The Trigger Control System and the Common GEM and Silicon Readout for the COMPASS Experiment

Diploma Thesis

by

Boris Grube

December 2001

Abstract

This thesis describes two basic parts of the data acquisition system of the COMPASS experiment at CERN. Both systems are based on reconfigurable hardware. Their architecture and functionality is the main topic of this thesis.

The first part deals with the Trigger Control System, that distributes the first level trigger and a very precise reference clock over a passive optical fiber network to the frontend electronics of all detectors. In addition the system provides unique event identifiers, that are needed for the event building.

The second part introduces the APV 25 based readout chain for the GEM and silicon detectors. This system also utilizes optical fibers for data transmission and is designed to stand high event rates. Using a special readout mode of the APV 25 frontend chip, that gives three samples per event, a very good time resolution of the detectors is achieved. The high trigger rates require an efficient zero suppression algorithm. The data sparsification, which is performed in hardware, features an advanced common mode noise correction utilizing a combination of averaging and histogramming.

Contents

List of Figures

Chapter 1

Introduction

COMPASS is a new fixed target experiment, that was built to study the structure and spectroscopy of hadrons with the aim to get a better understanding of the properties of quark-gluon-systems. These systems are described by **Q**uantum **C**hromo**d**ynamics (QCD). In contrast to the Quantum Electrodynamics the massless field bosons of the QCD – the gluons – are charged. This causes the non-Abelian structure of the QCD interaction, which makes the calculation of strongly bound states very difficult.

To obtain more information about the structure of the hadrons, the COMPASS experiment will study the scattering of highly energetic beam particles on a stationary target. Because COMPASS wants to study small effects, high statistics and thus high reaction rates are necessary. In addition the desired quantities can only be measured, if the interactions of the beam particles in the target can be reconstructed with an adequate accuracy from the secondary particles, that fly through the detector. This sets high standards on the performance of the detectors.

But not only concerning the requirements for the detectors, the experiment is demanding. Due to the needed high trigger rates also the processing of the huge amount of data, generated by these detectors, is a challenge for the data acquisition as well as for the off-line analysis. COMPASS features a data acquisition, that uses LHC type technologies to allow high trigger rates, minimum dead time and high data rates. To reach high performance and to be at the same time cost-effective, commercial available technologies are used, wherever it is possible. The whole readout is mainly based on optical fiber connections. The readout electronics is build up using **F**ield **P**rogrammable **G**ate **A**rrays (FPGAs)[1], which are a special type of re-

[1]FPGAs are multipurpose circuits that were introduced in 1985 by Xilinx. They consist of a symmetrical array of configurable logic blocks, that are interconnected with a routing network. The logic blocks can perform simple logical operations by using a look-up table that, is based on static RAM. Switching boxes determine, how the routing network connects the different logic blocks. By programming the look-up tables and the switching boxes the FPGA can be adapted to perform specific algorithms. The complexity of the algorithm is only constrained by the available hardware resources (number of logic blocks) and by the required speed. For more details see [Xil01a], [Xil01b],

configurable logic circuits. They give a great flexibility and reduce the development costs, while allowing to design tailored devices for the particular application.

Two components of the COMPASS data acquisition system – the Trigger Control System and the readout electronics for the GEM and silicon detectors – are the topic of this thesis. After a short overview of the COMPASS experiment the general structure of the COMPASS data acquisition is explained. The next chapter focuses on the Trigger Control System. It describes how the system is build up and how the constituents, controller and receiver, work together. It gives also an overview of the main functions and features of the system. The end of the chapter summarizes the performance of the system during the COMPASS year 2001 run and gives an outlook into the future development.

Chapter 5 deals with the readout system for the GEM and silicon detectors. It first gives an overview of the architecture of the system and then explains the particular components by following the data flow from the detector over the frontend chip and the readout electronics to the readout computer, that collects the data. The different processing steps of the data on their way to the readout computer are described. The proximate section shows, how the I^2C standard is used and adapted to allow the configuration of the readout system. The chapter closes with a summary of the performance of the readout chain during the year 2001 run of the experiment and it discusses possible improvements of the system.

[Xil01c] and [Nie00].

Chapter 2

The COMPASS Experiment

The **Co**mmon **M**uon and **P**roton **A**pparatus for **S**tructure and **S**pectroscopy (COMPASS) is a fixed target experiment at the CERN[1] **S**uper **P**roton **S**ynchrotron (SPS). It is a multipurpose, two-stage magnetic spectrometer, designed to operate with muon as well as with hadron beams. The beam particles are produced in a conversion target using the 400 GeV proton beam of the SPS. The hadron beam consists of protons, pions and kaons with a momentum of up to 300 GeV/c. The muon beam is produced by the decay of the secondary pions. The muons can have momenta of up to 190 GeV/c. The variety of the beams reflects the diversity of the physics in the medium to high energy range, intended to explore with this apparatus.

One major part of the physics program, which is described extensively in [Co96], is the direct measurement of the gluon helicity distribution $\Delta G(x)$ using the deep inelastic scattering of a polarized muon beam on a polarized target. This measurement could decide, whether the gluon polarization contributes to the spin of the nucleon.

Another possible contribution to the nucleon spin comes from polarized sea-quarks. This is investigated by the measurement of polarized lambda hyperons in the current fragmentation region of the semi-inclusive deep inelastic scattering of longitudinally polarized muons on an unpolarized target.

With the hadron beam the focus lies on the search for gluonic systems and exotic hadrons and on the spectroscopy of charmed baryons. In gluonic systems the color charge carrying gluons are excited and contribute to the quantum numbers of the particle. Gluonic degrees of freedom together with valence quarks form so called 'hybrids'. Particles, made only out of gluonic degrees of freedom, are called 'glueballs'. There are hints, that such kind of particles do exist, but to find them high statistics and a good reconstruction of their decay is necessary. Exotic hadrons are color neutral objects of more than three valence quarks. Whether bound states of these multi-quark systems are existing, is still not clear.

[1] European Laboratory for Particle Physics, Geneva, Switzerland

The spectroscopy of charmed baryons is interesting, because there is still only little knowledge about this particle family. Especially for the double charmed baryons still no experimental data are existing. The main reason for this is, that the production cross section for these *ccq* baryons is very small.

The third major goal for the hadron beam physics is the study of the mesonic structure using Primakoff scattering. In this type of reaction a real photon is produced by the scattering of a pion on a nucleus via the interchange of a quasi-real photon of the electric field of the nucleus (see [Kuh01]).

The physics program requires to observe rare processes at high statistics. To fulfill this need, the COMPASS spectrometer is designed to stand high particle fluxes of $2 \cdot 10^8$ muons/spill and $1 \cdot 10^8$ hadrons/spill. The detectors will be readout with trigger rates of up to 100 kHz. The high trigger rates require a small dead time for the frontend electronics. Considering that the roughly 250000 channels will produce an event size of about 20 Kbyte, this results in peak data rates of 2 Gbyte/sec, which is of course a challenge for the data acquisition.

In addition precise tracking over a wide momentum range is needed. Therefore the spectrometer has two stages, each equipped with small and large area trackers, electromagnetic and hadronic calorimeters and with muon identification. The trackers are compound detectors, combining large acceptance with high spatial resolution around the beam region. For large angle tracking straw tubes, drift chambers and multi wire proportional chambers are used. In the center of these detectors the small area trackers – micromegas and GEMs – are mounted. The high precision tracking with spatial resolutions down to a few microns is performed by silicon microstrip detectors, placed in the center of the GEM detectors and also stand-alone around the target region.

The whole apparatus is more than 50 m long and over 10 m wide (see figure 2.1). For most of the detectors the readout electronics sits right in their vicinity, so that the data are digitized very close to the frontends. This is an advantage, because digital data are much easier to handle than analog ones. On the other hand this means, that the first level trigger decision and other signals have to be distributed to all readout modules and that the digital data have to be accumulated and transported back to the readout computers. The first mentioned task is performed by the **T**rigger **C**ontrol **S**ystem (TCS, see chapter 4), the latter one by the **D**ata **Ac****q**uisition System (DAQ, see chapter 3). Special care has to be taken of the grounding of the electronics. This is one of the reasons, why both systems – TCS and DAQ – are based on insulating optical fiber connections.

The TCS has to guarantee a trigger latency of smaller than 1.4 μs (depending on the fiber length). For timing measurements in the spectrometer the different detectors have to be synchronized. Therefore the TCS provides a precise clock signal with a jitter below 50 ps RMS. The main challenges for the DAQ are high data throughput and very low dead time. This requires fast and efficient readout electronics, like for example the readout for the GEM and silicon detectors (see chapter 5).

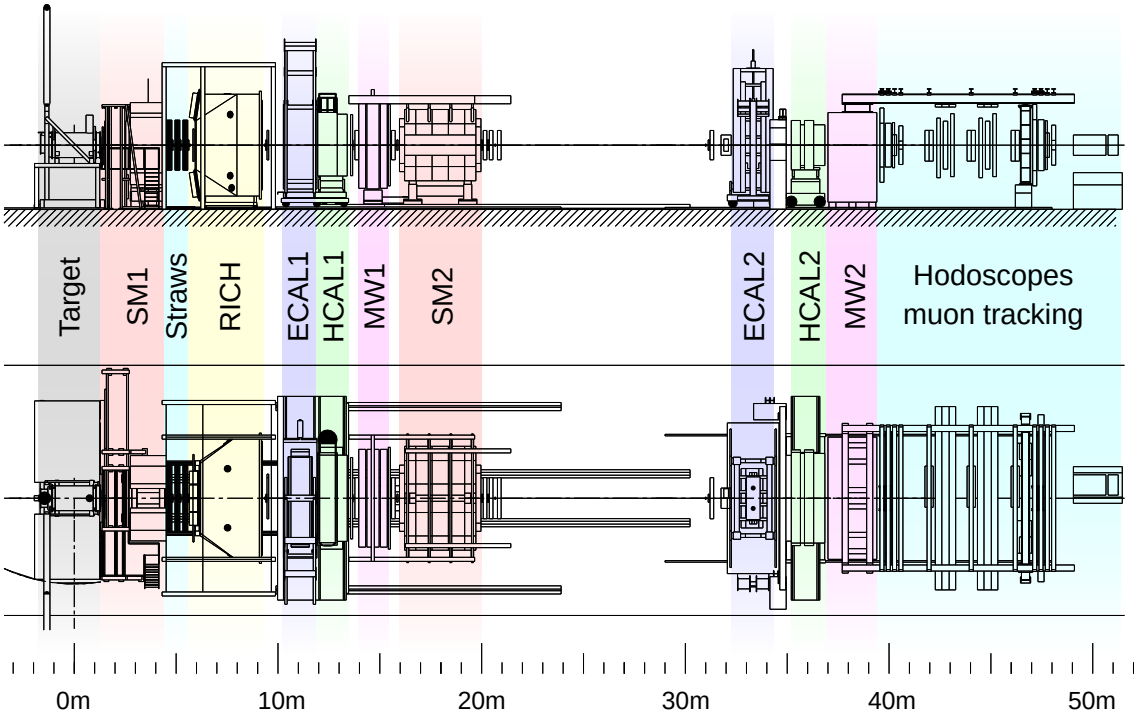


Figure 2.1: Side and top view of the COMPASS experiment (not all parts are labeled) [Kuh01]

An important fact for the operation and the readout of the COMPASS detector is, that the SPS accelerator does not generate a continuous particle beam. The machine works in a regular duty cycle of filling the accelerator with particles, accelerating these particles and finally extracting them in the form of a high energy beam (see figure 2.2). These particle extractions are called 'spills'. In the year 2001 run there was a 5.1 sec long time of extraction (on-spill time) followed by an 11.7 sec long break without beam (off-spill time). The whole data acquisition system runs synchronously to the SPS duty cycle.

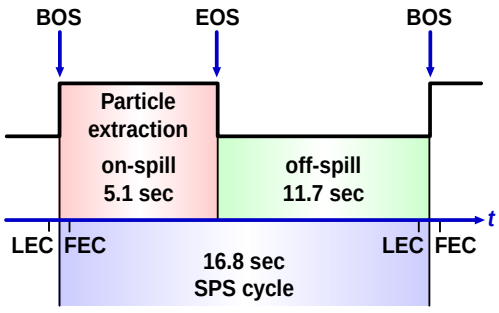


Figure 2.2: The SPS duty cycle and the spill structure.

Chapter 3

The COMPASS data acquisition system

The COMPASS data acquisition is a system designed to stand high trigger rates of up to 100 kHz and high data rates of up to 2 Gbyte/sec. To allow such a high performance, the data are digitized and concentrated as close to the frontend electronics as possible and a pipelined architecture is used. Figure 3.1 shows a schematic view of the system.

In general pipelines are sequential arrangements of processing units, in which each unit passes its results to the next one. In this way the processing units can work in parallel like in an assembly line. The frontend electronics for instance uses pipelines made of memory cells (see figure 3.2). With every clock cycle new data arrive on the input side of the pipeline. To prevent the old data from being overwritten, they are shifted one memory cell further. By doing that, the value stored in the last memory cell of the pipeline is of course discarded. Altogether the memory cells of the pipeline store the complete history of the incoming values of a certain time interval. In the frontend electronics these pipelines are used to bridge-over the time, until the first level trigger decision is made. This means, that the detectors are read out with a constant frequency and the trigger only marks the data in the pipeline, that have to be read out. To distinguish the data belonging to different events, they have to be marked with an unique identifier, which is one of the main tasks of the Trigger Control System. Other pipelines in the readout system help to de-randomize the data flow and to speed up the data processing. The de-randomization is needed, because the COMPASS beam has no special structure, that defines a certain timing between the events. Data buffers help to even out these fluctuations.

The data of the detector frontends are processed by the readout modules CATCH and GeSiCA. They multiplex the data and merge them with the event header delivered by the Trigger Control System. The event header information is later used to assemble the data from the different detectors to complete event data blocks. The readout modules furthermore bring the TCS reference clock to the frontends

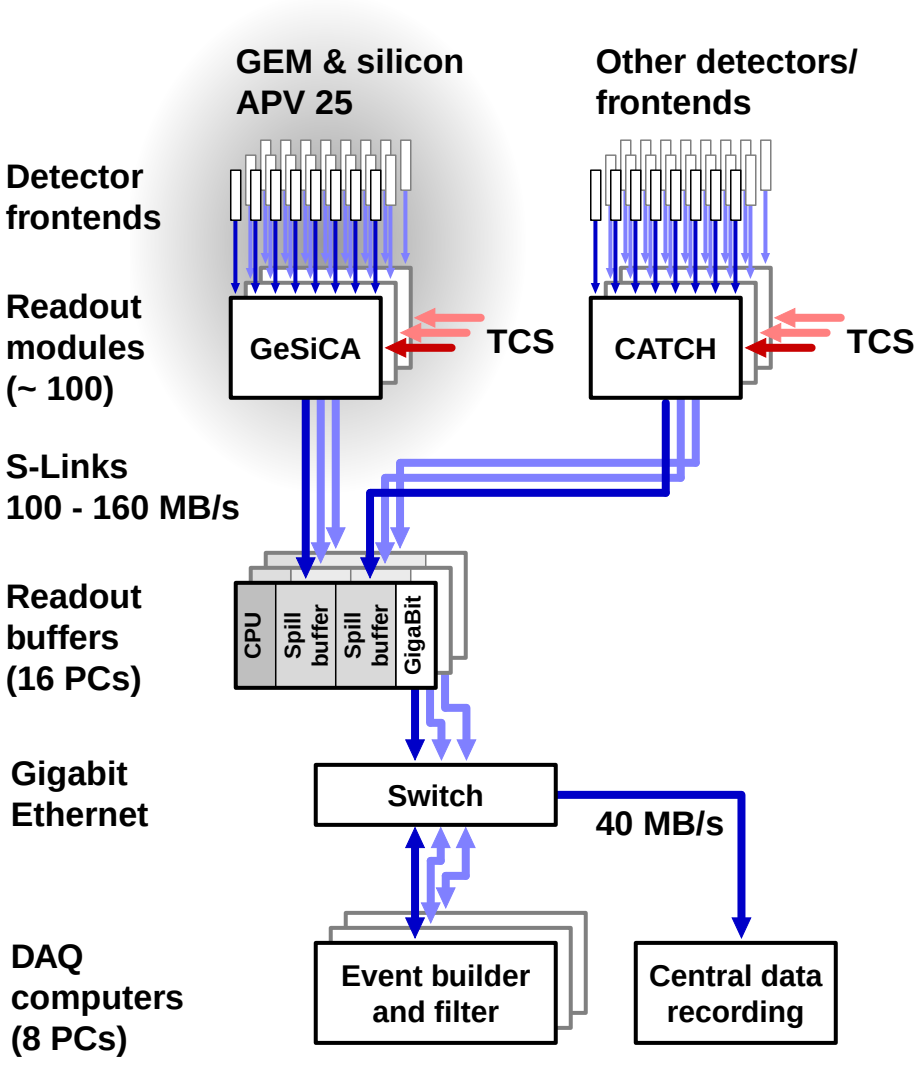


Figure 3.1: Schematic view of the COMPASS data acquisition system

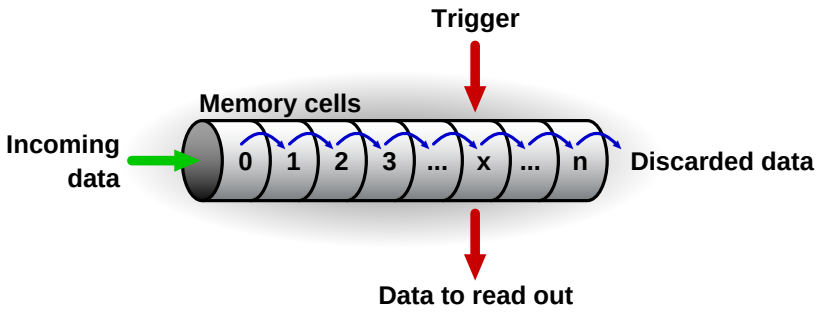


Figure 3.2: Memory pipeline used in the frontend chips to store the incoming data, until a first level trigger decision is made. The trigger pulse marks the memory cell to be read out.

and configure the frontend electronics. The TCS clock synchronizes the readout electronics of the detectors and serves as the reference for all time measurements.

The CATCH (**C**OMPASS **A**ccumulate **T**ransfer and **C**ache **H**ardware) is the general readout module in COMPASS and is used by nearly all detectors. The module is a development of the Universität Freiburg and is the central part of a flexible readout system, supporting a wide range of frontends. In contrast to this the GeSiCA (**GE**M **Si**licon **C**ontrol and **A**cquisition) module is optimized for the readout of the APV 25 frontend chip used by the GEM and silicon detectors.

The data stream generated by the readout modules has to go to the readout computers. In COMPASS optical S-Link[1] connections with a bandwidth of 100 to 160 Mbyte/sec are used for this purpose. The **R**ead**o**ut **B**uffers (ROBs) process the S-Link data. They are equipped with up to four so called 'Spill Buffers'. The Spill Buffer is a PCI[2] card, that holds a standard SDRAM DIM module[3] of up to 512 Mbyte capacity. The RAM acts like a huge FIFO[4] and is able to store the data of at least one spill. The caching of the data allows to make efficient use also of the off-spill time when no physics events are generated. The data rate is averaged over the whole SPS cycle, so that the peaks in the data rate, created during the on-spill time, which is only 30 % of the whole SPS cycle, are lowered. The spill buffers altogether transform the peak incoming data rate of 2 Gbyte/sec during the on-spill time into a continuous outgoing data flow of about 600 Mbyte/sec.

A Gigabit Ethernet[5] switch connects the ROBs with the Event Builders. An Event Builder computer collects all the data fragments of one event from the different ROBs and merges them into an event data block. This is done using the unique event identifier provided by the TCS receiver at the readout modules. The assembled event data blocks go then to the central data recording were they are written to tape. The connection to the central data recording has a bandwidth of maximum 40 Mbyte/sec. Therefore it is planned, to pre-process the data by an online filter, before they are written to tape. The online filter is a software, that will run on the event builder PCs. It will perform a rough tracking and will scan the events for

[1]The **S**imple-**Link** is an interface specification for data transfer from the frontend electronics to the readout electronics developed by CERN. The S-Link specification does not describe the physical link itself, so that there are several implementations using cables as well as optical fibers. See [Bij01]

[2]The **P**eripheral **C**omponent **I**nterconnect (PCI) bus is an industry standard maintained by the PCI Special Interest Group [Pci01]. Nowadays nearly all Personal Computers use this bus to connect the CPU to I/O devices like mass storage controllers, network adapters, graphics cards and so on.

[3]The **S**ynchronous **D**ynamic **R**andom **A**cess **M**emory **D**ual **I**n-Line **M**emory **M**odule (SDRAM DIMM) is an industry standard developed by the **J**oint **E**lectron **D**evice **E**ngineering **C**ouncil (JEDEC) [Jed01]. Memory compliant to this specification is widely used in personal computers and thus cheap and easy to get.

[4]The **F**irst **I**n **F**irst **O**ut (FIFO) is a special form of a data buffer, that conserves the order, the data are coming in. It is usually used to decouple data processing units.

[5]Gigabit Ethernet is a standard developed by the **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers (IEEE) [Iee01] for **L**ocal **A**rea **N**etworks (LANs). It is already quite widely used, so the necessary components are available for reasonable prices.

interesting topologies. To be able to run with a trigger rate of 100 kHz, the online filter has to reduces the amount of data by a factor 10.

Chapter 4

The Trigger Control System

The Trigger Control System of the COMPASS experiment is used to distribute triggers and control information to the readout modules CATCH and GeSiCA. The system is connected to around 100 destinations which are scattered 50 m along the beam line. It is based on a passive optical fiber network with a powerful single laser source and passive optical 1 to 32 tree couplers which split the light to 32 fibers. There are two levels of optical splitters so that in principal up to 1024 destinations can be connected to the system. The laser source is modulated with a frequency of 155.52 MHz.

The main functions of the TCS are:

- Distribution of the **F**irst **L**evel **T**rigger (FLT) signal with a latency smaller than 1.4 μs (depending on the distance to the destination).
- Distribution of the spill number, the event number and the trigger type for every broadcast FLT. The event builders use this information later to assemble the events.
- Provision of a time reference for all detectors with a precision of better than 50 ps RMS.
- Generation of the dead time by setting the BUSY signal in order to prevent the processing of triggers, which cannot be accepted by some parts of the apparatus.
- Generation and distribution of calibration and monitoring triggers, which are (if needed) preceded by a pre-trigger pulse.
- Activation/deactivation of different detectors or parts of detectors in data taking.
- Support for concurrent but independent operation of up to eight DAQs for debugging purposes (MultiDAQ mode; see section 4.5).

The TCS works synchronously to the SPS duty cycle (see figure 2.2). The **b**egin **o**f the **r**un (BOR) and **e**nd **o**f the **r**un (EOR) are performed at the **b**eginning **o**f the **s**pill (BOS). The events are counted inside the SPS cycle, which starts at the beginning of the spill and ends at the beginning of the next one.

4.1 Architecture and system overview

The COMPASS TCS consists of four parts:

- The TCS server software
- The TCS controller
- The TTC laser crate with the passive light distribution system [TTC00]
- The TCS receivers

The basic architecture of the TCS, and how it is embedded into the readout chain, is shown in figure 4.1.

The TCS controller and the TTC laser crate are located very close to the rack of the trigger logic to minimize the trigger latency. The first level trigger, which is generated by the trigger logic, goes to the TCS controller. For every FLT the TCS controller generates spill number, event number and trigger type, which altogether form the event header. To make the trigger latency as small as possible, the first level trigger and the belonging event header are broadcast through two independent data channels A and B. This decouples the distribution of the FLT from the distribution of the belonging event header. Each of the channels has a bandwidth of 38.88 MBit/sec. From the controller channel A and B go via LEMO cable to the TTC crate, where they are multiplexed and encoded to one serial 77.76 MBit/sec data stream, which is sent out to the passive optical fiber network. The network is driven by a single powerful laser diode, that sits in the TTC crate. The light of this diode is split by passive components, so that it can be distributed to up to 1024 destinations. The TTC laser crate is a development of the RD12 collaboration at CERN, which builds the **T**iming, **T**rigger and **C**ontrol (TTC) systems for the LHC detectors. It was modified to run with the COMPASS 38.88 MHz clock used throughout the whole experiment.

The TCS receiver decodes the information from the optical link, recovers the clock signal and sends both via VME P2 connector to the destination module (CATCH or GeSiCA). Every TCS receiver can be configured via the TCS controller using special addressed commands. These commands are generated by the TCS server program, which runs on an online computer and interfaces to the TCS controller using the VME bus. The TCS server also initiates the generation of calibration and monitoring triggers by writing a request for a certain trigger type into the TCS controller.

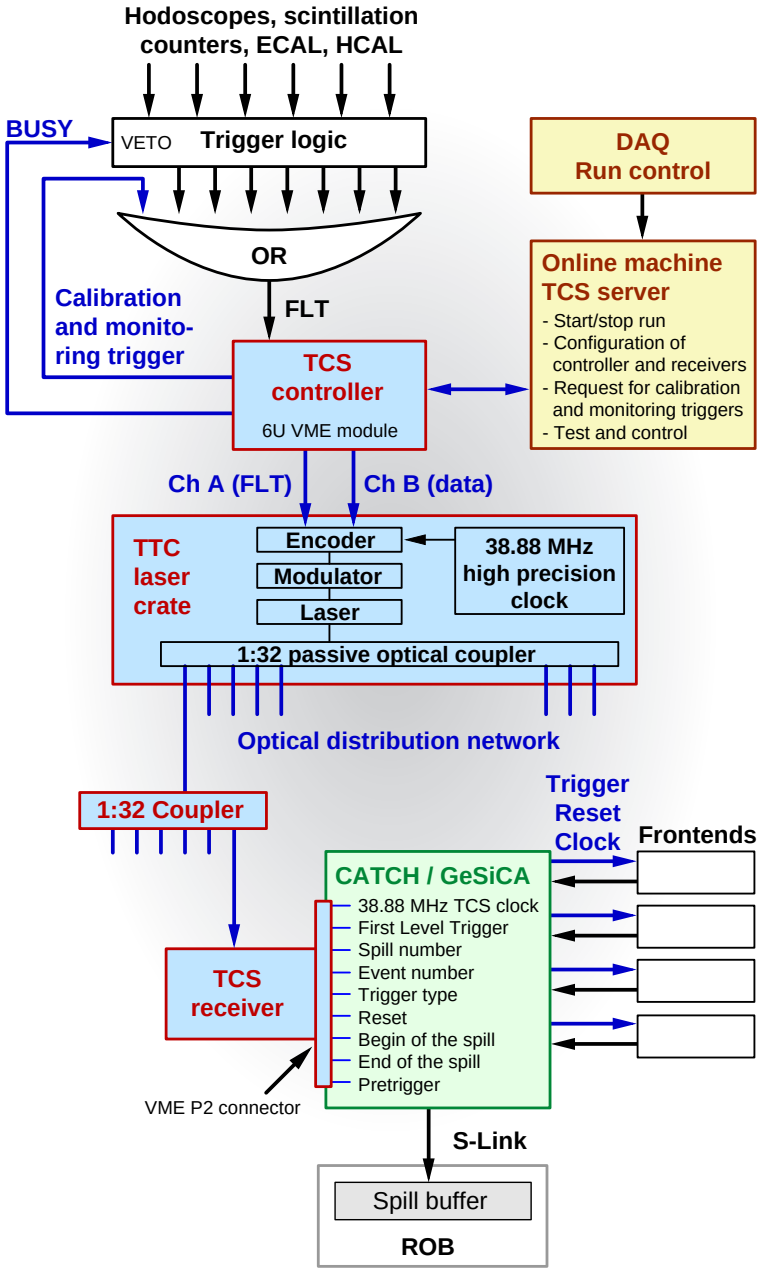


Figure 4.1: The architecture of the COMPASS Trigger Control System

4.2 Encoding and general data format

As shown in figure 4.1, the laser modulator is driven by an encoder, which is phase-locked to a 38.88 MHz clock and linked to the channels A and B of the TCS controller. The TCS has to deliver data and signals to many destinations. An important reduction in the cost, power consumption, size and mass of the TCS receiver is achieved by encoding these signals in such a way, that they can be transmitted over a single optical fiber, so that only one optoelectronic receiver per destination is needed.

The encoding method used by the TTC system has two data channels, which are **T**ime-**D**ivision-**M**ultiplexed (TDM) and encoded biphase mark[1] at 155.52 MBit/sec. This is a rate of the standard Sonet OC-3 (CCITT SDH STM-1)[2]. The four possible symbols, that can be transmitted in each period of the 38.88 MHz TCS clock, are shown in figure 4.2. The DC offset is well bounded and the signal transitions for timing reference are generated at the start and the end of every 12.86 ns interval.

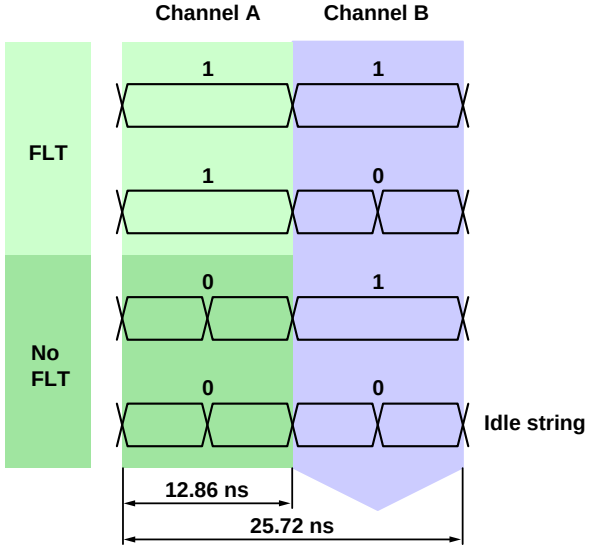


Figure 4.2: The four symbols of the 2-Channel-Time-Division-Multiplexed encoding used by the TTC

The TDM A channel, which is designed for minimum latency, is dedicated to broadcast the first level trigger signal, delivering a one bit decision every 25.72 ns. The distribution of the first level trigger is completely independent from the broadcasting of the belonging event information, which is done by channel B. The data on the B channel are sent in the form of so called 'commands'. There are two types of commands: broadcast and addressed commands (see section 4.9). The broadcast commands are processed by all receivers, whereas the addressed commands only by those with the right TCS receiver ID. The broadcasts are used to distribute the event

[1]The biphase mark encoding uses signal transitions instead of signal levels to transmit '1's and '0's. There are regular, fixed signal transitions, which serve as reference for the clock recovery. An additional signal transition in-between these fixed transitions is interpreted as a '0', if there is no transition a '1' was sent.

