relevance.

The status and error register receives and stores error information from the *TTC* receiver unit, which monitors the transmission itself for errors, and from the control unit, which checks for appropriate timing of the sequences and spurious triggers. The *err_seen*-signal announces the occurrence of a trigger error and causes the trigger logging unit to halt. Status and error flags are reset after the sending of an event fragment is complete (*eoe*). In order to ease system debugging, the unit incorporates counters for the type and quantity received triggers and trigger messages.

Logging of Incoming Triggers

The trigger logging unit tags incoming triggers and special events, such as SoD or EoD, with a time stamp and stores them in a dual-port block RAM. The time stamp, which is appended to the trigger type information, consists of the local bunch counter value at the time of the trigger arrival and the time span since the previous trigger event[5]. The latter is stored as 16 bit value, thus large enough to cover the full length of a trigger sequence[6]. If the upper limit of 512 entries is exceeded, the oldest value is overwritten. The error logging is halted if *err_seen* becomes active or if the logging is stopped by the user. Access to the logged information and control is given via PowerPC.

Shown below is the user interface and an example of a valid L2a trigger sequence received on *GTU* segment 00. *Interval* marks the time span to the previous entry in the trigger logger. *Bunch counter* displays the bunch counter value at the time of trigger arrival.

[5]Trigger event: L0, L1, L2a, L2r

[6]The trigger logger is operated with the *LHC* clock of 40.079 MHz.
The covered range of $2^{16} \cdot 2.495 \cdot 10^{-7}\text{s} = 1.635\text{ms}$ compares to the maximum length for the trigger sequence of 500 µs.

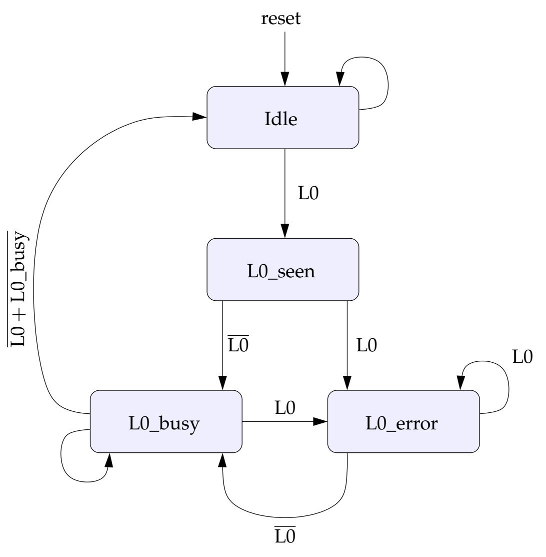
```
alidcsdcb0600:/ $ gtucom 2 help tg_log
  tg_log: Print type and time stamp of received triggers
  tg_log [ show [k] | reset | halt | continue | status ]
  Usage:
    show     : Show type and time stamp of the last received triggers.
               Up to k entries are shown (max k=512, default k=20)
    reset    : Reset and restart trigger logging
    halt     : Stop recording of triggers
    continue : Continue recording of triggers
    status   : Show current logging status

alidcsdcb0600:/ $ gtucom 2 tg_log show
     Trigger Events                                  Bunch counter    Interval
[L0]   [L1]   [L2a]   [L2r]                            [cyc d/x]         [us]
======================================================================
 L0                                                  0597 / 0x255      > 1600
       L1                                            0822 / 0x336       5.575
             L2a                                     0971 / 0x3CB      92.775
```

Error Detection

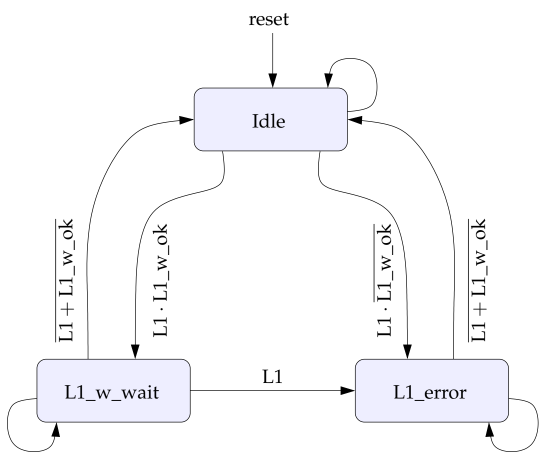
L0 Errors Detection of an erroneous L0 trigger is illustrated in fig. 5.5(a). Upon an L0, the state machine enters the *L0_seen*-state, where the start signals for the L0 busy counter and the L1 window timer are issued. The former sets the *L0_busy*-signal until the end of the L1 valid window. In a valid sequence, no other L0 is seen and the state machine proceeds to the *L0_busy*-state, waiting until *L0_busy* becomes inactive before returning to *idle*. Another L0 in this time period marks an L0 overlap error, therefore the *L0_error*-state is entered in which both counters are reset. Thus, the newly arrived L0 trigger is assumed to be correct. *L0_spurious* is set and recorded in the status register.

L1 Errors The L1 window timer produces the *L1_w_ok*-signal which defines the time-frame where the arrival of the L1 trigger is acceptable. Reception of an L1 while *L1_w_ok* is active lets the state machine, shown in fig. 5.5(b), move from the *Idle*-state to *L1_w_wait*, in which timers for the L1 and L2 message are started. In a valid sequence, the *Idle*-state is entered at the end of the valid window. If any L1 is received outside of the window or a second whilst in the *L1_w_wait*-state, the *L1_error*-state is entered. The *L1_spurious* flag is set active and the timers, which are used to monitor the correct arrival of the trigger messages, are started. Again, resetting the timer assumes the last received trigger to be correct, thus following the policy proposed in [JJ06].

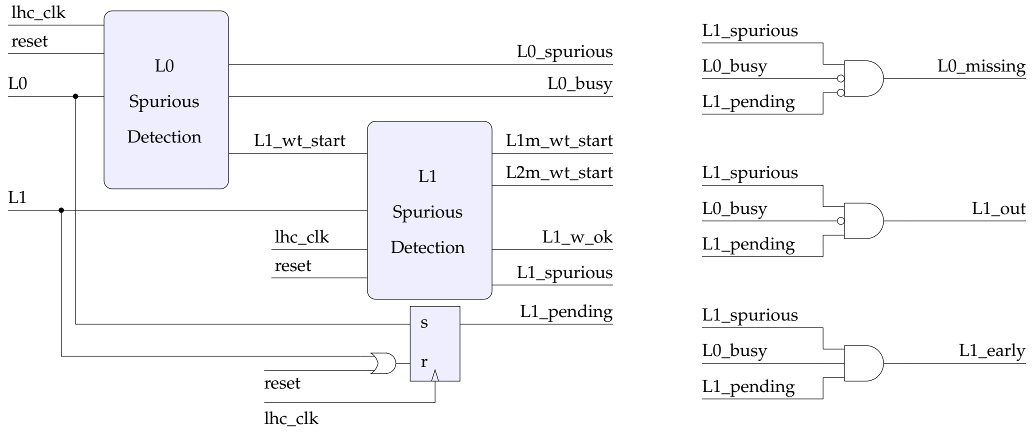
Setting *L1_pending* at an L0 allows, in combination with *L0_busy* and *L1_spurious*, to determine the errors *L0_missing*, *L1_early* and *L1_out* as shown in fig. 5.5(c). The internally



(a) L0 spurious detection


reset
Idle
$\overline{\text{L1} + \text{L1\_w\_ok}}$
$\text{L1} \cdot \text{L1\_w\_ok}$
$\text{L1} \cdot \overline{\text{L1\_w\_ok}}$
$\overline{\text{L1} + \text{L1\_w\_ok}}$
L1_w_wait
L1
L1_error

(b) L1 spurious detection



(c) L1 error detection

Figure 5.5: L0 and L1 error detection

available signals *L1_out* and *L1_early* are reported to the *DAQ* as L1 time violation error. The CDH of the event fragment and the record of the trigger logging unit provide sufficient information to clearly identify the cause of the error, which should only occur rarely in the run phase of the experiment.

As the requirements defining the timing of a valid trigger sequence might change in future, the corresponding parameters can be set at compile time. Making them adjustable by PowerPC requires only minor changes to the design. Since no second L0 trigger is allowed in $t_{\mathrm{L0,L1}}$, the error detection for L0 and L1 is feasible for multi-event buffering.

Trigger Messages When running in single-event mode, the control unit sets its *busy*-signal once a L0 trigger is received. It is active throughout the trigger sequence, thus inhibiting the *CTP* to start a new sequence.
L2_window_ok and *L1m_window_ok* are provided by the message timers and define windows in which the arrival of the trigger messages is valid. *L1m_window_ok* is set high with the L1 trigger, while the arrival of an L2 message is not valid until 80 µs after the L1. Both become inactive 500 µs after the L1. The successful reception of a trigger message is signaled by the *TTC* receiver unit with *L1m_rdy*, *L2a* or *L2r*. If these signals are received outside of the defined time-frame, the error is logged in the status register. In case of the L2 message, an L2 time violation is reported for $t_{\mathrm{arrival}} \leq 80\,\mu\mathrm{s}$, and an L2 timeout for $t_{\mathrm{arrival}} \geq 500\,\mu\mathrm{s}$.

Control Signal Generation

Operation with the Trigger System In order to control the data flow through a *GTU* segment, the *SMU* issues control signals to the *TMU*s and reports the busy status of the segment to the *TGU*, which communicates the global busy to the *CTP* based on the input from all segments. In single-event mode, a state machine, pictured in fig. 5.6, follows the trigger sequence and fulfills this task.
A received L0 trigger causes the transition from the *Idle*-state to *L1_wi*, in which *busy* is set and remains set throughout the sequence. If the recording of tracklets is requested, *data_capture* is also asserted. Once *L1_window_ok* is active, the *L1_wv*-state is entered. If the *CTP* decides not to issue a L1, the machine returns to *Idle* at the end of the L1 valid window, in which an *event_flush* clears possibly recorded tracklets.
A spurious L0 in the states *L1_wi* and *L1_wv* would cause the transition to *Tr_flush*-state, in which the command to drop the current event is sent to the *TMU*s. If tracklets have been recorded, they will be dropped. No changes occur if the *TMU*s have not taken any data. Since a spurious L0 resets the L1 timer, *L1_wi* is entered.
In the *L1_wv*-state, an incoming L1 continues the trigger sequence and moves the machine to *L2_wi*. Here it resides until the L2 window counter signals the *L2_window_ok*.

Being in the *L2_wv*-state, another L1 trigger would cause the restart of the window timers and set the state machine back to *L2_wi*.

No successful reception of an L2a or L2r message whilst in *L2_wv* makes the state machine move to *L2_timeout*-state, which initiates the sending of a CDH whose error field indicates an L2 timeout error. An *event_reject* issued to the *TMU*s clears the event from the event buffer.
In a valid sequence, an L2a or L2r is received in the L2 valid window. The latter causes the transition from *L2_wv* to *L2r_rcv*. The status of the error register determines if the state machine causes the readout unit to send a CDH to the *DAQ* indicating an error for the past trigger sequence[7]. If sending is complete or if no error has occurred, the machine returns to *Idle* after issuing an *event_reject*.
Upon reception of an L2a in the valid window, the machine enters the *Sod_EoD_dec*-state. If the received L2a message was the request for an SoD or EoD event fragment, the *TTC* receiver unit issues the corresponding *SoD* or *EoD*-signal one cycle after the L2a. They cause the transition to *SoD_EoD*, in which the sending of a CDH to the *DAQ* is initiated and an *event_reject* is issued to the *TMU*s. Once sending is completed (signaled by *eoe*), *busy* is unset and the *Idle*-state entered.
For a physics L2a, *start_readout* is signaled to the readout unit, which triggers the readout process described in chapter 6. Again, the state machine waits for the full event to be sent to the *DAQ*, thus inhibiting the generation of new trigger sequences by keeping *busy* active.

Simple Triggering For operation in the commissioning and debugging phase, where *CTP* or *LTU* might not be available, the *SMU* supports operation with 'simple trigger' pulses. The incoming pulse, interpreted as L0, triggers the generation of the necessary control signals with the correct timing.
Fig. 5.7 shows the simple trigger logic. Four trigger sources, which are selectable by PowerPC, are currently supported. *Tg_in_aux* and *tg_in_tgu* connect to the RJ-45 adapters on the LVDS backplane, while *tg_in_opt* can be used to receive an optical trigger pulse via channel A of the *TTC* system[8]. A single pulse of length $t = 25\,\mathrm{ns}$ suffices. Finally, *tg_in_ppc* allows software triggering via PowerPC.
If the unit is enabled by *tg_sel*, the timers for the L1 and L2a triggers are started after the falling edge of the incoming trigger pulse in order to emulate the behavior of the *ALICE* trigger. Between an L0 and the generated L1, which is issued after 6.2 µs, *expect_tracklets* is set active. *Tracklet_mode* determines whether to set the *data_capture*-signal active with L0 or L1. It is kept high until the event is completely read out. The readout unit is started by *start_readout* 90 µs after the L0.

[7]In case of sending a CDH only, the readout unit receives the *start_readout*-signal, and the flag *cdh_only* is set. This inhibits the sending of an *event_accept* to the *TMU*s by the MER/Dispatcher (see chapter 6.3.2) and lets the Dispatcher return to *Idle* after sending the CDH.
[8]E.g. utilizing a *TTC*vi and *TTC*ex.

