Università degli Studi di Ferrara

UNIVERSITY OF FERRARA

DEPARTMENT OF PHYSICS AND EARTH SCIENCE

DOCTORAL COURSE IN **PHYSICS**
CYCLE XXXVIII

Design, development and validation of DataPix4: a C++ framework for Timepix4-based detectors

Ph.D. Candidate:
Viola Cavallini

Supervisor:
Prof. Massimiliano Fiorini

Co-Supervisor:
Prof. Sebastiano Fabio Schifano

Coordinator: Prof. Paolo Lenisa

Settore Scientifico Disciplinare (SSD): PHYS-01/A

Academic Years 2022-2025

Abstract

The thesis addresses the growing demand in experimental physics for software solutions capable of managing increasing data rates, high precision requirements and complex detector architectures. The central element is the development and validation of DataPix4, a framework designed for operating detector systems based on the Timepix4, the latest ASIC produced by the Medipix4 Collaboration. It is a state-of-the-art hybrid pixel chip with a dense pixel matrix, multi-mode acquisition and sub-ns timing resolution, suitable for both high-energy physics and a wide range of other applications.

The development of DataPix4 was motivated by the requirements of the 4DPHOTON project, aiming to create an innovative photodetector by coupling a MicroChannel Plate (MCP) amplification stage to the Timepix4 ASIC, serving as a pixelated anode within a vacuum tube. This detector concept targets simultaneous high spatial and temporal resolution, critical for applications like the LHCb Ring-Imaging Cherenkov (RICH) detectors, particularly for the planned RICH Upgrade II scenario, which necessitates handling instantaneous luminosities more than five times higher than the current one.

The DataPix4 framework is implemented in C++, chosen for its optimization capabilities and ability to perform low-level data manipulation. The framework supports both Timepix4 communication channels: a Slow Control link (up to 1 Gbps),

for configuration and slow data read-out, and 16 Fast Links, which can achieve a maximum bandwidth of ~ 160 Gbit/s. The data acquisition supports both Timepix4 modes: data-driven for zero-suppressed, event-by-event read-out, and frame-based for synchronous, full matrix read-out, suitable for extremely high rates. Management is simplified through high-level methods. DataPix4 incorporates automated calibration and setup routines, also accessible via a dedicated Python-based Graphical User Interface (GUI). It also provides online data monitoring and online data analysis via a dedicated clustering algorithm, that saves ROOT files with comprehensive information on clusters, for complete offline analysis.

The framework's validation spans diverse experimental conditions. Timepix4 characterization measurements, exploiting the DataPix4 framework for control and acquisition, demonstrated a cluster timing resolution as low as (33 ± 3) ps. Energy calibration, performed using monochromatic X-rays at INFN Ferrara and the Elettra Synchrotron in Trieste, enabled pixel-by-pixel energy calibration and achieved an energy resolution < 1 keV. DataPix4 was employed during two test-beam campaigns at CERN, to characterize the first 4DPHOTON detector prototypes in a RICH configuration.

Beyond fundamental physics experiments, DataPix4 proved its versatility across applied physics. It is employed in a project aiming to develop a system for Energy-Resolved Radiography (ERR) and Multi-Energy Computed Tomography (MECT) of artworks. Furthermore, it was integrated into the Python-based PEPIControl system at INFN Trieste to facilitate high-resolution micro-CT scans. It was also adapted for use as a beam monitoring system at Fermilab's IOTA ring, where the online clustering output was redirected to a custom GUI for real-time beam diagnostics.

A final component of the work involved developing a custom Convolutional Neural Network (CNN) to classify particle tracks (Alpha, Electron, Muon, Photon) detected by Timepix4. After training on an annotated dataset, the model achieved an accuracy of 0.84 on the validation set. To prepare the network for efficient hardware acceleration, quantization was performed using the AMD Vitis AI environment. This quantized network is planned for integration into DataPix4 using a separate thread, enabling on-the-fly particle identification alongside data acquisition. The combination of the core DataPix4 framework with advanced machine-learning tools establishes a complete environment for configuring, operating and analysing data from Timepix4-based detectors.

Contents

Introduction

Nowadays, the Standard Model of Particle Physics represents our best understanding of how the fundamental constituents of the Universe interact to form the world around us. Although this model can explain very well what we see around us, it still fails to unfold some fundamental questions, related, for instance, to the origin of the matter-antimatter asymmetry in the Universe or the nature of dark matter, the gravitationally interacting but non-luminous component of the cosmos.

To investigate more in deep these phenomena, and to achieve a more complete description of the Universe, a new and more sophisticated generation of particle-physics experiments is foreseen for the next decades. These future experiments must push the boundaries of technology in many areas, from innovative materials to scalability, from component miniaturization to large-scale data handling. In this context, the development of advanced instrumentation for particle detection, with higher data rates, increased precision requirements and more complex architectures, calls for equally capable software solutions.

Among the new generation of detection technologies, hybrid pixel ASICs represent a major step forward, combining fine spatial granularity with precise timing and energy information. Timepix4, the latest chip developed by the Medipix4 Collaboration at CERN, is one of the state-of-the-art ASICs, that can be applied across various areas of high energy physics, ranging from charged-particle tracking to photodetectors. Its

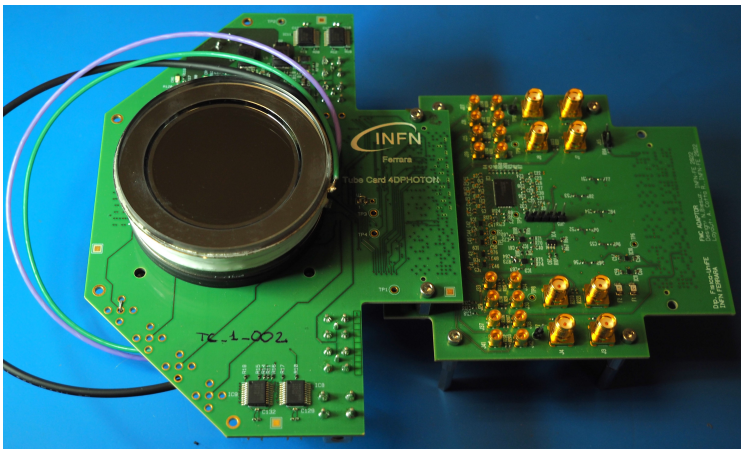


Figure 1: 4DPHOTON detector accommodated in the Tube Card developed at INFN Ferrara.

dense pixel matrix, multi-mode acquisition capabilities and sub-nanosecond timing resolution make it suitable for both high-energy physics environments and a broad range of applications.

Timepix4 plays a central role in the 4DPHOTON project, that aims at developing an innovative photon detector capable of delivering high spatial and temporal resolution simultaneously.

Within the project, Timepix4 serves as a pixelated anode, coupled to a MicroChannel Plate (MCP) amplification stage inside a vacuum tube, enabling precise reconstruction of photon arrival position and time. The first prototype of the detector is show in fig. 1. Achieving this goal requires not only advanced detector hardware, but also an equally capable software infrastructure for configuration, control and data acquisition.

For this reason, the development of a dedicated framework became a fundamental component of the project. DataPix4, the software at the core of this thesis, was conceived to provide a flexible, robust and efficient platform for operating Timepix4 across a variety of experimental conditions. Its purpose is to simplify the configuration of the ASIC, handle high-throughput data streams and provide real-time monitoring and basic analysis tools. While DataPix4 was initially motivated by the needs of the 4DPHOTON project, its modular design makes it suitable for any Timepix4-based setup, from beam tests to X-ray imaging experiments.

The validation of the framework required testing it in different experimental con-

ditions. These ranged from laboratory measurements, to characterize the spatial, temporal and energy response of Timepix4, to its use in test-beams and imaging setups. These studies made it possible to verify the correct functioning of DataPix4, to identify limitations and to refine some aspects of the software. They also provided examples of how the framework can be adapted to different hardware configurations and acquisition environments.

In parallel, part of the work focused on developing automated tools for data processing within DataPix4. One of these is a convolutional neural network designed to recognize particle tracks directly from Timepix4 data. The network was trained, evaluated and later quantized to operate efficiently on dedicated hardware platforms. Its integration into DataPix4 shows how machine-learning methods can be incorporated into the acquisition and analysis chain when needed.

The efforts described here reflect the broader challenge of preparing the instrumentation and software infrastructure required for the next generation of particle-physics experiments. By designing, developing and validating a dedicated framework for Timepix4, and by exploring innovative data-processing approaches, this thesis contributes to that long-term objective. The tools and results presented in the following chapters offer a foundation on which future experiments and detector systems can build.

The material presented in the thesis follows the progression of the work carried out during the development and validation of the DataPix4 framework.

Chapter 1 provides the experimental context by outlining the LHCb detector and the technologies that motivate the need for high-performance pixel systems. This detector is described both in its actual configuration and in its upgraded version, which is foreseen for the future run 5, where a completely novel design of LHCb sub-detectors is needed to cope with the foreseen increased luminosity.

Chapter 2 introduces the 4DPHOTON project, that aims to develop a novel single-photon detector, combining high spatial and temporal resolution, low noise, high-rate capability and a large active area in a single device. The Timepix4 ASIC is a fundamental part of the photodetector, thus a complete description is provided, emphasising its main characteristics, in particular the ones that shaped the software requirements.

Chapter 3 is dedicated to the DataPix4 framework, describing its design and implementation in detail. The chapter covers the overall software architecture, the meth-

ods used to configure the Timepix4 ASIC, and the different data-acquisition modes supported by the framework. It also presents the calibration procedures necessary to ensure accurate operation of the detector. In addition to the core functionality, the chapter describes the tools developed for practical use, including real-time monitoring of acquisition, online analysis of incoming data and the graphical user interface that allows routine tests and adjustments to be carried out efficiently.

The experimental validation of the DataPix4 framework is gathered in chapter 4, which presents measurements performed in laboratory conditions, with different setups and during test-beams. These studies illustrate how DataPix4 behaves under different operating modes and hardware configurations and they highlight the aspects of the software that were refined based on experimental feedback.

A final component of the work is the development of automated data-processing tools. Chapter 5 describes the convolutional neural network designed for particle-track identification, its training and quantization and its future integration into the acquisition chain. The combination of the core framework and the machine-learning tools establishes a complete environment for operating Timepix4 and handling its data in a consistent way.

CHAPTER 1

The LHCb Experiment

The LHCb experiment is one of the four large experiments located at the Large Hadron Collider (LHC) accelerator at CERN, in Geneva [1]. This experiment has been operating since 2010 and it is specifically designed for precision measurements of CP violation and rare decays of B and D hadrons [2]. Its primary goal is to search for evidence of new physics beyond the Standard Model of Particle Physics (SM), particularly in phenomena related to CP violation, which the SM cannot fully explain yet [3].

The LHCb detector is a multi-component system comprising a Tracking System, that exploits a Vertex Locator (VELO), an upstream tracker (UT) and a scintillating fibres tracker (sciFi) for a fast, online reconstruction of the events; a particle identification system, with two Cherenkov detectors, two calorimeters and a muon system, for a more advanced offline reconstruction; a magnet to bend particle trajectories; a multilayer trigger system. All the hardware is supported by a robust online system and an extensive computing and resources model [4]. The design emphasizes high resolution for vertex and momentum, efficient particle identification (especially for hadrons) and a high-bandwidth data acquisition system to achieve its physics goals.

1.1. LHCb overview

The LHCb experiment is a complex system designed and developed at CERN by the LHCb Collaboration, a group of scientists that, nowadays, counts over 704 people representing 69 different universities and laboratories from 17 countries [5]. LHCb aims to investigate the SM more in deep and, in particular, the asymmetry between matter and anti-matter in our Universe. To study the Charge conjugation and Parity (CP) violation, it is designed to collect data about the particles colliding inside LHC, in particular about the ones that contain the b-quark (or *beauty* hadron) and the c-quark (or *charm* hadron).

The detector and its systems are intricately designed to efficiently detect and reconstruct these heavy flavour particles. Its key components are listed below and shown in fig. 1.1 [6].

- **Tracking System**: Dedicated to the tracks reconstruction, it consists of three subdetectors.

  Vertex Locator (VELO): 52 modules of hybrid silicon pixel detectors, arranged in two retractable halves surrounding the interaction region. The detector comprises about 41 million pixels, each of size 55 μm × 55 μm, read out by the custom VeloPix ASIC and cooled by evaporative CO_2. Operating in a secondary vacuum separated from the LHC beam vacuum, the VELO provides precise measurements of track positions close to the interaction point, allowing the identification of displaced secondary vertices characteristic of b- and c-hadron decays. For machine protection, the two halves are retracted by 30 mm during injection and moved to their nominal position under stable-beams conditions [7].

  Upstream Tracker (UT): four planes of silicon strip detectors organised in two stations. UT is fundamental to significantly reduce the ghost tracks rate and to perform the tracking of particles decaying after the VELO.

  Scintillating Fiber Tracker (SciFi): consists of three stations (T1, T2, T3) each composed of four detection layers made of of scintillating fibres read out by silicon photomultipliers (SiPM) placed on one side of the fibres. This detector is located downstream of the dipole magnet and it is responsible for the tracking of the charged particles and the measurement of their momentum [8].

- **Magnet**: A warm dipole magnet used to bend charged particle trajectories to measure their momentum. It is located between the upstream and downstream tracking systems to optimize momentum measurement. It provides an integrated magnetic field of 4 Tm over the track path [9].

- **Particle Identification System**: Essential for distinguishing between different hadron types (pions, kaons and protons) across a wide momentum range and to identify and measure the energies and positions of electrons, photons and hadrons. It is composed by two Ring Imaging Cherenkov (RICH) detectors and two Calorimeters detectors, followed by a muon detection system. In particular:

  RICH1 and RICH2: Located respectively upstream and downstream of the magnet, they covers respectively low (1 to 60 GeV/c) and high (15 GeV/c to 100 GeV/c and beyond) momentum charged particles. Both RICH detectors use multi anode photomultiplier tubes (MaPMT), to detect Cherenkov photons. The RICH detector system will be described more in detail in section 1.2 [10].

  Calorimeter System: An Electromagnetic Calorimeter (ECAL) and an Hadronic one (HCAL), following the second RICH detector, provide information about the energies and positions of electrons, photons and hadrons, in addition to selecting transverse energy candidates for the Level-0 trigger. The ECAL, in particular, is a shashlik-type calorimeter for high-energy photon and electron detection, while the HCAL is made or iron and scintillating tiles to measure hadron energies. Located before the two calorimeters, a Scintillator Pad Detector (SPD) identifies charged particles to reject backgrounds, while a Preshower (PS) placed after a lead converter, aids in electron/pion separation [11].

  Muon System: Provides fast information and muon identification for both the High-Level Trigger (HLT) and offline analysis. It is composed of four stations (M2[1]-M5), which are separated by 80 cm iron filters. It primarily uses Multi-Wire Proportional Chambers (MWPCs), but the innermost region employs triple-GEM chambers for higher rate capability and radiation hardness [12].

- **Trigger System**: Designed to efficiently select interesting events for physics analysis from the vast number of collisions occurring at the LHC, it is composed by

[1] M1 has been removed during the Long Shutdown 2. It had a fundamental role in giving information to the L0 trigger level during Run 1 and 2, but it became useless after trigger changes.

two different levels, from the fastest, but less precise, HLT1, to the slowest, but capable of reconstructing the full event, HLT2. In Runs 1 and 2, the trigger system included a hardware stage (L0). With Upgrade I, this hardware stage could no longer sustain the required rates and was therefore replaced by a fully software HLT architecture.

High Level Trigger 1 (HLT1): implemented on GPUs, it performs a first, but already fairly complete, event reconstruction. It exploits information from the tracking system, the muon identification system and the calorimeters to reduce the colliding bunch rate from 30 MHz[2] to ~ 1.3 MHz.

High Level Trigger 2 (HLT2): implemented on CPUs, it performs a complete offline-quality reconstruction and selects the events to keep. It exploits information from all the sub-detectors of LHCb, including the RICH1 and RICH2 for hadron identification. Moreover, it is also used for real-time alignment and calibration constants of the sub-detectors [13]. HLT2 is called "Deferred Trigger", since LHCb provides a buffer of about 40 PB, enough to store up to one week of data while LHC is delivering collisions. HLT2 usually runs with a few days of delay. This trigger level produces a total bandwidth of about 10 GB/s.

- **Online System**: Ensures the transfer of data from front-end electronics to permanent storage, manages detector configuration, monitoring and synchronization with the LHC clock. This includes the Data Acquisition (DAQ) system, Timing and Fast Control (TFC) system, and Experiment Control System (ECS) [14].

The LHCb experiment has undergone and is planning further major upgrades to enhance its capabilities. These upgrades are crucial for allowing the experiment to operate at increasingly higher luminosities. The schedule, with the runs, shutdowns and upgrades is shown in fig. 1.2.

The first major upgrade, known as Upgrade I, was implemented during LHC Long Shutdown 2 (LS2, 2019-2021) and has enabled the LHCb experiment to operate at an instantaneous luminosity five times higher than its previous running periods, reaching $L = 2 \cdot 10^{33}\ \mathrm{cm}^{-2}\mathrm{s}^{-1}$. The primary motivation was to overcome the statistical limitations

[2] The LHC provides bunch crossings at 40 MHz, however, not all bunches collide (typically about 2100 colliding bunches out of 3564 possible). The read-out of the detector is still continuous at 40 MHz, but the system routes to HLT1 only the crossings with potential collisions, so about 30 MHz.

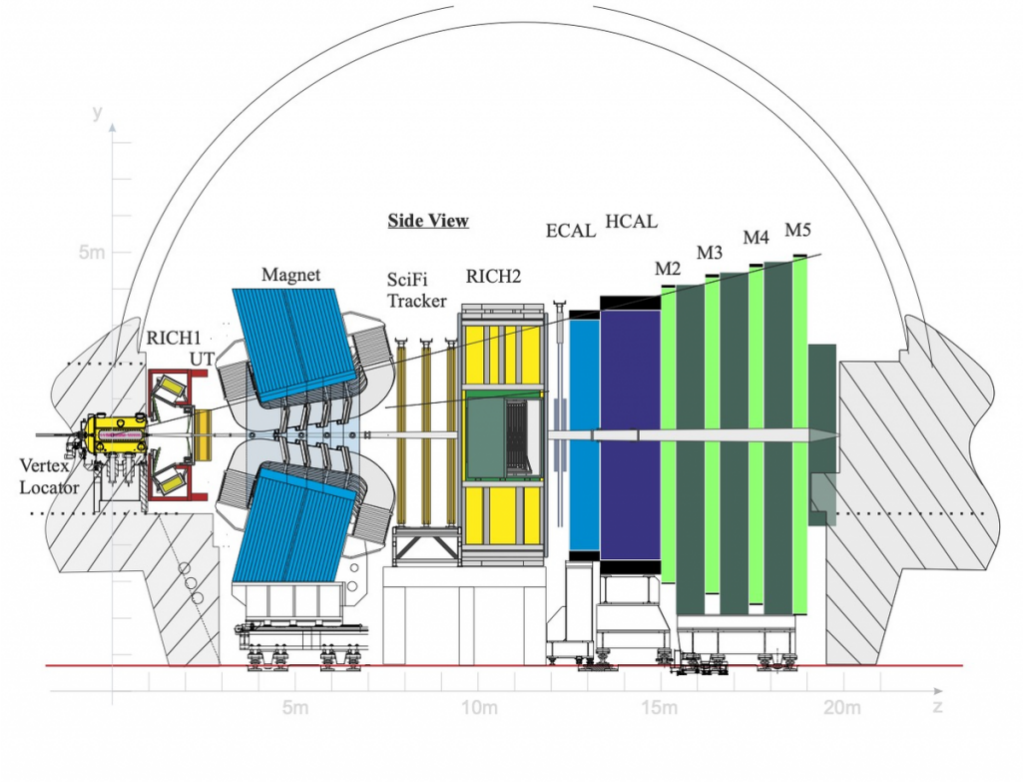


Figure 1.1: Current layout of LHCb detector, after Upgrade I. Image courtesy of CERN.

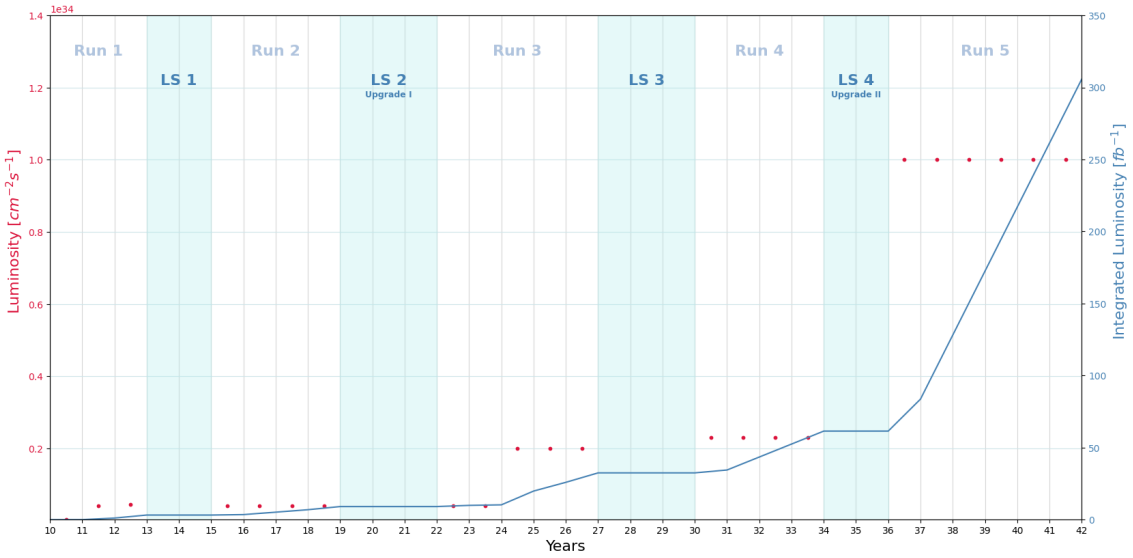


Figure 1.2: LHCb long term schedule.

of many key flavour physics observables and the saturation of event yield caused by the previous hardware-based Level-0 (L0) trigger.

The most significant change was the complete removal of the hardware L0 trigger and the adoption of an all-software trigger system. This allowed for the full detector to be read out at the LHC bunch crossing frequency of 40 MHz, and all trigger decisions to be made in software. This new strategy enabled an increase in trigger efficiency by a factor of two for many processes of interest [4].

During Upgrade I, the entire tracking system was upgraded. The previous VELO system was constituted by silicon microstrip sensor with a worse spatial and temporal resolutions with respect to the one installed nowadays. Moreover, in the former version, particle tracking was accomplished by a Tracker Turicensis (TT) and an Inner Tracker (IT) made by silicon microstrip detector located upstream of the dipole magnet and an Outer Tracker (OT) composed by three straw-tube modules stations placed downstream of the magnet. Also the photon detection system of both RICH1 and RICH2 was completely changed, as will be described more in detail in section 1.3.

In the initial RICHes, the photosensors used were Hybrid Photon Detectors (HPDs) equipped with silicon pixel array unable to guarantee operations at 40 MHz and with large charge-sharing effect. The front-end and read-out electronics of both the Electromagnetic Calorimeter (ECAL) and Hadronic Calorimeter (HCAL) were entirely redesigned and replaced to comply with the 40 MHz read-out frequency. The Scintillating Pad Detector (SPD) and Pre-Shower (PS) sub-detectors were removed due to their reduced role in the new software trigger. A complete revision of the computing model and rewriting of the experiment's software was necessary to handle the large increase in data volume and implement the real-time analysis (Turbo analysis) paradigm. The Upgrade I was designed to accumulate 50 fb^{-1} of data by the end of Run 4.

A second major upgrade, Upgrade II, is planned for installation during Long Shutdown 4 (LS4, 2033-2036). This upgrade is necessary for the full exploitation of the High Luminosity LHC (HL-LHC) for flavour physics, with a goal to accumulate an unprecedented 300 fb^{-1} of high-energy pp collision data by the end of its lifetime. The new detector is designed to operate at a maximum instantaneous luminosity of $1.5 \cdot 10^{34}$ $\text{cm}^{-2}\text{s}^{-1}$, almost an order of magnitude higher than Upgrade I, necessitating further advancements in detector technology. This higher luminosity will increase the average number of pp interactions per bunch crossing from around 6 in Upgrade I to

about 40 in Upgrade II [15].

The core strategy continues to be a software-based trigger (Real-Time Analysis, RTA), but with a strong emphasis on integrating fast-timing information from sub-detectors and employing radiation-hard technologies to cope with the increased particle rates and radiation doses. The system is being designed to handle an unprecedented throughput of 200 Tbit/s of data in real-time [15].

1.2. The RICH Detectors

The Ring-Imaging Cherenkov (RICH) detectors are a key component of the LHCb experiment at CERN, designed for charged particle identification. They play a crucial role in distinguishing between hadronic particle types such as pions, kaons and protons over a wide momentum range, from $2-100$ GeV/c [16].

The RICH system is comprised of two distinct detectors: RICH1 and RICH2, strategically placed within the LHCb spectrometer, as illustrated in Fig 1.1.

