Master Thesis

Firmware Implementations for the Upgrade of the Data Acquisition System of the Belle II Pixel Detector

Firmwareimplementierungen für das Upgrade des Datenerfassungssystems des Belle II Pixeldetektors

Author: Matthäus Krein

December 13, 2021

Supervisors: Dr. Thomas Geßler and Simon Reiter
Referees: Apl. Prof. Dr. Sören Lange and Prof. Dr. Claudia Höhne

Faculty 07

II. Physics Institute of Justs-Liebig-Universität Gießen

Abstract

The Belle II experiment is a high-precision particle physics experiment located at the research facility high energy accelerator research organization (jap. Kō Enerugī Kasokuki Kenkyū Kikō) (KEK) in Tsukuba, Japan. The experiment contains an asymmetric ring accelerator, where electrons and positrons will collide. Around the interaction point subdetectors are placed that together are referred to as the Belle II detector. Each subdetector exploits well known interactions of particles with matter, which enables the identification of those particles. The Belle II experiments studies subatomic physics to extend the Standard Model or find new interactions, which may create a completely new branch of physics.

The Online Selection Node (ONSEN) system is part of the data acquisition for the innermost subdetector, the Pixel Detector (PXD). At peak luminosity, the PXD will create a lot of data, which must be reduced. Therefore the ONSEN system selects data of specific regions of the PXD depending on track extrapolation that are created by other subdetectors.

The hardware of the ONSEN system is based on technology stemming from communication industry to handle large amounts of data and the firmware was specially developed for the usage of PXD data acquisition. The Compute Node Carrier Board v4.0 (CNCB v4.0) is a new revision and is planed to be used as a spare Carrier Board. The upgrades of the components of the board lead to necessary adjustments to the firmware.

The firmware is based on the previous structure of the Intellectual Property (IP) cores, that handle specific data processing tasks and monitoring. The introduction of a new architecture leads to rewriting of the IP cores. The CNCB v4.0 is only one part of the hardware used in the ONSEN system, therefore only relevant components are documented in this thesis. Moreover, in the progress of development the monitoring of the ONSEN system was modified. The goal is to use the CNCB v4.0 as a spare Carrier Board in the Belle II experiment in Japan. The usage of the CNCB v4.0 was tested and stable operation was verified.

Zusammenfassung

Das Belle II Experiment ist ein hochpräzision Teilchenexperiment und wird durchgeführt an dem Hochenergie-Beschleuniger-Forschungsorganisation (jap. Kō Enerugī Kasokuki Kenkyū Kikō) (KEK) in Tsukuba, Japan. Das Experiment besteht aus einem asymmetrischen Ringbeschleuniger in dem Elektronen und Positronen kollidieren. Um den Kollisionspunkt befinden sich Subdetektoren, die zusammen den Belle II Detektor ergeben. Die Subdetektoren nutzen die Wechselwirkung von Teilchen mit Materie aus, um Informationen über die Teilchen zu erhalten. Das Belle II Experiment untersucht die subatomare Physik, um das Standard Model zu erweitern oder neue Physik zu finden.

Das Online Selection Node (ONSEN) System ist Teil der Datenauslese des innersten Subdetektor, dem Pixeldetektor (PXD). Wenn die Ziellumi-nosität erreicht ist, werden viele Daten produziert, die reduziert werden müssen. Dazu wird das ONSEN System verwendet, um Daten aus bestimmten Regionen des PXD auszuwählen und weiterzuleiten abhängig von der Trajektorieextrapolation, die aus den Daten anderer Subdetektoren erstellt wurden.

Die Hardware des ONSEN Systems basiert auf Technologie aus der Kommunikationsindustrie, um große Mengen an Daten zu verarbeiten und die Firmware wurde speziell für Anwendung zur PXD Datenauslese entwickelt. Das Compute Node Carrier Board v4.0 (CNCB v4.0) ist eine neue Revesion und soll als Ersatz-PCB verwendet werden. Die Upgrades der Komponenten von dem PCB führen zu notwendigen Anpassungen der Firmware.

Die Firmware basiert auf der früheren Struktur der Intellectual Property (IP) Cores, die anwendungsspezifische Aufgaben durchführen. Jedoch müssen die IP Cores umgeschrieben werden, da eine neue Architektur eingeführt wurde. Das CNCB v4.0 is nur ein Teil der Hardware, die im ganzen ONSEN System genuzt wird, weshalb nur relevante Komponenten in dieser Arbeit dokumentiert werden. Zusätzlich wurde im Laufe des Fortschritts das Überwachungssystem modifiziert. Das Ziel dieser Arbeit ist den CNCB v4.0 als Ersatz-PCB fertig zu stellen, um ihn im Belle II Experiment verwenden zu können. Dies wurde getestet und die Funktion des CNCB v4.0 konnte bestätigt werden.

Contents

1 Theoretical Background

In this chapter, the basics of particle physics are outlined. This includes the Standard Model that describes the fundamental particles and their interactions. Furthermore, insights in CP violation are presented that are studied with the decay of B mesons, which are produced in a large number at the Belle II experiment.

1.1 The Standard Model and Fundamental Forces

The Standard Model (SM) describes the properties of fundamental particles. It is part of a unifying theory, which contain three fundamental forces. The electromagnetic force that defines the interaction of light and the binding of electrons to the nucleus, the weak force, which is observed in radioactive decays and the strong force that is dominant at short ranges and explains the binding of quarks. The gravitational force is not part of the SM, but efforts to extend the SM with a new boson exists [1].

The SM is divided into fermions, particles with half integer spin, and bosons that have a integer spin. Quarks are fermions that come in three families. They are distinguished by mass and electromagnetic charge. The first generation contains the up and down quark, the second generation the charm and strange quark and in the third generation are the top and bottom quark.
The other fermions in the SM are the leptons. Similar to the quarks, they also come in three families. There are the electrons, muons and taus with the electrons being the lightest particle and the tau the heaviest.

Correspondingly, three types of neutrinos exist that are named electron neutrino, muon neutrino and tau neutrino depending if it interacts via charged currents with electrons, muons or taus.

Moreover, fermions can interact via a force with other particles, if it contains a charge of the corresponding force. The properties of the fermions are summarized in the table 1.1. Each fermion comes with an anti-particle that carry the opposite charges. The electromagnetic force interacts with the electromagnetic charge, the strong charge is called color with three possible charges (red, green and blue) and anti-charges (anti-red, anti-green and anti-blue). The weak force uses the weak isospin as charge. Right handed neutrinos do not have a weak isospin and can not interact via the weak force.

Table 1.1 – Properties of fermions [2].

Fermion	Mass	EM charge [e]	Strong charge	Weak isospin
Quarks				
Up	2.16 MeV	2/3	r/g/b	-1/2
Charm	1.27 GeV	2/3	r/g/b	-1/2
Top	173 GeV	2/3	r/g/b	-1/2
Down	4.67 MeV	-1/3	r/g/b	1/2
Strange	93 MeV	-1/3	r/g/b	1/2
Bottom	4.18 GeV	-1/3	r/g/b	1/2
Leptons				
Electron	0.51 MeV	-1	0	-1/2
Muon	105.66 MeV	-1	0	-1/2
Tau	1.78 GeV	-1	0	-1/2
ν_e	0	0	0	1/2
ν_μ	0	0	0	1/2
ν_τ	0	0	0	1/2

The mass of the neutrino is zero according to the SM. It is known that the neutrino must have a mass, because of phenomena called neutrino oscillation.

The basic fundamental interaction is transferred by gauge bosons. The gauge boson of the electromagnetic force is the photon. The photon is a massless particle and has therefore a infinite interaction range. Particle interactions are described with the Quantum Field Theory. In case of electromagnetic interactions, the theory is named Quantum Electro Dynamics (QED). Particles are asserted as quantum fields that are extracted by quantising the Dirac field. Furthermore the interaction are fully described in the Lagrangian density composed of the free Dirac field, the free photon field and the electromagnetic-photon interaction.

The gauge boson of the strong force is the gluon. It is also massless, but carries strong charge. Because of this, a phenomenon called confinement takes place, where color charge on macroscopic scales is not observed. Therefore, strong interacting particles are always in a bound state. Most known structures are mesons, quark and anti-quark bindings, and baryons, three quarks or thee anti-quarks bindings. For other combination like tetraquarks, pentaquarks or glueballs candidates exist and those are currently debated. The Lagrangian density of the Quantum Chromo Dynamics (QCD) consists of terms for a quark propagator, quark-gluon vertices and gluon propagator that also describes 3-gluon-gluon vertices and 4-gluon-gluon vertices.

The weak force has a neutral boson Z with a mass of 91.2 GeV and electromagnetic charged boson $W^{\pm}$ with a mass of 80.4 GeV. The mass of the boson shortens the interaction range and weakens the interaction rate. The weak interaction breaks symmetries in contrast to the other fundamental forces that result in the loss of conservation rules. This enables the change of quark flavour, which is not possible with other forces.

The last particle of the SM is the Higgs boson. It has a mass of 125 GeV, no electromagnetic charge and a spin of 0. It is responsible for the mass of particles, since it mediates between the Higgs field, a quantization of the Klein-Gordan field, and massive particles [3] [4] [5].

1.2 CP-Violation

The CP violation is a violation of the charge conjugation operator C and the parity operator P. The parity operator handles an inversion of a spatial coordinate. In a general sense, it performs a transformation into its mirror image. The parity was thought to be conserved until the Wu experiment [6] discovered that the weak interaction breaks the parity symmetry.
The charge conjugation transforms a particle into its antiparticle. Only pairs of particle-antiparticles states like π^0 or J/ψ are eigenstates of the operator. The charge conjugation is also violated and the proof was the observation of $\pi^0 \to \gamma\gamma\gamma$ decays.
Since the combination of CP is conserved in the Wu experiment, it was thought this symmetry was conserved. The Cronin-Fitch experiment [7] showed that even the CP symmetry is broken, by observing a mixing of the decay of short living K^0_{S} and long living K^0_{L} mesons [8] [9].

The CP violation is explained by introducing mixing of quark flavours. The mixing of the type of quarks is convention and can be chosen between the u c t quarks or the d s b quarks. The theory originated from Nicola Cabibbo, who proposed the mixing of the down and strange quark. The theory was extended by Makoto Kobayashi and Toshihide Maskawa that proposed the mixing of three generations at a time where only three quarks were observed. It introduces a unitary mixing matrix V_{CKM} that mixes the pure down, strange and bottom quark states. An observed down quark d' consists also of contributions of the strange and bottom quark. The contribution depends of the CKM matrix elements. The mixing process is shown in the following equation:

$$\begin{pmatrix} d' \\ s' \\ b' \end{pmatrix} = \begin{pmatrix} V_{ud} & V_{us} & V_{ub} \\ V_{cd} & V_{cs} & V_{cb} \\ V_{td} & V_{ts} & V_{tb} \end{pmatrix} \begin{pmatrix} d \\ s \\ b \end{pmatrix} \tag{1.1}$$

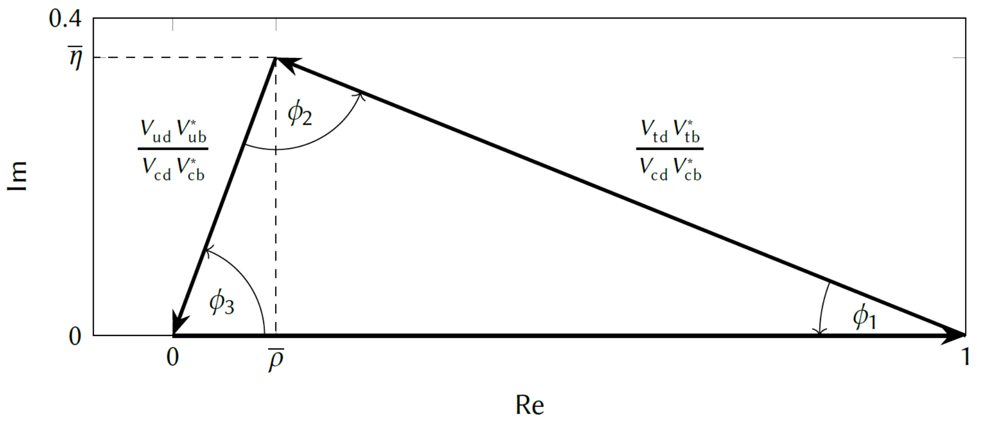

0.4
$\overline{\eta}$
0
Im
$\frac{V_{ud}V_{ub}^*}{V_{cd}V_{cb}^*}$
ϕ_2
ϕ_3
$\frac{V_{td}V_{tb}^*}{V_{cd}V_{cb}^*}$
ϕ_1
0
$\overline{\rho}$
1
Re

Figure 1.1 – Unitary triangle of the CKM matrix [11].

In general, all matrix elements can be complex numbers, leaving us with 18 free parameters. The unitary condition reduces the number to three parameters and six complex phases that can be chosen to one irreducible complex phase $e^{i\delta}$. This complex phase is the origin of the CP violation [10].

There are different representations of the CKM matrix. The Wolfenstein parametrization is as followed.

$$V_{CKM} = \begin{pmatrix} 1-\lambda^2/2 & \lambda & A\lambda^3(\rho - i\eta) \\ -\lambda & 1-\lambda^2/2 & A\lambda^2 \\ A\lambda^3(1-\rho-i\eta) & -A\lambda^2 & 1 \end{pmatrix} + \mathcal{O}(\lambda^4) \quad (1.2)$$

The relations are $\lambda = |V_{us}|$, $A = |V_{cb}|/|V_{us}|^2$, $\eta = \sin\delta|V_{ub}|/|V_{us}||V_{cb}|$ and $\rho = \cos\delta|V_{ub}|/|V_{us}||V_{cb}|$. All parameters are small numbers that indicate that CKM matrix is mostly diagonal. The CP violation happens in order of λ^3 in the elements V_{ub} and V_{dt}. More CP violations in other elements appear in at least an order of λ^4 or higher.
The unitary condition can be expressed as $V_{ud}V_{ub}^* + V_{cd}V_{cb}^* + V_{td}V_{tb}^* = 0$, which is also the condition for a triangle, shown in figure 1.1. By dividing by $V_{cd}V_{cb}^*$, we can fix one side of the triangle between 0 and 1. The angles of the triangle are also used to characterize the CKM matrix. The parameter $\overline{\rho}$ and $\overline{\eta}$ are variants of the Wolfenstein parameters [11] [12].

2 Belle II Experiment

The Belle II experiment is a high-precision particle experiment located in Tsukuba, Japan. It uses the SuperKEKB accelerator, an asymmetrical electron-positron collider and consists of the Belle II particle detector. The center-of-mass energy is $10.58\,\text{GeV}$, which is the resonance of $\Upsilon(4S)$ that mainly decays into B mesons. Therefore, the SuperKEKB is often called B-factory. The goals of the Belle II experiment are the precise measurement of particle decays and parameters of the CKM matrix. There is also the search for beyond Standard Model (BSM) including dark sector physics and magnetic monopoles. For this, it is designed with a high luminosity and precise measurement of interactions. In the following the activities of BSM and dark sector physics at the Belle II experiment are outlined.

The study of B meson decays may yield insights of BSM physics. Here, so-called penguin decays, radiative or electroweak decays of $b \to s$ containing loops, are studied. The measurement of anomalies in the decay may indicate BSM physics. Another BSM investigation is the branching ratios of $B \to \tau\nu$ and $B \to D^*\tau\nu$ that would be different, if charged a Higgs exists or lepton flavour violation of τ meson decays.

The invisible Z' boson extends the SM and is part of dark sector physics. It could explain the g_μ discrepancy and adds an $U(1)'$ gauge group to the SM. The Z' would only couple to μ and τ leptons with the coupling constant g'. A possible production of this particle is the process $e^+e^- \to \mu^+\mu^- Z'$. The search for Z' was carried out by the BaBar experiment, but because of the low statistics no evidence over 3σ was achieved. Axion-like particles (ALPs) are pseudo scalar particles with independent coupling and mass that couple to bosons. Possible processes with photons are photon fusion $e^+e^- \to e^+e^-a$ and ALP-strahlung $e^+e^- \to \gamma a$,

where a is the ALP. Small datasets of ALP analysis were take during early Belle II runs, while reconstructing three photons with total energy equal to the beam energy, assuming the ALPs lifetime is short enough to decay inside the detector.
Similar to the invisible Z' boson there is the search for the dark photon A' that extends the SM and could serve as the interaction particle of the hypothetical dark force. Kinetic mixing with the photons of the SM enables process between SM and dark sector particles. The particles produced by the decay of the dark photon depend on its mass. When the mass of the dark photon is the lightest of the dark sector particles, it decays into SM particles. Otherwise dark photons decay into dark matter that can be searched for with the process $e^+e^- \to \gamma_{ISR}A'$, where γ_{ISR} [13] is a initial state photon [14].

The progress of the Belle II experiment is divided into three phases. In the first phase, form February to June 2016, the particle beams were circulated, but not collided. The accelerator performance was tuned and beam background studies were taken. The second phase was from April to July 2018. First physics data was taken from beam collision with a peak luminosity of $5.55 \times 10^{33}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$. These measurements were preparations for the third phase, which started in March 2019. The goal is to achieve a record luminosity and detect rare events with optimal configuration obtained by the previous phases [15].

2.1 SuperKEKB

The SuperKEKB is an asymmetrical electron positron collider with a center of mass energy of 10.58 GeV. The energy of electron is 7 GeV and of the positron 4 GeV. Therefore, the produced particles are boosted to improve the measurement of fast decaying particles. A schematic of the accelerator is shown in figure 2.1. The SuperKEKB is an upgrade of the KEKB, the previous collider of the Belle experiment. It therefore reuses components like the tunnels and magnets. The electrons are obtained by a photocathode RF electron gun and accelerated with a linear accelerator. The positrons are produced by pair production of bremsstrahlung

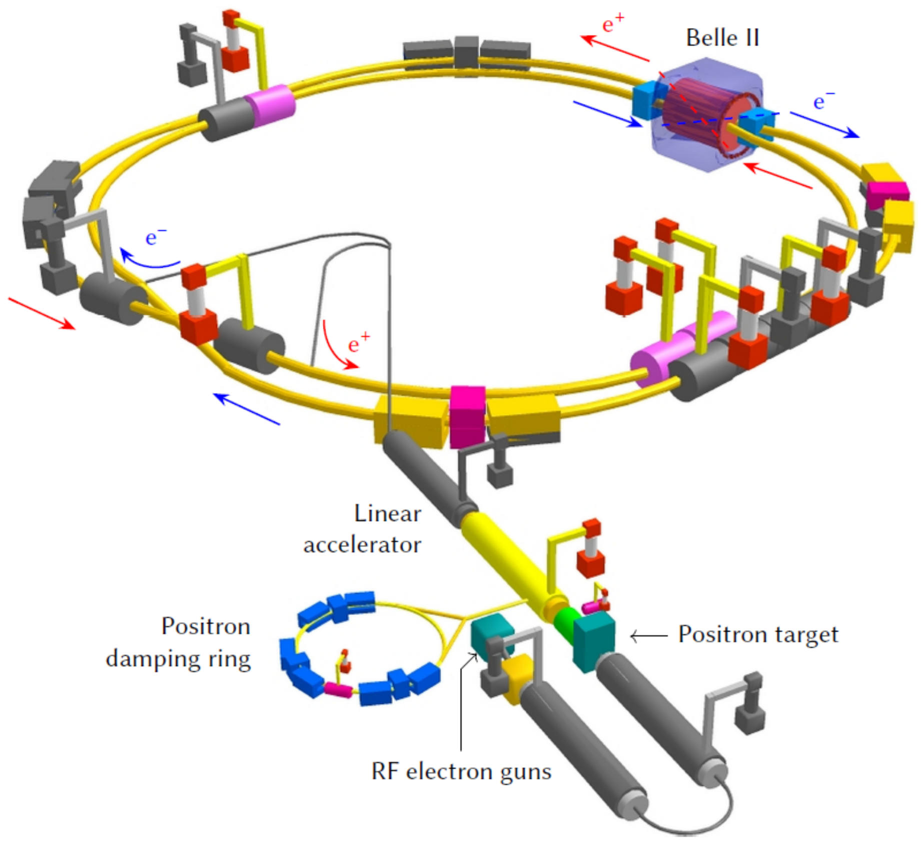


Figure 2.1 – Schematic of the SuperKEKB accelerator. Shown are the electron and positron ring with a circumference of 3016 m, the Belle II detector, linear accelerator and positron dumping ring [11].

photons and further on stored inside a positron damping ring. These photons are produced by firing the RF electron gun at a tungsten target. The SuperKEKB is designed to achieve a luminosity of $8 \times 10^{35}\,\mathrm{cm^{-2}\,s^{-1}}$, promoting it to the frontier of high luminosity accelerators. The luminosity for beams with equal beam size is given by:

$$L = \frac{\gamma_\pm}{2er_\mathrm{e}} \left(\frac{I_\pm \xi_{y\pm}}{\beta^*_{y\pm}}\right) \left(\frac{R_L}{R_{\xi_y}}\right) \tag{2.1}$$

Here, $\gamma_\pm$ is the Lorentz factor, where the index $-$ or $+$ stands for electron or positron, e is the elementary charge, r_e is the classical electron radius, $I_\pm$ is the beam current, $\xi_{y\pm}$ is the beam-beam parameter, $\beta^*_{y\pm}$ is the vertical beta function and R_L and R_{ξ_y} are reduction factors. While the Lorentz factor is limited by the ring circumference and the reduction factors are close to one, the only option is to increase the fraction $\left(\frac{I_\pm \xi_{y\pm}}{\beta^*_{y\pm}}\right)$ by manipulating the beam interaction and increasing the beam current. Instead of a head-on collision of the beam bunch, there is an angle between both bunches that allows high luminosity to be reached [11] [16].