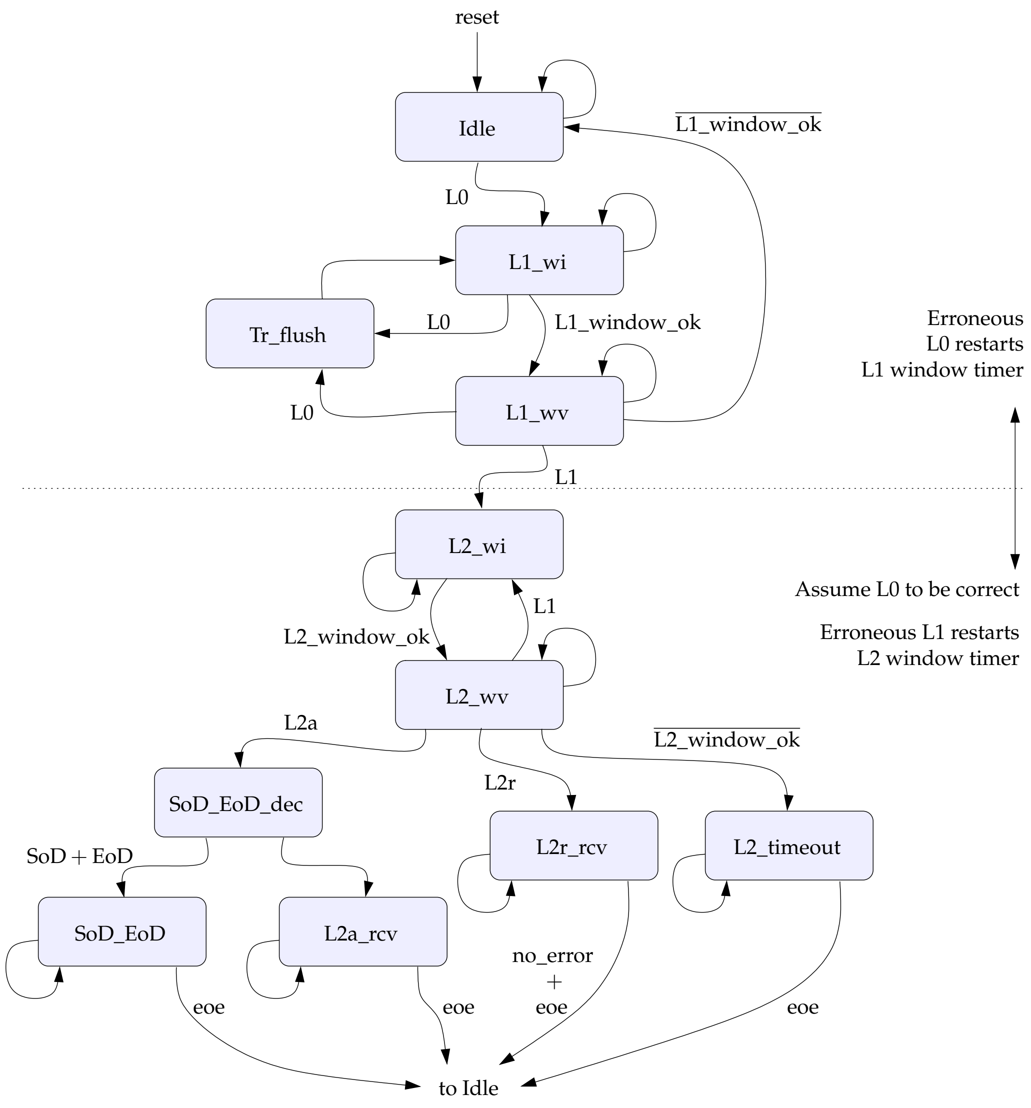


Figure 5.6: *SMU* control state machine. The *window_ok*-signals set by the L1 and L2 window timers define the valid time windows and are used to trigger state transitions.

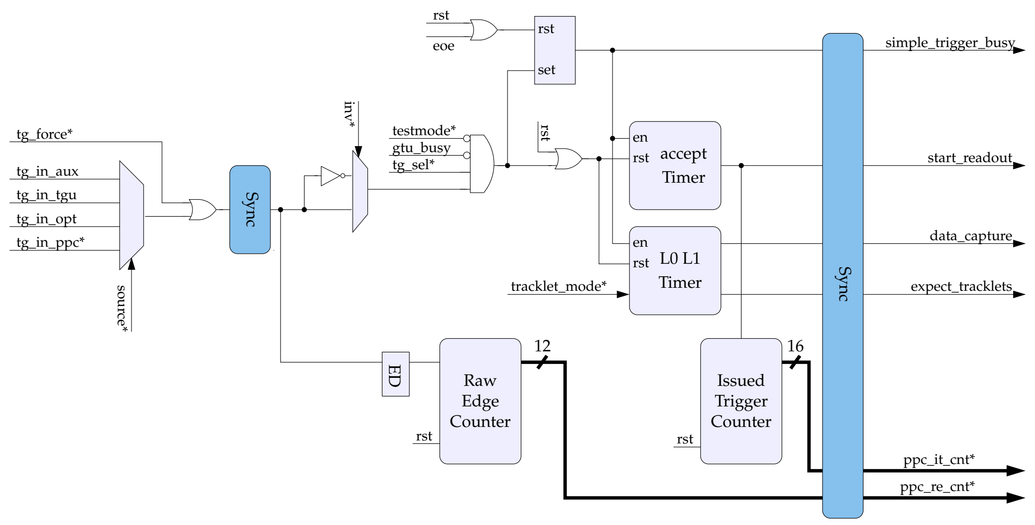


Figure 5.7: An overview of the 'simple trigger' logic. Signals marked by an asterisk can be controlled/monitored via PowerPC.

***Busy* Generation** The *busy* state of the *GTU* segment is reported to the *TGU*, which combines the information from all segments and communicates the *GTU* busy to the *CTP*. Fig. 5.8 illustrates the busy generation for one segment.
Tg_sel selects the *busy* from either the control state machine or the simple trigger unit. The contribution from each *TMU* is gated with the *TMU* active mask, which corresponds to the currently active boards, and combined to *tmu_busy*. The *ppc_hold_busy*-signal is set by the PowerPC during start-up, guaranteeing no spurious triggers during initialization. *Siu_channel_active* reflects the *DDL* state. As long as the *DAQ* is not ready to receive data, the *GTU* inhibits the sending of triggers.
Similar to the 'simple trigger' unit, an PowerPC-controllable inverter stage gives maximum flexibility in order to adapt to the given test environment.

5.4 Layout for Multi-Event Buffering

To increase the probability of finding interesting events, it is essential to minimize the dead time of the trigger detectors. The *GTU* is designed to buffer data from several[9] events, which are each sent by the Supermodules in the time following the L1 trigger.

As hinted by fig. 2.5, trigger sequences can be interleaved, when operating in multi-event mode. Contrary to the single-event mode, where the sequences are of the form L0-L1-L2x, sequences like L0-L1-L0′-L1′-L0″-L1″-L2x-L2x′-L2x″ are now possible. While an L0 must always be followed by its corresponding L1 (or none, in the case of no L1 issued), thus

[9]The buffers on the *TMU*s provide space for the data of five uncompressed events.

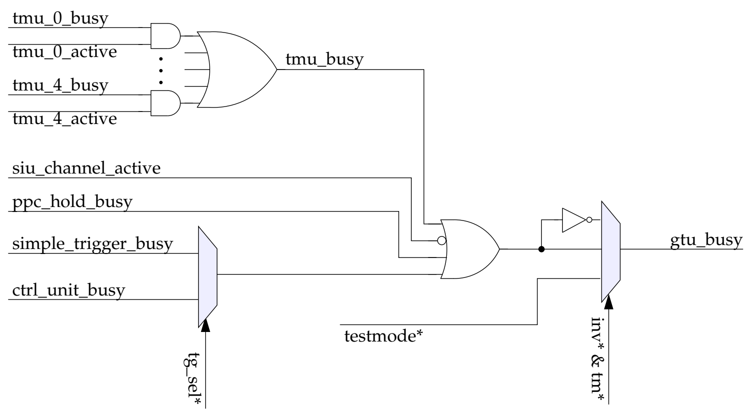


Figure 5.8: *Busy* generation for one *GTU* segment. *Testmode* provides capabilities to test the trigger/busy setup.

effectively inhibiting the start of a new sequence for $t_{\mathrm{L0,L1}} + \Delta t_{\mathrm{L1,Window}}/2$, it is valid to issue a new L0 already in the long gap before the arrival of the L2x. This, together with the fact that sequences might be arbitrarily afflicted with errors, makes controlling the *GTU* segment and the detection and reporting of errors a very complex task.

The sketch for a possible implementation for a multi-event buffering capable control and monitoring unit of the *GTU* is shown in fig. 5.9. It is assumed that the incoming trigger signal *L1_checked* is free of errors. As explained earlier, trigger errors on level of L0 and L1 have implications on the track matching entities and the event shaper units. Upon such a trigger error or on abortion of the trigger sequence, data integrity can be preserved by issuing an *event_flush* and a reset to the entities involved in track(-let) processing. An error detection entity similar to the one pictured in fig. 5.5 can be used to produce *L1_checked* by detecting the errors L0 missing, L0 overlap, L1 early and L1 spurious, and suppressing erroneous signals. Errors are, together with the current error state of the *TTC* system, logged in the *L0_L1 Error Register* and will be included in the corresponding CDH.

L1 Trigger and L1 Message Arrival Arrival of an error checked L1 initiates the capturing of the current bunch and event[10] counter values in the *L0_L1 Buffer*. All buffers are layouted as ring buffer-like structures, dimensioned to be larger than the maximum number of events storable in the *TMU* event buffers. *L1_wr_addr* is the buffer's write pointer and is incremented with every valid L1. In the *L1_L2 Timers* block, timers for the L1 and L2 message windows are started. Each entry in the *L0_L1 Buffer* is associated to one timer for the L1 and one for the L2 message. Finally, the current status of the *L0_L1 Error Register* is stored in the *L0_L1 Error Status* buffer upon a *L1_checked*. With the event

[10]Occasionally the event counter is also referred to as orbit counter.

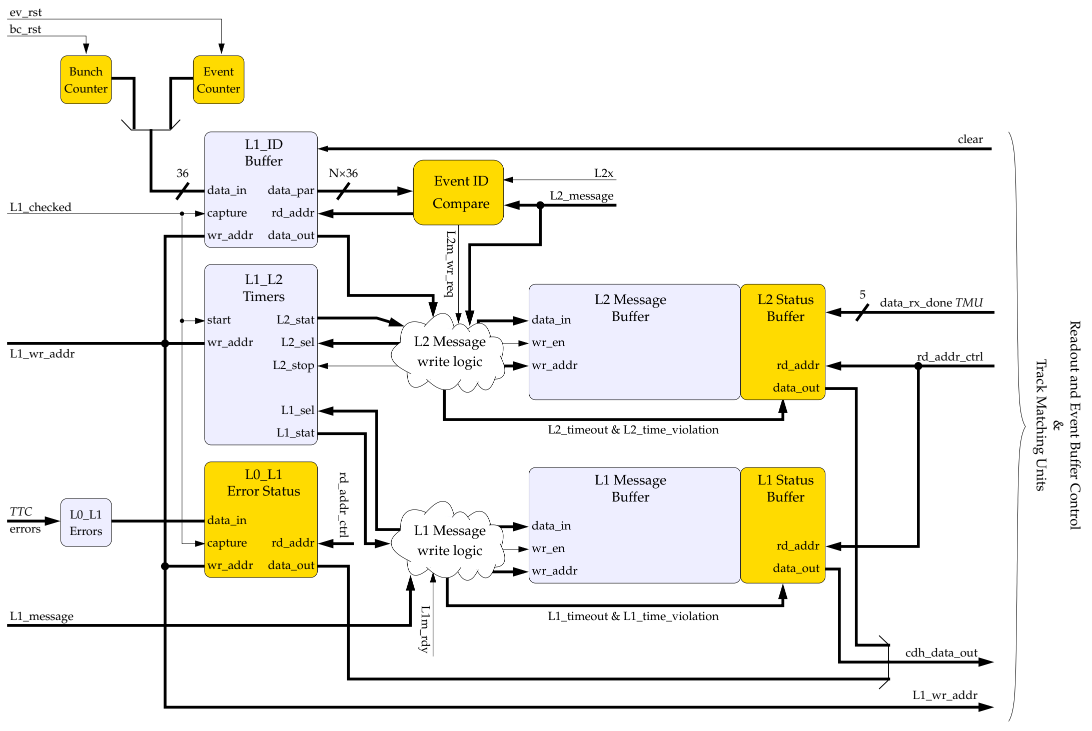


Figure 5.9: Multi-Event Buffering capable design.

identification and the error status, all information currently available for the trigger sequence is accessible under the same read address in the *L0_L1 Buffer* and the error status buffer. Having stored this information, the unit is ready to accept the next L1.