Cherenkov Effect

The Cherenkov effect arises when a charged particle propagates through a dielectric medium with a velocity v exceeding the phase velocity of light in that medium, c/n, where n is the refractive index [17]. When this threshold is met, the particle's electric field polarizes the medium locally; because the particle travels faster than the induced polarization can relax, the emitted electromagnetic disturbances interfere constructively, producing a coherent shock front of radiation. The resulting wavefront forms a cone around the particle's trajectory with a characteristic opening angle θ_c, given by the Frank-Tamm relation: $\cos\theta_c = \frac{1}{\beta n}$.

The Cherenkov effect plays a central role in experimental particle physics. Precise reconstruction of the Cherenkov angle enables the determination of a particle's velocity, which, when combined with momentum measurements, allows for particle identification.

RICH1 Detector

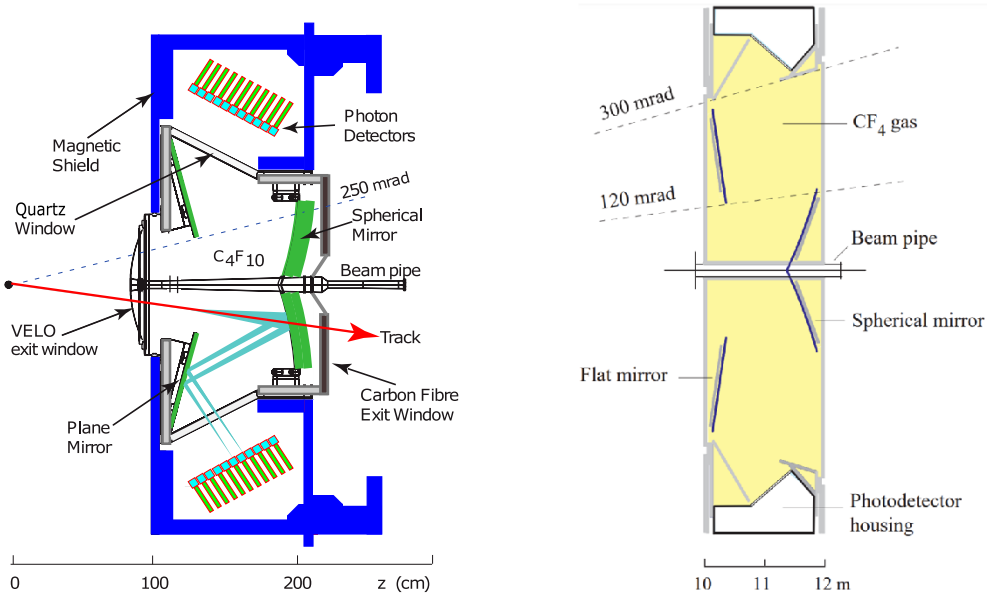
RICH1 is positioned upstream of the LHCb dipole magnet, closer to the proton-proton interaction region, immediately downstream of the Vertex Locator (VELO) exit window. Its location is optimized for identifying low and intermediate momentum particles, covering a momentum range from $2-40$ GeV/c during initial operations, and $2.6-60$ GeV/c after Upgrade I. It covers the full angular acceptance of the spectrometer, which is $25-300$ mrad horizontally and $25-250$ mrad vertically.

For its operations, RICH1 uses the C_4F_{10} gas with a refractive index of 1.0014, as a Cherenkov radiator. In its initial configuration during LHC Run 1, RICH1 also incorporated a 50 mm thick silica aerogel radiator at its entrance (fig. 1.3a). The aerogel was crucial for providing positive particle identification at low momenta and for separating kaons from protons, as particles below the kaon threshold for C_4F_{10} (9.3 GeV/c) would only be identified by the absence of Cherenkov light in the gas [18]. However, for LHC Run 2 and Upgrade I, the aerogel radiator was removed, and RICH1 operated with only C_4F_{10} gas [19].

The optical system of RICH1 features tilted spherical primary mirrors and flat secondary mirrors (fig. 1.3a). The initial design had a vertical optical layout, with two tilted spherical mirror surfaces (each composed of two CFRP mirrors, in total four quadrants) positioned above and below the beryllium beam pipe. These spherical mirrors had a radius of curvature of 2700 mm. The flat mirrors were tilted at 0.250 rad with respect to the y-axis. In the Upgrade I, the optical layout was significantly modified to halve the occupancy by increasing the focal length of the spherical mirrors. The mirrors are coated for high reflectivity across a broad wavelength range. The gas enclosure for RICH1 is sealed directly to the VELO exit window to minimize material budget and is constructed from a low-mass carbon-fibre/foam sandwich for the downstream exit window.

Before Upgrade I, photon detection was achieved using Hybrid Photon Detectors (HPDs), with 196 tubes, where photoelectrons were focused onto a silicon pixel arrays. Then, the original HPDs, limited by their 1 MHz output rate, were replaced by Multi-Anode Photomultiplier Tubes (MaPMTs) and new front-end electronics to support a 40 MHz read-out rate. The Upgrade I will be discussed more in detail in section 1.3.

The operation of RICH1 was fully automated and integrated into the Experiment Control System (ECS), continuously monitoring parameters like mirror movements,



(a) Schematic of RICH1 detector.

(b) Schematic of RICH2 detector.

Figure 1.3: Schematic of the two RICH detectors at LHCb.

gas quality, temperature and pressure, which were crucial for maintaining the refractive index and detector alignment.

RICH2 Detector

The RICH2 detector is situated downstream of the LHCb dipole magnet and it is primarily tasked with charged particle identification (PID) for particles in the higher momentum range, in particular for different hadron species, especially those involving B and C quarks [20].

In its initial configuration, RICH2 was designed to cover the high-momentum region, from approximately 15 GeV/c to 100 GeV/c. Its angular acceptance was more limited than RICH1, spanning 15 – 120 mrad horizontally and 15 – 100 mrad vertically. The detector exclusively used CF_4 gas as its Cherenkov radiator, with a refractive index of 1.0005 at 400 nm. The average path length for particles within the CF_4 radiator was approximately 167 cm.

The optical system of RICH2 was similar to RICH1, featuring tilted spherical focusing primary mirrors and secondary flat mirrors to reflect Cherenkov light images onto photon detector planes located outside the spectrometer's acceptance (fig. 1.3b).

Unlike RICH1's vertical division, RICH2 was divided horizontally. At the beginning of LHCb data taking, a total of 288 Hybrid Photon Detectors (HPDs) were employed for photon detection. These HPDs consisted of vacuum tubes with a photocathode, accelerating optics and a silicon pixel array to read out photoelectrons. The design of RICH2 also accounted for magnetic shielding, which attenuated stray magnetic fields from the LHCb dipole magnet to allow the HPDs to function effectively.

RICH2 consistently delivered excellent separation of hadronic particle types (pions, kaons and protons), which was critical for reducing backgrounds in complex decay analyses and enabling flavour tagging [20]. An automated real-time alignment and calibration system, implemented for Run 2, allowed hadron identification information from RICH2 to be used for the first time in the software-based High Level Trigger (HLT) for online event selection [16].

During Upgrade I, the photon detection planes in RICH2 were subdivided into two regions with different granularity: the central region was equipped with 1-inch MaPMT modules (pixel size 2.88×2.88 mm^2), while the peripheral regions used 2-inch MaPMTs (pixel size 6×6 mm^2). The total number of channels in the upgraded RICH system reached almost $200k$. The superstructure, magnetic shields, entrance and exit windows, and the optical system (mirrors and their supports) of RICH2 were retained without modification, as they were deemed capable of handling the expected occupancy and radiation levels. RICH Upgrade I will be discussed more in detail in the next section 1.3.

The second major upgrade of the LHCb detector, planned for Long Shutdown 4 (LS4), targets an instantaneous luminosity of up to 1.5×10^{34} cm^{-2}s^{-1} and an integrated luminosity of at least 300 fb^{-1}. For RICH2, key design criteria include keeping detector occupancy below approximately 30% and achieving a single photon Cherenkov angle resolution below 0.5 mrad. RICH Upgrade II will be discussed more in detail in section 1.4.

1.3. RICH Upgrade I

During the LHCb Upgrade I, which took place during the Long Shutdown 2 (LS2) of the LHC (roughly 2019-2021), the Ring Imaging Cherenkov (RICH) detectors underwent significant modifications. Since the LHCb experiment planned to operate at a five-fold

increased instantaneous luminosity (up to 2×10^{33} cm^{-2}s^{-1}), with a 40 MHz continuous read-out rate and a fully software-based trigger system, a complete overhaul of the RICH detectors and their read-out electronics was required, to manage the higher event multiplicity and maintain performance.

The main change to the RICH detector system was the complete replacement of the photon detection chain. The Hybrid Photon Detectors (HPDs) had embedded front-end electronics limited to a 1 MHz output rate, so they needed to be replaced. They were succeeded by new multi-anode photomultiplier tubes (MaPMTs), 8×8 pixels devices with a fill factor exceeding 80%. Specifically, 1-inch MaPMTs (2.88×2.88 mm^2 pixel size) were used in RICH1 and the central region of RICH2, while 2-inch MaPMTs (6×6 mm^2 pixel size) were deployed in the outer regions of RICH2, to optimize for varying occupancy while minimizing channel count. The MaPMT modules have a dark-count rate lower than 2.5 kHz/cm^2 and gain larger than 106. RICH1 was equipped with 1888 modules, while RICH2 has 768 1-inch modules and 384 2-inch modules.

To handle the increased data rates and radiation levels (up to 10^7 hits/s per pixel in central RICH1 and 2 kGy total dose), a custom 8-channel radiation-hard front-end ASIC named CLARO was designed, using the 350 nm CMOS AMS technology for the MaPMT read-out [21]. Each channel of the CLARO chip comprises an analog trans-impedance amplifier followed by a discriminator.

The chip operates in binary mode, outputting asynchronous digital pulses with a voltage swing of 2.5 V and variable length, which enables time-over-threshold measurements. CLARO offers extensive configurability. A 128-bit digital register allows for single-channel configuration, including the ability to attenuate input signals by factors of 1, 1/2, 1/4, or 1/8 to compensate for channel-by-channel gain differences. Individual thresholds can also be set and the discriminator offset can be cancelled. Calibration is performed by injecting known charges through a dedicated test capacitor, allowing the determination of the conversion between a threshold DAC code and the absolute charge.

The 8-channel modularity of CLARO perfectly matches the 8×8 pixel arrangement of the MaPMTs. This allows the ASIC to be placed as close as possible to the MaPMT anodes, minimizing parasitic capacitance at the input and reducing susceptibility to electromagnetic interference.

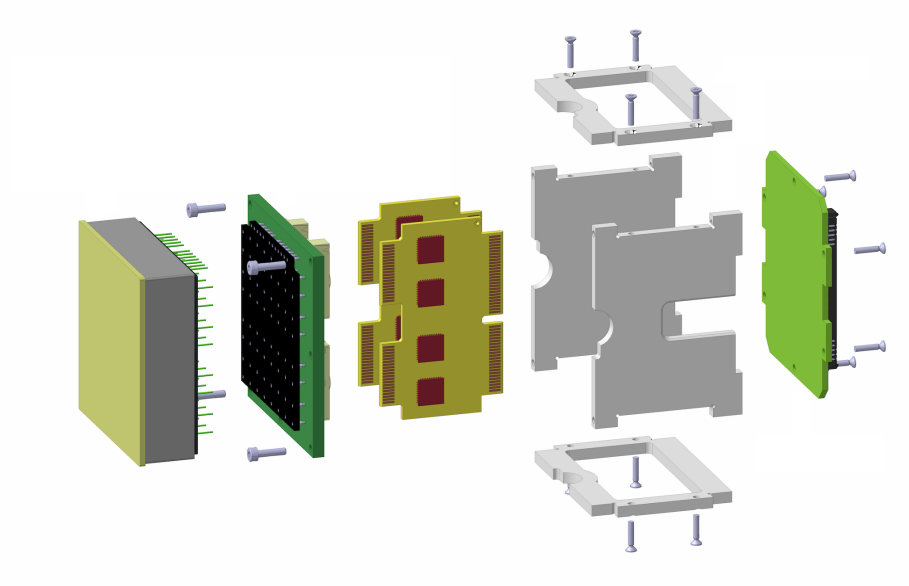


Figure 1.4: H-type Elementary Cell schematic. From the left, the MaPMT is the first element, then the FEB injector (in green), that connects the MaPMT to the FEBs (in yellow), equipped with 8 CLAROs each (in red). The magnetic field shield (in grey) covers the entire cell.

CLARO ASICs are integrated into the "Elementary Cells" (EC)[3], where a baseboard routes anode signals to Front-End Boards (FEBs) which host eight CLARO ASICs each. These FEBs then connect to a backboard that transmits the signals to Photon Detector Module Digital Boards (PDMDBs), where power and control signals for CLARO are generated. In fig. 1.4 a representation of an Elementary Cell is displayed. Starting from the left, the MaPMT is the first piece of the cell. Then, a FEB injector (in green) connects the MaPMT to the FEBs (in yellow), that are equipped with 8 CLAROs each (in red). Eventually, the magnetic field shield (in grey) covers the entire cell, to protect it.

A crucial aspect for the upgraded RICH system is the ability to time-stamp each photon signal in 25 ps bins and apply a 2 ns time gate to reduce out-of-time background and data bandwidth. This functionality is implemented in the PDMDB FPGA firmware, which receives CLARO signals sampled at gigabit rates.

CLARO was designed with low power consumption (approximately 1 mW per channel) and wide bandwidth (baseline restored in ≤ 25 ns). Given the harsh radiation en-

[3]There are two types of EC, an R-type elementary cell (EC-R) is designed to read out four 1-inch MaPMT, while H-type elementary cells (EC-H) read out a single 2-inches MaPMT.

vironment near the beam pipe, CLARO is radiation-hard by design.

To accommodate the new photon detectors, RICH1 and RICH2 support structures were modified. Lastly, the monitoring, control systems and calibration procedures were updated to adapt to the new photon detection chain and read-out infrastructure.

1.4. RICH Upgrade II

LHCb Upgrade II will be specifically designed to unlock the full potential of the High Luminosity Large Hadron Collider (HL-LHC) for flavour physics and beyond [15].

Scheduled for installation during the Long Shutdown 4 (LS4), from 2034 to 2036, this second major upgrade aims to enable the detector to operate at significantly higher instantaneous luminosities, reaching up to 1.5×10^{34} cm^{-2}s^{-1}, and an integrated luminosity of at least 300 fb^{-1}. This increase means dealing with an average of around 40 proton-proton interactions per bunch crossing, a substantial rise from the approximately 6 interactions in Upgrade I.

The LHCb Collaboration has developed three distinct scenarios - Baseline, Middle and Low - for the Upgrade II (Phase II) of its RICH detectors and the overall experiment. These scenarios represent different trade-offs between cost, detector capability and anticipated physics performance, all aimed at operating effectively during the High Luminosity LHC (HL-LHC) era.

The Baseline scenario represents the full realization of the Upgrade II vision, designed to operate at a maximum peak luminosity of 1.5×10^{34} cm^{-2}s^{-1}. This scenario aims for improved performance across all detector aspects to handle higher pile-up, maintain, or even exceed the current detector performance, and enhance the experiment's capabilities. The Middle scenario targets a lower peak luminosity of 1.0×10^{34} cm^{-2}s^{-1}. It aims for reduced cost and complexity, compared to the Baseline, while still providing robust tracking and PID performance. The reduction in luminosity (about 12% less integrated luminosity per year with round optics, or 20% with flat optics) allows for simplifications in detector design and reduced data throughput. The Low scenario also targets a peak luminosity of 1.0×10^{34} cm^{-2}s^{-1}. This is the most de-scoped option, with substantial cost savings leading to significant impacts on key performance parameters and a narrower breadth of the physics program.

Regardless of the scenario, the requirements for RICH upgrade includes improve-

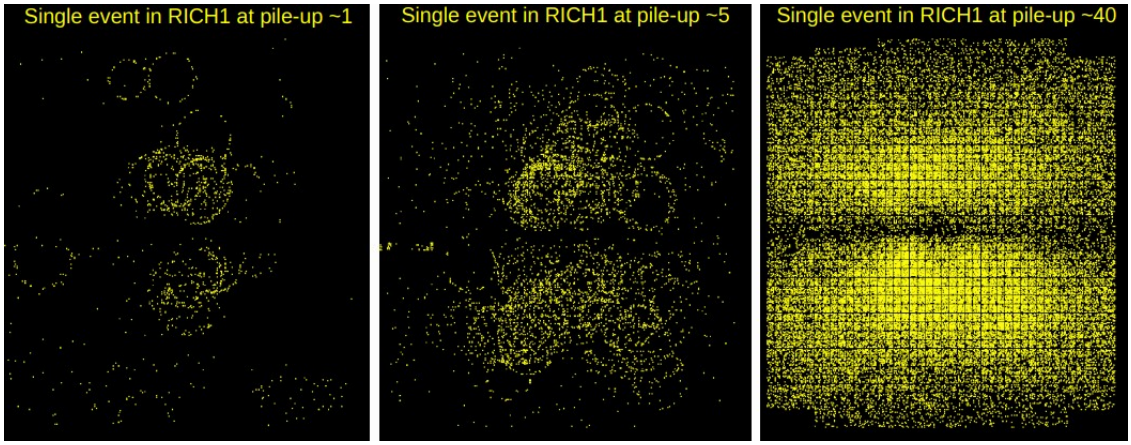


Figure 1.5: Example of Cherenkov rings from RICH1 detector. With a pile-up factor of ~ 40, it is not possible to distinguish and fit the rings.

ment in terms of Cherenkov angle, space and time resolutions. The requirements are based on studies on experimental data and simulations to optimize the particle identification algorithm. A comparison between an event in RICH1 after Upgrade I and a simulation with the event pileup that will be reached after Upgrade II is shown in fig. 1.5. While the Cherenkov rings can be easily distinguish in the plot on the left, it is not possible to correctly identify and fit the high-overlapping rings in the plot on the right, where the pileup factor is ~ 40.

For what it concerns the Cherenkov angle resolution, the goal is to achieve a single-photon angle resolution of below 0.5 mrad, significantly better than the current detector. In particular, for RICH1, the total Cherenkov angle resolution aims for 0.30 – 0.24 mrad, while for RICH2, the target is 0.13 mrad. A substantial reduction in pixel size is needed from the current ~ 2.6 mm to around ~ 1 mm. In the Baseline scenario, pixel size will be 1.4×1.4 mm^2 in high-occupancy regions, 2.8×2.8 mm^2 elsewhere. In the Middle/Low, it will be 2.0×2.0 mm^2 in high-occupancy regions, 2.8×2.8 mm^2 elsewhere (with 5.6×5.6 mm^2 in outer RICH2 for Low). Photon time-of-arrival must be below 100 ps to maintain excellent PID performance. Simulations indicate significant PID performance gains with a 150 ps time window, corresponding to a single-photon time-of-arrival resolution of 38 ps. Also, since LHC will operate at a 40 MHz crossing rate, to keep a good signal-to-noise ratio and a negligible dead time, a dark-count rate limited to below 100 kHz/mm^2 and a good radiation tolerance, are required.

Both RICH1 and RICH2 detectors will undergo major[4] design changes to their optical geometry. The current Multi-Anode Photomultiplier Tubes (MaPMTs) will be replaced by new technologies for Upgrade II[5]. At the moment, Silicon Photomultipliers (SiPMs) and Micro-Channel Plate Photomultiplier Tubes (MCP-PMTs), including Large Area Picosecond Photon Detectors (LAPPDs), are the leading candidates. These are chosen for their superior granularity and timing performance. In particular, SiPMs offer high photon-detection efficiency, especially at longer wavelengths, making them suitable for optimizing the photon detection spectrum towards the green region. They are also insensitive to magnetic fields and operate at low voltages. However, their main drawback is a high dark-count rate, particularly after strong irradiation, which would necessitate cryogenic cooling, neutron shielding and annealing to mitigate. On the other hand, MCP-PMTs are known for their low dark-count rates and excellent timing resolution, typically on the order of O(30 ps). Their primary limitations include restricted lifetime and gain saturation at high rates. However, at the moment there aren't photon detectors meeting all the requirements, so a consistent R&D campaign is ongoing. Part of the R&D campaign to study and design new candidates is the 4DPHOTON project. The project aims to design and develop an innovative detector based on a vacuum tube encapsulating a photocathode, a microchannel plate (MCP) and a pixelated CMOS read-out anode, specifically the Timepix4 ASIC. The project and the detector will be described more in detail in chapter 2.

In this context, the 4DPHOTON detector emerges as a strong candidate, as it is an innovative "hybrid" single-photon imaging detector that combines the strengths of various technologies, specifically addressing the critical requirements of the LHCb RICH Upgrade II. In particular, the Timepix4 ASIC, with approximately $230k$ pixels with a 55 μm pitch, offers extremely high granularity. It also provides sub-200 ps timestamp binning. The 4DPHOTON detector is developed to achieve timing resolution below 50 ps and a position resolution of $5-10$ μm. Moreover, the 4DPHOTON detector is capable of detecting up to 10^9 photons per second. This is critical for the HL-LHC environment, where the instantaneous luminosity leads to about 2000 charged particles per bunch crossing. In chapter 2, a more precise description of the project and the detector will be presented.

[4] In the Baseline scenario. While the design changes will be fewer in the other two scenarios.

[5] In the Low scenario, the outer regions of both RICHes may retain MaPMTs, which would result in insufficient timing resolution for high occupancy conditions.

CHAPTER 2

The 4DPHOTON Project

The 4DPHOTON project, funded by the European Research Council (ERC) under the EU Horizon 2020 programme and hosted by INFN, with contributions from CERN and the University of Ferrara, aims to develop a novel single-photon detector that implements excellent spatial and temporal resolution, low noise, high-rate capability and a large active area within a single device [22].

The core technology is a hybrid detector assembly within a vacuum tube, utilizing a MicroChannel Plate (MCP) coupled to the Timepix4 ASIC as a pixelated anode for precise spatial and temporal resolution.

Chapter 2 is dedicated to the 4DPHOTON project, outlining its conceptual design, main components and operational mechanisms (section 2.1). Since Timepix4 plays a fundamental role within the overall detector architecture, a comprehensive overview of its structure and operation is provided in section 2.2.

2.1. 4DPHOTON Detector Concept

The 4DPHOTON Project is an initiative funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 819627), hosted by the Italian National Institute for Nuclear Physics (INFN) and coordinated by Professor Massimiliano Fiorini, with significant contributions from the European Center for Nuclear Physics (CERN) and the University of Ferrara (UNIFE) [22].

The project addresses a critical gap in current imaging techniques: the lack of a single-photon detector that simultaneously offers excellent timing and spatial resolution, low noise, high rate capabilities and a large active area in a compact, but scalable device.

Existing detectors often compromise on some of these aspects, with many high-resolution measurements relying on gated detectors that lose photons due to their frame-based acquisition. The development of a new generation of photodetectors capable of overcoming these constraints would enable substantial progress across several research domains, including particle and applied physics.

Future high-energy physics experiments, such as LHCb, described in chapter 1, will require improved timing and spatial performance, together with the ability to sustain higher rates than those currently achievable. For example, present photodetectors deployed in the RICH systems (MaPMT) are not adequate for operation in high-luminosity environments: in conditions of high track multiplicity, in fact, Cherenkov rings tend to overlap (as can be seen in fig. 1.5), and global reconstruction algorithms fail to distinguish them, unless precise time-of-arrival information is available, as described in subsection 1.4. A precise timestamp and a fine granularity, that will keep the occupancy below a certain value, especially in the central region, are thus needed for the new photodetectors.

The 4DPHOTON detector is conceived as a "hybrid" assembly, integrating components based on different, independently optimised technologies, as shown in fig. 2.1. In particular:

- **Vacuum tube**: that encapsulates the entire detector, maintaining a pressure of approximately 10^{-10} mbar inside of it.

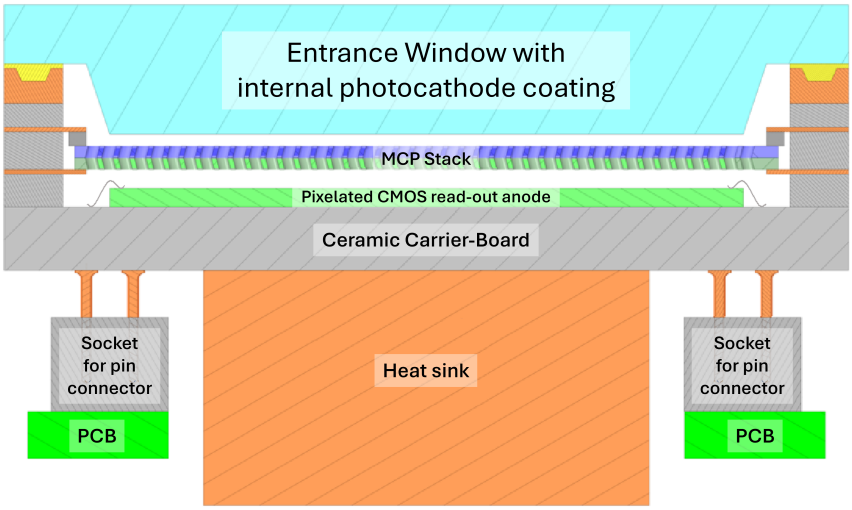


Figure 2.1: Simplified schematic of the 4DPHOTON detector. Here, a Chevron MCP stack configuration is displayed, but a Z-stack configuration is also possible.

