Compressed Baryonic Matter Experiment

Technical Design Report for the CBM

Online Systems – Part I

DAQ and FLES Entry Stage

The CBM Collaboration

July 2023

Technical Design Report for the CBM Online Systems – Part I

DAQ and FLES Entry Stage

The CBM Collaboration[1]

Imprint

Editors:
J. de Cuveland, D. Emschermann, V. Friese, I. Fröhlich, P. Gasik, D. Hutter, W.F.J. Müller, C. Sturm

Corresponding Editor and responsible:
P. Gasik (p.gasik@gsi.de)

With major contributions by:
A. Bercuci, I. Deppner, J. Frühauf, T. Galatyuk, M. Gumiński, N. Herrmann, P. Kähler, O. Keller, M. Kruszewski, J. Lehnert, P.-A. Loizeau, J. Michel, C. Pauly, F. Salem, F. Schintke, A. Rost, E. Rubio, J. Saini, C. Schiaua, D. Schledt, V. Sidorenko, A. A. Weber, W. Zabołotny

Approved by FAIR on 24 June 2023

Published July 2023 by GSI Helmholtzzentrum für Schwerionenforschung GmbH, Planckstraße 1, 64291 Darmstadt, Germany (http://www.gsi.de).

GSI is a member of the Helmholtz association of national research centers (http://www.helmholtz.de).

DOI:10.15120/GSI-2023-00739

Published only online.

Version 2c39a34

[1]The full list of the CBM members is given in Appendix C

The activities within the CBM project are supported by (status November 2022):

The National Natural Science Foundation of China under grants no. 11927901, 11420101004, 11735009, U1832118,

The Ministry of Science and Technology of China under grants no. 2018YFE0205200, 2016YFA0400100,

FAIR-CZ-OP (CZ.02.1.01/0.0/0.0/16 013/0001677) and FAIR-CZ (LM2018112), Czech Republic,

The European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072,

Bundesministerium für Bildung und Forschung, Germany, under grants 05P15VHFC1, 05P19PMFC1, 05P19PXFCA, 05P19RFFC1 and 05P19RGFCA,

Deutsche Forschungsgemeinschaft, Germany, under grant 405580046,

GSI Helmholtzzentrum für Schwerionenforschung, Germany,

Helmholtz International Centre for FAIR, Germany,

Helmholtz Graduate School HGS-HIRe,

Helmholtz Forschungsakademie Hessen für FAIR, Germany,

Frankfurt Institute for Advances Studies, Germany,

Variable Energy Cyclotron Centre Kolkata, India (EHEP&A Group), University of North Bengal, India,

IDUB-POB-FWEiTE-1 project granted by Warsaw University of Technology, Poland, under the program Excellence Initiative: Research University (ID-UB),

Romanian Ministry of Research, Innovation and Digitization NUCLEU Project Contract PN 19060103 and RO-FAIR Project FAIR-03/16.11.2020.

Contents

Preface

Despite the current uncertainties in the future of FAIR, the CBM collaboration is continuing its effort towards the construction of a high-resolution and high-rate experiment with the goal of exploring the region of high baryon density in the QCD phase diagram. Recently, the Critical End Point (CEP) of a possible first-order phase transition was predicted to be located at baryon densities and temperature which can be reached with beam energies available from the SIS100 accelerator. The CBM scientific program to search for the location of the first order phase transition and to identify the CEP is currently unique world-wide. In order to exploit this unique opportunity, CBM is preparing its readiness for the very first SIS100 beams to be extracted from the machine.

The data acquisition and processing concept of CBM is a novel and mandatory part of the program since the anticipated signatures cannot be extracted by means of a conventional, triggered acquisition system. The Online Systems (Part I) Technical Design Report contains the description of the hardware and FPGA design which forms the basis of the CBM data processing chain. Its acceptance is crucial for reaching the overall goals and is therefore pushed forward despite the lack of some information which is caused by the unacceptable Russian attack of Ukraine and the resulting sanctions instituted. The suspension of membership in the CBM collaboration of Russian institutes was endorsed by the CBM Collaboration Board on May 18, 2022. Besides the superconducting dipole magnet, CBM lost its centrality and event plane defining device, the Projectile Spectator Detector (PSD), and the Beam Fragmentation Time-Zero Counter (BFTC) which had been designed to deliver the event time for the highest interaction rates. The collaboration is currently engaged in discussing and agreeing replacements. However, for the purpose of advancing the online systems, in the current document we still use the original PSD and BFTC numbers and their original geometries, with the reasonable assumption that the replacement systems will have some similar properties in terms of data rate and volume. The architecture of the proposed system is certainly flexible enough to accommodate changes in the payload data structures and rates.

Darmstadt, November 16, 2022

Chapter 1

The Compressed Baryonic Matter Experiment

1.1 Exploring the phase diagram of nuclear matter

Substantial experimental and theoretical efforts worldwide are devoted to the exploration of the phase diagram of nuclear matter. Figure 1.1 illustrates the possible phases of nuclear matter and their boundaries in a diagram of temperature versus baryon chemical potential. Cold nuclear matter – as found in normal nuclei with a net-baryon density equal to one – consists of protons and neutrons (i. e., nucleons) only. At moderate temperatures and densities, nucleons are excited to short-lived states (baryonic resonances) which decay by the emission of mesons. At higher temperatures, baryon-antibaryon pairs are also created. This mixture of baryons, anti-baryons and mesons, all strongly interacting

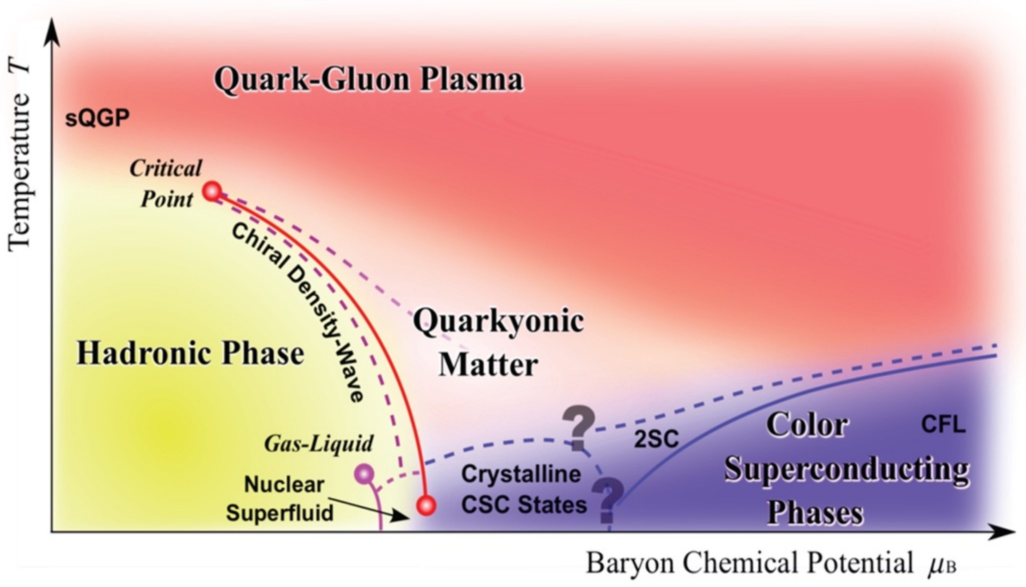


Figure 1.1: Sketch of the phase diagram of strongly-interacting matter (taken from [1])

particles, is generally called hadronic matter, or baryonic matter if baryons prevail. At very high temperatures or densities the hadrons melt, and their constituents, the quarks and gluons, form a new phase: the Quark-Gluon-Plasma (QGP). For very low net-baryon densities where the numbers of particles and anti-particles are approximately equal, Quantum Chromo-Dynamics (QCD) on the lattice predicts that hadrons dissolve into quarks and gluons above a temperature of about 155 MeV [2]. The inverse process happened in the universe during the first few microseconds after the big bang: the quarks and gluons were confined into hadrons. In this region of the phase diagram, the transition is expected to be a smooth crossover from partonic to hadronic matter [2]. Calculations suggest a critical endpoint at relatively large values of the baryon chemical potential [3]. Beyond this critical endpoint, at larger values of net-baryon densities (and for lower temperatures), one expects a first-order phase transition from hadronic to partonic matter with a phase coexistence region in between. A new phase of so called quarkyonic matter has been proposed to exist beyond the first order phase transition at large baryon chemical potentials and moderate temperatures [4]. High-density but cold nuclear matter is expected to exist in the core of neutron stars, and at very high densities correlated quark-quark pairs are predicted to form a color superconductor.

As illustrated in Figure 1.1, it is expected that the QCD phase diagram exhibits a rich structure at finite values of baryon chemical potentials, such as the critical point, the predicted first order phase transition between hadronic and partonic or quarkyonic matter, and the chiral phase transition. The experimental discovery of these prominent landmarks of the QCD phase diagram would be a major breakthrough in our understanding of the properties of nuclear matter. Of equal importance is the quantitative experimental information on the properties of hadrons in dense matter which may shed light on chiral symmetry restoration and the origin of hadron masses. In the laboratory, hot and dense nuclear matter is created at a wide range of temperatures and densities by colliding atomic nuclei at high energies.

The goal of the experiments at RHIC and LHC is to investigate the properties of deconfined QCD matter at very high temperatures and almost zero net-baryon densities. Several experimental programs are devoted to the exploration of the QCD phase diagram at high net-baryon densities. The STAR collaboration at RHIC scanned the beam energies and even conducted a fixed target program with the collider detector in order to search for the QCD critical endpoint [5, 6]. For the same reason, measurements are performed at the CERN-SPS with the upgraded NA49 detector (NA61) using light- and medium-sized ion beams [7, 8]. At the Joint Institute for Nuclear Research (JINR) in Dubna, a new heavy-ion collider (NICA) is being built with the similar goal to search for the high baryon density phase transition of nuclear matter [9]. However, due to luminosity or detector limitations, these experiments are constrained to the investigation of particles which are abundantly produced. In contrast, the Compressed Baryonic Matter (CBM) experiment at the Facility for Antiproton and Ion Research (FAIR) in Darmstadt is designed for precision measurements of multidimensional observables including particles with very low production cross sections using the high-intensity heavy-ion beams provided by the SIS100 accelerator.

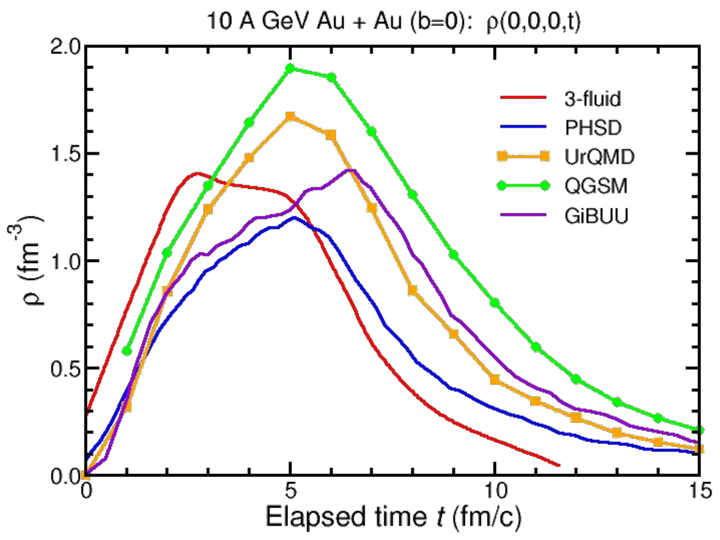


Figure 1.2: Baryon density as function of elapsed time for central Au+Au collisions calculated with different transport models [10]

The SIS100 accelerator at FAIR is very well suited to create high net-baryon densities. This is illustrated in Figure 1.2 which depicts results of transport code calculations for central Au+Au collisions. According to these calculations, densities beyond five times saturation density of $0.17\,\mathrm{fm}^{-3}$ can be produced already at beam energies of $10A\,\mathrm{GeV}$. Under these conditions the nucleons overlap, and theory predicts a transition to a mixed phase of baryons and quarks.

1.2 Diagnostic probes of the high-density fireball

Figure 1.3 depicts the evolution of a heavy-ion collision at FAIR energies as calculated with the UrQMD transport code [11], and illustrates the time of production and eventual emission of various particle species. Particles containing charm quarks are expected to be created in the very first stage of the reaction. Thus, D mesons and J/ψ mesons may serve as probes for the dense fireball and its degrees of freedom. Vector mesons like ω, ρ and ϕ mesons are produced continuously via $\pi\pi$ annihilation during the course of the reaction, and further decay either into mesons or into a pair of leptons. However, as leptons are not affected by final-state interactions, the dileptonic decay offers the possibility to look into the fireball. In particular, the short-lived ρ meson is a promising diagnostic probe of hot and dense nuclear matter. Also multi-strange hyperons and ϕ mesons carry information on the dense phase of the collision, in particular via their collective flow, due to their small hadronic cross sections. Finally, the bulk of the particles freezes out at densities below saturation density. To date, essentially only these bulk particles have been measured

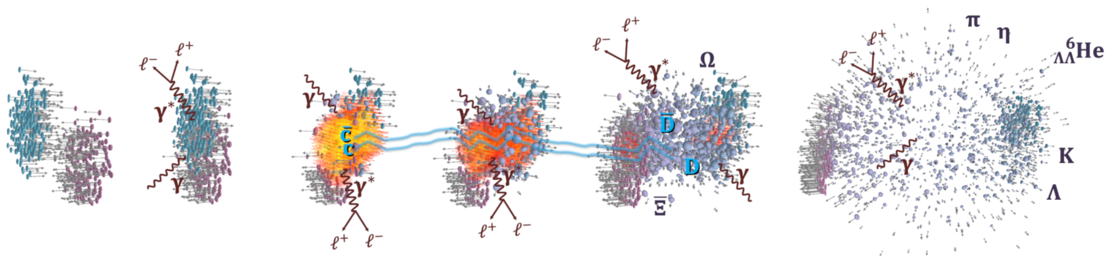


Figure 1.3: Au+Au collision at a laboratory beam energy of 10*A* GeV as calculated with the UrQMD model [11]: the initial stage where the two Lorentz-contracted nuclei overlap (left), the high density phase, and the final stage ("freeze-out") when all hadrons have been formed (right). Different particles are created in different stages of the collisions or escape from the interaction region at different times (see text). Almost 1000 charged particles are created in such a collision, most of them are pions. Figure credit: T. Galatyuk and F. Seck

in heavy-ion collisions at beam kinetic energies between 2 and 15*A* GeV (on stationary target).

Diagnostic probes of the dense stage of the fireball such as multi-strange baryons, dilepton pairs and charmed particles will be measured for the first time by the CBM experiment in this beam energy range. Therefore, the CBM experiment has a unique discovery potential.

The experimental challenge is to measure multi-differential observables and particles with very low production cross sections such as multi-strange (anti-)hyperons, lepton pairs, and particles with charm, with unprecedented precision. The situation is illustrated in the left panel of Figure 1.4 which depicts the multiplicities for various particle species produced in central Au+Au collisions as a function of the available energy in the center-of-mass frame. The data points are calculated using the thermal hadronization model based on the corresponding temperature and baryon-chemical potential [12]. Note that the dilepton decay of vector mesons, here illustrated for the ϕ meson, is suppressed by the square of the electromagnetic coupling constant $(1/137)^2$, resulting in a dilepton yield which is about six orders of magnitude below the pion yield, similar to the multiplicity of multi-strange anti-hyperons.

In order to produce high-statistics data even for the particles with the lowest production cross sections, the CBM experiment is designed to run at very high average reaction rates at current technological limits. For setups that incorporate the Micro Vertex Detector (MVD), this translates into a maximal average rate of 100 kHz. For measurements that do not require the ultimate vertex and tracking precision, an average interaction rate of up to 5 MHz can be realized, requiring the coverage of a peak interaction rate capability of 10 MHz for the expected beam parameters [14]. The rate capability of CBM exceeds the rate capabilities of other existing and planned heavy-ion experiments by orders of magnitude, as illustrated in the right panel of Figure 1.4.

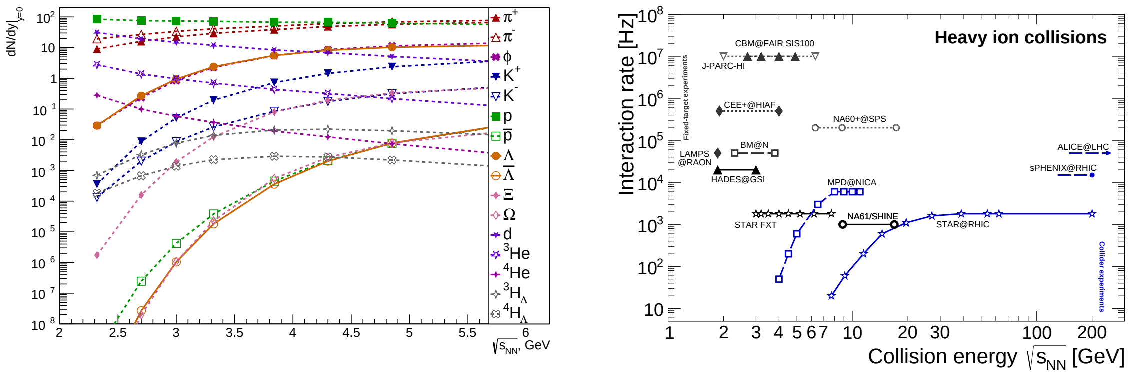


Figure 1.4: Left: Particle multiplicities for central Au+Au collisions as function of the available energy in the center-of-mass frame calculated with a statistical hadronization model [12]. Right: Interaction rates achieved by existing and planned heavy-ion experiments as a function of center-of-mass energy [13]. "STAR FXT" denotes the fixed-target operation of STAR.

1.3 Physics cases and observables

The CBM research program is focused on the following physics cases.

The equation-of-state of baryonic matter at neutron star densities

The relevant measurements are:

- The excitation function of the collective flow of hadrons which is driven by the pressure created in the early fireball.
- The excitation functions of multi-strange hyperon yields in Au+Au and C+C collisions at energies from 2 to 11A GeV. At sub-threshold energies, Ξ and Ω hyperons are produced in sequential collisions involving kaons and Λ, and are therefore sensitive to the density in the fireball.

In-medium properties of hadrons

The restoration of chiral symmetry in dense baryonic matter will modify the properties of hadrons. The relevant measurements are:

- The in-medium mass distribution of vector mesons decaying into lepton pairs in heavy-ion collisions at different energies (2 to 15A GeV), and for different collision systems. Leptons are penetrating probes carrying the information out of the dense fireball.

- Reference measurements of vector meson production in pN and pA collisions in order to separate in-medium effects from elementary production processes.
- Flow measurements of charged kaons in heavy-ion collisions.
- Yields and transverse mass distributions of charmed mesons in heavy-ion collisions as a function of collision energy.

Phase transitions from hadronic matter to quarkyonic or partonic matter at high net-baryon densities

In the SIS100 beam energy range densities beyond five times of the normal nuclear density are reached in central collisions between heavy-ions as indicated in Figure 1.2. A non-monotonous behavior in the excitation functions of sensitive observables would be indicative of a transition. The relevant measurements are:

- The excitation function of yields, spectra and collective flow of strange particles in heavy-ion collisions.
- The excitation function of yields and spectra of lepton pairs in the intermediate mass region in heavy-ion collisions.
- Event-by-event fluctuations of conserved quantities like baryon number, strangeness and net-charge number or proxys thereof in heavy-ion collisions measured with high precision as function of beam energy.

Hypernuclei, strange dibaryons and heavy multi-strange objects

Theoretical models predict that single and double hypernuclei, strange di-baryons and heavy multi-strange short-lived objects are produced in heavy-ion collisions with the maximum yield in the region of SIS100 energies. The planned measurements include:

- The decay chains of single and double hypernuclei in heavy-ion collisions.
- Search for strange matter in the form of strange di-baryons and heavy multi-strange short-lived objects. If these multi-strange particles decay into charged hadrons including hyperons they can be identified via their decay products.

Charm production mechanisms, charm propagation and in-medium properties of charmed particles in (dense) nuclear matter

The relevant measurements are:

- Cross sections and momentum spectra of open charm (D-mesons) in proton-nucleus collisions at SIS100 energies. In-medium properties of D-mesons can be derived from the transparency ratio $T_A = (\sigma_{pA} \rightarrow DX)/(A \times \sigma_{pN} \rightarrow DX)$ measured for different size target nuclei.

- Cross sections, momentum spectra and collective flow of charmonium (J/ψ) in proton-nucleus and nucleus-nucleus collisions at SIS100 energies.

The intended measurements at SIS100 including the results of simulations and count rate estimates are described in [15]. A general review of the physics of compressed baryonic matter, the theoretical concepts, the available experimental results and predictions for relevant observables in future heavy-ion collision experiments can be found in the CBM Physics Book [16].

1.4 The Facility for Antiproton and Ion Research (FAIR)

The international Facility for Antiproton and Ion Research (FAIR) [17] in Darmstadt will provide unique research opportunities in the fields of nuclear, hadron, atomic and plasma physics [18]. The research program devoted to the exploration of compressed baryonic matter will start with primary beams from the SIS100 synchrotron (protons up to 29 GeV, Au up to $11A$ GeV, nuclei with $Z/A = 0.5$ up to $14A$ GeV), available in the so-called FAIR Modularized Start Version (MSV) [17, Technical Document 1][19]. It might be continued with beams from a higher rigidity synchrotron, a possible upgrade to the FAIR project in the next phase of its operation. The layout of FAIR is presented in Figure 1.5. The beam extracted to the CBM cave reaches intensities up to 10^{11} protons and 10^{9} Au ions per second, with the following quality requirements: *i*) at a distance greater than 5 mm from the beam axis the beam halo is below 10^{-5} of the total beam intensity; *ii*) the intensity fluctuations of the spill structure is below 50 % (average value normalized to the maximum value), down to tens of nanosecond time scale [14].

1.5 Experimental setup

The CBM experimental strategy is to systematically perform both integral and differential measurements of almost all the particles produced in nuclear collisions (i. e., yields, phase-space distributions, correlations and fluctuations) with unprecedented precision and statistics. These measurements will be performed in nucleus-nucleus, proton-nucleus, and – for baseline determination – proton-proton collisions at different beam energies. The identification of multi-strange hyperons, hypernuclei, particles with charm quarks and vector mesons decaying into lepton pairs requires efficient background suppression and very high interaction rates. In order to select events containing those rare observables, the tracks of each collision have to be reconstructed and filtered online with respect to physical signatures. This concept represents a shift in paradigm for data taking in high-energy physics experiments: CBM will run without a hierarchical trigger system. Self-triggered readout electronics, a high-speed data processing and acquisition system, fast algorithms, and radiation hard detectors are indispensable prerequisites for successful operation of the experiment. Figure 1.6 depicts the CBM experimental setup for SIS100. The CBM experiment comprises the following components:

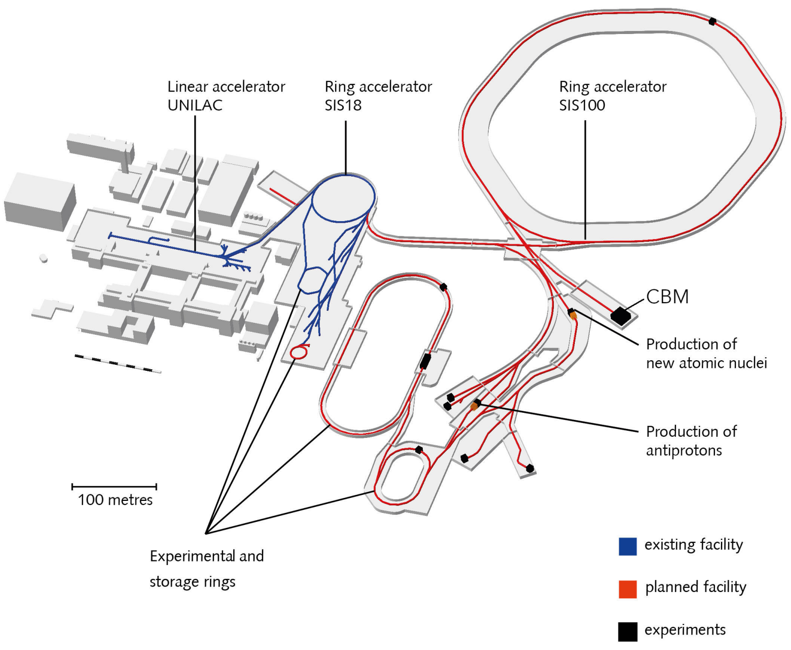


Figure 1.5: Layout of the Facility for Antiproton and Ion Research in Europe (FAIR)

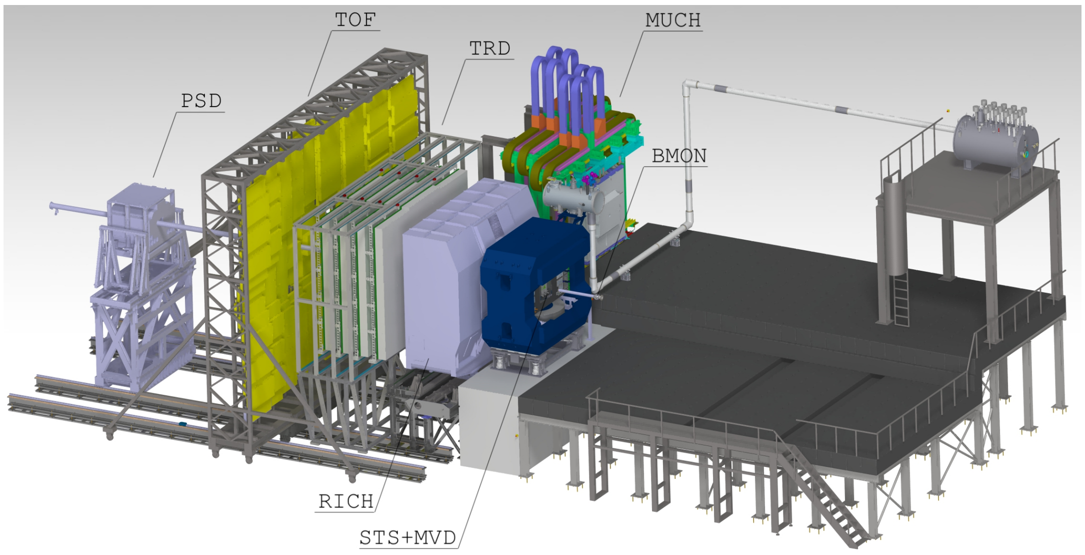


Figure 1.6: Drawing of the experimental setup of CBM

Dipole magnet

The CBM dipole provides a magnetic field integral of 1 T m needed to obtain a momentum resolution of $\Delta p/p < 2\,\%$ for track reconstruction at beam energies of the SIS100 synchrotron or its possible upgrade. The magnet is of the H-type with a warm iron yoke/pole and cylindrical superconducting coils. The wire has Nb-Ti filaments embedded in a copper matrix. The operating current and the maximal magnetic field in the coils are 686 A and 3.25 T, respectively. The magnet gap can accommodate the CBM tracking detectors with a vertical acceptance of $\pm 25°$ and a horizontal acceptance of $\pm 30°$. Further details are available in a corresponding TDR [20].

Micro-Vertex Detector (MVD)

The MVD will provide excellent spatial precision and low material budget as required for the identification of open charm particles and weakly decaying charged hyperons by the measurement of their displaced decay vertices. It consists of four planar stations equipped with thin and large-area Monolithic Active Pixel Sensor (MAPS) chips. The system layout can be adopted to the needs of a specific run, i. e., optimized w. r. t. vertexing (VX) or tracking (TR) capability, respectively. For the VX (TR) detector configuration, they are located in a vacuum from 5(8) cm to 20 cm downstream of the target. The VX detector geometry intends to achieve a precision for the determination of secondary vertices of about 50 µm to 100 µm along the beam axis. Further details are available in a corresponding TDR [21].

Silicon Tracking System (STS)

The STS is the main tracking device of the CBM experiment capable of providing track reconstruction and momentum determination of charged particles. The system comprises 8 detection layers equipped with double-sided silicon micro-strip sensors. The sensors are mounted onto lightweight mechanical support ladders and read out through multi-line micro-cables with fast self-triggering electronics at the periphery of the stations where cooling lines and other infrastructure can be placed. The MVD and STS will determine the tracks of charged particles inside the magnetic field over the length of 1 m downstream of the target. Further details are available in a corresponding TDR [22].

Ring Imaging Cherenkov Detector (RICH)

The RICH detector will provide the identification of electrons via the measurement of their Cherenkov radiation. This will be achieved using a gaseous RICH detector built in a standard projective geometry with focusing mirror elements and a photon detector. The detector will be positioned behind the dipole magnet about 1.6 m downstream of the target. It will consist of a 1.7 m long gas radiator (overall length approximately 2 m) and two arrays of mirrors and photon detector planes. The design of the photon detector plane is based on MAPMTs in order to provide high granularity, high geometrical acceptance, high detection efficiency of photons also in the near UV region and a reliable operation. Further details are available in a corresponding TDR [23].

Muon Chamber System (MUCH)

The concept of the muon detection system is to track the particles through a hadron absorber and thus perform a momentum dependent muon identification. The absorber/detector system is placed downstream of the STS, which determines the particle momentum. In order to reduce the detection of meson decays into muons the absorber/detector system is designed to be as compact as possible. It consists of four hadron absorbers made of iron and 12 gaseous tracking chamber layers arranged as triplets behind each iron slab. A fifth absorber can be installed in addition, between the last triplet and the Transition Radiation Detector. The setup allows for efficient event selection through the measurement of short track segments in the last tracking station triplet, and extrapolation of these tracks to the target. For J/ψ measurements at SIS100, a MUCH start version with three chamber triplets is sufficient. Further details are available in a corresponding TDR [24].

Transition Radiation Detector (TRD)

The Transition Radiation Detector, consisting of four detector layers grouped into one tracking station, will serve for particle tracking and for the identification of electrons and positrons with $p > 1.0\,\mathrm{GeV}/c$ ($\gamma \geq 1000$). The detector layers are located at approximately 4.1 m to 6.2 m downstream of the target, the total active detector area amounts to about $114\,\mathrm{m}^2$. The TRD readout will be realized in rectangular pads giving a resolution of $\sim$300 µm across and 3 mm to 30 mm along the pad. Every second TRD layer is rotated by 90°. Further details are available in a corresponding TDR [25]. For the inner part of the TRD detector a novel design with 2D spatial resolution is proposed.

Time-Of-Flight System (TOF)

An array of Multi-gap Resistive Plate Chambers (MRPC) will be used for hadron identification via TOF measurements. The TOF wall covers an active area of about $100\,\mathrm{m}^2$ and is flexibly located about 6 m to 10 m downstream of the target allowing to optimize especially the kaon reconstruction efficiency and purity over the full SIS100 energy range. The required time resolution is in the order of 80 ps. At small deflection angles close to the beampipe the pad size is about $5\,\mathrm{cm}^2$ corresponding to an occupancy below 5 % for central Au+Au collisions at $10A$ GeV. This area will be covered by the BFTC detector which is foreseen to supply the event time for the highest interaction rates from the projectile spectator fragments propagating with beam velocity. It is followed at larger polar angles with decreasing granularity by the inner and outer ToF modules, however, always preserving the maximal occupancy of a given readout channel of 5 %. Further details are available in a corresponding TDR [26].

Projectile Spectator Detector (PSD)

The PSD will be used to determine the collision centrality and the orientation of the reaction plane. The detector is designed to measure the number of non-interacting nucleons from a projectile nucleus in nucleus-nucleus collisions. The PSD is a fully compensating modular lead-scintillator calorimeter which provides very good and uniform energy resolution. The calorimeter comprises 44 individual modules, each consisting of 60 lead/scintillator layers. Further details are available in a corresponding TDR [27].

Beam monitor (BMON)

The BMON subsystem consists of two diamond based beam detector stations located in front (upstream) of the target chamber. The T0 station is foreseen to measure at moderate rates the start time of the reaction with a precision in the order of 50 ps. The HALO station will be used for beam monitoring, i. e., beam halo measurement.

Data acquisition and online event selection

High-statistics measurements of particles with very small production cross section require high reaction rates. The CBM detectors, the data acquisition and online event selection systems will be designed for peak event rates of 10 MHz, corresponding to a beam intensity of 10^9 ions/s and a 1 % interaction target, for example. The rate and amount of archived data, however, are limited by the archiving bandwidth and the costs of storage media to about 10^5 events per second. Therefore, measurements with peak event rates of up to 10 MHz require highly selective algorithms, which suppress background events (that contain no signal) by a factor of 100 or more in real time. The event selection system will be based on fast online data reconstruction running on a high-performance computer farm equipped with many-core CPUs and GPUs (located in the GSI Green IT Cube). Data acquisition and transport (DAQ) are described in this TDR, online event selection and other software systems will be described in a forthcoming TDR.

The mCBM experiment – a CBM full-system test setup

In order to verify the challenging requirements on the various detector subsystems and especially on the data acquisition and reconstruction system, the CBM collaboration has initiated the installation of a full-system test setup. With the mCBM setup [28], a full slice of the CBM detector with all of its detector subsystems can be evaluated concerning detector stability at particle and data rates up to the anticipated nominal SIS100 rates. In addition, the performance of algorithms for event reconstruction when exposed to real data can be inspected and optimized. The setup is operational since end of 2018 and delivered valuable performance numbers [29] which are partially used in the following chapters. A detailed description of mCBM and its results is presented in Chapter 7.

Chapter 2

Online System Overview

2.1 Overall architecture and scope of this document

A key design feature of the CBM experiment is its ability to measure at interaction rates of up to 10^7 events per second, which is to-date unprecedentedly high in nuclear collision experiments. The operation of the experiment at such high rates will give access to very rare probes, such as multi-strange anti-hyperons, hyper-nuclei or charmed hadrons. Naturally, this requirement sets strong constraints on the design of the detectors and their readout electronics, but also, and in particular, to the data acquisition system. Considering the typical average raw event size for minimum-bias Au+Au collisions of about 50 kB (cf. Sec. 3.3.1), a peak collision rate of 10 MHz leads to an instantaneous raw data rate of about 500 GB/s. Storage of this full raw data rate is prohibitively expensive, not only because of the necessary storage bandwidth, but also and predominantly because of the cost of the storage media.

This means that the raw data rate has to be reduced by at least two orders of magnitude as a real-time process before writing to persistent media. Such data reductions can only be achieved by the inspection of events w. r. t. physical signatures specific to the rare probes. Typical trigger signatures are e. g., off-target decays of hyperons, the detection of which requires at least partial event reconstruction up to track level. Such complex triggers cannot be realized in conventional trigger logic, nor are they well suited for FPGAs; they must be evaluated in software on CPUs and/or GPUs. In addition, the nature of heavy-ion collisions with several hundreds of charged tracks being produced per event does not allow the achievement of significant data reduction by simple low-level triggers implemented in trigger hardware, meaning that the full data rate has to be forwarded for inspection in software.

These conditions have led to the concept of the CBM data acquisition as a full free-streaming system without any hardware trigger. The readout electronics of all detector systems are autonomous and self-triggered; they create and push time-stamped hit messages on activation of analog readout channels above pre-defined thresholds. The task of the online systems is to collect, aggregate and deliver this data to the online compute farm, where event reconstruction and inspection up to the trigger decision is performed. The software trigger decides whether data are forwarded to storage; no trigger information is passed backwards in the readout chain. Thus, the system is not limited by latency, but by data throughput only.

A self-triggered readout system implies that the association of data from different detector elements to their physical collision event must be based on their timestamp alone, which is created in the front-end electronics (FEE). Hence, the FEE elements must be synchronized to sub-nanosecond precision by a central timing system. The classical "event building" task, as well as the actual high-level trigger decision, are shifted to the online compute farm FLES ("First-level Event Selector"), to which the readout hardware is connected via custom-developed optical links that handle clock and time distribution, data transfer, and control communication. The links are interfaced to the online farm with a custom PCIe card, the Common Readout Interface (CRI). On the one hand, the CRI forwards the clock and time information received from the Timing and Fast Control (TFC) system to the detector FEE[1], and on the other hand, it reformats the data received from the detector FEE into a form suitable for processing in the FLES.

It is evident that the online data selection task requires a substantial amount of computing power. Our current estimates amount to about 1 M HepSpec06 [30] at the highest interaction rates. The online computing capacity is, however, needed only during the experiment runtime, which will be a few months per year (cf. Sec. 3.2.1). This operation scenario combined with the need for custom links to the experiment electronics motivates the splitting of the compute farm into two parts, which are in anyway physically separated. The FLES entry nodes are commodity-hardware computers hosting the CRI interface cards to the detectors and perform "event building" in the sense that data containers ("timeslice components (TSCs)") comprising the entire experiment data in a given time interval are assembled. The input stage is data agnostic, meaning that data units are not modified or even inspected except for their timestamps. The entry nodes will be located in the server room in the CBM building, above the experimental area and outside of the radiation zone. Their number is primarily determined by connectivity to the experiment. In terms of the budgetary and operational responsibility, they are owned resources of the CBM collaboration.

Online data inspection and selection are performed on the processing nodes located in the Green IT Cube building, several hundreds of meters distant from the experimental area (Fig. 2.1), resulting in an optical fiber link length of about 1 km. Timeslice components assembled by the entry nodes are transferred over an InfiniBand network to the processing nodes, which are commodity-hardware, shared resources in the operational responsibility of FAIR-IT. A service-level agreement with FAIR will ensure that the needed resources are supplied exclusively for CBM during the experiment run times, while they can be used for other purposes between data-taking periods.

The hardware building blocks of the CBM readout systems are schematically depicted in Figure 2.2, grouped according to their physical location in the CBM experiment area, the CBM server room, and the GSI Green IT Cube. The location of the components roughly maps also organizational aspects: while FEE and parts of the CRI FPGA design

[1]See Sections 2.2.3 and 2.2.4 for this fundamental design decision.

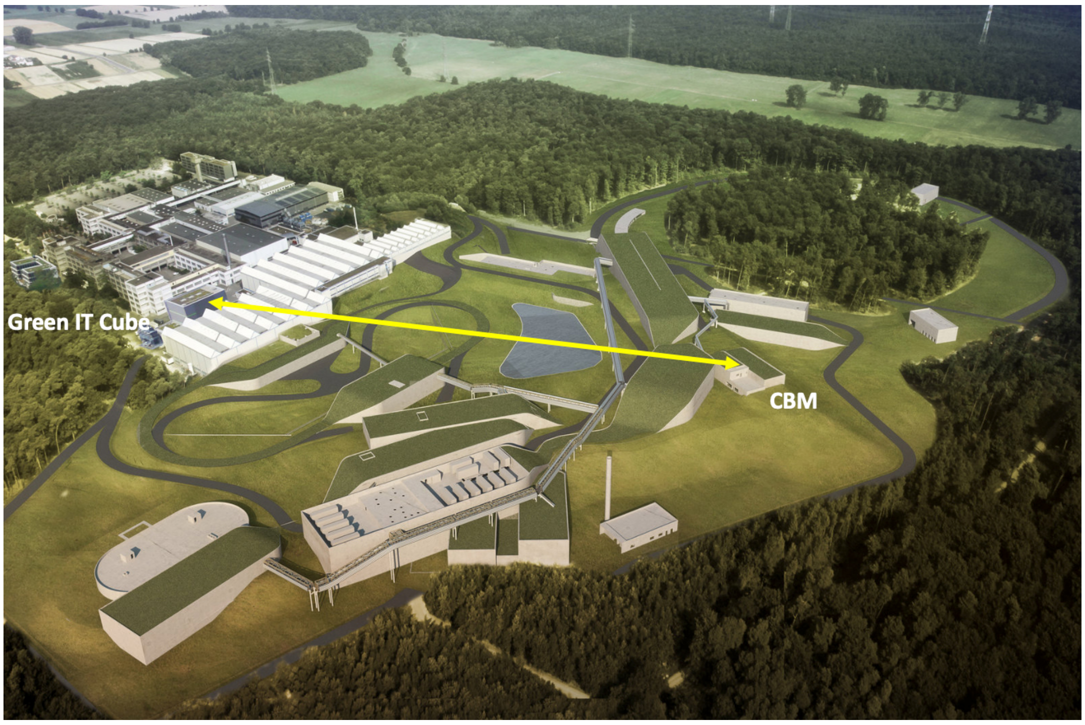


Figure 2.1: An aerial-perspective drawing of the completed FAIR campus as planned, showing the CBM experiment where our data will be sourced, and the Green IT Cube where the online data will be processed.

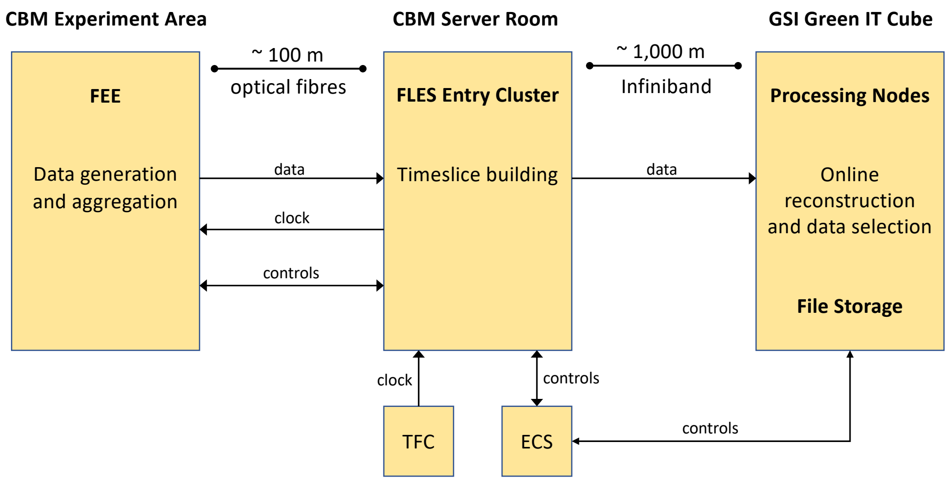


Figure 2.2: Schematic layout of the CBM readout system. FEEs are located at the detectors in the experimental area. The FLES entry cluster is hosted in the CBM server room. The processing nodes for online data processing and selection are shared GSI resources and are placed inside the Green IT Cube.

are subsystem-specific and thus in the organizational responsibility of the detector groups, the FLES entry cluster and the software system for timeslice building are central CBM tasks, as well as the control software framework, the TFC, and the InfiniBand connection to the processing nodes. Hardware resources in the GSI Green IT Cube (both computing and storage cluster) are procured and operated by GSI/FAIR and provided as a service to CBM during the experiment operation.

A complete description of the CBM online system must include details of both the involved hardware and the software systems. From a budgetary point of view, the two are quite distinct. The procurement of the required hardware will naturally come with substantial investment costs, which have to be elaborated and properly funded. Software developments, on the other hand, mainly consume workforce and entail little or no investment costs, which also implies that the timeline for the final design decisions is different compared to that for hardware development and procurement. These considerations have motivated splitting the Technical Design Report for the CBM online systems into two parts. The first part—this document—describes the general architecture with a focus on systems coming with substantial investment costs. It includes the First-level Event Selector FLES entry stage (Chapter 4), the Common Readout Interface card CRI (Chapter 5), and the Timing and Fast Control System TFC (Chapter 6). The forthcoming second part of the TDR will describe the software systems used for online data analysis and the operation of the experiment.

2.2 Readout architecture

2.2.1 Front-end electronics concept

The first stage of the detector readout is the analog processing and digitization of the detector signals. For triggered systems, the FEE would be guided by a trigger signal and this signal would determine at which time the analog signal is to be inspected, would serve as a reference for time measurements, and would help to organize the output data. The DAQ system would then request the data from the front-end electronics and would combine all data of a trigger to form an *event*.

In a self-triggered system such as is planned for CBM, the FEE has no such guidance. It inspects the analog signals and registers a *hit* when the detector signal is above a threshold. The only concept available to organize the data is that of *time*. Therefore, the data of each hit includes a timestamp. The FEE sends a continuous stream of hit data, usually in the form of hit messages. The DAQ system continuously receives all front-end data streams and packages the hits based on their timestamps into containers. This packaging is done in three stages: the CRI hardware generates a stream of microslices (cf. Sec. 4.2.1), from which the FLES software first generates timeslice components (cf. Sec. 4.2.2) and finally complete *timeslices* (cf. Sec. 4.4).

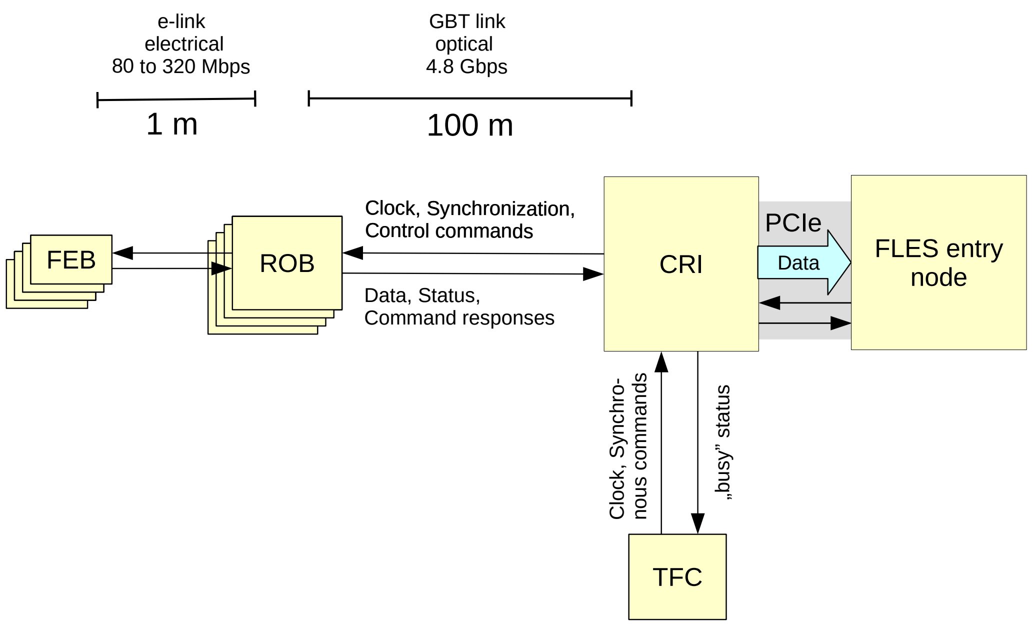


Figure 2.3: Schematic layout of the readout building blocks for one FLES entry node. An entry node can host more than one CRI. A CRI serves multiple ROBs, and a ROB usually serves multiple FEBs. The TFC is a central system common to all CRIs in CBM.

2.2.2 Data aggregation concept

Figure 2.3 shows the schematic layout of the readout building blocks for one FLES entry node. The signals of most of the CBM detector systems are digitized by radiation-tolerant ASICs. They are usually physically grouped on "Front-End Boards" (FEBs) which are mounted on the detector near the signal source. At the next stage, the data created by a group of FEBs are aggregated and sent over optical links from the detector to the FLES entry cluster for all further processing. The aggregation stages, often called "ReadOut Boards" (ROBs), are mounted on the detector near the FEBs they serve and are therefore also in a radiation environment. The components developed for the GBT project [31] perfectly match the CBM requirements for this aggregation stage and provide

- fully synchronous operation of the whole readout chain allowing clock distribution and messages with deterministic latency (useful for time synchronization),
- the GBTx [32], a radiation-hard concentrator ASIC,
- with the e-link concept [33] a robust electrical interface towards the front-end ASICs based on differential signals (SLVS and LVDS compatible),
- with the Versatile Link [34] radiation-hard optical transceivers for bi-directional (VTRx) and dual uni-directional (VTTx) links,

- and last but not least with GBT-FPGA [35] the essential FPGA design building blocks for system integration.

Based on this assessment, CBM decided in 2014 to use GBT and Versatile Link components in the data aggregation stage of most of the CBM detector systems. The procurement from CERN was started immediately after this decision to participate in the joint production for the LS2 upgrades of the LHC experiments.

The GBTx uses a custom protocol on the optical link. Therefore, the links are handled with a custom FPGA-based board named "Common Readout Interface" (CRI). The CRI board has a PCIe form factor, terminates the GBT links and interfaces them to the FLES entry node.

2.2.3 Flow of data and controls

A key design decision for the readout concept was the handling of the control channel for the front-end ASICs. The choice was driven by the STS, the most expensive and in terms of optical link count the largest CBM detector subsystem. The STS uses double-sided silicon strip detectors. The readout ASICs are operated with a local reference potential of $\pm$ half of the sensor bias voltage. The consequence is that each STS FEB is operated at its local reference potential, while the ROBs, which serve multiple FEBs, operate at the detector ground potential. A design goal of the STS FEE was to minimize the number of connections that cross a ground domain. This led to a design of the STS readout ASIC SMX that combines controls and data traffic over an AC-coupled e-link. This design concept was later adopted by other CBM detector subsystems.

Each ROB carries one or more GBTx that aggregate the e-link data streams onto optical links using the GBT frame protocol (referred to as GBT links). Because controls and data traffic is combined at the e-link level it is also combined on the GBT links. The CRI separates the two traffic classes. The hit data is transferred over a uni-directional high-bandwidth DMA channel to the FLES entry node, while the controls traffic is handled via a PIO interface.

2.2.4 Flow of clock and time

The concept of time is ubiquitous in a self-triggered DAQ system and manifests itself in the form of time counters, which provide a local time representation, and timestamps, which carry time as a property of an object. The front-end electronics in a self-triggered readout system adds a timestamp to each hit it digitizes. This timestamp typically reflects the point in time when the signal crosses the detection threshold and is determined at that time by reading the value of a local time counter. For a coherent operation, it is essential that all time counters in the system are incremented by frequency and phase-locked clocks (clock distribution) and that all time counters are properly synchronized (time distribution).

A key design decision was the choice of the base clock frequency from which all other clock frequencies are derived. The GBTx was designed for LHC experiments and it was natural to choose the nominal LHC bunch crossing frequency of $f_{\mathrm{LHC}} = 40.0789\,\mathrm{MHz}$ as the base clock rate from which all other frequencies are derived. This base frequency is deeply built into the hardware as the GBTx uses a VCXO with a very limited tuning range and the crystal is embedded in the GBTx ASIC package. The frequency f_{LHC} is of course an unusual choice outside of the LHC environment. The GBTx packaged for CBM has therefore a 40.0000 MHz reference crystal, and 40 MHz is the base clock rate from which all other frequencies in the CBM readout are derived.

The TFC system serves as the central clock and time master for the CBM readout tree. It is based on optical links which are operated with deterministic latency in the downlink (master to front end) direction.

Clock distribution The TFC distributes a 40 MHz master clock to all CRIs. This is implemented via clock recovery from the TFC downlink. The CRIs use this clock to operate all GBT links. The GBTx in the ROBs recover the 40 MHz clock from the link and generate frequency-locked and phase-controlled e-link clocks running at 40, 80, or 160 MHz. The readout ASICs in turn derive all internal clocks from the e-link clocks.

Time distribution The TFC distributes a master time to all CRIs. This is implemented via messages with deterministic latency (DLMs) sent out by the TFC master and used in each CRI to synchronize a local time counter. The GBT links are operated with deterministic latency in downlink direction[2]. This allows the implementation of mechanisms that synchronize the local time counters in the front-end electronics with the local time counter in the CRI and in turn with the TFC master time.

2.2.5 Time representation

The TOF system measures time with a binning of 50 ps, whilst periods of uninterrupted data-taking, which we colloquially refer to as *runs*, can last hours. It is impractical to use a unified time representation that supports the highest resolution used and is unique across hours[3]. Instead, different parts of the system use resource- or data-size-optimized time representations that can be converted to a unique absolute time in later processing.

The local time counters in the TFC system and the CRIs, as well as the timestamps of data containers sent to the FLES system, have a length of 64 bit, which is sufficient to represent the absolute time with the resolution of the system clock and never overflow during the lifetime of the experiment. For the local time counters in the FEE ASICs and the timestamps of the hit messages, such a length is neither practical nor necessary. In this domain, compact hit messages and good uplink bandwidth utilization are key design

[2]latency optimized mode in GBT-FPGA lingua

[3]50 ps resolution over 1 hour would require a 46-bit representation

criteria. The hit timestamps are short and unique only on a time scale of typically 10 µs. The size of the hit timestamp divides the timeline into epochs. So-called epoch messages are inserted into the uplink data stream at epoch boundaries. They often carry additional bits of the local time counter. The required length of the FEE local time counter is given by the timestamp length plus the number of epoch count bits carried in the epoch messages.

This concept leads to compact uplink data formats and also compact hardware designs. However, one consequence of this is that the interpretation of hit messages becomes context dependent. This adds some additional data processing complexity. For each uplink, the hit messages must be interpreted in the context given by the last received epoch message. When uplinks are aggregated, redundant epoch messages can be removed. The hit message is usually reformatted, with slightly expanded timestamps and reorganized channel and link addressing information optimized for further processing. When the aggregated data stream is packed into data containers, new epoch messages must be inserted at the beginning to ensure that the data in the container are self-describing.

Chapter 3

Requirements and Constraints

3.1 Introduction

Realization of the CBM experiment comes with substantial demands for data acquisition and for online processing. To plan and design an optimally functioning experiment, prior knowledge of their requirements is needed. The current chapter documents our best estimation on CBM data acquisition needs for operation at SIS100. The numbers are substantiated with experience gathered from the mCBM full-system test [28, 29], which has been on-going since August 2018. However, it must be cautioned that many issues remain uncertain, both concerning CBM design features as well as operating conditions not directly influenced by CBM. The numbers should be interpreted as estimates only.

The resource estimates were derived for full CBM operation. It is envisaged that CBM will start at Day-1 with interaction rates less than its highest design capacity, with the maximum interaction rates reached only after an initial learning and verification period. Staging scenarios, where applicable to the data acquisition system, will be described later in this document.

3.2 Operation of the CBM experiment

3.2.1 Annual run time and beam conditions

For the assessment of the CBM data acquisition and computing requirements, the following assumptions on the operating conditions are made:

1. The SIS100 machine is planned to be operated up to 6000 h (250 days) per year [36]. Accelerator-wise, slow extraction to fixed-target experiments can be efficiently combined with fast extraction to storage-ring experiments in a super-cycle. We assume a beam directed into the CBM cave for 100 days, hence 2400 h or 8.6×10^6 s.

2. During CBM beam times, the experiment run time is estimated to be 80 %, which amounts to 1920 h or 6.9×10^6 s per year.

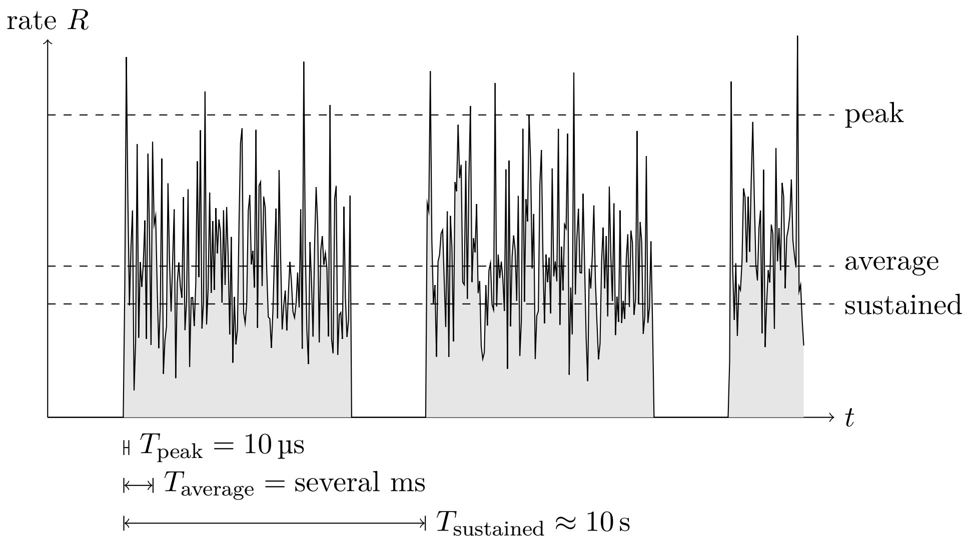

rate R
peak
average
sustained
t
$T_{\text{peak}} = 10\,\mu\text{s}$
$T_{\text{average}} =$ several ms
$T_{\text{sustained}} \approx 10\,\text{s}$

Figure 3.1: Schematic representation of the interaction rate over time (not to scale). Since the beam intensity fluctuates over time, a rate is always specified depending on an associated integration interval T.

3. An effective duty cycle of 75 % seems a realistic operation scenario, even when CBM needs to share the accelerator resource with another experiment operating in tandem using a fast extraction [14] mode. This gives 5.2×10^6 s beam on CBM target per year.
4. There are no predictions for the in-spill beam intensity variations to be expected from SIS100. CBM and HADES require the variations to be less than two (peak to average) [14]. For the resource estimates in this document, we assume a peak-to-average ratio of two. It should be noted that this means a substantial improvement of the beam quality with respect to SIS18.

3.2.2 Interaction rates and CBM setups

In regards to the interaction or data rates, the following terminology is used (cf. Fig. 3.1):

- Peak rate: the maximum instantaneous rate, averaged over a time window of 10 µs;
- Average rate: the mean in-spill rate, averaged over beam intensity variations on a time scale of ms;
- Sustained rate: the mean data-taking rate during operation, averaged over the machine duty cycle on a time scale of 10 s.

For the calculations in this document we assume:

$$\frac{R_{\text{peak}}}{R_{\text{average}}} = 2; \qquad \frac{R_{\text{sustained}}}{R_{\text{average}}} = \frac{3}{4}.$$

The CBM detector systems, except for the Micro Vertex Detector (MVD) and the Projectile Spectator Detector (PSD), are designed for a peak interaction rate of 10^7/s for minimum-bias Au+Au collisions at $p_{\text{beam}} = 12A\,\text{GeV}/c$. In Au+Au collisions, the MVD performance is limited by the load from delta electrons, produced by beam particles in the target, to an average beam intensity of 10^7/s, corresponding to an average interaction rate of 10^5/s for a 1 % target. The PSD is limited by the time constant of the wavelength shifters and photo-detectors to a peak interaction rate of 10^6/s.

