University of Pisa

DEPARTMENT OF INFORMATION ENGINEERING
Master's Degree in Electronic Engineering

Analysis, design and test of an HW/SW electronic system for data acquisition and control of the Mu2e Femilab Experiment

Candidate:
Micol Rigatti
Matricola 502192

Thesis advisor:
Prof. Sergio Saponara
Prof. Simone Donati

Research supervisors:
Franco Spinella
Elena Pedreschi

Academic Year 2019-2020

Alla mia famiglia,
Luciana, Adriano ed Elia.

Ringraziamenti

Desidero innanzitutto ringraziare il Professor Sergio Saponara nella fiducia accordatami in questo progetto per me così importante.

Ringrazio il Professor Simone Donati per la sua disponibilità e l'attenzione dedicatami nel corso della stesura di questo lavoro, ma soprattutto per i preziosi consigli.

Vorrei poi esprimere la mia sincera gratitudine sia all'Istituto Nazionale di Fisica Nucleare sia al Fermi National Accelerator Laboratory per la preziosa opportunità di studio che mi hanno offerto. Nello specifico ringrazio calorosamente il Dott. Franco Spinella, l'Ing. Elena Pedreschi e il Dott. Luca Morescalchi con cui ho avuto il piacere di collaborare e imparare ben oltre il lavoro di tesi.

Il ringraziamento più grande va ai miei genitori che mi hanno insegnato a non avere paura. Sapere che mi sarete sempre accanto è la luce che mi guida in ogni scelta. Grazie per aver sempre visto in me quello che io spesso non sono stata capace di vedere.

Grazie agli amici dell'Università. Ho condiviso con voi il peso delle lezioni, i troppi caffè del Polo B, i viaggi assonati in treno. Grazie a Giulio, Luca, Jacopo e Simone, il mio Special Team. Se ho una presunzione nella vita, è di essermi circondata delle persone più brillanti che ho incontrato. Senza di voi questi anni non avrebbero avuto lo stesso sapore.

Grazie agli amici di sempre: i nostri momenti più veri saranno il ricordo più vivido che conserverò di questi anni. Con voi ho condiviso ogni attimo. Grazie a Francesca, molto di più che l'amica migliore che chiunque possa desiderare: confidente, complice, motivatrice.

Una dedica speciale va a Francesca, Matteo e Francesco per aver reso l'America il periodo più bello. Siete stati l'inaspettato, l'ispirazione e la sfida a crescere, e siete stati soprattutto il divertimento.

L'ultimo ringraziamento va al Professor Saletti per la passione che mi ha trasmesso. E' il bene più prezioso che sento di aver acquisito in questi anni e so che non è qualcosa che si impara sui libri.

Infine so che chi non è stato esplicitamente citato saprà trovare il suo nome anche se non c'è scritto. Ti ringrazio per esserci stato: non sarei chi sono oggi senza di te.

Abstract

The purpose of the Mu2e experiment at Fermi National Accelerator Laboratory (Fermilab) is the research for the neutrino-less coherent conversion of the muon into an electron, in the field of an aluminum nucleus. The observation of this physics process would unambiguously demonstrate the existence of physics beyond the Standard Model. Although in the past there's always been a huge amount of experimental research activity, all the previous researches for this process have given null results. The experimental technique employed by Mu2e experiment has been designed to improve the sensitivity by four orders of magnitude with respect to similar experiments.

Mu2e is a complex experimental apparatus composed by a high intensity pulsed muon beamline and several independent particle detectors; including a straw-tracker and a crystal-based electromagnetic calorimeter. The calorimeter has been designed and will be built by the collaboration among the Italian National Institute of Nuclear Physics (INFN), the California Institute of Technology (Caltech) and the Fermi National Accelerator Laboratory (FNAL or Fermilab).

The Mu2e Tracker will precisely measure momentum of charged particles that traverse it. This is critical to distinguish the well-known momentum of the signal electrons from background particles that have different momenta. The momentum measurement can be made because a charged particle will trace a helical path through the uniform magnetic field of the Detector Solenoid and the radius of this helix is directly proportional to its momentum. The Mu2e calorimeter is vitally important in reducing backgrounds. Its primary purpose is to provide a set of measurements that complement the information from the Tracker and enable us to reject backgrounds due to reconstruction errors and cosmic ray interactions not vetoed by the cosmic ray veto. For real tracks, activity in the Tracker and in the calorimeter will be correlated in time. The combination of these two timing measurements provides a time-of-flight system that could be capable of providing particle identification information.

Both Tracker and calorimeter are challenging detector, operating in a hostile environment of 1 T magnetic field, a harsh radiation level and $10^{-}4$ Torr vacuum. Moreover, the detector will be installed inside an evacuated cryostat and will be accessible for maintenance only for an extremely limited number of weeks per year.

Operation in vacuum has an important impact on the detector design: all the components have to be vacuum-compatible and a dedicated cooling system is necessary to maintain electronic components within a range of temperature suited for safe long-term operation. Electronics needs to take into account also the expected high radiation levels, using radiation-hard components.

The goal of this thesis is the analysis, the design and the implementation of

the detector electronic responsible for the experiment data acquisition. The Mu2e Trigger and Data Acquisition (TDAQ) subsystem provides necessary components for the collection of digitized data from the Tracker and the Calorimeter, and delivery of that data to online and offline processing for analysis. It is also responsible for detector synchronization, control, monitoring, and operator interfaces. It provides a timing and control network for precise synchronization and control of the data sources and readout, along with a Detector Control System (DCS) for operational control and monitoring of all Mu2e subsystems. It is based on a "streaming" readout: his means that Tracker and Calorimeter detector data is digitized, zero-suppressed in front-end electronics, and then transmitted off the detector to the TDAQ system. The Mu2e TDAQ architecture is further simplified by the integration of all off-detector components in a "TDAQ Server" which functions as a centralized controller, data collector and data processor.

Most of the work done on this thesis is about firmware development on the boards responsible for data collection and control operation: the DRAC for the Tracker and the DIRAC for the Calorimeter. Three was the main goals: first create a functioning firmware for the DRAC at Fermilab, where the environment is set. At Fermilab is available a Test Stand with all the main parts that will compose the experiment. Second, reproduce the Fermilab Test Stand at INFN in Pisa to have an environment to test the DIRAC. The last one, validate DIRAC hardware with a modified version of the DRAC firmware. An additional task has been the reproduction of DIRAC hardware validation at the Test Stand at INFN, reaching the same developing point both in Fermilab and at INFN. This thesis set the groundwork to the future development of the DIRAC firmware.

Both DRAC, the Tracker digitizer and readout controller board, and DIRAC, the Calorimeter readout controller board, mount a Microsemi PolarFire FPGA MPF300TS-1FG1152. Microsemi offers the Microsemi Libero System-on-Chip (SoC) design suite comprehensive of development tools for designing in VHDL. The suite integrates industry standard Synopsys Synplify Pro synthesis and Mentor Graphics ModelSim simulation with constraints management, debug capabilities, and secure production programming support. Challenging step about the firmware development has been the data storage, the management of data flow, the optical interface and the board synchronization. Main purpose of both DRAC and DIRAC is store on a DDR3 memory data retrieved from the event observed by the detectors and send that data to the TDAQ. The two boards interface the TDAQ via a two 2.5 Gbps fibers connected to a Data Transfer Controller, so, part of the development was about the DDR3 control and part was about optical link management via a transceiver structure. The most demanding of this work has been the implementation of communication between the Read Out Controller and the TDAQ and the synchronization of the boards.

Structure of this document goes from general to specific. First is presented the experiment, goals and facility, then the TDAQ, environment within which is developed the work of this thesis. Two chapter are dedicated to the development boards, DRAC and DIRAC: characterization and work done for each. Last chapter is the most specific about the firmware analysis and development. Last last part of this thesis takes into account of future development.

Contents

Ringraziamenti **iii**

Abstract **v**

1 The Mu2e Experiment **1**
1.1 The Standard Model . . . 1
1.1.1 Charged Lepton Flavor Violation (CLFV) . . . 3
1.2 The Experiment . . . 4
1.3 The Fermilab Accelerator Complex . . . 5
1.3.1 The Accelerators Chain . . . 5
1.4 The Mu2e Experimental Facility . . . 6
1.4.1 Production Solenoid . . . 6
1.4.2 Transport Solenoid . . . 7
1.4.3 Detector solenoid . . . 8
1.4.4 The Tracker and the Electromagnetic Calorimeter . . . 8
1.4.5 Cosmic Ray Veto . . . 10
1.4.6 Trigger and Data Acquisition System . . . 11

2 The Mu2e Detectors **13**
2.1 The Tracker . . . 13
2.1.1 Mechanical Construction . . . 14
2.1.2 Front End Electronics . . . 16
2.1.3 Readout Controller . . . 18
2.2 The Calorimeter . . . 19
2.2.1 Mechanical Construction . . . 20
2.2.2 Calorimeter electronics . . . 23

3 TDAQ – Trigger and Data Acquisiton **27**
3.1 Requirements . . . 27
3.2 Architecture . . . 29
3.2.1 Readout Controllers . . . 31
3.2.2 Data Transfer Controller . . . 32
3.2.3 DTC/ROC Interface . . . 34
3.2.4 Run Control Host . . . 36
3.2.5 CFO - Command Fan-Out . . . 36
3.2.6 Event Building . . . 38
3.2.7 System Parameters . . . 39
3.2.8 Detector Control System (DCS) . . . 39
3.3 Timing System . . . 41

3.3.1 Timestamps . . . 45
3.4 TDAQ Software: *artdaq* . . . 45
3.4.1 *otsdaq* . . . 48

4 DRAC - Digital Readout Assembler & Controller 53
4.1 Data Transfer between the Digitizer and ROC . . . 54
4.1.1 Packet Definition . . . 57
4.1.2 Data Rate . . . 58
4.2 Clock . . . 58
4.3 DDR3 Memory . . . 60
4.3.1 Tracker Hit Data . . . 60
4.3.2 Memory Protocol . . . 62
4.4 Microprocessor . . . 65

5 DIRAC - DIgitizer ReAdout Controller 67
5.1 The Digitization system . . . 68
5.2 Test Stand . . . 70

6 SERDES 75
6.1 XCVR - Optical Links Management . . . 77
6.1.1 8b/10b Encoding . . . 78
6.1.2 Implementation . . . 79
6.1.3 Clock Distribution . . . 82
6.2 Communication Protocol . . . 84
6.2.1 Packet Protocol . . . 84
6.2.2 Firmware Structure . . . 90
6.2.3 Packet Managing . . . 97
6.3 ROCs Synchronization . . . 106
6.3.1 Loopback . . . 106
6.3.2 Timestamping . . . 108

7 Conclusions and future prospects 111

8 Appendix 115
8.1 Libero SoC . . . 115
8.2 Vivado . . . 121

Bibliography 123

Chapter 1

The Mu2e Experiment

This Chapter reports the physics motivations and the experimental techniques employed by the muon-to-electron-conversion experiment (*Mu2e*) at *Fermi National Accelerator Laboratory* (Fermilab) [4]. It also reports a brief overview of the Fermilab accelerator complex that provides the high intensity muon beam line and the Mu2e detectors. The Italian National Institute of Nuclear Physics (INFN), the California Institute of Technology (Caltech) and Fermilab are responsible of the design, construction and commissioning of the detector named *electromagnetic calorimeter*. At the moment of writing this Thesis, the calorimeter design has been almost completed, many parts have already been purchased and construction is beginning. Detector assembly will be completed within the year 2021 in time for Mu2e data taking planned for the following year.

1.1 The Standard Model

The theory named Standard Model of the Particle Physics provides a satisfactory model to explain the phenomenology of three among the four known fundamental forces[1]. It describes the interactions between the known elementary particles and how they are mediated by a "relative exchange" of particles (figure 1.1). Attempts to embed the gravitational force in this model have been carried out without satisfactory results[2]. Although the Standard Model predictions have been experimentally verified with high precision, we know this theory is incomplete and needs an extension to incorporate phenomena such as *neutrino oscillations*[3] and *dark matter*[4], both observed experimentally.

We basically know two different categories of elementary particles: *fermions*, the 1/2 spin[5] elementary constituents of matter, and *bosons*, the integer spin

[1]The four fundamental forces are: the electromagnetic, the weak, the strong and the gravitational force. The gravitational force is the only one not included in the standard model

[2]The theory able to link together general relativity and quantum mechanics would be the so called "Theory of everything", that would fully explain and link together all the physical aspects of the Universe

[3]Neutrino oscillation is a quantum mechanical phenomenon whereby a neutrino created with a specific lepton family number (lepton flavour: electron, muon, or tau) can later be measured having a different lepton family number.

[4]Dark matter is a form of matter thought to account for approximately 85% of the matter in the universe, and about a quarter of its total energy density. Its presence is implied in a variety of astrophysical observations, including gravitational effects which cannot be explained by accepted theories of gravity unless more matter is present than the visible one.

[5]In quantum mechanics the spin is an intrinsic form of angular momentum carried by

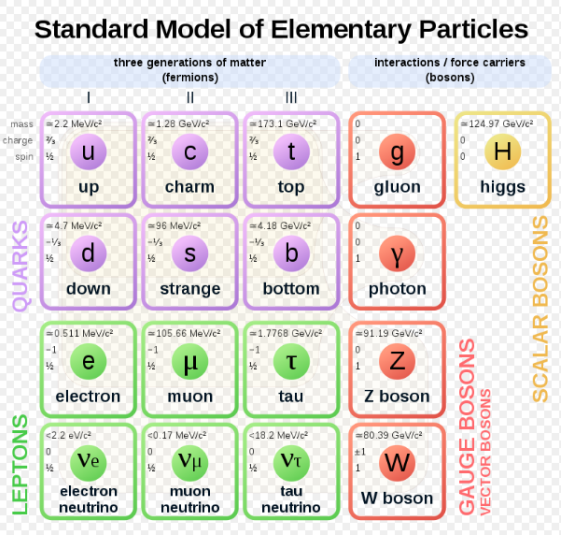


Figure 1.1: Summary table of the elementary constituents of matter, quarks, leptons and gauge bosons (image courtesy of Fehling, Dave. The Standard Model of Particle Physics: A Lunchbox's Guide. The Johns Hopkins University)

mediators of the fundamental forces.

Fermions are classified according to their interactions. There are six quarks (up, down, charm, strange, top, bottom), and six leptons (electron, electron neutrino, muon, muon neutrino, tau, tau neutrino). Pairs from each classification are grouped together to form a *generation*, with corresponding particles exhibiting similar physical behaviour. The defining property of quarks is that they carry color charge, and hence interact also via the strong interaction. A phenomenon called color confinement results in quarks being very strongly bound to one another, forming color-neutral composite particles (hadrons) containing either a quark and an antiquark (mesons) or three quarks (baryons). The familiar proton and neutron are the two baryons having the smallest masses. Quarks also carry electric charge and weak isospin. Hence they interact with other fermions both with electromagnetic and weak interactions. The remaining six fermions do not carry color charge and are called leptons. The electron, muon and tau leptons have electric and weak charge and show both electromagnetic and weak interactions. On the other hand, neutrinos do not have electric charge and they only interact weakly.

The the first generation includes the electronic leptons:

- The electron e;
- The electron neutrino ν_e;

the second one the muonic leptons:

- The muon μ;
- The muon neutrino ν_μ;

and the third one the tauonic leptons:

elementary particles, composite particles (hadrons), and atomic nuclei.

- The tau τ;
- The tau neutrino ν_τ.

Each member of a generation has greater mass than particles belonging to lower generations. Ordinary matter (i.e. atoms, neutrons and protons) is made of particles belonging to the first generation. Specifically, all atoms consist of electrons orbiting around atomic nuclei, ultimately constituted of up and down quarks. The second and third generation charged particles, on the other hand, decay with very short half-lives and are observed only in high-energy environments. Neutrinos pervade the universe, but they do not decay and rarely interact with baryonic matter.

In the Standard Model, bosons are defined as force carriers that mediate the strong, weak, and electromagnetic fundamental interactions. The interactions in physics are the ways in which the particles influence other particles. The Standard Model explains such forces as resulting from elementary particles exchanging other particles, generally referred to as force mediating particles (i.e. the bosons), as listed below:

- The photon γ mediates the electromagnetic force between charged particles. It is massless.
- The W^+,W^- and Z bosons mediate the weak interactions between particles of different flavours. They are massive.
- The *gluon* mediates the strong interaction between color charged particles (i.e. quarks). They are massless and with an effective color charge, hence they can also interact among themselves.
- The *Higgs particle* is a massive scalar elementary particle theorised by Peter Higgs in 1964. It plays a unique role in the Standard Model and provides a mechanism to explain why other elementary particles, except the photon and gluon, are massive. In particular, the Higgs boson explains why the photon has no mass, while the W and Z bosons are very heavy. The Higgs boson was observed at the Large Hadron Collider experiments in the year 2012.

1.1.1 Charged Lepton Flavor Violation (CLFV)

The muon and tau are unstable particles, while the electron is stable. The muon decays with a probability of approximately 100% to a muon neutrino ν_μ, an electron e, and an electron antineutrino $\bar{\nu_e}$[6]. In symbols, the previous decay is written as $\mu \rightarrow \nu_\mu e \bar{\nu_e}$.

In a small fraction of cases, also additional particles with a net charge equal to zero may be produced in the muon decay (e.g. a photon, or an electron-positron[7] pair). In all these processes the lepton flavour is conserved separately for every lepton family. In other words, in the final state there is a muon neutrino, which belongs to the same family and has the same lepton number as the parent muon;

[6] In particle physics, every type of particle has an associated antiparticle with the same mass but with opposite electric charge.

[7] the positron is the antiparticle of electron

and there are an electron and an electron antineutrino that have in total zero lepton number, since particles and antiparticles have opposite lepton numbers.

Extensions of the Standard Model, which is believed to be an incomplete theory since it does not incorporate observed phenomena as neutrino oscillations, predict also the existence of Charged Lepton Flavour Violating (CLFV) processes, which do not conserve the lepton flavours. Examples are the muon to electron conversion ($\mu \rightarrow e\gamma$)[8], which does not have the muon neutrino and the electron antineutrino in the final state, and also the *coherent neutrino-less muon conversion to an electron in the field of a nucleus* ($\mu N \rightarrow eN$). Since CLFV processes are expected within the Standard Model with a probability only of the order of 10^{-54}, but with much higher probabilities in the Standard Model extension, they are an extremely interesting field to search for physics beyond the Standard Model.

Mu2e has been designed and is currently being constructed at Fermilab to search for the neutrino-less muon conversion to an electron in the field of an aluminum nucleus. The current experimental limit on the branching factor of this process has been set to be 10^{-12} by the SINDRUM II experiment, performed at the Paul Scherrer Institut at Zurich. The Mu2e sensitivity will allow to observe muon conversion events if the probability of the process is larger than 10^{-17}. However, if nothing is observed, Mu2e will fix an upper limit to the probability of the process, gaining an improvement of four orders of magnitude over the current experimental limit (SINDRUM II).

The international Mu2e collaboration is completing the design of the various components, and soon will begin to build the experiment. The construction is expected to be completed within the year 2021. The beginning of data taking is planned for the year 2022, and will continue for about three years. Future upgrades of the experimental apparatus planned for the years 2025 and beyond will further improve the experimental Mu2e sensitivity of one order of magnitude.

1.2 The Experiment

The Mu2e experiment at Fermilab will be 10,000 times more sensitive than previous experiments looking for muon-to-electron conversion [5]. This precise and complex experimental apparatus will produce 200 million billion muons per year. The accelerator complex repurposes elements of the infrastructure that produced high energy proton and anti-protons beams for the Tevatron experiments to produce the high-intensity muon beams necessary for Mu2e and the Muon (g-2) experiments.

The Fermilab Booster will accelerate protons to the 8 GeV needed to produce the intense muon beam employed by Mu2e.

The protons will travel from the Booster to the Recycler where they will be stacked, bunched, and extracted to the Delivery ring. The Delivery ring is located in the repurposed Debuncher. Once in the Delivery ring the protons will be slow extracted and delivered to the Mu2e apparatus. A system of three superconducting solenoidal magnets (Production Solenoid, Transport Solenoid and Detector Solenoid) will transport the intense low-energy muon beam to the experimental area where Mu2e is located.

The 8 GeV protons will arrive in bunches from the Delivery ring and enter the Mu2e Production Solenoid at a slight angle to its axis and strike a tungsten

[8] γ is the symbol representing the photon.

production target about the size of a pencil. These collisions will create a cascade of particles, including pions that decay into muons. The magnetic field of the Production Solenoid will capture some of the muons and spiral them into the Transport Solenoid. Only about 1 in 300 protons that collide with the production target will generate a muon that moves into the Transport Solenoid. Throughout the experiment's projected three-year running period, roughly 10 billion muons per second will be stopped.

Muons in the Transport Solenoid will travel inside an evacuated vessel towards the Mu2e aluminum target. The Mu2e detector is the particle physics detector embedded inside the Detector Solenoid that provides a magnetic field in the detector region that allows the momentum of the conversion electrons to be accurately determined. The Mu2e detectors consist of two main parts: the magnetic spectrometer to measure particles momentum, and the electromagnetic calorimeter to measure particles energy and time of impact.

Improvements to the accelerator could extend the initial Mu2e sensitivity by a factor of ten or more. This is comparable to Mu2e initially producing a number of muons equivalent to all the grains of sand on the Earth's beaches. This would provide a valuable tool for physics research whether or not Mu2e discovers muon-to-electron conversion during its first, lower-intensity phase. If Mu2e does observe charged lepton conversion, an upgraded accelerator would enable Mu2e to study in depth the details of the conversion by providing more data. If Mu2e does not observe the conversion, the collaboration could continue the search with a wider net and also search for signs of never-before-seen physics in rare processes that have previously been out of reach of physics machines.

1.3 The Fermilab Accelerator Complex

Fermilab is located in Batavia, about 50 km west of Chicago, Illinois (USA). It is a US Department of Energy Laboratory, operated by the Universities Research Association (URA) since its founding in 1967 to 2006 [4]. Since 2007 Fermilab has been operated by the partnership between the University of Chicago and the University Research Association, named Fermilab Research Alliance (FRA). The name Fermilab was given to the laboratory in 1974 to honour the Italian Nobel prize Enrico Fermi.

Figure 1.2 shows an aerial view of the laboratory, which has played a major role in the field of high energy physics for the last forty years. Among its scientific achievements, it is worth mentioning the discovery of three among the four particles of the model's third generation: the bottom quark (May-June 1977), the top quark (February 1995) and the tau neutrino (July 2000).

1.3.1 The Accelerators Chain

The accelerator [3] complex is composed of several stages.

- The first stage is a *Cockcroft-Walton generator*, which turns hydrogen gas into H^- ions by flowing it into a container lined with molybdenum electrodes (a matchbox sized, oval shaped cathode and a surrounding anode, separated by 1 mm and held in place by glass ceramic insulators). A magnetron[9] is

[9] A high powered vacuum tube that generates microwaves using the interaction of electrons streams with a magnetic field, while moving past a series of open metal cavities (cavity resonators).

Figure 1.2: Aerial view of the Fermilab site. The Mu2e facility is close to the center. Many other experiments are hosted in the site (TEVATRON etc ...)

used to generate a plasma to form H^- ions close to the metal surface. A 750 keV electrostatic field is applied by the Cockcroft- Walton generator, and the ions are accelerated out of the container.

- The second stage is a Linear Accelerator (or *Linac*), which accelerates the ions to the energy of 400 MeV (approximately 70% of the speed of light). Just before entering the next accelerator, the ions pass through a carbon foil, where they lose the electrons producing a H^+ ion beam (called proton beam).
- The third section is the *Booster Ring*. The Booster ring is a 468 m circular accelerator that uses magnets to bend the proton beam in a circular path. The protons coming from the Linac travel around the Booster about 20000 times in 33 ms, in order to multiply the accelerating electric field. Each revolution gives the protons more energy, until the beam leaves the ring at approximately the energy of 8 GeV.
- Finally the protons are injected into the *Recycler Ring*, where they circulate while getting rebounded by a 2.5 MHz frequency system. The reformatted bunches are transported to the delivery ring, where they are slowly extracted from the Mu2e detector through a new external beamline (figure 1.3).

1.4 The Mu2e Experimental Facility

The Mu2e apparatus is extensively documented in the conceptual *Design and Technical Report* [2]. The layout of the muon beam line and the detector system are sketched in figure 1.4.

1.4.1 Production Solenoid

The Mu2e magnet system consists of three large superconducting solenoids. The first one in the chain of magnets is the Production Solenoid (PS), whose role is to collect and focus pions[10] and muons generated in interactions of an 8-GeV proton

[10]In particle physics, a pion is any of the three subatomic particles: π^+, π^-, π^0 (depending of their net charge). Pions are composed of a quark and antiquark pair, and are the lightest mesons. Charged pions most often decay into muons and muon neutrinos, while neutral pions generally decay into gamma rays.

Figure 1.3: Layout of the Mu2e facility (lower right) relative to the accelerator complex that provides the proton beam to the detector. Protons are transported from the Booster through the MI-8 beamline to the Recycler Ring, where they circulate while being rebounded by a 2.5 MHz RF system. The reformatted bunches are kicked into the P1 line and transported to the Delivery Ring, where they are slowly extracted for the Mu2e detectors

beam with a tilted high-Z target, by supplying a peak axial field between 4.6 T and 5.0 T and an axial field gradient of about 1 T/m, within a 1.5 m warm bore.

The PS is a challenging magnet because of the relatively high magnetic field and a harsh radiation environment that requires the state-of-the-art conductor, both in terms of the current-carrying capacity and structural strength. The PS coil is protected by a massive Heat and Radiation Shield (HRS).

1.4.2 Transport Solenoid

The role of the S-shaped Transport Solenoid (TS) is to filter and transport the muon beam (approximately 10^{11} muon/s) to the Detector Solenoid. It is composed of 14 superconducting units (solenoids and toroids) and is divided in five sections:

- a 1 m long straight section
- a 90-degree elbow, with 3 meters radius of curvature
- a second 2 m long straight section
- a second 90-degree elbow, similar to the first, that turns the beam line in a direction parallel to the first one
- a final 1 m long straigth section

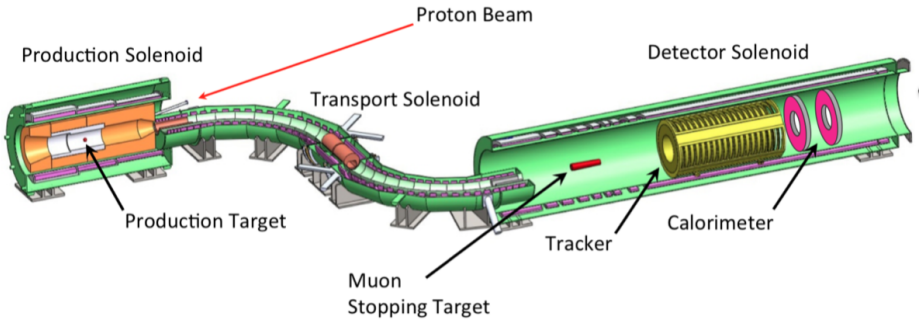


Figure 1.4: Mu2e apparatus: the proton beam enters from the right at the junction between the Production Solenoid and the Transport Solenoid, and strikes the production target. The cosmic ray veto system, which surrounds the Detector Solenoid, and the muon stopping monitor are not shown in this scheme (source: Mu2e experiment data center).

The resulting length of the Transport Solenoid is 13 m. To improve the purity of the muon beam, the Transport Solenoid has an absorber placed in its central part that stops charged particles (mainly antiprotons). A state-of-the-art collimator system is placed in the same zone to select only low energy muons with momentum below 0.08 GeV/c_0. Moreover, the S-shape of the solenoid removes neutral particles that travel in a straight direction.

1.4.3 Detector solenoid

The Detector Solenoid (DS) is a 11 m long component, with a decreasing magnetic field in the first sector (from 2 T to 1 T). It hosts the muon stopping target, schematically represented in figure 1.5.

Muons impacting the disks come to rest and replace the electrons lying in the $1s$ orbit of the aluminum atoms. The lifetime of the muon in the muonic atom is 864 ns. The non uniformity of magnetic field plays an important role in reducing the background coming from high energy electrons transported to the Detector Solenoid. The magnetic field gradient is generated introducing spacers to change the winding density of the superconducting cable, which is made of aluminum-stabilized $NiTi$.

The second sector of the Detector Solenoid houses the detectors: the *Tracker* and the *calorimeter*, which are described in more detail in the next Chapters. In this sector the field is relatively uniform and has intensity of 1 T.

1.4.4 The Tracker and the Electromagnetic Calorimeter

The Mu2e detector is located inside the evacuated warm bore of the Detector Solenoid in a nearly uniform 1 Tesla magnetic field and is designed to efficiently and accurately identify and analyse the helical trajectories of ∼105 MeV electrons in the high-rate time varying environment of Mu2e. The detector consists of a Tracker and an elecgtromagnetic calorimeter that provide redundant energy/momentum, timing, and trajectory measurements. A cosmic ray veto, consisting of both active and passive elements, surrounds the Detector Solenoid and nearly half of the Transport Solenoid and is used to identify possibile background events due to

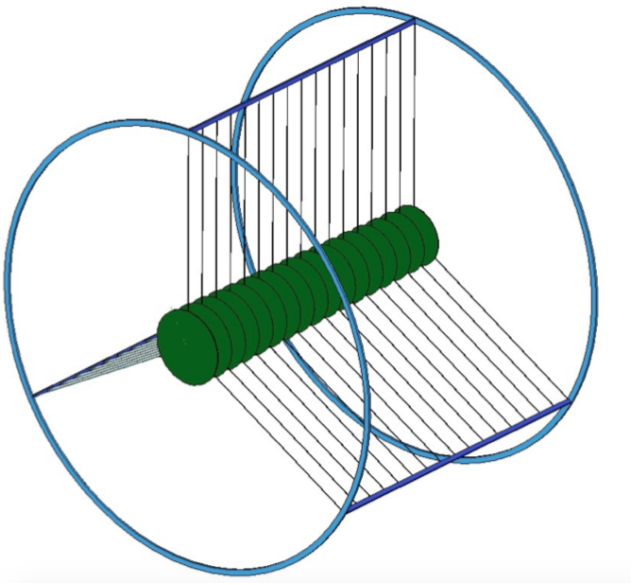


Figure 1.5: The Mu2e stopping target is made of 17 aluminum disks, 0.2 mm thick, spaced 5.0cm apart along the Detector Solenoid axis. The disks radii decrease from 8.3 cm at the upstream end to 6.53 cm at the downstream end (source: Mu2e experiment data center).

cosmic-rays. The Mu2e collaboration decided to use a Tracker design similar to the one developed by the MECO collaboration (figure 1.6).

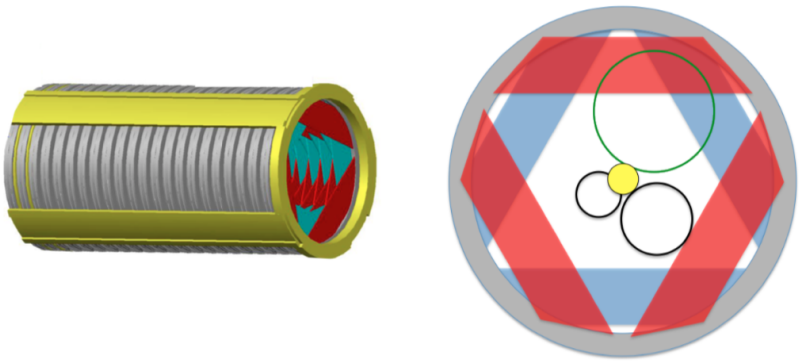
The Mu2e Tracker is designed to accurately reconstruct the helical trajectories of electrons in a uniform 1 Tesla magnetic field and measure their momenta. It operates at $1.33 \cdot 10^{-2}$ Pa vacuum to reduce multiple scattering[11] to a negligible level.

Given that multiple scattering in the Tracker material dominates the resolution on the measurement of the helix parameters, the mechanical structure of the detector has been made extremely light. The Tracker is made of *straw drift* tubes; and is called T-Tracker because the straws are transverse to the axis of the Detector Solenoid. The basic detector element is made of a 20 μm sense wire inside a straw tube filled with gas. The straws are 5 mm diameter tubes, made of 15 μm thick metallic Mylar. The Tracker is made of approximately 2000 straws arranged along 18 stations across the 3 m Tracker length. One Tracker plane consists of two layers of straws to improve the reconstruction efficiency and help to overcome the classic left-right ambiguity. A 1 mm gap between straws allows for manufacturing tolerance and expansion due to the internal pressure. A larger radius ring outside the active detector region supports the straws and the electronics boards.

Each straw has one preamplifier and one time-to-digital converter (TDC) placed on each tip, in order to measure the signal arrival time on both sides. It uses also analog to digital converters (ADC) to measure the total integrated charge, providing useful information for particle identification.

The Tracker has been designed to observe only electrons with energy greater than 53 MeV. Electrons below this threshold travel undetected in the central un-instrumented volume of the Tracker. They are approximately 3% of the

[11] Scattering produced by the presence of residual gas particles



(a) The Mu2e *Tracker*; some of the 18 station of the tracking system are visible on the right.

(b) Cross-sectional wiew of the Mu2e *Tracker*.

Figure 1.6: Mu2e Tracker layout. The picture displays the 18 tracking system stations. Thanks to this design, only the electrons with energies greater than 53 MeV are reconstructed. Lower energy electrons pass through the un-instrumented central part of the device and leave no track. This effect is due to the spiral motion of electrical charges in a uniform magnetic field (source: Mu2e experiment data center)

total electron flux coming from muon decays. Since momentum resolution is a crucial factor to suppress critical backgrounds, the Tracker is required to have a momentum resolution better than 180 keV for 100 MeV electrons.

The Mu2e calorimeter provides additional energy, position, and timing information for particles' trajectories reconstructed by the Tracker. The two detectors use different physical and technological processes to perform their measurements, to rely on uncorrelated error sources. This helps to reduce backgrounds and provides a cross check to verify the quality of signal events.

The calorimeter operates in the same solenoidal magnetic field and vacuum level as the Tracker. It handles a large flux of particles, mostly a low energy background of protons, neutrons and gamma rays produced by muon captures in the stopping target. It also manages a large flux of electrons coming from muons decays in the aluminum stopping target, and other produced particles during the beam injection.

1.4.5 Cosmic Ray Veto

Cosmic ray muons may initiate processes and produce particles that interact with the detectors and produce unwanted backgrounds. The simulation show that approximately one background event generated by cosmic ray muons may be erroneously reconstructed as a conversion electron signal per day. This source of background can be reduced to a negligible level by introducing passive and active shielding.

The *Cosmic Ray Veto* (CRV) surrounds the entire volume occupied by the Detector Solenoid and the downstream part of the Transport Solenoid. It consists of four layers of extruded scintillator strips with silicon photosensors and aluminum absorbers.

The cosmic ray induced background rate will be monitored between beam

spills and when the beam will be turned off. This allows to perform a direct measurement of the background level. The study of the background rate will be initiated as soon as the Detector Solenoid and the cosmic ray veto are in place.

1.4.6 Trigger and Data Acquisition System

The Trigger and Data Acquisition (TDAQ) systems provide hardware and software tools to record the digitised data received from the detectors. It is crucial to collect, organise, filter, build events, and make trigger decisions to validate physics and calibration data for the experiment. Data from the detectors will be processed, digitised and transferred to the TDAQ system. It also combines information from all the detector data sources and applies filters (triggers) to reduce this rate by a factor of several thousands, before the data get delivered to the offline permanent storage.

Signals from the detectors are amplified, digitised and transmitted to Readout Controllers (ROCs), that will be attached to the Mu2e detectors as a subsystem for data transfer. Digitisation is performed in the front end electronics, then, digital data is transmitted to the TDAQ over optical fibre. All communication between the TDAQ software framework and the ROCs go through the Data Transfer Controllers (DTCs).

A more detailed description of the TDAQ is the main purpose of this thesis, and is reported in the following chapters.

Chapter 2

The Mu2e Detectors

The most relevant Mu2e detectors are the straw tracker, the electromagnetic calorimeter and the cosmic ray veto. The Mu2e straw tracker has been designed to perform an accurate measurement of charged particles momentum. This measurement is critical to distinguish signal conversion electrons, that have the momentum equal to the muon rest mass, from background electrons produced in Michel decays, that have a broad momentum spectrum with the muon rest mass as endpoint. Charged particles momentum can be measured because they follow helical trajectories through the uniform magnetic field of the Detector Solenoid and the helix radius is directly proportional to the particle momentum. The tracker has been designed to intercept the helical path in many points and allow an accurate trajectory reconstruction.

The Mu2e calorimeter provides information that complements the particles trajectory reconstruction performed by the tracker and helps to reject backgrounds due to reconstruction errors and cosmic ray interactions undetected by the cosmic ray veto. Although the calorimeter energy resolution is not competitive with the tracker, even a coarse confirmation of the particle energy helps to reject backgrounds due to spurious combinations of hits generated by low energy particles. Moreover, for real particles traversing the entire detector, the activities in the tracker and calorimeter are correlated in time. This timing information provides a time-of-flight measurement extremely useful to reduce the background level in the detector.

2.1 The Tracker

The tracker [2] must accurately and efficiently identify and measure 105 MeV/c electrons while rejecting backgrounds and it must provide this functionality in a relatively unique environment. The tracker resides in the warm bore of a superconducting solenoid providing a uniform magnetic field of 1 Tesla; the bore is evacuated to 10^{-4} Torr. A key feature of Mu2e is the use of a pulsed beam that allows to reject prompt backgrounds by looking only at tracks that arrive several hundred nanoseconds after the proton pulse. The tracker must survive a large flux of particles during the early burst of "beam flash" particles that result from the proton pulse striking the production target, but it does not need to take data during this time. The Mu2e signal window is defined as $700 < t < 1695$ ns, where $t = 0$ is the arrival of the peak of the beam pulse at the stopping target. However, in order to study backgrounds such as radiative pion capture, the tracker must be

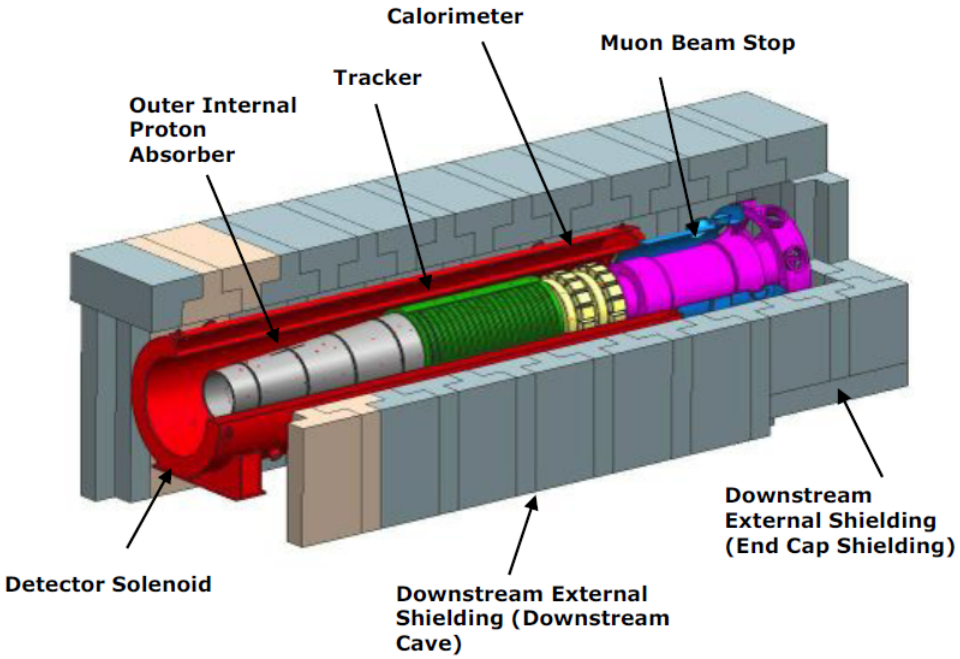


Figure 2.1: Mu2e Detector Solenoid with Tracker and Calorimeter.

fully efficient during the interval $500 < t < 1700$ ns; we take this as the tracker's live window. To calibrate using positrons from $\pi^+ \to \nu e^+$ decays the tracker must also be able to collect data, during special runs with reduced beam intensity, for $300 < t < 1700$ ns.

The Detector Solenoid provides a uniform 1 Tesla field in the region occupied by the tracker. To have good acceptance for signal electrons without being overwhelmed by DIO electrons (including electrons scattered into the active region), the active area of the tracker extends from about $40 < r < 70$ cm (where the radius r is measured from center of the muon beam). Mechanical support, readout electronics, etc. are to be placed at $r > 70$ cm, out of the way of both signal and DIO electrons. These dimensions depend on the size and geometry of the muon stopping target, the size of the muon beam, and the magnetic field properties; they have been optimized to maximize the acceptance to conversion electrons while minimizing the number of low energy electrons that intersect the tracker. The momentum resolution requirement is based on background rejection: the signal is sharply peaked, whereas backgrounds are broad (cosmic rays, radiative pion capture) or steeply falling (DIO electrons). For a Gaussian error distribution the requirement is that $\sigma < 180$ keV/c. This is simply a convenient reference point; the actual resolution is not Gaussian and may be asymmetric. Furthermore, scattering and straggling in material upstream of the tracker are significant contributors to the final resolution.

