Dissertation
submitted to the
Combined Faculties for the Natural Sciences and for Mathematics
of the Ruperto-Carola University of Heidelberg, Germany
for the degree of
Doctor of Natural Sciences

Put forward by
Dipl.-Phys. Manfred Kirchgessner
born in Bad Soden, Germany

Oral Examination: December 5, 2017

Control, Readout and Commissioning of the Ultra-High Speed 1 Megapixel DSSC X-Ray Camera for the European XFEL

Referees: Prof. Dr. Peter Fischer
Prof. Dr. Udo Kebschull

Abstract:

The goal of this thesis was to develop the software and firmware basis to control and read out the 1-Megapixel DEPFET Sensor with Signal Compression (DSSC) detector, which is being built for the European XFEL (EuXFEL). The DSSC detector proposes single photon resolution at 0.5 keV photon energy, high dynamic range of 10000 photons at a very high frame rate of 4.5 MHz.
During this thesis, the readout chain has been implemented, which receives the average data rate of 134 GBit/s from the detector and transfers sorted image data via four QSFP+ fiber cables to the DAQ system. Additional control software and firmware has been developed for commissioning of the first 1/16th megapixel prototypes. A multifunctional measurement and data analysis framework has been created which is used for characterization of the detector, particularly of properties of the complex readout ASICs. Additional parameter trimming routines to automatically adapt gain and offset parameters of a large pixel matrix have been developed in order to generate suitable configurations which have been applied during two measurement campaigns at the Petra III synchrotron. For integration into the new Karabo framework, which is provided by XFEL for beamline control, several software devices have been implemented.

Zusammenfassung:

Ziel dieser Arbeit war es, die Software- und Firmwarebasis zur Steuerung und Auslese des 1 Megapixel DEPFET Sensor with Signal Compression (DSSC) Detektors zu entwickeln, der für den European XFEL (EuXFEL) gebaut wird. Der DSSC Detektor verspricht Einzelphotonenauflösung bei 0,5 keV Photonenenergie, einem hohen Dynamikbereich von 10000 Photonen bei einer sehr hohen Bildrate von 4,5 MHz.
Während dieser Arbeit wurde die Auslesekette implementiert, welche die Bilddaten mit einer durchschnittlichen Datenrate von 134 GBit/s vom DSSC-Detektor empfängt, bildweise sortiert und über vier QSFP+ Glasfaserkabel zum DAQ-System überträgt. Des weiteren wurde für die Inbetriebnahme der ersten 1/16-Megapixel-Prototypen Steuerungssoftware und -firmware entwickelt. Ein multifunktionales Mess- und Datenanalyse-Framework wurde erstellt, das zur Charakterisierung des Detektors, insbesondere der Eigenschaften des komplexen Auslese-ASICs, verwendet wird. Zusätzlich wurden Parameter-Trimmroutinen für die automatische Anpassung der Verstärkungs- und Offsetparameter aller Pixel einer großen Pixelmatrix entwickelt, um geeignete Konfigurationen zu erzeugen, die bei zwei Messkampagnen am Petra III-Synchrotron eingesetzt wurden. Außerdem wurden innerhalb dieser Arbeit, für die Integration in das neue Karabo-Framework, welches von XFEL für die Steuerung der Beamline-Umgebung entwickelt wurde, mehrere Software-Devices implementiert.

Contents

Part I

Introduction and Presentation of the 1 Megapixel DSSC Camera

1

Introduction

To enter new territories of fundamental research, one has to make enormous effort nowadays. With the EuXFEL, a two billion euro project has been launched in Hamburg (Germany) this year, which will enable completely new approaches in fundamental research and a variety of scientific fields. It will be the largest free electron laser which has ever been built: a light source of the 4th generation producing ultra short, hard X-ray pulses in the Å region with an unparalleled brilliance which will surpass the strongest 3rd generation synchrotron sources by over a billion times. This new quality of light generation is driven by a super-conducting, linear accelerator (LINAC) which has originally been proposed for the large international linear collider (ILC). But it turned out that the quality of the generated electron beam was such high that it perfectly fits to the requirements of a 4th generation light source. With a pulse generation rate of 4.5 MHz and 27000 X-ray flashes per second, the EuXFEL will also present enormous challenges to the imaging detectors and the data acquisition systems. Since no existing detector will fit the requirements of the EuXFEL, three large 2D megapixel detectors have been designed for usage in the experiments at the different beamlines. The required objective of single photon resolution combined with a large dynamic range of 10000 photons will be realized by each detector for the respective photon energy range.

The **D**EPFET **S**ensor with **S**ignal **C**ompression (DSSC) detector will cover the lower energy range. It is designed and developed by a consortium of several contributors including DESY (Hamburg, Germany), Politecnico di Milano (Italy), University of Bergamo (Italy), the European XFEL GmbH (Germany) and Heidelberg University (Germany). The focal plane area of the megapixel detector as shown in figure 1.1 will fill about $21 \times 21\,cm^2$. By implementing high sensitive DEPFET sensors, which have already shown very good performance in low noise applications, single photon resolution of the DSSC detector will be achieved down to photon energies of 0.5 keV. By a novel design concept of the DEPFET sensor, a signal compression is introduced, leading to a non linear signal response which increases the dynamic range of the sensor to the requested 10K photons.

The DSSC detector implements full parallel readout of the sensor to deal with the fast event rate of 4.5 MHz. Therefore, every sensor pixel is bump bonded to a dedicated readout channel on an application specific integrated circuit (ASIC). The ASIC contains a complex signal processing chain including an integrating filter and an 8 to 9 bit ADC (analog-to-

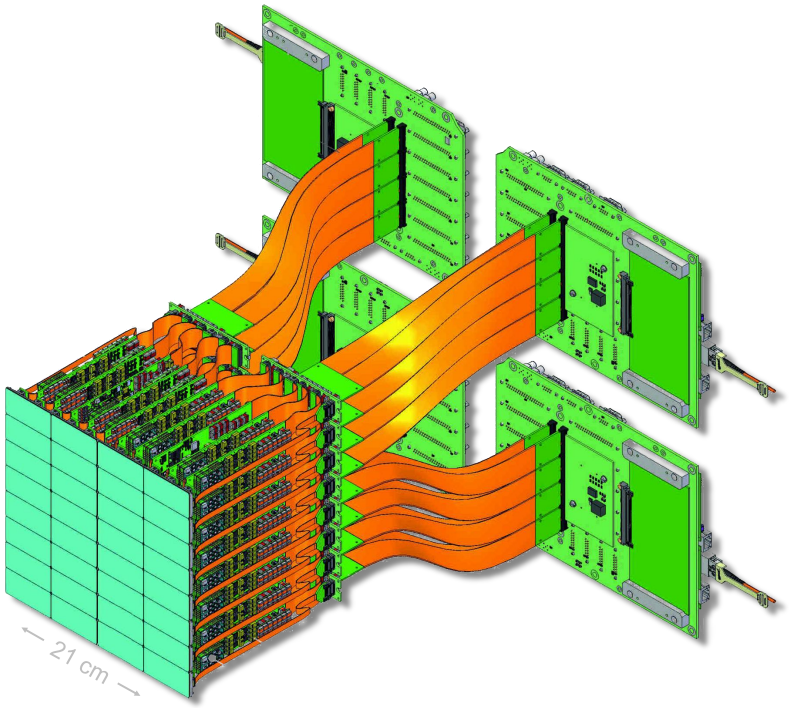


Figure 1.1.: **Artistic image of the DSSC megapixel detector.** (Courtesy of K. Hansen, DESY)

digital converter). Since the direct transfer of the pixel data is not feasible between two light flashes, the readout ASIC also contains an in-pixel SRAM memory which is used to store the digitized values for later transfer during the 99.4 ms pause that follows each X-ray pulse train. Not least because of the limited space in the readout pixels, the DSSC detector can only store 800 of the 2700 images per pulse train which remains the largest number among the other detectors. Acquiring at a repetition rate of 10 Hz, the DSSC megapixel detector generates a continuous data rate of over 134 GBit/s.

By implementing modern components, a compact design of the data acquisition (DAQ) is possible which realizes a handy transfer of the image data via only four optical QSFP+ fiber cables, each bundling four 10 Gigabit-Ethernet links. A complex readout chain transports the pixel data from the in-pixel SRAM in the ASIC to the first and second FPGA stage before the data is packed into UDP Ethernet packets and transferred out of the detector. The average utilization of the nominal 160 GBit/s data rate of the optical links thereby is round 84% which is a remarkably high value.

The second FPGA stage consists of four identical FPGA cards, implementing a Xilinx Kintex-7 FPGA (available since 2012). Each FPGA card combines the readout data of one quarter of the megapixel detector and generates standard Ethernet packets from the data stream before the data is sent out of the system. By implementing a 1 GB DDR3-1600 memory, which runs at 800 MHz, it is possible not only to buffer two complete data trains but also to change the order of the incoming images at full readout speed.

In this work, the major parts of the DAQ system of the DSSC detector have been developed.

As the central part, the firmware modules of the two FPGA stages of the readout chain will be described. Furthermore, the control software including the design of the central library to configure and control the whole detector system is presented in detail.

This thesis is divided into four parts. The first part comprises some basic facts about the EuXFEL and the working principle of free electron lasers. During this introduction, the DSSC detector is presented to the reader including some details about the utilized sensors of the DSSC detector as well as the hardware components of the detector system.

In the second part, the implemented firmware modules of the system FPGAs are described. This includes the configuration and synchronization modules and, as the main section of this part, the modules of the readout chain. At the end of the firmware part, some simulations and measurements are shown in order to underline the successful testing and operation of the firmware components.

The third part includes the control and configuration software of the DSSC DAQ. This also includes the data receiver library which implements a highly optimized, multithreaded approach to enable the packet reception from the 10 Gigabit-Ethernet links at full speed and real time pixel sorting in order to directly store and visualize the image pixels as located on the focal plane. The software part includes also some aspects of the self-written measurement and data analysis framework which is frequently used by different groups of the consortium in order to characterize the system performance. The central aspect of the measurement framework is a standardized format of the data which is stored during measurements which allows to visualize and analyze the measurements automatically in an application. This turned out to be a very useful instrument to compare and exchange the measurements between the different groups and also to repeat measurements in different setups or environments.

Additionally, the more complex, automated pixel parameter trimming routines are described. With a rising complexity of the test systems and a rising number of pixels, the pixel characteristics have to be tuned automatically. Therefore, automated parameter sweeps have to be measured and analyzed by software in order to optimize the pixel behavior in the whole pixel matrix. These routines have been developed during this thesis to prepare the system for calibration measurements with the first 1/16th megapixel DSSC Ladder cameras. The successful realization of these complex methods will also be presented.

The last section of the software part includes the integration of the control software into Karabo, the new software framework, which has been developed by XFEL for beamline control, DAQ and calibration. Since this framework is the main software which will be used during experiments at XFEL, large effort has been made to realize stable and reliable control of the DSSC detector using this framework.

The last part of this thesis practically summarizes all achievements from the software and firmware part. It shows the successful implementation of the previously described components, which have been finalized shortly before the second beamline measurement campaign at the Petra III synchrotron during which important calibration measurements have been performed. The performance of the DSSC Ladder system during this campaign will be presented at the end of this thesis.

2

Background & Motivation

Since the discovery of X-ray radiation in 1895 by Roentgen, the radiation type has always been a fascinating and important source for new findings and technologies. With the progress of X-ray generating devices, the quality of X-ray radiation steadily increased. The short-wave radiation enables spectacular insights into the world of molecules or viruses e.g. the description of the double helix structure of the DNA molecules [1]. Simply the number of 19 handed Nobel prices, which are linked to X-ray radiation, underlines its importance in science.

With the advancement in X-ray science, new challenges and requirements have emerged. In the beginning, especially the penetrating property of X-rays attracted the most attention which allows to visualize the hidden structures in the interior of an object. Soon it became clear that in theory the small wavelength of X-rays can be utilized to image also tiniest structures down to the Å region.

But X-rays imaging procedures are much more complicated compared to an optical microscope since there are no materials which can be used for lenses or mirrors. Additionally, an X-ray microscope faces the problem that a tiny object also reflects few light and therefore the light intensity has to be very high in order to generate detectable signals from very small structures. The intensity problem could be solved by multiplying the structure in a crystal and therefore increasing the signal strength which is generated e.g. by Bragg reflections. This technique has successfully been applied to investigate the structure of a variety of crystalline materials and, due to evolving techniques in crystallization processes, also of more complicated structures like, for example, complex proteins. But this technique soon reaches its limit since not all structures can be crystallized or because the production of large crystals is very expensive.

In order to measure a comparable signal from a smaller crystal, the light intensity has to be increased. This relationship leads to the request for always stronger X-ray sources. For X-ray light sources, the intensity is measured in brilliance or spectral brightness, which is expressed in units:

$$Brilliance = \frac{N}{t \cdot A(\Delta\Omega)/(\Omega) \cdot (\Delta\lambda/\lambda)} = \frac{photons}{s \cdot mm^2 \cdot mrad^2 \cdot 0.1\%BW} \tag{2.1}$$

The expression 0.1% BW describes a bandwidth of 10^{-3} around the central wavelength λ

of the beam. Consequently, the brilliance is maximized, if the photon flux, i.e. the number of photons per second in a very thin wavelength region is large and the photons are emitted very densely from a small spot with little angular divergence.

With the discovery of the parasitic radiation at the General Electric 70-MeV synchrotron in 1947, a new effect has been found which enables to build very strong light sources [2].

The parasitic occurrence of synchrotron radiation is also described as light source of the 1st generation.

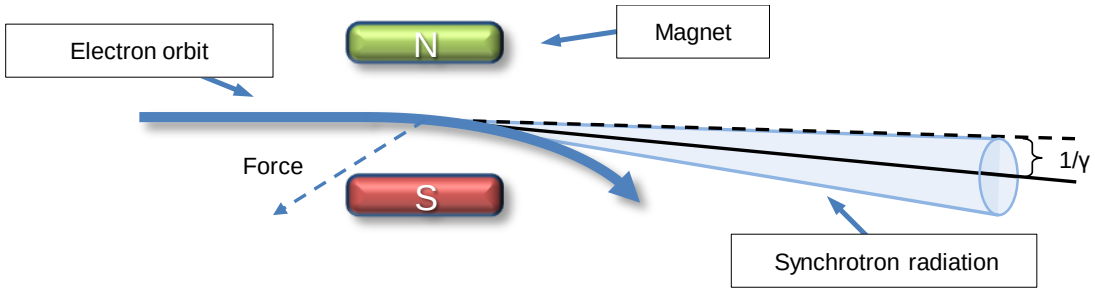


Figure 2.1.: **Generation of electromagnetic radiation from a relativistic electron beam.** Due to the Lorenz transformation, the emitted radiation is bundled in a narrow cone, tangential to the electron beam direction. The opening angle decreases with increasing energy γ of the electron beam. The power of the emitted synchrotron radiation scales with the forth power of the electron energy: $P_{rad} \propto \gamma^4$, where $\gamma = E_e/m_e c^2$.

Synchrotron radiation is generated from relativistic electrons which are deflected by a magnetic field (see figure 2.1). The induced orthogonal acceleration causes emittance of electromagnetic radiation. In case of relativistic particles, this radiation is strongly bundled in flight direction, forming a narrow cone which is directed tangential out of the machine. This type of light source provides very intense light with a continuous spectrum around the critical frequency

$$\omega_c = \frac{3c\gamma^3}{2r} \tag{2.2}$$

with the radius $r = r(B, \gamma)$.

The 2nd generation light sources are dedicated electron storage rings which are optimized for generation of synchrotron radiation. The introduced optimizations already cause a significant intensity increase.

The next evolution step to the 3rd generation light source has been made, when additional devices have been integrated into straight paths of the electron beams in order to further increase the brilliance of the emitted light.

So-called wigglers or undulators[1] apply alternating magnetic fields with a period of typically a few centimeters [4]. In these devices, the electrons are forced on a tight slalom

[1] A wiggler normally consists only of a few to a dozen alternating magnets, while an undulator is made up of significantly more magnets.

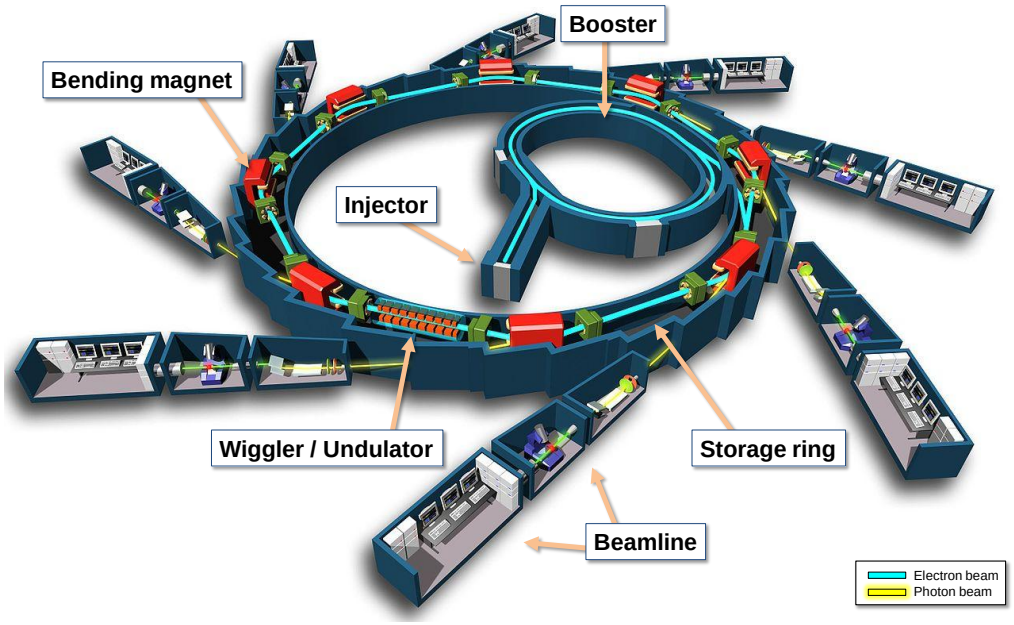


Figure 2.2.: **Schematic setup of a 3rd generation light source synchrotron.** Several beamlines, which contain the different experiments, are attached to the central electron storage ring. The electromagnetic radiation is generated in the bending magnets, wigglers or undulators [3].

path which leads to a greatly increased brilliance by four orders of magnitude, compared to the bending magnets of a synchrotron. The achieved light intensities enable to produce detectable signals from micro-crystals [5] which show only a few layers of their repetitive structure.

On the other side, the dramatically increased radiation dose introduces another problem. If the energy transfer to the sample during the imaging process becomes very high, the sample can be destroyed.

This problem can be dealt by cooling the probe down to very low temperatures which in turn has other drawbacks.

Another problem of X-ray microscopy with very small crystals is the decrease of the created interference signals. If the crystal becomes too small, the signal starts to vanish in the background. Since no lenses are available which can be used for X-ray radiation, X-rays can not be imaged like it is done in optical microscopy.

A lens works as a Fourier transformer and transfers the information from the frequency space into the real (local) space. Consequently, the diffraction patterns which are acquired in lensless X-ray diffraction imaging, contain information from Fourier space.

With Bragg reflections from periodic crystal structures, direct reconstruction is possible. But without crystallization, a reconstruction from the diffraction patterns requires phase information which is destroyed when the photon hits the detector. Nevertheless, with high computational effort and special techniques this problem can be solved.

Advanced techniques of lensless imaging require highly coherent X-ray light, as it is

generated e.g. by a LASER (**L**ight **A**mplification by **S**timulated **E**mission of **R**adiation). Since a synchrotron produces light with a continuous spectrum, coherent diffraction imaging (CDI) was only possible when the first light sources of the 3rd generation became available.

Consequently, X-ray microscopy faces several problems which hinder the progress towards higher spatial resolutions or the investigation of more complex molecular structures. Finally, the following lists summarizes the previously described effects:

- Small structures create only very small signals ⇒ increase of light intensity, or growing crystalline structures.
- Crystallization is expensive or not feasible with all structures.
- Smaller crystals are easier to grow but require higher light intensities.
- Very high light intensities destroy the sample during the imaging process ⇒ cooling of the sample.
- Cooling is not always feasible or changes the characteristics of the sample.

The long term goal is the imaging of single molecules and nanostructures, such as viruses, without first having to grow crystals. But the light intensities, which are necessary to achieve this goal, would destroy each molecule as soon as it is hit by the radiation. The only possibility is to compress the entire light intensity into such a short pulse, so that the light has already passed through the sample, before the parts of the molecule have begun to fly apart.

Exactly this is what the 4th generation light sources are able to provide. With the introduction of the first X-ray free electron lasers (XFELs), at the beginning of the 21st century, the first time monochromatic, highly coherent X-ray radiation of unparalleled intensity and brilliance became available. Their brilliance surpasses 3rd generation light sources by an unbelievable factor of 10 orders of magnitudes, visualized in figure 2.3. Additionally, the generated X-ray FEL pulses can be shorter than 100 fs, which is the timescale on which diffraction before destruction is possible [6].

2.1 Principle of an X-ray Free Electron Laser

Even if the name suggests a strong relation to a quantum laser, the principle of a free electron laser is more like a very sophisticated vacuum tube. An FEL consists mainly of two major components: an electron accelerator which generates a highly relativistic electron beam and a so called undulator (see figure 2.4), in which the electron beam is injected and the radiation is generated. The undulator consists of a long row of magnets of alternating polarity. The alternating magnetic field in the undulator forces the electrons on an tight oscillating path which induces spontaneous synchrotron radiation.

Due to special effects in the undulator the spontaneous emitted radiation is highly amplified which leads to strongly coherent, laser like radiation at the output of the FEL.

The basic principle of a free electron laser including the processes which happen in the undulator, will be roughly described in the following.

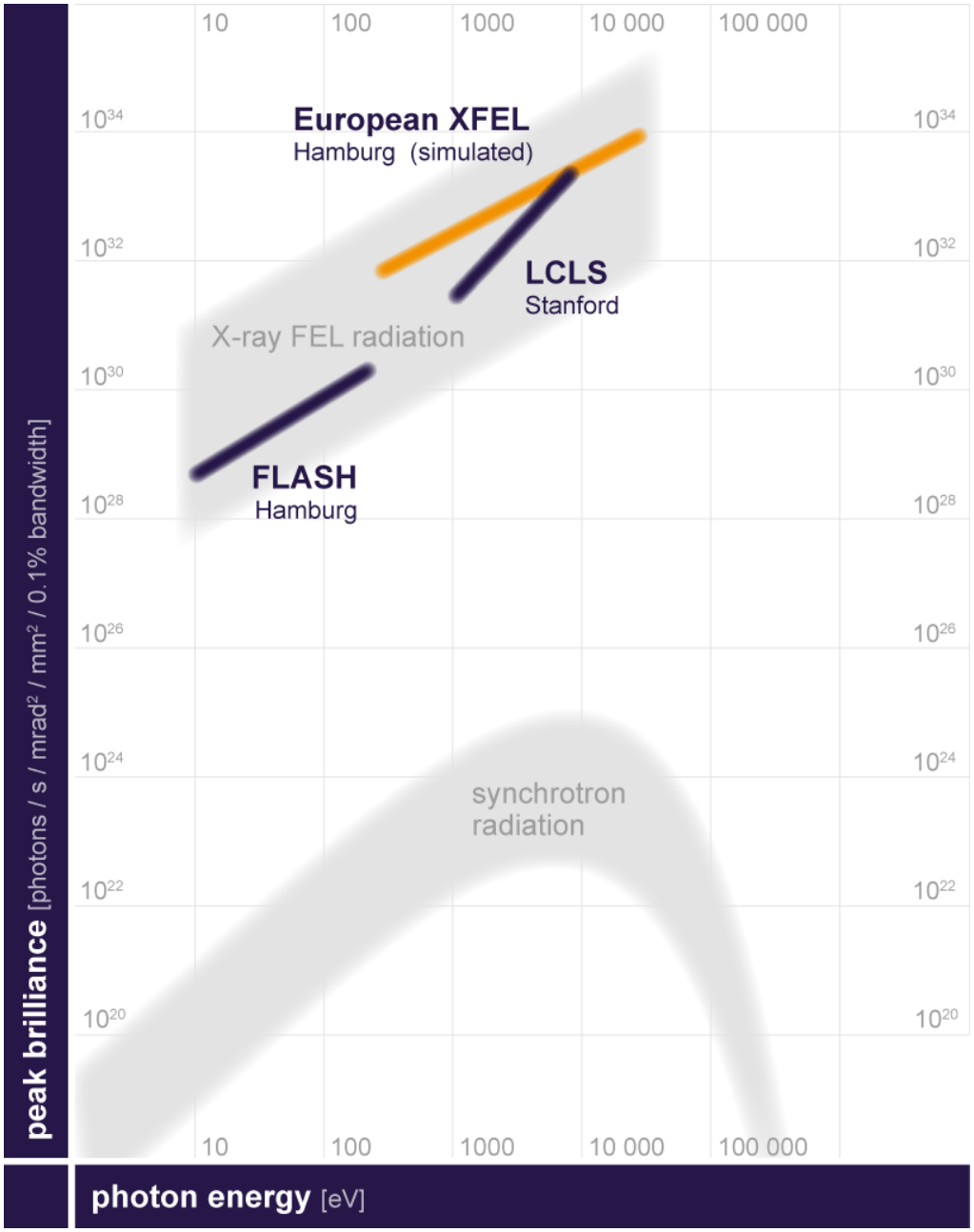


Figure 2.3.: **The peak brilliance of free-electron lasers compared to synchrotrons [7].** FELs exceed the brilliance of the 3rd generation synchrotron radiation sources by up to ten orders of magnitude.

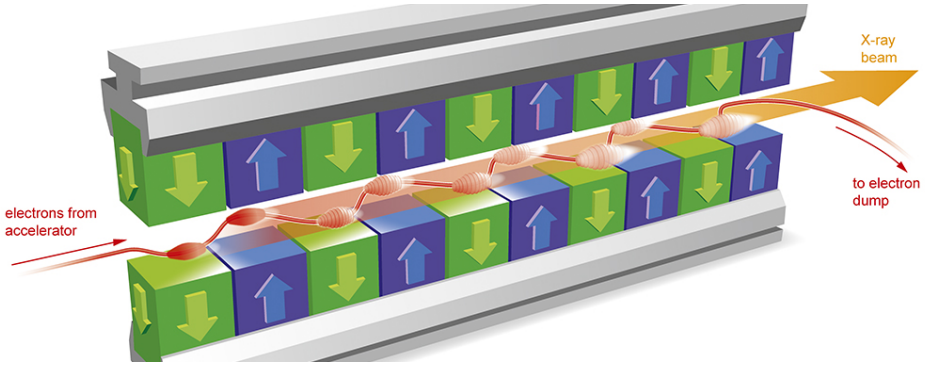


Figure 2.4.: **Schematic view of an undulator.** The alternating magnetic field forces the electron on an tight oscillating path which induces spontaneous synchrotron radiation [8].

The origin is a bunch of highly relativistic electrons which is injected into the undulator. The transverse and longitudinal velocities of the electrons are depended from the position z in the undulator, but can be averaged over one undulator period λ_u:

$$\overline{v_x} = \frac{cK}{\sqrt{2}\gamma}, \overline{v_z} = c(1 - \frac{1}{2\gamma_z^2}), \gamma_z = \frac{\gamma}{\sqrt{1 + K^2/2}}, K = \frac{eB_0\lambda_u}{2\pi m_0 c^2} \tag{2.3}$$

Ň Where the parameter K is the undulator parameter and $\gamma = E_e/m_e c^2$ the Lorenz factor. Since the electrons move always slower than the emitted light, a slippage occurs. After each undulator period, the delay of the electrons can be computed as:

$$\delta_z \approx (c - v_z)\frac{\lambda_u}{c} = \frac{\lambda_u}{2\gamma^2}(1 + K^2/2) \tag{2.4}$$

δ describes a phase advance of the electromagnetic wave. If δ is a multiple of the emitted wave length, constructive interference occurs which finally leads to the undulator equation:

$$\lambda_r = \frac{\lambda_u}{2n\gamma^2}(1 + \frac{K^2}{2}), \tag{2.5}$$

With n as an integer number. $n = 1$ represents the fundamental resonance and $n > 1$ higher harmonics. Since the resonance frequency of the FEL depends mainly on the electron energy and the undulator parameters, it can freely be chosen and the wavelength of the generated light hence tuned. The strong influence of the γ^2 factor in the equation leads to relaxed requirements on the undulator period if very small wavelength should be generated[2]. This tuneability differentiates a free electron laser significantly from a quantum laser, which can radiate only at defined wavelengths.

The intensity of the emitted light is strongly dependent of how the electrons are distributed along the z-axis inside the packet.

The spontaneously emitted light from randomly distributed electrons averages to an intensity which is proportional to the number of electrons of the packet ($I \propto N$). If all electrons would be at the same position, forming a macro particle of charge N, the

[2]Regarding relativistic electrons $\gamma = E_e/510$, E_e in keV.

light would be emitted fully coherent, resulting in an intensity which is proportional to N squared ($I \propto N^2$). Considering a typical number of electrons in a packet of 10^8 to 10^10, the difference in intensity is enormous. However, the generation of such a macro particle is technically not feasible, even with state of the art technology. But there exists a different technique which boosts the light intensity in a free electron laser.

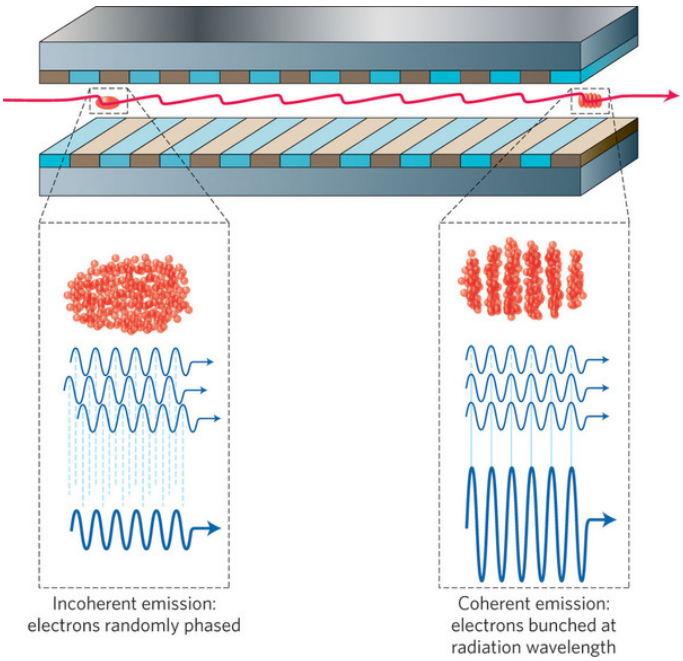


Figure 2.5.: **Bunching process in an undulator [9].** The former homogeneously distributed electrons interact with the emitted radiation. Electrons in front of the phase are slowed down and electrons after the phase of the emitted light are accelerated. Consequently, thin disks are formed. All electrons within one disk emit light synchronously while propagating through the undulator which highly amplifies the radiation intensity. This effect is proportional to the length of the undulator until saturation is reached.

During the propagation of the electrons through the undulator, frequencies near the resonance frequency λ_r are amplified. In this growing electromagnetic field, the electrons start to interact with the radiation as shown in figure 2.5. Besides the energy loss of the electrons to radiation, which decreases the electron velocity, some electrons gather energy from the electromagnetic field of the photons which leads to an acceleration. The effective fields from the magnetic field of the undulator in combination with the oscillating field of the electromagnetic radiation generate ponderomotive forces which lead to a modulation of the electron distribution.

The effective force accelerates electrons which are behind the field phase and decelerate the electrons which are ahead. Consequently, the electrons are bunched into thin discs while they travel along the undulator. The distance of the micro bunches is of the length of the emitted light. Since all electrons of a disk emit radiation at the same phase and, compared to the electrons of the successive disk, with a phase difference of 2π, a similar condition to the previously described macro particle is generated. When this state is reached, all electrons emit simultaneously photons of the same phase and wavelength which results in

fully coherent radiation of extremely high intensity.

The feedback of the generated radiation to the electrons can be summarized in three steps:

1. The emitted radiation travels faster than the electrons and interacts with the particles ahead.

2. The interaction modulates the electron distribution, forming thin disks

3. The creation of the disks leads to higher intensity of the electromagnetic field which increases the effect of 2). If the field becomes too strong, electrons slightly behind the radiation start to gather more energy from the field. This results in a saturation of the amplification.

As long as the radiation emittance in direction of the undulator is larger than the loss of radiation, a gain of intensity occurs. While at the beginning of the undulator the intensity increase is linear, it grows exponentially during the bunching process until the micro bunches can not further be compressed. After the saturation length z_{sat} the bunching is at a maximum and the electrons start to gather energy from the radiation field. At this point no further amplification of the radiation is possible. The development of the gain and different basic parameters of the FEL radiation over the undulator length is drawn in figure 2.6.

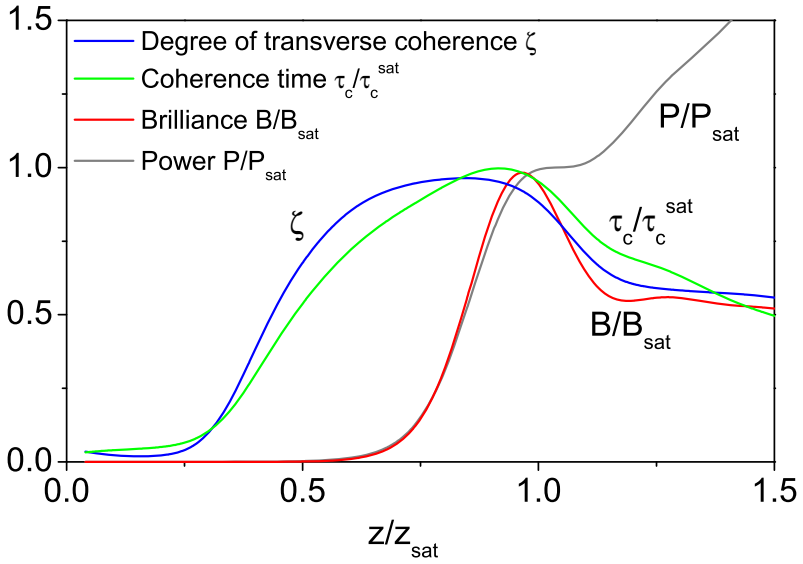


Figure 2.6.: **Schematic overview about the FEL properties along the undulator [10].** An FEL reaches the highest performance when the micro bunches can not be further compressed. The first saturation of an FEL in the soft X-ray region was reached in 2003 at the TESLA facility in Hamburg, Germany [11].

The bunching of the electrons requires strong electromagnetic fields from radiation. In order to start the bunching process directly at the entrance of the undulator, the process can be induced by a seed laser of the resonance wavelength. By inserting the undulator in

an optical resonator, the generated radiation can be trapped for several passes through the machine which additionally accelerates the bunching process.

Oscillator FELs are often used to generate light pulses in the visible region where conventional quantum lasers for seeding are available. For shorter wavelengths, where no highly reflective mirrors exists, X-ray resonators are not feasible[3]. Consequently, an X-ray FEL has to generate its full intensity during a single pass of the radiation through the undulator. The principle of an X-ray FEL is described in the following.

2.2 SASE Principle

The Self-Amplified Spontaneous Emission (SASE) principle is the technique which allows to generate high-intense FEL pulses in the X-ray region. Because there are no highly reflective mirrors available for X-ray radiation, the bunching process in the undulator has to be started from the spontaneously emitted radiation at the beginning of the undulator. Therefore, a very high beam quality in terms of electron emittance[4] and electron density in combination with very high electrical currents of several kilo Amperes have to be reached. Typically, a linear electron accelerator is required to provide a suitable electron beam.

A detailed mathematical description of the processes in a SASE FEL will go beyond the scope of this thesis. A variety of publications deal with the physics of free electron lasers and the SASE process: [14, 4, 15, 16]. A more general and well written description can be found in [9], which also includes a nice outlook in future FEL development. In [17], a comprehensive and detailed development of the mathematical model can be found.

A fundamental description of the FEL characteristics can be provided in a simplified one-dimensional approach, which already gives quite good results. Besides the former mentioned resonance wavelength λ_r and the undulator parameter K, the FEL or Pierce parameter ρ is used to describe some fundamental characteristics of a SASE FEL. The FEL parameter is given by [10, 18]:

$$\rho_{1D} = \frac{1}{4\gamma_0}\left[\frac{2I}{I_A}\left(\frac{\lambda_u K A_{JJ}}{\pi\sigma_e}\right)^2\right]^{1/3} \tag{2.6}$$

with the electron beam peak current I, the Alfven current $I_A \approx 17kA$, the undulator period λ_u and the Lorentz factor $\gamma_0 = E_0/m_e c^2$, representing the electron energy at the undulator entrance. The factor A_{jj} represents the coupling factor. For linear magnetic fields $A_{JJ} = [J_0(Q) - J_1(Q)]$ with J_n, the Bessel function of first kind and $Q = K^2/(2 + 2K^2)$. The factor σ_e is the rms transverse size of the electron beam.

The FEL parameter gives an estimation for saturation efficiency $\eta_{sat} \approx \rho_{1D}$ and radiation efficiency $\eta_r \approx 1/\rho_{1D}$ in the undulator. Since ρ_{1D} is inversely dependent on the electron energy $\gamma_0 = E_0/m_e c^2$, high electron energies lead to small values of the FEL parameter. In a SASE FEL, ρ_{1D} is in the order of $1/10000 \leq \rho \leq 1/1000$ [19]. This means a few thousand

[3] Proposals for X-ray resonators using Bragg crystal mirrors have been published [12, 13], but not applied so far.

[4] The beam emittance is the volume of the electron beam in phase space. A low emittance means a very dense electron packet, which does not spread during its path through the undulator.

undulator periods are required to reach saturation. The gain length of an FEL describes the travel distance during which the radiation power rises by a factor of $e = 2,718$:

$$L_g \approx \frac{\lambda_u}{4\pi\rho_{1D}} \tag{2.7}$$

In a SASE FEL, saturation is reached after ≈ 20 times the gain length $z_{sat} \approx 20 \times L_g$. With undulator periods of $\lambda_u \approx 50\,mm$, the saturation length of an X-ray FEL can easily become over 100 meters.

In the linear approximation, the radiation power in an FEL rises exponentially with the traveled distance z:

$$P(z) \approx \frac{E_0}{3\pi}\,\rho_{1D}^2\,\omega_r \exp\frac{z}{L_g} \tag{2.8}$$

Where $\omega_r = 2\pi c/\lambda_r$ is the resonance frequency. The effective power of the shot noise is assumed as the primary value which is amplified during the SASE process. With the power gain at saturation, which can be estimated by

$$G = \frac{P_{sat}}{P_{sh}} \simeq \frac{N_c}{3}\sqrt{\pi \ln N_c}, \tag{2.9}$$

where $N_c = I/\omega_r e \rho_{1D}$ is the number of cooperating electrons, the effective power of shot noise is given by:

$$P_{sat} = \rho_{1D} P_0 \Leftrightarrow P_{sh} \simeq \frac{3\rho_{1D} P_0}{N_c\sqrt{\pi \ln N_c}} \tag{2.10}$$

with the initial electron beam power $P_0 = \gamma_0 m_e c^2 I/e$.

The gain of a SASE FEL is hence proportional to the peak current I and the inverse of the FEL parameter ρ_{1D}.

A further parameter of interest is the coherence time of the laser radiation as a function of z with $z \leq z_{sat}$:

$$\tau_{coh}(z) \approx \frac{\pi}{\sigma_{\omega(z)}}, \tag{2.11}$$

with the radiation bandwidth

$$\sigma_\omega(z) = 3\sqrt{2}\rho_{1D}\omega_0\sqrt{\frac{L_g}{z}}. \tag{2.12}$$

The coherence length also shows a maximum at $z = z_{sat}$.

A very central parameter is the duration of the X-ray pulse τ. Since the velocity of the injected electrons is $v_e \approx c$, the length δ_e of the electron packet gives a good approximation:

$$\tau = \delta_e/c \tag{2.13}$$

For a given maximum electron density, the length of the electron packet is always dependent on the number of electrons which are contained in the packet. Consequently, the pulse duration can not be easily reduced, due to the current strength which is required to start the SASE process. Typical lengths are of the order of a few tens of μm which results in pulse durations of $\approx 100\,$fs.

These formulas can only give a rough guideline to estimate the main parameters of a SASE FEL but can be used to give a basic understanding of the dependencies of different parameters.

2.2.1 Progress of SASE FEL technology

Since the theoretical description from John Madey in 1971 of a device which is able to amplify X-ray radiation in a periodic magnetic field [14], lots of research and development followed over two decades until the first FEL laser shot in the X-ray region could be realized. The first radiation from a SASE FEL was reached in September 2000 at the Advanced Photon Source (APS) in Argonne at the National Laboratory. The FEL reached saturation at 385 nm in the near-UV region [20]. Only one month later, the TESLA facility in Hamburg, Germany, which implements a superconducting linear electron accelerator, also reached SASE operation the first time at a wavelength of 109 nm [21]. After continuous improvements of the superconducting technology, the first soft X-ray laser pulses have been generated by the Free Electron Laser in Hamburg (FLASH), which evolved from the TESLA facility. FLASH can be regarded as a smaller, early version of the EuXFEL. The generated light pulses of this SASE FEL reached in 2006 a wavelength of 13 nm (95 eV) [22], which could be further decreased to 4.12 nm (300 eV) in 2010 [23].

Today, the world record of the shortest laser wavelength generated by a SASE FEL is held by LCLS in Stanford, California. By injecting electrons, accelerated to up to 14 GeV, into an undulator of 132 m length, this machine is able to generate X-ray flashes of a wavelength of 0.12 nm (10 keV) [24]. But it is only a question of time until the new EuXFEL in Hamburg will put a new record with wavelengths below 1 Å, after the machine started its operation in September 2017.

2.3 Experiments at 4th Generation Light Sources

4th generation light sources will enable new insights into the world of molecules which have not been possible before. The enormous light intensity, which can be generated by a SASE FEL, promises to realize imaging of single viruses and complex molecules for the first time, without the need of difficult and expensive crystallization. Pulse durations of 100 fs and shorter will allow to dissolve processes on a time scale on which chemical reactions take place.

The low bandwidth and long coherence length of SASE FEL X-ray radiation enables new techniques and approaches in the field of lensless coherent diffraction imaging [25]. But reconstruction of structures from the acquired diffraction images is difficult and requires intense usage of computers. Powerful algorithms have been developed [26, 25, 27] to solve the phase problem which is needed in order to reconstruct the diffracting structure from the blurry diffraction image (see figure 2.7).

Furthermore, very special challenges are set on the experiment setup and its execution under the special conditions of an X-ray FEL beamline. Anything which is illuminated by the laser shot will immediately be destroyed because of the high radiation dose. The solution of this problem is given by the short pulse duration. As shown in experiments at FLASH [28] and LCLS [25, 29], it is possible to image complex structures with a single X-ray pulse from a SASE FEL, even if the sample is destroyed during the imaging process. The method is called 'diffraction before destruction' [30]. If the light pulse duration is in the order of 100 fs or shorter, the light has passed the sample before any movement of the exploding molecule can blur the image. The feasibility of this method has impressively been shown at FLASH with soft-X-ray radiation [28] see figure 2.7. Due to this technique, the radiation

intensity can now freely be increased since the destruction of the sample does not interfere with the resulting image. Theoretically, this enables to create and detect intense signals also from smallest structures.

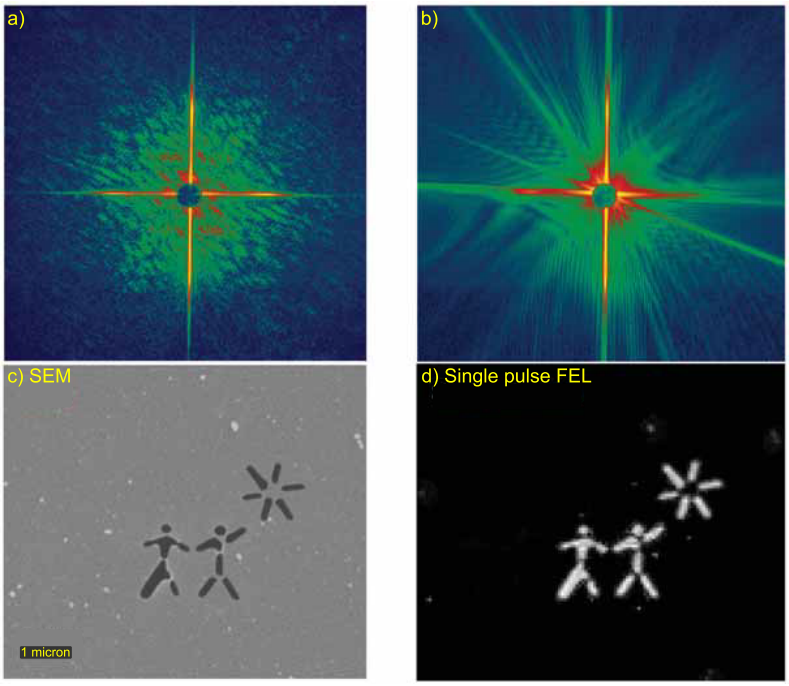


Figure 2.7.: **Diffraction before destruction prove of principle at FLASH [31].** c) Ground truth, before the sample has been destroyed. The ground truth has been acquired using a scanning electron microscope (SEM). a) Diffraction pattern of a single soft X-ray shot, before destruction. b) Diffraction pattern after destruction. The pattern suggests massive changes of the sample. d) Reconstructed pattern from a), the spacial resolution has been estimated to 62 nm.

Due to the destruction of the sample, sophisticated approaches to provide continuous supply of identical samples to the pulse trains are required if e.g. a video of a chemical process or a 3D model of a molecule should be acquired. One technique is, for instance, the sample injection which has been developed at LCLS and published in 2011 [29]. The goal of the experiment was to generate a 3D model of a Mimi virus, which is the largest known virus. Using a special sample injector, a very thin stream of aerosolized Mimi viruses is injected with high velocity (60 - 100 m/s) into the interaction zone of the experiment. The X-ray flashes hit the injected viruses with an arbitrary orientation. In order reconstruct the 3D model, 10^6 diffraction patterns had to be reconstructed and sorted by the virus orientation.

3

The European XFEL (EuXFEL)

After the new free electron laser technology advances and the first machines could be driven to generate intense light flashes of shorter and shorter wavelength, the scientists started to think about the construction of the largest FEL which could generate light flashes in the hard X-ray region of unparalleled intensity.

The cornerstone of the EuXFEL was laid down in 1992 when the new TESLA (Tera Electron Volt Energy Superconducting Linear Accelerator) technology development started. For shorter wavelengths, the beam quality had to improve. Using superconducting accelerator elements promised a great improvement in this challenge. Finally, the expectations have been fulfilled. The superconducting TESLA elements are able to generate a very thin and homogeneous electron beam which provides sufficiently high beam quality and electron currents for an advanced X-ray Free Electron Laser source of unprecedented peak brilliance[1] and full transverse coherence.

3.1 Historical Development of the European XFEL

In DESY, Hamburg, where the EuXFEL is constructed, there exists a lot of experience with constructing free electron lasers. The TESLA Test Facility for free electron experiments (TTF FEL) has been constructed in order to provide a linear accelerator with high quality electron beams. The TESLA Test Facility implemented a 100 m long linear accelerator which provides an electron beam with an electron energy of 233 MeV. Injected in an undulator, the first photon flashes have been generated by TESLA in 2000. The ultraviolet light flashes reached wavelengths down to 109 nm, a new world record at this time. By increasing the electron energy, the wavelength of the TESLA device could be further decresed down to wavelengths between 80 and 180 nm. By applying further optimizations for operation in the region between 80 nm to 120 nm, the maximum power gain of the order of $\approx 10^7$ (saturation) could be reached over the entire wavelength range [11], [21], [32].

From this very promising device, a new test facility emerged. The Free Electron Laser in Hamburg (FLASH) was build with the goal to generate laser light of only 6 nm wavelength. The new 260 m accelerator increases the electron energy up to 1 GeV. By injecting this

[1] Brilliance = [photons / s / mm^2/ mrad^2 / 0.1% bandwidth]

high energy electron beam into the undulators of 30 m length, the FLASH device generates high intensive ultra short laser flashes of wavelengths in the ultraviolet and soft X-ray region. When FLASH became operational in 2005, the FEL already reached a wavelength of 32 nm [33] during SASE operation - a new world record. By further upgrades of the accelerator modules, the electron beam energy could be increased and therefore, the generated wavelength decreased. After the upgrade in 2007 the design goal of 6.5 nm has been reached for the first time. During the last upgrade in 2009/2010, when the electron beam energy was additionally increased to 1.25 GeV, an even shorter wavelength of 4.1 nm has been published [23]. This means a fundamental advancement, because this wavelength opens the insight into the so called water window, the wavelength range between 2.3 and 4.4 nm, in which water is nearly transparent for light. This can be used to investigate probes in liquid solutions like in vivo biological experiments.

These promising results and experiences from the TTF and FLASH systems led to the very ambitious plans to build the large EuXFEL.

3.2 Basic Parameters of the EuXFEL

The previous achievements of X-ray FEL technology encouraged in 2007 the German government to start the development of the first superconducting X-ray FEL for ultra-short X-ray flashes and high repetition rates for life sciences and material research [34]. The construction started in 2009 in cooperation with the European Union, the state of Hamburg, Russia, Poland and 9 other contributing countries.

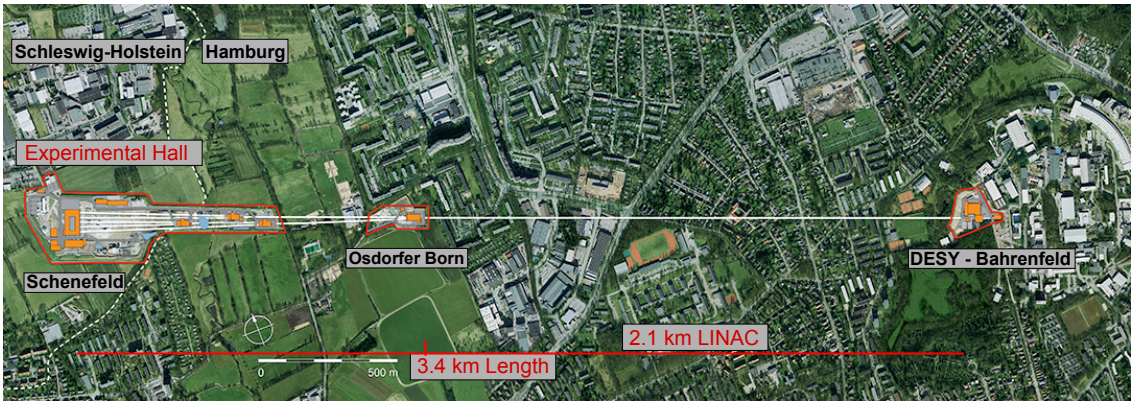


Figure 3.1.: **Layout of the EuXFEL in Hamburg [8].** The machine is built in a 3.4 km long tunnel below Hamburg from DESY to the experimental hall in Schenefeld. Most of the tunnel is filled with the 2.1 km long superconducting LINAC, which accelerates the electron beam to 17.5 GeV. In Osdorfer Born, the beam is distributed and transported to the undulators, in which the X-ray radiation is generated. Finally, the X-rays are transferred to the beamlines in the experimental hall.

The main parts of the EuXFEL can be summarized as follows:

- the injector,
- the superconducting linear accelerator (LINAC),
- the beam distribution system,
- the photon beamlines,
- the 5 experimental stations.

The injector is based on a high quality radio frequency (RF) gun. Electrons are extracted from a photo cathode by a visible laser and then accelerated to 120 MeV. Beam focusing elements and diagnostic equipment is implemented to ensure the high beam quality as well as a low emittance and low energy spread.

The tunnel contains the long superconducting LINAC of 1.6 km length, which accelerates the electrons to energies of up to 20 GeV [31], while the nominal electron energy during operation is aimed at 17.5 GeV. Two stages of beam compressors shorten the electron bunches by a factor of 100 which increases the peak electron current to 5 kA. This high electron current is required to trigger the SASE process in the undulators. Further equipment is installed to align and collimate the electron beam.

The beam distribution system at the output of the accelerator contains three very long high precision undulators. The SASE 3 undulator has a length of 100 m. SASE 1 and 2 even 200 m. Furthermore, the opening gap of each undulator can be varied providing certain tuneability of the wavelength of the generated laser pulses. Besides the three large SASE undulators, two smaller devices U1 and U2 for ultra short highly coherent X-ray radiation are planned in separate tunnels providing additional possibilities for high photon energy experiments.

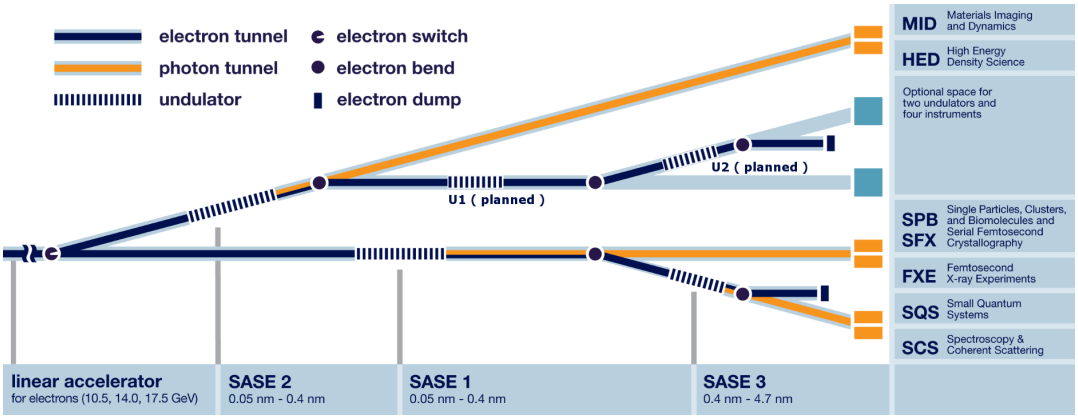


Figure 3.2.: **The beamlines at EuXFEL [31].** Five tunnels will provide spots for 10 different experiments. At commissioning, only three tunnels will be equipped. Each undulator is assembled from 5-meter-long segments. The strength of the magnetic fields and the distance between the magnetic rails can be changed which allows to tune the generated wavelength.

All five tunnels open out into the experimental hall. At each output, an experiment hutch provides space for 2 individual experiments.

The large variety of undulators and light generating devices will be used to provide a wide spectrum of coherent laser light of unparalleled intensity for scientific experiments. Main focus was aimed on the region around an energy of 12.4 keV. As shown in table 3.2 two of the three SASE undulators operate in this region.

	Photon energy [keV]	Wavelength [nm]	Polarisation	Tunability	Gap variation
SASE 1	12.4	0.1	Linear	No	Yes
SASE 2	3.1 - 12.4	0.4 - 0.1	Linear	Yes	Yes
SASE 3	0.3 - 3.1	4.9 - 0.4	Circ./Linear	Yes	Yes
U1,U2	20 - 100	0.06 - 0.013	Linear	Yes	Yes

Table 3.1.: **Design parameters of the five undulators at the European XFEL Facility corresponding to an input electron energy of 17.5 GeV [31].**

3.3 Timing Scheme of the Generated X-Ray Pulses

Additionally to the enormous light intensity, the European XFEL will provide a pulse number and a pulse rate which puts the EUXFEL years ahead on any other existing light source. At the beginning of each X-ray pulse stands an RF electron gun. The electron gun emits and generates the electrons which are injected into the main accelerator. For the EuXFEL, a very special high speed RF electron gun has been developed in the Photo Injector Test Facility at DESY, Zeuthen (PITZ) [35, 36].

The high speed RF electron gun is driven by a 1.3 GHz radio frequency and generates electron flux from 0.02 nC to 2 nC at a repetition rate of 10 Hz. During each RF pulse, a train of 2700 electron bunches with a distance of 220 ns is generated [37] resulting in a final pulse repetition rate of 4.5 MHz and a pulse rate of 27000 pulses per second. Between two pulse trains, a long pause of 99.4 ms follows, which is used for cooling of the electronics and to readout the image data from the pixel detectors. This scheme defines the final X-ray pulse scheme, which is shown in figure 3.3.

The PITZ RF Gun has been optimized for electron beams of high electron currents combined with very low transverse emittance, which is a critical parameter for the X-ray generation processes in the SASE undulators. This low emittance can only be conserved over the whole accelerator line by the highly optimized injector setup. Further information about the emittance compensation process can be found in [35].

3.4 Comparison with other SASE FELs

The EuXFEL is planned as the brightest and most brilliant X-ray coherent light source which has ever been built. Its image repetition rate is unparalleled by the existing FEL experiments. table 3.2 summarizes some basic facts about the EuXFEL in comparison with other existing SASE FELs.

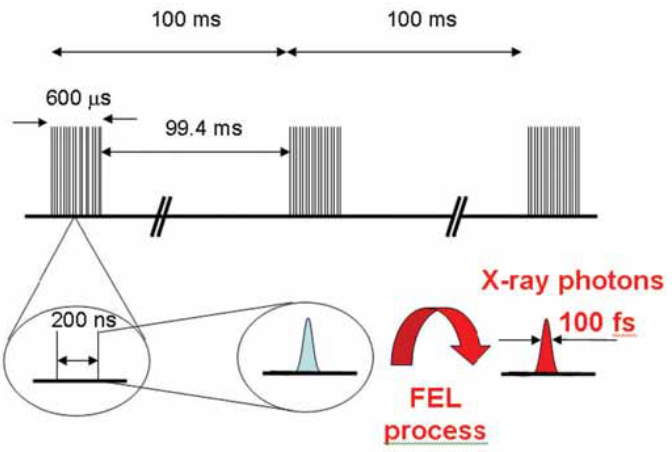


Figure 3.3.: **Pulse scheme delivered by the EuXFEL [31].** The maximum pulse repetition rate is 4.5 MHz. The 99.4 ms pause is used for cooling and readout of the detectors.

3.5 Planed User Applications

From the previous advances with other SASE FEL facilities, six different experiments have emerged, which will be installed at the EuXFEL beamlines in the EuXFEL experimental hall.

One of the major breakthrough of the FLASH experiments was the reconstruction of an image from a non crystalline structure which has been reconstructed from one single high-resolution diffraction image generated by one single laser shot [28]. This showed that it is possible to create images from nanoparticles viruses or cells, using extremely high light intensities without the need for crystallizing the samples first. This experiment laid the cornerstone for a variety of ultra-fast coherent diffraction imaging experiments of single particles, clusters and biomolecules. Besides femtosecond diffraction experiments, additional experiments for material science will be permanently installed at the beamlines [31, 38]:

- **FXE:** Femtosecond X-ray experiments: time-resolved investigations of the dynamics of solids, liquids, gases.
- **SBP/SFX:** Ultrafast coherent diffraction imaging of single particles, clusters and biomolecules: structure determination of single particles (atomic clusters, biomolecules, virus particles, cells), serial femtosecond crystallography.
- **HED:** High energy density matter: investigation of matter under extreme conditions using hard X-ray FEL radiation, e.g. probing dense plasmas.
- **MID:** Materials imaging and dynamics: structure determination of nanodevices and dynamics at the nanoscale.
- **SCS:** Spectroscopy and coherent scattering: Electronic and atomic structure and dynamics of nanosystems and of non-reproducible biological objects using soft X-ray.
- **SQS:** Small quantum systems: investigation of atoms, ions, molecules and clusters in intense fields and non-linear phenomena.

	LCLS	SACLA	European XFEL
Abbreviation for	Linac Coherent Light Source	SPring-8 Angstrom Compact FEL	European X-ray FEL
Location	California, USA	Japan	Germany
Start of commissioning	2009	2011	2016
Accelerator technology	normal cond.	normal cond.	supercond.
Number of light flashes per second	120	60	27000
Minimum wavelength	0.15 nm	0.08 nm	0.05 nm
Maximum photon energy of the laser light	8.3 keV	15.5 keV	24.8 keV
Maximum electron energy	14.3 GeV	$6-8$ GeV	17.5 GeV
Length of the facility	3000 m	750 m	3400 m
Number of undulators	1	3	$3-5$
Number of experiment stations	$3-5$	4	$6-10$
Peak brilliance	2×10^{33}	1×10^{33}	5×10^{33}
Average brilliance	2.4×10^{22}	1.5×10^{23}	1.6×10^{25}

Table 3.2.: **The EuXFEL, compared to actual FELs [7].**

Each experiment has its own, special demands on the light source at which it is permanently installed. For instance the angular coverage or the photon energies, which vary from 0.05 nm (24.8 keV) to 4.7 nm (0.26 keV) depending on the installed SASE undulator. In order to acquire the diffraction patterns from the different experiments, highly specialized X-ray detectors are required. The demanding environment at the EuXFEL beamlines propose challenging requirements to the 2D detectors:

- A pixelated focal plane of 1024×1024 pixels.
- A large sensitive area, since the outer regions of a diffraction image contain information about high frequencies, and thus information about small structures.
- High sensitivity, which means single photon resolution over the whole photon energy range
- Large dynamic range of up to 10^4 photons per pixel.
- The possibility to form a variable hole in the center of the detector to let the unscattered photon beam pass the detector without destroying the device.
- High image acquisition rates of 4.5 MHz are obligatory.

Since no detector type can provide single photon resolution for photon energies of 0.26 keV and at the same time for 24.8 keV, three different detectors are designed for the

energy ranges which are given by the three different SASE undulators. The three large 2D megapixel detectors will be available at the European XFEL experimental hall and are briefly presented below. Subsequently, the main properties and differences of the three detectors are summarized in table 3.3.

3.5.1 The Adaptive Gain Integrating Pixel Detector (AGIPD)

The AGIPD [39] detector is designed for high energy experiments in the range from 3 - 15 keV at SASE1 (SPB/SFX) and SASE2 (MID). It is developed by a collaboration of institutes, led by a group of DESY. The detector focal plane implements 500 μm thick PIN (p-in-n) Silicon sensors, which are segmented to pixels of 200 μm pitch. The sensors are bump bonded to custom readout ASICs, which implement a dedicated channel for every readout pixel. Each readout channel provides three different input capacitors, which are automatically selected by the incoming amount of charge. Each pixel contains an analogue storage matrix which provides space for 352 image values. Additionally, the respective gain setting is also stored for each image. During readout, the analogue signals are transferred to external 14-bit analogue-to-digital converters (ADCs). For each module (512×128 pixels) 64 ADCs are implemented in order to provide sufficient bandwidth to read out all channels. In combination with the respective gain setting information, the digitized values are transferred out of the system.

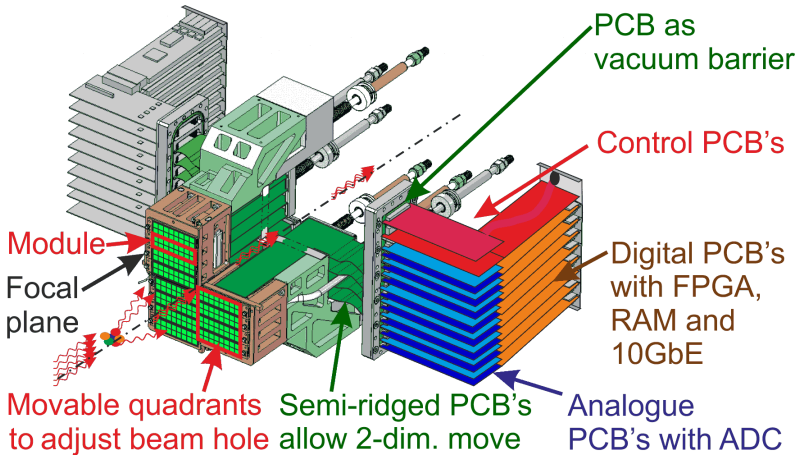


Figure 3.4.: **Schematic view of the AGIPD megapixel detector [39].** From the analogue in-pixel storage the signal value is digitized by external 14-bit ADCs during readout. The detector implements storage cells for 352 images.

The detector is split into 16 super-modules, each of which can be operated as an independent detector system. The readout chain of the AGIPD detector splits into four parts:

1. The focal plane, implementing 512 × 128 sensor and readout pixels,
2. A flexible board, which allows focal plane movement and leads out of the vacuum,
3. The ADC board, implementing 64 ADC channels,

4. A digital FPGA board, implementing Ethernet packet generation and a 10 Gigabit-Ethernet link.

Since the detector is foreseen for high energy photons, it has to deal with very high radiation doses. Even if the readout ASIC is shielded by the thick Si-sensor, the expected radiation dose of the first three years is expected to lay at more than 10 MGy, which has to be tolerated by the system.

3.5.2 The Large Pixel Detector (LPD)

The LPD [41] detector is named after its large pixel format with a 500 μm pitch. The main detector development is carried out by the Science and Technology Facilities Council (STFC) at the Rutherford Appleton Laboratory in Oxfordshire, United Kingdom. LPD will be used in the high energy range of 5 - 20 keV. At this photon energies, experiments in air environment are possible. Additionally, the detector provides a very high dynamic range of 10^5 photons at 12 keV. Thus, the LPD provides ideal properties for the FXE experiment at SASE1.

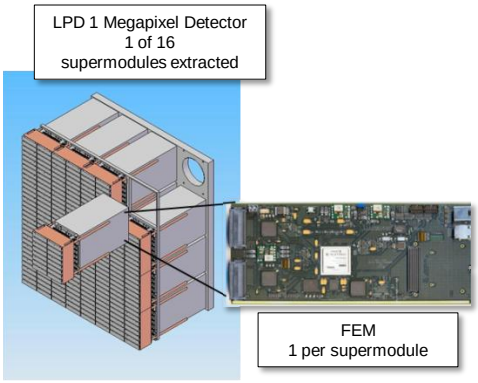


Figure 3.5.: **Schematic view of the LPD 1M Pix detector [40].**

The custom sensor of LPD is made of 500 μm thick high resistivity silicon. Since the readout ASIC shows a smaller pixel pitch than the sensor, an additional interposer die, which has to implement interconnections and through-vias, is glued between sensor and readout ASIC. In addition to the connection of the sensor to the readout pixels, the interposer provides additional protection for the ASIC from the high-energy X-ray radiation which leads to more relaxed design constraints.

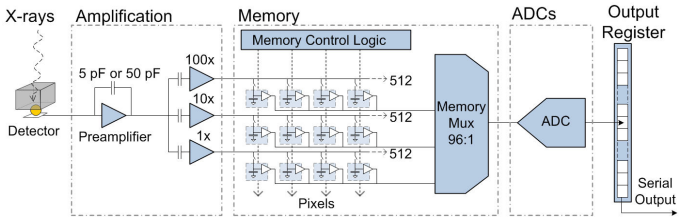


Figure 3.6.: **Architecture of the LPD readout ASIC [41].**

The first stage of the front-end in the readout ASIC is a preamplifier with two switchable feedback capacities of 5 pF resp. 50 pF. With the 50 pF-capacity enabled, the detector provides a dynamic range of 10^5 photons at 12 keV. Using the smaller capacity, better noise and higher gain can be achieved with significantly lower dynamic range. The pre-amplified signal is then fed into three parallel gain stages (see figure 3.6). At the output of all three gain stages, an analogue shift register of length 512 is connected, which stores the acquired signals during the image acquisition phase. The selection of the appropriate gain level, and

hence which value is digitized, is done by the data acquisition system during the readout phase. The digitization is realized by an on-chip successive approximation register (SAR) 12-bit ADC. In an ASIC, 16 ADCs are implemented, each connected to 32 pixels, which sums up to 512 pixels per ASIC. 128 ASICs are combined into a Supermodule. The megapixel detector is composed of 16 Supermodules.

