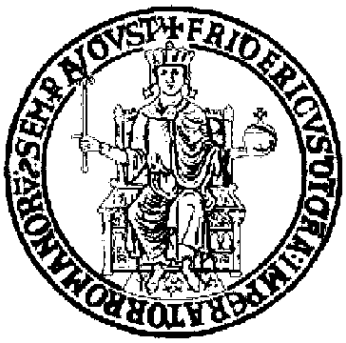
CERN-THESIS-2010-317



Università degli studi di Napoli “Federico II”

Dottorato di ricerca in Fisica fondamentale ed applicata
XXIII Ciclo

Luciano Capasso

The Monitoring system of the ATLAS Muon Spectrometer Read Out Driver

A thesis submitted for the degree in Doctor of Philosophy in Physics

Supervisor:

Prof. A. Aloisio

Advisors:

Prof. I. Ortosecco

Dott. G. Saracino

All truths are easy to understand
once they are discovered.
The point is to discover them.

Galileo Galilei

Contents

Introduction

This thesis describes the design and test of the monitoring system of the Muon Spectrometer Read Out Driver (ROD) of the ATLAS experiment.

The Large Hadron Collider (LHC) is a particle accelerator and collider located at CERN in Geneva (Switzerland). It ha been designed in order to provide proton-proton collisions with an energy of 14 TeV in the center-of-mass and a luminosity of 10^{34} cm^{-2} s^{-1}. The first beam in the Large Hadron Collider was successfully steered in the morning of 2008, 10_{th} September. From that day until now, there have been measurements and calibration operations, in order to increase the beams' energy and luminosity to reach the foreseen values written above. Actually it reached energies of 7 TeV and luminosities of 10^{32} cm^{-2} s^{-1}.

The LHC has four interaction points around its circumference. Around one of these points, the ATLAS detector has been built.

The general purpose detector ATLAS (A Toroidal LHC ApparatuS) has been designed primarily to search for the Higgs boson in a wide mass range and in its main decay channels. The Higgs boson is theoretically responsible for the electroweak symmetry breaking and for the particle mass generation.

ATLAS will also allow measurements within B meson and top quarks physics, the study of CP violation and the search for SUSY particles.

The ATLAS muon spectrometer is located in the outer region of the detector and it is fundamental for many of the program items of the ATLAS experiment. The production of high mass particles generates muons with high transverse momentum (p_T), so the selection and the precision measurement on muons with high p_T is of the outmost importance for the physics program.

In the barrel region of the muon spectrometer, Resistive Plate Chambers (RPC) are used as trigger detectors. Data from RPCs are collected by on-detector electronics and sent via optical fibres to the USA15 counting room, where they are received by RX boards and then transferred to the Read Out Driver (ROD) boards.

The ROD is a VME electronic board and it is part of the Data Acquisition (DAQ) environment of the ATLAS detector. It performs the event building, checks the correct data formatting and transmits the data to further data analysis levels. In the ATLAS DAQ system, each RPC ROD handles data coming from one of the 32 half-sectors of the ATLAS muon spectrometer. The ROD also controls some of the critical elements of the data acquisition system, such as the synchronization with the LHC collider, using some LHC signals and distributing them to the RX boards. The ROD event building algorithm is executed by Finite State Machines (FSMs).

My original contribution during the PhD school to the ATLAS experiment was to develop a monitoring system of the ROD Event Builder. Such system performs real time and statistical analysis of the ROD state machines dynamics. It is designed around a microprocessor interfaced with several custom peripherals. The system allows to profile the ROD activity by filling histograms, plots and transferring the ROD monitored data via SSH to a remote terminal. The monitoring environment also measures the elapsed time for each event, its length and keeps track of status and error words produced during the event building operation.

The first chapter of this thesis briefly describes the ATLAS experiment, the LHC collider and the main items of its physics program. The muon spectrometer and the its trigger system based on RPC detectors are described more deeply, while a general description of the ATLAS trigger and DAQ systems is also provided.

The second chapter is about the level 1 trigger system of the muon spectrometer. The trigger system segmentation, the algorithms for the detection of muons and the synchronization of the DAQ systems are presented. It is provided also a description of the trigger electronics

The third chapter describes the ROD board: in particular the chapter presents the hardware architecture of the ROD environment, the devices hosted on board and the electronics developed inside the FPGAs. Details on the Event Builder finite state machine and on its Engine are provided.

Chapter four presents the monitoring system of the ROD Event Builder Engine, which is the original contribution I gave to this PhD thesis work. Such system is based upon a soft-core microprocessor, connected to several

peripherals and embedded inside the same FPGA hosting the ROD Event Builder Engine. The hardware environment and the processor's software architecture are deeply described. Finally the results of the performed tests are presented.

CHAPTER ONE – THE ATLAS EXPERIMENT AND ITS DETECTOR

This chapter describes the architectures of the Large Hadron Collider, the ATLAS detector and its main physics goals. More in detail, a short description of the LHC with its main features is provided and the different sub-detectors of ATLAS together with the magnetic system are analyzed deeply. A particular attention is focused on the Muon spectrometer and its detectors, in both the Barrel and the End-cap sections. Finally an overview of the ATLAS trigger system is presented.

1.1 The Large Hadron Collider (LHC) main goals

LHC is the new proton-proton collider built at CERN. It has been designed primarily to search for the Higgs boson, the last piece of the *Standard Model*, a theory describing the interactions of point-like fermions with spin ½ (quarks and leptons) as mediated by bosons of spin 1 (gluons, W, Z and γ): the gluons mediate the strong interaction between the quarks, another boson (a photon) mediates the electromagnetic interaction between charged particles and three other bosons (W^+, W^- and Z^0) mediate the weak interaction. The Lagrangian of the Standard Model theory has a global gauge invariance under the symmetry group $SU(3)_C \otimes SU(2)_I \otimes U(1)_{Hy}$ where C stands for Colour, I for Isospin and Hy for Hypercharge. The interaction is the manifestation of a local gauge invariance, that implies moreover that all the masses of the fermions and bosons are equal to zero.

The existence of massive particles is justified through the *Spontaneous Symmetry Breaking* mechanism: the local gauge symmetry $SU(2) \otimes U(1)$ is broken by the existence of a *Higgs field* that implies the existence of a neutral scalar boson H^0, called *Higgs boson*. This last is thought to be responsible of the masses of all the particles. Until now the experimental data are in agreement with the Standard Model: the discovery of the neutral weak current in the '70s, the discovery of the gluons in 1979 and the discovery of $W^{\pm}$ and Z^0 bosons in 1983. The mass predictions for some of these particles strongly agree with the measured ones [1] [2], as shown in Table 1.

Quantity	Measured (GeV/c^2)	SM prediction (GeV/c^2)
Mass of W boson	80.399 ± 0.023	80.390 ± 0.018
Mass of Z boson	91.1876 ± 0.0021	91.1874 ± 0.0021

Table 1 – Comparisons between experimental data and Standard Model predictions.

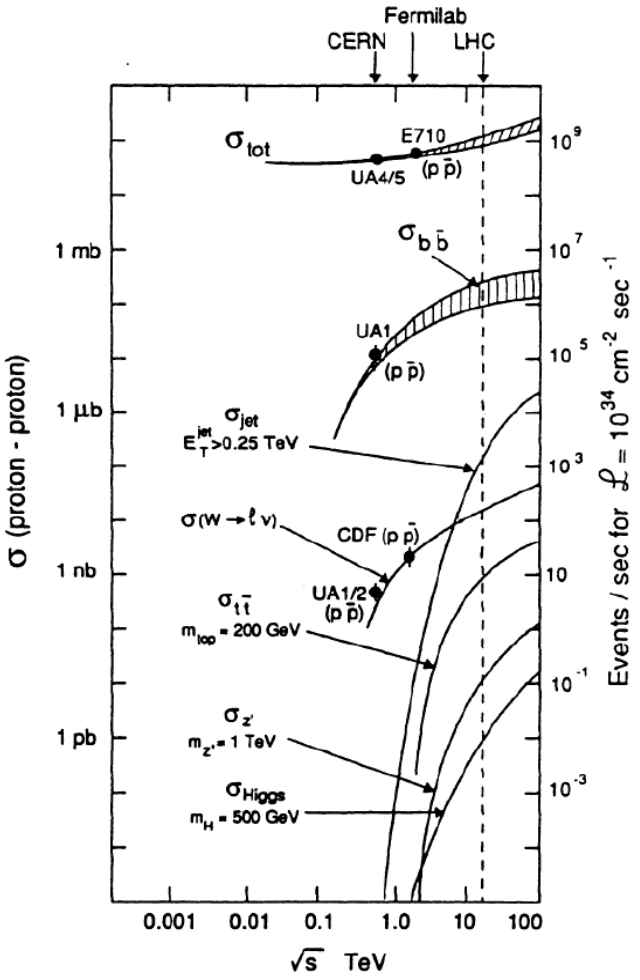


Figure 1 – Cross section for different processes with respect to the center-of-mass energy.

One of the physics goal of the LHC is to discover evidence of the existence of the Higgs boson. In Figure 1 are shown the cross sections of different processes with respect to the center-of-mass energy. The dotted line is the working line foreseen for the LHC when it will be fully operative (it should reach energies of 14 TeV and luminosities of 10^{34} cm^{-2} s^{-1}). Actually the LHC reached center-of-mass energies of 7 TeV and luminosities of 10^{32} cm^{-2} s^{-1}. It is possible to see the cross section of the Higgs boson, whose mass is hypothesized to be 500 GeV.

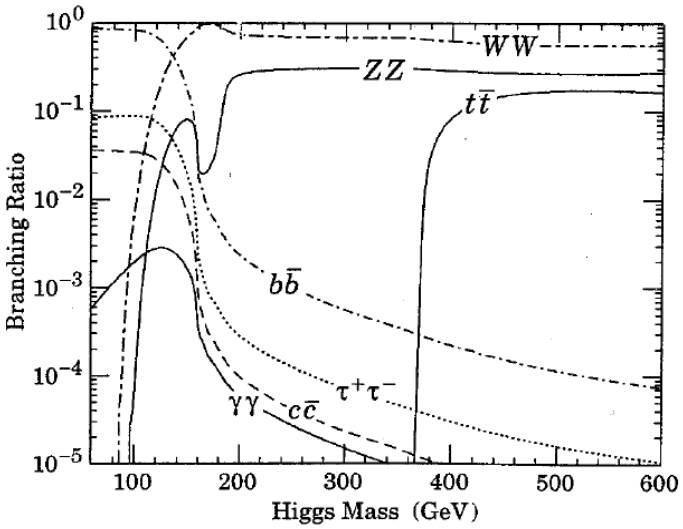


Figure 2 – Branching ratios for the Higgs, with respect to its mass.

Figure 2 shows the branching ratios of several decay channels for the Higgs boson, with respect to its mass. Some processes are difficult to be detected: for example, processes involving neutrinos in the final state require missing energy measurements, or processes with a background of hadronic jets have a difficult reconstruction procedure. The most easily detectable decays are:

$$H \rightarrow ZZ \rightarrow 4 \text{ leptons}$$
$$H \rightarrow ZW^* \rightarrow 2 \text{ leptons} + 2\ \nu$$
$$H \rightarrow ZZ^* \rightarrow 4 \text{ leptons}$$

Until now, particle accelerators have never reached the high energy levels of the LHC. This fact implies that every detector of the ATLAS apparatus must be built to give the better performances the actual technology allows, in order to

investigate as better as possible eventual physics phenomena beyond the Standard Model.

For example, one of the new studies that will be performed at the LHC is the determination of the quark top mass with a resolution less than 0.05 GeV/c2. The most interesting decay to reach such purpose is:

$$t\,t \rightarrow b\,b\,W(j\,j)\,W(l\,\nu)$$

so the ATLAS apparatus will have to be able to detect leptons in the final state and reconstruct secondary decay vertexes. These characteristics will be also used to investigate the B^0 meson physics, e.g. the rare decay study:

$$B^0_d \rightarrow \mu^+ \mu^- (X)$$

or the decay:

$$B^0_d \rightarrow J/\psi\, K^0_s$$

useful in CP violation studies.

In summary, LHC has been designed to investigate new physics in a energy range never reached before. In particular, the main goals of the LHC are:

- Searching for the Higgs boson;
- The investigation of the heavy quark physics (as the B and T quarks);
- Studying the CP violation;
- The study of an eventual composite quarks' structure;
- Searching for SUSY particles;
- The investigation of eventual new phenomena beyond the modern physics.

1.2 The LHC architecture

The LHC is a collider designed to accelerate two proton beams in opposite directions, in order to let them to collide with an energy of 14 TeV in the center-of-mass. The foreseen luminosity is 10^{34} cm^{-2} s^{-1}. As said before, the

actual reached values for the energy and the luminosity are respectively 7 TeV and 10^{32} cm^{-2} s^{-1}.

The beams are a succession of proton bunches, each one containing 10^{11} particles, spaced by 25 ns (corresponding to a frequency of 40 MHz) [3] [4].

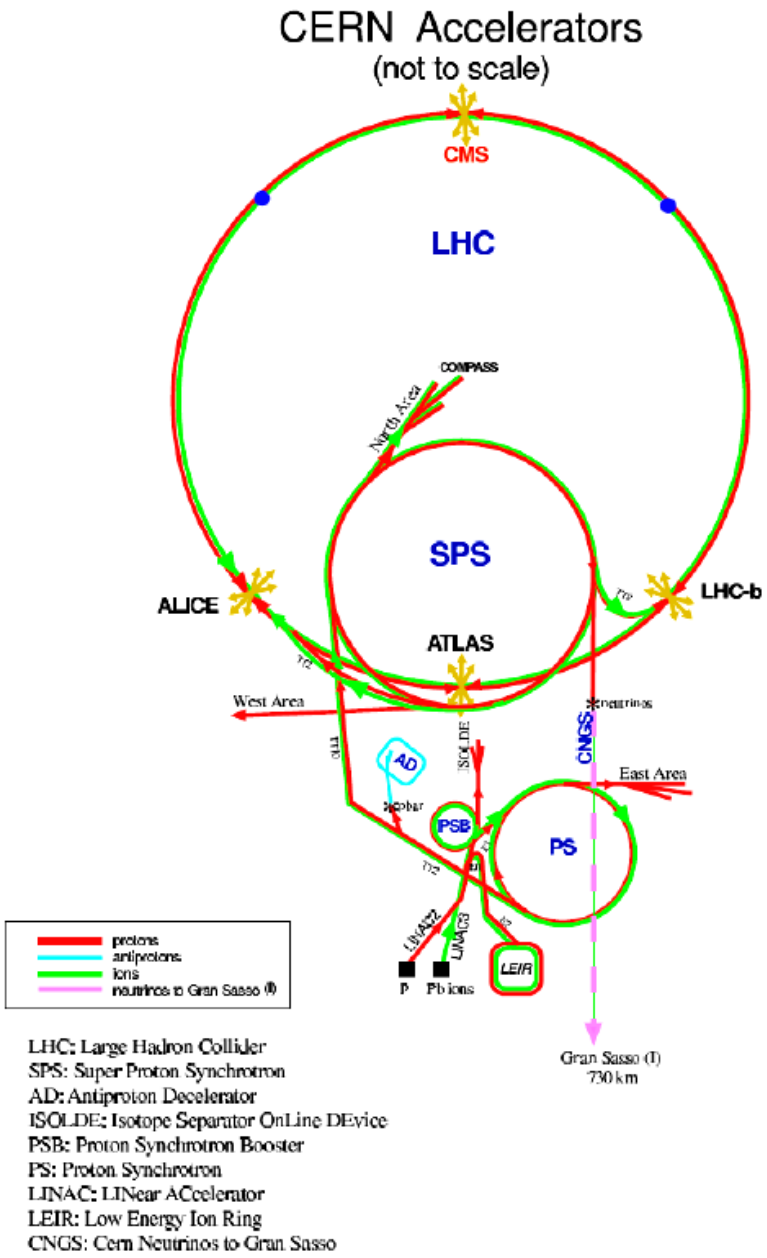


Figure 3 – Injection scheme for the LHC.

The luminosity of the LHC is defined as:

$$\mathcal{L} = f \frac{N^2}{4 \pi \sigma_x \sigma_y}$$

where N is the number of proton per bunch (10^{11}), f is the collision frequency (40 MHz) and σ_x and σ_y characterize the Gaussian transverse profile in the horizontal and vertical directions.

The LHC has been built in the LEP tunnel where it is fed by particle sources and different pre-accelerators. The protons are produced and accelerated to 50 MeV by the proton Linac, before being injected into the Proton Synchrotron Booster (PSB), that accelerates them to 1.4 GeV. From the PSB, the protons reach the Proton Synchrotron (PS), that accelerates them as far as 26 GeV. Finally, the Super Proton Synchrotron (SPS) injects the proton beams into the LHC with an energy of 450 GeV. This last accelerates every beam up to 7 TeV. In Figure 3 you see the injection scheme just described for the Large Hadron Collider. In Table 2 are shown some foreseen parameters of the LHC for proton-proton operation [5].

Parameter	**Value**
Energy (TeV)	7.0
Dipole field (T)	8.4
Coil aperture (mm)	56
Distance between apertures (mm)	180
Luminosity (cm^{-2} s^{-1})	10^{34}
Beam-beam parameter	0.0032
Injection energy (GeV)	450
Circulating current/beam (A)	0.53
Bunch spacing (ns)	25
Particles per bunch	10^{11}
Stored beam energy (MJ)	332
Normalized transverse emittance (mm)	3.75
R.m.s. bunch length (m)	0.075
Beta values at I.P. (m)	0.5
Full crossing angle (mrad)	200
Beam lifetime (h)	22
Luminosity lifetime (h)	10
Energy loss per turn (keV)	6.9
Critical photon energy (eV)	45.6
Total radiated power per beam (kW)	3.7

Table 2 – Parameters of the LHC for the proton-proton operation.

The two proton beams channels lie side by side, 194 mm apart, in order to let them use the same magnets' system. This arrangement enabled a cost saving of 30%, compared to a system with two separate magnets.

In order to accelerate the proton beams to the required energy and to bend them properly around the LEP tunnel, the magnetic fields need to be about 9 T. To reach such a high field, superconductive magnets are used. These lasts, in order to reach superconductive temperatures, are immersed in a pressurised bath of superfluid helium at about 0.13 MPa (1.3 bar) and a maximum temperature of 1.9 K [6].

1.3 The ATLAS detector

ATLAS is a detector built at one of the four LHC beam interaction point. It has a cylindrical symmetry along the beam axis and the cylinder has a 46 m length m and a 22 m diameter. Figure 4 shows the ATLAS architecture. Closest to the interaction point there is a tracking detector (called inner detector) and, from inner to outer radius, there are an electromagnetic calorimeter, an hadronic calorimeter and a muon spectrometer. All these sub-detectors are immersed in a magnetic field in order to bend the charged particle trajectories and measure their momentum.

In Figure 5 the ATLAS coordinate system in shown. The system is right-handed with the z-axis along the beam direction and the x-axis pointing toward the center of the LHC circumference; the y-axis points from the interaction point upward. Because of the detector's symmetry, a cylindrical (z, φ, θ) coordinate system is used. The azimuthal angle φ is referred to the x-axis, in the x-y plane; the polar angle θ is referred to the z-axis, in the r-z plane. Instead of the polar angle θ, the pseudorapidity variable is used. This last is defined as:

$$\eta \approx - \ln (\tan(\theta/2))$$

and allows to identify three regions in the detector, *Barrel*, *End-cap* and *Forward*, as show in Figure 6.

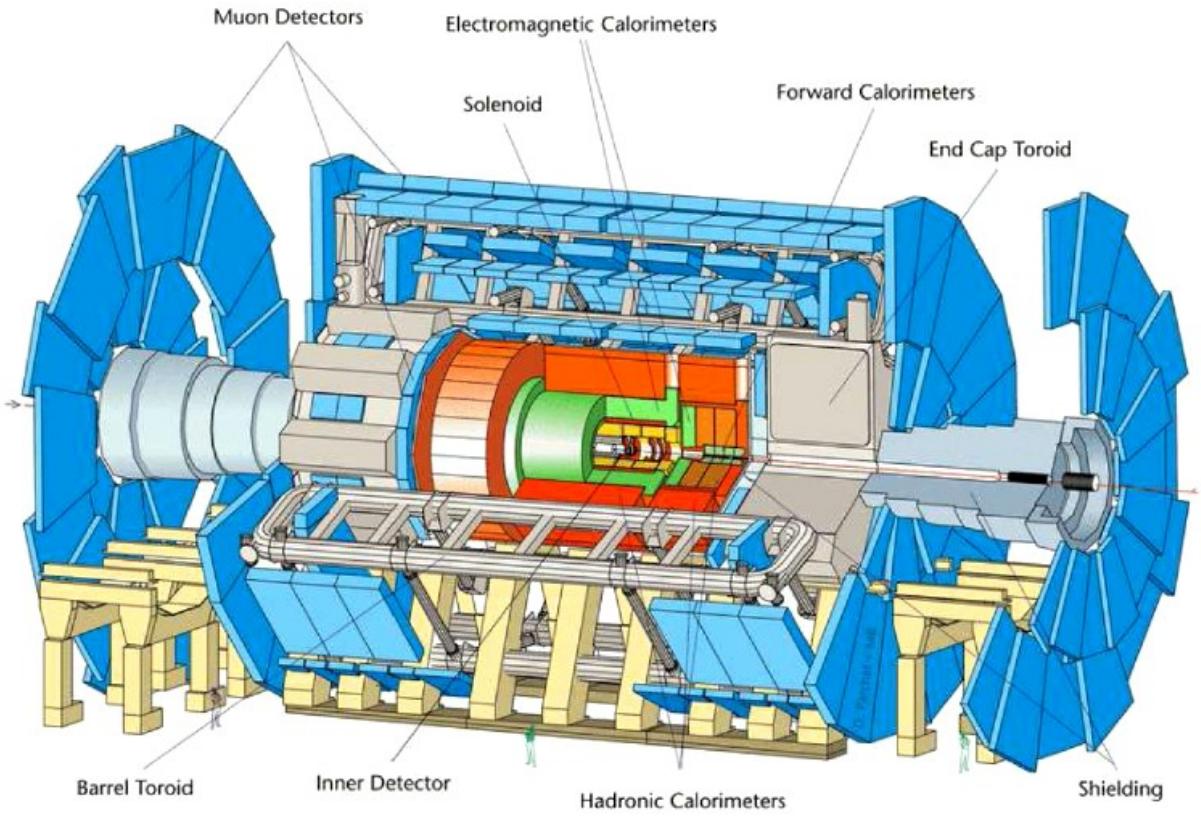


Figure 4 – The ATLAS detector architecture.

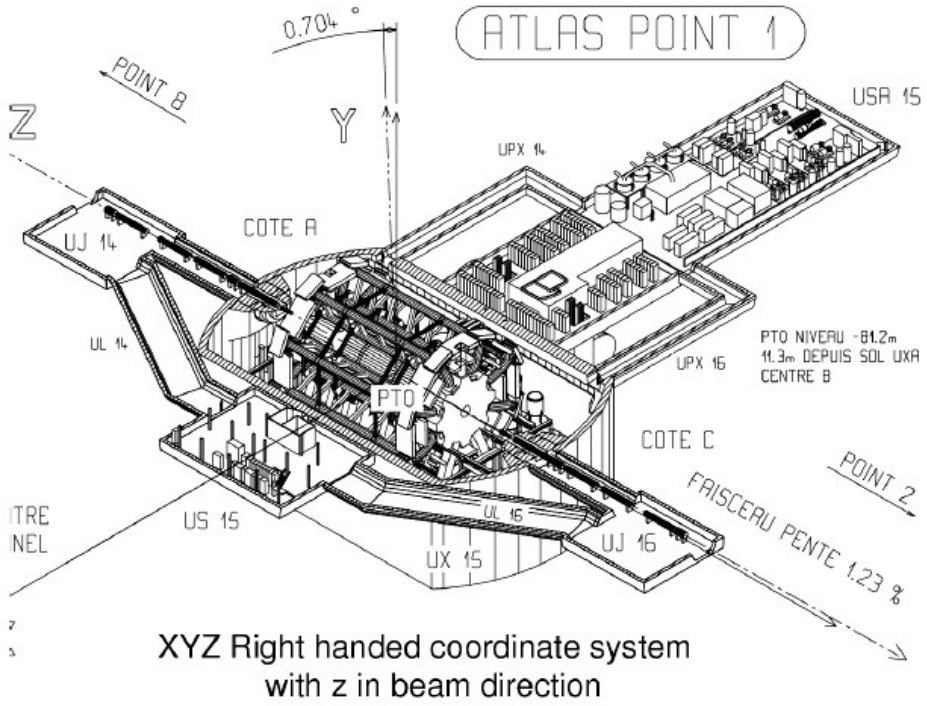


Figure 5 – The ATLAS coordinate system.

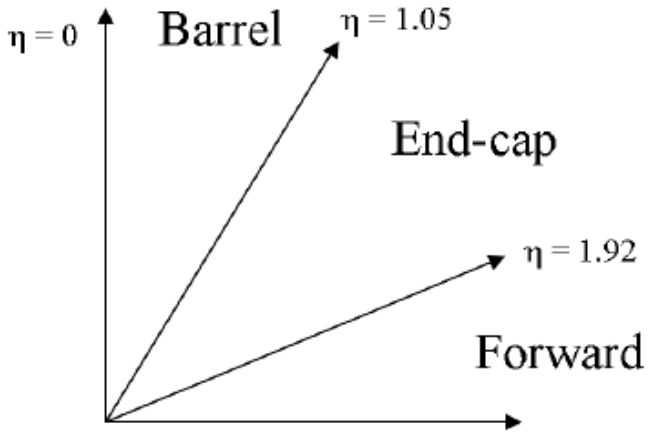


Figure 6 – The three detector's regions, defined by the pseudorapidity variable.

In order to give the highest performances, the ATLAS detector has been designed to satisfy the following requirements:

- A tracking system with a very high resolution, allowing the reconstruction of secondary decay vertexes of heavy quarks;
- A very good energy and direction resolution of the electromagnetic calorimeter, to identify correctly electrons and photons;
- A good hadronic calorimeter to achieve accurate energy measurements in order to perform jets identification and missing energy measurements;
- A muon spectrometer able to perform momentum measurements with a very high precision;
- Large acceptance in pseudorapidity and almost total coverage in azimuthal angle;
- High trigger efficiency for all interesting processes.

1.3.1 The magnetic system

Magnetic fields are essential in the ATLAS detector, in order to bend the trajectories of charged particles. There are two different structures that generate such fields [7]:

— a superconducting solenoid, producing a 2 Tesla magnetic field for the inner detector;
— an external toroidal superconducting magnet, producing a variable magnetic field from 3 to 6 Tesla (depending from the pseudorapidity) for the muon spectrometer.

The external magnet is made of three toroids, one in the barrel region and two in the end-cap regions, and each one is made of eight independent coils, arranged with an octagonal symmetry. It has on open structure to minimize the contribution of multiple scattering to the momentum resolution; most of the coils is cooled by liquid Helium at 4.5 Kelvin.

1.3.2 The inner detector

The Inner detector (shown in Figure 7) is located around the beam's interaction zone and has a cylindrical symmetry, with a length of 7 m and an external diameter of 2.3 m. A superconducting solenoid surrounds entirely the inner detector and it generates a magnetic field of 2 Tesla parallel to the beam axis. This field wraps up entirely the detector.

The main task of the inner detector is to reconstruct particles' tracks and to perform measurements of momenta of charged particles in the apparatus. Moreover, the inner detector gives the position of primary and secondary vertexes of decays of some short-lifetime particles, such as B-Hadrons:

$$B^0_d \rightarrow J/\psi \; K^0_s,$$

allowing to identify the produced K particles.

The core of the inner detector has a very high spatial resolution and it is made up of two concentric structures: one made by pixel detectors with fine granularity, in order to grant a high resolution; the other is made by SemiConductor Trackers (SCT), that contributes to measurements of vertex position and momenta. The outer region of the detector is made up of Transition Radiation Trackers (TRT), that are tube detectors arranged along the beam's axis in the barrel region and radially in the end-cap regions. These

sub-detectors have electron identification capability, by employing Xenon gas to detect transition radiation photons. The resolutions of the three sub-detectors making the inner detector are reported in Table 3.

Resolutions	$\sigma_{R\varphi}$ (μm)	σ_z (μm)
Pixel detector	12	66
SCT	16	580
TRT	170	-

Table 3 – Spatial resolutions of the inner detector.

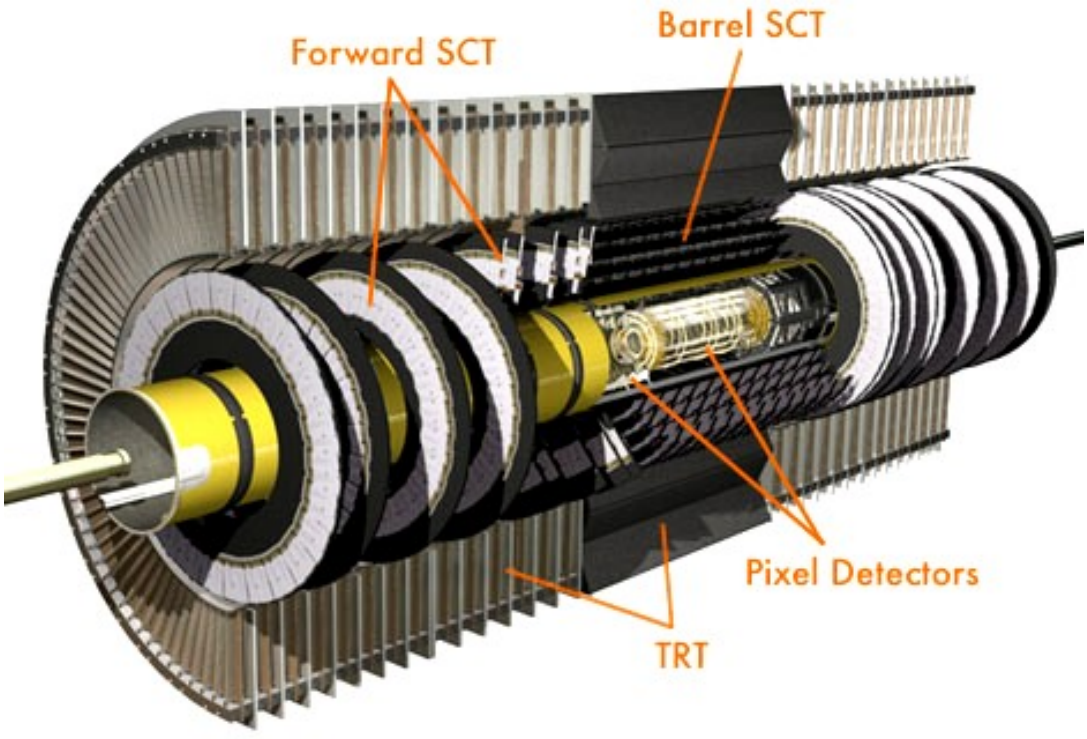


Figure 7 – Architecture of the inner detector.

1.3.3 The calorimeters

The energy resolution of calorimeters improves with increasing energy, making them suitable for use at high energy colliders. Physics requirements on calorimeters on LHC include:

- Accurate measurements of energies and positions of both electrons and photons;
- Measurements of energies and directions of jets;
- Measurements of missing transverse energy of events;
- Particle identification, including separation of electrons, photons and hadronic τ decays from jets;
- Event selection already at the first level of trigger.

The system is divided into an electromagnetic (EM) calorimeter with high resolution close to the interaction point and a large hadronic calorimeter behind with a coarser resolution. The first of them uses Liquid Argon (LiAr) as active medium while in the hadronic calorimeter different technologies are used depending on the environmental constraints like radiation dose. The calorimetric system is shown in Figure 8.

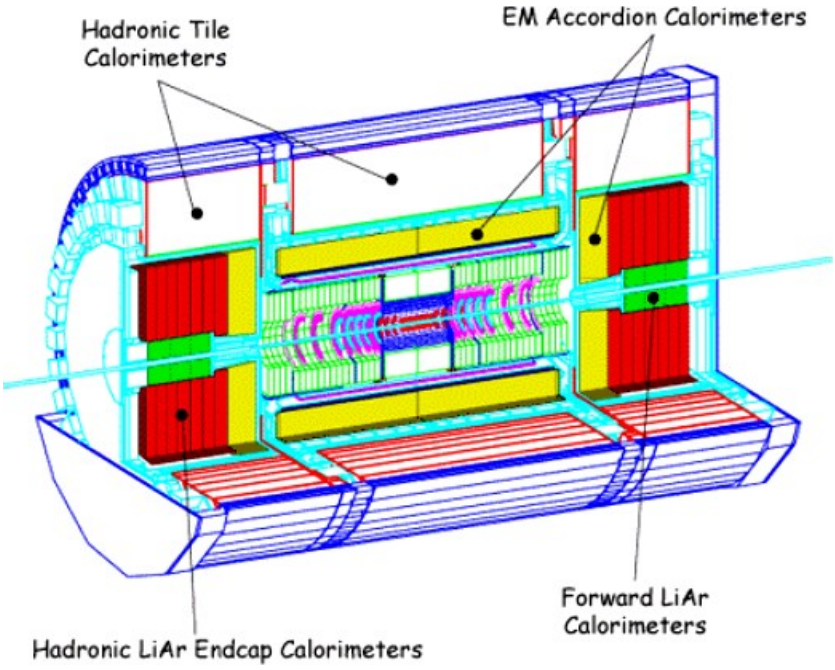


Figure 8 – The ATLAS calorimetric system.

The electromagnetic calorimeter is required to reconstruct electron and photons in the energy range from 2 GeV to 5 TeV.

The principal channels for this calorimeter are:

$$H \rightarrow Z\,Z^* \rightarrow 4e$$

$$H \rightarrow \gamma\gamma$$

These channels place the most stringent requirements on the electromagnetic calorimeter in terms of energy resolution, energy range and particle identification.

The EM calorimeter is a lead-Liquid Argon (LiAr) detector [8]. The barrel of the electromagnetic calorimeter covers $|\eta| < 1.475$ and the two identical end-caps cover $1.375 < |\eta| < 3.2$.

The Liquid Argon technology is radiation resistant and provides long-term stability of the detector response, excellent hermeticity, good energy resolutions and relatively easy detector calibration.

The ATLAS hadronic calorimeters cover the range of $|\eta| < 4.9$ using different techniques best suited for the widely varying requirements and radiation environment over the large η range. The calorimeter provides good resolution for high energy jets. The large η coverage will also guarantee a good E_{Tmiss} measurement, which is important for many physics signatures and in particular for SUper SYmmetry (SUSY) particle searches.

The hadronic barrel calorimeter is a cylinder divided in three sections: a central barrel and two identical extended barrels. They are sampling calorimeters with iron as absorber material and scintillating tiles as active material, called Tile Calorimeter [9].

At larger pseudo-rapidities, where higher radiation resistance is needed, the intrinsically radiation-hard LiAr technology was chosen: the hadronic end-cap calorimeters (EHCal) and the forward calorimeter (FHCal) with the front face 4.7 m far from the interaction point.

1.3.4 The Muon Spectrometer

One of the most important proof of interesting physics at the LHC is the creation of high-momentum muons during the interactions. The ATLAS muon spectrometer [10] has been designed to provide a high precision measurement of muon momentum and spatial position without the need of any other information from other detector data. Moreover, a homogeneous coverage up to large rapidity (| η | = 3) and high efficiency for identifying muons is required in order to achieve the physics goals of the experiment.

The muon spectrometer requirements are:

— A momentum resolution lower than 10%, up to values of transverse momentum pT ~ 1 TeV/c.
— A spatial resolution of ~ μm in the measurement of the particles' position in the direction and lower than 10 mm in the _ direction.
— A good performance in reconstructing events whose final states are characterized by the presence of 2 or 4 muons, that are events that can be related to a Higgs boson decay.
— A trigger system selectivity up to pT > 20 GeV/c.

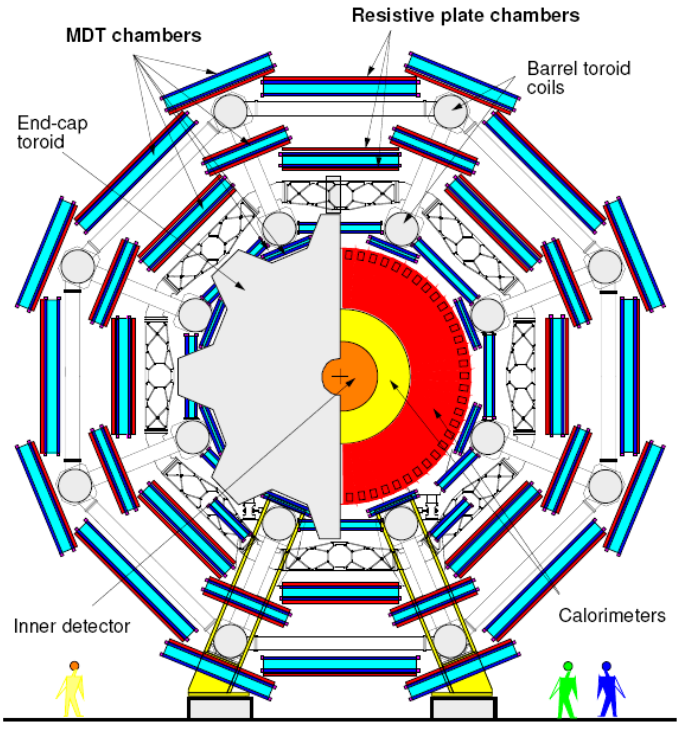


Figure 9 – **A section of the ATLAS Muon Spectrometer.**

The muon spectrometer is built in the outer region of the ATLAS apparatus. It is a tracking system in a toroidal magnetic field and it is made up of precision chambers and trigger detectors. The spectrometer has a barrel shape. It is 46 m long and 22 m wide.

The spectrometer has an octant symmetry in the R-φ plane and it is subdivided in barrel and end-cap spectrometers. In Figure 9 you can see a section of the detector.

The operation principle is based on the magnetic deflection of muon tracks in the magnetic field generated by three large toroidal magnets (one for the barrel and two in the end-caps). The magnetic field lines in the spectrometer are parallel to the φ direction: hence, the component of the particles' momenta in the φ direction is parallel to the magnetic field lines. If we define a cylindrical coordinates system, particles' trajectories are arcs of circumference in the r-z plane. The radius and the direction of the curvature of the trajectory in the r-z plane are depending, given the magnetic field value, by the momentum and by the charge of the particle. By reconstructing the trajectory, the charge and the momentum of the particle can be calculated.