CBM is designed to operate with flexible combinations of detector systems ("setups") serving different physics objectives. For this document, we consider the following setups, which are decisive for the readout and online processing requirements:

- Hadron setup: it comprises the Silicon Tracking System (STS), the Transition Radiation Detector (TRD), and the Time-of-Flight System (TOF). This setup gives access to hadronic probes like, e. g., multi-strange hyperons or hyper-nuclei. It can be operated at the highest interaction rates. Adding the PSD for the determination of the event plane (e. g., for flow measurements) limits the operation to a peak interaction rate of 10^6/s.
- Electron setup: it comprises the Hadron setup plus the MVD, the Ring-Imaging Cherenkov Detector (RICH), and the PSD. In this setup, both hadronic probes and electron pairs can be measured simultaneously. The MVD limits the operation of this setup to average interaction rates of 10^5/s.
- Muon setup: it comprises the STS, the Muon Detection System (MUCH), the TRD, and the TOF. This setup is exclusively used to measure muon pairs. It can be operated at the highest interaction rates.

The CBM beam times will be allocated to the setups according to the physics priorities agreed upon prior to each data-taking campaign. Switching between the setups necessitates moving or exchanging of detector subsystems.

3.2.3 Online data flow

Raw data delivered from the detector front-end electronics are aggregated by a number of different readout chains and streamed to the FLES entry cluster, a set of server computers equipped with custom FPGA interface cards ("CRI"). The DAQ system is designed to cope with a maximum instantaneous interaction rate of 10^7/s (Au+Au at $p_{\text{beam}} = 12A\,\text{GeV}/c$). The DAQ hardware and the FLES entry cluster will buffer in-spill beam intensity variations; the average over the duty cycle can be established in the entry cluster or in the compute cluster.

The size of the entry cluster is determined by bandwidth and connectivity to the experiment rather than by computing power. It will be located close to the experiment; the responsibility for its design and operation is with CBM. The entry cluster is connected via long-range InfiniBand to the compute cluster located in the Green IT Cube. The Flesnet

	hadron setup event size / kB		electron setup event size / kB		muon setup event size / kB		dark rate / MB/s	
	CRI	FLES	CRI	FLES	CRI	FLES	CRI	FLES
BMON	–	–	0.8	1.6	–	–	1	0
MVD	–	–	15.0	15.0	–	–	3 456	3 456
STS	18.4	18.9	18.4	18.9	18.4	18.9	7 500	7 160
MUCH	–	–	–	–	4.5	6.0	6 524	7 460
RICH	–	–	8.4	8.4	–	–	792	792
TRD	30.5	40.7	37.2	49.6	3.1	4.1	7 100	4 339
TOF	6.0	8.5	6.5	9.2	1.4	2.0	3 481	74
PSD	–	–	2.6	2.6	–	–	2	2
Sum	54.9	68.1	89.0	105.4	27.4	31.0	28 873	23 282

Table 3.1: Raw data average event sizes and dark rates sent to CRI and FLES stages for minimum-bias Au+Au collisions at $p_{\text{beam}} = 12A\,\text{GeV}/c$. The numbers include both raw data messages and overhead.

software running on entry and processing nodes aggregates raw data into timeslices, which are delivered to the processing nodes for processing in real-time. The role of online data processing is the selection of a subset of the raw data for permanent storage.

3.3 Data rates from detectors

3.3.1 Raw data event sizes and detector dark rates

The raw data event sizes were evaluated by simulation of minimum-bias Au+Au collisions at $p_{\text{beam}} = 12A\,\text{GeV}/c$ generated by UrQMD. The simulation yields the number of raw detector hits ("messages") per event for each detector system. This number is weighted by both the single-message size according to the current design of the front-end ASICs to determine the event size as delivered by the detector hardware to the CRI, and the single-message size within a FLES timeslice as delivered by the CRI design component of the detector to the FLES entry cluster.

The numbers include all anticipated background and overhead components that depend on the interaction rate. Where applicable, the effect of beam particles has been included in the average event sizes as a sum, assuming a beam/target interaction probability of 1 %. The dark rate numbers of the detectors include both the anticipated data from noise and digital readout overhead such as periodic epoch messages.

The data values are given in bytes, with the physical link bit rates being slightly higher due to the overhead introduced by the encoding of the links (e. g., from 8 bit to 10 bit). The overhead introduced by the microslice and timeslice containers is small (cf. Sec. 4.2.4) and not considered here.

	CRI	MB/s per CRI	
	count	average	maximum
BMON	1	162	162
MVD	5	991	991
STS	72	1 413	1 696
MUCH	33	1 130	1 469
RICH	8	204	204
TRD	68	3 054	3 970
TOF	25	1 708	2 220
PSD	1	266	266
Total	213	1 741	3 970

Table 3.2: Subsystem connectivity and required CRI bandwidths for the highest-rate setups in which each respective subsystem is used.

Table 3.1 summarizes our estimations. Individual contributions are discussed with further details in Appendix A. The event sizes given in Table 3.1 correspond to the largest collision system for CBM at SIS100 (Au+Au at $p_{\text{beam}} = 12A\,\text{GeV}/c$). The lower the collision energy or the smaller the number of collisions leads to smaller raw data events. For instance, the event size for Au+Au collisions at $p_{\text{beam}} = 4A\,\text{GeV}/c$ is 70 % of that of top SIS100 energy.

3.3.2 Raw data rate to the FLES stage

Table 3.2 summarizes the bandwidth requirements per CRI of the subsystems for the most data-intensive of the three setups and run conditions. Note that the RICH is designed for higher interaction rates in anticipation of a potential future MVD upgrade, so the load on its CRIs in the electron setup is modest compared to the other detectors. The PSD and beam monitor (BMON) subsystems are each assigned one CRI, although they do not fully utilize it, in order not to mix data from different systems on a single CRI.

As the layouts of the readout trees of the detectors have to follow geometrical and practical constraints, the distribution of the overall data rate of each subsystem across its CRIs will not be even. For the STS, the factor between data rate of the most active CRI and the average CRI data rate has been simulated to be approximately 1.2. For most of the other subsystems, this factor is estimated to be 1.3. With an additional factor-two safety margin, the maximum bandwidth required at the CRI/FLES interface is approximately 8 GB/s. The total number of CRIs will be around 200 (cf. Sec. 5.2.4).

3.3.3 Total raw data rate

Table 3.3 shows the data rates from the detectors to the FLES entry cluster for the different setups and run conditions. The “dark” column summarizes the data rate that is expected

setup avg. int. rate/1/s	hadron 5×10^6	electron 1×10^5	muon 5×10^6	dark 0
	GB/s	GB/s	GB/s	GB/s
BMON	–	0.2	–	0.0
MVD	–	5.0	–	3.5
STS	101.8	9.1	101.8	7.2
MUCH	–	–	37.3	7.5
RICH	–	1.6	–	0.8
TRD	207.6	9.3	24.8	4.3
TOF	42.7	1.0	9.9	0.1
PSD	–	0.3	–	0.0
Sum	352.1	26.4	173.9	23.3

Table 3.3: Average total data rates sent from the detectors to the FLES stage. The stated rates include raw data messages and overhead.

during commissioning without beam. The numbers follow from the considerations of Sections 3.2.2 and 3.3.1.

If averaging over the machine duty cycle is assumed to not happen before the Green IT Cube, the average in-spill interaction rates have to be applied, so these numbers describe the relevant requirements for both the FLES timeslice building operation and the transmission to the online processing in the Green IT Cube.

As the highest bandwidth requirement exists in the hadron setup, an upper limit estimation for the total data rate is 400 GB/s. This rate scales linearly with the average interaction rate (5×10^6/s in the hadron setup) and with the raw data event size (see Table 3.1). Applying a contingency factor of 1.5, we arrive at a minimum bandwidth requirement of 600 GB/s.

Chapter 4

First-level Event Selector

The First-level Event Selector (FLES) is the central data handling and event selection entity of the CBM experiment. It is the endmost part in the CBM readout chain. The FLES implements the interface to the detector readout, combines all data, and performs the global event selection by performing a full online reconstruction and topological analysis. The basic design is similar to a traditional high-level software trigger, as it is employed in virtually all modern high-energy physics experiments. However, the exclusive, software-only event selection approach of CBM adds distinct challenges not found in those classical systems.

This chapter describes the FLES entry stage which is defined as the data path between the interface to the subsystem readout chains and the interface to the online analysis.

4.1 Architecture

The FLES is designed as an HPC cluster. It will be built from both COTS and custom components. The scalable architecture is specially laid out for achieving the required throughput of the incoming experiment data with sufficient redundancy and for providing the necessary computational efficiency for flexible real-time data processing. The design goal is to be able to scale to input data rates exceeding 1 TB/s. The conceptual design of the FLES is shown schematically in Figure 4.1. The FLES cluster can be divided into two sub-clusters: An *entry node* portion located close to the experiment in the CBM service building and a *processing node* portion located at the central Green IT Cube data center at GSI approximately 1 km of cabling distance away from the experiment. Both portions share a common RDMA-enabled InfiniBand network.

Entry nodes are part of the FLES entry stage and provide the connectivity to the readout systems. They receive all data from the detectors and prepare them for subsequent handling and processing. Therefore, they will be equipped with FPGA cards, the CRI (cf. Chapter 5), which implement the custom readout link and the interface between custom electronics and host. Due to the direct link to the detector readout systems, the entry nodes have to be located relatively close to the detector. The readout links are designed only for short ranges. Additionally these links carry timing and fast control information

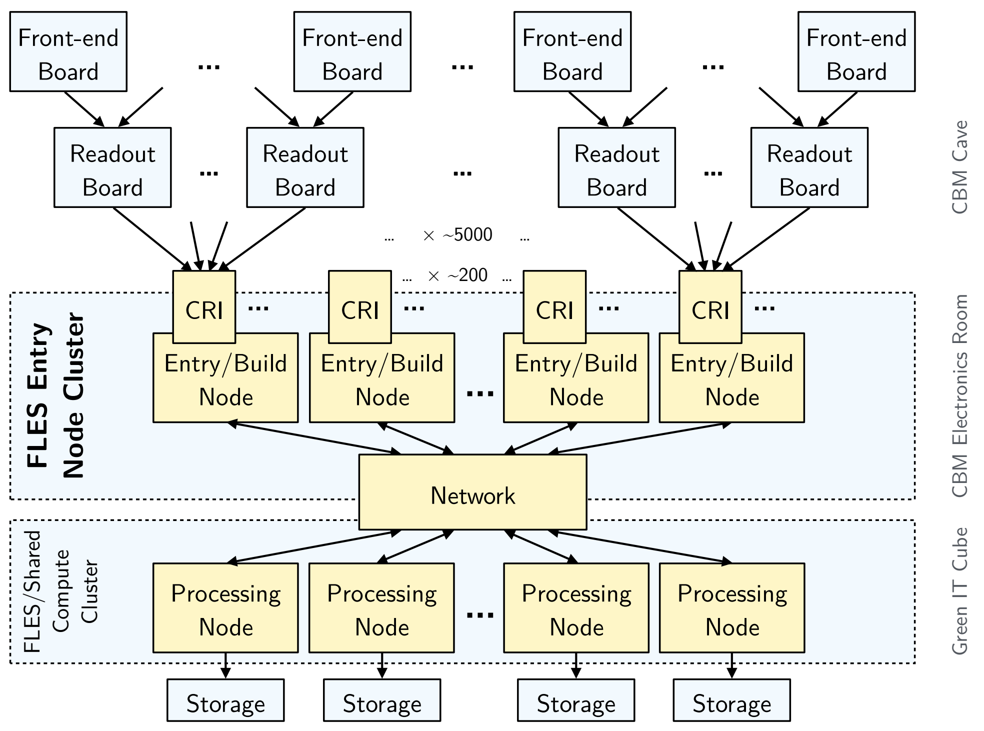


Figure 4.1: Overview of the First-level Event Selector (FLES) architecture and its position within the CBM data flow. The foreseen design consists of two clusters, an entry node cluster with the CBM-specific input interfaces, and a shared processing node cluster. To perform timeslice building, the entry nodes and both clusters are connected by a fast network.

which is susceptible to propagation delays. Further, the favorable multi-mode optics used here is not sufficient to span the distance to the Green IT Cube data center.

The processing nodes provide the needed computing power to execute the online analysis tasks. To achieve the required throughput and computational efficiency, it is foreseen to extensively make use of vectorized code and many-core architectures such as GPUs [37]. The processing nodes will be part of the common FAIR computing resources hosted in the central Green IT Cube data center. This allows for efficient resource sharing by permitting cross-experiment use during non-CBM beam times. The building yields an ideal infrastructure and allows for an economic and energy efficient use of servers. The local separation from the entry stage is possible because the FLES collects the experimental data untriggered and does not require fast access to the front-end electronics. However, with a linear distance of approximately 350 m to the CBM building, it is expected that the data transport has to bridge, including cable routing, a distance of about 1000 m (cf. Sec. 4.5.3). This distance exceeds common, fast intra-data center interconnects based on multi-mode optics and requires the application of medium-range or long-haul techniques. A detailed description of the network design is presented in Section 4.5.

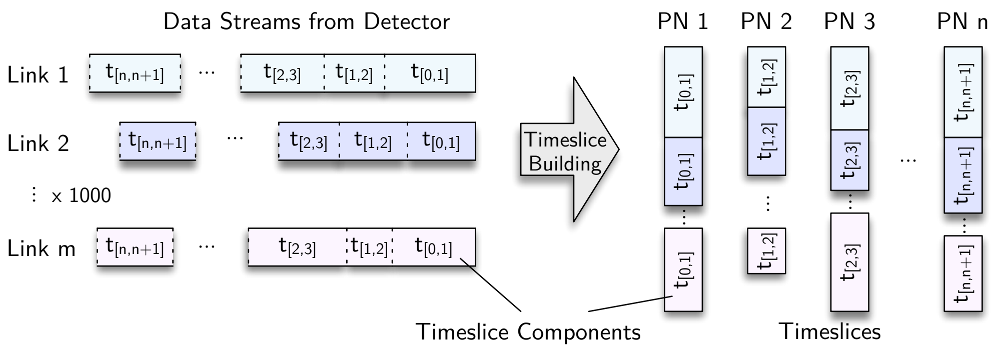


Figure 4.2: The conceptional idea of timeslice building. Data streams from the different detector links are partitioned into time intervals, labeled timeslice components, which cover the same period for all inputs. All components from one period are combined into a processing interval, called a timeslice, and sent to the same processing node (PN). Data of subsequent intervals are distributed to different nodes to balance the computational load.

4.1.1 Timeslice building

Data enters the FLES distributed across all entry nodes of the cluster. With input links associated with a specific geographic region of the detector, data from one physical event is distributed across the cluster. However, for efficient and scalable event selection, it is crucial that the analysis of a specific event can run locally, on a single node, with limited inter-node communication. Therefore, the FLES distributes the data before event selection, such that analysis tasks on different processing nodes can process containers with complete event data from the full CBM detector system independently from one another.[1] This process is called *timeslice building*, and its basic concept is shown in Figure 4.2. Input data streams are divided into time intervals, which cover the same period[2] for all inputs. Data of a single stream acquired in such an interval are called a *timeslice component (TSC)*. All TSCs of a given period are combined into a full processing interval, a *timeslice*, and transferred to a selected processing node. Consequently, a timeslice contains data from the entire CBM detector system, for a small, limited duration of the experiment. Subsequent timeslices are sent to different processing nodes to distribute the load across the cluster.

Timeslice building is similar to classical event building with the main differences being:

- Timeslice building has to be performed at the full primary detector data rate.
- The data combined to timeslices is distributed across the cluster and not reduced into a single stream.

[1] The analysis tasks may still need to exchange some information such as, for example, regarding calibration, but the corresponding amount of communication is negligible.
[2] With respect to the time the physics events took place.

- Due to the self-triggered nature of the CBM architecture, there is no explicit event separation or any other kind of global data subdivision available at this point in the processing chain. Thus, the timeslice building has to be performed using hit time information as a reference.

Timeslice building implies sustained, any-to-any communications between entry nodes and processing nodes. A fast network with sufficient bandwidth to handle the full FLES input data rate is needed. The high data rate and communication pattern favor an RDMA-enabled network. It is foreseen to use an InfiniBand network fabric. The developed FLES timeslice building framework makes heavy use of the provided RDMA capabilities.

Timeslice overlap To be able to analyze timeslices truly independently from each other, the boundaries between subsequent timeslices need special attention. Without an event definition at this stage, cuts to form TSCs are arbitrary with respect to the physics event contained. Hit messages of one event may end up in different TSCs. Analyzing incomplete event information at the boundaries can lead to analysis problems and false conclusions about the event. A trivial solution to this problem is to define an exclusion zone around the boundaries. Events with a reconstructed t_0 in this zone are considered potentially incomplete and are discarded. The required size of the exclusion zone depends on the interaction duration of a physics event with the detector, and more importantly, on the time calibration of the detector front-end electronics components. Even though all front ends have a synchronized clock, the absolute time synchronization may not be perfect. Similarly to spatial alignment, this can be shifted during the analysis. Thus, the apparent duration of an event before alignment can be stretched. As partitioning into timeslices can only be based on front-end time information, this can lead to unfavorably large exclusion zones.

For the FLES, a more elaborated solution without data exclusion zones is foreseen. Subsequent timeslices are created with an overlap. Events in the middle of the overlap region can be fully reconstructed for both timeslices. Depending on the reconstructed collision time t_0, the event is assigned to one or the other timeslice. The input interface has to facilitate the creation of such an overlap region. A more detailed discussion regarding the overlap region can be found in Section 4.2.4.

4.1.2 Processing stages

For the design of the entry stage, how the data is processed on the entry node is significant. Specifically, whether an analysis step altering the data set takes place before the timeslice building, or whether the received data is transferred directly to the processing nodes. Local data processing before timeslice building is especially useful if it can significantly reduce the data size, and therefore reduce the bandwidth required for timeslice building. If this is not the case, the data could be transferred to the processing nodes before performing any analysis step at the same cost, but with a much reduced complexity. This includes especially scaling to different load scenarios (cf. Sec. 4.1.3).

According to the current planning of the detector subsystems, there is no potential for savings expected from local processing before timeslice building. The detectors will deliver self-triggered, zero-suppressed data in already size-optimized data structures. A simple threshold-based reduction or compression in software is therefore excluded. Detectors plan to implement more elaborate compression in the upstream hardware components. The MVD, for example, foresees a cluster finder in the FEE [38]. With size-optimized detector data formats, it has to be expected that the data for the analysis will first need to be inflated, as the messages have to be decontextualized. This includes, for example, the conversion of local, compressed detector time to global time or mapping from channel identifiers to coordinates.

The design of the FLES entry stage is, therefore, based on the assumption that the majority of the data is passed on directly to timeslice building. Consequently, the data model and cluster design are optimized for a single processing stage on the processing nodes. Exceptions are link-local calibrations and QA calculations, which may require the complete history of a detector section to be continuously monitored, e. g., to calibrate the drift time of that detector. It is assumed, however, that these calculations require little computing power, are carried out in parallel to the analysis itself and do not change the content of the data stream.

4.1.3 Cluster size and scaling

The different construction stages and configurations of the experiment (cf. Sec. 3.2.2) have vastly different needs with respect to online event selection. Consequently, the FLES architecture has to support a wide range of scenarios. In terms of hardware resources, this will be mainly achieved by scaling the number of processing nodes. Nodes not used for the CBM online processing can be used for other analysis tasks or assigned to other experiments. The FLES framework has to support scaling the compute resources in this manner. The exact amount of computing power needed is a topic of active investigation. The working assumption for this document is that the full cluster consists of around 600 processing nodes. The computing requirements will be discussed in the Online TDR Part II.

The size of the entry cluster cannot be dynamically scaled as it terminates point-to-point connections to the detector systems. However it is foreseen to build this cluster in predefined steps. For the commissioning phase and early start-up phase, full connectivity to the detector, but very limited throughput and compute power is required. In this case, it is also feasible to scale down the number of entry nodes and network equipment to a minimum needed to provide sufficient connectivity. This allows to purchase some of the equipment much later in the installation phase. This start configuration of the cluster can then be upgraded to the full entry stage as needed. The full entry cluster is expected to consist of around 100 nodes. The detailed design and size of the FLES entry cluster is discussed in Section 4.5.

4.1.4 Flow control and congestion management

The FLES design has to take overload situations into account. Overload can happen for several reasons. For example, neither the PCIe host interface nor the timeslice building network guarantee a certain minimum bandwidth. Processing resources are naturally limited and may be insufficient to cope with the event data in edge cases. Furthermore, the input interface design may serve as an additional concentrator stage, which can dynamically share the PCIe bandwidth between more input links than are required to saturate the PCIe interface.

To be able to deal with overload conditions gracefully the FLES implements a closed back pressure path from the online processing interface on one end of the data transport chain to the detector subsystem interface on the other end to ensure data is not lost in an uncontrolled manner. The FLES does not foresee its own, fine-grained throttling mechanism. The resulting back pressure in the CRI is forwarded to the central throttling system, which determines how to cope with the situation. The exact mechanism here is the subject of ongoing studies (cf. Chapter 6).

The central throttling system and the closed back pressure path cover all critical cases. However, at the time of writing, it is not clear how efficient the central throttling will be in terms of discarding a minimal amount of data. It can, therefore, be of advantage to further limit the back pressure. This can be achieved efficiently by discarding all timeslice components forming a timeslice in a controlled manner before timeslice building. As this task requires coordination across all entry nodes, it is conducted by the central timeslice building framework.

4.2 Data model

The central task of the FLES entry stage is data handling, i. e., receiving the data from the detector front-end electronics and presenting these data in a well-structured format to subsequent timeslice building. To achieve efficient data handling, the design has to be tightly coupled to the needs of the timeslice building. Consequently, a suitable data model is needed that takes into account also subsequent data handling steps to avoid unnecessary reformatting at later stages. Nevertheless, timeslice building is the most demanding step, and thus, the driving factor.

Analyzing the needs of timeslice building (cf. Sec. 4.1.1), the data model should facilitate:

- *Efficient and subsystem-agnostic* data handling.
- *Partitioning of subsystem data into time intervals* that can be combined to processing intervals suitable for analysis, i. e., containing data from the same period of experiment time for all contributing subsystems and processable independently from each other.

- *Creating overlap regions* of configurable length between subsequent intervals to avoid losing events at the borders of processing intervals.

The specific nature of the self-triggered CBM front-end electronics leads to several challenges when trying to fulfill these requirements. The front-end electronics are designed to produce streams of hit and control messages in a private format, which differs for most subsystems, referenced to as *detector message format*. Representing single hits, these messages are small in data size but high in frequency. The STS, for example, expects 32-bit hit messages at rates of 10 MHz/cm^2 for the innermost regions [39, 22]. The private message format allows subsystem-specific optimization and more efficient data transport within the subsystem than a common format would. However, it presents a challenge for data handling in the FLES. At least the time information from each message is needed to partition the streams adequately. Trivial partitioning into blocks of fixed data size is not possible as detectors are self-triggered, and thus, data is zero-suppressed. Ergo, fixed-size data blocks do not correspond to time intervals. Additionally, detector message streams are in general stateful, i. e., the information depends on the history of the stream. One example is a commonly used data compression method using epoch messages (cf. Sec. 2.2.5). These *markers* carry the uppermost bits of the time information in order to reduce the size of the message timestamps [40, 41, 39]. Another example are time over threshold measurements creating separate messages for leading and trailing edge of a signal [41]. Consequently, handling data on the detector message level would require subsystem-specific logic along the whole FLES data chain. This is impractical for a readout system of this complexity as it prohibits the necessary separation between subsystems and DAQ logic. It would lead to an inflexible system in which every change in data format or addition of new subsystems, e. g., for test setups, needs to be reflected in the DAQ system as well. Also, the resulting system would be more prone to errors as the required non-uniform handling of corrupted data could more easily stall the whole chain.

Converting all detector data into a global and stateless format would solve some of the presented issues, but is not practical for a large, heterogeneous system like CBM. It contradicts subsystem-specific optimizations. Additionally, the expected inflation in data size, e. g., from converting each message timestamp to a full, global timestamp, makes such a step inefficient as it increases the needed network bandwidth for timeslice building. In the following section, a concept for structuring the input data streams is presented. It aims to allow for efficient FLES data handling without the need to convert all detector messages into a common global format or requiring subsystem-specific logic along the whole data handling chain.

4.2.1 Microslice concept

As a common data model for the FLES, the *microslice concept* has been developed. The underlying idea of this concept is to de-contextualize detector data streams by encapsulating them in a lightweight, globally defined container format.

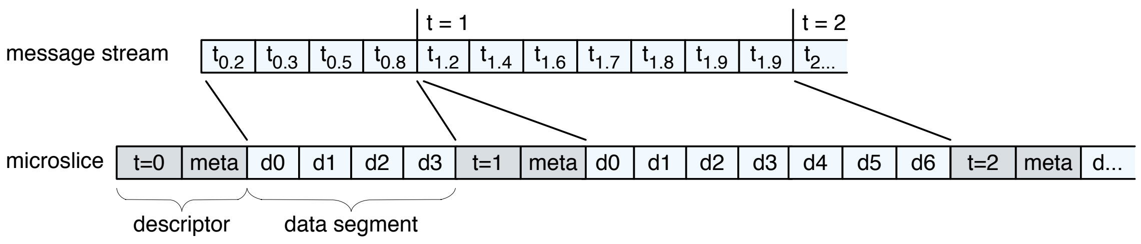

t = 1
t = 2
message stream
$t_{0.2}$ $t_{0.3}$ $t_{0.5}$ $t_{0.8}$ $t_{1.2}$ $t_{1.4}$ $t_{1.6}$ $t_{1.7}$ $t_{1.8}$ $t_{1.9}$ $t_{1.9}$ $t_{2...}$
microslice
t=0 meta d0 d1 d2 d3 t=1 meta d0 d1 d2 d3 d4 d5 d6 t=2 meta d...
descriptor
data segment

Figure 4.3: Partitioning of a detector message stream into microslices. The data stream is partitioned according to the message timestamps into fixed-length time intervals. Each microslice consists of a descriptor holding the start time and metadata in a global format and a self-contained data segment with all detector data related to the time interval of the microslice.

For traditional triggered experiments, the central trigger information gives a natural way of partitioning data. All data related to a specific trigger is grouped and added into a suitable data container. In CBM, the absence of such a global signal or any other kind of event definition at this stage of the readout chain prohibits such a partitioning scheme. Instead, the microslice concept foresees a time-based partitioning into short, context-free intervals packed into common container format. These containers are called *microslices*. Each microslice holds a predefined, constant period of data for a specific region of a detector. Microslices are designed to enable efficient, data-agnostic timeslice building, that is only relying on information provided by the metadata of the containers.

Figure 4.3 visualizes the microslice partitioning scheme. Each input channel is partitioned individually. A microslice consists of a descriptor in a global format and a block of subsystem-specific data. The descriptor provides all information needed for data handling and timeslice building—most importantly the start time of the data block in global time.[3] The data block holds all detector data of the specified time interval. To ensure microslices can be handled freely and independently from each other, the data block is required to be self-contained, i. e., the information does not depend on any previous or subsequent interval. In the case of epoch messages, for example, this can be achieved by simply repeating the latest epoch message at the beginning of each microslice. The exact format of the microslice data content can be chosen freely by the subsystems. This enables flexible, specifically tailored implementations for each subsystem. To differentiate this format from the raw detector messages, it is called *detector data format* as opposed to detector message format. Required conversions can be carried out independently and efficiently at the CRI stage, resulting in a good separation of the systems. It should be noted that the data content may still be a stateful stream as long as the initial state is provided. However, it is conceivable to transfer detector messages into a completely new format if beneficial at this stage.

To enable efficient, scalable data handling, microslices are produced at a given, fixed rate. The covered time interval is short compared to a timeslice, hence the name *microslice*.

[3]Message time stamps generally use subsystem-specific, local time.

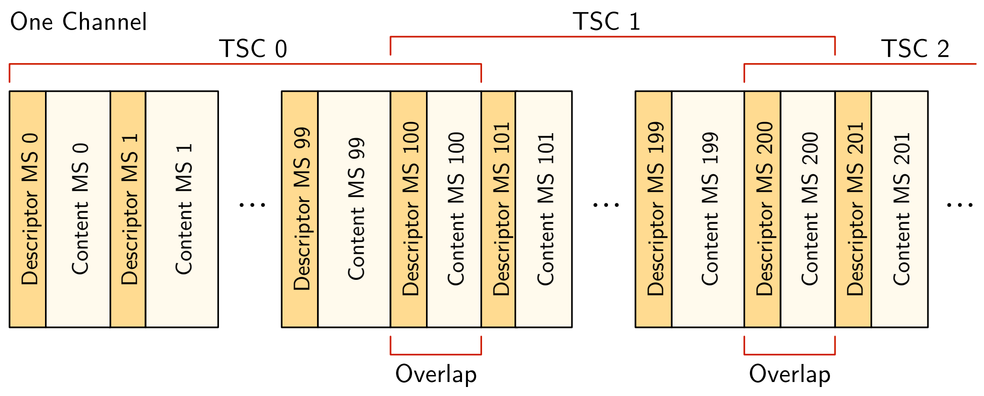


Figure 4.4: Microslice-based timeslice component (TSC) building. The time information in the microslice descriptor can be used to assign microslices to a TSC. Overlap, in this example one microslice, is created by assigning the same microslice to multiple TSCs. Different TSCs are sent to different processing nodes.

The fixed rate allows random access in the time domain. A timeslice can be created as a collection of microslices. There is no time overlap between microslices, whereas timeslices do foresee time overlap. The design aims to create precisely one microslice for each time interval. In case no data is available, an empty microslice will be generated. With this, the microslice can serve as a heartbeat to easily monitor the health of the readout system. Timeout conditions can be decided a priori as it is clear which microslice will arrive. Irregularities in the data stream, such as missing microslices, can be detected easily.

In summary, microslices are small, time-addressable subintervals that allow fulfilling the aforementioned requirements, including the creation of overlap during timeslice building. A microslice is the smallest entity that the FLES data path handles.

4.2.2 Timeslice component building

The microslice data model enables efficient FLES timeslice building based on handling only microslices. Figure 4.4 illustrates the idea of microslice-based timeslice component (TSC) building, which is the first stage of timeslice building.

The current implementation works with a common microslice rate and phase across all inputs, i. e., microslices across all inputs start at the same time and have the same length in time. This enables a very efficient timeslice building algorithm, as creating TSCs can rely on combining a given number of containers instead of checking every single timestamp. However, in the final system, it may be desirable to permit different microslice rates across the detector subsystems, and in extension possibly even microslices of arbitrary length. While this is fully supported by the general concepts, it will lead to a more complex timeslice building implementation. Therefore, it will have to be decided carefully if there is a requirement that warrants the additional complexity. In the following, a fully synchronized microslice production is assumed.[4]

Given input data streams in microslice format, timeslice building can be performed via the following algorithm. To create TSCs on an entry node, determine all microslices needed to cover the period of the TSC, including the necessary overlap. In the case of equally-sized microslices, this task is reduced to picking a fixed number of containers, e. g., 100 for the core region plus 1 microslice overlap in the example figure. Each TSC is then sent to the respective processing node. The collection of TSCs with the same index from all entry nodes constitutes a timeslice. Overlap can be achieved by doubling a certain amount of microslices at the boundary of subsequent TSC. For an overlap of one microslice, for example, the first TSC consists of microslice $[0, 100]$ and the second of microslice $[100, 200]$, which effectively doubles microslice 100.

The algorithm enables efficient data handling and network usage, as only TSCs and not single microslices are transmitted, which yields larger packages and less transaction frequency. Even more importantly, the network package size and transaction frequency do not depend directly on the size of the microslice, which allows altering the microslice size more freely.

4.2.3 Scaling to various running scenarios

The different CBM configuration and running scenarios present a wide range of input rates to the FLES. As discussed in Section 4.1, the FLES has to provide full connectivity, but the throughput capabilities should scale with the needs of the experiment.

The microslice concept takes this into account by foreseeing to adapt the length of the microslice to the respective scenario. The microslice length is chosen proportionally to the expected load such that the average microslice data size stays roughly constant. This keeps the fractional overhead of the data transport constant as opposed to a fixed overhead independent of the data rate. A reduced rate of microslices, with otherwise unchanged parameters, reduces the timeslice rate and thus the transaction frequency on the network. Assuming the number of processing nodes is reduced proportionally as well, the workload characteristics of a single processing node stay constant.[5] This allows omitting special scaling adaptations on the processing nodes.

4.2.4 Performance studies

To benchmark the microslice data model with respect to performance in the final system, it is interesting to check the introduced overhead and partitioning efficiency for microslice-based timeslice building.

[4]It is foreseen to adapt the microslice rate to different running scenarios (cf. Sec. 4.2.3). Thus a set of rates is needed.

[5]Not accounting for positive effects of the reduced processing complexity for lower interaction rates.

Target timeslice size

A prerequisite to judge the feasibility of the microslice concept is the timeslice target size. With most parts of the system still under development at the time of writing, the optimal timeslice size is unknown as it heavily depends on parameters such as the performance of the analysis software and the number of processing nodes. Therefore, it is too early to determine the final timeslice size. However, reasonable assumptions can be made.

The optimal timeslice target size is a trade-off between overhead and required buffer capacity. Larger timeslices are less demanding for the system, as overhead due to overlap and transaction frequency is reduced. On the other hand, the demand for buffer space for timeslice building depends directly on the size of the timeslice (cf. Sec. 4.5.2). For entry nodes, these are the buffers that allow to to parallelize the network transfers. For processing nodes, a timeslice queue is needed to derandomize timeslice availability and ensure maximum utilization of the compute resources. Large timeslices will withhold significant amounts of memory from the analysis task.[6] Excessively low transaction frequencies are also not preferable. If processing times reach the order of minutes, the system gets slow and ponderous to control.

As a working hypothesis, a timeslice length of 10 ms and a respective transaction rate of 100 Hz is assumed. With a total input data rate baseline of 1 TB/s, the average timeslice data size then calculates to 10 GB. Considering as an example 600 processing nodes, every node receives timeslices at 0.16 Hz or every 6 s. This results in both buffer space and interaction frequency in practical ranges and shows that also even larger timeslices could be feasible. As a result, in the following, a timeslice target size of 10 GB is assumed.

Needed timeslice overlap

An essential parameter to benchmark the data model is the required timeslice overlap. The overlap, on the one hand, depends on the temporal extent of the event to be measured. On the other hand, it depends on the accuracy of the assignment of measurements to time intervals. The duration of an event, i. e., the period in which the resulting particles interact with the detector, is dominated by the flight time of low-energy, secondary particles [42]. For CBM, this duration is in the range of 100 ns, which should be covered by the overlap.

How precisely a measurement can be assigned to a time interval depends on the intrinsic time resolution of the detector as well as the accuracy of the time distribution. The time resolution differs significantly between the different detector systems. The intrinsic time resolution for STS is about 5 ns while the integration time of the MVD is 5 µs. TOF aims for a total time resolution of 80 ps. These differences clearly show that the necessary overlap must be defined separately for each detector system.

[6]The buffers for analysis should be independent of the timeslice size as timeslices are transport containers independent of any physical properties. If required, a timeslice could be easily split into several slices after timeslice building.

The accuracy of the time distribution is mainly limited by the achievable calibration accuracy between the individual leaves of the time distribution tree. The clock, on the other hand, can be distributed with very high precision. The CBM time distribution system currently does not plan to directly determine the time offsets in the time distribution tree, e. g., through a round trip measurement. Instead, the necessary calibration parameters must be determined in software using physics events that are easy to analyze, e. g., traces of cosmic rays.[7] The calibration can be applied directly in the front-end hardware and as part of the analysis in software. With regards to the FLES input interface and timeslice building, only the accuracy of the hardware calibration is decisive, as only this is available before the analysis. The desired accuracy of the hardware calibration is the subject of ongoing discussions.

As this property of the time distribution system is not known, a perfect time distribution system is assumed in the following. This represents the more demanding scenario, as smaller required overlaps cause higher overhead. For the minimum required overlap, 100 ns are assumed.

Microslice overhead

The optimal size of a microslice is a trade-off between fine-grained access and small packaging overhead. The size of a microslice has to be comparably small to allow for efficient handling of timeslice overlap. On the other hand, smaller microslices introduce more overhead due to the overhead of the containers. These two sources of overhead are analyzed in the following. In general, the fractional overhead can be defined as

$$O = \frac{\text{overhead size}}{\text{data size}} .$$

Container overhead The container overhead is caused by the addition of metadata to each microslice. To determine an optimum point, it is useful to regard the overhead in relation to the core size of a TSC, s_{TSC}, and the average data size of a microslice, s_{MS}. As the system acts on time and not data size, these sizes have to be derived from the link data rate. Because the momentary data rate fluctuates, all calculations are based on average data rates. Consequently, all results must be interpreted as average values. The fractional container overhead O_{cont} of a TSC can be defined as

$$O_{\text{cont}} = \frac{o_{\text{cont}} \cdot n_{\text{core}}}{s_{\text{TSC}}} \tag{4.1}$$

with

$$n_{\text{core}} = \left\lceil \frac{s_{\text{TSC}}}{s_{\text{MS}}} \right\rceil$$

the number of core microslices and o_{cont} the absolute size of overhead for each container, i. e., the amount of data added to the original data. For the current hardware implementation o_{cont} originates from two sources. The first is the microslice descriptor that is

[7]The time distribution system guarantees the required timing stability even across link resets (cf. Sec. 6.1).

added to each container. The current descriptor size s_{desc} is 32 B, including additional metadata such as system identification and error flags. The second source is padding to a specific block size, which is introduced by the DMA engine (cf. Sec. 4.3.4). The current setting uses a block size s_{DMA} of 128 B. If the timeslice building copies continuous memory blocks, this padding cannot be easily removed and is therefore accounted as overhead here. Assuming random data sizes, the average padding is half the block size. Therefore, the container overhead size for the current implementation is given by

$$o_{\text{cont}} = s_{\text{desc}} + \frac{s_{\text{DMA}}}{2} = 96\,\text{B} \ .$$

Overhead from overlap The second contribution of overhead originates from the fact that the overlap has to be created from an integer number of microslices. In particular, this contributes if the length of a single microslice exceeds the length of the overlap interval. The overlap-related fractional overhead $O_{\text{over}}^{\text{MS}}$ can by defined as

$$O_{\text{over}}^{\text{MS}} = \frac{(s_{\text{MS}} + o_{\text{cont}}) \cdot n_{\text{over}} - l_{\text{over}} \cdot r_{link}}{s_{\text{TSC}}} \ . \tag{4.2}$$

with

$$n_{\text{over}} = \left\lceil \frac{l_{\text{over}} \cdot r_{\text{link}}}{s_{\text{MS}}} \right\rceil$$

the number of needed overlap microslices, l_{over} the minimal required length of overlap for a certain subsystem and r_{link} the average link data rate. As l_{over} depends on the subsystem, all overheads are calculated for individual TSCs and not for complete timeslices.

To test the feasibility of the overlap concept itself, Equation 4.2 can be altered to include all overlap, instead of only the extra overlap resulting from the limited fragmentation. The fractional overhead O_{over} created by the full overlap is then given by

$$O_{\text{over}} = \frac{(s_{\text{MS}} + o_{\text{cont}}) \cdot n_{\text{over}}}{s_{\text{TSC}}} \tag{4.3}$$

This distinction is particularly important for subsystems where the needed overlap approaches the length of the microslice. In this case, the minimal overlap contributes a noticeable offset to the overhead.

Total overhead Figure 4.5 depicts the different overhead components and the TSC fragmentation versus the microslice size for typical parameters assumed for fast detectors. For small microslices, the container overhead is the dominating factor. As expected, it is proportional to $1/s_{\text{MS}}$ as fewer and fewer microslices are created for the fixed-size TSC. For larger microslices, the overhead from overlap becomes dominating. As soon as a single microslice provides sufficient overlap, the overhead from overlap rises linearly with the microslice size. Due to the small l_{over}, only this linear part is visible. The overall overhead shows a minimum of 0.62 % at a microslice size of approximately 31 kB or 31 µs. More importantly, the overhead is well below 1 % for a wide range of microslice sizes. As the timeslice fragmentation is not crucial, the point of operation can be chosen freely. The resulting low overhead underlines the feasibility of the basic concept.

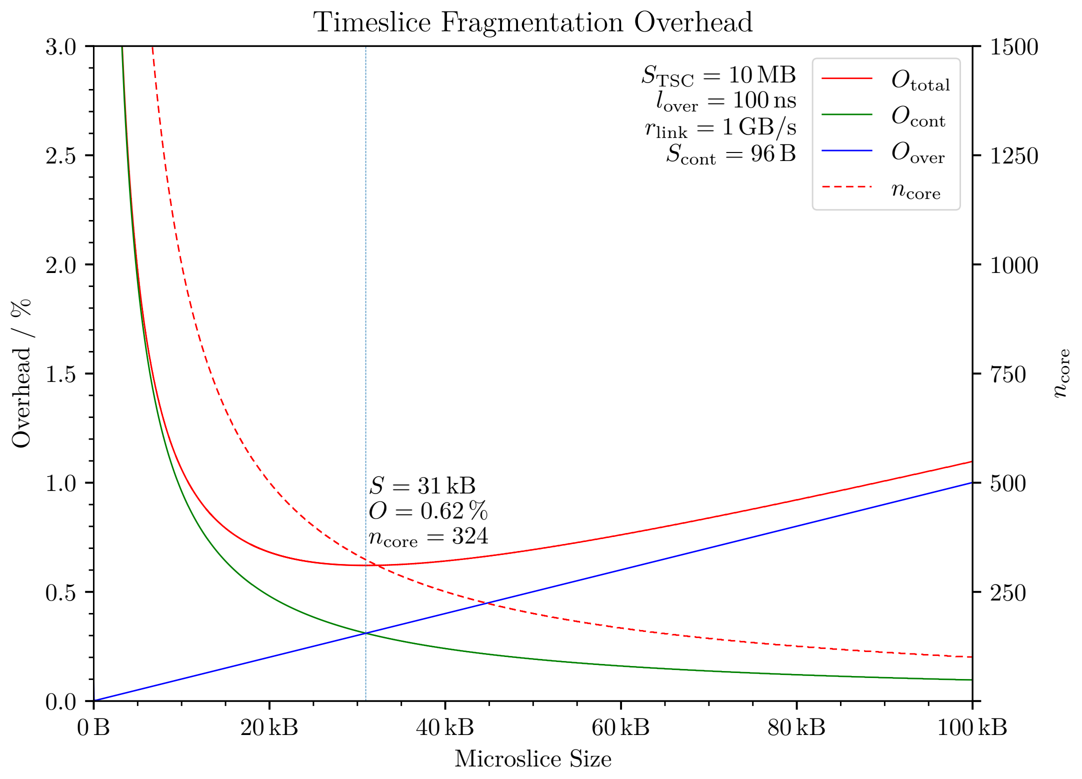


Figure 4.5: Microslice overhead for a fixed TSC size and different microslice sizes. The dashed line shows the timeslice fragmentation in the number of core microslices. The solid lines show the different overhead components. The total overhead shows a minimum at around 31 kB.

4.2.5 System constraints

A real-world system can only provide limited resources. Concerning the FLES data handling, in particular, the finite size of memory buffers has to be considered. Carelessly designed, the limited buffer space can lead to deadlock situations and stall the entire system. The following section introduces additional constraints to microslices to prevent such situations.

Maximum microslice data size

In order to lower the transaction frequency, the timeslice building handles microslices only in blocks. The current implementation uses a fixed block size of one TSC. It always accumulates all microslices for a full TSC before sending data through the network.[8]

[8]The following arguments are also valid for any other fixed-block-size implementation.

This facilitates a plain, easy to maintain implementation of the timeslice building with significantly reduced complexity compared to dynamic block sizes.

One potential issue though, arises when the input interface cannot deliver more microslices due to a lack of buffer space, yet the timeslice building will not free buffer space as it waits for a complete TSC. To prevent this deadlock, each input buffer must be able to hold at least one full TSC. If $s_{\text{MS, max}}$ is the maximum data size of a microslice and n_{MS} is the number of microslices per TSC the minimum buffer size B is given by

$$B = s_{\text{MS, max}} \cdot n_{\text{MS}} \; .$$

Guaranteeing this amount of buffer space is not trivial. The maximum microslice data size $s_{\text{MS, max}}$ can be rewritten as

$$s_{\text{MS, max}} = t_{\text{MS}} \cdot r_{\text{FEE, max}} \; ,$$

with t_{MS} denoting the length of a microslice and $r_{\text{FEE, max}}$ being the maximum data rate contributing to the microslice. In the final system, both factors may vary by several orders of magnitude. The rate $r_{\text{FEE, max}}$ depends on the maximum front-end data production rate. This rate can be many times higher than the expected data rate for a given scenario. It may even exceed the maximum possible interface data rate as all front-end derandomization buffers need to be included in the calculation. This implies that, in the worst-case, the input buffer space has to be many times larger than what is needed for typical operation. Secondly, the microslice length t_{MS} is used to adapt the system to different running scenarios (cf. Sec. 4.2.3). To keep the average data size of a microslice approximately constant, the the length of a microslice is increased for lower rates. This implies that the worst-case buffer size rises when scaling down the system.

The chosen solution to this dilemma is to limit the worst-case buffer space by defining a maximum microslice data size. Consequently, special considerations are required if microslice data content exceeds this size. Fortunately, the input buffers are in any case designed to hold several average-sized TSCs to parallelize timeslice building (cf. Sec. 4.5.2). This means the available buffer space is already significantly larger than an expected TSC. As a result, the maximum microslice data size can be chosen large enough to reduce the occurrence of overfull microslices at desired operation conditions to a minimum and treat overfull microslices as a rare error condition. In this case, it is acceptable to discard the data content of overfull microslices in favor of a running system. Such erroneous microslices need to be flagged accordingly to prevent incorrect analysis results. Otherwise, overfull microslices do not add additional system complexity to the data handling as they can be treated like any other microslice.

The microslice size limit is best enforced by the logic of the subsystem, as it is aware of the detector data format and can apply sophisticated algorithms to decide which data to discard in case of an overflow. This is, for example, useful in case a detector data format adds information in the form of a footer to the end of each microslice. Because an overflow can stall the whole FLES chain, the CRI hardware design enforces an additional hard limit for cases in which the subsystem design fails or has not implemented a soft limit. The idea is to set this hard limit slightly above a user-defined soft limit. To avoid unnecessary buffering and to allow for postmortem error analysis, only the overflow data is discarded in hardware and the associated overflow flag in the microslice descriptor is set. The current timeslice building does not treat overflow microslices in any special way. For low rates, the remaining data does not impose any performance limitation. For the highest rates, the network load may be further optimized by discarding the remaining data content before timeslice building in software. The analysis may implement special processing for timeslices containing truncated microslices or discard the whole timeslice.

Maximum microslice delay

Another critical parameter that affects the system health is the variance in microslice arrival time across different inputs. Even if the data for each microslice is produced in the same period for all inputs, the arrival time of the respective microslice may vary from channel to channel. Under normal operating conditions, the main contributions to the difference in arrival time are front-end and CRI derandomization buffers. The STS, for example, has a front-end drain time, potentially resulting in a corresponding time difference, of approximately 100 µs [43]. The input buffers equalize these differences before timeslice building. Normally, the required buffer space for this is negligible. A potential issue arises when a subsystem component or channel fails and starts delivering microslices at a wrong rate or no microslices at all. In these cases, the differences will grow infinitely until the buffer space is not sufficient anymore. Especially if the rate is too low or even zero, a single erroneous input could stall the whole system. Such situations must be detected, and countermeasures must be taken as early as possible.[9] While in particular, only the variance between the inputs and not the absolute delay is critical, determining the variance is not trivial. It would require distributed measurements of the arrival time across all inputs.

To facilitate an autonomous detection of error conditions in all stages of the readout tree, it is beneficial to define a maximum absolute microslice arrival delay. An absolute delay can be easily determined locally by comparing the microslice timestamp to the current experiment time. Each stage can implement timeouts and countermeasures based on the maximum delay. The timeslice component building, for example, can ignore microslices arriving after a timeout to avoid stalling the system. With the delay globally defined, it is in addition more coherent on how the system will react. Additionally, the current delay is a valuable system health indicator that can be published to the global monitoring.

4.2.6 Generation of microslices

Detailed knowledge about the subsystem data format is needed to generate valid microslices for detector message streams. The generating logic has to keep track of the

[9]A potential countermeasure is to insert special error handling microslices into the data stream to ensure subsequent stages do not stall (cf. Sec. 4.6.1).

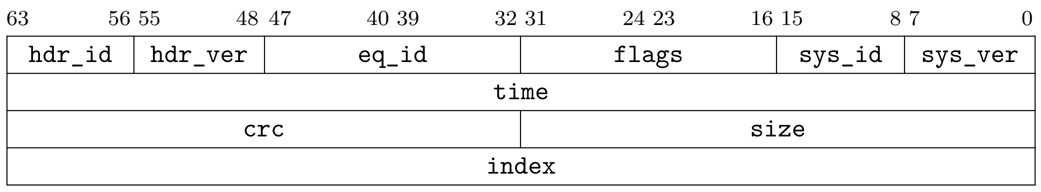


Figure 4.6: Structure of the 32-byte microslice descriptor.

message time information, convert it to global time, and partition the streams accordingly. For this reason the generation of microslices is a subsystem-specific task and will be part of the subsystem-specific logic of the CRI (cf. Appendix B).

The microslice generation process in the CRI hardware implies that, besides the general synchronization, potential front-end timing offsets have to be calibrated at least at the level of the CRI to enable the necessary conversion to global experiment time. The calibration effort required, however, can be limited utilizing the timeslice overlap mechanism. A resulting small, but defined uncertainty in time when assigning messages to microslices can be hidden in the generated overlap.

4.2.7 Microslice container format

This section discusses the details of the microslice format. Each microslice consists of a 32-byte microslice descriptor and a block of experiment data. As microslices by definition cover a constant interval of time and data production rates fluctuate, the size of the data block varies from microslice to microslice. The size of the data block is always an integer number of bytes. For each stream, exactly one microslice is generated per time interval. The microslice data contents may be empty if no data is available for the corresponding time interval.

Microslices are designed to enable subsystem-agnostic data handling. All information needed by the data handling logic is included in the microslice descriptor. The microslice descriptor is specially structured to provide a software friendly, self-contained description of a microslice. It it allows for an independent interpretation of a microslice, i. e., without knowledge of an explicit context like, e. g., the channel it was sent on. While this increases the descriptor size slightly, it adds significant flexibility to the handling of microslices. The explicit information eases the implementation of analysis steps and allows cross-checking the integrity of the system. To allow for easy software access, all fields are sized to match C/C++ built-in data types and are carefully aligned. In memory, the microslice descriptor represents a packed data structure and can be accessed by interpreting the memory region as a C struct. The structure of the microslice descriptor is displayed in Table 4.6. The individual meaning of and reasoning for each microslice descriptor field is given below.

Descriptor format identifier (`hdr_id`) The microslice descriptor contains a hierarchy of format identifier and version fields. Altogether, they define how to interpret the microslice descriptor itself as well as the data content. The hierarchy creates flexibility and separation of subsystems. The descriptor format identifier is the top level of the format hierarchy. It specifies the actual descriptor format and, therefore, the exact meaning of all other bits. The current value is 0xDD. This field is unique in that the bits cannot be easily redefined without breaking backward compatibility.

Descriptor format version (`hdr_ver`) The descriptor format version gives the descriptor revision of the microslice descriptor and defines the following fields. The value of 0x01 corresponds to the descriptor structure, as presented here. The version must be incremented if the descriptor format changes.

Subsystem identifier (`sys_id`) The subsystem identifier identifies the CBM subsystem that has generated the microslice. Identifiers are fixed for each subsystem and defined globally. The subsystem identifier defines the meaning of the subsystem-dependent bits in the descriptor. Note that it does not specify the format revision of the data content.

Subsystem format version (`sys_ver`) The subsystem format version specifies the format revision of the data content. Together with the `sys_id`, it defines the exact format of the data content. The subsystem data formats and the corresponding format version numbers are managed by the subsystems themselves. The value 0xFF is reserved for test purposes and shall not be used by the subsystems.

Equipment identifier (`eq_id`) The equipment identifier gives the unique number of the equipment on which the microslices are created. It thereby implements a source address and may be used to extend the address space of the detector data format. Together with the implicit FLES channel ID, it enables checking correct cabling.

Status and error flags (`flags`) The flags field holds status and error conditions that are relevant to the data handling or of global interest. It allows reacting on relevant conditions, e. g., a microslice overflow, early in the data path, without reading the data itself. At the time of writing, all flags are globally defined, but it is feasible to define subsystem-specific flags if a valid use case is presented. Each bit in the flags field signals a specific situation. If a specified situation occurs, the corresponding bit is set to one; otherwise, it is set to zero.

Microslice start time (`time`) Represents the start time of the microslice in TFC time format with ns precision. The value of `time` has to be incremented by the microslice interval length for each generated microslice during a run. To limit the time calibration precision required at CRI level, a defined small uncertainty in time is acceptable when assigning messages to microslices. Given the global microslice interval length T (e. g., 10 µs) and allowable time uncertainties $\Delta t_0, \Delta t_1$ (e. g., 100 ns), the microslice with the `time` field set to s may contain experiment data exactly from the time interval

$$t \in [s - \Delta t_0, (s + T) + \Delta t_1) \ . \tag{4.4}$$

Reserving 64 bit for the timestamp ensures that it does not wrap[10] and facilitates its usage as experiment time, i. e., absolute time reference that never resets.

[10]A timestamp overflow occurs only after $(2^{64} - 1) \cdot 10^{-9}\,\mathrm{s} \approx 585\,\mathrm{a}$

Checksum (`crc`) The checksum field holds a CRC-32C checksum of the microslice data content. This field is optional and only valid if the `CRC valid` flag is set. This is necessary as the hardware CRC module is optional.

Size (`size`) Gives the size of the microslice data content in bytes.

Buffer index (`index`) The buffer index describes the location of the microslice data block in memory. It is part of an index table that is created by the FLIM data handling and added to the descriptor. The index table is presented in more detail in Section 4.3.1.

4.3 FLES interface module

The stream of microslices produced by the subsystem-specific logic of the CRI must be forwarded to the hosts main memory for further processing. This task is fulfilled by an HDL module for the FPGA of the CRI, the FLES Interface Module (FLIM). It implements a common on-chip interface towards the subsystem specific logic and a high-throughput DMA engine to transfer the data to the host.

The FLIM is designed with a focus on efficient data handling. The FLES must be capable of processing all input data in quasi-real-time to avoid throttling the experiment. The more efficiently data can be processed, the fewer entry nodes are required. In high-throughput applications, the available memory bandwidth can quickly become the limiting factor. Often the number of times data is copied puts an upper limit on the achievable throughput. One paradigm allowing efficient, high-throughput data transport designs is, therefore, *zero-copy*. In practical terms, a zero-copy data flow means that the data delivered by the FLIM must be consumable by the timeslice building software without any additional memory copies. This requires a careful design of the memory management, the use of DMA, and a suitable synchronization scheme.

This section presents an efficient host interface architecture that enables zero-copy timeslice building and presents the corresponding implementation of the FLIM. The architecture was developed in context of and is described in more detail in [44].

4.3.1 Host interface memory model

The FLIM asynchronously receives microslices and has to forward them to the main memory of the host via DMA. The timeslice building continuously consumes microslices from the main memory. This forms a classical producer–consumer pattern. A memory management scheme that defines how producer and consumer exchange data is required. To facilitate a zero-copy data flow, the memory scheme must take into account the needs of the timeslice building process. The scheme must take into account the following aspects:

- The data model foresees microslices as the smallest entity of data handling.[11] The memory scheme should provide access and synchronization on the level of individual microslices.
- To avoid the need for large buffers on the CRI, the architecture must support the streaming of data to the main memory immediately without the need to wait for the entire microslice.
- Timeslice building has to combine microslices to a TSC. Processing of scattered data can be costly and unnecessarily complicated. It is most efficient if subsequent microslices are arranged in a dense, continuous block.
- Registering DMA buffers with the operating system is costly, so the scheme should support reusing existing DMA buffer space.
- The previous point results in a static input buffer. Memory assigned to these buffers is not available for other tasks. For efficient memory usage, the scheme should avoid memory fragmentation.

To avoid CPU copies, the appropriate placement of the necessary data objects in memory has to be performed directly by the DMA transfers initiated by the FLIM. Coordinating these transfers is a key point of the design. Because microslices have a variable size, it is not possible to issue fixed-size DMA transfer requests without creating memory fragmentation. Involving the device driver to calculate each transfer request ad hoc is inefficient and would require large buffers on the device to bridge the delay. Therefore, the buffer management has to run mostly in hardware, i. e., is part of the FPGA design.

Dual ring buffer memory scheme

The presented challenges can be solved by a memory scheme using two ring buffers. Figure 4.7 gives a schematic overview of the chosen buffer structure in the main memory of the entry node. All microslice data content is written as a continuous stream to the *data buffer*. For each transferred microslice content, the FLIM creates a full microslice descriptor by appending the size and position of the content in the data buffer to the initial microslice descriptor created by the microslice building. The resulting descriptor is written to the second, much smaller *descriptor buffer*, which has entries of fixed size. The ring buffers scheme naturally allows the reuse of DMA buffer space and provides a simple, lock-free synchronization mechanism via the exchange of read and write indices.[12] The memory management and write address calculation can be implemented on the FPGA without the need to involve the device driver for single transfers. No additional complex memory management in software is needed.

[11]The intended data size of a microslice and a TSC varies with the selection scenario (cf. Sec. 4.2.3). For this discussion, microslice sizes from 1 kB to 50 kB and a TSC size of 10 MB can be assumed.

[12]In the following, *index* refers to a ring buffer index in bytes rather than in elements if not denoted otherwise.