2.1.1 Mechanical Construction

The selected design for the Mu2e Tracker is a low mass array of straw drift tubes aligned transverse to the axis of the Detector Solenoid, referred to as the T-Tracker. The basic detector element is a 25 μm sense wire inside a 5 mm diameter tube made of 15 μm thick metallized Mylar, referred to as a straw. This choice is based

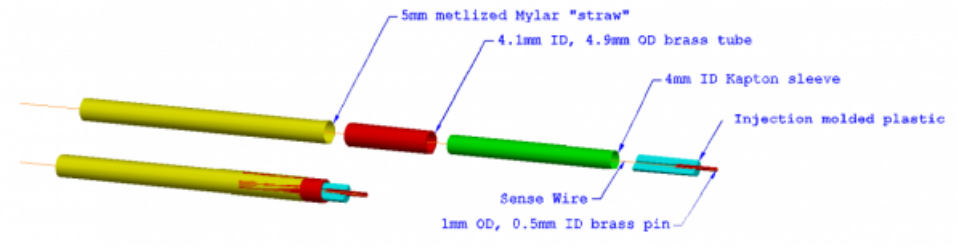


Figure 2.2: Straw termination, shown exploded and assembled. The brass tube connects to the straw with silver epoxy. The green insulator slips inside a brass tube (red) to prevent breakdown near the tube end. The sense wire is soldered into the brass pin, and epoxied to the injection molded plastic. After assembly the brass tube allows connection to the cathode while the brass pin allows connection to the anode.

on several points.

- The straw can go from zero to 1 atmosphere pressure differential (for operating in a vacuum) without significant change in performance.
- Unlike other types of drift chambers, each sense wire is mechanically contained within a straw. Thus, failures remain isolated and improve reliability.
- The transverse design naturally places mechanical support, readout electronics, cooling, and gas distribution at large radii.

The detector has $\sim$ 23,000 straws distributed into 20 measurement stations across a $\sim$ 3 m length. Each station provides a $\sim$ 200 μm measurements of track position.

Each straw is instrumented on both sides with preamps and TDCs [31]. Each straw has one ADC for dE/dx capability. The planned digitizer system is 50 MHz, 12-bit ADC. To minimize penetrations into the vacuum, digitization is done at the detector with readout via optical fibers. Electronics at the detector will not require an external trigger: all data will be transferred out of the vacuum to the TDAQ system, and a trigger may be implemented as part of the TDAQ.

The T-Tracker is made of 5 mm diameter straws, the assembly of which is shown in and figure 2.2. Each straw is made of two layers of $\sim$ 6 μm (25 gauge) Mylar, spiral wound, with a $\sim$ 3 μm layer of adhesive between layers, for a total wall thickness of 15 μm. The inner surface has 500 $\AA$ aluminum overlaid with 200 $\AA$ gold as the cathode layer. The outer surface has 500 $\AA$ of aluminum to act as additional electrostatic shielding and reduce the leak rate.

A 4.95 mm outer diameter brass tube is mechanically and electrically connected to each straw end using silver epoxy. An extruded kapton tube is placed inside the brass tube to protect against breakdown at the edge of the brass tube. Also an injection molded plastic insert is placed inside the kapton tube. Attached to a groove in the insert a small, U-shaped brass pin is placed. A 25 μm gold plated tungsten wire is soldered to the pin as well as epoxied to the plastic insert. Both brass parts are gold-plated to ensure good solder and epoxy joints.

Groups of 96 straws are assembled into panels as shown in figure 2.3. Each panel covers a 120° arc with two layers of straws. The double layer improves efficiency and helps determine on which side of the sense wire a track passes. A

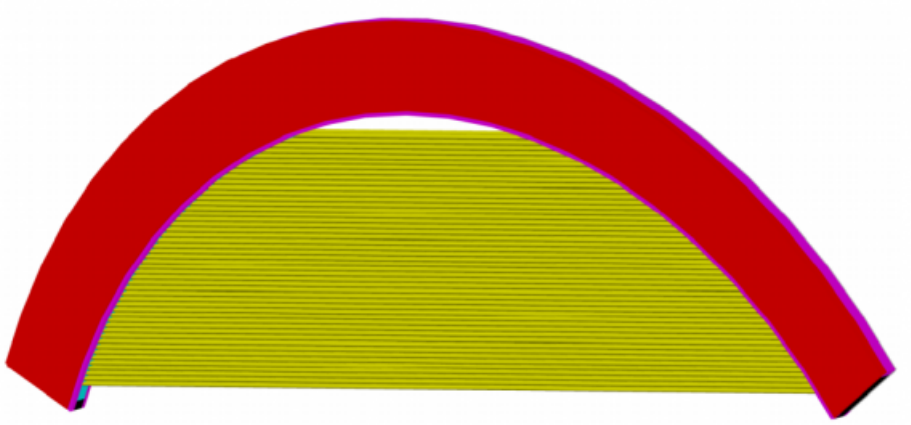


Figure 2.3: Design of one completed panel, with covers shown in red. The screws necessary to attach covers not shown.

1.25 mm gap is maintained between straws to allow for manufacturing tolerance and expansion due to gas pressure. This necessitates that individual straws be self-supporting across their span.

Six panels are assembled into a plane as shown in figure 2.4. Three 120° panels complete the ring of one face; another three panels, rotated by 30°, complete another ring on the opposing face. After the plane is assembled, a cooling ring is attached around the outer diameter. This arrangement has been found to give the best stereo performance and has been chosen despite the mechanical complications compared to the 60° rotation.

Eighteen stations are assembled into the complete tracker, shown in figure 2.5. Horizontal beams maintain longitudinal alignment of the rings. The thicker ring seen at the upstream end, and the two thinner rings placed between stations at the downstream end, stiffen the structure. Stiffening rings and beams are stainless steel, pending further analysis and value engineering.

2.1.2 Front End Electronics

To minimize penetrations through the cryostat, digitizers and zero-suppression logic are located directly on the detector [29]. Signals from the straws need to be amplified, digitized and trasmitted to the TDAQ. Front end electronics is defined as all electronics residing on the detector. Digitization is performed in the front end, then, digital data is transmitted to the TDAQ over optical fibre. The bulk of the electronics is dedicated to amplifying, digitizing, and transmitting signals from the $\sim$ 25 K straws, but in addition there are sensors for monitoring detector parameters; data from these must also be transmitted to the TDAQ system. Each straw is read out from both ends.

The front end electronics is divided in the following categories:

- Infrastructure: includes power and cooling.
- Preamplifierss: there is one at the end of each straw to amplify and send an analog signal via a PCB transmission line to digitizers.
- Digitizers: the digitizer receives and digitizes the signals from both ends of each straw and transmits that data to the Readout Controllers. The

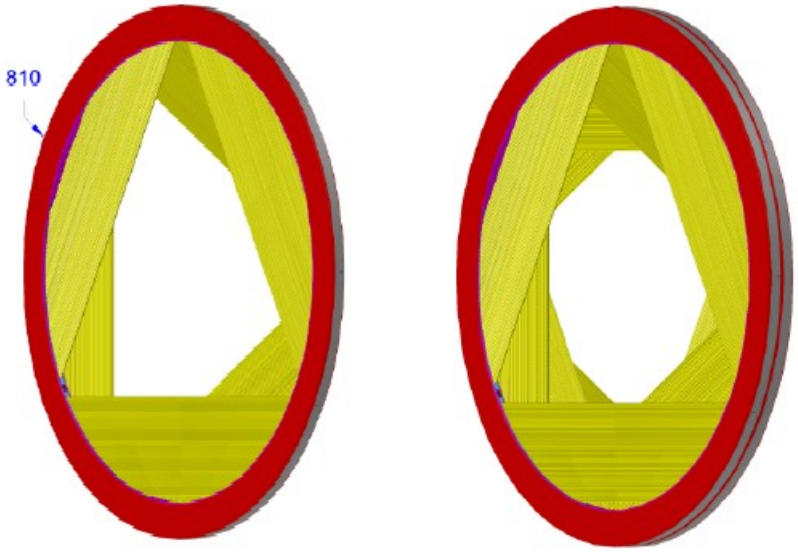


Figure 2.4: Design of one complete panel, with covers shown in red. The screws necessary to attach covers not shown.

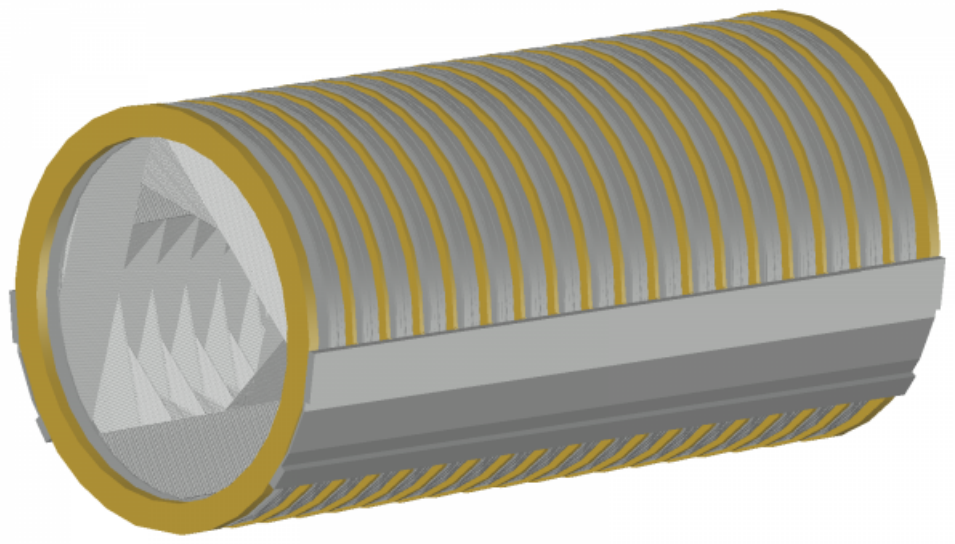


Figure 2.5: The assembled tracker, with 18 stations, 2 planes per station. Stations are shown in grey and the support structure in yellow.

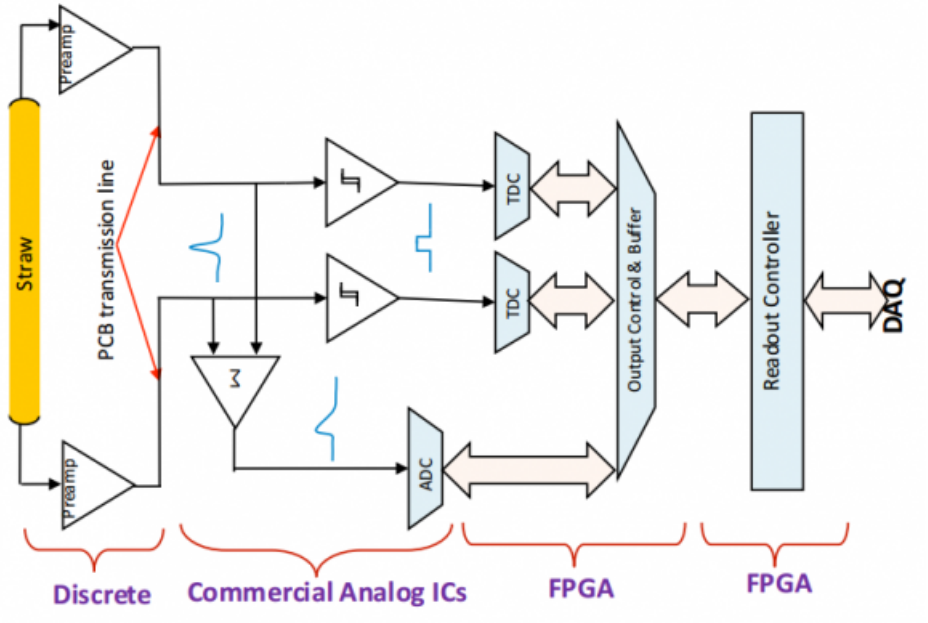


Figure 2.6: Schematic representation of the signal flow through the front end electronics.

digitized signals for each "hit" consist of two timing measurements, one at each straw end, and one amplitude.

- Readout Controller (ROC): interposes a link between the TDAQ and all other FEE components.

The time difference between the two ends is used as a measure of the hit position along the straw. The average time is used in the conventional fashion to measure the drift distance. The signal flow through the readout chain is shown in figure 2.6. Each straw has:

- 2 preamp channels, 1 for each end.
- 2 TDC channels, 1 for each end.
- 1 ADC channel, measuring sum of both ends.
- 1 High voltage feed, with disconnect.

There is a total of 46080 preamplifiers and TDC channels, and 23040 ADC channels.

The communication between the digitizer and ROC proceeds via LVDS signals (Low-Voltage Differential Signalling). For the current development there are four lines per 8 straws: clock, frame, and two data lines. However, depending on the final FPGA selection, and board layout issues, more options are available, such as using 8b/10b or similar SERDES (self-synchronizing) data transfer.

2.1.3 Readout Controller

The Readout Controllers' primary function is to receive data from the digitizer boards, buffer the data, and then transmit it to the TDAQ system. Buffering is

Figure 2.7: ROC connection to TDAQ. One of 22 servers shown.

needed to continue transferring data during the beam inter-spill time (836 ms out of each 1333 ms). Since all elements of the chain – digitizer, ROC, TDAQ – are programmable, communication is flexible. The first attempt settled on the same fixed-length format, 128 bits per hit, as used for digitizer to ROC data transfer, with an additional packet (content and length under discussion) for status and error information. The connection from ROC to TDAQ is via 2.5 Gbps full-duplex optical fibre links arranged in rings with multiple ROCs per ring (figure 2.7).

The ROC includes external DRAM for buffering; this allows data transmission to continue over a full 1.333 s Main Injector cycle. A single optical fibre readily handles several ROCs, motivating the ring architecture shown in figure 2.7. With 240 controllers, the total rate from the T-Tracker is 55 Gbps. The ROC also links the experiment's Slow Controls system to the digitizers and preamps. DACs, ADCs, and sensors are distributed through each panel and connect to the ROC via SPI and I2C.

2.2 The Calorimeter

The Mu2e detectors have been designed to reject backgrounds to a level consistent with a single event sensitivity for the $\mu N \rightarrow eN$ coherent conversion of the order of 10^{-17}. The electromagnetic calorimeter [8] is a vital link in the chain of background defences. Since the quality of track reconstruction is fundamental for background rejection, a particular concern is due to false tracks arising from pattern recognition errors, due to high rate hits on the detector. These errors are frequently due to accidental noise and lower energy electrons' hits. They may create a trajectory similar to a higher energy electron and mimic the muon conversion signal. One of the primary functions of the calorimeter is to provide a redundant set of measurements to complement the tracker data and minimize the level of backgrounds.

The electrons produced in muons' decay follow helical trajectories in the solenoidal magnetic field. Then, they hit the front face of the calorimeter crystals with a maximum energy in the 100 MeV range. In this energy regime a total absorption calorimeter employing a homogeneous continuous medium is required to meet the Mu2e energy and satisfy the time resolution requirements. The sensitive material could either be a liquid, such as xenon (Xe), or a scintillating crystal[1].

The Mu2e collaboration chose the scintillating crystals technology. Several types of crystals were considered, including barium fluoride (BaF2) and cesium iodide (CsI). The baseline design uses an array of relatively cheap undoped CsI

[1]A scintillator is a material that exhibits scintillation (the property of luminescence) when excited by ionizing radiation.

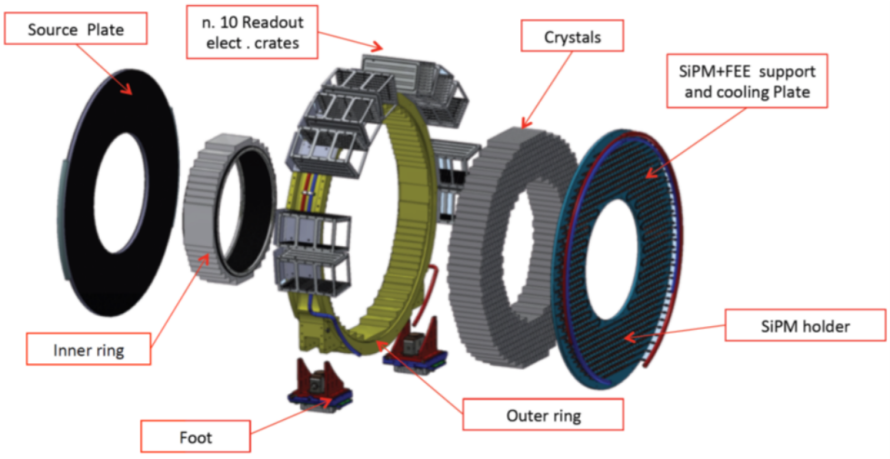


Figure 2.8: Exploded CAD view of one calorimeter disk. All the main components are indicated.

crystals, arranged in two annular disks (figures 2.8 and 2.9). Every crystal is read out by two large-area solid-state photodetectors (SiPMs)[2], preferred to standard photomultipliers because of the high magnetic field employed by Mu2e.

While the front end electronics (FEE) servicing the SiPMs (represented in figure 2.10) is mounted on the rear side of every disk, the voltage distribution, slow control and data acquisition boards are hosted in 10 crates mounted on the external lateral surface (figure 2.11).

A laser flasher system provides light to each crystal through a network of optical fibers, for relative calibration and monitoring purposes. A circulating radioactive liquid source system, housed in the front plate of the disks, provides absolute calibration and allows to determine the absolute energy scale.

The crystals are supported by a structure composed by two rings: the *Inner Ring* and the *Outer ring*, that can be slided along the beam line thanks to horizontal rails.

2.2.1 Mechanical Construction

The primary function of the electromagnetic calorimeter is to measure the electrons energy, position of impact and timing to complement the straw tracker information in the offline reconstruction of the particles trajectories. Moreover, the calorimeter provides fast information to the trigger for the online data selection. This leads to the following technical specifications for the detector:

- provide energy resolution of 5% at 100 MeV to confirm the electron momentum measurement performed by the tracker;
- provide timing resolution better than 0.5 ns to ensure that energy deposits in the calorimeter are in time with the hits reconstructed in the tracker;
- provide position resolution better than 1 cm to allow a comparison between the energy deposits position measured by the detector, and the extrapolated

[2]SiPMs stand for: *Silicon Photomultiplier*. It is a sensor that addresses the challenge of sensing, timing and quantifying low-light signals, coming from the crystals, down to the single-photon level, that is the sensitivity needed for this type of experiment

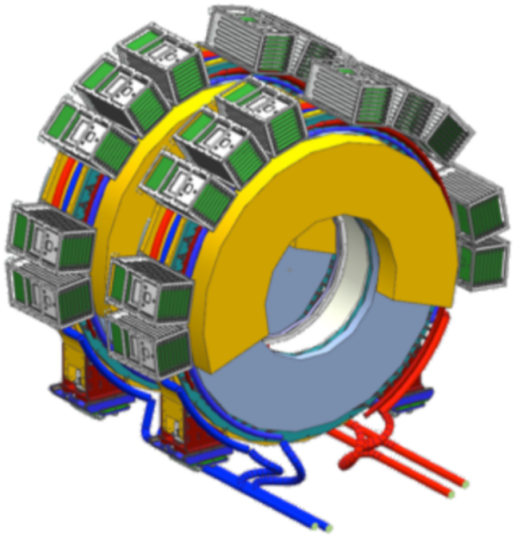


Figure 2.9: CAD model of the Mu2e electromagnetic calorimeter. The 20 custom crates host the boards for voltage distribution, slow control and data acquisition, are coloured in grey and green: The calorimeter can slide along the beam line direction thanks to horizontal rails.

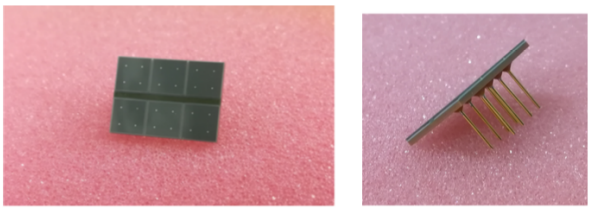


Figure 2.10: Overview of the SiPMs. The 8 pins connect the sensors to the front end electronic boards.

trajectories of the reconstructed tracks;

- provide additional information useful for particle identification that can be combined with the tracker information, improving the muon-electron separation;
- provide a trigger, either in hardware, or in software, or in firmware, that can be used to identify and select events with significant energy deposits;
- operate in the hostile, high-rate, Mu2e environment with intact functionality for radiation exposures up to 20 Gy/yr per crystal, and for a neutron equivalent flux up to 10^{11} $MeV n_{eq}/cm^2 yr$.

The calorimeter is located inside the cryostat (part of the detector solenoid). It is composed of two identical disks (figure 2.9). The inner cylinder is made of carbon fiber in order to minimize the amount of passive material in the region where spiralling electrons are mostly concentrated. The outer ring is made of aluminum, and can be as robust as required to support the crystals load (mainly the weight). The disks have an inner radius of 374 mm, outer radius 660 mm, and

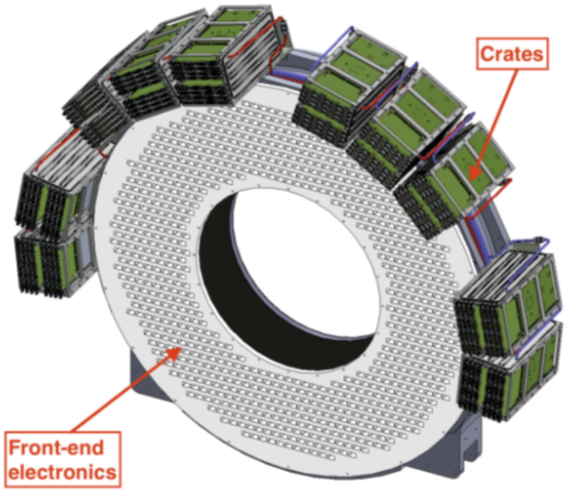


Figure 2.11: Backplate of one calorimeter disk. The front end electronics (FEE) is mounted on the holed plate, while the crates are externally fastened to the disk.

are made out of 674 staggered trapezoidal crystals. The crystals are 220 mm long with a square base and a side length of 34mm. Every crystal is wrapped with 8 layers of 25 μm thick Tyvek reflective film to maximize light transport within the crystal and minimize cross-talk among crystals.

The mechanical structure of each disk is composed of two coaxial cylinders (inner and outer ring in figure 2.8) supporting the weight of the crystals, and two plates connecting the rings together. The front plate facing the beam is made out of low radiation length material to minimize the electron energy deposit and preserve the electron energy measurement. It is designed to accommodate the calibration source circuit where a radioactive fluid flows. The back plate supports the photosensors, the front-end electronics, the cooling pipes and is made out of the polymer named PEEK (the acronym for *PolyEther Ether Ketone*). This material has the important following characteristics:

- it has an extremely limited outgassing rate, which is crucial for operation in vacuum;
- it has a low thermal conductivity (0.25 W/Km). This important feature will be explained later;
- it has a good mechanical strength and stiffness, so that the plate will have extremely limited deformations;
- it can be easily machined;
- it can be used in a high magnetic field environment.

Other important components of the calorimeter are the electronic boards. There are several species ofboards, each one with its function. Some of them provide the power to the front-end electronics (that act only as preamplifiers of the SiPMs signals) and to the SiPMs. They also perform the digitalization of the signals and are hosted in 20 *DAQ (Digital Acquisition) crates* (figure 2.11) positioned on the outer surface of the disks (10 for each disk), in order to gain

Number of Disks	2
Disk Inner and Outer Radius	374 mm,660 mm
Crystal Type, density, X0, RM	CsI,4.9 g/cm^3, 2.0 cm, 3.0 cm
Crystal Shape Parallelepiped	35 mm distance
Crystal Length	200 mm
Crystal Transversal Area	34x34 mm^2
Total number of crystals Disk 1+2	1348
Single crystal weight	1.14 Kg
Total scintillation mass	1540 kG
Number of SiPM/crystal	2
SiPM transverse dimension active area	12x18 mm^2
Total number of SiPMs	2696
Total number of LV/HV boards	136
Total Number of Digitizers	136
Total number of preamplifiers	2696
Power Dissipation AMP-HV	480 mW x 2696 = 1294 W
Power Dissipation LV/HV	4 W x 136 = 544 W
Power Dissipation Digitizer	31 W x 136 = 4100 W
Distance between disks	700 mm

Table 2.1: Summary of calorimeter parameters.

as much space as possible. Every crate hosts 9 boards, 8 of them provide power to the photosensors and perform signal digitization. The last one performs clock distribution.

A crucial function of the mechanical structure is to integrate a dedicated cooling system to cool down all the sensitive components. The heat production mainly comes from the electronic boards and the SiPms. The cooling is a critical system, since the calorimeter operates in vacuum and there is not any outer environment to extract heat spontaneously. A dedicated cooling system is thus necessary.

2.2.2 Calorimeter electronics

The entire calorimeter electronics can be divided in two subsystems with different functions and locations: the *front end electronics* (FEE) and the *digital acquisition electronics* (DAQ). The first subsystem is composed by the SiPMs and the front-end boards. It is placed in the backplate, facing the rear side of the crystals. The second subsystem is composed by the data acquisition boards, which perform the digitization of the analog signals received from the front-end boards. They also provide power and monitor the front end electronics status.

The front end electronics

The front end unit of the calorimeter is composed by the parts (shown in figure 2.12):

- one CsI crystal: the electrons impinging on the crystal's frontal surface penetrate the material and generate an electromagnetic shower. The pho-

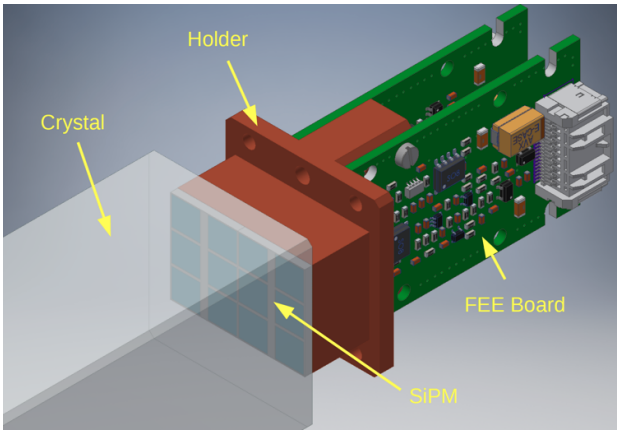


Figure 2.12: CAD model of one front-end unit. The brown structure is the copper mechanical support of the two SiPMs and front-end boards. This is the component connected directly to the backplate

tons produced by scintillation diffuse through the crystal volume and get transported to the rear side of the crystal;

- two SiPMs: they face the rear side of the crystal and convert the light produced by the scintillation into an electric signals;

- two front end boards: electrically connected to both the SiPMs. They provide power to the SiPMs and amplify their signals. The amplification has to be performed in the immediate proximity of the weak signal source, otherwise would be lost in electric noise;

- one mechanical support: made of copper to support the SiPMs and the FEE boards. It also acts as a "bridge" for heat transfer;

The interaction between the electrons and the Csl crystal generates photons, which diffuse through the crystal towards the photo-sensors. Every crystal has on its rear side the photo-sensors to convert light into electrical signals. There are two SiPMs per crystal electrically connected to one front end board. The reason for having two of them is to provide a more robust measurement and with minimal loss of data if one photo-sensor fails during data taking. The total resulting number of photo-sensors is 1348 per disk (2692 in total).

The front end electronics for the calorimeter readout consists of two discrete and independent chips (Amp-HV) placed on one unique front end board electrically connected to the back of the photo-sensor pins. The chips provide both the amplification and the local linear regulation to the photo-sensor bias voltage. Groups of 16 Amp-HV chips are controlled by one dedicated ARM controller placed on one interface board located in the DAQ crate. This board distributes low voltage and high voltage reference values, sets and reads back the locally regulated voltages. The Amp-HV is a multilayer double-sided discrete component board that performs out the two tasks of amplifying the signal and providing a locally regulated bias voltage, significantly reducing the noise loop-area.

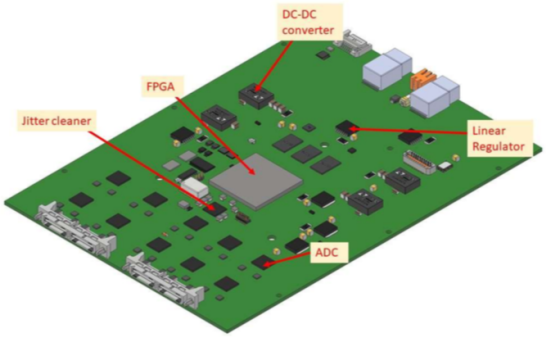


Figure 2.13: Components on a CAD model of the DIRAC board

DAQ electronics

The analog signals produced by the front end electronics are transmitted to the data acquisition boards, hosted in the DAQ crates. Since the main function of the data acquisition boards is to digitize and transmit the analog signals to the global Mu2e data acquisition, these boards are named *waveform digitizers* or *DIRAC* (Digitizer ReAdout Controller). Additional boards are necessary to provide and distribute power to the front end boards, monitor photosensors and front-end electronics performance. These boards are called *interface boards*. In the current design, there are 10 DAQ crates per disk, and each crate hosts 8 DIRAC and 8 interface boards (coupled together to form 8 "bigger boards"). Every crate also hosts one further board to provide the clock distribution.

The DIRAC boards is characterized by the following components (figure 2.13):

- 1 Field Programmable Gate Array (FPGA): it contains an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects, that allow the blocks to be "wired together". It is the most complex component of the board: logic blocks re configured to perform complex combinational functions. It processes the data received from the SiPMs previously digitized by the Analog to Digital Converters (ADC);
- 4 DC-DC converters: they transform the voltage received from the external power supply to the values required;
- 10 Analog to Digital Converter (ADC): they convert analog signals received from the front end boards into digital signals to be further processed by the FPGA and then transmitted to the DAQ system;
- 6 low dropout Linear Regulator: they provide low voltage high current outputs with a minimum of external components. It offers high precision and ultra-low dropout of 500mV in the worst case conditions;
- 1 Jitter Cleaner: it cleans the clock signal and distributes it to the ADCs and the FPGA mainly.

The interface board (also called "mezzanine board") is mechanically attached to the DIRAC and has these components:

- one voltage regulator: transforms the 28 V received from the power supplies (placed outside the cryostat) to the 8 V used by the front end electronics board;
- One ARM Controller and I2C drivers to monitor the performance of the front-end electronics.

In addition to improved background rejection, the calorimeter provides a robust approach to track reconstruction. Mu2e does not have an "event time"; all straw hits reconstructed within a μbunch therefore have to be considered by the track-finding algorithm and the track time is reconstructed as a track fit parameter. The standalone Mu2e track reconstruction attempts to find the 100 ns time slice within the μbunch with the maximum number of hits in it, and uses those hits to find a track. In the presence of the correlated in-time background produced by δ-electrons, such an approach relies strongly on the δ-electron hits being identified and excluded before execution of the track reconstruction, which at present uses a neural network-based procedure. A cluster produced by a track and reconstructed in the calorimeter can be used as a seed for the track finding. This simple cut cleans up the hits not related to real track and increase substantially the signal/noise ration, thus allowing to reconstruct tracks that are missed by the standalone tracking algorithm. This calorimeter driven track finding improves also the overall track finding efficiency.

The calorimeter system can also generate a fast trigger for the experiment that is independent of the Tracker. This trigger will take the form of an offline HLT/L3-like filter that can be used after streaming the events to the online computing farm, but before storing data on disk. The DAQ (Data Acquisition System) will read events from the tracking and calorimeter digitizers at a maximum throughput of 20 GByte/sec and the online farm will be able to fully reconstruct nearly all the streamed data. The calorimeter filter should be able to process the data in the online farm with the requirement of rejecting the background by a factor > 200. The most important aspect of this filter is that it is fully independent of the tracker, with completely different systematics due to environmental backgrounds. First studies show that, at the required level of rejection, the standalone calorimeter trigger will reach efficiency of 60-70 % that is more than enough to create samples for an unbiased estimate of the tracking trigger efficiency.

Chapter 3

TDAQ – Trigger and Data Acquisiton

The Mu2e Trigger and Data Acquisition (TDAQ) [2] subsystem provides necessary components to collect digitized data from the Tracker, Calorimeter, Cosmic Ray Veto and Beam Monitoring systems (Stopping Target Monitor and Extinction Monitor), and deliver that data to the online and offline processors for further analysis. It is also responsible for detector synchronization, control, monitoring, and operator interfaces. The Mu2e TDAQ is based on a "streaming" readout. This means that Tracker and Calorimeter detector data is digitized, zero-suppressed in front-end electronics, and then transmitted off the detector to the TDAQ system. While this approach results in a higher off-detector data rate, it also provides greater flexibility in data analysis and filtering, as well as a simplified architecture. The Mu2e TDAQ architecture is further simplified by the integration of all off-detector components in a "TDAQ Server" which functions as a centralized controller, data collector and data processor. A single TDAQ Server can be used as a complete standalone data acquisition/processing system or multiple TDAQ Servers can be connected together to form a highly scalable system.

3.1 Requirements

The TDAQ monitors, selects, and validates physics and calibration data from the Mu2e detector for final stewardship by the offline computing systems. The TDAQ combines information from about 450 detector data sources and applies filters to reduce the average data volume by a factor of at least 100 before it can be transferred to offline storage. The TDAQ also provides a timing and control network for precise synchronization and control of the data sources and readout, along with a Detector Control System (DCS) for operational control and monitoring of all Mu2e subsystems. Figure 3.1 shows a full view with focus on the interfaces connected to TDAQ: Tracker and Calorimeter ROCs, Detector Hall, WH Control Room.

TDAQ requirements are based on the following experiment attributes:

- Environment: the TDAQ system will be located in the surface level electronics room and connected to the detector by optical fiber. There are no radiation or temperature issues. The TDAQ will however be exposed to a

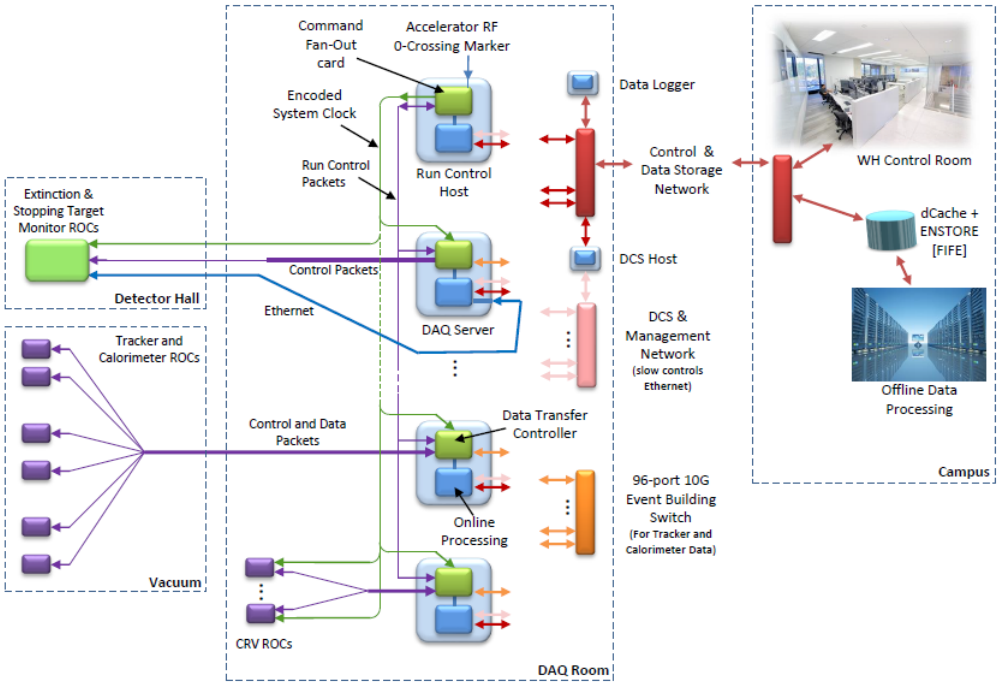


Figure 3.1: Full view of TDAQ and its interfaces with subsystems

magnetic fringe field from the detector solenoid at a level of about 20-30 gauss.

- Beam Structure: supercycle is the temporal window between two proton beams (1.4 s). Beam is delivered to the detector during the first 467 ms of each supercycle. During this period there can be up to eight 54 ms spills. Spills are proton pulses delivered to the target in the Production Solenoid. Each spill contains approximately 32 000 "μBunches", for a total of 256 000 μBunches in a 1.4 second supercycle. A μBunch is 1695 ns. Readout Controllers store data from the digitizers during the "live gate". The live gate width is programmable but is nominally the last 1000 ns of each μBunch.

- Detectors: the TDAQ system receives data from the following subdetectors:

  – Tracker – 20 736 straw tubes: 96 tubes per "panel", 12 panels per "station" and 18 stations. There are 216 Readout Controllers (one for each panel) located inside the cryostat. Straw tubes are read from both ends to determine hit location along the wire. The readout produces two TDC values (16 bits each) and typically six ADC values (10 bits each) per hit. The ADC values are the analog sum of both ends of the straw.

  – Calorimeter – 1610 crystals in 2 disks. There are 192 Readout Controllers located inside the cryostat. Each crystal is connected to two avalanche photodiodes (APDs). The readout produces approximately 25 ADC values (12 bits each) per hit.

  – Cosmic Ray Veto system – 21 504 Silicon Photomultipliers (SiPMs). There are 336 front-end boards (64 channels each), and 14 Readout

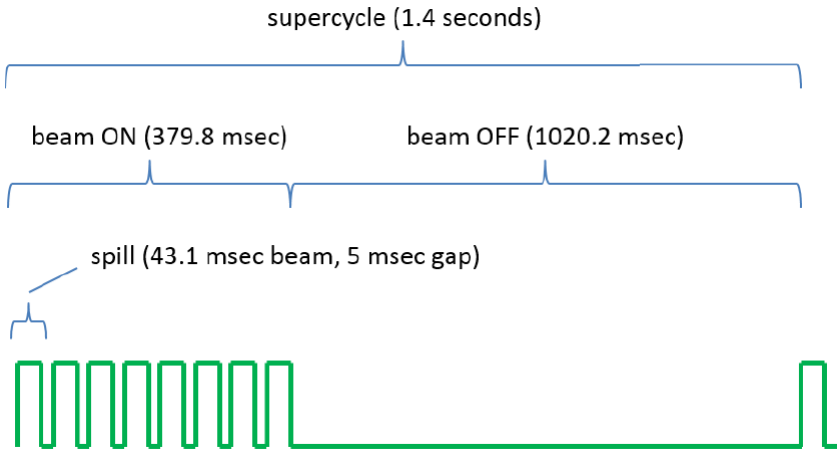


Figure 3.2: Supercycle temporal structure

Controllers (24 front-end boards each). The readout generates approximately 12 bytes for each hit. CRV data is used in the offline reconstruction, so readout is only necessary for timestamps that have passed the Tracker and Calorimeter filters.

– Extinction and Target Monitors – monitors will be implemented as standalone systems with local processing. A subset of information will be forwarded to the TDAQ for inclusion in the run conditions database and optionally in the event stream.

- Data rate: the detector will generate an estimated 150 $Kbytes$ of zero-suppressed data per μBunch, for an average data rate of about 90 $Gbytes/s$ when beam is present. To reduce TDAQ bandwidth requirements, this data is buffered in Readout Controller (ROC) memory during the spill period, and transmitted to the TDAQ over the full supercycle for an average data rate of about 28 $Gbytes/s$.

- Processing: the TDAQ system provides online processing to perform Tracker and Calorimeter filters. The goal of these filters is to reduce the data rate by a factor of at least 100, limiting the offline data storage to less than 7 $Petabytes/year$. Based on preliminary estimates, the online processing requirement is approximately 30 $TeraFLOPS$.

3.2 Architecture

Readout Controllers digitize and zero-suppress data at the detector. The data is then transmitted over optical links to TDAQ Servers in the surface level electronics room. Control information is sent from the TDAQ Servers to the Readout Controllers over the same bidirectional optical links. Data is exchanged between TDAQ Servers (via the Event Building Network) to form complete events. The TDAQ Servers filter these events and forward a small subset of them to offline storage.

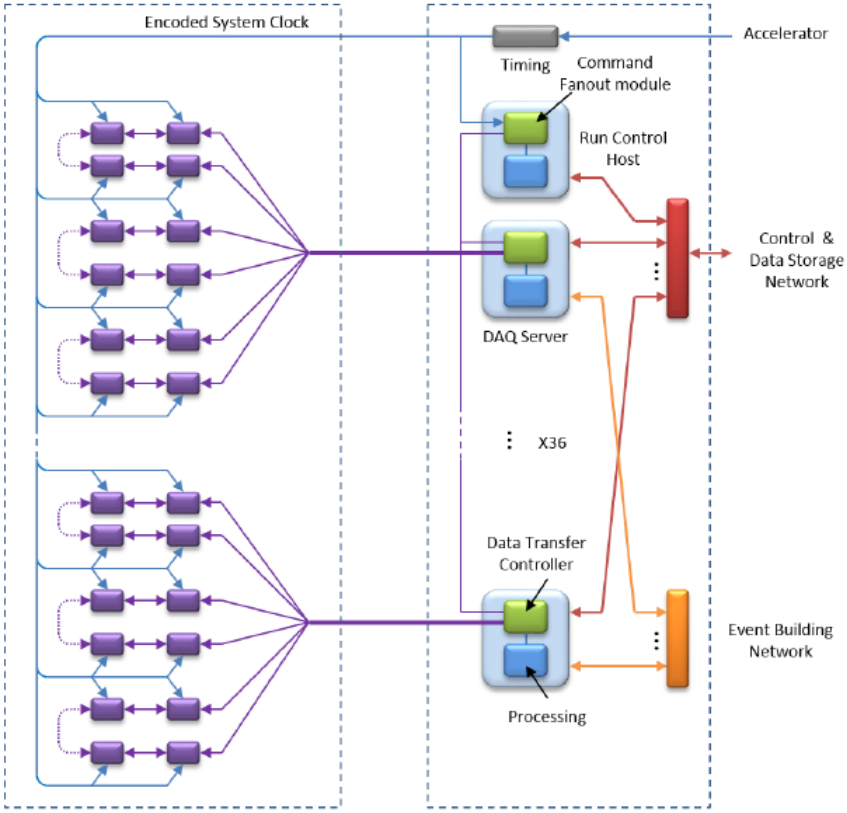


Figure 3.3: Basic system Mu2e TDAQ Architecture: on the left side, in violet, are shown the Readout Controllers situated on the detectors and on the right side is shown the actual TDAQ

The TDAQ interfaces to all other subprojects (table 3.1):

Subprojects	Interfaces
Tracker, Calorimeter, CRV	The TDAQ connects to detector readout controllers via optical links which carry fast control, slow control and data. The Timing system supplies an encoded System Clock to each detector subsystem.
Solenoids, Beamline	The TDAQ provides the infrastructure for slow control and monitoring, and readout of the stopping target monitor.
Accelerator	The TDAQ receives beam timing and status information from the accelerator for timing system synchronization. The TDAQ also provides the infrastructure for slow control and readout of the extinction monitor.
Civil	The civil construction subproject provides the surface level electronics room, power, and air conditioning for the TDAQ. It also supplies cable chases for connecting the detector hall electronics to the electronics room.