The readout chain of the LPD consists of two stages: the readout ASIC and the Front-End Module (FEM) [40]. The FEM implements two Spartan-6 FPGAs, a Xilinx Virtex-5 FPGA, which is connected to a DDR2 memory. The spartan-6 FPGAs are connected via 100 MHz LVDS to the 128 readout ASICS of a Supermodule and pass the data to the Virtex-5. In the Virtex-5, an algorithm selects the optimal signal to noise sample from the 3 gain settings from each pixel in order to reduce the amount of data. The selected values are finally transmitted to a plug-in FMC mezzanine card, which implements a 10 Gigabit-Ethernet connection to send the data from the detector to the XFEL-DAQ system.

3.5.3 The DEPFFET Sensor with Signal Compression (DSSC) Detector

Since the work, which is described in this thesis, is associated to the DSSC detector, only a brief summary of the basic properties of the DSSC detector is given at this point. The detailed description of the system starts at chapter 5.

The DSSC detector is the most sensitive detector among the large 2D detectors. It is designed for the low energy range from 0.25 - 6 keV. In the original design, the sensor pixels implement DEPFET sensors with a non linear characteristic which generates an internal signal compression. This type of sensor shows very low noise characteristics which enables single photon resolution down to photon energies of 0.5 keV. Due to technical difficulties and political and personal reasons, an alternative type of sensor in combination with a corresponding version of the readout front-end has been developed. The second sensor is a mini silicon drift detector (MSDD), which shows almost comparable properties and noise values.

Both sensor types are bump bonded on a readout ASIC, which implements one readout channel per sensor pixel. The current signal from the DEPFET sensor is then shaped by a trapezoidal filter for low noise performance. When using the MSDD sensor, the charge signal must first be translated into a current signal so that the circuits, designed for the DEPFET readout, remain usable. The signal from the filter is directly digitized by an in-pixel Wilkinson [42] 9-bit ADC. To store the digitized values during the image acquisition phase an SRAM memory is installed in each pixel which holds space for 800 images.

The good noise performance of the DSSC detector is required for the experiments which are installed at the low energy beamline SASE3: SQS, and SCS. Additionally, DSSC is also foreseen for SPB at SASE1.

3.5.4 Comparison of the three large 2D Detectors

The three large megapixel detectors provide special characteristics in order to optimally meet the needs of the different experiments. In table 3.3, the main differences are summarized.

	LPD	AGIPD	DSSC
Sensor Type	Silicon p-in-n diode	Silicon p-in-n diode	DEPFET / MSDD
Pixel Size	500×500 μm^2	200×200 μm^2	204×236 μm^2
Energy Range keV	3-13 keV	3 - 13 keV	0.25 - 6 keV
Dynamic Range	10^5 @ 12 keV	4×10^4 @ 12 keV	6000 @ 1 keV
Compression Concept	Three parallel gain stages, optimal gain is selected by DAQ during readout.	Automatic gain adaption during acquisition, by selection of feedback capacity.	DEPFET: nonlinear sensor characteristic. MSDD: nonlinar front-end characteristic.
Storage Concept	Analog shift register	Analog storage matrix	Digital SRAM memory
Storage cells	512	352	800
Number of 10GE Links	16	16	16 in 4 QSFP+ Links
Average Data Rate [GBit/s]	86	59	134

Table 3.3.: **Overview about the three megapixel detectors built for the EuXFEL experiments.**

4

The XFEL Beamline Environment

This chapter briefly summarizes the infrastructure which is provided by XFEL at the beamline.

This comprises the Clock and Control system, which provides global clocks and trigger signals for synchronization and the Trainbuilder device, which is defines the first level of the XFEL DAQ system.

Finally, a short general introduction in Supervisory, Control and Data Acquisition (SCADA) systems is given, including some examples which leads to the last section in this chapter, the presentation of the newly developed Karabo framework, the SCADA system for the XFEL beamline control and back-end DAQ.

Basic functionality and important information about the C&C system and the Trainbuilder systems will be given in the following. Additional information about these systems can also be found in resp.

4.1 The Clock and Control System

The Clock and Control (C&C) system provides global clock signals and timing telegrams which are used to synchronize the image acquisition phase with the accelerator operation and the incoming photon pulses.

The system hardware houses in a MTCA[1] crate. The crate provides a scalable platform which allows the use of commercial cards as well as custom self designed hardware combined in one very compact system.

One C&C system is built by a combination of a DAMC2 board connected to one CC RTM fanout card. In the same crate, one C&C pair can be configured as a master and extended with further slave pairs to provide support also for larger megapixel cameras.

For full operation, the crate requires also a commercially available high-performance CPU card, and a custom Timing Receive (TR) card, which was also designed by DESY. The Timing Receiver gets the signals from the global XFEL system and transmits a 4.5 MHz clock signal and trigger signals to the timing master. The slave cards just work as fanout

[1] Micro Telecommunications Computing Architecture: Widely used in large I/O bound physics applications. Modular, open standard for high performance switched fabric computer systems in a small form factor.

Figure 4.1.: **The DESY DAMC2 board and the C&C RTM connected [43].**

cards. For synchronization, the 4.5 MHz clock is divided into a 100 MHz clock, which is distributed to the FEMs with the other trigger signals. Triggers and Telegrams are also generated in the TR and distributed by the master-slave combination.

All signals from the slave cards are distributed via LVDS signals to the FEMs, which are transmitted by standard CAT Ethernet cables over distances up to 30 m. The C&C interface implements four individual signals for detector synchronization:

- The **99MHz clock**, derived from the 4.5 MHz accelerator clock.
- The **Start** signal, which signals the arrival of the next pulse train after exactly 15 ms.
- The **Veto** signal, which is used to mark the acquired image as not relevant. The memory address can then be overwritten by a later event.
- The **Status** signal. Over this signal a status from the FEE can be received.

The assignment of the signals to the RJ45 connector is shown in figure 4.2

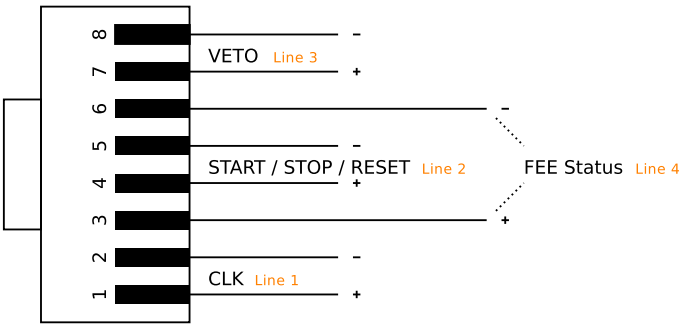


Figure 4.2.: **Signal assignment of the C&C interface to the RJ45 connector [43].**

All fast command telegrams are sent synchronously to the 99 MHz clock, which is derived from the 4.5 MHz clock from the TR. In order to provide system wide synchronism, this clock is also used by the detectors to derive relevant, internal clock signals.

As shown in table 4.1, the START signal transports the fast timing telegrams which are used to signal the start and the end of incoming trains.

Command	Command bits (4)	Payload (80)	Description
START	1100	(64) TrainID + (8) Bunch Pattern Index (8) Checksum	Signals incoming train after 15 ms
STOP	1010	none	Signals end of train
RESET	1000	none	Reset FEM
Reserved	1111	none	

Table 4.1.: **Fast command telegrams of the C&C protocol.**

Together with the START command, a 64-bit TrainID is transmitted. This long range counter value defines a unique identifier for each acquired pulse train. At the end of each pulse train, a STOP command signals the end of the train. The second signal transports the Veto commands.

4.1.1 Veto mechanism

During the image acquisition phase, the FEE produces pixel data for every single incoming bunch until the internal memory is filled. This generates a very high amount of image data which has to be stored inside the FEE until readout. But image storage is limited and the FEE can only store a fraction of the 2700 incoming pulses. In order to optimize the utilization of the available memory capacity, the Veto mechanism was introduced. Therefore, the C&C system is also connected to the XFEL Veto system [44]. The Veto system provides special Veto commands which can be used to mark an image as probably bad and enables the FEMs to reuse the memory position of the vetoed image.

The Veto commands are generated in a special device which is called Veto source. The Veto source can be any device that provides information that can be used to make a decision as to whether an image can be removed from the memory. Thereby, the latency, with which the information is provided, has to be as low as possible.

All Veto sources are connected to the global Veto unit which distributes the Veto decisions to the C&C system. Veto decisions are also distributed via the C&C interface.

Possible Veto commands are:

- **Veto:** if an image is marked as Veto, the measurement was most likely not successful and the FEE can overwrite the marked image.
- **GOLDEN:** if an image is marked as GOLDEN, there is a high likelihood that the measurement for that image was very good. The FEE has to ensure that this image will be kept in storage.
- **NO-Veto:** this is the default decision. If an image can neither be classified as Veto nor as GOLDEN, it will be marked as NO-Veto. Different reactions are possible depending on the policy of the Veto user.

A detailed specification of the different Veto commands is shown in table 4.1.1. Examples and further information can be found in [44].

Command	Command bits (3)	Payload (16)	Description
Veto	110	(12) BunchID + 0000	Signals BunchID to be vetoed
NO Veto	101	(12) BunchID + 0000	Signals BunchID to be not vetoed
GOLDEN	111	(12) BunchID + 0000	Identifies BunchID as golden
Reserved	100	(12) BunchID + 0000	-

Table 4.2.: **Description of the Veto commands.**

Every 2D detector has to implement a memory controller which reacts correctly on the Veto commands. Depending on the occurrence of Veto commands, images in the memory are not stored in temporal order. This means they have to be reordered by the DAQ system. Because this is a central part of the DSSC readout chain, the implementation of the Veto mechanism and the realization of the image-wise reordering in the DAQ will be described in section 7.3.4.

4.2 Trainbuilder

The Trainbuilder [45] represents the first DAQ device of the EuXFEL which is used for all three 2D detectors at the beamlines. The device receives and multiplexes the data streams from the **F**ront **E**nd **E**lectronics (FEEs) before the data is transferred to the PC Layer. One Trainbuilder device is connected to all 10 GBit/s data links from the FEEs of one detector. The received data is multiplexed and buffered by the a hierarchical system of FPGAs and switches in order to merge the event data of all FEEs of the same pulse train into one single link.

The device implements several FPGA[2] based ATCA[3] form factor boards which are equipped with a large number of high speed 10 GBit/s SFP+ links and high-performance cross point switches as shown in figure 4.3.

Always two 10G links are combined in a so called Front End Unit (FEU) which also includes one Virtex-5 FPGA 2GB DDR2 SDRAM memory and one additional fast 8 MB QDRII SRAM. Four FEUs are implemented on one Trainbuilder ATCA card and controlled by one master Virtex-5 FPGA. The FEU receives and buffers the image data. Furthermore, the pixel reordering algorithm is implemented, which sorts the pixel data in the order of the final image. Therefore, the fast QDRII SRAM is used, which allows to manipulate small chunks of data with high throughput.

The modular design of the FPGA boards allow utilization of the ATCA card either as the receiver or the transmitter. In order to transport the data from the receiver board to the transmitter board, a dedicated crosspoint switch is implemented, which operates at 3.125 GBit/s.

[2]Xilinx Virtex-5 FX100T
[3]Advanced Telecommunication Computing Architecture

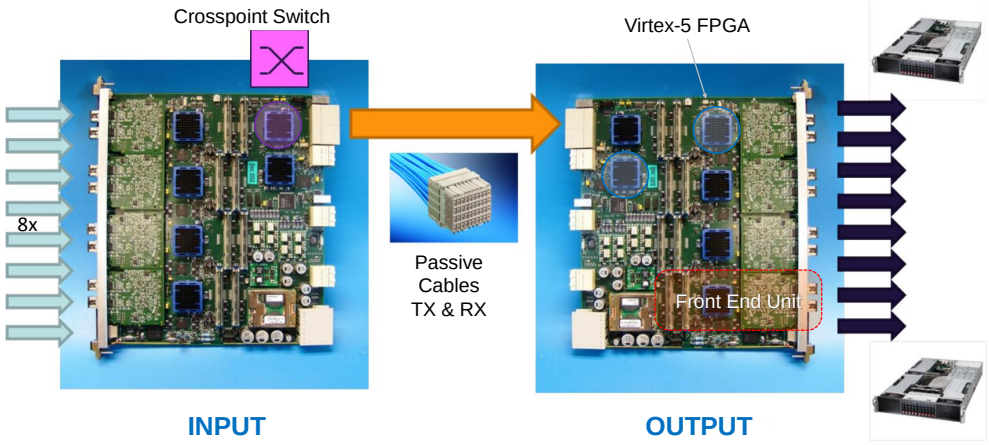


Figure 4.3.: **Two XFEL Trainbuilder ATCA boards [45].** Due to their modular architecture, an ATCA board can either be configured as an INPUT or an OUTPUT. One board implements five Virtex-5 FPGAs. One master FPGA and four FPGAs, each of which are assigned to one Front End Unit. The connection between two boards is realized via a cross point switch and passive cables.

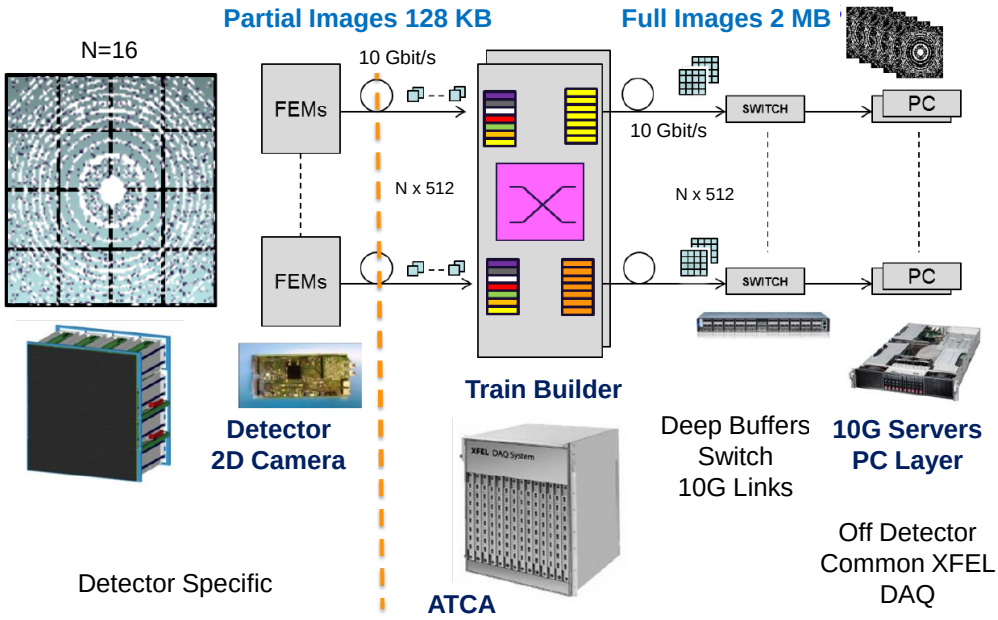


Figure 4.4.: **XFEL DAQ Environment [45].** One Trainbuilder collects all data from one megapixel detector via 16 10GE links. The Trainbuilder implements buffers and combines the data train wise, in order to transfer all data of one pulse train to the same receiver of the PC Layer.

Field	Size	Byte Num	Description
Frame ID	4 Byte	8184 - 8187	Up-counting number. Identical for all packets of the current train
Packet ID	3 Byte	8188 - 8190	Number of the current packet in the current train
Flags	1 Byte	8191	Bit 6 = EOF (End Of Train) set in last packet Bit 7 = SOF (Start Of Train) set in first packet

Table 4.3.: **The Train Transport Protocol (TTP).** It describes the last 8 bytes of each UDP packet as expected from a FEE module. The payload size is set to maximum 8192 including the TTP trailer bytes.

The FEEs of the 2D detectors are expected to store at least 512 image per train. Regarding the pixel word size of 2 Byte and the pulse train rate of 10 Hz, an average data rate of about 10 GByte/s is expected from a megapixel detector.

The performance of the Trainbuilder is sufficient to buffer and multiplex 512 image per pulse train. Since the DSSC detector can store up to 800 images per pulse train, it will not be possible to handle all acquired images of the detector using the presented Trainbuilder device.

4.2.1 Train Data Format

The combination of all images of one pulse train, including additional headers, trailer sections and meta data is defined as 'train data'. The Trainbuilder expects the train data in a given, standardized format, which is common for all three large 2D detectors. The specification describes the format of the Ethernet packets and also the composition of the different sections in the train data.

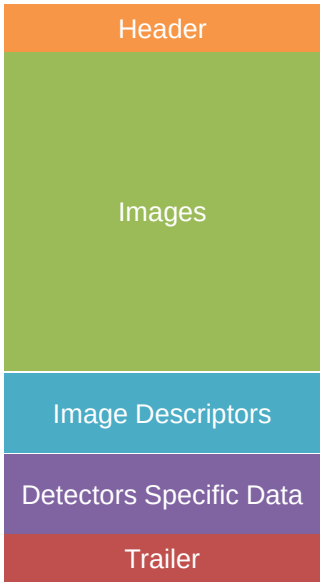


Figure 4.5.: Sections of the Train Data.

The data packets have to be transmitted as Jumbo UDP packets, which contain a payload of 8192 bytes. The required Ethernet [46], IPv4 [47] and UDP [48] headers add 42 bytes to the packet. The Train Trailer Protocol (TTP) defines, that the last 8 bytes of a UDP packet have to contain special meta information. The specified format of the single data words is always Big-Endian. This means the least significant byte is transferred first. The meta information of the TTP is shown in the following:

The specification for the Train Data Format (TDF) describes the train header and the expected meta data information, which is appended to the image data.

The train header bytes mainly contain the train ID, which allows the images to be assigned to a pulse train. Appended to the image data, the image descriptors contain the pulse IDs and cell IDs for each image. Due to the Veto mechanism, this information is required in order to assign the individual

images of the train data to the corresponding pulse in the pulse sequence and also to the SRAM cell in which the image has been stored.

This Image Descriptor block is followed by the Detector Specific data block. The content of this block is not further specified and can be filled with any meta information which may be important for the specific detector. In case of the DSSC detector, the Detector Specific data contains the sensor values, provided by the SIB and special trailer words which are described in section 7.1.1.

4.3 Karabo - The XFEL Control Software Framework

4.3.1 Introduction to SCADA Systems

In large physical experiments, like the EuXFEL, the global control and supervision framework faces enormous challenges. Thousands of devices provide real time data in several hundred thousands of I/O channels. Also large numbers of computers and whole computer clusters have to be connected and integrated into one large system. Software frameworks which provide solutions for these tasks are called SCADA systems (**S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition). Central services which have to be provided by a such a global control system are:

- Device configuration
- Device monitoring and process visualization
- Data management
- State awareness
- Automation
- Data logging / archiving
- Alarm handling

In general, a SCADA system unifies distant computers, devices and user interfaces in one large framework and provides access for large numbers of users. Very important for any SCADA system is its reliability and robustness. The realization of the distributed system requires platform independent communication between processes on the same and also on distant systems. This is typically implemented in so called Middleware applications which hides the complexity of the operating system and its hardware and infrastructure by providing a special communication protocol. Since the Middleware layer describes the central communication layer for all devices, certain central tasks like logging and data management functionality are in general included in available applications. A typical configuration for a SCADA system is visualized in figure 4.6.

During the last decades different commercial and open source frameworks became available and are utilized both industrial and in science. Prominent examples for open source SCADA systems used in physical experiments are EPICS [49], TANGO [50, 51] or DOOCS [52]. Since every system has its own advantages and no system implements all

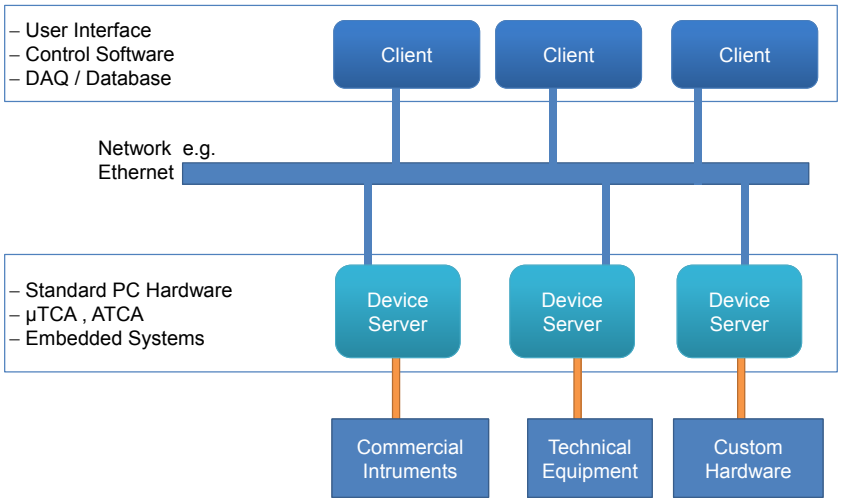


Figure 4.6.: **Typical setup of a SCADA system.** A central network connects device servers and client applications. The device server is a specialized application that provides distant access to the hardware components. The device server can run on any hardware, in large experiments often µTCA or ATCA systems are used, also embedded systems or standard PCs are possible.

features of the others, the choice of the SCADA system to be utilized in large experiments is always a highly discussed topic. Moreover, the comparison of the various systems is very difficult since they are not implementing the same abstraction layers. Some interesting publications which try to compare selected SCADA systems are for example [53] and [54].

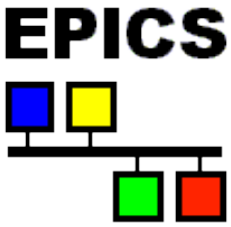


Figure 4.7.: EPICS Logo [49].

The central part of the EPICS (**E**xperimental **P**hysics and **I**ndustrial **C**ontrol **S**ystem) framework is the common high speed network protocol, called Channel Access (CA), which is specially designed for high bandwidth and soft real-time network communication. It can be used as an interface by almost any kind of application to connect to the global EPICS network. The EPICS environment brings a large library of software tools to integrate a variety of commercial and custom devices.

SCADA systems like EPICS and also TANGO (see below) are developed in large consortia and used by multiple large experiments like DESY, SLAC or KEK as examples for EPICS[49]. The enormous amount of invested development time in the order of more than hundreds man years has made them grow to reliable and well proven systems during the last decades.

Figure 4.8.: DOOCS Logo [52].

At DESY, a widely used system is **D**istributed **O**bject **O**riented **C**ontrol **S**ystem (DOOCS) [52]. It is developed since 1997 as a device based control system written in C++ with a focus on object oriented programming. Thus, hardware devices are represented in the framework as software devices, which are executed on device servers. One device object summarizes all methods and parameters which are required for its usage. A device server runs as a distinct process and can control an arbitrary number of devices. This allows independent execution of different tasks by different servers. The communication between devices is also controlled by the device servers which are connected by a network.

In a second hierarchy layer, hardware control devices can be controlled and summarized by middle-layer devices. In the middle-layer, much automation or safety functionality can be implemented independently of the client layer. The network communication is based on a server-client architecture model in which a central name server is used to manage the names of the available devices and servers for remote access. Properties and methods of distant devices can directly be accessed via Remote Procedure Calls.

User interaction is implemented by graphical clients executed on the respective workstation. To create graphical interfaces, DOOCS implements its own JAVA tool called JDDD[55].

Due to the successful implementation at DESY at HERA and the predecessor X-ray FEL FLASH, both the electron accelerator and the long undulators of the EuXFEL are also to be controlled with DOOCS [56].

Figure 4.9.: TANGO Logo [57].

The TANGO control software is a half commercial SCADA system, which is developed by a European consortium and has been first presented in 1999 [51]. Its first development and application started at ESRF in Grenoble, France. It was developed on top of the TACO architecture which first introduced the concepts of software devices that are running in device servers similar to DOOCS. Like in DOOCS, TANGO also provides Middle Layer devices for further abstraction.

TANGO is based on the **C**ommon **O**bject **R**equest **B**roker **A**rchitecture (CORBA), which is a specification for **application oriented** Middleware applications with the focus on object oriented programming. The specification enables platform independent communication between software written in different languages and running on different computers and platforms. In the CORBA specification, the network device communication is implemented by an object request broker (ORB) which provides the interface to remote procedure calls (RPC) on distant (c++/JAVA/Pathon) class objects. The TANGO network layer is also often called the *TANGO software bus* after the various hardware buses which allow to connect

large numbers of hardware devices to one communication network. Its modern architecture also implements mechanism for save execution in multiple threads. Further mentioned is always the high scalability of the TANGO control system and its light-weight device servers which can run also on embedded systems.

Until today, TANGO grew to a large consortium of 45 members and partners from both industry and research facilities all over Europe. Besides the ESRF, further large experiments use TANGO control. Just to name a few, there are several large synchrotrons e.g. Soleil in Paris, France, Elettra in Trieste, Italy, Solaris in Krakau, the first synchrotron in Poland and also Petra III at DESY in Hamburg, Germany, which are working with and participating in the TANGO control system. Despite the existing well proven SCADA systems, at XFEL the decision was made to develop an own software framework for beamline control and data management which will be presented in the following.

4.3.2 The Karabo Software Framework

For the control of the photon beamlines at the EuXFEL as well as the experiments and the associated detectors, XFEL develops an own new SCADA system. Its name is **Karabo**, which is the Sotho[4] word for 'I am the solution'. The first version of Karabo has been released in 2013[58],[59].

Detailed information about the design concepts of the Karabo framework are presented in the reference presentation [60], which is constantly updated. Because the implementation of the DSSC Karabo devices is an important section in this thesis basic features and possibilities of the Karabo framework will be presented in this section.

For operation of the large 2D detectors a variety of different systems have to be controlled: power supplies, high voltages, vacuum systems, and of course the detector control and the data acquisition. The data acquisition is one of the biggest challenges for the Karabo framework because of the large amounts of data which are produced by each 2D Megapixel detector of up to 16 GByte/s. Besides the data storage also data correction, calibration and data analysis will be implemented in Karabo. First results of near to real time data handling of detector data in Karabo have been already presented in [61].

The network communication in Karabo is based on a central message broker similar to the TANGO control system. The architecture is shown in 4.10. Device communication is implemented in Karabo by the Open Message Queue (OpenMQ) [62], which is a **message oriented** Middleware platform. OpenMQ is an implementation of the **J**ava **M**essage **S**ervice (JMS) [63] specification and runs as integral part of the Glassfish [64] application server. By using enterprise ready systems for these critical parts of the Karabo framework, it benefits from a well proved system.

The Hash Object

The central data structure in Karabo is the Hash object. A Hash is a fully recursive container class which implements key-value association for data access, where a key is a unique string and value can be of any type. The Hash supports hierarchical data structuring and preserves insertion order. For fast data access it is optimized for random key-based look-ups.

[4] Language from South Afrika, Lesotho and Botswana

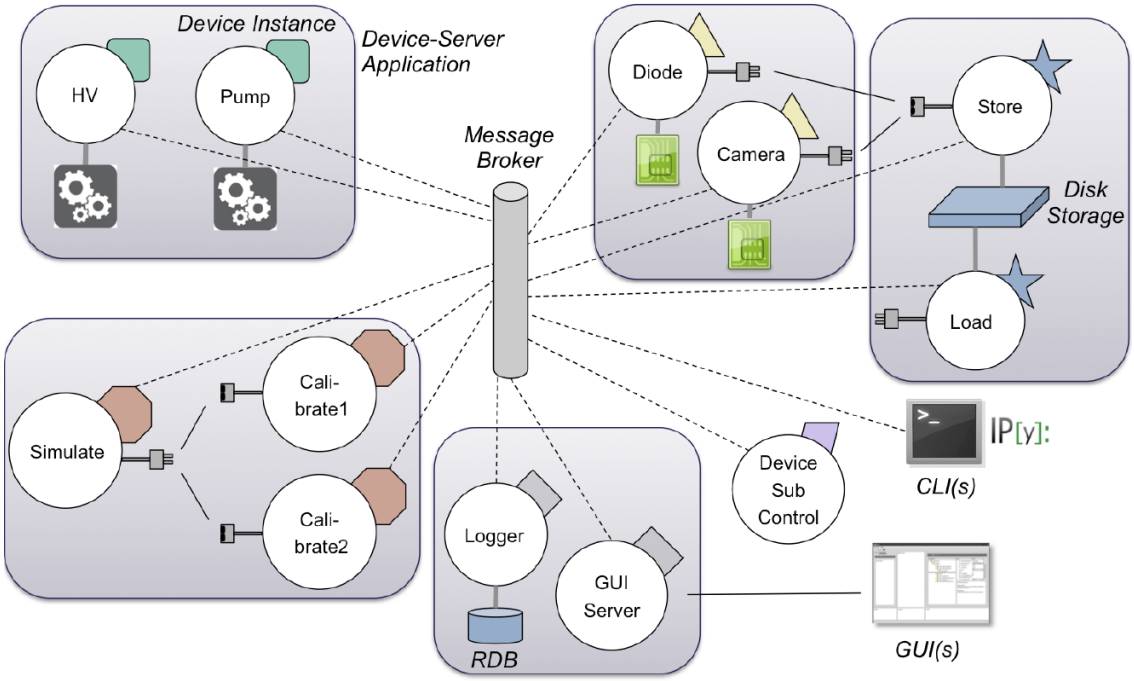


Figure 4.10.: **The Karabo architecture [58].** All devices are connected (dashed lines) to the central message broker, which supervises the message based device-to-device communication. The broker provides automated logging features and also alarm handling. The device server applications (indicated by rectangles) manage the access to all installed devices on the system on which the device server is executed. Central tasks of the framework functionality are implemented by special device servers like the GUI server or the data logger server. High-bandwidth point-to-point (p2p) connections (solid lines) can be defined to efficiently transfer large amounts of data (see text) between devices without interfering the central broker. The obligatory description of the device properties allows the Karabo GUI to automatically generate a user accessible representation of the device. This enables easy and quick integration of new devices into the graphical interfaces of the Karabo framework. The Karabo GUI and also the IPython, an interactive command line interface for python, provide full access to devices which are installed in the framework. This also includes adding of new devices, interactively generating of graphical user interfaces and controlling and monitoring of the devices.

Efficient Data Transfer in Karabo (p2p channels)

In contrary to the application oriented Middleware of TANGO, in which distant procedures are directly called and executed, the message oriented Middleware (MOM) architecture allows for a much looser coupling of the single components. This means by implementing message queues, clients can automatically operate at their own pace while the messaging infrastructure absorbs all messages at production rate in order to guarantee zero message loss. Thereby, communicating clients can either be executed on same machine in different threads or on different machines in the same network. The OpenMQ implements both point-to-point (p2p) connections for high bandwidth data transfer and publish-subscribe messaging for general device access. All critical aspects of producer-consumer problems, like race conditions, have already been solved and safely implemented in the OpenMQ standard. The standard also supports topics, which can be assigned to device servers and consequently all devices which are managed by this server. All devices within a topic can be associated with a particular access level, which can be used, for example, to allow only a certain group of devices to change parameters of devices in this topic.

While in general all device communication like parameter access and data logging is realized by the publish subscribe mechanism and all communication messages pass the central broker, the direct p2p messaging does not interfere the broker. In general a p2p connection sends data via TCP packets. If two devices are executed on the same system data is transferred directly between the processes. The p2p connections can dynamically be connected and disconnected during run time. Multiple senders and receivers are supported per channel and several data handling policies can be selected. So it is always possible to decide whether data should be copied or received in shared mode. In shared mode there are two configuration modes:

- **Round robin**: Equally distribute the data. If the next input input line is not ready, the writing output channel will be blocked until this input channel becomes free.
- **Load-balanced**: Data is sent always to the next available input channel. If the data can not directly be handled one of the available policies (see below) can be defined. Very efficient if data recipients are expected to have different processing times.

In any case, if the counter part of an input or an output is currently not available there has to be defined which will happen to the data. For each output, the following options are possible:

- **throw** an exception, which must be handled in the code.
- **queue** the data. Data will be appended to a queue until all memory is filled.
- **wait** until an input becomes available. Effectively a blocking write at the output.
- **drop** the data at the output and proceed without blocking.

The queue option can be very efficient when working with multiple senders and receivers but it can also gather significant amounts of memory if the amount of queued packets becomes too large. Consequently, it should be used carefully.

The data, which is transferred in a p2p channel, is always a Karabo Hash structure. Its format has to be declared at the output as well as at the input and the sender has to ensure that the transferred data fits to the declared structure. The Karabo framework can not check if the format of the transferred data is correct. Therefore, the developer has to make sure that the implemented data formats fit together.

The input and output channels which are provided by Karabo mean easy integration of complex concurrency functionality without facing the difficulties of race conditions and efficient data distribution. Karabo already provides an easy-to-use solution for this problem.

Additionally, Karabo supports event driven inter-process and device-to-device communication. Therefore, an own event loop has been implemented which is also known from the signal-slot model of the Qt library [65, 66].

Karabo Devices

A Karabo device is described by its properties and slots. Properties can be of various types like strings, integers, booleans or vectors and are always accessed by their 'key' string. Also various features like a description, alarm limits etc. can be assigned to each property. The slots define the access to the device methods. The description of the device properties and slots is mandatory and has to be added to the device Schema. Like in a Karabo Hash object, the device Schema can be nested and hierarchically structured in order to group things which belong together. The Schema defines the interface of the device to the outside within the Karabo framework. While all properties, similar to member variables, are accessible within all methods of a device, they also provide a self-description of the device for the integration into the Karabo framework. For example, other devices can access the properties described in the device schema.

A certain category of devices has to be mentioned at this point: the Middle Layer devices. These devices are specialized to access properties and slots of other devices. For simplified development, a special Python Middle Layer API has been introduced to describe these Karabo devices. This API reduces coding effort and provides convenient device control. Middle Layer devices are mainly used for simplified device usage and automated device control.

Karabo GUI

The GUI is written in C++ using Qt widgets. It defines the central control interface to the Karabo framework. By reading the given device Schema, the Karabo GUI generates automatically user accessible fields, check boxes and buttons. This automatic interface generation provides one of the big advantages of the Karabo framework because this allows instant testing, debugging and user access to a new device without the need of additional programming effort. The Karabo GUI also allows easy composition of more convenient graphical user interfaces by just drag and drop of device properties and buttons into the central scene field. More information about the Karabo GUI and its usage can also be found in the Karabo Documentation as well as in various tutorials.

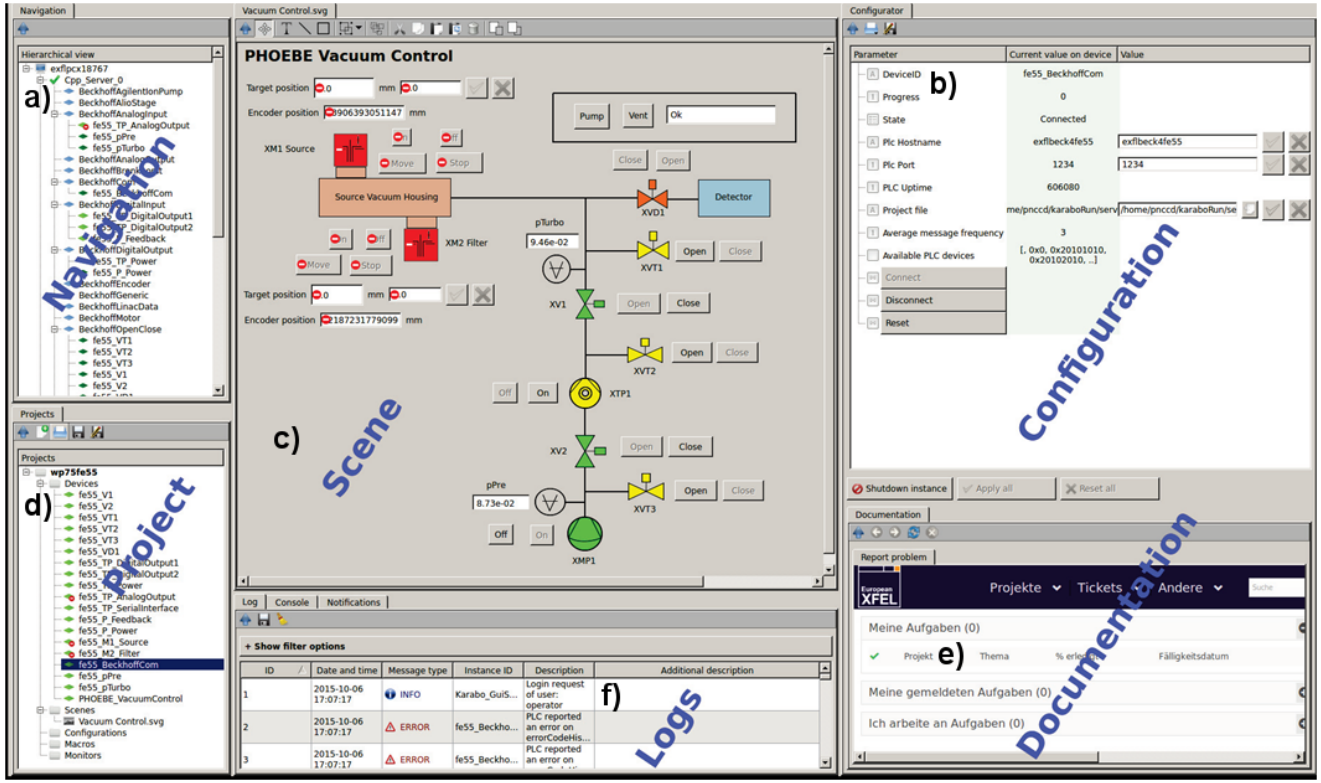


Figure 4.11.: **Screenshot of the GUI [67].** (a) The live Navigation, with computers, servers, device classes and devices. (b) The list of properties and slots of a device, automatically generated from its Schema. (c) A Scene, showing several device properties. (d) The Project, which can be used for grouping of devices and scenes. (e) The message Logs. (f) Access to online documentation.

Karabo Summary

The comparison of different control systems is difficult since each system has its own advantages and no system implements all the features of the others. This is why there exists tools that blur the boarders between the single systems. Special TANGO to EPICS and TANGO to DOOCS converters are available which can be used to make TANGO or EPICS devices available in the DOOCS clients and vice versa. At the end, with Karabo, a system has been created which combines most advantages of the other systems in one homogeneous system. Karabo combines the software framework, the user interface and the programming interface in a single application.

The standardized framework to control everything provides a certain advantage in terms of staff training and integration of different components.

Preliminary studies [61] show that in theory Karabo can provide sufficient performance for complex calibration pipelines which offer near-to-real time image data calibration and scientific data analysis.

In the authors point of view, the mandatory description of the device properties and methods which is simultaneously used to generate a user interface and a programming interface defines one of the big advantages of Karabo. This feature allows instant generation of a device in the Karabo Framework without the additional effort to write a user interface.

Software development in a large framework is always more complicated than in a small standalone application. But with some experiences in programming languages, Karabo can easily be learned and the Framework and the GUI bring a lot of useful tools and functionality like for example visualization, efficient distributed computing, data handling and logging. The experiences which have been gained with the framework are described in the final conclusion.

5

Presentation of the DSSC Detector

The **D**EPFTET **S**ensor with **S**ignal **C**ompression (DSSC) Detector [68, 69, 70, 71] is the most sensitive detector among the large 2D X-ray detectors at EuXFEL. In this part the main components of the detector and their interconnections will be described in order to give an overview about the system and its complexity. Major parts of this thesis deal with the implementation of the readout chain and the control software for the detector system. The hardware components build the platform for the described implementations. Consequently, a fundamental understanding of the working principle and the interplay of the different detector components is required to understand the complexity of the presented work.

In the beginning of this part, the properties of the proposed detector and the two different sensor types of the DSSC detector will be introduced. Also the principles of the two sensors types are described and some basic features and optimizations are presented, which have been introduced to decrease the noise and improve the dynamic range. In the following, the concept of the readout ASIC is shown. This section also includes a listing of the different test ASICs, which have been developed during the last years. Also the different test environments are introduced, which have been used to develop firmware and software while the real detector components have not been available.

At the end of this part the different components of the detector are presented in more detail including the boards which implement the high performance readout chain.

5.1 Properties of the Proposed Detector

The DSSC detector is a megapixel X-ray detector based on **DEP**leted **F**ield **E**ffect **T**ransistors (DEPFET) with a non linear characteristic. This type of silicon detector implements the first signal amplification intrinsically which results in a very small input capacity and a high signal to noise ratio. The strength of the DSSC DEPFET implementation is the speed of the detector combined with its high signal to noise ratio. In combination with sensitive readout electronics the DSSC Detector provides single photon detection down to photon energies of 0.25 keV [69]. At the same time, a large dynamic range of ten thousands of photons will be possible due to the non linear characteristic of the DEFPET sensors. Further values for the DEPFET proposal can be found in table 5.1.

The original proposal of the DSSC detector has foreseen only DEPFET sensors. But after

Energy Range	$0.25\ keV \leq E \leq 20\ keV$
Pixel number	1024×1024
Sensor Pixel Shape	Hexagonal
Sensor Pixel Pitch	$\approx 204 \times 236\ \mu m^2$
Input Capacitance	$100 fF$
Dynamic range photons/pulse/pixel	> 10000 photons for $E \leq 1keV$ > 4000 photons for $E = 0.5\ keV$
Resolution	Single photon detection @ $0.5\ keV$
Frame rate	$0.9 - -4.5\ MHz$
Stored frames per pulse train	800
Mean power consumption	$\approx 400\ W$ in vacuum
Operating temperature	$-20^{\circ}C$ optimum, room temp. possible
Average data rate	134GBit/s

Table 5.1.: **Design values of the DSSC detector [69].**

very promising results from the first test structures produced, there occurred problems with the production of the final large-format sensor tiles. It became clear that there would not be sufficient large-format sensors to build a megapixel camera. Additional political and personal restructuring processes at the sensor manufacturer lead to the decision to investigate a simpler detector approach using passive mini silicon drift detectors (MSDD), which are presented in section 5.3.1. Due to its passive nature, the MSDD requires a different readout front-end in the ASIC. The induced delays forced the consortium to take a risky step and include an early version of the MSDD front-end into the first large scale readout ASIC F1.

The MSDD front-end in F1 worked in principle but showed some critical limitations resulting in reduced performance and larger noise values than expected. A second version of the readout ASIC F2 has been designed with an improved front-end to solve the major issues. The F2 ASIC has been received in August 2017 and first tests show promising results. A brief listing of the most important test ASICs and large format engineering runs is given in 5.10.1. For the data path and the control software, the two different sensor approaches play only a minor role, since only the input stages are different and all further electronics of the readout chain remain identical for both sensor types.

5.2 Design Concept of the Detector System

The biggest challenge of the DSSC detector is the very low noise performance to reach single photon resolution in the lower energy range from 0.25 - 6 keV in combination with a large dynamic range and the fast readout speed of up to 4.5 MHz on a large scale 1M pixel matrix. In order to reach this goal, the original proposal for the DSSC detector implements a unique DEPFET detector technology which is newly designed for this project. The principle of the DEPFET detector is described in section 5.3.2.

The concept of the DSSC detector is depicted in figure 5.1. The DSSC detector is composed

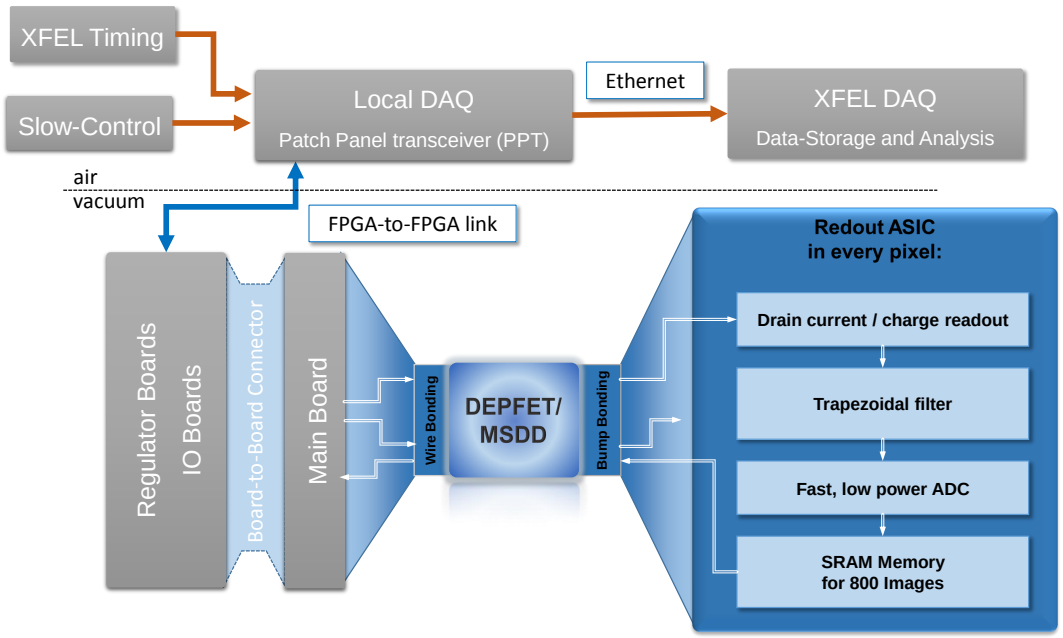


Figure 5.1.: **Block diagram of the DSSC detector [68].** For each sensor pixel, one readout channel is implemented in the ASIC, which is bump-bonded on the backside of the sensor. The sensor routes the ASIC signals via wire bonding to the Mainboard. Several PCBs implement the infrastructure for readout, control and power distribution.

of several building blocks. The three big blocks are: the focal plane, the readout boards and the power supply infrastructure. The focal plane comprises the sensitive X-ray sensor, the readout ASIC and the Mainboard. The readout chain is implemented by two FPGA based readout boards: the IO Board (IOB) and the Patch Panel Transceiver (PPT). These two boards will be described in detail in 5.6. The power supply is realized by PCBs, which provide large capacitor banks and fast switching voltage regulators. For interconnections between the different boards, several connection boards are implemented.

5.3 Sensor Technology

In this section, the two X-ray sensors of the DSSC detector are presented. Both types are based on a large volume of depleted silicon. The sensor characteristics are in both cases optimized by advanced doping schemes in order to realize low noise performance and high speed readout cycles of 220 ns duration.

Silicon is a very attractive material for X-ray radiation detection because of several reasons:

- The band gap of 1.12 eV enables easy generation of elevton-hole pairs (Only 3.6 eV are required in average).
- High charge carrier mobility ($\mu_n = 1450\ cm^2/V_s, \mu_p = 450\ cm^2/V_s$ at room temp.) lead to fast charge collection times, and thus allows high readout speed.

- The high material density of 2.33 g/cm^3 allows large energy loss of charged particles per moved distance.

Furthermore, the possibility to induce specially shaped electrical fields by precise doping allows sophisticated approaches for different types of sensors. Doping means an intentional change of the concentration of free charge carriers by introducing materials with missing (p-Type) or extra (n-Type) valence electrons from Group III resp. Group V. The strength of doping is indicated by a '+' for strong and '-' for weak doping (n^+, n^-).

The sensor size and pixel pitch for both types of sensor is identical. A system sensor tile implements 128×256 hexagonal pixels of size 204×236 μm^2.

The basic principles of silicon detectors can be found in various literature e.g. [72]. A very nice summary can also be found in the PhD thesis of Florian Erdinger [73].

5.3.1 Mini Silicon Drift Diodes

The mini silicon drift diode is an improved version of the simple pixelated n^+-in-n diode detector. For better understanding, the principle of the simple detector is roughly sketched.

In a pixelated n^+-in-n diode detector, the principle of a reverse biased diode is applied where free charge carriers, generated by incident photons, are separated by a strong electrical field which is generated by a high external potential. The high potential causes a large depleted region which is required to separate the generated charge carriers.

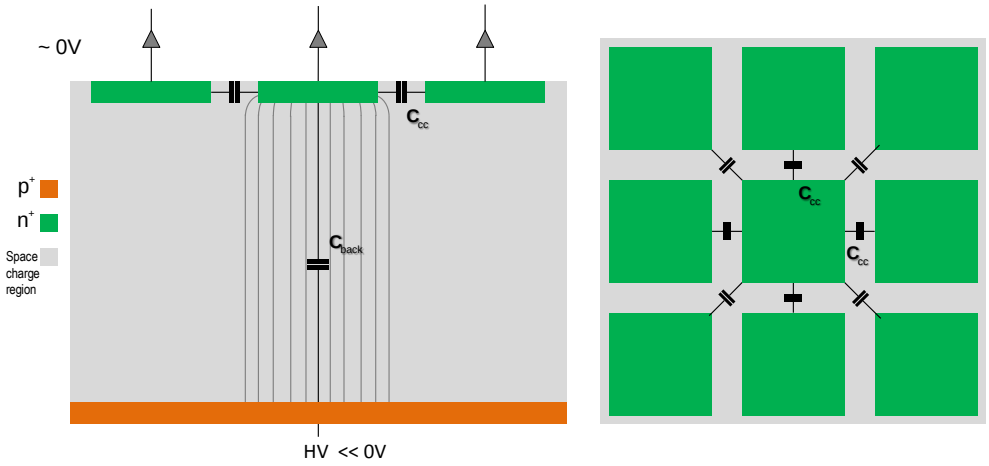


Figure 5.2.: **Schema of a pixelated n^+-in-n diode array [73].** The input capacitance mainly couples against the neighboring pixels. For good charge collection capabilities the anode needs to fill large area of the pixel.

In order to resolve the collected charge locally in a pixelated sensor, the n^+-doped anodes are separated which creates the readout anodes for each pixels. For good charge collection properties a homogeneous electrical field between the n^+ pixel anode and the p^+ backside is needed. Therefore, the pixel anodes have to cover a large area of the pixel. A main factor for the signal to noise characteristics of such a sensor is given by the capacitance of the charge collecting anode. For a charge collecting sensor the noise is described by the

equivalent noise charge (ENC), which can be calculated by [74]:

$$ENC = \frac{a^2}{\tau} C_{in} F_1 + 2\pi a_f C_{in}^2 F_2 + \tau b^2 F_3 \tag{5.1}$$

With the total input node capacitance C_{in} , signal processing dependent parameters F_1, F_2, F_3 , power spectral densities a, a_f, b and a time constant τ.

Due to the fast readout speed, the DSSC system is dominated by series white noise (first term in equation) [75]. Therefore, the noise in the DSSC system scales with the input capacitance C_{in} . The input capacitance of such a device can be split into two major parts as shown in figure 5.2. The anode shows a decoupled capacitance against the backside and a coupling capacitance to the neighboring pixels. The later rises with larger size of the anode and induces unwanted crosstalk from the neighboring pixels. Consequently, the design of such a device has to consider the maximum parasitic capacitance which can be tolerated, with respect to the charge collection abilities and vice versa.

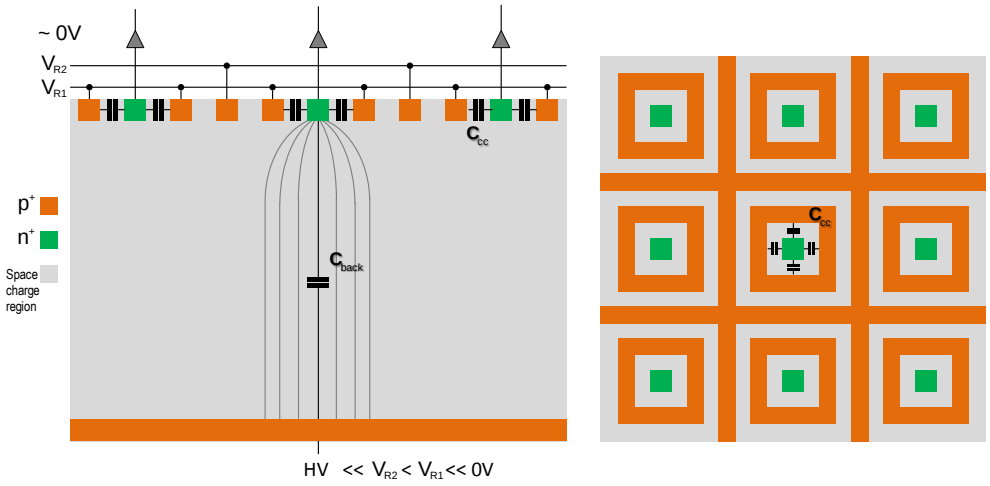


Figure 5.3.: **Schema of a Mini Silicon Drift Diode [73].** The p^+-doped drift rings induce a good shielding from the neighboring pixels. The field lines are bended towards the small anode due to the applied ring voltages V_{R1} and V_{R2}.

The silicon drift diode proposes an approach to solve this limiting dependency. By introducing several p^+-doped drift rings around the charge collecting anode, as shown in figure 5.3, the pixels are shielded against each other and the coupling capacitance basically vanishes. By connecting the rings to different potentials it is even possible to shape the electrical field below the small anode in a way to direct the electrons towards the center, and thus improve the charge collecting properties. Furthermore, this allows to decrease the size of the anode which significantly decreases the input capacitance of the detector. The p^+-doped ring regions also let the space charge region grow from the top to bottom which enables to deplete the sensor with lower potentials. This principle is called sidewards depletion [76].

Besides the intrinsic capacitance of the pixel anode, the connection of the sensor to a readout ASIC, e.g. by bump bonding, adds additional capacitance to the charge collecting

input node. In general, the contribution of the interconnection and the bonding dominates the input capacitance of an MSDD sensor pixel.

The improved performance of the silicon drift diode is payed with a higher effort during production due to the second doping step for the drift rings which results in higher costs. A matter of costs is also the choice of the bonding material which severely affects the input capacitance. But smaller balls and associated flip-chip processes are more risky and costly. The consortium has therefore decided to stick to standard C4 (Controlled Collapse Chip Connection), which was foreseen for the DEPFET. For the DEPFET, the input capacitance does not matter because a current signal is supplied. The choice of the alternative detector has been driven by the possibility to reuse much of the existing readout structures of the DEPFET front-end in the readout ASIC.

The detector type, which has been selected for the DSSC detector, is a small drift diode with only two rings around the anode. In principle, the MSDD sensor was developed from the DEPFET by replacing the transistor with a central diode.

5.3.2 DEPFET Sensor

The DEPFET device is basically a field effect transistor with a second *internal gate* located directly below the external gate. For improved charge collection, the central DEPFET is surrounded by several drift rings, in the same fashion as for the MSDD. The idea for this device has been invented by J. Kemmer and G.Lutz and has been proposed in [77]. In figure 5.4, a schematic drawing of a DEPFET pixel is depicted.

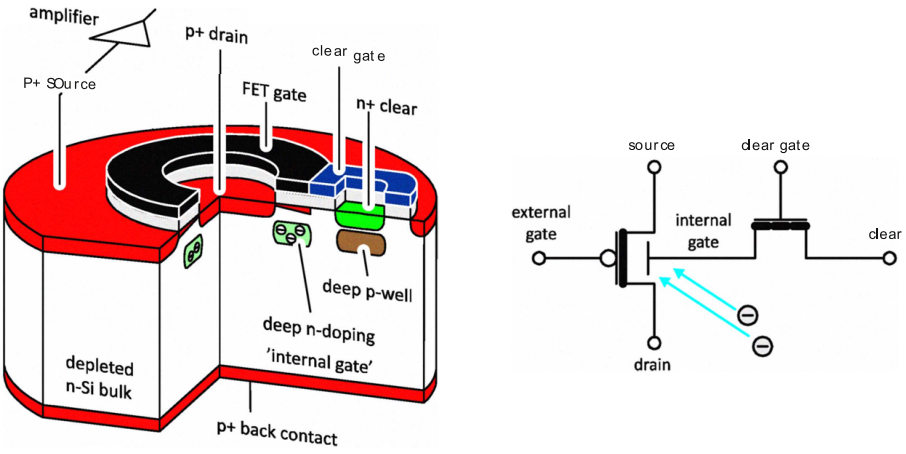


Figure 5.4.: **Schema of a DEPFET pixel and the according schematic drawing [78].** Below the outer gate, a deep n-doping implements a potential minimum for electrons that attracts negative charge carriers. Due to the changed potential, positive charge carriers are attracted below the external gate forming a conducting channel. In this way the deep n-doping works like an internal gate. The clear contact is used to remove the trapped charge carriers from the internal gate.

In the center of the pixel, a p-channel field effect transistor is located, which is placed on a high resistivity n-type bulk. The p-doped areas of the drift rings cause a growing of the space charge region from the top of the substrate due to the side depletion mechanism [76] like in the MSDD type. This mechanism causes the separation of the charge carriers which

are generated by an incoming photon. While the positive holes are drifting mainly to the negative backside contact, the electrons are guided by the drift ring potentials and attracted by the potential minimum (for electrons) which is shaped by a deep n-doping, directly below the external gate. The additional electrons in the region of the deep n-doping lead to a higher conductivity of the transistor. This causes a higher current for the applied source gate voltage. In this way the deep n-doping works like an internal gate.

To put the transistor in a proper operation point, a bias current has to be applied. The resulting output current is the sum of the applied bias current and the signal current. Consequently, the readout electronics has to subtract the bias to retrieve the final signal.

The sensitivity of the DEPFET current to signal charges $g_q = \Delta I_D / \Delta Q_{sig}$ is called transconductance. After [78] it can be computed by

$$g_q = \frac{\Delta I_D}{\Delta Q_{sig}} = \sqrt{\frac{2\mu_p \, I_D}{W L^3 \, C_{ox}}} \tag{5.2}$$

and is depending on the transistor dimensions W, L , the hole mobility μ_p and C_{ox} , the oxide capacitance between gate and channel per unit area.

The DEPFET converts the collected signal charge into a current. The implemented DEPFET has a typical amplification factor (in the linear region) of about $600\,pA/e$ [69]. With the energy to create an electron-hole-pair in silicon of 3.6 eV and the photon energy one can compute the number of incident photons. Consequently, due to the internal amplification, the DEPFET can be considered as an active pixel sensor.

The major advantage of the device is the fact, that the charge signal does not have to be transferred to the ASIC. It is collected in the internal gate, which has a very low capacitance and is very well shielded against disturbances. The demands on the ASIC are much less challenging, when a signal current is to be processed.

Furthermore, the internal gate is a high impedance node. This means, the collected charge remains trapped after the readout process which practically allows to read the signal multiple times. This can be used to lower the statistical variations of a measurement.

In order to remove the collected charge from the internal gate, an n^+ implant, the *clear* contact, is added next to the transistor. By applying a sufficient high potential, the collected charge can be removed from the internal gate. Since the n^+ implant can also drain signal charge from the depleted bulk, an additional deep p-well is added in order to prevent this effect. However, this additional potential makes the clear process even more difficult. In order to reset the sensor anyway, the *clear-gate* contact is added. A positive voltage pulse on the *clear-gate* contact induces an n-channel within the p-well which enables the drain of the signal charge. Thereby, the *clear-gate* pulse must enclose the actual *clear* pulse.

In order to meet the requirements of single photon detection capability at 1 keV photons in combination with large dynamic range of 10^4 photons, the common DEPFET design has to be adapted. The DEPFET sensor, which is used by the DSSC detector, implements a specially shaped internal gate which leads to a nonlinear signal response of the sensor resulting in the requested detector performance. This is a novel approach and has been tailored specifically for DSSC system.

As shown in figure 5.5, the internal gate is separated in several overflow regions of different sizes. The compression mechanism is based on the idea, that charge, which is

collected in the internal gate, does not contribute equally to the electrical field which opens the channel of the transistor. The electrons, which fill the gate at the beginning (small signals), generate a strong response while electrons, which are collected in overflow region 1 to 3 (strong signals), have less influence on the signal current.

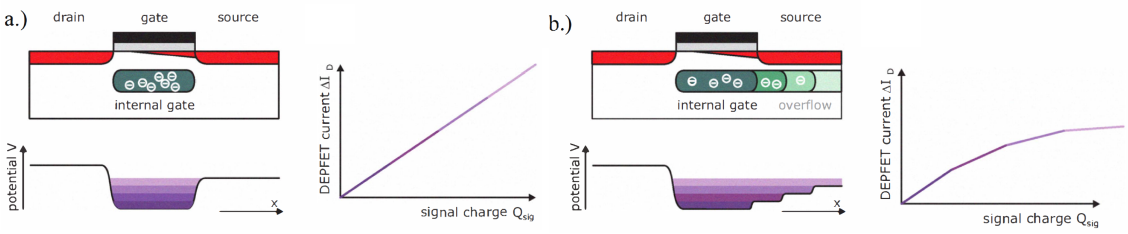


Figure 5.5.: **DEPFET with signal compression[78].** a) DEPFET with linear response. b) DEPFET with compressed characteristic.

The internal gate of the DEPFET provides a very low capacitance of typically below 50 fF. In combination with the internal amplification, the DEPFET sensor provides best noise performance for singe photon detection capabilities. Furthermore, due to the buried gate in the bulk, the coupling of the input nodes of neighboring pixels vanishes.

However, the good performance of the DEPFET detector has to be paid with high production costs and manufacturing time because of its complex design. Another limiting factor is its high power consumption. One single DEPFET detector pixel consumes about $100\,\mu A$ at about $5\,V$. Further power consumption for the readout circuit has to be added which adds up in total to a power dissipation of the DEPFET megapixel detector to about $1-2\,kW$ peak power. Since this high power cannot be provided (and cooled) during continuous operation, most of the signal processing circuits are switched off during the gap between the pulse trains, and is only activated for about $600\,\mu s$ during the data acquisition phase. This decreases the average power consumption to a value of about $450\,W$.

5.3.3 Concepts of the Sensor Readout Circuits

The current readout front-end of the readout ASIC is described in section 5.4.1. In this part, the main differences of the DEPFET front-end and the adapted MSDD charge readout circuit in the F1 ASIC is described.

In the right part of figure 5.6, the topology of the MSDD readout variant is shown. The circuit is not typical for charge sensitive detectors since there is no virtual ground (closed loop) on the input. The charge signal is directly converted into a voltage signal by the effective total input capacity C_{in} , which is mainly dominated by the interconnection and the bump. In the readout ASIC, mainly the gate capacitance contributes to the input capacitance. Using a high gain transistor the voltage signal is translated into a current signal which can be measured by the existing current readout front-end.

Since the DEPFET generates the current signal internally, the capacitance of the interconnection do not contribute to the system noise. Consequently, in the DEPFET design, no further optimizations which reduce the interconnection capacitance have been introduced. In order to not change the proven solder ball technology and assembly processes, the

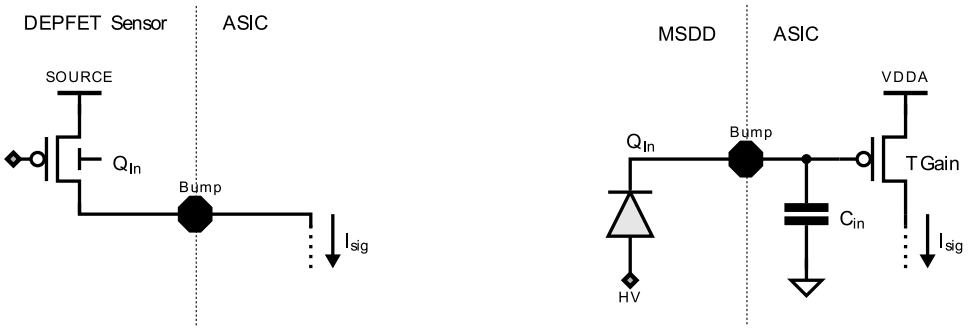


Figure 5.6.: **Schematic drawing of the sensor readout concepts [73].** The front-end in the readout ASIC is designed to evaluate the current signal from the DEPFET (left). For the adopted MSDD of the F1 ASIC (right), the charge signal is translated to a current signal using a high gain transistor $TGain$.

identical interconnect technology[1] has been used for the MSDD sensors. Consequently, the total input capacitance is expected to be in the order of 400 fF.

In order to compare the noise performance of the two readout variants one can in principle define a transconductance g_q of the MSDD readout, as it is expressed for the DEPFET (see equation 5.2). It can be described by $g_q = g_{m,TGain} \times Q_{in}/C_{in}$. To compensate for the larger C_{in} , the g_m of $TGain$ can be increased.

Besides the large input capacitance, the MSDD variant has another weakness. The current I_{sig} is very sensitive on variations on the supply voltage VDDA. Since this voltage is common for each pixel and supplies all analog pixel circuits, it fluctuates. These fluctuations couple into the current signal with the transconductance of gm. Additionally, the MSDD input node is very sensitive to crosstalk from the dense environment in the readout pixel. This effect is also dependent on C_{in} and further amplified by the large value $g_{m,TGain}$.

Because of the buried gate in the sensor bulk, the DEPFET readout is strongly shielded from crosstalk.

The coupling of the neighboring pixels in the MSDD front-end causes higher noise values and weaker performance of the F1 ASIC. The described inter-pixel coupling causes also a certain dependency of the current consumption of the neighboring pixels on the readout process which has also to be taken into account by some of the trimming algorithms presented in the software part in section 10.2.

5.4 The Readout ASIC

A high sensitive sensor requires an appropriate readout system in order to guarantee best signal to noise properties in the final images. Therefore, the DSSC consortium is developing a special readout ASIC which processes the signal current from the DEPFET detector pixels. As previously described, an adapted version was integrated to enable the read out of an

[1] IBM C4 - Controlled Collapse Chip Connection.

MSDD sensor. In this section the readout concept of the ASIC will be presented.

5.4.1 Signal Processing Concept

The ASIC was designed to provide best possible noise characteristics and fast signal processing during the available time frame, which is given by the pulse scheme of the EuXFEL as described in section 3.3.

2700 pulses with a distance of 220 ns are delivered, followed by a 99.4 ms pause. The goal is to store as much of the pulses as possible. Because an immediate read out cannot be realized, the taken images have to be stored and read out during the pause between the pulse trains. The DSSC readout concept realizes in-pixel digitization combined with a large in-pixel SRAM memory to store up to 800 events which will be the largest number for all of the three megapixel detectors at the EuXFEL.

The readout ASIC is custom made, developed by the DSSC consortium. The ASIC scheme can be divided into four major building blocks like shown in figure 5.7. The photon sensor and the respective front-end, the trapezoidal flip capacitor filter (FCF) [79], the ADC[42] and the SRAM memory.

The different blocks have been designed by different groups of the DSSC consortium, while the global control and integration of the different blocks into the final pixel has been executed by Florian Erdinger during his PhD thesis.

The DEPFET sensors are connected to the readout ASIC in drain readout mode which means that the bias current of the sensor has to be drained by the readout ASIC. For this purpose, the input branch implements a programmable current sink and a cascode transistor (*TCasc*). The cascode improves the transient behavior and output resistance of the DEPFET. The subsequent filter circuit provide a virtual ground from which the current sink can cancel the bias current. The bias current is fixed by the source-gate voltage of the amplifying transistor and can vary from pixel to pixel. An automated bias subtraction circuit is implemented to compensate for pixel-to-pixel variations. However, the current subtraction works not perfectly and a residual current is always left which has to be compensated by the filter.

The charge readout front-end to evaluate the signal from the MSDD is implemented in the F1 ASIC in a way which fits seamlessly with the existing processing chain for the current readout. The implemented idea is to use a PMOS transistor in the ASIC which basically replaces the DEPFET. A non-linear response providing a signal compression is implemented by the so called Triode Compression [80]. The current subtraction can identically be used during MSDD operation.

In both cases, the residual current is canceled from the final signal by the FCF.

