Cape Peninsula
University of Technology

DESIGN AND DEVELOPMENT OF THE ALICE CRU USER LOGIC FIRMWARE FOR THE MID READOUT CHAIN

by

DIEUVEIL ORCEL THYS-DINGOU

A thesis submitted in fulfilment of the requirements for the degree

Master of Engineering: **Electrical Engineering**

in the Faculty of **Engineering Built and Environment**

at the Cape Peninsula University of Technology

Supervisor: **Prof. A. RAJI**
Co-supervisor: **Prof. E.Z. BUTHELEZI and Dr. S.V. FÖRTSCH**

Bellville
Date submitted (March 2022)

DECLARATION

I, **DIEUVEIL ORCEL THYS-DINGOU**, declare that the contents of this thesis represent my unaided work and that the thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

Signed  **Date** 12/10/20222

ABSTRACT

A Large Ion Collider Experiment (ALICE) at the Large Hadron Collider (LHC) at CERN is going through a major upgrade during which some of its subdetectors are replaced with new ones, while others are equipped with new electronics to handle the expected higher collision rates in the current running period (Run 3), which will start in 2022. As part of the upgrade, certain subdetectors such as the Muon Trigger (MTR), renamed to Muon Identifier (MID), now operate in a continuous, triggerless readout mode, in addition to the previous triggered readout mode. Due to the increased quantity of data, typical methodologies are impossible to employ without massive efforts to expand the processing capacity. Since the new ALICE computing system cannot keep up with the increased data flow of the MID, a new processing algorithm has to be established.

The MID employs a Common Readout Unit (CRU) to interact with all subsystems of its new readout chain. The CRU, based on the PCIe40 hardware and the ARRIA 10 FPGA, is designed to meet the ALICE requirements. Its common firmware framework enables data taking in both continuous and triggered modes from most ALICE subdetectors and can be customized to meet the needs of any subdetectors through the use of a user logic component placed at the heart of the CRU firmware. This research project provides a new approach to processing the MID readout data based on the user logic component. Innovative methods for reducing the high bandwidth data rate and adaptations to ease data handling in the computing system have been introduced. In order to test and evaluate the user logic, a laboratory test bench equipped with a small-scaled MID readout chain has been developed at NRF iThemba LABS. Finally, the research findings and deliverables of this research can be used as a preliminary solution for a full-scaled user logic component, as well as by other postgraduate students for their studies.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my thesis external supervisors Prof. Zinhle Buthelezi and Dr. Siegfried FÖrtsch of the Department of Subatomic Physics at iThemba LABS. The doors to Prof. Buthelezi and Dr. FÖrtcsh offices were always open whenever I ran into trouble or had queries regarding my research or writing. They consistently guided me through hardships, allowing this research to be my work but steering me in the right direction whenever they believe I needed it. They made this project possible, and I am grateful for their insightful remarks on this thesis.

I would also like to thank Prof. Atanda Raji, my thesis internal supervisor. This thesis would not have been completed without his academic guidance and dedicated commitment to every step of the process. I would like to express my gratitude for your patience and understanding over the past two years.

In February 2020, I spent several weeks at CERN (Geneva, Switzerland) studying with Dr. Christophe Renard, Dr. Diego Stocco, and Dr. Fillipo Costa. My time at CERN has been extremely productive, and working with these specialists has been an incredible experience. Much of the analysis presented in chapters 2 and 3 is owed to my time at CERN. Dr. Renard graciously assisted me in strengthening my skills in FPGA development and he was quite tolerant of my knowledge gaps in the field. I would also want to thank Dr. Stocco and Dr. Costa for their assistance in validating the user logic component for this research project. The validation of the user logic could not have been completed without their enthusiastic participation and feedback.

I would also like to acknowledge Mr. Rene Monteverdi, Mr. Rony Kuriakose, and Mr. Nathan Boyles for their contributions to the development of the test bench at iThemba LABS.

I would not have been able to complete this study on my own without the contributions and support of many people, including Ms. Anu Joseph, family, and friends. Recognizing your contribution, whether direct or indirect, is an important element of my endeavour. Thank you very much!

The financial assistance of the SA-CERN program and iThemba LABS towards this research is acknowledged. Opinions expressed in this thesis and the conclusions arrived at, are those of the author, and are not necessarily to be attributed to the SA-CERN program or iThemba LABS.

DEDICATION

This thesis is dedicated to my mother, Doris, who has been a consistent source of inspiration and support throughout the hardships of graduate school and life. I am very grateful to have you in my life. This work is also dedicated to my grandparents, Gilbert and Angele Castanou, who have always loved me unconditionally and whose good example has inspired me to work hard for what I desire.

TABLE OF CONTENTS

Table of contents

CHAPTER 6

LIST OF FIGURES

LIST OF TABLES

ABBREVATIONS AND ACRONYMS

ADC	**-**	Analog-Digital Converter
ADULT	**-**	A DUaL Threshold
ALICE	**-**	A Large Ion Collider Experiment
BC	**-**	Bunch Crossing
CTP	**-**	Central Trigger Processor
CRU	**-**	Common Readout Unit
DCS	**-**	Detector Control System
FEE	**-**	Front-End Electronics
FEERIC	**-**	Front-End Electronics Integrated Circuit
FLP	**-**	First Level Processor
FPGA	**-**	Field Programmable Gate Array
GBT	**-**	GigaBit Transceiver
LABS	**-**	Laboratory for Accelerator-Based Science
LHC	**-**	Large Hadron Collider
LS2	**-**	Long Shutdown 2
LTU	**-**	Local Trigger Unit
MID	**-**	Muon Identifier
MTR	**-**	Muon Trigger
NRF	**-**	National Research Foundation
O²	**-**	Online-Offline
QGP	**-**	Quark-Gluon Plasma
RPC	**-**	Resistive Plate Chamber
RTL	-	Register Transfer Logic
TTC	**-**	Timing and Trigger Control
TTS	**-**	Timing and Trigger System
VHDL	**-**	VHSIC Hardware Description Language

NOMENCLATURES

Cavern – is a large subterranean space where the detector is situated.

Collision – particles smashing together in the detector.

Lead – is an elemental heavy metal particle present in the nature and industrialized product

LHC Run – is a running period of which the LHC goes online, it usually lasts about 4 years.

Muon – is a particle that is identical to an electron but heavier.

Plasma – is a matter heated to high temperature, it is so hot that electrons are torn from atoms, resulting in an ionized gas.

Proton – is a subatomic particle with a positive charge equivalent to an electron.

Subdetector – is a component of the detector (e.g., the MID subdetector is a component of the ALICE detector).

Submodule – is a module that is part of a bigger module.

CHAPTER 1

Introduction

Since the early days of its first employment in the 1960s, the distinctive qualities of Field Programmable Gate Array (FPGA) such as integration, flexibility, low power, and high bandwidth communication have allowed various new and critical approaches (Intel, n.d.). FPGAs are the result of multiple generations of sophisticated technology, and they are often recognized as one of the major components utilized in the data acquisition of detectors in high-energy physics experiments (Musa, 2008). However, the need for customized instruments entails ongoing research and development of new tools, electronics and instrumentation methods. This work provides a new customized FPGA firmware for one of the four collision points of the largest science experiment in the world, where technology is rapidly and continuously evolving.

The ALICE detector (ALICE Collaboration, 2008) at the Large Hadron Collider (LHC) (Evans L., 2008) at the European Organization for Nuclear Research (CERN) is undergoing a major upgrade during which some of its subdetectors are replaced with new ones, while others are equipped with new electronics to cope with higher collision rates planned for the following years. Like most of the subdetectors in ALICE, the MID is taking full advantage of today's FPGAs by trying to improve the way data are processed in its readout chain. This research aims to identify the best approach to process data from the MID subdetector using a customized user logic firmware. This user logic firmware is written in the VHDL programming language and can implement multiple features tailored to the specifics of the MID.

This chapter begins with background information on the experiment and then introduces the ALICE detector. The goals of the project are established, and an overview of the strategy used to achieve these goals is provided. The hypothesis, as well as the constraints and key contributions, are listed.

1.1. Background

1.1.1. ALICE experiment

CERN (Brüning, et al., 2004) is the world’s leading laboratory for nuclear and particle physics research located on the border of Switzerland and France. CERN houses the LHC, which is about 100 meters below the surface and 27 kilometres in circumference (CERN, 2008).

The LHC produces particle beams, i.e., proton-proton (p-p), proton-lead (p-Pb), and lead-lead (Pb-Pb) at ultra-relativistic energies to create and study the characteristics of a highly dense form of matter reminiscent of the early Universe a microsecond after the Big Bang (Giubellino, 2015). Spread along the LHC ring are four individual experiments positioned around the four collision points where the beams collide. As shown in Figure 1.1, one of these experiments is ALICE.

For a few millionths of a second after the Big Bang, the universe consisted of a hot plasma of deconfined elementary particles called quarks and gluons. A few microseconds later, this hot plasma known as the quark-gluon plasma (QGP) cooled further down to form hadrons, amongst others protons and neutrons, the fundamental building blocks of atomic matter. The conditions of the QGP can be created in high-energy heavy-ion collisions at the CERN LHC. ALICE is the detector at the CERN LHC dedicated to studying this strongly interacting matter, the QGP, and its properties by recording data in Pb-Pb collisions, which also requires information from pp and p-Pb collisions for a complete study.

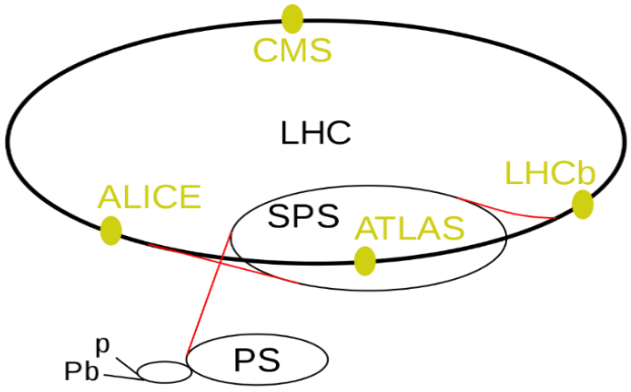

Figure 1.1: LHC ring with its four main experiments, ALICE, ATLAS, LHCb, and CMS as well as its super proton synchrotron (SPS), proton synchrotron (PS), for the generation of p and Pb beams (Horvath, 2006)

1.1.2. ALICE detector

To reconstruct and identify a myriad of particles created in these collisions, the ALICE detector illustrated in Figure 1.2, is using a set of 19 subdetectors extended over a length of 26 m and 16 m in height and width, weighing over 10 000 tons. The subdetectors encapsulated in a toroid magnet (L3) provide information about the mass, velocity, and

electric charge of the particles by measuring their tracks. Each subdetector is designed to study different aspects of the particles created in the collisions.

The ALICE detector consists of two main regions: the central barrel region and the forward region known as the Muon Spectrometer.

The central barrel detectors are surrounded by a solenoid L3 magnet providing a field of 0.5 T. At the center of the central barrel and closest to the beamline is the Inner Tracking System (ITS) composed of six layers of silicon detectors: Silicon Pixel Detector (SPD), Silicon Drift Detector (SDD), and Silicon Strip Detector (SSD). The ITS is encompassed by a cylindrical Time Projection Chamber (TPC), three-particle identification arrays of Time of Flight (TOF), a ring imaging of Cherenkov High Momentum Particle Identification Detector (HMPID), and a Transition Radiation Detector (TRD). The outer surface layer contains the Electromagnetic Calorimeters (EMCal), and the Photon Spectrometer (PHOS). Small-scale subdetectors used for global event identification and triggering such as the Zero Degree Calorimeter (ZDC), Photon Multiplicity Detector (PMD), Forward Multiplicity Detector (FMD), T0, and V0 are located on either side of the interaction point. On the three upper outside faces of the solenoid L3 magnet is A Cosmic Ray Detector (ACORDE). It consists of an array of plastic scintillator counters and provides accurate information about cosmic ray events.

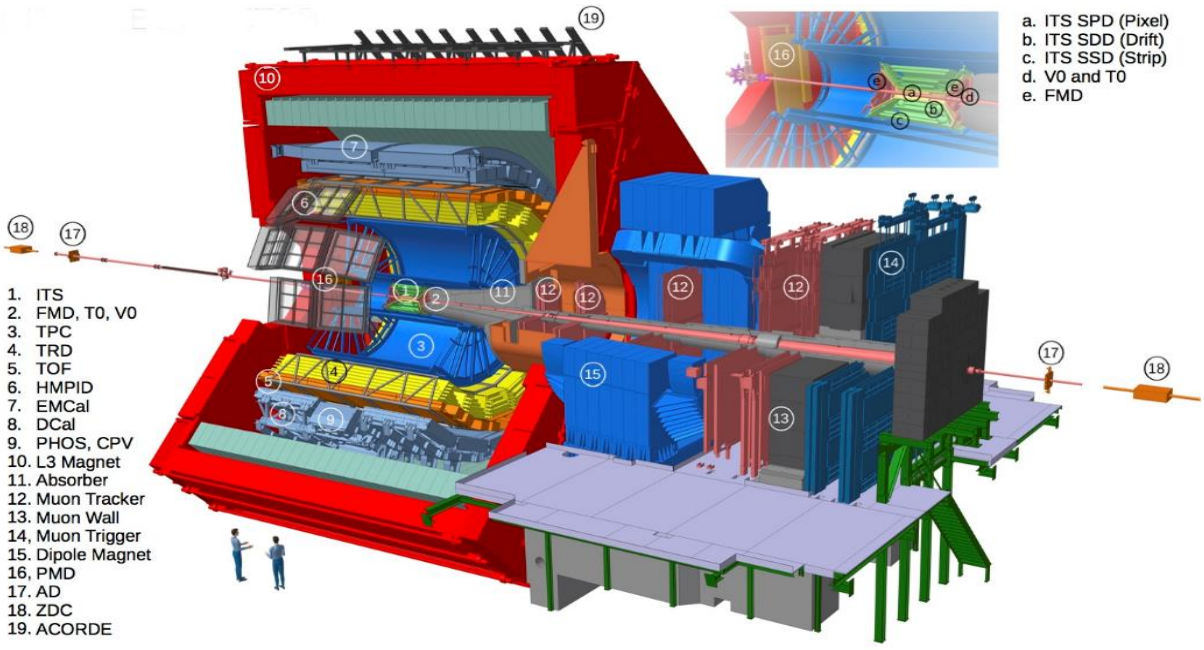

Figure 1.2: Schematic overview of the ALICE detector with its subdetectors. This picture was taken during the LHC Run 2 before the long shutdown upgrade 2 (Elena, 2017)

The Muon Spectrometer is designed to measure muon production from the decays of quarkonia, low mass vector mesons, heavy-flavor hadrons, and electroweak bosons (Consesa De Valle, 2007). The Muon Spectrometer has an angular acceptance of 171˚-178˚, corresponding to the pseudorapidity region $-4.0 < \eta < -2.5$. The Muon Spectrometer covers a total length of $\simeq$ 17 m and it is composed of the following components: front-absorbers to filter all particles except muons coming from the interaction point, a large dipole magnet, high-resolution Muon Tracking Chambers (MCH), a 120 cm thick iron wall (Muon Filter), and a Muon Trigger (MTR).

1.1.3. Muon Trigger

The MTR system is equipped with a configurable threshold to provide trigger signals for selecting events of interest and discarding events with only low momentum muons (p<4 GeV/c). As illustrated in Figure 1.3, the muon trigger is based on 72 single-gap Resistive Plate Chamber (RPC) detectors, arranged in 2 stations of 2 chambers, each at a distance of about 16.1 m and 17.1 m from the interaction point, respectively. Each RPC consists of two planes: a positively charged anode and a negatively charged cathode, both made of very high resistivity plate plastic material and separated by 2 mm of a gas mixture of Ar, CH2F4, C4H10, and SF6. Once a charged particle such as a muon passes through the chamber, it knocks electrons out of the gas atoms. These electrons in turn hit other atoms, causing an avalanche of electrons. Since the electrodes are transparent to the electrons, they are instead picked up by external metallic strips after a small but precise time delay. The combination of hit strips firing gives a prompt measure of the muon momentum, which is read-out by the front-end electronics, known as A DUaL Threshold (ADULT) card (Dupieux, 2006). The signals from the ADULT cards are then propagated to the readout electronics based on three programmable circuits (local, regional and Global) working in sequential mode at 40 MHz, to make immediate decisions about the validity of the data. The ADULT electronics were initially developed for streamer mode operation with a gas mixture for the LHC Run 1 (2010-2012). A few years later, a maxi-avalanche operation mode was introduced for the LHC Run 2 (2015 - 2018), where the signal amplitude was smaller than in the streamer mode, but still compatible with the minimum threshold of 7 mV set in the ADULT cards. The subdetector planes are mounted on a mechanical frame on rail support that can be moved to allow access to the chambers for maintenance purposes.

1.1 Background

The MTR will be called MID after the upgrade for Run 3 (2022 – onwards). Technical details concerning the new MID readout chain are described in chapter 2, as the work described in this thesis is focused on this specific subdetector.

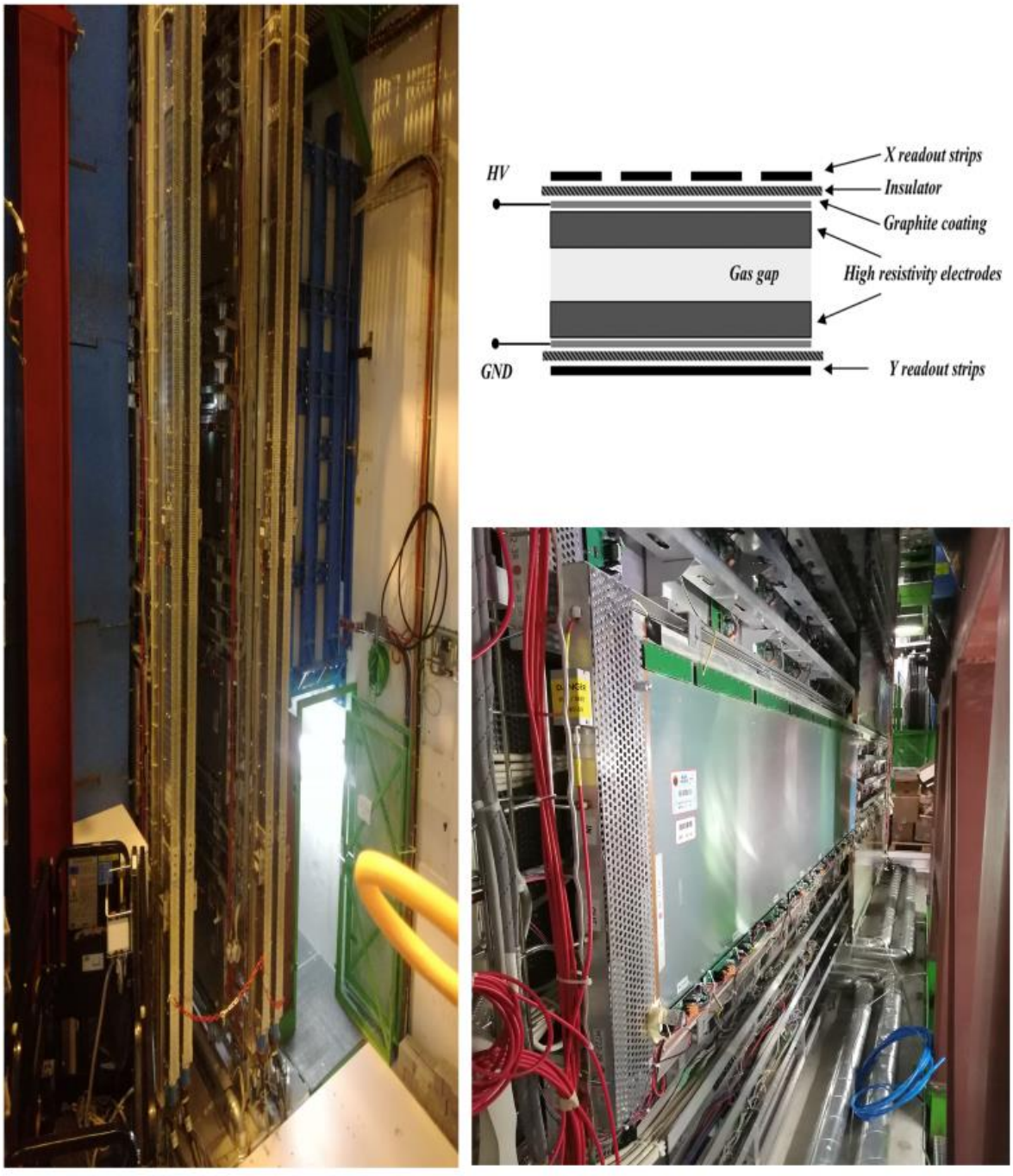

Figure 1.3: Left: View of the two trigger stations positioned behind the muon filter. Right-top: schematic view of the cross-section of the RPC. Right-bottom: an independent RPC module equipped with front-end electronics

(Adapted from Sauli, 2014)

1.2. LHC Run 3

Based on data collected in Runs 1 and 2 (~10 petabytes of raw data), ALICE is the leading heavy-ion experiment in the world and is quickly expanding the knowledge gathered in previous experiments all over the world. The LHC is currently going through three-years planned second Long Shutdown called (LS2), which started at the end of 2018 to prepare for Run 3. In line with the LHC upgrade, the ALICE detector is undergoing a major upgrade. This upgrade addresses the challenge of reading out lead-lead collisions at a rate of 50 kHz and proton-proton at 1 MHz and higher. At the center of the ALICE upgrade strategy, is a high-speed readout approach based on a Common Readout Unit (CRU). The CRU has been developed for detector data readout, concentration, reconstruction, multiplexing, and data decoding on the Online-Offline (O^2) computing system.

Many of the proposed physics observables require a change in the data-taking strategy, moving away from triggering a small subset of events to continuous online processing and recording of all events. To achieve these goals, the ALICE detector is being upgraded in such a way that all interactions will be scrutinized with precision. The upgrade entails the replacement of some subdetectors with new ones, making use of new technologies, while most others are being equipped with new front-end and readout electronic systems. The LHC Run 3 was planned to start in the middle of 2021 onwards but has been postponed to March 2022 due to the global pandemic (Schaeffer, 2020).

1.2.1. The upgrade: Muon Trigger to Muon Identifier

For the past 10 years since the beginning of LHC Run 1, the selection of single muon and di-muon events with a maximum trigger rate of 1 kHz was provided by the MTR, as well as muon identification. However, to cope with the increased luminosity of the LHC during Run 3, this current trigger strategy is no longer sufficient. The upgrade trigger strategy described in the letter of intent (ALICE Collaboration, 2012) does not require a muon trigger since all events of interest will be read out upon the interaction trigger before online selections. For this reason, as part of the upgrade, the MID will only play the role of muon identifier.

1.3. Problem statement

Several issues concerning the readout arose during the transition from MTR to MID. These issues (Stocco, 2020) were observed throughout a preliminary series of tests conducted on the MID readout chain at Subatech, Nantes, France, where the readout electronics were developed. The upgraded system showed limitations when running without data pre-analysis performed at the CRU firmware level. Among these limitations are large data rates, desynchronization of data, lack of hardware resources, and, many other minor issues related to the data format transmitted to the O^2 computing system. All these limitations had to be addressed urgently.

1.3.1. Large data rate

In the triggerless readout chain, all events are read out continuously. This results in a bandwidth of 3.2 Gbps generated by each data link in the chain. This large amount of data is a problem for the O^2 computing facility to conduct data processing concurrently without data compression at the CRU firmware level. The readout electronics data links are based on an 80-bit frame transmitted continuously at 40 MHz (25 ns), which corresponds to the LHC bunch crossing interval. The Bunch Crossing (BC) interval is the period between bunches of particles crossing each other in the LHC. In other words, it is the amount of time between collisions. On the other hand, as previously mentioned, one of the primary goals of the ALICE detector upgrade is to read out lead-lead collisions at 50 kHz (20 µs), and proton-proton collisions at 1 MHz (1 ns). This indicates that data worth analyzing are generated every 800 bunch crossings (20 µs / 25 ns = 800 BCs) during lead-lead collisions and every 40 bunch crossings (1 µs / 25 ns = 40 BCs) during proton-proton collisions. Data collected outside this frame are meaningless and must be suppressed. Retaining these data in the memory results in inefficiency and a waste of memory.

1.3.2. Desynchronization of data

The data obtained from all readout electronics occur simultaneously, at fixed periods, and are transmitted to the CRU over a wide set of optical links. However, differing transmission delays result in the data from the various links losing synchronization when transmitted to the O^2 computing system. Therefore, they cause more problems further along the chain at the synchronous and reconstruction levels.

1.3.3. Lack of hardware resources

The O^2 computing system is capable of handling the data rate from a single regional crate at the expense of two central processor units. Attempting to use a single processor to decode a single regional crate leads to irreversible data loss. Hence, it takes two processors to decode a single regional crate. However, the computer (Costa, 2019) used in the readout chain contains 20 processors. To decode data from the entire readout chain, the computer would need 32 processors, not to mention any additional processors required to perform further processing of the decoded data. As a result, processing data from the complete system is unfeasible using a single computer. Details on the architecture of the MID readout chain are described in the next chapter.

1.4. Research aim

The ALICE collaborators participating in the MID project are searching for new ways to process raw data. Many alternatives have been suggested, but most of them entail significant improvements in the existing readout chain. Some of the improvements require changing the algorithm implemented in the O^2 processing to cope with the large data rate, acquiring additional CRUs, and core processor computers to process data from the entire readout electronics. A thorough analysis review revealed that the most efficient and cost-effective solution is to take advantage of the existing high-speed FPGA incorporated in the CRU by designing a customized user logic firmware to meet the requirements of the readout chain.

The user logic is a specific subdetector component, that can be implemented in the CRU firmware through a specific compilation. It is developed by the subdetector teams and can perform low-level data processing and other additional features before forwarding data to the O^2 computing facility for further analysis. This research aims to improve the way data are processed in the MID readout chain using a customized user logic firmware before the start of the LHC Run 3.

1.5. Objectives

The research aim stated above is achieved through the following objectives:

- Review and analyze different components of the readout chain
- Select the best user logic algorithm to meet the MID requirements
- Monitor errors identified during the data acquisition
- Successfully validate the user logic simulation tests
- Successfully validate the user logic hardware tests
- Make recommendations for future improvements

1.6. Hypothesis

This study offers the possibility of designing and developing a stable and reliable user logic firmware that can improve the way data are processed in the MID readout chain. This can be achieved by developing an algorithm based on systems requirements. However, the difference in protocol between various systems of the readout chain makes it complex and can be time-consuming.

The main questions to be considered are whether a user logic prototype can be designed and tested to meet the requirements of the MID readout chain on time before the start of the commissioning phase of the MID-subdetector, and whether or not this prototype can be used to develop a realistic user logic capable of processing data from the entire readout chain, considering hardware and software restrictions of the approved FPGA.

1.7. Delineation

This thesis is limited to the design and development of the user logic firmware prototype capable of pre-analyzing data from 2 regional links of the MID readout chain. This research analyses in detail different systems used in the readout chain and improves the way data are processed in the CRU. The proposed scheme is developed after intensive research and a good understanding of the ALICE detector. Hence, the incorporation of the user logic component into the existing CRU firmware is done through conformance with established requirements and practice. Additionally, important technical decisions such as hardware, communication protocols, design tools, programming languages, and most relevant resource usage limit of the research in question have been established before the beginning of the research.

1.8. Collaboration and main contributions

In South Africa, the National Research Foundation (NRF) iThemba Laboratory for Accelerators Based-Science (LABS) is part of the ALICE Collaboration and contributes to the ALICE Muon Spectrometer upgrade, in particular the MID. In collaboration with the Cape Peninsula University of Technology (CPUT) and the University of Cape Town (UCT), NRF iThemba LABS is responsible for conducting research and developing the CRU user logic firmware for the MID readout chain, including setting up an in-house testbench data acquisition readout chain and the maintenance thereof.

The user logic project started in early 2018, of which the early research and findings are described in (Boyles et al., 2021). Due to the rapid evolution of the ALICE CRU software and firmware projects, a complete modification of the initial project was of paramount importance. In 2020, a new design and development of the user logic, based on realistic data acquisition requirements and availability of relevant readout components, led to the present study. Together with Dr. C.Renard (expert in the readout electronics at Subatech in Nantes, France), the requirements to process data from 2 data links of the readout chain were established. To keep track of the rapid evolution of the CRU software and firmware, regular and rigorous consultations and discussions took place with Dr. F.Costa (ALICE CRU software expert at CERN) and Dr. O.Bourrion (ALICE CRU firmware developer at the University of Grenoble, France). For what concerns the MID O^2 requirements, Dr. D.Stocco (MID O^2 expert at Subatech, France) was the main contact and source of information. His input was required since the outcome of this research is linked to the way data will be handled at the next phase of the data acquisition chain. As such, he was instrumental in setting up some additional requirements and constraints to facilitate the readability of the user logic output data.

Overall, the design and development of the ALICE CRU user logic firmware prototype for the MID readout chain are provided by the Electrical Engineering department at CPUT with support from various collaborators using facilities provided by the NRF iThemba LABS and advanced technology.