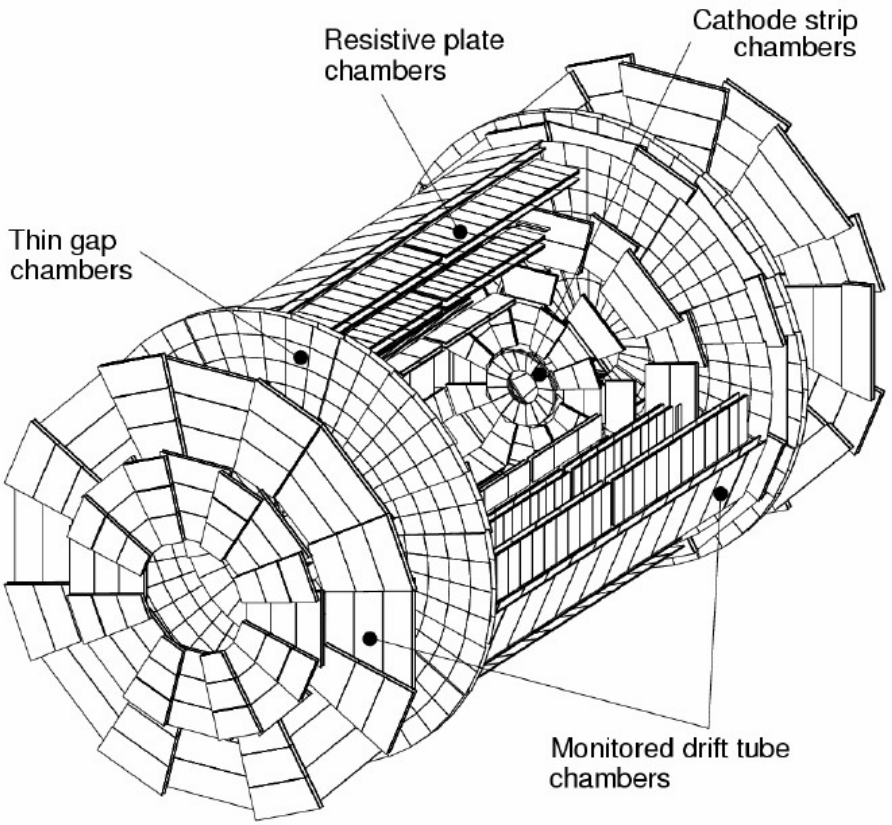


Figure 10 – Detectors compounding the Muon Spectrometer.

Four different kinds of gaseous detectors have been designed, both for precision measurements and for trigger chambers (Figure 10). In the barrel region, both precision chambers and trigger detectors are arranged in three cylindrical concentric surfaces, mounted at a distance of about 5 m, 7.5 m and 11 m from the beam axis, as shown in Figure 9; in the two end-cap regions the detectors are located on four vertical stations, at a distance of ~ 7 m, 10 m, 14 m and 22 m from the interaction point.

For position measurements, *Monitored Drift Tubes* (MDT) are used in the barrel region and *Cathode Strip Chambers* (CSC) are used in the region with high pseudorapidity ($|\eta|$ up to 2.7). For trigger measurements, stations of *Resistive Plate Chambers* (RPC) are used in the region with $|\eta| < 1.05$, and *Thin Gap Chambers* (TGC), multiwire proportional chambers, are used in the end-cap regions.

Monitored Drift Tubes (MDT)

MDT detectors [11] are cylindrical drift tubes with a 30 mm diameter and with a 50 μm W-Rn wire; the gas is a mixture of Ar(91%)-CH_4(4%)-N_2(5%). The drift time is lower than 480 ns and the average spatial resolution for a single tube is 80 μm.

In order to enhance the spatial resolution, each MDT chamber is made of two tracking planes, each made by a superposition of 3 or 4 layers of drift tubes.

The geometry and the position of the detectors is granted by an optical system called RASNIK [12] with a precision better than 30 μm. According to the azimuthal symmetry of the magnet, the spectrometer is divided in 16 sectors in the φ projection and every sector is completely covered by a chamber. The drift tubes are installed perpendicularly to the beam axis: so the chambers can measure muons' position in the η projection, reconstucting the track in the r-z plane. There is a superposition of 200 mm between two chambers of adjacent sectors, to avoid dead regions in the apparatus due to the presence of reinforcement structures and cables.

Cathode Strip Chambers (CSC)

Cathode Strip Chambers [13] are proportional multiwire chambers with cathode strips readout, used to achieve a better spatial resolution: they have to

reconstruct tracks of particles with higher transverse momentum. A precision measure is obtained by evaluating the charge on the cathode (made by several strips), induced by the avalanche formed on the anode wire. The gas contained into the chamber is a mixture of Ar(30%)-CO_2(50%)-CF_4(20%), the wires are supplied by 2.6 kV and the gas gain is about 10^4. The spatial resolution obtained is lower than 60 μm. As the MDT chambers, a CSC chamber is made of two planes, each made of four layers of detector.

CSC detectors are used at about ~ 7 m from the interaction point, in regions with high pseudorapidity, where a high flux of particles is expected. The main characteristic of the CSC detectors are a small drift time of the electrons (30 ns), a good timing resolution (7 ns) and a low sensitivity to the neutrons. Moreover, using strips orthogonal to the cathode strips, a measure of the second coordinate can be also performed.

Thin Gap Chambers (TGC)

Thin Gap Chambers [14] [15] are used for trigger purposes in the end-cap regions because of their very good rate capability and their long ageing characteristics. TGC are, like CSC detectors, multiwire proportional chambers. The gas mixture is made of 55% CO_2 and of 45% n-pentane (n-C_5H_{12}) and the chambers operate at an high voltage of about 3 kV. The small distance between wires and the electric field configuration ensure a short drift time and a good timing resolution of about 4 ns.

The anode wires, with a 50 μm diameter, are arranged parallel to the MDT wires and produce the trigger signal; read-out strips are orthogonal to the wires and are used to measure the second coordinate. TGC chambers have both a trigger purpose, in the end-cap region, and a precision chambers purpose (for the measure of the second coordinate) in the forward region, in the inner and middle stations of the spectrometer.

In particular, in the end-cap region, the detectors are installed at a distance of bout 14 m from the interaction point and placed on three different planes to form a triplet (M1) and two doublets (M2 and M3) of chambers. In Figure 11 is shown the TGC disposition in the end-cap regions.

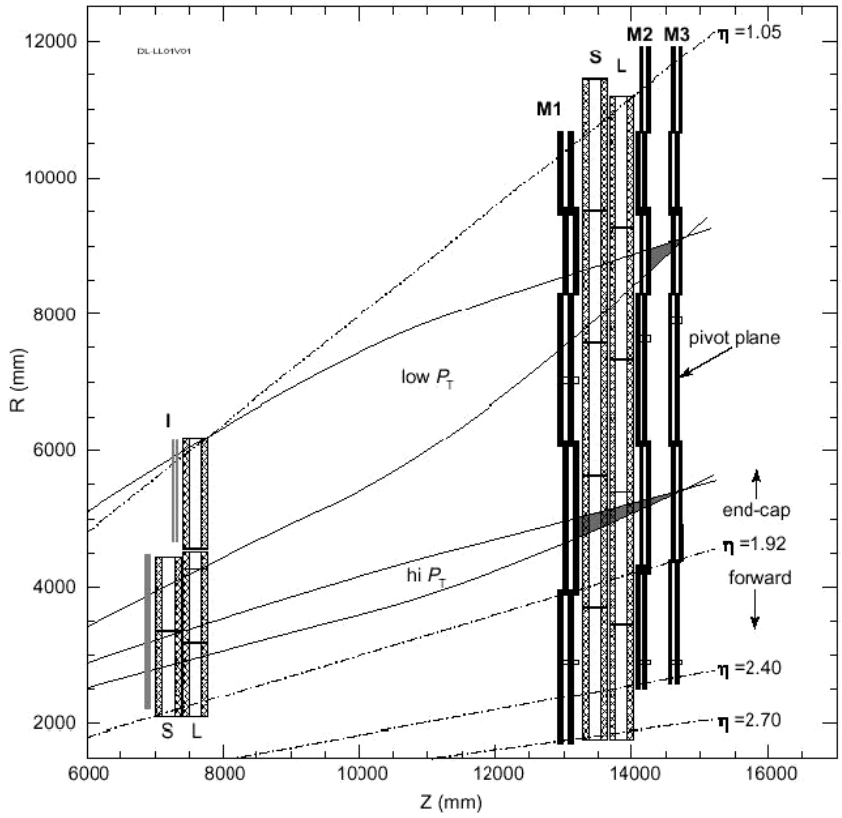


Figure 11 – Disposition of the Thin Gap Chambers in the end-cap region.

Resistive Plate Chambers (RPC)

Resistive Plate Chambers (RPC) detectors [16] [17] [18] are installed in the barrel region of the apparatus, for trigger purposes. RPCs are detectors that guarantee good timing performances at a moderate cost. Their typical timing and spatial resolution are respectively 1 ns ed 1 cm with an intrinsic detection efficiency greater than 98%.

RPCs are ionization detectors whose working principle is based on the discharge inside the gas. Figure 12 shows the section of an RPC detector. Two Bakelite planes delimit a gas gap in which a uniform electric field is applied, whose intensity is about 4.5 kV/mm. The bakelite planes, also called resistive plates because of their high resistivity (ρ = 1010 Ω cm), are externally coated with two thin graphite layers, connected respectively to high voltage and to ground.

A thin layer (few hundred microns) of PET is glued on the graphite electrodes in order to insulate the high voltage electrodes from the read-out strips, oriented in the X and Y directions.

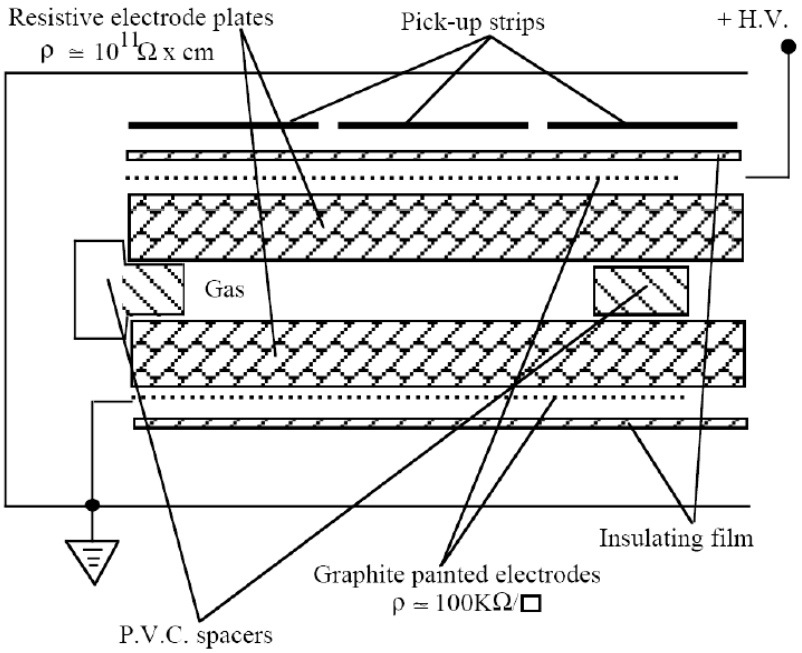


Figure 12 – Section of an RPC.

The graphite electrodes, because of their high resistance, are transparent to the electrical pulse created inside the gas gap, when a charged particle crosses the gap itself. For this reason, the signal can be read by induction on the read-out metallic plates, made up of copper strips.

The RPC just described has the structure of a two dielectric planar capacitor. The working model of the chamber is shown in Figure 13. The capacitors C and C_{gas} and the resistors R and R_{gas} describe the capacitance and the resistance of the bakelite planes and of the gas gap, respectively.

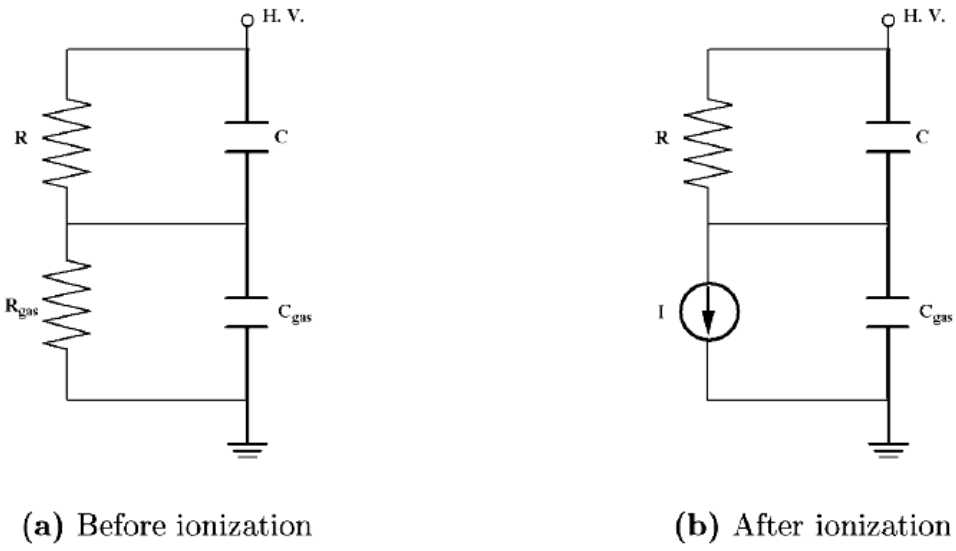


Figure 13 – Working scheme of an RPC detector.

In steady condition, when there is no ionization, the gas has infinite resistance ($R_{gas} = \infty$). So the high voltage HV is entirely applied on the gas gap.

When there is an interaction with charged particles, there is the production of electrons by ionization and the gas gap behaves like an ideal current generator. This current generator discharges the "gas capacitor" C_{gas} and some of the HV is gradually transferred to the resistive plates, described by the C capacitance. The system goes back to the initial conditions following an exponential law with a time constant:

$$\tau = R (C + C_{gas}) = \rho \varepsilon_0 (\varepsilon_r + 2 d / g)$$

where ε_r is the dielectric relative constant of the bakelite, d is the thickness of the bakelite plates and g is the thickness of the gas gap. For $\rho = 10^{10}$ Ω cm, the previous relationship gives τ = 10 ms. This value has to be compared with the time of production of the discharge, that is only ~ 10 ns. In this time interval, the electrode plates behave like insulators and the voltage on the gas gap is too low to feed the discharge. This quenching mechanism is at the basis of the working principle of the detector.

A chamber with resistive electrode plates can be divided in a large number of small "discharge cells" [19], each one independent from the others. An estimation of the extension of the area where the discharge is located is given by the formula $S = Q g / \varepsilon_0 V$, where Q is the charge delivered in the gas.

Therefore, the detector is "blind" only in a region of extension S, for a time τ. By this, the total charge produced inside the detector, and so the intensity of the electric field, has a direct influence on the rate capability of an RPC. A small value of Q has two important effects: it allows to have small currents in the detector (a crucial point to prevent detector ageing) and to have a high detecting efficiency, also when there is a high flux of ionizing particles.

The current produced by the discharge in the gas induces a signal on the pick-up electrode. The pick-up electrodes can be shaped as strips or square pads. The strip has the advantage to behave as signal transmission line of well defined impedance, allowing the transmission at large distances with a small loss of amplitude and timing information. Two orthogonal sets of strips are used; the pitch of every strip is 2-3 cm. Two contiguous strips are separated by a

0.3 mm wire, connected to ground to reduce the capacitive coupling of two adjacent strips.

The equivalent circuit of the read-out electrode (shown in Figure 14) can be described as a current generator, charging a capacitor C in parallel to a resistor R, where C is the electrode's capacitance and R is the resistance between the electrode and the ground.

Typical values for C and R are: C ~ 1 pF and R ~ 25 Ω; the time constant of the circuit results to be $\tau' = 25$ ps, much smaller than the rising time of the signal. So the read-out signal is not integrated, but the current circulating in the read-out electrode is every instant proportional to the current produced by the discharge inside the gas.

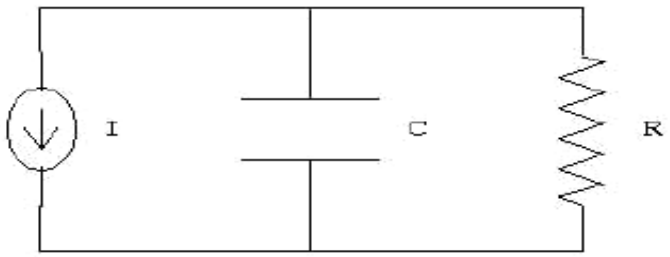


Figure 14 – Equivalent circuit of the read-out electrode of an RPC strip.

1.3.5 The trigger and the Data acquisition (DAQ) systems

Because of the enormous expected background, the design of an extremely selective trigger is needed. The interesting physics processes must be selected and accepted with high efficiency. For example, at the nominal luminosity of LHC, searching for a Higgs boson with a mass of ~ 80 – 100 GeV, one interesting event is expected every 10^{13} produced.

This is achieved by defining three different trigger levels (LVL1 [20], LVL2 and LVL3 [21], also called Event Filter). The first level receives data only from the detector, whilst each of the two following levels elaborates data from the detector and also the information collected at the previous level.

The architecture of the trigger system [22] and the three levels of the trigger are shown in Figure 15.

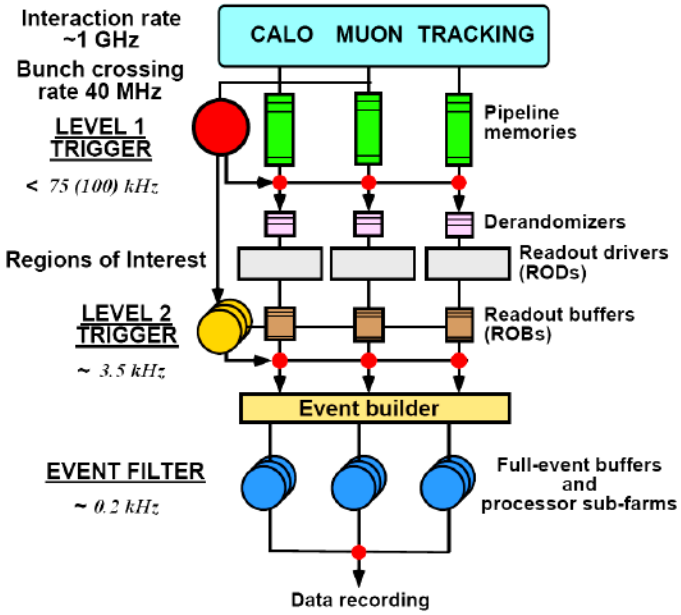


Figure 15 – The three levels of the trigger system.

The first level trigger must identify, without ambiguity, the bunch crossing of the interesting event, using only data coming from the calorimeters and from the spectrometer, and uses synchronous parallel processors working at the bunch crossing frequency 40 MHz. Programmable logic devices (FPGA e CPLD) and ASIC (Application Specific Integrated Circuit) are massively used.

Data produced by the detector are elaborated by the level-1 trigger, with a latency lower than 2.5 μs; during this time, information are stored in FIFO memories (First-In First-Out). The output frequency of data is 75 KHz, increasable up to 100 Khz.

In particular, the level-1 trigger structure (Figure 16), is composed by four subsystems:

— the *Muon Trigger Processor* receives data from TGC and RPC, respectively from the end-cap and the barrel regions; it detects events that have in the final states muons with a transverse momentum greater than a programmable threshold and identifies the spatial region the muon comes from;
— the *Calorimeter Trigger Processor* identifies electrons, photons, hadrons and jets, and measures the possible missing energy;
— the *Central Trigger Processor* collects and elaborates information from the trigger processors;

— the *TTC distribution* system (Timing, Trigger and Control) generates and distributes the control signals to the whole apparatus.

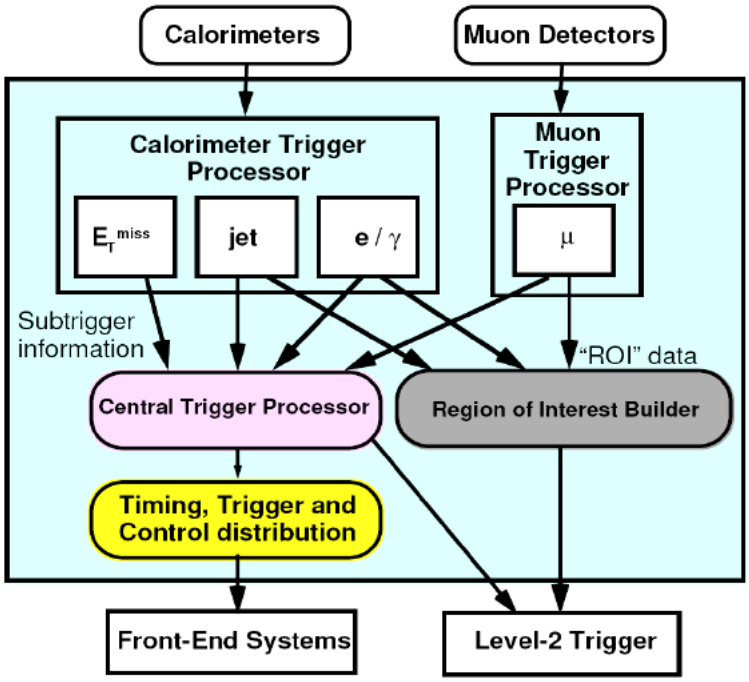


Figure 16 – The level-1 trigger.

When an event is accepted by the LVL1, a level-1 trigger signal is generated by the trigger processor: in this case, data stored in the L1FIFO buffer are transferred to the next elaboration levels. Data are read out from the front-end electronics systems of the detectors into Read Out Drivers (RODs) and then forwarded to Read Out Buffers (ROBs). Figure 17 shows the path of data from the detector to the elaborators, for the RPC detectors.

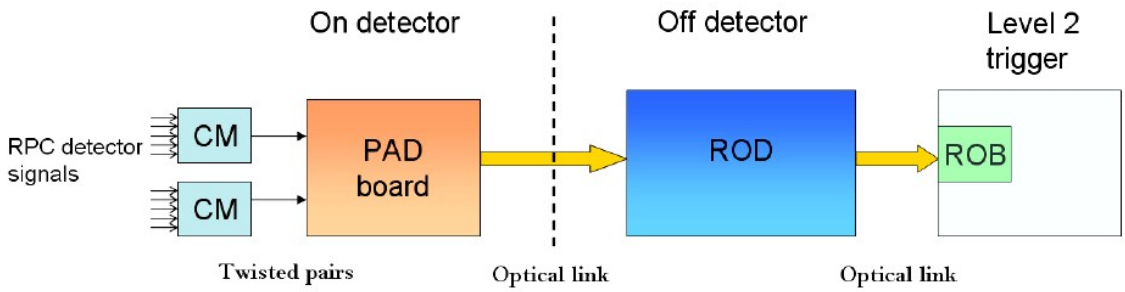


Figure 17 – The path of data from the detector to the elaborators.

All the existing electronics present in the trigger and DAQ data path from the on-detector ones up to the Read Out Drivers are developed specifically for the detector on which they are installed. The other electronics of the DAQ environment are the same for all the detectors. For example, the Read Out Drivers of the Muon Spectrometer can't be switched with the ones developed for the Calorimetric system, while the Read Out Buffers can.

The trigger system is divided into Regions of Interest (*ROI*), with a granularity ~ $\Delta\eta$ x $\Delta\varphi$= 0.1 x 0.1. For example, in the barrel region, the trigger system is divided into 1664 ROI. The level-1 trigger generates and transmits to the level-2 trigger processors the coordinates of the ROIs where an interesting event occurred. In this way, the level-2 trigger accesses only to that ROIs, reducing the amount of data to be elaborated. At this stage, full granularity and full precision data from most of the sub-detectors are available and are used by the LVL2 to select interesting physics events. Moreover, even if it is possible to access all the data of the event, only the information needed to decide whether accept or reject that event are acquired.

Data for the bunch crossing selected by the LVL1 trigger are stored in the ROBs during the LVL2 trigger latency (expected in the range of 1-10 ms). Then, if data are related to an interesting event, the LVL2 trigger promotes them to the following trigger level (otherwise data are deleted). The data elaboration, formatting and storing operations are called *event building*.

Whilst the level-1 trigger is based on an hardware designed for the specific application, both level-2 and Event Filter use commercial processors; moreover, the structures of computing and communication are quite similar. The differences between the level-2 and Event Filter are that level-2 trigger needs simple and fast algorithms, and the Event Filter uses elaboration algorithms similar to the ones used for the off-line analysis.

The level-2 trigger system is made of a sub-farm of commercial processors and the expected trigger frequency is ~ 3.5 KHz. An event accepted by the level-2 trigger is built by the Event Filter. The Event Filter uses offline algorithms, based on up to date calibration and alignment information, such as the magnetic field map. Most of the rejection power of the Event Filter comes from the use of complex algorithms and criteria that cannot be performed by the LVL2 trigger, because of processing time limits. The Event Filter sends data to

the mass memories with a bandwidth of ~ 10 – 200 Mbyte/s; a single event size should be ~ 1 Mbyte, so the output frequency of the Event Filter should be of ~ 200 Hz.

CHAPTER TWO – THE LEVEL 1 TRIGGER AND THE DATA ACQUISITION SYSTEM OF THE ATLAS MUON SPECTROMETER

This chapter describes the architectures of the first level trigger and of the Data Acquisition system (DAQ) of the ATLAS muon spectrometer.

The signals generated by the RPCs of the spectrometer when muons are detected undergo several elaborations before being sent via optical fibres to the counting room USA15. The algorithms of the coincidence matrix, for the generation of the trigger, and the structures of the other acquisition boards are described.

In this chapter, the transmission system on optical fibre will be presented. The receivers of such system produce data input to the Read Out Driver (ROD), a VME board whose main tasks are to receive and process trigger and data signals, to format them into frames in order to send them to the next electronics.

The ROD will be deeply described in the next chapter.

2.1 Overview of the Level 1 trigger system

The ATLAS trigger and data-acquisition system is based on three levels of online event selection. Each trigger level refines the decisions made at the previous level and, where necessary, applies additional selection criteria. Starting from an initial bunch-crossing rate of 40 MHz, the rate of selected events must be reduced to ~100 Hz for permanent storage. While this requires an overall rejection factor of 10^7 against ‘minimum-bias’ processes, excellent

efficiency must be retained for the rare new physics, such as Higgs boson decays, that is sought in ATLAS [23].

High transverse-momentum muons are identified using only Resistive Plate Chambers (RPCs) in the barrel and Thin Gap Chambers (TGCs) in the end-caps.

An essential requirement on the LVL1 trigger is that it should uniquely identify the bunch crossing of interest. Given the short (25 ns) bunch crossing interval, this is a non-trivial consideration. In the case of the muon trigger, the physical size of the muon spectrometer implies times-of-flight comparable to the bunch crossing period. In order to select about 75000 events per second from the 10^9 expected, the ATLAS level one trigger system has to be extremely selective and efficient. The requirements are:

- reconstruction of the track and discrimination of the event within the highest allowed LVL1 trigger latency of the experiment (2.5 μs);
- identification of the bunch-crossing of interesting events;
- identification of the Region Of Interest (ROI) of interesting events;
- high acceptance in pseudo-rapidity ($|\eta| < 2.4$).

For the physics researches at the LHC, two kinds of interesting events should be taken into account in the design of the trigger system of the spectrometer: the production of muons with transverse momentum p_T greater than 6 GeV/c in low luminosity regime, that will be indicated as *low* p_T events; the production of muons with values of transverse momentum p_T greater than 20 GeV/c, in high luminosity regime, that will be labelled as *high* p_T events. The spectrometer's RPCs are designed for identifying both low and high p_T events. In the Figure 18 a section of the barrel spectrometer is shown. In such figure, both the high p_T and the low p_T coincidence RPC planes are schematized. The first one is made up of a single layer of RPCs sub-detectors, while to reveal low pT muons two close RPCs layers are needed. The trigger algorithm will be described in section 2.3.

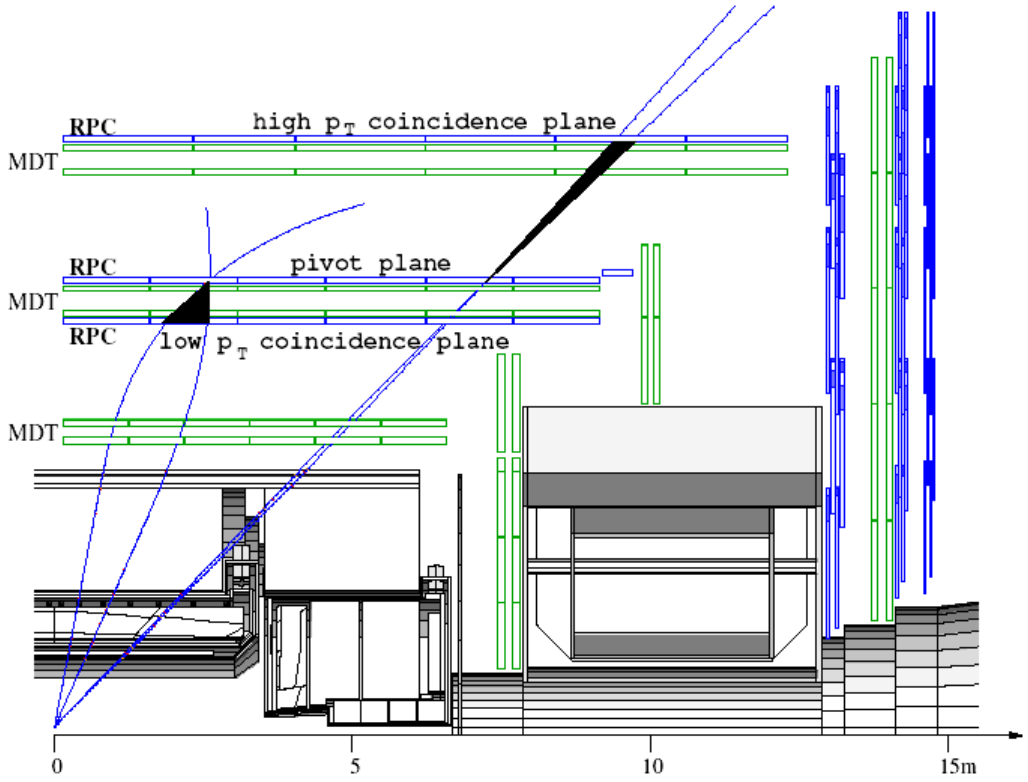


Figure 18 – Section of the barrel muon spectrometer.

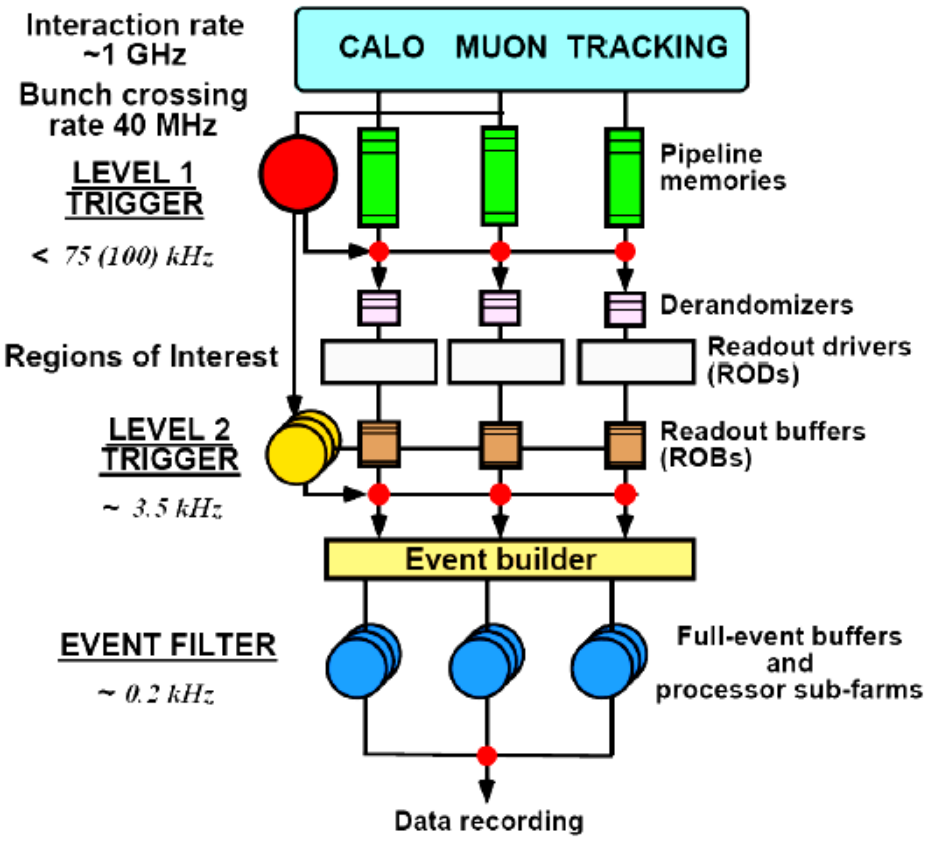


Figure 19 – The three levels of the trigger and the DAQ system.

2.2 The trigger and DAQ system architecture

In Figure 19 you can see the block diagram of the trigger and the Data Acquisition (DAQ) system architecture. Data produced by the detector are elaborated by the level-1 trigger, with a latency lower than 2.5 μs. During this time, information are stored in pipelines memories. The output frequency of data goes from 75 KHz up to 100 Khz.

In particular, the level-1 trigger structure is composed by four subsystems:

- the *Muon Trigger Processor* receives data from TGC and RPC; it detects events that have in the final states muons with high p_T.
- the *Calorimeter Trigger Processor* identifies electrons, photons, hadrons and jets, and measures the missing energy.
- the *Central Trigger Processor* collects and elaborates information from the trigger processors.
- the *Timing Trigger Control* (TTC) system generates and distributes the control signals to the whole apparatus.

When an event is accepted by the LVL1, the trigger processor generates a LVL1 accept signal: in this case, data stored in the L1FIFO buffer are transferred to the next elaboration levels. Data are read out from the front-end electronics of the detectors and sent to Read Out Drivers (RODs). Then they are forwarded to Read Out Buffers (ROBs).

In order to reduce the amount of data to be elaborated by the LVL2 trigger processors, the trigger system is divided into ROIs (Region of Interest), small detector areas with a granularity of $\Delta\varphi \, \Delta\eta = 0.1 \times 0.1$. The level-1 trigger generates and transmits to the second level only the coordinates of the ROIs where an interesting event occurred. This strategy allows to transmit only the information needed to decide whether accept or reject the occurred event.

Data for the bunch crossing selected by the LVL1 trigger are stored in the ROBs during the LVL2 trigger latency. Then the LVL2 trigger promotes such data (if considered interesting) to the following trigger level, called *Event Filter*.

The level-2 trigger frequency is ~ 3.5 KHz. An event accepted by the level-2 trigger is built by the *Event Filter*. The *Event Filter* uses offline algorithms, based

on calibration and alignment information, such as the magnetic field map. After its work, the Event Filter sends data to the mass memories with a rate between 10 and 200 Mbyte/s. As a single event size is around 1 Mbyte, the output frequency of the Event Filter is ~ 200 Hz.

2.3 The LVL1 trigger algorithm for muon detection

The aim of the trigger system of the muons spectrometer of the ATLAS apparatus is the detection of particles coming from the interaction vertex having a transverse momentum greater than a certain programmable threshold.

The muon trajectories are deflected into the apparatus by the toroidal magnetic field. The curvature will depend firstly by the momentum of the particle. In Figure 20 it's shown the functioning principle of the trigger system.

Let's suppose that our will is to detect a muon using two different stations of the spectrometer, called Layer 1 and 2.

P_1 and P_2 are the impact points respectively in the first and in the second Layer. Let the dotted line crossing P_2 and the interaction vertex be the axis of a cone with the vertex in P_2 (such axis could be seen as the trajectory of a particle in the limit p = ∞).

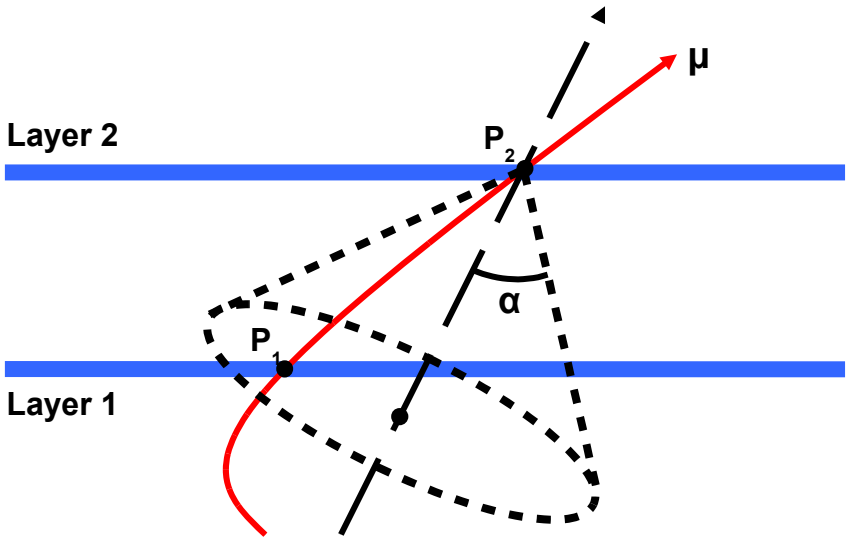


Figure 20 – Schematization of the trigger system functioning principle.

All the particles detected by the first station inside the cone volume have a momentum greater than a certain threshold p_t, related to the cone opening angle α. The greater is the threshold, the lower is the angle. In the spectrometer, the position of the particle in the station is given by the position of the read-out electrodes on which the signal is induced by the muon.

The trigger algorithm is based upon the signals of the two layers, taken with opportune timing coincidences: for example if the hit on the layer 2 is detected *before* the one on the layer 1, the trigger doesn't give the accept signal, because the only allowed muon trajectories go from the interaction point toward the extern of the spectrometer.

The coincidence conditions are written into a programmable memory that acts as a matrix whose raw and column indexes represent the positions of the strips on the detectors (Figure 21). The matrix, programmed according to the chosen trigger condition, indicates the validity (or not) of the coincidence between the signals corresponding to the matrix elements.

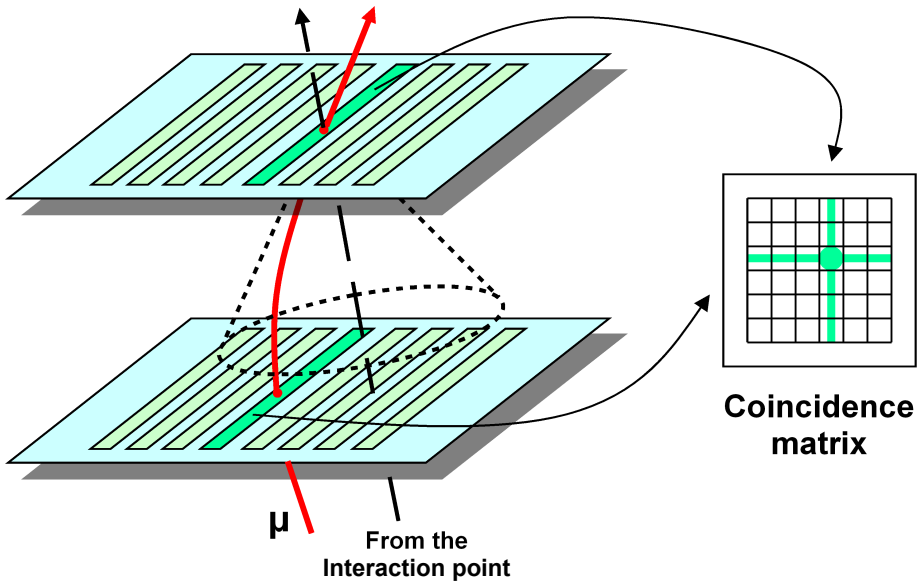


Figure 21 – The muon trajectories are reconstructed with a coincidence matrix between the detectors read-out strips' positions.