The FCF implements a trapezoidal weighting function which is realized by two integration phases. The first phase integrates the residual baseline signal on the capacitance right before the pulse arrives. During the flattop phase, the capacitor terminals are flipped by means of several switches which leads to an inverse integration during the second integration phase. If a photon hits the sensor during the flattop, a signal current is added to the baseline current. Due to the two integration phases with inverted signs, the baseline is subtracted

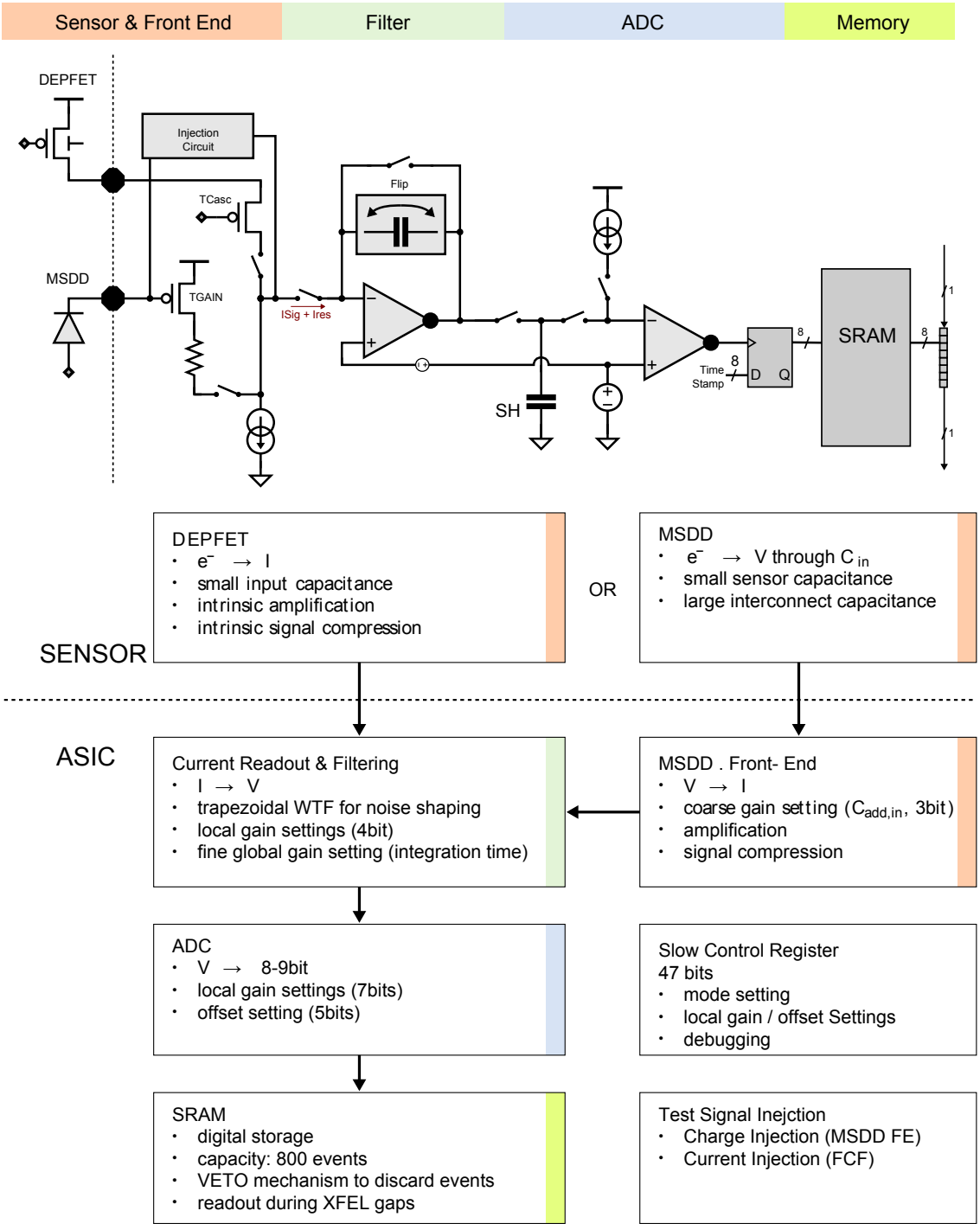


Figure 5.7.: **Concept of the DSSC readout pixel [73].** The readout is divided into several building blocks. The order of the readout is indicated by the arrows.

from the signal during the filtering process. The integration length can be varied with a granularity of $\approx 1.4ns$ and adjusted according to the image repetition rate. The double integration decreases the effect of pixel to pixel variations and increases the signal to noise ratio. Further information can be found in [79].

Besides the analogue shaping of the input signal, each pixel also includes an internal fast low power 9-bit Wilkinson-type ADC [42] combined with an internal SRAM.

The working principle of this type of ADC is as follows: the integrated voltage signal from the amplifier is stored on the sample and hold capacitor (SH). At a defined time point, the current source in the ADC is switched on and starts to charge the capacitor. Simultaneously, a fast 1400 MHz Gray-code counter is started. At the moment, when the voltage of the charging capacitor reaches the level of a reference voltage, the comparator in the ADC triggers the latching of the 9 bit time-stamp value from the Gray-code counter[2]. The Gray-code counters (one per pixel column) are located in the periphery and distribute the time stamps via coplanar waveguides over the whole pixel column. Detailed information about performance and further characteristics about the DSSC readout ASIC can be found in [81] and [82].

5.5 Detector Layout

The focal plane and the whole DSSC detector can be seen as a hierarchical composition in several levels. The top level is split into four quadrants which can be moved independently to form a gap in the center in order to allow the un-scattered photon beam to pass the detector without destroying the device. The four quadrants are electrically independent and can practically be driven separately.

The quarter million pixels of each quadrant are implemented on four so called Ladders. A Ladder forms the smallest hierarchy level of the DSSC detector which can be used as an independent imaging system. The focal plane of the Ladder implements two large monolithic 128×256 pixel sensors (DEPFET or MSDD). Each sensor carries eight 64×64 readout ASICs, which are bump bonded on the backside. The power and control signals are distributed from the Mainboard (MB), which is wire bonded to the backside edge of the sensor dies. Perpendicular to the MB, one IO Board (IOB) and four Power Regulator Boards (PRBs) are connected as shown in figure 5.9. The IOB comprises an FPGA, which implements the fast control of the power cycling, and readout of the data from the ASICs. The IOB is described in more detail in section section section 5.6.1.

On the PRBs, large capacitor banks are implemented which store the power required for the $600\,\mu s$- data acquisition sequence. The interconnection on the backside of the five boards is implemented by the Module Interconnect Board (MIB). Besides the PRBs and the IOB, there is one additional direct cable connection between the MB and the MIB: the so called Flex Cable (FC). This connection carries the digital JTAG control signals for the readout ASICs, the static sensor voltages and one analogue ASIC monitor signal.

The connection of one quadrant to the outside of the vacuum vessel is implemented by one flexible cable of $\approx$ 40 cm length, which is connected to the backside of the MIB. This

[2] A Gray-code counter is a binary counter that changes only one single bit per count. This counter eliminates problems with false codes during count transitions.

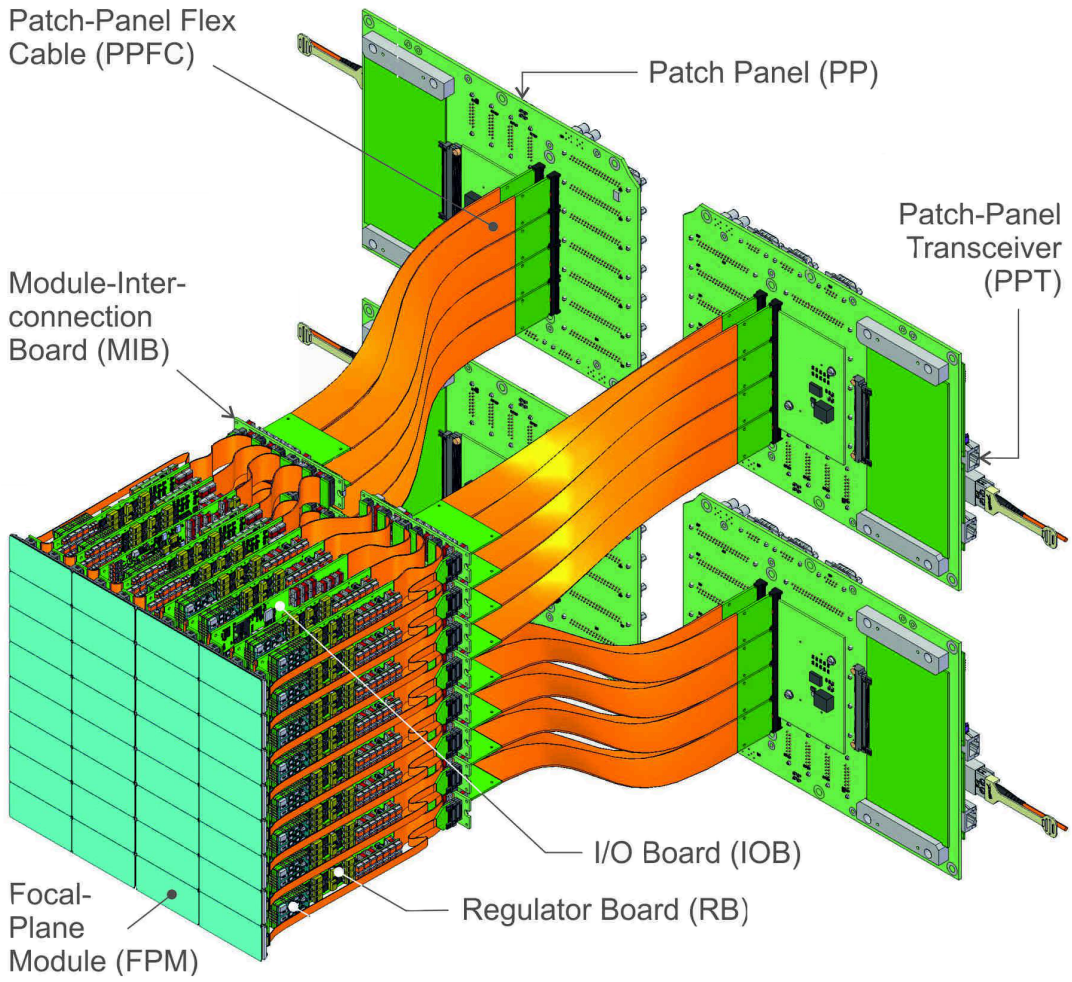


Figure 5.8.: **Layout of the DSSC detector.** (Courtesy of K. Hansen, DESY)

cable carries all supplies as well as the digital and analog signals of the DSSC quadrant. The length and signal transmission quality of this flexible cable is most critical for the high frequency signals which implement the FPGA-to-FPGA data transmission. At the outside of the vacuum, the flexible cable is connected to the Patch Panel, a large board which provides all connections for the power plugs and carries two further important cards. The Safety Interlock Board (SIB), and the Patch Panel Transceiver (PPT), which is the main control and DAQ device. One PPT is thereby responsible for one quadrant. The Patch Panel Transceiver implements the main FPGA of the detector system and implements essential parts of the readout chain. It will be described in detail in section 5.6.2.

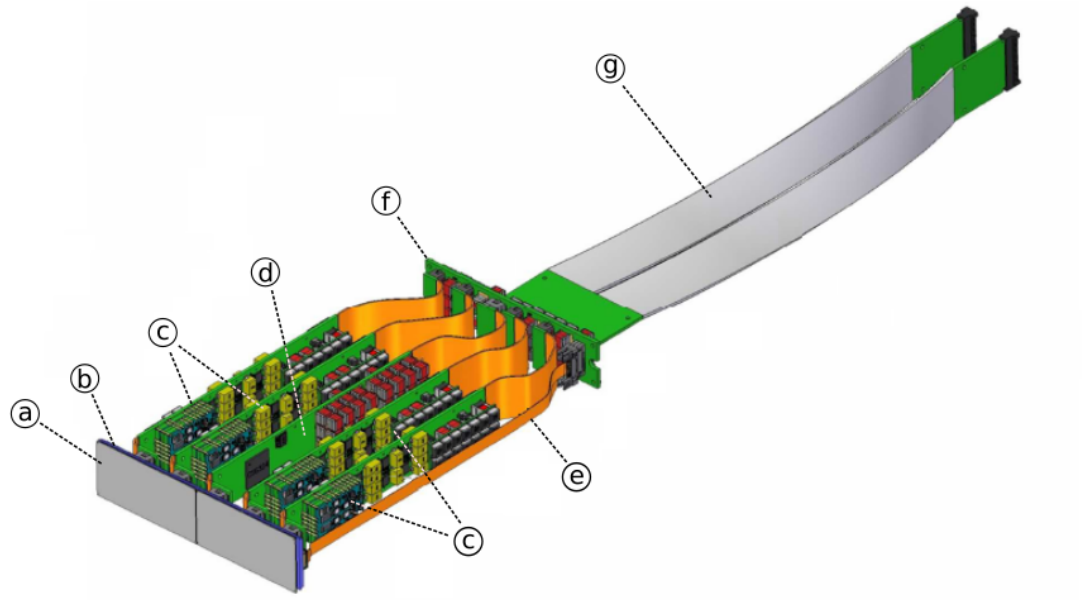


Figure 5.9.: **Details of a Ladder module.** a) Sensor with bump bonded readout ASICs on the backside. b) Mainboard. c) Power Regulator Boards. d) IO Board. e) Flex Cable. f) Module interconnect board. g) Flexible cable connection to the outside of the vacuum.

5.6 Components of the Readout Chain

In this section the building blocks of the readout chain are described. The main work of this thesis has been invested to develop the firmware for the devices described in this chapter. In order to give the reader an overview about the implemented electronics and details about the environment for the firmware implementation, the devices which build up the readout chain are described on the following pages.

5.6.1 Implementation of the IO Board

The IO Board (IOB) was developed by Thomas Gerlach, during the early phase of the DSSC project. It builds the first FPGA stage and is positioned in the vacuum directly connected to the MB. The primary task of the IOB is the data reception of all 16 ASICs of a Module and

the bundeling of all channels into one high speed serial data stream which transfers the data to the PPT FPGA. Secondary task are the generation of the signals to switch the ASIC and Sensor powers on the PRBs and to program the four clock buffers on the Mainboard.

Additional capacity banks and a temperature sensor are also implemented on the board.

The board is a ten layer PCB of size $229.3\,mm \times 22.6\ mm$. Its complex design requires high component density within the available space. Also a flexible four-layer Polyimide core is implemented to allow rectangular PCB to PCB connections.

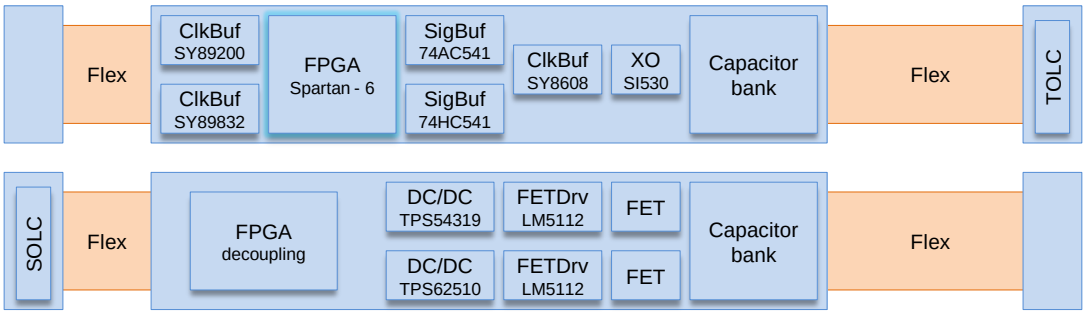


Figure 5.10.: **Building blocks of the IOB [83].** Top and bottom view.

The heart of the IOB is a Spartan 6 FPGA[3]. The FPGA receives the ASIC data via LVDS (Low Voltage Differential Signaling) and concentrates the 16 data connection to one high speed serial stream, which transports the data to the master FPGA stage on the PPT. Therefore, the Spartan - 6 FPGA implements three MGT lanes which are coupled to one logical 7.5 GBit/s link that implements the Aurora[4] protocol.

A detailed description of all building blocks of the IOB can be found in the thesis of Thomas Gerlach [83]. Only minor changes have been made on the physical design of the IOB since T. Gerlach published his thesis. The first draft of the firmware has been developed by T. Gerlach. During this thesis it has been extended and adapted to the final readout concept, which will be presented in chapter 7.

5.6.2 Design of the Patch Panel Transceiver

The **P**atch **P**anel **T**ransceiver (PPT) is the master readout and control device of the DSSC detector. It provides modern components for data transmission and intelligent timing control of the detector. The overall functionality can be divided into three areas:

- **Datapath**: A valid data rate of 22.4 GBit/s, distributed on the four quadrant IOB channels is sampled, formatted and reordered in a chain of multiplexers and data buffers. At the output of the processing chain the formatted data of each channel is forwarded to a 10 Gigabit-Ethernet output. Summarized in one QSFP+ transceiver, the PPT provides a nominal output rate of 40 GBit/s.

[3] XILINX XC6SLX45T-CSG324-3I
[4] Serial FPGA to FPGA data transmission protocol.

- **Timing**: Low jitter reference clocks for the ASIC must be provided and distributed, derived from the XFEL reference clock. Fast control telegrams are received and processed to provide intelligent control of all components of the detector, in order to guarantee system wide synchronicity.
- **Control**: A Gigabit Ethernet interface is implemented to provide slow control access to the system registers. This includes also the download of configuration data to the IOB and readout ASICS. Besides automated, timing sensitive start-up and shutdown procedures also stand alone trigger sequences are implemented to allow independent laboratory tests while the Detector is disconnected from the XFEL control system.

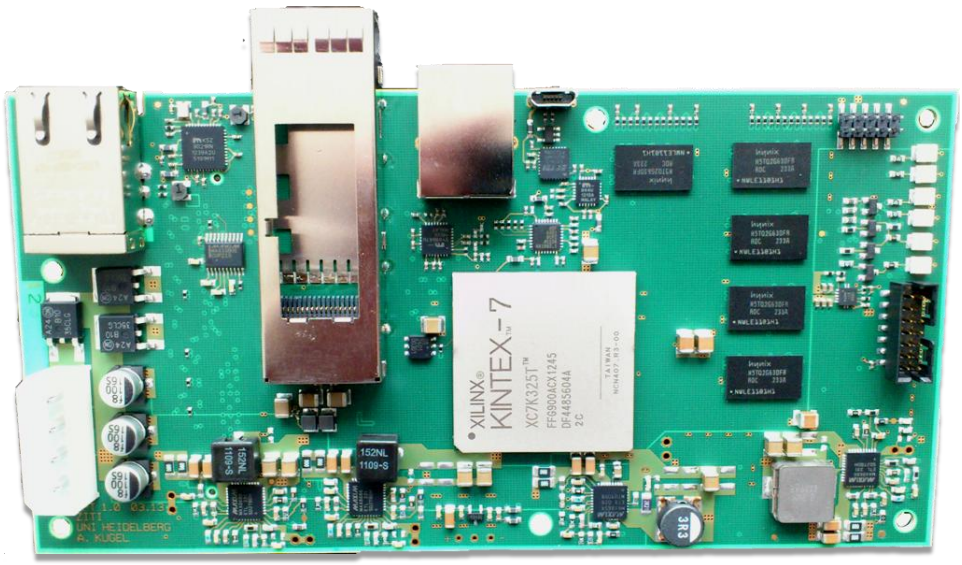


Figure 5.11.: **Image of the Patch Panel Transceiver (PPT).**

In order to achieve high performance, new FPGA technology and state-of-the art electronics are implemented. The PPT board is designed as a 14-layer PCB of size 80×160 mm, The central element of the PPT is a Xilinx Kintex-7 XC7K325T-FFG900-2 FPGA of the 2012 generation. It is a high I/O pin count device of intermediate size. The device provides 16 high-speed 12.5 GBit/s serial links, 12 of which are used for an unidirectional connection, on which the raw front-end data is received using the Aurora protocol. The remaining four channels are directly attached to a four-channel QSFP+ transceiver. This type of transceiver allows to bundle four 10 GE links within one single optical cable. Since the Kintex-7 generation already implements an integrated 10 GE PHY to drive directly the 10 GE links, a dedicated PHY device becomes obsolete for the fast link which gives additional space on the PCB for other components. A schematic overview is displayed in figure 5.12.

For data buffering and reordering, a high performance 1GB DDR3-1600 memory[5] is implemented which offers sufficient capacity to store the data of two complete readout cycles of a quadrant.

[5] 4x256MB 2G SK-Hynix H5TQ2G63DFR-RDC running at 800 MHz double data rate at 64-bit write width.

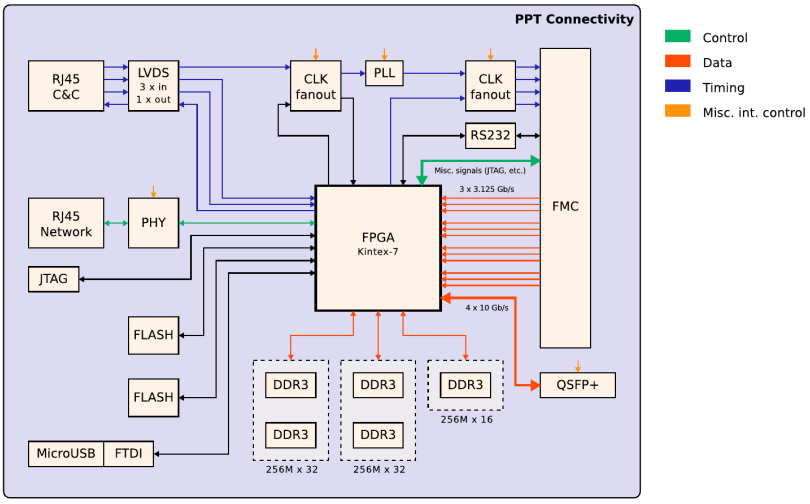


Figure 5.12.: **Building Blocks of the PPT [83].**

For control and configuration, the Kintex-7 FPGA implements a Gigabit Ethernet interface via RJ45 connection. In order to drive the Ethernet, especially the TCP-IP stack, a Microblaze microcontroller in instantiated in the Kintex-7 FPGA, which runs an embedded Linux operating system. For debugging and monitoring, a serial interface via MicroUSB is also provided which enables access to the Microblaze e.g. during the boot process. The softcore microcontroller is connected to a separate 256 MB DDR3 memory. Because the slow control Ethernet connection could not be connected to an internal high speed serializer, an external PHY is required to operate this link.

Inside the FPGA, the microcontroller is connected to a set of configuration registers which allow slow control of several custom design Verilog modules that are used for real-time control of the different detector parts. Furthermore, the Microblaze microcontroller allows to integrate automated, intelligent control sequences which are used for control or during initialization.

The interface for the C&C timing system is also implemented by an RJ45 connection. Over this connector, two LVDS timing signals and a 100 MHz synchronization clock are received and one status signal is transmitted (see section 4.1). Additional components like clock and signal buffers are used to improve the received C&C signals. An additional PLL combined with a clock mux switch allow the switching between the distributed system clock and a local clock from an oscillator which can be used during stand alone operation when the C&C system is not available.

A standard JTAG interface allows direct debugging of the FPGA firmware. This feature has been frequently used during the early development phase.

The specification of the PPT as well as the selection of the components was defined by Andreas Kugel in 2011. In order to profit from professional experience, the final design and layout of the PCB board was done by GED, Gesellschaft fr Elektronik und Design mbH, a German expert company for PCB design. First prototypes have been delivered in the beginning of 2014.

5.7 Booting and Initial Configuration

Essential for later operation is the ability to start the system automatically and to provide updates without direct access to the hardware. When the detector is placed in the vacuum container at the beamline, which describes the final beamline environment, manual access to the components is not always guaranteed. In order to realize automated initialization of the Kintex-7 FPGA on the PPT, two SPI flash memories have been added to the board and connected to the FPGA. After power up, the configuration bit file, which describes the FPGA firmware, is directly loaded from the flash memory. The memory is therefore directly connected to the corresponding initialization pins of the FPGA.

After the FPGA firmware is programmed, the Microblaze is automatically started. In the start configuration of the Microblaze is only one small piece of code available which is stored in the FPGA block RAMs. This code is the boot loader which describes the copying of the Linux kernel from the SPI flash memory into the memory of the Microblaze. At the end of the boot loader, the Linux is booted. Several scripts and tools are included the file system of the Linux image, and thus available directly as soon as the Linux is ready.

The bit file to program the IOB FPGAs is also contained in the Linux image. Due to this mechanism which uses the two non-volatile memories, self-initialization of the PPT is performed. This means, that only a few minutes after the system is switched on, a ready system is automatically provided.

For a detailed description of the single steps which are executed in the FPGA during this boot process please refer to the Xilinx User Guide UG470 [84].

After booting of the Linux kernel, an initialization script is automatically executed which starts all essential tasks which are required to operate the system. Most important tasks are:

- Mounting of the file system which is contained in the image.
- Reading of the serial number from the IC on the PPT and configure the IP address of the slow control Ethernet according to the retrieved number.
- Start the *SlowControlServer* process, which provides the control interface to the FPGA firmware.
- Also some daemons for general purpose are started: telnet, ftp and logging daemons.

The end of the start up sequence is indicated by an initial reset, which can visually be noticed by a special pattern of the implemented LEDs on the PPT.

The content of the flash memories can be updated directly from the Microblaze via the SPI interface. Therefore, the new binary file has to be loaded via FTP to the Microblaze and programmed into the memory using the **flashcp** command. After reprogramming of the flash ROM, the new firmware or Linux image can be activated by rebooting the PPT. This can either be done by simply switching the supply power off and on again or by setting a special register in the FPGA, which initiates the reconfiguration of the FPGA from the flash memory.

/dev/mtd0	device from which the Linux kernel is loaded
/dev/mtd1	device from which the FPGA firmware is loaded

Table 5.2.: **Assignment of the flash memories.**

5.8 Power-Cycling Principle

A big challenge in the DSSC detector is the supply of the required electrical power of the system. In order to minimize the power consumption all sensor and readout circuits are active only for the duration of 600 μs during image acquisition phase. The power supply is realized by the four power regulator boards. These boards implement fast voltage regulators and large capacitor banks that store the required electrical power to provide a constant voltage of $\approx$ 1.4 V at a current request of 2.9 A per regulator board for the whole image processing time. Additional capacitor banks are implemented also on the IOB and on the MIB to provide as much capacity as possible (see red cubic parts in figure 5.9). Another task of the regulator boards is to generate the fast clear pulses which are required to remove the signals from the DEPFET sensors after every acquisition cycle [69].

5.9 Safety Concept

High voltages and currents, vacuum and low temperatures must all be monitored and can do harm if things are going wrong. This is why the DSSC consortium also decided to prepare a safety concept for the detector. Therefore, a special microcontroller card, the Safety Interlock Board (SIB), was developed which monitors relevant factors during operation. The SIB can be mounted on the backside of the Patch Panel outside of the vacuum. The microcontroller monitors data from several sensors and if any environment factor reaches a critical level, it is able to trigger an emergency shut down of the system. The SIB monitors the temperature levels of the detector, therefore a temperature diode is implemented in each readout ASIC. One additional temperature sensor is located on the IO Board. Further sensors monitor the vacuum and air pressure inside the vacuum vessel. Also voltage levels and electrical currents are observed.

5.10 Prototyping and Development

In this section the environment is presented, which has been used during the early phase of the firmware and software development. Several test ASICs have been designed and measured in order to iterate towards design of the final readout ASIC. The different test ASIC generations are listed below. Since the final structures of the DSSC detector have not been available to test and develop the control and readout firmware and software of the detector system, several test environments have been designed and will be presented in the following.

5.10.1 Test ASIC Generations

The DSSC consortium developed the F1 ASIC, the first full format ASIC, between 2009 and 2016 [70]. The final design was gradually refined during a lengthy iterative process and

ASIC Name	Sumbission	Number of Pixels	Special Features
MM1	9.8.2010	8x8	First matrix ASIC. Impl. DDR memory, no digital control block.
MM2	9.8.2011	8x8	Second matrix ASIC. Impl. SRAM Memory.
CNTRL1	11.2011	4x1	First ASIC, which implements digital control. Improvements in ADC and FE.
MM3	2012	8x16	Third matrix ASIC, full length power chain. One half GCC ADC, one half In-Pixel counter.
F1	2014	64x64	First full scale matrix ASIC. Implements first version of Day-0 Mini-SDD FE parallel to DEPFET FE.
MM4	2014	8x16	Identical to F1 but smaller pixel matrix.
MM5	2014	8x16	Alternative experimental MSDD FE, designed by Florian Erdinger [85].
MM6	2014	8x16	Experimental improvements of MSDD FE.
D0M1	2015	1x4	No DEPFET FE, further experimental improvements in MSDD FE.
MM7	Q2 2016	8x8	Three different approaches for MSDD FE in order to find best version for F2 ASIC commissioning.
F2	Q1 2017	64x64	Further improvements in MSDD FE (CSA). Possibility to switch single metal layers to physically disconnect MSDD FE resp. DEPFET FE.

Table 5.3.: **Basic features and of the most important test ASIC verstions.**

refined further and further. This required a variety of different test ASICs developed and tested over the years. An overview of major development steps including all matrix test ASICs is shown in table 5.3. All mini-matrix ASIC versions can be connected to DEPFET detectors which allows to perform measurements using the early versions of the readout ASIC in order to gain experience over all building blocks and the performance of the readout pixel already in early states of the project.

Due to the weak noise performance of the first full format ASIC, F1, with MSDD sensors (see section 5.3.3), the development of the F2 ASIC, which implements an improved MSDD front-end, has been initiated. If the improvements in F2 work as expected, this ASIC version will be installed in the first DSSC megapixel camera.

5.10.2 The ASIC Test Setup

With rising complexity of the test ASICs, a modular test system was needed to operate and readout the new chips. In order to minimize PCB design effort a sophisticated test

environment was needed which allows testing of multiple ASIC generations. After the first version of the digital part was introduced in the CNTRL1 chip, the pin-out of the test ASIC has been more or less finalized. This allowed the design of a common setup which would remain compatible to the following ASIC generations. The final design of the ASIC Test Setup is shown in figure 5.13.

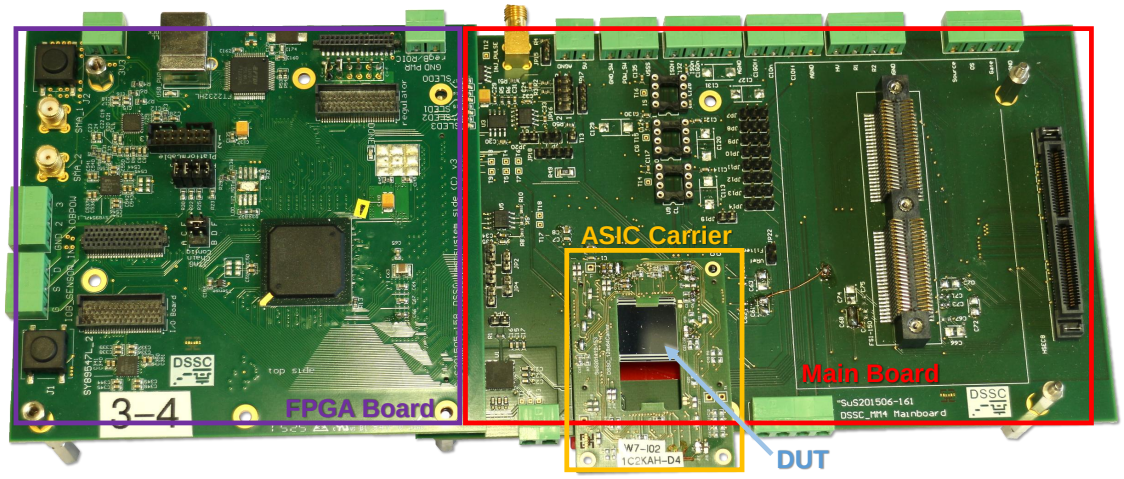


Figure 5.13.: **Image of the ASIC Test System.** The modular setup is composed of three boards. The ASIC Carrier board allows easy change of the DUT, which is glued and bonded on the carrier board. In this way, the setup can also be used for various ASIC generations.

The setup is composed of three PCB boards. The FPGA Board, the Main Board and the ASIC Carrier, on which the device under test (DUT) is glued. The FPGA Board implements the main control FPGA for configuration and readout of the test ASIC. Additional infrastructure like clock generation and debug signals are also available. A large connector provides large number of pins to be routed to the Main Board. The Main Board implements basically the infrastructure for the sensor supplies, a clock buffer of the same type which is implemented in the real system and lots of further debugging and testing features like e.g. a voltage DAC.

Furthermore, the FPGA Board implements connectivity for the operation of PRB and IOB prototypes in order to test functionalities of these devices in the early stages of the project.

The ASIC Test Setup is a very important part of the ASIC testing and development infrastructure of the DSSC project. To enable different groups of the consortium to take over measurement work, this setup has been distributed to different places. Because of its compatibility with various versions of the later test ASIC generations, it is frequently used by several people at different locations. Because of its importance for the consortium, the software which has been developed to control and configure the ASIC Test Setup and the ASIC became a very central part of the ASIC prototyping. This is also reflected by the design approach of the control software for the real detector system.

5.10.3 The PPT Test Setup

The PPT Test Setup defines an important intermediate step towards the final detector system. When the first PPT prototypes have been received from manufacturing, the development of this setup started. It implements all central components of a DSSC Ladder which implement configurable functionality. This makes the PPT Test Setup the first reduced prototype of the DSSC detector system which implements a realistic testing environment for the development of the detector firmware and software.

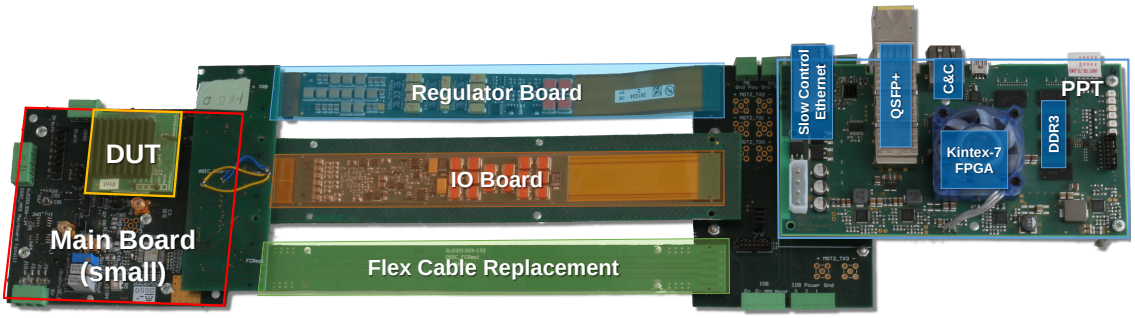


Figure 5.14.: **Image of the PPT Test Setup.** All basic components of the final Ladder are included: the PPT, one IOB, one PRB, one readout ASIC and the Main Board from the ASIC Test System.

The central purposes of this setup are the development of the detector control and the implementation of the readout chain. It implements the PPT, one IOB, one PRB and it is also compatible to the Main Board[6], which has originally been designed for the ASIC Test Setup. This compatibility allows to plug-in and operate identical ASIC Carrier boards in the ASIC Test Setup and the PPT Test Setup. With the connection of the ASIC Carrier board to the IOB and the PPT, all components of the DSSC readout chain are available. Also the communication from the software on a computer up to the focal plane can be tested using this test environment.

Although this setup allows only the connection of one ASIC in one module, it has nevertheless provided very good services to implement the complete readout chain. By inserting dummy data at the right locations, it is possible to emulate the missing data until the first Ladder prototypes, which implement all 16 ASICs, became available. The same applies to the control signals for programming the ASICs which led to a very quick commissioning of the first real Ladder systems.

By providing additional debugging and testing features compared to the real Ladder system, the PPT Test Setup became very important for the firmware and software development. Consequently, all modules and tasks of the firmware have initially been developed using this setup. Because of the high similarity of the PPT Test Setup to the real DSSC system, no special configuration files are required for operation. Consequently, identical firmware files can be used to program the FPGAs of the different systems.

[6]The small version of the Main Board is compatible to the normal version, but does not implement the infrastructure to apply the sensor voltages.

Part II

The Firmware of the DSSC Detector

The firmware implementation is divided in two chapters.

In the first chapter, the more general tasks and devices are described which are required to operate the system. Also the concept for synchronization of the single processes and devices of the detector is presented. The second chapter describes the readout chain of the detector, for which most of the work of this thesis has been spent.

The intention of the first chapter is to give the reader an impression of the communication between the single components of the detector system. An additional focus is placed on the synchronization of the different devices and processes in the detector which is described by regarding the main state machines. Because the Ladder defines the smallest unit which can be used as an independent imaging system, most descriptions refer to the operation of a single Ladder, but can easily be extended to quadrant operation. Since there is no electrical connection between the four quadrants of the megapixel camera, each quadrant can be seen as an independent device, at least from the perspective of the firmware.

The actual firmware development started with the first prototypes of the PPT boards. When the first implementations of the Microblaze microcontroller became operational, a small test setup has been produced which allowed to connect the PPT to the IOB. By step-wise extending this small test setup, it became more and more similar to the real detector system until all important components have been available (see Secsec:sec:ppttestsetup). This setup became the PPT Test setup which is an important part of the firmware development.

During the development of all firmware devices, a special focus was placed on stable operation during continuous image acquisition. The used programming language for all firmware modules is Verilog. Even though Thomas Gerlach used in his early version of the IOB firmware the programming language VHDL for his modules, all modules have been translated to Verilog. This work has been done by Jan Soldat to solve complications during firmware simulations and in order to simplify the further development of the modules.

6

Design of the System Control Firmware

This chapter describes the interaction between the different devices and the implementation of the main state machines which synchronize the different building blocks. From the digital control perspective, the DSSC detector is divided into three blocks: the Patch Panel Transceiver (PPT), the IO Boards (IOBs) and the readout ASICs. The PPT defines the master device which initiates all control and configuration sequences. The PPT firmware has direct access to all registers in the PPT fabric and indirect access to the IOB and ASIC registers via special controller modules. The image acquisition and the readout sequences are initialized from the PPT by means of fast telegrams, distributed throughout the system. Synchronization of the DSSC detector to the XFEL environment e.g. the photon pulses, is realized by synchronizing the PPT main state machine to the XFEL C&C system.

An overview about the connectivity of the DSSC system is depicted in figure 6.1.

The design of the firmware follows a certain concept: each task of the system is controlled by a distinct firmware module. A module can be of any complexity and is controlled via registers. A module which transfers data to a distant device provides a FIFO interface for buffering of multiple data words.

More complex modules implement own state machines. Depending on the module, this state machine can be started from two different origins: either manually from a register or, if it has to be synchronized to the image acquisition or the readout, from the main state machine. If it is controlled from a register it can only be controlled by one global instance. The global instance of the quadrant system is the Microblaze microcontroller which has direct access to all registers in the FPGA fabric and indirect access to the IOB and ASIC registers via special controller modules which are implemented in the PPT firmware. In order to simplify the bookkeeping of the register values in the control software, the Microblaze does not store any configuration values. New configuration values are just forwarded to the firmware modules.

The implementation of the system register configuration is described in section 6.2.

In section 6.3, the main state machines of the system and the most important tasks which have to be aligned for image acquisition are presented.

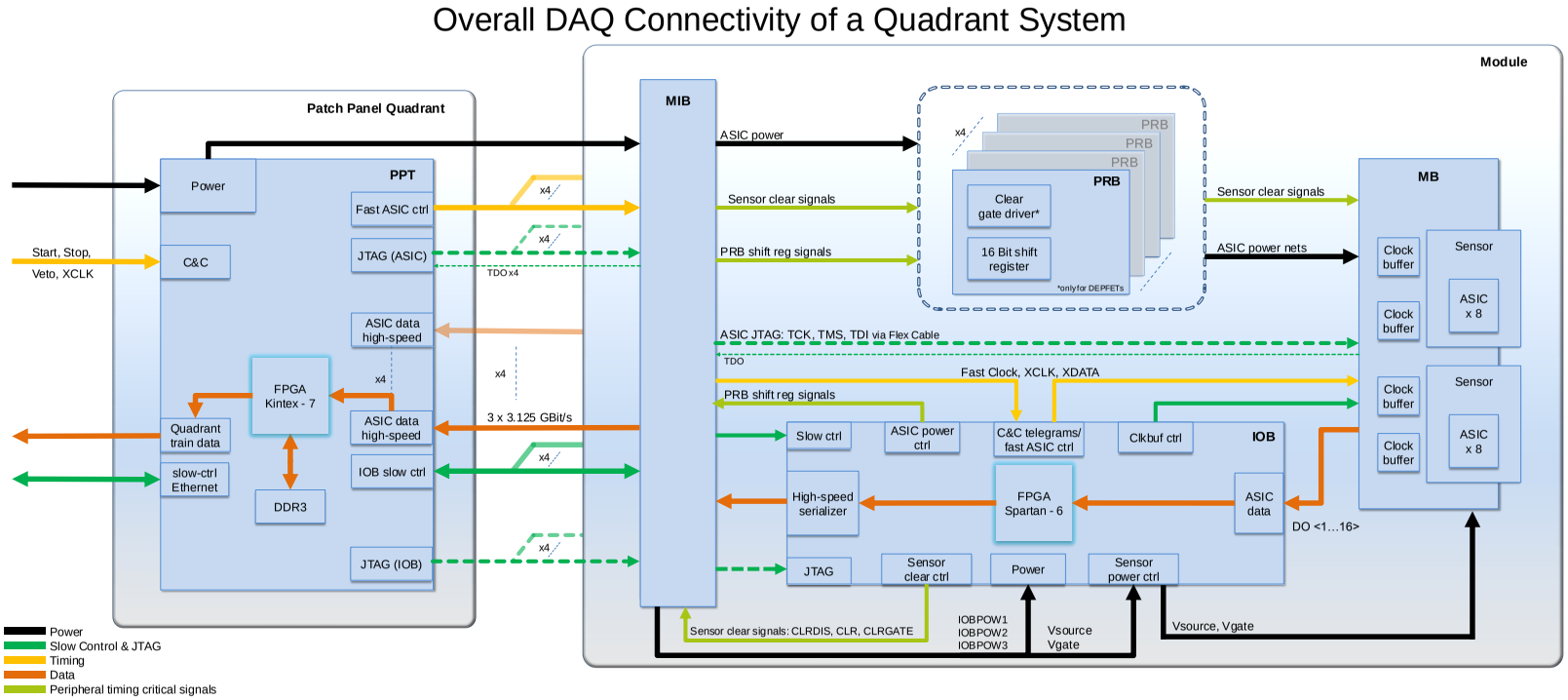


Figure 6.1.: **Overall connectivity of the quadrant system.** The Kintex-7 FPGA on the PPT acts as the master control and readout device. It controls four modules building one quadrant. A module implements one IOB which carries a Spartan-6 FPGA, the focal plane which consists of two large format sensors and 16 readout ASICs, and the four PRBs. While the readout ASICs are directly programmed from the PPT FPGA, the clock buffers and power regulators are configured from the IOB FPGA. The global synchronization of all components is implemented by the main state machine in the Kintex-7 which generates the global control and readout sequence. The readout is realized in three levels: from the readout ASICs, to the IOB FPGA and the third stage, the PPT FPGA in which the data is buffered in the DDR3 memory and image wise sorted before it is sent out via the QSFP+ (4×10 GBit/s) Ethernet link. The switched sensor supplies (VGate and VSource) and clear signals are required to operate DEPFET sensor, they remain unconnected if an MSDD sensor module is mounted.

6.1 Firmware Overview and Clock Distribution

All access to the PPT happens via the Ethernet interface. The Microblaze microcontroller in the FPGA runs an embedded Linux which is required to provide full TCP/IP functionality. From the Microblaze, the so called EPC interface (see section 6.2.1) is accessible which provides direct access to the internal registers in the firmware fabric. In this way, software which is executed on the Microblaze can access the firmware registers and control the firmware devices. Since the PPT FPGA is electrically connected to the IOB FPGA and the ASIC JTAG signals, the Microblaze has full access to all registers of the system.

A full overview about the connectivity of the firmware and digital parts of the readout ASICs is shown in figure 6.2.

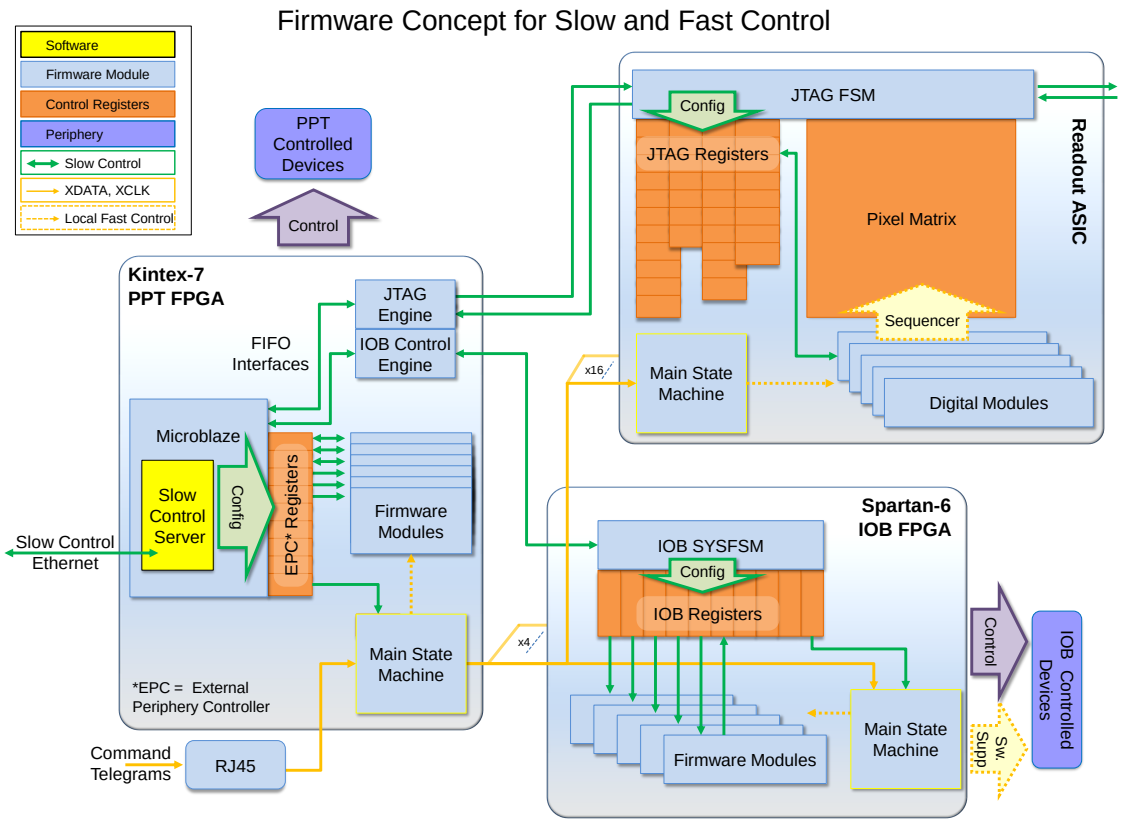


Figure 6.2.: **Overview about the firmware concept for slow and fast control.** The system is controlled and configured via the Slow Control Server running on the PPT. System synchronization is initiated from the main state machine on the PPT, which generates the fast control signals to synchronize the IOB and the readout ASICs. During image acquisition all peripheral devices are controlled from the main state machines on the respective devices. Only one of four IOBs is shown.

6.1.1 Implementation of the Microblaze

The design of the Microblaze microcontroller and its initial implementation was done by Jan Soldat at the beginning of the PPT firmware development. For convenient specification of the features of the Microblaze, Xilinx provides a comprehensive tool set: the **E**mbedded **D**evelopment **K**it (EDK) [86]. Using the Xilinx EDK, the composition of Microblaze features have quickly been finished in a very early state of the PPT firmware project.

The Microblaze is an embedded microcontroller which is provided as a soft core by Xilinx for instantiation in Xilinx FPGAs. It is a 32-bit RISC microcontroller which offers the user various customization options. Further details are given in [87].

For the implementation in the PPT FPGA, the standard processor infrastructure is extended by a set of controllers which allow usage of external devices from the Linux operation system (OS). The controllers can be integrated into the microcontroller design using the Xilinx EDK:

- A DDR3 memory controller, to access the external 256 MB DDR3-1600 memory.
- A Gigabit Ethernet controller, which is used for all slow control applications.
- A UART controller, to drive a USB interface for debugging.
- A GPIO[1] interface, which enables the access to the JTAG connection of the four IOBs and also to the JTAG configuration signals of the Kintex-7.
- Two SPI controllers, to access the flash memories from software.
- An EPC controller, to access the registers in the FPGA fabric.

For inter-device communication, the Microblaze implements AXI [88] buses, which is a Xilinx adoption of the AMBA bus, a widely used type of a microcontroller bus.

The clock speed of the Microblaze is 100 MHz. The processor clock is driven by an external oscillator providing a continuous clock of 200 MHz, which is divided by an internal PLL.

6.1.2 Clock Distribution

Besides the steadily running 100 MHz AXI clock of the Microblaze, the DSSC detector system implements several different clocks in separated clock domains.

The AXI clock from the Microblaze is the read and write clock for all connected registers and FIFOs. Since these registers are actually just slow control registers, the AXI clock domain can be regarded as asynchronous to the rest of the PPT firmware.

The main state machine of the system is driven by the PPTCLOCK, which is connected to a switchable clock input. The input clock of the PPTCLOCK can either be driven by the global clock from the C&C network or, while running in standalone mode, by the AXI clock from the Microblaze. The switching of the clocks is controlled by a clock multiplexer IC[2] in the FPGA periphery on the PPT.

[1] General purpose I/O
[2] Micrel SY89844UMG - a low jitter 2:1 LVDS multiplexer

The PPTCLOCK is the central clock, which is used to synchronize the whole detector system. It is distributed to the IOB and the readout ASICs, renamed as the XCLK signal, and is also used as reference clock for the fast 700 MHz ASIC clock, which drives the fast Gray-code counter of the ADC. The PPTCLOCK drives all firmware modules which are running synchronously to the main state machines: the SIB control module, the IOB control module as well as the JTAG control modules.

In the data path, several other clocks are used: the Aurora core outputs data synchronously to a separate clock. Also the DDR3 memory controller as well as the *Ethernet Engine* operate at their own clock domains. The clock domain crossings in the data path are always realized by FIFO stages, which support separated read and write clocks.

6.2 Register Configuration

All control of the system happens via the manipulation of registers. Three different types of registers can be distinguished after their location in the system:

1. the EPC registers, located in the firmware of the PPT FPGA
2. the IOB registers, located in the firmware of the IOB FPGA
3. the JTAG registers, located in the readout ASICs

All three types of system registers can be accessed from the Microblaze: the EPC registers directly, and the IOB and JTAG registers indirectly via custom control modules. For the access of the EPC registers, the Microblaze has been extended with the EPC controller. The EPC controller implements a simple wrapper interface for the AXI bus, which allows easy communication between the Microblaze and custom firmware modules.

There are two types of devices which can be controlled by the EPC protocol. One type are devices implementing just registers which constantly provide their configuration value to the firmware logic. The other type are devices connecting a read and a write FIFO to the EPC controller. FIFO devices can be used to transfer larger numbers of data words. The later type is used to program the IOB registers and of course the JTAG registers in the readout ASICs. A firmware module, which is connected to the EPC controller via one of the two EPC register types, is called 'EPC device'. Each EPC device can be accessed via a unique address. An EPC device can be anything from a simple register storing configuration values up to quite complex device which implement an own state machine.

In the following, the configuration of the different register types is described.

6.2.1 EPC Register Configuration

The EPC register configuration protocol is specified by Xilinx and contains only few signals:

on the firmware side, a self designed EPC register module implements the actual register which stores one or multiple 32-bit data words. A register can be selected by its 32-bit address. The implementation of read or write cycles of **multiword** registers is realized by a cyclic counter. While register values of the simple EPC register modules are constantly available in the firmware logic, a FIFO device implements a state machine which decodes the words passed from the FIFO.

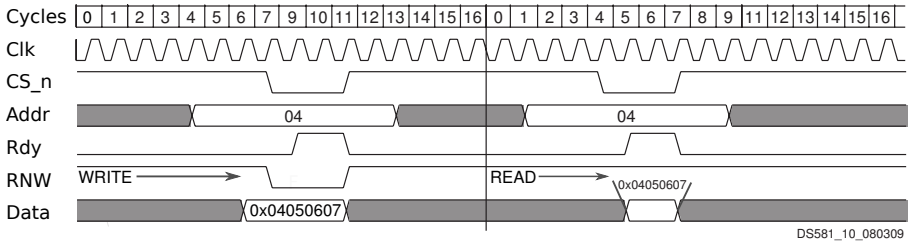


Figure 6.3.: **EPC write and read cycles[89].**

The signals of the EPC protocol are shown in figure 6.3. The **CS_n** signal activates the register access and signals that the address is valid. In case of a write cycle, the 32-bit data word has also to be valid during **CS_n** = 0. The **RNW** (ReadNotWrite) signal defines the type of register access. The **Rdy** signal is an input to the EPC controller and signals that the write procedure has been successful. During a read cycle, it signals that the read data is valid. If the **Rdy** stays low, the Microblaze waits until **Rdy** goes high. The **Rdy** signal is used to protect against writes to a full FIFO or reads from an empty FIFO. Further details of the EPC protocol can be found in [89].

6.2.2 IOB Register configuration

The IOB configuration registers are accessible from the Microblaze via the *IOB Control Engine*. For each of the four connected IOBs, one entity is instantiated. The physical connection for the bit transfer is implemented by a single 2.5 V signal line, which is used for both read and write transactions. The data transfer is realized by a simple one-wire protocol which has been defined by Thomas Gerlach. The communication is running synchronously to the global 100 MHz PPTCLOCK. On the PPT side, the *IOB Control Engine* implements the master device which initiates the data transfer. On the IOB side, the *syscntl* module serves the requests accordingly. The transfer protocol is shown in table 6.1.

Command	Start bits	Command bits	Payload	Description
READ	10	1000	<Addr> (16) + <RdData> (32)	Reads 32-bit data from address
WRITE	10	1001	<Addr> (16) + <WrData> (32)	Writes 32-bit data to address

Table 6.1.: **One-wire data transfer protocol between PPT and IOB.** A write / read sequence is always initiated from the PPT by sending the Start of Frame (SOF) bits '10'. During a read, the line direction is inverted and the wire is driven by the IOB.

The configuration registers in the IOB are implemented in one central register module which contains all configuration registers. The configuration values are connected to the corresponding firmware modules and are constantly available in the IOB firmware, similar to the implementation of the EPC registers.

6.2.3 ASIC JTAG Register Configuration

In case of the *JTAG Engine*, the FIFO interface buffers the configuration data which allows to change the clock speed of the JTAG interface in order to solve issues during the ASIC register programming.

The JTAG registers are directly programmed from the *JTAG Engine* in the PPT. There are two types of JTAG registers in the readout ASICs. One type of registers are described in a Verilog module and are contained in the synthesized, digital part of the readout ASIC. These registers are implemented in the global periphery and exist only once per ASIC. The second type are custom-designed shift registers which are placed in the pixel matrix and also in the periphery for the decoupling capacitors. The pixel registers are implemented once per pixel and connected in one long JTAG chain. Both register types are controlled by the same JTAG module[3] implementing the standard JTAG protocol [90].

In a Ladder module, the JTAG signals of all 16 ASICs are connected in a row. In order to realize reliable register programming in the long JTAG chains, the *JTAG Engine* implements a slow down mechanism which allows to decrease the programming speed. The slow down is realized by a clock divider, which runs synchronously to the global 100 MHz clock. The clock divider has a range from 0 to 255 which results in a range for the programming clock from 100 MHz to $\approx$ 392 kHz.

Especially in case of the pixel registers, the JTAG chain becomes fairly long: each readout pixel of the F1 ASIC provides 47 configuration bits. For a Ladder camera, the configuration bit stream sums up to a length of $4096 \times 16 \times 47 = 3014656$ registers which means 376 kB configuration data. Transfer of such a large number of configuration bits is not trivial. The *JTAG Engine* implements a FIFO at the input which can store only 16 kB. In order to prevent the Microblaze to write into a full FIFO, which would result in data loss, the FIFO full flag is used to signal that the *JTAG Engine* is currently not ready. This enables dynamic write access to the *JTAG Engine* from the Microblaze, directly from the firmware side, without the need of additional logic in the software.

For default JTAG programming, the FIFO is not required to have this size since the dynamic regulation works fine. In case if the speed of the JTAG programming becomes important, a special fast programming mode can be enabled. Depending on the JTAG programming speed, the dynamic regulation suffers from additional delays if the Microblaze induces undefined pauses e.g. while the process which writes into the FIFO is interrupted by the Linux OS. In order to realize interruption free, fast JTAG programming, the large FIFO can be preloaded with configuration data while the *JTAG Engine* is stopped. After all data is written into the FIFO, the *JTAG Engine* can be enabled and the data in the FIFO is programmed in a row without any induced delay from the software.

Further details about the implementation and the functionality of the single JTAG registers in the ASICs can be found in the PhD thesis of Florian Erdinger[73].

[3] The main state machine of the *JTAG Engine*, which is instantiated in the PPT FPGA, is also instantiated in the FPGA of the ASIC Test Setup.

6.3 Main State-Machines and Synchronization

The synchronization of the whole detector system to the incident photon pulses is a central task of the firmware. All devices and firmware modules have to be precisely aligned in order to have everything be prepared when the X-ray pulses arrive. Lots of counters and states have been introduced in order to realize a stable and flexible state machine which provides all required functionality for the data acquisition and readout sequences.

The global synchronization of the system is controlled via fast telegrams which are distributed via the XDATA lines (see figure 6.2). Besides the commands which have been defined in the ASIC control interface, additional fast commands have been introduced to start special tasks in the IOB. An overview about all fast telegrams and their purpose is shown in table 6.2.

Command	Command bits (5)	Understood by	Description
Start Burst	10000	IOB & ASICs	Start image acquisition sequence
Start Readout	10001	IOB & ASICs	Start of readout sequence
Veto	10010	ASICs	Re-use image slot
Start TestPattern	10011	IOB & ASICs	Start send out test patterns
Stop TestPattern	10100	IOB & ASICs	Stop send out test patterns
Power Up	10101	IOB	Start Power-Up Sequence

Table 6.2.: **Synchronization and control commands that are transferred on the XDATA lines from the PPT.**

The PPT sends each command to all four connected IO Boards from where they are further propagated to all readout ASICs. Thus, each command is always received by all devices which are connected to the XDATA lines. However, only by transferring the starting point of the image acquisition, the system cannot completely be synchronized. In order to realize synchronous state transitions during the image acquisition phase, the IOB and the PPT have to partly mirror the states of the ASIC main state machine.

Before the *Start Burst* command can initiate the image taking, the detector voltages have to be switched on and the whole system has to be ready for data taking. Different tasks, like the enabling of sensor voltages, the assignment of clear pulses and the alignment of the incoming *Veto* commands, have to be synchronized.

In general, the timings of the different tasks are aligned by various cycle counters. Especially in case of the IOB, which controls the switched ASIC powers, sensor voltages and clear signals, lots of counters are required to synchronize the different device controllers and to generate the final sequence. The correct assignment of all different counter values is realized by the control software. The dependencies and timings of the switched signals are shown in section 6.4. In the following, the main state machines of the digital devices of the system are described in order to give an overview about the different tasks during image acquisition and data readout.

6.3.1 ASIC Main State Machine

The main state machine of the readout ASIC controls the image acquisition sequence in the ASIC circuits and the readout procedure. Also a test mode is available in which the readout ASIC transmits configurable bit patterns on the output lines. This mode can be used to test the readout chain of the detector.

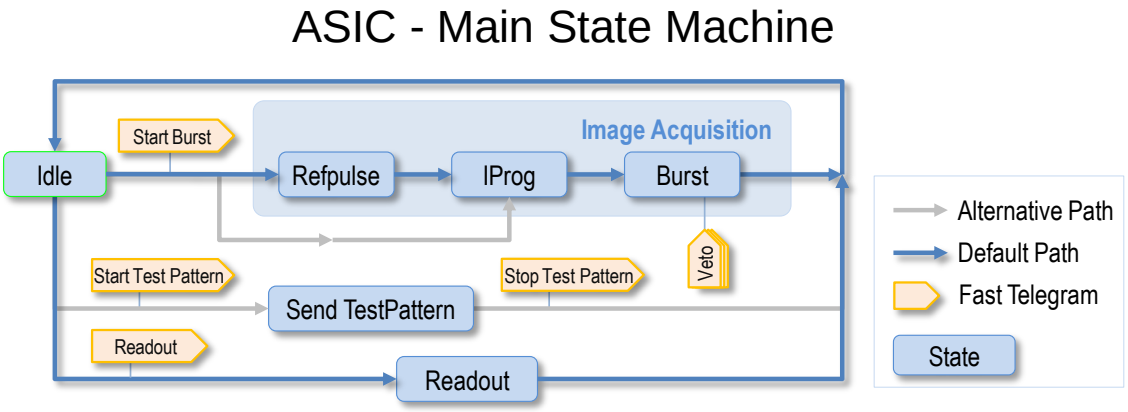


Figure 6.4.: **The master FSM of the readout ASIC.** The state machine is directly controlled by the fast telegrams. The image acquisition path and the readout path are active during every image acquisition cycle and are triggered by two independent telegrams. The *Veto* commands show effect only during the *Burst* state. The test pattern generation is switched on and off by separate fast commands.

During image acquisition, several states are stepped through. During the *IProg* state, the fine tuning of the current subtraction of the front-end is automatically trimmed. The actual image acquisition sequence is active during the *Burst* state. During this state, the ASIC sequencer is active and generates the signals to control the ASIC front-end, filter and ADC sequence. Also the *Veto* commands can only be sent, resp. are executed, during *Burst* to overwrite unwanted images.

The data readout is initiated by the *Start Readout* command. The readout can not be interrupted by a telegram and transfers always the full SRAM content. Besides the *Veto* commands during the *Burst* state, the ASIC state machine can only react on commands in *Idle* state.

6.3.2 PPT Main State Machine

The diagram of the PPT main state machine is shown in figure 6.5.

It can be divided into three major parts. The first part implements the decoding of the initial fast telegram and receiving of the train information from the XFEL C&C framework. Mainly the *Start Train* command and the train ID are sampled and the image acquisition cycle is initialized.

The second part implements the image acquisition cycle and mirrors the main state machine of the readout ASIC. This implementation has many advantages:

- the PPT knows exactly in which state the readout ASIC currently is,

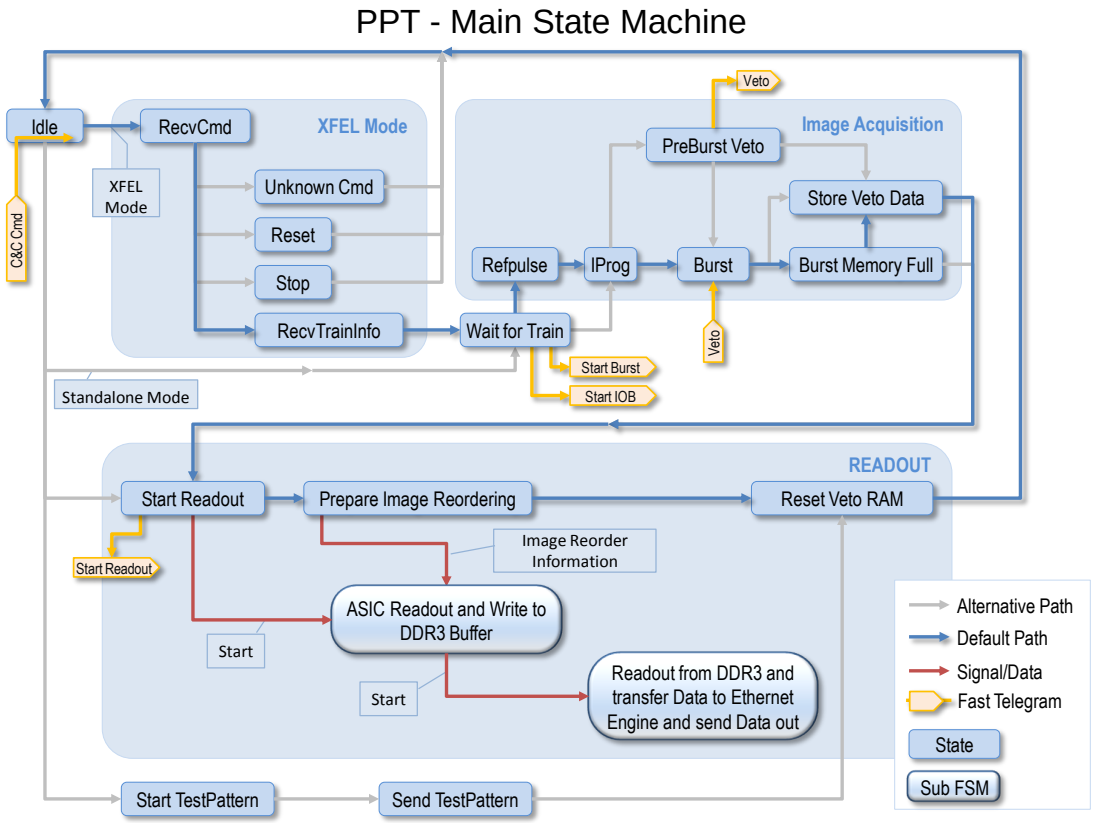


Figure 6.5.: **PPT main state machine.** The state machine can be divided into three major parts: the command interpretation, the image acquisition phase and the readout phase. The image acquisition phase is mirrored from the readout ASIC and must be synchronized to the IOB and ASIC operation. To manage the readout sequence, several separate state machines have been introduced which can operate independently allowing an overlap of the readout phase with other phases of the sequence.

- the cycle counters of the PPT main state machine can be configured with identical values as for the readout ASIC,
- the PPT can generate debug signals to indicate the current state of the readout ASIC while no access to the ASIC is possible,
- the Veto map can be generated (see later).

The additional *PreBurst Veto* state is used to generate a number of Veto commands at the beginning of the *Burst* state. This feature is frequently used during various types of measurements and allows to automatically erase the first images of a pulse train.

The third part initiates the readout sequence by sending a *Start-Readout* command to the IOB and ASICs and organizes the initialization of the *Veto* map for the next image acquisition sequence. The actual readout functionality is controlled by different state machines.

By distributing the readout functionality to different state machines, several tasks can be executed in parallel. This means the system can receive C&C telegrams already before the readout sequence has been finished. Additionally, a new train can already be written to the DDR3 buffer while the last packets of the previous train are still be sent out of the system. This is also visualized in figure 6.7.

Besides the default path, additional operation modes are available. For testing of the readout sequence or, for instance, of the pixel sorting in the software, the ASIC provides a *TestPattern* mode in which a configurable bit pattern is sent out of the ASIC serializer. This test mode is initiated by the *TestPattern* command and can manually be triggered from an EPC register.

6.3.3 IOB Main State Machine

The IOB main state machine is controlled directly from the PPT via fast telegrams. The default sequence is shown in figure 6.6. It is started by the *Power Up* command. After enabling the *Power Up* state, all controllers for the switched signals are started. The image acquisition states are also mirrored from the readout ASIC like in the PPT main state machine which enables the IOB main state machine to run synchronously to the readout ASIC. The current state information is distributed to the different controller modules, and thus can be used to align for example the clear pulses precisely to the readout sequence of the ASIC.

6.3.4 Sequence of the Switched Signals of the Detector System

The synchronization of the different detector components is also a very important task of the control firmware. The readout sequence of the front-end electronic has to be precisely aligned to the time point when X-Ray photons arrive. Therefore, the switched ASIC and sensor voltages have to be activated in advance and finally the readout has to be prepared and initiated after the image acquisition phase. The central entity which controls the synchronization of all system components is the main state machine in the PPT firmware. The PPT main state machine reacts directly on the command telegrams from the C&C interface and implements also a mechanism to internally generate triggers which allows to operate the system in a very useful standalone mode.