[2]The **S**ynchronous **O**ptical **Net**work (SONET) is a transmission architecture, from which in 1988 the **C**omité **C**onsultatif **I**nternational **T**élégraphique et **T**éléphonique (CCITT; later renamed to **I**nternational **T**elecommunications **U**nion - **T**elecommunications **S**tandardization **S**ector (ITU-TSS)) derived the **S**ynchronous **D**igital **H**ierarchy (SDH) standard. SDH defines a transmission system, which can be used as a physical transport medium in networking applications. 'Synchronous' means, that all transitions of the digital signals occur exactly at the same rate and that all clock signals in the synchronous network are derived from one primary reference clock. The **O**ptical **C**arrier Level 3 (OC-3) and the **S**ynchronous **T**ransport **M**odule Level 1 (STM-1) are building blocks of the SDH, describing the transport of data at a rate of 155 MBit/sec over an optical fiber.

header information and to synchronize the readout modules with the spill structure. With the addressed commands the TCS receivers can be configured. The general data format of the commands is shown in figure 4.3. Every command begins with a start bit, which is needed for synchronization on the data level. It is followed by the command identifier (two or four bits), the data and the 6 bit ECC checksum.

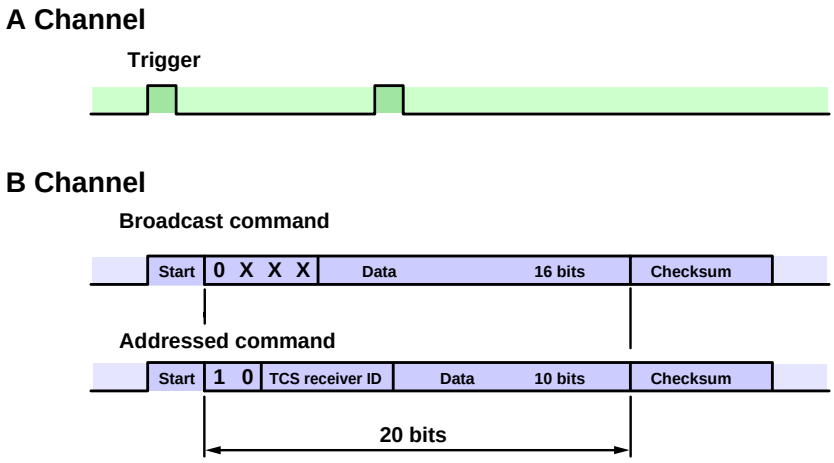


Figure 4.3: Typical examples for data transferred on channel A and B

4.3 The TCS controller

The TCS controller is the central part of the Trigger Control System. It processes the incoming first level triggers and generates the information, which is distributed through the optical fiber network to the TCS receivers. The data are flowing only from the controller to the receivers. The controller also provides a VME interface for the TCS server software. Through this interface the TCS server can start and stop the run, configure the TCS and read-back status information. Figure 4.4 shows a schematic of the TCS controller.

The main task of the TCS controller is the processing of the first level trigger signal, so that it can be broadcast to all receivers and the connected readout modules. Therefore the FLT pulse is synchronized to the 38.88 MHz TCS clock and sent out through channel A of the optical fiber network.

The next important information, which the controller has to generate for every trigger, is the event header. It is formed out of the 20 bit event number and the 5 bit trigger type. The event number is generated by a counter, which counts the FLTs inside one SPS cycle, starting at the beginning of the spill. The first event in a cycle has the number 1. The event counter is reset by the BOS signal. The trigger type allows to distinguish between physical, calibration/monitoring and special triggers (see section 4.7). Both the event number and the trigger type are broadcast to

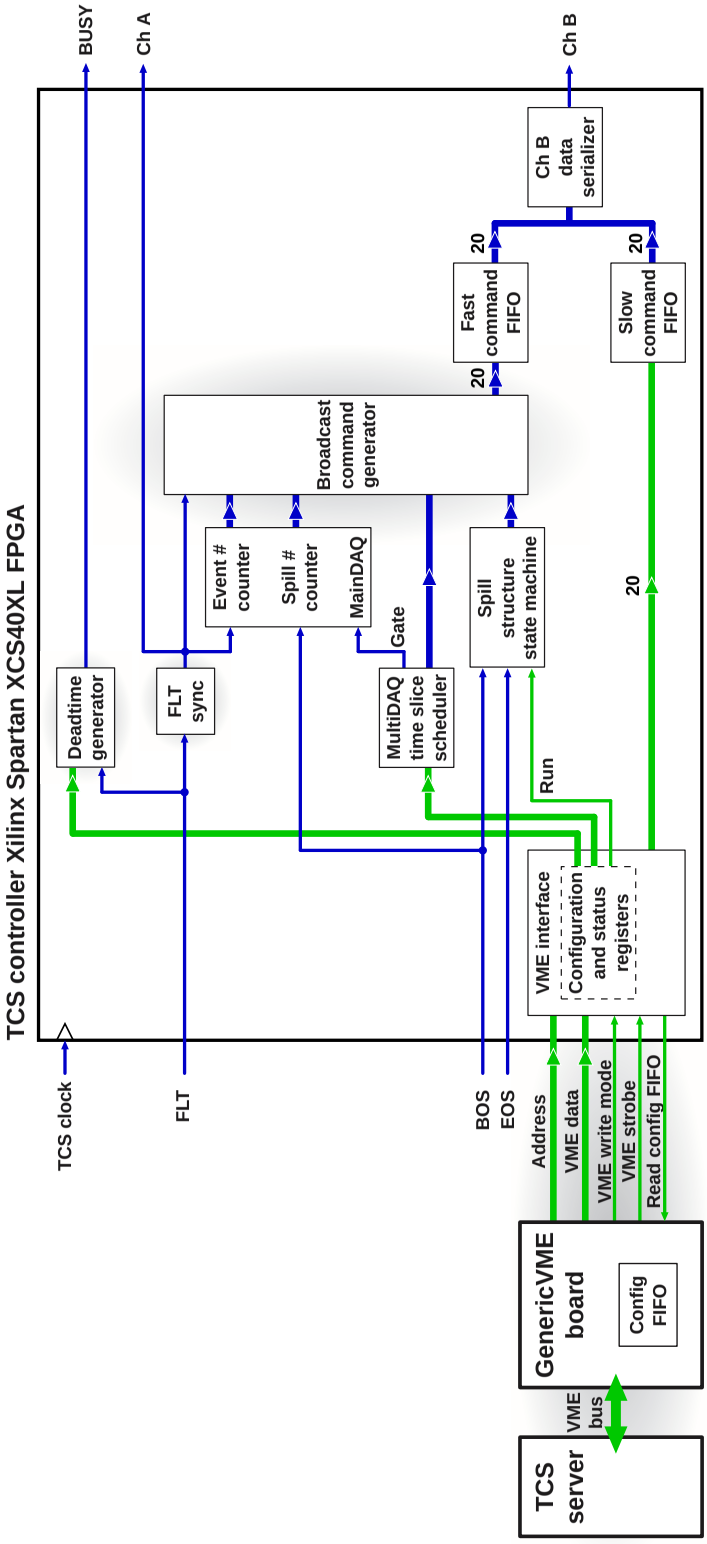


Figure 4.4: A schematic view of the TCS controller showing the basic functional units and the connection to the TCS server.

all receivers. To make the event header unique within a run, the TCS controller additionally generates the spill number. This 11 bit value counts the SPS cycles in the run. The first spill in a run has spill number 1. The spill number is distributed at the beginning of the spill simultaneously with a reset signal for the readout modules CATCH and GeSiCA (see section 4.8). The TCS receivers reformat the event header data and write them out to the readout modules. These modules merge the event header information with the appropriate event data, so that the data fragments from the different detectors can be later assembled to complete events by the event builders.

All broadcasts generated by the TCS controller are buffered in the so called 'fast command' FIFO, before they are sent out to the TTC laser crate. In addition to the fast command FIFO there is a 'slow command' FIFO, which is directly connected to the VME interface. The slow command FIFO has a lower priority (compared to the fast commands FIFO) and is only read out, when the fast command FIFO is empty. The TCS server writes addressed commands to configure the TCS receivers and request commands, which initiate the generation of calibration and monitoring triggers, into the slow command FIFO. Section 4.9 gives a reference on all TCS commands.

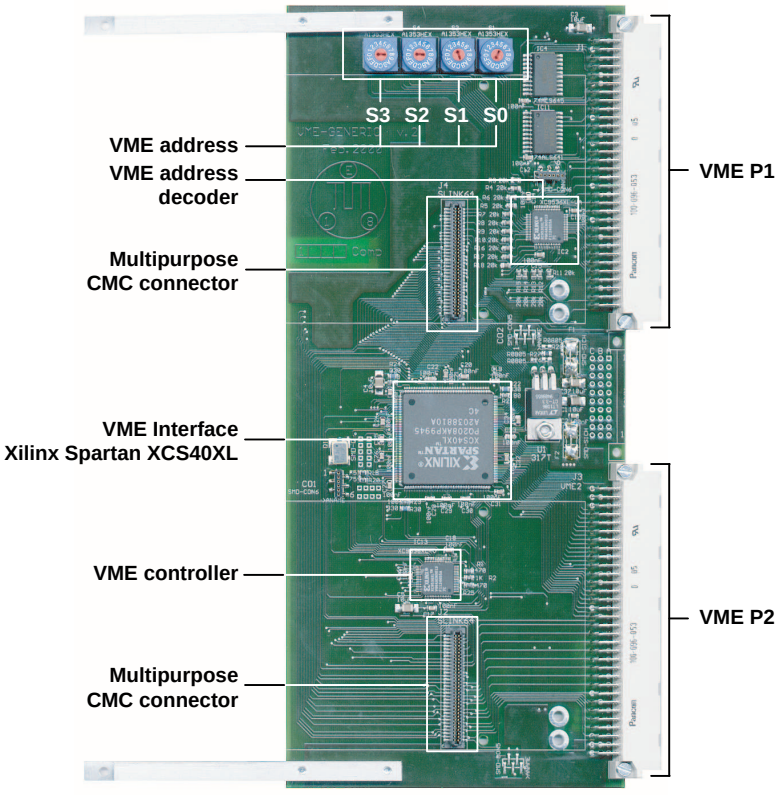


Figure 4.5: The GenericVME board

The controller also generates the dead time for the detectors by asserting the BUSY signal. The BUSY signal vetoes the first level trigger in the trigger logic. This prevents the processing of triggers, that cannot be accepted by parts of the spectrometer. For more details about the dead time generation see section 4.6.

In the MultiDAQ mode, which can be used for debugging purposes, the controller can allocate the first level triggers to up to eight independent DAQ systems. This mode allows to readout different detectors or parts of detectors by logically independent DAQs, that cannot interfere. The system broadcasts the triggers to a particular DAQ (which means to the readout modules that are assigned to this DAQ) in a way, that these triggers do not reach the readout modules, that are connected to the other DAQs. The different DAQs share the trigger and the TCS hardware using time division. The different DAQs rotationally get short time intervals, called 'time slices', during which they have exclusive use of the system. In the MultiDAQ mode one of the DAQs has the status of the so called 'MainDAQ', which means, that for this DAQ event and spill number are generated inside the TCS controller, whereas for the other parasitic DAQs the counting of the events and the spills is done locally in the TCS receivers. A more detailed description of the MultiDAQ mode is given in section 4.5.

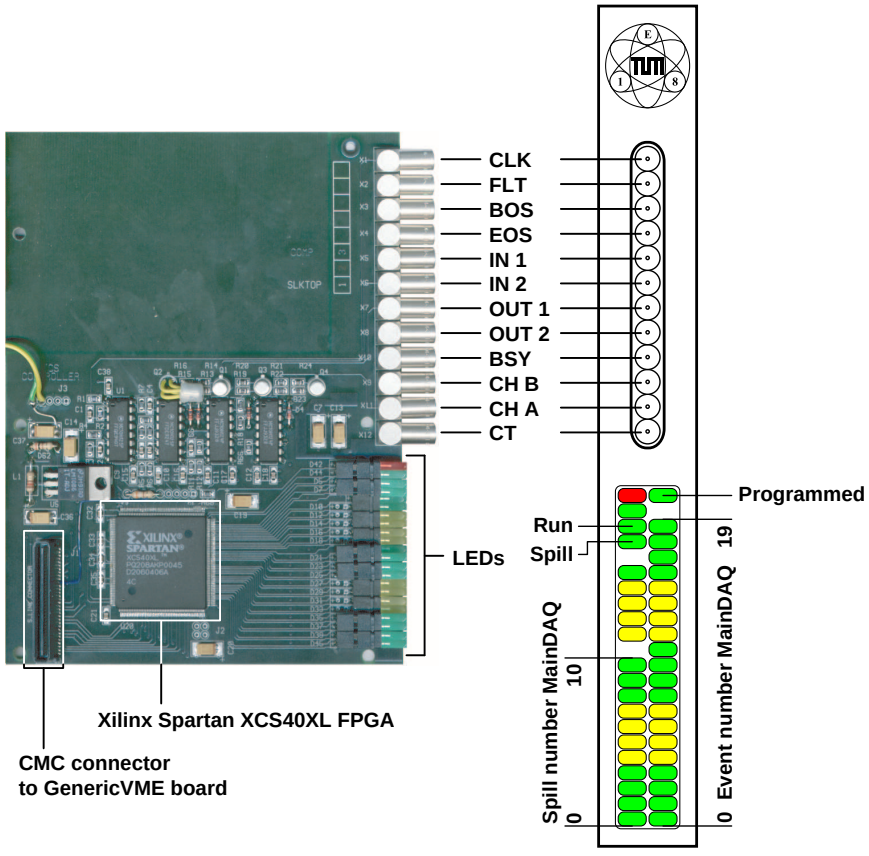


Figure 4.6: The TCS controller with its front panel

The TCS controller itself is a add-on card for the GenericVME board. The GenericVME board is a versatile 6U VME module, that provides the logic for the VME access (see figure 4.5). The controller (see figure 4.6) is plugged into one of the two CMC connectors of the GenericVME board.

The controller uses the VME interface provided by the GenericVME board to communicate with the TCS server. The interface between the TCS controller and the TCS server consists of seven 20 bit wide VME registers. The most important register is directly connected to the slow command FIFO. Whatever the TCS server writes into this register, will be sent out by the controller to the TCS receivers. Two registers are used to configure the fixed and the variable dead time parameters (see section 4.6).

The next two registers control the time slices of the MultiDAQ mode. They also start and stop the TCS and the DAQs. Each register stores four 5 bit values, which define the size of the time slices of four DAQs. Whenever at least one of these time slices is set to a non-zero value, the TCS will start running. If all time slices are set to zero, the TCS will stop. DAQs can be started and stopped by setting their time slices to non-zero values, regardless whether other DAQs are running in parallel or not (see section 4.5).

The sixth register is a status register which is read by the TCS server. It contains two bits which indicate, whether the TCS is running and whether there is on-spill or off-spill time. Using these bits the TCS server can follow the spill structure and for example request calibration and monitoring triggers during off-spill time. The software can also access the spill number of the MainDAQ.

The seventh register allows the TCS server to block the incoming first level triggers. With this feature it is possible to pause and continue a run.

Signal	**Act. H/L**	**I/O**	**Type**	**Short description**
CLK	-	I	NIM	Clock for controller, usually from TTC crate
FLT	L	I	NIM	First Level Trigger input from trigger logic
BOS	L	I	NIM	Begin of the spill signal
EOS	L	I	NIM	End of the spill signal
IN1	L	I	NIM	Spare input
IN2	L	I	NIM	Spare input
OUT1	-	O	NIM	Clock signal from GenericVME board
OUT2	H	O	NIM	Spare output
BSY	H	O	NIM	BUSY signal to veto trigger logic
CHB	-	O	ECL	Channel B output for TTC crate
CHA	-	O	ECL	Channel A output for TTC crate
CT	H	O	NIM	Separate output for artificial triggers

Table 4.1: The LEMO in- and outputs of the TCS controller.

Beside the VME interface the TCS controller has 12 LEMO connectors for the input and output signals. They use NIM[3] and ECL[4] as signal formats. Table 4.1 summarizes all connectors.

4.4 The TCS receiver

The TCS receivers are connected via the VME P2 backplane to the readout modules CATCH or GeSiCA. The receivers process the data sent by the TCS controller through the optical fiber network.

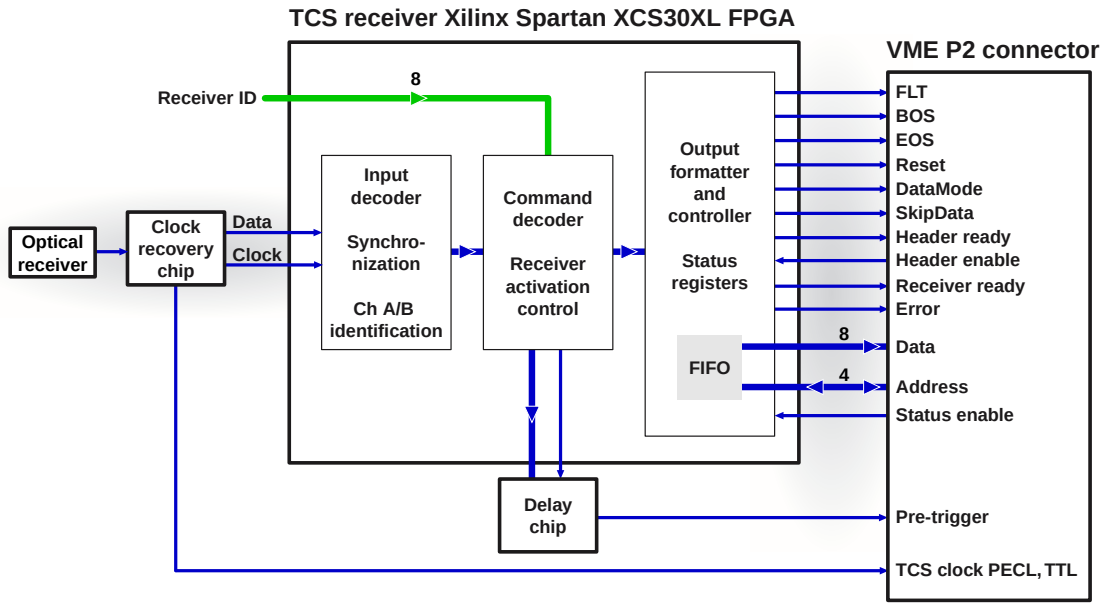


Figure 4.7: A schematic view of the TCS receiver

To do this the receiver first extracts the 38.88 MHz TCS reference clock from the signal of the optical receiver, which is done by a clock recovery chip. Inside the chip a **P**hase **L**ocked **L**oop (PLL) tries to match the frequency of an internal **V**oltage **C**ontrolled **O**scillator (VCO) with the one of the input signal. Assuming that the frequencies differ only very little, the PLL compares the phases of the fixed signal transitions that appear every 12.86 ns and the data transition in between with the phase of the 155.52 MHz clock generated by the internal oscillator. Phase differences are transformed into an error signal, that shifts the frequency of the VCO closer to the one of the input signal. Because the error signal can change the frequency only slightly, the system behaves inertial to short term fluctuations of the input phase, so that after the system has locked onto the input signal, the clock of the VCO is stable and has a low jitter.

[3]The **N**uclear **I**nstrument **M**odule (NIM) logic is a standard for fast digital signals
[4]The **E**mitter **C**oupled **L**ogic (ECL) is a standard signal format for differential digital signals

The TCS clock is used by the receiver itself and by the readout modules which forward the clock signal to the frontend electronics. Because the TCS clock serves as the global timing reference for the whole experiment, the quality of the signal has to be as good as possible. Especially the jitter of the clock should be below 50 ps RMS. To achieve this, the performance of several clock recovery chips was tested. As the result the CLC016AJQ chip from National Semiconductor was chosen. Because the jitter of the clock recovery chip is very sensitive to the noise on the power lines, the optical receiver and the clock recovery chip are powered through a DC-DC converter, which decouples them from the power supply of the rest of the board.

The receiver synchronizes to the optical link on the channel and the data level. It identifies and decodes the two time division multiplexed channels A and B. This identification relies on the fact that channel A transports only the first level trigger signal, whereas channel B transmits all the broadcast and addressed commands. This means, that there are much more '1's on channel B than on A. To allow the identification even when the system is idle, the TCS controller in this case regularly generates so called dummy commands, which just produce some data transfer on the B channel, but otherwise are ignored by the receivers. To synchronize the receiver on the data level, the start bits of the commands are used.

After the receiver has established the data connection to the controller it can decode and process the triggers, sent through channel A, and the broadcast and addressed commands on the B channel. The first level trigger signal goes right away to the VME P2 connector. The event header information, broadcast by the TCS controller, is reformatted into six bytes, which go via a FIFO buffer to the readout module (see table 4.2).

Byte No.	Bit assignment	
0	0 - 4	Trigger type [0..4]
	5 - 7	Spill number [0..2]
1	0 - 7	Spill number [3..10]
2	0 - 7	Event number [0..7]
3	0 - 7	Event number [8..15]
4	0 - 3	Event number [16..19]
	4 - 7	4 bits unused
5	0 - 7	Error byte [0..7]

Table 4.2: Format of the event header generated by the TCS receiver

To distinguish different receivers, they have an 8 bit TCS receiver ID. The ID is set by a DIL-switch on the card and is used by the addressed commands. Whenever an addressed command is received, the TCS receiver compares the receiver ID specified in the command with its own ID. The addressed command is only processed if both IDs match, otherwise it is ignored.

The receiver can also generate a pre-trigger pulse, that precedes the calibration and monitoring trigger of a particular type with configurable timing. This topic is covered in more detail in section 4.7.

Concerning the MultiDAQ mode the receivers count the spill and the event numbers of the non-MainDAQs. They also filter the triggers and broadcast commands according to the active DAQs and depending to which DAQ they are assigned to.

For monitoring purposes the TCS receiver provides status information like error registers, TCS receiver ID, configuration values etc., which can be accessed by the readout modules.

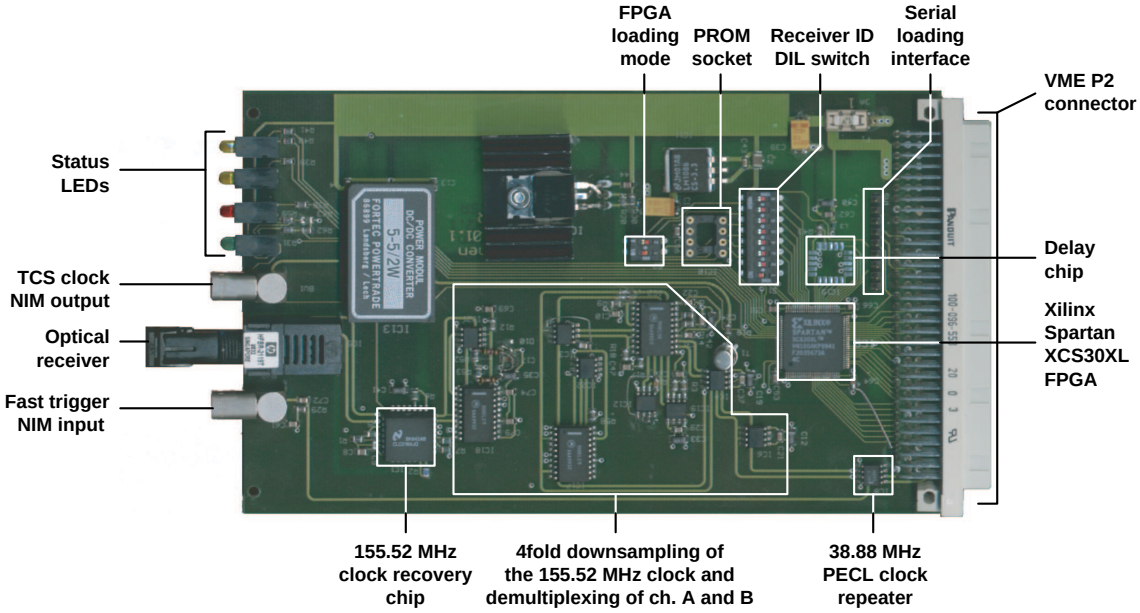


Figure 4.8: The TCS receiver board

The TCS receiver is a 3U VME card, which is plugged into the P2 backplane. Figure 4.8 shows a photograph of the board. Aside from the VME P2 connector and the optical receiver the card has two LEMO connectors. One LEMO provides the TCS clock in form of a NIM signal. The other one allows to sent so called 'fast triggers' to the readout module. If a detector requires a smaller trigger latency, this input can be used to bring the first level trigger on a faster path to the readout module. It can also be used for debugging purposes. Table 4.3 summarizes the in- and output signals of the TCS receivers.

4.5 The MultiDAQ mode

For debugging and testing purposes the TCS offers the so called MultiDAQ mode. In this mode the system supports up to eight logically independent DAQs. The DAQs are operated quasi-concurrently by using a time division scheme. Each DAQ gets its configurable time slice, which is the time interval, during which the TCS is exclusively allocated by the particular DAQ, so that this DAQ receives triggers. The controller cycles through all running DAQs in ascending order, granting them the

Signal	**Act. H/L**	**I/O**	**Src/ Dest**	**Type**	**Short description**
Ch. A, Ch. B	-	I	optical fiber	opt.	FLT and commands from the TCS controller
Receiver ID	H	I	switch	TTL[5]	ID number for the TCS receiver
PECL clock	-	O	A1-2,	PECL[6]	38.88 MHz high precision differential TCS clock
TTL clock	-	O	A11	TTL	TTL version of the PECL clock; has an uncertain phase shift compared to the PECL clock
NIM clock	-	O	LEMO	NIM	NIM version of the PECL clock; see above
Data [7:0]	H	O	A/C (4-7)	TTL	Data bus for the event header or status bytes
Address [3:0]	H	I/**O** **I**/O	A/C (8-9)	TTL	**Output:** byte number of the event header; **Input:** byte number of the status information
Header ready	L	O	A10	TTL	Goes LOW when event header data are waiting for readout; is HIGH if FIFO is empty
Header enable	L	I	C10	TTL	Read enable for the FIFO which buffers the event headers; if set to LOW and if 'Header ready' is also LOW the FIFO is read out at every rising edge of TTL clock
Status enable	L	I	C11	TTL	When LOW the byte of the status information specified by Address [3:0] appears on Data [7:0]
DataMode	H	O	A12	TTL	set together with FLT; switches the readout mode of the frontend electronics
Receiver ready	L	O	C12	TTL	is LOW when: - FPGA is successfully loaded - receiver gets 38.88 MHz clock - Ch. A and Ch. B are identified
BOS	H	O	A13	TTL	Begin of the spill signal
EOS	H	O	C13	TTL	End of the spill signal
FLT	H	O	A14	TTL	First level trigger
Pre-trigger	H	O	C14	TTL	Asynchronous to the TTL clock; used to trigger the generation of calibration pulses or test patterns
Reset	L	O	A15	TTL	Reset for the readout module
Error	L	O	C15	TTL	Set to LOW when error occurred
SkipData	L	O	A16	TTL	set together with FLT; when LOW the readout module sends out only headers
Fast FLT	- -	I O	LEMO C16	NIM TTL	input for fast trigger signal From NIM input converted to TTL

Table 4.3: The in- and output signals of the TCS receiver. For each signal the active level, the signal direction and the type are specified. Ax and Cx indicate the pins on the VME P2 connector.

configured time interval. This means, that only one DAQ can receive triggers (= can be active) at a time. The size of the time slices is in the order of milliseconds, so that

[5]**T**ransistor **T**ransistor **L**ogic (TTL) is a standard for digital signals

[6]The **P**ositive **E**mitter **C**oupled **L**ogic (PECL) is a standard for high speed differential digital signals that is used for the output of the TCS reference clock on the receiver.

the switching between the DAQs is quite fast, which allows a quasi-concurrent operation of the DAQs. Figure 4.9 illustrates the functional principle of the MultiDAQ mode.

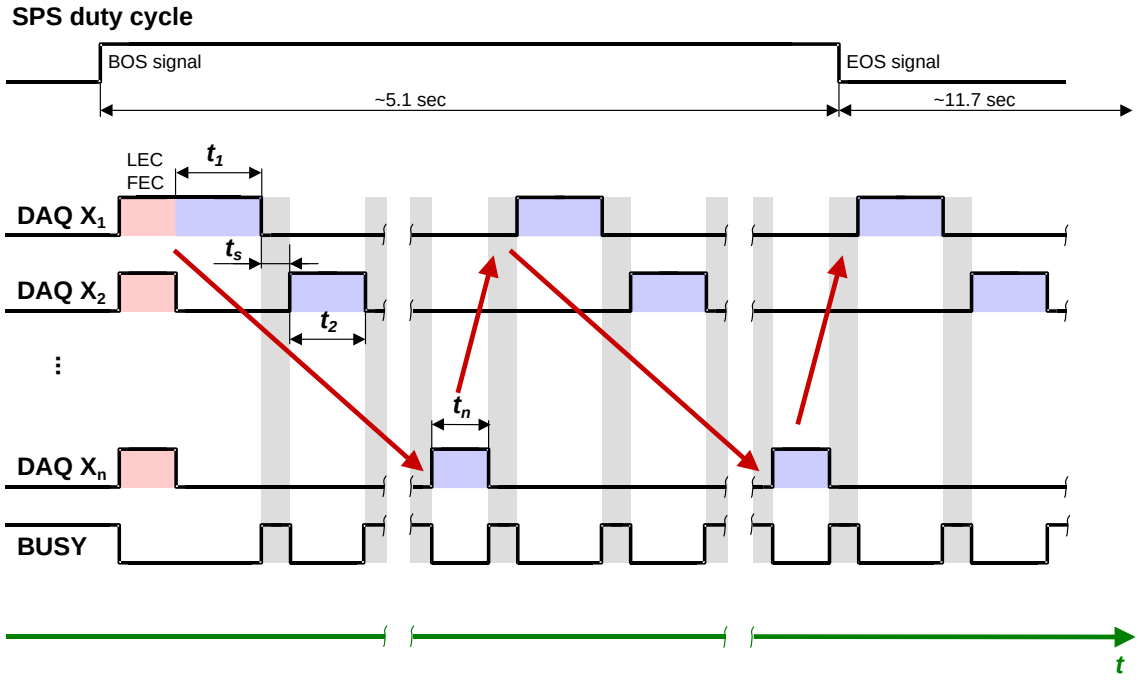


Figure 4.9: The functional principle of the MultiDAQ mode: The red arrow shows, how the controller cycles through the different running DAQs, granting them the configured time slices t_i. Only one DAQ is active at a time (blue areas). The only exception is at the beginning of the spill. At this time all running DAQs are active to receive the last and the first event in the cycle (red areas). The switching between the DAQs introduces additional dead time t_s (grey areas).

