George Victor Andrei

The Data Path of the ATLAS Level-1 Calorimeter Trigger PreProcessor

Dissertation

HD-KIP 10-67

Dissertation
submitted to the
Joint Faculties for Natural Sciences and Mathematics
of the Ruperto Carola University of
Heidelberg, Germany,
for the degree of
Doctor of Natural Sciences

presented by

Dipl.Phys.: George Victor Andrei
born in: Rosiori de Vede, Romania

Heidelberg, October 27, 2010

The Data Path of the ATLAS Level-1 Calorimeter Trigger PreProcessor

Gutachter: Prof. Dr. Karlheinz Meier
Prof. Dr. Reinhard Männer

There's no problem, only solutions.

J.L.

ZUSAMMENFASSUNG

Der Pre-Prozessor im "ATLAS Level-1 Calorimeter Trigger" liefert digitale Werte für transversale Energie in Echtzeit an nachfolgende Prozessoren, die physikalische Reaktionsprodukte erkennen sollen. Der Eingang besteht aus mehr als 7000 analogen Signalen von Zellen reduzierter Granularität in den Kalorimetern des ATLAS-Detektors. Die "Level-1 Trigger"-Entscheidung muß überprüfbar sein. Dazu werden vom Prozessor Kopien digitalisierter Echtzeit-Daten an die ATLAS Datenaufzeichnung gereicht. Zusätzlich stellt das Pre-Prozessor System mit dem standartisierten VME-Bus eine Schnittstelle zur Computer-Infrastruktur des Experiments zur Verfügung, worüber Konfigurationsdaten geladen und Kontrol- bzw. Monitor-Daten ausgelesen werden.

Ein zweckorientiertes System, welches sowohl den Transfer zur Aufzeichnung von Ereignisdaten in ATLAS als auch Datenaustausch über VME gewährleistet, wurde auf den 124 Modulen des Pre-Prozessor Systems in Form des "Readout Managers" implementiert. Das "Field-Programmable-Gate-Array (FPGA)" findet sich auf jedem der Module. Der erste Teil dieser Arbeit beschreibt die Algorithmen, die entwickelt wurden, um die Funktionalität des "Readout Managers" zu erfüllen. Der zweite Teil behandelt die Tests, welche durchgeführt wurden, um eine korrekte Funktion der Module sicherzustellen bevor sie bei CERN in der ATLAS-Kaverne installiert wurden.

ABSTRACT

The PreProcessor of the ATLAS Level-1 Calorimeter Trigger provides digital values of transverse energy in real-time to the subsequent object-finding processors. The input comprises more than 7000 analogue signals of reduced granularity from the calorimeters of the ATLAS detector. The Level-1 trigger decision must be verified. For this, the PreProcessor transmits copies of the real-time digital data to the Data Acquisition (DAQ) system. In addition, the PreProcessor system provides a standard VMEbus interface to the computing infrastructure of the experiment, on which configuration data is loaded and control or monitoring data are read out.

A dedicated system that ensures both the transfer of event data to storage in ATLAS and the data transfer over the VME was implemented on the 124 modules of the PreProcessor system in the form of a "Readout Manager". The "Field Programmable Gate Array" (FPGA) is located on each module. The first part of this work describes the algorithms developed to meet the functionality of the Readout Manager. The second part deals with the tests that were carried out to ensure the proper functionality of the modules before they were installed at CERN in the ATLAS cavern.

Contents

Chapter 1

Introduction

The Large Hadron Collider (LHC) is designed to accelerate and collide protons with a centre-of-mass energy of $\sqrt{s} = 14$ TeV, at a luminosity of $\mathscr{L} = 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$. The unprecedented high energy and luminosity will allow to study the existence of the Higgs boson, the particle believed to be at the origin of the spontaneous symmetry-breaking mechanism in the electroweak sector of the Standard Model, as well as to probe the theories beyond the Standard Model. ATLAS is one of the four major experiments installed at the LHC. It is a general-purpose experiment designed to fully exploit the physics opportunities offered by the LHC.

At the LHC's design luminosity, the expected event rate is in the order of 1 GHz. This raises an important experimental challenge for the ATLAS detector. The interesting physics processes occur at very small rates, in the order of a few Hz or even lower, being overwhelmed by the production of jets coming from QCD interactions. On the other hand, the raw data produced by the detector amounts to about 1 PByte/s, while the event data recording is limited by technology and resources to about 300 MByte/s. Therefore, the ATLAS Trigger and Data Acquisition systems have to reduce the initial event rate to the affordable mass storage rate, while efficiently selecting the potential rare physics events. To achieve this, the ATLAS Trigger system is organised into three levels of event selection.

The PreProcessor system is the first stage of data processing in the Level-1 Calorimeter Trigger, a major subcomponent of the first level of event selection. It receives about 7200 analogue trigger signals that describe transverse energy deposits in the ATLAS calorimeter subdetectors. The main task of the PreProcessor is to extract a corresponding digital energy value from each pulse and assign it to a specific proton-proton collision. Additionally, in order to allow calibration, monitoring and verifications of the Level-1 trigger system, the PreProcessor has to provide copies of the digital trigger data related with the accepted event to the Data Acquisition system. The PreProcessor is a VME-based system. It mainly consists of 124 PreProcessor Modules, each of which can process 64 trigger signals. The task of transferring the event related data to the Data Acquisition system is assigned to a Readout Manager FPGA located on each PreProcessor Module. The device collects the event data from distributed locations on the board, and sends it in a predefined format to the Data Acquisition system. Apart from this, the device has the tasks to transfer configuration data from the VME interface to various on-board programmable locations, and to collect trigger-independent monitoring data, accumulated in the hardware of

the PreProcessor Module, and provide it to VME. The first part of this thesis describes the algorithms developed to sustain the functionality of the Readout Manager FPGA. The second part of the thesis presents the tests that had been carried out to verify the proper functionality of the PreProcessor Modules, before they were installed at CERN in the electronics cavern of the experiment.

Chapter 2 provides a brief description of the Standard Model theory and of the search for the Higgs boson. The LHC machine and the ATLAS detector are described in chapter 3. An overview of the ATLAS Trigger and Data Acquisition systems is given in chapter 4, while the architecture and algorithms of the Level-1 Calorimeter Trigger are presented in more detail in chapter 5. The next two chapters form the core of this thesis. Chapter 6 is entirely dedicated to the Readout Manager FPGA, presenting in great detail the functionality of the device and the implemented algorithms, while chapter 7 describes the functional tests of the PreProcessor Module and shows examples of misbehaviours detected during the tests. Lastly, chapter 8 shows results recorded during the commissioning of the PreProcessor system at CERN, in standalone mode with the other trigger components and in combined runs with the whole ATLAS detector, and results obtained with the first LHC proton beam data.

Chapter 2

Physics Motivation

2.1 The Standard Model of Particle Physics

The Standard Model describes the physical world in terms of two kinds of particles and three fundamental forces. The particles are the *elementary fermions* and the *gauge bosons*. The fermions are half-integer spin particles that make up the matter, while the gauge bosons are integer spin particles that mediate the interactions between the fermions. The fermions are classified into two categories, *leptons* and *quarks*, according to the interactions they experience. There are six *flavours* of leptons, i.e. the electron (e), the muon(μ), the tau (*tau*) and three neutrino partners $(\nu_e, \nu_\mu, \nu_\tau)$, and six flavours of quarks, i.e. up (u), down (d), charm (c), strange (s), top (t) and bottom (b). These 12 matter particles form together three *generations*, each of which contains one pair of leptons and one pair of quarks, as shown in table 2.1. The fermions of the first generation are stable and they represent the elementary constituents of the ordinary matter. The fermions of the other two generations appear in cosmic rays or they are manufactured in high-energy experiments, and they eventually decay into particles of the first generation. Also, for each fermion there exists an *antiparticle* of the same mass and opposite electric charge.

The three fundamental forces described by the Standard Model are: the *strong* force, which is responsible for holding the quarks together, the *electromagnetic* force, which acts on all electrically charged particles and which is responsible for the bonding in chemical elements, and the *weak* force, which determines the transmutation of quarks and leptons. Attempts to include the fourth known fundamental force of nature, the *gravitation*, in the framework of the Standard Model have so far been unsuccessful. However, the gravitation is considered to be inconsequential at the subatomic scale because it is much weaker than the other three forces. The properties of the four fundamental forces are summarised in table 2.2.

The Standard Model of particle physics is formulated as a relativistic quantum gauge field theory. Each of the three fundamental forces is described as resulting from the exchange of spin-1 gauge bosons between the matter fermions. The gauge bosons arise from the requirement of a local gauge symmetry, i.e. the Lagrangian of the system has to be invariant under different transformations applied at each point in space and time. The field theory of the strong interaction is formulated as a non-abelian gauge theory with $SU(3)_C$ *color* symmetry, and it is called quantum

Fermions		Generation		Electric
	1	2	3	Charge
Leptons	e	μ	τ	-1
	ν_e	ν_μ	ν_τ	0
Quarks	u	c	t	+2/3
	d	s	b	-1/3

Table 2.1: The three generations of matter fermions.

chromodynamics (QCD). The strong force is mediated by eight[1] massless gauge bosons called *gluons*, which, like quarks, carry color charge. The electromagnetic interaction is described as an abelian gauge theory with the symmetry group $U(1)$. The respective theory is called quantum electrodynamics (QED). The mediator of the electromagnetic force is the *photon*, an electrically neutral and massless gauge boson. The mathematical formulation of the weak interaction is based on the gauge group $SU(2)$. The weak force is mediated by three massive gauge bosons, i.e. W^+, W^- and the neutral Z. In the 1960s, Glashow, Salam and Weinberg unified the description of the electromagnetic and weak interactions into one field theory called the electroweak theory. This is a non-abelian gauge theory based on the $SU(2)_L \, x \, U(1)_Y$ gauge symmetry group of the weak isospin (I) and the weak hypercharge (Y) [Mor04].

2.1.1   The Higgs Mechanism

In the 1960s, many theoretical researches were focused on formulating a relativistic field theory with local gauge symmetry that would describe both the electromagnetic and the weak interactions. The major difficulty that the physicists were trying to overcome was that in order not to destroy the gauge invariance the fermions and gauge bosons had to be massless, which was in contradiction with the experimental observations. Of a particular interest was the case of the weak interaction, which was known to be short-range, its influence being significant only up to 10^{-18} m. Consequently, the carriers of the weak force had to be fairly massive rather than massless.

In 1961, Glashow proposed the first $SU(2) \, x \, U(1)$ gauge theoretical model for the electroweak interaction [Gla61]. The theory possesses only a *partial symmetry*, meaning that the symmetry is broken by the addition of explicit mass terms in the Lagrangian. The theory is also notable for introducing for the first time the *neutral current*[2] carried by the Z boson, in addition to the then already known charged weak currents ($W^\pm$) and electromagnetic current (γ). The breakthrough came in 1964 when Higgs [Hig64], Englert and Brout [Eng64], and Guralnik, Hagen, and Kibble [Gur64] showed that masses can be generated dynamically by *spontaneous symmetry breaking*. The latter occurs when a system that is symmetric with respect to some

---

[1]the number of gauge bosons is given by the number of generators of the unitary group associated with each force. For a special unitary group of degree *n*, i.e. *SU(n)*, the number of generators is $n^2 - 1$.

[2]i.e. the interacting particles do not change their charges

Force	Relative Strength	Range (metres)	Mediators (gauge bosons)	Dynamic Theory
Strong	1	10^{-15}	8 Gluons (g)	Quantum Chromodynamics
Electromagnetic	10^{-2}	∞	Photon (γ)	Electroweak Theory
Weak	10^{-5}	10^{-18}	$W^{\pm}$, Z	"-"
Gravitation	10^{-38}	∞	Graviton (G)	General Theory of Relativity

Table 2.2: Summary of the properties of the four fundamental forces in nature. The gravitation is not incorporated in the Standard Model. The mediator of this force, called the *graviton*, is a massless, spin-2 particle [Mor04].

symmetry groups goes into a vacuum state that is not symmetric or not invariant. This framework was then applied by Weinberg [Wei67] and Salam [Sal68] to the electroweak theoretical model formulated by Glashow in 1961. In their work, Weinberg and Salam introduced an $SU(2)$ self-interacting doublet of complex scalar fields, with four degrees of freedom, of which neutral component does not vanish in vacuum, i.e. it has a non-zero value in its lowest energy state. This non-zero *vacuum expectation value* breaks spontaneously the electroweak $SU(2) \, x \, U(1)$ symmetry, but the Lagrangian of the system remains invariant under $SU(2) \, x \, U(1)$ transformations. Three of the four degrees of freedom of the doublet scalar field are *swallowed* by the $W^{\pm}$ and Z bosons, which in this way acquire mass. The remaining degree of freedom of the doublet corresponds to a spin-0 scalar particle, which was latter named as the *Higgs boson*. The fermion masses are also a consequence of the electroweak symmetry breaking. The same theory postulates that the doublet scalar field couples to fermions through Yukawa interactions. The mechanism through which $W^{\pm}$ and Z bosons acquire mass is referred to as the *Higgs mechanism*. A detailed account of the mathematical formalism that describes the Higgs mechanism can be found in e.g. [Djo08].

The Higgs boson is so far the only particle of the Standard Model of which existence has not been confirmed experimentally. However, experimental results achieved in the last forty years for the electroweak sector of the Standard Model provide strong support for the hypothesis of the Higgs mechanism. Of the most notable are the first observation of neutral current interactions in the Gargamelle bubble chamber detector at CERN (1973), the direct observation of the W and Z bosons by the UA1 and UA2 experiments (1983), both also located at CERN, the discovery of the top quark, which decays only through the weak force, by the CDF and DØ experiments at Fermilab (1995), or the high precision measurements of the mass of W boson and top quark by experiments at LEP[3] and Tevatron colliders.
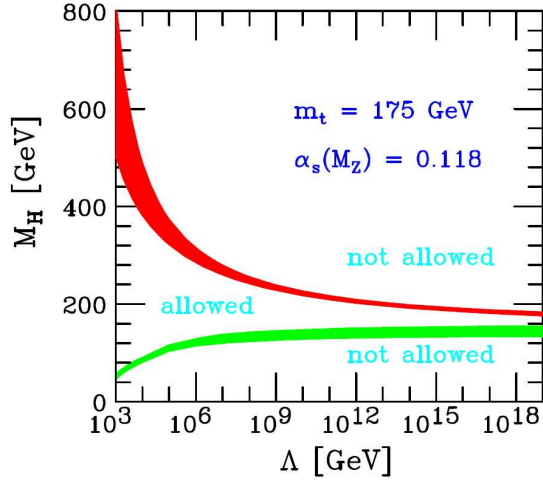
Figure 2.1: Triviality and vacuum stability bounds on m_H as a function of new physics scale Λ [Djo08].
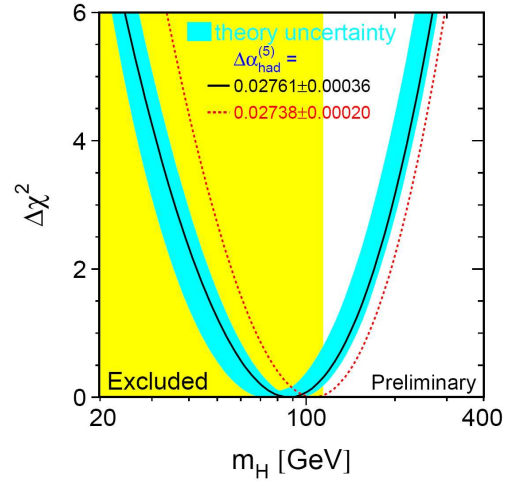
Figure 2.2: The $\Delta\chi^2 = \chi^2 - \chi^2_{min}$ versus m_H from the global fit to the electroweak data [LEP02].

2.1.2 The Search for the Higgs Boson

The Higgs Boson Mass

In the electroweak theory, the mass of the Higgs boson (m_H) is given as:

$$m_H = \sqrt{\frac{\lambda}{2}}\, v \quad , \tag{2.1}$$

where λ is the Higgs self-coupling parameter, while v represents the vacuum expectation value of the Higgs scalar field. The latter is fixed by the Fermi coupling constant (G_F) to:

$$v = \sqrt{2}\, G_F = 246 \ \textit{GeV} \quad , \tag{2.2}$$

which means that m_H scales only with $\sqrt{\lambda}$. Because the Higgs boson has not been yet experimentally observed, the self-coupling λ remains unknown, therefore m_H cannot be predicted by the Standard Model. However, theoretical limits for m_H can be derived from assumptions on the energy scale Λ within which the Standard Model is valid and beyond which new physics phenomena emerge. If the electroweak theory is assumed to be valid in the whole energy range, then the self-coupling λ will vanish when $\Lambda \to \infty$. Which also means that the larger the energy scale Λ, the smaller the self-coupling λ. But since the non-zero vacuum expectation value in the Higgs mechanism occurs only if $\lambda \neq 0$, Λ has to be limited to a certain physical scale. Since m_H is proportional with λ, this limitation gives an upper bound on m_H (*triviality bound*). On the other hand, if λ becomes too small, i.e. smaller than the Higgs-top quark Yukawa coupling ($\lambda_t = \sqrt{2}m_t/v$), the Higgs vacuum state will be unstable. Therefore, λ has to be large enough to balance the top quark contribution, and this gives a lower bound on m_H (*vacuum stability*

---

[3]Large Electron Positron.

bound). Thus, based on these two requirements and on a give value of Λ, the minimum and a maximum Higgs mass allowed can be determined (see figure 2.1). If the Standard Model is consistent up to the Planck scale ($\Lambda \sim M_{Planck} = 10^{19}$ GeV), beyond which gravitation effect cannot be longer neglected in respect to the other fundamental interactions, one obtains that m_H should lie within the range:

$$130 \ GeV \lesssim m_H \lesssim 180 \ GeV \quad . \tag{2.3}$$

If new physics occurs at a lower mass scale, the bound on m_H becomes weaker. For example, if $\Lambda \sim TeV$ then one expects m_H within the range [Djo08]:

$$50 \ GeV \lesssim m_H \lesssim 800 \ GeV \quad . \tag{2.4}$$

Apart from these theoretical constraints, direct searches of the Higgs boson at LEP established a lower bound on the Higgs mass of 114.4 GeV at 95% confidence level (CL)[4] [ALE03], while similar investigations at Tevatron currently exclude, in addition, the mass range of 162 to 166 GeV at 95% CL[5] [Tev10]. Further experimental constraints on the mass range of the Higgs boson are determined indirectly from precision fits of all measured electroweak observables. Figure 2.2 shows the $\Delta \chi^2$ of the fit to the electroweak precision data from LEP and SLC[6] as a function of m_H. The shaded band around the central curve indicates the theoretical uncertainties due to unknown higher order corrections, while the vertical band indicates the 95% exclusion limit determined from direct searches. The Higgs mass obtained from the fit is $m_H = 87^{+34}_{-26}$ GeV. Also, taking into account the 114.4 GeV lower limit, an upper limit of $m_H \leq 185$ GeV at 95% CL was obtained. [ALE08].

The Higgs Boson at the LHC: Production and Decay Channels

The search for the Higgs boson is one of the major goals for the experiments at LHC. The proton-proton (*pp*) collisions at a centre-of-mass energy of $\sqrt{s} = 14$ TeV will allow to search for the Higgs boson in a mass range from 100 GeV to about 1 TeV, in a various production and decay channels.

Because the coupling strengths of the Higgs boson to fermions and bosons are proportional to their masses, the Higgs production generally involves heavy particles, i.e. the W and Z bosons, the top quark and to a lesser extent the bottom quark. At the LHC, there are four main Higgs production processes:

- *gluon-gluon (gg) fusion* ($gg \rightarrow H$): the Higgs boson couples to gluons through a heavy-quark loop;

---

[4]the investigations were mainly carried out through the process $e^+e^- \rightarrow Z^* \rightarrow ZH \rightarrow (f\bar{f} + b\bar{b}/t\bar{t})$, known as the *Higgsstrahlung*, based on collision data at centre-of-mass energies up to 209 GeV. The LEP collaborations reported an 1.7σ excess of events above the expected background, smaller than the 5σ needed to certify the discovery of the Higgs boson [ALE03]. The LEP accelerator ended its operation in November 2000.

[5]at Tevatron, the Higgs boson is searched in $p\bar{p}$ collisions at the centre-of-mass energy $\sqrt{s} = 1.96$ TeV via gluon fusion production and associated production with W/Z bosons and $t\bar{t}$ pairs.

[6]Stanford Linear Collider (e^+e^-).
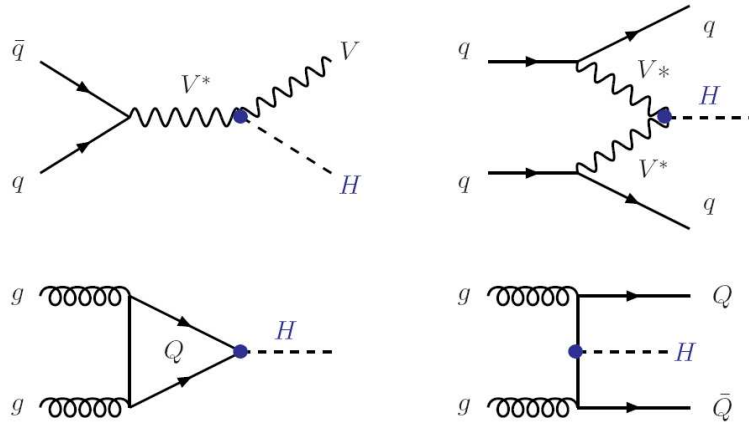
Figure 2.3: Feynman diagrams of the four main Higgs production mechanisms at LHC: associated production with W/Z (*upper left*), vector boson fusion (VBF) (*upper right*), gluon-gluon fusion (*lower left*) and associated production with heavy quarks (*lower right*) [Djo08].

- *vector boson fusion* (VBF) ($qq \rightarrow V^*V^* \rightarrow qq + H$): two incoming quarks, mostly light quarks, emit a pair of gauge bosons, which then fuse to form a Higgs boson;

- *associated production with W/Z* ($q\bar{q} \rightarrow V + H$): two incoming light quarks annihilate to a weak gauge boson, which then radiates a Higgs boson;

- *associated production with heavy quarks* ($gg \rightarrow QQ + H$): the process occurs mainly through gluon fusion, where the Higgs boson is emitted from both the external and internal quark lines.

The Feynman diagrams of these production processes are shown in figure 2.3, while the corresponding cross-sections are shown in figure 2.4a. The dominant production channel over the entire mass range will be the *gg* fusion, due to the large gluon content possessed by the high energy protons. The cross-section of the VBF process is roughly one order of magnitude smaller, but reaches the level of the *gg* fusion cross-section at very large M_H values. The other two processes have much smaller cross-sections, and they are relevant only in the mass range $m_H \lesssim 250$ GeV.

As for the production case, the coupling of the Higgs boson to mass favours the decay mode into the heaviest possible final state. This is also shown in figure 2.4b, which illustrates the branching ratios of the different decay modes of the Higgs boson. In the low mass region $m_H \lesssim 130$ GeV, the dominant decay mode is $H \rightarrow b\bar{b}$, followed by $H \rightarrow \tau\bar{\tau}/c\bar{c}$. Above $m_H \lesssim 180$ GeV, the Higgs boson almost exclusively decays into pairs of gauge bosons, i.e. $H \rightarrow W^+W^-$ and $H \rightarrow ZZ$, where one of the gauge bosons is virtual for $m_H < 2m_W(m_Z)$. Also, beyond $m_H > 2m_t$, the branching ratio of the decay mode $H \rightarrow t\bar{t}$ becomes significant, but it is still smaller than the branching ratios of the decay modes to W^+W^- and ZZ, due to the different dependence of the Higgs coupling with the mass scale, i.e. linear versus cubic. Although the Higgs boson does not couple to gluons or photons, its decay into these massless particles is also possible. The decay mode $H \rightarrow gg$ is mediated by heavy-quark loops and it has a significant branching ratio only in
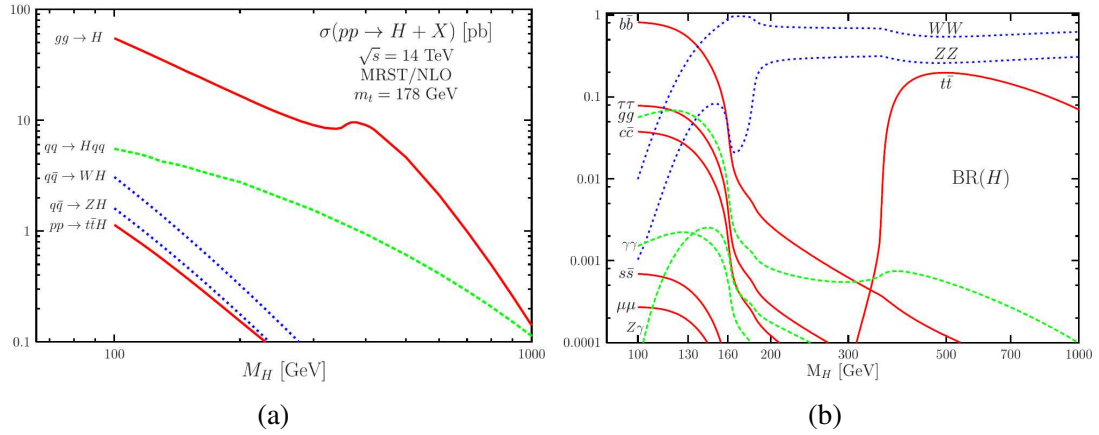
Figure 2.4: Cross-sections of the main Higgs boson production mechanisms at the LHC energies of $\sqrt{s} = 14$ TeV (*a*) and branching ratios of the main Higgs boson decays (*b*) as a function of the Higgs mass.
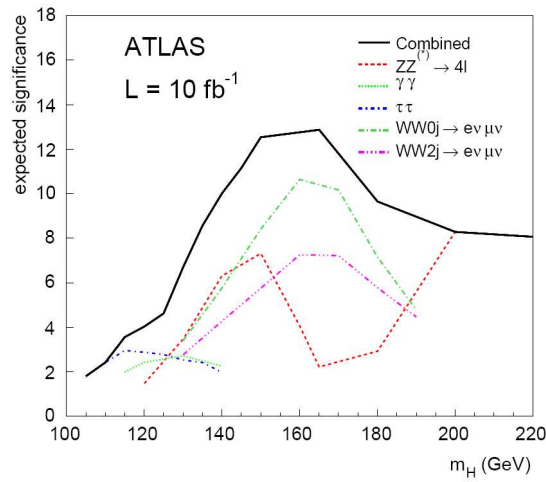
Figure 2.5: Expected statistical significance (expressed in σ's) for the Higgs boson in various channels in ATLAS as a function of the Higgs mass.

the low mass region, while the decay mode $H \rightarrow \gamma\gamma$ is very rare and it is mediated by charged fermion and W boson loops [Djo08].

Figure 2.5 shows the discovery potential for the Higgs boson in various production and decay channels, in the case of the ATLAS experiment. The highest sensitivity is given by the decays to weak gauge bosons. The W^+W^- mode dominates in the mass range of 140 to 200 GeV, while the ZZ decay to four isolated leptons (e/μ) gives the highest sensitivity above 200 GeV. When all channels are combined together, a 5σ discovery of the Higgs boson is possible with an integrated luminosity of 10 fb^{-1}, for m_H between the LEP limit of 114.4 GeV and ≈ 1 TeV, after the detector performance and the background systematics are understood.

Chapter 3

The ATLAS Experiment at the LHC

ATLAS is one of the four major experiments built on the Large Hadron Collider (LHC) accelerator ring. It is a general-purpose experiment for pp collisions, designed to observe the largest spectrum of physics processes expected from the LHC. This chapter gives a brief overview of the LHC machine and of the ATLAS experimental apparatus.

3.1   The Large Hadron Collider

The LHC is a circular particle accelerator located at the European Organisation for Nuclear Research (CERN[1]) near Geneva, Switzerland. It is installed in the 26.7 km long tunnel that formerly housed the LEP collider, and it is primarily designed to collide head-on two counter-rotating beams of protons with a centre-of-mass energy of $\sqrt{s} = 14$ TeV, at an unprecedented luminosity of $10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$. As a secondary program of running, the LHC will also collide two beams of heavy ions[2] with a total centre-of-mass energy of $\sqrt{s} = 1.15$ PeV and a peak luminosity of $10^{27}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ [Eva08]. The collisions occur in four interaction points, where four physics experiments are situated (see also figure 3.1):

- ATLAS (A Toroidal LHC ApparatuS) [ATL08a], designed to mainly investigate physics processes with high momentum transfer that are produced by pp collisions;

- CMS (Compact Muon Solenoid) [CMS08], also a general-purpose detector for pp collisions, designed to explore the physics at Terascale;

- LHCb (LHC beauty experiment) [LHC08a], designed for precision measurements of CP violation and rare decays of B-mesons;

- ALICE (A Large Ion Collider Experiment) [ALI08], dedicated to investigations of the quark-gluon plasma in heavy ion collisions.

Two other experiments, smaller in size, are located on either side of the ATLAS and CMS detectors respectively:

---

[1] Conseil Europeen pour la Recherche Nucleaire (fr.)
[2] i.e. $^{208}_{82}$Pb (lead) ions

- LHCf (LHC forward) [LHC08b], dedicated to the measurement of neutral particles emitted in the very forward region of the LHC collisions;

- TOTEM (Total Cross Section, Elastic Scattering and Diffraction Dissociation) [TOT08], which aims to measure the total *pp* cross-section and to study elastic and diffractive scattering at the LHC.

3.1.1   Machine Parameters

The LHC is supplied with protons or heavy ions from an injector chain that mainly comprises a linear accelerator (LINAC2), a Proton Synchrotron Booster (PSB), a Proton Synchrotron (PS) and a Super Proton Synchrotron (SPS) (see again figure 3.1). The injector chain produces and pre-accelerates the protons up to the kinetic energy of 450 GeV. The LHC uses superconducting radio-frequency cavities operated at 400.8 MHz to capture and further accelerate these particles up to 7 TeV/beam, the highest energy ever reached in a particle accelerator. The trajectories of these very energetic protons are bent in the curved sections of the accelerator ring by 8.34 T magnets. The respective magnetic field is produced by 1232 niobium-titanium superconducting dipole magnets, which are cooled to 1.9 K with superfluid helium[3]. Additionally, in the straight sections of the ring, 386 quadrupole magnets and several thousand corrector magnets are used to focus the beams and to maximise the luminosity at the collision points.

The machine luminosity is a useful measure of the accelerator performance. In addition, the luminosity is an important factor needed to ensure that interesting physics processes, which have a small cross-section, are produced at the highest possible rate. The machine luminosity can be approximated as:

$$\mathscr{L} = \frac{N^2 \cdot n_b \cdot f_{rev}}{A_{eff}} \quad [cm^{-2}s^{-1}] \quad , \tag{3.1}$$

where N is the number of particles per bunch, n_b the number of bunches per beam, f_{rev} the particle revolution frequency (11,246 kHz), while $A_{eff} = 4\pi\sigma_x\sigma_y$ is the effective interaction area, with σ_x and σ_y representing the Gaussian transverse profile of the bunches in the vertical and horizontal directions. For a given physics process with a known cross-section σ_{event}, the number of related events generated each second in the LHC collisions is:

$$N_{event} = \sigma_{event} \cdot \mathscr{L} \quad . \tag{3.2}$$

The proton beams are organised in *bunches*, each containing up to 1.15 x 10^{11} particles. The bunches have a longitudinal spread of about 7.7 cm, and they are spaced apart by 7.48 m [Brü04]. The latter parameter determines a bunch-crossing rate of 40.08 MHz and a total number of 3564 possible bunches in the circumference of the LHC ring. However, due to technical constraints imposed by the operation of the injection, acceleration and dumping systems, only 2808/3564 bunches can be filled per LHC ring [Bai03]. The RMS beam size parameters, i.e. σ_x and σ_y, will be tuned to about 16.7 μm for *pp* collisions in the centre of

---

[3]for comparison, other large accelerators like Tevatron (Fermilab) or HERA (DESY) use NbTi superconducting magnets cooled with normal liquid helium at temperatures slightly above 4.2 K, which allows to reach a magnetic field of around 5 T.
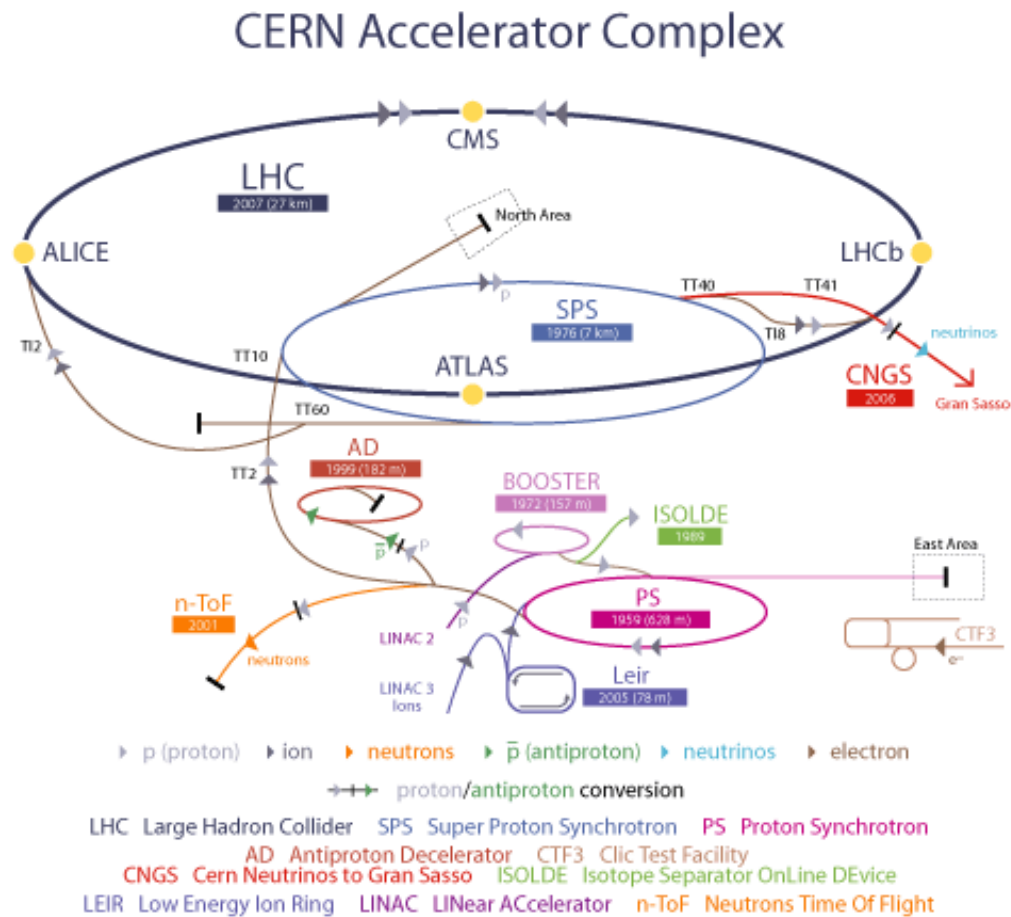
Figure 3.1: The Large Hadron Collider and its four main experiments [CER10c].

the ATLAS and CMS detectors, in order to determine a peak value of the luminosity in these interaction points of $10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$. For *pp* collisions at LHCb, the beam size will be enlarged by tuning the same parameters to approximatively 70.9 μm, in order to determine a peak luminosity of $10^{32}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ [Brü04]. The LHCb needs this lower luminosity in order to enable a high-precision reconstruction of the primary interaction vertex and of the secondary B-meson decay.

As a heavy ion collider, the LHC will accelerate the ions from the injection energy of 177.4 GeV/nucleon up to 2.76 TeV/nucleon, which determines the total collision energy of 1.15 PeV. In this operating mode, the bunches will contain up to 7 x 10^7 ions. Also, the nominal filling scheme will be based on 100 ns bunch spacing, and only 592 out of 891 possible bunch positions in the LHC will be filled with ions [Bai03]. This will lead to a peak luminosity of 10^{27} cm^{-2} s^{-1} for heavy ion collisions.

3.1.2   Experimental Challenges at the LHC

The total inelastic *pp* cross-section at $\sqrt{s} = 14$ TeV is predicted to be approximatively 79 mb [Sjo06]. According to equation 3.2, the LHC is then expected to produce a total rate of about 10^9 inelastic events per second at design luminosity. This means that at each bunch-crossing, i.e. $\sim$25 ns, a mean of 25 *pp* interactions are expected to occur[4]. The high event rate and the high energy of the colliding protons impose several experimental challenges, which set stringent requirements on the different techniques and technologies employed by the LHC detectors:

- **Pile up**. The inelastic events produced at LHC can be divided into two categories: *minimum bias* and *hard-scattering* events. The minimum bias events arise from strong long-range interactions between the constituents of the colliding protons. They are characterised by a small momentum transfer, such that the final state particles have large longitudinal momentum (p_L) and small transverse momentum (p_T), i.e. $\simeq$500 MeV. The hard-scattering events are produced via strong short-range *pp* interactions. These events are characterised by a large momentum transfer, which can lead to the production of heavy particles. The high p_T processes are relatively rare with respect to the minimum bias events, but they represent the main interests for the physics researches at LHC. At each bunch-crossing, about 23 low p_T events ($\sigma_{mb} \sim$69 mb) are produced simultaneously, overlapping the high p_T physics events of interest. Furthermore, the same low p_T events generate on average 1700 charged particles. This means that a response time of the detectors longer than 25 ns can lead to an overlap of signals from multiple consecutive bunch-crossings. These effects are collectively referred to as *event pile-up*.
  The pile-up of minimum bias events degrades the accuracy of the energy measurement, which consequently worsens the selection efficiency of the interesting high p_T objects. In order to be able to handle the particle fluxes and to reduce the influence of pile-up events, the LHC detectors must be segmented into finely granular readout cells. Additionally, the detectors must feature a fast response, i.e. typically $\leq$50 ns, in order to integrate the readout signals over a reduced number of bunch-crossing in each channel.

- **QCD background**. Another challenge for the experiments at LHC is the high production rate of QCD jets. These represent highly collimated cones of particles, which are generated in high p_T scatterings due to strong partonic interactions. Because of the large cross-sections for these processes, QCD jets are produced abundantly, overwhelming the rarer processes being sought at the LHC (*QCD background*). Figure 3.2 shows the cross-sections for various inelastic processes produced in both *pp* and *p$\bar{p}$* collisions, as a function of the centre-of-mass energy. It can be observed, for example, that the cross-section for jets with $p_T > 100$ GeV, at $\sqrt{s} = 14$ TeV, is about five orders of magnitude larger than the cross-section for the Higgs boson with a mass of 150 GeV. Since one of the dominant decay modes for the Higgs in that mass range is into two b-quark jets, the experimental signatures are difficult to isolate because of the QCD background. Instead, final states involving leptons or photons, or missing transverse energy (E_T^{miss}) or secondary vertices

---

[4]this takes also into account that only 2808 out of 3564 possible bunch positions are filled with protons.
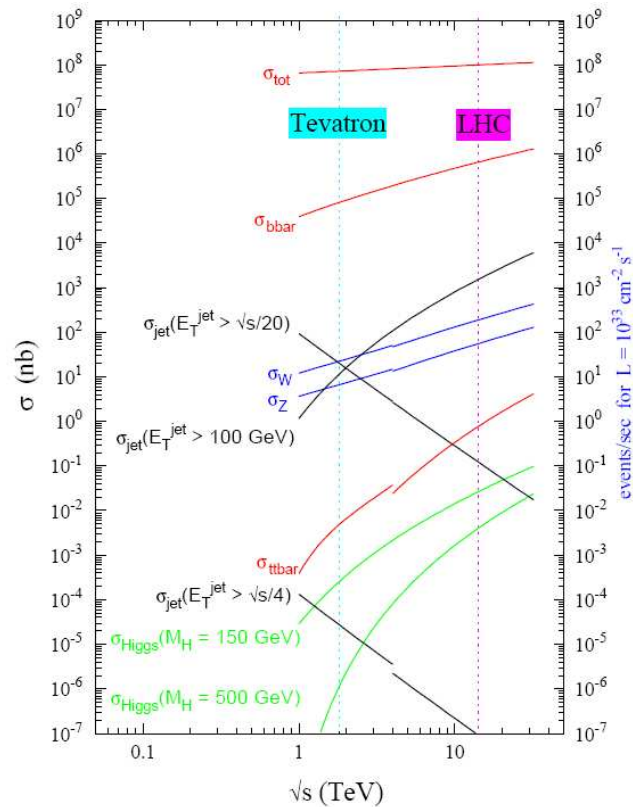
Figure 3.2: Production cross-sections and event rates for different processes as a function of the centre-of-mass energy [Gia04].

information have to be used, in order to suppress the background generated by QCD processes. For an efficient extraction of the rare signals of new physics from the large background of known processes, the experiments at LHC must feature excellent detector and trigger performance in terms of particle identification capabilities and energy resolution.

- **Radiation level**. Because the flux of particles generated in *pp* collisions is very large, the detectors at LHC are expected to operate in a high radiation environment. The highest radiation levels are expected to occur in the forward regions of the detectors, which will be exposed in ten years of LHC operation to a neutron fluence of up to 10^{17} neutrons/cm^2 and a γ-dose of about 10^7 Gy[5]. Thus, in order to avoid severe radiation damages, the detection material and the on-detector readout electronics must be radiation-tolerant. Also, the high radiation levels require the LHC detectors to operate reliably in long term. In case of failures, the access for maintenance in the experimental hall will be highly restricted and time consuming, which will lead to a long detector shut-down period.

---

[5]the Gray (Gy) is a radiological unit, which expresses the absorbed energy per unit mass of material (1 Gy = 1 Joule/kg).

3.2   The ATLAS Detector

ATLAS is a general-purpose detector for *pp* collisions at the LHC. The detector was mainly designed to search for new particles related to the electroweak symmetry breaking mechanism. In consequence, the detector optimisation was also guided by physics issues such as the sensitivity to the largest possible Higgs mass range. Other important objectives of the ATLAS physics program refer to precise measurements of known particle properties (e.g. $W^{\pm}$ bosons, t- and b-quark mesons) and the study of Standard Model (SM) processes at the TeV scale, as well as to investigations of the theories beyond the SM.

The necessary detection capabilities for a large spectrum of experimental signatures lead to the following set of design requirements for the ATLAS detector:

- efficient tracking and vertex reconstruction near the interaction point to measure the momentum of charged particles and to identify b-quarks and τ-decays;

- very good electromagnetic calorimetry, in terms of energy resolution and solid angle coverage, for electron and photon identification and measurements;

- hermetic hadronic calorimetry for accurate determination of the jet energy and reconstruction of the E_T^{miss};

- good muon identification and momentum resolution over a wide range of momenta, and the ability to determine unambiguously the charge of the high p_T muons;

- fast and highly selective trigger system, to reduce the initial event rate (10^9 Hz) to a level that can be handled offline ($\sim$200 Hz), while selecting efficiently the interesting physics processes;

- fast and radiation-tolerant electronics and sensor elements, and high detector granularity to handle the particle fluxes and to reduce the influence of overlapping events.

Figure 3.3 shows the layout of the ATLAS detector. It has a total diameter of 25 m, a total length of 46 m and it weighs about 7000 t, which makes it the largest of the four main detectors installed at the LHC. The detector is constructed with rotation-symmetry around the beam axis, and it consists of three complementary sub-detector systems assembled in concentric layers: an *Inner Detector*, which is immersed in a solenoidal magnetic field, an *Electromagnetic* and a *Hadronic Calorimeter* surrounding the Inner Detector, and at the outer region a *Muon Spectrometer* consisting of air core toroids with muon chambers. The following sections of the chapter give a brief overview of the different sub-detectors. A more detailed description of the ATLAS subsystems can be found in [ATL08a].

3.2.1   The Coordinate System

The coordinate system of the ATLAS experiment is a right-handed system, with the x-axis pointing radially towards the centre of the LHC ring, the z-axis following the beam direction[6], and

---

[6]the positive z-axis points to the LHCb detector (see also figure 3.1).
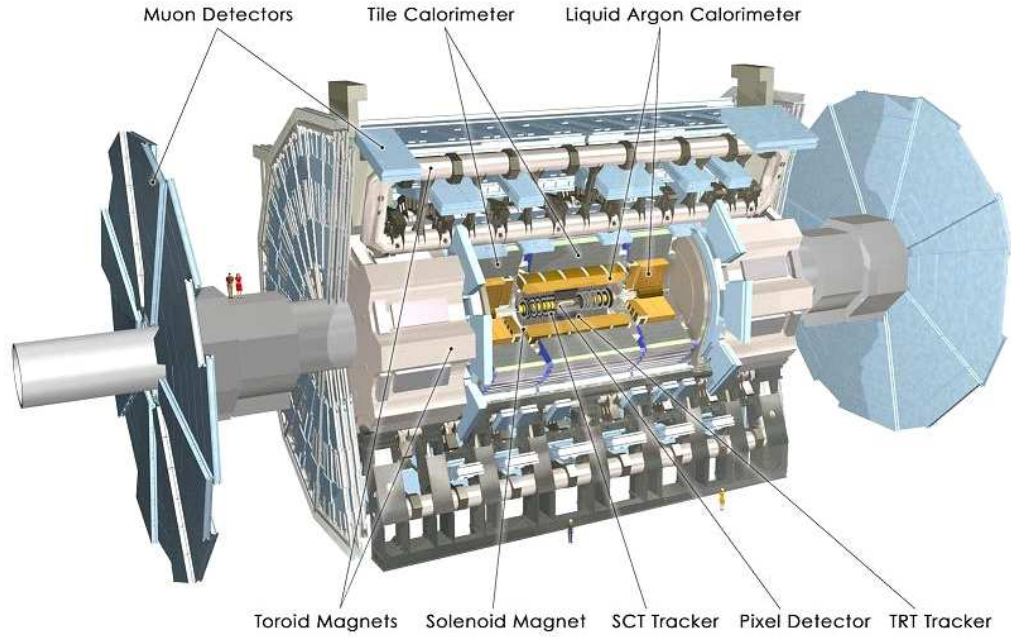
Figure 3.3: Cut-away view of the ATLAS detector [CER10a].

the y-axis pointing vertically upwards. The origin of the coordinate system is the nominal inter-action point in the centre of the detector. The azimuthal angle $\phi \in [0,2\pi]$ is measured around the beam axis, in the transverse plane formed by the x and y directions, so that the angle $\phi=0$ corresponds to the positive x-axis while $\phi=\pi/2$ corresponds to the positive y-axis.

The polar angle θ is measured from the beam axis with the positive z-axis:

$$\theta = arctan\left(\frac{p_T}{p_z}\right) \quad , \tag{3.3}$$

where p_T and p_z represent the perpendicular and parallel components of the momentum to the z-axis. The rapidity variable (Y) is defined as:

$$y = \frac{1}{2}\left(\frac{E+p_z}{E-p_z}\right) \quad . \tag{3.4}$$

The difference in rapidity between two particles is Lorentz invariant under a boosting along the z-axis. When the mass and the momentum of the particle are not known, the pseudorapidity variable (η) it is used to characterise the detected particle:

$$\eta = -ln\left[tan\left(\frac{\theta}{2}\right)\right] \quad , \tag{3.5}$$

In the massless limit ($p \gg m$), the pseudorapidity closely approximates rapidity. For the transverse plane ($\theta = 90°$) the pseudorapidity is zero, while for directions close to the beam axis ($\theta \to 0°$, $\theta \to 180°$) the pseudorapidity goes to infinity.
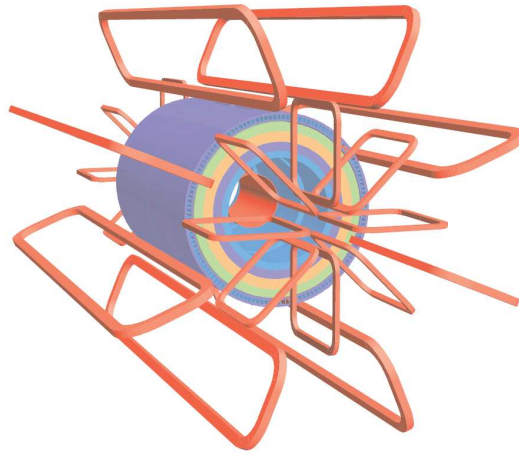
Figure 3.4:   The geometry of the ATLAS Magnet System. The solenoid magnet system is depicted inside
the calorimeter volume [ATL08a].

Three sides are defined for the ATLAS detector. The part along the negative z-axis is named as the C-side, the part along the positive z-axis is named as A-side, while B-side is the plane with $z = 0$.

3.2.2   The Magnet System

The ATLAS magnet system dominates the overall size of the ATLAS detector, being 22 m in diameter and 26 m in length. It consists of four large superconducting magnets used for particle identification and momentum measurements (see figure 3.4):

- a central *solenoid*, which provides the Inner Detector with a 2 T strong magnetic field along the beam axis;

- an outer system of three large air-core *toroids*, a Barrel Toroid (BT) and two End-Cap Toroids (ECT), which generates a magnetic field for the Muon Spectrometer of 0.5 T and 1 T respectively.

The solenoid superconducting magnet is built from a single-layer coil, and it shares the cryostat with the electromagnetic barrel calorimeter. This was done in order to reduce material thickness in front of the electromagnetic calorimeters, and thus to reduce the probability that the particles start showering before they reach the active part of the calorimeter[7]. For the same reason, the solenoid was designed to be shorter than the Inner Detector, i.e. 5.8 m versus 7.2 m. This geometry produces some non-uniformity in the magnetic field along the z-axis, which drops from the nominal value of 2 T at the interaction point to about 0.5 T through the ends of the Inner Detector.

The system of toroid magnets is designed to produce a magnetic field coverage in the range $0 < |\eta| < 2.7$. The three toroids consist each of eight coils assembled radially and symmetrically

---

[7]at normal incidence, the contribution of the central solenoid assembly is 0.66 radiation lengths [ATL08a].
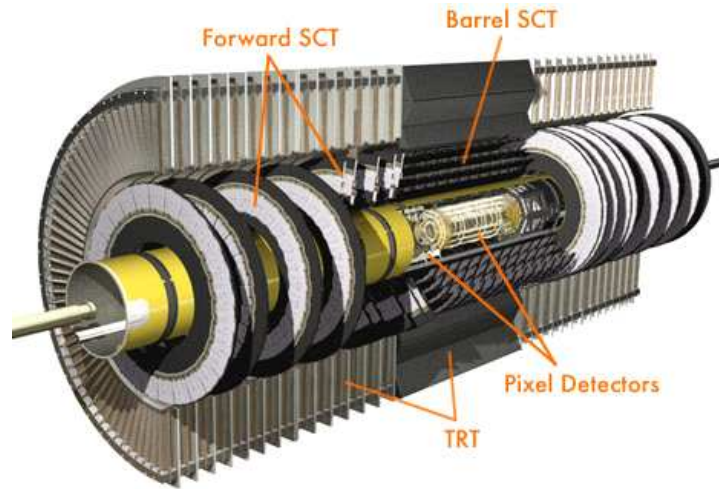
Figure 3.5: Cut-away view of the ATLAS Inner Detector [ATL10].

around the beam axis. The BT coils are contained in individual cryostats, and they are located in between the hadronic calorimeters and the Muon Spectrometer, while the eight coils of each ECT are assembled in a single large cryostat.

3.2.3 The Inner Detector

The Inner Detector (ID) is designed to reconstruct tracks and vertices with high efficiency. It measures the momenta of charged particle tracks and the decay vertices of short-lived particles, and it contributes, together with the calorimeters and the muon system, to the electron, photon and the muon identification. The 2 T solenoid field, in which the ID is immersed, allows to measure tracks with a transverse momentum $p_T > 0.5$ GeV. For particles with lower transverse momentum, the track reconstruction efficiency is limited due to a large material effect in the ID [ATL08b]. The low momentum determines small bending radii, and the respective particles cannot escape the ID, looping in the solenoidal field.

Figure 3.5 shows an overview of the ID. It consists of three different types of sub-detectors that cover the pseudorapidity range of $|\eta| < 2.5$. At the inner radii two high-resolution detectors, the Pixel Detector and the Semiconductor Tracker (SCT), are used for high-precision pattern recognition measurements, while at the outer radii the Transition Radiation Tracker (TRT) provides continuous tracking elements that enhance the pattern recognition and improve the momentum resolution over the pseudorapidity range $|\eta| < 2.0$. Each of these sub-detectors is divided into a barrel component and two end-cap components at either side. In the barrel region, the high-resolution layers are arranged in concentric cylinders around the beam axis, while in the end-cap region they are mounted on disks perpendicular on the same axis. In a similar way, the TRT components are adjusted in parallel to the beam axis in the barrel region and radially in the end-cap regions [ATL08a]. The ID sub-detectors are briefly described in the following sections.

The Pixel Detector

The Pixel detector is the component of the ID closest to the interaction point. It is made of 1744 modules, each of which consists of a 15.5 cm^2 wide and 250 μm thick silicon sensor. The nominal size of a standard pixel is 50 μm in the ϕ direction and 400 μm in z (barrel) or R[8] (end-cap) directions. This leads to an intrinsic accuracy of 10 μm in the R-ϕ plane and 115 μm in the z direction. The pixel modules are arranged in three layers in the barrel region, and in three disk layers in the two end-cap regions. The innermost layer in the barrel section is placed at about 5 cm around the beam axis. This provides a good vertex resolution, which is essential in order to separate the decay and production vertices of short-lived particles such as the B-mesons and the τ-leptons.

The principle of operation of the Pixel detector is based on a p-n junction in reverse bias. A charged particle passing through the detector produces pairs of movable electrons and holes along its path. The applied electric field separates the charge carriers and drifts them to the surface of the silicon sensor, where they are detected by charge sensitive amplifying electronics. Each silicon sensor provides 46,080 readout channels, which gives a total of approximatively 80.4 million readout channels. This represents almost half of the total number of channels available in ATLAS, i.e $\sim 10^8$.

Due to the proximity to the interaction point, the Pixel detector has to operate in an extremely hostile environment of radiation. The expected dose for the innermost barrel layer is expected to reach 500 kGy after approximately five years of LHC operation at design luminosity. For this reason, the layer has to be replaced after approximatively five years of operation. The other two barrel layers and the end-cap disks are expected to reach the same radiation dose after ten years of LHC operation [ATL08a].

The Semiconductor Tracker

The SCT provides four space points per track in the intermediate radial range of the ID, contributing to the measurement of the momentum, impact parameter and the vertex position. It also provides good pattern recognition by use of high resolution.

The SCT consists of 4088 silicon micro-strip modules, which are arranged in four concentric layers in the barrel region and nine disks on either end-cap side. Each layer consists of a double layer of silicon strips, with a 40 mrad stereo angle between them, in order to measure two coordinates. In the barrel section, one set of strips in each layer is arranged in parallel to the beam axis, to measure the ϕ angle, while the second set of strips is tilted in order to provide the z direction. In the end-cap sections, one set is arranged radially, while the other one is tilted with the same stereo angle. This geometry provides a spatial resolution per silicon module of about 17 μm in the R-ϕ plane, and 580 μm in the z (barrel) or R (end-cap) directions.

The silicon modules cover a surface of 63 m^2 and provide about 6.3 million readout channels [ATL08a].

---

[8]R is the radial distance from the beam line ($R = \sqrt{x^2 + y^2}$).

The Transition Radiation Tracker

The TRT makes use of the *transition radiation effect* for particle identification, and combines it with spatial resolution for particle tracking. A relativistic charged particle emits photons when it crosses the boundary of two materials with different dielectric constants. The energy radiated by the passing particle is linearly proportional to the Lorentz factor ($\gamma = E/m_0$). For electrons, the emitted transition radiation photons have energies in the X-ray range, while for heavier particles the emitted radiation has a considerably lower energy. Thus, the TRT contributes to the electron identification, by discriminating them from heavier charged particles.

The TRT is made of 4 mm diameter thin proportional drift tubes (straws). Each straw tube is equipped in the centre with a 31 μm diameter gold-plated tungsten wire, acting as an anode, and filled with a xenon-based gas mixture[9]. The cathode is represented by a 0.2 μm aluminium coating layer on the tubes.

The barrel section comprises 52,544 straws, each 144 cm long, arranged in 73 layers parallel to the beam axis. The two end-cap sections contain each 122,880 straws of 37 cm length, organised in 160 planes radially aligned to the beam axis. In both cases, the space between successive straw layers is filled with 15 μm thick polypropylene foils, which act as radiators. The adopted geometry ensures that particles with transverse momentum $p_T > 0.5$ GeV cross at least 36 straw tubes in the pseudorapidity range $|\eta| < 2.0$. Also, due to the same geometry, the TRT only provides a measurement in the R-ϕ plane, with an intrinsic accuracy of 130 μm per straw [ATL08a].

3.2.4   The Calorimetry

The ATLAS calorimetry plays an important role in the reconstruction of physics processes of prime interest. The calorimeters have to measure the energy and position of electrons, positrons, photons, isolated hadrons and jets, provide an accurate estimation of the missing transverse energy (E_T^{miss}), and contribute to the particle identification. Additionally, information from the calorimeters is used at the first level of triggering in order to identify the interesting physics events.

Figure 3.6 shows an overview of the ATLAS Calorimetry. The system is subdivided into an inner electromagnetic calorimeter, to measure mainly electrons, positrons and photons through their electromagnetic interactions, and an outer hadronic calorimeter, to measure mainly hadrons through their strong and electromagnetic interactions. Both subsystems are sampling calorimeters with full ϕ-symmetry and coverage around the beam axis, and cover together the pseudorapidity range $|\eta| < 4.9$. Each subsystem consists of a barrel and two end-cap components. In addition, the hadronic calorimeter is equipped in the forward regions with a dedicated system. Table 3.1 summarises the granularity and the pseudorapidity coverage of the ATLAS calorimeters.

The components of the ATLAS calorimetry are briefly described in the following sections.

---

[9]70% Xe (for an efficient absorption of the transition radiation photons), 27% CO_2 and 3% O_2 (for constant drift velocity over a large drift range, small electron deflection in magnetic fields, and ultraviolet photon quenching).
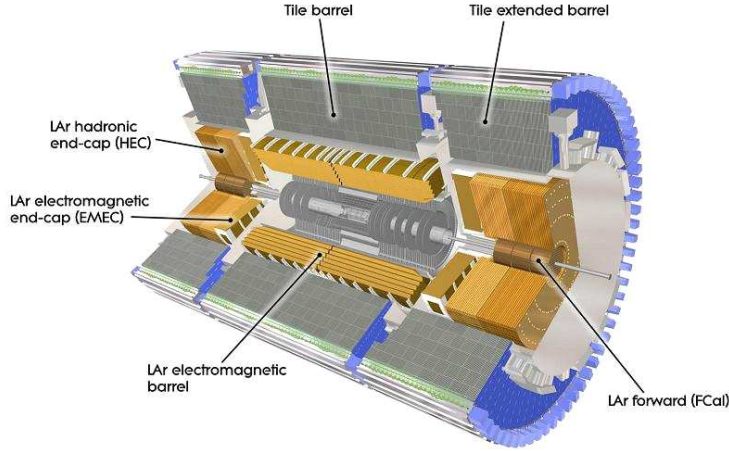
Figure 3.6: Cut-away view of the ATLAS Calorimetry system [ATL08a].

The Electromagnetic Calorimeters

The electromagnetic calorimeters are lead-liquid argon (LAr) detectors with accordion-shape absorbers and electrodes (see e.g. figure 3.7). The absorbers are made of lead plates. They are responsible for both inducing electromagnetic showers, as well as for absorbing the shower constituents of which energy is insufficient to create new particles. The gaps between the lead plates are filled with LAr, which provides the active layer of the calorimeter. The secondary electrons and positrons created in the electromagnetic shower ionise the liquid argon atoms, and the resulting ionisation signal is collected by electrodes located in between the absorber plates. The electrodes consist of three conductive copper layers. The two outer layers distribute the

Calorimeter	η coverage	$\Delta\eta$ x $\Delta\phi$ granularity	
Presampler	$|\eta| < 1.8$	0.025 x 0.1	
Electromagnetic Barrel	$|\eta| < 1.475$	0.003 x 0.1	(S1)
		0.025 x 0.025	(S2)
		0.050 x 0.025	(S2)
Electromagnetic End-Cap	$1.375 < |\eta| < 3.2$	0.003-0.1 x 0.1	
Hadronic Barrel	$|\eta| < 1.0$	0.1 x 0.1	
Hadronic Extended Barrel	$0.8 < |\eta| < 1.0$	0.1 x 0.1	
Hadronic End-Cap	$1.5 < |\eta| < 2.5$	0.1 x 0.1	
	$2.5 < |\eta| < 3.2$	0.2 x 0.2	
Forward Calorimeter	$3.1 < |\eta| < 4.9$	~ 0.2 x 0.2	

Table 3.1: The granularity and the pseudorapidity coverage of the ATLAS calorimeters.
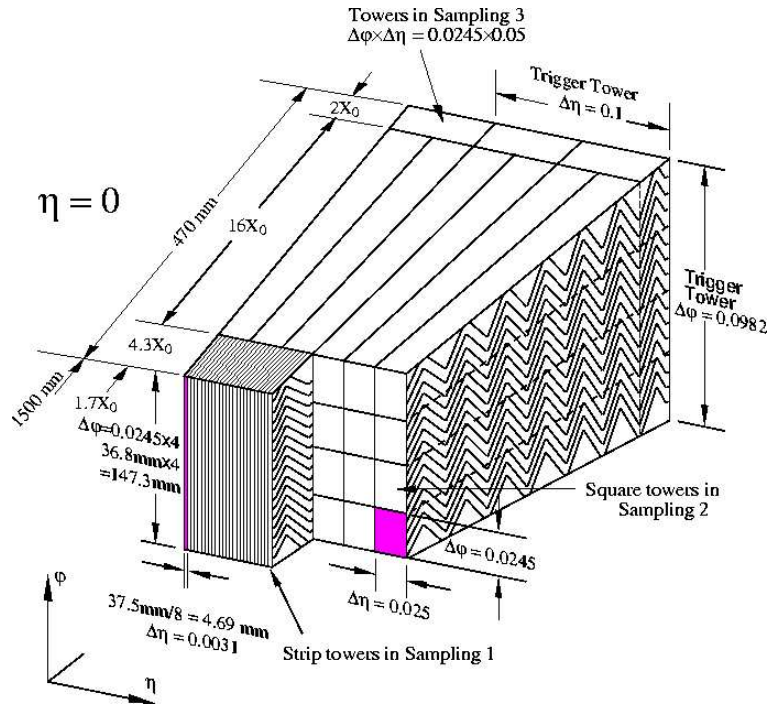
Figure 3.7: Segmentation of the LAr electromagnetic barrel calorimeter [ATL08a].

high-voltage across the LAr gaps, while the inner layer is used for reading out the ionisation signal via capacitive coupling.

The absorbers and the electrodes are *projective* to the interaction point in the η direction, and bent into an accordion shape to ensure that the incident particles cross both the absorbers and the active layers. This geometry provides a full azimuthal coverage, without any cracks, and a fast extraction of the ionisation signal at the rear or at the front of the electrodes.

The *electromagnetic barrel calorimeter* (EMB) is placed behind the central solenoid, inside the barrel cryostat that surrounds the ID cavity. It consists of two half-barrels, separated by a 4 mm gap at $z = 0$. Each half-barrel consists of 1024 lead absorbers, interleaved with electrodes. In order to achieve an uniform sampling fraction in ϕ, the thickness of the lead plates varies with η, i.e. 1.53 mm for $|\eta| < 0.8$, and 1.13 mm for $|\eta| > 0.8$. The three electrodes are positioned by honeycomb spacers at the half-distance between two neighbouring absorber plates. The folding angle of the absorbers and the electrodes varies with the radius, in order to keep constant the thickness of the LAr gap, i.e. 2.1 mm. At an operating voltage of 2 kV, this geometry determines a total electron drift time to the readout electrode of 450 ns.

Each half-barrel is mechanically divided into 16 modules, each of which covers an equal slice of the full azimuthal range. Figure 3.7 describes schematically a module of the EMB. It consists of three longitudinal sampling layers:

- *Front Sampling* (S1): it represents the layer closest to the interaction point. It is segmented into fine strips in the η direction, to provide a good resolution for γ/π^0 separation. The

depth of the layer corresponds to 4.3 radiation lengths (X_0);

- *Middle Sampling* (S2): it is arranged in square cells of very fine granularity (see again table 3.1), to provide a good position measurement of the deposited energy. The thickness of the layer corresponds to 16 X_0, to ensure that the largest energy fraction of the electromagnetic shower is absorbed within this layer;

- *Back Sampling* (S3): it is also divided into cells, but of a coarser granularity in η than the second sampling layer. Since only the highest energetic constituents (*the tail*) of the electromagnetic shower reach this layer, the energy resolution is not affected by the coarser granularity. The depth of the layer corresponds to 2 X_0.