1.9. Methodology

The research methods that are utilized for the development of this thesis are:

- Literature review: since in many cases the written literature is not available on the readout chain, the information was gathered by reading technical specification papers, IEEE published journals, conferences, interviewing specialist engineers in the data acquisition chain, and through the World Wide Web.

- Prototyping: Intel Quartus Prime Pro 18.1 (Intel, 2019) is the main software environment recommended and used to design and develop the user logic. For this thesis, two different prototyping approaches are implemented. The rapid throwaway method involves exploring ideas by quickly developing a prototype based on preliminary requirements which are then revised through simulation test feedback. Once validated, the evolutionary approach is then introduced. This method uses a continuous, working prototype that is refined after each iteration of hardware test feedback.

- Simulation tests: ModelSim Intel FPGA (Intel, 2020) is the simulation software used to verify the functionality of the user logic algorithm by analyzing each component of the model. A more advanced simulation is performed by merging the CRU firmware simulation files as well as the MID readout electronics firmware simulation files into a single testbench for more efficient and accurate results.

- Hardware tests: a readout testbench facility available at iThemba LABS is developed for practical work. Expected tests for conformance include testing of the user logic prototype using a fully-functional MID readout testbench set-up capable of emulating the same events generated by the main ALICE MID readout chain at CERN.

- Data collection: simulation and hardware tests are conducted to collect real data coming in and out of the user logic firmware. A comparison between the input and output data is done to achieve an effective assessment of the user logic algorithm.

1.10. Thesis outline

This thesis is organized into six chapters, which are outlined as follows:

Chapter 1, introduces and frames this study by reviewing the background of the ALICE detector. It goes into further depth on the LHC Run 3 upgrades, which resulted in the transition from MTR to MID. The issues observed are then encapsulated into a conventional problem statement. The research objectives are derived from the problem statement and the delineation, as well as the research contribution, is established. The methodology and approach used to investigate and consequently execute the aims of this study are defined.

Chapter 2, describes the newly enhanced MID readout chain. It highlights the role of each component and describes the technologies implemented to manage the increased data rates arising due to the LHC upgrade.

Chapter 3, concentrates on the architectural design of the CRU firmware, and its features and functionalities. A detailed discussion of the location of the MID user logic component in the CRU firmware, the multiple interfaces surrounding it, the choice of design specifications, their benefits and drawbacks, as well as the available FPGA resources are also presented.

Chapter 4, presents the design and implementation of the user logic component. It provides a detailed description of the functioning of each submodule and elaborates on how they are implemented in the user logic component.

Chapter 5, covers all tests performed on the user logic component. These tests are critical for evaluating the performance of the user logic. The rigorous evaluations are carried out throughout both the functional and hardware verifications. The key findings are presented in the form of tables and also include a thorough discussion of the outcome.

Chapter 6, concludes this thesis and makes recommendations for future study as well as the extension of the project. The academic and industrial benefits of the test bench at iThemba LABS are also explored.

CHAPTER 2

MID readout chain

As discussed in the previous chapter, the approach taken by ALICE is to read out all lead-lead events at an interaction rate of 50 kHz. The objective behind the upgrades is to significantly improve vertexing and tracking capabilities at low transverse momentum. In line with the ALICE upgrades, the MID readout chain is also being upgraded to support continuous readout operation after the LS2. This upgrade entails:

- New RPCs;
- New front-end electronics;
- New readout electronics

This chapter deals with the description of the MID readout chain and is organized as follows. Section 2.1 gives a brief overview of the readout chain. The upgrade of the RPCs is described in section 2.2, while the front-end and readout electronics upgrades are discussed in sections 2.3 and 2.4. The CRU is the heart of the readout chain, and its hardware architecture is discussed in section 2.5. The trigger architecture, online-offline computing system, and detector control systems are discussed accordingly in the following sections.

2.1. Overview

The readout chain block diagram designed for this study is shown in Figure 2.1. It consists of 21,000 strips connected to 72 RPC detectors spread over multiple Front-End Electronic Rapid Integrated Circuit (FEERIC) cards equipped with one or two customized Application-Specific Integrated Circuits (ASICs) (Manen et al., 2013). The strip pattern signals from the FEERICs are propagated to the readout electronics using high-speed Low-voltage Differential Signalling (LVDS) channels. The readout electronics act as the readout interface and are in charge of the first stage of the trigger decision. They are mounted on the upper gangways inside the cavern a little further away from the detector stations, where the radiation is low. Since the colliding beams will produce a lot of radiation in the area around the ALICE detector in the cavern, the readout electronics regional cards are equipped with Gigabit Transceiver (GBT) radiation hardening to operate under these conditions. The CRUs are the key components of the chain, they combine and multiplex data from multiple readout electronic cards as well as timing and trigger information generated from the Central Trigger Processor (CTP) via the Local Trigger Unit (LTU) before transmitting the data to the O^2 computing facility for processing and storage. The CRUs are mounted in computers housed in the intermediary computer room, called the counting room, away from the ALICE cavern

and thus, do not require radiation hardening, as is the case for the readout electronics. These computers can be reached over the network from the main Detector Control System (DCS). The DCS manages the readout chain by sending commands and monitoring the system. Experimental data are moved from the First Level Processor (FLP) to the Event Processing Node (EPN) for processing and storage. The EPN is an internal component of the O² computing system.

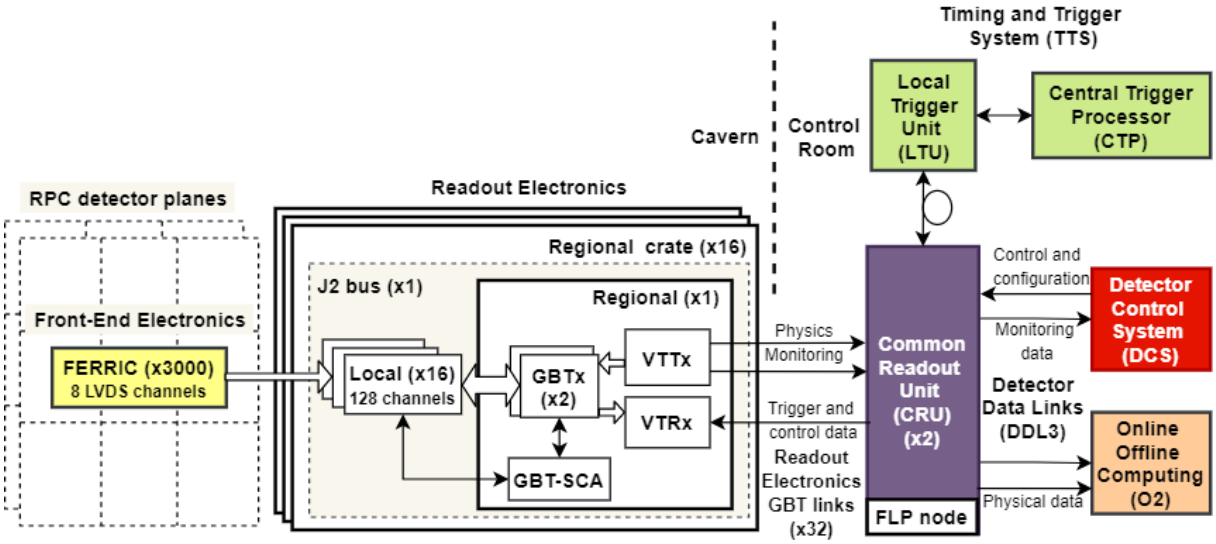

Figure 2.1: A schematic description of the MID readout chain architecture for Run 3

2.2. RPC detectors

In the ALICE cavern, three distinct forms of RPC are installed. They refer to long, short, and cut forms, respectively as illustrated in Figure 2.2. The beam pipe is accommodated by the short and cut forms.

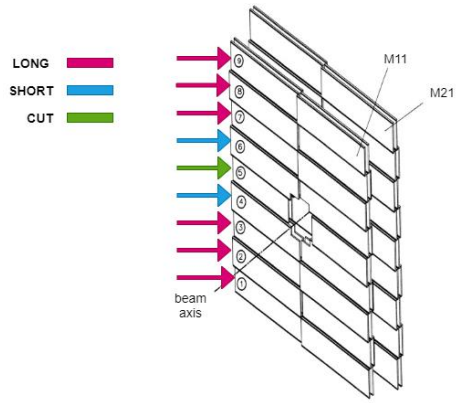

Figure 2.2: Three forms of RPCs found in the ALICE cavern

(Adapted from Blanc & Dupieux, 2008)

2.1 Overview

As briefly discussed in subsection 1.1.3, the current RPCs are composed of metallic strips made of copper and have three different pitch options: 1, 2, and 4 cm. The RPCs have one collection of strips on each side. The strips on either side of the RPCs are orthogonal to one another. In comparison to the dipole motion on charged-particle tracks, the vertical strips that have (y) hits are referred to as Non-Bending Planes (NBPs) and the horizontal strips that have (x) hits are referred to as Bending Planes (BPs) as shown in Figure 2.3.

The amount of RPC hits in Pb-Pb collisions is expected to exceed the highest counting rate of about 10 Hz/cm² up to 90 Hz/cm² (Ferreti, 2019). This is marginally similar to the maximum rated capacity of the detector during the LHC Run 2. This rise would also hasten the aging of the gas gaps, which will hit the end of their projected lifespan long before the end of Run 3, necessitating the replacement of certain gas gaps and other affected components.

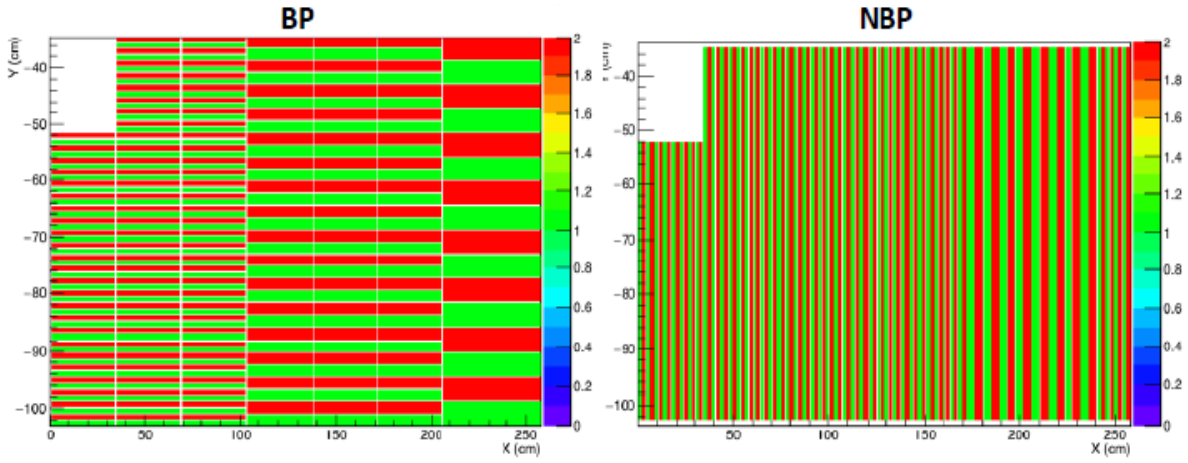

Figure 2.3: RPC non-bending and bending strip patterns.

These upgrades are distributed among three institutions. The Puricelli factory in Costa Masnaga (Italy) is responsible for redesigning the bakelite resistive electrodes, which feature a smoother surface for the bakelite used on the presently installed RPCs, the General Tecnica in Colli (Italy) is responsible for manufacturing the gas gaps for the new RPCs and, the National Institute for Nuclear Physics (INFN) in Torino (Italy) is responsible for checking and testing the performance of the new RPCs with cosmic rays. The installation of the new RPCs in the cavern started from July 2021, with the intent of installing 2 RPCs per day. In case of failure to meet this deadline, the MID will operate with the existing RPCs during Run 3 until the new RPCs are ready.

2.3. Front-End electronics

The RPC ADULT electronics have been replaced by the new FEERIC and unlike the ADULT, it performs amplification of analog signals from the RPCs. The FEERIC is an 8-channel ASIC, which uses low-cost AMS 0:35mm CMOS technology developed by the Laboratory of Physics Clermont-Ferrand. It is made up of a trans-impedance amplifier stage, a zero-crossing discriminator to limit time walk effects, and a one-shot to prevent retriggering during 100 ns and LVDS drivers. Table 2.1 summarizes the main specifications, and requirements of the FEERIC ASIC. In contrast to the ADULT card thresholds, which were set using an analog voltage distribution of just one threshold value per RPC, the FEERIC card thresholds would be set wirelessly during the LHC Run 3. Their values will be determined by fine-tuning the threshold based on the RPC efficiency while minimizing the operating high voltage.

The technology selected to accomplish this task is the ZIGBEE protocol (Farahani, 2008). It is a wireless technology established as an open universal norm to meet the special requirements of ultra-low-cost, low-power wireless IoT networks based on the IEEE 802.15.4 physical radio and works in unlicensed bands such as 2.4 GHz. The ZEGBEE is incorporated on the Atmel SAMD21 microcontroller (Microchip inc, 2021) and the program is based on Arduino libraries (I2C, SD cards, and Xbee module). This is then mounted on a printed circuit board called the Xbee cards. The master cards are connected to the DCS computer using ethernet, and the ZIGBEE (wireless) protocol is used to communicate from master to nodes.

Table 2.1: Requirements of the FEERIC ASIC (Manen et al., 2013)

Feature	Value or type
pulse polarity	positive or negative
number of channels	8
power consumption per channel	< 100 mW
input impedance	< 50 ohms
dynamic range	20 fC < q < 3 pC
time resolution	< 1 ns
time walk	< 2 ns
one-shot	100 ns
output format	LVDS
signal shape	square pulse 23±3 ns

As previously mentioned, the charge delivered within the gas gaps must be lowered to minimize aging and improve rate capabilities. This is achieved by operating RPCs with the same gas mixture but at a lower gain, in conjunction with the FEERICs, which perform amplification of the analog signal before discrimination. Hitherto 2384 + 336 spares FEERIC cards and 26 Xbee cards have been manufactured and installed in the ALICE cavern. The installation and commissioning of all FEERIC and Xbee cards concluded in July 2019.

2.4. Readout electronics

To cope with the new readout rates, the local and regional readout cards were redesigned. Since the triggering functionalities are abandoned, a more streamlined approach was introduced. The hardware implementation of the regional and local card is almost identical, minimizing the design and development effort by re-using the same hardware and altering the FPGA firmware. The global crate was replaced by a new regional crate. As shown in Figure 2.4, the readout electronics are divided into 16 vertical regions (8 on the left and 8 on the right side of the plane). Each vertical region is read out by a single regional card located in the regional crate. Each crate contains a backplane bus card called the J2 card, which provides ports to a regional card and up to 16 local cards.

2.4.1. Local card

For every bunch crossing, the local card receives binary data from LVDS channels, which indicates whether the corresponding channel has been struck or not. The local card is equipped with 16 LVDS input connectors (32 pins, for both the bending and non-bending planes). It is embedded with the Intel MAX 10 FPGA (10M50DCF484C7G) **(**Intel, 2021), for which its firmware is described in (Renard, 2021).

2.4.2. J2 bus card

The J2 bus card serves as an interface between the regional crate and the local/regional cards in terms of power, and it also serves as an interface between the local and regional cards in terms of data transfer. The J2 bus card has a 4-bit dip switch for assigning a specific identification to the regional crate, as well as three LEDs for monitoring the voltages (2.5V, 3.3V, and 5V) supplied to the regional and local cards.

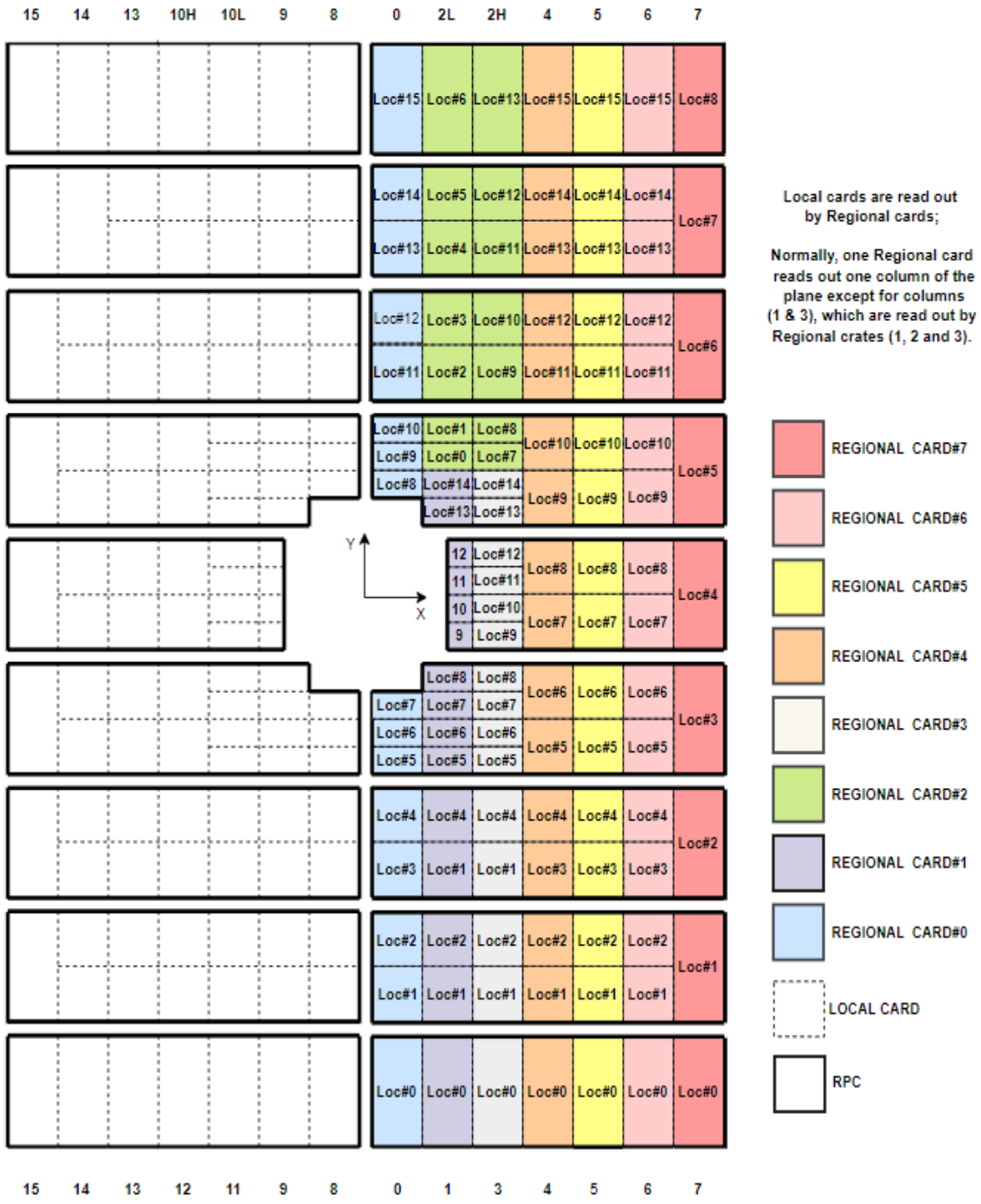

Figure 2.4: Geometry of the readout electronics. This diagram was created for this work. It illustrates the number of local cards distributed per column

2.4.3. Regional card

The regional card collects data from up to 16 local cards using the GBT protocol, which is discussed in the next section. Similar to the local card, the regional card is incorporated with the same Intel MAX 10 FPGA. However, unlike the local card, it is equipped with two bi-directional GBT optical links allowing transmission and reception of data to/from the CRU. The implementation of 2 GBT optical links per regional card enables complete regional crate data transfer. The firmware implemented in the regional card FPGA is a slightly modified version of the local card firmware, which is also described in (Renard, 2021).

2.4.4. Event data formats

Events are stored in the local and regional card multi-buffers for each trigger. The multi-event buffer in the local card carries strip patterns, therefore it is larger than the one found in the regional card. The event data formats of the local card and regional card are shown in Table 2.2 and Table 2.3, respectively.

Table 2.2: Local event format (Renard, 2021)

Coding of SOx, EOx, RESET, CALIBRATE Event in LOCAL	*Bits*	*Coding of PHY, ORBIT Event in LOCAL*	*Bits*	*Coding of self-triggered DATA Event in LOCAL*	*Bits*
START BIT (always '1') CARD TYPE (always '1'=LOCAL) LOCAL BUSY ('0'=OK; '1'=FIFO full) LOCAL DECISION (tracklet) ACTIVE ('0'=OFF; '1'=ON) REJECTING ('0'=OFF; '1'=ON) MASKED ('0'=OFF; '1'=ON) OVERWRITED ('0'=OFF; '1'=ON)	1 1 1 1 1 1 1 1	START BIT (always '1') CARD TYPE (always '1'=LOCAL) LOCAL BUSY ('0'=OK; '1'=FIFO full) LOCAL DECISION (tracklet) ACTIVE ('0'=OFF; '1'=ON) REJECTING ('0'=OFF; '1'=ON) MASKED ('0'=OFF; '1'=ON) OVERWRITED ('0'=OFF; '1'=ON)	1 1 1 1 1 1 1 1	START BIT (always '1') CARD TYPE (always '1'=LOCAL) LOCAL BUSY ('0'=OK; '1'=FIFO full) LOCAL DECISION (tracklet) ACTIVE (always '1'=ON) REJECTING (always '0'=OFF;) MASKED ('0'=OFF; '1'=ON) OVERWRITED ('0'=OFF; '1'=ON)	1 1 1 1 1 1 1 1
SOx EOx PAUSE (always '0') RESUME (always '0') CALIBRATE PHY (ignored) RESET ORBIT	1 1 1 1 1 1 1 1	SOx (always '0') EOx (always '0') PAUSE (always '0') RESUME (always '0') CALIBRATE (always '0') PHY RESET (always '0') ORBIT	1 1 1 1 1 1 1 1	Always '0'	8
LOCAL bunch counter	16	LOCAL bunch counter	16	LOCAL bunch counter	16
LOCAL board position in Crate (0-15)	4	LOCAL board position in Crate (0-15)		LOCAL board position in Crate (0-15)	
Status: "0xF"	4	Always '0'		Data: detector plane(s) (1 bit / plane)	
Status: Mask registers (SOx='1'|EOx='1') Data: all strip patterns (not masked) [(X4, Y4), (X3, Y3), (X2, Y2), (X1, Y1)]	32*4	N/A	0	Data: Only masked strip pattern(s) [(X4, Y4), (X3, Y3), (X2, Y2), (X1, Y1)]	32*i
Total number of bits	168	Total number of bits	40	Total number of bits	8*i
Bunches needed to send	21	Bunches needed to send	5	Bunches needed to send	9 to 21

Table 2.3: Regional event format (Renard, 2021)

Coding of SOx, EOx, RESET, CALIBRATE Event in REGIONAL	*Bits*	*Coding of PHY, ORBIT Event in REGIONAL*	*Bits*	*Coding of self-triggered DATA Event in REGIONAL*	*Bits*
START BIT (always '1') CARD TYPE (always '1'=LOCAL) LOCAL BUSY ('0'=OK; '1'=FIFO full) LOCAL DECISION (tracklet) ACTIVE ('0'=OFF; '1'=ON) REJECTING ('0'=OFF; '1'=ON) MASKED ('0'=OFF; '1'=ON) OVERWRITED ('0'=OFF; '1'=ON)	1 1 1 1 1 1 1 1 1	START BIT (always '1') CARD TYPE (always '1'=LOCAL) LOCAL BUSY ('0'=OK; '1'=FIFO full) LOCAL DECISION (tracklet) ACTIVE ('0'=OFF; '1'=ON) REJECTING ('0'=OFF; '1'=ON) MASKED ('0'=OFF; '1'=ON) OVERWRITED ('0'=OFF; '1'=ON)	1 1 1 1 1 1 1 1 1	START BIT (always '1') CARD TYPE (always '1'=LOCAL) LOCAL BUSY ('0'=OK; '1'=FIFO full) LOCAL DECISION (tracklet) ACTIVE (always '1'=ON) REJECTING (always '0'=OFF;) MASKED ('0'=OFF; '1'=ON) OVERWRITED ('0'=OFF; '1'=ON)	1 1 1 1 1 1 1 1 1
SOx EOx PAUSE (always '0') RESUME (always '0') CALIBRATE PHY (ignored) RESET ORBIT	1 1 1 1 1 1 1 1	SOx (always '0') EOx (always '0') PAUSE (always '0') RESUME (always '0') CALIBRATE (always '0') PHY RESET (always '0') ORBIT	1 1 1 1 1 1 1 1	Always '0'	8
REGIONAL bunch counter	16	REGIONAL bunch counter	16	REGIONAL bunch counter	16
REGIONAL crate position (0-15)	4	REGIONAL position crate (0-15)	4	REGIONAL crate position (0-15)	4
Status: Mask registers (SOx='1'| EOx='1') Data: All tracklet inputs (not masked)	4	Always '0'	4	Data: detector plane(s) (1 bit / plane)	4
Total number of bits	40	Total number of bits	40	Total number of bits	40
Bunches needed to send	5	Bunches needed to send	5	Bunches needed to send	5

2.4.5. Gigabit Transceiver protocol

The GBT protocol architecture was created at CERN, for use in the LHC, which requires high bandwidth as well as radiation protection (Moreira et al., 2010). Embedded in the regional cards is a radiation-hardened ASIC known as GBTx. This ASIC contains a high-speed serializer and deserializer that takes data and then transmits them through a laser transmitter. The laser transmitter utilized is a special component manufactured at CERN. The GBT optical link controller is implemented as a module in the CRU firmware. The GBT protocol operates in 3 different modes: standard frame, wide frame, and 8B/10B frame. Figure 2.5 depicts the standard frame mode used in the MID readout chain.

The standard frame is continuously transmitted during a single LHC bunch crossing. It starts with a 4-bit header field, which is necessary for frame-level synchronization of the data stream. Recognizing multiple valid headers implies a proper frame-locking. The opposite implies that the frame synchronization has failed and the frame synchronization cycle must be initialized. The header field can either provide a value “0x5” (data state), which indicates

that the frame includes legitimate data, or “0x6” (idle state), which indicates that the frame does not include valid data. The next four bits are used for slow control, the first two of which are for Internal Control (IC), which is reserved for controlling the GBTx ASIC. The last two slow control bits are for External Control (EC). The payload data and EC fields are not pre-assigned and are utilized for a variety of functions, including Data Acquisition (DAQ), timing and trigger signals, and experiment control, depending on the needs of the MID. The last 32 bits are utilized for forwarding Error Correction (FEC). The remaining 84-bit field, which includes the EC, has an associated bandwidth of 3.36 Gb/s, of which 3.2 Gb/s is allocated to the payload data.

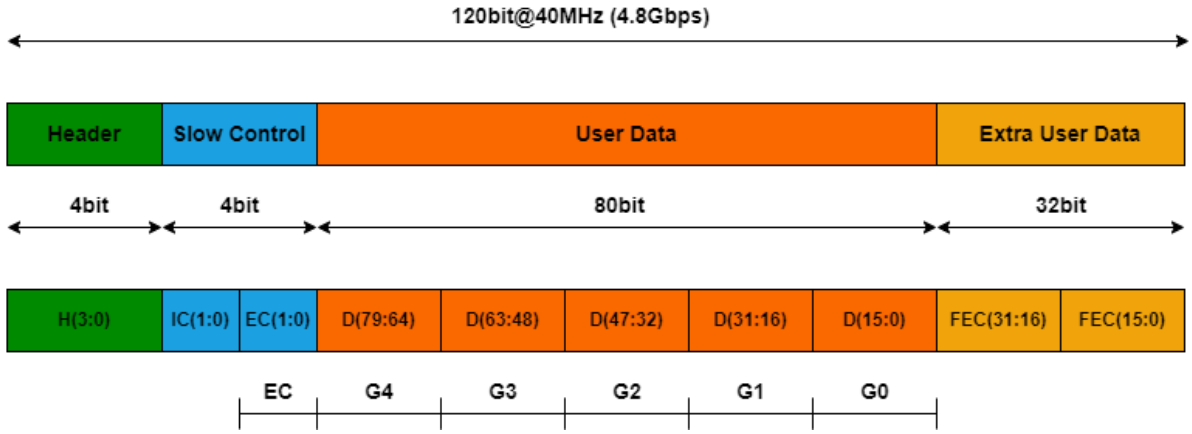

Figure 2.5: Block diagram of the standard mode encoding and decoding

Adapted from (Sierra-Polanco et al., 2018)

Before serialization, the data, EC, and IC fields are put through a scrambling process that concatenates them. In addition to the header, a Reed-Solomon (RS) encoder creates the 32-bit FEC based on the scrambled data. These scrambled data are then transmitted to a deserializer located on the other end, which converts them back to their original format. Both scenarios are represented in Figure 2.6.

The header is used to track frames and synchronize the receiver with the sender. The header is not affected by the scrambling therefore, it is easily detectable. When a GBT receiver is powered up, it enters a frame-lock initialization mode in which it searches for valid headers. After detecting a configurable number of frames with valid headers, it considers that the connection has been established and enters the frame tracking mode, in which it receives data and runs normally while searching for invalid headers.