The DAQs share the TCS hardware resources, but are logically completely independent. Every single DAQ has its own spill and event number counters and receives only the triggers, which are dedicated for this DAQ. The only triggers, which are received by all DAQs, are the *first* and *last events in the run* and *in the cycle*. They are sent out during the begin of the spill (see section 4.8).

Concerning spill and event number counting there is one DAQ – the so called MainDAQ – that is distinct from the other seven DAQs. The MainDAQ has its spill and event number counters in the TCS controller, whereas for all other DAQs theses numbers are counted locally in the TCS receivers. So the MainDAQ is more fault tolerant, because there is only one data source for the event header information.

The MultiDAQ feature requires the receivers to be assigned to a certain DAQ and that they filter the triggers according to this assignment. To allow the filtering, the TCS receiver has four levels of activation, which are depicted in figure 4.10. After power-up and loading of the FPGA the receiver is in the 'off' state, which means that it ignores all commands from the TCS controller except the reset and the 'switch on' commands. The 'switch on' command puts the receiver into the activation level 1. Per default the receiver is attached to no DAQ. Activation level 1 now allows to

assign a DAQ to the receiver. The receiver waits for the next broadcast command from the controller, that announces which DAQs are running. If the DAQ to which the receiver was attached to is running, the receiver goes to activation level 2 and decodes now all commands. The receiver waits, until a broadcast announces the begin of the active time slice for the assigned DAQ and goes then to the highest activation level, which means, that it removes the blocking of the triggers and is fully functional.

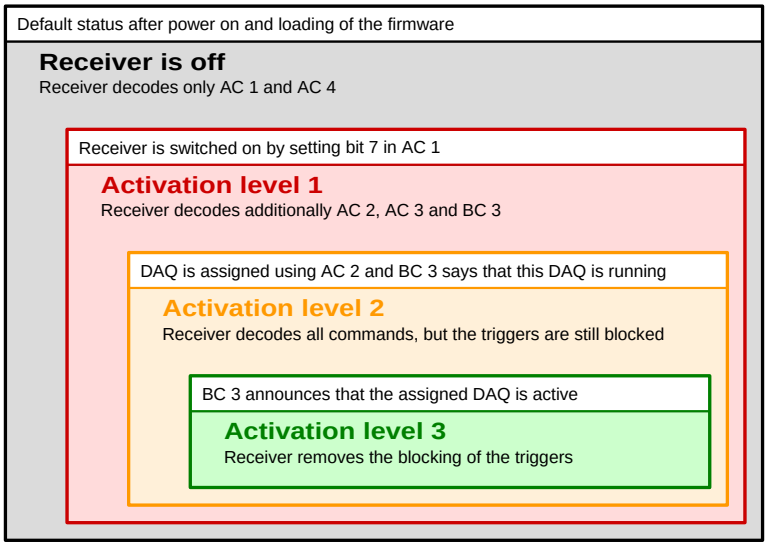


Figure 4.10: The four activation levels of the TCS receiver (see section 4.9 for reference on the broadcast and addressed commands BC x and AC x)

To allow a safe switching between the DAQs without mixing up triggers or data, extra dead time is introduced by the TCS controller. The switching procedure is shown in figure 4.11.

In the MultiDAQ mode the particular DAQs can be started and stopped without any constraints. It is possible to include/exclude a DAQ into/from data taking, while other DAQs (and thus the TCS) keep on running. This means, that it is also possible to go from the SingleDAQ mode (where only the MainDAQ is running) to the MultiDAQ mode and back without stopping the TCS, by starting and stopping the additional DAQs respectively. The starting and stopping of the DAQs is always performed at the beginning of the spill (see section 4.8).

4.6 Trigger logic, dead time and TCS

An important task of the TCS is the generation of the dead time for the experiment. This is done by the TCS controller. It generates the BUSY signal, that goes to the trigger logic and vetoes there the first level trigger. The dead time cannot be configured individually for each DAQ, but is the same for all DAQs. It consists of two components: First there is a so called 'fixed dead time' after every first level

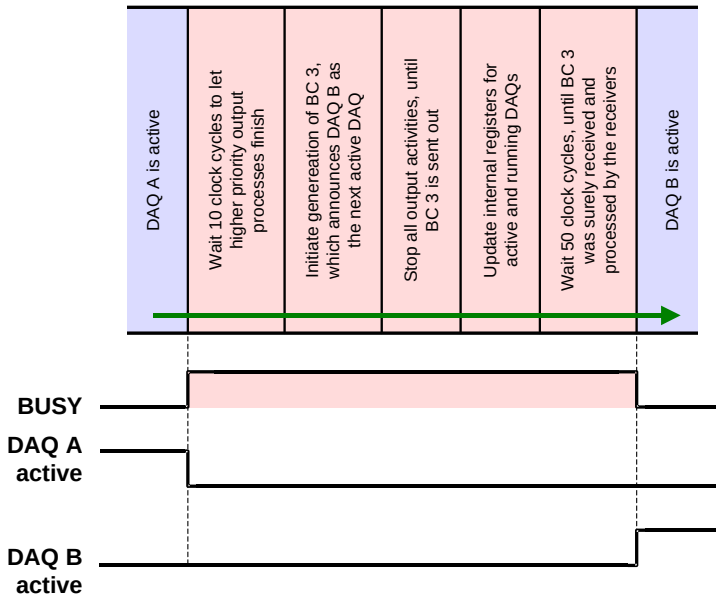


Figure 4.11: The switching procedure from one DAQ to another in the MultiDAQ mode

trigger. The fixed dead time can range from 175 ns to 13.61 μs in steps of 50 ns. The value can be set using a VME register in the controller.

In addition to the fixed dead time there is the so called 'variable dead time'. The variable dead time can be used to keep the average trigger rate below a certain value. The trigger rate is defined by the number of first level triggers, that are allowed during a particular time window. The time interval can be configured between 0.026 μs and 1685.6 μs in steps of 25 ns. The number of triggers can range between 1 and 15. If the number of allowed triggers is bigger than 1, the minimum time between two triggers is defined by the fixed dead time. Only when to many triggers are coming to close to each other, the variable dead time mechanism asserts the BUSY signal. One can adjust the tolerance of the variable dead time concerning short term transgressions of the average trigger rate by changing the number of allowed triggers. The higher the number of triggers is set, the higher the short-term overshoot in the trigger rate can be and the less the variable dead time blocks the incoming triggers. The variable dead time can be configured through VME registers in the TCS controller.

4.7 Trigger types, pre-trigger, calibration and monitoring trigger

The 5 bit trigger type is a part of the event header information, that the TCS generates for every first level trigger. This number allows to distinguish up to 32 different types of triggers. The most important trigger type is of course the

physics trigger. All events which are triggered by the external trigger logic are of this type. But there are also artificial triggers, generated by the TCS itself. With the *first* and *last event in the cycle* and *in the run* the SPS spill structure is marked in the data flow. There are also trigger types, that can be used for calibration and monitoring purposes. Table 4.4 shows the assignment of the different trigger types.

Trigger type		**Description**
dec	hex	
0	0	Physics trigger
1	1	Calibration/monitoring trigger type 1
2	2	Calibration/monitoring trigger type 2
⋮	⋮	⋮
27	1B	Calibration/monitoring trigger type 27
28	1C	**F**irst **e**vent in the **r**un (FER)
29	1D	**L**ast **e**vent in the **r**un (LER)
30	1E	**F**irst **e**vent in the **c**ycle (FEC)
31	1F	**L**ast **e**vent in the **c**ycle (LEC)

Table 4.4: The different trigger types of the TCS

The calibration and monitoring triggers are requested by the TCS server. The software writes the requests into the slow command FIFO of the TCS controller. The controller transmits this command to the TCS receivers and generates the artificial trigger of the requested type after a fixed time interval of 6.58 μs. The TCS receiver offers quite a lot of flexibility in handling the calibration and monitoring triggers. Every receiver filters the incoming events according to their trigger type. Physics and special events (last/first event in the cycle/run) are always accepted, but for the calibration and monitoring triggers the receiver accepts only those trigger types, that were activated during the configuration of the TCS. If a trigger is not accepted by the receiver, the SkipData signal is activated. This forces the readout module, connected to the receiver, to discard the data and to generate only headers for that particular event. Every receiver can be configured to accept an arbitrary subset of the calibration and monitoring trigger types.

The TCS receiver can generate a pre-trigger pulse, preceding the calibration and monitoring trigger with a configurable timing. The time interval between the pre-trigger and the following artificial trigger can be programmed in steps of 100.5 ps in a range from 100.5 ps to 6.58 μs (see subsection 4.9.2). The pre-trigger pulse can be used to trigger equipment that generates calibration pulses, like pulse generators, flash lamps and so on.

For calibration purposes some detectors have to use a different readout mode. For example the GEM and the silicon detectors have to switch off the zero suppression to measure pedestals. This is done by the DataMode signal, which can switch the readout module between two readout modes. The receiver sets the DataMode signal

together with the calibration and monitoring trigger. The behavior of the DataMode signal can be configured independently for each receiver ID and each trigger type.

In case of the MultiDAQ mode the things get a bit more complicated. Because the switching between the different DAQs happens on the millisecond scale, it would be difficult to synchronize the TCS server software with the time slices of the DAQs. This is why the TCS controller takes care, that the requests for the calibration and monitoring triggers are executed at the right time. Therefore the request command contains the number of the DAQ, for which the calibration and monitoring trigger should be generated. In the MultiDAQ mode the maximum rate of calibration and monitoring triggers, that can be generated by the TCS controller is lower compared to the SingleDAQ operation, because there is extra time the controller waits for the right DAQ to get active.

In practice this scheme of handling the calibration and monitoring triggers allows to assign trigger types to particular detectors. In this case only one detector is read out, if an artificial trigger is initiated, all others just deliver headers. In combination with the pre-trigger pulse and the DataMode signal the TCS supports a broad variety of calibration procedures.

4.8 The BOR, BOS and EOR procedures

The begin of the spill signal is quite important for the operation of the TCS. It initiates a sequence of actions, that synchronize the DAQ(s) with the SPS spill structure. The BOS signal triggers the generation of the *last* and *first event in the cycle*. The readout modules are reset at this time and the DAQ(s) (as well as the TCS) are started and stopped always at the beginning of the spill.

A 'start of the run' sequence for the SingleDAQ mode is depicted in figure 4.12. First the run control requests the TCS server to configure all receivers. The server generates a sequence of addressed commands and writes them via the VME bus into the slow command FIFO of the TCS controller, from where they are transmitted to the TCS receivers. After that the TCS server starts the TCS by writing a non-zero value for the duration of the time slice of the MainDAQ into the appropriate VME register of the TCS controller. This orders the TCS controller to start. The controller waits for the next begin of the spill signal, before it empties the fast command FIFO and waits for 1 ms. Thereafter all readout modules are reset with a broadcast command and the controller waits 200 ms, before it sends out the begin of the spill broadcast with the new spill number. It follows a second break of 200 ms. At the last step the artificial *first event in the run* is generated, the controller releases the BUSY signal and waits for FLTs from the trigger logic.

To stop a running TCS, the run control orders the TCS server to set the register values of the time slices of all running DAQs to zero. The TCS controller waits for the next begin of the spill signal and generates the artificial *last event in the run*.

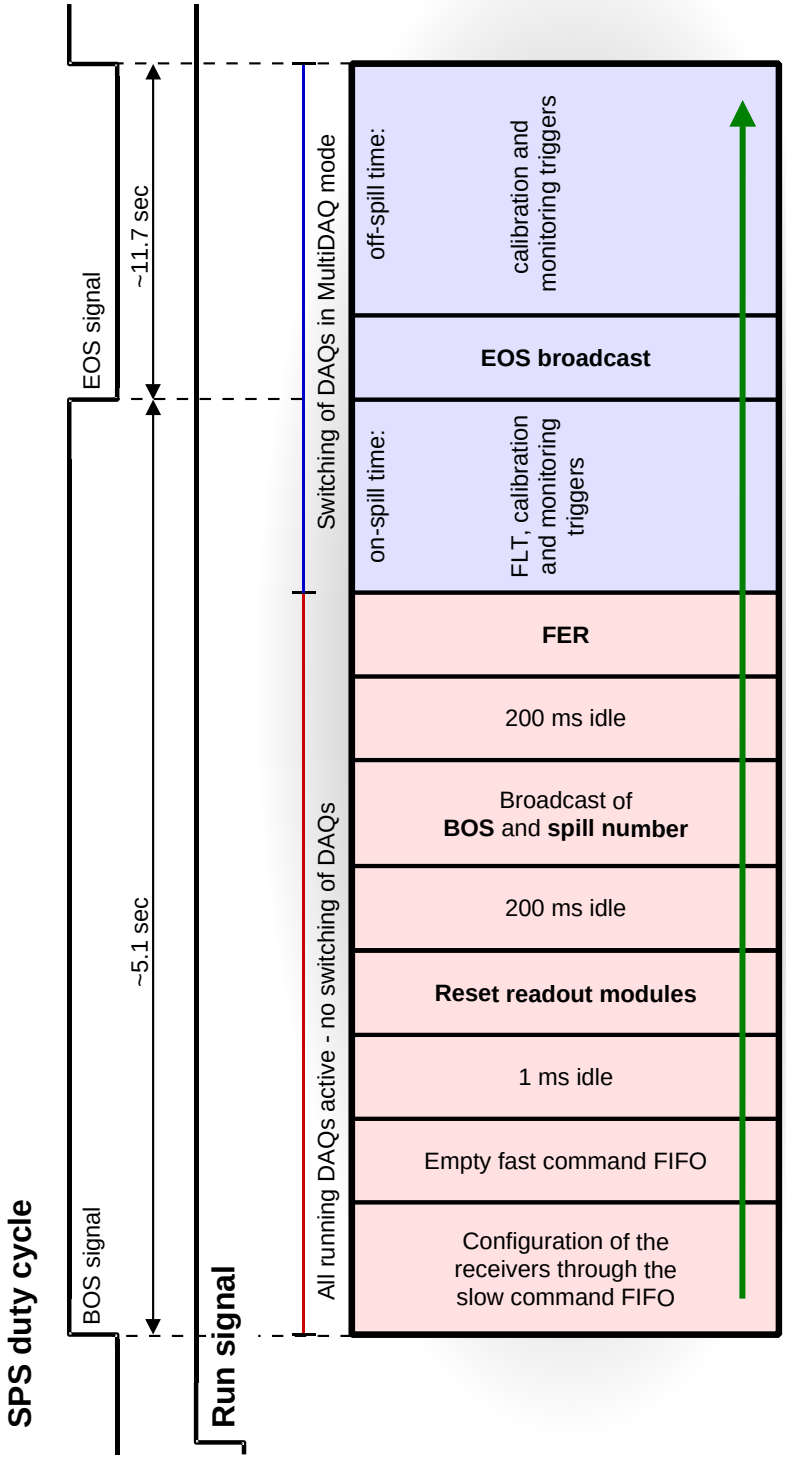


Figure 4.12: The begin of the run procedure for the SingleDAQ mode

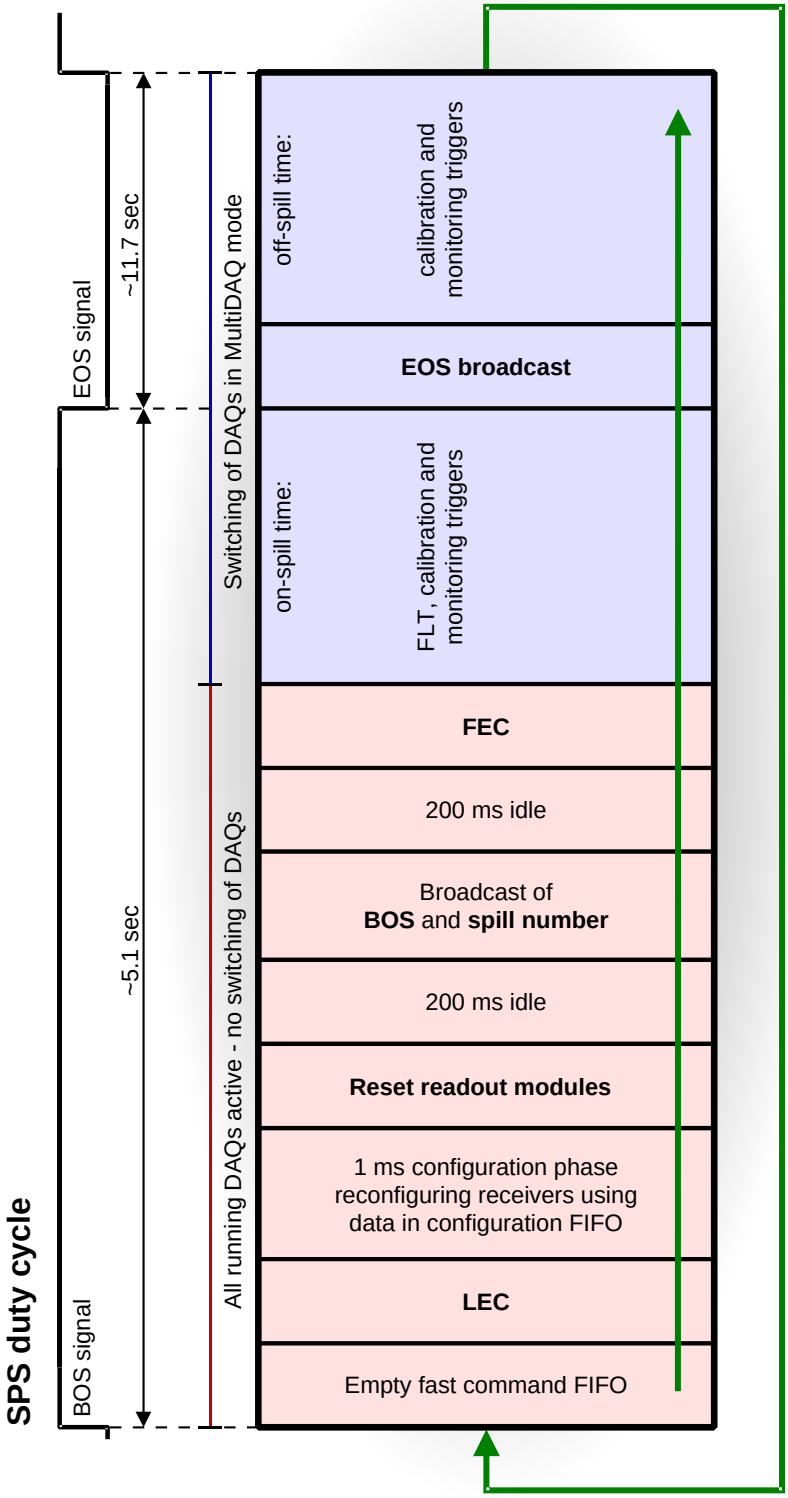


Figure 4.13: The begin of the spill procedure

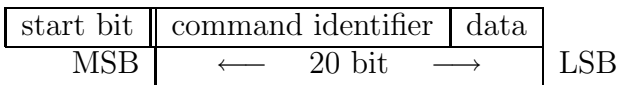
Then it blocks all FLTs by setting the BUSY signal, waits for 200 ms, sends a reset broadcast to all readout modules and stops the TCS.

If the TCS is already running, the sequence at the beginning of the spill (see figure 4.13) is quite similar to the one at the beginning of the run. The FLTs are already blocked by the BUSY signal since the end of the spill signal. When the TCS controller receives the BOS signal, it processes all the commands that are waiting in the fast command FIFO and generates the artificial *last event in the cycle*. Thereafter begins the so called 'configuration phase', which is 1 ms long. The configuration phase is needed for the MultiDAQ mode. The TCS allows to start a DAQ, even if other DAQs (and thus the TCS) are already running. This startup 'on the fly' requires, that the TCS receivers of the added DAQ are activated and configured at the right time, so that the broadcasts, which they receive, are similar to the ones, sent in the begin of the run sequence. This time is the configuration phase, because after this the receivers get a reset broadcast and 200 ms later a BOS broadcast with the new spill number. The *first event in the cycle* is broadcast after another break of 200 ms. So the only difference between starting in the SingleDAQ mode and hooking up an additional DAQ to an already running TCS is, that in the latter case the added DAQ(s) will get a *first event in the cycle* instead of a *first event in the run*. The same holds true, if one stops a DAQ, while other DAQs (and thus the TCS) keep on running. In this case the stopped DAQ will receive a last event in the cycle instead of a last event in the run.

Another issue is, that the configuration phase is only 1 ms long. To configure all the necessary TCS receivers by the TCS server software during this time, would be quite difficult. To overcome this problem, the 'configuration FIFO' was added. Due to limited space in the controller FPGA this FIFO sits in the FPGA of the GenericVME board. It is accessible via a VME address. The configuration FIFO is read by the TCS controller only during the configuration phase. So the TCS server can prepare the sequence of addressed commands, needed to start up the DAQ, during the off-spill time.

4.9 The TCS commands

All data transfers from the TCS controller to the TCS receivers are performed in the form of so called 'commands'. The general structure of a TCS command is the following:



There are two types of commands: broadcast and addressed commands. The broadcast commands carry data, that have to go to all receivers in the system, like for example event header information, announcements of the next active DAQ, begin

and end of the spill signals (BOS, EOS) and so on. All active receivers will process this information. In contrast to this the addressed commands are only accepted by the receivers with the matching receiver ID. These commands are used to configure the different receivers in the system.

There is an additional classification of the command set in terms of priority. The TCS controller has two FIFOs, which act as output buffers: one FIFO for the so called 'fast commands' and one for the 'slow commands'. The fast command FIFO has a higher priority, which means that the commands, stored in the slow command FIFO, are only executed, when the fast command FIFO is empty.

The slow command FIFO is accessible from outside via the VME bus. Usually it buffers the addressed commands, which are issued by the TCS server for configuration purposes. The fast command FIFO is only used internally by the controller to buffer the generated broadcast commands.

The start bit, which is needed for synchronization, is automatically generated by the controller and will be omitted further on. Unused bits will be marked with a '*'.

4.9.1 The broadcast commands (BC)

Broadcast commands are decoded and executed by all active receivers in the system. They are generated by the TCS controller and sent out through the fast command FIFO. There are three broadcasts BC 1, BC 2 and BC 3.

BC 1 is mainly used to distribute the begin of the spill and end of the spill signals together with the belonging spill number. In addition global resets for all TCS receivers and all readout modules can be sent.

BC 1:

0001		Rec. reset	EOS	BOS	*	Readout mod. reset	Spill No. [10:0]	
19	16	15	14	13	12	11	10	0

BC 2 carries the event number and the trigger type. Together with the spill number, sent by BC 1, the receivers can generate the complete event header information for the readout modules. Since the data payload of a broadcast command is maximum 16 bit, but event number and trigger type together give 25 bits, the BC 2 is split into two parts. Part 1 transmits the 5 bit trigger type together with the lower byte of the event number. Part 2 carries the remaining 12 higher bits of the event number.

In a stable system environment only part 1 will be sent to reduce the load and the latency on the data link. The TCS receivers will automatically add the missing 12 bit high part of the event number. Only when an overflow in the low byte of the event number occurs, which means every 256 events, the second part will be generated additionally to refresh the register for the high part in the receivers.

BC 2 Part 1:

0110		Event No. low [7:0]		***		Trigger type [4:0]	
19	16	15	8	7	5	4	0

BC 2 Part 2:

0111		****		Event No. high [11:0]	
19	16	15	12	11	0

The last broadcast BC 3 is needed for the MultiDAQ mode. It is part of the time division control for the different DAQs. Whenever the time slice of a DAQ is over, this command will be sent. It announces to all receivers, which DAQs are running and which DAQ will be active during the upcoming time slice. This is done using two bit masks each 8 bit wide. The mask of the running DAQs informs the receivers, whether DAQs were started or stopped (see section 4.5).

BC 3:

0010		Next active DAQ mask [7:0]		Running DAQs mask [7:0]	
19	16	15	8	7	0

There is a fourth type of broadcast command - the so called 'dummy command'. This command is only used to create some fake traffic on the B channel, so that the TCS receivers are able to identify channel A and B, in case the TCS is idle. The dummy command itself carries no data and is ignored by the receivers (see section 4.4).

Dummy command:

0101		0101010101010101	
19	16	15	0

4.9.2 The addressed commands (AC)

In addition to the data payload the addressed commands carry the 8 bit receiver ID. If the specified receiver ID in the addressed command matches the ID of a receiver, the command is executed by this receiver, otherwise it is ignored. The 256 different receiver IDs can be used to partition and group the receivers, for instance according to their affiliation to the different detectors. Addressed commands are usually generated by the TCS server, which writes them via the VME interface into the slow command FIFO of the controller.

The first addressed command AC 1 is used for basic configuration of the TCS receiver and is quite loaded with features. The most important function of the AC 1 is

switching on and off the receiver (see the activation levels described in section 4.5). AC 1 also allows to configure the behavior of the receiver concerning calibration and monitoring triggers. Each trigger type can be activated separately. In addition the DataMode signal and the generation of the pre-trigger pulse can be configured individually for the particular trigger type.

The receiver uses bit masks to manage this information. One bit mask stores the information, which trigger types the receiver accepts and which not. If the receiver gets an event header broadcast with a trigger type, that it does not accept, it will set the SkipData signal, which forces the readout module to send only headers for this particular event. A similar bit mask controls how the DataMode signal has to be set for the different trigger types. The DataMode signal can be used by the readout modules and the frontend electronics to switch between the standard readout mode and a special readout mode for calibration. The generation of a pre-trigger pulse is determined by the third bit mask. For each single event type the pre-trigger pulse can be activated. The timing of the pre-trigger pulse with respect to the following calibration and monitoring trigger is defined by two other commands (see AC 3 and calibration/monitoring trigger request command below).

Per default none of the 27 calibration trigger types is accepted, DataMode is set to '0' for all trigger types and no pre-trigger pulses are generated. AC 1 activates the features trigger-type-wise. For the specified trigger type there are three flags, that control the activation of the trigger type, the generation of the pre-trigger pulse and the setting of the DataMode signal. This means, that for every trigger type which one wants to change with respect to the defaults settings, one has to issue an AC 1, with the appropriate flags set.

AC 1:

10	Receiver ID [7:0]	*	Pre-trigger flag	Enable receiver	...
19 18	17 10	9	8	7	

...	Trigger type [4:0]	Accept trigger type flag	DataMode flag
	6 2	1	0

AC 2 is needed for the MultiDAQ mode. It sets the DAQ(s), to which the TCS receiver is attached. This is done using an 8 bit mask, so that in principle the receiver can be assigned to any subset of the maximum eight DAQs. In the MultiDAQ mode a receiver is attached to only one DAQ.

AC 2:

1101	Receiver ID [7:0]	DAQ mask [7:0]
19 16	15 8	7 0

Because AC 3 is concerned with the setting of the timing of the pre-trigger pulse, it will be discussed together with the pre-trigger request command below. The AC 4

allows an addressed reset of the TCS receivers and the readout modules. The global reset of all receivers and all readout modules in the system is already provided by the broadcast command BC 1 (see subsection 4.9.1).

AC 4:

1111		Receiver ID [7:0]		******		Rec. reset	Readout mod. reset
19	16	15	8	7	2	1	0

In the narrower sense the calibration and monitoring trigger request command is not an addressed command, because it does not contain a receiver ID. Instead of this the receivers are addressed by the trigger types they accept. The request command is generated by the TCS server and written into the slow command FIFO of the TCS controller. The command orders all TCS receivers, which belong to the specified DAQ, to generate the pre-trigger pulse (if this feature is enabled for the specified trigger type) and orders the TCS controller, to generate the belonging calibration and monitoring trigger for this DAQ. If the system is running in the MultiDAQ mode, the controller takes care, that the command is processed in the right time slice. During the BOR and BOS procedures all requests for calibration and monitoring triggers are ignored. This is also the case, if the DAQ, specified in the request, is not running.

The timing between the pre-trigger pulse and the following calibration and monitoring trigger is configurable in a range from 100.5 ps to 6.58 μs in steps of 100.5 ps. The time interval is defined by setting two values: a 'coarse delay' and a 'fine delay'. The coarse delay is set by the addressed command AC 3. It can be programmed individually for every TCS receiver ID. The TCS controller generates the calibration and monitoring trigger with a fixed timing, 6.58 μs after the request was sent to the receiver. The coarse delay value gives the time interval in units of TCS clock cycles, the TCS receiver waits, until it generates the pre-trigger pulse. This means the bigger the coarse delay value, the smaller the time interval between the pre-trigger and the calibration/monitoring trigger. The coarse delay shifts the pulse edge of the pre-trigger in steps of 25 ns. To allow a finer timing, a delay generator is mounted on the receiver card. It can shift the pre-trigger in steps of 1/256th of the TCS clock cycle or 100.5 ps. The fine delay value is used to program the delay generator. Like the coarse delay the fine delay shifts the pre-trigger pulse closer to the calibration/monitoring trigger.

AC 3:

1110		Receiver ID [7:0]		Coarse delay [7:0]	
19	16	15	8	7	0

Calibration and monitoring trigger request:

1100		Trigger type [4:0]		DAQ No. [2:0]		Fine delay [7:0]	
19	16	15	11	10	8	7	0

4.10 Summary and outlook

The Trigger Control System was used in the year 2000 run of COMPASS and in an enhanced version in the year 2001 run. In 2001 82 receivers were connected to the controller. The system behaved well and fulfilled all design requirements. In lab tests it was proven, that the system is able to distribute the event header information for the first level trigger at rates well above 100 kHz.