2.4 The segmentation of the muon spectrometer

The barrel muon spectrometer has been physically segmented in 16 different sectors, as shown in Figure 22. The chamber towers labelled with even numbers are built “around” the coils of the octagon-shaped magnetic system. These sectors are called *Small Sectors*, while the ones labelled with odd numbers are called *Large Sectors*.

The trigger and data acquisition system is segmented into 64 logic sectors: each of the 16 physical sectors (*Large* or S*mall*) of the spectrometer is divided in two, and a further division is done for $\eta > 0$ and for $\eta < 0$.

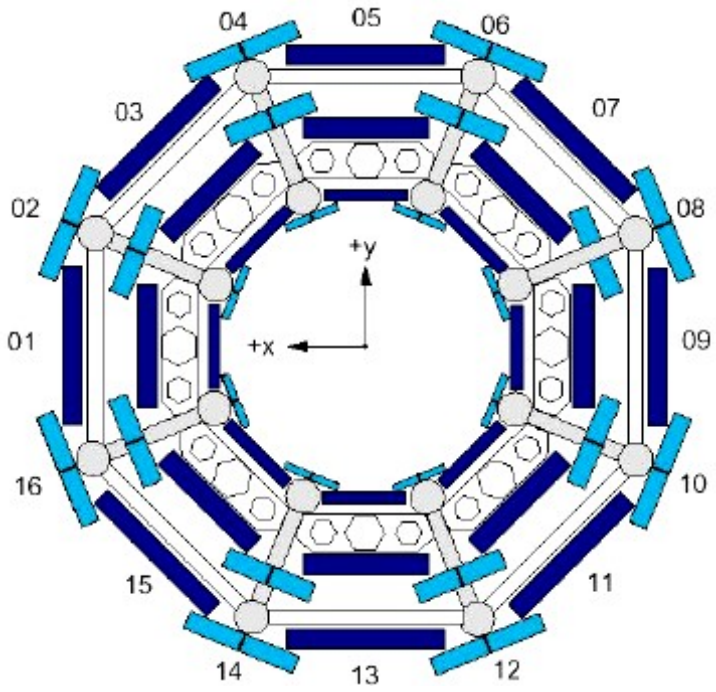


Figure 22 – The spectrometer segmentation in the barrel region.

2.5 The Resistive Plate Chambers

The RPC detectors are assembled in three different stations in each of the 16 sectors of the spectrometer, as you can see in Figure 23. The RPC1 and RPC2 stations are installed on the top and the bottom of the middle MDT precision chamber, labelled as BMS (Barrel Middle Small); the RPC3 station is at a greater distance from the beam axis, upon the MDT BOS (Barrel Outer Small) chamber.

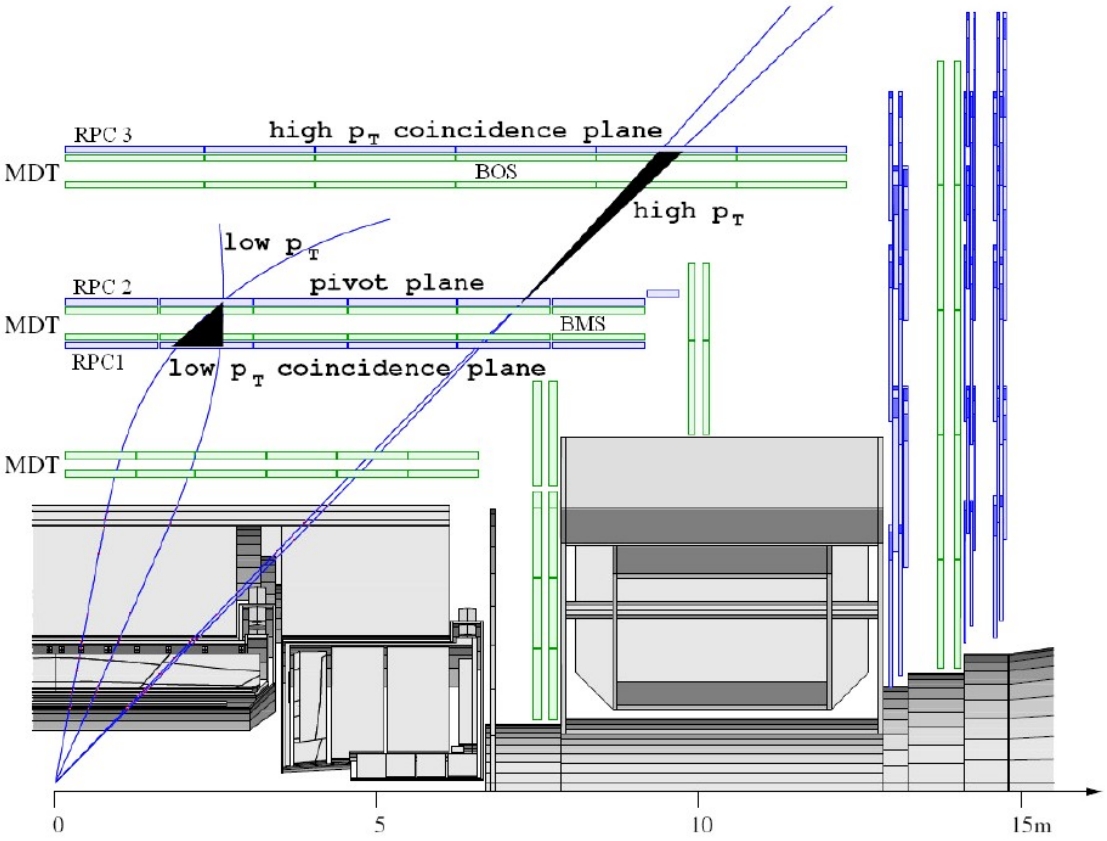


Figure 23 – Section of the muon spectrometer.

A *low* p_T event (p_T < 6 GeV/c) is detected if a signal is seen in the RPC1 station and one is detected in the RPC2 station, in the programmed trigger window. An *high* p_T event is detected if a *low* p_T coincidence is found and also a signal in the RPC3 station is detected. Both for *high* p_T and *low* p_T events, the time coincidence between signals must to be in a 20 ns interval.

Each RPC detector is made of two *RPC units* glued together. The junction point is shown in the upper side of Figure 24. The longest dimensions of the various units are limited by the propagation delay of the signals on the read-out strips of the chambers. In order to allow the correct identification of the bunch-crossing of the event, this delay must be lower than 11 ns.

Each RPC unit is made of two *RPC chamber* separated by insulating polycarbonate spacers. This structure is glued to a thick polystyrene panel that supports it (Figure 24).

Each RPC chamber is realized with two bakelite planes containing a gas mixture (95% $C_2H_2F_4$, 4% C_4H_{10}, 1% SF_6). The RPC is read-out by parallel electrode strips, glued to the surface of the bakelite plane.

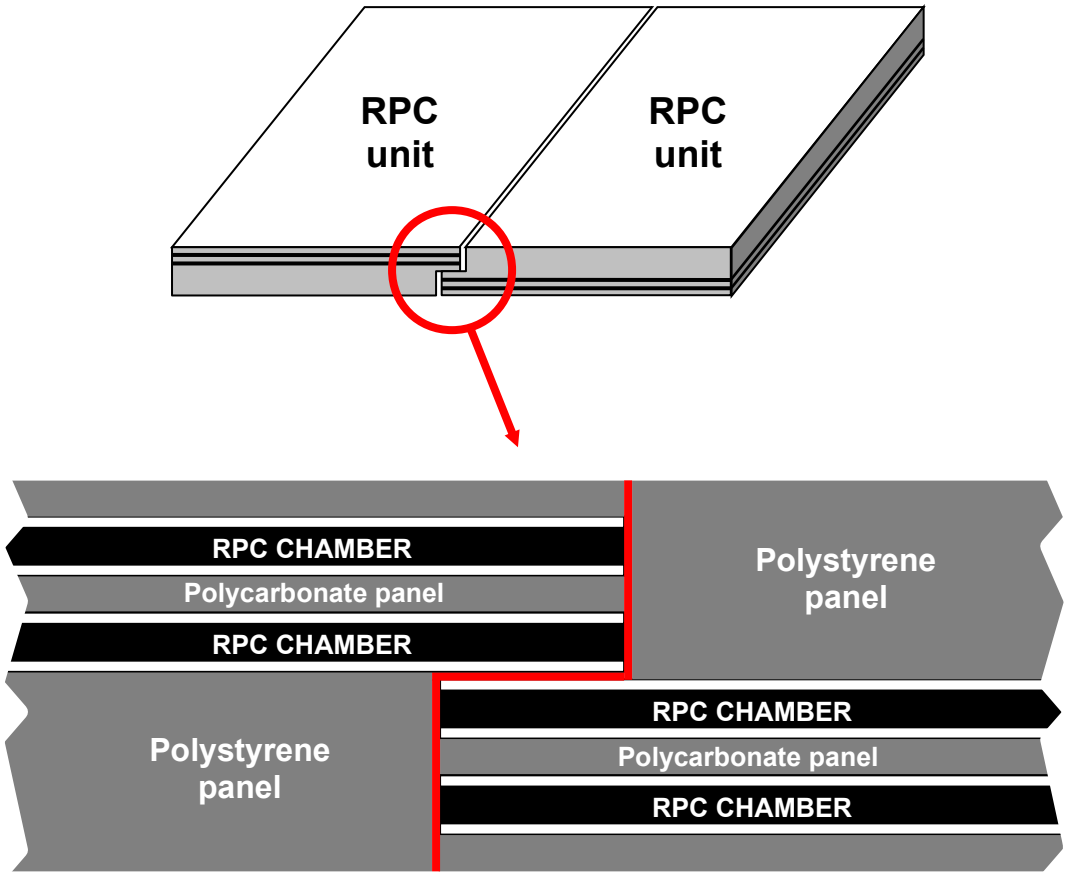


Figure 24 – RPC detector panel, made up of two units.

The RPC chambers are installed upon the RPC unit so that their read-out strips are mutually orthogonal. The strips oriented in the azimuthal direction, parallel to the magnetic field lines, measure the coordinates in the η projection, in the r-z plane. Such strips will be indicated as η strips. The strips oriented in the longitudinal direction, and so parallel to the beam axis, measure the azimuthal coordinates in the φ projection, in the x-y plane. Such strips will be indicated as φ strips.

The RPC detectors just described allow to locate the muon hit point with a resolution of ~ 10 mm.

2.6 The synchronization of the apparatus

In the beams of the LHC collider [24], protons are grouped in “bunches” of 10^{11} particles that interact every 25 ns. Such interaction is called *bunch crossing*. As said before, the ATLAS detector is made up of several sub

detectors. In order to correctly reconstruct and correlate data generated by all these sources, a very precise synchronization system have been developed. The entire ATLAS apparatus is a system working at the *bunch crossing* frequency of the LHC (40 MHz).

The DAQ and trigger systems are driven by a common clock signal, synchronous to the *bunch crossing* frequency of the collider. A periodic signal, called *orbit*, is associated to the revolution of the proton bunches in the LHC (Figure 25). This signal, generated by the control system of the LHC collider, has a period of 88924 ns and is made of 3564 elements (called *trains*), which are bursts of 72 bunches. Some "holes" (called *missing trains*), due to PS injector dead time, are visible inside the orbit. Such holes of different amplitude are crucial in order to establish the correct synchronization between the LHC orbit and the experiments' data acquisition systems.

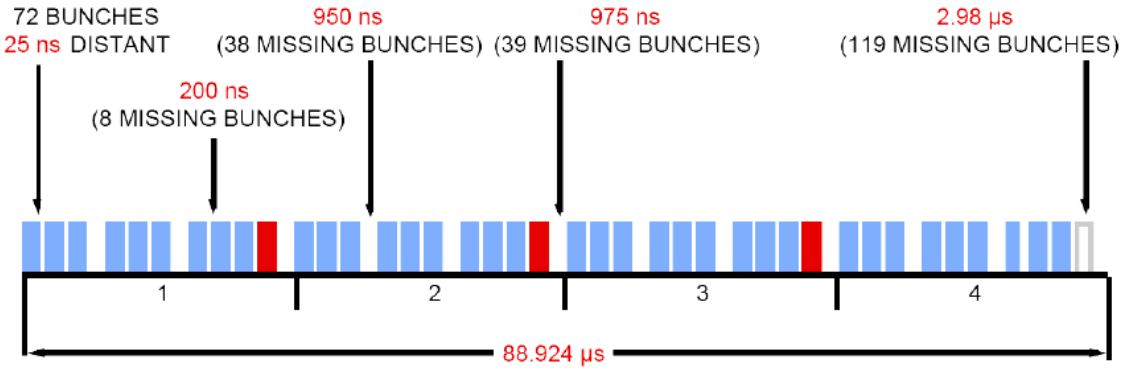


Figure 25 – The bunches' distribution in the LHC orbit signal.

In order to achieve the synchronization between the elaboration systems and the machine clock it has been built a transmission system, able to distribute the clock signal and all the other control signals to all the elements of the ATLAS apparatus. Because of the large dimensions of the detector, the reference clock signal must be transmitted over distances up to hundreds meters, in order to reach all the parts of the apparatus. This is one of the problems that have been faced to make the entire apparatus working at 40 MHz. The clock phase is controlled at a sub-nanosecond level, to avoid phase discrepancy between all the sub-systems of the ATLAS apparatus. So there is the essential need to use a system that allows a resynchronization procedure and that guarantees a clock recovery in the event of a loss of synchronization.

2.6.1 The Timing, Trigger and Control system

In Figure 26 it is shown the control architecture [25] designed for the LHC collider, in particular the Trigger Control System (TCS) and the system Timing, Trigger and Control (TTC). The Trigger Control System generates the timing and trigger signals. It receives from the LHC machine the clock and the orbit signals and receives from the Central Trigger Processor (CTP) of the ATLAS experiment the signal of validation of the Level 1 trigger (L1A, that means Level 1 Accept).

The TCS also manages the Reset signals, used to recovery the synchronization in case of a loss of information, and some service signals. A further task of the TCS is the control and the management of the calibration, synchronization and test signals for the apparatus subsystems.

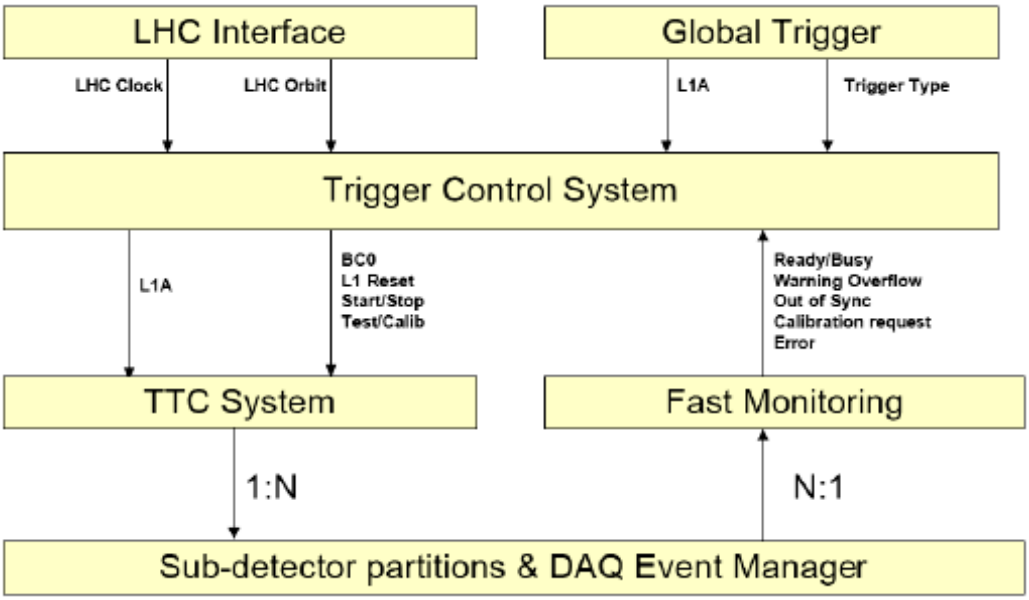


Figure 26 – The control architecture of the LHC.

The Trigger Timing and Control [26] system is responsible of the distribution of the timing and trigger signals to the entire detector and to the different entities of data elaboration. Moreover, in order to make a correct identification of interesting events, the on-detector electronics associate to data in each event a unique progressive number (Event Identifier, or EVID) and a number identifying the bunch crossing that generated the collision (Bunch Crossing Identifier, or

BCID). Such signals, together with the 40 MHz clock signal, the orbit signal, the L1A and other service signals, are sent from the TCS to the TTC. All these signals are coded and optically transmitted to the elaboration systems and toward different destinations in the ATLAS apparatus (Figure 27). Signals are reconstructed at the destination and are adapted to the protocols of every sub-detector.

The TTC incorporates facilities to compensate for particle flight times and detector electronics propagation delays. In addition it provides simultaneous transmission of synchronized broadcast commands and individually-addressed controls and parameters, such as channel masks and calibration data.

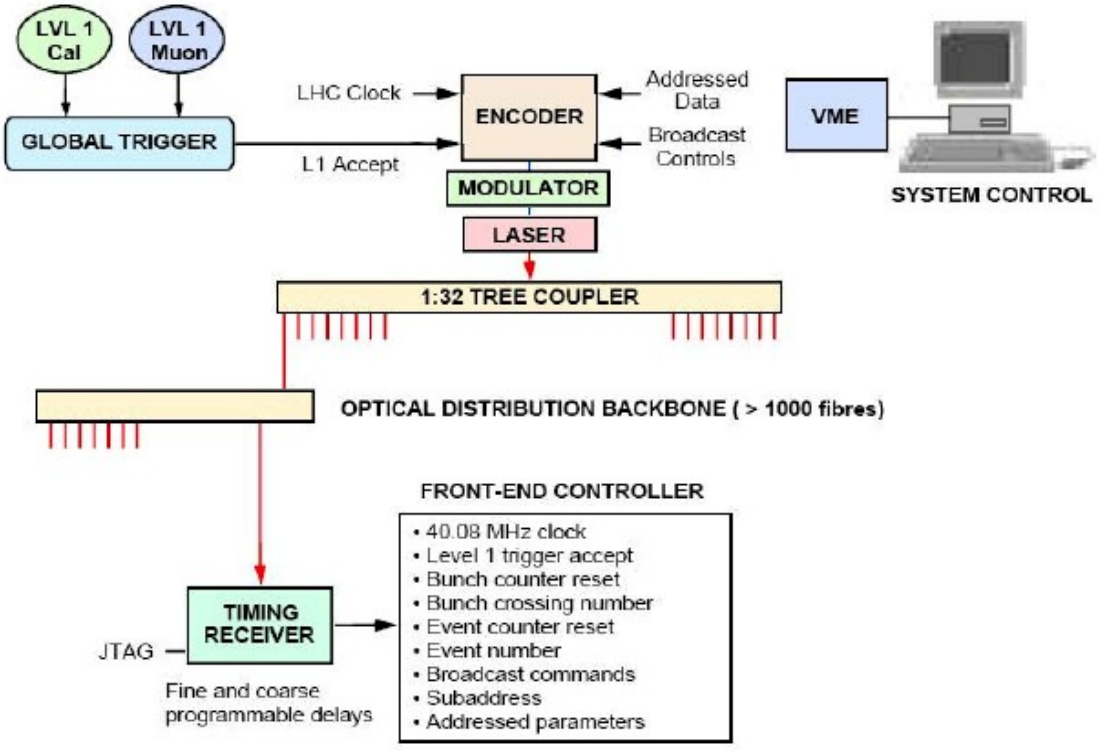


Figure 27 – The TTC system.

At each destination, a special timing receiver (TTCrx) delivers all the signals required by the electronics controllers.

The distribution to the different units of the apparatus is made by the TTCrq board (that hosts the TTCrx ASIC), shown in Figure 28, that reconstructs the signals Clock, L1A, Event Counter Reset and Bunch Counter Reset and makes them available on four different channels, so they can be used by the different elaboration structures.

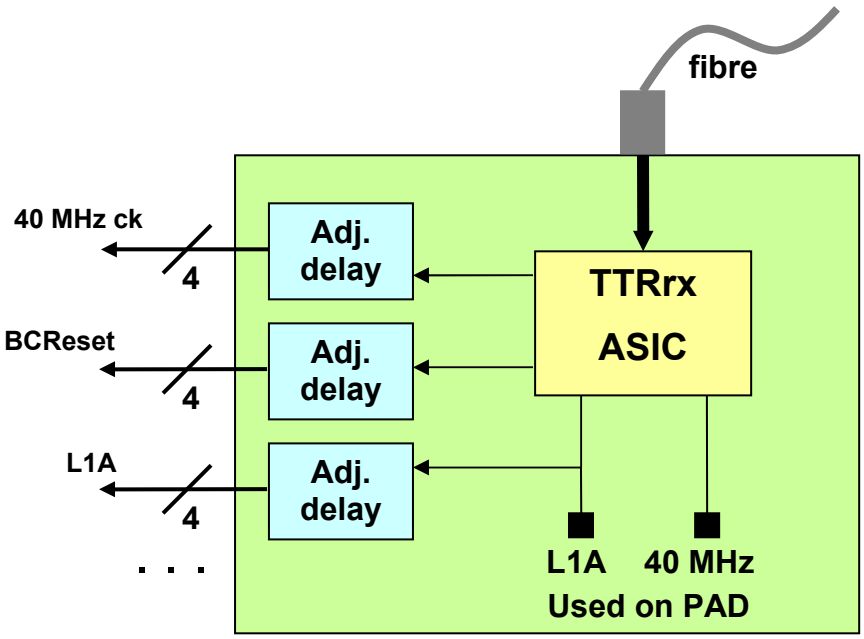


Figure 28 – The TTCrq board.

Starting from the TTCrx signals, the data acquisition boards (on-detector electronics) can generate the Event Identifier and the Bunch Counter Identifier. The EVID, incremented every time that a L1A signal occurs, is coded by 24 bit and is counted starting from the last EVID Reset. The BCID, incremented at every bunch of the orbit, is coded by 12 bit and is counted starting from the last BCID Reset.

2.7 The RPC trigger and DAQ electronics

The electronics of the RPC detectors are made of Amplifier Shaper Discriminator (ASD) boards, of Coincidence Matrixes (CM) and of PAD boards. The signals induced by muons on the RPC electrode strips are received by the ASD boards that amplify, discriminate by using a programmable threshold and shape them in width. Each ASD board houses eight acquisition channels (so eight RPC electrode strips).

The Coincidence Matrixes use the information produced by the ASD boards to execute the two trigger algorithms (*low* p_T and *high* p_T). Data produced by the CMs are elaborated by the PAD boards, that manage also the transmission to the processors of the next trigger and DAQ levels: a simplified path of the readout data and of trigger information is depicted in Figure 29.

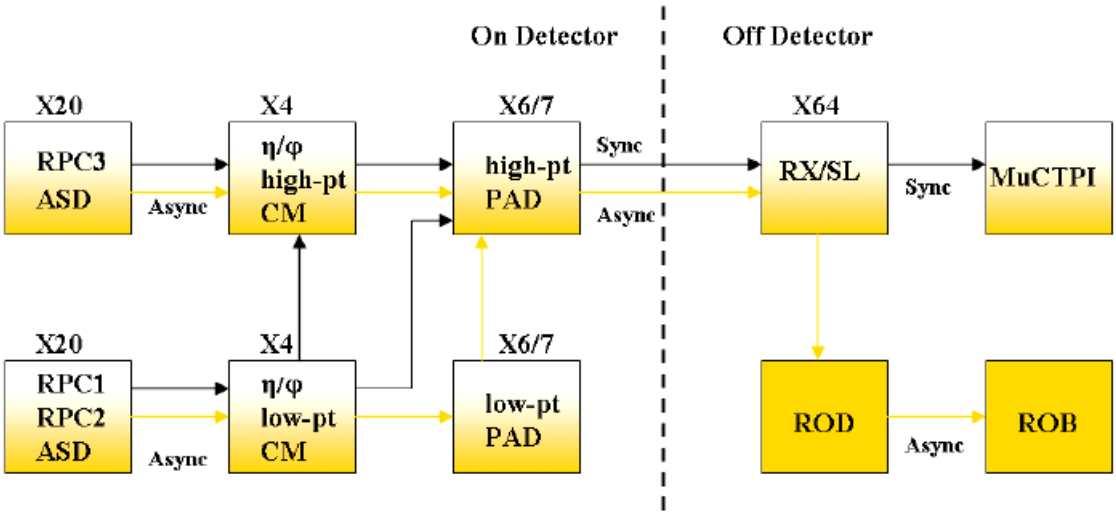


Figure 29 – Scheme of the on-detector and the off-detector electronics.

2.7.1 The Coincidence matrix

The ASD boards, installed on the RPCs, are connected to the Coincidence Matrixes [27]. These lasts are distant few meters and are hosted by the PAD logic boards. Every PAD board hosts four CMs, relative to a region of granularity $\Delta\eta \times \Delta\varphi \approx 0.2 \times 0.2$. It also hosts the modular board TTCrq and the modular board Link Tx, that transmits on optical fibre trigger and readout data to the next DAQ levels. In Figure 30 a picture of a PAD board is shown.

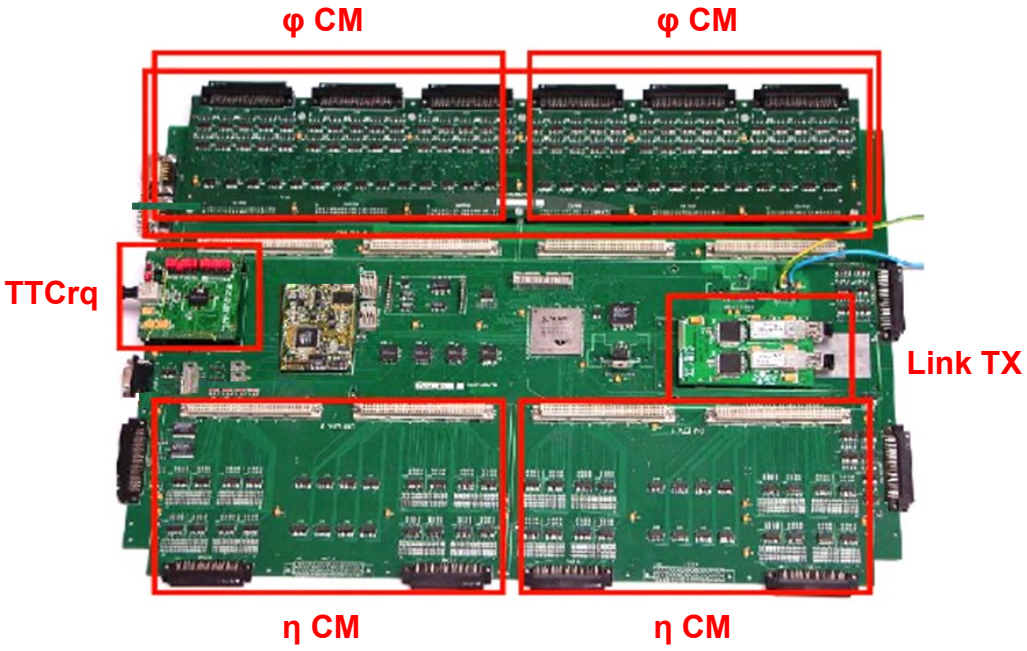


Figure 30 – A PAD board.

As said before, each RPC detector is made up of two RPC chambers glued together, so a “low p_T” event is detected resolving the coincidences defined between the signals of the four RPC chambers of the two inner detector planes. The coincidence is verified in a programmable matrix, depending on the fixed transverse momentum threshold. Every CM board allows to program up to three different trigger conditions. The coincidence can be also programmed with a majority 2/4, 3/4 or 4/4 check between the four detector planes.

As example, with a 3/4 majority, the trigger condition is valid only if at least 3 up to 4 signals received from the two trigger stations are detected, within the trigger window and in a timing coincidence of 20 ns. This majority algorithm partly allows reducing the background noise of the apparatus.

To analyze the “high p_T” events, information from “low p_T” CMs and from the two detector planes of the RPC3 station are transferred to the corresponding “high p_T” CMs, installed on the RPC3 chambers. The coincidence algorithm is the same as before: as for a “low p_T” event, the matrix searches a time coincidence between the signals, within a time interval of 20 ns, with a 2/4, 3/4, 4/4 majority check and in spatial programmable windows (called *roads*) depending on the threshold of the imposed transverse momentum.

Trigger data are elaborated inside the CM board and formatted into a frame. The *header* contains the code that identifies the board (CM) and two identifiers, FEL1ID and FEBCID, produced by the two counters of EVID and of BCID on the board. The data frame is then transmitted to the trigger processors. Read Out data are stored in FIFO (First In First Out) memories and wait to be transmitted (or not) to the next DAQ levels, depending on the decision of the level 1 trigger.

2.7.2 The PAD board

The information of two adjacent “low p_T” CMs in η direction and the corresponding information of the two CMs in φ direction are elaborated by the “low p_T” PAD Logic board. For the “high p_T” trigger algorithm, data arriving to the “high p_T” PAD logic board come from both the CM “low” and from the CM ”high” (Figure 31). The PAD Logic “low p_T” board and the four CM “low p_T” are installed

on the RPC2 station, while the PAD Logic "high p_T" board and the four CM "high p_T" are installed on the RPC3 station.

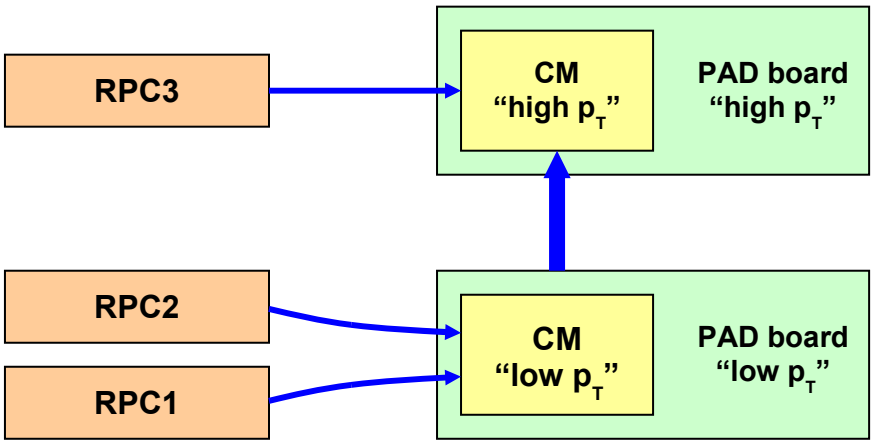


Figure 31 – Data path for the "high p_T" events.

One PAD Logic board covers a granularity region of $\Delta\eta \times \Delta\varphi \approx 0.2 \times 0.2$, whereas the dimension of a Region Of Interest is $\Delta\eta \times \Delta\varphi \approx 0.1 \times 0.1$: so each PAD Logic contains information on 4 ROIs. A *Small* sector of the spectrometer is managed by 7 PADs, while a *Large* one is managed by 6 PADs.

The main task of the PAD board is to perform a further elaboration of read-out and trigger data produced by the CMs. These data are combined and two different frames with different information (for read out and for trigger) are produced and sent via optical fibre to the counting room *USA15*. The other tasks of a PAD board are: the identification of the Region Of Interest for the event validated by the trigger system that combines information both in η and in φ direction; the transmission of trigger information, of the BCID and of other signals. Furthermore, using a TTCrq modular board, each PAD receives the trigger signals from the TTC and distributes them to the four coincidence matrixes and to the PAD logic. In Figure 32 a data flow diagram of the PAD board is shown.

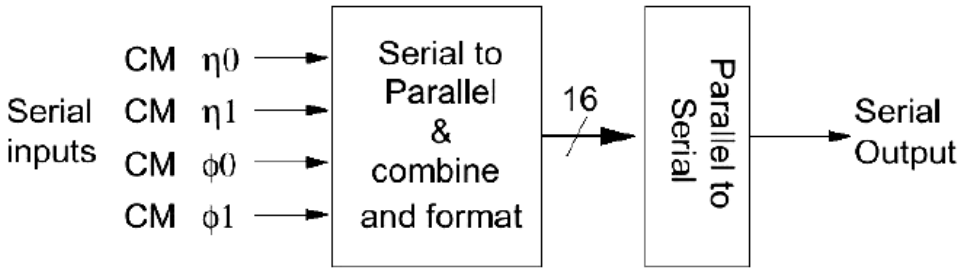


Figure 32 – The PAD board data flow diagram.

2.7.3 The data path architecture

Read-out data and Trigger data from a PAD are transmitted via optical fibre to a RX/SectorLogic (RX/SL) board. The trigger data path must be rigorously synchronous, at the 40 MHz of the LHC machine, but the read out data path is asynchronous, because interesting data (and so the L1A signal that validates them) are not produced in the detector at every bunch crossing. The RX/SL board is therefore made up of two main parts, one asynchronous (the read-out section, named RX) and one synchronous (the trigger section, named SL) with the machine clock. The first one is dedicated to the reception, the elaboration and the transmission of read-out data to the Read Out Driver (ROD) board, while the second one is responsible of the reception of trigger data and of the transmission of them to the Muon Trigger Central Processor Interface (μTCPI).

The ROD is a VME electronic board and it is responsible of the management and the elaboration of the read-out data coming from the PAD boards of a whole sector (*Large* or *Small*) and of the transmission of data to the next TDAQ levels (Figure 33).

The ROD crate is located in the counting room USA15, at about 80 meters from the beam interaction point. Such crate hosts two independent structures, each one made up of two RXSector Logic boards (slaves) and one ROD board (master). Every crate is controlled by a VME central unit CPU. Every ROD is connected, through a custom bus (RODbus), both to the trigger section (SL) and to the read-out section (RX) of every RX/SL board. The trigger section (SL) of the RX/SL board receives trigger data over optical fibre from the PAD boards, transmits its data to the interface to the trigger processor μTCPI (muon Trigger Central Processing Interface) and transfers diagnostic information to the ROD.

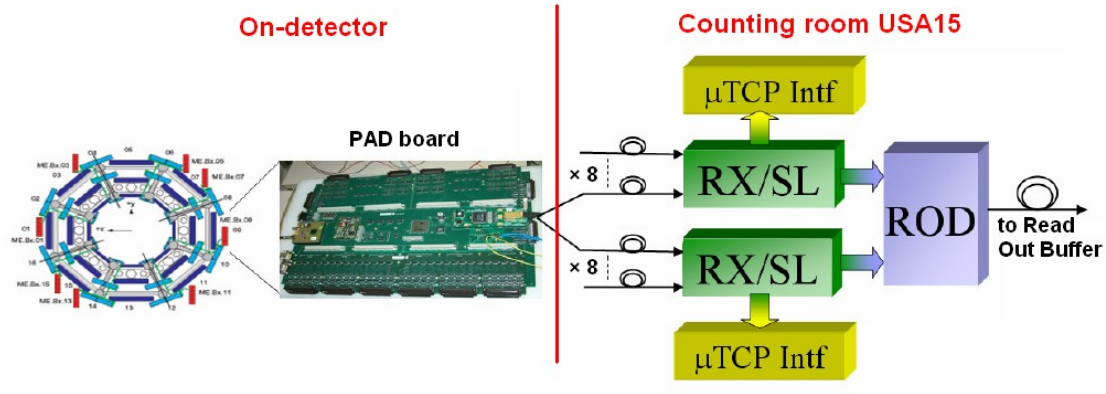


Figure 33 – Data path from the PAD to the ROD.

The read-out section (RX) receives read-out data over optical fibre from up to eight PAD boards and, after elaboration, transmits them to the ROD.

The task of the ROD board is to parse the received data frames, to check their coherence and to build a data structure for all the RPCs of one of the 32 sectors of the spectrometer. Each ROD sends the event fragments to the next DAQ levels for further event building and analysis. The ROD hardware environment and its architecture will be deeply described in the next chapter.

As just said, every ROD manages the information coming from one of the 32 sectors into which the spectrometer is divided (16 towers, each one divided in two sectors, for $\eta > 0$ and $\eta < 0$). To each one of these sectors correspond two logic sectors into which the trigger system is divided (so 64 trigger logic sectors, in total). These lasts are managed by 7 PADs, if it is a *Small* sector, or by 6 PADs, if it is a *Large* sector. Each RX/SL manages the data produced by one of the 64 trigger logic sectors of the spectrometer, receiving information from 6 or 7 PADs. The number of the PAD boards connected to a ROD, through the RX/SL boards, will therefore be given by $6 \times 2 = 12$ (for a *Large* sector) or by $7 \times 2 = 14$ (for a *Small* sector). A total of 32 structures, similar to the one shown in the right side of Figure 33, are needed to manage the flux of trigger and read out data of the whole spectrometer.

The ROD will be described in detail in the next chapter.

CHAPTER THREE – THE READ OUT DRIVER OF THE ATLAS MUON SPECTROMETER RPCs

The ROD is part of the DAQ system of the ATLAS muon spectrometer. in this chapter I describe the hardware environment and the architecture of the board, together with the block diagram of the Event Builder, responsible of the event building process. The different devices present on the board are presented too.

3.1 The ROD board in the DAQ system

The Read Out Driver [28] is a VME board located in one of the off-detector data acquisition subsystem shown in Figure 34. These structures are installed in the USA15 counting room, at about 80 meters from the beam interaction point. In this structure the ROD manages readout data of one of the 32 half-sectors into which the spectrometer is divided (considering the $\eta > 0$ and $\eta < 0$ regions).

Each sector managed by the ROD corresponds to two of the 64 logic sectors into which the trigger system is divided. Each ROD is linked, via a custom bus (RODbus), with two RX-SL boards, that have the task to receive and elaborate trigger and read-out data from the on-detector electronics.

The RODbus is a bus that allows the boards to exchange data, control and timing signals.

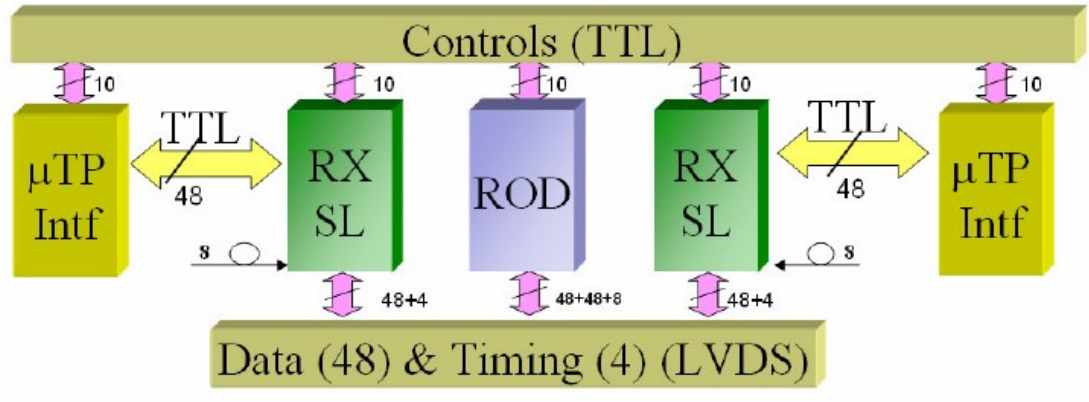


Figure 34 – The Read Out Driver in the DAQ system.