Table 3.1

3.2.1 Readout Controllers

Readout Controllers (ROCs) are not part of the TDAQ system, but rather are included separately in each detector subsystem. The number of Readout Controllers and the estimated data rate for each subdetector are listed in the following table 3.2:

Detector	Number of ROCs	Average Rate per ROCs MB/s	Total Data Rate GB/s	Number Optical Links	Number TDAQ Servers
Tracker	216	83	18	108	18
Calorimeter	192	42	8	72	12
CRV	14	214	3	14	3

Table 3.2

Readout Controllers have the main purpose of data collection, buffer management and processing. They are based on an FPGA architecture. This FPGA provides the high-speed serial transceivers (SERDES) for the optical links and manages all kind of communications: both data transfer then Detector Control System (DCS) "slow control" operations. ROC's firmware is still in development: one of the possible approach is to embed a microcontroller, which handles and

is responsible for initializing the FPGA. Because all communication is normally routed to or through the FPGA, there must be a failsafe way to reload the FPGA in the event of firmware corruption. A watchdog timer will restart the microcontroller on loss of System Clock, or if any of several FPGA and microcontroller check signals are outside nominal timing windows. This will automatically reload a "golden" version of the FPGA and microcontroller firmware from dedicated SPI memory, providing a known-good DCS connection. DCS commands can then be used to remotely load new software/firmware into the application program memory. A DCS "run" command must be sent to the microcontroller to cause it to switch from golden to application program memory. Readout Controllers in or near the detector will be exposed to a high neutron flux. SRAM based FPGAs are sensitive to radiation induced single-event upset (SEU) in the configuration and application memory. Mu2e Readout Controllers in higher radiation areas will use Microsemi PolarFire series FPGAs [11] which provide on-chip microcontroller and SERDES and a number of features to mitigate SEU, including flash based configuration and ECC protected memory and registers. Commercial integrated circuits can typically tolerate total dose of at least 100 Gy without significant degradation. In the region where the Tracker and Calorimeter ROCs are located, total dose is estimated at 10 Gy/yr.

A block diagram for a digitizer/ROC is shown in figure 3.4. As a basic description, ROCs receive and phase aligns as necessary a "punched" or "encoded" System Clock with frequency +/- 10% of 40 MHz. A clock generator multiplies the recovered System Clock to drive the digitizer sample clocks (typically 50-100 MHz). The marker encoded on the System Clock is the time-zero reference for an event window (i.e. 1695 ns μBunch during beam ON). A local timestamp counter (driven by the sample clock) measures the time offset within the event window the timestamp data. The microcontroller and FPGA interface logic operates from a local oscillator, independent of the System Clock. The ROC receives a Heartbeat Request packet for each event window. This packet contains event window readout control information, along with the System Timestamp.

Data from the digitizers is zero-suppressed, formatted and written to the ROC Data Buffer during the beam spill. Data packets are read from the Data Buffer and transmitted on the optical link during the full accelerator supercycle. The buffer is large enough to hold at least 1 second of ROC output data, and uses ECC memory for SEU mitigation.

3.2.2 Data Transfer Controller

The Mu2e Data Transfer Controller (DTC) [25] collects data from multiple detector Readout Controllers, optionally performing event building and data preprocessing. The DTC module provides an interface between the Mu2e Readout Controller (ROC) modules, and the Trigger and Data Acquisition (TDAQ) servers running the TDAQ online software framework. For Mu2e, the DTC (figure 3.5) is implemented using a commercial PCIe (Peripheral Component Interconnect Express) card located in the TDAQ Server. It is based on the HiTech Global Kintex-7 (HTG-K700) PCI Express expansion card. This card features an eight lane Gen 2 PCI Express interface, a DDR3 SODIMM socket, and a 400 pin FMC connector, all wired to a Xilinx K325T Kintex-7 FPGA. The FMC connector allows the installation of an FMC card with the optical fiber interface. This provides for multi-gigabit serial links for up to six ROC Links, a port for data

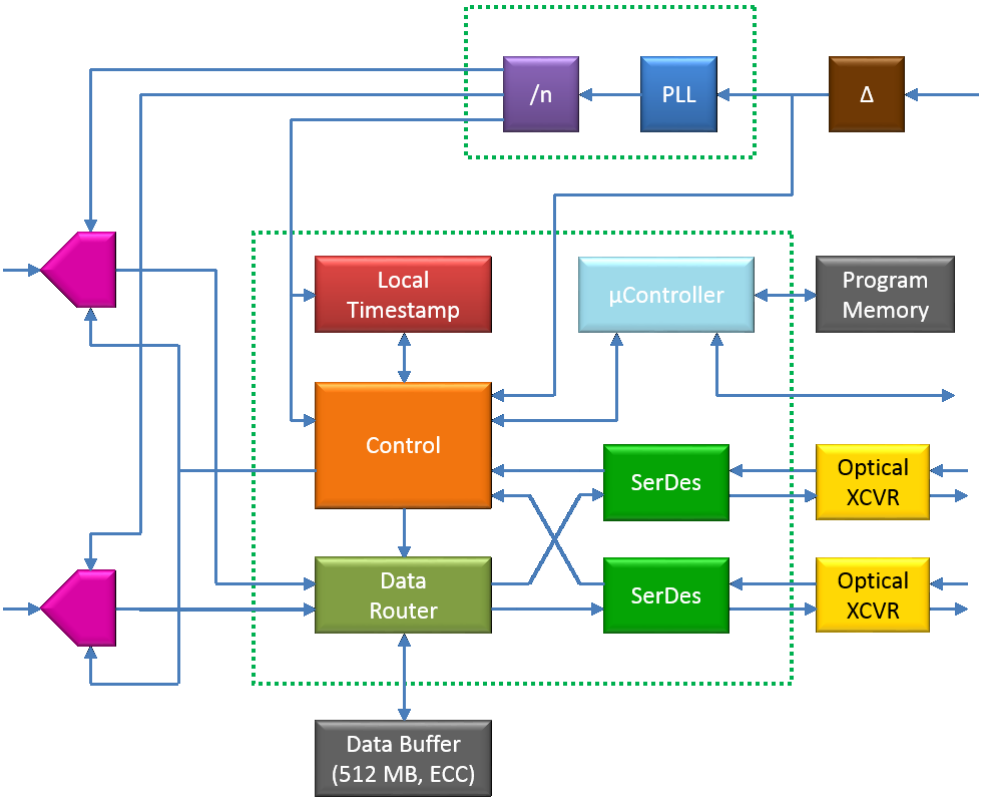


Figure 3.4: Basic Digitizer/Readout Controller Architecture.

exchange for hardware event building, and a port for the Command Fan-Out (CFO) interface. Firmware for the DTC's FPGA is based on a modified reference design provided by Xilinx.

The central component of the Mu2e TDAQ system is a commercial 3U server, which manages data collection from the Readout Controllers, Event Building, and Online processin. There are a total of 36 TDAQ servers, occupying four racks in the electronics room. The servers used for pilot system development are Supermicro X10DRD-iTP with dual E5-2680v3 processors and four 8GB ECC DDR3 2133 memory modules.

High-speed serial ports are provided by an adapter module which plugs into the FMC (FPGA Mezzanine Card) connector on the PCIe card. This adapter has eight bidirectional SFP+ (enhanced Small Form-factor Pluggable) ports, and can be used with optical or copper cabling. Six of the ports are used to connect to Readout Controller rings optical links. One port can be used to connect to the Event Building Network to exchange data between DTCs. The last port is used to communicate with the Run Control Host computer.

The DTC receives Heartbeat packets from the Run Control Host. These packets are forwarded on each attached ROC ring link. Data packets from the Readout Controllers are returned on the same links. The DTC multiplexes data from six links into one timeslice which is then transferred to the Server over PCIe, or to other DTCs via the Event Building Network.

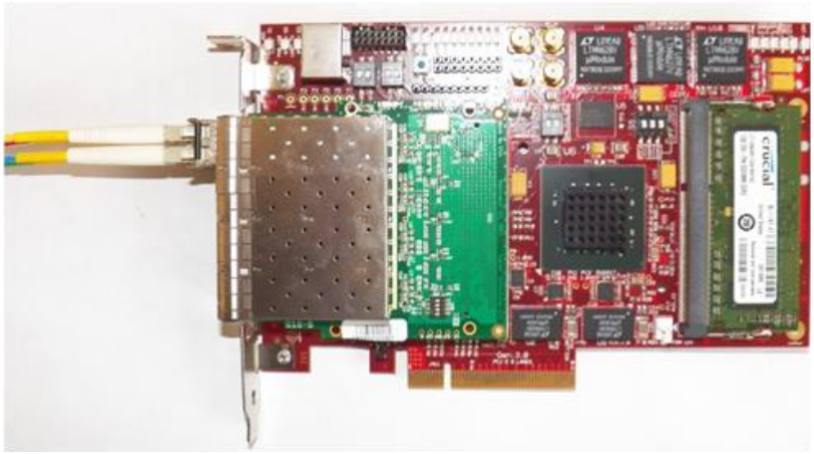


Figure 3.5: Data Transfer Controller (PCIe FPGA card and 8 port SFP+ FMC adapter).

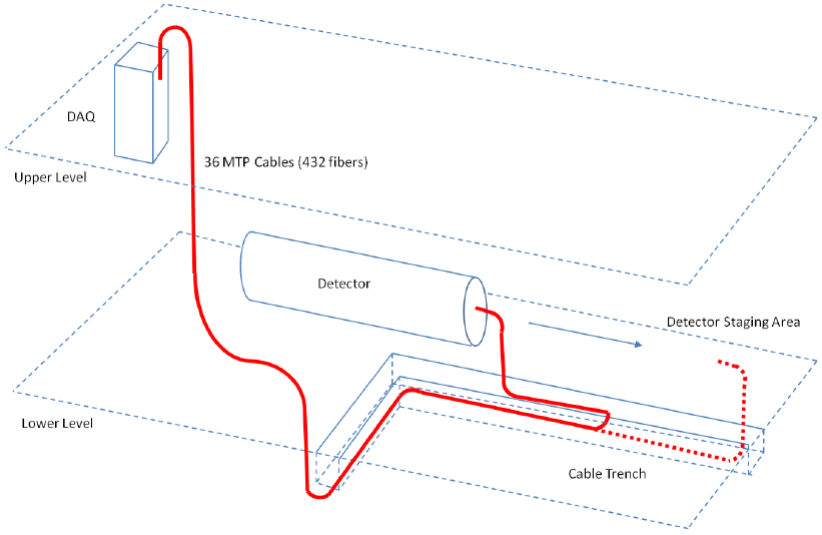


Figure 3.6: Optical cabling from Detector to TDAQ

3.2.3 DTC/ROC Interface

The detector Readout Controllers connect to the TDAQ Servers via redundant optical links. Six bidirectional links are bundled in one 12-fiber MTP cable. For Readout Controllers inside the detector vacuum, the fiber is brought out through a sealed feedthrough at the end of the DS cryostat. The boundary between the detector and TDAQ is defined as the optical connector outside the detector. MTP-LC breakout cables run upstairs to the electronics room where the TDAQ servers are located (figure 3.6). Each cable contains six bidirectional data links. For the full Mu2e system, there are 36 of these cables (a total of 216 links). Each link can support a data rate of 300 MBytes/sec, for an aggregate bandwidth of approximately 60 GBytes/sec.

Eighteen cables (108 links) are used for the Tracker, with four Tracker Readout Controllers sharing each pair of redundant links (figure 3.7). Twelve cables (72 links) are used for the Calorimeter, with six or eight Calorimeter Readout

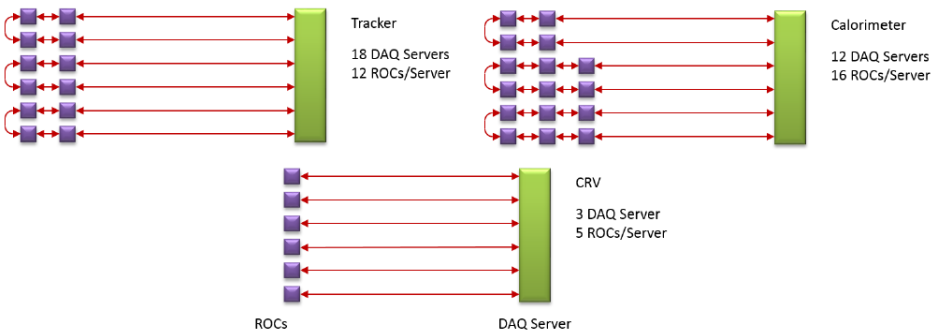


Figure 3.7: ROC Connections to TDAQ Servers (DTCs): redundant links are used to increase system reliability.

Controllers sharing each pair of links. Three cable (18 links) is used for the CRV, with one CRV Readout Controllers on each link. The Extinction and Target Monitors use one cable each (12 links).

Redundant links are used to increase system reliability, since repair would require warmup and removal of the entire detector. Each ROC has two ports and either port can be used for control/readout. In normal operation, half of the ROCs in a redundant loop will operate on one side of the loop and half on the other. The data rate will average 160 MBytes/sec per link with two Tracker or three Calorimeter Readout Controllers. If a link fails, the ROCs attached to that link can be read from the other port.

Optical links carry both control and data packets. The link interface is implemented in FPGA firmware. This means that the ROC FPGA must be operational in order to download new microcontroller software or FPGA firmware via the optical link DCS channel. To prevent loss of the DCS connection, the ROC microcontroller boot program will be located in protected memory. This program contains the DCS software and basic FPGA firmware to operate the link interface. An independent watchdog circuit will restart the ROC in failsafe mode if the external System Clock is interrupted, or if any of several internal watchdog signals are outside nominal timing windows.

The front-end component that will enable the connection is a bi-directional module composed of both optical transmitter and receiver: the Versatile Transceiver (VTRx) developed by CERN [33]. Components situated on the detectors at the front-end must meet strict requirements imposed by the operational environment for radiation and magnetic field tolerance. The VTRx development aims to minimally customize a commercial form factor bidirectional transceiver module that features a direct optical connector interface. The VTRx is to be based upon the SFP+ module MSA. Figure 3.8 shows the block diagram comparison of the VTRx with a standard SFP+. The VTRx is simplified versus the standard transceiver because both the Limiting Amplifier and microcontroller are removed. The I2C interface of the VTRx allows the user to set the operating point of the transmitter by programming the LDD ASIC.

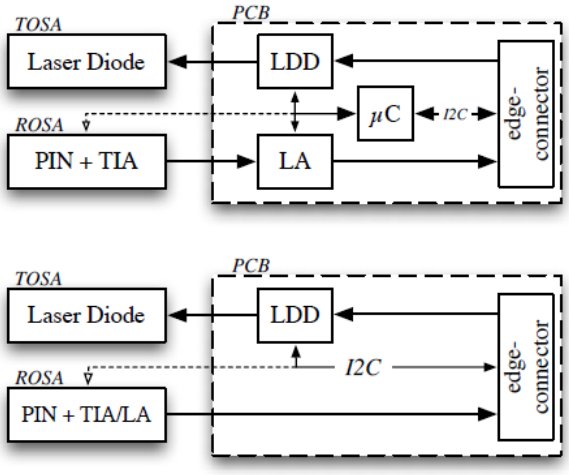


Figure 3.8: Block diagram comparison of a standard SFP+ transceiver (up) and the VTRx (down) showing: Transmitter Optical Sub-Assembly (TOSA), Laser Diode Driver (LDD), Microcontroller (μC), Limiting Amplifier (LA), TransImpedance Amplifier (TIA), PIN photodiode.

3.2.4 Run Control Host

The Run Control Host receives beam status and timing information from the Accelerator Controls network, and operator commands from the remote control room. The Command Fanout (CFO) module in the Run Control Host is responsible for generating and synchronizing Heartbeat packets. It sends a Heartbeat control packet for each event window. The CFO contains a set of standard Heartbeat packet templates (normal readout, calibration, no operation, etc.), and a default list mapping these packets to each of the ~800,000 potential event window periods in a 1.4 second supercycle. The CFO also maintains the System Timestamp which it sends with each Heartbeat packet. The Run Control Host can instruct the CFO to override the default packet on any clock or series of clocks.

The DTCs receive control and timing information from the Run Control Host on the DTC Control rings as shown in figure 3.9. These rings operate at 2.5 Gbps (3.125 Gbps 8b/10b encoded). The Command Fanout (CFO) card/optical link adapter in the Run Control Host is physically identical to that used for the DTCs.

3.2.5 CFO - Command Fan-Out

The Command Fan-Out Module (CFO Module) [24] provides an interface between the CFO Host and the DTCs. The CFO is based on the HiTech Global Kintex-7 (HTG-K700) PCI Express expansion card. This card features an eight lane Gen 2 PCI Express interface, a DDR3 SODIMM socket, and a 400 pin FMC connector, all wired to a Xilinx K325T Kintex-7 FPGA. The FMC connector allows the installation of an FMC card with eight SFP+ slots. This provides for multi-gigabit serial links for up to eight DTC Links, and ports for the System Clock and Super Cycle Start inputs. Firmware for the CFO's FPGA is based on a modified version of the DTC code.

The CFO module firmware is based on the Kintex-7 FPGA Targeted Reference Design. This reference design is provided by Xilinx Corporation to allow designs utilizing the high bandwidth capabilities of PCI Express, DDR3 memory, and

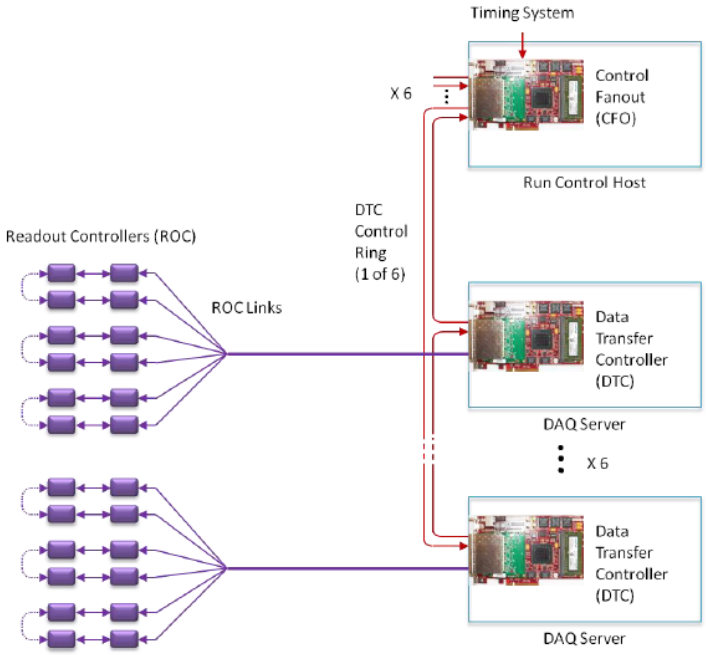


Figure 3.9: Control Hierarchy: the Command Fanout (CFO) module in the Run Control Host is responsible for generating and synchronizing Heartbeat packets. It sends a Heartbeat control packet for each event window and maintains the System Timestamp for all the Data Transfer Controller. DTC forward timing information to ROCs.

high-speed I/O to be implemented with significantly less design time required. To do this, the reference design provides code that implements the PCI Express Interface, the DDR3 memory interface, and a high bandwidth DMA engine. Two high bandwidth ports are provided to allow user developed code to be connected to the reference design. In the case of the CFO module firmware, the DTC link controller code is connected to the two high bandwidth user ports. One port is dedicated for DTC data, and the other is dedicated to DTC status and slow controls.

A DMA channel provided by the Northwest Logic DMA Back-End Core provides a DMA engine to transfer data between PCI Express and the DDR3 memory on the CFO card. DMA Channel 0 is used to write the clock cycle specific Readout Request information to the Readout Request Information Table in CFO memory.

Once configured, the CFO will issue a Heartbeat Packet for each System Clock cycle (μBunch) of a Super Cycle. A System Clock period is 1695 ns. Four bytes of cycle specific data are taken from the Heartbeat Information Table and output along with the timestamp in the Heartbeat Packet. For loopback diagnostic functionality, a given readout link's SERDES can be put into loopback mode. All packets queued for transmission on that link will then appear in the input buffer of the associated readout link.

From a software point of view, all packets are 16 bytes in length. Note that the hardware appends four bytes to the packet before transmitting it over a high speed link. These four extra bytes are then stripped from the packet by the hardware upon reception. The extra bytes are noted in the packet descriptions

0x00	No Operation
0x01	Heartbeat Packet
0x02	Data Request Packet
0x03	Set Event Tag
0x04	Increment Event Tag
0x05	Wait
0x06	Loop
0x07	Do Loop
0x08	Repeat
0x09	End
0x0a – 0xff	Reserved

Table 3.3: CFO Instruction Table.

below as a two byte 8b-10b K/D character header and two CRC bytes. The K/D character header, and the CRC bytes are not included when generating the CRC value. These characters will not appear in the input buffer with received packets when in loopback mode. About 8b-10b codification a more in deep section is dedicated later in this document.

The CFO Instruction Table is stored in memory on the CFO module. Populating the table is done via a DMA transfer to DMA port 0. Each table entry is 64 bits wide and corresponds to one CFO Instruction. The size of the table (corresponding to the number of instructions) is configurable via the CFO Instruction Table Size Register. Table entries take the form as shown in 3.3.

The CFO module receives the timestamp preset value from the CFO Host. The interface mirrors the DTC Link interfaces, in that it is an SFP+ socket populated by a multi-mode optical fiber transceiver operating at either 2.0 Gbps or 2.5 Gbps decoded (2.5 Gbps or 3.125 Gbps 8b10b encoded). Of the eight SFP+ channels on the CFO, all are available for connection to DTC links. The CFO also connects to the System Clock and the Super Cycle Start signals. All Readout Request Packets are queued by the CFO until the next positive System Clock edge.

3.2.6 Event Building

Each Readout Controller collects data from a small subset of the detector. The Event Building (EVB) function combines these subsets to form a complete detector data set for analysis by an online processor. Event building is typically done in a switching network.

The event building function can be performed by software in the Server or by firmware in the DTC FPGA:

- Server option ("software" event building) - the event building network is a commercial 10Gbase-T Ethernet or Infiniband switch (Ethernet is more widely used, but Infiniband switches are less expensive with less software overhead). Event fragments are copied from the DTC to the Server over PCIe. EVB input and output buffers are in Server memory, and the processor handles all buffer management. The advantage of the Server option is that the event building software is part of the existing artdaq framework and is therefore easier to maintain. The disadvantage is the additional processing

load for event building and management of the network interface. Standard Linux Ethernet or Infiniband drivers can be used, or the driver can be replaced with a low-level frame buffer interface to eliminate much of the overhead. Internal buffering in the switch (with standard network flow control) is used to automatically order packets. The software event building option will be implemented first. If performance and scaling are satisfactory, then this is the preferred method. Tests of artdaq on a small, four server system with Infiniband networking have demonstrated a throughput of approximately 900 Mbytes/sec per server. In the artdaq EventBuilder process, the fragment receiver layer receives data from the Data Transfer Controller, and is responsible for sending the data to the correct event builder process, using standard Message Passing Interface (MPI) protocols. The event-building layer receives data from the fragment receivers, collating them into complete events. Complete events are then sent to another thread in the same process for event processing. The event processing layer runs the art event-processing framework, which performs the data filtering.

- DTC option ("hardware" event building) - the event building network is a commercial 10G SFP+ Ethernet switch. A port on each DTC card is connected to the switch via a direct-attach SFP+ copper cable. Input and output buffers are in DTC memory, and FPGA firmware handles the buffer management. Complete events are then copied from the DTC to the server over PCIe. No IP stack is necessary. The switch is programmed with a static MAC address table (one entry for each TDAQ server). The EVB network operates synchronously. Each server loops through its input buffers sending up to 2 KBytes packets. The servers are programmed to start this rotation based on their switch position (server 0 starts with input buffer 0, server 1 starts with input buffer 1, etc.). Synchronous operation provides several advantages; 1) the synchronous network is inherently non-blocking so no flow control is necessary, 2) the amount of buffering needed in the switch is minimized, and 3) it makes diagnostics easier since all transfers are deterministic. The advantages of the DTC option are that it offloads the processor in the TDAQ Server and allows use of the DTC FPGA for pre-processing or triggering on fully assembled events if needed. The disadvantage is that the FPGA firmware for event building is more difficult to develop and maintain than the artdaq event building software.

3.2.7 System Parameters

The TDAQ system is highly scalable. If necessary, it can be expanded by increasing the number of TDAQ Servers. Parameters for the initial 36 server configuration are listed in table 3.4.

3.2.8 Detector Control System (DCS)

The Detector Control System (DCS) [27] is the window, for experimenters and detector experts, onto the status and health of the Mu2e detector. DCS must archive and present graphical user interfaces of both detailed and high level displays of power supplies, liquid and gas system's operational data, environmental temperatures and magnetic field strength, and status and run condition information for

Parameter	**Value**
TDAQ Servers	36
Detector Optical Links	216
System Bandwidth	40 $GBytes/sec$
Online Processing	40 TFLOPS
Input Event Size (average)	150 $KBytes$
Input Event Rate	200 KHz
Input Data Rate	$\leq$ 32 $GBytes/sec$
Rejection Factor	$\geq$ 100
Output Event Size (average)	150$KBytes$
Output Event Rate	$\leq$ 2000 Hz
Output Data Rate 80% uptime	$\geq$ 280 $MBytes/sec$
Offline Storage	$\sim$7 $PBytes/year$

Table 3.4: Parameters for the initial 36 server of the TDAQ system.

the data acquisition of every portion of the detector. The goal is for the DCS to not only encapsulate all of this information, but to present the information in a meaningful way to facilitate running and debugging the experiment.

Referring to the Tracker, the TDAQ Server have access to the same Ethernet network as the DCS, and so the TDAQ Server will be the DCS interface to the ROCs. The TDAQ Servers will receive permits for ROC Configure operations and firmware changes and will handle the details of configuring ROCs and downloading TDAQ Server PCI board firmware. DCS will monitor values at the ROC level, and send permit flags to the TDAQ Servers for various configuration options.

Per ROC, there are $\sim$2 voltage/current pairs, $\sim$10 temperature readings, and $\sim$10 other registered parameters. Expected monitoring period is $\sim$10 seconds. The total expected Tracker ROC contribution to DCS then for 240 ROCs is 5,280 channels. 240 ROCs is not the effective number of ROCs in the Tracker but an estimate from above. There are also high voltage supplies associated with the Tracker, a gas system for flow through the straw tubes, and a liquid cooling system for front end and ROCs. The high voltage will be a voltage/current reading for each of the 20 stations. The gas and liquid systems will likely require temperature and pressure readings at the source, and then another reading pair at each of the 20 stations. The high voltage, gas, and liquid systems then add another 124 channels to DCS. The total expected Tracker contribution to DCS is roughly 6,000 channels sampled every 10 seconds, which comes out to roughly 10 Kbps assuming 2 byte channel samples. The calculation is here:

(2 Power + 10 Temperature + 10 Registers) * 240 ROCs = 5,280 Total ROC Channels

20*2 HV + (21*2 + 21*2) Gas and Liquid Temp/Pressure = 124 Total Other Channels

5,280 ROC Total + 124 Other Total = 5,404 Total Tracker Channels

$\sim$2 Bytes per Channel per 10 seconds * 5,944 Channels = $\sim$1.1 KBps = $\sim$10 Kbps

Referring to the Calorimeter, the TDAQ Server is expected to have access to the same Ethernet network as the DCS, and so the TDAQ Server will be the

DCS interface to the ROCs as well as in the Tracker. There is a liquid cooling system and a laser calibration system for Calorimeter within the detector hall of the building that may require a few monitored values.

Per ROC, there are ∼2 voltage/current pairs, ∼10 temperature readings, and ∼10 other registered parameters. Expected monitoring period is ∼10 seconds. The total expected Calorimeter ROC contribution to DCS then for 240 ROCs is 5,280 channels. As well as for the Tracker, 240 ROCs is not the effective number of ROCs in the Calorimeter but an estimate from above. There are also a liquid system and laser calibration system associated with the Calorimeter. The liquid system will have 2 temperature/pressure readings along with ∼10 other registered parameters. The laser calibration system will have 2 voltage/current readings along with ∼10 other registered parameters. These two systems add another 28 channels to DCS. The total expected Calorimer contribution to DCS is roughly 4,000 channels sampled every 10 seconds, which comes out to roughly 10 Kbps assuming 2 byte channel samples. The calculation is here:

(2 Power + 10 Temperature + 10 Registers) * 192 ROCs = 4,224 Total ROC Channels

(2*2 + 10) Laser System + (2*2 + 10) Liquid System = 28 Total Other Channels

4,224 ROC Total + 28 Other Total = 4,252 Total Tracker Channels

∼2 Bytes per Channel per 10 seconds * 4,252 Channels = ∼0.8 KBps = ∼10 Kbps

EPICS has been chosen for DCS implementation. EPICS is open source originally developed at Argonne, and at Fermilab has most recently been implemented in the NOVA experiment and MicroBooNE experiment. Control System Studio was used for NOVA and MicroBoooNE to make the GUI on top of EPICS.

To account for channels unaccessible to ROCs and outside the scope of other monitoring systems (e.g. detector hall magnetic field sensors), DCS will distribute CAN bus and Ethernet endpoints within the detecor hall to provide a path to DCS. The baseline implementation plan is to use EPICS and Control System Studio.

3.3 Timing System

The TDAQ will generate a continuous Mu2e System Clock [26] with frequency of 40 MHz at the Run Control Host Clock Fanout module (CFO). The CFO also receives the RF-cavity 0-crossing marker from the Accelerator. Note that this marker signal is synchronous with the arrival of proton pulses every 1695 ns to Mu2e and is only active through a ∼43 ms spill (Figure 3.10). There will be at least 100 μs of markers without proton pulses to start each spill. The time between spills is arbitrary, minimum is on order ∼5 ms.

The CFO outputs a “punched” or “encoded” clock indicating the start of the Mu2e event window, which is synchronous with the Accelerator marker during beam ON to ∼10 ns accuracy - the “punch” or “marker” is a change of the duty cycle of two cycles of the clock to either 25%/75% duty cycle, or 75%/25% duty cycle as shown in Figure 3.11. These two encodings are alternated so that a lost marker can be identified by the ROCs.

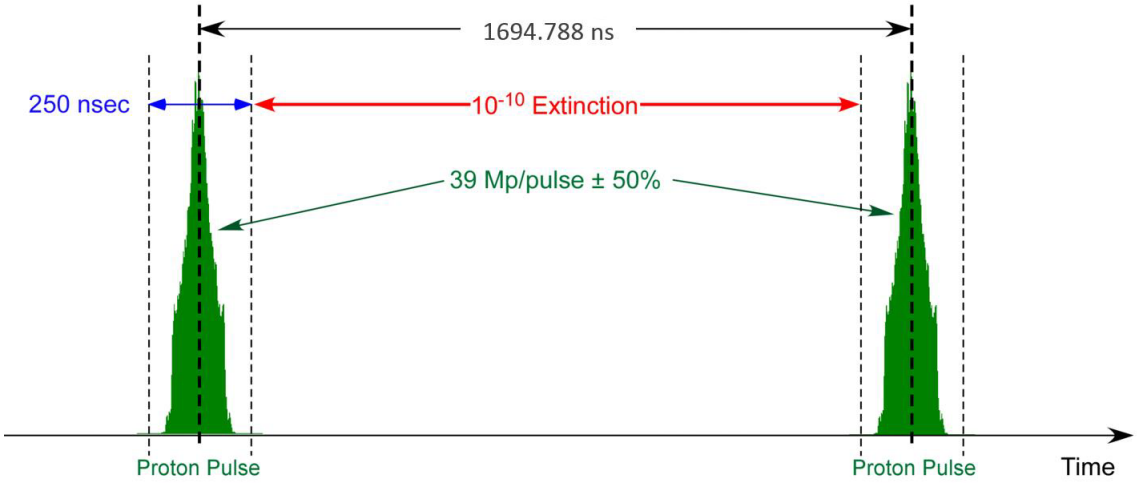


Figure 3.10: Beam Structure: supercycle is the temporal window between two proton beams (1.4 s). Beam is delivered to the detector during the first 467 ms of each supercycle. During this period there can be up to eight 54 ms spills. Spills are proton pulses delivered to the target in the Production Solenoid. Each spill contains approximately 32 000 "μBunches", for a total of 256 000 μBunches in a 1.4 second supercycle. A μBunch is 1695 ns.

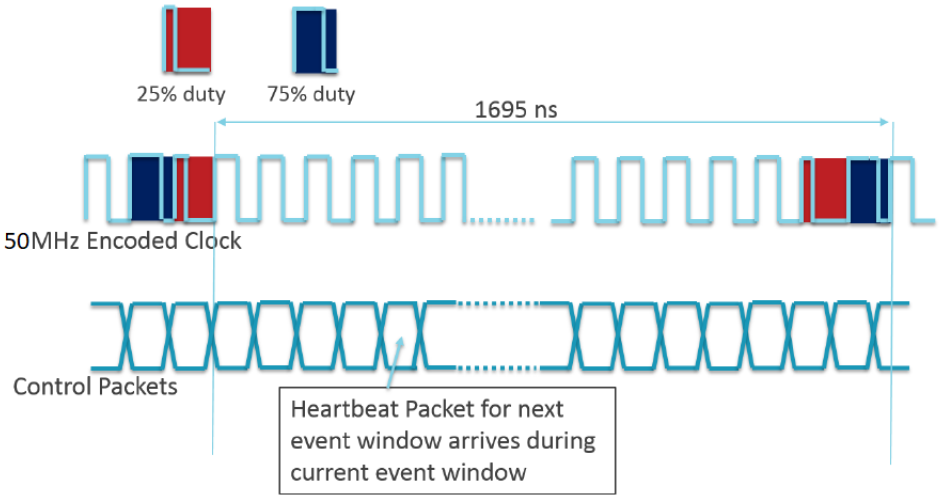


Figure 3.11: Encoded clock fanned out by the CFO. It indicates the start of the Mu2e event window. The "punch" or "marker" is a change of the duty cycle.

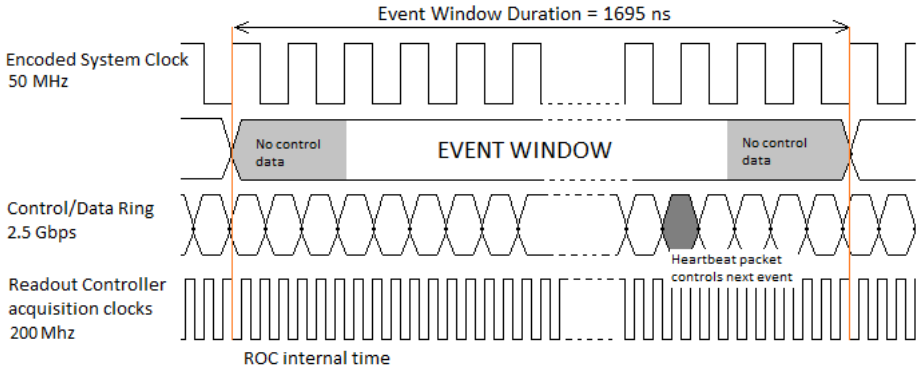


Figure 3.12: Timing diagram for synchronization signals and control data.

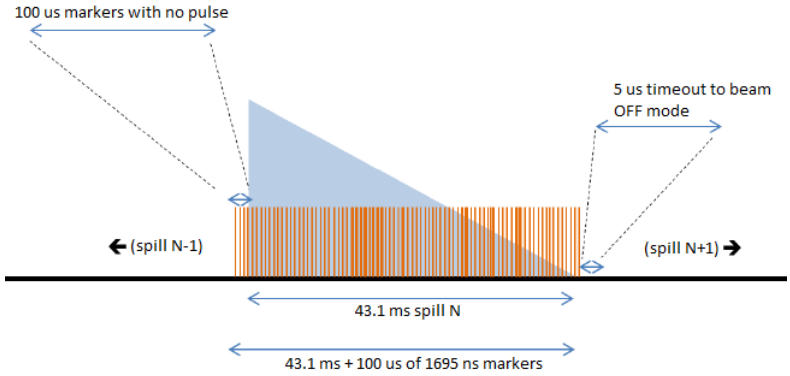


Figure 3.13: Diagram of Accelerator delivery ring marker relative to spill to Mu2e.

This encoded clock will be fanned out and distributed to outside the cryostat in the detector hall. Because of the grounding requirements, the TDAQ will distribute the signal optically from the TDAQ room to the Detector Hall (i.e. 20 optical fibers). Then the optical can be converted to electrical co-ax (SMA) in the detector hall on detector ground. The detector subsystems are responsible for distribution inside the detector vacuum and further fanout stages as needed (i.e. 18 encoded clock signals to 216 Tracker ROCs and 192 Calorimeter ROCs).

The end of the marker on the encoded clock marks the start of the next event window. Event windows are contiguous in time although detectors may have a "live gate" within an event window. During beam ON, the event windows will have duration 1695 ns, and during beam OFF, the event windows can be different duration but a multiple of the System Clock period. Identifying beam ON versus beam OFF can be done independently per 43.1 ms spill using two configurable timeout parameters such as 100 μs leading into beam ON and 5 μs leading into beam OFF (Figure 3.13).

The data links to the front ends will be used to send a 16 byte Heartbeat packet describing the next event window before each event window begins. This packet includes an 8 bit TDC value predicting the relative phase of the next event window and the System Clock to better than 1 ns resolution. This packet also provides "live gate" info and a 48 bit Mu2e System Timestamp (no wrap-around for 15 years) labeling the next event window. Heartbeat packets are not transmitted during the period 50 ns before and after the event window marker. The front

System Clock Frequency	40 MHz
System Clock Jitter	< 500 ps
ROC-to-ROC Synchronization	5 ns
Event Window Duration	1695 ns min
Optical Fiber Type	62.5/125 μm Multi-mode
Optical Fiber Light Wavelength	1310 nm
Optical Fiber Connector Type	LC
Optical Module Type	SFP
Acceptable Optical Transceiver Module	Finisar FTLF1217P2xTL Avago HFBR-57E5APZ

Table 3.5

ends can use the information distributed from the TDAQ however needed (i.e. if 10 ns resolution is enough, the front ends can ignore the 8 bit TDC value). There are at least two ways to recover the System Clock at the front ends. Most FPGAs can recover a clock to better than 200 ps jitter. External to FPGA clock recovery and jitter cleaner circuits can recover the clock to better than 1 ps jitter.

The System Clock is not guaranteed to arrive to all detector ROCs phase aligned. Phase alignment at individual ROCs is accomplished by an adjustable delay at the ROC clock input. Additional alignment is provided by software or firmware calibration and may result in internal timestamp synchronization offsets. Each ROC generates its own internal high speed digitization clocks, phase locked to the System Clock. A Clock Generator on the ROC is programmed via the DCS connection to drive the digitizers at any N/M multiple of the System Clock. Each ROC also generates an internal timestamp for timing data within the Event window. This phase alignment and internal timestamp synchronization offset can be calibrated across detectors using cosmic rays or proton pulses.

To accommodate the new point-to-point topology for the Tracker and Calorimeter, forced by the design decision to go to a single bi-directional VTRx at the ROC, the System Clock and event marker will be transmitted over the serial link running at 4.0 Gbps. The 40 MHz System Clock will be represented by a special clock marker. Both the clock marker and the event marker will be represented by two 8b-10b K-characters, that are only transmitted on a System Clock edge. By only transmitting on System Clock edges, the System Clock can be extracted and event markers can be extracted associated with a System Clock edge. The markers must be received with fixed latency with respect to the source to maintain ROC-to-ROC synchronization. Fixed latency over the serial links is achieved by removing as much elastic buffering as allowed in the SERDES throughout the data path. Note that after extracting the 40 MHz System Clock at the timestamping front-end ROC, the clock can be scaled up by integer multiples (e.g. 200 MHz) to be used to timestamp data. Accounting for the variable latency of the serial datapath from ROC to ROC is handled by coarse and fine granularity delay offsets implemented at the timestamping front-end. Determining the offset to apply is achieved by loopback to determine the latency of the datapath.

3.3.1 Timestamps

Each Readout Controller generates its own internal timestamp for data within an event window. This internal timestamp counter is driven by the digitization clock and is reset at the beginning of an event window. It may be 1 or 2 bytes, depending on the resolution of the detector. The digitization clock frequency is determined by the ROC and can be different across different detectors. In addition to the internal timestamp generated by each ROC, there is a System Timestamp generated by the Command Fanout Module in the Run Control Host. This is a six byte value which increments for each event window. It has a range of at least 15 years. It can be stopped and restarted at any value as long as the new start value is higher than the previous stop value. The System Timestamp can be correlated with actual calendar time, or the high bytes can represent a Run Number, supercycle, etc. The System Timestamp is sent by the CFO to the DTCs at each System Clock. The DTCs broadcast the timestamp to all attached ROCs in a Heartbeat packet. The DTCs also send a System Timestamp as part of Data Request packets, and the ROCs return the System Timestamp in the Data Header packet. Sending the System Timestamp directly to the ROCs for each event window (instead of relying on a timestamp generated from the ROC itself) avoids loss of ROC event synchronization with the rest of the system as a result of missing or extra decoded event windows. The System Timestamp counter increments for every event window. The event windows are contiguous during a run, whether or not there is beam. This allows readout (e.g., acquisition of calibration or pedestal data) at any time in the accelerator supercycle. The System Timestamp is the only value used for event identification in the TDAQ system. Events are not renumbered following various stages of filtering.