2.2 Belle II Detector

The Belle II detector is an asymmetric barrel shaped particle detector used to observe particles in an area created by an angle ranging from 17° to 150° surrounding the electron beam (boost direction). A cross section of the detector is shown in figure 2.2. The detector contains several subdetectors that are explained in the following, beginning with the innermost subdetector. All the subdetectors except the KLM are placed inside a 1.5 T magnetic field.

The Pixel Vertex Detector (PXD) is the subdetector closest to the interaction point. It is arranged of 40 DEPFET modules to determine the decay vertex of short lived particles. Because of its position, data is dominated by background. Further details of the subdetector and the DEPFET principle are given in the next subsection 2.3, since this thesis is mainly related to the PXD.

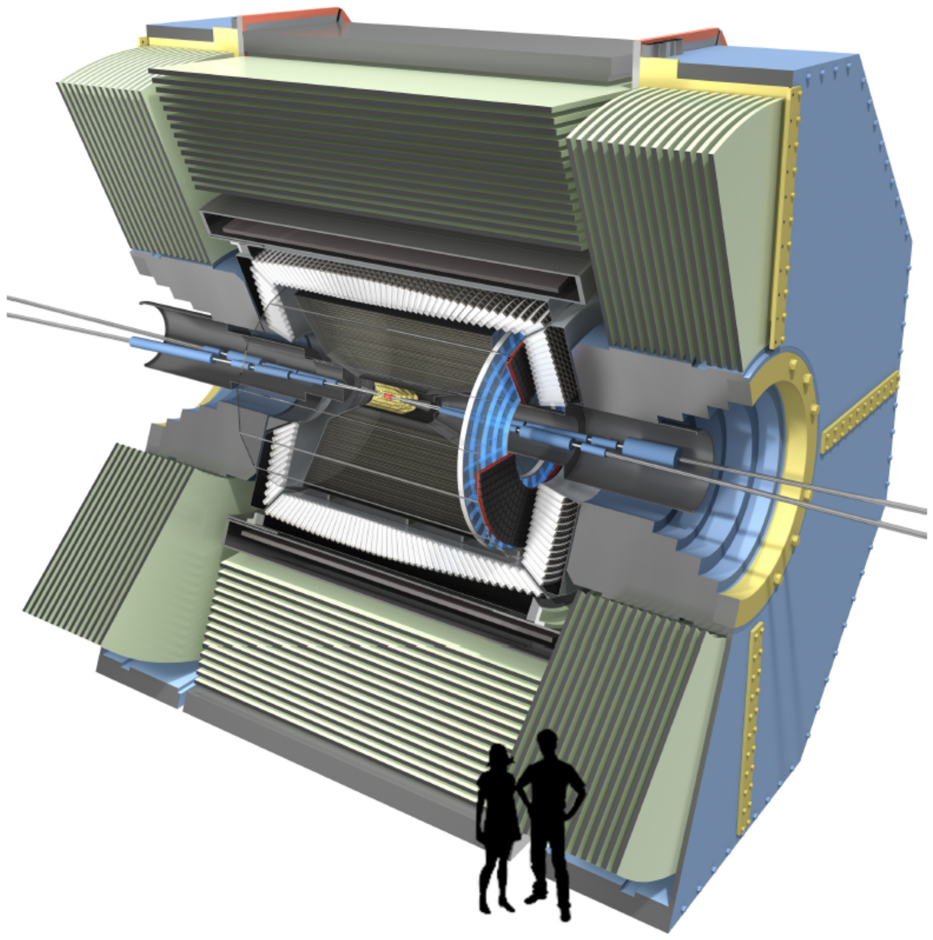


Figure 2.2 – Outline of the Belle II detector. Shown are all subdetectors (PXD in red; SVD in yellow; CDC as grey wires; TOP located at barrel region around the CDC; ARICH as transparent blue disc; ECL as white crystals; KLM as dark sea green sheets) [11]

The Silicon Vertex Detector (SVD) surrounds the PXD. The SVD and PXD together are referred to as the Vertex Detector (VXD). It contains four layers of Double-sided Silicon Strip Detectors (DSSDs). Charged particles pass through the bulk region of the DSSD and produce electron hole pairs that drift to either nearest p- or n-doped strip. This information is used to reconstruct tracks of charged particles.

The Central Drift Chamber (CDC) consists of a large volume filled with a mesh of wires and a gas mixture of helium and ethane. The magnetic filed of the solenoid bends the track of charged particles that ionizes the gas mixture. The electrons drift to the wires, where they are detected. From the time information and electron drifting speed the particle trajectory is reconstructed. Furthermore the particle type can be extracted from the momentum, calculated form the bending of the particle path, and mean energy loss.

The Time of Propagation Counter (TOP) is a type ring-imaging Cherenkov detector (RICH) localized in a layer around the CDC. A cone of Cherenkov radiation is produced by charged particles that moves faster than light through a medium. The opening angle θ is defined by the velocity β and the medium reflection index n with the relation $\cos(\theta) = 1/\beta n$. The TOP is composed of 16 rectangular sheets of quartz, where on one end a mirror is placed and on the other a detector. Ideally, the Cherenkov cone undergoes total reflection inside the quartz until it is measured with the detector. The purpose is to distinguish between $\pi^{\pm}$ mesons and $\mathrm{K}^{\pm}$ mesons that emit different cone angles, because of the mass difference of those particles.

The Aerogel ring-imaging Cherenkov detector (ARICH) is another type of RICH detector placed in boost direction next to the end-cap region next to the CDC. The charged particles produce Cherenkov radiation, when passing the 4 cm thick aerogel layer, if they exceed the speed limit in the medium. The radiation is detected by a photon detector plane. The ARICH is used to distinguish $\pi^{\pm}$ mesons from $\mathrm{K}^{\pm}$ mesons in the region between 14.78° and 33.71° surrounding the electron beam that otherwise would not be detected by the TOP.

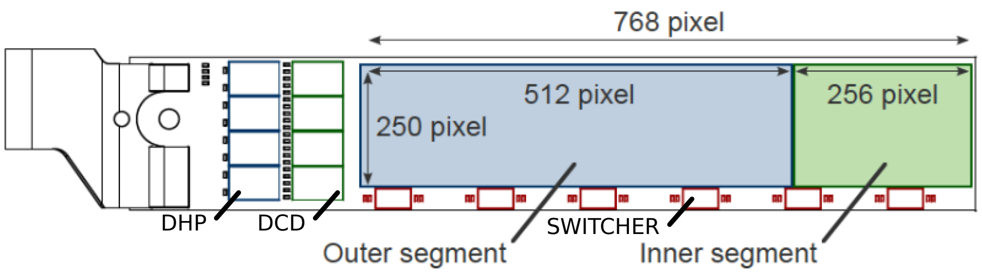


Figure 2.3 – Single half-ladder module. The SWITCHERs, DCDs and DHPs are shown at the rim [5].

The Electromagnetic Calorimeter (ECL) consists of 8736 thallium-doped caesium iodine crystals that are placed around the barrel, front and back end-cap region of the Belle II detector. Scintillation light is produced by electromagnetic interacting particles. The main purpose of the ECL is to measure the electron and photon energy.

The K_L and muon detector (KLM) measures neutral K_L mesons and muons. It is composed of iron plates, where kaons produce hadronic showers and mouns are slowed down. Between the iron plates resistive plate chambers (RPCs) are placed. Here, the muons produce a measurable signal. The signals of the muons are paired up with the CDC tracks, while neutral kaons do not produce a signal [5] [17].

2.3 PXD and DEPFET

The PXD is the most important subdetector concerning the subject of this thesis. Therefore, the detection principle and read out system is described in the following.
As mentioned, the PXD is the innermost subdetector and reconstructs precisely vertices of short lived particles. It is made of n-typed silicon wafer and contains 7.68 million pixels. In total 40 single half-ladder modules, shown in figure 2.3, are used. Every module is composed of 768 $\times$ 250 pixels and Application-Specific Integrated Circuits (ASICs) mounted on the rim that allow control and read-out.

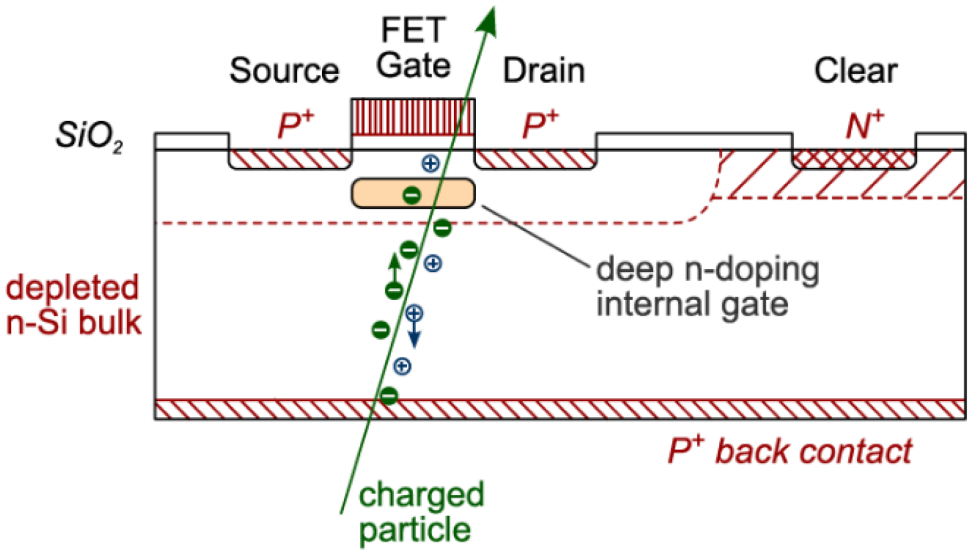


Figure 2.4 – Working principle of a DEPFET sensor [5].

The pixels are Depleted Field-Effect Transistors (DEPFETs) that are depicted in figure 2.4. The DEPFET principle is based on a p-MOSFET by adding p-doped back contact, where a negative voltage is applied. Bias voltage applied to the back contact will deplete the n-doped silicon bulk from charge carriers, making the MOSFET susceptible to charged particles. Those create electron hole pairs, where the holes drift to the p-doped back contact, because of the negative voltage applied, and electrons drift to the deep n-doped internal gate. The charge of the internal gate will increase the source drain current. Before saturation of the source drain current is reached, a positive voltage is applied at the n-doped clear to dispose electrons at the internal gate.

The read-out is carried out by three types of on-board mounted ASICs. The SWITCHER controls voltages and therefore the timing of read-out. The Drain Current Digitizer (DCD) uses an ADC to amplify and digitalize the extracted current. The DCD passes the data to Data Handling Processor (DHP), which buffers the data until all hits of the triggered event are measured. Per module six SWITCHERs, four DCDs and four DHPs are used.

Furthermore, the data from one module will be bundled using the Data Handling Engine (DHE), that then passes the data to the Data Handling

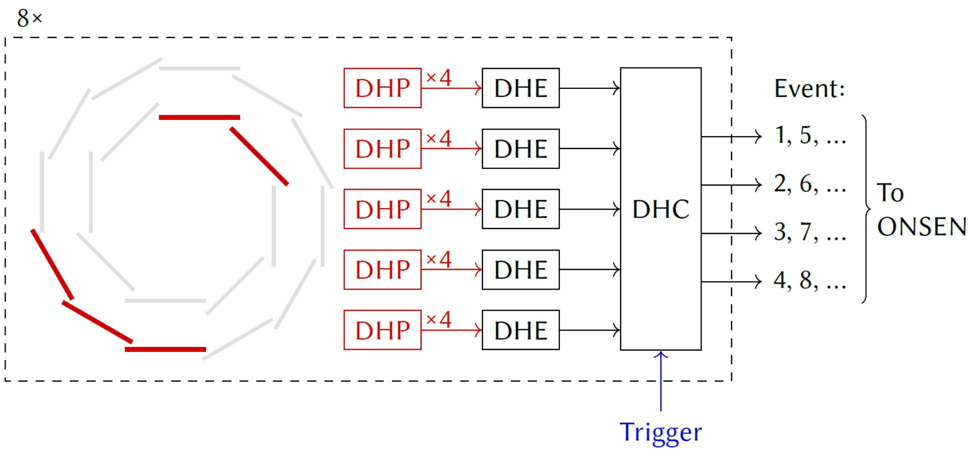


Figure 2.5 – External read-out data bundling of the PXD [11].

Concentrator (DHC). The DHC groups the data of five modules. In order to spread the occupancy, two of the inner layer and three of the opposite side outer layer modules are packed together as depicted in figure 2.5. Depending on the event number of the trigger system, the DHC outputs the data to one of four data lanes. In order to use all 40 modules, eight DHC are necessary [11] [17].

2.4 Trigger and Data Acquisition

The trigger system is used to detect specific data patterns that are produced by physics phenomena. On the first level, a individual hardware base trigger system is used for fast analysis. Most important trigger sources are the CDC that provides the tracks of particles, ECL for the number of clusters and energy deposit, ARICH and TOP, which obtain timing information and the KLM that detects muon tracks. A unified trigger signal, the Level-1 trigger, is build out of the Global Decision Logic (GDL) that uses the individual trigger sources as an input.
Other triggers are utilized to measure the luminosity from wellknown interactions or random triggers to detect the background.

After the data is detected, a data acquisition system is used to combine the data for further analysis. A schematic of the data acquisition system is shown in figure 2.6. All subdetectors except the PXD have similar data

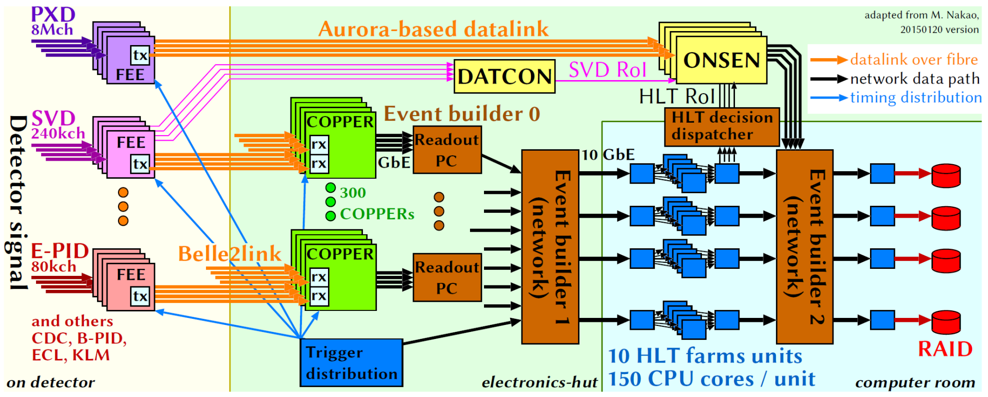


Figure 2.6 – Simplified schematic of DAQ system of Belle II [11].

rates and therefore contain similar data acquisition systems. The hardware based trigger system is accomplished by the Frontend-Electronis (FFE) boards. Afterwards, the data is sent to the Common Pipeline Platform for Electronis Read-out (Copper). Together with the CPUs of read-out PCs, they read timing and trigger signals and use this information for two staged event building (event building 0 and event building 1). The data is passed to the High Level Trigger (HLT), a PC farm that performs further event building (event building 2). This include the full reconstruction of the event. Afterwards, the data is permanently stored. The efficiency involving B meson decays is more than 99%.
The PXD has a data rate about a magnitude higher than the other subdetectors and needs support of a data reduction system. This involves a Online Selection Node (ONSEN) system that selects data of Region of Interest (RoI) of the PXD. The RoI sources are the Data Acquisition Tracking and Concentrator Online Node (DATCON) and the HLT. The DATCON system uses SVD data to reconstruct the particle track to the pixels of the PXD. The HLT creates RoIs making use of every subdetector except the PXD. Since the production using RoIs of the HLT is time consuming, the ONSEN system is capable of storing raw PXD data up to 5 seconds. After the selection, the ONSEN system forwards data for further event building. The data reduction with this processes is 1/10. More details to the ONSEN system is given in the next chapter [11] [17].

2.5 ONSEN System

The ONSEN system is part of Belle II data acquisition and is responsible to buffer and reduce unprocessed PXD data. This system was developed by a work group at the Justus-Liebig-University Gießen, Germany.
The design is based of xTCA architectures used in telecommunication industry to keep up with luminosity of the SuperKEKB. This consists of an Advanced Telecommunication Computing Architecture (ATCA) and the Advanced Mezzanine Card (AMC) shown in figure 2.7 and 2.8.

The current setup of the ONSEN system at the Belle II experiment utilizes an ATCA shelf with 14 slots for PCBs with characteristic size constraints, cooling fans, power supplies and a full-mesh backplane. The PCBs of the ATCA shelf are self developed and are called Compute Node Carrier Board (CNCB), illustrated in figure 2.9. The CNCB can host up to four AMCs. Since this thesis is focused on the firmware update, more details to CNCB will follow in the next chapter.

The full-mesh backplane connects the CNCBs with each other. The link between the CNCB is called fabric channel. The numbering of connections trough the ATCA shelf is shown in table A.1. The AMCs are smaller PCBs, which can be plugged into the CNCB and connect via LVDS links. They contain Virtex-5 FX70T FPGA for data processing, 4 GiB DDR2 as RAM, 64 MiB Flash as storage and optical or Ethernet ports.

A rear transition module (RTM) is connected to the back of the ATCA shelf that provides additional ports. A programmer is connected to RTM, which is used to program bitstreams to FPGAs of the CNCB, the AMC or the CPLD.