In particular, data and timing signals are transmitted in LVDS standard in order to have high frequency, low skew and low jitter, while control signals (as busy, reset, offline), run at lower rate and are transmitted in TTL standard. In the next paragraphs a detailed description of the RODbus will be given.

3.1.1 The RX section of the RX/Sector Logic board

The RX section of the RX/Sector Logic board receives read-out data from PAD boards, via up to eight optical fibre. Such data are then elaborated and transmitted to the ROD, over the RODbus backplane. The main task of the RX/Sector Logic board is to format in a single frame the readout data coming from different PAD boards. Data are sorted according to the same event number and bunch-crossing parameters and some control bits are added. Events with the same bunch-crossing identifiers, even if arrived to the RX/SL board at different times, are grouped in the same RX Frame. The format of the RX Frame is shown in Figure 35.

The RX section logic also checks the correctness of data frames, analysing header and footer in input from the PAD. The RX/SL board has a VME interface that can be used both for the board configuration and for the test of data transmission to the ROD.

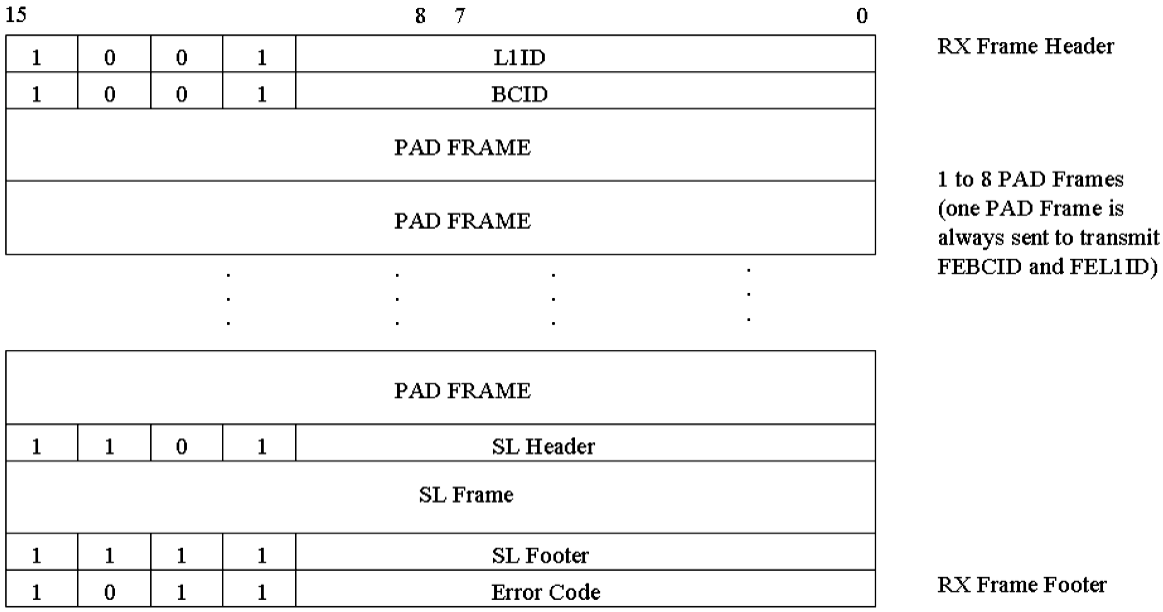


Figure 35 – Format of the RX Frame.

3.1.2 The RODbus

In the subsystem that hosts the ROD board there are two RX/SL boards and two Muon Central Trigger Processor Interfaces, whose task is to interface the trigger section of the nearest RX/SL board to the Muon Central Trigger Processor.

The custom backplane RODbus has been designed to handle all the functionalities needed for this section of the trigger and data acquisition system of the ATLAS spectrometer. Front and rear view of the RODbus are shown in Figure 36. It is a 10 layer PCB (Printed Circuit Board) and it fits into the VME64x rear side: connectors on the bottom are assigned to control signals (i.e. busy, reset, diagnostics) driven in TTL standard; the three upper connectors (that match the J0 connector of the VME64X) host high frequency data and timing signals, that are transmitted in LVDS standard.

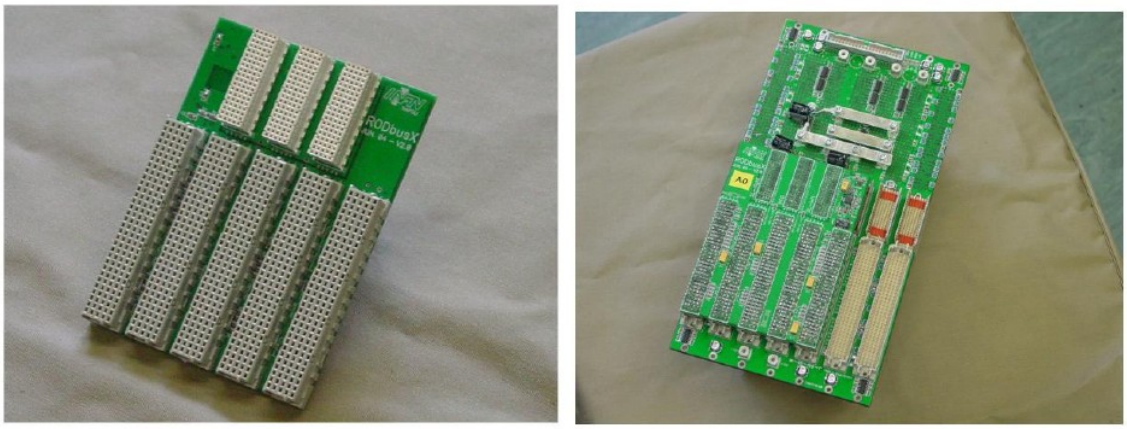


Figure 36 – The RODbus backplane.

As said before, trigger and read-out information arrive on the RX/SL boards, via optical fibre, and read-out data are sent to the ROD via the RODbus: each RX/SL sends 48 bit at 40 MHz. The signals received by the ROD from the LHC (via TTC) are sent on the RODbus too. Such signals are forwarded to RX/SL boards and to the μTCPIs. For each RX/SL there is a 18 bit TTL shared bus for control signals. The RODbus also allows transmission of data from each RX/SL to the nearest μTCPI over a 48bit TTL private bus. On the LVDS domain, the backplane is made of differential pairs, routed as edge-coupled microstrips. The noisy TTL lines are routed on separate planes and connectors and are terminated as VME lines.

3.1.3 The Read Out Driver

The main task of the ROD is to receive information from each of the two RX/SL boards connected to the ROD via RODbus and to perform a further framing, before transmitting data to the next DAQ levels: the Read Out Buffers (ROB). The scheme of the ROD event builder is shown in Figure 37.

The frames coming from each RX/SL have an RX Header, a certain number of data words (payload) and a footer. The two 16-bit Header words contain information on L1A and BCID of the event and are checked by the ROD Event Builder Engine; the payload of each frame is not inspected by ROD.

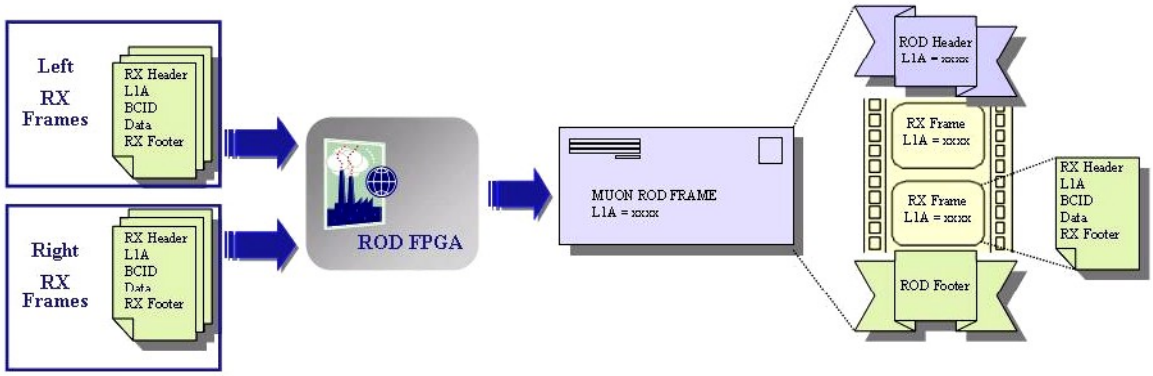


Figure 37 – Scheme of the ROD event builder.

The ROD produces a new frame, the MUON ROD FRAME, that has a ROD Header (pertaining to a specific L1A value), a payload made up of the data coming from the selected RX/SL boards, and a ROD Footer.

In the event building procedure, the ROD performs also a control of syntactic and logical coherence upon the information coming from the RX/SL boards. In particular, the ROD detects errors in data transmission or mismatch between the L1ID and BCID codes (generated by the On-detector boards and embedded in the RX/SL boards frames) and the corresponding codes transmitted by the TTC to the ROD. Data coming from the RX/SL pertaining to the same event number and Bunch Crossing parameters (L1ID and BCID) are selected and written in the payload of the Muon ROD frame. If there is a discrepancy between L1 and/or BCID identifiers, data are transmitted but flagged with one or more error bits.

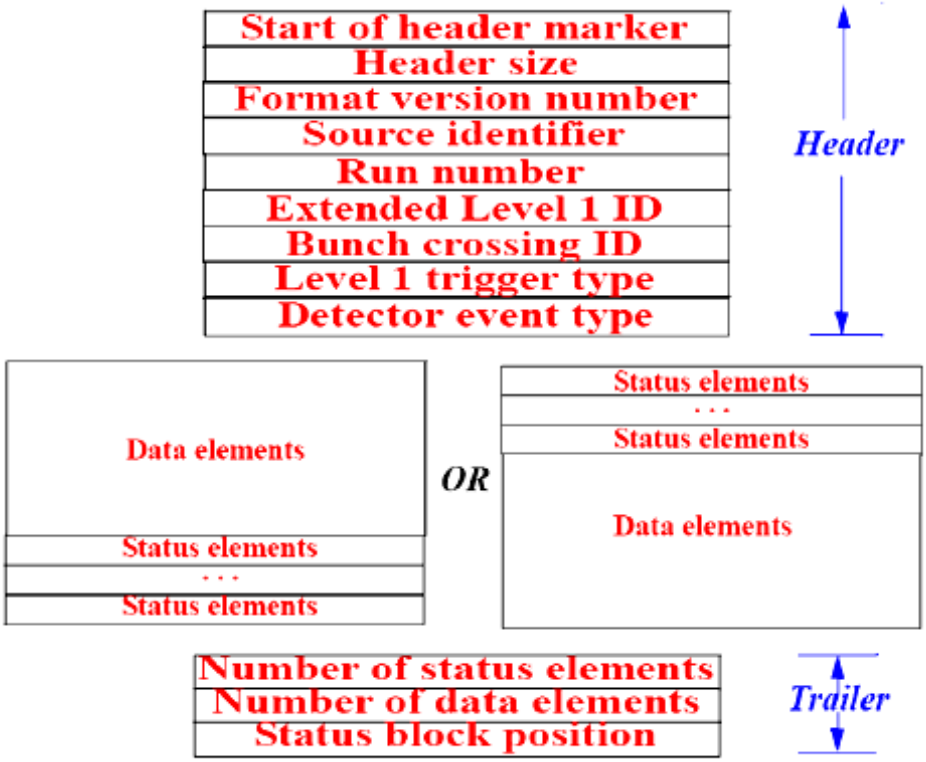


Figure 38 – The output data format of the ROD board.

The output data format of the ROD board [29] is shown in Figure 38: the frame starts with a ROD Header (pertaining to a specific EVID value), includes as a payload the frames coming from the RX/SL boards and ends with a Footer containing status and error flags. Output data are 32 bit words and both ”Header” and “Trailer” (or “Footer”) are made of more words.

Some ROD features have been specifically designed to manage the operations of flow control and error handling. These features allow a user to obtain information about events and errors occurred and to retrieve information about the internal working status of the board.

As already mentioned, the ROD also manages the timing signals of the trigger and data acquisition system. For this purpose, the ROD hosts a TTC receiver module (the TTCrq board) from which it receives control signals to be forwarded to the RX/SL boards. The main timing signals are the LVL1Accept, generated by the first level trigger processor to validate data related to a specific bunch-crossing, and the reset signals BCR and ECR, needed to clear respectively the Bunch Counter and the Event Counter registers. The ROD also provides the RX/SL boards with a clock signal synchronous to LHC.

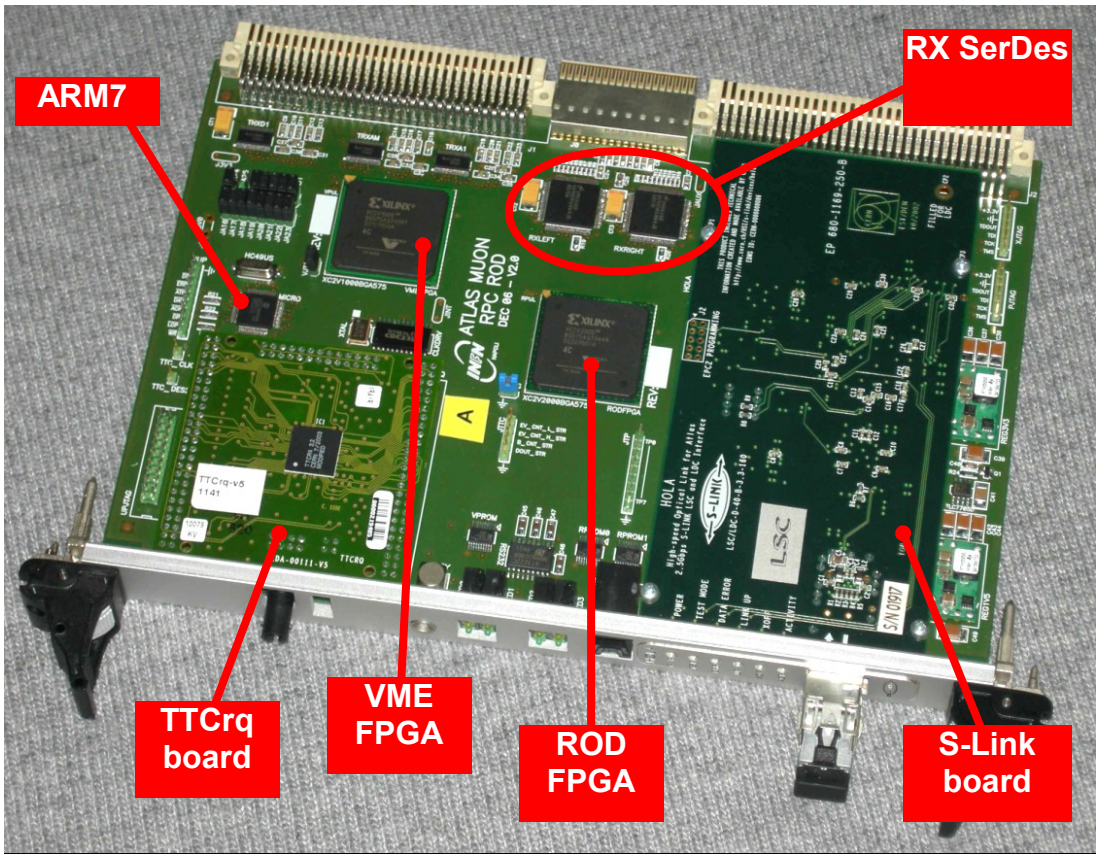


Figure 39 – The Read Out Driver VME board.

The ROD (Figure 39) has a VME interface that allows the user to access the Event Builder function via a VME CPU, on which a specifically designed software runs.

Other monitoring features of the ROD include the control of power supply, of the RODbus backplane's temperature and the of the TTC timing setting. This is achieved by an ARM7 microcontroller, via an I^2C communication protocol [30].In Figure 39 a photo of the ROD board is shown. The board has the form factor of the VME 64X 6U and is equipped with two VIRTEX II [31] XILINX FPGAs, labelled in Figure 39 as *VME FPGA* and *ROD FPGA*. The board also hosts a microcontroller (labelled as *ARM7*) and the deserializers (*RX SerDes*) that receive data via the RODbus backplane from the RX/SL boards. In Figure 39 are also shown the *TTCrq* modular board, receiver of the TTC optical system, and the transmitter modular board *S-Link*, responsible of the data transmission to the next acquisition levels. Each of these elements will be discussed on later.

Figure 40 shows a block diagram of the ROD board. In this scheme it is possible to see the VMEbus connected to the VME FPGA and the RODbus connected with the two SerDes.

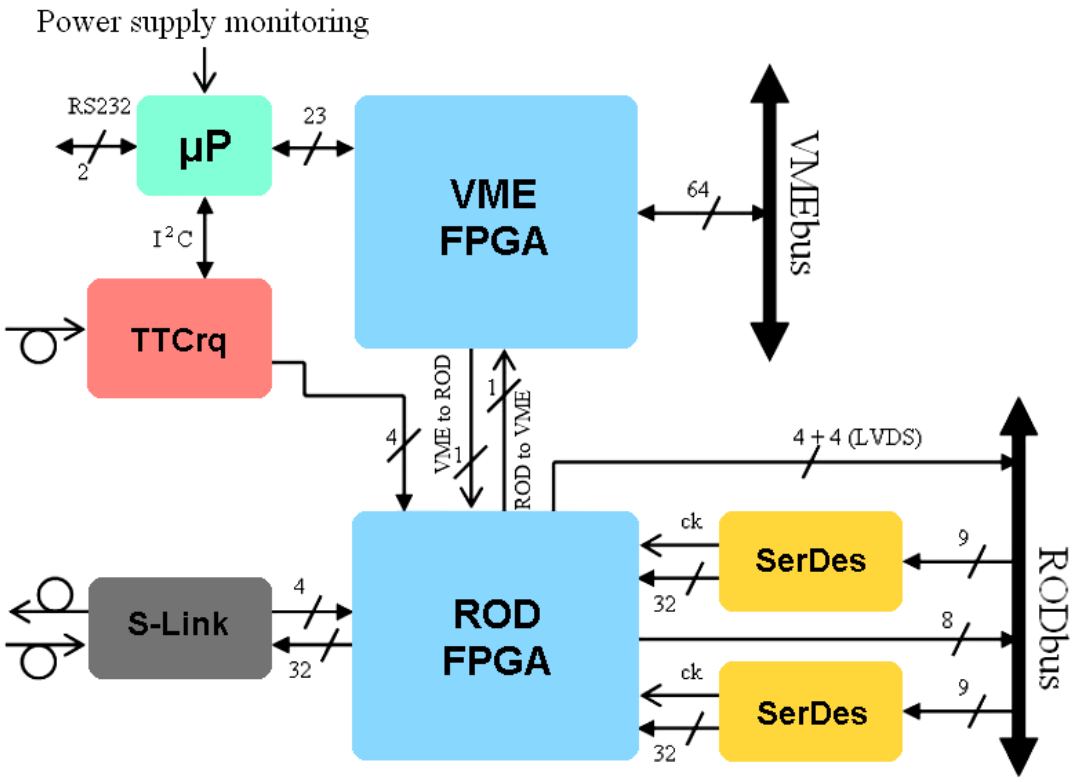


Figure 40 – ROD Block diagram.

3.1.4 The VME FPGA

The VME FPGA is a Virtex-II 1000 bg575 FPGA, produced by XILINX. Its main task is to interface the whole ROD board with the VMEbus and so, via the VME CPU, with the ATLAS DAQ environment. The connection with the ROD FPGA, that hosts the logic to perform the *Event building*, is made by a serial custom protocol that will be described in the next paragraph. The VME FPGA is also connected to microcontroller (labelled with "µP" in Figure 40), that allows the communication via I^2C protocol, with the temperature sensors, power supply sensors and TTCrq module.

The VME FPGA is fed with the 40 MHz board clock which is internally multiplied to 80 MHz with a DCM [32]. The VME FPGA hosts eight 32-bit registers, used to set and program the different functionalities of the FPGA. The meaning of each register is described in Appendix A.

3.1.5 The serial custom protocol between the FPGAs

The ROD FPGA communicates with the VME FPGA via a serial synchronous custom protocol, carried out by two unidirectional connections. As shown in Figure 40, data from the VME FPGA toward the ROD FPGA are transmitted on the vme_to_rod link, while data from the ROD FPGA toward the VME FPGA are transmitted on the rod_to_vme link.

In the following, the different operations that can be performed via the serial protocol between the FPGAs are described. The FPGA that behaves as the Master of the communication is always the VME FPGA, managing both the write (for data and for address) and read operation. As a consequence, the ROD FPGA can transmit data only if the VME FPGA had previously requested them. The serial protocol allows to set the target address to write data into the ROD FPGA. It also allows to read data from every register of the ROD FPGA. In order to access the ROD FPGA‘s internal registers, two consecutive steps are required:

1. First, a write access is necessary, in order to set the address of the target register for a write or read transaction;
2. Then a further access is required in order to transmit data to be written in the previously addressed destination register or to receive data to be read from the previously addressed destination register.

The *ROD FPGA* internal registers are identified by a 8-bit address, but only 16 registers are used; each register is 32 bit wide.

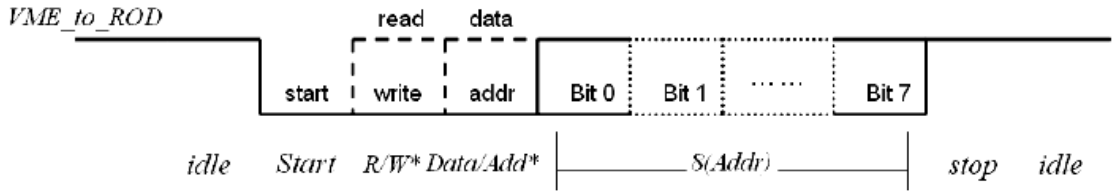


Figure 41 – Addressing the destination register on the ROD FPGA.

Figure 41 describes the serial protocol, depending on the operation to be performed. The *vme_to_rod* line is normally at the logic level “1”: when an operation is started, the *vme_to_rod* net goes down for a clock cycle, indicating

the start of a transaction. The next bit will be “0” or “1”, to flag a write operation to the *ROD FPGA* or a read operation from the *ROD FPGA*. The third bit of the sequence is “0” for address setting, “1” for a data transaction. The next bit-field will be 8 bit long, if the operation is related to address setting, or 32 bit, if the operation is related to data. Then, the transaction ends and the net goes to the logic level “1”.

3.1.6 The ROD FPGA

The ROD FPGA is a XILINX Virtex-II 2000 bg575 FPGA. As previously said, its main task is to perform the event building algorithm on data transmitted by the RX/SL boards. After being elaborated by the Event Builder Engine, data are transferred to the next levels of data acquisition system via the S-Link system.

The ROD FPGA is also responsible for correctly framing data and for the management of the S-Link system. The ROD FPGA receives four timing signals (L1Accept, Bunch Crossing Reset, Event Counter Reset and the 40 MHz clock) from the TTC and distributes them, by LVDS connections over the RODbus backplane, to the two RS/SL boards. The ROD FPGA also hosts the receiver logic of the serial custom protocol that allows the VME FPGA (and so the VME bus and the VME CPU) to have access to ROD FPGA registers and data.

The ROD FPGA hosts 16 32-bit registers, used to configure the different functionalities of the FPGA. Such functionalities are, as example, the setting of the different options in the event building procedure, the activation of the various error handling procedures or the setup for debug procedure. The meaning of each register is described in the Appendix A.

The ROD FPGA, like the VME FPGA, uses a clock frequency of 80 MHz. This feature allows the whole Event Builder Engine to work two times faster than the clock frequency of the ATLAS apparatus and to achieve a higher bandwidth.

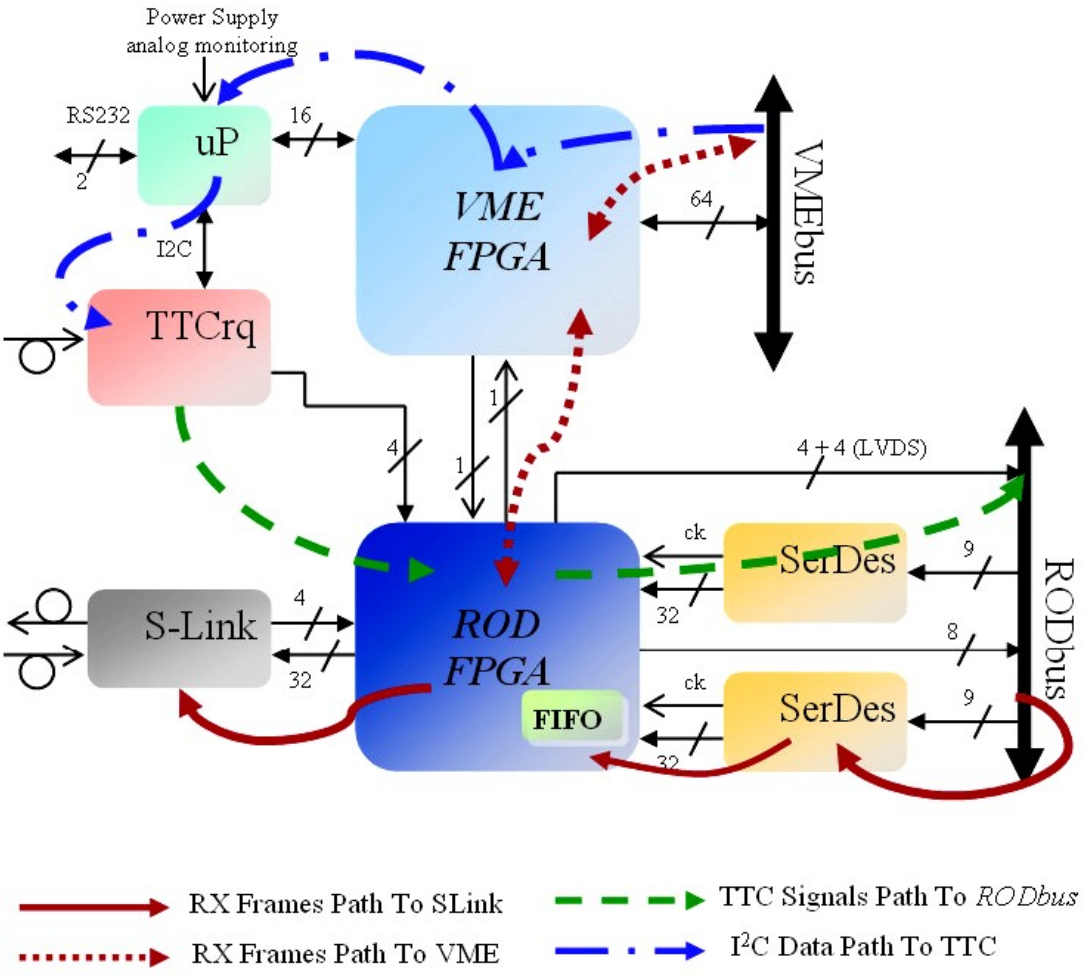


Figure 42 – The main paths on the ROD board.

3.1.7 The Data and Control paths

Figure 42 shows the main Data and Control paths on the ROD board. The most important path links the ROD board to the RX/SL boards via RODbus.

This path is shown on the lower side of the Figure 42, with the continuous line. Data from the RODbus enter the SerDes and then the ROD FPGA. In the ROD FPGA data are buffered in a 32-bit x 4K words FIFO (First In First Out) memory. The task of this FIFO, as will be explained in the next paragraphs, is to decouple the different clocks arriving on the ROD FPGA and to synchronize all data with the board clock. When data are read from the FIFO, they are processed by the Event Builder Engine. Then data are sent over S-Link to the next acquisition levels or to the VMEbus for monitoring.

The data path to the VMEbus is shown with the dotted line in Figure 42. This path has a great importance for debugging the Event Builder Engine because it

allows to monitor the correctness of data at every step of the event building process. In fact, data can be read when they are written by the SerDes, or when they are read from the external FIFO, or also when the ROD MUON FRAME is complete and ready for the next acquisition levels. When ATLAS will be running, the DAQ system will be able to sample events under software control, reading frames with specific L1IDs, that can be set by the user.

The path of the timing signals is shown with the hatched line in Figure 42. The four timing signals transmitted by the TTC are received by the ROD FPGA and distributed on the RODbus in LVDS standard.

The path of the signals for the control of the TTC is shown with the hatch-and-point line in Figure 42. TTC is controlled by the ARM7 microcontroller (μP) via an I^2C interface, whereas the microcontroller receives information from the VME FPGA.

3.1.8 The serializers (SerDes)

The most relevant problem in the communication between the RX/SL boards and the ROD is the great amount of data (48 bit at 40 MHz) involved in the transaction. In fact, the number of bits involved in every transmission cycle is much higher than the number of connections available on the VME backplane.

For this reason a custom backplane was designed to guarantee the required performances in the data acquisition system.

In the adopted solution, a commercial chipset is used to serialize 48 bit data on 8 serial channels, with a transmission frequency of 280 MHz.

Such devices, produced by *National semiconductors*, are the DS90CR483 and the DS90CR484 [33], respectively the transmitter (or “serializer”) and the receiver (or “deserializer”) of the chip-set. The data transmission architecture is shown in Figure 43.

The transmitter accepts as input 48-bit data words, that are multiplexed on 8 pairs of lines. Another pair of lines is used to transmit the synchronization signal (clock). The nine signals are transmitted using the LVDS standard [34] over differential couples.

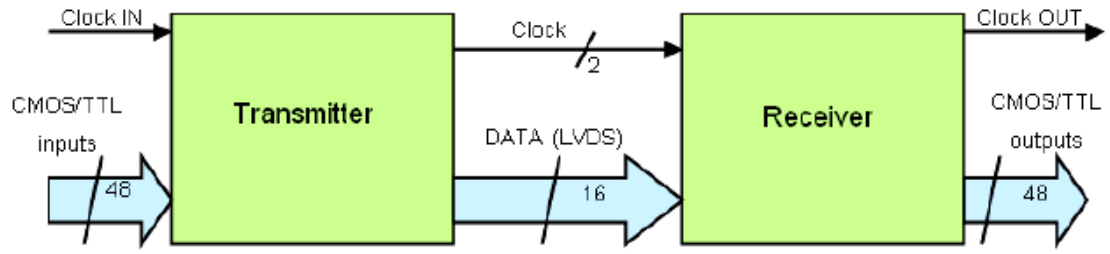


Figure 43 – The data path from serializer to deserializer.

The receiver accepts as input the nine serial LVDS channels (data+clock) and recovers the original 48 bit words, synchronizing them with the transmitter's clock signal. The basic idea to transmit data over multiple serial connections is the following. The DS90CR483 and the DS90CR484 devices acquire parallel data, transfer them with the described "pseudo-serial" protocol, over few lines, and return parallel data. The total interconnections' number is reduced and the interference effects are limited by the differential transmission. In each period of the external clock, six bit of data word and an additional bit (DC Balance) are sent on every connection. The task of the DC Balance bit is to limit, through a particular inversion algorithm, the fluctuations of the mean value of the voltage on the transmission line [33]. The transmission clock frequency can be seven times greater than the external clock: in particular for this application, the transmission bandwidth over each line is 280 Mbit/s.

3.1.9 The TTCrq

As said in the previous chapter, the LHC TTC (Timing, Trigger and Control) system delivers the level 1 trigger accept signal and all the timing signals necessary to synchronise the detectors, i.e. the 40 MHz LHC clock, the Event Counter Reset (ECR) and the Bunch Counter Reset (BCR)signals.

A special timing receiver ASIC (TTCrx) has been developed by CERN for the TTC systems. The ROD board hosts the TTCrq modular board with the TTCrx ASIC, that produces the TTC signals, as will be explained next. Four of these signals (the Level 1 Accept, the Bunch Crossing Reset, the Event Counter Reset and the 40 Mhz clock) are forwarded by the ROD to the RX/SL boards over the RODbus LVDS timing connections.

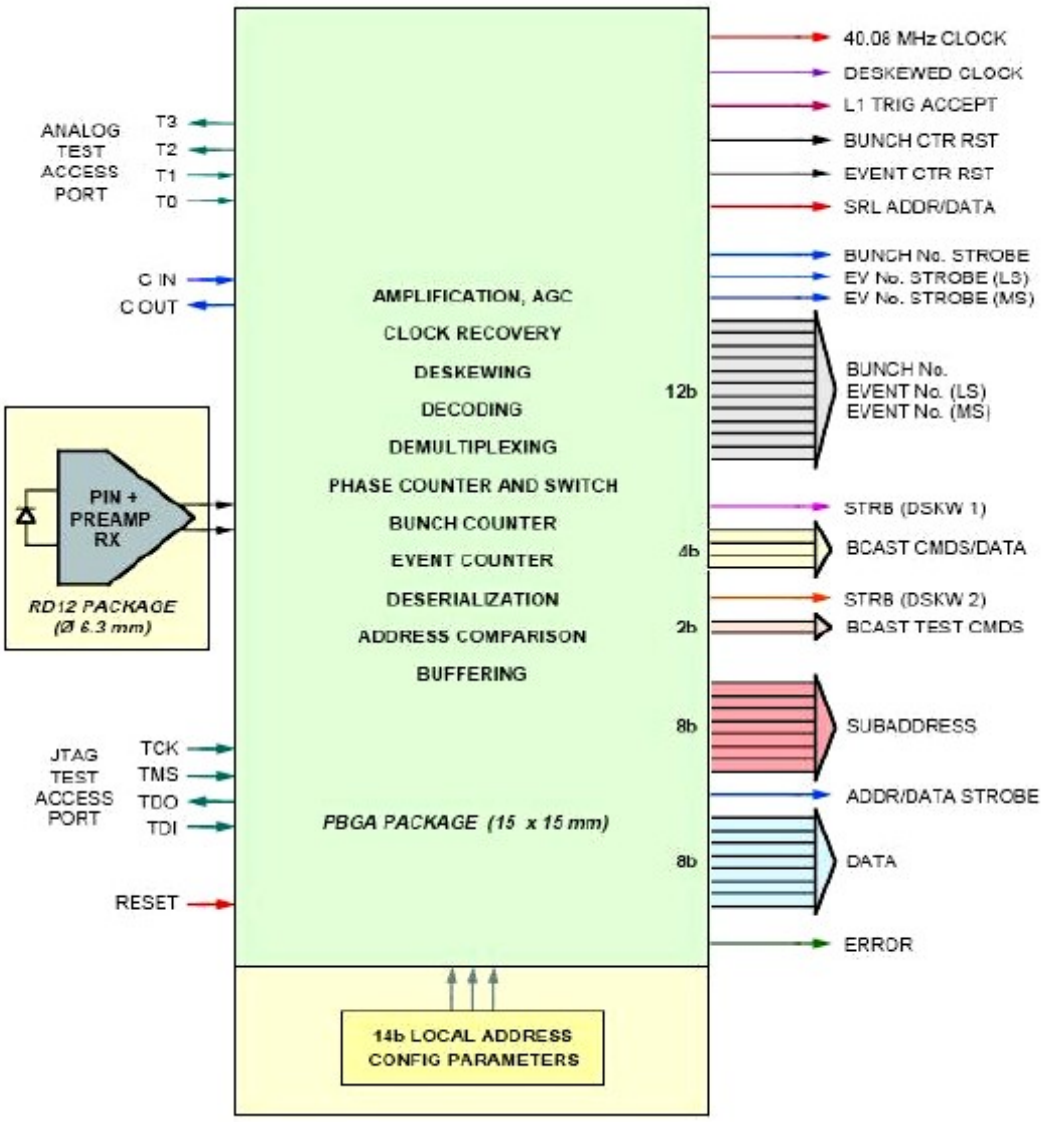


Figure 44 – The TTCrx ASIC pinout.

As indicated in the pinout scheme in Figure 44, the TTCrx [35] accepts a single input from the TTC photodetector/preamplifier and generates a full range of decoded and deskewed signals for the electronics controllers.

The ASIC comprises an analogue part (including the postamplifier and clock recovery/fine deskew PLLs) and a digital part (including decoding, demultiplexing, bunch counter, event counter and command processing sections).

In order to obtain a fine deskew resolution, the phase of the clock may be adjusted in steps of 104 ps, over the full 25 ns Bunch-Crossing interval. This allows to take into account possible propagation variations due to differences in time-of-flight and optical fibre path length.

The bunch counter number is also provided for external use by a 12-bit counter, which delivers a unique Bunch Crossing number synchronously with the corresponding first level trigger decision. During the 2 clock cycles following trigger accept, for which the central trigger logic (global trigger) inhibits the generation of new triggers, the corresponding 24-bit event number is delivered on the same 12 output lines as the bunch number. Unlike the bunch counter, the event counter need not be reset periodically but rolls over after every 16 millions events (about every 3 minutes at the level-1 trigger rate). All the TTCrx event counters are initialised by a broadcast command and may be reset by such command during any gap in the LHC bunch structure.

The transmission sequence for the level 1 accept signal, the 12-bit bunch number and the 24-bit event number is also called the “Trigger sequence”. The timing of the “Trigger sequence” is shown in Figure 45.

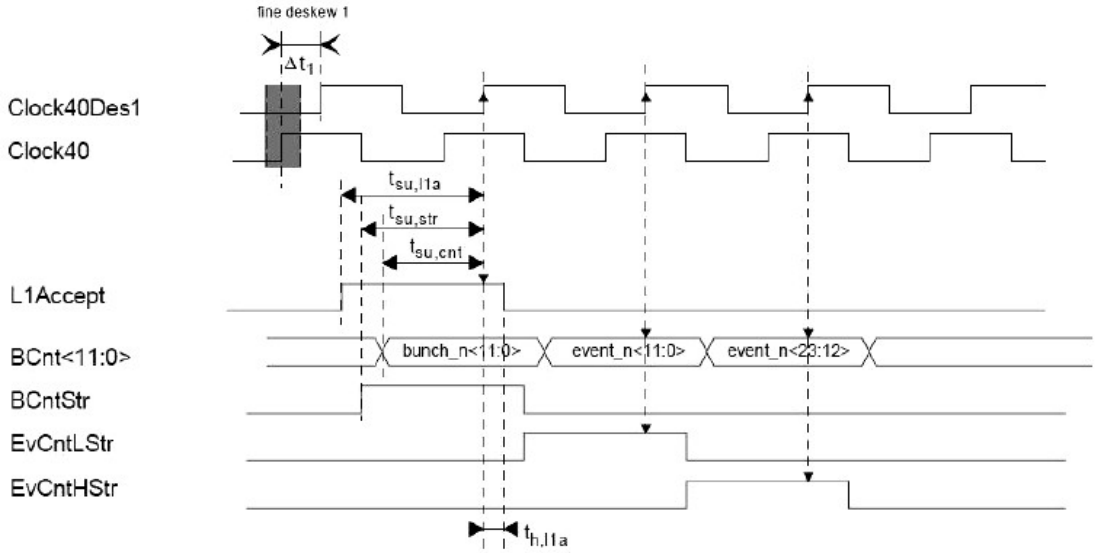


Figure 45 – The timing of the “Trigger sequence”.