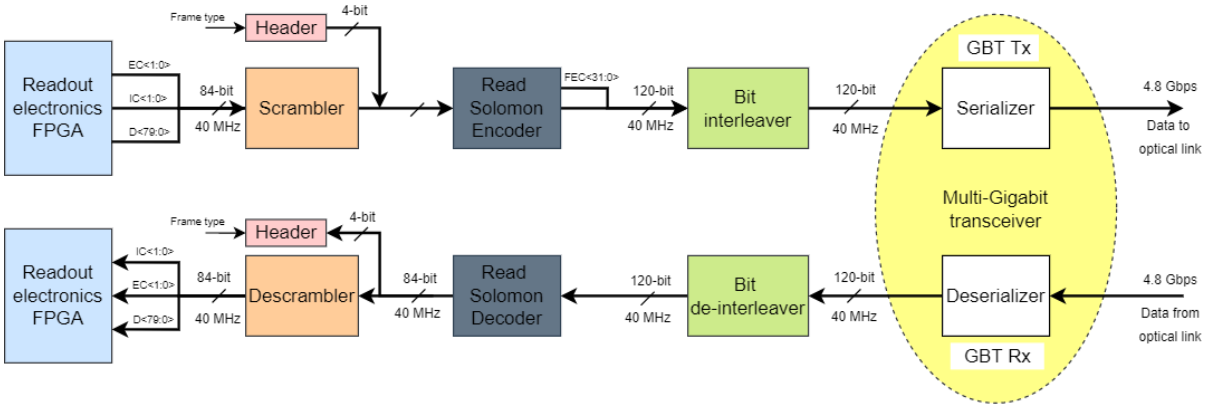

Figure 2.6: Block diagram of the standard mode GBT encoding and decoding

Adapted from (Moreira et al., 2015)

The header is used to track frames and synchronize the receiver with the sender. Therefore, it is not affected by the scrambling and, it is easily detectable. When a GBT receiver is powered up, it enters a frame-lock initialization mode in which it searches for valid headers. After detecting a configurable number of frames with valid headers, it considers that the connection has been established and enters the frame tracking mode, in which it receives data and runs normally while searching for invalid headers.

Once it is determined that a configurable number of consecutive frames is invalid, the synchronization is considered lost and the initialization mode is re-entered. This usually requires multiple invalid frames; hence, an accidental violation of a single random frame is not enough to cause channel synchronization. The data field (80 bits) of the GBT frame is used for data transmission. GBT frames are divided into control frames, data frames, and the header contains data valid only for the latter. The frame starts with a 4-bit identification header. Four headers are defined: IDLE, Start Of Packet (SOP), End Of Packet (EOP), and the Single Word Transaction (SWT). The IDLE frame does not contain any information. SOP and EOP, as the names suggest, mark the beginning and end of the detector data packet, which contains various packet-related metadata. The SWT frame contains any data used for a specific control or data transmission. On the GBT uplink, SWT frames are transmitted between data frames, that is, between EOP and SOP control frames. In the MID readout chain, the SWT frame is used to access the register bus on the regional card. The 2 bytes in the EC payload of the GBT frame are routed to a special slow control ASIC called GBT-SCA (Caratelli et al.., 2015). As mentioned above, the chip is part of the regional card. The communication between the CRU and SCA is handled by the CRU firmware described in the next chapter.

The GBT-SCA has a large number of communication modules, including various protocols. The communication between the regional FPGA and GBT-SCA is carried out through the high-level serial link control protocol. The protocol is based on commands. In contrast to the direct reading and writing of registers, the transaction contains the command ID, transaction ID, and data required by the command. The command ID indicates what the GBT-SCA chip will do, such as read or write registers or perform operations. Each command transaction returns a batch with the same transaction ID. The return package contains status information and returned data. The slow control IC is used for GBTx register access, configuration, and monitoring. This field can also control the laser transceivers that use the main communication modules on the GBTx chip, which can be accessed through its registers.

2.4.5.1. Electrical-links

The GBTx chips on the regional cards communicate with up to 8 local card FPGAs using the standard GBT frame mode. It consists of connecting the GBTx chip and the regional FPGAs through duplex serial electrical links (e-links). Each GBT bi-directional optical link of the readout chain is made up of 10 serial e-links (8 local e-links + 2 regional e-links).

Each of the e-links implemented in the readout chain consists of three signal lines:

- Differential Clock line (dClk+/dClk-): Clock driven by GBTx to the local/regional FPGA
- Differential Downlink data output (dOut+/dOut-): Data from GBTx to the local/regional FPGA
- Differential Uplink data input (dIn+/dIn-): Data line from the local/regional FPGA to GBTx

The MID readout chain is configured to operate at the maximum e-links data rate of 320 Mb/s, with a maximum of 2 e-links per group. As mentioned earlier, each e-link is composed of one differential clock line (dClk+/dClk-), one differential downlink output (dOut+/dOut-), and one differential uplink input (dIn+/dIn-). Thus, the maximum number of differential e-link signals per group is 3 x 2 = 6, equivalent to 6 signal pins per group. Overall, a total of 6 x 5 = 30 configuration pins are dedicated to the e-links. To provide the greatest possible signal quality and transmission reliability, the physical e-link connections are assumed to be differential transmission lines with a differential impedance of 100 Ω and a suitable termination line at the receiver end.

The diagram depicted in Figure 2.7 was created for this study to illustrate how the GBTx chip interacts with the readout electronics via e-links.

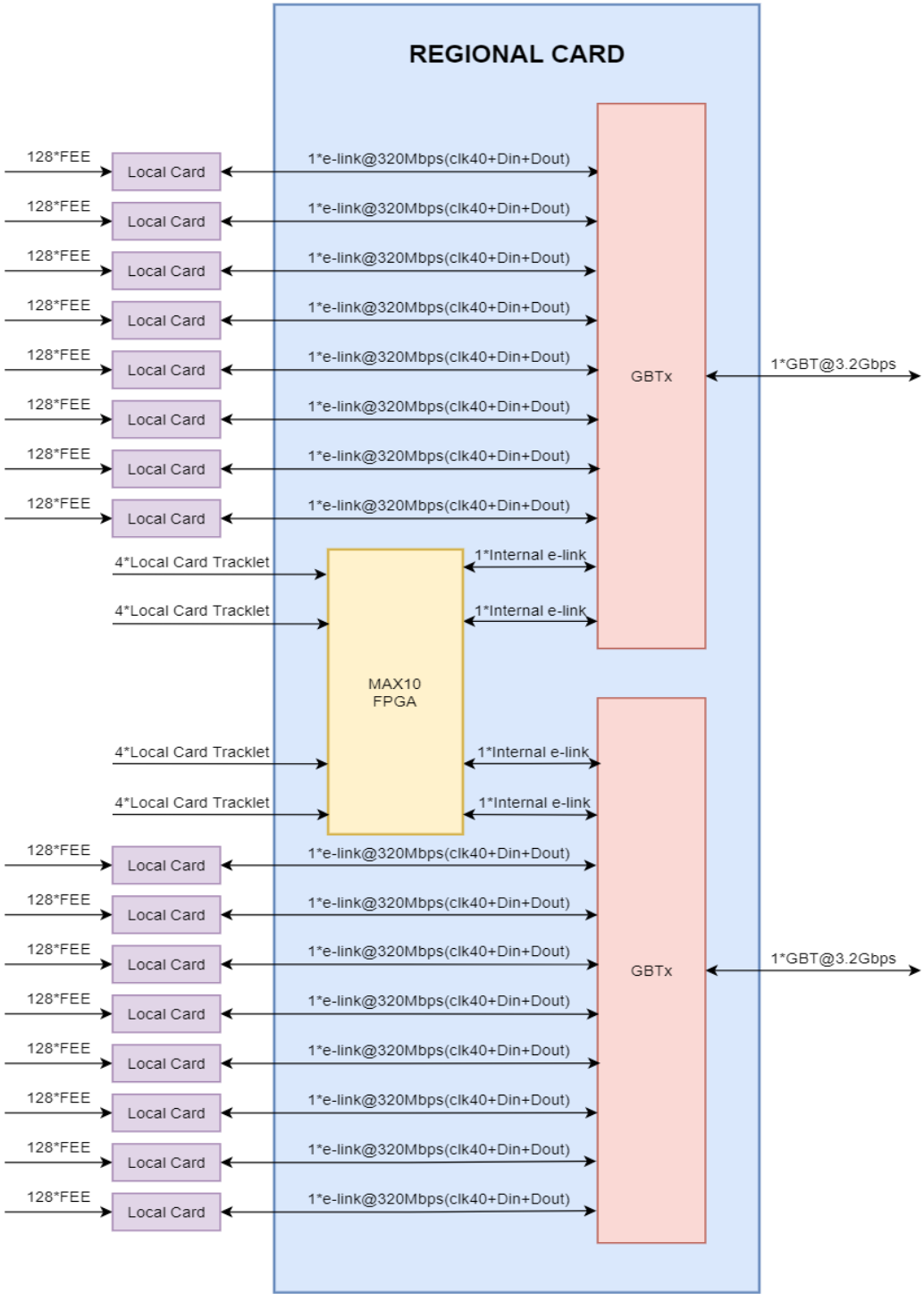

Figure 2.7: E-link configuration between the GBTx chip and the readout electronics. This diagram is created for this work

2.5. Common readout unit

As mentioned in the previous chapter, this study aims to design a bespoke user logic component that will be incorporated into the existing high-speed CRU FPGA. As a result, it is crucial to comprehend the functionalities of the CRU component. This section summarizes the clock and the hardware architectures of the CRU. The CRU firmware is covered later in the next chapter.

2.5.1. Clock tree architecture

The clock tree is designed to utilize a single reference clock for all CRU communication links except for the PCIe interface, which utilizes a built-in 100 MHz crystal oscillator. The CRU card can either be used independently with a built-in 40 MHz crystal oscillator or with a recovered clock retrieved from the TTS optical link. On the other hand, the TTS transceiver requires a constant 240 MHz reference frequency before initialization, which is generated locally with the help of a Phase-Locked Loop (PLL) SI5344 (Skyworks, 2018). The clock recovered from the FPGA is transferred to a high-performance SI5345 PLL (Skyworks, 2018) for jitter attenuation after it has successfully been locked to the incoming stream. The clocks extracted from the SI5345 PLL are then utilized to run the FPGA logic. The SI5345 PLL uses I^2C communication to switch between local and recovered TTS clock modes. The clock generated from the built-in 100 MHz crystal oscillator is utilized to run many other features of the FPGA, including initialization and hardware monitoring.

2.5.2. Hardware architecture

The CRU card is based on an Intel ARRIA 10 FPGA (10AX115S3F45E2G) (Intel, 2022). It is equipped with two Small Form-factor Pluggable (SFP+) connections. One is used for the TTS connection, and the other is used as a backup. The connections from/to the readout electronics are ensured by up to 4x12 bi-directional channel modules, (Broadcom, 2005). These modules can connect to up to 24 GBT links. However, concerning the MID, 32 GBT links are necessary to transfer data from the complete readout electronics. Hence, 2 CRU cards are utilized, one for each side of the plane, and each connected to 16 GBT links**.**

The CRU is equipped with a PCIe edge connector on the rear end, that provides a dual PCIe interface. This interface is synchronized with a 250 MHz reference frequency provided through the connector. The ARRIA 10 FPGA is also linked to temperature and current sensors, as well as an electrically erasable programmable read-only memory with a unique

identifier assigned by the manufacturer during board construction. Other protocols are used to communicate with various peripheral devices. Additionally, tri-color LEDs are installed on the CRUs for maintenance purposes and to easily identify a specific machine among others in the server farm. Finally, the FPGA can be programmed using either a Joint Test Access Group (JTAG) connector, which is useful for software debugging in the laboratory, or a quad Serial Peripheral Interface (SPI) flash module.

Figure 2.8 depicts a functional overview of the hardware emphasizing the characteristics utilized in ALICE CRU. The clock tree, as well as the FPGA and its connections with the different components of importance, are depicted.

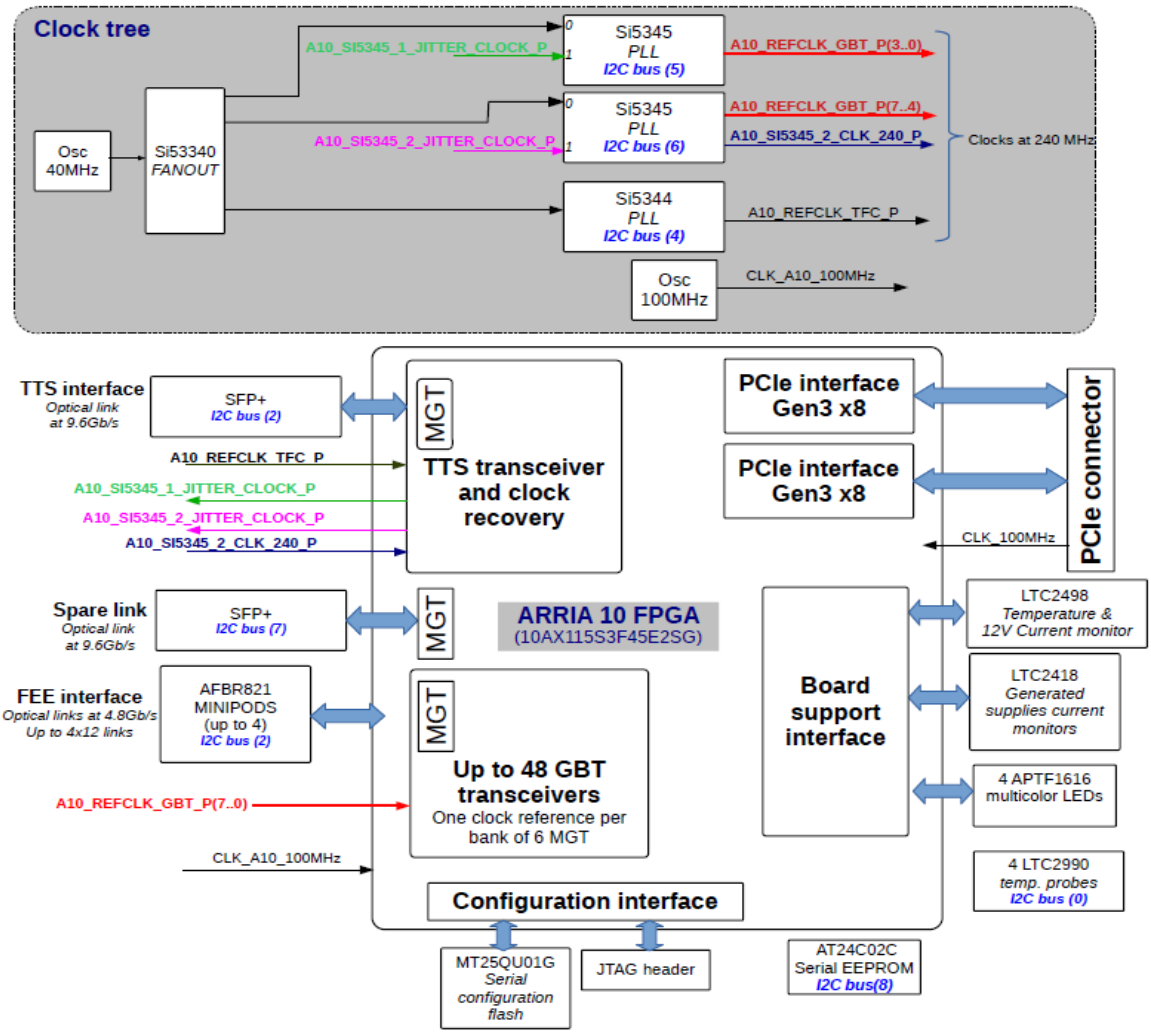

Figure 2.8: A functional overview of the hardware, highlighting the functions used in the CRU (Bourrion et al., 2021).

2.6. Trigger architecture

The ALICE trigger architecture is an amalgamation of multi-link technologies based on several protocols. It has been optimized to function in coherence with the MID subdetector, allowing its readout chain to operate synchronously and efficiently (Kvapil et al., 2021). The trigger architecture relies on the Trigger and Timing distribution System (TTS) ability to efficiently distribute the critical timing and trigger information from the Central Trigger Processor (CTP) to the readout electronics via the Local Trigger Unit (LTU) and CRUs with constant latency over bi-directional 10-Gigabit Passive Optical Network (PON) links. This allows the MID to be read out in continuous and triggered readout mode operations.

2.6.1. Central Trigger Processing

The CTP is an electronic board that receives inputs from a set of triggers from contributing detectors and generates trigger decisions for all subdetectors (Evans et al., 2016). It interacts with up to 24 LTUs, one of which is dedicated to MID. The CTP is essential in trigger architecture as it provides periodic HeartBeat (HB) triggers as well as customized software triggers to the LTU for both continuous and triggered readout mode operations.

2.6.2. Local Trigger Unit

The LTU serves as an interface between the CTP and CRUs. It provides a clock, Orbit, and external trigger inputs as well as allows monitoring and control using ethernet bus protocols. The LTU is a 6U VME-type board equipped with a Xilinx Kintect FPGA with 2 Gigabytes of DDR4 memory (Krivda et al., 2018). It can be configured in two different ways (global and stand-alone). In global mode, the LTU acts as a transparent interface between the CTP and the CRU. It converts signals and provides online monitoring. Contrary, in the stand-alone mode, the LTU emulates the CTP protocol, allowing the MID team to perform tests, and calibration activities independently of the CTP, when the CTP is either unavailable or not necessary.

2.6.3. Continuous and triggered readout modes

An important requirement from ALICE is that the majority of subdetectors including the MID must implement a new type of readout mode on their systems. This new type of readout mode is called continuous readout mode and differs from the current practice.

In this mode, data are no longer bounded by physics trigger but rather by various data streams, namely HeartBeat Frames (HBFs) with a predetermined period of (89.4 µs) (Costa et al, 2017). These HBFs will then be aggregated by the CRU firmware into larger blocks called Time Frames (TFs) and transmitted to the O² systems for reconstruction and error handling. The HBF boundaries are determined by the HeartBeat triggers, which are transmitted by the CTP via the LTU.

The readout electronics cards in the MID readout chain are modified to handle the combination of Physics and HeartBeat triggers. Each regional and local card autonomously tags the data using the copy of the LHC Orbit and the bunch crossing ID. For continuous readout mode, the payload data are sent as a continuous flow of successive frames each preceded with a header containing the time-based tagging. The triggered mode operates in the same way as the continuous mode with a few variations, it only sends a payload data block preceded with a header upon reception of physics triggers. Figure 2.9 shows how the physics and HeartBeat triggers are used for the continuous and triggered readout modes.

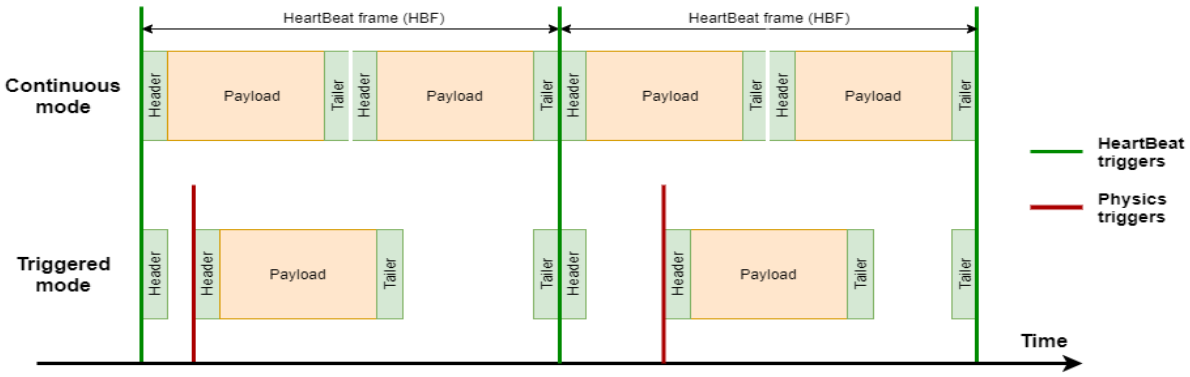

Figure 2.9: Continuous and triggered mode trigger configuration

Adapted from (The ALICE Collaboration, 2014)

2.6.4. Passive Optical Network message

The PON is a point-to-multipoint network architecture that uses optical splitters to enable an Optical Line Terminal (OLT) to interact with several Optical Network Units (ONUs). As illustrated in Figure 2.10, the PON technology allows the timing and trigger message to be split among multiple CRUs of the readout chain using a single link.

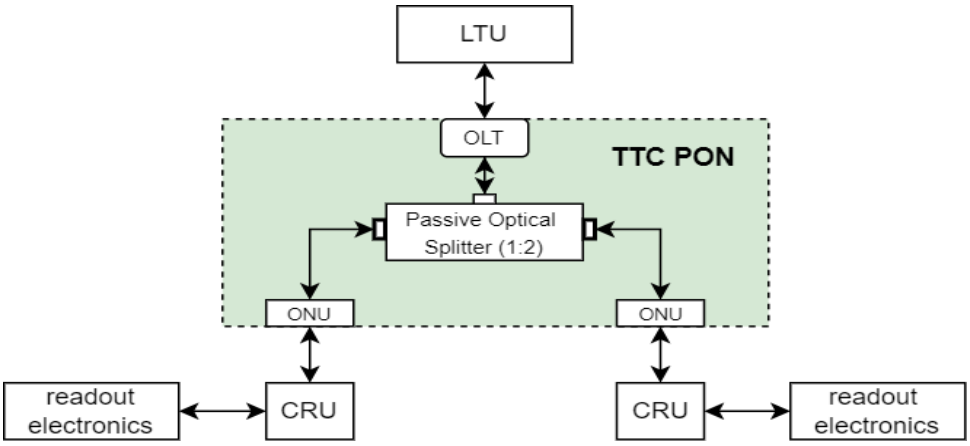

Figure 2.10: Representation of the PON architecture implemented in the MID readout chain

Adapted from (Mitra, 2018)

The PON downstream (CTP to CRU) and upstream (CRU to CTP) messages are described as follows:

- The PON downstream message is based on a 240-bit word transmitted synchronously with the LHC clock from CTP to the CRUs. The PON internally uses 40-bit, leaving 200-bit available for the subdetectors use. The PON downstream message is summarised in Table 2.4. The trigger type information is described in Table 2.5.
- Upon reception of an HB trigger, each CRU of the readout chain transmits the PON upstream message of 56 bits to the CTP, alternatively called HeartBeat acknowledge message (HBam). The HBam carries information about the CRU status. The CTP then collects the HBam from all CRUs acknowledging that data have been successfully collected.

Table 2.4: PON downstream message (Bourrion et al., 2019)

No. of Bit	Name	Description
<31:0>	TType	Trigger Types data
<11:0>	BCID	Bunch crossing identification
<31:0>	Orbit	Orbit counter
<0:0>	TTValid	Trigger Type data valid
<7:0>	HBM header	HeartBeat message header
<31:0>	First ORBIT of TF/HBMTF	HeartBeat message Time Frame
<0:0>	HBMValid	HeartBeat message valid

Table 2.5: Description of the Trigger Type bits (Bourrion et al., 2019)

Bit	Name	Description
0	Orbit	Orbit flag
1	HB	HeartBeat flag
2	HBr	HeartBeat reject flag
3	HC	Health Check
4	PhT	Physics Trigger
5	PP	Pre-Pulse Calibration
6	Cal	Calibration trigger
7	SOT	Start of Continuous
8	EOT	End of Continuous
9	SOC	Start of Triggered Data
10	EOC	End of Triggered Data
11	TF	Time Frame
…	…	Spare
29	TPCSync	TPC synchronization
30	TPCReset	TPC reset
31	TOF	TOF special trigger

2.6.5. MID custom trigger type format

The TTC-PON trigger types contain useful information to accommodate various subdetectors in the ALICE experiment, although not all triggers are utilized by the MID. A special request from the MID team is to reduce the bandwidth transmitted to the subdetector readout electronics by compressing the 32-bit TTC-PON trigger types into a bespoke 8-bit trigger types format that will easily be interpreted by the readout electronics. This task is handled by the CRU firmware through a specific configuration of registers, and the agreed-upon format is described in Table 2.6.

Table 2.6: Custom trigger type format implemented to accommodate for the MID readout electronics (Renard, 2021)

MID trigger type	CTP trigger type	CRU message to Readout electronics	FEE trigger code
SOx (Start Of Run)	9: SOC 7: SOT	Update internal ORBIT, BCID, bunch counter Transmit command to all e-links Reset event buffers Start assembling events Start sending events	0x80
EOx (End Of Run)	10: EOC 8: EOT	update internal Orbit, BCID & bunch counters Transmit command to all e-links Assemble last events Send last events	0x40
TF (Timeframe)	11: TF	Transmit command to all e-links	0x20
RUNNING (Run status)	14: RS	Transmit command to all e-links	0x10
CALIBRATE	6: CAL	Update internal Orbit, BCID & bunch counters Transmit command to all e-links	0x08
PHY	4: PhT	Update internal Orbit, BCID & bunch counters Transmit command to all e-links	0x04
RESET	12: FEErst	Update internal Orbit, BCID & bunch counters Transmit command to all e-links Stop assembling events Stop sending events	0x02
ORBIT	0: ORBIT	Update internal Orbit, BCID & bunch counters Reset internal MID's bunch counter Transmit command to all e-links	0x01

2.7. Online-Offline computing system

The O^2 is a new computing system implemented to support both online and offline reconstructions (Buncic et al., 2015). Its architecture is made up of hundreds of thousands of processes that are spread over several nodes, and perform readout, processing, and storage. The architecture is shown in Figure 2.11. The online reconstruction is based on two types of computing nodes (FLP and EPN), while the offline reconstruction relies on the connection of multiple high-performance clusters, i.e., the Grid to move data from one storage to another.

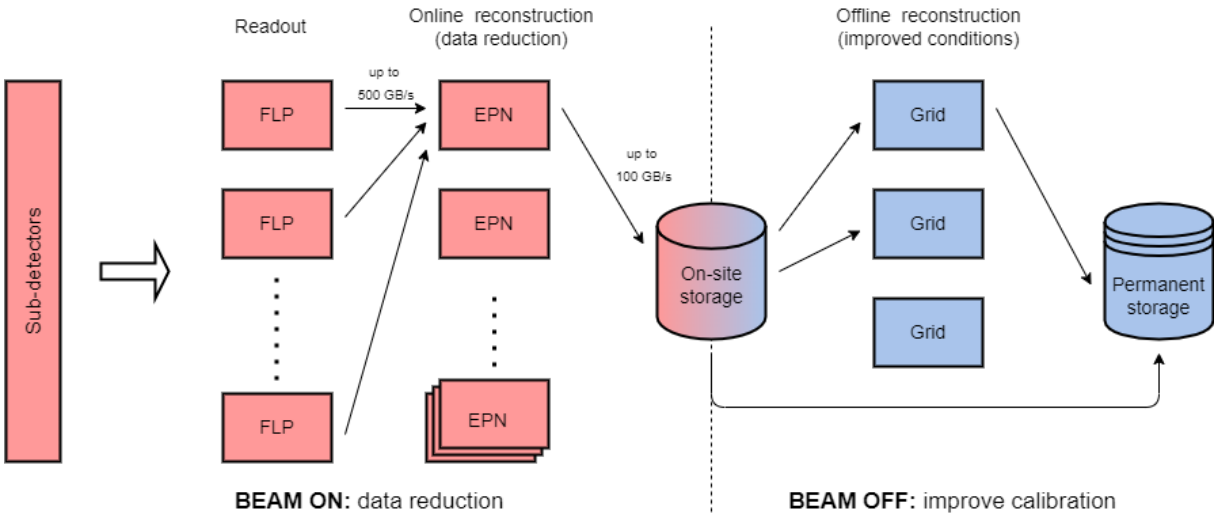

Figure 2.11: O² computing system architecture

Adapted from (Eulisse et al., 2019)

2.7.1. First Level Processor

Data from various subdetectors including the MID are transmitted to dedicated DELL POWEREDGE R740 rack servers (Costa, 2019), namely, FLP nodes, which house the CRUs as the hardware interface to the front-end and/or readout electronics depending on the subdetector architecture (refer to Figure 2.1 for the MID readout chain architecture). In total, 270 FLP nodes are used in the O² computing system (Richter et al., 2019). Each FLP compresses, merges, splits data into Sub-Time Frames (256 HeartBeat Frames from a single FLP), and stores them until they are forwarded to the EPN. As a result, the subdetector data are analyzed instantly on the EPN.

2.7.2. Event Processing Node

The EPNs of the O² computing system offer computational resources for data reconstruction. A many-to-many data distribution network configuration recomposes all Sub-Time Frames obtained from the subdetector FLPs over a Time Frame period (~20 ms) on one of the dedicated EPNs in a rational order (Nešković et al., 2018). Each EPN reconstructs these Sub-Time Frames and uses a variety of tasks based on the subdetector (e.g., clusterization and tracking for individual subdetectors) as a means to compress raw data-related information and reduce the size of each Sub-Time Frames from 500 GB/s to an aggregate rate of up to 90 GB/s before forwarding them to the on-site storage.

2.7.3 Data storage

The EPN uses on-site storage as a buffer between online and offline data processing. To prevent overwhelming the EPN nodes already overburdened by data gathering and processing responsibilities, data transfer operations between the online system and the Grid are handled by specialized nodes of the order of 10 Data Movers (DM) (Buncic et al., 2015). The on-site storage is physically separated from the EPN nodes, but it nevertheless still provides a global storage area that is accessed by all EPNs and DMs. To minimize data pile-ups in the EPNs and DMs, a large bandwidth protocol is implemented.