- **Entrance window**: a silicate window, transparent in the infrared, visible and ultraviolet ranges, is positioned at the top of the tube and serves as the photon entrance (in light blue in fig. 2.1).

- **Transmission photocathode**: a high quantum efficiency photocathode, that is deposited on the inner surface of the entrance window, converting incoming photons into photoelectrons.

- **Micro Channel Plate (MCP) Stack**: configured in either a Chevron [23] or Z-stack [24] arrangement, to reduce ion feedback, and with an end-spoiling geometry of 1d, 2d or 3d depth, the MCP multiplies the photoelectrons emitted by the photocathode. In 4DPHOTON, they operate at gain of a few 10^4, instead of the standard gain of 10^6, to increase the lifetime of the detector by one order of magnitude.

- **Pixelated CMOS read-out anode**: a bare Timepix4 Application-Specific Integrated Circuit (ASIC), a 65 nm CMOS technology chip developed by the Medipix4 collaboration at CERN, known for its high resolution and rate capability. It acts as the embedded anode, sensing the electron clouds produced by the MCP stack. In green in fig. 2.1. Timepix4 will be deeply discussed in the next section 2.2.

- **Ceramic Carrier-Board**: the bottom cap of the tube assembly. It carries Timepix4, providing a mechanical and electrical interface between the inner and the outer parts of the detectors. In grey in fig. 2.1.

- **Cooling System**: a fundamental part of the system, since Timepix4 can reach a power consumption of 5 W, generating heat inside the vacuum tube. Heat is extracted via a liquid cooling system: a copper heat exchanger mounted below the ceramic carrier (with approximately 4 cm^2 contact area) removes heat from the region closest to Timepix4. Additionally, a copper ring mounted on the entrance window cools the detector from the top side, providing thermal stability from both ends.

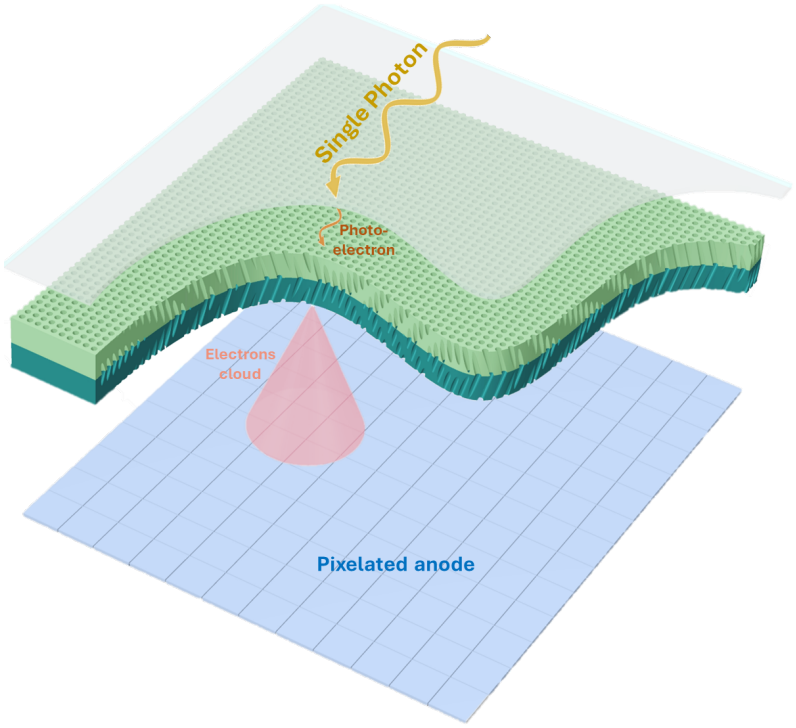


Figure 2.2: Representation of the 4DPHOTON detector operating principle.

When a photon hits the entrance window of the 4DPHOTON detector, a precise sequence of events unfolds to convert the optical signal into digital data. A schematic is shown in fig. 2.2.

In particular, a single photon (in yellow in fig. 2.2) enters the detector through the transparent input window and it interacts with the high Quantum Efficiency (QE) photocathode coated on the inner part of this window. This interaction causes the photon to be converted into a photoelectron (in orange in fig. 2.2) via the photoelectric effect. The photocathode is strategically placed close to the Microchannel Plates (in green in fig. 2.2) to minimize the transverse motion of the photoelectron, preserving spatial resolution. This newly generated photoelectron is then accelerated by an electric drift field towards the MCP stack. Here, it frees and accelerates other electrons, multiplying the electric signal and producing an "electron cloud" (in red in fig. 2.2), that is carried by a second drift field onto the pixelated CMOS read-out anode, a bare Timepix4 ASIC (in light blue).

The first prototypes of the 4DPHOTON detector have been produced and characterized during 2024-2025 [25]. An extensive investigation of the detectors' performance have been carried out during two test-beams at CERN SPS beamline and will be described in section 4.2.

Timepix4, and the conversion mechanism to obtain all the information about the single photon starting from the electron cloud, will be discussed more in detail in the next section 2.2.

A picture of the first 4DPHOTON detector prototype is show in fig. 2.3. The detector is placed on the table upside down, so the connection pins, placed on the back of the detector, can be seen. The entrance windows, on the opposite, is on the side leaned on the table.

2.2. The Timepix4 ASIC

The Timepix4 Application-Specific Integrated Circuit (ASIC) represents the latest generation in a series of highly advanced pixel detectors developed by the Medipix4 Collaboration at CERN [26]. A picture of the ASIC is shown in fig. 2.4. The Medipix Collaboration, established in 1998, has consistently driven ASIC developments for radiation imaging, with a focus on both Medipix (photon counting) and Timepix (time-stamping and energy measurement) families [27].

The Timepix4 ASIC has a large active area of 6.94 cm^2 (approximately 24.7 mm × 29.96 mm). It comprises a 512 × 448 arrays of square pixels, each with a 55 μm pitch.

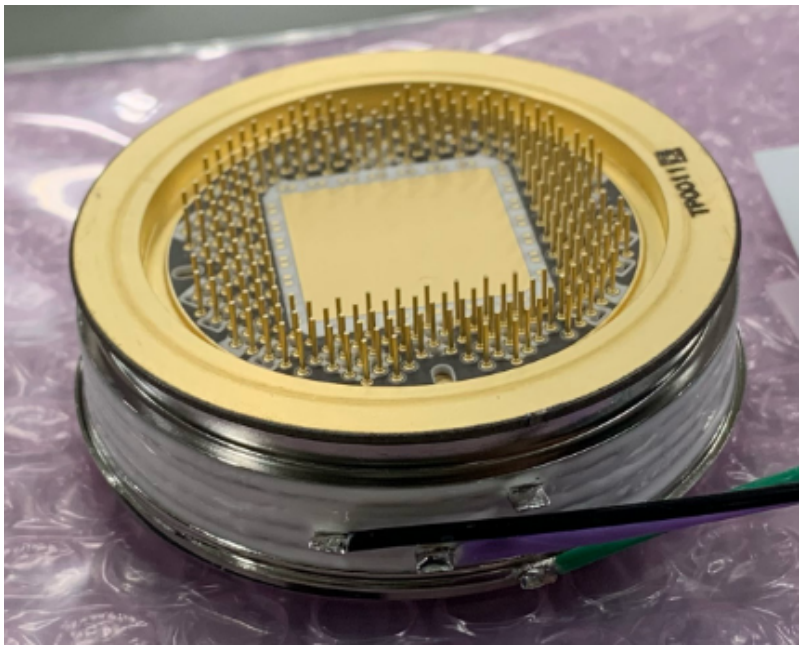


Figure 2.3: Picture of the first 4DPHOTON detector prototype.

A significant design feature is its 4-side buttable architecture, meaning it can be tiled on all four sides to create larger detection areas. This is achieved by integrating the peripheral electronics and Through-Silicon-Vias (TSVs) within the pixel matrix area, which significantly reduces dead space [28].

In can operate in 2 mode, namely data-driven and frame-based, that will be described more in detail in subsection 2.2.2. In data-driven mode, output packets are 64-bit long and they encode information about pixel coordinates, event time-of-arrival and time-over-threshold. The energy resolution is < 1 keV, while the time resolution has a timing bin size of 195 ps. In frame-based mode, every pixel of Timepix4 matrix has a 8 or 16-bit counter that increments every time the pixel is hit. After a fixed time interval, the entire matrix is read and all the information of the *frame* (not zero-suppressed) is sent out. The maximum rate for this modality is 5 Ghits/mm^2/s.

Timepix4 has two different connections, the Slow Control link, a 1 Gbps connection used for configuration and slow read-out, and 16 Fast Links, that can be configured to reach up to 10.24 Gbps, for a total bandwidth of ~ 160 Gbps when acquiring data.

Timepix4 can be bump-bonded to different sensors (an example shown in fig. 2.4) or can be used bare as electron detector (for example in the 4DPHOTON detector de-

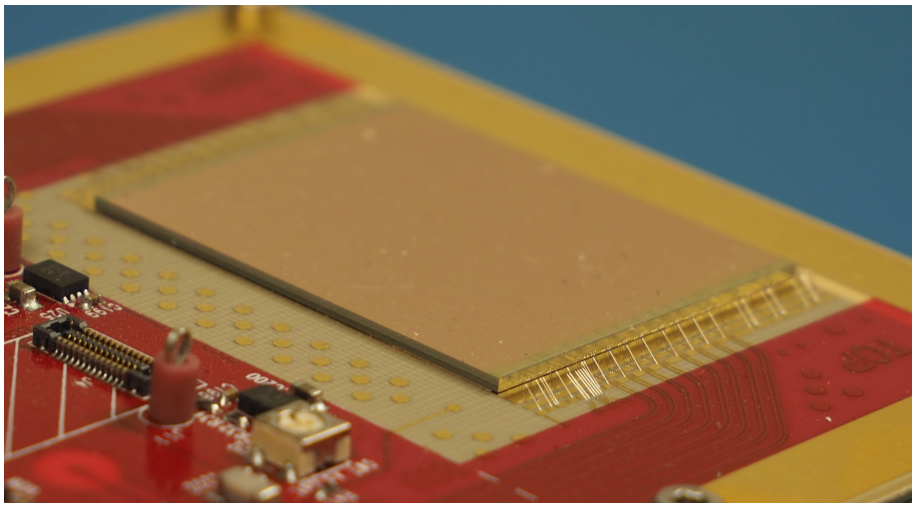


Figure 2.4: The Timepix4 ASIC bump-bonded to a Si sensor and mounted on a chipboard. The wire-bond are visible on Timepix4 side, to connect it to the Chipboard below.

scribed in section 2.1). This versatility allows to exploit Timepix4 in different scenarios, detecting particles in different energy ranges, depending on the sensors bump-bonded to the ASIC. In particular, different applications will be described more in detail in subsection 2.2.3.

2.2.1 The Timepix4 Pixel Matrix

The Timepix4 ASIC features a matrix of 512×448 pixels, with a pitch of 55 μm. This configuration results in a total sensitive area of approximately 6.94 cm^2 with approximately $230k$ pixels. Each pixel enables simultaneous measurement of the Time-of-Arrival (ToA) and the Time-over-Threshold (ToT) for detected signals. The Time-To-Digital converter (TDC) provides a nominal bin size of 195 ps, corresponding to a 56 ps r.m.s. timing resolution per pixel. The pixel matrix also boasts a low equivalent noise charge of $50-70$ e^-, allowing for low operating thresholds around $500-1000$ e^-.

Each pixel in the Timepix4 ASIC integrates analog and digital front-end electronics, in particular:

- **Analog front-end**: each pixel features a Charge-Sensitive Amplifier (CSA) with a programmable feedback capacitance, allowing for different gain modes. Then, the amplified signal is compared against a threshold value by a discriminator

and, although a global (chip-wide) threshold is applied, each pixel also has a 5-bit local threshold setting (trim DAC) to compensate for pixel-to-pixel baseline variations and equalize thresholds across the matrix. The front-end is programmable to be sensitive to either electrons or holes (polarity).

- **Digital front-end**: when a signal exceeds the threshold, the digital front-end processes it to measure the arrival time. This is a three-steps process: firstly, the coarse ToA is determined by the 40 MHz reference clock cycle in which the preamplifier output crosses the threshold, then the fine ToA (fToA) is determined by a Voltage-Controlled Oscillator (VCO) that oscillates at a nominal frequency of 640 MHz. Counting its oscillations until the next 40 MHz clock rising edge improves the timestamp granularity from 25 ns to 1.56 ns.

  Finally, the ultra-fine ToA (ufToA) is determined by four internal phases of the 640 MHz VCO that are latched on the subsequent 40 MHz clock rising edge after the signal crosses the threshold. This further divides the 1.56 ns bins into 8 smaller intervals, providing a nominal bin width of 195 ps.

  This is explained more in details in sections 4.1.1 and a scheme is shown in fig. 4.1. Simultaneously with ToA, the Time-over-Threshold (ToT) is measured. This is the duration the preamplifier output signal remains above the threshold.

Moreover, each pixel can be masked and test-pulses can be enabled. A schematic of the front-ends of a Timepix4 pixel is shown in fig. 2.5.

Each pixel has a dedicated 8-bit register, where users can set the pixel fine- digital to analog converters (DAC) value (5 bits), power enabling (1 bit), test-pulse enabling (1 bit) and mask enabling (1 bit). In DataPix4, the framework developed for Timepix4 management and described in chapter 3 and 4, an equalization routine, described in subsection 3.5.1, tests all the pixels and finds the best DAC values for each one of them, saving the values into a file, so they can be loaded during future Timepix4 configuration. The same routine recognizes also the noisy pixels[1], saving a binary mask into a file that can be loaded during Timepix4 configuration to automatically mask all the noisy pixels.

[1] Noisy pixels are pixels whose front-end electronics exhibit an abnormally high rate of spontaneous triggers unrelated to genuine charge deposition in the sensor, resulting in a great number of unreal events during data taking.

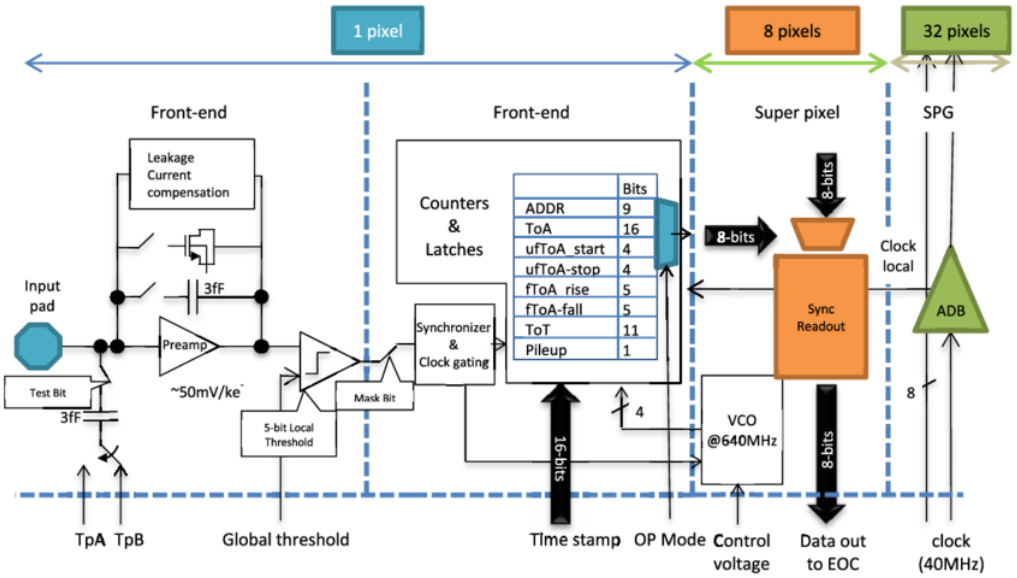


Figure 2.5: A block diagram of the Timepix4 pixel cell and super pixel architecture

The pixel structure itself is organized into two independent matrices ('top' and 'bottom') of 256 × 448 pixels. These are further grouped into SuperPixels (SP) of 2x4 pixels that share a common TDC and a VCO, and SuperPixel Groups (SPG) of four SuperPixels. A representation of Timepix4 pixel matrix organization is shown in fig. 2.6.

Since the two matrices do not communicate among them, each one has its own registers for configuration and each one has its own read-out channels with an independent physical link. Next to the matrices there are three peripheries: a top, a bottom and a central one. The top and the bottom ones (together "edge peripheries") are mostly used for read-out and contains electronic components like 8 serializers, 8x8 routers, phase-locked loops (PLLs), 224 End-Of-Columns (EOC) blocks and I/O pads. The centre periphery contains electronic components as well, but it also features sensors, like temperature and power supply ones, and analog blocks (like Analog-to-Digital-Converters (ADC) and Digital-to-Analog-Converter (DAC)).

While in fig. 2.5 the centre periphery appears between the two matrices, in the real design it is placed under them, so no gap is present between top and bottom matrix.

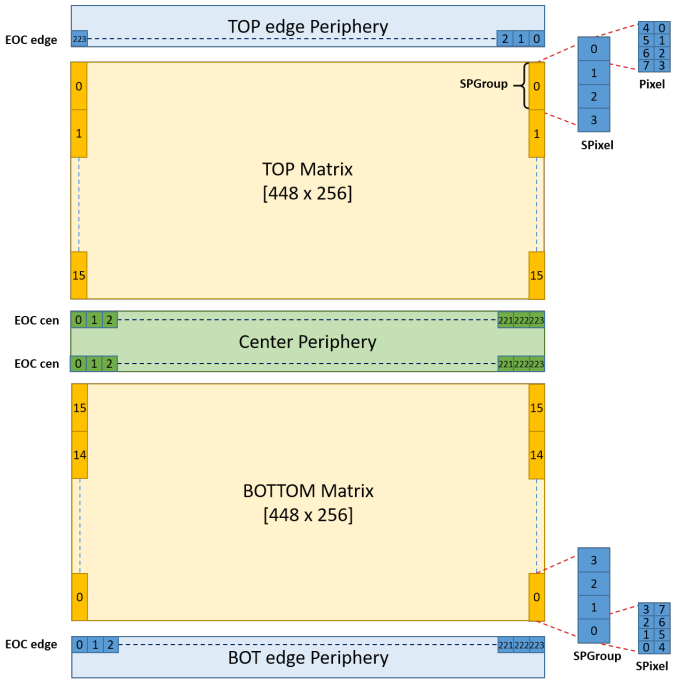


Figure 2.6: Pixel configuration address map, with respect of the Timepix4 matrix organization.

2.2.2 Acquisition modes

Timepix4 offers two distinct acquisition modalities: data-driven (which can operate either in 'ToA-ToT' configuration or in 'photon-counting' mode) and frame-based (using 8- or 16-bits counters).

The data-driven mode is characterized by its zero-suppressed, continuous, event-by-event read-out. In this mode, data packets are generated and transmitted off-chip only when a pixel detects a hit exceeding a predefined threshold. This architecture allows Timepix4 to handle high data rates, with a maximum throughput of 2.5 Ghits/s (equivalent to ~ 3.58 Mhits/mm^2/s), making it particularly suitable for applications where precise timing and per-pixel event information are critical.

In contrast, the frame-based mode operates with a non-zero suppressed read-out of the entire pixel matrix. In this mode, Timepix4 collects all hits within predefined, fixed time intervals called "frames", capturing a complete snapshot of the detector activity. This mode is optimized for extremely high photon counting rates, capable of handling up to approximately 5 Ghits/mm^2/s. The achievable frame rates depend on

the counter depth and the available read-out bandwidth. With the full 160 Gbps read-out bandwidth, Timepix4 can achieve frame rates of up to 86.5k frames per second (kfps) in 8-bit mode and 43.2 kfps in 16-bit mode.

In the next paragraphs, they will be described more in detail.

Data-Driven ToA-ToT Acquisition Mode

The data-driven ToA-ToT acquisition mode is designed for detecting individual particles with high precision in both spatial and temporal coordinates. In this mode, each pixel hit generates a 64-bit data word, which is encoded using the Aurora 64b/66b standard communication protocol [29]. These data packets carry detailed information, including pixel address, coarse and fine ToA, ultra-fine ToA start/stop codes, coarse ToT and a pileup bit. The pileup bit provides a warning if a second event occurs in the same pixel, while the readout of the first event is still ongoing. A schematic representation of the structure of a data-driven 64-bits packet is shown in table 2.1.

This data acquisition mode has been exploited for the majority of the measurements described in chapter 4, since it provides high resolution time and energy information for each event. To launch a data-driven ToA-ToT measurement, a dedicated routine is also present in DataPix4, the framework developed for Timepix4-based detectors management.

Field	Width [#bits]	MSB[2]	LSB[3]	Description
Top	1	63	63	Top: 1, Bottom: 0
EoC	8	62	55	EoC edge address
SPGroup	4	54	51	SPGroup address
SPixel	2	50	49	SPixel address
Pixel	3	48	46	Pixel address
ToA	16	45	30	ToA coarse value @40 MHz
ufToA Start	4	29	26	UltraFine ToA start code
ufToA Stop	4	25	22	UltraFine ToA stop code
fToA Rise	5	21	17	Fine ToA rise @640 MHz
fToA Fall	5	16	12	Fine ToA fall @640 MHz
ToT	11	11	1	ToT coarse value @40 MHz
Pileup	1	0	0	No Pile-up:0, Pile-up:1

Table 2.1: 64-bits packets in ToA-ToT data-driven modality.

Data-Driven Photon-Counting Acquisition Mode

The data-driven photon-counting acquisition mode, unlikely the ToA-ToT one, does not explicitly provide individual Time-of-Arrival (ToA) or Time-over-Threshold (ToT) measurements for each photon hit in the data packet. For each detected event, this mode provides, instead, the pixel coordinates and a photon count for that pixel, using a 24-bit counter for each pixel. A threshold can be set to define the minimum number of hits a pixel must register before it generates an output data word. However, the actual output of a data word might be delayed by a priority encoder in the Timepix4 read-out architecture. If this occurs, the pixel's counter will continue to increment during this waiting period, potentially leading to data words with counts exceeding the initial threshold.

The output word, like the other data-driven acquisition mode, is a 64-bit long data packet, with the information provided in table 2.2.

Field	**Width [#bits]**	**MSB**	**LSB**	**Description**
Top	1	63	63	Top: 1, Bottom: 0
EoC	8	62	55	EoC edge address
SPGroup	4	54	51	SPGroup address
SPixel	2	50	49	SPixel address
Pixel	3	48	46	Pixel address
UNUSED	22	45	25	-
EventCount	24	23	0	PC counter

Table 2.2: 64-bits packets in Photon-Counting data-driven modality.

Frame-Based 8-bit Acquisition Mode

Unlike data-driven modes where only hit pixels transmit data, in frame-based mode, the entire detector matrix is read out synchronously at fixed time intervals, regardless of whether a pixel registered a hit. For each incoming hit above a defined threshold, a pixel counter is incremented. In this specific mode, the counters have a programmable depth of 8 bits. Actually, each Timepix4 pixel counter has a total of 16 bits, so in the 8-bit frame-based, the counter is split into two 8-bits counters, enabling continuous

[1] Most Significative Bit
[2] Least Significative Bit

read/write operations. While one counter is actively counting incoming hits, the other is being read out.

When the matrix is read out, each Super Pixel (SP) generates a 64-bit data word, and the count information from all 8 pixels belonging to the same SP is combined into a single 64-bit data packet. The pixel address is not explicitly encoded in the output word; instead, it is inferred from the position of the 8-bit data within the packet. The position of the 8-bit data within the 64-bit packet uniquely identifies which of the eight pixels in the SP generated it, while the global pixel coordinates within the full 448 × 512 matrix are determined from the ordering of the data packets in the readout sequence.

The Timepix4 matrix is divided into 8 'segments' per periphery (L0-L3, for the left quadrant, and R0-R3, for the right one). Each segment correspond to 1/8 of the total number of the double columns (28 double columns). In particular, for example, the L0 segment includes the 1st, 5th, 9th, ... until the 109th double column. A single half matrix consists of 64 Super Pixels per double column, leading to a total of 1792 data packets being read out per segment.

This implies that, if a single data packet is lost, the entire frame must be ignored, since it is not possible to reconstruct the hits position anymore inside the segment.

The packet sequence is shown in table 2.3, while the packet structure is shown in table 2.4.

Frame-Based 16-bit Acquisition Mode

The frame-based 16-bit acquisition mode is, like the 8-bit frame-based, a non-zero-suppressed full frame read-out. For this mode, the pixel counters exploit all their 16 bits as a single 16-bits counter, so it is not possible to have a continuous read/write and a dead time between the frames[4] is present.

Each Super Pixel (SP) within the Timepix4 ASIC generates 64-bit data words for output, but, when operating in the 16-bit counting depth mode, each Super Pixel generates two 64-bit data packets to contain the count data, instead of just one. The structure of these packets dedicates 16 bits to each of four pixels. In this mode, like the 8-bit frame-based, the pixel address is not explicitly encoded within the output data word itself. Instead, it is inferred from the data word's position within the overall data packet.