3.4 TDAQ Software: *artdaq*

The software architecture is based on *artdaq* [6]. This software runs on TDAQ servers and on dedicated control and monitoring computers. *artdaq* is a toolkit of C++ 2011 libraries and programs for use in the construction of TDAQ systems. It provides functionality that includes the following:

- management of the readout and configuration of the TDAQ hardware. This makes use of experiment-supplied software components.
- routing of data between threads within a process, between different processes, and between different machines, and for assembling complete events from these data.
- encapsulation of the data being routed, and support for experiment-specific raw data formats to provide type-safe data access.
- event analysis and filtering using the art event-processing framework.
- basic control and monitoring applications.
- infrastructure for distributing configuration data to TDAQ processes.

The *artdaq* data acquisition toolkit is used to build the Mu2e TDAQ software system. *artdaq* provides software applications for managing the data flow as well

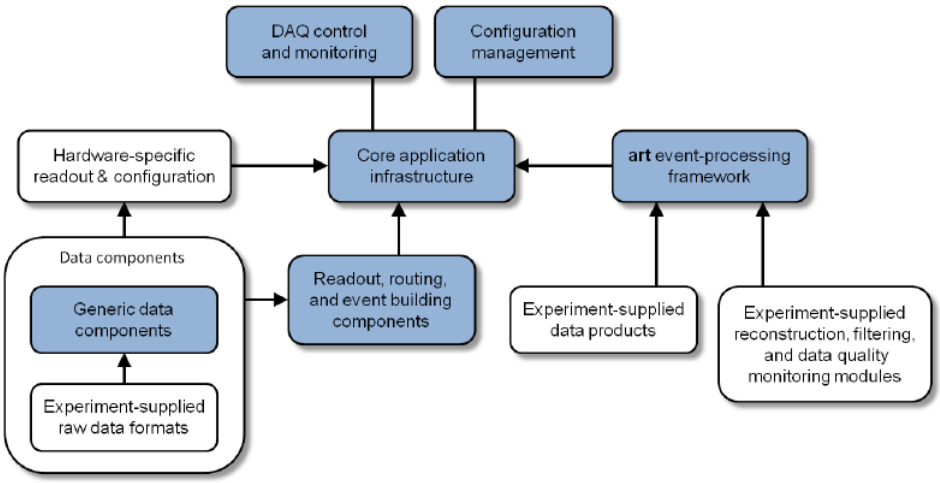


Figure 3.14: *artdaq* Architecture: core components are shown with a blue background, while experiment-supplied components are shown with a white background.

as libraries and applications for encapsulating the data, analyzing the data, and performing other basic data acquisition functions. The core data-flow applications in *artdaq* consist of the following:

- BoardReaders that configure and read out hardware modules, and send data fragments to EventBuilders,
- EventBuilders that assemble full events and pass the events to instances of the *art* analysis framework for reconstruction and filtering,
- Aggregators that organize events in time order, write them to disk, and analyze them to monitor the quality of the data.

These applications are shown in Figure 3.15 along with additional components that are part of *artdaq*. The additional components include infrastructure for sending and receiving control messages, managing the state of individual processes and the full system, logging messages to central loggers and viewers, and the sending and parsing of configuration parameters.

The toolkit is designed to provide core functionality while allowing experiments to customize the hardware readout and event analysis as needed.

Key term are of the TDAQ Software are: online DAQ software, *artdaq*, *art* and *otsdaq* [28].

The term "online DAQ (data acquisition) software" refers to the software used to monitor, select, and validate physics and calibration data for the experiment. It is easy to creep the scope beyond above. For example, the above often involves the need for some control of the front-end electronics, so the extreme would be for all control of the front-end electronics to go through the online DAQ software. Other scope creeping features might include configuration parameters, configuration change tracking, user access permissions, user preferences, etc.

Acronym for "*art* data acquisition" *artdaq* is a data acquisition toolkit which provides functionality for data transfer, event building, event reconstruction and

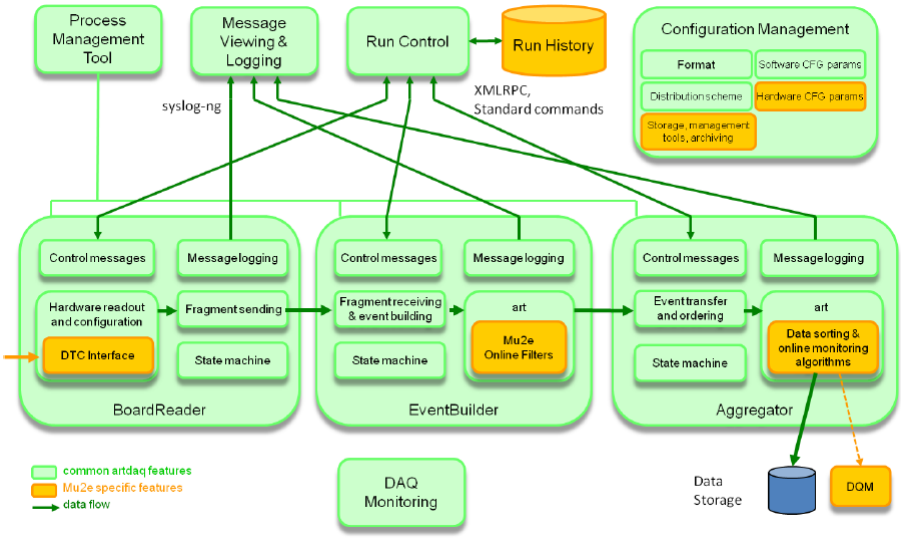


Figure 3.15: *artdaq* components. Applications and infrastructure components that are shown in green are part of the core *artdaq* toolkit. Components shown in orange are modules that experiments provide to read out their specific hardware and perform their specific analyses and monitoring.

analysis (using the *art* analysis framework), process management, system and process state behavior, control messaging, local message logging (status and error messages), DAQ process and *art* module configuration, and the writing of event data to disk in ROOT format. In general, the *artdaq* toolkit includes one or many ways to do things, and it is left to the experiment to choose the tools from the toolkit and provide the glue for a complete system.

art is not an acronym, it is an event-processing framework for particle physics experiments, like Mu2e. Experiments use the *art* framework to build programs that process data in a variety of contexts: high-level software filters, online data monitoring, calibration, reconstruction, simulation, and analysis. Mu2e offline uses *art*, Mu2e online uses *artdaq* and thus *art* as well. For example, the Mu2e online trigger decision is made by a set of *art* modules running in the online environment (but primarily developed in the offline environment).

otsdaq is an acronym for "off-the-shelf data acquisition." *ots* for short. It is the online DAQ software framework that Mu2e has chosen. *otsdaq* uses the artdaq DAQ framework under-the-hood to provide data handling flexibility and scalability. In addition, *otsdaq* provides a web interface to configure, control, and monitor the online DAQ software entities. In general, *otsdaq* has chosen the tools from the *artdaq* toolkit for the experiment, and provided the glue for a coherent experience for all users (shifters, experts, etc.) from Chrome or Firefox.

otsdaq and *artdaq* are developed by the Fermilab Scientific Computing Division and developments are in two directions: server side and web side.

About the online DAQ software development, server side is C++. User code is added through plugins (C++ classes inheriting from the appropriate class). Types of Mu2e online DAQ software plugins are:

- Front-end interfaces - code to communicate with an external device, e.g. there's a plugin for the DTC, and for each type of ROC

- art modules - e.g. trigger modules, online monitor modules
- artdaq Fragment Generators - code to decode data and transmit to artdaq event builders
- Data processors - code for custom data handling, e.g. datastream-to-ROOT for Visualizer
- Configuration table handlers - code for custom handling of configuration data, e.g. to output FHiCL, or provide helper-abstraction functions like getVolume() of object with size specified in configuration parameters.

Web side is HTML and JavaScript. User code is added in the form of web-apps through .html files (including the appropriate .js and .css files). Any custom user web-apps for Mu2e is not been generated yet, but the facility is present. For example, overlaying Calorimeter ROC temperature color-coded on a 3-D representation of the detector with slider controls to set thresholds, this would be a custom user web-app.

All data filtering and triggering in the Mu2e TDAQ architecture is done in firmware or software. The production TDAQ will use 36 dual-CPU servers. The online processing system must handle a total rate of 200,000 events per second, an average of 5,600 events per second per server. The *art* analysis framework will be used as the environment in which the online processing algorithms are executed. It provides the infrastructure for running software modules that are provided by experimenters and managing the data that is analyzed and produced by the analysis modules. It has been developed at Fermilab for use in current and future intensity frontier and cosmic frontier experiments, and it is currently used in the offline environments of the Mu2e, NOvA, LBNE, and other experiments. It is also currently used in the TDAQ system of the DarkSide-50 experiment, which is also *artdaq*-based. The use of the same analysis framework online and offline has substantial advantages, most notably the ability for physicists to develop algorithms independently of the full TDAQ system and move them to the online environment when they are ready. Within the TDAQ system, EventBuilder processes handle the starting of *art* threads and the transfer of full events to *art* for analysis. It also handles the configuration of the *art* framework and the analysis modules using the configuration parameters that it receives from Run Control. The same configuration language is used to configure *artdaq* processes as is used to configure *art*.

As part of the software interface to the DTC, a Linux device driver for communicating over the server PCIe bus is being developed. The driver will be responsible for managing the buffers into which the data is written when it is received from the ROC, responding to the interrupts when DMA transfers complete, notifying the user code that data is available, and delivering the data to the user code.

3.4.1 *otsdaq*

otsdaq is the Ready-to-Use data-acquisition (DAQ) solution aimed at test-beam, detector development, and other rapid-deployment scenarios. As stated earlier, *otsdaq* uses the artdaq DAQ framework under-the-hood, providing flexibility and scalability to meet evolving DAQ needs and provides a library of supported

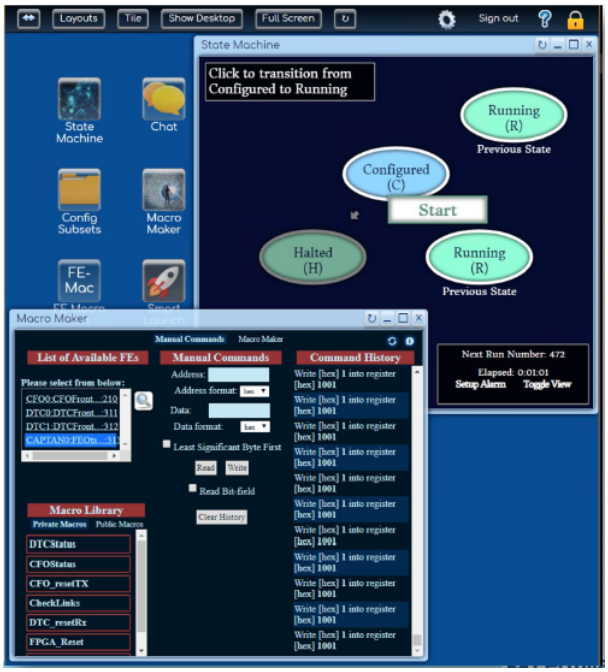


Figure 3.16: *otsdaq* web desktop environment: open windows are the State Machine and the Macro Maker app.

front-end boards and firmware modules which implement a custom UDP protocol. Additionally, an integrated Run Control GUI and readout software are provided, preconfigured to communicate with *otsdaq* firmware.

otsdaq comes as a web page. The ots web desktop environment is your portal to all of the possibilities of *otsdaq*. Briefly, desktop features are: same user on multiple browser tabs, monitors and computers, configurable desktop window icons and folders with access permissions, window layout presets (Global and per user) and window manipulations (Tile, resize, move, minimize, maximize, refresh, close).

Front-end interfaces are plugins that are considered to be the specifics for how (i.e. C++ to write and read) to interface to a device external to *otsdaq*. In particular, *otsdaq* is used to control the DTCs. The DTC Client library is the low level interface code to the TDAQ Data Transfer Controllers. It provides the PCIe interface functionality, and implements handling the packet protocol that the DTCs and readout controllers use in Mu2e. The *otsdaq* front-end interface plugin for the DTC is a wrapper around the DTC Client library and the DTC Board Reader. *otsdaq* presents a State Machine, visible in figure 3.16, that allows to easily configure and run DTCs. Macro Maker is a tool that allows the user to execute front-end interface writes and reads, and build sequences of writes and reads, i.e. macros. Macros can be saved per user or made public for all users. Macro Maker is useful for low level debugging of front-end interfaces, and early development. Macros can be exported to C++ or directly to a target plugin as a FE Macro. FE Macros are C++ member functions of a front-end interface plugin class. The primary utility is that, with no user effort, FE Macros are available through the web-interface - through the FE Macro Test web-app or custom user web-apps. FE Macros have strings or numbers as input and output arguments. The FE Macro Test web app also runs generic private and public macros from

Macro Maker. The concept of Macro Maker mode is that anyone (e.g. a firmware developer) who just wants to use front-end interface plugins with FE Macros, or generic macros, could use this simplified mode without tracking configuration changes or using the state machine. A FHiCL parameter file is used to import the configuration. When Macro Maker mode is launched, the state machine is automatically transitioned through to the Configured state.

Another feature is the configuration tree. The configuration tree defines the hierarchical relationship between all entities in the online DAQ system, and all of their parameters. When *otsdaq* is launched, the executables that start are the ones enabled in the configuration tree for that node. Then later, when the state machine transitions to the Configured state, the children of the executables are instantiated based on the parameters defined by the chosen configuration alias. Configuration alias maps to a configuration tree which fully defines the online DAQ configuration (likely, the configuration alias string and the translation to group name and group key gets recorded in the run conditions database). A configuration tree can have multiple roots and multiple branches (as shown on the bottom-right). At the lowest level, *otsdaq* stores configuration data in tables and tracks table changes as versions. The configuration tree is an abstraction extracted from groups of tables. Any entity that needs configuration parameters can have read-only access to the configuration tree API - with calls like getChildren(), getNode(), and getValue() - and access to the table plugins and their helper abstraction functionality.

Other features are the console and the code editor. The console web-app help users exist remotely and remove the need to access the linux terminal. The console core functionality is built on *artdaq* message facility. Messages have labels, line numbers, and severity; they can be filtered, and user preferences are saved per user. Printouts to the terminal, log files, or the web console can be generated from any user plugin code by using the ots output macros. The code editor web-app is a tool that allows for editing and viewing of source code and text files. Configurable permission levels give write access or not. The code editor has vertical and horizontal view split, and can spawn multiple browser tabs and windows. The code editor might help developers standardize code format, encourage collaboration, and allow for remote development.

ots uses *artdaq* database as its external database interface. *artdaq* database is a JSON document based database which can be persisted on the filesystem or by *mongodb*. When a new table version is created by *ots*, a new JSON document is created in *artdaq* database. For redundancy and high availability, *mongodb* is used. The approach is to have replica sets which each maintain the same data set, and then a cron job that automates daily backups to a directory tracked by TiBS (Fermilab Core Computing backup/restore service).

Data processing is the primary responsibility of the online DAQ. Mu2e's event window data will be processed through *artdaq* modules. However *ots* allows for data processor plugins in general (i.e. interfacing to *artdaq* makes use of particular data processor plugins provided by *otsdaq* core functionality). Data processor plugins inherit generic data handler functionality, and can add custom handling beyond that. For example, an aspect of the *ots* visualization tools make use of specialized data processor plugins that generate ROOT objects that can then be viewed in the web desktop. Users can make *ots* data processor plugins for any purpose they dream up. When the *artdaq* data processor plugin is used in *otsdaq*,

users have access to the flexibility and scalability of *artdaq*. The *artdaq* data processor plugin instantiates an *artdaq* Board Reader with a Fragment Generator plugin. Based on the configuration of the online DAQ system, the user can also instantiate *artdaq* Event Builders, Dispatchers, and Data Loggers. For Mu2e, there will be a Board Reader for each DTC, one Tracker/Calorimeter Event Builder per server, (each running the trigger algorithm with as many art analyzer processes as fit on the server, around 20), a second-level Event Builder which will integrate CRV data, several Data Loggers on dedicated nodes for writing data to online storage, and several Dispatchers to provide online data quality monitoring. *artdaq* tracks a large number of metrics covering pretty much everything about event rate and dataflow, which can be enabled at the metric plugin level; the user can send a subset of metrics to EPICS, everything to *Ganglia*, and only the most important ones to a file, all at the same time.

Chapter 4

DRAC - Digital Readout Assembler & Controller

The DRAC board is the Mu2e Tracker digitizer and readout controller board (figure 4.1) [23]. It sits on the outer edge of each Mu2e Tracker panel and services the entire panel via 12 bit 50 Mbps ADCs (MAX19527) digitizing the hit energy from each of the 96 straws. The time of the hits from the two ends of the straws is digitized inside two Microsemi PolarFire FPGAs (MPF300TS-1FG1152), called DIGI HV and DIGI CAL. A third Microsemi PolarFire FPGA, called ROC (Readout Controller), is connected to each DIGI via four 5 Gbps SERDES lanes and to the TDAQ via a two 2.5 Gbps fibers connected to a Data Transfer Controller. The DRAC acts as a mezzanine board of the Digital Mezzanine Board (DMB) which is glued on the panel itself. The DMB is mostly an analog board providing power to the DRAC and the two Analog Mezzanine Boards (AMB) which house the preamplifiers receiving signals from the two end of the straws (figure 4.2).

Microsemi offers the Microsemi Libero System-on-Chip (SoC) design suite comprehensive of development tools for designing with flash FPGAs, SoC FPGAs, and Rad-Tolerant FPGAs [21]. The suite integrates industry standard Synopsys Synplify Pro synthesis [13] and Mentor Graphics ModelSim simulation with constraints management, debug capabilities, and secure production programming support. The Libero SoC v12.0 release supports SmartFusion2, IGLOO2, RTG4, and PolarFire devices.

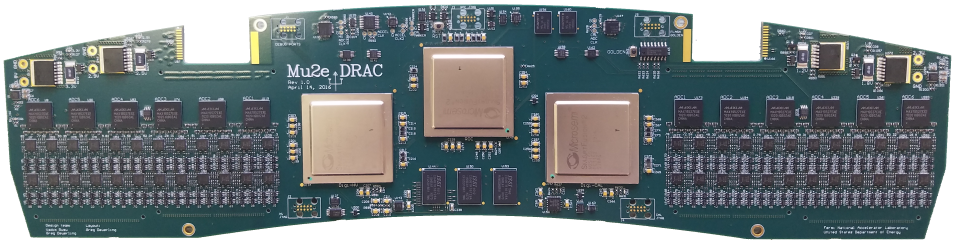


Figure 4.1: DRAC Board

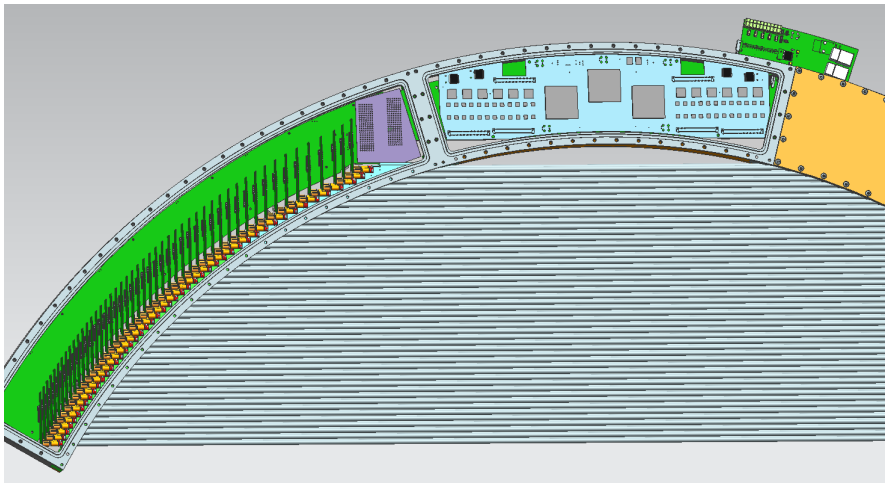


Figure 4.2: Tracker panel: in green are shown the ADCs (MAX19527) digitizing the hit energy from each of the 96 straws while in light blue is shown the DRAC board.

4.1 Data Transfer between the Digitizer and ROC

One Read Out Controller (ROC) receives data from digitizers which are connected to a panel of 96 Tracker straw detectors [23]. A "Digitizer" describes a front-end consisting in part of ASICS or FPGAs, preamplifiers, communication protocols, and interface ICs. However the ROC/Digitizer interface described below is assumed transparent to the front-end architecture. One ROC controls data flow from, and sets Digitizer parameters for, a panel. It receives output and status data from the Digitizers and sends this information to the TDAQ. It receives slow control parameters and action requests from the TDAQ, and communicates pertinent information to the Digitizers. Depending on the number of planes, the Mu2e Tracker has a nominal 216 ROCs. The Digitizer receives signals from both ends of 4 (or 8) straws and multiplexes 4 of these into one output buffer before sending a packet of data to the ROC on 1 (or 2) LVDS lines (Low Voltage Differential Signal lines). Digitized data transmission from the output buffer to the ROC is clocked at 200 MHz. A block diagram of the ROC electronics is shown in figure 4.3.

Every 1695 ns during a super-cycle, each ROC sends to its Digitizers a signal indicating the start of a μspill and a 200 MHz clock. These are used by the digitizer to synchronize event time with respect to the μspill start (table 4.1).

The ROC also sends to the Digitizer the lowest 4 bits of the μspill number and a programmable gate which starts and ends data digitization during a μspill. Both the μspill number and the gate follow the μspill start on the same LVDS line. Nominally a μspill start occurs every 1695 ns for 54 ms (table 4.1) and the data gate follows after a nominal latency of 670 ns. Thus hit data is digitized for a period of some 1025 ns. This data is stored in a local FIFO output buffer at the digitizer before being sent to the ROC. Transfer from the output buffer begins as soon as data appears in the buffer and continues until all data collected during a μspill has been transferred. Data transmission is then latent for at least one clock pulse before data transfer from the next μspill begins. After inserting the data collected for a μspill in the buffer, the digitizer inserts an end-of-μspill

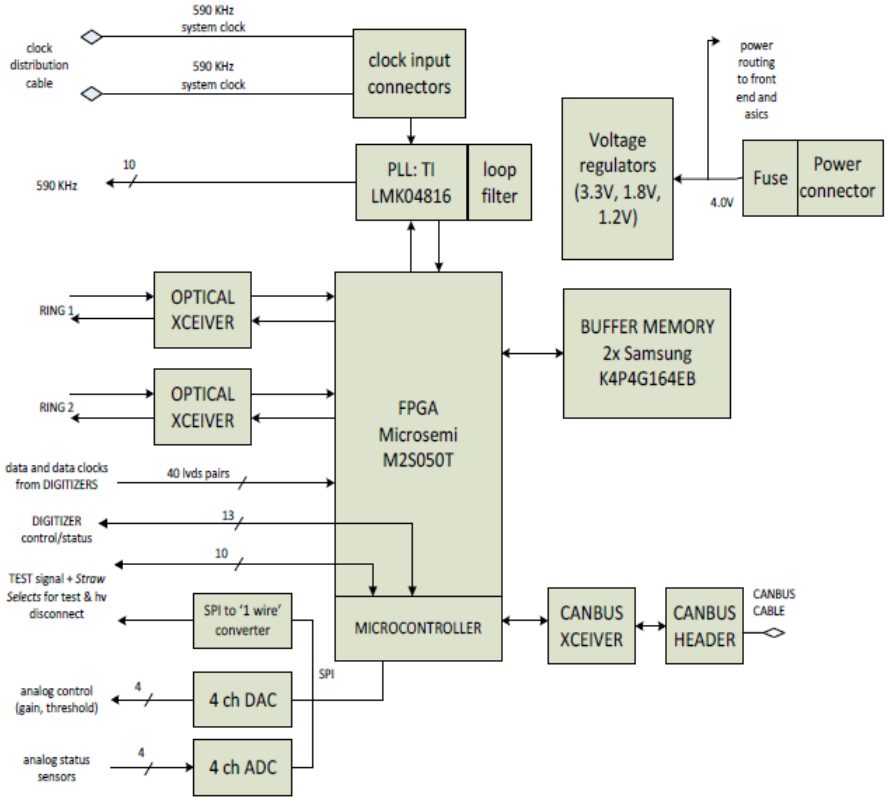


Figure 4.3: A block diagram of the ROC electronics.

data packet. This allows data transfer from the FIFO buffer to be transferred without loss of synchronization even if the transfer continues beyond a μspill. The amount of data collected during a μspill is determined by the single event rates of the straws, and the output buffer must have sufficient depth to provide an average data transfer from the 4 multiplexed straws during a super-cycle. Digitizer performance and status are communicated to and from the ROC through status words in the header and appended end-of-μspill data packets. Table 4.2 gives the total number of required LVDS lines.

A timing diagram describing the synchronization between the Digitizer, output buffer, and data-transfer to the ROC is shown in figure 4.4. It is assumed that digitized data is collected and stored without pileup in a local buffer. The TDCs record an event time with respect to the μspill start using a 16 bit word. This bit

Parameter	**Value**	**Units**
Duty Factor (Total Spill Time ÷ MI Cycle Lenght)	32	%
Duration of each spill	54	ms
Spill On Time per MI cycle	497	ms
Spill Off Time per MI cycle	836	ms
Time Gap between 1^{st} set of 4 and 2^{nd} set of 4 spills	36	ms
Time Gap between spills	5	ms
Pulse-to-pulse intensity variation	±	%

Table 4.1: Timing summary.

Number of Channel/ASIC	LVDS per Channel	Purpose	Total Number
4	1	Data Transfer	24
8	1	Data Frame	12
8	1	Clock	12
4	96	μspills	1
4	96	Acquisition Gate	1

Table 4.2: The number of LVDS lines required for communication between a ROC and a Digitizer.

size is sufficient to provide a 30 ps time resolution over a μspill read time. Each recorded event requires a TDC signal above threshold from at least one straw end. Nominally, timing from both straw ends is recorded, and the Digitizer decides which signals are processed based on stored parameters. The shaping time of the straw anode signal is approximately 120 ns, and the summed analog output from both straw ends is digitized at 50 MHz (20 ns) by an ADC with 8 ENOB. Threshold, gain, and the length of the dead-time after a straw signal are sent by the ROC to the digitizer between super-cycles. This information along with other slow control and status data, are sent over a 3-wire SPI bus.

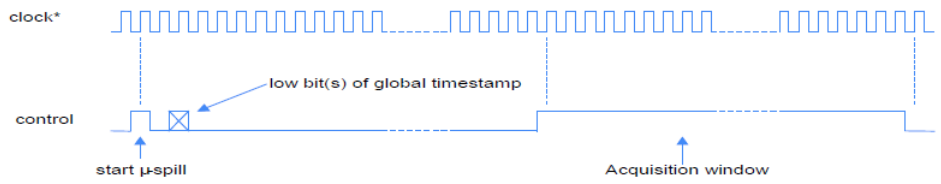


Figure 4.4: A timing diagram for the ROC to digitizer signal transfer. The architecture shows the 200 MHz clock aligned with the μspill and the μspill ID with the data capture gate. The lowest 4 bits of the μspill are transferred on this LVDS line.

4.1.1 Packet Definition

Each event (hit) is composed of a data packet having a fixed length of 128 bits. The Digitizer organizes the data packet and puts it into an output FIFO buffer. The packet has the following structure:

a. 16 bit header - The header contains information to uniquely specify this is a packet header, a channel identifier to specify the channel so the ROC can assign the hit to a wire number, and a packet checksum

b. 16 bit -TDC left straw end

c. 16 bit -TDC right straw end

d. 10 bit ADC

e-j. repeated as d above 7 times with zeros added if the ADC does not contain eight 10 bit words. Data is cut at eight 10 bit ADC words.

At the end of all the data packets of a specific μspill, a final, a 32 bit "end-of-file" packet containing end of μspill information including digitizer status, errors, and the parameters used by the Digitizer to process the signal is inserted, as shown in figure 4.5. If the digitizer has no events during a μspill, only the end-of-file packet is buffered and transmitted to the ROC. This allows the ROC to synchronize and count μspills. The ROC compares the 4 digit μspill number returned by the digitizer in the data packet header with that assigned by the ROC to μspill. This data is re-bundled, including a header, the ROC ID, and status, and transmitted to the TDAQ via an optical data link. The ROC always reports to the TDAQ, adding a "no-data" data packet for each digitizer when "no-data" is present. The global μspill number is received by the ROC from the TDAQ which is locally stored. The ROC extracts the lowest 4 bits of this number, and transmits these to the digitizers. The digitizers in term transmit these bits back to the ROC at the end of the μspill data packet as a synchronization check.

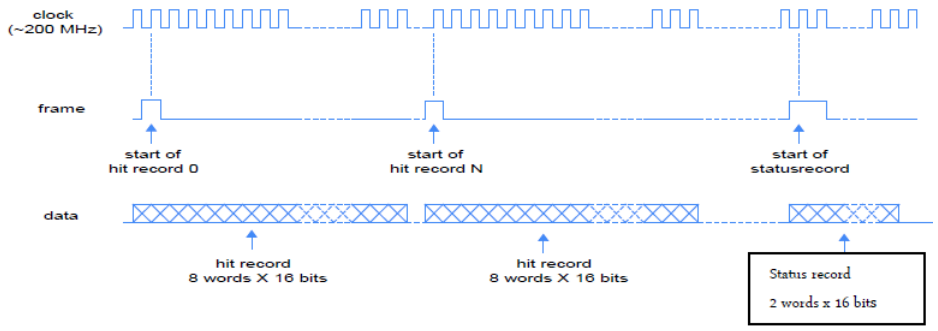


Figure 4.5: timing diagram for the digitizer to ROC data transfer. The architecture allows transfer across a μspill. The example shows 2 buffered μspills, and an end-of- μspill data transfer.

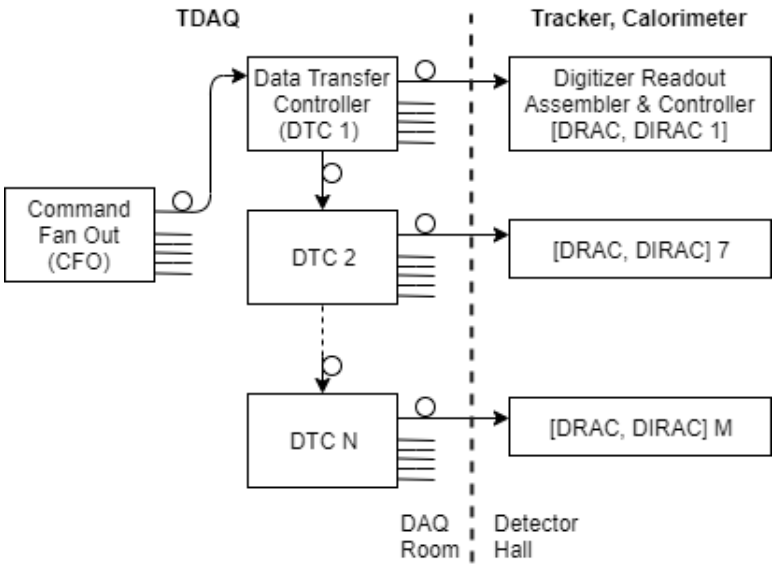


Figure 4.6: Timing tree: in this point-to-point topology, timing is merged with data and the clock is fanned out by the CFO.

4.1.2 Data Rate

At 200 MHz, a packet of 128 bits can be transferred every 640 ns. An additional 32 bits must be added as an end-of-file marker after the data μspill hit data is buffered. Thus on average 2.4 data packets plus an end-of-file ($[1695ns - 160ns]/640ns = 2.4$) can be transferred to the ROC during a μspill. Single rates in the straws are balanced in front of the Digitizer to obtain an average singles rate of 150-200 kHz per straw for a panel.

4.2 Clock

The TDAQ will generate a continuous Mu2e System Clock at 40 Mhz at the Run Control Host Clock Fanout module (CFO). The CFO receives the RF-cavity 0-crossing marker from the Accelerator, which is synchronous with the arrival of proton pulses every 1695 ns and it is always at a fixed phase. The CFO outputs an encoded clock indicating the start of the Mu2e event window, which is synchronous with the Accelerator marker. This encoded clock will be fanned out and distributed to outside the cryostat in the detector hall. Figure 4.6 shows the timing tree. This System Clock is not guaranteed to arrive to all detector ROCs phase aligned. Phase alignment at individual ROCs is accomplished by an adjustable delay at the ROC clock input. The CFO outputs also the start-of-event-window marker and an Heartbeat packet to specify the Event Window.

Tracker and Calorimeter are accessed by a single bi-directional VTRx at the ROC, so the System Clock and event marker will be transmitted over the serial link. All timestamping frontends extract a System Clock and start of Event Window marker. The trick is synchronizing the resulting Event Windows, i.e. synchronizing the timestamp reset to T=0, across all timestamping front ends. T=0 defines the start of a Mu2e Event Window and occurs in response to each start-of-event-window marker. Event Window is define at CFO. Signals travel to different boards and through fibers and cables of different lengths and they must

line up the Event Window at timestamping front ends. To line up Event Windows the approach is to delay each front end to match front end with longest latency. Delay is determined calculating the Loopback signal. It consists in determine the round trip time by sending a marker and returning it many times and taking the average time. Once the timing path latency is known for each front end, it is possible to calculate each front end's delay offset define as "longest timing path latency" minus "individual timing path". At the start of each run,the front ends can be configured to apply their own delta delay. The idea is a two register approach: a Delay Event Window Start register, which uses units of timestamping clocks for coarse adjustment to delay the start of the Event Window, i.e. when the timestamp is reset to T=0, and a Clock Phase Offset register, which uses units of "ticks" for fine phase adjustment of timestamping clock. Usually a "tick" represents $\sim$ 200ps.

All timestamping front ends label their digitized data with a timestamp to indicate when that data was sampled. Data sampled at the same moment in time, in two different timestamping frontends of the same subsystem, must be labeled with identical timestamps. Each subsystem's front ends may have a different sampling frequency, i.e. different timestamp granularity which involves different timestamp incrementing frequency. Tracker run at 200 MHz and Calorimeter at 280 MHz, so the timestamp frequency need to be an integer multiple of the Mu2e System Clock at 40 MHz, otherwise T=0 may not exactly lineup.

A clock receiver/driver/conditioner IC on the ROC PCB receives a 200 MHz System Clock with a 590 kHz μspill identifier from the TDAQ. The conditioner aligns the 200 MHz signal edge with the μspill ID and distributes both signals to the ROC and the digitizers. The TDAQ clock is presented to each receiver over two dedicated differential lines with optical isolation. The dual transfer is used for redundancy. The clock signals to the ROCs come from a clock distribution system external to the Tracker. The nominal clock jitter specification is 200 ps. The clock interface consists of the following:

a) Two dedicated differential clock pairs are input to the ROC at 200 MHz, with optical isolation upstream in the clock distribution network. The second clock pair is included for fault tolerance.

b) A dedicated jitter cleaner PLL device, with at least 13 differential output pairs. Jitter on each pair must be less than 200 ps RMS. Absolute clock skew across channels is not critical, because the effect can be calibrated, but clock skew over time should be stable and consistent across power cycling events. The proposed clock conditioning IC is Texas Instruments LMK04800. It has jitter specifications of 150 fs RMS from 100 Hz to 20 MHz and has programmable delay on each of the 14 output channels.

Timing between different ROCs must be phase aligned. Input phase alignment can be obtained by matching the PCB trace lengths between all output lengths as closely as possible, by compensating for the mismatched delays using programmable delays in the ROC inputs, and/or programmed delay in the clock driver.

Each digitizer re-generates its own TDC, ADC, and data-transfer clock from the 200 MHz sent by the ROC. Only the μspill identifier must be common and phase aligned (compensated) between all ROCs. Other internal clocks from

various digitizers do not have to be aligned. Data transfer is gated by a framing window tied to the transfer clock as described above. The gate closes when all the data associated with a μspill is transferred. Global phase alignment between different ROCs may be obtained by calibration of the arrival times of the μspill at the different ROC.

4.3 DDR3 Memory

The Mu2e Tracker readout board, DRAC, has a 6 Gb DDR3 memory (3x2 Gb IS46TR16128A DDR3 external memories [9]) serving as a buffer for the Tracker hits information until the TDAQ comes to retrieve the data. The DDR3 memory is serviced by the ROC PolarFire FPGA. These memories have JEDEC speed grade DDR3-1333H, i.e. able to work with 6 ns clock: DDR3 allows to transfer 8 bits of data on every edge of a clock running at 4x the clock speed.

4.3.1 Tracker Hit Data

As stablished in the DRAC DDR3 Readout document [22], each Tracker hit comprises some geographical information to uniquely identify the straw, the time of the pulses of the two end of the straw from the start of the μspill, the time over threshold (TOT) for the two pulses and a variable number of 12 bit ADC samples for the sum of the two pulses. This data is passed via 4 SERDES lanes using 10 to 8 bit encoding, hence the natural width of the incoming data is 32 bits. We assume that only 16 of the 32 bits are used in the following and refer to this as DIGI word. The proposed definition of data received from the DIGI FPGAs is in table 4.3.

The length of the hit is required to be in multiples of 64-bit DDR3 words, which is the natural width of R/W to DDR3 memory. DIGI words 0 to 3 will be mapped to bit [15:0], [31:16], [47:32] and [63:48] of DDR3 words 0, respectively. Up to 1020 ADC samples are allowed but in normal running operations, we expect to save a fixed "standard" number of 12 ADC samples, for a total of 20xDIGI words and 5xDDR3 words per hit. The μspill number coming with the DTC Data Packet Request is used to allocate the starting address of 1 KB blocks of DDR3 memory space. Only the 4 LBS of the μspill number are stored in DIGI word 4. The number of ADC samples saved for the hit can be between 12 and 1020 in steps of 4, so that the end-of-hit word is at the boundaries of a DDR3 word. The actual ADC count for a given sample fills bit[11:0] of the DIGI word while the 4 MSB contain a rotating counter to help in the detection of missed words. The hit trailer starts with the 0xEFEF end-of-hit word, followed by the end-of-spill padding word, set to be 0xD01E for a typical hit and 0xD0FE for the last hit of the spill. The very last DIGI word repeats the μspill number for redundancy and contain info on the status of the digitizer and an error code. The digitizer status is envisioned to contain info on the kind of data being passed (physics vs TRK vs TDAQ diagnostic, on spill vs off spill, etc). A summary of the error codes for a given spill will be passed to the DTC. The ROC memory manager firmware will take care of calculating the total number of hits for a given spill, as well as removing the number of ADC sample words and the hit trailer, effectively writing to memory 4xDDR3 words per standard hit. The ROC DTC interface firmware will further compact the ADC samples bits to fit the 16x16-bit DTC Tracker data

DDR3 word	DIGI word	DIGI Word Content
0	0	[15]=1/0 for non/standard (i.e. 12) number of ADC samples [14:0] = straw index (0 thru 23039)
	1	HV-end TDC sample [15:0]
	2	CAL-end TDC sample [15:0]
	3	[15:8] CAL-end TOT; [7:0] HV-end TOT
1	4	[15:12] μspill number; [11:0] Number of ADC samples (S)
	5	[15:12] 0x0; [11:0] ADC sample 0
	6	[15:12] 0x1; [11:0] ADC sample 1
	7	[15:12] 0x2; [11:0] ADC sample 2
2	8	[15:12] 0x3; [11:0] ADC sample 3
	9	[15:12] 0x4; [11:0] ADC sample 4
	10	[15:12] 0x5; [11:0] ADC sample 5
	11	[15:12] 0x6; [11:0] ADC sample 6
3	12	[15:12] 0x7; [11:0] ADC sample 7
	13	[15:12] 0x8; [11:0] ADC sample 8
	14	[15:12] 0x9; [11:0] ADC sample 9
	15	[15:12] 0xA; [11:0] ADC sample 10
4	16	[15:12] 0xB; [11:0] ADC sample 11
...	...	...
M	4*M	[15:12] 0x(4M%16); [11:0] ADC sample 4*M
	4*M+1	0xEFEF End of hit word
	4*M+2	[15:8] = 0xD0; [7:0] = 0xFE/0x1E for last/not last hit in spill
	4*M+3	[15:12] = 0xD; [11:8] = μspill number; [7:4] Digitizer Status; [3:0] Error Code

Table 4.3: Proposed definition of the Tracker hit data: M-1 is the number of 64-bit words to be written to DDR3 memory, equal to the number of ADC samples plus the 4 header and TDC words, divided by 4.

DDR3 word	DTC word	Content
0[63:48]	0	[15]=1/0 for not-last/last packet of the hit; [14:0] = straw index (0 thru 23039)
0[47:32]	1	[15:0] HV-end TDC sample
0[31:16]	2	[15:0] CAL-end TDC sample
0[15:0]	3	[15:8] CAL-end TOT; [7:0] HV-end TOT
1[63:48]	4	[15:12] ADC Sample 1; [11:0] ADC sample 0
1[47:32]	5	[15:8] ADC Sample 2; [7:0] ADC Sample 1
1[31:16]	6	[15:4] ADC Sample 3; [3:0] ADC Sample 2
1[15:0]	7	[15:12] ADC Sample 5; [11:0] ADC sample 4
2[63:48]	8	[15:8] ADC Sample 6; [7:0] ADC Sample 5
2[47:32]	9	[15:4] ADC Sample 7; [3:0] ADC Sample 6
2[31:16]	10	[15:12] ADC Sample 9; [11:0] ADC sample 8
2[15:0]	11	[15:8] ADC Sample 10; [7:0] ADC Sample 9
3[63:48]	12	[15:4] ADC Sample 11; [3:0] ADC Sample 10
3[47:32]	13	[15:12] ADC Sample 13; [11:0] ADC sample 12
3[31:16]	14	[15:8] ADC Sample 14; [7:0] ADC Sample 13
3[15:0]	15	[15:8] Preprocessing(8 bits); [7:4] Reserved; [3:0] ADC Sample 14

Table 4.4: Tracker data packet definition

packet reported in table 4.4. If more than 15 ADC samples are used, bit [15] of DTC word 0 will be set high as long as the last DTC packet for that hit is being sent. The remaining DTC words 0 to 3 will be copied for each data packet, and the unused ADC sample words will be passed as zeros.