The 12-bit bunch number generated by the TTCrx is required by the synchronization algorithms and permits the study of correlations between event data and the LHC orbit. The 24-bit event number suffices to detect possible problems of event ordering or loss in the data readout and event building. Additional information, which makes the event identification unique, is added to data at later stages of the DAQ chain.

The TTC also allows to transmit synchronised broadcast and individually-addressed commands and data. The broadcast commands are decoded by all TTCrx’s and can be up to 256, encoded in 8 bit words. The

individually-addressed commands are sent to specific chips with a defined identification number (a 14bit ID number).

Since the broadcast protocol is common to the whole apparatus, the most interesting protocol for the ROD board logic is the individually-addressed commands protocol, reported in Figure 46. The net data contained in the TTC packet amounts to 16 bits. It is divided into an 8-bit DATA byte, and an 8-bit SUBADDRESS byte. At the output of the TTCrx, the net 16-bit data content appears on the Dout<7:0> and SubAddr<7:0> pins and DoutStr validates the signal.

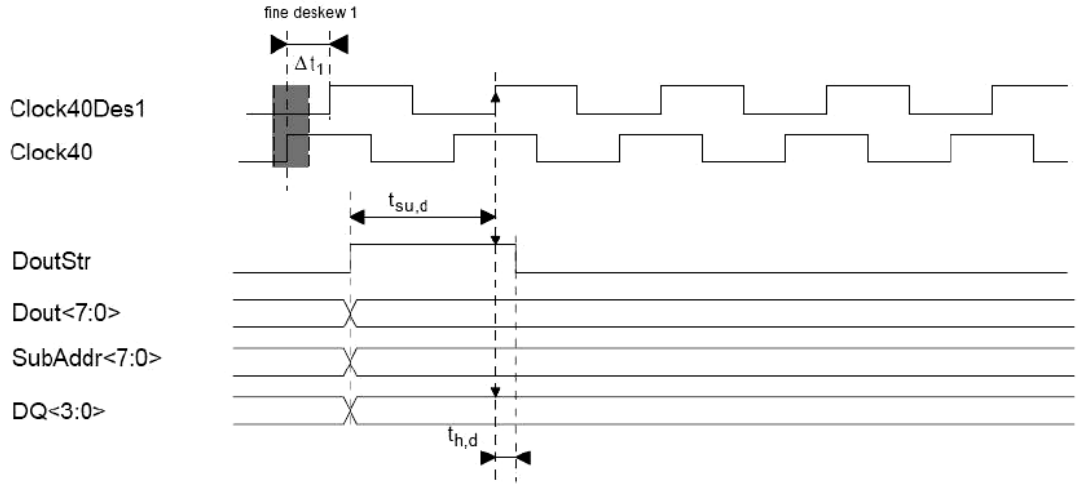


Figure 46 – The timing of the individually-addressed commands and data.

The transmission of 32 bits of a generic individually-addressed word will take 4 individually-addressed command.

When the TTCrx receives an individually-addressed command and decodes the 14-bit ID, the six bit SUBADDRESS <7:2> field will have to match six bit written in one of the ROD FPGA registers, whereas the two bit SUBADDRESS <1:0> will span over the possible four values:

- “00” will validate the lower 8 bits of the word, i.e. word_data <7:0>;
- “01” will validate the word_data <15:8>;
- “10” will validate the word_data <23:16>;
- “11” will validate the word_data <31:24>.

One of the information transmitted by an individually-addressed command is the “Trigger Type word”. The level-1 (LVL1) trigger system generates an 8-bit “Trigger Type word” for each accepted bunch crossing. The most-significant bit of the LVL1 trigger word (“trigger-mode” flag) is used to distinguish between physics triggers and calibration/test triggers. The meaning of the remaining 7 bits is different for these two cases and is described in [36]. As an example, in physics trigger mode some of the remaining bits can indicate that calorimeter and muon triggers might be treated differently, or that low p_T muon triggers might require a special treatment.

The TTC data used by the ROD FPGA Event Builder Engine are the 24-bit Bunch Crossing ID word, the 12-bit Level1Accept ID word and the 8-bit Trigger Type word. These words are stored in two internal FIFOs, whose features are described in the next paragraphs. One 36-bit FIFO will host the 24-bit Bunch Crossing ID and the 12-bit Level1Accept ID and this FIFO will be labelled as TTC L1 FIFO from now on. An 8-bit FIFO will host the Trigger Type 8 bit data and will be labelled as Trigger Type FIFO. Both the TTC L1 FIFO and the Trigger Type FIFO can host 511 words. Each TTC word (Bunch Crossing ID, Level1Accept ID and Trigger Type) is received by the ROD and stored in the corresponding FIFO; then it is used by the Event Builder Engine in the ROD FPGA to build the ROD MUON FRAME, with the format presented in the next chapter.

3.1.10 S-Link

S-LINK has been developed at CERN and can be thought of as a virtual ribbon cable, moving data or control words from one point to another. It can be used to connect any layer of front-end electronics to the next layer of read-out. S-LINKs can be constructed using different physical media and therefore even if S-LINKs may have different timing characteristics and data transfer rates, all will have to comply to [37].

In Figure 47 the conceptual scheme of S-Link is reported.

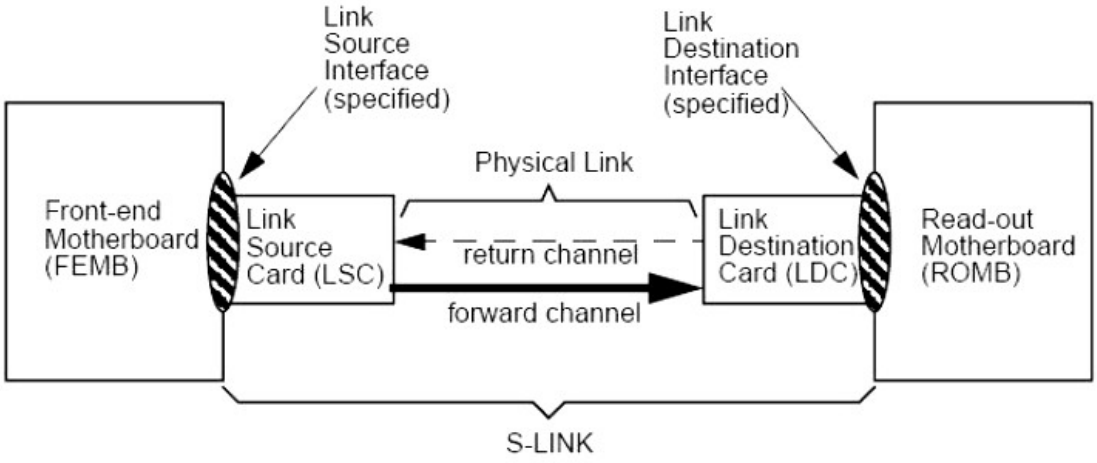


Figure 47 – Conceptual scheme of an S-Link.

In addition to simple data movement, S-LINK includes the following features:

- Error detection.
- Return channel for flow control and for return line signals (duplex version only).
- Self-test function.

The physical link can be conceived as the interface between the Source, on the Front End Electronics, and the Destination. The source is a Front-end Motherboard (FEMB) with the addition of a Link Source Card (LSC); the destination is a Link Destination Card (LDC) on a Read-out Motherboard (ROMB). In many applications a uni-directional data channel is required.

In some cases the user might require to send flow control and other information back to the FEMB. The S-LINK, therefore, can be constructed in either a *simplex* or a *duplex* version.

In the duplex version, a return channel exists which allows the LDC to pass information back to the LSC. The main function of this return channel is to transmit flow control commands from the ROMB to the FEMB. Thus a duplex SLINK transmits only when the ROMB is available to read the data. When the ROMB is unavailable, data transfers from the FEMB to the S-LINK are inhibited.

In particular, the ROMB can assert a signal, XOFF, at the LDC which forces it to transmit an XOFF code to the LSC. This stops data transmission from the LSC. When the signal is de-asserted, an XON code is transmitted to the LSC which allows transmission to resume. This is exactly what happens on the ROD.

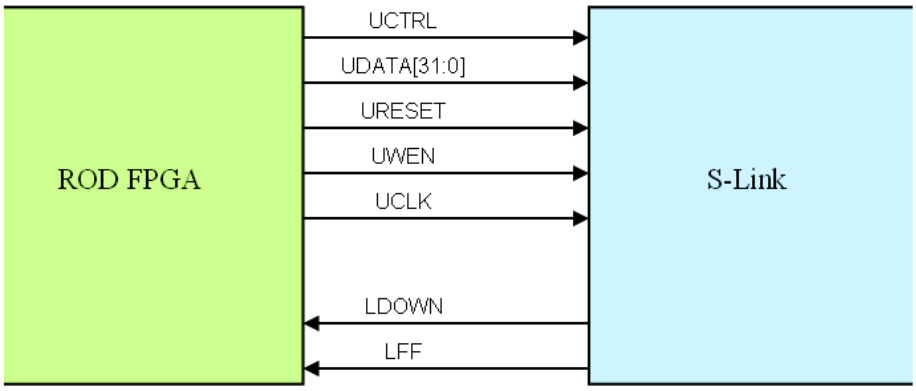


Figure 48 – The connections between ROD FPGA and S-Link.

Figure 48 shows the connections between the S-Link and the ROD board, that can be conceived as a FEMB; the mezzanine that is hosted by the ROD board represents the LSC. The two pins that must be controlled by the ROD board are:

1. *Link Down* (LDOWN): When low indicates that the S-LINK is not operational. It is asynchronous and can go low due to:
   - S-LINK failure; in this case LDOWN is latched low until cleared by a reset cycle.
   - S-LINK is undergoing a reset cycle; in this case LDOWN goes high when reset cycle is complete.
   - S-LINK is in test mode.
2. *Link Full Flag* (LFF): Data *shall* only be written to the S-LINK when this line is high. After it goes low, up to two more words *may* be written.

The signals that are generated by the ROD and are inputs to the Slink are:

1. *User Clock* (UCLK): Data is transferred to the S-LINK on the low-to-high transitions of UCLK when UWEN is low. This is a free running clock.
2. *User Write Enable* (UWEN): When low enables data to be transferred to the S-LINK on the low-to-high transition of UCLK. Synchronous with UCLK.

3. *User Reset* (URESET): When low initiates a reset cycle and is asynchronous.
4. *User Data* (UDATA): Data on these lines are transferred to the LSC on a low-to-high transition of UCLK when UWEN is low. UserData[3...0] are ignored if UCTRL is low. User Data are synchronous with UCLK.
5. *User Control* (UCTRL): When low indicates that the data to be transmitted is a control word. Causes UserData[3..0] to be ignored. Synchronous with UCLK.

Figure 49 shows the timing of data transmission from the ROD board (i.e. the FEMB) to the S-Link mezzanine. Data transfer to the LSC is based on writing to a FIFO memory. The ROD provides a free-running clock (UCLK) which shall be present at all times. When data are to be transferred to the LSC, the write-enable line (UWEN) is set low and the data words are transferred on each subsequent low-to-high transition of UCLK. Data can only be transferred to the LSC when Link Full Flag (LFF) line is high. If this line goes low, the S-LINK will be able to receive up to two more data words (this is to allow the ROD logic time to react to LFF). After transferring the extra words, the ROD should not try to transfer datato the S-LINK or data may be lost.

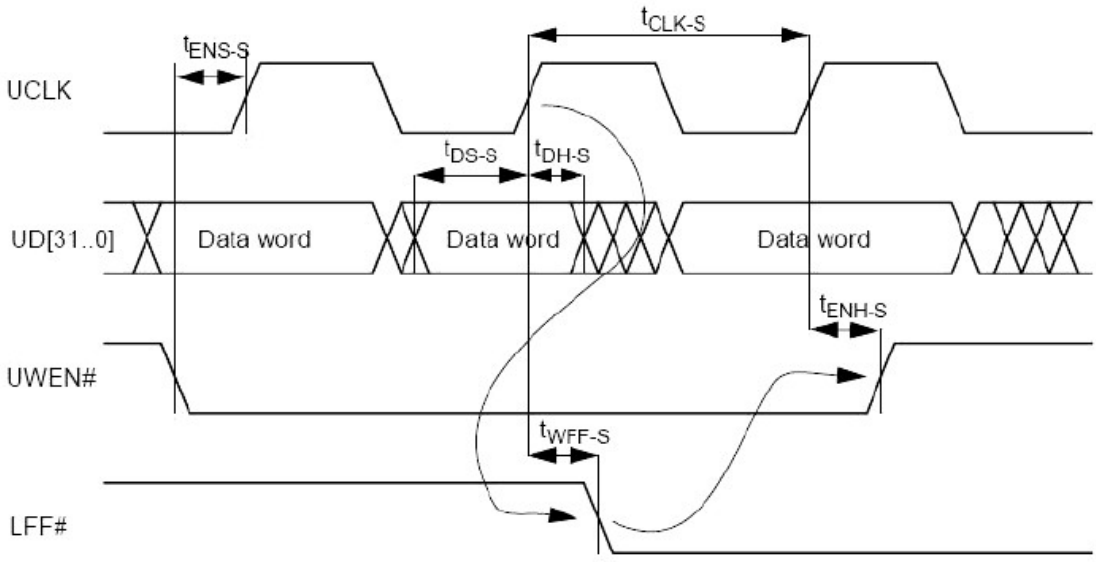


Figure 49 – The timing of data transmission on S-Link.

3.1.11 The FIFO memories

FIFO memories used inside the ROD FPGA are obtained using its internal resources, in particular the hardware RAM blocks. In Figure 50 the pinout scheme of the Virtex II internal FIFO is displayed. Such FIFOs work with synchronous read and write protocols. The input is controlled by a write clock (Wr_CLK) and by the input pin Write Enable (Wr_En). Data (DIN) are written in the FIFO on every rising edge of the Wr_CLK, when Wr_En is asserted. The output of the FIFO is controlled by the read clock Rd_CLK and by the input pin Read Enable (Rd_En). Data are put on the FIFO's output pins (DOUT) on every rising edge of the Rd_CLK, when Rd_En is asserted.

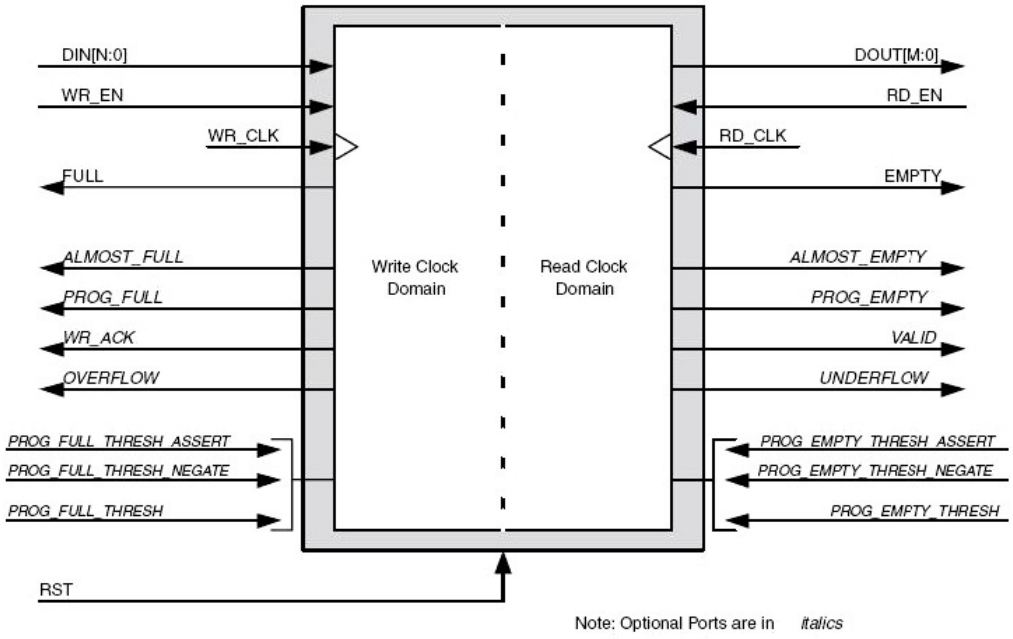


Figure 50 – The pinout scheme of the VIRTEX II internal FIFOs.

Also the FIFO occupancy flags can be seen: full, empty, almost full and almost empty. It's worth to point out the inputs *Programmable Empty Threshold* and *Programmable Full Threshold* and the corresponding outputs *Programmable Empty* and *Programmable Full*. The *Programmable Empty Threshold* and *Programmable Full Threshold* inputs allow programming the threshold values for the assertion and de-assertion of the *Programmable Empty* and *Programmable Full* flags.

Their values can be programmed by software and can be set by access to internal registers of the ROD FPGA, as can be seen in Appendix A. The *ROD*

FPGA internal FIFOs play a crucial role on the ROD board. Their function is to allow storing data and also acting as a buffer between different clock domains. This feature is very important because the board local clock is a 40 MHz local clock, not synchronous to the LHC clock.

3.1.12 Clock domains and clock muxes

The ROD has four concurring clock domains on board. These clock domains are:

- the 40 MHz clock recovered by the TTCrx receiver of the TTC signals;
- the 40 MHz clock generated by the local oscillator on the board;
- the 40 MHz clock recovered by the SerDes receiving data from the left RX/SL;
- the 40 MHz clock recovered by the SerDes receiving data from the right RX/SL;

Now will follow the descriptions of three design strategies used to match all the clock domains.

***The use of internal FIFOs*.** All clock domains are at 40 Mhz, but it's not possible to predict the phase relationships between them. For this reasons, the use of FIFO memories to match the different clock domains is essential.

Two FIFOs have the task to receive data from the SerDes and to decouple the SerDes clocks from the board clock: from now on, the FIFOs will be labelled as *SerDes FIFOs*. The SerDes FIFOs are 32bit x 4K words deep.

Two FIFOs have the task to receive data from the TTC and to decouple the TTC clock domain from the board clock. These FIFOs are the TTC L1 FIFO and the Trigger Type FIFO described previously.

Two FIFOs have the task to buffer the ROD Frames produced by the Event Builder Engine, that are to be sent out of the board. One FIFO hosts the 32-bit data for the S-Link and from now on this FIFO will be labelled as *S-Link FIFO*.

The other FIFO also hosts data produced by the Event Builder Engine, but only with the purpose of sending them on demand to the VME bus; from now on

this FIFO will be labelled as *VME FIFO*. Both the S-Link FIFO and the VME FIFO are 32 bit x 4K words deep.

***The definition of a new 80 MHz clock domain*.** This is obtained by using 2 DLLs hosted upon the ROD FPGA and the VME FPGA.

***The S-Link transmitter fed with a 40 MHz clock*,** obtained by the division of the 80 MHz ROD FPGA clock. This allows to tune the phases of the S-Link clock with the phase of data sent to S-Link: indeed, in this way all signals are synchronous with the same clock.

Figure 51 shows the spread of the different clock domains over the architecture of the ROD board.

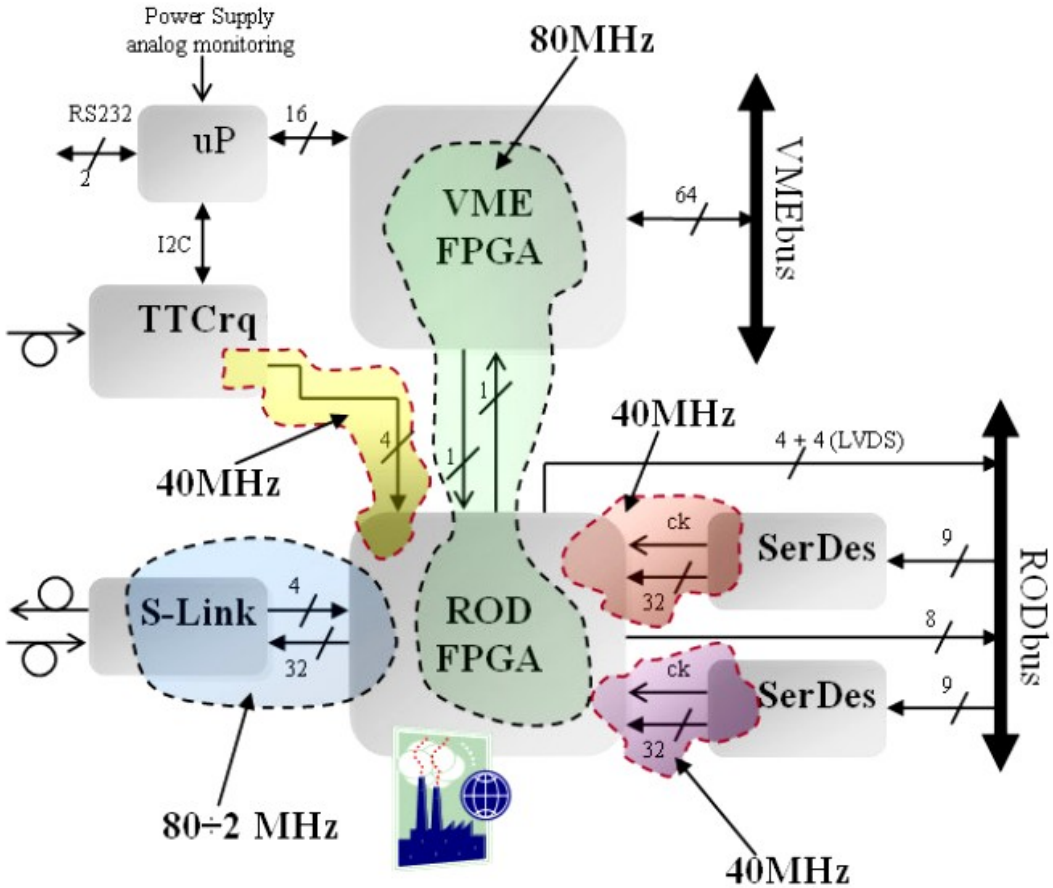


Figure 51 – The spread of the clock domains on the ROD board.

3.1.13 The ARM7 microcontroller

The microcontroller mounted on the ROD board is an NXP LPC2138 [38], based on a 32/16 bit ARM7TDMI-S CPU, that combines the microcontroller with embedded high speed Flash memory.

In Figure 52 the block diagram of the LPC2138 is shown. As can be seen, the ARM7TDMI core is surrounded by many interfaces (or bridges) that allow the communication with the internal connections network.

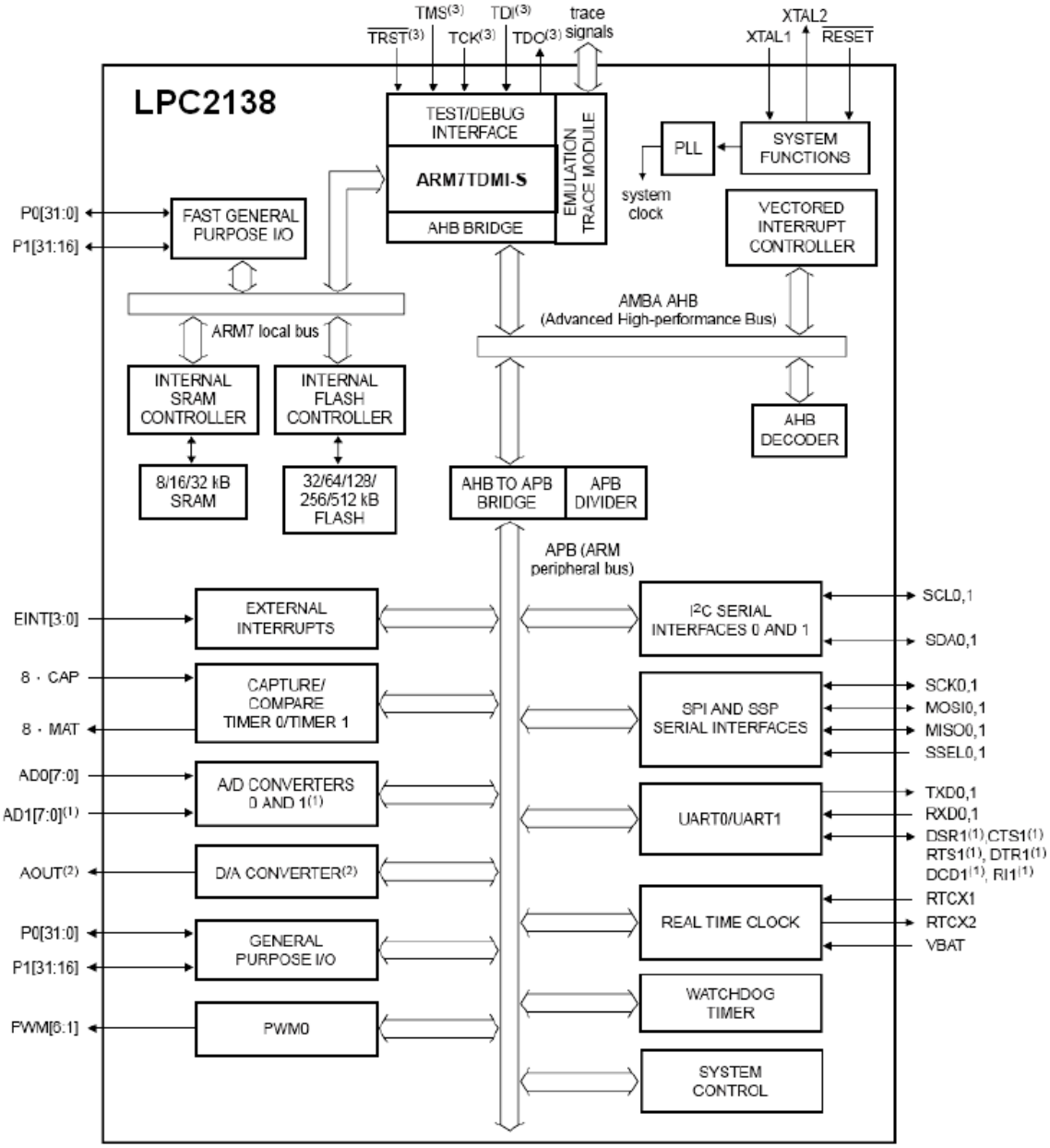


Figure 52 – The block scheme of an LPC2138 microcontroller.

Among the other connections, we can see:

— the *ARM Peripheral Bus*, that connects the ARM7 core to the internal peripherals (I2C interfaces, serial interfaces, A/D and D/A converters, etc.);
— the *ARM7 local bus*, that connects to the internal memory (SRAM or Flash) controllers;
— the *Advanced High-performance Bus*, that connects the core to the Vectored Interrupt controller, whose main task is to assign interrupts priorities from the various peripherals.

The foremost feature of the ARM7 microcontroller is that it is a register based load-and-store architecture: a user program must load data from memory into the CPU registers, process this data and then store the result back into memory. Unlike other processors no memory-to-memory instructions are available. The programmer's model of the ARM7 consists of several registers:

— *Current Program Status Register* (CPSR) containing flags that report and control the operation of the ARM7 CPU;
— 15 user registers: the 13^{th}, the 14^{th} and the 15^{th} are respectively used as *Stack* register, *Link* register and *Program Counter*.

The processor has seven different operating modes in order to handle exceptions (like *Fast interrupt*, *Interrupt*, *Memory error* and so on). Normally, the application code runs in the *user* operating mode, but when an exceptions occurs, the CPU changes mode: the registers from the first to the twelfth remain the same, while the Stack and Link registers are replaced by a new pair of registers, unique to that operating mode. This means that each mode has its own Stack and Link registers. In this way, at the end of every exception, the addresses stored inside the Stack and the Link registers during the user mode can be restored.

In addition, the *Fast Interrupt* mode has duplicate registers also from the seventh to the twelfth and all the exception modes have one more personal

register: the *Saved Program Status Register* (SPSR), in which the CPRS of the user mode is be saved during exceptions.

To increase the performance of instructions' execution, the ARM7 has a three stage pipeline, with the FETCH, DECODE and EXECUTE operations. The hardware of each stage is designed to be independent and up to three instructions can be processed simultaneously. The pipeline is most effective in speeding up sequential code. However a branch instruction will cause the pipeline to be flushed.

The task of the microcontroller on the ROD board is to allow communication between the VME CPU and the TTCrq module via I^2C protocol. In this way, the internal configuration registers of the TTCrq module can be accessed and the TTC module can be programmed. The microcontroller also allows sensing the values of the temperature, voltages and EM activity upon the LVDS bus on the RODbus, in order to monitor its environment.

3.2 The Event building algorithm

The dataflow of the ROD FPGA is shown in Figure 53. Data from each RX/SL board arrive at the FPGA through the corresponding RX SerDes, together with the recovered 40 MHz clock. Data are formatted in frames that are made of a RX Header, a certain number of data words and a Footer. The Header contains the EVID and BCID of the event. The Footer contains a control code.

RX Data are received by the SerDes Interface module and are stored in SerDes FIFOs. Signals from TTC (clock, EVID, BCID and controls) are received by the TTCrq Interface module and are stored in the pertaining FIFO. From all these FIFOs, data are read and elaborated by the Event Builder Engine, synchronously with the 80 MHz internal clock.

As said before, the ROD MUON FRAME produced by the Event Builder Engine is made of 32-bit words. It starts with a Header pertaining to a specific EVID value, includes the frames coming from the RX/SL boards as a payload and ends with a Footer, containing status and error flags. Event Builder output data are stored in the S-Link FIFO and then read out by the S-Link transmitter and sent across the optical link to the Read Out Buffer.

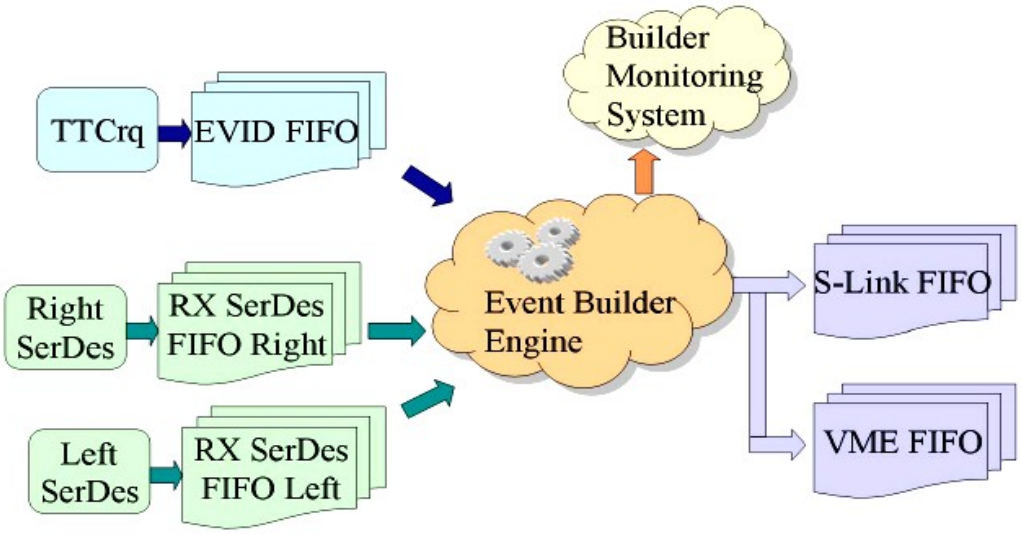


Figure 53 – The dataflow of the ROD FPGA.

3.2.1 The Event Builder Engine

The Event Builder Engine is the core of the ROD board as it performs the framing of the ATLAS readout data. In order to guarantee real-time performance, it as been designed around three Finite State Machines (FSMs), working cooperatively. The Event Builder Engine is made of a master FSM called *Frame Maker*, that builds the ROD Muon Frame, and two identical FSM called *FIFO reader*, that are used to read the RX Frames from the RX SerDes FIFO and to check the correctness of the RX Frame.

The three FSM are one-hot coded and, in order to build the engine, around one hundred states have been used. For this reason, describing its state-by-state behaviour is very hard. However, it is possible to define some “macrostates”, each corresponding to a task that is performed by the cluster of FSM. The bubble diagrams of the different FSMs, in terms of the corresponding Macrostates are shown in Figure 54 and Figure 55.

In Figure 53 you can see a bubble named *Builder Monitoring System*. This is my original contribution to the ROD design. The task of the Monitoring System is to perform real time and statistical analysis of the the Event Builder FSMs dynamics. It can also fill histograms, measure the elapsed time and length for each built event, keep track of status and error words and transfer monitored data via VME. The Monitoring system will be deeply presented in the next chapter.

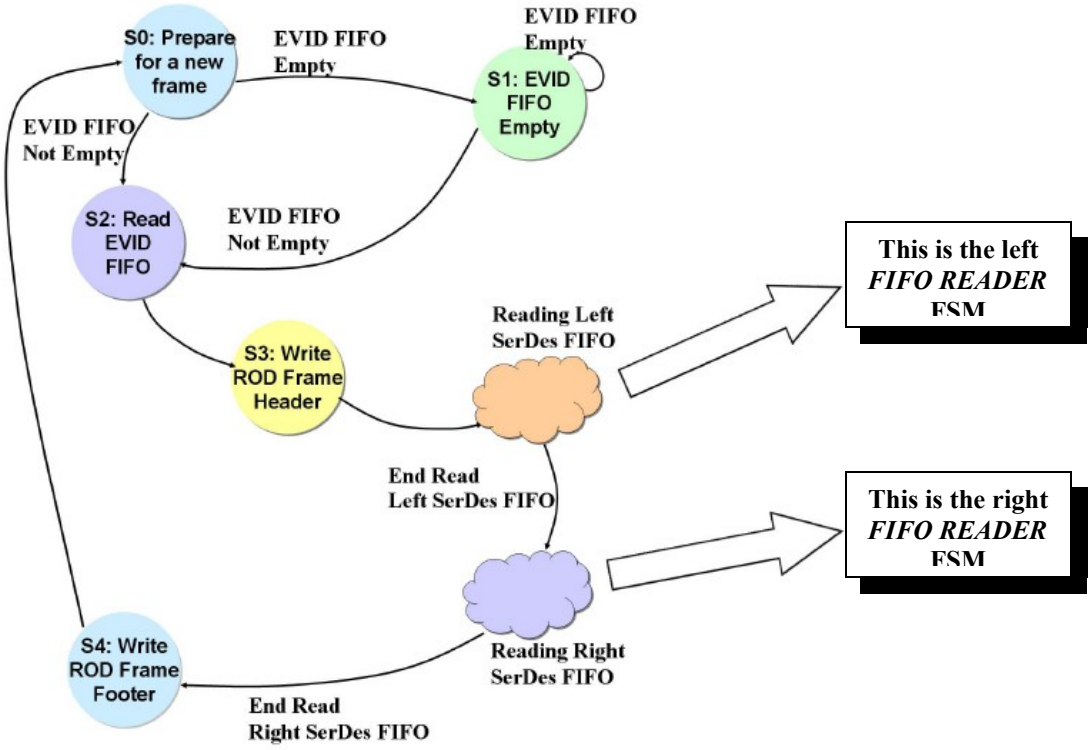


Figure 54 – The *Frame Maker* Finite State Machine.

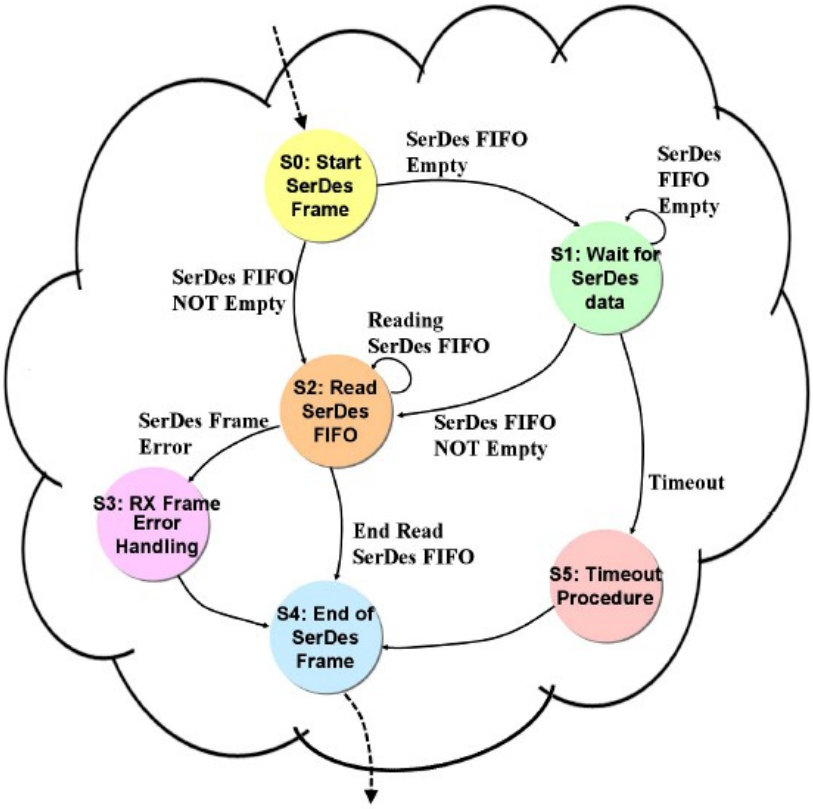


Figure 55 – The *FIFO Reader* Finite State Machine.