[4]The time is proportionate to the read-out bandwidth. Using all links at maximum speed, the read-out time is approximately 22 μs.

Packet	Description
Idle	Idle data word before frame 1 start
Frame start	Frame 1 starts
Segment start	First segment of frame 1 starts
Data	First data word of frame 1
N x Data	Rest of first segment data words
⋮	⋮
Segment end	First segment of frame 1 ends
N2 x Segments	Remaining segments of frame 1
⋮	⋮
Frame end	Frame 1 ends
Idle	Idle data word before frame 2 starts
Frame start	Frame 2 starts
N3 x Frames	Remaining frames until read-out disabled
⋮	⋮

Table 2.3: Control and Data Packet sequence in frame-based modality.

Field	Width [#bits]	MSB	LSB	Description
Pixel0	8	63	56	Counter value of Pixel 0
Pixel4	8	55	48	Counter value of Pixel 4
Pixel1	8	47	40	Counter value of Pixel 1
Pixel5	8	39	32	Counter value of Pixel 5
Pixel2	8	31	24	Counter value of Pixel 2
Pixel6	8	23	16	Counter value of Pixel 6
Pixel3	8	15	8	Counter value of Pixel 3
Pixel7	8	7	0	Counter value of Pixel 7

Table 2.4: 64-bits packets in 8-bits frame-based modality.

The read-out sequence is the same as the one shown in table 2.4, but, since now each 64-bits packet can hold information about half the pixels (4 pixels instead of all 8), we need to read twice the number of the packets to create a complete frame. In particular, the first frame (between a `frame start` and a `frame end` control packets) will have all the information about the pixels number 0, 1, 2, 3 (inside a Super Pixel) of all Timepix4, while the second frame will contain all the other pixels. Merging the two frames, the entire pixel matrix can be reconstructed. Data packet structure is shown in table 2.5.

Field	Width [#bits]	Bit stream position
Pixel 0/4	16	[63:56] << 8 + [47:40]
Pixel 2/6	16	[55:48] << 8 + [39:32]
Pixel 1/5	16	[31:24] << 8 + [15:8]
Pixel 3/7	16	[23:16] << 8 + [7:0]

Table 2.5: 64-bits packets in 16-bit frame-based modality. The first time, counters of pixels number 0, 1, 2 and 3 are sent out, while the second time, the other pixels' counter are sent.

2.2.3 Timepix4 applications

The Timepix4 ASIC can be bump-bonded to different sensors or can be used coupled, for example, with MCP stacks. This high versatility allows the use of Timepix4 in a wide range of applications, from elementary and particle physics, to imaging and medical physics, from archaeometry to physics for cultural heritage. Some of Timepix4 and 4DPHOTON applications will be presented in chapter 4.

The whole Timepix and Medipix chips family is exploited in multiple fields of application [30]. In particular, there are 4 macro-areas, listed in the following paragraphs.

Space Dosimetry and Aerospace Applications

The goal of using Timepix-based systems in space is to develop advanced radiation detectors capable of providing precise, real-time monitoring to better protect astronauts and critical spacecraft systems, particularly as NASA prepares for future deep-space missions. Compared to previous NASA instrumentation, Timepix-based systems are

smaller, lighter and more power-efficient, offering a compact and low-power solution suitable for constrained space environments.

Beyond measuring the overall radiation dose, these detectors can also determine the spatial distribution of radiation as it passes through the pixelated sensor.
By analysing the characteristic traces left by particles in the detector, different types of radiation (including protons, heavy ions and electrons) can be classified and quantified, allowing for detailed characterization of the radiation environment.

Timepix chips have already been deployed on several important NASA missions. For example, five chips have been operating on the International Space Station (ISS) since 2012, monitoring the radiation field as a function of time and providing precise measurements of particle spectra, charge and velocity within the spacecraft [31].

More recently, Timepix4 has been integrated into the Artemis Program and Lunar Exploration missions. During Artemis I, four Timepix chips were flown: three within the HERA [32] detector and one as part of the first deep-space biology experiment, helping to study radiation effects beyond low Earth orbit.

Figure 2.7: The Timepix system is shown on the ISS Cupola. On-orbit imagery is courtesy of www.spaceflight.nasa.gov.

Medical Imaging and Therapy

Conventional radiography and computed tomography (CT) rely on X-ray photons to produce images, but traditionally these images are black-and-white and based solely on the total X-ray energy absorbed by tissues. The Timepix chips represent a major technological breakthrough, as they allow the energy of each individual photon to be measured. By analysing the energy spectrum recorded in each pixel, the scanner can distinguish between different materials based on their density and atomic composition.

For example, tissues that appear identical in standard greyscale images—such as fat and water—can now be differentiated, and subtle features like calcium deposits or disease markers can be clearly identified. This colour X-ray imaging technique produces clearer and more accurate images, offering significant potential to improve diagnostic accuracy for clinicians.

The first 3D colour X-ray images of human extremities were acquired in 2018 using a scanner developed by MARS Bioimaging Ltd [33] that employed Timepix3 detectors.

Building on this technology, subsection 4.3.2 presents micro-CT scans carried out at INFN Trieste using Timepix4, demonstrating the next generation of high-resolution, colour-sensitive X-ray imaging for biomedical applications.

Material Analysis and Cultural Heritage

The most detailed application of Timepix technology in the field of cultural heritage concerns the authentication and elemental analysis of artworks. In particular, the Czech start-up InsightART has developed a robotic X-ray scanner named RToo that exploits Timepix detectors to analyse large paintings (up to 2 m^2) and atypically shaped objects (such as statues). An interesting case study involved the painting *The Madonna and Child*, attributed to the Renaissance master Raphael, but with a questioned authenticity. InsightART scanned the artwork over three days to obtain 11 new high-resolution maps of the elemental composition of the paint taken at different X-ray wavelengths and a group of experts confirmed that the painting was original, demonstrating the power of Timepix-based imaging for non-invasive art diagnostics [34].

Building on this approach, subsection 4.3.1 describes an ongoing research project focused on developing a hybrid pixel detector system for X-ray imaging of cultural her-

itage objects, based on the Timepix4 ASIC coupled with a Cadmium Telluride (CdTe) sensor.

Education

The use of Timepix-based detectors in schools is focused on transforming STEM education by allowing students to physically observe and experiment with concepts like particle physics. The Minipix detector is a commercially available USB read-out system that incorporates the Timepix chip, making the system very handy and capable of running on any PC [35]. Designed for educational purposes, it allows students to observe in real-time the traces that the particles interacting with Minipix leave. An intuitive graphical user interface displays the event and the statistics, allowing students also to download the data and analyse it offline with custom algorithms.

CHAPTER 3

DataPix4: A C++ Framework for Timepix4-based systems

DataPix4 is a C++ framework for the management of the Timepix4 ASIC [36]. It allows to configure the setup, start data acquisition, save the data and visualize some statistics in real-time. In this chapter, a full description of the framework will be given, starting from the choices that were made when designing the software to the actual implementation details. A brief explanation on how to use the software will be given, explaining all its functionalities.

A section will then be dedicated to the clustering algorithm used for the online and offline data analysis. Furthermore, a section will be dedicated to the Graphical User Interface that has been created to simplify the use of DataPix4 when performing some fundamental tests on Timepix4. All the tests performed during the development of DataPix4 software will be explained in details showing the main features and results obtained.

3.1. Requirements and General Structure

For the complete management of the Timepix4 ASIC, a flexible software is needed. This is required to set up a communication channel with the chip and to fully configure it, exploiting the different registers it provides. Once Timepix4 is ready to acquire data, the software must be able to receive and store securely on disk all the data packets. While collecting data, it should also display some real-time statistics, so that the user can monitor data collection and extrapolate event information while Timepix4 is acquiring.

In addiction to those fundamental functionalities, the software must also be user-friendly, with a steep learning curve for beginner user, so that they can begin working with Timepix4 even without strong programming skills. At the same time, it should also allow expert users to manage every low-level aspect of Timepix4, so that every custom configuration is possible.

Since Timepix4 has a maximum bandwidth of 160 Gbit/s, when using the data-driven mode with all fast links enabled, the data acquisition software must be optimized enough to receive and save all the data.

With these prerequisites taken care of, I decided that the most suitable programming language for the implementation of the software was C++. It allows for low-level manipulation of data, both for configuration and read-out, with the possibility to further improve software optimization. Being an object-oriented language, it also allows to build higher-level objects, creating a hierarchy of classes resulting in a strong, but flexible, software structure.

All the fundamental components of the hardware setup have a software counterpart that will be described more in detail in the following sections. In addition to this, the user can create custom classes and add them to DataPix4, to communicate with other devices while using Timepix4. For example, it is possible to create classes for the management of devices like power supplies or robotic arms, that can be managed and exploited during acquisitions from the same C++ script.

DataPix4 classes can also be enclosed into Python wrappers, so that they can be treated like Python's objects and can be included into any Python's framework. An example of a DataPix4 integration into a Python's framework can be found in section 4.3.2.

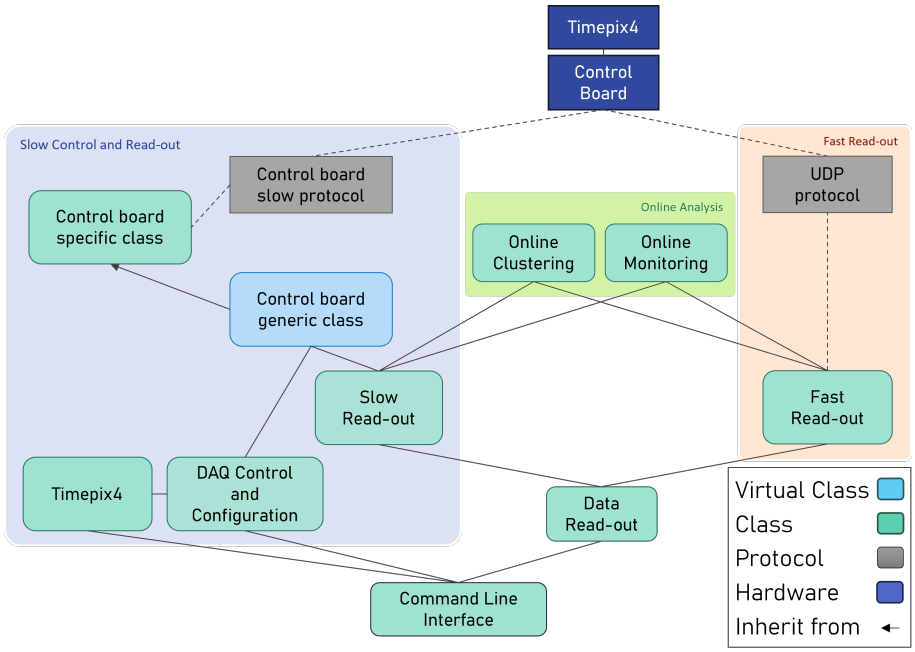


Figure 3.1: DataPix4 Classes Scheme

As mentioned before, DataPix4 has a solid structure made of C++ classes. A diagram can be seen in fig. 3.1. The three main parts are highlighted with different colours. Starting from the left, the Slow Control and Read-out portion, in the violet box, contains all the classes dedicated to Timepix4 and control board configuration, as well as data read-out via Slow Control. On the right, the classes dedicated to fast read-out, in the orange box, are included in the second part. Eventually, in the middle, green box, online monitoring and clustering classes, that can be used with both read-out modes, are included in the third group.

The use of a control board virtual class (in light blue in fig. 3.1) makes DataPix4 easy to adapt to any type of control board that supports Timepix4. If a change in the control board is needed, the user just needs to create a new control board derived class, using the virtual one as base class. There are only 4 methods that need to be implemented: two for reading and writing a Timepix4 register and two to perform the same operation on a control board register. In addition to those, a method to perform a software reset of Timepix4 is recommended, as well as one to perform a full configuration of the control board fast links. All the methods in the control board virtual class rely only on those four.

For the communication, the slow control usually uses a client-server paradigm with a custom protocol, according to the control board in use. For the fast read-out part, on the contrary, the UDP protocol is usually chosen and implemented in the control boards [37]. This ensures a high-speed communication between the board and the computer, suitable for Timepix4 high data rates. If the custom control board does not support UDP protocol for fast links communication, some methods of the fast read-out class may need to be re-implemented. The online analysis part does not need to communicate with hardware, so no modification is needed even in the case of changing in communication protocols.

Once the custom class is implemented and the custom Control Board communication protocol is correctly installed and configured, all the other classes of DataPix4 become available and the user can start working with Timepix4.

In this way, DataPix4 can be easily adapted to work with every setup that supports Timepix4.

DataPix4 is compiled and tested under Ubuntu 20.04 LTS and Ubuntu 24.04 LTS. It could also be compiled and run under other Linux distribution, according to the pre-requisite of the control board communication protocol. It was compiled using respectively g++ version 9.4.0 and 13.3.0, but can be compiled with every compiler that supports C++ 17. The installation relies on cmake and make.

To use DataPix4 with a custom control board, it is necessary to add all the instructions that are needed to compile the slow control communication protocol and to correctly link its libraries to the custom control board derived class to the `CMakeLists.txt` file.

3.2. Setup Configuration

The first thing to do when using the Timepix4 ASIC is the configuration of both the chip and the setup. As mentioned before, DataPix4 can be easily extended with dedicated classes for external devices, so that all of them can be configured from the same interface. This is especially useful when the configuration order matters: in this case, the user can write the configuration instructions in the correct order within the C++ script and does not need to manually synchronize different scripts that configure different components.

The Timepix4 ASIC can be configured via slow control though its registers. As mentioned in section 2.2, Timepix4 has more than 150 registers, each of which has 16 or more bits that can be set. To configure the ASIC, the user needs to set its registers with the correct values and in the correct order. This can be very challenging, especially because changing just one bit in a register can change the entire ASIC configuration. For this reason, different methods that allows the user to configure different aspects of Timepix4 were written. High-level methods allow the user to configure the chip in a simple and fast way, for a standard, but complete, configuration. On the other side, lower-level methods can be exploited by expert user to personalize the standard configuration, or configure the entire chip from scratch, by writing small groups of registers, or even single ones.

Regardless of the expertise of the user and the choice of the configuration method, a DAQ object is required to communicate with the setup. Using as example the control board developed at INFN Ferrara, called IDAQ [38], the object will be created by calling the constructor of a derived control board class, as shown below.

```
1 DAQ* board = new IDAQ(ip_address, port);
```

The constructor has two mandatory arguments, an IP address and a port, that will be used to create a communication channel with the control board.

Since the control board can accommodate more than a Timepix4 ASIC, a method, `vector<ID> addTpx4(int N)`, to assign them an ID and/or an index is needed. The ID and the index must then be passed as firsts arguments to all the other DAQ methods, to send the request to the right Timepix4 ASIC.

Once the DAQ object is instantiated, the configuration process can begin. As mentioned before, Timepix4 must be configured through its registers. The values written to these registers determine its behaviour and the specific functions it will perform (such as different types of acquisitions).

To set it up properly, the correct values need to be written to these registers and this can be done using the methods of the virtual control board class of DataPix4. All the API in this class relies on two fundamental methods: `readTpx4()` and `writeTpx4()`. The first one performs a reading of a Timepix4 register, while the second one performs a writing. In the virtual class, only the prototypes are present and they must be implemented in the control board specific class, since their functioning is strictly dependent

on the communication protocol used by the control board.

When implementing the `readTpx4()` and `writeTpx4()` methods, the concurrence must be taken into consideration. DataPix4 is a multi-threaded software and, as will be described in the next subsection 3.3.1, the read-out via slow control exploits different threads that read dedicated Timepix4 registers in parallel. This, however, implies that the `readTpx4()` and `writeTpx4()` methods must behave like atomic functions, otherwise concurrency problems or undefined behaviour may occur. For example, in the case where, to perform a single Timepix4 register reading, different control boards registers needs to be written (for example indicating which Timepix4 register to read, how many bytes is the register long, ...), a mutex or an equivalent synchronizing method needs to be exploited when implementing the two methods, so that they can be thread-safe.

To write a single Timepix4 register, the `writeTpx4()` method should be used. This takes as input the `ID` of the register and the `value` that needs to be written. Additionally, the `size` (in bits) of the value can also be specified through an optional third parameter[1]. Since some registers can be up to 512 bytes long, and there are no standard data types longer than 64 bits in C++ [39], the `value` must be passed as an array of `unsigned char`, so that there are no limitations on the length of the data to be written.

However, since the majority of registers are 16 or 32 bits long, a simplified version, that accept an unsigned 32-bit integer[2] as the `value` argument, has been implemented. An example of a writing can be found in the box below.

```
1  // Creating an array of unsigned char
2  unsigned char *val = (unsigned char *)malloc(2 * sizeof(char));
3  val[0] = 0x00d0;
4  val[1] = 0xfe01;
5
6  // Write value on register
7  int err = board->writeTpx4(tpx_id, idx, MATRIX_CONF, val, 16);
8  if (!err) {
9      std::cout << Correctly written on Timepix4 reg: << std::hex << MATRIX_CONF << std::endl;
10 }
```

[1] If not specified, the length is obtained from a map of register stored in RAM. Although this is a fast operation, it still takes time, so if the user wants to optimize more its code, the register length should be passed.

[2] If the register is 16 bits long, the value is truncated.

```
free(val);
// Same operation, but with the simplified version
uint32_t value = 0x00d0fe01;
int err = board->simpleWriteTpx4(tpx_id, idx, MATRIX_CONF, value);
if (!err) {
    std::cout << Correctly written on Timepix4 reg: << std::hex << MATRIX_CONF << std::endl;
}
```

To read the value of a Timepix4 register, the `readTpx4()` function should be called. This takes as arguments the ID of the register and, optionally, its length (in bits) and returns a pointer to an object derived from `ctrlBoardReplyType`. This is a virtual class defined in the control board virtual class and it is designed to temporarily store the result of a readTpx() operation. This is required because different control boards can use different communication protocols that may return different data types [3]. When working with a custom control board, it's necessary to create a `ctrlBoardReplyType` derived class, which must implement the 5 methods for data conversion: `toUint16()`, that returns, when possible, a `uint16_t` value, `toUint32()` and `toInt32()`, that returns, when possible, respectively a `uint32_t` and a `int32_t` value, `toString()`, that returns a `String`, and `toFourUInt64()`, that returns a vector of four `uint64_t`. The last one can be exploited when reading acquisition data from slow control 256-bit registers, since the data packets are 64-bits long and a register can store up to four of them. The reply object should keep the original value in the `data` attribute, with the data type used by the control board.

When calling `readTpx4()`, the user can directly convert the value into a standard type, making the `ctrlBoardReplyType` derived class transparent.

An example is shown in the box below.

```
// Read register and save the reply in a 16-bit unsigned integer
uint16_t reply = board->readTpx4(tpx_id, idx, MATRIX_CONF)->toUint16();
std::cout << "Data read: " << reply << " from reg: " << MATRIX_CONF << std::endl;
```

To perform multiple readings, a dedicated `readTpx4Repeat()` method has been implemented. This takes as arguments the ID of the register and the number of reading it has to perform. Additionally, register length and a timeout parameters can be

[3] For example, the SPIDR4 control board returns a `string`, while other boards might return values in other data types, like array of unsigned integer or char.

passed. This approach allows the overhead of a single function call to be distributed across multiple reads, rather than making one call for each individual register read. The `readTpxRepeat()` method is exploited by the slow read-out thread, that will be discussed more in detail in section 3.3.

A hierarchy of methods was created based on the two fundamental functions mentioned earlier. These functions, which perform single register reads or writes, were used to build methods that configure one or a few registers to adjust specific settings (such as setting the shutter's open or close time). These, in turn, were used to create higher-level methods that configure groups of registers for more general settings (e.g., adjusting all internal clock settings for Timepix4). At the top of the hierarchy there are a few high-level methods, described below, which offer a simple yet complete chip configuration.

Depending on the user's experience level and the need for customization, they can choose which methods to use and how detailed they want to be when modifying specific settings.

In addition to high-level methods, to make the Timepix4 setup more user-friendly, a configuration file has been created, where the user can specify the main parameters to be set (e.g., shutter opening time, test-pulse activation, fast-link or slow-control usage, etc.). This file is loaded by a specific function that reads it and creates a `struct` of type `Tpx4Config` with the configuration values. The resulting `struct` can then be passed as an argument to the `confTimepix4()` method, which handles all the main settings of the chip. Additionally, it can be used within the C++ script to retrieve the configuration values, in case the user needs them.

An example of a configuration file can be found in Appendix A, while an example of a Timepix4 configuration can be found in the box below.

```
// Create a configuration struct from the configuration file
Tpx4Config confStruct=Tpx4::createTpxConfStructFromFile("tpx4.conf");
// Use the struct to configure Timepix4
board->confTimepix4(tpx_id, idx, confStruct);
// Can use values from the configuration struct
int shu_open = confStruct.Values["SHUTTER TIME ON"];
std::cout << "Shutter open for: " << shu_open << " s" << std::endl;
```

In addition to the `confTimepix4()` method, that performs a basic, but complete,

configuration of the ASIC, the `confThreshold()` one configures the Timepix4 DACs that set the global matrix threshold. This methods takes as argument a *struct* of type `Tpx4ThresholdConf` and the `Peripheryfrom` which Timepix4 receives its slow control clock (top, bottom or both).

The *struct* can be created using a dedicated method from the Timepix4 class called `createTpxThrConfFromStruct` that takes as argument the `Tpx4Config` *struct* created before, a boolean value that indicates whether to perform or not a linearization of the threshold and the value to set as threshold. An example is shown in the box below.

```
// Create a threshold configuration struct
Tpx4ThresholdConf thrStruct = Tpx4::createTpxThrConfFromStruct(confStruct, linearize_thr, thr);
// Change struct's values at run-time
thrStruct.verbose = true;
// Configure Timepix4 thresholds using the struct
board->confThreshold(tpx_id, idx, thrStruct, TOP);
```

Once those two high-level functions have been called, Timepix4 is almost ready to start the data acquisition, with the mode specified in the configuration file. This approach for the configuration is useful for the first tests, to assure everything is working properly and to check if Timepix4 is responding well. It is also convenient for a first basic configuration at the beginning of every C++ script, that can then be fine-tuned using lower-level methods.

After a general configuration of the ASIC, also the pixels of the Timepix4 matrix must be configured. The matrix, described more in details in subsection 2.2.1, is composed of 448x512 pixels with a pitch of $55\mu m$. Each pixel has a 8-bit long register that can be set to change the pixel's behaviour, in particular: 5 bits may be used to set the local DACs used to tune the global threshold value, 1 bit is to enable or disable the power supply, 1 bit can be used to mask the pixel and the last bit can be used to enable test-pulse on it. To simplify the pixels configuration, a `pixelConf()` method may be called to convert the four values into a 8-bit unsigned valued that can be written on the pixel's register.

Since manually writing the $\sim 230k$ Timepix4 registers can be extremely time-consuming, the `configPixels()` method can be used. This takes as argument a 3D vector[4] of un-

[4]Timepix4 indexes its pixels with a [double column, super pixel, pixel] coordinate system. The vector must be formatted according to those coordinates.

signed 8-bit integers, one for each pixel, and a `Periphery`, choosing from `TOP` or `BOT`, in which the values should be written. It is necessary that the 3D vector is formatted according to the Timepix4 coordinates system (double column, super pixel, pixel), but the user can choose to configure all the pixels with a (X, Y) coordinate system and then use a dedicated function to convert the coordinates, called `addrColRowToEoCSpPix()`, as shown in the example below.

```
// Define a new data type for pixels configuration
typedef vector<vector<vector<uint8_t>>> tpxConfigMatrix;
vector<int> addr(4);
bool power = true, tp = false,  mask = false;

// Create the two pixels vectors
tpxConfigMatrix pixelConfigTop(224, vector<vector<uint8_t>>(16,vector<uint8_t>(32)));
tpxConfigMatrix pixelConfigBot(224, vector<vector<uint8_t>>(16,vector<uint8_t>(32)));

// Load pixels' DAC values from a file into a 2D vector
vector<vector<int>> dac_thr_pattern(MATRIX_SIZE_X,vector<int>(MATRIX_SIZE_Y));
Timepix4::fileToMatrix("dac_values.dat", dac_thr_pattern);

// Load masking map from a file into a 2D vector
vector<vector<int>> mask_pattern(MATRIX_SIZE_X,vector<int>(MATRIX_SIZE_Y));
Timepix4::fileToMatrix("mask_pattern.dat", mask_pattern);

// A double for-loop configurates each pixel
for (int x = 0; x < MATRIX_SIZE_X; x++) {
  for (int y = 0; y < MATRIX_SIZE_Y; y++) {
    // Convert [X,Y] coordinates into [double column,super pixel,pixel]
    addr = board->addrColRowToEoCSpPix(y, x);
    // Top and bottom must have two separated arrays
    if (addr[0] == TOP) {
      pixelConfigTop[addr[1]][addr[2]][addr[3]] = board->pixelConf(dac_thr_pattern[x][y],
      power, tp, mask_pattern[x][y]);
    } else {
      pixelConfigBot[addr[1]][addr[2]][addr[3]] = board->pixelConf(dac_thr_pattern[x][y],
      power, tp, mask_pattern[x][y]);
    };
  };
};
```

```
33 // Call the API that writes the configurations on the registers
34 board->ConfigPixel(tpx_id, idx, pixelConfigTop, TOP);
35 board->ConfigPixel(tpx_id, idx, pixelConfigBot, BOT);
```

Other methods have been implemented for a quicker setup of the pixel matrix, for example the `ConfigAllPixels()`, that takes as arguments four parameters (DAC value, power supply, test-pulse and mask) and sets every pixel with the same values, or the `ConfigSuperPixelCol()`, that configures every double column with the same four values, instead of every pixel. The method described in the example above, however, is a great compromise between user-friendliness and flexibility in the configuration, and can be exploited to load custom pattern for DAC values, test-pulse and mask, as shown in the code above (`dac_thr_pattern` and `mask_pattern` matrices). According to the user needs, other configuration methods may be called. The most frequent are:

- `configureTP()`, that configures the fundamental parameters of test-pulse (for example the number of test-pulses to send, the time duration of a test-pulse and the time length between two pulses, whether the test-pulses is external and/or digital, and the linking between the shutter and the test-pulses).
- `enableCTPR()`, that enables the test-pulses only in the double columns that have at least one pixel enable to receive it.
- `configFastLinks()`, that configures the control board's fast links, for example setting the server's IP addresses and ports, and setting the link's number and speed.