4.3.2 Memory Protocol

Packed data are streamed from DIGI and CAL FPGA to the ROC FPGA. ROC firmware handles storage in the DDR3 memory. The PolarFire FPGA DDR subsystem [16] is made up of the following soft and hard blocks:

- DDR controller (soft)
- Training logic (soft)
- I/O lane (hard)
- Phase-locked loop (PLL) (hard)

All these blocks are implemented to handles the data buffering data. The DDR subsystem accepts read and write commands using the AMBA Advanced Extensible Interface 4 (AXI3/4) and translates the AXI/native interface requests into command sequences required by SDRAM devices. The DDR controller

module then issues these commands to the PHY module, which sends and receives data to and from the DDR SDRAM. It can also automatically perform DDR initialization, refresh, and ZQ calibration functions.

The DDR controller is a soft IP core that consists of the following blocks:

- Control and timing block — contains the main controller logic.
- Initialization control block — performs the initialization sequence after system reset (RESET_N) is deactivated or when dynamic reinitialization control (CTRLR_INIT) is pulsed.
- Bank management block — keeps track of the last opened row and bank to minimize command overhead.
- Refresh/ZQ calibration control block — performs automatic refresh/ZQ calibration commands to maintain data integrity.
- Queue control block — allows new requests to be accepted on every clock cycle as long as the queue is not full.
- Data control block — handles multiplexing and de-multiplexing of data flowing to and from the DDR SDRAM devices.
- Multi-burst block — allows requests longer than the programmed memory burst length. Also handles requests with starting addresses not aligned on a burst boundary, breaking the requests up as necessary to prevent data wrapping. The queue-based implementation of the core's interface enables the DDR controller to optimize throughput and efficiency by looking ahead into the queue to perform precharges before the read and write commands are issued.

The DDR subsystem requires a dedicated PLL to generate the clocks, which are then distributed throughout the subsystem using HS_IO_CLK routes, dedicated pads, and fabric clock routing. This PLL generates aligned clocks for all sub-blocks for smooth operation and synchronous communication with the user logic. The PLL generates the following required clocks:

- REF_CLK — This clock is routed to the PHY for clocking the DDR memory device.
- HS_IO_CLK — This clock routed to I/O lanes and the training logic.
- HS_IO_CLK_270 — HS_IO_CLK phase shifted by 270. This clock is also routed to I/O lanes and the training logic.
- YS_CLK— This clock is routed to the DDR controller, training logic, and user logic in the fabric .

The HS_IO_CLK and REF_CLK clocks are generated with the same frequency and phase. The REF_CLK to SYS_CLK ratio is 4:1.

Block diagram in figure 4.7 shows how the DDR Interface is embedded on the DRAC firmware. The diagram highlights the AXI Interface and the DDR3 Controller itself and how they are served by FIFOs. Data are provided by DIGI FIFO and converted from 32 to 64 bit, that is the DDR3 word length. Data

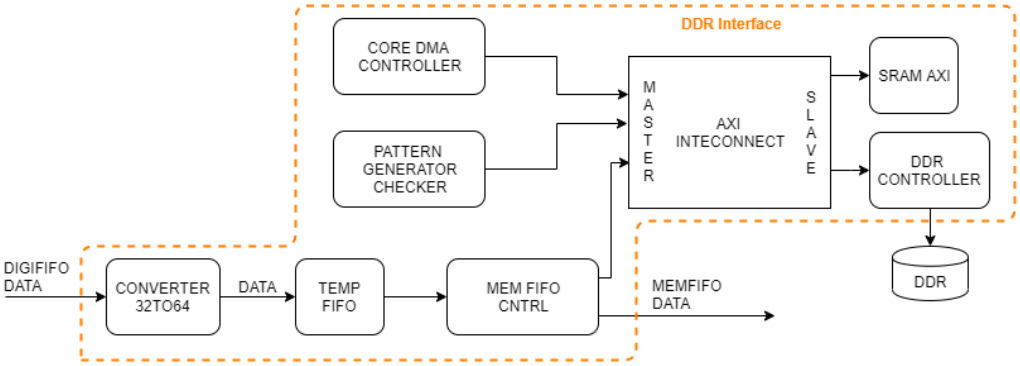


Figure 4.7: Block diagram of DDR Interface. This part of the firmware handles the DDR3 memory both for writing and reading and for the management of the DDR3 itself. It uses an AXI protocol.

are stored in the TEMP FIFO with the purpose of reaching depth of a page to efficiently use the DDR3 writing. Pages are 1 KByte long. Firmware is still developing. In this beta version, the DDR3 write enable is high when the TEMP FIFO is full. This simplification made possible estimate writing and reading speed and validate the hardware. This implementation provide two blocks: the Pattern Generator Checker and the MEM FIFO Control. The first one generate a series of known pattern to test the memory. One of the test pattern is the counter: using a counter helps detect errors on consequential reading and writing. The MEM FIFO Control is responsible to determine data path. The complete data path starts from the DIGI and CAL FPGAs, goes through the TEMP FIFO, is stored in the DDR3, read from the DDR3 and written in the MEM FIFO and then sent to the SERDES block to be interfaced with the TDAQ system. Alternative data path are the ones which excludes the DDR3: it is possible to retrieve data from DIGI and CAL FPGAs and store them in TEMP FIFO first and in MEM FIFO then. This path obviously is necessary to be sure about the data content without involving complications due to DDR3 management. Another path involves the Pattern Generator. In this case not the data content but the DDR3 management is tested. Patterns are generated and written and read to and from the DDR3, and then follow the same path as data. The last path involves a SRAM. This path is used to test the AXI interface avoiding the DDR3 and the DDR3 Controller. These path are select by the MEM FIFO Control wich present an interface to the microprocessor instantiated in this firmware.

DDR Interface is a part of the ROC firmware that must manage both data storing, data transmission and control operations. Figure 4.8 shows how DDR Interface is embedded inside the DRAC ROC Firmware. TOP SERDES manages communication between the Trigger and Data Acquisition (TDAQ) and the Mu2e detector subsystem Readout Controllers (ROCs). When Data Transfer Controller (DTC) asks for Data Packet, TOP SERDES handles the request and set controls for memory reading. DIGI Interface receives digitized time of the hits stored inside the two PolarFire FPGAs DIGI HV and DIGI CAL. They are connected to the Readout Controller via four 5 Gbps SERDES lanes. Insiede the block DIGI Interface, two SerDes blocks handles the 5 Gbps SERDES standard and data are stored in two different FIFO: DIGI FIFO and DAQ FIFO. The first one handles data in 32 bit and is connected to the DDR switch, the second one in 16 bit and is

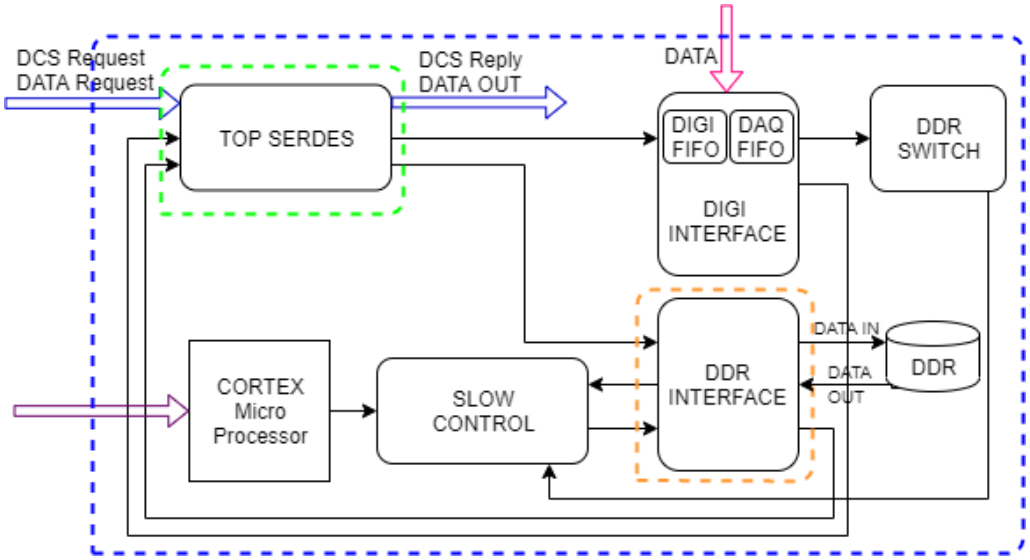


Figure 4.8: Block diagram of the DRAC Firmware. In blue is surrounded what is individuated as ROC Firmware: the main component are the DDR Interface and TOP SERDES.One handles the memory and the other manages commands. DIGI Interface handles data throughput from two PolarFire FPGAs DIGI HV and DIGI CAL and the CORTEX MicroProcessor provide user access and control.

connected to TOP SERDES. This duplicity needs as debug mode: create multiple paths allows to test functionality of the several blocks bypassing some of them. DDR Switch select which between data from DIGI FIFO or from MEM FIFO send to the DDR Interface to be written in memory. Also this represents a debug path. The firmware provides two other important block: the microprocessor and the Slow Control.

4.4 Microprocessor

The DRAC firmware provides an embedded microprocessor. It is instantiate with a support scope: when the system was at the beginning of his development, the microprocessor was necessary to easily interface with the FPGA. ROC has tow main port to the outside: one to the DTCs, and by extension the TDAQ, and one to the digitizers providing data. Data are simulated and will be simulated until the end, when the board will be mounted in the detector in the experiment. Data simulation can be reached via the microprocessor. The easy solution is writing a program which feed the data path with simulated data. On the other side, data request from the DTC was inaccessible until the DTC firmware was almost complete. Also in this case, simulated request was possible via the microprocessor. Once the system will be complete, there will be no need for the microprocessor. At this level of development, microprocessor is still useful for debug and it is under discussion the possibility to maintain it in the firmware for further uses.

Microsemi PolarFire FPGAs support Cortex-M1 soft processors that can be used to run user applications and it is possible to build it using the Libero SoC design suite [14].

The user application can be stored in the sNVM, uPROM or SPI Flash. In this firmware the user application is stored in sNVM. At device power-up, the PolarFire

System Controller initializes the designated LSRAMs with the user application from sNVM and releases the system reset. The Cortex-M1 soft processor exits the reset and starts executing the application. During the Libero design flow, the required non-volatile memory must be specified for the fabric RAMs initialization. Then, the Fabric RAM initialization client must be created. The created fabric RAMs initialization client is stored in the sNVM. Once the IP is instantiated and synthesized, before the place and route step is necessary to define the input and output attribute. A peculiar step is the Configure Design Initialization Data and Memories. This process requires the user application executable file (hex file) as input to initialize the LSRAM blocks after device power-up. The hex file is provided along with the design files. When the hex file is imported, a memory initialization client is generated for LSRAM blocks. The Cortex-M1 is suitable even via SoftConsole. The empty Cortex-M1 project requires the hardware abstraction layer (HAL) files, Cortex microcontroller software interface standard (CMSIS) files, and the following peripheral drivers: CoreGPIO and CoreUARTapb. Once downloaded and installed is possible to write a C++ program running on the Cortex-M1. Right now on the microprocessor in the DRAC firmware is compiled a program able to index the MEM FIFO Control. It allows to select different data path and data pattern. It stamps data read from the DDR3 to confront them with what is expected. It allows to predict the correct or not behavior.

Chapter 5

DIRAC - DIgitizer ReAdout Controller

The DIRAC board is the Mu2e Calorimeter readout controller board (figure 5.1). Also referred as Waveform Digitizer subsystem (WD), it is an electronic printed circuit board that digitizes analog data, serializes it and sends it upstream to the TDAQ via a fiber optic transceiver. The WD must also perform some digital signal processing (DSP) operations, removing data below threshold (zero suppression) as well as providing the mean charge and time for each channel by means of running averages.

To limit the number of pass-through connections the DIRAC will be hosted inside the cryostat as shown in figure 5.2. This is a very harsh environment and requires to follow special design rules. In particular [8]:

- The boards will be operated in vacuum (10-4 torr) and they will be quite difficult to be serviced (the DS will be opened no more than once per year). This imply to follow in the design high reliability rules, commonly used in space applications.

- The thermal dissipation will be mainly for conduction, so the mechanics of the board (and of the supporting brackets or crates) and the PCB will are designed to keep account of this. The design follows extremely low power requirements but nevertheless an appropriate cooling system has been designed.

- The environment will be radioactive, both due to neutrons and ionizing particles. The boards are specified to sustain a TID of 15 Krads and a NIEL 1 (Mev eq) of around 2x1011 n/cm2 over the data taking period (safety factors not included).

- Inside the cryostat, a strong magnetic field is present. The boards are specified to sustain a magnetic field of 1 Tesla. This implies that at the component level only air wounded inductors will be allowed and a special care will be needed in the choice of the DC-DC converters.

The design is based on a SoC component (FPGA + CPU integrated in the same package) belonging to the Microsemi Polarfire family, model MPF300TS-1FG1152 [10]. It combines up to 481K logic elements consisting of a 4-input look-up table (LUT) with a fractureable D-type flipflop with 64 x 12 two-port μRAM block

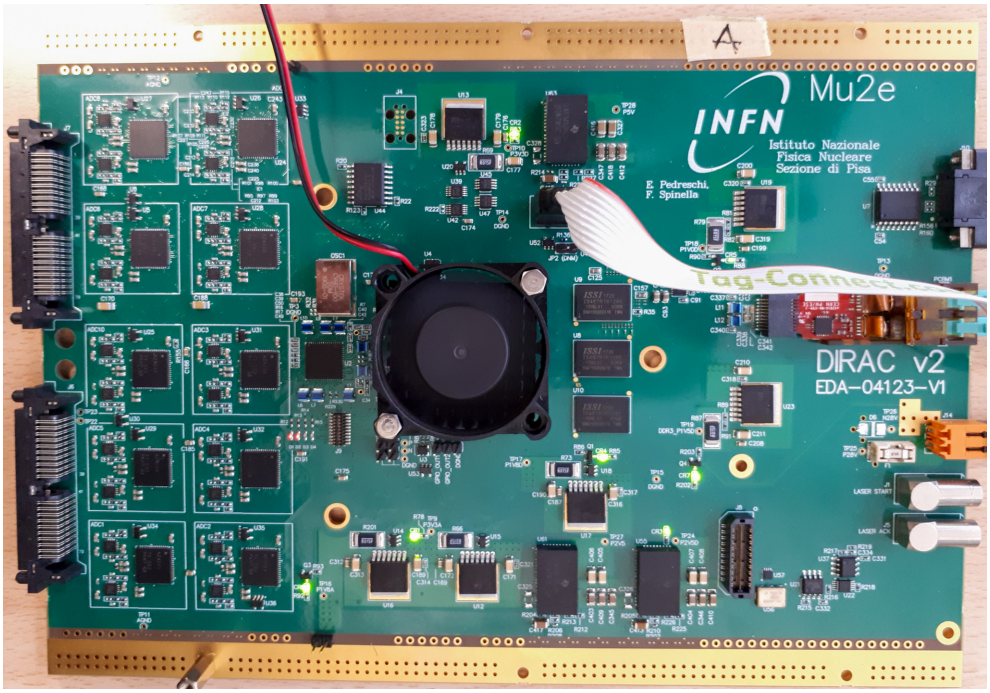


Figure 5.1: DIRAC Board

implemented as an array of latches 18 x 18 math block with a pre-adder, a 48-bit accumulator, and an optional 16 deep x 18 coefficient ROM. It presents a built-in μPROM, modifiable at program time, readable at run time for user data storage and an high-speed serial connectivity with built-in multi-gigabit multi-protocol transceivers from 250 Mbps to 12.7 Gbps. It mounts an high-speed I/O (HSIO) supporting up to 1333 Mbps DDR3 memory with integrated I/O digital.

The ADC has been selected to be the ADS4229 from Texas Instruments. It converts with a maximum sample rate up to 250 MHz with 12 bits of resolution. The guiding parameters were the relative low cost, in comparison to the competitors, and the extreme low power. Each ADS4229 includes 2 converters, so a total of 10 parts are needed for each board.

Microsemi offers the Microsemi Libero System-on-Chip (SoC) design suite comprehensive of development tools for designing with flash FPGAs, SoC FPGAs, and Rad-Tolerant FPGAs. The suite integrates industry standard Synopsys Synplify Pro synthesis [13] and Mentor Graphics ModelSim simulation with constraints management, debug capabilities, and secure production programming support. The Libero SoC v12.0 release supports SmartFusion2, IGLOO2, RTG4, and PolarFire devices.

5.1 The Digitization system

The calorimeter is composed of 1348 crystals, each equipped with 2 arrays of SiPMs, for a total of 2696 fast analog signals that must be digitized after being amplified and shaped by the front-end electronics (FEE).

The overall scheme for the Calorimeter readout electronics is shown in figure 5.3. The front-end electronics (FEE) consists of two discrete and independent chips (Amp-HV), for each crystal, directly connected to the back of the SiPM pins.

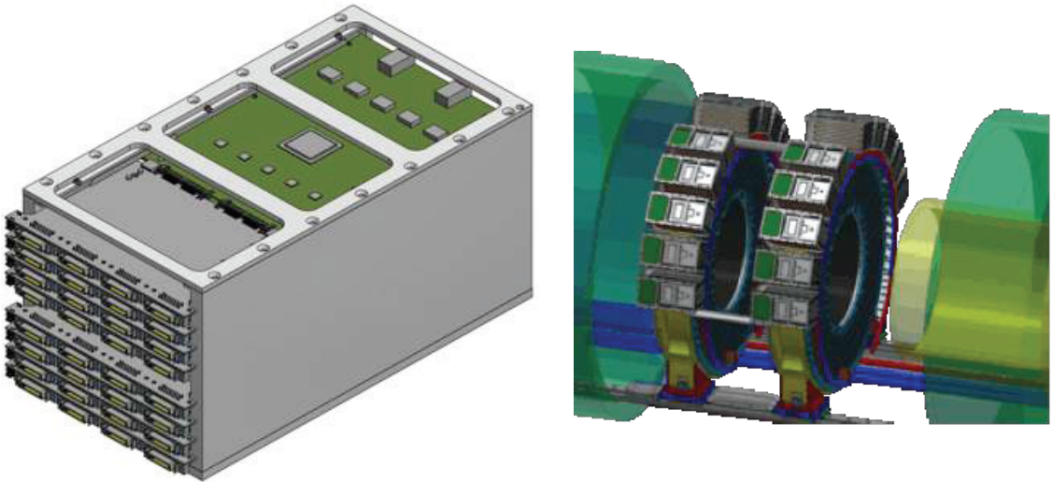


Figure 5.2: CAD drawing of the crate hosting boards and drawing of crate inside the Calorimeter.

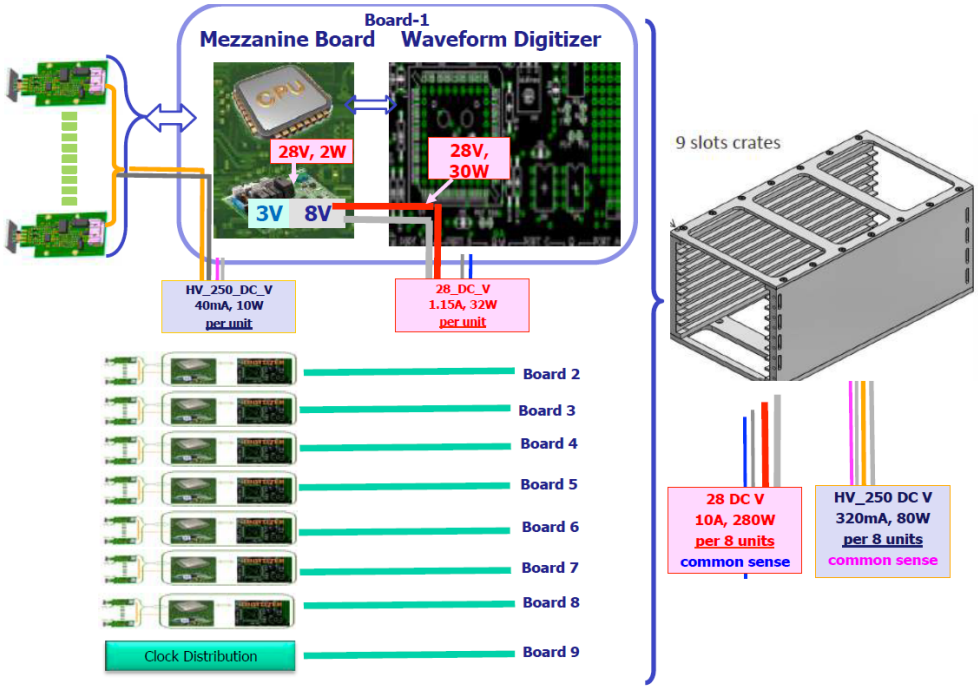


Figure 5.3: Overall schematic of the calorimeter electronics. The drawing represents the distribution of the electronics and the connections to one crate. Each crate can host up to 8 MB+WD boards and 1 clock distribution board.

These provide the amplification and shaping stage, a local linear regulation of the bias voltage, the monitoring of current and temperature on the sensors and a pulse test. Each disk is subdivided into 34 similar azimuthal sectors of 20 crystals. Groups of 20 Amp-HV chips are controlled by a dedicated mezzanine board (MB), where an ARM controller distributes the LV and the HV reference values, while setting and reading back the locally regulated voltages. Groups of 20 signals are sent in differential way to a digitizer module (WD) where they are sampled and processed before being optically transferred to the DAQ system. The Detector Control System parameters read out/set by the MB are passed via I2C to the WD that then communicates them to the DCS system through an optical link.

The Mezzanine board (MB) has many tasks to fulfill but in general is the interface between the FEE chips and the Digitizer board while performing all the needed Detector Control System steps. The MB should: receive in input the 20 differential FEE signals and provide them to the digitizer board, set all the FEE parameters such as HV, preamplifier gain and pulse register, monitor HV and SiPM temperature and current, provide the low voltage for the FEE. All the DCS information are handled by a CPU system architecture, consisting of a series of ARM processors. The ARM processor controls each of the 20 connected Amp-HV cards. In the NIM MB Prototype all these operations are done through firmware in the CPU via a standard Ethernet connection.

The 2696 SiPMs will be readout through 136 WD boards, each supporting 20 channels. Each card is 233 x 160 mm long, with the FEE connectors located on the short edge. Half of the cards reads one of the disk. The cards will be hosted on 10 crates for each disk, each comprising up to 9 slots, 6 or 8 MB+WD boards, while the 9th slot is reserved for the clock distributor.

5.2 Test Stand

The Calorimeter has been designed and will be constructed by the collaboration among the Italian National Institute of Nuclear Physics (INFN), the California Institute of Technology (Caltech) and the Fermi National Accelerator Laboratory (FNAL or Fermilab). Part of the work was rebuilding the Test Stand situated at the FNAL Feynman Computing Center (FCC) in the INFN Laboratory. Test Stand is the environment which reproduce the TDAQ system.

First, the TDAQ Server has been arranged. A single TDAQ Server can be used as a complete standalone data acquisition/processing system or multiple TDAQ Servers can be connected together to form a highly scalable system. In the reproduction of the system, all environmental requirements are not tightening. The Server is based on a ASUS Z97 - K motherboard with 1 TB Hard Disk (WD10EZRX-00L4HB0) and 120 GB SSD Disk (Samsung SSD 840 EVO). On the Server is installed Scientific Linux 7.7.

Scientific Linux is an Enterprise Linux rebuild sponsored by Fermi National Accelerator Laboratory. It provides a stable, scalable, and extensible operating system for scientific computing and supports scientific research by providing methods and procedures for enabling the integration of scientific applications with the operating environment. Scientific Linux is a rebuild of Red Hat Enterprise Linux (property of Red Hat Inc NYSE:RHT).

On Scientific Linux are installed all packages and repository to start *artdaq*. *artdaq* is a toolkit of C++ 2011 libraries and programs for use in the construction

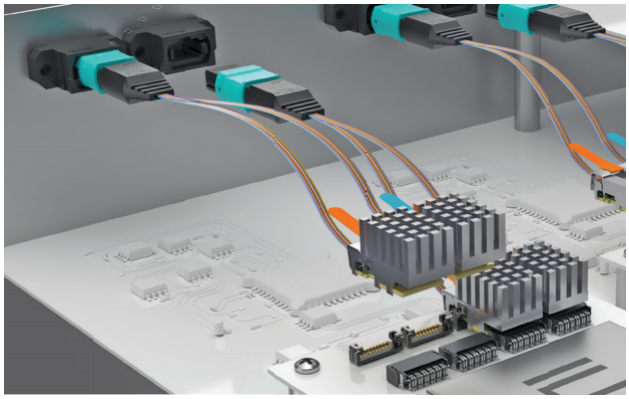


Figure 5.4: FireFly Micro Flyover System by Samtec with the MPO end.

of TDAQ systems; it provides software applications for managing the data flow as well as libraries and applications for encapsulating the data, analyzing the data, and performing other basic data acquisition functions. On *artdaq* runs *otsdaq*. *otsdaq* uses the *artdaq* DAQ framework under-the-hood to provide data handling exibility and scalability. It is the online DAQ software framework and provides a web interface to configure, control, and monitor the online DAQ software entities. *otsdaq* and *artdaq* are developed by the Fermilab Scientific Computing Division.

otsdaq interfaces to the Data Transfer Controller (DTC).The DTC module provides an interface between the Mu2e Readout Controller (ROC) modules, and the Trigger and Data Acquisition (TDAQ) servers running the TDAQ online software framework. The DTC is implemented using a commercial PCIe (Peripheral Component Interconnect Express) card located on the motherboard of the TDAQ Server. It is based on the HiTech Global Kintex-7 (HTG-K700) PCI Express expansion card. This card features an eight lane Gen 2 PCI Express interface, a DDR3 SODIMM socket, and a 400 pin FMC connector, all wired to a Xilinx K325T Kintex-7 FPGA. The FMC connector allows the installation of an FMC card with the optical fiber interface. This card is called the Timing Card and mounts the FireFly Micro Flyover System by Samtec [30] (figure 5.4).

FireFly is the an interconnect system that allows the flexibility of using micro footprint high-performance optical interconnects with the same connector system of low-cost copper. FireFly optical cable systems provide the flexibility to achieve higher data rates to 28 Gbps and/or greater distances, simplifying board design and enhancing performance. On the timing card the ECUO Series is mounted. It allows up to 28 Gbps per channel via optical cable for greater reach. It is been chosen because the footprint allows for higher density close to the data source and because of the simple use system with easy insertion/removal and trace routing, no through-holes, and surface mount connector system. Fibers are OM3 multi-mode with 12 fibers. The end of the cable is a MPO (MTP) high-density connector for panel applications and minimal keep-out areas on the board. Timing card with the Firefly modules provides for multi-gigabit serial links for up to six ROC Links, a port for data exchange for hardware event building, and a port for the Command Fan-Out (CFO) interface.

The optical fibers are OM3 multimode. “OM” stands for the minimum Modal Bandwidth (MBW) requirement. OM1, OM2, and OM3 are determined by the ISO 11801 standard, which is based on the modal bandwidth of the multi-mode

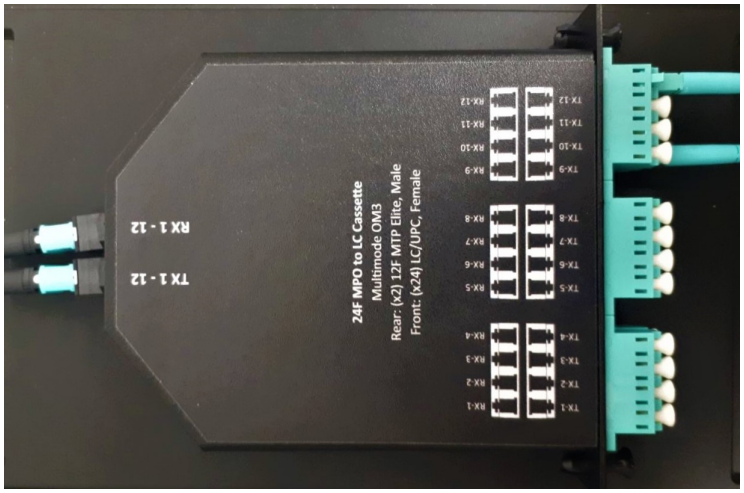


Figure 5.5: Fiber cassette: it is the optical splitter from 12(x2) MTP Male to 24 LC/UPC Female.

Cassette Channel	**TX**	**RX**
1	Unused	CFO Downstream
2	CFO Upstream	Unused
3	ROC 5	ROC 5
4	ROC 3	Unused
5	CFO Downstream	CFO Upstream
6	ROC 2	ROC 1
7	ROC 4	ROC 4
8	Unused	EVB
9	ROC 0	ROC 3
10	ROC 1	Unused
11	EVB	ROC 2
12	Unused	ROC 0

Table 5.1: Mapping of channel numbers on the fiber cassette.

fiber. OM3 has a suggested jacket color of aqua, its core size is 50 μm. It supports 10 Gigabit Ethernet at lengths up to 300 meters. OM3 fiber specifies an 850-nm laser-optimized 50-micron fiber with an effective modal bandwidth (EMB) of 2000 MHz/km. It can support 100-Gbps link distances up to 100 meters.

From the Firefly module gets out two MTP cables (one for transmission and one for reception) of 24 optical fibers (12 for transmission and 12 for reception) which needs to be split in 24 LC/UPC cable. Splitting is finalized by the optical divider in figure 5.5. Channel numbers on the fiber cassette are not mapped 1 to 1 with the DTC channels. The mapping is reported in table 5.1. As looking at the front of the cassette, TX connections are on the left and RX connections are on the right. Cassette channels are numbered 1 to 12 (not 0 to 11), with cassette channel number 1 on the top for both TX and the RX cassette connections.

Optical fibers are connected to the ROCs. In this Test Stand ROC is linked on channel 0, so the connection is TX (side DTC) on 9 and RX on 12.

For the first test of this Test Stand, the ROC on the DIRAC is simulated on

Figure 5.6: On the left: the HW-USB-II-G Xilinx programmer with JTAG connection. On the right: the Finisar FTLX8574D3BCV (50 nm VCSEL 1G/10G Dual-Rate) used as bi-directional communication module between DTC and ROC.

the AVMPF300TS-20-NA Evaluation Board from Microsemi [19]. It supports the same Microsemi PolarFire FPGA (MPF300TS-1FG1152) that is on the DIRAC. In the operational mode, the front-end component that enable the fibers connection to the ROC is a bi-directional module composed of both optical transmitter and receiver: the Versatile Transceiver (VTRx) [33] developed by CERN. In this Test Stand, the VTRx is not available, so it is replaced by a Finisar FTLX8574D3BCV (50 nm VCSEL 1G/10G Dual-Rate) in figure 5.6. The Finisar connector has an hot-pluggable SFP+ footprint compatible with the AVMPF300TS-20-NA. The external power supply is set to 3.3 Volts to power the SFP+, and not 2.5 Volts like the VTRx requires.

DTC Firmware

To enable communication between DTC and ROC, DTC needs to pe programmed. It has Xilinx K325T Kintex-7 FPGA on board, so it uses Xilinx tools for development, programming, test and debug. Program and debug is performed via Vivado Design Suite. In particular it uses the Integrated Logic Analyzer (ILA) included in the Vivado Suite. A Xilinx programmer with a JTAG connection is necessary: in this Test Stand it has been used a HW-USB-II-G as shown in figure 5.6.

Programming is performed by configuring the memory device and using it to load firmware on the FPGA. Every time the DTC is powered off, firmware is not lost because it is retrieved by memory. Device part number is$mt28gu01gaax1e - bpi - x16$ and the firmware is an .mcs file. A power cycle of the Server loads the FPGA and allows the BIOS to configure the PCI connection.

The customizable Integrated Logic Analyzer (ILA) IP core provided by Vivado [37] is a logic analyzer core that can be used to monitor the internal signals of a design. The ILA core includes many advanced features of modern logic analyzers, including Boolean trigger equations, and edge transition triggers. Because the ILA core is synchronous to the design being monitored, all design clock constraints that are applied to your design are also applied to the components inside the ILA

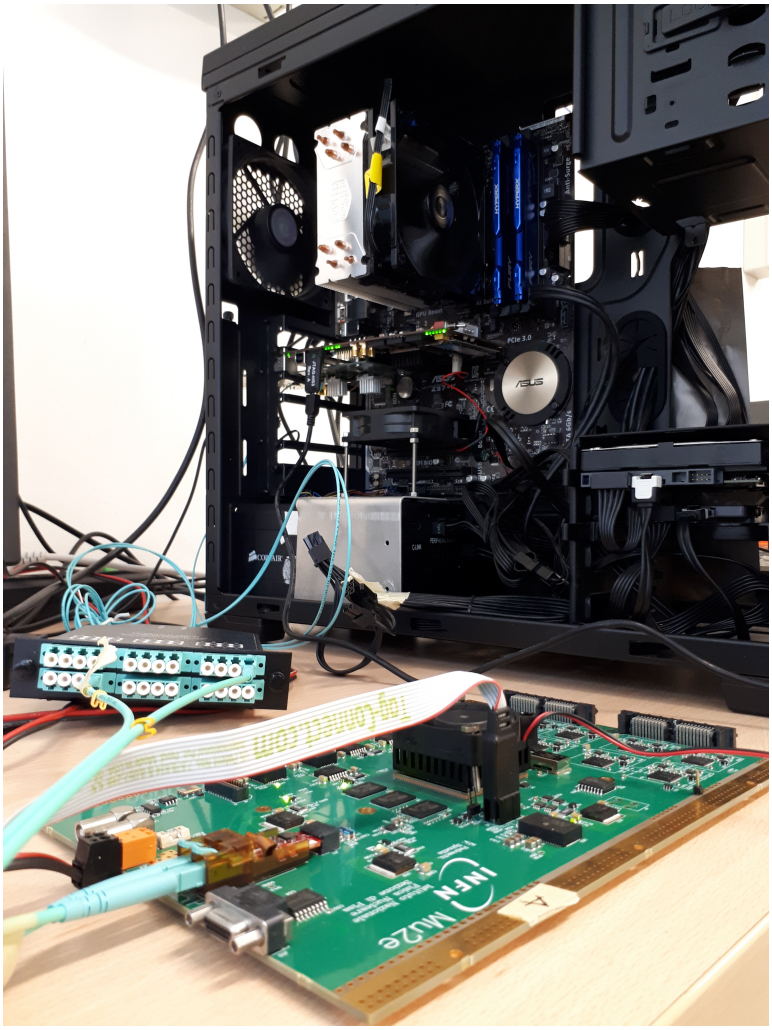


Figure 5.7: Part of the Test Stand replicated in Italy. In figure: the server with the DTC mounted inside, the DIRAC board and the fibers multiplexer.

core.

Signals in the FPGA design are connected to ILA core clock and probe inputs. These signals, attached to the probe inputs, are sampled at design speeds and stored using on-chip block RAM (BRAM). The core parameters specify the number of probes, trace sample depth, and the width for each probe input. Communication with the ILA core is conducted using an auto-instantiated debug core hub that connects to the JTAG interface of the FPGA. After the design is loaded into the FPGA, use the Vivado logic analyzer software to set up a trigger event for the ILA measurement. After the trigger occurs, the sample buffer is filled and uploaded into the Vivado logic analyzer. This data can be displayed using the waveform window. Regular FPGA logic is used to implement the probe sample and trigger functionality. On-chip block RAM memory stores the data until it is uploaded by the software. No user input or output is required to trigger events, capture data, or to communicate with the ILA core.

Chapter 6

SERDES

DRAC and DIRAC firmware are conceptually divided into three blocks. The first one is responsible to collect data from Digitizer for DRAC and ADCs for DIRAC. This is peculiar for the two boards because it depends on detector's feature. The middle block is the one which handles memory and is similar for both firmware. Some differences are in memory management because of the data format. Data format is detector dependent but concept under memory addressing is the same. Last block handles communication between the Trigger and Data Acquisition (TDAQ) and the Mu2e detector subsystem Readout Controllers (ROCs). This block is identical for both boards because the interface is shared.

The three blocks were developed in parallel defining the interfaces between them, then joined together to form what is the DRAC and DIRAC firmware. To develop the block that handles interface with the TDAQ, the AVMPF300TS-20-NA Evaluation Board [19] from Microsemi was used. It supports the same Microsemi PolarFire FPGA (MPF300TS-1FG1152) that is on the DRAC and DIRAC. In figure 6.1 is shown the board and in figure 6.2 the features on the board.

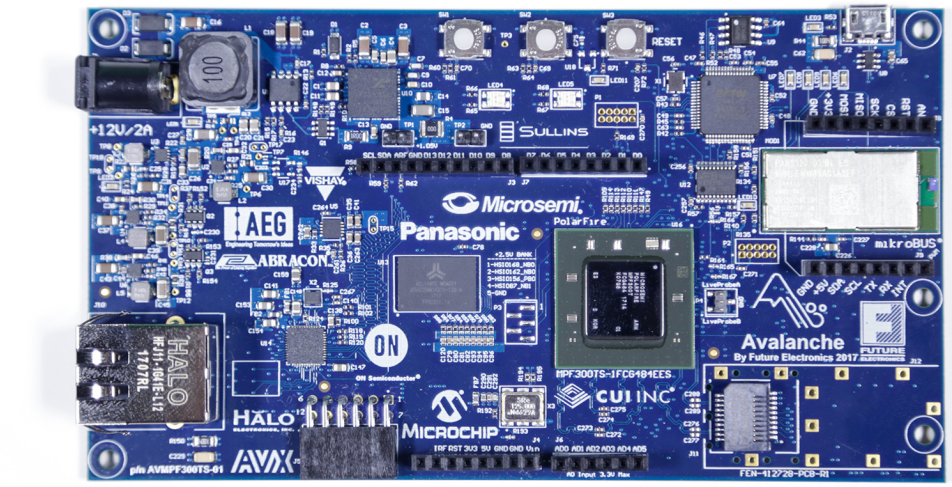


Figure 6.1: Microsemi AVMPF300TS-20-NA. This evaluation board is the one used to develop SERDES firmware.

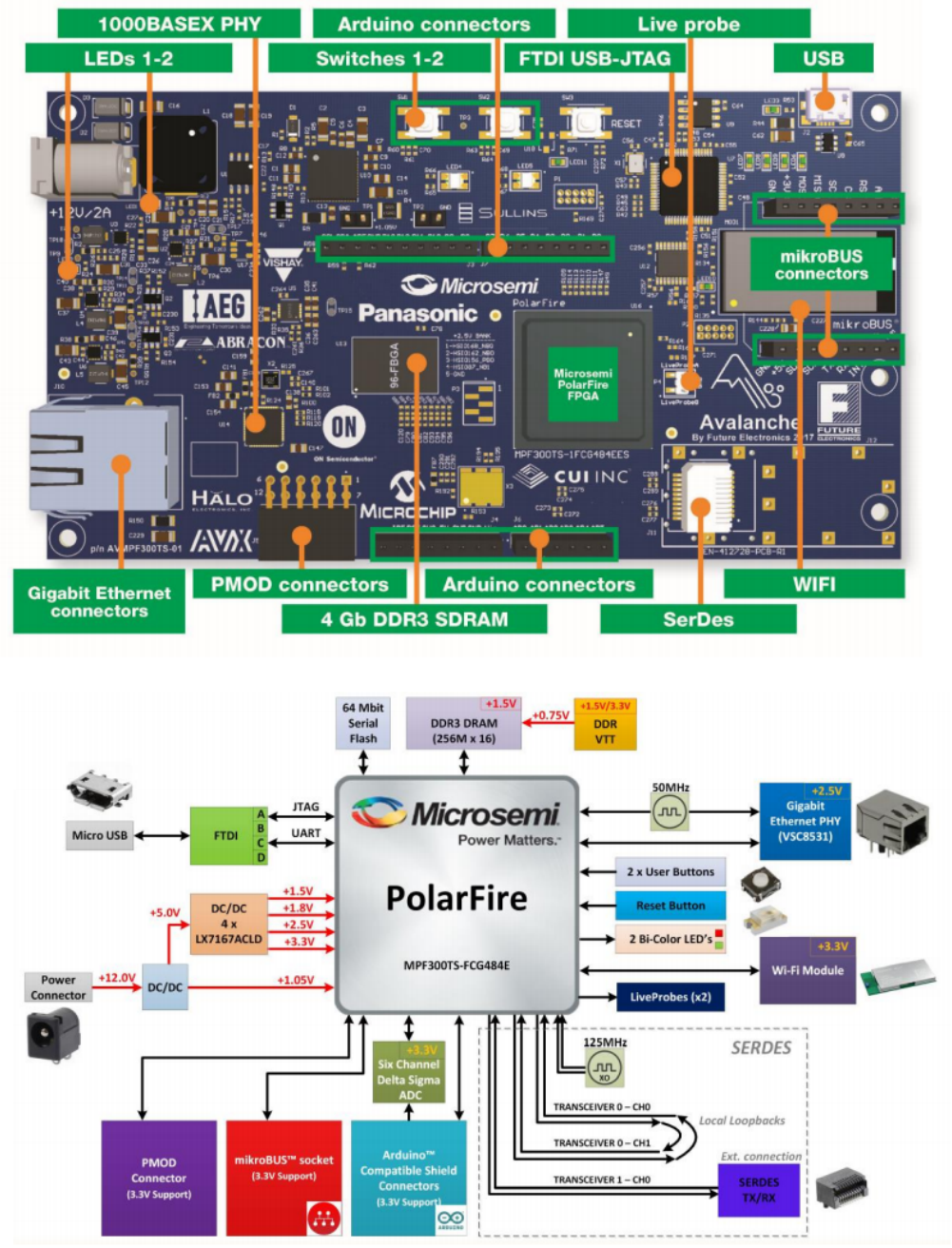


Figure 6.2: Microsemi AVMPF300TS-20-NA. The board has an MPF300TS-FCG484EES FPGA, a 1GbE interface w/PHY (VSC8531), a WiFi Module (PAN9320), a Serdes SFP Cage, a DDR3 SDRAM (256Mx16), 64Mbit SPI Flash.