In a normal sequence, the L1 and L2x trigger messages follow in chronological order. Since the L1 message carries no identification, it is accepted and written to the *L1 Message Buffer* as long as it arrives in the valid window of the sequence currently under investigation. A survey of the L1 timers and the write logic for the L1 message is shown in fig. 5.10. The window timers produce a window valid, *w_ok*, and a *timeout*-signal. If the message arrives in the valid window of the active L1 sequence (determined by *L1_sel*), the message is stored in the *L1 Message Buffer*. If no message arrives inside the valid window, the logic detects the message timeout and marks the entry invalid by setting the *L1m_timeout* bit in the *L1 Status Register* before it advances to the next sequence by incrementing *L1_sel*.

L2x Arrival The successful arrival of an L2x message is signaled by an L2x trigger, generated by the *TTC* receiver unit. Arriving chronologically, the event identification content of the message must be compared against all entries of the *L0_L1 Buffer* in order to attribute the message content to a certain trigger sequence and detect missing sequences. Due to the uncertainty in the L1 arrival time, comparison to one buffer entry implies a

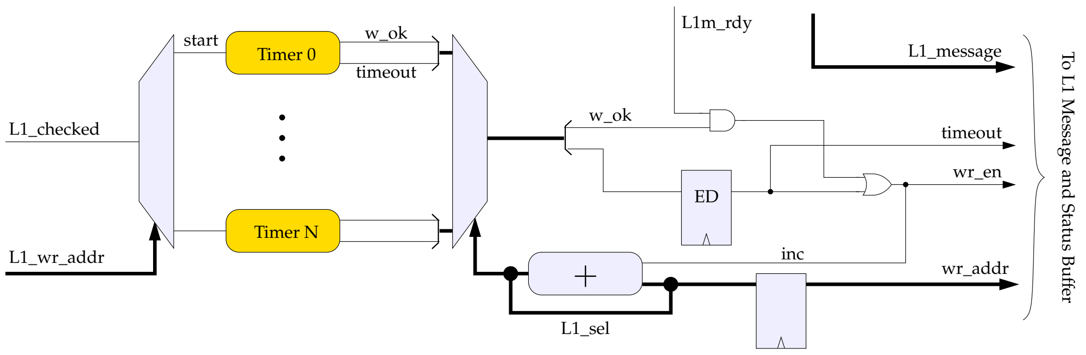


Figure 5.10: Window timers monitoring of the arrival of the L1 message and the write logic. The window valid signal, *w_ok*, is active from 0 to 500 µs after the L1 trigger. The message is either stored in the buffer, or a timeout error is set in the *L1 Status Buffer*.

comparison against all values in the range $BCID_{\mathrm{L1}} \pm {}^{1}\!/_{2} \cdot t_{\mathrm{L1,Window}} \cdot f_{\mathrm{lhc\_clk}}$. Plenty of resources in the *SMU* allow the comparison to be implemented either subsequently or in parallel. In the former case, the delay induced by the comparison must be considered when evaluating the L2 timer signals, *L2_w_ok* and *L2_timeout*.

The comparison is sketched in fig. 5.11 in parallel form. The entries concerning the identification of the event, BCID_{L2} & $\mathrm{EventID}_{L2}$, of the arriving L2 message are compared to the values, which were captured on the local bunch and event counter at time of a L1 arrival. The use of bunch and event count uniquely identifies an event on a time scale of approx. 24 minutes. Thus, the comparison yields *cmp_result* which masks the entries of the *L0_L1 Buffer* and has exactly one bit set for a valid trigger sequence, thus indicating a comparison hit. A write request signal *L2m_wr_req* and a *L2m_cmp_ok* inform the L2 buffer write logic of an L2 event and the state of the comparison. *Cmp_result* can be translated to give the write address *L2m_wr_addr* of the active trigger sequence. $BCID_{\mathrm{L1}}$ and the L2 message content are combined and stored in the *L2 Message Buffer*, if the evaluation of the L2 valid window is successful. Concerning valid trigger sequences and *SMU* status, all data necessary to build the CDH is gathered. This is indicated by the *smu_ready* status bit in the *L2 Status Buffer*.
In case of a missing L1 trigger or a spurious L2 trigger, the comparison yields no hit. The L2 message is dropped and the error is marked in the *L2 Status Buffer* entry, that is scheduled for readout next. If more than one hit is found, a serious error has occurred. To identify the source of error, the contents of the trigger logger and the events, which are all marked as erroneous, have to be analyzed.

The write logic for the *L2 Message Buffer* checks, if the L2 message window is valid. If so, the data is stored at position *L2m_wr_addr* in the *L2 Message Buffer* and the corresponding L2 timer is stopped. The L2 timers have access to the associated entry in the *L2 Status Buffer*, and set the *L2_timeout* flag if the valid window has passed. In case of an L2 timeout

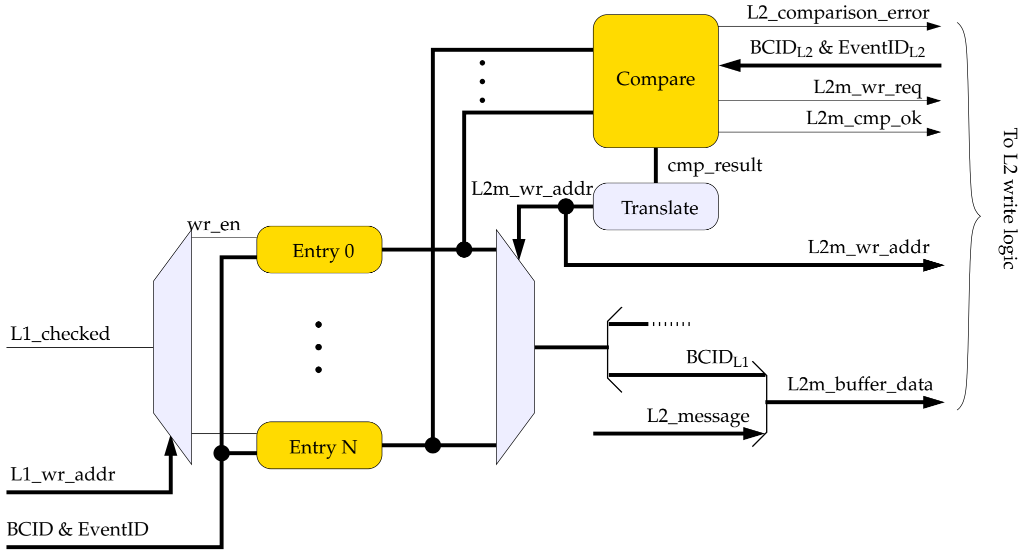


Figure 5.11: The comparison of locally stored and global event identification. For a valid trigger sequence, exactly one bit of *cmp_result* is active, indicating the *L0_L1 Buffer* entry the arrived L2x is associated with.

error, where one message got missing and a chronological order is no longer given, one entry in the message buffer is missed out. The timer responsible for the missed out entry sets the *L2_timeout* flag.

With setting active the *data_capture*-signal, the *TMU*s guarantee to add one event to the event buffers if no *event_flush* is issued. The successful adding of an event to the event buffer, communicated by *data_rx_done*, is noted in the *L2 Status Buffer* as *tmu_ready*. With *tmu_ready* and *smu_ready* or *L2_timeout* or *L1m_timeout* set, the event is ready to be processed by the unit controlling readout and event buffers.

Readout and Event Buffer Control The readout and event buffer control unit constantly monitors the difference of *L1_wr_addr* and *rd_addr_ctrl* in order to detect the number of events in the queue and set the *GTU busy* state. The latter is set, if the difference reaches a threshold which corresponds to the maximum number of events the event buffer can hold. A PowerPC-programmable threshold would allow simple switching between single and multi-event mode.
A non-zero difference[11] indicates that an event is scheduled for transmission to the *DAQ*. Once the event is ready for readout, the event status data, which is the combined content of the status and message buffers, is analyzed for special events (SoD or EoD), the type

[11]The problems of full/empty distinction for circular buffers is avoided by dimensioning the buffers larger than the maximum number of allowed entries.

of trigger (L2a, L2r) and the error status. The CDH header is assembled and the readout or rejection of events is initiated.

Breaking down error handling into several stages, which map to the current activities of the *GTU* greatly reduces complexity. The readout and event buffer control has available a complete record[12] of the event, which also reflects the state of the event buffers and trigger reception. Thus assembling the CDH and issuing control signals to the *TMU* is trivial.
It might be conceivable, especially in the commissioning phase, to set the *busy*-signal in case certain of certain trigger errors, e.g. missing L1 triggers or spurious L2. Events in the queue would be sent to the *DAQ* and allow, together with the records of the trigger logging unit, for a detailed error analysis.

[12]Except for the information about the *TMU* status, which is only accessible upon transmission of the *TMU* header words.

6 Readout and Processing of TMU Data

Between an L0 and an L1 trigger, the Track Matching entities reconstruct particle tracks from tracklet data and compute the particles transverse momenta. On the basis of the latter, the *GTU*'s contribution to the L1 trigger is calculated. Furthermore, upon reception of a L2a trigger, the event data stored in the buffers of the *TMU* is sent to the *DAQ*. In its role as a concentrator board for one detector stack, the Supermodule Unit collects data from the *TMU*s. Track data is processed and forwarded to the Trigger Unit with minimum latency. In the case of raw data, the data is gathered and its header analyzed for errors, before a well-formed data stream is assembled and sent to the *DAQ*.

Presented in the following sections is the readout structure developed in this thesis. After a survey of the data path from *TMU*s through the *SMU* to the *DAQ*/*TGU*, the important building blocks are described in detail.

6.1 Layout and Data Format

The data path from *TMU* to *SMU* shown in figure 6.1 can be divided into several entities: On the *TMU* side, the LVDS backplane interface connects to the Event Buffering Unit and the Track Matching design. A state machine attached to the dual-clock transmission FIFO handles the event readout upon request. Event and track data as well as control and status information of the *TMU* are transmitted via the backplane.
On the side of the *SMU*, the interface also provides ports for the transmission of control signals. A monitoring entity inspects the incoming event data header which is simultaneously written to the receive FIFO for errors. Incoming track data is forwarded to the track merger, which communicates the L1 contribution of the segment to the *TGU*. Five instances of this interface manage data transmission to the *TMU*s of a segment.

Upon a L2a, the readout logic, which consists of Multi-event Readout Unit and Dispatcher, receives a start signal from the *SMU* control and requests an event from the *TMU*s. The Dispatcher executes the readout. A start signal (*event_accept*), communicated to all Track Matching Units, induces the sending of event data preceded by an index word and header[1]. After the CDH has been updated with the information gathered from the *TMU* headers, the data stream is assembled and written into the *SIU* buffer. As indicated by the signal marked red in figure 6.1, the whole process is subject to reverse flow control

[1]To simplify matters, index word and header might also be condensed to just 'header'.

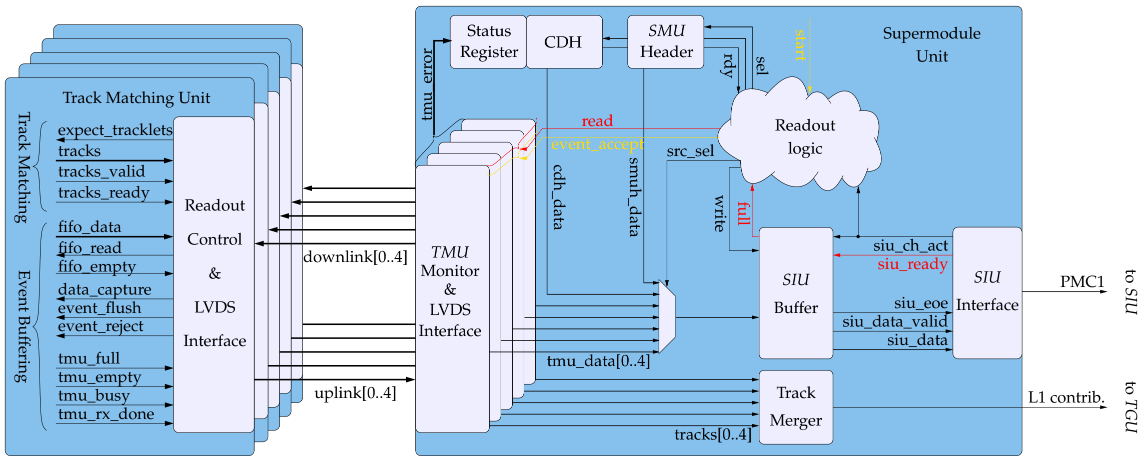


Figure 6.1: Survey of the data path from *TMU* to *SMU*. Marked in red are the signals implementing the reverse flow control. Signals used to initiate the readout of an event are labeled in orange. Both are transmitted via the backplane using the downlink lines. Please note that not all signals are shown on the *SMU*.

due to the saturation of the Detector Data Link.
The *SIU* buffer detaches the *SIU* interface from the rest of the data path, which was layouted to deliver 32 bit words at a rate of 60 MHz. This is desirable to saturate the *SIU*, whose operation frequency can be adjusted via PowerPC.