In the ONSEN system, nine CNCBs and a total of 33 AMCs are used. The firmware bitstreams differ, further dividing the CNCBs and AMCs into one Merger AMC, which merges RoIs of the two different sources, one Merger Carrier that distributes the merged RoIs to the Selector Nodes, eight Selector Carriers, which further distribute the RoIs between their four Selector AMCs and 32 Selector AMCs that filter PXD data depend-

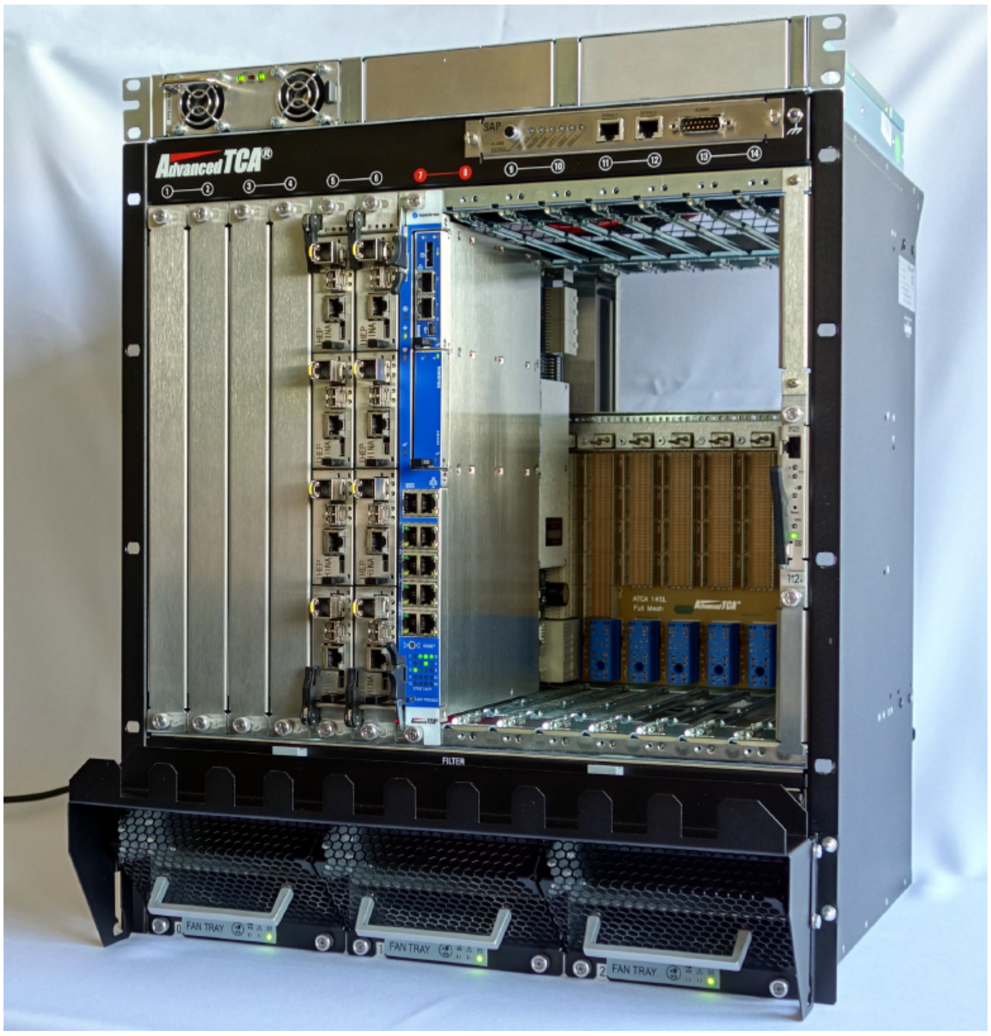


Figure 2.7 – Picture of the ATCA shelf. There are four dummy boards plugged in slot one to four. In slot five and six are two CNCBs that host four AMCs. A switch is shown in Slot 7, which is not used in the ATCA any more. On the right side of the shelf the backplane can be seen. Above the backplane is space for to connect to the RTM that is plugged from behind into the ATCA shelf [11].

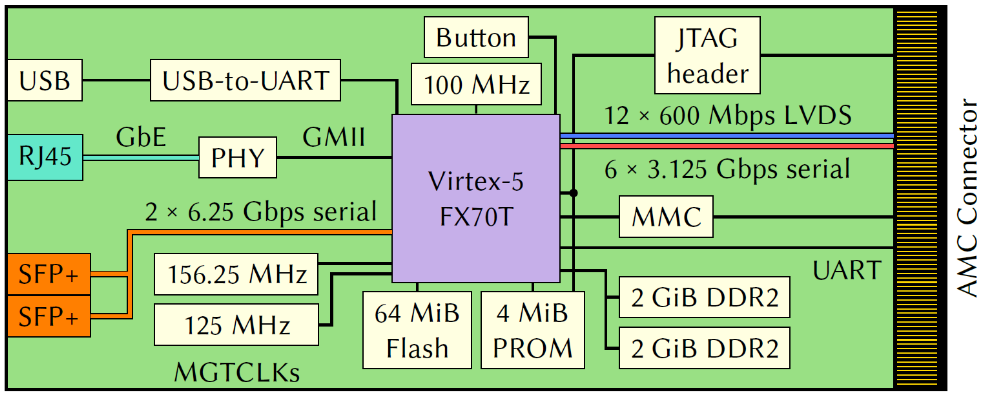


Figure 2.8 – Schematic of a AMC v4.0 [11].

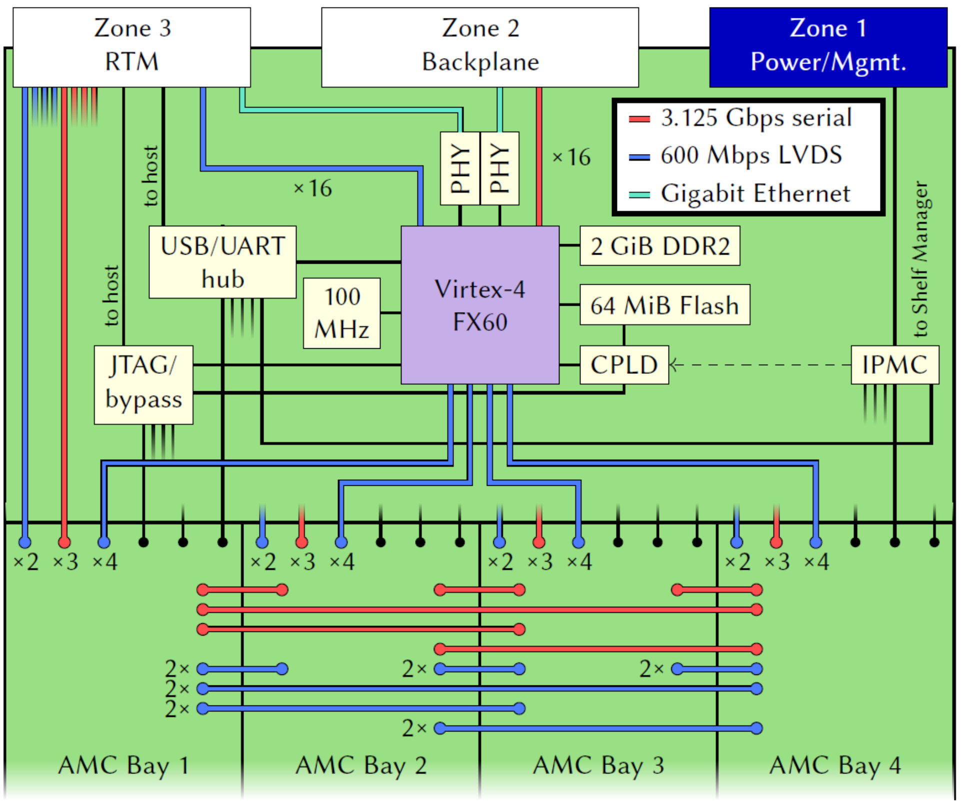


Figure 2.9 – Schematic of a CNCB v3.3 [11].

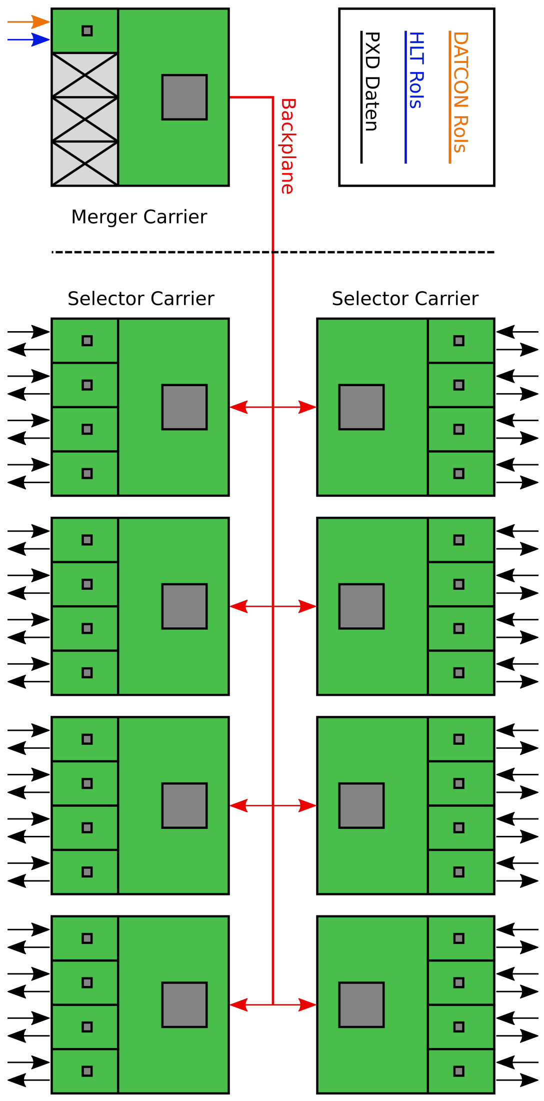


Figure 2.10 – RoI and data flow through the Belle II experiment setup of the ONSEN system.

ing on the RoI. A layout of the data flow through the ONSEN system is in figure 2.10 [11] [18].

3 Compute Node Carrier Board v4.0

This section consists of general information of the Compute Node Carrier Board (CNCB) v4.0, which is the hardware upgrade of the CNCB v3.3.

The hardware development of the Compute Node Carrier Board is foreseen for the Panda experiment, but the firmware is constructed such that it is compatible with the ONSEN system at the Belle II experiment. The board was designed by an IHEP group (Institute of High Particle Physics) in Beijing, China. The CNCB v4.0 was produced two times and is planed to be used as an emergency spare at the Belle II experiment.

3.1 Motivation

The CNCB v3.3 consists of outdated hardware components and can no longer be produced. A new version, the CNCB v4,0, was designed as an update to the Merger Carrier. For this, a new firmware for the FPGA of the Merger Carrier has to be developed.
The CNCB v4.0 introduces different and updated components, which leads to adjustments to the existing firmware that is used as a template. This allows the use of newer protocol standards in the firmware and one to one replacements are needed for alternative components.

3.2 CNCB Upgrade

The CNCB v4.0 is shown in figure 3.1. The most important changes of the upgrade are discussed in this section. We upgraded the FPGA

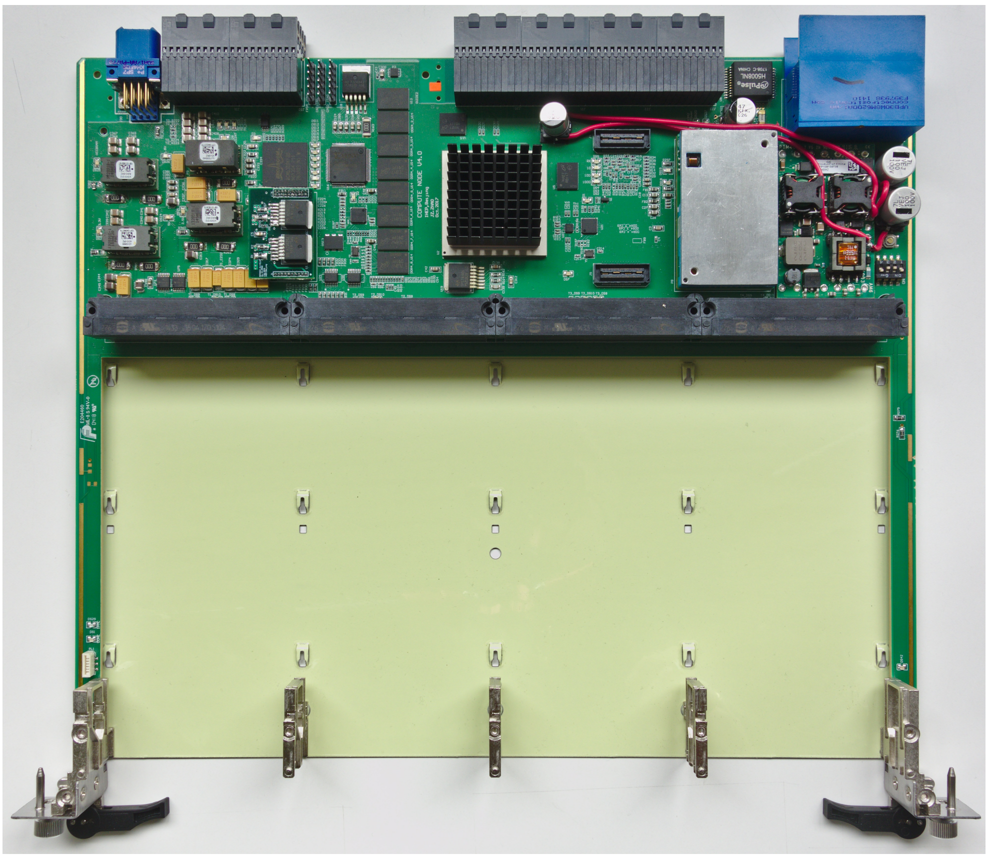


Figure 3.1 – Photograph of the CNCB v4.0.

to Kintex UltraScale with about ten times the resources. The differences between the FPGA are shown in table A.2. Also the memory was upgraded from 2 GiB DDR2 to 16 GiB DDR4 and the number of Multi-Gigabit Transceiver (MGT) port increased from 16 to 32. The firmware of the AMC was adjusted to make usage of MGT links, but is otherwise unchanged. The functionality of the PowerPC CPU was removed and replaced by a soft core Microblaze. A Microblaze is an CPU created with the resources of the FPGA. The Microblaze is called soft core, because the routing and placement of components inside the FPGA differs, depending on other occupied resources.

3.2.1 Components of the Compute Node Carrier Board

A schematic of the CNCB v4.0 is shown in figure 3.2. In this section, the most important parts and their function are mentioned. Beginning at the top left, there is the connector to the RTM as well as a power supply. To the right, there is a connector to the backplane with another power supply. Underneath both connectors, there are DC/DC converters mainly containing capacitors and resistors to provide the specific voltages for every component.
Starting from the left, the first logic chip is a 1 GB Ethernet-Switch that is used for connections to the Ethernet network. The Ethernet network connects the the CNCB and the AMCs with the RTM.
The Complex Programmable Logic Device (CPLD) is used for the JTAG connection to the FPGA that allows programming. There are four different modes, which can be selected with a switch on the far right of the board. Every mode uses a different input port. The first mode is to program the FPGA with a programmer that is connected to the RTM. The inputs are three 6-pins located to the right of the RTM connector. The leftmost pins are for direct FPGA programming, the ones in the middle are used to forward the bitstream to the FPGA, if the CPLD is initialized and the ones on the right are used to program the CPLD itself.
There are also Analog to Digital Convertes (ADCs) to measure and monitor voltages. On the backside of the CNCB, more ADCs are located to monitor the temperature.
The CNCB contains eight 2GB DDR4 chips, which are used as Random Access Memory (RAM) for the Microblaze and global byte storage.
The data processor of the board is the FPGA Kintex UltraSacle. All of the firmware changes are connected to the firmware of the FPGA.
Next to the FPGA, there are two flash chips. Since the FPGA is a volatile hardware, the flash chips are used as storage for the automatic programming of the FPGA, when the CNCB is turned on.
There are two Intelligent Platform Management Controller (IPMC) connectors to attach to one additional IPMC board. The IPMC operates the shelf manager, cooling fans and power input.
A button is placed on the far right side of the board to reset the CNCB

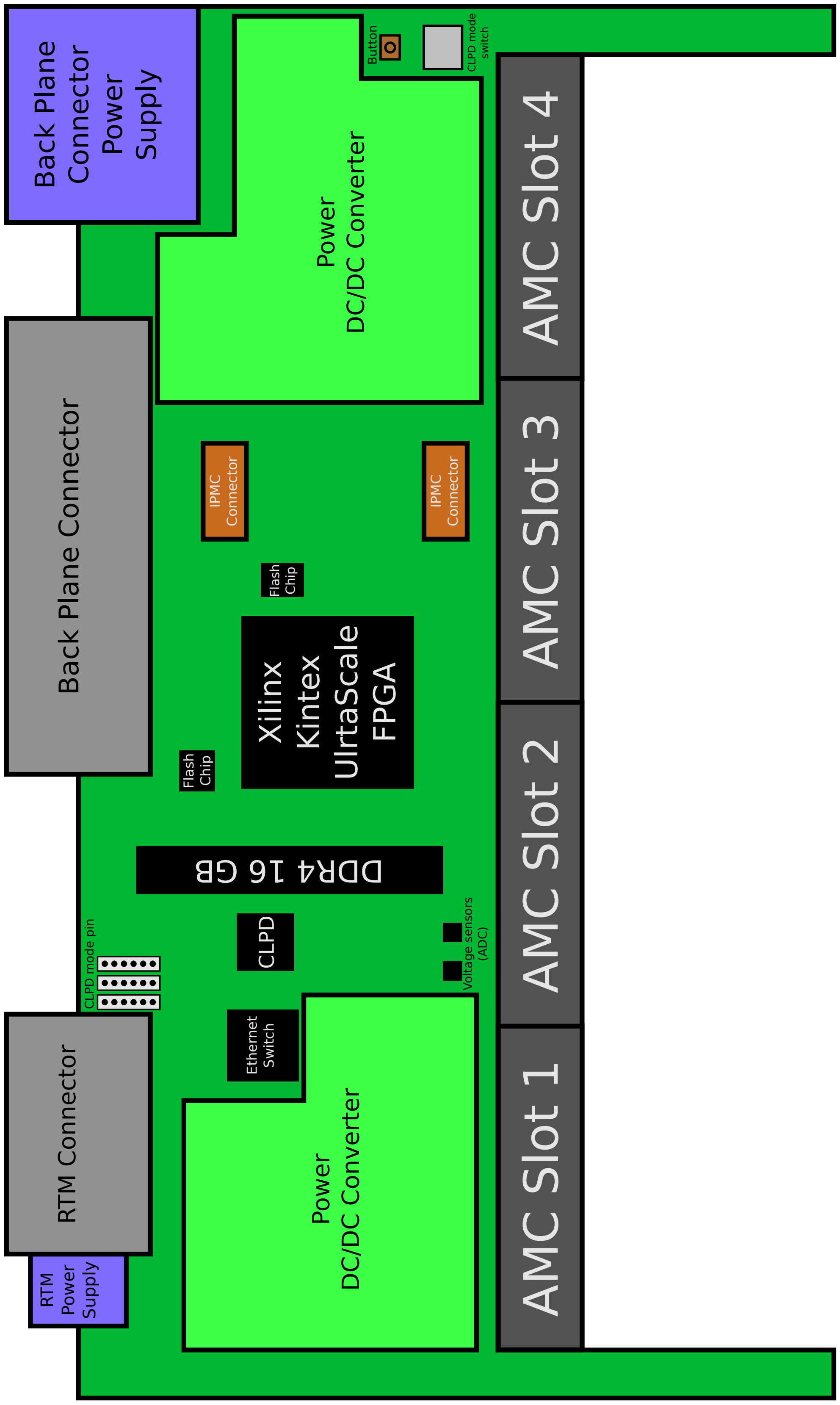


Figure 3.2 – Schematic of the CNCB v4.0.

manually. In addition, the CNCB utilizes four AMC connectors through which power is supplied to the AMCs and data can be exchanged [18].

3.3 Block Design of the Merger Carrier

The block design contains the whole firmware, which will be implemented in the FPGA. It comprises several Intellectual Property cores (IP cores) that are shown as a black box. The IP cores can be further divided into peripheral cores and custom cores, where peripheral cores do not process or work with the RoI data and custom cores, which are specifically designed to work with the data. The block design, peripheral cores and templates for the custom cores are created with the Embedded Development Kit (EDK).