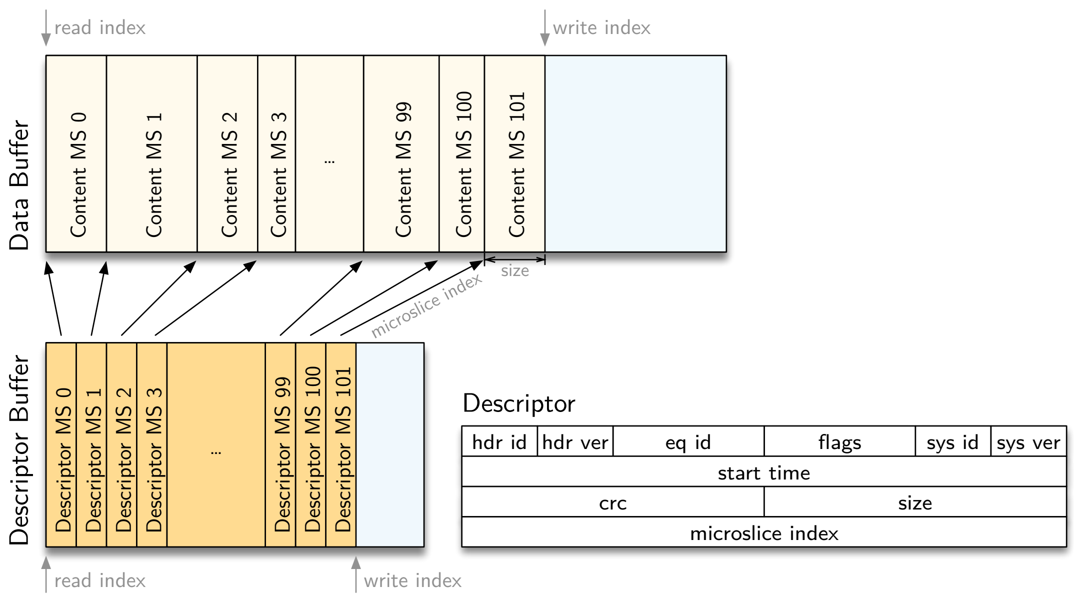


Figure 4.7: Dual ring buffer memory scheme. All microslice data content is stored consecutively in the data buffer. The microslice descriptors are stored in a separate descriptor buffer. The microslice content location is defined by the index and size fields in the descriptor.

The scheme allows efficient, individual, or blockwise access to microslices. With only fixed-size, densely-packed elements, random access to stored descriptors is possible via simple pointer arithmetics. The corresponding data content is given by the microslice index and size fields of the descriptor. Since the data content is stored as a contiguous stream, a block of microslices can be accessed with information from the first and last descriptor of the block only. Technically it would be sufficient to store only the microslice index and size in the descriptor buffer. However, the microslice data model allows timeslice building based on metadata stored in the microslice descriptor. The timeslice building process does not need to access the microslice data content. Storing the full descriptor in the descriptor buffer increases the locality of the metadata and allows fast access without dereferencing multiple pointers.

The microslice index stored in the descriptor primarily has to point to the data content in the data ring buffer. However, random access to microslices via the descriptor table is also useful after moving a block of microslices to another buffer, e. g., after timeslice building. To facilitate the reuse of the descriptor table without rewriting the indices, the microslice index is a monotonically increasing byte index. In contrast to an offset, it does not reset when the ring buffer wraps. From such an index, the memory address in the initial input data buffer filled by the FLIM, as well as in any other buffer, can be calculated without knowing the history of the data. The address offset o_n^{initial} of the microslice content n within the initial data ring buffer can be calculated from the index i_n and buffer size s

as:

$$o_n^{\text{initial}} = i_n \bmod s \ .$$

After moving a block of data to any other linear buffer, the new address offset of the microslice content o_n^{new} can be calculated from the index i_n by subtracting the index of the first microslice in the block i_0:

$$o_n^{\text{new}} = i_n - i_0 \ .$$

The first index is known locally. Information like the size of the original ring buffer is not relevant.

One downside of a ring buffer scheme is that freeing buffer space is strictly consecutive, and a single element can block significant amounts of buffer space. However, under normal operating conditions, the time required for timeslice building should not vary widely. Also, the buffers will be in the order of several gigabytes, covering multiple timeslice building blocks. For the exceptional case that a specific TSC cannot be processed, an appropriate error handling within the timeslice building process is needed in any case.

Timeslice component building algorithm

The dual ring buffer scheme enables efficient TSC building. The two buffers separate data from control information. The control flow only needs information from the descriptor buffer. A TSC building example is depicted in Figure 4.8. The algorithm[13] starts by reading the first and last associated microslice descriptors and calculating the start and end address of the data block. Next, the addresses are compared to check if the ring buffer wraps within the data segment. Data is available in one or two contiguous blocks, depending on whether the buffer wrapped. The same applies to the descriptor buffer. Thus a TSC can be described by a gather list with two to four entries, independent from the number of microslices it contains: One entry for the data block and one for the descriptor block if none of the buffers wrapped. Three entries if one of the buffers wrapped and four entries in the case both buffers wrapped. The gather list can be passed to the timeslice building sender to transfer the segments to the corresponding processing node.

4.3.2 Microslice interface

The subsystem logic of the detector partitions the detector message stream and creates microslices. In consequence, the interface between the subsystem logic and FLIM is designed to handle data on the microslice level. It is referred to as the *microslice interface*. The following section gives an overview of the FLIM microslice interface, with a focus on design decisions and logical aspects. A detailed interface definition, including all signals, can be found in [45].

[13]This simple algorithm implies that the buffer can hold at least one full TSC, which can be guaranteed by limiting the data size of a microslice (cf. Sec. 4.2.5).

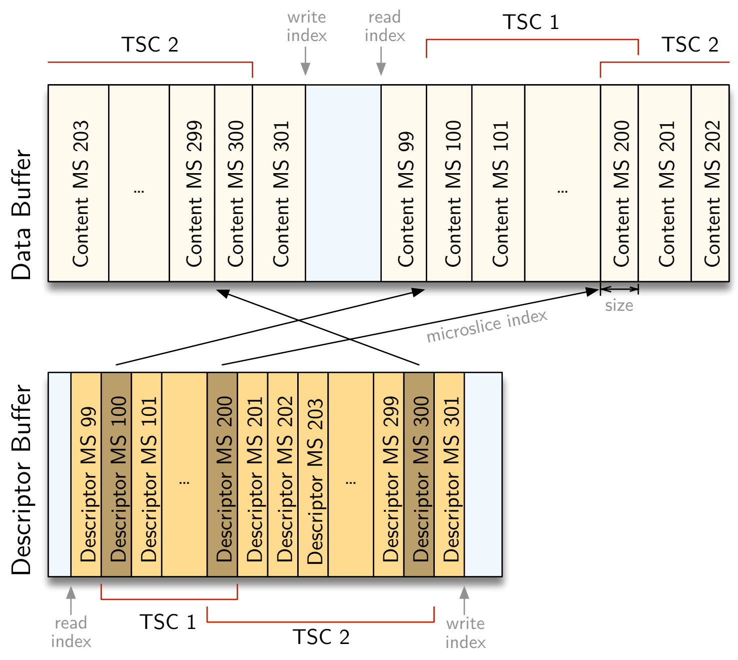


Figure 4.8: Timeslice component building with dual ring buffers. Independent of the number of microslices, each TSC can be described as a gather list with a maximum of four entries. In the example above, TSC 1 consists of two contiguous blocks (one descriptor plus one data block). For TSC 2, the data buffer wrapped, resulting in three blocks (one descriptor plus two data blocks). A warp in both buffers within one TSC would result in four blocks. Minimal memory access is needed. Only the descriptors marked in gray are read to create the gather lists.

The microslice interface can be divided into three sections: a status interface signaling some FLES system status, a configuration interface for setting the detector subsystem configuration, and the main data interface for receiving microslice data. The data interface must provide a method to associate a data set with a specific microslice. This is ensured by using a streaming interface with packet support. Each microslice data set is represented by exactly one packet. The interface implementation is derived from the AXI4-Stream [46] specification to follow a well-known interface definition. The subsystem logic streams microslice data along with a set of meta-information, e. g., the start time, to the FLIM. It does not create FLES specific data structures like the microslice descriptor. Such data structures are created internally in the FLIM module from the given metadata. This saves additional data multiplexers in the user design and provides an abstraction to change the underlying communication with the FLES.

For efficient resource usage of the FPGA of the CRI, it is essential to avoid the need for extensive data buffering. The microslice interface design facilitates designs with minimal buffering needs by allowing to stream available microslice content immediately. Information such as the data size of a microslice, which would require buffering the entire content, does not need to be known in advance. Additionally, the subsystem logic can hold the data transfer at any time by raising the corresponding flow control signals if no data from the detector is available, avoiding large derandomization buffers.

A microslice can be any integer number of bytes, independent of the data word width. A set of additional flags allows the subsystem logic to signal specific states and error conditions on a microslice basis. The subsystem logic is expected to generate precisely one microslice for each time interval and deliver microslices in chronological order. If no data is available for a given interval, an empty microslice is expected. This substantially eases the implementation of timeouts and data consistency checks in the later stages of the FLES data path.

4.3.3 Hardware design

The FLIM functionality is implemented in an HDL module which is part of the FPGA design of the CRI. Figure 4.9 depicts a general overview of the resulting hardware architecture. The design can be split into the following major building blocks:

- The microslice interface, handling all communication with the subsystem design.
- A DMA engine, coordinating the transfer of microslices to the host memory.
- A system bus and bus bridge, allowing configuration and status access from the host.

Communication with the host is established via a shared PCIe Interface. The FLIM is associated with its own PCIe physical function (cf. Sec. 5.3.1). The main building blocks are discussed in more detail in the following section.

Main data path

For most subsystems it is not feasible to merge all CRI input links into a single microslice stream due to the resource constraints on the needed merging stage. Thus, a FLIM has to be able to handle multiple microslice streams. The FLIM associates each input with a fully independent DMA channel and corresponding individual buffer structure in the main memory of the host. Consequently, the FLIM does not merge any data. The data path handles data exclusively on the microslice level and is agnostic to the microslice content.

As Figure 4.9 shows, the main data path is split into individual channel sections, which operate in parallel. Each channel consists of a microslice interface module that handles the data protocol and a DMA channel engine creating the needed memory write request TLPs.

To support high-throughput as well as high-connectivity setups with the same design, the available PCIe bandwidth is dynamically shared between all channels. A multiplexer joins the TLP streams from the individual DMA channels. Any stream can be operated at full bandwidth without changes to the hardware design or configuration, as long as the

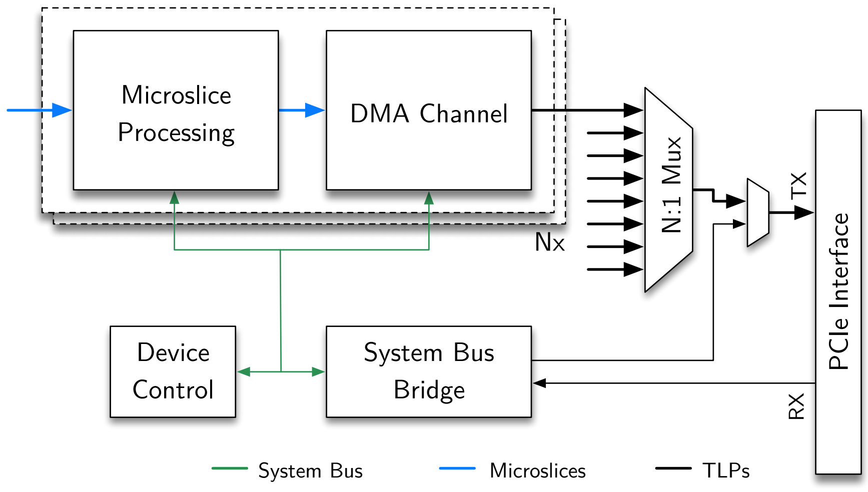


Figure 4.9: Building blocks of the FLIM hardware design. Each input is associated with a separate microslice processing module and channel of the DMA engine. All channels dynamically share the PCIe bandwidth. The bus bridge memory maps the system control bus to a PCIe BAR for configuration access.

aggregate input data rate does not exceed the PCIe bandwidth. This results in a very flexible architecture that has no interlink dependencies besides shared PCIe bandwidth.

Configuration bus

The host must be able to configure and control the FLIM hardware modules, e. g., the DMA engine. Additionally, access to status information is essential for system control.

The FLIM design utilizes an in-system bus to implement a flexible and resource-efficient configuration infrastructure. Each building block is equipped with a separate register file storing the configuration and status information. All register files are connected to a central system bus. Crossings between the different clock domains of the design are implemented on the bus level. In this case, signal synchronization is needed only for the bus signals instead of for each register. The architecture additionally relaxes the timing requirements as the configuration registers can be placed close to the attached logic. The timing of the bus can be relaxed by adding register stages, each of which is limited to the width of the bus.

The configuration bus of the FLIM is fully separated from the configuration of the subsystem. It utilizes a separate bus bridge module receiving and sending only TLPs associated with the PCIe function of the FLIM. The full system bus address space is bridged to a PCIe BAR, therefore making it available for direct memory-mapped access via the device driver. No additional communication protocol is used. The bridge receives request TLPs from the integrated PCIe core and creates the matching system bus operations. For read requests, a completion TLP is generated and delivered back to the core.

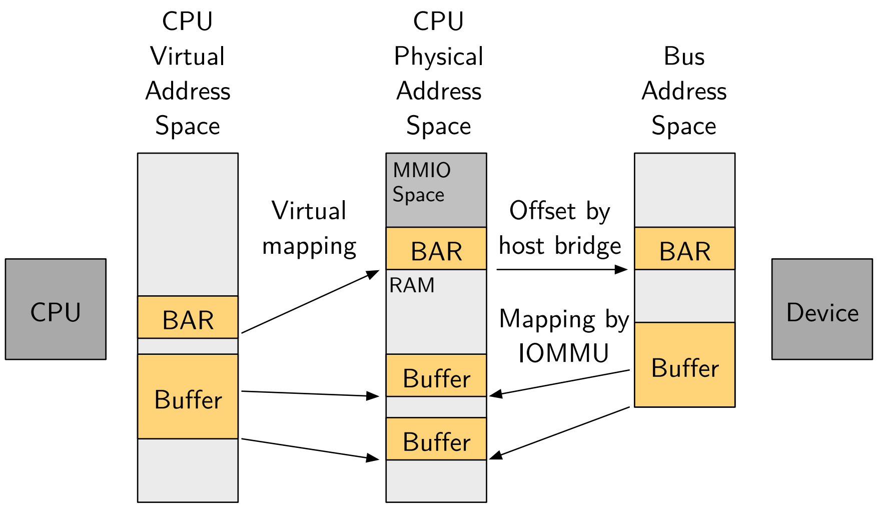


Figure 4.10: Address spaces and possible mappings in the memory subsystem of a modern Linux system. IOMMUs and host bridges can create mappings between physical and bus addresses, e. g., mapping scattered physical pages to a contiguous block of bus addresses. Figure adapted from [47].

4.3.4 DMA implementation

The following section presents the hardware implementation of the DMA host interface that is required for each DMA channel.

Handling of large DMA buffers

Due to the page-based virtual memory subsystem, a buffer that appears continuous to an application is, in general, not continuous from the view of a PCIe device. Figure 4.10 gives a schematic overview of the address spaces found in a Linux system. A PCIe card operates on the bus address space. In older systems, the bus addresses space is identical to the CPU physical address space. However, in general, PCIe host bridges and IOMMUs can produce additional, arbitrary mappings between bus and physical address.

The memory management in modern Linux systems does not support the allocation of large contiguous buffers in physical address space. In case of a direct mapping between physical and bus address space, the PCIe device sees fragmented buffers. DMA designs classically deal with this situation by performing *vectored IO*. The target buffer is described by a list of memory descriptors, a *scatter/gather list*. To transfer data to or from a fragmented memory region, the DMA engine uses the scatter/gather list to create the corresponding requests. Allocating memory in user space gives very little control with respect to how that memory is organized in the physical address space. Tests show that buffers are fragmented on page level. With regular 4 kB pages and large buffers, this can lead to long scatter lists. Such a list can have tens of megabytes and cannot be

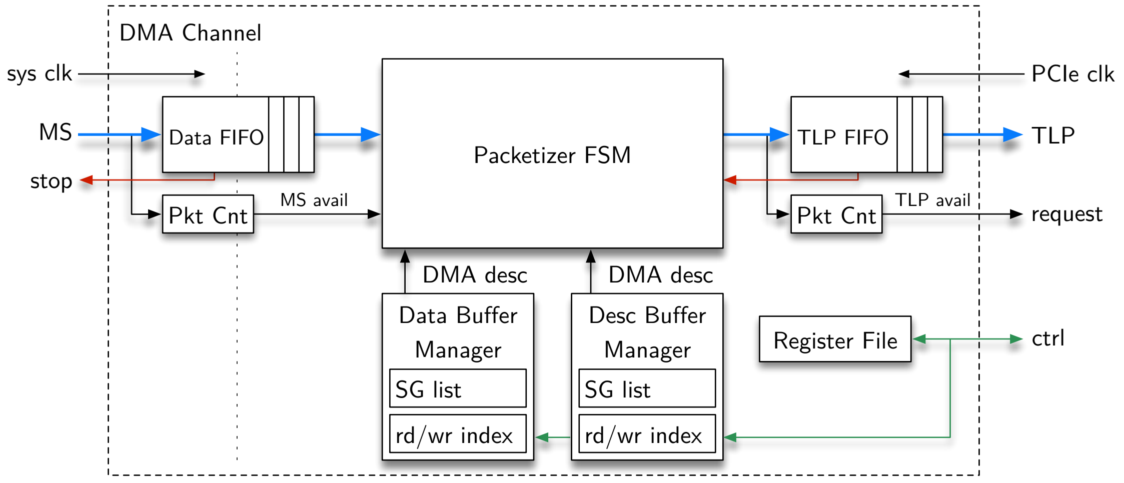


Figure 4.11: Schematic of the DMA engine. A central packetizer FSM implements the dual buffer and indexing logic and creates write request TLPs. Each ring buffer is managed by a separate buffer manager that feeds DMA descriptors with write address and length to the packetizer. A TLP is created either if data to fill the full payload is available, or if a microslice needs to be finalized.

stored efficiently in the FPGA. With large scatter/gather lists, a continuous list exchange mechanism between device and host would be required. Instead, the FLIM DMA design exploits the IOMMU feature found in all modern x86-64 CPUs. The IOMMU maps blocks scattered in physical address space to one or very few contiguous blocks in the bus address space. The resulting buffer descriptor can be easily stored in the FPGA. This limits the design complexity, as it obviates having a list update mechanism and leaves more bus bandwidth for data transfers. Tests at the full FLIM rate have not shown a measurable performance impact of the IOMMU on the PCIe throughput. An additional benefit of employing the IOMMU is the memory protection it can provide. With configured IOMMU, a device can only access its assigned address space. This increases system stability in the case of malfunctioning devices.

DMA channel logic

The FLIM hardware design implements the dual ring buffer DMA scheme. Figure 4.11 shows a schematic overview of a single DMA channel. The central element is a packetizer FSM, which creates PCIe write request TLPs from the input data. The packetizer implements the dual buffer and indexing logic. The payload capacity of a TLP is much smaller than the average microslice size.[14] For an incoming microslice, the packetizer continuously creates write TLPs until the full content is written to the data buffer. Subsequently, it extends the microslice descriptor with the microslice index and creates a single write TLP for the descriptor buffer. For empty microslices, only a descriptor write is performed.

[14]The maximum payload size of a PCIe TLP is 4096 B. The host system can further limit the size.

The packetizer has to know to which addresses the TLPs should be written. To keep the design flexible, the packetizer implementation is independent of the buffer management and address calculation. Instead, both buffers are managed by separate buffer managers. They store the read and write indices as well as the scatter lists. The buffer managers perform all address calculations and supply the packetizer with a DMA descriptor of bus address and length for each package. If any of the buffers is full, the respective manager does not supply a descriptor until enough space is available. To eliminate the need for time-critical communication with the host, the current implementation of the buffer manager stores the full scatter list in the FPGA. The device driver simply needs to update the read index to free buffer space, which is usually not time-critical. Care needs to be taken not to issue memory operations forbidden by the PCIe standard, such as transfers crossing a page boundary or exceeding the supported maximum payload size of the host machine. To reduce the complexity of packaging and address calculation, the buffer managers work with configurable TLP payload sizes. The TLP payload size for descriptor writes is determined by the fixed descriptor size. Microslices, however, can be of any number of bytes. For the last data TLP of a microslice, the packetizer pads the payload to the static packet size.

Synchronization scheme

The DMA transfers to the host memory are performed autonomously without involving the host CPU. Thus, in addition to the transfers, a synchronization mechanism between the DMA engine and user application is needed. Vice versa, the DMA engine needs to be notified if parts of the buffer can be reused.

Device-to-host synchronization In general, synchronization from device to host can be realized with interrupts or with polling. In the case of interrupts, the PCIe card sends an interrupt to the host to announce that new data, e. g., a microslice, is available. This will cause the operating system to call the registered interrupt handler, which in turn triggers the next steps to handle the data. The advantage of the interrupt method is that the system is only notified if new data is available, and the CPU is otherwise free for other tasks. The disadvantage is the comparatively high cost of handling the interrupt, which includes a context switch to kernel mode. This can become a performance issue, especially for high-rate applications like the FLES. Consequently, most devices utilize polling mode under high load [48]. In polling mode, the application periodically monitors if new data is available. While this uses additional CPU resources when no data is available, it is more efficient in a high-load scenario. In the FLIM scenario, the defined microslice frequency and relaxed latency constraints allow reducing the polling frequency to a minimum. The FLIM, therefore, implements only a polling mode, as an additional interrupt mode increases complexity without promising benefits in a high-rate scenario.

Polling relies on monitoring suitable information provided by the DMA engine. Ideally, this information is present in the main memory to minimize read transactions via the PCIe bus. For a ring buffer scheme, one choice would be to write the write index to the main memory.

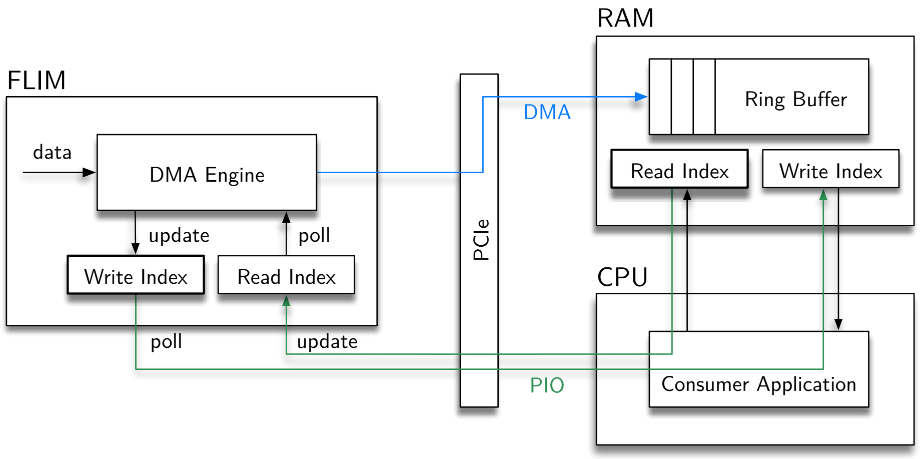


Figure 4.12: Non-blocking synchronization scheme between the FLIM DMA engine and a consumer application. For better clarity, only a single ring buffer is shown. Each ring buffer is associated with two indices. After writing data to the main memory, the DMA engine updates its write index. The consumer application asynchronously polls the write index to determine if new data is available and updates its read index when data is processed. To release buffer space, it updates the read index on the device. The FLIM hardware design implements atomic access to the shared indices stored on the device to prevent potential race conditions.

However, PCIe write operations are in general not atomic.[15] The PCIe specification only requires that updates to a memory address are observable in increasing address order. The granularity of these updates is not defined and depends on the implementation of the host system (cf. [49, Sec. 2.4.3]). This means that the CPU might observe the first part of a word updated while the other part contains an old value, e. g., when reading 64 bit from the main memory. A safe polling mechanism requires the implementation of an additional mutex or handshake mechanism. Additional complexity and operations from such a mechanism reduce the advantages of polling a location in the main memory. To guarantee atomic access to the read index, the FLIM design implements the polling via PIO reads to a memory location on the PCIe device. In this case, the FLIM hardware design determines the update granularity and can ensure that a valid index is returned.

To increase the throughput, the data path includes multiple derandomization buffers between the DMA channel logic creating the write request TLPs, and the actual PCIe interface. Multiplexing of the TLPs from the DMA engine and read completion TLPs from the bus bridge occurs after these buffers (see Fig. 4.9). Consequently, the write index managed by a DMA channel cannot be used for synchronization as the PIO read completion for the index can pass a buffered DMA write request. A second descriptor write index is managed

[15]Newer PCIe standards specify atomic operations, which could help. Because they were not widely supported at the time of development, this option was not further investigated.

to overcome this limitation without restricting flexibility in buffer design. A counter keeps track of the descriptor writes requests passed to the PCIe interface after buffering. From that point in the data path, strict PCIe ordering guarantees the PIO read completion will not pass the DMA descriptor write request.

Write index polling is only needed for the descriptor buffer. The data buffer requires no explicit device to host synchronization. Because the descriptor is always written after the data content, the state of the data buffer can be safely derived from the index and size information provided by the descriptors.

Host-to-device synchronization After the user application consumed the microslices, the memory sections have to be freed for new DMA transfers. This host-to-device synchronization is less complex as the user application can synchronously trigger it. The device driver updates the read index managed by the DMA channel for each buffer via PIO. As for the write index, the hardware implementation guarantees the proper modify granularity. Working with read indices implies that there is no need to acknowledge single microslices. The user application can group operations and free larger blocks, e. g., one or multiple TSCs with a single update. This drastically reduces the interaction frequency between the user application and the device.

4.3.5 Hardware monitoring system

The FLIMs are part of a large, distributed system, in which a single component may slow down or stall the entire system. In such a system, it is a challenging task to determine the overall system state and identify potential bottlenecks or overload situations. Precise information from all individual components is needed to assemble a full picture.

The FLIM hardware design, therefore, incorporates sophisticated hardware monitoring features. The design continuously collects status and error information as well as performance statistics for several key areas of the data flow, e. g., the microslice interface, the individual DMA channels, and the shared PCIe engine. For these points measurements like momentary data throughput, buffer fill states and generated back pressure are performed. For each DMA channel, statistics on the three possible sources of back pressure, as well as the microslice processing rate, are collected. A channel can stall either because one of the two target buffers is full or because the shared PCIe interface is busy, e. g., handling data from another channel. Collecting statistics on the data path has proven to be a valuable tool for monitoring the system state in real-time. Notably, the information on the various sources of back pressure helps to identify bottlenecks and misconfiguration quickly. For example, high back pressure from only the descriptor buffer directly hints to an insufficient size of the descriptor buffer for the given running scenario.

All measurements and status information are retrieved from the hardware via the FLIM device driver. An accompanying application publishes this information to the prototype of the central FLES monitoring system. It periodically collects metrics from all FLIMs in a node and sinks it to a central Influx time-series database. Status and history can be

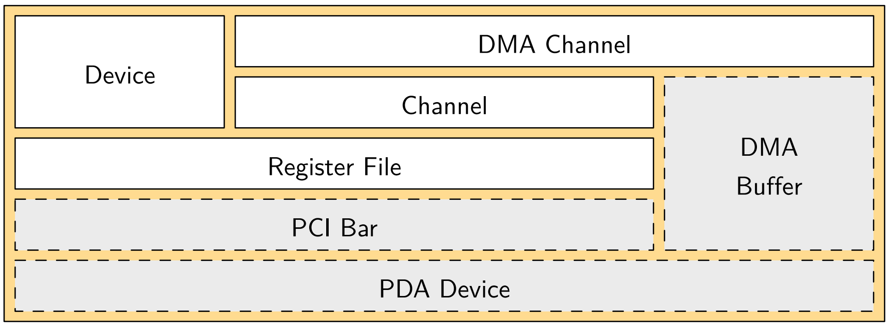


Figure 4.13: Overview of the FLIM device library. The classes give access to the configuration options of the corresponding hardware building blocks. The PDA user-space library (dashed) contributes the fundamental, generic components.

retrieved using a Grafana web application. Alternatively, the information can be presented on the local console. The monitoring information is a crucial component of all existing readout setups and is heavily used by the detector subsystems to monitor their status and data rates.

4.3.6 Device driver and library

In order to use the PCIe device of the FLIM and allow interaction with an application, a device driver is needed. The objective of the device driver is to provide a generic interface to the user application and to abstract from hardware implementation details.

The FLIM device driver is based on the PDA micro driver framework [50]. The PDA implements a minimal, generic kernel module and an accompanying user-space communication layer. The kernel module handles only the necessary kernel-space functions such as binding the PCIe device or interfacing the DMA API of the kernel [47]. The accompanying user-space library provides an API for generic device discovery, memory-mapped access to the address spaces of the device, and registering DMA buffers. All device-specific functions and the user interfaces are implemented in the custom-developed device library *libcri*.

Figure 4.13 gives a schematic overview of the device driver and software stack. Marked in white is the libcri, while the underlying PDA components are marked in gray. The libcri is fully object-oriented and written in C++. A lightweight C++ wrapper interfaces the PDA user-space library that is implemented in C. The structure of the library reflects the logical structure of the FPGA design. The classes represent building blocks and their configuration options. Building blocks can be easily added or exchanged with different implementations without influencing other blocks. The *device* class handles all device-global functions and serves as a factory for the *channel* class. Each instance of *channel* represents one FLIM channel. For a channel, the user application can instantiate a *DMA* *channel* to set up DMA transfers. All classes use the underlying *register file* class to gain access to the individual register files.

Configuration and status access

One issue a device driver implementation has to solve is consistent access to status and configuration registers in the device address space. The FLIM hardware design maps the address space of the internal system bus via a PCIe BAR to the physical address space of the host system. Access from software is then granted by memory-mapping the BAR address space. However, the internal layout of this address space, e. g., the addresses and functions of specific registers, is determined by the FPGA design. Hard coding addresses into the driver, or even the user software is inflexible and prone to errors if the address layout changes. For more flexibility, the automated FLIM hardware design build flow generates a map of register address constants. The map is not flat but separated in local register addresses and base offsets of the individual register files. The separation allows modeling repeating register files, e. g., those of the individual channels, without duplication of code. Each hardware register file is represented by an instance of the *register file* class implementing the offsets of the individual register files. With this, reading and writing registers does not require explicit address calculation for every access, and therefore, is less prone to errors.

Except for debug purposes the raw register access is not exposed through the user API of libcri. Instead, higher-level functions are provided by the individual modules. These are modeled after the functionality a given module provides and wrap the required register accesses internally. This guarantees the abstraction and separation needed in a project of this scale and allows to change or extend specific implementations later on.

Shareable DMA buffers

One task of the device driver is to prepare the DMA buffers and configure the DMA engine of the device. For a zero-copy data flow, it is necessary to share the FLIM DMA buffer with other DMA devices, e. g., an InfiniBand HCA. The driver has to support this feature. The FLIM device driver uses user-space allocated buffers to facilitate DMA buffer sharing independently of the architecture of the device driver of the other device. Memory is allocated in user space and passed to the device driver. The driver maps the pages to the bus address space of the device and prepares them for DMA but does not allocate the memory itself. This circumvents problems due to buffer ownership and allows to pass the same buffer to multiple drivers for registration with the device.

A consequence of the user-space allocation is that the buffer is usually heavily fragmented in the physical address space. In a setup with direct mapping of bus addresses to physical addresses, this would lead to long, inefficient scatter lists. This situation is resolved by employing the IOMMU of the host system to map the scattered physical memory to a continuous segment in bus address space. In order to set up a DMA channel, the user application passes two buffers to the device driver. The kernel module registers the buffer with the DMA API of the kernel and returns a bus address space scatter list. The registration step pins the memory pages and configures the IOMMU. The user-space device library uses the scatter list, which is now very short,[16] to configure the DMA engine of the FLIM.

4.3.7 Data publishing server

Integration of the FLIM into the general FLES data transport framework as a data source requires a performant application interface that is capable of transporting the full input data stream without requiring extensive processing resources or creating additional overhead. With the goal of zero-copy timeslice building in mind, the transport framework needs direct access to the DMA buffers filled by the FLIM and a possibility to register them for RDMA. Furthermore, there are potentially multiple consumers that need to access data of the same FLIM channel: Apart from timeslice building, the input interface design also has to facilitate node-local analyses, e. g., channel-local QA processes. The application interface must therefore provide efficient access to the data without copying, a multi-consumer synchronization mechanism, and a method to share the DMA buffers with multiple devices.

A combination of a publishing agent and a shared-memory-based inter-process interface allows meeting these aforementioned demands. A publisher agent, called *CRI server*, handles all device-specific features like configuration and has ownership of the DMA buffers. It is the only process that interfaces the device driver to communicate with the PCIe device. Data is published and accessible via a POSIX shared memory, which contains the synchronization structures, i. e., read and write indices, as well as the actual DMA buffers. Each CRI server process handles one CRI PCIe interface with all its channels. This segmentation allows efficient management of device-global resources, yet still provides full isolation between different channels.

The consumer applications access data via a generic, dual ring buffer interface that gives access to the data and descriptor buffer as well as the corresponding read and write indices (cf. Sec. 4.3.1). This low-level interface does not predefine any access patterns to the data and descriptor segments and thus does not limit the implementation options of the timeslice building algorithm. For more convenient access to single microslices, e. g., for QA, a high-level interface wrapper has been developed. For each active FLIM channel, the shared memory holds a dual ring buffer structure, i. e., a data and descriptor buffer pair, and corresponding read/write indices. To avoid copies, the buffer segments are written directly from the FLIM via DMA. The indices, however, are managed by the server process to provide a proper abstraction from the hardware implementation and enable multiple consumers for a single channel.[17]

[16] Usually, it contains only one entry.
[17] See Section 4.3.4 for more information on the hardware/software synchronization scheme.

Multiple consumers per channel can be supported by managing a separate read index per consumer. In this case, the CRI server writes only the least advanced read index to the device. Multiple consumers per channel increase the write index polling rate. The indices are cached in the shared memory to reduce the load on the PCIe interface in such cases. To be able to enforce a minimum polling frequency, the hardware write index is provided with a timestamp. The consumer application can define the maximum age of the write index during a request. Older indices are automatically updated from the device. The CRI data publishing allows for efficient data handling by providing direct data access and separating control information from the main data flow.

4.3.8 Evaluation

A key measure of the FLIM performance is the achievable data throughput from the microslice input to the main memory. It depends not only on the FLIM hardware design but also on the interplay with the software components and the host system. Measuring the throughput in a real system allows analyzing the performance with all influencing factors.

Figure 4.14 shows the results of a throughput measurement as a function of the microslice lengths with a prototype hardware design on a HTG-K7 PCIe card. The HTG-K7 is based on a Xilinx Kintex-7 FPGA and provides a x8 PCIe 2.1 interface. The host system has been equipped with an Intel Xeon E3-1220 and 16 GB 1333 MHz DDR3 memory. Data sources are internal microslice pattern generators set to a maximum output of 1 GB/s. The data rate limit includes a 24-byte microslice descriptor that is part of the protocol. The DMA engine is configured for a packet size of 128 B. Data is consumed by a simple consumer process, which calculates the data rates and discards the data immediately.

To emulate the final system realistically the microslice pattern generators generate microslices at a fixed frequency. In consequence, the microslice data size will depend on back pressure. The resulting measurement determines the achievable throughput for a given microslice length. In the context of the FLES application, this is more relevant than throughput versus packet size. For a better orientation, the upper abscissa shows the maximum microslice data size, which is derived from the microslice length and the maximum data rate of the pattern generator. It should be noted that this is only the upper limit and not necessarily the actual data size of the microslice.

For a precise analysis, the throughput for different components of the data stream is measured individually. The dashed lines show the achieved throughput of pure microslice data content and thus give the possible payload throughput for a given microslice length. The solid lines show the total data rate over the PCIe interface. In addition to the microslice data payload, this includes the microslice descriptors and the overhead from the DMA package padding. This number represents the load on the PCIe interface. The upper line shows the theoretical PCIe limit for 128-byte packages and 64-bit address cycles. The shown range of microslice length is chosen to include exceptionally short

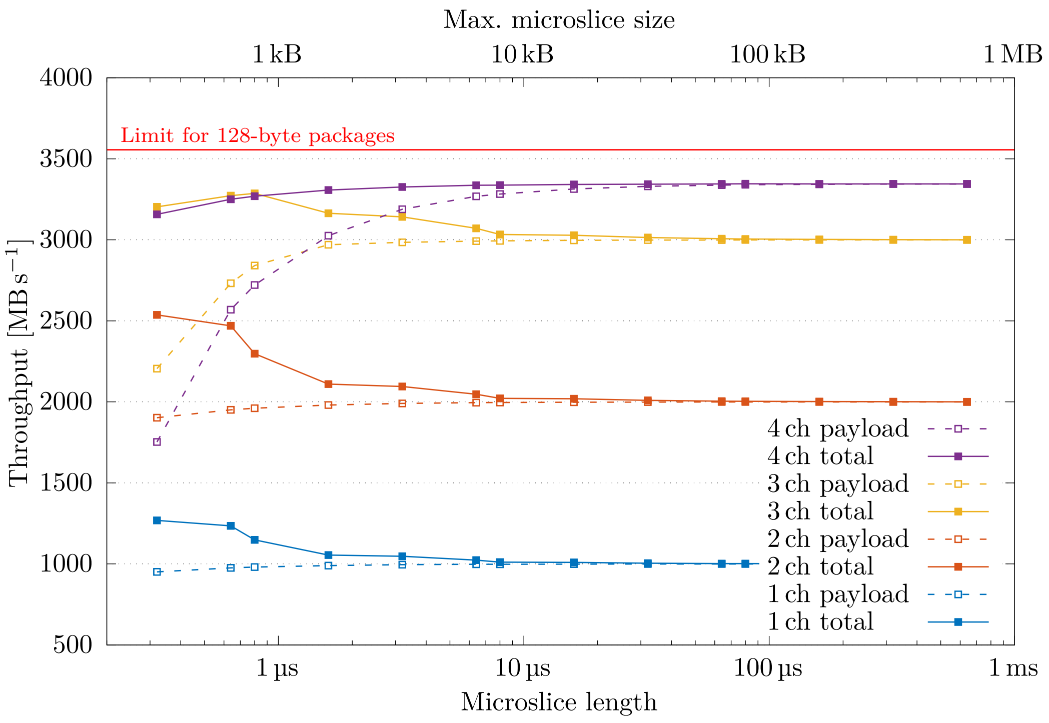


Figure 4.14: FLIM DMA throughput measured with microslice pattern generators for different microslice lengths and numbers of active channels. The upper abscissa shows the maximum data size, which is derived from the length and maximum pattern generator data rate of 1 GB/s. For each setting the total data throughput over the PCIe interface, as well as the microslice payload data rate, are shown.

and long microslices in order to determine the impact of such extremes on the FLIM architecture.[18]

Up to two active channels, the total data rate is less than the limit of the PCIe interface. The payload throughput is determined by the rate delivered by the pattern generator. The slight drop in payload rate for exceptionally short microslices is caused by the higher fraction of link protocol descriptors compared to longer microslices, which are included in the rate limit of the pattern generator but are not accounted for as payload. The PCIe bandwidth used is considerably higher than the payload throughput for short microslices. This overhead is caused by the additional descriptors and the DMA padding inefficiency.[19] The irregularity of the overhead curve is due to the measured sizes having different padding efficiencies. The overhead vanishes for larger microslice lengths, as expected from Figure 4.5.

For three channels, the total bandwidth needed for very small lengths exceeds the PCIe

[18]The expected microslice length for full timeslice building is roughly between 10 µs to 50 µs (cf. Sec. 4.2.4).
[19]If necessary, the efficiency for short microslices can be optimized by reducing the DMA packet size at the cost of maximum throughput for longer microslices.

limit. Due to the resulting back pressure, microslices contain fewer data. As a consequence, the achievable payload throughput drops significantly in this region. The total throughput is the highest of all measurements. For lengths above approximately 1 µs, the PCIe bus is no longer the limiting factor, and the payload curve sets to resemble more and more the behavior of the data source.

For four channels, the accumulated pattern generator rate limit of 4 GB/s exceeds the available PCIe bandwidth, and the total throughput resembles the bandwidth of the PCIe interface for all lengths. The drop in PCIe efficiency for small length compared to three channels is caused by the one-third increase in the package frequency that the interface has to handle. The achievable payload throughput for small microslice lengths is further reduced compared to three channels and approaches the PCIe limit for larger lengths. For the microslice target region between 10 and 50 µs, the payload throughput is already above 99 % of the measured PCIe limit.

The achieved throughput can be compared to the theoretical PCIe bandwidth. The theoretical limit of the PCIe interface depends on the protocol overhead from TLP headers. For 128-byte packages and 64-bit addresses, the absolute maximum theoretical limit calculates to 3556 Mbit/s. It must be expected that fluctuations in the host system, e. g., from housekeeping tasks or buffers shared with other devices, further reduce this figure. The measured maximum FLIM PCIe throughput is 3345 MB/s. This calculates to 94 % of the theoretical limit after protocol overhead and 86 % before protocol overhead. An analysis via in-system monitoring shows that the FLIM DMA engine uses nearly all available PCIe cycles to send data. The transmission rate is, therefore, limited solely by back pressure from the host system.

Similar measurements with a focus on scaling to more sources have been performed with the CRI1 board (cf. 5.2.2) and a prototype hardware design. The test system used is equipped with two Intel Xeon E5-2650 v4 CPUs and 128 GB of 2400 MHz DDR4 main memory. The host supports up to eight x16 Generation-3 PCIe extension cards, two of which each share a PCIe switch on the mainboard. For the tests three CRI1 cards were installed, each connected to a separate PCIe switch. One CRI1 was connected to NUMA node 0 (bus address 0x:00.1), while the other two cards were connected to NUMA node 1. Each CRI1 features two PCIe endpoints combined by an additional on-board PCIe switch and is enumerated as two PCIe devices.

Figure 4.15 shows the achievable payload throughput for different numbers of active PCIe endpoints. The measurement was performed at a microslice size of 100 µs, a DMA package size of 256 B, and 6 microslice streams per endpoint. To mimic a realistic scenario, the DMA buffers are distributed equally across both NUMA domains. The first CRI1 in each domain only uses memory on the local NUMA node. The memory for the second card in domain 1 is interleaved across both NUMA nodes, requiring a certain amount of inter-CPU communication.

The measurement shows a very good scaling of the throughput with the number of devices. A single device reaches up to 7.06 GB/s, which equates to 95 % of the theoretical maximum at the given package size. Throughput per CRI1 decreases slightly to around 6.65 GB/s

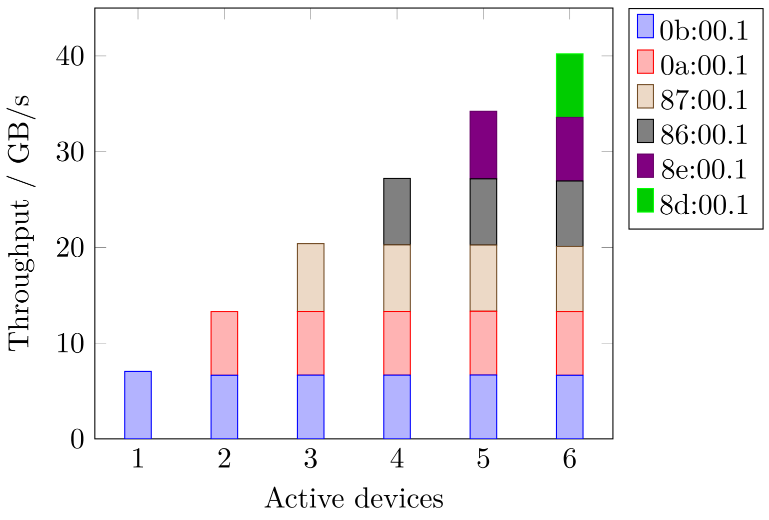


Figure 4.15: Measured payload throughput for multiple CRI1s in one host. Each CRI1 features two PCIe devices. The total throughput scales nearly perfectly with the number of devices and CRIs.

when both endpoints of one card are used simultaneously. This effect is not present if multiple endpoints from different CRIs are used, leading to the conclusion that it is a limitation of the CRI1's PCIe switch. The total measured throughput reaches 40.2 GB/s with a fair distribution across all devices. The measurements show that it is possible to operate a host with 3 CRI1s in a realistic scenario without significant performance decrease. It can be expected that a more modern host system with support for PCIe Generation 4 or 5 will provide similar or better performance.

4.4 Timeslice building

In the CBM experiment, detector raw data consists of a continuous stream of detector messages with no immediate association to a collision event. To efficiently manage these raw data, it is first packaged into Microslices (see Sec. 4.2.1), each representing data from a specific short amount of time and a specific part of a detector. The microslices for a certain time interval are then combined over all detector parts in a single node. This *timeslice building* replaces the traditional step of *event building* in a triggered experiment.

As several important event selection scenarios in CBM require an analysis of the full event, this timeslice building has to be performed at the full data rate generated by the detectors. To implement overlapping timeslices (see Sec. 4.1.1), it should in addition be possible to duplicate a configurable number of microslices at the timeslice borders, thus generating timeslices that can be analyzed independently without any loss of efficiency. Designed for an incoming data rate exceeding 1 TB/s, timeslice building is a challenge even for today's computer architectures. The implementation has to be carefully optimized for memory bandwidth usage and network load to reach an economically feasible solution.

In this section, critical aspects of a suitable software design are discussed. It also includes the presentation of the results of a software implementation, which addresses these challenges.

4.4.1 Timeslice building implementation

Memory performance Limited memory bandwidth is one of the most important bottlenecks in today's computers, and memory interface speed continues to increase slower than CPU core performance or density. The general method of mitigating this situation is the use of caches. However, in the case of timeslice building, the required data buffer sizes are too large to fit in a cache. To achieve good use of the available resources, it is therefore crucial to minimize memory access in the respective software.

Optimizing memory performance in the timeslice building application involves two aspects: First, whenever possible, DMA should be used to perform memory accesses bypassing the CPU. Second, whenever possible, data should be passed between different processes on a machine via a shared memory region. The first aspect is already implemented on the input side by the FLIM, which independently stores the data in the memory of the computer via DMA (see Sec. 4.3). For the subsequent transfer via the network, an RDMA-capable network allows direct transfer from the memory of a source computer to the memory of a target computer mostly without the intervention of the CPU at maximum efficiency. Computationally expensive transfer protocols such as TCP/IP are thereby avoided.

RDMA-enabled network RDMA-enabled networking technology has a favorable architecture that allows for efficient data transfer using remote DMA (RDMA) semantics. Data is directly transferred from and into buffers managed by the user application. Using an optimized buffer structure, no memory bandwidth has to be wasted for dispensable copy operations.

InfiniBand, which supports RDMA as a core feature, is currently the leading networking technology at the fastest HPC clusters in the world. In the TOP500 list of June 2022[20], of the top 100 clusters, 79 were using InfiniBand. Especially for use in heterogenic installations, alternative network standards exist, such as RoCE, which combines RDMA semantics with Ethernet technology.

The CBM timeslice building implementation provides the flexibility to use any of these RDMA-enabled networks. However, the majority of the tests have been performed using InfiniBand technology. Most current installations feature HDR generation hardware at 200 Gbit/s.

[20]See https://www.top500.org/lists/top500/2022/06/

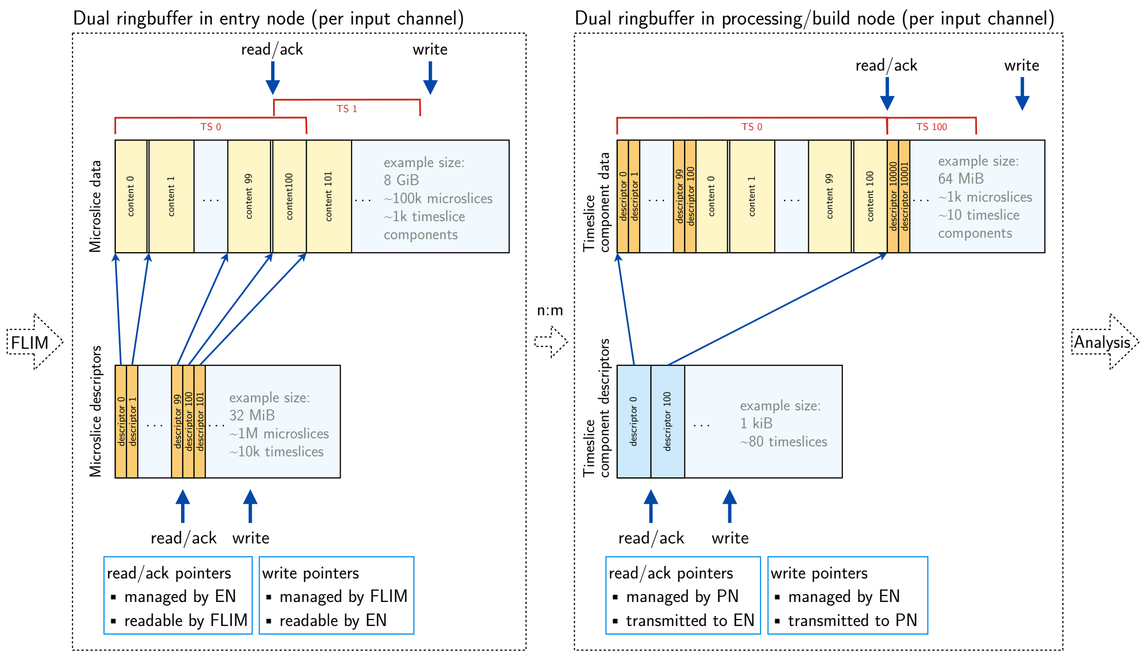


Figure 4.16: Overview of Flesnet buffer management. Both sender and receiver maintain sets of dual ring buffers to store data and descriptors.

Shared memory To optimize memory performance on the individual nodes, we employ POSIX shared memory segments whenever data is to be transferred between hardware and software components or between different software processes. Since all buffers are located in user space, there is no need to copy data between user space and kernel space. This consistent use of shared memory segments eliminates wasteful copy operations or even time-consuming serialization/deserialization cycles. By using the same memory areas both as source/destination for (R)DMA transactions and for interprocess communication, true zero-copy operation is achieved.

The presented concepts and paradigms for high-performance timeslice building have been implemented continuously in the *Flesnet* open-source data distribution software [51]. The software is written in the C++-17 language with a limited number of external dependencies.

4.4.2 Zero-copy data flow

The buffer structures used for handling detector data in Microslices are shown in Figure 4.16. Per input channel, the FLIM (see Sec. 4.3) writes to a large dual ring buffer, consisting of a data buffer and a descriptor buffer. The data buffer holds the raw sequence of microslice data contents, each zero-padded for best PCIe transfer efficiency, with data volume varying according to the current detector data rates. The descriptor buffer, on the other hand, holds exactly one constant-size descriptor per microslice and is used as

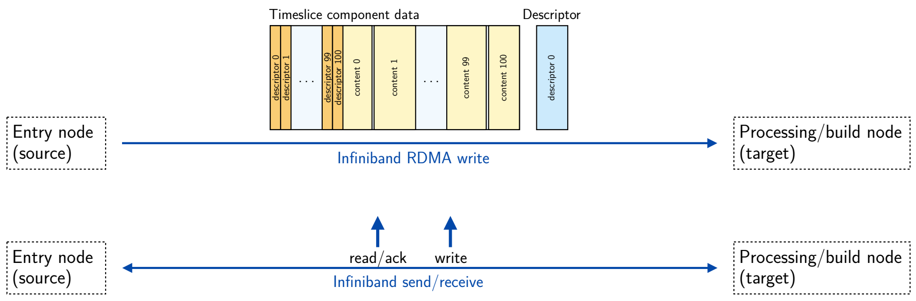


Figure 4.17: InfiniBand Transactions during timeslice building. Data and descriptors are transmitted via RDMA write, asynchronous read/write pointer updates independently via send/receive.

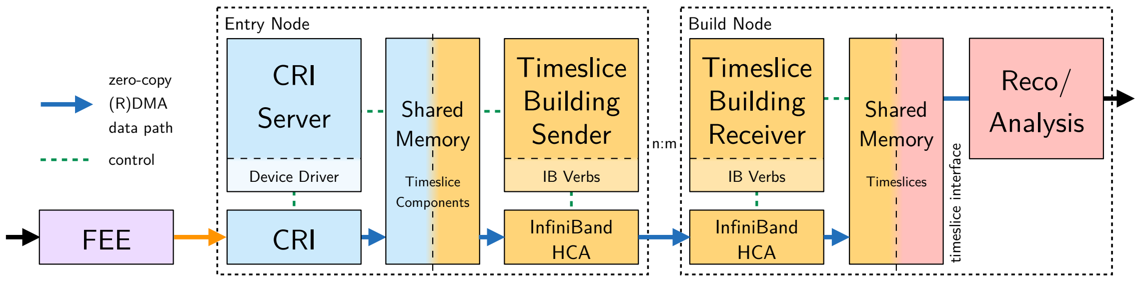


Figure 4.18: Overview of the main data path in the FLES. The CRI server and the timeslice building applications coordinate the data flow based on metadata. The main data transfer is performed directly via DMA by the CRI and InfiniBand HCA hardware.

an index table to the data buffer. In this scheme, synchronization between the readout hardware and the software processes can be reduced to an occasional update of the read and write pointer positions.

The timeslice buffer on the target build node can follow the same dual ring buffer concept, but now with a sequence of microslices as the variable-size data structure, and timeslice component descriptors for indices and metadata. The InfiniBand transactions employed for the data transfer between the entry node and the build node are depicted in Figure 4.17. The data itself is transmitted through an InfiniBand RDMA write transaction, while the necessary synchronization between both ends is reduced to asynchronous updates of the read and write pointers.

It is foreseen to migrate from this basic scheme to a more dynamic memory layout for the timeslice buffers and more tightly scheduled network transfers in the future. This will enable additional flexibility and control, especially with varying data volumes. However, the straightforward scheme presented here already delivers high performance and reliability in testing (cf. Sec. 4.4.6 and 4.4.7).

Figure 4.18 summarizes the zero-copy (R)DMA data path for raw detector data from the front-end electronics to the online analysis in software. Controlled by the CRI server,

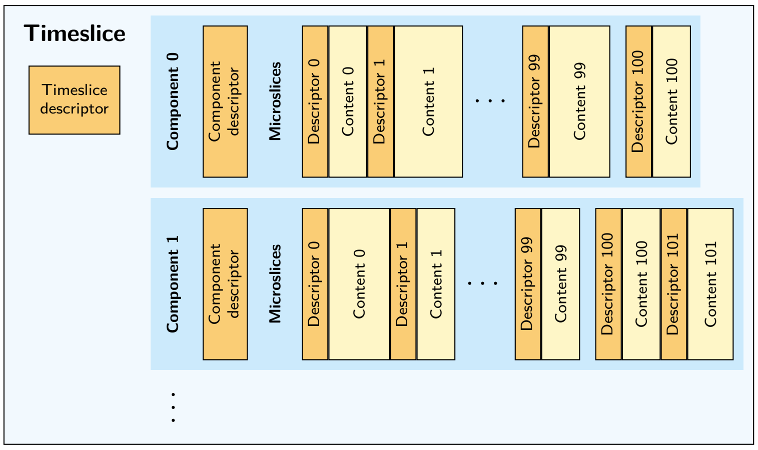


Figure 4.19: The timeslice object is the primary data structure at the interface to the online reconstruction code.

the CRI FLIM writes all incoming data via DMA directly to a shared memory section. The server application publishes the corresponding synchronization information to the same shared memory. The timeslice building sender process on the entry node acts as the consumer of the FLIM data. Having registered the buffer sections in the shared memory for RDMA, it creates gather lists for TSCs based on the descriptor information and passes the information, together with the appropriate destination address, to the InfiniBand driver. The InfiniBand HCA transfers the data via RDMA to a memory segment on the corresponding target build node, which is also set up as a shared memory.

As a result, the primary data flow is handled exclusively by DMA transfers without involving the CPU. The consistent use of DMA, together with relying on comparably small amounts of control information, ensures that the timeslice building works very efficiently with respect to necessary compute resources.

4.4.3 Online software interface to timeslices

On the target nodes, the timeslice data is stored in a shared memory segment during timeslice building, ready to be consumed by the online analysis software chain. The primary data structure presented to the online analysis software is the timeslice object as shown in Figure 4.19. It consists of a number of timeslice components (TSCs), each containing the data of a single FLIM input channel. Each TSC contains a sequence of Microslices with a configurable number of duplicated microslices at the end to allow for a fully independent analysis of different timeslices. The data structure includes metadata in corresponding descriptor structures at timeslice, timeslice component, and microslice levels. The FLES timeslice API classes are designed to provide identical access methods to both online data in the shared memory and previously stored, serialized timeslices (cf. Fig. 4.20).

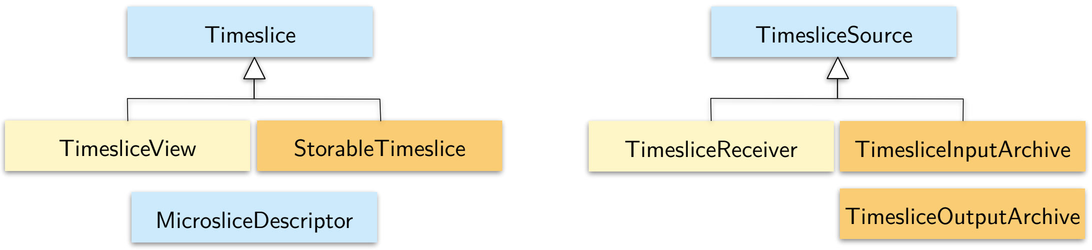


Figure 4.20: The FLES timeslice API main access classes provide identical access to online and stored timeslices.

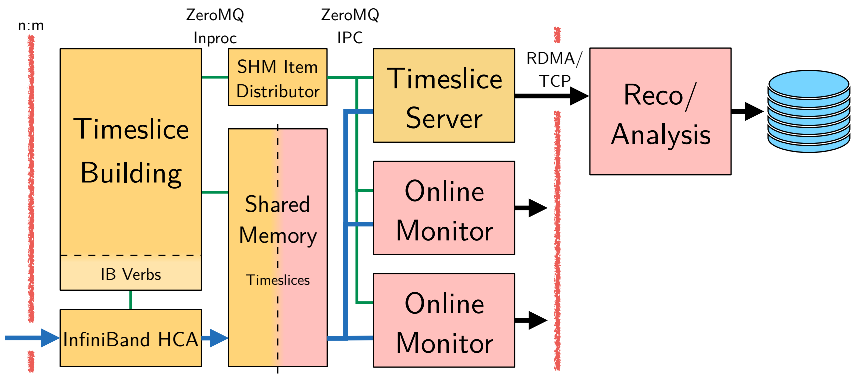


Figure 4.21: The timeslice interface as a part of the data chain. The interface employs a dedicated distributor for *shared memory items* to connect timeslice producer and consumers.

While direct access to the timeslice data in the shared memory is highly efficient, there are also requirements concerning the flexibility of distributing these timeslices to respective analysis processes. This includes the demand for:

- several independent timeslice consumer processes,
- control over back pressure generation, and
- flexibility in starting and stopping consumers.