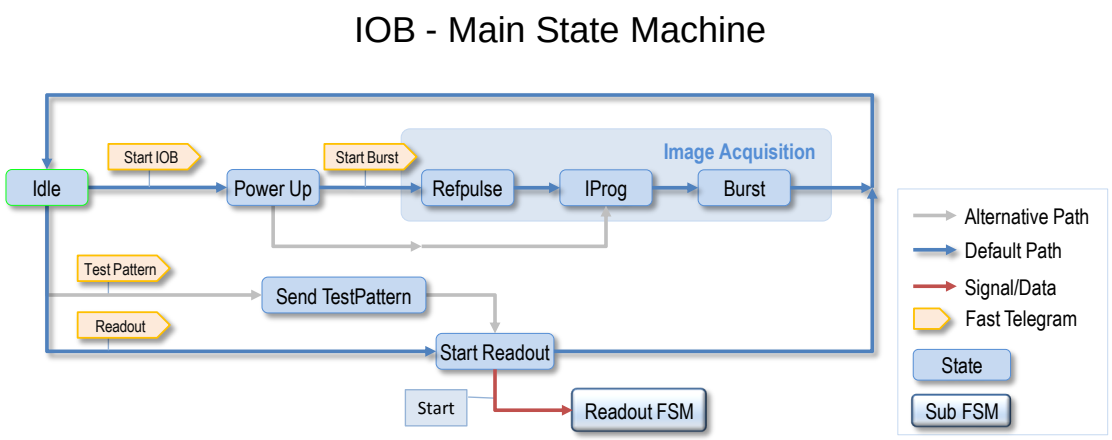


Figure 6.6.: **IOB main state machine.** It is controlled via the fast telegrams. The image acquisition states are mirroring the states from the readout ASIC. When the *Power Up* state is entered the different controllers for the analogue ASIC supplies, the clear signals and the DEPFET powers are started. Consequently, all switched-signals-controllers in the IOB can refer to the same starting point. The separation of the readout control into a second state machine enables an overlap between readout and the next power up sequence.

At this point, a summary of the large number of tasks and signals is given which have to be aligned in order to enable correct operation of the system. The description starts with the tasks of the global sequence and their chronological relations. The timings of the switched signals are shown afterward.

The global image acquisition and readout sequence can be divided into four phases:

1. The preparation phase which is described by the time span between the incoming *Start Train* signal from the C&C system and the first incoming photon pulse.

2. The actual image acquisition phase during the *Burst* state.

3. The primary readout phase in which the image data is transferred to the PPT.

4. The secondary readout phase in which the train data is read from the DDR3 image buffer and transferred via the fast 10 Gigabit-Ethernet link.

The relations of the different phases of the global sequence are shown in figure 6.7.

During operation, the XFEL laser generates every 100 ms a train of photon pulses. Typically, the *Start Train* telegram is sent 15 ms before the first photon pulse. Since the primary readout phase takes longer than the remaining 85 ms, the main phases must be able to overlap, which is implemented by several independent state machines, as described above.

After reception of the *Start Train* signal and the train ID, the PPT waits for the X-ray pulse train to arrive. During the wait phase, the *Power Up* command is sent to signal the IOB to start the power up sequence of the switched power signals. Shortly before the first photon pulse arrives, the *Start Burst* command is sent which brings the system synchronously into

Figure 6.7.: **Unscaled timing diagram of the global image acquisition and readout sequence.** The central tasks and states are visualized. The sequence is started by the *Start Train* command, which is received from the C&C network. The arrows indicate the dependencies between the different FSMs. Independent state machines allow overlapping of the different phases e.g the following pulse train is acquired and readout while the data of the first train is still being transmitted.

the *IProg* state. During the *IProg* and *Burst* state, the system can react on *Veto* commands from the C&C network.

Directly after the image acquisition phase, before the readout can start, the received *Veto* commands have to be evaluated in the PPT FPGA in order to generate the information which is required for the image sorting (see section 7.3.4). The readout of the focal plane is initiated by a *Start Readout* command on the XDATA line. During the primary readout phase, the image data is received from the readout ASIC and transferred from all four IOBs in parallel to the PPT where it is written into the large DDR3 image buffer. During this phase, the DDR3 address controller uses the previously generated *Veto* information to sort the image data. At the end of the primary readout phase, the image data is contained in one large block in inside the DDR3 image buffer. After the last word of the train data is written, the secondary readout phase is started. During this phase, the train data is read from the image buffer in blocks and packed in Ethernet packets before being transmitted over the four 10 Gigabit-Ethernet links. The implementation of the different tasks of the readout chain are presented in the following chapter.

Since each of both readout phases takes about 90% of the available time between two *Start Burst* telegrams, it cannot be avoided that they overlap. Consequently, the DDR3 memory controller has to provide alternating read and write accesses at full incoming data rate.

6.4 Generation of Switched Signals

For energy saving reasons, not needed ASIC and DEPFET supply voltages are switched off during the long ASIC readout. The switching of the effected signals introduces additional complexity which is implemented on the IOB and the PRB. The timing of the switched signals thereby is generated by the IOB FPGA.

A detailed timing diagram of the switched signals and supply voltages is shown in figure 6.8.

The sequence is initiated by the external trigger signal from the C&C network and managed by the PPT Main State Machine as shown in the previous section. The actual control of the shown signals is implemented in the IOB firmware. For every signal, a separate controller module is instantiated which implements various counters to generate the desired sequence.

Three groups of switched signals can be differentiated: The ASIC supply voltages, the DEPFET supply voltages, and the CLEAR signals to reset the DEPFET (see section 5.3.2).

The switches for the ASIC supply voltages are located on the PRBs and controlled by a shift register, which is programmed by the IOB FPGA. The different ASIC supplies can easily be switched by according bit patterns of the shift register.

The VSOURCE and the VGATE supplies for the DEPFET sensor are switched on the IOB and the switches are implemented by dedicated MOSFET transistors. For the VSOURCE switch, a special p-channel MOSFET[4] is implemented which can handle the high current of 7 A, which is consumed by the net. VGATE is a low current net, which consumes only a few 100 mA.

[4] Fairchild FDS6681Z -30V P-Channel PowerTrench.

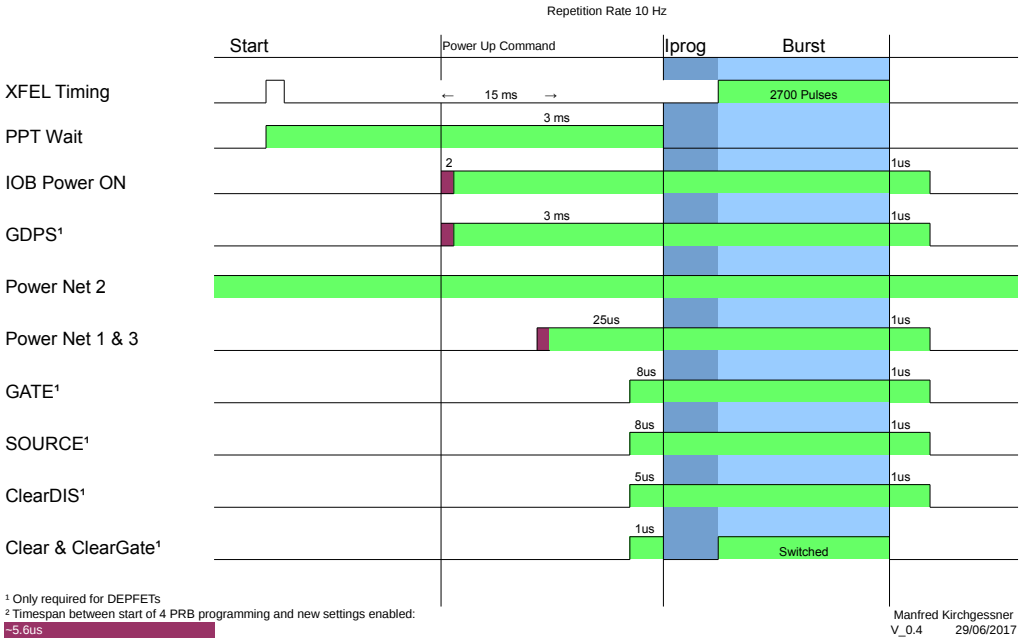


Figure 6.8.: **Detail view of the switched signals and their default timings between Start and end of Burst.** The duration of each active signal can be adjusted in the IOB with a granularity of 10 ns. The reference time point for the sequence and all computations for the various counter values is defined by the estimated time point when the first photon pulse arrives. The bottom four signals are required only for DEPFET sensors.

The high voltage levels of the CLEAR signals can not be switched by the IOB. For this purpose a special Gate Driver (GD) PCB can be soldered on the PRB. The GD provides circuits which allow switching of voltages of up to 25 V. The **G**ate **D**river **P**ower **S**upply (GDPS) signal which enables the GD functionality is also switched by the shift register on the PRB.

If an MSDD sensor is mounted, the CLEAR and DEPFET signals are not connected.

6.4.1 Summary

The synchronization of the different components of the DSSC detector is a difficult task and could only be solved in the firmware with a large number of counters controlled by various state machines. The global synchronization is realized by the fast telegrams, which distribute the coarse reference points to the different devices. The fine alignment is locally adjusted by small entities, which have to be configured correctly. In the final system a large number of different counters must be adjusted to realize the precise alignment of the switched signals to the image acquisition sequence.

Due to the complexity of the system, the compilation of the sequence and the determination of the values for all sequence counters in the PPT, the IOB and the readout ASIC can only be realized by the control software. For convenient usage, the software provides a minimum subset of the important timing constraints. The specified timing constraints have to be translated to the configuration values of the sequence counters. Thereby, the

reference point of all counters in the IOB is the start point at which the *Power Up* command is received. For the operator of the system, the reference point is always the start of the image acquisition phase. In the final implementation, the global sequence can be easily adapted by the control software by manipulating a few time constraints which in turn serve to configure the implemented arrangement of firmware modules.

6.5 Debug Features and Simulation Mode

Firmware development is always a very time consuming task, especially for such complex two level FPGA systems like the DSSC DAQ chain. During development, the single firmware modules have to be simulated to verify their behavior before they can be implemented in the real hardware[5]. The firmware simulation grew rapidly with the increasing number of implemented features of the system. The integration of the individual modules into the system and their behavior within the readout sequence required a comprehensive simulation which comprises the whole readout chain. This large simulation made it possible to monitor and evaluate all important details of the firmware interaction and is presented in the following.

Beside simulation also debugging of the signals in the FPGA is possible. Therefore, the Xilinx ChipScope application is used to directly access signals in the firmware logic.

For this purpose, the DSSC system provides certain debug and monitoring features which turned out to be very useful for firmware and software development. These features will be presented at the end of this section.

6.5.1 System Simulation and Simulation Mode

Verification of firmware modules is realized by integrating the modules into the large system simulation which comprises all central building blocks of the digital space of the detector:

- All devices controlled by EPC registers.
- The full data path in the PPT FPGA.
- A DDR3 memory simulation module.
- The IOB firmware.
- The digital part of the readout ASIC, including an SRAM module which delivers simulated data during the readout simulation.
- A simplified version of the SIB which answers to the commands from the PPT.
- A simulation dummy for the PRBs which reacts on the signals from the IOB.

Just the Microblaze is not included in the simulation, since its operation system can hardly be simulated. Nevertheless, in the real system, all configuration data for the EPC registers are delivered by the microcontroller. Consequently, a Verilog module is needed in

[5] The utilized simulation software for the Verilog modules has been the Incisive (NCSim) Functional Simualtor from Cadence.

the simulation which implements the configuration of the EPC registers to initiate firmware module behavior.

Another challenge of the simulation is the definition of an initial state, which represents a realistic configuration of the system. In the full system simulation, the very large number of configuration registers requires an automated sequence to generate the initial state. Especially the large number of JTAG registers of the ASIC have to be configured before the ASIC can operate at all, also in simulation.

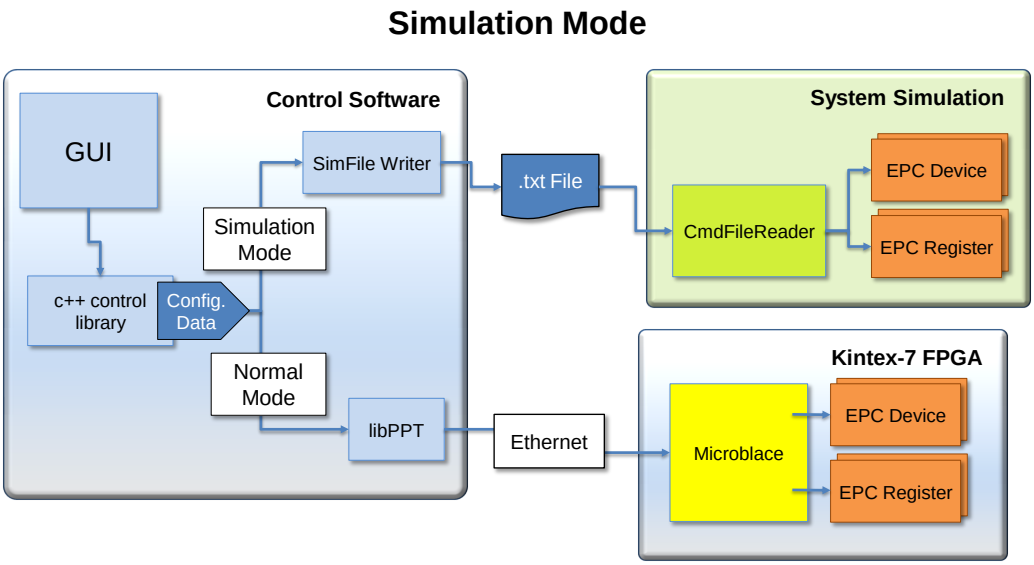


Figure 6.9.: **Simulation Mode.** By activating simulation mode int the control software all configuration data is redirected into a text file which can be used to configure the system simulation.

Therefore, the *CmdFileReader* module has been introduced. The module is able to read bytes from a text file and to inject it synchronously into the system simulation as visualized in figure 6.9. Effectively, in simulation the *CmdFileReader* replaces the Microblaze and reproduces the configuration interface of the EPC registers. In this way, the whole detector system simulation can be initialized and controlled from a text file.

However, the generation of this text file has also to be automated as much as possible. This has been solved already in a very early state of the project: Already in the control software of the ASIC Test System a method could be introduced which automatically redirects the configuration data to a text file. In this way, large numbers of configuration Bytes can easily be provided to the simulation in the identical order like it is sent to the real system[6].

Because of its high value for firmware development, this mode is also implemented in the control software for the real detector system resp. the PPT Test Setup. Unfortunately, things are more complicated in this environment because the control software does not communicate directly with the EPC registers but via Ethernet and the *SlowControlServer* (see section 9.2) application which runs on the Microblaze. Most of the functionality of the *SlowControlServer* has also been integrated into the simulation mode in order to allow

[6]This simulation mode can even be used to find certain issues in the control software

control of the large system simulation from the control software.

6.5.2 FPGA Firmware Debugging

In order to monitor the status and the behavior of the firmware modules implemented in the real system, the Xilinx FPGAs provides a possibility to access and monitor selected signals and registers during operation. This is done by connecting special hardware blocks of the FPGA to the desired signals. Using the Xilinx Chipscope Analyzer [91] application, it is possible to monitor the behavior of these signals with a graphical user interface. A screenshot of this application is depicted in figure 6.10.

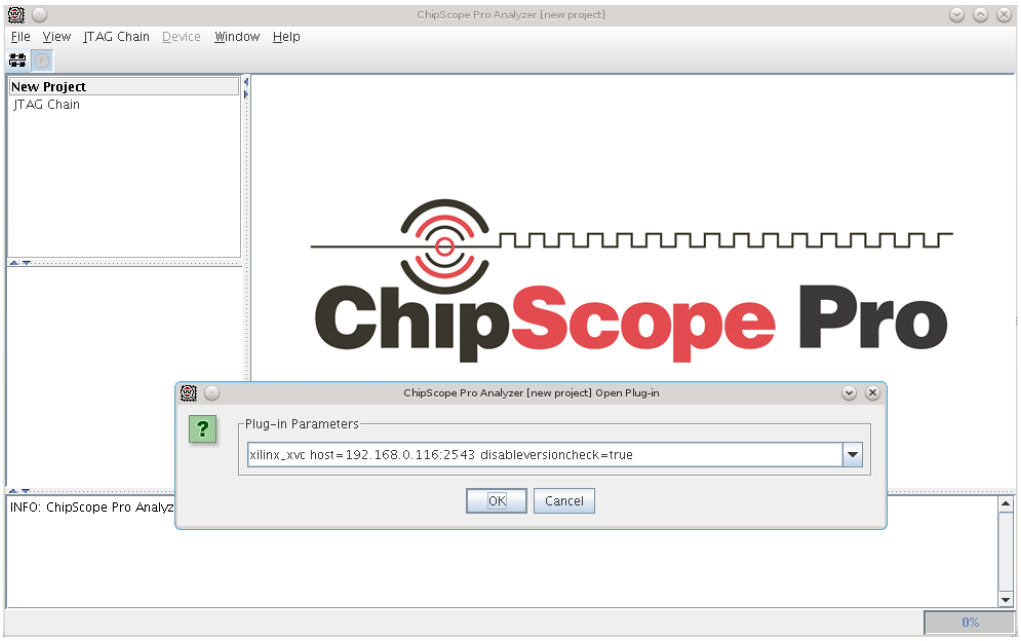


Figure 6.10.: Screenshot of the Xilinx Chipscope Pro Analyzer[91].

The direct access to signals in the FPGA logic is realized by integrating logic analyzers (ILA) directly into the target design. By connecting a special debugging cable directly to the JTAG signals of the FPGA, the ILA can be accessed from a computer. However, in a multi-stage FPGA system, the direct access to these signals is not given for every device. Especially in the DSSC system, in which the IOB FPGA is operated in vacuum where there is no way to connect a debugging cable.

Since the IOB FPGAs is programmed from the PPT FPGA, their JTAG signals are connected and accessible from the Microblaze. For this configuration, Xilinx provides a tool implementing a TCP/IP-based protocol which can be used instead of a physical debugging cable. The Xilinx Virtual Cable (XVC) [92] protocol can be used to access the debug features of a distant FPGA within an embedded environment. In figure 6.11, the connection scheme of the FPGA debugging signals of the DSSC system is shown.

By connecting the JTAG signals of the Kintex-7 FPGA on the PPT to general purpose I/Os, the XVC can be used to access the debugging features of the IOB FPGAs and even of the Kintex-7 FPGA itself.

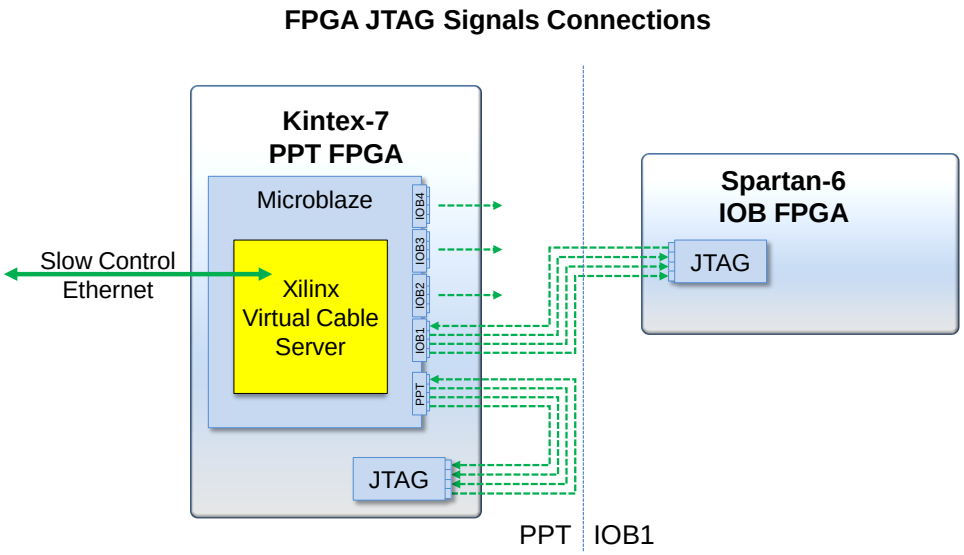


Figure 6.11.: **Connection of the JTAG signals of the system FPGAs.** By running the Xiling Virtual Cable Server on the Microblaze, the JTAG interface of the IOB FPGA and even of the PPT FPGA itself, can be accessed. Via the Ethernet connection of the PPT, the Xilinx Chipscope Software can connect to the XVC server and access the debugging logic inside the IOB FPGAs from remote.

The XVC protocol is implemented by a custom server application which runs on the Microblaze. The ChipScope application can connect via Ethernet to the server as if it was a physical cable. Also the usage of the application is identical. The XVC server for the Microblaze has been implemented by Jan Soldat.

In the current implementation of the PPT and the IOB firmware, ILAs (Integrated Logic Analyzers) are inserted at several positions. The main state machines and important parts of the data path can be monitored. Additional ILAs are inserted in controllers of the IOB. Since the debugging features do not slow down the operations in the firmware there is no need to remove them from the final design.

There exist also another tool in the ChipScope Pro Serial I/O Toolkit, allowing measurements of the transfer quality of the high-speed links of the DSSC system. This Integrated Bit Error Ratio Test (IBERT) can easily be integrated into a firmware design and connected to the respective high speed transceivers. It allows also injection of bit-errors on purpose for testing. The evaluation of the IBERT measurements is implemented in the ChipScope application which provides a graphical interface for the IBERT IP-core. The tool is used to measure the bit error rate of the serial connections of the data path from the IOB to the PPT and of the 10 Gigabit-Ethernet links of the QSFP+ connection. The measurements and results are described in section 8.

7

Implementation of the Readout Chain

This chapter describes the implementation of the high performance readout chain [93] which transfers the image data from the in-pixel memory of the readout ASIC to the fast 10 Gigabit-Ethernet links of the Patch Panel Transceiver. Since the data path of one quadrant can be regarded as an independent system, all data rate computations and discussions refer to the quadrant system. However, all computations can easily be translated to the full megapixel camera which consists of four quadrants.

By implementing modern components, the PPT has become a very compact board, providing at the same time enough performance to deal with the large amount of image data which is produced by the quarter megapixel camera during each incident X-ray pulse train. An average data rate of roughly 34 GBit/s is transferred which is a constant utilization of 85% of the available 40 GBit/s bandwidth during continuous operation. At these high data rates, the data stream is not only packaged into Ethernet packets and transferred but also image-wise resorted. The specification prescribes that the images in the output stream are sorted in the same chronological order as they have been acquired. This is also realized in the data path within the Kintex-7 FPGA.

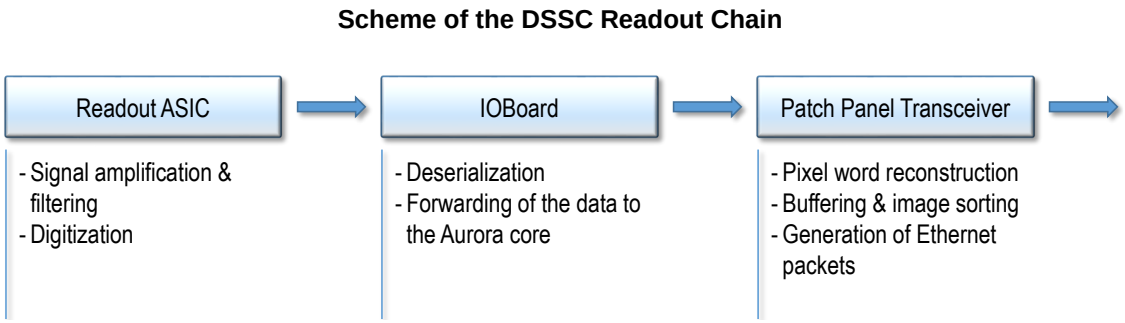


Figure 7.1.: **Scheme of the readout chain of the DSSC detector.** It is implemented in three hardware levels. The major tasks of the modules are summarized.

The readout chain comprises the path of the data which starts at the 64 channel parallel readout from the quadrant ASICs over the FPGA-to-FPGA connection between the IOB and the PPT, the image wise reordering up to the generation of the Ethernet packets. Each

component and task has to provide at minimum the bandwidth of the previous one in order to not stall the whole path and lose data. The highest demands are placed on the DDR3 memory, since the data must be written into the buffer and read out again during the same cycle. Only by using the DDR3 memory in this way, it is possible to change the order of the images in the stream without interrupting the transmission. The management of the different tasks requires several interlinked state machines which have been presented in the previous chapter.

At the output, the readout data has to be transferred in a given, standardized format, which is common for all three megapixel detectors. The specification comprises a defined format for the single Ethernet packets and also a certain format for the Train Data in total. The description of the Train Data Format can be found in section 4.2.1.

The scheme of the readout chain in figure 7.1 summarizes the major tasks.

The high data rates that are required to transfer all data out of the system have been the critical factor for all implementations in the data path and will be discussed at the end of the chapter. Another critical factor is the data transfer quality between the different devices. All data transfer connections have been measured and their bit error rate has been evaluated. The results are presented in chapter 8.

7.1 Readout Concept of the ASIC

The 64 x 64 pixel readout ASIC implements an in-pixel SRAM memory to store the 9-bit ADC values of 800 images. During the readout phase, the full image storage has to be read out and shifted through a serializer. A detailed description of the ASIC readout scheme can be found in [73].

Generally, the ASIC readout scheme was designed with the aim of simplicity and robustness. Therefore, the pixel matrix is readout in parallel which avoids fast clocks in the shift registers. The shift chains in the ASIC transport the pixel data from the top of the chip to the bottom and from the sides to the center, where the serializer is located. Consequently, the pixel data is sent out in reverse order for the left half of the ASIC.

The serializer is implemented as a 10-bit shift register and transmits data at a frequency of 350 MHz. The 10th bit of the serializer is implemented as a logical XOR of the valid data bits which is used as a checksum to verify correct data transmission for every single 9-bit data word.

At the serializer output the image data is transferred to the IOB via LVDS signals.

7.1.1 ASIC Trailer Words

Besides the pixel data, the readout ASIC sends out 9 additional words, which are also appended to the Train Data in the data stream. The first additional word is transferred at the beginning of the readout. It describes a configurable bit pattern and can be used to check for correct data transmission from the ASIC. The other eight words are transferred after the last pixel word. These words are listed in the following table 7.1.1.

The readout ASIC comprises a temperature measurement circuit[1] which is sampled two times during the data acquisition phase. The first measurement is taken directly before

[1] A diode and a distinct ADC

Word Number	Valid Bits	Description
0	9	Temperature0 8:0
1	9	Temperature1 8:0
2	9	VetoCnt 8:0
3	4	VetoCnt 12:9
4	9	JtagRegsXorOut 8:0
5	9	JtagRegsXorOut 17:9
6	3	JtagRegsXorOut 20:18

Table 7.1.: **The 7 ASIC trailer words, which are sent out at the end of each readout after the pixel data.**

data acquisition, and the second measurement is started with the readout command. These two temperature values are the first two words of the ASIC trailer, followed by the veto count. The Veto count is generated in the digital part of the readout ASIC and can be used to check the Veto acceptance of the ASIC. The last three words of the ASIC trailer are the checksums of all configuration bits of the ASIC. All configuration registers include an XOR chain, including the pixel registers (192512 cells). Using this XOR chain the ASIC can send out a checksum over all configuration bits which signals unexpected single bit-flips.

The eight words are transferred per ASIC and appended to the image data in the detector specific part in the Train Data.

7.2 The IOB Data Receiver

The FPGA on the IOB defines the first FPGA state of the readout chain. In this stage the data from 16 readout ASICs is sampled and concentrated into one data stream which transports the data to the PPT. At the input, the IOB receives a total input rate of 16×350 MHz $=$ 5.6 GBit/s via 16 LVDS connections on general purpose I/Os. At the output, a triple link Aurora[94] channel is implemented to transfer the data via three 3.125 GBit/s high speed links to the PPT.

7.2.1 The Aurora Protocol

Aurora describes a light weight serial communication protocol from Xilinx which is used for high speed data transfer e.g. from one FPGA to another. The Spartan-6 FPGA on the IOB supports data transfer rates via the Aurora protocol of up to 3.125 GBit/s. It allows to combine multiple links to one logical channel for synchronous data transfer on several connections. By using an 8B/10B core single-bit and most multi-bit errors can be detected. The protocol is implemented in the system in simplex and streaming mode[2]. The Aurora IP core provides in streaming mode a simple interface, which shows a 96 bit wide parallel data input running at 78.125 MHz. The data transfer is simply initiated by assigning a 'valid' signal. Additional logic is required to deal with the short phases in which the Aurora core is synchronizing to the receiver side from time to time. During these phases the core

[2]Also duplex and frame based modes are available.

signals that it is not ready and data transmission has to be stopped for a few cycles. The synchronization phases last usually not more than 10 clock cycles and reduce the maximum average throughput of the Aurora link by less than 0.5%.

7.2.2 The IOB Data Path

The data path in the IOB implements mainly two tasks. Recovery of the serial data which is received from the readout ASICs and the transfer of the 16 individual data streams to the parallel interface of the Aurora core.

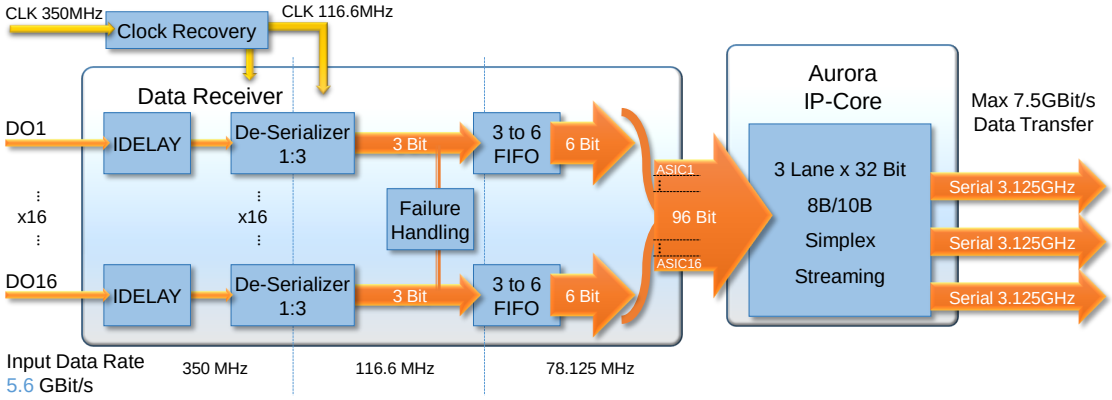


Figure 7.2.: **The IOB data path.** Recovery of the serial data from the 16 ASIC channels is realized by 16 data receiver modules. Each module implements a 1-to-3 'ISERDES' deserializer which implements distinct logic cells in the FPGA I/O region. The 3-bit stream is filled into a 3-to-6 FIFO that provides a buffer and implements the clock domain crossing. To compensate for misalignment of the incoming data to the data clock, each channel provides a distinct delay module.

The data path implementation is depicted in figure 7.2. The most critical part is the efficient transfer of the ASIC data to the Aurora IP-core. Therefore, the whole concept of the IOB data path is mainly driven by the scheme which is used to distribute the data from the 16 ASIC channels to the 96 bits of the Aurora interface. During readout, a constant stream of 5.6 GBit/s has to be transferred to the Aurora core which implements a streaming interface. The logical consequence is to simply stream the input data to the output. By assigning always six bits of the Aurora interface to one particular readout ASIC, the input data streams can continuously be propagated to the Aurora core. In order to provide a buffer for short transmission pauses and to cross the clock domains, the recovered ASIC data is filled into a 3-to-6 FIFO. The combination of the 1-to-3 deserializer and the 3-to-6 FIFO is a robust and simple way to recover the ASIC data compared to a much more complex 1-to-6 deserializer. Since the maximum data rate of the Aurora core of 7.5GBit/s is larger than the input rate, eventually accumulated data in the FIFOs can quickly be propagated to the PPT shortly after the pause.

7.2.3 Channel Failure Handling

The 3-to-6-FIFOs are also used to synchronize the input streams: if of one ASIC channel sends data on a different clock edge than the others, the FIFOs compensate any timing differences. By NOR-ing all 16 'fifo_empty' signals as the valid flag for the Aurora interface, the FIFO stage guarantees that the data is synchronously passed to the Aurora interface. On the other side, the whole data path could be blocked if one ASIC fails and doesn't send any data. Therefore, a mechanism has been introduced which detects problems and generates dummy data in case of an ASIC failure. The generated data from the broken channel is then propagated to the following data path as if it was real data and consequently no further logic is required to handle this failure. The dummy data generation can also be activated on purpose in order to simulate readout data during testing.

The failure of a single channel in a Ladder does not occur rarely as it has been noticed during frequent use of different Ladder systems. A failure can be caused by broken ASICs that do not deliver data at all for any reason or by incomplete initialization of single readout ASICs. The later can be fixed automatically from software in order to complete the initialization. Consequently, the number of ASICs which actually send out data can change in exceptional cases from train to train. The automated error handling logic allows fast failure recovery and provides stable system operation during readout and also until the system is correctly initialized. If an ASIC is once brought into working state, the problem does not occur again as long as the ASIC remains powered.

7.3 The Data Path in the PPT FPGA

The Kintex-7 FPGA on the PPT is the heart of the DSSC detector system in terms of control and data acquisition. The small format of the PPT can easily lead to underestimate the performance of the device. Latest FPGA technology enables the small format board to handle the high average throughput of approximately 34 GBit/s which is continuously generated during the pulsed operation at 10 Hz.

At the data path input, the high speed links of the four IOBs enter the FPGA implementing the Aurora protocol. From the continuous input data streams, the 10-bit data words have to be reconstructed from four Module channels combining 64 ASIC channels. Before the image data can be sent out of the system, it has to be brought into the Train Data format. Therefore, the address controller of the large DDR3 memory manages the storage of the header, trailer and additional metadata to the correct addresses in the memory. In a similar way, the address controller implements the image reordering during the transfer of the data to the memory. After all image data and additional meta data have been written into one continuous memory block, the Train Data is readout and packed into the Ethernet packets which transport the data out of the system via four 10 Gigabit-Ethernet links combined within one single QSFP+ cable.

In the following, the implemented data path IP-cores are introduced and the basic functionality of the corresponding controllers are presented.

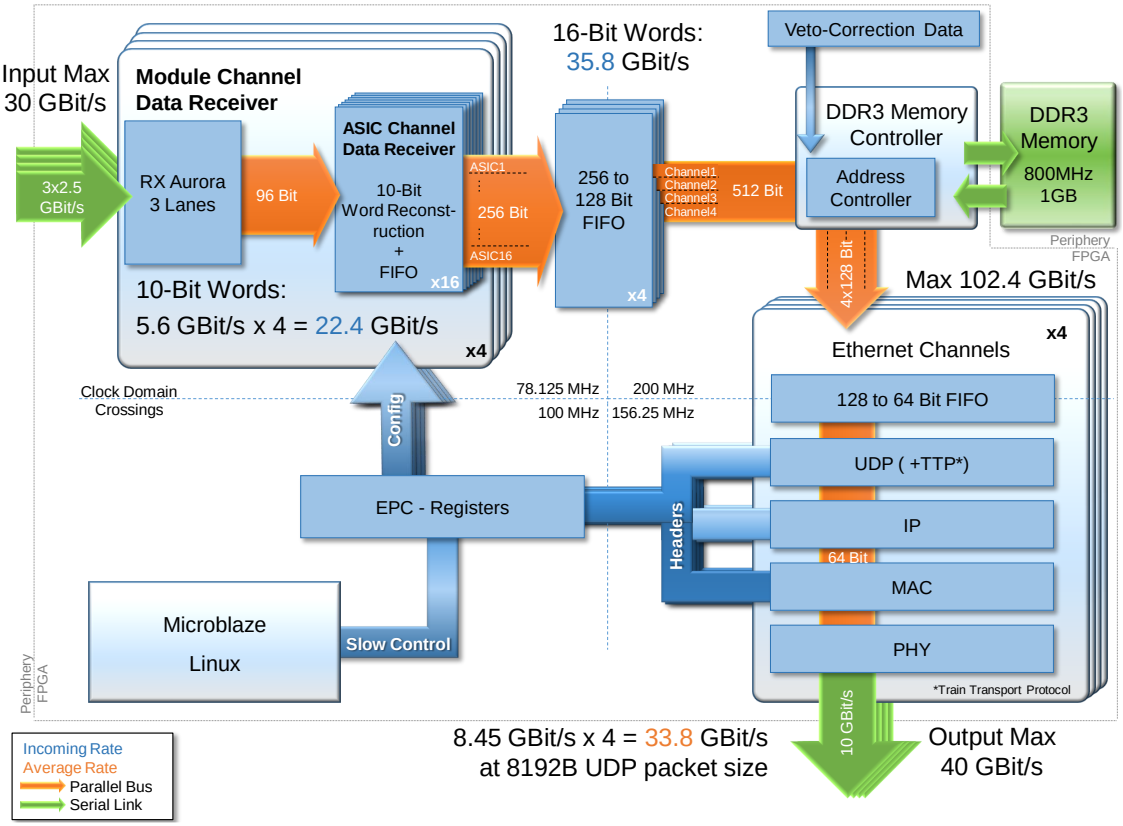


Figure 7.3.: **The PPT Data Path.** The data streams of four Aurora channels from the IOBs is received and the pixel words are re-constructed. In the DDR3 memory the Train Data is generated before it is packed into standard UDP Ethernet packets and transferred via the four 10 Gigabit-Ethernet links. Each module channel can be activated individually. The image re-ordering is implemented in the DDR3 address generator while the images are written to the memory.

7.3.1 General Remarks

Each of the three main tasks of the data path include a central IP-core implementing the connection to the external. The implementation and the integration of the generated module into the data path always follows a similar pattern:

1. Specification of the IP-Core parameters
2. Integration of the IP-Core into the data path through a wrapper module
3. Assignment of configuration registers
4. Verification of the reset and initialization sequences using simulation
5. Synchronization of the firmware module to the corresponding readout sequence
6. Implementation of additional features like the image re-ordering or Ethernet header generation

Even if 1. - 5. take also a large amount of time and are no simple tasks at all, the descriptions in the following sections will deal mostly with the last point. Especially, 4. has been the most challenging item before each of the modules could be used at all since the initialization sequence need to be implemented carefully in order to bring the IP-core automatically and reliably to operation after power up or after a reset of the system.

The Aurora IP-core in the PPT Data Receiver implements the counter part to the IP-core in the IOB. It provides the same parallel interface which outputs the data in a 96-bit wide bus and a maximum bandwidth of 7.5GBit/s. For each of the four connected IOBs, one Data Receiver channel is instantiated.

7.3.2 About the IP-core Implementation

The DDR3 memory controller implements the communication and control for the memory hardware. To operate the high speed memory devices, several signals require precise relations. Therefore, the delay and setup timings for various signals have to be correctly defined. Various timings of the control signals have to be specified according to the implemented memory type which can be derived from the device documentation. Using the correct timings, the IP-core generator can create the memory controller such that it fits to the implemented memory device. In order to integrate the memory controller into the data path, an address generator and a wrapper module have to be developed. To supply the high bandwidth of the DD3 memory, the interface runs at 200 MHz and provides a parallel interface of 512 bits. In total, the memory controller provides a data rate of maximum 102.4 GBit/s during burst operation which allows to read or write up to 256 words during one burst operation. Since the image data has to be written and read again in the same cycle from the memory, the high bandwidth is really required.

The last controller in the data path implements the interface to the 10 Gigabit-Ethernet links. The connection of the **PHY**sical layer of the high speed connection is implemented by a so called PHY. Depending on the performance of the FPGA, an external IC is required for this layer. However, the Kintex-7 FPGA is able to implement the PHY for the 10 Gigabit-Ethernet links internally. Actually, the physical layer is implemented in three combined

modules: the Physical Coding Sublayer (PCS), the Physical Medium Attachment (PMA), and the Physical Medium Dependent (PMD) sublayers comprise the physical layers of the Ethernet protocol[95] in the Kintex-7 FPGA.

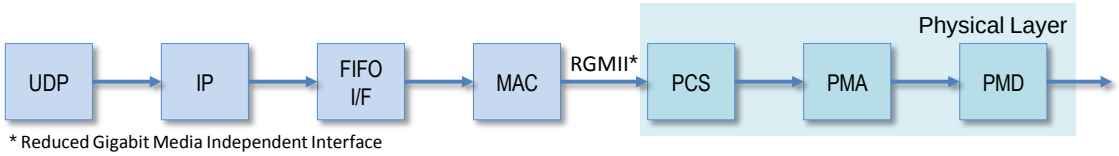


Figure 7.4.: **Firmware modules of a typical Ethernet interface in an FPGA.** For packet generation the header fields of the UDP, IP and MAC protocols have to be added to the payload data. The physical layer implements the high speed serializer in several modules and are connected by the MAC module.

The PHY IP-core is not directly connected to the data path. Via a parallel RGMII[3] interface, the PHY is connected to a MAC (Media Access Control) IP-core which provides a 64-bit parallel interface for integration into the data path. The MAC interface runs at 156.25 MHz which computes to an effective bandwidth of 10 GBit/s. Both IP-cors, the MAC and the PHY, are integrated into the Ethernet Engine which provides data buffers and several modules generating the data packets by extending the data stream with the required UDP, IP and MAC headers. Also the trailer data for the TTP format is inserted. The generation of the Ethernet packets is presented in section 7.3.5.

7.3.3 Overview

In figure 7.3 the building blocks of the PPT Data Path are shown. The data streams of the four Ladder modules are processed separately almost individual data path channels which only come together in the DDR3 memory. Each channel can individually be enabled in order to support the operation of one to four camera Ladders. Consequently, each of the four 10 Gigabit-Ethernet links of the QSFP+ transceiver is assigned to one distinct Ladder module.

The data from the four Aurora channels is received in four individual Data Receiver modules which assign always 6 bit of the 96-bit interface to one ASIC channel. One Data Receiver module splits into 16 individual ASIC channels which implement the 10-bit word reconstruction from the continuous 6-bit wide input stream. The reconstruction is realized by a barrel shifter whose shift offset is determined by the first '1' in the stream which defines the beginning of the data transmission. At the output of every ASIC Channel Data Receiver the 9-bit binary value which is decoded from the 9-bit Gray-code pixel value is stored in a FIFO.

In the next step, the 9-bit words are extended to 16 bits and stored in the 256-to-128-bit FIFOs. The output of these FIFOs are combined to one data stream which is then transferred to the large DDR3 memory. The 256-to-128-bit FIFOs are used to buffer the incoming pixel words during read phases of the memory controller or before each write cycle while the DDR3 controller is preparing the memory access. The preparation phases before each

[3]Reduced Gigabit Media Independet Interface

memory access decrease the bandwidth of the memory. Therefore, the DDR3 controller supports a burst mode which allows to write or read up to 256 words within one cycle.

At this point of the datapath the pixel word size is increased from 9 bit to the specified Train Data word size of 16 bit. The increase almost doubles the required bandwidth of the further modules in the data path. Nevertheless, the DDR3 controller as well as the Ethernet Engine can handle the increased data rates. During alternating read and write cycles of maximum burst length a maximum bandwidth of over 88GBit/s can be reached which is sufficient to handle incoming data.

Furthermore, from the implementation point of view the 16-bit pixel words provide certain advantages: 16-bit words can nicely be combined to fill the 512-bit interface of the memory controller as well as the 64-bit interface of the MAC IP-core. The integer relation simplifies the word counting and also the filling of the Ethernet packets a lot.

In order to simplify the memory addressing during the readout from the DDR3 memory, all data is stored in one continuous memory block, which can be read out straight forward from start to end during the secondary readout phase. Due to the image-wise reordering, the readout of the image data from the memory cannot start before the complete train has been stored.

During the DDR3 read phase, the data of all four channels is read out synchronously from the memory. Always 128 bits of the output data from the DDR3 memory is filled into the large 128-to-64-bit FIFO at the input of the Ethernet Engine. Each input FIFO is able to buffer the data of up to two whole Ethernet packets. Because the MAC IP core does not allow to stop the incoming data stream during packet transmission, the generation and sending of an Ethernet packet cannot start before all packet data is available in the input FIFO. The input FIFO implements also the clock domain crossing to the Ethernet clock and the buffering of the data from the fast block reads from the memory.

During a read burst, the high bandwidth of the DDR3 interface brings a risk to overload the Ethernet Engine. In order to control the output data rate of the memory controller a throttle has been introduced which allows to reduce the output data rate of the DDR3 memory and thereby also the average output data rate of the detector. The throttle can also be used in order to provide compatibility to Ethernet receiver devices with lower performance. For example the number of images which are contained in a train can be reduced by configuration. By increasing the value of the DDR3 read throttle, the duration to transfer the decreased number of data can be stretched to fit the available time between two readout cycles. In this case the transfer rate of each data packet remains at 10GBit/s but the average data rate is reduced due to the inserted pauses between the single Ethernet packets.

7.3.4 Image Reordering in the DDR3 Address Generator

The sorting of the images into chronological order is prescribed from XFEL in the specification of the Train Data format. Due to the Veto mechanism, which enables the readout ASIC to overwrite already assigned memory cells, it can happen that the order of the images in the ASIC memory is not chronological. However, the order in which the images are read out of the ASIC cannot be changed and always runs from address 0 to 799. In order to restore the chronological order of the images in the output stream, the DDR3 address controller is

designed to reorder the images during readout. Therefore, the controller has to be supplied with the Veto-Correction data.

In this section the general concept of the memory management is presented and the algorithm, which generates the Veto-Correction data is described.

Memory Management in the DDR3 controller

The PPT implements four 256 MB DDR3 memory devices, which can store 1 GB of data. Since one Quadrant data train of 800 images contains roughly 400 MB, the size of the buffer is sufficient to buffer more than two complete data trains. The idea is to write all data of one train, including train header, image data and additional meta data in chronological order in one connected memory block from where it can be read out consecutively as depicted in figure 7.5. Therefore, the address space of the memory is divided into two 512 MB memory blocks which are successively filled and read out.

The area where the image data is stored is subdivided into smaller 512 KB areas which are filled with the data of one image. Due to the extension of the pixel data size to 16 bits, the image areas can be filled seamlessly and additionally the addressing can avoid complex arithmetical operations. By providing the information in the FIFO where the next image in the incoming data stream is to be stored, the images can be sorted in chronological order. However, this can only be realized by introducing image oriented write phases, which leads to shorter write cycles. But short write phases are costly because of the preparation overhead before each write cycle. On the other side, the memory controller has to provide high data rates to deal with the high rate of incoming data. To solve this, the write controller dynamically determines the number of cycles for the next write burst from the number of available words in the 256-to-128-bit FIFO and with respect to the remaining data of the currently transferred image. The implemented writing scheme automatically compensates short write-bursts at the end of the image transfer, with longer write bursts at the beginning of the next image without loosing too much data transfer rate. This scheme is also critical to provide the high bandwidth of the memory controller when each write burst is followed by a long read burst, while the readout of the previous train has not finished and the next readout cycle has already been started.

Since the Train Data is already stored in the correct format, the readout of the Train Data area is realized straight forward from start to end. The length of the read bursts is adapted to the size of the Ethernet packets. The maximum length of a read burst is 256 words. Each word contains 512 bits, which are equally distributed to the four Ethernet channels. Consequently, 16 Bytes can be transferred from the memory to the Ethernet Engine per DDR3 word or in other words, one 8KB Ethernet packet is filled after four read bursts.

Generation of the Veto-Correction Data

When the readout ASIC receives a Veto command during the *Burst* state, its address generator re-uses a former assigned address for the next write process. The address which is overwritten is determined by a shift register of (statically) configurable length. For every write operation, the address is inserted into the shift register. The output of the shift register consequently presents the address which was written to several cycles before. Therefore, the Veto command has to be sent at the right moment when the to be vetoed address pops

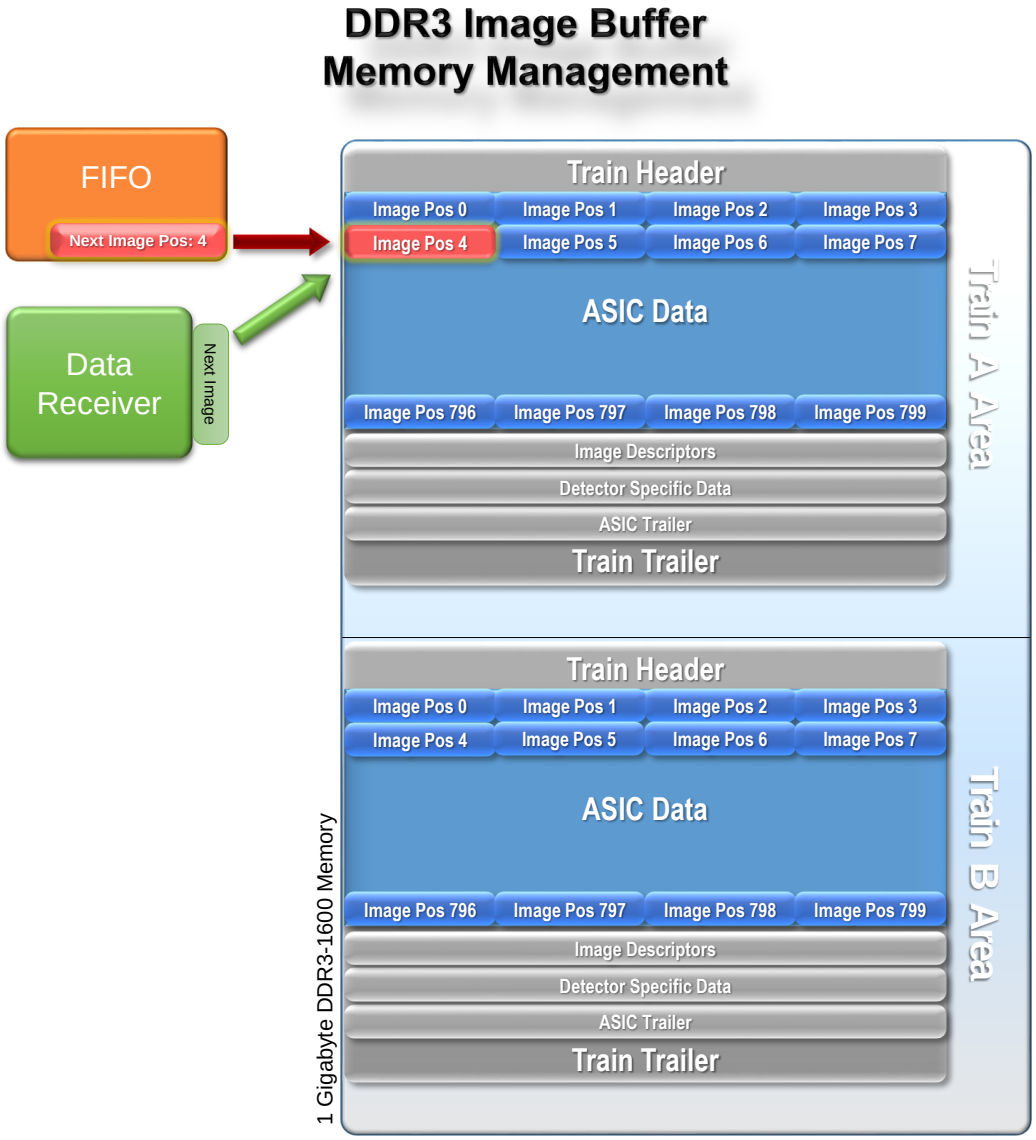


Figure 7.5.: **DDR3 Memory Management.** The 1 GB DDR3 memory on the PPT is divided into two Train Data blocks, which are filled alternately. The ASIC data region is subdivided into the image areas, which seamlessly store the image data. Due to the pixel word size of 2 Bytes, the addresses computation of the single image regions is fairly simplified.

out of the shift register. The length of this shift-register is the so-called Veto-latency and defines the maximum response time of the Veto units.

The final order of the images and their corresponding pulse-ID in the ASIC memory, and thereby in the data stream, cannot be retrieved from the incoming veto commands alone. In order to generate the final assignment of SRAM addresses and Pulse-IDs, the address controller of the readout ASIC is mirrored in the control firmware including the Veto decoding unit. The internal SRAM address controller receives the Veto commands synchronously to all real readout ASICs and therefore is able to reconstruct the pulse-ID of each image in the data stream. A simple example pattern is shown on page 104. Because a once-reused SRAM cell can also be vetoed multiple times, much more complicated patterns are possible.

The internally generated information can then be used to generate the re-ordered image positions for the DDR3 memory controller (Veto-Correction data). During the few milliseconds that are available between the image acquisition and the long readout, the generated information is sorted and processed within several stages of RAMs and FIFOs in order to get at the end the correct sorting scheme as shown in figure 5. The task of the algorithm is to restore the correct order and the assignment of the image number to the position of the image area in the DDR3 memory. The implementation of this algorithm is shown in figure 7.6.

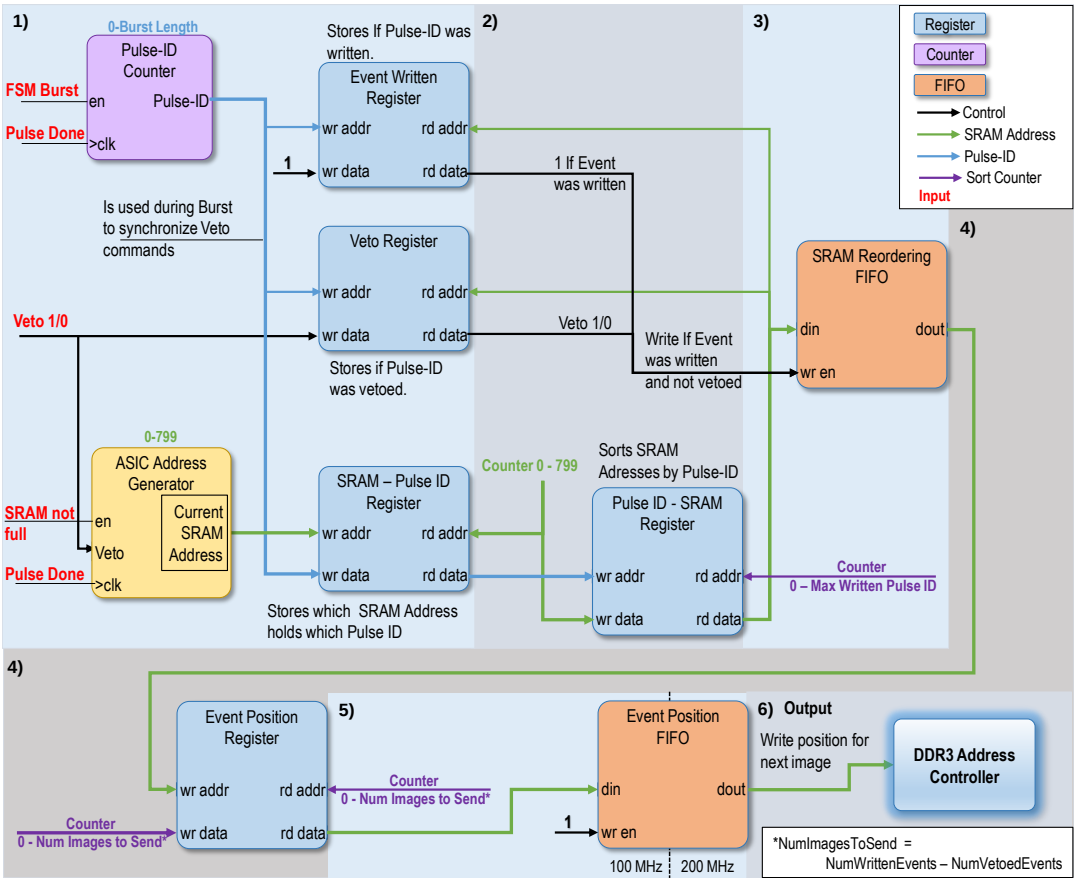
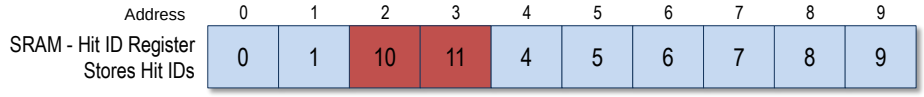


Figure 7.6.: **The Veto-Correction algorithm.** The aim is to generate for each image in the input stream its final position in the output stream, the Veto-Correction data. Using several stages of RAMs and FIFOs the Pulse-ID and SRAM addresses are sorted and arranged to get the required information. Within steps 1) - 3) the non vetoed Pulse-IDs are identified and assigned to the corresponding SRAM address and thereby its position in the input stream. 4) From the input position the final position in the output stream is determined and 5) provided to the DDR3 Address Controller in a FIFO.

The algorithm works in five consecutively executed tasks:

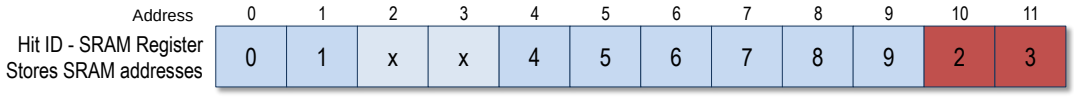
1. The first task is active during image acquisition when the ASIC is in *Burst* state. The **internal ASIC Address Generator** mirrors the behavior of the readout ASICs and generates identical addresses to store the actual Pulse-IDs at the corresponding address in the **SRAM-Pulse-ID Register**. In this way the final assignment of the Pulse-IDs to the SRAM addresses can be retrieved from the register directly after the ASIC memory is filled.

   Assuming events 2 and 3 have been vetoed and the Veto-Latency is 8 the following pattern would have been generated:



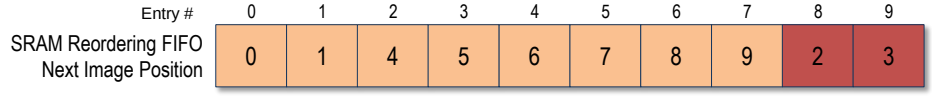
2. All Pulse-IDs from SRAM address 0-799 are read from the register. By storing the address to the data and vice versa, their assignment is inverted and the SRAM addresses are stored in the next register, sorted by their pulse-ID.

   In the example, the pattern would look like the following. Because of the Vetos, the values at address 2 and 3 have not been written and remain undefined.

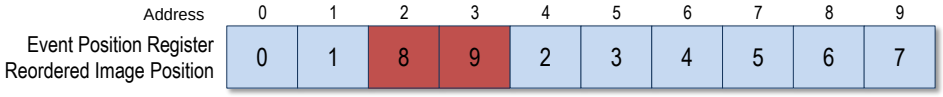


3. The pulse-IDs of non vetoed and only actually written events[4] are written into a FIFO. This FIFO then stores the position in the data stream of the image that should be written to the DDR3 memory next. Or in other words, it keeps the positions of the images in the input data stream sorted by their DDR3 memory image area which is determined in the next step.

   The entries of vetoed and also of not written events are removed:

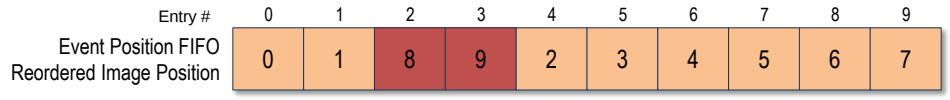


4. The assignment of position and entry number is inverted and stored. The register then holds the Veto-Correction data, or the information where in the DDR3 memory the respective image is to be stored.



[4] Not written events occur if the 800 memory cells of the readout ASIC are filled before the last X-ray pulse has been arrived.

5. The Veto-Correction data is filled in the **Event Position Fifo** which provides the data to the DDR3 address generator. By storing the data in a FIFO it can easily be retrieved in the right order from the DDR3 memory clock domain. Also the FIFO automatically indicates when it is empty which can be used to detect the end of the data stream.



Because the Veto-Correction information is required during readout, the readout sequence can only start when the first Veto-Correction word is stored in the **Event Position FIFO**. Consequently, the reorder algorithm introduces a small delay after the image acquisition phase of maximum 52 μs before the data readout can be triggered.

Verification of the Veto-Correction

A comprehensive verification of the Veto-Correction Algorithm is required to make sure that it works reliably and does not mix up images in the final image stream. In order to verify the presented modules, a system simulation has been setup in which the memory has been filled with generated, random Veto patterns. At the end of the simulation, the reordered image numbers are printed in a file from where they can be read and evaluated after the simulation has been finished. 1000 image acquisition cycles have been simulated in combination with a large number of Veto commands. Up to 1000 Veto commands per cycle have been sent which leads to high fraction of double and triple Vetos for single SRAM cells. In order to simplify the verification, the pixel data has been modified to represent always the image ID. However, a pixel word has only 9 bits in the readout ASICs. At least 12 bits are required to cover the full image ID range from 0 to 2900. But since the pixel words are extended to 16 bits during the processing in the Kintex-7 FPGA, at least in this simulation, the upper 3 bits can be recovered in order to restore the full Image ID in the data for validation. At the output of the DDR3 controller, the image wise sorted data can be compared to the expected pattern.

The simulation contains all modules of the control logic and the data path from the readout ASIC up to the output of the DDR3 controller. At the output of the DDR3 controller, the final order of the image can be evaluated.

An SRAM simulation model has been included which implements the pixel SRAM cells of a readout ASIC. The simulation model is controlled by the digital module of the ASIC SRAM controller, which generates realistic memory patterns. Also the DDR3 and Aurora simulation models are utilized. The results of the simulation are presented in the following following.

In the first step, the number of detected Veto commands and the assignment to the vetoed events have been checked. In figure 7.7, the generated Veto commands are compared with the Vetoed events as they are assigned in the *Veto Register*. Since no dark green dots are drawn, all Veto commands have been correctly assigned.

In the next step, the order of the Veto scrambled image IDs, as they where read out of the *SRAM - Pulse ID Register* is checked against the expected order from the readout ASICs.

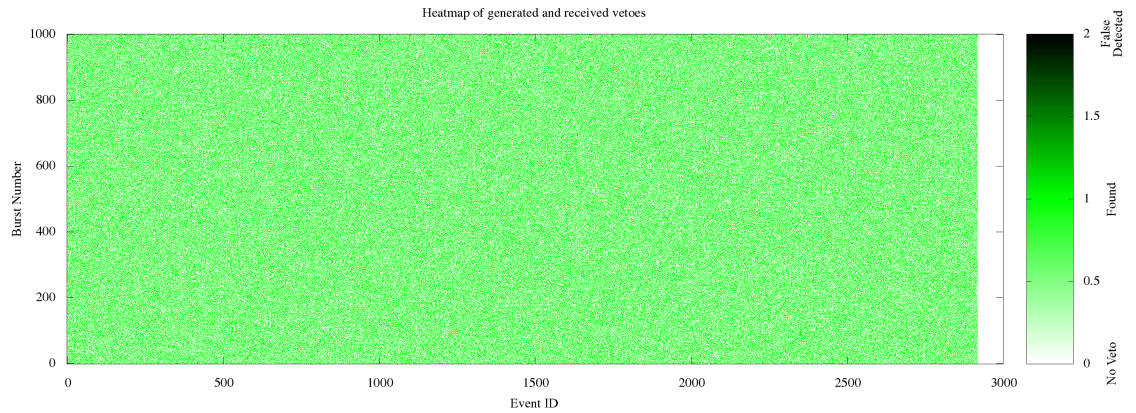


Figure 7.7.: Comparison of vetoed events and correctly detected Vetos in the Simulation. 1000 Bursts have been simulated and around 800 Vetos have been randomly generated per Burst. No incorrect assignments have been found.

Since both devices implement identical firmware modules no differences could be observed.

The scrambled image IDs of 1000 simulation runs are shown in the left part of figure 7.8. The scrambled order is clearly visible. Due to the specified Veto Latency of 80 images combined with the large number of Veto commands, large shifts can occur. For instance, a blue dot in the orange area indicates an SRAM cell which has been Vetoed over ten times.

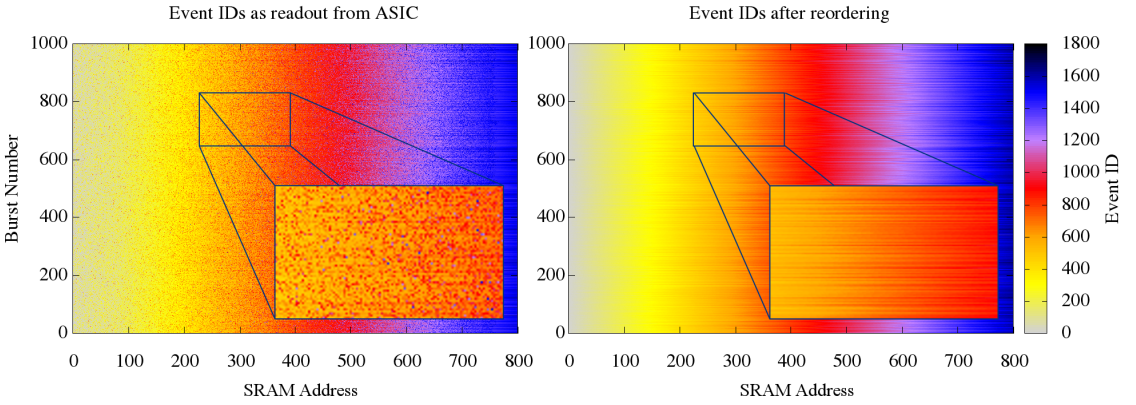


Figure 7.8.: Left: Veto Scrambled Image IDs as readout from the ASICs. Blue dots in red indicate SRAM addresses which have been vetoed twice, in orange even three times. Right: Image IDs after sorting.

On the right side of figure 7.8, the reordered Event IDs are printed like they are readout from the DDR3 memory. The line-shaped patterns result from the variable number of Veto commands in every run. The larger the number of Veto commands, the larger the spectrum of colors in the color map. Nevertheless, the color gradients are smoothly changing which signals a correct sorting of the images. The correct sorting of the data from the simulation is also successfully verified using a C++ script.

An automated verification in the real system has not been done, since there is no feasible

method to generate data which can be automatically and reliably assigned to a defined image Id. This would require a special data injection in the ASIC which is not existing. Nevertheless, during all manual tests, the correct functioning of the modules in the real system could reliably be confirmed. The verification can, for instance, be done by sending multiple test veto patterns and acquiring special data patterns which allow manual assignment of the data to a certain image ID.

Veto-Correction - Summary

The implemented algorithm provides a robust and reliable method to generate the image write positions for the reordering in the DDR3 address controller. The correct behavior of the firmware module has been verified in comprehensive simulations in which random Veto patterns could be generated and the correct re-ordering could automatically be checked.

Also extreme cases can be handled by the algorithm. If large number of Veto commands reduce the number of valid images in the ASIC memory, the data path automatically reduces the number of images in the Train Data, such that only valid images are sent out of the detector.

For debugging and testing, the Veto-Correction can be disabled. In this mode the *Event Position FIFO* has to be filled with up-counting positions according to the number of images in the Train Data.

7.3.5 Ethernet Packet Generation

The Ethernet Engine provides the interface to the fast 10 Gigabit-Ethernet links and generates UDP Ethernet packets from the incoming data stream. The data transfer is realized by the MAC and PHY IP-cores which pass the data to the high speed links of the FPGA. In four independently working channels the wrappers for the IP-cores and the packet generators are implemented. In order to process the data also in standard PC components the specification prescribes that the data at the output of the detector has to be packed into a standard format which describes UDP Ethernet packets. The format includes the Ethernet, IP and UDP headers which have to be inserted at the beginning of each data packet. The packet size is defined to 8192 KB payload including 8 trailer Bytes of the Train Transport Protocol (TTP). All packets beside the last one have to have the same size. The 42 Bytes of the Ethernet Headers and the TTP trailer Bytes are shown in table 7.2.

At the input, the Ethernet Engine is filled burst wise with the data from the DDR3 controller. Since the read bursts have a much higher bandwidth than the Ethernet Engine, the data has to be buffered in a 16 KB FIFO. Two complete Ethernet packets can be buffered. When all data for one packet is available in the input FIFO the packet generation immediately starts.

The modules of the Ethernet Engine are shown in figure 7.9.