The *ALICE TRD DAQ* Data Format (see table 6.1), which has recently been specified ([dCR07]), defines the structure of the data stream transmitted from *GTU* to *DAQ* for each physics event.
All events are preceded by the Common Data Header, according to the specifications for the *ALICE DDL* ([DJVdV07]). Next are the Supermodule Index Word and the Supermodule Header. The header length is specified in the index word, as are Stack Mask[2] and Trigger Bit[3].
Consecutively follow Index Word and Header for the stacks. They are only transmitted if the Stack Mask indicates the presence of the corresponding *TMU*. Again, the length of the header is specified in the index word, as is the optical link mask which announces the active half-chambers.
Finally the data content for stacks announced in the Stack Mask follows. Each stack data block consists of data from the links, whose presence is indicated by the Link Mask of the stack. The Stack 'i' Link 'j' Data Content consists of detector raw data which is preceded by tracklet data, if the Trigger Bit is set. The end of tracklet data is indicated by exactly two tracklet end markers (0x10001000), the end of raw data by one to four raw data end markers (0x00000000), due to the 128 bit padding in the event buffers.

[2]Stack Mask: also referred to as *TMU* mask
[3]Trigger Bit: also referred to as Tracklet Bit

Size (32 bit words)	Id	Content
8	Common Data Header	as specified in [DJVdV07]
1	Supermodule Index Word	header length, trigger bit, stack mask
var	Supermodule Header	board id
1	Stack 0 Index Word	header length, link mask
var	Stack 0 Header	*TMU* status and error information
1	Stack 1 Index Word	
var	Stack 1 Header	
1	Stack 2 Index Word	
var	Stack 2 Header	
1	Stack 3 Index Word	
var	Stack 3 Header	
1	Stack 4 Index Word	
var	Stack 4 Header	
var	Stack 0 Data Content	Link 0 Data Content ⋮ Link 12 Data Content
var	Stack 1 Data Content	
var	Stack 2 Data Content	
var	Stack 3 Data Content	
var	Stack 4 Data Content	

Table 6.1: The *ALICE TRD DAQ* Data Format. Content for Stack Header, Index Word and Stack Data is shwon as an example for Stack 0. The Supermodule and stack header contents shown reflect the current development status and might be subject to changes in the future.

6.2 Data Transmission via the LVDS Backplane

The interconnection between *SMU* and the *TMU*s of a *GTU* segment is established by a custom built LVDS backplane. Due to the maximum density of high-speed differential board-to-board connectors, only 65 lines (3 in down[4] and 10 in up direction per *TMU*) comprise the physical connection.

6.2.1 Design Requirements

Control information between the *SMU* and the *TMU*s as well as event raw and trigger data is transmitted via the LVDS backplane. The latter is available only after the tracklet

[4]Backplane direction: A signal originating at the *SMU* and arriving at a *TMU* is referred to as downlink. The opposite direction is denominated uplink.

data from the supermodule has been processed. As explained earlier, transmitting track data with minimal latency is crucial. A second important aspect is the utilization ratio of the Detector Data link to the *DAQ*. In order to achieve maximum link utilization, thus effectively minimizing the dead time of the *GTU*, data transmission via the backplane has to be sufficiently fast. Those were the main design considerations for the backplane interfaces and the readout structure on both sides.

The Supermodule Unit manages the data stored on the *TMU*s according to the directives given by the *ALICE* trigger system. In order to operate, the *TMU*s need information about the current state of the trigger sequences and control signals. Furthermore, it is essential to monitor the status of the system and to report errors which might have occurred by assembling a special Common Data Header (CDH). The requirements to the designs on both sides of the backplane can be summarized:

- Transmission of track data to the *SMU* with highest priority and low latency
- Readout of event data with reverse flow control
- Guarantee error-free transmission of track and event data
- Transmission of control signals from *SMU* to *TMU*
  The Track Matching design takes instructions when to expect tracklet data. Depending on the trigger, the Event Buffering Design must reject or drop an event. To record tracklets together with raw data for a given event, data recording must be controllable from the *SMU*.
- Transmission of status signals from *TMU* to *SMU*
  In order to administer the events in the *TMU* memory, the *SMU* needs information on the buffer fill levels. The *TMU busy* and *data_rx_done*, the latter indicating the adding of an event to the event buffer, must be available to the *SMU*.
- Monitor *TMU* status and error information and include it into the CDH
  It must be known to the *SMU*, whether any errors occurred on the *TMU*s prior to the transmission of the CDH. Each *TMU* includes this information in the prepended header.

6.2.2 Interfaces on the *TMU* Side

Figure 6.2 illustrates the detailed structure of the LVDS backplane interface on the *TMU* side. For the transmission of signals via the backplane in up and down directions, the two DDR stages provide 40 -bit and 8 -bit ports at 120 MHz single data rate, which are translated to 10 bit and 2 bit at 240 MHz DDR for transmission over the backplane. The 40 -bit uplink is logically divided in two 20 -bit channels, *up_high* and *up_low*.

The interface to the Track Matching design consists of four signals. The *expect_tracklets* signal is asserted by the *SMU*. It defines a time frame between L0 and L1 in which tracklets must be expected. Once the Track Matching design has finished processing supermodule tracklets, it sets *tracklets_rdy*. Track data (*tracks*) is then strobed with *tracks_valid*.

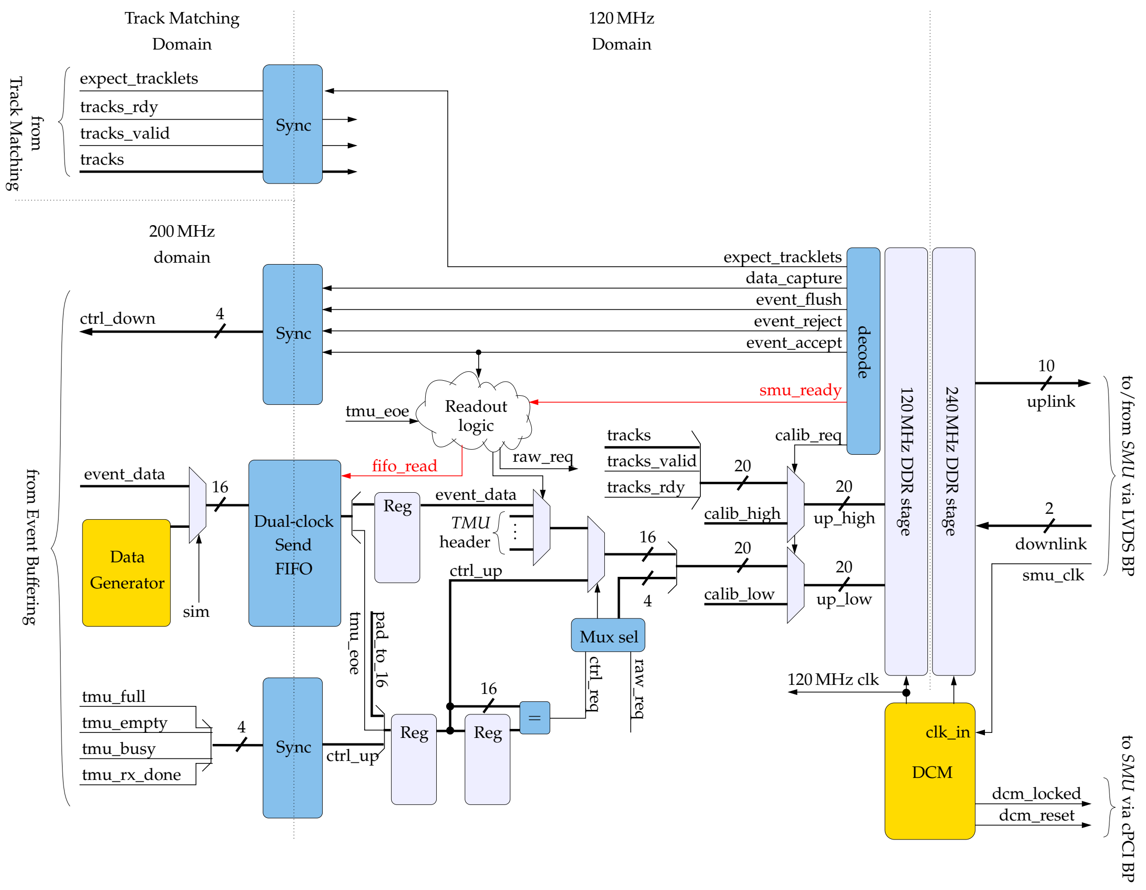


Figure 6.2: The backplane interface on the *TMU* side. Signals implementing the reverse flow control originating at the *SIU* are marked in red.

The deassertion of *tracks_rdy* signals that the last track has been sent.
Due to their time critical nature, these signals are transmitted to the *SMU* using the *up_high*-channel exclusively to ensure minimum latency.[5]

The control signals arriving at the *TMU* via the downlink lines are decoded and made available to the interface.[6] The *data_capture* signal as well as *event_flush* and *event_reject* are connected to the Event Buffering Design. *Data_capture* enables raw data recording on the *TMU*s. When asserted at the time of a L0, tracklet data and event data are recorded. If asserted after a L1, only event raw data is recorded. *Event_reject* causes the oldest event from the event buffer to be dropped, whereas *event_flush* aborts buffering of the currently incoming event.

Originating at the Event Buffering Unit, *tmu_full* and *tmu_empty* reflect the state of the event buffer. *Tmu_rx_done* indicates that the reception of raw data from the supermodule

[5]Two bits on *up_high* are currently unused.
[6]Eight lines would suffice for currently six control signals. However, due to historical reasons, encoding the transmission has not been removed.

for the present event is complete. Together with *tmu_busy* and *tmu_eoe*, these signals form the control data to be sent to the *SMU*.

On the *up_low*-channel, transmission of control signals to the *SMU* takes precedence over event related data. Two successive register stages, *ctrl_r1* and *ctrl_r2*, receive a maximum of 16 control signals. A difference between the register contents generates a *ctrl_req* request, upon which the content of *ctrl_r1* is transmitted. Therefore, the channel can be used to simultaneously transmit raw data and control signals issued by the *TMU* with minimum delay.

Event_accept causes the Event Buffering Unit to fetch data for an event from the event buffer and serves as start signal for the readout logic. If *smu_ready* is active, i.e. the *SMU* receiver FIFOs are not full and ready to receive, the readout logic requests the transmission of event data and starts with the *TMU* header. It then issues FIFO read commands until the end-of-event flag is encountered. The readout then is complete, and the logic returns to its idle state. The end of event is signaled to the *SMU* by sending a control word.

In order to operate the design synchronously to avoid delays caused by additional synchronizer stages, the interface clock is distributed by the *SMU* to the *TMU* via the LVDS backplane. The design and the DDR stages operate using 120 MHz and 240 MHz clocks, both phase matched to the reference clock.

6.2.3 Interface on the *SMU* Side

The layout of the LVDS backplane design on the *SMU* side, shown in figure 6.3, provides the necessary ports for communication with a *TMU*.
Similar to the *TMU* side, two DDR stages serve as interface between the 240 MHz DDR backplane domain and the user logic which is operated at 120 MHz. Trigger related data, arriving exclusively on the *up_high* channel, is directly forwarded to the Track Merger Unit, which processes the data and communicates a contribution to the *TGU*[7].

Arriving on the *up_low* channel are 16-bit raw data (including the *TMU* headers) and control words, together with a 2-bit id and valid tag. If the data is a control word and valid, the control register is updated and the signals are made available to the rest of the *SMU* design.

Buffering *TMU* Header Data In order to report correct event status information, the headers sent by the Track Matching Unit have to be monitored for errors prior to the assembly of the CDH and transmission to the *DAQ*. Thus, an additional buffering stage is required in the *SMU*.

[7]Since the trigger design is not available, functionality of the Track Merger design is not yet implemented.