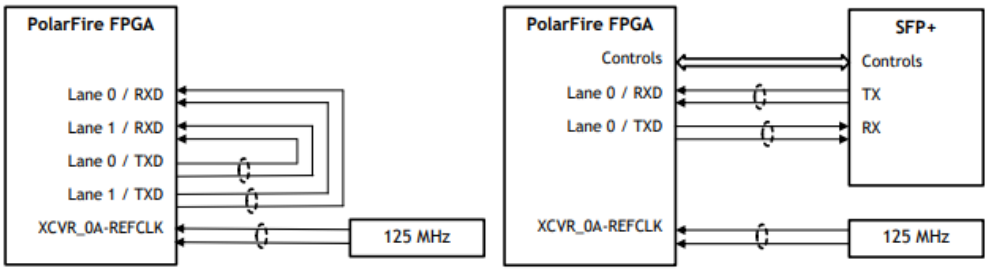


Figure 6.3: XCVR0 Interface and XCVR1 Interface.

The main purpose of the firmware developed on the evaluation board is to manage communication between the Trigger and Data Acquisition (TDAQ) and the Mu2e detector subsystem Readout Controllers (ROCs). This is an high speed communication, so it uses a Serializer/Deserializer (SerDes) block. Because of this feature, all firmware on the AVMPF300TS-20-NA Evaluation Board takes the name of SERDES. In this chapter are illustrated the main feature of SERDES firmware: optical links management, commands interpretation and experiment synchronization.

6.1 XCVR - Optical Links Management

First feature of SERDES is implementation of optical fibers infrastructure. ROCs interface to the Trigger and Data Acquisition (TDAQ) via optical links and at firmware level it translates as a transceiver module: the XCVR [17].

The PolarFire FPGA family includes multiple embedded low-power, performance-optimized transceivers. Each transceiver has both the physical medium attachment (PMA), protocol physical coding sub-layer (PCS) logic, and interfaces to the FPGA fabric. The transceiver has a multi-lane architecture with each lane natively supporting serial data transmission rates from 250 Mbps to 12.7 Gbps. The transceiver includes all required analog functions for high-speed data transmission between devices over printed circuit boards (PCB) and high-quality cables.

The PolarFire AVMPF300TS-20-NA device has sixteen transceiver lanes, which can be accessed through a PCB loopback and the SPF+ connector on the board. In this project, the SPF+ connector is used to mount the VTRx, the Versatile Transceiver developed by CERN. It is a bi-directional module composed of both optical transmitter and receiver.

A 125 MHz clock oscillator with an accuracy of ± 50 ppm is available on the board. This clock oscillator is connected to the FPGA fabric to provide transceiver reference clock.

Figure 6.3 shows the XCVR0 and the XCVR1 interface of the Avalanche Board. The XCVR0 interface has two lanes: Lanes 0 and 1 are directly routed together to form a loopback. The XCVR0 reference clock is routed directly from the 125 MHz differential clock oscillator to the PolarFire device. The XCVR0 TXD pairs are capacitive coupled to the PolarFire device. Serial AC-coupling capacitors are used to provide common-mode voltage independence.

The XCVR1 interface has one lane that is connected to the SFP+ connector. Lanes 0 is directly routed to the SFP+ connector. The XCVR0 reference clock can be used with the XCVR1 interface.

The PolarFire transceiver is divided into four distinct transmit (Tx) and receive (Rx) blocks: PMA, PCS interface block (including a dedicated PCIe PCS), transmit PLL (Tx PLL) and a reference clock inputs.

The high-speed PMA blocks connect to the FPGA fabric through the PCS block. The PMA generates the required clocks and converts the transmit data from parallel to serial, and receive data from serial to parallel. Each PMA block includes a connection to a PCS block and associated interface to the FPGA fabric making up a transceiver lane. The PCS interface block provides several industry-standard interfaces for use in protocol-specific designs. A group of four transceiver lanes is called a quad. Each quad has a local transmit PLL used exclusively within the four transceiver lanes. Additional transmit PLLs are shared between quads.

The transceiver lanes include PMA receiver and transmitter sub-modules. These PMA sub-modules include the input and output buffers, signal conditioning circuits, CDRs, and transceiver.

The receiver deserializes high-speed serial data received through the input buffer by creating a parallel data stream for the FPGA fabric and recovering the clock information from the received data. The receiver portion of the PMA includes the receiver buffer, the clock and data recovery (CDR) unit, and the deserializer. The deserializer within the receive PMA passes deserialized data to the PCS block across a data bus up to 40-bits wide of the PMA-PCS interface, which provides the data path to the gearing logic before the data is passed to the FPGA fabric.

The receive Clock and Data Recovery (CDR) PLL can lock onto the input reference clock or the incoming data stream to be able to re-time the incoming data. The deserializer is closely coupled with the CDR, and translates the data from a serial to a parallel stream.

The deserializer has a bit-slip feature for word alignment. In this mode, the CDR slips to the next bit from the deserializer. This feature helps with building word-alignment logic in the fabric and adjusts the alignment of the deserialized word by 1-bit in either direction when the bit-slip feature is active, reducing the uncertainty by ensuring deterministic latency. This feature is supported by the transceiver configurator.

6.1.1 8b/10b Encoding

The communication uses a 8b/10b encoding. In telecommunications, 8b/10b is a line code that maps 8 bit words to 10-bit symbols to achieve DC-balance and bounded disparity, and yet provide enough state changes to allow reasonable clock recovery. This means that the difference between the counts of ones and zeros in a string of at least 20 bits is no more than two, and that there are not more than five ones or zeros in a row. This helps to reduce the demand for the lower bandwidth limit of the channel necessary to transfer the signal [34].

As the scheme name suggests, eight bits of data are transmitted as a 10 bit entity called a symbol, or character. The low five bits of data are encoded into a 6 bit group (the 5b/6b portion) and the top three bits are encoded into a 4 bit group (the 3b/4b portion). These code groups are concatenated together to form the 10 bit symbol that is transmitted on the wire. The data symbols are often referred to as D.x.y where x ranges over 0–31 and y over 0–7. Standards using the 8b/10b encoding also define up to 12 special symbols (or control characters) that

can be sent in place of a data symbol. They are used to indicate start-of-frame, end-of-frame, link idle, skip and similar link-level conditions. At least one of them (i.e. a "comma" symbol) needs to be used to define the alignment of the 10 bit symbols. They are referred to as K.x.y and have different encodings from any of the D.x.y symbols.

The 8b10b trans-coder is protocol independent, in other words, it does not include a protocol-specific word aligner or word alignment state machine. Comma-detection is supported in the transceiver module in this firmware. The serial data must be aligned to comma-alignment boundaries before being used as parallel data. Without proper alignment, the incoming 8b10b data does not decode correctly. The comma character (K28.5) is used for alignment purposes as its 10-bit code is guaranteed not to occur elsewhere in the encoded bit stream.

In this case idle character is 3CBC.

The 8b10b PCS block performs the comma code-word detection and alignment operation. The comma character is used by the receive logic to align the incoming data stream into 10-bit words. The alignment comma descriptions (K28.1, K28.5, and K28.7) are defined in section 36.2.4.9 of the IEEE 802.3.2002. A comma is identified when there is a match across any 8 consecutive bits to 00111110 or 11000001 patterns. The only legal 10b characters, which contain series of bits are K28.1, K28.5, and K28.7. In 802.3 specification definition, there is no occurrence of two legal 10b characters sent in a sequence containing the comma pattern, which drastically reduces the chance that a symbol aligner can falsely lock. Alignment status per lane is indicated by the LANE_RX_VAL output pin going to high only after the PMA CDR locks onto an incoming data stream. Word Aligner can lock onto an incorrect alignment causing disparity errors and/or code violations from the 8b10b decoder. In this case, the word aligner needs to be reset to find a new alignment. This can be done by using the PCS_ARST_N reset. The fabric logic needs to monitor the LANE_RX_CODE_VIOLATION and LANE_RX_DISPERROR to determine when to issue a PCS_ARST_N and find a new alignment.

6.1.2 Implementation

High-speed serial protocols are supported using multiple transceiver building blocks in the transceiver configurators in the Libero design software. The Libero configurator allows the user to set the reference clock and data rates for particular protocols. This information is then used to properly generate the configuration settings for the PMA, and the associated interface logic.

Transceiver Reference Clock Configurator
The Transceiver Reference Clock Configurator is used to build the correct reference clock input to the transceiver and to the Tx PLL. In the project the Reference Clock is set in a differential mode, with fabric clock output enabled but not connected: when enabled, a port is exposed for fabric routing.

Transmit PLL Configurator
The Transceiver Transmit PLL Configurator is used to build the correct transmit PLL to the transceiver. In the project, the Reference Clock Source comes from the Transceiver Reference Clock Configurator at 125 MHz with a Desired Output Bit Clock of 4000.000 Mbps (2000.000 MHz).

Transceiver Interface Configurator

The Transceiver Interface Configurator is used to build the transceiver based on protocol requirements. Lanes can be up to 4; in the project is 1. Enhanced Receiver Management is enabled without Receiver Calibration. About PMA Settings, the data rate is set to 4000 Mbps with a clock division factor on TX of 1: the computed PLL base data rate is 4000 Mbps and the computed bit lock frequency is 2000 Mhz. CDR locks on data with a dedicated reference clock source of 125 MHz. About PCS Settings, PCS-fabric interface width is 20, so the computed FPGA interface frequency is 200 MHz. On the PMA mode, the CDR bit-slip port is enable.

Core Reset

CoreReset will allow synchronization of the resets to the user-specified clock domain into which each reset is feeding, so that while assertion is asynchronous, negation is synchronous to the clock. It generates a reset which is asserted asynchronously by one of multiple potential sources and which negates scynchronously to a specified clock. This ensures that recovery time of downstream logic is met and that all flip flops come out of reset in the same clock period. Output of this block is FABRIC_RESET_N, which may be used to reset user logic in the fabric. It is an active low reset, which asserts asynchronously, but negates synchronously to CLK.

Core PCS

Core PCS provides the 8b10b function for the physical coding sublayer for Gigabit Ethernet as defined in the IEEE 802.3z specification. The 8b10b is a marriage of two sub-blocks, the 5b6b and the 3b4b encoder/decoders. The purpose of the encoder/decoders is to convert 8 bit data into a 10 bit code that contains an equal number of 0's and 1's. In addition, the code is built so that no more than five consecutive 0's or 1's is ever transmitted. Core PCS is designed to work directly with a variety of standard transceiver devices. A set of generic signals provides a data and command interface to system logic. Core PCS provides a user interface and a transceiver interface. The user interface consists of transmit data, receive data, and several control and status signals used to qualify the data. The transceiver is responsible for serializing transmit data and deserializing receive data. In addition, the transceiver is designed to resynchronize the serial stream whenever it detects illegal coding errors.

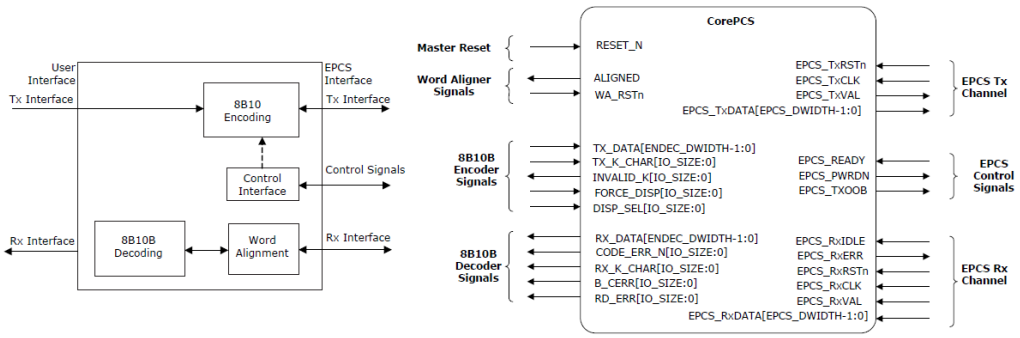


Figure 6.4: Core PCS block diagram and I/O Signal Diagram.

CorePCS consists of three major blocks, as described below. All signals on the Tx Interface are clocked using EPCS_TxCLK and all signals on the Rx interface are clocked using EPCS_RxCLK. EPCS stands for external physical coding sublayer.

When generic/parameter EPCS_DWIDTH is set to 20 bit or greater, the 8b10b transmitter is a pipe-lined structure that converts parallel command or data information into parallel encoded values. Command and data information are qualified by the TX_K_CHAR[IO_SIZE-1:0] bus. TX_K_CHAR[IO_SIZE-1] corresponds to the upper data byte on TX_DATA[ENDEC_DWIDTH-1:0] and TX_K_CHAR[0] is for the lower byte. The data on the TX_DATA bus is continuously registered into the transmitter. The transmitter will encode and send the upper byte first followed by the lower bytes. Because of the pipe-lined nature of the transmitter, the first encoded data will be driven on the TX_DATA bus several cycles after it is registered into the transmitter. All data input information is valid; however, command possibilities are limited. If the transmitter detects a bad command, then it will assert the INVALID_K signal. The core of the transmitter consists of a data encoder, a command encoder, and a disparity calculator. Each encoder calculates a 4B and 6B code for the input data. The correct code, command or data, is then selected based on the original input value of TX_K_CHAR. The disparity calculator determines whether the encoded value needs to be inverted to maintain the correct running disparity. The input FORCE_DISP[IO_SIZE-1:0] can be used to forced the data being registered into the transmitter to a selected running disparity. The input DISP_SEL[IO_SIZE-1:0] is used to select the running disparity. FORCE_DISP [IO_SIZE-1] and DISP_SEL [IO_SIZE-1] corresponds to the upper data byte on TX_DATA [ENDEC_DWIDTH-1:0] while FORCE_DISP [0] and DISP_SEL [0] is for the lower byte. Finally, the code is registered and sent to the transceiver on the TX_DATA bus.

When generic/parameter EPCS_DWIDTH is set to 20-bit or greater, the 8b10b receiver is a pipe-lined structure that converts parallel 10 bit encoded values and converts them to parallel command or data information. Command information is indicated by the RX_K_CHAR [IO_SIZE-1:0] bus signals asserted high. The data on the upper byte of the RX_DATA bus is the first decoded value in the sequence.Receive data is first registered into parallel registers. The codes are decoded in parallel moving from stage to stage. Several signals qualify the validity of the information on RX_DATA. RX_DATA contains good information whenever both the CODE_ERR_N is inactive (high) and ALIGNED is active (high). If ALIGNED is low or CODE_ERR_N is low, then some problem exists in the transmission. Whenever the receiver loses sync (ALIGNED is low) the transceiver will resynchronize the data on subsequent COMMA commands (K28.5). When sync is re-established, the ALIGNED will again be driven high after the pipeline has been flushed of potentially bad data. The error check block monitors the incoming codes and checks for illegal codes and bad running disparity. Whenever an error in the 8b10b code is detected, the CODE_ERR_N is asserted. If several codes in a row are received with errors, then the 8b10b will assume that synchronization with the transceiver has been lost and will deactivate ALIGNED and assert the COMMA_DET_EN signal. The number of consecutive errors required to force a resynchronization is fixed to 4. The transceiver will then resynchronize the data using COMMA codes. The 8b10b responds by asserting ALIGNED indicating that the transceiver has reacquired sync. The Core PCS supports word alignment for COMMA characters K28.5. Word alignment must be performed in the receiver before the data can make it through the core. Word alignment is achieved by transmitting a burst of consecutive COMMA characters before transmitting the data, however the number of consecutive COMMA characters is configurable when

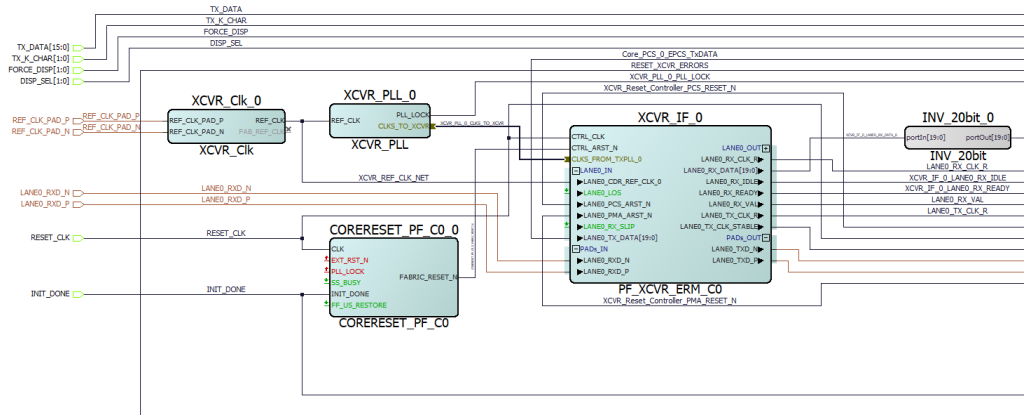


Figure 6.5: Reference Clock, Transmit PLL and Transceiver Interface blocks in SERDES Firmware.

PROG_COMMA_EN is enabled.

Word alignment shift is controlled by the parameter/generic SHIFT_EN. It is required that word alignment shift is enabled when Core PCS is configured for a 20 bit or greater EPCS received channel and it is handling data from protocols with a continuous stream of COMMA characters for word alignment followed by a stream of data which required the 16 bit data received to be in the upper and lower bytes after the last COMMA characters is received.

6.1.3 Clock Distribution

The XCVR is the block responsible for clock recovery.

In serial communication of digital data, clock recovery is the process of extracting timing information from a serial data stream to allow the receiving circuit to decode the transmitted symbols [35]. Clock recovery from the data stream is expedited by modifying the transmitted data. Wherever a serial communication channel does not transmit the clock signal along with the data stream, the clock must be regenerated at the receiver, using the timing information from the data stream. As seen, the receiver generates a clock from an approximate frequency reference, and then phase-aligns the clock to the transitions in the data stream with a phase-locked loop (PLL). This is one method of performing a process commonly known as clock and data recovery (CDR).

In order for this scheme to work, a data stream must transition frequently enough to correct for any drift in the PLL's oscillator. The limit for how long a clock-recovery unit can operate without a transition is known as its maximum consecutive identical digits (CID) specification. To ensure frequent transitions, some sort of self-clocking signal is used, often a run length limited encoding; in this case the 8b/10b encoding is used.

Transceiver uses a reference clock to lock and be able to re-time the incoming data. This is a 125 MHz reference clock provided by a clock oscillator available on the board and connected to the FPGA fabric. It takes name of REF_CLK_PAD and it is differential. Reference clock is handled by the Configurator to be useful by the PLL. PLL multiplies the reference clock to provide a 2 GHz clock to the transceiver logic (CLKS_TO_XCVR). The transceiver recovers the clock from data stream and outputs two clocks: LANE0_RX_CLK and LANE0_TX_CLK.

REF_CLK_PAD	125 MHz
CLKS_TO_XCVR	2000 MHz
LANE0_RX_CLK	200 MHz
LANE0_TX_CLK	200 MHz
EPCS_TXCLK	200 MHz
Clk_40Mhz	40 MHz
ALGO_CLK	40 MHz
RESET_CLK	160 MHz
SLOW_CLK	40 MHz

Table 6.1: Clocks used in the design.

LANE0_TX_CLK takes name of EPCS_TXCLK and drives the response logic at 200 MHz, LANE0_RX_CLK takes name of EPCS_RXCLK and drives the receiving logic at 200 MHz. These clocks are manipulated to obtain Clk_40MHz at 40 MHz with a duty cycle of 60% and ALGO_CLK at 40 MHz. In table 6.1 are shown all the clock derived by the recovered clock.

Bitslip Function

Clock recovery allows to retrieve the clock from a stream data, but without a Bitsplip function it locks to the clock frequency but with a random relative phase. A crucial point in the system is the synchronization of all ROCs in the detector, so it is vital that the clock is both in frequency than in phase identical. Figure 6.6 shows how bits are captured to perform clock recovery [1].

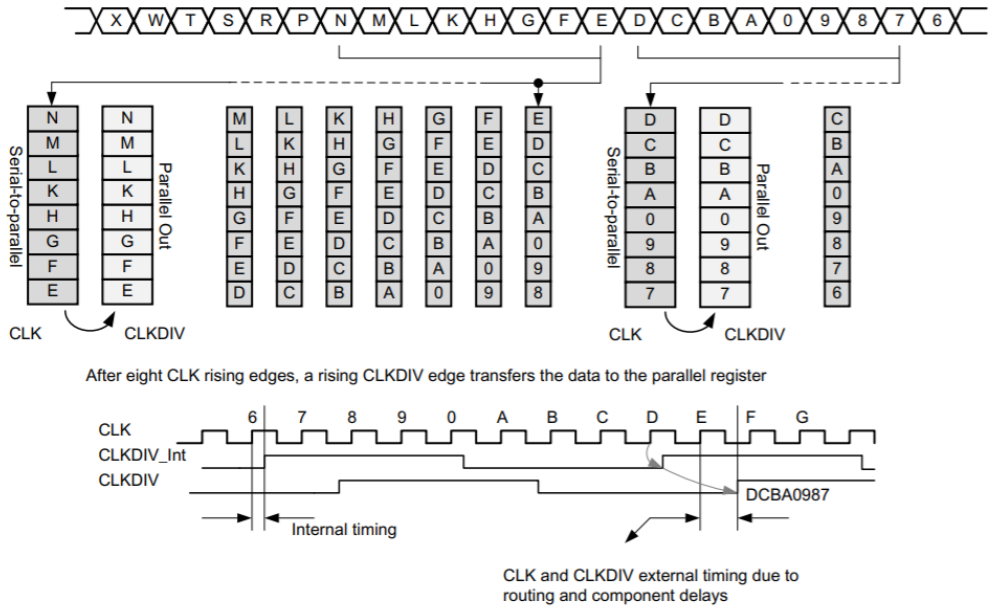


Figure 6.6: Scheme of clock recovery mechanism.

Data capture starts with bit value 7. Bits are shifted into the serial-to-parallel

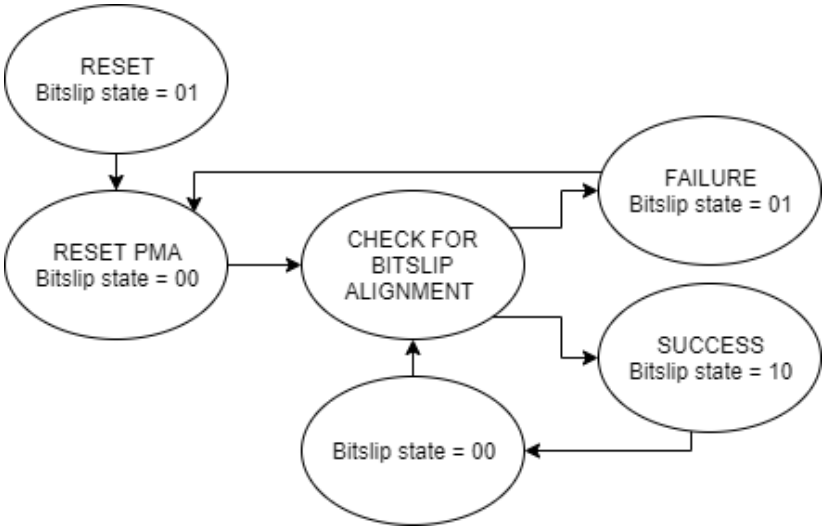


Figure 6.7: State machine of the Bitslip function.

register at the CLK clock rate. The vertical stacked blocks represent that register. These blocks show that the bit of value 7 is shifted in first and then ends at the bottom. The last bit shifted in is the bit with value D. The internal register captures the contents from the serial-to-parallel register when it contains eight captured bits. At the next CLKDIV rising edge, the content moves to the parallel output register. That register then contains the value DCBA0987. As soon as data is transferred from the serial-to-parallel register into the internal storage register, consecutive new data is shifted in the serial-to-parallel register.

The Bitslip function consists in the search of a special pattern inside the data stream with a fixed position. Every time clock is recovered, the Bitslip function checks if the pattern is aligned in the desired position, if is not, it resets the PMA to start another clock lock process. As shown in the state machine of the Bitslip function in figure 6.7, the process is reset PMA, check for alignment, in failure reset and go on until succeed.

6.2 Communication Protocol

The Mu2e Trigger and Data Acquisition (TDAQ) system is responsible for data processing, detector synchronization, control, monitoring, and operator interfaces. All these functions are implemented in ROCs firmware and access via Data Transfer Controller (DTC). DTC interfaces to ROCs via a defined packet protocol to accommodate TDAQ request, so ROCs firmware has the main function of command interpreter.

6.2.1 Packet Protocol

This section defines the data packet protocol used to exchange data between the ROCs and the TDAQ firmware and software [32]. The DTC modules are the immediate interfacing hardware to the ROCs. All communication between the TDAQ software framework and the ROCs must go through the DTCs. This implies that the DTC must be aware of the packet format templates, but not necessarily the packet detail.

The core functionality includes data stream packets (Data Header and Data Payload packets) and Detector Control System (DCS) packets (DCS Read and Write packets). The DCS is responsible for providing information about the status and health of the Mu2e detector. The DTC modules each have two Direct Memory Access (DMA) ports to provide high bandwidth data transfer between the PCI express and the DTC module's onboard memory. DMA channel 0 is reserved for Readout Request, Data Request, Data Header, and Data packets. DMA channel 1 is used for Detector Control System (DCS) Request and Reply packets.

Packets follow the 8b10b encoding. Eight bits of data are transmitted as a 10 bit character. The data symbols are referred to as D.x.y where x ranges over 0–31 and y over 0–7. Control characters, called comma characters, are used to indicate start-of-frame, end-of-frame, link idle, skip and similar link-level conditions. One of them needs to be used to define the alignment of the 10 bit symbols. They are referred to as K.x.y and have different encodings from any of the D.x.y symbols. The control symbols within 8b/10b are 10 bit symbols that are valid sequences of bits (no more than six 1s or 0s) but do not have a corresponding 8 bit data byte. They are used for low-level control functions.

Each packet has a Packet Type value. The Packet Type is a 4-bit field which maps as follows (table 6.2):

0	DCS Request
1	Heartbeat (broadcast)
2	Data Request
3	Reserved
4	DCS Reply
5	Data Header
6	Data Payload
7	DCS Additional Block Write Payload
8	DCS Reply Additional Block Read Payload
9-15	Reserved

Table 6.2: Packet Type values.

Detector Control System Packets

DCS Request packets are queued for transmission to the ROCs on DMA channel 1. The DCS Reply packets generated by the ROCs in response to the DCS Request packets are then transferred from the DTC to the TDAQ server via DMA channel 1.

DCS Request Packet

<table>
<tr><td colspan="3">K28.0</td><td colspan="2">D0.y</td></tr>
<tr><td>Valid</td><td>Reserved</td><td>ROC Link ID [10:8]</td><td>Packet Type (0x0)</td><td>Hop Count [3:0]</td></tr>
<tr><td colspan="3">Block Op Packet Count [15:6]</td><td>Reserved [5:4]</td><td>Op Code [3:0]</td></tr>
<tr><td colspan="5">Op1 Address [15:0]</td></tr>
<tr><td colspan="5">Op1 Write Data [15:0] or Block Op Word Count [15:0]</td></tr>
<tr><td colspan="5">Op2 Address [15:0] or Block Write Data0 [15:0]</td></tr>
<tr><td colspan="5">Op2 Write Data [15:0] or Block Write Data1 [15:0]</td></tr>
<tr><td colspan="5">Reserved or Block Write Data2 [15:0]</td></tr>
<tr><td colspan="3">CRC high</td><td colspan="2">CRC low</td></tr>
</table>

Table 6.3: DCS Request Packet

The Op Code in the DCS Request Packet is defined as follows (table 6.4):

Bit Position	Definition
1:0	0 := Read(s), 1:= Write(s), 2:= Block Read, 3:= Block Write
2	Double Operation
3	Request Acknowledgement

Table 6.4: Op Code in the DCS Request Packet

DCS Reply Packet

K28.0			D0.y	
DMA Byte Count High			DMA Byte Count Low	
Valid	Reserved	ROC Link ID [10:8]	Packet Type (0x4)	Hop Count [3:0]
Block Op Packet Count [15:6]			Status [5:4]	Op Code [3:0]
Op1 Address [15:0]				
Op1 Read Data [15:0] or Block Op Word Count [15:0]				
Op2 Address [15:0] or Block Read Data0 [15:0]				
Op2 Read Data [15:0] or Block Read Data1 [15:0]				
Reserved or Block Read Data2 [15:0]				
CRC high			CRC low	

Table 6.5: DCS Reply Packet

Data Stream Packets

Data stream packets are queued for transmission to the ROCs on DMA channel 0. The Data Reply packets generated by the ROCs in response to the Data Request packets are then transferred from the DTC to the TDAQ server via DMA channel 0.

Heartbeat Packet

K28.0			D0.y	
Transfer Byte Count High			Transfer Byte Count Low	
Valid	Reserved	ROC Link ID [10:8]	Packet Type (0x1)	Hop Count [3:0]
Event Window Tag byte 1			Event Window Tag byte 0	
Event Window Tag byte 3			Event Window Tag byte 2	
Event Window Tag byte 5			Event Window Tag byte 4	
Event Mode Byte 1			Event Mode Byte 0	
Event Mode Byte 3			Event Mode Byte 2	
Delivery Ring RF Marker TDC [15:8]			Event Mode Byte 4	
CRC high			CRC low	

Table 6.6: Heartbeat Packet (broadcast)

The Heartbeat Packet format is similar to the DCS Request Packet, but it is a broadcast packet. As such, all ROC modules accept the packet as well as decrement the Hop Count and transmit the packet to the next ROC unless the

Hop Count is zero. The DTC should always populate the Hop Count field with the max Hop Count for the link. This is to keep the last ROC in the link from transmitting the Heartbeat Packet back to the DTC module. Referring to the example above, in a link with six ROCs, the maximum number of hops is five. The DTC should populate the Hop Count field with the number five in this case. The packet contains ROC Readout Request information specific to the Event Window. During on-spill, the Delivery Ring RF Marker TDC 8 bit field is an unsigned value time prediction (in units of 800 MHz - 1.25ns periods) of the offset position of the Delivery Ring Marker with respect to the start of the Event Window, which is specified by the rising edge of 40 MHz - 25 ns System Clock and the Event Window Marker. During off-spill, the TDC value will be 0. The Heartbeat Packet source is the CFO and contains Event Window specific data that varies from one Event Window to the next. Data for each Event Window is stored in the CFO Heartbeat Information Table in CFO memory. This Table may be static during a run or may be configured over the PCIe on a per Super Cycle basis. The Heartbeat Packet contains the Event Window Tag, along with partition information, in the form of the Event Window Mode, needed to configure the ROCs for the next Event Window, which is specified to begin on the next Event Window Marker. This information is used to control event reconstruction/filtering, ROC data taking, and ROC settings that must be synchronized to a specific μBunch (synchronous resets, digitization enable/disable, zero-suppression enable/disable, calibration signal injection, live gate expansion, etc.). Heartbeat packets can also be used to synchronize commands to precise times within a μBunch, by specifying the ROC internal timestamp (time offset from the start of the μBunch). This would be used, for example, to request calibration signal injection at a particular data sample time. In this case, the partition number in the Readout Request packet indicates that it is a calibration event and another field in the packet dynamically selects the internal timestamp.

Data Request Packet

<table>
<tr><td colspan="3">K28.0</td><td colspan="2">D0.y</td></tr>
<tr><td colspan="3">Transfer Byte Count High</td><td colspan="2">Transfer Byte Count Low</td></tr>
<tr><td>Valid</td><td>Reserved</td><td>ROC Link ID [10:8]</td><td>Packet Type (0x2)</td><td>Hop Count [3:0]</td></tr>
<tr><td colspan="3">Event Window Tag byte 1</td><td colspan="2">Event Window Tag byte 0</td></tr>
<tr><td colspan="3">Event Window Tag byte 3</td><td colspan="2">Event Window Tag byte 2</td></tr>
<tr><td colspan="3">Event Window Tag byte 5</td><td colspan="2">Event Window Tag byte 4</td></tr>
<tr><td colspan="3">Reserved</td><td colspan="2">Reserved</td></tr>
<tr><td colspan="2">Reserved [15:8]</td><td>Debug Type [7:4]</td><td>Reserved [3:1]</td><td>Debug[0]</td></tr>
<tr><td colspan="2">Reserved [15:11]</td><td colspan="3">Debug Packet Count [10:0]</td></tr>
<tr><td colspan="3">CRC high</td><td colspan="2">CRC low</td></tr>
</table>

Table 6.7: Data Request Packet

The Data Request Packet format is similar to the Heartbeat Packet format,

but it is not a broadcast packet. The Debug field is a single bit field, with '1' indicating to ROCs that this is a data request for debugging purposes. If the Debug bit is high, then the Debug Packet Count field indicates the number of debug data packets requested from the specified ROC. The Debug Type field indicates the debug packet type desired.

Data Header Packet

<table>
<tr><td colspan="4">K28.0</td><td colspan="4">D0.y</td></tr>
<tr><td colspan="4">Data Block Byte Count High</td><td colspan="4">Data Block Byte Count Low</td></tr>
<tr><td>Valid</td><td>Reserved</td><td colspan="2">Subsystem ID [10:8]</td><td colspan="2">Packet Type (0x5)</td><td colspan="2">ROC Link ID [3:0]</td></tr>
<tr><td colspan="3">0b00000</td><td colspan="5">Packet Count [10:0]</td></tr>
<tr><td colspan="4">Event Window Tag byte 1</td><td colspan="4">Event Window Tag byte 0</td></tr>
<tr><td colspan="4">Event Window Tag byte 3</td><td colspan="4">Event Window Tag byte 2</td></tr>
<tr><td colspan="4">Event Window Tag byte 5</td><td colspan="4">Event Window Tag byte 4</td></tr>
<tr><td colspan="4">Data Packet Format Version [15:8]</td><td colspan="4">Status [7:0]</td></tr>
<tr><td colspan="4">Event Window Mode [15:8]</td><td colspan="4">DTC ID [7:0]</td></tr>
<tr><td colspan="4">CRC high</td><td colspan="4">CRC low</td></tr>
</table>

Table 6.8: Data Header Packet

The 8 bit Data Packet Format Version number is intended to be a persistent mapping to a particular packet type for the duration of the experiment, so that data can be decoded even if packet definitions change in the future. The Packet Count 11 bit field is a count, 0 to 2047, indicating the number of Data Payload packets to follow the Data Header packet. The Subsystem ID 3 bit field is defined for each detector subsystem (0 = Tracker, 1 = Calorimeter, 2 = CRV, 4 = STM, 5 = ExtMon).

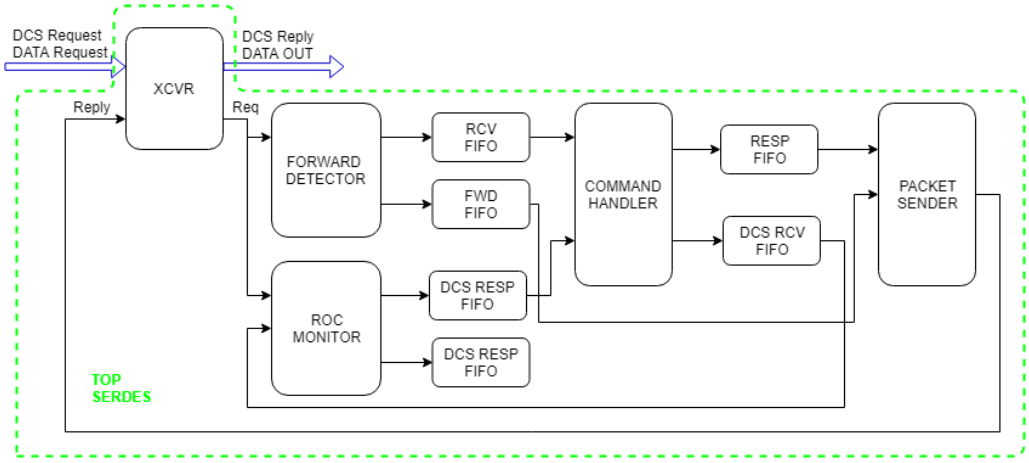


Figure 6.8: Block diagram of the firmware which handles communication between ROCs and DTC on the TDAQ.

6.2.2 Firmware Structure

The simple idea behind communication is to interpret packets sent by the DTC and set the controls for the data flow. As seen, XCVR decodes data and recover the 200 MHz clock distributed to all the logic.

The block diagram in figure 6.8 shows an overview of the firmware. The concept behind the system is "dividi et impera": each block takes care of a task in the command interpretation and accomplishing of DTC request and it is accessible by a standard interface on a bidirectional bus. Main blocks are Forward Detector, Command Handler, Roc Monitor and Packet Sender, while other blocks are used as storage of data and management of controls (e. g. FIFOs). Forward Detector, Command Handler and Packet Sender interfaces to Roc Monitor, which implements the slow protocol which allows the register access. Each main component of the ROC handles data requests and makes decisions using state machines. Below are described the state machines and the role of the main components.

Forward Detector

Forward detector (figure 6.9) is the first block that receive packets, so it is the one responsible for routing and forwarding. It uses two clocks: $EPCS\_RXCLK$, that is the RX Serdes Clock at 200 MHz, and $ALGO\_CLK$, the Roc Monitor Slow Control Clock at 40 MHz. It interfaces with Receive FIFO and Forward FIFO, with Roc Monitor, with 8b10b CorePCS and with a CRC calculation block. It is also responsible for detection of markers and for retransmission. First task of Forward Detector is to check if XCVR is locked; once it is locked, it sets all latches and clear all register and variables. After this general reset, Forward Detector is ready to receive request packet as RX_DATA. First of all, it distinguishes markers from actual data packets. Another more in-depth section will be devoted to the markers later in this chapter.

Focusing on the packets and following block diagram in figure 6.9, from a Reset state, the machine goes to an Idle state if no signals or variables are in an inappropriate status. The Idle state identifies the start of a valid packet

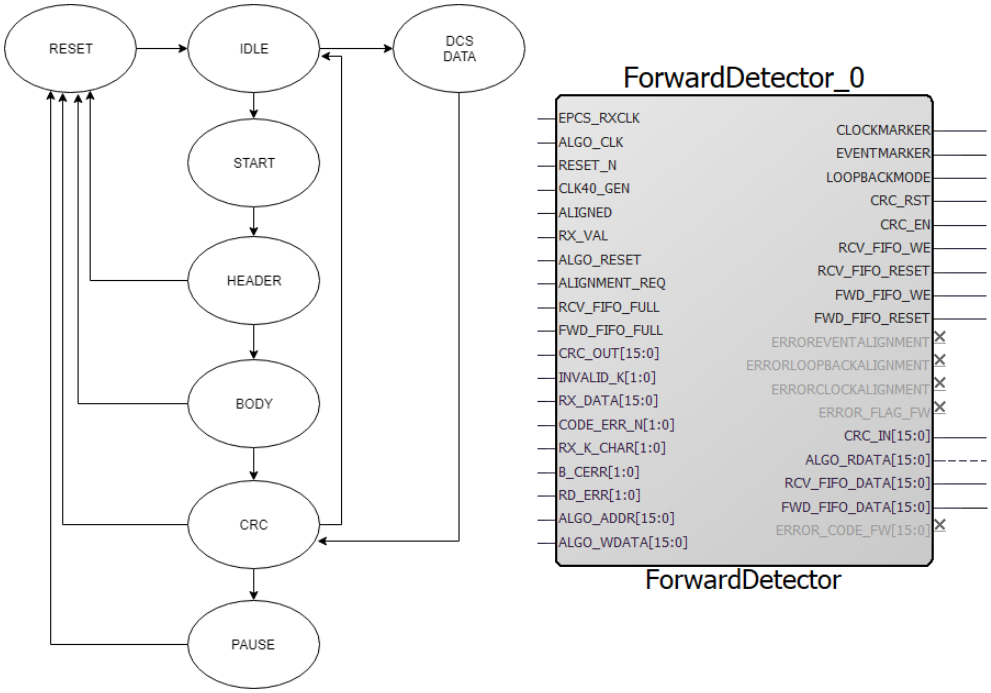


Figure 6.9: State Machine inside Forward Detector and Libero block

K28.0-D0.y	**Packet Type**
0x1C00	DCS Request Packet
0x1C01	Readout Request Packet
0x1C02	Data Request Packet
0x1C10	Event Marker
0x1C11	Clock Marker

Table 6.9: Packet Type values.

and retrieve information about the *packetType* as shown in table 6.9. Data are forwarded to the Receive FIFO and to the Forwad FIFO and machine goes to Start state.

Start state is necessary to ask for the CRC calculation. About CRC a more in-depth section will be devoted later in this chapter. Once the CRC is enabled, next state is the Header state.

In the Header state the body Packet Type and the Hopcount of the packet are retreived. Packet types with they correspondent value are described in 6.10:

For certain packets, like DCS Request, Heartbeat and Data Request, the Hopcount is decremented. For packet which should be forwarded, so the one with Hopcount zero and for DCS Reply, DCS Read Payload, Data Header and Data Payload, the Write Enable of the Forward FIFO is set to one and the state changes to Body. In state Header, destination of the packet is also checked and there is a double check between the body Packet Type and the header Packet Type. From this state on, packets are written inside the FIFOs, ready to be read by other blocks like Command Handler and Packet Sender.

Packet Type	Value
DCSRequest	0
Heartbeat	1
DataRequest	2
DCSReply	4
DataHeader	5
DataPayload	6
DCSWritePayload	7
DCSReadPayload	8

Table 6.10: Packet Type values.

The last state is the CRC, where the last two byte of the packet are matched with the calculated CRC. If they match the packet is not corrupted, otherwise Forward Detector asks for a retransmission. About CRC and Retrasmission, a section is dedicated later.