The Ethernet headers are inserted by three packer modules: the UDP-Packer, which also adds the TTP trailer Bytes, the IP-Packer and the MAC-Packer, which implements also the wrapper for the MAC IP-core. The input interface of the MAC IP-core has a 64-bit wide bus which runs at 156.25 MHz. Consequently, the bus width of all modules in the Ethernet Engine is 64-bit. The transfer of the 128-bit input word to 64-bit word width is implemented in a 128-to-64-bit FIFO which also implements the clock domain crossing from the 200

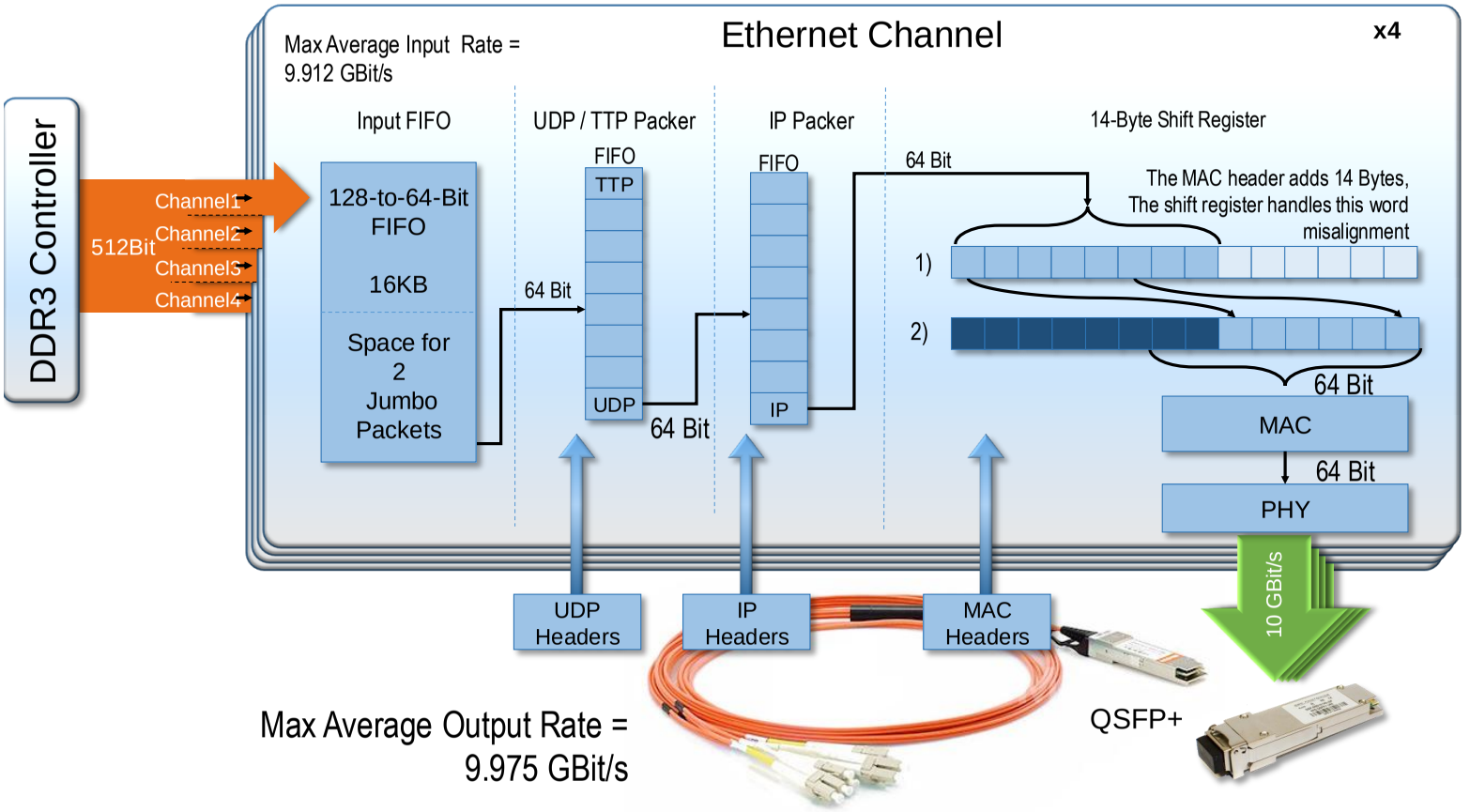


Figure 7.9.: **The modules of the Ethernet Engine.** In order to insert the Ethernet Headers into the data stream, each channel implements two FIFO stages and a 14-Byte shift register. The shift register in the MAC Packer compensates the misalignment of the 14-Byte MAC Headers to the 8-Byte Ethernet words. After the MAC header, with each clock cycle always 2 Bytes of the new word and 6 Bytes of the previous one are transferred to the MAC core.

	Bit0		Bit16		Bit 31	
MAC	Receiver MAC Address* (48 Bit)					
	Sender MAC Address* (48 Bit)					
14 Bytes	Ethernet Type (0x0800)*		-			
IP	Ip4=4|len=5	Service (0x0)	IP Pkt Length (auto)			
	Identi.= Packet ID (16 Bit)		flags| FO-2(0)	FO-1(0)		Flags =3'b000
	ttl (0xFF)	Proto 0x11=UDP	IP Header CRC (auto)			FO=Fragm.Offs.
	Source IP Address*					FO=14'b0
	Dest. IP Address*					
24 Bytes	Options and Padding = 32'b0					
UDP (8KB)	Source Port*		Destination Port*			*Programmable
8 Bytes	Length (8200)**		CRC (0x0) not used			**8184+8UDP
Data	Payload					+8TTP=8200
8184 Bytes	... total 8184 Bytes Payload					
TTP - Trailer	LSB	Frame ID (32 Bit)			MSB	
8 Bytes	LSB	Packet ID	(24 Bit) MSB	Resvd|EOF|SOF		

Table 7.2.: **Ethernet Headers and TTP Trailer Bytes.** Total 8236 Bytes per Ethernet packet, 54 Bytes overhead.

MHz DDR3 memory clock to the 156.25 MHz Ethernet clock. In the UDP and the IP packer modules a small FIFO is inserted to buffer the input stream during the header insertion at the start of the packet. This enables the different packer modules to operate independently from the following modules. The distribution of the IP and the UDP header insertion into two independent modules allows to generate logical UDP packets which are transferred in multiple standard IP packets. The use of Jumbo[5] packets makes this functionality obsolete, but it is still included and can be reactivated as required.

The concept of the Ethernet Engine is to provide high flexibility in terms of the incoming data rate from the DDR3 controller, customizing of the Ethernet headers, and as little dependency from external control signals, in order to provide an autonomously running module. Of course all address and port fields of the Ethernet headers can be specified for all channels separately by FPGA registers as indicated in table 7.2. This allows convenient integration in a standard Ethernet network.

7.4 Summary of Data-Rates and Bandwidths

The Aurora Core provides a theoretical maximum bandwidth of 7.5 GBit/s. Due to regular synchronization cycles which are automatically introduced by the IP-core, the actual achievable rate during continuous operation is slightly lower. Because the line rate of the Aurora channel is 25% larger than the input rate from the readout ASICs, the transmission pauses during the synchronization phases can easily be compensated.

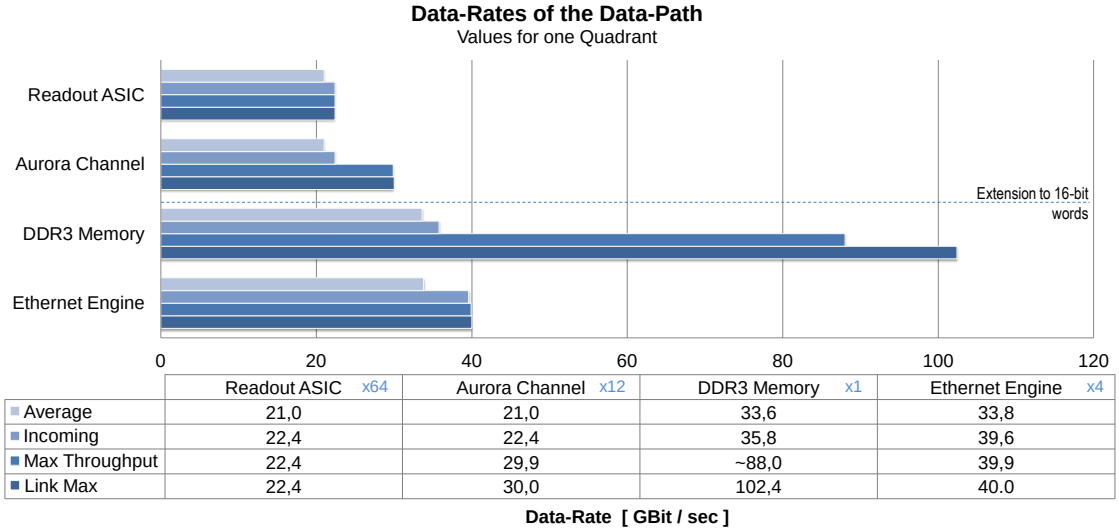
The DDR3 memory requires several 200 MHz cycles for preparation before each read or write cycle. This is limiting the bandwidth of the controller especially while alternating read and write bursts are applied. In order to check if the preparation cycles mean a problem

[5]The default IP standard assumes a Maximum Transmission Unit (MTU) of 1500 Bytes. But with the introduction of the Jumbo packet size the MTU can be increased up to 9000 Bytes.

for the data transfer, the maximum throughput was simulated and afterward tested in the hardware. The resulting throughput during alternating read and write bursts, containing 256 words of 512-bit, has been determined to over 88 GBit/s. The throughput lowers rapidly with smaller burst lengths. On the writing side, the burst length is dynamically determined from the current number of words in the 256-to-128-bit FIFOs in order to provide flexible and as fast as possible transfer to the memory.

The interface of the MAC core is a 64-bit parallel input running at 156.25 MHz, which means a real 10 GBit/s data rate. Between two Ethernet packets, the MAC core assigns a not-ready signal just for 2 clock cycles during which no data can be transferred. Regarding the total packet size, including the Ethernet headers, of 8192 + 46 = 8238 Bytes, which are transferred during 1030 clock cycles, the resulting maximum line rate of the 10 Gigabit links which can be continuously utilized lays over 9.978 GBit/s.

The four 16 KB FIFOs at the input of the Ethernet Engine are able to receive the full 102.6 GBit/s rate from the DDR3 controller. Since the input rate from the DDR3 controller is much higher than the summarized bandwidth of the four Ethernet channels, a throttle has been introduced to the output of the DDR3 controller. This throttle can be used to limit the average data rate of the DDR3 read controller and therefore also the average output rate of the Ethernet Engine. In combination with a reduced number of images which are transferred per Train Data, this feature can also be used to adapt the output rate of the detector to a receiver device with less performance.



	Readout ASIC x64	Aurora Channel x12	DDR3 Memory x1	Ethernet Engine x4
Average	21,0	21,0	33,6	33,8
Incoming	22,4	22,4	35,8	39,6
Max Throughput	22,4	29,9	~88,0	39,9
Link Max	22,4	30,0	102,4	40.0

Figure 7.10.: **Summary of the data-rates of the datapath.** The rates are computed for one full quadrant. The number of channels are indicated for each module. The average value is computed over the whole 100 ms readout cycle. The incoming rate describes the actual data rate in the module during operation. Due to the extension of the pixel word-size from 10 to 16 bits, the incoming data-rates increase by a factor of 1.6.

In figure 7.10 the data rates of the four data path modules are summarized. An increase of the incoming data rates is caused by the extension of the pixel word size to 16 bits (+60%) and by the insertion of meta data (+0.01%) and the Ethernet headers (+0.66%). The

maximum throughput is actually reached during phases when sufficient data is provided at the input of the corresponding module.

8

Selected Measurements

In this part presents selected measurements which have been executed to evaluate the transmission quality of the serial links and the stability of the data acquisition.

A reliable and stable functioning data acquisition is essential for the usage of the detector. This includes also reliable transmission of the data bits via the serial connections between the different detector parts. Especially, the transmission via Multi-Gigabit links is not trivial and reliable operation has to be verified. In order to quantify the transmission quality of the serial links in the detector system, several tests have been implemented.

Special attention was paid to the Gigabit links of the Aurora and the Ethernet channels. To evaluate the connection quality of high speed links in an FPGA, Xilinx provides a special IP-core which implements an integrated bit error rate test (IBERT). Therefore, the Chipscope application provides a special graphical interface which allows convenient parameter manipulation to optimize the transmission quality. Also the bit error rate can directly measured using the IBERT tool.

After the single components have been successfully tested, the operation of the complete readout chain from the ASICs up to the receiving computer has been evaluated using generated bit patterns from the first Ladder camera prototype. A few words are also spent on this measurement.

Finally, the utilization of the logic cells in the IOB and the PPT FPGA are summarized.

8.1 Serial Links and Transmission Quality

The most critical transmission line in the detector system is the high speed link from the IOB to the PPT. In a thin and flexible cable of about 40 cm length, three differential pairs which transport a 3.125 GHz signal are combined which is a fairly challenging setup for a reliable data transmission at this frequency.

In order to evaluate the data transmission from the IOB to the PPT, the analog signal quality was measured and tested for a stable connection. Therefore, the internal synchronization logic of the Aurora core helped by signaling the successful initialization of the transmission channel which is only possible if the signal quality is good enough to realize bit transmission with a very low bit error rate.

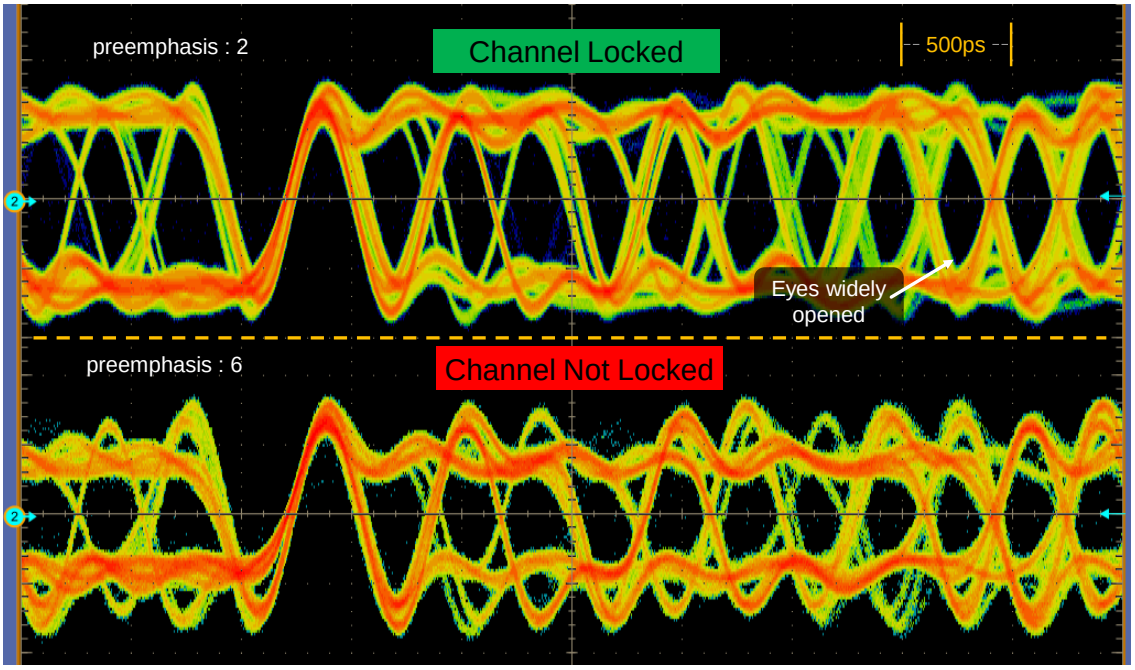


Figure 8.1.: **Oscilloscope measurements of the signal quality.** The signals have been measured at the input of the Aurora channel on the PPT. Tow different settings are shown. Best results have been achieved with swing = 10 and preemphasis = 2.

In order to optimize the signal quality, the electrical characteristics of the high speed serializer in the Spartan-6 FPGA on the IOB can be changed by two parameters. One parameter changes the voltage 'swing', resp. the amplitude of the differential signals, and the 'preemphasis' parameter, which changes the steepness of the bit transitions. The effect of these parameters is shown in figure 8.1.

After connecting four IOBs to the PPT and loading a firmware which implements an IBERT IP-core at all high speed serial links, a bit error rate test was executed and run for several days. At the bottom of figure 8.2 the IBERT GUI is shown. It summarizes the automated bit error tests for all 12 high speed links of one PPT. The resulting bit error rates and the detected bit errors are listed. In all channels, no single bit error could be detected.

Also to test the transmission quality of the QSFP+ Ethernet link the IBERT was used. Therefore two PPT boards have been connected with one QSFP+ to QSFP+ cable and on each Kintex-7 FPGA an IBERT core has been instantiated. The result of a long time measurement is shown in figure 8.3. After over 67 hours also no single bit error could be detected on any channel.

After the functionality and the signal quality of the serial links has been verified, the stability of the complete readout chain has been tested. Therefore, the available Ladder camera has been configured to send out a defined bit pattern in each pixel in all images. Since two ASICs on this Ladder do not properly work only 14 ASIC channels can be measured but it can be assumed that this setup still produces representative measurement results. During this test, the whole readout chain has been operated under realistic conditions in vacuum and with temperature control enabled. The test data has been packed into the

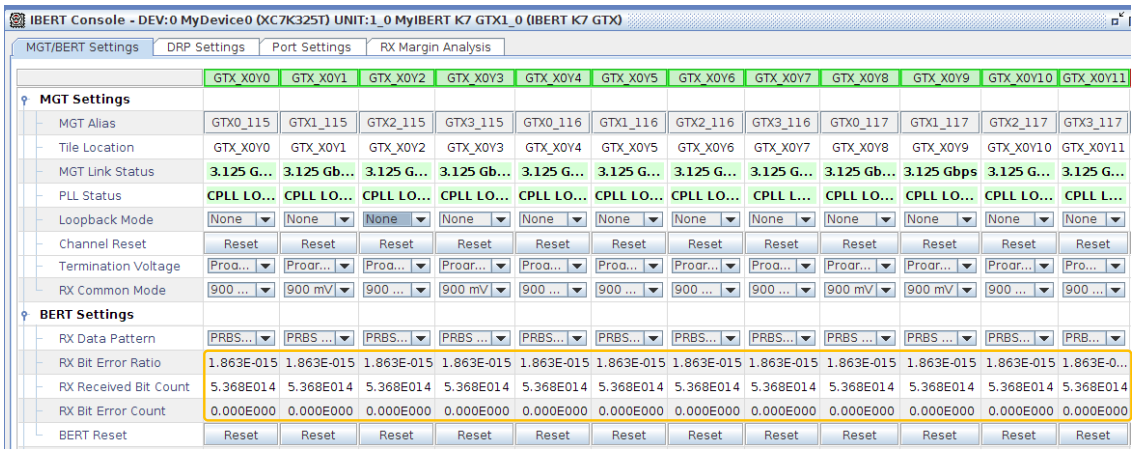


Figure 8.2.: **The IBERT Interface in the XILINX Chipscope application.** It provides convenient access to the bit error rate test functionality of the IP-core. After x hours no single bit error could be detected.

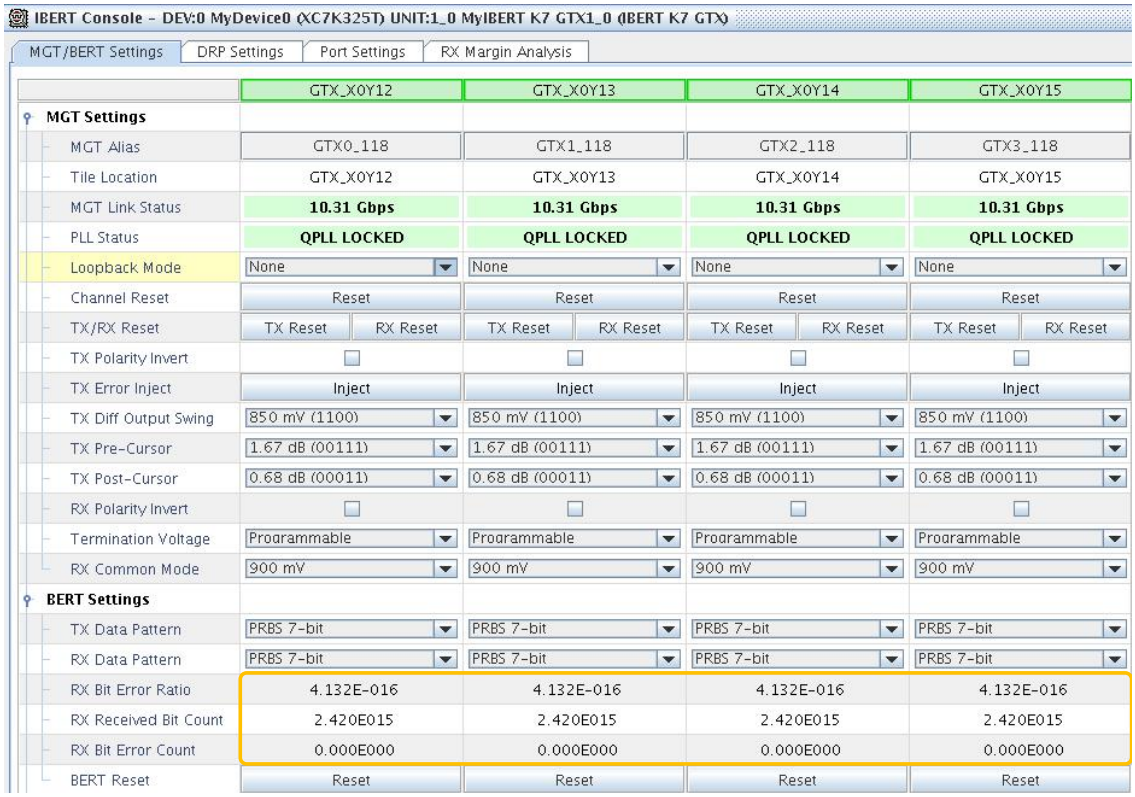


Figure 8.3.: **The IBERT Interface in the XILINX Chipscope application.** It provides convenient access to the bit error rate test functionality of the IP-core. After 67.2 hours no single bit error could be detected.

standard train data format and received and checked by a software application running on a PC.

The test run over one night and 79001 trains have been received and checked for errors. After 36.2×10^{12} words no single error has been detected. This can be computed into an upper limit of the bit error rate for each ASIC channel to 3.8×10^{-13} or for the whole data path to 1.7×10^{-14}. This test shows the overall functionality of the readout chain and the reliable and correct transmission of the pixel data from the readout ASIC to the PC layer has been verified under realistic conditions.

The following table summarizes the results from the previously presented measurements:

Bit Error Rates				
Link	QSFP+	Aurora	ASIC	Data Path
Data Rate	10 GBit/s	3.125 GBit/s	350 MBit/s	10 GBit/s
Checked Bits	2.42E15	5.37E14	25.86E12	579.87E12
Bit Error Rate	<3.72E-16	<1.86E-15	<3.87E-14	1.72E-14
Hours wo Error	>67.2	>47.76	>20.64	>16.08

Table 8.1.: **Summary of the bit error rate measurements.** In all measurements no single bit error has been detected.

8.2 FPGA Utilization

Just for completeness, the FPGA utilization of the actual design is listed for the PPT and the IOB FPGA. The debug logic has not been removed from the design since it does not slow down the operation even though it reserves a large fraction of the utilized logic cells and in particular one quarter of the reserved block ram cells of the design. The Microblaze microcontroller defines the largest block in the design and the other components implement similar numbers of logic cells.

In table 8.2 it can be seen that the logic in the Kintex-7 FPGA is hardly 40% used. Even over 70% of the block rams are free.

Consequently, the IOB and the PPT FPGAs provide certain space for future developments and further integration of additional functionality into the FPGA.

Kintex 7 XC7 K325T -2 FFG900				
	Register	LUT	36K BRAMs	18K BRAMs
DataReceiver	18599	19300	28	68
DDR3 Controller	14670	18627	0	0
10GBE Controller	13300	13838	24	0
Debug Logic	8199	4950	32	0
	20075	21554	21	2
Other	21876	23611	31	16
Total Used	84058	88508	128	89
Available	407600	611400	445	890
Utilization	20.6%	43.4%	28.8%	10.0%

Table 8.2.: **PPT FPGA Device Utilization.** Hardly 40% of the logic is implemented in the actual design.

Spartan 6 SLX45T -3 CSG324			
	Register	LUT	Block-Memory
DataReceiver	4370	4230	18
Aurora Core	481	321	0
Debug Logic	4093	1563	40
Config. Register	659	595	0
Total Used	10772	9050	58
Available	163728	81864	116
Utilization	7%	11%	50%

Table 8.3.: **IOB FPGA Device Utilization.** Roughly 10% of the logic is used, and half of the utilized logic is implemented by the debugging logic, which can optionally be excluded.

Part III

Software Development for the DSSC Detector

This part describes the software architecture which has been developed during this thesis to control and readout the DSSC detector system. Within two chapters, the two major software developments are described which have been developed to operate the system during the final utilization at the beamline and during the long process of prototyping and development.

Even if there is only one detector system, several applications have been developed which build a whole framework to control, read out and measure the detector system. Also several reduced test setups have been assembled which are used to evaluate and develop the single components of the DSSC detector. An overview is visualized in figure 8.4.

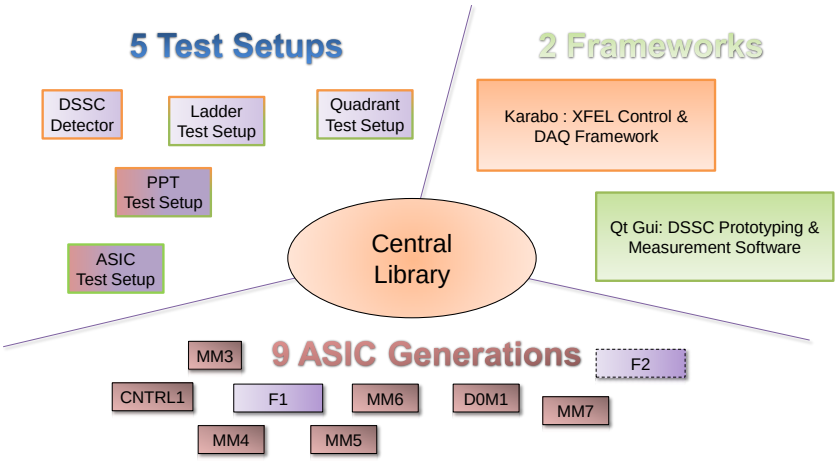


Figure 8.4.: **Design Goal for the Control Software.** A central library is needed to operate the variety of Test Setups and ASIC generations. Also two different software frameworks are used in which the control applications are written. The goal is to combine as much as possible within one library which implements central functionality for all setups and ASIC generations. The different compatibilities are indicated by the box and frame colors.

While the Ladder and Quadrant Test Setups are just reduced versions of the final megapixel DSSC camera, the ASIC Test Setup and the PPT Test Setup implement also different components or interfaces as described in section 5.10. For example, the ASIC Test Setup implements a USB interface and the other setups can only communicate via Ethernet. Also the readout data is transferred via completely different physical links. Furthermore, the detector control has to be implemented in two different software versions: In the Karabo framework, used at the beamline, and in the Qt software version, which is used for prototyping and measurements using the PPT and ASIC Test Setup.

Between the different setup types, the software frameworks and the various ASIC generations certain compatibilities exist. While the ASIC Test Setup can only be operated with the QtGui, the PPT, Ladder and Quadrant Setups can be operated with both frameworks. On the other side, using the ASIC or PPT Test Setup, all 9 ASIC generations can be operated. The Ladder and Quadrant Setups can operate only the large format ASICs, F1 and in future F2.

Since in all setups ASICs are operated, there exist important features, which have to be

provided in each implementation of the control software. This is why the central library is such important for the software development. It allows to implement functionality only once and provide it to all compatible parts which are able to use it.

The central library could only be realized, by developing a complex class hierarchy which makes strong use of the inheritance functionality of the C++ programming language. This enabled to integrate even the more complex features of the software in a way where they are accessible from both frameworks and all different setups.

The central functionality of the implemented software environment can be summarized to:

- Initialization and configuration of the system
- Reception and sorting of the image data
- Automated measurements and data analysis
- Automated trimming of pixel parameters.

While the automated trimming methods are less important for ASICs with smaller pixels matrices, the other functionality is required for all ASIC generations.

The realization of the central library and the software environment to operate the detector and the test systems is presented in this part within three chapters.

In the first chapter, the hierarchical class structure of the central library is presented. Also the distribution of system specific functionality to the derived classes is shown in order to describe how the different setup types are supported. As a central aspect of the control software, a special class is presented which implements the register value management. Finally the implementation of the software which received the image data on the computer is presented. I order to provide real time data reception and pixel sorting at the high data rates, the data receiver implements mutli-threaded data sorting and elaborate data management methods.

The second chapter deals with the more complex features of the software environment. This includes a framework for automated pixel parameter measurement and a corresponding data analysis application which is widely used in the consortium. Due to the complex class structure of central library this measurement and analysis framework could be made available for all different test setups and ASIC generations. The same is valid for the automated trimming routines which are presented at the end of the second chapter. The trimming algorithms have been developed to automatically adapt the configuration of the large numbers of readout pixels in order to compensate for the inhomogeneities of the sensor and analog circuits in the readout pixels. With the composition of larger format cameras, which implement larger numbers of pixels, the automated trimming of pixel parameters gains more and more importance. During this thesis, a number of trimming algorithms have been developed which implement this challenging task for all 65K pixels of an entirely populated Ladder camera. These trimming algorithms will be described in section 10.2.

The developed architecture for the new Karabo framework is presented in the third chapter. Karabo implements all control and DAQ functionality at the EuXFEL beamline and the experiments. Therefore, several Karabo devices are presented which have been

implemented to perform measurements using the first Ladder camera prototype during an important measurement campaign at the Petra III beamline and will later be used to operate the megapixel camera.

9

Development of the Central Library and Control Application

In this chapter, a general overview about the software which has been developed to control and configure the DSSC detector is given. In addition, a basic description of the class hierarchy is given to show how the compatibility for the various environments could be realized.

Large effort was also made in the functions and methods for measurements and data analysis. A the basic the data format as well as a selection of the capabilities of the measurement and analysis framework will be described in section 10.1.

In order to perform measurements, a data receiver software is needed which receives the image data from the detector. Because of the high data rates, the data receiver has to be implemented carefully with a focus on efficient data management and effective pixel sorting. Pixel sorting is required in order to display the image data in a suitable format which represents the spacial pixel arrangement in the ASIC.

9.1 Design Approach of the Class Hierarchy

With respect to the large number of different test setups, software frameworks and ASIC generations the implementation of a central library becomes quite complex. The final structure has been developed over time with several intermediate steps which will not be presented here. In order to shorten the description only the final design is presented.

In figure 9.1 the design approach is visualized. The two software frameworks have full access to all features of the central library. Since the hardware access is different for the different test setups, the central library can only provide and interface for the hardware access. The implementation of the special hardware access is implemented in derived classes which are specific for the implemented setup or the connected test ASIC.

9.1.1 Class Structure

The development of the class hierarchy was strongly driven by the advances in ASIC development. While the software for the CNTRL1 test ASIC has already grown into a fairly

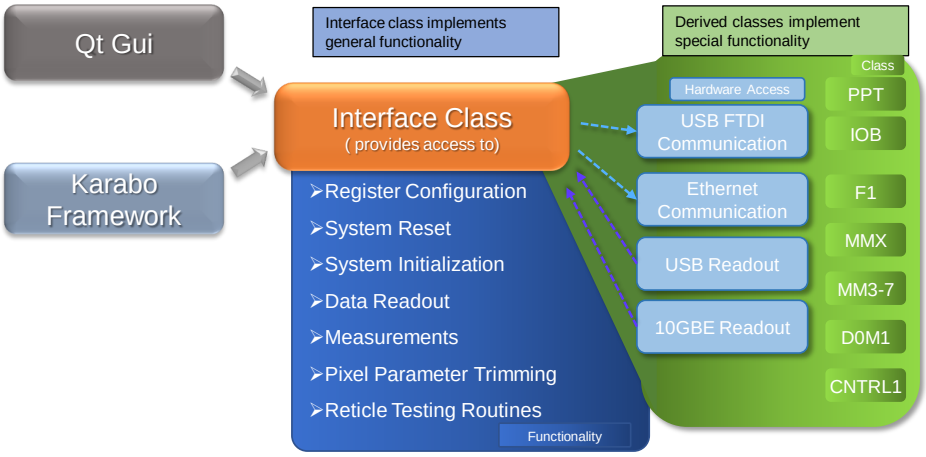


Figure 9.1.: **Design Approach for the Software.** The two user interfaces access the hardware through one common interface class. This class directly implements general functionality and provides the interface to the special implementations via the C++ inheritance functionality using derived classes. This implementation allows to reuse most of the central functionality for different ASIC generations and both frameworks. By masking also the physical communication interface, it is even possible to reuse software implementations in both the ASIC Test System implementing the USB communication, and the PPT environments which implement the communication via Ethernet.

productive environment that already provided many functionality, the next ASIC generation showed some incompatibilities and the software could not have been easily reused.

With the first restructuring of the C++ code, the support for more than one ASICs generation could be realized which described the first step to the final class hierarchy. The first redesign was realized in cooperation with Florian Erdinger and Jan Soldat. The goal was to integrate all common functionality and most of the FPGA control parts into one common class which implements the base class for the derived control classes for different ASIC generations. Unique ASIC features have been extracted into derived classes for each test ASIC generation (e.g. *SuS::CNTR1*, *SuS::F1* or *SuS::MMX*). This approach became more and more successful with the finalization of the digital control part in the ASIC in the first large format ASIC F1.

After the first differentiation, the *SuS::CHIP* class summarizes all FPGA configuration and readout functionality as well as common JTAG and pixel register interactions. Consequently, the derived ASIC classes solely define the ASIC unique features such as the number of pixels and certain special parameter manipulations. For the ASIC Test System which implements the FTDI USB interface, this level of abstraction would implement all required functionality as shown in the bottom part of figure 9.2.

Another large software redesign started when the PPT Test System became more and more important. The PPT Test System included the first time all important parts of the final readout chain of the DSSC detector. During this restructuring, the *SuS::ChipInterface* class was introduced which masks the physical communication interface in a way which allows the exchange of the hardware communication layer. This enabled for example the PPT Setup to access all measurement functionality which before has only been available in

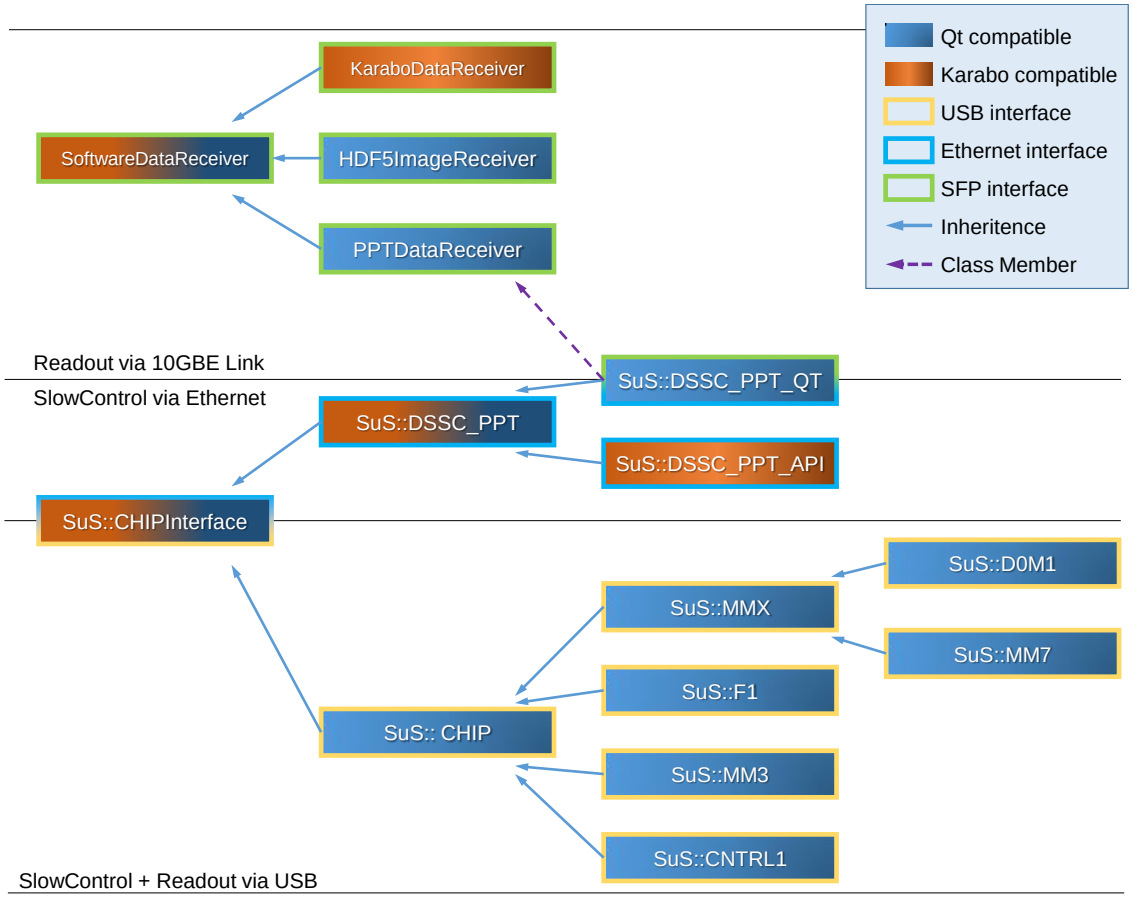


Figure 9.2.: **Inheritance diagram of the DSSC control and readout software.** The class provides pure virtual functions, which provide the access to readout and control functionality without implementation. The hardware specific implementation is realized in the derived classes.

the ASIC Test Setup.

By implementing the *SuS::ChipInterface* class in a GUI, all functionality of the central library becomes available. Especially the complex functions like measurements, ASIC testing and pixel parameter trimming can be re-used which has been the main goal of this hierarchical implementation.

In the final implementation, the *SuS::ChipInterface* class provides the interface for all central features. Additional effort has been made to integrate as much intelligence of the derived classes into the *SuS::ChipInterface* class, which enables as much code recycling as possible.

System specific configurations like the configuration of the FPGA registers or specific hardware components like PLLs and the switched power regulators can only be provided by the derived classes which control the respective test setup environment.

In the first derived level, the *SuS::CHIP* and the *SuS::DSSC_PPT* class implement the system specific hardware functionality like the slow control communication layer and FPGA register configuration. Without going too deep into detail, the main control parts of these two classes are listed in the following, in order to give an impression about the complexity of differentiation in this level.

The SuS::CHIP Class

This class implements the slow control communication of the ASIC Test Setup. The communication with the hardware happens via a USB connection to an FTDI FIFO IC which is connected to the FPGA. The USB communication is realized by a library which is provided by the FTDI vendor. For convenient usage the IomPackage class has been introduced which is a wrapper for the read and write functions of the FTDI library. Using the IomPackage wrapper class, the *SuS::CHIP* implementation strongly benefits from the simplified generation of the USB packets.

Main functionality of the *SuS::CHIP* Class are:

- Firmware upload to the System FPGA.
- Implementation of the USB communication: Creation of the byte stream to configure the FPGA registers. For example the JTAG commands, generated in the *SuS::ChipInterface* class, have to be extended with an address and additional control bytes.
- Implementation of drivers for the FPGA firmware: configuration of PLL, DAC, SPI devices.
- Readout of the ASIC data: trigger acquisition cycle and fetch data from the FPGA.
- Generation of the configuration data for the LMK and the PLL on the Mainboard.
- Control of the SPI devices on the Mainboard: ADC and DAC.
- Enabling switched powers of the Regulator Board replacement which is implemented only in ASIC Test Setup.

Since the configuration data, which is programmed into to the readout ASIC registers, is the same for both environments, the generation of the bit stream could be integrated into the *SuS::ChipInterface* class. The same could be realized with the checking of the read-back data. Consequently, all ASIC register configuration and checking logic has to be implemented only once in the *SuS::ChipInterface* class, and can be accessed automatically by all derived ASIC classes.

The SuS::DSSC_PPT Class

This class implements the slow control communication with the *SlowControlServer* running on the Microblaze microcontroller on the PPT FPGA. It is the central class for all implementations which communicate with the PPT and therefore also for all Ladder and detector control applications.

The main features are listed below:

- Firmware upload to the PPT and the IOB FPGA.
- Implementation of the communication with the Microblaze: all data is packed into TCP packets which can be decoded by the *SlowContolServer* running on the Microblaze (see section 9.2).
- Generation of the configuration data for the LMK on the Mainboard. The configuration must be passed from the Microblaze to the LMK controller on the IOB which configures the clock buffer accordingly.
- Manual control of the Regulator Boards for debugging.
- Configuration of the PLL on the PPT.

In contrary to the *SuS::CHIP* class, the *SuS::DSSC_PPT* class does not implement any data readout functionality. It is only a control and configuration class. The PPT transfers the image data via a distinct physical interface, the 10 Gigabit-Ethernet link. This is also reflected by the software. The implementation of the data reception from the 10 Gigabit-Ethernet interface and the unpacking of the data from UDP TTP packets is realized in the *SoftwareDataReceiver* class as shown in figure 9.2. The implementation of the readout classes is described in more detail in section 9.3.

As visualized, the *SuS::DSSC_PPT* class is derived further into GUI specific classes which basically provide certain simplifications for the communication of the GUI with the interface: the *SuS::DSSC_PPT_API* class for the Karabo implementation and the *SuS::DSSC_PPT_QT* class for the QtGui implementation.

The most important extension of the *SuS::DSSC_PPT_QT* class with respect to its parent class is the implementation of the readout functionality using the *SoftwareDataReceiver*. Additionally, it implements Qt specific *signals* and *slots*, which are used to increase the usability of the GUI.

By introducing the *SuS::DSSC_PPT_API* class, the *SuS::DSSC_PPT* hardware control class became compatible to the Karabo framework including all its already implemented functionality.

The implementation of the various classes and redesign of the software structure has been made a lot of effort. Before the last big redesign, the central classes implemented lots of features from the Qt library. Since the use of the Qt library is not feasible in a class which is used in the Karabo framework, these parts had to be replaced in all available classes by functions from the standard C++ library. Despite the high expenditure, the redesign has paid off since it enabled the Karabo device to access all existing and well proofed hardware communication functions of the *SuS::DSSC_PPT* class. This allowed rapid Karabo development since the implementation of the API for the Karabo integration has been mainly reduced to the creation of the user interface and the states of the Karabo framework.

The big advantage of this implementation is obvious. All hardware control can already be tested and verified in the Qt environment which provides more functionality and allows much faster software development. Also the development time for the Karabo device reduces and the effort is basically limited to the introduction of new buttons and to the adaption of the API to the interface of the *SuS::DSSC_PPT* class. All functional code WHICH interacts with the DSSC hardware has to be written, tested and also documented only once.

9.2 Configuration Register Representation

A central part of the control software is the manipulation of the different configuration registers of the detector hardware. Typically, the control software keeps an own representation of all register values in the memory in order to keep track of the current state of the hardware and to check against the read back data. Also storage and loading of configurations from a database or simply from files has to be possible. A special configuration register class, the *ConfigReg* class, has been introduced which combines all related tasks. It provides several hierarchy levels in order to reflect the register structures of the detector hardware and the readout ASIC. The elaborate structure of the *ConfigReg* class enable the representation of every configuration register type in the DSSC detector. This allows unified representation of all registers in the software, in the stored configuration files and also in the respective user interface.

The realization is presented in the following:

Typically, a register type can be structured in four levels: individual *SignalBits*, multi-bit *Signals*, *Modules* that combine a number of signals and *ModuleSets* containing all *Modules* having an identical set of *Signals*.

Additional requirements are summarized below:

- Compatibility with different types of registers: FPGA registers in PPT and IOB, JTAG and Pixel registers.
- Access of registers and signal values by name using strings.
- Representation of hierarchical structures like in the pixel register. See figure 9.3
- Configurable order of register *Modules* within the configuration stream.
- Support of read only bits
- Register bits must be assignable to a signal in arbitrary order.

When this thesis has been started a first implementation already existed which represented the basic structure of the pixel registers and already met most of the mentioned requirements. Also a graphical tool to create a new register structure and to manipulate register values has also been available. This first XML file based implementation has been written by Jan Soldat during his Diploma thesis. With rising complexity of the readout ASICs and increasing numbers of pixels and configuration values, the performance of the XML parsing and parsing of the file structure became too slow for interactive usage. The re-implementation of this structure has been carried out by the author and a much faster implementation could be achieved.

The hierarchical structure of the configuration register class is described in figure 9.3.

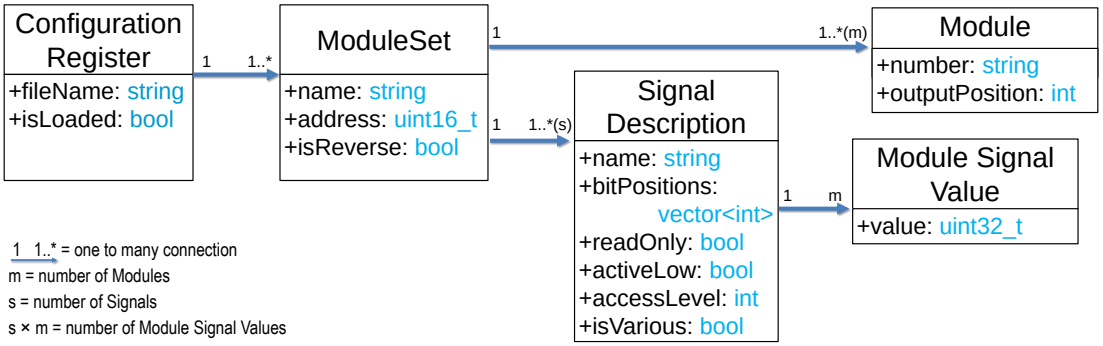


Figure 9.3.: **Logical structure of the configuration register representation.** For m modules and s signals, $m \times s$ module signal values have to be stored. The structure is implemented by the *ConfigReg* class and allows to represent all kinds of configuration registers in the DSSC detector.

One *ConfigReg* contains a number of *ModuleSets*. A *ModuleSet* has always a name, an address and contains all registers which can be accessed by its address. In case of the ASIC registers, this can be for example the Pixel Register or any of the global JTAG registers. Further, a *ModuleSet* can be flagged as reverse which inverts the order of the bits in the configuration stream[1]. One *ModuleSet* keeps all information about an arbitrary number of *Modules*. Every *Module* has a name and has a defined position within the configuration stream. This allows for example to name the *Modules* of the pixel register by its position in the image but to sort the configuration stream after the order in the JTAG chain. Therefore, all bits of a *Module* must be configurable block wise which applies for all register types in the DSSC detector. A *Signal* has a number of parameters summarized in the *SignalDescription*. The only *Signal* parameter which can be different in every *Module* actually is its **value** that has to be stored for every *Module* and every *Signal*.

The *Signal* parameter **isVarious** has a very special meaning: This parameter is used to greatly improve the performance of the GUI. It keeps track if a *Signal* has identical values in all *Modules* or not. Using this parameter, the GUI can quickly access and display this information without continuously checking all module values.

The final implementation of the *ConfigReg* class provides fast storage and loading of configurations using human readable text files. The structure is kept as simple as possible

[1] In contrary to the JTAG registers, the bit stream of the pixel registers starts with the LSB.

and shows better performance as the former XML structure. In case of a pixel register configuration, which stores the configuration of all 65536 Ladder pixels, the loading introduces only hardly noticeable delay.

The numbers for the four different configuration registers, computed for a single Ladder system, are summarized in table 9.1

Register Type	Number of ModuleSets	Is Reverse	Number of Modules	Number of Signals	Number of Bits per Module
PPT FPGA	30	FALSE	1	1-32	32-192
IOB FPGA	55	FALSE	4	1-7	32
JTAG	10	FALSE	16	1-23	16-94
Pixel	1	TRUE	65536	31	47 (F1)

Table 9.1.: **Overview about the four different configuration register types.** Numbers are computed for a Ladder system (1/16th of the megapixel detector).

In order to improve the performance during run time operation the complete configuration is hold in memory packed into a convenient class structure that organizes the configuration values and implements convenient access to all registers. All register value access is only possible by name. Therefore, the name of the *ModuleSet*, a selection of *Modules* and the name of the *Signal* must be given. Especially the usage of a string for selection of *Modules* allows flexible *Signal* value manipulation of arbitrary subsets of *Modules*. Functions, which generate sorted bit streams, are provided in order to keep all bit ordering inside the *ConfigReg* class. The bit streams only have to be packed into interface specific container packets before they are sent to the hardware. Even the comparison of read back data with the original configuration which is kept in memory is realized inside the *ConfigReg* class which allows convenient usage also for different register types.

The SlowControlServer Process

All configuration data which is sent from the PC via Ethernet packets to the PPT has to be received and interpreted by a software process running on the Microblaze in the Kintex FPGA (see section 6.1.1). The name of this process is *SlowControlServer*.

The *SlowControlServer* is a self developed light weight TCP server written in C++ which uses a standard Linux socket to listen on port number 2384. The implemented socket allows only one connection at the same time.

The *SlowControlServer* process has direct hardware access to the EPC (**E**xternal **P**eripheral **C**ontroller) registers, which are implemented in the FPGA logic. The access to the EPC registers is realized by mapping the EPC register device which is installed in '/dev/mem' into the user space. This can be done using e.g. the *mmap* command.

The basic configuration packet to access an EPC register is shown in figure 9.4. In theory, these two commands can be used to operate the whole detector since every single register of the system (EPC, IOB, JTAG) can be accessed through an EPC device and each EPC device is controlled by an EPC register. For debugging and during the early development phase this simple implementation provided certain advantages. However, it is very inefficient to send one TCP packet to write one single word. This is why a number of other commands have been introduced which are used to trigger more complex operations. Especially timing

critical operations have been integrated into a special command in order to avoid variable delays which often occur during data transfer via Ethernet.

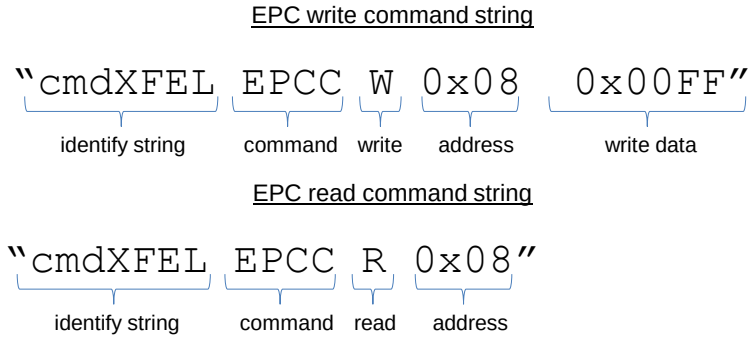


Figure 9.4.: **Basic EPC read and write commands.** These two commands define the minimum set of TCP commands that are needed to configure the whole detector system. However, in order to improve the efficiency of the communication several other commands are available that are understood by the *SlowControlServer*. All commands must start with the identify string that protects the slow control server from bad TCP packets. The identify string is followed by a four capital command string, followed by the command options. All available commands are listed in the appendix C.

At this point it is not needed to present every single command which is implemented in the interface of the *SlowControlServer*. A detailed description can be found in the appendix C. However, two interesting examples will be shown which are important for the system initialization and its operation:

1. FINT - The command for fast ASIC initialization
2. DCTW - Direct Write for JTAG Programming

FINT - The Command for Fast ASIC initialization

TCP content: **cmdXFEL FINT moduleNumber**

This command triggers a timing critical procedure which controls the powering up of the F1 readout ASICs. When the digital supply voltage is switched on the registers in the readout ASIC start in an undefined state. Unfortunately, some combinations of configuration bits generate shortcuts in the ASIC circuits resulting in a highly increased current consumption.

Since the voltage regulators can provide sufficient power to hold the supply voltage for a short period on the required level, the configuration can be corrected by programming the concerning ASICs registers with an initial configuration. Unfortunately, the pixel register is part of these registers and provides the largest number of configuration bits. At this point, the direct pixel programming mechanism of the readout ASICs helps a lot, where all pixels can be programmed at once. This reduces the number of configuration bits by a factor of 4096. The affected registers of the F1 ASIC are:

- Pixel Register - the register of the pixel matrix

- JTAG SRAM Control - manual control of the SRAM control lines
- Global FCSR 0 - control of the global decoupling caps
- Global FCSR 1 - control of the global decoupling caps

The eight steps of the fast initialization sequence are listed below.

1. Draw the ASIC reset to minimize current consumption during the following process,
2. Load all required configuration bytes into FIFO of the JTAG engine,
3. Enable all ASIC powers to avoid shortcuts in decoupling capacitors,
4. Wait until power has reached supply level,
5. Start JTAG programming,
6. Wait for end of JTAG programming,
7. Disable analog ASIC powers,
8. ASICs are now ready for real configuration.

Especially step 4, waiting for a precise period of time, can not exactly be realized in software which runs on a standard Linux OS. The process has to wait for a minimum period which is required until the voltage reaches its final level, in the current implementation this is realized by a no-op loop of configurable length. Since the regulator has sufficient reserves at the end a variable delay of a few microseconds is uncritical for the sequence. This relaxed condition made the implementation of a hardware timer not necessary.

DCTW - Direct Write or Accelerated JTAG Programming

TCP content: **cmdXFEL DCTW numCharacters**
TCP content: **cmdXFEL DCTX len;byte;byte;byte...**
length information and all bytes are transferred as hex strings:
$250 \rightarrow$ 'FA'

Using the EPC write function to send a large number of configuration bytes is very inefficient since only a single word can be sent per TCP packet. Regarding the length of the pixel registers of a full Ladder, the total number of bytes to be transmitted sums up to over 385 kB. While the data transfer alone on a 1 GBit/s network connection would add no noticeable delay, the decoding of the TCP packets became really slow on the Microblaze using the primitive implementation. Consequently, a faster data transfer has been implemented using a direct write mechanism which allows to transfer many bytes in one TCP packet.

The direct write protocol is a handshake protocol and requires two packets from the PC side. The data transfer is initiated by the **DCTW** command followed by the number of characters which will follow in the data packet. This enables the *SlowControlServer* to

prepare a buffer which has the right size to store all data incoming data. The first command is acknowledged in order to inform the software that everything is prepared for the data transfer. In the next step, the **DCTX** is sent followed by a semicolon separated list of the configuration bytes. The first word has to be the number of entries in the list. The length and the bytes are added to the string as characters in hexadecimal notation.

A previous implementation implemented FTP file transfer to send the configuration data to the SlowControlServer. This already showed a certain acceleration of the pixel register programming compared to EPC write commands.

However, file operations on the Microblaze are slow and the configuration has to be read again from the file into the user space. Using the direct write mechanism, the configuration data is received directly in the application and can instantly be passed to the JTAG Engine. By introducing the direct write mechanism, JTAG configuration operations could be accelerated by at least a factor of two. Using the current implementation a pixel register programming cycle of the whole 16 ASICs Ladder without read back is possible below 2 seconds.

9.3 The Multithreaded Software Data Receiver

The DSSC camera requires a fast and stable readout system which is able to deal with heavy throughput. One link of the camera transfers image data with the full data rate of the 10 Gigabit-Ethernet link. In order to provide real time packet reception and pixel data reordering high-performance hardware and software infrastructure has to be provided.

The receiver hardware, which is foreseen for the final megapixel system, is the Trainbuilder. This device is designed to receive, reorder and multiplex the data stream at full speed from a complete megapixel detector system as described in section 4.2. XFEL also provides a software version for the Karabo environment which can be used for single Ladder usage and testing, called Karabo::PcLayer Device.

But as long as these components are not available for the DSSC consortium, an own software development has been required which implements the data reception and also the pixel reordering from one 10 Gigabit-Ethernet links of the detector to operate a single Ladder system.

In order to make it compatible with the available software structure it has been developed as a library which can easily be used in different applications.

Several tasks have to be implemented by the data receiver software - always with high attention on performance:

- Receive the UDP packets from one SFP channel (one Ladder module) from a UDP socket.
- Unpack the image data from the UDP/TTP[1] format.
- Implementation of the pixel reordering in order to provide sorted images for further image processing.
- Extraction of meta data like TrainID, Image descriptors, ASIC trailers etc.

[1] TTP = **T**rain **T**ransport **P**acket

As already described in the firmware part, the output data from the PPT has always the same format independent of the number of connected readout ASICs which are sending valid data. Consequently, the number of received data only depends on the number of images which are send out of the detector. In order to avoid sorting of invalid data, the software has to provide a possibility to adapt to the available ASIC configuration by for example exclude certain ASICs from sorting.

In order to reach sufficient performance on modern computer systems to deal with the enormous data rates, a multi-threaded approach is required. The most time consuming tasks have been identified and distributed to several simultaneously working CPUs. The presented approach also implements pipelined operations. The algorithm of the data receiver software is visualized in figure 9.5. Additional information about the implemented classes and their usage can be found in the appendix D where the interested reader can find an example code which illustrates the easy to use interface of the ring buffer and thread safe queue.

The idea behind the implementation is the efficient memory management in which only during start up a defined amount of memory is allocated and divided into slots which can train wise be accessed. Slots are managed by the *SlotManager*, which provides thread safe access to two independent slot queues.

The *free queue*, holding pointers to unused memory slots, and the *valid queue*, holding pointers to memory areas filled with valid data that can be processed by a connected application. The *SlotManager* thereby implements the safe access from several producer and consumer threads to both queues. Furthermore, a mechanism has been introduced to ensure that a producer never has to wait for free slots. If currently no *free slot* is available because all previously allocated memory area slots are assigned to the *valid queue*, the first slot is reinserted into the *free queue*, and thus the contained data is released for overwriting. A consumer thread fetches the valid data slot, and hence blocks it from overwriting until it returns the slot to the *free queue* of the *SlotManager*. In order to prevent that all slots of the *SlotManager* are fetched by several consumers, a sufficient number of slots has to be defined at the start. An automatic increase of slot number is avoided to prevent extensive memory allocation.

The combination of ring buffer and thread safe slot queue in the *SlotManager* allows flexible distribution of tasks to arbitrary number of threads. The ring buffer thereby provides the memory management of a previously defined amount of memory and the slot queue organized the thread safe access to the memory slots. By the combination of ring buffers and slot queues in distinct classes the introduced mechanisms can easily be utilized for various tasks.

The integration of the two *SlotManagers* in the *Software Data Receiver* is shown in figure 9.5.

The different tasks are sequenced in the following way: The *Packet Receiver Thread* receives the UPD packets from a UDP socket and queues the received packets directly into a *PacketQueue*. A *PacketQueue* stores all packets from one train, and provides thread safe access to the contained data packets.

The pixel data sorting is managed by the *SortThreadManager*. The *SortThreadManager*

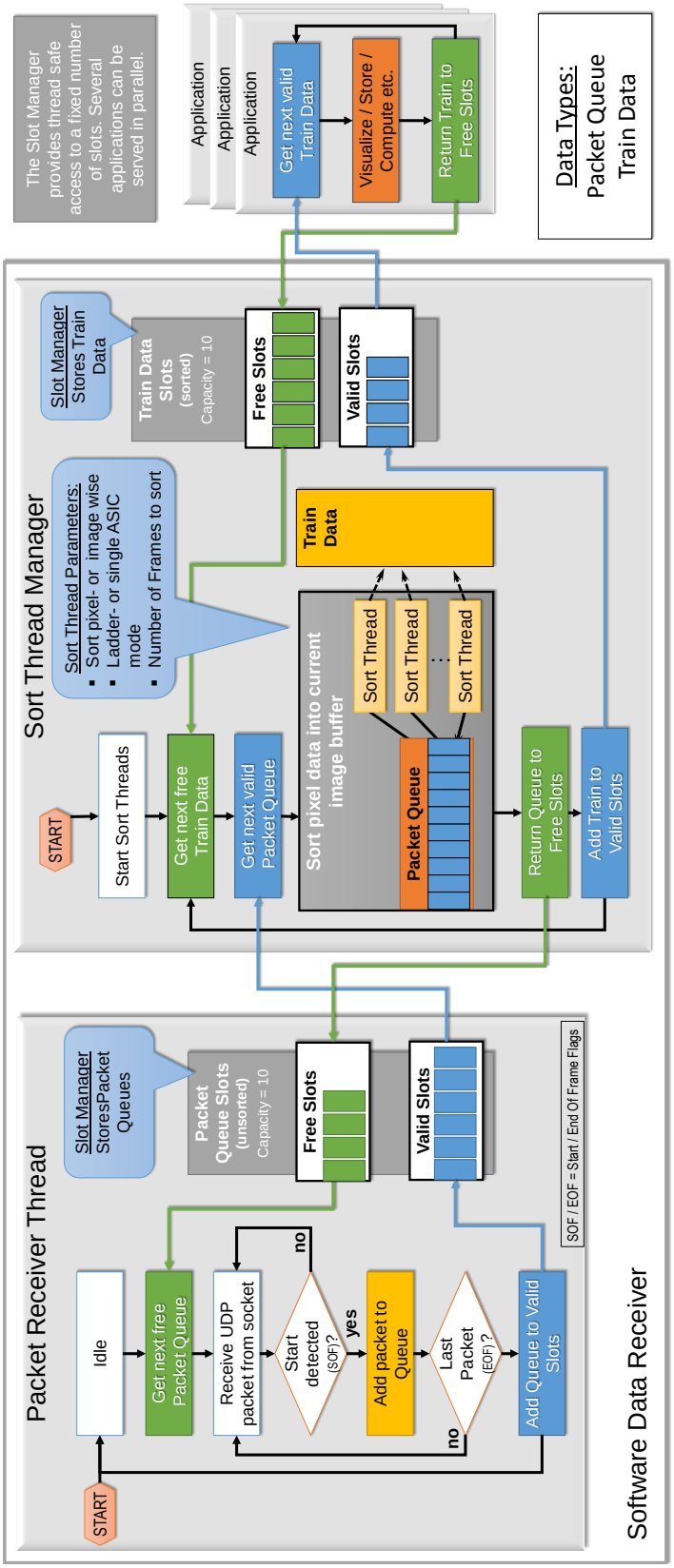


Figure 9.5.: **Algorithm of the *Software Data Receiver*.** The data is received and sorted in two steps. The packet receiver thread receives the UPD packets from a UDP socket and inserts the packets into a *PacketQueue*. The *PacketQueue* is passed to the *SlotManager*. The second step is the reordering of the pixel data and filling of the train data slots. By using multiple threads the pixel sorting is strongly accelerated. On the output side of the *Software Data Receiver*, the valid sorted train slots can be fetched by one or multiple applications.

generates and initializes a pool of several threads and manages their access to the available memory slots. A valid *PacketQueue*, which is filled with all packets of an 800 images train, contains 12815 packets. In order to distribute the packets efficiently to the different sort threads the *SortThreadManager* can simply connect the sort threads to the packet queue. All further data management can be handled by the queue implementation.

Figure 9.6.: **Pixel reordering modes.** Pixel wise is best for storing the data of pixel parameter measurements. Image wise is best for image visualization.

The sorter threads require only few configuration parameters. Since the number of transmitted frames per train can be changed, the sorter threads must be informed how many images are expected. Further, the number of connected ASICs which are producing valid data can be configured which allows switching between single ASIC or Ladder operation. The pixel reordering scheme can also be configured in order to optimize the output for different applications. Two different use cases are currently implemented which are shown in figure 9.6: for image visualization and also for the HDF5 file format, which is used in the Karabo environment, the order of the pixel data is preferred image wise. On the other side, the pixel measurements prefer the image data pixel wise, which means all images of the first pixel are stored in a block, followed by all image data of the second pixel and so on. The implementation of the sorting algorithm is implemented using a large static look-up table which is utilized by all sorter threads and stores the aim position of each pixel entry. Thus, the reordering scheme can easily be changed by changing the table entries.

The inheritance structure of the data receive software is shown in the upper part of figure 9.2. Despite the number of data to be sorted the *Software Data Receiver* only requires the socket port number to work. Thus, it can easily be integrated in different applications. There are currently three different applications which utilize the *Software Data Receiver*:

- The Qt environment, **PPT_MAIN_GUI**, in pixel wise ordering mode.
- A stand alone **Data Receiver GUI** in image wise ordering mode which just connects to the socket and stores the data into the DSSC HDF5 format.
- The **Karabo::DsscDataReceiver** device sending received and sorted data via a p2p channel to other Karabo devices like an image preview or a storage device.

9.3.1 Benchmark of the Software Data Receiver

In order to evaluate the efficiency of the implemented pixel sorting algorithm, a benchmark application has been implemented. During this benchmark generated train data has been sorted. One train contained 800 images. The generation of the data has been excluded from the time measurement. The required time to sort 8000 trains has been measured for different numbers of sort threads. The benchmark has been executed on three different server systems. The result is shown in figure 9.7.

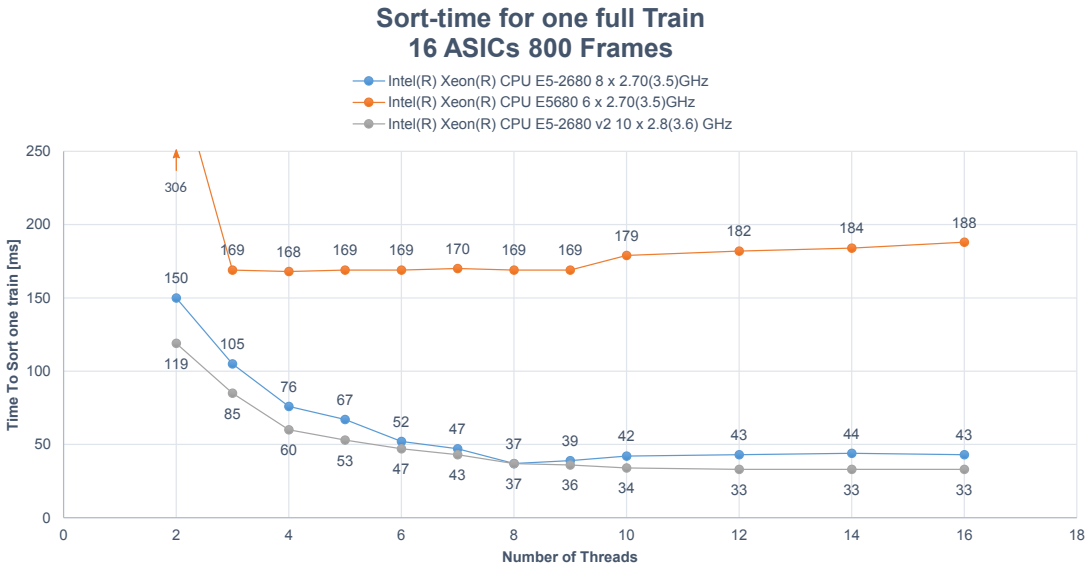


Figure 9.7.: **Benchmark of the *Software Data Receiver*.** The benchmark has been executed on three different machines for multiple number of sort-threads.

Real-time pixel sorting is achieved by a value below 100 ms. Both of the two faster machines reach this value easily by implementing a sufficient number of threads.

The duration for sorting directly decreases with the number of used sort threads. This shows that the presented implementation is able to distribute the work efficiently on several threads. The fastest sorting speed is reached by each CPU when the number of threads becomes the number of cores which are implemented on one CPU die (always half of the total cores). Since a further cache level is required to transfer data from one CPU die to the other die, an additional increase of threads does not further increase the speed of the sorting algorithm.

The presented result shows that the implementation of the *Software Data Receiver* allows real time pixel sorting. In combination with a fast 10 Gigabit-Ethernet card, reception and sorting of the image data from a single Ladder system at full speed can be realized on available machines which could be confirmed during various measurements.

10

Complex Methods for Measurements and Parameter Trimming

In this chapter the measurement and analysis framework as well as the complex trimming algorithms are presented. Both features require reliable system configuration and a stable data readout.

While the measurement features are mostly applied for ASIC prototyping and characterization the trimming methods are more likely applied for commissioning of systems implementing a large number of pixels.

10.1 Measurement Framework and DataAnalyzer

The measurement framework defines a central part of the DSSC test setup environment which is used by different groups in the DSSC consortium. It is a basic feature of the available QtGui applications and is available for the ASIC and PPT Test Setups and also in the Ladder Test Setup. The measurement framework defines a variety of automated measurements. Different types of measurements are supported. In general a measurements changes an ASIC parameter, triggers an ASIC data readout and stores the read data into a file. Measurements are also carried out using external devices, such as oscilloscopes, multimeters or pulse generators, which can be remote-controlled.