In the block design, the inputs and outputs of an IP core can be plugged into other IP cores or be external. External ports are assigned to ports of the FPGA. The full block design is shown in figures A.3 to A.6 in the appendix. The peripheral cores handle the connections of other components of the board e.g. the Ethernet-Switch, addressing the DDR4 or the soft core Microblaze. Since the custom cores play a much more important role and almost have no connection to the peripheral core, a simplified block designed was created, which only shows the custom cores (figure 3.3). The inputs are on the left of the core and the outputs are on the right. The connections for slave register and interrupts between the custom IP cores and the Mircoblaze (MB) are indicated. For simplifications, not every port is shown. Unused ports, ports not relevant for this thesis e.g. clocks, reset ports and similar ports are left out or are grouped together.

This schematic focuses on the RoI data flow through the Merger Carrier. The data passes first through the Aurora core that is called `cncb4_mgt_-aurora_axis_1`. It is in the Aurora protocol format, which consists of a bandwidth of 1 bit and clock speed of 3.125 GHz. The data will be converted to the AXIS protocol, a protocol with a bandwidth of 32 bits and a clock speed of 78.125 MHz. The AXIS protocol is used as data

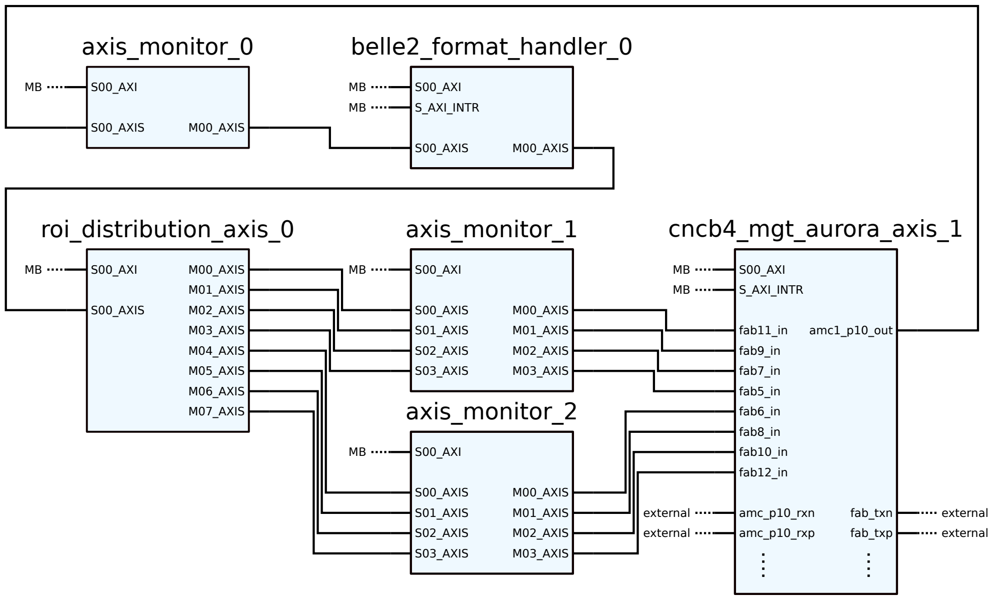


Figure 3.3 – Simplified schematic of the block design.

bus between and inside cores and is explained in further detail in the next section. After the data is converted, it will be sent via the port `amc1_p10_out`. The input and output ports correspond to AMC slot one. The ports of other slots are available and functional, but not shown. Then the RoI Data passes through the first Monitor core (`axis_monitor_0`). This core counts the number of incoming data. The Format Handler core (`belle2_format_handler_0`) checks the data format. The data with a wrong format or checksum will be discarded. Afterwards, the RoI Distribution core (`roi_distribution_axis_0`) determines, which data will be passed to which Selector Carrier. The number of Selector Carriers, used in the ONSEN system, is selectable. The outgoing data will be counted again with Monitor cores (`axis_monitor_1` and `axis_monitor_2`). Because the maximum number of inputs is four, two Monitor cores are utilized. Then the data will be converted to the Aurora protocol and sent through the backplane to the Selector Carrier. The ports `fab_txp` and `fab_txn` are external and examples for the outgoing data port. For every possible fabric channel, those ports exists. There are also external ports to convert incoming data from the fabric channels to the AXIS protocol and also external ports to send converted data to the AMC, but those are not in use.

3.4 Custom IP Cores

Before explaining the structure of each custom IP core, general information of the VHDL code structure used in the cores are presented.
The AXI Stream (AXIS) protocol, which is the interface used between and in the inside of the cores, consists of several ports that are classified as a slave or as a master. Slave interfaces are indicated with the letter (`s`) and is a data input, but not every meta data port of the slave interface is an input. The opposite is valid for the master interface that is indicated with the letter (`m`).
An overview of every port is given in the following table:

Table 3.1 – The AXI Stream ports.

	Slave	Master	Bandwidth	Mode
`TDATA`	Input	Output	32	-
`TKEEP`	Input	Output	4	Active high
`TVALID`	Input	Output	1	Active high
`TREADY`	Output	Input	1	Active high
`TLAST`	Input	Output	1	Active high

The data is forwarded by the `TDATA` port. This port has a minimum data granularity of 8-bit. Only bandwidths of 32-bits are used for the whole project. The port `TKEEP` indicates which 8-bit packet of `TDATA` is asserted. Since the data is logically converted with a 10-to-8-bit converter, there could be 8-bit packages, which are left over if data is packaged only in 32-bits. The port `TVALID` indicates if the `TDATA` is valid. If the data is not valid, it will not be processed further. In addition, the port `TREADY` has the opposite flow of slave or master interface and implies, if the destination is ready to process more data. This port will eventually, after all buffers and storage are depleted and hold long enough, extend to the beginning of data input of the Merger Carrier. This native flow control will create back pressure and stop incoming data. The port `TLAST` shows the end of the incoming data packet.
The CNCB v3.3 uses a different but very similar protocol, the LocalLink (ll) protocol. It contains the same ports as mentioned above with dif-

ferent names and one additional port. The start-of-frame port does not exist anymore. Similar to `TLAST`, it indicated the start of a data packet.

In all custom cores slave registers exist. Slave registers are special bits that can be addressed with the Microblaze. These bits allow the usage of monitoring and controlling the operation of the Merger Carrier. A slave register has 32 bits, but several slave registers can be present in one core. These register can be only readable or readable and writeable.
The Microblaze runs a Linux operating system for the slow control. There are two slow control programs to control an monitor the operation of the ONSEN system. Node is a program that allows for direct use of the slave registers. The program EPICs uses process variables (PVs) to monitor and write to special bits. EPICs works with node and adds a graphical interface. Both programs initialize configurations. Details to the slow control are in section A.1.2.
For the Aurora core and the Format Handler core interrupts are used. These interrupts are created by already existing logic. The Microblaze takes care of registering the interrupt. The interrupt logic can have up to 32 individual interrupts per core. While Node registers the total number of interrupts of each core, slow control logic exists to detect individual interrupts.

The general VHDL structure consists of a top file, which determines the ports of the core in the block design. The top file instantiates the interrupt and slave register logic as well as more core specific logic, which itself can instantiate more VHDL files.

3.4.1 Aurora Core

The Auroa core is used to convert the protocol from the Aurora protocol to the AXIS protocol and back. For data sending or receiving MGTs are utilized. The tree diagram in figure 3.4 shows the instantiation structure of this core. The top file instantiates the Aurora wrappers, slave register and interrupts. There are in total 16 wrappers, which all contain a MGT. The MGT is implemented with `aurora_sender1`, a Xilinx `aurora_8b10b` core [19] that takes care of receiving and sending data.

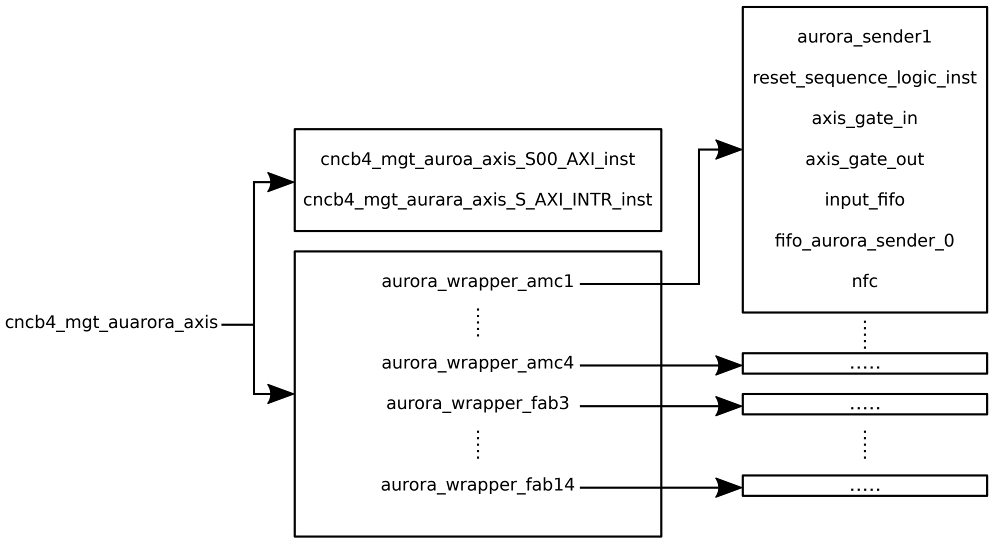
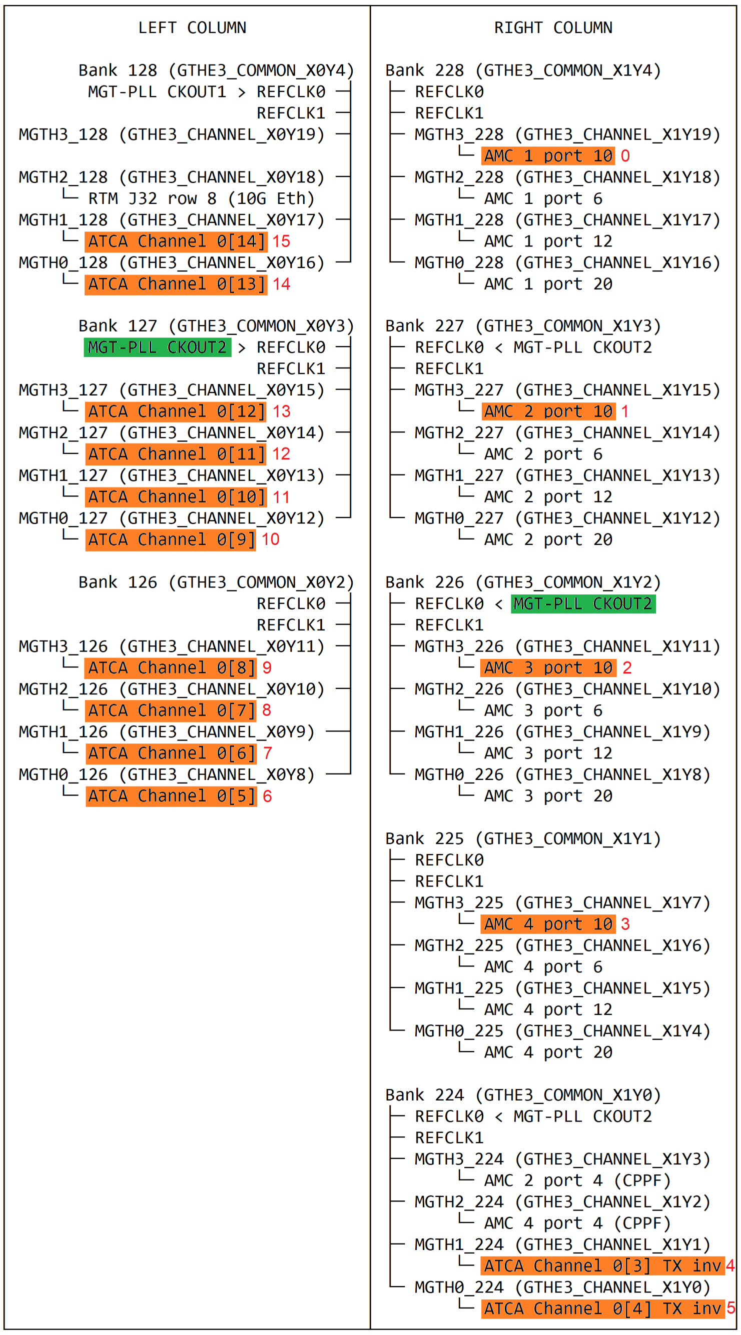


Figure 3.4 – Hierarchy of the instantiation of the Aurora core.

Every `aurora_8b10b` has two separate converters. One is used to convert incoming data from the Aurora protocol to the AXIS protocol and the other one is used to convert data back from the AXIS protocol to the Aurora protocol. Because we only receive data from the AMCs, we only use the conversion to the AXIS protocol. Although the initialization of the unused conversion is necessary for the flow control. For the outgoing data of the `aurora_sender1` of the fabric channels, we only utilize the converters to the Aurora protocol.
The Aurora protocol has a 1-bit bandwidth, a packet of 10 bins, a fast clock of 3.125 GHz and uses serial data transfer. The data has two different representations. One representation contains more `'1'` and the other one more `'0'`. This is used to reduce the charge of the ports, which would cause data loss if ignored. The MGT bundles 10-bits to a 8-bit and bundles four 8-bits to one 32-bit data word of the AXIS protocol. The 10 to 8-bit conversion allows for error detection and key-characters that are part of the Aurora protocol and contain meta data. Since all the 32-bits are send parallel, the clock speed is greatly reduced by forty times to 78.125 MHz. For this procedure the MGT needs reference clocks.

The resources of the FPGA are divided into two columns that are physically unconnected as shown in the figure 3.5. In the column, multiple banks exists. The right column bank 225 to 228 contain the MGTs for



```
                 LEFT COLUMN

       Bank 128 (GTHE3_COMMON_X0Y4)
        MGT-PLL CKOUT1 > REFCLK0 ─┐
                         REFCLK1 ─┤
MGTH3_128 (GTHE3_CHANNEL_X0Y19) ─┤

MGTH2_128 (GTHE3_CHANNEL_X0Y18) ─┤
   └ RTM J32 row 8 (10G Eth)     │
MGTH1_128 (GTHE3_CHANNEL_X0Y17) ─┤
   └ ATCA Channel 0[14] 15       │
MGTH0_128 (GTHE3_CHANNEL_X0Y16) ─┘
   └ ATCA Channel 0[13] 14

       Bank 127 (GTHE3_COMMON_X0Y3)
        MGT-PLL CKOUT2 > REFCLK0 ─┐
                         REFCLK1 ─┤
MGTH3_127 (GTHE3_CHANNEL_X0Y15) ─┤
   └ ATCA Channel 0[12] 13       │
MGTH2_127 (GTHE3_CHANNEL_X0Y14) ─┤
   └ ATCA Channel 0[11] 12       │
MGTH1_127 (GTHE3_CHANNEL_X0Y13) ─┤
   └ ATCA Channel 0[10] 11       │
MGTH0_127 (GTHE3_CHANNEL_X0Y12) ─┘
   └ ATCA Channel 0[9] 10

       Bank 126 (GTHE3_COMMON_X0Y2)
                         REFCLK0 ─┐
                         REFCLK1 ─┤
MGTH3_126 (GTHE3_CHANNEL_X0Y11) ─┤
   └ ATCA Channel 0[8] 9         │
MGTH2_126 (GTHE3_CHANNEL_X0Y10) ─┤
   └ ATCA Channel 0[7] 8         │
MGTH1_126 (GTHE3_CHANNEL_X0Y9)  ─┤
   └ ATCA Channel 0[6] 7         │
MGTH0_126 (GTHE3_CHANNEL_X0Y8)  ─┘
   └ ATCA Channel 0[5] 6
```

```
                 RIGHT COLUMN

Bank 228 (GTHE3_COMMON_X1Y4)
├ REFCLK0
├ REFCLK1
├ MGTH3_228 (GTHE3_CHANNEL_X1Y19)
│    └ AMC 1 port 10 0
├ MGTH2_228 (GTHE3_CHANNEL_X1Y18)
│    └ AMC 1 port 6
├ MGTH1_228 (GTHE3_CHANNEL_X1Y17)
│    └ AMC 1 port 12
└ MGTH0_228 (GTHE3_CHANNEL_X1Y16)
     └ AMC 1 port 20

Bank 227 (GTHE3_COMMON_X1Y3)
├ REFCLK0 < MGT-PLL CKOUT2
├ REFCLK1
├ MGTH3_227 (GTHE3_CHANNEL_X1Y15)
│    └ AMC 2 port 10 1
├ MGTH2_227 (GTHE3_CHANNEL_X1Y14)
│    └ AMC 2 port 6
├ MGTH1_227 (GTHE3_CHANNEL_X1Y13)
│    └ AMC 2 port 12
└ MGTH0_227 (GTHE3_CHANNEL_X1Y12)
     └ AMC 2 port 20

Bank 226 (GTHE3_COMMON_X1Y2)
├ REFCLK0 < MGT-PLL CKOUT2
├ REFCLK1
├ MGTH3_226 (GTHE3_CHANNEL_X1Y11)
│    └ AMC 3 port 10 2
├ MGTH2_226 (GTHE3_CHANNEL_X1Y10)
│    └ AMC 3 port 6
├ MGTH1_226 (GTHE3_CHANNEL_X1Y9)
│    └ AMC 3 port 12
└ MGTH0_226 (GTHE3_CHANNEL_X1Y8)
     └ AMC 3 port 20

Bank 225 (GTHE3_COMMON_X1Y1)
├ REFCLK0
├ REFCLK1
├ MGTH3_225 (GTHE3_CHANNEL_X1Y7)
│    └ AMC 4 port 10 3
├ MGTH2_225 (GTHE3_CHANNEL_X1Y6)
│    └ AMC 4 port 6
├ MGTH1_225 (GTHE3_CHANNEL_X1Y5)
│    └ AMC 4 port 12
└ MGTH0_225 (GTHE3_CHANNEL_X1Y4)
     └ AMC 4 port 20

Bank 224 (GTHE3_COMMON_X1Y0)
├ REFCLK0 < MGT-PLL CKOUT2
├ REFCLK1
├ MGTH3_224 (GTHE3_CHANNEL_X1Y3)
│    └ AMC 2 port 4 (CPPF)
├ MGTH2_224 (GTHE3_CHANNEL_X1Y2)
│    └ AMC 4 port 4 (CPPF)
├ MGTH1_224 (GTHE3_CHANNEL_X1Y1)
│    └ ATCA Channel 0[3] TX inv 4
└ MGTH0_224 (GTHE3_CHANNEL_X1Y0)
     └ ATCA Channel 0[4] TX inv 5
```

Figure 3.5 – Resource placement of the MGTs in the Kintex UltraScale.

the AMC. While multiple ports exists, only one port is needed. In this project port 10 is used to connect to the AMC. The ports in usage are marked orange. As for the reference clock, one clock was chosen per column. The chosen clocks are indicated with a green color. Since the MGT uses its build-in clock, the VHDL code differs slightly to the other MGT of the column. This `aurora_8b10` automatically uses its build-in clock and outputs it. Then the clock is distributed to the other MGTs of the column that contain no shared logic with the reference clock of the bank. The MGTs of the fabric channels 3 and 4 also use this clock, because they are in the right column. The number of the fabric channel is written in the squared brackets. The remaining fabric channels are in the left column. Here, the same VHDL code is utilized as for the AMC. Fabric channel 11 uses its build in reference clock, which is shared with the other fabric channels of the left column. The red number behind each MGT indicates the numbering of the signals that is used for the specific Aurora wrapper. For the upper 16 bits of the slave registers, the same order is kept. A summary can be found in the table 3.2.
Moreover, the MGT has two resets that have to be sequenced in the right order. For this, special logic is implemented that activates the resets, if the global reset or soft reset signal is set externally or by the slave registers.

The Aurora wrapper contains two asynchronous First In First Outs (FIFOs). FIFOs buffer data and are often used if data is converted, because small deviations of the clock speed can cause data loss. Asynchronous FIFOs are able to read and write independently with different clocks. The output FIFO is placed after data conversion to AXIS and the input FIFO is placed before data conversion to the Aurora protocol. The output FIFO creates a signal, if it is almost filled. This signal will be passed to the state machine of the native flow control (NFC). This state machine will then communicate with the `aurora_8b10b` with a bundled signal `S_AXIS_NFC`, which contains a `TREADY`, `TVALID` and a `TDATA` signal. The `TDATA` signal is a 4-bit vector and differs significantly since it does not contain RoI data. The `aurora_8b10b` then sends a signal to the AMC to stop data output. Directly before the output FIFO and after the input FIFO a AXIS gate is placed. The AXIS gate can stop the data

Table 3.2 – General information to MGT usage of the Aurora Core.