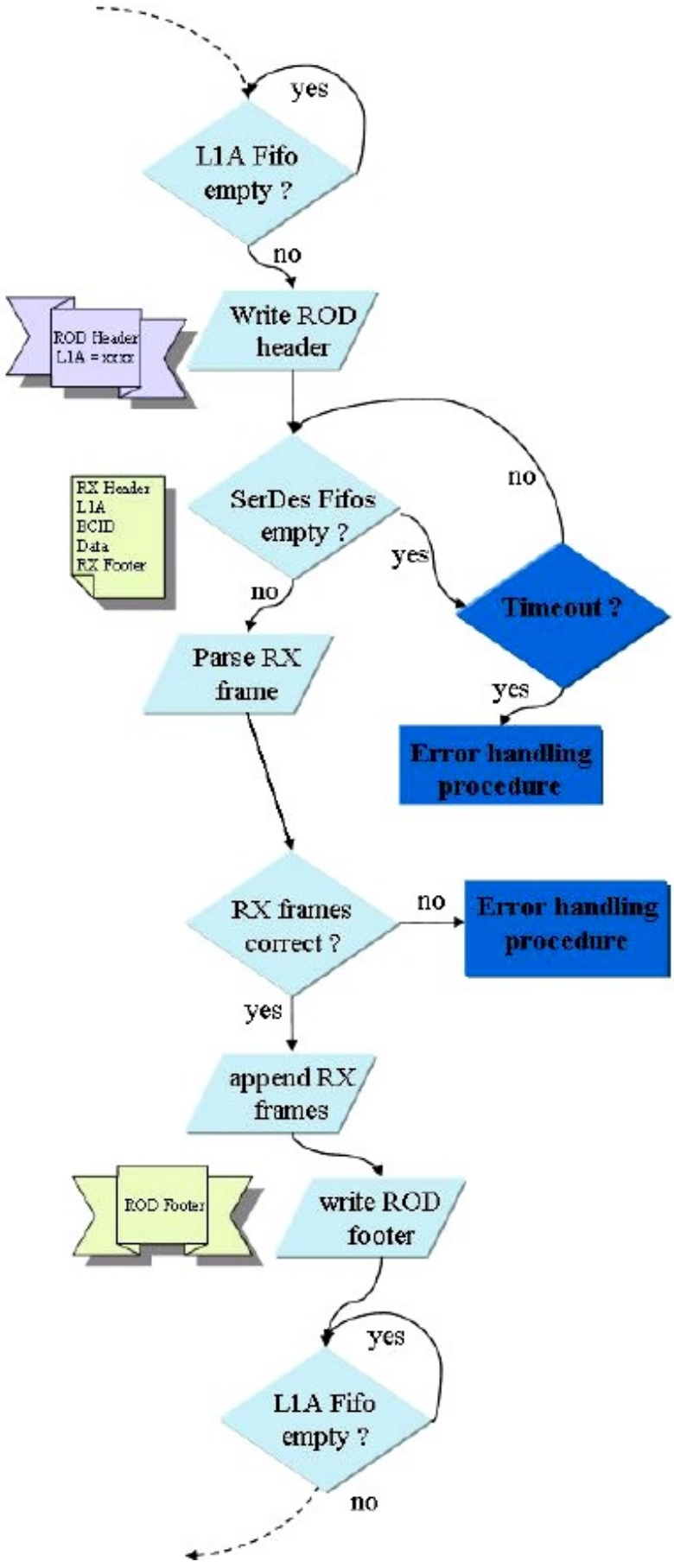


Figure 56 – The Event Builder Engine algorithm.

The basic algorithm of the Event Builder Engine is shown in Figure 56. Every ROD MUON FRAME is relative to a specific EVID value received by the TTC. The Event Builder Engine checks the empty flag of the EVID FIFO and waits for a EVID to be processed.

When a EVID is available, the Event Builder Engine starts writing a valid header into the output S-Link FIFO. The Header contains nine control words, such as the Start of Frame, the board identifier code and information about the current EVID and BCID value.

Then the Event Builder Engine waits for data from the RX/SL boards. As in the previous step, the Event Builder Engine checks the empty flag of each SerDes FIFO. The received RX frames are parsed to find a Header. If the frame is correctly formatted and the embedded EVID and BCID words match the current one, it is appended to the ROD frame. The ROD Event Builder does not inspect the payload of the RX frames: it only counts its total length.

The ROD frame is closed by a Footer, containing status words, error flags and the total word count. Then the Event Builder Engine restarts and waits for a new EVID value.

3.2.2 EVID errors handling

Some of RX frame's control fields are parsed by the Event Builder Engine in order to check their correctness. In particular:

- Header, EVID and Footer are always parsed.
- The BCID can be optionally parsed, depending on the value of a specific bit in a register of the Configuration Register File.
- The total length of the RX Frame is counted.

While reading an RX frame, different errors can occur:

- EVID and BCID errors;
- Formatting error (missing Header of Footer);
- Length error;
- Timeout error.

On the left column in Figure 57 there is the correct typical situation, i.e. a succession of correctly formatted RX frames, each with a consecutive EVID value. EVID and BCID errors occur when there is a mismatch between the codes transmitted by the TTC and the value contained in the RX Frame Header. There are two kinds of EVID errors. The first one is recoverable and it is shown in the right column of Figure 57. While the first RX frame is correctly related to the TTC EVID value, the second one has a corrupted EVID value. In this example, the error is clearly due to a bit flip. The corrupted RX frame will be appended to the ROD MUON FRAME with the EVID value set to 1000 (i.e. the TTC value), with an error flag set in a ROD MUON FRAME Footer. Then, if the next RX frame has the correct EVID value (i.e. 1001), there is no loss of synchronization and the framing proceeds without further errors.

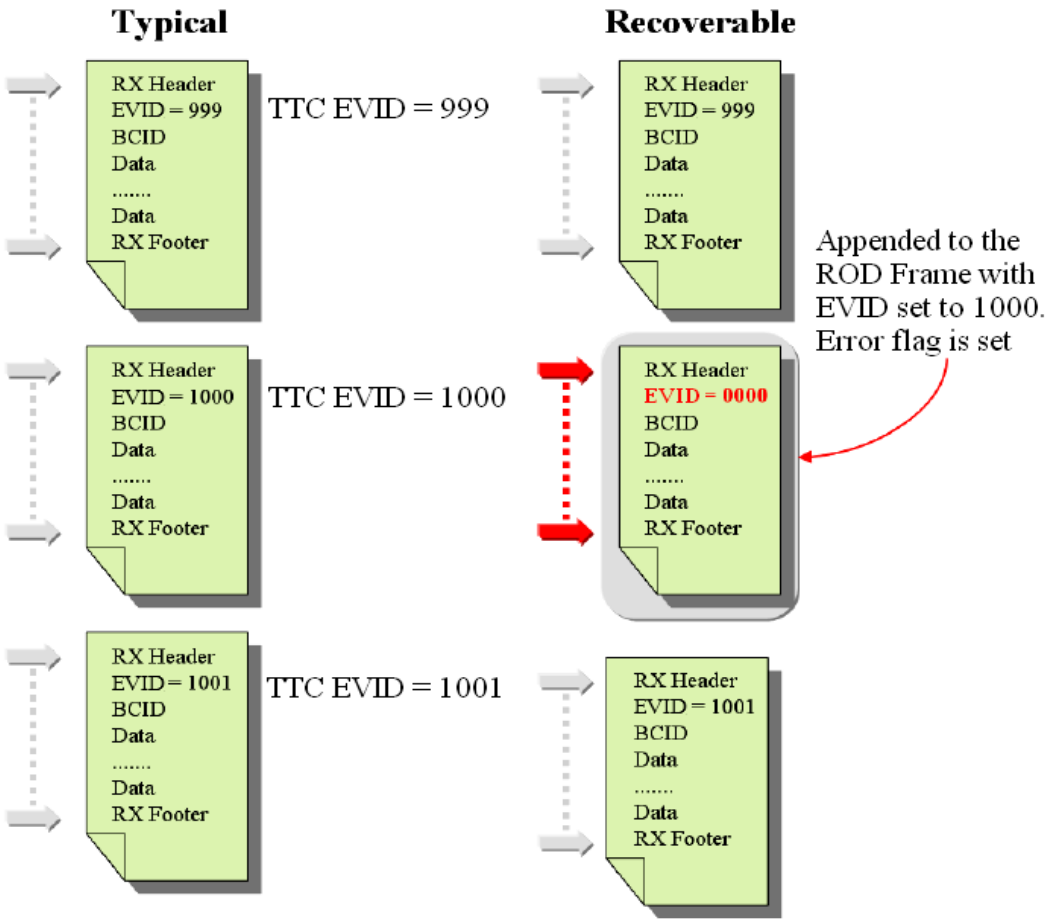


Figure 57 – The occurrence of a recoverable EVID error.

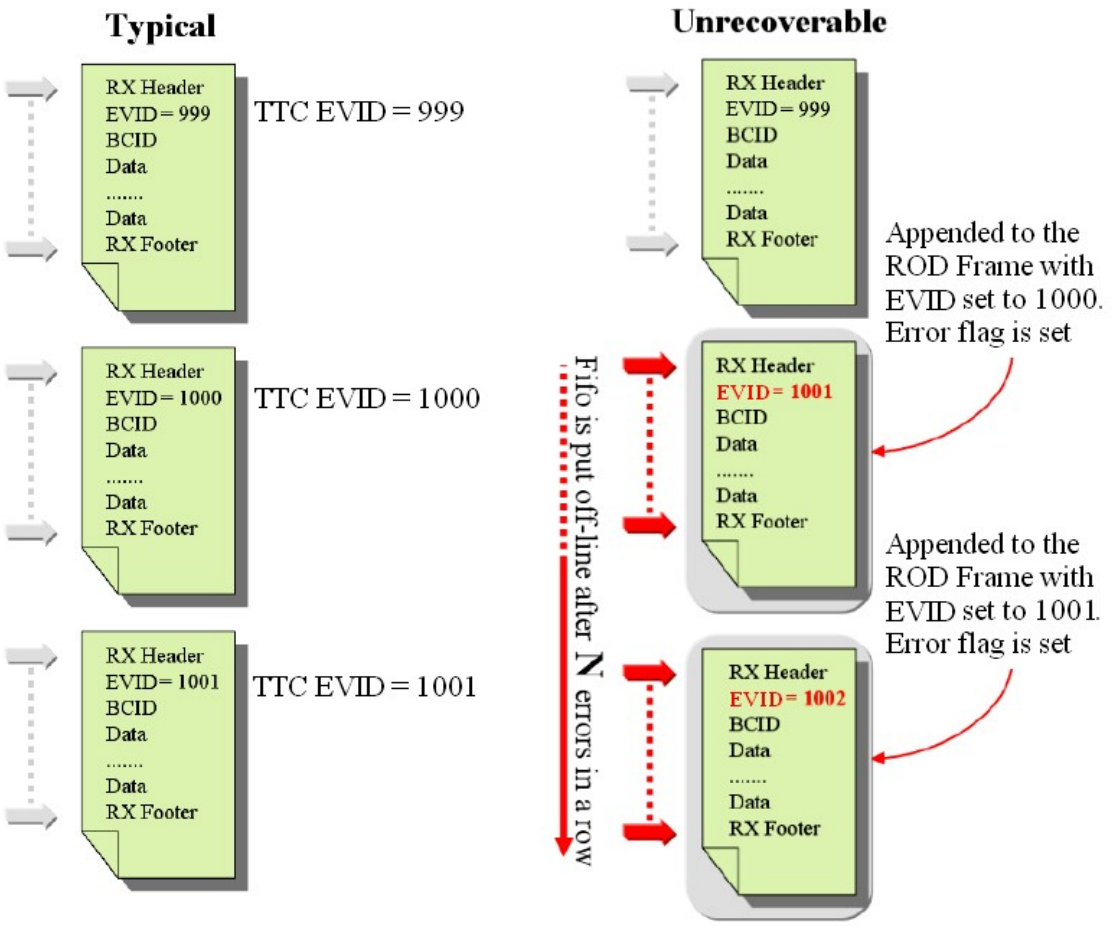


Figure 58 – The occurrence of an unrecoverable EVID error.

The second error is unrecoverable and it is shown in the right column of Figure 58. Also in this case, the first RX frame is correctly related to the TTC EVID value, however the second one has a corrupted EVID value. In this example, the error is due to a missing frame.

As before, the corrupted RX frame will be appended to the ROD MUON FRAME with the EVID value set to 1000, with an error flag set in a ROD MUON FRAME Footer. The next RX Frame will also have a mismatch between its own EVID value and the TTC EVID. Hence, the second corrupted RX frame will be appended to the ROD MUON FRAME with the EVID value set to 1001, with an error flag set in a ROD MUON FRAME Footer. All the EVIDs in the following frames will not match the corresponding TTC EVID values. After N errors (where N is a programmable value in a specific register of the Configuration Register File) the SerDes FIFO with the synchronization problem will be put offline.

3.2.2 Frame syntax errors handling

Three different RX frame syntax error procedures are implemented: Figure 59 shows examples of occurrence of such errors.

In the left column, the so-called *Incomplete Frame* error is shown. An Incomplete Frame error is triggered when the RX Frame Footer is missing. In this case, there will be a realignment on next Header: this means that when the next RX Header is read from the SerDes FIFO, an error flag is set in the Footer of the ROD Muon Frame and the frame is closed; the next RX frame will be correctly available, starting from an Header.

In the central column, the so-called *Corrupted Frame* error is shown. A Corrupted Frame error is identified when the RX Frame Header is missing. There will be a realignment on next Header: this means that all SerDes FIFO's data are read out and dropped until the next RX Header is found. In this case, a de-synchronization between frames can occur and there is no workaround.

A *Length error* is identified when the maximum allowed length of an RX frame is exceeded: this value is called *maxlenght* and it is written in a specific register of the Configuration Register File.

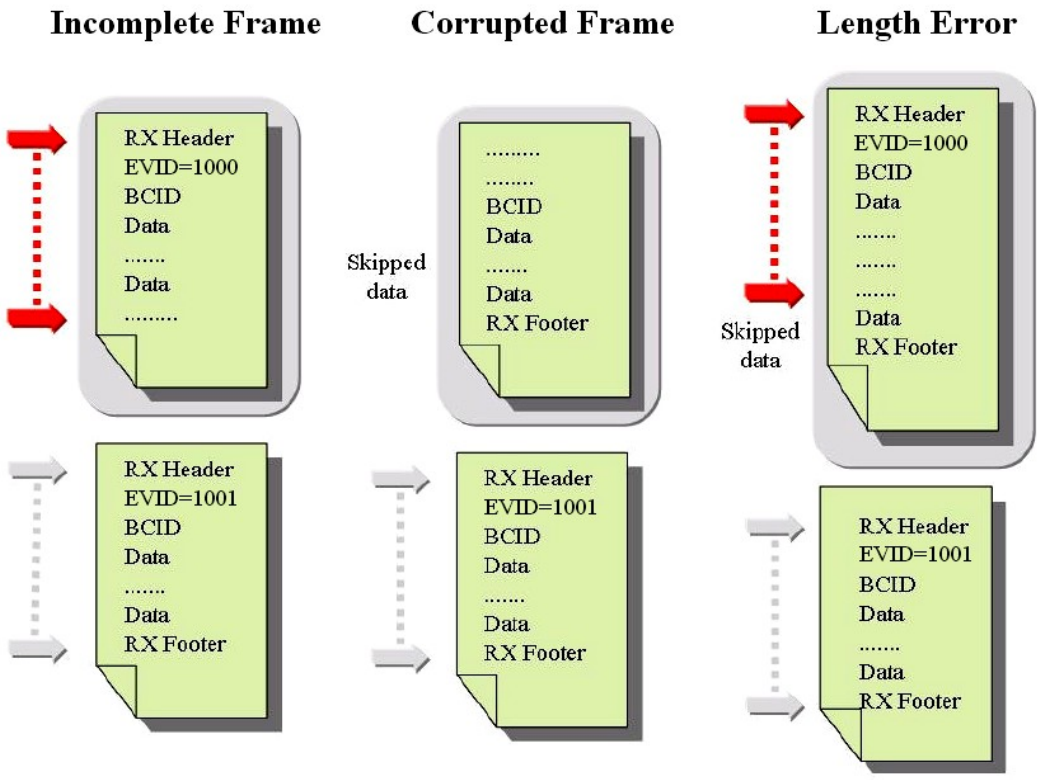


Figure 59 – The possible RX frame syntax errors.

A Length error check is necessary in order to avoid bandwidth penalties deriving from a noisy sector and to prevent from other formatting errors. An example of Length error is shown on the right column of Figure 59. In the case of a Length error, all RX frame's data exceeding *maxlength* are skipped. There will be a realignment on next Header: an error flag is set in a status word of the ROD Muon Frame and the frame is closed; then, all SerDes FIFO's data are dropped until the next RX Header is found. In this way, the next RX frame will be correctly aligned.

A *Timeout* error occurs when a SerDes FIFO goes empty during the event building process and no data is available in the FIFO within a programmable window. After a Timeout, the RX frame processing is halted, and the ROD frame is closed setting an error flag. The builder then realigns on the next available RX Header as soon as new data become available.

CHAPTER FOUR – THE MONITORING SYSTEM OF THE ROD EVENT BUILDER

This chapter describes the monitoring system of the Read Out Driver Event Builder. Such system is my original contribution to this thesis. Firstly an overview of the monitoring environment will be given. Then the hardware and the software architectures will be described. A description of the user software interface will also be provided. Finally simulations and test results will be shown.

4.1 The monitoring environment

In the data acquisition environment described in the previous chapters, a monitoring system is fundamental to analyze the activity of the Finite State Machines of the Event Builder, in order to find eventual errors or anomalous behaviours.

During my PhD, I developed a system embedded in the ROD FPGA, working in parallel with the Event Builder Engine. It is able to monitor the Builder Engine and, if required, to report monitored data via VME or UART.

The monitoring system is a hardware software co-design: it is based upon a microprocessor whom several hardware peripherals are connected to.

Its tasks are:

- To help to investigate upon eventual Event Builder errors;
- To allow to perform fine tuning and debugging of the Finite State Machines of its engine;

— To help the search of anomalous behaviours of the Read Out Driver or of the previous electronics, by keeping trace of states occupation and of the time spent in the event building process;
— To allow to easily find eventual unbalanced FIFO activities, that could be caused by errors from the on detector electronics.

In order to perform the tasks listed above, the monitoring environment should be able to fill histograms of the Event Builder FSMs' state occupation and plot trends of the event length of the event building time. It should also monitor the FIFO occupation flags and the error flags generated by the Event Builder Engine during the building process.

The most important requirement is to reach real-time performances: as said before, the Event Builder Engine is made up of different Finite State Machines that could change state every clock cycle. Keeping trace of such high rate changes means that is necessary a system able to "react" at every clock cycle. Furthermore, in order to track and debug the algorithm, the system must be able to process the data just acquired and to display them locally or send them to a remote computer via VME. A microprocessor is very useful for statistical analysis and data elaborations, but typical microprocessor latencies are several orders of magnitude higher of what needed in this application, synchronous to the 80 MHz clock of the ROD board (clock period of 12.5 ns). This means that a processor would never be able to reach the mandatory real-time performances.

In order to have the computational power and flexibility of a microprocessor, together with the high speed performances of a hardware solution, I adopted a hardware/software co-design approach, based upon an embedded microprocessor, whom different hardware peripherals are connected to. The peripherals' responsibility is to manage the real-time operations, while the microprocessor is responsible of the off-line data elaboration.

The microprocessor I chose is the soft-core Xilinx MicroBlaze v.7.00 [39], embedded in the ROD FPGA, and the development tools used to implement its environment and its application code (written in C language) are Xilinx ISE [40] and EDK [41] respectively. The MicroBlaze will be briefly described in the next paragraphs.

Every monitoring session is started by providing an OPerative CODE (OPCODE) to the monitoring system via a remote computer, using the SSH protocol. Such instruction reaches the VME CPU, that sends it to the MicroBlaze. The processor initializes its environment and waits for a building operation from the ROD Event Builder Engine to start. When the Event Builder starts building a new event, the monitoring process begins.

During the process the hardware peripherals work in real-time keeping trace of the Builder FSM state occupation, of the FIFO flags and the error flags produced by the Event Builder Engine and of the event length and event building time.

At the end of the building operation, monitored data are sent via VME and, if required, elaborated by the processor.

In order to give instructions to the monitoring system, a software interface has been developed ad-hoc.

4.2 The MicroBlaze processor

The MicroBlaze [39] is a soft-core processor, designed for FPGAs from Xilinx. The MicroBlaze is implemented entirely by using the memory and logic devices of Xilinx FPGAs. Its flexibility consists in the fact that its hardware environment is fully customizable: a user can add Intellectual Properties (included in the development tool's libraries) or also create its own peripherals, in VHDL, and add them to the MicroBlaze project.

The processor is a 32-bit RISC architecture. Its working frequency is programmable, and its maximum possible value depends on the FPGA upon the which it is implemented and on the design architecture. In our case, the FPGA is a Xilinx Virtex-II and the processor working frequency is 80 MHz.

MicroBlaze instruction execution is pipelined into three stages to minimize latency: *Fetch*, *Decode*, and *Execute*.

For most instructions, each stage takes one clock cycle to complete. Consequently, the number of clock cycles necessary for a specific instruction to complete is equal to the number of pipeline stages (i.e three clock cycles), and one instruction is completed on every cycle. A few instructions require multiple

clock cycles in the execute stage to complete. This is achieved by stalling the pipeline. In Figure 60 the three stage pipeline of the processor is shown.

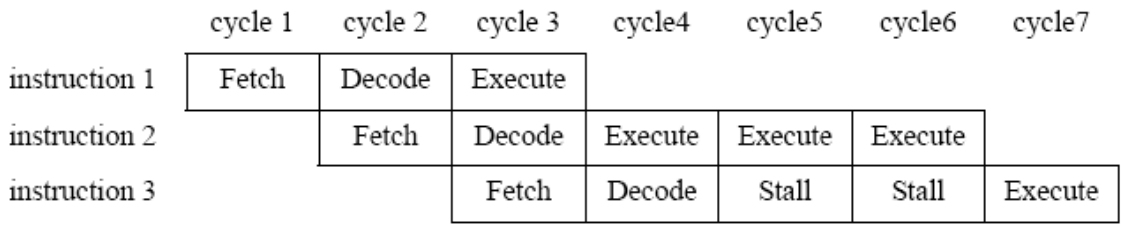


Figure 60 – The three stage pipeline behaviour of the Microblaze.

The MicroBlaze is implemented with a Harvard memory architecture: instruction and data accesses are done in separate address spaces. Each address space has a 32-bit range, that allows to handle up to 4 GB of instructions and data information.

The processor does not separate data accesses to I/O and memory: it uses memory mapped I/Os. This means that every peripheral register is accessed exactly like memory registers and, in some cases, by using the same memory interface. In our architecture, the processor has two interfaces for memory accesses:

- The LMB (Local Memory Bus) provides single-cycle access to on-chip dual-port block RAM;
- The OPB (On-chip Peripheral Bus) interface provides a connection to both on-chip and off-chip peripherals and memory blocks;

The LMB [39] is a synchronous interface, with a 32-bit wide address and data busses. It is used primarily to access on-chip block RAM. It uses a minimum number of control signals and a simple protocol to ensure that local block RAM are accessed in a single clock cycle. This feature makes the LMB suitable for interfacing the processor to application code's block RAM, allowing fast instruction fetching without interferences due to memory accesses by peripherals.

The On-chip Peripheral Bus (OPB) is an IBM interface designed for easy connection of on-chip peripheral devices (both masters and slaves) and block RAMs. In Figure 61 you can see the physical implementation of the OPB. Since the OPB supports multiple master devices, the address bus and data bus are implemented as a distributed multiplexer. The Xilinx version [42] supports 32-bit address and data busses, while the IBM architecture can support up to 64-bit address and data busses.

In our architecture, I connected on the OPB several peripherals and also a block RAM, reserved to the data produced by them.

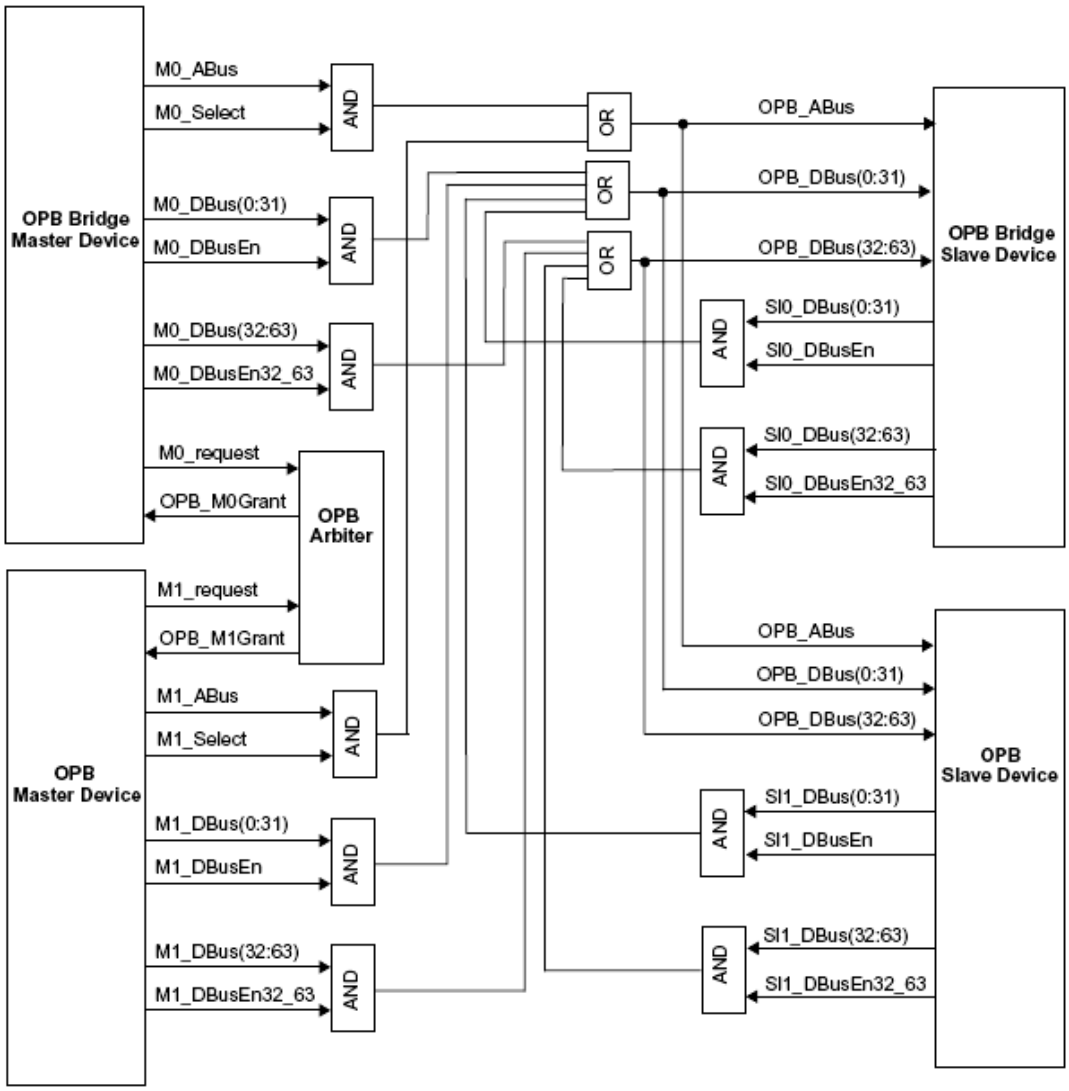


Figure 61 – Physical Implementation of the OPB.

In Figure 62 it is shown a simplified block scheme of the MicroBlaze design I developed. You can see the processor connected to an instruction memory block via the LMB bus, while a data memory block is connect via the OPB bus. To this last, also several peripherals are connected. In the example scheme shown in Figure 62, you can see a General Purpose I/O (GPIO) block, needed to transfer data in input and/or output direction, a UART interface and a custom peripheral.

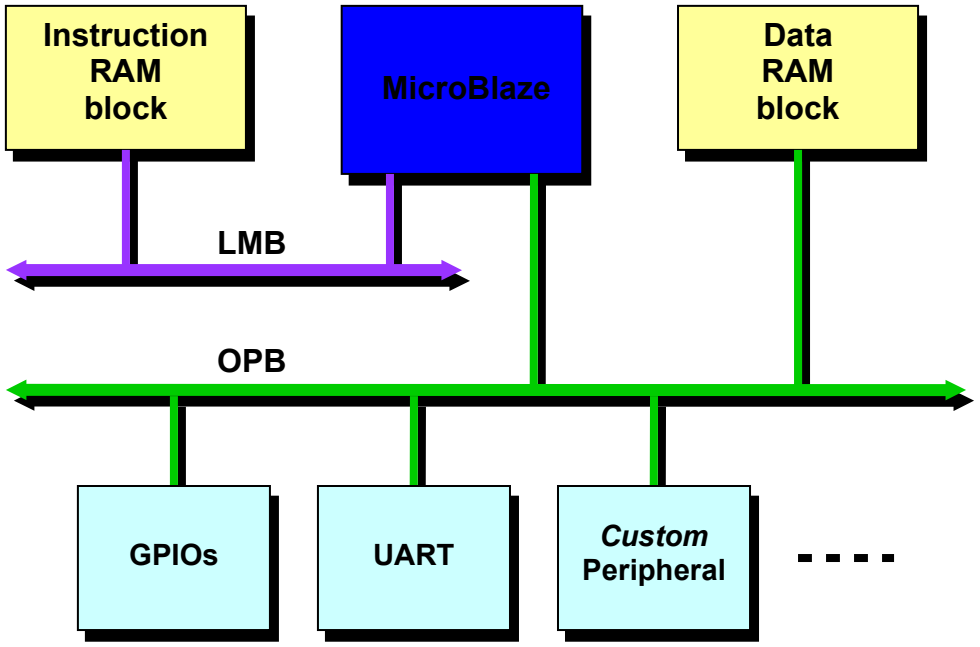


Figure 62 – Block scheme of a MicroBlaze design.

4.3 MicroBlaze environment overview

In Figure 63 a block diagram of the monitoring environment inside the ROD FPGA is shown.

On the top side of the image you can see the MicroBlaze connected via LMB bus to the instruction RAM block (labelled as LMB BRAM). On the right side, it is possible to see the OPB bus connecting a data RAM block (labelled as OPB BRAM) and several peripherals to the processor.

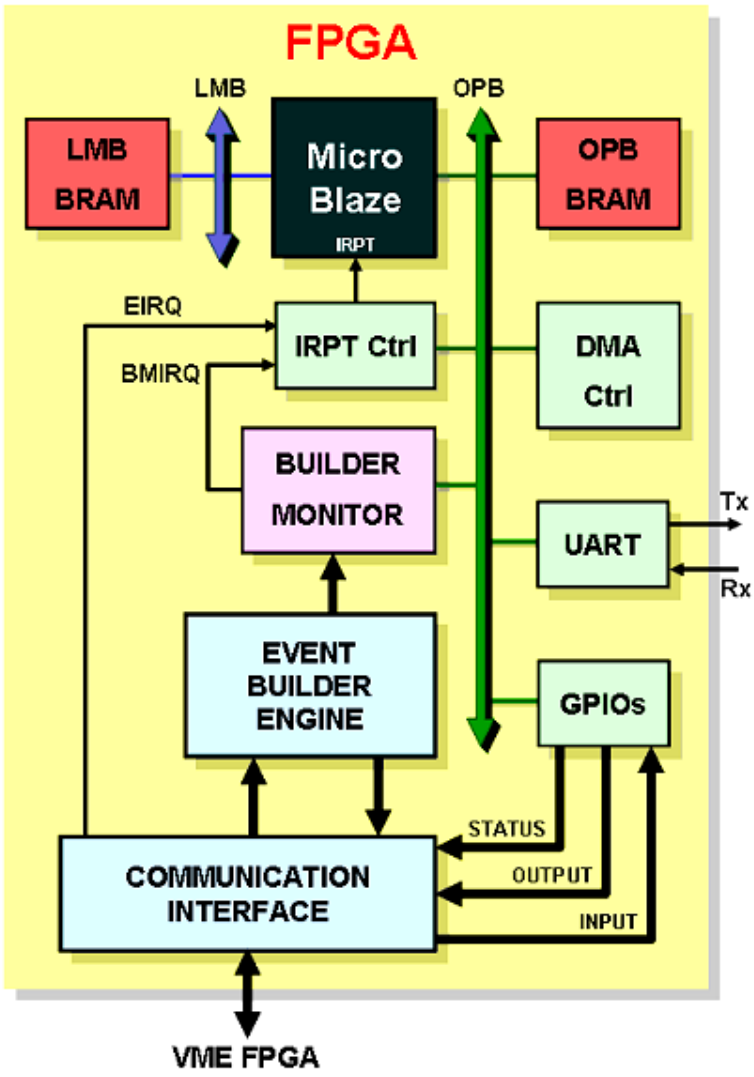


Figure 63 – Block diagram of the monitoring environment.

The core of my project is the Builder Monitor device (BM), designed specifically to perform the real-time monitoring of the Event Builder Engine. The BM is interfaced with the Event Builder Engine and, like all the other MicroBlaze peripherals, with the OPB bus. This allows the processor to save monitored data in the OPB block RAM.

In order to perform such data transfer with low latencies, I inserted into the project a Direct Memory Address Controller (DMA Ctrl). This device is able to perform high data transfers (a word every two clock cycles) without the intervention of the processor: a software driven transfer would take a time latency 4-5 times higher.

I designed the system in order to have low reaction times respect to user-given instructions and BM data storing requests. To increase such timing

performances, I based the processor application code upon interrupt requests. As you can see in Figure 63, I designed two interrupt sources:

- *External Interrupt ReQuest* (EIRQ), triggered every time a user gives an instruction to the monitoring system;
- *Builder Monitor Interrupt ReQuest* (BMIRQ), managed by the Builder Monitor device at the end of a monitoring operation, in order to let the MicroBlaze to save in the OPB block RAM the data just acquired.

To manage the interrupt sources' priorities and requests, I inserted in the design an interrupt controller, connected to the unique MicroBlaze interrupt line.

The serial protocol between the FPGAs, described in the paragraph 3.1.5, is represented in Figure 63 by the Communication Interface (CI) block. in order to manage all the communications with the VME FPGA, I designed a custom protocol with the CI. I implemented the signal of such protocol by using several General Purpose I/O (GPIO) devices.

I also added a UART (Universal Asynchronous Receiver and Transmitter) device, in order to print on a terminal the results of processor elaborations (histograms, plots and so on) and debug operations.

In the following paragraphs, I will present a description of the custom protocol I developed to manage the communications with the VME. A deep presentation of all the monitoring environment devices will also be provided.

4.3.1 The custom protocol with the Communication Interface block

As said before, every monitoring operation is started by an external OPCODE (Operative CODE) to the monitoring system, provided by a user via a remote computer. Such OPCODE reaches the VME CPU, that delivers it to the monitoring environment and then to the MicroBlaze.

In order to do this, the CI block, managing the connection between the VME FPGA and the ROD FPGA (paragraph 3.1.5), must be interfaced somehow to the monitoring system. I developed the communication protocol to perform such interface, in which the CI block is the master.

The interface is made up of two data buses (named *input* and *output* bus, respect to the monitoring system), the External Interrupt ReQuest signal (EIRQ) and the ACKnowledgement signal (ACK). All this signal are implemented by using several General Purpose I/O (GPIOs) devices, available in the MicroBlaze libraries.

In Figure 64 it is shown a block scheme of the custom communication protocol between the CI and the monitoring system.

The input bus is divided into two fields:

- the less significant three bytes contain an eventual data sent from the CI to the monitoring system;
- the remaining most significant byte contains the OPerative CODE (OPCODE). This is a word codifying the instruction to be executed by the processor.

The output bus is also divided into two fields:

- the less significant 28 bits contain an eventual data provided by the monitoring system;
- the most significant three bits are *Status* signals, all in negative logic:
    - *DAV* (Data AVailable) is the bit 29. When its value is a logic 0, it means that one or more data are available for being transferred from the monitoring system's data memory to the Communication Interface;
    - *DONE/DP* (DONE/Data Processing) is the bit 28. When an OPCODE involves the generation of several data words, this signal goes to 0 and does not change until the user reads the last of the available data. A logic 1, on the other hand, means that the processor is not executing any instructions and it is ready to receive a new OPCODE.
    - *LOP* (Legal OPCODE) is the bit 27. When its value is a logic 0, it means that the last received OPCODE has been recognized.

The CI starts every operation with the MicroBlaze by asserting a logic 0 upon the interrupt request line EIRQ. The falling transition (from a logic 1 to a logic 0) upon this line triggers the interrupt controller, that generates an interrupt request to the microprocessor, letting it to jump to the external interrupt routine. Such service routine reads from the input data bus the OPCODE and the eventual data from the relative fields.

The interrupt routine also manages the three status signals of the communication protocol: for example, if the received OPCODE is recognized as valid, the LOP flag will be asserted.

After the modification of the status field of the output bus, the interrupt routine provides the acknowledge by asserting a logic 0 upon the line ACK. When the CI receives the ACK stimulus, it reads the output bus and de-assert EIRQ. Data and signals just read by the CI block are then sent to the VME FPGA.

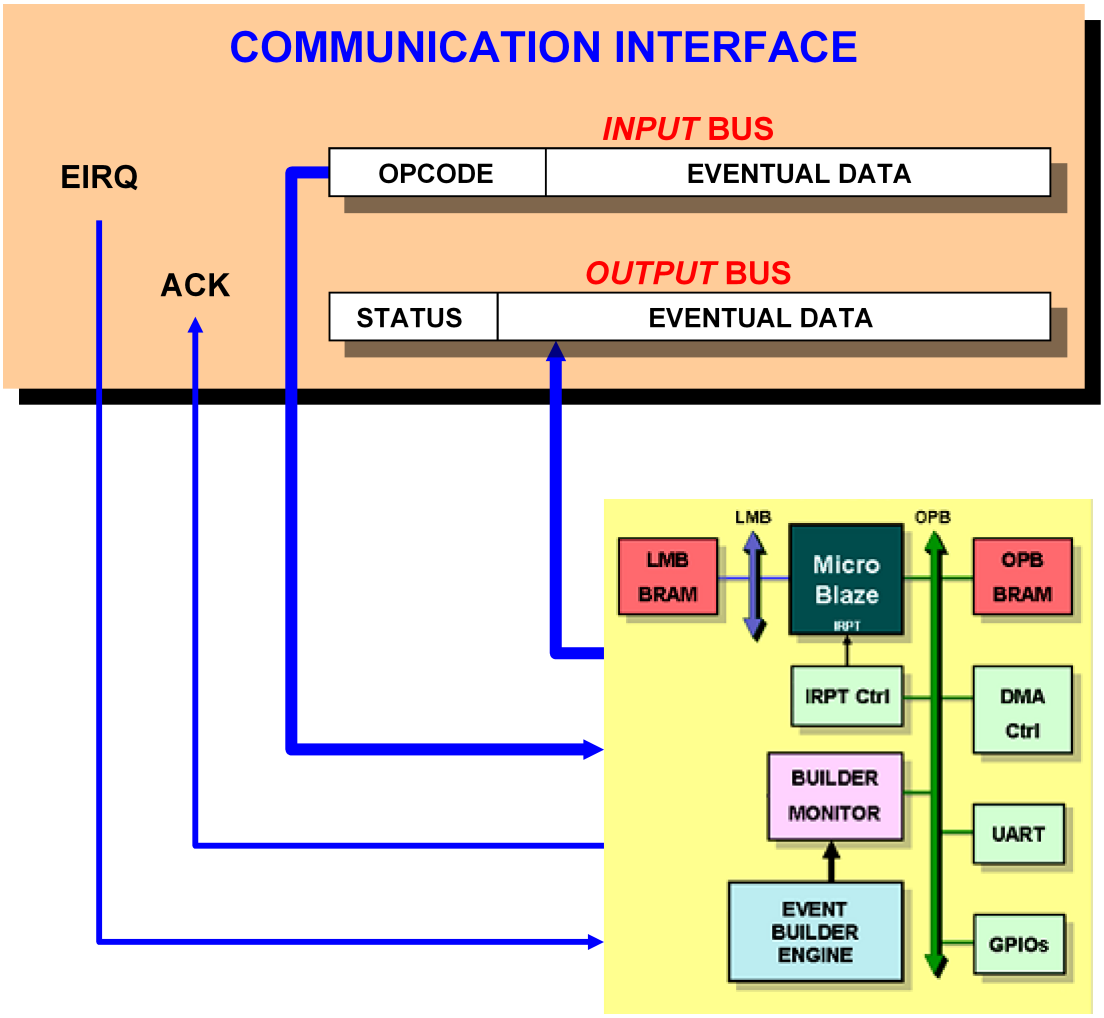


Figure 64 – Scheme of the protocol between the CI and the monitoring system.