As all other blocks in the firmware, Forward Detector manages error detection and is supplied by an interface to Roc Monitor. It provides use of registers; about how the interface is built, Roc Monitor is the reference section. Important registers in Forward Detector are: writing 1 at address 1, start a reset pulse, writing at address 3 fill a dummy register which can be read at address 26; register 0 returns the status of Forward Detector, register 1 returns errors, register 2 returns the Packet Type; registers from 5 to 13 return information about marker offsets and registers from 20 to 25 return counters.

Markers

Communication protocol provides both packet and markers. markers are special character used to accomplish specific task. In this firmware are defined 6 different markers as specified in the following table (6.11): markers are necessary because they set the environment in which the Packet Protocol works. Each of them is in-depth explained in other section of this document, but the operation is common.

DTC sends markers to the ROCs and ROCs can send them back. To enhance robustness, DTC sends double markers this way: first word is the marker and

Marker	Reversed	Type
0x1C10	0x1CEF	Clock Detected
0x1C11	0x1CEE	Event Detected
0x1C12	0x1CED	Loopback Mode Detected
0x1C13	0x1CEC	DTC Error
0x1C14	0x1CEB	DTC Error
0x1C15	0x1CEA	Retransmission Request

Table 6.11: Markers code and value.

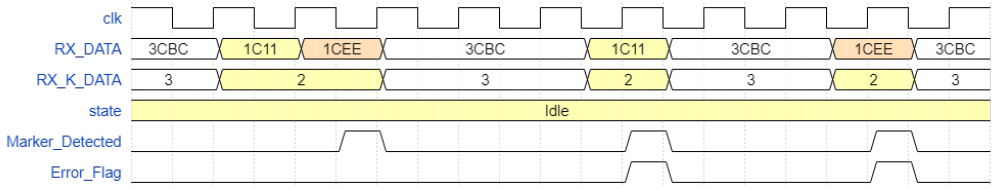


Figure 6.10: Detection of markers: ROC is able to detect both double and single markers, but when a single marker comes, ROC set an error flag.

second word is the 8 LSB of marker reversed. On the receiving side, ROC is able to detected both double markers and one marker. If only one of the double markers is detected, an error flag is raised (figure 6.10). K-characters enable markers identification. An important feature is the ability to recognize both normal than flipped data, also if markers.

Once a marker has been identified, the Forward Detector remains in the Idle state: markers are meant to be used like flags, for example a *RetransmissionRequest* marker enable a signal forwarded to Packet Sender.

Command Handler

Command Handler handles DCS Request and Data Packet Request. This means that is responsible for the decoding of the DCS packets and for the communication with the DDR interface.

It is clocked by the 40 MHz ALGO_CLK. It interfaces with the Receive FIFO, the DCS Response FIFO, the Response FIFO, the DCS Receive FIFO and with Roc Monitor.

Referring to the state machine in figure 6.11, after a fabric reset the state is Reset and goes immediately to Idle. The Idle state, if the FIFO is not empty, start reading commands. Commands are read from Receive FIFO where Forward Detector writes packets. If there are commands to be read, the state changes from Idle to Rcv, in which decisions are taken by the Packet Type: if the Packet Type is zero then it is a DCS Request, if the Packet Type is one then it is a Readout Request and it is necessary to acquire the timestamp to read, if the Packet Type is two it is a Data Request and it is necessary to acquire the timestamp to return data. From Rcv there are three possible state changes corresponding to the Packet Type: DcsReq, ReadReq and DataReq. From each state there is the possibility to change to a Drain state when error occurs.

A Data Request can be handled in a debug mode or not. SERDES does not draw on memory to retrieve actual data collected from the detector. It only provides the interface to the DDR3 Interface module for the DRAC and DRAC firmware. Data are simulated and Command Handler manages how to pack up them. When not in debug mode, next state from DataReq is DataReqHeader. Here the header packet is generated and written in Response FIFO. Next State is DataReqDataRead, where fake data are sent. In case of debug mode, as shown in figure 6.12, next state from DataReq is DbgData and then DbgDataHdrGen, where is generated a debug Data Header, and at the end DbgDataGen. This mode is necessary to test the SEU SRAM and the SEU DDR3.

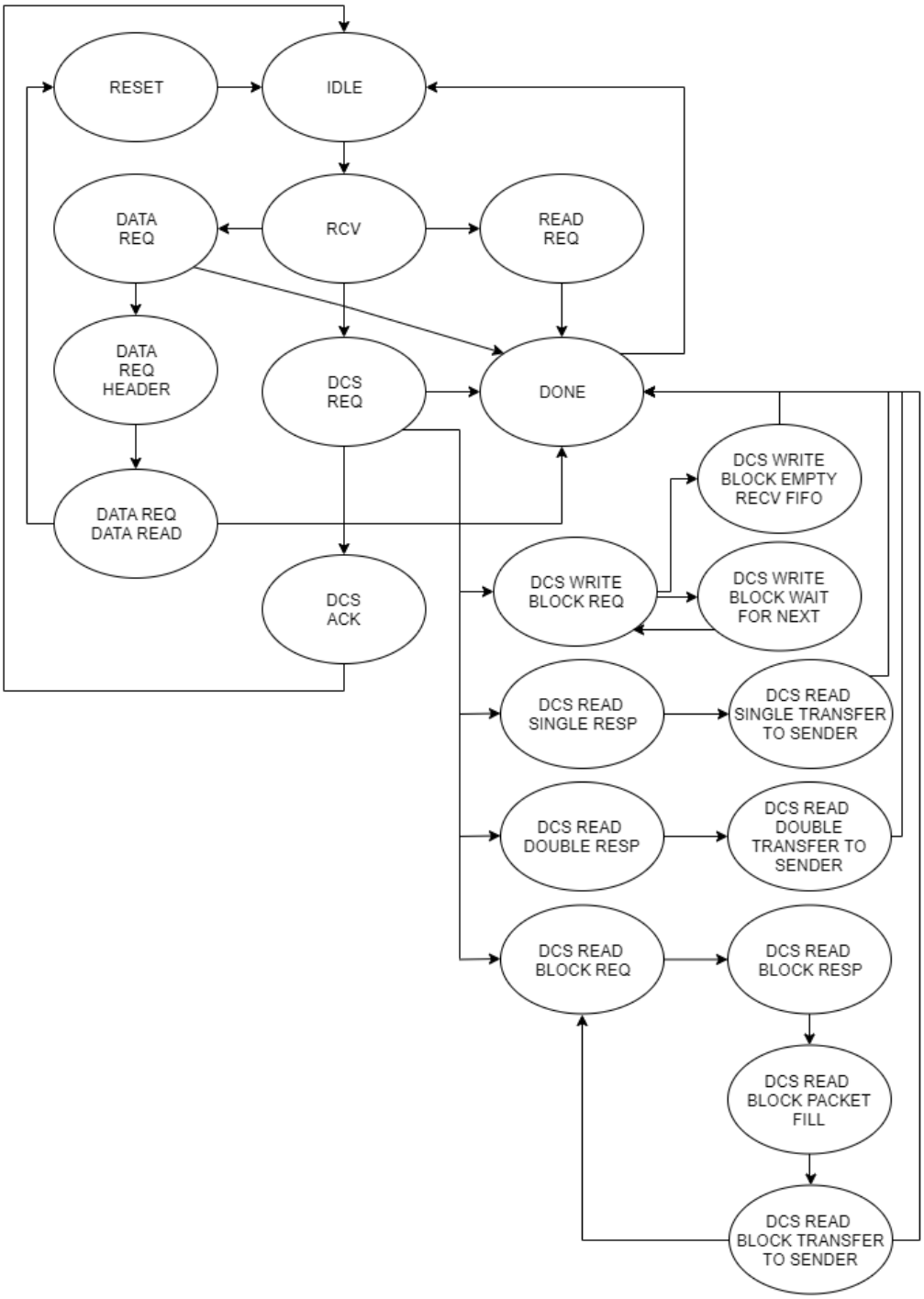


Figure 6.11: State Machine inside Command Handler

Figure 6.12: State Machine of the Debug Mode in Data Request inside Command Handler

A DCS Request is handled entering in the DcsReq state. First the Op Code (table 6.8) and the address are identified to know the type of DCS request once it is captured. Depending on the code the request can be a read, a write, a block read or a block write and the operation can be single or double. Interpreting request is done scanning bytes of the DCS Request packet. Some information can be retrieve from the first byte, other from the following. For example the first byte distinguish reading and writing, but only the third one inform if it is a single or a double operation. Scanning bytes allows to define variables which drives the behavior of the state machine. Without going into too much detail, in figure 6.11, each change of state can be follow. Both reading and writing rely on DCS Receive FIFO. DCS Receive FIFO has an interface to Roc Monitor which is responsible for access to register and store replies in DCS Response FIFO. From DCS Response FIFO, Command Handler retrieves replies and sends them to Response FIFO. The Ack state is necessary to handle the DCS Reply Ack Packet.

Command Handler submits the same interface to Roc Monitor as Forward Detector. Reading corresponding register is possible to acquire information about the timestamp on packets.

As all other blocks in the firmware, Command Handler manages error detection. In vector *errors* are coded: undefined packet type received error, invalid data readout timestamp error, been to drain state error, invalid packet format error and read timeout error.

Roc Monitor

Roc Monitor gives an APB Advanced Peripheral Bus interface for all blocks in the firmware and handles user space addressing. It is clocked by the 40 MHz ALGO_CLK to provide the Slow Control Network. Slow Control consists in all these signals which allow status information retrieving. At Packet level it manages the DCS Request, so it accomplish registers reading and writing. There are two different register access: one directly on the Roc Monitor registers and one in other block's registers. The second one is called Block Reading and Block Writing and is performed by defining a user space.

Forward Detector, Command Handler and Packet Sender connects to Roc Monitor via the following 16 bit signals:

```
ALGO_CLK      : in std_logic;
ALGO_RESET    : in std_logic;
ALGO_ADDR     : in std_logic_vector(gAPB_DWIDTH-1 downto 0);
ALGO_WDATA    : in std_logic_vector(gAPB_DWIDTH-1 downto 0);
ALGO_RDATA    : inout std_logic_vector(gAPB_DWIDTH-1 downto 0);
```

ALGO_ADDR defines both user space and register address this way: ALGO_ADDR[7:0] identifies the block, ALGO_ADDR[15:8] identifies resister address.This way is possible to defines 256 block each with 256 different registers. ALGO_WDATA and ALGO_RDATA are 16 bit data bus.

Each block submits the template interface to Roc Monitor defined as follow:

```
signal locAddr        : unsigned(ALGO_LOCADDR_DWIDTH-1 downto 0);
signal wAddr          : unsigned(ALGO_WADDR_DWIDTH-1 downto 0);
signal rAddr          : unsigned(ALGO_RADDR_DWIDTH-1 downto 0);
signal wData          : std_logic_vector(ALGO_WADDR_DWIDTH-1 downto 0);
signal algo_rdata_sig : std_logic_vector(gAPB_DWIDTH-1 downto 0);
signal we             : std_logic;
```

locAddr is extracted by ALGO_ADDR as the 8 LSB bits and is the block identifier. It is compared to ALGO_LOC_ADDR to address the block. Each block has a different ALGO_LOC_ADDR, so it is a unique identifier.

wAddr also is extracted by ALGO_ADDR as the 8 MSB bits. It represents the write address. *wAddr* is used to setup the read address *rAddr*. *rAddr* is defined writing it at *wAddr* zero. Data are *wData* that is ALGO_WDATA[14:0] while the MSB of ALGO_WDATA is the write enable. *algo_rdata_sig* is simply data read in registers and is ALGO_RDATA.

Packet Sender

Packet Sender is responsible for the transmission and retransmission of packets.

It is clocked by the 200 MHz EPCS_TXCLK and use the 40 MHz ALGO_CLK to interface with Roc Monitor. It interfaces with the Response FIFO and the Forward FIFO and with a CRC calculation block.

Referring to figure 6.13, from a Reset state, Packet Sender goes to an Idle state. If Forward FIFO is not empty, it can forward the packet, so it enters in the Fwd state, otherwise it checks if a retransmission request occurs and enters in the WaitForRetransmitData state. Retransmission feature is discussed below. If there are packets to send, so if Response FIFO in not empty, it changes state to WaitForRespData. There is also a Dbg state which handles debug packet generation. State WaitForRespData needs to decide if grab or not the header depending on the packet to forward. In one clock cycle, the new state is Resp, which handles sending packet from Response FIFO. The first word of the sent packet contains its index and a value distinguishing the Packet Type. Depending on the Packet Type, in state Resp is assembled the packet to be sent.

Packet Sender submits the same interface to Roc Monitor as Forward Detector.

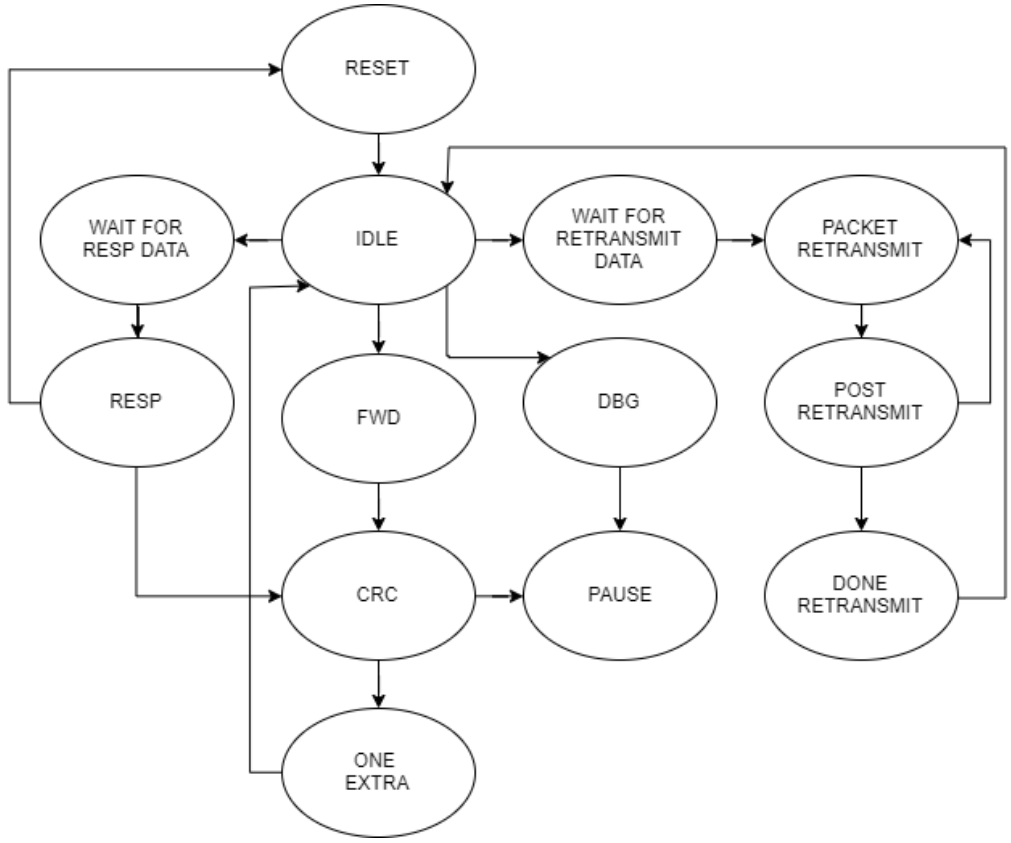


Figure 6.13: State Machine inside Packet Sender

Reading corresponding register is possible to acquire information about packets like the index, the type and the source.

6.2.3 Packet Managing

As already described, DTC can perform DCS Request and Data Packet Request. Data Request deals with data coming from event detection. When an event occurs, data about the event are collected and stored inside memory and retrieved when the TDAQ is ready to process them. DCS Request deals with monitoring and control of boards. DCS Requests translates as reads and writes of ROC's internal registers.

Following is described how firmware handles these Requests and how Retransmission is implemented.

Read and Write Request

There are several types of Read and Write Request: single, multiple and Block. They are all based on the single, that is following described. Multiple ones are simply request with incremental adressing while Block Request deals with the user space and the Block firmware adressing.

A Read request is a DCS request: the packet is described in figure 6.3 and is composed by 10 words of 16 bits. The Op Code distinguishes the Read request and the Operation Address defines the address to be read. The request is handled by Forward Detector, which forward the packet to the Receive FIFO. On the

Receive FIFO the first two words (K28.0 and D0.y and DMA count) and the last one (CRC) are discarded. Once the entire packet is written inside the FIFO, it results not empty and Command Handler changes his state from Idle to Received. On the Received state it interprets the Op Code and enters in the DCS Request state. On the DCS Reveice FIFO is written the address to be read. DCS Request are handled by Roc Monitor. It extracts the address as *roc_mon_addr* and return the read data as *apb_read_data*. Data and address are stored inside DCS Response FIFO and read by Command Handler. Command Handler packs up the reply with the read data and send that to the Response FIFO where data are read and sent out by Packet Sender.

A Write request is similar to a Read request because it is still a DCS request: the packet is described in figure 6.3 and is composed by 10 words of 16 bits. The Op Code distinguishes the Write request and the Operation Address and the Operation Data define the address and the data to be written. The request is handled by Forward Detector, which forward the packet to the Receive FIFO. On the Receive FIFO the first two words (K28.0 and D0.y and DMA count) and the last one (CRC) are discarded. Once the entire packet is written inside the FIFO, it results not empty and Command Handler changes his state from Idle to Received. On the Received state it interprets the Op Code and enters in the DCS Request state. On the DCS Reveice FIFO is written the address and the data. DCS Request are handled by Roc Monitor. It extracts the address as *roc_mon_addr* and data as *roc_mon_data*.

Block Read and Block Write Request

"Block" refers to VHDL Block inside the firmware and a Block Request allows to read and write registers only defined inside that parts of the firmware. In particular are defined as Block Forward Detector, Packet Sender and Command Handler. They divide the user space. User space is addressed by ALGO_LOC_ADDR, a constant vector of 8 bit that gives space to 256 blocks. A Block Request is performed relying on single Read and Write Request, so the path through the different Blocks and FIFO is the same, the difference is the sequence of them that allows to perform the Block Request. Performing a Block Write Request relies on a Double Write. Operations are:

1. Writing on address 12 of Roc Monitor that corresponds to write inside algo_adress_sig. algo_adress_sig is the ALGO_ADRR[15:0] routed to all Blocks in the firmware. ALGO_ADRR[7:0] addresses Blocks and ALGO_ADRR[15:8] addresses registres inside that Block. For example, writing 0308 at address 12 means addressing register 3 in Block 8, that is Forward Detector.

2. Writing on address 13 of Roc Monitor, that contains algo_wdata, data to be written inside the address specified in the previous step.

algo_wdata is routed to ALGO_WDATA. ALGO_WDATA is both data as ALGO_WDATA[14:0] and write enable as ALGO_WDATA[15]. So actual data written inside Block register are 15 bit long.

Performing a Block Read Request relies on a Double Write and a Read. Operations are:

1. Writing on address 12 of Roc Monitor that corresponds to write inside algo_adress_sig. ALGO_ADRR[7:0] addresses Blocks and ALGO_ADRR[15:8] addresses registres inside that Block. It is necessary because it allows to specify rAdress, that is the register to be read. Access to rAdress is allowed writing inside register 0 of the desired Block. So the right sequence is write inside 12 algo_adress_sig = Block to be read. For example, for Forward Detector it is algo_adress_sig = 0008. This is because the register to write inside Block ALGO_ADRR[15:8] is 0.

2. Writing on address 13 of Roc Monitor, that contains algo_wdata, register to be read inside the Block specified before. For example, to read address 26 decimal in Forward Detector, it is necessary write 1A hexadecimal on address 13 of Roc Monitor.

3. Read from address 22 of Roc Monitor, that contains ALGO_RDATA.

Heartbeat and Data Request

A Data Request occurs when the TDAQ is ready to process the event. Each Data Request is preceded by an Heartbeat. The Heartbeat Packet source is the CFO and contains Event Window specific data that varies from one Event Window to the next. It contains the Event Window Tag, along with partition information, in the form of the Event Window Mode, needed to configure the ROCs for the next Event Window, which is specified to begin on the next Event Window Marker. The packet is described in figure 6.6: it is composed by 10 words of 16 bits. The request is handled by Forward Detector, which forward the packet to the Receive FIFO. On the Receive FIFO the first two words (K28.0 and D0.y and DMA count) and the last one (CRC) are discarded. Once the entire packet is written inside the FIFO, it results not empty and Command Handler changes his state from Idle to Received. On the Received state it interprets the Packet Type and enters in the Read Request state. In this state the Event Window Tag is acquired and the state is Done.

The packet is described in figure 6.7: it is composed by 10 words of 16 bits. The request is handled by Forward Detector, which forward the packet to the Receive FIFO. On the Receive FIFO the first two words (K28.0 and D0.y and DMA count) and the last one (CRC) are discarded. Once the entire packet is written inside the FIFO, it results not empty and Command Handler changes his state from Idle to Received. On the Received state it interprets the Packet Type and enters in the Data Request state. Data Request state acquires the Event Window Tag. Next state is Data Request Header; here are set signal to ask data from the DDR3 Memory and is generated the packet header to be sent back to the DTC. The header requires how many data are in the requested Event Window, the Event Window Tag and the status. The header is passed to Response FIFO and then sent out by Packet Sender. Command Handler manages the forward of data from the DDR3 Memory to the DTC stacking packet of 10 words of 16 bit of data.

Data Retransmission

The goal of data retransmission is to enhance robustness of the DTC ROC protocol in both directions and provides transmission of data packets if a packet error is

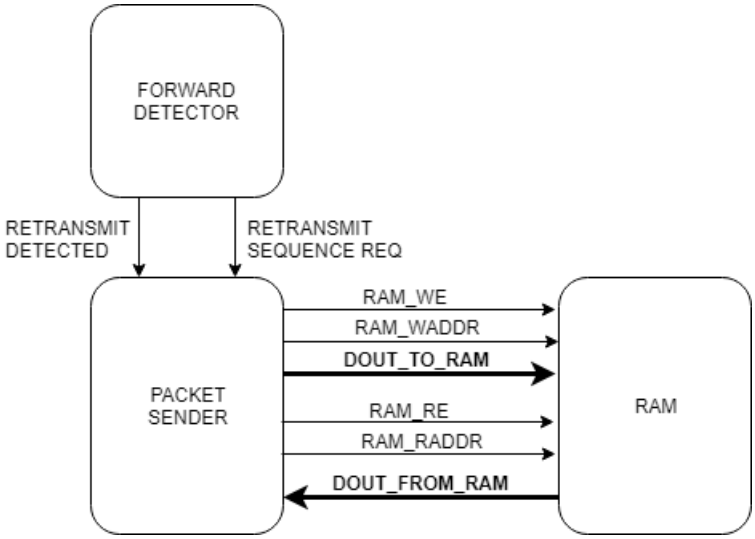


Figure 6.14: Block diagram of the fimware involved in the retransmission.

detected. ROCs handle requests from the TDAQ by identifying type, forwarding to the right path and correctly replying. These tasks are managed by the three main blocks: Forward Detector, Command Handler e Packet Sender. Forward Detector analyzes and decides the type of request and forwards it to Command Handler. Command Handler interprets the request and decides which reply, DCS or data packet, to send to the DTC and Packet Sender sends it. In this path finds place the Data Retransmission block. Before transmission, each reply is stored in a dual-port RAM by counts of 8. When ROC receives a retransmission request, stored data are read and sent to DTC. In figure 6.14 are shown the main blocks involved.

CRC - Cyclic Redundancy Check

Both DTC and ROC are able to detect packet errors. Each packet includes a 16 bit trailing Cyclic Redundancy Check word. CRC is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short check value attached, based on the remainder of a polynomial division of their contents. CRCs are so called because the check (data verification) value is a redundancy (it expands the message without adding information) and the algorithm is based on cyclic codes. A CRC is used because it is simple to implement in binary hardware, easy to analyze mathematically, and particularly good at detecting common errors caused by noise in transmission channels. The CRC polynomial used is: $crc[15:0] = 1 + x^3 + z^7 + x^{12} + x^{14} + x^{16}$ and it is calculated on the entire packet except of the K/D character header.

DTC calculates and includes CRC in each generated request. ROC receives the packet, calculates the CRC on that packet and compares it to the CRC received from the DTC. If the two CRCs do not match, the packet is corrupted. The same is performed in the opposite direction.

In ROC firmware there are two CRC calculation blocks, as shown in figure 6.15. The first one, RX_ForwardDetector, calculates CRC on the packet received from the DTC. Inputs of the block are the CRC enable (CRC_EN) and the packet from the DTC (DATA_IN[15:0]) and the output is the calculated CRC

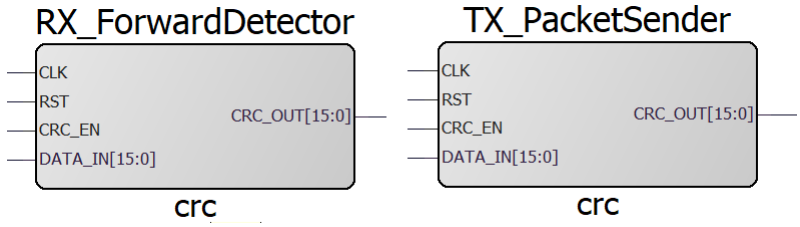


Figure 6.15: Blocks in the firmware that handle CRC calculation.

(CRC_OUT[5:0]). The enable and the CRC_OUT are sent and received by Forward Detector. Forward Detector compares the calculated CRC and the received one and decides whether or not for a retransmission. The other CRC calculation block, TX_PacketSender, calculates CRC on packets sent by Packet Sender, which also enables the calculation (CRC_EN). The calculated CRC is merged to the packet and sent to the DTC which will evaluate the need or not for a retransmission.

Error Detection

From the perspective of ROC firmware, when the sent CRC and the calculated by DTC CRC do not match, a retransmission request comes. A retransmission request is interpreted by the Forward Detector as a marker (figure 6.16). Marker are always at least 2 words of 16 bit packet. While the first word is the actual marker, the second word is used to enhance robustness: it is composed by the 8 least significant bit of the first word reversed. In the case of retransmission request the packet is 3 words long with the first word 1C15 hexadecimal, the second one 1CEA hexadecimal and the third one is the RETRANSMIT_SEQUENCE_REQ, which numerates the last uncorrupted packet received. When the DTC sends to the ROC this marker, Forward Detector raises a flag of RETRANSMIT_DETECTED with the corresponding RETRANSMIT_SEQUENCE_REQ number and forwards this information to Packet Sender.

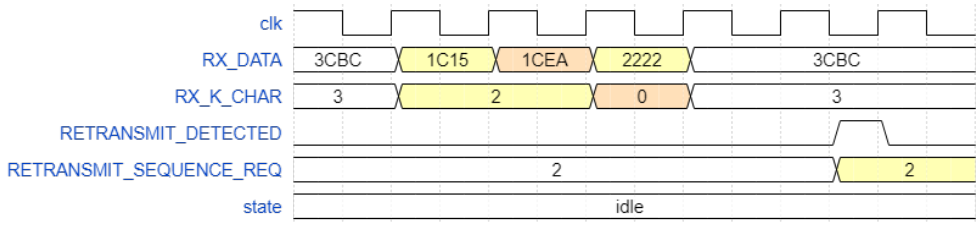


Figure 6.16: Marker in Forward Detector which identify a retransmission request. In this case, RETRANSMIT_SEQUENCE_REQ is 2.

RAM Storage

Each transmitted packet is stored in a dual-port RAM of 128 bits of depth and 16 bit of width. Packet Sender has the task of managing the RAM and the retransmission itself. In the first place it is responsible for the storage of data: it addresses the RAM and enables writing and reading.

RAM is 128 rows depth, so it can hold up to 8 packets. The addressing provides that the 3 most significant bits of the address are the index of the packet and the 4 least significant bits scans the words of the packet. As for the writing, this task is carried out by the signals resp_packet_count and retransmit_word_count_latch :

$$RAM\_WADDR <= resp\_packet\_count(2 downto 0) \& retransmit\_word\_count\_latch$$

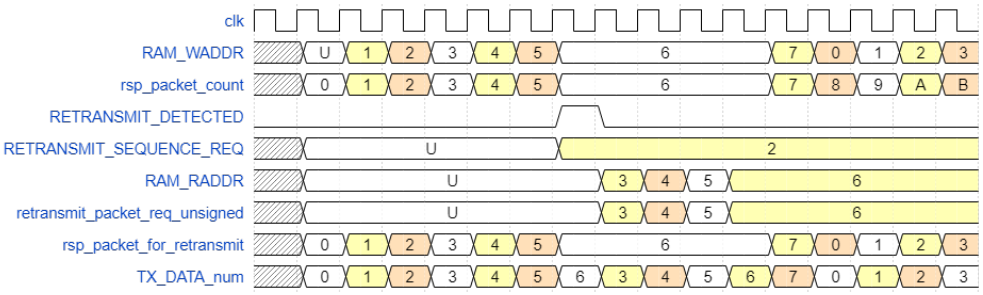


Figure 6.17: RAM addressing: as RAM_WADDR and RAM_RADDR are shown only the 3 MSB of the address that indexes the packet number

Signal resp_packet_count is the counter of how many packets are sent. It increases both for block sent packets that for multiple sent packets. For example, if DTC asks for a block read, it will increase of how many packet was asked as block number; if DTC sends two different register read request, it will increase of two. It always increases and is set to zero only when a reset occurs. Signal retransmit_word_count_latch is also a counter which scans the words in the packet. It is only used to address RAM.

As for the reading, the same strategy of addressing is used: the 3 most significant bits of the address are the index of the packet and the 4 least significant bits scans the words of the packet. In this case the responsible signals are retransmit_packet_req_unsigned*and*retransmit_read_count:

$$RAM\_RADDR <= retransmit\_packet\_req\_unsigned\&retransmit\_read\_count$$

Signal retransmit_packet_req_unsigned stores the index of the packet DTC is asking for. ROC retransmits packet from that index on, indeed Retransmit_packet_req_unsigned increases each retransmitted packet until it reaches the value of the last sent packet before the retransmission request. Signal retransmit_read_count is also a counter which scans the words in the packet. It is only used to address RAM.

Packet Sender enables RAM writing whenever a packet is sent and disables it during retransmission.

Each packet has an index (referred as TX_DATA_num in the waveform 6.17): this index is in the header of each packet and it is used by the DTC to know which number of packet require in case of retransmission request. Without retransmissions, that index match with resp_packet_count, while when a retransmission occurs, that index matches with the index of the last sent packets before retransmission. In case of retransmission, resp_packet_count will always be greater of the index on packet because the actual number of the total sent packets is the sum of transmitted and retransmitted packet.

State Machine

Each main component of the ROC handles data requests and makes decisions using state machines. Retransmission is a sequence in the Packet Sender state machine. The main signals involved are:

```
retransmit_det_sig                : std_logic;
retransmit_packet_req             : std_logic_vector(2 downto 0);
retransmit_packet_req_unsigned    : unsigned(2 downto 0);
sendCnt                           : unsigned(4 downto 0);
retransmit_read_count             : unsigned(3 downto 0);
retransmit_word_count             : unsigned(3 downto 0);
retransmit_word_count_latch       : unsigned(3 downto 0);
retransmitCnt                     : unsigned(7 downto 0);
rsp_packet_for_retransmit         : unsigned(2 downto 0);
retransmit_txdata_reg             : std_logic_vector(15 downto 0);
```

First of all, retransmit_det_sig and retransmit_packet_req are the latch that sample RETRANSMIT_DETECTED and RETRANSMIT_SEQUENCE_REQ from Forward Detector. When a retransmission request occurs, retransmit_det_sig is set to 1 and retransmit_packet_req stores the index of packet DTC is asking for. Retransmit_txdata_reg are data replies for DTC and are send to the RAM too.

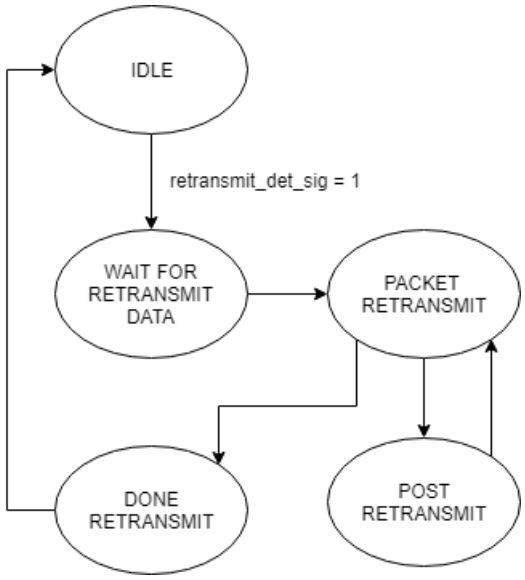


Figure 6.18: State diagram of retransmission

As shown in the state diagram 6.18, starting from an Idle state, Packet Sender enters in a state of WaitForRetransmitData when retransmit_det_sig is set to 1, so when a retransmission request occurs. SendCnt is set to 1 and retransmitCnt is set to hexadecimal "FF". It will be the counter that will manage the retransmission. In WaitForRetransmitData, retransmitCnt starts to decrease and gives timing to the following operation: retransmit_packet_req_unsigned takes index of the packet to retransmit, the manual reset is handled, retransmit_read_count is cleared and the transaction signal is updated to allow the state change to packetRetrasmit. In PacketRetransmit RAM is addressed and read, so that data can come out; if the retransmission asks for multiple packets, next state is PostRetransmit and then back to PacketRetransmit in an iterative way, otherwise, next state is

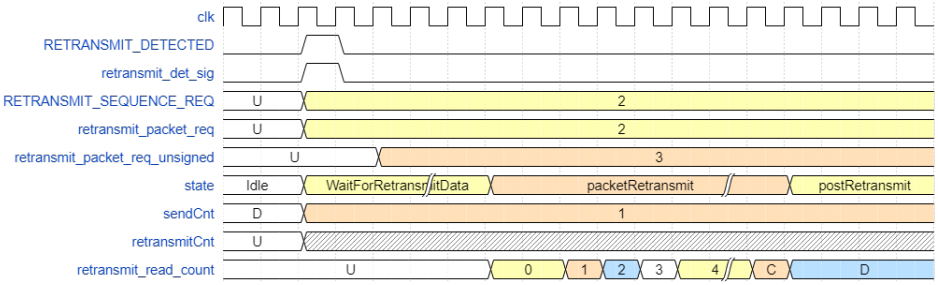


Figure 6.19: State changes: when a retransmission request is detected, state goes from Idle to WaitForRetransmitData and the counter *retransmitCnt* starts giving time. When each signal is set, state changes to postRetransmit where the actual retransmission happens.

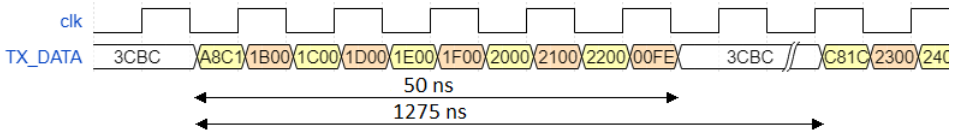


Figure 6.20: Timing specification in TX DATA

DoneRetransmit. Signal retransmit_read_count handles access to the RAM during the reading. When the retransmission is done, the last state is Idle.

Timing

An important requirement is the speed of retransmission. In a block read, a reply is sent every 1275 ns. During the retransmission, retransmitted replies maintain almost the same rate: 1270 ns. Read data waiting for the end of the retransmission are considered pendent. So, when the retransmission is done, pending data are transmitted every 90 ns. When all pending data are transmitted, replies start again with a rate of 1275 ns. The worst case in timing constraint is a retransmission request of 8 packets; retransmission request cannot be for more than 8 packets because of the depth of the RAM. Sending 8 packets requires $8 * 50ns + 7 * 1225ns = 8975ns$, where, as shown in figure 6.20, 50 ns is the length of a packet of 10 words and 1225 ns is the time between two packets. Retransmit 8 packets and send the 8 pending data requires $8*1270ns+7*90ns+50ns = 10480ns$. So, in the worst case, retransmission will take $10480ns - 8975ns = 1505ns$ extra time.

Managment of the Reset

When a reset occurs, all signals are set to 0. This implies that the writing address starts again from 1 and data are overwritten. Counters of transmitted packet are cleared and the index in the header of TX_DATA starts from 1. There are two possible reset: FRABRIC_RESET and a soft reset. The first one set all signals and clear everything in the first interaction with the firmware after the download on the FPGA. It only happens once. The second one is the one capable to clear signals and registers when needed. It can be access by writing in a dedicated register.

Referring to the soft reset there are 2 tricky scenarios: a retransmission request right after a read request and a retransmission request after some read requests. In the first scenario this is what happens: with the reset everything is set to

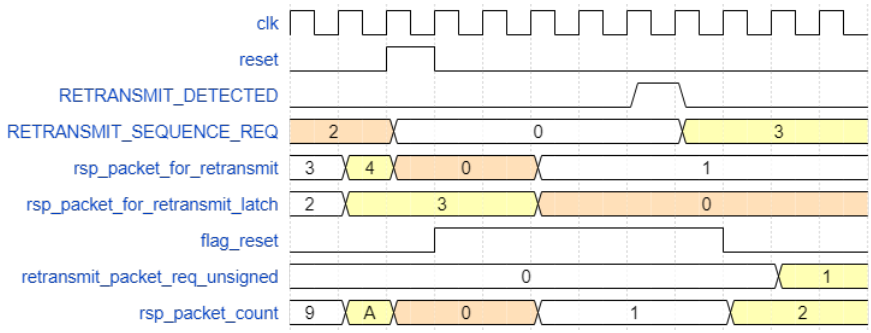


Figure 6.21: Waveform and timing of signals which handle the soft reset

0, so the reply to the following read request is stored at address 1 of the RAM. Before the reset, the writing address of the RAM could be any between 0 and 7, so the last sent reply before the reset could have any index between 0 and 7. For example a possible sequence would be: DTC asks for data, ROC replies with packet number 4, a reset occurs, DTC asks for new data, ROC replies with packet number 1. If DTC asks for a retransmission on packet number 1, it will send as RETRANSMIT_SEQUENCE_REQ the last uncorrupted data it received, that is 4. ROC then starts to retransmit packet from address 5 to address 1 of the RAM, while in the correct behavior DTC should receive only packet number 1. First step to handle this event is to detect a condition which identify the sequence "retransmission request after read request after soft request". Soft reset and FRABRIC_RESET show the same effects on signal, so it is not easy to isolate the sequence. Signal rsp_packet_for_retransmit stores the index of sent data and it always increase unless is set to zero because of a reset. Monitoring this signal gives a condition on the reset. Another condition is necessary to distinguish the soft reset from the FRABRIC_RESET. This condition is met using a latch on rsp_packet_for_retransmit which indicates an undefined value before a fabric reset and the previous value of rsp_packet_for_retransmit before a soft reset; with this discrepancy is possible to build a conditional statement. When the sequence is detected, a flag_reset is raised. In WaitForRetransmitData flag_reset is checked and if it is high, retransmit_packet_req_unsigned is forced to 1. This means that, whenever RETRANSMIT_SEQUENCE_REQ is, ROC will always send data stored in address 1 of the RAM.

The other tricky scenario is when, after a soft reset, there is a retransmission request after some read requests. When the soft reset occurs, flag_reset is high and forces retransmit_packet_req_unsigned to 1. This is a wrong behavior when the retransmission request actually asks for a correct RETRANSMIT_SEQUENCE_REQ. It is necessary to set flag_reset back to 0 when there is more than one read request. The condition is met with a check on rsp_packet_count. When there is more than one sent packet, so more than one read, flag reset is set to 0.

6.3 ROCs Synchronization

The most challenging aspects of the system is the synchronization of all subsystems. The Tracker is made of approximately 2000 straws arranged along 18 stations across the 3 m Tracker length. There are 216 Readout Controllers (one for each panel) located inside the cryostat. The Calorimeter is composed by 1610 crystals in 2 disks. There are 192 Readout Controllers located inside the cryostat. Detector dimension are such that ROC are physically placed even at a distance of meters one from each other. Despite their location, ROC must be able to recognized a single event and to label it with the right temporal reference. This different of location translates also in different path data must cross via the optical fibers from the ROC to the TDAQ. Indeed the TDAQ system will be located in the surface level electronics room and connected to the detector by optical fiber, which can have a lenght difference of meters.

First trick adopted to face this challenge is to distribute a common System Clock to all the subsystem to allow alignment. The TDAQ will generate a continuous Mu2e System Clock with frequency of 40 MHz at the Run Control Host Clock Fanout module (CFO). The CFO also receives the RF-cavity 0-crossing marker from the Accelerator. ROCs receive and phase aligns as necessary the encoded System Clock which carries a marker. The marker encoded on the System Clock is the time-zero reference for an event window (i.e. 1695 ns μBunch during beam ON).

Requirements on the synchronization are such that the scheme must accommodate links of different length, must address phase differences between ROC FPGAs, must handle loss of lock on the links with the final goal of an intrinsic precision of the Reference Clock at the ROCs better than 500 ps [7].

6.3.1 Loopback

All timestamping frontends extract a System Clock and start of Event Window marker. Event Windows need to be synchronized: T=0 defines the start of a Mu2e Event Window and occurs in response to each start-of-event-window marker. Signals travel to different boards and through fibers and cables of different lengths and they must line up the Event Window at timestamping front ends. To line up Event Windows the approach is to delay each front end to match front end with longest latency. Delay is determined calculating the Loopback signal. It consists in determine the round trip time by sending a marker and returning it many times and taking the average time. Once the timing path latency is known for each front end, it is possible to calculate each front end's delay offset define as "longest timing path latency" minus "individual timing path". At the start of each run, the front ends can be configured to apply their own delta delay.

The ROC synchronization procedure provides measurement of the signal propagation time through fibers and electronics by using the loopback technique and, based on the loopback measurement, set coarse (5 ns) plus fine (250 ps) delays at ROC to match the longest path. At the end of this procedure, T=0 (i.e., the beginning of the Event Window) will be synchronized across all ROCs with an expectation of $\pm$ 250 ps.