The online timeslice interface fulfills these requests. As illustrated in Figure 4.21, it supplements the shared memory with an item distributor entity, which communicates with timeslice producer and consumers over local ZeroMQ Inproc and IPC sockets [52]. Depending on the readout scenario the built timeslices can either be forwarded to the online analysis running on a different server, or they can be consumed directly on the build node. If raw timeslices need to be stored, the *Timeslices Server* process on the build node can be replaced or augmented by a *Timeslice Archiver* process.

Decoupling the start and stop of producer and consumer processes requires careful implementation and testing because of the large number of possible event sequences. This

Policy	Effect
QueueAll	All item are queued and eventually delivered; may create back pressure.
PrebufferOne	Opportunistic delivery; keeps consumer busy but may skip items.
Skip	Always wait for the newest item, do not queue; may skip items.

Table 4.1: Different queuing modes can accommodate a variety of consumer use cases.

solution makes it possible to independently control data transport and online analysis (through the EDC), and also allows for potentially failing consumer processes.

The timeslice consumer API allows for several parameters to be chosen per consumer. These parameters include an *offset* and *stride*, which specifies a controlled subsampling of the timeslices, and a queuing policy (cf. Table 4.1). Using these parameters, a wide variety of scenarios can be covered from dependable timeslice archiver processes, best-effort online analysis, and highly responsive online monitoring, to full online analysis.

All consumer processes share access to the timeslice data from the same read-only shared memory segment and only correspond with the distributor agent through short messages. Thus, there is no need to copy any raw data in memory, which maximizes memory efficiency and performance on the node.

4.4.4 Load management

The timeslice building system is designed to operate continuously at the full data rate of the CBM subdetectors for maximum efficiency. During operation, however, it cannot be ruled out under all circumstances that the available bandwidths will be exhausted at certain points. To ensure stable and robust operation, it is necessary to plan for these exceptional load scenarios and to handle them appropriately.

The timeslice concept is well suited for discarding data in a controlled manner and thus throttling the data rate. Since the timeslices can be analyzed completely independently, they can also be discarded individually without damaging the remaining data. Only an unavoidable loss of efficiency occurs, which corresponds to the proportion of discarded timeslices. Since the timeslices correspond to fixed time intervals, it is always immediately apparent for which periods data are available.

In the event of a backlog caused by the online analysis software, Flesnet first fills the memory buffers in the build nodes, which allows us to buffer over the beam profile and smooth the data rate. We plan to use a load balancing mechanism to dynamically adjust the load on the build nodes to compensate for major fluctuations in timeslice sizes and online processing performances. If the backlog persists despite these measures, complete

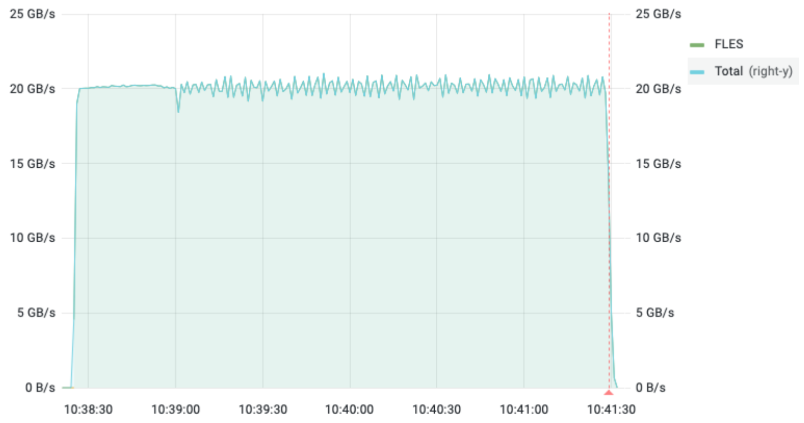


Figure 4.22: Data rate for timeslice building using test pattern data on two nodes in the mCBM experiment, from run number 1350 on 2021-06-18

timeslices are discarded by not transferring the corresponding timeslice components and releasing the respective buffer memory on the entry nodes. In the exceptional case that the timeslice building network is overloaded, the same mechanism can be used to throttle the data rate.

4.4.5 Downscaling options

While the main focus of the implementation is on high performance and reliability in a large setup, scalability to small development setups is also a design goal. The high-performance RDMA-based transport implementations using the InfiniBand Verbs API or Libfabric are complemented by a simple ZeroMQ transport. In this mode of operation, no RDMA-enabled network is required, but data can be transferred transparently over various network technologies, such as Ethernet. This enables straightforward operation of timeslice building also in small lab setups.

Similarly, the high-throughput timeslice interface is complemented by a simple ZeroMQ publish-subscribe interface to timeslices, which eases the deployment of comparatively undemanding online monitoring tasks. This high downscaling capability allows the software to be used for all data-taking operations in CBM, even during development, which helps to establish core concepts and stable interfaces early on.

4.4.6 Demonstrator systems

Various Flesnet demonstrator systems have been built during the course of development, and they have been and are being used in various detector beam tests as well as smaller ongoing experiments. The most recent and extensively tested is the one at the ongoing mCBM [29] experiment. Here, in addition to the performance of the system, aspects of reliability, robustness, and controllability are put to the test.

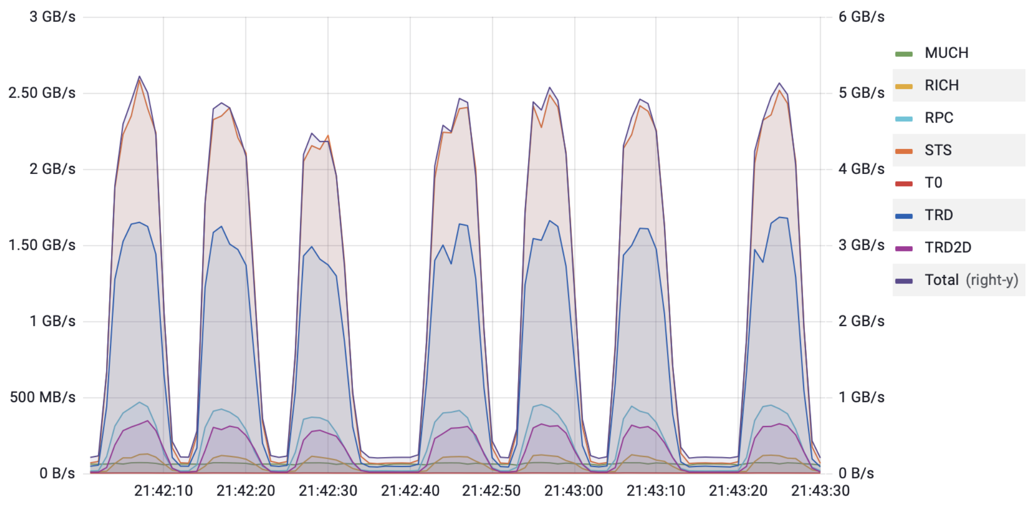


Figure 4.23: Data rate per subsystem in the mCBM experiment, example from run number 2448 on 2022-06-16

In the mCBM experiment, we have already employed a variable number of entry nodes (1–6) and timeslice build nodes (1–4) in a productive setup. For dry runs without detectors, pattern data can be generated artificially in the CRIs in the entry nodes. A test run with two entry nodes (16 channels in total) and two timeslice build nodes resulted in a total throughput of about 20 GB/s for timeslice building without further optimization (see Fig. 4.22), proving swift operation of the data chain from CRI to timeslices.

Figure 4.23 shows as an example the data rates of the participating detectors in the June 2022 beam time for a short period. Here timeslices were built over InfiniBand RDMA, combining online data from seven individual subdetector systems (28 input channels) in parallel. The setup consisted of six entry nodes and three build nodes. Data was recorded as serialized raw data objects to SSDs and HDDs. The peak data rate of 5.3 GB/s during spills was smoothed well by buffering to an average recording rate of 2.4 GB/s.

This multi-node setup includes an early version of a central configuration and control system for online data taking. In conclusion, it demonstrates the successful productive operation of the timeslice-building FLES/DAQ chain of CBM.

4.4.7 Timeslice building scalability

High performance of the timeslice building process in a large system is crucial in achieving the CBM design goals. While there is substantial experience with the performance of the Flesnet implementation in the scope of the mCBM experiment, benchmark testing on an existing HPC system allows us to scale this to larger systems [53].

The results presented in Figure 4.24 were measured on a subset of nodes of a system with 2x Intel Cascade Lake Platinum 9242 (CLX-AP) CPUs with 48 cores and 384 GB

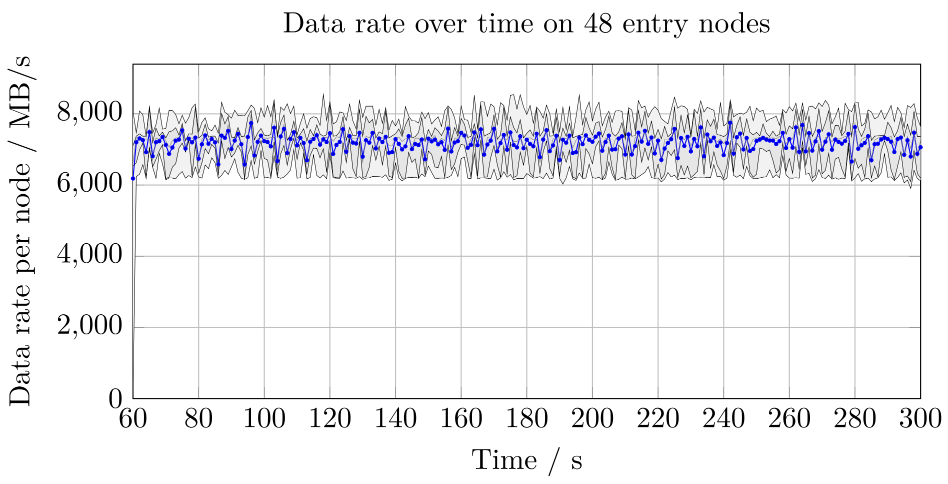


Figure 4.24: Measured average timeslice building data rate per entry node. Gray areas and lines denote minimum/maximum, 10/90 percentiles, and median.

RAM, using Libfabric to run Flesnet on the OmniPath network of the system [54]. Using an allocation of 96 nodes (operating 48 as entry nodes with artificial data sources, 48 as timeslice build nodes), the measured sustained data rate per node is approximately 7.2 GB/s, with spontaneous fluctuations in the data rate per node typically below 10 %. This corresponds to 73 % of the maximum sustained bandwidth of 9.9 GB/s reported by Intel for uni-directional transfers per node and results in an aggregate maximum sustained data rate of approximately 345 GB/s. This data rate is already close to the expected maximum rate for CBM at SIS100 (cf. Sec. 3.3.3).

4.5 Cluster and network design

The FLES is split into two portions, an entry portion housed near the experiment and a processing portion housed in the Green IT Cube. Both portions are foreseen to be constructed as Beowulf clusters from mainly COTS components. The entry cluster is exclusive to CBM and houses specialized custom hardware, i. e., the CRI PCIe cards to connect to the detector systems. The design is flat without additional stages. As a high-speed interconnect an InfiniBand fabric is foreseen. The processing portion is part of the central FAIR IT general-purpose cluster. These resources will be shared with other experiments. However, it is foreseen that at least some resources are exclusively assigned to CBM during CBM beam periods to allow for sufficient control and separation to reliably run the experiment. This section describes the FLES cluster and network design in more detail. As the design of the processing cluster is driven by many factors besides CBM the discussion will focus mainly on the entry cluster and CBM specific requirements.

State-of-the-art server technology At the time of writing a modern server CPU can provide 64 cores and 128 PCIe generation 4 lanes. The next CPU generation is expected to shift to PCIe Gen5 doubling the available bandwidth. For interconnect InfiniBand HDR technology provides 200 Gbit/s links. A typical HDR HCA has a PCIe Gen4 x16 host interface and fits into a half-height half-length (HHHL) PCIe slot. The next InfiniBand generation, labeled NDR, doubles the bandwidth and will provide 400 Gbit/s links. To be able to saturate these links a PCIe Gen5 x16 host interface is required. The same numbers apply to suitable Ethernet equipment. The following discussions are mainly based on existing hardware. It should be noted that the concept is not limited to these technologies. It is foreseen to use the most recently available technology for the final design if it is more cost-efficient.

4.5.1 Cluster size and scaling

The FLES has to support the wide range of the CBM running scenarios. Additionally, the design has to support the commissioning of the experiment. Purchasing compute resources as late as possible is due to the steady advances in technology usually most cost-efficient. This raises the question of how to size and scale the needed resources. It is planned to construct the entry cluster in two steps. An early commissioning-focused setup and the final SIS100 setup.

In an early, commissioning-focused setup connectivity can be assumed as the limiting factor. The FLES must be able to connect all detector systems. However, the data rates of the systems and needed processing resources should be comparably low. In a scenario with local timeslice building (cf. Sec. 4.5.4), the processing cluster can be scaled freely as it houses no specialized hardware and there are no stringent requirements on the network. The entry cluster needs to be able to house all CRI cards. The customs hardware, i. e., detector electronics, and CRI card must be decided much earlier than COTS hardware to allow for development and production. It therefore can be assumed the CRI card will be equipped with a PCIe gen3 x16 interface or similar. With a modern server, it is feasible to operate up to ten PCIe cards in a single, dual-CPU node. In a connectivity-focused setup the output bandwidth of an entry node can be designed with a high blocking factor, thus it is sufficient to operate a single HCA per node. This leaves place for eight CRIs per entry node. Assuming the setup will consist of approximately 200 CRI boards (cf. Sec. 5.2.4), an absolute minimum number of 25 entry nodes are needed. However, this does not take into account separation of subsystems. It should be noted that, depending on the available systems at purchase, it may be more cost-efficient to operate more nodes with fewer PCIe requirements. This will be a decided upon purchase.

For the CBM SIS100 setups, the limiting factor is network throughput and computational power. The entry stage must be able to cope with the full output data rate of the detectors. As the entry nodes must be compatible with the custom CRI hardware and require more stringent tests and adaptions it is not foreseen to upgrade the cluster stepwise with given running scenarios but support the most demanding case from the beginning. This is reasonable as the majority in value of the equipment is in processing nodes.

The number of entry nodes is determined by the maximum throughput and buffer capability a node can provide. Considering the substantially different data rates per CRI of different subsystems it is feasible to balance the per node date rate by equipping nodes of different subsystems with a different number of CRIs. Additionally for subsystems with highly unbalanced data rates between CRIs, entry nodes can be balanced with low and high occupancy CRIs. To minimize interference between systems it is not foreseen to mix CRIs from different subsystems in a single node.

The computational power provided by the processing nodes can be scaled easily by adding more nodes. This can be done more freely than adding additional entry nodes as it requires no hardware modifications. More nodes will also increase the network capacity in the processing portion. Timeslices can be distributed across these additional nodes. The amount of computational power needed is subject to the second part of the document and will be discussed there.

4.5.2 Timeslice building buffer size

To perform timeslice building, both entry and processing nodes require buffers to store the data temporarily. The following section shows that the lower limit on buffers needed depends directly on the size of the timeslices and the available network bandwidth. The derivation for the presented equation can be found in [44]. In the real system, additional buffer capacity is required, for example, to compensate for unavoidable fluctuations in transfer and processing times which are not part of the discussion here.

Dimensioning of the entry node buffers

First, the required buffer size for the entry nodes is studied. Considering the data flows in the overall system, data rates must naturally adhere to

$$R_{\text{tot}} \leq K_{\text{EN}}^{\text{tot}} \leq K_{\text{PN}}^{\text{tot}}$$

with R_{tot} the total data rate of the timeslice building, $K_{\text{EN}}^{\text{tot}}$ the total output capacity of the entry nodes and $K_{\text{PN}}^{\text{tot}}$ the total input capacity of the processing nodes.

Giving the limited input capacity K_{PN} of a single processing node, multiple nodes must receive TSCs simultaneously to utilize the network. Consequently, each entry node must buffer data for each receiving processing node. With s_{TS} the target size of a timeslice the total amount of required entry node buffers can be expressed as

$$B_{\text{EN}} = s_{\text{TS}} \cdot R_{\text{tot}} \cdot \frac{1}{K_{\text{PN}}} \,. \tag{4.5}$$

It can be seen that the required buffer space in the entry nodes directly depends on the input capacity of the individual processing nodes. A higher K_{PN} thus has a direct positive effect on the required buffer size of the entry node. The total input capacity of the processing nodes $K_{\mathrm{PN}}^{\mathrm{tot}}$, however, is not used. This circumstance has to be considered primarily in a system with much more processing than entry nodes. Analyzing the bandwidth requirements in such a system suggests connecting the processing nodes at a reduced bandwidth. However, (4.5) shows that this directly increases the buffer space requirements on the entry nodes. Since the total input capacity of the processing stage is not included in the equation, the network can be designed with a corresponding blocking ratio without having to increase the required buffer space.

Dimensioning of the processing node buffers

For efficiency reasons, processing nodes should be operated at the highest possible utilization in order not to waste compute capacity.[21] To avoid processing nodes waiting for data a timeslice queue before analysis is needed. The buffer size needed for timeslice building on the processing nodes is then:

$$B_{\mathrm{PN}} = s_{\mathrm{TS}} \cdot n_{\mathrm{queue}} \cdot n_{\mathrm{PN}} \ . \tag{4.6}$$

with s_{TS} the maximum size of a timeslice, n_{queue} the number of queue slots and n_{PN} the number of processing nodes.

For a real system, additional slots for derandomization may be needed. Whether this is the case depends on the distributions of timeslice building and processing times. In general, a higher surplus in K_{PN} allows for broader distributions without the need for additional slots. This shows that also the buffer space on the processing nodes depends on their respective input bandwidth K_{PN}.

As a result, the memory requirements for both entry and processing nodes can be determined from the given equations. It has been shown that a surplus on K_{PN} relaxes the demands on buffers for timeslice building on both types of nodes. Considering this in the design of the timeslice building network allows optimizing the cost by finding the ideal ratio between network resources and host memory.

4.5.3 Inter-cluster connection

The split of the FLES cluster into an entry portion housed near the experiment and a processing portion housed in the Green IT Cube implies that all experiment data has to be transmitted between these two sites. Both clusters foresee InfiniBand as their main interconnect. This suggests using InfiniBand for the inter-cluster connection as well, saving the need to bridge to a different network technology. The planned cable route runs clockwise around the southern FAIR campus, bridging a cable distance of approximately 1000 m.[22] This exceeds the typical application range of standard InfiniBand equipment

[21] If the cluster provides excess compute capacity, nodes can be removed from the online system and utilized for other tasks.

[22] The current planning foresees up to 10 144-fiber trunk cables.

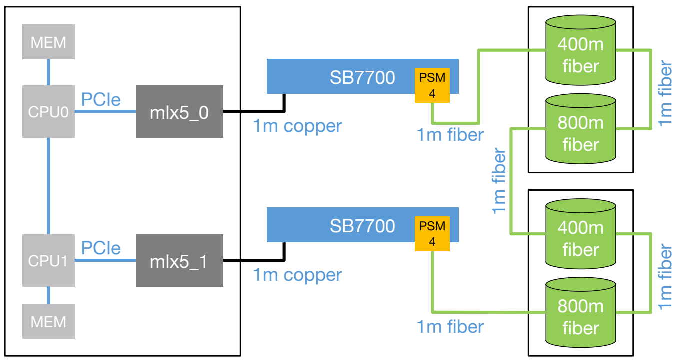


Figure 4.25: Overview of the long-haul test setup in its maximum configuration using 2.4 km of fiber. For shorter distances fiber elements are removed from the link by changing the patches.

which is, e. g., for EDR speeds below 100 m. Specialized long-haul InfiniBand equipment is available but targets reach from 10 km to 40 km. Such equipment has a substantially higher port cost and would not be cost-efficient for the FLES application. The FLES concept aims to use standard InfiniBand switches and mid-range optical transceivers to connect the cluster.

The physical reach of the connection is mainly dependent on the optics used. The distance implies single-mode optics. For this a large number of suitable optical transceivers are available in the market. A good match are for example Ethernet BASE-FR4-style transceivers capable of reaching 1 km over OS2 fibers. Another possible option are PSM4-style transceivers. This style of transceivers can be more cost-efficient as they do not employ CWDM optics to multiplex 4 lanes to a single fiber but use 4 pairs of fiber instead. While this style usually is employed below 500 m some versions cover the needed 1 km. The exact choice of the transceiver is dependent on the final InfiniBand equipment and will be made accordingly. Purchasing suitable transceivers is not expected to be critical.

Another limit for any reliable network connection is imposed by the available buffers needed to compensate for the link delay. Any data needs to be buffered at the sender until it is acknowledged by the receiver. The amount of buffers needed is defined by the bandwidth-delay product $S_{\text{buff}} = B \cdot t_{\text{rtt}}$. InfiniBand implements the transport layer responsible for reliable transmission in hardware. Consequently, the buffers are provided by the InfiniBand hardware and are more limited than software buffer approaches. Fortunately, current InfiniBand switches allow, at least partially, the collapsing of buffers reserved for quality of service (QoS) features into a larger buffer at the expense of QoS. In theory, this should increase the reach of the network sufficiently for CBM demands. To prove the feasibility of this concept a real-world network end-to-end test was performed.

A schematic overview of the setup is depicted in Figure 4.25. All tests were performed using Mellanox InfiniBand EDR (100 Gbit/s) equipment. An Intel Skylake-SP based server with 2 Mellanox ConnectX-5 HCAs served as data source and sink. Both HCAs were connected

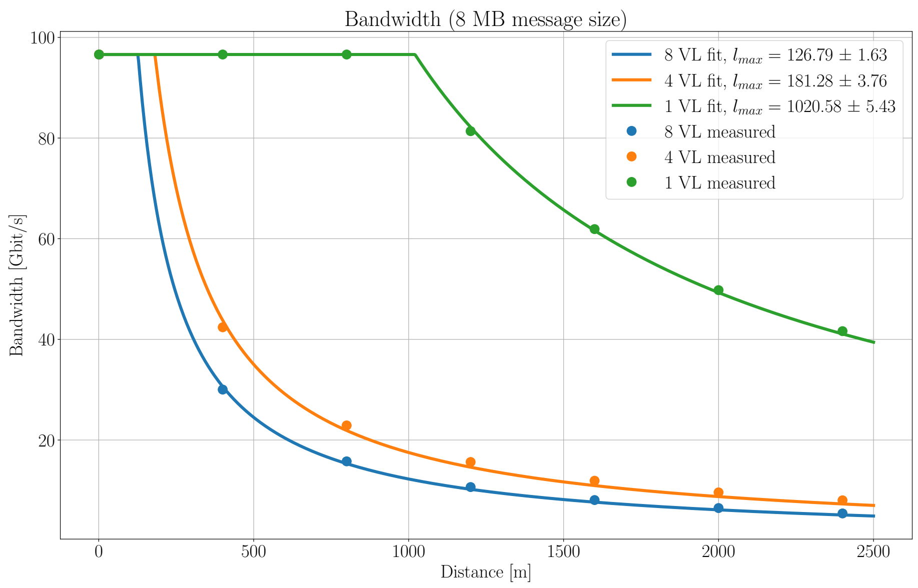


Figure 4.26: InfiniBand EDR bandwidth measured for different virtual lane settings and distances. The line is the fitted theoretical behavior, $d_{\max}$ is the calculated limit without bandwidth reduction.

to one SB7700 switch each using short copper cables. The connection under test was realized via a single-mode fiber test setup allowing to test connections with distances of up to 2400 m in increments of 400 m and Mellanox PSM4 transceivers certified for up to 2 km [55].

The used switch supports up to eight virtual lanes. The InfiniBand standard implies that each virtual lane has an independent, non-shared set of transmit buffers. Configuring the switch to fewer virtual lanes is expected to collapse buffers of non-used lanes to a larger buffer. Bandwidth measurements were performed for configurations with eight, four, and one virtual lanes. For each configuration, the seven possible distances starting with 2 m were measured. All distances were cross-checked by measuring also the RTT and calculating the speed of light.

Figure 4.26 shows the results of these tests. The dots are the measurements. The line is the fitted theoretical behavior. The expected bandwidth is limited by the maximum bandwidth $B_{\max}$ the physical link is capable of and the bandwidth-delay product. Once the available buffers are exceeded the bandwidth is expected to drop $\propto 1/d$. The resulting theoretical bandwidth can be written as:

$$B(d) = \begin{cases} B_{\max} & \text{if } d \leq d_{\max} \\ B_{\max} \cdot \frac{d_{\max}}{d} & \text{if } d > d_{\max} \end{cases} \tag{4.7}$$

where $d_{\max}$ is the distance at which the buffers are exceeded. For the plot $d_{\max}$ is determined from the measurements via a least squares fit for $B_{\max} \cdot \frac{d_{\max}}{d}$.

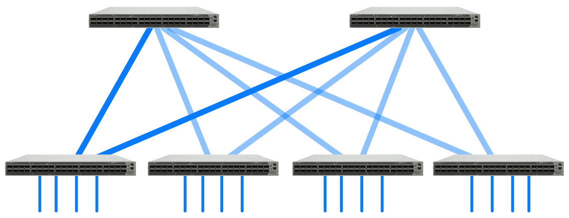


Figure 4.27: A generic, two-level, fat-tree style network; constructed as a variant of a 3 stage Clos network.

The measurements show a very good agreement with the theory. The maximum possible reach for one virtual lane, without loss of bandwidth, has been calculated to be approximately 1020 m. The maximum distance for one virtual lane is approximately eight times the distance for 8 virtual lanes leading to the conclusion that in this case buffers are fully collapsed. For four virtual lanes, the calculated maximum distance is significantly less than what would be expected from a doubling of buffers. This may be caused by a more complex buffer behavior but does not pose an issue for the foreseen use case.

The measured distance is sufficient for the needs of CBM. Additionally, there were no issues observed when running the links beyond this limit. So in case, the cabling distance increases it is still possible to employ this method at the cost of bandwidth reduction. It is expected that also future generations of InfiniBand switches will allow to collapse buffers as this feature is also employed by specialized InfiniBand long-haul equipment. In this case, also buffers of different ports are collapsed which is not possible with the standard firmware.

4.5.4 Network architecture

The split into two sub-clusters leaves different options on how to design the FLES network. InfiniBand fabrics are commonly constructed with fat-tree style architecture[23]. A generic example of a two level fat tree is given in Figure 4.27. The long connections between the FLES sites should be switch-to-switch connections (in contrast to connecting directly to an HCA on one or both sides). First, the current HCAs do not provide enough power budget for the currently available transceivers. It is possible that this will change in the future, but it would be unreasonable to limit choices. Second, being able to switch connections will allow data aggregation, load balancing, and fault-tolerance for the long connection and thus reduce the number of overall needed links.

The simplest network architecture is to split a single fat tree over both clusters. An example for such a design is depicted in Figure 4.28. The example is based on InfiniBand HDR (200 Gbit/s) technology. The given numbers are the maximum possible values for this specific configuration and serve for illustration purposes only. The final network will be scaled depending on the needs and available technology. The spine of the network and

[23]Technically constructed as a variant of a Clos network and not as a fat tree in a strict sense.

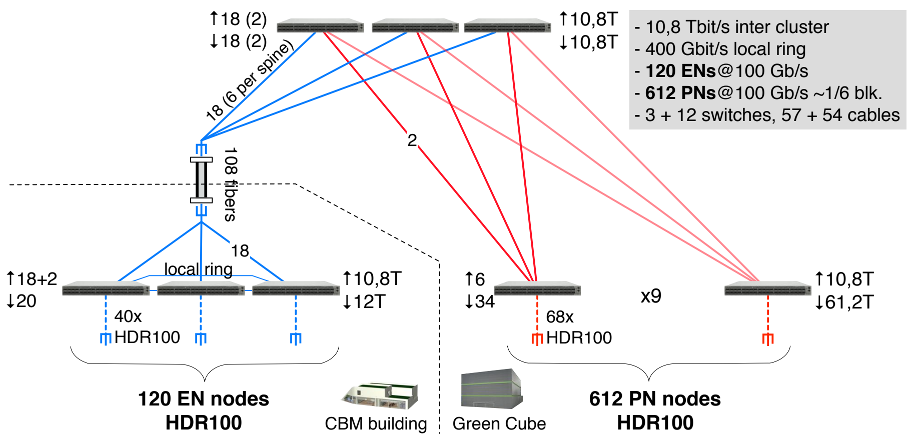


Figure 4.28: Possible network architecture connecting the entry and processing cluster via a common fat-tree network.

most leaf switches are situated in the Green IT Cube. Some of the leaf switches are placed at the CBM site and cross-connected with long fiber connections to the spine forming a fat tree.

Data flows primarily from entry to processing nodes and there are more processing nodes. Thus the network could be designed with an asymmetric blocking ratio with higher blocking on the processing site. However, this architecture has several disadvantages. The communication pattern imposed by the nature of the timeslice building problem requires any-to-any communication between entry and processing nodes. Furthermore, without explicit scheduling, all entry nodes target a single processing node at the same time. While it has been shown that these challenges can be handled, it requires a well-balanced network and fine-grain control over the routing patterns to gain good performance. Otherwise, the communication is prone to problems like head-of-line blocking. Considering the FAIR cluster is shared with other experiments and especially network resources might not be exclusively assignable to CBM also during CBM beam times this can pose a serious issue. Secondly, this architecture implies that there is no substantial local interconnect between the entry nodes without using parts of the processing cluster. This makes it impossible to run any CBM activities such as commissioning and cosmic campaigns when the shared processing resources are blocked by other users or are not available.

To circumvent these issues and increase flexibility and scalability a different network architecture is foreseen. An example for this architecture is depicted in Figure 4.29. This architecture foresees a fat-tree timeslice building network at the entry cluster site to perform local timeslice building. Instead of sending timeslice components directly to the memory of the appropriate processing nodes, timeslice building is performed locally among the entry nodes. Full timeslices are built to the memory of the entry nodes and forwarded as single packages to the processing node. This leaves the most critical communication patterns in the local, non-shared network. This local network could be directly connected to the Green IT Cube network similarly to the previous example. However, it is more practical to keep

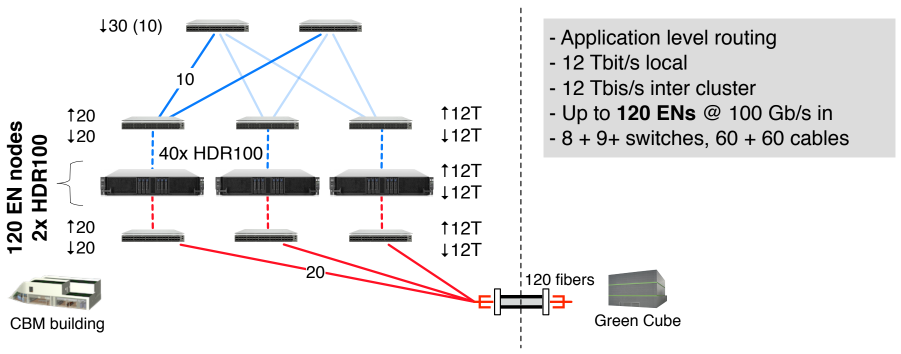


Figure 4.29: Foreseen network architecture featuring an independent timeslice building network at the CBM site with application level routing to a shared Green IT Cube network.

the networks separated and create this connection on application level. This allows fully independent control over both resulting networks, routing patterns, and communication protocols. The additional network ports needed can be gained cost-efficiently by utilizing dual-port HCAs. The switch layer terminating the long fibers can be scaled independently of the local network which makes it easier to scale to lower rates for initial commissioning setups.

Additionally, this option is very flexible in terms of network technology used within the FAIR cluster and to connect the two sites. In case another network, e. g., Ethernet is used in the Green IT Cube, the entry nodes can easily be equipped with this technology without the need for specialized hardware gateways. The additional resources needed to realize this architecture are reasonable. The given example provides similar connectivity and slightly higher bandwidth as the first example at the cost of two additional switches.

4.6 System integration

4.6.1 Discussion of failure modes

Building and commissioning a high-energy physics experiment is a highly complex task. As most systems are unique, without a full-scale prototype, some errors or instabilities will be introduced into the final setup. Also, the FLES entry stage serve as readout infrastructure in detector prototype tests, which cannot be expected to operate flawlessly. Therefore, the FLES must be able to cope with errors of other systems without affecting the stability of the FLES itself. During beam periods, CBM has to run reliably and continuously to record as much data as possible. It is essential to identify and handle failure modes efficiently to maximize system availability.

Redundancy and internal failures Implementing redundancy on the entry node level or further upstream is not a viable option. This would require some kind of switchable network between entry nodes and detector electronics, which is not feasible. Entry nodes, CRIs, and front-end links have to be treated as unique resources. Depending on the running scenario and exact failure mode, it can be acceptable to record data with only a subset of a detector subsystem. It is, therefore, crucial that a failing resource does not impact data from other parts of the detector or block the entire system. Upstream failures must not cascade downstream.

The FLES architecture assists this by clearly separating parallel elements. Input channels, as well as complete CRIs, can be operated independently from each other without global interactions between them. A failing CRI, for example, will not impact the readout of other CRIs.[24] Input synchronization is achieved implicitly via the microslice timestamp and thus does not need explicit internal error handling.

Failing or stalling data consumers, e. g., the timeslice building or analysis task will stop consuming data and eventually create back pressure. As the entry stage implements a closed back pressure path, no immediate handling is needed. However, the back pressure must be monitored and reported to the EDC to take countermeasures.

Failures of upstream components Failures of upstream components can be distinguished into two categories: Failure modes for which only the microslice content is affected, but the integrity of the microslice container is uncompromised, e. g., in case of failing front-end ASICs or data transport to the CRIs. And failure modes for which the microslice generation itself or delivery of microslices is erroneous, e. g., in case of a failing subsystem microslice builder.

The microslice data model allows handling the first category without additional measures, as it is data agnostic and all necessary information for data handling is available in the unaffected microslice descriptor. Any corrupt microslice content will not impact data handling within the FLES.[25] The defined maximum size of a microslice protects the FLES of being flooded with data in such cases (cf. Sec. 4.2.5). The data analysis software, which is aware of the expected data format, has to detect and handle such errors.

If the generation of microslice is erroneous, the situation is more complicated. If some microslice streams are created at a wrong rate or not received at all, the input buffers before timeslice building will fill up and eventually stall the entire system if the situation is not appropriately handled. Assuming such errors occur rarely, it would not be wise to introduce a sophisticated treatment of such a situation. The increase in system complexity would not be reasonable. Instead, the active data taking can be terminated and continued after the error has been corrected or after the faulty component was excluded. For such an approach to work, restarting data taking has to be reliable and fast. A matching concept for an efficient and modular system startup is presented in Section 4.6.2. If such errors

[24] Assuming the failure does not crash the PCIe subsystem of the node.

[25] In case the detector system is aware of corrupt data, it can set the appropriate flag in the microslice descriptor to signal such situations. The entry stage can relay the information to the EDC.

frequently occur, terminating the data taking is not an option, and the error should be handled dynamically. A feasible concept is to substitute missing or erroneous microslice streams with dummy data, i. e., to create empty microslices. This way, subsequent elements of the data path do not need any specialized handling capabilities. Without any data content, the system overhead of an empty microslice is negligible. Flagging the injected microslice allows to easily distinguish between injected and recorded data based on the microslice descriptors. This is important to be able to calculate the actual physical acceptance of the detector during the analysis. An additional out-of-band channel can be used to signal an error condition to the EDC system. Both information channels can be used by the QA to determine the global system state and potentially abort the run if the loss of data is considered severe.

In any case, a reliable failure-detection method is required. Especially failure modes in which microslices are created at a wrong rate, e. g., when a subsystem unnoticeably lost synchronization, can be difficult to detect. For that reason, the data model requires empty microslices and defines a maximum microslice delay (cf. Sec. 4.2.5). Microslices then serve as a heartbeat signal to determine if a source is healthy, even if the source has no data to deliver. The maximum delay allows identifying rate mismatches. Receiving no microslice within a given time span decidedly indicates that the corresponding data source is misbehaving.

4.6.2 Modular configuration and startup concept

Starting a large and complex experiment, that comprises inter-dependent subsystems, presents a considerable challenge on its own. All elements have to be configured properly and in the correct order to work together. For CBM, these subsystems include all detector FEE, the TFC, the EDC, and all components of the FLES. The detectors, for example, need time information from the TFC to function correctly. A common issue, especially during commissioning and in the early phases of the experiment, is that a detector subsystem is submitting either erroneous data or no data at all. This is, in particular, a problem if the start of data sending is synchronized between all subsystems and coupled to the start of data recording. In this case, the error stays undetected until recording is started and leads to a failing run. The required cleanup and restart procedure costs time otherwise used to record data. Coupling the systems also means that the error probabilities are combined, and the overall system becomes tedious to start.[26]

Additionally, it might not be possible to derive the exact state of a detector subsystem exclusively from configuration results, and an analysis of its data stream is needed to determine the quality of data. Having to start the full timeslice building and analysis chain for such a check is unnecessarily time-consuming. Instead, this functional check can often take place locally on the entry node. This asks for a method to enable local QA before timeslice building.

[26]The reliability of a serially coupled system is the product of the reliability of the individual components.

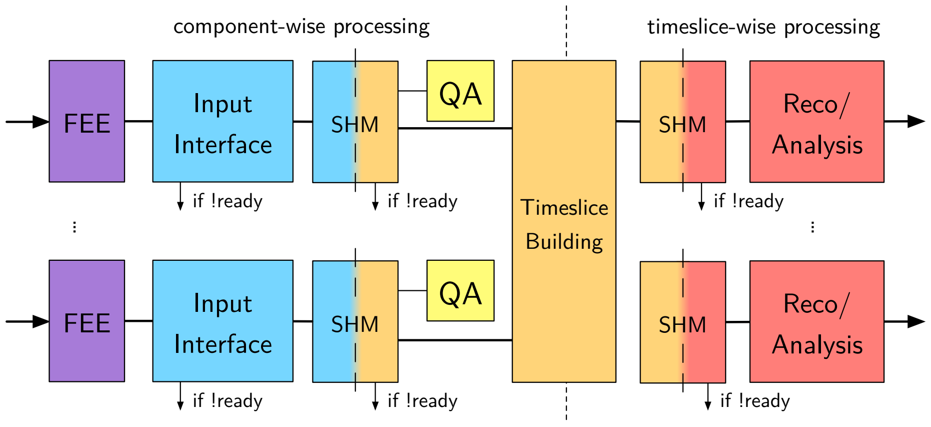


Figure 4.30: Separated data paths in the FLES. Timeslice building is the only stage that requires the full data stream from all detector sources. All other processing happens independently, either component-wise or timeslice-wise. Processes along the chain are decoupled via shared memories. If a consumer is not ready, data can be discarded in the previous stage.

With this in mind, a startup concept for the CBM data taking was developed. It exploits the free-streaming nature of the readout system, to minimize dependencies and couplings of individual components. It follows two main paradigms:

- Decouple the individual branches of the readout-tree. The state of one branch should not impact the states of other branches.
- Decouple data producers from data consumers. The state of a consumer should not influence the state of a producer.

Both paradigms go hand in hand. Figure 4.30 shows a schematic overview of a possible system configuration. As there is no central trigger mechanism, the readout channels already work independently. It is important not to introduce any artificial global coupling in later stages. The entry stage design maintains the separation by handling individual input components independently. However, for timeslice building data streams have to be synchronized globally. The microslice data model achieves this synchronization solely based on the microslice timestamp. It is not necessary to introduce additional synchronous events, e. g., a synchronous start of all input data streams. Streams can be handled without relying on stream history. This allows all branches of the readout tree to start independently of each other in any order. It also means that individual branches may be reset or reconfigured, while others are left in their current state.[27]

Similarly, the FLES data chain is designed in a way that no strict startup order is required. Shared memory interfaces can be utilized to separate different stages into individual processes. If sources are allowed to discard data when the consumer is not available, the coupling between producer and consumer can be eliminated. Consumers can be freely

[27]This requires that branches can (re-)synchronize with the TFC timing system without global operations.

attached and detached without any changes to the detector FEE or CRI configuration. In the configuration phase, any stage, upstream or downstream, may be restarted without influencing the other stages. Obviously, during data taking, a missing consumer has to be regarded as an error condition. With decoupled producers and consumers, a local QA process can attach to the CRI data publisher and execute any checks before starting the timeslice building. Once the state of the system is verified, the timeslice building software can seamlessly attach.

In summary, implementing both paradigms delivers a flexible, easy to start system with minimal dependencies. Any branch of the readout tree can be configured and reconfigured independently without influencing the rest of the system. Together with the modular startup order, fast reconfiguration cycles of the system are possible as only the affected components need to be configured or restarted, and the overall system availability is increased.

Chapter 5

Common Readout Interface

5.1 Introduction

The Common Readout Interface (CRI) is an FPGA-based PCIe card that serves as an input interface between the ReadOut Boards (ROBs) connected to the front-end electronics (FEE) and the input stage of the First-level Event Selector (FLES). The CRI is the last hardware stage of the CBM data aggregation concept (see Sec. 2.2.2) and a central element in the data and controls flow (see Sec. 2.2.3).

See Figure 5.1 for an overview of the CRI's major building blocks. Most CBM detectors use the GBTx [32] data aggregator ASIC. In CBM, data and control traffic is combined on the GBT links[1]. The CRI provides therefore three interfaces

- to the FLES via the FLES Interface Module (FLIM)
- to the EDC via the control bus bridge and DCA
- to the TFC system

[1]The design decision is described in Section 2.2.3.

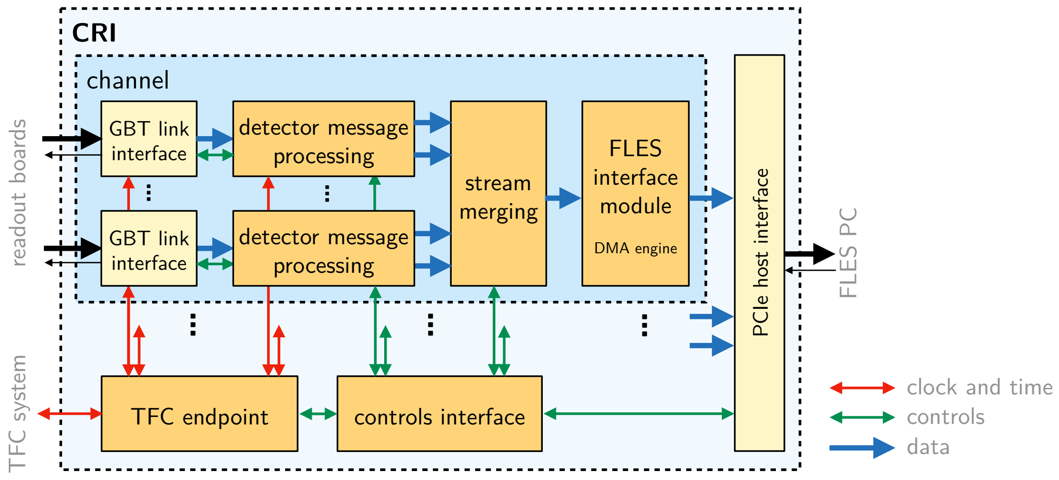


Figure 5.1: Common Readout Interface (CRI) overview

and acts as traffic multiplexer/demultiplexer:

- slow control command requests and potentially fast control throttling requests distributed via the TFC are multiplexed on the downlinks.
- the uplink traffic is demultiplexed:
  - the hit data stream is sent via the FLIM interface
  - the controls returns are sent to the controls system interface
  - status and alarm messages are buffered for inspection via the controls interface and potentially trigger the alarm system or are sent to the TFC for central *congestion* handling.

The CRI is also a central element in the clock and time flow (see Sec. 2.2.4). The 40 MHz system clock and time information is received from the TFC over an optical link. All GBT links run synchronously with the system clock[2] and are operated with deterministic latency in the downlink direction. This allows the system clock to be forwarded to the FEE and the local time counters in each readout ASIC to be synchronized via the downlinks. A timestamped hit message is sent over the uplinks when the FEE detects a detector signal above threshold. The CRI receives these hits, typically reformats them and, based on the received timestamp, packages them into microslice containers (see Sec. 4.2.1).

The CRI is a hardware component used by all CBM detector systems, regardless of whether they use GBT links or other link concepts as in the case of the RICH detector. Each readout chain requires a dedicated FPGA design because the raw data formats and associated data processing as well as front-end control protocols differ. However, the designs share a common base structure providing the interfaces to the central systems, i. e., FLES, TFC and controls via the control bus bridge and DCA.

Details about the CRI board as well as the core FPGA design components are presented in this chapter. The interfaces to the FLES and TFC systems are presented in the respective Chapters 4 and 6.

5.2 The CRI board

5.2.1 Hardware requirements

It is obvious that a CRI board should have a large number of optical link interfaces for the GBT links, a PCIe interface with adequate DMA throughput, and an FPGA large enough to implement all the required functions. There are numerous boards on the market that meet these generic requirements. However, the CRI role as the central element in the flow of clock and time (cf. Sec. 2.2.4) leads to very specific and unusual hardware requirements.

In a standard board design, the reference clocks of the communication links and the system clock of the processing logic come from different sources. The link reference clocks must

[2] 40 MHz GBTx frame rate and 4.8 Gbit/s raw link speed (cf. Sec. 2.2.4)

have very low clock jitter and are therefore routed on the board via dedicated clock lines and not via the normal FPGA clock tree. The clock concept is therefore part of the physical board design and cannot be changed afterwards via the FPGA configuration. Furthermore, in usual systems, different boards operate with their local clock frequencies which differ by the typical tolerance of crystal oscillators. This is sufficient for the usual communication protocols, since elasticity buffers and flow control ensure proper data transfer, and small latency fluctuations during transmission are not relevant.

The CRIs receive their clock from the TFC downlink, and the optical link interfaces and the processing logic operate with phase-locked clocks. In effect, the FEE interface logic on all CRIs runs fully synchronously. This requires a very specific clock distribution of the board. An additional difficulty is, that the jitter of a clock recovered from an optical link is much higher than the jitter required for a transmit clock[3]. The recovered clock must therefore be filtered with a low bandwidth PLL to reduce the clock jitter in a stage often called *jitter cleaner*. Last but not least, it must be ensured that the system starts up properly. The fast transceivers, like an MGT on a Xilinx FPGA, require that the reference clock closely matches the link frequency to properly receive an incoming link.

These requirements are similar across experiments using GBTx-based readout and led to the development of custom boards, e. g., the PCIe40 board used by LHCb and ALICE [57, 58] or the BNL-712 board used by ATLAS [59, 60].

The requirements can be implemented most easily with modern DDS-based clock managers. They provide very good jitter filtering and offer, unlike an analog PLL, a well-defined startup frequency[4]. This leads to a clock distribution with the key features

- the clock recovered from the TFC downlink is the input of a DDS-based PLL, which acts as a jitter cleaner and generates the required phase-locked clocks.
- the reference clocks of *all* optical link transceivers are driven by the clock manager. The well-defined startup behavior ensures that the TFC downlink locks when it comes up. All links send data with the same frequency.
- the required system clocks for the FPGA logic come from the same clock manager. This ensures that the link transceivers and the processing logic run synchronously and elasticity buffers can be bypassed, essential for the implementation of deterministic latency.

Further requirements for the CRI board result from the overall environment:

[3] The clock data recovery circuit in the link receiver is optimized for best data recovery, the internal PLL has a high bandwidth to find the optimal sampling point even in case of inter-symbol interference. This gives a good bit error rate, but high jitter. The link transmitters, in turn, must be operated with a clock with very low jitter to achieve the best signal quality. See [56] for tests with an early prototype.

[4] All-digital designs not only provide a well-defined startup frequency, but can also *freeze* a once-locked frequency in case the input signal is lost. This greatly simplifies the handling of start-up and loss of input signal. With conventional VCO-based PLLs, the frequency can drift very far from the nominal value. The GBTx uses a VCXO-based PLL with a very narrow pull range to solve this problem.

- The optical connection between the experiment and the server room is implemented with OM4 fibers in MTP-24 bundles (cf. Sec. 5.5). Optical transceivers must support MTP bundles and the GBTx bit rate of 4.8 Gbit/s.
- The optical connection to the TFC consists of a single fiber pair. An additional optical transceiver, such as an SFP, is required for this connection.
- To minimize porting and maintenance efforts, the FPGA chip should be compatible with the GBT-FPGA core [35], which is available from CERN in an open GitLab repository [61].
- The uplink traffic in the GBT link containing the readout data is usually much higher than the downlink traffic. Therefore, the GBTx is often used in a configuration, where a single "master" GBTx connected to both uplink and downlink fibers is accompanied by additional "slave" GBTxs connected only to uplink fibers. Dedicated radiation-hard optical components – VTTx (with two transmitters) and VTRx (with one transmitter and one receiver) [34, 62] support that configuration. Additionally, the GBT-SCA ASIC [63] provides the I^2C communication between the master and slave GBTxs. ROBs with a single GBTx and a VTRx are referred to as ROB1, while ROBs with three GBTx, a GBT-SCA, a VTRx, and a VTTx are referred to as ROB3. These connection topologies must be supported by both the optical connection system and the CRI board.

5.2.2 The CRI1 prototype board

In absence of the final CRI board, the development and verification of the individual subsystem readout chains is based on the BNL-712 v2 card [59], which was specifically designed at BNL for the termination of GBT links and is used by the ATLAS experiment for the FELIX system [60]. This board serves as a CRI prototype and has the CBM internal designation CRI1. The CRI1 board (see Fig. 5.2) is a 16-lane PCIe Gen 3 board equipped with a Kintex UltraScale `XCKU115` FPGA. This FPGA is divided into 2 Super Logic Regions (SLR). To efficiently transfer data from both SLRs, each of them implements a separate 8-lane PCIe interface. A PCIe Switch (`PEX 8732`) aggregates these interfaces to a single 16-lane PCIe Port to the host. Eight Avago MiniPOD transceivers surrounding the FPGA provide a total of 48 bidirectional optical links. These links are connected to two MTP-48 connectors present on the PCIe I/O bracket.

In the deployed version, only 47 MGTs are routed to the MiniPods. One MGT is routed to a Timing Mezzanine Card (TMC) socket, which allows the card to be equipped with different timing interfaces. Two different TMCs are currently used in mCBM: a `TTC-PON` type adapter[5] to operate the CRI1 as TFC Endpoint, and for usage as a TFC master, a modified adapter[6] to synchronize with the GSI/FAIR accelerator's WhiteRabbit infrastructure (cf. Sec. 6.3).

[5]the SFP cage is used for the TFC link

[6]equipped with two `SN65LVDS1` LVDS drivers used as receivers for PPS and 10 MHz clock inputs

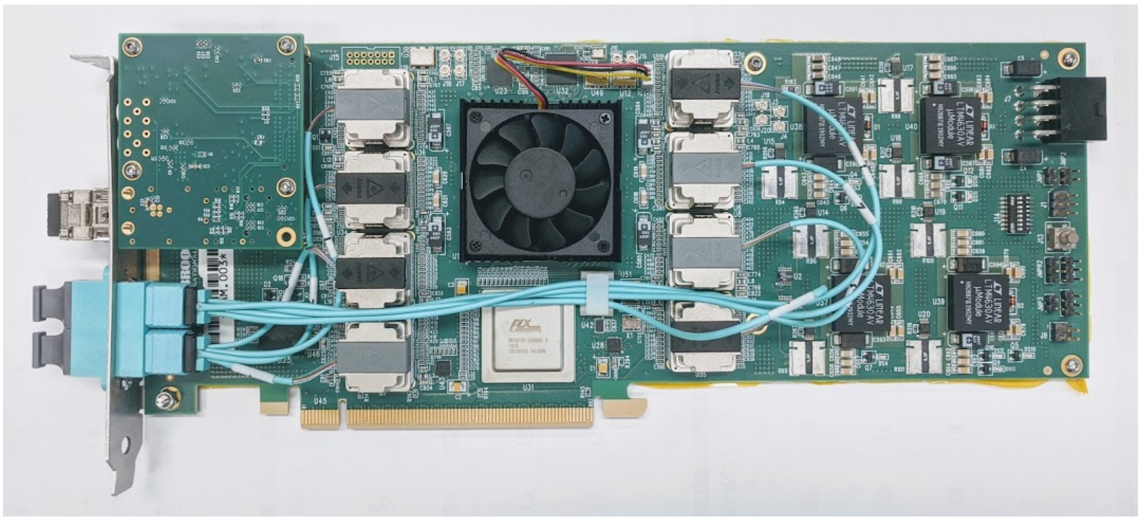


Figure 5.2: Photo of the CRI1 board (BNL-712 v2). The Kintex UltraScale FPGA is hidden below the black fan. It is surrounded by 8 Avago MiniPod transceivers, which interface to the optical fibers. The optical coupler on the left hand side consists of two MTP-48 connectors. The mezzanine PCB in the top left holds the interface to the TFC system. A PCIe switch combines the two SLR of the FPGA to the PCIe Gen3 x16 interface. The board is powered by voltage converters located on the right hand side of the PCB.

The CRI1 uses the SiLab `Si5345` clock manager chip. In the configuration used in CBM, it receives the clock recovered from the TFC downlink, acts as a *jitter cleaner*, and generates the reference clocks for the optical links as well as the system clock for the FPGA logic. This clock manager is a DDS-based all-digital design and perfectly fulfills the clock system requirements outlined in Section 5.2.1 (see also [64]).

CBM participated in a joint production run of the BNL-712 at BNL and acquired enough boards for the operation of mCBM and the ongoing development of the readout chain. The CRI1 prototype board is in use in mCBM since 2021 and will be used for the TFC system (cf. Chapter 6) and possibly for one of the smaller detector systems in the final CBM setup.

5.2.3 The CRI2 production board

Some components of the CRI1 board, most notably the Avago MiniPOD transceivers, have been declared end-of-life. Therefore, the planning of a successor board was started, the CRI2, based on the experience gained from CRI1. The CRI2 has a very similar overall concept as the BNL-712 board, but is based on newer components. Since the total cost of the entry nodes is dominated by the cost of the CRI boards, optimizing the overall cost per GBT link is an important development goal. The current plans have a design target of 36 GBT links and the following key features:

subsystem	ROB3	ROB1	GBT links	CRI1	CRI2	CRI boards
BMON	–	16	16	1		1
MVD	40	–	120		5	5
STS	576	–	1728		72	72
RICH	–	–	–	8		8
MUCH	216	120	768		33	33
TRD[8]	544	–	1632		68	68
TOF[9]	–	564	564		25	25
PSD	–	–	16	1		1
TFC	–	–	–	6		6
total	1376	700	4844	16	203	**219**

Table 5.1: Readout components per subsystem

- use Samtec FireFly optical transceivers
- use 3 MTP-24 connectors, offering optical connectivity for 12 ROB3 or 36 ROB1
- use a Xilinx Kintex UltraScale+ `XCKU15P` FPGA with 44 GTH (16.3 Gbit/s) and 32 GTY (32.75 Gbit/s) transceivers
- use 36 GTH for 3 × 12 GBT links
- use 1 or 2 GTH for the TFC connection
- use 16 GTY for the PCIe interface (Gen3, single x16 or dual x8 [7])
- use a clock manager similar to the `Si5345`

This board is less complex than the CRI1 in both hardware PCB design and FPGA design because the FPGA is not split into multiple SLRs and no PCIe switch is used.

5.2.4 Production planning

The full readout system of CBM will consist of 219 CRI boards. Table 5.1 summarizes the foreseen readout components for all subsystems.[8,9] Small subsystems like RICH and TFC intend to use the already acquired CRI1 cards, while larger subsystems like STS require large quantities of boards, which make a CRI2 production necessary. The required number of CRI boards depends on the readout architectures of the individual subsystems, which is explained in more detail in Appendix B.

Based on the current design, we have evaluated the consumption of the available logic resources of the designated FPGA. As expected, the resource consumption scales heavily

[7]dual interface for use with PCIe bifurcation (cf. Sec. 5.3)

[8]The numbers refer to the baseline configuration of a full TRD-1D. In case the inner zone is constructed using the alternative TRD-2D design, about 30 % additional hardware is required in the readout chain. In this case the TRD-1D plans to use 48 (cf. Sec. B.2.1) and the TRD-2D up to 26 CRI2 boards (cf. Sec. B.3.1).

[9]The numbers do not include the BFTC detector foreseen in the innermost part of the TOF wall. We estimate 3 additional CRI2 boards will be required to read out this system (cf. Sec. B.4.1).

with the number of used optical links. With an envisaged number of 24 links, the typical utilization of the available LUTs and the BRAM will be less than 70 %. This estimate is, however, conservative.

One should note that for the current FPGA design, a few features are still missing, e. g., the alarm handling and the fast control. On the other hand, there is room for optimization, e. g., the usage of URAM instead of BRAM so that we expect that the utilization can be further reduced. This would allow the increase of the number of links up to that limited by the hardware. The granularity of the systems which use the ROB3 is three links, so a number of links of 27, 30, 33 and 36 would be possible, and for the TOF, which uses ROB1, any number of links may be chosen. This will allow us to aim for the most cost efficient solution after some further investigation and development,

It is important to realize that no upgrade concept is needed for the CRI2. The function of the CRI is to serve as an interface between a well-defined number of GBT links (4.8 Gbit/s) into the entry nodes. As long as the CRI2 to be implemented meets these requirements, no further improvement in performance can be achieved through further upgrades.

5.3 FPGA design prototype

The CRI's FPGA design is the major part of the CRI, defining most of its functionality. The CRI fulfills the same task for all CBM detector systems. Consequently, the required FPGA designs for the individual systems overlap considerably in their functionalities. Figure 5.1 gives an idea of the needed components. To limit the development and integration effort, the design paradigm is to share as many components as possible between different designs. Naturally, all designs share the interface modules to the central systems, i. e., the FLIM, the TFC Endpoint, and the controls interface engine. The common controls interface implies that the modules also share a common on-chip bus, a Wishbone bus in this case, and a common controls driver. The front-end interface, detector message processing, and stream merging are inherently subsystems specific as they implement communication and processing specific to the detector architecture and front-end ASIC. However, also here common modules can be found, such as the GBT interface used by most of the detectors or the ASIC communication protocol engine HCTSP shared between the SMX and SPADIC ASICs (and thus by STS, MUCH, and TRD).

Many components of the FPGA design, especially the time synchronization of the front ends and the processing of detector messages, are far from trivial. To prove the feasibility of the concept, CRI1 prototype FPGA designs for all foreseen subsystems have been implemented and are continuously used in the mCBM demonstrator (cf. Chapter 7). Wherever practical, the components have been written in a generic fashion to allow an easy transition to the CRI2 board.