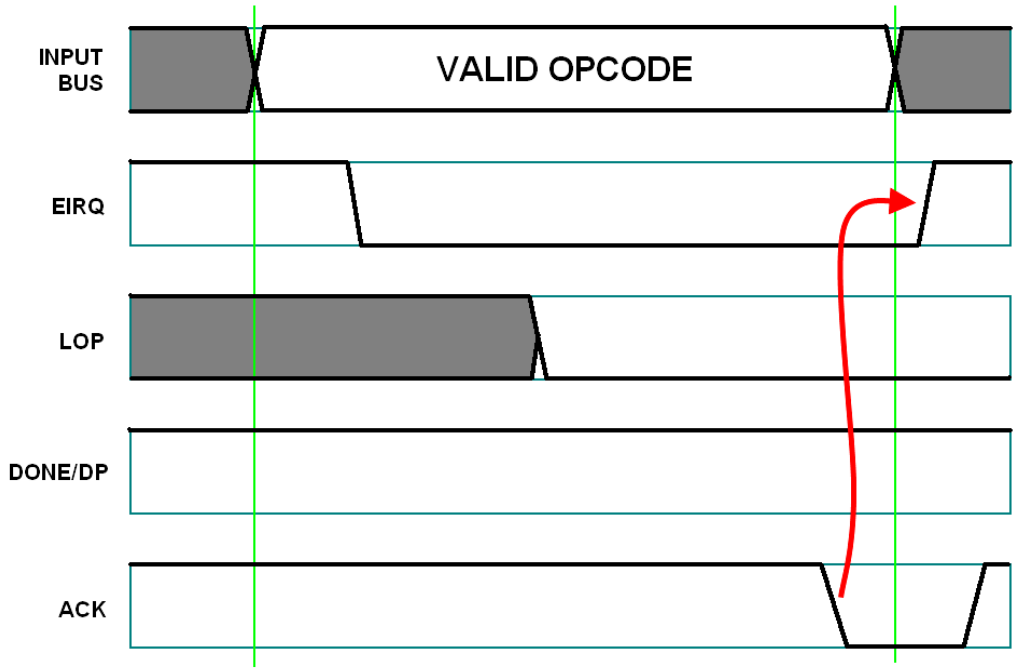


Figure 65 – A fast-OPCODE provided to the processor.

In Figure 65 a communication example is shown: it represents a recognized *fast-OPCODE* (not involving the production of data) provided by the user. It is possible to see the EIRQ signal going down.

This falling edge transition makes the processor to read the OPCODE from the Input Bus, to modify opportunely the status bits (in the example only the LOP flag) and eventually providing the Acknowledgement. When ACK goes low, the CI interface releases the Input Bus and de-asserts EIRQ.

In Figure 66 it is possible to see a data producing OPCODE, followed by a *Read* operation. The protocol signal transitions on the left side of the image respect the rules just described but, in addition, the signal DONE/DP goes down, indicating that the OPCODE just received generated several data words to be sent. When the first of such words becomes available, the processor asserts the DAV signal.

The right side of the image shows a *Read* access. You can see the ACK signal strobing a logic 1 upon DAV and a 0 upon DONE/DP. This means that the data buffer to be sent to the VME has not been entirely read yet, but the processor has not yet prepared the next data word to be sent.

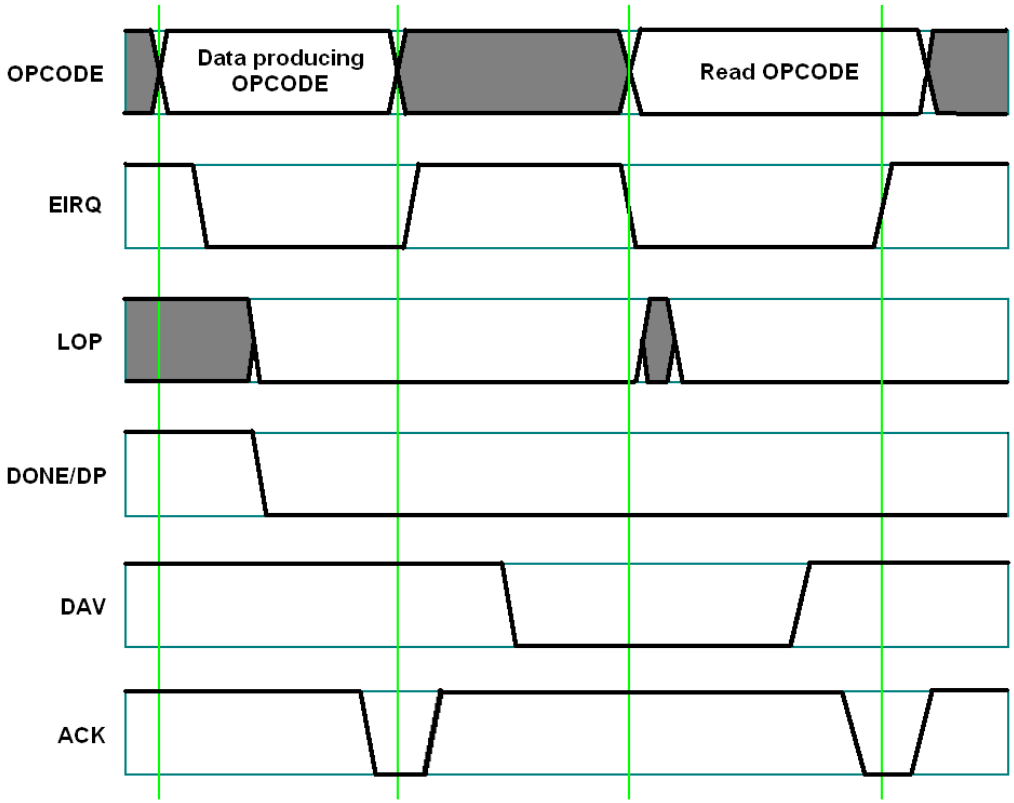


Figure 66 – A data producing OPCODE, followed by a *Read* operation.

4.3.2 The Builder Monitor

The Builder Monitor (BM) peripheral is the most important element in this environment as it performs the real-time monitoring of the Event Builder Engine's Finite State Machines.

Since the Event Builder is described by 17 macro-states (paragraph 3.2.1), the BM contains a register file of 17 32-bit counters (one for each Event Builder state). In this way it is possible to count how many clock cycles the Builder Engine spends in each state. This *profiling* allows to understand if there are anomalous delays or pauses in any state. Moreover, as it is possible to distinguish the Frame Maker states from the left/right FIFO Reader ones, it is possible to discover eventual unbalanced FIFO activities by checking the correspondent state occupations.

The BM can work in two modes: *one time* and *logging*. When the *one time* mode is set, the peripheral monitors only the next event being built, while when the *logging* mode is selected, the events are monitored continuously: the counters are not reset after each event, providing an integral analysis of the

FSMs. The monitoring, in the *logging* mode, proceeds until the overflow of one of the 17 counters occurs.

The block diagram of the BM is shown in Figure 67.

The *Builder State* bus (on the left side of the figure) transmits the current state of the Builder Engine FSMs to the Builder Monitor. The bus *Data* (on the bottom side of the figure) contains some words produced by the Event Builder Engine.

Such words are stored into four internal data registers. They contain information about the event:

- *Event Length* contains the length of the event just built, expressed in number of words;
- *Event Time* is the number of clock cycles spent by the Event Builder in the building procedure;
- *Error Flags* is a word made up of flags indicating the errors occurred during the event building;
- *FIFO Flags* contains the FIFO status flags;

The signal *Building Is On* is asserted and de-asserted by the Builder Engine, respectively at the start and at the end of an event building process. When this signal is active, the *Counting Management Logic* block enables the monitoring by incrementing the counter correspondent to the actual Event Builder state.

On its de-assertion, the four data registers are also acquired and stored.

Every time a monitoring process ends (in both the working modes), the *Interrupt Management Logic* block asserts an interrupt signal request, called BMIRQ in the right side of the figure. The correspondent handler routine starts a DMA data transfer operation in order to push into the OPB BRAM memory all the data just acquired by the monitor.

I designed the inner logic of the Builder Monitor device ad a Finite State Machine, shown in Figure 68.

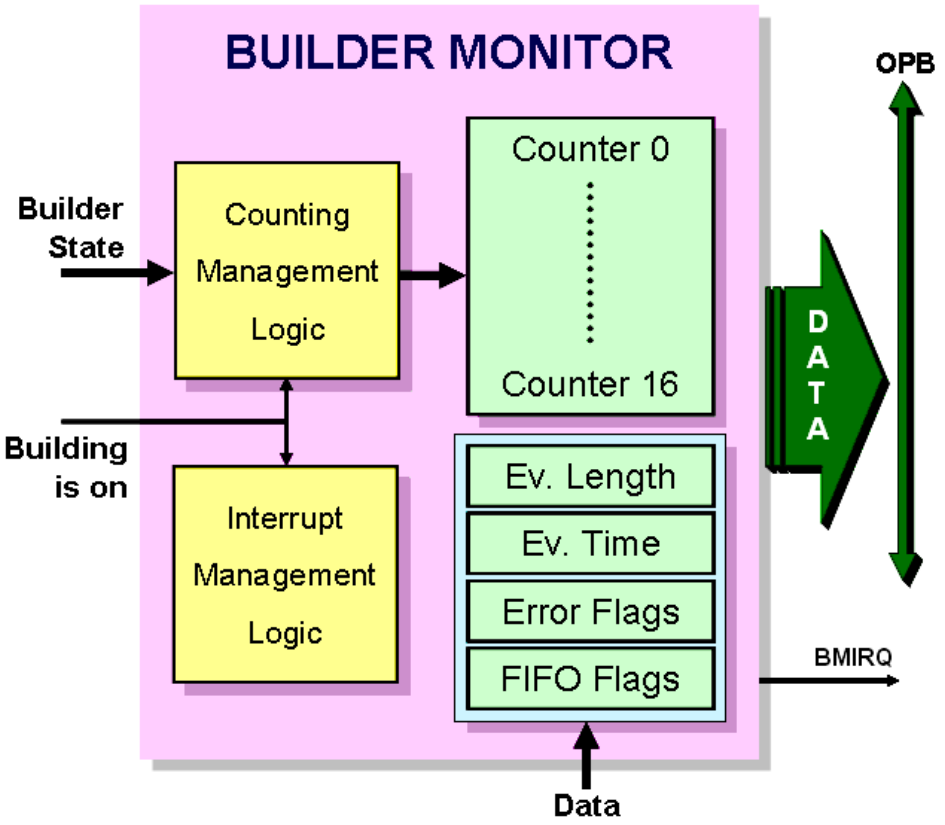


Figure 67 – Block diagram of the Builder Monitor peripheral.

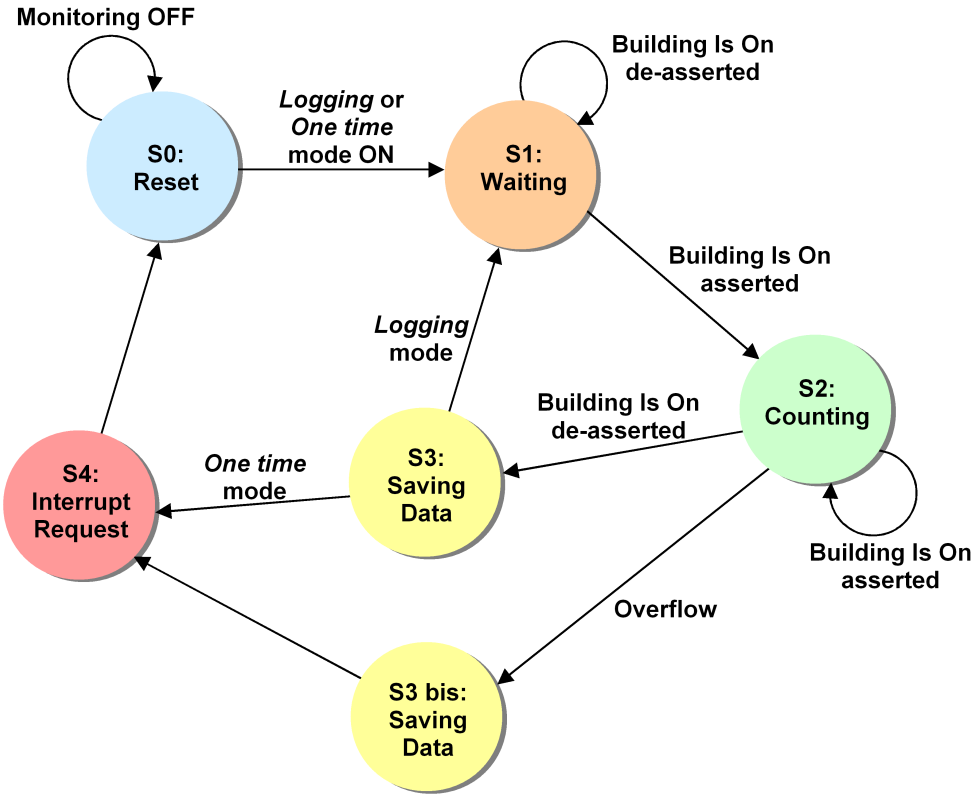


Figure 68 – Finite State Machine implementing the Builder Monitor logic.

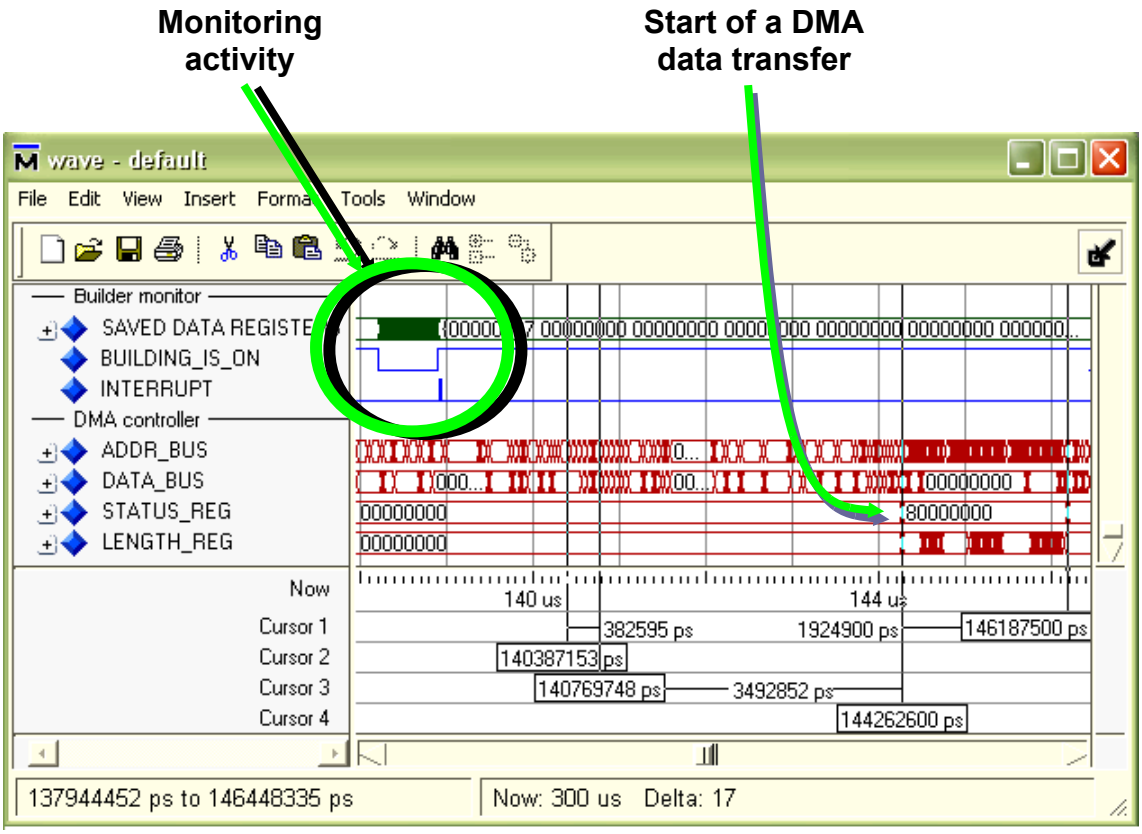


Figure 69 – Simulation of an Event Builder monitoring session.

In Figure 69 a simulation of an Event Builder monitoring session is shown.

In, at the left of the screenshot, several internal signals of the BM peripheral are listed.

On the left side of the screenshot window, in the section labelled *Builder monitor* and *DMA controller*, you can see the list of some internal signals respectively of the Builder Monitor and of the DMA Controller peripherals. In the circle it is highlighted the monitoring activity. It is possible to see the fast changes of the *SAVED DATA REGISTERS* signals (representing the 17 BM inner counters) when the *BUILDING_IS_ON* signal is asserted. At the end of the monitoring process, it is possible to see a pulse upon the *INTERRUPT* signal (the BM Interrupt Request).

At the right of the image, the start of a DMA data transfer is pointed by an arrow.

4.3.3 The DMA Controller device

Data transfer operations between the BM peripheral and the memory are optimized by using a Direct Memory Access controller (DMA Ctrl) [43].

The DMA Controller used is a library device, providing simple Direct Memory Access (DMA) services for peripherals and memories on the OPB bus. The controller moves a programmable quantity of data from a source address to a destination address without processor intervention. In our case, the source address is the 17 BM inner counters, while the destination is the OPB BRAM.

In Figure 70, the DMA Controller block diagram is reported.

Once started, the DMA operation proceeds by reading source-address data into the internal 16-word data buffer, then writing the data from the internal buffer to the destination address. This repeats until all data is moved. The registers update as the DMA operation progresses.

The status of the DMA operation is available in the DMA Status Register (DMASR): it contains several flags indicating occurred errors, busy status, timeouts and so on.

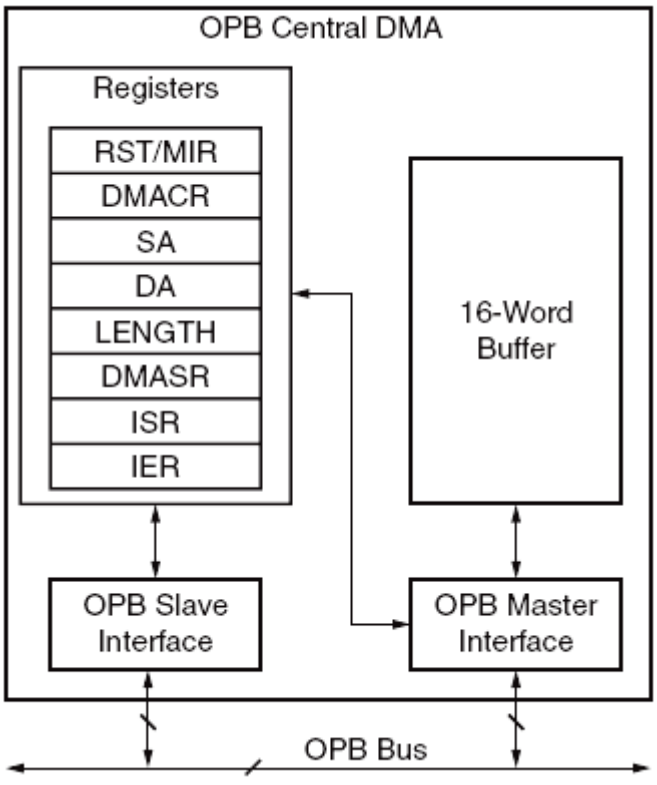


Figure 70 – DMA Controller block diagram.

A DMA operation is set up and started by writing values into the following registers:

- DMACR. This is the Control Register, needed to initialize the Controller. For example it is possible to specify the number of bytes compounding each word to be transferred;
- SA: contains the source address for the transfer;
- DA: contains the destination address for the transfer;
- LENGTH: Contains the number of bytes to transfer during the all DMA operation. Writing into this register will start the DMA operation.

ISR and IER are respectively the Interrupt Status Register and the Interrupt Enable Register.

The peak transfer rate is up to a word (32 bits) every two clock cycles. The DMA controller reduces the data transfer latency on the OPB by a factor 4 for each 32-bit word, with respect to software driven cycles.

4.3.4 The Interrupt Controller device

In order to manage the two interrupt sources of the environment – EIRQ and BMIRQ – I inserted into the design an interrupt controller peripheral (IRPT Ctrl) [44].

The Interrupt Controller is used to collect the interrupts from the sources and then apply prioritization to them. The interrupt service captures and multiplexes the two interrupt signals into a single interrupt output line that is connected to the microprocessor's internal interrupt controller. The service also provides local registers that the processor application code can utilize to read interrupt status, set up priorities to the interrupt sources, set up masking criteria and perform interrupt clearing for the individual interrupts. It is also possible to assign the proper service routines to the corresponding interrupt sources: in this way, the processor is able to jump to the assigned routine when the corresponding interrupt request is asserted. Should the two interrupt request signals be asserted simultaneously, the controller would register both, serve firstly the one with the highest priority (EIRQ, in our case) and eventually the latter.

4.4 The software architecture

I designed the processor OPCODE set in order to include commands relative to data acquisition and processing: if required, the MicroBlaze can plot bar charts of the 17 BM internal counters and trends of the Event Length and Event Time with respect to the number of event monitored. A simplified flow chart of the code is shown in Figure 71.

The application code is made up of a Main routine and two interrupt service routines, pertaining to the external interrupt request (EIRQ routine) and to the BM interrupt request (BMIRQ routine).

The Main routine is responsible of the execution of the received instruction. It manages also the DMA data transfers. Besides the initialization procedures, the Main routine tests the flags *OPCODE* and *BMIRQ*, respectively set by the EIRQ and BMIRQ routines.

When the OPCODE flag is set, the MicroBlaze executes the last instruction received. If the BMIRQ flag is set, the processor initializes a DMA data transfer from the BM peripheral to the OPB BRAM memory.

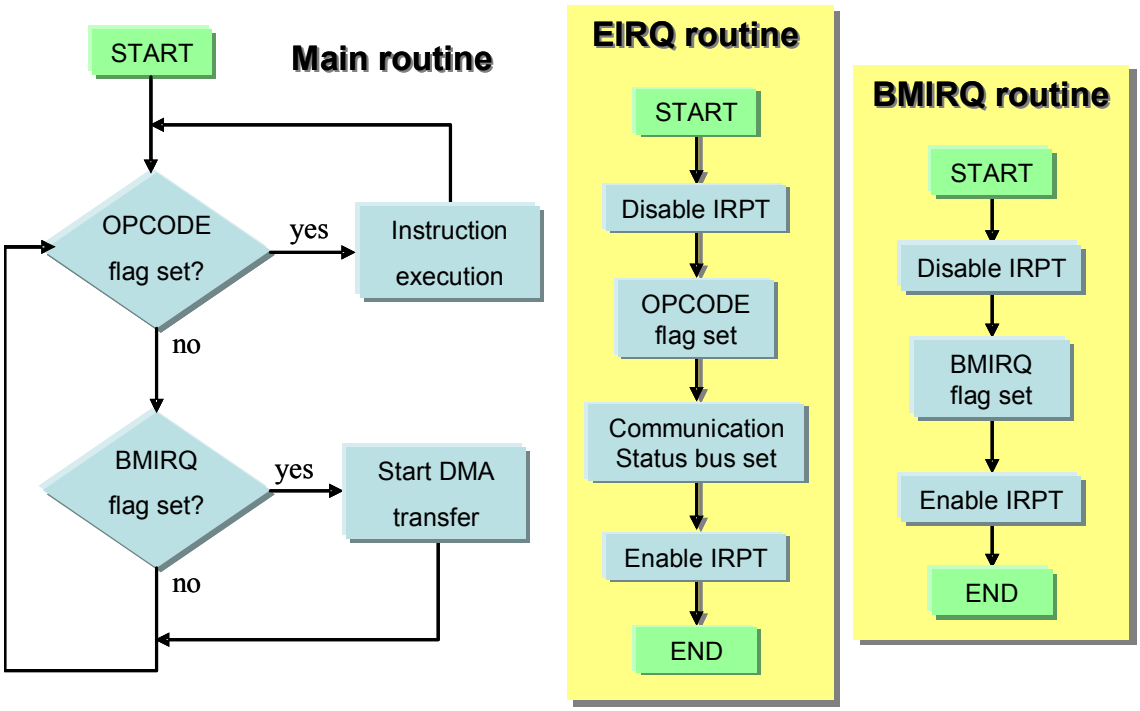


Figure 71 – Flow chart of the application code.

At the end of the instruction's execution or DMA data transfer, a report and debug information can be printed, if required, via UART on a terminal and then the application code returns to the polling of the flags.

The EIRQ routine is accessed every time the EIRQ signal is triggered by the Communication Interface. To avoid serving other interrupts while the one just received is still in elaboration, the EIRQ routine firstly disables interrupt requests. Then the OPCODE flag and the *Status* field of the Output bus (paragraph 4.3.1) are set, the ACK signal is provided and finally the interrupt requests are re-enabled.

The BMIRQ routine is triggered by the BM interrupt request. As for the EIRQ, this routine firstly disables interrupts, then it sets the BMIRQ flag and, before exiting, it eventually restores the interrupt requests.

4.5 Simulations and tests

In order to test the BM performances and reliability, I synthesized a benchmark of events, studied to cover as many Event Builder states as possible. The events format respect the RX Frame format, described in the paragraph 3.1.1. The benchmark also forces accesses to infrequent states of the FSMs and checks the BM response to Event Builder possible timing errors.

I validated the benchmark of events by performing a simulation of the BM's behaviour and then I run the BM through the same events, in order to compare its results with the simulated ones. In each test session, the BM always gave the same results seen in the simulation. These preliminary tests were preformed upon a ROD board installed in a laboratory located inside the department of Physics of the University of Naples "Federico II".

In Figure 72 you can see the simulation window of one of the benchmark events created for the testing of the monitoring system. It is possible to see the start of the event building operation, corresponding to the assertion of the *BUILDING_IS_ON* signal.

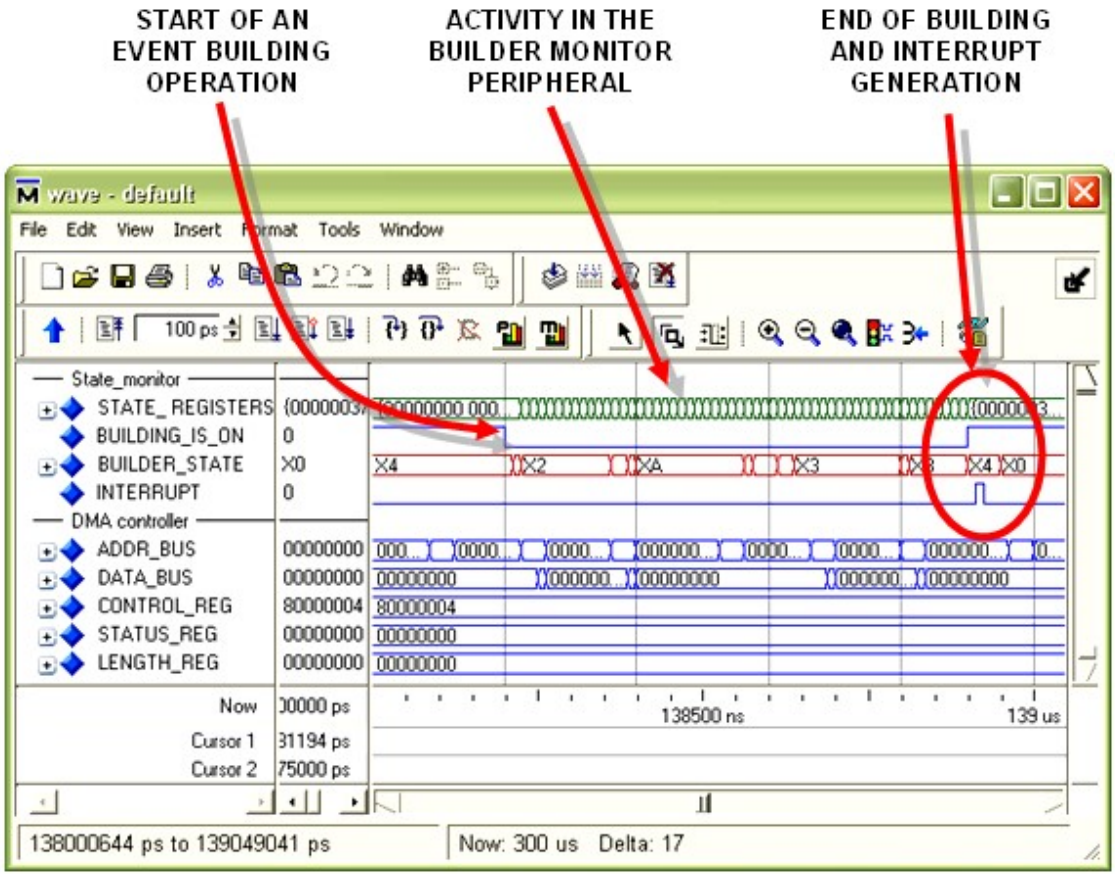


Figure 72 – Simulation of an event provided to the ROD Event Builder Engine.

As in the ATLAS environment the ROD board builds events with high rates, I performed simulations of thousands of event building operations executed continuously, by filling the ROD FIFOs before giving the start of the building procedure. In order to do this I wrote a software able to create automatically “random events”, made up of random data but respecting the RX frame format shown in the paragraph 3.1.1.

In Figure 73, a screenshot image shows a bar chart of some data produced by the Microblaze and representing the Event Builder States’ profiling for an event processed by the Builder Engine. At the top, the Event Time (in clock cycles) and the Event Length (in number of words) are written. The bar chart is divided in three sections, one for each FSM of the Event Builder (FRAME MAKER, FIFO READER RIGHT and FIFO READER LEFT). On the x-axis, the states of the three FSMs of the Event Builder Engine are reported, while the y-axis indicates the clock cycles spent by the machines in each state.

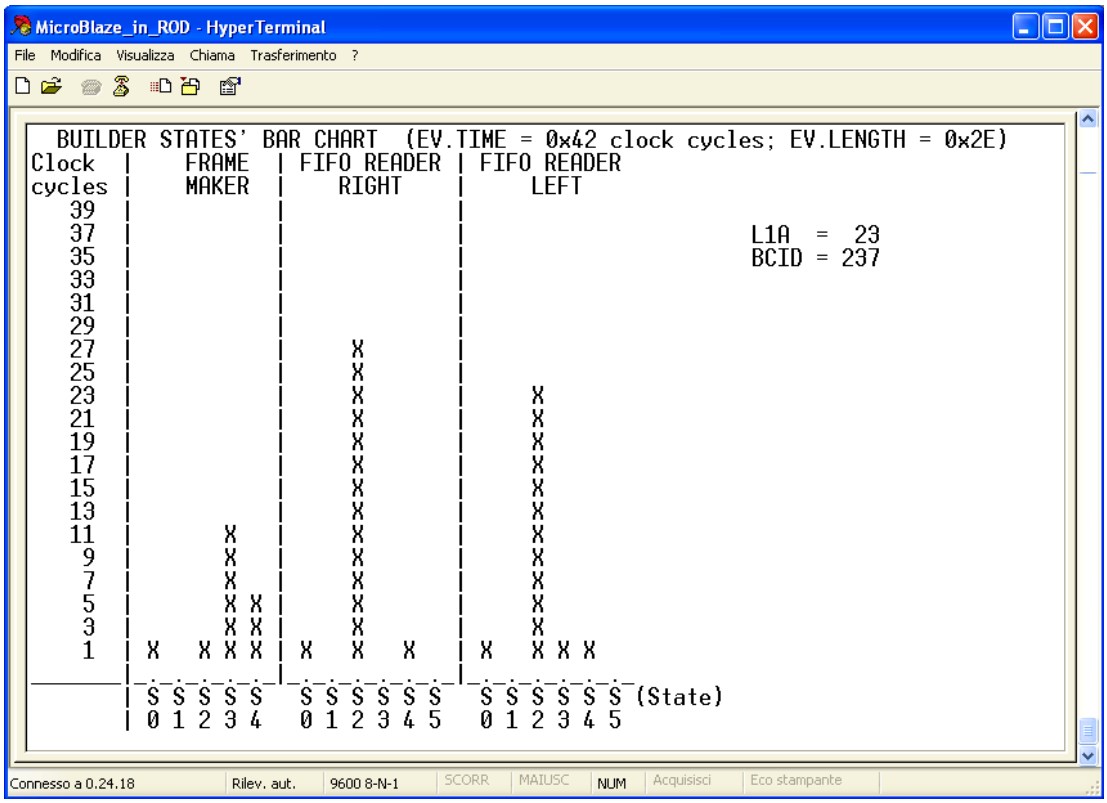


Figure 73 – Bar chart of a Builder States' profiling (laboratory).

At the right side of the bar chart, the Level 1 Accept number (L1A) and the Bunch Counter ID (BCID) of the monitored event are written.

The bar chart allows to find anomalous behaviour of the Event Builder. If the FSMs dwell on some states rather than others, we are able to understand where the error source is and eventually fix it. It also permits to easily see unbalanced data flow due to noisy RPC read-out channels or uneven loading of the DAQ system.

In Figure 74 and Figure 75 the Event Length (in number of words) and the Event Time (in nanoseconds) data registers are plotted respect to the Event number. The first point (Event nr. = 1) of both plots corresponds to the Event data shown in Figure 73.

We can also check the correlation between event lengths and event building elapsed times. A non-linear relation between them could be due to a delayed arrival time of data from the detector.

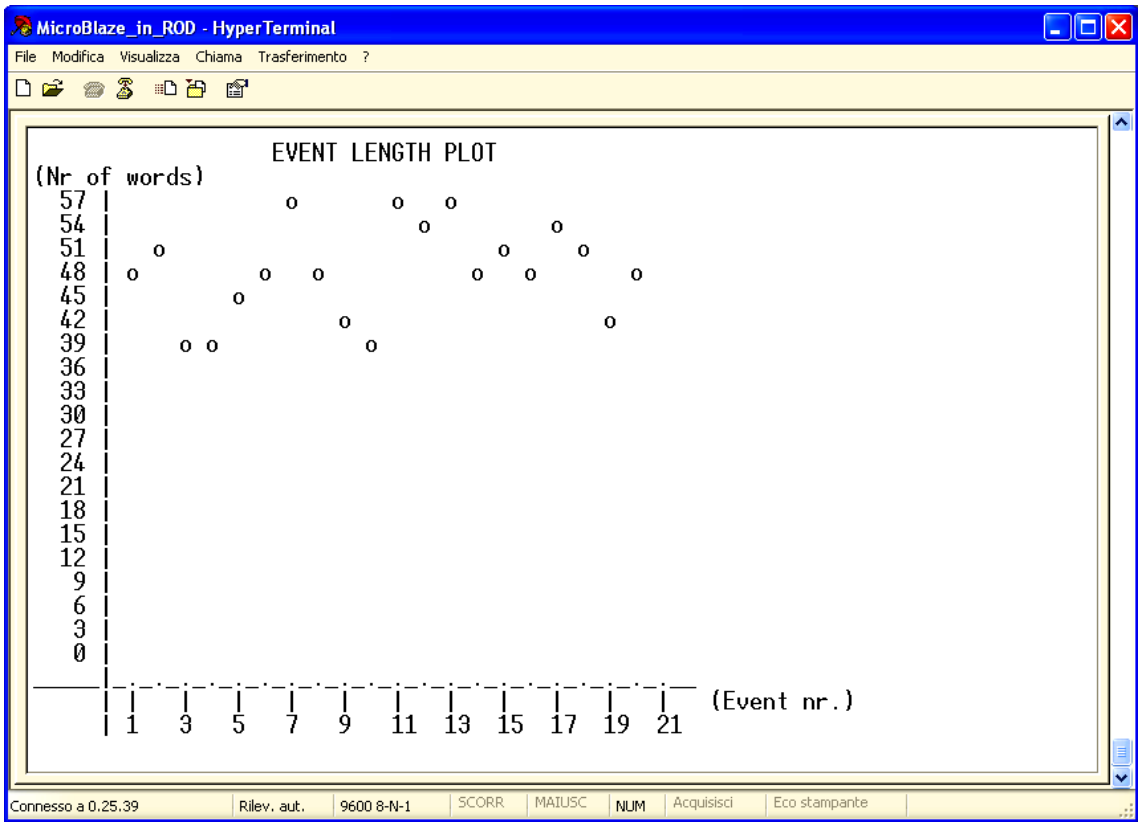


Figure 74 – Plot of the Event Length data register respect to the Event number.

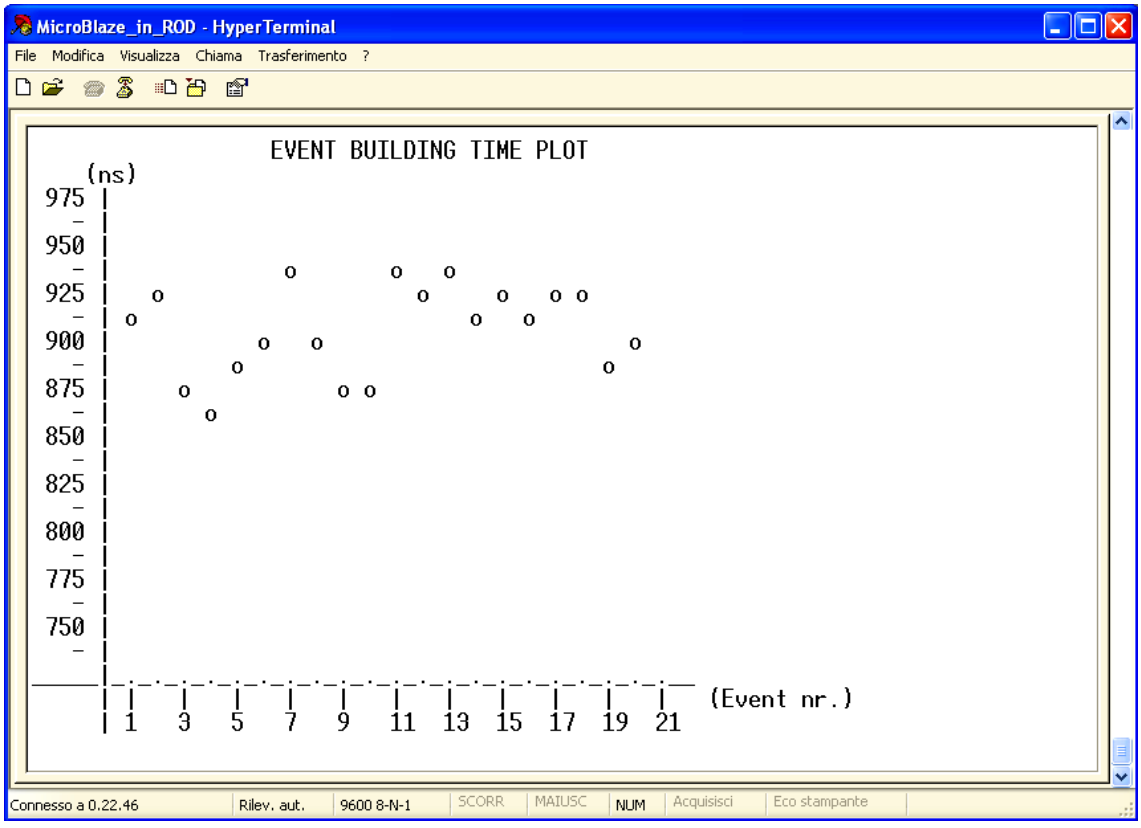


Figure 75 – Plot of the Event Time data register respect to the Event number.