On the ROC firmware, the loopback signal is treated like a marker. Loopback corresponds to 0x1C12 and 0x1CED; once the ROC detects such sequence, simply sends back to the DTC the same marker. This way is possible to calculate

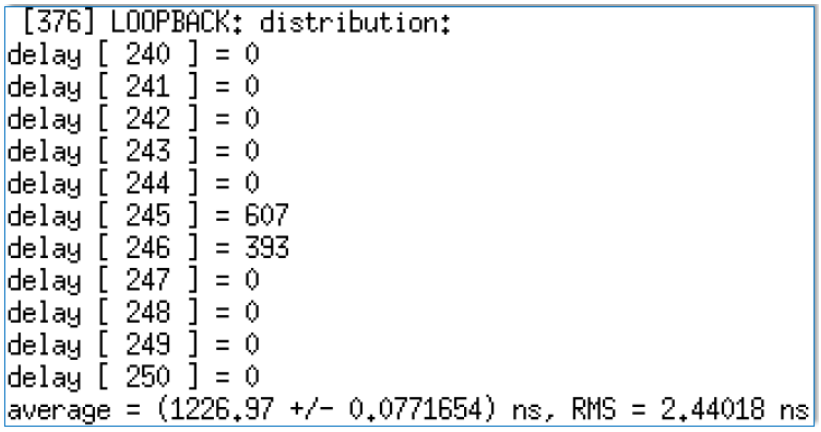


Figure 6.22: Result of a loopback measurement. This screen shows how the delay is around 245 and 246, that correspond to an average of 1226.97 ns with an error of $\pm$ 0.0771 ns.

difference between marker sent and received: this is the latency to cross the ROC Firmware. A similar procedure is repeated in the DTC and CFO firmware. Operatively the measurement is driven by software:

1. Establish links: the connection is achieved configuring the DTC via *ots*.
2. Put DTC and ROC into "loopback mode".
3. CFO sends loopback signal.
4. CFO measures time for loopback signal to return.
5. Record value.
6. Go to 2. Repeat N times.
7. Make distribution of values.

The result of the measurement is an average value in which the uncertainty of one measurement is equal to the RMS (Root Mean Square) as shown in figure 6.22.

Once the loopback is measured, it gives information about the delay of the data path. It needs to be stable and reproducible in order to be reliable. Loopback stability implies link phase stability (from lock to lock) and vice versa. If the system clock marker obtained from optical link is correct, it implies link phase stability and system clock stability, so the loopback measurement for a given setup must be reproducible. During the completion of this thesis the two terms of this statement have been tested. First term says that with a given setup the loopback measurement must be always the same. Reproducing the measurement, results are always different. This implies that the something is wrong: there could be an error in the measurement procedure, in the ROC firmware, in the system clock distribution, in marker detection and so on. Some errors have been excluded performing a measurement in loop. Disconnecting the ROC and closing the optical fibers in loop on the DTC, the value of the delay is always the same for the given setup. This last test reduces the area of investigation for the error to the ROC and how it manages the loopback signal.

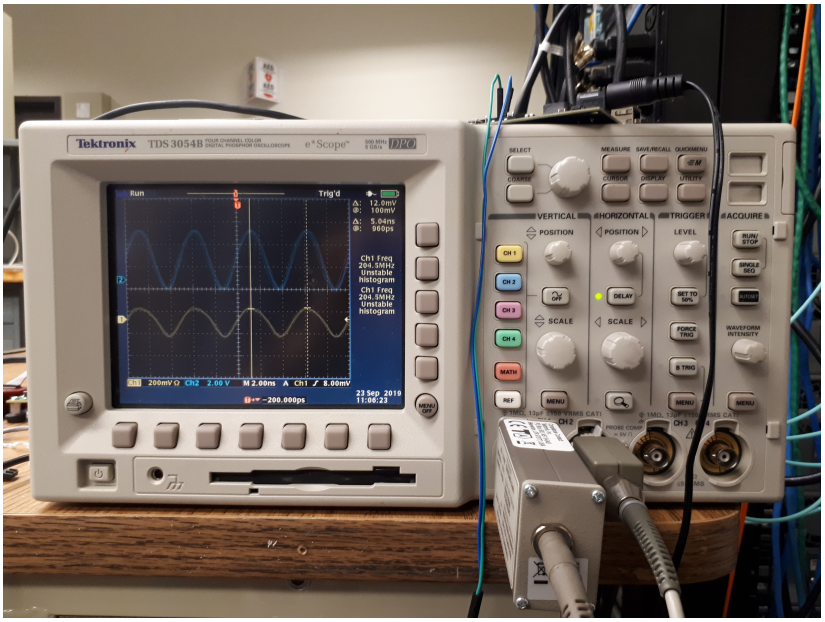


Figure 6.23: On the oscilloscope are shown the DTC clock in blue and the ROC clock in yellow. They have the same frequency and, at each reconfiguration, also the same relative phase.

Also the other term of the statement has been tested. It says that if the system clock marker obtained from optical link is correct, the link phase and the system clock are stable. To observe the stability of phase and frequency of the clock, the oscilloscope has been used. The test consists in the observation of the clock coming from the DTC and the one that ROC retrieves from the System Clock and recovers. This two clocks, compared on the screen of the oscilloscope must have the same frequency and the same relative phase each time the ROC tries to realign or the DTC is reconfigured. Realignment is simply performed by disconnecting and reconnecting the fibers from the VTRx on the ROC. This test has shown that the relative phase is always the same, that is what is expected with a correctly functioning system.

The statement: “if the system clock marker obtained from optical link is correct, link phase and system clock are stable, so the loopback measurement for a given setup must be reproducible ”if and only if” statement, so both terms must be verified. At the time of this thesis, only one of the two results true and it is up to future development of the system to fix this synchronization problem.

6.3.2 Timestamping

Timestamping is fundamental in the system to elaborate data and allow the reconstruction of the event. Each event is labeled with the event window of belonging and with a relative time indicator. The event window is passed to the ROC via the 16 byte Heartbeat packet describing the next event window before each event window begins. This packet includes an 8 bit TDC value predicting the relative phase of the next event window and the System Clock to better than 1 ns resolution. This packet also provides “live gate” info and a 48 bit Mu2e System Timestamp (no wrap-around for 15 years) labeling the next event window. It can be stopped and restarted at any value as long as the new start value is higher

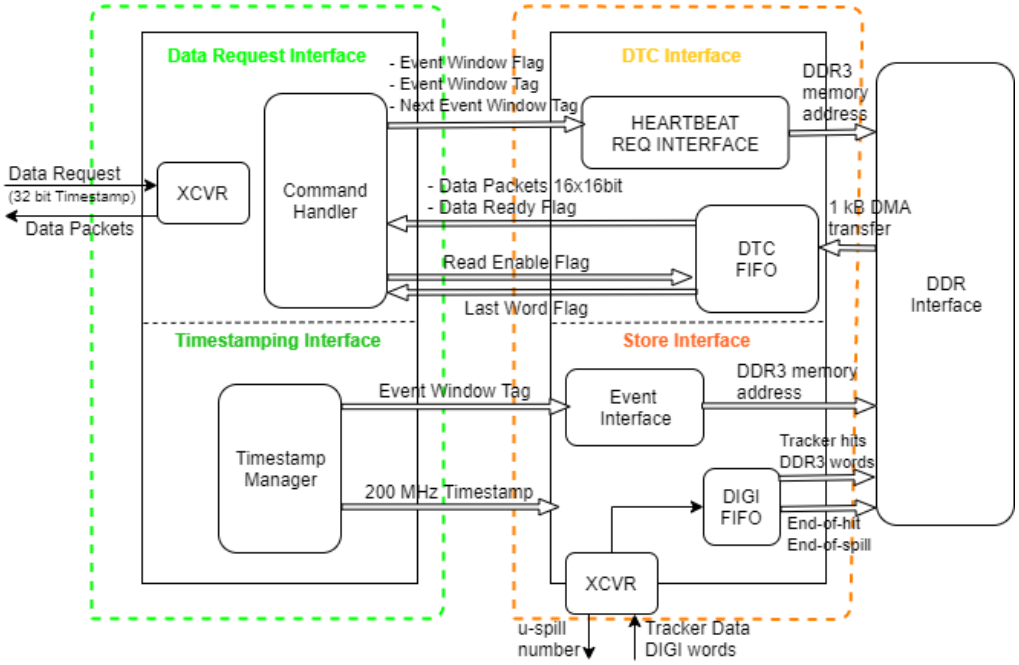


Figure 6.24: Block diagram of the firmware handling the timestamp. In figure is shown the interface between the SERDES firmware (in green) and the one serving the DDR3 (in orange) and the signals exchanged between them.

than the previous stop value. The System Timestamp can be correlated with actual calendar time, or the high bytes can represent a Run Number, supercycle, etc. The front ends can use the information distributed from the TDAQ however needed (i.e. if 10 ns resolution is enough, the front ends can ignore the 8 bit TDC value).

Each Readout Controller generates its own internal timestamp for data within an event window. This internal timestamp counter is driven by the digitization clock and is reset at the beginning of an event window. It may be 1 or 2 bytes, depending on the resolution of the detector.

ROC firmware is already developed to interpret the event window tag and the System Timestamp and to produce his internal timestamp. This labels are necessary to address the DDR3 in an efficient way.

Figure 6.24 shows a block diagram of the firmware which handles this aspect. The bottom part of the diagram deals with the logic needed to use the event window tag and the internal timestamp to label data. The top part of the diagram deals with the logic needed to service a DTC Packet Request. It is the one responsible to retrieve from an Heartbeat the information about the event window and the System Timestamp.

At this point on the firmware development, the interface is established but not implemented, so the addressing of the DDR3 and the labels on the packet are still not present. The one described now are ideas on how to implement the system.

At the beginning of each spill, the Timestamping Interface sends the μspill number, or event window tag, to the Store Interface which generates the DDR3 memory address. At this adress are stored the Tracker hits received during that spill. To address memory is necessary to combine information of the event window tag with the internal timestamp. Probably the internal timestamp will be an

information added also on the packet of the data event. Another information necessary for the interface is the number of data for each hint. It is necessary to know how many data to read from the DDR3 once a data event is requested from the DTC.

Right now, on a preliminary version of the system: the hits from the DIGI FPGAs are stored in the DIGI FIFO. Based on the FIFO_EMPTY signal, on the number of 64 bit words available to be read and on the detection of the end-of-spill signal, the DDR3 Interface will start the write to memory. The idea is writing all of the hits observed in one spill using a single 1 KByte memory block write. For standard hits with 12 ADC samples corresponding to 4x64 bit per hit to the DDR3, 1 KByte block will fit 32 hits. In 6 Gb memory, we will be able to fit $\sim$0.75M spill, allowing for almost 1.3 s latency for the TDAQ before the memory will fill up, assuming continuous hit collections every 1.7 μs even during off-spill time.

Chapter 7

Conclusions and future prospects

The beginning of Mu2e data taking is expected in the year 2022. My work of thesis has contributed to the progress of the firmware development for the two detector of the Mu2e: the straw-tracker and a crystal electromagnetic calorimeter. I have worked with the INFN (Italian National Institute of Nuclear Physics), University of Pisa and Fermilab to converge different works to a common point and lay the foundation to the next step into the development of the experiment. My work has been divided into three main phases. First one, which took place at Fermilab, was about learning the system. Here I had the task to learn about the TDAQ environment and be able to use it and reproduce it. Fermilab has at its disposal a Test Stand which reproduces the electronic infrastructure necessary to test the system. Once learned about the goals of the experiment and how the TDAQ was supposed to work, I had the chance to start working on the firmware of the ROC. During my time in Fermilab I had the task to make the link between ROC and DTC work. During my time of developing, I met with the TDAQ group on a biweekly basis to discuss the design choices. At the end of my period at Fermilab, the optical link between ROC and DTC was established and working. This result has involved both VHDL programming via Libero SoC and knowledge of high level language to make *ots* and Scientific Linux works on the server and even familiarity with debugging tools like Vivado, Active HDL and Identify. My main task was the developing of SERDES, but I had the opportunity to interface with the team working on the DIRAC firmware from the University of Michigan, being involved in the development of the firmware handling the DDR3. Much of this work was about design choices, interface management and tests.

Second phase of my work was about reproduction of the Fermilab Test Stand in the INFN laboratories of Pisa. The calorimeter has been designed and will be built by the collaboration among the INFN, the California Institute of Technology (Caltech) and the Fermilab. To enable tests on the electronic of the Calorimeter, it is necessary to have a functioning Test Stand in Pisa and have the understanding of how to use it. The core of my task in this phase was to recreate the Test Stand, make it works and collaborate with the INFN group to transfer to them the knowledge gained at Fermilab. This phase was the most technical because was about the selection of the hardware to compose the server and the installation of Scientific Linux.

Third phase was about hardware validation on the DIRAC. DIRAC is the

Calorimeter readout controller board developed by the INFN group and it is at his second version. The validation has been made at Fermilab, to share the results with the Fermilab TDAQ team and to use the complete Test Stand. The main tests were about the optical connection, the DDR3 memory and the FPGA mapping. All test has been completed and have led to good results.

A conclusion of this work of thesis, a fourth phase has been accomplished: reproduce the hardware validation of the DIRAC at the Test Stand in Pisa. This task has a dual reason: on one side having a double validation with a slightly different environment strengthens the obtained results and on the other side it demonstrate the validity of the Test Stand in Pisa. All the test taken at Fermilab has been reproduced with the same results.

The end of my work has marked the point in which INFN and Fermilab have the same infrastructure to work on the experiment in parallel and to reduce significantly times and spaces. Before now, every test behaved reaching Fermilab, with a large use of time, energy and money. My thesis allowed the validation of the DIRAC hardware, laying the groundwork for the future firmware development.

Right now the firmware has the skeleton of all the functionality needed for the system. Limited to the firmware on the DIRAC and the DRAC, there are several steps to carry on the project:

1. Solve the bug on the loopback measurement. The ROC synchronization procedure provides measurement of the signal propagation time through fibers and electronics by using the loopback technique and, based on the loopback measurement, set delays at ROC to match the longest path. At the end of this procedure, T=0 (i.e., the beginning of the Event Window) will be synchronized across all ROCs with an expectation of $\pm$ 250 ps. With a given setup a correctly working loopback measurement must always gives the same value. At the end of my work, this measurement was not working: several measurement gave always different results.

2. Complete the test on the interface between SERDES firmware and the DDR3 Interface firmware. As shown in figure 6.24, the interface has been implemented but still not tested. Next step is to test the exchange of signals between the two borders.

3. Define DDR3 addressing. This will be different on DIRAC and DRAC because data are different in content and quantity. The addressing will be based on the Event Window Tag and on the internal timestamp. It must be a smart addressing both to accommodate the DDR3 architecture and to meet the TDAQ requirements. Indeed the memory efficiency can be maximized optimizing reading and writing with the refreshes. As regards the TDAQ, data will be requested stating by their labels. Matching labels with addresses could speed up the storing and the retrieving. A big design choice will be on how to handle this aspect.

4. Calculate in a more accurate way the effective amount of data associated with an event to study, in temporal terms, the DDR3 depth. How much time of data are the ROC able to store? Based on this calculation will be decided how long the packet of the Calorimeter will be and how many information can be stored. This is a trade off between the data throughput and the time necessary to collect all the event.

5. Eliminate the microprocessor from the firmware. Right now it is instantiated to debug the DDR3. Whit a more complete version of the firmware, the system will be able to full the memory with data, to ask for them and to read them via *ots*. At the moment, a Pyhton script handles this aspect.

6. Polish the SERDES firmware. Creating several version has led to layering of unused structure. All the unnecessary needs to be removed from the firmware to have a clean version easier to manage and more efficient in terms of synthesis.

7. Study the use of resources on the FPGA in order to optimize. A timing analysis is necessary. An example of optimization is the encoding of states in the state machines of the different blocks. Right now it is left to the synthesizer, but a one hot encoding provided by designer can be the best suited.

The points listed above are next nearest step, but more and more steps will arise gradually with the development. Being part of a larger system, not only the firmware needs to be carried on, but all the Test Stand needs to grow up with it. First of all *otsdaq*. During my work I had to update and create new structure on *ots* to provide support on the firmware tests. Also the DTC firmware needs a continuous debugging and updating. Next goals will be the Integration Meeting in March and the Test Beam. This setpoint will mark next challenges on the system development.

Chapter 8

Appendix

8.1 Libero SoC

Microsemi Libero System-on-Chip (SoC) design suite [21] is being used to develop the firmware. The first firmware was developed on the v12.1 version. The DIRAC's firmware has been updated to the v12.2 version; this update resulted in an update of most of the IP cores. Libero is not backwards compatible. Last version of the firmware had been imported in Libero SoC v12.3, released on 10 December 2019. The decision of continuously update the the firmware to the latest Libero release is based on two aspects: the first one is the new release solve some bug of the tool helping in the development, the second one is that the IP cores in the project risk to not work anymore falling behind updates. It easier to go to consequential update compared to use an old version of the project and at the end update that to the newest release.

The suite integrates industry standard Synopsys Synplify Pro synthesis and Mentor Graphics ModelSim simulation with constraints management, debug capabilities, and secure production programming support.

The Design Flow includes the following design steps: create, constrain, implement, configure hardware, program design, debug design, handoff.

First step consist in the project creation. Project specifications are listed in the report 8.1.

Family	PolarFire
Device	MPF300TS_ES
Package	FCG484
Speed Grade	STD
Core Voltage	1.0V
Part Range	EXT
Default I/O technology	LVCMOS 1.8V
Restrict Probe Pins	Yes

Table 8.1: Project specification.

The Libero screen looks like in figure 8.1. On the left is situated the Design Flow. Main step in the Design Flow are:

- Simulation

- Synthesize
- Manage Constraints
- Pin assignment
- Place and Route
- Verify Timing
- Verify Power
- Programming (Generate FPGA Array Data, Generate Bitstream)
- Debug Design (SmartDebug, Identify Design)
- Export (Pin Report, BSDL File)

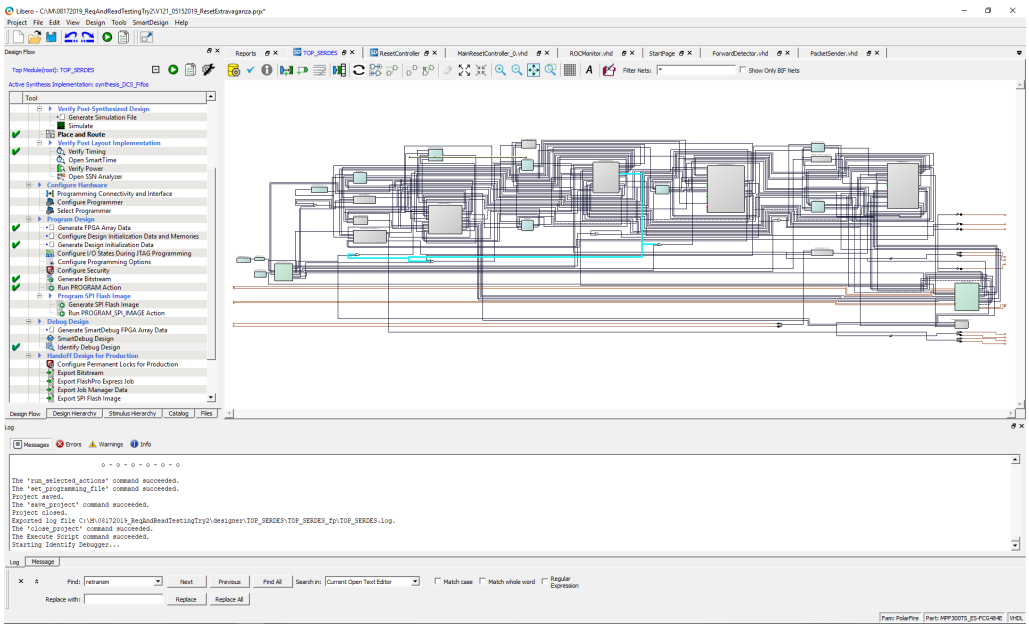


Figure 8.1: Screenshot of the project on Libero.

Almost each one of this step results in a reports. The Design Report Tab lists all the reports available for the design, and displays the selected report.

Constraint Manager

Defined project specifications, next step is the detection of timing constraints [20]. Constraint files are as important as design source files; they are used throughout the FPGA design process to guide FPGA tools to achieve the timing and power requirements of the design. For the synthesis step, SDC timing constraints set the performance goals whereas non-timing FDC constraints guide the synthesis tool for optimization. For the Place-and-Route step, SDC timing constraints guide the tool to achieve the timing requirements whereas Physical Design Constraints (PDC) guide the tool for optimized placement and routing (Floorplanning). For Static Timing Analysis, SDC timing constraints set the timing requirements and design-specific timing exceptions for static timing analysis. Libero SoC provides the Constraint Manager as the cockpit to manage the design constraint needs. In the project are used both derived constraints generated from Libero than peculiar

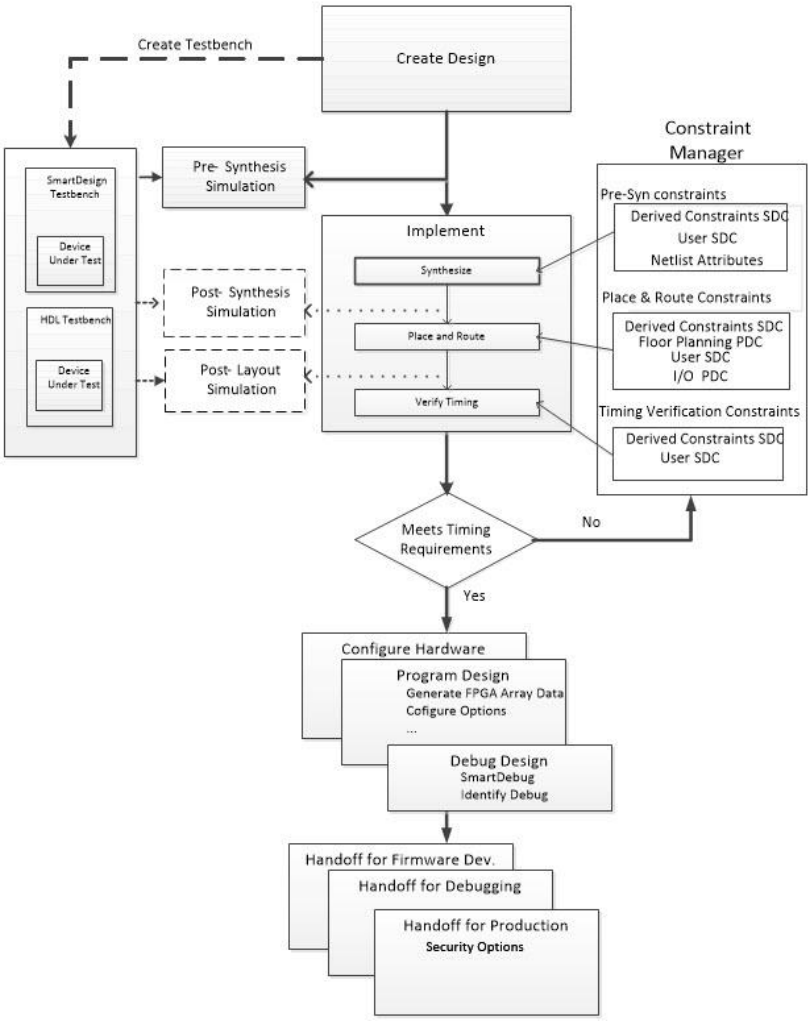


Figure 8.2: Libero SoC Design Flow

constraint defined for the system. Constraint management looks like a single centralized graphical interface to create, import, link, check, delete, edit design constraints and associate the constraint files to design tools in the Libero SoC environment. The Constraint Manager allows to manage constraints for SynplifyPro synthesis, Libero SoC Place-and-Route and the SmartTime Timing Analysis throughout the design process. About Synthesis Constraints, the Constraint Manager manages these synthesis constraints and passes them to SynplifyPro: Synplify Netlist Constraint File (*.fdc), Compile Netlist Constraint File (*.ndc), SDC Timing Constraints (*.sdc), Derived Timing Constraints (*.sdc). About Place and Route Constraints, the Constraint Manager manages these constraints for the Place-and-Route step: I/O PDC Constraints (*io.pdc), Floorplanning PDC Constraints (*fp.pdc), Timing SDC constraint file (*.sdc). The Constraint Manager manages the SDC timing constraints for Libero SoC's SmartTime, which is a Timing Verifications/Static Timing analysis tool. SDC timing constraints provide the timing requirements (e.g. create_clock and create_generated_clock) and design-specific timing exceptions (e.g. set_false_path and set_multicycle_path) for Timing Analysis.

Libero SoC manages four different types of constraints:

- I/O Attributes Constraints – Used to constrain placed I/Os in the design. Examples include setting I/O standards, I/O banks, and assignment to Package Pins, output drive, and so on. These constraints are used by Place and Route.

- Timing Constraints – Specific to the design set to meet the timing requirements of the design, such as clock constraints, timing exception constraints, and disabling certain timing arcs. These constraints are passed to Synthesis, Place and Route, and Timing Verification.

- Floor Planner Constraints – Non-timing floorplanning constraints created by the user or Chip Planner and passed to Place and Route to improve Quality of Routing.

- Netlist Attributes - Microsemi-specific attributes that direct the Synthesis tool to synthesize/optimize the, leveraging the architectural features of the Microsemi devices. Examples include setting the fanout limits, specifying the implementation of a RAM, and so on. These constraints are passed to the Synthesis tool only.

About the I/O attribute, in the following tables (table8.2 and 8.3)are listed how banks are defined in the DIRAC firmware and how pin are assigned.

Bank	**VDDI**	**Vref**	**AutoCalib**
0	1.5	0.75	50 ms
1	1.8	0.9	50 ms
2	3.3		50 ms
3	3.3		
4	3.3		50 ms
5	3.3		50 ms
6	1.8	Unassigned	50 ms
7	1.5	0.75	50 ms

Table 8.2: Bank definition.

PIN	**name**
K2	STAMP_P
K3	STAMP_N
H2	CLK_P
H1	CLK_N
L3	RX
N3	TX
J3	calsda
J4	hvsda

Table 8.3: PIN assignment.

Synthesize
Double-clicking Synthesize runs synthesis on the design with the default settings

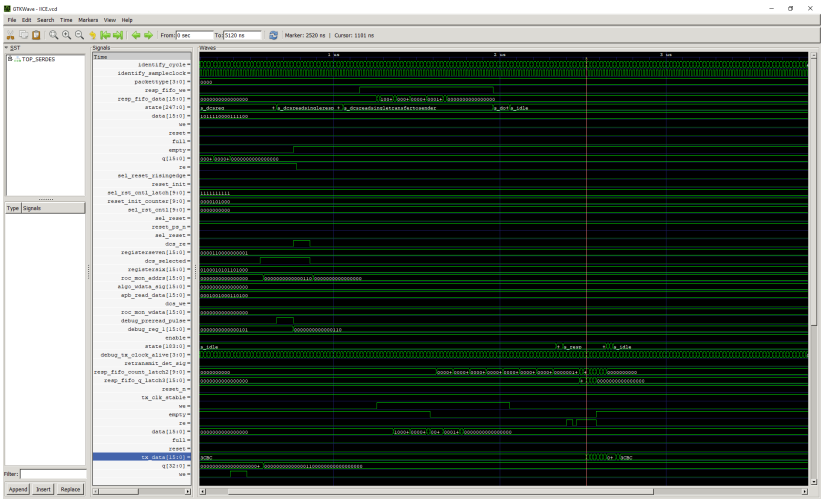


Figure 8.3: Screenshot on Identify Debug Design: it shows the waveform of the instrumented signals.

specified in the synthesis tool. Opening the tool interactively, synthesis must be completed from within the synthesis tool. The default synthesis tool included with Libero SoC is Synplify Pro ME.

Synplify Pro ME is the default synthesis tool for Libero SoC. Synplify compiles and synthesizes the design into an HDL netlist. The resulting *.vm files are visible in the Files list, under Synthesis Files.

Identify Instrumentor

Once created the HDL, the instrumentor [15] is used to define the specific signals to be monitored. Saving the instrumented design generates an instrumentation design constraints (idc) file and adds constraint files to the HDL source for the instrumented signals and break points. The design is synthesized and then run through the remainder of the process. After the device is programmed with the debuggable design, the debugger is launched to debug the design while it is running in the target system. The information required to instrument a design includes references to the HDL design source, the user-selected instrumentation, the settings used to create the Intelligent In-Circuit Emulator (IICE), and other system settings.

The IICE parameters determine the implementation of one or more IICE units and configure the units so that proper communication can be established with the debugger. Multiple IICE units allow triggering and sampling of signals from different clock domains within a design. Each IICE unit is independent and can have unique IICE parameter settings including sample depth, sampling/triggering options, and sample clock and clock edge. During the subsequent debugging phase, individual or multiple IICE units can be armed.

The IICE parameters common to all IICE units defined for an instrumentation include the IICE device family as defined by the synthesis tool, the communication port and if optional skew-resistant hardware is to be used. All IICE units in a multi-IICE configuration share these same parameter values. In this case the device family is PolarFire. The Communication port parameter specifies the type of connection to be used to communicate with the on-chip IICE. The connection

types used is *builtin*: indicates that the IICE is connected to the JTAG Tap controller available on the target device.

The IICE type parameter is a read-only field that specifies the type of IICE unit currently selected – regular (the default) or rtd (real-time debugging). The Buffer type parameter specifies the type of RAM to be used to capture the on-chip signal data. In this instrumentation is set to FPGA Memory. The Sample depth parameter specifies the amount of data captured for each sampled signal. Sample depth is limited by the capacity of the FPGAs implementing the design, but must be at least 8 due to the pipelined architecture of the IICE. Sample depth can be maximized by taking into account the amount of RAM available on the FPGA. As an example, if only a small amount of block RAM is used in the design, then a large amount of signal data can be captured into block RAM. If most of the block RAM is used for the design, then only a small amount is available to be used for signal data. In this case, it may be more advantageous to use logic RAM. In this instrumentation sample depth is 128.

The Sample clock parameter determines when signal data is captured by the IICE. The sample clock can be any signal in the design that is a single-bit scalar type. Care must be taken when selecting a sample clock because signals are sampled on an edge of the clock. For the sample values to be valid, the signals being sampled must be stable when the specified edge of the sample clock occurs. Usually, the sample clock is either the same clock that the sampled signals are synchronous with or a multiple of that clock. The sample clock must use a scalar, global clock resource of the chip and should be the highest clock frequency available in the design. The source of the clock must be the output from a BUFG/IBUFG device. The Clock edge radio buttons determine if samples are taken on the rising (positive) or falling (negative) edge of the sample clock. The default is the positive edge.

The IICE Controller tab selects the IICE controller's triggering mode. All of these instrumentation choices have a corresponding effect on the area cost of the IICE.

SmartDebug Design

Microsemi's SmartDebug [18] tool complements design simulation by allowing verification and troubleshooting at the hardware level. SmartDebug provides access to non-volatile memory (sNVM), SRAM, transceiver, uPROM, and probe capabilities. PolarFire have built-in probe logic that SmartDebug can access through the Active Probe and Live Probe features, which enables designers to check the state of inputs and outputs in real-time without re-layout of the design.

Identify Debug Design

Before a design can be debugged, the instrumentor is first used to define the specific signals to be monitored and then to generate an instrumentation design constraints (idc) file containing the instrumented signals and break points. The design is synthesized and the device is programmed with the debuggable design. The debugger is then launched to analyze the design while it is running in the target system [12]. Figure 8.3 shows a screenshot of a test on Identify. It allows to see waveform of the instrumented signals in a window of depth defined during the instrumentation (see section Identify Instrumentor). Still during the instrumentation are defined watchpoints and breakpoints, which purpose is to

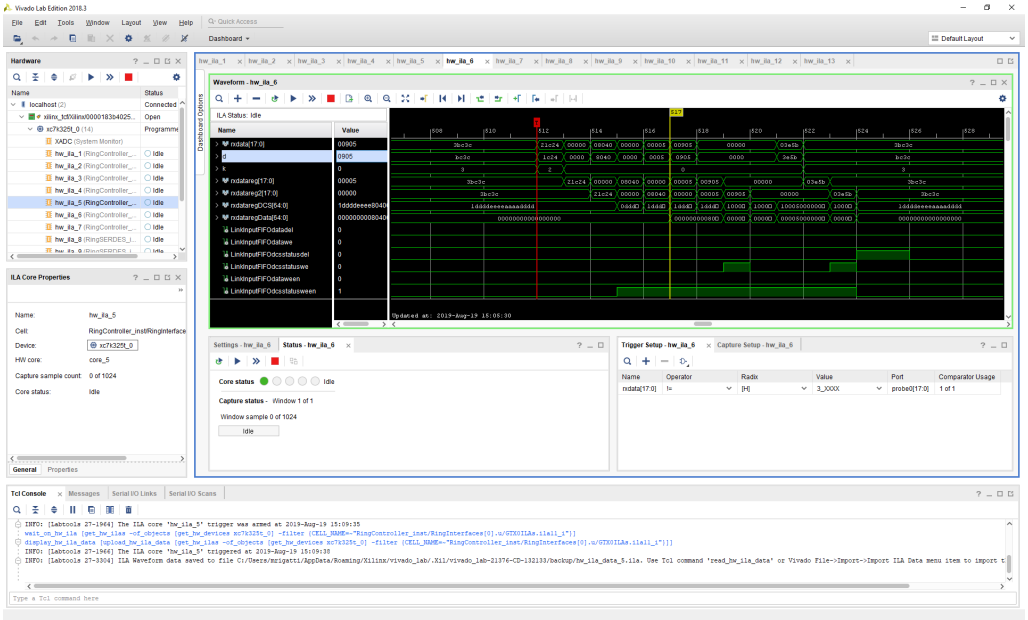


Figure 8.4: Screenshot of the waveform window in the logic analyzer of Vivado. It shows a packet in the communication between DTC and ROC.

cause a trigger event to occur. The trigger event causes sampling to terminate in a controlled fashion. Once sampling terminates, the data in the sample buffer is communicated to the debugger and then displayed in the GUI. The sample buffer is continuously sampling the design signals. Consequently, the exact relationship between the trigger event and the termination of the sampling can be controlled by the user.

8.2 Vivado

The Vivado Design Suite is architected to increase the overall productivity for designing, integrating, and implementing systems using the Xilinx UltraScale and 7 series devices, Zynq UltraScale+ MPSoC device, and Zynq-7000 All Programmable (AP) SoC [36]. In this project has been used to program the DTC, that mounts a Xilinx K325T Kintex-7 FPGA and to debug the fiber link between DTC and ROC. Figure 8.4 shows a screenshot of a test on the communication between DTC and ROC, in particular a read request on a ROC's register. Version used are the 2018.1 and the 2019.2, that is the one with the last release of the DTC Firmware. Of the Suite, only the Hardware Manager has been used for the purposes of this thesis. The Hardware Manager handles the customizable Integrated Logic Analyzer (ILA) IP core [37], a logic analyzer that can be used to monitor the internal signals of a design. The ILA core includes many advanced features of modern logic analyzers, including boolean trigger equations and edge transition triggers. Signals in the FPGA design are connected to ILA core clock and probe inputs. These signals, attached to the probe inputs, are sampled at design speeds and stored using on-chip block RAM (BRAM). The core parameters specify the number of probes, trace sample depth, and the width for each probe input. Communication with the ILA core is conducted using an auto-instantiated debug core hub that connects to the JTAG interface of the FPGA.

After the design is loaded into the FPGA, via the logic analyzer software a trigger event for the ILA measurement can be set. After the trigger occurs, the sample buffer is filled and uploaded into the Vivado logic analyzer. You can view this data using the waveform window as show in figure 8.4. Regular FPGA logic is used to implement the probe sample and trigger functionality. On-chip block RAM memory stores the data until it is uploaded by the software.

In the project data from and to DTC was monitored thanks to an instrumentation on the DTC FPGA. Via Vivado was possible to see the packets used in the DTC ROC communication. The trigger was set on the idle word 3BC3C, or in all other significant word able to give information about the status of the link. The clk input port is the clock used by the ILA core to register the probe values. It is the same clock signal that is synchronous to the design logic that is attached to the probe ports of the ILA core. For this reason, DTC needs to be configured before using the logic analyzer. Configuration is set via *ots*.

Bibliography

[1] Xilinx - Marc Defossez. *Bitslip in Logic*. 2014 (cit. on p. 83).

[2] For the United States Department of Energy Fermi Research Alliance FRA, ed. *M2e Technical Design Report*. 2014 (cit. on pp. 6, 13, 27).

[3] United States Department of Energy Fermilab. *Accelerators*. 2020. URL: `https://fnal.gov/pub/science/particle-accelerators/index.html` (cit. on p. 5).

[4] United States Department of Energy Fermilab. *Fermilab History*. 2020. URL: `https://history.fnal.gov/index.html` (cit. on pp. 1, 5).

[5] United States Department of Energy Fermilab. *Mu2e Experiment*. 2020. URL: `https://mu2e.fnal.gov/` (cit. on p. 4).

[6] Rick Kwarciany Glenn Horton-Smith Eric Flumerfelt. *Trigger and DAQ Architecture and Status*. Mu2e Collaboration Meeting, Fermilab, United States Department of Energy. 2019 (cit. on p. 45).

[7] Ryan Rivera Gregory Rakness. *Summary of Mu2e synchronization status - Mu2e-doc-db-24019 v1*. Mu2e electrical integration meeting, Fermilab, United States Department of Energy. 2019 (cit. on p. 106).

[8] INFN, ed. *The Mu2e Calorimeter Final Technical Design Report*. 2014 (cit. on pp. 19, 67).

[9] Inc. Integrated Silicon Solution. *IS46TR16128A Datasheet*. 2015 (cit. on p. 60).

[10] Microsemi. *PO0137 - Product Overview PolarFire FPGA*. 2019 (cit. on p. 67).

[11] Microsemi. *Polarfire FPGAs*. 2020. URL: `https://www.microsemi.com/product-directory/fpgas/3854-polarfire-fpgas#documentation` (cit. on p. 32).

[12] Microsemi. *Synopsys - Identify Microsemi Edition, Debugger User Guide*. 2018 (cit. on p. 120).

[13] Microsemi. *Synopsys Synplify Pro for Microsemi Edition - User Guide*. 2016 (cit. on pp. 53, 68).

[14] Microsemi. *TU0778 - PolarFire FPGA: Building a Cortex-M1 Processor Subsystem*. 2015 (cit. on p. 65).

[15] Microsemi. *TU0780 - Using Identfy ME with Libero SoC*. 2019 (cit. on p. 119).

[16] Microsemi. *UG0676 User Guide PolarFire FPGA Memory Controller*. 2019 (cit. on p. 62).

[17] Microsemi. *UG0677 User Guide PolarFire FPGA Transceiver 5.0*. 2019 (cit. on p. 77).

[18] Microsemi. *UG0773 - User Guide: PolarFire FPGA SmartDebug Software v12.0*. 2019 (cit. on p. 120).

[19] Future Electronics - Microsemi. *Avalanche Development Board User's Guide*. 2019 (cit. on pp. 73, 75).

[20] Future Electronics - Microsemi. *UG0679 - User Guide: Timing Constraints Editor Libero SoC v12.0*. 2019 (cit. on p. 116).

[21] Future Electronics - Microsemi. *UG0758 - User Guide: PolarFire FPGA Design Flow Libero SoC v12.1*. 2018 (cit. on pp. 53, 115).

[22] Jose Berlioz Rivera Monica Tecchio. *DRAC DDR3 Readout - Mu2e-doc-db-26902*. Fermilab, United States Department of Energy. 2019 (cit. on p. 60).

[23] Vadim Rusu Monica Tecchio. *The Tracker Readout Controller Specifications*. Fermilab, United States Department of Energy. 2018 (cit. on pp. 53, 54).

[24] Ryan Rivera Richard Kwarciany. *Mu2e Command Fan-Out Module Hardware User Guide, Mu2e-doc-5619*. Fermilab, United States Department of Energy. 2018 (cit. on p. 36).

[25] Ryan Rivera Richard Kwarciany. *Mu2e Data Transfer Controller Module Hardware User Guide - Mu2e-doc-4097*. Fermilab, United States Department of Energy. 2018 (cit. on p. 32).

[26] Ryan Rivera Richard Kwarciany. *TDAQ Timing Distribution Specification - Mu2e-doc-7048*. Fermilab, United States Department of Energy. 2018 (cit. on p. 41).

[27] Ryan A. Rivera. *Mu2e Detector Control System (DCS) Design and Specification*. Fermilab, United States Department of Energy. 2016 (cit. on p. 39).

[28] Ryan A. Rivera. *Mu2e Online DAQ Software Status*. Mu2e Collaboration Meeting, Fermilab, United States Department of Energy. 2019 (cit. on p. 46).

[29] Vadim Rusu. *Tracker Electronics*. Fermilab, United States Department of Energy. 2018 (cit. on p. 16).

[30] Samtec. *Firefly Modules*. 2020. URL: `https://www.samtec.com/optics/optical-cable/mid-board/firefly#` (cit. on p. 71).

[31] Monica Tecchio. *Requirements for the mu2e Tracker Front End Electronics*. Michigan University. 2014 (cit. on p. 15).

[32] Ryan Rivera Tomonari Scott Miyashita. *Mu2e Readout Controller Packet Protocol - Mu2e-doc-4914*. Fermilab, United States Department of Energy. 2019 (cit. on p. 84).

[33] J. Troska. *Versatile Link Technical Specification - EDMS Document No. 1140665*. CERN, Geneva, Switzerland. 2012 (cit. on pp. 35, 73).

[34] Wikipedia. *8b/10b*. 2020. URL: `https://en.wikipedia.org/wiki/8b/10b_encoding` (cit. on p. 78).

[35] Wikipedia. *Clock Recovery*. 2020. URL: `https://en.wikipedia.org/wiki/Clock_recovery` (cit. on p. 82).

[36] Xilinx. *Integrated Logic Analyzer v6.2 - Vivado Design Suite: LogiCORE IP Product Guide PG172*. 2016 (cit. on p. 121).

[37] Xilinx. *UG910 (v2018.1) - Vivado Design Suite: User Guide*. 2018 (cit. on pp. 73, 121).