2.8. Detector control system

The DCS is used to monitor and control readout electronics from various subdetectors located in the ALICE cavern. It accesses the readout electronics via the FLP-CRU through the GBT links. The primary protocol considered for communication between the FLP and DCS is called Alice Low-Level Front-end (ALF) on the FLP side and Front-End Device (FRED) on the DCS side (Tkácik et al., 2020).

In the current MID readout chain configuration, the ALF can read/write registers on the regional and local card firmware modules and publish data in the DCS control room using a Distributed Information Management (DIM) service (Tkácik et al., 2020).

The DIM is a communication system for distributed/mixed environments, that provides a network transparent inter-process communication layer. The FLP node hosts a DIM server, which acts as a bridge between the DIM network and the CRU driver, enabling the DCS to interact with the readout electronics from the control room without requiring physical access to the FLP node.

CHAPTER 3

CRU firmware

This chapter presents the current CRU firmware implemented in the MID readout chain and illustrates how the user logic component fits into it. The architecture of the CRU firmware is described as well as the design requirements and specifications of the MID user logic component.

3.1. Introduction

The standard approach of delivering raw data to the O^2 system is no longer sufficient to fulfill the needs of the newly enhanced MID subdetector. As a result, an alternative option was presented to the collaboration (ALICE Collaboration, 2014). Since the introduction of version 1.0.0 at the beginning of 2018, the CRU firmware can be configured in two different manners. The first configuration is the common mode, which interacts through various interfaces with various systems enabling the possibility to read out any subdetector without conducting any first stage data analysis in its CRU firmware. This is also known as the “CRU firmware without user logic”. The second configuration is the user logic mode, which is only available to high-performance subdetectors, such as MID, that requires first stage data analysis before online and offline reconstruction. In user logic mode, the MID readout electronics data are forwarded to the user logic component for analysis. It is the responsibility of the MID team to decide on how these data will be analyzed in the user logic component.

One of the most important features of the CRU firmware is its ability to switch between common and user logic modes without reloading distinct firmware on the FPGA. For testing and debugging purposes, the CRU firmware can also be configured to run both modes simultaneously (Bourrion et al., 2021).

3.2. Firmware description

The CRU firmware architecture is illustrated in Figure 3.1. From left to right, the main interfaces are the GBT wrappers, Board Support Package (BSP), Datapath Wrappers (DWs), Timing and Trigger Control (TTC), Dedicated Data Generator (DDG), slow control, and PCIe endpoints. All of these interfaces provide indispensable functionalities to the CRU firmware, and at the heart of it all is the user logic component, which will be unique to MID.

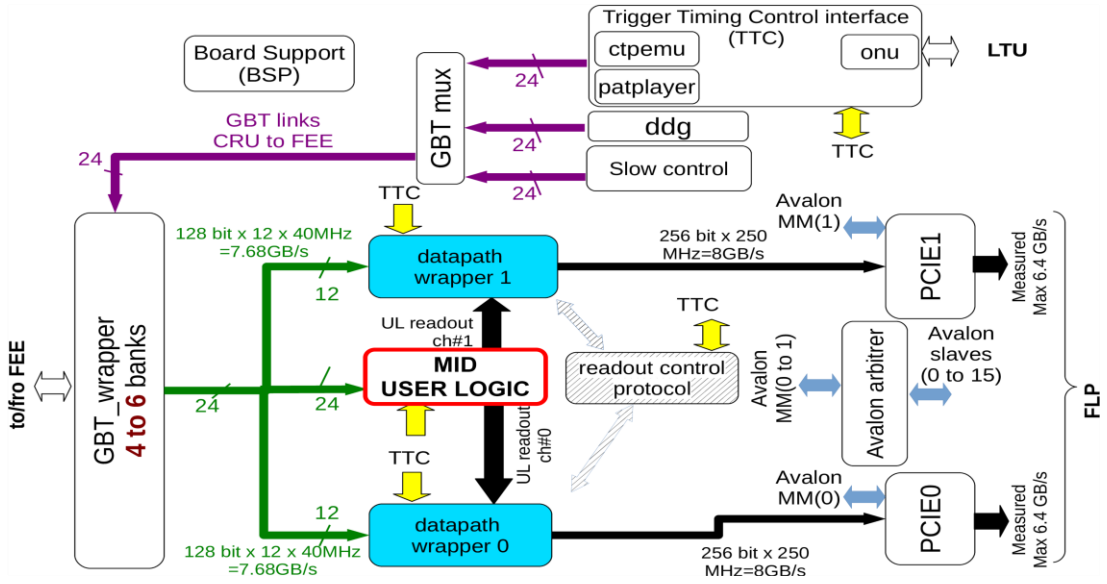

Figure 3.1: CRU firmware architecture

Adapted from (Bourrion et al., 2021)

3.2.1. GBT wrapper

The GBT wrapper acts as a conduit between the MID readout electronics and the CRU firmware. It is made up of up to six banks, each with six bidirectional GTB links. The GBT wrapper can connect up to 36 GBT links in total, but only 24 are made available to the subdetectors. The remaining is preserved as a backup, and the configuration of the banks is done via slow control registers. The GBT links, as previously stated, are bidirectional. The purple bus line in Figure 3.1 depicts the uplink direction (CRU to FEE), whereas the green bus line represents the downlink direction (FEE to CRU).

3.2.2. Datapath wrappers

In contrast to user logic mode, where all available GBT uplinks (FEE to CRU) are directly attached to the user logic component, and subsequently diverted to the datapath wrappers after first-stage analysis, the GBT uplinks in common mode are uniformly distributed throughout the two identical datapath wrapper blocks. Depending on the mode selected in the CRU firmware, each datapath wrapper receives trigger information, gathers and combines raw data from the subdetector readout electronics over up to 12 GBT links, and/or utilized compressed data from the user logic readout channels as input. These data are then delayed for a few clock cycles to allow for the inclusion of the Raw Data Header (RDH). The RDH enables the O² system to identify all data blocks transmitted by each subdetector in the ALICE experiment.

3.2.2.1. Raw Data Header

The data blocks transmitted by the MID readout electronics must be identified and arranged in such a way that they can easily be recognized and processed efficiently in the receiving FLPs. As a result, a standard RDH format has been designed and deployed. The RDH format illustrated in Table 3.1 is generated by either the datapath wrappers in common mode or the user logic component in user logic mode to provide the necessary information required to identify the structure of the data in the O^2 system.

Table 3.1: Raw Data Header format

Adapted from (Costa, 2021)

RDH 0 [128-bit]				
32-bit	FEEID [31-16]		Header size [15-8]	Header version [7-0]
32-bit	Reserved [31-16]		System ID [15-8]	Priority bit [7-0]
32-bit	Memory size [31-16]		Offset [15-0]	
32-bit	DW [31-28]	CRU ID [27-26]	Packet cnt [15-8]	Link ID [7-0]
RDH 1 [128-bit]				
32-bit	Reserved [31-12]			BC [11-0]
32-bit	Orbit [31-0]			
32-bit	Reserved [31-0]			
32-bit	Reserved [31-0]			
RDH 2 [128-bit]				
32-bit				
32-bit	Reserved [31-24]	Stop bit [23-16]	Page cnt [15-0]	
32-bit	Reserved [31-0]			
32-bit	Reserved [31-0]			
RDH 3 [128-bit]				
32-bit				
32-bit	Reserved [31-16]			
32-bit	Reserved [31-0]			
32-bit	Reserved [31-0]			

The RDH fields are either populated using data transmitted by readout electronics as well as the LTU, or ctpemu (when the LTU is unavailable). The RESERVED fields are initialized to zero and saved for future usage. Table 3.2 provides a more in-depth description of various RDH fields.

Table 3.2: Field description of the Raw Data Header

Adapted from (Costa, 2021)

RDH	Name	Size in bit	Description	Default value
0	Header version	8	Header version number	0x06
	Header size	8	Size of the RDH in byte	0x64
	FEEID	16	FEE identification	-
	Priority bit	8	Fast forward packet	-
	System ID	8	Unique ID assign to subdetectors	0x25 (MID)
	Offset packet	16	Payload size before next RDH	0x2000
	Memory size	16	Size of the subdetector payload	0x2000
	Link ID	8	Number used to identify the link	-
	Packet counter	8	Counter to keep track of packets	-
	CRU ID	12	Number used to identify the CRU	-
	DW	4	Number used to identify wrappers	-
1	BC	12	Trigger orbit from TTC	-
	Orbit	32	Trigger bunch crossing from TTC	-
2	TRG type	32	Trigger type from TTC	-
	Page counter	16	Counter to keep track of pages	-
	Stop bit	8	A bit to identify the last RDH page	-
3	Detector field	32	Subdetector specific field	0xA003(MID)
	PAR bit	16	Field used by the subdetector	-

3.2.3. Board Support Package

The BSP provides access to GBT link parameters such as temperature and optical power, as well as the FPGA serial number. It also allows the user to reboot the FPGA into a stable condition. The approach used is to divide the flash memory into two sections, one for the stable firmware version and one for the beta version (e.g., version generated after implementing the user logic component). In the event of a power interruption or defective user logic component, the CRU firmware can simply be restored to a stable state.

3.2.4. Timing and Trigger Control interface

The basic function of the TTC interface is to communicate with the LTU and provide timing and trigger information to other interfaces. The TTC interface is divided into three distinct sub-components: ONU, CTP emulator (ctpemul), and Pattern player (patplayer). As the name implies, the ONU is named after the optical network unit found in the TTC-10G PON. This sub-component handles the communication with the LTU. The ctpemu is used for testing and debugging. During standalone tests, it may also be used to emulate the same information provided by the LTU-CTP. Finally, the patplayer provides a programmable sequence to the readout electronics when a HeartBeat or physics trigger bit is supplied by the ONU or ctpemu.

3.2.5. Detector Data Generator

The DDG is capable of imitating any subdetector behavior and reproducing the data transmitted when the subdetector front-end and readout electronics are not available. The injection of the DDG data is made possible by simply configuring the GBT wrappers to operate in internal-loop-back mode. This enables the system to generate data without the need to connect to a physical subdetector. The DDG is an essential component for evaluating and testing the CRU firmware and software.

3.2.6. Slow Control

The slow control component distributes control sequences and collects status information from the readout electronics via GBT links using the SWT protocol. To enable slow control read/write access, the CRU firmware must be configured to SWT protocol using the GBT-MUX component.

3.2.7. PCIe Data Management

The MID data stream passes through the Data Management interface, which moves it from the two Datapath Wrappers to the FLP memory. However, in order to avoid mixing up the data flow, the data stream is transmitted to the FLP server through a dual endpoint PCIe gen. 3x8 interface. This means that each half of the total GBT links is linked to a single endpoint through its datapath wrapper. Hence, the data flow is distributed equally among the two endpoints. Furthermore, communication with the Software is also achieved through the PCIe interface.

3.3. Firmware resource usage

A significant amount of effort was invested by the CRU experts in order to find a reasonable trade-off between lowering the ARRIA 10 FPGA resources and delivering a flexible firmware capable of addressing the demands of the majority of subdetectors encountered in the ALICE experiment. As shown in Table 3.3, the most recent firmware version (v3.10.0) implemented in the CRU at iThemba LABS uses about 123k/427k (29%) Adaptive Logic Module (ALM) and 1084/2713 (40%) Read Access Memory (RAM) blocks of the available resources. As a result, the MID readout chain has enough available resources to implement a fairly complicated user logic algorithm in the CRU firmware. However, to allow for future enhancements, it is advised that after inserting the user logic component, the overall ALMs and block RAMs consumption should be limited to less than 75%.

Table 3.3: FPGA resource usage of the CRU firmware before insertion of the user logic

Resource name	Total in ratio	Total in percentage
Logic utilization (in ALMs)	123,381 / 427,200	29 %
Pins	369 / 960	38 %
Block memory bits	17,514,564 / 55,562,240	32 %
RAM Blocks	1,084 / 2,713	40 %
Digital Signal Processing Blocks	0 / 1,518	0 %
RX channels	41 / 72	57 %
TX channels	41 / 72	57 %
Phase Locked Loops (PLLs)	59 / 144	41 %

Special attention will be given to the ALMs and RAM blocks consumption throughout the implementation of the user logic, as it might consume a significant amount of these resources. The user logic design will not make use of the DSP blocks available in the FPGA, hence, the amount of DSP blocks used will remain unchanged. The RX and TX channels as well as the PLLs used in the CRU firmware will also remain unchanged.

3.4. User logic component

Before implementing the user logic component into the CRU firmware, it is important to understand the current condition of the data and the impact that the user logic will have on them.

The CRU firmware receives 80-bit of data from a single GBT link. Following the completion of a HeartBeat Frame, which takes about (89.4 µs / 25 ns = 3576 BCs or LHC clock cycles), the CRU firmware would have acquired (80-bit x 3576 BCs = 28608-bit) of raw data. These raw data would be sent to the O² system in a packet form. During lead-lead collisions, each of the readout electronic cards is expected to generate around 4 events per Heartbeat Frame, this is based on the number of collisions occurring every 89.4 µs (i.e., 20 µs / 89.4 µs = 4). The regional card takes 5 BCs to transmit an event while the local card takes up to 21 BCs (refer to Table 2.2 and Table 2.3). This results in 40-bit per regional event and 160-bit for the 4 events, as well as 168-bit per local event and 672-bits for 4 events. As mentioned in section 2.4.5.1, a single GBT link is composed of 8 local e-links and 2 regional e-links, leading to a total amount of valuable data allocated in the packet of ((160x2) + (672x8) = 5696-bit). As stated in section 2.7.1, the FLP transmits data to the EPN per Sub-Time Frame, which is equivalent to 256 HeartBeat Frames. This indicates that scientists handling the

offline reconstruction must go through 256 packets to identify (5696-bit x 256 = 1458176-bit ~ 182 KB) out of (28608-bit x 256 = 7323648-bit ~ 915 KB) in order to reconstruct the events. This is quite inconvenient for the MID, especially when all 32 GBT links are operational.

The user logic takes a different approach to process the MID raw data by only transmitting valuable data to the O² system. Hence, decreasing the amount of data transmitted to the EPNs by 80%, and improving the readability of the data.

Following extensive discussions with the CRU firmware experts at CERN, members of the MID team, and subsequent update meetings, the following user logic requirements, and specifications were established.

3.4.1. Design requirements

The user logic component must adhere to the readout control protocol shown in Figure 3.2. This protocol is pretty straightforward: the user logic receives input flow and routes it to the datapath wrapper memories through user logic readout channels. Data are transmitted in 256-bit words with Start Of Packet (SOP) and End Of Packet (EOP) signals to indicate the beginning and end of the packet. The valid signal indicates the validity of the data. The RDH is inserted at the start and end of the packet. It includes a Page counter that provides packet identification inside the associated HeartBeat Frame as well as a Stop bit that indicates whether or not the last packet for this specific HeartBeat Frame has been transmitted.

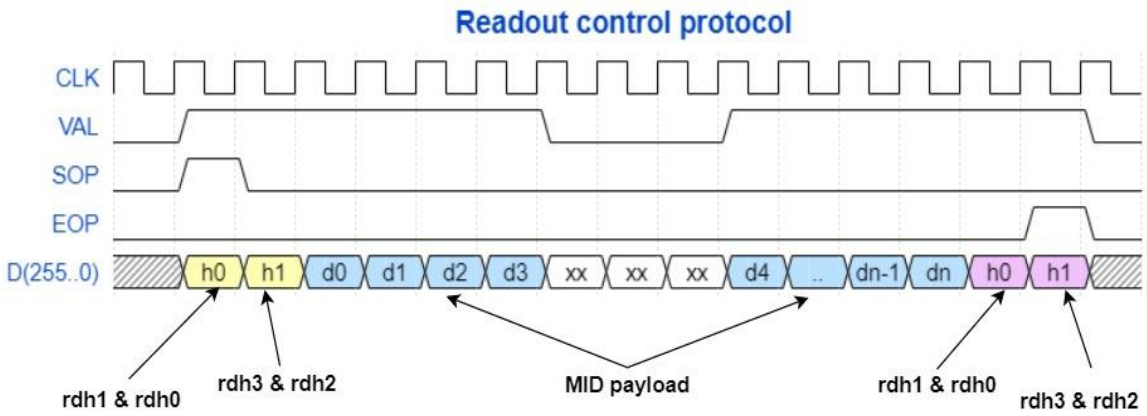

Figure 3.2: Data flow from the user logic component to the datapath wrappers using the readout control protocol

Adapted from (Bourrion, 2015)

During update meetings, additional requirements were introduced, which are detailed below:

- A packet must have a maximum size of 8 KB, which should include a payload and a valid RDH.
- The RDH must have all fields correctly filled out by the user logic.
- Each packet must have a valid page counter, which should be reset at each HeartBeat Frame transition, and a STOP bit must be set.
- There must be a minimum instantaneous gap of one clock cycle between the preceding EOP and the next SOP (this limitation is due to the datapath wrapper component, which requires some delay to properly store data in the memory).
- In triggered readout mode, packets should only contain physics event data and all remaining events should be discarded by the user logic with the exemption of the orbit event.
- If no valid data are transmitted by the readout electronics, the use-logic should generate an empty packet with a valid RDH upon receiving a HeartBeat trigger (this allows the O² systems to keep track of different HeartBeat Frames).

3.4.2. Design specifications

The user logic design specifications are intertwined with the challenges raised in the problem statement (section 1.3). In order to overcome these challenges and achieve the goals stipulated in the objectives (section 1.5), an agreement with the MID team resulted in the design specifications outlined below.

3.4.2.1. Event identification

The GBT wrapper forwards the readout electronics data to the user logic component with a specific bit indicating whether or not the data transmitted are valid. These valid data will then be stored in appropriate registers and event identification will be performed instantly. At this level, the primary goal is to perform zero suppression by identifying the relevant event information (start bit, card type, etc…) transmitted by the local and regional cards.

In contrast to the regional card, which transmits event data with a fixed size of 5 bytes (40-bits), the local card transmits event data with a variable size ranging from 5 to 21 bytes (40 to 168-bit). The local event data contains header information as well as strip patterns from

the four MID detection planes attached to it. The last four less significant bits of the fifth byte indicate which of the four detection planes is transmitting data (1-bit per plane). The user logic will use these bits to predict the size of the event data and only collect strip patterns from the identified detection planes. This allows complete accuracy of the event identification and eventually leads to a drop-in data rate transmitted to the O^2 systems.

3.4.2.2. New regional and local event formats

As shown in Figure 2.7, a single regional crate contains two GBT optical links, each of which carries information from 10 distinct e-links (8 local e-links + 2 Regional e-links). In total, four regional e-links originate from the same crate. Hence, the event data delivered by these four e-links are nearly identical with exception of their 5th byte, which is not very useful to the O^2 systems as it does not provide the current position of the e-link in the GBT frame. In order to differentiate these regional e-links and facilitate the decoding process at the O^2 level, the user logic will assign a unique ID block to each of these four regional e-links based on their location in their respective GBT frame.

A similar issue can also be observed with the local e-links, although they provide valuable information, they quickly become a challenge once two or more crates are involved. Each crate of the readout chain contains about the same number of local cards with the same local card ID ranging from 0 - 15. Once data from multiple crates are merged, they quickly become a pool of untraceable data. As a result, the existing regional and local event formats need to be altered. As depicted in Figure 3.3, the new regional event format includes an additional unique ID block, whereas the new local event format includes a new crate ID block to indicate where it originated.

3.4.2.3. Synchronization

To solve the issue of desynchronization, the user logic will make use of the handshaking protocol. Data from each e-links will be routed through a pair of buffers, referred to as the sender and the receiver respectively. The sender and the receiver will continuously exchange information via several signals.

This handshaking protocol will rely heavily on the Orbit trigger bit, which is found in the local and regional event data. Its purpose will be to filter and synchronize the data. The sender

will send a synchronization request signal to the receiver upon receiving an Orbit trigger. The receiver, in turn, will react with an acknowledge signal only after a third-party submodule has successfully collected the synchronization request from all e-links belonging to the same GBT link. By monitoring the Orbit trigger, it is easier to keep all e-links and GBT links synchronized. The Orbit trigger is also used to reset the bunch crossing counter in the readout electronics firmware, indicating the beginning of a new HeartBeat Frame.

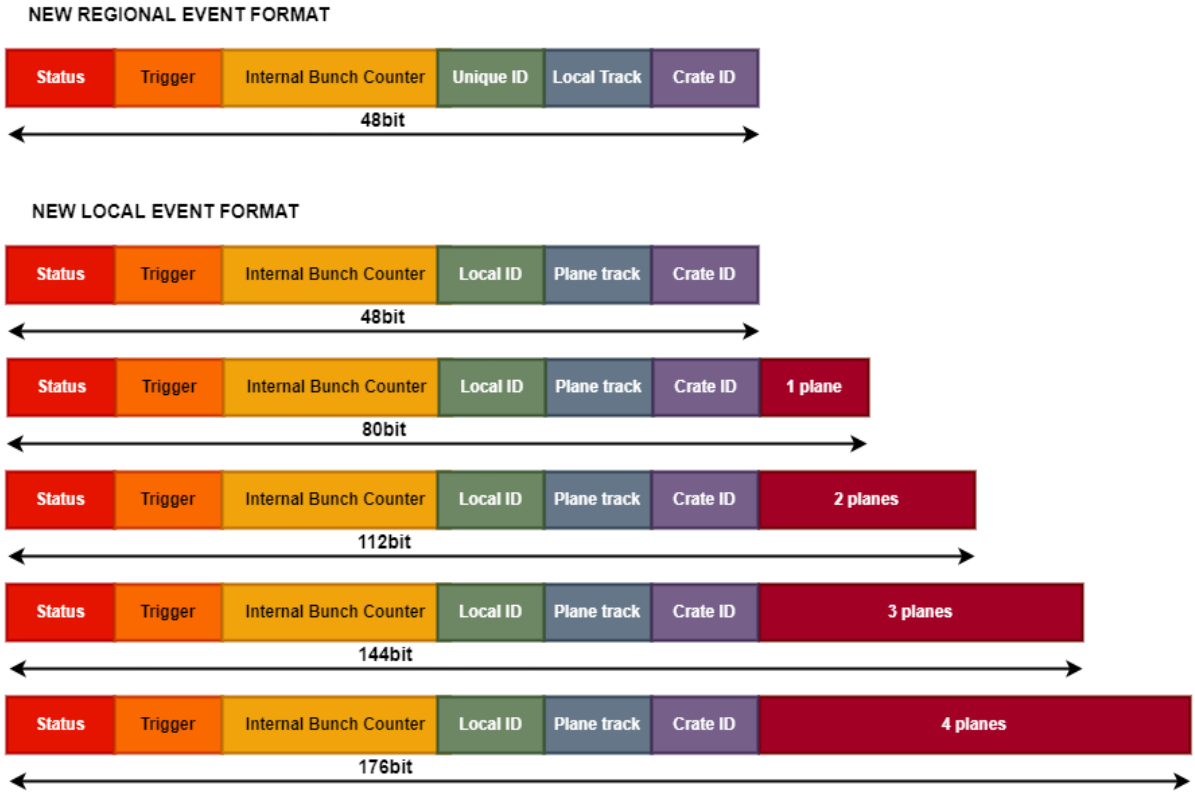

Figure 3.3: New regional and local event formats created for this work

Although this approach appears to be the most efficient, it should be noted that there are certain caveats to it. In the event of a failure or if one of the readout cards stops transmitting data for whatever reason while the run is still ongoing, the receiver module will remain idle until the sender sends a request, which will be unlikely to occur. This will affect the synchronization process and eventually lead to system failure. To overcome this sort of event, a very complex logic must be implemented in the synchronizer. This is described in more detail in the next chapter.

3.4.2.4. Payload data block formats

Following the data synchronization, the data belonging to the same Orbit or HeartBeat Frame will be arranged in small blocks of 256-bits in ascending order. A counter will be implemented to ensure that the 8 KB limit is not exceeded. These blocks of data, namely payload data, will then be combined with the appropriated RDH blocks and streamed as packets to the Datapath wrappers while adhering to the readout control protocol. However, to properly construct a custom O^2 software algorithm that would analyze the payload, a pre-defined data block format has to be agreed upon within the MID team before the implementation of the user logic component into the readout chain. For this specific purpose, two different approaches have been investigated:

Fixed data block format

A way to decrease the data rate and allow fast-tracking is to introduce a fixed data block format. All pre-analyzed events collected from a single GBT link can be used to generate data blocks of 256-bits composed of a header (32-bit) and a body (224-bits). Since each of the two regional e-links is linked to up to 4 local cards. The Track field found in these regional events reveals which of the four local cards is transmitting data belonging to the same event (1-bit per local card). Taking this into consideration, a fixed structured data block composed of a regional event (header) and a local event (body) can be implemented and is described in Table 3.4. The number of data blocks generated per regional event will be computed based on the Track field information, and it will range from 1 to 4 blocks. The header information will be duplicated for each data block generated from the same regional event. The data flow of the fixed data format is illustrated in Figure 3.4.

Table 3.4: Field description of the fixed data block format

HEADER (regional event) [32-bit]	BODY (local event + reserved) [264-bit]
Trigger [8-bit]	Card ID [4-bit]
Internal bunch counter [16-bit]	Tracklet [4-bit]
Unique ID [4-bit]	Strip patterns [128-bit]
No. local cards fired [4-bit]	Reserved [88-bit]

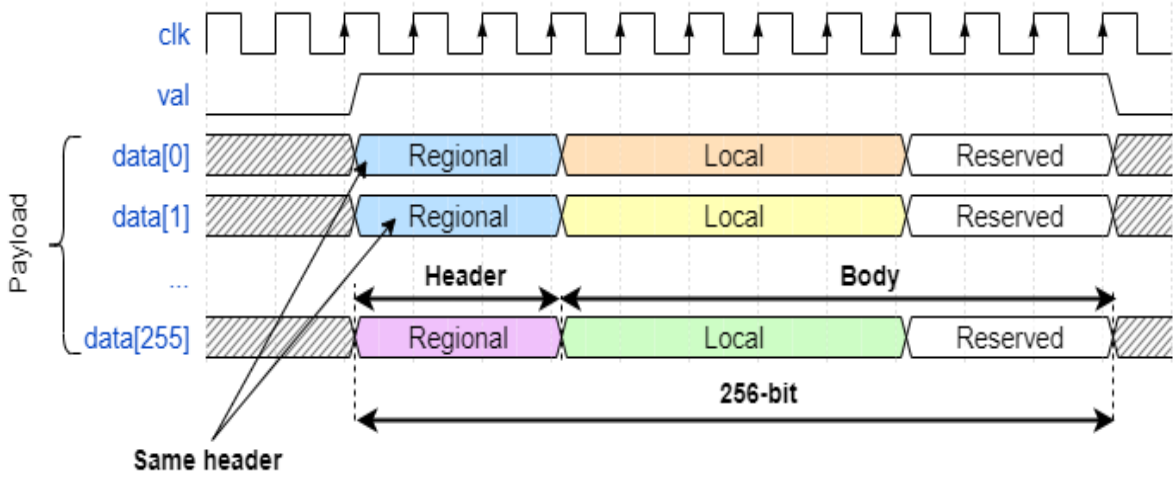

Figure 3.4: A example of the fixed data block format designed for this work

Serialized data block format

Another way to significant data rate reduction can be achieved by concatenating all pre-analyzed events (local & regional) collected from a single GBT link, and cutting them into multiple data blocks of 256-bit. This limits the number of blocks stored in the memory. Hence, reduce the packet size. The sequence in which these events are concatenated is completely arbitrary and depends on their arrival time. Figure 3.5 depicts the data flow of the serialized data format and provides a sense of what can be expected if implemented.

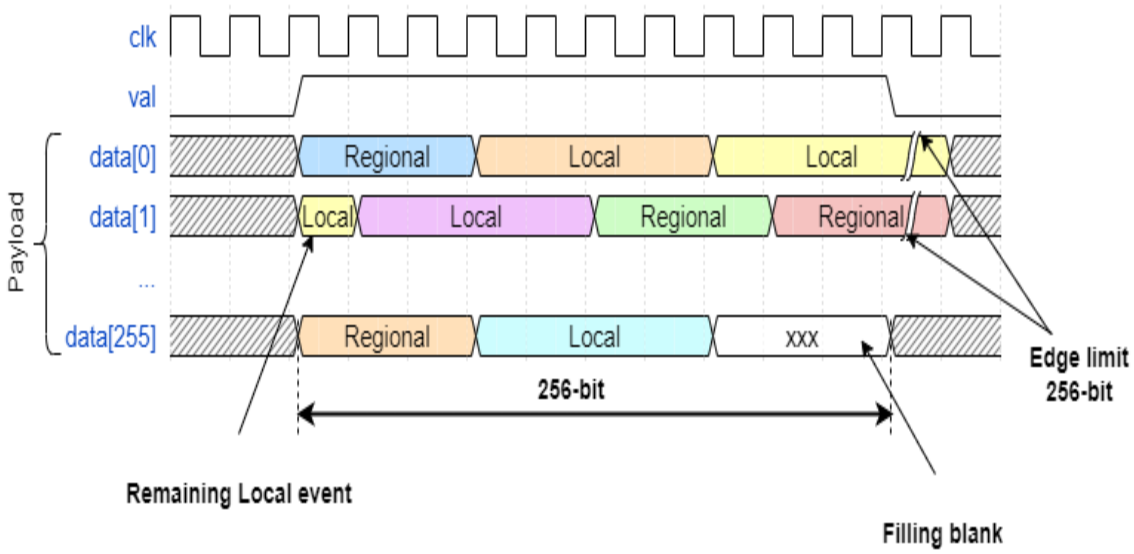

Figure 3.5: An example of the serialized data block format designed for this work

Notwithstanding the fact that both options significantly reduce the data rate, they each offer advantages and disadvantages. The serialized data format will be slower as it will require 32 clock cycles (8-bit x 32 = 256-bit) to fill up a single data block. It will be less efficient in terms of readability due to certain events being chopped if the data block reaches the maximum limit of 256-bit. However, it will be more compact with a lower data rate. The fixed data format, contrastingly, will only take a few clock cycles to fill up a data block, it will provide greater readability, and will facilitate the debugging process. The fixed data format, nonetheless, will occasionally duplicate the header information and include a Reserved field, which must be filled with zeros if an extra local event cannot be accommodated. This contradicts the goal of this research, which is to only transmit valuable information.