The measurement framework is a very central part of the ASIC prototyping and therefore multiple persons have been involved in writing code for all different types of measurements: namely Florian Erdinger, Jan Soldat and Manfred Kirchgessner. The development started with the first test samples of the CNTRL1 test ASIC. Because it soon became clear that the implementation of new measurement functions for each test ASIC generation would be inefficient, large effort has been made to keep as much control routines of the measurements compatible to as much test ASIC generations as possible. In general, a new ASIC generation keeps most of the existing control registers and adds new functionality.

The first design of the Measurement class structure has been proposed by the author. Only few changes have been required since then. The main purpose of this class is to simplify the generation of new types of measurements and to organize everything which concerns measurements within one class. This can be e.g. the generation of the output file,

initialization of the data structure and the preparation of the system before the measurement can be started. A distinct GUI class implements the user interface to access the Measurement features.

In the Measurement class all hardware access is realized via the central *SuS::CHIP-Interface* class. This directly allows to use all implemented measurement functions with all ASIC generations and setup environments. About 90% of the implemented measurements [1] are usable for all ASICs and in both Test Setup types. Since the GUI is implemented in Qt it is only available in Qt environments. The layout of the actual *Measurement Widget* is shown in figure 10.1.

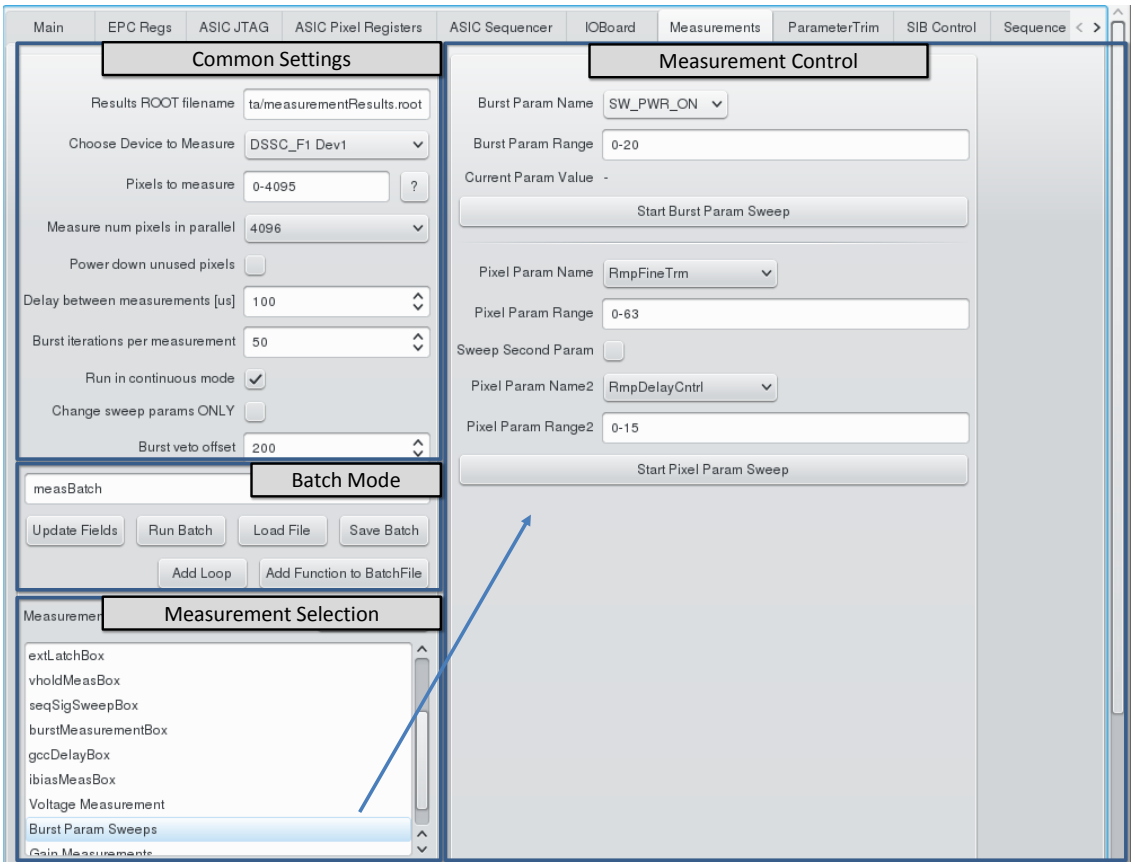


Figure 10.1.: **The Qt Measurement Widget.** Four parts are highlighted: The common settings control general parameters which are used in all measurements, the Batch Mode (see text), the measurement selection, which switches the view in the measurement control part from where the measurements can be configured and started.

The measurement control field shows only the relevant buttons and fields for the selected measurement type. General settings which can be used in every measurement, like the output file name, the pixel numbers to be measured and the numbers of iterations are always

[1] Like sweeps of pixel parameter settings or Internal DAC Setting, Pixel Injection Signal Setting, Pixel Delay Setting, ADC Gain Setting...

visible. The displayed Measurement group, shows two types of generic measurements. The selection boxes contain all entries from a specific parameter type. By selection of the parameter and definition of the range, the measurement can be flexibly configured. In the lower part, the Pixel Param Sweep can be triggered. Two pixel parameters can be selected from a box which content is filled dynamically with the signals of the currently active ASIC version. In this way the one function which is triggered by the Pixel Param Sweep Button enables various types of measurements of arbitrarily combined parameters.

The *Measurement Widget* provides even more automated and flexible measurement functionality. With increasing number of parameters in the Widget and further understanding of the readout ASIC characteristics, more sophisticated measurement combinations became required. By introducing the **Batch Mode**, Florian Erdinger created a very useful tool which improves the usage of the already available measurements a lot.

The Batch Mode reads special text files which can be used to access all control fields of the GUI and also to trigger all available measurements which can be triggered from the GUI. By implementing also nested loops in an own description language, arbitrary combinations of measurements can be started and run automatically in a defined order. This allows acquisition of measurement sequences of various parameters without writing C++ code. Since all parameters of the GUI can be accessed by the Batch file, it can also be used to store GUI configurations which can be used, for instance, as a starting point for measurements. The **Batch Mode** reduces error sources and improves the usability of the GUI enormously. To give an impression how a Batch file works a short and simple example is shown in listing 10.1.

10.1.1 Data Format in Measurements

The format of the acquired data is a key parameter for efficient data analysis. Since most of the measurement functions aim for characterization of pixel parameters a pixel wise data storage format is preferred to an image wise format. For data acquisition and measurement reconstruction important configuration parameters must also be included in the measurement files in order to keep track of the system status during the measurement. A basic decision, which was made early in the project, was the usage of root files for storage of measurement data. The root library [96] provides a large number of analysis tools like fit functionality, histograms and of course plotting and also the possibility to add meta information to the image data. An overview of the main benefits of the root library is listed in the following:

- C++ compatible,
- Comprehensive documentation,
- Close integration into the Qt library,
- Plotting: histograms, line plots and 2D color maps,
- Fit functionality,
- Build-in file compression,

Batch Mode Example

```
#THIS IS A COMMENT
#Asignment of GUI variables:
pixels = 0-127
burstVetoOffset = 100
parallelPixels = 0
iterations = 15
pwrdUnusedPixels = 1
rootFileName = RmpFineTrm.root
dacSweepRange = 3000-5000;100
fcfBuffer = 1

#Functions start with '!'
!intDacSweep

!setJtagRegSignal(Global Control Register,all,PxInjectionSig,0)
!setPixelRegsSignal(all,RmpFineTrm,1)
!setSequencerParameter(RampLength,150)

!beginLoop(IRMP=0-20;2)
  !setRootFile(RmpFineTrm_irmp=IRMP.root)
  !setPixelRegsSignal(all,RmpFineTrm,IRMP)
  !intDacSweep
!endLoop
```

CodeListing 10.1.1: Simple batch mode example which shows parameter definition, function calls and loops. Loops can also be nested. The loop parameter will be replaced in every occurrence. If a Batch file is loaded in the GUI, all parameters are filled into the corresponding fields up to the first function call marked by an exclamation point.

- Container classes for image and meta data,
- Version control.

In order to show the evaluation capabilities of the data analysis framework a short description of the data format will be given. The defined root file format is common for all measurements of the Measurement class which read out image data from the ASICs. The data structure is visualized in figure 10.2.

In a root file, the measurement data is stored in trees and branches[2]. Each tree contains one ore several branches and each branch stores a (large) set of entries. An entry is defined by one single data set which is acquired during one measurement iteration. In order to optimize the data structure for pixel-wise data analysis, all measurements from one pixel are stored in the same tree. If several measurement have been acquired for the same pixel, the pixel-tree contains several branches.

On the other hand, when the same measurement was performed for several pixels, several trees are generated and each tree keeps one branch with a set of entries representing the

[2]The data access within one tree is thereby rather cheap and in contrary the switching between trees is rather costly.

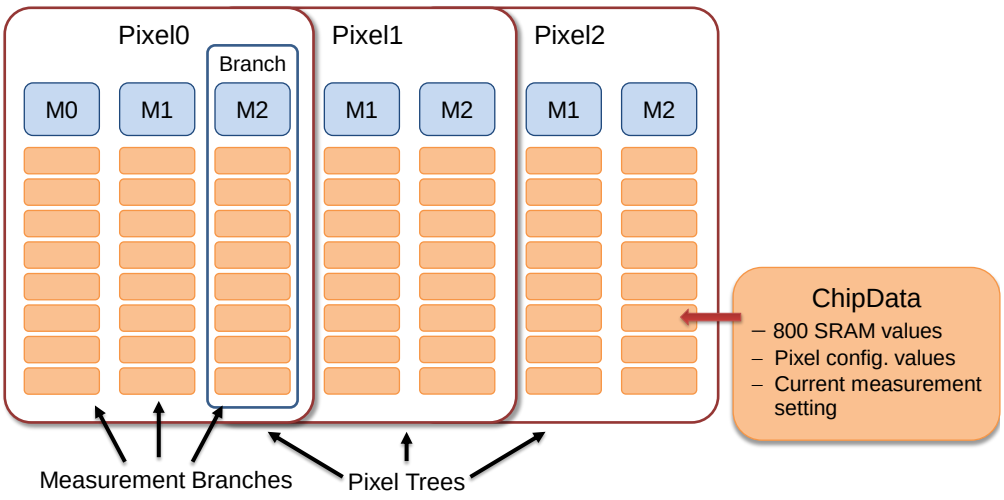


Figure 10.2.: **Scheme of the root file storage structure.** All measurements of the same pixel are stored in the same root tree as separate branch. The branch elements store all data for all iterations and settings linearly in the branch.

measurement points of the measured pixel. In order to keep the hierarchy simple, the data sets of all acquisition cycles for the same measurement are stored linearly in one branch.

The data set which is stored in a pixel branch is the *ChipData* structure. It contains all SRAM data from the measured pixel and also certain configuration data. Therefore, the root library allows to store complex data types of any data type or format. Even whole C++ class types are possible which enables to include intrinsic functionality into the data structure. For instance, data decoding or some computations can be implemented directly by the data set class.

One negative aspect of the root frame work is the weak support of parallel data processing. Since the access to a root file can only happen in one single thread, multithreaded data analysis is faced certain difficulties and has not been implemented in the *DataAnalyzer* during this thesis. But since especially the pixel by pixel processing could profit greatly from parallel analysis, the introduction of multithreaded data analysis is definitely be worth to be investigated.

10.1.2 The DataAnalyzer Application

The utilization of a uniform measurement format provides great possibilities for data analysis. With the introduction of the *DataAnalyzer* Application a tool became available offering a standardized data analysis platform for any kind of measurement which implements the defined measurement format. The *DataAnalyzer* provides a multifunctional but intuitive user interface for a variety of plotting and fitting methods to visualize and analyze the measured data. A screen-shot of the *DataAnalyzer* GUI is shown in figure 10.3. It was initiated and mainly development by the author and is still constantly evolving and extended with new functionality. During 4 years of software development a large number of analysis, fitting and plotting functions have been added into the *DataAnalyzer*. A complete description of all available functionality would definitely go beyond the scope of this thesis.

But since the *DataAnalyzer* became such an important tool for the DSSC consortium a detailed description of the most important functions can be found in the appendix E.

The *DataAnalyzer* is actually used by all groups of the DSSC consortium which are involved in the ASIC development. It became the main measurement analysis tool for all ASIC Test Setups, Ladder Test Setups and also the detector calibration group intensively uses the analysis functionality of the DataAnalyzer.

Since the Karabo devices are not allowed to implement root functionality, the data format which is written by Karabo devices is HDF5 [97]. HDF5 is an image container format which also allows storage of meta data. It supports hierarchical data structures as well as data compression. To use the *DataAnalyzer* functions anyway, a tool has been written allowing to convert the HDF5 files into *DataAnalyzer* compatible root files (see section 11.2.3). This enables the *DataAnalyzer* to remain the main data analysis application for all measurements until the analysis will be moved into the Karabo framework.

10.2 Trimming Algorithms

The effective gain of the front-end circuit, and thus the digital output of the ADC, depend on a variety of factors and parameters. Each pixel of the DSSC readout ASIC provides several adjustment settings, which allow the compensation of critical parameters of the readout circuit like the front-end gain, the ADC gain and the ADC offset. Some details are given in section 5.4.1.

The trimming possibilities, which are shared by both front-end types, are given by the following knobs:

- Change of the filter feedback capacitance.
  - Parameter: FCFCap_En
  - Setting Range: $1 - 15$
  - Capacitor Sizes: $\approx$ 1pF - ≈ 13.8 pF
  - Gain Range: $100\% - 7.2\%$
  - Granularity - from high to low gain: $30\% - 0.8\%$
- Change of the ADC Ramp current.
  - Parameter: RmpFineTrm
  - Setting range: $0 - 63$
  - Gain range: $\pm 64\%$
  - Granularity: $\approx 2\%$
- Change of the start of the ADC ramp
  - Parameter: RmpDelayCntrl
  - Setting range: $0 - 16$
  - Offset range: $0 - 960$ps

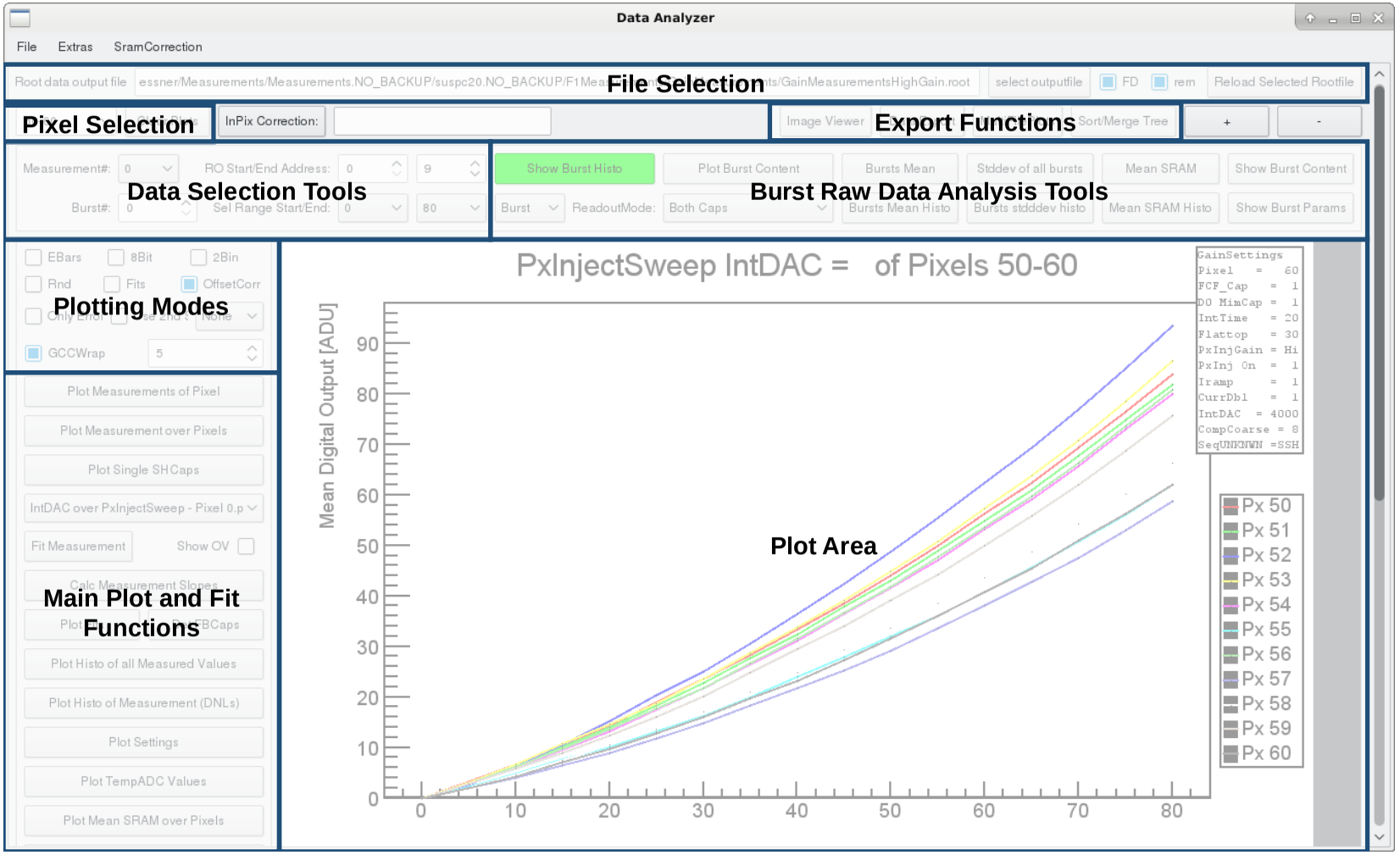


Figure 10.3.: **The *DataAnalyzer* GUI.** It is divided in several functional entities. In the top part general functions like the measurement file selection and several data export functions are located. The **Data Selection Tools** allow exclusion of certain data points like e.g. the first N SRAM entries of each burst. Using the different **Plotting Modes**, general plot features can be enabled like error bars, rounded values or special data correction. Also fit functions can be enabled for almost all plot outputs. The **Burst Data Analysis Tools** provide lots of functionality to investigate the raw data under different aspects. These functions allow access to every single data point and provide different histogram and mean value generation functions. The main measurement analysis functions are located in the left area. These functions plot the measured data vs the measurement parameter under different aspects. Also fitting functions and plots with special output format are available.

  - Granularity: 60ps or $\approx 1/12$ ADC bin

The following parameters are only available in MSDD operation:

- Change of the input capacity.
  - Parameter: D0_EnMimCap
  - Setting range: $0 - 7$
  - Capacitor sizes: 1p, 0.5p, 0.2p
  - Gain range: $100\% - 38\%$
  - Granularity - from high to low gain: $\approx 15\%$
- Change MSDD front-end transistor gain
  - Parameter: Internal DAC Setting
  - Setting Range: $0 - 8192$ global for one ASIC
  - Gain can be varied only for whole matrix at once

A very important task for commissioning of the mega pixel detector is the optimization and trimming of the pixel parameters in order to homogenize the performance of the whole system. Since the *system calibration* [98, 99] is an own work package in the DSSC consortium, this section deals only with the trimming of certain pixel parameters. The presented parameter trimming defines the first step which prepares the system for calibration measurements. Because of the large number of pixels, the trimming of the different pixel parameters can only be done by an automated algorithm. Generally, the adoption of the pixel parameter is implemented in an iterative feedback loop, which receives the current pixel configuration and the digital output of the detector (images) at the input and outputs an improved pixel configuration as shown in figure 10.4.

Even if the pixel parameter trimming is also an important part of the calibration work package, the described algorithms have been mostly developed by the author. Because no other routines have been available in advance to the first important beamline measurements, the commissioning of the detector has been started using the first trimming algorithms, which are presented in the following.

Each readout ASIC provides several knobs to trim the characteristics of the readout circuits. For each parameter, a simple method to realize this trimming has been foreseen but since the ASIC shows a different behavior than expected, things turned out to be much more complicated. Especially in the large full format ASIC F1, which is implemented in the first Ladder camera prototypes, complex inter-pixel dependencies exist which make the optimization of the pixel matrix very difficult. Lots of ideas had to be tested and several tries did not succeed until the presented algorithms have been found. And due to the complexity of the system and because of these inter-pixel dependencies, the algorithms will produce a configuration which slightly differs from the theoretical best configuration. Nevertheless, algorithms have been found which at least produce a suitable configuration allowing to perform nice measurements with MSDD and DEPFET Ladder systems.

In order to understand every idea behind the implemented trimming algorithms a very detailed understanding of the front-end circuits is required. Since all details of the front-end

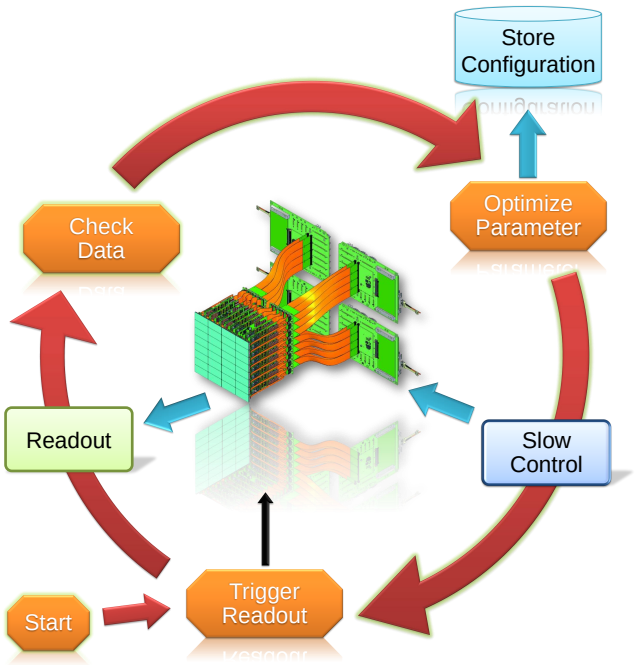


Figure 10.4.: **Concept of the pixel parameter trimming.** An improved pixel configuration is iteratively determined from the digital output of the detector until the trimming goal is achieved.

circuits can not be described in this thesis, only the basic principles of the front-end will be mentioned which are directly linked to the discussed trimming parameter. In order to avoid going too much into detail, a more or less phenomenological description of the effects is given. For insights into the design and detailed description of the front-end circuits the reader is invited to have a look on the following publications: [73, 70, 79, 80, 75, 81].

The goal of all presented trimming algorithms is the compensation of variation in the single pixel characteristics in order to homogenize the characteristics of the whole pixel matrix. All algorithms are tested and implemented to work with the available single Ladder cameras, consisting of 16 readout ASICs or 65K pixels.

Several trimming algorithms will be presented in this section. The following list gives an overview of the implemented routines:

- Finding of a suitable setting of the current compensation DAC in order to enable correct pixel operation.
- Trimming of the ADC gain for all pixels by changing the ADC ramp current
- Trimming of the pixel delay in order to align the offset peak centered in the current ADC Bin.
- Trimming of the front-end Gain using the ADC ramp current setting

- Trimming of the clock *deskew* in order to improve the DNL[3] of the pixel ADC

10.2.1 Trimming of the Bias Current Compensation

The DSSC front-end has been optimized for the current read out of the DEPFET sensors. In case of the DEPFET front-end, the bias current from the sensor is rather constant across the sensor and given by the applied sensor voltages.

In the later integrated MSDD front-end, the charge signal is translated by the $TGain$ transistor into a current signal before it enters the FCF. Assuming the gate-source voltage of this transistor is constant, it also carries a static bias current. The MSDD front-end can also produce a static bias current. Even if the double sampling mechanism compensates for remaining bias currents flowing into the filter, its performance, especially its dynamic range, suffers from large remaining or changing bias currents which don't origin from photon signals. In order to minimize the unwanted currents flowing into (or out of) the filter, a special automated current compensation circuit has been introduced which adapts for total compensation of the bias current. Since this circuit can only perform the *fine trimming* of the current compensation, a voltage DAC has been introduced, which is used for a first coarse adaption of the current subtraction. The trimming of this coarse DAC is described in this section.

With the assumption of static bias currents, the trimming of the coarse DAC would be straight forward. But in the real system, things are not ideal and the F1 ASIC also shows unintended behavior. Unfortunately, in the F1 readout ASIC, the automated current compensation circuit cannot perfectly compensate the bias current. The supply current consumption varies from the *IProg* phase to the *Burst* phase. Due to these variations, the compensation circuit adjust to the slightly different bias current during the *IProg* phase which results in an additional current flowing into the filter.

A proper coarse DAC setting can be found by evaluating the residual current flowing into the filter after the *IProg* phase. This can be measured in single integration mode.

As an additional complication, the voltage drop in a pixel is strongly dependent on the quality of the current compensation configuration in the surrounding pixels. As far as it is understood, the current consumption of a pixel varies with the current which is drained by the coarse DAC. If the coarse DAC is completely mis-trimmed, its current consumption rises in a way which leads to noticeable voltage changes. This means, if a valid setting for certain pixels has been found and the configuration in the surrounding pixels changes, the trimming of the coarse DAC has to be checked and probably adapted. This leads to an iterative approach in which the pixels showing the worst current compensation are stepwise identified. In the next step, the coarse DAC setting is adapted only in the worst pixels. This is repeated until all pixels show acceptable bias current compensation which is signaled by an output value in the target region.

The development of the mean digital output[4] of the single integration measurement for all pixels of an F1 ASIC is shown in figure 10.5.

At the end, the trimming algorithm got relatively complex and still requires manual guidance by the user and therefore can only be performed half automated. The behavior of

[3] Differential Non-Linearity

[4] Mean value of all SRAM addresses of a pixel

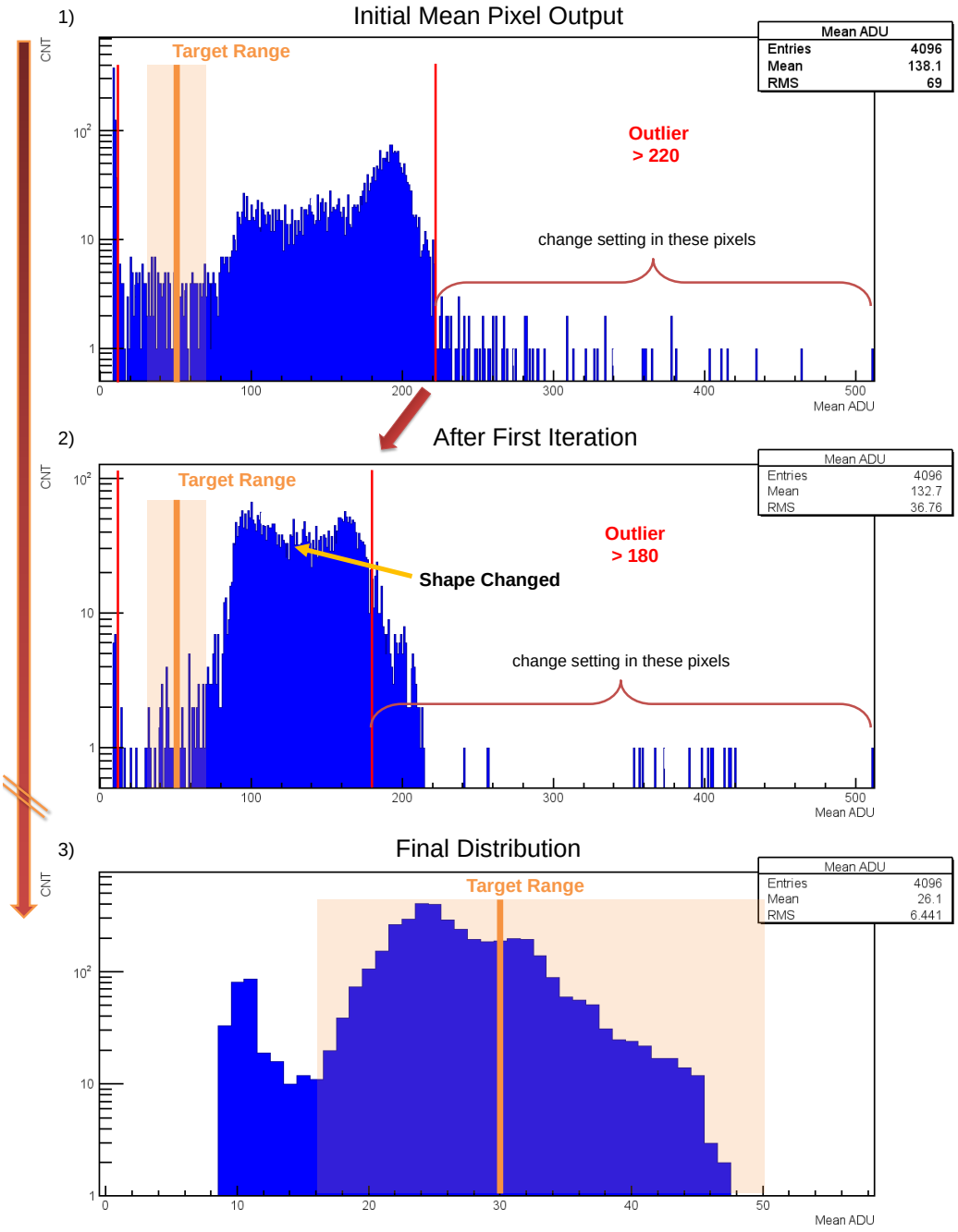


Figure 10.5.: **Mean digital output of a single integration measurement during the comp coarse (CC) trimming for 4096 pixels.** 1) Initial output, all CC settings the same. Limits for outliers can be defined to < 10 and > 220 . 2) After first trimming step, CC setting has only been changed in outlier pixels. The shape of the distribution has significantly changed. Limits are now < 10 and > 180 . 3) Final output after several iterations. Successful trimming since all pixels show an output near to the target region.

the readout ASIC has carefully been investigated and lots of ideas have been tried before the shown procedure has been found which at least mostly results in a working configuration also for the highest gain settings.

- **Step 1:**In order to measure the actual remaining current flowing into the filter that cannot properly be subtracted, the ASIC is set into single integration mode.
- **Step 2:**In single integration mode, the mean digital output for all pixels is measured and displayed in a histogram. From the histogram shape, the lower and upper limits defining the outlier pixels are determined by the operator.
- **Step 3:** The configuration is only improved in the outlier pixels. In general, the output of *all* pixels has changed. In the next step, new limits are defined until no further improvement can be achieved.

For shorter integration lengths, i.e. lower FCF gain, large residual currents can be tolerated without saturating the circuit. Consequently, if the selected gain setting is high and the target integration length is rather long, the trimming can be started using a shorter integration length in order to eliminate the 'early' outliers. This improves the behavior also with a longer integration length.

Especially for high bias currents, a start configuration with lowest current subtraction in all pixels results in fast trimming and also reasonably reproducible results.

Particularly in the case of high bias currents, a special start configuration, which contains the lowest current subtraction setting in all pixels, results in fast trimming and also reproducible results.

But the method has to be carefully applied. If for example the upper limit has been defined too low, and too much pixels change their configuration during the next trimming step, the configuration can become fairly bad and the distribution will expand a lot. This means in the most cases that the operator has to start from the beginning.

Despite the described difficulties, one advantage of the algorithm is that it can be applied to a whole Ladder at the same time. Additionally, if a suitable configuration has been found for a high gain setting, the same current compensation configuration can also be used for all lower gain settings. Nevertheless, the actual implementation takes lots of time to succeed and hence must be automated at some point.

Alternative Coarse DAC Trimming Methods

When operating ASICs with smaller pixel matrices everything becomes simpler since the supply voltages in the ASIC remain more stable and inter pixel dependencies remain negligible. Consequently, the trimming algorithm can avoid looking on the actual residing current in the pixel during the burst and implement the originally foreseen method. This method regards the actual voltage level during the *IProg* phase which can directly be digitized by the ADC and stored in the first SRAM address. Depending on the voltage value the current compensation can be defined accordingly. The software realization just looks at the resulting ADC value and tries to settle it in an adequate region. This approach is mainly applied when mini matrix chips are operated in the ASIC Test System, mainly for DEPFET operation but also in MSDD mode.

10.2.2 ADC Gain Trimming

In the pixel, the ADC gain can be varied by changing the ramp current strength which loads the sample and hold (SH) capacitor. For a given setting, the ADC gain varies from pixel to pixel and shows a broad distribution over the whole pixel matrix.

The ADC gain trimming algorithm has been developed in order to adapt the ADC gain distribution of all Ladder pixels to a defined gain within the granularity of $\approx 2\%$ of the ramp current trim parameter. An illustration of the algorithm is shown in figure 10.6. The resulting distribution of the trim parameters give an indication for the gain spread of all pixel ADCs. Furthermore, the remaining width of the gain distribution gives information about the ADC performance and if the ADCs work as expected. Furthermore, the development of this algorithm gave lots of experience using the readout ASIC, especially the dependencies of the gain to the configuration of the neighboring pixels became strongly visible. In the first implementations, these inter pixel relations have not been considered and the results have differed from the expectations. However, by introducing re-measuring and checking steps into the algorithm, these problems could be handled.

The trimming quality can be determined by measuring the distribution of the ADC gains after the algorithm has finished. The ADC gain can be measured by measuring an ADC characteristic using the internal voltage DAC which is implemented in the readout ASIC and fitting a linear function to the measurement curve. The slope of the measurement curve hence is given in units ADC counts (ADU) per DAC setting. A value of 0.0362 ADU/DAC setting corresponds to the nominal ADC gain, and has been defined as the target gain during trimming. The ADC slope values for all working pixels of a Ladder camera are plotted in figure 10.7. After trimming, apart from a few outlier pixels, 99.99% of the Ladder pixels show an ADC gain with less than 2% distance to the target value.

The final distribution of the ADC gain settings shows that 1/4 of the available gain settings have been assigned to compensate the pixel-to-pixel variations. This fits well to the expectations. Results of the ADC gain trimming algorithm for a single ASIC have been published by Christian Reckleben et. al. in [82].

10.2.3 Pedestal Position Trimming

Centering the position of the noise peak of a pixel in the respective ADC bin is critical for system operation and noise computations. Because the calibration of the system is based on the offset value of the ADC characteristic, all bin to photon count computations refer to this setting. Further, the precise position of the noise peak determines the threshold for detecting one photon or zero photons which is a very critical aspect if false positives have to be avoided. Finally, the calculations of the system's noise value strongly depends on the position of the Gaussian distribution with respect to the center bin.

Therefore, in each pixel a certain delay can be added to the ADC conversion that slightly shifts the offset value between 0 ps and 960 ps [81]. In total, there are 16 pixel delay settings which provide a granularity of 60 ps or $\approx 1/12$ ADC bin.

In order to find the best setting for every ASIC pixel, a method has been implemented which applies for all 16 pixel delay settings for the active gain configuration and automatically determines the best setting for the offset parameter. The best setting is determined by measuring one or several dark frames and computing the width (the RMS value) of the

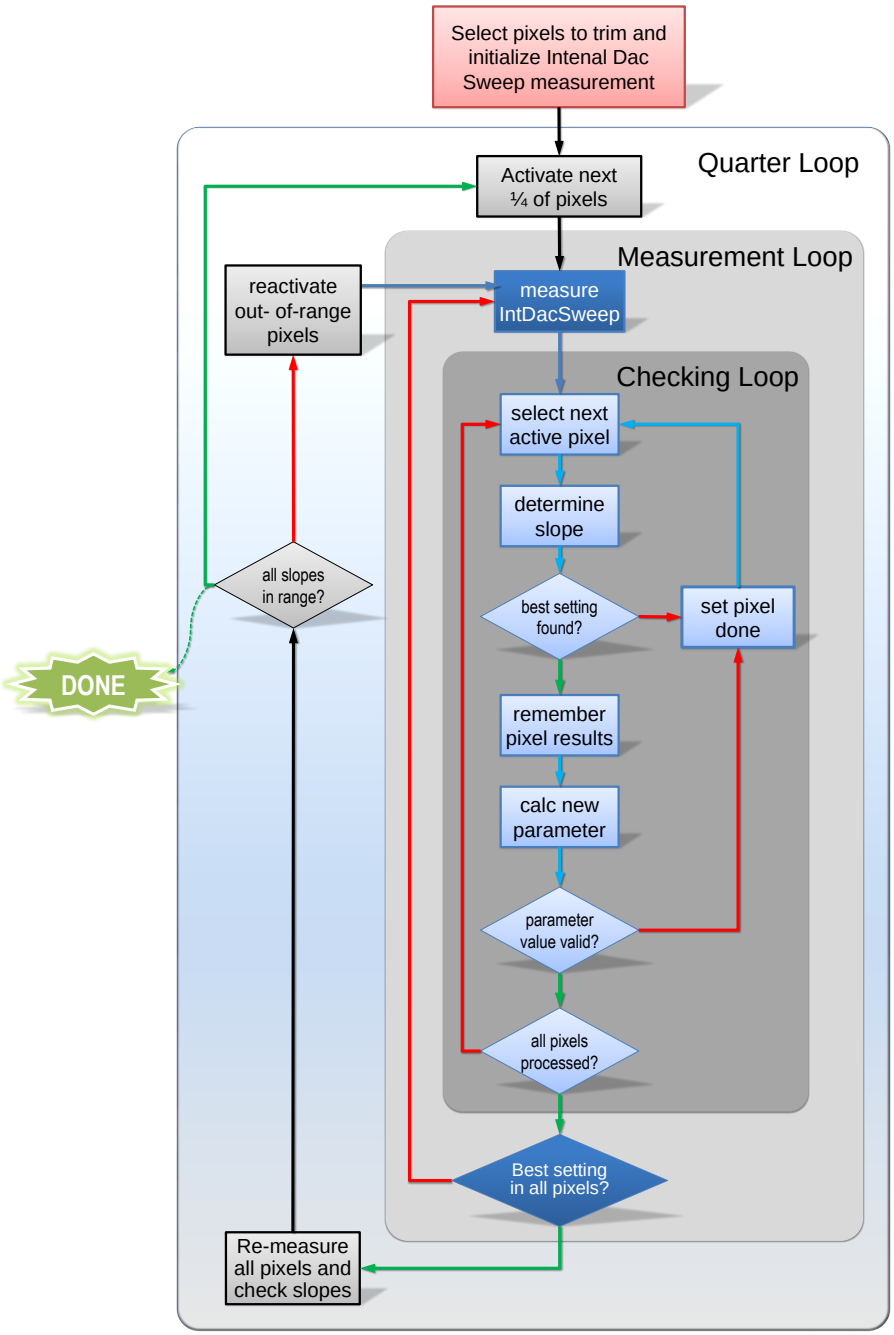


Figure 10.6.: **Scheme of the ADC gain trimming alorithm.** The pixel's ADC gain is determined by measuring a whole ADC characteristic using the internal voltage DAC of the readout ASIC. Generally, ten measurement points per ADC characteristic are sufficient for the linear regression. By comparing the computed slope to the target slope, the next, probably better fitting setting is computed by pixel. If the best value is found, the pixel is removed from further evaluation, until no pixels for analysis remain. An additional checking step tests at the end if the previously chosen setting is still the best. If not, the trimming for this pixel continues.

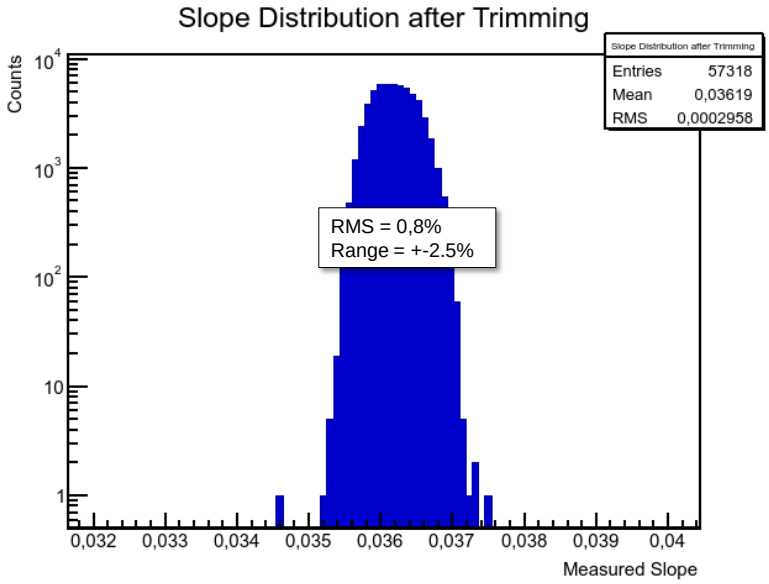


Figure 10.7.: **ADC gain distribution after trimming.** Target slope was 0.0362. Two full ASICs and some pixels have been switched off during trimming. 99.99% of trimmed pixels end in the expected 2% range.

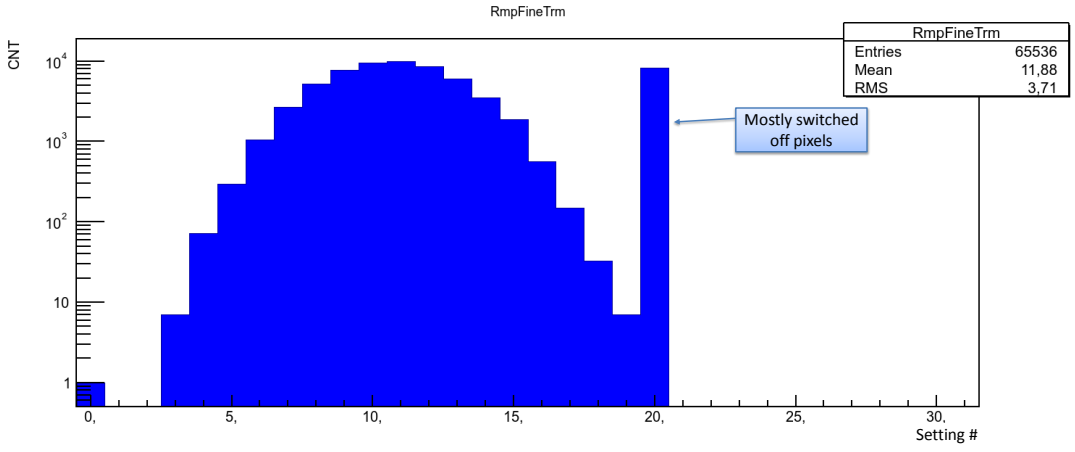


Figure 10.8.: **Distribution of the resulting ADC gain setting in the Ladder.** Start setting was 20. 16 of 64 (1/4) settings are assigned to compensate the pixel to pixel variations.

bias peak. For a noisy bias signal the best setting can easily be determined by selecting the setting which shows the smallest width. If the digitized signal is very sharp and the actual ADC bin is a wide bin, the bias signal can hit only one single bin throughout the whole measurement and consequently the RMS value becomes zero. If this happens for more than one offset setting, the algorithm selects the center of the longest section which shows zero RMS values in a row respectively the bin center.

As in the other trimming algorithms a checking step is added after the new configuration has been applied, which tests if the selected configuration actually is the best configuration and improves the configuration in some pixels if necessary.

The starting point and the resulting distribution of the RMS values from an example trimming procedure are shown in figure 10.9.

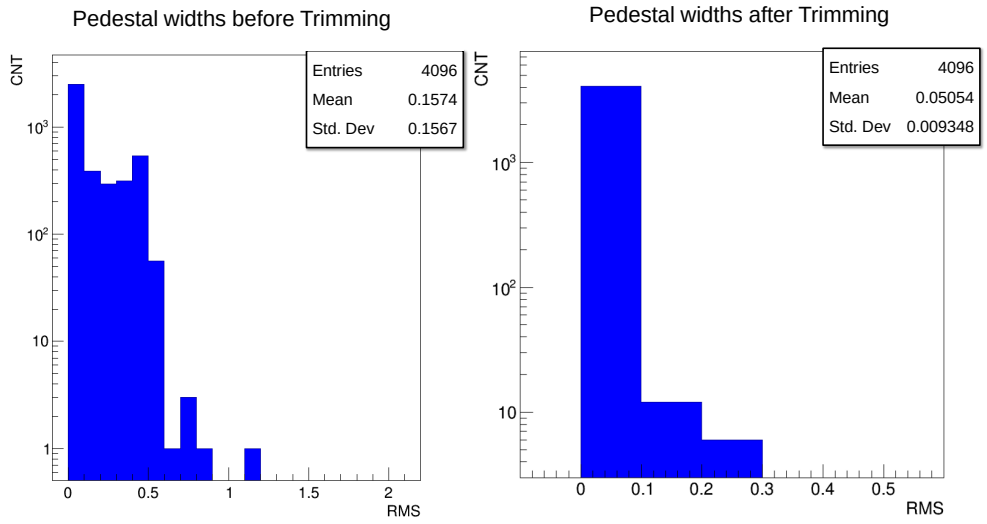


Figure 10.9.: **Result of the pedestal position trimming of one ASIC.** The pedestal peak RMS value of a pixel defines the quality of the trimming. The goal is to position the pedestal in the center of the actual bin which results in a low RMS value. If the bin is wide and the noise is low, the RMS value can become even 0.

10.2.4 Clock Deskew Trimming for DNL Optimization

The *clock deskew* setting changes the duty cycle of the clock which generates the Gray-code counter values. Consequently, the ADC bin width is influenced by this setting. Since an ASIC implements only one clock deskew circuit per ASIC half, the deskew setting changes always the ADC characteristic of 2048 pixels. For trimming 16 different deskew settings are available.

A homogeneous width of the ADC bins is important, especially for the digitization of small photon counts [100]. For single photon sensitivity, the assignment of photons and bins is in the lower region 1:1. Consequently, the system requires especially in the lower ADC count range equal bin widths. By fine stepping through the ADC characteristic, the bin width can be determined. Therefore, the same method is implemented which is used to measure the ADC gain but with much smaller step width of the DAC. To determine the

quality of the ADC bin width, the differential nonlinearity (DNL) of the ADC is computed for a defined bin range. The DNL is given by the measured bin width divided by the ideal bin width D_i minus one:

$$DNL_i = \frac{V(Bin_{i+1}) - V(Bin_i)}{D_i} - 1 \tag{10.1}$$

In order to evaluate the DNL value of all bins in the measured range an rms DNL value is computed. Since the mean value of all DNL_i is 0 by definition the rms can be computed as follows:

$$RMS_{DNL} = \sqrt{\frac{\sum_{i=0}^{N} DNL_i^2}{N}} \tag{10.2}$$

The optimal deskew setting is determined by a brute force approach. After measuring the ADC characteristic for all 16 clock deskew settings, the RMS_{DNL} value is computed for all pixels and the average $\overline{RMS_{DNL}}$ for both ASIC halves is computed. The deskew setting which shows the lowest $\overline{RMS_{DNL}}$ value is kept in each half.

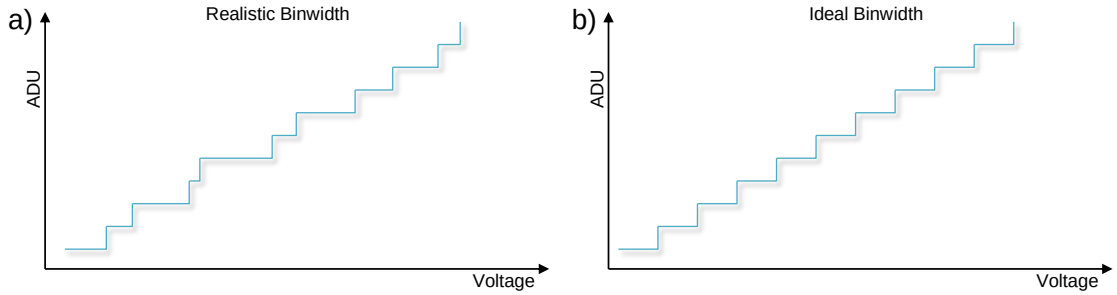


Figure 10.10.: **Realistic and ideal ADC characteristisc.** In a real ADC, the bin width varies. In the pixel ADC, a wide bin is normally followed by a thin bin.

10.2.5 Front-End Gain Trimming

The front-end gain is the most critical parameter for system performance. It strongly influences signal noise and dynamic range of every single readout pixel. Also the gain distribution over a pixel matrix on the ASIC or Ladder level is a very important parameter in order to evaluate the system design and its adjustment abilities.

For the final conversion of the digital output value to the number of collected photons, the pixel gain and offset has to be known. This step represents the system calibration. As a first step to calibrate the system to the proposed energy resolution, it is attempted to calibrate the system gain roughly to one ADC bin per collected 1 keV photon within the linear region of the detector characteristic. Details about the calibration strategy for the non-linear system characteristics are presented in [98].

In order to prepare the calibration measurements and as long as the actual system gain is unknown, it is necessary to adjust the pixel gain to a suitable value in order to produce

useful measurements using X-ray sources. The finding of this initial configuration is the goal of the presented trimming algorithm.

Therefore, the gain of all pixels of the matrix is adjusted to approximately the same value which is preferably high. In order to show the correct behavior and full functionality of the system, it is attempted to minimize the gain deviations down to the range corresponding to the granularity of the adjustment parameters e.g. the ADC gain. In the final resulting configuration the distribution of the adjustment parameters gives an indication for the gain variations in the readout pixels.

As listed above, two ASIC parameters are available to change the pixel gain: The ADC gain and the value of the FCF capacitor. Theoretically, both parameters can be used to adapt the pixel gain. But since the granularity of feedback capacitor setting is rather coarse compared to the ADC gain setting, changes on the capacitor size can only be used for a rough adjustment. In general, the FCF capacitor size is increased to adjust the system for high photon energies or slower operation modes with long integration times for the best noise performance. Additionally, a change in the feedback capacity size also changes the dynamic range of the pixel. Consequently, since the first measurements aim for a system characterization at 1 keV per ADC bin where a rather high pixel gain is requested, the feedback capacity setting is set to the lowest value in all pixels before trimming and the front-end gain is equalized by adjusting the ADC gain. The final gain deviations should lay in the order of the granularity of the ADC gain setting of $2 - 3\%$.

In DEPFET mode, there is also a third parameter, the gate-source voltage, which increases the gain of the DEPFET sensor. But this voltage can only be globally changed for a whole DEPFET sensor tile and is not relevant for the parameter trimming algorithms.

However, in MSDD mode, there are two additional configuration parameters which influence the gain of the pixel: In each pixel, an additional capacitor can be connected to the input node, which reduces the signal generated by the collected charge and hence the dynamic range. The second parameter is the voltage of the internal voltage DAC, which defines the reset voltage of the front-end transistor *TGain* and thereby the transistor's conductance $g_{m,TGain}$ (see section 5.3.3).

Some considerations how to chose the best setting for this global voltage are following at the end of this section.

Trimming of the Pixel Gain by Adjusting the ADC Gain

In order to homogenize the pixel performance of the whole pixel matrix, the ADC gain is adjusted in each pixel to compensate for the pixel-to-pixel gain differences and to provide a similar gain in all pixels. The difference to the previously presented ADC Gain trimming method is the signal source which is used to measure the pixel gain. In this method the front-end and the FCF is involved which induce additional complications.

The signal is generated by the internal pixel injection [101, 102], which injects a configurable charge signal into the MSDD front-end or a current into the FCF. The signal injection into the MSDD front-end is enabled if the trimming is for an MSDD camera, the later is important for the operation of a DEPFET sensor. The signal strength is adjusted globally for all pixels in parallel.

The trimming algorithm is kept quite simple in order to run stable and produce quick results. The easiest way to measure the gain of a pixel is measuring the output at a given signal and compare it to the bias value when no signal is applied. For simplicity, only the two points are acquired to determine the front-end gain. If a signal from the linear region is selected, the result should give a representative value which works for the presented trimming method.

Since all pixels show a slightly different length of the linear region and because no algorithm exists so far to automatically determine the range of the linear region, the signal, which is used during trimming, has to be selected carefully from the linear region by the operator before the trimming can be started.

The target value for the system gain is defined by the operator. As long as the exact gain value is unknown which belongs to the 1 keV per bin setting, the operator can define a setting which can be reached best by all pixels. It turned out that it is a good procedure to start with the highest ADC gain setting in all pixels and choose the target gain value as the value which can still be reached by the pixels showing the lowest gain. All other pixels are then trimmed to the same value by reducing the ADC gain.

In order to accelerate the gain adaptation, the next parameter value is computed with respect to the current difference to the target value. This small detail leads to a speed up of the algorithm of about $\approx 60\%$, compared to a constant step size of always ± 1.

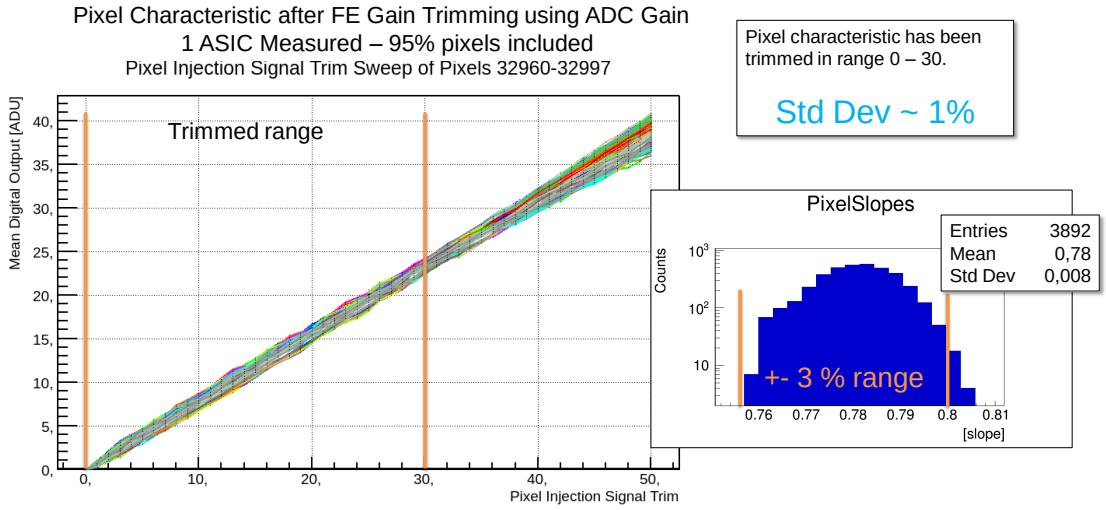


Figure 10.11.: **Pixel gain distribution after trimming.** One ASIC has been trimmed, and evaluated using the Pixel Injection. 95% of pixels could be trimmed in the target region $\pm 2\%$.

In figure 10.11, a measurement with fully trimmed Ladder is presented. One Ladder ASIC has been measured in order to present the trimming quality. In order to improve the visibility of the trimming quality only the best 95 % of all measured pixels are shown. The resulting 3% range to the mean value is a quite good result and confirms that the algorithm works as expected.

Optimization of the Global DAC Setting for MSDD Mode

Before the MSDD sensors can be operated with the F1 ASIC, the reset voltage of the MSDD front-end transistor *TGain* has to be adjusted by selecting a suitable setting for the global voltage DAC, which also determines the final gain of the MSDD front-end. But the setting of the DAC can only be set globally for a whole ASIC, since there is only one DAC per ASIC. Consequently, this parameter cannot be optimized separately for every pixel which means that most of the pixels cannot be operated at their optimal gain setting. The setting can be optimized by measuring the gain for several values of the reset voltage. For each value, the gain of all ASIC pixels is determined. The voltage which results in the highest average gain of the ASIC is considered as the best value.

In order to automate a series measurement over several global DAC settings, a set of well trimmed pixel configurations has to be provided in advance, since each global DAC configuration requires a special current subtraction setting in every pixel.

For measuring the pixel gain, the internal Pixel Injection is used.

In figure 10.12, the gain variations for three different pixels for different reset voltages are displayed. Each pixel shows a maximum at a different voltage setting. Also large gain differences are visible.

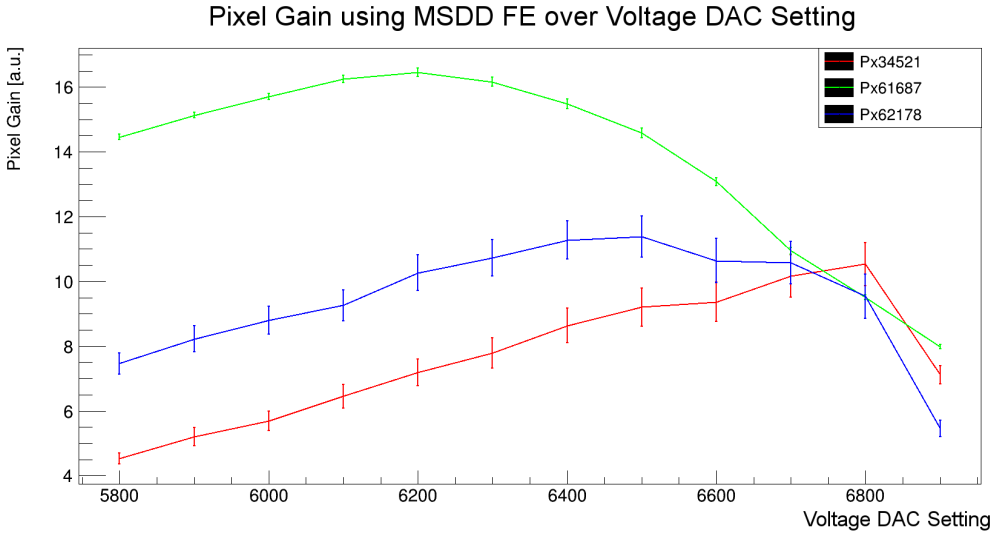


Figure 10.12.: Pixel gain values, measured by means of Pixel Injection measurements, over different reset voltage DAC setting. Large gain differences as well as different positions of the maximum gain are visible.

Discussion

The signal generation in the described front-end trimming methods is not able to produce identical behavior like a real photon signal from the sensor. For example, the charge signal which can be injected into the MSDD input node also shows a pixel-to-pixel variation of about 10%. Consequently, even if the ADC gain could perfectly adjust the pixel gain in all

pixels, an evaluation of a real photon signal from the sensor would produce a distribution which shows a spread of about 10%.

This means, that the internal signal generation methods can not directly be used for final system calibration. However, considering a tunable X-ray source, one can theoretically calibrate the Pixel Injection which theoretically allows to adjust the pixel gain indirectly.

11

The DSSC Karabo Devices

In the final environment at EuXFEL, all detector control will be integrated into the Karabo framework. The software framework controls and monitors all devices, from the vacuum pumps to the experiments, up to the large 2D detectors, which are required to operate and monitor the beamline.

To operate the DSSC megapixel detector at the EuXFEL beamline, a Karabo device is required which enables seamless integration into the XFEL DAQ and calibration environment (see section 4.3.2).

Since the megapixel system has not been available during this thesis, the integration of the smaller DSSC Ladder system is presented. However, an outlook is given how the integration of the megapixel system can be realized.

In order to gain realistic experiences using the Karabo framework, the whole control and DAQ environment is required, for instance, to perform measurements and online data monitoring.

While the proposed infrastructure from XFEL is still under development, own Karabo devices have been developed to enable the usage of the developed Karabo control device during realistic conditions.

The concept of the developed Karabo measurement environment is visualized in figure 11.1.

11.1 The Karabo::DsscPpt Control Device

The *Karabo::DsscPpt* (KPPT) defines the interface to the DSSC detector. It will be the main control device to operate the detector during beamline experiments. Thus, it has to provide full access to all detector hardware in order to initialize and operate the detector. This includes automated initialization sequences, status monitoring and error handling. Since it is only a control device and XFEL has its own implementation for the DAQ of the image data for the large 2D detectors the KPPT device does not implement any data receiving capabilities.

In order to access the available methods from the central library, it implements the directly derived *SuS::DSSC_PPT_API* class. This allows the Karabo device directly to re-use all well proven functionality. Consequently, critical parts like the initialization sequence,

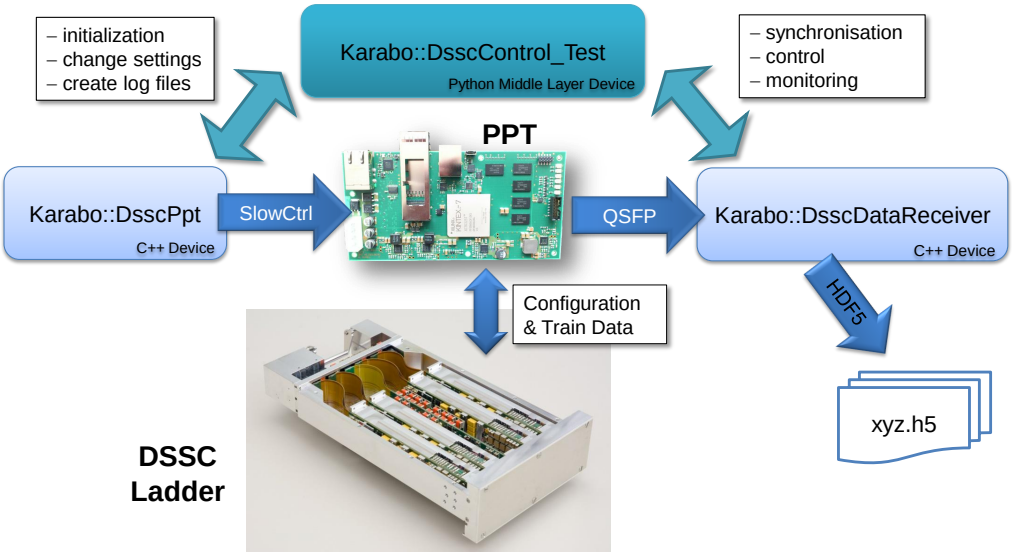


Figure 11.1.: **Overview of the DSSC Karabo devices.** The *Karabo::DsscControl_Test* (KCRTL) device implements master control of the The *Karabo::DsscPpt* (KPPT) control device and the *Karabo::DsscDataReceiver* (KDR) device, which implements the data reception and pixel reordering. The KCRTL provides a simplified interface to initialize and control the other devices. Its main feature are automated parameter sweeps including generation of measurement log files.

register programming and power switching are directly available in the Karabo framework. The KPPT device implements basically the interface to the Karabo GUI.

For simplicity only a subset of the GUI functions are also implemented in the Karabo device. A basic feature is the access to the system registers. Because of the large number of different register values, a generic interface has to be provided in order to not overflow the Karabo GUI with parameters. This generic interface is shown in figure 11.2. It can clearly not very comfortably be used by hand but it provides a nice interface for other devices and allows convenient grammatically parameter changes.

To integrate the detector system into the beamline run control a comfortable and easy to user interface has to be provided. This also includes an automated initialization sequence which brings the detector into a defined state where the system is ready to acquire images.

The Karabo framework implements a special state hierarchy which has to be implemented by the developed devices. Various states are defined which can be used to describe the current state of a device. The state machine of a Karabo device is important for monitoring and integration into the EuXFEL DAQ system.

The KPPT device required only few states which are stepped through during the initialization sequence. The implemented states are shown in figure 11.3. Always one DSSC Ladder can be controlled by the KPPT device. By selecting the IP address of the PPT, the Ladder to connect can be chosen. For initialization of the whole system, a configuration file must be defined and the InitSystem routine has to be triggered. Afterward, the detector is in an active state in which it is ready for image acquisition. By enabling XFEL Mode, the system directly reacts on trigger signals from the EuXFEL C&C system and starts to send

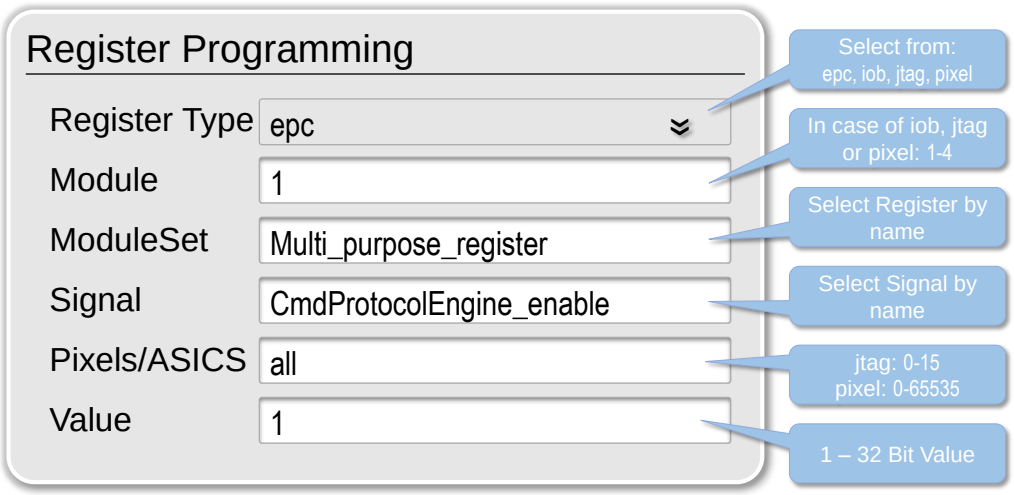


Figure 11.2.: **The register manipulation interface of the Karabo::DsscPpt control device.** Every single register can be accessed using the displayed fields. The register type can be selected. A convenient format has been defined to select arbitrary patterns of Modules or Pixels. ModuleSet and Signal names have to be defined by the user and cannot (yet) be selected.

out the acquired images. Alternatively the KPPT device implements a stand alone mode, which can be used for testing if the C&C system is not connected.

The KPPT is a control device only and does not implement any data receiving functionality. Since in the typical experiment setup, the control application is running on a different machine than the data acquisition both tasks cannot be implemented by the same device. The lack of data receiver capabilities limits the re-usability of the parent class member functions to the control functions only. Thus, pixel parameter trimming and measurement functionality cannot be provided by the KPPT device alone.

11.2 Karabo Devices for Beamline Measurements

Since measurement, such as parameter sweeps, are very important for experiments at beamlines and for calibration, two additional devices have been developed to implement the missing features in the Karabo framework.

The data reception and visualization is implemented by the *Karabo::DsscPpt* (KPPT) device, which is derived from the *Software Data Receiver* class. To perform automated measurements the KDR device has to be synchronized to the parameter changes initiated by the KPPT device and vice versa. This can easily be realized in Karabo using a Middle Layer Device. Therefore the *Karabo::DsscControl_Test* (KCRTL) device has been introduced.

Another big advantage of automated measurements is the possibility to automatically generate configuration log files after any configuration change which allows comfortable data logging and measurement reconstruction.

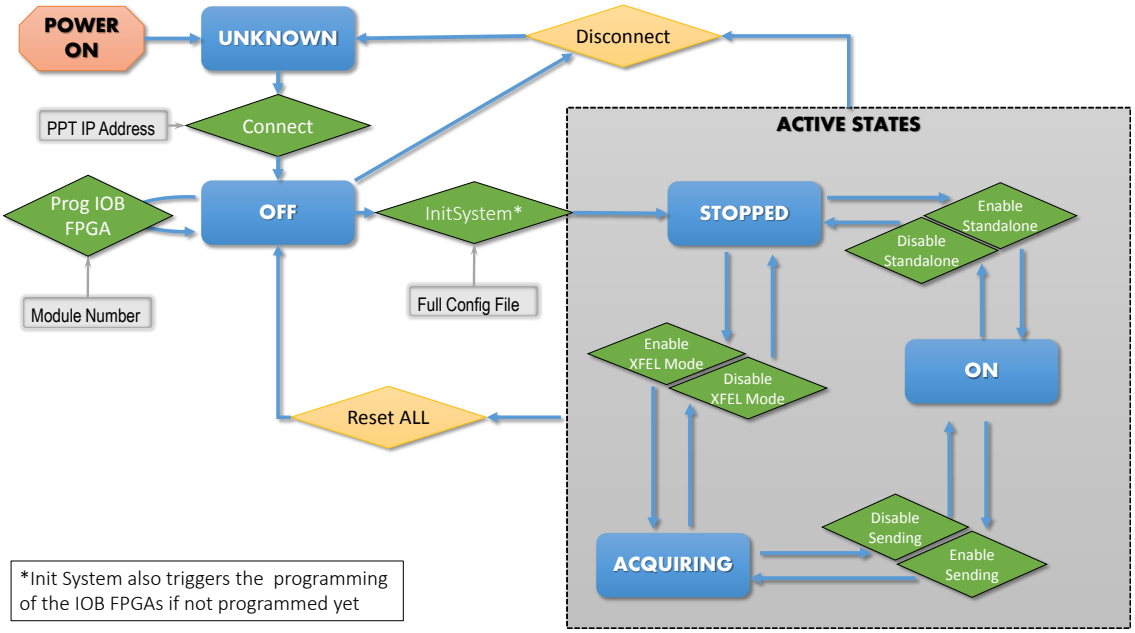


Figure 11.3.: **The Karabo::DsscPpt state machine.** The device starts in the *Unknown* state. By connecting to the PPT using a selected IP address the device gets access to the system. After power up, the IOB FPGAs are not configured and have to be programmed once. By starting the *InitSystem* method, the automatic system initialization is triggered. Afterwards, the system is ready for data acquisition and reacts on C&C telegrams. Also an alternative operation mode is available where trigger signals are internally generated. During all active states it is possible to safely reconfigure system registers or change ASIC configurations.