The total material seen by an incident particle in front of the EMB is about 2.3 X_0 at $|\eta| = 0$, and increases up to 6 X_0 at $|\eta| = 1.475$, since the distance travelled by the particles increases also with η. In order to correct for the energy lost in the ID, central solenoid and the cryostat wall, the two half-barrels are preceded by a presampler detector. This is made of an 11 mm thin LAr layer, which is subdivided into 64 identical azimuthal cells, i.e. 32 per half-barrel. The total active thickness of one EMB module is larger than 22 X_0, increasing from 22 X_0 at $|\eta| = 0$ up to 33 X_0 at $|\eta| = 1.3$. Also, each EMB module and the corresponding presampler cells in front of it provide together a total of 3424 readout channels.

The *electromagnetic end-cap calorimeters* (EMECs) are contained in two cryostats together with the hadronic end-cap and the forward calorimeters. Each EMEC consist of two co-axial wheels, each of which being subdivided into eight wedge-shaped modules, without introducing any discontinuity along the azimuthal angle. The outer wheel ($1.475 < |\eta| < 2.5$) consists of 768 lead absorbers, while the inner wheel ($2.5 < |\eta| < 3.2$) is made of 256 lead absorbers. As for the EMB, the thickness of the lead plates varies with η, in order to optimise the energy response of the EMECs, i.e. 1.7 mm in the outer wheel, and 2.2 mm in the inner wheel. The electrodes are again positioned in the middle of the gaps by honeycomb spacers, at 2.1 mm from the neighbouring absorber plates. In the outer wheel, the ionisation signals are read out from both sides of the electrodes, as in the case of the EMB, while in the inner wheel, the signals are read out only from the back side, due to high radiation levels at the front side.

In the pseudorapidity range $1.5 < |\eta| < 2.5$ the EMECs are segmented in three longitudinal sampling layers, in the same way as the EMB. In the pseudorapidity ranges $1.375 < |\eta| < 1.5$ and $2.5 < |\eta| < 3.2$ the EMECs are segmented in only two longitudinal sampling layers, and have a coarser transverse granularity. Also, in the range $1.5 < |\eta| < 1.8$ each EMEC is preceded by a presampler, which consists of two 2 mm thin LAr layers equipped with readout electrodes. The total active thickness of one module of the EMEC is larger than 24 X_0. Also, each module provides 3984 readout channels, including 96 channels from the presampler.

The energy resolution that ATLAS expects to achieve with the LAr electromagnetic calorimeters is:

$$\frac{\sigma(E)}{E} = \frac{10\%}{\sqrt{E}} \oplus 0.7\% \quad , \tag{3.6}$$

where the first term represents the *stochastic term*, which takes into account the statistical fluctuations in the shower development, while the second term represents the *constant term*, which
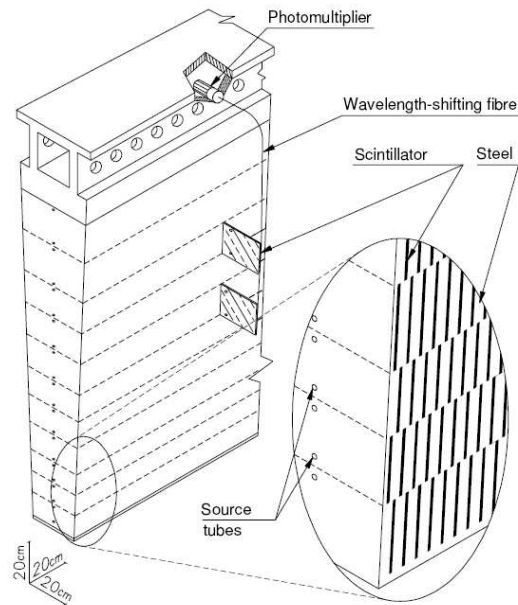
Figure 3.8: Schematic view of a Tile module [ATL08a].

accounts for all detector imperfections and calibration imprecisions. The symbol $\oplus$ represents the quadratic sum operator.

The Hadronic Calorimeters

The hadronic calorimeter covers the pseudorapidity range $|\eta| < 4.9$ and uses two different sampling techniques: steel-scintillating tiles in the barrel region, and copper-LAr in the end-cap and forward regions.

The *Tile calorimeter* uses steel as absorber and plastic scintillating plates (*tiles*) as active medium. It is located behind the LAr electromagnetic calorimeter, and it is divided into a central barrel and two extended barrels. Each component is further subdivided into 64 modules of equal azimuthal size, i.e. $\Delta\phi =\sim 0.1$. Figure 3.8 shows a schematic view of a module of the Tile calorimeter. Within the module, the scintillating tiles are inserted in a rigid steel structure, perpendicular on the beam axis and staggered in depth. They are 3 mm thick, and are made of optical transparent granulated polystyrene doped with scintillating additives[10]. An ionising particle crossing the Tile module induces the production of ultaviolet light in the polystyrene material, which is then converted by the scintillating additives to visible blue light. Two wavelength-shifting fibers collect this light at the edges of each tile, shift it to a longer wavelength, and guide it to the input of two photomultipliers (PMTs). The PMTs are housed by a rigid support girder at the outer edge of the module, together with the front-end electronics. The Tile module is segmented into a three-dimensional cell structure, which provides three radial sampling depths.

---

[10] *1.5% para-terphenyl* (PTP) and 0.44% *1,4-bis(5-phenyloxazol-2-yl) benzene* (POPOP), both acting as wavelength shifters.

Each cell is obtained by grouping wavelength-shifting fibers from corresponding tiles into two bundles, i.e. one per side of the module, and coupling the bundles to two PMTs. The granularity of the cells is $\Delta\eta = \sim 0.1$, for the inner two cells, and $\Delta\eta = \sim 0.2$, for the outer one, while the thickness of the sampling depths is approximatively 1.5, 4.1 and 1.8 interaction lenghts (λ_{int}) at $\eta = 0$.

The central barrel is separated from each extended barrels by a gap of about 70 cm. These gaps provide space for cables and services for the ID, and for power supplies and services for the LAr EMB. The respective regions are only partially instrumented, with special modules made of steel-scintillator, which provide the same sampling fraction as the rest of the Tile calorimeter, and with thin scintillator counters. These devices allow partially to recover the energy lost in the crack regions of the detector. The Tile calorimeter has a total radial length of approximatively 7.4 λ_{int}, and provides about 10,000 readout channels.

The *hadronic end-cap calorimeters* (HECs) use copper plates as absorbers and LAr as active medium. Each HEC consists of two cylindrical wheels arranged concentrically around the beam axis. Each HEC wheel is constructed from 32 identical modules, and it is divided into two longitudinal readout segments. In the wheels closer to the interaction point the modules are made of 24 copper plates, each 25 mm thick, while in the outer wheels the sampling fraction is coarser, the modules being made of 16 copper plate, each 50 mm thick. In both wheels, the three electrodes divide the LAr gap into four equal drift zones of 1.8 mm. Each zone is supplied with a high voltage of 1.8 kV via the outer electrodes, which determines an electron drift time of about 430 ns. The two HECs provide together a total of 5632 readout channels. The active part of each HEC corresponds to approximatively 12 λ_{int}.

The *forward calorimeters* (FCALs) are placed at high η, at a distance of approximatively $\sim$4.7 m from the interaction point. Therefore, the FCALs have to cope with a particularly high level of radiation. Each FCAL is divided into three 45 cm deep modules. The first module (FCAL1) uses copper as absorber and is optimised for electromagnetic measurements. The other two modules (FCAL2, FCAL3) are made of tungsten and measure predominantly the energy of hadronic interactions. Each FCAL module consists of a metal matrix with regularly spaced longitudinal channels filled with electrodes. Because of the high density of the design, the LAr gaps are much smaller than in the other ATLAS LAr calorimeters, i.e. 0.25 mm (FCAL1), 0.35 mm (FCAL2), 0.5 mm (FCAL2). Consequently, the electron drift times are shorter, e.g. 60 ns in FCAL1. Each FCAL is approximately 10 λ_{int} deep and provides 1762 readout channels.

The energy resolution expected for the hadronic calorimeters is [ATL97]:

$$\frac{\sigma(E)}{E} = \frac{50\%}{\sqrt{E}} \oplus 3\% \qquad , \qquad |\eta| < 3,$$

$$\frac{\sigma(E)}{E} = \frac{100\%}{\sqrt{E}} \oplus 10\% \qquad , \qquad 3 < |\eta| < 5, \qquad\qquad (3.7)$$

3.2.5 The Muon Spectrometer

The Muon Spectrometer is designed with a twofold functionality:

- to detect the charged particles that penetrate the calorimetry, and to determine their momentum by measuring the bending of their tracks in the toroidal magnetic field;
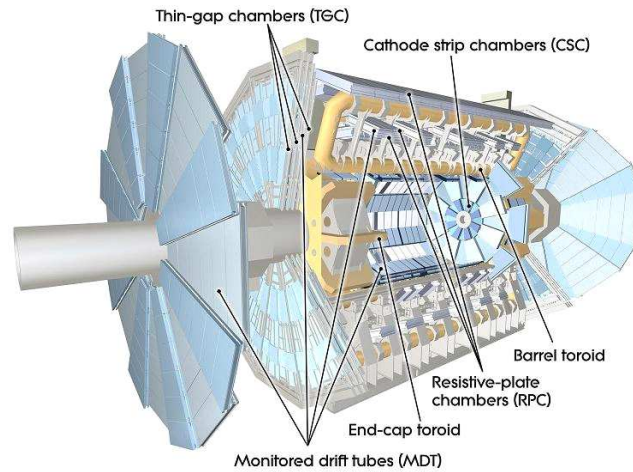
Figure 3.9: Cut-away view of the ATLAS Muon Spectrometer [ATL08a].

- to trigger on the detected particles.

The first task is accomplished by two high precision-tracking chambers, Monitored Drift Tubes (MDTs)[11] and Cathode Strip Chambers (CSCs), which cover together the pseudorapidity range $|\eta| < 2.7$. The second task is delegated to two types of trigger chambers, Resistive Plate Chambers (RPCs) and Thin Gap Chambers (TGCs), which cover the pseudorapidity range $|\eta| < 2.4$ (see also figure 3.9). The two subsystems of the Muon Spectrometer are briefly described in the following sections.

The Precision-tracking Chambers

The high precision-tracking system aims to measure the muon transverse momentum with a resolution of approximatively 10% for 1 TeV tracks. The system consists of three barrel layers and four end-cap disks at each side. The arrangement of the layers and the disks is optimised for full coverage and momentum resolution. The barrel layers are arranged as three concentric cylindrical shells around the beam axis, at radii of approximately 5 m, 7.5 m, and 10 m, while the end-cap disks are positioned perpendicularly to the beam axis, and they are located at distances of approximately 7.4 m, 10.8 m, 14 m, and 21.5 m from the interaction point.

The precision measurement of the track coordinates are performed with MDTs over most of the pseudorapidity range covered by the tracking system. Exception make the innermost end-cap disks ($2.0 < |\eta| < 2.7$), which are equipped with CSCs.

The *MDT chambers* consist of three to eight layers of drift tubes. Each drift tube consists of a 30 mm diameter aluminium tube, which is operated at 3 bar absolute pressure with a gas mixture of Ar-CO_2 (93% - 7%). The ionisation charge produced by a muon traversing the tube is collected at a central tungsten-rhenium (W-Re) anode wire of 50 μm diameter. The maximum

---

[11]*monitored* is given by an internal chamber alignment system that continuously monitors the relative position and the potential deformations of the chamber due to gravitational forces.

drift time, from the walls of the tube to the anode wire, is about 700 ns. The achieved spatial resolution is 80 μm per tube and 35 μm per chamber. In total, the muon tracking system comprises 1150 MDT chambers, which provide together approximatively 354,000 readout channels.

The *CSC chambers* are multi-wire proportional chambers with two cathodes, each segmented into strips orthogonal to the anode wires. The CSCs are preferred to MDTs in the innermost end-cap disks, the closest disks to the interaction point, because they provide a higher granularity, which is needed to withstand the high rate and the background conditions. Besides this, the CSCs offer the advantage of a smaller electron drift time (i.e. <40 ns), a good time resolution (i.e. 7 ns), a good two-track resolution and a low neutron sensitivity. The total number of CSC chambers used in the tracking system is 32, and these provide together approximatively 31,000 readout channels

The Trigger Chambers

The trigger chambers have to provide bunch-crossing identification, provide a trigger with well-defined p_T thresholds, and measure the muon coordinate in the direction orthogonal to that determined by the precision-tracking chambers. The system uses RPC chambers in the barrel region ($\eta \leq 1.05$), and TGC chambers in the end-cap regions ($1.05 \leq \eta \leq 2.4$). The RPCs are mounted together with the MTDs, so that two RPCs sandwich the MDTs of the middle barrel layer, while the third RPC is located close to the outer layer (see e.g. figure 4.3). In the end-cap, one TGC is placed in front of the second MDT disk, while two other TGCs are installed behind the same disk. A fourth TGCs if placed in front of the innermost tracking layer.

The *RPC chambers* are gaseous detectors providing a typical space-time resolution of approximatively 1 cm x 1.5 ns. The active element of the RPC is a narrow gas gap, which is formed by two resistive plates placed in parallel to each other at a distance of 2 mm. The primary ionisation electrons are multiplied in avalanches by an uniform electric field of about 4.5 kV/mm, and the resulting ionisation signal is read out via capacitive coupling by metal strips placed on both sides of the detector. An RPC chamber consists of two independent detector layers, each measuring the η and ϕ coordinates simultaneously. A muon passing all three RPCs in the barrel region will determine six measurements of track points. This redundant measurement provides an efficient rejection of the fake tracks from noise hits. The muon trigger system uses 606 RPC chambers, which provide in total 373,000 readout channels.

The *TGC chambers* are similar in design to multi-wire proportional chambers, but with a smaller anode wire-to-wire distance (1.4 mm) than the wire-to-cathode distance (1.8 mm). This feature leads to a short drift time and thus to a good time resolution. The TGC chambers are mounted in such a way that the wires measure the R coordinate. The ϕ coordinate is measured by radial copper strips applied on the back side of the cathode plates. A total of 3588 TGC chambers are used in the system, providing together 318,000 readout channels [ATL08a].

3.2.6 The Trigger, Data Acquisition and Detector Control Systems

The task of the *ATLAS Trigger* system is to reduce the initial event rate to an affordable mass storage rate, while efficiently selecting interesting candidate events from the enormous background produced in *pp* collisions. The *Data Acquisition* (DAQ) system is mainly responsible

for transfering the selected detector event data to the permanent storage medium, as well as for managing the configuration, control and monitoring of the ATLAS sub-detectors during the data-taking. The two systems, often referred as to one entity - TDAQ, interact closely with a third system, *Detector Control System* (DCS), to ensure reliable physics data-taking.

The DCS monitors continuously the operational conditions of the detector hardware, e.g. gas pressures, power supply voltages or temperatures across modules. It signals any observed mis-behaviour by means of warning and error flags, and, if needed, takes automatically appropriate corrective actions to bring the detector hardware into a safe state. The DCS is also responsible for the communication with external systems and services. The most notable is the bi-directional communication with the LHC. The latter provides overall status of the accelerator (e.g. shut-down, filling) or beam parameters (e.g. luminosities, beam sizes and profiles), and receives from DCS various types of information, like distribution of observed background in the sub-detectors, or measured luminosities and trigger rates. Other external systems to which the DCS interacts are the Detector Safety System (DSS), which monitors and prevents situations that can cause major damages, and the CERN-wide safety and alarm system [ATL03].

The Trigger and Data Acquisition systems are described in more detail in the next chapter.

Chapter 4

The ATLAS Trigger and Data Acquisition Systems

As described in section 3.1.2, the harsh environment at the LHC imposes severe requirements to the design of the detector and of the trigger system. Two of the main challenges, related to operating at design luminosity of $10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, are represented by the pile-up of the low p_T events and by the large QCD jet production that dominates the rate of high p_T events. In order to reduce the impact of the pile-up, the ATLAS sub-detectors are segmented into finely granular cells and feature good time resolution. However, the reverse of a low detector occupancy is a large number of readout channels and, consequently, an enormous amount of raw data. For the given bunch-crossing rate of 40.08 MHz, the approximatively 10^8 readout channels of the ATLAS detector produce a total of 1 PByte/s of raw data, a rate that cannot be sustained for mass storage with nowadays technologies. The task of the trigger system is, therefore, to reduce the initial interaction rate of 10^9 events/s (= 1 GHz) to a rate suitable for permanent storage, by rejecting the majority of the background and selecting with maximum efficiency the potential rare physics events.

The trigger system has in addition to deal with another major operational constraint, the high bunch-crossing rate. At design luminosity, 25 new events will be produced with each bunch-crossing, and the trigger will have to make a decision for each event. However, given the interval between two consecutive bunch-crossings, i.e. 25 ns, it is not feasible to process the sub-detector event data and make a trigger decision in such a very short time. The solution to this problem is to accomplish a full trigger decision in successive stages, by using pipelined trigger processing and readout architectures.

4.1   The Architecture of the TDAQ System

In order to meet all these requirements, ATLAS has adopted a trigger system with three levels of event selection: Level-1 (L1), Level-2 (L2) and Event Filter (EF). Figure 4.1 describes schematically the architecture of the ATLAS Trigger and Data Acquisition (DAQ) systems, which are often collectively called the TDAQ system.
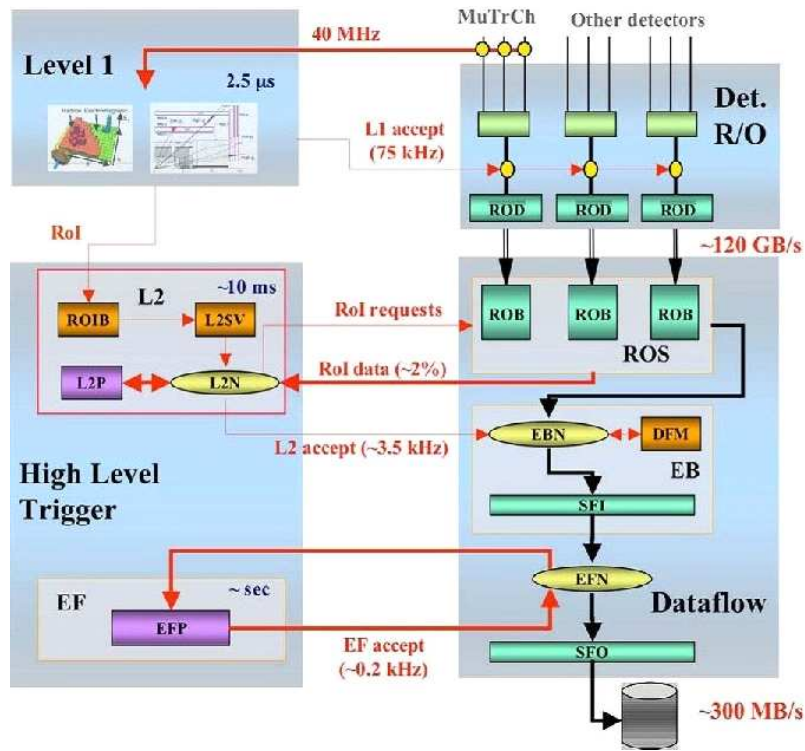
Figure 4.1: Block diagram of the ATLAS Trigger and Data Acquisition systems. The three levels of event selection are shown on the left-hand side, while the DAQ components are pictured on right-hand side.

The *L1 trigger* performs the initial event selection. It uses reduced-granularity data from all the calorimeter subsystems and the muon trigger chambers, to reduce the bunch-crossing rate of 40.08 MHz to about 75 kHz (upgradeable to 100 kHz). While the L1 makes its decision, the finely-granular detector data is stored at 40.08 MHz in pipeline memories located on or near the detector. The depth of these buffers corresponds to 2.5 μs, and defines the maximum allowed decision time, i.e. *latency*, for the L1 system[1]. This means that if the L1 does not provide a decision in less than 2.5 μs, the detector data is lost, being overwritten by new event data from subsequent bunch-crossings. In order to cope with this stringent timing constraint, the L1 is based on pipelined custom-made high-speed electronics, running synchronously with the LHC bunch-crossing frequency. When the L1 accepts an event, all the detector data stored in pipeline memories is transferred to another set of storage elements, called Read-Out Buffers (ROBs), where it is kept until it is validated or discarded by the L2 trigger. In the same time, the L1 supplies the L2 with so-called *Regions of Interest* (RoIs). These represent the η-ϕ coordinates

---

[1]in fact, the 2.5 μs latency for the L1 represents a compromise between the cost for deeper memory buffers and longer trigger processing time. Also, the actual target latency of the L1 trigger is 2 μs, out of which ∼1 μs is accounted for the transmission of the detector signals over long analogue cables, from the experimental hall to the input of the L1 system, in the electronics cavern.
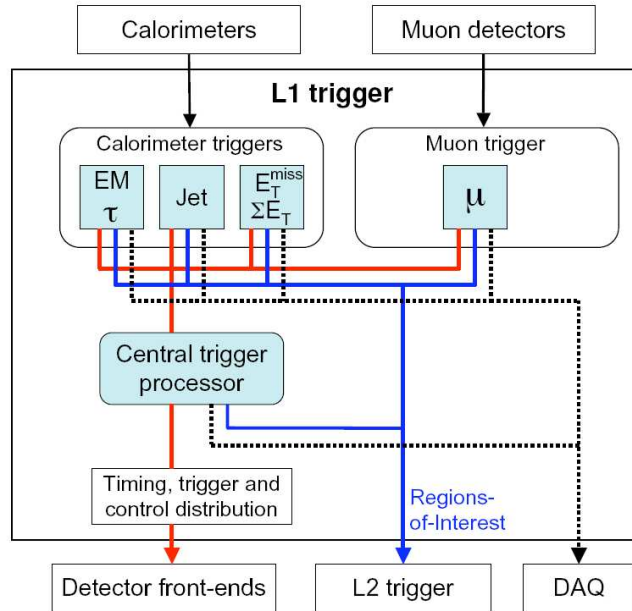
Figure 4.2: Block diagram of the L1 trigger.

of the detector regions within which the L1 trigger has identified interesting physics signatures, and the selection criteria that were passed, e.g. type of signature, energy threshold, etc.

The next trigger levels, L2 and EF, are often collectively referred as to the *High-Level Trigger*. Both systems are software-based, running on a large network of commercial computers, and have access to the full-granularity and full-precision calorimeter and muon data, as well as to the ID tracking data. The L2 trigger uses the RoI information to limit the amount of data that has to be accessed from the ROBs to about 2%. Based on this, the L2 further reduces the rate to 3.5 kHz with an average processing time of about 40 ms. When the L2 decision is positive, an Event Builder (EB) system retrieves the related event data from the ROBs, assembles it in a single formatted data structure, and transfers it to the EF. The latter uses offline analysis algorithms to reduce the event rate down to the more affordable storage rate of ~ 200 Hz, within an average processing time of ~ 4 s. The average size of an ATLAS event, after zero suppresion and with nominal sampling, is around 1.3 MByte, which leads to a maximum storage through-put of about 300 MByte/s [ATL08a]. The three trigger levels and the components of the DAQ system are briefly described in the following sections.

4.2 The Level-1 Trigger

The L1 trigger searches for signatures from high p_T muons, electrons, photons, jets, and τ leptons decaying into isolated hadrons. It also selects events with large E_T^{miss} and large total transverse energy ($\sum E_T$). In order to reach a decision in less than 2.5 μs, the L1 accesses only reduced-granularity data from the calorimetry and the muon detector. For the same reason, the ID tracking data is not used at this stage. Due to the huge number of readout channels provided

by the ID, the time needed to reconstruct the tracks and decay vertices would exceed the latency of the L1 system.

Figure 4.2 shows a block diagram of the L1 trigger system. It consist of three main components: a *Calorimeter Trigger* (L1Calo) and a *Muon Trigger* (L1Muon), to process the input from the calorimeters and the muon detector, and a *Central Trigger Processor* (CTP), which combines the results of the two trigger subsystems, in order to reach an overall L1 trigger decision.

4.2.1 The Calorimeter Trigger

The L1Calo is a pipelined digital system, designed to process about 7200 analogue trigger signals of coarse granularity, i.e. mostly $\Delta\eta \ x \ \Delta\phi = 0.1 \ x \ 0.1$, from the entire ATLAS calorimetry, within a fixed latency of $\sim 1 \ \mu$s. The analogue trigger signals describe energy deposits in the calorimeters. The L1Calo extracts this information, assigns it to the corresponding bunch-crossing, and uses it to identify various physics objects with high E_T and to compute global and scalar energy sums. The results of these investigations are then discriminated against programmable energy thresholds, and the obtained multiplicities are passed to the CTP. Additionally, the L1Calo provides the L2 trigger with RoIs information whenever the overall L1 trigger's decision is positive. The architecture and the algorithms of the L1Calo are presented in more detail in the next chapter.

4.2.2 The Muon Trigger

The L1Muon uses the tracking measurements performed by the RPCs (in the barrel region) and the TGCs (end-cap) subsystems of the Muon Spectrometer, to identify candidates with p_T above six programmable thresholds, and to assign them to the correct bunch-crossing. For an efficient reconstruction of the p_T over a wide range, the logic of the L1Muon divides the six programmable thresholds into two groups: the first three thresholds are associated with a low p_T trigger, and cover the approximative E_T range of $6 - 9$ GeV, whereas the last three thresholds are associated with a high p_T trigger, and cover the E_T range $\sim 9 - 35$ GeV.

The L1Muon trigger algorithm is based on coincidence of hits in the different muon trigger chambers. By using one of the stations as a *pivot plane*, the algorithm searches for time-correlated hits in the other two stations (*confirm planes*). In the barrel region, the pivot plane is considered the middle RPC (see RPC2 in figure 4.3), while the outermost TGC (TGC3) is used for the same reason in the end-cap regions. The identification of a coincidence is performed within a geometrical road, of which centre is defined by the line of conjunction of the hit in the pivot plane with the interaction point. This line can be seen as the path of a muon with infinite momentum that originates from the interaction point. Such a muon will traverse the detector without being deflected by the toroidal magnetic field. The width of the geometrical road is related to the chosen p_T threshold, i.e. the higher the p_T threshold, the narrower the road. For an efficient reconstruction, the algorithm uses all six p_T thresholds in the same time, which results in as many geometrical roads around the infinite-momentum path. A low p_T trigger is then formed when a coincidence in three out of the four layers given by RPC1 and RPC2 (barrel) or TGC2 and TGC3 (end-cap) is found. A high p_T trigger requires a low p_T coincidence and, in addition, a hit in one of the two layers of RPC3/TGC1 (see again figure 4.3). The algorithm is
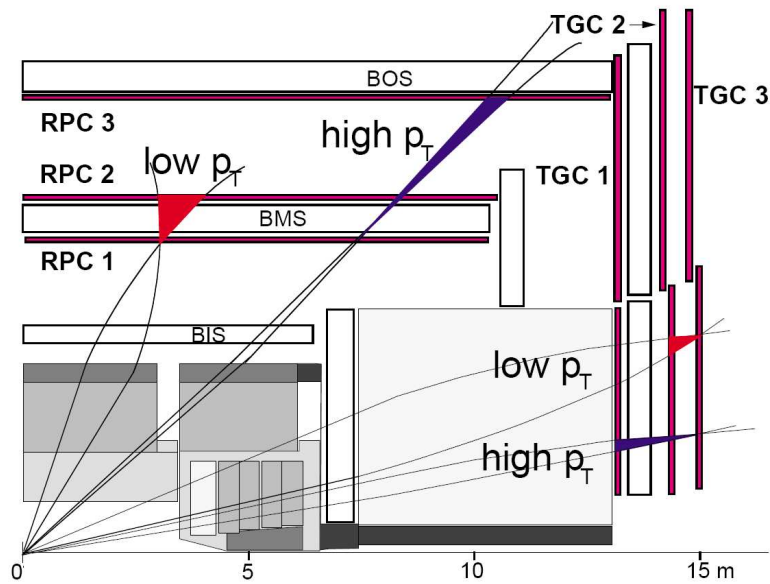
Figure 4.3: The ATLAS Muon Trigger Chambers and examples of muon tracks generating triggers.

performed simultaneously in both projections (η-ϕ in the barrel, R-ϕ in the end-cap), in order to reduce significantly the rate of accidental triggers due to low-energy particles, i.e. thermalised slow neutrons leaking from calorimeter and shielding materials.

The results obtained in the barrel and end-cap region are combined into one set of threshold multiplicities for each bunch-crossing, and sent to the CTP. As in the case of the L1Calo trigger, the L1Muon provides the L2 trigger with RoIs whenever the L1 trigger accepts an event.

4.2.3 The Central Trigger Processor

The CTP combines the information received from the L1Calo and the L1Muon to make the overall Level-1 trigger decision, within a latency of $\sim$ 100 ns. For this, the CTP implements a trigger menu of up to 256 programmable trigger items, each of which represents a logical combination of 1 to 256 trigger conditions. For example, a trigger item could be the combination of two trigger conditions: an e/γ candidate with $E_T > 10$ GeV, and a muon candidate with $p_T > 15$ GeV. The overall L1 trigger decision is then obtained by logically OR'ing all the items of the trigger menu. The output trigger signal generated by CTP is called the Level-1 Accept (L1A), and it is distributed to all the ATLAS sub-detectors and readout electronics via the *Timing, Trigger and Control* (TTC) system. A positive L1A initiates the transfer of event related data from the front-end (FE) pipeline memories to the ROBs. First, the event data is transferred to derandomising buffers, to accommodate the maximum instantaneous L1 output rate without introducing significant dead-time. Subsequently, data from multiple derandomisers is merged into a predefined ATLAS format by Readout Driver (ROD) modules, to reduce the number of links, and then the resulting data streams are sent to the ROBs (see again figure 4.1). A positive L1A will also determine the L1 subsystems to read out to DAQ copies of the digital

data on which the trigger decision was based, in order to allow a verification of the proper functioning of the system.

The CTP is also responsible for generating two types of dead-time in the system: preventive dead-time and dead-time through a *busy* signal from the DAQ system. In fact, the CTP is the only entity in ATLAS that has the ability to introduce a dead-time. The preventive dead-time is needed to avoid the overflow of the FE derandomisers. It is a programmable parameter, the typical settings use a minimum of 4 bunch-crossings after each L1A and up to 8 L1As within a window of 80 μs. The second type of dead-time is generated when one or more RODs flag that their data buffers are almost filled. In such cases the CTP slows down the L1 output rate, by introducing a dead-time until the RODs can safely buffer data again.

Last but not least, in addition to forming the trigger decision, the CTP has the task to receive timing signals from the LHC machine, e.g. the 40.08 MHz bunch-crossing clock, and provide them to all the ATLAS sub-detectors via the TTC system [ATL08a].

4.3   The High-Level Trigger and the DAQ

The *Level-2 trigger* consists of a large network of commercial PCs ($\sim$ 500), linked by a high-capacity switched network. It uses detector data at full granularity in order to refine the selection made by L1. Because the number of readout channels available at this stage is huge, and because the average processing time for one event is $\sim$ 40 ms, the L2 cannot access the complete detector data. Instead, it uses the RoI information provided by the L1 to access only a small fraction ($\sim$ 2%) of the event data stored in the ROBs. The RoI information is received by an RoI Builder (RoIB, see again figure 4.1), which assembles it into a single data structure and forwards it to the L2 supervisor (L2SV). The latter requests the related data from the ROBs, assigns it for analysis to one of the processing units of the L2 (L2PUs), and delievers the L2PU's result, i.e. accept or reject, to the Data Flow Manager (DFM). The ROBs are physically implemented on PCI mezzanine cards, located in Readout System (ROS) units, which are implemented on server-class PCs. When an event is rejected by the L2 trigger, the DFM requests the ROSs to delete the related data from the ROBs. If the L2's decision is positive, then the DFM instructs an event-building node called Sub-Farm Input (SFI) to pull out the event related data from the ROBs, and to assemble it in a single formatted data structure. When the event-building process is completed, the full reconstructed event is provided to the Event Filter. Subsequently, the DFM notifies the ROSs to delete the event related data from the ROBs.

The *Event Filter* is made of $\sim$ 1600 computing farm nodes, each of which consists of two 2.5 GHz quad-core CPU's [Ara09]. Because it has access to the full ATLAS event, the EF can run algorithms adapted from the offline reconstruction. The average processing time per event at this stage is $\sim$ 4 s. The events selected by the EF are sent to the Sub-Farm Output nodes (SFO), and from there to the CERN's central data-recording facility, for mass storage.

Chapter 5

The ATLAS Level-1 Calorimeter Trigger

The Level-1 Calorimeter Trigger (L1Calo) is a pipelined digital system using commercial and custom electronics. It is entirely housed outside the experimental hall, in a separate underground electronics cavern called USA15. The system receives and processes 7168 analogue signals from all the ATLAS electromagnetic and hadronic calorimeters. The input signals, often called *trigger-tower* signals, describe deposits of transverse energy (E_T) in multiple calorimeter cells, and they are formed by analogue summation in the detector's front-end (FE) electronics. The granularity of the trigger towers is $\Delta\eta$ x $\Delta\phi = 0.1$ x 0.1 for $|\eta| < 2.5$, and coarser at larger pseudorapidity values. The L1Calo provides results to the CTP within 2 μs after the *pp* collision has occurred. About 1 μs of this latency is assigned to the electronics of the L1Calo, the rest being accounted for cable propagation delays upstream and downstream of the system. In order to cope with the timing requirement, the L1Calo algorithms are implemented in Application-Integrated Circuit (ASIC) and Field-Programmable Gate Arrays (FPGAs).

5.1 The Architecture

The L1Calo consists of three main subsystems: the *PreProcessor* (PPr), the *Cluster Processor* (CP) and the *Jet/Energy-sum Processor* (JEP) (see figure 5.1). The PPr receives and digitises the 7168 analogue trigger-tower signals, extracts a corresponding E_T value from each pulse and identifies it with a specific bunch-crossing. The digital results are then transmitted in parallel to the subsequent subsystems. The CP identifies isolated clusters of electrons, photons, taus or hadrons. The JEP identifies jet objects and computes global sums of total, missing and jet-sum E_T, based on a coarser granular input, i.e. 0.2 x 0.2 in $\Delta\eta$ x $\Delta\phi$. The two processors discriminate against programmable thresholds the E_T of each identified object and the computed energy sums. The resulting threshold *multiplicities* in each processor are first merged to obtain system-wide results, and then sent to the CTP. The latter combines the data received from the L1Calo and L1Muon to form the Level-1 Accept (L1A) decision. Upon receiving the L1A signal from the CTP, the L1Calo subsystems transmit event related data to the DAQ system, via dedicated
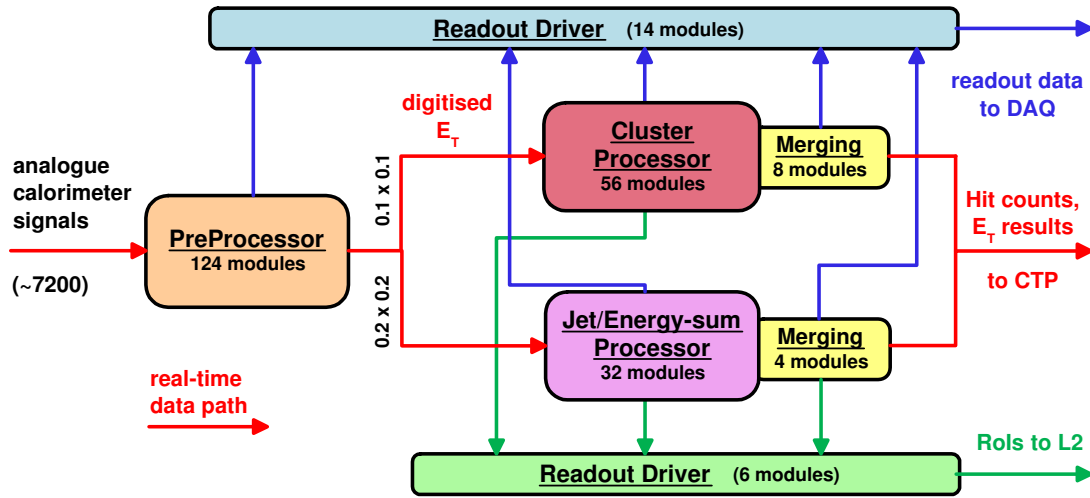
Figure 5.1: Block diagram of the Level-1 Calorimeter Trigger system.

Readout Driver (ROD) modules. These data consist of trigger-tower energies and processing results, to allow the verification and the monitoring of the trigger operation. In the same time, RoI information, comprising the type and location of the identified objects and the energy sums, is passed to the L2 RoI Builder via another set of ROD modules [L1C08a].

All the L1Calo components are 9U[1] VME systems. The PPr and the ROD use the standard VMEbus system, while the CP and JEP use a custom backplane and a reduced VMEbus implementation. The L1Calo components and algorithms are presented in more detail in the following sections.

5.2   The Analogue Input

The signals received from the LAr and Tile calorimeters have a different shape. In case of the LAr, the charge collected by the readout electrodes determines a triangular waveform with less than 1 ns rise time, and a linear decay that corresponds to the electron drift time in the LAr gap (see figure 5.2a). The ionisation pulse from each calorimeter cell is first amplified, to reduce the sensitivity to noise in the next stages of the FE electronics, and then shaped by bipolar shapers, to optimise the signal-to-noise ratio. The output of the shaper is a signal with a rise time of ~ 50 ns and a long negative undershoot. The positive part of the signal spans over 5 bunch-crossings, and its integral and peak amplitude are proportional to the energy deposition in the calorimeter cell [Liq08]. In case of the Tile calorimeter, the current pulses provided by the PMTs are much faster, mainly due to the optical properties of the wavelength shifter fibers. These signals have a rise time of about 5.5 ns and a FWHM[2] of about 15 ns. In order to allow a common processing of the LAr and Tile calorimeter signals in the trigger system, the PMT

---

[1]the electronics modules are 366 mm high and 400 mm wide.

[2]full-width at half maximum

(a) [Liq07] (b) [Til96]
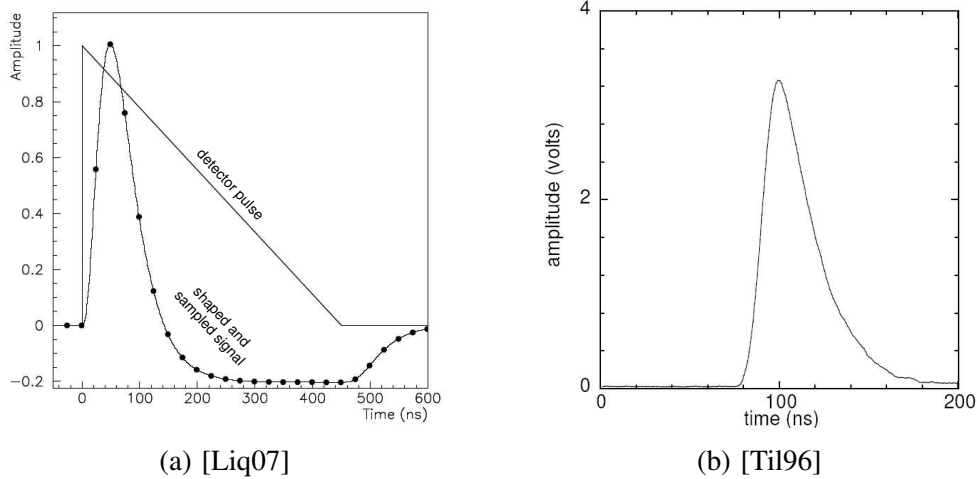
Figure 5.2: LAr ionisation signal and front-end electronics response after bipolar shaping (a), and unipolar shaped signal from the Tile calorimeter (b).

output is shaped to an unipolar signal with a FWHM of $\sim$ 50 ns, similar to that of the LAr pulses [Til96] (see figure 5.2b).

The signals from multiple cells are summed by the FE electronics of each calorimeter system to form trigger-tower signals for the L1 system. The analogue summation is performed separately for the electromagnetic and hadronic layers of the calorimetry. The number of calorimeter cells summed to form a trigger tower varies from a few in the end-caps up to 60 in the electromagnetic barrel. The trigger towers are adjacent and projective to the interaction point in the η direction, and they cover the full depth of the given calorimeter layer. The granularity of the towers is $\Delta\eta$ x $\Delta\phi = 0.1$ x 0.1 for $|\eta| < 2.5$, and coarser at larger pseudorapidity values. The analogue sums from the LAr EMB and EMEC calorimeters are proportional to the E_T, and the amplitudes of these signals are linear up to the maximum level specified by the trigger, i.e. 256 GeV E_T = 2.5 V. The analogue sums from the Tile and the LAr HEC and FCAL calorimeters describe only the raw energy E deposited in the trigger towers. Also, in the overlap region between the EMB and EMEC ($1.375 < |\eta| < 1.475$) and in the FCAL the trigger sums are only preliminary. As described a bit later, the final summations and the conversion to E_T for the hadronic signals are performed by an off-detector electronics system, located in the USA15, upstream of the L1Calo.

The trigger-tower signals are driven differentially from the detector to the USA15 cavern over 616 16-way twisted-pair copper cables of variable length. The shortest cables are those providing signals from the LAr barrel calorimeter, i.e. $\sim$ 30 m, while the cables carrying signals from the Tile extended barrel calorimeters are the longest, i.e. $\sim$ 70 m. In addition to the trigger-tower signals, the cables from the Tile calorimeter transport also signals from the third sampling layer. These signals are addressed to the L1Muon, to serve as additional information for background suppression, and thus they have to be separated from the trigger-tower signals. This operation is done by a system of patch panels, called Tile Calorimeter Patch Panels (TCPP) (see figure 5.3). The Tile trigger-tower signals and all the analogue sums from the LAr calorim-
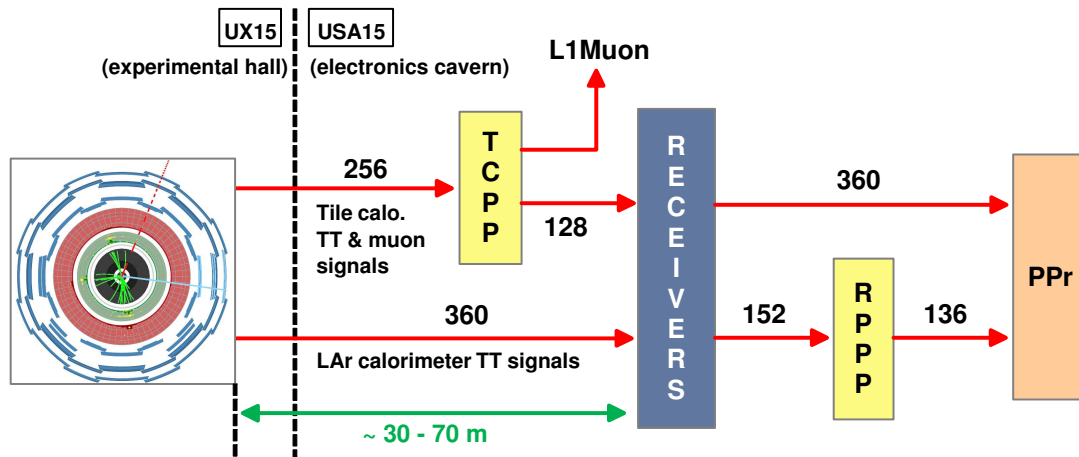
Figure 5.3: The handling of the analogue trigger sums from the calorimeters to the input of the L1Calo. The numbers given in the figure indicate the amount of twisted-pair cables needed to transport the signals between various systems.

eters arrive at the input of a Receiver/Monitor system, of which main functions are summarised in the following:

- **Gain adjustment.** The Receiver system uses linear variable-gain amplifiers (VGAs) to adjust the amplitude of each input signal to the correct voltage scale, i.e. 10 mV = 1 GeV E_T. For signals from the hadronic calorimeters (Tile, LAr HEC and FCAL) the gain factor is proportional to $sin\theta$, in order to convert the respective trigger-tower energies from raw E to E_T. The VGAs are also used to compensate for signal attenuation ($\sim$40%) in the long cables from the detectors. The reference voltage of each VGA is controlled via a 12-bit Digital-to-Analogue Converter (DAC);

- **Signal summing.** In two cases the Receiver system performs further analogue summations: for the overlap region between the EMB and EMEC, and for the hadronic layers of the FCAL (FCAL2, FCAL3), to reduce the η granularity at the input of the L1Calo;

- **Signal re-ordering.** The analogue signals arrive at the input of the Receiver system in an order mainly determined by the structure of the calorimeters. The Receiver system rearranges the signals at the output in the order required by the L1Calo, i.e. typically one 16-way twisted-pair cable provides a 0.4 x 0.4 pattern in the η-ϕ space;

- **Monitoring.** In the FE electronics of the calorimeters, the analogue signals from the individual cells are sampled and digitised at 40.08 MHz, and these values are then stored in pipeline buffers during the L1 trigger latency. Therefore, with the experimental hall closed, there is no possibility to examine the properties of the analogue cell signals at that stage. This is only possible in the Receiver system, via the analogue trigger sums. The Monitor part of the Receiver system provides the facility to pick off a programmable number of input trigger-tower signals, and display or measure them with an oscilloscope [Cle06], [Eis03].

A number of output signals from the Receiver system are routed to the input of a Receiver to PreProcessor Patch Pannels system (RPPPs) where further ordering of the trigger sums is performed [L1C07b]. The RPPP output and the rest of the output signals from the Receiver are then routed to the input of the PreProcessor system, the first processing stage of the L1Calo.

5.3 The PreProcessor

5.3.1 Tasks

The tasks assigned to the PPr system can be divided into *pre-processing* and *readout* tasks. The pre-processing tasks refer to the preparation of the analogue calorimeter trigger signals for digital processing in the CP and JEP system, and they can be summarised as follows:

- **Conditioning of the analogue input signals.** The trigger-tower pulses arrive at the input of the PPr as differential signals, with amplitudes of ± 1.25 V. The signals have to be converted to single-ended form, and an appropriate gain and baseline level has to be applied in order to map the signals into the digitisation window, i.e. 1 V.

- **Digitisation.** The analogue signals have to be digitised with 10-bit resolution at the bunch-crossing frequency of 40.08 MHz. Also, in order to improve the energy resolution, the digitisation strobe should be adjusted in such a way as to enable the sampling of the analogue pulses at their maximum amplitude.

- **Synchronisation of the trigger-tower data.** The trigger-tower signals arrive in the PPr asynchronously with respect to each other. This is due to the different time-of-flight of the particles from the interaction point to the calorimeter, and due to the different signal path-lengths from the detector to the input of the PPr. Therefore, after digitisation, the trigger-tower pulses originating from the same event have to be aligned in time, to allow the L1 trigger to make a proper decision with respect to each bunch-crossing.

- **Bunch-crossing identification (BCID).** The trigger-tower pulses are several bunch-crossings in width and have amplitudes proportional to E_T. For each pulse, the PPr has to find the peak maximum, extract the corresponding E_T and assign this value to a specific LHC bunch-crossing. The BCID algorithms have to be adapted to all pulses in the linear range 0-256 GeV and in the saturated region.

- **Final E_T calibration.** The output of the BCID algorithms is a 10-bit E_T value, which includes also the baseline level applied during the conditioning of the analogue pulses. This contribution has to be subtracted, and then the new E_T value has to be mapped from 10-bit to 8-bit, in order to reduce the data width on the links to the L1Calo processors. This leads to a nominal E_T resolution of 1 GeV per count.

- **Data preparation for transmission.** The PPr has to provide the JEP with 0.2 x 0.2 elements in order to reduce the number of links between the two systems. The jet data is obtained by summing four 8-bit E_T values from four adjacent trigger towers. Also, the summation has to be performed separately for electromagnetic and hadronic trigger

towers. The resulting value is 10 bits wide, and it has to be truncated to 9-bit data, with a resolution of 1 GeV per count. To the CP, the PPr has to provide the 8-bit calibrated E_T value. In order to reduce the number of links to the CP, the PPr has to multiplex into one serial stream the 8-bit E_T results from two trigger towers adjacent in ϕ.

- **Data transfer to L1Calo processors.** The pre-processing results have to be sent to the CP and JEP via 11 m long serial Low-Voltage Differential Signalling (LVDS) links, at a data rate of 400 Mbit/s. The PPr has to ensure the reliability of the transmission over the long cable links, so that the error rate is negligible.

The readout tasks refer to providing different data types to the DAQ and DCS systems in order to ensure a continuous controlling and monitoring of the system and of the trigger itself:

- **L1 accepted data.** This consists of multiple 10-bit digital samples which describe the development of each pulse, and of the corresponding 8-bit E_T calibrated values. These data are sampled at different points along the real-time pre-processing chain, and pipelined into dedicated memories. Upon the reception of the L1A signal from the CTP, the PPr has to extract the data related to the accepted event from the memories, and send it to the DAQ system.

- **L1 unbiased monitoring data.** The PPr has the possibility to compute energy rates and spectra for each trigger channel, based on the 10-bit digital samples or the 8-bit E_T values. The accumulation of this monitoring data as well as its retrieval from the system is done independently of the L1 decision.

- **Non-event based data.** This includes environmental parameters that ensure the operation of the hardware, i.e. supply voltages and temperatures across different parts of the system. These data are send to the DCS system.

5.3.2   Hardware Realisation

The PPr system consists of 124 hardware-identical *PreProcessor Modules* (PPMs), each of which being designed to process 64 analogue trigger-tower signals from four input cables. The PPMs are organised into eight crates[3]. Two of these crates are equipped with 14 PPMs, while the other six hold 16 PPMs. Section A.1 in appendix A gives additional details about the PPr crate organisation and coverage of the experiment (see e.g. figure A.5). Apart from PPMs, each crate holds also:

- one 9U *Timing Control Module* (TCM), which interfaces the PPr to the TTC and DCS systems: it receives from the TTC system LHC protocol signals, e.g. the 40.08 MHz bunch-crossing clock, the L1A signal, etc., and distributes them to all the PPMs in the crate, and forwards temperature and voltage information from the PPMs to DCS;

- one 6U *Single Board Computer (SBC)*, acting as the crate controller, to configure, control and monitor the PPMs over the VMEbus interface;

---

[3]9U VME crates with 21-slot VME64xP backplanes (VIPA-standard).
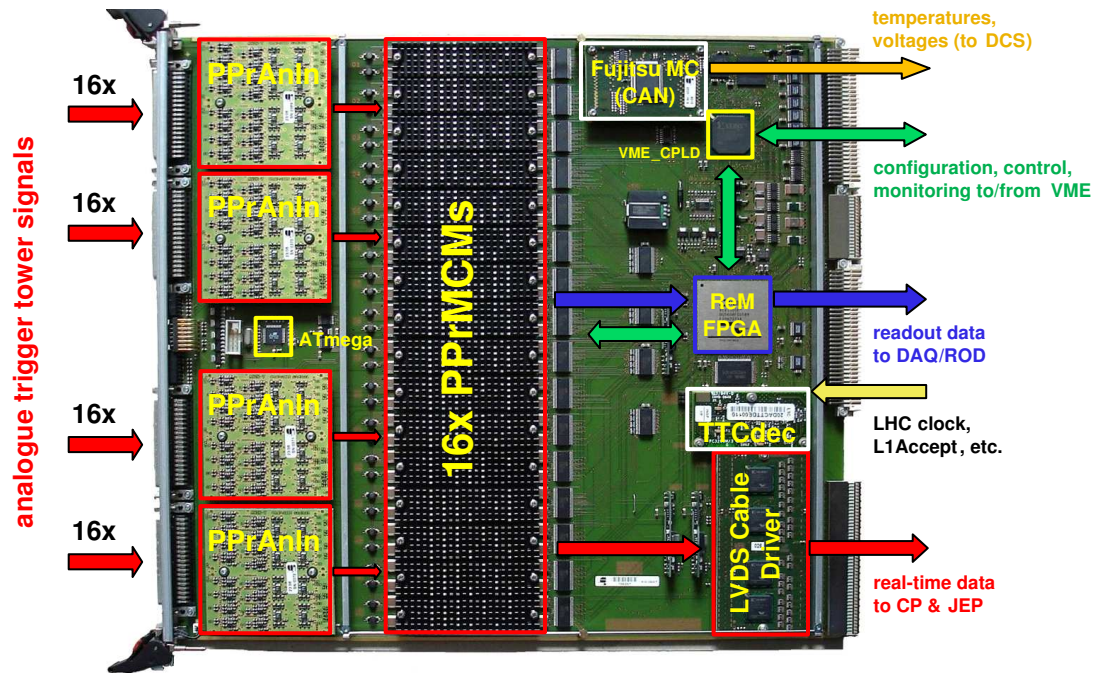
Figure 5.4: The PreProcessor Module.

- 14 or 16 *Rear G-Link Transmitter Modules - Optical Tx* (RGTM-Os), mounted on the back side of the PPr crate, to transfer the PPM event related data to a corresponding ROD module.

Figure 5.4 shows a photograph of a PPM. The module is configured as a 9U Printed Circuit Board (PCB) that mainly carries 23 daughterboards and several bonded programmable devices. The 64 input differential signals are received through four connectors on the front panel, and conditioned on four 16-channel *Analogue Input Boards* (PPrAnIns). The conditioned single-ended signals are then routed to 16 4-channel *Multi-Chip Modules* (PPrMCM), where the main signal processing takes place. The E_T results are first serialised and converted to LVDS form on the PPrMCMs, and sent to an *LVDS Cable Driver* (LCD) card which forwards them in real-time to the L1Calo processors. The event data is accumulated on the PPrMCMs. Upon the receipt of the L1A signal, the data related to the accepted event is sent to a *Readout Manager FPGA* (ReM_FPGA), which formats it and sends it to DAQ, via corresponding RGTM-O and ROD modules. The ReM_FPGA has also the tasks to transfer configuration and control data from the VME to on-board programmable locations, and to collect and deliver monitoring data to VME. The LHC protocol signals arrive on the PPM encoded into a single serial stream. A *Timing, Trigger and Control Receiver Chip* (TTCrx), located on a TTC Decoder (TTCdec) daughtercard, decodes the input stream and delivers the protocol signals to the ReM_FPGA, which further distributes them to multiple locations on the PPM. Lastly, a Fujitsu microcontroller collects information about temperatures and voltages on the PPM via an ATmega microcontroller, and sends them to DCS over a CAN-bus interface.
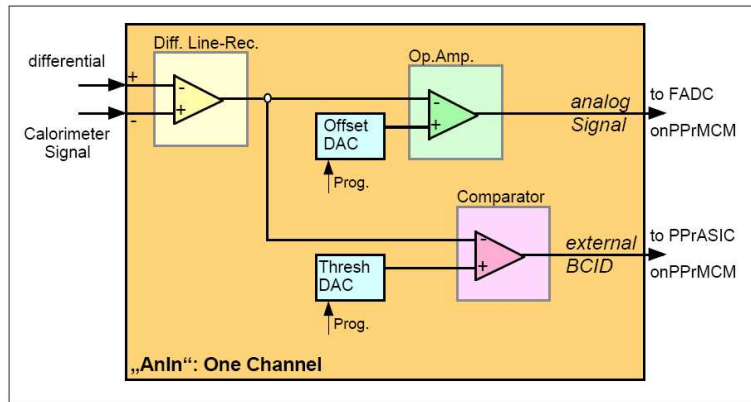
Figure 5.5: Analogue signal-handling in one PPrAnIn channel.

The following sections present briefly the implementation of the pre-processing and read-out tasks in hardware. Additional details about the functionality of the PPM will emerge from chapters 6 and 7.

Conditioning of the Analogue Input on the PPrAnIn Boards

The input differential signals are routed directly from the front-panel connectors to the four PPrAnIn boards. Each PPrAnIn receives and prepares for digitisation 16 input signals. Figure 5.5 describes schematically the analogue signal processing performed in one channel of a PPrAnIn board. The input analogue signal is first routed to a differential line-receiver, which converts it to a single-ended signal. Subsequently, the signal is fanned out into two copies. One copy is taken to an operational amplifier, which rescales the signal in order to match the digitisation window. The analogue trigger-tower signals have amplitudes up to 2.5 V, while the digitisation voltage range is 1 V wide[4]. The rescaling is achieved by applying a fixed signal-gain factor $G = 0.43$ for the operational amplifier. In the same time, an 8-bit programmable Digital-to-Analogue Converter (DAC) supplies a DC-level offset through the non-inverting input of the operational amplifier, in order to shift the baseline of the single-ended signal and to ensure that the amplitude of the signal is entirely visible to digitisation. The voltage offset can be set in steps of 2.4 mV, from 1.65 V, which corresponds to a DAC setting 0, up to 2.26 V (255). The output of the operational amplifier is then routed to a corresponding PPrMCM for digitisation and further processing.

The other copy of the initial single-ended signal is taken to the input of a comparator, which discriminates between the input signal and a voltage threshold provided by another 8-bit programmable DAC. The voltage threshold can be adjusted in steps of 10 mV. The output of the comparator is a digital signal, which is set to the logic value 1 whenever the input signal exceeds the voltage threshold, and to 0 otherwise. This means that the comparator marks the LHC bunch-crossing during which the signals is above the applied threshold. The digital signal, called *External BCID* (ExtBCID), is routed to a corresponding PPrMCM, where it is used by the BCID algorithms.

---

[4]2.4 V (V_{REF}) $\pm$ 0.5 V.
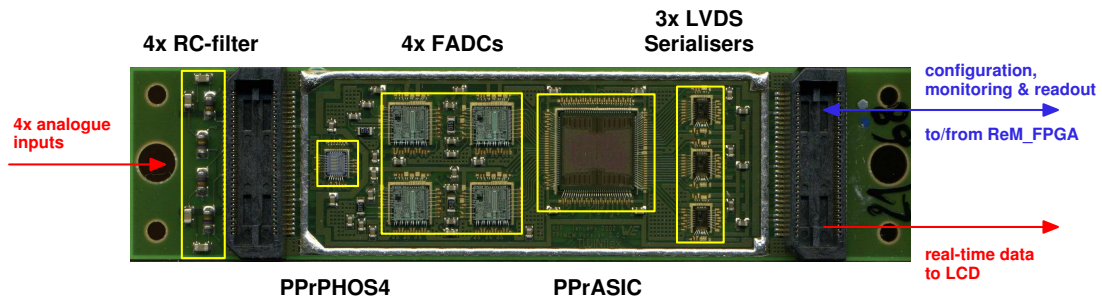
Figure 5.6: The PreProcessor Multi-Chip Module. In order to prevent oxidation, the dies and the wire bonds are covered with a glob-top material, and the area in between the connectors is encapsulated with a brass lid. In order to remove the heat produced inside the enclosed area, a heat-sink is mounted on the backside. This can be observed in figure 5.5.

The DACs on the four PPrAnIn boards are set over Serial Peripheral Interface (SPI) data buses. The configuration data is first loaded over the VME to the ReM_FPGA, and then transferred from the ReM_FPGA to the DACs over the SPI buses.

Signal Processing on the PPrMCMs

The main signal processing is performed on the 16 PPrMCMs. Each PPrMCM carries 9 wire-bonded *naked dies* that process four input single-ended trigger-tower signals (see figure 5.6):

- four *Flash Analogue-to-Digital Converters* (FADCs) to digitise the input signals;

- one *PHOS4 timing chip* (PPrPHOS4) to strobe the four FADCs;

- one *custom-built 4-channel ASIC* chip[5] (PPrASIC), which performs the trigger-specific data processing: synchronisation of the signals from the same bunch-crossing, BCID and final E_T calibration, and data preparation for transmission to L1Calo processors;

- three *LVDS serialisers* for serial transmission of the digital E_T values to the CP and JEP systems.

At the input stage of the PPrMCM, each analogue signal is viewed by a low-pass filter, to reduce the high-frequency noise component[6]. Subsequently, the signals are digitised by the FADCs, with 10-bit resolution[7] and at the bunch-crossing frequency of 40.08 MHz. The 1 V input voltage range of the FADCs corresponds to $0 - 256$ GeV E_T deposited in a given trigger tower. This means that one FADC count corresponds to $\sim$244 MeV.

---

[5]the chip was designed at the Kirchhoff-Institute for Physics, Heidelberg. Its die size is 8.370 mm x 8,375 mm using a 0.6 μm CMOS technology.

[6]the low-pass filter reduces also the amplitude of the input signal by about 20%. In order to compensate for this effect, the signals are actually rescaled on the PPrAnIn boards to match a voltage window of 1.2 V [Web08].

[7]in fact, the FADCs have a 12-bit resolution, but the two least significant bits are discarded, i.e. the output pins are not connected to the design.

The digitisation strobes are provided by the PPrPHOS4. The device allows also to adjust the phase of the strobes with 1 ns resolution, within the LHC bunch-crossing clock period of 25 ns, in order to sample the input signals at their maximum amplitude. The phase-adjustment can be configured separately for each digitisation strobe. The configuration data is loaded from the VME to the ReM_FPGA, and then transferred to the PPrPHOS4 over an Inter-Integrated Circuit (I^2C) data bus.

The resulting 10-bit digital signals are routed to the PPrASIC, where they are digitally pre-processed. The functionality of the PPrASIC is described in the next section. For the moment it should be mentioned that the real-time output of the PPrASIC consists of three 10-bit parallel data streams. Two of these streams provide calibrated E_T values for the CP, while the third stream provides jet E_T sums for the JEP. The data streams are taken to the input of the three LVDS serialisers, each of which serialises one stream to a rate of 400 Mbit/s, and transmits it to the LCD board.

Trigger-specific Data Processing on the PPrASIC

The four 10-bit FADC data streams are processed in parallel in the PPrASIC. Figure 5.7 describes schematically the data processing operations performed in one PPrASIC channel. At the input stage, the FADC data is first registered with the 40.08 MHz LHC clock. Because the phase of the digitisation strobes is displaced with respect to the LHC clock in the PPrPHOS4, the PPrASIC offers the possibility to latch the input data either on the positive or the negative edge of the LHC clock, in order to avoid metastability failures. The choice between the two options is done according to the size of the phase-displacement, i.e. the delay setting applied in the PPrPHOS4. After latching, the FADC data is passed through a 4-bit deep x 10-bit wide[8] synchronous FIFO[9], where it is delayed with a programmable number of bunch-crossings, in order to align it with the digital signal data processed in the other PPr channels. The depth of the FIFO corresponds to 16 bunch-crossings or 400 ns, which is sufficient to compensate for delays due to differences in cable length[10] as well as for delays induced in the electronics upstream of the PPr system.

The 10-bit output of the synchronisation FIFO is then fanned out and routed to the input of the BCID algorithms. The PPrASIC implements three methods for bunch-crossing identification: one method for non-saturated pulses, a second one for saturated pulses, and a third one that uses the output of the comparators on the PPrAnIn boards, i.e. the ExtBCID signals, to check the consistency of the first two methods. The first BCID method uses a digital pipelined Finite Impulse Response (FIR) filter to sharpen the pulse, and a peak-finder to identify the pulse maximum and mark the corresponding bunch-crossing. The FIR filter stores in internal registers five consecutive 10-bit samples: one from the current bunch-crossing, and other four samples from the previous and the following two bunch-crossings. These values are first multiplied with 4-bit configurable coefficients, and then the five resulting values are summed together. The first

---

[8]i.e. 16 locations, each of which 10 bits wide

[9]First-In First-Out memory buffer.

[10]the twisted-pair cables that carry trigger-tower signals, from the detector to the electronics in USA15, have lengths between 30 and 70 m. Thus, assuming a signal propagation delay in the twisted-pair cables of ∼5 ns/m, the maximum delay due to differences in cable length can be estimated to ∼200 ns.

and the last coefficients are signed, i.e. they can take values from -7 to +7, therefore the largest possible result yielded by the summation is 16-bit wide. This value (S_n) is then passed to a Peak-Finder module, which compares it with similar data obtained for the previous (S_{n-1}) and the following (S_{n+1}) bunch-crossings. If one of the following two conditions is fulfilled:

$$S_{n+1} \leq S_n > S_{n-1} \quad ,$$
$$S_{n+1} < S_n > S_{n-1} \quad , \tag{5.1}$$

then the PeakFinder asserts an 1-bit flag to mark the current bunch-crossing. Each of the two conditions presented above define an operational mode, and the choice between the two modes is done via a configurable PPrASIC parameter. The "BC Mark" flag is sent to the BCID Decision block, of which functionality is described a bit later.

At high luminosity, pulses describing various energy deposits pile up and give rise to saturated signals. In the calorimeter front-end electronics, the analogue saturation of the trigger-tower signals occurs at about 3 V. However, in the PPr the digitisation window of the FADC is limited to 2.5 V. This means that amplitudes equal to or higher than this value will be represented by the maximum 10-bit digitisation count, i.e. 1023. For very large saturated pulses, the saturation level extends over multiple consecutive bunch-crossing. In these cases, the peak-finder algorithm described above cannot reliably identify the correct bunch-crossing. Therefore, a separated BCID method was implemented for saturated pulses. The method assumes that the the shape of the rising edge is not distorted by signal saturation, and thus the peaking time is still ∼50 ns. In consequence, two samples before the virtual peak are located on the rising edge, the first sample pointing to the origin of the pulse. Upon detecting a saturated FADC value, i.e. 1023, the algorithm discriminates the samples from two previous bunch-crossings against two programmable thresholds, to determine the origin of the pulse. Based on the results provided by these comparisons, the peaking time is added to the origin of the pulse in order to determine the position of the virtual peak. The corresponding bunch-crossing is marked by asserting an 1-bit flag, which is then sent to the BCID Decision block. A more detailed description of the BCID method for saturated pulses is given in [Pfe99].