As mentioned previously, the CRI1 FPGA features two SLRs and the board routes one PCIe interface to each of them. Naturally, the CRI1 FPGA design has to follow this partitioning. The full data and control path is duplicated for each SLR. Only the TFC

Endpoint and other unique resources are instantiated once, and the respective signals cross the SLR boundary. Duplicating the data path into two independent portions is easily possible because, for resource efficiency, it is not foreseen to merge all inputs into a single microslice stream, but to use multiple FLIM DMA channels. These channels are designed to work independently of each other and do not need to share a common PCIe interface (cf. Sec. 4.3.3). Even without the constraint from multiple SLRs, it can be beneficial to employ a similar partitioning in the CRI2 design. Splitting the available PCIe lanes into two independent interfaces with half the width can substantially relax the demands on the data multiplexing before the PCIe core. To support such a case without extra components like a PCIe switch, the board design can make use of PCIe bifurcation. Whether this option is more efficient is subject to ongoing research.

The following sections give a more detailed overview of the FPGA design, with a focus on the common modules. The FLIM is described in Section 4.3. Details about the subsystem-specific design prototypes can be found in Appendix B. Unless explicitly noted, the information applies to both flavors of the CRI board.

5.3.1 The PCIe host interface

Communication with the host is implemented via a PCIe interface [49]. All PCIe protocol layers up to the transaction layer are provided by the FPGA's PCIe core. The current CRI1 core is interfaced via multiple AXI4-Stream interfaces, essentially exchanging PCIe TLPs extended with some metadata, with the user logic. All additional logic like bus bridges mapping the PCIe address space to an on-chip bus or DMA engines has to be implemented in user logic. While the details of the core interface are in general FPGA-specific, the basic idea of providing a TLP-based interface is present in most FPGAs. Consequently, any developed logic can usually be ported to a new FPGA with reasonable effort.

In the case of the CRI, the PCIe interface provides host communication for two major tasks: the data path to the FLES and the control system steering the subsystem logic and front-ends. Requirements for these tasks differ widely. The FLES interface requires a high-throughput DMA engine to provide sufficient bandwidth (cf. 4.3.4). For the control interface, data transfer via PIO is sufficient, but it must be able to handle different subsystem designs and communication with possible faulty components connected to the FPGA (cf. 5.3.2). While both tasks use the same physical PCIe interface, it is favorable to separate them logically. This allows for better optimization and more independent development. The CRI design implements this separation via different PCIe physical functions[10]. In addition to a separated address space, which could also be achieved by mapping to different BARs, this allows the binding of different device drivers to each function. With this, a custom kernel driver is only needed for the FLES interface. Without the need for DMA buffers, the control portion can be based on the *generic PCI UIO driver*

[10]A PCIe physical function is a full-featured PCIe function which is discovered, managed, and controlled like any other PCIe device.

which is provided by any modern Linux operating system. Keeping the custom kernel driver focused on a single task allows for better maintainability.

From the point of view of FPGA logic, the PCIe core still provides the same TLP interface, only with the appropriate header bits set. This is not different from splitting into different BARs. An arbiter module is used to route incoming requests (i. e., PIO) to their appropriate destination and stream merge outgoing completions. Outgoing requests (i. e., DMA) are routed directly to the core as they are only used by the FLES interface.

5.3.2 Controls interface

The CRI has two control realms (cf. Fig. 5.4): the FLIM configuration bus (cf. Sec. 4.3.3) and the control bus for all other register resources, which is described in this section. The control bus is implemented as a Wishbone bus.

Each CRI design is based on the same common infrastructure entities so that the DCA and detector control software can handle each design with the same interface and basic methods. All these entities are stored and maintained at a central place in a repository (cf. Sec. 5.3.5) to ensure that they are always up-to-date.

The address space of the control bus is divided into two regions. The lower part of the address space contains registers at fixed addresses[11] in an area called Zeropage, which is further described below. The upper part contains registers (and memory) that are different for each design. During the compilation process of the FPGA design, these addresses are dynamically created together with address decoders and stubs used in the FPGA design and configuration files used by the access software. This ensures that the software always uses the correct address configuration. Since the control bus is implemented as a Wishbone bus, the AGWB tool is used as an address generator, which is further described below.

Control bus interface

The control bus interface is a system of bridges that convert the controls TLP stream provided by the TLP switch into control bus transactions. This bridge system provides two independent interfaces: one is used for the Zeropage, and the second one is used for the dynamic part, which is generated via the AGWB. With this concept one ensures that failures of the control bus itself can be monitored via a Zeropage access.

The key design factor for the bridge that handles the AGWB generated main part of the Wishbone bus was error handling, especially for accesses to Wishbone bus addresses that are not connected to an object. In a simple *direct-mapped* approach, where the PCIe PIO accesses are translated 1-to-1 into Wishbone accesses, the only way to handle such a condition is a *bus error* on the PCIe side, which would be forwarded as a *segment fault signal* to the software. This is very difficult to handle in a modular fashion, especially in a multi-threaded process like the DCA.

[11] these addresses are equal per convention for all designs

The bridge implementation takes advantage of the observation that the Wishbone bus in the CRI uses 32-bit data words, while the PCIe transactions and the controlling software can easily handle 64-bit data words. This allows error control and status information to be placed in the upper bits of a 64-bit data exchange. The bridge maps each Wishbone bus address to a 64-bit PCIe address and adds a few special registers for error handling in the Zeropage. A write transaction is implemented as posted write, but Wishbone errors are recorded in the bridge. A read transaction returns the value read from the Wishbone bus and the error status of that read and previous writes in the upper bits of the 64-bit data word returned over PCIe. This allows most Wishbone access sequences to be handled with one PCIe PIO per Wishbone access, which is essential for good performance. Only the case of a write operation, where the error status must be immediately known, requires two PCIe PIOs. The application software uses a very thin library that encapsulates the bridge protocol. Wishbone reads and writes are done via methods calls[12]. They come in two variants, which either return the error status as a return code or throw an exception in case of an error. In most cases, the latter leads to more compact code and modular error handling.

Wishbone address generation

One of the obstacles is that the register set differs for each design. In addition, each design could be compiled with different configurations. Handling the different register sets by an external database and/or scripts could be tedious, as one must ensure that they are always consistent with the current designs. In this context, it should be noted that changing information in different places should always be avoided.

In addition, handling all blocks and registers in a single component appeared to be extremely inconvenient. Routing of signals connected to those registers between the blocks and through the multiple hierarchy levels was messy and error-prone. To avoid this, the registers should be located close to where the connected signals are used. This means, that a hierarchy is always recommended.

Therefore, the design flow uses the “Address Generator for Wishbone” (AGWB) [65]. The main function of the AGWB is to create control and status Wishbone registers, as well as hierarchy blocks via an XML description.

Zeropage

Besides error information about failed transactions on the control bus and some test and identification functions, the most important part of the Zeropage is the storage of the “BuildInfo”, a ROM in ASCII format at a fixed address, which contains information about the design loaded into the FPGA, the compile time, the git commit id, and the git path. Using this information, the correct address map generated by AGWB can be

[12]The software overhead of method call and other data handling is negligible compared to the time of a PCIe PIO transaction.

located and used to configure the DCA. In addition, the Zeropage contains information about the FPGA device DNA and the state of the CRI alarm system.

CRI alarm system

As the FPGA has limited logic resources, it is important that any actions that can be performed in software are not implemented in hardware logic. One such example is a recovery of a link, where certain steps must be executed in a specific order, but precise timing on the system clock level is not required. This must not be confused with monitoring, which also is done frequently but does not require an immediate response or action. Therefore, a common and specified CRI alarm system is provided, consisting of a bus connected to all relevant building blocks and an interface in the Zeropage monitored by the DCA. Together with the corresponding handling of each type of alarm, this forms the global alarm system.

To save resources on the FPGA, the bus itself is implemented as a sequential daisy chain that starts and ends on the Zeropage. It is important to note that the alarm bus can only transmit one alarm at a time, which is continuously forwarded to the Zeropage ("ringing") until the alarm is muted by the node. The first level alarm handler in the DCA will identify the alarm source node, mute the alarm, and queue a request for the second level handler. This frees up the alarm bus and allows multiple alarms from separate sources to be processed in parallel.

The alarms have several priority levels:

warn: A situation that requires attention but has no immediate impact on data taking.

error: A situation that can be resolved by software interaction but does not require a reset of the entire CRI or a stop of the run, e. g., a link has been lost and can be resynced.

fatal: A situation where the CRI must be reset or the data is completely corrupted, e. g., lost TFC synchronization.

It should be noted that the CRI alarm system bus is not designed to carry a large amount of data, only basic source information. More detailed information must be retrieved by the handler through registers of the alarm source node that return the node state. An alarm response time on the ms scale is considered sufficient. Therefore, the control bus interface does not provide an interrupt mechanism, and the alarm handling is based on polling of the Zeropage with an appropriate rate.

5.3.3 TFC Endpoint module

All interaction between CRI and the TFC network is handled by the integrated TFC Endpoint module, which abstracts from the internal TFC-related mechanisms. The Endpoint is accessible by other modules and provides the TFC clock and time counter. It also enables the CRI to receive fast control commands and to send *congestion* status data to the TFC Master.

The clock manager hardware on the CRI board guarantees that all clocks are continuously available, regardless of the synchronization state, and are glitch-free even during synchronization state transitions (cf. Sec. 5.2.1). Upon successful synchronization, they are phase-locked with the TFC Master clock.

As to operation modes, the Endpoint can have three states:

free-running/standalone: the CRI runs on a local free-running root clock. This is the default state at startup, before the Endpoint synchronizes with the TFC link. The clocks have a frequency close to the nominal value, the time counter starts from zero or a value set via the controls interface. The state can also be forced using corresponding control bus registers for standalone operation.

synchronized: the CRI is synchronized with the TFC link, the clocks are phase-locked to the TFC master clock, and the CRI time counter has the correct time. The transition into this state is signaled via the CRI alarm system (cf. Sec. 5.3.2) and the DCA to the EDC, which in turn will ensure that all on- and off-board components will be initialized, starting with the GBT links and ending with the FEE ASICs.

desynchronized: the CRI has lost synchronization with the TFC link and is running on the local free-running clock. The clocks have a frequency close to the nominal value, the time counter is incremented and slowly deviates from the nominal time. The state is interpreted as a fault, and the Endpoint does not attempt to resynchronize until explicit user intervention via the control bus. The transition into this state is also signaled via the alarm system to the EDC, which will start recovery procedures.

5.3.4 The GBT interface core

GBT-FPGA [35] is an FPGA core that enables communication with the GBTx chip. It was developed at CERN as part of the GBT project [31]. Currently, it is available as an open source IP core [61]. The official repository provides core components that implement the transmission and reception data paths (both in standard and deterministic latency variants) as well as the configuration of multi-gigabit transceivers (MGTs) for several FPGA families. Another repository [66], which is a part of the GBT project, provides a slow control interface for GBTx chips.

Latency-optimized multi-link design

The GBT-FPGA core used in the CBM experiment operates in latency-optimized mode for downlinks, which is essential for the synchronization of the front-end ASICs. The data is timestamped on the front-end ASICs, so the latency of uplinks is not important and GBT-FPGA can work in standard mode in this direction.

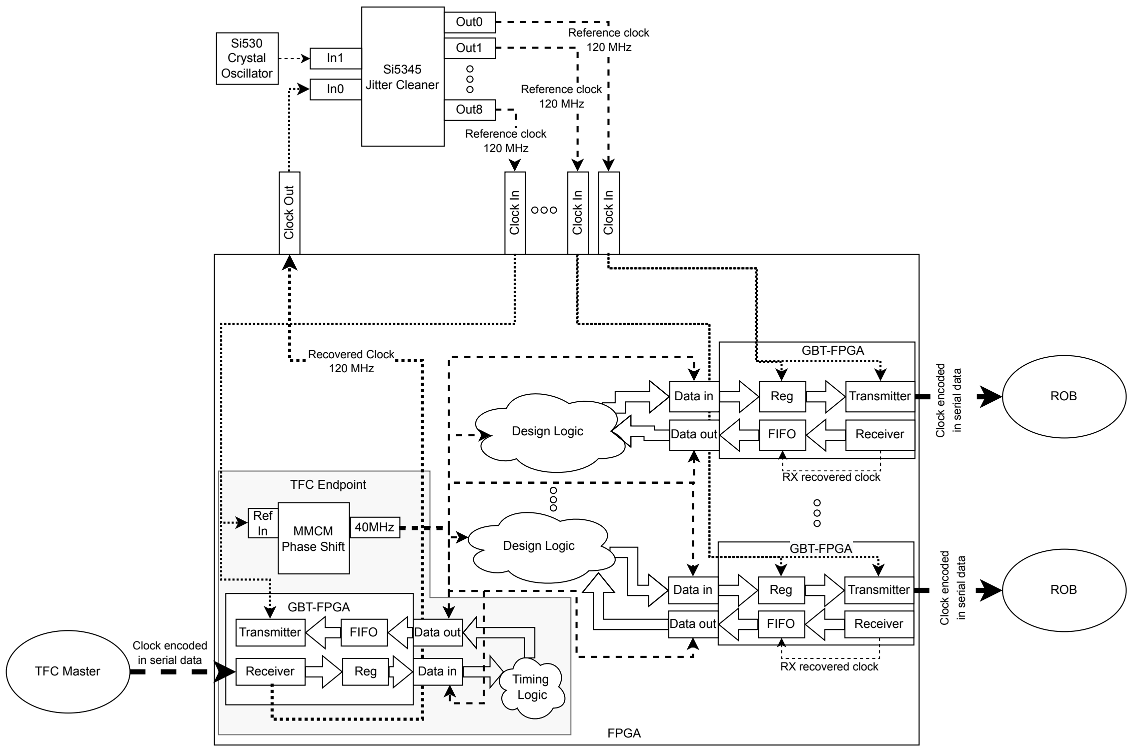


Figure 5.3: Routing of clocks and flow of data in the CRI1

In latency-optimized mode, the GBT-FPGA core in the TFC Endpoint produces a recovered clock of 120 MHz, which is synchronized with the clock received from the TFC downlink (see Sec. 6.3). The recovered clock drives the `Si5345` zero-phase-shift jitter cleaner, which produces the low-jitter 120 MHz reference clocks for transmitters in the GBT-FPGA cores (see Figure 5.3). In addition, the 120 MHz reference clock is delivered to an FPGA MMCM that generates a 40 MHz frame clock that is common to all GBT-FPGA cores. The MMCM provides a fine-phase alignment functionality. A dedicated software routine uses the `Rx header` signal from GBT-FPGA to ensure the correct phase of the generated 40 MHz word clock[13]. The word clocks and the frame clock are synchronous due to a single source, but may be slightly phase-shifted due to fine differences in trace and buffer latencies. Since a CDC between frame and word clock would be unsafe in such a design, the frame clock and the word clock are shifted by a constant phase. The value of this shift was determined by a calibration procedure that tested different phase shifts in search of values that produce CDC errors in GBT-FPGA gearboxes. This calibration value was selected for the whole experiment.

In the presented design, the latency between the common frame clock of the CRI and different GBTx chips may differ slightly due to phase offsets between the different word clocks. Phase shifts of less than a nanosecond are expected, which is below CBM synchro-

[13]For a 40 MHz signal generated from a 120 MHz reference, there are three possible phases. Only one of them is correctly synchronized with the data stream transmitted via the TFC link.

nization requirements. As a safety measure, the GBT link errors are monitored since an increased error rate may indicate CDC errors.

Adaptation of GBT-FPGA for CRI

For the usage in CBM, several minor improvements to the official core have been made. Most of them focus on additional control and diagnostic methods suitable for CBM use. Several status and control registers have been instantiated in the central core wrapper. Some of the registers are instantiated per link (error counters and status registers), while others are common for the whole bank (configuration and controls). Other modifications were introduced to simplify debugging. The pattern generator and checker have been modified so that custom test patterns can be specified via a control register. It is also possible to disable data scrambling and descrambling, an option useful for loopback tests.

In addition to the improvements to the core itself, a dedicated wrapper component has been created that instantiates the GBT-FPGA and the GBTx slow control interfaces in configurations specific to the readout boards used in the experiment. Such encapsulation simplifies sharing of the core between different detectors.

5.3.5 Design integration and workflow

The CRI FPGA designs are composed of common and subsystem-specific modules. To ease the integration of all needed components into functioning designs and promote component sharing between designs, all modules are consolidated in a common repository.[14] All FPGA designs are built from this common repository. This allows for centralized management and integration of designs and is especially advantageous when common modules need to be updated or fixed. All changes can easily be checked against all designs and in most cases do not require the involvement of all subsystem designers.

The development workflow follows the common Git fork and pull flow. Designs used for data taking are required to come from the official CBM hardware repository. The full FPGA build flow for all designs is automated and requires no human interaction. This enables easy integration into any CI/CD application. Currently, the GitLab CI/CD is used in conjunction with Docker containers that provide everything needed for the CRI build flow. Not only does this CI/CD integration enable automated reviews of all designs, the resulting FPGA configurations are also used as the official designs for data taking, guaranteeing full traceability of the used module versions. In addition to the FPGA designs, the build flow produces the register mappings for each design, which are centrally available as CI/CD artifacts.

[14]External components shared with other projects are added via submodules.

5.4 The device control agent

The *Device Control Agent (DCA)* and the FLIM data publishing server (cf. Sec. 4.3.7) are the only software components described in this TDR. In both cases, the software design is closely intertwined with the FPGA design with the goal of only implementing performance or time critical functions in the FPGA and doing the rest in software. Such a *hardware-software co-design* helps to optimize both hardware cost and personnel cost.

The task of the DCA is to control all logic on the CRI board except for the FLIM section and DMA data path (see Fig. 5.4). The DCA acts as a *user-space driver* that has exclusive access to the CRI device registers and provides a high-level interface for the higher layers of the CBM control software. A DCA process runs on each entry node, manages all CRIs of that entry node, and serves as a *gateway* between the CRIs and the higher control software layers, collectively referred to as EDC, that typically execute on other nodes in the DAQ network. The key architectural features are:

- the Wishbone control bus is accessed via the CRI controls interface (cf. Sec. 5.3.2) with PCIe PIOs.
- the DCA is multi-threaded and can handle multiple CRIs as well as CRIs with multiple Wishbone buses (e. g.: CRI1 has a Wishbone bus for each SLR). Each Wishbone bus has a dedicated worker thread.
- the DCA exports a remote procedure call (RPC) interface for the higher layers of the controls stack.
- RPC error handling is done via exception forwarding.
- the DCA provides a high-level functional interface, all register transactions that must be *atomic* are encapsulated in a method.
- for debug purposes, low-level direct register access is provided too.
- the DCA can serve multiple clients simultaneously, allowing control and monitoring functions to be implemented as separate processes.

The DCA is implemented in C++ while the higher controls layers are currently implemented in Python. The RPC interface is implemented with `MsgPack` [67] over `ZeroMQ` [52].

The DCA software is structured in a framework part common to all readout chains and an easily extendible set of software modules, which represent devices. The framework part provides all required common services like RPC handling, thread and object creation, as well as logging and monitoring interfaces. The device classes describe either FPGA-*internal* functional units, like a VHDL entity, or FPGA-*external* entities, which are accessed over communication links, such as an I^2C device on the CRI board, a GBTx reached over a GBT link, or a readout ASIC reached over an e-link which is transported over a GBT link (see Fig. 5.5). All device classes inherit a large set of common functionality from a base class.

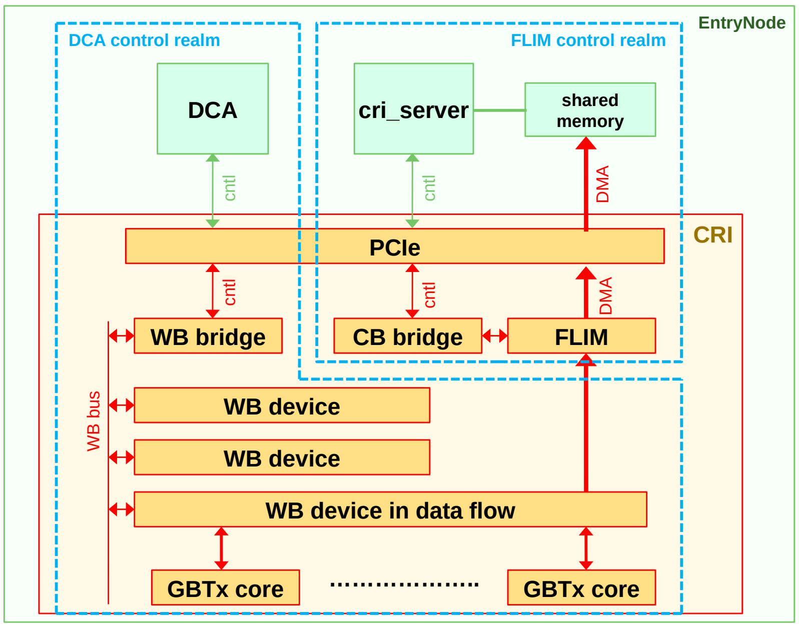


Figure 5.4: The CRI has two control realms: the FLIM configuration bus (cf. Sec. 4.3.3) controlled by the FLES data publishing server (cf. Sec. 4.3.7), and the Wishbone control bus for all other register resources controlled by the DCA.

All setup and configuration of a DCA process is done via RPCs, the DCA does not read configuration files. After DCA startup, only a service object is available to scan the system and search for CRI devices. Control software requests a CRI device scan via RPC and creates a worker thread and a device object for each detected CRI and each Wishbone bus. In the next step, the control software inspects the Zeropage (cf. Sec. 5.3.2) of each Wishbone bus. The "BuildInfo" is used to determine all further information about the specific FPGA design, usually via the artifacts management of the build flow (cf. Sec. 5.3.5). A key element is the AGWB-generated system description (cf. Sec. 5.3.2) with a complete description of the Wishbone register structure. Based on this information, the control software finally creates all device objects for FPGA-*internal* entity instances, and with further information from configuration management, also the device objects for FPGA-*external* components.

All device objects have a unique identifier and are associated with the worker thread of the corresponding Wishbone bus. For FPGA-*internal* objects, this is the bus to which the entity instance is connected. FPGA-*external* objects are always reached via a communication bridge (see Fig. 5.5), such as an I^2C controller or a GBT link controller, which is an FPGA-*internal* object. FPGA-*external* objects are therefore associated with the worker thread of the communication bridge.

RPC requests sent to the DCA contain the device class name, the unique identifier of the device object, the method name, and the `MsgPack`'ed argument list. The first stage of RPC processing in the DCA, the corresponding C++ object and method are determined

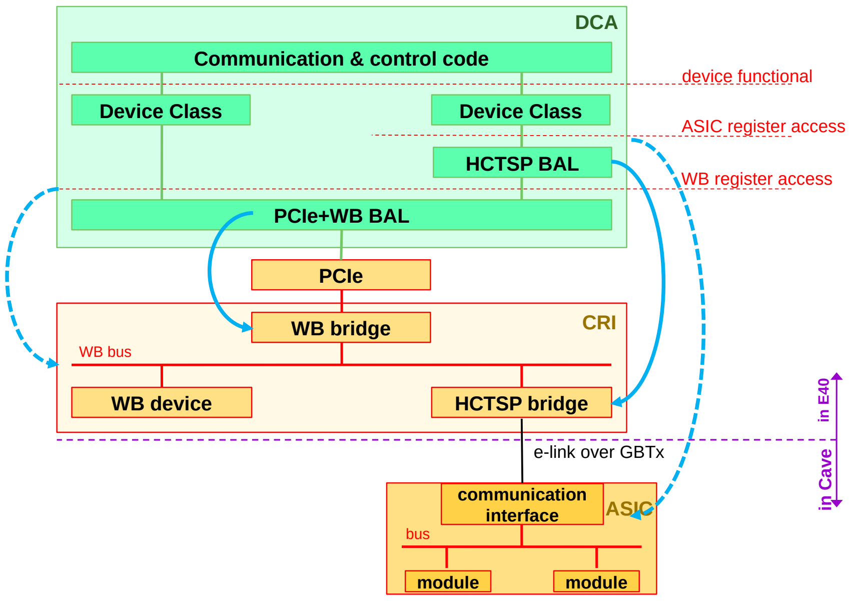


Figure 5.5: The DCA class/object model reflects the structure of FPGA *internal* modules, including communication bridges, and FPGA *external* components that are reached via communication links, such as a readout ASIC reached over an e-link transported over a GBT link. In the latter case, the connection to an SMX or SPADIC is shown, where the HCTSP controller in the CRI is handled by a corresponding bridge abstraction layer (BAL) in the software stack.

and the request is forwarded to the proper worker thread for execution. This allows the Wishbone buses to operate independently and concurrently, but guarantees that RPCs for a given bus are handled sequentially, and thus the atomicity of transactions on a given hardware object.

The DCA supports queued and scheduled execution of device object methods in addition to pure RPC-driven processing. The event loop of each worker thread interleaves these activities with RPC processing so overall atomicity is maintained. Such activities are usually started via an RPC request and can be used, for example, to implement periodic monitoring tasks. To further support this use case, the DCA framework includes an abstract interface for time series databases and currently concrete sink implementations for InfluxDB V1 and V2.

Last but not least, the DCA contains the first-level handler for the CRI alarm system (cf. Sec. 5.3.2). In simple cases, the alarm handling can be performed locally by the DCA with the mechanisms described in the previous paragraph. If the handling is complex or requires a large set of parameter values, the DCA will forward the alarm to a registered handler in the control system.

In any case, all anomalies and significant events are logged by the DCA. The DCA framework includes an abstract logger interface and currently concrete sink implementations

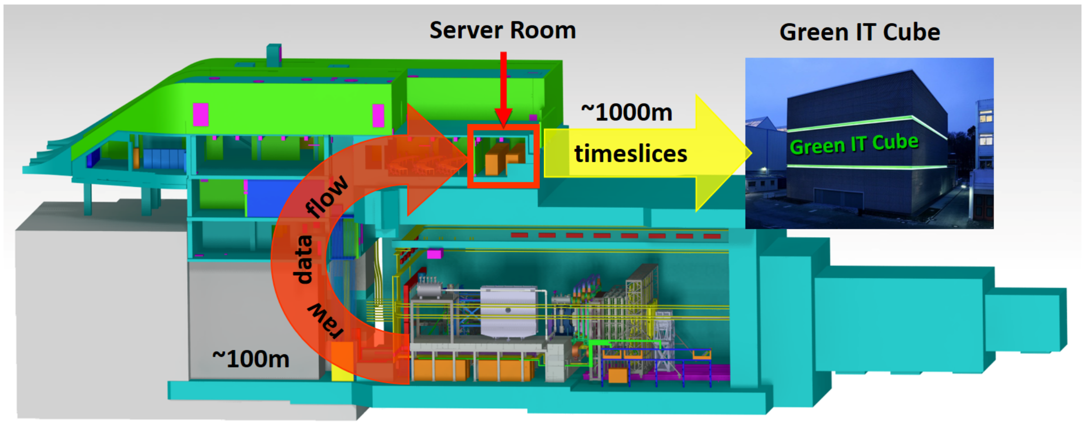


Figure 5.6: The CBM building hosting the experimental area (below ground level) and the server room.

for rsyslog and InfluxDB V1 and V2. The logger messages are structured and have a detailed set of keys, including the object identifier and a message id, for easy analysis and aggregation.

5.5 Physical connections and layout

In most cases, the data produced by the detector front-end electronics in the experiment cave (level E10 of the CBM building) is aggregated by GBTx data concentrator ASICs and sent over optical links to the FLES entry nodes in the server room located on top of the experiment cave (in level E40 of the CBM building, room E40.017). These links run at 4.8 Gbit/s, use OM4 multi-mode fibers and are terminated by CRI2 or CRI1 PCIe cards[15] in the entry nodes. After suitable preprocessing, the data is sent from the entry nodes in the CBM building to the worker nodes in the GSI Green IT Cube. The overall geometry and concept is visualized in Figure 5.6.

From a readout perspective, the detector data originates from the optical transceivers installed in the vicinity of the detector electronics. The optical links from the cave (E10) to the server room (E40.017) are subdivided into the following segments: on-detector cabling, flexible patch connection, optical backbone from the cave, and patch connections in the server room.

On-detector cabling The connections from the on-detector data sources to an MTP patch panel are performed with LC-to-MTP-24 fan-in cables. For the BMON, MVD and STS subsystems these MTP cave patch panels are located on the left and right side of the

[15]One of the smaller detector systems might use CRI1 boards (cf. Sec. 5.2.2), which can be easily accommodated with the described connection concept.

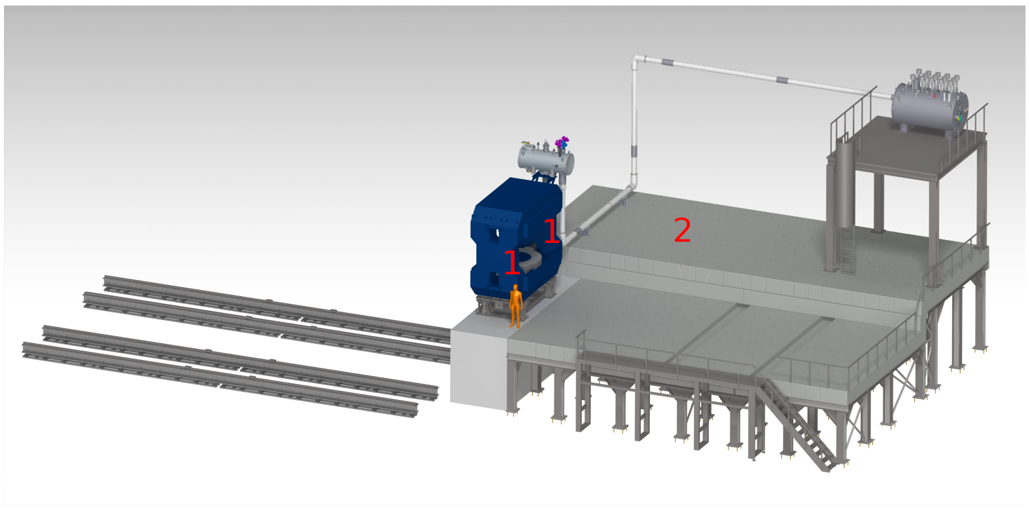


Figure 5.7: Sketch of the location of optical patch panels in the cave. Detector systems located inside the dipole will interface to patch panels located on both sides of the dipole magnet (1). All other subsystems will connect to the optical backbone in a rack located below the upstream platform (2).

dipole magnet (Pos. 1 in Fig. 5.7). For the RICH, MUCH, TRD, TOF and PSD subsystems, which can be individually displaced, the MTP subsystem patch panel is located at the base of the respective subsystem support structure.

Flexible patch connection The movable subsystems (RICH, MUCH, TRD, TOF and PSD) are interfaced with MTP-24 patch connections from their MTP patch panel on the support structure to a dedicated MTP cave patch rack, located below the upstream platform (Pos. 2 in Fig. 5.7). These connections will be flexed during repositioning operation the detectors, and are therefore replaceable.

Optical backbone from the cave A set of high-density (6 $\times$ 24 fiber) trunk cables is installed from the location of the fixed patch panels (Pos. 1 and 2 in Fig. 5.7) in the cave to the server room, providing MTP-24 interconnections.

Readout patch connections in the server room In the server room, the MTP backbone originating from E10 will end in MTP patch panels distributed across the racks. The MTP-24 fiber connections match the interface of the CRI2 cards. Short MTP-24 patch connections will be performed between the termination of the MTP backbone in the rack patch panels and the MTP couplers on the CRI2 boards.

Optical connections summary The optical connection from the detectors to the E10 patch panels, depicted as Positions 1 and 2 in Figure 5.7, is considered part of the respective subsystem project. The trunk cables from E10 to E40 as well as the cables from the CBM building to the Green IT Cube are considered common infrastructure. They ensure a uniform system that is centrally planned and installed. The fiber cabling in the CBM building, based on 144-fiber cables which are summarized in Section 5.2.4, requires 120 MPO/MTP trunk cables (100 m OM4, incl. 10% spares). Optical cables of this type have been successfully used in the mCBM setup.

Chapter 6

Timing and Fast Control System

6.1 Overall concept and requirements

The Timing and Fast Control (TFC) system serves as the central clock and time master for the CBM readout tree. It distributes a common clock and a common time to all CRIs (cf. Sec. 2.2.4). In addition, the TFC provides a *fast control* path that allows sending *congestion* status information on the TFC uplinks (CRIs to TFC Master) and the distribution of commands for *throttling* and other system-wide state control in the TFC downlink direction.

The overall design goal of the CBM timing system is to ensure that after a reset of an individual link or component, or even upon a full system restart, the time difference between any pair of time counters in the system (TFC, CRIs, and FEE) is the same as before. This guarantees stability of the time differences between hits over partial or full system initialization. This is sufficient in CBM because the accelerator delivers a continuous beam. In LHC experiments, the front-end electronics often requires a precise data sampling phase relative to the bunch crossing and the orbit structure defines an observable absolute time structure. In CBM, no precise clock alignment is required at the FEE level. Residual offsets of the time counters are absorbed in the time calibration, which converts the FEE-determined timestamps to physical times of the hit. Only the *reproducibility* of all relative timings is essential. The required precision of the relative alignment at FEE level (before calibration) is determined by the FLES data model (cf. Sec. 4.2). The CRI packs hits into microslices based on the hit timestamps, and later stages build TSCs with one, or optionally multiple, microslices overlap (cf. Sec. 4.2.2 and Fig. 4.4). The microslice overlap has to accommodate all timestamp uncertainties (cf. Sec. 4.2.4), and thereby sets the scale of required alignment accuracy of local FEE time counters.

The required timing *stability*[1] is determined by the CBM detector properties. The TOF system aims at a system resolution of 80 ps. TOF uses a dedicated on-detector clock distribution (cf. Sec. B.4.1 and Fig. B.8) that is driven via a single dedicated GBT link.

[1]the timing *stability* is determined by the reproducibility of relative times after link resets and the time drift caused by thermal and other environmental factors. Not to be confused with clock *jitter*.

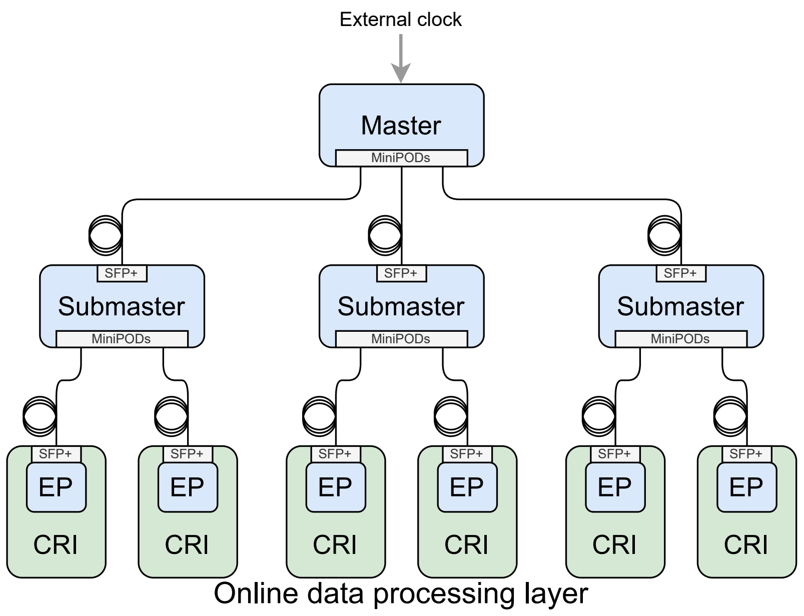


Figure 6.1: Topology of the TFC network

Therefore, the time-of-flight resolution depends only on the stability of the TOF internal clock distribution[2]. The RICH detector uses MAPMTs with a transit time jitter of 350 ps FWHM, and the timing information is used in background rejection cuts with typically $\pm 5\sigma$ width. All other CBM detectors have time resolutions of a few ns at best. The *stability* requirement is therefore set by the RICH detector and taken as 200 ps.

6.2 Implementation

The TFC system implementation is based on GBT links between FPGAs, with FPGA cores on both ends of the links. The cores provide communication with deterministic latency in the downlink direction, which is essential for time distribution, and a very low latency data path in the uplink direction for fast *congestion* status collection. The receiving CRI recovers the clock from the TFC downlink bitstream and, after appropriate jitter cleaning (cf. Sec. 5.2.1), uses it as the system clock as well as the reference clock for all GBT links it provides.

The expected number of about 200 CRI boards (cf. Sec. 5.2.4) cannot be controlled by a single master unit via direct point-to-point links. Therefore, the TFC system uses a scalable hierarchical multi-level topology as shown in Figure 6.1. This topology contains three node classes:

Master: connects to an external frequency and time reference, generates the 40 MHz master clock and has the master time counter, and distributes clock and time information

[2]This clock distribution covers all GET4-based systems and therefore includes T0.

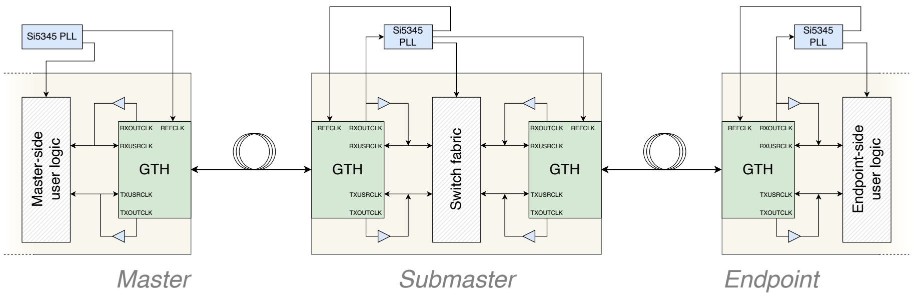


Figure 6.2: Clock cascading scheme

to the downlinks. It collects status information from all CRI boards, processes it, and issues data throttling and other fast control commands on the downlinks.

Submaster: guarantees the scalability of the system. It is responsible for forwarding the clock as well as timing information and fast control commands with deterministic latency in the downlink direction from the incoming Master link to the outgoing Endpoint links. In addition, it receives status information from many incoming uplinks from Endpoints, then aggregates and forwards this over a single uplink connection towards the single TFC Master.

Endpoint: is a CRI, receives the time information, executes fast control commands received from the Master, and sends *congestion* information towards the Master. For Endpoint integration and states see Section 5.3.3.

Because the hardware requirements for the TFC Submaster nodes are very similar to the CRI hardware requirements (cf. Sec. 5.2.1), it is planned to use the same boards, CRI1 or CRI2, for the CRI role and the TFC Master and Submaster roles.

6.3 Clock forwarding and time synchronization concept

System clock and time is represented in the TFC Master, Submaster and the Endpoint by a 40 MHz clock signal and a 64-bit time counter that increments with the 40 MHz clock.

The TFC Master node is connected to an external time and frequency normal, such as the 10 MHz and PPS outputs of a time provider, and uses a zero-delay PLL to generate the 40 MHz system clock as well as the reference clocks to operate the optical transceivers. At the start of the TFC Master, the time counter is initialized such that it holds TAI time measured in units of 25 ns relative to the Unix epoch time of 00:00:00 UTC on 1 January 1970. This time representation is monotonous, unambiguous and convenient. The time intervals of the microslices created in the CRI are expressed in TAI time, as are the TSC time intervals, so consistency checks can be easily performed in subsequent software layers (cf. Sec. 4.2.5).

Submaster and Endpoint nodes use the recovered clock from the incoming link as reference for a local PLL, which serves as jitter cleaner and generates the local system clock as well as the reference clocks to operate the optical transceivers (cf. Sec. 5.2.1 for CRI hardware details). In the CRI1 board a SiLab `Si5345` is used. See Figure 6.2 for the overall concept and Figure 5.3 for the CRI1 clock routing. The same process is repeated finally for each GBT link in the GBTx, where a VCXO-based PLL serves as jitter cleaner.

The synchronization of local time counters throughout the whole system relies on the forwarding of messages with deterministic latency (DLMs). On the TFC downlinks this deterministic latency is achieved with suitably configured GBT-FPGA cores (see Sec. 5.3.4 for further detail). The GBT links provide a deterministic latency path from the CRI to the e-links which control the FEE ASICs. This is used to align the local time counters in the FEE. The HCTSP protocol used in the SMX and SPADIC ASICs is one concrete example (cf. Sec. B.1.4). At this stage, it is also possible to introduce time offsets to compensate for optical link delay differences or analog and digital processing delay differences in the front-end electronics.

The TFC system currently under development does not compensate for timing drifts, e. g., from thermal drifts of electronics or optical fiber propagation delays. It is under investigation to add such compensation for the TFC links as well as for GBT links for critical detector systems. A natural option is to use TCLink [68] cores. This core was developed in the context of the lpGBT project [69] and integrates easily with the lpGBT-FPGA cores [70]. How it can be integrated with GBT-FPGA cores is under study.

6.4 Fast control

6.4.1 Background

The CBM self-triggered front-end electronics generate continuous streams of hit data. The readout chain is essentially a series of FIFO buffers, communication paths with fixed or limited bandwidth, and aggregation stages. The first layers of FIFO buffers are located in the FEE ASICs, others along the CRI data processing path.

When the DAQ system is operated near the throughput limit, the FIFO buffers will fill up. At some point, data will be lost because a FIFO is full. FIFO overflow situations are monitored and recorded, often in the form of high-priority messages, so that later analysis stages can discard time ranges with incomplete data to protect against incorrect reconstruction. But without further precautions, this data loss will be random across the system and lead to the collection of incomplete event information. Already a small fraction of random data losses can render the entire data useless for physics analysis. The goal of the fast control is to provide mechanisms to synchronize the losses in case of system overload such that time ranges are coherently discarded and the data send to FLES is mostly complete event information.

The sensitivity of CBM depends on how much of the DAQ bandwidth can be used practically without degrading the data quality. A key limiting factor is the beam fluctuation seen in synchrotrons with slow extraction [71]. Practical experience with SIS18 beams from the HADES experiment shows that it can be substantial and on a 100 µs time scale (see Figure 5 of [72]). The specification for SIS100 beams to CBM and HADES is to keep intensity fluctuations (peak to average) below a factor of two [14]. The fast control part of the TFC provides mechanisms to ensure stable operation at high beam intensities even when the beam fluctuation requirements are not met.

The effect of intensity fluctuations could be mitigated with larger FIFOs or increased uplink bandwidth. However, there are not only financial but also technical limits. Especially in the very dense STS setup, space and power constraints are major limitations and allow only moderate bandwidth reserves. Due to these conditions, the SMX ASIC used in the STS was equipped with detailed FIFO monitoring and with data throttling controls [73]. The SMX can send FIFO-full alert messages and supports OFF/ON style throttling[3] as well as a FIFO-clear mechanism. An initial, preliminary study has shown that these mechanisms can be used to improve high-load performance [74].

6.4.2 Implementation

The task of the fast control part of the TFC is to

- collect *congestion* information
- generate and distribute throttling commands
- disseminate system-wide synchronous state changes

The data transport between TFC Endpoints and the Master is done via GBT links, as outlined in Section 6.2. The uplink frames are fully available for fast control, the downlink frame is structured into fields for time distribution and fast control.

The protocol and interface details are still under development, so only brief considerations for a design option will be given here.

This design approach starts with a classification of *congestion* sources. For a given CRI design, one will have a small number of source types, like FIFO fill state in the entry and exit stage of the CRI data processing chain, or FIFO-full states received from the connected readout ASICs. For each source type, the number of objects is known. This leads to a very simple but generic concept

- for each source type, the CRI aggregates the status of the *congestion* objects into a number with a given width. For sources that provide only binary information, this can be easily implemented as the population count of a bit vector. If the number of objects is greater than the field width, the value saturates. Other algorithms are

[3]This mechanism of the SMX *discards* hits before the local channel FIFO. Not to be confused with the well-known XOFF/XON *flow control* that acts on data transport channels.

conceivable if more information is available for a source type, such as a FIFO fill level.

- the TFC uplink frame is divided into fields, each containing such a source type value.
- the Submaster aggregates information received from the Endpoints by adding the values for the same type. The simplest method is a saturating addition, but more sophisticated functions are possible.
- the Master performs the same aggregation for the information received from the Submasters. The throttling decision is based on the aggregated values, simplest being a value over a programmable threshold logic[4].
- the throttling state is broadcast from Master to all Endpoints. In case the final target is a FEE ASIC, e. g., the SMX, this is forwarded by the CRI.

[4]Such a threshold logic allows to tolerate a small amount of congestion and can be used to optimize overall performance.

Chapter 7

Evaluation by mCBM

A CBM full-system test-setup named mCBM@SIS18 (short “mCBM”) was set up 2017 and 2018 at the GSI experimental area Cave D (HTD)[1] [28]. It has been taking SIS18 beam since 2019 within the FAIR phase-0 program of GSI / FAIR. The primary aim of the mCBM experiment is to commission and optimize the CBM detector prototypes or pre-series productions, the readout system, and FLES components under realistic experiment conditions up to the top CBM interaction rates of 10 MHz. In particular, it enables the verification of the concepts presented in this TDR. The current setup covers all stages of the foreseen online systems:

- The readout tree between the CRIs and the detector electronics.
- The FLES entry stage with entry nodes and CRIs.
- A fast timeslice building network and a long-haul connection between the experiment and the Green IT Cube.[2]
- A small FLES compute cluster.
- A prototype connection to the current GSI compute farm.
- A prototype of the TFC system including the connectivity to the GSI’s WhiteRabbit time distribution network.
- The needed service and infrastructure nodes as well as the foreseen experiment control networks.

The CRI1 DAQ hardware and corresponding first FPGA design and software versions were successfully tested for the first time during the ^{16}O beam block in July 2021. O+Ni collisions at 2.0A GeV kinetic bombarding energy were measured at a maximum collision rate of about 1 MHz as a first commissioning run (#1588). The following chapter presents further details of the experiment and the results of performed tests and recent beam times.

[1]Located at the beam entrance of the experimental area Cave C (HTC), hosting the nuclear structure experiment R^{3}B.

[2]While this connection is only 300 m for mCBM it uses the same technology as foreseen for the final connection.

7.1 mCBM experimental setup

As depicted in Figure 7.1, the mCBM experiment is positioned downstream of a solid target under a polar angle of about 25° with respect to the primary beam towards a beam dump, which is located 7 m downstream at the south end of the experimental area. mCBM does not comprise of a magnetic field, and therefore measures charged particles produced in the nucleus-nucleus collisions traversing with straight trajectories through the detector stations. The mCBM setup includes detector stations of all CBM detector subsystems. A photograph of the present setup is shown in Figure 7.2.

As depicted in the GEANT geometry in Figure 7.1 as well as being shown in the photograph of Figure 7.2, the following CBM detector subsystems are installed in the mCBM experiment[3]:

- the fast and segmented diamond counter for time-zero (T0) determination positioned 20 cm upstream of the target is used during the high-rate tests and is read out by the GET4 front end with one CRI1,
- the Silicon Tracking System (STS) subsystem, which is equipped with 2 stations, 5 STS ladders in total, is read out by the SMX front end with one CRI1,
- the Muon Chamber (MUCH) subsystem, which consists of 2 GEM modules placed between the STS and TRD subsystems as well as an RPC module mounted downstream of the TOF detector, is read out by the SMX front end with two CRI1,
- the Transition Radiation Detector (TRD) subsystem, comprising of one detector module for the inner TRD region (TRD-2D), is read out by the FASP front end with one CRI1, and is complemented with two large detector modules for the outer TRD region (TRD-1D), which is read out by the SPADIC front end with one CRI1,
- the Time-of-Flight (TOF) subsystem with RPC modules grouped into two stacks, is read out by the GET4 front end with one CRI1,
- and the Ring Imaging Cherenkov (RICH) subsystem using two aerogel radiators, placed directly behind the TOF detector, and delivers a second measurement of the particle velocity in a selected acceptance window, is read out by the DiRICH front end with one CRI1.

7.2 Data acquisition hardware setup

The CBM DAQ and data transport system as it is used for the mCBM experiment is sketched in Fig. 7.3. As of 2022, the mCBM cave was interfaced to the DAQ container by means of 3 trunk cables, each containing 144 multi-mode optical fibers. The entry stage of the mCBM FLES consisted of 6 entry nodes hosting a total 14 CRI1 cards (see

[3]the PSD detector participated in 2020 and 2021. A single module was installed at an angle of 5° relative to the beam axis.

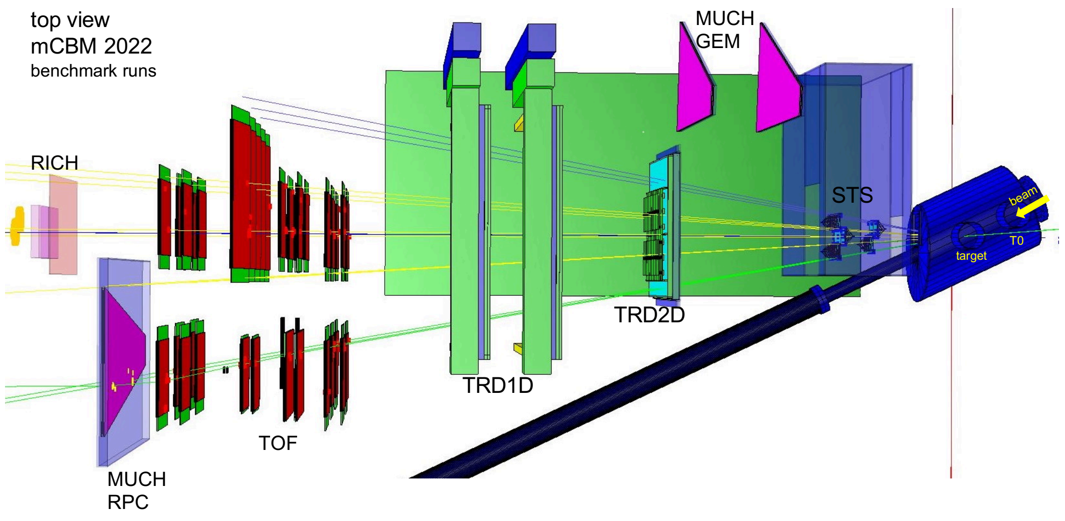


Figure 7.1: The mCBM GEANT geometry as of May/June 2022: the beam enters from the right propagating along the beampipe (blue). The detector systems are positioned under a polar angle of about 25° with respect to the primary beam.

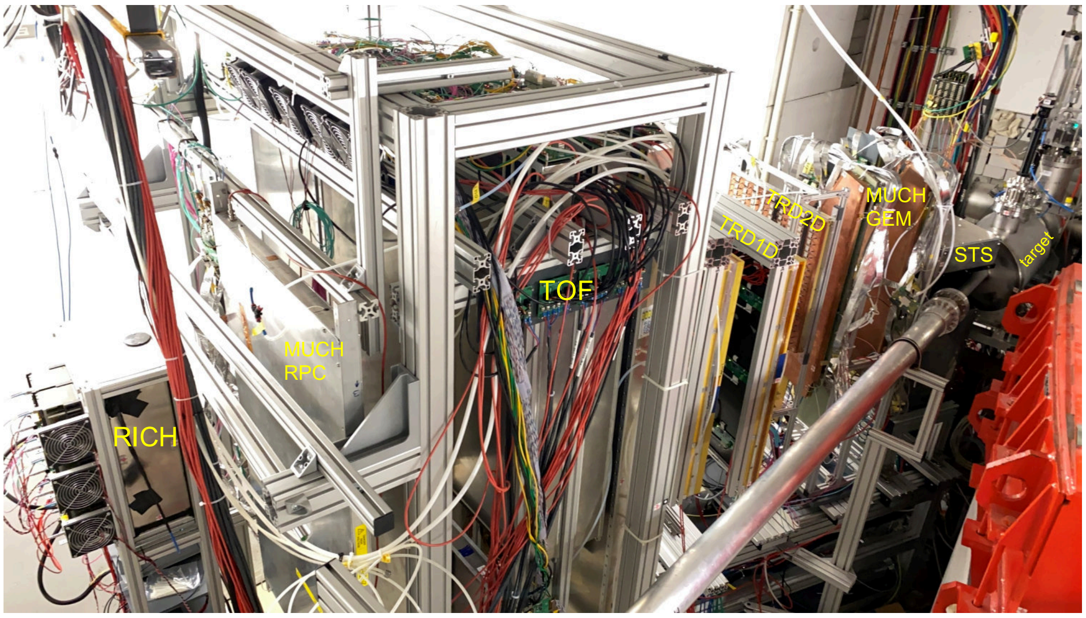


Figure 7.2: Photograph of the mCBM setup as of March 2022, the beam enters from the right.

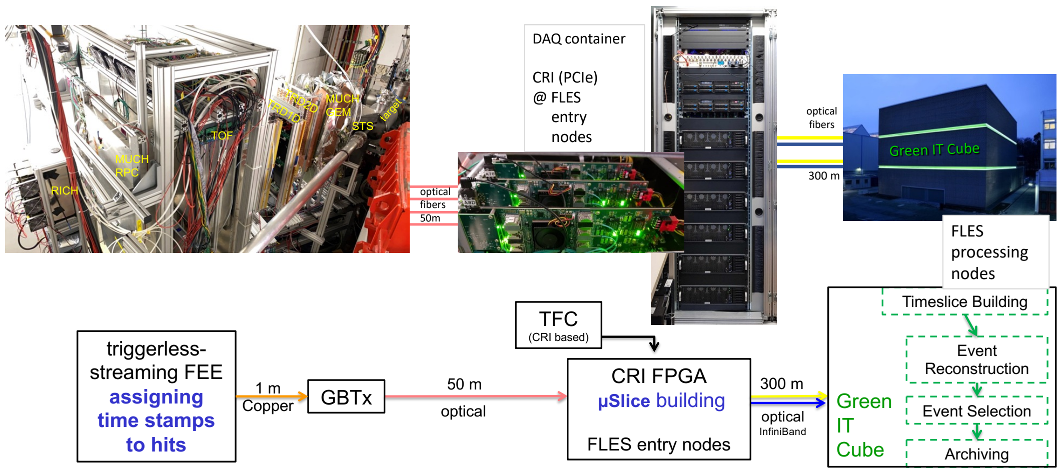


Figure 7.3: Sketch of the mCBM DAQ and data transport system.

titlepage of this document), synchronized with one TFC-Master node. Figure 7.4 shows the mCBM entry node rack as of September 2022, which serves as a CBM prototype installation. Describing the installed units from top to bottom, a WhiteRabbit Time Provider (2 HU) receives timing information from the GSI accelerator facility and forwards this information to the TFC-Master. The TFC-Master and a backup system are hosted in 2 individual (2 HU) nodes. Furthermore, an InfiniBand HDR switch (1 HU) is used for the interconnection of the entry nodes with the processing nodes in the Green IT Cube. 6 entry nodes (4 HU) are each populated with up to 3 CRI1 cards, which terminate the optical connections originating from the mCBM cave.

The long distance connection to the Green IT Cube is realized by a 300 m long trunk cable holding 144 single-mode optical fibers. A mix of 4 EDR and 2 HDR InfiniBand links offering 800 Gbit/s total bandwidth is used to forward the timeslice components from the entry nodes to the processing nodes in the Green IT Cube, where additional 4 processing nodes and the FLES master server are located. The processing nodes are equipped with a second InfiniBand interface allowing them to forward the data to the GSI Virgo cluster for further analysis and storage. The size of the readout system at mCBM is about 5% of the final setup at SIS100.

7.3 Data path software

The mCBM setup allows to test the full data path from the detectors, via the CRI up to the interface to the online processing software. The data flow in the FLES part is governed by a collection of software components named *Flesnet*. Microslices produced by the CRI

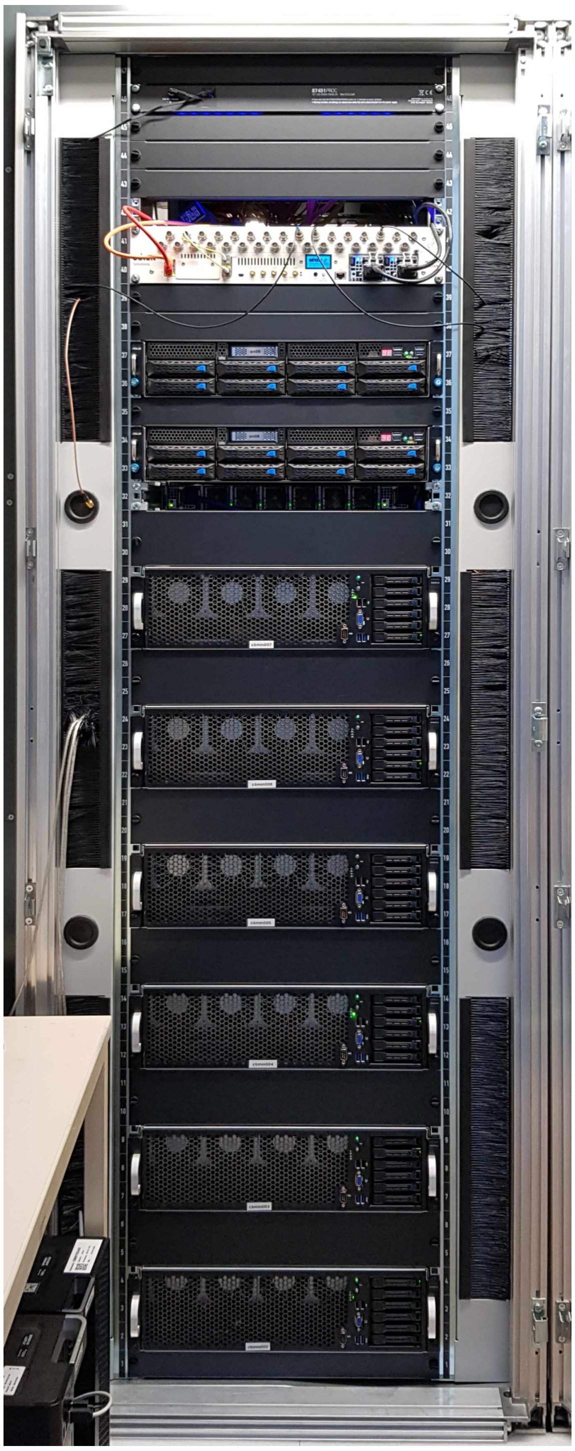


Figure 7.4: Photograph of the mCBM entry node rack as of September 2022, serving as a prototype installation for CBM.

hardware design are transferred via DMA to the main memory of the entry node and published in a shared memory segment. Subsequently, Flesnet performs timeslice building and delivers timeslices to the processing nodes. The Flesnet software then publishes the timeslices to one or more online processing tasks and, in the current version, simultaneously stores them to local disks in the form of timeslice archive (TSA) files.