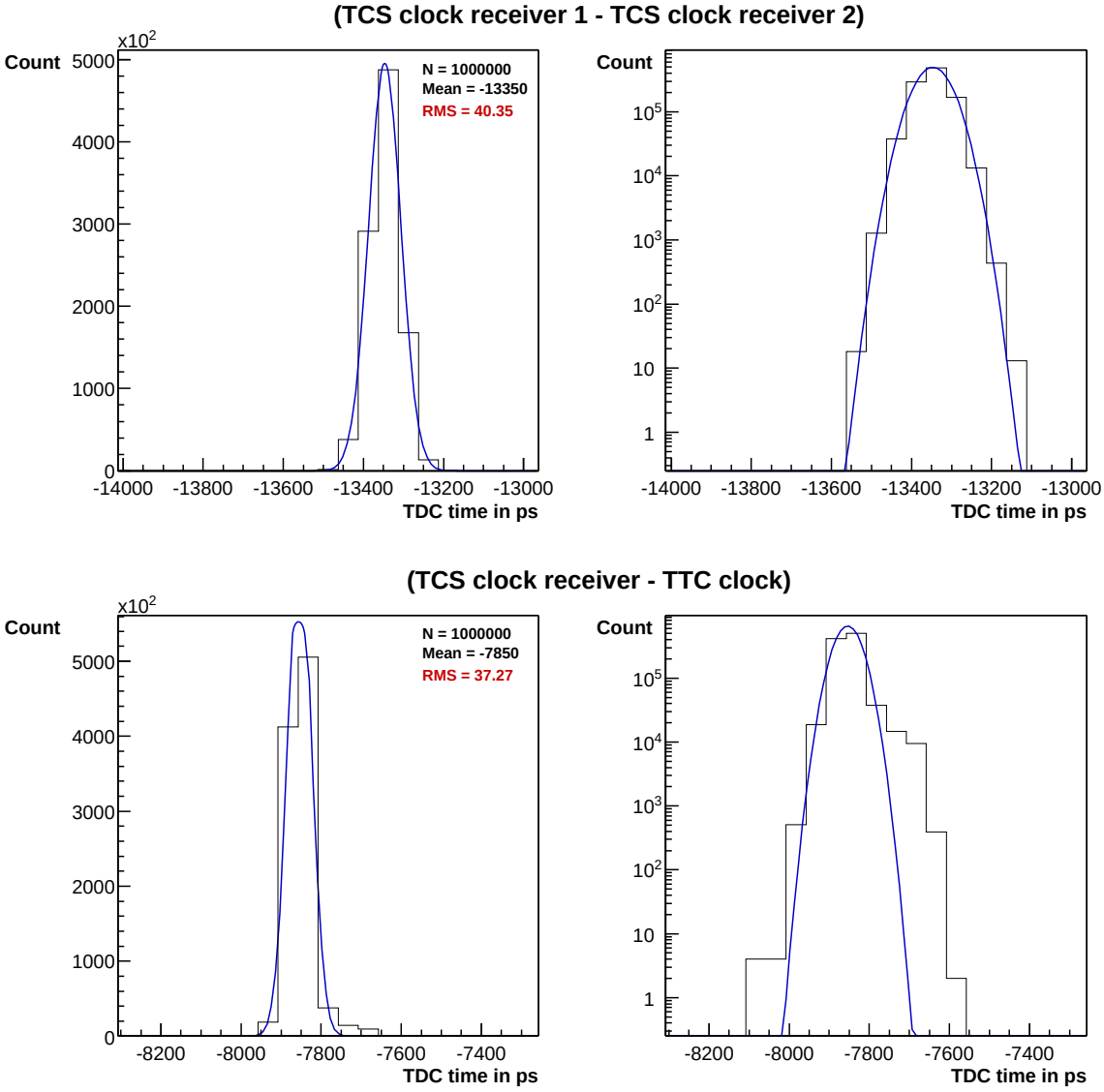


Figure 4.14: The upper two graphs show the distribution of the time difference, measured between the rising edges of the TCS clocks of two different receivers, on the left side in linear on the right one in logarithmic scale. A Gaussian is fitted to the distribution. The RMS value of the distribution is well below 50 ps. The lower diagram shows the distribution of the time difference between the rising edges of a receiver TCS clock and that of the high precision clock of the TTC crate.

The jitter of the TCS reference clock was measured using a gateable low jitter PECL to NIM converter and a CAMAC[7] LeCroy TDC model 2228A, which provides a time resolution of 50 ps. Figure 4.14 shows the resulting distributions. The jitter between the TCS clocks of two receivers is 41 ps RMS and thus well below the required 50 ps RMS.

But of course there is still enough room for improvements. Some of the features of the system were not planned from the beginning. This is why the design of the hardware and the FPGA programs is only suboptimal, especially concerning the TCS controller. It misses for example the necessary in- and output connectors to run eight DAQs in the MultiDAQ mode. In addition the capacity of the FPGA chip of the controller is already exhausted with the present program, so there is no space for additional features.

A new version of the controller hardware, consisting only out of one big FPGA, is planned for the year 2002 run. This controller version will have sufficient in- and output connectors and it will allow to include the non MainDAQ counters into the controller, making the design cleaner and freeing resources in the TCS receivers. On this occasion also the command set and the VME interface to the TCS server software will be revised and made more convenient. It turned out, that the presently used NIM busy logic for the dead time is quite unconvenient and delicate to set up. The new controller will incorporate the busy logic and will allow individual dead time settings for all DAQs. In addition it will provide a pre-scaler for the first level triggers to allow basic adjustment of the incoming trigger rate.

[7]The **C**omputer **A**ided **M**easurement **A**nd **C**ontrol (CAMAC) is a standard bus system for the transmission of digital data

Chapter 5

The GEM and silicon readout system

The central part of the GEM and silicon readout chain is the APV 25 frontend chip. The features and requirements of this chip constrain the design of the readout electronics. Therefore this chapter will start with a description of the APV 25.

The particular design of the frontend electronics is governed by the properties of the belonging detector. Because the APV was originally designed for silicon detectors, some adaptations are needed in the GEM frontend electronics. The basic functional principle of the both detectors and the design of their frontend electronics are described.

The data of the APVs are digitized by ADC cards, which also perform the zero suppression. The digital data of four ADC cards go via optical fibers to the GeSiCA readout module, which sends them through optical S-Link to the Readout buffer. After an overview section, that introduces the architecture of the system, the proximate sections will explain the particular components of the readout chain in more detail by following the data flow from the APV frontend chips to the Readout Buffers. Special focus will be given to the zero suppression in the ADC card.

5.1 The frontend chip APV 25 S0

The APV 25 (see figure 5.1) is a development of the CLRC[1] for the silicon tracker of the CMS experiment, that will be build at the Large Hadron Collider at CERN. The APV 25 is an ASIC[2], that combines analog signals with digital control. The chip is manufactured in a radiation hard deep submicron (0.25 μm) CMOS process[3].

[1] **C**entral **L**aboratory of the **R**esearch **C**ouncils

[2] **A**pplication **S**pecific **I**ntegrated **C**ircuit

[3] Manufacturing process for semiconductor components and integrated circuits that uses **C**omplementary **M**etal-**O**xide-**S**emiconductor (CMOS) field-effect transistors

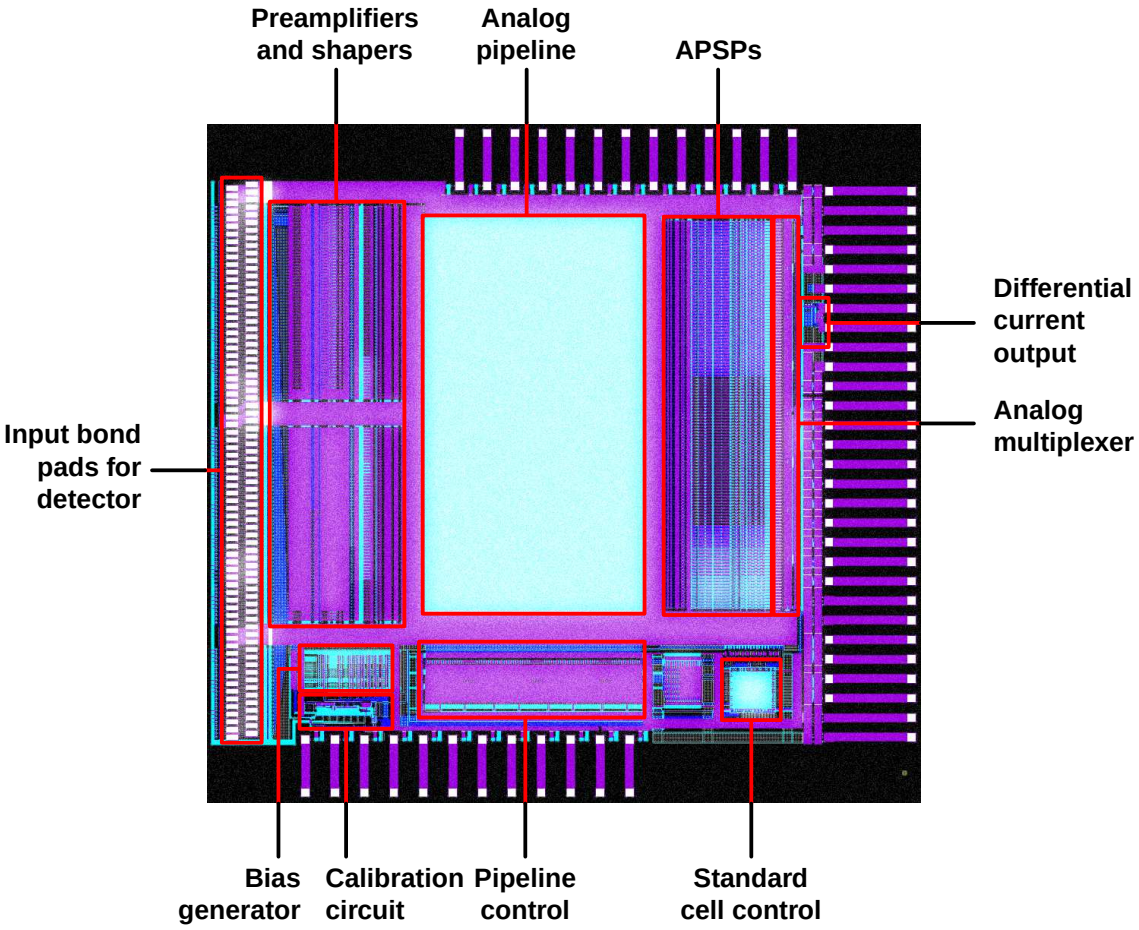


Figure 5.1: The APV 25 S0 frontend chip [Tur00]

The chip features 128 channels, each having a low noise 50 ns CR-RC shaping amplifier and a 192 samples deep analog pipeline. The amplifier integrates the current pulse from the input and transforms it into a well defined voltage pulse. A unity gain inverter at the shaper output allows to change the signal polarity, so that the signals coming from the p- and the n-side of the silicon detector can be switched to equal polarity. The sampling frequency of the APV is 40 MHz. An analog 128 to 1 multiplexer puts the data to the analog differential output line. The multiplexing leads to a nonconsecutive channel order of the analog samples. The readout frequency used in the present design is 20 MHz, but the APV itself supports up to 40 MHz.

The analog memory pipeline of the APV is needed to compensate the latency of the first level trigger. The pipeline is made out of switched capacitor elements. 160 of the 192 memory cells are organized like a ring buffer. They cache the incoming data. With the 40 MHz sampling rate the data can be delayed by maximum 4 μs with respect to the time, the particle crosses the detector. The ring buffer has a write and a trigger pointer. The distance between the two pointers defines the trigger latency,

which is a configurable parameter. In case of an incoming trigger the trigger pointer flags the pipeline column to read out. If a pipeline column was flagged for readout, it is copied into one of the remaining 32 memory elements, which are organized like a FIFO. There the data are stored, until they are sent out by the multiplexer.

The APV 25 offers three different modes of operation: 'Peak', 'Deconvolution' and 'Multi' mode. In the Peak mode one pipeline column (which means one sample of the signal pulse) is flagged by the trigger for readout. Usually the timing is set in a way, that the sample which is read out, sits at the peak of the signal pulse shape.

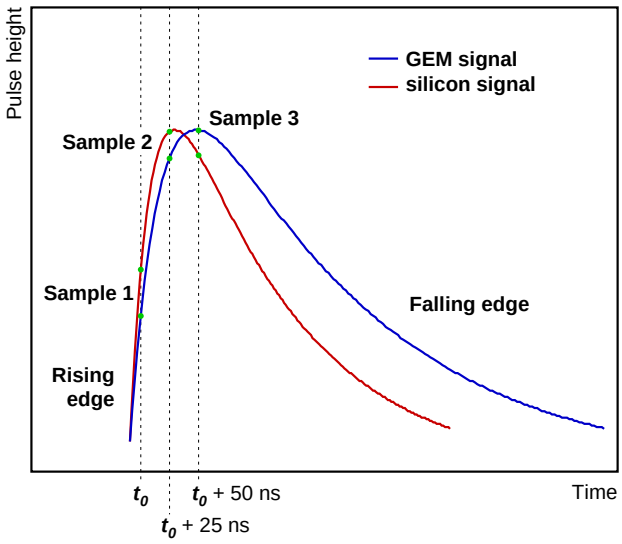


Figure 5.2: This graph shows the optimal timing for the three samples of the Multi mode. The GEM signal is quite slow, so the best time resolution can be achieved, if all samples sit on the rising edge of the signal, so that the last sample is at the signal peak. Because the silicon signal is faster, the second sample should be at the peak, whereas the other two samples sit at the rising and the falling edge of the signal respectively.

Because the COMPASS beam has no special time structure, the particles may cross the detectors at any time. If one would read out the detector in the Peak mode, the time resolution could not be better than the period of the sample clock which is 25 ns. That's why the GEM and the silicon readout uses the Multi mode of the APV. In the Multi mode the trigger flags three consecutive pipeline columns for readout. This means, that with every event the APV writes out three samples of the signal shape, which are spaced by 25 ns in time. This allows a partial reconstruction of the pulse shape and strongly enhances the achievable time resolution (see figure 5.2). It also helps to compensate to some extend the effect of pile-up at high beam intensities. Pile-up means, that the signals of two or more different events are so close to each other, that they partly overlap. By looking at the time evolution of the combined signal, it is possible to extract the signal with the right timing.

In the Deconvolution mode every trigger also flags three consecutive samples. The three samples are passed through a three weight FIR filter[4]. The filter performs

[4]A **F**inite **I**mpulse **R**esponse (FIR) filter is a linear digital filter, that operates in the time do-

a deconvolution of the three samples, so that the signals which are superimposed due to the pile-up are disentangled. The output is just one sample, like in the Peak mode. The Deconvolution mode cannot be used with the GEM detectors, because the filters are optimized for the signal properties of a silicon detector. The silicon detectors also use the Multi mode instead of the Deconvolution mode. This has two reasons: In the first place the Deconvolution mode was designed to be used in collider experiment, where the bunch structure of the beam defines the event timing. Because the COMPASS beam has no structure and the silicon signal is relatively fast compared to the period of the sample clock, the Deconvolution mode does not work efficiently. In addition one can get a much better time resolution using the Multi mode and the readout electronics for the silicon and the GEM detectors can be identical.

During data taking the APV is controlled by only one LVDS[5] line called TRIG. A double pulse spaced by one clock cycle ("101") on this line acts as a reset. A single pulse of one clock cycle duration ("100") is a trigger. Because of the Multi mode this pulse will select three consecutive pipeline columns for readout. The trigger pulses have to be separated by at least two clock cycles. The third sequence "110" is a calibrate request. It triggers the on-chip pulse generator, that injects a predefined charge into the input of the pre-amplifiers of 16 channel groups. The resulting pipeline data may then be triggered for readout a latency period later.

The APV puts out the data using differential currents in the range of ± 4 mA with an analog gain of about ± 1 mA per MIP[6] for a silicon detector. The signals of the GEM detector have larger amplitude. If there are no data to write out, the chip sends every 70 clock cycles (of the 20 MHz readout clock) synchronization pulses called 'ticks'. When an event is triggered, the chip waits until the next tick, before it starts to send out data. The data set of one pipeline column is called 'frame' and consists of four parts: the 3 bit digital header, the 8 bit digital column address, the error bit and the 128 analog samples (see figure 5.3).

The first three parts of the frame contain only digital information. A logical '0' is represented by a current of $+4$ mA a logical '1' by a -4 mA current. The three consecutive '1's of the header tag the begin of the data. The next 8 bit represent the column address, that stored the data at the trigger time. Because the trigger latency is constant, the column address should also be constant. By monitoring the column addresses, one can check the synchronicity of many APVs. This allows to identify and remove data from bad memory locations. The error bit is active low, a '1' means, that there is no error. If the error bit is set to '0' the error register (see subsection 5.8.1) should be checked for more detailed information about the error.

---

main. The impulse response is the output of the filter, arising from a short impulse as input. [Leo87] Every channel of the APV has an **A**nalog **P**ulse **S**hape **P**rocessor (APSP), that performs the filtering, which is a weighted sum of the three samples. [Jon99]

[5] **L**ow **V**oltage **D**ifferential **S**ignal

[6] **M**inimum **i**onizing **p**article

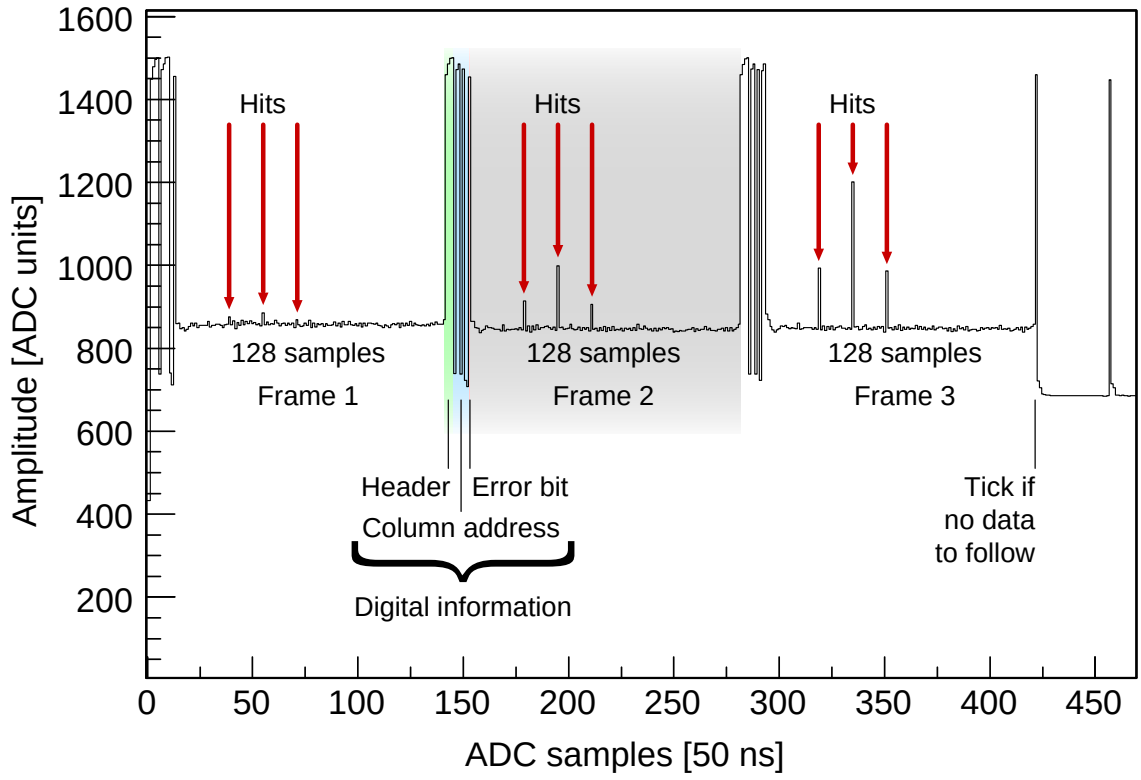


Figure 5.3: This is an example for raw data, taken in the Multi mode from an APV 25, that was connected to a GEM detector. One can see the three frames containing the data from the three consecutive samples. In the second frame the different parts are labeled. The data also contain a cluster of three hits, marked by the red arrows. The rise of the signal from the first frame to the third frame is nicely visible. The strips, which are hit, have the numbers 49, 50 and 51. That the signal pulses do not lie adjacent, is the effect of the output multiplexing.

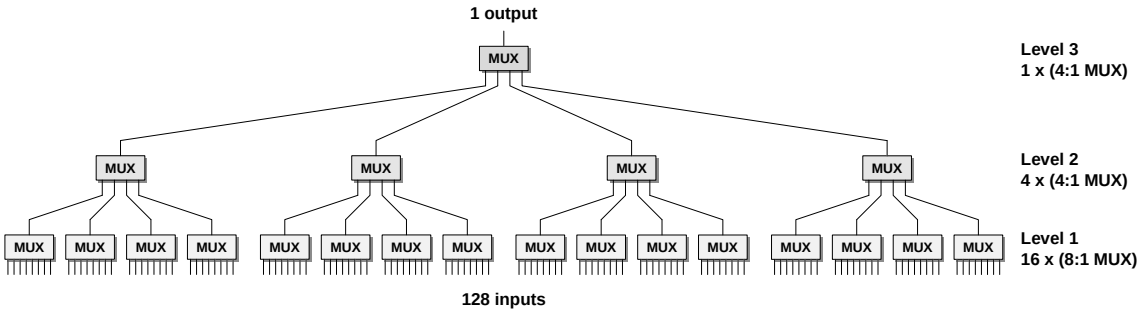


Figure 5.4: The three level tree structure of the analog output multiplexer of the APV. The multiplexer on the highest level has the highest priority.

The 12 bit of digital information are followed by the 128 analog samples. The samples of the 128 channels are multiplexed into a single line. The multiplexer has a three level tree structure (see figure 5.4) and due to this the data are shuffled and have to be reordered. In software this can be done using the following formula:

$$\text{Channel } \# = 32 \cdot (n \text{ MOD } 4) + 8 \cdot \text{INT}(n/4) - 31 \cdot \text{INT}(n/16)$$

In hardware the data are de-multiplexed using three cascaded counters.

More detailed information about the APV 25 S0 can be found in [Jon99], [Jon00] and [Ray00].

5.2 The COMPASS GEM detector

The COMPASS experiment uses **G**as **E**lectron **M**ultipliers (GEMs) for small angle tracking of charged particles. The GEM detectors were developed by the Gas Detector Development group of the CERN-EP-TA1 division [Sau97], [Bac00].

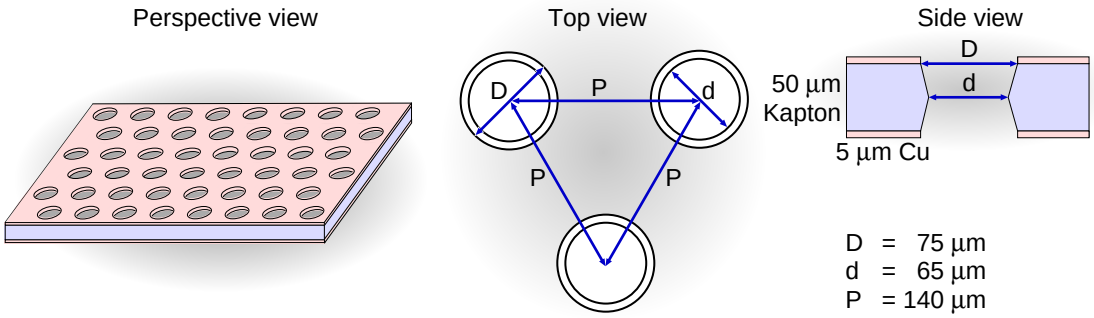


Figure 5.5: The geometric parameters of the COMPASS GEM foil

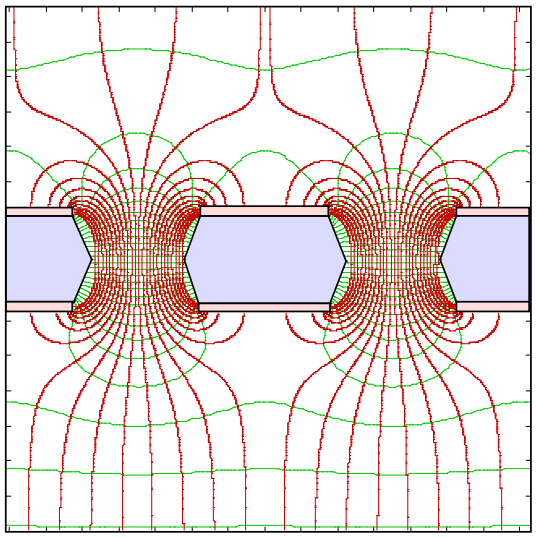


Figure 5.6: The electrical field in the holes of a GEM foil [Bac99]

The functional principle of the GEM is based on thin insulating polymer foils that are coated from both sides with a thin metal layer. The foils used for the COMPASS GEMs are 330 $\times$ 330 mm^2 big and consist of a 50 μm thick Kapton foil, which is coated from both sides with 5 μm of copper. Into this foils a regular pattern of small circular holes is etched, so that they form a two-dimensional hexagonal pattern. Due to the etching process the holes are not cylindrical, but have a double conical shape. The diameter in the metal layer is 75 μm. In the middle of the Kapton layer the diameter is 10 μm smaller. The centers of the holes are spaced by 140 μm (see figure 5.5).

By applying a potential difference between the two metal layers of the foil, strong electric fields are created inside the holes. For example a voltage difference of 200 V

leads to an electric field of about 40 kV/cm. These regions of high field can be used to make avalanche multiplication of electrons. To bring the electrons to the holes, the foils are put into a homogeneous drift field. Figure 5.6 shows the electric field map in a GEM hole, calculated with the field simulation program MAXWELL. For the electrons the GEM foil is nearly completely transparent, because all the electrical field lines, coming from above the multiplier foil, go through the holes to the other side of the foil. The electrons follow these field lines into the holes. There they multiply in avalanches and exit the hole on the lower side. The gain, reachable with a single GEM foil, is well above 10^3. But to get a reasonable signal to noise ratio, gains in the order of $10 \cdot 10^3$ to $50 \cdot 10^3$ are needed. This can be achieved by coupling two or more GEM foils, so that the electrons produced by the first foil go to the next foil, where they make another multiplication.

In the COMPASS GEMs three multiplier foils are stacked one upon each other, so that there are three sequential gas amplification stages. In this way a high gain can be obtained, while at the same time the detector can stably be operated even in a heavily ionizing background, as it is generated by a high energy hadron beam. This is because the single GEM foils can be operated at lower gains, which means lower voltages. Due to the lower voltage discharges between the two copper coated sides of the foil are less probable.

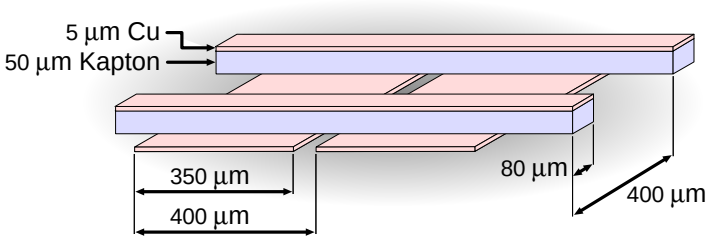


Figure 5.7: The two-dimensional GEM readout [Sim01a]

The two-dimensional readout consists of a PCB[7] with $768 + 768$ copper strips with a pitch of 400 μm (see figure 5.7). The perpendicular strips of the two coordinates sit on two layers, that are separated by 50 μm thick Kapton ridges. The different strip widths of 80 μm for the upper and 350 μm for the lower layer are chosen in a way, that the charge sharing is equal between the two coordinates.

Figure 5.8 shows a schematic cut view of the COMPASS GEM chamber. Two honeycomb plates, one at the top and one at the bottom, provide the mechanical support and stability. To reduce the amount of material in the beam these honeycomb plates have holes in the central region. The drift region is 3 mm thick. The GEM foils and the readout are separated by 2 mm. To support the GEM foils, so that they can withstand the electrostatic forces, a spacer grid, made out of glass fibers, is placed in between them. This grid consists of about 300 μm thick strips, which divide the area of the detector into 16 cells plus the central sector. This grid of course causes

[7] **P**rinted **C**ircuit **B**oard

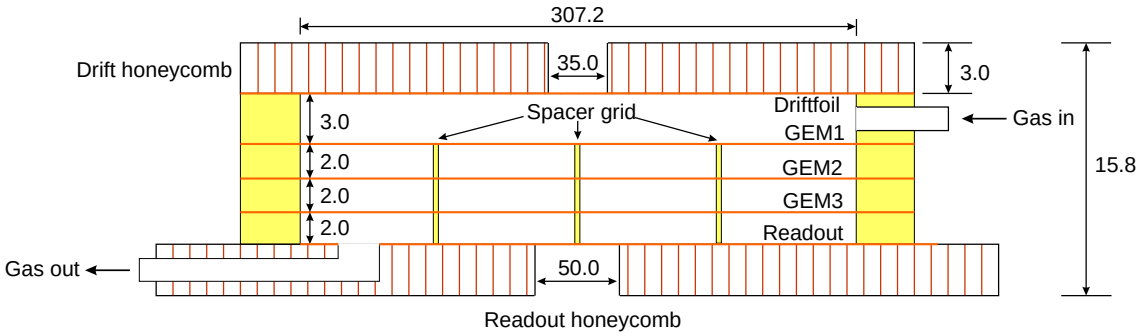


Figure 5.8: A schematic cut view of the COMPASS GEM chamber. All dimensions are in millimeters [Sim01b]

a local loss of efficiency. The frontend cards and the PCB with the readout strips are glued onto the lower honeycomb plate. For noise shielding this honeycomb is covered with a 10 μm thick aluminum foil. The upper side with the readout electronics and the high voltage distribution is wrapped into aluminized Mylar foil, that is supported by a 50 μm thick Kapton frame.

Figure 5.9 shows a completely mounted GEM detector (only the noise shielding on the top is missing). The frontend chips are an integral part of the detector. The APVs are mounted on frontend cards. These are especially thin PCBs, that hold three APVs. Figure 5.10 shows a closeup of such a card. Two frontend cards per coordinate read the overall 1536 channels of a GEM detector. The 12 APVs are connected through two repeater cards to one ADC card.

The APV 25 was developed to read out silicon detectors, so it has no built in protection against too high pulses on the input. Even if due to the triple GEM design the probability for a discharge, induced by heavily ionizing tracks, is quite low, it is still necessary to protect the frontend chips against high signal pulses, which can upset and even damage the entire APV chip.