A copy of the 16-bit FIR filter output (S_n) is routed to a DropBits module, which truncates the input to a 10-bit data word. The selection of the 10 bits is done according to the value of a programmable StartBit parameter. In his turn, the StartBit has to be defined according to the settings applied for the FIR filter coefficients, in order to maximise the energy resolution. For example, if the FIR filter coefficients are set to {0,0,1,0,0}, then the six most significant bits (MSBs) of the 16-bit S_n sum will always be set to zero, while the other 10 bits will reflect the FADC data for the current bunch-crossing. In this case, the DropBits module has to be be configured to always cut off the six MSBs. The resulting 10-bit data is then sent to a Look-Up Table (LUT), which extracts the final E_T for the trigger algorithms. The LUT is a 10-bit deep memory buffer that stores 8-bit configurable E_T values. The 10-bit input data is used as memory address, so that the 8-bit LUT output reflects the memory content indicated by the input address. The 8-bit E_T values are set in such a way as to allow for pedestal subtraction, noise suppression and final E_T calibration in the same time. The 8-bit LUT output is sent to the BCID Decision block.

The task of the BCID Decision logic is to allocate the three BCID algorithms to energy

"BC Mark" Register (3-bit)			BcidDecision Registers (8-bit)			
PeakFinder	Saturated	ExtBCID	BitNr	Reg1	Reg2	Reg3
0	0	0	0	0	0	0
0	0	1	1	1	1	0
0	1	0	2	1	0	0
0	1	1	3	1	1	0
1	0	0	4	1	1	1
1	0	1	5	1	1	1
1	1	0	6	1	1	1
1	1	1	7	1	1	1

Table 5.1: The combinations of the three "BC Mark" bits and their relation to the bit number of the three BcidDecision registers. Note that the first BcidDecision register is allocated to the upper energy region, the second to the middle region and the third to the lower region. Also, all three BcidDecision registers are configurable. The 8-bit values given in this table represent default settings for these registers, and they are shown here only as an example.

levels. For this, the module divides the 10-bit energy range into three regions, via two programmable energy thresholds, and discriminates a 10-bit energy input against these thresholds. The energy input is given either by the 10-bit output of the synchronisation FIFO or the 10-bit output of the DropBits module, the choice between the two data sources being done via a DecisionSource configurable parameter. In the same time, the BCID Decision maps the 1-bit "BC Marks" received from the PeakFinder and Saturated algorithms and the ExtBCID signal into a 3-bit register, after they were previously synchronised. The resulting 3-bit value indicates the bit number of three 8-bit configurable BcidDecision registers, each register being allocated to an energy region. For example, if the PeakFinder result is 1, the Saturated result is 0 and the ExtBCID bit is 1, then this will indicate the fifth bit of the BcidDecision registers (see e.g. table 5.1). If the respective bit of any of these registers is set to 1, and the 10-bit energy input falls in the related energy range, then the BCID Decision module validates the 8-bit calibrated E_T value received from the LUT. Otherwise, if these conditions are not met, the BCID Decision sets the same E_T value to zero. This also means that the off-peak digital samples will always be set to zero.

Data Preparation and Transfer to the L1Calo Processors

These E_T results of the BCID Decision logic, called in the following BCID-LUT values, are sent to the output stage of the PPrASIC, where they are prepared for transmission to the L1Calo processors. For the CP, the PPrASIC multiplexes at each bunch-crossing clock tick the 8-bit BCID-LUT data from two channels into one 10-bit parallel data stream. This leads to two PPrASIC output data streams for the CP. For the JEP, a copy of the 8-bit BCID-LUT data from all four channels is summed into one 9-bit 0.2 x 0.2 data word. This is achieved in two steps, first by pre-summing the data from two channels, and then by summing the two preliminary results and

Figure 5.7: Block-diagram of pre-processing, readout and monitoring operations in one PPrASIC channel (modified from [The05]).

One PPrASIC channel

truncating the final 10-bit sum to 9 bits. The bunch-crossing multiplexing and jet sum algorithms are described in section 7.2.5. Also, the mapping of the PPrASIC channels in the output streams to CP is explained in section A.2.

The three 10-bit parallel data streams are routed from the PPrASIC to the PPrMCM-LVDS serialiser chips, which convert the input data to LVDS serial data streams and send them to the LCD daughtercard. The LCD receives in total 48 serial streams from the 16 PPrMCMs. Its tasks are to duplicate the signals near the ϕ edges of the PPM, in order to allow the algorithms of the L1Calo processors to scan efficiently the boundaries of the quadrants in azimuth, and to drive all the signals over 11 m long cables to the object-finding processors. For the latter task, the LCD applies an RC pre-compensation, to minimise the degradation of the signals during the transmission over the long LVDS cables.

Readout and Monitoring Operations on the PPM

The event-based data that the PPr has to provide to DAQ consists of 10-bit raw FADC data, 8-bit BCID-LUT data and the 1-bit "BC Marks" provided by the three BCID algorithms. This data is continuously accumulated in two pipeline memories in each PPrASIC channel (see PipeMem-ADC and PipeMem-LUT in figure 5.7). Upon the receipt of the L1A signal from the CTP, each PPrASIC copies a programmable number of event data words from the pipeline memories into corresponding derandomiser buffers, and then transfers the respective data to the ReM_FPGA, over two serial interfaces. The ReM_FPGA assembles the event data from all 16 PPrASICs in a pre-defined ATLAS format, and transfers it to the DAQ. For various testing and debugging applications, the ReM_FPGA keeps in local memory buffers an unformatted copy of the PPrASIC data, from where it can be accessed over the VME interface.

The PPrASICs have also the capability to accumulate channel-wise energy deposition rates and distributions of deposited E_T above programmable thresholds, based either on the 10-bit raw FADC data or the 8-bit BCID-LUT data. The respective algorithms are implemented in two modules, called RateMetering and Histogramming. The monitoring data produced by these modules can be read out from the PPrASIC independent of the L1 trigger decision. The task to retrieve this data from all the 16 PPrASICs is assigned to the ReM_FPGA. The device places the monitoring content into an on-board Static RAM (SRAM) memory, from where it can be accessed over the VME. The implementation of the Rate Metering and Histogramming tools in the PPrASIC, the content of the data they provide and the mechanism through which this data is read out from the PPrASICs are described in section 6.6. Also, the functionality of the ReM_FPGA is presented in great detail throughout the next chapter.

The PPrASICs can also inject digital test-vector data in the pre-procesing chain, in order to allow a functional verification of the digital part of the L1Calo system. The test-vector data is loaded from the VME, over the ReM_FPGA and the serial interfaces, to dedicated 11-bit wide playback memories in the PPrASICs. Upon the activation of a corresponding operational mode, the PPrASICs insert the content of the playback memories into the synchronisation FIFOs, replacing the 10-bit FADC data and the 1-bit ExtBCID signal.

Last but not least, the PPMs provide temperature and supply voltage values to the DCS system, in order to allow the monitoring of the physical aspects of the hardware. The temperatures are measured by means of diodes implemented in the PPrASICs and in the ReM_FPGA, while

the supply voltage values are provided by an on-board power manager. These data are received by the ATmega microcontroller, which provides them to the Fujitsu microcontroller, for transmission to DCS, and to the VME for local monitoring and debugging purposes.

5.4 The Cluster Processor
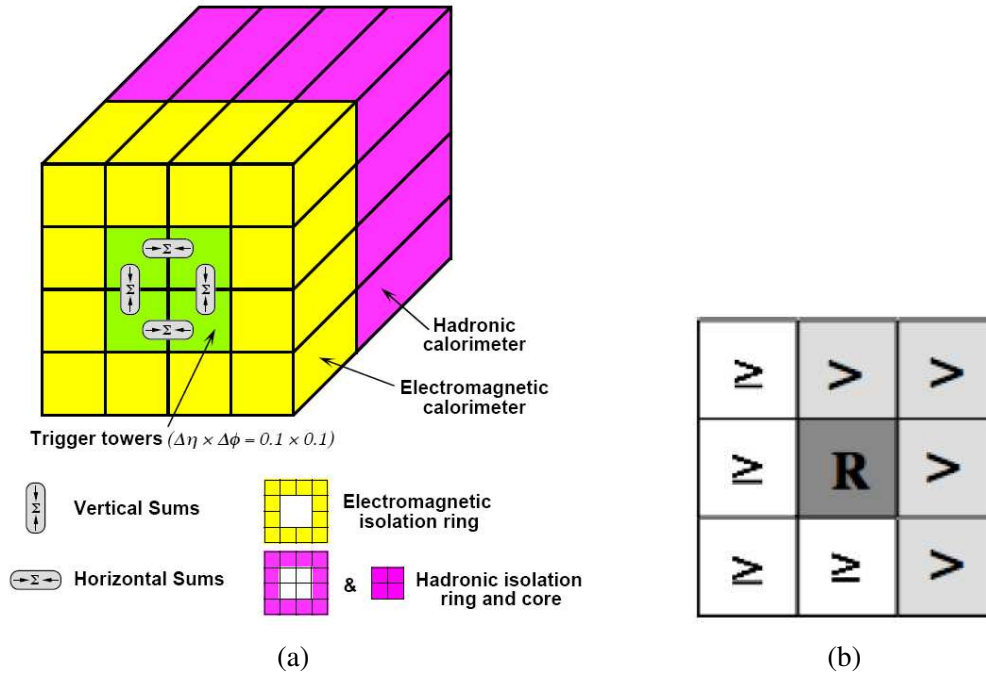
5.4.1 Trigger Algorithms

The Cluster Processor identifies isolated electrons, photons and hadronic τ decays, and computes multiplicities of identified e/γ and τ candidates that pass various E_T threshold conditions. These investigations use the 8-bit calibrated trigger-tower E_T values provided by the PPr system, and they extend out to $|\eta| < 2.5$, which is the limit for precision measurements with the Inner Detector and the electromagnetic calorimeters.

The e/γ and τ/hadron algorithms are based on a window of 4x4 trigger towers in both the electromagnetic and hadronic calorimeter layers (see figure 5.8a). The window is divided into two regions: a *core*, formed by the central 2x2 trigger towers, and an *isolation ring* given by the remaining 12 towers. The algorithms slide the window by one tower in both the η and ϕ directions, within the allocated trigger space. At each step, several E_T sums are computed and evaluated against programmable thresholds.

The e/γ algorithm searches for narrow high E_T electromagnetic showers that extend over maximum two trigger towers, and that do not penetrate the hadronic calorimeter, while the τ/hadron algorithm searches for τ decays into collimated clusters of hadrons that lead to energy depositions in both the electromagnetic and hadronic calorimeters. For this, the algorithms first compute the following E_T sums:

- four *electromagnetic clusters*, to measure the E_T of the candidate trigger object. These are obtained by summing each two adjacent towers in the electromagnetic core, and they will be in the following referred to as S_1;

- one *hadronic core cluster*, by summing the four towers of the hadronic core (S_2);

- four *hadronic clusters*, by summing each of the four S_1 sums with the S_2 sum (S_3);

- one *electromagnetic isolation sum*, by summing the 12 towers of the corresponding isolation ring (S_4).

- one *hadronic isolation sum*, by summing the 12 towers of the corresponding isolation ring (S_5);

These sums are then compared against programmable E_T thresholds in order to identify an e/γ or a τ/hadron trigger candidate object within the given algorithm window. The following

(a)   (b)

Figure 5.8: Schematic view of the trigger algorithms performed by the Cluster Processor: the sliding window of 4x4 towers and the E_T sums used for the e/γ and τ/hadron algorithms (a), and the RoI declustering (b).

conditions must be fulfilled:

$$
\begin{aligned}
S_1 &> E_T^{em,cluster} & , \\
S_2 &\leq E_T^{had,core} & , \\
S_3 &> E_T^{had,cluster} & , \\
S_4 &\leq E_T^{em,isolation} & , \\
S_5 &\leq E_T^{had,isolation} & ,
\end{aligned}
\tag{5.2}
$$

where the first and the third conditions are sufficient to be fulfilled by only one S_1 sum or one S_3 sum respectively, while the second condition is applied only to the e/γ algorithm, in order to ensure that the shower is contained in the electromagnetic calorimeter.

The algorithms contain 16 sets of programmable E_T thresholds, each set representing a combination of cluster and isolation thresholds. Eight of these thresholds are directly assigned to the e/γ algorithm, while the other eight can each be programmed to be used either by the e/γ or by the τ/hadron algorithm.

When a candidate object is identified, a corresponding 3-bit multiplicity counter is incremented[11]. However, because the algorithm window is slid by one tower in both the η and ϕ

---

[11] when the counter is set to its maximum value, i.e. 7, no further incrementation is permitted.
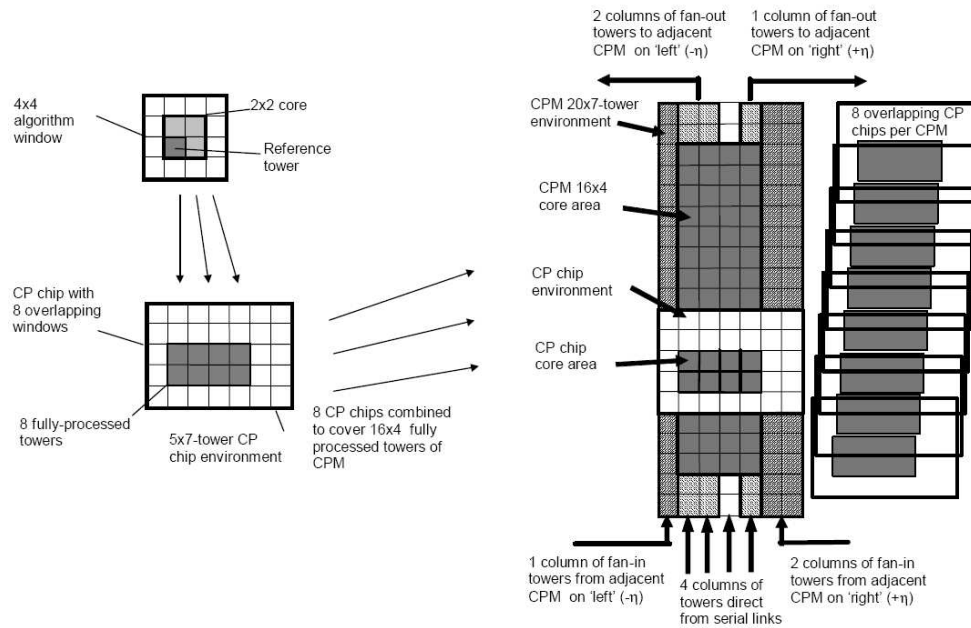
Figure 5.9: The handling of the trigger-tower data on the CPM [Hil06].

direction, there is a significant probability to identify the same object in more than one overlapping window. In order to avoid the multiple counting of a candidate object, an additional condition is imposed for both the e/γ and τ/hadron algorithms. The central 2x2 towers of both the electromagnetic and hadronic cores are first summed together to form a *cluster RoI*, and this is then required to be a local maximum compared with its eight overlapping nearest neighbours, as shown in figure 5.8b. This requirement is sometimes referred as to *declustering*. The (η,ϕ) coordinates of the local maximum are then included in the RoI information for the L2 trigger. The 16 multiplicity results are sent to the CTP [L1C04], [L1C08a].

5.4.2 Hardware Realisation

The Cluster Processor is a four crate system. Each crate processes 8-bit calibrated E_T values from one ϕ quadrant, and hosts 14 *Cluster Processor Modules* (CPMs) that carry out the cluster-finding algorithms, two *Common Merger Modules* (CMMs) that produce system-wide results, and one TCM to distribute the LHC protocol signals in the crate and to transfer environmental parameters to DCS.

Each CPM can process 64 algorithm windows arranged in an array of 4x16 (ηxϕ) trigger towers. In order to form the overlapping windows, each CPM uses additional trigger-tower information from regions adjacent in η and from the boundaries of the corresponding ϕ quadrant, so that the trigger space viewed by each module consists of 7x20 (see figure 5.9). Since the cluster-finding algorithms require data from both the electromagnetic and hadronic calorimeters, this means that each CPM receives E_T data from a total of 280 towers. Out of these, 160 E_T values are received directly from the PPr system via 80 serial LVDS links. The remaining
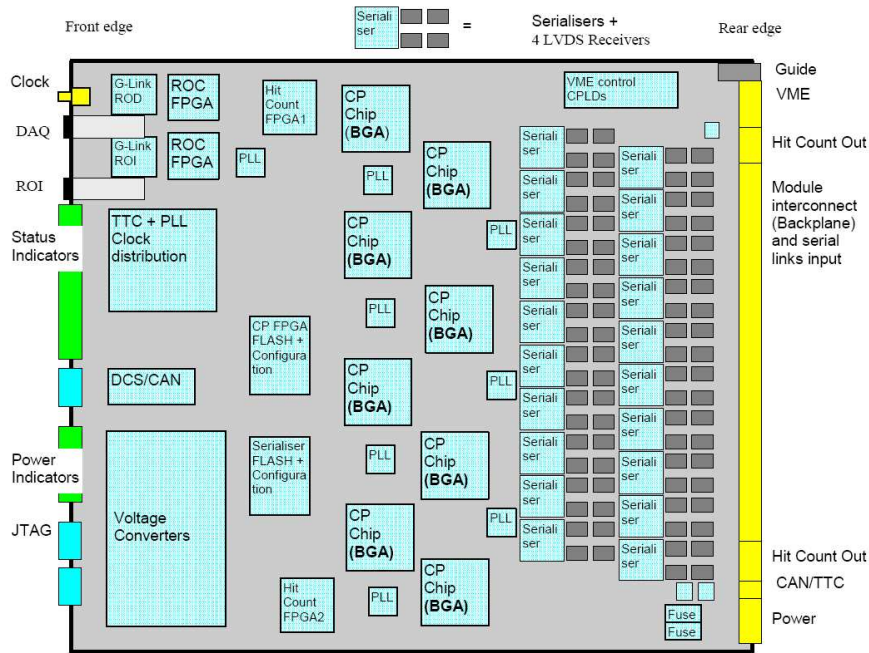
Figure 5.10: Block-diagram of a Cluster Processor Module [Hil06].

120 E_T values are received from the neighbouring CPMs within the same crate, over the backplane. The cluster-finding algorithms are carried out on each CPM by eight *CP FPGAs*. Each device processes eight overlapping windows arranged in a 4x2 array of trigger towers within a 5x7 region (see again figure 5.9).

The CPMs are implemented as 9U PCBs. Figure 5.10 shows a block-diagram of the CPM. The 10-bit serial data streams from the PPr enter the module through the crate backplane. They are first converted to parallel form, and then routed to 20 *Serialiser FPGAs*, which re-serialise the data at 160 Mbit/s and transmits it to the CP FPGA. Three quarters of the resulting data streams are duplicated and sent to the neighbouring modules over the crate backplane. The CP FPGAs de-serialise and de-multiplex the input data streams and process the trigger-tower information. The results of this processing indicate which thresholds were passed, and they are sent to two *HitCount FPGAs*. These devices compute overall multiplicity results for an entire CPM, and transmit these data to the two CMMs in the crate. In order to monitor the performance of the CPM, one *Readout Controller* (ROC) FPGA provides to the DAQ trigger-tower E_T values, as received by the Serialisers FPGAs from the PPr, and processing results from the CP FPGAs. A second ROC FPGA is employed to transmit RoI information to the L2 trigger. The transfer of readout data to DAQ is initiated upon the receipt of an L1A signal from the CTP. The L1A signal and the other LHC protocol signals received via the TCM are decoded by a TTCdec card, identical to the one used on the PPM. Also, a Fujitsu microcontroller collects temperature, supply voltage and current information from the board, and provides it to the DCS, via a CAN-bus on the backplane and the TCM in the crate.

The two CMMs produce overall crate results, by summing the multiplicity data from all
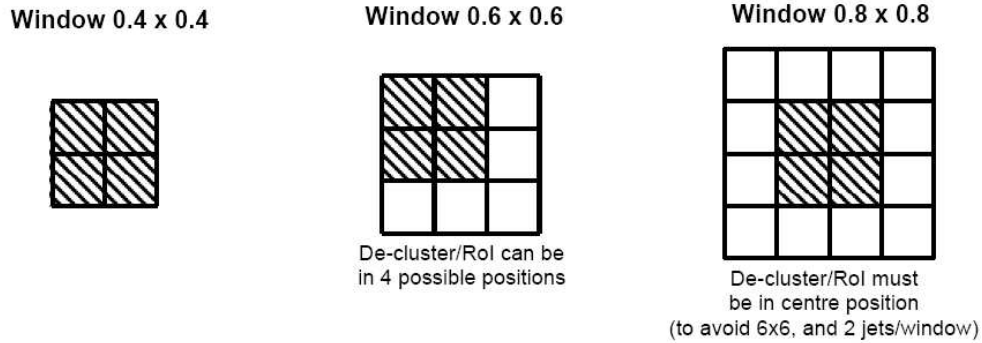
Figure 5.11: The algorithm windows of the jet trigger. The shaded areas indicate the RoIs.

CPMs. One CMM in the four CP crates combines the results from all four crates to produce system-wide results, which are then send to the CTP. For testing and debugging purposes, the module, crate and the system results are provided to the DAQ [Hil06], [L1C08a].

5.5 The Jet/Energy-sum Processor

5.5.1 Trigger Algorithms

The Jet/Energy-sum Processor identifies hadronic jets and computes global sums of total, missing and jet-sum E_T, based on the $\Delta\eta$ x $\Delta\phi = 0.2$ x 0.2 jet sums provided by the PPr system[12]. The jet algorithm covers the region $|\eta| < 3.2$, which represents the limit of the end-cap acceptance[13], while the total and missing E_T algorithms cover the entire ATLAS trigger space.

The Jet Trigger

The jet algorithm is based on a 0.2 x 0.2 *jet element*, which is formed by summing the electromagnetic and hadronic 9-bit E_T sums received from the PPr. The jet element is used to construct *jet windows* of different sizes: 2x2, 3x3 or 4x4 jet elements, i.e. 0.4 x 0.4, 0.6 x 0.6 and 0.8 x 0.8 in $\Delta\eta$ x $\Delta\phi$ (see figure 5.11). The windows are slided in steps of one jet element in both the η and ϕ directions. At each step, a total jet E_T is computed for each window by summing the jet elements, and this energy value is then evaluated against programmable thresholds.

As for the e/γ and τ/hadron algorithms, a local maximum test is performed in order to avoid counting multiple times the same object. For this, a 2x2 *jet RoI* is constructed by summing four adjacent jet elements. For a jet window with 2x2 or 4x4 jet elements there is only one possible window for each RoI, while for a window with 3x3 jet elements four possible windows surround

---

[12]note that the PPMs that process trigger-tower signals from the end-cap regions $2.9 < |\eta| < 3.2$ provide 0.3 x 0.2 jet sums, due to the trigger-tower granularity in those regions (see e.g. figures A.3, A.4 and A.5).

[13]a jet algorithm for the FCAL regions, i.e. $3.2 < |\eta| < 4.9$, is currently not implemented, but foreseen. The possible scenarios for a trigger on forward jets are presented in [L1C04].

each RoI (see again figure 5.11). In this latter case, only the window with the highest E_T sum is used for the local maximum test.

The jet algorithm consists of eight sets of thresholds, each of which is a combination of a jet E_T threshold and a choice of a jet window size. When a local maximum is found and the total E_T sum of a jet window is greater than the applied threshold, a corresponding 3-bit multiplicity counter is incremented. The coordinates of the jet RoI are passed to the L2 trigger, while the eight multiplicity results are sent to the CTP.

Energy-sums Triggers

The *total E_T* is computed by summing the E_T of all jet elements. The resulting energy value is compared against four thresholds, which can take values up to 2 TeV in 4 GeV steps. Each comparison produces an 1-bit result, which is then sent to the CTP.

For the *missing E_T* trigger, it is assumed that the vectorial sum of the measured E_T equals the same sum of the undetected E_T (E_T^{miss}). Therefore, the E_T^{miss} is estimated according to the formula:

$$E_T^{miss} = \sqrt{\left(\sum_{j=0}^{N} E_{x,j}\right)^2 + \left(\sum_{j=0}^{N} E_{y,j}\right)^2} \qquad , \qquad (5.3)$$

where N represents the total number of jet elements, while $E_{x,j}$ and $E_{y,j}$ represent the projection of the measured E_T on the x- and y-axis respectively for a given j jet element. The latter are calculated as follows:

$$E_{x,j} = E_T \cdot cos(\phi_j) \quad ,$$
$$E_{y,j} = E_T \cdot sin(\phi_j) \quad , \qquad (5.4)$$

where ϕ_j represents the azimuthal coordinate of the given jet element.

Thus, the E_T^{miss} algorithm multiplies first the E_T of each jet element with $cos\phi$ and $sin\phi$ to obtain the transverse energy components E_x and E_y, and then determines the global $\sum E_x$ and $\sum E_y$ values. Lastly, a LUT is employed to perform the final quadrature addition and to compare with eight thresholds in a single step. As for the total E_T algorithm, each comparison produces an 1-bit result, which is then sent to the CTP.

The *total jet E_T* is based on the eight 3-bit multiplicity results of the jet trigger algorithm. The jet counts are converted to transverse energy values by multiplying them with appropriate factors that take into account that a jet candidate could have passed more than one E_T threshold. The resulting E_T values are then summed and compared against four thresholds, and the results are sent to CTP [L1C04], [L1C08a].

5.5.2   Hardware Realisation

The Jet/Energy-sum Processor is a two crate system. Each crate processes 9-bit jet E_T sums from two opposite ϕ quadrants (*back-to-back*), and hosts 16 *Jet/Energy-sum Processor Modules* (JEMs) that carry out the jet and the energy-sum algorithms, two CMMs and one TCM.

Figure 5.12: Block-diagram of a Jet/Energy-sum Processor Module [Sch08].

Each JEM processes a *core* region of 4x8 (ηxϕ) jet elements. As for th CP, in order to form the overlapping windows, the jet algorithm requires additional information from the neighbourhood of the core region. Therefore, the effective region *scanned* by each JEM is 7x11 jet elements wide. Out of the total of 77 jet elements, 44 are received directly from the PPr, the remaining 33 jet elements being received from the neighbouring modules in the same JEP crate.

The JEM is also a 9U PCB. Figure 5.12 shows a block-diagram of the JEM. Each JEM received data from the PPr over 88 serial LVDS links. Out of these, 44 streams provide 0.2 x 0.2 electromagnetic jet sums, while the other 44 provide similar hadronic jet sums. The 10-bit input data enter the module though the crate backplane, and they are routed to four input daughtercards, named as R,S,T,U in the same figure 5.12. Each input daughtercard can handle 24 data streams, and consists of four 6-channel LVDS deserialisers and one *Input FPGA*. The deserialisers convert the 10-bit input data to parallel form. The Input FPGAs extracts the 9-bit jet sums, and forms 10-bit jet elements by summing corresponding electromagnetic and hadronic jet sums. The jet elements are then multiplexed at 80 MBit/s and sent to a *Jet FPGA*, which performs the jet-finding and the total jet E_T algorithms. Also, the Input FPGAs compute the $E_{x,j}$ and $E_{y,j}$ values (see again equation 5.4) and the jet element E_T sum, and send them to the another processor FPGA, called *Sum FPGA*, which carries out the other two energy-sum algorithms. Additionally, the Input FPGAs duplicate 3/4 of the jet elements and send them via the crate backplane to the neighbouring modules. The multiplicity counts computed by the Jet FPGA and the Sum FPGA are sent to the two CMMs in the crate.

As the CPM, the JEM has two Readout Controller (ROC) FPGAs which collect and transmit

readout data to the DAQ and RoI information to the L2 trigger. The modules are hosted on a readout daughter module (RM). The readout data consists of jet sums received from the PPr, and jet and energy-sum results. The JEM hosts also a TTCdec card, to decode the LHC protocol signals, and a CAN controller to provide environmental parameters to DCS.

The two CMMs in the crate have a hardware design identical to the CMMs used by the CP system, but they run different versions of firmware on local FPGAs. This is possible because both CP and JEP use a common custom backplane. As in the CP case, the two CMMs produce crate-level multiplicity results, and one CMM in the two crates merges the results from the two crates to produce system-wide results. In contrast to the CP, the CMMs of the JEP system do not send data to DAQ, but to the L2 trigger. These consists of total values of E_x, E_y and E_T and the threshold results from the total E_T, E_T^{miss} and the total jet E_T algorithms [Sch08], [L1C08a].

Chapter 6

The Readout Manager of the PreProcessor Module

The main task of the PPr system is to prepare the 7168 analogue trigger-tower signals for digital processing at subsequent stages in the L1Calo. The input signals are conditioned and converted to digital form, and a transverse energy (E_T) value is extracted from each pulse and assigned to the correct bunch-crossing. These digital E_T values form the basis of the trigger decision. They are transmitted in *real-time* to the CP and JEP systems, which use them to identify *small* and *large* physics objects and to compute global energy sums. Subsequently, the two processors count *hit* multiplicities of the different types of trigger objects, and send them to the CTP. Based on this information, as well as on information received from the Muon Trigger, the CTP takes a decision with respect to each event. If positive, then the CTP distributes the L1A signal to all the ATLAS sub-detectors and readout electronics systems, and the finely-granular detector data related to the accepted event is examined by the next levels of the ATLAS trigger system.

This means that the bulky analogue input from the calorimeters is reduced at the output of the L1 trigger to only one bit, i.e. the L1A, and that the analogue calorimeter trigger signals are available only in the PPr system. Therefore, in order to allow verification of the trigger decision and of the operation of the L1Calo components, the PPr must provide *real-time* digital values related to the accepted event to the DAQ system. Raw FADC data and calibrated E_T values are permanently extracted from the pre-processing chain and pipelined into separated scrolling memories (see figure 6.1). Upon the receipt of the L1A signal, each PPM reads out the event related data from the memories, and sends it to the DAQ via an external ROD module. This operation is handled by an on-board Field-Programmable Gate Array (FPGA) based device, called Readout Manager (ReM_FPGA). The device collects the event data from the distributed locations, processes it in the specified ATLAS format, and provides it to a Rear G-Link Transmitter Module - Optical Tx (RGTM-O), mounted on the back side of the PPr crate, for high speed serialisation and transmission to the ROD.

In addition to the readout path, the PPr must provide an interface to the computing infrastructure of the experiment, for configuration, control and monitoring of the system. The interface is realised via a standard Versa Module Eurocard data bus (VMEbus). The PPM is configured as a VME slave, and its communicating partner on the VMEbus is the crate controller CPU mounted
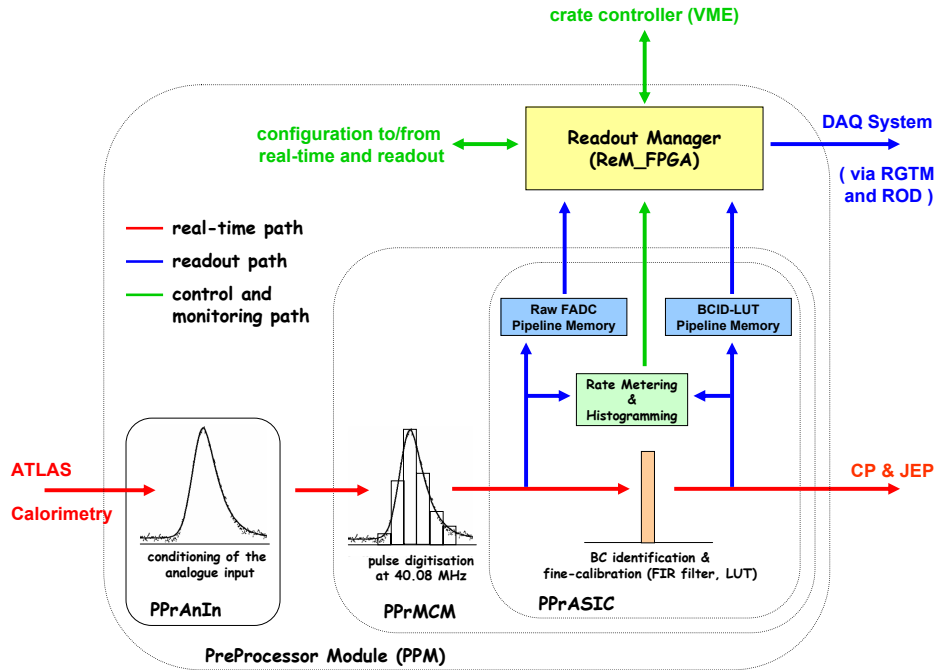
Figure 6.1: Simplified view of the main data paths on the PreProcessor Module.

in the first slot of the PPr crate. The interface is mainly used for setting up the PPM. All the PPM on-board components provide sets of configurable parameters to allow flexibility in steering or debugging the operation of the system. Apart from that, the interface is used for retrieving the monitoring data produced by the PPrASICs. Rates and energy spectra, of *real-time* raw FADC or calibrated energies, are continuously accumulated for each trigger channel, independent of any trigger decision. These data can be read out periodically from the system, and used for monitoring the activity in the ATLAS calorimeters. Last but not least, the interface is used for testing and debugging the operation of the PPM in a *laboratory environment*. Unformatted copy of the event data is made available at low bandwidth to VME, to compensate for the absence of the DAQ *requisites*.

As for the readout case, the tasks of collecting and transferring data from VME to PPM on-board components, and in reverse direction, are delegated to the ReM_FPGA. This chapter describes in detail the functionality of the ReM_FPGA and the implemented algorithms. The device is a Xilinx VirtexE (XCV1000-E) FPGA, and its behaviour was defined using the Verilog hardware description language (HDL).
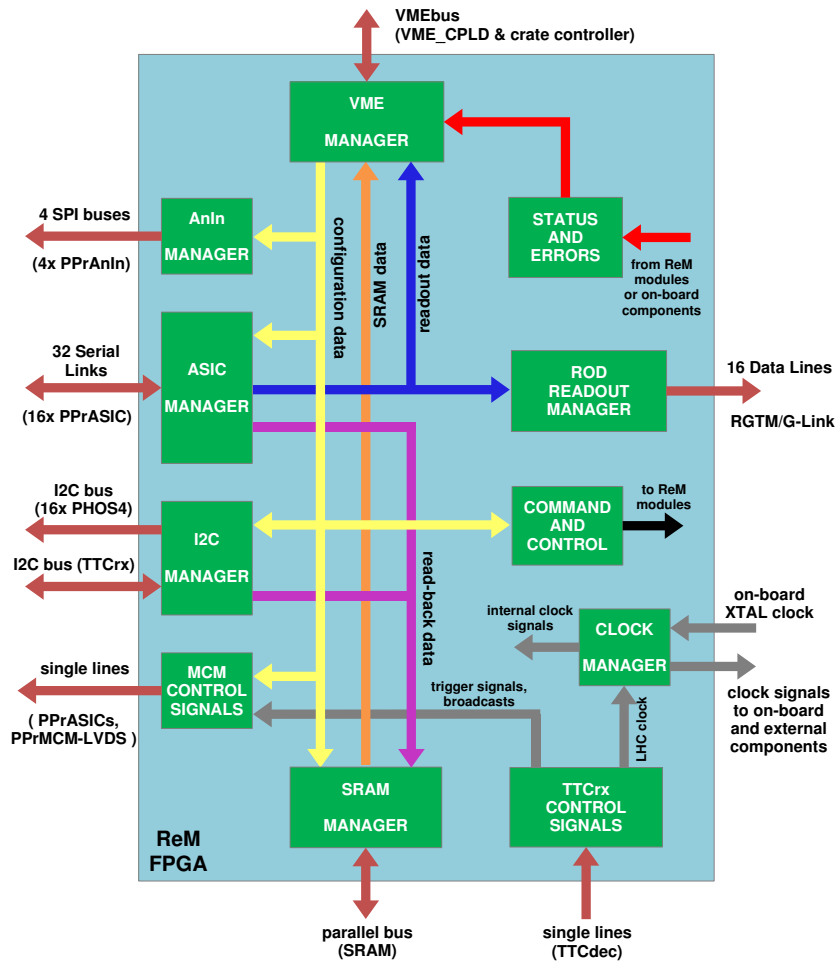
Figure 6.2: Block diagram of the of the ReM_FPGA.

6.1 Functional Overview

Figure 6.2 shows a block diagram of the ReM_FPGA. The device interfaces to several distinct communication systems, to receive and transmit data from and to multiple PPM on-board and external components:

- a VME data bus connects the ReM_FPGA with the crate controller CPU. Software applications, running on the crate controller, transfer configuration data over the VMEbus to the ReM_FPGA, which then distributes it to on-board destinations. Same applications can instruct the ReM_FPGA, via command and control registers, to collect readback data (i.e. configuration data from the sources and PPrASIC monitoring data) or PPrASIC event related data, and to place it in locations *visible* to VME. Additionally, the controlling software can access over the VME information about the operational state of the PPM. The ReM_FPGA gathers into dedicated VME registers bitwise status and error data received

from the interfaced devices or generated by its internal algorithms;

- four unidirectional Serial Peripheral Interface (SPI) data buses are accessed by the ReM_FPGA to transfer configuration data to the DACs mounted on the four PPrAnIn boards;

- two Inter-Integrated Circuit (I^2C) data buses provide the ReM_FPGA with access to the 16 PPrPHOS4s and to the Timing, Trigger and Control Receiver Chip (TTCrx), located on the TTC Decoder (TTCdec) daughtercard. The ReM_FPGA transfers configuration data to these devices, and reads back configuration data only from the TTCrx (the PPrPHOS4 is not *readable*);

- 32 bidirectional serial data links interface the ReM_FPGA to the 16 PPrASICs. The ReM_FPGA writes trigger configuration data to the PPrASICs, and receives, in a multiplexed stream, readback, readout and status data;

- one bidirectional parallel data bus connects the ReM_FPGA with an on-board Static RAM (SRAM) device. The SRAM acts as a physical extension to the ReM_FPGA's internal memory resources. The ReM_FPGA stores in the SRAM copies of the settings loaded from VME and data read back from the PPrASICs and the TTCrx;

- 16 unidirectional data lines are used by the ReM_FPGA to transfer the processed PPrASIC event related data to the ROD, via the RGTM-O;

- multiple single data lines provide the ReM_FPGA with control and status data from the communicating devices, or are used by the ReM_FPGA to transmit similar information to the same devices;

Apart from these, the ReM_FPGA has the task to supply its communicating partners with clock pulses. The ReM_FPGA receives these signals from two PPM components. The first one is an on-board crystal oscillator (PPM_XTAL), which provides a 40.00 MHz pulse train. The second component is the TTCdec card, which provides the 40.08 MHz LHC Bunch-Crossing Clock or, when the LHC clock is not available, a 40.00 MHz clock pulse from a local crystal oscillator. The PPM_XTAL clock is provided to the RGTM-O, as part of the transport protocol of event data to this device, and drives the related logic in the ReM_FPGA. The clock pulse supplied by the TTCdec card is distributed to all devices on the *real-time* data path, and it drives most of the ReM_FPGA's internal logic. Additionally, the input clock from the TTCdec card is used by the ReM_FPGA to generate a clock pulse for the transport protocol of the I^2C and SPI data buses, i.e. 100 kHz and 2 MHz respectively.
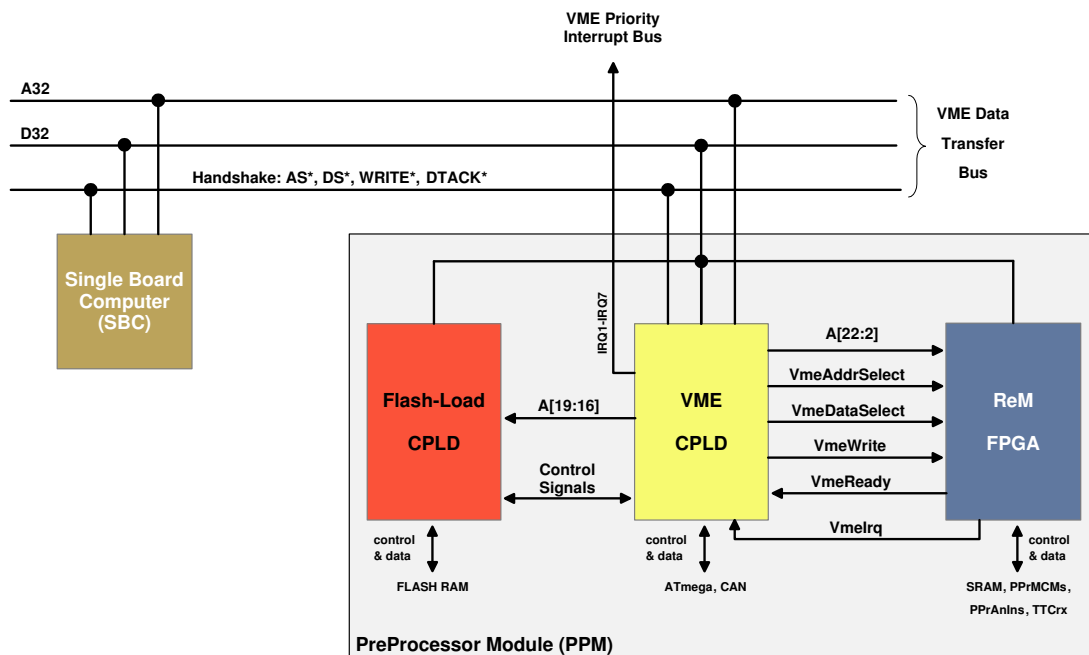
6.2 Communication with the On-board and External Devices

6.2.1 The Interface to VME

Each PPr crate is equipped with a Single Board Computer[1] (SBC) which serves for the purposes of setting up, controlling and monitoring the operation of the PPMs. The module is connected

---

[1]VP315 from Concurrent Technologies

Figure 6.3: Schematic representation of the VME implementation for the PreProcessor System.

to all PPMs via a VMEbus system, on the backplane of the PPr crate. The way the data is transferred between the SBC and the PPMs follows the standard VME protocol [VME87]. Figure 6.3 describes schematically the communication between the two modules on the VMEbus. It should be noted that the figure does not attempt to describe the whole set of VME sub-buses and handshake signals which the modules access during their communication, but only those that are relevant for the operation of the ReM_FPGA. The SBC, being the crate CPU unit, plays the role of the *master*. It takes control of the VME data bus, and initiates read/write bus cycles to transfer data to and from one of the *slave* PPMs installed in the same crate. The *slave* protocol is implemented on the PPM in a Complex Programmable Logic Device called VME_CPLD. The non-volatile characteristic of the device ensures that the VME interface is enabled as soon as the system starts up. But, due to the reduced internal resources generally featured by CPLDs, no complex processing of the VME requests can be performed on the device. Instead, the attributions of the VME_CPLD are mainly limited to decoding and distributing the VME requests to other two devices on the board: the ReM_FPGA and the FlashLoad_CPLD[2].

The VME addressing lines encode information about the device that has to take over and process the incoming request. The VME model implemented in the PreProcessor system uses the A32 addressing mode. Each PPM occupies 8 MByte of address space, by using the first 23 least significant bits of the address bus (A[22:0]) (see figure 6.4). The upper two bits of

---

[2]the device administrates the communication with a local non-volatile Flash RAM memory. Firmware binaries for the ReM_FPGA, and the four FPGAs of the LVDS Cable Driver (LCD) daughtercard, can be uploaded into the Flash RAM memory via the VME interface, to later facilitate a fast loading to the related devices. More details about the functionality of the FlashLoad_CPLD and the Flash RAM memory are presented here [Han09b].

this range (A22, A21) are used to further subdivide the address space. The first 2 MByte are allocated for access to status and control registers of the VME_CPLD and the FlashLoad_CPLD devices, while the other 6 MByte are allocated for access to all registers and memory locations interfaced by or available in the ReM_FPGA. The other bits of the A32 address bus are used for crate and module identification. The four most significant bits (A[31:28]) define the Crate ID, and they are set to a fixed value[3], i.e. 0xC (in hexadecimal). The next five most significant bits (A[27:23]) encode the *geographical address*, i.e. they encode the position of the module which is being accessed in one of the 21 slots of the crate. The VME_CPLD compares these five bits against a hardwired *slot address*, available on the backplane of the crate, to determine whether the incoming VME request is addressed to the PPM on which it operates.

The 32-bit VME address bus is only routed to the VME_CPLD. The device distributes a sub-set of addressing lines to the ReM_FPGA or the FlashLoad_CPLD, according to the values of the bits A22 and A21. Although the PPM-VME addressing mode allows byte-wise access on the bus, all the ReM_FPGA locations in the address space are 4-byte aligned. This means that the two least significant bits of the address space (A[1:0]) are never used for addressing the ReM_FPGA. As a consequence, only 21 bits (A[22:2]) are routed from the VME_CPLD to the ReM_FPGA. The input 32-bit data bus (D32) is fanned out on the board and routed to all three devices. The handshake protocol between the VME_CPLD and the other two *slaves* ensures that only one device has *write access* to the data bus at a time, in order to preserve the integrity of the data on the VMEbus.

The model of communication between the VME_CPLD and the ReM_FPGA is similar to the *master-slave* communication realised between the SBC and the PPM on the VMEbus. Upon identifying a VME request for the ReM_FPGA, the VME_CPLD distributes the related address data and a set of control signals to the ReM_FPGA, while holding the VMEbus cycle active. These control signals are copies of the main handshake signals involved in the SBC-PPM communication, and therefore they permanently reflect the state of the VMEbus cycle: VmeAddressSelect (AS*[4]), VmeDataSelect (DS*) and VmeWrite (WRITE*) (see again figure 6.3). The ReM_FPGA responds with an acknowledge signal, i.e. VmeReady (DTACK*), after it has latched the input VME address and data, and it has finished processing the VME request. The acknowledge signal determines the VME_CPLD to end the current VMEbus cycle. Additionally, the ReM_FPGA can generate an interrupt signal, and provide it to the crate controller via the VME_CPLD and the VME Priority Interrupt Bus. The hardware level can be programmed, on a scale from 1 to 7 (IRQ1-IRQ7), via a configuration register in the VME_CPLD. Also, the 8-bit interrupt vector table can be configured via the same register [Sch09]. However, currently neither the design of the ReM_FPGA's firmware nor the ATLAS Online Software consider this aspect.

---

[3]the ATLAS Online Software and the test/debug applications are installed and running on the SBC. Therefore a separate ID for each PPr crate is not needed. Also, the Run Controller, the component of the Online Software which coordinates the data-taking activities in the experiment, is connected to each PPr crate via high-speed Ethernet, and gets told explicitly which host to access. In contrast, an indivudual crate ID is needed to access the Fujitsu microcontroller and the TTCdec via the Timing Control Module (TCM). This ID is hardwired by a dedicated module, which is mounted on the back side of each PPr crate, in one of the *free* backplane slots.

[4]the asterisk behind the signal name indicates a low-active signal. This notation is used throughout the current and the next chapters.
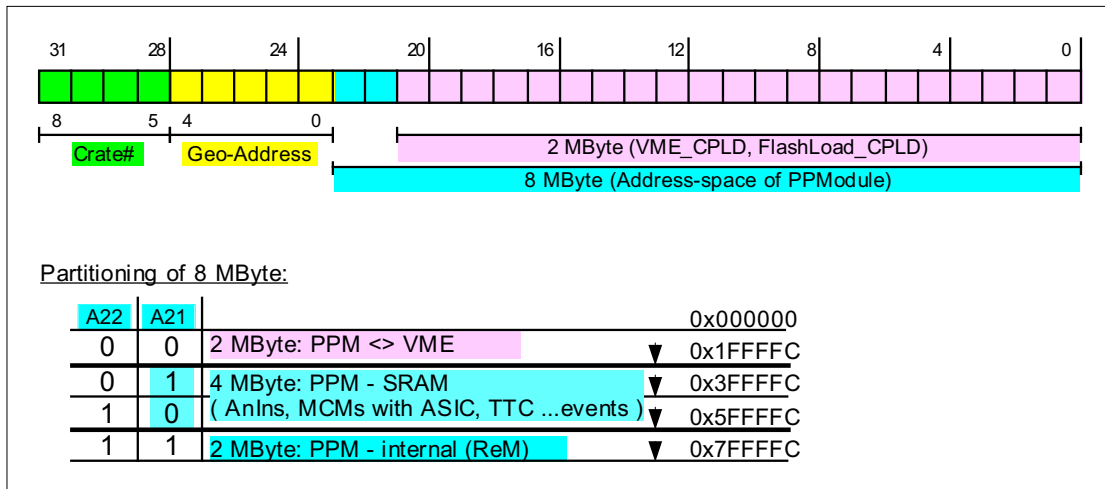
Figure 6.4: The address space of the PreProcessor Module [Han09b].

The actual processing of the VME request in the ReM_FPGA consists of transferring data between the VMEbus and the register or memory locations indicated on the address bus, in the format requested by the *consumer*. In addition, the ReM_FPGA has to ensure that the processing is realised within the maximum allowed period of time for the VME cycle. The VMEbus system limits the duration of the cycle[5] to avoid the communication to hang, when either the *master* has issued an address that the *slave* does not recognise, or when the *slave* takes too long time to resolve the request of the *master*. If the time out condition is met, then the VMEbus system generates a Bus Error (BERR) signal, which in ATLAS operation indicates a failure of the system, and may abort the operation of the supervising software. Certain VME requests, like the transfer of configuration data to the PPrAnIn-DACs or PPrPHOS4s, or the readback of register data from the PPrASICs or TTCrx, cannot be completed within one VME cycle, due to the timing characteristics of the data buses accessed by the ReM_FPGA during the transfer. In such cases, the ReM_FPGA closes the VME cycle as soon as it has acknowledged the request, and flags in dedicated VME status registers the progress of the operation and the availability of the requested data. Last but not least, the ReM_FPGA does not provide a fail-safe mechanism for those cases in which the software tries to access an unallocated address. Therefore, it is the responsibility of the software to issue a valid address. The VME address space of the ReM_FPGA is presented in appendix B, section B.1.

6.2.2 Access to the On-board SRAM

A synchronous static RAM[6] device is implemented on the PPM to extend the internal memory resources of the ReM_FPGA. The SRAM is mainly used to facilitate the transfer of data to VME. Readback of configuration data from PPrASICs or TTCrx or monitoring data from PPrASICs, cannot be collected and delivered to VME in one transfer cycle, due to their size and

---

[5]this parameter is configurable, and it is typically set to 16 or 256 μs [Joo05]

[6]K7M323625M from Samsung Electronics [Sam03]

time of acquisition. The ReM_FPGA assembles these data into dedicated blocks in the SRAM, from where the software can access it through the VME. Other usage of the memory refers to storing configuration data loaded from VME. There are two distinct cases. In the first one, the ReM_FPGA places a copy of each setting loaded from VME, which is designated to other on-board components than the ReM_FPGA, to serve as *reference*. In the second case, the software loads directly several distinct configuration patterns for each PPrASIC Playback Memory in dedicated locations in the SRAM, and then instructs the ReM_FPGA to pick up a certain pattern and transfer it to the PPrASICs (see section 6.4). A detailed description of the way the data is stored in the SRAM is given in section B.2.

The SRAM has a 20-bit address range, which is entirely mapped to the VME address space. Each SRAM memory location is 36 bits wide, and although the ReM_FPGA has access to the full width, only the lower 32 bits are entirely *visible* to VME. The other four bits are mostly unused and not accessible from VME. The exception occurs for those memory blocks which store readback of configuration data. The corresponding upper four bits of these locations are used by the ReM_FPGA to indicate the validity of the stored data. The mechanism that sets these flags, and the way the bits are made visible to VME, is presented in section 6.4.

6.2.3 The Serial Interfaces to the PPrASICs

As previously mentioned in section 6.1, the ReM_FPGA is connected to the 16 PPrASICs to transfer trigger configuration data and custom commands, and to receive event data, read-back of configuration and trigger independent monitoring data, and status information. For these purposes, each PPrASIC provides two custom, bi-directional and synchronous serial interfaces, each of which serves two PPrASIC channels[7] (see also figure 5.7). This means that the ReM_FPGA is connected to the 16 PPrASICs via a total of 32 serial data buses.

Figure 6.5 describes the control and data lines of the serial communication, and the mechanism of transferring and receiving data, as implemented on both devices. The first thing that should be mentioned is that the width of the data words transmitted over the serial bus is 13 bits. The boundaries of the each data word are defined by a FrameBit. The signal is used at both ends to transfer new data and to latch the incoming one. Upon the occurrence of the FrameBit, the transmitting end copies the data into an OutputShiftRegister, and on the next event of Serial-Clk it starts sending the data, bit-by-bit (see *DataOut*). The receiving end captures the data bits (*DataIn*) into an InputShiftRegister until the next occurrence of the FrameBit, when the whole content is copied into an InputRegister. The latter operation is necessary to further allow the latching of the incoming data bits without overwriting the previous data word. The protocol of the PPrASIC serial interface requires the external device to *master* the communication. Which means that the ReM_FPGA has to provide the FrameBit and the SerialClk signals to the PPrA-SIC. Also, the same protocol does not include any hand-shake mechanism. The success of a

---

[7]the reason for this implementation is rather historical. The present four-input channel PPrASIC was initially designed to process only two trigger signals, and to provide only one serial port for configuration and readout [IHE99]. Also, the PPrMCM design was different, being adapted to host two such PPrASICs. Later studies showed that it was possible to integrate four channels into one chip, and thus reduce the complexity of the PPrMCM layout. The design of the current PPrASIC was in principle obtained by duplicating the channel- and serial interface-wise functionality of the previous design, while adapting the existing global features and adding new ones.
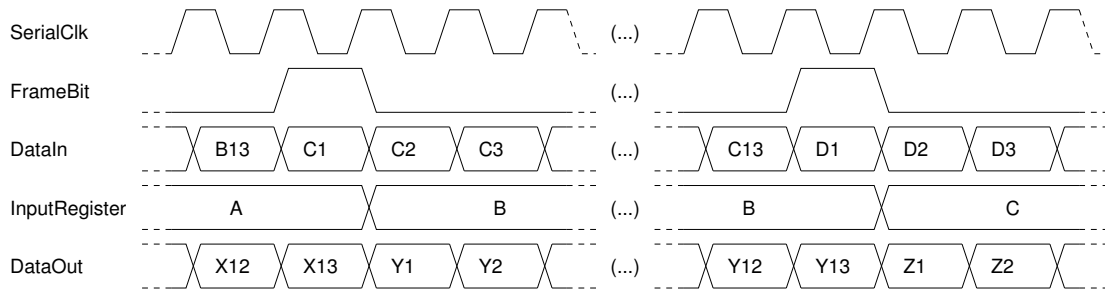
Figure 6.5: Timing waveform for the signals of the PPrASIC serial interface [Hus02].

write operation to the PPrASIC is reflected in the data provided by the device.

More details about the format in which the PPrASIC and the ReM_FPGA transfer data to each other over the serial interface, as well as details about the content of the respective data, are presented in various sections throughout the chapter.

6.2.4 The Interface to DAQ System

The top priority of the ReM_FPGA, during normal operation in the ATLAS experiment, is to provide event data to the DAQ System. The event data is accumulated channel-wise in pipeline memories, in the 16 PPrASICs, and it is transmitted to the ReM_FPGA upon receiving a positive decision from the trigger system. The ReM_FPGA gathers the event data from all PPrASICs, processes it in the specified ATLAS format, and sends it as serial streams to the DAQ System, via a ROD module.

The transfer of event data from the ReM_FPGA to the ROD is intermediated by the *slave* RGTM-O card [Mah05], which is mounted on the back side of the PPr crate. Figure 6.6 describes schematically the relationship between the ReM_FPGA and the RGTM-O, as well as the readout path to DAQ. The ReM_FPGA is connected to the RGTM-O via a synchronous and uni-directional 20-bit data bus. Only 16 lines of the bus are actively used, each of them transporting the processed event data of one PPrASIC. On the RGTM-O, the 16 input readout streams are first buffered and further serialised into one stream by a Gigabit Rate Serial Transmit Chip [Agi], and then sent via G-Link to the ROD by an optical transmitter [Inf]. Additionally, the ReM_FPGA provides the RGTM-O with a 40.00 MHz clock (GLinkClk), obtained from the PPM on-board crystal oscillator, and three control signals: Data Available (DAV*), RESET* and LaserDis-abled (Laser_Dis). The DAV* frames the serial event data streams on both the ReM_FPGA and the RGTM-O's transmitter chip. The RESET* initialises the internal registers of the same device[8], while Laser_Dis signal is routed to the RGTM-O's optical transmitter, to switch the device on[9].

---

[8] the documentation of the Gigabit transmitter chip specifies that the external device must keep the reset signal active for at least five clock periods, in order to ensure the success of the operation. The ReM_FPGA keeps the signal active for eight clock periods.

[9] the optical transmitter requires the Laser_Dis signal to be low-active and kept in this state for at least $10\,\mu$s. The ReM_FPGA drives the signal high, as its state is inverted by logic on the RGTM-O, prior to the arrival in the optical transmitter (see figure 6.6 and the related documentation [Per05]). Also, the ReM_FPGA drives the signal
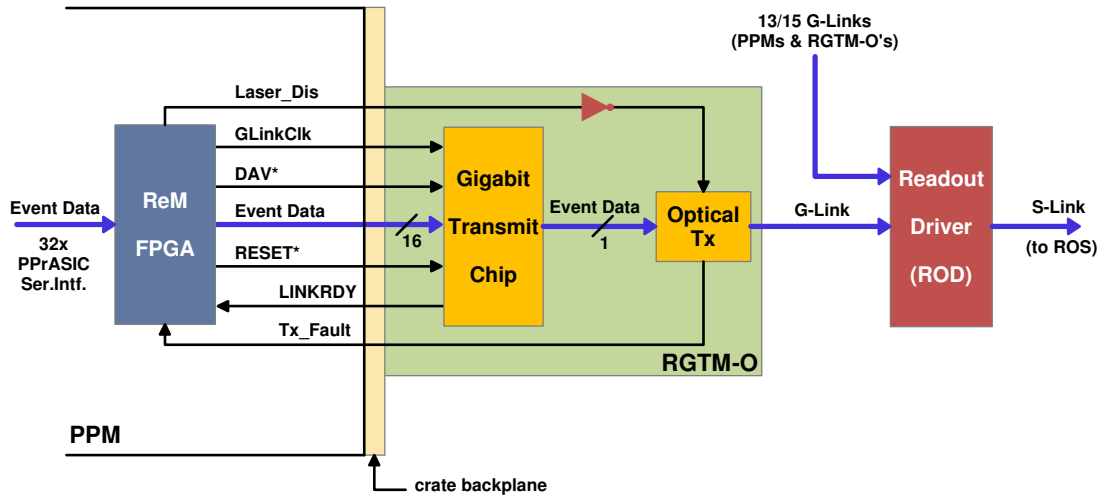
Figure 6.6: The interface to the DAQ system.

Two status signals are routed from the RGTM-O to the ReM_FPGA: LINKRDY and Tx_Fault. The LINKRDY signal indicates that the RGTM-O's transmitter chip is ready to send data to the ROD, while Tx_Fault indicates an eventual faulty operation of the laser. The strategy implemented in the ReM_FPGA's firmware, for transmitting the event readout data to the DAQ System, does not consider the gating of the data flow with these two status signals. A recovery mechanism would be very complex and unnecessary. The eventual faulty operation of the RGTM-O is recognised by the ROD [Bar08], and the replacement of the defective module is the most appropriate solution for such cases. Therefore, the two bits are only mapped by the ReM_FPGA in the first ReM_Status register, in order to provide a status of the system to VME (see table B.22). In addition, the state of the LINKRDY signal is reflected by one LED on the front panel of the PPM.

6.2.5   The SPI Interface to the PPrAnIn-DACs

As mentioned in the previous chapter, the conditioning of the analogue calorimeter trigger signals is performed by four PPrAnIn boards. Each PPrAnIn receives and prepares for digitisation 16 analogue trigger-tower signals. The signals, which arrive as differential pairs, are first converted to single-ended and then rescaled to match the digitisation window of the PPrMCM's FADCs. Additionally, the baseline of the single-ended signal is adjusted, by adding a DC-level offset, and a comparator indicates when the same single-ended signal has exceeded a given threshold. The offset and the threshold are configurable parameters, and they are provided by 8-bit programmable DACs.

Each PPrAnIn is equipped with four identical DACs[10], each of which has eight programmable channels. Two of these DACs are used for adjusting the baseline of the signals,

---

permanently high, as there is no operational need to deactivate the optical transmitter.

[10]MAX529CAG from MAXIM [Max94]
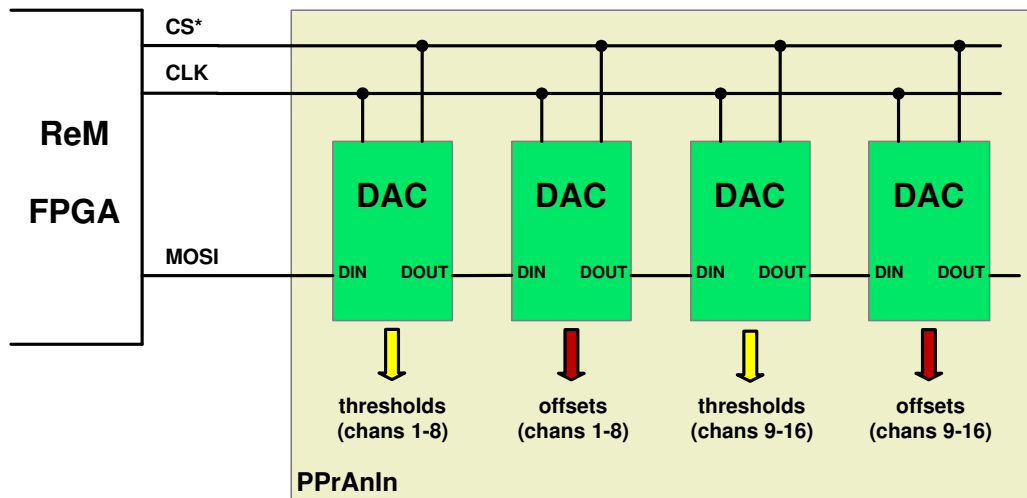
Figure 6.7: The implementation of the SPI bus on the PPM for the case of one PPrAnIn board.

while the other two are used for setting up the threshold at the input of the comparator. The ReM_FPGA is connected to all four DACs via a synchronous Serial Peripheral Interface (SPI) Bus. There are four physical SPI buses implemented on the board, each one providing access to a corresponding PPrAnIn board. Figure 6.7 illustrates the implementation of the SPI bus in the case of one PPrAnIn board. The ReM_FPGA transfers the data to the DACs over the *Master Out, Slave In* (MOSI) line. The four DACs are daisy-chained, such that only the first DAC is directly connected to the MOSI line, while the other three are connected to the DOUT pin of the preceding DAC. A particularity of the SPI bus implemented on the PPM is that it does not contain the *Master In, Slave Out* (MISO) electrical line. Which means that the DACs cannot be read back[11].

The data transferred by the ReM_FPGA to each DAC consists of an 8-bit address, identifying one of the eight internal DAC channels, and an 8-bit offset or threshold data. Which means that during one write cycle to all four DACs the ReM_FPGA transmits 64 bits. On the ReM_FPGA, the output data is clocked with a 2 MHz clock, which is internally derived by dividing the LHC clock. The same clock pulse is provided to all DACs via the CLK line, to allow the latching of the incoming data on each device. In parallel, the ReM_FPGA asserts the *Chip Select* (CS*) signal to control the operation of the DACs. When the signal is driven low, each DAC shifts the input data into a 16-bit register, while sending the least significant bit of the same internal register to the DOUT pin. This operation ensures the propagation of data through the daisy chain. Simultaneously with the arrival of the 64th data bit in the first DAC, the ReM_FPGA drives the CS* signals high. Upon the detection of a low-to-high transition of the CS* signal, all four DACs disable the CLK input, and update their DC outputs according with the digital data stored in the internal shift register [Sch09].

---

[11]the readback would have been possible if the DOUT pin of the last DAC was connected to the ReM_FPGA, via the MISO line. The reason that had motivated the decision, to leave out the MISO line, is that only the constant DC output of the DACs plays a significant role, and this can be deduced by analysing the FADC data [Han09a].

6.2.6   The I^{2}C Interfaces to the PPrPHOS4s and the TTCrx

Two standard serial Inter-Integrated Circuit (I^{2}C) buses are implemented on the PPM. One of the I^{2}C buses connects the ReM_FPGA with all 16 PPrPHOS4 timing chips [CER92], to allow the configuration of the digitisation strobes, while the other I^{2}C bus connects the same FPGA device with the TTCrx chip [Chr04] of the TTCdec card [Qia05], for configuration and readback of the TTCrx registers.