Following a comprehensive analysis, it was determined that the serialized data format would be implemented in the MID user logic component. The O^2 software algorithm will reconstruct the events by analyzing each byte of the data blocks collected during each Time Frame.

3.5. Summary

In this chapter, the architecture of the CRU firmware is described and the different CRU readout mode configurations are explained. The main interfaces and their functionalities are also discussed and the communication protocols are elaborated. Finally, the MID user logic component is introduced, and its design requirement and specifications are described. The next chapter will dive into the design and implementation of the user logic.

CHAPTER 4

Design and implementation of the user logic

This chapter delves into the design and implementation of the user logic component. It must be emphasized that the study aims to deliver a functional user logic prototype. Thus, this user logic firmware was designed to process data from a single MID regional crate composed of two GBT links.

4.1. Architectural design

The user logic is designed in a sequential manner, with all processing occurring one after another from an input-output perspective. The objective of this approach is to facilitate error tracking. The user logic design consists of three main segments, each of which is linked to a specific interface of the CRU firmware (see Figure 3.1). A representation of the user logic block diagram and its interfaces is represented in Figure 4.1. Starting from the top is the TTC segment (grey), which receives data from the timing and trigger system through the TTC interface. Next is the GBT segment (blue), which receives data from the readout electronics via the GBT wrappers, analyses them then combines them with the RDH extracted from timing and trigger information before transmitting them to the O² system via the datapath wrappers. The GBT segment is the only part of the design that can be duplicated through parameterization. Hence, enabling the possibility to process multiple GBT links, allowing for improvement and adaption to diverse testing scenarios. The last segment is the Avalon (orange), which provides configuration and monitoring through the PCIe interface.

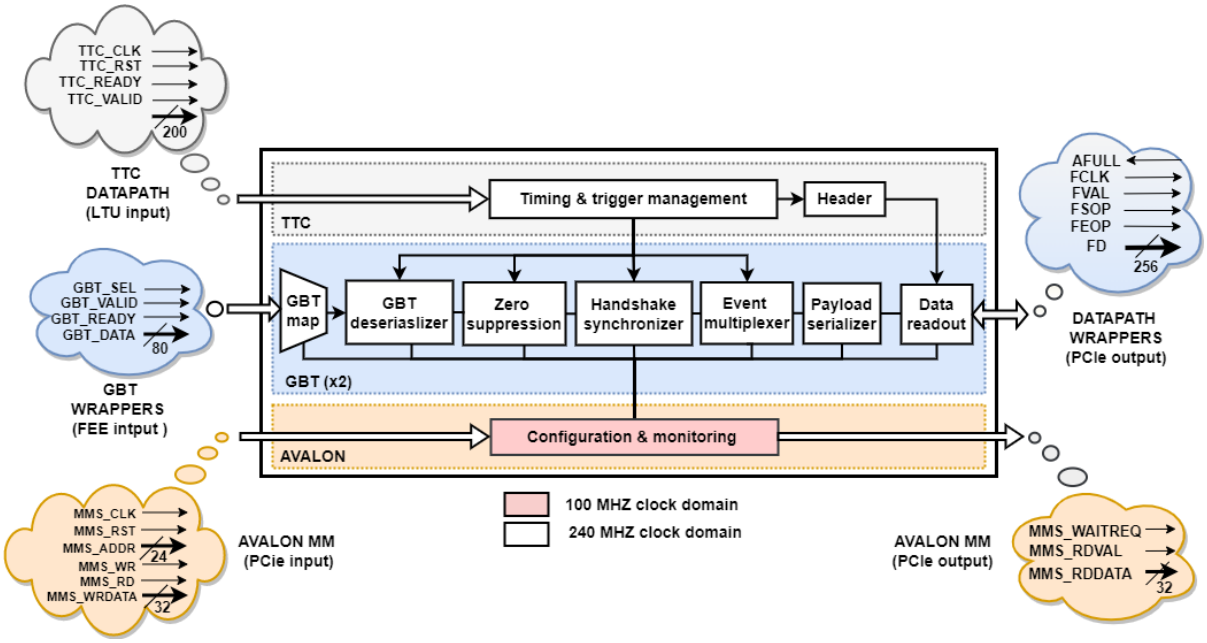

Figure 4.1: Structure of the user logic design showing the three main segments as well as the port specifications and data flow of each segment. This diagram was created for this work

4.2. Implementation

This section describes in detail the functionality of the various submodules implemented in the user logic design and is organized as follows. Clocks and resets are extremely essential in digital electronics, and they receive special attention in subsection 4.2.1. The timing and trigger management, as well as header submodules, are detailed in subsections 4.2.2 and 4.2.3. The GBT mapping and deserializer are discussed in subsections 4.2.4 and 4.2.5, respectively, whereas the synchronization process is explained in subsections 4.2.6. The event multiplexer, payload serializer, and data readout submodules are discussed in the following subsections. This section concludes with a brief explanation of the configuration and monitoring submodule.

4.2.1. Clock and reset management

The user logic requires several clock signals to extract, synchronize and monitor the e-links input data transmitted by the GBT links. Since the intention is to operate synchronously with the host firmware, the user logic takes full advantage of the already available clock signals provided by the CRU clock tree (see Figure 2.8). The faster clock (240 MHz) is utilized for data processing, while the slower clock (100 MHz) is used for signal configuration and data monitoring.

To ensure that the system runs efficiently, the user logic firmware must be set to a known state at the beginning of each acquisition. The reset management incorporated in the design can either be provided externally through slow control via the Avalon MM register or generated internally using the trigger information from the LTU as a catalyst. The external and internal reset signals are referred to as hard and soft resets, respectively.

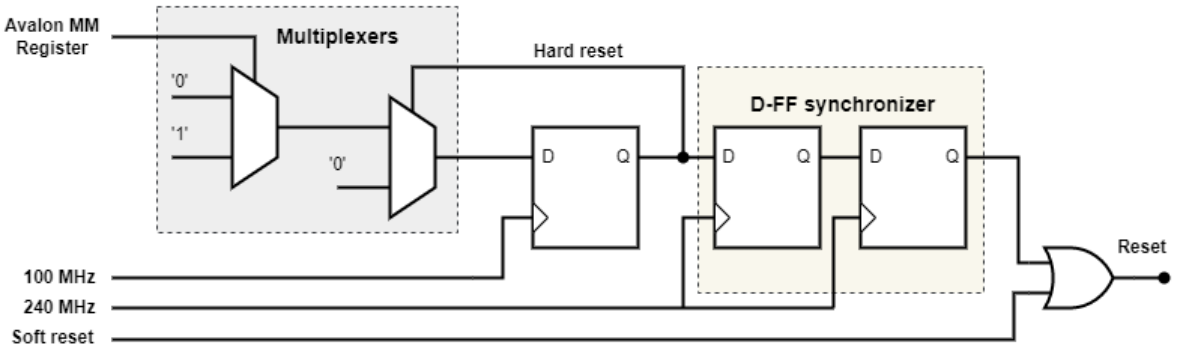

Figure 4.2: Reset management circuitry designed for this work

As illustrated in Figure 4.1, the Avalon register dedicated to externally reset the user logic is attached to a multi-stage multiplexer, then stored into a Delay Flip-Flop (D-FF) register. This combination of circuits prevents the reset management to generate a hard reset signal for no more than a single clock cycle (100 MHz). Furthermore, to address the issue of clock domain crossing, and to acknowledge the acceptance of the slow control instruction before spreading it across the circuit, the hard reset signal is attached to a two-stage D-FF synchronizer. Finally, using an OR-Gate logic, the hard and soft reset signals are computed into a single synchronous reset signal, which is implemented throughout the design. This reduces the amount of work required in the Quartus Timing Analyzer tool as the majority of the work is already done by the CRU firmware developers.

4.2.2. Timing and trigger management

The main task of the timing and trigger management is to administrate the data acquisition. It receives the trigger information from the LTU via the CRU firmware and uses this information to generate signal pulses to activate other parts of the design. It is made up of three components: Decoder, Mode selector, and a Pulser.

To track changes more effectively, the decoder examines the data transmitted by the LTU and categorizes them into three groups (trigger, Orbit, and bunch crossing). These data are temporarily stored in registers before being transmitted to the Header module for RDH generation. The trigger information extracted by the decoder contains trigger signals, which indicates whether the subdetector is running in continuous or triggered readout mode. These signals are used by the mode selector to switch back and forth between different modes of operation. The SOC and EOC signals mark the start and end of the continuous mode, whereas the SOT and EOT indicate the start and end of the triggered mode. Finally, once the readout moded is selected, the Pulser generates an internal soft reset pulse followed by other customed pulse signals used to activate different parts of the design in a pipeline manner, with each part being activated one after the other.

4.2.3. Header

The header submodule monitors, stores, and updates the necessary information required to populate the RDH fields, in particular, triggers, bunch crossing ID, and the Orbit ID (refer to Table 3.1 for detail about the RDH fields). It relies on a three-stage finite state machine to

achieve these tasks. The timing and trigger information vary at regular intervals of 40 MHz. To keep track of the changes, the header closely monitors the SOx, EOx, and HeartBeat pulses transmitted by the timing and trigger management, and stores the updated information into a dedicated FIFO. This updated information is subsequently merged into the RDH fields upon request. The header and the timing and trigger management submodules rely heavily on the TTC information to operate properly. For this specific reason, they can only be initialized using a hard reset.

4.2.4. GBT mapping

As described in the previous Chapter, the user logic has access to 24 incoming GBT links. However, for this study, only 2 GBT link inputs are required. To provide flexibility in the user logic component, a GBT mapping submodule is implemented to select as input 1 out of 12 GBT links. This is done using a multiplexer and a 4-bit select signal, which can be configured externally via Avalon MM. The output of the multiplexer is then attached to a register for synchronization. Figure 4.3 shows the circuit diagram of a single GBT mapping submodule.

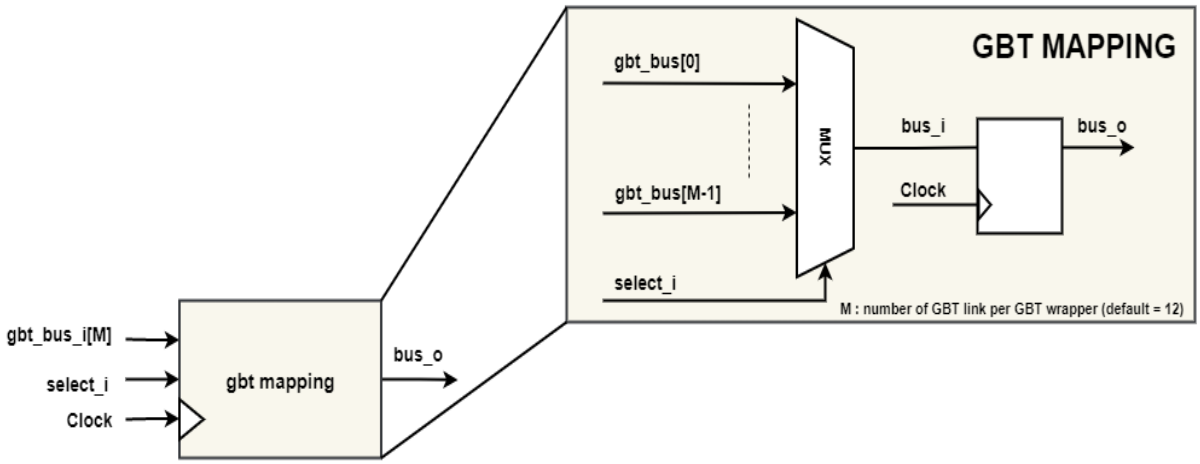

Figure 4.3: GBT mapping circuitry diagram designed for this work

For better coverage, two GBT mapping submodules are implemented in the user logic, each of which reads 1 out of 12 GBT link inputs and both combined can be configured to select 2 out of the 24 GBT links available.

4.2.5. GBT deserializer

The GBT deserializer is used in conjunction with the GBT mapping submodule. The MID readout electronics data are presented to the user logic in the form of GBT buses, each of which consists of Valid and Enable flags, as well as an 80-bit raw data bus. The approach taken is to analyze each local and regional e-links separately. As a result, the GBT deserializer converts the 80-bit data stream into 10 independent parallel blocks of 8-bits (byte) and forwards them along with a copy of the valid and enable flags to the zero suppression submodules for data identification. Figure 4.4 shows how the GBT data stream is deserialized and distributed among 10 independent zero suppression submodules.

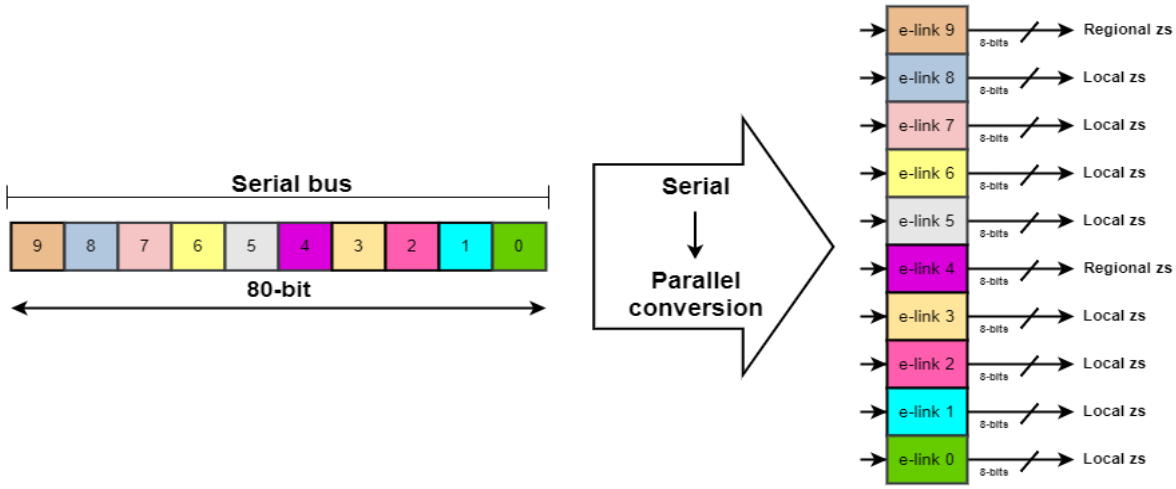

Figure 4.4: The schematic diagram of the GBT deserializer designed for this work

4.2.6. Zero suppression

The primary task of zero suppression is to suppress non-valuable information by appropriately identifying events transmitted by a specific e-link channel. As already mentioned in the previous chapter, an event lasts from the moment the start bit and card type are identified until a programmable number of clock cycles (referred to as an event frame window) are completed. Depending on where the data originated, the event frame window can range from 5 up to 21 clock cycles.

The zero suppression is based on a finite state machine algorithm, that is implemented in a pipelined manner, and uses a rule check and a counter to transit from one state to another. The rules associated with the finite state machine evaluate every byte fragment of the events as they arrive. Each rule checks whether or not the byte fragment matches the rule's criteria previously presented in Table 2.2 and Table 2.3. A counter is used to keep track of the number of clock cycles required to successfully identify a complete event. This algorithm consists of six states, which are used to validate the coherence of various byte fragments transferred across the local/regional e-link. Figure 4.5 shows a representation of the finite state machine diagram implemented in the zero suppression finite state machine.

The state machine register is initialized to an idle state, every time the reset signal is asserted. It remains in an idle state until the GBT and DAQ signals are enabled, before proceeding on to the status state.

The status state is used to determine the correct start bit and the card type of the given e-link event. After successfully identifying these two parameters, the counter is incremented, the 1st byte of the event is stored and the next state is asserted. In case of failure to identify the relevant parameters, the state register remains in the status state and will only return to idle once the GBT and DAQ signals are disabled.

The trigger state checks the second byte of the event data. It ensures that all incoming trigger bytes do not violate any of the fundamental trigger rules. Among these rules are the following:

- SOx and EOx trigger bits cannot be activated simultaneously;
- SOx and Eox trigger bits must be activated in conjunction with an Orbit trigger;
- The Calibration trigger bit is always followed by a self-triggered event;

Once all of the above requirements are met, the second byte is stored and the counter is incremented.

The bunch crossing information is composed of the third and fourth-byte fragments of the event. Hence, the bunch counter state collects these byte fragments in two stages. The four most significant bits of the third-byte fragment must always be zero; anything other than that should be treated as noise. As a result, the event will be marked as corrupted and the state register will return to the status state. If no errors are identified, the bunch crossing information is stored, the counter is double-incremented, and the next state is asserted.

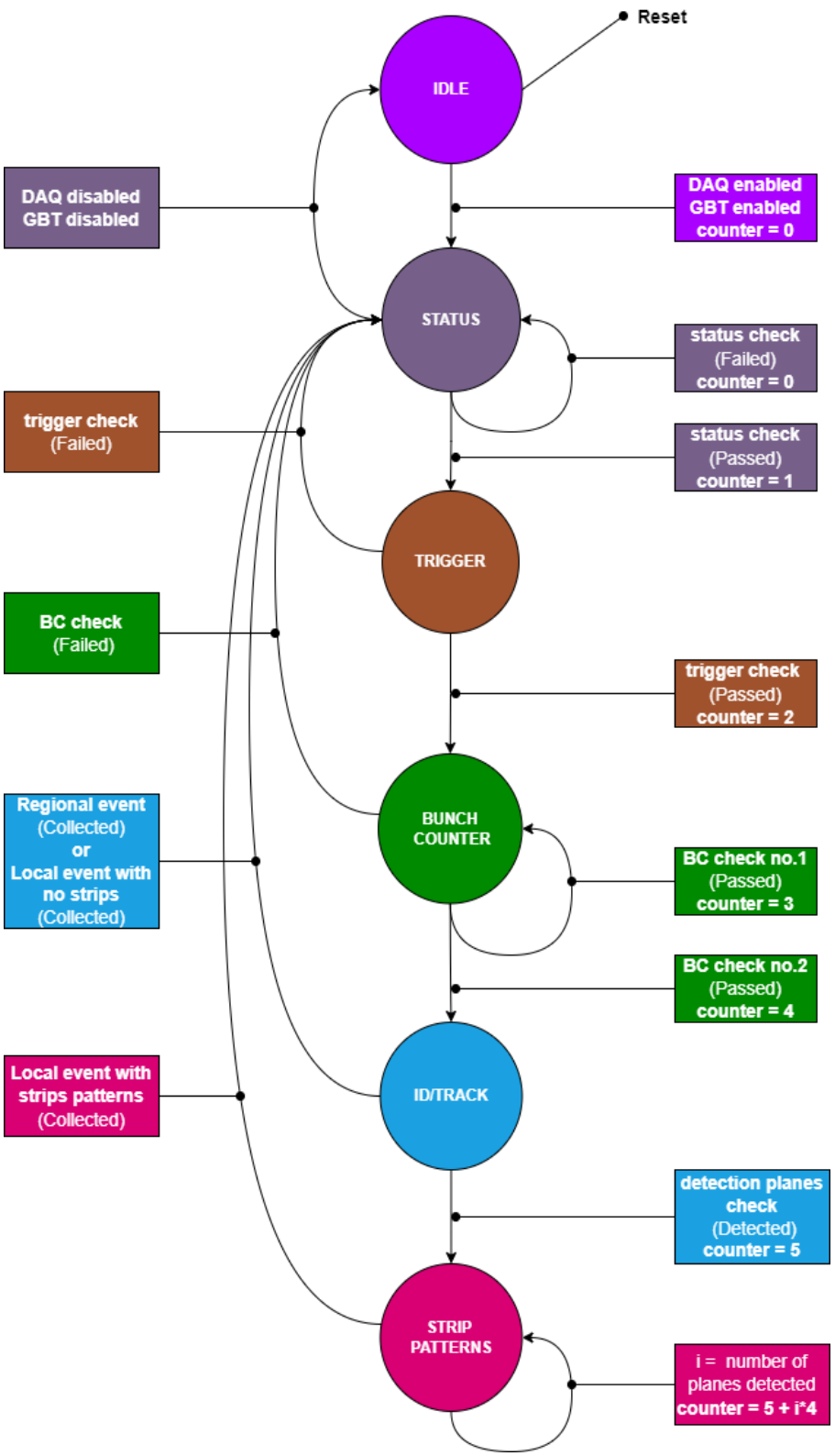

Figure 4.5: Finite state machine diagram implemented in the zero suppression. Each bubble represents a state of the finite state machine designed for this work

The ID/Track state is very crucial as it determines the identity given to the card as well as the length of the event. At this point, the card type of the event is already known (regional or local). Based on this information, the size of the event can be computed. For all regional and local events with no detected strip patterns, the event size does not exceed five bytes. In order to be ready for the next upcoming event, the state register returns to the status state and the event collected is forwarded to the next phase of the data acquisition. For all local events with detected strip patterns, the event size can range from 9 to 21 bytes. As a result, the fifth byte is stored, the counter is incremented and the next state is asserted.

The strip patterns state has no rule checks and relies completely on the event size information computed during the preceding state. For each detection plane fired, the event size is extended by four bytes, so is the counter. The state register remains in the current state until the event is entirely collected, at which point it returns to the status state. Similar to the previous state, the event collected is forwarded to the next phase of the data acquisition.

4.2.7. Handshake synchronizer

The handshake synchronizer is implemented throughout the user logic to synchronize all e-links and subsequently all GBT links. It consists of a pair of FIFOs working in parallel and managed by a slave controller. The synchronization is achieved by carefully monitoring the trigger bits found in the event data while compensating for the latency caused by various optical and e-links during the data acquisition. It is done in three phases and they are described as follows.

The first phase of the synchronization focuses on determining whether or not the e-link pathway attached to the synchronizer is active or inactive. By default, all e-link pathways are presumed to be inactive until proven otherwise by the slave controller. This prevents the synchronizer from transmitting data outside the acquisition window. The sender FIFO stores and holds the event data transmitted by the zero suppression submodule until it receives a request signal from the slave controller. Following acknowledgment of the request and release of data, the slave controller analyses it in search of a potential SOx trigger. Any event data pulled from the sender FIFO is discarded until a valid SOx trigger is found in the event data, which enables the Active signal and marks the beginning of the acquisition window.

The second phase is devoted to data filtering. All events retrieved from the sender FIFO within the acquisition window are handled by a filtration process before being written to the receiver FIFO, except for the Orbit and EOx events, which do not require filtering and are instead moved to the third phase of the synchronization. The filtration process operates in two different ways. In continuous readout mode, all event data are automatically transferred to the receiver FIFO. However, in triggered mode, as previously stated in subsection 3.4.1, only physics events with a similar bunch crossing identification as the one found in the TTC information are permitted to be written to the receiver FIFO, the remaining events are discarded by the filtration process.

The SOx, Orbit, and EOx events are set with the highest priority in the readout electronics firmware. This implies that regardless of the state of the memories, these events are always transmitted to the user logic component. Hence, by relying on these occurrences, complete synchronization is attained. However, as previously stated in subsection 3.4.2.3, mistakes do occur. Any active e-links may unexpectedly cease to transmit data within the acquisition window and jeopardize the entire synchronization. To overcome such events, the master controller uses the HeartBeat trigger found in the TTC information as a backup signal to keep the synchronization cycle running while ignoring data from all defective e-links. Figure 4.6 shows a representation of the handshake synchronizer designed for a single e-link.

Due to the difference in buffer sizes and specifications between the regional and local event data, two types of handshake synchronizers have been implemented for each card type. The local handshake synchronizer is designed as described above. On the other hand, the regional handshake synchronizer requires additional features such as the computation of a custom ID in its event data format. This is achieved between the second and third phases of the synchronization.

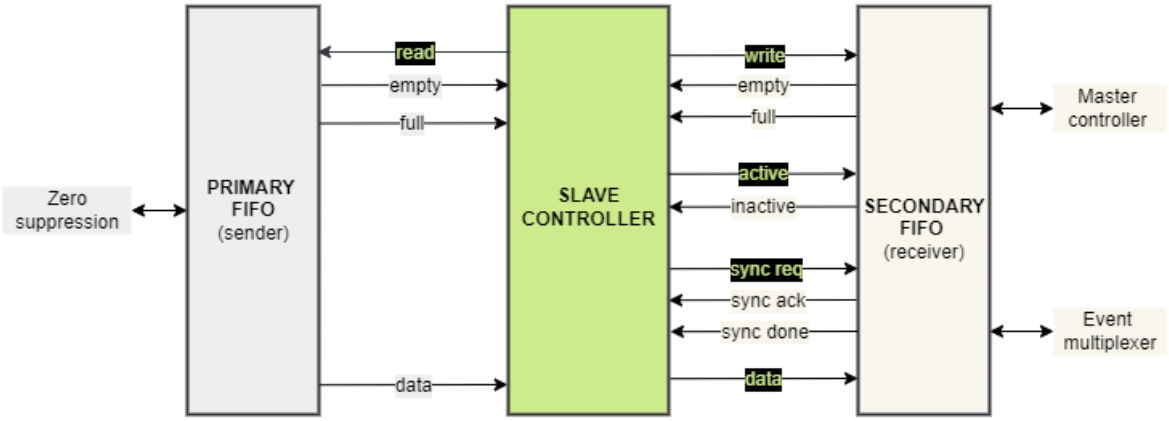

Figure 4.6: Illustration of a single e-link handshake synchronizer created for this work. The key components, as well as the necessary signals required to perform the synchronization, are shown

4.2.8. Event multiplexer

The concept behind the event multiplexer is to decode and extract event data from all handshake synchronizers belonging to the same GBT link and transmit them in the form of a byte segment to the next layer of the acquisition for packetization. However, despite the zero suppression, the data rates remain too high. The rate at which data are written to the receiver FIFO in the handshake synchronizer is faster than the rate at which data are read. As a result, the receiver FIFO fills up faster than expected and starts dropping data. This is solved by implementing two event multiplexers instead of one to deal with a single GBT link. Each of them collects event data from 5 different handshake synchronizers (1 regional + 4 local) and operates independently of one another. A single event multiplexer is depicted in Figure 4.7. The diagram shows how the event multiplexer interacts with various handshake synchronizers via a Priority Encoder. A zoom-in on the event multiplexer is shown on the right side of the diagram. It highlights the key components of the submodule, which includes a finite state machine, multiplexer, and decoder.

A Priority Encoder algorithm is implemented to assign a priority level to a handshake synchronizer before transmitting data. Its output is equivalent to the receiver buffer status of the handshake synchronizer with the highest priority. Due to the difference in length of their respective event data, collecting a regional event requires less time than collecting a local event. Therefore, the Priority Encoder is designed in such a way that the regional synchronizer always takes precedence over the local synchronizers.