Instance	Code	GTHE_Channel	Clock domain	SR assignment
AMC 1 port 10	`aurora_wrapper_without_shared`	`X1Y19`	MGT-PLL of Bank 226	0 or 16
AMC 2 port 10	`aurora_wrapper_without_shared`	`X1Y15`	MGT-PLL of Bank 226	1 or 17
AMC 3 port 10	`aurora_wrapper`	`X1Y11`	MGT-PLL of Bank 226	2 or 18
AMC 4 port 10	`aurora_wrapper_without_shared`	`X1Y7`	MGT-PLL of Bank 226	3 or 19
Fabric Channel 3	`aurora_wrapper_without_shared`	`X1Y1`	MGT-PLL of Bank 226	4 or 20
Fabric Channel 4	`aurora_wrapper_without_shared`	`X1Y0`	MGT-PLL of Bank 226	5 or 21
Fabric Channel 5	`aurora_wrapper_without_shared`	`X0Y8`	MGT-PLL of Bank 127	6 or 22
Fabric Channel 6	`aurora_wrapper_without_shared`	`X0Y9`	MGT-PLL of Bank 127	7 or 23
Fabric Channel 7	`aurora_wrapper_without_shared`	`X0Y10`	MGT-PLL of Bank 127	8 or 24
Fabric Channel 8	`aurora_wrapper_without_shared`	`X0Y11`	MGT-PLL of Bank 127	9 or 25
Fabric Channel 9	`aurora_wrapper_without_shared`	`X0Y12`	MGT-PLL of Bank 127	10 or 26
Fabric Channel 10	`aurora_wrapper_without_shared`	`X0Y13`	MGT-PLL of Bank 127	11 or 27
Fabric Channel 11	`aurora_wrapper`	`X0Y14`	MGT-PLL of Bank 127	12 or 28
Fabric Channel 12	`aurora_wrapper_without_shared`	`X0Y15`	MGT-PLL of Bank 127	13 or 29
Fabric Channel 13	`aurora_wrapper_without_shared`	`X0Y16`	MGT-PLL of Bank 127	14 or 30
Fabric Channel 14	`aurora_wrapper_without_shared`	`X0Y17`	MGT-PLL of Bank 127	15 or 31

by setting its `TVALID` and `TREADY` port to low. The general structure of the Aurora wrapper is shown in figure 3.6. As before, the AXIS signals are bundled and shown as one data lane. Irrelevant ports are missing as well as a pulse synchronizer to convert the reset signal to another clock domain. The gates only use the `TVALID` and `TREADY` signals of the AXIS interface and are therefore drawn out of and into the grouped line. The `init_clock_in` is created by a clock wizard with a frequency of 50 MHz and is used to initialize the gigabit transceiver. The clock wizard is a peripheral core to create frequencies with a defined phase. The instance `aurora_wrapper_without_shared`, which is used for every wrapper except AMC 3 and fabric channel 11, has the `gt_refclk1_out` (156.25 MHz) and `user_clk_out` as an input from the Aurora wrapper in the same column. The destination of inputs and outputs of the Aurora wrapper is indicated. Note that both conversion lanes are labelled, but only one lane is used per wrapper. The `aurora_8b10b` also creates hard, soft and frame errors that are connected to the interrupt logic. This is only indicated by the port `err`.

The Aurora core has seven slave registers that contain status signals of the `aurora_8b10b`, the `enable` of the input and output gate of the Aurora wrapper and reset signals. The table 3.3 shows the assignments of the slave registers and if the slave register can be written to or not.

A block design of the whole are Aurora core is shown in figure 3.7. The dots inside of the Aurora core indicate that there are more signals of other Aurora wrappers, which are not shown. In this case, the signals get bundled together to 16-bit vectors. For the resets, the dotted lines mean that the signals come from the `aurora_wrapper` of its corresponding column.

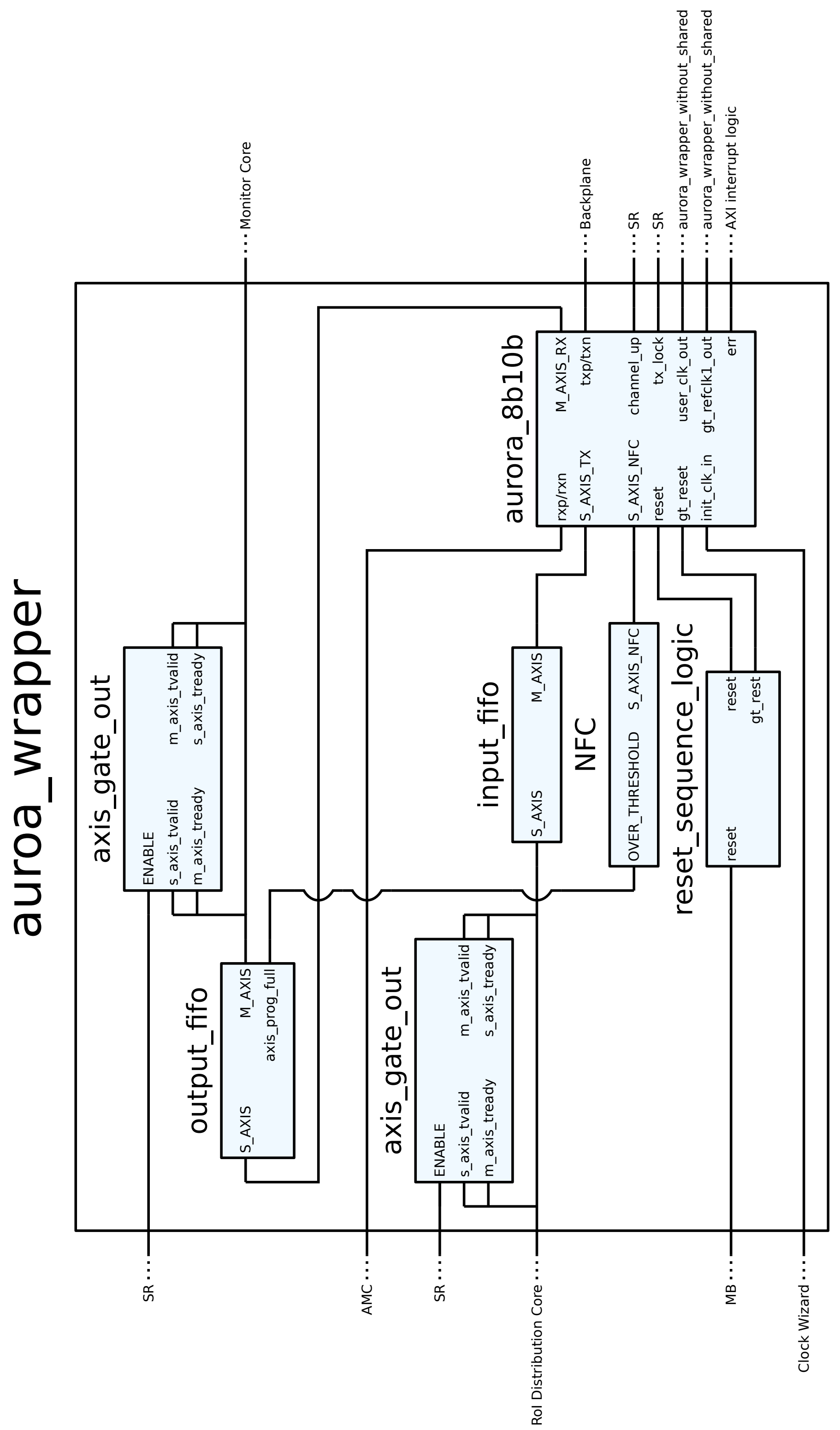


Figure 3.6 – Block design of the Aurora wrapper.

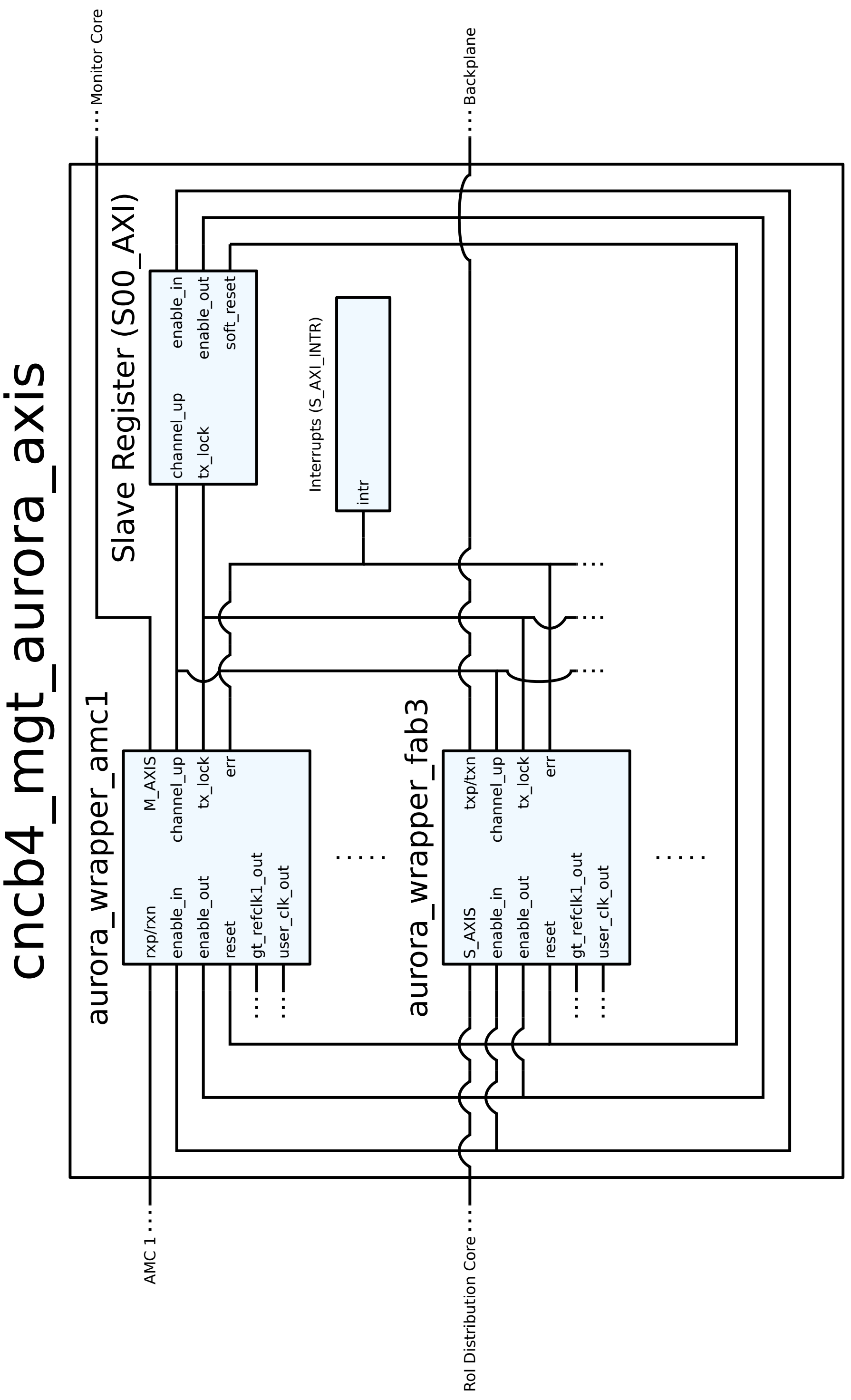


Figure 3.7 – Block design of the Aurora core.

Table 3.3 – Slave register assignments of the Aurora core.

Name	SR	Bits	Write status	Description
tx_lock	0	15 downto 0	Non writeable	Indicates the connection with the transceiver port
channel_up	1	31 downto 16	Non writeable	Indicates the connection with Aurora
enable_out	4	15 downto 0	Writeable	Opens the gate before second FIFO
enable_in	4	31 downto 16	Writeable	Opens the gate after first FIFO
reset_out	6	0	Writeable	Resets core

3.4.2 Monitor Core

The Monitor core counts the number of passed-through data. It is placed after and before the conversion of the Aurora core. Its instantiation hierarchy is shown in figure 3.8. The top file instantiates up to four `axis_counter`. Since there are eight data lines after the RoI Distriubtion core, two Monitor cores are used to count the data. The logic of counting data is written in the `axis_counter`. The data is divided into frames, which are further divided into words. Basically, a frame consists of meta data words, RoI data words and a checksum word. While the number bits, which make up the word is fixed to 32, the number of words, which a frame contains is arbitrary. More details are in section A.3.2. To count the number of frames, we use the `TLAST` port, which should only be high at the end of a frame. The number of frames and words are displayed with the slaver registers. The counters for frames and words are 32 bits long, therefore the Monitor core contains eight slave registers to display the

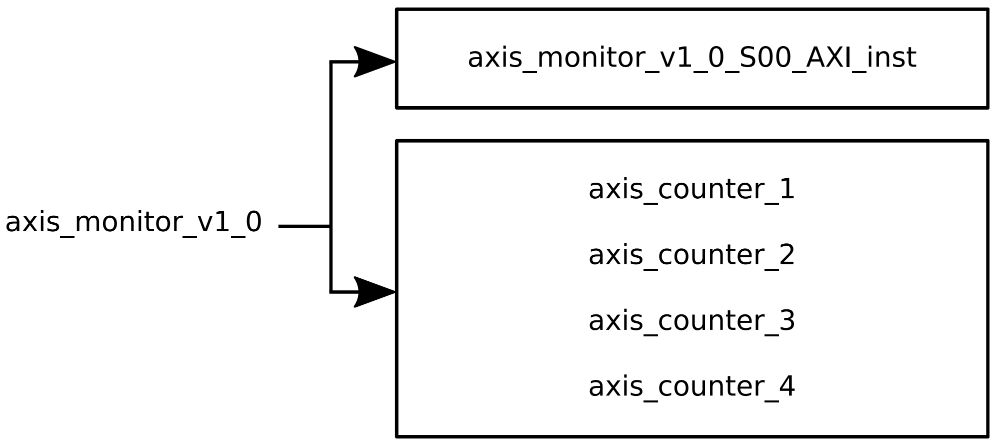


Figure 3.8 – Hierarchy of the instantiation of the Monitor core.

number of frames and words of every possible data lane. The assignment of slave register are listed in table 3.4.

Table 3.4 – Slave register assignments of the Monitor core.

Name	SR	Bits	Write status	Description
axis_1_word_count	0	31 downto 0	Non Writeable	Number of counted frames and words.
axis_1_frame_count	1	31 downto 0	Non Writeable	
axis_2_word_count	2	31 downto 0	Non Writeable	
axis_2_frame_count	3	31 downto 0	Non Writeable	
axis_3_word_count	4	31 downto 0	Non Writeable	
axis_3_frame_count	5	31 downto 0	Non Writeable	
axis_4_word_count	6	31 downto 0	Non Writeable	
axis_4_frame_count	7	31 downto 0	Non Writeable	

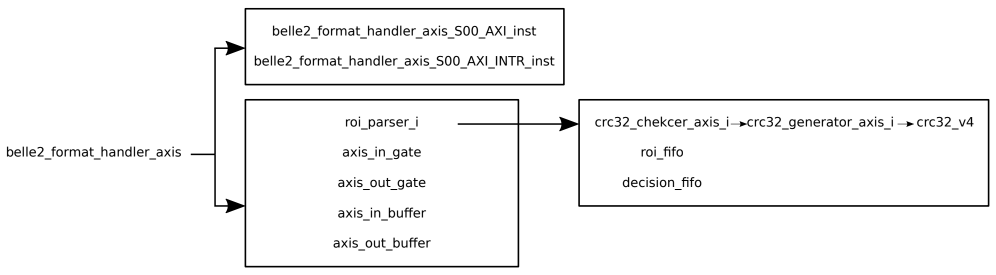


Figure 3.9 – Hierarchy of the instantiation of the Format Handler core.

3.4.3 Format Handler Core

The Belle II Format Handler core is used to verify the checksum and data format. A detailed description of the data format is given in section A.3.2. The VHDL code structure is shown in figure 3.9.

The top file instantiates two gates to stop data processing, two buffers that enable further resource placement, a RoI parser, which checks data, slave register and interrupts. The main logic is contained in the RoI parser. It is surrounded by buffers, that infer FIFOs with a depth of two. This improves the timing for critical paths of the AXI stream interface. There are two gates that are placed before the first buffer and after the second buffer. The gates contain an enable signal to allow data forwarding. For the gates the same structure is used as in the Aurora core. The RoI parser contains a state machine that progresses depending on the state of the data format. The list of errors, which can be detected are in table A.3.

The crc (cyclic redundancy check) is outsourced to the crc checker, which instantiates the crc generator and `crc_v4`. The crc checker contains a state machine to verify, when the crc check is done and if it is correct or not. It also calculates the start of the frame. The crc generator prepares the data to be used by the `crc_v4`. This includes optional bit and byte reflection and crc reset assignment. The `crc_v4` calculates the checksum of a whole 32 bit word. The polynomial to encode the checksum in hexadecimal representation is `04C11DB7`, which is used in Ethernet connections. As initial value it uses a string of zeroes and after this the calculated value is used as a base until the frame is finished. When the frame is finished and correct, the crc should be again a string of zeroes.

The Format Handler core contains 17 slave registers, which mainly infer signals for the PXD parser. The PXD parser is not yet implemented and necessary, if the AMCs of the ONSEN system will be upgraded. With the slave register we can control gate status, reset and the RoI parser options: downscaler factor, roisender factor and error mask. The slave register assignment is listed in table 3.5. Only slave registers 1 to 3 are writeable.

Table 3.5 – Slave register assignments of the Belle II Format Handler core.

Name	SR	Bits	
`m00_axis_tready`	0	0	Input `TREADY` signal.
`m00_axis_tvalid_buffer`	0	1	Output `TVALID` signal
`s00_axis_tready_buffer`	0	2	Output `TREADY` signal
`s00_axis_tvalid`	0	3	Input `TVALID` signal.
`enable_ptr_slv_plb`	1	0	PXD parser signal.
`enable_axis_out`	1	4	Enables output gate.
`enable_axis_in`	1	5	Enable input gate.
`enable_output`	1	8	Sets the signals `m_axis_tvalid_i`, `decision_fifo_wr_en`, `EVT_ID_FIFO_WR_EN`, `PLB_FIFO_WR_EN` and `PLB_INTR` to low.
`ASYNC_RST`	1	9	Rest for the entity `reset_synchronizer`.

trigger_number_shift	1	15 downto 12	PXD parser signal.
roisender	2	15 downto 0	Activates ROISENDER_FACTOR.
downscaler	2	31 downto 16	Activates DOWNSCALER_FACTOR.
error_mask	3	31 downto 0	Enables error detection of the corresponding bit in table A.3.
current_trigger	4	31 downto 0	PXD parser signal.
dhe_id_4	5	7 downto 2	PXD parser signal.
dhe_id_3	5	13 downto 8	
dhe_id_2	5	19 downto 14	
dhe_id_1	5	25 downto 20	
dhe_id_0	5	31 downto 26	
dhe_occ_0	6	31 downto 0	PXD parser signal.
dhe_occ_1	7	31 downto 0	
dhe_occ_2	8	31 downto 0	
dhe_occ_3	9	31 downto 0	
dhe_occ_4	10	31 downto 0	
dhe_occ_max_0	11	31 downto 0	PXD parser signal.
dhe_occ_max_1	12	31 downto 0	
dhe_occ_max_2	13	31 downto 0	
dhe_occ_max_3	14	31 downto 0	
dhe_occ_max_4	15	31 downto 0	
cm63_0	16	31 downto 26	PXD parser signal.
cm63_1	16	25 downto 20	
cm63_2	16	19 downto 14	
cm63_3	16	13 downto 8	

`cm63_4`	16	7 downto 2	

There is only one interrupt signal, which is high if at least one of the errors is detected. The kind of error will be written to a FIFO, which can be later on read by the slow control. This step is not implemented yet.