In both cases the ReM_FPGA is the *master*, and the other devices are the *slaves*. Figure 6.8 illustrates the implementation of the two I^{2}C buses on the PPM, and the logic in the ReM_FPGA that masters the data transfer over these buses. The ReM_FPGA provides each slave with 8-bit data via the *Serial Data* (SDL) line, and with a 100 kHz clock pulse, derived from the LHC clock, via the *Serial Clock* (SCL) line. The ReM_FPGA sends four distinct types of data, each of them being loaded on the bus at certain steps during the communication, as defined by the I^{2}C protocol: a *slave address*, which identifies the device to which the data is addressed, a *local address*, which indicates the location on the slave that has to be read or written, the actual configuration data, and the control bits that drive the I^{2}C communication. The *slave* functionality of the TTCrx and the PPrPHOS4 is different. The TTCrx provides an acknowledge bit in response to each byte received from the ReM_FPGA, and, if readback is requested from one of its registers, it provides the corresponding 8-bit register data. The PPrPHOS4 responds as well with an acknowledge bit, but it does not provide a read-access to its registers.

The properties of the standard I^{2}C bus require all the devices connected to the SCL and SDA lines to have open-drain or open-collector outputs. The ReM_FPGA fulfils the requirement only for the SDA lines, while the SCL lines are driven via a push-pull output. The reasons that led to this implementation are mainly related to the characteristics of the PPrPHOS4-I^{2}C bus, and with the operation of the PPrPHOS4 chip. Early work, with a prototype PPM, revealed that the PPrPHOS4 cannot acknowledge the clock properly if the SCL line is pulled up [Mah04]. The relatively long lines of the PPrPHOS4-I^{2}C bus, determine a high bus capacitance and, subsequently, a slow rising edge of the clock. Additional reflections or noise on the clock line, combined with the absence of a Schmitt trigger at the input stage of the PPrPHOS4, determine the device to double-latch the input clock [Sch09]. By driving the SCL line through a push-pull output, a faster rising edge of the clock is obtained, and the described effect is minimised. The I^{2}C bus that connects the ReM_FPGA with the TTCrx chip is considerably shorter, and it does not present similar problems. However, for simplification of the firmware design, the corresponding SCL line is also driven through a push-pull output.

The I^{2}C protocol is maintained in the ReM_FPGA by two modules: *I2CDataTransfer* and *I2CMasterCore*. Although the latter module bears the name *master*, the actual master of the I^{2}C communication is the I2CDataTransfer module. The I2CMasterCore, which is entirely based on source code, provides only the interface to the I^{2}C bus [Her03]. This in principle means, that the module has to be instantiated twice in the firmware design, such that each instance interfaces one I^{2}C bus. But, because the TTCrx and the PPrPHOS4s are never addressed simultaneously over the VME, and in order to minimise the usage of internal FPGA resources, the following strategy has been adopted. The I2CMasterCore is implemented only once, so that it *sees* only one I^{2}C bus with 17 slaves, i.e. 16 PHOS4s and one TTCrx. The I^{2}C related signals produced by this module, i.e. output data (SDOUT) and clock and data output enable (COE*, DOE*), are fanned
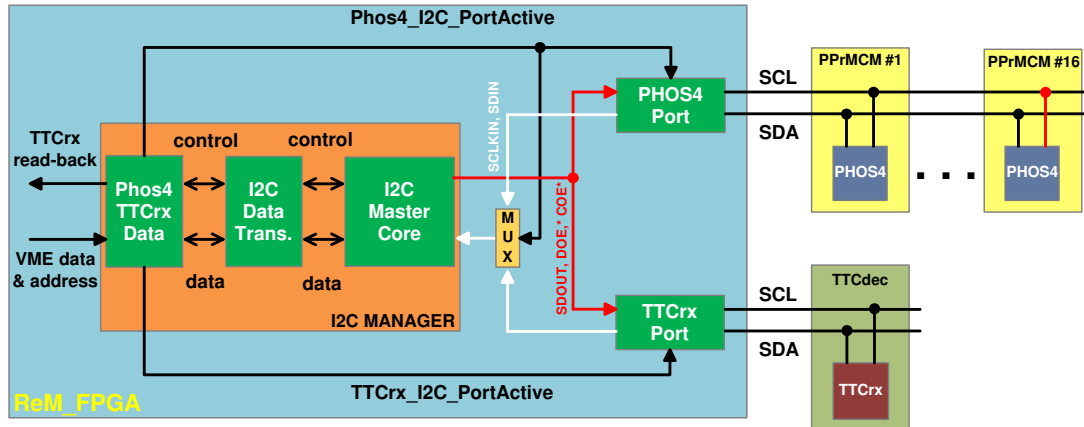
Figure 6.8: The implementation of the I^{2}C buses on the PPM.

out and routed to two functionally identical modules, *PHOS4Port* and *TTCrxPort*, where all I^{2}C
I/Os are implemented. The SDOUT and DOE* signals fed the open-collector logic, while the
COE* signal is used by the push-pull logic. The I2CMasterCore provides also an output clock
signal (CLKOUT), but since the SCL line is driven through a push-pull output, this signal is not
used at all and, for the same reason, not shown in figure 6.8. The transfer of data over each I^{2}C
bus is gated by two *output enable* signals, Phos4_I^{2}C_PortActive and TTCrx_I^{2}C_PortActive, to
ensure that I2CMasterCore accesses only one bus at a time. The two enable signals are asserted
by a *Phos4TTCrxData* module. Based on the input VME address, this module makes a clear
distinction between a PPrPHOS4 and a TTCrx *request*, and asserts the corresponding enable
signal, while simultaneously deactivates the other one. Additionally, the same functional module
provides the I2CDataTransfer with appropriate slave and local addresses, and with configuration
data from VME.

The input I^{2}C clock and data signals are also routed to the I2CMasterCore. Each of the *Port*
modules provides two such signals from the corresponding I^{2}C bus, which are then multiplexed
according to the logic value of the Phos4_I^{2}C_PortActive enable signal. When the input I^{2}C
data provides configuration data read back from the TTCrx, the content is transferred from the
I2CMasterCore to the SRAM, via the I2CDataTransfer and Phos4TTCrxData modules.

Last but not least, a special behaviour of the TTCrx chip was observed when the LHC clock
is not present. As long as the latter occurs, an internal watchdog circuit initiates periodically
an automatic reset of the chip, which affects as well the I^{2}C interface [Han09a]. Therefore, in
order to avoid the communication to hang when the TTCrx is in reset state, the ReM_FPGA
does not gate the data transfer with the acknowledge bit sent by TTCrx. However, in order
to allow the verification of the I^{2}C transfer in normal operation mode, the ReM_FPGA maps
the acknowledge bits, received from the TTCrx at different steps of the communication, in the
second second ReM_Status register (see table B.23). In a similar way, the ReM_FPGA ignores
the acknowledge bits sent by the PPrPHOS4s during the I^{2}C data transfer, and maps these bits
into a dedicated VME status register called *PHOS4_Acknowledge* (see section B.3.5).

6.2.7   Control and Status Signals

The ReM_FPGA provides and receives multiple control and status signals from the on-board components. These signals are transmitted over single lines, independent of any bus protocol. The information they carry and their handling in the ReM_FPGA is described in the following subsections.

The Level-1 Protocol Signals

These include the L1A signal and the synchronous bunch counter (BCR) and event counter reset (ECR) signals. All three signals are generated by the CTP[12]. The L1A signal indicates that the CTP has accepted an event, while the BCR and ECR signals synchronise corresponding local counters on the ATLAS front-end and readout electronics. The CTP fans the signals out to the TTC system, which distributes them via optical fibers to the detector electronics. In each PPr crate, a *Timing Control Module* (TCM), located in the right most slot, receives the TTC signal, converts it to electrical form, and distributes it to each PPM over the crate backplane [Gee06]. On the PPM, the signal is directly routed to the TTCdec card, where the TTCrx chip extracts the L1A, BCR and ECR signals from the stream, re-synchronises them with the LHC clock, and routes them on separate channels to the ReM_FPGA.

In the ReM_FPGA, each input is fanned out twice. One copy is distributed to all 16 PPrA-SICs[13], while the other copy is internally used by the logic that processes and transfers the PPrASIC readout data to DAQ (see section 6.5). Additionally, the protocol signals are registered twice in the ReM_FPGA, one time at the input stage and another time at the output stage, as shown in figure 6.9. This implementation was adopted after it was observed that, if the copies provided to the PPrASICs are routed asynchronously through the FPGA, their alignment with the clock supplied to the same devices (MCMClk) significantly differs from the one set by the TTC. Both the MCMClk and the system clock (SysClk), which operates the registering of the signals, represent delay-compensated version of the LHC clock (see also section 6.3). The consequence of the double registering is that the L1A, BCR, and ECR signals will arrive in the PPrASICs with a delay of 50 ns. To cope with the situation, the controlling software must adjust accordingly the read pointers of the scrolling memories which record event data in the PPrASIC.

The Local Protocol Signals

For the case when the PPM is operated in standalone mode, on a set-up that does not include a TTC system, the ReM_FPGA can generate on request local protocol signals, i.e. local trigger and counter reset signals (see LL1A, LBCR and LECR in figure 6.9). The request is addressed over the VME, via three dedicated registers. The bit field content of these registers, as well as

---

[12]the BCR signal is, in fact, a reflection of the ORBIT signal provided by the LHC machine, and which defines the 3564 bunch-crossings LHC turn [Pau05].

[13]in order to reduce the usage of I/O pins on the ReM_FPGA, and to reduce the complexity of the PPM layout, the duplication of each protocol signal into 16 other signals is not performed in the ReM_FPGA, but on dedicated buffers located on the main board. This implementation extends to all control and clock signals distributed from the ReM_FPGA to the PPrASICs.
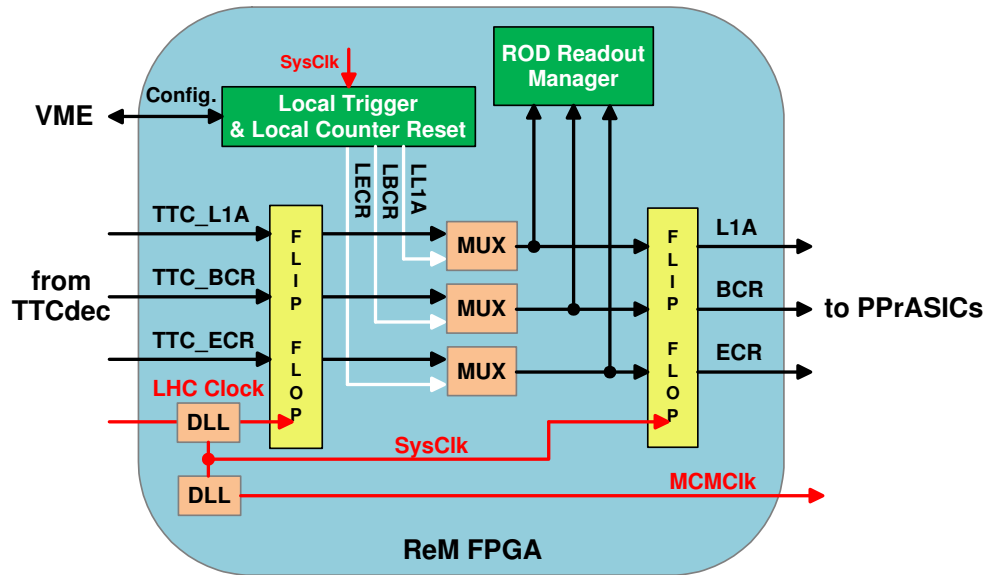
**Figure 6.9:** The handling of the Level-1 and the local protocol signals in the ReM_FPGA.

the various modalities provided to generate a local trigger, is described in details in appendix B (see sections B.3.18 and B.3.19).

Once generated, each signal is multiplexed with the corresponding TTC protocol signal, so that it gets distributed to the same locations. In order to prevent any accidental generation of local trigger and counter reset signals, while the PPM is used in the ATLAS data-taking runs, one can configure the ReM_FPGA to operate in a so called *DAQ_Mode*. As long as this mode is enabled, the ReM_FPGA will deny any VME request for local protocol signals and correspondingly flag the refusal via the first ReM_Error register (see table B.26). The DAQ_Mode is described in section 6.4.6. Also, applications that make use of the local protocol signals are described in the next chapter.

Clock Selection on the TTCdec Card

The *real-time* pre-processing performed on the PPM is driven by the LHC clock. The signal is delivered by the TTC system, via the same optical streams that contains the protocol signals. The TTCrx recovers the clock signal from the input stream and provides one non-deskewed and two deskewed copies to the output. As described in section 6.3, one of deskewed LHC clocks is routed to the ReM_FPGA which then delivers it to all the on-board devices located on the *real-time* path.

When the PPM is operated on a set-up that does not include a TTC system, and hence is not supplied with an LHC clock, the TTCdec daughtercard can provide the ReM_FPGA, in compensation, with a 40.00 MHz clock pulse from a local crystal oscillator (XTAL). On the TTCdec, the XTAL clock and the two deskewed LHC clocks from TTCrx (Clock40Des1, Clock40Des2) are fed into a multiplexing logic, of which output is controlled by two signals
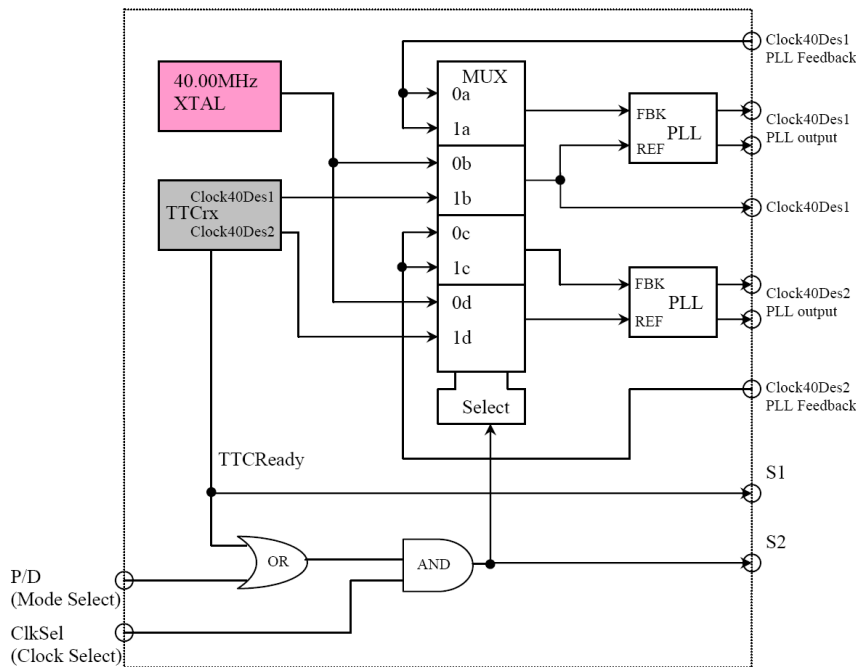
Figure 6.10: The clock selection scheme on the TTCdec [Qia05].

provided by the ReM_FPGA (see figure 6.10).

The first signal, $P\backslash D$, defines two operational modes for the ReM_FPGA: *Protected* and *Debug*. In *Protected* mode, the ReM_FPGA has to select only the deskewed LHC clocks, and to disable the data sent on the external interfaces whenever the LHC clock is not available. This latter aspect is flagged by the TTCrx, via the *TTCReady* signal. When asserted, the signal indicates that the TTCrx receives external TTC signals and that the phase-locked loop (PLL) of its internal clock and data recovery circuit has locked. This signal is routed to the ReM_FPGA via the S1 output. In *Debug* mode, the ReM_FPGA can either select the XTAL clock as a source or automatically change the clock source [Qia05]. As the clock pulse supplied by this multiplexing logic drives also most of the ReM_FPGA's internal logic, the *Debug mode* is permanently enabled, i.e. $P\backslash D$ is always set to the logic value of "0", so that the clock source is automatically changed.

The second control signal provided by the ReM_FPGA, *ClkSel*, indicates actually which clock source has to be selected. If the signal is driven high then the TTCrx is selected, otherwise the XTAL. The ReM_FPGA asserts this signal according with the logic value of the TTCReady signal, so that the XTAL clock is automatically selected when the LHC clock is not available. The source selected by the multiplexing logic is reflected in the S2 output, which is also routed to the ReM_FPGA. Together with the S1 output, this signal is mapped in the first ReM_Status registers. Additionally, the state of the S1 and S2 signals is reflected by two LEDs on the front panel of the PPM.

Broadcast bits	Description
6'b00_0010	Stop LVDS sync pattern (PPM specific)
6'b00_0011	Start LVDS sync pattern (PPM specific)
6'b01_0000	Start playback (global command)

Table 6.1: The TTC broadcast commands for the PPr system.

The TTC Broadcast Commands

Certain functional or testing procedures require a synchronous operation of all PPMs in the system. In order to determine the lock condition of the LVDS receivers on the L1Calo processors, the PPrMCM-LVDS transmitters should simultaneously generate and transmit a synchronisation pattern. Also, test applications with PPrASIC playback data, which verify the timing and performance of the digital part of the L1Calo system, require a synchronous start of the playback memories. The only possibility to realise this global synchronisation is by distributing broadcast commands via the TTC system.

The broadcast commands encode specific actions for the system in an 8-bit data packet. The TTCrx chip recovers the data from the input serial stream and provides it in parallel to the ReM_FPGA. The two least significant bits of the broadcast data are reserved by the TTC system for bunch counter and event counter reset commands, and they are normally set to zero when broadcast commands are distributed to L1Calo [Lan08]. For this reason, the corresponding lines are not routed from the TTCdec card to the ReM_FPGA. The remaining six bits encode global and module specific commands. Currently, one global and two specific commands are addressed to PPM. Table 6.1 shows the combination of bits and the corresponding description for each command.

Additionally, the TTC provides two strobes, BrcstStr1 and BrcstStr2, to validate the broadcast data. The strobes are differently synchronised with the two deskewed LHC clocks on the TTCrx (Clock40Des1, Clock40Des2). The BrcstStr1 signal and the lower four command bits are synchronised to Clock40Des1, while BrcstStr2 and the upper two bits are synchronised either with Clock40Des1 or Clock40Des2. As the ReM_FPGA's firmware uses exclusively Clock40Des1 (see section 6.3), data is latched only on the occurrence of BrcstStr1. After the content of the broadcasted message is decoded, the ReM_FPGA distributes corresponding control signals to the PPrASICs or the PPrMCM-LVDS transmitters, via dedicated single lines.

The same actions listed in table 6.1 are available as VME commands, to provide a similar service when the PPM is tested in standalone mode (see section B.3.17). The related VME and TTC commands are OR'ed, so that if distributed simultaneously a common action is taken.

The PPrPHOS4 Status

The PPrPHOS4 chip provides digitisation strobes to the four FADCs mounted on the same PPrMCM. Each strobe is obtained by delaying the input LHC bunch-crossing clock in steps of 1 ns, up to 25 ns, with respect to a reference clock. Both the LHC clock and the reference

clock are supplied by the ReM_FPGA (see 6.3). On the PPrPHOS4 chip, the timing reference is provided to each channel via a Delay Locked Loop (DLL). In parallel, a phase detector continuously compares the input reference clock against the DLL output, and the result is routed off the chip. The PPrASIC mounted on the same PPrMCM monitors the phase detector output to determine the proper operation of the PPrPHOS4. If the frequency of the phase detector output signal is lower than 2 MHz or higher than 6 MHz, or the signal has a constant value, the PPrASIC asserts a *FrequencyLost* bit. This status bit is then routed from the PPrASIC to the ReM_FPGA, where it is packed with other 15 similar bits into a VME status register (see section B.3.6).

The External BCID Signals

As mentioned in section 6.2.5, comparators on the PPrAnIn board indicate when the rising edge of the single-ended signals has exceeded a programmable threshold. The digital output of each comparator, named *External BCID*, is sent to a corresponding PPrASIC in order to be used in the BCID logic. As each PPrAnIn board performs analogue processing for 16 input signals, 16 External BCID signals will result from the comparison. In parallel, a copy of these signals is logically OR'ed on the PPrAnIn board, and the result is sent to the ReM_FPGA. Therefore, the ReM_FPGA receives in total four such External BCIDs, one from each PPrAnIn board. The signals are used by ReM_FPGA to built a local trigger, for testing and debugging applications. The implementation of the local trigger is described in section B.3.18.

The VME_Reset Signal

This is a low-active signal which is generated by the controlling software, via a configuration register in the VME_CPLD, to initialise the logic of the ReM_FPGA. A copy of this signal is distributed by the ReM_FPGA to all 16 PPrASICs, for the same purpose.

The TTCrx_Reset Signal

This signal initialises the TTCrx and is generated by the ReM_FPGA, following a VME request addressed via the ReM_Control register (see also B.3.22). Initially, the signal was coupled to the VME_Reset, so that the TTCrx was initialised in the same time with the ReM_FPGA and the PPrASICs. The decision to implement a separate function for TTCrx was taken after it was observed that the reset of the TTCrx device sometimes generates a glitch in the clock provided by the TTCdec.

   In more words, when the reset state is activated, the TTCrx de-asserts the TTCReady signal, which, as previously explained, determines the clock multiplexing logic of TTCdec to select the XTAL as clock source. As soon as the reset state is released, and its internal PLL has locked, the TTCrx re-asserts the TTCReady signal, which determines the same multiplexing logic to consider the TTCrx as clock source. This switching between the two clock sources, from TTCrx to XTAL and then back to TTCrx, generates sometimes a clock glitch, which affects irreversibly the logic of the ReM_FPGA[14]. One example of logic damaged by a clock glitch is the generation

---

[14]the "clock glitch" problem was only very recently spotted. So far, it is not clear whether the glitch is generated by the clock multiplexing logic of the TTCdec or by the ReM_FPGA's DLLs (see section 6.3). The only clear evidence

of the FrameBit signal, which controls the serial communication with the PPrASICs (see again section 6.2.3). The signal represents the least significant bit of a 13-bit shift register. Within this register, one bit of logic value "1" is shifted on the occurrence of each clock event, so that a FrameBit signal is generated every 13th clock cycle. The observed behaviour was that, sometimes, after a reset to TTCrx, the FrameBit does not get asserted again. This has led to the conclusion that the clock glitch determines the logic of the shift register to latch the shift bit at a wrong time, and thus to change its logic value from a "1" to a "0". The main consequence of losing the FrameBit is the inability of the ReM_FPGA and the PPrASICs to transfer data to each other. Additionally, the communication with VME is affected. When the ReM_FPGA is requested to transfer data from VME to PPrASICs, the ReM_FPGA asserts the acknowledge bit (i.e. VmeReady, see again figure 6.3) only after the data transfer is completed. As the data transfer cannot occur in the absence of the FrameBit, the acknowledge bit is never asserted, and this determines the crate controller to generate a VME bus error signal. Last but not least, since the TTCdec clock is distributed by the ReM_FPGA to the PPrASICs, one can expect the PPrASIC logic to be as well affected by the clock glitch.

The only way to restore the correct operational state of the ReM_FPGA and of the PPrASICs is by re-initialising their logic via a VME_Reset. As this signal was initially also distributed to the TTCrx, it was decided to implement two separate reset functions, and request the controlling software to always send a VME_Reset after an initialisation of the TTCrx chip. This implementation does not prevent a further clock glitch to occur, but it allows the logic on the ReM_FPGA and the PPrASICs to recover from an eventual clock glitch.

Finally, the ReM_FPGA permanently monitors the locking state of the TTCrx via the S1 status bit of TTCdec. A *TTCrx_LockLost* bit is asserted in the _Status register each time the S1 signal makes a high-to-low transition. The status bit is cleared by a VME_Reset, so that the high-to-low transition induced by a reset to TTCrx is not considered. Additionally, the logic that sets the TTCrx_LockLost bit is driven by the PPM_XTAL clock, to ensure that it does not get damaged by a clock glitch. Thus, whenever this status bit is asserted, one can count on a malfunction of the system, eventually caused by a clock glitch.

Status Signals to the Front Panel of the PPM

Two status signals are generated on the ReM_FPGA in order to activate two LED indicators on the front panel of the PPM board. The first signal indicates that an L1A was either received from the TTC or locally generated in the ReM_FPGA. The second signal indicates that at least one of the PPrPHOS4 status bits, i.e. the FrequencyLost bits delivered by the PPrASICs, is set. Both signals are stretched to about 400 μs wide, so that the active state of the LED indicators can be sensed by the human eye.

---

is that the DLLs do not deactivate their LOCKED signals when the clock glitch occurs. This was proven by counting the high-to-low transition of each LOCKED signal, i.e. the lock is lost. Whenever a clock glitch has occured, the value of the respective counters was set to zero.

6.3   Clock Management

Another task of the ReM_FPGA is to supply various on-board and external devices with clock pulses (see figure 6.11). The ReM_FPGA, in his turn, is supplied with clock pulses from two sources: the TTCdec and the PPM_XTAL. As described in the previous section, the TTCdec can provide either two deskewed versions of the 40.08 MHz LHC clock (Clock40Des1 and Clock40Des2) or a 40.00 MHz pulse train from a local XTAL. These pulses are routed to the PPM board via three channels. The first two channels provide a delay-compensated version of the Clock40Des1/XTAL and Clock40Des2/XTAL clock pulses, while the third channel provides the original Clock40Des1/XTAL clock pulse output of the multiplexing logic (see again figure 6.10). The ReM_FPGA uses and distributes to other devices only the clock pulse supplied on the third channel. The reason for this choice is that the PPrMCM-LVDS transmitters, which are supplied with a clock pulse by the ReM_FPGA, require a clock jitter value smaller than 150 ps [Nat02]. This specification is fulfilled only by the non-compensated clock pulse [Sch09]. The other two clock signals are also routed to the ReM_FPGA, but the corresponding input pins are disabled in the firmware design.

The second clock source, the PPM_XTAL, provides the ReM_FPGA with a 40.00 MHz pulse train [Jau]. This clock signal, and the one received from TTCdec, are routed in the ReM_FPGA to the input of a clock management scheme. The respective logic, located in the *ClockManager* functional block, employs internal FPGA DLLs to obtain delay-compensated and equally symmetric[15] versions of the input clock pulses, and delivers the resulting signals to internal and external *consumers*. The clock scheme is illustrated in figure 6.11. Additionally, table 6.2 lists the produced clock pulses and the devices to which they are delivered.

The input PPM_XTAL clock is only used to produce the serial frame clocks that drive the transfer of event data from the ReM_FPGA to the RGTM-O, i.e. intGLinkClk and extGLinkClk. The choice for the PPM_XTAL was triggered by yet another clock-jitter issue. In an early version of the ReM_FPGA's firmware design, intGLinkClk and extGLinkClk were obtained from the deskewed LHC clock. During functional tests of the DAQ interface, performed with this firmware version, it was observed that bits were constantly lost during the transmission to the ROD. Although the documentation does not provide any jitter specification, it was assumed that a large jitter value, above the tolerance, on the input clock provided by the ReM_FPGA, determines the RGTM-O's Gigabit transmitter chip to wrongly latch the input data. The assumption was later proven to be correct, when the firmware design of the ReM_FPGA was changed to generate the serial frame clocks from the input PPM_XTAL clock [Sch09]. Also note that, in order to ensure a reliable latching operation on the Gigabit transmitter chip, the intGLinClk and extGLinkClk clock signals are 180-degree phase-shifted.

All the other clock pulses are obtained from the input supplied by the TTCdec. This includes also the I^2C- and SPI-bus clocks, which are derived by dividing the system clock (SysClk) down to the needed frequency. The division is realised in logic, in other functional blocks of the ReM_FPGA, and not via DLLs.

An important functional aspect of the clock management scheme is that all the DLL feedback loops are realised inside the ReM_FPGA. In the case of the DLL that provides the system

---

[15]i.e. 50%-50% duty cycle.
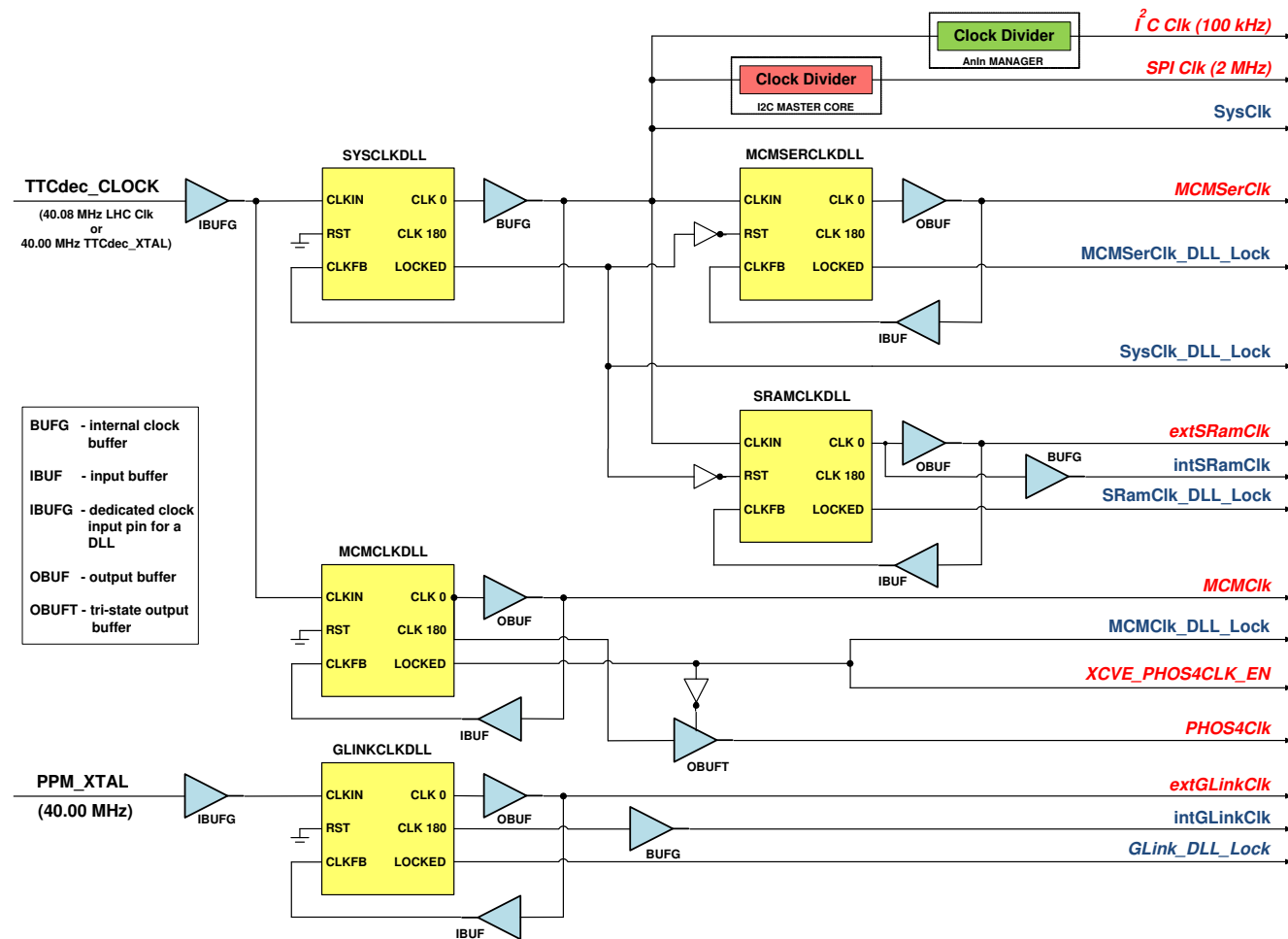
Figure 6.11: The clock management and distribution scheme on the ReM_FPGA. Note that the representation of the DLLs is simplified, to describe only the pins which are used in the firmware design. Also, the output clock signals of whose name is given in italics are routed off the FPGA and provided to external components.

Clock Name	Consumer(s)	Usage
SysClk	ReM_FPGA	system clock, drives most of the logic
extSRamClk	SRAM	operational clock
intSRamClk	ReM_FPGA	drives the SRAM related logic
MCMClk	PPrASICs	drives the pre-processing logic
	PHOS4s	used for generating FADC strobes
	PPrMCM-LVDS	operational clock
	Transmitters	
MCMSerClk	PPrASICs	drives the serial interface logic
Phos4Clk	PHOS4s	reference clock for delaying the FADC strobes
I^2C_Clock	PHOS4s	I^2C-bus clock
	TTCrx	”
	ReM_FPGA	”
SPI_Clock	PPrAnIn-DACs	SPI-bus clock
	ReM_FPGA	”
extGlinkClk	RGTM-O	serial frame clock
intGlinkClk	ReM_FPGA	”

Table 6.2: The clocks signals produced in the ReM_FPGA.

clock (SYSCLKDLL), the feedback line connects the internal clock distribution network of the FPGA to the clock feedback pin of the DLL (CLKFB), to allow the DLL to eliminate the delay between the source clock and the individual loads within the FPGA. In all the other cases, the feedback line connects only the external output pin of the FPGA, through which the given clock pulse is delivered to an external device, to the corresponding CLKFB pin. Which means that the DLLs only eliminate the propagation delay from their output (CLK0) to the external output pin. In principle external feedback loops can be implemented, the layout of the PPM board contains such electrical lines. However, these feedback lines are currently disabled by the physical absence of a corresponding resistor. But, even if the lines would be activated, the actual gain of using the external feedback loops is not significant. That is because the feedback lines are not routed back from the clock *consumers*, as none of them provides such an output, but from different clock fan-out and distribution points on the PPM board, which in most of the cases are located at a closer distance to the ReM_FPGA than to the clock *consumers*. Which means that the actual propagation delay cannot be completely eliminated, and that the compensation which would be achieved with external feedback loops is comparable with the one achieved by the current implementation [Sch09].

For monitoring and debugging purposes, all the DLL-LOCKED signals are gathered into a VME status register (see table B.13). Additionally, the LOCKED signal of the MCMCLKDLL unit is routed off the chip to yet another clock multiplexing logic on the PPM board, to indicate that the ReM_FPGA is able to provide a reference clock to the PPrPHOS4s (Phos4Clk). This multiplexing logic ensures that the PPrPHOS4s are permanently supplied with a reference clock,

by selecting the PPM_XTAL as clock source whenever the ReM_FPGA's firmware is reloaded or not loaded at all. The clock multiplexing scheme is described in greater detail in the reference documentation of the PreProcessor Module [Han09b].

6.4 Distribution of Configuration Data

One of the important tasks of the ReM_FPGA is to realise the transfer of configuration data from VME to on-board locations. There are five types of configurable devices on the PPM to which the ReM_FPGA is connected: the PPrASICs, the TTCrx chip, the PPrAnIn-DACs, the PPrPHOS4s and the PPrMCM-LVDS transmitters. Each device provides a set of parameters that allow the control and customisation of their operation. All these parameters are mapped to the VME address space of the PPM, and the allocated VME addresses encode the necessary information to identify their location on the board. The ReM_FPGA decodes this information and transfers the data in the format requested by the destination device and by the protocol of the accessed interface. In parallel, the ReM_FPGA places two copies of the input VME configuration data in the SRAM. The first copy is stored as *reference*, while the other one is used to flag the change in configuration, and it is stored in SRAM until overwritten by related readback data.

Once the PPM is set up and ready to take data, the ReM_FPGA can be configured to operate in the *DAQ_Mode*. As long as this mode is enabled, the ReM_FPGA denies any configuration request for the real-time and readout paths. Additionally, the ReM_FPGA denies a configuration request when other processes, that access the SRAM device, are in progress. In both cases, the ReM_FPGA closes properly the VME cycle and flags the refusal via the two ReM_Error registers.

6.4.1 PPrASIC Configuration

The PPrASIC provides the largest number and variety of configurable parameters among all the PPM on-board devices. The operation of each PPrASIC channel can be configured via an individual set of 34 *channel registers*. Additionally, a set of 5 *global registers* provides common parameters for two channels controlled by the same serial interface, so that two such sets cover all four PPrASIC channels. A detailed description of the channel and global registers is given in the PPrASIC manual [Hus02]. Apart from registers, each channel contains two memories: an 8-bit deep x 11-bit wide *Playback Memory*, which is used for technical verifications of the digital part of the trigger system, and a 10-bit deep x 8-bit wide *Look-Up Table (LUT) Memory* which is used for fine energy calibration, pedestal subtraction and noise suppression within the real-time path.

Two methods are implemented in the ReM_FPGA for setting up the PPrASIC. In the first method, the ReM_FPGA receives the register or memory data directly from the VME, and transfers it to the PPrASICs within the same VME cycle. The second method is exclusively dedicated to setting up the Playback memory. The configuration data is first loaded to SRAM, and then the ReM_FPGA is requested to collect it and transfer it to the PPrASICs.

A special case is represent by the PPrASIC channel register that allows the control of the
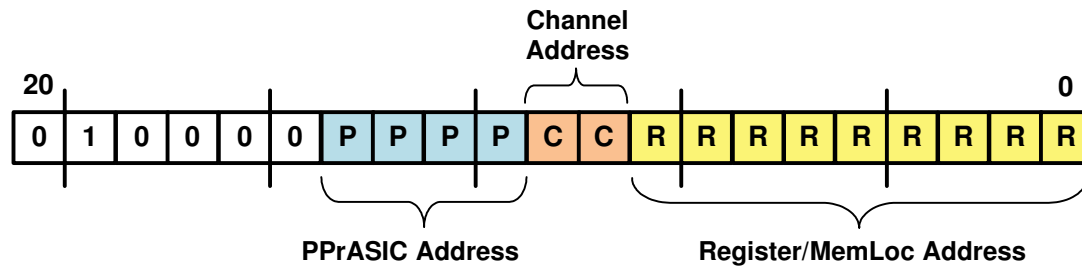
Figure 6.12: The encoding of the PPrASIC register and memory locations in the VME addressing lines. The scheme describes only the VME addressing lines which are routed to the ReM_FPGA (A[23:2], see also section 6.2.1).

Rate Metering and Histogramming operations, i.e. *ChannelReg17*. The *enable* bit fields of this register are set only via dedicated VME registers, in order to synchronise the respective operations in all the PPrASICs.

Loading from VME

The PPrASIC registers and memories are mapped differently to the VME address space. The registers are set up individually, thus each register is allocated an unique VME address. The memory locations are mapped in such a way that multiple consecutive locations can be configured via one VME address, within the same VME cycle (see appendix B, tables B.4 and B.5). This implementation reduces the time needed to load the PPrASIC memories, and provides an economic way for storing the corresponding data in SRAM.

In either case, the allocated VME addresses encode all the information needed to identify the location where the input VME data has to be transfered. Figure 6.12 shows the encoding scheme used for the PPrASIC registers and memories. The lower nine VME address bits (A[8:0]) provide simultaneously information about the type and the *local address* of the configurable location. The type indicates whether the incoming VME data is addressed to a channel or global register or a memory. The local address carries a different information for registers and memories. In the former case it indicates the register number, while in the latter case it indicates to which of the two memories is the data addressed. An individual memory location number is not needed, as the PPrASIC increments automatically the local write pointer with each configuration data received for the respective memory. Thus, the software must ensure that data is contiguously written to each of these memories. The following six bits of the VME addressing lines (A[14:9]) indicate the channel and the PPrASIC to which the register or memory belongs. The ReM_FPGA extracts this information and uses it separately for two purposes. The first purpose is to identify the serial interface on which the data has to be transfered. This is done by combining the channel and the PPrASIC addresses. The second purpose is to built a 13-bit *CfgAddress* command word, which has to be sent to the identified PPrASIC prior to the actual configuration data. The command indicates the configuration operation and the location on the PPrASIC where data has to be loaded. This latter information is built by the ReM_FPGA based on the type, channel and local addresses, and in the format requested by the PPrASIC (see also [Hus02]).

The configuration data is extracted from the input 32-bit VME data. All the PPrASIC locations are up to 11 bits wide, which means that each register or memory data can be transfered in one 13-bit frame over the serial interface. The format in which the configuration data is transfered over the VME to the ReM_FPGA is directly related to the way the register and memory locations are mapped to the VME address space. As the registers are set up individually, the corresponding configuration data is provided via the lower 11 bits of the VME data. In this case, the ReM_FPGA extracts only these bits and send them on the appropriate serial interface, in one frame. The Playback and LUT memory data are sent over the VME in the format indicated in tables B.4 and B.5. In this case, the ReM_FPGA separately extracts the data corresponding to each location, and sends it over the serial interface in the increasing order of the local memory address.

In parallel to transferring data to the PPrASIC, the ReM_FPGA places two copies of the 32-bit input VME data in the SRAM. The first copy is placed in an *MCM Reference* block, which holds copies of all configuration data transfered to PPrASICs, PPrPHOS4s and PPrAnIn-DACs. As each SRAM location is 36 bits wide, and the ReM_FPGA has access to all these bits, the upper four bits are automatically set as well. However, as neither the software nor the ReM_FPGA uses the upper four bits of an SRAM location in the Reference block, these bits are set for convenience to zero. The second copy of the VME data is placed by the ReM_FPGA in a second block, called *MCM Readback*, which stores data read back from the sources. In this case, the ReM_FPGA explicitly sets the upper four bits to value 4'b0001, to indicate that new configuration data has been loaded in the corresponding PPrASIC location, and that the content of the Readback location has to be updated with the actual data stored by the source. When the latter happens, the ReM_FPGA will set the upper four bits to value 4'b0000, to indicate that the data stored in the Readback location is *up-to-date* (see section 6.6.3). The SRAM locations, where the ReM_FPGA copies the configuration data, are indicated by the same VME address that encodes the PPrASIC information. The location in the Reference block is given by the lower 20 bits of the VME address, while the location in the Readback block is obtained by adding a constant offset to the previously obtained 20-bit address. This constant offset determines an identical structure and size for the Reference and Readback blocks (see also section B.2.1).

The way the ReM_FPGA handles the transfer of configuration data to both the PPrASIC and the SRAM is schematically described in figure 6.13 and explained in the following. The protocol with the VME_CPLD device, and subsequently with the VMEbus, is maintained in the ReM_FPGA by a *VmeManager* module. The module receives from the VME_CPLD control signals that indicate the validity of the input VME address and data (VmeAddrSelect, VmeDataSelect), and the type of the VME request (VmeWrite). Upon determining the beginning of a VME cycle, the VmeManager latches and decodes the input VmeAddress, to determine which functional module in the ReM_FPGA should take over and process the VME request. If the decoding identifies a request for transferring configuration data from VME to a PPrASIC location, the VmeManager distributes a strobe signal (ASIC_WriteStr) to an *AsicConfiguration* module, which masters all the write operations to the PPrASICs. Upon receiving this strobe, the module latches and processes the input VmeAddress and VmeData as described above, and sends the configuration data to a corresponding *AsicSerialInterfaces* module. As mentioned in section 6.2.3, the latter module contains the serial interface protocol, and it is instantiated 16 times in the firmware design, so that each instance realises the transfer of data over the two
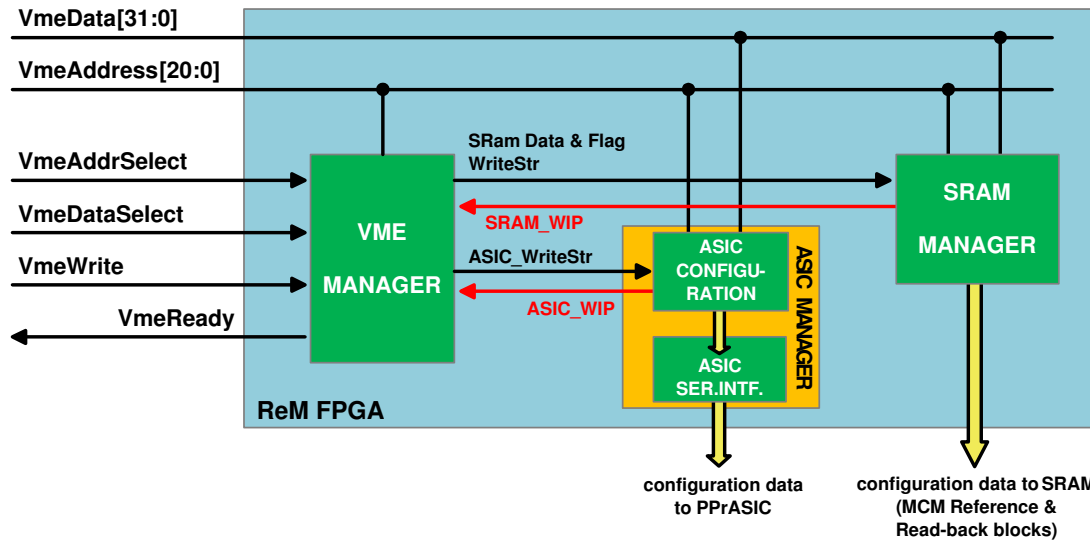
Figure 6.13: Control logic in the ReM_FPGA managing the transfer of configuration data from VME to the PPrASIC and SRAM.

serial interfaces of the one PPrASIC. In consequence, AsicConfiguration has to always provide AsicSerialInterfaces with data for both serial connections. As the VME request points to a certain PPrASIC location, which is served by only one serial interface, the AsicConfiguration will provide the identified serial port with the obtained CfgAddress word and the configuration data, and set the data for the second serial port to zero, which indicates the PPrASIC that the input has to be ignored.

Due to the dual information carried by the input VmeAddress, the same decoding performed in the VmeManager identifies a request for writing the input configuration data to the SRAM. Correspondingly, the VmeManager distributes simultaneously two strobes (SRamData_WriteStr and SRamFlag_WriteStr) to an *SramManager* module, which handles all the data transfers between the ReM_FPGA and the SRAM. The first strobe will determine the SramManager to copy the VmeData to the Reference block, while the second strobe will determine the SramManager to copy the same data to the Readback block and set the 4-bit flag accordingly.

Meanwhile, the VmeManager keeps the VME cycle open and monitors the status of the data processing in the AsicConfiguration and the SramManager. Each module provides the VmeManager with a *write-in-progress* signal (ASIC_WIP, SRAM_WIP), which is asserted upon the receipt of the VmeManager's strobes, and de-asserted when the data transfer is completed. The AsicConfiguration needs significantly longer time to complete the data transfer than the SramManager, due to the serial transmission. This time can extend up to 2 μs when the LUT memory is configured, as the AsicConfiguration has to transmit the 13-bit CfgAddress command and four 13-bits data words. In comparison, the SramManager transfers the same data in parallel and in a compact format, so that up to 4 clock periods are needed for each write operation. As both the PPrASIC serial interface and the interface to SRAM are operated with the same clock frequency, it means that the transfer of data to SRAM will be completed much earlier than the

transfer to PPrASICs. Therefore, when data is transferred to the PPrASICs the VmeManager monitors only the busy signal provided by the AsicConfiguration module, and closes the VME cycle upon detecting a high-to-low transition of this signal.

Loading from SRAM

As previously stated, the ReM_FPGA provides an alternative method for setting up the PPrASIC Playback memories. In this method, the playback configuration data is first loaded from VME to the SRAM, and then the ReM_FPGA is instructed to collect it and load it in the corresponding PPrASIC locations.

The VME data is stored in a dedicated block in SRAM, called *Reference Playback Patterns*, which is located outside the already mentioned MCM Reference and Readback blocks. The block is subdivided into eight sub-blocks, each of which provides space for 64 *playback patterns*[16], i.e. one pattern for each PPM channel. The format in which the playback data has to be loaded from VME and stored in the SRAM is identical with the format used in the previous method (see again table B.4). Additionally, the playback patterns have to be arranged, in each sub-block, in the increasing order of the PPM channel number, as shown in table B.8.

The transfer of data from VME to SRAM is handled by the VmeManager and SramManager functional blocks, and their communication is pretty much similar to the one previously described. As this operation does not imply a direct transfer of configuration data to any other on-board component, the VmeManager will issue only one strobe for the SramManager (SramData_WriteStr), and will close the VME cycle as soon as the SramManager indicates the transfer of the input 32-bit VME data is completed. Once the reference playback data is stored in SRAM, the ReM_FPGA can be instructed to pick it up and transfer it to the PPrASICs. The request is addressed via a the ReM_Command register. For the current purpose, the register provides eight commands, each one initiating the transfer of data from a single block to the PPrASICs. The bit field representation of the command register and the way the commands have to be addressed over the VME are described in appendix B, in section B.3.23.

Figure 6.14 describes schematically how the data transfer is realised. The VME command is received by two functional blocks: the *CommandRegister*, which holds the related logic for the ReM_Command register, and the *AsicPlaybackLoader*, which handles the transfer of data from SRAM to PPrASICs. The CommandRegister checks the validity of the VME request and, if the verification is successful, it distributes a LoadPlayback command to the AsicPlaybackLoader. Upon receiving this signal, the module first decodes the input VME data, to determine the SRAM block that has to be read out, and then it starts transferring the data. The module reads, via the SramManager, contiguously each memory location of the indicated block, and transfers the respective data to the AsicConfiguration module. In parallel with the latter operation, the AsicPlaybackLoader writes back the same data to SRAM, in the MCM Reference and Readback blocks, and requests the SramManager to set the readback flags accordingly. All the SRAM addresses, as well as the address which encodes the PPrASIC location where the data has to

---

[16]in this context, a playback pattern represents the total amount of 256 11-bit configuration data which is loaded in one PPrASIC Playback memory. The data may contain calorimeter-like pulses or data previously recorded by DAQ, and it is injected in the pre-processing chain to verify the digital part of the system. An application that uses playback patterns is described in the next chapter, in section 7.2.5.
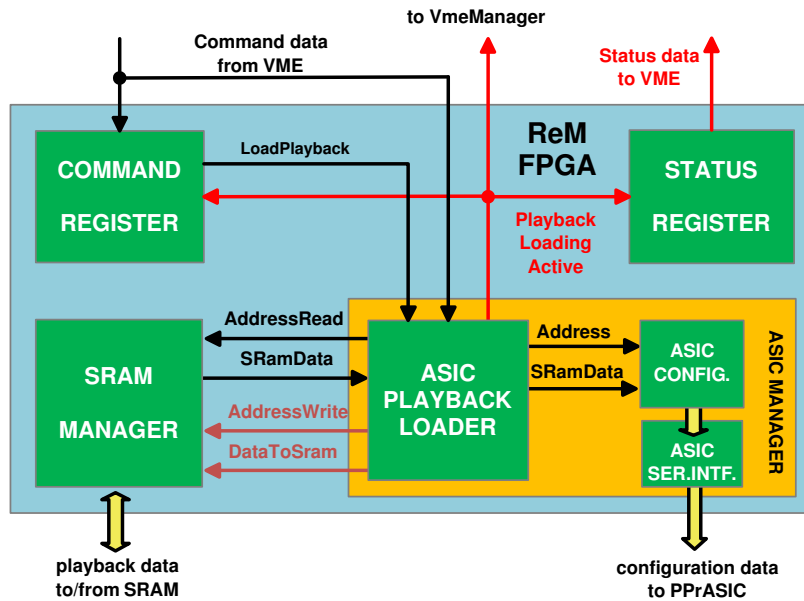
Figure 6.14: Control logic in the ReM_FPGA managing the transfer of playback patterns from SRAM to the PPrASICs.

be transferred, are generated by the AsicPlaybackLoader. Also, the same module generates all the read/write strobes expected by its communicating partners, and coordinates the data transfer according to the busy signals produced by these modules. This means that AsicPlaybackLoader will read a new SRAM memory location, only after the SramManager and the AsicConfiguration modules have signalised the completion of the current transfers.

The status of the entire operation is indicated by the AsicPlaybackLoader via the *Playback-LoadingActive* bit. The bit is asserted as soon as the module receives the LoadPlayback signal, and cleared immediately after the data from the last SRAM memory location is transferred to the PPrASICs. This bit is permanently delivered to three functional modules: the CommandRegister, VmeManager and the StatusRegister. As long as the bit is set, CommandRegister denies any further command to be executed, while VmeManager denies any VME request which implies an access to the SRAM. The StatusRegister provides the PlaybackLoadingActive bit to VME, via the second ReM_Status register.

Synchronisation of the Rate Metering and Histogramming Operations

The Rate Metering and Histogramming are two operational modules of the PPrASIC, which produce channel-wise and trigger independent energy rates and spectra. A detailed description of these monitoring tools is given in section 6.6.3. The respective operations are enabled or disabled via two bits in the PPrASIC channel register 17: *RateEnable* and *HisEnable* (see table 6.3). In order to synchronise the Rate Metering or the Histogramming operation in all 64 PPM channels, the RateEnable and HisEnable bits of each channel register 17 are set simultaneously, via dedicated VME registers (see section B.3.4). This is realised as follows:

Bit Nr.	Description
0	**RateEnable**
1	RateSource
2	**HisEnable**
3	HisSource
4-5	HisOpMode

Table 6.3: The bit field content of the PPrASIC channel register 17 [Hus02].

- when register 17 is configured separately for each PPrASIC channel, the ReM_FPGA ignores the values sent from VME for RateEnable and HisEnable, and sets these bits to zero. Simultaneously, the input VME data is stored in an internal 6-bit register, called *AsicReg17*;

- when an enable or disable request is received via the dedicated VME registers, the ReM_FPGA sets the RateEnable or HisEnable bits to the logic value "1" or "0". The other four bit fields of channel register 17 are set according to the values stored by the AsicReg17, to keep unchanged any previous configuration. The resulting 6-bit configuration data is then sent simultaneously over all 32 serial interfaces.

The consequence of this implementation is that any configuration strategy should foresee identical settings, in all 64 channels, for each of the other four parameters of register 17. Otherwise, the ReM_FPGA can overwrite a previous configuration when using the data stored in the AsicReg17.

All these operations on the channel register 17, and the simultaneous transfer to the 16 PPrASICs, are performed in the ReM_FPGA by the AsicConfiguration. The module first builds a common 13-bit CfgAddress, which indicates that the incoming configuration data has to be written to both channel registers 17 served by a given serial interface. Then, it loads, at subsequent steps, the CfgAddress and the configuration data to all 32 output shift registers, to realise the simultaneous transfer to the 16 PPrASICs. For monitoring and debugging purposes, AsicConfiguration flags permanently the values loaded in the PPrASICs for the RateEnable and HisEnable bits. The respective flags are mapped in the first ReM_Status register, as *RateMeterIsEnabled* and *HistogrammingIsEnabled* (see table B.22).

6.4.2 PPrAnIn-DAC Configuration

Trigger Configuration Data

As mentioned in section 6.2.5, the analogue processing performed on the four PPrAnIn boards provides two configurable parameters for each channel: an 8-bit *offset*, to shift the baseline of the corresponding single-ended signal, and an 8-bit *threshold* to detect the rising edge of the same signal. These parameters are loaded from VME, via the ReM_FPGA and a corresponding SPI bus, into 8-channel serial DACs. Each PPrAnIn board is equipped with four such devices,
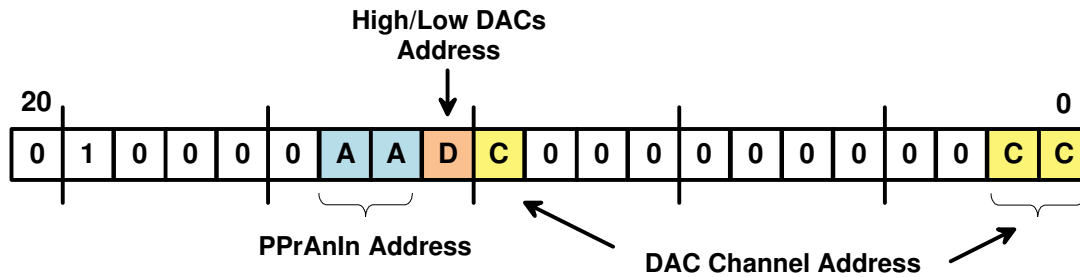
Figure 6.15: The encoding of the PPrAnIn-DAC locations in the VME addressing lines.

as shown in figure 6.7. The leftmost two DACs, labelled as the *High DACs*, provide offset and threshold voltages for the analogue processing of the first 8 trigger signals, while the remaining two DACs, the *Low DACs*, provide similar output for the analogue processing of the other 8 trigger signals.

The offset and the threshold parameters of each trigger channel are mapped together in one VME address, so that they are simultaneously set within the same VME cycle (see table B.3). As for the PPrASIC registers and memories, the allocated VME addresses encode the necessary information to identify their location on the PPM. Figure 6.15 shows the corresponding encoding scheme. The lower two bits and the twelfth bit of the VME address indicate together the DAC channel that has to be configured. The thirteenth bit points to one of the two groups of DACs, while the following two bits identify the PPrAnIn board on which the DACs are located, and thus the SPI bus over which the data has to be transferred.

Each PPrAnIn-DAC is programmed via 16 data bits. The upper eight bits contain an address information, which identifies one of the eight DAC channels. The ReM_FPGA builds this address, based on the channel information encoded in the VME address, and in the format expected by the device (see the related documentation for more details [Max94]). The lower eight bits contain the actual configuration data, which the ReM_FPGA extracts from the input 32-bit VME data. As described in section 6.2.5, the four DACs of one PPrAnIn board are daisy-chained, so that each write operation to any of these DACs implies a transfer of 64 data bits. As the offset and threshold parameters are set together during the same VME cycle, it means that always two DACs will be simultaneously programmed. The 32 data bits designated to these DACs will contain the address and the configuration data delivered from VME, while the other 32 data bits will all be set to zero, which indicates the respective DACs that the input data must be ignored.

The transfer of data from the VME to the PPrAnIn boards is handled by the ReM_FPGA in a similar manner as in the case illustrated in figure 6.13. Upon detecting a configuration request for the PPrAnIn-DACs, the VmeManager distributes corresponding strobe signals to an *AnInManager* functional module, which manages the processing and transfer of data to the PPrAnIn-DACs, and to the SramManager, which will place two copies of the input data in the SRAM. The major difference with respect to the mentioned case, is that the VME cycle cannot be kept open until the AnInManager completes the data transfer, as 32 μs are needed to transfer the 64 data bits. As a consequence, the AnInManager generates five busy signals. The first one indicates that the module has finished processing the data in the format required by the
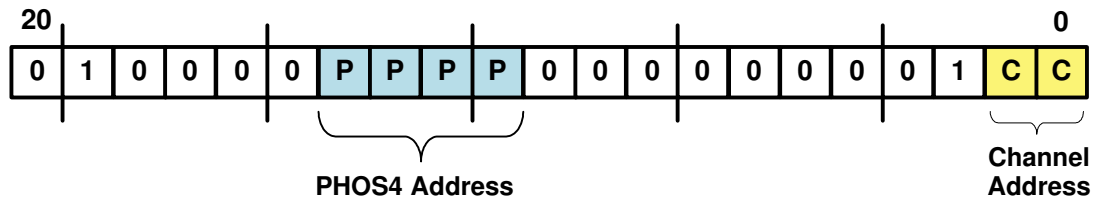
Figure 6.16: The encoding of the PPrPHOS4 locations in the VME addressing lines.

PPrAnIn-DACs. This signal is delivered to the VmeManager module, which closes the VME cycle upon its arrival. The other four bits, *AnIn1_Spi_WIP*, *AnIn2_Spi_WIP*, *AnIn3_Spi_WIP* and *AnIn4_Spi_WIP*, indicate the progress of the data transfer over a corresponding SPI bus, and they are mapped into the first ReM_Status register [Sch09].

The DAC Full-Buffered Mode

The PPrAnIn-DACs can be programmed to operate in a *Full-Buffered Mode*, which provides a more precise conversion of the digital input and a faster settling time of the analogue output than the default *Unbuffered Mode*. For this purpose, eight VME registers are provided (see table B.1). The data content expected by the ReM_FPGA from VME, and the sequence in which these registers should be accessed is presented in section B.3.17. Also, the eight registers are mapped outside the VME-SRAM address space. Which means that in this case the input VME data will not be copied to SRAM. Furthermore, the transfer of data from VME to PPrAnIn-DACs is handled only by the VmeManager and the AnInManager functional modules, as described at the previous point.

6.4.3 PPrPHOS4 Configuration

The PPrPHOS4 provides only one 5-bit configurable parameter for each of its four channels. As a consequence, the VME addresses allocated for transferring configuration data to the PPrPHOS4s encode only two parameters: a 2-bit *ChannelAddress*, which identifies one of the four channels, and a 4-bit *Phos4Address*, which points to one of the 16 PPrPHOS4 devices (see figure 6.16).

The 5-bit configuration data is loaded from the VME, via the ReM_FPGA and the corresponding I^{2}C bus. Table 6.4 lists the 5-bit data expected from VME and the significance of each setting. The first 28 values represent the actual PPrPHOS4 configuration data. The first 25 values, i.e. 0 to 24, specify the number of nanoseconds by which the FADC strobe is delayed with respect to the reference clock. The value 26 activates the phase detector output described in section 6.2.7, while values 25 and 27 determine the PPrPHOS4 to generate a constant output signal. The next three values listed in the same table, i.e. 28 to 30, are redefined in the ReM_FPGA to a valid PPrPHOS4 configuration setting, and they are only used for test and debugging purposes.

The last setting, i.e. 31, is used for initialising the I^{2}C related logic of the PPrPHOS4s. This implementation was determined by some operational aspects of the PPrPHOS4. First of all, the device does not provide a reset function. The only possibility to reset its logic is by power cycling

VME Input Data	Significance
0-24	delay in ns
25	constant 0 at output
26	phase detector output
27	constant 1 at output
28	redefined as 25 (for tests only)
29	redefined as 27 (”-”)
30	redefined as 0 (”-”)
31	PPrPHOS4-I^2C initialisation

Table 6.4: The expected 5-bit VME configuration data for the PPrPHOS4s (adapted from [CER92]).

the PPM board. Moreover, after it is supplied again with power, the chip will initialise its I^2C related logic only on the occurrence of the first I^2C clock event. However, the ReM_FPGA does not distribute permanently this clock signal to the PPrPHOS4s, but only during a data transfer over the bus. Which means that the first 8-bit data sent by the ReM_FPGA will not be processed because the PPrPHOS4s will be busy initialising their I^2C interface logic. For this reason, it was chosen to implement a *dummy* write operation to the PPrPHOS4s, via the setting 31, in order to stimulate the initialising procedure. In the beginning of this operation the ReM_FPGA sends an *invalid* I^2C slave address, so that none of the PPrPHOS4s will try to process the input data in the eventuality that the initialisation is finished before the dummy I^2C transfer is completed.

The I^2C slave addresses generated by the ReM_FPGA for the PPrPHOS4s are given in table 6.5. Additionally, the table lists the I^2C addresses generated for the TTCrx, but they will be explained in the next subsection. The four bits given in bold ([5:2]) represent the *PHOS4Address* which the ReM_FPGA extracts from the input VME address. These bits are compared by each PPrPHOS4 against an unique 4-bit *PPrMCM slot address*. The latter is hardwired on the PPM motherboard and delivered to the PPrPHOS4 through one of the PPrMCM connectors [Sch09].

I^2C Address	Slave Device
7'b0**00_00**00	PPrPHOS4_#1 (upper PPrMCM on the PPM board)
7'b0**00_01**00	PPrPHOS4_#2
...	...
7'b0**11_11**00	PPrPHOS4_#16 (lower PPrMCM)
7'b000_0010	TTCrx (I2C_pointer register)
7'b000_0011	TTCrx (I2C_data register)
7'b100_0000	dummy PPrPHOS4 address (initialisation)

Table 6.5: The I^2C slave addresses generated for the PPrPHOS4s and the TTCrx.
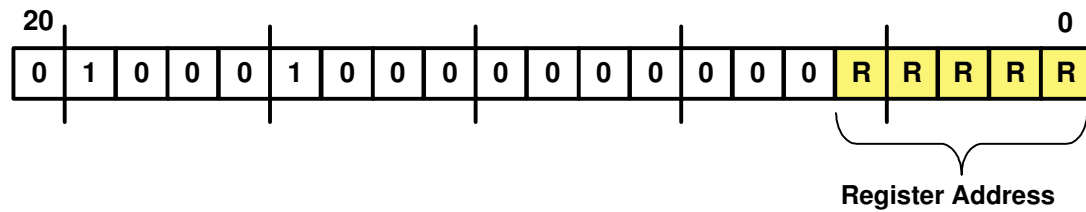
Figure 6.17: The encoding of the TTCrx register number in the VME addressing lines.

Apart from the slave address, the ReM_FPGA provides also a local address, which indicates the PPrPHOS4 channel to which the input configuration data is addressed. This information is given by the 2-bit *Channel Address* encoded in the VME address, and it is packed together with the 5-bit configuration data in one 8-bit data word, as indicated in the specifications of the device [CER92].

The transfer of configuration data from the VME to the PPrPHOS4s is handled by the Vme-Manager and the three functional modules previously shown in figure 6.8: Phos4TTCrxData, I2CDataTransfer and I2CMasterCore. Upon receiving a corresponding write strobe from the VmeManager, the Phos4TTCrxData module enables the transfer the PPrPHOS4-I^2C bus, and processes the input VME address and data. The obtained 7-bit slave address and the 8-bit data word are then passed to the I2CDataTransfer module. As mentioned in section 6.2.6, the I2CDataTransfer is the actual master of the I^2C communication, while I2CMasterCore only provides the interface to the I^2C bus. The communication between the two modules and the data transfer over the I^2C bus is realised sequentially. First, the I2CDataTransfer appends a write bit to the 7-bit slave address, and then sends the resulting byte to I2CMasterCore, requesting this data to be transferred over the I^2C bus. The I2CMasterCore serialises the received data and flags, via local registers, the progress of the transfer. The I2CDataTransfer checks permanently this status and sends the next data byte, containing the actual configuration data, when the previous transfer is completed.