Data Output Channel	Settings and comments.
Full Ladder Frame	Frame number 0-799 to be shown can be selected.
Single ASIC Frame	ASIC number 0-15 and Frame number 0-799 to be shown can be selected.
Single Pixel SRAM Content	Pixel number 0-65535 to be shown can be selected. All 800 SRAM entries of the selected pixel are plotted.
Single Pixel Histogram for Spectrum Measurements:	Pixel number 0-65535 can be selected. Displays a histogram of all 800 SRAM entries of selected pixel. Content can be accumulated or reset after each train.
Single ASIC Histogram for Spectrum Measurements:	ASIC number 0-15 to be shown can be selected. Displays a histogram of all 800 SRAM entries of all 4096 pixels. Content can be accumulated or reset after each train.

Table 11.1.: **Visualization schemes of the KDR device.**

11.2.1 The Karabo::DsscDataReceiver device

This Karabo device implements the UDP packet reception and pixel reordering of the Train Data which is send out by the PPT via the 10 Gigabit-Ethernet links. Additionally, this device provides data recording into HDF5 files as well as several visualization and quick analysis functionality which are described in the later part of this section.

Like the KPPT device, this device is written in C++ and also profits from previous software developments since it implements the *KaraboDataReceiver* which is a derived version of the *SoftwareDataReceiver* class. Consequently, the device can directly re-use the existing features which have been presented in section 9.3. This means, all critical reordering and memory management functionality, which have been intensively tested in the Qt environment, can directly be used.

Additionally, the KDR device implements several adaptions and extensions for integration and control of the data receiver class. Most important extension is the packing of the data into the Karabo Hash structure in order to supply several p2p output channels. For each output channel, the device got armed with a number of tools for online data visualization and online data processing. Some of the implemented visualization outputs are shown in figure 11.4.

Several methods for data visualization are implemented. In order to send the data to the GUI or to other devices, a p2p output channels is created for each visualization method. The different methods are presented in table 11.2.1.

Several data correction and pre-processing methods have been implemented in the *DataAnalyzer* which allow removing of systematic effects from the data. The most practical methods have also been implemented in the KDR device to provide instant data correction. The different methods can separately be enabled and are applied for all listed visualization tasks in table 11.2.1:

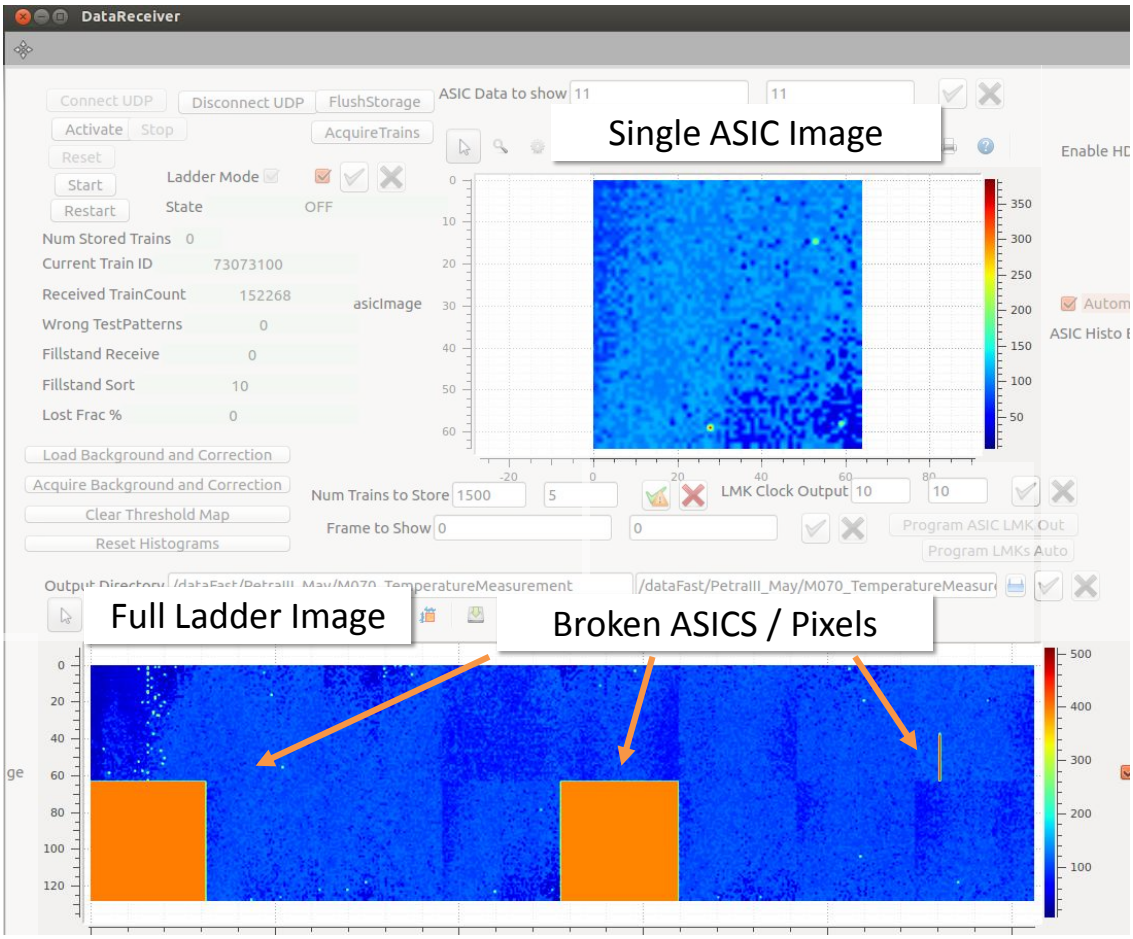


Figure 11.4.: **Section of the Karabo Data Receiver Scene.** Two Karabo visualization Widgets are shown: the full Ladder image and the single ASIC image. Received image data is directly displayed in the scene.

Correction Mode	Properties
GCC Wrap Correction	The GCC Gray Code Counter in the readout ASICs shows sometimes unusual behavior when the 9th bit of the counter toggles from 0 to 1, which leads to missing codes. This can be detected and easily be corrected by adding the 9th bit (+256) manually to the readout data.
Background Subtraction	A background map can be generated for every pixel and subtracted from the current frame. The background map is generated by computation of the mean pixel values of N frames. For convenient usage this map can directly be acquired or loaded from a previously stored HDF5 file.
SRAM Correction	Within the SRAM content of a pixel certain patterns or drifts can occur when operating with high front-end gain, which leads e.g. to unwanted spreading of the pedestal peak in a histogram. By acquiring a map for all pixels and each SRAM cell, a correction shift for each pixel and SRAM cell can be computed and applied to the data. An example can be found in the appendix E.6. For convenient usage this map can directly be acquired or loaded from a previously stored HDF5 file.
Threshold application	Considering very low photon flux, signals can be lost within noise and have somehow be emphasized. If a signal above the threshold is detected the photon count in the corresponding pixel is incremented. Mostly used in combination with the other correction methods.

Table 11.2.: **Data correction modes of the KDR device.**

On a high-performance PC, the application of all data correction modes shows no significant slowdown of the data reception and reordering since the data correction is only applied on the visualized data.

The KDR device provides real time data reception and pixel sorting at full speed as expected, since it implements the identical algorithm which has been used for the benchmark presented in section 9.3.1. If the HDF5 file recording is enabled, the reception speed decreases drastically since the file writing depends on the hard disk speed. Even with a fast Solid State Drive (SSD), which theoretically provides a writing speed of 1 GByte/s , only one train out of ten can be stored. Several approaches to accelerate the HDF5 files recording did not succeed, since the HDF5 library can not be used for parallel writing.

DSSC HDF5 Data Structure

Beside data reception and visualization, the KDR device provides also a method to store the image data into the HDF5 file format. Therefore, a special HDF5 structure has been defined which is shown in figure 11.5. This structure is used by all applications of the DSSC software environment.

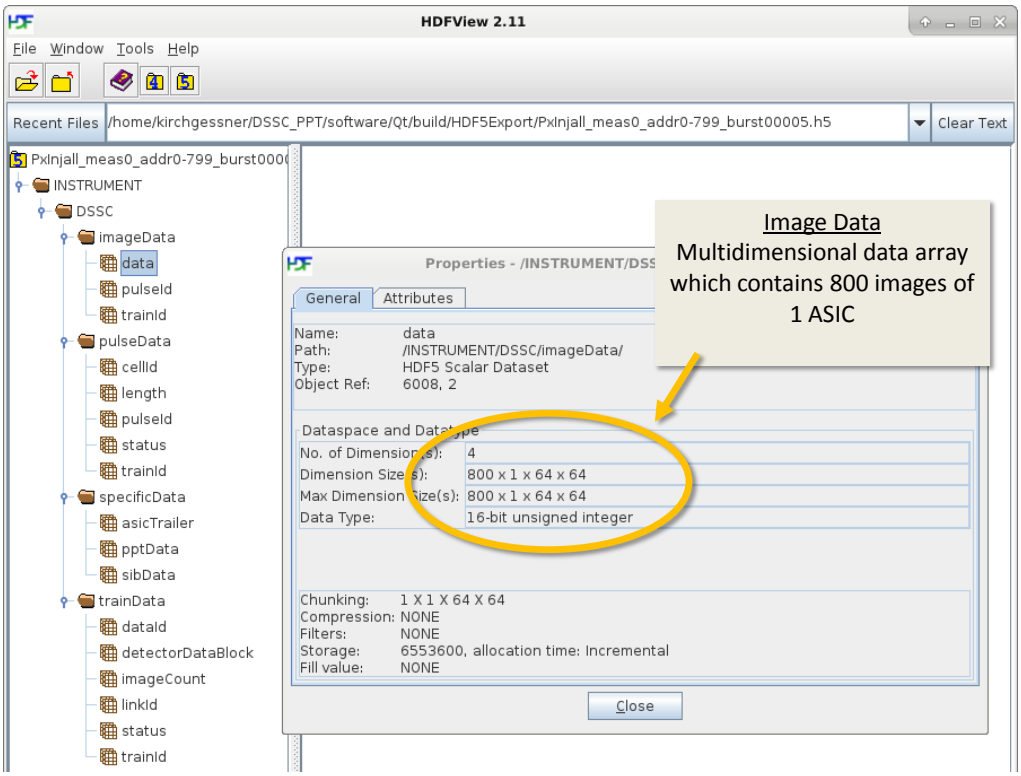


Figure 11.5.: **The DSSC HDF5 data structure.** Image data and meta data are contained in the hierarchical data format of the HDF5 file which allows simple access of the different data fields. The image data are stored in a multi-dimesional data array.

Data Storage Modes

Storage capacities and data management becomes a critical aspect while operating a system which produces as large amounts of data as the DSSC detector system. Even when operation only a single Ladder resp. one sixteenth of the megapixel system, the data rates are difficult to handle.

Regarding the size of a single uncompressed HDF5 file containing all 800 images from one pulse train which is about 106 Megabytes, the total amount of data which has to be stored during every minute sums up to 64.2 GB. At this data rate a common 3 TB hard disk is filled in below 47 minutes.

Several concepts have been introduced in order to deal with this problem:

For testing and also to decrease the required storage amount an alternative readout mode has been made available to the KDR device. Like in the Qt environment, the data sorting can be switched from Ladder readout mode to single ASIC readout mode. In the later, only the data of one selected readout ASIC is sorted and stored in the output data. This alternative readout mode of course reduces the amount of stored data by a factor of 16 and allows more efficient data storage if only one ASIC is to be measured.

Several procedures have been investigated to decrease the required local hard disk space:

1. Online data compression using system calls and gzip [103].
2. Enabling of the HDF5 built-in file compression.
3. Online data transfer to a distant large capacity network storage.

Data compression is always a trade-off between computing and acquisition speed and used storage capacity. If the computing power is larger than the available memory capacity (and speed), it is worth to think about online data compression. Of course the compression algorithm has to be loss-free.

1. gzip system calls: An automated data compression procedure using the Linux gzip application helps decreasing the occupied disk space. By starting system calls, a whole directory can be automatically be compressed without interfering the data reception process. The gzip compression reduces the final file size by a factor of three, but takes additional computing power on the machine which receives and reorders the images. A negative aspect of this method is that all data has to be uncompressed before it can be analyzed.

2. HDF5 compression: The HDF5 library provides built-in compression of several levels from 0 to 10. It can easily be enabled and the decompression is automatically applied when the file is opened. The compression is loss-free and reduces the file size also by a factor of three in all levels from 1 to 10. Since the HDF5 library does not support writing of several files at once[1], this compression method introduces a significant reduction of the HDF5 file generation speed and therefore an increase of the number of lost trains.

[1] Multithreaded writing could not be achieved during several approaches.

3. Data transfer: Assuming a large storage system which is connected to the receiving computer via a high speed network connection (1 GBit/s or faster), it is possible to transfer the acquired data directly to the distant storage. This could increase the effectively usable storage capacity a lot. While running in single ASIC mode the required data rate is decreased by a factor of 16. This means that a 1 GBit/s connection is already fast enough to transfer the acquired data instantly to the distant storage at acquisition speed which allows to acquired measurements independently of the local disk capacity. The data transfer can be realized by starting rsync [104] system calls [2].

Summary: Because both data compression modes provide similar compression abilities, the more convenient build-in compression of the HDF5 files has been kept in the KDR device. Since the problem of limited disk space could basically be solved by introducing the possibility to transfer the data to a distance computer during data acquisition, the data compression methods are not

11.2.2 The Karabo::DsscControl_Test Middle Layer Device

The Karabo::DsscControl_Test (KCTRL) device is a typical Middle Layer Device. It is implemented using the middle layer device python API provided by Karabo. It connects to the KPPT and the KDR devices and provide a simplified user interface to initialize the devices and to perform automated measurements. Several types of measurements are implemented which allow the most important ASIC parameter sweeps and most of the measurement functionality of the Qt environment. Therefore, a generic interface has been designed which allows to select one or two measurement parameters from a list which contains the most important ASIC parameters. The interface is shown in figure 11.6.

Since certain ASIC parameters require certain additional system configuration changes, the KCTRL device automatically enables required ASIC features to enable correct system behavior during the measurements. Additional configurations include, for example, changing several required bits of the pixel configuration which are required to properly enable the pixel injection.

The advanced configurations are automatically applied depending on the triggered measurement type.

In general, a measurement is defined by one or two selected sweep parameters and the number of trains that should be acquired per setting. For each setting, a separate directory is generated in the specified measurement directory in which the current configuration files, all Train Data files in the specified DSSC HDF5 format and one additional image data summary file are stored. The data summary file contains the pixel mean and RMS values of all measured pixels which are computed from all 800 SRAM cells of all acquired train of this setting[3]. The summary file can comfortably be used for a quick look at the measurement characteristics. At the end of a measurement, a measurement information file is generated containing all information about the measurement which helps to automatically find the recorded data files and measured parameters. The Measurement Information file can be read by software and is used for automatic data analysis or data extraction as shown in the next section.

[2] At DESY this concept has been tested using a computer cluster which provides additional 300 Tera-Bytes disk space, connected via 1 GBit/s network connection.

[3] Depending on full Ladder or single ASIC operation of the Karabo::DsscDataReceiver.

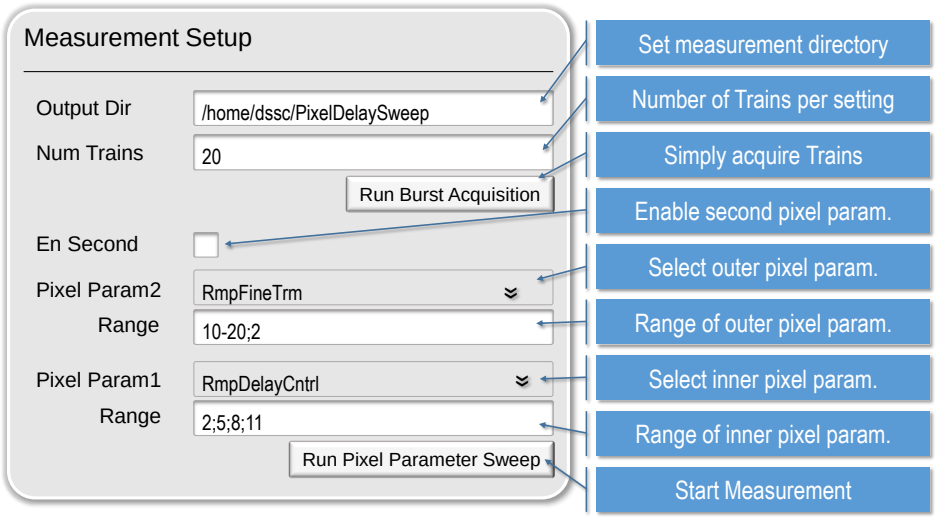


Figure 11.6.: **Simplified view of the measurement control provided by the Karabo::DsscControl_Test device.** Two different measurement types can bee triggered. A very simple acquisition which just stores the specified number of trains and a more complex measurement which allows to step through two independent parameters of the pixel register.

Available measurement types provided by the KCTRL device are:

- **Simple Train Acquisition**: Records N trains and stores the current system configuration
- **Pixel Parameter Sweeps**: Varies one or two pixel parameters in a nested loop.
- **Global Parameter Sweeps**: Varies one global ASIC parameter like the pixel injection signal setting or the internal DAC voltage setting. Like in the Pixel Parameter Sweeps, an additional pixel parameter can be enabled. The global parameter setting range is stepped through for each pixel parameter setting of the given range.
- **Burst Parameter Sweep**: Varies a parameter of the global control sequence. This can be used for example to move the time point of the image acquisition relatively to the incoming photons to acquire a so called *weighting function*. This is a very important measurement in order to verify the quality of the system synchronization with a pulsed photon source.

The integration of the automated measurement features into the KCTRL enabled the consortium to perform measurements using the available infrastructure of the vacuum chamber FENICE (see section 12.1) in which the image reception is implemented separately from the control, on a second high-performance computer.

11.2.3 Analysis of Karabo Measurements - The HDF5 to Root Converter

Changing the main storage file format always means many difficulties. Since measurement analysis has been started using root files and the *DataAnalyzer*, all existing analysis tools require root files as input file format. In order to keep the existing analysis methods, a converter tool has been introduced to read whole HDF5 measurements and pack everything into the root file structure.

The *DsscHDF5ToRootConverter* tool provides convenient usage of the *DataAnalyzer* for HDF5 data analysis. After selecting the Measurement Information file, it automatically packs the data of the selected subset of pixels into a correctly formatted root file which allows to use the nice data analysis method of the *DataAnalyzer* like it is familiar from the Qt measurements.

During beamtimes, this feature has been extensively used for quick view on the measurement data as well as final data analysis. By having this tool, all advantages of the Karabo framework and the DSSC devices can be obtained while at the same time all possibilities of the existing *DataAnalyzer* tool are available.

Besides the conversion of all data into the root file format, two additional conversion modes are implemented:

- For quick analysis the additionally generated image data summary file can be used to generate a root file which contains only an overview of the measurement. Especially for measurements which have stored large number of trains per setting, the root file generation is accelerated a lot. This tool can only be used to plot mean values over measurement settings. It is not useful for X-ray spectra or stability measurements.

- Direct histogram export into ASCII files. For each measurement setting, a histogram is generated containing all measurement values of a pixel. All pixel histograms are exported to a text file which only contains the hit bins and their hit count per pixel. This method greatly compresses the data files without losing too much information[4]. These files can further be imported in different analysis tools which are existing in other groups of the DSSC consortium.

11.3 Outlook for Quadrant and Megapixel Devices

During this thesis, the full megapixel DSSC detector has not been available. It is planned to be commissioned not until early 2018. Nevertheless, most required preparations have been taken into account to provide fast and easy extension of the control environment from single Ladder setups to Quadrant setups and finally also to **four** Quadrant setups. Due to the modular construction of the Quadrant, implementing the Ladder as the smallest part which can independently be operated, the differences of the single Ladder to the four Ladder setup, from the control software point of view, consists mainly in the addressing of the registers. This means, the configuration of different Ladders mostly differs in the addressing of the module, respectively the active IOB, which is transferring the JTAG commands and

[4] Train and pulse/SRAM cell number of data points is lost.

the calibrated configuration values for the camera head. Consequently, only few extensions in the control application are required to allow switching of the currently active module in order to reroute the control and configuration signals to a different module.

Similar considerations are valid for the control of a four Quadrant system.

Each Quadrant is controlled by one PPT. For initialization and control of a four Quadrant system the function calls for a one Quadrant system just have to be duplicated and propagated to four PPTs. This is a typical task of a Karabo Middle Layer Device: *Karabo::DsscMPix*

By re-implementing the basic functionality of the KPPT API, the *Karabo::DsscMPix* device can easily be integrated into the XFEL Run Control environment.

12

Software Development - Conclusion

Software development for a framework which is in an early state like Karabo brings always certain complications. Since Karabo itself is a new and self designed software framework from XFEL, it is under steady evolution. During the development phase of the Karabo device, a new major release of Karabo version 2.0 has been published. In consequence, all existing devices had to be adapted to the new framework version.

Furthermore, debugging and testing of new functionality is always more difficult and more time consuming in a complex environment than in a stand alone application.

Due to the special design of the software structure, it became possible to shift as much software development for the Karabo devices into the simpler Qt environment. By re-using all central functions of the created library, the implementation effort for the Karabo devices could be greatly reduced.

The implemented software package, which has been developed during this thesis, grew to a really complex system of methods and functions which provide a user friendly operation of the DSSC detector system.

The one-click initialization method became the central method to start the detector system. It brings the whole detector system automatically into a state where it is directly possible to acquire images and perform measurements, which improved the usability of the system a lot.

The measurement framework including the analysis functionality of the *DataAnalyzer* application has been a very successful implementation and has been widely used by different scientists in the DSSC consortium.

With the introduction of the automated pixel parameter trimming functions, it became possible to find a configuration with which a complete Ladder system could be operated also with the highest front-end gain modes which could never been achieved by hand.

The high data rates of the detector provide very challenging requirements to the receiver software. To receive the data from a 10 Gigabit-Ethernet, which actually utilizes over 99% of its bandwidth, is not trivial at all. If the large amount of data is also to be reordered at the incoming data-rate, the receiver software can only deal with the high data rates, by implementing multiple CPUs and efficient data management. It has been shown that if enough computing power on multiple CPUs is provided, the presented implementation of the Software Data Receiver is able to deal with the given data rates by assigning the different

tasks flexibly to multiple threads. The self-written thread safe queue in combination with a special managed ring buffer thereby plays a central role in the efficient management of the data sorting on multiple CPUs.

Part IV

Commissioning of the first Ladder Camera

12.1 First Steps with the Ladder Prototypes

When the first Ladder camera prototypes with working DEPFET and MSDD sensors have been assembled, the readout chain and control software have already been in a state in which the whole readout chain was functional. After only few adaption in the control software also the configuration routines worked reliably with 16 ASICs which allowed rapid commissioning of the first Ladder prototypes. An image which shows the successful configuration of a Ladder is depicted in figure 12.1.

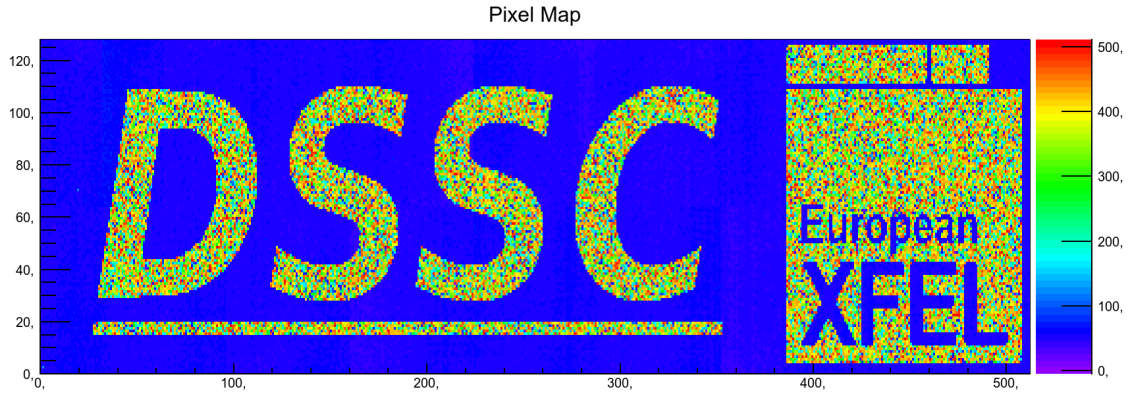


Figure 12.1.: **DSSC Logo, generated by a special configuration of the pixel registers.** Due to a special configuration of the power-down bit in the pixel registers, the DSSC logo has been generated. This image shows, that the pixel configuration is correctly assigned to the configuration stream and also that the pixel data is correctly received from the PPT and correctly sorted by the software.

The first Ladder camera, which has been delivered to XFEL in August 2016, implements a Mini SDD focal plane. It has been mounted in the vacuum vessel *FENICE*, which contains an active temperature control system and can be mounted to the vacuum system of a synchrotron beamline[105].

12.2 Measurements at the Petra III Beamline

The MSDD Ladder camera and the *FENICE* chamber have been moved to the beamline P04 at the Petra III synchrotron at Hamburg during December of 2016 and a second time to the beamline P65 also at Petra III during May 2017. During the two beamtimes, various important measurements have been performed with the goal to characterize the performance of the MSDD sensor in combination with the F1 readout ASIC with a pulsed X-ray source under realistic conditions.

Results of these measurement campaigns have already been published by the consortium in [105] (December at P04) or are in preparation for publication [106].

These measurement campaigns have been an important milestone for the progress of the DSSC project. It could be shown that a fully assembled Ladder system can be operated at a pulsed X-ray light source. The system has been synchronized to an external clock and

trigger source, and it has been shown that the image acquisition phase could be precisely aligned to the incoming photon pulses.

These beamline measurements have also been a demanding test for the implemented control software, the data acquisition and the readout chain. While the first beamtime has been operated using the Qt environment and a Karabo data receiver, which has been developed by Mattia Donato, during the second beamtime, the full functionality of the presented Karabo devices has been available.

During both beamtimes, the whole system worked very reliably without any problems which can be considered as a great success. During the second beamtime, the last small problems of the Karabo environment could be solved and its usability largely improved.

The KCTRL device showed its functionality during a variety of measurements. The presented Karabo devices worked flawlessly and their new functionality brought big advantages, compared to the first beamtime measurements in December 2016, when the Karabo devices have not been available yet.

The HDF5 file format could successfully be integrated into the data analysis work flow which leads to quick and efficient analysis of the measured data produced during the second measurement campaign. The KDR device run at sufficient speed on the implemented system to receive and sort 10 of 10 trains per second. Unfortunately, it has not been possible to accelerate the writing speed of the HDF5 library to utilize the full 1 GB/s bandwidth of the fast SSD hard disks which have been installed in the data receiver system. Only 1/10th of the expected data rate could be achieved while writing the data of all 16 ASICs to HDF5.

On the other hand, due to the reduced write speed, the stored data could be transferred directly to a distant storage cluster via the transmission methods of the Karabo devices. By sending the data directly to a cluster, it has been possible to collect much more data than physically available in the data receiver system.

The reliable operation of the system has further been confirmed during continuous measurements which have continuously been performed after the return of the *FENICE* chamber. A variety of measurements have been acquired over several weeks using the Karabo devices. During this time the Ladder system in *FENICE* ran continuously over several weeks without any single reset without any problem.

In order to show the reliability of the system in combination with the implemented Karabo environment, one measurement is presented in the following as an example, which summarizes most of the work presented in this thesis and as a proof that the component and implemented features are reliably working.

The following measurement shows a step-wise change of the 64 ADC gain settings, while the input voltage of the ADC has been systematically been changed. All pixels in the Ladder are measured in parallel. All 64 measurements of one selected pixel are plotted. The successful execution of this measurement and the generation of this result requires a very reliable configuration mechanism of the pixel and JTAG registers. Furthermore, the readout chain has to work over several hours. In total, 26880 trains have been recorded for this measurement which are distributed over 64 ADC gain settings over 21 voltage DAC settings and 20 iterations per measurement step. As can be seen in the plot, no single outlier was measured, since a false configuration or readout would clearly be visible in this plot.

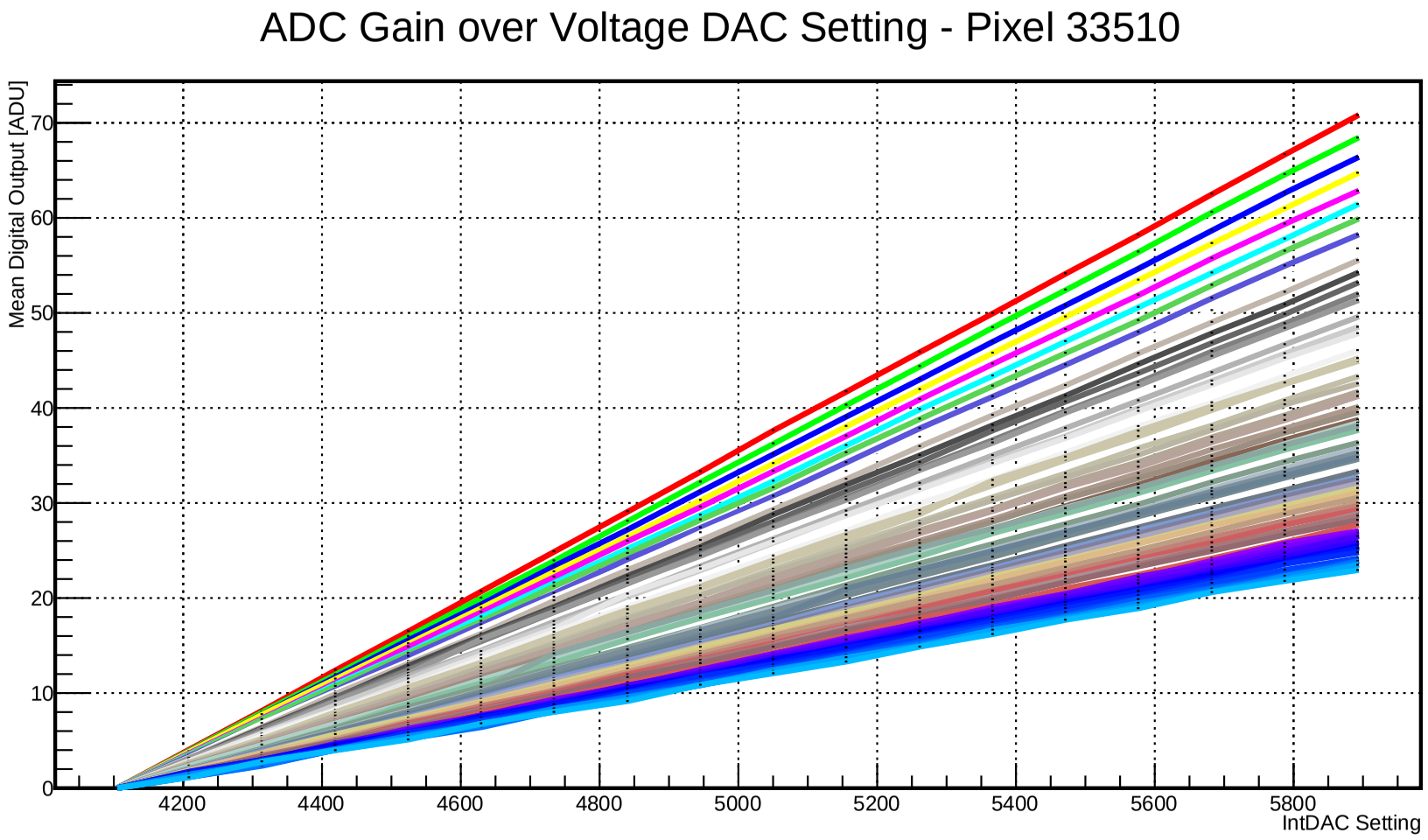


Figure 12.2.: **Measurement of all ADC gain settings.** 64 different ADC gain settings have been measured over a part of the ADC characteristic. Measured in the *FENICE* vacuum chamber with the MSDD Ladder. FE and sensors have not been active during this measurement. Red line = setting 0. Pixel 33510 has been arbitrarily selected. All pixels of the Ladder have been measured.

13

Conclusion

13.1 Conclusion

In this thesis, the DSSC project and central parts of the developed software and firmware environment have been presented. The challenging environment of the EuXFEL provides very demanding requirements for the implemented electronics to deal with the very high image and data rates. Using modern components and state of the art sensor technology, the DSSC detector implements small form factor devices which provide suitable performance to meet the given requirements.

In this thesis, the high-speed data path has been presented which transports the readout data from the focal plane to the outgoing fiber cables. A total data rate of over 134 GBit/s is transferred during default system operation. The stable operation and the reliable data transmission from the focal plane to the 10 Gigabit-Ethernet links could be confirmed.

All data links could be measured and showed error-free data transmission over several hours, or even days, also in a realistic vacuum and temperature controlled environment. The implemented IP-cores could be integrated into the data path in such a way that their maximum bandwidth is best utilized which is necessary, in particular in case of the Ethernet Engine, in order to achieve the required performance. The average bandwidth utilization of round 84% of the 40 GBit/s Ethernet link capacity is only one of the achievements of the data path implementation.

By introducing an elaborate address controller for the DDR3-1600 image buffer, the Train Data generation as well as the image wise reordering at full writing speed could be realized. This enables the generation of the required Train Data format, in which the images are sorted in chronological order. One of the most complicated steps to achieve this, has been the development of the algorithm which generates the Veto information that is required for the DDR3 address controller.

The correct sorting functionality could be confirmed by means of comprehensive simulations. The correct behavior has also been verified during a variety of measurements.

With the successful implementation of the control and synchronization firmware right

before the first Ladder camera prototypes became available, the testing and evaluation of the system performance in the final configuration could seamlessly be executed. The methods to configure the different detector parts have been reliably implemented as confirmed during long-term measurements.

A slight shortcoming exists with respect to the programming speed of the long JTAG chain registers passing through the pixel registers. While the desired programming speed of the JTAG registers would be at a clock rate of over 33 MHz, reliable register programming can only be ensured by a reduced clock rate of less than 4.5 MHz.

The complicated process to find a suitable class hierarchy really payed off as it offers a sustainable advantage for the future software development. The implemented class structure realizes an extended compatibility of the central library for different application environments which retrospectively justified the high effort necessary to rewrite large parts of the software.

The implemented inheritance structure finally allows all future developments to be automatically available in the various test setup environments and in both the Qt and Karabo application.

In order to provide a flexible and easy-to-use measurement environment, which is currently used by multiple groups in the consortium, a central measurement format has been introduced which enables the corresponding *DataAnalyzer* application to analyze the taken measurements under a variety of aspects down to the lowest level of single memory cells. The utilization of the *DataAnalyzer* enables the consortium partners to perform measurement analysis and visualization without the need of any code writing. The provided convenient and uncomplicated way to analyze measurements simplifies the gain of feedback from taken measurements and allows to quickly measure for example the performance of the ASIC circuits under a variety of aspects.

Thanks to the HDF5-to-root converter, the *DataAnalyzer* application can also be used for measurements taken with the Karabo environment and remains an important data analysis software in the DSSC project.

The new Karabo framework for beamline control was built with the purpose to integrate everything within one single software environment. Even if the usage of a newly developed framework is risky and comes always with its own teething troubles, the Karabo framework can be assumed to be a success. This is underlined by the positive experiences gained with Karabo and the Karabo devices, which have shown very robust and reliable operation during the second Petra III beam measurement campaign and also during several long-term measurements.

The tasks of this thesis have been further extended with the development of the complex algorithms which have been developed to automatically optimize the parameters of a large number of pixels. Trimming algorithms are required in order to find a suitable configuration which allows to use the the available Ladder cameras during the important beamline measurements and also to provide a suitable setting from which the system calibration can be started.

As presented at the end of this thesis, both measurement campaigns have been very successful, not least because the DAQ as well as the control software worked flawlessly. Also a robust synchronization of the system to an external clock signal and the photon pulses from the synchrotron have been verified. The two measurement campaigns at Petra III meant important milestones for the DAQ development and the commissioning of the first Ladder camera prototypes.

13.2 Summary of Own Contributions

During the last years, while this thesis was created, many different tasks could be accomplished. Central parts of the system firmware and software have been successfully developed. After the development of the data path which transports the image data out of the detector and of the control software to configure and control the system additional challenging tasks have been added to the list of responsibilities.

In the following, the contributions of the author to the DSSC project are listed. The detailed list is divided into two parts. The first part lists the contributions concerning the control firmware and the data path.

In the second part, the contributions to the software are listed. Of course, this list includes only a subset of the tasks which have been implemented during the last years. Mainly, the central features to operate the system and to generate the presented measurements are listed.

13.2.1 Contributions Concerning Firmware

The following tasks related to the system firmware have been implemented by the author:

- the register programming in PPT, IOB and readout ASICs from the PPT.
- a 1-to-3 de-serializer per ASIC channel in the IOB including ASIC data transmission failure detection, error handling and dummy data injection.
- the 3×3.125 GBit/s Aurora channel from the IOB to the PPT.
- the 10-bit pixel word reconstruction from the serial input streams and Gray-code decoding in the PPT.
- the DDR3 memory controller, which includes the finding of the correct device timing values and implementation of an automated reset procedure.
- an address controller and the integration of the memory controller into the data path.
- the online image sorting using the DDR3 address controller based on the Veto data.
- the Ethernet Engines including the required IP-cores and the packet generation.
- the system synchronization to the incoming photon pulses.
- the communication with the SIB.
- the transmission quality tests of the different high speed links.

13.2.2 Contributions Concerning Software

The following tasks related to the system software have been implemented by the author:

- the slow control communication with the Microblaze microcontroller including the SlowControlServer running on the Microblaze.

- the automated methods to initialize and power up the whole detector system including a special sequence, which is required to deal with the bug in the F1 decoupling capacitors.
- the efficient software representation of the control register types including the possibility to store and load configuration files.
- the multithreaded software data receiver which allows real time packet reception and pixel sorting at 10 GBit/s.
- the basis for automated parameter measurements and analysis, by introducing a fixed data format to store measurement data in root files.
- the *DataAnalyzer* application, for convenient data visualization and fitting of the acquired measurement data.
- the automated pixel parameter trimming functions for: current subtraction, ADC gain, pixel delay, front-end gain and clock deskew (DNL).
- the presented Karabo devices and all their features.
- the HDF5 file write and read wrapper classes.

13.3 Outlook

The main work for integration of the DSSC detector into the Karabo DAQ and calibration environment has just begun. When the integration of the LPD and the AGIPD are completed during the end of this year, all efforts will be concentrated on DSSC, to ensure the timely commissioning of the DSSC megapixel detector by spring of 2018.

Large effort is still to be done to integrate the complex trimming functions and further functionality concerning calibration into the Karabo framework. But since LPD and AGIPD are already gaining experiences with the Karabo infrastructure at XFEL, DSSC will be able to profit from previous developments.

Within the next months, the assembly and mass production of components will start. The first Quadrant test stands are already set up and steadily used during various tests. All necessary extensions for the control central library are already implemented such that the implementation of a quadrant and later megapixel Karabo control device can quickly be realized. After the recent receiving of the first F2 ASICs, which implement an improved version of the MSDD front-end, the first measurements could confirm that the new design works as expected. Since the improvements in the F2 ASIC implement an automated method to subtract the base line current in the pixels, the most complicated preparation step will become obsolete. Further improvements promise to reduce inter-pixel dependencies which would improve the ASIC behavior a lot and make trimming much easier. All progress suggests, that the DSSC consortium in march 2018 will deliver an sophisticated camera system that meets the proposed characteristics.

Part V

Appendix

A

Appendix A: DSSC ASIC Pixel Numbering

In order to not lose the overview about the large number of Quadrants, Ladders, ASICs and Pixels of a megapixel system, a naming and numbering scheme has been introduced.

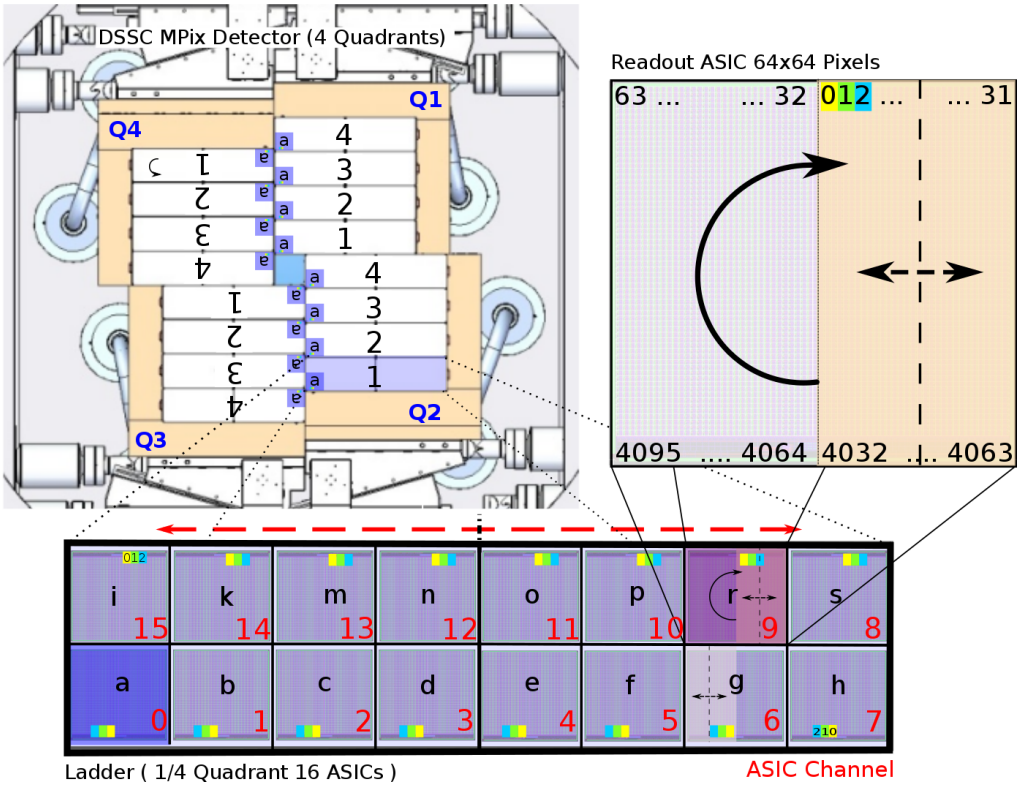


Figure A.1.: The numbering scheme of the Quadrants, Modules and ASICs of the megapixel DSSC detector.

Figure A.1 shows also the naming and alignment of the four Quadrants and their implemented Ladders 1-4. At the bottom, the layout of a Ladder and its readout ASICs is shown. The data of one Ladder is always transmitted over one of the four SFP Ethernet Links of

IOB Channel Number	SFP Channel Number (PhysID)	Y-Coordinates Left Quadrants	Y-Coordinates Quadrants Right
1	3	384-511	0-127
2	2	256-383	128-255
3	1	128-255	256-383
4	4	0-127	384-511

Table A.1.: **Assignment of the IOB Module channels to the SFP links of the PPT and Quadrant Y-coordinates**

one PPT. The assignment of the IOB channels to the SFP channels is shown in table A.1.

The alignment of the Ladders in the four quadrants is shown the Quadrant overview. Q1 and Q2 describe the default alignment as it is used for default descriptions in which Ladder ASIC 'a' is located at the bottom left corner of the Ladder. The quadrants Q3 and Q4 are rotated, consequently the ASIC layout of the respective Ladders are also rotated by 180 degrees which is indicated by the position and alignment of the small ASIC 'a' in the image.

The order of the pixel readout can be described as follows:

In the readout ASIC, the pixel data is read out of the full pixel matrix in parallel. For efficient and stable data readout, the data is shifted by slow clocks from the outer pixel columns to the middle, from top to bottom, where the serializer is located.

Consequently, the ASIC is read out from the middle to the outside, from bottom to top. As a results the order of the pixels in one ASIC half is mirrored.

Additionally, the upper and the lower row are rotated against each other by 180 degree. The bottom row shows the **default** orientation, and the upper row the **rotated** orientation.

In default orientation, the readout starts in the middle and continues to the left. In the rotated orientation, which is displayed in the upper right corner of figure A.1, the pixel readout continues to the right. After 32 pixel words, the read out continues in the middle to the other direction. This procedure is repeated from bottom to the top (in default orientation).

The data in the output stream are sorted in the order of the readout channel, as they are connected to the IOBs. The order, with respect to the position on the Ladder, is indicated by the red number in the Ladder image. In order to restore the correct sorting, the upper row has to be mirrored as indicated by the red arrow.

In this way, the pixel sorting and image reconstruction can be implemented by only a few geometrical operations:

1. a mirror operation of one ASIC half
2. a rotation of the ASIC in the upper row
3. another mirror operation, also only for the upper row.

B

DSSC Train Data Format

The DSSC Train Data Format describes the Bytes which are transferred out of the link from the PPT.

Since each link implements the same format, all numbers refer to the number of Bytes which are transferred on one 10 Gigabit-Ethernert fiber cable.

The Train Data which is transferred for one *Module* contains in total 104.870.704 Bytes for for a standard 800 images train. This amount of data is packed into 12815 Jumbo packets carrying 8192 Bytes payload.

The 5 blocks of the Train Data frame and the total number of Bytes of each block is listed in figure B.1. Each block is padded to 32 Bytes.

<table>
<tr><th>Train Header 64 Bytes</th><th>Train Header Bytes 64B</th><th>Header</th></tr>
<tr><td>Image Data

Image data only
104.857.600 Bytes
Incl TTP
104.960.096 Bytes</td><td>Event Data

8184B + 8B TTP per packet
12812 packets</td><td>Event Data</td></tr>
<tr><td>Image Descriptors
800 x 16B =
12800 Bytes</td><td>800 Pulse IDs
800 Cell IDs
…</td><td>PPT Data</td></tr>
<tr><td rowspan="5">Detector Specific Data

416 Bytes</td><td>PPT & SIB Status 160B</td><td>PPT Data</td></tr>
<tr><td>ASIC Temperature 64B</td><td>ASIC Trailer Data</td></tr>
<tr><td>Veto Cnt 64B</td><td rowspan="3">From all 16 Channels</td></tr>
<tr><td>ASIC Config XOR 96B</td></tr>
<tr><td>Test Pattern 32B</td></tr>
<tr><th>Train Trailer 32 Bytes</th><th>Train Trailer Bytes 32B</th><th>Trailer</th></tr>
</table>

Figure B.1.: **DSSC Train Data Format.**

C

Communication Protocol and interfaces in the Ladder System

In the following, all implemented commands are listed which are understood by the SlowControlServer.

This gives an overview about the intelligent functionality of the server application running on the Microblaze and its capabilities.

Only one command can be executed in parallel. In general, the SlowControlServer receives a string and decodes it after execution a return value is transmitted back to the sender which allows synchronization of the processed.

The transmission protocol is as follows:

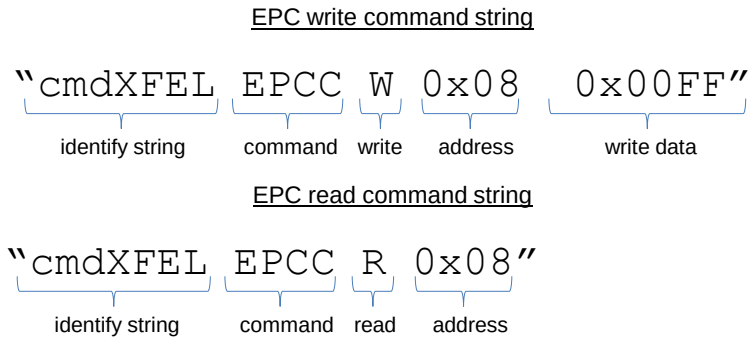


Figure C.1.: **Basic EPC read and write commands.** These two commands define the minimum set of TCP commands that are needed to configure the whole detector system. However, in order to improve the efficiency of the communication several other commands are available that are understood by the *SlowControlServer*. All commands must start with the identify string that protects the slow control server from bad TCP packets. The identify string is followed by a four capital command string which is followed by the command options.

Command	Parameters	Return Value	Description
EPCC	R Addr | W Addr Data		EPC Command. direct read or write command to access the PPT FPGA registers. Data access is only 4 Byte wise. Registers containing more than 32 bit are accessed by consecutive writes/reads.
IOBC	R IOBNr Addr | W IOBNr Addr Data	0 data	IOB Command. Shortcut to send directly read or write commands to the IOB. IOBNr = 1-4, Addr = uint16_t, Data = uin32_t
SIBC	R|W		SIB Command. Direct read/write access to the UART interface towards the SIB. F = flush UART rx FIFO. The SIBC handles the manual uart control enable flag automatically. In theory the SIB UART could be controlled using the EPCC command.
EXEC	String	0	Execute shell command. Direct access to the Microblaze busybox shell. Used e.g to start the IOB FPGA programming.
SPIC	Data		SPI Command. Deprecated user interface for SPI control. Was introduced to control the external DAC on the Test System Mainboard. Data is a hex stream.
RPSC	Data		RS232 Command. Depreciated command to send commands over the RS232 interface of the Microblaze. Data = ASCI. characters are directly send to '/dev/ttyUL1'.
FLFI	-	0/1 = OK/Err	Update Firmware flash ROM. Triggers firmware flash ROM update with the content of the file in '/tmp/DSSC_PPT_TOP.bin'. Firmware flash update takes about 30 minutes.
FLLI	-	0/1 = OK/Err	Update Linux flash ROM. Triggers linux flash ROM update with the content of the file in '/tmp/simpleImage.xilinx.bin'. Linux flash update takes about 20 minutes.
UPIF	-	0/1 = OK/Err	Update IOB Firmware. Overwrites the original IOB firmware file in '/opt/IOB_Firmware.xsvf' with the file in '/tmp/IOB_Firmware.xsvf'. Afterward, the IOB FPGA has to be programed e.g. using an EXEC command.
UPDS	-	0	Update Software. Moves all executables from /tmp/ to /opt/ and overwrites existing files. Does not restart the SlowControlServer.

Table C.1.: **Commands of the slow control server.**

Command	Parameters	Return Value	Description
REST	-	0	Restart FPGA. Reprogramming the FPGA and rebooting the Microblaze
BURS	-	0	Manual trigger of one burst sequence including readout.
READ	-	0	Manual trigger of one readout sequence only.
TEST	-	0	Manual trigger of one test pattern sequence including readout.
SIGL	-	0	Trigger Single Cycle: A sequence of single cycles is triggered. The number of cycles can be defined in the Single_Cycle_Register: iterations. In contrary to the BURS and READ command this command can also be used in continuous_mode or XFEL mode.
DCTW	numBytes		Direct Write: This command initializes a direct write command to the PPT FPGA logic. It was introduced to transfer large number of configuration bytes as required in the JTAG configuration. compared to the transfer of the data via FTP the direct write is about 2 times faster. Especially the pixel register programming is accelerated a lot by this function.
DCTX	Byte[;Byte]		Semicolon separated list of command bytes in hex values. Number of command bytes must fit to number of announced numBytes in the previous DCTW command
STAT	-	0XXX_XXXX	Read System Status: command for regular status polling. 7 Bits are Valid:Return = 0 everything ok. Bit0: GlobalReset,Bit1: No IOB Programmed, Bit2: Aurora Not Locked, Bit3: PPT PLL Not Locked, Bit4: All PRB Off, Bit5: IOB ASIC PLL Not Locked, Bit6: Sending Dummy Data
SIBR	-	0;Byte;	Read SIB Register: Read all 72 Bytes from SIB register.After a leading 0 all data bytes are appended as hex values to the response string.

Table C.2.: **Cont. commands of the slow control server.**

Command	Parameters	Return Value	Description
PLL0	[I/E/R][S/X]	6-Bit-STATUS + String	Program FPGA PLL. Deprecated function to program PPT FPGA PLL. I = internal (inside FPGA), E = External (dedicated PLL IC), R = read back PLL Status, S = Standalone (Clock is generated by oscillator on PPT), X = XFEL Mode (clock is received from C&C). The answer string contains the current configuration and status bits in one status word and also as a human readable string:PLL0 6-Bit-STATUS PLL CE: 1/0 PLL LD: 1/0 PLL MUXOUT: 1/0 FPGA PLL LOCKED:1/0 CLIN_SEL: 1/0 CLOUT_SEL: 1/0
JTG0	-	0/1 = OK/Err	Send JTAG Commands from file. If available all bytes listed in the file in /tmp/cmdsFromSoftware are send to the selected JTAG Engine. This the file has to be transmitted in advance via FTP to the Microblaze file tree.
JTGC	Chain# Bytes	0/1 = OK/Err	Send direct JTAG Command, similar to DCTX but only to send data to JTAG Enine 1-4
JTGW	Chain#	0/1 = OK/Err	Wait JTAG Done. Depending on the JTAG Clock Speed the JTAG engine requires certain time to send the configuration to the ASIC. The JTAG Engine signals if all data has been sent to the ASIC. This Command blocks all communication until the JTAG Engine is done. Chain# in 1-4.
FINT	Chain#	0/1/2	Do Fast Init. For quick initialization the JTAG Engine's FIFO can be loaded while the engine is disabled. This command triggers an automated sequence that switches the digital powers on and immediately starts to configure the readout ASICs. Therefore the JTAG Engine's FIFO has to be loaded with all required configuration data. Chain# in 1-4.
TEMP	-	FPGA Temperature	Read the PPT FPGA Temperature. Return value in Celsius.
SERN	-	uint64_t	Read PPT serial number from IC.
BILT	-	int	Read build date and time stamp of Linux image.

Table C.3.: **Cont. commands of the slow control server.**

Command	Parameters	Return Value	Description
RSTA	1/0	0	Reset All: EPC Registers / DDR3 Controller / Ethernet Engine / Data Receiver / IOB / ASIC. 1 = RESET 0 = release.
RSTE	1/0	0	Reset EPC Register
RSTD	1/0	0	Reset Datapath. Can be used to reset the Aurora link manually, if not automatically ready.
RSTR	1/0	0	Reset DDR3 controller.
RSTI	15-0	0	Reset IOB. One hot coded IOB reset.

Table C.4.: **Cont. commands of the slow control server.**

D

Code Example for Software Data Receiver Classes

The classes, which have been implemented to realize convenient and safe multithreaded computations and efficient data management for the *Software Data Receiver*, have been a major achievement to realize real time data reception and pixel sorting.

In the following examples several custom classes are used. A short summary is given in the following:

- **ThreadSafeQueue:** implementation of the thread safe access to a queue which can be accessed by multiple threads on both the producer and the consumer side. Several techniques are included which allow to make a consumer wait, if the producer has not finished and will probably add further data to consume. A queue can also be flagged as 'finished'. If the queue is empty a consumer knows then, that it can stop waiting and finish processing.

- **PacketQueue:** a derived class from the *ThreadSafeQueue*. It implements additional methods which allow simplified access to a packet objects filled with data from the UDP socket.

- **SlotManager:** implementation of a ring buffer of a fixed number of memory areas. By implementing two *ThreadSafeQueues* for 'free' and 'valid' slots it allows flexible access from multiple threads to the two kinds of memory areas. If the consumer is slower than the producer, the manager can change the assignment of a 'valid' slot to 'free', which allows to overwrite the data.

The implementation of the *SlotManager* allows efficient data management in the given case, that not every 'valid' slot has to be processed. If the data production rate is higher than the processing application can handle, data will be lost. But in order to not overflow the system, it has to be thrown away in any case. The advantage of this implementation is the flexible adaption to the speed of the consumers while providing always a memory slot to the producer side.

The following examples show how the classes can be implemented in the code and how the producer and consumer threads are connected to the implemented queue structures.

In the first listing 1, the scheme of the data receiver is shown, in which the received packets are filled into the *PacketQueue*. In theory the queue can also been filled by several threads and the code example would look exactly the same.

In the shown example, the *packetQueueSlots* is a *SlotManager* object, which defines the ring buffer that implements a special data management scheme. The *SlotManager* manages a fixed number of memory areas, which have previously been allocated, and internally marks the different areas as 'free' and 'valid'.

For the data reception the slots of the *SlotManager* are implemented as *PacketQueue* objects, but can theoretically be defined as any type.

The data receiver can pick a *PacketQueue* from a *Free Slot* and simply fill the received packets from the socked into the retrieved *PacketQueue*. At the end, when the last packet has been received, the filled *PacketQueue* is returned to the *SlotManager* which assigns it internally to the 'valid' slots.

```
1   // for each Train to be received
2   void fillPacketQueue()
3   {
4     PacketQueue * currentQueue = packetQueueSlots.getNextFreeSlot();
5
6     while(true){
7       auto & currentPacket = recvPacket(m_udpSocket,currentQueue);
8
9       if(currentPacket.isValid())
10      {
11        currentQueue->increaseValidPacketCount();
12        if(currentPacket.isLastPacket()){
13          packetQueueSlots.addValidStorage(currentQueue);
14          break;
15        }
16      }else
17      {
18        currentQueue.reset();
19        waitForNextTrain();
20      }
21    }
22  }
```

Listing 1: This shows, how a *PacketQueue* object is filled with data packets. The *packetQueueSlots* member keeps a number of pointers to previously allocated memory areas, which can be filled with data of a full train without the need to allocate new memory.

The unsorted packets in the *Valid Slots* have to be sorted which is implemented in the next step. In listing 2 the connection of several sort threads to the *SlotManagers* which hold the slots for the *PacketQueues* at the receiver side and the *Train Data* objects for the application is shown.

The number of threads have to be signaled to the *PacketQueue* before the sorting is started. The *sortPackets* function should be a thread safe static function.

```
1   void sortReceivedData()
2   {
3     while(run){
4       PacketQueue * validPackets = packetQueueSlots.getNextValidSlot();
5       TrainData * freeTrainData = trainDataSlots.getNextFreeSlot();
6
7       validPackets->open(sort_thread_count);
8
9       vector<thread*> sortThreads;
10      for (int i=0; i<sort_thread_count; ++i){
11        sortThreads.push_back(new std::thread(&sortPackets,validPackets,freeTrainData));
12      }
13
14       // wait for all sort threads have finished
15      for(auto sorter : sortThreads){
16        if(sorter->joinable()){
17          sorter->join();
18        }
19        delete sorter;
20      }
21
22      trainDataSlots.addValidSlot(freeTrainData);
23      packetQueueSlots.addFreeSlot(validPackets);
24    }
25  }
```

Listing 2: This shows, how slots from the *SlotManagers* can be distributed to several sort threads. After processing, the slots have to be returned to the respective manager.

In the next listing 3, the connection of an application to the *SlotManager* is shown. In this way, an arbitrary number of applications can be connected to the manager. However, a problem occurs if the number of connected applications is larger than the number of allocated slots. The ring buffer requires always one slot which is flagged as free or valid. If all slots are assigned to an application, no slot remains and a producer thread has to wait until an application returns the memory back to the manager.

```
1   void processTrainData()
2   {
3     while(run){
4       currentTrainData = trainDataSlots->getNextValidSlot();
5
6       process(currentTrainData);
7
8       trainDataSlots->addFreeSlot(currentTrainData);
9     }
10  }
```

Listing 3: This shows, how an application can be connected to the *SlotManager*. In this way any number of application can be connected for parallel data processing.

The last listing shows the *sortPackets()* function as an example for a function which can be executed by the sort threads. At the input, it gets a *PacketQueue* object and a pointer to the Train Data memory area. In theory, both objects have to implement thread save access, since the function is executed in parallel by multiple threads. In case of the *PacketQueue*, the thread safe access is implemented internally which allows multiple threads to fetch packets from the front of the queue. Since all threads sort data of different packets, each thread writes to a different area in the Train Data memory. Consequently, the thread safe access is realized by the access scheme to the memory.

```
1  void sortPackets(PacketQueue * validPackets, uint16_t * freeTrainData)
2  {
3    //sorting function just pop next valid packet and sort
4    PacketData *act_packet;
5
6    // if the queue is empty, all packets have been sorted.
7    while( (act_packet = validPackets->pop_front()) != nullptr ){
8      sortPacketData(*act_packet,freeTrainData);
9    }
10
11   //close queue to notify that sorter has finished
12   validPackets->close();
13 }
```

Listing 4: This shows an example for a function which is connected to a *ThreadSafeQueue*. In this way any number of application can be connected for parallel data processing.

E

The DataAnalyzer Application

The *DataAnalyzer* is a software tool which offers a lot of functionality to analyze the data from measurements using the DSSC test and detector systems.

An image of the graphical interface is shown in Figure E.1.

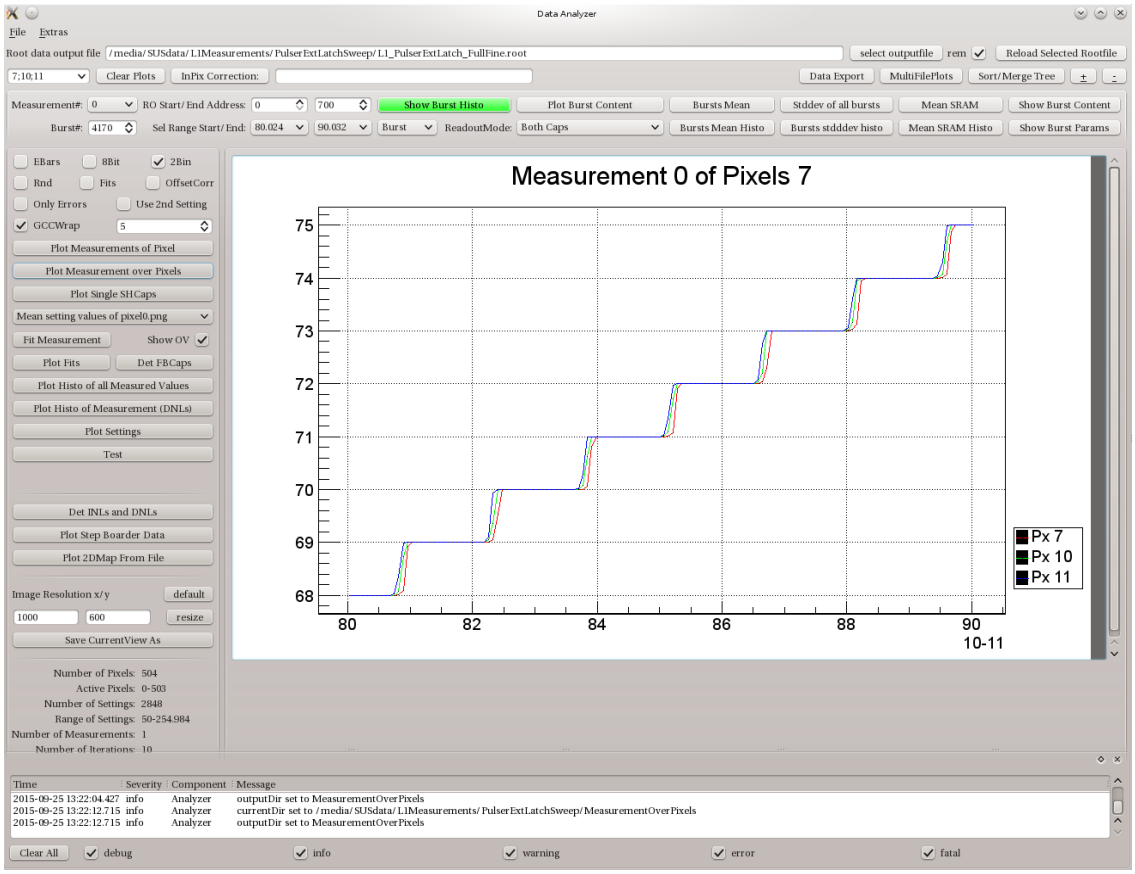


Figure E.1.: **Graphical user interface of the *DataAnalyzer*.**

E.1 Data Structure

The structure of the stored data has some clear requirements. It has to represent the structure of the system which is measured. Further, it has to be optimized to plot and analyze parameter sweeps which are pixel-wise recorded. For example if some unexpected effects have been measured, it must be possible to find some additional information in the measurement data about the system. To modulate this, not only the ADC values of the measured pixels have to be stored but also additional configuration data.

All the first measurements had the goal to investigate certain behavior of single pixels on single configuration parameter changes, like e.g. the change of the *IRamp Current*, or the *Pixel Delay* setting. Consequently, the data structure was designed to sort the data for each pixel by setting. This allows fast access to all data which has been recorded for the same pixel.

Later, a second Data Structure was introduced to accelerate data analysis of spectrum measurements. These measurements are characterized by large numbers of iterations with a static configuration during which no parameter is changed. The description of both data structures can be found later in this section.

E.1.1 Root structure

For recording of image data in combination with configuration data, the root library was used. This library implements advanced possibilities to store structured data and has the big advantage that it automatically compresses the data during recording. Furthermore, root offers a lot of plotting and analysis functionality. Finally, it is compatible to the Qt library with is used to implement the *DataAnalyzer GUI*.

In root, data is stored in trees and branches. The scheme is shown in figure E.2.

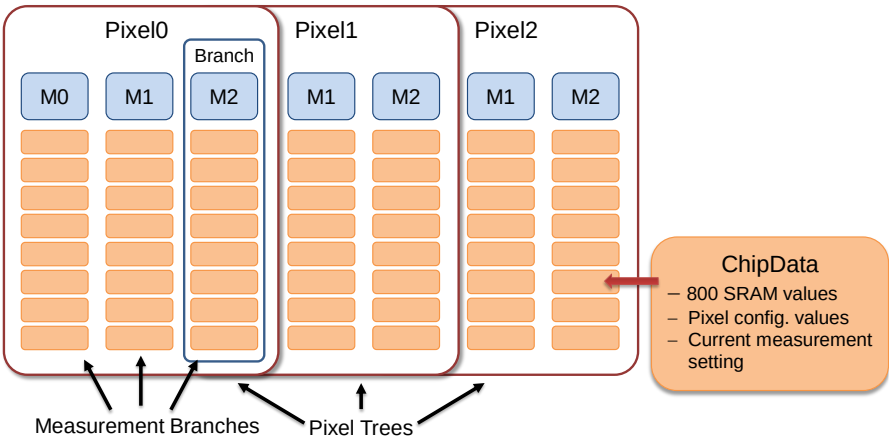


Figure E.2.: **Scheme of the root file storage structure.** All measurements of the same pixel are stored in the same root tree as separate branch. The branch elements store all data for all iterations and settings linearly in the branch.

Each tree can contain a set of branches and a branch stores a set of entries. An entry can be single data set from one measurement point. Then one branch contains all measurement

points of one measurement. In order to optimize the data structure for pixel-wise data analysis all measurements from one pixel are stored in the same tree. If several measurement have been done for the same pixel, the pixel-tree contains several branches. On the other hand, when the same measurement was performed for several pixels, several trees are stored and each tree keeps one branch with a set of entries representing the measurement points. In order to keep the hierarchy simple the measurement points for all measurement iterations for the same measurement setting are stored flatly in the branch as separate entries.