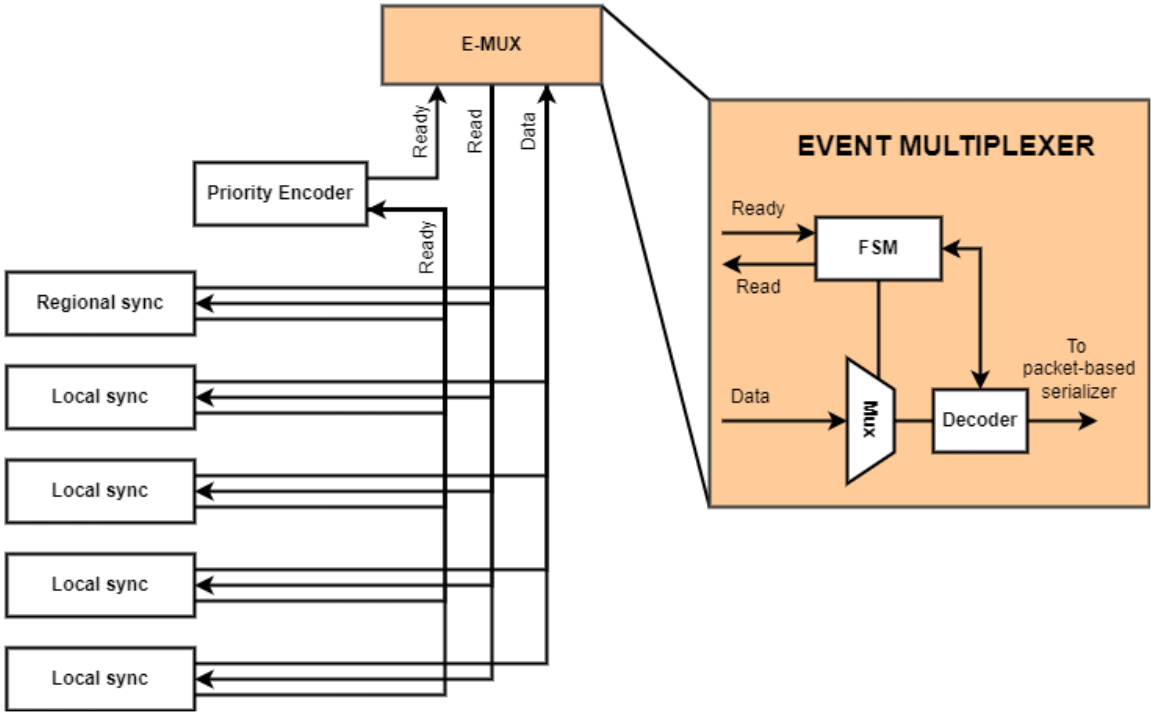

Figure 4.7: Implementation of the e-link multiplexer from a conceptual standpoint. This diagram was designed for this work

A finite state machine is designed to control the multiplexer and the output logic as well as to ensure that data are delivered only after the preceding event data has been properly decoded. It receives a Ready signal from the Priority Encoder, indicating that an event data is ready to be read out, and then responds to the highest priority synchronizer with a Read signal, instructing it to move data to the decoder through a multiplexer.

The decoder operates similarly to the zero suppression submodule with fewer checks done. It receives a signal from the finite state machine indicating the card type of the selected synchronizer which is currently transmitting data. Based on this information, the number of clock cycles required to slice the event data into byte fragments can be computed. The new regional format requires 6 clock cycles, whereas the new local event is dependent on the plane tracklet located in the fifth byte of the event. Furthermore, as shown in Figure 3.3, the new local event format includes a new crate identification block, that needs to be incorporated in the event data before packetization. Since the regional synchronizer always has priority over the local synchronizers, the crate identification found in its event is distributed among all local synchronizers.

4.2.9. Payload serializer

The payload serializer is designed to gather data byte fragments from two event multiplexers (upper and lower parts of the GBT) and organize them in compact data blocks of 256-bit before passing them to the data readout submodule. The structure diagram of the payload serializer designed for this work is illustrated in Figure 4.8.

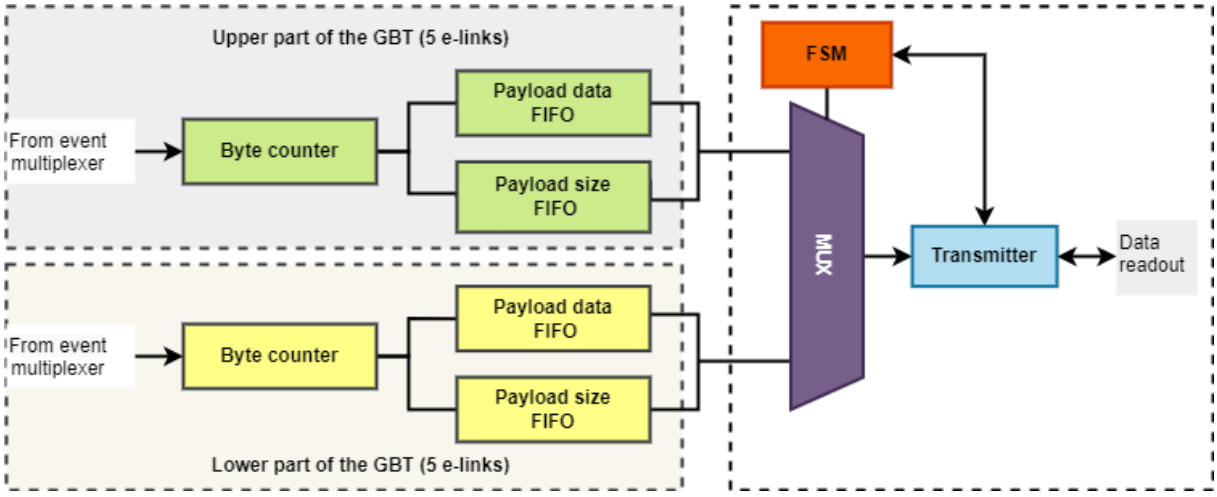

Figure 4.8: Structure diagram of the payload serializer. The main components of this submodule can be observed as well as the data flow

In order to facilitate the decoding of the data at the O^2 level, the payload serializer uses a preselected data block format that was agreed upon before implementation of the user logic (refer to Figure 3.5 for more details about the format). The data bytes collected from each event multiplexer are saved in a register until they are ready to be moved to the “Payload data FIFO”. The payload serializer relies on a byte counter to keep count of the number of incoming data bytes stored and waits until it reaches a maximum limit of 32 bytes before passing it onto the “Payload Data FIFO”. It should be noted that the total number of data blocks collected during a HeartBeat Frame period is referred to as payload size.

One of the design requirements stipulated in subsection 3.4.1 is to prevent data blocks from two separate HeartBeat Frames from being mixed during the HeartBeat Frame transition. This is accomplished by computing and storing in an additional FIFO, namely “Payload size FIFO”, the total amount of data blocks acquired for each HeartBeat Frame. This enables the payload serializer to extract payload data based on their sizes.

Finally, the payload serializer includes a finite state machine that operates in tandem with a multiplexer and a transmitter. The role of the finite state machine is simple. It receives a transmission request signal from the data readout, extracts the payload size and the data from their respective FIFOs before transmitting them. The transmission is achieved in sequences through the multiplexer. First, it transmits data from the upper part of the GBT then only after it transmits data from the lower part.

4.2.10. Data readout

As mentioned in the preceding chapter, the user logic component must adhere to the CRU readout control protocol and provide data to the datapath wrappers in the form of packets (refer to Figure 3.2 to see the protocol). This is achieved using the data readout submodule designed to work in correlation with the header and payload serializer submodules.

The data readout is very straightforward, it does not check the consistency of the data, nor has knowledge of the format or content of the data it receives. Its sole purpose is to combine the RDH and payload to form packets and route them to the datapath wrapper. Like many other submodules implemented in the user logic, it is based on a finite state machine, which is controlled by the value of its state signal. First, it receives a header-ready signal from a header submodule indicating that a new HeartBeat trigger has occurred and the RDH content is ready to be updated, in return it sends back a signal to tell the header submodule to advance the data and proceed with the update. Since the payload serializer submodule is also in sync with the HeartBeat trigger, permission to transmit data to the datapath wrapper is often granted to the payload serializer a few clock cycles after the RDH have been updated and transmitted. The stop bit, page counter, and other information required in the RDH format are computed based on the size of the payload and the number of data blocks transmitted. A data block counter is implemented to make sure that the packet does not exceed 8 KB.

It might happen that the packet serializer has no payload to send, i.e., no valuable events were detected during the HeartBeat Frame, in this case, the payload size will be zero, and only the RDH will be transmitted. This is known as the sync packet, it has a fixed size and fixed format, but the information in it may vary.

There was no particular requirement set as to which of the two datapath wrappers the user logic should utilize; this decision was left to the subdetector. This research seeks to process data from two GBT links, the packets created from these GBT links data can either be sent to the same datapath wrapper or can be distributed among the two datapath wrappers. This decision is taken based on the configuration set in the GBT mapping submodule. Packets created from the first 12 GBT links data (0-11) can be transmitted to datapath wrapper 0, while the remaining (12-23) can be transmitted to the datapath wrapper 1. Figure 4.9 shows a scenario in which packets from two GBT links are moved from the user logic to different datapath wrappers via the data readout channels. The rectangles represent packets that have been transmitted throughout time. Each packet consists of the RDH and payload. The green colour indicates that the payload originated from GBT link #12, whereas the red originated from GBT link #0.

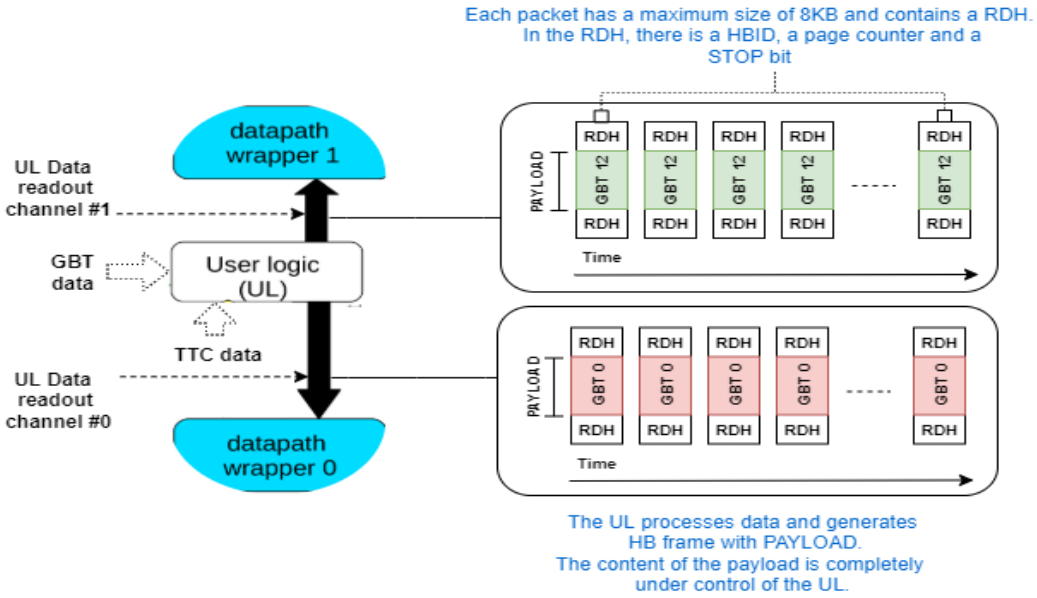

Figure 4.9: Illustration of the user logic component showing how its data output is moved to the datapath wrapper channels. This diagram was created for this work

4.2.11. Configuration and monitoring

As mentioned throughout this chapter, several signals can or must be configured externally to improve the interoperability of the user logic design. This is accomplished through the use of the Avalon MM interface, which is already present in the CRU firmware. The Avalon MM interface is attached to the PCIe and serves as a link between the FLP software and the

CRU firmware. It enables the direct reading and writing of specialized registers, making the configuration and monitoring of the user logic design a lot easier. Thus, signals from numerous submodules are routed out to the Avalon MM registers to record the internal circuitry of the user logic once it has been integrated into the FPGA. This is useful for debugging purposes and managing errors.

4.3. Error handling

Monitoring the most critical submodules is an effective way to keep track of the stability of the user logic component. However, the major challenge is how to proceed once an error has been detected. The typical approach is to manually reset the user logic firmware. This is not always convenient, especially in the middle of an acquisition when data collection is very crucial. Hence, it is necessary to implement an algorithm to automatically handle some errors without interfering with the DAQ. During the period allocated to this study, only a few errors have been identified, and as the design expands more errors will eventually emerge. The errors identified thus far can be divided into two categories based on their severity levels: configuration and functional errors.

Configuration errors occur when configuration registers are not properly configured and have an impact on the functionality of the user logic component. These errors will linger until a hard reset or reconfiguration is performed, and they can be rather serious at times. However, depending on which parameters were impacted, the user logic can still operate successfully in the midst of these errors. To prevent these errors from happening, the parameters inserted are fed back into their initial registers, where they can be visualized and adjusted when necessary.

Functional errors are the most severe since they can prevent the user logic from functioning properly and, in extreme cases, cause the system to crash. The major cause of these errors is the transmission delay between finite state machines and FIFOs. To avoid such occurrences, a two-stage D-FF synchronizer is attached to the input and output of all FIFOs throughout the design. Furthermore, the soft reset is designed in a specific way to ensure that all finite state machines are always in an idle state at the beginning of each acquisition. In contrast to the hard reset, which resets everything, the soft reset simply initializes a subset of the design, allowing the user logic to automatically reset itself without having to reconfigure the Avalon MM registers.

CHAPTER 5

Verification, testing, and results

The architecture outlined in the previous chapter has been tested to ensure that, the design specifications and requirements of the user logic are met. This chapter elaborates on the methods used to perform the verification and testing of the user logic firmware before concluding with a detailed discussion of the readability and validity of the data.

The chapter starts with a discussion on the functional verification of the user logic using software simulations. Methodologies used for reducing the probability of bugs occurring in the design are also covered. The chapter then goes on to describe the test bench that was built for the hardware testing. The focus is then shifted to hardware validation and testing, with a primary focus on the tests that have a significant impact on the user logic component.

As per the CRU expert recommendation, any subdetector wishing to implement their custom user logic should utilize the suggested tools for synthesizing and simulating their code. For this reason, all HDL files produced for this project were written, compiled, and simulated in VHDL using Intel Quartus Prime Pro and ModelSim.

5.1. Functional verification

Functional verification is the most important part of design and development. It ensures that the system operates as it should. However, even though particular debugging capabilities have been implemented, there are no straightforward ways to determine the cause of any unexpected errors once the firmware is integrated into the FPGA. As a result, complex test benches are written to evaluate and validate various parts of the user logic component before the insertion of the code in the FPGA. The main objective of functional verification is to facilitate the testing process by providing dummy data to the Device Under Test (DUT) and automatically checking the validity of its response. The functional verification implemented in this design is divided into two sections: the first section is the module-based verification, which focuses on validating each module of the user logic individually, and the second part is the system-based verification, which combines various systems of the MID readout chain into a single test bench for a more complex verification.

5.1.1. Module-based verification

By creating HDL test bench files for the most important submodules, the functionalities of these submodules can be thoroughly evaluated using ModelSim. Several HDL test bench files developed for this study have the same architecture depicted in Figure 5.1, which contains two or three input files providing dummy data that are fed into the stimulus generator included in the entity test bench file and applied to the input ports of the Design Under Test (DUT). An output process is running in parallel with the DUT, it collects the output data from the DUT and writes them to an output file for further analysis.

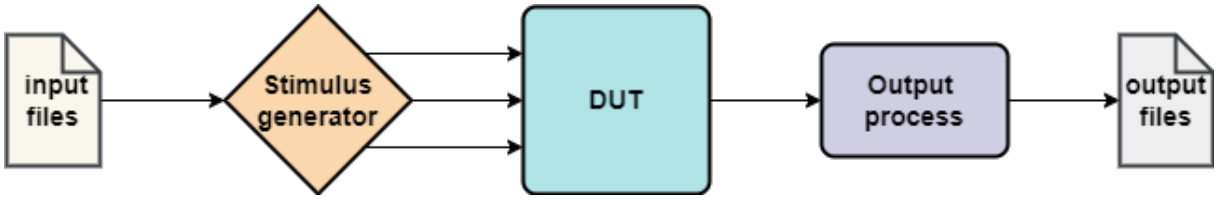

Figure 5.1: Module-based verification test bench model designed for this work

5.1.1.1. Stimulus generator

In order to create an efficient simulation, the stimulus generator implemented in the module-based verification is made up of three primary components, each of which emulates a distinct input interface.

Multi-clock generator

Due to the difference in clock domains from different systems interfacing with the user logic component, a multi-clock generator is implemented in the stimulus to drive the simulation by providing clock signals to the DUT. As illustrated in Figure 5.2, the multi-clock generator replicates the three main clock signals introduced in the CRU clock tree architecture (refer to section 2.5.1 and Figure 2.8).

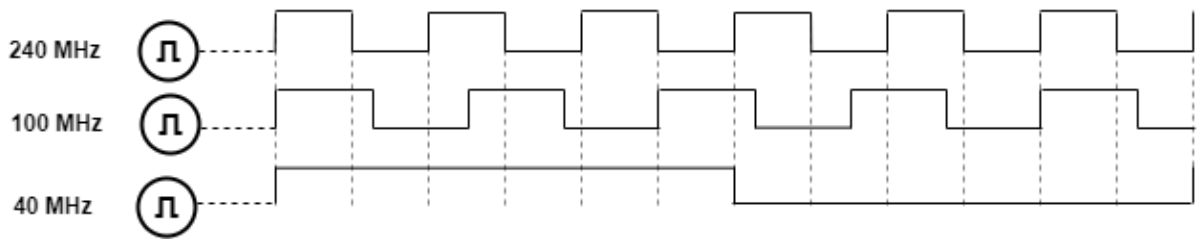

Figure 5.2: multi-clock generator waveforms showing the three main clock signals

TTC data generator

As the name implies, the TTC data generator emulates the TTC interface and feeds dummy timing and trigger information retrieved from the TTC input file to the DUT. The user logic component heavily relies on the trigger information to determine the type of readout mode of operation. As a result, the readout mode used in the simulation can be selected by manually enabling or disabling the SOC or SOT trigger bit located in the TTC input file. However, the continuous readout mode is sufficient to evaluate the functionality of the entire design.

GBT data generator

Similar to the TTC data generator, the GBT data generator emulates the downlink bus line of the GBT wrapper described in section 3.2.1. It extracts the GBT data from the GBT input file and transmits it to the DUT. Although it is not required, the GBT data generator can duplicate the same input information to multiple GBT links, increasing the scalability of the simulation. Nevertheless, a single GBT link is sufficient to evaluate the functionality of the design. The TTC and GBT generators and their waveforms are illustrated in Figure 5.3. The waveform diagrams show how data are taken out from the files and transmitted to the DUT. It is also worth noting that the TTC valid signal always arrives ahead of the GBT valid signal. This is done so that the system may be initialized before processing the GBT data.

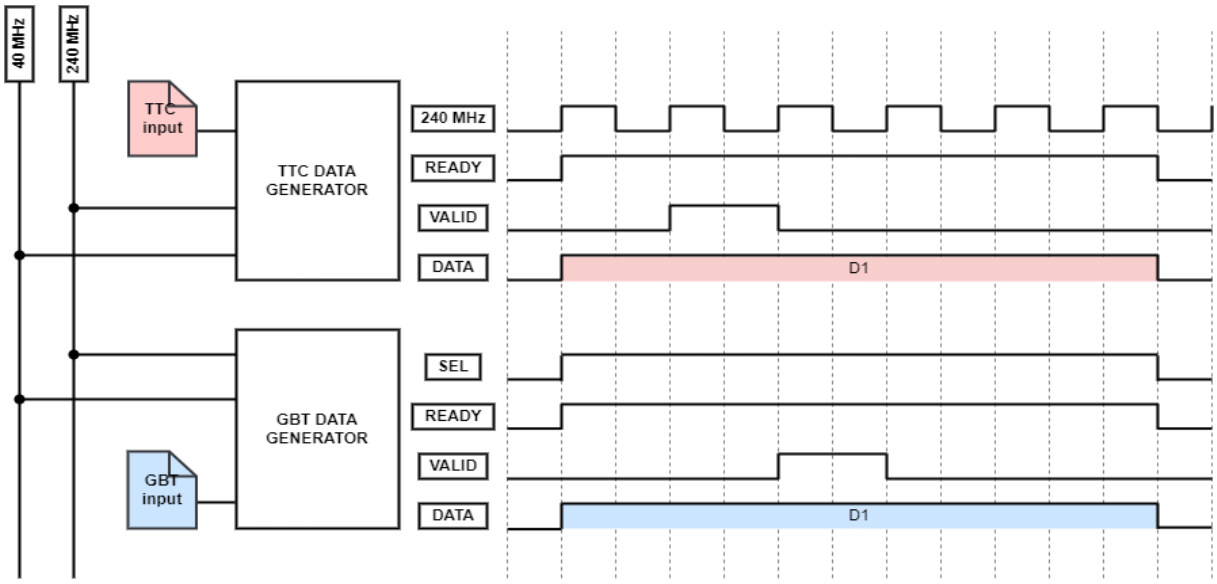

Figure 5.3: TTC and GBT generators, as well as their waveforms. This diagram was made for this work

5.1.1.2. DUT

As mentioned in the previous chapter, the user logic is designed sequentially, with each submodule designed one after the other. The same methodology is applied in the functioning verification. Each of the DUT is tested one after the other starting from the bottom to the top level of the hierarchy. Once a DUT has been validated as an isolated submodule, it is merged with the following DUT, and its output is utilized as input.

5.1.1.3. Output process

The output process uses assertions to detect unlawful transactions in the DUT. Assertions are brief pieces of simulation code included in the VDHL files that report when particular violations occur. The advantage of having assertions is that they report bugs when a submodule is tested alone as well as when it is tested in conjunction with other submodules as part of a larger hierarchy. The bugs detected are categorized into four severity levels (note, warning, error, and failure). All severity levels are reported in the ModelSim command line window as well as in the output files generated during the simulation. The severity failure is the most critical one, it terminates the simulation instantly upon identifying a bug.

5.1.2. System-based verification

The verification of the design from the top level of the hierarchy might be hampered by both, the limited amount of input data available and the amount of simulation time required to test a larger portion of the code. As a result, it is most preferable to test each submodule independently. However, whilst isolated tests seem to be the preferred solution, they might not have the ability to detect all issues related to the inter-communication between submodules in the architecture. Therefore, it is still necessary to validate the design from the top level of the hierarchy.

As shown in Figure 5.4, the functional verification implemented in the user logic includes a system-based verification, which was created to validate the user logic as a single component together with the rest of the readout chain. It combines the CRU firmware modules (provided by the CRU experts), the readout electronic firmware modules (provided by Dr. Christophe Renard), as well as the user logic firmware (the outcome of this research) into a single test bench for a more sophisticated verification.

The advanced stimulus generator, in-cooperation with the readout electronics and CRU firmware, provides the user logic component with all of the data required to execute a complete verification. Data entering and exiting the user logic component are routed to a distribution process, which writes them to distinct text files based on their origin. These files (input and output) are then sent to an algorithm developed by the O² expert that compares them to ensure data consistency and identifies errors. Finally, a report is generated stating the number of errors detected as well as the number of events missing from the user logic output file. The same input data can also be utilized in a standalone simulation of the user logic to pinpoint errors detected and enables rectification.

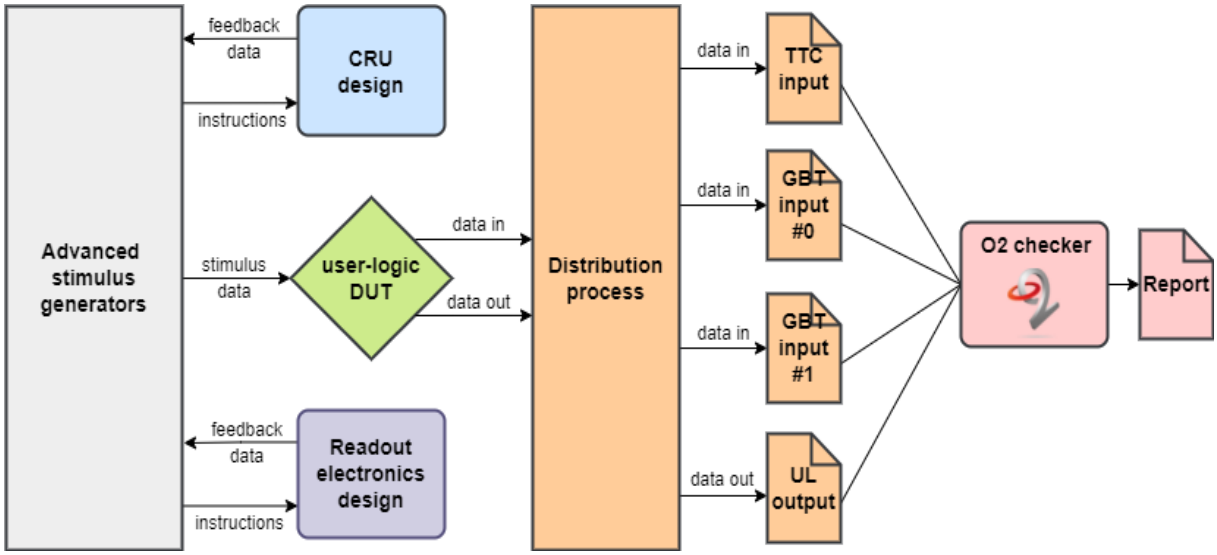

Figure 5.4: System-based verification model designed for this work. The diagram shows how various systems are joined to form an advanced verification

The module and system-based verifications can be replicated by following the instructions mentioned in Appendix B, which also provides the simulation stress test results as well as the necessary files to execute these simulations.

5.2. Hardware tests

The user logic component is an important part of the MID upgrade. It seats at the heart of the CRU firmware and, if not designed properly, it might jeopardize the functioning of the entire MID readout chain. To ensure that such an event does not occur, a test bench has been developed at iThemba LABS to perform extensive tests on the user logic prototype.

This section dives into the architecture and evolution of the hardware test bench as well as the methods used to perform a successful acquisition.

5.2.1. Test bench layout

A small-scale MID readout chain test bench is set up in the laboratory room S64 at iThemba LABS. This test bench has evolved during this Master Degree. The initial setup consisted of the Arria 10 Development Kit that was used as CRU, and a Versatile Link Demo Board (VLDB) (Raul Martin et al., 2017), which was used as temporary readout electronics. The Arria 10 Development Kit is loaded with an older version of the CRU firmware, which is similar to the one found on the final CRU board with much fewer features. The VLDB is a demonstration and development board specially designed to accommodate the GBT protocol. The VLDB board contains a GBTx chip attached to a single GBT optical link and e-links exposed on mini-HDMI ports, a GBT-SCA chip, and custom FeastMP radiation-hard DC-DC converters developed at CERN. It is used to comprehend the basic operation of the CRU and GBT protocol, however, it does not emulate the MID data. Figure 5.5 shows the initial setup of the test bench.

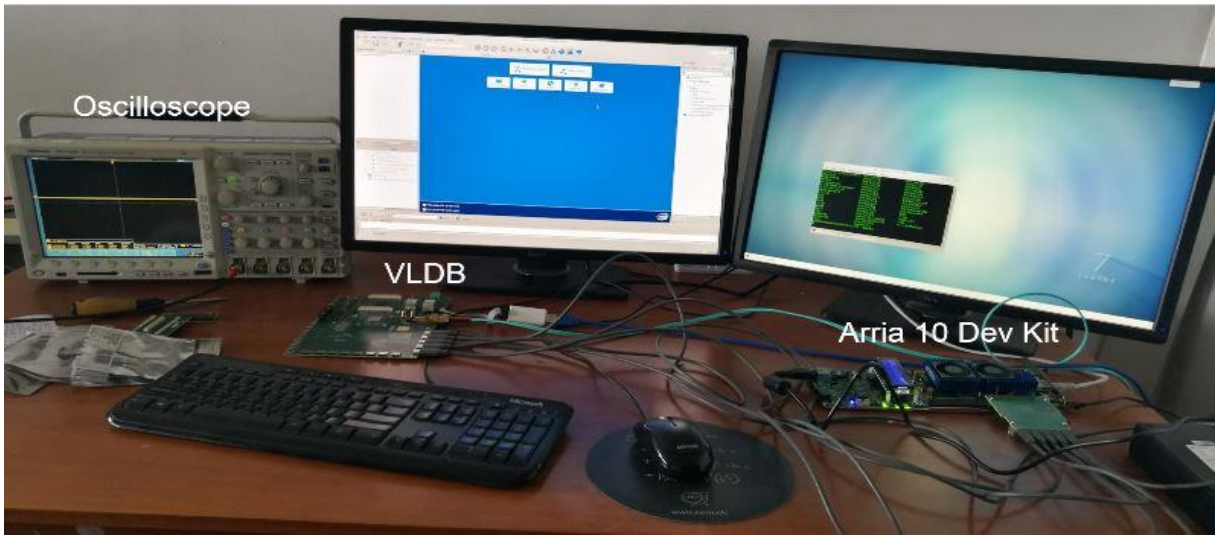

Figure 5.5: Test bench initial setup, it includes the ARRIA 10 FPGA Development Kit, a VLDB, and an oscilloscope

As the project evolves, it became important to test the user logic component on a test bench capable of providing the MID data. The VLDB was shortly replaced with a readout electronics prototype board, namely MID proto (Renard, 2021), capable of emulating the subdetector data. The MID proto is based on two Altera Cyclone V FPGAs, one with seven local event

generators and the other consists of two regionals. The MID proto card, like the VLDB, is equipped with a GBTx chip coupled to a single GBT optical link which serves as a data link between the onboard FPGAs and the CRU. The MID proto was used to test the user logic firmware with a single GBT optical link, before acquiring the CRU board and the FLP server. Each internal generator is attached to an e-link, allowing the proto card to provide realistic events data to the user logic component. Figure 5.6 illustrates the MID proto card.