The figure 3.10 shows a simplified block design, that indicates the data flow through the Format Handler core as well as important signals.

3.4.4 RoI Distribution Core

The RoI Distribution core is used to multiplex the AXIS interface to a set number and select the ports to distribute to. For the internal structure the older LocalLink protocol is still in use. To implement the function, the top file converts AXIS protocol to LocalLink and back. The instantiation chain is shown is figure 3.11.

The top file instantiates slave registers and the previous top file of the RoI Distribution core. Its main purpose is to convert the protocols and pass signals to the salve registers. The entity `fork_trigger_packets` multiplexes the data ports. In the case that one or more Selector Carrier can not progress data, we need to stop data intake. For this, the seven `ll_ready_fork` are instantiated to synchronize the flow control ports of the LocalLink protocol. The selection is carried out by eight `trigger_packet_filter`. With the eight signals `TRG_NUM_LUT` that can be set via the slave registers, the selection can be programmed. This entity looks for the trigger number of a frame and decides depending of the contents of its `TRG_NUM_LUT` to pass through the frame. Since the trigger number is in the second word of a frame, the frame is buffered in a FIFO. A second decision FIFO is used to buffer the contents of `TRG_NUM_LUT` until the frame is finished.

A simplified block design of the RoI Distribution is shown figure 3.12. It focuses on the data pass through. Since the protocol conversion is not relevant, it is left out.

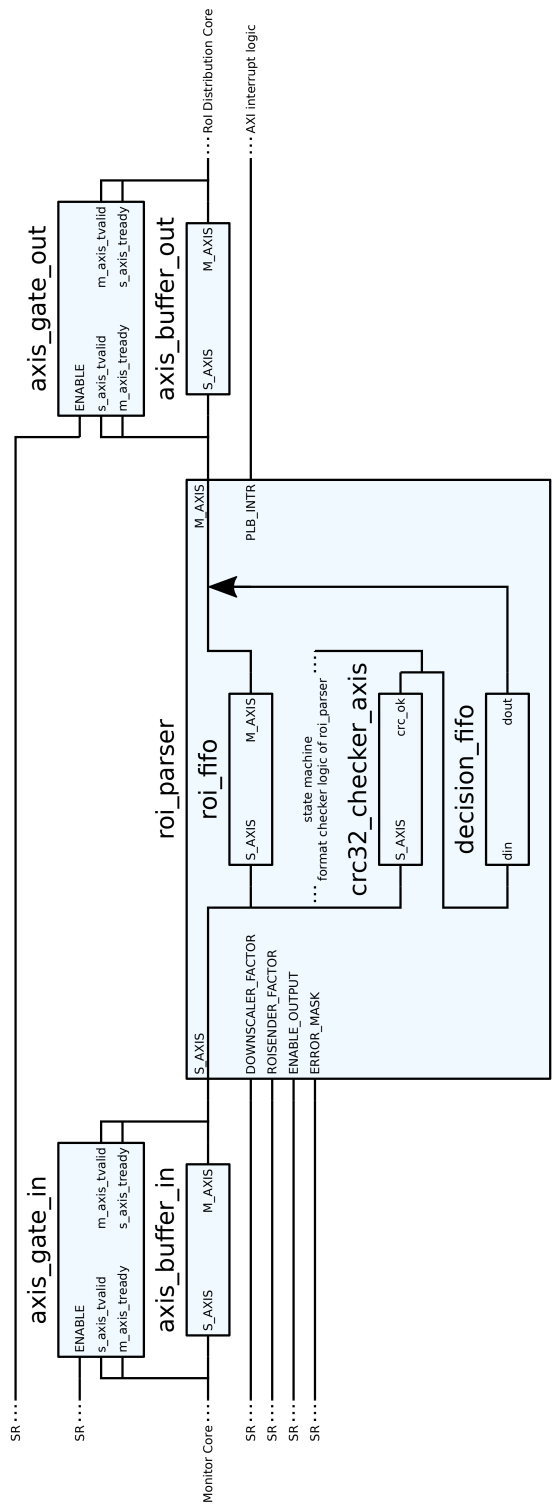


Figure 3.10 – Block Design of the Format Handler core.

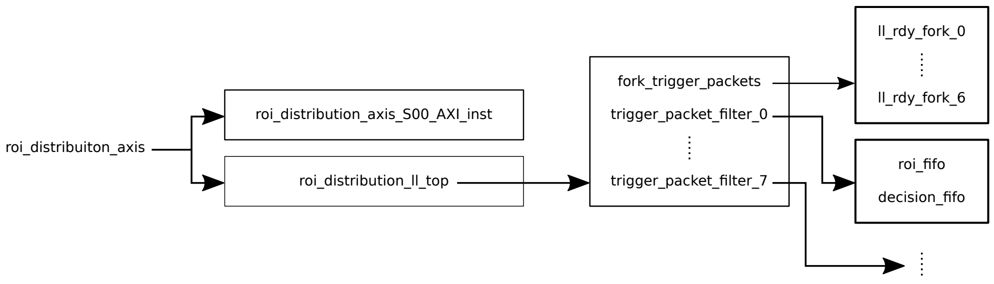


Figure 3.11 – Hierarchy of the instantiation of the RoI Distribution core.

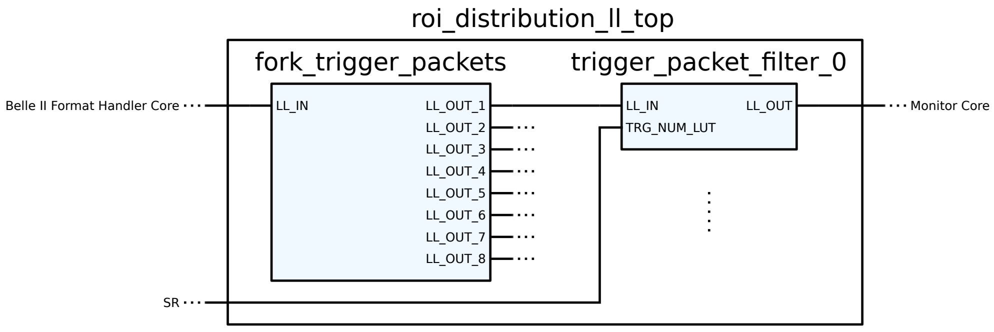


Figure 3.12 – Block Design of the RoI Distribution core.

The RoI Distribution core contains 20 slave registers. The `ENABLE_PORT` signal is especially importantly, to activate the distribution of RoIs and the contents of `TRG_NUM_LUT_1` to `TRG_NUM_LUT_8`. Only slave register 0 is not writeable. A list of the slave register assignment is in table 3.6.

Table 3.6 – Slave register assignments of the RoI Distribution core.

Name	SR	Bits	Description
reset	1	8	Resets core
ENABLE_PORT	1	31 downto 24	Enables port
TRG_NUM_LUT_1	2	31 downto 24	Specific filter option by trigger number
TRG_NUM_LUT_2	2	23 downto 16	
TRG_NUM_LUT_3	2	15 downto 8	
TRG_NUM_LUT_4	2	7 downto 0	
TRG_NUM_LUT_5	3	31 downto 24	
TRG_NUM_LUT_6	3	23 downto 16	
TRG_NUM_LUT_7	3	16 downto 8	
TRG_NUM_LUT_8	3	7 downto 0	
DHEID_LUT_1(63 downto 32)	4	31 downto 0	Specific filter option by DHE ID (process and occurring area)
DHEID_LUT_1(31 downto 0)	5	31 downto 0	
...	...	...	
DHEID_LUT_8(63 downto 32)	18	31 downto 0	
DHEID_LUT_8(31 downto 0)	19	31 downto 0	

4 Additions to the Firmware

In this chapter the progress of additions to the existing firmware is documented. First the laboratory testing setup is outlined, since it differs in terms of components.

The laboratory testing setup is used in Gießen to verify the operation the implementations. It contains one CNCB v4.0 as Merger Carrier, a xFP v4.0 as Merger AMC and one CNCB v3.3 as Selector Carrier. The firmware of the Merger Carrier is in progress. For the Selector Carrier and Merger AMC the firmware, which is used in the setup of Belle II experiment, is implemented. There are small changes to the firmware of the Merger AMC to use the MGT ports instead of LVDS ports of the previous setup. The Selector Carrier is in logical slot 2, the Merger Carrier is in logical slot 13 of the ATCA shelf and the Merger AMC is plugged in the first slot of the Merger Carrier. A programmer is utilized to program the FPGA of Selector Carrier, Merger Carrier or Merger AMC as well as the CPLD of each CNCB via a JTAG connector with JTAG protocol. For sending merged test RoIs (TCP protocol) and for the slow control Ethernet connection is used (TCP and UDP protocol).

A summary of additions is shown in table 4.1. Further exploitation is given in the next sections.

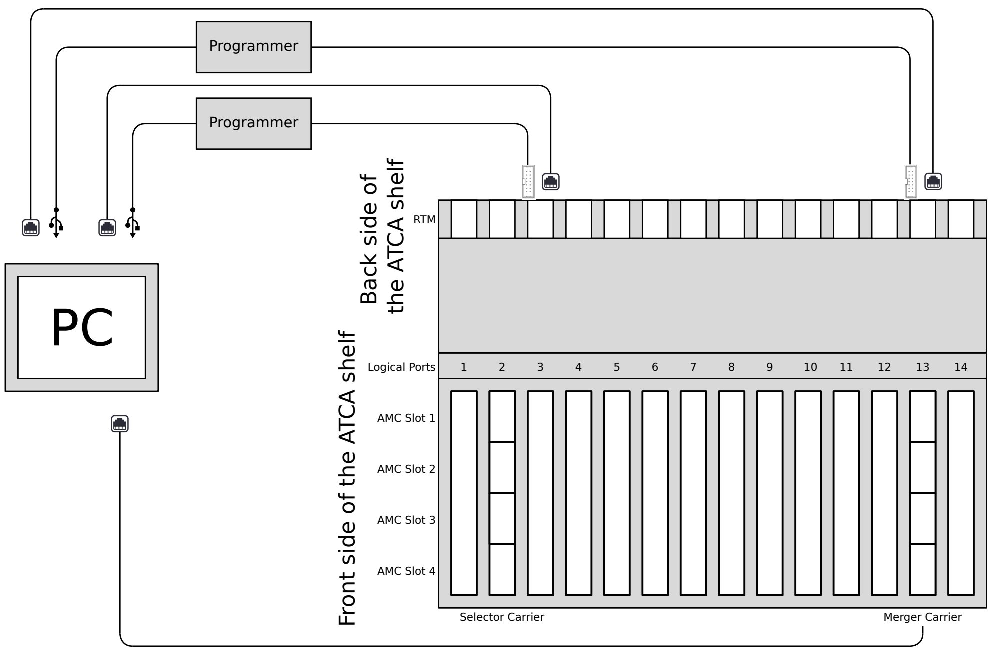


Figure 4.1 – Laboratory setup for testing[1].

Table 4.1 – Summary of additions to the firmware.

Addition	Description
Update interface protocol	The Local Link protocol is replaced by the AXIS protocol.
Fabric channel extension	Fabric channels are added for the Aurora core, so that the CNCB v4.0 can be used in all slots of the ATCA shelf.
Reset sequence logic	A sequenced reset is initialized to prevent MGT errors of the Aurora core.
Core resets	The cores are individually resettable with the slow control.
Interrupts	Interrupts of the Aurora core and the Belle II Format Handler core are processed and are displayed with the slow control.

4.1 Aurora Core Fabric Channel Extension

The extension of more fabric channels uses the code of the existing fabric channel as template. More Aurora Wrapper without shared logic are instantiated. The corresponding status and interrupt signals are also already implemented. The connection of these specialized bits are in table 3.2. Fabric channel 3 to 10 and 12 to 14 were added. Remark that only fabric channel 11 uses the built-in clock and that the reference signals are distributed to the other fabric channels depending of the column, where they are placed. The instantiation for fabric channel 5 is shown in listing 4.1. The AXIS interface ports are inputs or outputs of the conversion and `fab5_txp`, `fab5_txp`, `fab5_rxp` and `fab5_rxn` are the serial data inputs and outputs. Most of the remaining ports are slave register signals, interrupt signals or buffer signals.

```vhdl
aurora_wrapper_fab5 : entity work.aurora_wrapper_without_shared
    generic map(
        prereset_time => prereset_time,
        gt_reset_time => gt_reset_time,
        postreset_time => postreset_time
    )
    port map(
        channel_up => channel_up(6),
        txp => fab5_txp,
        txn => fab5_txn,
        rxp => fab5_rxp,
        rxn => fab5_rxn,
        tx_lock => tx_lock(6),
        reset => reset,
        reset_sequenced_out => reset_sequenced(6),
        init_clk_in => init_clk_in,

        m_axis_clk => fab5_out_axis_clk,
        s_axis_clk => fab5_in_axis_clk,
        user_clk => fab_user_clk_zw,
        sync_clk => fab_sync_clk_zw,
        gt_refclk1 => fab_gt_refclk1_zw,
        pll_not_locked => fab_pll_not_locked_zw,

        s_axi_tx_tdata => fab5_in_tdata,
        s_axi_tx_tkeep => fab5_in_tkeep,
        s_axi_tx_tlast => fab5_in_tlast,
        s_axi_tx_tvalid => fab5_in_tvalid,
        s_axi_tx_tready => fab5_in_tready,

        --ports fifo
        m_axis_tvalid => fab5_out_tvalid,
```

[1]Graphics of USB, RJ45 and JTAG port imported from [20] [21] [22].

```
        m_axis_tready => fab5_out_tready,
        m_axis_tdata => fab5_out_tdata,
        m_axis_tkeep => fab5_out_tkeep,
        m_axis_tlast => fab5_out_tlast,

        --ports axis_gate
        enable_out => enable_out(6),
        enable_in => enable_in(6),

        hard_or_soft_err => hard_or_soft_err(6),
        frame_err => frame_err(6)
    );
```

Listing 4.1 – Instantiation of fabric channel 5.

4.2 Aurora Core Reset Sequence Logic

The reset of the transceiver of the Aurora wrapper needs to be sequenced. There are two reset signals a `gt_reset` and a `reset`. The `gt_reset` resets only the gigabit-transceiver that runs on the `user_clk` domain, while the `reset` is general a signal for the surrounding logic. During the `gt_reset` the `user_clk` will be suppressed, which causes that the `reset` does not reach all portions of the programmable logic. The `init_clk` is always active and initializes the `user_clk` after the `reset` is set back to low and is also the domain of the signal `reset`.

The correct reset sequence is shown in figure 4.2. After a reset signal is produced by a soft reset from slave registers or general reset for all cores, the signal `reset` will be set to high. After at least 128 `user_clk` cycles the `gt_reset` can be switched to high. Both resets should be active for about one second. Then the `gt_reset` will be set to low and after that the `reset` will be set also to low synchronous to the `user_clk`. The sequence reset is shown in listing 4.2. Since using the `user_clk` will lead to complex logic with two state machines that need to exchange signals and have to be synchronized to the other clock domain, an easier solution was chosen. In the code, there is only one state machine synchronous to the `init_clk`. The working principle is depicted in figure 4.3. To compensate that the `init_clk` is slower than the `user_clk`, the number of cycles are multiplied by 16. Therefore, the ratio between the clocks should not exceed 16. The state machine contains three sates. It begins with an idle

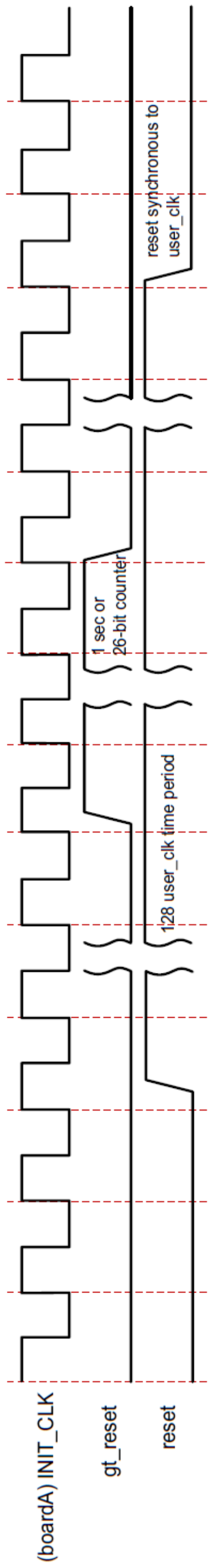


Figure 4.2 – Reset sequence for the `aurora_8b10b` [19].

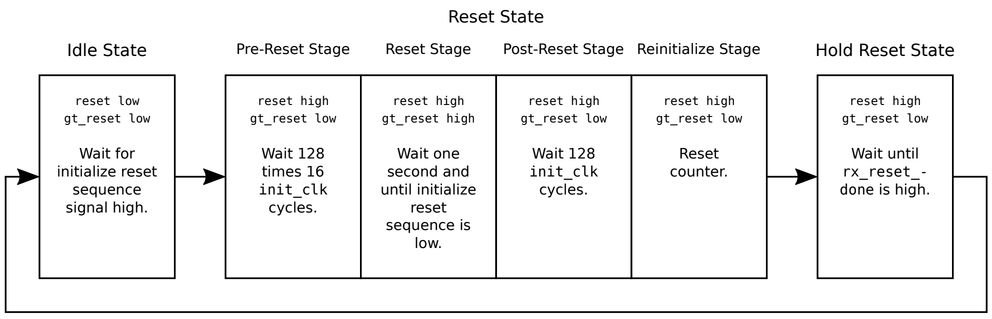


Figure 4.3 – Working principle of the reset sequence.

state, where we progress to the reset state, if the soft reset or global rest is initialized. Inside the reset state we have four different stages, divided by a counter. The breakpoints are generics that can be adjusted in the options of the core. In the pre-reset stage, we only set the signal reset signal to high. Here, two signals `reset_i` and `reset_unsync` are used for timing. The `reset_unsync` will be synchronized to the `user_clk` domain and therefore always be delayed to the `reset_i` signal. To begin the reset sequence as fast as possible, the `reset` will be set to high synchronized with the `init_clk`, but it is important to set the `reset` to low synchronized with the `user_clk`. For this, `reset_unsync` is synchronized to the `user_clk` and called `reset_sync`. By combining the `reset_i` and `reset_sync` with an OR-gate the reset signal is extended to be set to low with the `user_clk`. In the next stage, `reset` and `gt_reset` are set to high for around one second. If the soft reset or global reset is still high at this moment, we want to stay in this stage. Therefore, we stop incrementing the counter. Then the `gt_reset` will be set back to low and we wait for a third timer. At the end, we set the counter to zero and switch to another state to hold the reset. Here, we hold the signal `reset` until the signal `rx_resetdone` of the `aurora_8b10` is high. Then we go back to the beginning idle state. The logic was tested and verified to run stable with a test bench.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library xpm;
use xpm.vcomponents.all;


entity reset_sequence_logic is
    generic(
```

```vhdl
        prereset_time : integer;
        gt_reset_time : integer;
        postreset_time : integer
    );
    port(
        reset_sync_init_clk : in std_logic;
        user_clk : in std_logic;
        init_clk : in std_logic;
        rx_resetdone : in std_logic;

        reset : out std_logic;
        gt_reset : out std_logic
    );
end reset_sequence_logic;

architecture arch_imp of reset_sequence_logic is

    signal reset_i :std_logic := '0';
    signal reset_unsync :std_logic := '0';
    signal reset_out_sync :std_logic := '0';

    signal counter : unsigned(31 downto 0) := x"00000000";
    constant prereset_time_i : integer := prereset_time;
    constant gt_reset_time_i : integer := gt_reset_time + 16*
    prereset_time_i;
    constant postreset_time_i : integer := postreset_time +
    gt_reset_time_i;

    type state_type_reset is (idle_r, reset_r, hold_reset);
    signal state_reset : state_type_reset := idle_r;

begin

    process (init_clk) is
    begin
    if rising_edge(init_clk) then
        -- reset_unsync will be synchroniced with the user_clk.
        -- Since reset will be set to reset_i or reset_sync_out
        -- reset_sync_out should be assigent to '0' after
        -- reset_i is assigent.
        reset_i <= '0';
        reset_unsync <= '0';
        gt_reset <= '0';
        case state_reset is
            when idle_r =>
                reset_i <= '0';
                if reset_i = '0' then
                    reset_unsync <= '0';
                end if;
                if reset_sync_init_clk = '1' then
                    state_reset <= reset_r;
                end if;
            when reset_r =>
                counter <= counter + 1;
                -- Worst case: ratio between init_clk and user_clk is 16
                if counter <= 16*prereset_time_i then
                    reset_i <= '1';
                    reset_unsync <= '1';
                elsif counter > 16*prereset_time_i and counter < gt_reset_time_i then
                    reset_i <= '1';
                    reset_unsync <= '1';
                    gt_reset <= '1';
                elsif counter = gt_reset_time_i then
                    reset_i <= '1';
                    reset_unsync <= '1';
                    gt_reset <= '1';
                    -- Hold reset and gt_reset if reset_sync_init_clk is still active
                    if reset_sync_init_clk = '1' then
                        counter <= counter;
                    end if;
                elsif counter > gt_reset_time_i and counter < postreset_time_i then
                    reset_i <= '1';
                    reset_unsync <= '1';
                else
                    reset_i <= '1';
                    reset_unsync <= '1';
                    counter <= x"00000000";
                    state_reset <= hold_reset;
                end if;
            when hold_reset =>
                reset_i <= '1';
                reset_unsync <= '1';
                if rx_resetdone = '1' then
                    state_reset <= idle_r;
                end if;
        end case;
    end if;
    end process;
    -- reset_i is synced to init_clk and reset_out_sync is
    -- synced to user_clk. reset_out_sync will be later
    -- assigent to '0' than reset_i
    reset <= reset_i or reset_out_sync;

xpm_cdc_sync_rst_init_clk : xpm_cdc_sync_rst
   generic map (
      DEST_SYNC_FF => 4,   -- DECIMAL; range: 2-10
      INIT => 1,           -- DECIMAL; 0=initialize synchronization registers to 0, 1=initialize
                           -- synchronization registers to 1
      INIT_SYNC_FF => 0,   -- DECIMAL; 0=disable simulation init values, 1=enable simulation init values
      SIM_ASSERT_CHK => 0  -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   )
   port map (
      dest_rst => reset_out_sync,
      dest_clk => user_clk,
        src_rst => reset_unsync
    );

end arch_imp;
```

Listing 4.2 – Reset sequence logic.

4.3 Core Resets

The system can be reseted with a global reset signal for all cores. For an individual soft reset, we create a new signal that can be accessed via the slow control. This signal is not set to a slave register, but is set by writing a specific string to one of the slave registers. For every core, we choose slave register 0, because it is not writeable and only used for status signals. Therefore, modifications to read the logic for every core were created likewise as in listing 4.3.