In parallel with this transfer, the input VME data is copied to the SRAM. As the whole I^2C cycle takes much longer than the transfer to SRAM, the VmeManager will close the VME cycle as soon as the SramManager has flagged the completion of its operations. The progress of the I^2C transfer is indicated by the Phos4TTCrxData. The module asserts a corresponding bit, i.e. *Phos4_WIP*, after the input VME address and data has been latched, and clears it immediately after the I2CDataModule indicates that I2CMasterCore has finished transferring the last data byte. The status bit is mapped in the first ReM_Status register.

6.4.4 TTCrx Configuration

The TTCrx provides twenty 8-bit user-accessible registers for controlling and monitoring its operation. As only one TTCrx chip is present on the PPM, the VME addresses allocated for configuring the device encode only one parameter: *RegisterAddress* (see figure 6.17).

The transfer of configuration data from VME to the TTCrx is realised in a similar manner as in the case presented in the previous subsection. There are a few differences. First of all,

upon receiving a write strobe indicating a write operation to the TTCrx, the Phos4TTCrxData module enables the TTCrx-I^{2}C bus, while it simultaneously disables the PPrPHOS4-I^{2}C bus. Then, during a write operation to the TTCrx, the ReM_FPGA has to generate two I^{2}C slave addresses (see again table 6.5). That is because the TTCrx occupies two consecutive addresses in the I^{2}C space [Chr04]. The first address localises an I2C_pointer register on the device, to which the ReM_FPGA has to transfer the VME RegisterAddress parameter. The second address localises another register, I2C_data, to which the ReM_FPGA has to transfer the VME configuration data. Another particularity of the current operation is given by the fact that the numbers allocated by the TTCrx to its user-accessible registers are not contiguous (see table B.6). This numbering leads to "gaps" in the VME address space. In case a VME request is received via one of these invalid addresses, the ReM_FPGA will not process it, and it will correspondingly assert a bit in the first ReM_Error register (see *Denied_TTCrx_UnReg* in table B.26). Also, as in the previous case, the status of the data transfer to TTCrx is indicated via a dedicated bit in the first ReM_Status register.

As in the case of the data transfer to the PPrASICs, the ReM_FPGA places two copies of the input VME data in the SRAM. The first copy is written to a *TTCrx Reference* block, which holds copies of all configuration data transfered to the TTCrx, while the second copy is written to a *TTCrx Readback* block, which stores data read back from the source (see also section B.2.2). Additionally, the ReM_FPGA sets the upper four bits of the corresponding SRAM location to value 4'b0001, to indicate that new configuration data has been loaded to the given TTCrx register.

6.4.5 PPrMCM-LVDS Configuration

The operation of the PPrMCM-LVDS transmitters can be controlled via three parameters: *TClkRF*, *DEn* and *LvdsSync*. The first parameter allows to select either the rising or falling edge of the operational clock for strobing in the PPrASIC data. The second parameter enables the output of the transmitter, while the third parameter determines the same device to send a synchronisation pattern. All three parameters are configured via the ReM_FPGA's *MCM_Control* register (see B.3.17).

6.4.6 Configuration Restrictions

As mentioned in the introductory part of this section, the ReM_FPGA denies a VME configuration request in two cases. The first case can be directly induced, by configuring the ReM_FPGA to operate in the DAQ_Mode. As long as this mode is active, any configuration request, that implies a change of the pre-processing and readout operations, is denied. This refers to configuration requests addressed to the PPrASICs, PPrAnIn-DACs, PPrPHOS4, TTCrx, PPrMCM-LVDS transmitters, and to those internal registers of the ReM_FPGA that allow the configuration of the event data processing and transmission. In the case of the PPrASIC, the channel registers that allow the configuration and the control of the Rate Metering and Histogramming operations are excepted from this restriction, since the respective monitoring tools are completely decoupled from the readout path and they only use the pre-processing results (see also section 6.6.3).

In the second case, the ReM_FPGA denies a configuration request if another process, that demands access to the SRAM device, is in progress. The restriction is mainly necessary because the SRAM is a single-ported memory, and thus only one internal process can access it at a time. One can probably implement an algorithm that queues other processes demanding access to SRAM until a current one is completed, but, besides of a significant increase in the usage of FPGA resources, this will considerably slow down the operations executed the ReM_FPGA. This restriction applies for any configuration request addressed to the PPrASICs, PPrAnIn-DACs, PPrPHOS4, TTCrx, the PPrMCM-LVDS transmitters.

In both cases, the ReM_FPGA will close immediately the VME cycle and flag the refusal via dedicated bits in the two VME error registers (see also section B.3.24).

6.5 PPrASIC Event Data Formatting and Transmission to DAQ

A mandatory task of the PPr system is to provide event related data to the DAQ System, in order to allow the verification, calibration and monitoring of the trigger. The event related data is accumulated in the PPrASICs, and transferred over the serial interfaces to the ReM_FPGA upon the receipt of the L1A signal. In the ReM_FPGA, the event data is collected in dedicated buffers, processed in the specified ATLAS format, and transmitted to the DAQ system, via the RGTM-O device and a corresponding ROD module.

6.5.1 PPrASIC Event Data Format and Transfer on the Serial Interface

The PPrASIC event data mainly consists of 10-bit raw FADC data and 8-bit BCID-LUT results. For each of its four channels, the PPrASIC continuously extracts these data from the real-time pre-processing path, and stores them in two separate pipeline memories (see e.g. figure 5.7). Each memory is 7-bit deep x 11-bit wide, and is operated at the LHC clock frequency. Which means that each 11-bit data word is available in the pipeline memory for roughly 3.2 μs. This period of availability extends over the 2.5 μs overall latency of the L1 trigger, and, in addition, it compensates for the propagation time of the L1A signal from the CTP to the input of the PPrA-SICs. Apart from the raw FADC and the BCID-LUT data, the PPrASIC stores three additional 1-bit parameters in the pipeline memories: the ExtBCID signal provided by the PPrAnIn board, and the results of the internal linear and saturated BCID algorithms, i.e. PeakFinding BCID and Saturated BCID. These parameters are often referred to as the *BC Marks*, as they indicate the bunch-crossing that contains the maximum of the input calorimeter signal.

The readout of event data from the pipeline memories and its transfer over the serial interface are initiated by the L1A signal. On the arrival of this signal, each data word related to the accepted event is copied out from the pipeline memories into another set of buffers, called *derandomisers*, and then transferred over a corresponding serial interface to the ReM_FPGA. The derandomisers have an identical size with the pipeline memories, and they compensate for the L1A rate fluctuations, by storing the event data words until their transfer over the serial interface is completed. The number of event related data words, which are read out from each channel-memory, is a configurable parameter, and it represents an important aspect for both the operation of the DAQ and the algorithms implemented in the ReM_FPGA. The PPrASIC allows up to 127

(a) [Han09b]   (b) adapted from [Hus02]
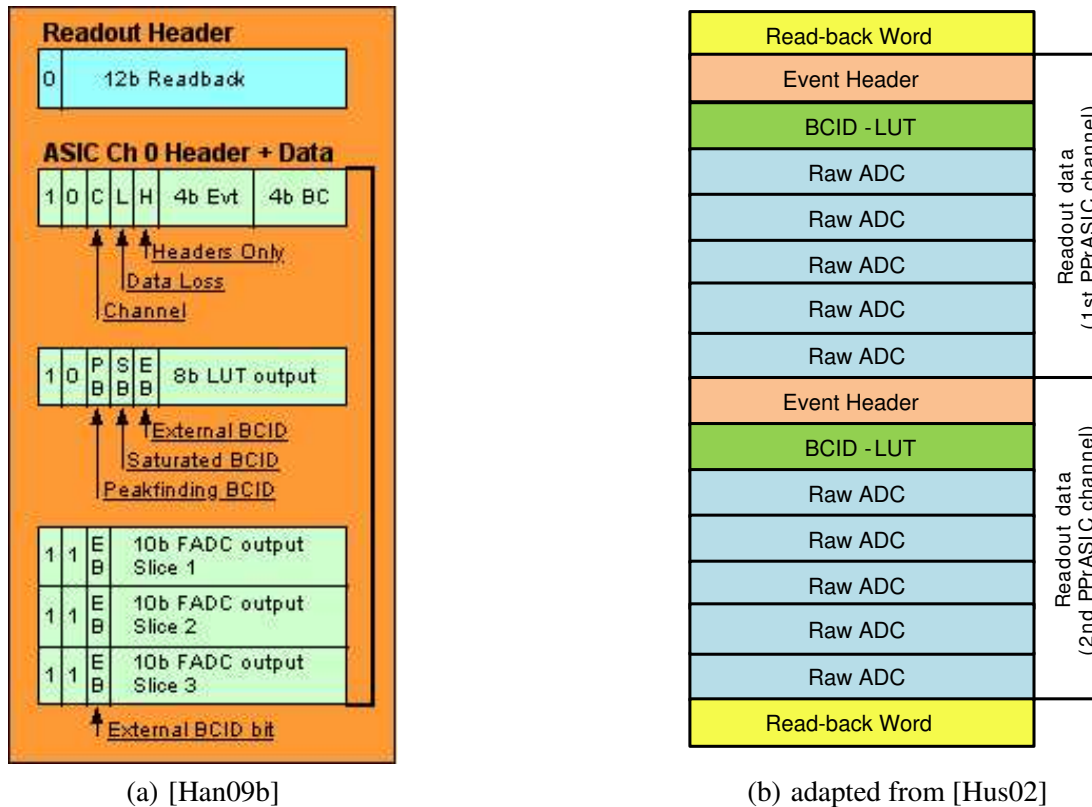
Figure 6.18:   The PPrASIC event channel-data format (a), and the multiplexing scheme of the Readback
and Readout channels on the serial interface (b). The former depicts the case of a readout
with one BCID-LUT and three FADC samples, while in the latter five FADC samples are
read out.

raw FADC and 7 BCID-LUT samples to be read out. However, in normal physics running, the
DAQ can maximally handle a readout with five FADC samples, because the separation between
two consecutive L1As is constrained by the CTP to a minimum of five bunch-crossings. For
calibration runs, the DAQ can be configured to operate with a slightly larger number of readout
samples, but still well below the maximum provided by the PPrASIC. As it will be described in
section 6.5.3, the ReM_FPGA can process and provide to DAQ only readouts of maximum 16
samples (FADC+BCID), due to limited internal buffering resources. Outside the DAQ mode,
for technical verifications of the system, the ReM_FPGA can buffer and provide via the VME
interface the entire readout range of the PPrASIC (see section 6.8).

   The format and the succession in which the PPrASIC transmits the event channel-data over
the serial interface is shown in figure 6.18a. The first data written out is an 11-bit *EventHeader*
word, which the PPrASIC prepares in parallel with accumulating data in the pipeline memories,
and stores in the BCID-LUT derandomiser upon the arrival of the L1A. Most of the information
provided by the EventHeader is processed by the ReM_FPGA, in order to verify the synchroni-
sation between the two devices, and to set status bit fields in the readout stream to DAQ. The

first eight lower bits of the header equally map the output of the PPrASIC bunch-crossing[17] and event counters. The remaining three bit fields of the header provide status of the readout operation. The *Channel* bit indicates to which of the two PPrASIC channels, served by the given serial interface, belongs the incoming event data. The *HeadersOnly* bit indicates that one of the two derandomisers is almost full, and the available space allows only to copy the headers, while the *DataLoss* bit indicates that the derandomisers are full and all the event data is lost. The next data words contain the actual event data. The BCID-LUT data is provided together with the corresponding "BC Marks", while the raw FADC data is packed together with the ExtBCID bit, to indicate the response of the PPrAnIn comparator-threshold for the given digitised value.

The upper two bits of each 13-bit data word shown in figure 6.18a are used as flags, to help the receiving device distinguish between the different data types sent by the PPrASIC over the serial interface. As mentioned in section 6.2.3, the PPrASIC provides four types of data: configuration, monitoring, event and status data. The transmission protocol groups these data types into three streams, which are multiplexed on the serial output into a fixed scheme, as illustrated in figure 6.18b. The first stream contains the configuration, monitoring and status data, and it is called *Readback Channel*. The other two streams contain the event data from the two channels served by the serial interface, and they are called *Readout Channels*. The identification of the Readback and Readout words is done via the two most significant bits of each data word. When the bit is set to 0 it indicates a Readback Word, and when is set to 1 it indicates a Readout word (see again figure 6.18a). Subsequently, the second most significant bit is used to distinguish between the different data types within each channel. For the Readout channel, the EventHeader and the BCID-LUT words are flagged with a 0, while the FADC words are flagged with an 1. The Readback channel uses a slightly different scheme for flagging its own data words, the third most significant bit being used to identify the status data. The content of the Readback channel and the flagging scheme are described in section 6.6.1.

When no event data has to be transmitted, the PPrASIC sends only Readback words on the serial interface. As soon as an L1A signal is received, and the event data is ready for transmission, the PPrASIC sets the highest priority for the Readout channels. This means that the Readback channel is delayed until the transmission of the current event data is completed. Also, depending on the L1A frequency, a new event data block may be already available in the derandomisers before the transmission of the previous event block is completed. In such cases, the PPrASIC intercalates one Readback word in between the two event data blocks, to additionally help the receiving end identify the start and the end of an event data blocks (see again figure 6.18b).

6.5.2   Reception of PPrASIC Data in the ReM_FPGA

In the ReM_FPGA, the multiplexed data stream is received by a corresponding AsicSerialInterfaces module. As already mentioned in the previous sections, there are 16 such modules in-

---

[17]the actual PprASIC bunch-crossing counter is 12-bit wide, covering the total length of the LHC orbit, i.e. 3564 bunch-crossings. For bandwidth considerations, only the lower 4 bits of the counter output are transferred over the serial interface. The respective value is sufficient to determine the synchronisation of the PPrASIC with the ReM_FPGA and the other trigger components. The full output of the counter is internally used by the Histogramming operation (see section 6.6.3)
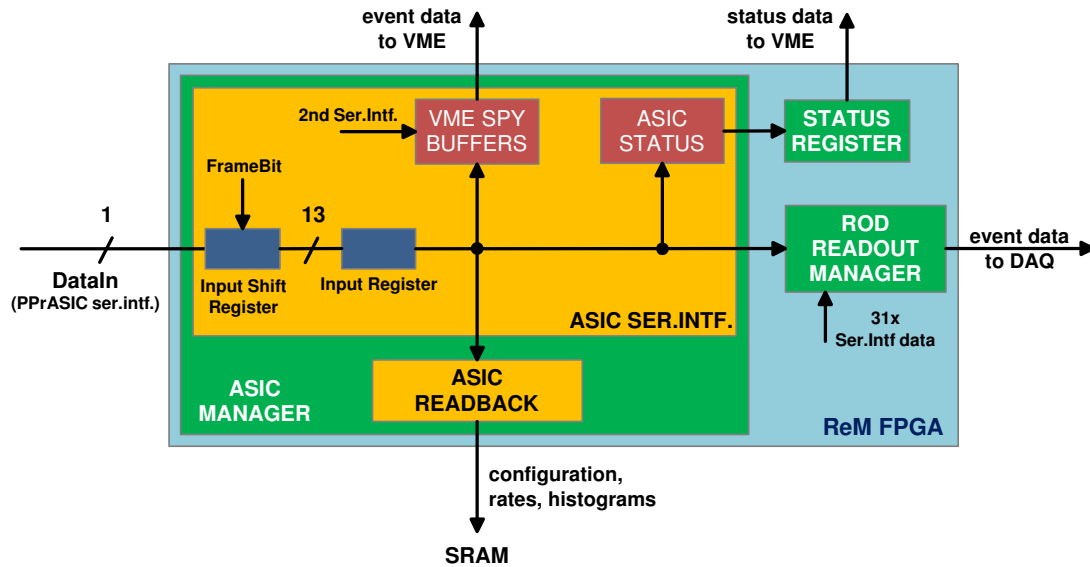
Figure 6.19: The fanning out and the processing of the PPrASIC serial interface data in the ReM_FPGA.

stantiated in the design of the ReM_FPGA, each instance handling the data transfer, in both directions, over the two serial interfaces of one PPrASIC. After it is converted from serial to parallel format, the input data is fanned out into four streams, and provided to four different internal logic units, for further processing (see figure 6.19). This means that the de-multiplexing of the input data stream is performed locally, each logic unit selecting only the relevant data. Two of the fanned out data streams are processed by the AsicSerialInterfaces module. In one case, the event data block is extracted and stored, in the original format, in internal VME-accessible memory buffers. In the second case, the AsicSerialInterfaces module extracts the status information from the Readback channel, and provides it to VME via the StatusRegister module. The third copy of the input PPrASIC data stream is routed to the *AsicReadback* module, which extracts the configuration and monitoring data from the ReadbackChannel, and writes it to SRAM. These three operations will be described in section 6.6. The last copy is delivered to the *RodReadoutManager* module, which has the tasks of collecting, processing and transmitting the event data from all 32 serial interface streams to the RGTM-O device. The functionality of the RodReadoutManager is described in the following subsections.

6.5.3 Collecting the PPrASIC Event Data

The data collection task implies to extract the event related data and write it to local memory buffers. The role of the memory buffers is similar to that of the PPrASIC derandomisers. They compensate for the L1A rate fluctuations, by storing the current event data until the transfer to the DAQ is completed. Additionally, the buffers act as *synchronisers*, at the boundary of two different clock domains. As mentioned in sections 6.2.3 and 6.2.4, the serial interfaces to the PPrASICs and the RGTM-O are operated at different clock frequencies. The interface to
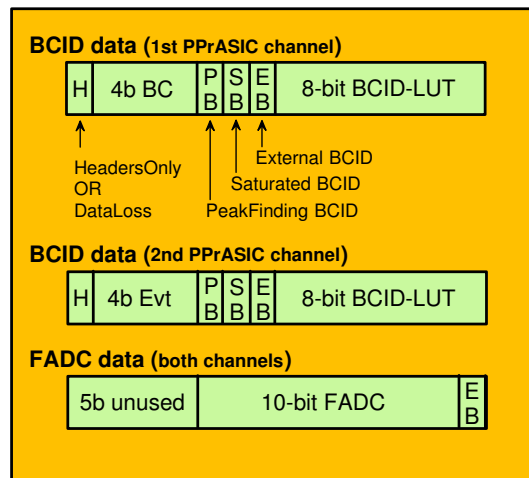
Figure 6.20: Longitudinal storage of the input PPrASIC event data in the local dual-ported memories.

PPrASIC is driven by the 40.08 MHz LHC clock, while the interface to RGTM-O uses the 40.00 MHz PPM_XTAL clock. As the two clock domains appear asynchronous to each other, the RodReadoutManager must ensure that data is synchronised, when it passes from the first clock domain into the second one, in order to avoid metastability failures. Although, for a data bus with multiple consecutive values, asynchronous FIFOs (First-In First-Out) are the most appropriate *synchronisers*, the current implementation uses dual-ported memories, for the flexibility they offer in terms of operating the read and write pointers.

The de-multiplexing and the extraction of event data is performed separately for each of the 32 streams. Also, one 8-bit deep x 16-bit wide dual-ported memory is employed to store the selected data from one input stream. In order to separate off the Readout data from Readback data, the RodReadoutManager monitors the most significant bit of each incoming data word. A transition from 0 (Readback) to 1 (Readout) indicates the beginning of the event data block. Upon detecting it, the RodReadoutManager discards the upper two control bits of each following BCID-LUT and FADC data word, and writes the remaining 11 bits to the memory. Additionally, most of the data carried by each EventHeader word is written as well to the memory. This data will be later used for setting status bits in the readout stream to DAQ. Figure 6.20 shows the longitudinal format in which the event data is stored in the dual-ported memories. The BCID-LUT data and the "BC Marks" are stored exactly in the format provided by the PPrASIC, while the 10-bit FADC data and the ExtBCID bit are first swapped, and then written to the buffers. In both cases, this is the format in which the respective 11-bit data is transmitted to the ROD. For an efficient usage of the memory space, the EventHeader data is written to the same memory locations that store the BCID-LUT data and the three "BC Marks". The HeadersOnly and DataLoss bits of each header are first logically OR'ed, and then each result is written in the upper position of the corresponding memory location. The 4-bit bunch-crossing number is extracted only from the first header, and it is stored together with the data of the first channel. The value is extracted only from one header because the bunch-crossing counter is globally implemented in the PPrASIC, and in consequence, all four channel-headers generated by the

PPrASIC during an event readout provide the same counter value. A similar implementation is adopted for the event number, which, for simplicity of operations in the firmware, it is extracted only from the second header, and stored in the memory together with the data of the second channel. The Channel bit of the EventHeader is discarded, the respective information being deduced from the structure of the PPrASIC event data block.

Another important aspect of the buffering process is the transversal storage of the event data words in the dual-ported memories. As it will be described a bit later, the processing in the specified ATLAS format requires to serialise the event data from the two serial interfaces of a given PPrASIC, by concatenating the two event blocks. Which means that the readout of the event data from the memory buffers and its transmission to the DAQ necessitates longer time than writing the same data to the buffers. At medium and high L1A frequencies this leads to multiple consecutive events accumulated in the buffers, possibly hitting the storage limits. In these situations, the RodReadoutManager must ensure that only integral event blocks are stored in the buffers, and that the unread event data is not overwritten by further incoming event data. In order to accomplish these requirements, logic in the RodReadoutManager divides the entire depth of each memory buffer into equally sized *event blocks*. The number and the depth of these blocks depend on the size of the PPrASIC event block, i.e. they depend on the number of FADC and BCID-LUT samples which the PPrASIC is configured to read out from its pipeline memories. The current design of the ReM_FPGA offers support for six different combinations of FADC and BCID-LUT samples, under the assumption that all the PPrASICs are configured to read out the same number of samples. Also, the ReM_FPGA needs to know in advance the settings applied to the PPrASICs. This is done via a dedicated VME register (see section B.3.14).

Table 6.6 lists the supported combinations of readout samples. They were chosen to meet different operational necessities. In normal physics running, the DAQ uses only the '3+1' or '5+1' readout modes, in order to cope with the transport limitations at maximum L1A rate. The other modes are intended for calibration studies that check the properties of the calorimeter trigger pulses or the BCID algorithms. Also listed in table 6.6 is the number of event blocks that the RodReadoutManager allocates for each readout mode. This number is determined in the following way:

Selected readout scheme $(N_{FADC} + N_{BCID-LUT})$	No. event blocks (in DP memories)
3+1	16
5+1	**16**
7+1	16
9+3	8
11+5	8
15+1	8

Table 6.6: The combinations of FADC and BCID-LUT samples currently supported by the ReM_FPGA's firmware. The values given in bold specify the default mode.

- if the sum of the FADC and BCID-LUT numbers is smaller or equal to 8, then the 256 locations of each memory are organised into 16 blocks, so that each block contains 16 memory locations.

- if the same sum is greater than 8 and smaller or equal to 16, then the depth of the memory is organised into 8 blocks, each of which contains 32 memory locations.

In case of the '3+1', '5+1' and '9+3' readout modes, the allocated memory blocks will not be fully occupied. For example, if the PPrASIC is configured to read out 5 FADC samples and one BCID-LUT sample, then the event data block transmitted over the serial interface will consists of 14 data words: ten FADC, two BCID-LUT and two EventHeader data words. Since the EventHeader data is stored together with the BCID-LUT data in the local buffers, this means that only 12 out of the 16 memory locations will be actually written with data. The other four memory locations are left empty, the write pointer being moved to the start address of the next memory block, when a new PPrASIC event block is received. The advantage of this method is that the PPrASIC event data blocks are integrally stored in the local memories. Additionally, with this method new readout modes can be easily added, under the assumption that the sum of the readout samples is not greater than 16.

6.5.4   The G-Link Event Data Format

As mentioned in section 6.2.4, the transfer of event data from the ReM_FPGA to the RGTM-O is realised via 16 single data lines, each of which is used for transporting the event data provided by a corresponding PPrASIC. The serial format in which the data is written out to RGTM-O is shown in figure 6.21, and it is often referred as to the *G-Link Event Data Format*. Apart from PPrASIC event data, each serial stream transports as well three other types of data: a global 12-bit Bunch-Crossing (BC) number, a set of 10 error bits (Errors) and a longitudinal parity bit (GP). The four components of the serial streams are described in the following.

The Bunch Crossing Number

The ReM_FPGA has to tag each event with the BC number of the corresponding L1A, in order to allow verifications of the PPM timing with respect to the other components of the trigger. The BC number is provided by a 12-bit counter running in the RodReadoutManager. The counter is incremented with each LHC clock event, and it is cleared by the TTC Bunch Counter Reset (BCR) signal. Upon the receipt of an L1A, the output of the counter is written to a 16-bit deep x 12-bit wide asynchronous FIFO, where it is queued until the RodReadoutManager extracts it for processing and transmission to DAQ.

In the G-Link event data format the 12-bit BC number precedes the actual event data, and it is transported transversally by the first 12 serial frames. The corresponding bit field in the format of the last four serial frames is, therefore, always set to zero (see again figure 6.21).

The PPrASIC Event Data

For each serial frame, the RodReadoutManager has to concatenate the event data originating from the four channels of one PPrASIC, in the succession indicated in figure 6.21. First the

event data of the first two channels is transmitted, labelled in the same figure as ASIC Ch_A and ASIC Ch_B, and then the event data of the other two channels, ASIC Ch_C and ASIC Ch_D. Each channel block consists of 11-bit FADC and BCID related data, which are serialised exactly in the format in which they are stored in the dual-ported memories (see again figure 6.20).

The Error Bits

A set of 10 error bits is provided in each serial frame, to indicate the status of the PPM readout operation. The first five error bits are set from VME, while the remaining five bits are set by the RodReadoutManager, as a result of an internal processing. The significance of the error bits, and the way they are set is presented in the following:

- **PPrASIC channels are disabled (CD)**[18]**.** This is a group of four error bits which flags that one or more PPrASIC channels have been disabled by software. Noisy, dead or hot channels have to be disabled, as they can disturb the trigger operation. The supervising software has the possibility to disable a PPrASIC channel, by setting the content of the corresponding LUT memory to zero. The effect of this measure is that the 8-bit BCID-LUT data will be permanently set to zero. Therefore, in order to avoid ambiguities when the event data is analysed, the situation has to be flagged accordingly. The ReM_FPGA needs to know in advance which channels have been disabled, because it cannot deduce the information otherwise. Two 32-bit VME registers are provided for this purpose, each bit of each register being dedicated to one PPM channel, i.e. 64 bits covering 64 PPM channels (see also section B.3.16). When event data is sent to DAQ, the RodReadoutManager copies these VME bits, and loads them to corresponding fields in each serial frame.
  The same flagging method is applied for the *unused* channels of the PPMs that process trigger-tower signals from the end-cap calorimeters. Those PPMs are supplied through the input connectors with only 32 analogue signals (see e.g. figure A.5). Correspondingly, only the output of the respective 32 channels is physically connected to the real-time path of the trigger. However, since all the PPMs used in the system are hardware identical, the other 32 channels are still connected to the readout path and provide event data. Hence, in order to avoid ambiguities, the event data originating from these PPrASIC channels is as well flagged.

- **The PPrMCM is absent from the module (MA).** In the very early commissioning phases of the L1Calo, some of the PPMs were operated with less than 16 PPrMCMs mounted on the board. Therefore, at that time this bit was meant to flag that the corresponding PPrMCM is physically absent from the PPM board. Since this scenario is excluded during the regular operation of the system in the ATLAS experiment, the definition of the MA error bit was adapted, to indicate that the PPrMCM is rather functionally than physically absent. The RodReadoutManager asserts the bit when all the four channels of the given PPrASIC are flagged as *disabled* by the supervising software, as described above.

---

[18]the text given in bold represents the definition of the error bits, as established by the L1Calo collaboration (see also [Bar08]).
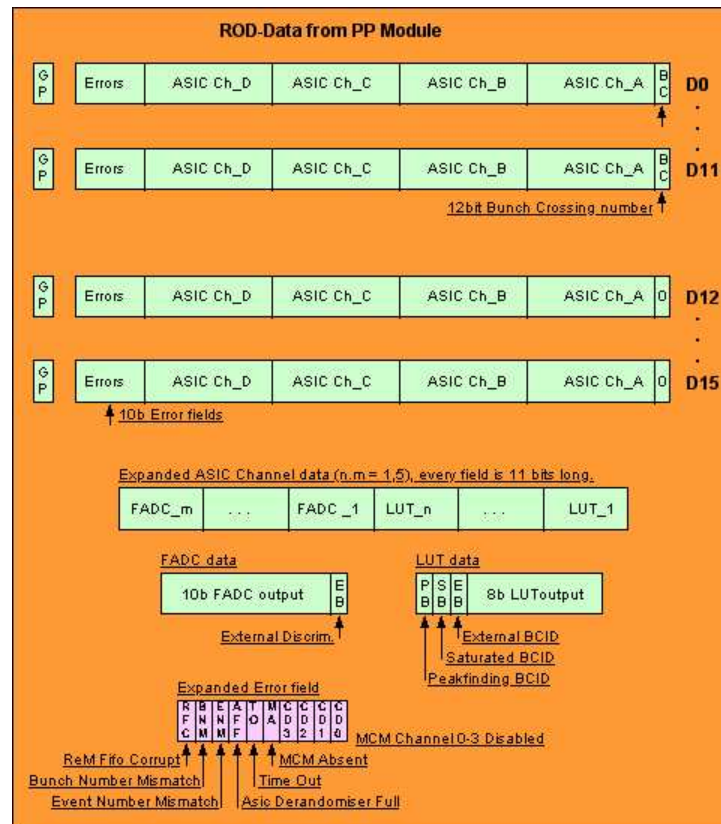
Figure 6.21: The G-Link event data format.

- **One or more channels did not produce data in response to L1A (TO).** The bit is often referred as to the *TimeOut* bit. In the RodReadoutManager, one such a bit is implemented for each PPrASIC serial interface. Since only one TO bit can be transmitted per G-Link serial frame, the bits assigned to the two serial interfaces of the same PPrASIC are merged into one bit. The default value of each TO bit is 1, indicating that no data has been received. When the first event data word is written to a dual-ported memory, the corresponding TO bit is set to 0. After the last but one event data word is stored in the memory, the bit is logically OR'ed with the corresponding bit of the second serial interface, and the result is queued into a 16-bit deep x 1-bit wide asynchronous FIFO. Then, after the last data word is written to the memory, the default value of the TO bit is restored.

- **L1As are coming too fast and the readout FIFO in the PPrASIC has become full (AFF).** When this occurs, the PPrASIC sends only event headers, and flags the respective status via the HeadersOnly and DataLoss bits. In consequence, the event data sent by the ReM_FPGA to DAQ is invalid, representing data from the previous event readout. The invalidity of the event data is flagged by the AFF error bit. The RodReadoutManager sets this bit according to the values of the HeadersOnly and DataLoss flags. As described in the previous section, the two flags of each EventHeader are logically OR'ed, and the

result is written to a corresponding local buffer. The AFF bit is then obtained by logically OR'ing the four results related to the same PPrASIC.

- **Event Number Mismatch (ENM).** When asserted, the bit indicates a mismatch between the 4-bit Event Number (EvNum) provided by the PPrASIC, and a reference computed by the ReM_FPGA. Both numbers are obtained by incrementing corresponding counters each time an L1A is received. Also, the two counters are operated with the LHC clock, and they are set to zero by the TTC Event Counter Reset (ECR) signal. As described in the previous section, the EvNum of the PPrASIC is extracted from the second EventHeader of each input data stream, and stored in a corresponding local buffer. The local reference is stored in a 16-bit deep x 4-bit wide asynchronous FIFO, upon the receipt of the L1A. When event data is sent to DAQ, the RodReadoutManager extracts the PPrASIC EvNum from two related buffers, and compares them separately with the local reference. The ENM bit is then obtained by logically OR'ing the two results.

- **Bunch Number Mismatch (BNM).** When asserted, the bit indicates a mismatch between the 4-bit BC number provided by the PPrASIC and the lower four bits of the 12-bit BC number computed by the ReM_FPGA. The comparison between the two values is performed as in the previous case. First, the PPrASIC BC numbers stored by two related buffers are compared separately with the local reference stored in the dedicated FIFO, and then the BNM bit is obtained by logically OR'ing the two results.
  The PPrASIC bunch-crossing counter is implemented in a similar way as the ReM_FPGA counter. It is incremented with each LHC clock event, and it is reset to zero by the TTC Bunch Counter Reset (BCR) signal. In spite of that, it was observed that the actual value provided by the PPrASIC does not represent the BC number of the L1A, but the next BC number. This occurs because the decision to store the output of the BC counter in the BCID-LUT derandomiser, together with the other data composing the EventHeader, is taken only one clock event after the arrival of the L1A. In order to eliminate the offset between the two counter values, the following solution was implemented in the RodReadoutManager. First, the lower four bits of the local reference are copied into a 4-bit *BcNum_Short* register, and then the obtained value is incremented by 1, to match the number provided by the PPrASIC. The usage of the BcNum_Short register is needed in order to preserve the 12-bit BC number, as this value represents the correct time stamp, and it is transfered to the ROD for verifications of the PPM timing.
  However, this solution is reliable as long as there is a minimum spacing of one clock period between the L1A and the BCR signals. When the two protocol signals are received simultaneously, the logic of the PPrASIC gives priority to the BCR signal, which sets the BC counter to zero. This value is then provided via the EventHeader, due to the above mentioned late decision, and the RodReadoutManager detects a mismatch between the counter values in spite of the incrementation. This situation was first observed while the PPr system was taking part in cosmic-rays runs, at CERN. Due to the fact that the cosmic ray particles arrive in the detector randomly with respect to the timing of the experiment, a trigger decision may be taken during any of the 3564 bunch-crossing "time marks".

As the BCR signal is generated and distributed to all readout systems only on the last bunch-crossing, i.e. 3563, there is a non-zero probability to generate simultaneously the two protocol signals. In these cases, the RodReadoutManager detects a mismatch, by first incrementing the value 3 to 4, and then comparing it to the value 0 provided by the PPrA-SIC. This situation is excluded in physics runs with LHC beam. The current LHC bunch filling scheme foresees, for a bunch spacing of 25 ns, that no collision will occur in the last 119 bunches of the orbit [Bai03]. In consequence, no L1A will be generated within the specified period of time.

Since the ReM_FPGA has to cover all the situations, the RodReadoutManager compares the BC number of the PPrASIC against four bits zero, whenever the L1A and the BCR are received on the same bunch-crossing. As the comparison between the two counter values is performed during the transmission of event data to DAQ, thus much later after the co-incidence is detected, the RodReadoutManager stores, for each event, a *coincidence* flag into a 16-bit deep x 1-bit wide asynchronous FIFO. This bit will later indicate whether the BC number of the PPrASIC has to be compared against the incremented value (the coincidence bit is set to 0), or against four bits zero (1).

A similar problem was observed for the PPrASIC event number counter, which counts the events starting from 0, when the L1A and the ECR arrive simultaneously, and from 1 otherwise. However, as the ReM_FPGA does not have to transmit the event number to the DAQ, the problem was solved by implementing the local event counter in a similar way as the PPrASIC counter.

- **ReM_FPGA FIFO corrupt occurs if FIFO overflows (RFC).** When all the data blocks of the dual-ported memories are occupied with unread event data, the RodReadoutManager stops buffering further events until the stored data is entirely read out and transmitted to DAQ. Correspondingly, as long as the restriction is in place, the RodReadoutManager asserts the RFC bit of each G-Link serial frame. Additionally, a copy of the RFC bit is provided to VME, for debugging purposes, via the first ReM_Status register (see *RodEventsNotBuffered* bit in table B.3.21).

The Global Parity Bit

While sending data to the ROD, the RodReadoutManager computes an odd-parity bit for each serial frame, and appends it after the last error bit. Upon receiving the input PPM data stream, the ROD computes a similar parity bit, and compares it with the bit provided by the ReM_FPGA, in order to detect eventual transmission errors.

6.5.5   The Transfer to RGTM-O

The RodReadoutManager starts transferring the data to the RGTM-O when at least one PPrASIC event data block is entirely stored in the local memory buffers. In order to facilitate the description of the transfer mechanism, figure 6.22 gives an overview of the operations performed in the RodReadoutManager. Additionally, figure 6.23 shows behavioural simulation results for the case of two readouts with one BCID-LUT and three FADC samples. In both figures, the descrip-

tion mainly focuses on the case of the first two serial input data streams, as the treatment of the other input streams is identical.

In the RodReadoutManager, each input PPrASIC data stream is received by a functionally identical *RodBuffer* module. The main task of this module is to extract the event data from input stream, and store it in a corresponding local buffer. Upon detecting the first EventHeader, the module writes to the buffer each following BCID-LUT and FADC data word, until their sum equals the double sum of the settings specified from the VME (*NumFadc*, *NumBcid*). Additionally, the RodBuffer module asserts a *WriteInProgress* bit, to flag the status of the write operation, and sets a *time-out* (TO) error bit to zero, to indicate that the corresponding serial interface has provided data in response to the L1A. The WriteInProgress flag is asserted upon writing the first BCID-LUT data word, and cleared when the last but one FADC data word is stored. This status bit is then routed to a *WriteControl* module, where it is logically OR'ed in two successive steps with the other 31 similar bits, to form a global indicator of the write operation to the 32 local buffers (see *ROB_WIP* in figure 6.23). The high-to-low transition of the obtained signal, i.e. write operation completed in all RodBuffer modules, is used by the WriteControl for three purposes.

The first purpose is to increment a 4-bit *NextBlockToWrite* counter, which indicates to each RodBuffer module the memory block where the next PPrASIC event data has to be stored. The counter is incremented by 1, if the sum of NumFadc and NumBcid is smaller or equal to 8, and by 2, if the same sum is greater than 8 and smaller or equal to 16 (see again table 6.6). Upon detecting a new event block in the input data stream, each RodBuffer module sets the local write pointer to the start address of the next memory block, by appending four bits zero to the 4-bit value provided by the NextBlockToWrite. The usage of a block counter is mainly necessary for the case when one or more serial interfaces do not provide data during a single or more consecutive readout operations. As long as this situation holds, the corresponding local write pointers are not incremented. Thus, in the eventuality that the same serial interfaces start sending again data, after a number of event readouts, the corresponding local write pointers have to be re-aligned with the other write pointers, to ensure that all 32 event data blocks are stored at the same locations, in each memory buffer.

The second purpose, for which the high-to-low transition of the ROB_WIP signal is used, is to store the *time-out* bits in the allocated FIFO (see *WriteEnable_TOFifo*). The third and the last purpose is to generate a *WriteDone* strobe, which indicates the availability of the event data in the local buffers. The WriteDone strobe and the Empty status flag of the "Time Out" FIFO (*Empty_TOFifo*) are the signals that initiate the transfer of event data to the RGTM-O.

The data transfer operation is coordinated by a *ReadControl* module. If the memory buffers have previously contained no data, and a new PPrASIC event block has just been written to the buffers, the ReadControl starts retrieving the data from the FIFOs and the dual-ported memories upon the receipt of the WriteDone signal. At first, the ReadControl generates a *GetFifoData* strobe, to extract the bunch-crossing number, the event number and the coincidence flag from the FIFOs. The lower four bits of the 12-bit BC number are copied in a separate register, and processed in order to cancel the offset between the related PPrASIC and ReM_FPGA counters (*BCNum_ShortCounter*), while all the 12 bits of the FIFO output (*BCNum_FifoOut*) are transferred via the first 12 data links. The loading of these bits in the serial output stream, as well as the loading of all the other data types, is controlled by several strobes, which are issued based on

the value of an internal bit counter, and of the NumFadc and NumBcid settings. The first strobe, *BcBit_RdStr*, will simultaneously activate the "data available" frame signal (*DAV_BAR*[19]), and load the BC bits in the same streams. The next two strobes determine the loading of the event data from the dual-ported memories. As for the write operation, a 4-bit *NextBlockToRead* indicates the start address of the memory block that has to be read out. Unlike the write pointers, which are locally manipulated during the writing of data to the buffers, the read pointers are globally controlled, to ensure that data is read out simultaneously from each memory, and only from the location indicated by the global pointer (*ReadAddress*). First, the ReadControl extracts and transfers the event data provided by the first serial interface (*SerIntf1_RdStr*), and then the event data of the second serial interface (*SerIntf2_RdStr*).

The error bits are transferred in two steps. The first strobe, *ErrBits1_RdStr*, will determine the insertion of the CD and MA bits in the serial stream. In the same time, the signal determines the extraction of the TO bit from the FIFO, and the extraction and processing of the PPrASIC bunch-crossing number, event number and the header flags stored in the dual-ported memories. The TO and RFC bits, and the AFF, ENM and BNM bits returned by the respective processing are inserted in the output stream upon the release of the *ErrBits2_RdStr* strobe. After that, the ReadControl issues a *Parity_RdStr* strobe, which appends the global parity bit to the output stream, and determines the deactivation the DAV_BAR signal on the following clock event.

The transfer of the entire serial stream to the RGTM-O necessitates longer time than the operation of writing the input PPrASIC data to the buffers. For a readout with one BCID-LUT sample and three FADC samples, which is the case illustrated in figure 6.23, it takes 3.25 μs to buffer the input data, and 4.7 μs to read it out, pack it with additional data, and transfer it. Which means that if a second L1A is received in, roughly, less than 4.7 μs, the corresponding event data block will be available in the memory buffers before the current transmission is completed. Subsequently, the second WriteDone strobe will be generated at a time when the ReadControl module is busy transferring data, and thus it cannot process it. In consequence, always after completing a transfer, the ReadControl checks the level of occupancy in the memory buffers, by reading the Empty_TOFifo status flag. If this is set to zero, then the ReadControl initiates a new data transfer. This is a reliable solution, because the TO bits are written to the FIFO shortly before the buffering of the input data is completed, and read out also shortly before the transfer to RGTM-O is finalised. Additionally, the release of the WriteDone signal is always delayed with six clock periods, with respect to the WriteEnable_TOFifo, to allow the ReadControl to either detect a non-empty FIFO or the WriteDone strobe.

The evaluation of the Empty_TOFifo flag, and the preparation for transmission of the next bunch-crossing number, require together six clock periods. This means that the data transfer, and the activation of the DAV_BAR signal, will only occur after this delay. For debugging or testing purposes, the ReadControl can be instructed from VME to extend the delay between two consecutive "data available" signals (*DAV_Gap*), with up to 15 clock ticks (see section B.3.15). This setting will also delay the data transfer, with the same number of clock periods. The eventual extension is performed after the evaluation of the Empty flag, and only if the flag indicates that event data is available in the buffers. This is done in order to allow the algorithms of the ReadControl module to return in an operational state in which they can process a further WriteDone

---

[19] the DAV* signal described in section 6.2.4
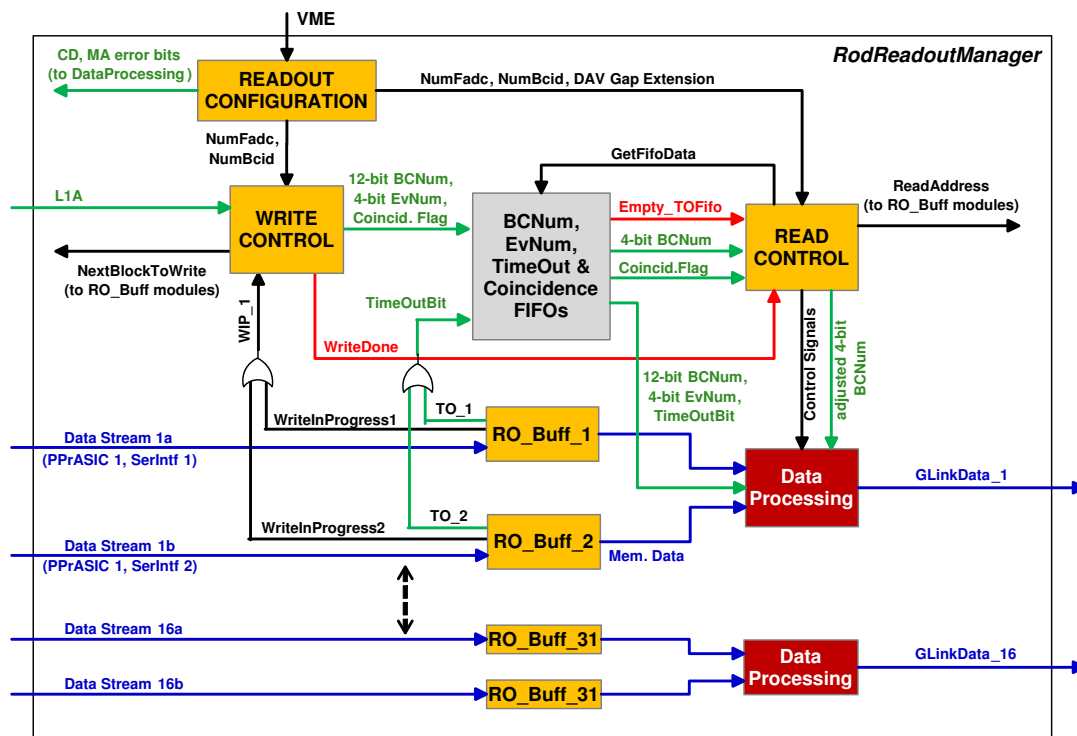
Figure 6.22: The event data collection, processing and transfer to RGTM-O, as performed in the Ro-dReadoutManager.

strobe. If the flag indicates empty buffers, then an extension of the DAV_Gap with more than 6 clock periods, i.e. the delay between the WriteEnable_TOFifo and WriteDone signals, can cause the ReadControl to miss an eventual WriteDone strobe. Which will postpone the data transfer until a second WriteDone strobe is received.

Last but not least, the *Achilles' heel* of the current implementation is the scenario in which one or up to 31 serial interfaces provide only event headers. This operational mode indicates that either one or both derandomisers of the given interfaces have become full, and that they cannot buffer any further event data from the corresponding pipeline memories. In this case, the WriteInProgress pulses, generated by the corresponding RodBuffer modules, will be significantly shorter, and they will irreversibly affect the mechanism that initiates the transmission of the actual event data to the RGTM-O. Such a scenario, with less than 32 serial interfaces entering this operational mode, has never been so far observed. As all the PPrASICs are configured to copy the same amount of data from the pipeline memories to the derandomisers, one expects the same operational behaviour in all 16 PPrASICs. A different situation will then point to a defective hardware. However, this very unlikely scenario has to be covered by the ReM_FPGA's algorithms. The solution foreseen for the next updates of the firmware design is to permanently de-couple the corresponding WriteInProgress signals from the generation of the ROB_WIP signal, and assert accordingly the AFF error bit, until the system is reset.
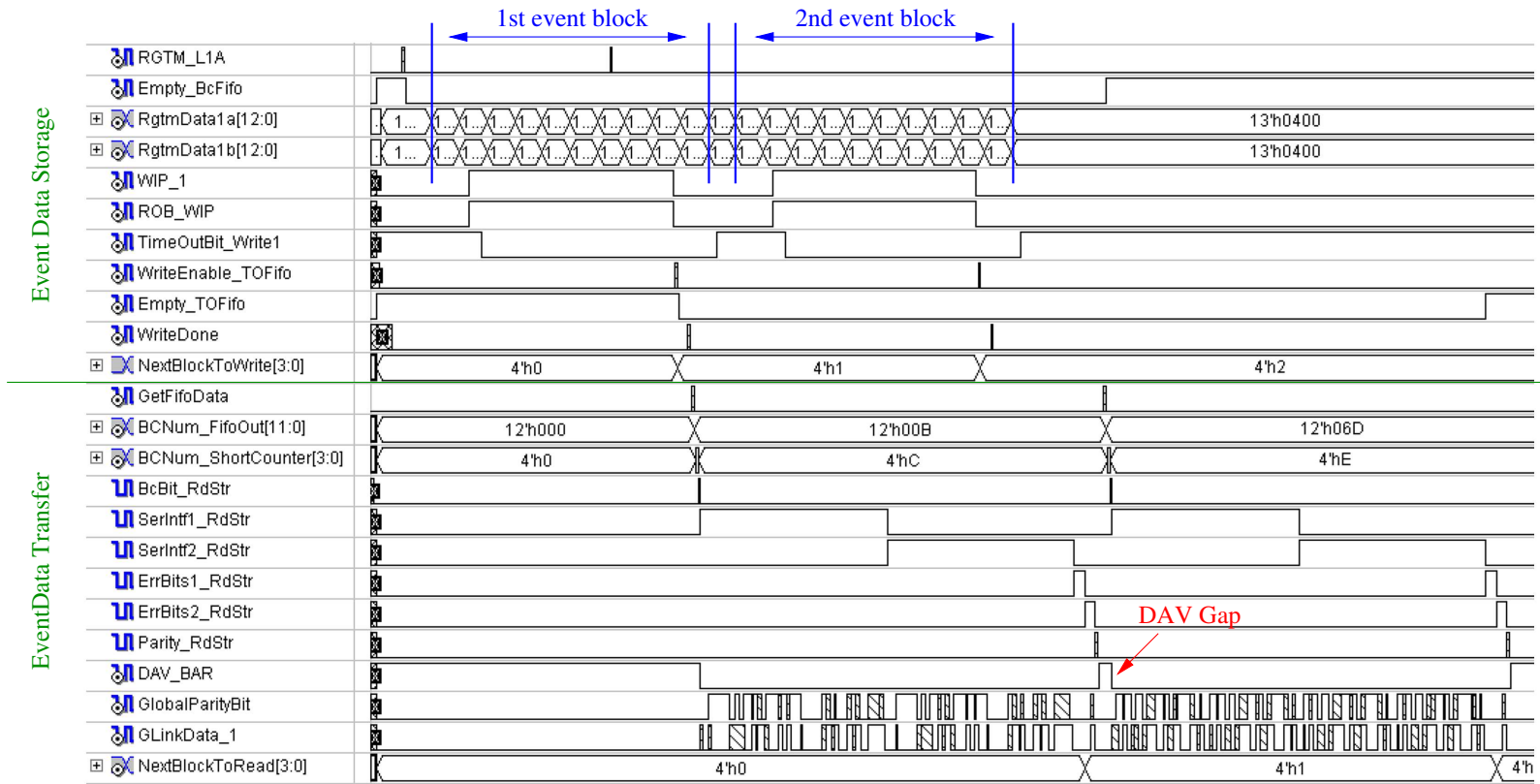
Figure 6.23: Behaviour simulation of the RodReadoutManager module, for the case of two readouts with one BCID and three FADC readout samples.
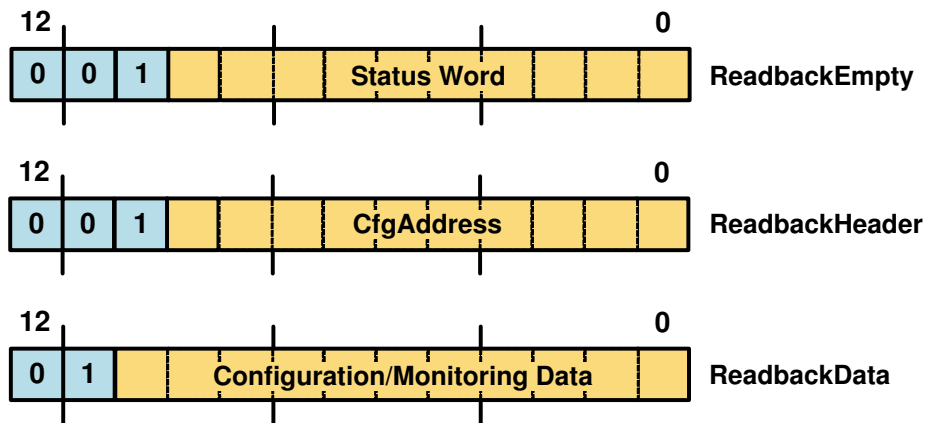
Figure 6.24:   The format of the PPrASIC readback data on the serial interface (adapted from [Hus02].

6.6   Collecting the PPrASIC Readback Data

Another major task assigned to the ReM_FPGA is to collect the PPrASIC readback data and store it in locations accessible over the VME. As previously described in section 6.5.1, the PPrASIC provides three types of data via the *Readback Channel*: configuration, monitoring and status data. The configuration and the monitoring data are provided on request, while the status data is sent permanently when no request for the previous data types is addressed.

The configuration data provides the value stored at an indicated PPrASIC register or memory location. The ReM_FPGA write this data to a corresponding SRAM location, in the MCM Readback block, from where it can be read out over the VME, and compared with the similar setting loaded by the software and the copy stored in the MCM Reference block, in order to verify the setting up procedure and the proper functioning of the hardware. The monitoring data consists of channel-wise and trigger unbiased energy rates and spectra, which are produced based on *real-time* 10-bit raw FADC data or 8-bit BCID-LUT results. This data is read out periodically and used for detecting faulty calorimeter trigger channels (see section 8.1). Additionally, the same data can be used for calibrating the trigger-tower energies or for monitoring the LHC beam conditions [Mül08b]. The rates and the histograms are provided over the serial interface upon separate requests. The ReM_FPGA assembles each data type into a dedicated block in the SRAM. Finally, the status data provides bit-wise information about the state of various processes in the PPrASIC. This data is packed by the ReM_FPGA into VME-accessible registers.

6.6.1   The PPrASIC Readback Data Format On the Serial Interface

Figure 6.24 shows the format in which the PPrASIC transfers the readback data over the serial interface. There are three types of 13-bit readback words, each type being flagged correspondingly via the most significant bits. The first word, called *ReadbackEmpty*, provides a 10-bit status data, of which content is listed in table 6.7. The first four bits indicate the availability of the monitoring data in the two channels served by the accessed serial interface. The following

Bit Nr.	Description
0	Rate Available (Channel 1)
1	Rate Available (Channel 2)
2	Histogram Available (Channel 1)
3	Histogram Available (Channel 2)
4	FADC Pipeline Stopped (Channel 1)
5	BCID-LUT Pipeline Stopped (Channel 1)
6	FADC Pipeline Stopped (Channel 2)
7	BCID-LUT Pipeline Stopped (Channel 2)
8	Playback Mode Active
9	Frequency Lost (PPrPHOS4 status)

Table 6.7: The PPrASIC status data provided via the ReadbackEmpty data word.

four bits indicate that the writing of the raw FADC and BCID-LUT data to the pipeline memories has been stopped by the user [20]. The last but one bit flags that at least one of the two channels have been configured to operate in playback mode, while the last bit provides a copy of the FrequencyLost signal described in section 6.2.7. In the ReM_FPGA, these ten bits are extracted by each AsicSerialInterfaces module, and provided to the StatusRegister module, which packs them separately into VME-accessible registers (see figure 6.19 and sections B.3.8 to B.3.13).

The other two 13-bit data words form the actual readback block. The *ReadbackHeader* word starts the readback block, and indicates the PPrASIC location which is being read out, while the *ReadbackData* word provides an 11-bit configuration or monitoring data. In the ReM_FPGA, the ReadbackHeader and the control bits of the ReadbackData are discarded, and the remaining 11-bit data is written to a corresponding location in the SRAM. The following subsections present the algorithms implemented in the ReM_FPGA's firmware for retrieving the configuration and the monitoring data from the PPrASICs.

6.6.2 Readback of Configuration Data

As mentioned in section 6.4.1, when configuration data is transferred from VME to a PPrASIC location, the ReM_FPGA places one copy of the input 32-bit VME data in the MCM Readback block, and sets the upper four bits of the corresponding SRAM location to value 4'b0001, to flag the change in configuration. Additionally, the upper four bits, called in the following as the *readback flags*, are set by the ReM_FPGA to the same value when a VME_Reset signal is received. The reason for doing this is that the VME_Reset removes any configuration data

---

[20]there are two distinct cases in which the writing of event data to the pipeline memories in stopped. In the first case, the user explicitly requests the interruption, via a specific command addressed over the serial interface. In the second case, the same state is indirectly induced. The user has the possibility to read out the whole content of any pipeline memory via the serial interface. While this operation is in progress, the writing of event data to the accessed pipeline memory is paused. None of these two cases is considered by the current design of the ReM_FPGA's firmware.

previously loaded to the PPrASIC registers, by restoring the default values. In consequence, immediately after the reset state is released, the ReM_FPGA sets the lower 32 bits of the entire MCM Readback block, for convenience, to zero, and the corresponding readback flags to value 4'b0001. When configuration data is read back from a PPrASIC location and placed in the SRAM, the ReM_FPGA sets the readback flags of the corresponding SRAM location to value 4'b0000, to indicate that the respective data was retrieved from the source.

The readback of PPrASIC configuration data is initiated from the VME. Two methods are provided for the current purpose. In the first method, the ReM_FPGA starts automatically the readback process, when the readback flags associated with a PPrASIC location are read out over the VME, and if their value is set to 4'b0001. In the second method, the ReM_FPGA can be instructed to ignore the value of the same flags, and start the readback process upon a similar VME read operation to these bits.

Conditioned Readback

The upper four bits of the entire MCM Readback block are mapped to VME as described in section B.3.2. When the flags are read out from VME, the ReM_FPGA checks first their value. If this is found to be set to 4'b0001, then the ReM_FPGA proceeds with verifying if the respective flags are associated with a PPrASIC register or memory location. This verification is necessary for couple of reasons. First of all, multiple locations in the MCM Readback block are allocated for storing PPrAnIn-DAC and PPrPHOS4 data (see table B.2). Although these devices cannot be read back, the ReM_FPGA places a copy of the related configuration data in these locations, and sets the corresponding readback flags to value 4'b0001, due to the identical format of the MCM Reference and Readback blocks. Then, a significant number of memory locations in the same blocks are not at all used. The VME addresses related to the respective SRAM locations do not encode a valid PPrASIC register or memory location. Although no input configuration data is written to these locations, the upper four bits set also by the ReM_FPGA to value 4'b0001, when a VME_Reset signal is received. In consequence, the ReM_FPGA must verify that the read-flags accessed by the VME are not associated with any of the locations mentioned above. If the verification returns a negative result, then the ReM_FPGA only provides the value of the respective flags to VME, and does not consider initiating a readback operation. In case of a positive result, then the ReM_FPGA starts reading back the related data.

The exact information about the PPrASIC location that has to be read out is obtained from the input VME address, in the same manner as described in section 6.4.1. Based on this information, the ReM_FPGA builds also a 13-bit *StartReadback* command word, which indicates to the identified PPrASIC a readback request and the location from where the data has to be retrieved (see also [Hus02]). The number of ReadbackData words, sent by the PPrASIC in response to the StartReadback command, depends on the type of requested data. If a register is read out, then the PPrASIC provides the respective 11-bit data via one ReadbackData word. If memory data is requested, then the PPrASIC transfers the whole memory content, and not the data stored in a single memory location. Thus, in this case, the number of ReadbackData words equals the depth of the accessed memory. Upon receiving the readback data from the PPrASIC, the ReM_FPGA extracts the 11-bit configuration data, and writes it to a corresponding location in the MCM Readback block, in the format indicated in table B.2. Simultaneously, the ReM_FPGA sets the
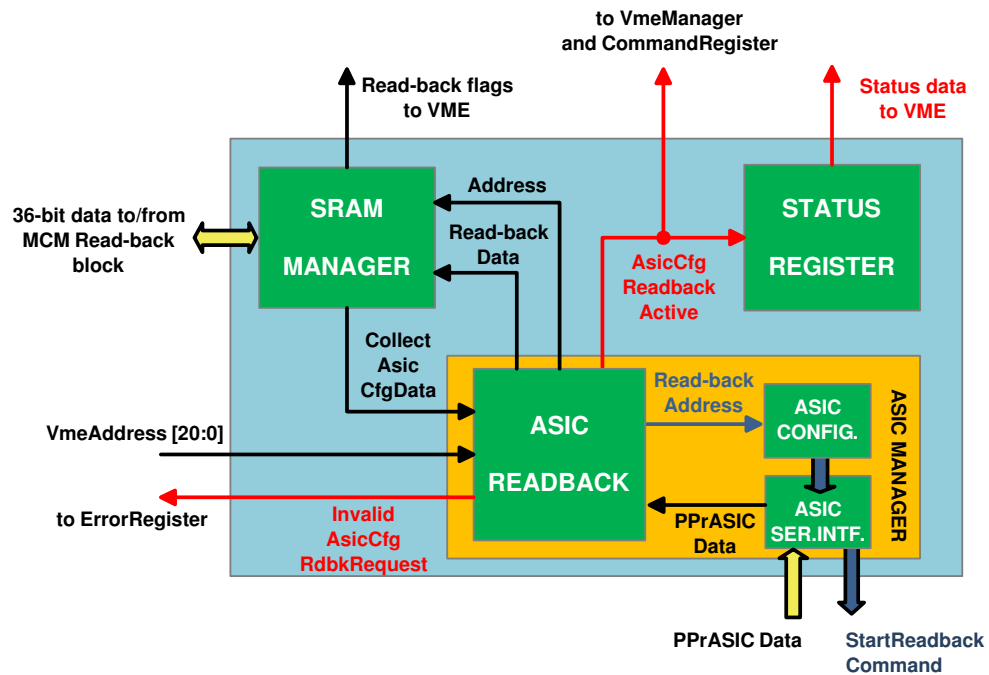
Figure 6.25: Control logic in the ReM_FPGA managing the readback of configuration data from the PPrASICs.

readback flags to value 4'b0000, to indicate that the data stored in the respective location was read back from the PPrASIC.

Figure 6.25 describes schematically the logic implemented in the ReM_FPGA's firmware for retrieving the configuration data from the PPrASICs, and transferring it to the SRAM. The value of the readback flags is checked by the SramManager. If this is set to 4'b0001, the SramManager distributes a *CollectAsicCfgData* command to another functional module, called *AsicReadback*, which handles all the readback operations from the PPrASICs. Upon receiving this signal, the AsicReadback latches the input VME address and uses it for three purposes. The first one is to verify if the readback flags are associated with a PPrASIC location. If the verification returns a negative result, then the AsicReadback cancels any further processing of the readback request, and flags the situation via the *InvalidAsicCfgRdbkRequest* bit of the first ReM_Error register. The second purpose is to generate a *Readback Address*, which is provided to the AsicConfiguration module. Based on this information, the latter module identifies the PPrASIC and the serial interface that have to be accessed, and builds the 13-bit StartReadback command. The third purpose is to obtain the address of the SRAM location where the PPrASIC readback data will be stored.

A copy of the input data stream from the accessed serial interface is routed to the AsicReadback. The module monitors permanently the two control bits of each incoming 13-bit data to identify the ReadbackData word. When this occurs, the module discards the control bits and writes the remaining 11-bit configuration data to the SRAM, via the SramManager. Addition-

ally, the AsicReadback requests the SramManager to set the corresponding readback flags to value 4'b0000.

The status of the readback operation is flagged by the same module. An *AsicCfgReadback-Active* bit is asserted as soon as the readback request is validated, and cleared immediately after the last configuration data word is placed in the SRAM. The status bit is permanently provided to three functional modules: VmeManager, CommandRegister and StatusRegister. When the bit is set, the VmeManager and CommandRegister modules deny any VME request that implies the access to the SRAM. The StatusRegister provides the respective bit to VME, via the first ReM_Status register.

Unrestricted Readback

For debugging and test purposes, the ReM_FPGA can be instructed to initiate the readback process independent of the value stored by the readback flags. This operational mode is activated via the *ForcedConfigReadback* bit of the ReM_Control register (see table B.24). When the bit is asserted, the SramManager ignores the value of the readback flags, and generates the CollectAsicCfgData command upon receiving the data from the SRAM device. Once the readback process is started, the logic administrating the ReM_Control register clears the ForcedConfigReadback bit. Which means that the bit has to be asserted again, if the next readback operation is intended to be realised under the same conditions.

6.6.3   Readout of PPrASIC Energy Rates and Spectra

In the PPrASIC, the monitoring functionality is implemented in two modules, called *Rate Metering* and *Histogramming*. The Rate Metering evaluates the input FADC or BCID-LUT values against a programmable energy threshold, during a specified period of time, while the Histogramming maps the same input values into histogram bins. In both cases, the FADC and the BCID-LUT data are permanently fed from the real-time pre-processing path (see again figure 6.1). Which means that the data produced by the two algorithms is not biased by the L1 trigger decision.