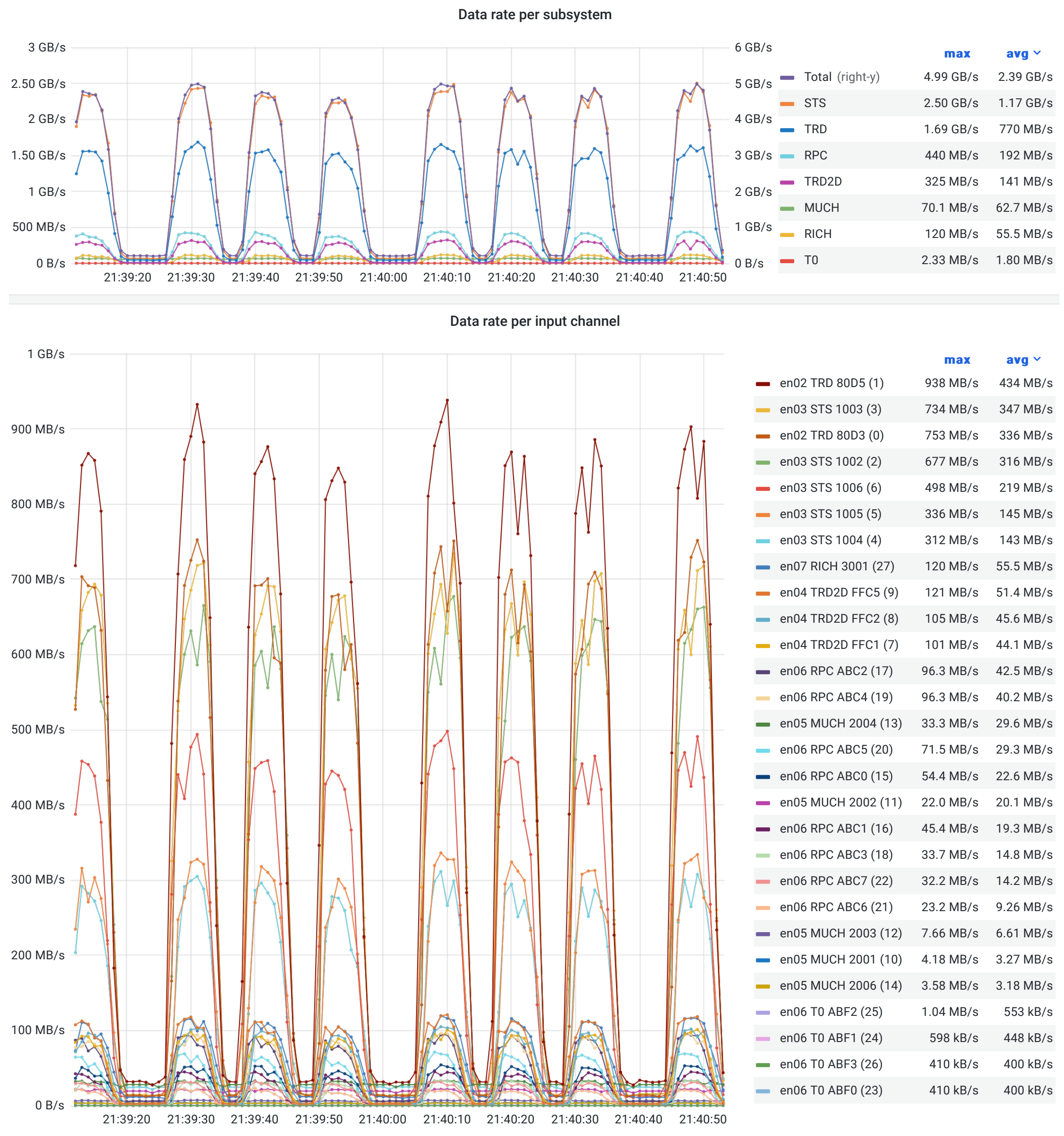
A control system for the FLES operation coherently manages the various Flesnet processes on the cluster and allows us to configure, start, and stop individual data-taking runs. While greater flexibility may be desirable for the operation at full CBM size, the present demonstrator software components already cover the whole data path and provide the full functionality for productive data-taking at a high level of efficiency. The Flesnet software components collect operational metrics at several key points in the data chain. These metrics are collected and analyzed on a central monitoring server, allowing to verify the operation of the system.

7.4 Data path performance

Figure 7.5 shows the measured input data rates per input channel and aggregated values per subsystem. Data shown here as an example of data flow and buffer utilization of a mCBM run originate from run number 2448 on June 16, 2022, taken in Au+Au collisions at $1.23A$ GeV kinetic projectile energy with an average collision rate of 300 - 400 kHz. Data were received by Flesnet through CRI cards in 6 entry nodes with a total of 28 FLIM channels active. Seven subsystems were participating in this run: STS, TRD +TRD-2D, TOF (named RPC in the figures), MUCH, RICH, and T0. As expected, the beam spill profile of the SIS18 facility is visible in the data rates of all subsystems except T0[4]. The STS system provides the largest contribution of approximately 50 % of the overall data volume. The total in-spill input data rate in this run reaches 5 GB/s. The average data rate per input channel varies significantly between different channels because the mCBM hardware setup was not optimized for balanced rates. At the data rates observed here, however, Flesnet can handle these variations successfully. In the full CBM experiment, special care will be taken to balance the average data rates across input channels.

The top plot in Figure 7.6 shows the input data rate on the timeslice build nodes. During timeslice building, the experiment raw data are aggregated from all inputs and distributed fairly to the three active build nodes over the InfiniBand network. In this setup, each build node stores the full timeslices on several internal disks. The maximum aggregate write throughput to these disks is below the maximum in-spill data rate of 2 GB/s, causing the timeslice buffer utilization (middle and bottom plots in Figure 7.6) to rise during spills. The plots illustrate how the spill breaks are then used to store the remaining data, smoothing the overall data rate successfully across spills. Even though the spill variation is expected to be less pronounced in SIS100, smoothing the data rate is an essential feature. Overall, data were recorded to local storage at an average sustained data rate of 2.4 GB/s.

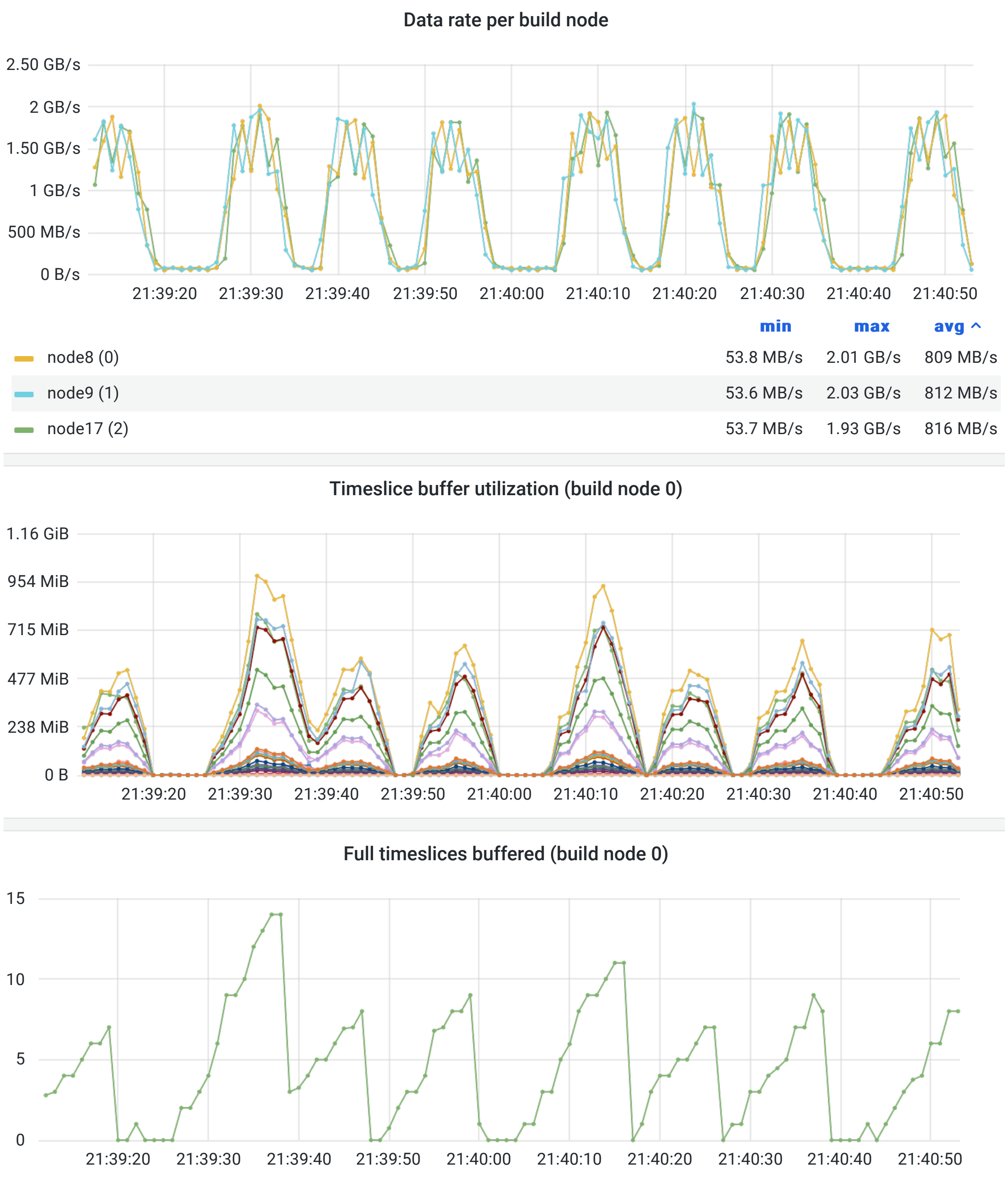
[4]Due to a misaligned beam, mainly missing the T0 diamond counter.



		max	avg
🟪	Total (right-y)	4.99 GB/s	2.39 GB/s
🟧	STS	2.50 GB/s	1.17 GB/s
🟦	TRD	1.69 GB/s	770 MB/s
🟦	RPC	440 MB/s	192 MB/s
🟪	TRD2D	325 MB/s	141 MB/s
🟩	MUCH	70.1 MB/s	62.7 MB/s
🟨	RICH	120 MB/s	55.5 MB/s
🟥	T0	2.33 MB/s	1.80 MB/s

		max	avg
🟥	en02 TRD 80D5 (1)	938 MB/s	434 MB/s
🟨	en03 STS 1003 (3)	734 MB/s	347 MB/s
🟧	en02 TRD 80D3 (0)	753 MB/s	336 MB/s
🟩	en03 STS 1002 (2)	677 MB/s	316 MB/s
🟥	en03 STS 1006 (6)	498 MB/s	219 MB/s
🟧	en03 STS 1005 (5)	336 MB/s	145 MB/s
🟦	en03 STS 1004 (4)	312 MB/s	143 MB/s
🟦	en07 RICH 3001 (27)	120 MB/s	55.5 MB/s
🟧	en04 TRD2D FFC5 (9)	121 MB/s	51.4 MB/s
🟦	en04 TRD2D FFC2 (8)	105 MB/s	45.6 MB/s
🟨	en04 TRD2D FFC1 (7)	101 MB/s	44.1 MB/s
🟪	en06 RPC ABC2 (17)	96.3 MB/s	42.5 MB/s
🟨	en06 RPC ABC4 (19)	96.3 MB/s	40.2 MB/s
🟩	en05 MUCH 2004 (13)	33.3 MB/s	29.6 MB/s
🟦	en06 RPC ABC5 (20)	71.5 MB/s	29.3 MB/s
🟦	en06 RPC ABC0 (15)	54.4 MB/s	22.6 MB/s
🟪	en05 MUCH 2002 (11)	22.0 MB/s	20.1 MB/s
🟪	en06 RPC ABC1 (16)	45.4 MB/s	19.3 MB/s
🟩	en06 RPC ABC3 (18)	33.7 MB/s	14.8 MB/s
🟥	en06 RPC ABC7 (22)	32.2 MB/s	14.2 MB/s
🟧	en06 RPC ABC6 (21)	23.2 MB/s	9.26 MB/s
🟪	en05 MUCH 2003 (12)	7.66 MB/s	6.61 MB/s
🟦	en05 MUCH 2001 (10)	4.18 MB/s	3.27 MB/s
🟨	en05 MUCH 2006 (14)	3.58 MB/s	3.18 MB/s
🟪	en06 T0 ABF2 (25)	1.04 MB/s	553 kB/s
🟪	en06 T0 ABF1 (24)	598 kB/s	448 kB/s
🟩	en06 T0 ABF3 (26)	410 kB/s	400 kB/s
🟦	en06 T0 ABF0 (23)	410 kB/s	400 kB/s

Figure 7.5: Incoming microslice data rates for mCBM run number 2448, taken in Au+Au collisions at 1.23A GeV kinetic projectile energy with an average collision rate of 300 - 400 kHz, June 2022.



	min	max	avg ^
🟨 node8 (0)	53.8 MB/s	2.01 GB/s	809 MB/s
🟦 node9 (1)	53.6 MB/s	2.03 GB/s	812 MB/s
🟩 node17 (2)	53.7 MB/s	1.93 GB/s	816 MB/s

Figure 7.6: Timeslice building data rates and buffer utilization for mCBM run number 2448, taken in Au+Au collisions at 1.23 A GeV kinetic projectile energy with an average collision rate of 300 - 400 kHz, June 2022.

7.5 Synchronization

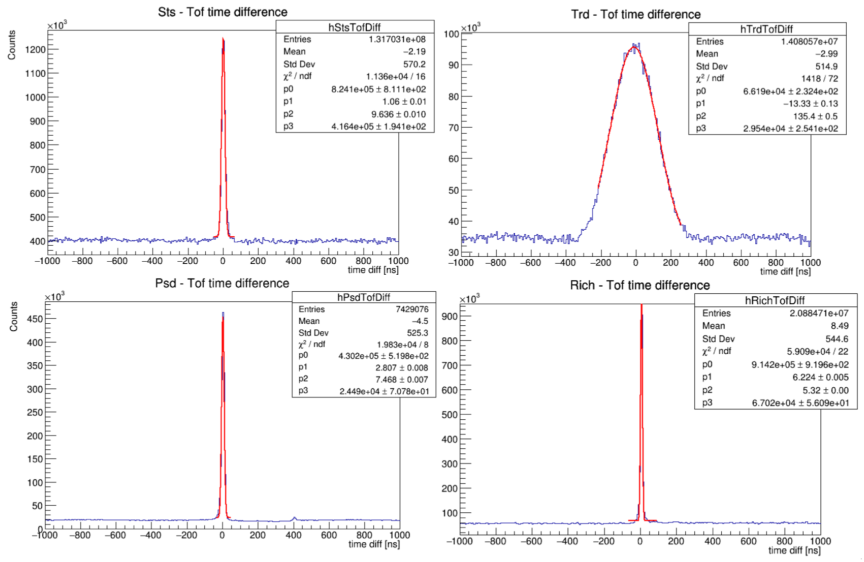


Figure 7.7: Stable timing: time difference measured by the detector subsystems STS, TRD, PSD, and RICH with respect to the TOF system (run 1588, O+Ni at 2.0*A* GeV, July 2021)

As a key requirement of a self-triggered streaming DAQ system, the CRI branches of all detector subsystems were stably synchronized over the full run period. Hence, a stable timing of the front-end electronics of all detector subsystems could be observed within the expected time resolution as it is visible in Figure 7.7. Here, all time differences are displayed within the given time window measured by the STS, TRD, PSD and RICH detector subsystems with respect to the time of the TOF system (time diff $= t_{\text{subsystem}} - t_{\text{TOF}}$). The measured time results from time stamps assigned to the raw hit messages, which were converted into digis with a common time representation during the unpacking stage of the data analysis. Individual time offsets of the subsystems were corrected during the unpacking stage; no detailed time calibration procedures were applied. As observed, the subsystem time offsets remain constant during the run as well as between runs.

7.6 Data analysis results

A preliminary CBM data analysis chain is being developed. As a first step, the raw data messages stored in timeslice archive (TSA) files are being unpacked by a unified unpacking scheme separating the framework from the algorithmic part. Accordingly, the subsystem-specific unpacking converts the raw data messages into digi classes within this common architecture, applying additionally individual subsystem time offset corrections as well

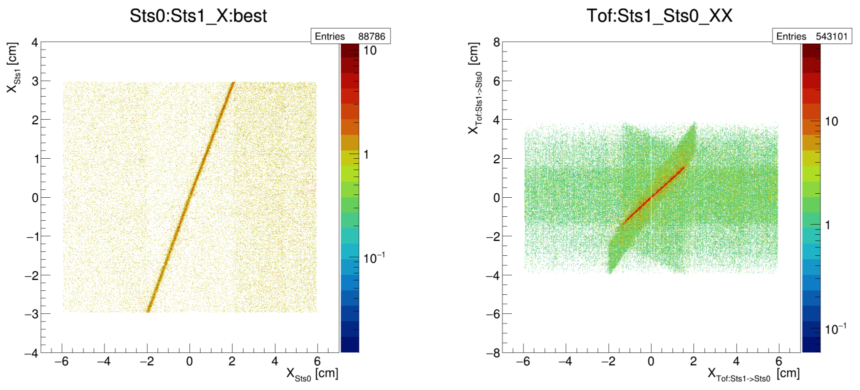


Figure 7.8: Correlation between spatial coordinates (x) of both STS stations STS 0 and STS 1 (left figure) and between both STS stations and TOF (right figure), run 1588, O+Ni at 2.0A GeV, July 2021

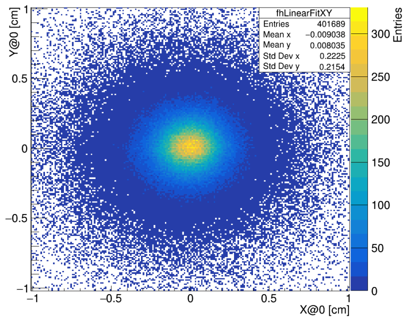


Figure 7.9: Vertex reconstruction based on tracks formed from hits (in a row of stacked modules) of the TOF wall and STS stations, run 1588, O+Ni at 2.0A GeV, July 2021

as involving the detector channel mapping and (partially) first calibration steps of the corresponding detector subsystem.

After data unpacking, a first, simplified event building is performed based on a time cluster search within a time window defined by the timing response of a reference detector system. The digi times of the reference detector are then taken as seeds for setting the time windows of the detector systems by adapting the window width according to the corresponding timing response. The identified event candidates are further filtered by digi-based trigger conditions cutting, e. g., on digi multiplicities of selected detector stations or their combinations. For mCBM data taken during the commissioning run O+Ni at 2.0A GeV in July 2021, the trigger condition of the event candidate selection requests $N_{\mathrm{TOF\ digi}} \geq 6$. Detector hits are then reconstructed by the subsystem-specific hit finding algorithms applied to the selected digis.

Supplementary to the correlation in time (Figure 7.7), correlations of spatial coordinates are presented in Figure 7.8, exemplary between both STS stations STS 0 and STS 1

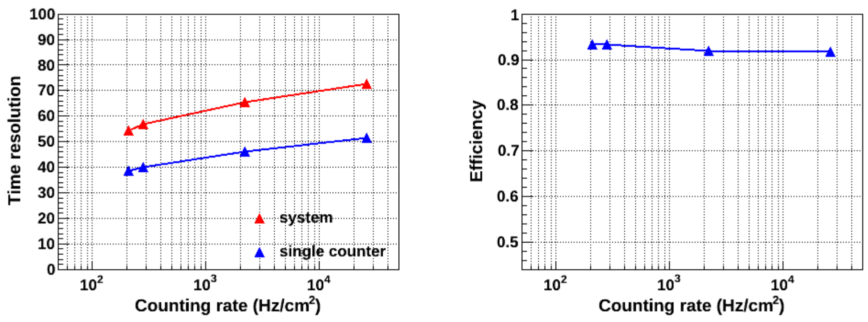


Figure 7.10: Sufficient stability of the readout system: the TOF RPC time resolution (left figure) for the system as well as RPC single counters as a function of the charged particle flux at high detection efficiencies (right figure), taken in O+Ni collisions at 2.0*A* GeV with 10^8 - 10^{10} beam particles per second, July 2021.

(left figure) and between STS stations and TOF modules (right figure). The displayed correlations are based on tracks formed from hits in both STS stations as well as including hits of the TOF wall. The resulting vertex reconstruction is shown in Figure 7.9. The dimension of the reconstructed vertex matches reasonably with the beam spot measured on the scintillation target of the beam diagnostics station, upstream in front of the mCBM target chamber.

During the mCBM campaigns, high-rate studies could be performed in nucleus-nucleus collisions with collision rates up to 10 MHz benefiting substantially from the highest available beam intensities at GSI SIS18. As an example, Figure 7.10 shows the observed time resolution of the TOF RPC modules as a function of the measured charged particle flux (left figures) at high detection efficiencies (right figure). The readout system provides sufficient stability to achieve 40 ps timing resolution. At the highest counting rates of about 25 kHz cm^{-2} the observed TOF time resolution reduces to the measured 70 ps for the system and 52 ps for RPC single counters at moderate efficiency degradation, well in line with the requirements on the CBM TOF system.

To further validate the readout and data processing concept of CBM, Λ production yields in nucleus-nucleus collisions are measured with mCBM as a benchmark observable, which will allow comparison with published data by the FOPI and HADES experiment. Hence, the first benchmark run has been measured on May 26, 2022, taking Ni+Ni collisions at 1.93*A* GeV kinetic projectile energy. With a total run duration of 5 h 55 min, approximately 5×10^9 collisions were collected on 30 TB timeslice archive (TSA) files storing the raw data messages. The second benchmark run was taken on June 16 – 18, 2022, measuring Au+Au collisions at 1.23*A* GeV with a total run duration of 34 h 33 min collecting approximately 2×10^{10} collisions on 180 TB timeslice archive (TSA) files. A detailed data analysis of the 2022 data has started. A first, preliminary data scan confirms a high data quality with stable synchronization. The benchmark runs show that the readout system performs reliably over long periods.

Appendix A

Data Rate Considerations per Subsystem

This appendix contains additional information and remarks concerning the event sizes and data rates for some of the individual detector subsystems. This information is intended as additional background to the summarized data in Chapter 3.

A.1 BMON

The BMON subsystem consists of two components, T0 and HALO. Both systems have only a negligible influence on the overall data rates.

A.2 MVD

From the simulation, we assume 500 hits per minimum-bias event from the actual collision event, plus 3000 hits caused by delta electrons from 100 beam particles. On top of this, there is a considerable contribution of dark rate and frame overhead data. The expected data rates add up to 3.5 GByte/s of dark rate and noise, 1 GByte/s due to delta electrons and about 0.5 GByte/s of actual hit data emanating from the particle interactions. The nature of the sensor that does not provide information about signal height does not allow for a significant reduction of data rate from noise and delta electrons in the preprocessing stage [75].

A.3 STS

Data rates for the STS were determined from detector simulations of Au+Au collisions at $12A\,\mathrm{GeV}/c$ using the geometry v19a which implements all major parts of the STS support structure. Event sizes for the STS vary slightly in the different setups because of shadowing from delta electrons by the MVD detector and backward scattered particles from the MUCH absorbers, when these systems are present in the setup. As these effects have only a small influence on the total data rate, they have been neglected here for simplicity and the rates for the MUON setup are used, which is the worst case for the STS overall data rates.

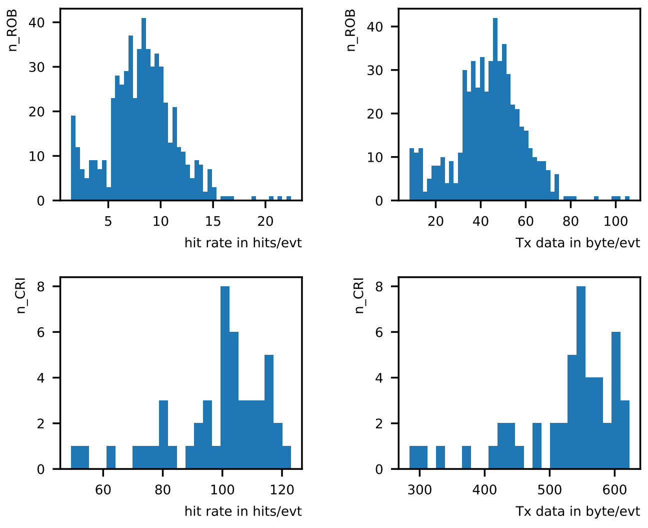


Figure A.1: STS hit and data rate distributions for ROB and CRI

The simulations provide (a) particle rates on the sensor from Au+Au interactions, (b) delta electron rates from beam-target interaction. Here a delta electron contribution from 100 beam particles for each interaction is assumed for a 1 % interaction target. (c) noise rates, which are given per time; numbers are calculated from the known noise performance of STS modules. At the moment the same conservative noise level is assumed for each module.

For the present simulations, a hit generation threshold of 4000 electrons was used, which corresponds to a noise threshold of 4σ ENC. The overall weight of the different contributions to the total rate is 89.5 % from interactions, 8 % from delta electrons (with large local variations) and 2.5 % from noise. Simulated particle and delta electron rates given in occurrences per sensor area can be translated into hit rates for each SMX ASIC in the STS system.

To derive data rates from the hit rates, additional contributions must be added for `TS_MSB` frames which are generated in addition to the hit frames and for the 8b/10b encoding of all uplink frames. `TS_MSB` frames are generated for each 800 ns time window with hits present in a given serial readout link. The `TS_MSB` rate cannot be calculated in a straightforward way, since it depends on the temporal distribution of the hits over the 800 ns time periods; for ASICs with multiple uplinks it also depends in the distribution of all hits over the available uplinks. Therefore all estimates of required ASIC readout bandwidth were done not based on the full uplink bandwidth of 10.67 MFrame/s, but based on the guaranteed hit bandwidth of 9.4 MHit/s (in case of the maximum `TS_MSB` rate of 1.25 M_TS_MSB/s). The maximum `TS_MSB` rate is also used for the estimates of total data rates. Hit frames and `TS_MSB` frames consist of 3 byte with 8b/10b encoding; thus 30 bit per frame are transmitted from the front-end ASICs to the CRI level. The distribution of hits and uplink

avg. int. rate	10^7	10^6	0	1/s
avg. message count per event	425	425	–	
avg. message rate	4.25×10^9	4.25×10^8	3.5×10^6	1/s
safety overhead	50	50	–	%
message rate incl. 50 % safety	6.4×10^9	6.4×10^8	3.5×10^6	1/s
message size	12	12	12	B
raw data rate	7.7×10^{10}	7.7×10^9	4.2×10^7	B/s
TrbNet readout trigger rate	100	20	10	kHz
absolute TrbNet overhead	7.5×10^9	1.5×10^9	7.5×10^8	B/s
relative TrbNet overhead	$\sim$10	$\sim$20	∞	%
total data rate to CRI	1×10^{11}	1×10^{10}	8×10^8	B/s
DiRICH backplane count	$\approx$200	$\approx$200	$\approx$200	
fiber count (DiRICH to CRI)	$\leq$376	200	200	
link speed (DiRICH to CRI)	$\leq$4.8	2.4	2.4	Gbit/s
bit rate to CRI (8/10b)	1×10^{12}	1×10^{11}	8×10^9	bit/s
CRI count	8	$\geq$6	$\geq$6	
fiber links per CRI	$\leq$47	34	34	
data rate per CRI	1.25×10^{10}	1.7×10^9	1.3×10^8	B/s

Table A.1: Expected hit and data rates for the CBM RICH detector assuming Au+Au collisions at $p_{\text{beam}} = 12A\,\text{GeV}/c$ and different interaction rate scenarios.

data for all ROBs and CRI (for the case of 12 ROB per CRI) are shown in Figure A.1.

For the data transfer out of the CRI, the number of hits is preserved (no preprocessing or data rejection). Each hit is expanded in its address and timestamp (to a 3.2 µs interval), resulting in 4 Byte per hit message. A `TS_MSB` expansion similar to the front-end protocol is done with one `TS_MSB` for each 3.2 µs bucket of a unit (Dproc) processing 14 readout links. This will lead to a worst case `TS_MSB` overhead in the CRI_OUT data of 2.16 GB/s.

A.4 RICH

The numbers given in Table A.1 are based on an estimate of an average of 425 hits per event on the RICH photon detector (as simulated for minimum bias Au+Au collisions at $p_{\text{beam}} = 12A\,\text{GeV}/c$). This average includes hits due to (simulated) background, noise, laser pulser, and dark rate. The latter can be regarded as negligible compared to the simulated average hits per event at the full interaction rate. We assume an additional 50 % hits per event to account for unexpected (i. e., not yet simulated) additional hits caused, for example, by fluorescence light in the detector, underestimated cross talk on the MAPMTs, radiation background, or after-pulsing.

Each hit message consists of a leading and trailing edge time, plus (in most cases) an epoch message, with a total message size of $3 \cdot 4\,\text{B} = 12\,\text{B}$. Epoch messages are generated per individual channel, they are skipped if a second hit in the same channel occurs within the same epoch (≈10 µs, >100 kHz per channel), which will be only rarely the case. Additional overhead is caused by the TrbNet protocol due to event headers and readout trigger timestamps from each of the DiRICH front-end modules. This overhead scales with the constant TrbNet readout trigger rate which can be reduced from ∼100 kHz at full rate operation down to ∼10 kHz at low rate operation to reduce the overall data rate. Currently, we estimate this overhead to be 94 words times the number of combiner modules per readout trigger.

The number of CRI cards is not only defined by the total data rate, but also by the number of optical links from the detector. A minimum of 200 optical links (single link per readout module, initially running at 2.4 Gbit/s) has to be received on the CRI side. Later, this number might be extended to 2 optical links (with up to 4.8 Gbit/s each) on a selected number of readout modules, employing an upgraded DiRICH combiner module (not yet developed). This upgrade is not needed for the initial operation of the RICH at a reduced rate (up to an interaction rate of 10^6/s). Proper load balancing across all CRI boards will be achieved by optimized allocation of the individual fibers, which cover a broad load spectrum depending on the individual readout position.

The contribution of MAPMT photon dark rate (due to thermal emission of photons) to the overall data rate can be assumed to be very low. We estimate an average dark rate of 50/s per channel, or 3.5×10^6/s for the full RICH detector. This number is considerably larger (factor 3 to 4) than recent observations at mCBM, and has to be seen in relation to the 4.25×10^9/s total hit rate at an interaction rate of 10^7/s. The MAPMT dark rate shows only a weak threshold dependence (within a reasonable range) and can thus be regarded as constant. Other sources of additional noise were taken into account already in the 50 % safety overhead.

A.5 TRD

There is an ongoing decision process to replace the previously planned full TRD design, for clarity also referred to as full TRD-1D, with a proposed alternative design of a “inner TRD-2D” plus surrounding “outer TRD-1D”. While this section refers to both variants, the values in Table 3.1 are calculated using the default full TRD-1D, not its proposed variant. The proposed TRD-2D variant for the inner section features a higher channel density (factor 1.4), but instead of the time-resolved sampling approach of the TRD-1D, it uses single measurements, which yields significantly smaller hit messages. We, therefore, expect the variant to not cause an increase in data rate relative to the numbers presented here.

The event sizes have been calculated based on UrQMD simulations of minimum bias Au+Au collisions at $p_{\text{beam}} = 12A\,\text{GeV}/c$ and GEANT3 transport through the CBM hadron, electron, or muon (“J/psi”) setup, respectively, according to TRD geometry version `v20b`. The comparison of the TRD rates in the muon setup and the electron setup reflects the effect of the MUCH absorbers on the produced particles. The so-called “forced-neighbor trigger” mechanics of the TRD SPADIC front end has been included explicitly in the message calculation, which explains part of an increase with respect to earlier estimates. The number of ADC samples is set to 7. The detector gain was set to 42 ADU (MIP hit central on detector pad, channel amplitude), the differential threshold to 8 ADU. Uncertainties from the current state of implementation of materials in the TRD structures as well as from triggering delta electrons have been determined by variation to be about 8 %, which is included in the given numbers already. An additional 50 % is included as a contingency factor, e. g., for simulation uncertainties (in particular: GEANT3 underestimates hadronic interactions), further material budget in CBM not yet considered, and detector gain variations as far as not caught by local threshold adjustment.

The design value for sample-to-sample fluctuations of the front-end electronics connected to the detector is better than 2 ADU (Gaussian width), such that noise triggers are neglected in the dark rate estimates. Nevertheless, given the spatially extended electrical structures of the TRD, a certain level of pick-up triggers from electric effects is expected. As the final electrical (grounding) scheme of the TRD cannot yet be tested and characterized to full extent, the corresponding pick-up rate expectations have a larger uncertainty level. We calculate here with a mean trigger rate of up to 800/s, leading to a dark data rate of 3.2 GB/s for the full TRD, or 2.2 GB/s for the outer TRD (TRD-1D), respectively.

Epoch messages of 3 B are generated per e-link with 62.5 kHz, leading to an additional overhead rate of 3.9 GB/s for the full TRD, or 2.7 GB/s for the outer TRD (TRD-1D) on e-link level from readout electronics to the FPGA level. The overall dark rates on the CRI input side, i. e., from pick-up triggers and epoch overhead, yield sums of 7.1 GB/s for the full TRD or 4.9 GB/s for the outer TRD (TRD-1D), respectively, which corresponds, e. g., to a fraction of dark rate from the overall TRD data rate of 4.4 % during the highest projected interaction rates of 5×10^6/s interactions at the target in case of the hadron setup. On the FPGA level the epoch messages can be suppressed to one per microslice stream. Currently, the number of microslice streams per CRI is not fixed. Assuming two CROBs per microslice stream and 12 CROBs per CRI leading to six microslice streams per CRI, the overall epoch message dark-rate (overhead rate) can be reduced to negligible values of 72 MB/s for the full TRD and to 48 MB/s for the outer TRD.

A.6 TOF

The TOF data rates are obtained based on Dubna Cascade Model [76] simulations of minimum bias Au+Au collisions at $p_{\text{beam}} = 12A\,\text{GeV}/c$ and GEANT4 transport through the CBM hadron setup. Beside the Monte Carlo particle rate, the simulation includes several additional contributions to the total data rate. These are a cluster size of 1.3, afterpulses contributing with a factor of 1.3, an error message rate of 1 %, a detector dark rate of $1\,\text{s}^{-1}\,\text{cm}^{-2}$, and the epoch rate of 39 062.5 Hz per microslice stream. This leads to a total message rate of 7.1×10^9/s. With a message size of 6 B, one obtains a total data rate of 42.62 GB/s at an interaction rate of 5 MHz.

Simulation shows an even higher message rate in the electron setup compared to the hadron setup by 8 %, caused by secondary particles produced in the RICH and TRD detector materials. However, the electron setup is assumed to be operated only at 100 kHz interaction rate leading to a total data rate of 0.92 GB/s. The comparison of the TOF rates in the muon setup reflects the effect of the MUCH absorbers on the produced particles. It is reduced by a factor of about 4.3, which leads to a total data rate of 9.80 GB/s.

A.7 PSD

The PSD system has only a negligible influence on the overall data rates.

Appendix B

Data Sources

This appendix contains detailed descriptions of the individual subsystem's readout architecture components and specific CRI FPGA design components.

B.1 SMX-based systems: STS and MUCH

B.1.1 SMX front-end ASIC

The SMX ASIC is a 128 channel integrated circuit designed for the readout of the CBM STS and MUCH detectors [77, 78, 79]. It comprises of an analog front-end with two-path processing by a timing comparator and a continuous-time 5-bit analog-to-digital converter, with a digital back-end with time pre-sorting, advanced monitoring and throttling features. The 320 Mbit/s serial links are scalable and targeted for GBT-based data acquisition.

The data processing in two parallel branches for time and ADC information generates self-triggered hit messages. Each channel provides an 8-hit deep FIFO storage. Hit losses due to a filled channel FIFO or an analogue pilepup situation trigger a channel "event missed" (EM) flag, which is added to the next hit generated in this channel. The ASIC is typically operated with an external 160 MHz clock, the time counter and the data readout utilize a double data rate clock of 320 MHz.

ASIC data readout is done using up to 5 differential uplinks which can be individually enabled and are electrically compatible with the GBTx e-links at the receiving end. Hits are read out from all non-empty channel FIFOs in a partially time-sorted sequence (using bits 13 to 6 of timestamp) and sent via the next available enabled uplink. Sorting based on all timestamp bits would be pointless here because the relation of physical hit time and recorded timestamp depends on amplitude and the chronological order of the physical hit times is not preserved.

Frame Type	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hit	0	7-bit channel address							5-bit ADC > 0x00					TS<9:8>		Timestamp<7:0>								EM
TS_MSB	1	1	Timestamp<13:8>						Timestamp<13:8>						Timestamp<13:8>						4-bit CRC			

Figure B.1: SMX hit frame and `TS_MSB` frame structure

In each uplink, hits are split into a `TS_MSB` frame with redundant 6 MSB of the time counter and a hit frame (see Fig. B.1). The `TS_MSB` frame is only sent if it differs from the `TS_MSB` value of the previously transmitted hit. The uplink frames are 8b/10b encoded for DC-balancing on the AC-coupled uplinks, resulting in 30-bit frames at 320 MBit/s [80].

B.1.2 STS

The readout chain of the STS consists of front-end boards (FEB8) with 8 SMX ASICs, connected by an electrical e-link interface to the GBTx based STS readout boards (STS-ROB) which act as data concentrators and also incorporate the Versatile Link based optical interface to the CRI system.

FEB8 front-end board

Each side of an STS double-sided sensor with 1024 strips is read out by one FEB8 front-end board with 8 SMX ASICs. Each FEB8 is operated at the bias potential of the connected sensor side and all communication signals from and to the digital back-end interface are AC coupled. The digital back-end interface consists on the downstream side (back-end to front-end ASICs) of a single common clock to all 8 ASICs and a common downlink for control commands in a multi-drop architecture. The communication protocol supports 4 address bits and allows to address each ASIC on a FEB8 individually or to broadcast commands simultaneously to all ASICs. The upstream (readout) interface consists of a variable number of uplinks per ASIC, each with a capacity of 10.66 MFrame/s and a minimum rate for hit frames of 9.4 MHit/s. The expected STS ASIC hit rates (cf. Sec. 3) range up to 32 MHit/s, which can be handled by the maximum of 5 readout links per ASIC. The available bandwidth allows introducing a safety margin in the designed readout bandwidth of more than 50 % for almost all ASICs.

Depending on the expected hit load for a given sensor, 1, 2 or 5 e-links per ASIC are used, resulting in three readout variants of the FEB8 board, called FEB8-1, FEB8-2 and FEB8-5 with 1, 2 and 5 uplinks for readout per ASIC. The three readout variants will be realized with two hardware variants of the FEB8; the FEB8-1 will use the FEB8-2 hardware implementing two e-links per ASIC on the FEB8 PCB and the same 40-pin flexible flat cable connector. The difference will be only in the back-end connectivity on the STS-ROB, supporting either 8 or 16 uplinks from the FEB8.

STS readout board STS-ROB

The STS readout board (STS-ROB) is a data concentrator board including the electrical-to-optical readout interface based on the radiation-tolerant GBTx ASIC and Versatile Link devices developed by CERN and others [31, 62]. The STS-ROB is a ROB3 type board, meaning it carries 3 GBTx ASICs, one (“master”) connected to the bidirectional VTRx optical transceiver for control and readout, and two (“slaves”) unidirectionally connected

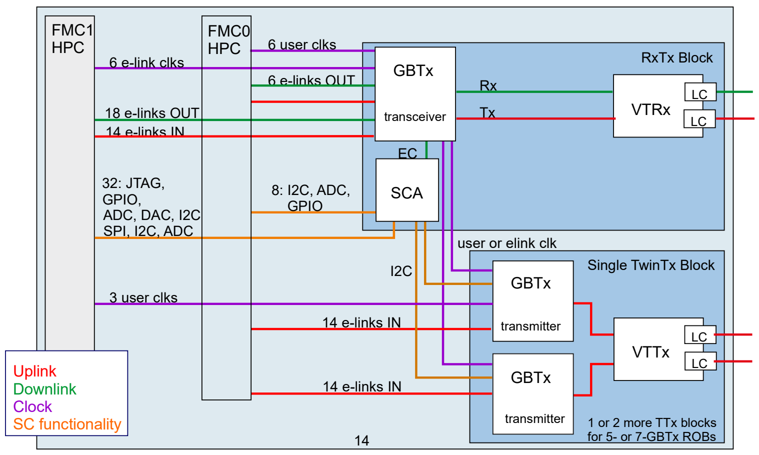


Figure B.2: STS Readout Board Block Diagram

to a VTTx twin transmitter device for readout and control responses (see Fig. B.2). It is functionally equivalent to the CBM CROB, which is in use at mCBM for the readout of mSTS, mMUCH and mTRD. The STS-ROB is however built in a special, elongated form factor, which allows for placement in the densely packed STS detector.

The slave GBTx are controlled via I^{2}C interfaces of the GBT-SCA ASIC. The GBTx devices are operated in widebus mode (without forward error correction in the optical uplink) at 320 Mbit/s e-link speed matching the SMX readout. With these settings, each GBTx can provide 14 uplinks. The resulting 42 uplinks closely match the 40 uplinks as multiples of 8 uplinks from up to 5 FEB8 boards.

On the downlink side, for connecting to a maximum of 5 FEB8 boards (FEB8-1), the STS-ROB uses 5 of the master GBTx phase adjustable clocks configured for 160 MHz, as well as 5 out of the 12 available 160 Mbit/s downlinks of the master GBTx. The phase adjustable clocks are derived from the clock-data recovery of the GBTx optical downlink, the individual clock phases and uplink input delays are determined in the front-end link synchronization procedure which is part of the HCTSP protocol.

Connectivity from FEB8 via STS-ROB to CRI

The FEB8 variants with a different number of uplinks require flexible connectivity between one or multiple FEB8 and an STS-ROB. Due to geometrical constraints in the mechanical setup of the STS, FEB8 to STS-ROB connections will be done separately for each side of the mechanical support frames (C-frame). A fixed sequence of up to 40 FEB8 with 8, 16 or 40 uplinks each is connected to a stack of STS-ROBs. A single STS-ROB will connect to either 5 FEB8-1, 1 FEB8-5, or 2 FEB8-2 and 1 FEB8-1 using different interface cards.

The STS-ROBs from all ROB stacks in each of the 4 symmetric quadrants of the STS detector extending over all 8 detector stations will be mapped consecutively to a sequence of CRI. The STS-ROBs are connected to the optical backbone with LC-MTP-24 fan-ins.

Station	FEB8-1	FEB8-2	FEB8-5	STS-ROB	CRI2
0	48	40	64	104	13.0
1	96	96	16	84	10.5
2	136	64	8	68	8.5
3	192	32	0	56	7.0
4	144	72	0	64	8.0
5	168	48	0	60	7.5
6	200	48	0	68	8.5
7	224	56	0	72	9.0
sum	1208	456	88	576	72

Table B.1: Quantities for the devices in the STS readout for each station. We assume that 8x STS-ROBs connect to 1x CRI2. STS-ROBs of neighboring stations can share the same CRI2, resulting in fractional values in the last column.

Each STS-ROB provides 1x Rx- and 3x Tx-links combined from its VTRx and VTTx device. The MTP-24 connector carries 12x Rx and 12x Tx fibers, allowing to interface 4x STS-ROB boards on 1x MTP-24 connector. The CRI2 will be designed with 3x MTP-24 connectors. In terms of optics 1x CRI2 can be interfaced to up to 12x STS-ROB (= 36 GBT links), 60x FEB8-1 and 480x SMX ASICs. In terms of hardware design extrapolated from the CRI1, we assume that 8x STS-ROB (= 24 GBT links) will be connected to 1x CRI2, see Table B.2. Since STS-ROBs can be connected from subsequent neighboring ROB stacks (also on different half units) to a single LC-MTP-24 fan-in, they can be efficiently connected to the CRI up to the maximum number of supported STS-ROBs in hardware design. An inefficiency in the ROB-CRI mapping may maximally occur in one single CRI per quadrant.

The readout tree of a single CRI consists of a mix of FEB8-1, FEB8-2 and for the first three stations also FEB8-5 boards. Device counts for each station and the full STS are shown in Table B.1. Based on these we derive the dimension of the optical interface in Table B.2.

The total amount of electronics in the STS system results in

- 14016 SMX ASICs
- 1752 FEB8
- 20480 e-link uplinks
- 576 STS-ROB
- 72 CRI2

B.1.3 MUCH

The front-end electronics of the MUCH subsystem consists of Front End Boards (FEBs) populated with 2 SMX ASICs. Although each ASIC provides 128 channels, in the MUCH

	article	quad	half	full
1	GBT links per CRI	24	24	24
2	MTP cable type	24	24	24
3	ROB-3	144	288	576
4	GBT links total	432	864	1728
5	CRI cards total	18	36	72
6	MTP patch cables	36	72	144
7	1x MTP-24 to 24x-LC fan-out	36	72	144
8	MTP 12-fold adapter PP	3	6	12
9	MTP patch panel [HU]	1	2	3

Table B.2: Optical interface for the STS. The optical connections inside the STS are arranged in quadrants (1st column). Connections to the two patch panels will be performed for the left or right half of the STS (2nd column). The overall quantities are shown in the 3rd column. Lines 1–3 show input parameters from the FPGA design (1), the optical connectivity (2), and the detector design (3). Based on these, we derive the quantities in lines 4–9.

use case only 64 channels are read out per ASIC, to ease the complexity of the PCB design. The use of 64 channels is perfectly matched to the fan-out capability of modern PCB technology. As channel fan-out is needed, there is no benefit from employing the high density channel count provided by the SMX. Thus, the MUCH electronics will comprise of 128 channels per FEB board. Each ASIC will be read out using 2 e-links, so in total, each FEB board will be read out using 4 e-links. The MUCH detector system comprises of 4 stations, 2 GEM stations upstream followed by 2 RPC stations downstream. Each station of MUCH consists of 3 layers. The first two layers will be realized in GEM technology whereas stations 3 and 4 will be realized using trigger-RPC technology at low gain.

MUCH-GEM

GEM detectors will be used in MUCH station-1 and station-2 due to high particle density. Station-1 has 16 GEM detector modules per layer, 3 layers, 48 detector module in total. Each module is equipped with 18 MUCH-FEBs. The total number of e-links for station-1 is $16 \times 3 \times 18 \times 4 = 3456$. Station-2 has 20 GEM detector modules per layer, 3 layers, 60 detectors in total. Each module is read out by 15 MUCH-FEBs. The total number of e-links for station-2 is $20 \times 3 \times 15 \times 4 = 3600$.

The MUCH-FEBs of station-1 and station-2 can be interfaced using either ROB3 or ROB1 configurations, in the following the focus will be on using the ROB3.

Station-1:

- Number of ROB3 per module: 2
- Total number of ROB3 for station-1: $2 \times 48 = 96$

- Total number of GBTx for station-1: $96 \times 3 = 288$

Station-2 (ROB3 usage without module crossing):

- Number of ROB3 per module: 2
- Total number of ROB3 for station-2: $2 \times 60 = 120$
- Total number of GBTx for station-2: $120 \times 3 = 360$

The ROB3 are connected to the optical backbone with LC-MTP-24 fan-ins. Each ROB3 provides 1× Rx and 3× Tx links. The MTP-24 connector carries 12× Rx and 12× Tx fibers, allowing to interface 4× ROB3 boards on 1× MTP-24 connector. The CRI2 will be designed with 3× MTP-24 connector. Therefore, each CRI2 can interface to up to:

- $3 \times 4 = 12$ ROB3 per 3 × MTP-24 connectors,
- $9 \times 12 = 108$ MUCH-FEB,
- 216 × SMX ASICs.

It is expected that the logic resources of the CRI2 will allow to connect up to 8 ROB3 (= 24 GBT links) per CRI2. So the total number of GBTx required for GEM detectors of MUCH sums up to $288 + 360 = 648$. Therefore, the total number of CRIs for GEM detectors of MUCH system sums up to $648/24 = 27$. See Table B.3 for a summary on MUCH readout.

MUCH-RPC

RPC detectors will be used in station-3 and station-4 of the MUCH detector system. Station-3 and station-4 each consist of 20 RPC detector modules per layer, 3 layers, 60 detectors in total. Each module consists of 4 MUCH-FEBs. While the 3 MUCH-FEBs covering the inner region will be read out with 4 e-links, the 1 MUCH-FEB in the outer region will be read out with only 2 e-links, which sums up to $3 \times 4 + 1 \times 2 = 14$ e-links. The total number of e-links for station-3 and station-4 is each $20 \times 3 \times 14 = 840$.

The RPC detector of MUCH station-3 and station-4 can be interfaced by ROB1, which can each read out up to 14 e-links and for this case perfectly match the uplink connectivity.

Station-3:

- Number of ROB1 per module (4 FEBs, 14 e-links): 1
- Total number of ROB1 for station-3: $20 \times 3 \times 1 = 60$
- Total number of GBTx for station-3: $60 \times 1 = 60$

Station-4:

- Number of ROB1 per module (4 FEBs, 14 e-links): 1
- Total number of ROB1 for station-3: $20 \times 3 \times 1 = 60$
- Total number of GBTx for station-3: $60 \times 1 = 60$

The ROB1 are connected to the optical backbone with LC-MTP-24 fan-ins. Each ROB1 provides 1× Rx and 1× Tx links. The MTP-24 connector carries 12× Rx and 12× Tx fibers, allowing to interface 12× ROB1 board on 1× MTP-24 connector. The CRI2 will be designed with 3× MTP-24 connectors. Therefore in terms of optical connections, each CRI can be used to interface to up to:

- $3 \times 12 = 36$ ROB1 per 3× MTP-24 connectors,
- $3 \times 36 = 108$ MUCH-FEB (using ROB1),
- 216 × SMX ASICs.

It is expected that the logic resources of the CRI2 will allow to connect up to 24 ROB1 (= 24 GBT links) per CRI2. So the total number of GBTx required for RPC detectors of MUCH system sums up to $60 + 60 = 120$. Therefore, the total number of CRI for RPC detectors of MUCH system sums up to $60/24 + 60/24 = 6$. can reduce the number of CRI from 6 to 5 nos. See Table B.3 for a summary on MUCH readout, according to which the full MUCH will have 768 GBT links connected to 33 CRI2 cards.

	article	numbers
1	GBT links per CRI	24
2	MTP cable type	24
3a	ROB3	216
3b	ROB1	120
4	GBT links total	768
5	CRI cards total	33
6	MTP patch cables	66
7	1x MTP-24 to 24x-LC fan-out	66
8	MTP 12-fold adapter PP	6
9	MTP patch panel [HU]	2

Table B.3: Optical interface for the MUCH. Lines 1–3 show input parameters from the FPGA design (1), the optical connectivity (2), and the detector design (3). Based on these, we derive the quantities in lines 4–9.

B.1.4 SMX data flow and processing

The SMX ASIC implements a dedicated HCTSP [80] protocol. The complete stream of uplink HCTSP frames is delivered to the HCTSP bridge, described in detail in the following. The extracted frames with hit data enter the data processing block. The HCTSP bridge separates the data into individual streams corresponding to single SMX e-links. The whole transmission chain between the e-link outputs of the SMX ASICs in the FEB8 and the hit frames output in the HCTSP is transparent, introducing only a certain latency.

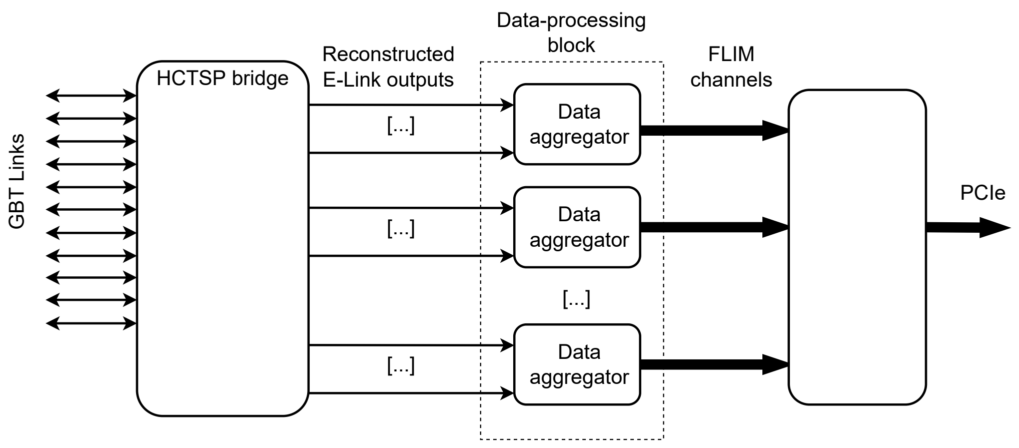


Figure B.3: General structure of the data flow for SMX

The number of e-link inputs depends on the number of ROB3 blocks handled by the HCTSP bridge and is equal to 42 inputs per ROB3. The total number of e-link inputs for typical numbers of GBT links is shown in Table B.4.

Nr of GBT links	Nr of ROB3 blocks	Nr of e-link inputs	
24	8	336	(baseline)
36	12	504	(performance)

Table B.4: The considered CRI2 configurations and resulting number of e-link inputs

The theoretical throughput of a single e-link is equal to: $320\,\text{MHz} \div 30 = 10.66\,\text{MHz}$ (encoded frame size is 30 bit). Due to the protocol overhead, it is safe to assume that the Hit frame rate will be no higher than 10 MHz. A single ROB3 generates no more than 42 e-links $\times\, 10\,\text{MHz} = 420\,\text{MHit/s}$. A single hit frame on the output of the data processing block is 32 bit, so a theoretical maximum throughput is 13.44 Gbit/s per ROB3. A data processing block is going to create microslices from one or two ROB3.

The HCTSP provides the frames with a 40 MHz clock and a clock enable flag. Such a low frequency clock would not allow utilizing the FPGA potential, so it was decided to use 160 MHz clock for the data processing. Both 40 MHz and 160 MHz are synchronous (have a common source in an FPGA MMCM). The general diagram of the data flow and processing scheme is shown in Figure B.3.

HCTSP protocol

The "Hit Control Transfer Synchronous Protocol" (HCTSP) is a protocol for data communication with a detector readout ASIC [80]. As such communication needs a robust and optimized solution, the HCTSP is a custom protocol adjusted to some of the ASICs used in the CBM experiment. The HCTSP is used in the SMX (STS and MUCH subsystems) and SPADIC (TRD-1D subsystem) chips.

The protocol maximizes the hit data throughput in the uplink (from ASIC to CRI) direction and data integrity in the downlink (from CRI to ASIC) direction. It is fully synchronous. Both uplink and downlink frames use 8b/10b encoding for DC-balance, and the frames in each direction have constant lengths and are transferred continuously.

The downlink frames support 4 request types: `no_op` (no operation), `WRaddr`, `WRdata`, `RDdata` (used for register access with 14-bit payload). A 4-bit chip address enables both individual chip addressing on FEBs and broadcasting messages. Since full-path delay can reach 1 µs, multiple commands in-flight are supported by using a sequence number. Each register access command should be acknowledged on the uplink. The error correction scheme is the modified selective repeat ARQ, where not-acknowledged requests are retransmitted, and register values can be verified by readback. The frames are 60-bit long (after 8b/10b encoding) and last 375 ns. They consist of `K28.5` comma character, 4-bit chip address, 4-bit sequence number, 2-bit request type, 14-bit payload (address/data) and 16-bit CRC (0x62CC).

The uplink communication contains mainly hit data but also control responses and special information (e. g., alerts). After all optimizations a throughput of 9.41 MHit/s/link is achieved, which results in a 71 % link occupancy for the design target of 250 kHit/s/channel in a 5 link/ASIC configuration. The frames have the constant length of 30 bit (after 8b/10b encoding), last 92.75 ns and support 5 frame types:

- dummy hit: equivalent of no op, used to fill the link when idle and to transfer the time counter MSB to keep the synchronization
- hit: containing: 7-bit channel address, 5-bit ADC value, 10-bit time counter LSBs as hit timestamp, 1-bit error status marker
- `TS_MSB`: serves as epoch message, transfers triplicated 6-bit time counter MSBs and 4-bit 0x9 CRC
- RDdata ack: acknowledge message for RDdata command, contains 15-bit register content, 3-bit sequence number LSB and 3-bit 0x9 CRC
- Ack: general acknowledge message with 2-bit ack-type: ack, nack, alert, 4-bit sequence number, 1-bit configuration parity, 4-bit status, 6-bit current timestamp (or 0x00 depending on the configuration register), 4-bit 0x9 CRC

The HCTSP protocol does not have its own standard or specification document. However, it is described exhaustively in a publication [80] and the SMX V2 Manual [81].

HCTSP bridge

The HCTSP bridge is a logic component responsible for the communication with both the SMX and the SPADIC ASIC. It is named bridge, as it connects two segments of the system using different communication protocols, Wishbone and HCTSP.

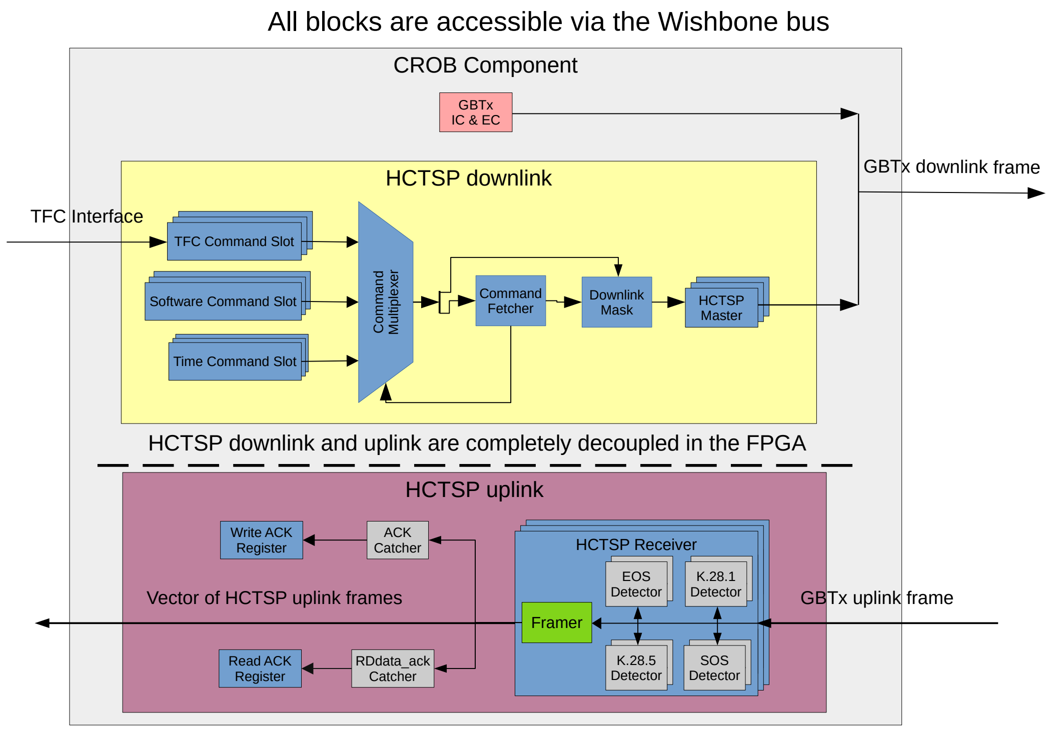


Figure B.4: Structure of the HCTSP bridge

Figure B.4 presents the structure of the HCTSP bridge. All blocks are accessible via the Wishbone bus. The downlink and uplink modules are completely independent. Such a design approach moves part of the complexity from the FPGA logic to the software, which is much easier and faster to update. The software side allows for:

1. clock phase characterization,
2. data phase characterization,
3. FEBs synchronization,
4. uplink masking,
5. command retransmission.