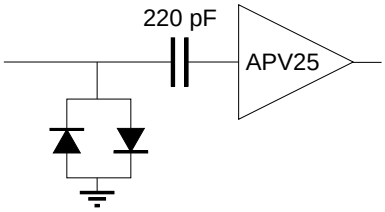


Figure 5.11: The protection circuit in the GEM readout [Sim01a]

The protection circuit sits on the frontend cards and consists of two diodes and a capacitor (see figure 5.11). The diodes ground every signal, that has an amplitude above 0.6 V, which is the bandgap of silicon. Since the APV was designed for silicon detectors[8], it has no built in coupling capacitor at the input. To make an AC coupling to the APV input, a serial capacitor, which is about ten times larger than the strip capacitance, is placed after the diodes. This is done to avoid disturbance or dysfunction of the APV due to saturation of single channels or due to increased power consumption, caused by

[8]Silicon microstrip detectors usually already have a capacitive coupling between the doped silicon strips and the aluminum readout strips, so that there is no capacitor needed to connect the readout strips to the frontend chip

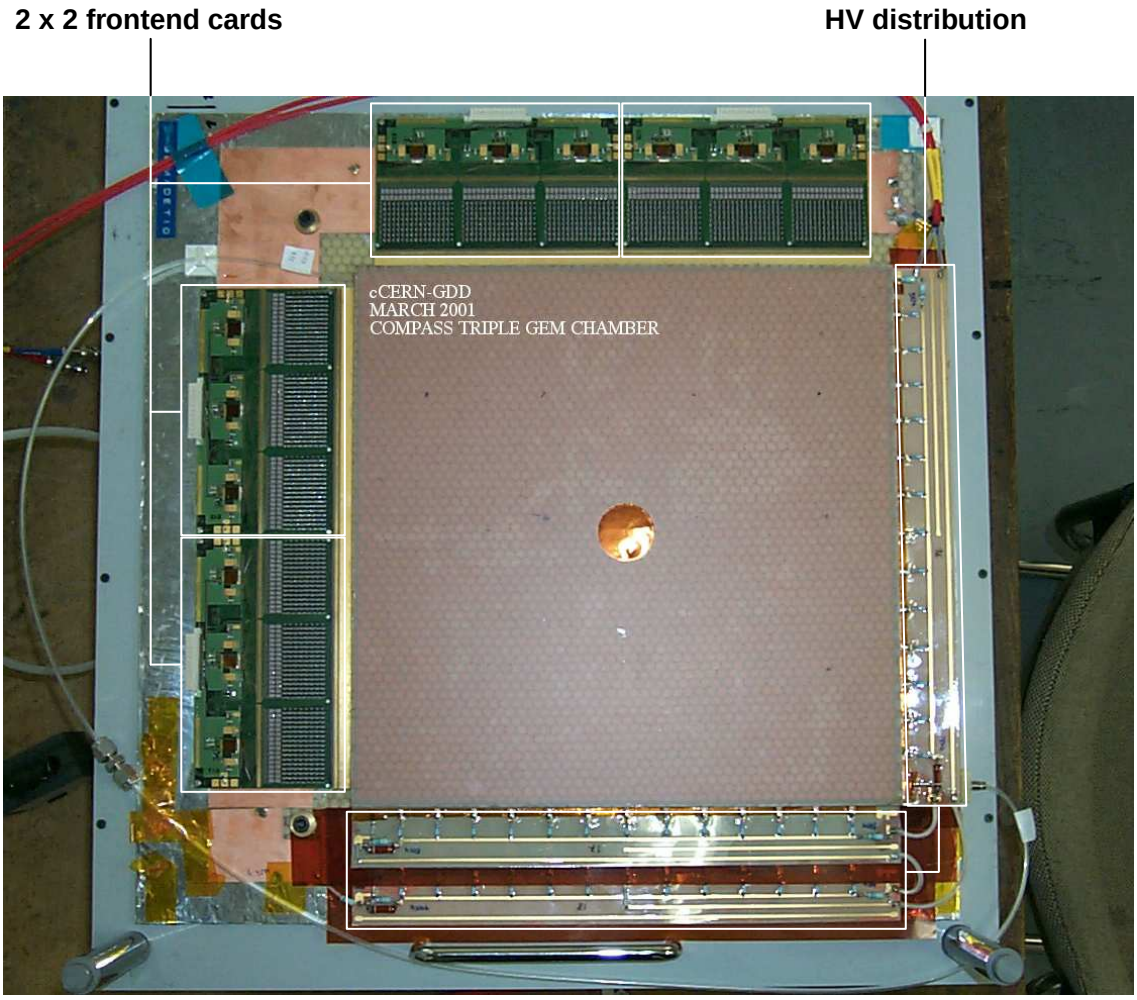


Figure 5.9: A completely assembled GEM chamber [Ket01a]

asymmetric leakage currents of the diodes. Of course the protection circuit affects all other signals as well. The AC coupling lowers the signal amplitude by roughly 10 %. In addition the diodes act like capacitances to ground, which produce extra noise. To compensate the lower signal amplitude and the increased noise, caused by the protection circuit, the gain of the GEMs has to be increased, in order to reach again the full efficiency for MIP detection.

A comprehensive description of the COMPASS GEM detectors can be found in [Sim01a] and [Sim01b].

5.3 The COMPASS silicon detector

In the COMPASS experiment double-sided silicon microstrip detectors are used for the high precision small angle tracking. The detectors were produced by SINTEF

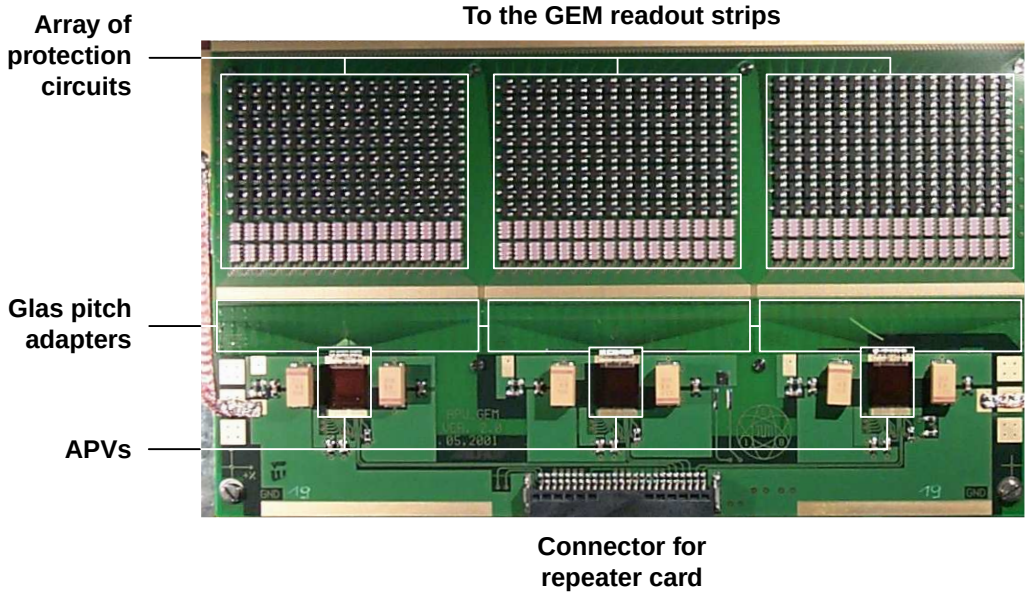


Figure 5.10: The GEM frontend card [Ket01a]

(Oslo, Norway) [Ber97]. The design is based on a development of the Halbleiter-Labor (HLL) of the Max-Planck-Institut für Physik (Munich, Germany), done for the HERA-B[9] experiment at DESY[10] [Abt99]. Figure 5.12 shows a cut view of the SINTEF silicon.

In contrast to gas detectors there is no multiplication of the primary charge, deposited by the ionizing particle in the silicon detector. The signal depends only on the thickness of the detector. The average energy loss is about 390 eV or 108 electron-hole-pairs per micrometer. The high density of the detector material leads not only to a high energy loss, but also the range of the highly energetic secondary electrons is small. This gives a good spatial resolution. To minimize the multiple scattering, the detector should be as thin as possible. The COMPASS silicon detector has a thickness of 280 μm, so that in the average $3 \cdot 10^4$ electron-hole-pairs are created. This signal can be detected with low noise electronics, as it is provided by the APV 25 frontend chip.

In intrinsic silicon electrons are lifted by thermal excitation into the conduction band, leaving holes in the valence band. The density of electrons and holes are equal. At room temperature the intrinsic carrier density of silicon is around $1.45 \cdot 10^{10}$ cm^{-3}. This means in a silicon piece of 1 cm^2 area and 280 μm thickness there are about

[9]HERA-B is a fixed target experiment, using the proton beam of the proton-electron accelerator HERA at DESY

[10]**D**eutsches **E**lektronen **Sy**nchrotron, Hamburg, Germany

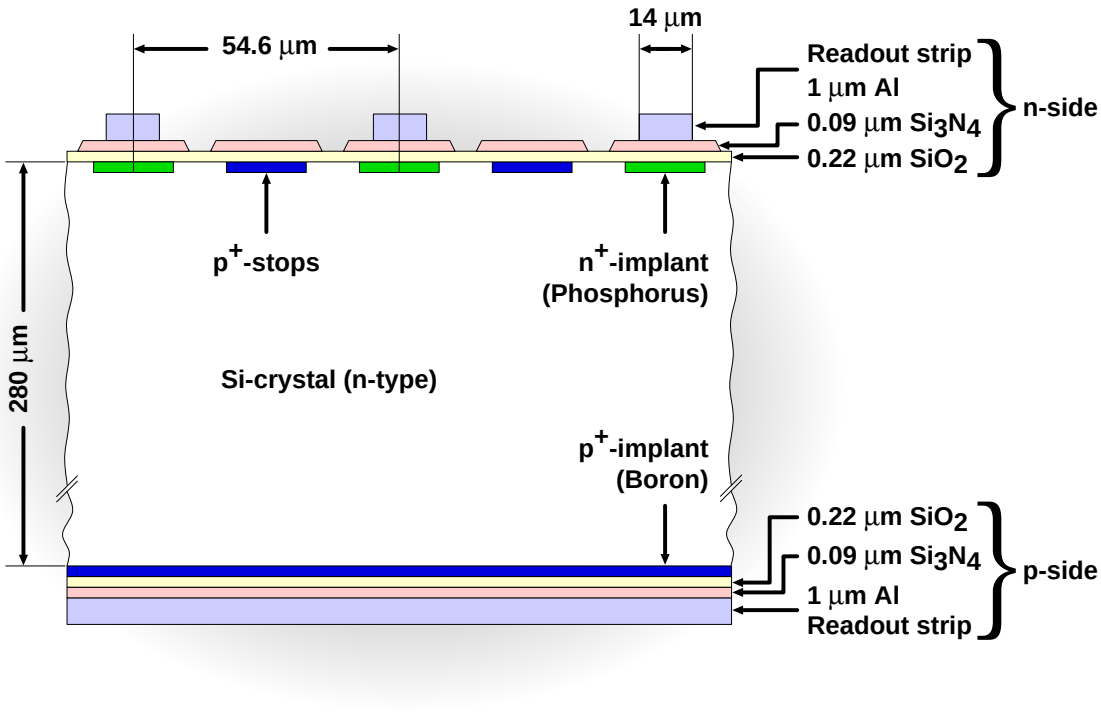


Figure 5.12: A schematic cut view of the SINTEF silicon microstrip detector [Wag01b]

$4{\cdot}10^8$ free carriers, which is four orders of magnitude above the signal to be measured. To lower the number of free carriers the silicon detector is depleted through a reverse biased p-n-junction. A p-n-junction is the interface between two regions of opposite type. The n-region is doped with electron donors, which are elements of the fifth group and thus have five electrons in their valence shell. One of these electrons has an energy level just below the conduction band, so that at room temperature all these electrons are lifted into the conduction band and therefore are the majority carriers. The p-region is doped with electron acceptors from the third group. There holes are the majority carriers.

Due to the gradient in the electron and the hole densities, there is a diffusive migration of the majority carriers across the junction. The carriers recombine in the junction. This leaves a net negative charge on the p-side and a net positive charge on the n-side. The space charge in the depletion zone is created by the immobile acceptor and donor ions in the lattice. The space charge is a barrier for the migration of the majority carriers. In the equilibrium state there is no flow across the junction. The main point is, that in the depletion zone there are no free carriers. Electron-hole-pairs, which are created in the depletion zone, are separated by the electric field of the space charge. The electrons go to the n-side, the holes to the p-side, where they can be read out. But if the electron-hole-pairs are created in the neutral, non-depleted zone, they recombine with the free carriers and are lost. This means that the active region of the silicon detector is the depletion zone. To make

this zone as big as possible, an external bias voltage with the same polarity as the barrier voltage is applied. This increases the height of the barrier and the thickness of the depletion zone. In the ideal case the whole thickness of the silicon is depleted.

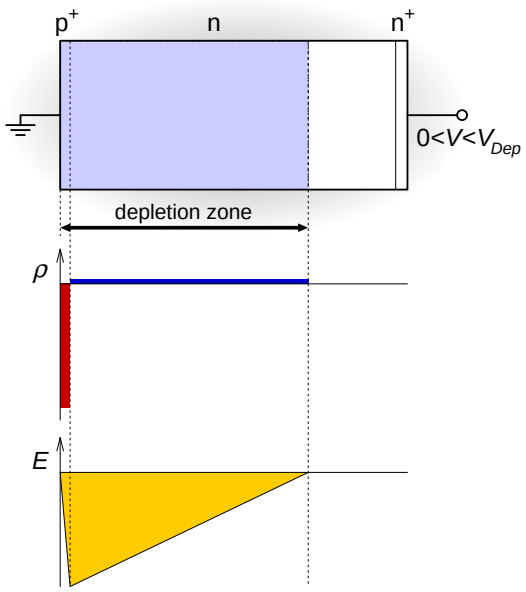


Figure 5.13: The double-sided silicon detector with the p^+-n-junction. Due to the high dopant concentration in the p^+ side the depletion zone extends widely into the only slightly doped n-bulk. The charge density ρ and the electric field E in the depletion zone are shown. The depletion voltage V_{Dep} is reached, when the depletion zone extends up to the n^+-side. [Wag01b]

The bulk of the microstrip detector consists of n-type silicon. Because the depletion zone extends into the p- and the n-region in inverse proportions to the concentrations of the dopants, the material is chosen in a way, that there are much more electron acceptors than donors (p^+-n-junction). This has the effect, that the depletion region is very wide on the n-side and very shallow on the p-side (see figure 5.13). The slightly doped n-region is the active volume of the detector, whereas the p^+-region only depletes the bulk of free carriers and thus can be shallow. To make the detector position sensitive, the p^+-side is divided into an array of narrow strips. With a strip pitch in the same order as the width of the charge cloud, created by an ionizing particle, one can determine the position of the particle by reading out the strips. The SINTEF silicon has 2047 strips with a pitch of 25.875 μm on the p^+-side, but only every second strip is connected to the readout. The readout strips have a width of 15 μm, the intermediate strips are 8 μm wide. The intermediate strips enhance the spatial resolution by capacitive charge division. The charge collected on the intermediate strips is transferred by capacitive coupling to the neighboring readout strips with the effect, that more events have a double hit. Double hits allow the determination of the center of gravity of the charge distribution and thus improve the achievable spatial resolution, compared to single hits, where the track position is only given by the strip number [Pei92].

The other side of the silicon detector has a layer of highly doped n^+-silicon, that prevents the depletion zone from reaching the back plane of the detector. The n+-layer also gives a good ohmic contact. The COMPASS silicon detector is double-sided. This means, that the n^+-side is also segmented into strips. The strips are perpendicular to the ones on the p^+-side. In this way one detector can measure two projections of the particle crossing. To allow a readout of the p^+- and n^+-strips, they are metallized with aluminum. A thin insulating silicon oxide layer between the silicon and the aluminum creates a capacitance, so that the signals go via AC coupling to the readout chip. To reduce the number of pinholes in these

coupling capacitors and to lower the probability for breakdown, there is an additional thin intermediate layer of silicon nitride. The silicon nitride layer is also segmented into strips (see figure 5.12).

Due to the production process of the detector, fixed positive oxide charges are introduced in the silicon oxide layer at the interface to the silicon. The positive charges attract electrons, so that they form an electron accumulation layer connecting the n^+strips. The electron accumulation layer acts as a conductor and short-circuits the n^+-strips. This leads to a wide spread of the charge, so that a position measurement of the particle crossing is nearly impossible. To insulate the n^+-strips from each other intermediate p^+strips – so called p-stops – are implanted between the n^+-strips. On the n^+-side the COMPASS silicon detector has 1280 n^+-strips that are 15 μm wide and 1279 15.667 μm wide p^+-strips both with a pitch of 54.667 μm. Only the n^+-strips are connected to the readout. Because there are no intermediate strips and thus no capacitive charge division, the cluster size[11] of the n-side is smaller, compared to the one of the p-side, which means, that the spatial resolution is worse.

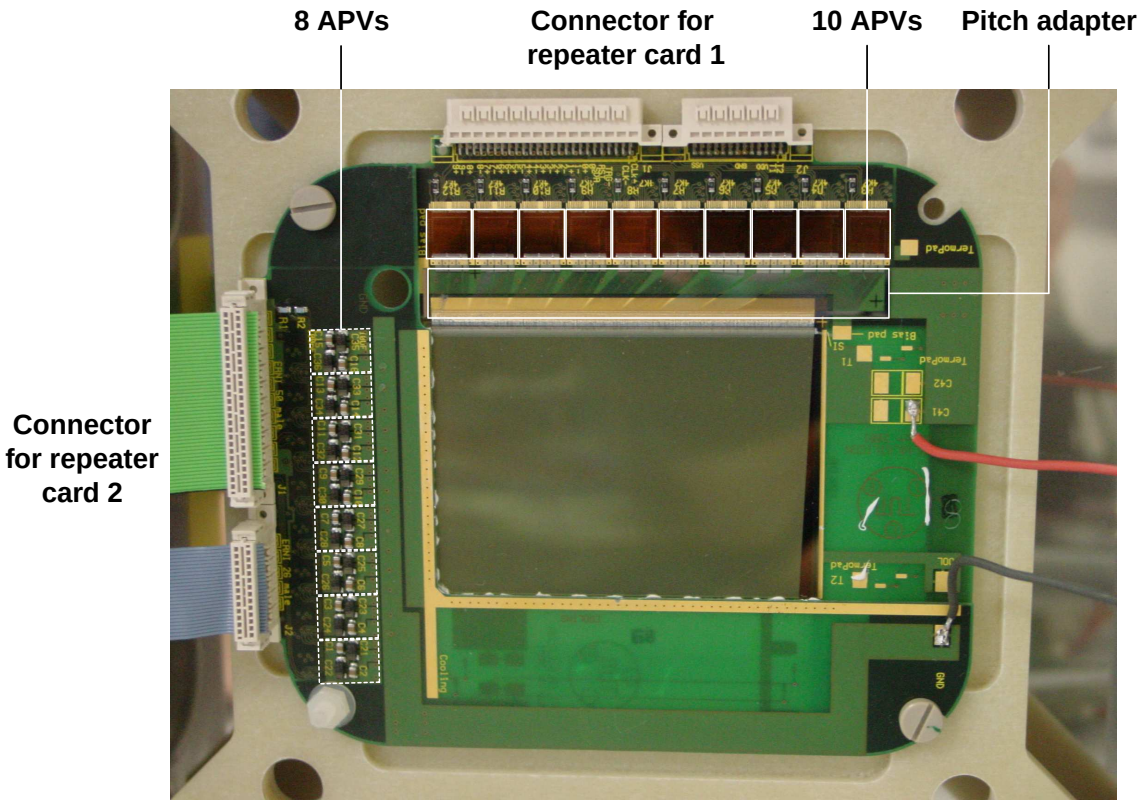


Figure 5.14: The COMPASS silicon detector [Wag01b]

The APV 25 frontend chips are mounted on two L-shaped boards around the detector. The 1280 channels on the n-side are read by 10 APVs the 1024 on the p-side

[11] The cluster size is the number of hit strips per particle crossing

by 8 APVs. The signals from each side go via a repeater card to one ADC card. Because of the bias voltage, that depletes the detector, the readout strips on the n- and the p-side of the detector lie on different potentials and so do the two L-boards and the connected ADC cards. But since the ADC cards have only an optical connection to the GeSiCA readout module, they are electrically completely insulated from each other and from the readout module. Figure 5.14 shows the photograph of a mounted silicon detector.

5.4 System overview

The GEM and silicon readout system consists of 4 parts (see figure 5.15):

- The Frontend chip APV 25 S0
- The Repeater card
- The ADC card
- The GeSiCA readout module

The analog differential outputs of the APVs go via short flat-cable through a repeater card to the ADC card. In case of the GEM detector one ADC card handles all twelve APV chips of a whole chamber. In the silicon frontend one ADC card reads ten (for the 1280 channel n-side) and eight (for the 1024 channel p-side) APVs respectively. The analog signals are digitized by 10 bit differential ADCs. The zero suppression logic processes and sparsifies the digital signals. After formatting the data to 32 bit words, they go via long optical fibers to the GeSiCA readout module. For this data path the HOTLink protocol[12] is used. Aside from offering high bandwidth for data transfer, the optical fibers insulate the ADC cards from the readout module. This is especially important for the silicon frontend, where the two ADC cards, that read out one silicon detector, are sitting on different potentials.

The GeSiCA module is a 9U VME module, able to process up to four incoming HOTLink data streams, which is equivalent to 48 APVs or 6144 channels. After de-serialization the data are buffered in a FIFO. Via a common bus the merger unit takes the data from the FIFOs, labels them with the event header, it gets from the TCS receiver, and writes the formatted data to the S-Link card which sends them to the Readout Buffer.

GeSiCA also encodes the TCS reference clock, the trigger and the reset signal, provided by the TCS receiver, and distributes these signals through the optical

[12]The **H**igh-speed **O**ptical **T**ransceiver **Link** is a high speed serial point to point link developed by CYPRESS [Cyp01]. It offers a bandwidth of 160 to 330 Mbit/sec and works with several transmission media like fiber-optics, coax cable or twisted/parallel pair cable.

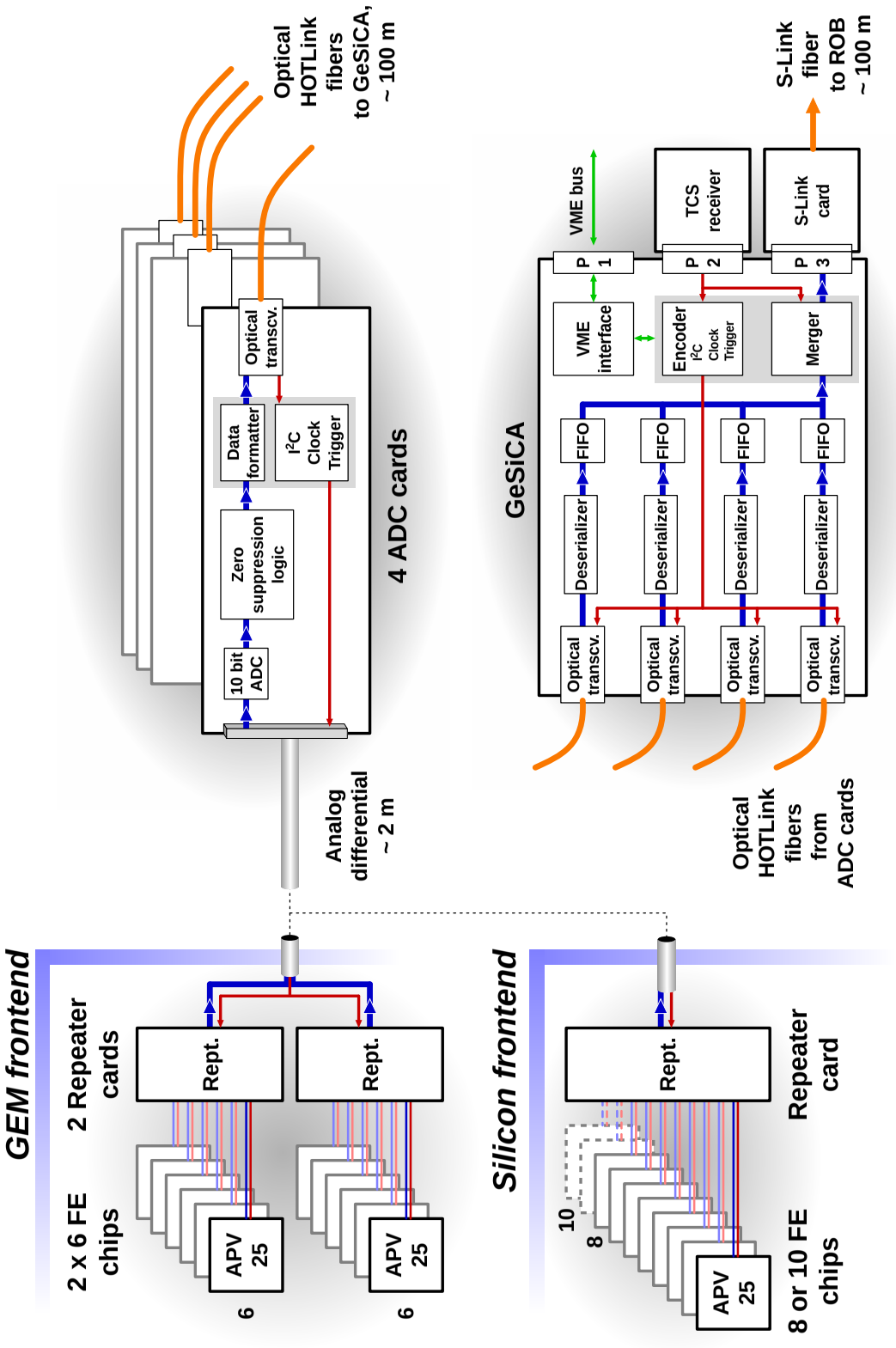


Figure 5.15: The common GEM and silicon readout system

fibers to the ADC cards and the connected APVs. To allow easy configuration and status monitoring of the frontend electronics, which means ADC cards and APVs, the GeSiCA provides an I^2C interface, which is accessible via the VME bus (see section 5.8). An encoder transforms the I^2C protocol, so that it can be sent together with the trigger and reset signals over the optical fiber. The ADC card decodes the I^2C signals from the fiber and forwards them to the APVs.

5.5 The data format

The COMPASS experiment uses the data acquisition software framework DATE version 3.71 for the processing of the data, received by the Readout Buffers [DAT]. This software was developed by the DAQ group of the ALICE experiment [AL95], that is planned at CERN's LHC. The package provides programs and routines for readout, monitoring, error reporting, run control and event building. All data recorded with the COMPASS DAQ have the DATE raw data format. The data words are always 32 bits wide.

The hierarchical tree-like structure of the DAQ is represented by a nested data structure with several data blocks, preceded by headers (see [DAT] and [Fis00]). The events are assembled by the **G**lobal **D**ata **C**ollector (GDC) software, that runs on the event builder PCs. This software performs the event building and generates the global event headers, which frame the event data. The GDC gets the event data fragments from the **L**ocal **D**ata **C**oncentrators (LDCs). This kind of program runs on the Readout Buffer PCs. The LDCs mark the data, they get from the readout modules, with a subevent header. The subevent data blocks are divided into several parts, representing the different equipments, the LDC reads. Each equipment data block again is identified by an equipment header.

The readout modules have to provide all the necessary information like event and spill number, equipment number, trigger type, event size, data format description, status and error information, so that the LDCs and GDCs can build the different headers. All this information is provided by the S-Link header. This header precedes the data block, that is sent out for every event by the readout modules. The definition of the S-Link header is shown in table 5.1. All S-Link data transfers are framed by begin (`0x00000000`) and end markers (`0xcfed1200`). These words are removed by the Readout Buffers, so that they are not visible in the recorded data.

The header itself consists of three words. The first word contains the 5 bit trigger type, the 10 bit source ID and the 16 bit event size. The trigger type comes from the TCS receiver. The source ID is the unique equipment number of the readout module. It is set via the VME bus and defines, which basic decoding routines must be applied to the data. The lower 16 bits contain the event size, which is the number of 32 bit words. In contrast to the header words the start and end marker words are not included in the event size. The second word of the S-Link header carries the

31										20								10								0

S-Link begin marker

0x00000000

S-Link header

0	Tr. type (5)	Source ID (10)	Event size (16)
0	Spill no. (11)	Event no. (20)	
0x0000		TCS error byte	0x01

⋮

Data

⋮

S-Link end marker

0xcfed1200

Table 5.1: The definition of the S-Link header

31										20								10								0

ADC header

10000000		ADC ID (8)	ADC block size (16)
11	Data format (6)	Time tag (24)	

Table 5.2: The definition of the ADC header

31										20								10								0

APV header

Chip ID (4)	Local event no (12)		APV block size (16)
10000	APV header 3rd frame (9)	APV header 2nd frame (9)	APV header 1st frame (9)

Table 5.3: The definition of the APV header

31										20								10								0

Latch all data format

1	SF	ADC value 3rd frame (10)	ADC value 2nd frame (10)	ADC value 1st frame (10)

Sparse data format

Channel no. (7)	ADC value 3rd frame (9)	ADC value 2nd frame (8)	ADC value 1st frame (8)

⋮

Last word

1	0	Baseline offset 3rd frame (10)	Baseline offset 2nd frame (10)	Baseline offset 1st frame (10)

Table 5.4: The definition of the APV data formats

20 bit event number and the 11 bit spill number, that are delivered by the TCS. The error byte from the TCS receiver is sent with the last S-Link header word.