The PPrASIC Rate Metering

Figure 6.26 describes schematically the algorithm implemented in the Rate Metering module. The operation is started as soon as the *RateEnable* bit is asserted. A 20-bit *DataCounter* is then incremented each time the input 10-bit FADC or the 8-bit BCID-LUT data exceeds the specified threshold value (*RateEtThreshold*). The selection of the input data source, as well as the setting up of the programmable parameters, is done during the configuration procedure, via appropriate PPrASIC channel registers. The energy threshold is 10 bits wide, in order to cover the entire dynamic range of the FADC. In case the BCID-LUT data is selected as source, two bits zero are appended to each incoming 8-bit BCID-LUT data, so that the obtained data word matches the width of the threshold. The period of time, within which the input data is compared against the threshold, is specified in units of 1024 bunch-crossings, i.e. 25.6 μs. This is achieved by
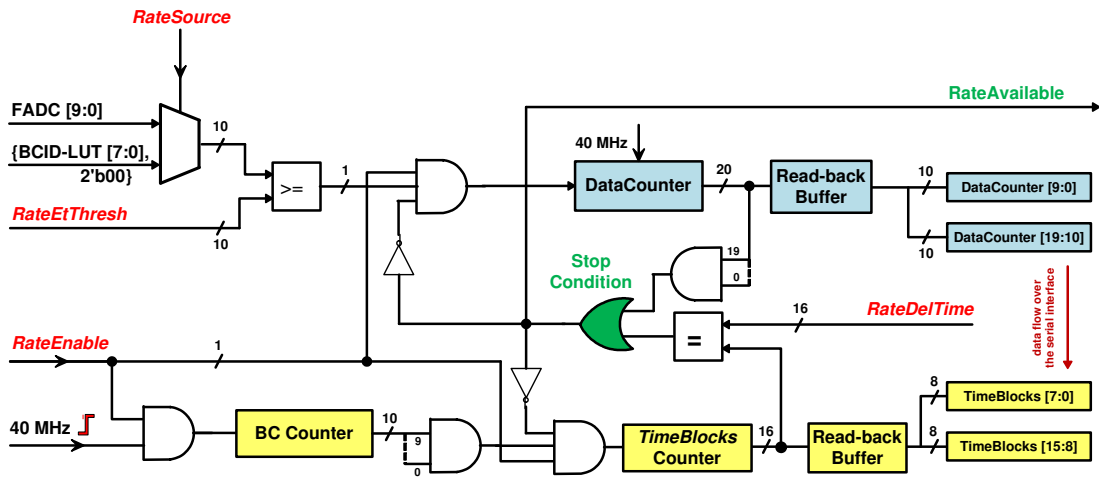
Figure 6.26: Schematic representation of the PPRASIC Rate Metering operation. The parameters given with italic fonts are user-programmable.

combining a 10-bit *BC Counter*, which is incremented with each LHC clock event, and a 16-bit *TimeBlocks* counter, which is incremented each time the BC Counter overflows.

The Rate Metering operation stops when either the DataCounter overflows, or when the value stored in the TimeBlocks register equals the value specified by the user (*RateDelTime*). When any of these conditions is met, the content of the DataCounter and TimeBlocks registers is copied into dedicated readback register buffers. Simultaneously, a *RateAvailable* flag is asserted, to indicate the completion of the operation, and that the data is available for readout. This bit is mapped by the logic of the serial interface in the ReadbackEmpty word, as previously shown in table 6.7, and cleared upon a readout requests for the Rate Metering data. After the data is copied in the readback buffers, the content of the DataCounter and TimeBlocks registers is cleared, and the Rate Metering operation restarts automatically, if the RateEnable bit is still asserted. This also means that if the readback buffers are not read out before the stop condition is met again, the new results will overwrite the previous ones. When a readout is requested, the 20-bit counting result and the 16-bit time information are extracted from the readback buffers, and sent over the serial interface in the format and the succession indicated in figure 6.26 [Hus02].

The PPrASIC Histogramming

The algorithms implemented in the Histogramming module are presented schematically in figure 6.27. The module maps the input FADC or BCID-LUT values, that exceed a programmable threshold, into histogram bins, in order to obtain a distribution of the energy depositions in the given trigger tower. The histogram is realised by means of memory locations, the same 8-bit deep x 11-bit wide memory used for storing the playback data being also used for the current purpose. Thus, the histogram has 256 bins, each of which allows a maximum of 2047 counts. The energy range and the resolution of the histogram depend on the selected input data source. If the BCID-LUT is used, then these are set to 256 GeV and 1 GeV respectively. In case the
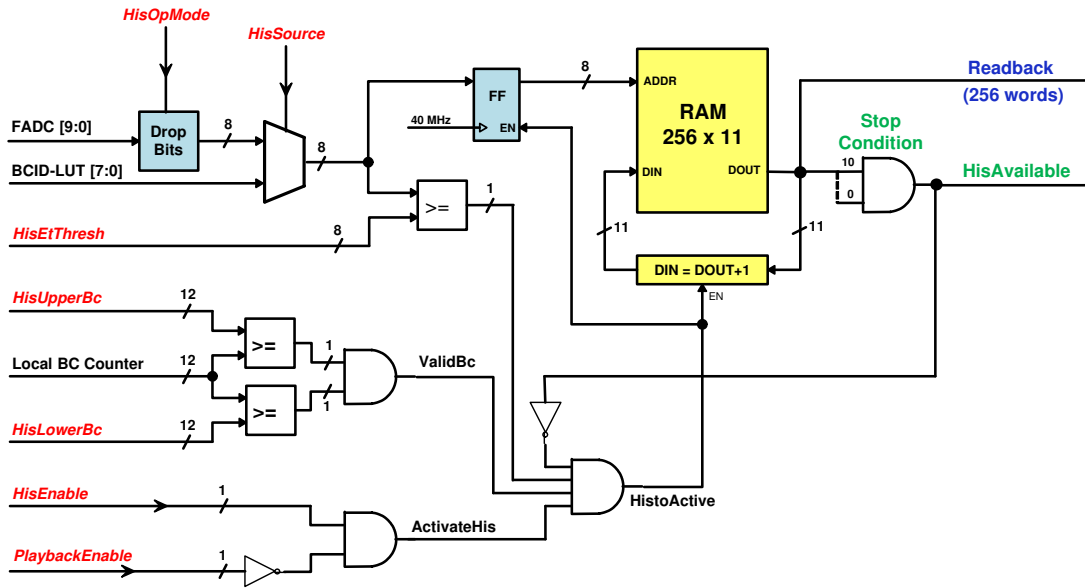
Figure 6.27: Schematic representation of the PPRASIC Histogramming operation. The parameters given with italic fonts are user-programmable.

FADC data is selected, then two bits of the 10-bit input data are dropped, according to the value of a user-programmable parameter (*HisOpMode*). Three truncation modes are available, each one determining a specific energy range and resolution for the histogram, as shown in table 6.8.

As the Histogramming and the Playback processes share the same memory, the former is activated only when the latter is disabled (*ActivateHis*). Additionally, the Histogramming operation can be restricted to a certain range of bunch-crossings. The upper and lower limits of this range are defined by two user-programmable parameters (*HisUpperBc*, *HisLowerBc*). The Histogramming module compares these values with the output of the local 12-bit bunch-crossing counter, and enables the histogramming of the input data if the latter value is within the specified limits (*ValidBc*)[21].

The input BCID data or the truncated FADC data are permanently compared against an 8-bit

---

[21]this mode is disabled when the limits of the bunch-crossing range are set to the same values. In consequence, the ValidBc will be permanently asserted.

HisOpMode	Dropped Bits	Range (GeV)	Resolution (GeV)
0	**2 LSBs**	**0 - 256**	**1**
1	1 MSB, 1 LSB	0 - 128	0.5
2	2 MSBs	0 - 64	0.25

Table 6.8: The histogramming modes of the 10-bit FADC data. The values given in bold specify the default mode.

energy threshold. When the Histogramming operation is activated, the content of the memory location indicated by the accepted input data is first read out and evaluated. If not saturated, then the respective value is first incremented and then written back to its location[22]. If saturated, then the Histogramming operation stops, and the situation is flagged via the *HisAvailable* bit. Unlike the Rate Metering data, which is copied to dedicated readback buffers, the Histogramming data is kept in the memory until it is read out over the serial interface. When this is requested, the PPrASIC transfers the 11-bit content of each memory location via one ReadbackData word, and simultaneously de-asserts the HisAvailable flag. When the readback is completed, the Histogramming module first clears the memory content, and then restarts the operation. [Hus02].

Readout Operations in the ReM_FPGA

The ReM_FPGA retrieves the Rate Metering and Histogramming data during separate readout operations. In both cases, the operations are initiated from VME, via corresponding bits in the ReM_Command register (see table B.25). Upon receiving such a request, the ReM_FPGA retrieves the indicated monitoring data type from all 16 PPrASICs, and stores it in dedicated Readback blocks in the SRAM. The structure of these blocks, and the format in which each data type is stored, is described in sections B.2.4 and B.2.5. Note that the Rate Metering data is reassembled to its original width, after it was split in the PPrASICs, for serial transmission to the ReM_FPGA.

In the ReM_FPGA, the readout operation is coordinated by the same AsicReadback. The module receives from CommandRegister a strobe signal that indicates the readout operation and the type of data that has to be collected. In its turn, AsicReadback generates appropriate addresses for the AsicConfiguration module, that indicate the same readout operation, the data type and the location from where the data has to be retrieved. Based on this information, the AsicConfiguration generates appropriate StartReadback commands for each PPrASIC. The retrieval of monitoring data is performed channel-by-channel, starting with the first channel of the PPrASIC located on the upper PPrMCM, and ending with the last channel of the PPrASIC located on the lower PPrMCM. At each step, AsicReadback writes first the received data to the SRAM before issuing a new address for the AsicConfiguration module. While the transfer is in progress, the AsicReadback asserts an appropriate flag, indicating the status of the operation. The flag is cleared as soon as the last data word is written to the SRAM. As in the similar cases described in the previous sections, the flag is provided to the same functional modules, i.e. VmeManager, CommandRegister and StatusRegister, for the same two purposes: to deny further VME requests that imply an access to the SRAM, and to provide the status of the operation to the VME.

The consequence of reading out the Histogramming data channel-by-channel, is that the previous set synchronisation, of the respective operation in all 64 PPM channels, is lost. This is because, as mentioned in the previous subsection, the Histogramming operation restarts only after the data is read out from the memory. Thus, if the synchronisation is still wanted, the Histogramming operation has to be first disabled and then re-enabled, after the readout from all 64 channels is completed. The Rate Metering is not affected by the readout implementation.

---

[22]the entire operation requires ten clock ticks to be completed. During this time no other input data can be histogrammed.

This is because the Rate Metering operation restarts immediately after the data is copied in the readback buffers. As the latter operation is performed independently on the ReM_FPGA's request, the synchronisation of the Rate Metering operations in all 64 channels is conserved.

6.7   Readback of TTCrx Configuration Data

The ReM_FPGA has also the task of reading back the TTCrx register data. The concept implemented for this operation is identical to the one implemented for retrieving the PPrASIC register and memory data. The readback flags are set to value 4'b0001, when configuration data is transferred from VME to TTCrx, or when a VME_Reset signal is received[23]. Then, when the flags are read out over the VME, the readback operation is initiated if either their value is set to 4'b0001 or the ForcedConfigReadback bit is asserted.

The transfer of data from the TTCrx to the SRAM is managed by the same three functional modules that realise the transfer of configuration data from VME to TTCrx: Phos4TTCrxData, I2CDataTransfer and I2CMasterCore (see again figure 6.8). The Phos4TTCrxData module is the actual coordinator of the readback operation. Upon receiving a *CollectTTCrxCfgData* command from the SramManager, the module latches the input VME address in order to validate the readback request, to determine the TTCrx register number and generate the corresponding I2C_pointer address, and to obtain the SRAM address where the data read back from the TTCrx has to be placed. When the 8-bit register data is received from the TTCrx, the Phos4TTCrxData module writes it to SRAM, via the SramManager, and requests the same module to set the readback flags to value 4'b0000. Also, the Phos4TTCrxData is the module that flags the progress of the readback operation. The status bit, i.e. *TTCrxCfgReadbackActive*, is permanently provided to the VmeManager, CommandRegister and StatusRegister modules, to deny further VME requests that imply the access to SRAM, and to provide the status of the data transfer to the VME.

6.8   Spying the PPrASIC Serial Interface Data Over the VME

When the PPM is operated in standalone mode, on a test platform that does not include DAQ facilities, the ReM_FPGA can be instructed to store a copy of the PPrASIC event data into VME-accessible buffers, to allow the testing or debugging of the module. When this is requested, the ReM_FPGA extracts from the input stream of each serial interface the first arriving event data block, and writes it without any additional processing to a corresponding 9-bit deep x 16-bit wide internal memory buffer. The depth of the buffers was specially chosen, to allow even the storage of the largest possible PPrASIC event block, i.e. to 127 raw FADC and 7 BCID-LUT samples for each PPrASIC channel.

Also for testing or debugging purposes, the ReM_FPGA can be instructed to store additional data from the input PPrASIC data stream. Two methods are provided. In the first one, the ReM_FPGA stores in the same memory buffers the first arriving event block, and all the 13-bit

---

[23]as mentioned in section 6.2.7, a VME_Reset is always required after a TTCrx_Reset, in order to ensure that the ReM_FPGA's logic recovers from an eventual clock glitch. Thus, for this reason and in order to also minimise the usage of internal FPGA resources, the TTCrx readback flags are set upon a VME_Reset signal, as the PPrASIC readback flags, and not upon a TTCrx_Reset signal.

data words that follow the respective event block, until the buffers get full. This method gives the possibility to store multiple event blocks, of small and medium size, and in the same time to analyse the content of the readback words sent in between two consecutive event blocks. In the second method, the ReM_FPGA stores all the data words received over the serial interface, until the memory buffers saturate.

The operational modes described above are enabled via dedicated bits in the ReM_Control register (see table B.24). Also, the modality in which the accumulated data can be accessed from the VME is presented in section B.3.1.

6.9 Design Implementation

As mentioned in the introductory part of the chapter, the ReM_FPGA is a Xilinx XCV1000-E device, and its behaviour was designed using the Verilog HDL. The created source code was embedded in the FPGA using the design tool provided by the vendor[24].

Table 6.9 lists the most relevant types of FPGA resources used by the current firmware design, and the corresponding utilisation factor. The basic functional block in the VirtexE FPGA is the logic cell. Each logic cell mainly consists of a 4-input LUT, used as combinational function generator, a flip-flop, to optionally register the LUT output, and carry logic, for efficient arithmetic operations. Two such cells are combined into one logic slice, which represents the basic programmable block, and two slices form together a Combined Logic Block (CLB) [Xil06]. The current design of the ReM_FPGA's firmware uses three quarters of the available slices. However, the relevant number for estimating the utilisation level is the number of used slice LUTs, as the logic is mostly implemented by them. When the design is not hitting the limits of the available resources, the implementation tool takes the freedom to use as many as needed slices in order to minimise the routing delays. In consequence, some of the slices counted as *occupied* have only one LUT element implemented. The large number of used flip-flops, approaching the number of used LUTs, indicates that the design is mostly based on sequential logic.

The VirtexE device also incorporates 96 4 kb configurable Block RAM (BRAM) memories.

---

[24]Xilinx Integrated Software Environment (ISE), v10.1

Resources	Used	Available	Utilisation
Number of occupied Slices	9,416	12,288	76%
Number of Slice Flip-Flops	10,517	24,576	42%
Number of 4-input LUTs	12,823	24,576	52%
Number of Block RAMs	96	96	100%
Number of GCLKs	3	4	75%
Number of GCLKIOBs	2	4	50%
Number of DLLs	5	8	62%

Table 6.9: Usage of FPGA resources.

These elements are used up in the current design. A number of 32 BRAMs are used for buffering the PPrASIC event data, while the other 64 BRAMs are used for spying the PPrASIC serial interfaces. As a consequence of the full utilisation of BRAMs, the asynchronous FIFOs of the RodReadoutManager module were built from the slice LUTs, each such element acting like a 16 x 1-bit RAM. Also listed in table 6.9, are the resources used by the clock management scheme. Two global clock I/O buffers (GCLKIOB) receive the PPM_XTAL and the TTCdec clocks, and three global clock lines (GCLK) distribute with low skew the SysClk, intGLinkClk and intSRamClk clocks, output by the management scheme, to all the flip-flops and memory blocks.

Functional tests of the Readout Manager FPGA are presented throughout the next chapter.

Chapter 7

The Functional Tests of the PreProcessor Module

Before the PPMs were installed in the electronics cavern of the ATLAS experiment, their proper operation had been tested in the laboratory environment, in Heidelberg. An extensive test procedure was developed to establish the functions of the PPMs in short and long periods of operation. The modules were tested both individually as well as in a crate configuration similar to that of the system in use at CERN. The pre-processing of the input analogue signals, the readout operation and the transmission of the real-time data over long LVDS cables were checked with a dedicated VME-based system, which emulates both a ROD module and the receiving stage of the L1Calo processors. Additionally, a periodic monitoring of the temperatures and voltages across each board was performed during the tests, to verify the operating conditions of the modules.

This chapter presents in details the procedure implemented for testing the functionality of the PPMs.

7.1 Overview

A total number of 160 PPM mainboards were produced, in order to enable a consistent batch of spare modules above the number required by the full coverage of the experiment, i.e. 124 PPMs. Each manufactured mainboard had been first subjected to a series of preliminary tests, to ensure the basic operational state of the boards as a preparation for the actual functional tests [Sch09]:

- visual inspection with a microscope of the solder state of the active and passive components and of the connectors mounted on the board;

- manual verification of the main electrical circuits with a digital multimeter, to ensure that no short or open circuit is present on the board;

- power-up tests before and after the mounting of the daughtercards;

- uploading of firmware binaries into the VME_CPLD, Flash_CPLD and the ATmega microcontroller devices. Subsequently, an automated verification of the voltages and temper-

atures across the board was performed, by retrieving the respective values via the ATmega microcontroller and the VME interface (see also section 7.3);

- uploading of firmware binaries for the ReM_FPGA and for the FPGAs of the LVDS Cable Driver (LCD) daughtercard, over the VME into the Flash memory, and subsequent verification of the downloading procedure from the Flash to the corresponding devices.

The functionality of each fully equipped PPM board, validated by the initial tests, was then verified using a two-step test procedure. During the first step, the PPM board was operated individually on the test platform, and an extended set of automated functional tests was performed. In case of misbehaviours, the identified faulty components were replaced and the entire set of functional tests was repeated, until a stable hardware configuration was obtained. During the second step of the test procedure, the PPM board was operated together with other 15 PPMs in a crate configuration similar to the one in use at CERN, and its functionality was tested over a long period of operation. In case of additional misbehaviours, the faulty components were replaced and the entire two-step test procedure was repeated. Upon a successful completion of all the functional tests, the module was labelled as *operational*, and all the tests results were recorded in a dedicated database [Dat].

The following sections describe the functional tests performed during the adopted two-step test procedure.

7.2   Single Board Tests

7.2.1   The Test Setup

The setup used for the individual testing of the PPM boards is schematically described in figure 7.1. It is a VME-based setup consisting of:

- one custom-built 9U VME crate;

- the PPM under investigation;

- one function generator[1] and one DC power supply[2] providing the analogue input;

- two custom-built modules, the Readout Transfer Card (RTC) and the Universal Receiver Unit (URU), to capture and analyse the PPM real-time and readout output data;

- one *home-brew* Single Board Computer (SBC) acting as the crate controller;

The Analogue Input

The function generator is used for testing the analogue processing performed on the PPM. The device is operated in single-shot mode, which means that a single analogue pulse is generated upon the receipt of an external trigger signal. The type of waveform signal generally used during

---

[1] Agilent 33250A, 80 MHz Function\Arbitrary Waveform Generator
[2] Instek GPS-4303 Multi-Output Power Supply
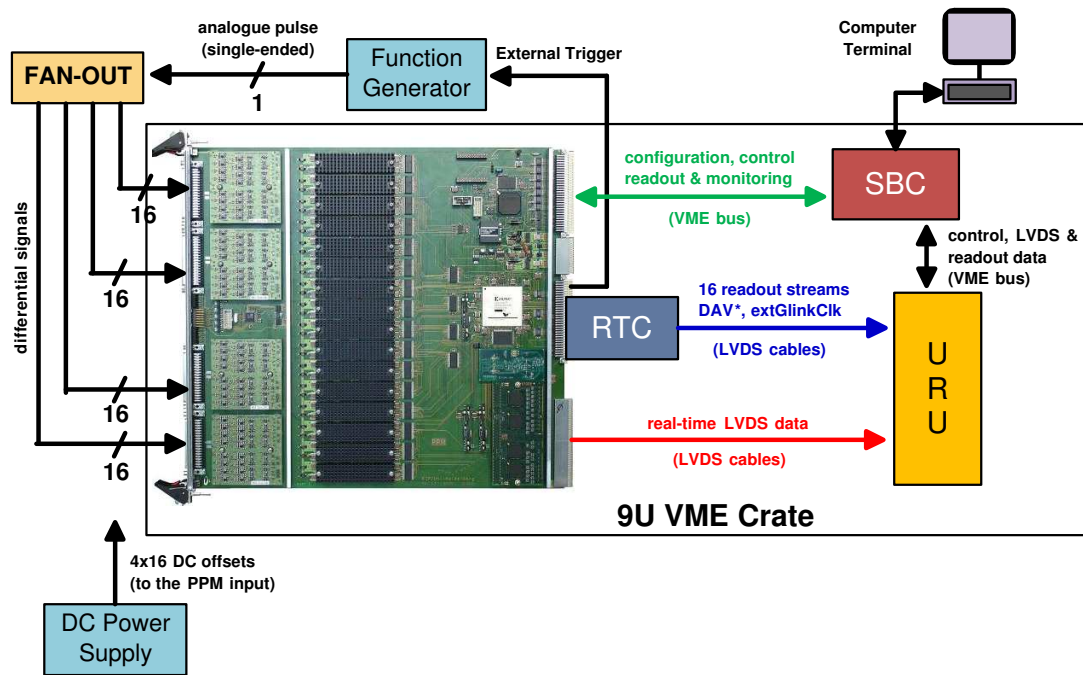
Figure 7.1: Schematic representation of the setup used for the single PPM board tests.

the tests is shown in figure 7.2. It represents a triangular pulse with a rising edge of roughly 50 ns, similar to that of the calorimeter trigger-tower signals, a slightly longer falling edge, and an amplitude value equivalent to about 900 FADC counts. This analogue pulse is routed to a 1 to 64 cascaded fan-out system, and the resulting copies are fed as differential signals to the input of the PPM. The external trigger signal, which enables the output of the function generator, is created by the ReM_FPGA upon a VME request, as described in section B.3.18.

The DC power supply is used for generating an additional offset to the value provided by the PPrAnIn-DACs, in order to allow a full scan of the digitisation window (see section 7.2.2). The DC power supply and the function generator are used during separate tests, thus the DC offset is not applied to the waveform pulse produced by the function generator.

The Universal Receiver Unit and the Readout Transfer Card

The URU is a modular VME system designed to record the real-time LVDS and the readout data transmitted by the PPM. It consists of a 6U VME motherboard and two Common Mezzanine Cards (CMC): CMC LVDS Multiplexer (CMC_Mux) and CMC LVDS Receiver (CMC_Rx) (see figure 7.3). The main functionality of the motherboard is to ensure the transfer of data over the VMEbus, between the two daughtercards and the crate controller. Its central components are a Xilinx XC4010XL FPGA and two 13-bit deep x 16-bit wide dual-ported memories[3] (DPRAMs). The FPGA is connected to the VME data bus through the DPRAMs. Its role is to transfer con-

---

[3]IDT7025-S20PF from Integrated Device Technology.
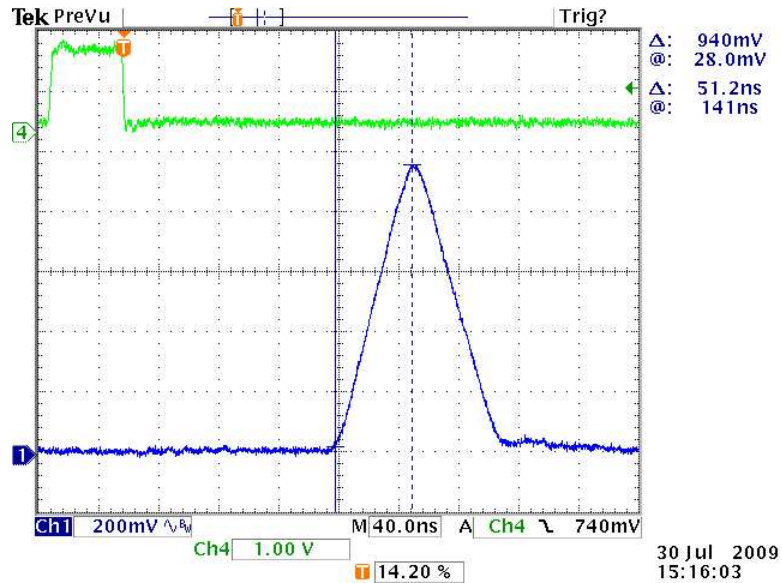
Figure 7.2: Oscilloscope shot of the pulse used to test the analogue processing on the PPM. The pulse seen in the upper left corner of the picture represents the external trigger signal that enables the output of the function generator [Sch09].

figuration and control data from the VME to the two daughtercards, and to collect and provide to VME status and memory data from the same modules.

The CMC_Mux is an FPGA-based board designed to either sample the PPM real-time LVDS data or to verify *on-the-fly* the integrity of the data transmission. The 10-bit serial LVDS output from the PPM is transported to the input of the CMC_Mux via 22 assemblies of shielded parallel cables, each of which is 15 m long[4] and carrying four twin-pair signals. On the CMC_Mux, the signals are converted to single-ended form and routed to the on-board FPGA[5]. The functionality of this device is twofold. First, the FPGA selects the four signals delivered by a pre-selected cable assembly, converts them back to LVDS form, and routes them to the CMC_Rx card via a 20 cm long cable assembly of the same type as described above. The selection of the input signals is done by the FPGA according to the value stored by an internal VME-configurable register. On the CMC_Rx, the serial data streams are converted to parallel form by four LVDS deserialiser chips, compatible with the serialisers of the PPM, and the resulting 40-bit wide real-time data is routed back to the FPGA of the CMC_Mux card. Additionally, each deserialiser recovers the 40 MHz transmit clock from the input streams, and provides it as well to the same device. At this stage, the FPGA can be configured to operate in two modes. In the first mode, the device stores every 25 ns the input 40-bit data into a 14-bit deep x 40-bit wide VME-accessible local memory buffer, until the respective storage medium fills up. Subsequently, the full occu-

---

[4]in comparison, the LVDS output cables used in the configuration of the L1Calo trigger system are of the same type but only 11 m long. The usage of longer cables during the functional tests was meant to enhance the reliability of the data transmission over the distance used by the trigger system.

[5]Xilinx Virtex-II 1000 (XC2V1000)
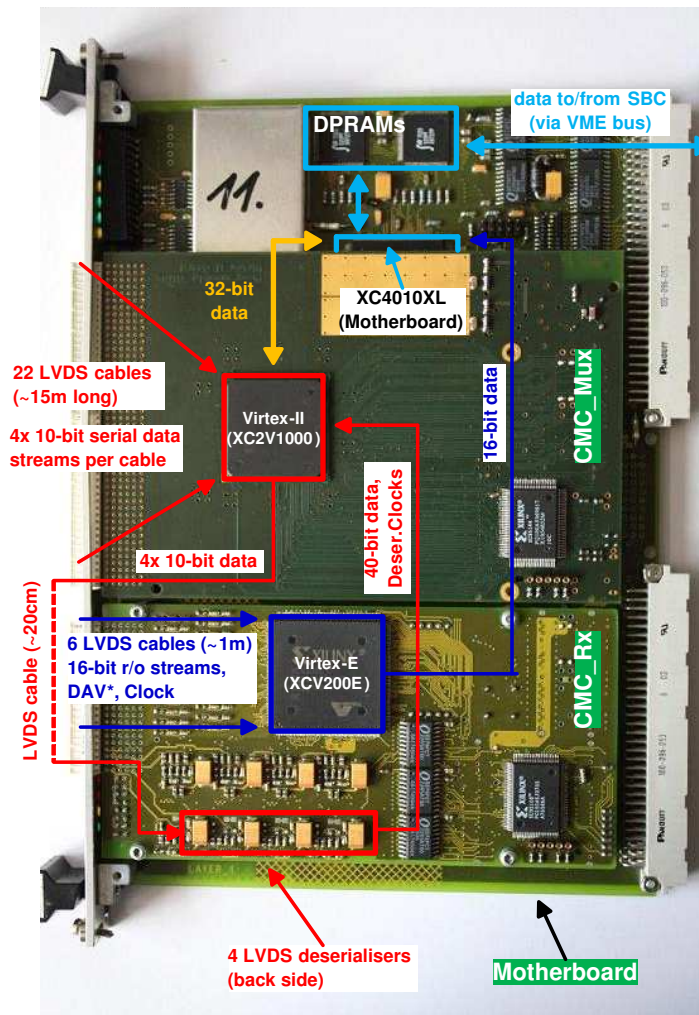
Figure 7.3: The Universal Receiver Unit (URU).

pancy of the buffer is flagged to VME via a dedicated status register. In the second mode, the FPGA computes an odd-parity bit, separately for each of the four 10-bit input data words, and compares it with the odd-parity bit embedded in the input data word, in order to detect eventual transmission errors (see also section 62). A set of four 8-bit VME-accessible counters, one per input data word, are incremented with each detected mismatch, until the respective counters overflow.

Apart from de-serialising the real-time LVDS data, the CMC_Rx card has also the functionality of capturing the PPM readout data. The 16 readout streams, the DAV* signal, and the serial frame clock transmitted by the ReM_FPGA are first received by the RTC card, which is mounted on the rear side of the crate backplane (see again figure 7.1). The RTC consists only of 24 differential line drivers, that have the role to transfer the input single-ended signals to the CMC_Rx, over six 1 m long LVDS cable assemblies of the same type as previously described. On the

CMC_Rx, the readout streams are first converted back to single-ended, and then routed to an on-board FPGA[6]. The device monitors the state of the DAV* signal, and stores the input 16-bit parallel data into a VME-accessible local memory buffer, upon detecting a high-to-low transition of the frame signal, i.e. the beginning of the event data stream (see figure 6.23). The decoding of the readout streams is then performed offline, by the controlling software.

7.2.2   The DAC Scan Test

As mentioned in the previous chapters, the PPrAnIn boards convert the input differential signals to single-ended, and rescales the obtained pulses in order to match the 1.0 V digitisation window of the PPrMCM's FADCs, i.e. 1.9 - 2.9 V. Additionally, two 8-bit programmable DACs on each PPrAnIn board allow to adjust the baseline of the single-ended signals, in steps of 2.4 mV within the dynamic range of 1.65 - 2.26 V. This adjustment is needed in order to ensure that the amplitude of the input signals is entirely visible to the FADCs, and that each signal is digitised with the same offset.

These latter tasks are accomplished by performing a typical calibration procedure, called the *DAC Scan test*. The test determines a linear relationship between the 8-bit DAC setting and the 10-bit FADC counts, in order to compute the DAC offset value needed to set the same pedestal in each PPM channel. The linear dependency between the two parameters is determined by measuring the noise level. Since the electronic baseline and the noise contribution are likely to vary from channel to channel, the linearity is determined separately for each of the 64 PPM channels, and an appropriate DAC offset is computed in each case. This operation of setting a common pedestal value at the board level would then allow further functional tests or debugging applications to achieve comparable results in all PPM channels. The pedestal value typically used during the present tests is 40 FADC counts, which ensures that the test analogue pulse previously shown in figure 7.2 will neither saturate nor underflow the FADCs.

Apart from this, the DAC Scan is an useful application for testing the functionality of multiple on-board components. By recording and analysing the 10-bit raw FADC data for each 8-bit DAC setting loaded into the system, one verifies the transfer operation of configuration data from the ReM_FPGA to the PPrAnIn-DACs, over the SPI bus, as well as the proper operation of the PPrAnIn-DACs and the PPrMCM-FADCs. Also, since the raw FADC data is retrieved from the board via a readout operation, one can evaluate the performance of the respective processes in the PPrASICs and the ReM_FPGA.

The test procedure is automated by software applications running on the SBC, and it is performed three times. During the first run, all the analogue input cables are disconnected from the front-panel of the PPM, in order to avoid noise contributions from third parties. Then, the software generates the 256 possible DAC values in a random sequence, and transfers them sequentially to the PPrAnIn-DACs, via the VMEbus, the ReM_FPGA and the SPI bus, as described in sections 6.2.5 and 6.4.2. After loading each DAC setting, the software generates a local L1A signal, and extracts the raw FADC data from the event blocks stored in the ReM_FPGA's VME Spy Buffers (see section 6.8). The 8-bit configuration data and the corresponding 10-bit raw FADC data are then stored in channel-wise profile histograms. Additionally, for debugging pur-

---

[6]Xilinx Virtex-E 200 (XCV200E)
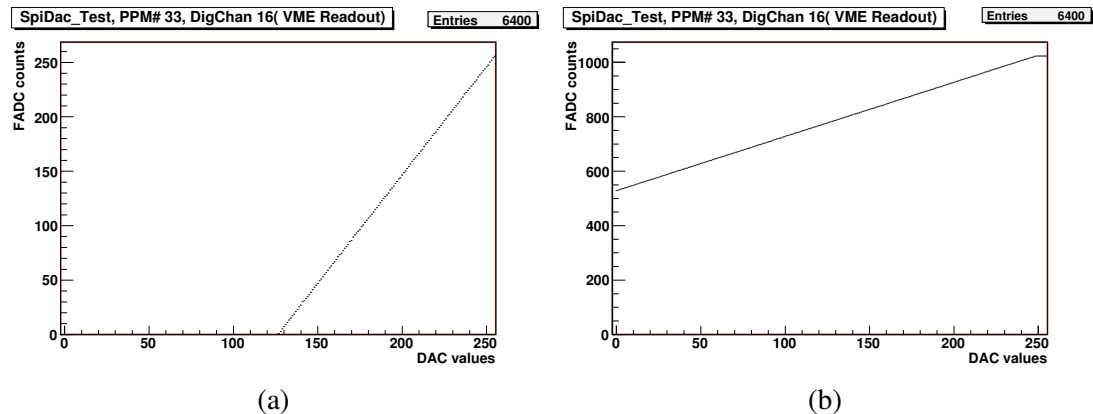
Figure 7.4: Examples of DAC Scan results achieved with 0 V (a) and 1.1 V (b) external DC offset.

poses, the FADC data is stored in 64 x 256 one-dimensional histograms, each histogram storing the data recorded for a given DAC setting and PPM channel. The readout mode used during the current tests was '5+1', which means that five FADC data words were read out for each channel and DAC setting. However, in order to achieve a larger statistics, the software can be configured to perform several consecutive DAC scans. After the last DAC scan is completed, a linear fit is applied to the obtained data, and the DAC value needed to set the pedestal to 40 FADC counts is then computed based on the slope and the y-intercept parameters returned by the linear fit. Figure 7.4a shows an example of channel profile histogram, describing the mean of the FADC counts as a function of the DAC value, as obtained after five consecutive DAC scans. For small DAC settings the FADC is in underflow, and thus the digitisation returns a zero count. The content of the respective bins is set to zero, and their range is excluded from the linear fit. Additionally, in order to account for those situations in which only the largest noise amplitudes are *seen* by the digitisation, resulting in small fluctuations of the bin content, all the *leading* bins of which content is non-zero but smaller than 5 mean FADC counts are excluded as well from the range of the linear fit.

The second and the third runs of the DAC Scan test are exclusively dedicated to verifying the linearity of the PPrMCM-FADCs in the upper part of the dynamic range. Since the dynamic range of the PPrAnIn-DACs overlaps only the lower part of the range of the PPrMCM-FADCs, i.e. 1.9 - 2.26 V, an additional DC offset is needed in order to shift the electronic baseline in each channel to the upper range of the digitisation window. This offset is provided by the external DC power supply, through the input connectors of the PPM (see again figure 7.1). During the second run the device is set to provide a 0.6 V DC offset, which allows to investigate the middle part of the dynamic range, while during the third run the power supply is set to provide a 1.1 V DC offset, which extends the investigations up to the saturation region. Figure 7.4b shows the results recorded with an external DC offset of 1.1 V. It should be noted that the external DC offset also enables a complete investigation of the DAC output, since in the previous case an evaluation was possible only for the upper range of DAC settings (see again figure 7.4a).

Figure 7.5 shows a few examples of misbehaviours detected with the DAC Scan test. The effect seen in the first plot (see figure 7.5a) was identified to be produced by an open DAC pin,

(a)

(b)

(c)

(d)

(e)
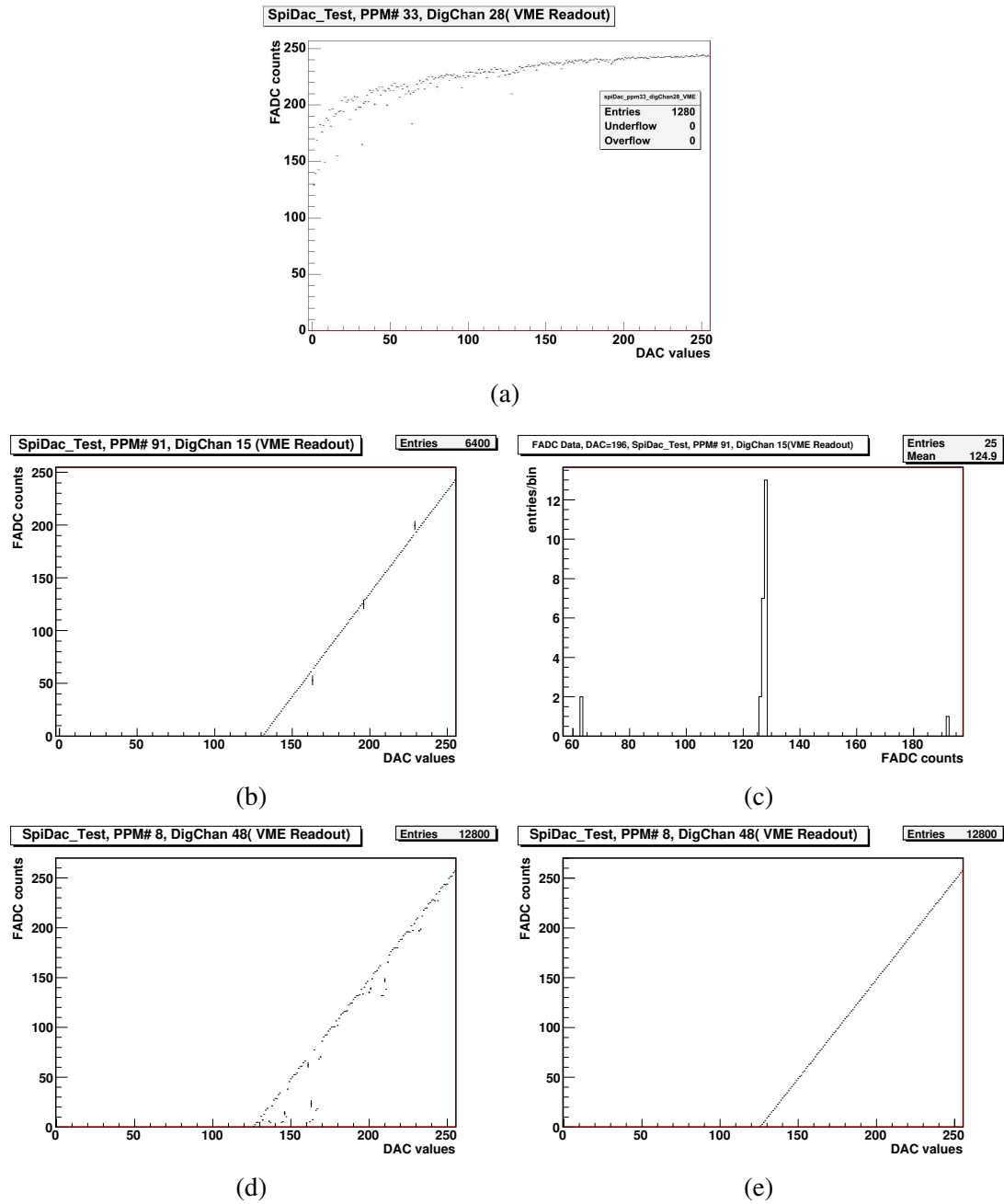
Figure 7.5: DAC Scan results pointing to faulty hardware components and erroneous firmware operations: open DAC pin (a), stuck and missing FADC bits (b,c), and improper transfer of configuration data over the SPI bus (d). The last plot (e) shows the DAC Scan results achieved after a rework of the ReM_FPGA firmware.

and it was completely removed after re-soldering the chip. The example shown in figure 7.5b illustrates a more common misbehaviour observed during the current tests. For three DAC settings, the mean of the recorded FADC data shows a significant deviation from the expected value. By analysing the content of the respective data, it was observed that the situation is generated by the FADC, which sometimes fails to properly update the digital output, when the value of several consecutive bits has to be changed from 0 to 1 or in the reverse order. This situation is illustrated in figure 7.5c, which shows the FADC data recorded for the DAC setting 196. It can be seen that most of the recorded FADC values are 128 (1000 0000, in binary representation) and 127 (111 1111), which is in agreement with the expected mean value for the given DAC setting. But, the actual mean FADC value deviates from the expected one due to two outliers, 63 (11 1111) and 192 (1100 0000). In both cases the FADC fails to assert correctly the 7th bit of the digital output. In the first case, the bit value is set to zero, suggesting that the failure has probably occurred upon a change in the output from 128 to 127. In the second case, the same bit is erroneously set to 1, suggesting that the bit got stuck upon a change in the output from 127 to 128.

Finally, the case illustrated in figure 7.5d describes an erroneous DAC Scan result due to an improper mastering in the ReM_FPGA of the data transfer over the SPI bus. The plot shown in figure 7.5e shows the results achieved after the respective firmware implementation was re-worked.

7.2.3   The External BCID Test

The External BCID (ExtBCID) signal is the binary output of the comparator on the PPrAnIn board, which discriminates between the input signal and the programmable threshold voltage provided by one of the two dedicated PPrAnIn-DACs. The binary output is set to logic value "1" as long as the input signal is above the given threshold, and to "0" otherwise. The 16 ExtBCID signals generated by each PPrAnIn board are routed to four corresponding PPrASICs, where they are registered with the system clock, synchronised with the incoming FADC data, and fed to the input of the BCID Decision Logic. A copy of the synchronised ExtBCID signals is provided by each PPrASIC in the event data block, for verifications of the pre-processing algorithms (see e.g. figure 6.18a).

The current functional test is performed in order to verify the correct assertion of the ExtBCID signal. At first, the offset values computed during the DAC Scan test are loaded in the system, in order to bring the pedestal of each channel to the same voltage level. Subsequently, a predefined threshold value, common to all channels, is loaded to the corresponding PPrAnIn-DACs. The 8-bit value typically used during the current test is 64, which corresponds to approximatively 157 FADC counts. This value positions the analogue threshold below the middle three FADC samples of the future digitised input pulse. This means that the ExtBCID data associated with the respective digital values is expected to be set to 1. The analogue pulse used during this test is the triangular waveform shown in figure 7.2.

Once the PPM is configured, the external trigger signal and the local L1A are generated via the ReM_FPGA, as described in section B.3.18. Subsequently, the event data is retrieved from both the ReM_FPGA and the CMC_Rx, and the ExtBCID data provided by both readout blocks is extracted and compared with the expected result [Sch09].

7.2.4   The FADC Test

This functional test represents an extension of the investigations carried out during the DAC Scan test. If at that step the proper operation of the FADCs is verified by studying their response to a constant input DC offset, the current test application investigates the response of the FADCs to a variable input voltage.

   The analogue input signal used during this test is the same triangular pulse previously shown in figure 7.2. Upon generating an external trigger signal and a local L1A, the pulse is fed to the input of the PPM, and subsequently five FADC samples describing the digitised input pulse are read out from each channel. Since the function generator outputs an identical pulse each time it is triggered by the PPM, the FADCs will always sample approximatively the same regions of the input pulse. In order to increase the efficiency of the testing method, the PHOS4 setting is stepped from 0 to 24. This operation delays the digitisation strobes with respect to the system clock, with a number of nanoseconds corresponding to the given PHOS4 setting, and determines the FADCs to sample at different points along the pulse profile.

   During the execution of the test, the controlling software generates 10,000 events, i.e. local L1As, for each PHOS4 setting. The FADC data resulting from these events is collected in 64 x 5 x 25 local memory buffers, each of which stores data associated with a given FADC and time-slice. The analysis of the accumulated data is performed separately for each buffer. The software determines the largest and the smallest recorded FADC value, and flags an FADC as *problematic* if the arithmetic difference between the two outliers is larger than 50 FADC counts. This limit was applied after it was observed that in most of the *good* cases, the *spread* is between 30 and 50 FADC counts. This relatively large value is induced by a few effects. First, a change in the input voltage value while the FADC is sampling determines a less accurate digitisation result. The degree of accuracy is proportional to the slope of the input signal, such that a large change in the input voltage determines a large deviation of the digital result from the true value. This effect was observed to be dominant in the data associated with sampling points on the rising and falling edges. Other effects contributing to the large spread of FADC values are a significant jitter of the pulse timing, which is induced by the function generator, and electronic noise from the analogue chain.

   A spread larger than 50 FADC counts is mostly generated by missing or stuck FADC bits, as in the example previously shown in figure 7.5c [Sch09].

7.2.5   Real-Time LVDS Data Tests

As mentioned in section 5.3.2, the real-time output from the PPrASIC consists of three 10-bit parallel data streams. Two of these streams provide trigger-tower transverse energy values for the Cluster Processor (CP), while the third stream provides jet sums for the Jet/Energy-sum Processor (JEP). The data streams are serialised at a rate of 400 Mbit/s by the three PPrMCM-LVDS transmitters[7], and then routed to the LVDS Cable Driver (LCD) daughtercard. The LCD, which receives a total of 48 serial streams from the 16 PPrMCMs, duplicates the signals near the ϕ boundaries of the PPM, in order to provide the overlap needed by the sliding algorithms

---

[7]the actual rate output by each LVDS transmitter is 480 Mbit/s, because the device appends one high-state start bit and one low-state stop bit in order to frame each input 10-bit data word [Nat02].

of the L1Calo processors, and drives all the signals over 11 m long cables to the subsequent processors, while applying an RC pre-compensation to minimise the signal degradation.

The functional tests described in this section are carried out in order to check the digital part of the pre-processing chain and the integrity of the 10-bit real-time data after a transmission over long LVDS cables. The test procedure is mainly based on the comparison between the expected result, deduced from a "known input", and the recorded output from the PPM. The input data and the expected result are provided by the software. The input is represented by several pre-defined digital test patterns, which are loaded individually in the PPrASIC playback memories, as described in section 6.4.1, and then injected in the real-time pre-processing path. The latter occurs only when the PPrASIC is configured to operate in a so called *playback mode*. Upon activating it, the PPrASIC selects the 11-bit wide content of the playback memory as input to the pre-processing logic, replacing the 10-bit raw FADC and the 1-bit ExtBCID data (see again figure 5.7). The input data is then processed, serialised and transmitted over the 15 m long LVDS cables to the URU, where it is collected in local memory buffers by the CMC_Mux card. Subsequently, the recorded data is retrieved by the software from the buffers, and compared with the expected result.

Because the content of the real-time data provided to CP differs from the content of the same data provided to JEP, the test applications developed for the two data paths are presented separately in the following.

The LVDS Output to Cluster Processor

The trigger-tower data sent to the CP consists of 8-bit BCID-LUT results. The two PPrASIC real-time data streams to CP provide together pre-processing results from all four trigger channels. This is realised by multiplexing the BCID-LUT data from two channels into one output stream, and it was implemented in order to halve the number of cable links between the PPr and the CP. The multiplexing scheme benefits from a functional aspect of the BCID Decision Logic, which blanks out the sample following the identified bunch-crossings. This aspect allows the usage of two consecutive bunch-crossing in order to multiplex over the same output link two non-zero energy values, each originating from a different channel. Furthermore, in order to allow the receiving end to assign the incoming data to the correct trigger channel and bunch-crossing, one 1-bit *Bunch-Crossing Multiplexing* (BcMux) flag it is attached to each BCID-LUT result. Figure 7.7 illustrates the multiplexing of the channel data in two consecutive bunch-crossings, and the values assigned to the BcMux flag. When associated to the first non-zero data, the Bc-Mux flag indicates the channel to which the respective energy value belongs, i.e. "0" - first channel, "1" - second channel, while when it is associated with the second consecutive non-zero data it indicates the bunch-crossing to which the second energy value belongs, i.e. "0" - previous BC, "1" - current BC. The multiplexed 8-bit BCID-LUT data and the BcMux flag are packed together with an odd-parity bit into a 10-bit data word, and sent in this format down to the CP (see figure 7.6).

The tests performed for the data links to CP focus on verifying the BCID-LUT result and its associated BcMux flag, and the integrity of the 10-bit real-time data after the transmission to URU. It has to be mentioned that in the former case the goal is to ensure that the PPrASICs are operational, rather than to study the performance of the pre-processing algorithms. This had
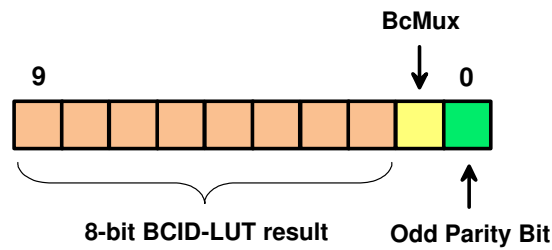
Figure 7.6: The content of the 10-bit real-time data sent to the Cluster Processor.

been rigorously investigated during the production stages of the PPrASICs and PPrMCMs, and the achieved results were documented [Web08]. Also, during the current functional tests the PPM boards were equipped only with PPrMCMs validated by the production tests.

The test applications for the data links to CP are described in the following:

- **Connectivity test.** The 32 PPrMCM serial streams containing pre-processing results for the CP are equally received by two dedicated FPGAs located on the LCD daughtercard. The task of these FPGAs is to duplicate the signals transmitted by the PPrMCMs located in the upper two and the lower two slot positions on the board, and to route all the signals to the backplane connector. The fan-out produces eight additional serial streams, which gives a total of 40 data links to CP. The signals are transported to the CP via 10 LVDS cable assemblies of the type described in section 7.2.1. Tables A.1 to A.3 show the mapping of the output signals on the LVDS cable connectors (see the fifth column in each table). The order in which the signals are listed in the tables corresponds to the order in which they are mapped to the backplane connector. This is also the order in which the signals are routed in the current test setup to the input of the CMC_Mux board.

  The current test is performed in order to verify the fan-out and the mapping of the output signals. Additionally, the test gives the possibility to spot, at an early stage and in a simple
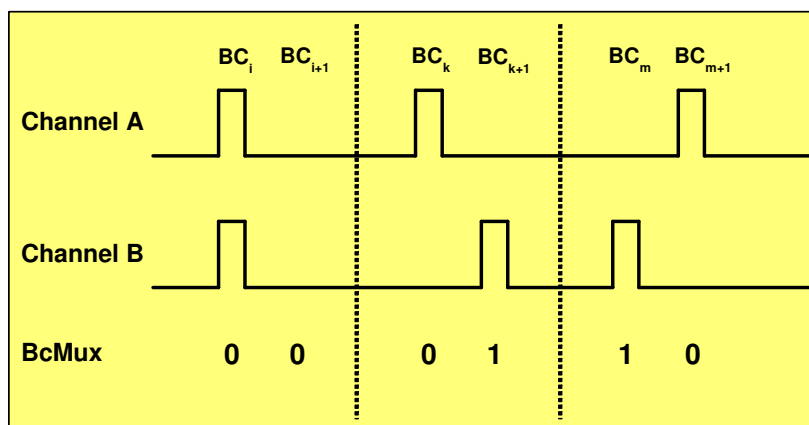
Figure 7.7: The PPrASIC bunch-crossing multiplexing scheme.

manner, eventual hardware misbehaviours caused by other components along the digital part of PPM's real-time path. Last but not least, the test serves as a verification tool for the proper routing of the LVDS signals from the PPM to the CMC_Mux.

During the test each playback memory is loaded with a flat signal of an amplitude that indicates the corresponding channel number. Since an input flat signal will always produce a constant zero BCID-LUT result, the PPrASICs are configured in this case to operate in a transparent mode, called *BypassBcMux*. When the mode is activated, the PPrASIC provides on the real-time output to CP the 10-bit input data to the LUT (see figure 5.7). Which means that the BCID Decision Logic and the BcMux algorithms are bypassed. Furthermore, if the FIR filter is also configured to operate in a transparent mode[8], then the PPrASIC provides on the real-time output the 10-bit raw FADC data or the 10-bit playback data, according to the operational state of the *playback mode*. Since each real-time output link to CP pairs up two channels, and since the BcMux logic is bypassed, only the data from one channel can be output at a time. The selection of the channel is done via a user-configurable parameter, i.e. *ChannelSelect*.

This combination of operational modes allows a direct comparison between the input data, i.e. the test pattern loaded in the playback memory, and the output recorded by the CMC_Mux card.

- **Data integrity tests.** These tests check the integrity of the 10-bit real-time data after the transmission over the long LVDS cable links. Two methods were implemented.

  The first method is based on the evaluation of the odd-parity bit. The bit provided by the PPrASIC indicates whether the total number of bits set to logic value "1" in the 10-bit data word is *odd* or *even*. In the former case the odd-parity bit is set to 1, while in the latter to 0. The FPGA of the CMC_Mux card computes as well an odd-parity bit based on the received 10-bit data, and compares it with the bit provided by the PPrASIC. The evaluation of the parity bit is performed *on-the-fly*, for each incoming 10-bit data word, and simultaneously for each of the four input data streams. The eventual mismatches indicate a hardware misbehaviour, and they are recorded by four 8-bit VME-accessible counters. This method has a well-known limitation. In case an even number of bits have changed their value during the transfer from the PPrASIC to the FPGA of the CMC_Mux, and none of these bits is the parity bit, then the comparison still returns a positive result. Which means that the error remains undetected.

  The second method implemented for the current functional tests is complementary with the first method. It is based on the direct comparison between the 10-bit input data and the recorded output. As previously described, the PPrASIC can be configured to provide on the real-time output to CP the data stored in the playback memories. Four digital patterns were used as input data during the current tests:

  - a ramp signal with an incrementation step of one digital bit per bunch-crossing (see figure 7.8a);

  - three identical sets of non-saturated triangular pulses, each pulse having a different amplitude (see figure 7.8b);

---

[8]FIR filter coefficients are set to {0,0,1,0,0}

– a set of saturated pulses with different saturation levels (see figure 7.8c);

– a *stress pattern* consisting of a repetitive block of 10-bit data words, of which binary representation describes a special succession of 1's and 0's (see table 7.1).

In all four cases, the first location of the playback memory is loaded with a number which is unique with respect to all the other 255 values of the test pattern. The role of this number is to unambiguously mark the beginning of the digital pattern, in order to facilitate the comparison between the input data and the recorded output. The usage of a *marker* is determined by the fact that the FPGA of the CMC_Mux starts writing the input data to the local buffers only upon the receipt of an appropriate command from VME. Thus, the writing operation is not synchronised with the beginning of the playback pattern.

During the tests the content of the playback memories is rolled continuously, which leads to multiple patterns accumulated in the local buffers. When data is retrieved, the software searches for the first occurring *marker*, and then it compares the remaining set of data with the input digital pattern. This works as long as the *marker* itself is not affected by transmission errors, e.g. missing bits. When this occurs, the whole data content is dumped in files, and the analysis of the errors is performed offline.

Also, it should be mentioned that all the patterns used during these tests are only 10-bit wide. The eleventh bit, which emulates the ExtBCID signal, is for convenience permanently set to zero.

Finally, the parity test is performed only with the digital patterns containing the saturated and the non-saturated pulses, as they generate multiple non-zero BCID-LUT results. Also, since during the parity test the data is not stored in the buffers, but analysed *on-the-fly*, the comparison stops only when requested from the VME, or when the four 8-bit counters overflow.

- **BCID-LUT test.** This test is in principle performed in order to verify the BCID-LUT result and the BcMux flag, but its results can serve as reference in understanding errors originating from different locations on the board. The three possible combinations of the BcMux flag, shown previously in figure 7.7, are each verified during a separate test. In

Hexadecimal	Binary
55	00 0101 0101
AA	00 1010 1010
155	01 0101 0101
2AA	10 1010 1010
333	11 0011 0011
CC	00 1100 1100
3FF	11 1111 1111
0	00 0000 0000

Table 7.1: The 10-bit data words composing the stress pattern.

Playback pattern: Ramp

11-bit data

playback memory location

(a)

Playback pattern: Triangles

11-bit data

playback memory location

(b)

Playback pattern: Saturated pulses

11-bit data

playback memory location

(c)
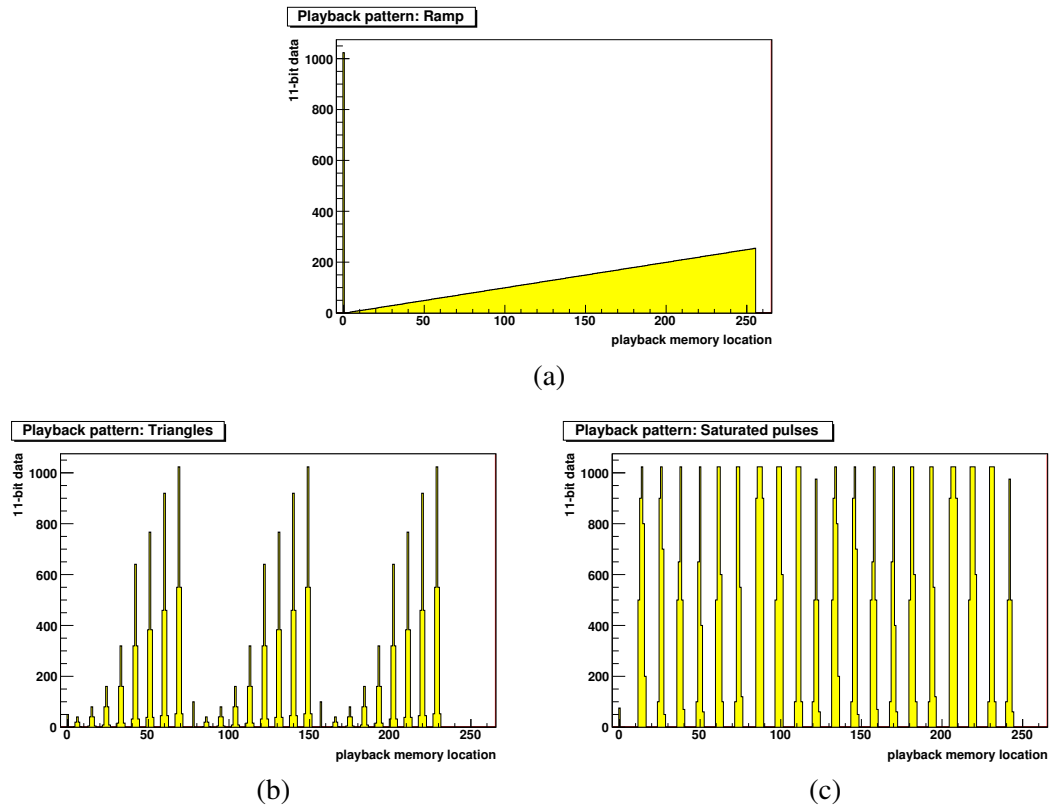
Figure 7.8: Digital test patterns used for checking the integrity of the real-time data to CP.

the first case, the playback mode is enabled simultaneously in all channels, while in the other two cases the input FIFOs are used in addition, in order to delay the channel data with respect to each other. The simultaneous activation of the playback mode is controlled from the VME, via the ReM_FPGA, as described in section B.3.17.

The input data used during the current test is represented by the digital patterns shown in figures 7.8b and 7.8c. The expected result is represented in each case by a 256 data words long pattern, describing the multiplexed BCID-LUT result for each bunch-crossing. Since the BcMux flag indicates the channel and the bunch-crossing to which the incoming data belongs, its value is automatically validated by a positive test result. In case of errors, the whole data set is dumped in files, and the analysis of the errors is performed offline.

The LVDS Output to Jet/Energy-sum Processor

The trigger data sent to the JEP consists of 0.2 x 0.2 ($\Delta\eta$x$\Delta\phi$) jet sums. These values are in principle obtained by first summing the 8-bit BCID-LUT results from each two PPrASIC channels (*JetPreSum*), and then by adding the two results (*JetSum*) (see e.g. figure 5.7). However, in order to preserve information about an eventual saturation in one trigger channel, JetPreSum is set to the maximum value (i.e. 511) if at least one of the two BCID-LUT results is saturated
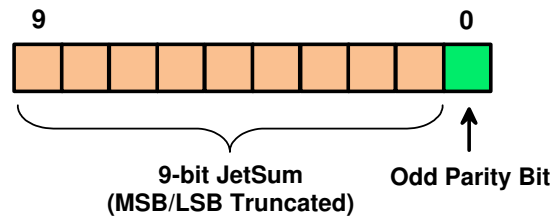
Figure 7.9: The content of the 10-bit real-time data sent to the Jet/Energy-sum Processor.

(255). Correspondingly, JetSum is set to the maximum value (1023) if at least one JetPreSum is saturated. Since JetSum is likely to have a value larger than 511, thus to be 10 bits wide, three modes of transferring this data to the JEP were implemented:

- **TruncateMSB.** The most significant bit (MSB) of the 10-bit JetSum is dropped, and the remaining 9 bits are packed together with the corresponding odd-parity bit into a 10-bit data word (see figure 7.9). This word is then sent to JEP;

- **TruncateLSB.** Similar with the previous mode, the least significant bit (LSB) being dropped instead;

- **NoTruncation.** The parity bit is dropped, and the 10-bit JetSum is transferred integrally to JEP. This mode is only intended for technical tests, and it is analogous to the Bypass-BcMux mode implemented for the data links to CP.

The tests performed for the data links to JEP are similar to those carried out for the data links to CP. They check the mapping of the output signals to JEP, the JetSum data, and the integrity of the 10-bit real-time data after a transmission over the corresponding cable links:

- **Connectivity test.** The 16 PPrMCM serial streams providing jet sums to the JEP are equally received by a second set of two FPGAs on the LCD daughtercard. As in the case of the FPGAs handling the outputs to CP, the task of these devices is to duplicate some of the received signals, and then route all the signals to the backplane connector. The fan-out scheme is a little bit more complex than in the previous case, because some duplication of signals is required for a jet-trigger in the forward regions of the calorimeter (see the sixth column in tables A.1, A.2 and A.3). The fan-out produces 17 additional serial streams, which gives a total of 33 data links to JEP. The transportation of these signals to the input of JEP requires 12 LVDS cable assemblies.
  In order to verify the fan-out and the mapping of the output signals each playback memory is loaded with a special pattern, which consist of only one non-zero value. This number was chosen in such a way, that the obtained 10-bit JetSum for each PPrMCM is unique with respect to the other sums. Then, the PPrASICs are configured to use the NoTruncation mode, for an easier identification of each channel.

- **Data integrity tests.** As for the case of the data links to CP, two methods were implemented in order to check the integrity of the 10-bit real-time data. The first method checks

the odd-parity bit, for both TruncateMSB and TruncateLSB modes. The second method uses the NoTruncation mode, in order to compare the 10-bit input data with the recorded output. Both methods use as input data the digital patterns shown in figures 7.8b and 7.8c.

- **JetSum test.** This test is carried out in order to verify the formation of the jet element, by using the TruncateMSB and TruncateLSB modes, and the two digital patterns mentioned above. Also, in order to increase the number of truncated JetSum values in the output stream, the input FIFOs are used for delaying the channel data.
  Finally, the expected result, to which the recorded data is compared, is represented in each case by a 256 data words long pattern, which describes the JetSum for each bunch-crossing.

Results

The most frequent type of error observed during the present functional tests was given by a bit systematically missing in the recorded 10-bit data. This means that the bit was permanently set to zero, thus determining a mismatch between the expected result and the recorded data. In the large majority of the cases, this type of error was found to originate from the PPrMCMs. Typically, the identification of a PPrMCM as error source is done in successive steps. First, upon detecting a systematic mismatch in the data provided by a certain CP or JEP link, the corresponding PPrMCM is moved to a different slot on the board, where previously no error was observed, and then the tests are repeated. If the same mismatch is now detected in the data link corresponding to the new slot position, the PPrMCM is considered as being the device generating the observed error. In most of the cases, the cause for this type of error seems to be a broken wire bond. The parity bit appears to be set correctly, but due to the missing bit in the remaining 9-bit data, the reconstructed parity value disagrees with the recorded value. This suggests that the odd-parity bit is correctly assigned in the PPrASIC, and that the missing bit is probably generated by an open connection, either at the output of the PPrASIC or at the input to the LVDS transmitters[9]. In other cases, the reconstructed and the recorded parity values match each other, which apart from a broken wire bond, points as well to a misbehaving PPrASIC.

Another type of error observed during the tests was given by bits shifted in the recorded 10-bit data word. This error was found to originate as well from the PPrMCM, but due to its nature, it is believed to be induced by a faulty serialisation in the LVDS transmitter.

Also, other errors were identified to originate from the LCD card, and to be caused by an improper soldering of the pre-compensation circuit. The partial or total absence of the pre-compensation was resulting in weak signals at the end of the 15 m long cables. Subsequently, the deserialisers on the CMC_Rx card were falsely interpreting the input data, generating bit errors or being unable to lock on the input signal.

---

[9]the devices mounted on the PPrMCM and their wire bonds are sealed by glob-top, in order to prevent their oxidation. Several attempts to remove the coating material, without damaging the wire bonds or the chips, have so far failed. Thus, the current hypothesis cannot be verified in practice.