Once Timepix4, its pixels matrix and, if needed, the other parameters are set, the user can begin the data acquisition. More information about the data acquisition can be found in the next section 3.3.

3.3. Data Acquisition read-out

Timepix4 has two communication links that can be exploited to receive data: one slow-control connection, also used for configuration, and 16 fast links, with a programmable speed up to 10 Gbit/s each.

Following the structure of the hardware architecture, DataPix4 provides two read-out classes, one for each communication channel, called `slowReadout`, for read-out via slow control, and `fastReadout`, to exploit fast-links. Both classes are designed to be high-level and user-friendly, with a simple constructor, some APIs for a fine-tuning of read-out settings and a few methods to start and stop the data acquisition. Despite the high-levelness, it is still possible to reach the lower-level objects using dedicated *getter* methods.

The `slowReadout` class will be described more in detail in subsection 3.3.1, while the `fastReadout` will be described in subsection 3.3.2. Both objects can be connected to online monitoring and online clustering classes, setting the corresponding flag. The online visualization and clustering will be described in subsections 3.4.1 and 3.4.2.

3.3.1 Slow Read-out

For the slow communication, used for Timepix4's configuration and slow read-out, a dedicated slow control link may be exploited, supporting data transmission up to a maximum bandwidth of 1 Gb/s. This interface is intended for non–time-critical operations and provides a reliable channel[5] for accessing configuration registers, monitoring system status and retrieving small volumes of acquisition data when high throughput is not required.

The communication between Timepix4 and the computer, when using the slow control link, is done using a client-server paradigm implemented through the control board protocol. The control board acts like a server and its firmware accepts the computer's requests, performs the corresponding operations on Timepix4 e sends back the reply to the client.

As dictated by the standard client–server model, the control board is not allowed to transmit data autonomously to the client; it can only reply to explicit requests from the client. Consequently, when the slow-control link is used for data read-out, the client must continuously poll the server in order to retrieve the available data packets.

Timepix4 has two 256-bits long registers, one for each matrix periphery, that can contain up to four 64-bits data packets each. During data acquisition, these registers accumulate packets coming from the corresponding top or bottom periphery. To re-

[5] Slow control communication uses the control board protocol which is usually reliable, ordered and error-checked.

trieve the data, the client needs to read the value of the registers, using the `readTpx4()` method (or the more efficient `readTpx4Repeat()` one).

The reading of those registers is destructive so, every time the register is accessed, the data packets are replaced with newer ones from an internal Timepix4 FIFO. If data is not read frequently enough, the incoming packets will overwrite unread ones, resulting in data loss.

For this reason, the `slowReadout` class launches one or more dedicated threads that continuously poll the register(s) to read the acquisition data.

When creating a `slowReadout` object, the user just needs to specify the desired output format of the data (raw or decoded) and the destination path. Additional features, such as online data visualisation and online analysis, can also be enabled through two optional flags, as illustrated in the example below.

```
#include "slowReadout.h"

// Set the online analysis flags
bool onlineMonitoring = true, onlineClustering = false;

// Creates one object to acquire data
// A control board object pointer, as well as the Timepix4 ID and index,
// are necessary parameters to acquire data
slowReadout *slowThread = new slowReadout(&board, tpx_id, idx, rawReadout, outputPath,
    onlineMonitoring, onlineClustering);
```

By default, the read-out object will read data from both Timepix4 peripheries, top and bottom, but the user can change this setting using the corresponding method `setPeriphery(Periphery p)`.

In addiction to this, the length of the buffer that is used to save data into RAM memory before writing it to disk can be configured with the `setBufferingLen(int len)` method.

Lastly, if the user does not need to save data on disk[6], the `setWriteOnFile(bool writing)` method may be called. Once the object is configured, the user can start the acquisition.

[6] For example, if the hitmap saved by the online monitoring thread is enough for the user's needs. Or if the user needs to save only the clustered events (root file in output from online monitoring), discarding the raw data.

```
// Start slow read-out acquisition
slowThread->startSlowReadout();
```

When calling the `startSlowReadout()` method, a thread for each Timepix4 half-matrix[7] is created and starts continuously reading the corresponding read-out register. The choice of launching one thread for each half-matrix was made for optimization purposes: by using to the `readTpx4Repeat()` method instead of repeatedly calling `readTpx4()`, the overhead of a single function call is spread over multiple register's reads, instead of using one call for a single read. This, however, works only when reading the same register multiple times, and since there is one register for each half-matrix, then a read-out thread for each half must be created.

Once the acquisition is completed, the `stopSlowReadout()` function terminates the dedicated thread(s) and saves the last data on file.

```
// Stop slow read-out acquisition
slowThread->stopSlowReadout();
```

Since the read-out threads use the same methods exploited by the Timepix4 configuration functions, in the event of a changing in the control board, the user does not need to change the slow read-out class.

3.3.2 Fast Read-out

As mentioned in section 2.2, Timepix4 also has 16 high-speed links, 8 for each half-matrix, with a programmable bandwidth that can be set from 40 Mb/s to 10.24 Gb/s per link, for a total bandwidth of ~ 160 Gbit/s when using all links at maximum speed.

Timepix4 links are routed into the control board, that captures the raw data and sends to the computer UDP packets. The software counterpart of Timepix4 fast links connection, the `fastReadout` class, has been designed to use only UDP communication protocol and to be independent from the other classes, so it can be exploited, without any modification, with every Control Board[8].

UDP is a connection-less communication protocol, so there isn't a negotiated con-

[7] If the user decides to read both matrices, otherwise just one thread is instantiated, for the chosen half-matrix.

[8] As long as it supports UDP communication protocol for fast read-out

nection between the two parts and the packets are sent from the server to the client without any guarantee of delivery, ordering or duplicates [37]. Sending only data packets (no handshakes or acknowledgments, for example), the communication, however not always reliable, is fast and optimized, without unnecessary overheads or latency.

Since this is a one-way communication, meaning that the control board cannot receive data from fast links, all the communication parameters must be configured through slow control using dedicated methods. The Timepix4 registers that contain information about the fast read-out, for example how many links to use and their speed, can be set using methods from the `DAQ Control` virtual class described in section 3.2. On the other side, the control board configuration must be done through the `configFastLinks()` method, that should be implemented ad-hoc in the control board specific class, since every board may need to be configured in a different way. Information that needs to be set on the control board may include, for example, IP addresses and ports of the server.

Similarly to the `slowReadout` class, also the `fastReadout` has a constructor that needs a few fundamental parameters to create an object. The two main methods to start and stop the read-out are called `startFastReadout()` and `stopFastReadout()`.

An example, in the box below, instantiates a `fastReadout` object. The user needs to specify the IP addresses and the ports that will receive the data, in addition to the output data path and the same two flags of the slow read-out class, to specify whether to use online monitoring and/or clustering.

```
#include "fastReadout.h"

// Create two vectors with IPs and ports
std::vector ips<std::string> = {ip1, ip2};
std::vector ports<int> = {p1, p2};
// Set the online analysis flags
bool onlineMonitoring = true, onlineClustering = false;
fastReadout *fastThread = new fastReadout(ips, ports, outputPath,
                                          onlineMonitoring, onlineClustering);
```

The `fastReadout` object, as with the `slowReadout` one, has some *setter* and *getter* methods, in particular: `setTimeout_us(int timeout_us = 100)` sets the UDP socket timeout in μs, `setMaxSize(int maxSize = 1e9)` sets the maximum size (in

bytes) of the output file (when this size is reached, the file is closed and a new one is opened), `setVerbose(bool verbose = true)` makes read-out threads print how many packets are captured every second, `setWriteOnFile(bool write_on_file = true)` allows the dump of the captured raw data on file[9] and `setOutputPath(string outputPath)` sets the path in which store the output data[10].

Once the read-out is fully configured, the data acquisition can start.

```
1 // Start data acquisition
2 fastThread->startFastReadout();
```

The `startFastReadout()` method instantiates two threads for each (IP, port) pair. Those two threads uses a reader-writer paradigm on a shared circular array. In particular, one thread binds to the port and keeps listening on it, receiving the UDP data packets and writing them on the shared buffer. The other one reads the data from the buffer and store it on file. The shared buffer is not a single big data array, it is, instead, an array of pointers that point to smaller 9002 bits long char arrays. Each one of the smaller arrays will contain a UDP data packet (up to 9000 bytes with Jumbo Frames) and the last 2 bytes will contain the actual number of bytes stored (not all the data packets will be 9000 bytes long). A schematic of the shared buffer is presented in fig. 3.2.

This approach was chosen for a more optimized management of the server's memory. Moreover, since a thread interacts with the structure only to write or read a single small array, without modifying the big buffer, only 2 mutexes are necessary. The first one is for the index that keeps track of the element that is currently written and protects the small array while the writer is accessing it. The other one is complementary, to protect the small array that is currently read.

When the acquisition is finished, to correctly join all the threads, the stop function should be called.

```
1 // Stop data acquisition
2 fastThread->stopFastReadout();
```

The `fastReadout` threads accept every UDP packet that does not exceed the maxi-

[9] This method can be exploited when performing test acquisition (for example to align the setup) with online analysis on. In this way, the user can view the data in real-time without using disk space.

[10] This is useful to change path in between data acquisitions, without re-allocating a read-out object.

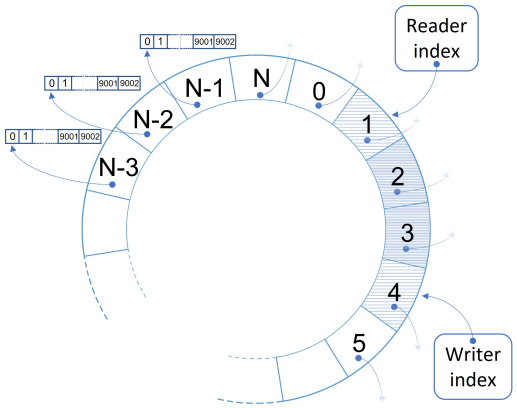


Figure 3.2: Schematic of the shared buffer between the reader and the writer read-out thread. Each element is a pointer to a 9002 elements array that will store a UDP packet. In white, the elements with no data, in blue the elements containing the data and in light blue the elements with data that is currently being read or written in the array.

mum size of 9000 bytes. This design makes the control board effectively transparent to the entire `fastReadout` class, enabling hardware changes without requiring modifications to those methods. Additionally, the computer used for fast data acquisition can differ from the one used for configuration. The `fastReadout` class can be exploited both for data-driven and frame-based data acquisition. However, the online clustering cannot be used in frame-based mode.

3.4. Real-time data visualization

When performing a data acquisition, having real-time insight into what is happening is essential, for monitoring performance and quickly identifying potential issues. For this reason, DataPix4 provides two tools for real-time data visualization: the first one, described more in detail in the next subsection 3.4.1, is an online plot displaying a 2D histogram of photon counting events, allowing the user to immediately observe spatial distributions and activity patterns. The second one, described in subsection 3.4.2, is a group of 4 plots that summarize various statistics about the clustered events, processed with the custom algorithm described in the same subsection.

3.4.1 Online data monitoring

During a data acquisition, the user often needs immediate feedback on how the data taking is progressing in order to monitor system performance and detect any irregularities. This is possible when using the online data monitoring tool, an independent thread that reads the data and displays a 2D histogram that highlights the pixels that have been hit, providing a real-time overview of detector activity.

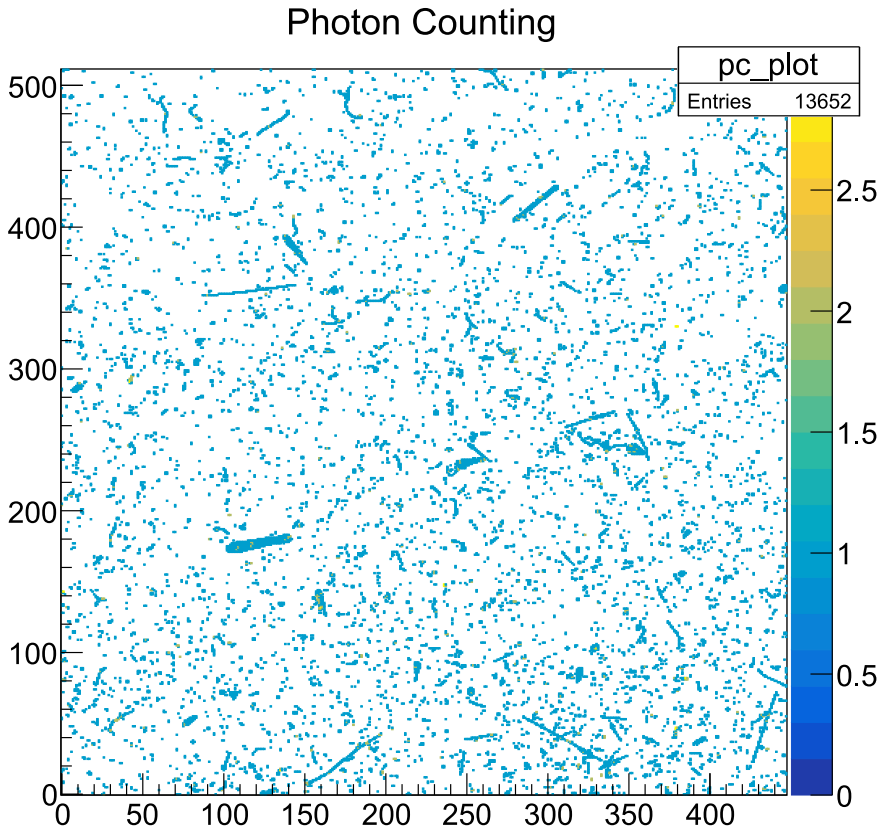


Figure 3.3: Online monitoring plot. This is an example from a data acquisition performed during a test-beam at CERN using a Timepix4 bump-bonded to a 300 μm Si sensor. In the plot, different tracks left by different particles are visible.

An example of a hitmap is shown in fig. 3.3. It is a 2D histogram with 448 bins on the X axis and 512 bins on the Y axis (the same as the Timepix4 pixel matrix). Every bin is incremented by 1 unit every time the corresponding Timepix4 pixel is hit, resulting

in a hitmap showing the distribution of the hits. A counter, in the top-right corner, shows how many events Timepix4 has detected. At the end of the acquisition, the plot is saved in `pdf` format.

If the `onlineClustering` flag is also set to `true`, a slightly different plot is displayed, showing also information about the clustered events. This will be described more in detail in the next subsection 3.4.2.

The user can use the online data monitoring both when acquiring through the slow control and the fast links. Both constructors, in fact, have the `onlineMonitoring` flag that can be set to `true`, as shown in subsections 3.3.1 and 3.3.2. In those cases, when the data acquisition starts, a new C++ `std::thread`, responsible for the online monitoring, is instantiated and launched.

Meanwhile, the read-out threads are started in a slightly different way, such that they will write data not only on disk, but also in a dedicated memory shared between them and the online monitoring thread. This thread is independent from the read-out and it simply reads the data packets from the shared memory, decodes them and updates the plot. The shared memory is protected with two `std::mutex` similarly to the buffer shared between the reader and the writer read-out threads.

The online monitoring thread is slower than the read-out, so sometimes it may not keep up with the rate. In this case, the read-out threads will not be paused to wait for the monitoring, they will, instead, overwrite the old data in the shared memory with new data. In this way, even if the online hitmap cannot display all the data, it is still correctly written on file.

If the user knows in advance that the rate will be high, it is possible to increase the dimension of the shared memory between the threads, by calling a dedicated low-level method. Also, it is possible to slightly modify the online monitor code to decode and plot only one out of every N events. The plot is update every second by default, but it can be changed according to the needs.

The plot is done using ROOT, a C++ Framework developed at CERN [40]. Using a `TApplication`, an object that creates a ROOT Application Environment, it is possible to display plots and updating them during the execution of a C++ program.

3.4.2 Online data analysis

When creating the read-out object, the user can set another flag in the constructor, namely the `onlineClustering` argument. When this flag is set to `true`, a dedicated C++ `std::thread` is instantiated and a shared memory between the read-out and the clustering threads is allocated. Using the same mechanism described for the online monitoring in subsection 3.4.1, the read-out threads write data both on disk and in the shared memory, where the online clustering thread can read the data.

The `onlineClustering` thread runs an online clustering custom algorithm (described more in detail in the following paragraph) that aggregates the hits based on time and space proximity, creating clusters. Each cluster is saved as an entry in a `TTree`, a ROOT data structure representing a columnar dataset [41]. A `TTree` has multiple `TBranches` [42], used to store information about each entry. In this case, the information that are saved into different `TBranches` are:

- `ClusterID`: *int*, a progressing number representing the ID of the cluster.
- `ClusterSize`: *int*, the number of pixels hits in the cluster.
- `XCenterOfGravity`: *float*, the weighted average of the X coordinates for all hits within the cluster.
- `YCenterOfGravity`: *float*, the weighted average of the Y coordinates for all hits within the cluster.
- `Charge`: *float*, the total charge of the cluster.
- `ToT`: *float*, the total Time-over-Threshold of the cluster.
- `ToACenterOfGravity`: *double*, the weighted average Time-of-Arrival for all hits in the cluster.
- `Xs`: *vector*, all X coordinates of the hits in the cluster.
- `Ys`: *vector*, all Y coordinates of the hits in the cluster.
- `ToAs`: *vector*, all the Time-of-Arrivals of the hits in the cluster.
- `ToTs`: *vector*, all the Time-over-Threshold of the hits in the cluster.
- `Charges`: *vector*, all charges of the hits in the cluster.

At the end of the data acquisition, the `TTree` is saved into a ROOT `TFile` [43].

If the `onlineMonitoring` flag is also set to `true`, some information regarding the clustered events are also displayed in real-time. A shared memory between the online clustering and monitoring threads is instantiated and the clustered data is sent from the clustering thread to the monitoring one, that displays it.

Although the four plots can be customized, for example if the user needs to monitor other parameters, the default is the visualization of:

- 2D events hitmap: the same hitmap displayed when the online clustering is not in use, showing a photon counting 2D histogram of the events (an example is shown in fig. 3.3).
- 1D ToT histogram: an histogram of the Time-over-Threshold of the clusters (calculated as the sum of the Time-over-Threshold of all the hits included in the cluster).
- 1D cluster size histogram: an histogram showing the distribution of the cluster dimension (calculated as the number of hits in the cluster).
- 1D charge histogram: an histogram of the total charge of the clusters (calculated as the sum of the charges of the hits). The charge is calculated starting from the ToT, using a non-linear equation described more in details in subsection 3.5.3.

An example of the online monitoring plots, when the clustering is also `true`, is shown in fig. 3.4. The plots are taken from a data acquisition performed with the 4DPHOTON detector during a test-beam at CERN, described more in detail in section 4.2. An example of how the plots can be changed, to fulfil the request from different users, is presented in subsection 4.4, where the online clustering thread has been connected to a pre-existing custom interface for online monitoring of the entire experimental system.

Regardless of the value of the other parameters, when the `onlineClustering` flag is set to `true`, a `TFile` containing the `TTree` with the clustered information is produced.

If the `onlineMonitoring` flag is also set to `true`, a PDF file, containing the four plots with information about the clusters, is also created, as mentioned in the previous subsection 3.4.1.

The clustering algorithm will be discussed more in detail in the next paragraph.

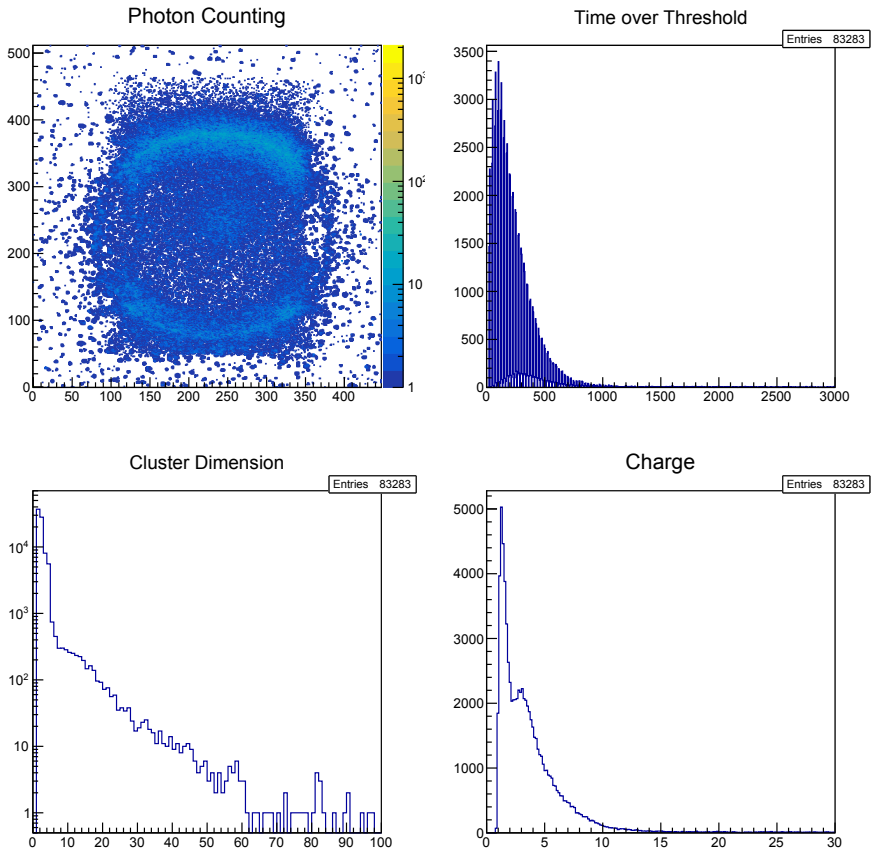
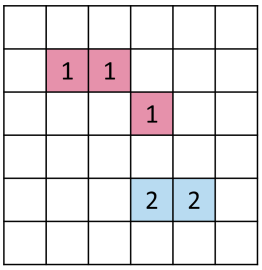


Figure 3.4: Online monitoring plot when clustering is on. This example was taken from an acquisition performed with the 4DPHOTON photodetector during a test-beam at CERN. On the upper left, a Cherenkov ring can be seen.

Clustering Algorithm

When an ionizing particle interacts with a sensor, it ionizes the atoms inside, leaving a cloud of electron-hole pairs. If an electric field is applied, those electrons (or holes, depending on the electric field) can drift within the sensor, accelerating and finally reach Timepix4 ASIC. Here, every electrons' cloud will hit one or more Timepix4 pixels, generating a data packet with all the information about timing and energy described in the previous sections. We say we have a *hit* every time an electrons' cloud is detected by a pixel. Defining an *event* as the entire interaction between an ionizing particle and the sensor at which Timepix4 is connected, we call *cluster* a group of hits that belong to the same event.



(a) Space proximity between hits. Only hits in adjacent pixels can be part of the same cluster.

(b) Time proximity between hits. Only hits close in time can be part of the same cluster.

Figure 3.5: Space and time proximity in clustering algorithm. Hits that can be grouped in the same cluster are close in space and time and are shown in figure with the same colour and number.

For example, if we look at the online monitoring plot in fig. 3.3, we can see different tracks. Each track is a cluster and it is made up of hits, one for each pixel. Each cluster corresponds, ideally, to a single physics event, namely a particle interacting with the detector.

The clustering algorithm designed for the online and offline Timepix4 data analysis is based on two assumptions: two hits that belong to the same cluster should be close in space and time. For what it concerns the space proximity, two hits are considered part of the same cluster if they hit adjacent pixels (in horizontal, vertical or diagonal)[11]. On the other side, they have time proximity if the difference between their Time-of-Arrival is not greater than a Δt_{hits} value, that can be set by the user and that defines the maximum time distance between two hits belonging to the same cluster. An illustration is shown in fig. 3.5.

Since the algorithm must run when the data acquisition is ongoing, it processes one hit at a time and takes decision based on the current hit and only the hits arrived before.