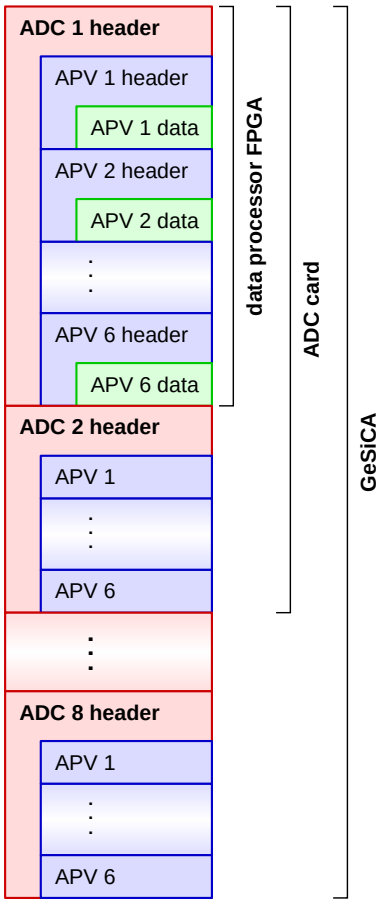


Figure 5.16: The structure of the S-Link data block for the GEM and silicon detectors, sent by the GeSiCA module

The data after the S-Link header are divided into smaller blocks according to the APVs and the data processor FPGAs on the ADC cards, that are read by the GeSiCA module (see figure 5.16). One GeSiCA board reads maximum four ADC cards with eight data processor FPGAs, each connected to six APVs. For each data processor FPGA an ADC header is generated, which consists of two words. Their definition is shown in table 5.2. The first word contains the 8 bit ADC ID, that distinguishes the chips on the different ADC cards. The lower 16 bit are reserved for the ADC block size, that gives the number of 32 bit data words (including headers), sent by the data processor FPGA, specified by the ADC ID. In the second word bits 24 to 29 describe the data format in more detail. Bit 24 states whether the APVs are read out in the Peak mode ('0') or in the Multi mode ('1'). If bit 25 is set to '0', the data are in the latch all format, which means all channels are written out (see below). A '1' in bit 25 means, that the data are zero suppressed. Bit 26 says whether the pedestals were subtracted ('1') or not ('0'). Bit 27 has the same function for the common mode offset correction. Bit 28 is not used and the 29th bit states, that the data processor FPGA had to cut the event ('1'), because the data buffers on the ADC card went full. The rest of the second ADC header word is used by the time tag. The time tag gives the arrival time of the trigger at the data processor FPGA in units of 3.2 μs (64 cycles of the 20 MHz clock).

The ADC data block consists of six APV data blocks, each preceded by a two word long APV header (see table 5.3). The first word of the APV header contains the 4 bit chip ID, the local 12 bit event number and the 16 bit APV block size. The chip ID identifies the different APVs, that are read out by the data processor FPGA. The local event number is generated by the data processor FPGA. It can be used to check the synchronicity with the TCS and also among the different ADC cards. The APV block size gives the number of 32 bit words (including header words), coming from the APV, specified by the chip ID. The second APV header word carries the decoded APV headers, taken from the digital part of the three APV frames. The 9 bit APV header includes the 8 bit column address and the error bit.

The APV data blocks can have two different formats: latch all and sparse (see table 5.4). In the latch all mode the ADC values of all 128 channels per APV are written out with the full 10 bit resolution. This mode of course creates big amounts of data and is only intended to be used for debugging and calibration purposes. One 32 bit data word contains the three ADC values of one channel. The 128 channels are put out in ascending order. Bit 30 is the suppression flag. It says whether this channel contains a hit ('0') or would have been suppressed ('1'), if the sparsified mode would be active. This allows to check the performance of the zero suppression algorithm.

In the sparse mode the pedestal subtraction and the common mode noise correction are activated. In the Multi mode the timing is normally set in a way, that the three samples are taken at the rising edge of the signal, with the third sample sitting in the vicinity of the signal peak (see section 5.1). Only the channels, where the amplitude of the third frame is above a certain threshold value, are recorded. Compared with the latch-all mode, the ADC values are written out with a lower precision, so that the values of the three frames fit together with the seven bit channel number into one 32 bit word. Because the third sample sits at the signal peak, this ADC value is put out with a nine bit precision, where only the least significant bit is discarded. Since the other two samples are expected to have smaller amplitudes, for them in addition the most significant bit is omitted.

The last word of the APV data contains the baselines of the three APV frames, computed by the common mode noise correction unit in the data processor FPGAs (see subsection 5.6.1). The format of this word is the same as for the latch all data words. The values can be used to check the common mode noise correction.

The full description of the GEM and silicon data format can be found in [Ket01b].

5.6 The ADC card

The main task of the ADC card is the digitization and sparsification of the data, it gets from the maximum 12 APVs, that can be connected to the board (see figures 5.17 and 5.18). A big FIFO buffer de-randomizes the data, before they are sent through the optical HOTLink connection to the GeSiCA board.

Each APV input is connected to a 10 bit differential ADC, that digitizes the analog signals at a rate of 20 MHz. The main data processing is done by two FPGAs each connected to six ADCs. They perform the zero suppression including common mode noise correction and format the data into 32 bit words. Inside the chip the six data streams of the APVs are processed in parallel by a pipeline, that does the zero suppression in four steps. First the incoming data are reordered to restore the physical channel order. On the second stage the pedestals for each channel are subtracted. Then the common mode noise offset is computed and at the last step a threshold cut is applied to the common mode noise corrected data. The fourth stage

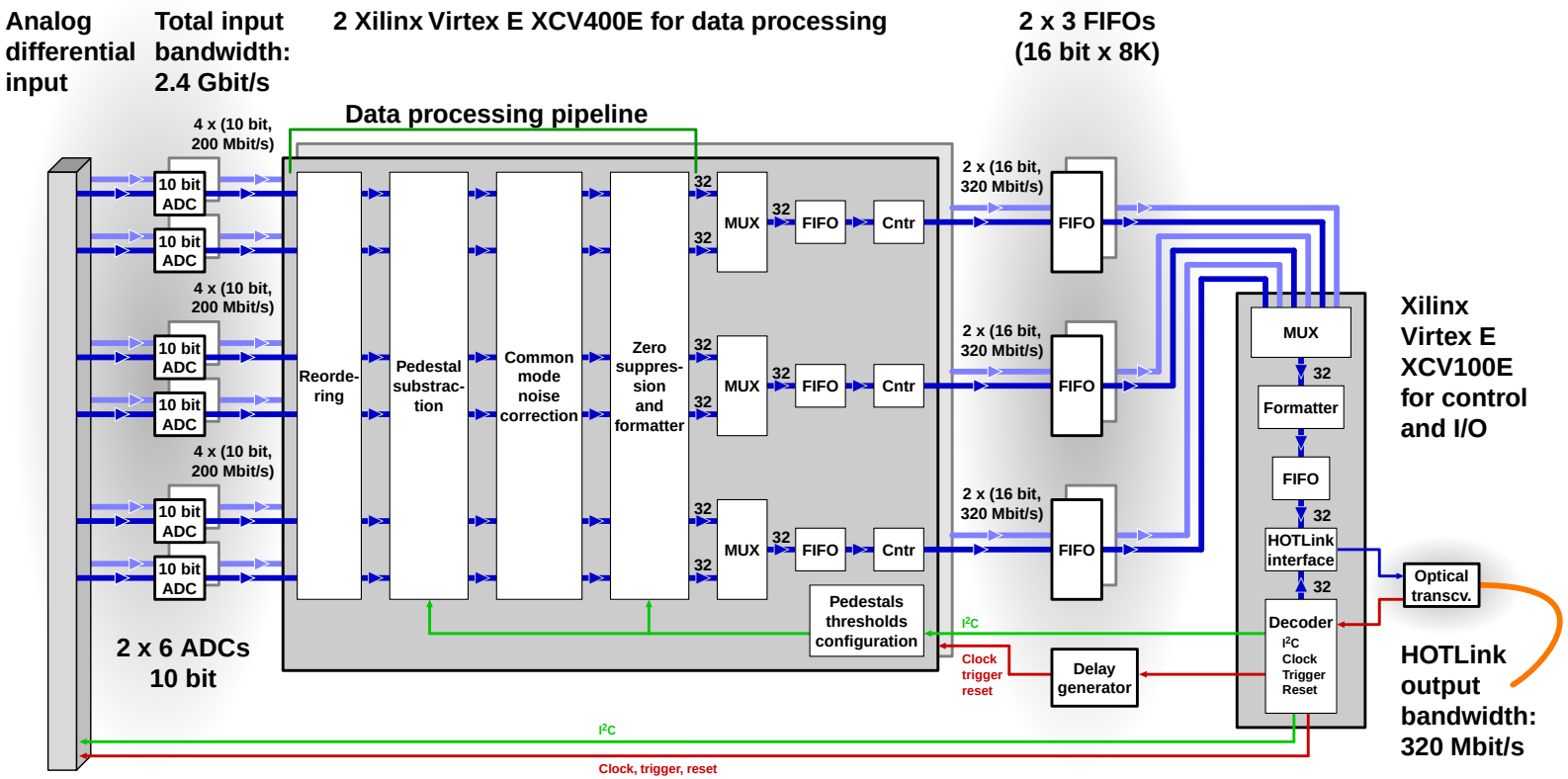


Figure 5.17: Schematic view of the ADC card

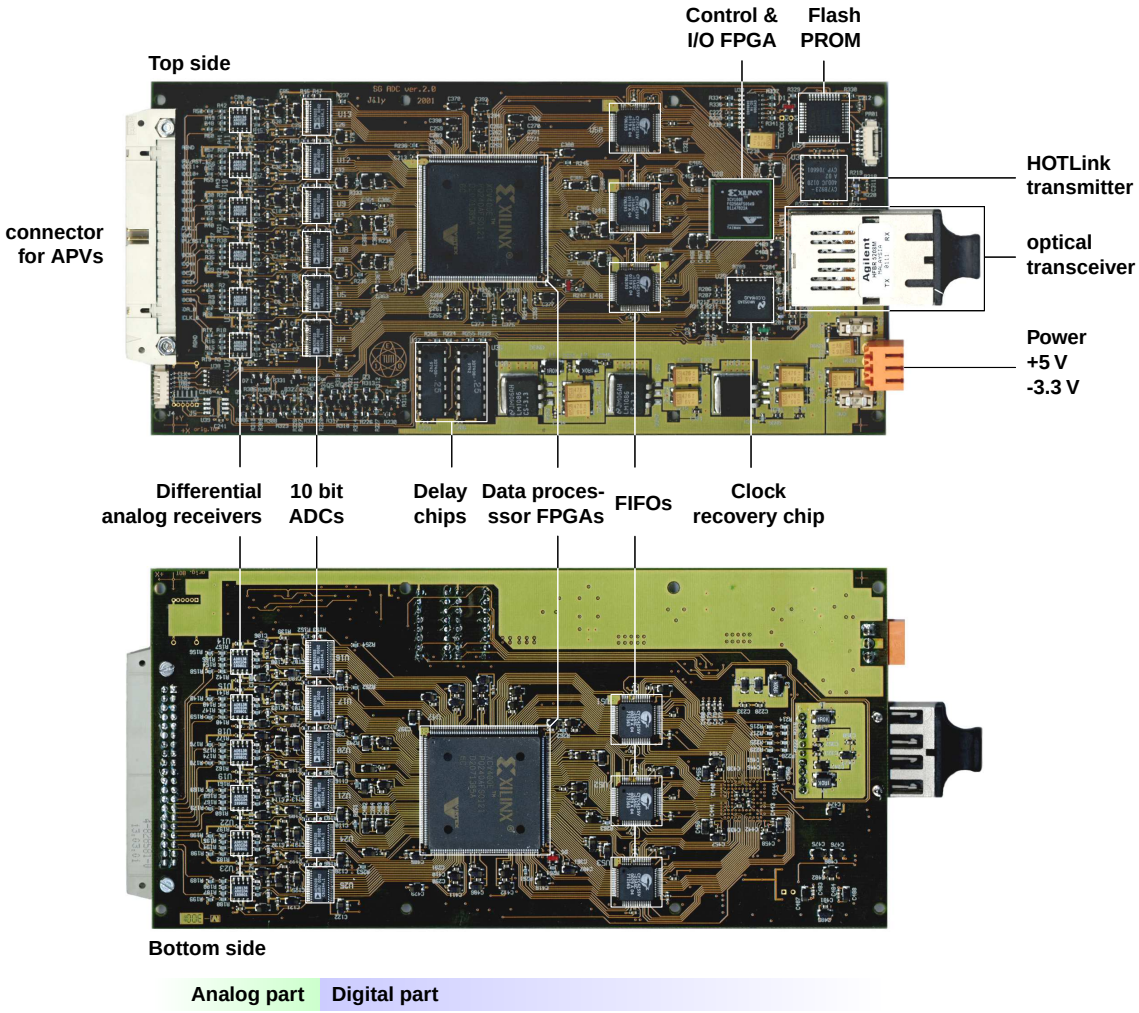


Figure 5.18: The two sides of the ADC card

also formats the data and tags the start and the end of the data blocks, belonging to the different APVs.

The 32 bit wide data streams of two APVs are then sequentially multiplexed into an internal output FIFO. Afterwards the data are split into two 16 bit words and written into an external FIFO. These CYPRESS CY7C4255V FIFOs are 8K words deep and 18 bits wide, so that they can buffer each up to 4K 32 bit wide data words plus two extra bits, that are used for data flow control.

The six CYPRESS FIFOs for the two data processor FPGAs are read by the data control and I/O FPGA. This chip rejoins the two 16 bit words from the FIFOs to 32 bit data words and sequentially multiplexes the six incoming data streams. It also performs a final formatting of the data by adding headers for the different data blocks and calculating the data block sizes. After that the data are written into an internal FIFO, which buffers them for the output via the optical HOTLink.

The HOTLink transmitter sends the data byte-wise. Therefore the 32 data words are split into four bytes. The serialized and encoded data from the HOTLink transmitter go to the optical transceiver, which sends them through the optical fiber to the GeSiCA readout module.

From the signal of the optical receiver a decoder unit extracts the trigger and the reset signals. The decoder unit in addition provides an I^2C interface. It decodes the I^2C transfers sent by the GeSiCA module through the optical fiber. For the transfers in the opposite direction the HOTLink transmitter is used. The clock signal, which is retrieved by the clock recovery chip, the I^2C bus, the trigger and the reset signal are distributed to the APV chips and go via delay chips to the two data processor FPGAs. From there the delayed clock signal is distributed to the ADCs. The programmable delay chips allow to compensate the phase difference between the data clock of the APVs and the clock of the ADCs, introduced by the cable connection between APV and ADC. The I^2C bus is used to configure the data processors and the APVs (see section 5.8) and to set the delay chips.

The system has to be able to run with the maximum trigger rate, permitted by the APV. In this extreme case the APV constantly sends out frames, which after digitization give data flow of 200 Mbit/s (10 bit ADC at 20 MHz). This means, that the zero suppression has to reduce the amount of data by at least a factor of 7.5 to match the combined input bandwidth of 2.4 Gbit/sec of the twelve ADCs with the bandwidth of the HOTLink output of 320 Mbit/sec. Short-time variations in the output data rate are compensated by the big CYPRESS FIFO buffers, but in average the zero suppression logic has to discard at least 87% of the channels, which means 111 out of the 128 channels per APV.

5.6.1 The data processor FPGA and the zero suppression

The two data processor FPGAs on the ADC card are Xilinx Virtex E XCV400E chips [Xil01c]. They each read six ADCs and apply different corrections to the data, so that at the end a threshold cut can discriminate the channels with a hit. Because zero suppression means discarding data, special care about the noise has to be taken. There are two types of noise in the system. First there is the 'detector noise', that consists of thermal noise, noise from the capacitances, noise from the amplifiers and so on. This kind of noise is individual for each channel and there are no correlations between the channels. The second type of noise is the so called 'common mode' noise. It is the component of the noise, whose amplitude is the same for all channels of one APV chip. This means, the common mode noise describes the fluctuation of the baseline of the APV chip. This effect has to be corrected in the hardware before the threshold cut is applied. The same holds true for the slow baseline walks due to changes in the operating conditions of the detector.

The hardware algorithm for the zero suppression with common mode noise correction originated from the software algorithm, used to analyze data, taken in the latch all

mode [Ket01a]. The software first reorders the channels. Then the pedestals are subtracted, which shifts the baseline of the signal in the region around zero. The calculation of the real signal baseline of the APV frame uses an array, in which the channels of the chip are sorted by their amplitude (see figure 5.19).

The program calculates the median value of the sorted amplitudes. In case there is an uneven number of channels, because some of them were flagged to be ignored, the median is just the channel in the middle of the array. For an even number of channels the median is the mean value of the two amplitudes, that lie around the middle of the array. The median amplitude `MEDIAN_AMP` is used to identify and cut away the hit strips using a threshold cut:

(1) `AMPLITUDE[i] - MEDIAN_AMP` $\geq$ `THRESHOLD`

Where `AMPLITUDE[i]` is the array value at index `i` and `THRESHOLD` is set to 20. This cut is done, because the hit strips do not contain any information about the baseline of the APV.

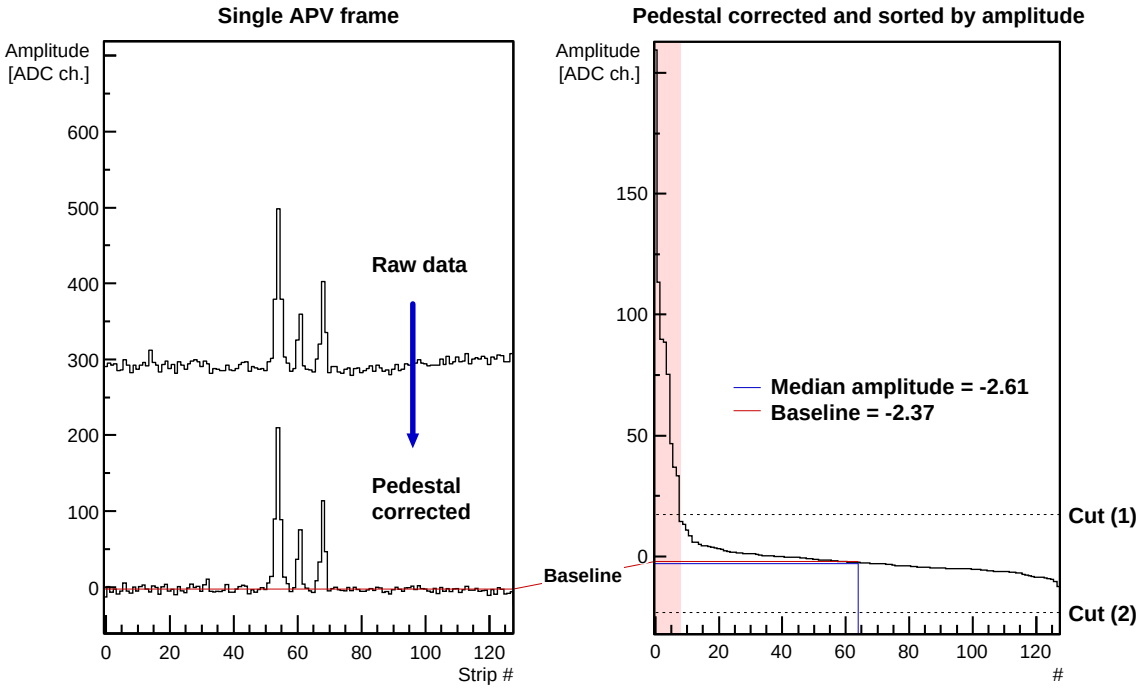


Figure 5.19: The software algorithm for the common mode offset calculation: The left graph shows an APV frame of a GEM detector with some hits. In the diagram on the right the pedestal corrected data are sorted according to their amplitude. The blue line marks the median of the frame. The red line is the calculated baseline of the frame. It is the mean amplitude of the strips in the white region of the diagram. The strips in the red region were excluded from the mean calculation by the cuts (1) and (2) (see text). The cuts avoid, that hit strips distort the baseline calculation. One can see, that the median amplitude and the real baseline are very close. [Ket01a], [Fri01], [CBT01a]

To keep the array of the sorted amplitudes symmetric and unbiased, not only the high amplitudes of the hit strips, but also some strips with low amplitudes have to be cut away:

(2) `AMPLITUDE[CENTER + NR_CUT] - AMPLITUDE[i]` $\geq$ `THRESHOLD`

The reference amplitude for this cut is not the median amplitude `MEDIAN_AMP` $\approx$ `AMPLITUDE[CENTER]`, but the one with the index, that is shifted by the number of strips `NR_CUT`, which have been cut away by condition (1).

After the two cuts the array contains only non hit strips (see figure 5.19). The mean value of these channels is taken as the real baseline of the signal and the APV frame is corrected for this. As the last step a threshold cut on the amplitude of the third frame identifies the hit channels.

Of course in the hardware the sorting of 128 channels according to their pulse height is not feasible within the 140 clock cycles, the FPGA has to process one APV frame. To determine the fluctuating baseline of the APV the hardware implementation uses a different but similar approach [Kon01]: The hardware algorithm histograms the accumulated frequency of the amplitudes within a window around the assumed baseline of the frame. The utilized histogram has 32 bins. The entry in each bin is the number of channels, that have an amplitude bigger or equal to an assigned bin 'value', defined as two times the bin number, so that amplitudes in the interval from 0 to 62 are histogrammed. If the strip amplitudes lie in the proper range, the baseline of the frame is approximately the biggest bin value, that has an entry bigger or equal to 64. This is because compared to this bin value, at least half of the channels in the APV frame have an amplitude bigger or equal, so that this value is close to the median amplitude.

Because of the limited dynamical range of the histogram, the amplitudes have to be shifted in the right region, so that the histogramming can work. Therefore the hardware calculates the average pulse height of the pedestal corrected APV frame. If the frame contains no hits, the average pulse height lies very close to the median amplitude. Signals obviously create a difference of the average and the median amplitude, but for sufficiently low occupancy the average pulse height gives an estimate for the median amplitude, close enough for the histogramming method to work. The channel amplitudes are shifted in a way, that the average amplitude comes to lie on value 32, in the center of the histogram explained earlier.

To save hardware resources the pedestals, that are loaded into the data processor FPGAs, are by 100 ADC units smaller than the real pedestals, so that after the pedestal subtraction the baseline of the frame and thus the average amplitude sits around 100. This means, that except from extreme cases, the average amplitude is always bigger than 32, so that the hardware has to shift the amplitudes only in one direction. This saves a comparator and the sign logic for the adder/subtracter.

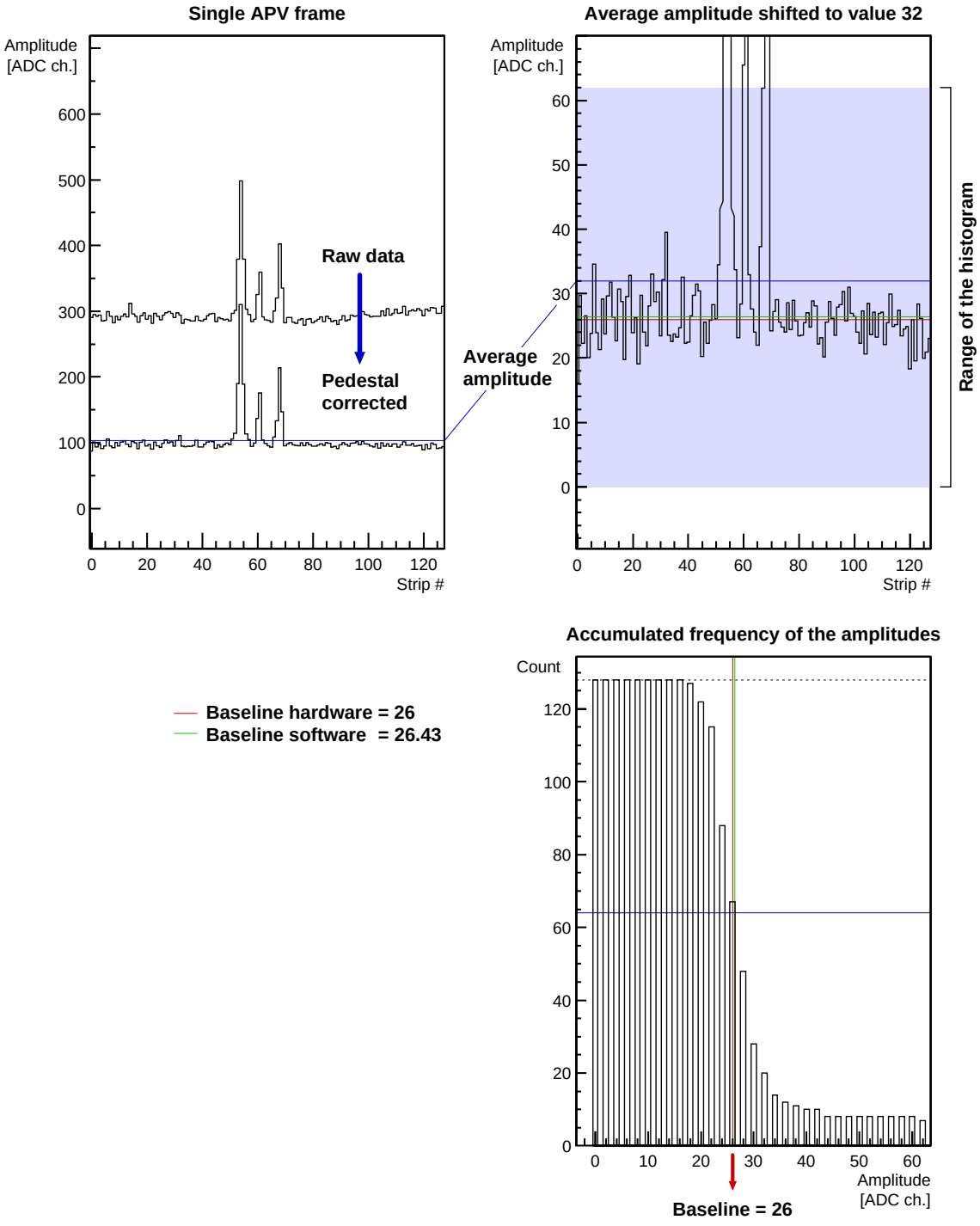


Figure 5.20: The hardware common mode noise correction algorithm (same frame like in figure 5.19): Due to the smaller pedestal values the baseline of the pedestal corrected frame sits around 100. From this data the hardware calculates the mean amplitude (blue). Then the frame is shifted down, so that the average amplitude lies at 32. The upper right graph shows an enlarged view of the frame with the histogramming range. The histogram shows the accumulated frequency of the amplitudes in the frame. The biggest amplitude with an entry above 64 is taken as the baseline of the signal (red). For comparison the baseline calculated by the software algorithm is drawn in (green). [Kon01], [Fri01], [CBT01a]

The full algorithm used for the hardware zero suppression of one APV frame comprises of six steps (see figure 5.20):

1. Pedestal subtraction
2. Calculation of the average amplitude of the APV frame
3. Shifting of the amplitudes so that the average amplitude lies at a value of 32
4. Computation of the baseline of the frame using a histogram with 32 bins
5. Subtraction of the baseline
6. Threshold cut on the amplitude of the third frame to suppress the channels without a hit

The hardware algorithm is implemented as a pipeline, that processes the data of the six ADCs, that are read by one data processor FPGA, in parallel. The pipeline has multiple stages, that perform the above steps of the zero suppression algorithm. Figure 5.21 shows the pipeline in detail.

The input stage processes the six incoming 10 bit ADC values. It decodes the digital information at the beginning of the APV frames. The resulting 9 bit APV header value contains the 8 bit column address and the error bit. The 10 bit values of the analog samples of the six ADCs are joined together to a 60 bit wide bus. Due to the output multiplexing of the APVs the channels have to be brought back to their physical order. This is done using a 128 word deep RAM buffer (see below). The write address generated for this RAM corresponds with the physical channel number. For monitoring of the synchronicity the FPGA generates a local event number, which can later be compared with the event number generated by the TCS.

The input stage also subtracts the pedestals from the incoming ADC values. The pedestals are written into the pedestal RAM of the FPGA (see section 5.8), which stores the 128 pedestals of all six APVs, read out by the FPGA.

The second stage of the pipeline computes the average pulse height of the pedestal corrected data of one frame. This value is later used to perform the common mode noise correction. Because the number of channels is a power of two, the calculation of the average value is quite easy: six 17 bit counters accumulate the ADC values. The average amplitudes are given by the upper 10 bits of the 17 bit counter values.