A typical data readout from the readout ASIC contains the data from all readout pixels and all SRAM cells. This means for each measured pixel and each measurement point (iteration) a whole set of measurement values has to be stored. Additionally, parts the system configuration should be saved besides the pixel data. For this purpose, the root library offers a possibility to store complex data structures as a branch entry. Even includes complex C++ class objects implementing class members and class methods can be injected. The data class can be added by generating the tree branches by the **bronch()** function instead of the **branch()** function.

Adding the data as C++ class structure has the advantage that some intrinsic features like the data decoding or some computations can be shipped directly by the data class and doesn't have to be implemented by the unpacking code.

E.1.2 The ChipData Structure

The actual implementation of the data class *ChipData* is visualized in listing 5.

The basic elements are the vector members. The data vector keeps the 16 bit data words. The settings vector keeps a set of settings which can be for example a DAC setting and also the measured DAC voltage. The setting is mostly used for the x-axis value and also from the settings vector the number of iterations per setting can be determined. The two vectors *params* and *paramNames* are dynamic containers to store the configuration data. Storing the configuration data for each burst makes it possible to reconstruct the whole system status during any measurement cycle. Of course, this produces some overhead, but compared to the 800 data words (in case of the *DSSC_F1* chip) the configuration parameters make up only a small part of the data structure.

This structure allows data analysis up to every single SRAM value of every measured pixels.

E.1.3 The ChipDataHisto Structure

In order to accelerate certain measurement types, which are based on histograms containing a large number of iterations measured with the same configuration parameters, like it is required in spectrum or noise measurements, the data class was extended to store histograms instead of the data vector. The declaration of the *ChipDataHisto* class is shown in 5. The derivation from the *ChipData* class is not optimal, because besides the histogram, also the data vectors from the *ChipData* class are stored though they are containing invalid data. But this small overhead can be neglected because only one histogram needs to be stored per pixel and measurement. In case of the *DSSC_F1* chip, this overhead makes up only 1600 Bytes per pixel and measurement.

```
class ChipData
 {
   private:
   int getParamIndex(const std::string& paramName);

   public:
     ChipData( const ChipData& chipData);
     ChipData(std::vector<int> _params,
              std::vector<std::string> _paramNames,
              int _numData, int _channel = 0);
     ChipData(int numParams, int numData);
     ChipData();
     virtual ~ChipData();

     void copy(ChipData const& _chipData);
     void setData(UShort_t *data);
     void setChannel(int chan);

     void setParamValue(std::string paramName, int value);
     int getParamValue(std::string paramName);

     void changeSetting(double set1, double set2 = 0.0);
     bool operator==(ChipData const& _chipData);
     void operator=(ChipData const& _chipData);

     UShort_t max();
     UShort_t min();

     int numData;

     std::vector<int> params;
     std::vector<std::string> paramNames;
     std::vector<double> settings;
     std::vector<UShort_t> data;

     int channel;

     static const int c_numSettings;

     ClassDef( ChipData, 3 )
 };
```

Listing 5: **Data structure ChipData: stores burst and configuration data.**

```
lass ChipDataHisto : public ChipData

 public:
   ChipDataHisto();
   ChipDataHisto(int numParams, int numData);
   virtual ~ChipDataHisto();

   TH1I histo;

   ClassDef( ChipDataHisto, 1 )
;
```

Listing 6: **Data structure ChipDataHisto: stores the data in histograms.**

E.2 Analyzer Structure

The *DataAnalyzer* application allows to loads the data from a root file, generation of plots and also performs further analysis of the data. For visualization, the *DataAnalyzer* implements Qt based GUI. Since the root library also includes a Qt based plot Widget the plotting functionality can easily be integrated in the Qt application.

In the following, all mentioned function names are the names of the functions as they can be found in the *DataAnalyzer GUI*.

After loading a root file, the *DataAnalyzer* offers several selection possibilities[1] which can be used to get a fast overview about the quality of the measurement or to exclude certain data from the later analysis:

- Pixel selection: all pixel, a subset or a single pixel.
- Measurement selection: one of the acquired measurements can be selected.
- SRAM cell selection: a range from 0 - 799 number of SRAM cells can be defined.
- Setting range selection: the start and end setting, which has been acquired can be defined.

There are two separate analysis modes which depend on the data structure in the root file. As mentioned above the two different data structures are

The structure type is detected during loading of the file. For each type different functions are available. In the histogram mode, only a subset of functions is available. Mosty the features for spectrum analysis are applicable, which include fitting and visualization features.

On can divide the *DataAnalyzer* in two basic fields:

- Burst-wise analysis
- Setting-wise analysis

[1] All mentioned selection possibilities are valid for every function in the *DataAnalyzer*

E.3 Burst-wise Analysis

Burst wise data analysis offers the possibility to check the quality of a measurement. Therefore, a set of functions is provided:

- *Show Burst Histo*: is the basic function to plot a histogram of the selected burst. Every single burst of all measured pixels and measurements can be selected.
- *Plot Burst Content*: plots the data of a single burst over the pulse number. This can be used to show some fading behavior at the beginning or the end of the burst. Depending on the measurement, often the first SRAM values differ from the rest, so it can be checked which values should better be excluded from the later analysis.
- *RO Start/End Address*: the user can manually decide if certain pulse numbers (SRAM addresses) should be ignored during later analysis and while generating plots.

The next complexity level is formed by the investigation of several bursts of the same setting. If the same configuration was measured a number of times, for each setting a number of bursts are available. The data of all iterations for every measurement can be analyzed by the following functions:

- *Show Setting Histo*: plots all values of all bursts taken with the selected setting into one histogram. The histogram shows mean and rms values.
- *Plot Setting Content*: plots all values of all bursts taken with the selected setting like it is done by *Plot Burst Content* over the iteration number and the SRAM address.
- *Bursts Mean*: plots the mean value for each burst of the selected setting over the iteration number. This function is very useful to identify system instabilities over time from burst to burst. If no parameter is changed and the mean value moves from burst to burst, this could indicate e.g. some temperature sensibilities in the setup. Additionally, a fast floating average is plotted which has a kernel of 5.
- *Bursts Mean Histo*: plots the same data in a histogram. The mean and rms values are also listed.
- The *Mean SRAM*: function generates a mean value for every SRAM address for all bursts of the selected setting. If time correlated systematic effects vary the output of the readout ASIC during every burst, this function can be used to visualize this. Additionally, a fast floating average is plotted which has a kernel of 5.
- *Mean SRAM Histo*: can be used to show the data from *Mean SRAM* in a histogram. he mean and rms values are also listed.
- *Stddev of all bursts*: plots the rms of every burst of the selected setting over the iteration number.
- *Bursts stddev histo* can be used to show the data from *Stddev of all bursts* in a histogram. he mean and rms values are also listed.

E.4 Analyzer Modes

The *DataAnalyzer* provides a large set of special modes, which can be used to pre-convert the measured data or to change the output of the analysis function. In some rare cases, the conversion from the raw pixel data into valid decimal values does not work correctly. This can be manually corrected during the data analysis, using the provided readout modes:

- *Both Caps*: this is the standard readout mode , which shows all measured values without any conversion
- *SH Cap0*: this mode includes only the even SRAM cells in all analysis functions
- *SH Cap1*: this mode includes only the uneven SRAM cells in all analysis functions
- *MM3 Invert B0 at B1 Low*: this mode inverts bit 0 if bit 1 is zero. This is required in measurements measuring in-pixel-counters if the conversion went wrong
- *MM3 Invert B0 at B1 High*: this mode inverts bit 0 if bit 1 is one. This is required in measurements measuring in-pixel-counters if the conversion went wrong
- *MM3 Invert B0*: this mode inverts bit 0 in any case. This is required in measurements measuring in-pixel-counters if the conversion went wrong[2].
- *MM3 Convert from GCC*: converts raw, but reordered, gray code counter (GCC) data into decimal values.
- The *Address Skip Mode*: can be used in very special cases if not every SRAM cell, but every N-th cell contains valid data. When activated the user can specify the number of cells to skip. This can be any value between 0 and 800.

E.4.1 Additional Quick Features

In order to specialize the data analysis, or to add some special features to the generated plots, the Data Analyzer provides a set of checkboxes, which can arbitrarily combined. Not all features are feasible or produce useful plots such that some features can not be activated for all outputs.

- *EBars:* if checked, all plots are generated with error bars. In the most graphs, the error bars are calculated from the standard deviation of the setting histograms[3].
- *8Bit:* if checked, the 9th bit is cropped. All graphs wrap at 255 instead of 511.
- *2Bin:* if checked, the zero bit is cropped. This can be used e.g. in DNL and INL calculations to remove the variations of the fast-clock-duty-cycle[4].

[2] If the rising edge of the RMP signal was delayed until the next falling edge of the fast clock arrives.
[3] A setting histogram contains all data from all iterations of the same setting.
[4] the ADC count is generated by the rising and the falling edge of the fast clock. Merging of the two least significant bits removes all duty cycle effects in the ADC characteristics.

- *Rnd:* if checked, the mean values of the setting histograms are rounded. This gives a good impression of the widths of the single ADC bins.
- *Fits:* if checked, almost every output image gets a fit. Normally, a linear function is fitted to the data. In case of a histogram a Gaussian function is fitted.
- *OffsetCorr:* if checked, some measurement (in case it makes sense) become offset corrected, which means, that from every value the setting zero value is subtracted.
- *Only Errors:* if checked, only the rms values are plotted.
- *Use 2nd Setting:* It is possible to add a second setting to the measuring points. This can be used to change the unit of the x axis e.g. from setting number to voltage. If checked the x-axis shows the values of the second setting.
- *GCCWrap:* this can correct an effect of the GCC. The 9-th bit has a very slow transition from zero to one. Consequently, the ADC characteristics drops after 255 to 0 and rises after 5 again to 260. This generates real bad errors in any fit function. The GCCWrap value can be used to plane this gap. This means all values, which are smaller than the GCCWrap value are increased by 256.
- *Correction:* three different data correction modes are implemented: None, Linear, Full. Very complex operations are implemented behind this functionality, which will be described in section E.6. To perform the correction, comprehensive data analysis is required, which generates large sets of correction data. The data is sorted in the measurement directory and is automatically loaded if it is needed after a restart of the application.

The size and resolution of the output can also be changed. The default resolution is 1000 times 600 pixels, but it can be changed by the user to any size. If the image view area is currently visible which shows a static image loaded from a file, the plot has to be regenerated in order to change the resolution. If the root plot area is the current visible plot area the size of the plot can be changed immediately by clicking the *resize* button. The root plot area offers also a lot of useful functionality. The most important function of the root plot area is the re-scale function of the x- and y-axis. Using this, it is possible to zoom in and out of the plot. By clicking on the *Save CurrentView As* button, the current view of the root plot area can be stored in a file. Different types are possible:
pnd,pdf,C,root

E.5 Setting-Wise Analysis

Setting-wise analysis summarizes all plot functions, which plot mean values of all iterations of a setting over the setting value. This can be done over a set of selected pixels, or over all measurements of one single pixel.

- The function *Plot Measurements of Pixel* generates for every selected pixel one plot, which contains the graphs of all measurements of this pixel. Normally, in every measurement a second parameter is changed besides the parameter, which is changed

in the settings parameter. This function can be used to compare the influence of the second measurement parameter.

- The function *Plot Measurement over Pixels* can be used, to visualize the data of the same measurement over different pixels. This provides an overview over the measurement characteristics in different pixels.

- *Plot Single SH Caps* is a more complex function. This function analyzes some ASIC intrinsic parameter. For every selected pixel and only the selected measurement two graphs are plotted. For every burst the overall mean value is calculated and also the mean value of all second SRAM entries. The first graph shows the difference between the mean value and the mean value of the even SRAM cells, the second graph shows the deviation of the uneven SRAM cells. This function can be used to show the difference of the *Sample and Hold* [5] capacitors.

Of course there is also a method to plot the setting against its number. This makes sense for all measurement which keep a physical size in the setting like time or voltage. The *Plot Setting* function plots the value of the setting over its number. Also a fit and the residual to the fit is generated.

E.5.1 Fitting of Measurements

The *DataAnalyzer* provides the possibility to fit a linear function to the measured data. Also here the pixels and the measurement can be selected. There are two options for fitting, one fast option, which applies a self written linear regression method, and a slower method, which uses root fits and generates for each fit also a second plot with the residual.

Both methods produce several additional output files:

- *Slope Histogram:* A histogram, in which all slopes of all fitted pixel measurements are included.

- *Slope Map:* A 2D colormap, in which all slopes of all fitted pixel measurements are plotted.

- *Slope vs Setting:* if more than one measurement is included in the root file, a plot is generated which plots all measurement slopes for one pixel over the measurement setting. This plot is generated for all selected pixels.

- *Summary Text File:* the file contains a large set of parameters: the mean value of the RMS of all measurement points : MeanRMS the offset and the slope of the linear fit including the errors and the mean RMS of the residual.

The *Det FBCaps* function uses fits of measurement graphs, like in the *Slope vs Setting* output, to determine the capacitor ratios of the feedback capacitors in the front-end circuit of the readout pixel. This function produces only for very special measurements valid data. This could be an integration time sweep or a pixel injection sweep for several different

[5] The Sample and Hold capacitors (S&H Caps) are used in the pixel ADC to sample the next measurement during the current ADC conversion. Typically they have a difference of about 0.1%

feedback capacity settings. To determine the ratios, the slopes of all measurements are compared. Besides the slope also parasitic capacities are taken into account as an offset in order to improve the results. The output is a graph of the computed capacity ratio over the capacity setting. In order to give a guideline for the capacitor ratios, the design values are included in the routine.

DNL-INL Computation

In order to investigate the quality of the in-pixel ADC, the uniformity of the widths of the single ADC bins can be determined. Therefore, the DNL and INL is calculated.

The definition of the DNL (differential non-linearity) is:

$$DNL_i = \frac{d_i}{m} - 1 \tag{E.1}$$

with d_i, the width of the i-th bin and m the mean/ideal bin width. The width can be any unit, depending on the measurement. Consequently, the DNL can have a value in [-1,$\inf$).

The definition of the INL (integral non-linearity) is

$$INL_i = \frac{Y_i}{F(x_i)} - 1 \tag{E.2}$$

with Y_i the measured ADC value at the measurement point x_i and $F(x_i)$ the computed ADC value from an ideal function which was determined by fitting the measurement.

In order to summarize the DNL and INL computations, a further parameter is introduced: the DNL and INL **RMS** value. It is defined as follows:

$$DNLRMS = \sqrt{\frac{\sum_{i=0}^{N} DNL_i^2}{N}} \tag{E.3}$$

$$INLRMS = \sqrt{\frac{\sum_{i=0}^{N} INL_i^2}{N}} \tag{E.4}$$

For both the following is valid: the closer the RMS value is to zero the more uniform is the ADC.

The *DataAnalyzer* offers two different ways to determine the DNL and INL. It can not be said that one of the two methods is the better one. This mostly depends on the measurement and the statistics of the measurement. To determine the DNL and INL, the measurement must provide at least one measurement point on almost every ADC bin. If single bins are not hit, the algorithm can handle this, but it makes no sense to determine the width of single ADC bins, if most of the bins are not hit at all.

Plot Histo of all Measured Values: is from the algorithm point of view the simpler method. It adds all single SRAM entries of a whole measurement into one large histogram. The histogram bin count is used to calculate the DNL compared to the mean bin count. From the DNL values the INL value is determined:

$$INL_i = \frac{\sum_{j=0}^{i} D_j}{i * M} - 1 \tag{E.5}$$

with D_j the j-th histogram bin count and M the mean histo bin count.

Det INLs and DNLs provides a more complex algorithm. The algorithm moves along the measurement values from small ADC values to large ADC values (or the other direction, depending on the measurement). The bin width is determined, by detecting values of the next bin. Because of noisy transitions, a minimum number of values in the next bins has to be defined by the user, which signals, that the measurement curve has propagated to the next larger bin. This prevents the analysis method from proceeding too fast to the next bin.

In order to increase the precision of the algorithm, a 'next bin' value is not determined by using the mean value of all SRAM entries of a burst (or all bursts of the same setting). In noisy measurements, in which more than the current bin and the next bin is hit, the mean value is not the correct value. To determine the position of the next bin in a more precise way, the number of values which are in all the larger bins is compared to the number of values which are in all the smaller bins. If the ratio of these numbers is larger than 0.5 it is a value of the next bin. This sorting of the measured values in values depending to the current and the next larger bin is called *Step Boarder Data* and can be visualized by the function *Plot Step Boarder Data*.

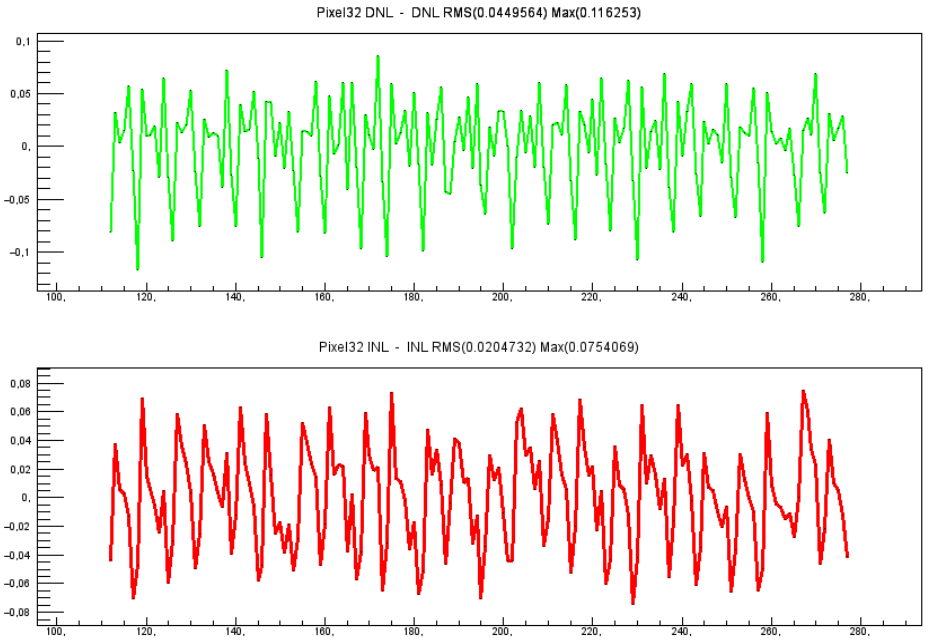


Figure E.3.: **Upper graph: DNL. Lower graph: INL.** The plot shows a very uniform ADC measurement with a DNL_RMS of below 5% and an INL_RMS of round 2%.

Both functions produce the same output:

- For every analyzed pixel, a plot, with the measurement graph and a fit. The fit is the same fit which is used to determine the INL (In the Histogram method the fit and the plot are generated after the analysis from the DNL and INL values).
- For every analyzed pixel, a text file, which stores all data from the DNL and INL algorithm: i, Setting, Value, FitValue, StepWidth, DNL, INL, movingDNL

- For every analyzed pixel, a plot is created which shows in the upper graph the determined DNL values over the ADC bin and in the lower graph the determined INL values over the ADC bin (see figure E.3).
- For every column in the summary text file, a 2-D color map is generated which visualizes the results of the DNL and INL calculations. Further information can be found in the next section E.7.
- A summary text file, which contains for every selected pixel a set of computed values: Pixel, MaxDNL, MeanDNL,DNL RMS, MaxINL, MeanINL, INL RMS, NumBinsFound

E.6 Systematic Corrections

Using high gain setting in the readout ASIC the image data of a pixel shows some kind of systematic oscillation from one image to the next. since it is a systematic effect it can be computed and removed from the data. A large version of this effect is shown in figure E.4. By enabling the correction field the correction data for all pixels and each SRAM address is computed and stored in the *SRAMCorrection.txt* file in the measurement directory.

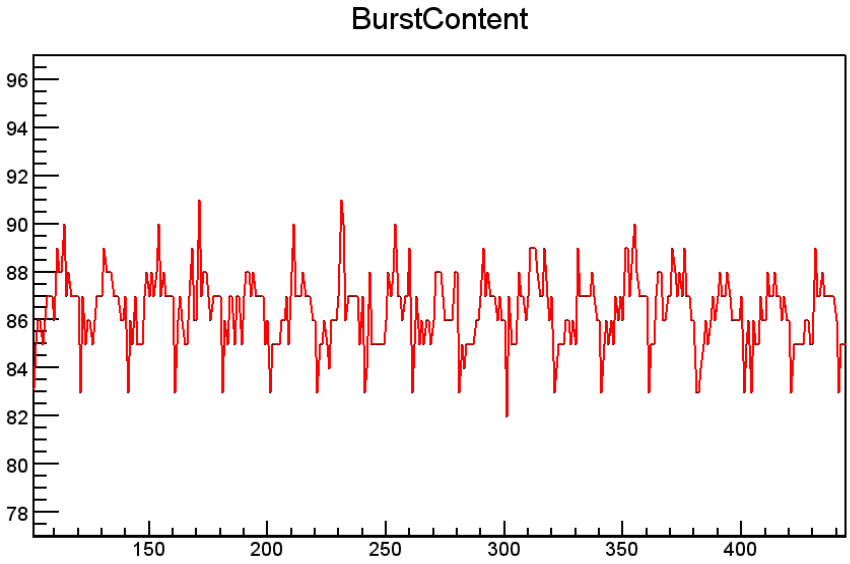


Figure E.4.: **Image data from pixel 1024.** The measured pixel was configured to the highest front-end and ADC gain in order to express the systematic effect as strong as possible. The amplitude is about 8 ADU.

The correction data is determined by the mean value for every SRAM cell of all measured iterations of the currently selected setting. Exactly like in the *Mean SRAM* output function.

An example computation for 64 pixels is shown in figure E.5. A very regular pattern can be seen, which is automatically corrected by the method.

The SRAM correction data, which is stored for every SRAM address and for all pixels in the measurement, can be applied to every singe data point. This can be used, for instance,

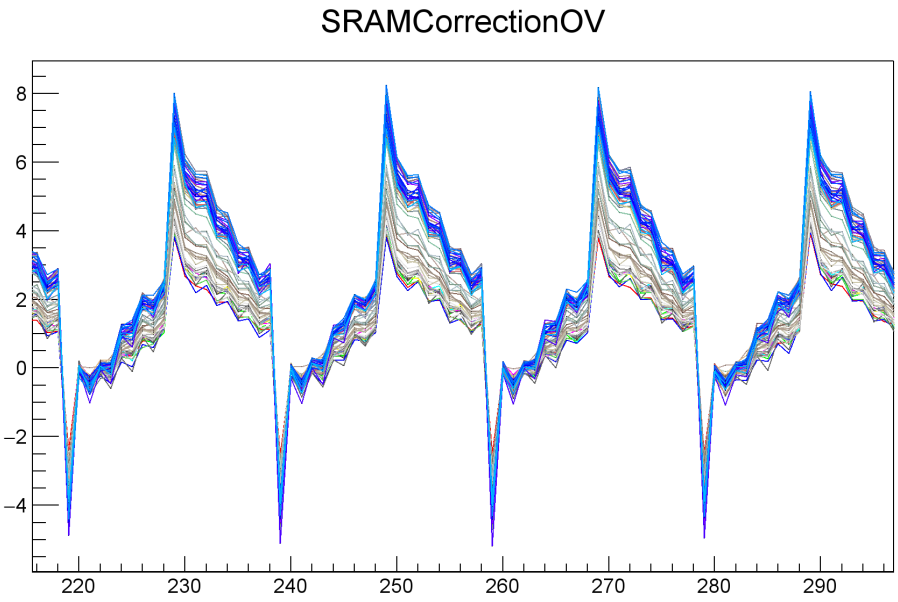


Figure E.5.: **Mean SRAM value of 300 images (iterations) of 64 pixels from one measurement.** The amplitude, frequency and shape of the systematic effect can clearly be seen in all selected pixels.

to improve the quality of a spectrum measurement as shown in figure E.6. Since ADC values are integer values, and the SRAM correction value is recorded as a double, the correction can not directly be applied. In order to improve the correction, a random shift is applied which corresponds to the decimal value of the SRAM correction. For instance, if the SRAM correction value for address number 10 is 5.5, the pixel value is shifted by -5 and an additional shift of -1 is applied with a probability of 50 %. In this way, a spectrum measurement which is spread by a strong oscillation can be fairly improved.

E.7 2D-Color-Map Generation

A function was introduced to produce 2D color maps for any kind of measurement from a text file. The function can be called by *Plot 2DMap From File*. In the following, the structure of the text file is described:

- **Comments** are lines, which start with a ' #' sign.
- The **header** can contain some control sequences:
  - *####* **Columns** indicate the number of values on the x-axis of the 2D map. This can be e.g. the number of pixels per column of a readout ASIC.
  - *####* **StartColumn** indicates that the second column is **not** the first column to generate a 2D map.
  - *####* **Number of Measurements** indicates that more than one measurement was taken for all pixels. For every measurement a separate 2D map is generated.

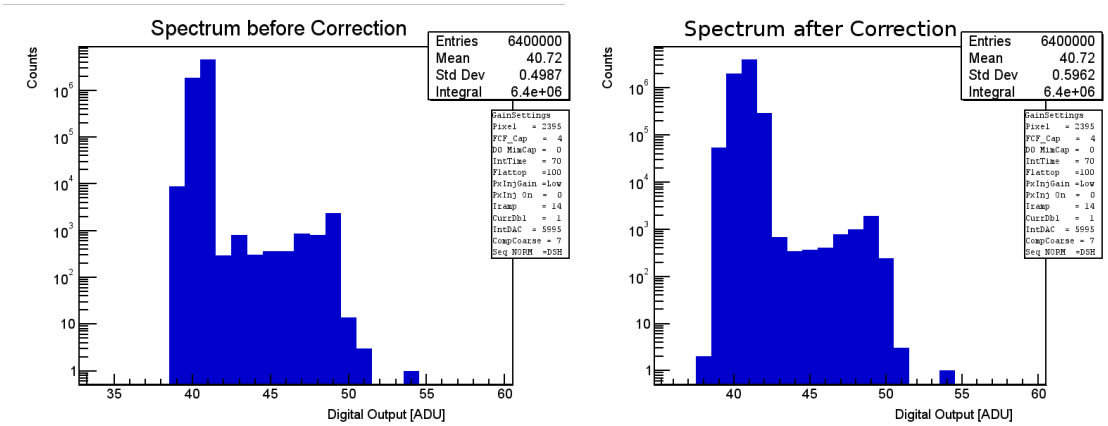


Figure E.6.: **Effect of the SRAM correction.** The systematic effect has only been max 0.9 bins. But the effect is clearly shown by the much more uniform shape of the spectrum, after correction on the right side.

The single measurements must be arranged first according to the pixel number and second according to the measurement number.

The color map of the plot is defined by 100 color steps. The scale of the color map is adapted to the range of values which shall be plotted. Figure E.7 shows ad 2D color map which was generated by a DNL and INL computation.

E.8 Image Viewer

Most of the functionality, introduced so far, are investigating the measurement data pixel wise. But there are also measurements, in which the 2D image is important. Therefore, the *Image Viewer* was introduced. It is a separate widget, which can be activated by the *Image Viewer* button.

It displays the pixel wise sorted data in 2D color maps. Every single SRAM address of all taken bursts can be regarded. By selecting several SRAM addresses of the same burst, also mean values can be displayed. The *Image Viewer* console is shown in figure E.8.

The features of the *Image Viewer* are listed in the following:

- *Single Image:* if checked, only the first selected Image (SRAM address) of the first selected Burst is plotted. This is valid for all other features. If unchecked, the mean value of the selected SRAM and Burst range is plotted.
- *Subtract Baseline:* if checked, a previously defined baseline is subtracted from the currently selected image. This is active for all available features. The baseline can be recorded by clicking the *Set Baseline* button. The actual image is then set as baseline.
- *Show Hist:* the histogram of the selected range is plotted.
- *Show RMS:* the RMS map is plotted.
- *En Hexagonal Pixels:* the color map is filled with hexagonal pixels

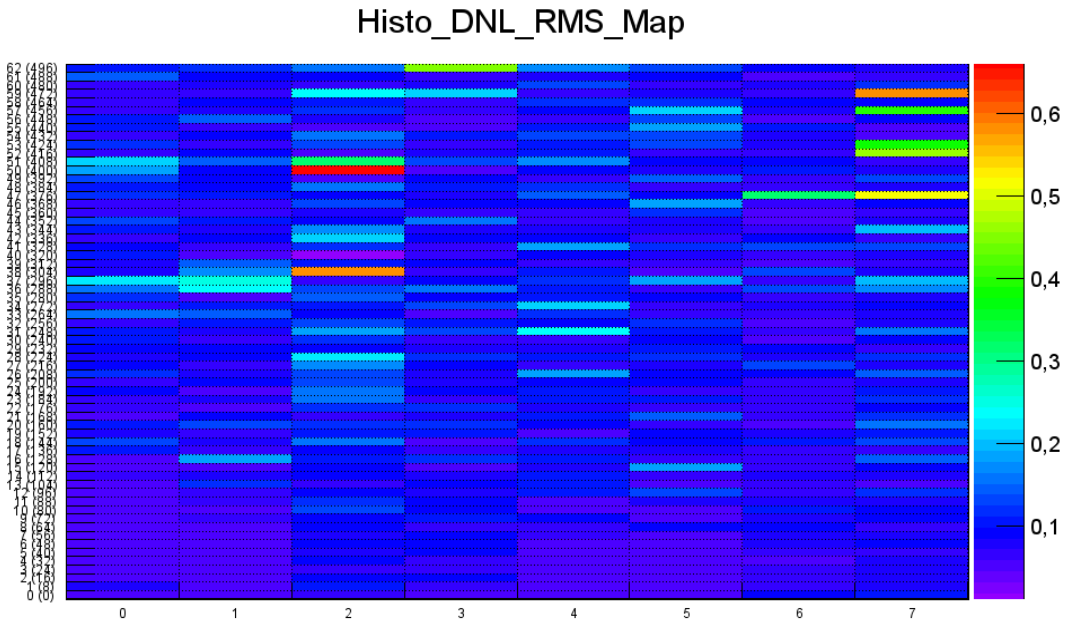


Figure E.7.: 2D color Map showing the DNL RMS of a 4096 pixel measurement. Number of columns was set to 64.

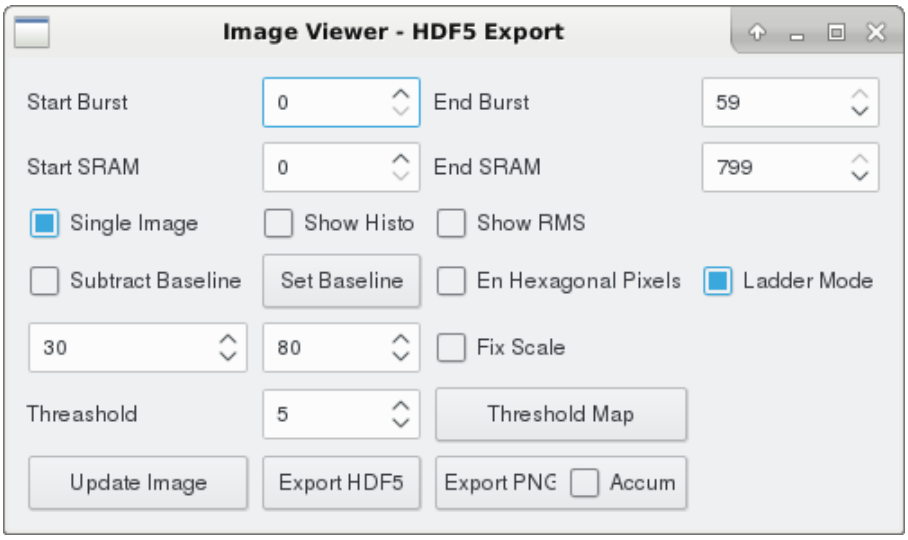


Figure E.8.: **Image Viewer Console.** The *Image Viewer* console provides a variety of outputs (see text).

- *Ladder Mode:* the output image is shown in Ladder format: 512x128 pixels. If disabled, only one ASIC 64x64 pixels is plotted. The ASIC number can be selected.
- *Fix Scale:* the color scale or the x-axis of the histogram can be fixed to a defined range.
- *Threshold:* a threshold map is generated. This counts for each pixel the number of entries which are above the threshold. To use this function a suitable baseline has to be defined.

As a special feature, the *Image Viewer* provides also the possibility to export the measured data into png images or HDF5 files, which also stores image wise ordered data. In order to keep compatibility to other software tools, for every exported train, one distinct HDF5 file is generated. The HDF5 are exported using the DSSC HDF5 library, which is used by all HDF5 tools in the consortium to provide compatible HDF5 files from different sources. The files are identical to the files which are generated by the Karabo software.

If the *Accum* checkbox is activated before export, the actual exported image is always the mean image of all previously exported image.

An example of the *Image Viewer* Output is shown in figure E.9.

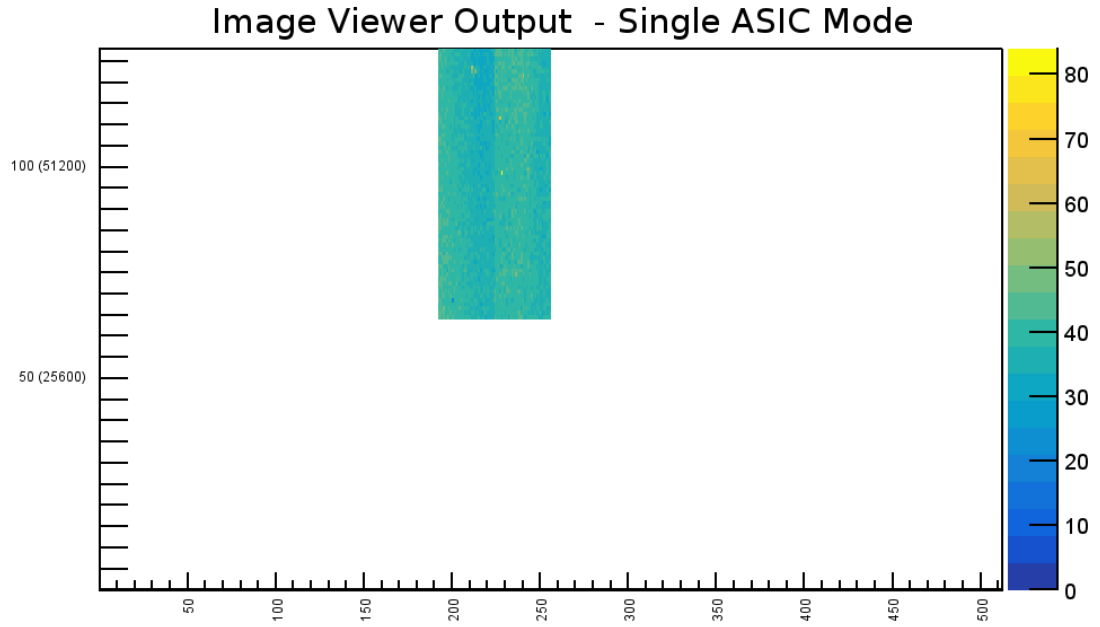


Figure E.9.: **Image Viewer Output.** Using the *Image Viewer* it is possible to generate images in Ladder format or single ASIC format. The current image shows a Ladder format image of a single ASIC measurement (ASIC 11).

E.9 Data Export

Sometimes it is required to export the data from a measurement into a different format, e.g. to enable other users to analyze data with their own requirements and their own software

or scripts. Therefore there are various methods available. The button *Data Export* opens the export selection.

- *Export Data to ASCII*: Exports the measurement data in histogram format to a text '.dat' file. The ordering is: Pixel -> Measurement -> Setting -> Iteration -> Histogram Data. The format is: Device-ID PixelNr Setting BinCenter BinContent
- *Export Data to ASCII with Voltage Settings*: same as above but the Setting is replaced by the measured voltage, if available.
- *Export Multiple Data to ASCII*: required several root files which are meant to be exported. The export is identical to the first method. For every file, a separate directory is generated, in which the export is stored.
- *Export all Data to CSV*: exports every single measured value, (all pixels, all SRAM addresses) into a 'csv' file.
  - The output order is: Pixel -> Measurement -> Setting
  - The output format is: Setting;Data0;Data1 .... DataN. With N the number of iterations times values per Iteration/Burst (SRAM addresses).

  For analysis, the pixel and measurement information must be taken from the numbers in the header and the line number.
- *Export Mean Data to CSV*: exports only the mean values of every burst into a 'csv' file. This decreases the size of the output file a lot.
  - The output order is: Pixel -> Measurement -> Setting
  - The output format is: PixelNr;MeasurementNr;Setting;Mean;RMS

E.10 HDF5 File Import

With the first measurement which have been taken with the Karabo framework and have been stored in the HDF5 file format, a HDF5 to root converter became strongly needed.

For easiest usage, the converter has been seamlessly integrated into the *DataAnalyzer* application. A Karabo measurement generates a special HDF5 file, which contains all required information to find the recorded configuration and image data. Even complex measurements of up to two parameters are supported by the measurement and import environment.

The importer recognizes the file format and the number of ASICs which are included in the image data. Using this information the operator is asked which pixels should be exported into a new root file. Depending on the memory of the actual computer even 16 ASICs can be included in the root file.

Afterward, the conversion automatically starts, which usually takes several minutes up to hours, depending on the number of acquired images. After the conversion has finished, the root file is automatically loaded for direct analysis.

Besides the full image data, two alternative import modes are implemented: a quick import mode which reads the overview HDF5 files, which are generated per measurement setting. This can be used for a quick view if the settings have correctly been stepped through.

But since the method imports only histograms, the information which value has been stored in which SRAM cell is lost.

also a direct export mode is available, which reads the histogram files, and directly exports the histograms in text files in the same format, like it is done by the *Export Data to ASCII* export function.

E.11 Multi-Measurement Plots

In order to provide additional flexibility, it is also possible to combine arbitrary graphs in the same plot. This even can be graphs from different root files. The only limitation is, that the y-axis of the graphs must have about the same range, which is automatically given if the y axis shows e.g. ADC BIN values. The button *MultiFilePlots* opens a window in which this function can be controlled. As long as the button remains checked, all generated graphs are added to the left field of the multi-plot window. This window keeps the graphs also when a new root-file is loaded. From the left field, the user can select the graphs which are to be plotted in the common output.

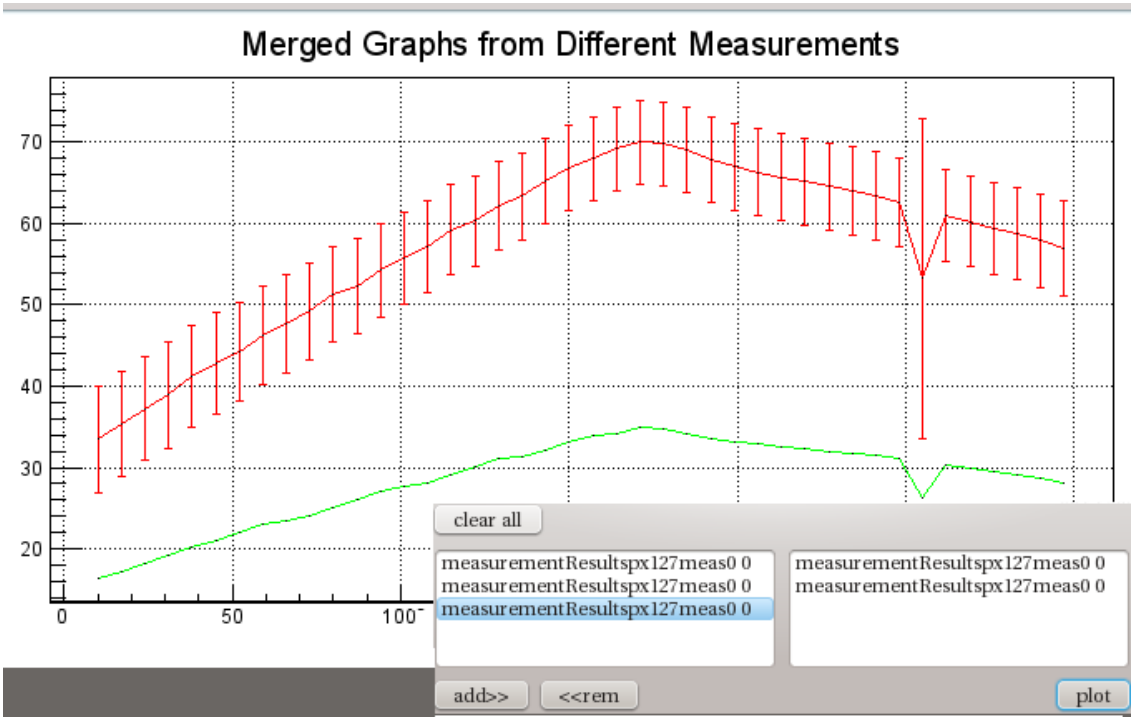


Figure E.10.: **The multiplot window and the output is shown.** The *'add'* and *'rem'* buttons can be used to change the graphs which are to be plotted in the output.

E.12 Plot TempADC Values

In the *DSSC_F1* readout ASIC, a temperature diode and an ADC is implemented. The temperature values are read out and appended to the image data in the ASIC trailer. Since

the PPT reads the TempADC values and includes the values in the Detector Specific block at the end of the Train Data, the temperature values are available for every train which is sent out of the detector.

Both the root and the HDF5 files contain the data, and consequently it can easily be plotted by the DataAnalyzer.

An example output of a long term Karabo Measurement is shown in figure E.11. Beside the TempADC values, the floating mean value is plotted.

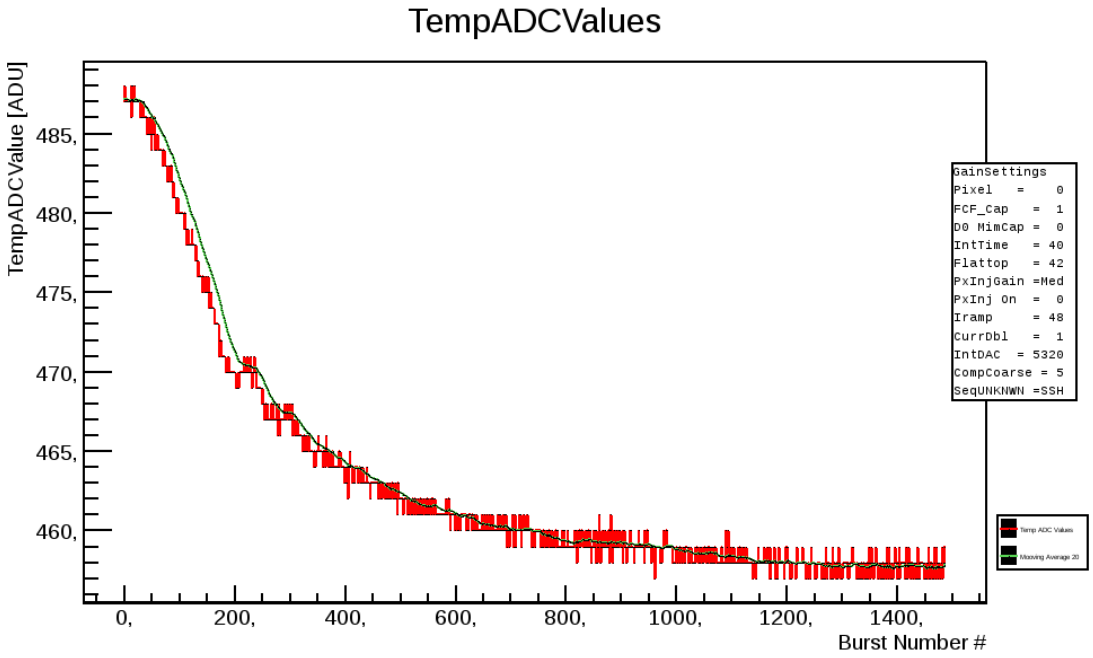


Figure E.11.: **Output of the TempADC during a temperature measurement.** During the measurement, the temperature has been changed from $+10°$C to $-20°$C.

Bibliography

[1] J. D. Watson and F. H. C. Crick. Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, Apr 1953.

[2] F. R. Elder, A. M. Gurewitsch, R. V. Langmuir, and H. C. Pollock. Radiation from electrons in a synchrotron. *Phys. Rev.*, 71:829–830, Jun 1947.

[3] Synchrotron Soleil EPSIM 3D/JF Santarelli. Schema de principe du synchrotron. https://commons.wikimedia.org/wiki/File:SchOct 2005.

[4] A. Motz. Undulators and 'free-electron lasers. *Contemporary Physics*, 20:547–568, 1979.

[5] Janet L Smith, Robert F Fischetti, and Masaki Yamamoto. Micro-crystallography comes of age. *Current opinion in structural biology*, 22(5):602–612, 2012.

[6] R. Neutze, R. Wouts, D. van der Spoel, E. Weckert, and J. Hajdu. Potential for biomolecular imaging with femtosecond X-ray pulses. *Nature*, 406:752–757, August 2000.

[7] European XFEL. European XFEL in Comparision. https://www.xfel.eu/facility/comparison/index_eng.html, 2016.

[8] European XFEL. XFEL Facility Overview. https://www.xfel.eu/facility/overview/-index_eng.html.

[9] Brian W. J. McNeil and Neil R. Thompson. X-ray free-electron lasers. *Nat Photon*, 4(12):814–821, Dec 2010.

[10] EA Schneidmiller, MV Yurkov, et al. *Photon beam properties at the European XFEL*. DESY, 2011.

[11] J Rossbach. Demonstration of gain saturation and controlled variation of pulse length at the {TESLA} test facility {FEL}. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 507(12):362 – 367, 2003. Proceedings of the 24th International Free Electron Laser Conference and the 9th Users Workshop.

[12] Roberto Colella and Alfredo Luccio. Proposal for a free electron laser in the x-ray region. *Optics communications*, 50(1):41–44, 1984.

[13] Kwang-Je Kim, Yuri Shvyd'ko, and Sven Reiche. A proposal for an x-ray free-electron laser oscillator with an energy-recovery linac. *Physical review letters*, 100(24):244802, 2008.

[14] John M. J. Madey. Stimulated emission of bremsstrahlung in a periodic magnetic field. *Journal of Applied Physics*, 42(5):1906–1913, 1971.

[15] E.L. Saldin, E.A. Schneidmiller, and M.V. Yurkov. The physics of free electron lasers. an introduction. *Physics Reports*, 260(4-5):187 – 327, 1995.

[16] MR Howells and Brian M Kincaid. The properties of undulator radiation. In *New Directions in Research with Third-Generation Soft X-ray Synchrotron Radiation Sources*, pages 315–358. Springer, 1994.

[17] Peter Schmüser, Martin Dohlus, and Jörg Rossbach. *Ultraviolet and soft X-ray free-electron lasers: introduction to physical principles, experimental results, technological challenges*, volume 229. Springer Science & Business Media, 2008.

[18] R Bonifacio, C Pellegrini, and LM Narducci. Collective instabilities and high-gain regime in a free electron laser. *Optics Communications*, 50(6):373–378, 1984.

[19] Claudio Pellegrini and Joachim Stöhr. X-ray free-electron lasers: principles, properties and applications. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 500(1):33–40, 2003.

[20] JW Lewellen, SV Milton, E Gluskin, ND Arnold, C Benson, W Berg, Sandra G Biedron, M Borland, Y-C Chae, RJ Dejus, et al. Present status and recent results from the aps sase fel. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 483(1):40–45, 2002.

[21] J. Andruszkow et al. First observation of self-amplified spontaneous emission in a free-electron laser at 109 nm wavelength. *Phys. Rev. Lett.*, 85:3825–3829, Oct 2000.

[22] Wet al Ackermann, G Asova, V Ayvazyan, A Azima, N Baboi, J Bähr, V Balandin, B Beutner, A Brandt, A Bolzmann, et al. Operation of a free-electron laser from the extreme ultraviolet to the water window. *Nature photonics*, 1(6):336–342, 2007.

[23] *FIRST LASING IN THE WATER WINDOW WITH 4.1 NM AT FLASH*, FEL Experiments and Projects, 2011.

[24] P. Emma, R. Akre, J. Arthur, R. Bionta, C. Bostedt, J. Bozek, A. Brachmann, P. Bucksbaum, R. Coffee, F.-J. Decker, Y. Ding, D. Dowell, S. Edstrom, A. Fisher, J. Frisch, S. Gilevich, J. Hastings, G. Hays, P. Hering, Z. Huang, R. Iverson, H. Loos, M. Messerschmidt, A. Miahnahri, S. Moeller, H.-D. Nuhn, G. Pile, D. Ratner, J. Rzepiela, D. Schultz, T. Smith, P. Stefan, H. Tompkins, J. Turner, J. Welch, W. White, J. Wu, G. Yocky, and J. Galayda. First lasing and operation of an ångstrom-wavelength free-electron laser. *Nature Photonics*, 4:641–647, September 2010.

[25] Henry N Chapman and Keith A Nugent. Coherent lensless x-ray imaging. *Nature Photonics*, 4(12):833–839, 2010.

[26] S. Marchesini, H. He, H. N. Chapman, S. P. Hau-Riege, A. Noy, M. R. Howells, U. Weierstall, and J. C. H. Spence. X-ray image reconstruction from a diffraction pattern alone. *Phys. Rev. B*, 68:140101, Oct 2003.

[27] David Shapiro, Pierre Thibault, Tobias Beetz, Veit Elser, Malcolm Howells, Chris Jacobsen, Janos Kirz, Enju Lima, Huijie Miao, Aaron M Neiman, et al. Biological imaging by soft x-ray diffraction microscopy. *Proceedings of the National Academy of Sciences*, 102(43):15343–15346, 2005.

[28] Henry N Chapman, Anton Barty, Michael J Bogan, Sébastien Boutet, Matthias Frank, Stefan P Hau-Riege, Stefano Marchesini, Bruce W Woods, Sasa Bajt, W Henry Benner, et al. Femtosecond diffractive imaging with a soft-x-ray free-electron laser. *arXiv preprint physics/0610044*, 2006.

[29] M. M. Seibert et al. Single mimivirus particles intercepted and imaged with an x-ray laser. *Nature*, 470(7332):78–81, Feb 2011. 21293374[pmid].

[30] Henry N Chapman, Carl Caleman, and Nicusor Timneanu. Diffraction before destruction. *Phil. Trans. R. Soc. B*, 369(1647):20130313, 2014.

[31] Massimo Altarelli, Reinhard Brinkmann, Majed Chergui, Barry Decking, Winfried Dobson, Stefan Dsterer, Gerhard Grbel, Walter Graeff, Heinz Graafsma, Janos Hajdu, Jonathan Marangos, Joachim Pflger, Harald Redlin, David Riley, Ian Robinson, Jrg Rossbach, Andreas Schwarz, Kai Tiedtke, Thomas Tschentscher, Ivan Vartaniants, Hubertus Wabnitz, Hans Weise, Riko Wichmann, Karl Witte, Andreas Wolf, Michael Wulff, and Mikhail Yurkov. The European X-Ray Free-Electron Laser Technical design report. http://www.xfel.eu/documents/technical_documents/, 2007.

[32] J Rossbach. Observation of self-amplified spontaneous emission in the wavelength range from 80 to 180 nm at the {TESLA} test facility {FEL} at {DESY}. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 475(13):13 – 19, 2001. FEL2000: Proc. 22nd Int. Free Electron Laser Conference and 7th F {EL} Users Workshop.

[33] V. Ayvazyan, N. Baboi, J. Bähr, V. Balandin, B. Beutner, A. Brandt, I. Bohnet, A. Bolzmann, R. Brinkmann, and Brovko et al. First operation of a free-electron laser generating gw power radiation at 32 nm wavelength. *The European Physical Journal D - Atomic, Molecular, Optical and Plasma Physics*, 37(2):297–303, Sep 2006.

[34] European Comission. ESFRI ROADMAP 2016. http://ec.europa.eu/research/infrastructures, 2016.

[35] Frank Stephan, Mikhail Krasilnikov, Eberhard Jaeschke, Shaukat Khan, R. Jochen Schneider, and B. Jerome Hastings. *High Brightness Photo Injectors for Brilliant Light Sources*, pages 1–38. Springer International Publishing, Cham, 2014.

[36] M. Krasilnikov et al. Pitz experience on the experimental optimization of the rf photo injector for the european xfel. *Proceedings of the 35th International Free Electron Laser Conference*, 35:160–169, 2013.

[37] W. Decking and T. Limberg. European XFEL Post-TDR Description. http://www.xfel.eu/documents/technical_documents/, Feb. 2013.

[38] Instruments XFEL. Flash free-electron laser flash. https://www.xfel.eu/facility/-instruments/index_eng.html, 2017.

[39] A Allahgholi, J Becker, L Bianco, A Delfs, R Dinapoli, P Goettlicher, Heinz Graafsma, D Greiffenberg, H Hirsemann, S Jack, et al. Agipd, a high dynamic range fast detector for the european xfel. *Journal of Instrumentation*, 10(01):C01023, 2015.

[40] John Coughlan, Sam Cook, Chris Day, Rob Halsall, and Saeed Taghavi. The data acquisition card for the large pixel detector at the european-xfel. *Journal of Instrumentation*, 6(12):C12057, 2011.

[41] Andreas Koch, Matthew Hart, Tim Nicholls, Christian Angelsen, John Coughlan, Marcus French, Steffen Hauf, Markus Kuster, Jolanta Sztuk-Dambietz, Monica Turcato, et al. Performance of an lpd prototype detector at mhz frame rates under synchrotron and fel radiation. *Journal of Instrumentation*, 8(11):C11001, 2013.

[42] D. H. Wilkinson. A stable ninety-nine channel pulse amplitude analyser for slow counting. *Mathematical Proceedings of the Cambridge Philosophical Society*, 46:508–518, 7 1950.

[43] E Motuk, M Postranecky, M Warren, and M Wing. Design and development of electronics for the euxfel clock and control system. *Journal of Instrumentation*, 7(01):C01062, 2012.

[44] P. Gessler et al. Specification of the VETO system at European XFEL. XFEL, Technical report, 2012.

[45] J Coughlan, C Day, J Edwards, E Freeman, S Galagedera, and R Halsall. The trainbuilder atca data acquisition board for the european-xfel. *Journal of Instrumentation*, 7(12):C12006, 2012.

[46] Over Plastic Optical Fiber. Ieee standard for ethernet. 2017.

[47] Jon Postel et al. Rfc 791: Internet protocol. 1981.

[48] Jon Postel and User Datagram Protocol. Rfc 768. *User datagram protocol*, pages 1–3, 1980.

[49] National Laboratory in Argonne and National Laboratory in Los Angeles. Epics - experimental physics and industrial control system. http://www.aps.anl.gov/epics/, 2004.

[50] JM Chaize, J Meyer, M Pérez, A Götz, E Taurel, and WD Klotz. Tango: An object oriented control system based on corba. *ICALEPCS1999, Trieste*, 1999.

[51] A Götz, E Taurel, JL Pons, P Verdier, JM Chaize, J Meyer, F Poncet, G Heunen, E Götz, A Buteau, et al. Tango a corba based control system. *ICALEPCS2003, Gyeongju, October*, 2003.

[52] G Grygiel, O Hensler, and K Rehlich. Doocs: a distributed object oriented control system on pcs and workstations. *PCaPAC96, DESY, Hamburg*, 1996.

[53] O. Barana, P. Barbato, M. Breda, R. Capobianco, A. Luchetta, F. Molon, M. Moressa, P. Simionato, C. Taliercio, and E. Zampiva. Comparison between commercial and open-source {SCADA} packages - a case study. *Fusion Engineering and Design*, 85(3 - 4):491 – 495, 2010. Proceedings of the 7th {IAEA} Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.

[54] P Duval, Z Kakucs, D Golob, M Kadunc, I Kriznar, M Plesko, A Pucelj, and G Tkacik. The babylonization of control systems part ii-the rise of fallen tower. In *ICALEPS*, volume 3, page 513, 2003.

[55] Elke Sombrowski, A Petrosyan, K Rehlich, and P Tege. Jddd: a java doocs data display for the xfel. *ICALEPCS07, Knoxville, TN*, 2007.

[56] Raimund Kammering, Winfried Decking, Lars Fröhlich, Olaf Hensler, Torsten Limberg, Sascha Meykopff, Kay Rehlich, Vladimir Rybnikov, Josef Wilgen, and Tim Wilksen. The virtual european xfel accelerator. In *15th International Conference on Accelerator and Large Experimental Physics Control Systems*, number PUBDB-2015-05125. Beschleunigerphysik, 2015.

[57] Tango control. Tango website. http://www.tango-controls.org/.

[58] Burkhard Heisen, Djelloul Boukhelef, Sergey Esenov, Steffen Hauf, Iryna Kozlova, Luis Maia, Andrea Parenti, Janusz Szuba, Kerstin Weger, Krzysztof Wrona, and Christopher Youngman. Karabo: An Integrated Software Framework Combining Control, Data Management, and Scientific Computing Tasks. page FRCOAAB02. 14th International Conference on Accelerator and Large Experimental Physics Control Systems, San Francisco (USA), 6 Oct 2013 - 11 Oct 2013, Oct 2013.

[59] Astrid Muennich, Steffen Hauf, Burkhard Heisen, Friederike Januschek, Markus Kuster, Philipp-Michael Lang, Natascha Raab, Tonn Rueter, Jolanta Sztuk-Dambietz, and Monica Turcato. Integrated Detector Control and Calibration Processing at the European XFEL. 2015.

[60] Burkhard Heisen. Karabo: The european xfel software framework - design concepts. https://docs.xfel.eu/alfresco/d/a/workspace/SpacesStore/9b331f2f-fe2e-4ece-850d-96b486207f10/2013_0 3_Karabo_Design_Concepts.pptx, 2014.

[61] Astrid Münnich, Steffen Hauf, BC Heisen, Friederike Januschek, Markus Kuster, Philipp-Michael Lang, Natascha Raab, Tonn Rüter, J Sztuk-Dambietz, and Monica Turcato. Integrated detector control and calibration processing at the european xfel. *arXiv preprint arXiv:1601.01794*, 2016.

[62] Oracle. Quick introduction to open message queue. https://mq.java.net/overview.html, 2013.

[63] Mark Wutka. *Special edition using Java 2*. Que, Indianapolis, IN, enterprise ed edition, 2001. Mode of access: World Wide Web. - Made available through: Safari Books Online, LLC.

[64] Oracle. Glassfish - worlds first java ee 7 application server. https://glassfish.java.net/, 2017.

[65] Jasmin Blanchette and Mark Summerfield. *C++-GUI-Programmierung mit Qt 4: die offizielle Einführung*. Pearson Deutschland GmbH, 2009.

[66] The Qt Company Ltd. Qt Documentation. http://doc.qt.io/, 1994-2017.

[67] Burkhard Heisen, Martin Teichmann, Kerstin Weger, and John Wiggins. Karabo-GUI: The Multi-Purpose Graphical Front-End for the Karabo Framework. page WEPGF153. 15th International Conference on Accelerator and Large Experimental Physics Control Systems, Melbourne (Australia), 17 Oct 2015 - 23 Oct 2015, Oct 2015.

[68] M Porro, L Andricek, L Bombelli, G De Vita, C Fiorini, P Fischer, K Hansen, P Lechner, G Lutz, L Strüder, et al. Expected performance of the depfet sensor with signal compression: A large format x-ray imager with mega-frame readout capability for the european xfel. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 624(2):509–519, 2010.

[69] M. Porro et al. Development of the depfet sensor with signal compression: A large format x-ray imager with mega-frame readout capability for the european xfel. *Transactions on Nuclear Science*, 59:3339–3351, December 2012.

[70] P. Fischer, M. Bach, L. Bombelli, G. De Vita, F. Erdinger, S. Facchinetti, C. Fiorini, K. Hansen, S. Herrmann, P. Kalavakuru, M. Manghisoni, M. Porro, and C. Reckleben. Pixel readout asic with per pixel digitization and digital storage for the dssc detector at xfel. In *IEEE Nuclear Science Symposuim Medical Imaging Conference*, pages 336–341, Oct 2010.

[71] F. Erdinger. Dssc mm3 matrix chip. DSSC document, 2013.

[72] Glenn F Knoll. *Radiation detection and measurement*. John Wiley & Sons, 2010.

[73] Florian Erdinger. *Design of Front End Electronics and a Full Scale 4k Pixel Readout ASIC for the DSSC X-ray Detector at the European XFEL*. PhD thesis, 2016.

[74] Helmuth Spieler. *Semiconductor detector systems*, volume 12. Oxford university press, 2005.

[75] L Bombelli, C Fiorini, S Facchinetti, M Porro, and G De Vita. A fast current readout strategy for the xfel depfet detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 624(2):360–366, 2010.

[76] Emilio Gatti and Pavel Rehak. Semiconductor drift chamber an application of a novel charge transport scheme. *Nuclear Instruments and Methods in Physics Research*, 225(3):608–614, 1984.

[77] J. Kemmer and G. Lutz. New semiconductor concept. *Nucl. Instr. and Meth. A*, 253:365–377, 1987.

[78] P Lechner, L Andricek, S Aschauer, A Bähr, G De Vita, K Hermenau, T Hildebrand, G Lutz, P Majewski, M Porro, et al. Depfet active pixel sensor with non-linear amplification. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2011 IEEE*, pages 563–568. IEEE, 2011.

[79] L. Bombelli, C. Fiorini, S. Facchinetti, M. Porro, and G. De Vita. A fast current readout strategy for the {XFEL} depfet detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 624(2):360 – 366, 2010. "New Developments in Radiation Detectors"Proceedings of the 11th European Symposium on Semiconductor Detectors11th European Symposium on Semiconductor Detectors".

[80] C Fiorini, B Nasri, S Facchinetti, L Bombelli, P Fischer, and M Porro. A simple technique for signal compression in high dynamic range, high speed x-ray pixel detectors. *IEEE Transactions on Nuclear Science*, 61(5):2595–2600, 2014.

[81] K. Hansen, C. Reckleben, I. Diehl, M. Bach, and P. Kalavakuru. Pixel-level 8-bit 5-ms/s wilkinson-type digitizer for the {DSSC} x-ray imager: Concept study. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 629(1):269 – 276, 2011.

[82] Christian Reckleben, Karsten Hansen, Pradeep Kalavakuru, Janusz Szymanski, Florian Erdinger, Peter Fischer, Manfred Kirchgessner, and Jan Soldat. A 64-by-64 pixel-ADC matrix. In *2015 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC) : [Proceedings] - IEEE, 2015. - ISBN 978-1-4673-9862-6 - doi:10.1109/NSSMIC.2015.7581817*, page 7581817. 2015 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), San Diego, CA (USA), 31 Oct 2015 - 7 Nov 2015, IEEE, 2015.

[83] Thomas Gerlach. Development of the daq front-end for the dssc detector at the european xfel, 2013.

[84] Xilinx. 7 series fpgas configuration - ug470. https://www.xilinx.com/support/documentation/user_guides/ug471_7Series-_SelectIO.pdf, 2016.

[85] F Erdinger, P Fischer, and M Porro. A novel signal compression circuit for charge collecting pixel detectors. In *Proc. IEEE Nuclear Science Symp.*, pages 8–15, 2014.

[86] Xilinx UG683. Edk concepts, tools, and techniques, Apr 2011.

[87] Xilinx UG081. Microblaze processor reference guide. https://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf, June 2008.

[88] Xilinx UG761. Axi reference guide. https://www.xilinx.com/support/documentation-/ip_documentation/axi_ref_guide/v13_4/ug761_axi_ reference_guide.pdf, Jan 2012.

[89] Xilinx PG127. Axi external peripheral controller (epc) v2.0. https://www.xilinx.com/support/documentation/ip_documentation/axi_epc/-v2_0/pg127axi-epc.pdf, Oct 2016.

[90] Ieee standard for reduced-pin and enhanced-functionality test access port and boundary-scan architecture. *IEEE Std 1149.7-2009*, pages 1–985, Feb 2010.

[91] Xilinx UG029. Chipscope pro software and cores. https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/-chipscope_pro_sw_cores_ug 029.pdf, Oct 2012.

[92] Xilinx XAPP1251. Xilinx virtual cable running on zynq-7000 using the petalinux tools. https://www.xilinx.com/support/documentation/application_notes/xapp1251-xvc-zynq-petalinux.pdf, Apr 2015.

[93] M Kirchgessner, J Soldat, A Kugel, M Donato, M Porro, and P Fischer. The high performance readout chain for the dssc 1megapixel detector, designed for high throughput during pulsed operation mode. *Journal of Instrumentation*, 10(01):C01011, 2015.

[94] Xilinx DS797. Logicore ip aurora 8b/10b v8.1. https://www.xilinx.com/support/documentation/ip_documentation/-aurora_8b10b/v8_1/aurora_8b10b _ds797.pdf, Apr 2012.

[95] Xilinx PG051. Tri-mode ethernet mac v8.3. https://www.xilinx.com/support/documentation/ip_documentation/-tri_mode_ethernet_mac/v8_3/pg 051-tri-mode-eth-mac.pdf, Oct 2014.

[96] Rene Brun and Fons Rademakers. Root an object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1):81 – 86, 1997.

[97] The HDF Group. Hierarchical Data Format, version 5. http://www.hdfgroup.org/HDF5/, 1997-2017.

[98] G Weidenspointner, R Andritschke, S Aschauer, P Fischer, S Granato, K Hansen, P Lechner, G Lutz, D Moch, M Porro, et al. Strategy for calibrating the non-linear gain of the dssc detector for the european xfel. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2011 IEEE*, pages 468–473. IEEE, 2011.

[99] David Moch, Georg Weidenspointner, Stephan Schlee, Alexander Bähr, Matteo Porro, Stefan Aschauer, Florian Erdinger, Peter Fischer, Manfred Kirchgessner, Jan Soldat, et al. Calibration of the non-linear system characteristic of a prototype of the dssc detector for the european xfel. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2014 IEEE*, pages 1–4. IEEE, 2014.

[100] Stephan Schlee, G Weidenspointner, M Porro, Markus Kuster, and D Moch. Methods for calibrating the gain and offset of the dssc detector for the european xfel. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2015 IEEE*, pages 1–2. IEEE, 2015.

[101] Emanuele Quartieri and Massimo Manghisoni. Performance of a high accuracy injection circuit for in-pixel calibration of a large sensor matrix. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2011 IEEE*, pages 677–681. IEEE, 2011.

[102] Massimo Manghisoni, Daniele Comotti, Emanuele Quartieri, Peter Fischer, and Matteo Porro. Pixel-level charge and current injection circuit for high accuracy calibration of the dssc chip at the european xfel. *IEEE Transactions on Nuclear Science*, 60(5):3852–3861, 2013.

[103] Jean loup Gailly. The gzip home page. http://www.gzip.org/, 2015.

[104] Wayne Davison. rsync. https://rsync.samba.org/, 2015.

[105] Mattia Donato, Karsten Hansen, Pradeep Kalavakuru, Manfred Kirchgessner, Markus Kuster, Matteo Porro, Christian Reckleben, and Monica Turcato. First functionality tests of a 64×64 pixel dssc sensor module connected to the complete ladder readout. *Journal of Instrumentation*, 12(03):C03025, 2017.

[106] G. Weidenspointner et al. Dssc prototype ladder operation and performance study at petra iii / p04. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2017 IEEE*, page in preparation. IEEE, 2017.

Acknowledgement

First of all, I would like to thank Christina, my future wife, for her steady support and incredible patience during the past years. I would also like to thank my parents for always being there for me and supporting me in all my decisions.

I thank my supervisor, Prof. Dr. Peter Fischer, for giving me the opportunity to be part of this exciting project and giving me the chance to be creative and let ideas develop. I would like to offer my special thanks to my colleagues Dr. Florian Erdinger and Jan Soldat for their constant support and advices and the excellent cooperation.

My thanks are also extended to the rest of the DSSC consortium. It was a great pleasure both from the scientific and human perspective to work in this project. Frequent assistance provided by Monica Turcato, working directly with the FENICE vacuum chamber, was greatly appreciated since without her, no remote work would have been possible.

I want to thank all other colleagues at the Chair for the friendly atmosphere and for many answers on many questions.

I also thank all proof readers and of course the referees for reporting on my thesis. Many others I would like to thank here, but I can not all explicitly mention. I sincerely hope that I have never missed to say thank you in the respective situations and that nobody feels forgotten.