In particular, when a hit is processed, two fundamental information are calculated, its coordinates in a (X, Y) system and its Time-of-Arrival in nanoseconds. This last one is corrected exploiting the heartbeats, periodic control packets embedded in the data stream that carries a precise time-of-arrival reference for synchronization, the VCO

[11] The adjacent pixels are the default distance, but it can be changed exploiting a parameter that indicates the maximum distance between two hits (default is 1).

corrections, described more in detail in sub section 3.5.2, and the time-walk correction formula [44]. In addition to those, also the Time-over-Threshold and, accordingly, the charge is found and saved. Then, the searching for nearby hits begin, looking for hits in adjacent pixels with a timestamp closer than the Δt_{hits} chosen by the user.

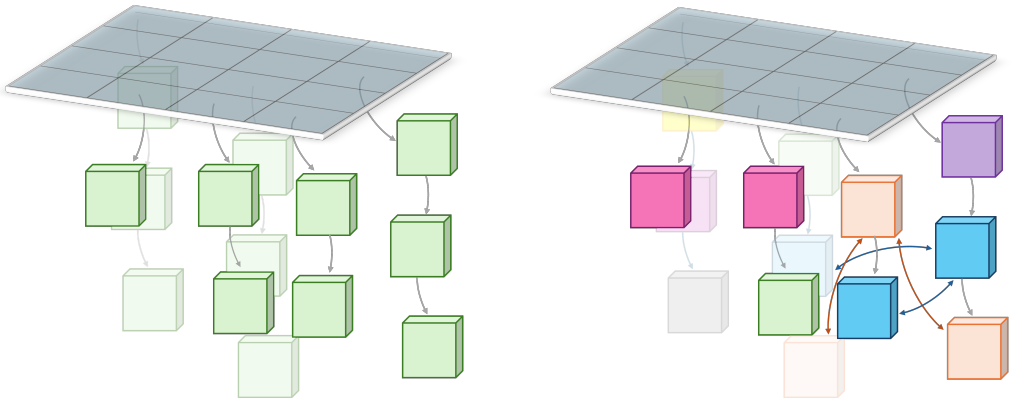
There are three possible scenarios:

- No compatible hits are found: a new cluster must be started.
- One or more compatible hits are found and they are all part of the same cluster: the current hit is added to this cluster.
- Two or more compatible hits are found and they belong to two or more different clusters: if the current hit could potentially belong to more than one clusters, it means that the two (or more) clusters are, actually, a single big one. So all the clusters found are merged into a single one and the current hit is added to it.

To store information about the hit that has been processed, I decided to use a 448×512 2D `std::vector` of pointers to a linked list. The linked list is made of multiple `struct` describing a hit, resulting in a 2D matrix with the same number of elements as Timepix4' pixels, each of which has a list of all the hits that has detected. An illustration is shown in fig. 3.6, where a simplified Timepix4 4x4 pixel matrix is represented in grey and a linked list for each pixel begins from the corresponding element. Fig. 3.6a represents the involved data structures omitting the clustering information. Every hit is shown as a cube, every pointer as an arrow. Fig. 3.6b represents the same data structures, but showing also information about clusters (hits in the same colour). When a cluster is found, the hits' pointers are connected between the hits (pointers of the same colours in the figure). For illustration purposes, only the pointers of two clusters have been drawn, but in the real situation all the pointers of all the clusters are present.

The `struct` containing information about the hit is defined as shown in the box below. It has some fields containing the fundamental information about the hit (its coordinates, ToA, ToT and charge), then it contains the pointers to previous and next element, to create a linked list.

To optimize the space (and the time) used by the algorithm, there is no data structure containing information about the clusters. All the hits belonging to the same cluster are connected exploiting pointers, creating a structure of graphs below the virtual Timepix4 pixel matrix, as illustrated in fig. 3.6.



(a) Timepix4 matrix with non-clustered hits.

(b) Timepix4 matrix with clustered hits (same colour and pointers[12]).

Figure 3.6: Illustration of Timepix4 pixel matrix (in grey) with a linked list of hit for each pixel.

In order to keep things simple, organized and optimized, there are not real fully connected graphs. For each cluster, one of the hits (usually the first one that arrives) is marked as 'master hit' and it will be the representative of its cluster. It will have a `std::vector` of pointers to all the other hits in its cluster and, on the opposite, all the other hits will have a pointer to their master. In this way, a tree is created and the algorithm can access, from every hit, all the other hits in the cluster following at most 2 pointers. All the cluster information (es. size, charge, weighted coordinates, ...) are calculated when the cluster is written on file, as will be described shortly.

To keep track of all the clusters, a `std::map<uint64_t, hit*>` is instantiated. Its keys are the clusters ID and the corresponding value is a pointer to the master hit of the cluster.

In the box below, the definition of the hit's `struct` can be found.

```
// Struct representing a hit
typedef struct hit {
    uint64_t ID;      // Unique identifier of the hit
    uint16_t X;       // X coordinate of the pixel hit
    uint16_t Y;       // Y coordinate of the pixel hit
    double timestamp; // ToA of the hit (corrected using heartbeat and timewalk)
    float  tot;       // Time-over-Threshold of the hit
```

[12]For illustration purposes, only the pointers of clusters blue and orange have been drawn, but, in reality, pointers between hits in every cluster are present.

```
8     float charge;    // Charge of the hit (calculated from ToT)
9     uint64_t datapacket;    // Original 64-bit data packet
10    uint64_t clusterID;     // Unique identifier for the cluster
11    hit *previous;  // Pointer to previous hit
12    hit *next;      // Pointer to next hit
13    hit *master;    // Pointer to master hit of the cluster
14    bool ismaster;  // Flag indicating whether this hit is master for the cluster
15    std::vector<hit*> other_hits;   // Vector of other hits in cluster (only for master hit)
16    bool overwrite; // Flag indicating if hit can be overwritten (already saved on file)
17 } _hit;
```

Since the data acquisition can be quite long, it is not useful to store in RAM information about hits and clusters that Timepix4 has detected long before the hit that is currently processed. Even if Timepix4 internal structure and communication protocols cannot guarantee that all the data packets arrive in order, we can estimate a maximum ΔT between the master hit of a cluster and the current processed hit that we can exploit to assert that the cluster will not receive any more hits. The ΔT is the second parameter that the user must choose. It is an upper bound, so the user does not need to be precise and can indicate an high value (for example different tens of seconds). The more precise this value is chosen, the more optimized the RAM space will be and the faster will be the neighbour search. The ΔT sets the maximum event reordering tolerated by the real-time clustering algorithm. The processed hits does not need to be strictly time-ordered, but hits in the same cluster may be interleaved only with hits whose time-of-arrival lies within $\pm\Delta T$ of the cluster time.

Every time a hit is processed, the first N[13] clusters from the cluster `std::map` are taken into consideration. Their master's timestamp is compared to the current hit's one and, if it is greater than the ΔT previously described, the cluster is marked as "to be closed".

When a cluster is closed, all its fundamental information are calculated (for example the total charge of the cluster, the weighted X and Y, ...). A complete list of the information can be found in the bullet list in the subsections 3.4.2. All the information are then written on disk[14], the entry is deleted from the cluster map and all the hit's

[13] By default, N is set to 10, but it can be changed according to need.

[14] If the `onlineMonitoring` flag is set to `true`, they are also written in the shared memory between the clustering and the monitoring thread, so the statistics can be displayed.

`struct` are marked as “over-writable” (by setting the corresponding flag to `true`).

When a new hit is processed, before allocating new space for a `struct`, the linked list starting from the element at the corresponding coordinates is scrolled and, if one of the hit is marked as “over-writable”, its information will be replaced with the current hit’s one, saving computational time.

The algorithm space complexity in the worst case scenario is $O(N)$, but, if the ΔT parameter is correctly tuned, the space complexity becomes $O(\frac{max\_rate}{\Delta T})$, where the max_rate constant is the maximum data rate per second detected by Timepix4.

The complexity in the best case only depends on the rate, not on the total number of elements, because old elements are written on disk and substituted with new ones, recycling the memory space. Once the memory is allocated, it is not de-allocated until the end of the algorithm, so the space in use is always proportional to the maximum number of hits that have a delta Time-of-Arrival less or equal to ΔT.

The algorithm time complexity in the worst-case scenario is $O(N^2)$, while can reach up to $O(N)$ in the best case. To better analyse the complexity, we can have a look at the pseudo-code shown in the box below.

```
// While the acquisition is ongoing
while (hit) {
    // Get or calculate X, Y, ToA, ToT and charge
    preprocess_hit(&hit);
    // Check adjacent pixels to find compatible clusters
    int cluster_found = scan_near_pixels(&hit);
    switch (cluster_found) {
        // No cluster found: start new cluster
        case NO_NEIGHBORS:
            create_new_cluster(&hit);
            break;
        // All hits found belong to one cluster: add hit to cluster
        case ONE_NEIGHBOR:
            add_to_cluster(&hit);
            break;
        // Hits found belong to two ore more clusters: merge those clusters and add hit to it
        case MORE_NEIGHBORS:
            fuse_clusters(&hit);
            break;
    }
```

```
21      // Before starting a new hit processing, check if
22      // some clusters can be written on file
23      int cluster_to_close = check_old_clusters();
24      if (cluster_to_close) {
25          for (auto cluster : cluster_to_close) {
26              // Write on file and free memory
27              write_on_file(cluster);
28          }
29      }
30  }
```

The while loop that encloses all the code will have N iterations, with N = number of hits. The `preprocess_hit(hit)` function has a $O(1)$ complexity, since it takes into consideration only one hit. In the same way, the `create_new_cluster(hit)` and the `add_to_cluster(hit)` will have a time complexity of $O(1)$.

The `fuse_clusters(hit)` function can take up to $O(N')$ time in the worst case scenario, where N' is the number of hits already processed[15].

The `scan_near_pixels(hit)` function, in the same way, can take up to $O(N')$ in the worst case scenario, for the same reason of the `fuse_clusters(hit)` one. The for-loop at the end will take at maximum $O(N')$, if all the clusters can be written on file.

In the end, the time complexity will be: $O(N*(1+N'+(1+1+N')+N') = O(N*N') = O(N^2)$. If we consider the average case, however, we could reach a time complexity of $O(N*k)$, where k is a constant that depends on the mean hit rate over time and on the ΔT value. This value is obtained considering a constant value k for the time spent in the `scan_near_pixels(hit)` functions. This is possible if the data acquisition has a quasi-constant rate with events equally distributed across Timepix4 pixel matrix. In this way, the algorithm needs to process a number of hits that remains approximately constant, since past events are written on file and deleted from memory.

Further improvement, using `std::map`[16] instead of linked list to store the hits, are foreseen, to reduce the worst-case time complexity from $O(N^2)$ to $O(n\log n)$.

[15] This can happen if all the hits belongs to clusters that must be merged together. In this case, all hits must be taken into consideration, changing, if needed, their pointer to the master hit (or, for the master, editing the other_hits pointer vector).

[16] Implemented as red-black tree, with logarithmic complexity for search, removal and insertion operations [45].

3.5. DataPix4 Graphical User Interface

The DataPix4 Graphical User Interface (DataPix4 GUI) is a Python-based graphical application developed to manage, configure and execute automated test procedures for the Timepix4 ASIC. The software provides a unified environment to launch acquisition or calibration routines through an intuitive user interface.

The main window is divided into two logical sections: a test selection and path management panel, and a configuration and terminal output panel. The layout is dynamically updated according to the selected test type, allowing the user to adjust only the parameters relevant to the chosen acquisition mode.

There are five possible routines and a reset that the user can execute:

- **Pixel Matrix Equalization**: described more in detail in subsection 3.5.1, it aims to find the best local DAC value for each pixel of the matrix, to mitigate individual variation in the pixel response.
- **VCO Calibration**: described more in detail in subsection 3.5.2, its goal is to study the differences in the oscillation periods of the Timepix4 Voltage-Controlled Oscillators, to allow post-processing correction of the events ToA.
- **ToT Calibration**: described more in detail in subsection 3.5.3, it performs a non-linear fit for each pixel, finding the best parameters to convert the ToT of the event into the deposited charge value.
- **Data-Driven Acquisition**: described more in detail in subsection 3.5.4, it runs a data acquisition in ToA-ToT data-driven mode, with the parameters specified by the user.
- **Frame-Based Acquisition**: described more in detail in subsection 3.5.5, it runs a data acquisition in 8-bit or 16-bit frame-based mode, with the parameters specified by the user.
- **Timepix4 Reset**: it launches a reset of all Timepix4 registers, setting all of them to their default value.

When the user chooses the routine through the drop-down menu, a configuration frame appears, with parameters necessary for the selected routine. Some examples are

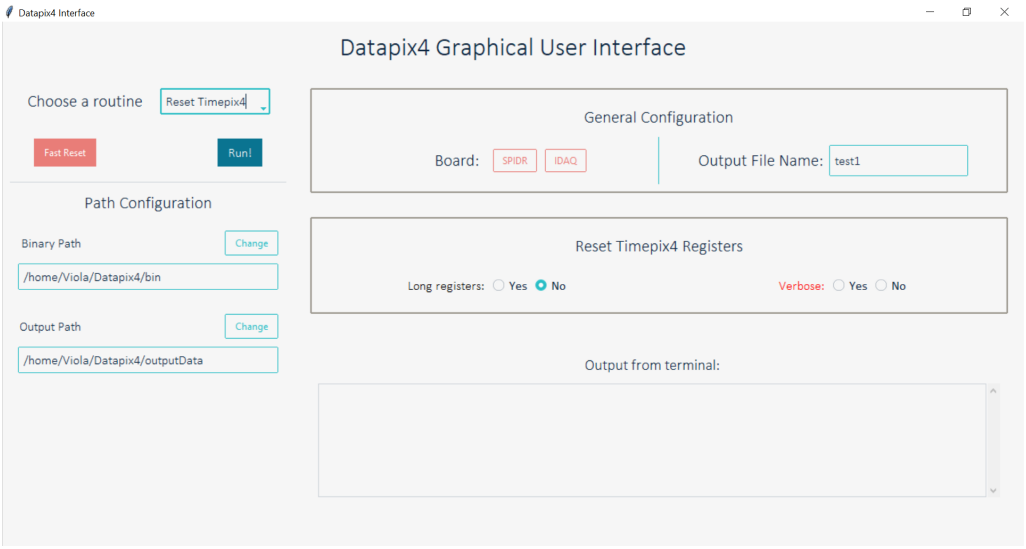


Figure 3.7: DataPix4 GUI raises errors when the user tries to run routines without setting the parameters.

shown in fig. 3.9, 3.11, 3.12, 3.14 and 3.15. When the parameters are correctly set, the user should click the `Run!` button to start the routine. If something goes wrong, the `Fast Reset` button can be exploited, resetting Timepix4 to its idle state. A field-check on the parameters is automatically performed before launching the routine, raising an error if some values are missing. Fig. 3.7 shows the GUI with two errors (the red fields): the board is not selected and the verbosity is not set.

The DataPix4 GUI was designed to be fully cross-platform and independent of the underlying operating system. In fact, it relies only on standard Python libraries (namely `Tkinter` [46] and `ttkbootstrap` [47]), so it should run on any system that supports these dependencies. The graphical interface was successfully tested on Ubuntu 20.04, Ubuntu 24.04 and Windows 10, but it is expected to run on any modern operating system with a compatible Python environment. While the GUI itself is platform-agnostic, the underlying DataPix4 acquisition software may be subject to additional restrictions, as its communication layer depends on the specific control board used. To mitigate these limitations, the GUI can also be deployed on a remote system and configured to execute the DataPix4 scripts via SSH or other network-based methods, allowing users to control the Timepix4 system from any computer, regardless of local system compatibility.

3.5.1 Timepix4 Matrix Equalization and Noisy Pixels

The equalization routine for the Timepix4 detector is a preliminary setup procedure designed to optimize the detection efficiency across the entire sensor matrix by adjusting threshold settings. The main objective of equalization is to mitigate individual variations in pixel characteristics and local noise levels. In addition to this, the procedures also search for noisy pixels, i.e. pixels whose front-end electronics exhibit an abnormally high rate of spontaneous triggers unrelated to genuine charge deposition in the sensor, resulting in a great number of unreal events during data taking.

Because of intrinsic non-uniformities in the ASIC fabrication and design, individual pixels in the Timepix4 matrix can exhibit slightly different electronic characteristics. In particular, the gain and threshold levels of each pixel may vary, leading to non-uniform response across the matrix.

To compensate for these variations, each pixel in Timepix4 is equipped with a dedicated 8-bit configuration register. Within this register, 5 bits are used to control a local DAC that fine-tunes the discriminator threshold. With the pixel-by-pixel equalization, those 5 DAC setting bits are adjusted to align the pixels effective thresholds and gains to a common reference.

The equalization process aims to identify a common baseline threshold across the matrix. Ideally, when the global threshold is set to zero, all pixels should exhibit their maximum noise rate, corresponding to the condition where the effective discriminator threshold is approximately 0 e^-. In practice, however, the actual baseline differs slightly from pixel to pixel.

To determine the true baseline for each pixel, the value of the local DAC (used to fine-tune the threshold) is scanned over its full range while the global threshold remains fixed at 0 e^-. For each DAC setting, the number of detected noise events is recorded for every pixel. The noise rate as a function of the local DAC code typically exhibits a peak corresponding to the effective zero threshold of that pixel. The position of this maximum identifies the local DAC shift required to align the pixel's individual baseline with the global reference. A 2D plot showing the local DAC shift for each Timepix4 pixel is shown in fig. 3.8.

This procedure provides, for each pixel, the offset between the global zero level and the pixel's intrinsic zero threshold. Applying the corresponding local DAC correction

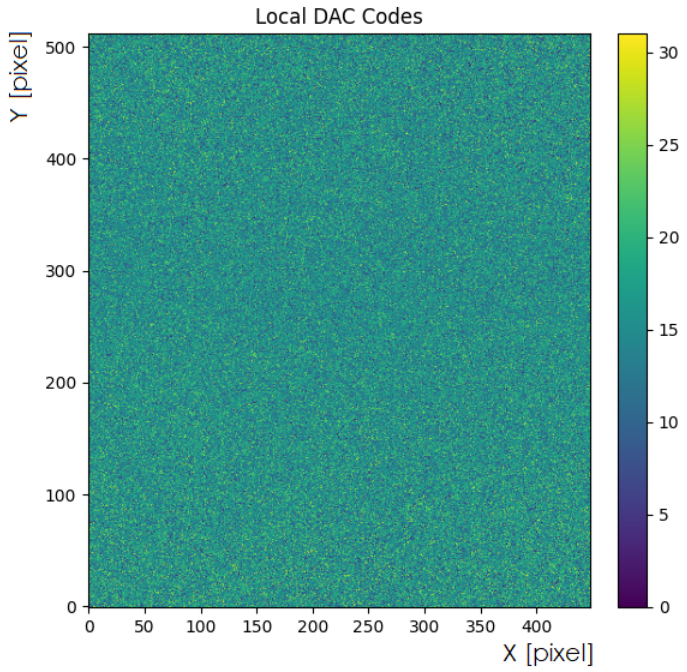


Figure 3.8: 2D plot of the local DAC shifts found using the equalization procedure.

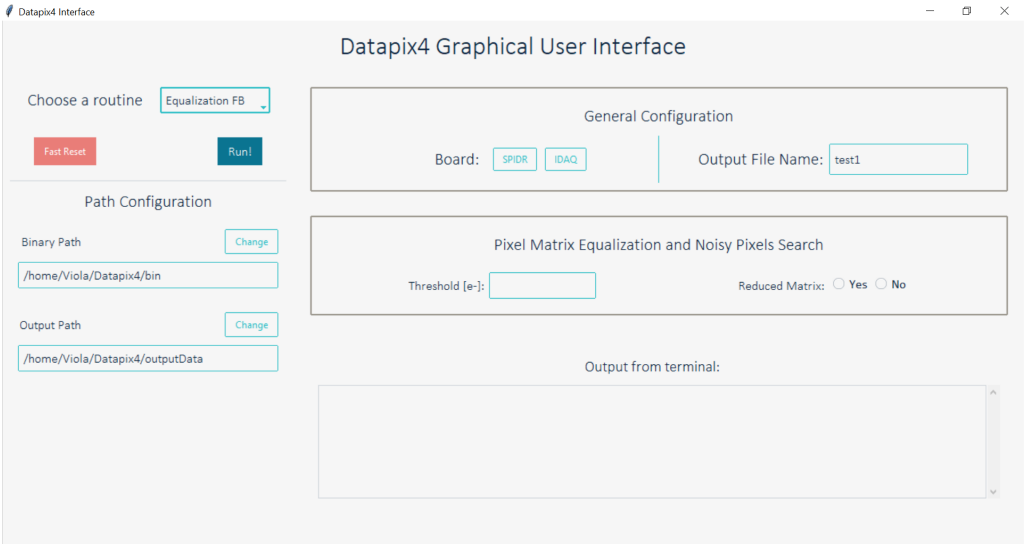


Figure 3.9: DataPix4 GUI when running the equalization procedure.

equalizes the noise response across the matrix, ensuring that all pixels operate at almost the same effective threshold.

Because this equalization method in data-driven mode would produce an unnecessary large amount of data, the acquisition is performed in frame-based read-out mode, keeping the shutter open for only 200 μs.

At the end of the equalization procedures, two different files are created, one containing the local DAC values of the pixel and one with a mask for the noisy pixels.

Fig. 3.9 shows the interface when the equalization procedure is selected. The user can select the threshold (in equivalent electrons) to optimise the search for noisy pixels. The reduced matrix parameters allow the user to perform an equalization on a smaller region of the pixel matrix, if they want to perform data acquisition enabling just the central region of the ASIC.

3.5.2 Voltage-Controlled Oscillators Calibration

In the Timepix4 ASIC, the Voltage-Controlled Oscillators (VCOs) are the key elements of the time-to-digital conversion (TDC) system used to measure the Time-of-Arrival (ToA) and Time-over-Threshold (ToT) of signals at the pixel level.

As explained more in detail in section 2.2.1, an events timestamp is obtained in three steps: the coarse ToA from the 40 MHz global clock (25 ns bins), the fine ToA from a 640 MHz VCO (1.56 ns bins) and the ultra-fine ToA from four internal VCO phases, yielding a final time resolution of 195 ps.

It has been observed that the different VCOs are oscillating at slightly different frequencies [28] [48]. If the frequency is not properly corrected, both the ufToA and fToA bin widths will be inaccurately estimated.

The calibration of the Voltage-Controlled Oscillators (VCOs) in Timepix4 is performed by injecting a series of test pulses at different controlled phases of the global clock. Specifically, the test pulse is triggered at 16 distinct phase values uniformly distributed between 0 and 25 ns. For each injected phase, the Time-of-Arrival (ToA) of the recorded events is measured, providing a set of points that describe the linear relationship between the ultra fine time counter ($ufToA_{cycle}$) and the test pulse phase (ϕ). The calibration data are then fitted with a linear function: $ufToA_{cycle} = p_0 + p_1 \cdot \phi$, where p_0 represents the offset and p_1 the slope of the fit. The slope parameter p_1, expressed

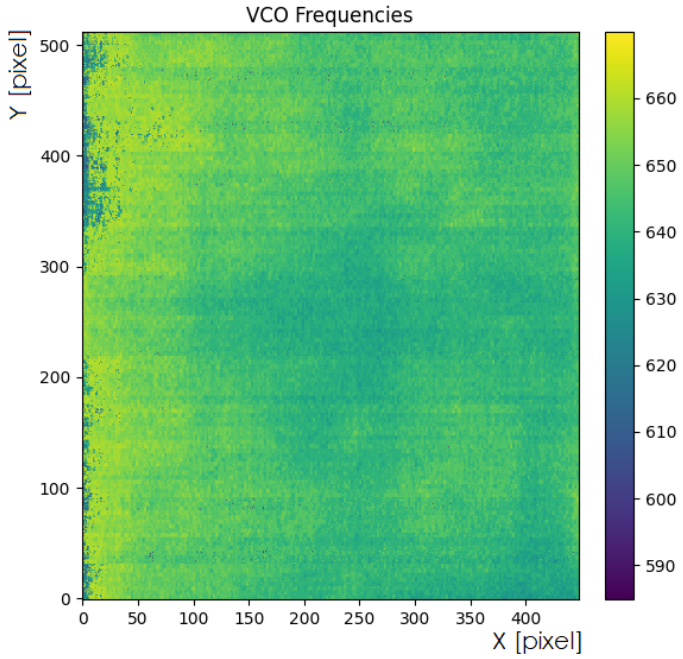


Figure 3.10: 2D distribution of the VCO frequency across Timepix4 matrix.