Two stages of the data processing pipeline, the common mode offset calculation and the output stage, rely on results of the previous stage, that can be obtained only *after* the processing of the data of the whole APV frame. This requires the data to be buffered. Due to the bandwidth requirements of the output stage the buffer has a special structure. It consists of two pages, each having three blocks of single ported memory and some registers. One of the two pages is in write mode, while the

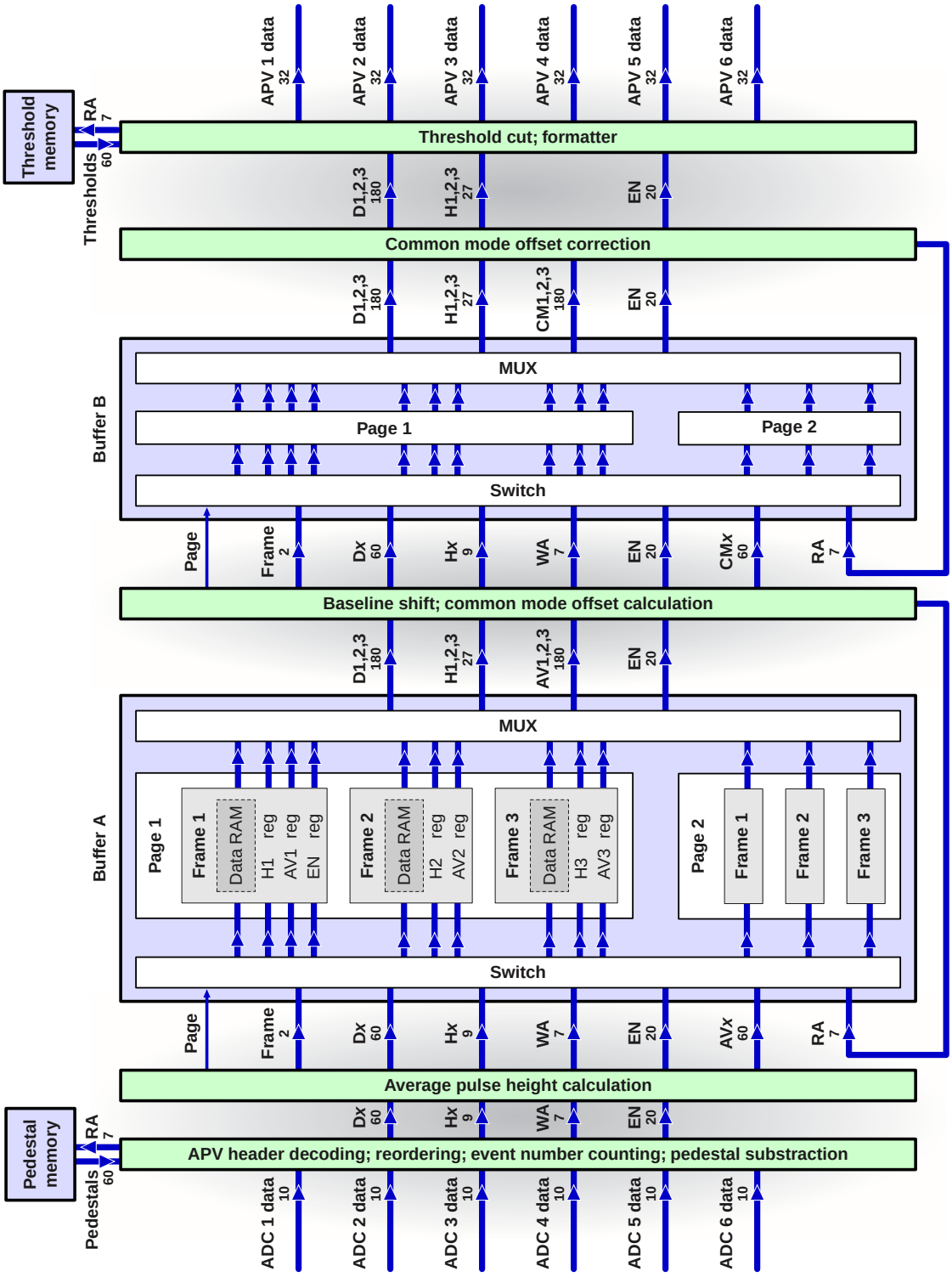


Figure 5.21: This is a schematic view of the pipeline in the data processor FPGA. The different steps of the pipeline are decoupled by two special buffers. The abbreviations for the data busses have the following meaning: 'Dx' - APV data of frame x, 'Hx' - digital APV header of frame x, 'AVx' - average amplitude of frame x, 'CMx' - common mode offset of frame x, 'WA' - write address for RAM, 'RA' - read address for RAM, 'EN' - local event number

other page is in read mode, so that the buffer can be filled with new data, while the data from the previous event are still read by the downstream pipeline stage. This concept is known as 'swinging buffers'. The three 60 bit wide and 128 words deep RAM blocks in each page store the data of the three frames, that the APVs send in the Multi mode. The input signals `PAGE` and `FRAME` determine, to which page and which memory block the input data are written. The write address specifies the destination memory cell. In addition also other data like the APV headers and the local event number have to be stored together with the frame data. Therefore registers are used (see figure 5.21). On the output side of the buffer a multiplexer switches the data of the two pages. The output busses combine the data of the three frames into 180 bit (frame data, average amplitude, common mode offset) and 27 bit (APV headers) wide busses respectively. This gives a peak output memory bandwidth of 3.6 Gbit/sec (180 bits at 20 MHz). The memory data on the output are selected with the read address bus. For simplicity the two buffers A and B have the same structure, but the design of buffer A might be a point for future improvement and optimization.

The third stage computes the common mode offset of the APV frame. Therefore the data are shifted in a way, that the average pulse height, calculated in the stage before, comes to lie on value 32. Then the accumulated frequency of the amplitudes in the range from 0 to 62 is histogrammed and from this, the real baseline of the APV frame is calculated. While flowing through the 'common mode' stage, the data are moved from buffer A to buffer B. After the processing of the APV frame the calculated baseline of the APV frame is written into buffer B.

The next stage reads this baseline from buffer B and subtracts it from the data. The output stage takes the common mode noise corrected data and applies a threshold cut to the amplitude of the third frame. The threshold values can be set individually for each channel (see section 5.8). They are stored in the threshold memory. This allows the handling of noisy or active strips. The sparsified data are then formatted into 32 bit wide words and are divided into APV data blocks (see section 5.5). The APV headers and the second word of the ADC header are also generated by the formatter, but they are only partly filled. The block sizes and the first word of the ADC header are later added by the control and I/O FPGA (see subsection 5.6.2). The six 32 bit wide data streams go through multiplexers, internal FIFOs and external CYPRESS FIFOs to the control and I/O FPGA. For data flow control and to tag the data blocks two additional bits are used, so that the external CYPRESS FIFOs are not 16 bit but 18 bit wide.

5.6.2 The control and I/O chip FPGA

For the control and I/O FPGA a Xilinx Virtex E XCV100E [Xil01c] is used. It multiplexes the data of the two data processor FPGAs and sends them through the optical fiber to the GeSiCA readout module using the HOTLink protocol. The chip also decodes the trigger, reset and I^2C signals from the optical receiver. These

signals are distributed to the APVs and to the data processor FPGAs. Figure 5.22 shows a schematic view of the control and I/O FPGA.

The FPGA reads the six CYPRESS FIFOs, that buffer the data, generated by the two data processor FPGAs. It brings these data into the final format (see section 5.5), so that the GeSiCA module has to add only the S-Link header. The data from the FIFOs are already divided into blocks for the different APVs. The blocks are tagged by the data processor FPGAs using two extra bits. The data also contain the APV headers and the second word of the ADC header, but the data block sizes in these headers are still empty.

The control and I/O chip reads two CYPRESS FIFOs concurrently. The three FIFOs of each data processor are read out sequentially. Thus the input data bandwidth is 640 Mbit/sec (2 x 16 bit at 20 MHz), which is twice the HOTLink output bandwidth. First a controller unit merges two 16 bit data words from the FIFOs to one 32 bit data word. To estimate the block sizes, the incoming data words are counted. The data are buffered in a single ported RAM, which is 32 bits wide and 896 words deep, so that it can store the data of a complete event. After every data block the controller completes the headers of this block, so that at the end the memory contains the finished data of the event. As the final formatting step the total event size of the ADC card is calculated and written into the memory. This value will be sent out first to allow the event size calculation for the S-Link header in the GeSiCA module. When the preparation of the data is finished, another controller unit copies them into a FIFO buffer, which decouples the data formatting from the transmission via the optical HOTLink.

Because the HOTLink connection is the bottleneck of the data flow, it constrains the performance of the whole system and has to be used as efficiently as possible. The internal FIFO buffer after the RAM avoids wasting of the HOTLink output bandwidth. Since the memory buffers, that are used for the block size calculations, are single ported, they can either be written or read. This means that the whole process of reading the CYPRESS FIFOs, formatting the data and reading the memories, so that the next event can be processed, must not take longer than the HOTLink transmitter needs to write out the data.

A rough calculation of the maximum event size, that can be transferred efficiently, which means without introducing any wait states for the HOTLink output, can be made by assuming, that the number of data words, coming from both data processing FPGAs, is equal. If the total number of data words is n, then the readout of the CYPRESS FIFOs takes $2n$ clock cycles. The HOTLink needs $4n$ clock cycles to write out the data. Copying the formatted data from the memories to the empty FIFO buffer with a depth of m words takes $5/4m + 4(n - 5/4m)$ clock cycles. The first summand $5/4m$ represents the first phase, where the controller can fill the FIFO at full speed (32 bits at 40 MHz, which gives 1.28 Gbit/sec), while the HOTLink reads the FIFO at a quarter of that data rate. After the FIFO got full, the writing speed of the remaining data words matches the HOTLink output speed. This is were the second summand comes from. It is desirable, that the time, needed to write out

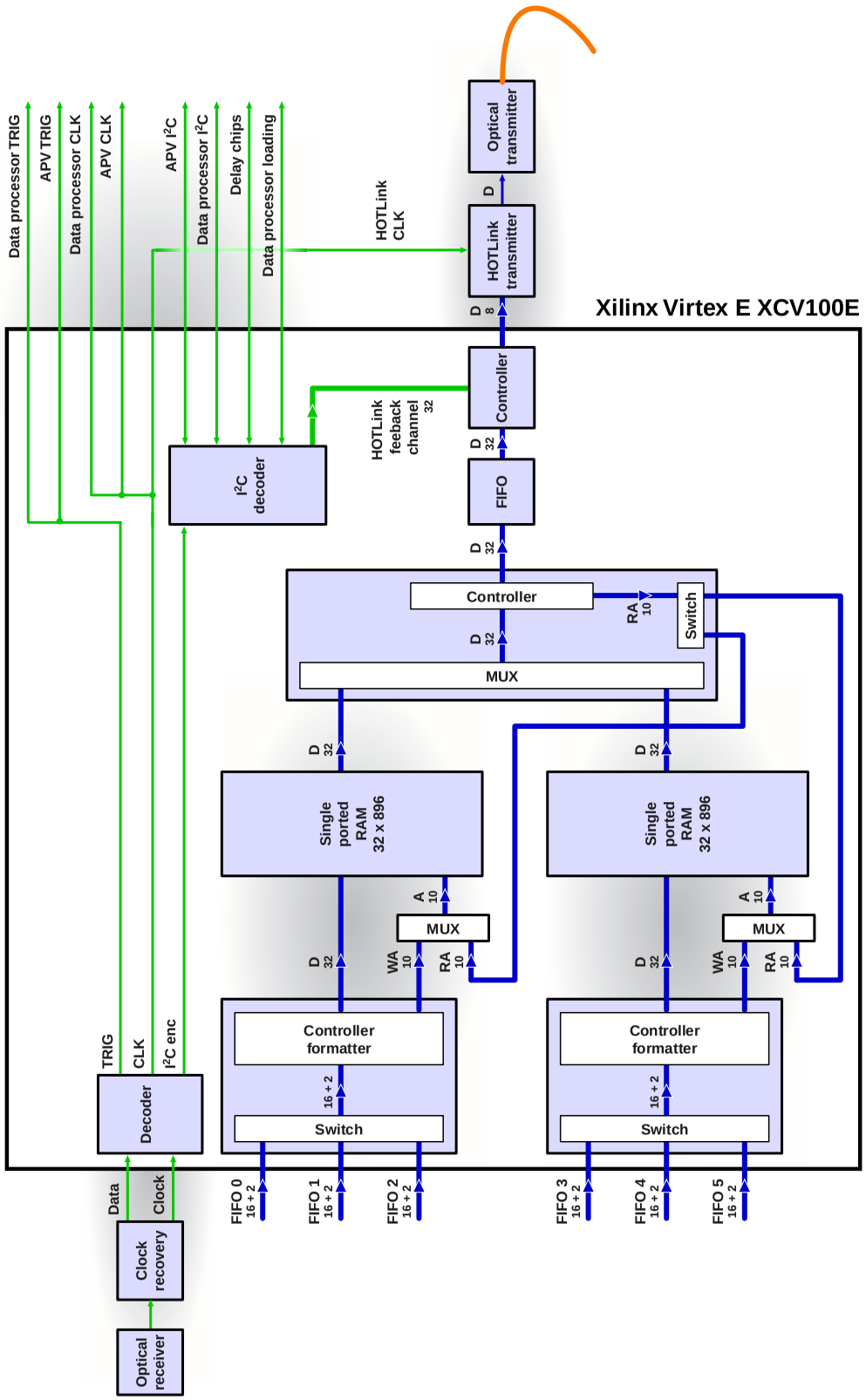


Figure 5.22: Schematic view of the control and I/O FPGA. The abbreviations for the busses have the following meaning: 'D' - data, 'WA' - write address, 'RA' - read address, 'A' - address, 'CLK' - clock signal, 'TRIG' - signal that carries trigger and reset, 'I^{2}C enc' - encoded I^{2}C from the optical fiber

the data, is bigger than the one needed to prepare the data:

$$4n > 2n + \frac{5}{4}m + 4(n - \frac{5}{4}m)$$

Due to the limited resources the FIFO buffer is $m = 512$ words deep. So that events of up to 960 words can be handled efficiently without any waste in the HOTLink bandwidth. In reality this value is of course lowered by the wait states and latencies in the controller units and by asymmetrical event sizes of the two data processor FPGAs, because in this case for part of the data the input bandwidth drops down to the HOTLink bandwidth. As the formula shows, the main handle for optimization and improvement is the input speed, which in principle could be doubled. But nevertheless if the events have an occupancy of lower than about 50 %, the full HOTLink output bandwidth is always used, so that the possible maximum trigger rate is not degraded. Latch-all data cannot be handled efficiently, but this mode is only used for calibration purposes, where the required trigger rates are very low.

At the output the 32 bit data words are split into four bytes and written to the HOTLink transmitter. This unit serializes and encodes the data. The encoded signal goes to the optical transmitter, that puts it onto the optical fiber.

The second main task of the control and I/O FPGA is the decoding of the incoming optical data stream and the distribution of the clock signal. The clock recovery chip on the ADC card extracts an 80 MHz clock out of the signal from the optical receiver. The decoder unit divides this clock by two. The resulting 40 MHz clock is used by the control and I/O FPGA itself and is distributed to the HOTLink transmitter, the APVs and via the delay chips to the data processor FPGAs.

From the serial data signal, provided by the clock recovery chip, the decoder extracts the `TRIG` signal. This signal transmits the trigger and the reset signal using sequences of three consecutive bits (see `TRIG` signal in section 5.1). The signal is distributed to the APVs and via the delay chips to the data processor FPGAs.

As the third signal the decoder separates the encoded serial I^2C transfers. The I^2C decoder transforms this signal back into the standard I^2C protocol. The I^2C bus goes to the data processor FPGAs and to the APVs. It is also used to set the delay chips and to load the programs of the data processor FPGAs. For read-back operations, which are needed for status monitoring, the I^2C decoder provides a feedback channel, that goes over the HOTLink data path to the I^2C interface in the GeSiCA module. This means, that I^2C operations block the HOTLink for other data transfers and thus should not be performed during data taking.

5.7 The GeSiCA readout module

The GeSiCA readout module has four main functions:

- Sequential multiplexing of the four incoming HOTLink data streams
- Merging of the TCS event headers with the belonging data
- Putting the event data blocks into the S-Link data format
- Transmission of the configuration data from/to the ADC cards and the APVs

Figure 5.23 shows a schematic view of the GeSiCA module.

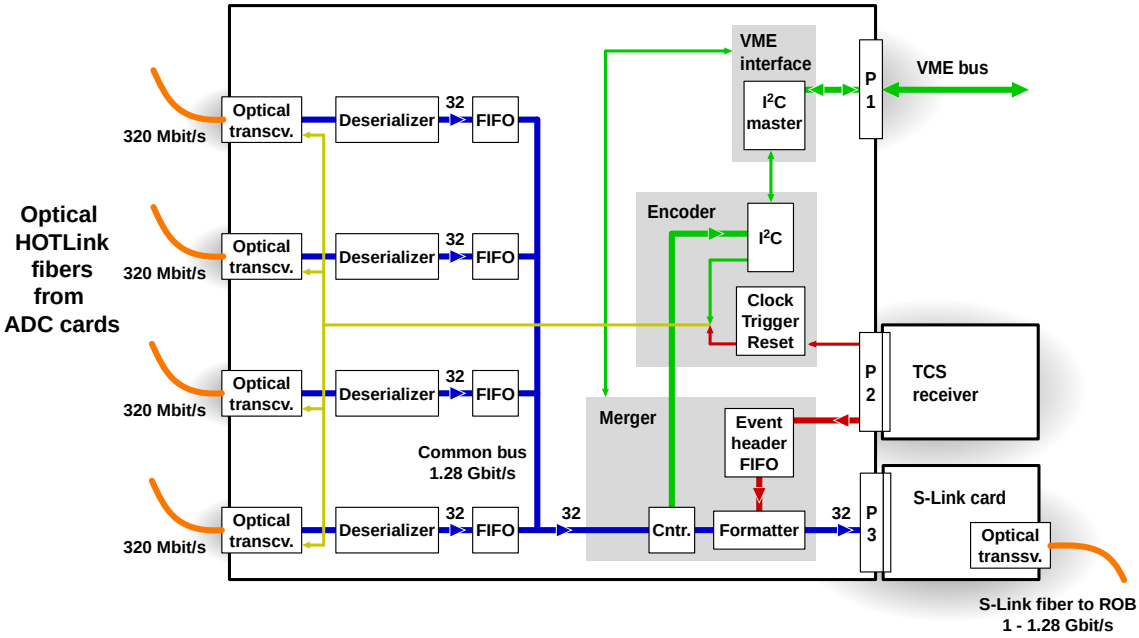


Figure 5.23: Schematic view of the GeSiCA readout module

The optical HOTLink receivers deliver the data from the ADC cards byte-wise. A de-serializer joins four bytes to 32 bit words. This unit also recognizes and tags the start and the end of the data blocks. The 32 bit data words are then buffered in a 4K word deep and 32 bit wide FIFO, that is built out of two parallel CYPRESS CY7C4245V chips. The FIFOs of the four HOTLinks are connected to the merger via a common bus.

For every event data block the merger sequentially multiplexes the data from the CYPRESS FIFOs. It reads the event header from the TCS receiver and buffer it in a separate internal FIFO. Using these TCS event information, the formatter unit

creates the S-Link header and writes both, header and data, to the S-Link card, which sends them to the Readout Buffer.

In addition to handling the ADC data, the GeSiCA readout module also provides a VME interface for configuration. The APV and the ADC card use the I^2C protocol for accessing the configuration values. Therefore GeSiCA's VME interface has a built in I^2C master, that initiates all I^2C transfers (see section 5.8). To avoid having several cables connecting an ADC card and a GeSiCA board, the encoder unit transforms the I^2C protocol, so that it can be sent over the optical fiber. The encoder also sends the trigger and the reset signals of the TCS receiver over the fiber to the ADC cards, where they are decoded and forwarded to the APVs.

The whole readout chain has to work synchronously to the TCS reference clock. This is achieved by clocking the optical transceivers with the TCS clock from the TCS receiver. The clock recovery chip on the ADC card retrieves the TCS clock from the optical data stream. The clock signal is used by the ADC cards and is also distributed to the APVs.

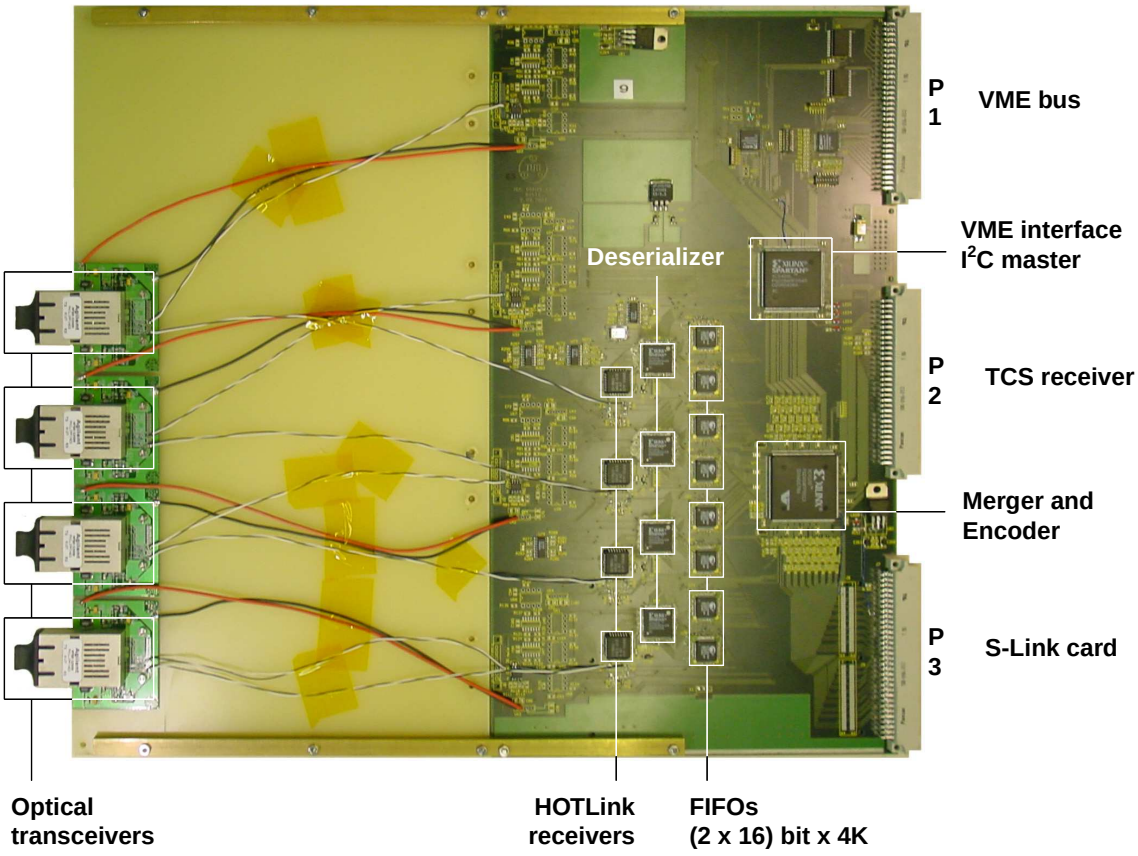


Figure 5.24: The GeSiCA board

Figure 5.24 shows a photograph of the GeSiCA board. The card itself is a half-size 9U VME module. Because the form-factor turned out to be rather unhandy, an expansion board, which holds the optical transceivers and the front panel, was added.

5.8 Configuring the readout system

The configuration of the readout system is based on the I^2C bus, which is a development of Philips Semiconductors. It is a slow, serial, bidirectional 2-wire bus for inter-IC communication. One bus line, which is called SDA, carries the data, the other one, that is called SCL, the clock signal. Each device, connected to the bus, is accessible by its unique address in a 7 bit address space. There is a simple master/slave relationship between the devices. Transfers are always initiated by a master. The I^2C bus allows multiple masters. Data scrambling due to simultaneous accesses of two or more masters is prevented by arbitration mechanisms and collision detection. The data are transferred byte-wise.

Each data bit, sent over the SDA line, is qualified by the clock signal on the SCL line, which is always generated by the master. The data on the SDA line must be stable during the HIGH period of the clock. The data signal is only allowed to change, when the clock signal is LOW. I^2C transfers are always framed by two special signals. The START signal initiates an I^2C transfer, the STOP signal finishes it. The START condition is defined by a HIGH to LOW transition of the SDA line, while the SCL line is HIGH. Similarly a LOW to HIGH transition of the SDA line, while SCL is HIGH, gives the STOP condition (see figure 5.25). START and STOP are generated by the master and they are the only situations, when the SDA line is allowed to change, when SCL is HIGH. The master can chain multiple I^2C transfers by sending a START instead of a STOP signal.

An I^2C transfer can transmit an arbitrary number of bytes. Each byte, sent over the bus, has to be acknowledged by the receiver. Therefore the master generates an extra clock pulse after every transferred byte. During this clock cycle the transmitter releases the SDA line, so that the receiver can pull it down to acknowledge the receipt of the data byte. The bytes are sent with the **m**ost **s**ignificant **b**it (MSB) first.

Figure 5.25 shows a complete I^2C transfer, transmitting two bytes of data. The first byte of an I^2C transfer is always generated by the master and contains the 7 bit I^2C address and as the **l**east **s**ignificant **b**it (LSB) the data direction bit ($R/\bar{W}$). If $R/\bar{W}$ is set to '0', a master-transmitter wants to send data to the slave-receiver, specified by the I^2C address (WRITE). A '1' indicates, that a master-receiver wants to read data from a slave-transmitter (READ). In the latter case the transfer direction changes after the first acknowledge of the slave-receiver. This means, the master-transmitter, which sent the first byte, turns into a master-receiver and waits for the data from the slave, that becomes a transmitter. More information about the I^2C specification can be found in [Phi00] and [Phi01].

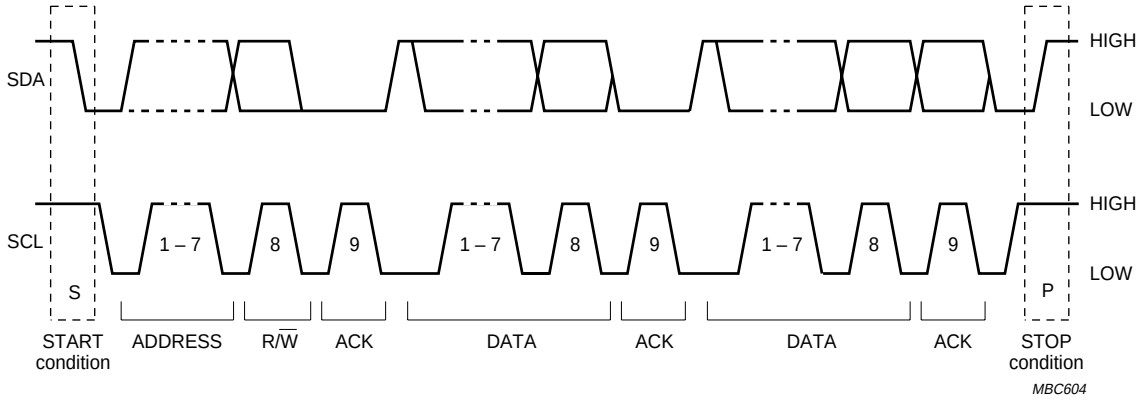


Figure 5.25: A complete I^2C data transfer [Phi00]

The I^2C specification is based on wires as transmission medium. But between the GeSiCA module and the ADC card there is only a bidirectional optical fiber connection. This means, that the I^2C data have to be encoded to go together with the TCS clock, the reset and the trigger signal from the GeSiCA module to the ADC card. For the way back the optical HOTLink path is used. Because there is only one I^2C master in the readout system, arbitration mechanisms are not needed.

The main problem with sending I^2C transfers over an optical fiber is, that the acknowledge procedure cannot work in the standard way. Normally the transmitter SDA line goes in a high impedance state after each transferred byte, so that the receiver can drive the acknowledge pulse. This kind of operation is not possible with an optical fiber. Instead of an acknowledge bit, a special byte is sent back for acknowledge. Because the I^2C bus is slow compared to the fiber connection, the overhead created by this procedure is negligible. In the GeSiCA module and in the control and I/O FPGA of the ADC card special encoder/decoder units transform the I^2C transfers, so that the optical fiber connection is completely transparent for the devices, connected to the I^2C bus.

To allow an efficient and convenient use of the I^2C address space, the ADC cards have an additional ID. The I^2C START condition is transformed into the transmission of a whole word, that contains the ID of the ADC card. This activates the I^2C bus on the specified ADC card. The bus stays active, until a STOP signal was received. This means, that the I^2C devices, connected to different ADC cards, can have the same addresses, because they are distinguished by the ID of the ADC card. For example the two data processor FPGAs have always the I^2C addresses $10xxxxx$ and $11xxxxx$, where the 5 bit part $xxxxx$ is used for internal addressing of the configuration registers (see subsection 5.8.2).

5.8.1 Setting and reading the APV 25

The I^2C interface of the APV 25 S0 uses the standard 7 bit addressing. The two most significant address bits are always "01", the remaining 5 bits are defined by the bonding of the address pads of the particular APV. The address "11111" is reserved for broadcast. This means, that up to 31 APVs can share the same I^2C bus, while having unique addresses. To access the internal registers of the APV, a so called 'command register' has to be set. The upper seven bits of this register specify the internal address and the least significant bit the direction of the transfer (LSB = '0' for writing, LSB = '1' for reading).

A write access to an APV register is an I^2C transfer with three data bytes: The first byte carries the I^2C address and the R/$\bar{W}$ bit, that is set to '0'. The second byte sets the command register, which addresses the APV register. The LSB of the command register is set to '0' to switch to write access. The third and last byte transfers the register value.

A read access consists of two I^2C transfers: first a write access to the command register sets the internal address pointer and afterwards a read access reads the addressed value. Both operations transfer two data bytes: The first byte of the first I^2C transfer contains the I^2C address of the APV and the R/$\bar{W}$ bit, which is set to '0'. The second byte of this transfer sets the command register to address the internal APV register. The LSB of the command register is set to '1' to activate the read access. The second I^2C transfer is a READ transfer. Therefore the R/$\bar{W}$ bit is set to '1'. The second byte is sent by the APV, which is now an I^2C slave transmitter and contains the addressed register value.

Since the I^2C SDA line is of open drain type, the data, read with an broadcast transfer, are the logical AND of all APV outputs. This feature is very useful for testing the error registers of a group of APVs simultaneously.

Table 5.5 shows the address mapping of the internal registers. A more detailed description of the APV registers can be found in [Jon00].

5.8.2 Configuring the ADC card and the GeSiCA module

Each data processor chip possesses an I^2C address space of 5 bits. For the FPGA on the top side of the ADC card the two most significant bits of the 7 bit I^2C address are always '10', for the one on the bottom side always '11'. Because the data processor FPGAs have a big number of configuration values, an addressing scheme similar to the one of the APV is used. This means, that in addition to the real I^2C address also the next byte serves as an address, so that the total address space per data processor is 13 bits. In this 13 bit address the five bits from the real I^2C address give the lower part, the eight bits from the following I^2C transfer the upper part. Table 5.6 shows the mapping of the addresses.

Reg. name	**Address** [6:0] = **Command reg.** [7:1]	**Function**
IPRE	0010000	Pre-amplifier input FET current bias
IPCASC	0010001	Pre-amplifier cascode current bias
IPSF	0010010	Pre-amplifier source follower current bias
ISHA	0010011	Shaper input FET current bias
ISSF	0010100	Shaper source follower current bias
IPSP	0010101	APSP current bias
IMUXIN	0010110	Multiplexer input current bias
ISPARE	0010111	Not used
ICAL	0011000	Calibrate edge generator current bias
VFP	0011001	Pre-amplifier feedback voltage bias
VFS	0011010	Shaper feedback voltage bias
VPSP	0011011	APSP voltage level adjust
CDRV	0011100	Calibrate output mask
CSEL	0011101	Calibrate delay select
MODE	0000001	Mode of operation of the chip
LATENCY	0000010	Delay between write and trigger pointers
MUXGAIN	0000011	Sets gain of multiplexer
ERROR	0000000	Holds error flags (read only)

Table 5.5: Address mapping of the internal APV registers

The procedure for the read and write operations is basically the same like for the APV registers with the only difference, that the registers in the data processor are wider than eight bits, so that the configuration values are transferred using two bytes. This means, the writing of a configuration value consists of four I^2C transfers. To read back register values first two bytes have to be written to set the address pointer and then the two bytes of the configuration value can be read.

The GeSiCA readout module has only two configuration values, that can be easily accessed via one VME register. The first value is the source ID, that has to be unique within the COMPASS DAQ system. It identifies the GeSiCA and defines the basic decoding routines, that have to be performed on the data. The second value specifies, which of the four optical HOTLinks of the GeSiCA board are connected to an ADC card.