A command slot is a set of registers, containing information needed to send a particular command to the SMX. To save resources, the priority of command slots is configurable in a static way in hardware description sources. The command fetcher is responsible for choosing which command should be sent, based on the command slot's content and priority. The downlink mask allows sending a command to a particular FEB or even a particular SMX within FEB.

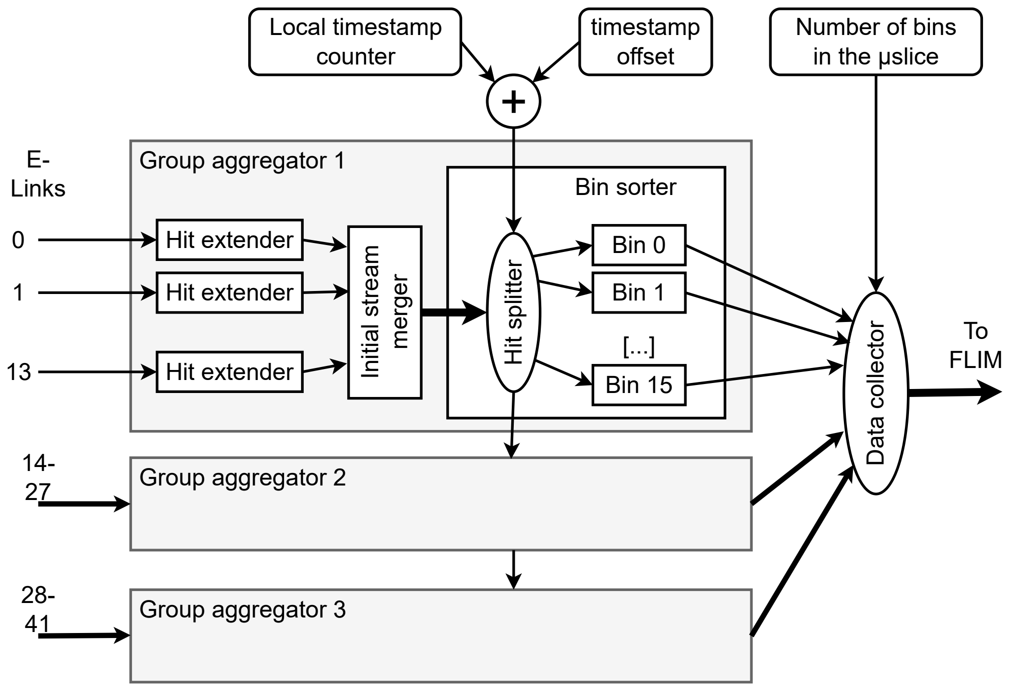


Figure B.5: Block diagram of the data aggregator for e-links from a single ROB3. The 42 e-links are processed in three groups of 14. The sorter uses 16 bins.

Data aggregator with bin sorter

The processing of data is based on their aggregation and delivery to the FLIM module. Therefore, the data processing block consists of multiple data aggregators working in parallel. In the current FPGA design, the data aggregator handles the data from a single ROB3, and delivers them to a single FLIM channel as shown in Figure B.5.

The main task of the aggregator is to combine the hits delivered by the connected e-links, sort them according to their timestamps so that they can be assigned to the appropriate microslices, and finally create microslices transmitted to FLIM. The following paragraphs describe the stages of this process. The data aggregator used in the current CRI1 prototype FPGA design uses a bucket sorter which provides limited timestamp-based sorting of the hit data.

SMX frame reconstruction The hit data sent by the SMX contain only 10 bits of the timestamp. For further sorting and assignment to the microslice it is necessary to reconstruct more bits of the timestamp. This task is done by the *hit_extender* block using timestamp bits from a hit frame and the preceding `TS_MSB` epoch message from the same e-link. There is one such block for each e-link.

The *hit_extender* waits for a `TS_MSB` epoch message and validates it (see next Subsection).

A timestamp field from valid `TS_MSB` epoch messages is stored in a register. The timestamp of following hit frames is reconstructed by combining the stored `TS_MSB` value and timestamp field of a hit frame.

`TS_MSB` epoch message validation The `TS_MSB` epoch message does not provide a method to verify its validity with 100 % certainty, moreover there is no alternative method of acquiring the timestamp from a dropped `TS_MSB` epoch message. All hits following a dropped `TS_MSB` epoch message would have a invalid timestamp. Taking the above into consideration, it was decided to attempt to recover a corrupted information, when possible, even if this may lead to accepting corrupted `TS_MSB` epoch messages.

In the verification of the `TS_MSB` epoch message the CRC field is not taken into account, because a single bit swap in a CRC field would invalidate the whole frame. Instead, a frame is considered valid if it has at least two identical `TS` fields. This way a single bit error cannot invalidate the frame.

Initial stream merging The SMX frames arrive on the input of the data processing module in a pipeline running with the 40 MHz clock. However, because SMX transmits the frames encoded in 30 bits at a frequency of 320 MHz (see Sec. B.1.2), the average time between their arrival cannot be shorter than 93.75 ns.

The initial stream merging may be performed by browsing input pipelines using the round-robin approach, receiving each found hit data and delivering it to the sorter. The initial stream merger works with a 160 MHz clock, so it is capable of merging 93.75 ns×160 MHz = 15 e-links without a risk of dropping data.

The 42 e-links coming from a single ROB3 are divided into 3 groups of 14 e-links. Each group is merged with a single merger.

Bin sorter The bin sorter is used to split hits (that may come in non-chronological order) into groups registered in certain time periods (bins), which is required to generate the microslices.

The bin sorter uses an FPGA Block RAM to buffer the data. The memory is split into a certain number of identical address spaces. A hit is written to an adequate bin based on its timestamp. The number of hits written to the particular bin is stored in a register. Based on the *local time counter*, the sorter decides when arrival of the next hit belonging to the particular bin is unlikely and closes the bin for writing. The settable *timestamp offset* enables compensation of the data transmission delay to ensure closing bins at the right time. It is possible to request outputting the content of a certain closed bin. The output data word may be composed of multiple parallel frames, so the readout may be significantly faster than filling of the buffer.

The bin sorter works with a 160 MHz clock.

An example implementation of the bin sorter is composed of 16 bins. The bin number for the particular hit is defined by chosen four bits of its timestamp. If we select bits 13 to 10, then a single bin receives data from a 3.2 µs interval (for bits 12 to 9 - 1.6 µs). The capacity of the bin is equal to 1024 hits. This configuration is used because 16 bins of 1 µs interval are sufficient to compensate the worst-case latency when the SMX is connected with 5 e-links. Sorting resolution of 3.2 µs or 1.6 µs is acceptable for the experiment. Assuming equal distribution of hits, we may expect the maximum number of hits from 14 SMXs within 3.2 µs equal to $14 \times 1\,\mu s/93.75\,ns \approx 478$. Of course the fluctuations of the hit intensity may result in uneven distribution of hits between bins, leading to overflow. The overflown bin is marked with a dedicated flag.

In case of bins with length of 1.6 µs the expected maximum number of hits is $14 \times 1\,\mu s/93.75\,ns \approx 239$, and the risk of overflow is significantly reduced. The price of shorter bin duration is the reduced period of time in which the hits are sorted.

A significant advantage of a 1024-hit bin capacity is an ability to read out corresponding bins from 3 sorters in a time shorter than bin duration (if 4 frames are read in parallel), so there is no bottleneck.

Data collector The data collector collects data from multiple bin sorters and creates a single data stream. The collector is informed when a certain bin in the sorters is closed for writing. After that, the collector requests the data from the closed bin from all sorters sequentially. Multiple frames may be read out in one clock cycle. Bins with a capacity of 1024 hits are read with 4 frames in parallel so it is possible to read out 3 sorters in less than the duration of the bin (3.2 µs or 1.6 µs). It is also possible to merge data from 2 ROB3 boards by instantiating 6 sorters in parallel and reading 8 frames in one clock cycle.

In addition, the *data collector* generates the microslices.

B.2 SPADIC-based system: TRD-1D

B.2.1 The TRD-1D chain

The TRD-1D utilizes the SPADIC as the front-end ASIC, which combines analogue pre-amplifier, signal shaper, 9-bit ADC and digital message building in one chip [82, 40]. One ASIC connects to 32 cathode-pads of the detector. Incoming charge is translated via analogue shaping into signals of characteristic shape, proportional to the amount of charge being injected. The 9-bit 16 MHz ADC of each channel is running continuously and transmits to the subsequent hit logic, which releases the digital message building in case of the configured self-trigger condition being fulfilled. Adjacent channels are automatically co-triggered in order to enable comparatively large threshold values with still complete reading of the overall charge deposition on the segmented cathode pad plane. The number of transmitted ADC samples can be configured. A typical choice is 7 ADC samples.

The exact time information and the charge value of a hit are calculated/reconstructed from the transmitted ADC samples of the channels of the corresponding detector hit. This processing can be realized in the CRI stage (*online feature extraction*) or in software, a case separation depending on the reconstruction complexity, e. g., due to signal pile-up, is possible.

Groups of 16 channels are assigned to 1 uplink (320 Mbit/s e-link), thus one 32-channel ASIC features 2 uplinks, while being configured and controlled via 1 downlink. In case of hit rate excess, counting of lost triggers is transmitted for diagnosis, while still the link bandwidth is used for transmission of as much hits as possible. Transmission of epoch messages is ensured.

The Front-End Boards of the TRD-1D come in two form factors, adapting to the channel geometry of different TRD module types. One FEB type is equipped with 3 SPADICs (FEB3) and the other one with 5 SPADICs (FEB5).

In the next readout stage the FEBs are connected to ROB3 (cf. Sec. B.1.2).

The mapping of SPADIC FEBs to ROB3 is as follows:

1. 4 FEB5 ($4 \times 10 =$) 40 e-links interfacing 1x ROB3
2. 6 FEB3 ($6 \times 6 =$) 36 e-links interfacing 1x ROB3

For the SPADIC readout the CRI2 is foreseen to connect to 8 ROB3. The 8 ROB3 combine either 32 FEB5, 160 SPADICs, 320 e-links or 48 FEB3, 144 SPADICs, 288 e-links.

For a TRD fully read out with SPADIC, there will be in total 544 ROB3, connected to 1024 FEB5 and 1728 FEB3.

If the inner modules utilize the TRD-2D, these will be read out differently with the FASP ASIC, as described in Section B.3.1. In this case (described in Table B.5) the TRD-1D will reduce to in total 384 FEB5 and 1728 FEB3, connected to 384 ROB3 interfaced by 48 CRI2 cards.

	article	inner	outer	full
1	GBT links per CRI	24	24	24
2	MTP cable type	24	24	24
3	ROB3	160	384	544
4	GBT links total	480	1152	1632
5	CRI cards total	20	48	68
6	MTP patch cables	40	96	136
7	1x MTP-24 to 24x-LC fan-out	40	96	136
8	MTP 12-fold adapter PP	4	8	12
9	MTP patch panel [HU]	1	2	3

Table B.5: Optical interface for the TRD-1D. Lines 1-3 show input parameters from the FPGA design (1), the optical connectivity (2), and the detector design (3). Based on these, we derive the quantities in lines 4-9.

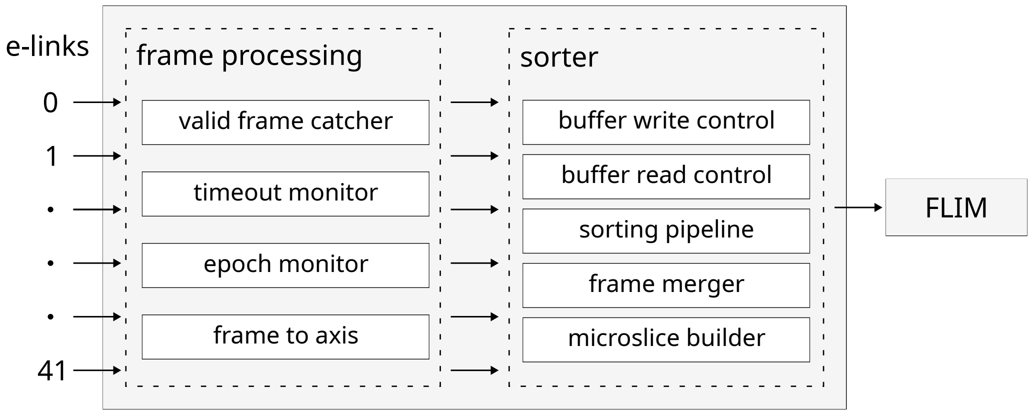


Figure B.6: Schematic overview of the SPADIC CRI FPGA design

B.2.2 The SPADIC data processing

The initial data handling after the GBT-FPGA is handled by the HCTSP bridge (cf. Sec. B.1.4). Following the HCTSP bridge, there is one data processing module per ROB3, which processes up to 42 e-links (cf. Fig. B.6). The *data processing* module is split into two main parts: first the *frame processing* module, where the frames are prepared for merging in the *sorter* module.

Frame processing

The task of the frame processing is to ensure the structural validity of the incoming data. This is achieved in several steps with the following submodules: The *valid frame catcher*, the *timeout monitor*, the *epoch monitor*, and the *frame to axis*. These submodules are replicated for each e-link.

The *valid frame catcher* ensures that the frames adhere to the message protocol and discards the frames if they do not. This module is a safeguard to catch any random fluctuation in the input stream, which was not detected by the 8b10b decoder in the HCTSP bridge.

The *timeout monitor* counts the time in between incoming `TS_MSB` frames. If the time exceeds at least six epoch periods ($6 \times 16\,\mu s$ **reference**) the e-link is considered timed out and a special timeout `TS_MSB` is generated and every 16 µs after.

The *epoch monitor* module keeps track of the incoming `TS_MSB` frames. It aims to detect and replace missing `TS_MSB` frames.

The *frame to axis* is an interface which converts the frames to the AXI4-Stream protocol with an `id` and `last` side channels.

Sorter

The *sorter* module is responsible for merging the input stream into a single time sorted microslice stream. The basic working principle is to buffer the data from one epoch for each e-link. The buffered epoch is then sorted. Currently, 21 e-links are sorted into one output stream. This results in two 32-bit streams, which are then merged into a single 64-bit stream. It consists of the following submodules:

The *buffer write control* keeps track of the fill status of the buffers, how many epochs are currently stored in each buffer, when the buffers are ready to be read from and ensures correct overflow handling. There is one FIFO per e-link.

Should a buffer run full a counter keeps track of the `TS_MSB` frames that could not be written into the buffers. Once the buffers free up the lost `TS_MSB`, frames are generated and marked with a flag showing that the buffer had been full.

The *buffer read control* module controls the data flow from the buffers into the sorting pipeline.

The *sorting pipeline* module generates one sorted output stream from n parallel input streams. Currently, there are two *sorting pipelines* instantiated, each processing 21 input links.

The *frame merger* module combines the output of the two sorting pipelines into one 64-bit word. Additionally, it ensures proper microslice packaging, i. e., it strips the incoming `TS_MSB` frames of their `TLAST` flag, counts the processed epochs and and asserts `TLAST` on the last word of the microslice (after the full number of epochs of a microslice have been processed).

The *microslice timestamper* is a simple module which assigns the current microslice time in ns to the microslice stream.

The *microslice control* module handles the flow control of the microslices. It suppresses or enables the microslice stream to the FLIM according to the microslice time window, which is used for synchronous starting of all subsystems. Additionally, it provides a rudimentary back pressure handling, by ensuring that no `TLAST` frame will be discarded due to back pressure. Whenever a `TLAST` frame cannot be sent, it is registered in a counter. When the back pressure is relieved microslice control generates as many empty microslices as missed microslices have been registered in the counter.

B.3 FASP-based system: TRD-2D

B.3.1 The TRD-2D readout chain

The TRD-2D detector uses the FASP ASIC as analog front-end. Here, each FASP connects 16 pads from the detector and provides 16 analog outputs which are digitized by discrete ADCs. The analog outputs are flat-top signals. A digital time signal is also provided by

FASP, indicating the flat part of the flat-top. A version of FASP including ADCs for the output channels is currently under development.

The output from the ADCs and the time signals are sent to the digital part of the front-end (GETS), implemented using a PolarFire FPGA (which was chosen because of it's radiation tolerant properties), where the digital stream is produced and sent to GBTx using 320 Mbit/s e-links. For each FASP, one e-link is used. The clock and control signals are received by GETS via GBTx. The 40 MHz clock received from GBTx feeds a PLL on the FPGA where the clocks required by FASP, ADCs and internal processing are generated.

FASPs and ADCs are operated at either 40 or 80 MHz (selectable), however, the timestamps always provide 12.5 ns binning. On the GETS output a stream of *nanoslices* is produced and sent to GBTx. Each nanoslice includes words corresponding to 128 periods of 12.5 ns, corresponding to 64 periods of the CBM clock. It terminates with an epoch message which includes an epoch number. Because the digital part of the front end is implemented using an FPGA, it is possible to optimize the design later.

Currently, the front end is implemented using 2 boards interconnected with a large stacking connector. The first board (FASPRO-DR) includes 6 FASPs and connects to the second board (GETS) which includes 2 FPGAs, one for every 3 FASPs. The GETS board connects to a C-ROB3 board using 4 SATA cables, 3 for the 6 e-links corresponding to the 6 FASPs and one for clock and control signals. Between the 2 FPGAs on the GETS board there are interconnects used to send clock and control. The GETS board also includes 2 DACs for generating the threshold voltage for FASPs on the FASPRO-DR board.

	article	numbers
1	GBT links per CRI	24
2	MTP cable type	24
3	ROB3	208
4	GBT links total	624
5	CRI cards total	26
6	MTP patch cables	52
7	1x MTP-24 to 24x-LC fan-out	52
8	MTP 12-fold adapter PP	5
9	MTP patch panel [HU]	2

Table B.6: Optical interface for the TRD-2D. Lines 1-3 show input parameters from the FPGA design (1), the optical connectivity (2), and the detector design (3). Based on these, we derive the quantities in lines 4-9. This is based on the conservative assumption of 24 GBT links per CRI2, which is matter of further optimisation (cf. Sec. 5.2.4).

Each C-ROB3 accommodates 6 FASPRO-DR/GETS stacks. Overall, for each of the 40 TRD-2D chambers, 30 FASPRO-DR/GETS stacks and 5 C-ROB3 boards are used. It is in principle possible to accommodate 7 stacks for one C-ROB3, however, this would

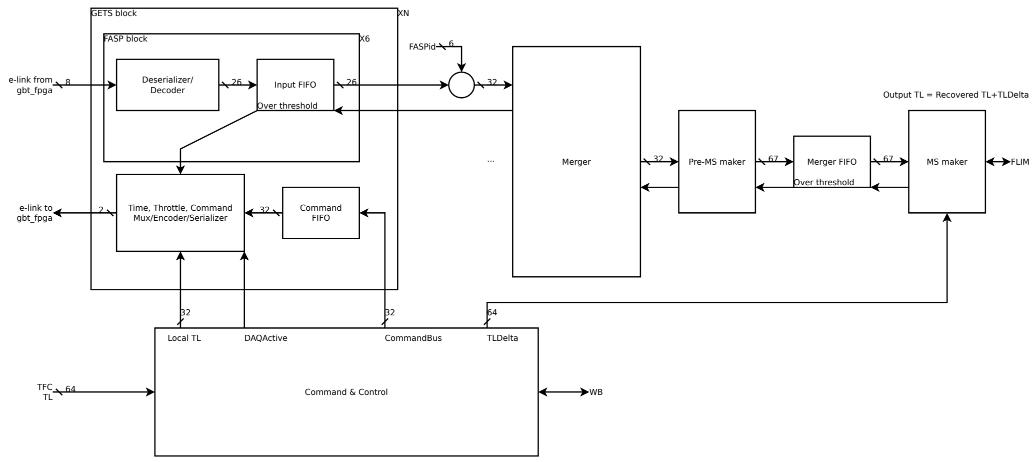


Figure B.7: Schematic overview of the TRD-2D CRI FPGA design

require an active FMC adapter board to distribute the clock and, more important, it would distribute at least one set of output signals of a FASPRO-DR/GETS stack to all 3 GBTx chips on the C-ROB3. Also, using 7 stacks for 1 C-ROB3 would reduce the total number of C-ROB3s only if some C-ROB3s connect stacks from different chambers, which may complicate the mechanics and the final assembly.

Using the experience gained by integrating the current readout chain, the final solution will be to integrate the main components, FASP, ADCs, FPGA, GBTx, GBT-SCA, etc. in one single board. This will result in a reduced material budget and several optimizations, especially if the high-speed transceivers from the FPGAs are used as uplinks to CRI replacing the slave GBTx chips.

According to Table B.6 the full TRD-2D will have 208 ROB3 providing 624 GBT links connected to 26 CRI2 cards.

B.3.2 The FASP data processing

The TRD-2D specific HDL blocks process the data sent by the front end described in the previous subsection. Because the digital part of the TRD-2D front end is implemented using an FPGA and may be changed, the development concentrated on the main data stream. The slow control part will be fully developed after the optimization of the main data stream.

Data flow and processing

The block diagram of the data flow in the TRD-2D specific part of CRI is presented in Fig. B.7. The data received from GBT-FPGA is first decoded and deserialized. A 30/32 bit serializer/deserializer is used over the e-link which may transmit 30 bits of which only

26 are currently used. The remaining 4 bits may be used later for a control response channel from the FEE to the CRI.

The decoded 26-bit word is pushed into one input FIFO. The "over threshold" signal of this FIFO is sent back to GETS as a throttling indication. The read part of the input FIFO feeds the merger. For each 26-bit stream entering the merger a 6-bit static FASPid label is attached, resulting in a 32-bit stream.

The merger performs the time-based merging of the input streams. The nanoslice timestamp is recovered for each stream. The merger generates an output nanoslice timestamp and sequentially plays the input streams corresponding to the active nanoslice. The throttling status of each channel is reconstructed and an error message is written at the output for each throttled FASP and for each nanoslice. Using this mechanism, it is possible to identify the cause for an empty nanoslice either due to lacking data from the pads or because the channel was throttled for a given FASP. The writing of the error messages may be suppressed by configuration.

The output of the merger is a stream of nanoslices with the input channels merged. This stream is processed by several blocks, including a FIFO, to make a microslice adequate to be sent to FLIM. The back pressure from FLIM is propagated back to all merger input channels and eventually to all GETS.

The operation of the chain is controlled by a `DAQActive` signal generated from comparing the TFC time counter with a register (`msboundary`) set through the slow control interface. If the TFC time counter has exceeded the `msboundary` then `DAQActive` is asserted. At the assertion of `DAQActive`, the TFC time counter is saved in a register (`TLDelta`) and a counter (*Local time counter*) is started from 0. This counter is used as a time counter for the entire chain. It is sent, in a deterministic time, to GETS where it is used for the generation of the time counter used to timestamp the messages. The timestamp recovered for the streams entering the merger refer to this local time counter. The reconstructed timestamp in the stream sent to FLIM is obtained in the last step by adding `TLDelta`.

Slow control

The current implementation for slow control is minimal, as the flexibility provided by the fact that GETS is implemented using an FPGA permits the delay of the development until enough experience is accumulated regarding the required controls. This strategy should permit the development of an optimized GETS which is important as the FPGA is placed on the detector and requires some treatment of radiation-induced upsets. Also, a minimal GETS translates into minimal power consumption and heat dissipation of the front end.

Currently, the slow control consists of a 32-bit channel broadcasted from CRI to all GETS for setting some registers, the most important of which is the one programming the DAC which provides the threshold voltage for FASP. When specific registers are written in CRI through the slow control channel, the corresponding values are written to the broadcast channel. For the rest of the time, a no-op word is written. No channel from GETS to CRI is currently implemented, however, such a channel may be easily implemented using some of the unused 4 bits of the main data stream.

B.4 GET4-based systems: TOF and BMON

B.4.1 TOF

The TOF subsystem consist out of two different types of front-end electronics (TOF-iROB and TOF-oROB). The TOF-iROB is optimized for the inner part of the TOF wall and the TOF-oROB is optimized for the outer part of the TOF wall. The main difference is in the mechanical design of these two parts of the wall and the resulting mechanical constraints for the FEE. The internal functionality, however, is the same and based on a single GBTx configured with normal GBT frames for the uplink.

For the inner wall the FEE sits on the backside of the gas box housing the MRPC detectors. Each FEE card has 32 input channels. Four PADI ASICs [83] are read out from 8 GET4 ASICs [84, 85]. Up to 5 FEE cards are served by one inner wall readout PCB (TOF-iROB). Each second GET4 is the SPI Master for the upfront sitting pre-amplifier and discriminator ASIC PADI. Via SPI commands the threshold on the input stage and the stretch factor of the discriminated signal can be set by the slow control path of GET4.

The TOF-iROB sits on the outer frame of the inner wall to minimize the radiation dose on these PCBs and optimize the usage of 40 e-links provided by one GBTx. The TOF-iROB is interconnected with patch, data and power cable with the FEE. The distances for these cables will not exceed 2 m. Due to mechanical arrangements not all TOF-iROBs will be equipped with 5 FEE cards.

For the outer wall the readout PCB (TOF-oROB) is connected directly to the TDC FEE sitting in a frame on the outer modules. Always two TOF-oROBs can read out one detector module housing 5 MRPCs and 10 PADI with GET4 FEE. The pre-amplifier sits inside the gas box, while the GET4 TDCs are connected outside.

Each TDC PCB houses 8 GET4 ASICs connected on the one side to the TOF-oROB with the GBTx and on the other side to the feed through PCB to connect to the PADI FEE.

Like for the inner wall, always 40 GET4 TDCs are read out via one GBTx. The e-links of GET4 and GBTx are configured to 80 Mbit/s which leads to a possible hit-rate of 1.8 MHits per GET4. It is considered to read out at least 24 GBTx ASICs with one CRI2 (cf. Sec. 5.2.4). Due to mechanical constraints on the outer wall not always all MTP-24 fibers can be be used, but the overhead is negligible.

Therefore, the readout topology of one CRI will look like the following:

- 1x GET4 = 4 Channel
- 1x GET4 FEE = 8x GET4 = 8 e-links 80 Mbit/s uplink and 20 Mbit/s downlink
- up to 5x GET4 FEE = 1x TOF-(i/o)ROB = 40 e-links = 1 GBTx

- up to 24x TOF-(i/o)ROBs = 1x CRI

According to Table B.7 the full TOF will have 614 ROB1 providing 614 GBT links connected to 20 CRI2 cards.

	article	outer	inner	BFTC	overall
1	GBT links per CRI	24	24	24	24
2	MTP cable type	24	24	24	24
3	ROB1	436	128	50	614
4	GBT links total	436	128	50	614
5	CRI cards total	19	6	3	28
6	MTP patch cables	38	12	6	56
7	1x MTP-24 to 24x-LC fan-out	38	12	6	56
8	MTP 12-fold adapter PP	4	1	1	6
9	MTP patch panel [HU]	1	1	1	3

Table B.7: Optical interface for the TOF. Lines 1-3 show input parameters from the FPGA design (1), the optical connectivity (2), and the detector design (3). Based on these, we derive the quantities in lines 4-9.

Clock distribution system for CBM TOF

The CBM TOF System aims for high precision measurement with a system resolution in the order of 80 ps. Therefore it is foreseen to distribute a very precise and stable 160 MHz clock in LVDS inside the cave. For stable data transmission and to be in sync with the other subsystems the 160 MHz copper clock needs to be phase stable with the GBTx clock, which is provided by the CRI via the TFC system. A scheme of the TOF clock tree with clock and control signal paths is shown in Figure B.8.

The TFC (shown on the right hand side in blue) distributes the global time information and the system clock of CBM via optical connection to all CRIs. A TFC slave, implemented on each CRI FPGA synchronizes itself to the TFC time and provides all needed clocks and the actual time information.

As the TOF FEE in the cave has its own low jitter clock distribution system, an optical link between a CRI and a GBTx inside the cave is required. The GBTx will recover the 160 MHz for the FEE from the optical link and will receive, via a deterministic link, a `SYNC` pulse. This pulse is needed to set or reset the internal time counter of the GET4. The `SYNC` signal can be used to check the synchronicity between all GET4. At the time of the `SYNC` signal, the internal 12-bit time counter of GET4 must be zero. Otherwise it will be reset to zero and an error will be issued. The “CLK / SYNC / CNTR Board”, which is a TOF-iROB in a different configuration, (see upper left part of Fig. B.8) is used for this task. Figure B.9 illustrates the connectivity of the ToF-iROB when it acts as a CLK / SYNC / CNTR board.

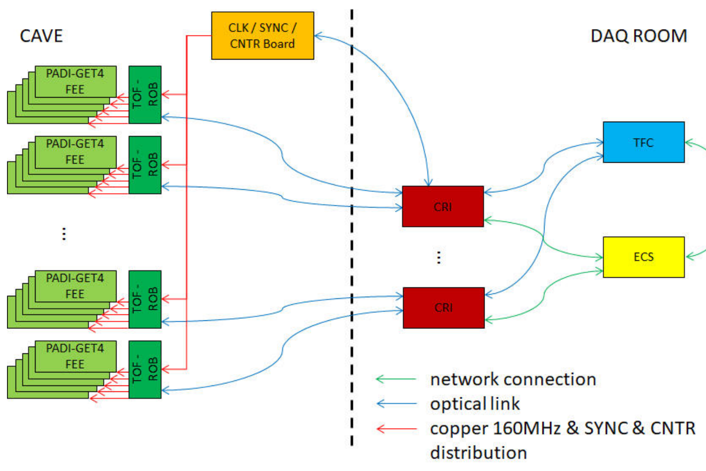


Figure B.8: Block Diagram of TOF Readout System

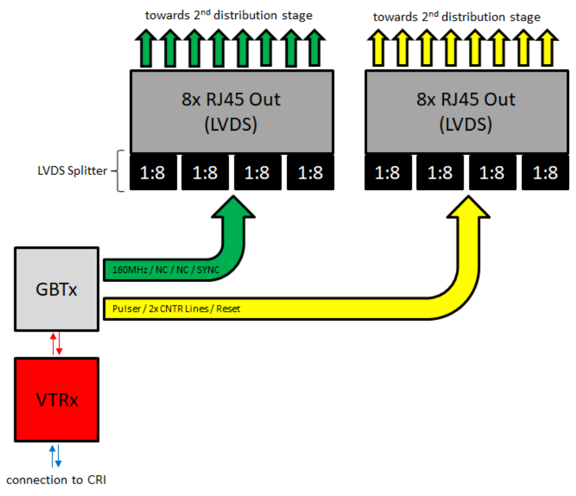


Figure B.9: Block diagram of the ToF-iROB used as CLK / SYNC / CNTR Board

The key component is the GBTx, which receives the `SYNC` signal and some control signals from the CRI via the optical link. These signals are send out via the e-links of the GBTx towards the FEE. In addition it generates a high quality 160 MHz clock on one of its dedicated clock outputs. The signals are duplicated eight times on the PCB and send out on the RJ45 outputs for direct connection to the readout boards or the 2nd distribution stage inside the cave.

B.4.2 BMON

The BMON subsystem consists of two diamond-based beam detector stations located in front of the CBM target chamber. One station is suited for a precise T0 measurement (BMON-T0), the second one will be used for beam monitoring, i. e., beam halo measurement (BMON-HALO). The T0-system aims for high precision time measurements with a system resolution in the order of 50 ps, which is needed for particle identification. The system should be able to handle beam intensities up to 10^7 ions/s, at higher beam intensities the BFTC system will take over. The HALO-station is foreseen to be used as an independent beam monitoring system, nevertheless it can be useful for background rejection and therefore the data of the system will be integrated into the CBM data stream. The readout system is currently under development and evaluation. It is planned to follow a "TOF-based" approach which reuses readout infrastructure developed by the TOF group (see previous section).

The diamond based T0 sensor will be equipped with a metallization arranged in 16 stripes on both sides. A strip orientation in x and y-directions will allow a position information of the beam particles. The analogue sensor signals will be sent to the PADI ASIC which are read out by the GET4 ASIC. In total, 32 analogue detector channels are read out by 32 GET4 ASICs in order to achieve the maximum rate capability, which will be up to 1 MHz per sensor strip. A dedicated Front-End Board is equipped with four GET4 ASICs, therefore in total eight boards are needed for the T0-system. Data of the 8 GBTx is sent to a single CRI board. The planned readout scheme of the T0 detector is schematically shown in Figure B.10.

The HALO detector will consist of a mosaic arrangement of four diamond sensors, each metallized with four stripes on both sides. This results in a total of 32 channels having to be read out which will be read out exactly like the T0 station. Sensor signals will be split and sent to a "stand-alone" DAQ system in order to realize a beam monitoring system which can be operated independent from the CBM DAQ. The beam monitoring system which was demonstrated in [72] is planned to be used for this purpose. The sensor orientation and the readout scheme is schematically shown in Figure B.11.

According to Table B.8 the full BMON will have 16 ROB1 providing 16 GBT links connected to 1 CRI2 card.

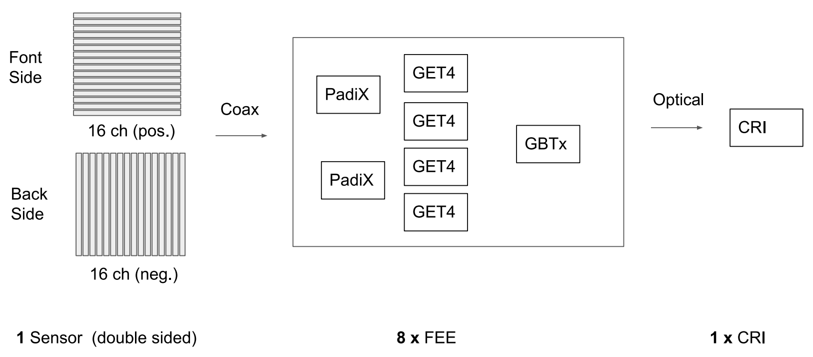


Figure B.10: Planned readout scheme of the double-sided T0 sensor. In total, 32 detector channels are read out by 16 GET4 ASICs in order to achieve the maximum rate capability. Data of the eight GBTx is sent to a single CRI board.

B.4.3 The GET4 data processing

One of the 47 optical links, which are available at the CRI1, is used to distribute a 160 MHz synchronous clock and a `SYNC` pulse to the GET4 (cf. Sec. B.4.1). For this purpose the TOF-ROB board in the iROB version is used. In the current FPGA design version from the other 46 optical links only 32 are used to connect GET4s through the TOF-ROB board, one TOF-ROB board has one GBTx and this GBTx is connected to 40 GET4 through 40 e-links.

At SIS100 the TOF will use the CRI2, which will allow for 24 optical links and a separate TFC connection. The CRI1 controls 32 GBTx connected to 1280 GET4, giving a total of

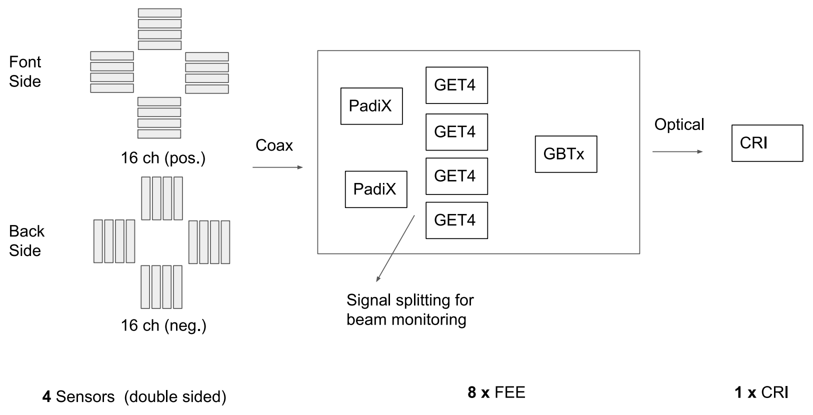


Figure B.11: Planned readout scheme of the HALO station. A mosaic arrangement of four diamond sensors is foreseen. The primary beam will be guided through the middle part where no sensor is located. In total, 32 detector channels are read out by 8 GET4 ASICs in order to achieve the maximum rate capability. An LVDS fan-out of the detector signal is needed for an independent beam monitoring system.

	article	rack
1	GBT links per CRI	16
2	MTP cable type	24
3	ROB1	16
4	GBT links total	16
5	CRI cards total	1
6	MTP patch cables	2
7	1x MTP-24 to 24x-LC fan-out	2
8	MTP 12-fold adapter PP	1
9	MTP patch panel [HU]	1

Table B.8: Optical interface for the BMON. Lines 1-3 show input parameters from the FPGA design (1), the optical connectivity (2), and the detector design (3). Based on these, we derive the quantities in lines 4-9.

5120 channels. In case of the CRI2, it could control 24 GBTx connected to 960 GET4, giving a total of 3840 channels.

In the following, a brief description of the GET4 clock generation module, download channel (TDC slow control) and upload channel (TDC data flow and processing) is given. This is also depicted in Figure B.12.

Clock and sync generation

This module is instantiated only one time and it is used to control the TOF-iROB board, which is connected to the CRI1 using an optical link (GBT link). The module controls two e-links, the first one is used to generate a 160 MHz clock used as the main clock for the GET4, which runs synchronously with the 40 MHz experiment clock. The second one generates a `SYNC` signal needed to synchronize all GET4 with the time counter.

To synchronize the GET4 the clock module calculates the time counter value for the next `SYNC` pulse and writes it in a GET4 register, and when the `SYNC` pulse arrives the GET4 internal time counter will be updated. After that, each 25.6 µs (4096 x 6.25 ns) the GET4 will send an epoch message and between the epochs the hit messages with extra timing information. The `SYNC` pulse period must be a multiple of 4096 of the GET4 clock and can be set through a register.

GET4 download channel

The GET4 download channel entity is instantiated 32 times and allows users to configure and control the GET4 and reconfigure it automatically if a communication problem is detected with the GET4. It consists of two submodules: the first one implements the GET4 control channel and the second one uses this channel to reconfigure the GET4 if

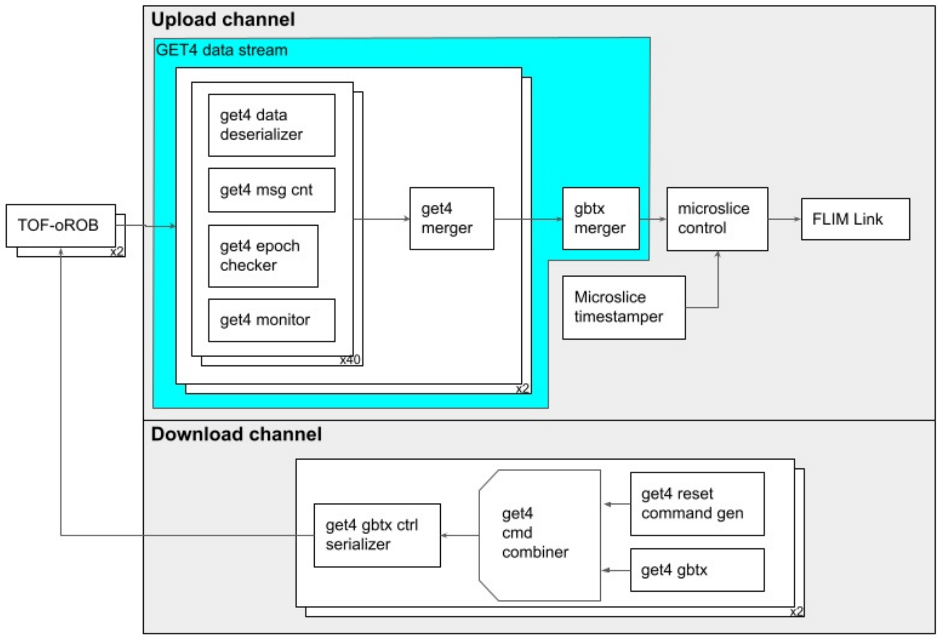


Figure B.12: Block diagram of the TOF CRI FPGA design

necessary. The GET4 reconfiguration module checks the integrity of the enabled GET4, in particular that epoch messages arrive every 25.6 µs, and that they match with the CRI time counter.

GET4 upload channel

The GET4 upload channel entity is instantiated as many times as FLIM links exists (16 times), and it consists of a microslice timestamper and a GET4 data stream.

GET4 microslice timestamper The microslice timestamper module converts the GET4 timestamp given in system clocks to nanoseconds and generates a pulse when the microslice duration has been reached. This signal will be passed to the common module microslice control.

GET4 data stream This module merges up to 80 GET4 data streams from two TOF-ROB cards into one FLIM stream. Once the GET4 has been initialized, it will start to send data through an 80 Mbit/s serial link. This data stream is composed of four types of messages: error, control, hit and epoch. The epoch message will be sent approximately every 25.6 µs and contains coarse timing information. If a hit is detected, a hit message will be sent containing the duration and the fine timing information. To reconstruct the absolute timing, the coarse information (from the epoch message) and the fine information (from hit message) must be used. This format was used in order to optimize the data overhead. Error and control messages can be sent in order to notify some error situations or status of the GET4. The data streams originating from the 80 GET4 must be time sorted and prepared to be written into the FLIM module. The GET4 data stream module consists of several submodules, the first one deserializes the GET4 data stream, the second one merges the data between two epochs of one GBTx (40 GET4) and writes it into a dual buffer memory. The last submodule reads out the FIFO data from three GBTx and merges it into one microslice/FLIM stream. This FIFO is needed in order to save the GET4 messages until the next epoch arrives because the epoch message must not be sent by the 40 GET4 at the same time.

B.5 MVD

The MVD readout chain follows the general concept for GBTx based systems. Apart from its own pixel sensor ASIC, it also employs dedicated ROB boards due to the required placement close to the target box.

B.5.1 MVD front ends

The MVD detector consists of MIMOSIS pixel sensors [86]. The exact configuration of the detector will vary depending on the experimental setup, and contain up to 300 individual sensor ASICs [21]. Hit information from each of its 516 096 self-triggered pixels is collected, encoded and stored for each readout cycle.

These cycles (“frames”) have a nominal duration of 5 µs. As hits of one frame cannot be subdivided into shorter time ranges, a microslice length of less than the frame duration should be avoided.

The data output of each sensor is realized by up to 8 e-links. The actual amount of data per unit area varies greatly with the position inside the detector, even within the 4.2 cm^2 of active area of one individual sensor. The total data bandwidth of 2.56 Gbit/s is only sufficient because of heavy spatial averaging and buffering of data on-chip. Therefore, several links per sensor will be used to transmit data in parallel. Depending on the corresponding maximum hit rate, the sensors are configured to operate in either a x2, x4 or x8 link configuration. The internal bandwidth of each sensor (10 Gbit/s) is chosen such that it will be able to handle all hits even during short periods of highest beam intensity fluctuations [75].

During these periods, data can be buffered in an on-chip memory. If this buffer happens to run full, the sensors provide an automatic rate reduction mechanism: The size of all subsequent frames is limited to below the available output bandwidth until the buffer reaches an acceptable level again. According to simulations, this mechanism is expected to trigger only rarely and on few, most-exposed sensors or during exceptional beam conditions [87]. In case of an overflowing buffer, a corresponding flag is activated in the frame output stream and signals truncation of the last hit data.

In total, 900 e-links will be used to transport the expected 5 GB/s of data at the nominal interaction rate for heavy ions. These links will be routed from the sensors through flex cables of very low material budget to the outside of the detector and further on through shielded, flexible cables to the outside of the vacuum chamber. Here, dedicated MVD-ROBs will be placed directly on the vacuum flanges. Like the ROB3 used in other subsystems, it will contain 3 GBTx, 1 GBT-SCA, and two transceiver modules (VTRx and VTTx). Each MVD-ROB will provide up to 30 e-links as data uplinks, 15 x 40 MHz clock lines and 2 e-links for GBT-SCA-based slow control to the MIMOSIS pixel sensors.

The geometric layout as well as the grouping of up to 8 e-links into one readout stream does not allow for a perfect usage of all available GBT uplinks so that a total of up to 120 4.8 Gbit/s GBT links will be required.

As one CRI board supports at least 24 GBT links (cf. Sec. 5.2.4), the complete MVD readout would require a minimum number of 4 boards. In order to ease the allocation of resources for the processing of pixel data per CRI board, at least 5 CRI boards will be employed. The amount of preprocessing possible on the FPGA needs to be assessed once the amount of free resources is known. Possible operations include cluster finding within one frame, merging clusters across frames or suppressing large clusters of hits. All these operations can be implemented in FLES software as well if required. At least one full data buffer per sensor (up to 50 kB) needs to be buffered to guarantee that frames from all sensors can be aligned.

Figure B.13 shows a block diagram of the MVD readout chain.

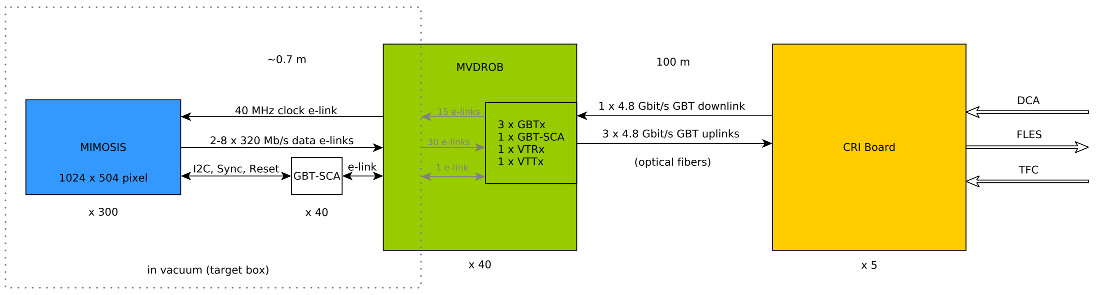


Figure B.13: The MVD readout scheme. Black arrows indicate connections between single instances of the main building blocks. The stated multiples of each building block are rounded and depend on the final MVD configuration.

B.5.2 MVD data processing

The data format of the sensor consists of 16-bit words of various types: header and subheaders, data, trailer and idle words. These have been designed to allow the readout electronics to automatically detect bit and word alignment as well as synchronize the up to 8 data streams from one sensor.

Sensor level data preprocessing

The sensor output is already zero-suppressed and encodes pixel hit data according to individual pixel locations on the matrix. A simplified cluster finding operation at the level of pairwise grouped pixel columns further compresses the hit data for up to 4 directly adjacent pixels.

Data processing on CRI boards

The hit data of MIMOSIS pixel sensors is tagged with a frame counter. An initial synchronization resets this counter to zero on all sensors. The FPGA design running on the CRI board will detect and trigger re-synchronization in case the frame counters begin to deviate. The sensors record hits within a defined frame time of 5 µs. The same duration will be applied as incremental timestamps to newly generated microslices. Those hold the merged hit data from all sensors connected to the corresponding CRI board and are forwarded to the FLES entry nodes. Mandatory data processing involves at least the removal and transformation of frame counters, pixel addressing, encapsulation and alignment words. The encoding of pixels within one sensor is replaced by a global sensor ID and pixel address, allocating 29 bits for each hit pixel. In this simplest form, the data format of an MVD microslice is a list of fixed-size entries representing hit pixels grouped by frame time. Depending on the available RAM on the final CRI board, further cluster finding may be possible. This would allow reduced resource usage on FLES nodes in the Green IT Cube computing farm and optimize the bandwidth of transmitted data [38].

B.5.3 MVD controls

With an intrinsic time resolution in the microsecond range, the use of high-precision timing on the e-links is not required. A common 40 MHz clock is distributed via e-links which is driving the PLL circuits on each sensor. Configuration and timing are controlled using the various buses and discrete I/Os of the GBT-SCA ASIC. These ASICs are placed in close vicinity of the sensors and provide

- an I^2C bus for each group of two sensors for loading pixel masks and threshold levels
- control signals to reset and synchronously start all sensors
- switch power lines to individually enable and disable groups of sensors
- ADCs for monitoring supply voltage and possibly current consumption
- additional control lines for basic environmental monitoring like temperature

Due to the physical setup of the detector in 8 individual half-planes, these services will be provided from separate GBT-SCAs that are distinct from those controlling the MVD-ROBs and are located inside the vacuum vessel. In the MVD, GBT-SCA can be employed to generate synchronization signals because the required timing precision is only on the order of 100 ns which is within the capabilities of the chip if the control link to the GBT-SCA is operated by the CRI FPGA.

B.6 RICH

B.6.1 The RICH readout chain

The CBM RICH detector front-end electronics (FEE) is an FPGA based design. The FEE is a composition of submodules with a backplane as the basic structure. It is designed to be as light tight as possible to be used as the connection part between radiator volume and readout volume of the detector. The side of the PCB pointing to the radiator gas hosts the H12700B-3 Hamamatsu MAPMTs with their recommended Pin Header for PMT signals, ground and high voltage connection. The inner side of the PCB is equipped with SAMTEC connectors. At these connectors the FEEs, the DiRICH boards, a data combiner board (Combiner board) and a power distribution board (power module) are connected. One standard backplane can host 6 MAPMTs in a 3×2 array. On the readout side, one MAPMT is connected to two individual DiRICH boards, resulting in 12 DiRICH boards per backplane.

The backplane connects the Serializer/Deserializer (SerDes) data lines of the DiRICH boards and additional low voltage differential signal (LVDS) lines with the combiner board. As the RICH FEE is used in triggered and self-triggered experiments, a trigger signal distribution is needed. The trigger signals are distributed from the combiner board to a trigger fan-out chip on the power module and further on to the individual DiRICH boards.

The heart of the RICH front-end electronics is the DiRICH board. Each of these cards hosts the pre-amplification stage of 32 MAPMT pixels, threshold setting infrastructure for all channels (two peripheral Lattice MachXO3LF-4300E FPGAs) and the Time-to-Digital Converter (TDC) including the TrbNet based data transport (central Lattice ECP5UM-85F-8BG381C FPGA).

In the case of the DiRICH, the already existing LVDS inputs of the central Lattice ECP5 FPGA are used as discriminators in order to filter the readout stage input for signals above a certain threshold. The threshold of each channel is configured individually with the two additional Lattice MachXO3 FPGAs, supplying the LVDS inputs with a threshold voltage. These two MachXO3 store the threshold voltage for each channel in the internal flash memory. The values are loaded at each power cycle but could also be set by an SPI connection from the central ECP5 FPGA.

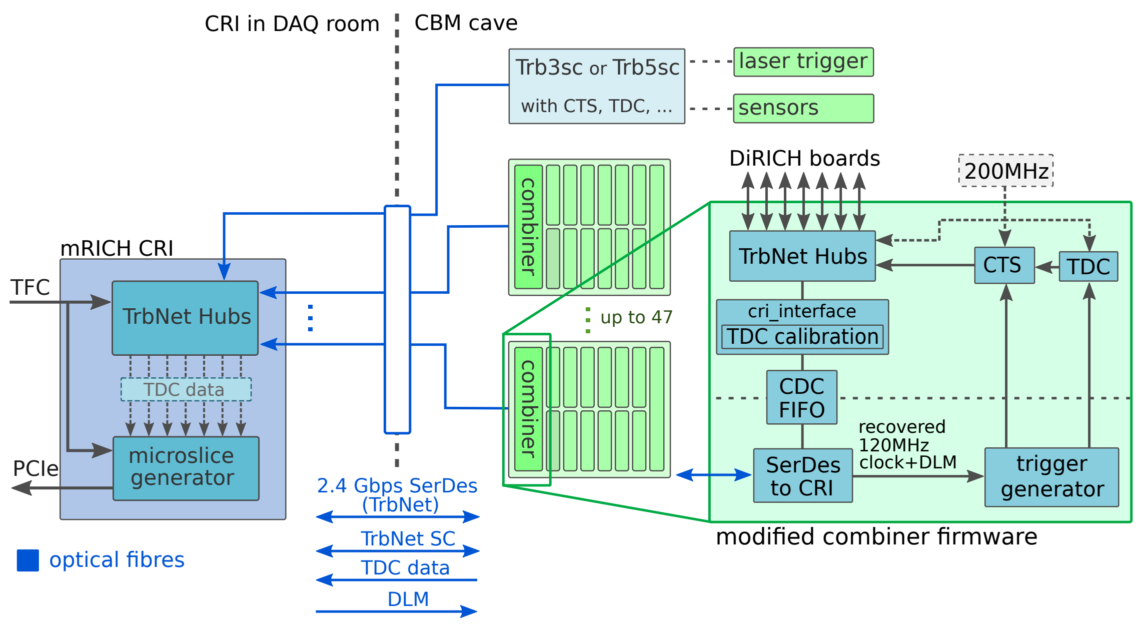


Figure B.14: Schematic overview of the CBM RICH readout concept. The combiner is a standalone readout board with an own RICH internal CTS and on board trigger generation. TDC data is calibrated and transmitted on the virtual data channel via TrbNet. A dedicated hub system on the CRI for CBM RICH data combines the combiner data and handles the slow control connection. Generated microslices are transmitted via PCIe to the FLES entry node.

The central FPGA of a DiRICH supports 32 input- and a reference-time TDC channel with rising and falling edge measurement by the use of a stretcher. A tapped delay line TDC is implemented on the FPGA, giving a resolution of roughly 10 ps and precision of around 12-30 ps, depending on the calibration method. The TDC data is locally buffered in a ring buffer until a trigger signal arrives. The size of the ring buffer and the trigger rate limits the possible data rate of the TDCs as only a limited amount of data words could be stored for a DiRICH per trigger signal. The standard buffer size of the DiRICH readout, storing all hits from the latest trigger, has a size of up to 499 words (1996 Byte). This limited buffer size could lead to losses of TDC data. To avoid this issue the readout rate has to be increased to a certain value, depending on the the interaction rate. To be independent from the general microslice rate and to deal with these readout rates, a sub-trigger is introduced. The data transport via SerDes from the DiRICH to the combiner board operates with 2.0 Gbit/s and thus not limiting the throughput.

All 12 DiRICH boards on a backplane send data via their SerDes media interfaces to the combiner board. The combiner board, equipped with a Lattice ECP3-150EA-8FN1156C FPGA, is the central component on a backplane for the data connection to the higher level readout structures. The core usage of the combiner board is the hub functionality. Data from a CRI is distributed to all DiRICH boards and vice versa. Next to the distribution of the trigger information from the RICH internal central trigger system (RCTS) and the data from the front-end, also the slow control is distributed to all DiRICH boards. The combiner is equipped with a 200 MHz clock that is distributed to all connected DiRICH boards. It is used as the base clock for the TDCs and the system clock of TrbNet.

Each of the combiner boards is connected to a CRI via optical fibers. The data is transported via SerDes and uses a 240 MHz connection. The 240 MHz originates from the 120 MHz clock which is recovered from the 40 MHz clock of the TFC system and is therefore synchronous to the CBM time. The combiner board uses clock recovery on the media interface connecting to the CRI to operate its media interface to the CRI with the 120 MHz. The main part of the ECP3 FPGA design, the link to the DiRICH boards as well as the DiRICH boards itself operate on the local 200 MHz clock. A dedicated FPGA based TDC stage is used to correlate the global CBM clock to the local FEE clock. This TDC data from the combiner board is part of the merged data of the hub. Due to the TDC design the reference channel information is always shifted by one readout and therefore, a rearrangement of the reference times of the TDC is performed later on the CRIs.

The TrbNet network and the TDCs are designed to operate in a triggered mode. To deal with the self-trigger of the CBM RICH, a pseudo trigger signal, based on the microslice rate, is generated. As the microslice rate could be rather small and as introduced earlier, a certain readout rate is needed to overcome possible data losses in the FEE buffers, additional sub-triggers can be generated. The sub-trigger feature generates, e. g., 16 readouts during one microslice. On the CRI the 16 readouts are combined to the data of one microslice. The generation of the sub-trigger is part of the CRI FPGA design. The trigger information is transmitted to the combiner boards by deterministic latency messages (DLMs). Each of these messages contains an 8 bit information package to distinguish single sub-triggers. A word alignment at each startup as well as phase alignment are performed to achieve a deterministic latency. Measurements in the laboratory proved the fixed latency for the transmitted DLM words.

Each combiner board has a RICH internal central trigger system (RCTS). The RCTS generates the readout signals and controls the full readout of the connected TrbNet system downwards the readout tree. Based on the external trigger information from the DLM and the status of the FEE, a trigger signal as well as trigger messages are transmitted and readout is initiated.

The use of one RCTS on each combiner is unique in the TrbNet world. Due to this concept each backplane of the RICH detector operates independently. This allows to add other boards than a combiner to the readout chain. Especially the use of TrbNet-compatible hardware (like the “Trb5sc” or “Trb3sc”) for sensor readout as well as additional TDC measurements of, e. g., a reference signal from a laser source could be implemented.

The TDC data from the FEE and the local TDC on the combiner is merged in the hub of the combiner. To allow for fast online data processing, an FPGA based online calibration is part of each combiner board. The TDC data from each channel is calibrated separately by means of a linear calibration. The data transmitted towards the CRI and packed into the microslice on the CRI is already calibrated and no additional fine-time calibration on the unpacking stage is needed.

The TDC data, as well as slow control data, is transmitted towards the CRI via the 240 MHz SerDes connection. The data transport is based on TrbNet. The data format of the transmission between combiner and CRI follows a specific protocol, shown in [88, p. 135].

The control of the FEE uses the slow control channel of TrbNet. All hub stages in the readout tree combine the slow control information up to the CRI. On the CRI a dedicated TrbNet link, connecting to a hub-board for slow control or a TrbNet-DCA bridge could be used. The TrbNet-DCA bridge allows to communicate with the TrbNet slow control via the CBM specific DCA. The communication interface is inspired by the known TrbNet readout softwares and allows the use of these existing tools also via the TrbNet-DCA bridge.