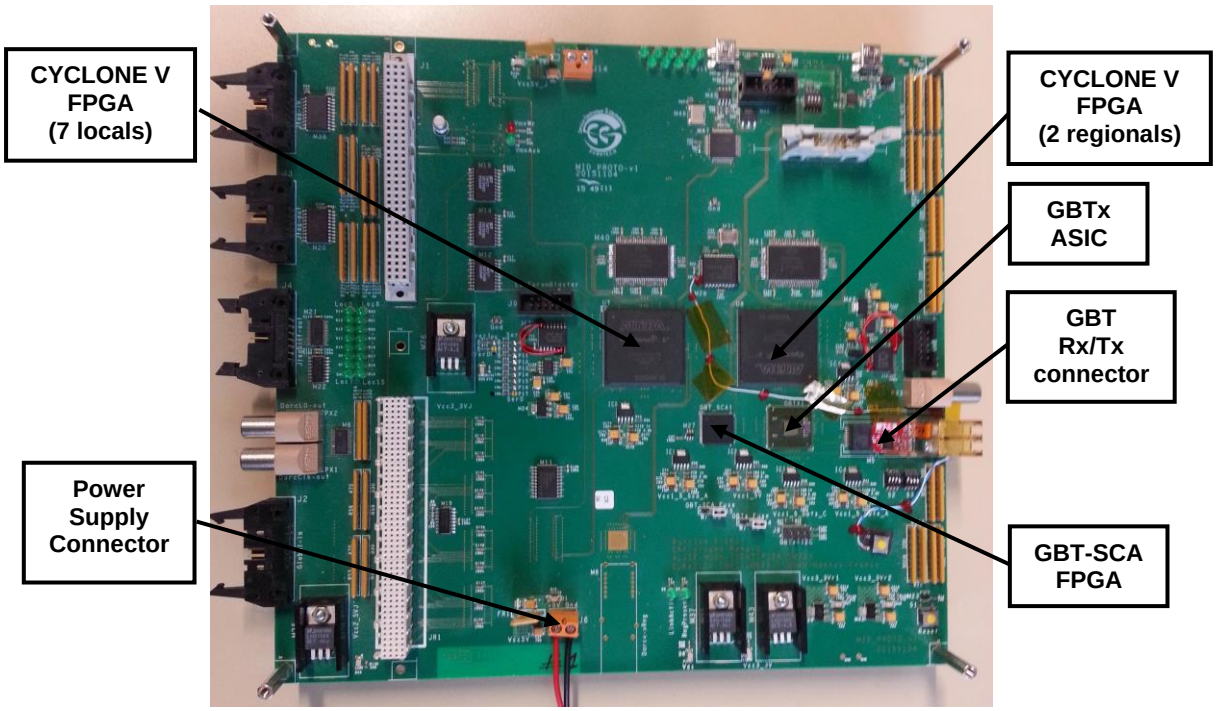

Figure 5.6: MID readout prototype card and its main components (Renard, 2021)

A new test bench was implemented towards the end of this study to extend the capabilities of the user logic component and match the test scenarios performed with readout electronics at Subatech (where the readout electronics cards were designed). The new test bench is a scaled-down replica of the MID readout chain without the RPC detectors. It includes a fully equipped VME crate (1 x FEERIC emulator board, 16 x local, 1 x regional, and 1 x J2 bus boards), a FEERIC board emulator, an LTU, CRU, and FLP.

Figure 5.7 depicts the new test bench setup and illustrates how various components are linked together. The full setup can be observed on the top-left, and the bottom-left is the fully equipped VME crate, excluding the FERRIC emulator board. The local and regional cards are plugged into the crate via the J2 bus card sitting at the back of the crate.

Three cables are exiting the regional card, two of which are optical cables, and connect the regional card to the CRU. The latter is a USB (2.0) cable connected to a Centos PC, which is used to configure and program the local and regional FPGAs. Moving to the top-right is the LTU, which uses an OLT module, which can be used to interact with multiple CRUs via a splitter (refer to section 2.6.4). For this application, only one CRU is needed, the connection between the LTU and the CRU is done via an optical attenuator (15 dB), a single-mode SC to SC optical cable, and an ONU module. The LTU uses an Ethernet cable to interact with the FLP software, which runs on CentOS 7.3, as recommended by CERN experts. Finally, on the bottom-right is the CRU board enclosed in the FLP server. The CRU board is internally attached to the FLP via the PCIe connectors, its FPGA can be programmed using the PCIe interface or via its integrated USB blaster programmer, which connects to the FLP server using a micro USB (2.0) cable.

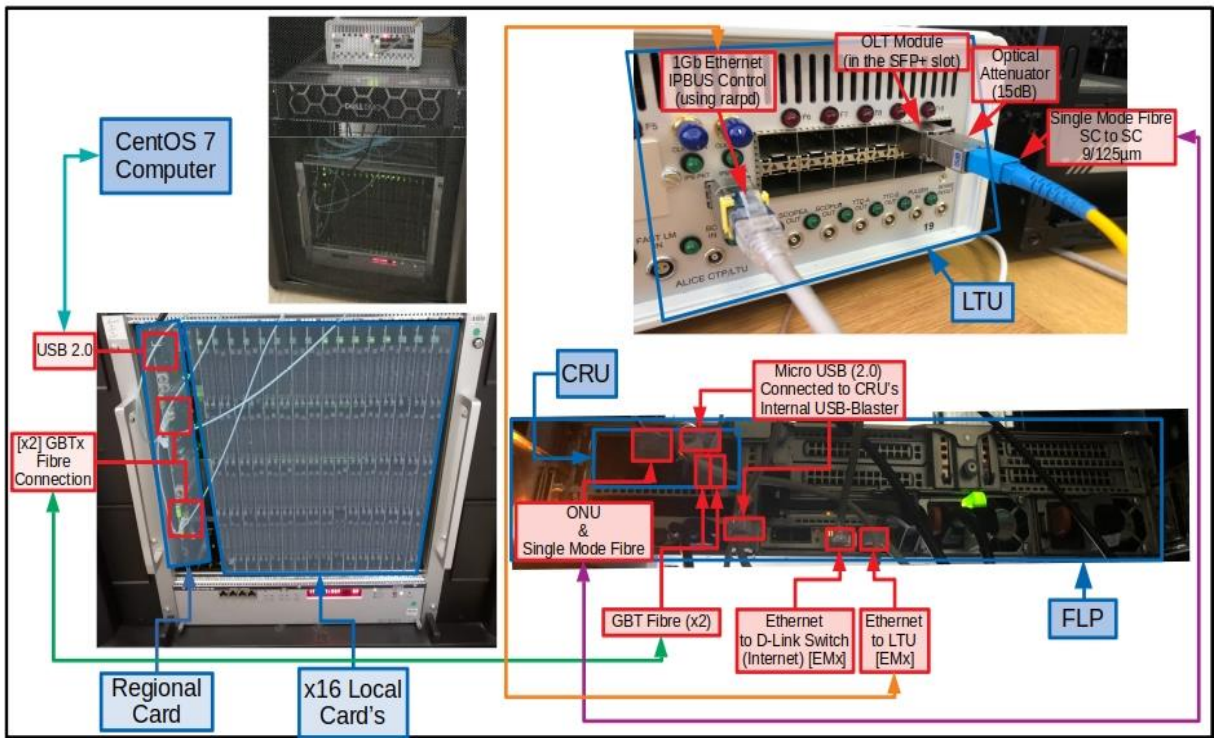

Figure 5.7: New test bench located in the laboratory S64 at iThemba LABS. This diagram was created for this work. It shows the main components of the readout chain, and how they are connected

Figure 5.8 illustrates a fully loaded VME crate with the FEERIC emulator board. On the right, an overview of the VME crate is displayed, and on the left, a zoom-in within the VME crate is presented, with the FEERIC emulator board plugged in.

The FEERIC emulator board was designed by the university of LPC Clermont Ferrand in France (where the front-end electronics were designed). It complements the test bench by emulating the strip patterns data extracted from the RPC detectors. As it can be seen in Figure 5.7, each local card has 4 input connectors that correspond to the 4 chambers of the MID subdetector. The FEERIC emulator board provides strip patterns data (X1Y1, X2Y2, X3Y3, and X4Y4) to each local card via 4 ribbon cables. In total 64 ribbon cables are used.

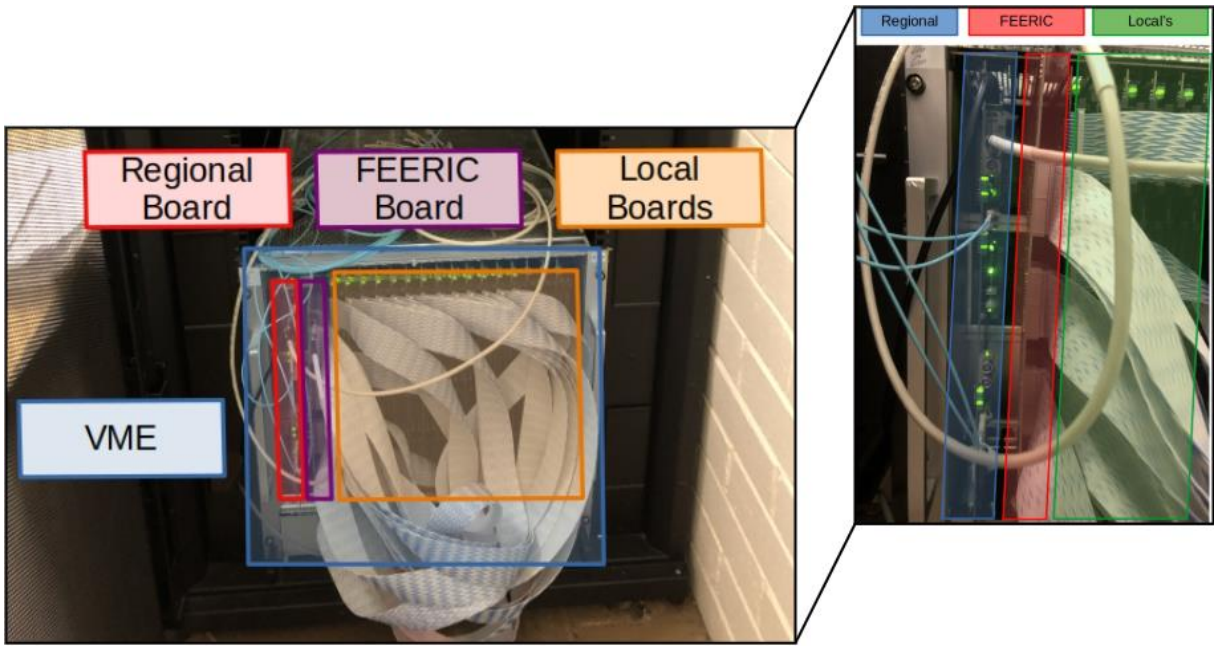

Figure 5.8: Fully loaded VME crate with the FEERIC emulator board. This illustration is created for this work

5.2.2. Insertion and compilation of the user logic component

The CRU firmware specialists have created a dedicated folder (DETECTOR-UL) to house the user logic component before compilation. This folder is located in the CRU firmware repository available on GitLab (Bourrion, 2016). The sub-folder containing all user logic files must be named after the subdetector (e.g., DETECTOR-UL/MID), and deployed as a CRU firmware submodule. The compilation of the CRU firmware can be done the old-fashioned way, using the intel Quartus prime pro GUI interface, or by using command lines. The most common way of running the compilation is through the command line, which can be achieved by diving into the following directory (cru-fw/preint/syn-mid) and running the following commands: “make ip_gen;” and “make synthesis”. A detailed explanation of how to compile

the CRU firmware is described in Appendix C. The compilation time before and after implementing the user logic component remains roughly similar (~ 4-5 hours). This is due to the timing constraints, which have a significant influence on the compilation time since the fitter attempts to minimize routing delays in order to match the required clock frequencies.

5.2.3. Test bench configuration

Although all required software (Quartus, ModelSim, ReadoutCard, FLP suite, etc…) are already installed in the FLP server. It is still necessary to properly configure the readout chain before proceeding with the hardware tests. The configuration can be achieved using the FLP server, which is linked to all components of the chain. First, the CRU firmware has to be configured to operate in “user logic mode” and the two GBT links connected to the regional card must be synchronized and locked to the CRU clock using the GBTx programmer software installed on an external PC. This step is necessary before going back to the FLP server to configure the internal GBT MUX of the CRU firmware, which is used to implement the custom MID trigger format (see section 2.6.5). The last component to configure is the LTU, this process is pretty straightforward. Step by step instructions on how to configure the test bench and the commands used are also given in Appendix C.

5.2.4. Data acquisition

Following the configuration of the test bench, the data acquisition is achieved using the CTP emulator graphic user interface program installed on the FLP server. As mentioned in section 2.6.2, the LTU can be configured to operate in standalone mode and emulates the CTP protocol. This enables the MID team to run the data acquisition independently of the CTP electronic board. Figure 5.9 shows a screenshot of the CTP emulator. It is designed to assist in setting up and conducting the acquisition on the FLP server without the need to know the functioning of the DAQ system. Furthermore, the program simplifies the selection of both continuous and triggered readout mode operations and provides, among other things, the customization of the data acquisition. However, it should be noted that the CTP emulator is designed to accommodate the majority of the subdetectors, and some functionalities such as (TPC-SYNC, TPC RST, TOF RST) are made for specific subdetectors and are not utilized in the MID data acquisition. The functionalities required to execute the MID data acquisition are highlighted in red in Figure 5.9. To begin an acquisition, the user can simply click on the start button and press stop to end the acquisition. The physics and Calibration

triggers are not mandatory in the continuous readout mode, although they can be included.

Data handling on the FLP server is taken care of by the ReadoutCard program. The program receives data packets from the user logic through the PCIe interface. The integrity of the packets is verified through the information present in the packet headers. A customized MID DAQ tool is implemented to detect anomalies of the payload included in the packets. The validity of the payload is confirmed by analyzing local and regional events extracted from the payload. An alternative way of verifying the packets is to configure the CRU in common and user logic mode. In this mode both the raw data and user logic packets are decoded and compared for potential errors. The hardware test results and instructions for reproducing these results are available in Appendix D.

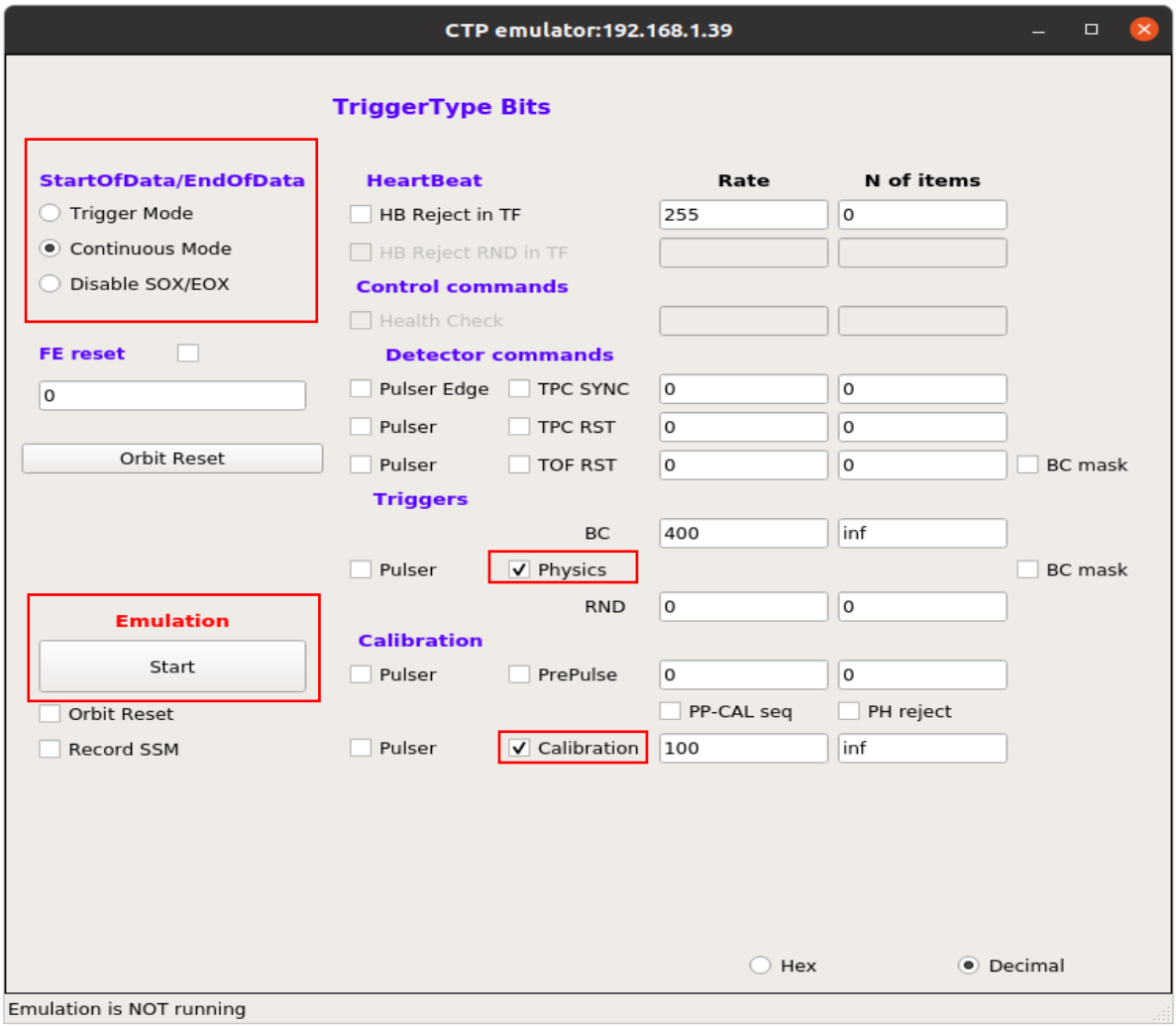

Figure 5.9: Screenshot of the CTP emulator graphical user interface, which is used to conduct a data acquisition in the test bench

5.3. Results

This section analyses the simulation and hardware test results.

5.3.1. Discussion of the results

The simulation stress results of the working user logic prototype were generated in ModelSim and a portion of it is illustrated in Figure 5.10. This was achieved after spending several hours of debugging and moving back and forth between the module-based and system-based verifications. Although, the system-based verification was mostly used for lengthy, and the module-based verification was used for short simulations.

As can be observed, the user logic output packets fulfill the readout control protocol criteria indicated in section 3.4.1, which states that packets created by the user logic must begin and terminate with the SOP and EOP signals, and each packet must be enclosed by the RDHs. The serialized data block format (illustrated in Figure 3.5), which relies on the concatenation of local and regional event byte fragments into multiple data blocks was also successfully implemented in the payload and the desired results were obtained. A clear comparison of data before and after being processed by the user logic is illustrated in Figure 5.10. It shows the final outcome of the data before and after the user logic firmware has performed all processing stages. The simulation tests validate the aim of this research, which is to enhance the way data are processed and only transmit valuable information using a very specific format as stated in chapter 2.

The hardware tests could not be completed until a configuration error in the CRU firmware file was resolved, which caused the data acquisition to crush every time the user logic component was integrated into the CRU firmware. The problem was first considered to be caused by metastability in the data when it was moved between the Avalon and the GBT segments, given the two segments had independent input clocks (see Figure 4.1). However, by inserting a clock synchronizer delay between the two segments that were in sync with the Avalon clock and using Intel Quartus timing analyzer tool to check for potential timing errors, it was discovered that this was not the source of the issue since the user logic continued to corrupt the readout chain.

5.3 Results

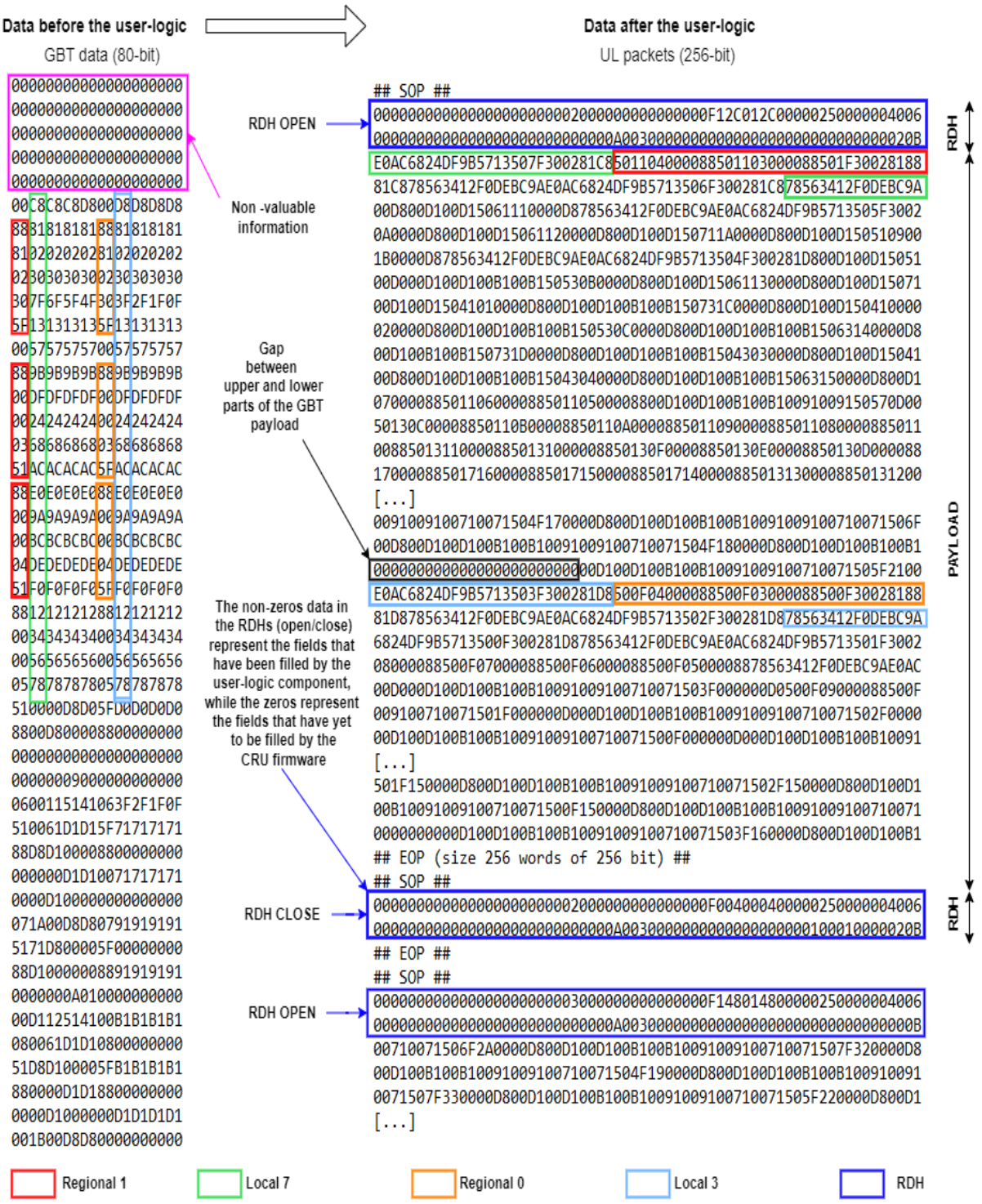

Figure 5.10: Simulation results extracted from ModelSim during the tests. This image compares data before and after being processed by the user logic

A consultation with the CRU experts revealed that the error was due to a parameter error in the "cru-mid.qsf" file located under the directory "/home/flp/cru-fw/preint/syn-mid/". Initially, the setting in this file was for subdetectors that use the GBT protocol in wide frame mode and had to be modified since the MID subdetector uses the GBT protocol in standard frame mode (see section 2.4.5 for details on the GBT protocol). Eventually, this issue was resolved and the hardware tests were performed. However, as expected, the hardware tests result did not reflect the simulation results at first but were refined after each iteration until complete accuracy was achieved.

5.3.2. Resource usage

The CRU firmware combined with the user logic component use about 160k (38%) ALMs and 1355 (50%) RAM blocks of the available resources. These results were obtained after integrating and compiling the CRU firmware with the user logic component. Table 5.1 provides a summary of the total FPGA resource used.

Table 5.1: FPGA resource usage of the CRU firmware after insertion of the user logic component

Resource name	Total in ratio	Total in percentage
Logic utilization (in ALMs)	160,282 / 427,200	38 %
Pins	369 / 960	38 %
Block memory bits	19,982,660 / 55,562,240	36 %
RAM blocks	1,355 / 2,713	50 %
Digital Signal Processing Blocks	0 / 1,518	0 %
RX channels	41 / 72	57 %
TX channels	41 / 72	57 %
Phase Locked Loops (PLLs)	59 / 144	41 %

Based on the information obtained in Table 3.3 and Table 5.1, the resources usage of the user logic component can be computed. It uses around 37k (9%) ALMs and 271 (10%) RAM blocks of the overall resources. These findings meet the requirement of this study, but are not good enough as the long-term aim is to process data from 16 GBT links while keeping the overall RAM consumption below 75%. Optimization of the RAM usage in the user logic will be required in order to process data from the complete readout chain. However, this task falls outside the scope of this study.

5.3.3. Performance

During the final phase of the research, the performance of the user logic component was evaluated using a series of stress tests performed in ModelSim. The tests were carried out using the system-based verification technique, which includes the CRU and readout electronics modules. The advanced stimulus was configured in such a way that an unlimited number of valuable events was transmitted to the user logic in order to test its limits.

The tests aimed to analyze its stability and reliability, as well as to monitor the status of its FIFOs. The longest stress test lasted 18 hours, and the results obtained are illustrated in Table 5.2.

From the table below, it can be observed that none of the FIFO busy flags were detected and no valuable events were rejected. The user logic went through 18 hours of simulation without detecting any errors in the design. Furthermore, FIFO 64x8 and the FIFO 16x8 have the fewest number of words recorded. These FIFOs could have been replaced with simple registers to reduce the RAM consumption in the FPGA. However, they were left as they are to facilitate the expansion of the design. More details are provided in the next chapter.

Table 5.2: Stress test results

Submodule	FIFO size	Highest number of words recorded in ratio	Highest number of words recorded in percentage	Number of busy flag raised	Number of events rejected
Header	64x8	1/8	12.5%	0	0
Local Handshake synchronizer	168x64	13/64	20.3%	0	0
	168x128	126/128	98.4%	0	0
Regional Handsgake synchronizer	40x64	20/64	31.2%	0	0
	40x128	76/128	59.3%	0	0
Payload serializer	256x256	88/256	34.3%	0	0
	16x8	1/8	12.5%	0	0

Overall, the user logic performed admirably during the stress testing. The findings shown in Table 5.2 demonstrate that the user logic component is stable, reliable, and built to withstand any form of collisions without issues. It is important to highlight that the tests were performed in continuous readout mode, which is enough for evaluating the design.

The next chapter will conclude the study by summarizing the key research findings and their contribution thereof. It will review the constraints and propose recommendations to improve and facilitate the expansion of the user logic design.

CHAPTER 6

Conclusions and recommendations

6.1. Conclusions

The ALICE detector is undergoing a major upgrade. The goal of the upgrade is to address the challenges of reading out lead-lead collisions at 50 kHz and proton-lead collisions at 1 MHz and higher. As part of the upgrade, the MTR subdetector was renamed to MID to support both continuous and triggered readout modes. This research aimed to enhance the way data are handled in the newly upgraded MID readout chain using a customized user logic component before the commencement of LHC Run 3 in 2022. The methodology implemented in this research shows that it is feasible to considerably reduce the large data rate and the amount of work performed at the O^2 level by 80%. The user logic prototype has passed the simulation tests and so fulfills the MID requirements. Its implementation into the CRU firmware can be considered successful based on the design requirements mentioned in section 3.4.1. The user logic has also been evaluated using a small-scaled readout chain test bench located at iThemba LABS. The results obtained indicate that with some optimizations, this prototype can immensely contribute to the development of a full-scale user logic capable of processing data from the entire MID readout chain.

6.2. Recommendations

This thesis proposes a refined approach for handling MID raw data during LHC Run 3. However, certain challenges need to be investigated in order to expand the user logic architecture and improve the performance of the readout chain. As a result, future effort should concentrate on the optimization of the RAM usage indicated in section 5.3.2 and development of new features. Based on the results and conclusions given, the following recommendations are made.

As mentioned in section 4.2.4, the user logic component has access to 24 incoming GBT links and for this study, only two GBT links were used. To provide flexibility to the design, two GBT mapping submodules were implemented to select as input any 2 out of the 24 GBT links available. However, this will no longer be the case once a full-scaled user logic component is incorporated in the CRU firmware, as it will require too many registers. This leaves two possible solutions: either get rid of this feature and only use the required number of GBT links (16 out of 24) while discarding the remaining links, or improve this feature and use the remaining links as spare links. The first solution is the easiest but may pose a problem if any of the links fails. The second solution will also use the required number of

GBT links but will provide an option to replace any of the 16 GBT links with a spare link via Avalon MM in case of failure.

The FIFO sizes implemented in the user logic component were chosen to accommodate the stress tests during simulation. A quick suggestion for the future is to leave them as they are. The results outlined in section 5.3.3, have proven that these FIFO sizes can handle any type of collision. To overcome the RAM consumption issue, these FIFOs can be optimized to use fewer resources in Quartus prime pro.

Finally, the user logic registers can be accessed by following the instructions described in Appendix E. However, only someone who understands VHDL coding can decipher the information extracted from these registers. A Python script could be written to ease the configuration and monitoring of the user logic component using a graphical user interface.