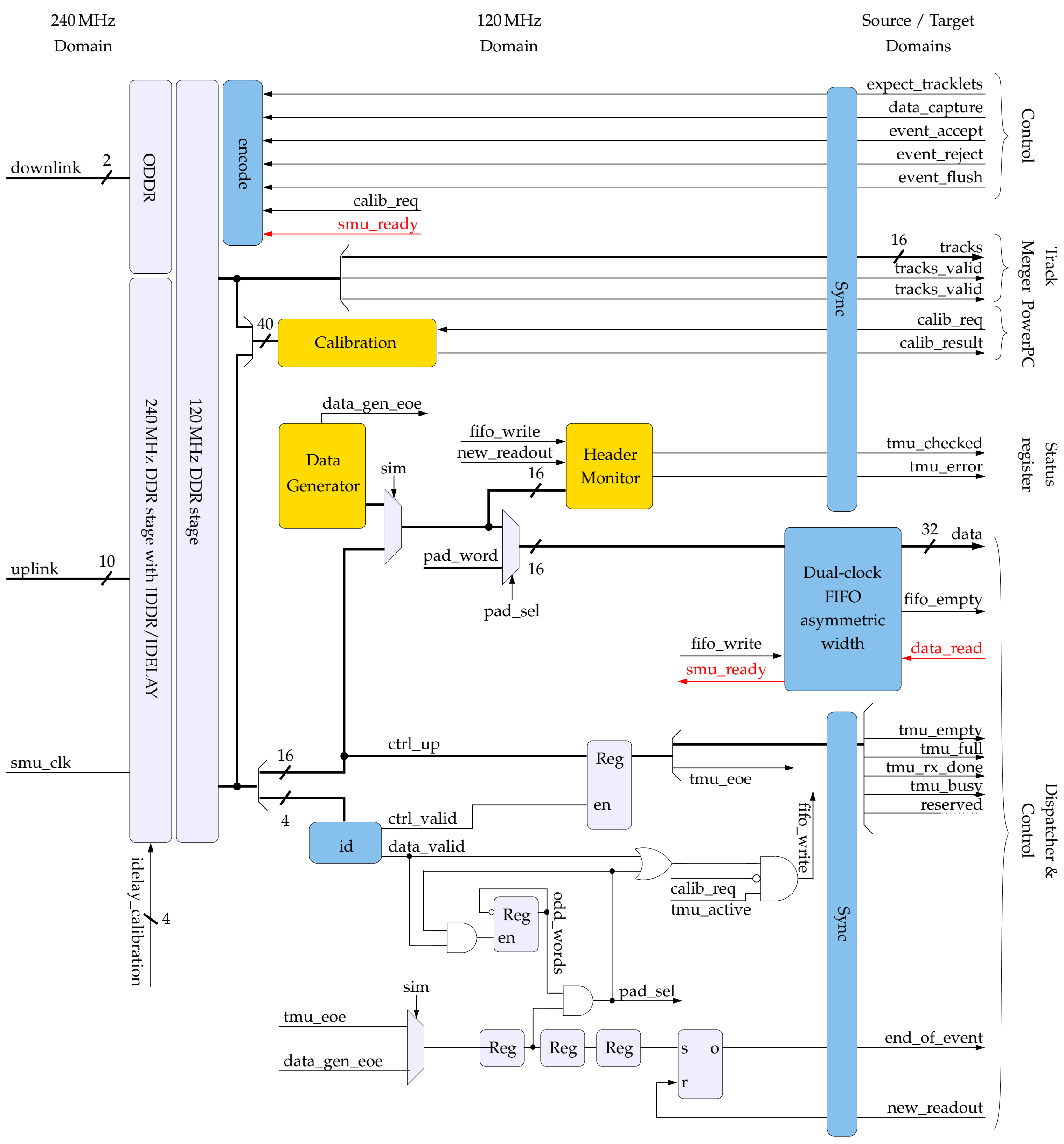


Figure 6.3: The LVDS backplane interface on the *SMU* side. An interface instance exists for every *TMU*.

Used for buffering is a dual-clock FIFO with asymmetric read and write aspect ratio which can hold 64 16 bit words. The reverse flow control is realized by the programmable-full signal of the FIFO, here denoted *smu_ready*. Simulation shows a delay of eight clock cycles for the transmission of *smu_ready* to the *TMU* readout state machine. Thus, a threshold of 48 entries gives sufficient margin to compensate for the latency caused by the backplane transmission and ensures the uninterrupted arrival of all header words (currently planned are 16×16 bit), which are the first data sent by the *TMU* for any given event.
By accessing the internal storage matrix in an asymmetric manner, the FIFO combines two written 16 bit words to a 32 bit word on the read side. Since the FIFO does not allow partial words to be accessed, this has implications on the status flags *full* and *emtpy* ([Xil06a]). Of importance here is the behaviour of the *empty* flag, which is active when one or no 16 bit words are stored inside the FIFO. The dispatcher, described in detail in section 6.3, evaluates *end_of_event* and *fifo_empty* in the readout process for a segment. If both are active, it is assumed that all data from this *TMU* has been read, and the dispatcher advances to the next state.
To avoid corruption of an event which would occur if one 16 bit word erroneously remained in the FIFO, write data, if necessary, gets aligned to 32 bit words by padding with a raw data end marker[8]. Furthermore, writing to the FIFO is disabled during calibration mode and if the *TMU* is inactive.

Simultaneously to the storage in the FIFO, the header of the arriving event data is monitored in the Header Monitor Unit. The header contains detailed information on the state of the optical data transmission between supermodule and *GTU*[9]. Once all header words have been checked, the signals *tmu_checked* and *tmu_error* are set, the latter reflecting the condition of the *TMU*. A *new_readout* resets the Header Monitor for analysis of the next event.

Backplane IDELAY Calibration On the *SMU* side, IDELAY modules are utilized to correctly adjust the capturing of incoming backplane data. Virtex-4 devices provide such modules in every user input cell. A 64-tap delay allows adjustment of the delay in steps of 78 ps, incoming data can thus be delayed by more than 1.1 clock cycles when operating at 240 MHz.

During development it became apparent that an automatic mechanism for the calibration of these IDELAY settings is needed to ensure error-free transmission. Setting a constant value for the IDELAY cells at compile time did not suffice and resulted in frequent bit errors when transmitting via the backplane. Therefore, a two-phased auto-calibration

[8]This case should never occur, since that data is aligned to 128 bit SRAM lines in the *TMU*s. However, this safety measure proved valuable during development.
[9]Currently, the *TMU* design does not support the sending of header data. The exact format is yet to be specified.

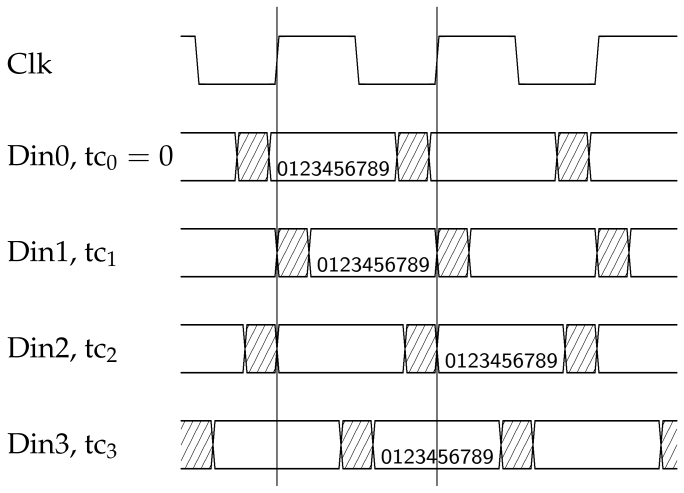


Figure 6.4: IDELAY calibration. The incoming test pattern is shifted with respect to the capturing clock by varying the length of the delay line (tc) in the IDELAY modules. Din0 is the undelayed signal. Din1 and Din2 are delayed by tc_1 and tc_2, respectively, and illustrate sampling at the edges of the valid window. Din3 is sampled with sufficient setup and hold margins, in this case the tap count is determined by $\mathrm{tc}_3 = (\mathrm{tc}_2 + \mathrm{tc}_1)/2$.

process has been implemented. First, the *TMU*s are probed for response to determine the current mask of active *TMU*s. Then the correct tap count is determined for each *TMU* and a mean value is calculated.

The *SMU* monitors the status of the DCMs instantiated on the Track Matching Units. To minimize resource utilization, only one reserved signal on the CompactPCI backplane (*all_dcms_locked*) is used, to which all boards are connected. If *dcm_locked* is inactive, the *TMU* drives *all_dcms_locked* low. If active or if the board is removed, the signal is undriven and pull-ups arrange for a high signal level.
After power-up of the segment, the PowerPC calibration software on the *SMU* issues a calibration request (*calib_req*) once the backplane DCMs indicate a stable clock signal with proper frequency and phase. Communicating this request to all *TMU* boards initiates sending of a 40-bit test pattern.
Calib_result originates from the continuous comparison of this incoming pattern against a reference pattern. On a higher design level, it is masked with the *TMU* mask and combined with *calib_result* from the other *TMU*s to form the *calib_ok*-signal, which is routed to the PowerPC and evaluated by the software.

Once the PowerPC request is acknowledged by the *SMU* hardware, the current *TMU* mask is determined. Since the PowerPC only receives the combined *calib_ok* signal, the active mask is set by software to mask out all *TMU*s but one. When scanning through the whole tap count range, an active *calib_ok*, in this case equal to *calib_result*, signals the presence and readiness of the *TMU*. Repeating this procedure for all boards gives the *TMU* active mask.

As illustrated in figure 6.4, the correct tap count value for capturing data can be determined by measuring the window in which data is correctly recorded. While incrementing the tap count, data is sampled and *calib_ok* evaluated. When the sampled data matches the reference value on all *TMU*s, *calib_ok* is set to active. The tap count values which correspond to a transition of *calib_ok* mark the borders of the valid window.

6.3 Data Transmission to the Data Acquisition System

6.3.1 SIU Buffer and Interface

Detaching the *SIU* Clock Domain The specification for the *SIU* operating frequency f_{siu} has recently been downgraded from 60 MHz to 50 MHz. To comply with the specification, a dual-clock buffer detaches the *SIU* clock domain from the rest of the data path, which is designed to deliver 32-bit words at a rate of 60 MHz. The Digital Clock Managers, which produce the *SIU* clocks, are configurable and can deliver several frequencies in the range under consideration. Accessing the DCM through the Dynamic Reconfiguration Port allows to change the operation parameters in the running design. The DRP protocol is implemented as PowerPC software. *DCLK* provides synchronous timing for the port. Further details about the configuration of the DCMs can be found in [Xil06b].
It is possible to undertake tests in the final setup in which the data stream is monitored for errors in large volumes for increasing values of f_{siu}. Data generators implemented at several stages in the *GTU* can be utilized to deliver well-defined test patterns. Such tests are desirable to obtain the maximum *SIU* interface frequency, since this has direct impact on the *GTU* dead time.

A detailed illustration of the buffer is shown in figure 6.5. The FIFO synchronizes data and the end-of-event-signal. On the read side, data is transmitted whenever the FIFO is not empty and the *SIU* signals 'ready'. When the synchronization word is encountered, it is dropped and *siu_eoe_out* is pulsed to signal the end of the event.
Two DCMs are instantiated: The 'freq_gen'-DCM receives the 200 MHz system clock f_{sys} and generates f_{siu} according to:

$$f_{\text{siu}} = f_{\text{sys}} \cdot (M/D)$$

with $2 \leq M, D \leq 32; M, D \in \mathbb{N}$. The second DCM provides a clock ($f_{\text{siu},270}$) with the same frequency as f_{siu}, but phase shifted by 270 degrees.

[10]The wiring of the DCMs does not give the best result in terms of jitter, therefore Xilinx recommends a

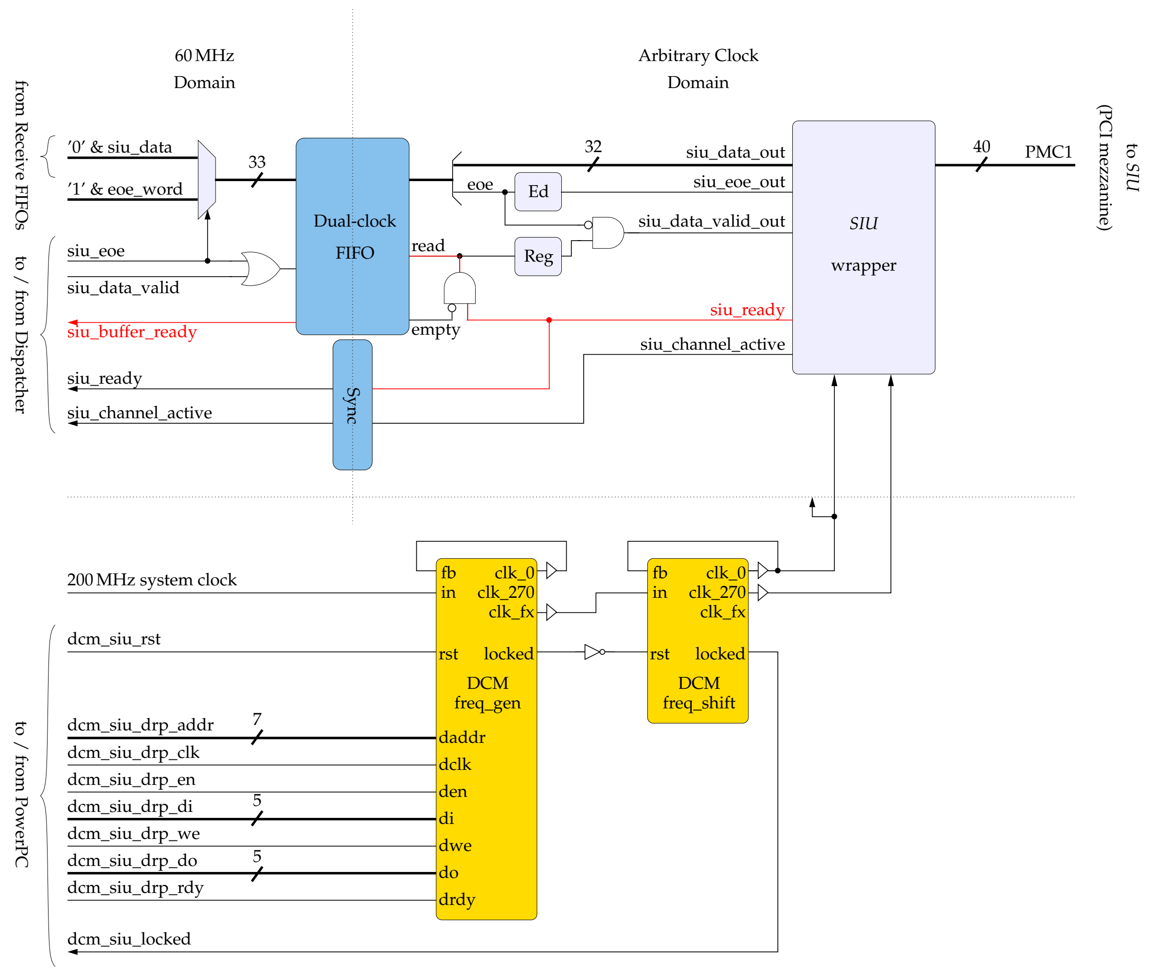


Figure 6.5: Survey of the *SIU* buffer and interface entity. Separate DCMs[10] provide the adjustable clocks for *SIU* interface.