More detailed information on the implementations, the protocols and the overall communication is provided in [88]. Information about the TDC data and TrbNet in provided in [89, 90].

B.6.2 The RICH data processing

The electronics subtree of the RICH connecting to a single CRI was introduced in Section B.6. For the CBM RICH detector, the current planning foresees the CRI 1.0 as the final CRI version. In the FPGA of the CRI 1.0 the data generated by the front-end electronics is processed as follows:

The RICH CRI provides 47 data links. Here, each link receives FEE data, as well as slow control data, in the TrbNet data format [90]. The data from the fiber connections is received by the TrbNet media interface operated at 240 MHz and packed into a 16-bit word, that is part of a dedicated media interface data type.

The data from all these links is furthermore fed into a hub structure (see Fig. B.15): Each of the two SLRs of the FPGA are equipped with an identical hub structure to handle the FEE data. On SLR 0, up to 24 links are connected, while SLR 1 will connect to 23 links and the TFC system. Each of the hub structures is constructed as shown in Figure B.15. One control hub connects with one link to the DCA bridge (TrbNet slow control) and with two links (but also be extendable to, e. g., 4 links) to a second layer of two hubs.

The 32 bit data words and additional control signals corresponding to a link from the FEE are furthermore transmitted to a *trb parser* entity. Data from the *trb parser* is transmitted as an AXI4-Stream to a microslice filter entity (*ms filter*). After the *ms filter*, the data is forwarded via AXI4-Stream to an entity *ms gen*, the microslice generator. The output of *ms gen* is in the microslice format and can be transmitted to the FLIM module. A state machine controls the data stream towards the FLIM and, e. g., discards data.

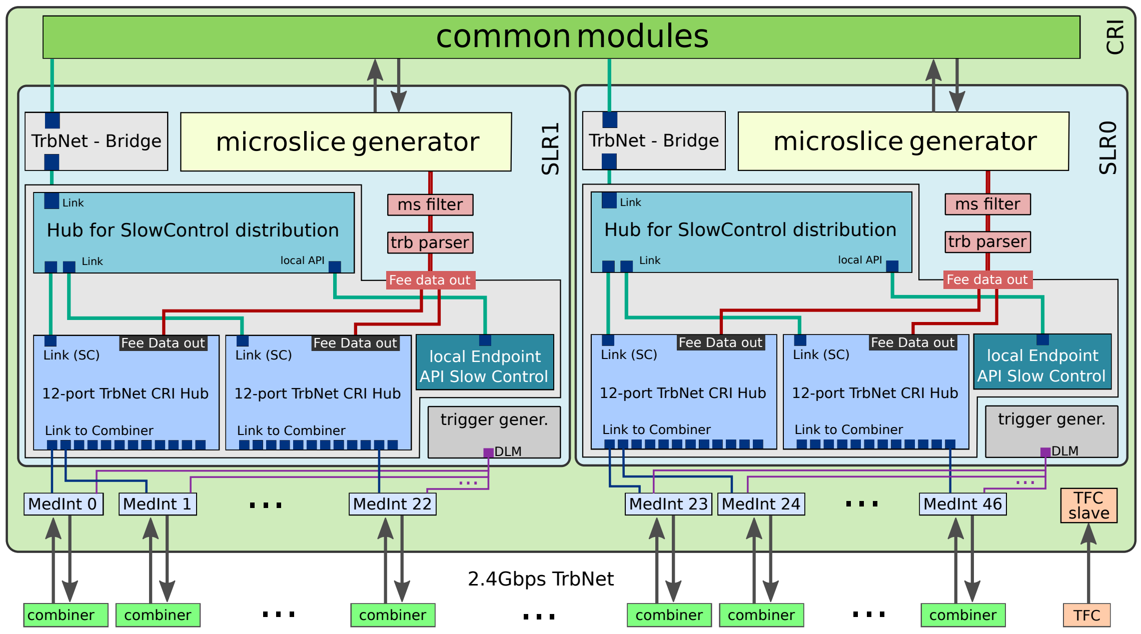


Figure B.15: Schematic overview of the RICH CRI FPGA design

TRB parser The *trb parser* entity receives TRB data from a hub data interface and parses the content. It expects TDC data words from different DiRICH endpoints as well as a RICH internal Central Trigger System (RCTS) packet as the last sub-sub-event. Only valid TRB data is output to an AXI4-Stream interface with configurable data width and a prepended header with meta-info. In case of any format error or buffer overflow, the complete input packet is discarded. The output data of the parser is furthermore aligned with the FLIM data width.

Microslice filter The *ms filter* entity filters TRB-data packets according to their trigger time. It requires the trigger time to be present in the first 64-bit word of the input-data packet. The microslice filter receives packets via an AXI4-Stream interface and controls the data flow depending on the time of the microslice or not valid data. The source logic should contain a buffer large enough to contain the largest packets that may arrive. It should assert a valid signal for a new packet only after it has buffered the packet completely and verified it to be valid.

Microslice merger The *ms gen* entity is the merger for the data from all connected valid AXI4-Stream links (links from combiner boards). Next to the merging of the different links, the entity also controls the packaging of the microslice with FEE data generated by sub-triggers. The full control mechanism is based on the selected time limits of the current run as well as the TFC time counter.

B.7 PSD

The PSD detector (cf. Sec. 1.5) readout chain had been developed by a group from INR, Moscow. A detailed description could not be completed for this document due to the circumstances described in the Preface. The replacement of the PSD detector will have different front-end electronics, but is expected to have a similar data volume. Therefore, for completeness, only a very brief summary of the PSD readout concept is given here.

The signals from the Silicon Photomultipliers were, after suitable analog shaping, continuously digitized with sampling ADCs. All further processing, such as baseline correction and further filtering, hit detection, and estimation of the time and amplitude of a hit, was performed in an FPGA.

The prototype used an ADC board initially designed for the ECAL detector of the PANDA experiment. This 64-channel board is based on two Kintex 7 (`xc7k160`) FPGAs and `LTM9011` ADCs with 14-bit resolution and a digitization rate of up to 125 Msps. The communication between the ADC board and the CRI was realized with an FPGA-to-FPGA GBT link, allowing to handle clock and time distribution as it is done between a TFC Submaster and a CRI (cf. Sec. 6.3).

One PSD detector module was successfully tested in the 2020 (see [91]) and 2021 (see Fig. 7.7) mCBM campaign. More information about the PSD readout chain can be found in [92, 93].

Appendix C

The CBM Collaboration

Aligarh, India, Department of Physics, Aligarh Muslim University
N. Ahmad, M.D. Azmi, M.M. Khan, O. Singh

Beijing, China, Department of Engineering Physics, Tsinghua University
Zhi Deng, Dong Han, Yuanjing Li, Botan Wang, Yi Wang, Qiunan Zhang[1], Xianglei Zhu

Berlin, Germany, Zuse Institute Berlin (ZIB)
A. Reinefeld, F. Salem, F. Schintke

Bhubaneswar, India, Institute of Physics
B. Mallick, P.K. Sahu, S.K. Sahu

Bhubaneswar, India, National Institute of Science Education and Research (NISER)
V.K.S. Kashyap, B. Mohanty, R. Singh

Bucharest, Romania, Horia Hulubei National Institute of Physics and Nuclear Engineering (IFIN-HH)
A. Bercuci, M. Petriş, M. Petrovici, L. Radulescu, C. Schiaua

Bucharest, Romania, Atomic and Nuclear Physics Department, University of Bucharest
M. Cãlin, A. Jipa, I. Lazanu, O. Ristea, N.G. Tuturas

Budapest, Hungary, Eötvös Loránd University (ELTE)
M. Csanád

Budapest, Hungary, Institute for Particle and Nuclear Physics, Wigner Research Centre for Physics, Hungarian Academy of Sciences
G. Balassa, Gy. Wolf

Chandigarh, India, Department of Physics, Panjab University
L. Kumar

Chongqing, China, Chongqing University
Liang-ming Pan, Qiqi Wu, Wenxiong Zhou

Darmstadt, Germany, Facility for Antiproton and Ion Research in Europe GmbH (FAIR)
E. Clerkin, J. Eschke[2], P. Gasik[2], O. Keller, P.-A. Loizeau, W.F.J. Müller, A. Rost[3], K. Schünemann[2], A. Senger, P. Senger[4], D. Smith

Darmstadt, Germany, GSI Helmholtzzentrum für Schwerionenforschung GmbH (GSI)
M. Al-Turany[9], M. Bajdel, D. Bertini, O. Bertini, P. Dahm, H. Darwish[4], H. Deppe, M. Deveaux[4], A. Dubla, I. Elizarov, D. Emschermann, H. Flemming, P. Foka, U. Frankenfeld, V. Friese, J. Frühauf, S. Gorbunov, J.M. Heuser, R. Holzmann, K. Ismail, R. Kapell,

R. Karabowicz, I. Keshelashvili, M. Kiš, K. Koch, P. Koczoń, F. Kornas[3], D. Kresan, A. Lebedev, J. Lehnert, S. Löchner, O. Lubynets[4], A. Lymanets, O. Maragoto Rodríguez[4], A.M. Marin Garcia, J. Markert, D. Miskowiec, F. Nickels, J. Pietraszko, D. Ramirez[4], D. Rodríguez Garces[4], A. Rustamov, C.J. Schmidt, I. Selyuzhenkov, M. Shiroya[4], C. Simons, C. Sturm, M. Teklishyn[5], J. Thaufelder, A. Toia[4], M. Traxler, F. Uhlig, I. Vassiliev, O. Vasylyev, R. Visinka, A. Wilms, S. Zharko, P. Zumbruch

Darmstadt, Germany, Institut für Kernphysik, Technische Universität Darmstadt
T. Galatyuk[2], V. Kedych, W. Krueger, F. Seck

Dresden, Germany, Institut für Strahlenphysik, Helmholtz-Zentrum Dresden-Rossendorf (HZDR)
B. Kämpfer[10], D. Stach

Frankfurt, Germany, Frankfurt Institute for Advanced Studies, Goethe-Universität Frankfurt (FIAS)
A. Belousov, N. Bluhme, J. de Cuveland, H. Hartmann, K. Hunold, D. Hutter, I. Kisel, G. Kozlov, R. Lakos, V. Lindenstruth[2], A. Redelbach, F. Weiglhofer, G. Zischka

Frankfurt, Germany, Institut für Kernphysik, Goethe-Universität Frankfurt
V. Akishina[2], J. Andary, H. Appelshäuser, B. Arnoldi-Meadows, C. Blume[2], H. Cherif[2], M. Esen, I. Fröhlich[2], D. Giang, S. Gläßel, B. Gutsche, M. Koziel, J. Michel, C. Müntz, A. Rodríguez Rodríguez[2], F. Roether, D. Spicker, J. Stroth[2]

Frankfurt, Germany, Institute for Computer Science, Goethe-Universität Frankfurt
A. Adler, T. Janson, U. Kebschull, D. Schledt

Gießen, Germany, Justus-Liebig-Universität Gießen
M. Becker, M. Beyer, M. Dürr, C. Feier-Riesen, R. Haas, C. Höhne[2,11], S. Lebedev, J.H. Otto, A.A. Weber[11], F. Zorn

Guwahati, India, Nuclear and Radiation Physics Research Laboratory, Department of Physics, Gauhati University
B. Bhattacharjee, S. Gope

Hefei, China, Department of Modern Physics, University of Science & Technology of China (USTC)
Dongdong Hu, Yongjie Sun, Zhengyang Sun, Kaiyang Wang, Tianxing Wang, Xinjian Wang, Junfeng Yang, Jian Zhou

Heidelberg, Germany, Physikalisches Institut, Universität Heidelberg
I. Deppner, N. Herrmann[2], Yue Hang Leung, D.A. Müller, E. Rubio, Y. Söhngen, P. Weidenkaff

Heidelberg, Germany, Institut für Technische Informatik, Universität Heidelberg
P. Fischer

Indore, India, Indian Institute of Technology Indore
S.K. Kundu, A. Roy, R. Sahoo

Jammu, India, Department of Physics, University of Jammu
A. Bhasin, A. Gupta, S. Mahajan, S.S. Sambyal

Karlsruhe, Germany, Karlsruhe Institute of Technology (KIT)
M. Balzer, J. Becker, T. Blank, M. Caselle, V. Sidorenko, E. Trifonova, K.L. Unger, M. Weber

Kharagpur, India, Indian Institute of Technology Kharagpur
A.K. Singh

Kolkata, India, Department of Physics, Bose Institute
S. Biswas, S. Chatterjee, R. Das, Supriya Das, S.K. Ghosh, S.K. Prasad, S. Raha, R. Ray, S. Roy[2], A. Sen

Kolkata, India, Department of Physics and Department of Electronic Science, University of Calcutta
A. Bhattacharyya, A. Chakrabarti, R. Ganai, G. Gangopadhyay

Kolkata, India, Variable Energy Cyclotron Centre (VECC)
A. Agarwal, Z. Ahammed, P.P. Bhaduri, Souvik Chattopadhyay, Subhasis Chattopadhyay[6], A.K. Dubey, C. Ghosh, M. Mandal, E. Nandy, J. Saini, V. Singhal

Kraków, Poland, AGH University of Science and Technology (AGH)
K. Kasiński, R. Kłeczek, W. Kucewicz, P. Otfinowski, R. Szczygieł, W. Zubrzycka

Kraków, Poland, Marian Smoluchowski Institute of Physics, Jagiellonian University
J. Brzychczyk, D. Gil, P. Lasko, K. Łojek, Z. Majka, R. Płaneta, P. Staszel, A. Wieloch

Kyiv, Ukraine, High Energy Physics Department, Kiev Institute for Nuclear Research (KINR)
O. Kshyvanskyi, V. Kyva, V. Militsija, M. Pugach, V. Pugatch, D. Storozhyk

Kyiv, Ukraine, Department of Nuclear Physics, Taras Shevchenko National University of Kyiv
O. Bezshyyko, L. Golinka-Bezshyyko, I. Kadenko, O. Lavoryk, V. Plujko

Münster, Germany, Institut für Kernphysik, Westfälische Wilhelms-Universität Münster
A. Andronic, R. Berendes, D. Bonaventura, F. Fidorra, N. Heine, P. Kähler, Ch. Klein-Bösing, M. Kohn, A. Meyer-Ahrens, H. Morgenweck, P. Munkes, A. Puntke, H. Schiller, P.M. Schneider, L. Wahmes, R. Weber, J.P. Wessels

Prague, Czech Republic, Czech Technical University (CTU)
P. Chaloupka, R. Dvořák, K. Haismanová, O. Hofman, V. Petráček

Pusan, Korea, Pusan National University (PNU)
In-Kwon Yoo

Řež, Czech Republic, Nuclear Physics Institute of the Czech Academy of Sciences
L. Chlad[7], P. Chudoba, A. Kugler, A. Prozorov

Srinagar, India, Department of Physics, University of Kashmir
A. Ahmad, S.A. Bhat, T.A. Bhat, W.A. Bhat, M.F. Mir

Tübingen, Germany, Physikalisches Institut, Eberhard Karls Universität Tübingen
K. Agarwal, S. Bhalerao, Susovan Das, S. Khan, V. Klochkov, S. Mehta, I. Panasenko[5], H.R. Schmidt[2], E. Volkova

Varanasi, India, Department of Physics, Banaras Hindu University (BHU)
A. Kumar, S. Pandey, B.K. Singh, C.P. Singh

Warsaw, Poland, Faculty of Physics, Warsaw University of Technology
J. Pluta, D. Wielanek, H. Zbroszczyk

Warsaw, Poland, Institute of Electronic Systems, Warsaw University of Technology
M. Gumiński, M. Kruszewski, P. Miedzik, K. Poźniak[8], R. Romaniuk, W. Zabołotny[8]

Warsaw, Poland, Faculty of Physics, University of Warsaw
T. Matulewicz, K. Piasecki

Wuhan, China, College of Physical Science and Technology, Central China Normal University (CCNU)
Sheng Dong[1], Feng Liu, Xiaofeng Luo, Shusu Shi, Zhongbao Yin, Xiaoming Zhang, Yu Zhang, Daicui Zhou

Wuppertal, Germany, Fakultät für Mathematik und Naturwissenschaften, Bergische Universität Wuppertal
K.-H. Becker, J. Förtsch, K.-H. Kampert, V. Patel, C. Pauly, D. Pfeifer, T. Povar, P. Subramani

Yichang, China, College of Science, China Three Gorges University (CTGU)
Sheng-Qin Feng, Ke-Jun Wu, Tao Xiong, Sheng Zheng

The following colleagues from Russian institutes did contribute to the development of CBM but are not listed as authors following the decision of the CBM Collaboration Board on May 18, 2022:

A. Akindinov, P. Akishin, E. Akishina, I. Alekseev, E. Alexandrov, I. Alexandrov, A. Andomina, E. Atkin, N. Baranova, S. Belogurov, D. Blau, A. Bychkov, A. Demanov, D. Dementiev, E. Dorenskaya, V.V. Elsha, O. Fateev, D. Finogeev, O. Golosov, S. Golovnya, M. Golubeva, D. Golubkov, S. Gorokhov, F. Guber, Yu. Gusakov, D. Ivanishchev, V. Ivanov, A. Ivashkin, A. Izvestnyy, N. Kargin, D. Karmanov, N. Karpushkin, E. Kashirin, A. Kazantsev, G. Kekelidze, A. Khanzadeev, F. Khasanov, A. Kiryakov, S. Kiselev, L. Kochenda, A. Kolozhvari, M. Korolev, I. Korolko, P. Kravtsov, A.V. Kryanev, E. Kryshen, I. Kudryashov, V. Ladygin, I. Lobanov, E. Lobanova, N. Lyublev, A. Makhnev, A. Malakhov, D. Malkevich, V. Manko, M. Merkin, K. Mikhailov, S. Morozov, Yu. Murin, V. Nikulin, P. Parfenov, S. Parzhitskiy, O. Petukhov, V. Plotnikov, M. Prokudin, E. Rostchin, A. Ryabov, Yu. Ryabov, I. Segal, A.R. Serazetdinov, A. Shabanov, A. Shabunov, A. Semennikov, A.D. Sheremetiev, S. Shirinkin, M. Shitenkow, I. Sibiryak, N. Sukhov, R. Sultanov, D. Svirida, A. Taranenko, V. Troshin, Yu. Tsyupa, A. Vorobiev, A. Voronin, I. Yushmanov, Yu. Zaitsev, N.I. Zamiatin, A. Zinchenko, I. Zivko

Additional affiliations:

[1]Physikalisches Institut, Universität Heidelberg, Heidelberg, Germany

[2]GSI Helmholtzzentrum für Schwerionenforschung GmbH (GSI), Darmstadt, Germany

[3]Institut für Kernphysik, Technische Universität Darmstadt, Darmstadt, Germany

[4]Institut für Kernphysik, Goethe-Universität Frankfurt, Frankfurt, Germany

[5]High Energy Physics Department, Kiev Institute for Nuclear Research (KINR), Kyiv, Ukraine

[6]Department of Physics, Bose Institute, Kolkata, India

[7]Czech Technical University (CTU), Prague, Czech Republic

[8]Faculty of Physics, University of Warsaw, Warsaw, Poland

[9]also: European Organization for Nuclear Research (CERN), Geneva, Switzerland

[10]also: Technische Universität Dresden, Dresden, Germany

[11]also: Helmholtz Research Academy Hesse for FAIR

List of Figures

List of Tables

Glossary

ADC analog-to-digital converter 135, 137, 145, 149, 150, 152–154, 165, 171

ADU analog-to-digital units 135

AGWB Address Generator for Wishbone 101, 102, 108

alarm system system for rapid notifications (see 5.3.2) 94, 103, 104, 109

ALICE A Large Ion Collider Experiment 95

API application programming interface 65–67, 75–78

ARQ Automatic Repeat Request 145

ASCII American Standard Code for Information Interchange 102

ASIC application-specific integrated circuit 25–27, 32, 89, 93, 94, 96, 99, 104, 107, 109, 110, 116–118, 132, 137–145, 149, 150, 152, 156, 159, 160, 163, 165

ATLAS A Toroidal LHC Apparatus 95, 96

AXI4-Stream stream variant of the Advanced eXtensible Interface 4 57, 100, 151, 169, 170

BAR Base Address Register 59, 66, 100, 101

BFTC Beam Fragmentation Time-zero Counter 7, 18, 98, 159

BMON beam monitor 19, 33, 110, 131, 159, 161

BNL Brookhaven National Laboratory 96, 97

BRAM block random access memory 99

CBM Compressed Baryonic Matter 7, 10, 12, 13, 15, 17, 19, 21–27, 29–31, 33, 35–39, 41, 42, 44–46, 52, 71, 72, 77–81, 84, 86–88, 90, 91, 93, 94, 96–99, 104–107, 110, 112–114, 116, 117, 119, 120, 122–124, 127, 129, 133–135, 137, 139, 145, 157, 159, 166–169

CDC clock domain crossing 105, 106

CERN European Organization for Nuclear Research 10, 26, 96, 104, 138

CI/CD continuous integration - continuous delivery 106

COTS commercial off-the-shelf 35, 80, 81

CPU central processing unit 19, 21, 54, 60–63, 70, 72, 75, 79, 81

CRC cyclic redundancy check 53, 145, 148

CRI Common Readout Interface 22, 24–27, 31–33, 35, 40, 42, 49–54, 57, 58, 67, 68, 71, 74, 75, 79–82, 89, 92–109, 113–119, 122, 124, 127, 132–135, 137–140, 142, 143, 145, 150, 154–157, 159, 160, 162, 164–169, 171

CRI1 CRI prototype board 70, 71, 96–100, 105, 107, 110, 115, 116, 119, 120, 122, 140, 147, 160, 161

CRI2 CRI production board 97–100, 110, 111, 115, 140, 142–144, 150, 153, 154, 156, 157, 159–161

CROB Common ReadOut Board 135, 139

CWDM coarse wavelength-division multiplexing 84

DAC digital-to-analog converter 153, 155

DAQ data acquisition 19, 24, 26, 31, 41, 79, 107, 116, 117, 119, 120, 122, 127, 159

DCA Device Control Agent 93, 94, 101, 103, 104, 107–109, 169

DDS direct digital synthesis 95, 97

DiRICH DIRC and RICH FEE and digitizer board 120, 134, 166–168, 170

DLM deterministic latency message 27, 116, 168

DMA direct memory access 26, 47, 53, 54, 58–70, 72, 74, 75, 94, 100, 101, 107, 124

DNA unique device identifier 103

downlink connection in DAQ to FEE direction 27, 94–97, 104, 105, 113–117, 145, 146, 150, 156

e-link electrical link between GBTx and FEE ASIC 25–27, 107, 109, 116, 135, 137–144, 147–154, 156, 159–161, 163–165

ECAL Electromagnetic Calorimeter 171

EDC experiment and detector control 77, 89, 90, 93, 104, 107

EDR Enhanced Data Rate — InfiniBand specification typically associated with 100 Gbit/s links 84, 85, 122

epoch time interval defined by the size of a timestamp (see 2.2.5) 28, 133, 151–153, 161–163

epoch message message that marks the beginning of a new epoch (see 2.2.5) 28, 32, 41, 42, 133, 135, 145, 147, 148, 150, 153, 161–163

FAIR Facility for Antiproton and Ion Research 7, 10, 11, 15, 22, 23, 36, 80, 83, 87, 88, 96

FASP Fast Analog Signal Processor 120, 150, 152–155

fast control handling of congestion, throttling and system-wide state changes in the TFC (see 6.4) 35, 94, 99, 104, 113, 115–117

FEB Front-End Board 25, 26, 140–143, 145, 146, 150, 159

FEB8 8-ASIC Front-End Board 138–140, 143

FEE front-end electronics 22–24, 26–28, 39, 90, 92–95, 104, 113, 116, 118, 155–157, 159, 166, 168–170

FELIX Front-End LInk eXchange 96

FIFO first in – first out queue 116–118, 137, 152, 155, 163

FLES First-level Event Selector 22–27, 31–41, 43, 44, 46, 48, 49, 52, 53, 57, 58, 62, 64, 67, 68, 74–76, 79–81, 83, 84, 86, 88–91, 93, 94, 100, 101, 108, 110, 113, 116, 119, 120, 122, 124, 164, 165, 167

FLIM FLES Interface Module 53–59, 61–70, 72, 73, 75, 93, 94, 99–101, 107, 108, 124, 147, 152, 155, 162, 163, 169, 170

FMC FPGA Mezzanine Card 154

FPGA field-programmable gate array 7, 21, 22, 26, 31, 35, 53, 54, 57, 58, 61, 62, 65, 66, 68, 93–109, 114, 119, 135, 137, 144, 146–148, 153–155, 157, 160, 164–169, 171

FSM finite-state machine 61

FWHM full width at half maximum 114

GBT GigaBit Transceiver 25, 26, 96, 98, 99, 104, 137, 140, 142, 143, 153, 156, 164

GBT link optical link using GBT frame protocol 26, 27, 93, 94, 96–99, 104, 106–109, 113, 114, 116, 117, 143, 144, 154, 157, 159, 161, 164, 171

GBT-FPGA GBT FPGA core 26, 27, 96, 104–106, 116, 151, 154

GBT-SCA GBT Slow Control Adapter ASIC 96, 139, 154, 164–166

GBTx GigaBit Transceiver ASIC 25–27, 93–96, 104–107, 110, 116, 137–139, 142, 143, 153, 154, 156, 157, 159–161, 163, 164

GEANT GEometry ANd Tracking simulation code 120, 121

GEANT3 GEometry ANd Tracking simulation code; version 3 134, 135

GEANT4 GEometry ANd Tracking simulation code; version 4 135

GEM gas electron multiplier 120, 141, 142

GET4 Gsi Event-driven Time-to-digital Converter 114, 120, 156, 157, 159–163

GETS Generic Event Time-stamping Streamer 153, 155

GPU graphics processing unit 19, 21, 36

GSI GSI Helmholtzzentrum für Schwerionenforschung 19, 22–24, 35, 96, 110, 119, 122, 129

HADES High-acceptance Di-electron Spektrometer 30, 117, 129

HALO beam halo detector 19, 131, 159, 160

HCA host channel adapter 66, 74, 75, 81, 84, 86, 88

HCTSP Hit Control Transfer Synchronous Protocol 99, 109, 116, 139, 143–146, 151

HDL hardware description language 53, 58, 154

HDR High Data Rate — InfiniBand specification typically associated with 200 Gbit/s links 72, 81, 86, 122

hit registered detector signal 21, 24, 26–28, 32, 38, 41, 94, 113, 116, 117, 131–135, 137, 138, 143–145, 147–150, 161–165, 167, 171

HPC high-performance computing 35, 72, 79

HU height unit 122

I^{2}C Inter-Integrated Circuit 96, 107, 108, 139, 165

I/O input–output 96, 165

InfiniBand a computer networking standard 22, 24, 31, 35, 38, 66, 72, 74, 75, 78–81, 83–86, 122

IOMMU input–output memory management unit 60, 61, 66, 67

JINR Joint Institute for Nuclear Research 10

LHC Large Hadron Collider 10, 26, 27, 113

LHCb Large Hadron Collider beauty 95

lpGBT low-power GBT 116

lpGBT-FPGA lpGBT FPGA core 116

LSB least significant bit 145

LUT look-up table 99

LVDS Low-voltage differential signaling 25, 96, 157, 160, 166

MAPMT Multi-Anode Photo-Multiplier Tube 17, 114, 133, 134, 166

MAPS monolithic active pixel sensor 17

mCBM CBM full-system test setup 19, 29, 78, 79, 96, 97, 99, 112, 119–126, 128, 129, 134, 139, 171, 178

MGT multi-gigabit transceiver 95, 96, 104

microslice container with detector data (see 4.2.1) 24, 32, 41, 43–46, 48, 50, 51, 53–56, 58, 61, 62, 64, 68, 69, 71, 73, 75, 89, 94, 113, 115, 135, 144, 147–149, 152, 155, 162, 163, 165, 167–170

MIP minimum ionizing particle 135

MMCM Mixed-Mode Clock Manager 105, 144

MRPC multi-gap resistive plate chamber 18, 156

MSB most significant bit 138, 145

MSV Modularized Start Version 15

MUCH Muon Detection System 18, 31, 99, 111, 120, 124, 131, 134, 136, 137, 140–143, 145

MVD Micro Vertex Detector 12, 17, 31, 33, 39, 45, 110, 131, 163–166

MVD-ROB MVD ReadOut Board 164, 166

NICA Nuclotron-based Ion Collider fAcility 10

NUMA Non-uniform memory access 70

PADI Preamplifier Discriminator 156, 159

PANDA Antiproton Annihilation at Darmstadt 171

PCB printed circuit board 97, 98, 138, 141, 156, 166

PCIe Peripheral Component Interconnect Express 22, 26, 40, 58–71, 73, 80, 81, 89, 93, 94, 96–102, 107, 110, 167

PDA Portable Driver Architecture 65

PIO programmed I/O 26, 63, 64, 100–102, 107

PLL phase-locked loop 95, 115, 116, 153, 165

PMT Photo-Multiplier Tube 166

POSIX Portable Operating System Interface 67, 73

PPS pulse per second 96, 115

PSD Projectile Spectator Detector 7, 18, 31, 33, 111, 120, 127, 136, 171

QA quality assurance 39, 67, 90, 92

QCD quantum chromodynamics 7, 10

QGP quark–gluon plasma 10

QoS quality of service 84

RAM random-access memory 80, 148, 165

RCTS RICH internal Central Trigger System 167, 168, 170

RDMA remote direct memory access 35, 38, 67, 72, 74, 75, 78, 79

RHIC Relativistic Heavy Ion Collider 10

RICH Ring-Imaging Cherenkov Detector 17, 31, 33, 94, 98, 111, 114, 120, 124, 127, 133–135, 166–170

ROB ReadOut Board 25–27, 93, 96, 133, 139, 140, 163

ROB1 ReadOut Board with one GBTx 96, 98, 99, 141–143, 157, 159

ROB3 ReadOut Board with three GBTx (one master, two slaves) 96, 98, 99, 138, 141, 142, 144, 147–151, 153, 154, 164

RoCE RDMA over Converged Ethernet 72

ROM read-only memory 102

RPC remote procedure call 107–109, 120, 129, 141

RPC single-gap resistive plate chamber 142, 143

RTT round-trip time 85

SATA Serial AT Attachment 153

SerDes Serializer/Deserializer 166–169

SFP Small Form-factor Pluggable 96

SIS100 Schwerionensynchrotron 100 7, 10, 11, 14, 15, 17–19, 29, 30, 33, 80, 81, 117, 122, 124, 160

SIS18 Schwerionensynchrotron 18 30, 117, 119, 124, 129

slow control control transactions usually driven by software 94, 104, 106, 154–156, 161, 164, 167–169

SLR Super Logic Region 96–100, 107, 169

SLVS Scalable low-voltage signaling 25

SMX STS/MUCH-XYTER 26, 99, 109, 116–118, 120, 132, 137–140, 142, 143, 145–149

SPADIC Self-triggered Pulse Amplification and Digitization ASIC 99, 109, 116, 120, 134, 145, 149, 150

SPI Serial Peripheral Interface 156, 166

SPS Super Proton Synchrotron 10

STAR Solenoidal Tracker at RHIC 10, 13

STS Silicon Tracking System 17, 18, 26, 31, 33, 41, 45, 50, 98, 99, 110, 117, 120, 124, 127–129, 131, 132, 137–141, 145

STS-ROB STS ReadOut Board 138–140

T0 Time Zero 19, 114, 120, 124, 131, 159

TAI Temps Atomique International 115

TCLink time-compensated link 116

TDC time-to-digital converter 156, 161, 166–170

TDR Technical Design Report 17–19, 39, 107, 119

TFC Timing and Fast Control 22, 24, 25, 27, 52, 90, 91, 93–99, 103–105, 113–119, 122, 155, 157, 160, 168–171

time counter counter representing the local time (see 2.2.5) 26–28, 94, 104, 113–116, 137, 138, 145, 148, 155, 157, 161, 162, 170

timestamp property of an object (e. g., hit or container) carrying the time (see 2.2.5) 22, 24, 26–28, 41–43, 50, 52, 68, 89, 91, 94, 113, 133, 137, 145, 147, 148, 153, 155, 162

TLP Transaction Layer Packet 58, 59, 61–63, 70, 100, 101

TMC Timing Mezzanine Card 96

TOF Time-of-Flight System 18, 27, 31, 45, 98, 99, 111, 113, 114, 120, 124, 127–129, 135, 136, 156, 157, 159, 160

TRB TDC Readout Board 170

TrbNet TRB Network 134, 166–169

TRD Transition Radiation Detector 18, 31, 99, 111, 120, 124, 127, 134, 135, 150

TRD-1D outer TRD with SPADIC readout 98, 120, 134, 135, 145, 149, 150

TRD-2D inner TRD with FASP readout 98, 120, 124, 134, 150, 152–154

TSA timeslice archive 124, 127, 129

TSC timeslice component 22, 24, 37, 38, 43, 44, 46–49, 54, 56, 57, 64, 75, 82, 113, 115

uplink connection in FEE to DAQ direction 27, 28, 94, 96, 104, 113–115, 117, 118, 132, 137–140, 143, 145, 146, 150, 154, 156, 164

URAM ultra random access memory 99

UrQMD Ultra-relativistic Quantum Molecular Dynamics 11, 12, 32, 134

UTC Coordinated Universal Time 115

VCO voltage-controlled oscillator 95

VCXO voltage-controlled crystal oscillator 27, 95, 116

Versatile Link radiation-hard optical link 25, 26, 138

VHDL VHSIC Hardware Description Language 107

VTRx Versatile TransReceiver 25, 96, 138, 140, 164

VTTx Versatile Twin-Transmitter 25, 96, 139, 140, 164

WhiteRabbit open source timing and control network 96, 119, 122

Wishbone open source bus architecture 99, 101, 102, 107–109, 145, 146

XML Extensible Markup Language 102

Zeropage Zeropage 101–103, 108

Bibliography

[1] K. Fukushima and T. Hatsuda. “The phase diagram of dense QCD.” In: *Rept. Prog. Phys.* 74 (2011), p. 014001. DOI: 10.1088/0034-4885/74/1/014001. arXiv: 1005.4814 [hep-ph] (cit. on p. 9).

[2] A. Bazavov et al. “Chiral crossover in QCD at zero and non-zero chemical potentials.” In: *Phys. Lett. B* 795 (2019), pp. 15–21. DOI: 10.1016/j.physletb.2019.05.013. arXiv: 1812.08235 [hep-lat] (cit. on p. 10).

[3] Wei-jie Fu, Jan M. Pawlowski, and Fabian Rennecke. “QCD phase structure at finite temperature and density.” In: *Phys. Rev. D* 101.5 (2020), p. 054032. DOI: 10.1103/PhysRevD.101.054032. arXiv: 1909.02991 [hep-ph] (cit. on p. 10).

[4] A. Andronic et al. “Hadron production in ultra-relativistic nuclear collisions: quarkyonic matter and a triple point in the phase diagram of QCD.” In: *Nucl. Phys.* A837 (2010), pp. 65–86. DOI: 10.1016/j.nuclphysa.2010.02.005. arXiv: 0911.4806 [hep-ph] (cit. on p. 10).

[5] L. Adamczyk et al. “Bulk Properties of the Medium Produced in Relativistic Heavy-Ion Collisions from the Beam Energy Scan Program.” In: *Phys. Rev. C* 96.4 (2017), p. 044904. DOI: 10.1103/PhysRevC.96.044904. arXiv: 1701.07065 [nucl-ex] (cit. on p. 10).

[6] M. S. Abdallah et al. “Measurements of Proton High Order Cumulants in $\sqrt{s_{\mathrm{NN}}} =$ 3 GeV Au+Au Collisions and Implications for the QCD Critical Point.” In: *Phys. Rev. Lett.* 128.20 (2022), p. 202303. DOI: 10.1103/PhysRevLett.128.202303. arXiv: 2112.00240 [nucl-ex] (cit. on p. 10).

[7] Marek Gazdzicki. “Ion program of NA61/SHINE at the CERN SPS.” In: *J. Phys. G* 36.6 (May 2009), p. 064039. DOI: 10.1088/0954-3899/36/6/064039 (cit. on p. 10).

[8] András László. “The NA61/SHINE Experiment at the CERN SPS.” In: *Nucl. Phys. A* 830.1 (2009). Quark Matter 2009, pp. 559c–562c. ISSN: 0375-9474. DOI: 10.1016/j.nuclphysa.2009.09.047 (cit. on p. 10).

[9] D. Blaschke et al. “Topical issue on exploring strongly interacting matter at high densities - NICA white paper.” In: *Eur. Phys. J. A* 52 (2016), p. 267. ISSN: 1434-601X. DOI: 10.1140/epja/i2016-16267-x (cit. on p. 10).

[10] I. C. Arsene et al. “Dynamical phase trajectories for relativistic nuclear collisions.” In: *Phys. Rev. C* 75 (2007), p. 034902. DOI: 10.1103/PhysRevC.75.034902. arXiv: nucl-th/0609042 (cit. on p. 11).

[11] S. A. Bass et al. "Microscopic models for ultrarelativistic heavy ion collisions." In: *Prog. Part. Nucl. Phys.* 41 (1998), pp. 255–369. DOI: 10.1016/S0146-6410(98)00058-1. arXiv: nucl-th/9803035 [nucl-th] (cit. on pp. 11, 12).

[12] A. Andronic et al. "Hadron production in central nucleus-nucleus collisions at chemical freeze-out." In: *Nucl. Phys.* A772 (2006), pp. 167–199. DOI: 10.1016/j.nuclphysa.2006.03.012. arXiv: nucl-th/0511071 [nucl-th] (cit. on pp. 12, 13).

[13] Tetyana Galatyuk. "Future facilities for high μ_B physics." In: *Nucl. Phys. A* 982 (2019), pp. 163–169. ISSN: 0375-9474. DOI: https://doi.org/10.1016/j.nuclphysa.2018.11.025 (cit. on p. 13).

[14] FAIR Project. *FAIR Operation Modes – Reference Modes for the Modularized Start Version*. EDMS-2374493 (requires CERN login). Sept. 2020. URL: https://edms.cern.ch/document/2374493 (cit. on pp. 12, 15, 30, 117).

[15] P. Senger, V. Friese, et al. *Nuclear matter physics at SIS-100*. CBM Report 2012-01. 2011 (cit. on p. 15).

[16] B. Friman et al., eds. *The CBM physics book: Compressed baryonic matter in laboratory experiments*. Vol. 814. Lecture Notes in Physics. 2011, 980 p. ISBN: 978-3-64213-292-6. DOI: 10.1007/978-3-642-13293-3 (cit. on p. 15).

[17] *Convention concerning the Construction and Operation of a Facility for Antiproton and Ion Research in Europe (FAIR)*. Bundesgesetzblatt 2014 II p. 42. Oct. 2010. URL: https://www.auswaertiges-amt.de/en/aussenpolitik/themen/-/248658 (cit. on p. 15).

[18] M. Durante et al. "All the fun of the FAIR: fundamental physics at the facility for antiproton and ion research." In: *Physica Scripta* 94.3 (Jan. 2019), p. 033001. DOI: 10.1088/1402-4896/aaf93f (cit. on p. 15).

[19] C. Sturm and H. Stocker. "The facility for antiproton and ion research FAIR." In: *Phys. Part. Nucl. Lett.* 8.8 (Dec. 2011), pp. 865–868. DOI: 10.1134/S1547477111080140 (cit. on p. 15).

[20] Alexander Malakhov and Alexey Shabunov, eds. *Technical Design Report for the CBM Superconducting Dipole Magnet*. CBM Technical Design Reports. GSI, Oct. 2013, 80 p. URL: https://repository.gsi.de/record/109025 (cit. on p. 17).

[21] M. Deveaux et al., eds. *Technical Design Report for the CBM Micro Vertex Detector*. CBM Technical Design Reports. GSI, May 2022, 157 p. URL: https://repository.gsi.de/record/246516 (cit. on pp. 17, 163).

[22] Johann Heuser et al., eds. *[GSI Report 2013-4] Technical Design Report for the CBM Silicon Tracking System (STS)*. CBM Technical Design Reports. GSI, Oct. 2013, 167 p. URL: https://repository.gsi.de/record/54798 (cit. on pp. 17, 41).

[23] Claudia Höhne, ed. *Technical Design Report for the CBM Ring Imaging Cherenkov Detector*. CBM Technical Design Reports. GSI, June 2013, 215 p. URL: https://repository.gsi.de/record/65526 (cit. on p. 17).

[24] Subhasis Chattopadhyay et al., eds. *Technical Design Report for the CBM : Muon Chambers (MuCh)*. CBM Technical Design Reports. Darmstadt: GSI, Nov. 2015, 190 p. URL: https://repository.gsi.de/record/161297 (cit. on p. 18).

[25] Christoph Blume, C. Bergmann, and D. Emschermann, eds. *Technical Design Report for the CBM Transition Radiation Detector (TRD)*. CBM Technical Design Reports. FAIR, Oct. 2018, 165 p. DOI: 10.15120/GSI-2018-01091 (cit. on p. 18).

[26] Norbert Herrmann, ed. *Technical Design Report for the CBM Time-of-Flight System (TOF)*. CBM Technical Design Reports. GSI, Oct. 2014, 182 p. URL: https://repository.gsi.de/record/109024 (cit. on p. 18).

[27] Fedor Guber and Ilya Selyuzhenkov, eds. *Technical Design Report for the CBM Projectile Spectator Detector (PSD)*. CBM Technical Design Reports. GSI, July 2015, 75 S. URL: https://repository.gsi.de/record/109059 (cit. on p. 19).

[28] The CBM Collaboration. *mCBM@SIS18 - A CBM full system test-setup for high-rate nucleus-nucleus collisions at GSI/FAIR*. Beamtime Application. June 2017. DOI: 10.15120/GSI-2019-00977 (cit. on pp. 19, 29, 119).

[29] C. Sturm and N. Herrmann. "Achievements of the mCBM beam campaign 2021." In: *CBM Progress Report 2021*. GSI Helmholtzzentrum für Schwerionenforschung, 2022, pp. 224–230. ISBN: 978-3-9822127-0-8. DOI: 10.15120/GSI-2022-00599 (cit. on pp. 19, 29, 78).

[30] *HEPiX - Benchmarking Working Group*. URL: https://w3.hepix.org/benchmarking.html (cit. on p. 22).

[31] P. Moreira et al. "The GBT Project." In: *Proceedings, Topical Workshop on Electronics for Particle Physics (TWEPP09)*. CERN, 2009, pp. 342–346. DOI: 10.5170/CERN-2009-006.342 (cit. on pp. 25, 104, 138).

[32] P. Moreira, J. Christiansen, and K. Wyllie. *GBTx Manual*. 2021. URL: https://cds.cern.ch/record/2809057 (cit. on pp. 25, 93).

[33] S Bonacini, K Kloukinas, and P Moreira. "E-link: A Radiation-Hard Low-Power Electrical Link for Chip-to-Chip Communication." In: *Proceedings, Topical Workshop on Electronics for Particle Physics (TWEPP09)*. CERN, 2009, pp. 422–425. DOI: 10.5170/CERN-2009-006.422 (cit. on p. 25).

[34] L Amaral et al. "The versatile link, a common project for super-LHC." In: *Journal of Instrumentation* 4.12 (Dec. 2009), P12003–P12003. DOI: 10.1088/1748-0221/4/12/p12003 (cit. on pp. 25, 96).

[35] S. Baron et al. "Implementing the GBT data transmission protocol in FPGAs." In: *Proceedings of the Topical Workshop on Electronics for Particle Physics, TWEPP 2009*. 2009, pp. 631–635. DOI: 10.5170/CERN-2009-006.631 (cit. on pp. 26, 96, 104).

[36] FAIR Project. *General Specification for the FAIR Accelerator Facility Project*. FAIR Document. EDMS Id: 1365092 v.3. FAIR, Sept. 2020. URL: https://edms.cern.ch/document/1365092/3 (cit. on p. 29).

[37] Ivan Kisel and the CBM collaboration. “Event Topology Reconstruction in the CBM Experiment.” In: *Journal of Physics: Conference Series* 1070 (Aug. 2018), p. 012015. DOI: 10.1088/1742-6596/1070/1/012015 (cit. on p. 36).

[38] Qiyan Li et al. “Online data preprocessing for the CBM Micro Vertex Detector.” In: *CBM Progress Report 2016*. Darmstadt: GSI, 2017, p. 14. ISBN: 978-3-9815227-4-7. URL: https://repository.gsi.de/record/201318 (cit. on pp. 39, 165).

[39] Pierre-Alain Loizeau. *Messages data format*. Internal Note. July 2019 (cit. on p. 41).

[40] Tim Armbruster. “SPADIC - a Self-Triggered Detector Readout ASIC with Multi-Channel Amplification and Digitization.” PhD thesis. Ruperto-Carola University of Heidelberg, 2013. DOI: 10.11588/heidok.00014981 (cit. on pp. 41, 149).

[41] Pierre-Alain Loizeau. “Development and test of a free-streaming readout chain for the CBM Time of Flight Wall.” PhD thesis. Ruperto-Carola University of Heidelberg, 2014. DOI: 10.11588/heidok.00017081 (cit. on p. 41).

[42] Volker Friese. personal communication. June 17, 2016 (cit. on p. 45).

[43] Walter F. J. Müller. personal communication. 2018 (cit. on p. 50).

[44] Dirk Hutter. “An Input Interface for the CBM First-level Event Selector.” PhD thesis. Johann Wolfgang Goethe University, Frankfurt am Main, 2020. URL: https://nbn-resolving.org/urn:nbn:de:hebis:30:3-591599 (cit. on pp. 53, 82).

[45] Dirk Hutter and Jan de Cuveland. *The FLES Detector Input Interface*. Technical Note. Feb. 2016 (cit. on p. 56).

[46] *AMBA 4 AXI4-Stream Protocol Specification 1.0*. ARM. Mar. 2010. URL: https://developer.arm.com/documentation/ihi0051/a/ (visited on 02/13/2022) (cit. on p. 57).

[47] David S. Miller, Richard Henderson, and Jakub Jelinek. *Dynamic DMA mapping Guide*. URL: https://www.kernel.org/doc/Documentation/DMA-API-HOWTO.txt (cit. on pp. 60, 65).

[48] Corbet. *Driver porting: Network drivers*. LWN article. Apr. 2003. URL: https://lwn.net/Articles/30107/ (visited on 2003) (cit. on p. 62).

[49] *PCI Express Base Specification Revision 2.1*. PCI-SIG. Mar. 2009. URL: https://pcisig.com/specifications (cit. on pp. 63, 100).

[50] Dominic Eschweiler. “Efficient Device Drivers for Supercomputers.” PhD thesis. Johann Wolfgang Goethe University, Frankfurt am Main, 2015. URL: https://ubffm.hds.hebis.de/Record/HEB396120016 (cit. on p. 65).

[51] Jan de Cuveland and Dirk Hutter et al. *CBM FLES Timeslice Building*. Git Repository. URL: https://github.com/cbm-fles/flesnet (cit. on p. 73).

[52] *ZeroMQ - An open-source universal messaging library*. URL: https://zeromq.org/ (cit. on pp. 76, 107).

[53] Farouk Salem et al. "Scheduling data streams for low latency and high throughput on a Cray XC40 using Libfabric." In: *Concurr. Comput. Pract. Exp.* 32.20 (2020). DOI: 10.1002/cpe.5563. URL: https://doi.org/10.1002/cpe.5563 (cit. on p. 79).

[54] Farouk Salem and Florian Schintke. "Large-Scale Performance of the Data-Flow Scheduler (DFS) and FLESnet." In: *CBM Progress Report 2021*. Darmstadt: GSI Helmholtzzentrum für Schwerionenforschung, 2022, pp. 170–171. ISBN: 978-3-9822127-0-8. DOI: 10.15120/GSI-2022-00599 (cit. on p. 80).

[55] D. Hutter, J. de Cuveland, and V. Lindenstruth. "Preparations for the mCBM FLES Setup." In: *CBM Progress Report 2017*. Darmstadt: GSI Helmholtzzentrum für Schwerionenforschung, 2018, p. 185. ISBN: 978-3-9815227-5-4. DOI: 10.15120/GSI-2018-00485 (cit. on p. 85).

[56] M. Gumiński et al. "Time and clock synchronization with AFCK for CBM." In: *Proceedings, XXXVI Symposium on Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2015, Wilga, Poland*. SPIE, 2015, p. 96622V. DOI: 10.1117/12.2205798 (cit. on p. 95).

[57] J.P. Cachemiche et al. "The PCIe-based readout system for the LHCb experiment." In: *Journal of Instrumentation* 11.02 (Feb. 2016), P02013–P02013. DOI: 10.1088/1748-0221/11/02/p02013 (cit. on p. 95).

[58] J.P. Cachemiche. *PCIe40: A Common Readout Board for LHCb and ALICE*. Presentation on ACES 2018 workshop. 2018. URL: https://indico.cern.ch/event/681247/contributions/2929079/ (cit. on p. 95).

[59] Kai Chen et al. "A Generic High Bandwidth Data Acquisition Card for Physics Experiments." In: *IEEE Transactions on Instrumentation and Measurement* 69.7 (2020), pp. 4569–4577. DOI: 10.1109/TIM.2019.2947972 (cit. on pp. 95, 96).

[60] Alexander Paramonov. "FELIX: the Detector Interface for the ATLAS Experiment at CERN." In: *EPJ Web Conf.* 251 (2021), p. 04006. DOI: 10.1051/epjconf/202125104006 (cit. on pp. 95, 96).

[61] Sophie Baron, Eduardo Mendes, and Julian Mendez. *GBT-FPGA – Firmware dedicated to the communication with the GBTx ASIC*. Git Repository. URL: https://gitlab.cern.ch/gbt-fpga/gbt-fpga (cit. on pp. 96, 104).

[62] C. Soós et al. "The Versatile Transceiver: towards production readiness." In: *Journal of Instrumentation* 8.03 (Mar. 2013), pp. C03004–C03004. DOI: 10.1088/1748-0221/8/03/C03004 (cit. on pp. 96, 138).

[63] A. Caratelli et al. "The GBT-SCA, a radiation tolerant ASIC for detector control and monitoring applications in HEP experiments." In: *Journal of Instrumentation* 10.3 (Mar. 2015), pp. C03034–C03034. DOI: 10.1088/1748-0221/10/03/C03034 (cit. on p. 96).

[64] Silicon Labs. *AN699: FPGA Reference Clock Phase Jitter Specifications*. URL: https://www.skyworksinc.com/-/media/SkyWorks/SL/documents/login/application-notes/AN699.pdf (cit. on p. 97).

[65] Wojciech M. Zabołotny et al. "Control and Diagnostics System Generator for Complex FPGA-Based Measurement Systems." In: *Sensors* 21.21 (Nov. 2021), p. 7378. DOI: 10.3390/s21217378 (cit. on p. 102).

[66] Sophie Baron, Eduardo Mendes, and Julian Mendez. *GBT-SC for FPGA: VHDL module for GBTx and SCA configuration through GBT links.* Git Repository. URL: https://gitlab.cern.ch/gbtsc-fpga-support/gbt-sc (cit. on p. 104).

[67] *MessagePack - An efficient binary serialization format.* URL: https://msgpack.org/ (cit. on p. 107).

[68] E. Brandao De Souza Mendes, S. Baron, and M. Taylor. "TCLink: A Timing Compensated High-Speed Optical Link for the HL-LHC experiments." In: *Proceedings of the Topical Workshop on Electronics for Particle Physics, TWEPP 2019.* 2019. DOI: 10.22323/1.370.0057 (cit. on p. 116).

[69] P. Moreira. *The lpGBT: a radiation tolerant ASIC for Data, Timing, Trigger and Control Applications in HL-LHC.* Presentation at TWEPP-2019. URL: https://indico.cern.ch/event/799025/contributions/3486153 (cit. on p. 116).

[70] *LpGBT-FPGA – Firmware.* Project home page. URL: http://lpgbt-fpga.web.cern.ch/doc/html/ (cit. on p. 116).

[71] R. Singh et al. "Slow Extraction Spill Characterization From Micro to Milli-Second Scale." In: *Journal of Physics: Conference Series* 1067 (Sept. 2018), p. 072002. DOI: 10.1088/1742-6596/1067/7/072002 (cit. on p. 117).

[72] A. Rost et al. "Performance of the CVD Diamond Based Beam Quality Monitoring System in the HADES Experiment at GSI." In: *Proc. 10th International Particle Accelerator Conference (IPAC'19), Melbourne, Australia, 19-24 May 2019.* 2019, pp. 2507–2509. DOI: 10.18429/JACoW-IPAC2019-WEPGW019 (cit. on pp. 117, 159).

[73] K. Kasinski, R. Szczygiel, and W. Zabolotny. "Back-end and interface implementation of the STS-XYTER2 prototype ASIC for the CBM experiment." In: *Journal of Instrumentation* 11.11 (Nov. 2016), pp. C11018–C11018. DOI: 10.1088/1748-0221/11/11/c11018 (cit. on p. 117).

[74] Xin Gao et al. "Throttling strategies and optimization of the trigger-less streaming DAQ system in the CBM experiment." In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 978 (2020), p. 164442. DOI: 10.1016/j.nima.2020.164442 (cit. on p. 117).

[75] J. Michel and Qiyan Li. *MVD: Hit Rates, Link Bandwidth and Interaction Rates.* CBM Technical Note TN-19002. Mar. 2019. URL: https://indico.gsi.de/event/8579/contributions/37302/ (cit. on pp. 131, 163).

[76] M. Baznat et al. *Monte-Carlo generator of heavy ion collisions DCM-SMM.* 2019. arXiv: 1912.09277 [nucl-th] (cit. on p. 135).

[77] Krzysztof Kasinski et al. “STS-XYTER, a high count-rate self-triggering silicon strip detector readout IC for high resolution time and energy measurements.” In: *2014 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*. 2014, pp. 1–6. DOI: 10.1109/NSSMIC.2014.7431048 (cit. on p. 137).

[78] K. Kasinski et al. “Characterization of the STS/MUCH-XYTER2, a 128-channel time and amplitude measurement IC for gas and silicon microstrip sensors.” In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 908 (2018), pp. 225–235. DOI: 10.1016/j.nima.2018.08.076 (cit. on p. 137).

[79] K. Kasinski, R. Kleczek, and R. Szczygiel. “Front-end readout electronics considerations for Silicon Tracking System and Muon Chamber.” In: *Journal of Instrumentation* 11.2 (Feb. 2016), pp. C02024–C02024. DOI: 10.1088/1748-0221/11/02/c02024 (cit. on p. 137).

[80] K. Kasinski et al. “A protocol for hit and control synchronous transfer for the front-end electronics at the CBM experiment.” In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 835 (2016), pp. 66–73. DOI: 10.1016/j.nima.2016.08.005 (cit. on pp. 138, 143–145).

[81] K. Kasinski et al. *SMX2 and SMX2.1 Manual v4.00*. 2021 (cit. on p. 145).

[82] Tim Armbruster et al. “Multi-channel charge pulse amplification, digitization and processing ASIC for detector applications.” In: *2012 IEEE Nuclear Science Symposium and Medical Imaging Conference Record (NSS/MIC)*. 2012, pp. 697–702. DOI: 10.1109/NSSMIC.2012.6551195 (cit. on p. 149).

[83] M. Ciobanu et al. “New Models of PADI, an Ultrafast Preamplifier–Discriminator ASIC for Time-of-Flight Measurements.” In: *IEEE Transactions on Nuclear Science* 68.6 (2021), pp. 1325–1333. DOI: 10.1109/TNS.2021.3073487 (cit. on p. 156).

[84] Harald Deppe and Holger Flemming. “The GSI event-driven TDC with 4 channels GET4.” In: *2009 IEEE Nuclear Science Symposium Conference Record (NSS/MIC)*. 2009, pp. 295–298. DOI: 10.1109/NSSMIC.2009.5401741 (cit. on p. 156).

[85] Harald Deppe and Holger Flemming. *The GSI event-driven TDC with 4 channels GET4 - Manual for Version 1.23, 1.30 and 2.00*. 2019. URL: https://wiki.gsi.de/pub/EE/GeT4/get4.pdf (cit. on p. 156).

[86] M. Deveaux et al. “Observations on MIMOSIS-0, the first dedicated CPS prototype for the CBM MVD.” In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 958 (2022), p. 162653. DOI: 10.1016/j.nima.2019.162653 (cit. on p. 163).

[87] P. Sitzmann et al. “Estimating the required data bandwidth of the CBM-MVD pixel sensors.” In: *CBM Progress Report 2016*. Darmstadt: GSI, 2017, p. 17. ISBN: 978-3-9815227-4-7. URL: https://repository.gsi.de/record/201318 (cit. on p. 164).

[88] Adrian Amatus Weber. “Development of readout electronics for the RICH detector in the HADES and CBM experiments - HADES RICH upgrade, mRICH detector construction and analysis.” PhD thesis. Justus-Liebig-Universität Gießen, 2021. DOI: 10.22029/jlupub-288 (cit. on p. 169).

[89] J. Michel et al. “The upgraded HADES trigger and data acquisition system.” In: *Journal of Instrumentation* 6.12 (Dec. 2011), pp. C12056–C12056. DOI: 10.1088/1748-0221/6/12/c12056 (cit. on p. 169).

[90] Jan Michel. “Development and implementation of a new Trigger and data Acquisition system for the HADES detector.” link to full text. PhD thesis. Johann Wolfgang Goethe-Universität Frankfurt, 2012. URL: https://ubffm.hds.hebis.de/Record/HEB314568271 (cit. on p. 169).

[91] D. Finogeev et al. “The first mPSD beam test results at mCBM in 2020.” In: *CBM Progress Report 2020*. Darmstadt: GSI Helmholtzzentrum für Schwerionenforschung, 2021, p. 128. ISBN: 978-3-9815227-9-2. DOI: 10.15120/GSI-2021-00421 (cit. on p. 171).

[92] D. Finogeev et al. “Readout system of the CBM Projectile Spectator Detector at FAIR.” In: *JINST* 15 (2020), p. C09015. DOI: 10.1088/1748-0221/15/09/C09015 (cit. on p. 171).

[93] D. Finogeev et al. “Development of readout chain for CBM Projectile Spectator Detector at FAIR.” In: *J. Phys. Conf. Ser.* 1690 (2020), p. 012059. DOI: 10.1088/1742-6596/1690/1/012059 (cit. on p. 171).