4.5.1 Commissioning at CERN

During the ATLAS technical stop occurred in the early days of Sept. 2010, the ROD FPGA firmware, integrated with the monitoring system, was installed on a ROD board in order to begin the commissioning in the experimental environment.

Such board is located with the other RODs in the ATLAS counting room (called USA 15), nearly 80m from the ATLAS detector. In Figure 76 you can see a picture of the VME crate hosting the ROD board upon the which the monitoring system is installed.

Firstly, a test of the correct functioning of the communication protocol was performed, by connecting via SSH to the ROD and providing instructions to the MicroBlaze.

Then, at the end of the technical stop, the Event Builder was tested during physics runs, by monitoring events produced by the collisions inside the LHC.

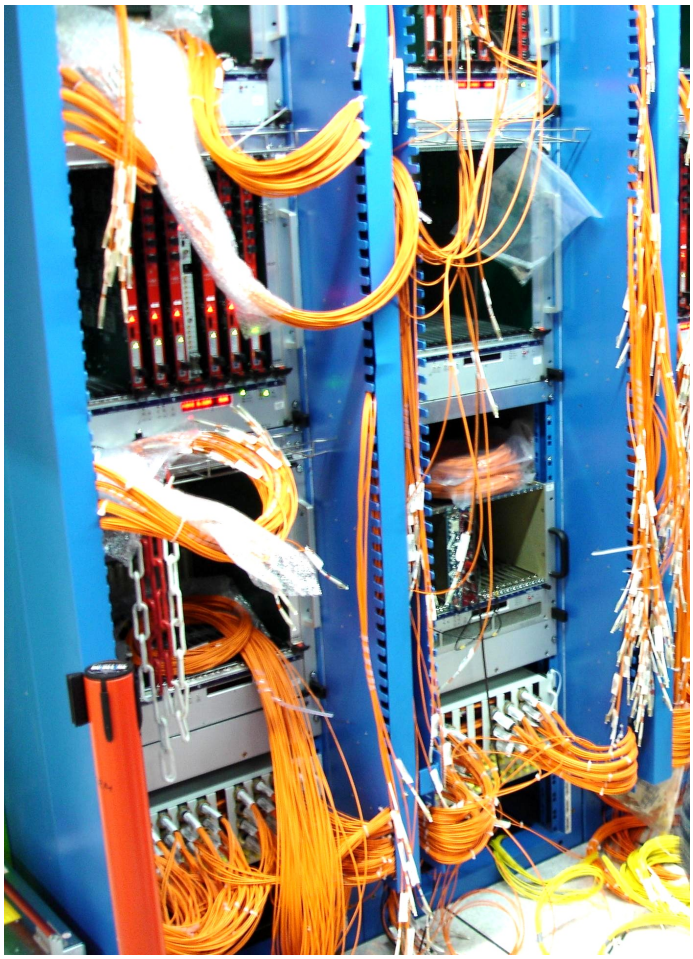


Figure 76 – The ROD in a VME crate at the USA 15 counting room.

In Figure 77 you can see a histogram of the Event Builder monitoring data, taken at the CERN during an ATLAS run. You can see the analogy with Figure 73. In this last, the S2 state occupation of both the left and right FIFO Reader FSMs are lower than the ones reported in Figure 77: this is due to the fact that the built event reported in Figure 77 was greater than the one represented in Figure 73 (as you can check by comparing the Event Length words reported on the upper-right side of both the histograms).

It is possible to see also that the S1 state occupancy of the left FIFO Reader FSM is higher than zero. From Figure 55, it is clear that the S1 is a *waiting* state, in which the FSM waits for data produced by the SerDes.

From Figure 54, it is also possible to see that the left SerDes FIFO is the first one to be read. This means that an S1 occupancy state higher than zero is acceptable for the left FIFO Reader FSM, because it indicates that the Event Builder Engine waited for the SerDes FIFO to be filled.

Moreover, as both the left and right SerDes FIFOs should be filled together, at the end of the work of left FIFO Reader FSM, the Engine should find the right SerDes FIFO already filled and the relative S1 state occupancy should be zero.

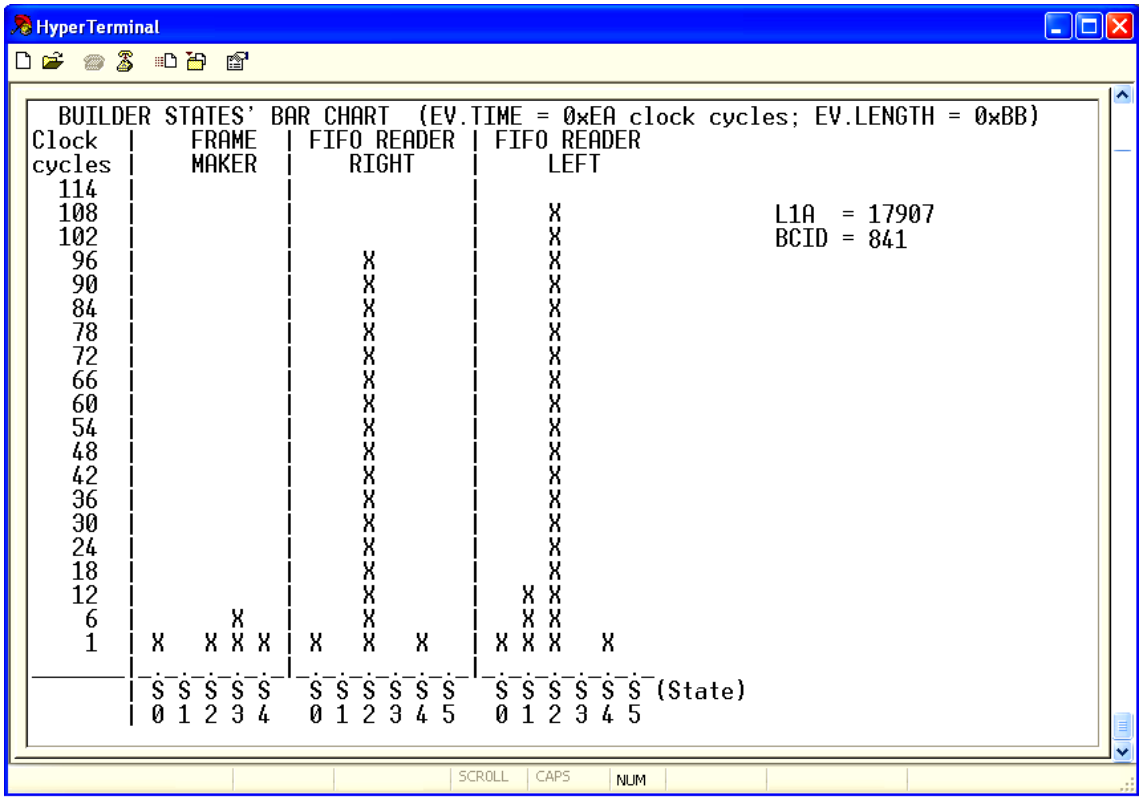


Figure 77 – Bar chart of a Builder States' profiling (CERN).

if such occupancy was been higher than zero (in Figure 77), it would have highlighted a possible unbalanced activity of the FIFO Reader FSMs or of the left/right SerDes channels.

The monitoring system is useful also to provide the ROD data rate. By performing a *logging* mode monitoring (paragraph 4.3.2), it would be possible to understand the amount of data words produced by the Event Builder Engine in a certain time interval.

The tests at the CERN will continue until the next technical stop, then the monitoring system will be installed upon the remaining ROD boards.

Conclusions

During my PhD research activity I developed a monitoring system for the Event Builder of the ATLAS Muon Spectrometer Read Out Driver.

During the initial phases of the project, I had to choose the design philosophy upon which basing my work. In order to do this, I studied the Read Out Driver (ROD) board, its tasks and its architecture.

The ROD is a VME board that receives via optical fibres read-out data from the on-detector electronics of the ATLAS' RPC Muon spectrometer.

The main task of the ROD is to arrange all the data fragments of one sector of the spectrometer in a unique event. This is made by the *Event Builder* logic, a cluster of Finite State Machines (FSMs) that parses the fragments, checks their syntax and builds an event containing all the sector data. Such FSMs are implemented inside a Xilinx Virtex-II FPGA, hosted upon the ROD board.

As such FSMs could change state every clock cycle, a mandatory task of the monitoring system I had to design was to reach real-time performances.

Moreover, a very useful feature a monitoring system should have is the possibility to make elaborations upon the monitored data and to present them in a manner suitably clear.

A processor is a perfect solution in data elaboration and presentation, but it would be too slow to perform real-time monitoring. On the other hand, a hardware solution is very fast and can easily reach real-time performances, but it is too less flexible in data elaboration.

I chose to adopt a hardware-software co-design, by developing an embedded system based upon a microprocessor and several hardware devices. Such system is designed inside the same FPGA hosting the Event Builder.

The processor is interfaced to two RAM memories, several general purpose devices (like an Interrupt Controller, a DMA Controller, a UART interface, and so on) and to a custom device, named *Builder Monitor*.

This last is responsible of the Event Builder Engine monitoring and I designed it as a Finite State Machine. Since the Event Builder can be described

by 17 macro-states, the Builder Monitor (BM) contains a register file of 17 32-bit counters (one for each Builder Engine state). In this way it is possible to count how many clock cycles the Builder Engine spends in each state. This *profiling* allows us to understand if there are anomalous delays or pauses in any state.

The BM can work in two modes: *one time* and *logging*. When the *one time* mode is set, the peripheral monitors only the next event being built; when the *logging* mode is selected, the events are monitored continuously: the counters are not reset after each event, providing an integral analysis of the FSMs.

At the end of an event building process, the Builder Monitor saves information upon the event just built. Such information are the Event length (in number of words), the Event building time (in clock cycles), the Error flags (asserted by the Builder Engine during the building process) and the FIFO occupancy flags.

When the real-time performances are no more needed, at the end of the event building operations, the microprocessor can build histograms of monitored data, draw plots of the event length and event time words respect to the event number (in order to give a trend of the Event Builder activity), report via UART debug information and send via VME monitored data.

In order to commit the system, many simulations were performed, covering a lot of situations the Event Builder Engine could face.

The monitoring system was deeply tested in laboratory and it is now implemented in a single ROD board at the CERN, installed the ATLAS experiment environment, in order to perform the final commissioning.

Up to now, all the tests performed were successful.

Appendix A

In this appendix, the meanings of all the 32-bit internal registers of the two FPGAs used on the ROD boards are presented.

A.1 The VME FPGA registers

In the VME FPGA, eight 32-bit configuration registers have been implemented. The meanings of each register are now explained. The RESETS register concerns to the reset pins - of the devices connected to the VME FPGA - that can be driven by the VME FPGA. Figure A.1 shows the data of the RESETS register. All bits are active low and are readable/writable (R/W).

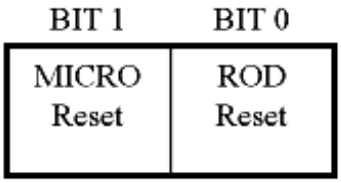


Figure A.1 – The VME FPGA RESETS register.

Bit 0 is the reset of the ROD FPGA. Bit 1 is the reset of the Microcontroller hosted by the board. The remaining bits are reserved. Fig. A.2 shows the content of the *Geographical Address* register. It's made of six bits that allow to identify the position of the board inside the VME crate. It's a *Read Only* register. The remaining bits are reserved.

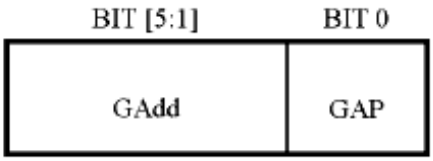


Figure A.2 – The content of the *Geographical address* register.

Figure A.3 shows the data of the *TTC_Config* register. It's the register for the configuration of the TTC receiver. The three least significant bits (*TTC Ready*, *TTC Locked* and *TTC Error*) are *Read Only* and allow to retrieve information about the working status of the TTC receiver. The *QPLL mode* (bit 3) allows to choose the frequency multiplication mode (120 MHz or 160 MHz) of the QPLL (*Quartz based Phase-Locked Loop*) of the TTC receiver.

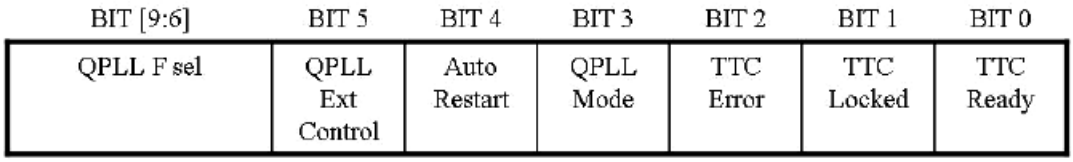


Figure A.3 – The content of the *TTC_Config* register.

The *Auto restart* (bit 4) and the *QPLL Ext control* (bit 5) allow to select the operations to be performed by the QPLL of the TTC receiver each time the lock is lost [see qpll_manual for more details]. The 4-bit field *QPLL F sel* (bit [9:6]) allow to control the *VCXO* (*Voltage Controlled Crystal Oscillator*) free running oscillation frequency. The remaining bits are reserved. Figure A.4 shows the data of the *Microcontroller_data* register. It's the register that allows the data exchange between the VME FPGA and the microcontroller. It's made of two bytes that host data that are involved in the transactions with the Microcontroller. The remaining bits are reserved.

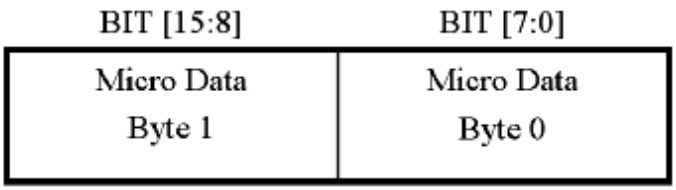


Figure A.4 – The content of the *Microcontroller_data* register.

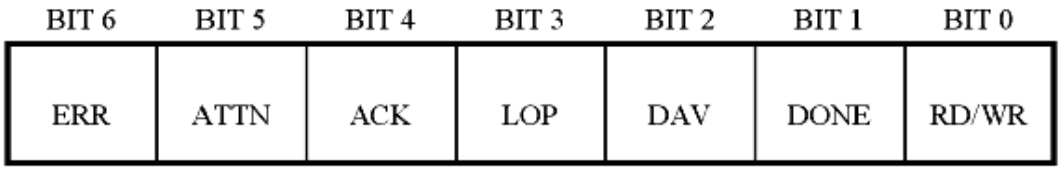


Figure A.5 – The content of the *Microcontroller_control* register.

Figure A.5 shows the data of the *Microcontroller_controls* register. It's the register that contains information about the data exchange between the VME FPGA and the microcontroller: it contains seven bits that all are *Read Only*. RD/WR (Bit0) shows the kind (read or write) of operation performed in the transaction. All the other bits - DONE (bit 1), DAV (bit 2), LOP (bit 3), ACK (bit 4), ATTN (bit 5) and ERR (bit 6) - are used by the communication protocol between the VME CPU and the microcontroller itself. The remaining bits are reserved. The remaining three registers are general purpose registers and are currently reserved.

A.2 The ROD FPGA internal registers

In the ROD FPGA, sixteen 32-bit configuration registers have been implemented. The meanings of each register are now explained. The FIFO_RESET register concerns to the reset pins of the ROD FPGA's six internal FIFOs. Figure A.6 shows the data of the FIFO_RESET register. All bits are active low and are R/W.

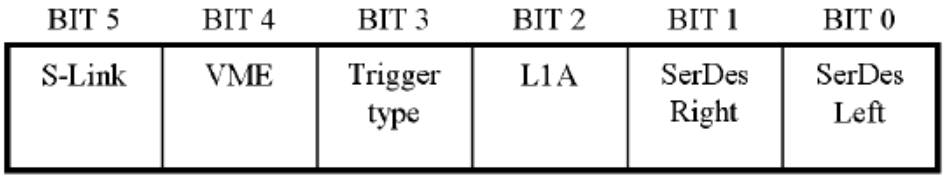


Figure A.6 – The FIFO_RESET register.

The two least significant bits (i.e. bit0 and bit1) are referred to the reset pins of the two FIFOs that receive data from the two *SerDes* (*SerDes* FIFOs); the bit2 and bit3 are referred to the reset pins of the two FIFOs that receive *Level1* data and *TriggerType* data from the TTC receiver (*L1A* FIFO and *Trig_Type*

FIFO); the bit4 and bit5 are referred to the reset pins of the two output "destination FIFOs" written by the *Event Builder* logic: such FIFOs host data to be transferred to the S-Link or to the VME bus (*S-Link* FIFO and *VME* FIFO). The remaining bits are reserved. This register is also useful because the programmable flags' thresholds of a FIFO can be set only when the FIFO reset pin is low. Figure A.7 shows the data of the *FIFO_SerDes_PAF_Threshold* register. All bits are R/W.

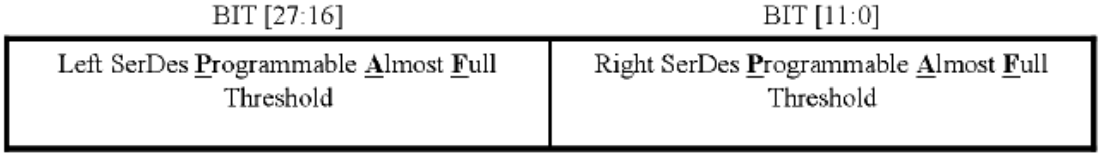


Figure A.7 – The *FIFO_SerDes_PAF_Threshold* register.

This register is used to set the *Programmable Almost Full Threshold* value for the two FIFOs that receive data from the two *SerDes*. It's made of two 12-bit fields, each for every SerDes FIFO, that allow to choose any value between 0 and 4095. The remaining bits are reserved. Figure A.8 shows the data of the *FIFO_TTC_PAF_Threshold* register. All bits are R/W.

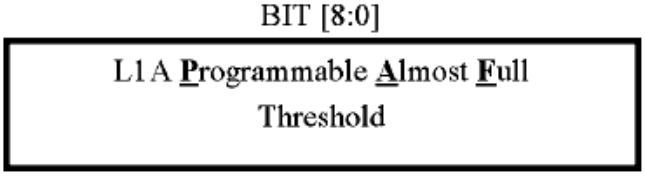


Figure A.8 – The *FIFO_TTC_PAF_Threshold* register.

This register is used to set the *Programmable Almost Full Threshold* value for the two FIFOs that receive *Level1* data and *TriggerType* data from the TTC receiver. It's made of a 9-bit fields, used for both TTC FIFO (*L1A* FIFO and *Trig_Type* FIFO), that allow to choose any value between 0 and 511. The remaining bits are reserved. Figure A.9 shows the data of the *FIFO_VME/SLink_PAF_Threshold* register. All bit are R/W.

BIT [27:16]	BIT [11:0]
S-Link Programmable Almost Full Threshold	VME Programmable Almost Full Threshold

Figure A.9 – The *FIFO_VME/SLink_PAF_Threshold* register.

This register is used to set the *Programmable Almost Full Threshold* value for the two destination FIFOs, written by the *Event Builder Logic*. It's made of two 12-bit fields, each for every FIFO, that allow to choose any value between 0 and 4095. The remaining bits are reserved. Figure A.10 shows the data of the *FIFO_flags* register. All bits are *Read Only*.

BIT [29:25]	BIT [24:20]	BIT [19:15]	BIT [14:10]	BIT [9:5]	BIT [4:0]
Full Almost Full Progr. A. Full Almost Empty Empty *S-Link*	Full Almost Full Progr. A. Full Almost Empty Empty *VME*	Full Almost Full Progr. A. Full Almost Empty Empty *Trigger type*	Full Almost Full Progr. A. Full Almost Empty Empty *L1A*	Full Almost Full Progr. A. Full Almost Empty Empty *SerDes Left*	Full Almost Full Progr. A. Full Almost Empty Empty *SerDes Right*

Figure A.10 – The *FIFO_flags* register.

This register is used to gather the flags of all the ROD FPGA internal FIFOs. Five bit for every FIFO are displayed, for a total of 30 bit. The remaining two bits are reserved. Figure A.11 shows the content of the *Clock_config* register.

BIT [19:16]	BIT [15:12]	BIT [11:8]		BIT3	BIT 2	BIT 1	BIT 0
TTC Clock Activity	SerDes Right Clock Activity	SerDes Left Clock Activity		DCM Clock locked	TTC BUFGMux selection	SerDes Right BUFGMux selection	SerDes Left BUFGMux selection

Figure A.11 – The *Clock_config* register.

The three least significant bits of the register allow the selection of the clock sources for the different clock domains in the ROD FPGA. The bit3 is a *Read Only* bit that allows to monitor the *lock* pin of the *Digital Clock Manager* of the FPGA. Moreover, there are three 4-bit fields *bit[11:8]*, *bit[15:12]* and *bit[19:16]*: each field is *Read Only* and is useful to monitor the activity of the selected clock

source in every clock domain of the ROD FPGA. The remaining bits are reserved. Figure A.12 shows the content of the *Event_builder_config* register.

BIT [31:24]	BIT [23:18]	BIT [17:8]	BIT 6	BIT 5	BIT [4:3]	BIT [2:1]	BIT 0
Timeout	Error Number	Max Length	ARM/ STOP	VME FIFO rules	Data Destinations	Data Sources	Event Builder ON

Figure A.12 – The *Event_builder_config* register.

The bit 0 is the *Event_Builder_ON* bit: writing zero in this location starts the *Event_Builder_engine*. When the *Event_Builder_ON* bit is active, some operations (such as the setting of the source *Serdes* FIFO or the destination FIFO) are not possible. The 2-bit field *Data Source* (bit [2:1]) allows the user to set the source *SerDes* FIFO: it cannot be modified if *Event_Builder_ON* bit is 0. The 2-bit field *Data Destination* (bit [4:3]) allows the user to set the source *destination* FIFO and it cannot be modified if *Event_Builder_ON* bit is 0. The possible choice is between the S-Link FIFO only, the VME FIFO only, both FIFOs or no FIFO selected. Table A.1 shows the different configurations of these two fields.

BIT values	DATA Destinations	BIT values	DATA Sources
00	Both VME and SLink FIFO Selected	**00**	Both Sources Selected
01	SLink FIFO Only Selected	**01**	SerDes Left Source Selected
10	VME FIFO Only Selected	**10**	SerDes Right Source Selected
11	No Destination FIFO Selected	**11**	No Source Selected

Table A.1 – The different configurations for the *Data Source* and *Data Destination* fields.

The 1-bit field *VME FIFO rules* (bit [5]) allows the user to select between the two modes of trapping event builder's data in the VME FIFO. This field is read by the trap engine only when the *Data Destination* value is 00. The two possible values of this field refer to the following functionalities: trap the next available L1A value (*VME FIFO rules*=0); trap a specific L1A value, written into the *Event_Trap_config* register (*VME FIFO rules*=1).

The 1-bit field *ARM/STOP* (bit [6]) allows the user to arm or stop the functionality of trapping event builder's data in the VME FIFO. The 10-bit field *Max_length* (bit [17:8]) allows the user to specify a value to define the *Jumbo frame* length. The 6-bit field *N* (bit [23:18]) allows the user to specify the value of consecutive errors before excluding a SerDes FIFO. The 8-bit field *Timeout* (bit [31:24]) allows the user to specify the Timeout value, described in the fourth chapter. Figure A.13 shows the content of the *Event_builder_status* register. All the bits are *Read Only*.

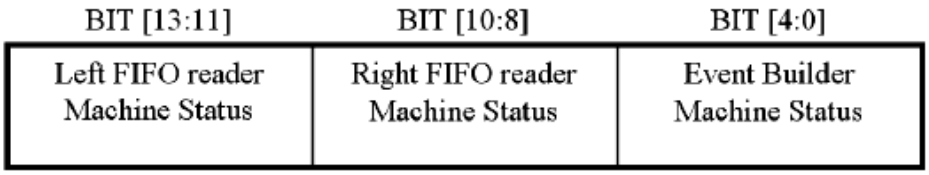


Figure A.13 – The *Event_builder_status* register.

The 5-bit field *Event Builder machine status* (bit [4:0]) allows the user to monitor the internal status of the Event Builder machine, described in the fourth chapter. The 3-bit field *Right fifo reader Machine status* (bit [10:8]) and the 3-bit field *Left fifo reader Machine status* (bit [13:11]) allow the user to monitor the internal status of the two *SerDes* FIFO reader machines, described in the fourth chapter. The remaining bits are reserved. Figure A.14 shows the content of the *Trigger_Type_config* register.

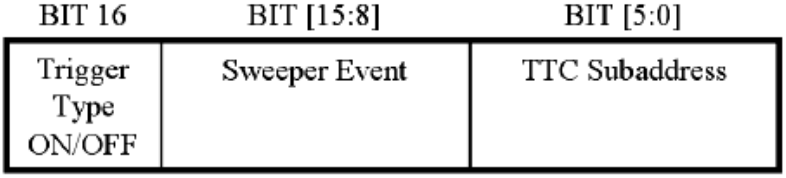


Figure A.14 – The *Trigger_Type_config* register.

The 6-bit field *TTC subaddress* (bit [5:0]) allows the user to set the value of the correct board's subaddress, for the reception of an individually broadcast command from the TTC. The 8-bit field *Sweeper event* (bit [15:8]) allows the user to set the value for the so-called *Sweeper Event*: the default value is 0x07. Every time a *Sweeper event* occurs - and so every time this code is broadcast from the TTC – some of the ATLAS run parameters have been changed. The 1-bit field *Trigger Type ON/OFF* (bit [16]) allows the user to exclude (or activate) to read data from the *Trigger_Type* FIFO, during the *Event_Building* procedure. The remaining bits are reserved. Figure A.15 shows the content of the *Run_Number_config* register.

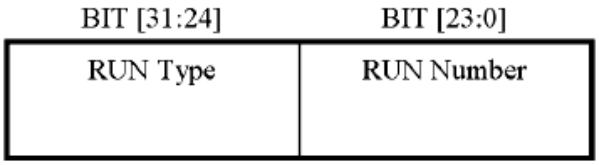


Figure A.15 – The *Run_Number_config* register.

The 24-bit field *Run_Number* (bit [23:0]) allows the user to set the value of the *Run Number*, used to build the ROD Muon Frame. The *Run Number* field is incremented when the *Sweeper Event* is received. The 8-bit field *Run_Type* (bit [31:24]) allows the user to set the value of the *Run Type*, used to build the ROD Muon Frame. Figure A.16 shows the content of the *Slink_Control_Word_config* register.

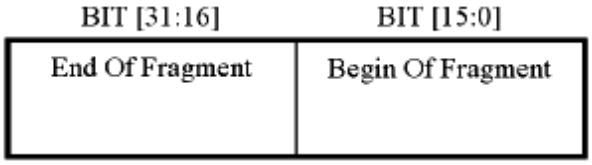


Figure A.16 – The *Slink_Control_Word_config* register.

The 16-bit field *Begin_Of_Fragment* (bit [15:0]) allows the user to set the value of the *Begin_Of_Fragment* field, used to build the ROD Muon Frame. The 16-bit field *End_Of_Fragment* (bit [31:16]) allows the user to set the value of the *End_Of_Fragment* field, used to build the ROD Muon Frame The default values of these fields are respectively: 0xb0f0 and 0xe0f0. Figure A.17 shows the content of the *Busy_config* register. All bits are set to zero at power-up.

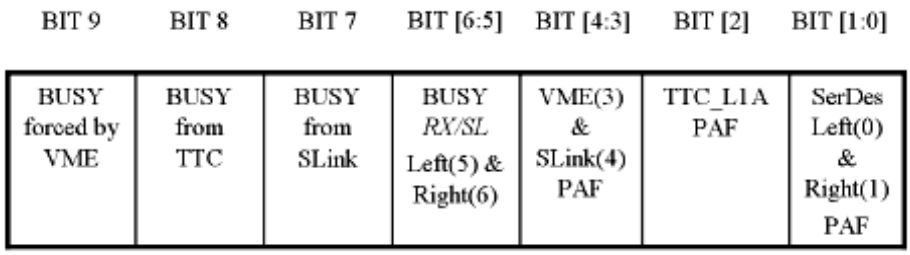


Figure A.17 – The *Busy_config* register.

The 5-bit field *Busy_masks* (bit [4:0]) allows the user to exclude (or not) the internal FIFOs' busy signals from the ROD's global busy. The various bits are referred to *SerDes Left* FIFO (bit[0]), *SerDes Right* FIFO (bit[1]), *TTC_L1A* FIFO (bit[2]), *VME* FIFO (bit[3]), *S-Link* FIFO (bit[4]). The bit from bit[5] to bit[8] allow the user to exclude (or not) the external busy sources from the ROD's global busy. The various bit are referred to *RX/SL Left* busy (bit[5]), *RX/SL Right* busy (bit[6]), *S-Link* busy (bit[7]), *TTC* busy (bit[8]). The bit *Busy_forced_from_VME* (bit[9]) allow the user to force busy via VME. The remaining bits are reserved. Figure A.18 shows the content of the *Detector_event_type* register.

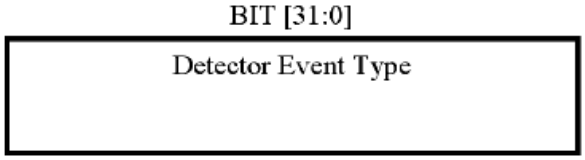


Figure A.18 – The *Detector_event_type* register.

The 32-bit field *Detector_event_type* (bit [31:0]) host user-programmable information for the *Event Builder Machine*, i.e. the user can set the value of the *Detector_event_type*, used to build the ROD Muon Frame. Figure A.19 shows the content of the *Event_counter_reset_ID* register.

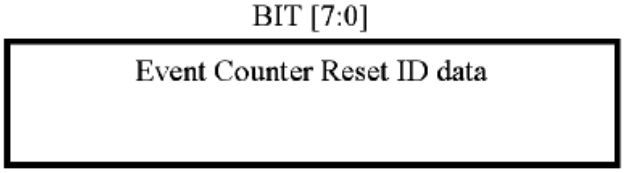


Figure A.19 – The *Event_counter_reset_ID* register.

The 8-bit field *Event_counter_reset_ID* (bit [7:0]) host user-programmable information for the *Event Builder Machine*; in particular, the user can set the value of the *Event_counter_reset_ID*, that is used to build the *Extended_L1ID* field in the ROD Muon Frame. The remaining bits (bit [31:8]) are reserved. Figure A.20 shows the content of the *Event_Trap* register.

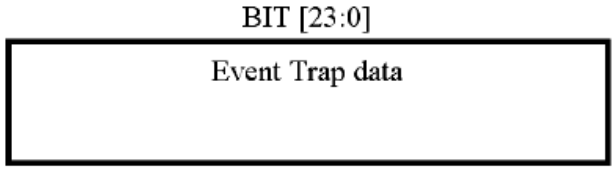


Figure A.20 – The *Event_Trap* register.

The 24-bit field *Event_Trap_data* (bit [23:0]) allows the user to set the value of the *Event_Trap_data*, that is used to identify the event to be trapped into the *VME* FIFO during the *Event_Building* procedure. The remaining bits (bit [31:24]) are reserved. Figure A.21 shows the content of the *Actual_L1A* register. The register is *Read_Only*.

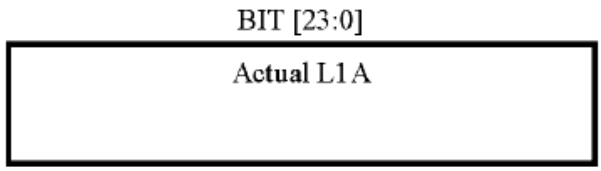


Figure A.21 – The *Actual_L1A* register.

The 24-bit field *Actual_L1A_data* (bit [23:0]) allows the user to monitor the current L1A data, read from the L1A_FIFO by the *Event_Builder Machine*. The remaining bits (bit [31:24]) are reserved.

References

[1] – *The LEP Electroweak Working Group*
URL: **http://lepewwg.web.cern.ch/LEPEWWG/**

[2] – *Particle Data Group, GAUGE AND HIGGS BOSONS*
URL: **http://pdglive.lbl.gov/Rsummary.brl?nodein=S044&fsizein=1**

[3] – *The Large Hadron Collider Accelerator Project*, LHC CERN-AC-93-03.

[4] – *The Large Hadron Collider conceptual Design*, LHC CERN-AC-95-05.

[5] – ***The Large Hadron Collider*, LHC** CERN-AC-95-02.

[6] – *LHC Design Report Volume I*.
URL: **https://ab-div.web.cern.ch/abdiv/Publications/LHC DesignReport.html**

[7] – ATLAS Collaboration, *ATLAS Magnet System Technical Design Report*, CERN/LHCC/97-18, ATLAS TDR 6 (1997).

[8] – *ATLAS Liquid Argon Calorimeter Technical Design Report*, CERN/LHC/96-41, ATLAS TDR 2.

[9] – *ATLAS Tile Calorimeter Technical Report*, CERN/LHCC/96-42, ATLAS TDR 3.

[10] – *Muon Spectrometer Technical Design Report*, CERN-LHCC/97-22.

[11] – G. Viehhauser, *Detector Physics of ATLAS Precision Muon Chamber*, PhD Thesis, CERN.

[12] – H. Groenstege, *The RASNIK/CCD 3D Alignment System*, ATLAS Note MUON-NO-155.

[13] – V. Polichronakos, *A Proposal to Use Cathode Strips Chambers (CSC) for the ATLAS Forward Muon System*, ATLAS-MUON-94-038.

[14] – Y. Ari et al., *Thin Gap Chamber: Performance as a Time and Position Measuring Detector*, Scientifica ACTA: Resistive Plate Chambers and related Detectors; Vol. XI, n.1 pagg. 349-358, Università degli Studi di Pavia.

[15] – E. Duchovni et al., *Possible Utilization of Thin Gap chambers in the ATLAS Muon System*, ATL-MUON-93-023.

[16] – R.Santonico, *Topics in Resistive Plate Chambers*, Scientifica ACTA: Resistive Plate Chambers and related Detectors; Vol. XI, n.1, Università degli Studi di Pavia.

[17] – R. Santonico and R. Cardarelli, *Development of Resistive Plate Counters*, Nucl. Inst. Meth. A 187 (1981).

[18] – R. Santonico and R. Cardarelli, *Progress in Resistive Plate Chambers*, Nucl. Inst. Meth. A 263 (1988).

[19] – R. Santonico, *RPC: Status and Perspectives*, Proceedings of "RPC93, II International Workshop on Resistive Plate Chambers and related detectors" Scientifica ACTA, Vol. VIII.

[20] – ATLAS Collaboration, *ATLAS First Level Trigger Technical Design Report*, CERN/LHCC/98-14.

[21] – ATLAS Collaboration, *ATLAS High-level Triggers, DAQ and DCS Technical Design Report*, CERN/LHCC/2000-17.

[22] – ATLAS Collaboration, *The ATLAS Data Acquisition and High-Level Trigger: Concept, Design and Status*, ATL-DAQ-CONF-2006-016.

[23] – ATLAS Collaboration, *Level-1 trigger: Technical Design Report*.
URL: http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/TDR/V1REV1/L1TDR_intro.pdf

[24] – B.G. Taylor, *Timing Distribution at the LHC*, Presented at the 8th Workshop on Electronics for LHC Experiments, Colmar, Sept. 2002.
URL: http://ttc.web.cern.ch/TTC/intro.html

[25] – J. Varela, *Timing and Synchronization in the LHC Experiments*
URL: http://cdsweb.cern.ch/record/478248/files/p77.pdf

[26] – B.G. Taylor, *TTC distribution for LHC detectors*, IEEE Trans. Nuclear Science, Vol.45, 1998, pp. 821-828

[27] – V. Bocci et al., *The Coincidence Matrix ASIC of the Level-1 Muon Barrel trigger of the ATLAS Experiment*, IEEE Trans. Nuclear Science, Vol.50, n.4, 2003
URL: http://sunset.roma1.infn.it/muonl1/docs/publications

[28] – A. Aloisio, L. Capasso, F. Cevenini, M. Della Pietra and V. Izzo, *The Read-out Driver for the RPC of the ATLAS Muon Spectrometer*, Real-Time Conference, 2007 15th IEEE-NPSS
D.O.I.: 10.1109/RTC.2007.4382754

[29] – C. Bee et al., *The raw event format in the ATLAS Trigger & DAQ*, ATLDAQ-98-129
URL: **http://atlas.web.cern.ch/Atlas/GROUPS/DATABASE/project/event/TDAQ-event-format-0019v24.pdf**

[30] – Philips Semiconductors, *The I2C-bus specification*
URL: http://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus.html

[31] – URL: **http://direct.xilinx.com/bvdocs/publications/ds031.pdf**

[32] – URL: **http://www.xilinx.com/support/documentation/ip_documentation/dcm_module.pdf**

[33] – *DS90CR483/ DS90CR484 48-bit LVDS Channel Link SER/DES*.
URL: http://cache.national.com/ds/DS/DS90CR483.pdf

[34] – Texas Instruments, *LVDS Application and Data Handbook*
URL: http://focus.ti.com/lit/ug/slld009/slld009.pdf

[35] – J. Christiansen, A. Marchioro, P. Moreira* and T. Toifl, *TTCrx Reference Manual, A Timing, Trigger and Control Receiver ASIC for LHC Detectors*
URL: http://ttc.web.cern.ch/TTC/TTCrx_manual3.9.pdf

[36] – *Definition of the trigger-type word*, ATL-DA-ES-0022.

[37] – *The S-Link Interface Specification*
URL: **http://www.cern.ch/HSI/s-link/**

[38] – NXP, *LPC2131/32/34/36/38 Data Sheet*
URL: http://www.nxp.com/acrobat_download2/expired_datasheets/LPC2131_32_34_36_38_3.pdf

[39] – Xilinx, *MicroBlaze Processor Reference Guide*, UG081 v9.0.
URL: http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf

[40] – Xilinx, *Xilinx ISE 9.1i Software Manuals and Help - PDF Collection*.
URL: http://www.xilinx.com/itp/xilinx9/books/manuals.pdf

[41] – Xilinx, *Embedded System Tools Reference Manual*, UG111 v7.0.
URL: http://www.xilinx.com/support/documentation/sw_manuals/edk91i_est_rm.pdf

[42] – Xilinx, *OPB Usage in Xilinx FPGAs*
URL: http://ece427web.groups.et.byu.net/dokuwiki/lib/exe/fetch.php?id=documentation&cache=cache&media=opb_usage.pdf

[43] – Xilinx, *OPB Central DMA Controller (v1.00c)*, Product Specification.

[44] – Xilinx, *Interrupt Control (v1.00a)*, Product Specification.