5.9 Summary and outlook

In the year 2001 run of the COMPASS experiment 14 GEMs and 1 silicon detectors were read out with the system described in this chapter. The data of these altogether 24000 channels were digitized and processed by 16 ADC cards, which were connected to 5 GeSiCA boards.

The main goal of the run was the commissioning of the detectors and the readout, so that the present version of the readout chain is still missing some features to give

Address (13)	**Configuration value**
0x0000	10 bit pedestal for channel 0 of APV 0
⋮	
0x007F	10 bit pedestal for channel 127 of APV 0
0x0080	10 bit pedestal for channel 0 of APV 1
⋮	
0x00FF	10 bit pedestal for channel 127 of APV 1
0x0100	10 bit pedestal for channel 0 of APV 2
⋮	
0x017F	10 bit pedestal for channel 127 of APV 2
0x0180	10 bit pedestal for channel 0 of APV 3
⋮	
0x01FF	10 bit pedestal for channel 127 of APV 3
0x0200	10 bit pedestal for channel 0 of APV 4
⋮	
0x027F	10 bit pedestal for channel 127 of APV 4
0x0280	10 bit pedestal for channel 0 of APV 5
⋮	
0x02FF	10 bit pedestal for channel 127 of APV 5
0x0300	10 bit threshold for channel 0 of APV 0
⋮	
0x037F	10 bit threshold for channel 127 of APV 0
0x0380	10 bit threshold for channel 0 of APV 1
⋮	
0x03FF	10 bit threshold for channel 127 of APV 1
0x0400	10 bit threshold for channel 0 of APV 2
⋮	
0x047F	10 bit threshold for channel 127 of APV 2
0x0480	10 bit threshold for channel 0 of APV 3
⋮	
0x04FF	10 bit threshold for channel 127 of APV 3
0x0500	10 bit threshold for channel 0 of APV 4
⋮	
0x057F	10 bit threshold for channel 127 of APV 4
0x0580	10 bit threshold for channel 0 of APV 5
⋮	
0x05FF	10 bit threshold for channel 127 of APV 5
0x1000	Threshold that distinguishes between analog '0' and '1' in the digital part of the APV frame
0x1001	Maximum number of clock cycles after the trigger without APV header; after this time readout is forced
0x1002	Sets the APV that serves as synchronization reference
0x1003	Sets the output data format

Table 5.6: Address mapping of the internal APV registers

the full performance. In the latch-all readout mode the system reached trigger rates of up to 1 kHz. With an event size of 6259 32 bit words (48 APVs, each generating 128 data words and 2 header words, plus 8 data processor FPGAs, each giving 2 header words, plus the 3 S-Link header words) per GeSiCA this gives a data rate of about 25 Mbyte/s for each GeSiCA board. The main bottleneck here was the merger unit of the GeSiCA module. During the reading of the external CYPRESS FIFOs and the writing of the internal output FIFO the merger unit performs wait states, so that only a fraction of the full bandwidth is used.

With the the sparsified readout mode trigger rates of up to 20 kHz were reached. The performance was governed by two things: first of course by the occupancy of the detectors and thus by the performance of the zero suppression algorithm and second by the fact, that there was no buffering of the triggers in the data processor FPGAs, so that the triggers were not allowed to come closer than 21 μs, which is the time the APVs need to write out the complete event. Figure 5.26 shows the occupancy of some GEM detectors in the high intensity muon beam using a 3 σ threshold cut. The average number of hit strips lies between 14 and 32 which corresponds to occupancies of 1.9 % and 4.2 %. In figure 5.27 the same is shown for the silicon detector. Here the average occupancy lies at 11 hits (p-side) and at 12 hits (n-side) respectively, which is both around 1 %.

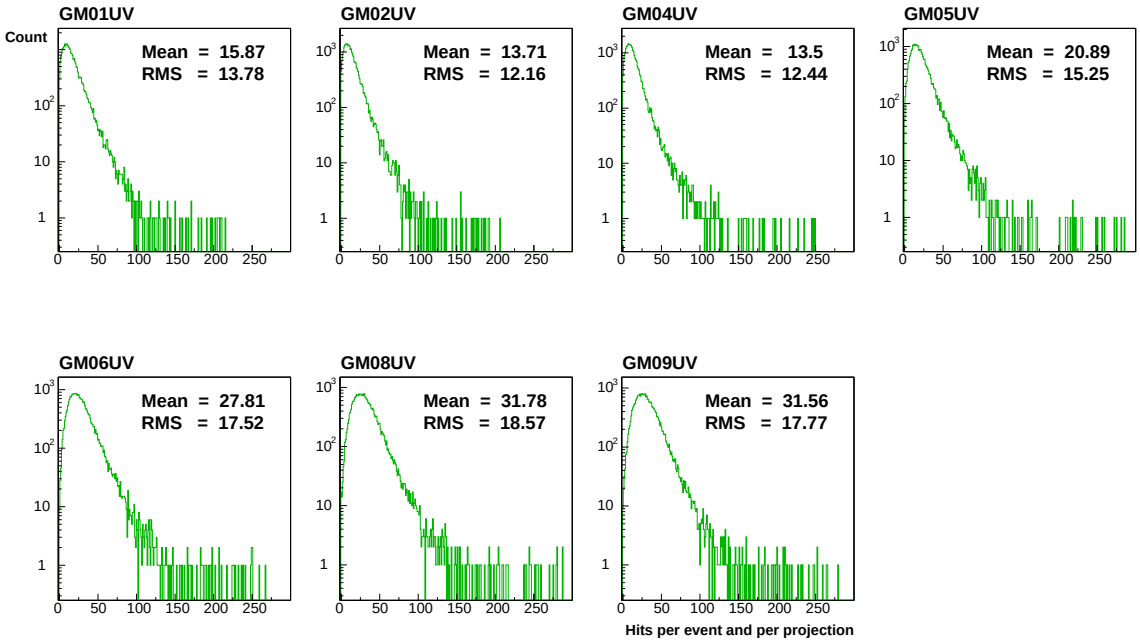


Figure 5.26: The occupancy of seven GEM detectors in the high intensity muon beam. The higher the module number, the more downstream the particular detector sits. The widening of the beam is nicely visible: The average occupancy rises from about 15 hits or 2.0 % for GM01, that sits right after the first magnet, to a value of 32 hits or 4.2 % for the last two detectors, which are located approximately 20 m downstream from the target behind the second magnet. The occupancy rises, because the central region of the GEM detectors is deactivated, so that only the part of the beam, that is far away enough from the beam axis, is detected. [Ket01a], [CBT01f]

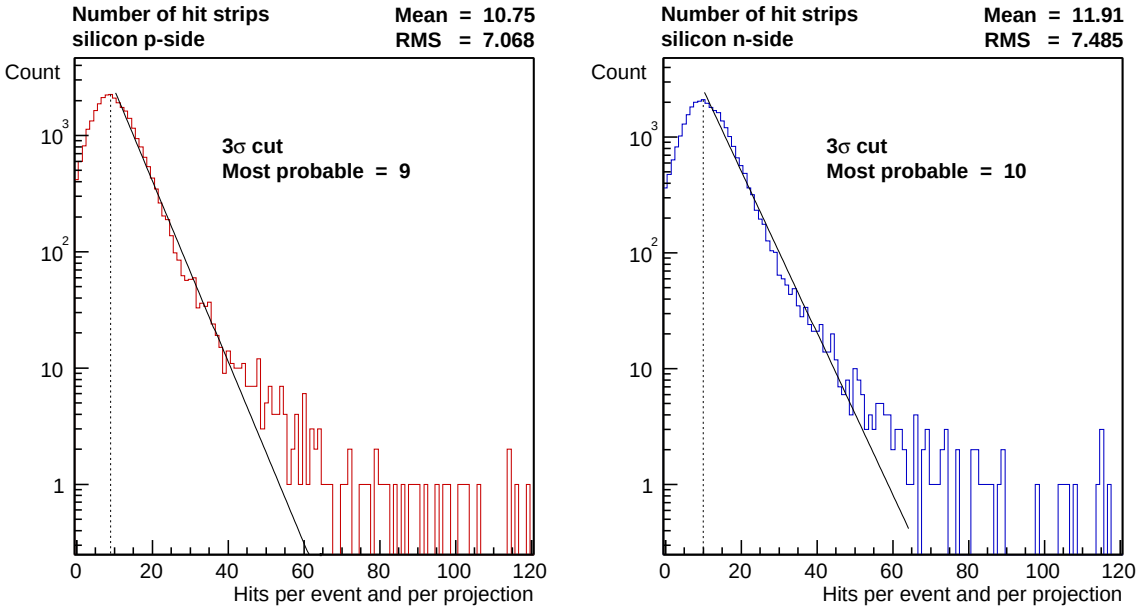


Figure 5.27: The occupancy of the two sides of the silicon detector [Wag01a], [CBT01d]

From the noise point of view the system behaved well. The ADC card itself introduces a noise of one ADC channel, which is equivalent to 300 electrons. This noise comes mainly from the quantization noise of the ADCs. For the GEM detectors the noise per strip is around 1650 electrons or 5.5 ADC channels for raw data. With common mode noise correction this value is lowered by 1 ADC channel to 1350 electrons. For the silicon detector the strip noise is about 6 ADC channels or 1800 electrons for raw data and 5 ADC channels and 1500 electrons with common mode noise correction applied (see figure 5.28).

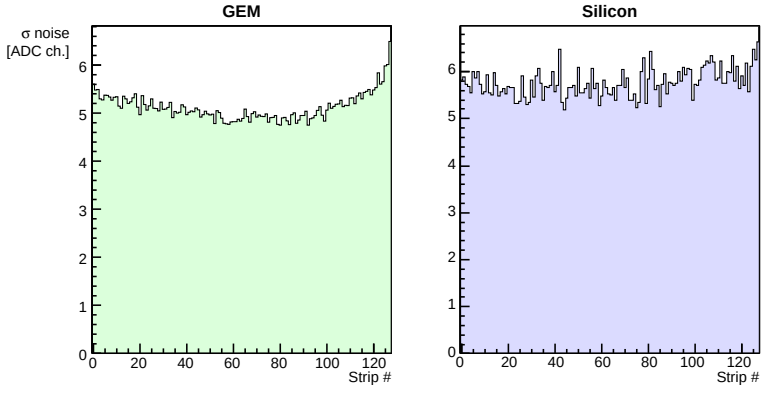


Figure 5.28: The strip noise of the GEM and the silicon detector without common mode noise correction [Ket01a], [CBT01b], [Wag01a], [CBT01e]

The distribution of the baseline values, calculated by the hardware using the histogram of the accumulated amplitude frequencies (see subsection 5.6.1), is shown in figures 5.29 and 5.30. The diagrams were made using the last word of the APV data block (see section 5.5) and they show only the result of the histogramming method. The effect of the shifting of the average amplitude is not visible. If there were no hits, the average amplitude of the frame should be very close to the baseline, so that the distribution of the frame baseline, that is calculated by the hardware, should be a Gaussian, that is centered at 32. If there are hits in the frame, the average value is drawn away from the baseline to higher amplitudes. This means, that the distribution of the hardware baseline is shifted to lower values and that the distribution becomes asymmetric, which is nicely visible in the pictures.

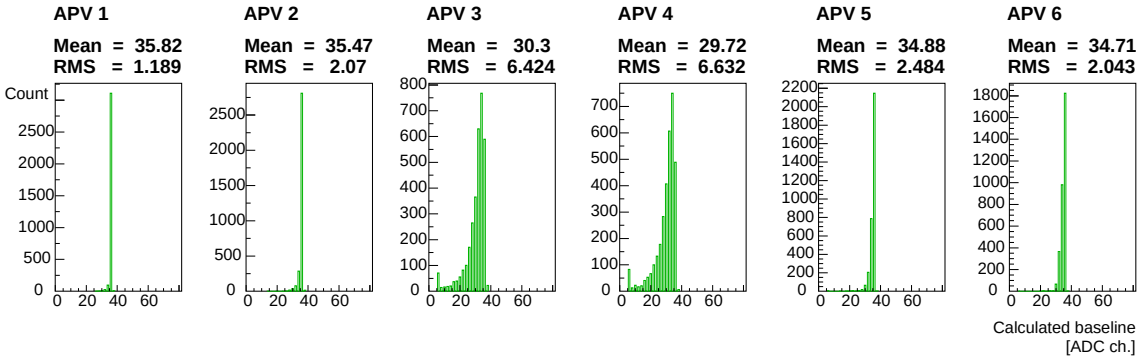


Figure 5.29: The distribution of the frame baselines of the six APVs of one side of a GEM detector calculated by the hardware. The influence of the beam in the middle of the chamber is clearly visible. Due to the hits the average amplitude is drawn away from the real baseline of the frame, so that the distribution becomes asymmetric and is shifted to lower values [Ket01a], [CBT01c]

This first big scale test of the GEM and silicon readout chain proved, that the system architecture works and that it will be possible to reach the specification goal of 100 kHz trigger rate. To speed up the system from the presently 20 kHz some modifications and enhancements have to be implemented.

First the data processor FPGAs should be enabled to accept triggers, that come closer than 21 μs. The implementation of a trigger buffer is relatively easy and that it was not already implemented in the year 2001 run, was just due to the lack of time. Second priority has the enlargement of the data bandwidth on the GeSiCA board, which means, that the read and write controller for the FIFOs have to be optimized, so that they do not need the wait states anymore. Beside this also some optimization of the data processor FPGA has to be done, because the present design already occupies nearly 90 % of the hardware resources of the chip. The starting point will be the redesign of the buffer A in the data pipeline (see subsection 5.6.1).

Further on a better treatment of buffer overflows will be implemented in the data processor FPGAs and in the GeSiCA module. To allow the measurement of pedestals with latch-all data during the off-spill time, it will be made possible to switch the readout mode 'on the fly'. In the farer future it also planned to increase the readout

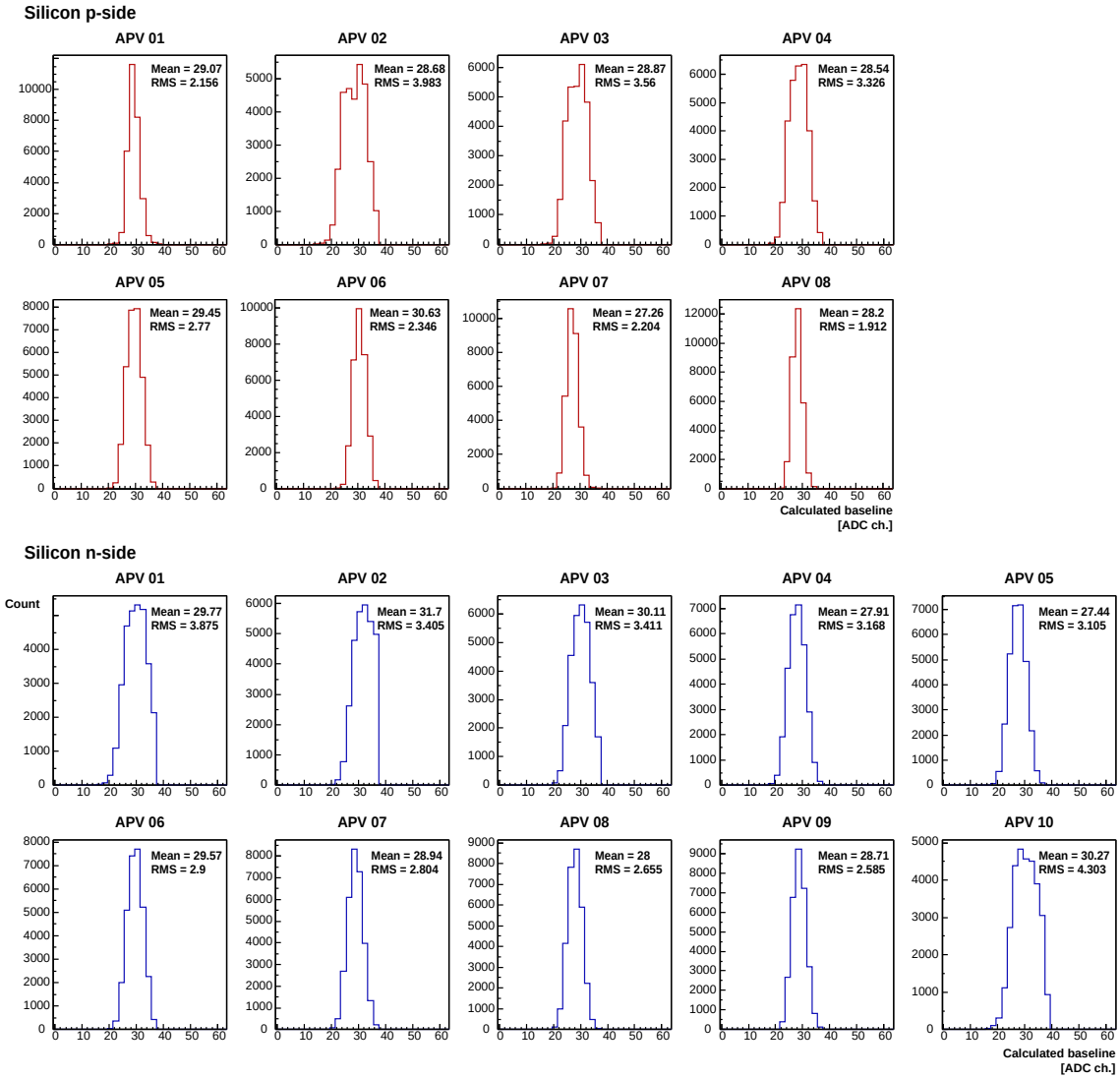


Figure 5.30: The distribution of the frame baseline of the silicon detector calculated by the hardware. The upper part shows the p-side with eight chips (red) the lower part the n-side with ten APVs (blue) [Wag01a], [CBT01d]

speed of the APV from 20 MHz to 40 MHz. This will be needed to stand the 100 kHz trigger rate. In the run of the year 2002 the whole system will be enlarged to 20 GEM and 4 silicon detectors, which give in total nearly 40000 channels.

Chapter 6

Conclusions

The two systems presented in this thesis, the Trigger Control System and the common GEM and silicon readout chain, together provide a complete system to read out GEM and/or silicon detectors at high trigger rates. The TCS delivers the trigger, reset and clock signals for the detector frontend. The GeSiCA module combines the data from the ADC cards and labels them with the TCS event header. Because all digital data are transferred via optical fibers, the system is flexible and easy to handle concerning grounding and noise.

The year 2001 run of the COMPASS experiment proved both systems to be working and performing well. The TCS fulfilled all specifications, whereas the GEM and silicon readout still has to be improved to reach 100 kHz trigger rate.

Appendix A

The software environment for the FPGA development

For the FPGA design of both projects, the TCS and the GEM silicon readout, the Xilinx Foundation software on Windows NT was used. Because the development work on the TCS was started earlier, it is still a combined schematic/VHDL design with VHDL at the lower levels and schematics used to build higher level blocks. This approach of course has the drawback, that it is quite time-consuming to implement changes, because of all this 'painting' of wires and busses. In addition the VHDL optimization acts only locally on the particular modules. To make the work more efficient it was decided to start the GEM silicon readout as a pure VHDL project. Unfortunately the Foundation software provides more a patch- than a framework for bigger VHDL projects, so we were forced to look for alternative solutions. One of the most annoying problems was the rather simple editor, but with XEmacs for Windows [XEm01] and the VHDL Mode package by Reto Zimmerman [VHD01] a really great replacement was found. The absence of a version control system was overcome by interfacing the CVS on a UNIX machine unsing WinCvs [Win01] in combination with PuTTY [PuT01] as a SSH client. The only thing still badly missing is a native VHDL simulator.

It seems that the Synopsys FPGA Express V3.5 compiler has problems with nested 'for ... loop' statements and hierarchical data structures, built out of nested arrays. The memory consumption during compilation and implementation is very high and some designs were not even linkable because the compiler produced a memory overflow. 512 Mbyte RAM plus the same amount of virtual memory turned out to be the minimum configuration to run the compilation. In addition the times for compilation and implementation are quite long. On a 1 GHz machine the whole process lasts about 1.5 h. It turned out, that the compilation is quite sensitive on the options, set in the synthesis options and the synthesis constraints, resulting sometimes in not working programs.

Bibliography

[Abt99] I. Abt et al. Double sided microstrip detectors for the high radiation environment in the HERA-B experiment. *MPI-PhE* **99-05** (1999).

[AL95] ALICE Collaboration. Technical proposal for A Large Ion Collider Experiment at the CERN LHC. *CERN/LHCC/95-71* (1995).

[Bac99] S. Bachmann, A. Bressan, D. Mörmann, L. Ropelewski, F. Sauli, and A. Sharma. Charge amplification and transfer processes in the gas electron multiplier. *Nucl. Inst. Methods* **A438**, 376 (1999).

[Bac00] S. Bachmann, A. Bressan, M. Capéans, M. Deutel, S. Kappler, B. Ketzer, A. Polouenkov, L. Ropelewski, F. Sauli, E. Schulte, L. Shekhtman, and A. Sokolov. Discharge mechanisms and their prevention in the gas electron multiplier (GEM). *CERN-EP/2000-151* (2000).

[Ber97] SINTEF Electronics & Cybernetics, R. W. Bernstein, and T. Westgaard. Technical specifications *and* test specifications for MPI HERA-B doublesided silicon. (1997).

[Bij01] E. van der Bij. CERN S-LINK home page, `http://hsi.web.cern.ch/HSI/s-link/`.

[CBT01a] COMPASS beam time. Run 11453, spill 1, event 8, GM04UV, proj. 3, APV 3, sample 2. (2001).

[CBT01b] COMPASS beam time. Run 11973, GM01UV, proj. 3, APV 1, sample 2. (2001).

[CBT01c] COMPASS beam time. Run 12451, GM08UV, proj. 3, sample 2. (2001).

[CBT01d] COMPASS beam time. Run 12601. (2001).

[CBT01e] COMPASS beam time. Run 12737, proj. 2, APV 4, sample 1. (2001).

[CBT01f] COMPASS beam time. Run 13198, GM01UV – GM09UV, proj. 3. (2001).

[Co96] The COMPASS collaboration. A proposal for a Common Muon and Proton Apparatus for Structure and Spectroscopy. *CERN* **SPSLC 96-14** (1996).

[Cyp01] CYPRESS. HOTLink design considerations, `http://www.cypress.com/hotlink/index.html`.

[DAT] CERN ALICE DAQ Group. ALICE DATE user's guide. *ALICE-INT-2000-31 v.2* .

[Fis00] H. Fischer, J. Franz, A. Grünemeier, F. H. Heinsius, L. Henning, K. Königsmann, I. Konorov, M. Niebuhr, T. Schmidt, H. Schmitt, and L. Schmitt. The COMPASS online data format. *COMPASS Note 2000-8* (2000).

[Fri01] J. Friedrich, `Jan.Friedrich@Physik.TU-Muenchen.de`. Private communication, 2001.

[Iee01] IEEE. Home page, `http://www.ieee.org`.

[Jed01] JEDEC. Home page, `http://www.jedec.org`.

[Jon99] L. L. Jones, M. J. French, Q. Morrissey, A. Neviani, M. Raymond, G. Hall, P. Moreira, and G. Cervelli. The APV25 deep submicron readout chip for CMS detectors. *CERN* **99-09** (1999).

[Jon00] L. L. Jones. APV25-S0 user guide version 2.1, `http://www.te.rl.ac.uk/med/`.

[Ket01a] B. Ketzer, `Bernhard.Ketzer@cern.ch`. Private communication, 2001.

[Ket01b] B. Ketzer, B. Grube, I. Konorov, and F. Simon. Decoding the APV. *COMPASS Note 2001-10* (2001).

[Kon01] I. Konorov, `Igor.Konorov@Physik.TU-Muenchen.de`. Private communication, 2001.

[Kuh01] R. Kuhn. *Simulations for the Measurement of the Polarizabilities of the Pion at COMPASS*. Diploma thesis, TU München, November 2001.

[Leo87] W. R. Leo. *Techniques for Nuclear and Particle Physics Experiments*. Springer, 1987.

[LWS94] G. Lehmann, B. Wunder, and M. Selz. *Schaltungsdesign mit VHDL*. Institut für Technik der Informationsverarbeitung Universität Karlsruhe, 1994, `http://www-itiv.etec.uni-karlsruhe.de/FORSCHUNG/VEROEFFENTLICHUNGEN/lws94/download.html`.

[Mäd] A. Mäder. *VHDL Kurzbeschreibung*. Fachbereich Informatik Universität Hamburg, `http://eda.ei.tum.de/forschung/vhdl/src/VHDL_by_Maeder.ps.gz`.

[Nie00] M. Niebuhr. *Entwicklung eines 250-MHz-Zählers mit totzeitfreier Auslese für das COMPASS Experiment*. Diploma thesis, Universität Freiburg, November 2000.

[Pci01] PCI SIG. Home page, `http://www.pcisig.com/home`.

[Pei92] A. Peisert. Silicon microstrip detectors. *DELPHI* **92-143 MVX 2** (1992).

[Phi00] Philips. The I^{2}C-bus specification version 2.1, `http://www.semiconductors.philips.com/acrobat/various/I2C_BUS_SPECIFICATION_3.pdf`.

[Phi01] Philips. The I^{2}C web site, `http://www.semiconductors.philips.com/buses/i2c/support/`.

[PTVF95] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C, 2nd Edition*. Cambridge University Press, 1995.

[PuT01] Simon Tatham. PuTTY home page, `http://www.chiark.greenend.org.uk/~sgtatham/putty/`.

[Ray00] M. Raymond et al. The CMS tracker APV25 0.25 μm CMOS readout chip. In *6th Workshop on Electronic for LHC Experiments*, Crakow, Poland, September 2000.

[Sau97] F. Sauli. GEM: A new concept for electron amplification in gas detectors. *Nucl. Inst. Methods* **A386**, 531 (1997).

[Sch98] M. Schierloh. *Einsatz programmierbarer Logikbausteine in der COMPASS-Ausleseelektronik*. Diploma thesis, Universität Freiburg, December 1998.

[Sim01a] F. Simon. *Commissioning of the GEM Detectors in the COMPASS Experiment*. Diploma thesis, TU München, November 2001.

[Sim01b] F. Simon, C. Altunbas, J. Friedrich, B. Grube, S. Kappler, B. Ketzer, I. Konorov, S. Paul, A. Placci, L. Ropelewski, and F. Sauli. GEM detectors for COMPASS. In *7th International Conference on Advanced Technology and Particle Physics*, Como, Italy, October 2001.

[TTC00] RD 12 Collaboration. Status report on the RD-12 project. *CERN/LHCC 2000-002* (2000).

[TTC01] RD 12 Collaboration. Home page, `http://ttc.web.cern.ch/TTC/intro.html`.

[Tur00] R. Turchetta, G. Cervelli, M. J. French, J. Fulcher, G. Hall, L. L. Jones, P. Moreira, Q. Morrissey, A. Neviani, E. Noah, and M. Raymond. Design and results from the APV25, a deep submicron CMOS front-end chip

for the CMS tracker. In *4th International Symposium on Development and Application of Semiconductor Tracking Detectors*, Hiroshima, Japan, March 2000.

[VHD01] Reto Zimmerman. Emacs VHDL Mode, `http://opensource.ethz.ch/emacs/vhdl-mode.html`.

[Wag01a] R. Wagner, `Robert.Wagner@Physik.TU-Muenchen.de`. Private communication, 2001.

[Wag01b] R. Wagner. *Commissioning of Silicon detectors for the COMPASS experiment*. Diploma thesis, TU München, December 2001.

[Win01] WinCvs. Home page, `http://www.wincvs.org/`.

[XEm01] XEmacs. Home page, `http://www.xemacs.org`.

[Xil01a] Xilinx. Spartan and Spartan-XL Families Field Programmable Gate Arrays, `http://www.xilinx.com/partinfo/databook.htm`.

[Xil01b] Xilinx. Virtex 2.5 V Field Programmable Gate Arrays, `http://www.xilinx.com/partinfo/databook.htm`.

[Xil01c] Xilinx. Virtex-E 1.8 V Field Programmable Gate Arrays, `http://www.xilinx.com/partinfo/databook.htm`.

Acknowledgments

First I would like to thank Prof. Stephan Paul for giving me this subject and allowing me to spent some time at CERN. I really enjoyed this.

Then many thanks go to Igor Konorov, who introduced me into the FPGA business. He was a good teacher with great patience and I learned a lot from him.

I am also grateful to Lars Schmitt for the discussions, his support in the development of the TCS and the readout system and for reading and correcting this thesis.

Next I would like to thank Wolfgang Liebl, who allowed me to take over his prototype FPGA designs of the TCS.

Of course there is also Heinz Angerer. I have to thank him for his practical help and advice, especially for the jitter measurements of the TCS.

I am much obliged for the support, the GEM team – especially Bernhard Ketzer, Jan Friedrich and Frank Simon – provided during the testing and debugging phase of the readout chain in the year 2001 run. Jan also helped me to create some pictures – special thanks for that. I am as well indebted to the silicon crew – Michael Wiesmann, Robert Wagner and Rita de Masi – for their help and support.

Also Roland 'the man page' Kuhn deserves many thanks for his kind help. I bothered him with a lot of TEX and Linux questions. He also allowed me to use his TEX style file.

Am meisten möchte ich mich natürlich bei meinen Eltern bedanken. Durch sie kam ich an die TU München und ihre Unterstützung und Förderung ermöglichte mir, mich auf mein Studium zu konzentrieren.

Last but not least I want to thank Xilinx for the bugs and 'features' they implemented into their software, which were the reasons for the one or the other very exciting extra night shift. And by the way, if somebody is looking for examples how *not* to design a UI, the Foundation package is a rather complete collection.

Own contributions

Of course the two systems, presented in this thesis – the TCS and the common GEM and silicon readout – were build and developed not by one single person, but in the framework of the COMPASS collaboration. My task was mainly the developement and debugging of programs for the FPGAs.

I started to work on the TCS when I still was a technical student. The work was based on prototypes of the TCS controller and receiver made by I. Konorov and W. Liebl. I took over the FPGA designs and extended the TCS command set to the full functionality. I also added the MulitDAQ mode and the VME interface. Then I tested and debugged the system together with I. Konorov and L. Schmitt and measured the jitter of the TCS reference clock.

For the GEM and silicon readout I designed the data processing parts in the conrol and I/O FPGA and in the data processor FPGA, which both sit on the ADC card. The hardware was developed by I. and G. Konorov. In the year 2001 run I. Konorov and I tested and debugged the whole readout chain.