```
        when b"000" =>
                   -- additional soft reset logic
                     if (S_AXI_WSTRB = "1111" and S_AXI_WDATA = x"0000000A")
     then
                          slv_reset_n <= '0';
                     end if;
```

Listing 4.3 – Additional read logic for soft resets.

The `000` is the local address, which stands for slave register 0. When the specific string in hexadecimal `0000000A` is read, the inverted reset signal is set to low. Otherwise, it is high. Furthermore, this will reset the slave registers and all other logic of the core.

4.4 Interrupts of Aurora Core and Belle II Format Handler Core

The interrupt logic was already created with a template for the Aurora core and Format Handler core. Here, it is necessary to connect the specific signal to the interrupt of the AXIS interrupt logic. In the case of the Aurora core, the `aurora_8b10b` produces soft, hard and frame errors. The maximum number of interrupts is 32. Since we have four `aurora_8b10b`

for the AMCs and twelve `aurora_8b10b` for the fabric channels, we have too many error signals. Therefore, the soft and hard errors are combined to one error with an OR-gate. The signals are further synchronized from the local user clock to the `s_axi_intr_aclk`. Inside the AXIS interrupt logic an example design is deleted and the soft-or-hard errors and frame errors are connected.
For the Format Handler core only one interrupt exists. It is set to high every time an error is detected by the RoI parser. It is connected to the AXIS interrupt logic as described for the Aurora core.

The total number of interrupts is displayed with Node. The interrupt logic differs from the previous implementation, therefore the Node scripts need to be adjusted. Node is built with scripts that can be categorized in definitions, address location and logic. The address is written in the state machine of the AXIS interrupt logic. By addressing the registers of the state machine, it is possible to enable and disable interrupts. The acknowledgement of an interrupt is managed with a `manage_interrupt()` function, that writes the content of the pending interrupt signal to the acknowledge interrupt signal. Furthermore, the PVs that can be also accessed by EPICs, PV names were created and the address was defined.

4.5 Results of Tests

The firmware was tested in the local laboratory of Gießen. The testing setup is shown in figure 4.1. For this RoI test data is sent to the Merger AMC, then passed through the Merger Carrier and the Selector Carrier, where the data will be discarded to avoid a bottleneck in the system. The RoI test data is sent as HLT RoIs and is merged with with dummy DATCON RoIs. The Merger AMC and Selector Carrier contain Slow Control logic. They are able to monitor the number of frames and words. A script is used to send data. The script is configurable, i.e. the number of frames and event frequency are selectable.

Concerning the extension of fabric channels of the Aurora core 100 frames were sent and were observed at all monitoring stages of the Merger AMC, Merger Carrier and Merger Selector. Moreover, the corresponding slave register used for the specific fabric channel of the second or third Monitor core displayed the correct number of frames and words. It can be concluded the extension of fabric channels are successfully implemented.

The reset sequence of the Aurora core was tested with a simulation using a test bench. Since simulating a timer of an magnitude of one second takes long, the longest timer was reduced to a magnitude of one micro seconds. The waveform diagram for the `gt_reset` and `reset` signal matches the behaviour depicted in figure 4.2. In addition, the logic was tested on the CNCB v4.0 and the Aurora Channel connection was established.

The resets of the cores was tested by writing the hexadecimal string `0000000A` to the slave register 0. The observation of the slow control system showed that all slave register entries were set back to the original status.

To test the interrupts of the Aurora core, the system is transitioned between an unload and load status. This will reset the Aurora channel connection and occasionally cause an interrupt, that was observed with the slow control software Node. For the Belle II Format Handler core interrupts the format of data must be changed. Since the Merger AMC will always discard corrupted data, changing the RoI test data will not cause an interrupt of the Merger Carrier. By adjusting the DATCON dummy RoIs it is possible to change the format after the Merger AMC discard function. The observation showed that the Merger Carrier will not pass through the RoIs and interrupts are created. In theory every frame should cause one interrupt. The number of interrupts depends on the event frequency, because the operation speed of the Microblaze. A event frequency 100 Hz about 10% of interrupts, at 30 Hz about 90% of interrupts and with a frequency of 5 Hz all interrupts are detected.

5 Summary and Outlook

The Belle II experiment is located at the asymmetric electron-positron collider SuperKEKB at the KEK research facility in Tsukuba, Japan. The Belle II detector is placed at the interaction point to measure particle properties and precise locations of decay vertices. The Online Selection Node (ONSEN) system is part of the data acquisition system of the PXD and is used to reduce the data load. The ONSEN system utilizes Region of Interests (RoIs) created from two different sources. The RoIs are constructed from data taken by other subdetectors and allow for the selection of specific event signatures. This system can be divided into a Merger part, which mergers the different RoI sources, and a Selector part that forwards data depending on its location.

The Compute Node Carrier Board v4.0 (CNCB v4.0) is a new revision and is planed to be used as a spare Merger Carrier. Adjustments must be developed for the new Merger Carrier. Therefore additions to the custom Implementation Program (IP) cores are implemented without changing internal data processing. There are four custom IP cores used in the project: The Aurora core handles data protocol conversion, the Monitor core counts passed though data frames and words, the Belle II Format Handler core verifies data certitude and format and the RoI Distribution core divides data into eight separate data lanes.

With the additions the systems is able to be used in different slots of the ATCA shelf. The custom IP core are now resettable by using the slow control program Node. Moreover the interrupts are read with the Microblaze and forwarded to Node. For better stability a reset sequence logic was implemented to create an ordered reset signal for the high speed links.

Table 5.1 – Summary of additions to the firmware.

Addition	Description
Update interface protocol	The Local Link protocol is replaced by the AXIS protocol.
Fabric channel extension	Fabric channels are added for the Aurora core, so that the CNCB v4.0 can be used in all slots of the ATCA shelf.
Reset sequence logic	A sequenced reset is initialized to prevent MGT errors of the Aurora core.
Core resets	The cores are individually resettable with the slow control.
Interrupts	Interrupts of the Aurora core and the Belle II Format Handler core are processed and are displayed with the slow control.

Concluding, all data handling cores are implemented. A summary of additions is shown in table 5.1. The functionality was tested by sending RoIs test data and confirmed to run stable. Therefore the CNCB v4.0 is almost able to be used as a spare Merger Carrier in the Belle II experiment.

The adjusted firmware was implemented in the CNCB v4.0 of the local laboratory. Although the firmware is stable and applicable not all functions are implemented yet. Continuing the implementation should be realized to prevent errors. The most important functions are listed in the following.

One of the missing features is to fully implement all changes of state, that are requested by the run control. This includes after unloading of the system the cut off of connections and closing of gates. Loading of the system should recreate all connections, opening of the gates, resetting of

all cores and loading of configurations. This will ensure complete deletion of data of the previous data processing.
Another feature is the specific error detection of the Belle II Format Handler core. The number of total interrupts will be displayed, but which specific interrupt caused the error is not. This is not an essential feature. The function can be realized using a Data Mover core, which acts as a FIFO with a width to cover all interrupts. The individual interrupt will be sent to the Data Mover core, where the corresponding register will be read by the slow control.

Future upgrades of the ONSEN system may contain the usage of the CNCB v4.0 as a Selector Carrier, where only minor changes are necessary. If a new revision of AMC is planned, more adjustments are needed. For example a pixel parser must be implemented into the Belle II Format Handler core.

Appendix

A.1 Firmware for the Microblaze

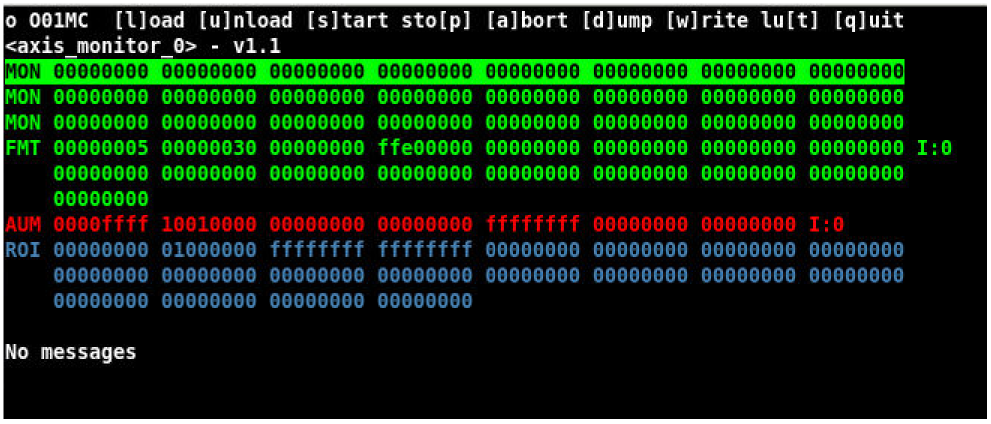
A.1.1 Linux

The Microblaze allows the usage of an Operating System (OS). For the OS, we use Linux as an interface between user and hardware. This allows writing and reading from the memory and addresses that are utilized by the FPGA. The generation of the OS requires a Microblaze core and certain peripheral cores like an interrupt manager.

In order to compile the Linux kernel, it needs to know locations and addresses of the FPGA peripherals. For this, a device tree is generated, which lists components including the IP cores. Furthermore, drivers are created using `tcl` and `mdd` scripts. A cross compiler is used to create the kernel image, which will be later on programmed on the Microblaze. In the compilation process, scripts are involved that run the slow control system Node. For the cross compiler system buildroot is used. Moreover, at the start of the OS a bootloader is utilized [23].

A.1.2 Slow Control

The slow control is used to monitor status and manipulate settings for the operation of ONSEN. There are two slow control systems utilized. Node is an self made script based slow control system used as development tool and for debugging purposes. The other slow control system is EPICs, which employs Process Variables (PVs) and embedding of a graphical interface. A PCB, the Input Output Controller (IOC), attached to the CNCB allows the use of the slow control system.



```
o 001MC  [l]oad [u]nload [s]tart sto[p] [a]bort [d]ump [w]rite lu[t] [q]uit
<axis_monitor_0> - v1.1
MON 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
MON 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
MON 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
FMT 00000005 00000030 00000000 ffe00000 00000000 00000000 00000000 00000000 I:0
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000
AUM 0000ffff 10010000 00000000 00000000 ffffffff 00000000 00000000 I:0
ROI 00000000 01000000 ffffffff ffffffff 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000