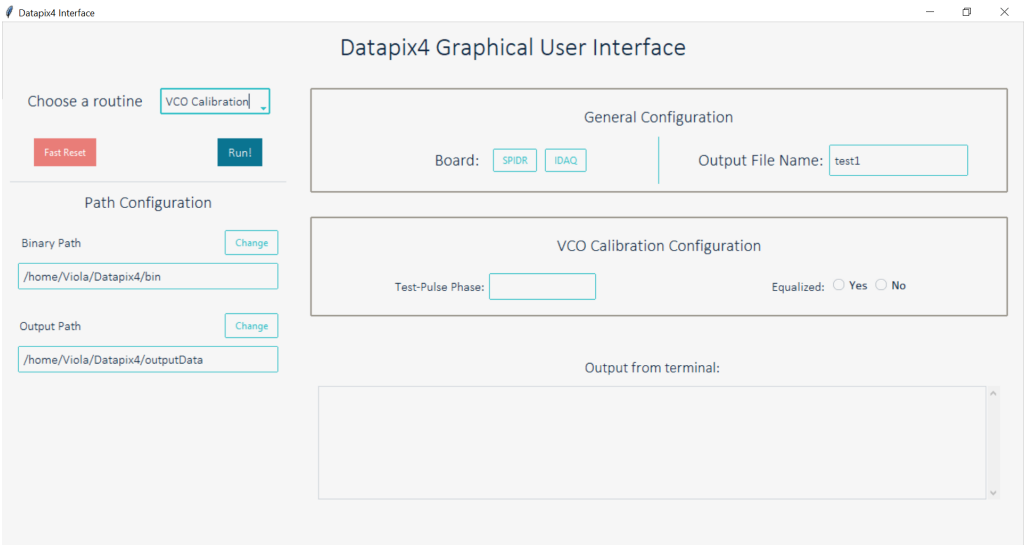


Figure 3.11: DataPix4 GUI when running the VCO calibration procedure.

in Hz, is directly related to the VCO oscillation frequency ν_{VCO} through the relation $\nu_{VCO} = p_1/8$, where the coefficient 8 corresponds to the ratio between the width of the fine ToA (fToA) bins and the ultra-fine ToA (ufToA) bins.

The VCO calibration procedure returns a file with the ν_{VCO} value for each pixel in Timepix4 matrix, that can be loaded during the data analysis, correcting the ToA of the event. A 2D plot displaying an example of the distribution of the VCO frequency is shown in fig. 3.10.

Fig. 3.11 shows the interface when the VCO calibration routine is selected. The user may choose the number of test-pulse phases to scan and should select if the pixel matrix of the chip has already been equalized (in this case, the equalization file, containing the DAC values, is loaded and exploited during calibration).

3.5.3 Time-over-Threshold Calibration

The Timepix4 ASIC provides, for each detected event, the measurement of the Time-over-Threshold (ToT). Since this time interval is related to the amplitude of the input signal, the ToT value can be used to estimate the charge deposited in the pixel. The relationship between the deposited charge Q and the measured ToT is described by the function $Q(TOT) = p_0 + p_1 \cdot ToT - \frac{p_2}{ToT - p_3}$, parametrized by four coefficients p_0, p_1, p_2, p_3 that account for the specific response of each pixel [44].

Because of intrinsic variations in the front-end electronics, these parameters differ slightly from pixel to pixel, making a per-pixel calibration necessary. The calibration is performed using the internal test-pulse injection circuit. This circuit injects a signal with a known and selectable amplitude, allowing the controlled delivery of a defined charge. By varying the test-pulse amplitude, a set of signals with different charges is generated for each pixel. For each injected pulse, the corresponding ToT value is measured and stored. The resulting ToT-charge data points are then fitted, independently for each pixel, using the calibration function. The four parameters are saved into four dedicated files. These can then be loaded during data analysis to convert the raw ToT measurements into deposited charge values.

Fig. 3.12 shows the interface when the ToT calibration procedure is selected. The user, among the other parameters, can choose which fit to perform. For a full calibration, all the fits should be performed. The fit described in the paragraph above is the

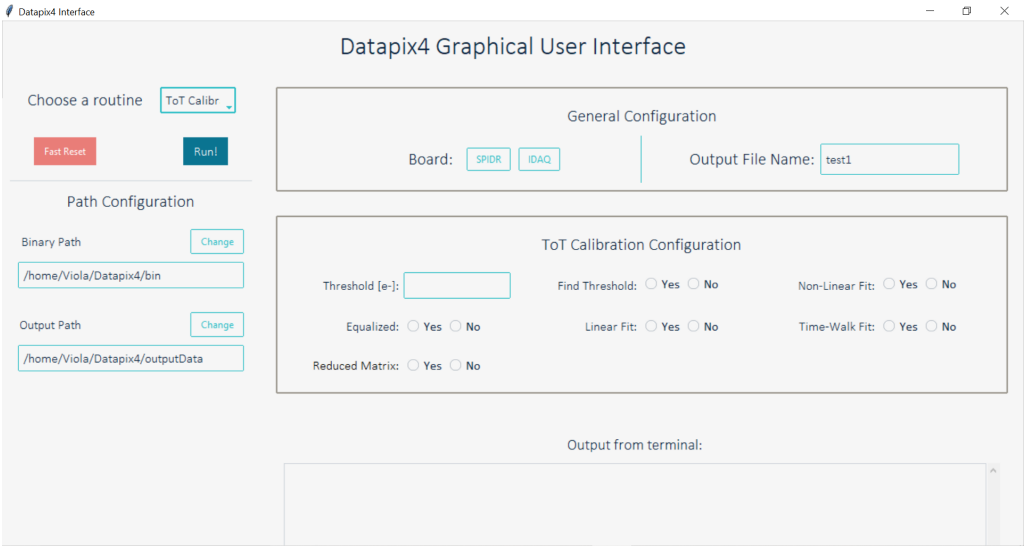
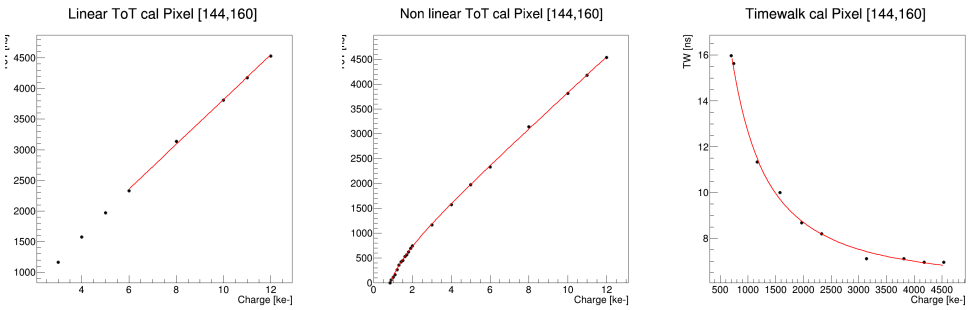


Figure 3.12: DataPix4 GUI when running the ToT calibration procedure.



(a) Example of a linear ToT-charge fit performed on one pixel.

(b) Example of a non-linear ToT-charge fit performed on one pixel.

(c) Example of a time-walk fit performed on one pixel.

Figure 3.13: Different fits performed by the ToT calibration procedure.

non-linear fit, but, in some particular cases, the user could need also a linear fit between ToT and charge. Lastly, the time-walk fit procedure performs a pixel-by-pixel calibration that extrapolates the curve parameters according to the following equation [49]: $TW(Q) = \frac{p_0}{Q^{p_1}+p_2} + p_3$.

An example of the three fits is displayed in fig. 3.13.

3.5.4 Data-Driven ToA-ToT Acquisition

In data-driven acquisition mode, Timepix4 operates with a zero-suppressed, event-by-event read-out. This design ensures that the it only transmits data packets when a specific pixel registers a signal that exceeds a predefined energy threshold, thereby reducing the overall data volume compared to the frame-based approach in low occupancy applications such as particle tracking.

In the Time-of-Arrival and Time-over-Threshold (ToA-ToT) mode, each detected event triggers the generation of a 64-bit packet containing detailed temporal and spatial information, including pixel address, coarse and fine Time-of-Arrival (ToA), ultra-fine ToA start and stop codes, coarse Time-over-Threshold (ToT) and a pile-up bit. A more exhaustive explanation of the ToA-ToT data-driven modality is discussed in the corresponding paragraph of subsection 2.2.2, while a representation of the 64-bit packet in this modality is shown in tab. 2.1.

Since this modality provides precise timing and charge information, it is ideal for single-particle or time-resolved measurements.

Fig. 3.14 shows the interface when the data-driven acquisition is selected. The user can configure the global threshold (in e^-), the time (in seconds), of each data taking and the number of total runs. Additionally, the user can choose whether or not loading the equalization files (created in the homonymous routine) and using the test-pulse.

3.5.5 Frame-Based Acquisition

In the frame-based acquisition mode, Timepix4 performs a non-zero-suppressed, synchronous read-out of the entire pixel matrix at fixed time intervals, known as frames. Each pixel counts the number of events that exceed the predefined threshold over fixed, periodic intervals known as *frames*.

Instead of generating event-by-event packets, the system performs a simultaneous read-out of all pixels at the conclusion of each frame.

Two frame depths are available: 8-bit and 16-bit. In the 8-bit mode, the full 16-bit physical counter of each pixel is partitioned into two smaller, alternating 8-bit registers. This implementation enables continuous read/write operation without data loss, as one 8-bit counter can be actively recording new events while the data from the other 8-bit counter is being simultaneously read out. In the 16-bit mode, instead, the full counter depth is used. While this mode offers a significantly higher dynamic range for counting events in a single frame, the entire counter must be read out before the next frame can begin recording, which may introduce a small gap or dead time between acquisitions.

Frame-based operation is particularly suited for high-rate photon counting applications and it is described more in detail in subsection 4.2.3. Tables with information about the data packet in 8- and 16-bits frame-based modality are shown respectively in 2.4 and 2.5.

Fig. 3.15 shows the interface when the frame-based acquisition is selected. The user can configure, in the same way as the data-driven acquisition, the global threshold (in e^-) and the time (in seconds) of each data taking, the loading of the equalization files and the test-pulse. Additionally, the user can set the maximum number of frame and the read mode (8-bit or 16-bit).

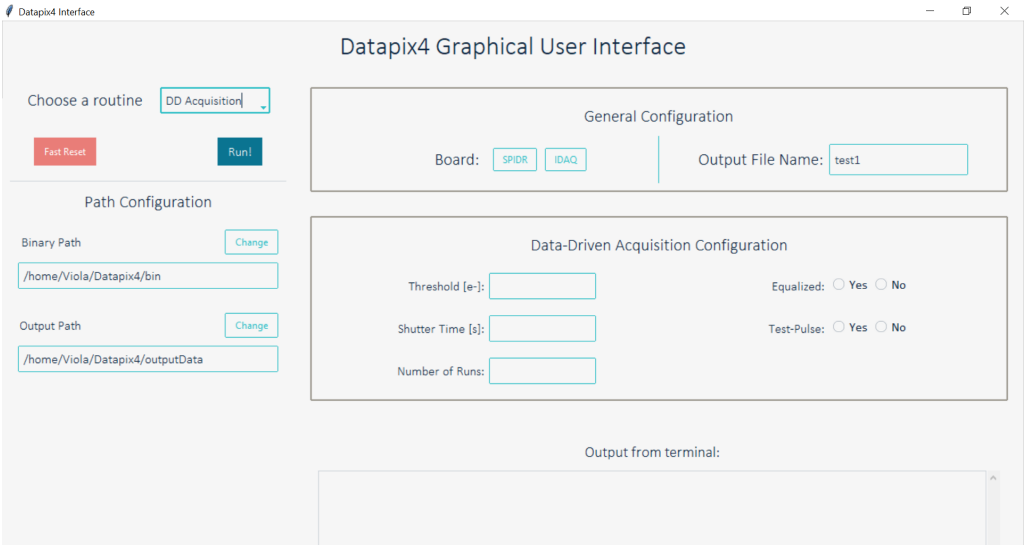


Figure 3.14: DataPix4 GUI when running a data-driven data acquisition.

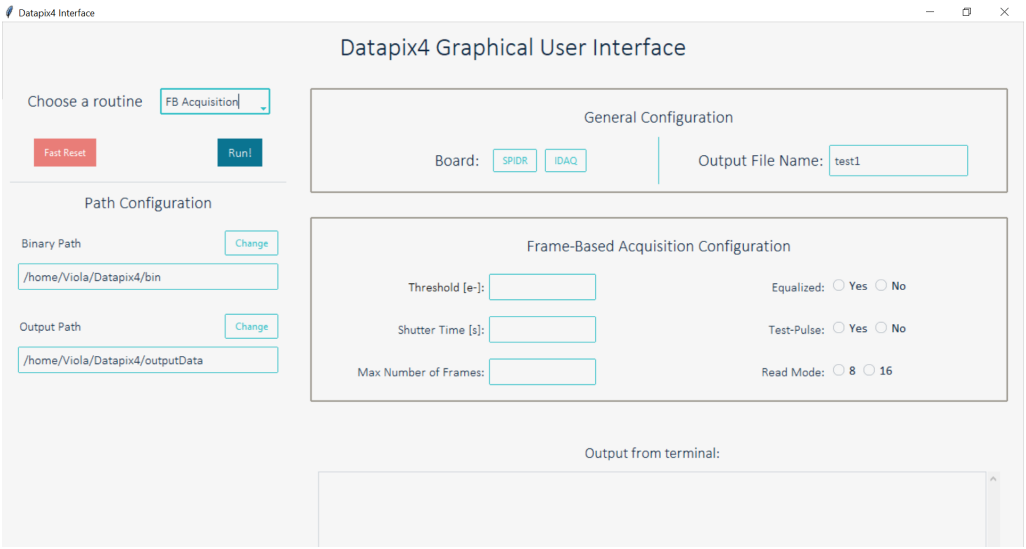


Figure 3.15: DataPix4 GUI when running a frame-based data acquisition.

CHAPTER 4

DataPix4: measurements and applications

DataPix4 has been exploited in various experiments of different kinds, that will be described in this chapter, thanks to the combination of two key factors. First, Timepix4's multi-purpose nature, which allows it to detect different types of particles depending on the sensor it is bump-bonded to. Second, the software's flexibility, which makes it possible to adapt it to the specific hardware setup in use.

In this chapter, some examples of DataPix4 applications will be given. Timepix4's characterizations measurements, described in section 4.1, were performed to study its spatial, temporal and energy resolution. Measures were taken at the University of Ferrara, at INFN Ferrara and during a beam-time at Elettra Synchrotron in Trieste. More details will be given in subsections 4.1.1, 4.1.2 and 4.1.3.

A test-beam campaign at CERN to test the first 4DPHOTON detector prototype will be presented in section 4.2.

Lastly, an example of DataPix4 integration into an existing framework for imaging application will be described in section 4.3.2.

4.1. Timepix4 characterization

The comprehensive characterization of Timepix4 is crucial for accurately assessing its performance across a wide array of demanding applications, including high-energy physics experiments, nuclear physics, medical and biology applications. To understand and optimize Timepix4's capabilities, extensive measurements and calibration procedures have been undertaken [50] and will be described more in detail in the next subsections. In particular:

- **Timing Resolution Performance**: Detailed characterizations of Timepix4 assemblies have been performed, using a 100 μm thick n-on-p silicon sensor and illuminating the detector with picosecond pulsed infrared lasers. These measurements have demonstrated a single pixel timing resolution of 107 ± 3 ps r.m.s. for signals above 50 ke^-. This resolution can be improved to 33 ± 3 ps r.m.s. when considering multi-pixel clusters by exploiting oversampling of timing information and a weighted average of individual pixel ToA values. Those measurements will be described in subsection 4.1.1.

- **Energy Resolution and Calibration**: Evaluations of Timepix4's energy resolution and calibration have been performed using monochromatic X-rays at the University of Ferrara and at the Elettra synchrotron facilities in Trieste, Italy. A crucial aspect is the pixel-by-pixel energy calibration to convert the Time-over-Threshold (ToT) signal into deposited energy. An improved calibration method merges data from monochromatic X-rays with internally generated test pulses, extending the calibration range and enhancing energy resolution, particularly at lower energies where non-linearity is observed. This combined approach yields an energy resolution better than 1 keV and achieves 6.4% at 40 MHz, with accuracy better than 1.2% in energy reconstruction. Those measurements will be described in subsections 4.1.2 and 4.1.3.

All the measurements described in this thesis have been carried out exploiting the DataPix4 framework, contributing, at the same time, to the software debugging and improving.

4.1.1 Timepix4 timing measurements

Extensive measurements and calibration procedures have been conducted at the University of Ferrara to thoroughly characterize Timepix4 timing performance, which is crucial for applications such as the 4DPHOTON project, aiming for single visible photon detection with excellent timing and spatial resolution (described in detail in chapter 2).

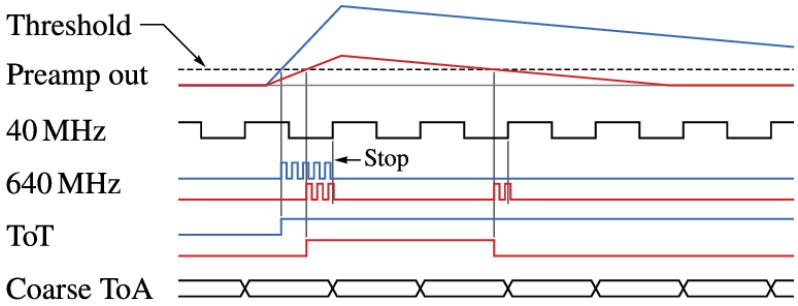
As described in subsection 2.2.1, Timepix4 employs a sophisticated time-to-digital conversion (TDC) process that occurs in three main steps (a schematic of the clocks is shown in fig. 4.1).

- **Coarse Time-of-Arrival (ToA)**: The initial step determines the 40 MHz clock cycle in which the preamplifier output signal crosses the set threshold. This provides a coarse time measurement with a bin width of 25 ns.

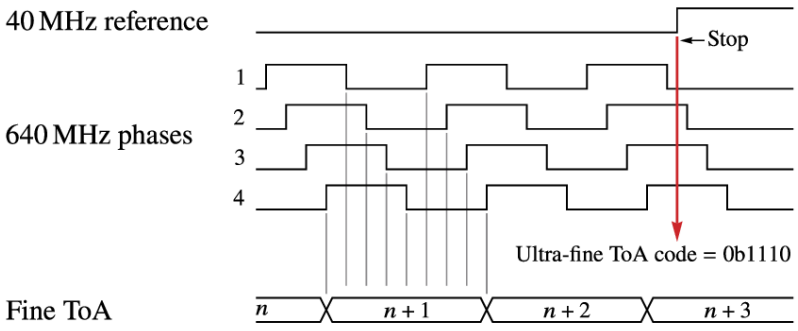
- **Fine Time-of-Arrival (fToA)**: When a signal exceeds the selected threshold, a Voltage-Controlled Oscillator (VCO) within the SuperPixel (SP) group is activated, oscillating at a nominal frequency of 640 MHz.
  The number of 640 MHz clock cycles counted until the next 40 MHz clock rising edge refines the ToA measurement, improving the bin width to 1.56 ns. These first two measurements are shown in fig. 4.1a.

- **Ultra-fine Time-of-Arrival (ufToA)**: Further precision is achieved by latching 4 internal phases of the 640 MHz VCO, which subdivides the 1.56 ns bins into eight smaller intervals, each with a nominal width of 195 ps. This corresponds to a nominal r.m.s. resolution of 56 ps. This measurement is shown in fig. 4.1b.

To characterize its timing performance, a Timepix4 bump-bonded to a 100 μm thick n-on-p silicon sensor and the SPIDR4 read-out system were used. The setup involves generating synchronized signals: a pulse generator (Active Technologies PG-1072) with a low inter-channel jitter ($\sim$ 7 ps r.m.s.) produces two synchronized pulses. One pulse is sent directly to a Timepix4 digital pixel to serve as a reference signal (ToAref). The second pulse triggers a picosecond pulsed diode laser (PicoQuant PDL-800 B with LDH-P-1060 or LDH-P-C-405 laser head), generating an infrared or blue/UV laser pulse with a short duration (36 ps FWHM for infrared, 56 ps FWHM for 405 nm) and low jitter ($<$ 20 ps r.m.s.). The laser light is attenuated, collimated and focused onto



(a) Coarse ToA, fToA and ToT measurements are obtained with a 40 MHz and a 640 MHz clocks.



(b) UfToA measurements is obtained by four copies of a 640 MHz clock shifted.

Figure 4.1: Schematic of the three steps to measure Time-of-Arrival [51]

the sensor's back-side, illuminating a small region (a Gaussian shape with a standard deviation of about 1.4 pixels). The overall jitter contribution from the laser setup and pulse generator has been measured to be negligible, at around (10.9 ± 0.1) ps r.m.s. A picture of a part of the setup (Timepix4 and the laser) is shown in fig. 4.2. Here, the motion setup can be seen (three Zaber mover to change the laser position in a precise and remote way. The laser collimator can be seen as well and, below it, a chipboard mounts the Timepix4 used for the characterization.

Measurements are performed in data-driven mode with a threshold of 1000 e^- after pixel equalization (described more in detail in subsection 3.5.1).

The overall timing resolution is affected by several factors, some intrinsic and others correctable. The first correction is about the Voltage Controlled Oscillator (VCO) frequency, since it was observed that different VCOs across the Timepix4 matrix oscillate at varying frequencies. An automated calibration procedure has been studied and

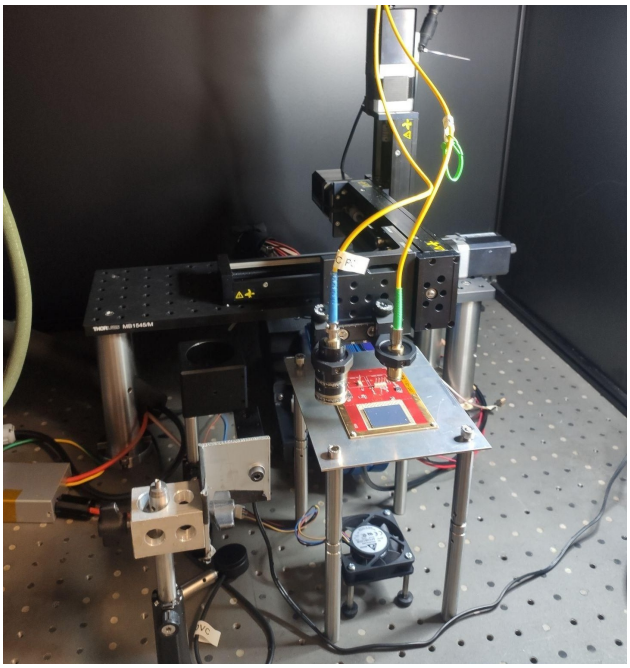


Figure 4.2: Laser setup in use at the University of Ferrara for the timing measurements with Timepix4. The chipboard, with Timepix4, is mounted on a custom support. The laser position can be remotely adjusted exploiting the three movers.

implemented[1], exploiting internal digital test-pulses, synchronized with the 40 MHz 25 ns reference clock, that are sent to one pixel per TDC. By varying the phase of these test-pulses relative to the reference clock, the VCO frequency for each TDC is determined. A map of VCO frequencies revealed a distribution peaked at 640 MHz but spanning a range of approximately 40 MHz [48].

After correcting for VCO frequency variations at the analysis level, the timing resolution for the difference between a central pixel and the reference signal significantly improves to (126 ± 1) ps r.m.s. [48].

The second factor to take into account is the "time walk" effect, where the signal arrival time depends on the deposited charge. Measurements are performed by finely varying the laser attenuation, in order to cover a continuous-like charge range ([$\sim$ 10 ke^-;200 ke^-]). The time difference is plotted against the charge (ToT) to reveal the time walk spectrum. This distribution is fitted with a function like $TW_{ToT} = p_0/Q^{p_1} + p_2 + p_3$ to model the charge-dependent delay.

[1] Now it's integrates into DataPix4 Framework. A detailed description can be found in subsection 3.5.2

Finally, combining information from multiple pixels in a cluster can significantly enhance timing precision. For clusters with more than one pixel, each pixel's ToA is corrected for time walk. Then, a "cluster ToA" is calculated as a weighted average of individual pixel ToA values, using the corresponding deposited charge (ToT) as the weight. An analysis on the cluster charge range shows that the cluster timing resolution reaches a minimum of (79 ± 1) ps r.m.s. for cluster charges between approximately 900 ke^- and 1300 ke^-. After subtracting the reference signal's contribution in quadrature, the true cluster timing resolution reaches (33 ± 3) ps r.m.s. [48].

All those fundamental measurements led to important changing and improvement in the DataPix4 software, debugging the existing API and routines, writing new optimized ones and testing the whole framework while taking data.

The measurement methods and results were published in [44].

4.1.2 Timepix4 energy calibration at INFN Ferrara

Another set of measurements were carried out at the Larix-A facility, an underground laboratory located at the University of Ferrara, Italy. These tests were part of a preliminary characterization aiming to study the detector's response to X-rays and gamma radiation, specifically to evaluate its energy calibration and energy resolution [52]. While those measurements are usually carried out in synchrotrons or similar systems, using an intense monochromatic X-ray source, having an equivalent system inside a laboratory could improve accessibility and cut costs, so it is of significant interest.

The setup featured a 500 μm thick Silicon sensor bump-bonded to a Timepix4 ASIC. Timepix4 was mounted on a custom chipboard from NIKHEF and connected to the SPIDR4 read-out system. The read-out bandwidth employed two fast links operating at 2.56 Gbit/s. DataPix4 was used for configuration, data read-out and data analysis.