6.3. Research and industry applications

This study resulted in the creation of a test bench where undergraduates and post-graduates can learn and understand the fundamentals of FPGA development. The test bench was developed in such a way that it may easily be adapted to future FPGA applications. It can be used to understand high-speed data transmission in High Energy Physics experiments, as well as to contribute to the studies on radiation-hard DC-DC converters and other related topics. In terms of industrial use, this test bench can be utilized to design and test the full-scaled user logic component as well as to maintain it on behalf of the MID team during the LHC Run 3 and 4. Furthermore, this test bench can also be used for training purposes, especially for engineers and technicians.

References

ALICE Collaboration, 2008. The ALICE experiment at the CERN LHC*., Journal of Instrumentation (JINST),* 3(08), p259.
https://doi.org/10.1088/1748-0221/3/08/S08002

ALICE Collaboration, 2012. Upgrade of the ALICE Experiment: Letter of Intent*, Journal of Physics G: Nuclear and Particle Physics,* CERN-LHCC-2012-012*,* 41(8), pp.107-109.
https://doi.org/10.1088/0954-3899/41/8/087001

ALICE Collaboration, 2014. Technical Design Report*: Upgrade of the ALICE Read-out & Trigger System, C*ERN-LHCC-2013-019, ALICE-TDR-015, p. 165.
http://cds.cern.ch/record/1603472/files/ALICE-TDR-015.pdf?version=6

Blanc, A. & Dupieux, P., 2008. The trigger system of the ALICE muon spectrometer at the LHC. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors, and Associated Equipment.* Elsevier, 604(2), pp. 301-303.
https://doi.org/10.1016/j.nima.2009.01.186

Bourrion, O., Bouvier, J., Costa, F., Dávid, E., Imrek, J., Nguyen, T.M., & Mukherjee, S., 2021. Versatile firmware for the Common Readout Unit (CRU) of the ALICE experiment at the LHC. *Journal of Instrumentation (JINST),* 16(05), p. 1-17.
https://doi.org/10.1088/1748-0221/16/05/P05019

Bourrion, O., Evans, J., Imrek, J., Jusko, A., Kluge, A., Krivda, M., Kvapil, J., Lietava, R., Pérez Moreno, L.A., Villalobos-Baillie, O. & Willsher, E., 2019. Interface between CTS-CRU and CTS-Detector Front Ends. *Trigger Notes for Developers.*
https://readthedocs.web.cern.ch/ate/files/113214939/113214965/1/1557935021000/CTS_CRU_FE_interface.pdf

Bourrion, O., 2015. *CRU firmware Gitlab, Readout protocol.* [Private website]
https://gitlab.cern.ch/alice-cru/cru-fw/-/tree/master/DWRAPPER

Bourrion, O., 2016. CRU Firmware project GitLab*.*
https://gitlab.cern.ch/alice-cru/cru-fw

Boyles, N., Buthelezi, Z., Winberg, S. & Mishra, A., 2021. User Logic Development for the Muon Identifier Common Readout Unit for the ALICE Experiment at the Large Hadron Collider. *Instrumentation and Detectors*.
https://doi.org/10.48550/arXiv.2104.05476

Broadcom, 2005. *High-Speed Networking Fiber Optics.*
https://www.broadcom.com/products/fiber-optic-modules-components/networking
[Accessed 05 July 2021]

Buncic, P., Krzewicki, M. & Vande Vyvre, P., 2015. Technical Design Report for the Upgrade of the Online-Offline Computing System*, CERN*, CERN-LHCC-2015-006, ALICE-TDR-019.
http://cds.cern.ch/record/2011297/

Brüning, O.S.(ed), Collier, P.(ed), Lebrun, P.(ed), Myers, S.(ed), Ostojic, R.(ed), Poole, J.(ed) & Proudlock, P., 2004. LHC Design Report*, Accelerators and Storage Rings,* CERN.
https://doi.org/10.5170/CERN-2004-003-V-1

Caratelli, A., Paillard, C., Bonacini, S., Kloukinas, K.,Marchioro, A., Moreira, P. &., De Oliveira, R., 2015. The GBT-SCA, a radiation tolerant ASIC for detector control and monitoring applications in HEP experiments, *Journal of Instrumentation (JINST)*, 10, C03034. https://doi.org/10.1088/1748-0221/10/03/C03034

Consesa Del Valle, Z., 2007. Performance of the ALICE muon spectrometer. Weak boson production and measurement in heavy-ion collisions at LHC*, CERN-THESIS-2007-102,* Nantes: Universite de Nantes, pp. 117-132. https://inspirehep.net/files/a09d6ba3024ce006f182765b5231cce9

Costa, F., Kluge, A., & Vande Vyvre, P., 2017. The detector read-out in ALICE during Run 3 and 4. *Journal of Physics: Conference Series,* 898, 032011, pp. 1-4. https://doi.org/10.1088/1742-6596/898/3/032011

Costa, F., 2019. Assessment of the ALICE O2 readout servers. *24th International Conference on Computing in High Energy and Nuclear Physics (CHEP*), 245, pp.6. https://doi.org/10.1051/epjconf/202024501013

Costa, F., 2021. *CERN Alice O2 Group Gitlab, Raw Data Header (RDH).* [Private webite] https://gitlab.cern.ch/AliceO2Group/wp6-doc/-/blob/master/rdh/RDHv6.md

Dupieux, P., 2006. A new front-end for better performances of RPC in streamer mode*.* Nuclear Instruments and Methods in Physics Research: *Proceeding 6th International workshop on Resistive Plate Chambers and Related Detectors (RPC 2001)*, 508(2), pp. 185-188. https://doi.org/10.1016/S0168-9002(03)01348-2

Elena, B., 2017. Particle identification performance at ALICE, Nuclear Experiment: *Proceedings of the Fifth Annual Conference on Large Hadron Collider Physics*, 15-20 May 2017. Shanghai: Shanghai Jiao Tong University. https://doi.org/10.48550/arXiv.1709.00288

Eulisse, K., Konopka, P., Krzewicki, M., Richter, M., Rohr, D. & Wenzel, S., 2019. Evolution of the ALICE Software Framework for Run 3*.* The European Physical Journal WEB Conferences*: 23rd International Conference on Computing in High Energy and Nuclear Physics (CHEP 2018)*, 214, 05010, pp. 2-4. https://doi.org/10.1051/epjconf/201921405010

Evans, D., Jusko, A., Krivda, M., Lietava, L. & Moreno, P., 2016. *Trigger System Design Review,* Birmingham: CERN, 2(1), pp. 21-31. https://indico.cern.ch/event/533412/contributions/2172791/attachments/1291065/1922927/CTPLTU12.pdf

Evans, L., Bryant, P., 2008. LHC Machine. *Journal of Instrumentation (JINST)*, 3(08), p165. https://doi.org/10.1088/1748-0221/3/08/s08001

Farahani, S., 2008. ZigBee and IEEE 802.15.4 Protocol Layers. In: *ZigBee Wireless Networks and Transceivers.* s.l.:Elsevier inc, pp. 33-135. https://doi.org/10.1016/B978-0-7506-8393-7.X0001-5

Ferreti A, 2019. The upgrade of the RPC-based ALICE Muon Trigger. *Journal of Instrumentation (JINST):* 14th Workshop on Resistive Plate Chambers and Related Detectors, 14, C06011, June 2019. https://doi.org/10.1088/1748-0221/14/06/C06011

References

Giubellino, P., 2015. The Big-Bang in the Lab*. Science and Technology Festival (TechFest),* Powai: IIT Bombay, [PowerPoint presentation]. https://indico.cern.ch/event/357092/contributions/1766869/attachments/710437/975250/Techfest_IIT_Mumbai_Jan_2015.pdf

Horvath, A., 2006. LHC ring with its four main experiments, *Wikipedia.* https://en.wikipedia.org/wiki/Large_Hadron_Collider#/media/File:LHC.svg

Intel, 2019. Quartus Prime Pro 18.1 Software https://www.intel.com/content/www/us/en/software-kit/664780/intel-quartus-prime-pro-edition-design-software-version-18-1-for-linux.html? [Accessed 06 07 2020].

Intel, 2020. ModelSim*-Intel® FPGA Edition Software. intel.com/content/www/us/en/software/programmable/quartus-prime/model-sim.html [Accessed 2 April 2020].

Intel, 2021. Intel® MAX® 10 FPGA Device. https://www.intel.com/content/www/us/en/products/details/fpga/max/10.html [Accessed 22 December 2021].

Intel, 2022. Intel® Arria® 10 FPGA Device. https://www.intel.com/content/www/us/en/products/details/fpga/arria/10.html [Accessed 07 March 2022].

Intel, n.d. FPGAs Resources Center. https://www.intel.com/content/www/us/en/products/details/fpga/resources/overview.html [Accessed 28 June 2021].

Kvapil, J., Bhasin, A., Bombara, M., Evans, D., Jusko, A., Kluge, A., Krivda, M., Roman, L.,Sanket, K.N., et al., 2021. ALICE Central Trigger System for LHC Run 3*.* European Physical Journal (EPJ) Web of Conferences: *Proceeding 25th International Conference on Computing in High-Energy and Nuclear Physics (CHEP 2021)*, 251(2), pp. 1-10. https://doi.org/10.48550/arXiv.2106.08353

Krivda, M., Evans, D., Graham, K.L., Jusko, A., Lietava, R., Villalobos Baillie, O., Zardoshti, N., Sefcik, M., Kralik, I. & Perez Moreno, L.A., 2018. The ALICE Trigger System for LHC Run 3. *Proceeding of the Topical Workshop on Electronics for Particle Physics (TWEPP),* p.149, 11-14 September, California: University of Santa Cruz. https://doi.org/10.22323/1.313.0149

L., Musa, FPGAS in high energy physics experiments at CERN, 2008 International Conference on Field Programmable Logic and Applications, pp. 2-2. https://doi.org/10.1109/FPL.2008.4629896.

Manen, S., Dupieux, P., Joly, B. & Jouve, F., 2013. FEERIC, a very-front-end ASIC for the ALICE Muon Trigger Resistive Plate Chambers*. Conference: Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)* Seoul: IEEE, pp. 1-4. https://doi.org/10.1109/NSSMIC.2013.6829539

Microchip inc, 2021. SAM D21 microcontroller.
https://ww1.microchip.com/downloads/en/DeviceDoc/SAM-D21-DA1-Family-Data-Sheet-DS40001882H.pdf
[Accessed 14 August 2020].

Mitra, J., David, E., Mendez, E., Khan, S.A., Kiss, T., Baron, S., Kluge, A. & Nayak, T., 2018. Trigger and timing distributions using the TTC-PON and GBT bridge connection in ALICE for the LHC Run 3 Upgrade, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors, and Associated Equipment*, Volume 922, pp. 119-133, ISSN 0168-9002.
https://doi.org/10.1016/j.nima.2018.12.076

Moreira, P., Ballabriga, P., Baron, S., Bonacini, S., Cobanoglu, O., Faccio, F., Fedorov, T., Francisco, R., Gui, P. & Hartin, P., 2010. The GBT Project, Engineering: *Proceedings of the Topical Workshop on Electronics for Particle Physics (TWEPP)*, pp.342-346.
https://doi.org/10.5170/CERN-2009-006.342

Moreira P, Christiansen J & Wyllie K, 2015. Draft: GBT Manual*, CERN.*
http://padley.rice.edu/cms/OH_GE21/gbtxManual_2016.pdf

Nešković, G., 2018. Data Distribution and Load Balancing for the ALICE Online-Offline System, [PowerPoint presentation]
https://indico.cern.ch/event/587955/contributions/2935761/attachments/1678768/2701788/CHEP18_WP5_O2_Data_Dist_rev1.pdf
[Accessed 14 August 2020].

Renard, C., 2021. MID readout electronics project website. *Subatech.*
http://www-subatech.in2p3.fr/~electro/projets/alice/dimuon/trigger/upgrade/index.html

Richter, M., Krzewicki, M. & Eulisse, G., 2019. Data Handling in the ALICE O2 Event Processing. European Physical Journal (EPJ) Web of Conferences: *23rd International Conference on Computing in High Energy and Nuclear Physics (CHEP)*, 214, 01035.
https://doi.org/10.1051/epjconf/201921401035

Sauli, F., 2014. Resistive plate chambers. *Gaseous Radiation Detectors: Fundamentals and Applications*, pp. 344-364.
https://doi.org/10.1017/CBO9781107337701.014

Schaeffer, A., 2020. New schedule for CERN's accelerators and experiments.
https://home.cern/news/news/accelerators/new-schedule-cerns-accelerators-and-experiments
[Accessed 5 December 2020].

Skyworks, 2018. Jitter Attenuator.
https://www.skyworksinc.com/-/media/Skyworks/SL/documents/public/data-sheets/Si5345-44-42-D-DataSheet.pdf
[Accessed 2021 April 25].

Sierra-Polanco, T., Milanés, D. & Vera, C.E., 2018. Configuration of operation modes for a scintillating fiber subdetector in the LHCb experiment. *Neogranadina Science and Engineering,* Bogotá, 28(2), pp. 43-62.
https://doi.org/10.18359/rcin.2854

References

Stocco, D., 2020. *MID issues* [Interview] (23 April 2020).

Tkácik, M., Jadlovský, J., Jadlovská, S., Koska, L., Jadlovská, A. & Donadoni, M., 2020. FRED - Flexible Framework for Frontend ElectronicsControl in ALICE Experiment at CERN. *Processes,* 8(5), pp. 565.
https://doi.org/10.3390/pr8050565

Appendices

Appendix A

Source code

The project's source code is publicly available at the following link:
https://github.com/dthysdin/Meng-Thesis/tree/master/hdl

Appendix B

Instructions for reproducing the simulation test results

Note:

The simulation stress test results obtained during this study, are publicly available at the following link: https://github.com/dthysdin/Meng-Thesis/tree/master/sim/ul_output_files

This section is intended to provide instructions on how to reproduce the above-mentioned test results, which are enough to validate the functionalities of the user logic component. However, if interested, one could take a step further and run the system-based verification, which provides a more in-depth understanding of the entire readout chain.

B1. Run the module-based simulation

The user logic component can be tested independently by following the instructions below.

1. Clone the user logic repository on GitHub using the command below:
   ```
   git clone –recursive https://github.com/dthysdingou/Meng-Thesis.git
   ```
2. Launch ModelSim*-Intel® FPGA Edition Software
3. Use the ModelSim command window and navigate to “Meng-Thesis/sim”
4. Execute the command below to start the simulation
   ```
   do ul_run_file.do
   ```
5. Wait until the simulation is completed
6. The output result files can be obtained under “Meng-Thesis/sim/ul_output_files”
7. The input files used for this simulation are obtained from a system-based simulation performed during this study. These files are under: “Meng-Thesis/sim/ul_intput_files”.

B2. Run the system-based simulation

The user logic component can be tested together with other systems of the readout chain by following the instructions mentioned below.

1. Clone the MID readout electronics firmware using the command below.
   ```
   git clone –recursive https://gitlab.cern.ch/alice-mid/readout.git
   [Permission required]
   ```
2. Clone the CRU firmware using the command below.
   ```
   git clone --recursive https://gitlab.cern.ch/alice-cru/cru-fw.git
   [Permission required]
   ```
3. Use ModelSim to navigate to “readout/mid_common-fw/simulatiuon/modelsim/”
4. Run the command below to start the simulation.
   ```
   do mid_user_test_run_msim_vhdl_linux.do
   ```

Appendix C

Instructions for setting up the test bench at iThemba LABS

Note:

In this section, we will assume that all software required to configure the MID test-bench is already installed and operational. Instructions on how to install that software is provided in the following link: install the required software [permission is required].

After installing the necessary software, the test bench must be properly configured before proceeding with the hardware tests. This is accomplished by following the instructions outlined below. Please keep in mind that certain commands are executed on the FLP server while others are executed on a Centos PC.

C1. Clone the CRU firmware GitLab repository [FLP server]

In order to provide better code management, keep the design secure, and accessible to everyone working on the ALICE upgrade, the CRU firmware is hosted on a GitLab repository, which can be retrieved using the command mentioned in Appendix B, B2(3).

C2. Compilation of the user logic with the CRU firmware [FLP server]

The compilation of the CRU firmware with the user logic component can be performed by the commands below.

```
cd cru-fw/preint/syn-mid
make ip_gen; make synthesis
```

C3. Program the ARRIA 10 FPGA on the CRU [FLP server]

The SRAM object file (.sof) is generated after the compilation must be used to program the CRU FPGA. This is accomplished using the command below.

```
quartus_pgm -c 1 -z --mode=JTAG --operation="p;cru.sof@1"
```

C4. Initialization of the system [FLP server]

After programming the Arria 10 FPGA on the CRU, the system must be initialized before the tetsts. This must be done as root user (repeat if not successful).

```
su -
for i in 3b 3c ; do echo 1 > /sys/bus/pci/devices/0000\:$i\:00.0/remove; echo 1 > /sys/bus/pci/rescan; done
```

C5. Set the ReadoutCard environment [FLP server]

The ReadoutCard environment is a program developed by the CRU team to allow the user to configure the system. This program can be set using the following command:

```
module load
```

C6. Detect the CRUs in the system [FLP server]

This command lists all the CRUs installed in the system. This includes the PCI addresses, the sequence number, the serial and Endpoint IDs as well as the CRU firmware and user logic version.

```
roc-list-cards
```

As expected, this program successfully detected the only CRU card present in the test bench and gave the following output:

```
================================================================
 #   Type   PCI Addr   Serial   Endpoint   NUMA   FW Version   UL Version
----------------------------------------------------------------------------------------------------------------
 0   CRU    3b:00.0    0000        0         1     da7521f4     da7521f4
 1   CRU    3c:00.0    0000        1         1     da7521f4     da7521f4
================================================================
```

As it can be observed, the CRU card is equipped with a PCIe edge connector, that provides a dual PCIe interface, each of which has a unique address (3b:00.0 and 3c:00.0). The endpoints can be seen as each of the CRU firmware output channels, i.e., the user logic data readout #0 is linked to the datapath wrapper #0, which is then liked to the PCIe interface #0. The FW and UL versions represent the latest commit tag provided by GitLab before the code was compiled.

C7. Configure the CRU [FLP server]

The “roc-config” is the command used to configure the ReadoutCard software. It requires several parameters depending on the sort of test the user wishes to perform. The command below displays information about the function of each parameter.

```
roc-config -h
```

In order to test the functionalities of the user logic component, the following parameters are required: --i#0 is the endpoint ID to be configured, --clock=ttc specifies that the clock is taken

from the TTC, --links=0-1 indicates GBT links #0 and #1 are enabled, --data=STREAMING indicates that the GBT datapath is in standard mode also known as streaming mode, --downstream=MID specifies that the downstream trigger data uses the custom MID trigger format (see section 2.6.5 for more details about the format), --pon indicates that the PON upstream flag is enabled, --onu=1 implies that the ONU address for the PON is 1, --user-logic means the user logic path is enabled, --force tells the system to disregard any previous setting, --user-and-common-logic denotes that the system will operate in both user logic and common modes and finally, --bypass means the cru firmware checker is bypassed.

```
roc-config --i=#0 --clock=ttc --links=0-1 --data=STREAMING --downstream=MIDTRG --pon –onu=1 --user-logic --force --user-and-common-logic --bypass
```

C8. Configure the GBTx [Centos PC]

The GBTx chip in the regional card must be configured in order to synchronize the Rx and Tx transmission clocks between the readout electronics and the CRU. The configuration can either be achieved via the GUI interface using the GBTx programmer or via a python script provided by Dr. Stocco. It is important to note that both procedures require root user access.

GUI interface

The GBtx programmer GUI can be launched by running the command below.

```
java -jar /root/gbtxprogrammer/releases/programmerv2.20180725.jar
```

After launching the GUI interface, make sure to check the dongle state, which should display “v1.d”. In case of error, check the USB cable connected to the regional card. Check that GBT link numbers #2 and #3 display below the scan button, pick #2, and set the watchdog to “OFF”. Click on the “import.” button and select the text file from the location listed below:

```
/home/CentosPC/gbtxprog/config/gbtx_mid_link0.tx
```

Click on “write GBTx” and on “read GBTx”, if successful the state should be locked to “idle, 18h”. Set the watchdog to “ON”, then under the scan button pick #3 and repeat the process mentioned above. This time, import the text file from the location listed below:

```
/home/CentosPC/gbtxprog/config/gbtx_mid_link1.txt
```

Repeat the write and read procedure until the state is successfully locked in “idle, 18h” then exit the program.

Python script (preferred option)

The GBTx chip can be configured using the commands below.

```
su -
cd /home/CentosPC/gbtxprog/
source ../venv/bin/activate
python3 gbt_vldb.py --config-file2 /home/CentosPC/gbtx_mid_link0.txt --config-file3 /home/CentosPC/gbtx_mid_link1.txt config
```

C9. Verify the configuration [FLP server]

The following command is used to ensure that the configurations executed in B7 and B8 are successfully implemented.

```
roc-status –i=#0
```

As expected, this command successfully recognized all parameters implemented in the CRU, and produced the following output:

```
CRU ID: 0
TTC clock | Fixed offset
User and Common Logic enabled
```

Link ID	GBT Mode	Loopback	GBT MUX	Mode	Datapath	RX freq(MHz)	TX freq(MHz)	Status	Power(uW)
0	GBT/GBT	None	TTC:MIDTRG	Streaming	Enabled	240.47	240.47	UP	316.0
1	GBT/GBT	None	TTC:MIDTRG	Streaming	Enabled	240.47	240.47	UP	369.4
2	GBT/GBT	None	TTC:MIDTRG	Streaming	Disabled	216.76	240.47	DOWN	0.0
3	GBT/GBT	None	TTC:MIDTRG	Streaming	Disabled	197.40	240.47	DOWN	0.0
4	GBT/GBT	None	TTC:MIDTRG	Streaming	Disabled	239.96	240.47	DOWN	0.0
5	GBT/GBT	None	TTC:MIDTRG	Streaming	Disabled	199.07	240.47	DOWN	0.0
6	GBT/GBT	None	TTC:MIDTRG	Streaming	Disabled	239.45	240.47	DOWN	0.0
7	GBT/GBT	None	TTC:MIDTRG	Streaming	Disabled	198.14	240.47	DOWN	0.0
8	GBT/GBT	None	TTC:MIDTRG	Streaming	Disabled	196.66	240.47	DOWN	0.0
9	GBT/GBT	None	TTC:MIDTRG	Streaming	Disabled	234.56	240.47	DOWN	0.0
10	GBT/GBT	None	TTC:MIDTRG	Streaming	Disabled	240.69	240.47	DOWN	0.0
11	GBT/GBT	None	TTC:MIDTRG	Streaming	Disabled	240.59	240.47	DOWN	0.0

Appendix D

Instructions for reproducing the hardware test results

Note:

The hardware test results and the output report file obtained during this study are publicly available at the following link: https://github.com/dthysdin/Meng-Thesis/tree/master/hw

This section explains how to reproduce the above-mentioned test results using the MID test bench available at iThemba LABS. The tests are performed using two terminal windows, one for storing data and the other for sending triggers using the LTU GUI interface.

D1. Source the O² program (Terminal 1)

This program was created by Dr. Stocco to allow users who are unfamiliar with the system to run the acquisition. It first sets up the O² environment and then waits for the CRU firmware to transmit data upon receiving triggers. As mentioned in Appendix A, the CRU is configured to operate in both common and user logic mode. As a result, this program stores both the raw and the compressed data extracted from the user logic. The maximum amount of raw and compressed data storage is set to “100 MB”.

```
. ~/setupO2.sh
 alienv enter O2/latest
/home/flp/daq_utils/scripts/launch_acquisition.sh -i "#0" -t external -u 2 -s "100M" -fl
```

D2. Source the LTU program and launch the GUI (Terminal 2)

Set the LTU environment and launch the LTU GUI interface using the following commands:

```
. /home/flp/setupLTU.sh
qtltu
```

Once the LTU control v1a window pops up, select “open” then click on the “CTP emulator “ button to access the emulator interface (refer to Figure 5.9 to see how to configure the emulator). At this stage, the LTU is ready for data acquisition and testing. To begin the acquisition, click "start", and to halt it, click “stop”. It merely takes a couple of seconds to fill up a 100 MB file. Keep an eye on Terminal 1 as it will notify you as soon as the files are full.

D3. Read the output data raw files (Terminal 1)

Following the completion of the acquisition, three distinct files are created. The raw data from the GBT links #0 and #1 can be found in “readout files_0_0.raw” and ”readout files_0_1.raw”, respectively, while the output data from the user logic can be found in “readout files_0_15”.

The instructions listed below show the commands used to read these files.

```
cd ../ul_common/
o2-mid-rawdump readout_file_0_0.raw | less
o2-mid-rawdump readout_file_0_1.raw | less
o2-mid-rawdump readout_file_0_15.raw | less
```

D4. Execute the O² checker (Terminal 1)

The O² checker compares the three files created during the DAQ to ensure data consistency and identify errors. The command required to execute the checker is listed below.

```
o2-mid-raw-ul-checker --feeId-config-file /home/flp/daq_utils/config/feeId_mapper.txt --crate-masks-file /home/flp/daq_utils/config/crate_masks.txt --bare-filenames readout_file_0_0.raw,readout_file_0_1.raw --ul-filenames readout_file_0_15.raw
```

D5. Read the report file (Terminal 1)

A report file is created right after the checks have been completed. This file details the number of errors found and events missing from the user logic output file. The command listed below enables the user to open the report file.

```
more check_ul.txt
```

In most cases, no faults will be found. The hardware tests can be redone by repeating the preceding stages. Otherwise, shut all terminal windows to bring the tests to an end.

Appendix E

Instructions for accessing the registers

Note:

The CRU team uses the “roc-reg” tools to read and write the CRU firmware registers. These tools use low-level functions for directly interacting with the CRU over the PCI-e interface. They are intended for development and debug purposes. The “roc-reg-read” command is used to read the register while the “roc-reg-write” is used to write to the register. These commands use four parameters: “--i”, which is the endpoint ID of the CRU, displayed in the output of the “roc-list-cards” command (see Appendix C). The second parameter is “channel”, which is used to access the Avalon bus, the channel to be used is 2. The third parameter is “address”, which is the registered address assigned by the user logic. All addresses related to the user logic component always start with “--add=0xC8”. The fourth parameter varies based on the purpose of the action to be executed. During reading, the fourth parameter is “--range”, which determines the range of the register to be read. During write, the fourth parameter is “--val”, which represents the value written to the register.

The following commands are used to read/write the user logic registers. These commands are executed on the FLP server.

==========

E1. Read all Avalon registers under

```
roc-reg-read-range --i=#0 --ch=2 --add=0xc80004 --range=31      # Read all registers
```

==========

E2. Write MID Reset

```
roc-reg-write --i=#0 --ch=2 --add=0xc80000 --val=0x00000001    # Reset
```

==========

E3. Read/Write MID CRUID register

```
roc-reg-write --i=#0 --ch=2 --add=0xc80004 --val=0x00000001   # Write CRUID = 1
roc-reg-write --i=#0 --ch=2 --add=0xc80004 --val=0x00000000   # Write CRUID = 0
roc-reg-read-range --i=#0 --ch=2 --add=0xc80004 --range=1      # Read CRUID
```

==========

E4. Read/Write MID switch register content

```
roc-reg-write --i=#0 --ch=2 --add=0xc80008 --val=0x00000000 # Write switch (0)
roc-reg-write --i=#0 --ch=2 --add=0xc80008 --val=0x00000001 # Write switch (1)
roc-reg-write --i=#0 --ch=2 --add=0xc80008 --val=0x00000002 # Write switch (2)
roc-reg-write --i=#0 --ch=2 --add=0xc80008 --val=0x00000003 # Write switch (3)
roc-reg-read-range --i=#0 --ch=2 --add=0xc80008 --range=1     # Read switch
```

==========

E5. Read/Write GBT Mapping

roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x00000000 #UL link 0 => GBT link 0
roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x00000001 #UL link 0 => GBT link 1
roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x00000002 #UL link 0 => GBT link 2
roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x00000003 #UL link 0 =>GBT link 3
-- ..... # .....
roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x00000009 #UL link 0 => GBT link 9
roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x0000000A #UL link 0 => GBT link 10
roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x0000000B #UL link 0 => GBT link 11

roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x00000000 #UL link 1 => GBT link 0
roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x00000001 #UL link 1 => GBT link 1
roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x00000002 #UL link 1 => GBT link 2
roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x00000003 #UL link 1 => GBT link 3
..... # .....
roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x0000000A #UL link 1 => GBT link 10
roc-reg-write --i=#0 --ch=2 --add=0xc8000C --val=0x0000000B #UL link 1 => GBT link 11
==========

E6. Read/Write MID synchronization

roc-reg-write --i=#0 --ch=2 --add=0xc80014 --val=0x00000080 # Write sync value
roc-reg-read-range --i=#0 --ch=2 --add=0xc80014 --range=1 # Read sync value
==========

E7. Read trigger register

roc-reg-read-range --i=#0 --ch=2 --add=0xc80034 --range=1 # Read trigger monitor
==========

E8. Read datapath wrapper registers

roc-reg-read-range --i=#0 --ch=2 --add=0xc80038 --range=1 # Read DWrapper#0 register
roc-reg-read-range --i=#0 --ch=2 --add=0xc8003C --range=1 # Read DWrapper#1 register
==========

E9. Read gbt registers

roc-reg-read-range --i=#0 --ch=2 --add=0xc80040 --range=2 # read GBT#0 and GBT#1
==========