***SIU* Interface Timing** Data is presented to the *DDL* with the unshifted clock f_{siu}. The clock *foCLK*, which is derived from the phase shifted clock $f_{\text{siu},270}$, is connected to the *SIU* interface and serves as free running clock, used to synchronize the data transfer via the *DDL*. The Source Interface Unit requires a setup time for the data bus *fbD[31..0]* of 8 ns before the rising edge of *foCLK*, while 0 ns is specified for the hold time ([RG07]).
Figure 6.6 shows logic analyzer measurements of the *SIU* interface timing for the extrema of the considered frequency range. The clocks and data are provided as described above. Shown is the transfer of a status word from *SMU* to *SIU*. For $f_{siu} = 50\,\text{MHz}$, figure 6.6(a) exhibits a setup time $t_{\text{setup,50Mhz}} = 12.68\,\text{ns}$, while figure 6.6(b) shows $t_{\text{setup,60MHz}} = 10.64\,\text{ns}$ for the 60 MHz operation frequency. The timing requirements are met in both

different implementation. However, the implementation is suitable, since jitter is of no concern in this application.

cases, and a safe margin for distortions (variations in the duty cycle of *foCLK*) is provided.

6.3.2 Event Readout Control

The readout logic which controls and initiates the readout of events and administers the transmission to the *DAQ* consists of two entities: Multi-event Readout Unit (MER) and Dispatcher.

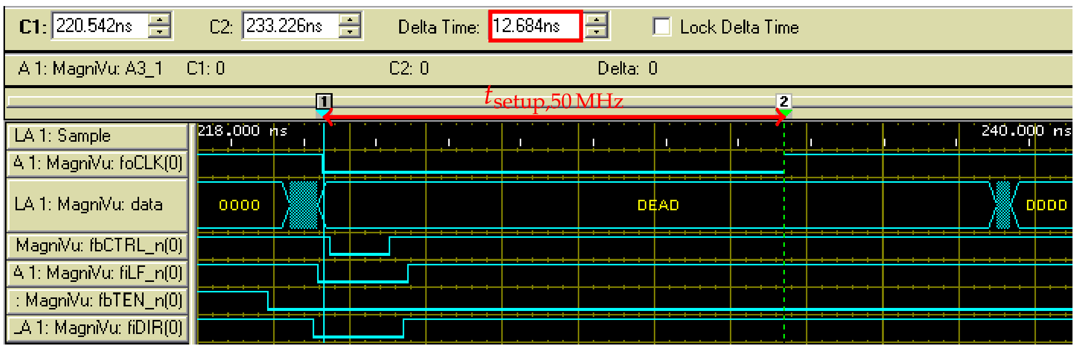
A start signal sent by the *SMU* control unit and received by the MER triggers the readout process. The MER starts the Dispatcher and the entities involved in readout subject to current operation mode. In order to be able to run without external trigger for testing purposes, the MER features a mode where the start signal triggers the readout of *nrun*[11] events, using either the data from the *TMU*s or from the event generators situated in the *SMU* LVDS backplane instances[12]. The data source is determined by the *local_evgen_mode* flag. Operating MER with the default value $nrun = 1$ is compatible with triggering by external entities as used throughout advanced stages of development.

For a valid physics event, the MER causes the Track Matching Units to deliver event and header data by sending an *event_accept*. Since only complete events can be requested from the event buffers, the MER evaluates the status of the buffers of all active *TMU*s prior to sending the *event_accept* and starting the Dispatcher.
No event is requested if the local event generator (Evgen pre Rx Fifo, see figure 6.8) is chosen as data source or if the *cdh_only* flag is set. The latter originates from the control unit and is set in the case of special events, such as SoD or EoD, or errors, where the Common Data Header only is to be sent without payload.

The Dispatcher is started by the MER once the entities involved in readout signal 'ready'. It assembles the data stream according to the chosen format specified in table 6.1 and implements the reverse flow control. All read/write requests to FIFOs and data header entities and the settings of the data path multiplexer are generated here.
The structure of the Dispatcher state machine, pictured in figure 6.7, is chosen to reflect the form of the data stream. After leaving the *idle*, the CDH is written into the *SIU* buffer FIFO, once it has been updated according to the header information transmitted by the *TMU*s. The state machine then advances from *send CDH* to *end CDH*. The *cdh_only* flag is evaluated here. If active, as in case of SoD and EoD events, the end of data transfer sequence is signaled by pulsing *siu_eoe* and the state machine returns to *idle*. If inactive, sending continues with all words of the *SMU* header, followed by the *TMU* headers, before the machine passes through *send TMU0 header* to *send TMU4 header*. Here, the first

[11]The number of events register, *nrun*, is PowerPC-controlable.

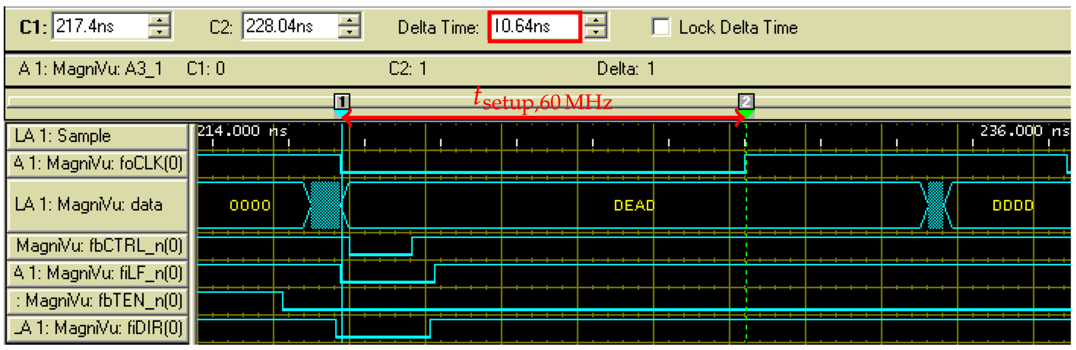
[12]The interaction of *TMU*s and *SMU* was tested with this mode in the early stages of development, where external trigger possibilities were not available.



C1: 220.542ns C2: 233.226ns Delta Time: 12.684ns Lock Delta Time

A 1: MagniVu: A3_1 C1: 0 C2: 0 Delta: 0

(a) 50 MHz



C1: 217.4ns C2: 228.04ns Delta Time: 10.64ns Lock Delta Time

A 1: MagniVu: A3_1 C1: 0 C2: 1 Delta: 1

(b) 60 MHz

Figure 6.6: *SIU* data transfer at 50 MHz and 60 MHz interface frequency. In both cases, the timing requirements are met if the clock *foCLK* is phase shifted by 270 degrees. Signals shown in the timing diagram are:
foCLK : Interface clock, free running.
data : Data line. The lowest 16 bit of the transfered word were measured.
fiDir : Direction. When high, data is transfered from *SMU* to *SIU*.
fbTEN_n : Enable. When low, data transfer from *SMU* to *SIU* is enabled.
fiLF_n : Flow control. When low, data transfer has to be halted.
fbCtrl_n : Control. Low indicates the presence of a status or command word.

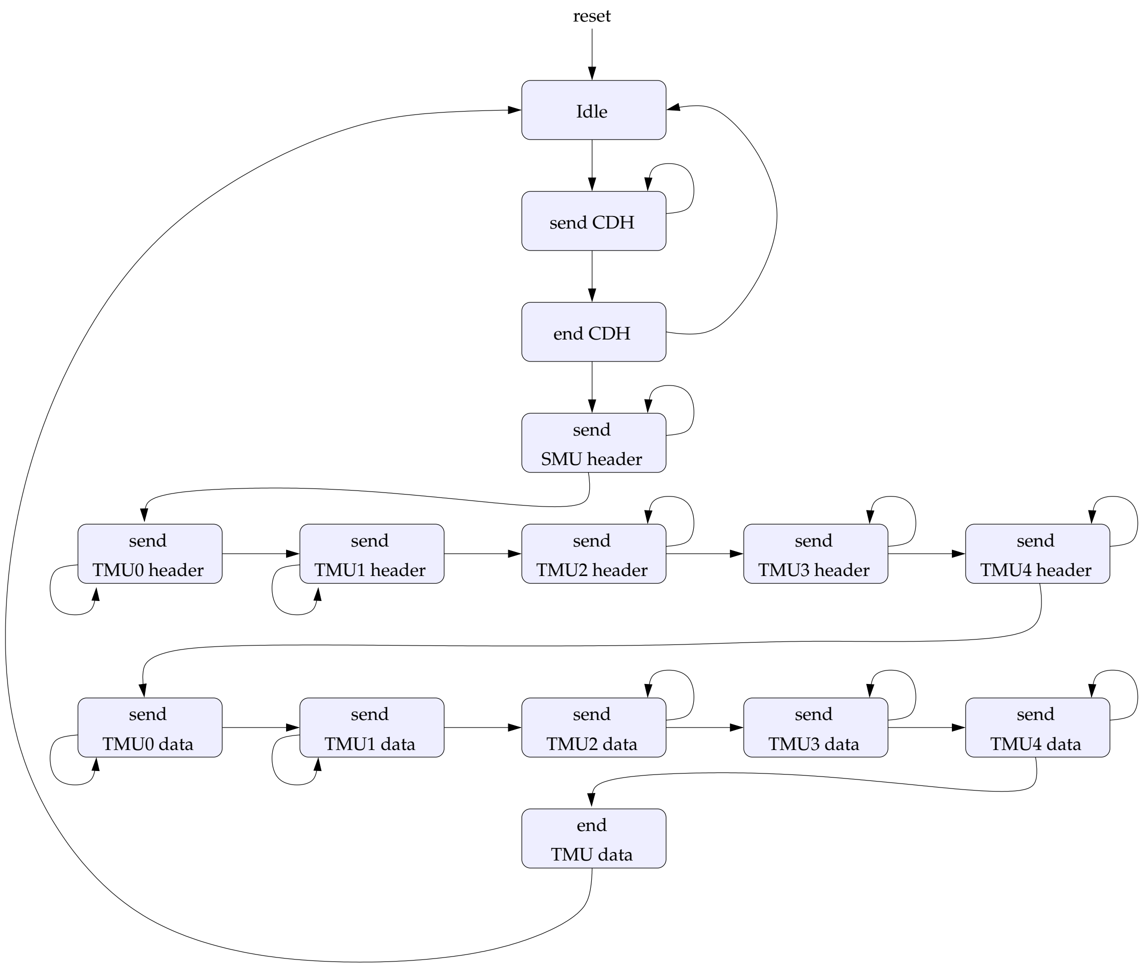


Figure 6.7: the dispatcher state machine

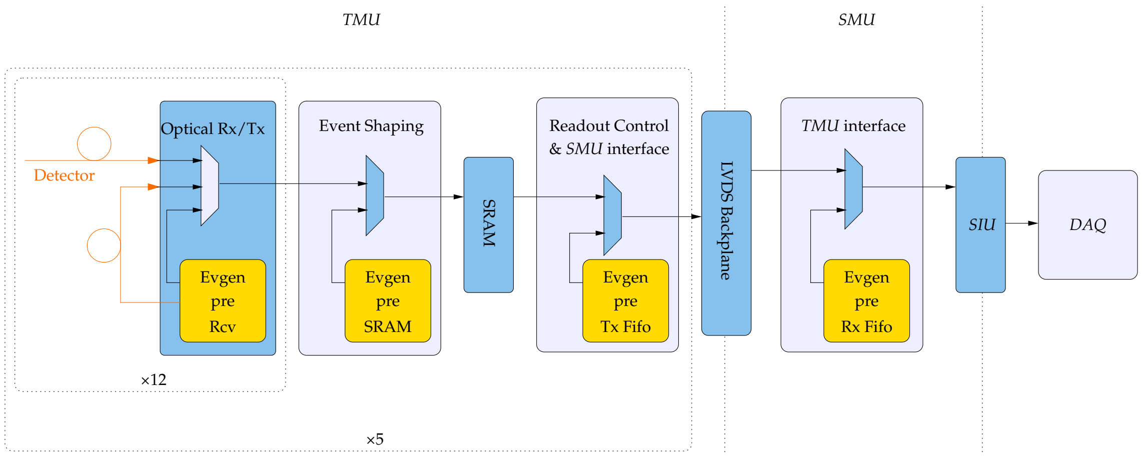


Figure 6.8: Event generators at the different stages of the data path shown for one *GTU* segment. The possibility switch them into the data path on demand provides powerful debugging feasibility. Only the entities containing event generators are shown.

n words[13] in the *SMU* backplane receiver FIFOs are sent as header, if the *TMU* is active. Next follows the transmission of event and tracklet data for active *TMU*s in the states *send TMU0 data* to *send TMU4 data*. The state transition between the send data states is triggered when the flags *end_of_event* and *fifo_empty*, coming from the backplane interface (figure 6.3) of the current *TMU*, become active. This signals that the *TMU* sub-event has been written into the *SIU* buffer. Prior to returning to idle, the system marks the readout of the complete segment by pulsing *siu_eoe*, which is synchronized in the *SIU* buffer FIFO and communicated to the *SIU* interface.