7.2.6   Additional Tests for the ReM_FPGA

A significant part of the ReM_FPGA's functionality is indirectly verified during the tests described in the previous sections. The distribution of VME configuration data to the on-board devices, or the transfer of PPrASIC event data over the regular interface to DAQ, are some of the operations intensively exercised during the tests.

The only function that cannot be checked at this stage is the communication with the TTCrx, because the current test setup does not include TTC facilities. As mentioned in section 6.2.6, in the absence of the input bunch-crossing clock, the TTCrx stays in a reset state, and it cannot be accessed over the I^2C bus interface. This becomes possible during the full-crate tests (see section 7.3).

The following test applications were developed and performed in order to enlarge the investigations on the functionality of the ReM_FPGA, as well as to allow a more systematic verification of some of the algorithms implemented in the device. Additionally, the same applications provide a good indication about the operational state of the accessed devices.

PPrASIC Readback Test

As mentioned in section 6.6.2, the ReM_FPGA initiates a readback of PPrASIC register or memory data only if the corresponding upper four bits, called readback flags, in the MCM Readback block are set to value 4'b0001, or if the ForcedConfigReadback mode is enabled. The ReM_FPGA sets the readback flags to value 4'b0001 in two situations: either after a VME_Reset, or when configuration data is transferred from VME to the respective locations.

Two methods were implemented for testing the readback operation. The first method generates a VME_Reset, and then reads back the default values stored in the PPrASIC register and memory locations. The ReM_FPGA places this data in the SRAM, in the MCM Readback block, and sets the corresponding flags to value 4'b00000. Subsequently, the data is read out from the SRAM and compared with the expected values. Then, the default values are read back once again from the PPrASIC, this time by enabling the ForcedConfigReadback mode, and checked again against the expected values.

The second method loads first a pre-defined set of register and memory data to the PPrASICs, and then reads out the respective locations, and compares the two sets of data. As in the previous method, the ForcedConfigReadback mode is enabled, and the data is read back once again from the PPrASIC and compared with the input values.

Several PPrASICs with an internal faulty operation were identified with these methods. Some of the register or memory data provided by these PPrASICs was corrupted, due to missing or shifted bit values.

Rate Metering and Histogramming Test

The aim of this test is to ensure that the ReM_FPGA handles correctly the transfer of energy rates and histograms from the PPrASICs to the SRAM, and that the data received from the PPrASICs does not contain bit errors. The test is usually performed after a DAC Scan test, by measuring the noise level.

The readout of the PPrASIC rates is verified in three steps. First, the 10-bit energy threshold is set well above the pedestal, and the 16-bit timing parameter is set at maximum in each PPrASIC channel. This configuration determines the Rate Metering process to stop only upon the overflow of the internal 16-bit time counter (see again figure 6.26). The data read out from the PPrASIC consists of the 20-bit data counter value and the 16-bit timing counter value, which in this case are expected to be set to 0 and 65535 (FFFF, in hexadecimal) respectively. Once the ReM_FPGA has placed the data in the SRAM, the integrity of the timing value is checked. At the second step, the energy threshold is set below the pedestal, while the time setting is kept unchanged. This determines the Rate Metering to stop upon the overflow of the internal data counter. Similar to the previous case, the integrity of the data counter is verified once the data is available in the SRAM. During the last step, the energy threshold is placed again well above the pedestal, and a distinct timing parameter value is applied for each channel. This distinct value is meant to act as a *channel identifier*. After the data is read out, the timing values received from the PPrASIC are checked in order to ensure that the ReM_FPGA has placed the channel data in the expected order in the SRAM.

The readout of the PPrASIC histograms is verified in only one step. The 8-bit energy threshold is set well below the pedestal, in order to determine a histogram bin to saturate in a short period of time. After reading out the content of the memories, the integrity of the 11-bit saturated value is checked.

Event Readout Tests

These tests are performed in order to check the processing of the PPrASIC event data in the specified ATLAS format. The investigations are carried out using the analogue pulse from the function generator, and they are mainly based on the comparison between the readout data captured by the CMC_Rx test card and the unformatted copy stored by the ReM_FPGA in the VME Spy buffers.

The FADC and BCID-LUT data words, as well the three "BC Marks", i.e. PeakFinding BCID, Saturated BCID and External BCID, are verified via a direct comparison between the related data stored in the two buffers. Also, since the first five error bits, i.e. the four CD bits and the MA bit, are set from VME, their value can be directly checked. The 12-bit bunch-crossing number is evaluated by comparing the incremented value of its four least significant bits with the 4-bit bunch-crossing counter value available in the EventHeader. Subsequently, the result of this comparison is used to evaluate the BNM bit. The AFF error bit is verified by checking the value of the HeadersOnly and DataLoss flags from the Event Header, while the ENM bit is checked by comparing the 4-bit event counter value from the same EventHeader with a reference computed by the software.

The proper verification of the time-out (TO) bit requires special testing conditions. When set to logic value 1, the bit indicates that the corresponding PPrASIC did not send event data in response to an L1A. This situation can be artificially induced. The PPrASIC can be configured to stop sending data over one or both of its serial interfaces. Subsequently, the ReM_FPGA will set the TO bit to 1 upon receiving event data from the other PPrASICs. This allows then to verify the mechanism that sets the TO bit.

Also, the mechanism behind the RFC bit can be properly verified only by inducing a faulty

state. When set to logic value 1, the bit indicates that the dual-ported memories of the Ro-dReadoutManager are full, and that no further events will be buffered until the data stored by the memories is read out and transferred. To induce this situation, one needs to generate very high L1A frequencies. The present test application generates L1As with a frequency in the order of a few Hz. This is because a new L1A is generated only after the data of the current event is analysed and the eventual errors are logged. For this reason, the functionality of the RFC bit was verified during a separate test. The L1As were generated with a frequency in the order of a few MHz, and the assertion of the RFC and the DAV* signals was monitored with the help of an oscilloscope.

Access to SRAM

This test is performed in order to verify the 32-bit access to the SRAM from the VME, as well as the storage functionality of the device. For this, 32-bit checkered patterns[10] are first loaded from VME to each SRAM location, and then the SRAM content is read back over the VME and compared with the input data.

7.3   Full-crate Tests

The PPMs that have passed the single board tests are installed and operated in a crate configuration similar to that of the system in use at CERN, and their functionality is tested in a long period of operation. Figure 7.10 describes schematically the test setup. This consists of:

- 1 standard PPr crate (21 slot VME backplane);

- 16 PreProcessor Modules (PPMs);

- 1 Timing Control Module (TCM);

- 1 Readout Transfer Card (RTC) and 1 Universal Receiver Unit (URU);

- 1 standard Single Board Computer (SBC);

Additionally, a TTC Encoder/Transmitter (TTCex) module, operating in a standard TTC crate, generates and provides to the TCM a 40.08 MHz pulse train via an optical link [Tay]. The TCM converts the signal to electrical form, and then distributes it over the backplane to the TTCdec of each PPM. The presence of the external bunch-crossing clock brings the TTCrx back in an operational state, and thus enables the access over the I^2C bus to this device.

The functionality of the PPMs is verified by repeating most of the tests performed during the previous testing stage. The supervising software is adapted to consider the presence of 16 PPMs in the crate, and to transfer same configuration and control data to all modules during a given functional test. However, since the setup contains only one RTC and one URU, the real-time LVDS and the readout tests can be performed for only one PPM at a time. The strategy adopted

---

[10]32-bit data words consisting of alternating 0's and 1's. A 32-bit data word in which all the bits are set to 1 (FFFFFFFF, in hexadecimal) is in general avoided, as this value also indicates a data clash on the VME bus.
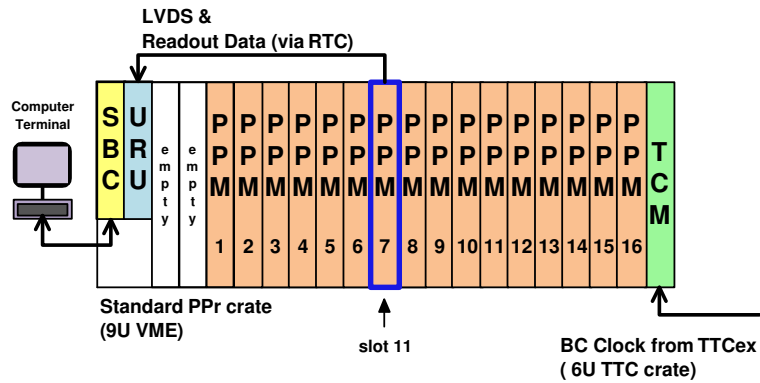
Figure 7.10: Schematic representation of the full-crate test setup.

in these cases was to define the eleventh slot of the crate as the permanent position for testing, because this slot corresponds to the hottest position in the crate. Thus, in this way one can verify the proper operation of the PPMs under the *hardest* conditions.

During the execution of the functional tests, the software reads out periodically from the ATmega microcontroller digital values of voltages across temperature measuring-diodes on the 16 PPrMCMs, in order to verify the operating conditions of the 16 PPMs in the very dense assembly. Figure 7.11 shows a two-dimensional histogram describing average temperature values for each PPM and PPrMCM, as recorded during a real-time LVDS test. It can be seen that, as previously mentioned, the hottest region in the crate is centered around the eleventh slot.This, as well as the general temperature profile, is due to the configuration of the cooling system. Two sets of fans, located at the bottom side of the crate, distribute a high-flux cold air stream upwards through the crate, to force the removal of the heat. Thus, the temperature values increase from the bottom to the top of the modules. Also, the hot region corresponds to the space between
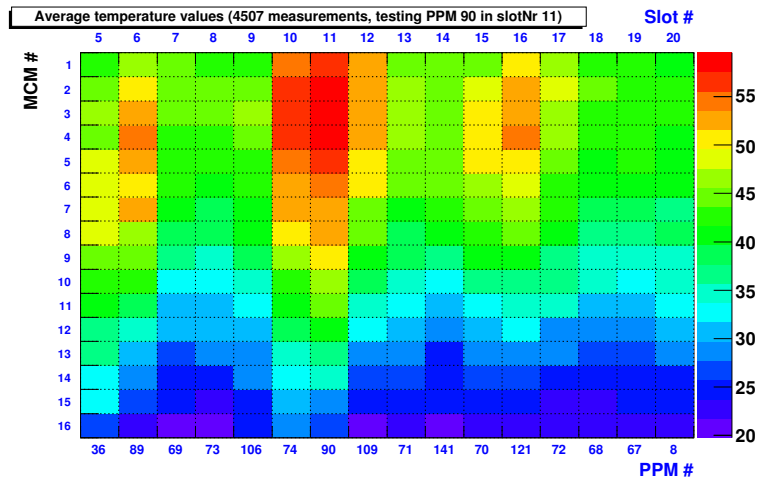
Figure 7.11: PPrMCM temperature map (slot number is given on the top).

the two sets of fans, where the air flux is less intense. However, the temperatures are in general below 60°C, which defines a safe operating state.

Apart from temperatures, the software also monitors different operating voltages. As in the previous case, the corresponding digital values are obtained from the ATmega microcontroller, which receives copies of the operating voltages from the power manager of the board. Also, the ATmega provides the same temperature and voltage values to the Fujitsu microcontroller, which then transfers them via the CAN-bus interface to the DCS. This functionality of the PPM was verfied as well, but at a later time than the current functional tests [Kho09].

7.4   Summary

No major mismanufacturing of the PPM boards was observed during the tests. The visual inspections or the automated tests have detected only minor solder problems, which were fixed in the local laboratory. Also, all the identified functional problems were localised on the daughtercards, and these were replaced with spare components.

The test procedure presented in this chapter is an ongoing procedure. It serves for maintaining a consistent number of *valid* spare PPMs, for the operation of the system at CERN, as well as for debugging and understanding the nature of the errors reported from CERN.

Chapter 8

The PreProcessor Operation in the ATLAS Experiment

The installation of the PPr system in the electronics cavern of the ATLAS experiment was completed at the end of 2007. Since then, the PPr has been involved in various integration and commissioning activities with the interfacing systems. In the summer of 2008, the whole ATLAS detector went through a commissioning phase with cosmic muons, to prepare for the first LHC beam. After a a short single-beam run in the autumn of 2008, ATLAS continued to collect cosmic muon events for detector alignment and calibration purposes. In November 2009 the LHC was restarted, and the first *pp* collision candidates were observed by ATLAS. This chapter presents brief overview of the results obtained during the integration and commissioning of the PPr system with the interfacing system, as well as during the cosmic muon and the LHC beam runs.

8.1   Integration and Commissioning Tests

The 124 PPMs are installed in eight PPr crates, which in their turn are housed in four racks. Figure 8.1 shows two photographs presenting the PPr system as installed in the underground USA15 electronics cavern. The leftmost picture (figure 8.1a) shows in the foreground four fully equipped PPr crates, housed in two racks, with analogue input cables connected to the front panel of the PPMs. These PPMs receive and process trigger-tower signals from the A-side of the calorimeters. The two racks seen in the background hold the electronics of the Receiver system. The second picture (figure 8.1b) shows a view from the rear of a PPr crate. Once can observe the massive LVDS output cabling, which transports the real-time data from the PPr to the L1Calo processors, as well as the RGTM cards (identifiable by the green LEDs) and the G-Link fibers, which are used for transmitting the PPr readout data to corresponding ROD modules.

During and after the installation of the system *in situ*, the main activities were focused on integrating the PPr with all the interfacing systems. One of the first tests done was to verify the connectivity and mapping of the analogue input and digital output cables. An improper soldering of the cables on the connectors or a wrong mapping of the signals upstream or downstream

(a)   (b)
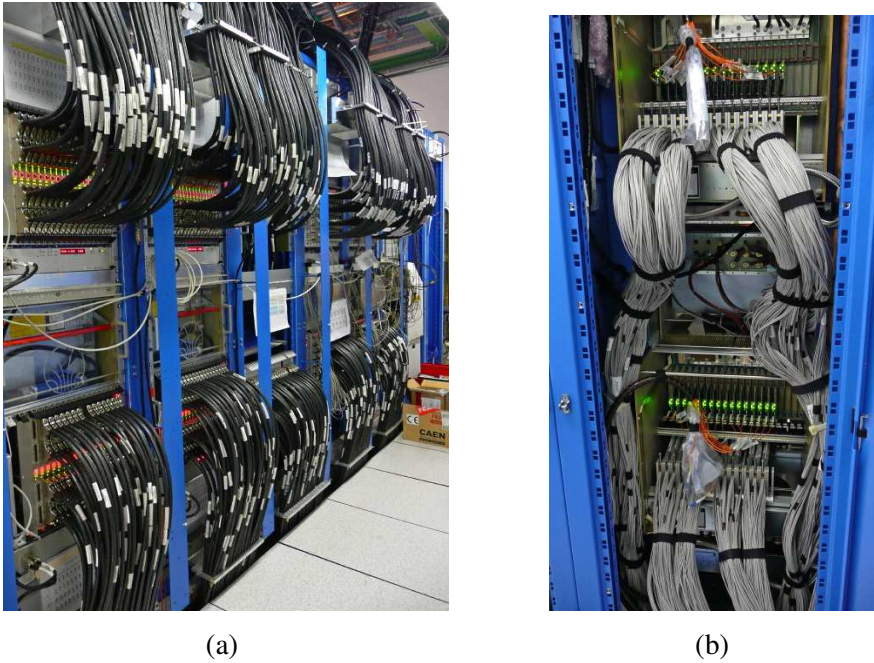
Figure 8.1: Views of the PPr system in USA15: four fully equipped PPr crates and front panel cabling (a, *foreground*), and output cabling at the rear of a PPr rack (b) [L1C07a].

of the PPr can lead to an erroneous trigger decision. The verification of the analogue inputs was performed using detector calibration systems [Ach07]. The Tile and LAr calorimeters have very precise charge-injection and pulser systems, which allow them to insert signal patterns of configurable amplitudes in their front-end electronics [ATL08a]. The digital links to the L1Calo processors were verified by loading a distinct pattern in the playback memory of each PPr channel. Only a few minor problems were found during the connectivity and mapping tests, e.g. interchanged cables due to swapped labels or faulty LVDS cables or connectors, and they were fixed.

The calorimeter calibration systems were also used to perform initial timing and energy calibration studies. In the first case, *coarse* and *fine* timing settings were derived, to compensate for the different analogue cable lengths from the calorimeters to the input of the trigger system. The *coarse* timing settings align all the input trigger signals to the same bunch-crossing. These values are set in steps of 25 ns, and they are loaded in the system as configuration parameters for the synchronisation FIFOs of the PPrASICs. The *fine* timing settings provide a further alignment of the analogue trigger signals, to ensure that each pulse is sampled at its peak. These values are set in steps of 1 ns, and they represent the configuration delay parameters for the PPrPHOS4s. In order to determine the *fine* timing settings, the PPrPHOS4 setting is stepped up within the 25 ns bunch-crossing interval, while the calorimeter calibration systems fire periodically the same pattern. This method allows to reconstruct with a resolution of 1 ns the pulses received from the calorimeters. The maximum amplitude of each pulse is then obtained offline, by fitting the resulting signal shapes [Mor09]. Figure 8.2 illustrates the case of a signal pattern from the
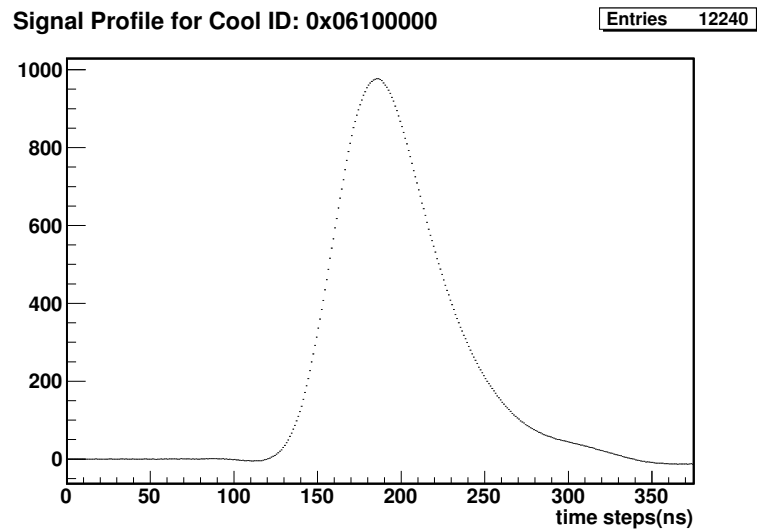
Figure 8.2: Reconstructed Tile calibration pulse [Chi09].

Tile calorimeter, which was reconstructed with the method described above. In this example, the PPr was configured to read out 15 FADC samples, therefore the reconstructed shape spans over 375 ns.

For the energy calibration study, the calorimeter calibration systems were employed to deliver pulses of different amplitudes for each channel. The energy values extracted from these pulses by the PPr were then compared against the energy given by the calorimeters, in order to derive appropriate *calibration constants* for each trigger tower. These initial results were later checked during the cosmic muon runs and after the first LHC proton beam data was recorded. Also, new *coarse* and *fine* timing settings had to be recalculated, to take into account the time-of-flight of the particles through the detector. The main difficulty for the timing calibration, during the cosmic muon or the LHC beam runs, is that the PPrPHOS4 delay setting cannot be varied, because a re-configuration of the PPr real-time data path during the data-taking is forbidden. Apart from this, the calorimeters deliver only physics pulses, at a large variety of amplitudes, and the PPr can be configured to read up to five FADC slices, to cope with the maximum L1 output rate (i.e. 100 kHz). Therefore, in these cases the recorded pulses are fitted with a dedicated function, which combines the Landau and the Gaussian functions (see e.g. [L1C08b]).

The accuracy of the energy results depends also on the efficient suppression of the noise component in each trigger channel. The measurement of the noise level is usually performed when the electronics upstream of the PPr is in operation. The procedure involves two steps. At first, appropriate receiver gain settings and a common pedestal value are set for all trigger channels. The pedestal is typically set to 32 FADC counts. This value is obtained by performing a DAC scan, in a similar manner as described in section 7.2.2. Subsequently, PPr FADC event data is accumulated channel-wise over a significant period of time, and then noise thresholds are derived offline for each trigger channel, based on the resulting RMS pedestal values. Figure 8.3a shows the result of a DAC scan for one PPr channel, while figure 8.3b presents the result obtained after a pedestal measurement, for the same PPr channel. An overview of the RMS pedestal

(a)

(b)

(c)
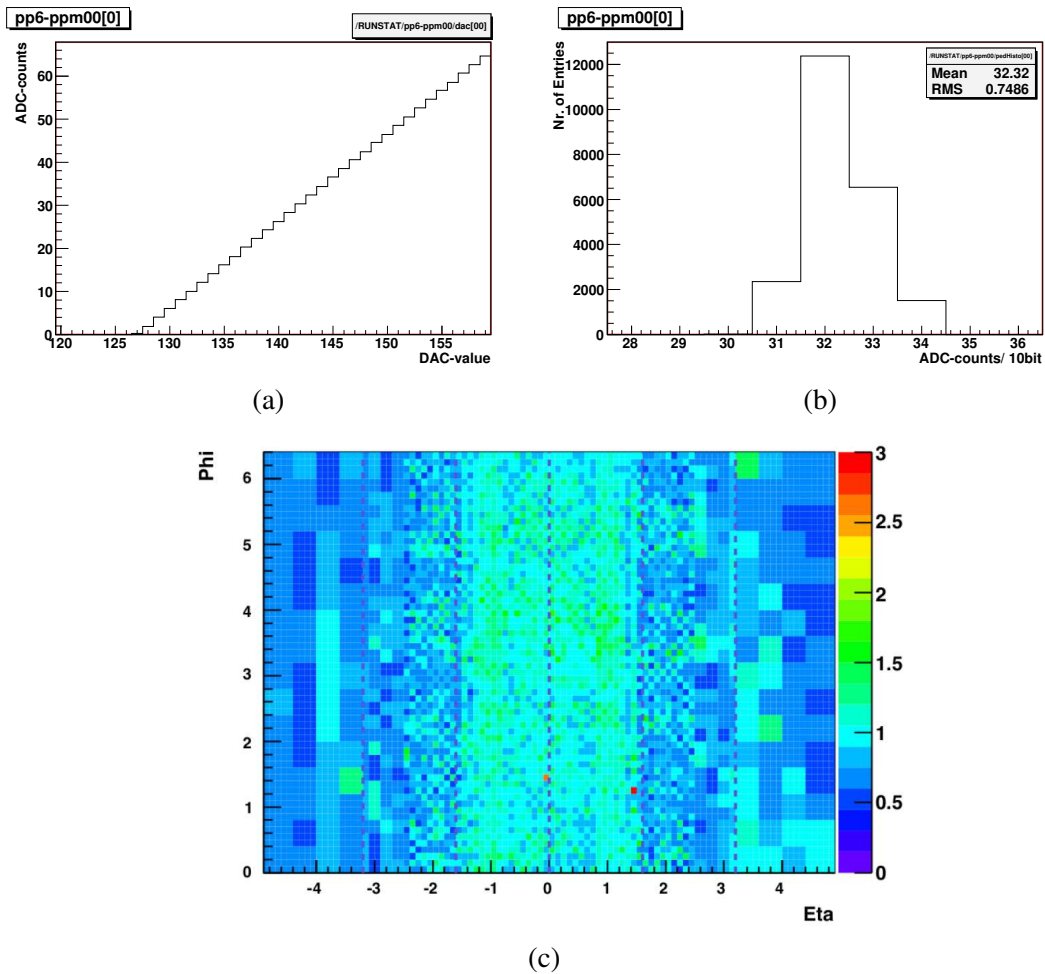
Figure 8.3: DAC scan and pedestal measurement for one PPr channel (a, b) [Chi09], and RMS pedestal values for the *electromagnetic* section of the PPr [Mor09].

values obtained in the electromagnetic part of the system, during the same measurement, is shown in figures 8.3c. The results are given in FADC counts, and they are plotted in a two-dimensional histogram, with η along the x-axis, and ϕ up the y-axis. It can be noticed that the noise contribution is up to 2 FADC counts (≈ 0.5 GeV) in most of the trigger channels, and it decreases at large pseudorapidity values. The latter is due to the $E \rightarrow E_T$ conversion applied in the electronics chain upstream of the PPr.

The pedestal measurements allow also to identify channels with a large noise content or with a faulty operation. These channels have to be disabled from the real-time path of L1Calo, as they can negatively influence the trigger performance. Their masking is typically done by setting the content of the corresponding PPrASIC-LUTs to zero, so that the BCID-LUT data sent by the PPr to the L1Calo processors is permanently zero on these channels. Another possibility to identify noisy, hot or dead channels is by monitoring the Rate Metering data provided by the PPrASICs. A dedicated procedure was implemented in the ATLAS Online Software to

Figure 8.4: Identification of problematic channels based on the PPr Rate Metering data [Mül08a].

Figure 8.5: DCS display showing the PPrMCM temperatures recorded in one PPr crate.

read out every 2 s the Rate Metering data over the VME interface, and display it into a GUI panel. This method offers two main advantages: firstly, a problematic channel can be detected also during a data-taking run, not only during a calibration run, and secondly, the monitoring of the trigger channels is performed independently of the L1 decision. The GUI is made available to the shift crew, which can take immediate action upon identifying a problematic channel. The same software tool archives channel-wise the Rate Metering data, to facilitate to a system expert the debugging of the observed problem [Mül08a]. Figure 8.4 illustrates an example of noisy trigger tower identified with the Rate Metering tool. The panel seen in the background is the GUI used for displaying the Rate Metering data. The fact that only couple of channels seem to have data is due to additional cuts imposed in software, to facilitate the identification of the eventual problematic channels. The histogram seen in the foreground of the same figure shows the Rate Metering *history* for one of the channels that have produced rates above the applied cuts.

Lastly, the PPr has been also integrated and commissioned with the Detector Control System (DCS). On each PPM, the Fujitsu microcontroller collects information about temperatures and voltages across the board, and sends it to DCS over the CANbus interface. The microcontroller sends also warning or error messages to DCS, if the PPrMCM temperatures or the supply voltages exceed pre-defined thresholds. In the current configuration, an warning flag is raised upon detecting a temperature value above $70°C$ or a fluctuation of a supply voltage from its nominal value larger than $\pm 5\%$. An error flag is then raised when a temperature value exceeds $80°C$ or when a supply voltage fluctuates with more than $\pm 10\%$ [Kho10]. Upon receiving an error flag, the DCS switches off the correponding crate. Figure 8.5 shows the format in which the PPrMCM temperatures measured in a PPr crate are displayed in the DCS control panel. Also, the DCS monitors the temperature in the racks and the crate power supply voltages.

8.2 Cosmic Muon Runs

The commissioning of the ATLAS detector with cosmic muons was started in 2005, in parallel to the detector installation. In summer 2008, when all subsystems were ready for data taking, ATLAS went through an intense period of commissioning with cosmic muons, to prepare the detector for the first LHC beam.

Cosmic muons occur when a primary cosmic particle undergoes nuclear collisions with atmospheric nuclei. These collisions produce a large number of charged π-mesons and kaons, which quickly decay into muons. Since muons have a relatively long lifetime and do not undergo strong interactions, but only lose part of their energy via electromagnetic processes, the most energetic cosmic muons are able to pass through the 75 m thick rock overburden, which roofs the experimental cavern[1], and through the ATLAS detector. At the passage through the ATLAS calorimeters, the cosmic muons mainly behave as minimum ionising particles, but they also radiate brehmsstrahlung photons. In most of the observed cases, the energy deposited was in the order of several GeV in the Tile hadronic calorimeter, the outer layer of the calorimetry, and a factor 10 lower in the LAr electromagnetic calorimeters [Hoe10].

---

[1]in fact, based on the angle of the tracks reconstructed by the RPCs of the Muon Spectrometer, it was observed that most of the cosmic muons detected by ATLAS enter the experimental cavern from the two access shafts [Wen09].

Most of the trigger-tower pulses produced by these events are typically only a few counts above the pedestal. The challenge was to discriminate these pulses from noise, and thus to trigger on genuine physics events. To accomplish this, L1Calo was seeded with an electron/photon E_T threshold of 3 GeV, and with τ-like and jet-like thresholds of 5 GeV [L1C08b]. Figure 8.6 shows the example of a cosmic event triggered by L1Calo, using the τ and jet thresholds. Additionally, figure 8.7 shows a comparison between the transverse energy measured by L1Calo, and the transverse energy detected in the cells of the electromagnetic calorimeters. The plot takes into account only the trigger-tower energies above the applied threshold, i.e. 5 GeV in this case, and the energy sum of the corresponding calorimeter cells. A good correlation between the two sets of energy values can be observed, although the calibration of the two systems was still in a preliminary phase.

Beside the participation in the trigger, the cosmic runs represented for the PPr a good opportunity to perform measurements of the noise level, identify hot and dead channels, and monitor the functionality of the PPMs in long periods of operation, as well as to tune its timing and energy calibration methods.

8.3 LHC Beam Runs

The First LHC Proton Beam

In September 2008, the first single-beams of protons were successfully circulated around the LHC accelerator ring. During several days, a single proton bunch, filled with 2 x 10^9 protons, was circulated without acceleration at the injection energy of 450 GeV. Gaining experience from the commissioning runs with cosmic muons, ATLAS was ready to record and analyse all the events generated by the passage of the single beam.

In the beginning of these commissioning activities, the tertiary collimators, placed on each side of the four main experiments[2], were being closed for safety. This was done in order to protect the LHC machine and the detectors from an eventual misalignment of the beam, and to allow corrections to be made, if necessary. Also for safety reasons, some of the sub-detectors prone to radiation damage were turned off or operated with reduced high voltage[3]. Each collision between the proton beam and the closed collimator produced a spray of particles, mostly muons, that reached the experimental caverns. In case of ATLAS, most of these particles hit the detector, generating a so-called *beam-splash* event. Figure 8.8 shows the first beam-splash event seen by the ATLAS, and which was triggered by L1Calo. In this example the beam was circulated clockwise around the accelerator, hence the collision occurred upstream of the A-side of the detector (i.e. right-hand side in the lower left plot). The particles reaching the ATLAS detector generated more than 100,000 hits in the muon chambers and left a huge energy deposition in the calorimeters, i.e. more than 1000 TeV in the hadronic calorimeter and several TeV in the electromagnetic calorimeter [Cos09].

The beam-splash events proved to be very useful in determining the timing alignment of the

---

[2]in ATLAS: 140 m from the interaction point.

[3]in ATLAS: Pixel detector, barrel SCT (*off*), End-Cap SCT, Muon System and Forward Calorimeters (*reduced voltage*) [Cos09]

Figure 8.6: Cosmic event triggered by the L1Calo τ and jet algorithms [L1C08b].

Figure 8.7: Comparison between the E_T reconstructed by L1Calo (x-axis) and the E_T measured in the full readout of the LAr electromagnetic calorimeters. The results are based on the 2008 cosmic running [L1C08b].
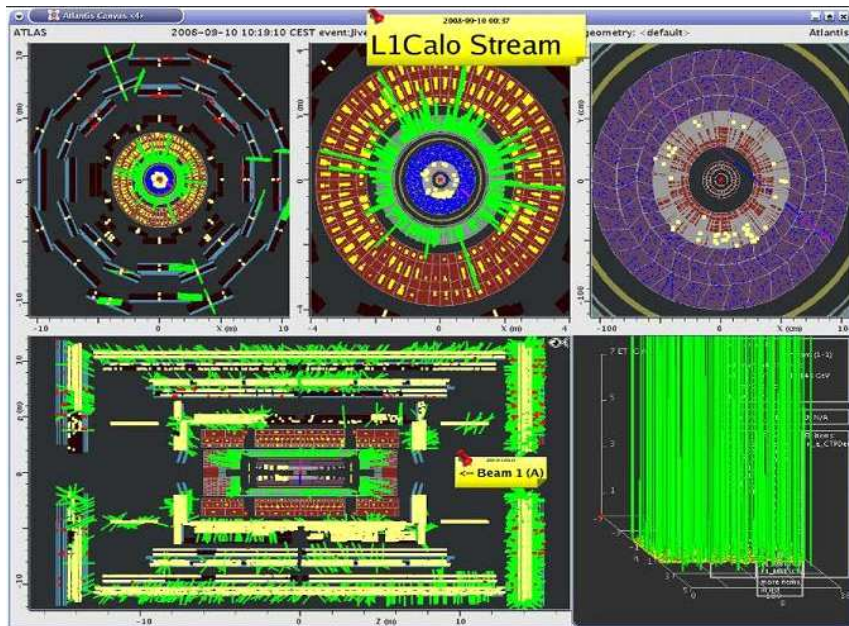
Figure 8.8: Beam-splash event triggered by L1Calo in September 2008 [L1C08b].
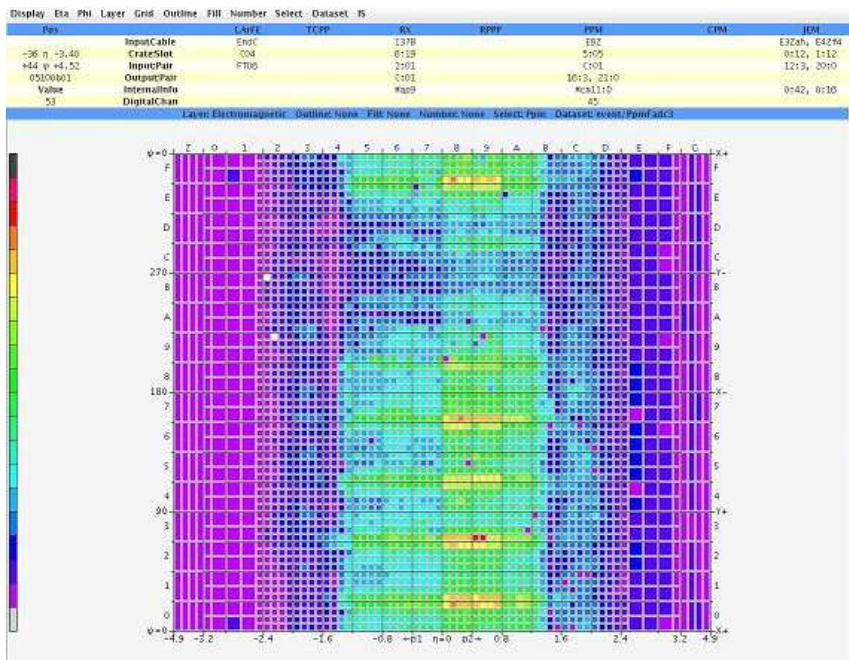
Figure 8.9: Energy deposition from a beam-splash event as reconstructed by L1Calo [Tri09].

L1Calo trigger with respect to the LHC bunch-crossings clock. Such an analysis is based on the assumption that the particles resulted from the collision with the collimators travel almost parallel to the beamline, and enter the detector almost in the same time, within a few nanoseconds. Figure 8.9 shows an example of a beam-splash event, as seen in the L1Calo trigger. It is a two-dimensional η-ϕ plot, with η along the x-axis and ϕ up the y-axis, describing the relative transverse energy deposited in each electromagnetic trigger tower. These energy values represent the middle 10-bit FADC sample provided by each PPM in the readout stream to DAQ. A difference between the A- and C-side of the electromagnetic layer can be noticed. This was determined to be due to one bunch-crossing tick misalignment of the PPMs that process the data from the C-side of the electromagnetic calorimeter, and it was corrected as a result of these observations [Tri09]. Also in the same plot, two structures in ϕ can be observed: an eight-fold structure, which is due to the eight coils of the A-side end-cap toroid magnet, and a low energy response region around $\phi = 3\pi/2$, which is believed to be due to the support structure of the ATLAS detector.

The First Collisions

After a break of one year, caused by an incident in one of the sectors of the accelerator ring [Baj09], the LHC has resumed its operation in November 2009. In the beginning, the LHC repeated the commissioning procedure performed in 2008, by circulating only one beam at a time in the accelerator. During this period, ATLAS recorded more than 100 beam splash events, produced on both sides of the detector, which were mainly used to synchronise the various subdetectors [Hoe10]. At the end of November 2009, the LHC circulated for the first time two beams simultaneously in the accelerator, at the injection energy of 450 GeV, and steered them to collide with $\sqrt{s} = 900$ GeV. Figure 8.10 shows a three-dimensional view of the first collision candidate observed by ATLAS.

After stable-beam conditions were reached, the LHC increased the number of proton bunches in each beam, from one to four, as well as the intensity of proton bunches, reaching to $\sim 1.5 \times 10^{10}$ protons per bunch at the start of the fills. The stability of the beams has determined ATLAS to switch on the Pixel and the SCT detectors, which had not been previously used due to safety reasons. With the full detector active, ATLAS has recorded about half a million collision at $\sqrt{s} = 900$ GeV [ATL09]. After a few days of operation in this configuration, the LHC ramped for the first time to 1.18 TeV/beam. This has led to *pp* collisions with a centre-of-mass energy of $\sqrt{s} = 2.36$ TeV, and it has made the LHC the highest energy particle collider in the world, superseding the $\sqrt{s} \approx 2$ TeV achieved by Tevatron, at Fermilab, Chicago. Figure 8.11 shows the first collision event at $\sqrt{s} = 2.36$ TeV observed by ATLAS. It represents a 2-jet candidate with uncalibrated transverse energies of approximately 6 GeV and 16 GeV.

In March 2010, the LHC set a new record, by accelerating the protons to 3.5 TeV/beam and colliding them at $\sqrt{s} = 7$ TeV. Figure 8.12 shows a three-dimensional view of one of the first collisions at observed by ATLAS at this centre-of-mass energy. This run mode is planned to be maintained for about 18 months, to allow a safe commissioning of the accelerator and of the detectors. After that, the LHC will start increasing the energy and the luminosity up to the design values of $\sqrt{s} = 14$ TeV and $\mathscr{L} = 10^{34}\,\mathrm{cm^{-2}\,s^{-1}}$ respectively.
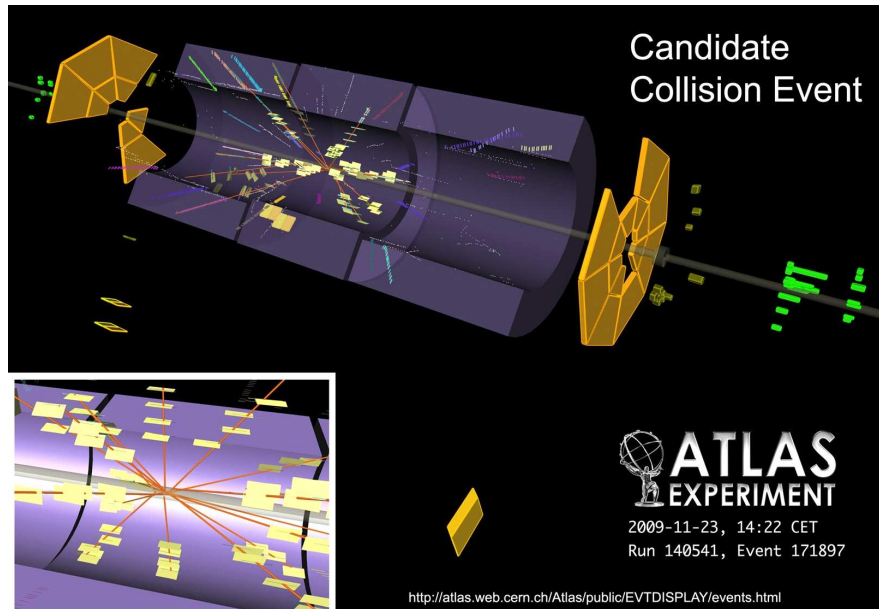
Figure 8.10: The first *pp* collision candidate recorded by ATLAS. The centre-of-mass energy of the collision was $\sqrt{s} = 900$ GeV [CER09b].
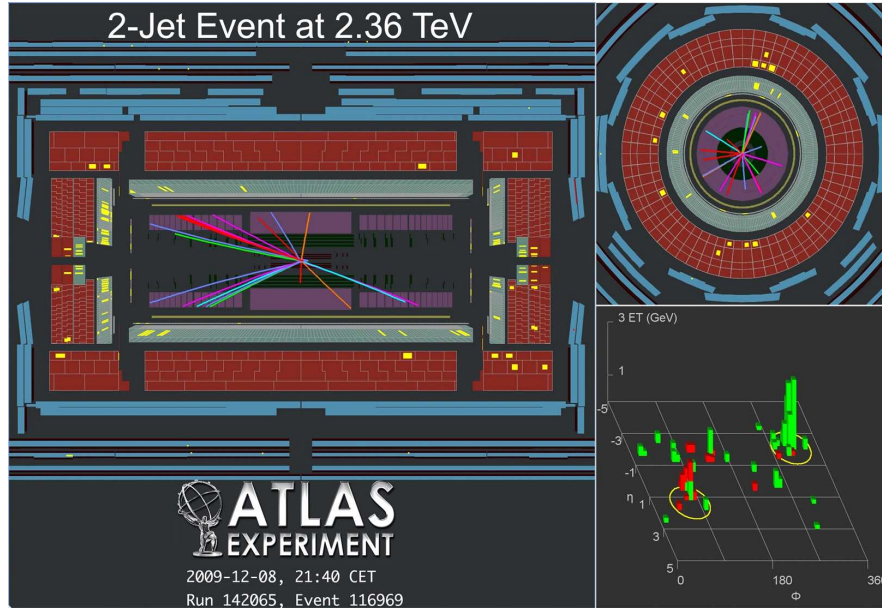
Figure 8.11: The first *pp* collision with the centre-of-mass energy of $\sqrt{s} = 2.36$ TeV recorded by ATLAS. [CER09a].
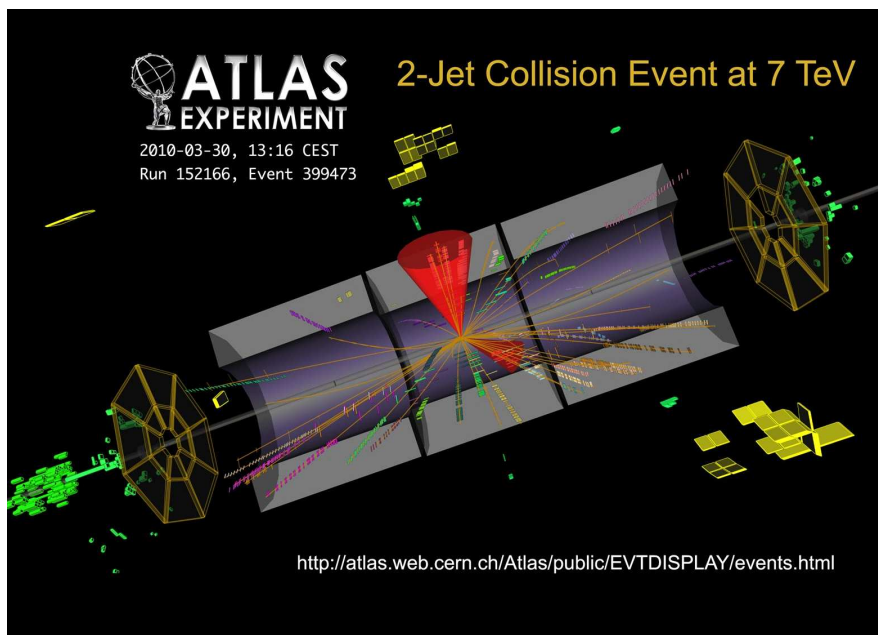
Figure 8.12: One of the first *pp* collisions with the centre-of-mass energy of $\sqrt{s} = 2.36$ TeV observed by ATLAS. [CER10b].

Chapter 9

Summary and Conclusions

The first part of this thesis has described the algorithms developed to meet the functionality of the Readout Manager FPGA (ReM_FPGA). The tasks assigned to the ReM_FPGA can be summarised as *data routing* and *data formatting*. The device interfaces to several distinct communication systems to transfer data to and from various on-board and external locations, in the format requested by the each communication system and destination device. The major task of the ReM_FPGA is to transfer PPM event data to the DAQ system. The ReM_FPGA collects the event data from the 16 PPrASICs, and sends it to DAQ in a predefined ATLAS format.The proper operation of the event readout task in the ReM_FPGA is of a special importance. Offline calibration studies or online software applications that monitor and verify the performance of the Level-1 trigger during the data-taking physics runs are based also on the data delivered by the PPMs to DAQ. The readout operation was tested intensively both in Heidelberg, in the laboratory environment, and at CERN, and it is currently running stably and error-free. The rare cases in which a misbehaviour was reported were due to either an improper configuration of the system or a defective hardware.

Another important task of the ReM_FPGA is to retrieve the RateMetering and Histogramming data from the PPrASICs and to store it in memory locations accessible over the VME interface. The respective data is used by online software applications to monitor the activity in the calorimeters or to identify problematic channels upstream of and in the PPr system. This operation is as well stable, no misbehaviour being reported. The same stability is also observed for other important operations performed by the ReM_FPGA, e.g. the transfer of configuration data from the VME to various on-board programmable locations, or the readback of configuration data from the PPrASICs and the TTCrx devices. For the cases in which functional misbehaviours are detected, the ReM_FPGA provides extended sets of VME registers that gather bitwise status and error data, received from the interfaced devices or generated by its internal algorithms, to help investigate these problems.

The second part of this thesis has presented the functional tests that had been carried out to ensure the proper operation of the PPMs, before they were installed at CERN. For this, a two-step test procedure was developed. During the first step, the PPMs were operated individually, and an extended set of automated functional tests was performed. The conditioning and digitisation of the analogue input was verified by employing a waveform generator to provide analogue pulses

with a peaking time similar to that of the genuine calorimeter signals. The digital processing and the transmission of real-time data over long LVDS cables was checked with digital playback data, consisting of either calorimeter-like pulses or various stress patterns. The PPM real-time data was captured and analysed by an Universal Receiver Unit (URU), a custom-built VME module, which emulates the receiving stage of the L1Calo processors. The readout operation was checked with both analogue inputs and digital patterns. A second custom-built module was used in this case to transfer the event data from the backplane connectors to the same URU. For more consistency of the readout tests, the formatted event data received by the URU was also compared with an unformatted copy, stored by the ReM_FPGA in VME-accessible memory buffers.

During the second step of the implemented test procedure, the PPMs were operated in a crate configuration similar to the one used in the experiment, and their functionality was verified over a long period of operation by repeating most of the tests performed during the first step. Additionally, the operating conditions of the PPMs in the dense crate assembly were checked by monitoring periodically PPrMCM temperatures and various on-board supply voltages. All the functional misbehaviours identified during these tests were localised on the daughtercards, and these were replaced by spare modules. Other problems referred to mis-soldering of passive components on the main board or on the daughtercards, and these were fixed in the local laboratory.

The installation of the PPr system at CERN was completed by the end of 2007. Since then, the PPr has participated in various integration and commissioning runs with the other components of the trigger system and with the whole ATLAS detector, which helped tune its operation. At the time this thesis is completed, the PPr system is operated successfully in data-taking physics runs with LHC proton beam data.

Appendix A

PreProcessor System: Channel Mappings

A.1 The Mapping of the PPM Channels to Detector Coordinates

The PPr receives the 7168 analogue trigger sums as differential signals, via 496 twisted-pair cables. All the input cables are identical, containing 16 differential analogue signal pairs. Most of the cables transport 16 analogue trigger-tower signals. Exception make the cables that carry trigger sums from the end-cap regions $2.4 < |\eta| < 3.2$. In these cases, only 8 differential pairs are populated with trigger signals.

The PPr consists of 124 hardware-identical PPMs, each PPM being designed to receive and process 64 differential analogue signals from four input cables. The input signals enter each PPM board through four SUB D 37c connectors. Figure A.1 illustrates the pinning on one input connector, and the adopted numbering convention for the 16 signal pairs.

Each trigger tower is identified by a set of η-ϕ coordinates[1] and a corresponding calorimeter layer type, i.e. electromagnetic or hadronic. Therefore, based on this information and on the labels defined on the input connectors, a correlation between the each PPM channel and the corresponding detector region is obtained. This is illustrated in figures A.3 and A.4, which show the trigger-tower granularity in the first ϕ-quadrant of the electromagnetic and hadronic calorimeter

---

[1] e.g. in the ATLAS Online software, the coordinates of a trigger tower are considered the central values of the η and ϕ bins.
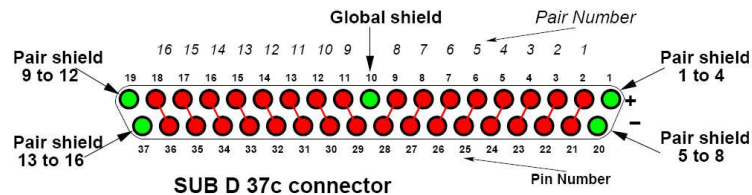
Figure A.1: Pin usage and assignment of differential pairs on the input PPM connectors [L1C07b].

layers, as seen by the PPr system. In each plot the η coordinates are displayed at the top, while the ϕ coordinates are shown on the left side. Note that, for simplicity, the ϕ coordinates are given in units of $\pi/32$. In the same figures, the four input connectors are each represented by a distinct colour, while the number attached to each trigger tower represents the label of the corresponding input signal pair on the given connector. The granularity in the electromagnetic and hadronic layers is in general identical. Exception makes the FCAL region, where the granularity of the electromagnetic FCAL1 trigger towers is $\sim$, while the towers in the hadronic FCAL2 and FCAL3 layers are ~ 0.8 *x* 0.4 wide, and they are identified by the same set of η-ϕ coordinates.

The trigger-tower granularity in the other three ϕ quadrants is identical with the one of the first quadrant. This is illustrated in figure A.5, which describes in the same diagram both the electromagnetic and hadronic layers. In order to simplify the figure, the labels of the input signal pairs are omitted, but the mapping described in figures A.3 and A.4 holds for the other three ϕ quadrants, too. Figure A.5 shows also the crate organisation of the PPr system. The 124 PPMs are housed in eight VME crates. The *fat* blue boxes indicate the η-ϕ space covered by each PPM. The number given within each box indicates the slot position of the given PPM in the corresponding crate. The red boxes shown at the bottom of the schematics indicate the eight PPr crates. The boxes are arranged in such a way to point to the PPMs they hold, and thus to η-ϕ space they cover. Note that two crates, i.e. 2 and 3, are equipped with only 14 PPMs, while the other six crates hold 16 PPMs. Also, the crates are organised in such a way that four of them process electromagnetic trigger-tower signals, while the other four process hadronic trigger-tower signals. However, an exception occurs for the crates number 4 and 5, where the PPM in slot 5 in each of these crates receives and processes trigger-tower signals from the electromagnetic FCAL1 layer.

A.2   The Mapping of the Analogue and Digital Channels on the PPM

On the PPM, the 64 input analogue differential signals are conditioned by four 16-channel PPrAnIn boards. The 64 *analogue* channels are labelled from 1 to 64, counting from the top to the bottom of the module. The connectivity between the signal pairs on the input connectors and the analogue channels on the PPM is shown in the first two columns of tables A.1, A.2 and A.3. In the first column, the connectors are indicated by the letter *C*, followed by a corresponding number from 1 to 4. As for the analogue channels, a similar top-bottom numbering convention is adopted for the front-panel connectors. The numbering of the signal pairs on each connector corresponds to the assignment shown previously in figure A.1.

After conditioning on the PPrAnIn boards, the resulting single-ended signals are reordered at the input of the PPrMCMs, to allow the bunch-crossing multiplexing algorithm on the PPrA-SICs to combine into pairs data from trigger towers with the same ϕ coordinate, as expected by the CPMs. This reordering of signals is illustrated schematically in figure A.2, for the case of one PPrMCM. The same scheme holds for the other 15 PPrMCMs. Four trigger-tower signals, labelled 1-4, are received on the PPM through four pin pairs (16-13) of a front-panel connector, and conditioned in the first four channels of a corresponding PPrAnIn board. The single-ended signals are then reordered at the input of a corresponding PPrMCM, so that the data from the

Figure A.2: Trigger-cell ordering in the PPrMCM [Han09b].
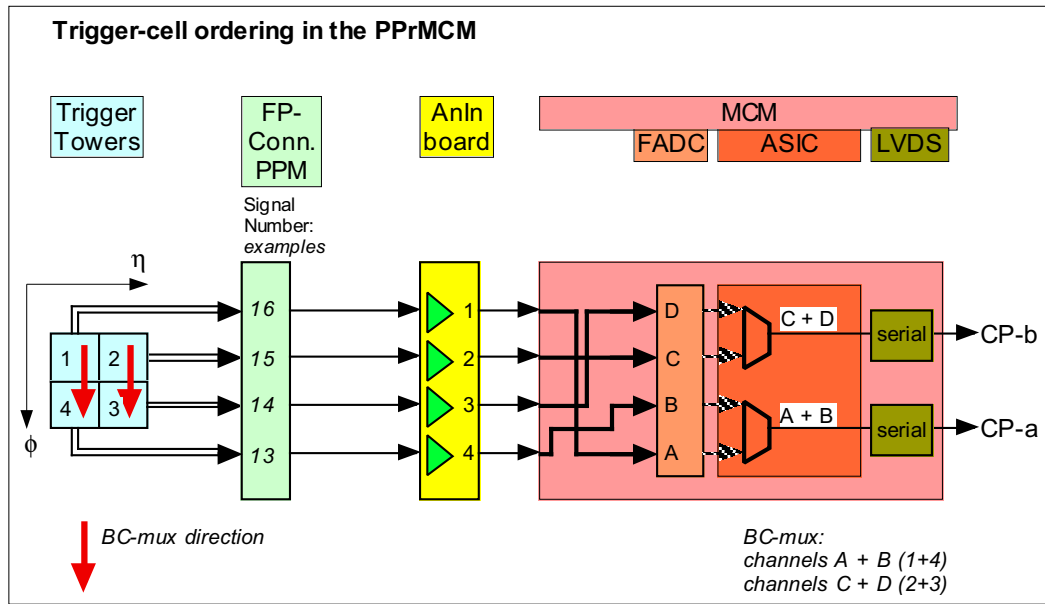
digital data from the trigger towers 1 and 4 is multiplexed in the PPrASIC into one stream to CP (CP-a), and the data from the other two trigger towers (2,3) is multiplexed into another stream (CP-b).

On each PPrMCM, the four input analogue signals are digitised by four single-channel FADCs, and the resulting digital signals are routed to the input of the 4-channel PPrASIC for further processing. There are 16 PPrMCMs on the board, which gives a total of 64 digital channels. These are labelled also 1-64, again counting from the top to the bottom of the PPM. The fourth and the fifth columns in tables A.1-A.3 describe the correlation between the analogue channels, after reordering at the input of the PPrMCMs, and the digital channels.

Each PPrASIC provides three output parallel data streams on the real-time path. Two of these streams contain bunch-crossing multiplexed $\Delta\eta$ x $\Delta\phi = 0.1$ x 0.1 energy depositions for the CP, while the third stream provides 0.2 x 0.2 energy deposition sums for the JEP. The streams are serialised and converted to LVDS form on the PPrMCM, and then sent by the LCD daughtercard to the L1Calo processors via cable assemblies which carry four LVDS signals. Each PPM is designed to provide real-time data via 10 cable assemblies to CP and 12 cable assemblies to JEP. The last two columns in tables A.1-A.3 describe the mapping of the output LVDS signals on the cable connectors. The signals are listed in the tables in conformity with their mapping to the backplane connector. Note also that the signals near the ϕ edges of the PPM are fanned out, to allow the algorithms of the L1Calo processors to investigate the boundaries of the ϕ quadrants. These signals are indicated in the upper part of table A.1 and in the lower part of table A.3. A more detailed description about the pin usage on the backplane connector and the LVDS cable connectors is given here [Mah06]. Also, the connectivity between the PPr and the L1Calo processors is described in great detail here [L1C07b].

Figure A.3: The mapping of the analogue trigger-tower signals from the ***electromagnetic*** calorimeters on the input connectors of the PPMs, for the first φ-quadrant and both η < 0 (*left*) and η > 0 (*right*).

Hadr. trigger towers, 1st phi quadrant, negative eta

PPM Analogue Input Mapping
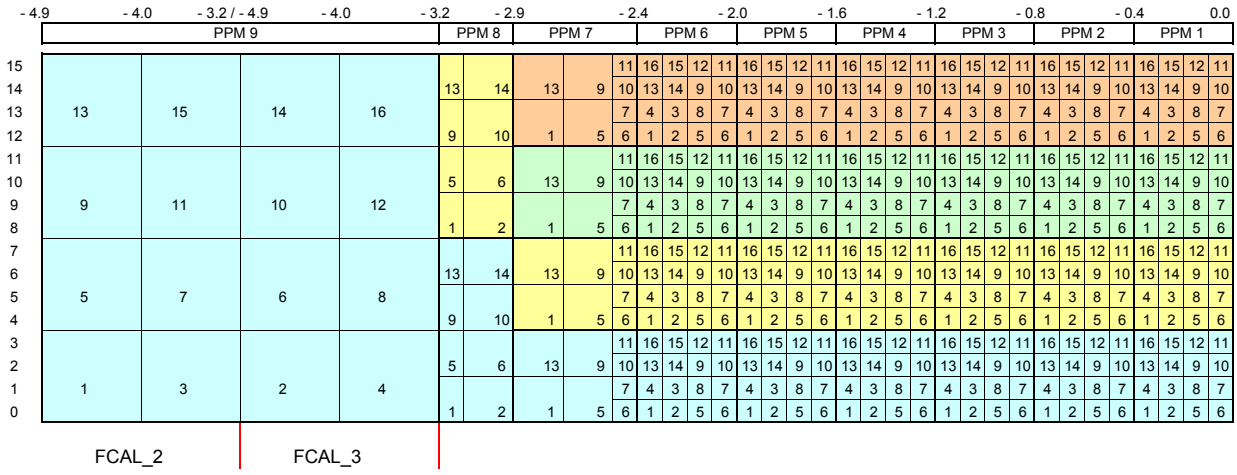
Hadr. trigger towers, 1st phi quadrant, positive eta

FCAL_2   FCAL_3

FCAL_2   FCAL_3
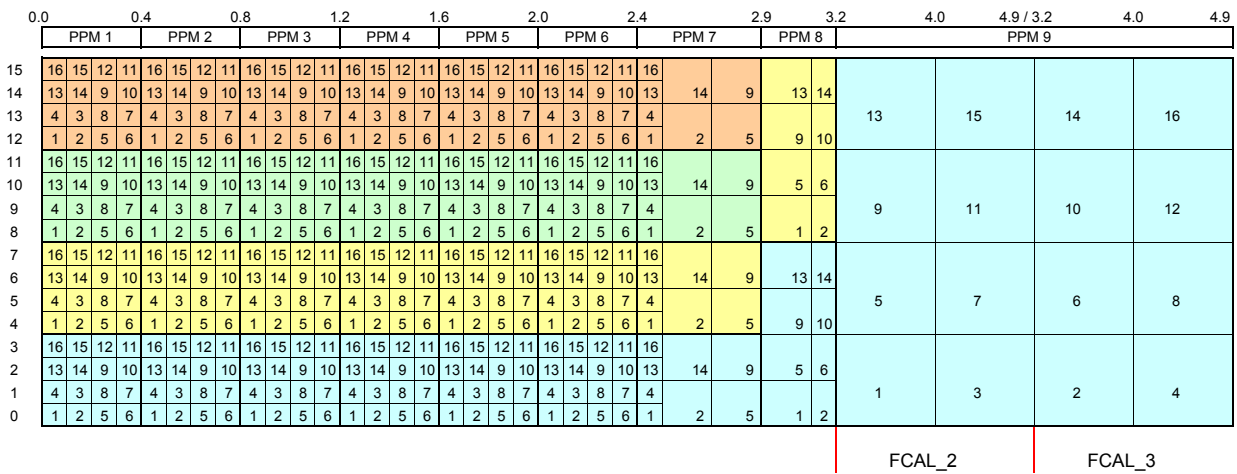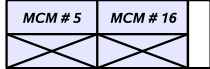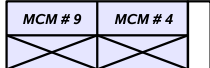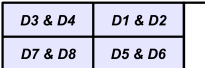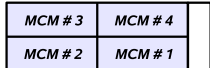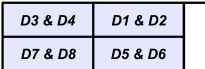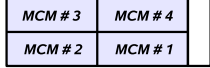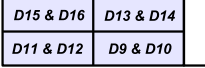
Connector 1   Connector 2   Connector 3   Connector 4

Figure A.4: The mapping of the analogue trigger-tower signals from the *hadronic* calorimeters on the input connectors of the PPMs, for the first ϕ-quadrant and both $\eta < 0$ (*left*) and $\eta > 0$ (*right*).

Figure A.5: The complete coverage of the ATLAS trigger space by the PreProcessor system.

Table A.1: Analogue and digital channel mapping on the PPM (1) (modified from [Mah06]).

Conn. Pair	Analogue ch #	MCM slot #	MCM ch map	Dig. ch #	PPM to CPM LVDS Connector	PPM to JEM LVDS Connector
						MCM # 5 / MCM # 16
						MCM # 9 / MCM # 4
						MCM # 8
					D3 & D4 / D1 & D2 / D7 & D8 / D5 & D6	MCM # 3 / MCM # 4 / MCM # 2 / MCM # 1
C1_16	A1	1	A1	D1		
C1_15	A2		A4	D2		
C1_14	A3		A2	D3		
C1_13	A4		A3	D4	D3 & D4 / D1 & D2 / D7 & D8 / D5 & D6	
C1_12	A5	2	A5	D5		
C1_11	A6		A8	D6		
C1_10	A7		A6	D7		
C1_9	A8		A7	D8		
C1_8	A9	3	A9	D9		MCM # 3 / MCM # 4 / MCM # 2 / MCM # 1
C1_7	A10		A12	D10		
C1_6	A11		A10	D11		
C1_5	A12		A11	D12	D15 & D16 / D13 & D14 / D11 & D12 / D9 & D10	
C1_4	A13	4	A13	D13		
C1_3	A14		A16	D14		
C1_2	A15		A14	D15		
C1_1	A16		A15	D16		

Table A.2: Analogue and digital channel mapping on the PPM (2) (modified from [Mah06]).

Conn. Pair	Analogue ch #	MCM slot #	MCM ch map	Dig. ch #	PPM to CPM LVDS Connector	PPM to JEM LVDS Connector
C2_16	A17	5	A17	D17		
C2_15	A18		A20	D18		
C2_14	A19		A18	D19		
C2_13	A20		A19	D20	D19 & D20 | D17 & D18 / D23 & D24 | D21 & D22	
C2_12	A21	6	A21	D21		
C2_11	A22		A24	D22		
C2_10	A23		A22	D23		
C2_9	A24		A23	D24		
C2_8	A25	7	A25	D25		MCM # 7 | MCM # 8 / MCM # 6 | MCM # 5
C2_7	A26		A28	D26		
C2_6	A27		A26	D27		
C2_5	A28		A27	D28	D31 & D32 | D29 & D30 / D27 & D28 | D25 & D26	
C2_4	A29	8	A29	D29		
C2_3	A30		A32	D30		
C2_2	A31		A30	D31		
C2_1	A32		A31	D32		
C3_16	A33	9	A33	D33		
C3_15	A34		A36	D34		
C3_14	A35		A34	D35		
C3_13	A36		A35	D36	D35 & D36 | D33 & D34 / D39 & D40 | D37 & D38	
C3_12	A37	10	A37	D37		
C3_11	A38		A40	D38		
C3_10	A39		A38	D39		
C3_9	A40		A39	D40		
C3_8	A41	11	A41	D41		MCM # 11 | MCM # 12 / MCM # 10 | MCM # 9
C3_7	A42		A44	D42		
C3_6	A43		A42	D43		
C3_5	A44		A43	D44	D47 & D48 | D45 & D46 / D43 & D44 | D41 & D42	
C3_4	A45	12	A45	D45		
C3_3	A46		A48	D46		
C3_2	A47		A46	D47		
C3_1	A48		A47	D48		

Table A.3: Analogue and digital channel mapping on the PPM (3) (modified from [Mah06]).

Conn. Pair	Analogue ch #	MCM slot #	MCM ch map	Dig. ch #	PPM to CPM LVDS Connector	PPM to JEM LVDS Connector
C4_16	A49		A49	D49		
C4_15	A50	13	A52	D50		
C4_14	A51		A50	D51		
C4_13	A52		A51	D52	D51 & D52 / D49 & D50	
C4_12	A53		A53	D53	D55 & D56 / D53 & D54	
C4_11	A54	14	A56	D54		
C4_10	A55		A54	D55		
C4_9	A56		A55	D56		MCM # 15 / MCM # 16
A57	A57		A57	D57		MCM # 14 / MCM # 13
A58	A58	15	A60	D58		
A59	A59		A58	D59		
A60	A60		A59	D60	D63 & D64 / D61 & D62	
A61	A61		A61	D61	D59 & D60 / D57 & D58	
A62	A62	16	A64	D62		
A63	A63		A62	D63		
A64	A64		A63	D64		
					D63 & D64 / D61 & D62	MCM # 15
					D59 & D60 / D57 & D58	MCM # 16
						MCM # 10 / MCM # 9
						MCM # 13 / MCM # 8
						MCM # 1 / MCM # 12

Appendix B

Readout Manager FPGA: Registers and Memory Locations

B.1 The VME Address Space for the Readout Manager

The following table gives an overview of the VME address space allocated to the ReM_FPGA. It should be mentioned that all the VME addresses given in this appendix refer only to the first 23 bits of the A32 address bus, i.e. they do not include the geographical and the crate address bits. Also, the data width is 32 bits, as defined by the VME.