No messages
```

Figure A.1 – Interface of the slow control system Node.

Node

The slow control system Node is implemented during Linux kernel compilation. Node detects User Space Input Output (UIO) registered cores and is used to read and write to the content of the slave registers and number of total interrupts. In figure A.1 all slave registers of every UIO core as well as interrupts, if available, are shown. The cores are referenced with abbreviated names, followed by the content of the slave registers in a hexadecimal representation and interrupts [24].

EPICs

EPICs enables the usage of graphical interface Control System Studio (CSS). Similar to scripts of Node, the PVs are defined in scripts that are implemented in the compilation of the Linux kernel. PVs can have different applications. Some of the PVs monitor bits of the slave register and other PVs use scripts to be calculated by an arbitrary input.
EPICs consists of a server interface and a client interface. On the server side, the IOC will provide access to the PV. One or more clients are able to request the content of PVs. The request will send a data packet to the network and searches for the PV at the server interface. After the PV is found, a connection between server and client will be established.

Depending on the type of request the connection will be closed or held [24].

A.2 Hardware Information

A.2.1 Backplane Connections of the ATCA Shelf

There are two numbering systems for the slots of the ATCA shelf: The physical slots numbering, which counts the slots from left to right and the logical slot numbering that is often referred to in development. In the table A.1, the fabric channel connection of the two slots are listed underneath the slots. To connect a CNCB in physical slot 12 to a CNCB in physical slot 2, we use the column of slot 2. Here, we search for the number 12 written on the left side of the number pairs. The corresponding channel of that row is the connection we need to use to connect the CNCB in slot 12 to a CNCB in slot 2. In this case, we need to use fabric channel 11 in the firmware of the CNCB in slot 12.

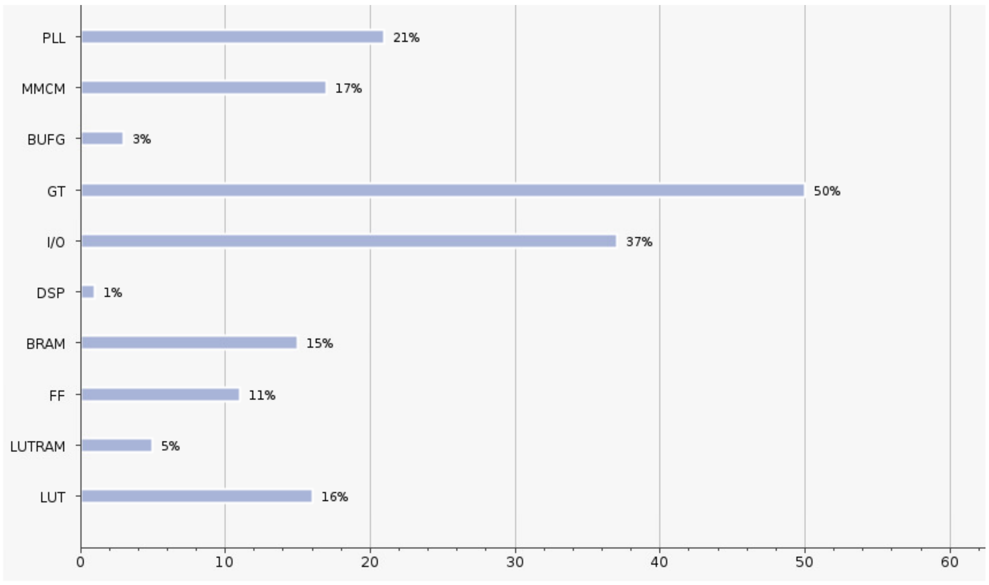


Figure A.2 – Utilization of resources of the FPGA.

Table A.1 – Connection between slots of the ATCA shelf [24].

Physical Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Logic Slot	13	11	9	7	5	3	1	2	4	6	8	10	12	14
Channel														
13	14-13	14-11	14-9	14-7	14-5	14-3	14-1	14-2	14-4	14-6	14-8	14-10	14-12	14-13
12	12-12	13-11	13-9	13-7	13-5	13-3	13-1	13-2	13-4	13-6	13-8	13-10	13-12	12-13
11	11-12	12-11	12-9	12-7	12-5	12-3	12-1	12-2	12-4	12-6	12-8	12-10	11-11	11-13
10	10-12	10-10	11-9	11-7	11-5	11-3	11-1	11-2	11-4	11-6	11-8	11-10	10-11	10-13
9	9-12	9-10	10-9	10-7	10-5	10-3	10-1	10-2	10-4	10-6	10-8	9-9	9-11	9-13
8	8-12	8-10	8-8	9-7	9-5	9-3	9-1	9-2	9-4	9-6	9-8	8-9	8-11	8-13
7	7-12	7-10	7-8	8-7	8-5	8-3	8-1	8-2	8-4	8-6	7-7	7-9	7-11	7-13
6	6-12	6-10	6-8	6-6	7-5	7-3	7-1	7-2	7-4	7-6	6-7	6-9	6-11	6-13
5	5-12	5-10	5-8	5-6	6-5	6-3	6-1	6-2	6-4	5-5	5-7	5-9	5-11	5-13
4	4-12	4-10	4-8	4-6	4-4	5-3	5-1	5-2	5-4	4-5	4-7	4-9	4-11	4-13
3	3-12	3-10	3-8	3-6	3-4	4-3	4-1	4-2	3-3	3-5	3-7	3-9	3-11	3-13
2	2-12	2-10	2-8	2-6	2-4	2-2	3-1	3-2	2-3	2-5	2-7	2-9	2-11	2-13
1	1-12	1-10	1-8	1-6	1-4	1-2	2-1	1-1	1-3	1-5	1-7	1-9	1-11	1-13

Table A.2 – FPGA comparison [25].

	Virtx-4 FX60 (CNCB v3.3)	Virtex-5 FX70T (AMC)	Kintex UltraScale 060 (CNCB v4.0)
Registers	50k	44k	663k
LUTs	50k $\times$ 4-input	44k $\times$ 6-input	332k $\times$ 6-input
DSP Slices	128	128	2760
BRAM	4 Mb	5 Mb	38 Mb
MGT	16 $\times$ 6.5 Gbps	16 $\times$ 6.5 Gbps	32 $\times$ 16.3 Gbps
CPU	PPC405	PPC440	-

A.2.2 FPGA Hardware Specifications

Comparison between the CNCB v3.3, the AMC and the CNCB v4.4 is shown in table A.2. The CNCB v4.4 does not utilize a CPU. Instead, it uses a Microblaze, because the PowerPC insertion is not supported anymore. The operation of the Microblaze is slower, but the monitoring capabilities with EPICs are verified. The percentage of occupied hardware by the firmware is shown in figure A.2.

A.3 Firmware Details

A.3.1 Full Block Design

The block design of the whole project is shown in figures A.3 to A.6.

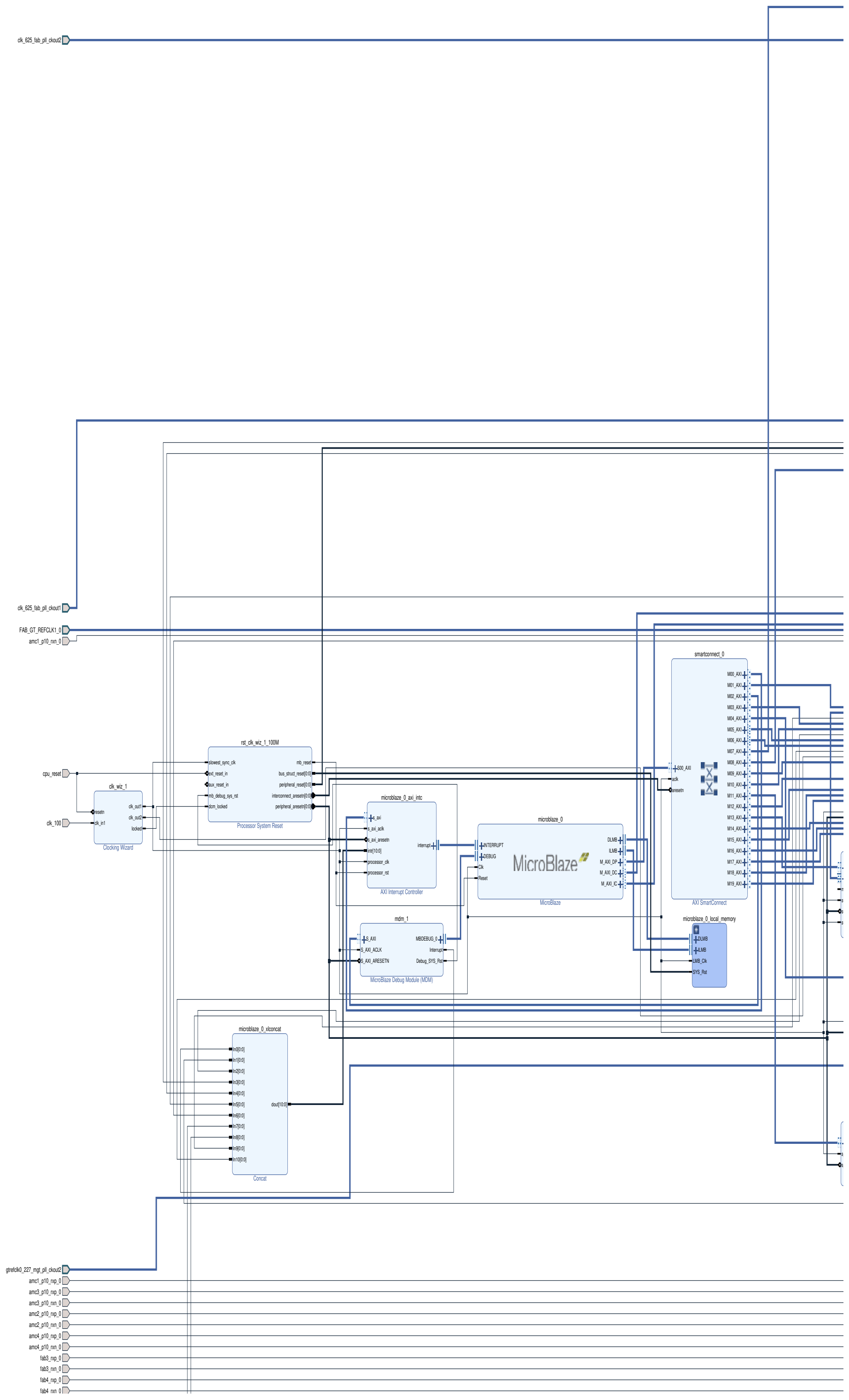


Figure A.3 – Top left side of the block design.

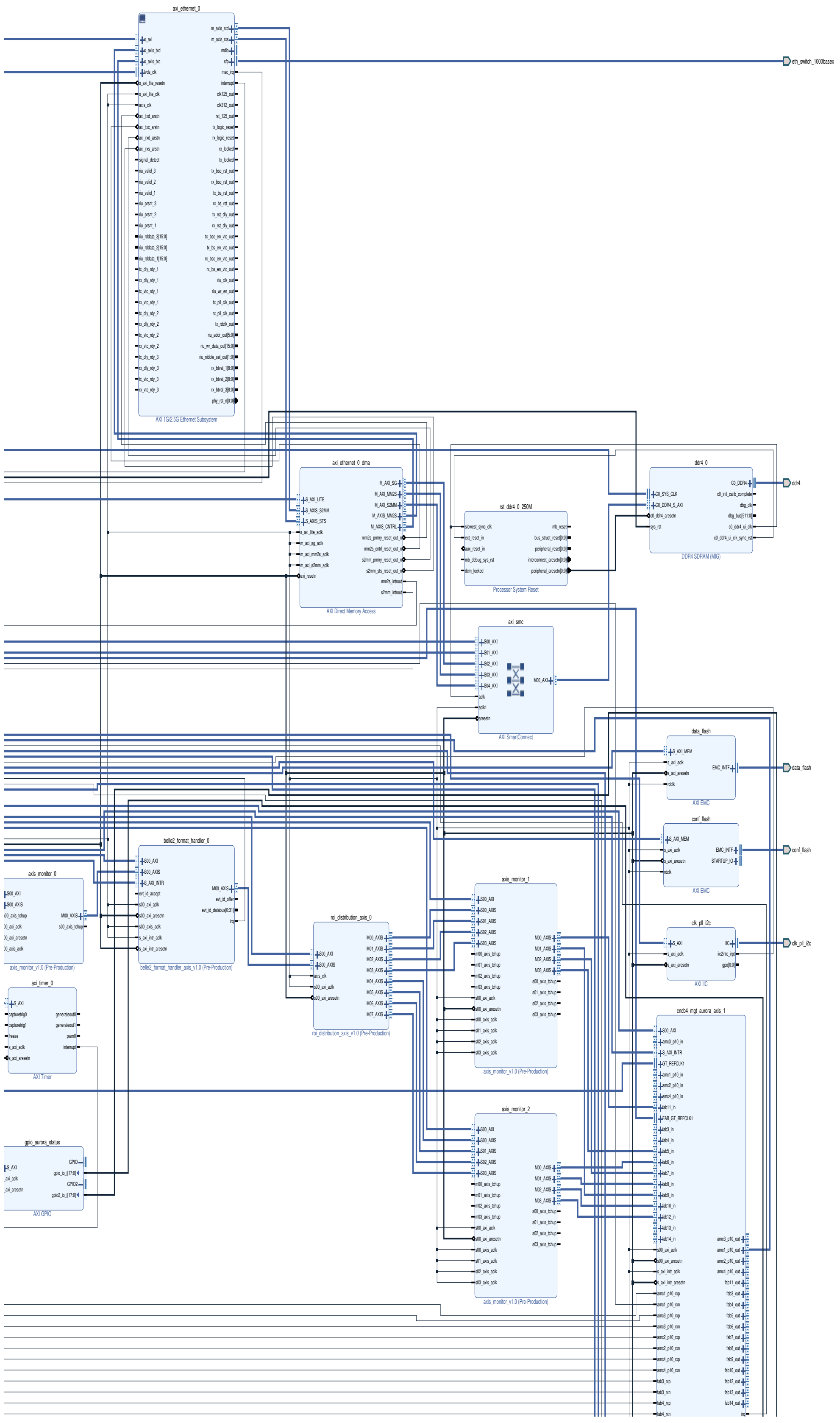


Figure A.4 – Top right side of the block design.

fab5_rxp_0
fab5_rxn_0
fab6_rxp_0
fab6_rxn_0
fab7_rxp_0
fab7_rxn_0
fab8_rxp_0
fab8_rxn_0
fab9_rxp_0
fab9_rxn_0
fab10_rxp_0
fab10_rxn_0
fab11_rxp_0
fab11_rxn_0
fab12_rxp_0
fab12_rxn_0
fab13_rxp_0
fab13_rxn_0
fab14_rxp_0
fab14_rxn_0

Figure A.5 – Bottom left side of the block design.

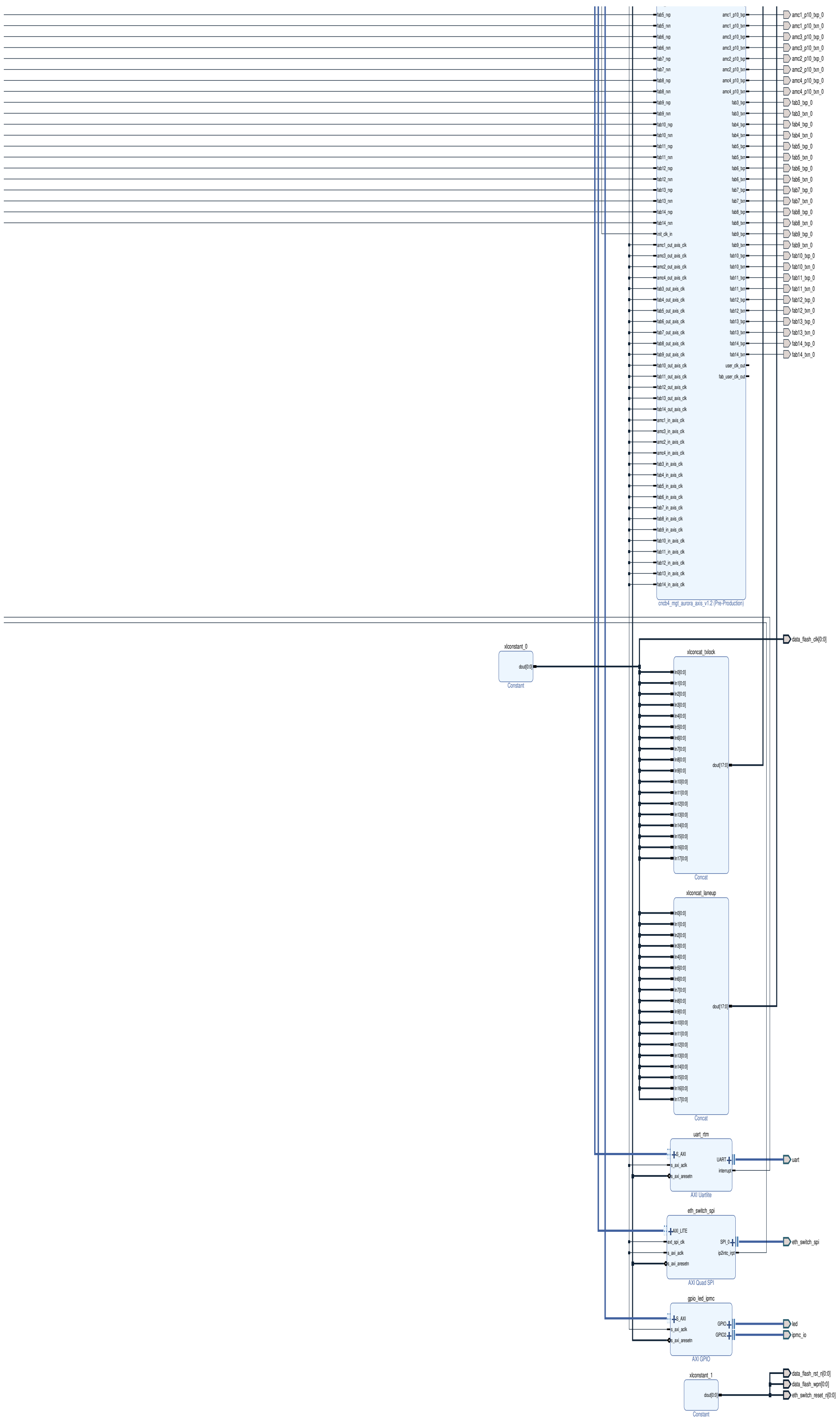


Figure A.6 – Bottom right side of the block design.

A.3.2 RoI Data Format

The data format of a frame is depicted in figure A.7. The data is divided into 32-bit words, that together build a frame. The meta data consists of a header word, trigger information of the HLT and DATCON. Followed by the RoI data, which should always be an even number of RoI words. This is because every DATCON RoI has a HLT RoI partner. At the end of frame is the checksum word.

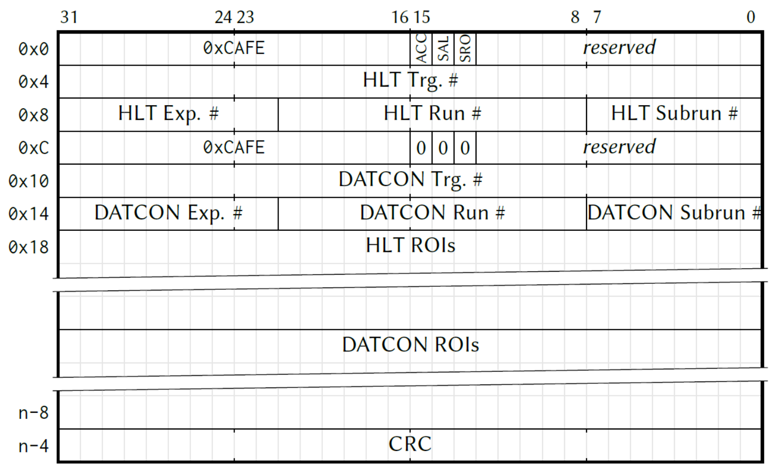


Figure A.7 – Schematic of merged data format [11].

A.3.3 Errors of the Belle II Format Hander Core

The RoI parser of the Format Handler core checks the data format for errors. The list of errors with a description is written in the table A.3.

Table A.3 – Detectable errors of the Belle II Format Handler core [24].

Error	Bit	Description
ERR_BIT_CRC	31	The checksumme is incorrect.
ERR_BIT_FRAME_SIZE	29	The frame size exceeds RoI FIFO depth.
ERR_BIT_MAGIC	28	Incorrect header word.
ERR_BIT_UNEXP_TLAST_0	26	TLAST high inside header.

`ERR_BIT_UNEXP_TKEEP`	25	Wrongly assigned `TKEEP` inside header.
`ERR_BIT_DATCON_ACC`	24	ACC Bit set to high. This error can be only produced, if data source is only DATCON.
`ERR_BIT_MISALIGNED`	23	Uneven number of RoIs.
`ERR_BIT_DATCON_SIZE`	22	Number of RoIs exceeds limit. The limit can be set between 1 and 256. The default limit is 128. This error can be only produced, if data source is only DATCON.
`ERR_BIT_DATCON_TRIG_MM`	21	Trigger number of DATCON and HLT source does not match.

Bibliography

[1] Tom W. B. Kibble. The standard model of particle physics, 2014.

[2] P.A. Zyla et al. Review of Particle Physics. *PTEP*, 2020(8):083C01, 2020.

[3] Paul Langacker. *The Standard Model and Beyond*. Series in High Energy Physics, Cosmology, and Gravitation. CRC Press, second edition, 2017.

[4] Glenn Elert. `https://physics.info/standard/`. Last accessed on 24.11.2021.

[5] Katharina Dort. Search for Highly Ionizing Particles with the Pixel Detector in the Belle II Experiment. Master's thesis, Justus-Liebig-Universität Gießen, 2019.

[6] C. S. Wu, E. Ambler, R. W. Hayward, D. D. Hoppes, and R. P. Hudson. Experimental test of parity conservation in beta decay. *Phys. Rev.*, 105:1413–1415, Feb 1957.

[7] J. H. Christenson, J. W. Cronin, V. L. Fitch, and R. Turlay. Evidence for the 2π decay of the k_2^0 meson. *Phys. Rev. Lett.*, 13:138–140, Jul 1964.

[8] Stephen G. Gasiorowicz and Paul Langacker. Elementary Particles in Physics. University of Pennsylvania, 2005.

[9] Part I Phenomenology of Elementary Particles, 2008. Inproceedings of MAPFis Lecture: Experimental Particle and Astroparticle Physics.

[10] Konrad Kleinknecht and Burkhard Renk. Unitarity Triangle from CP invariant quantities. *Johannes-Gutenberg Universität Mainz*, 2008.

[11] Thomas Geßler. *Development of FPGA-Based Algorithms for the Data Acquisition of the Belle II Pixel Detector*. PhD thesis, Justus-Liebig-Universität Gießen, 2015.

[12] Y. H. Ahn, Hai-Yang Cheng, and Sechul Oh. Wolfenstein Parametrization at Higher Order: Seeming Discrepancies and Their Resolution. *Physics Letters B*, 703:571–575, 2011.

[13] Qiang Li and Qi-Shu Yan. Initial State Radiation Simulation with MadGraph. 2018. Prepared for submission to JHEP.

[14] Michel Bertemes. Dark Sector Searches at Belle II, 2020. Belle II Collaboration.

[15] F. Abudinén, I. Adachi, and P. et al. Ahlburg. Measurement of the integrated luminosity of the Phase 2 data of the Belle II experiment. *Chinese Physics C*, 44(2), 2020. Belle II Collaboration.

[16] Kazunori Akai, Kazuro Furukawa, and Haruyo Koiso. Superkekb collider. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 907, 2018.

[17] T. Abe, I. Adachi, K. Adamczyk, et al. Belle II Technical Design Report, 2010. pp. 76-89, 139-144, 202-208, 220-225, 250-253, 284-285, 313-319, 404-407.

[18] Thomas Geßler. *Private messages*.

[19] Xilinx. *Aurora 8B/10B v11.0 LogiCORE IP Product Guide*, 2016. Vivado Design Suite PG046.

[20] https://logowik.com/usb-symbol-vector-logo-4538.html. Last accessed on 26.11.2021.

[21] https://de.depositphotos.com/vector-images/rj45.html. Last accessed on 26.11.2021.

[22] https://www.allaboutcircuits.com/technical-articles/jtag-connectors-and-interfaces/. Last accessed on 26.11.2021.

[23] Eli Billauer. http://billauer.co.il/blog/2011/08/linux-microblaze-howto-tutorial-primer-1/, 2011. Last accessed on 26.11.2021.

[24] Simon Reiter. *Private messages.*

[25] Thomas Geßler. Compute Node Status and Upgrade, 2017. PANDA DAQ-FEE-Event Filtering Workshop.

Danksagung

Zuallererst möchte ich mich bei Apl. Prof. Dr. Jens Sören Lange bedanken, der mir nicht nur die Arbeit an diesen Projekt ermöglicht hat, sondern auch Perspektiven diese Arbeit fortzusetzen. Weiterhin bedanke ich mich bei meinen Betreuern Dr. Thomas Geßler und Simon Reiter, die mich eingearbeitet haben und mich bei meiner Arbeit immer unterstützt haben. Besonderen Dank möchte ich Simon geben, der sehr ausführlich komplizierte Zusammenhänge erklärt und mich in die Belle II Kollaboration integriert hat. Die Arbeitsatmosphäre der AG Lange ist eine der angenehmsten und freundlichsten, die ich kenne. Ich bedanke mich vielmals bei alle Mitglieder dieser AG. Schließlich möchte ich meinen Dank bei meiner Familie und Freunden ausdrücken. Ich bedanke mich bei meinen Eltern für ihre moralische und finanzielle Unterstützung. Insbesondere danke ich meiner Freundin Franziska und ihrer Familie, die mich ebenfalls in sehr vielen Aspekten unterstützt haben.

Selbstständigkeitserklärung

Hiermit versichere ich, die vorgelegte Thesis selbstständig und ohne unerlaubte fremde Hilfe und nur mit den Hilfen angefertigt zu haben, die ich in der Thesis angegeben habe. Alle Textstellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen sind, und alle Angaben die auf mündlichen Auskünften beruhen, sind als solche kenntlich gemacht. Bei den von mir durchgeführten und in der Thesis erwähnten Untersuchungen habe ich die Grundsätze gute wissenschaftlicher Praxis, wie sie in der ‚Satzung der Justus-Liebig-Universität zur Sicherung guter wissenschaftlicher Praxis' niedergelegt sind, eingehalten. Gemäß § 25 Abs. 6 der Allgemeinen Bestimmungen für modularisierte Studiengänge dulde ich eine Überprüfung der Thesis mittels Anti-Plagiatssoftware.

Ort, Datum	Unterschrift