6.4 Test Results

For tests with high data rates and defined test patterns, the presented *SMU* design and the design of the *TMU* feature event generators at various critical stages in the data path. Via PowerPC, the amount and type of data they produce can be configured and they can be switched into the data path on demand. Figure 6.8 shows their location in the data path.
Pattern generators feeding optical transmitters may be used to produce test data. Via optical (or electrical) loopback, these can be injected into the first stage of the data path. The event generator located prior to the SRAM controller is especially designed for SRAM testing. Two generators are located near the LVDS interfaces on both sides of the LVDS backplane. Comparison of data from both event generators received in the *DAQ* allows

[13]The number of *TMU* header words is specified at compile time.

localization of possible error sources to the *TMU* or *SMU*. Generation of an event is triggered by the *data_capture*-signal which is issued by the *SMU* at the time of either a L0 or L1 trigger.

The presented design has been successfully tested at CERN and is currently used for supermodule production testing.

Supermodule Readout in Münster Currently, a *GTU* segment is used at IKP, Münster, in the production of supermodules. The segment is fully equipped with one *SMU* and five *TMU*s. The goal of the tests was the automated readout of the supermodule using the full data path through the *GTU* to the *DAQ*. With no *LTU* available, the *SMU*'s capabilities for simple triggering are used. Figure 6.9 shows the test setup.

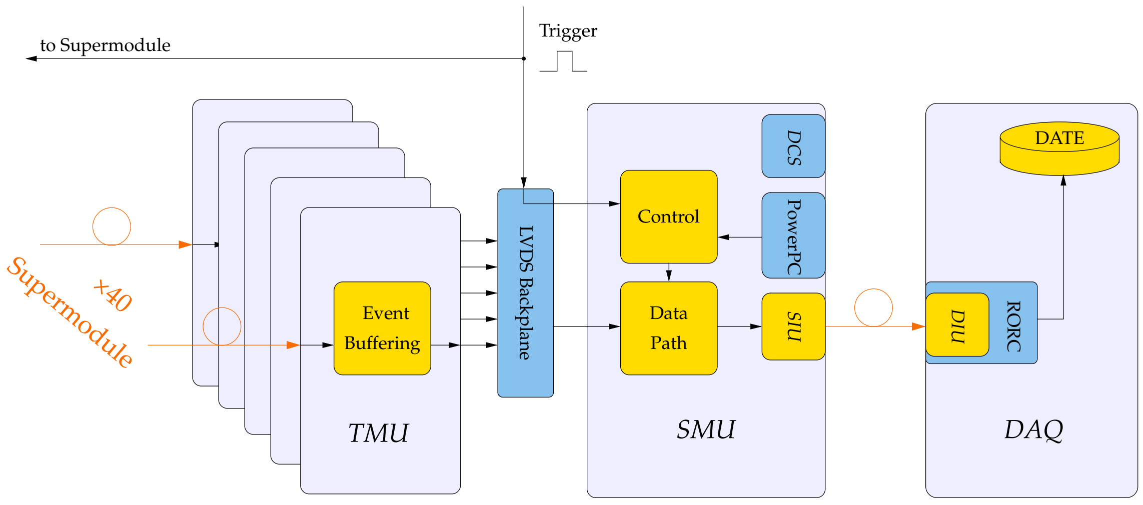


Figure 6.9: Test setup in Münster

40 fibres from four layers of five stacks are connected to the Track Matching Units. The *GTU* system is configured via PowerPC to generate a valid, artificial CDH for each event transmitted to the *DAQ*, as no CTP event information is available. This CDH contains the *SMU* board ID, and a value that is incremented with each event for *EventID2*. One *DDL* fibre connects the *SIU* to the *DIU* in the *DAQ* PC. Event data is written out in ROOT[14]-format.

During the test, high-voltage for the drift chambers was disabled. However, with powered supermodule electronics, over 2000 noise events were successfully taken at a trigger rate of 10 Hz. Figure 6.10 and figure 6.11 visualize the recorded data for layer 2 of the supermodule. Shown in the upper half is the channel noise, given by the rms of the 30 time-bins. The MCM noise, shown in the lower half, is the rms of the 21 channels for each MCM.

[14]http://root.cern.ch

Tests at CERN At CERN, the event generators located in the optical transmitters were used to simulate incoming event data from the supermodule. Even with no supermodule powered at this time, testing of the full chain to the data acquisition and *HLT* was possible. Tests were conducted with up to four *GTU* segments, using the *TRD LTU* partition for the emulation of *CTP*. Several tests with the *DAQ* system and the *HLT* were conducted with L2a rates of up to 1 kHz. The *DAQ* system was configured for event building in the Global Data Collectors.

Events could successfully be built from the event fragments sent concurrently by four *GTU* segments. Enabled data format checks at the LDCs indicated that the *SMU*s produced valid Version 2 CDHs. With an event fragment size of 1.8 MB and a *DDL* throughput of 180 MB/s, the reverse flow control originating at the *DAQ* has been successfully tested. Manually slowing down the *DAQ* readout rate ultimately resulted in a decrease in the trigger rate, showing the correct propagation of the *SIU* busy through the *GTU* to the trigger system.

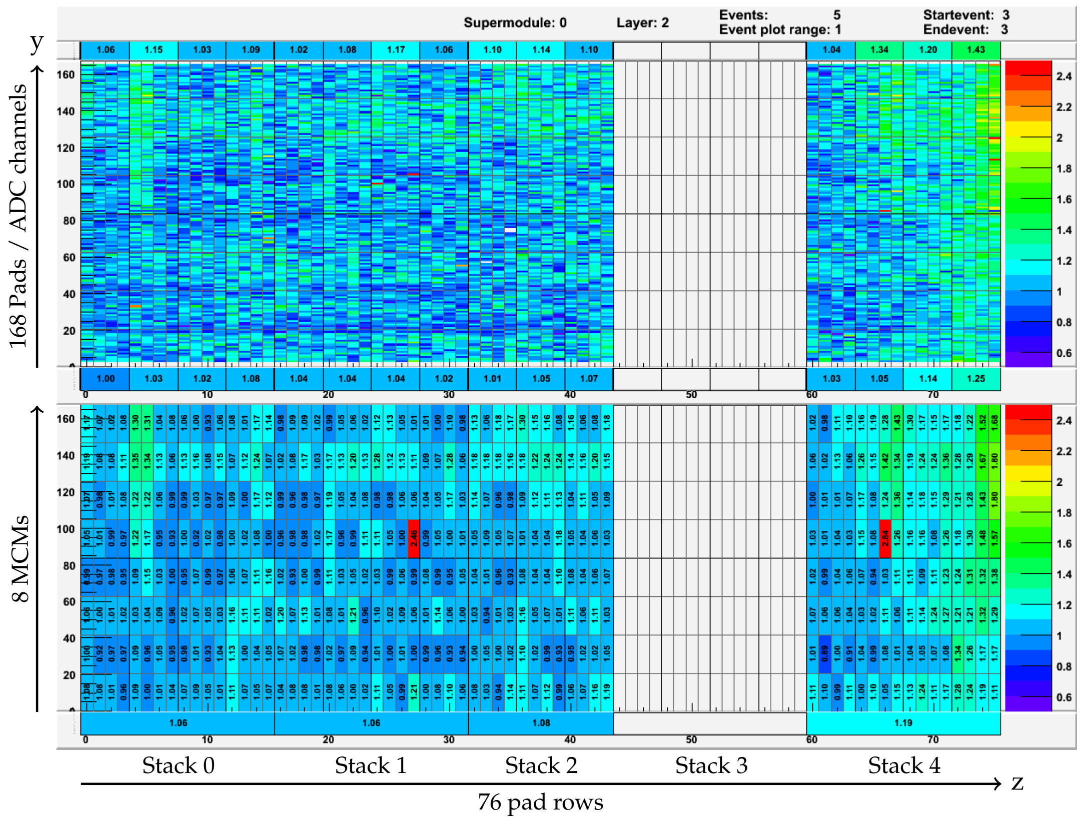


Figure 6.10: Noise plot of the supermodule layer 2 taken with the setup shown in figure 6.9 in Münster. The Track Matching Unit associated with stack 3 was disabled by setting the *TMU* mask accordingly.
The upper half shows the noise for each separate ADC channel. In the lower half, the average noise for each of the MCM's 21 channels is plotted.
In stack 1 and 4 individual ADC channels with strong noise increase the MCM's mean noise, as indicated by the red blocks. Stronger noise at the right side of stack 4 is due to cables, which are attached in close vicinity. Less noise is expected once the metal topping of the supermodule is mounted.
With thanks to Tom Dietel for the visualization.

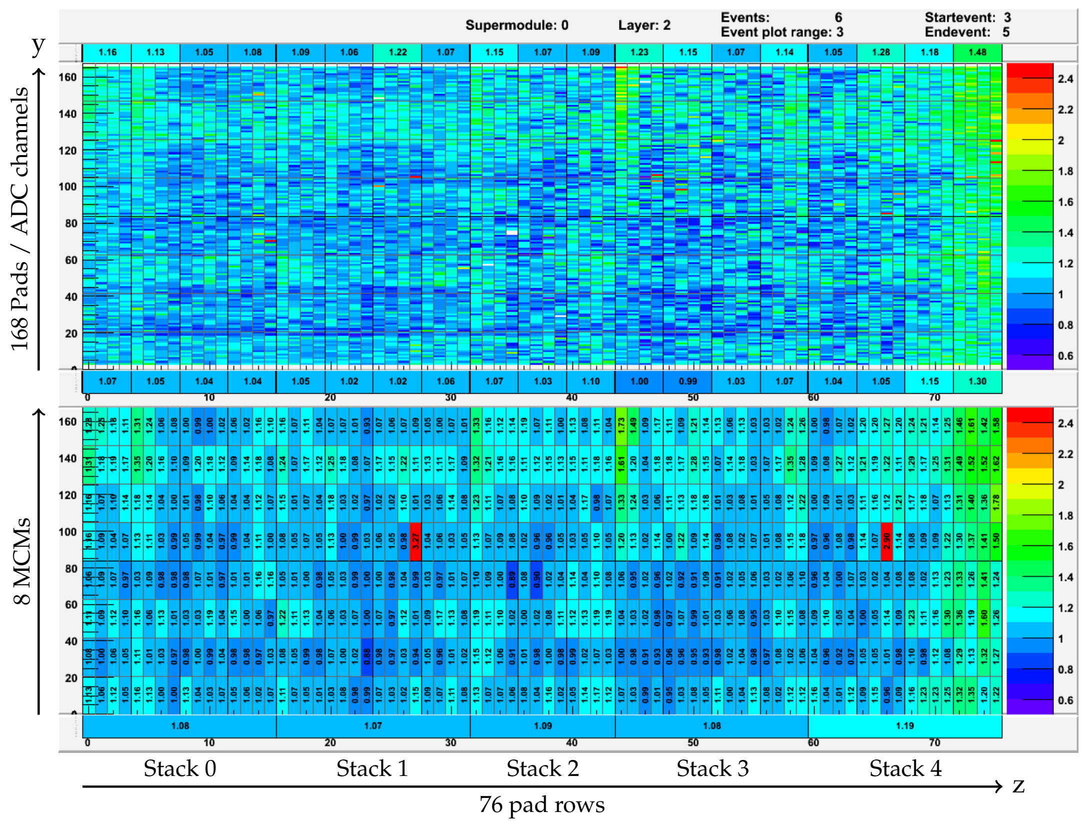


Figure 6.11: A second noise plot taken in Münster. All stacks and the associated *TMU*s are active. Again, the two channels with strong noise are clearly visible.

7 Conclusion

A sophisticated, hierarchical trigger system, required to deal with the huge amounts of data produced in heavy-ion collisions, issues directives for the operation and readout of the *ALICE* detectors. Dedicated, fast detectors enable the preselection of interesting physics events by distinct observables and give a contribution to the trigger for slower, high-resolution tracking devices.
In its function as a trigger detector, the Transition Radiation Detector preprocesses and reduces experiment data between Level0 and Level1 triggers in order to yield parameterized track segments. From these track segments, the Global Tracking Unit (*GTU*) reconstructs particle tracks and transverse momenta by means of massive parallel computation. Based on that, a contribution to the Level1 trigger is formed and available to the Central Trigger Processor (*CTP*) after 6 µs. Following a positive Level1 decision, the *GTU* stores arriving raw data for subsequent readout. To minimize *TRD* dead time and improve overall *ALICE* performance, the *GTU* is capable of buffering multiple raw data events, which are administered by the Supermodule Unit (*SMU*) according to the instructions given by the *CTP*.

This thesis specifies the requirements to the *SMU* and addresses the development and integration of an appropriate FPGA design into the GTU project. The design implemented in this thesis has been successfully tested at CERN and is currently used for supermodule production testing at the IKP, Münster.