   The content of each data block or single register is detailed in the next sections. A reference to the corresponding section is indicated, for each entry of the following table, in the column *Reference*.

Table B.1: Overview of the VME address space for the ReM_FPGA

VME Address	Block/Register Name	Size	Reference
On-board SRAM			
0x200000	MCM Reference	0x20000	B.2.1
0x220000	MCM Readback	0x20000	”-”
0x240000	TTCrx Reference	0x100	B.2.2
0x240100	TTCrx Readback	0x100	”-”
0x240200	*FREE SRAM SPACE*	0x1FE00	
0x260000	Reference Playback Patterns	0x40000	B.2.3
0x2A0000	*FREE SRAM SPACE*	0x60000	
0x300000	Reference PPrASIC Rates	0x200	B.2.4
0x300200	Readback PPrASIC Rates	0x200	”-”
0x300400	*FREE SRAM SPACE*	0xFC00	
0x310000	Reference PPrASIC Histograms	0x8000	B.2.5
0x318000	Readback PPrASIC Histograms	0x8000	”-”
			Continued on next page

Table B.1 – *continued from previous page*

VME Address	Block/Register Name	Size	Reference
0x320000	*FREE SRAM SPACE*	0x2E0000	
ReM Command, Control, Status and Error Registers			
0x600000	VME Spy Buffers	0x100	B.3.1
0x600100	*NOT ALLOCATED*	0x1FF00	
0x620000	SRAM-MCM Readback Flags	0x20000	B.3.2
0x640000	*NOT ALLOCATED*	0x100	
0x640100	SRAM-TTCrx Readback Flags	0x100	B.3.3
0x640200	*NOT ALLOCATED*	0xFE00	
0x650000	Rate Metering Enable	0x4	
0x650004	Rate Metering Disable	0x4	
0x650008	Histogramming Enable	0x4	
0x65000C	Histogramming Disable	0x4	
0x650010	*NOT ALLOCATED*	0x1AFEEC	
0x7FFEF8	PHOS4_Acknowledge	0x4	B.3.5
0x7FFEFC	PHOS4-DLL Status	0x4	B.3.6
0x7FFF00	AnIn_1 Low	0x4	B.3.7
0x7FFF04	AnIn_1 High	0x4	”-”
0x7FFF08	AnIn_2 Low	0x4	”-”
0x7FFF0C	AnIn_2 High	0x4	”-”
0x7FFF10	AnIn_3 Low	0x4	”-”
0x7FFF14	AnIn_3 High	0x4	”-”
0x7FFF18	AnIn_4 Low	0x4	”-”
0x7FFF1C	AnIn_4 High	0x4	”-”
0x7FFF20	*NOT ALLOCATED*	0x18	
0x7FFF38	Rate Meter Status (1)	0x4	B.3.8
0x7FFF3C	Rate Meter Status (2)	0x4	”-”
0x7FFF40	Histogramming Status (1)	0x4	B.3.9
0x7FFF44	Histogramming Status (2)	0x4	”-”
0x7FFF48	Raw Pipeline Status (1)	0x4	B.3.10
0x7FFF4C	Raw Pipeline Status (2)	0x4	”-”
0x7FFF50	BCID Pipeline Status (1)	0x4	B.3.11
0x7FFF54	BCID Pipeline Status (2)	0x4	”-”
0x7FFF58	Playback Status	0x4	B.3.12
0x7FFF5C	PHOS4_SerIntf Status	0x4	B.3.13
0x7FFF60	ROD Readout Samples	0x4	B.3.14
0x7FFF64	G-Link DAV Gap	0x4	B.3.15
0x7FFF68	Disabled ASIC Channels (1)	0x4	B.3.16
0x7FFF6C	Disabled ASIC Channels (2)	0x4	”-”
0x7FFF70	MCM_Control Register	0x4	B.3.17
		Continued on next page	

Table B.1 – *continued from previous page*

VME Address	Block/Register Name	Size	Reference
0x7FFF74	*NOT ALLOCATED*	0xC	
0x7FFF80	Local Trigger Delay	0x4	B.3.18
0x7FFF84	Local Trigger Configuration	0x4	B.3.18
0x7FFF8C	Local Counter Reset	0x4	B.3.19
0x7FFF90	*NOT ALLOCATED*	0x40	
0x7FFFD0	ReM Firmware Version	0x4	B.3.20
0x7FFFD4	ReM_Status Register (1)	0x4	B.3.21
0x7FFFD8	ReM_Control Register (1)	0x4	B.3.22
0x7FFFDC	ReM_Control Register (2)	0x4	B.3.22
0x7FFFE0	ReM_Command Register	0x4	B.3.23
0x7FFFE4	ReM_Error Register (1)	0x4	B.3.24
0x7FFFE8	ReM_Status Register (2)	0x4	B.3.21
0x7FFFEC	ReM_Error Register (2)	0x4	B.3.24
0x7FFFF0	*NOT ALLOCATED*	0x10	

B.2 Data Storage in SRAM

B.2.1 MCM Reference and Readback Blocks

The MCM Reference and Readback blocks have an identical structure and size. They are divided into 16 sub-blocks of equal size, each sub-block storing data related to a given PPrMCM (see table B.2).

The way the data is organised in each sub-block is shown in table B.3. Note that apart from PPrPHOS4 and PPrASIC data, each sub-block holds as well the corresponding PPrAnIn-DAC settings for the given PPM channels, in order to efficiently use the SRAM space. The last column of the same table describes the format in which the ReM_FPGA writes both the configuration and the readback data to SRAM. This is also the format in which the ReM_FPGA expects and delivers the data on the VME interface. The format of the PPrASIC Playback and LUT memory data represents a special case. In order to reduce the time needed to configure these memories, as well as to efficiently use the SRAM space, the data of multiple consecutive memories locations are packed into one 32-bit VME data word and stored in this form in the SRAM (see tables B.4 and B.5).

Also, note that the first column of tables B.2 and B.3 provides only an address offset. This offset has to be added to the corresponding base addresses specified in table B.1 in order to locate the given data in the VME address space.

Table B.2: Internal division of the MCM Reference and Readback blocks.

Address Offset	Sub-Block Name	Size
0x0	PPrMCM_01	0x2000
0x2000	PPrMCM_02	0x2000
0x4000	PPrMCM_03	0x2000
...	...	...
0x1C000	PPrMCM_15	0x2000
0x1E000	PPrMCM_16	0x2000

Table B.3: Organisation of data in one MCM data sub-block.

Address Offset	Field Name	Size	Data Format (32-bit)
0x0	PPrAnIn-DAC data Ch# 1-4	0x10	0x00YY00ZZ YY/ZZ = 8-bit threshold/offset
0x10	PPrPHOS4 data Ch# 1-4	0x10	0x000000YY YY = 5-bit PHOS4 delay
0x20	PPrASIC Global Regs# 1-5 (PPrASIC Ch# 1-2)	0x14	0x00000YYY YYY = 11-bit PPrASIC data
0x34	*FREE SRAM SPACE*	0xC	
0x40	PPrASIC Global Regs# 1-5 (PPrASIC Ch# 3-4)	0x14	0x00000YYY YYY = 11-bit PPrASIC data
0x54	*FREE SRAM SPACE*	0xC	
PPrASIC Channel #1			
0x60	PPrASIC Playback Memory	0x200	see table B.4
0x260	*FREE SRAM SPACE*	0x20	
0x280	PPrASIC LUT Memory	0x400	see table B.5
0x680	PPrASIC ChanRegs# 1-34 (PPrASIC Ch# 1)	0x88	0x00000YYY YYY = 11-bit PPrASIC data
0x708	*FREE SRAM SPACE*	0x158	
PPrASIC Channel #2			
0x860	Playback, LUT & channel register data	0x800	same format as for channel #1
PPrASIC Channel #3			
0x1060	Playback, LUT & channel register data	0x800	same format as for channel #1
PPrASIC Channel #4			
0x1860	Playback, LUT & channel register data	0x7A0	same format as for channel #1

Table B.4: The format in which the PPrASIC Playback data is transfered through the VME and stored in SRAM. Data related to each two consecutive playback memory locations is packed in one 32-bit data word (YYY and ZZZ are each 11 bits wide).

Address Offset	0x0YYY	0ZZZ
0x0	PlaybackMem Loc #2	PlaybackMem Loc #1
0x1	#4	#3
0x2	#6	#5
...	...	...
0x126	#254	#253
0x127	#256	#255

Table B.5: The format in which the PPrASIC LUT data is transfered through the VME and stored in SRAM. Data related to each four consecutive LUT memory locations is packed in one 32-bit data word (YY,ZZ,WW and UU are each 8 bits wide).

Address Offset	0xYY	ZZ	WW	UU
0x0	LUT Loc #4	LUT Loc #3	LUT Loc #2	LUT Loc #1
0x1	#8	#7	#6	#5
0x2	#12	#11	#10	#9
...	...	...	...	...
0x254	#1020	#1019	#1018	#1017
0x255	#1024	#1023	#1022	#1021

B.2.2   TTCrx Reference and Readback Blocks

As in the case of the MCM data blocks, the TTCrx Reference and Readback Blocks have an identical structure, so that the related register data in the two blocks are separated by a constant memory address offset. Table B.6 shows the way the data is stored in both blocks. Note that the address offset given in the first column has to be added to the corresponding VME base address of each block (see table B.1), in order to reconstruct the actual VME address allocated for each register. Also, the TTCrx register data is stored in the lower 8-bits of the corresponding SRAM locations.

Table B.6: The TTCrx registers as stored in the corresponding Reference and Readback blocks.

Addr. Offset	Reg.Nr.	Register Name
Timing Registers		
0x0	0	Fine Delay 1
0x4	1	Fine Delay 1
0x8	2	Coarse Delay
Control Register		
0xC	3	Control
0x10		*FREE SRAM SPACE*
Error Counter Registers		
0x20	8	Single Error Count<7:0>
0x24	9	Single Error Count<15:8>
0x28	10	Double Error Count<7:0>
0x2C	11	Double Error Count<15:8>
0x30		*FREE SRAM SPACE*
ID Registers		
0x40	16	ID<7:0>
0x44	17	MasterModeA<1:0>, ID<13:8>
0x48	18	MasterModeB<1:0>, I2C_ID<13:8>
Configuration Registers		
0x4C	19	Config 1
0x50	20	Config 2
0x54	21	Config 3
Status Register		
0x58	22	Status
0x5C		*FREE SRAM SPACE*
Bunch Counter Registers		
0x60	24	Bits <7:0>
0x64	25	Bits <15:8>
Event Counter Registers		
0x68	26	Bits <7:0>
0x6C	27	Bits <15:8>
0x70	28	Bits <23:16>

B.2.3   Reference Playback Patterns

This SRAM space stores reference patterns for the PPrASIC Playback memory. The space is divided into 8 compact and equally sized data blocks, so that each block stores 64 patterns, i.e. one per PPM channel. The repartition of these eight blocks is shown in table B.7. Also, the

order in which the playback patterns have to be deposited in each of the eight blocks is shown in table B.8. This is also the order in which the ReM_FPGA is reading and transferring the data to the PPrASICs.

Last but not least, the format in which the ReM_FPGA expects the patterns to be stored in SRAM is identical to the format previously presented in table B.4.

Table B.7: Internal division of the Reference Playback Patterns data block.

VME Address	Block Name		Size
0x260000	Playback Block #1		0x8000
0x268000	—"—	#2	0x8000
0x270000	—"—	#3	0x8000
0x278000	—"—	#4	0x8000
0x280000	—"—	#5	0x8000
0x288000	—"—	#6	0x8000
0x290000	—"—	#7	0x8000
0x298000	—"—	#8	0x8000

Table B.8: The ordering of the reference playback patterns in one SRAM block according to the PPrASIC channel number and to the PPM digital channel number.

PPrASIC Nr.	PPrASIC Channel Nr.	PPM Digital Channel Nr.
1	1	1
(the upper PPrASIC	2	2
on the board)	3	3
	4	4
...	...	...
16	1	61
(the lowest PPrASIC	2	62
on the board)	3	63
	4	64

B.2.4   Reference and Readback Rates

These two blocks store PPrASIC Rate Metering data and have an identical size. The Readback Rates block stores the Rate Metering data read out from the PPrASICs, in the format shown in table B.9. The ordering of data according to the PPM digital channel number and the PPrASIC number is identical to the ordering previously presented in table B.8. The Reference Rates block is meant as a storage place for Rate Metering data that reflects "good conditions". This data is loaded from VME, in a format left free to the choice of the user.

Table B.9: The format in which the PPrASIC Rate Metering data is stored in SRAM, in the Readback Rates block.

VME Address	32-bit SRAM Data	PPM Digital Channel Nr.
0x300200	12'b0, DataCounter[19:0]	1
0x300204	16'b0, TimeBlocks[15:0]	1
...	...	...
0x3003F8	12'b0, DataCounter[19:0]	64
0x3003FC	16'b0, TimeBlocks[15:0]	64

B.2.5   Reference and Readback Histograms

These two blocks store PPrASIC Histogramming data and have an identical size. The Readback Histograms block holds the Histogramming data read out from all the 16 PPrASICs. Each channel-histogram is stored in the same format in which the playback data is stored in other blocks (see table B.4), while their ordering with respect to the PPM digital channel number and the PPrASIC number is realised as shown in table B.8. The Reference Histograms block is meant as a storage place for histograms that reflects "good conditions". This data is loaded from VME, in a format left free to the choice of the user.

B.3   Control, Command, Status and Error Registers

B.3.1   The "VME Spy Buffers" Registers

A number of 64 read-only registers are provided for accessing the PPrASIC serial interface data accumulated in the VME Spy Buffers. There are 32 such memory buffers, each one storing data from a corresponding serial interface. This means that two VME registers are allocated for each buffer. The first register provides status information about the data storage operation in the respective buffer, in the format indicated in table B.11. The second register provides the actual PPrASIC data, via the lower 13 bits of the 32-bit VME data word.

No local address needs to be delivered from VME. The ReM_FPGA increments the local read pointer after each VME read operation to the second register, and simultaneously updates

the data content of the first register. The succession in which the 64 registers are mapped to VME, is shown in table B.10.

Table B.10: The mapping of the VME Spy Buffers.

VME Address	Accessed Data	PPrASIC Nrs.
0x600000	Mem. Status	PPrASIC_1, 1st Ser.Intf.
0x600004	Mem. Data	PPrASIC_1, 2nd Ser.Intf.
...	...	...
0x6000F8	Mem. Status	PPrASIC_16, 1st Ser.Intf.
0x6000FC	Mem. Data	PPrASIC_16, 2nd Ser.Intf.

Table B.11: The content of the first VME Spy Buffers register.

Bit Nr.	Bit Name	Description
0-8	Data Counter	number of 13-bit data words available in the buffer
9-11		*NOT USED*[1]
12	Overflow	"1" = the maximum buffering capacity has been exceeded
13	Underflow	"1" = VME attempts to read data, although the buffer is empty
14	Full	"1" = the maximum buffering capacity has been reached
15	Empty	"1" = no data is stored in the buffers
16-31		*NOT USED*

B.3.2 The "SRAM-MCM Readback Flags" Address Block

Each VME address of this block maps the readback flags of a corresponding memory location from the MCM Readback block. Only a read access to the readback flags is provided, their value being exclusively set by the ReM_FPGA. Upon a read request, the ReM_FPGA will pack the readback flags in the lower four bits of the 32-bit VME data, and set the remaining 28 bits to zero.

B.3.3 The "SRAM-TTCrx Readback Flags" Address Block

Each VME address of this block maps the readback flags of a corresponding memory location from the TTCrx Readback block. Only a read access to the readback flags is provided, their value being exclusively set by the ReM_FPGA. Upon a read request, the ReM_FPGA will pack

---

[1] the "NOT USED" bits are permanently set to zero.

the readback flags in the lower four bits of the 32-bit VME data, and set the remaining 28 bits to zero.

B.3.4 The Rate Metering & Histogramming Enable/Disable Registers

These registers allow to simultaneously enable or disable the Rate Metering and Histogramming operations in all 64 PPM channels. The registers are implemented as commands, thus the input VME data is not used. When read, the data loaded to the VMEbus contains 32 bits zero.

B.3.5 The "PHOS4_Acknowledge" Status Register

This register indicates whether the accessed PPrPHOS4s have acknowledged the data transferred by the ReM_FPGA over the I^2C bus. As described in section 6.4.3, during the configuration of a PPrPHOS4, the ReM_FPGA transfers first the corresponding 8-bit *slave address*, and then the 8-bit configuration data. After each transferred byte, the addressed PPrPHOS4 responds with a low-active bit, if the input was acknowledged, or with a high-active bit if otherwise. These bits are mapped by the ReM_FPGA in the PHOS4_Acknowledge status register, in the order indicated in table B.12. The default value of the bits is zero.

Table B.12: The PHOS4_Acknowledge Status Register.

Bit Nr.	Bit Name	Description
0	Addr_Ackn_1	"0" = I^2C address acknowledged (PHOS4_#1)
1	Data_Ackn_1	"0" = I^2C data acknowledged (PHOS4_#1)
...	...	...
30	Addr_Ackn_16	"0" = I^2C address acknowledged (PHOS4_#16)
31	Data_Ackn_16	"0" = I^2C data acknowledged (PHOS4_#16)

B.3.6 The "PHOS4-DLL Status" Register

This register gathers the FrequencyLost signals, which indicate the operational status of the 16 PPrPHOS4s, and the LOCKED output from the internal DLLs of the ReM_FPGA. The proper operation of the PPrPHOS4s is flagged by a logic "0", while the lock state of the internal DLLs is flagged by a logic "1".

Table B.13: The PHOS4-DLL Status Register.

Bit Nr.	Bit Name	Description
0	MCM1_Phos4Err	*FrequencyLost* signal for the PPrPHOS4 on MCM#1
1	MCM2_Phos4Err	— " — MCM#2
		Continued on next page

Table B.13 – *continued from previous page*

Bit Nr.	Bit Name	Description	
2	MCM3_Phos4Err	—''—	MCM#3
3	MCM4_Phos4Err	—''—	MCM#4
4	MCM5_Phos4Err	—''—	MCM#5
5	MCM6_Phos4Err	—''—	MCM#6
6	MCM7_Phos4Err	—''—	MCM#7
7	MCM8_Phos4Err	—''—	MCM#8
8	MCM9_Phos4Err	—''—	MCM#9
9	MCM10_Phos4Err	—''—	MCM#10
10	MCM11_Phos4Err	—''—	MCM#11
11	MCM12_Phos4Err	—''—	MCM#12
12	MCM13_Phos4Err	—''—	MCM#13
13	MCM14_Phos4Err	—''—	MCM#14
14	MCM15_Phos4Err	—''—	MCM#15
15	MCM16_Phos4Err	—''—	MCM#16
16	SysClk_DLL_Lock	The LOCKED output of the corresponding DLL	
17	GLinkClk_DLL_Lock	—''—	
18	McmClk_DLL_Lock	—''—	
19	McmSerClk_DLL_Lock	—''—	
20	SRamClk_DLL_Lock	—''—	
21-31		*NOT USED*	

B.3.7 The ''DAC Full-Buffered Mode'' Registers

Eight 32-bit write-only registers are provided for the activation of the full-buffered mode on the PPrAnIn-DACs. Each register serves two DACs located on the same PPrAnIn board. The group of two DACs and the PPrAnIn board are indicated in the names of the registers. The former is indicated by a suffix, i.e. _High or _Low, while the latter is indicated by a number, from 1 to 4, where 1 refers to the PPrAnIn located in the upper slot on the PPM, while 4 refers to the PPrAnIn located in the lower slot.

In order to enable the full-buffered mode simultaneously in both DACs, the following 32-bit data word has to be delivered from VME: 32'b00ff00ff. The upper 16-bits provide the 8-bit address and the 8-bit data for the first DAC, while the lower 16-bits describe the similar data for the second DAC. In case the operational mode is intended to be enabled in only one DAC, then the corresponding 16 bits should all be zeroed.

The 64 data bits, which the ReM_FPGA transfers during one write operation to the PPrAnIn-DACs, are obtained by merging the 32-bit data provided by the *High* and the *Low* registers related to the same PPrAnIn. The transfer of the 64 data bits from the ReM_FPGA to the indicated PPrAnIn board is initiated only by a VME write operation to the *High* register. Which means that the related *Low* register has to be written in advance. If this latter information is not pro-

vided, then the ReM_FPGA will send 32 bits zero to the *Low* DACs, which indicate that the input data should be ignored.

B.3.8 The "Rate Metering Status" Registers

These two registers gather the "Rate Available" bits provided by the PPrASICs via the *ReadbackEmpty* data words. The format in which the bits are grouped in these registers is shown in table B.14. Also, the relationship between the PPM digital channel numbers and the PPrASIC channel number is identical with the one previously presented in table B.8. The default value of these bits is 0, i.e. "rates are not available".

Table B.14: The grouping of the "Rate Available" bits in the two "Rate Metering Status" registers according to the PPM digital channel number.

BitNr	First Register	Second Register
0	PPM digital channel #1	PPM digital channel #33
...	...	...
31	PPM digital channel #32	PPM digital channel #64

B.3.9 The "Histogramming Status" Registers

These two registers gather the "Histogram Available" bits provided by the PPrASICs via the *ReadbackEmpty* data words. The format in which the "Histogram Available" bits are grouped in the two VME status registers is identical with the format shown in table B.14. The default value of these bits is 0, i.e. "histograms are not available".

B.3.10 The "Raw Pipeline Status" Registers

These two registers gather the "FADC Pipeline Stopped" bits provided by the PPrASICs via the *ReadbackEmpty* data words. The format in which these bits are grouped in the two VME status registers is identical with the format shown in table B.14. The default value of these bits is 0, i.e. "the raw pipeline memories are active".

B.3.11 The "BCID-LUT Pipeline Status" Registers

These two registers gather the "BCID-LUT Pipeline Stopped" bits provided by the PPrASICs via the *ReadbackEmpty* data words. The format in which these bits are grouped in the two VME status registers is identical with the format shown in table B.14. The default value of these bits is 0, i.e. "the BCID-LUT pipeline memories are active".

B.3.12 The "Playback Status" Register

This register gathers the "Playback Mode Active" bits provided by the PPrASICs via the *ReadbackEmpty* data words. The format in which these bits are grouped is shown in table B.15. The default value of these bits is 0, i.e. "playback mode is disabled".

Table B.15: The grouping of the "Playback Mode Active" bits in the "Playback Status" register.

BitNr	PPrASIC Channel Nr.	Ser.Intf Nr.
0	1	1
1	1	2
...	...	...
30	16	1
31	16	2

B.3.13 The "PHOS4_SerIntf Status" Register

This register gathers the "Frequency Lost" bits provided by the PPrASICs via the *ReadbackEmpty* data words. These bits represent copies of the status bits that bear the same name, and which are delivered by the PPrASICs via single lines (see also section B.3.6). The format in which the "Frequency Lost" bits are grouped in the current register is identical with the format previously presented in table B.15. The default value of these bits is 0, i.e. "no PPrPHOS4 error".

B.3.14 The "ROD Readout Samples" Register

Via this register the ReM_FPGA is indicated the number of FADC and BCID-LUT samples which the PPrASICs are configured to read out. Currently, the ReM_FPGA offers support for six combinations of readout samples, as listed in table B.17. The choice for one of these readout settings is made via the least three significant bits of the register (see table B.16). Note that '5+1' is the default operational mode. Also, if the input configuration data is *invalid* the ReM_FPGA will consider the same mode.

The register is also readable. Apart from the configuration data, the ReM_FPGA provides to VME the numbers of FADC and BCID-LUT samples selected by a previous configuration operation. If no selection was previously done, then the ReM_FPGA provides the corresponding default values.

Finally, it is recommended to reset the ReM_FPGA before selecting a different combination of readout samples. As mentioned in section 6.5.3, the storage of the PPrASIC in the internal memory buffers is organised according to the chosen readout settings. A change, for example, from the '5+1' mode to the '11+5' mode will also change the number of memory blocks allocated for storing the event data in each buffer. If the system was previously in use, the write

and read pointers of the memory buffers may be set to values that conflict with the new internal organisation of the memory. This can be avoided by setting the write and read pointers back to zero via a VME reset.

Table B.16: The bit fields of the ROD Readout Samples register.

Bit Nr.	Field Name	Access
0-2	Configuration Data	read/write
3	NOT USED	
4-6	$N_{BCID-LUT}$	read-only
7	NOT USED	
8-11	N_{FADC}	read-only
12-31	NOT USED	

Table B.17: The expected VME configuration data for selecting one combination of readout samples.

Configuration Data	$(N_{FADC} + N_{BCID-LUT})$
3'b000	3+1
3'b001	**5+1**
3'b010	7+1
3'b011	9+3
3'b100	11+5
3'b101	15+1
3'b110	*5+1*
3'b111	*5+1*

B.3.15 The "G-Link DAV Gap" Register

Due to processing issues, currently the ReM_FPGA needs six clock ticks before transmitting the next event data to the RGTM-O, if this is already available in the local memory buffers. Via the "G-Link Dav Gap" register this timing can be additionally increased by up to 15 clock ticks. The delay applies to all serial frames. Also, the default value of the register is zero, i.e. no additional delay.

B.3.16 The "Disabled ASIC Channels" Registers

Two 32-bit VME registers are provided for flagging the disabled PPrASIC channels in the read-out stream to RGTM-O. Each bit of each register corresponds to one digital PPM channel, i.e. 64 bits covering 64 PPM channels. The first register is dedicated to the first 32 channels. The mapping of the respective channels is realised in the increasing order of the channel and bit numbers, such that the least significant bit maps the PPM digital channel 1, while the most significant bit maps the PPM digital channel 32. Correspondingly, the second register is dedicated to the other 32 channels, the least and the most significant bits mapping the digital channels 33 and 64. The default value of all 64 bits is zero, indicating that the channel is available.

B.3.17 The MCM_Control Register

This register is used for generating global commands for the PPrASICs and the LVDS transmitters. All bits are readable from VME, but only the lower three bits are writable. The other two bits are permanently set to constant values. Also, the default value of the first three bits is zero.

Table B.18: The MCM_Control Register. The numbers given in parentheses indicate the value that has to be loaded to initiate the respective command.

Bit Nr.	Bit Name	Description
0	VMESyncPlayback	Start (1) synchronously the playback data in all 64 channels
1	VMELvdsSync	Start (1) or Stop (0) sending LVDS synchronisation patterns
2	MCMLvdsTClkRF	Select the rising (1) or falling (0) edge for strobing the input data to LVDS transmitters
3	MCMLvdsDEn	Enable the output of the LVDS transmitters. The bit is permanently set to 1 (true)
4	MCMSyncReadout	The feature is not used, therefore the bit is permanently set to 0 (false)
5-31		*NOT USED*

B.3.18 The Local Trigger Registers

When the PPM is operated on a test set-up that is not equipped with a TTC system, the ReM_FPGA can be instructed to generate local trigger signals [Sch09]. Two modes are provided for generating a local trigger, both being configurable via two R/W VME registers, *Local Trigger Configuration* and *Local Trigger Delay* (see tables B.20 and B.19).

In the first mode, the ReM_FPGA generates local L1As according to a preloaded configuration. The user can choose the number of L1As to be generated (see *L1A_Pulses*) and the delay between each two consecutive L1As (*Delay_Next_L1A*). Additionally, the user has the possibility

to generate one trigger pulse for an external waveform generator, and synchronise the first L1A (*Delay_First_L1A*) with the arrival on the PPM of the signal produced by the external device.

In the second mode, the ReM_FPGA generates a local trigger according to the logic value of the four ExtBCID signals received from the PPrAnIn boards (see *The External BCID Signals* in section 6.2.7). The user can select which of the four signals should be used by the ReM_FPGA (*ExtBCID_Mask*). A local L1A will then be generated each time one of the selected signal has a logic value of "1". Since the L1A pulse must be one clock period wide, the ReM_FPGA imposes a *dead time* on the following clock event, independent on the logic value of the ExtBCID signals.

When the generation of local triggers is enabled (*Enable_L1A*), the ReM_FPGA asserts a *LocalTrigger_Busy_Status* bit in the first VME status register to indicate the progress of the operation (see section B.3.21). This bit should be polled by the controlling software before sending a similar request to the ReM_FPGA. Also, as long as the ReM_FPGA is configured to operated in *DAQ_Mode*, any write request to the local trigger registers is denied (see sections 6.4.6 and B.3.22). This refusal is flagged by corresponding bits in one VME error register (see section B.3.24).

Table B.19: The Local Trigger Configuration Register.

Bit Nr.	Bit Name	Description
0-7	L1A_Pulses	number of L1As to be generated
		(default is 1, when all bits are set to zero)
8-11	ExtPulse_Width	width (in clock ticks) of the external trigger pulse
12-15	ExtBCID_Mask	selects the ExtBCID signals that will be used for
		generating a local trigger signal
		(trigger mode is disabled if all bits are set to zero)
16-31	Delay_Next_L1A	delay (in clock ticks) for the following L1A
		(default: 1 clock tick)

Table B.20: The Local Trigger Delay Register.

Bit Nr.	Bit Name	Description
0-14	Delay_First_L1A	delay (in clock ticks) for the first L1A
15	Enable_L1A	start generating L1As
		(1 - enable, 0 - disable)
16-30	Delay_ExtTrigger	delay (in clock ticks) for the external trigger pulse
31	Enable_ExtTrigger	start generating external trigger
		(1 - enable, 0 - disable)

B.3.19   The "Local Counter Reset" Register

The ReM_FPGA can also be instructed to generate local bunch-crossing and event counter reset signals. For this purpose one 2-bit wide VME register is provided (see table B.21). The software can request the ReM_FPGA to generate either one of the two signals at a time, or both signals simultaneously, by writing a logic "1" in the corresponding bit fields. After the signals are generated, the ReM_FPGA sets back the bit fields to a logic "0".

Table B.21: The Local Counter Reset Register.

Bit Nr.	Bit Name	Description
0	Local_BcCntRst	generate reset signal for the bunch-crossing counters
1	Local_EvCntRst	generate reset signal for the event counters

B.3.20   The "Firmware Version" Register

This 32-bit register is read-only. It provides a *main version* number, via the upper 16 bits, and a *sub-version* number, via the lower 16 bits.

B.3.21   The ReM_Status Registers

The ReM_FPGA provides two read-only VME status registers, which gather corresponding information from different internal functional modules or from external devices. The bit fields of these registers are presented in the following two tables, (B.22 and B.23).

Table B.22: The first ReM_Status Register.

Bit Nr.	Bit Name	Description
0	DAQ_Mode	"1" = DAQ_Mode is enabled
1	PPM_Load	"1" = configuration mode is active (i.e. DAQ_Mode is disabled)
2	SRamInit_Busy	"1" = busy initialising the SRAM Readback blocks
3	VmeSRamRead_Status	"1" = busy transferring data from SRAM to VME
4	VmeSRamWrite_Status	"1" = busy writing data from VME to SRAM
5	RdbkSRamWrite_Status	"1" = busy writing readback data to SRAM
6	Asic_RIP	"1" = busy sending readback commands to PPrASIC
7	Asic_WIP	"1" = busy writing configuration data to PPrASIC
8	GlobalReadbackActive	"1" = global "readback in progress" indicator
9	AsicCfgReadbackActive	"1" = busy reading back the PPrASIC registers
		Continued on next page

Table B.22 – *continued from previous page*

Bit Nr.	Bit Name	Description
10	TTCrxCfgReadbackActive	"1" = busy reading back the TTCrx registers
11	RateReadbackActive	"1" = busy collecting the PPrASIC rates
12	HistoReadbackActive	"1" = busy collecting the PPrASIC histograms
13	RateMeterIsEnabled	"1" = Rate Metering has been enabled from VME in all PPrASICs "0" = operation has been disabled or not at all enabled
14	HistogrammingIsEnabled	"1" = Histogramming has been enabled from VME in all PPrASICs "0" = operation has been disabled or not at all enabled
15	LocalTrigger_Busy_Status	"1" = LocalTrigger module is busy generating trigger pulses
16	MCMLvdsSync	"1" = LVDS sync pattern is enabled
17	MCMTclkRF	LVDS transmitters latch the input PPrASIC data on the rising (1) or the falling (0) edge
18	LINKRDY	"1" = RGTM-O transmitter chip is ready to send data
19	Tx_Fault	"1" = faulty operation of the RGTM-O's optical transmitter, or the RGTM-O device is not mounted
20	Empty_InputRoFifos	"1" = the local readout buffers are empty
21	Full_InputRoFifos	"1" = the local readout buffers are full
22	RodEventsNotBuffered	"1" = the local readout buffers are in overflow and further writing is denied until the buffers are empty again
23	AnIn1_Spi_WIP	"1" = busy writing data to the 1st PPrAnIn board
24	AnIn2_Spi_WIP	"-" 2nd PPrAnIn board
25	AnIn3_Spi_WIP	"-" 3rd PPrAnIn board
26	AnIn4_Spi_WIP	"-" 4th PPrAnIn board
27	PHOS4_WIP	"1" = busy writing data to a PPrPHOS4
28	TTCrx_WIP	"1" = busy writing data to the TTCrx
29	TTCrx_ClockStatus_1	S1 signal (TTCReady) from TTCdec "1" = TTCrx' PLL has locked
30	TTCrx_ClockStatus_2	S2 signal from TTCdec
		"1" = TTCrx selected as clock source "0" = XTAL selected as clock source
31	TTCrx_PorD	*Protected\Debug* mode (TTCdec clock selection) always set to "0" (= *Debug*)

Table B.23: The second ReM_Status Register.

Bit Nr.	Bit Name	Description
0	PlaybackSRamRead_Status	”1” = AsicPlaybackLoader reads the playback data stored in SRAM
1	PlaybackSRamWrite_Status	”1” = AsicPlaybackLoader writes the playback data back to SRAM
2	PlaybackLoadingActive	”1” = busy transferring playback data from SRAM to the PPrASICs
3	TTCrx_LockLost	”1” = TTCrx has lost the lock at least once ”0” = otherwise
4	TTCrx_I2C_ptr_WR_Ackn	”0” = TTCrx has acknowledged the I2C_pointer address (WRITE)
5	TTCrx_RegNr_Ackn	”0” = — ” — the register number
6	TTCrx_I2C_data_Ackn	”0” = — ” — the I2C_data address
7	TTCrx_Data_Ackn	”0” = — ” — the 8-bit config. data
8	TTCrx_I2C_ptr_RD_Ackn	”0” = TTCrx has acknowledged the I2C_pointer address (READ)
4-31		*NOT USED*

B.3.22 The ReM_Control Register

This register provides six bits to configure the operation of the ReM_FPGA, and one bit to generate a reset for the TTCrx chip. All bits have a default value of ”0”, i.e. *disabled*.

A particularity of this register is the way it is mapped to VME. As it can be seen in table B.1, the register is mapped twice, at two consecutive VME addresses. The first address is used only for setting the DAQ_Mode bit, while the second one is used for setting the other bits. The reason for this implementation is to prevent the DAQ_Mode to be enabled by accident. Also, the register is readable via both VME addresses.

Table B.24: The ReM_Control Register.

Bit Nr.	Bit Name	Description
0	DAQ_Mode	denies VME configuration requests for ReM_FPGA and on-board devices
1	ForcedConfigReadback	overwrites the readback flags of the data stored in the MCM Readback block
2	RoBuffReset	resets the VME Spy buffers (data is lost)
3	RoBuffEnable	enables the data storage in the VME Spy buffers
4	SingleEvent	only the first arriving event data block is buffered
		Continued on next page

Table B.24 – *continued from previous page*

Bit Nr.	Bit Name	Description
5	SpyBehindEvent	buffer the first arriving event data block, and then all the following data words, until memory becomes full
6	SpySerialInterface	buffer each data word received over the serial interface until the memory becomes full
7	TTCrxResetBar	reset signal for the TTCrx (it is inverted at the output stage)
7-31		*NOT USED*

B.3.23   The ReM_Command Register

The register provides a set of ten VME commands for the ReM_FPGA, as shown in table B.25. The first two commands determine the ReM_FPGA to collect the PPrASIC Rate Metering and Histogramming data from all 64 PPM channels, and place it in the SRAM. The other eight commands are dedicated to the loading of reference playback patterns from SRAM to the PPrASICs, so that each command indicates the ReM_FPGA the playback block that has to be read (see also section B.2.3).

The commands can only be addressed individually. The last column of table B.25 indicates the expected VME data for each command. Note that in case the input VME data has a different content, no command will be executed. The situation will then be flagged via the *InvalidRem-Command* bit, in the second VME error register (see table B.27), until the next valid command. Additionally, the execution of a command is refused when a process initiated by a previous command is still in progress, or when the ReM_FPGA is reading back configuration data from the PPrASICs or the TTCrx. This is because the SRAM is a single-ported memory, and thus only one process can access it at a time. Finally, the execution of the playback commands is refused if the DAQ_Mode is activated (see section 6.4.6). All these situations are flagged by dedicated bits in both VME error registers (see section B.3.24).

The command register is also readable. The data which is provided upon a VME read request is either the previous data loaded to this register or 10 bits zero, if no command was addressed after a VME reset or a start up of the system.

Table B.25: The ReM_Command Register.

Bit Nr.	Bit Name	VME data
0	CollectAsicRates	0x1
1	CollectAsicHistos	0x2
2	LoadPlaybackBlock_1	0x4
3	LoadPlaybackBlock_2	0x8
4	LoadPlaybackBlock_3	0x10
Continued on next page		

Table B.25 – *continued from previous page*

Bit Nr.	Bit Name	VME data
5	LoadPlaybackBlock_4	0x20
6	LoadPlaybackBlock_5	0x40
7	LoadPlaybackBlock_6	0x80
8	LoadPlaybackBlock_7	0x100
9	LoadPlaybackBlock_8	0x200
10-31	*NOT ALLOCATED*	

B.3.24 The "ReM_Error" Registers

These registers gathers error bits generated by various functional blocks of the ReM_FPGA. The vast majority of these bits are generated by the VmeManager, which denies the VME requests in four cases:

- when the DAQ_Mode is enabled all configuration requests for the real-time and readout paths are denied. The error bits related to these refusals are labelled in with the suffix "_Daq";

- when a readback operation is in progress, the access to the devices that are read back and to the SRAM is denied. The corresponding error bits are labelled with the suffix "_Rdbk";

- when ReM_FPGA is loading playback patterns from SRAM to the PPrASICs, any read or write access to these devices is denied. The corresponding error bits are labelled with the suffix "_LdPb";

- when a write access for a non-user accessible TTCrx register is received. Only one error bit is produced in this case, Denied_TTCrx_UnReg;

The other error bits are generated by the logic of the ReM_Command register and by the logic that handles the readback operations from PPrASIC and TTCrx. In either case the bits flag an invalid request addressed from VME.

The default logic value of all bits in both register is "0".

Table B.26: The first ReM_Error Register.

Bit Nr.	Bit Name	Description
0	Denied_AsicWrite_Daq	write access to PPrASIC registers is denied
1	Denied_AsicWrite_Rdbk	"-"
2	Denied_RateEnable_Rdbk	"enable" the Rate Metering operation is denied
3	Denied_RateDisable_Rdbk	"disable" the Rate Metering operation is denied
		Continued on next page

Table B.26 – *continued from previous page*

Bit Nr.	Bit Name	Description
4	Denied_CollectRates_Rdbk	readback of rates from PPrASICs is denied
5	Denied_HistoEnable_Rdbk	"enable" the Histogramming operation is denied
6	Denied_HistoDisable_Rdbk	"disable" the Histogramming operation is denied
7	Denied_CollectHistos_Rdbk	readback of histograms from PPrASICs is denied
8	Denied_SRamDataRead_Rdbk	read access to SRAM is denied
9	Denied_SRamDataWrite_Daq	write access to SRAM is denied
10	Denied_LocalTrigDelay_Daq	write access to LocalTriggerDelay register is denied
11	Denied_LocalTrigCfg_Daq	write access to LocalTriggerConfiguration register is denied
12	Denied_LocalCtrRst_Daq	write access to LocalCounterReset register is denied
13	Denied_RodRoSamples_Daq	configuration of the readout samples is denied
14	Denied_GLinkDavGap_Daq	configuration of the DAV gap is denied
15	Denied_AsicChansDis1_Daq	configuration of the disabled PPrASIC channels is denied
16	Denied_AsicChansDis2_Daq	"-"
17	Denied_AnIn_Daq	write access to PPrAnIn DACs is denied
18	Denied_AnIn_Rdbk	"-"
19	Denied_Phos4_Daq	write access to PPrPHOS4s is denied
20	Denied_Phos4_Rdbk	"-"
21	Denied_McmCtrl_Daq	write access to MCM_Control register is denied
22	Denied_TTCrx_Daq	write access to TTCrx registers is denied
23	Denied_TTCrx_Rdbk	"-"
24	Denied_TTCrx_UnReg	write request for non-user accessible TTCrx register
25	InvalidAsicCfgRdbkRequest	readback request to an invalid PPrASIC address
26	InvalidTTCrxCfgRdbkRequest	readback request for a non-user accessible register
27-31		*NOT USED*

Table B.27: The second ReM_Error Register.

Bit Nr.	Bit Name	Description
0	Denied_SRamDataRead_LdPb	read access to SRAM is denied
1	Denied_SRamDataWrite_Rdbk	write access to SRAM is denied
		Continued on next page

Table B.27 – *continued from previous page*

Bit Nr.	Bit Name	Description
2	Denied_SRamDataWrite_LdPb	"-"
3	Denied_AsicWrite_LdPb	write access to PPrASIC registers is denied
4	Denied_RateEnable_LdPb	"enable" the Rate Metering operation is denied
5	Denied_RateDisable_LdPb	"disable" the Rate Metering operation is denied
6	Denied_CollectRates_LdPb	readback of rates from PPrASICs is denied
7	Denied_HistoEnable_LdPb	"enable" the Histogramming operation is denied
8	Denied_HistoDisable_LdPb	"disable" the Histogramming operation is denied
9	Denied_CollectHistos_LdPb	readback of histograms from PPrASICs is denied
10	Denied_LoadPB_Daq	loading of playback patterns from SRAM to PPrASICs is denied
11	Denied_LoadPB_Rdbk	"-"
12	Denied_LoadPB_LdPb	"-"
13	InvalidRemCommand	invalid command received from VME
14	Denied_AnIn_LdPb	write access to PPrAnIn DACs is denied
15	Denied_Phos4_LdPb	write access to PPrPHOS4s is denied
16	Denied_TTCrx_LdPb	write access to TTCrx registers is denied
17-31		*NOT USED*

List of Figures

List of Tables

List of Acronyms

Acronym	Explanation
ALICE	A Large Ion Collider Experiment
ATLAS	A Toroidal LHC AparatuS
ASIC	Application-Specific Integrated Circuit
BCID	Bunch-Crossing Identification
BCR	Bunch Counter Reset
BT	Barrel Toroid
BRAM	Block RAM (FPGA)
BUFG	Internal Clock Buffer (FPGA)
CAN	Controller-Area Network
CDF	Collider Detector at Fermilab
CERN	European Organization for Nuclear Reasearch
CL	Confidence Level
CP	Cluster Processor system
CPM	Cluster Processor Module
CPU	Central Processing Unit
CPLD	Complex Programmable Logic Device
CMC	Common Mezzanine Card
CMC_Mux	CMC LVDS Multiplexer Card
CMC_Rx	CMC LVDS Receiver Card
CMM	Common Merger Module
CMOS	Complementary Metal-Oxide-Semiconductor
CMS	Compact Muon Solenoid
CSC	Cathode Strip Chambers
CTP	Central Trigger Processor
DØ	DZero experiment
DAC	Digital-to-Analogue Converter
DAQ	Data Acquisition system
DAV	Data Available
DC	Direct Current
DCS	Detector Control System

continued from previous page

Acronym	Explanation
DSS	Detector Safety System
DESY	Deutsches Elektronen Synchrotron (Ger.)
DFM	Data Flow Manager
DLL	Delay-Locked Loop
EB	Event Builder
ECR	Event Counter Reset
ECT	End-Cap Toroid
EF	Event Filter
EMB	Electromagnetic Barrel Calorimeter
EMEC	Electromagnetic End-Cap Calorimeter
FADC	Flash Analogue-to-Digital Converter
FCAL	Forward Calorimeter
FE	Front-End
FIFO	First-In First-Out
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FWHM	Full-Width at Half Maximum
GCLK	Global Clock Line
GCLKIOB	Global Clock I/O Buffer
HDL	Hardware Description Language
HEC	Hadronic End-Cap Calorimeter
HERA	Hadron-Elektron-Ringanlage (Ger.)
HLT	High-Level Trigger
I^2C	Inter-Integrated Circuit
IBUF	Internal Buffer (FPGA)
IBUFG	Dedicated Clock Input Pin for DLLs (FPGA)
ID	Innner Detector
IRQ	Interrupt Request
ISE	Xilinx Integrated Software Environment
JEM	Jet/Energy-sum Processor Module
JEP	Jet/Energy-sum Processor system
L1	Level-1 Trigger
L1A	Level-1 Accept Signal
L1Calo	Level-1 Calorimeter Trigger
L1Muon	Level-1 Muon Trigger
L2	Level-2 Trigger
L2SV	Level-2 Supervisor
L2PU	Level-2 Processing Unit
LAr	Liquid Argon

Acronym	Explanation
LED	Light-Emitting Diode
LEP	Large Electron Positron Collider
LINAC	Liniar Accelerator
LHC	Large Hadron Collider
LHCb	LHC beauty Experiment
LHCf	LHC forward Experiment
LSB	Least Significant Bit
LUT	Look-Up Table
LVDS	Low-Voltage Differential Signalling
MCM	Multi-Chip Module
MDT	Monitored Drift Chambers
MSB	Most Significant Bit
OBUF	Output Buffer (FPGA)
OBUFT	Tri-state Output Buffer (FPGA)
PC	Personal Computer
PCB	Printed Circuit Board
PCI	Peripheral Component Interconnect
PHOS4	Delay Chip
PLL	Phase-Locked Loop
PMT	Photomultiplier
PPM	Pre-Processor Module
PPr	Pre-Processor System
PPrAnIn	Pre-Processor Analogue Input Board
PPrASIC	Pre-Processor ASIC
PPrMCM	Pre-Processor MCM
PPrPHOS4	Pre-Processor PHOS4
PS	Proton Synchrotron
PSB	Proton Synchrotron Booster
QCD	Quantum Chromodynamics
RAM	Random Access Memory
ReM_FPGA	Readout Manager FPGA
RGTM-O	Rear G-Link Transmission Module - Optical Tx
ROB	Readout Buffer
ROD	Readout Driver
ROS	Readout System
RoI	Regions of Interest
RoIB	RoI Builder
ROC	Readout Controller
RPC	Resistive Plate Chambers

continued from previous page

Acronym	Explanation
RPPP	Receiver to PreProcessor Patch Pannels
RST	Reset Signal
RTC	Readout Transfer Card
SBC	Single Board Computer
SCT	Semiconductor Tracker
SFI	Sub-Farm Input
SFO	Sub-Farm Output
SM	Standard Model
SPI	Serial Peripheral Interface
SPS	Super Proton Synchrotron
SRAM	Static Random Access Memory
SU	Super Unitary Group
TCM	Timing Control Module
TCPP	Tile Calorimeter Patch Panels
TDAQ	Trigger and Data Acquisition system
TGC	Thin Gap Chambers
TOTEM	Total Cross Section, Elastic Scattering and Diffraction Dissociation Experiment
TRT	Transition Radiation Tracker
TTC	Timing, Trigger and Control
TTCex	TTC Encoder/Transmitter
TTCdec	TTC Decoder Card
TTCrx	TTC Receiver Chip
UA	Underground Area experiments
URU	Universal Receiver Unit
USA15	Main ATLAS Underground Electronics Cavern
UX15	ATLAS Underground Experimental Cavern
VGA	Variable-Gain Amplifier
VME	Versa Module Eurocard
VIPA	VME International Physics Association
XTAL	Crystal Oscillator

Bibliography

[Ach07]   R. Achenbach et al., *First Measurements with the ATLAS Level-1 Calorimeter Trigger PreProcessor System*, Proc. Topical Workshop on Electronics for Particle Physics, Prague 2007, pp. 222–226.

[Agi]   Agilent Technologies, *Low Cost Gigabit Rate Transmit/Receive Chip Set with TTL I/Os*, Technical Data.

[ALE03]   ALEPH, DELPHI, L3, and OPAL Collaborations, the LEP Working Group for Higgs Boson Searches, *Search for the Standard Model Higgs Boson at LEP*, Phys.Lett. **B656** (2003), 61.

[ALE08]   ALEPH, CDF, DØ, DELPHI, L3, OPAL, SLD Collaborations, the LEP Electroweak Working Group, the Tevatron Electroweak Working Group and the SLD electroweak and heavy flavour groups, *Precision Electroweak Measurements and Constraints on the Standard Model*, arXiv:0811.4682v1.

[ALI08]   ALICE Collaboration, *The ALICE experiment at the CERN LHC*, JINST **3** (2008), S08002.

[Ara09]   I. Aracena, *Operational Experience of the ATLAS High-Level Trigger with Single-Beam and Cosmic Rays*, ATL-DAQ-PROC-2009-044, 2009.

[ATL97]   ATLAS Collaboration, *The ATLAS Calorimeter Performance*, Technical Design Report, CERN/LHCC 96-40, 1997.

[ATL03]   ATLAS HLT/DCS/DAQ Group, *The ATLAS High-Level Trigger, Data Acquisition and Controls*, Technical Design Report, CERN/LHCC 2003-022, 2003.

[ATL08a]   ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3** (2008), S08003.

[ATL08b]   ATLAS Collaboration, *Expected performance of the ATLAS experiment : detector, trigger and physics*, CERN-OPEN-2008-020, December 2008.

[ATL09]   ATLAS e-News, *Status snapshot*, 14 December 2009,
http://atlas-service-enews.web.cern.ch/atlas-service-enews/2009/news_09/news_status12-09.php.

[ATL10]   ATLAS Public Web Pages, *Detector Description: Inner Detector*, 2010,
           http://www.atlas.ch/inner-detector.html.

[Bai03]   R. Bailey and P. Collier, *Standard Filling Schemes for Various LHC Operation
           Modes*, LHC Project Note 323 (Revised), 2003.

[Baj09]   M. Bajko et al., *Report of the Task Force on the Incident of 19th September 2008 at
           the LHC*, LHC-PROJECT-Report-1168, March 2009.

[Bar08]   B. M. Barnett et al., *ATLAS Level-1 Calorimeter Trigger Read-out Driver*, Module
           Specifications, Version 1.2.2, 2008 (last revised 13.05.2008),
           http://hepwww.rl.ac.uk/Atlas-L1/Modules/ROD/ROD-spec-version1_2_2.pdf.

[Brü04]   O. Brüning et al., *The LHC Design Report - Volume 1: The LHC Main Ring*, CERN,
           2004,
           http://lhc.web.cern.ch/LHC/LHC-DesignReport.html.

[CER92]   CERN Microelectronics Group, *PHOS4 - 4 Channel Delay generation ASIC with
           1 ns resolution*, Rev 1.2, January 1992,
           http://lhcb-elec.web.cern.ch/lhcb-elec/meetings/lhcbweek_
           february00/Delaychip%20SpecificationsV1.pdf.

[CER09a]   CERN Document Server, *First 2.36 TeV Collision Events recorded by the ATLAS
           experiment*, CERN-EX-0912226 01, December 8th, 2009.

[CER09b]   CERN Document Server, *First 900 GeV Candidate Collision Events observed at the
           ATLAS experiment*, CERN-EX-0911200 02, November 23rd, 2009.

[CER10a]   CERN Document Server, *Computer generated image of the whole ATLAS detector*,
           CERN-GE-0803012 05, 2010.

[CER10b]   CERN Document Server, *ATLAS 7 TeV Collision Events recorded*, CERN-EX-
           1003060 04, March 30th, 2010.

[CER10c]   CERN Public Web Pages, *The accelerator complex*, 2010,
           http://public.web.cern.ch/public/en/Research/AccelComplex-en.html.

[Chi09]   Childers J.T. and Wessels M., Kirchhoff Institute for Physics, University of Heidel-
           berg, private communication, 2009.

[Chr04]   Christiansen, J., Marchioro, A., Moreira, P., Toifl, T., *A Timing, Trigger and Control
           Receiver ASIC for LHC Detectors*, Version 3.9, October 2004,
           http://ttc.web.cern.ch/TTC/TTCrx_manual3.9.pdf.

[Cle06]   W. Cleland et al., *Receiver/Monitor System for the ATLAS Liquid Argon Calorime-
           ters*, ATL-AN-EN-0043, Version 4, 2006,
           https://edms.cern.ch/file/347184/4/receiver.pdf.

[CMS08]   CMS Collaboration, *The CMS experiment at the CERN LHC*, JINST **3** (2008), S08004.

[Cos09]   M. Costa et al., *Commissioning of the ATLAS detector with cosmic rays and first LHC beams*, J. Phys. Conf. Ser. **171** (2009), 012100.

[Dat]   *Database for the ATLAS Level-1 Calo Trigger*, http://www.kip.uni-heidelberg.de/atlas/db/DbPPr/welcome.html.

[Djo08]   A. Djouadi, *The Anatomy of Electro-Weak Symmetry Breaking. I: The Higgs boson in the Standard Model*, arXiv:hep-ph/0503172v2.

[Eis03]   Eisenhandler, E. and Gillman, A.R., *Specification of Level-1 Receiver/Monitor System for the ATLAS Tile Calorimeter*, ATL-DA-ES-0034, Version 2, 2003, https://edms.cern.ch/file/390469/2/ReceiverSpecV2.pdf.

[Eng64]   Englert F. and Brout R., *Broken Symmetry and the Mass of Gauge Vector Mesons*, Phys. Rev. Lett. **13** (1964), 321–322.

[Eva08]   L. Evans and P. Bryant, *LHC Machine*, JINST **3** (2008), S08001.

[Gee06]   C. N. P. Gee, *ATLAS Calorimeter First Level Trigger - Timing Control Module*, Version 1.0.5, September 2006, http://hepwww.rl.ac.uk/Atlas-L1/Modules/TCM/TCM_V1_0_5.pdf.

[Gia04]   F. Gianotti, *Physics at the LHC*, Physics Reports **403-404** (2004), 379–399.

[Gla61]   Glashow, S.L., *Partial Symmetries of Weak Interactions*, Nucl. Phys. **22** (1961), 579–588.

[Gur64]   Guralnik, G.S., Hagen, C.R., Kibble, T.W.B., *Global Conservation Laws and Massless Particles*, Phys. Rev. Lett. **13** (1964), 585–587.

[Han09a]   P. Hanke, Kirchhoff Institute for Physics, University of Heidelberg, private communication, 2009.

[Han09b]   P. Hanke, *The PreProcessor Module for the ATLAS Level-1 Calorimeter Trigger*, Version 1.0, Kirchhoff Institute for Physics Heidelberg, 2009, http://hepwww.rl.ac.uk/Atlas-L1/Modules/PPr/PPMod_Wrup.pdf.

[Her03]   R. Herveille, I^2C *- Master Core Specifications*, Revision 0.9, July 2003, http://www.latticesemi.com/documents/i2cm.pdf?jsessionid=ba30408ea6ca$F2$93$C.

[Hig64]   Higgs, P.W., *Broken Symmetries and the Masses of Gauge Bosons*, Phys. Rev. Lett. **13** (1964), 508–509.

[Hil06]   S. Hillier et al., *ATLAS Level-1 Calorimeter Trigger Cluster Processor Module*, Project Specification (Post PRR), version 2.03, 2006, http://hepwww.rl.ac.uk/Atlas-L1/Modules/CPM/CPM_Specification_2_03.pdf.

[Hoe10]   A. Hoecker, *Commissioning and early physics analysis with the ATLAS and CMS ex-periments*, arXiv:1002.2891v1, Comments: Lecture notes from the 5th Latin American School of High-Energy Physics, Recinto Quirama, Colombia, March 15-28, 2009.

[Hus02]   D. Husmann et al., *Pre-Processor ASIC, User and Reference Manual*, November 29, 2002,
http://www.kip.uni-heidelberg.de.

[IHE99]   IHEP Heidelberg Group, *Specification of the PreProcessor ASIC for the ATLAS Level-1 Calorimeter Trigger*, July 1999,
http://www.kip.uni-heidelberg.de/atlas/projects/construction/PP_ASIC/.

[Inf]   Infineon Technologies, *SFP - Small Form-factor Pluggable, Multimode 850 nm 2.125 and 1.0625 Gbit/s Fibre Channel, 1.25 Gigabit Ethernet Transceiver with LC Connector*, V23818-M305-B57.

[Jau]   Jauch Quartz, *VX3 3.3V, Surface Mount Crystall Oscillator*.

[Joo05]   M. Joos, An Introduction to VMEbus, 2005 (last revised 11.10.2005),
https://twiki.cern.ch/twiki/pub/Atlas/VmeConfiguration/vme-tutorial.pdf.

[Kho09]   A. Khomich, *PPM DCS Status*, L1Calo internal talk, Level-1 Calorimeter Trigger Joint Meeting, 2009,
http://indico.cern.ch/conferenceDisplay.py?confId=45489.

[Kho10]   A. Khomich, Kirchhoff Institute for Physics, University of Heidelberg, private communication, 2010.

[L1C04]   L1Calo Collaboration, *ATLAS Level-1 Calorimeter Trigger Algorithms*, ATL-DAQ-2004-011, Version 1.0, 2004.

[L1C07a] *L1Calo Photographs: Work in USA15*, 2007,
http://hepwww.rl.ac.uk/Atlas-L1/pages/Photographs.html.

[L1C07b] L1Calo Collaboration, *Level-1 Calorimeter Trigger: Cable Mappings and Crate Layouts from Analogue Inputs to Processors*, ATL-DA-ES-0036, Version 2.2, 2007,
https://edms.cern.ch/document/399348/2.2.

[L1C08a] L1Calo Collaboration, *The ATLAS Level-1 Calorimeter Trigger*, JINST **3** (2008), P03001.

[L1C08b] L1Calo Collaboration, *First data with the ATLAS Level-1 Calorimeter Trigger*, ATL-DAQ-PROC-2008-006, 2008.

[Lan08]   M. Landon, *Use of TTC System*, Version 1.1, 2008,
http://atlas-l1calo.web.cern.ch/atlas-l1calo/doc/out/TTCBusy.pdf.

[LEP02]   LEP Collaborations ALEPH, DELPHI, L3, OPAL, the LEP Electroweak Working Group and the SLD Heavy Flavour and Electroweak Groups, *A Combination of Preliminary Electroweak Measurements and Constraints on the Standard Model*, arXiv:hep-ph/0503172v2.

[LHC08a]   LHCb Collaboration, *The LHCb Detector at the LHC*, JINST **3** (2008), S08005.

[LHC08b]   LHCf Collaboration, *The LHCf detector at the CERN Large Hadron Collider*, JINST **3** (2008), S08006.

[Liq07]   Liquid Argon Collaboration, *ATLAS Liquid Argon Calorimeter Back End Electronics*, JINST **2** (2007), P06002.

[Liq08]   Liquid Argon Collaboration, *ATLAS Liquid Argon Calorimeter Front End Electronics*, JINST **3** (2008), P09003.

[Mah04]   K. Mahboubi, *PreProcessor Module: known problems and modifications*, L1Calo internal talk, Level-1 Calorimeter Trigger Joint Meeting, November 2004, http://indico.cern.ch/conferenceDisplay.py?confId=a044562.

[Mah05]   K. Mahboubi et al., *Rear G-Link Transmitter Module - Optical Tx (RGTM-O) for the Pre-Processor*, Version 1.0, 2005, http://www.te.rl.ac.uk/esdg/atlas-flt/.

[Mah06]   K. Mahboubi, *Channel Mapping on the PreProcessor Module*, Draft Version, 2006, http://www.kip.uni-heidelberg.de/atlas/projects/construction/PP_Module/ppm_channel_mapping.pdf.

[Max94]   Maxim Integrated Products, *Octal, 8-bit, Serial DACs with Output Buffer*, 1994.

[Mor04]   see e.g. references in: Morii, E., Lim, C.S., Mukherjee, S.N., *The Physics of the Standard Model and Beyond*, World Scientific Publishing Co. Pte. Ltd., 2004.

[Mor09]   J. Morris, *Analogue Input Calibration of the ATLAS Level-1 Calorimeter Trigger*, Proc. Topical Workshop on Electronics for Particle Physics, Paris 2009, pp. 196–199.

[Mül08a]   F. Müller, *Rate Metering Archiving and Analysis Software*, L1Calo internal talk, Level-1 Calorimeter Trigger Joint Meeting, November 2008, 2008.

[Mül08b]   F. Müller, *Rate Metering for the ATLAS Experiment*, Kirchhoff Institute for Physics, University of Heidelberg, Diploma thesis, 2008.

[Nat02]   National Semiconductor, *DS92LV1021 and DS92LV1210, 16-40 MHz 10 Bit Bus LVDS Serializer and Deserializer*, 2002.

[Pau05]   T. Pauly et al., *ATLAS Level-1 Trigger Timing-In Strategies*, Proc. 11th Workshop on Electronics for LHC and Future Experiments, Heidelberg 2005, pp. 274–278.

[Per05]   V. Perera, *RGTM-O Schematics (PC3284M/2)*, Rutherford Appleton Laboratory, 2005,
            http://www.te.rl.ac.uk/esdg/atlas-flt/.

[Pfe99]   Pfeiffer U. and Wolfgang Hötzel, *Bunch-Crossing Identification for saturated calorimeter signals*, ATL-DAQ-99-009, 1999.

[Qia05]   W. Qian, *TTC Decoder Card Specifications*, Version 1.1, November 2005,
            http://hepwww.rl.ac.uk/atlas-l1/Modules/TTC/TTCDec_Version1.1.pdf.

[Sal68]   A. Salam, *Weak and electromagnetic interactions*, Proc. 8th Nobel Symposium on Elementary particle theory, relativistic groups and analyticity, Stockholm 1968, pp. 367–377.

[Sam03]   Samsung Electronics, *K7M323625M, 1Mx36 & 2Mx18-bit Flow Through NtRAM^{TM}*, Technical Data, Revision 2.0, 2003.

[Sch08]   Schäfer U. and Silverstein S., *ATLAS Level-1 Calorimeter Trigger Jet/Energy Processor Module*, Project Specification (Post PRR), version 1.2d, 2008,
            http://hepwww.rl.ac.uk/Atlas-L1/Modules/JEM/JEMspec12d.pdf.

[Sch09]   K. Schmitt, Kirchhoff Institute for Physics, University of Heidelberg, private communication, 2009.

[Sjo06]   T. Sjoestrand et al., *PYTHIA 6.4 physics and manual*, JHEP **05** (2006), 026.

[Tay]   B. Taylor, *TTC laser transmitter (TTCex, TTCtx, TTCmx) User Manual*, RD12 working document, Rev. 2.0,
            http://ttc.web.cern.ch/TTC/TTCtxManual.pdf.

[Tev10]   Tevatron New Phenomena & Higgs Working Group, *Combination of Tevatron searches for the Standard Model Higgs boson in the W^+W^- decay mode*, Phys. Rev. Lett. **104** (2010), 061802, arXiv:1001.4162v3.

[The05]   The ATLAS group at the KIP Heidelberg, *Production Readiness Review of ASIC/MCM Assembly: Software for testing and test procedures*, ATL-DA-ES-0041, 2005,
            https://edms.cern.ch/file/566138/1.0/PRR_AsMcmSoft_1.0.pdf.

[Til96]   Tile Calorimeter Collaboration, *ATLAS Tile Calorimeter*, Technical Design Report, CERN/LHCC 96-42, CERN, 1996.

[TOT08]   TOTEM Collaboration, *The TOTEM Experiment at the CERN Large Hadron Collider*, JINST **3** (2008), S08007.

[Tri09]   *ATLAS Approved Trigger Plots*, 2009,
            https://twiki.cern.ch/twiki/bin/view/Atlas/ApprovedPlotsTrigger.

[VME87]   VMEbus International Trade Association, *The VMEbus Specifications*, VITA Publication, 1987.

[Web08]   P. Weber, *ATLAS Calorimetry: Trigger, Simulation and Jet Calibration*, Ph.D. thesis, 2008.

[Wei67]   Weinberg, S., *A Model of Leptons*, Phys. Rev. Lett. **19** (1967), 1264–1266.

[Wen09]   T. Wengler, *LHC/ATLAS 2009/10 running*, seminar presentation, July 2009, http://iktp.tu-dresden.de/IKTP/Seminare/IS2009/Thorsten_Wengler.pdf.

[Xil06]   Xilinx, *Virtex-E 1.8V FPGA, Detailed Functional Description*, 2006, http://www.xilinx.com/support/documentation/data_sheets/ds022-2.pdf.

Acknowledgements