Special efforts are taken in order to guarantee fault-tolerant operation even with erroneous instructions given by the *CTP*. Events that have arrived from the supermodule and are stored in the event buffers carry no identification tag, except the chronological order in which they are stored. In order to retain data integrity, *TTC* system traffic is monitored and checked for spurious triggers. All errors are reported to the higher level control systems, as they might cause desynchronization of data and event identification, which might not be detected for several hours.
Due to the large number of possible errors and the complexity of the *GTU*, error handling is a challenging task and complexity increases manifold when operating with multiple interlaced trigger sequences. In order to verify the correct operation of the *GTU* in such an environment and to speed up debugging in the commissioning phase, monitoring units are made capable of reflecting the detailed state of the system.

The readout of event data upon a Level2 accept is designed to provide maximum utilization of the Detector Data Link and flexibility for future improvements. The developed data path interfaces the event buffers of all five Track Matching Units via the LVDS backplane and manages the correct assembly of the event fragment for transmission to the Data Acquisition System. Once started, the readout unit independently handles the readout of an event. It is compatible to various trigger schemes and can easily be adapted to meet future requirements. Decoupling of the fast clock domain of the data path from the *DDL* interface domain and adjustable clock generation allow fine-tuning of the *DAQ* transmission for best performance.

With the conclusion of this thesis, a design is available that supports readout of event data in a setup with a *SMU*, five *TMU*s, a *TGU*, the *DAQ* and the *CTP*, which can be used for the forthcoming integration tests at cern. With a thoroughly tested data path and single-event buffering, improvements to the system can be approached. The next step is the implementation of multi-event buffering, as sketched in this thesis.

Appendix A

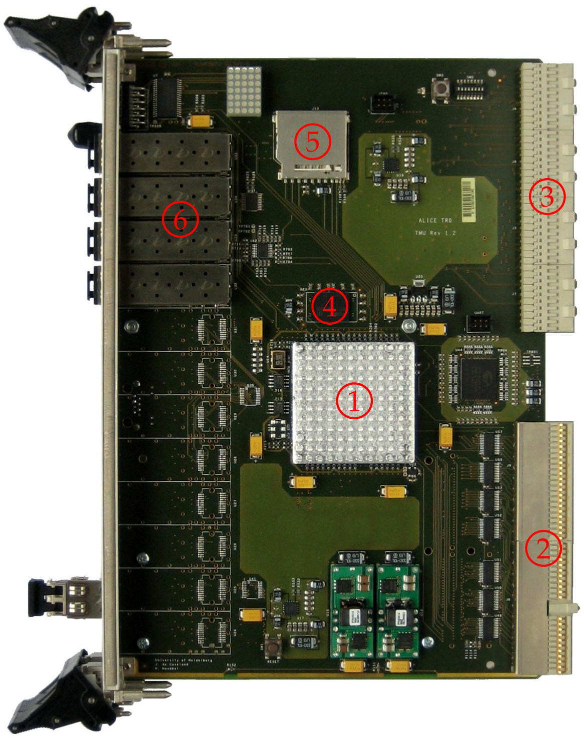
Pictures of the *SMU* and *TMU* Boards

Presented here are photographs of the *SMU* and *TMU* boards.

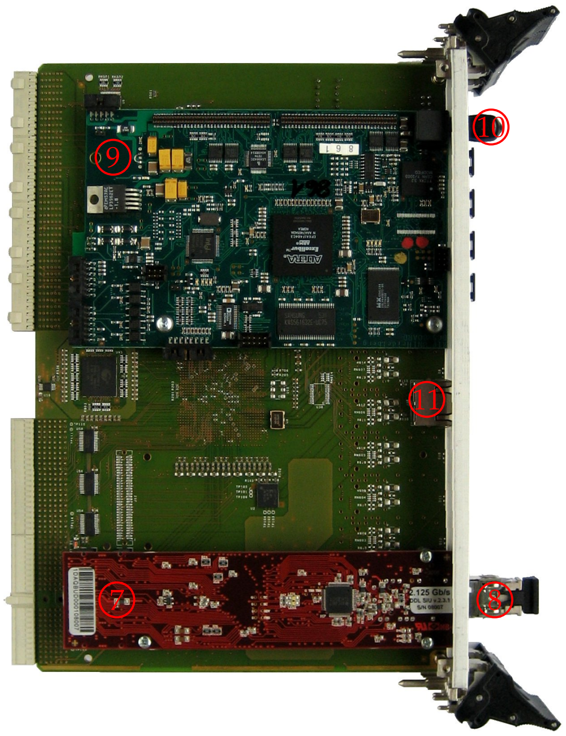
Realized as 6U-CompactPCI© boards, they share an identical 14-layer PCB design, but vary in the components that are equipped. Common to all boards is a high-end FPGA of the Virtex™-4 family. This large device provides 94896 configurable logical blocks and 768 freely usable I/O pins along with a variety of specialized functional blocks.

The Configuration for a Supermodule Unit equipped with four *SFP*s is shown in fig. A.1. On its back, the Source Interface Unit and the Detector Control System board are placed.

Fig. A.2 shows a *TMU* board, fully equipped with 12 *SFP*s to take in the optical fibres from the Supermodule.



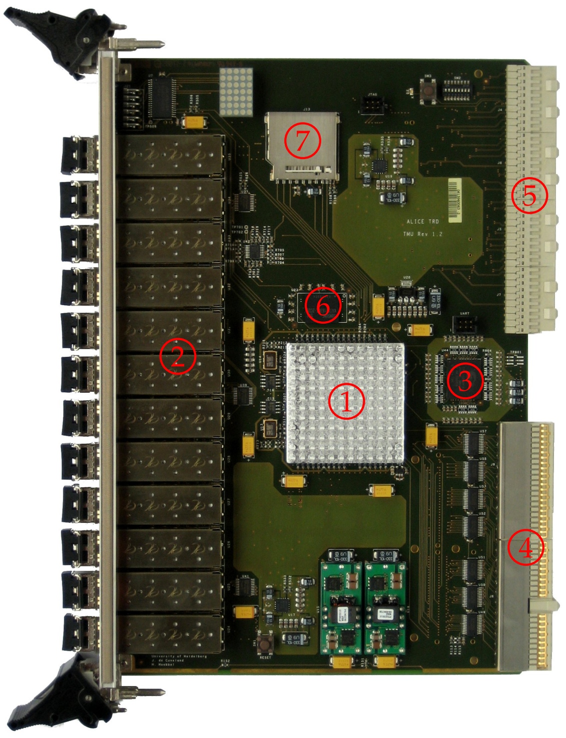
(a) top view



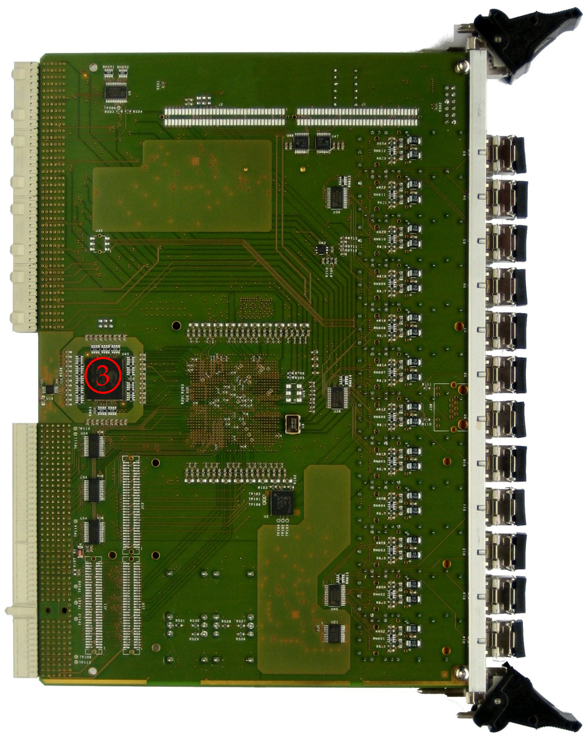
(b) bottom view

Figure A.1: Images of the SMU board

(1)	Virtex-4-FPGA
(2)	CompactPCI backplane
(3)	LVDS backplane
(4)	64 MByte SDRAM
(5)	SDCard slot
(6)	Optical transceiver modules
(7)	DDL SIU board
(8)	Optical transceiver module
(9)	Detector Control System Board
(10)	Optical TTC fibre connector
(11)	Ethernet connector



(a) top view



(b) bottom view

Figure A.2: Images of the TMU board

(1)	Virtex-4-FPGA
(2)	Optical transceiver modules
(3)	Event buffering SRAMs
(4)	CompactPCI backplane
(5)	LVDS backplane
(6)	64 MByte SDRAM
(7)	SDCard slot

Bibliography

[ALI01] ALICE COLLABORATION, **2001**. *ALICE Technical Design Report of the Transition Radiation Detector.* Tech. Rep. CERN/LHCC 2001-021, CERN, Geneva.

[ALI05] ALICE COLLABORATION, **2005**. *ALICE Technical Design Report of the Computing.* Tech. Rep. CERN/LHCC 2005-018, CERN, Genf.

[Alm05] ALME, J., **2005**. *Digital module requirement specification.*

[Alt02a] ALTERA CORPORATION, **2002**. *Avalon Bus Specification.* Version 1.2.

[Alt02b] ALTERA CORPORATION, **2002**. *Excalibur Hardware Reference Manual.* Version 3.1.

[CER04] CERN, **2004**. *Layout of the ALICE detector (CERN-Poster-2004-004).* URL `http://doc.cern.ch//archive/electronic/cern/others/multimedia/poster/poster-2004-004.pdf`.

[CMMT04] CHRISTIANSEN, J., MARCHIORO, A., MOREIRA, P. and TOIFL, T., **2004**. *TTCrx Reference Manual.*

[dC] DE CUVELAND, J. *Online Track Reconstruction of the* ALICE *Transition Radiation Detector at* LHC *(CERN).* Ph.D. thesis, University of Heidelberg, Kirchhoff-Institut of Physics, Heidelberg. Publication planned.

[dC03] DE CUVELAND, J., **2003**. *Entwicklung der globalen Spurrekonstruktionseinheit für den ALICE-Übergangsstrahlungsdetektor am LHC (CERN).* Diplomarbeit, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg.

[dCR07] DE CUVELAND, J. and RETTIG, F., **2007**. *ALICE TRD DAQ Format.*

[DJVdV07] DIVIÀ, R., JOVANOVIC, P. and VAN DE VYVRE, P., **2007**. *Data Format over the ALICE DDL.*

[Gut02] GUTFLEISCH, M., **2002**. *Digitales Frontend und Preprozessor im TRAP1-Chip des TRD-Triggers für das ALICE-Experiment am LHC (CERN).* Diplomarbeit, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg.

[JGM] J GORBAN, I. M. and MARKOVIC, T. *Opencores: uart16550.* URL `http://www.opencores.org/projects.cgi/web/uart16550/overview`.

[JJ06] JOVANOVIC, P. and JUSKO, A., **2006**. *Theory of errors....*

[Jov04] JOVANOVIC, P., **2004**. *Central Trigger Processor - Preliminary Design Review.*

[MS86] MATSUI, T. and SATZ, H., **Oct. 1986**. *J/ψ suppression by quark-gluon plasma formation*. *Physics Letters B*, 178:416–422.

[Oya06] OYAMA, KEN, **Dec. 2006**. *Homepage of the* TRD *pre-trigger System*. URL `https://alice.physi.uni-heidelberg.de/oyama/PreTrigger/pkwk/index.php?TRDPre-TriggerSystem`.

[PRSZ01] POVH, B., RITH, K., SCHOLZ, C. and ZENTSCHE, F., **2001**. *Teilchen und Kerne: eine Einführung in die physikalischen Konzepte*. Springer, Berlin ; Heidelberg ; [u.a.], 5., korrigierte u. erw. Aufl. ed. ISBN 3-540-65928-5.

[Ret07] RETTIG, F., **2007**. *Entwicklung der optischen Auslesekette für den ALICE-Übergangsstrahlungsdetektor am LHC (CERN)*. Diplomarbeit, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg.

[RG07] RUBIN G, S. C., **Jun. 2007**. *ALICE DETECTOR DATA LINK: Hardware Guide for the Front-End Designers*. Tech. rep.

[Sat03] SATZ, H., **2003**. *Limits of confinement: The first 15 years of ultra- relativistic heavy ion studies*. *Nucl. Phys. A*, 715:3–19.

[Sch07] SCHUH, M., **2007**. *Entwicklung und Implementierung eines Steuersystems für die Trigger- und Datenausleselogik des ALICE-Übergangsstrahlungsdetektors am LHC (CERN)*. Diplomarbeit, Universität Heidelberg, Kirchhoff-Institut für Physik, Heidelberg.

[Stö00] STÖCKER, H., **2000**. *Taschenbuch der Physik*. 4., korrigierte Aufl.,. Thun; Frankfurt am Main. ISBN 3-8171-1628-4.

[Xil04] XILINX INC., **2004**. *XAPP058: In-System Configuration Using an Embedded Microcontroller*. XAPP058, Version 3.1.

[Xil06a] XILINX INC., **2006**. *LogiCore FIFO Generator User Guide*. UG175, Version 3.2.

[Xil06b] XILINX INC., **2006**. *Virtex-4 Configuration Guide*. UG071, Version 1.4.

[Y^+06] YAO, W.-M. ET AL., **2006**. *Review of Particle Physics*. *Journal of Physics G*, 33:1+. URL `http://pdg.lbl.gov`

Erklärung zur selbständigen Verfassung

Ich versichere, dass ich die vorliegende Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, im September 2007

Stefan Kirsch