The characterization proceeded in two main phases. Initial tests were performed using radioactive sources such as ^{55}Fe, ^{241}Am and ^{137}Cs. The radioactive sources served to validate the detector's internal test-pulse calibration by comparing it against emissions with known radioactive decays.

Subsequently, an X-ray tube with a Mo-anode and suitable Aluminium (Al) filtration, along with a tunable quasi-monochromatic irradiation setup, were employed. The system was developed at the University of Ferrara for archaeometry and medical

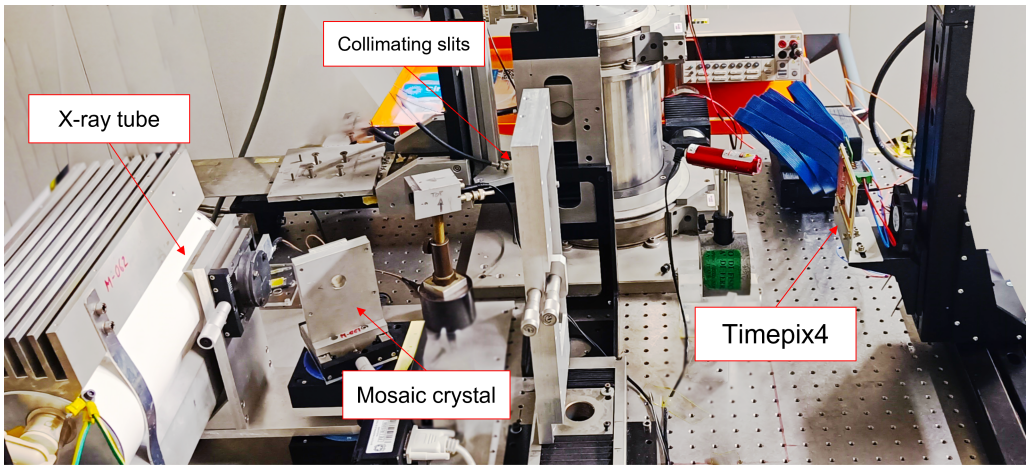


Figure 4.3: Setup at Larix facility in Ferrara, Italy. On the left, the X-ray tube and the crystals (on the movers) can be seen. On the right, Timepix4 on NIKHEF chipboard is mounted on a custom support.

applications [53].

This system is based on Bragg diffraction from a Highly Oriented Pyrolytic Graphite (HOPG) mosaic crystal. The X-ray beam, at the exit of the X-ray tube, is collimated by three lead slits, each 2 mm wide. Then the mosaic crystal, measuring 60×28 mm^2 with a 1 mm thickness, is positioned such that adjusting the angle between the X-ray source and the crystal allows for tuning the beam's energy according to Bragg's law [2]. Both the X-ray tube and the mosaic crystal are mounted on motorized goniometers. The setup is shown in fig. 4.3.

Six distinct energy values, from 10 keV to 35 keV in 5 keV steps, were studied using the quasi-monochromatic system. Both full-matrix and individual-pixel energy spectra were acquired and Gaussian fits were performed on the energy peaks to determine mean energy values and standard deviations.

Results were published in [52].

Following the characterization of the silicon–Timepix4 assembly, an experimental study of a 500 μm gallium arsenide (GaAs) sensor bump-bonded to a Timepix4 detector has been carried out. GaAs sensors provide significantly higher absorption efficiency than silicon in the 20–50 keV X-ray energy range due to their higher atomic number and density. Compared to high-Z materials, GaAs also exhibits reduced fluorescence

[2] Bragg's law explains how the angle of incidence θ, the wavelength λ and the inter-planar spacing of the crystal d are connected. The resulting formula is: $n\lambda = 2d\sin\theta$, where n is the diffraction order.

yield, resulting in improved image contrast.

The Timepix4-GaAs assembly was characterized using the same read-out system from NIKHEF and the DataPix4 software, operating in both frame-based and data-driven acquisition modes. Frame-based measurements, performed under direct X-ray irradiation, were used to evaluate flat-field stability and spatial resolution. Data-driven measurements, on the opposite, were carried out using X-ray fluorescence (XRF) photons generated from metal targets, with the detector positioned off-axis to suppress the primary beam. A mixed energy calibration approach combining internal test pulses with external XRF reference lines was applied to enable pixel-wise energy calibration and subsequent analysis of charge-sharing and cluster formation. Results were published in [54].

4.1.3 Timepix4 energy calibration at Elettra Synchrotron

During a beam-time in November 2023, comprehensive energy calibration measurements and evaluations of the Timepix4 detector at the Elettra synchrotron facility in Trieste, Italy, were conducted. The core objective was to perform a pixel-by-pixel energy calibration to accurately convert the raw Time-over-Threshold (ToT) signal into deposited energy, thereby optimizing the detector's energy resolution [49].

The experiments were carried out at the SYRMEP (SYnchrotron Radiation for Medical Physics) beamline of the Synchrotron, which is equipped to provide monochromatic X-rays [55]. This beamline can precisely select photon energies ranging from 8.5 keV to 40 MHz, with a high energy monochromaticity uncertainty of approximately $\Delta E/E \approx 2 \times 10^{-3}$. This is possible thanks to two Si (111) crystals in Bragg diffraction geometry, whose relative angle can be tuned with respect to the synchrotron beam, selecting the energy of the photons. Moreover, the intensity of the beam can be tuned either through aluminium filters or by de-tuning the second crystal and it is constantly monitored though a ionization chamber connected to a control application in control-room.

The X-ray beam was shaped by tungsten slits, resulting in a width of 28.6 mm, which fully covered the detector along the X-axis. Vertically, the beam was 3 mm high and had a Gaussian-like profile [49].

The Timepix4 assembly that was used for these measurements was bump-bonded

to a 300 μm thick silicon (Si) pixelated sensor reverse biased at 100 V and connected to the SPIDR4 read-out system produced by NIKHEF [56]. The acquisitions were taken in ToA-ToT data-driven mode (more details in subsection 2.2.2), using 2 of the 16 fast-links, each operating at 2.56 Gbps. Timepix4 pre-amplifier was set to high-gain mode, a threshold of 1000 e^- (equivalent to ~ 3.62 keV) was applied and the entire pixel matrix was equalized.

For uniform irradiation, the detection system was scanned vertically through the laminar X-ray beam at a constant speed of 2 mm/s. To ensure high statistical accuracy and manage data file sizes, approximately 4000 hits per pixel were collected at each energy level. The movements were made possible by a motorized hexapod that controlled the setup position and motion. The whole setup, except for Timepix4 and its control board, was managed using a custom Python framework. For the Timepix4 and SPIDR4 configuration and for the data acquisition, DataPix4 was used. This beam-time was in November 2023, so DataPix4 was not fully developed yet, and the read-out monitoring and analysis modules were not written. Moreover, the scripts and APIs were sometimes modified and optimized while taking data, so it was not possible to fully integrate DataPix4 into the Python system control software (it was successfully integrated during another campaign, described more in details is subsection 4.3.2).

To manage the motorized hexapod while acquiring data (for example, to move Timepix4 in between acquisitions), some APIs for the Python framework were implemented, and simple Python's script were launched exploiting `system` directives inserted into DataPix4 code.

A picture of the setup is shown in fig 4.4. The SPIDR4 read-out system is mounted on an hexapod and the Timepix4 chipboard is connected to the control board by a flexible cable, so that the board can be moved away from the beam. In the background, the second hexapod supporting the ionization chamber can be seen.

To perform the X-rays calibration, flat-field images at nine specific X-ray energy values: 8.5, 10, 12, 16, 20, 24, 28, 32 and 38 keV were acquired. For each pixel and each sampled energy, a Time-over-Threshold (ToT) distribution was generated. These average ToT values were then plotted against the nominal X-ray energy (E) for each pixel and fitted with the four-parameter Timepix4 calibration function: $ToT = a \cdot E + b - c/E - t$. Before analysing the data, the hits were clustered and only single-pixel events (where the photon charge was fully collected within one pixel) were considered. This

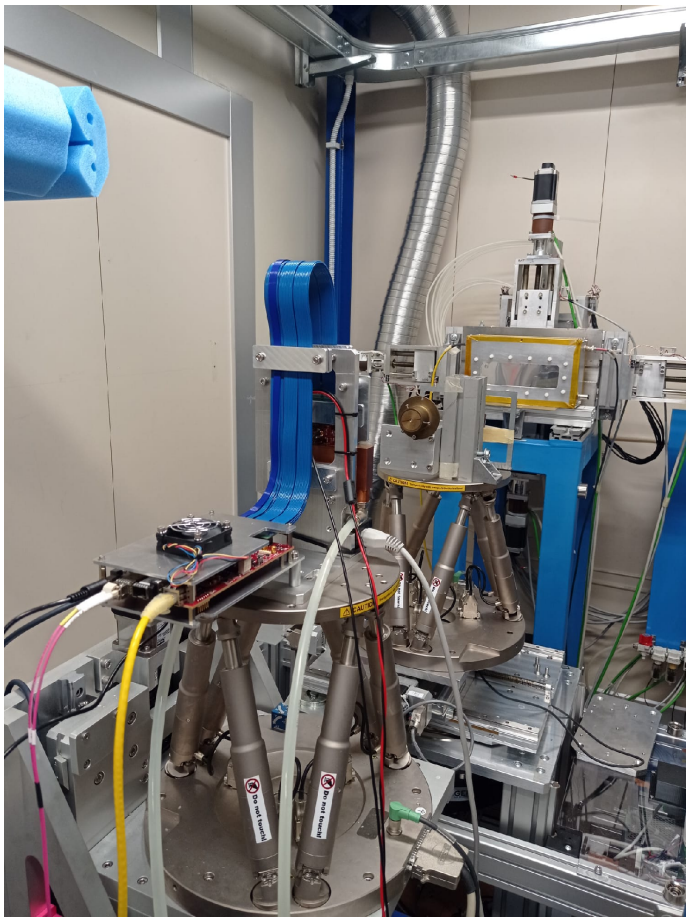


Figure 4.4: Picture of the setup at Elettra Synchrotron. In the foreground, the SPIDR4 control board in connected to Timepix4 through a flexible cable (in blue) and the setup is mounted on an hexapod. In the background, the ionization chamber is present, on a second hexapod.

approach ensured the calibration accurately reflected individual pixel responses without being convoluted by charge sharing across multiple pixels. The final energy calibration was obtained using both data from synchrotron X-ray beams (acquired during the beam-time) and the data internally generated using Timepix4 test-pulses. The results were published in [49].

4.2. Test-Beam Campaigns at CERN

Two comprehensive test-beams were conducted at the CERN SPS North Area, specifically in the H8 beamline [57], as part of the 4DPHOTON project, on November 2024 and November 2025. The test-beams' primary objective was to characterize the performance of the first 4DPHOTON detector prototypes as the photosensor of a Ring Imaging Cherenkov (RICH) system. The experimental configuration was maintained almost identically across both test-beam sessions.The primary change was related to to the control board: the SPIDR system was exploited during the first test-beam, whereas the second relied on the IDAQ control board. More information about the setup will be presented in subsection 4.2.1.

The H8 beamline provided a hadron secondary beam composed of, approximately, 80% protons, 20% pions and about 1% muons, with a negligible fraction of electrons. The particles' energy was assessed at $E = (140 \pm 1.8)$ GeV/c. The maximum beam intensity was around 10^5 particles per spill, with each spill lasting 4.8 seconds. The beam section was measured to be Gaussian in perpendicular directions to the beamline, with a standard deviation of $\sigma_x \sim 3.76 \pm 0.04$ mm and $\sigma_y \sim 3.28 \pm 0.04$ mm.

4.2.1 Setup Configuration

The experimental setup included three main components: a tracking system, a Ring-Imaging Cherenkov (RICH) system and a timing reference system.

Positioned upstream of the main experimental setup, the tracking system consisted of two Timepix4 v2 assemblies bump-bonded to 300 μm thick silicon sensors. The detectors were spaced 100 cm apart to achieve high precision in reconstructing particle trajectories.

The RICH system was custom designed and enclosed within a dark box and located downstream of the tracking system. It comprised a Cherenkov radiator and an optical system. The 4DPHOTON detector was placed inside the dark box too. This system was designed to project the Cherenkov rings onto the 4DPHOTON detector as a ring with an expected diameter of 18.0 mm and a dispersion of 0.9 mm[3]. Several commercially available fused silica lenses from Thorlabs Inc. were used to focus the Cherenkov pho-

[3] A Geant4 simulation of the optic system was made to study the configuration.

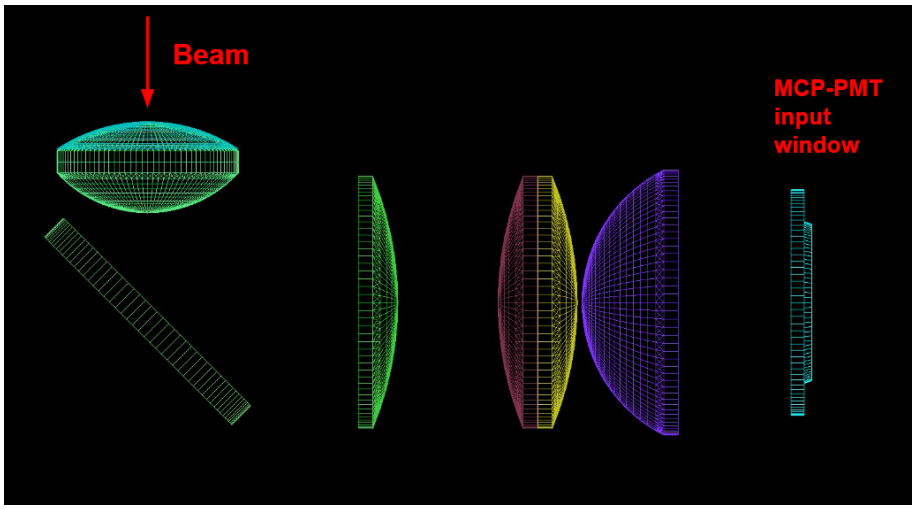


Figure 4.5: Schematic of the test-beam RICH optics system. After an initial lens that acts as Cherenkov radiator, a multiple lens system focuses the Cherenkov ring into the photocathode plane. The mirror, with an incident angle of 45°, allows to place the detector out of the particle beam.

tons. A schematic of the RICH system is shown in fig. 4.5.

The 4DPHOTON prototype under test was operated at a temperature of −10 °C[4]. The MCP was biased at 2100 V, with voltage differences of $V_{MCP-PHC}$ = 75 V (between MCP and photocathode) and $V_{Tpx4-MCP}$ = 150 V (between Tpx4 and MCP). This working point was chosen for stability during data taking, although the MCP had been tested up to 2400 V at Ferrara laboratory. Although various 4DPHOTON prototypes, with different MCP stack numbers and configuration, and different end-spoiling depths, have been produced, the one used during the test-beam was characterized by a Z-stack configuration [24] and 1d end-spoiling.

The timing reference system was placed inside a custom dark-box downstream of the Ring-Imaging Cherenkov (RICH) system along the beam direction. It consisted of a pair of Cherenkov detectors that were tilted at an angle of 65° with respect to the beam axis. The core of these detectors comprised thin plates of borosilicate glass (BK7) and their signals were read out by standard MCP-PMTs. Moreover, to check the beam intensity and to align the entire setup to the beam, a pair of scintillators have been

[4]Under these operating conditions, the Timepix4 dissipates more than 3 W of power, leading to significant heating of the detector. Elevated temperatures can increase the dark count rate and reduce the gain of the MCP.

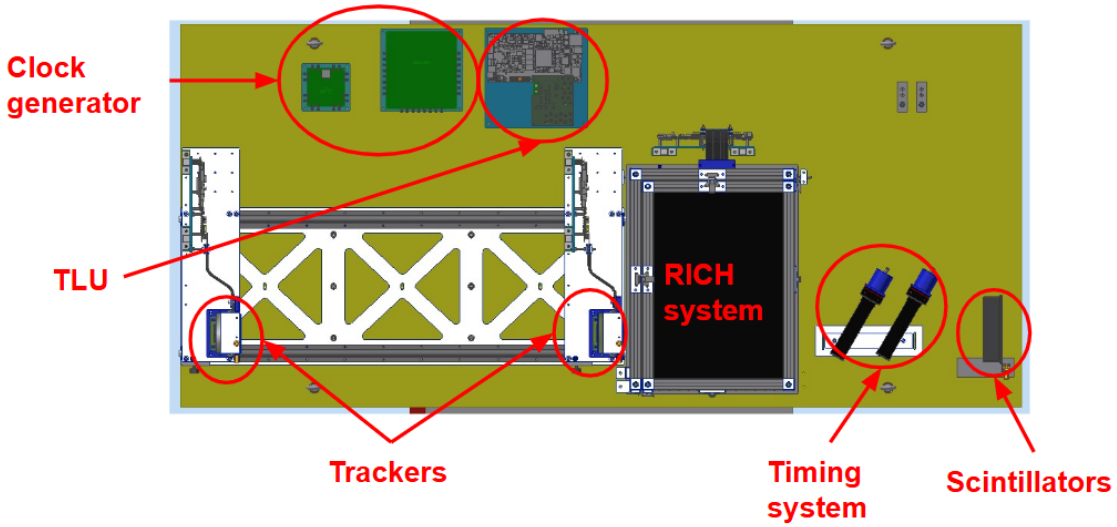


Figure 4.6: Schematic of the test-beam setup (from above). Different components are visible: 2 trackers, RICH system, timing system and scintillators are on beam axis, while the electronic components and the 4DPHOTON detector are outside the beam.

exploited. A custom Trigger Logic Unit (TLU) (described more in detail in subsection 4.2.3), synchronized with the SPS, was used to generate shutter signals, which were distributed simultaneously to both the 4DPHOTON detector and the tracking systems. Additionally, an external clock generator supplied a common reference clock to ensure synchronization among the three Timepix4 devices.

The Timepix4 assemblies (trackers and 4DPHOTON) were managed by the SPIDR4 control boards from NIKHEF (during the 2024 test-beam) and the IDAQ boards developed by INFN (during the 2025 one) and configured and read out using the DataPix4 software. A schematic of the entire setup is shown in fig. 4.6.

4.2.2 Network and Data flow

The network scheme and data flow was designed to handle setup configuration, high-speed data acquisition, real-time analysis and storage with back-up for multiple detectors and instrumentation simultaneously.

All the instrumentation was connected to a local network, to manage the entire setup from the control-room, located outside the beam area. In particular, I designed three main local subnets, connected through a computer called `Instrumentation PC`.

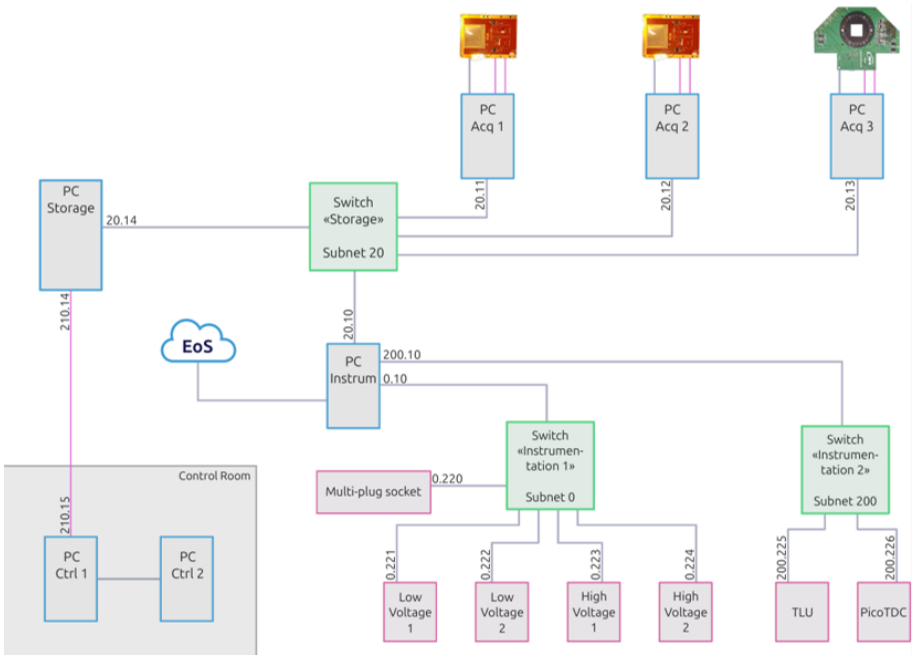


Figure 4.7: Schematic of the connection and local subnets for test-beam instrumentation. Blue boxes represent the computers, while the pink ones are electronic components connected to the network. Green boxes are network switches.

A schematic of the connections and network is presented in fig. 4.7.

The three detectors (2 Timepix4 bump-bonded with Si sensor and a 4DPHOTON detector) were connected, each one, to a dedicated computer, the `Acquisition PC` (`PC Acq 1`, `PC Acq 2` and `PC Acq 3`), which is necessary to configure them and acquire data. Those three computers were connected to the same local network, together with the `Storage PC`, a dedicated computer with hard disks for data back-up. Each `Acquisition PC` had a `sync_data.sh` script run by a cron-job every hour, to synchronize all the data with the `Storage PC` hard disks. The `sync_data.sh` script exploited `rsync` to compare files on both computer and copy the new ones into the storage. Once a day, a `delete_old_files.sh` script was manually launched on each `Acquisition PC` to delete all files that had already been backed-up, to free some space on their 500 GB SSD.

The `Storage PC` was used, like its name suggests, as a storage for all the data acquired. In addition to this, it was configured with a cron-job every several minutes to find the most recent data files (belonging to the same acquisition) and performing

a fast offline clustering, saving just the PDF plot of the resulting hitmap (similar to fig. 3.4). Those plots were useful to manually check, during the data taking, if everything was working and acquiring correctly. A script to perform a full clustering (saving ROOT files with all information) was also present in the computer and launched when needed.

The 4 power supplies, two for low voltages and two for high voltages[5], where connected, together with a multi-plug socket, to a local network called `Instrumentation 1`. The TLU and the timing system, instead, were linked to `Instrumentation 2`. The two local networks could talk to each other through the `Instrumentation PC`.

An additional back-up, both for detectors' data and for instrumentation logs and data, was performed online, using EOS data storage system [58].
Since the only computer connected to the internet was the `Instrumentation PC`[6], the `sync_data_eos.sh` script was launched from it. The script compared, exploiting `rsync`, the data folder in the `Storage PC` and the EOS folder, creating a back-up of each data file. In this way, all the data, coming from detectors but also from instrumentation, was saved at least twice, with one copy safely stored online.

From the control-room, we could manage all the instrumentation, configuring the devices, starting and stopping them, taking data. We also could force data clustering, syncing and backing-up, in case something went wrong with the automatically cron-jobs. Having two computers in the control-room allowed us to work in parallel, for example launching data taking with one computer and analysing data with the other. Both the control-room's computer had two fundamental scripts `config.sh` and `acquire.sh`, to launch, respectively, a complete configuration of the 3 Timepix4-based detectors and a data acquisition. To manage all the computers and the instrumentation outside the control-room, the SSH protocol was used. Exploiting the SSH key pairs (private-public), also the cron-job could use SSH connections without human intervention.

[5] One low voltage and one high voltage for the two trackers, while the other two for the 4DPHOTON detector.

[6] The `Storage PC` had no Wi-Fi module and there was no wired internet connection in the experimental hall.

4.2.3 Data Acquisition

The core mechanism for initiating and managing data acquisition was based on synchronizing the detectors directly with the timing signals from the SPS beam line, mediated by a custom electronic unit: the Trigger Logic Unit (TLU). A clock generator provided an external reference clock to synchronize the three Timepix4 devices. The TLU was responsible for generating key timing signals, specifically the shutter signals, based on the SPS signals. No hardware triggers were employed for general data acquisition, except for a shutter signal that initiated at the start of each beam spill.

The Timepix4 devices were configured to use external T0_sync and external shutter signals. The Extraction Warning signals from the SPS were sent to the TLU. The TLU then used these signals to generate the external T0_sync signal and a 5-seconds long external shutter (each spill was 4.8 seconds long). The T0_sync signal defined a common zero-time reference from which ToA measurements were made, by resetting the time counters, while the shutter signal defined the start and the end of the acquisition. This configuration allowed the acquisition to start right before the beam spill injection and stop at the end of it.

When the beam was on, the data acquisition was manually launched from one of the computer in the control-room. The three Timepix4 were synchronized and configured to acquire data exploiting external shutter signals. A user-defined parameter specified how many spills Timepix4 needed to acquire and the data taking was automatically stopped when reaching the total spill number. Online monitor plots (similar to the one in fig. 3.3), one for each Tracker and one for the 4DPHOTON detector, were displayed in a second monitor connected to the `Control PC`, in the control-room, allowing the user to see the collected data in real-time.

During the test-beams, more than one Terabyte of data was successfully acquired and stored, with hundreds of Gigabytes processed online and offline, allowing real-time monitoring of results.

4.2.4 Results

The overall configuration successfully produced observable Cherenkov rings on the 4DPHOTON detector during both the test-beams. Data confirmed the appearance of the Cherenkov ring with the expected size and position, alongside a background

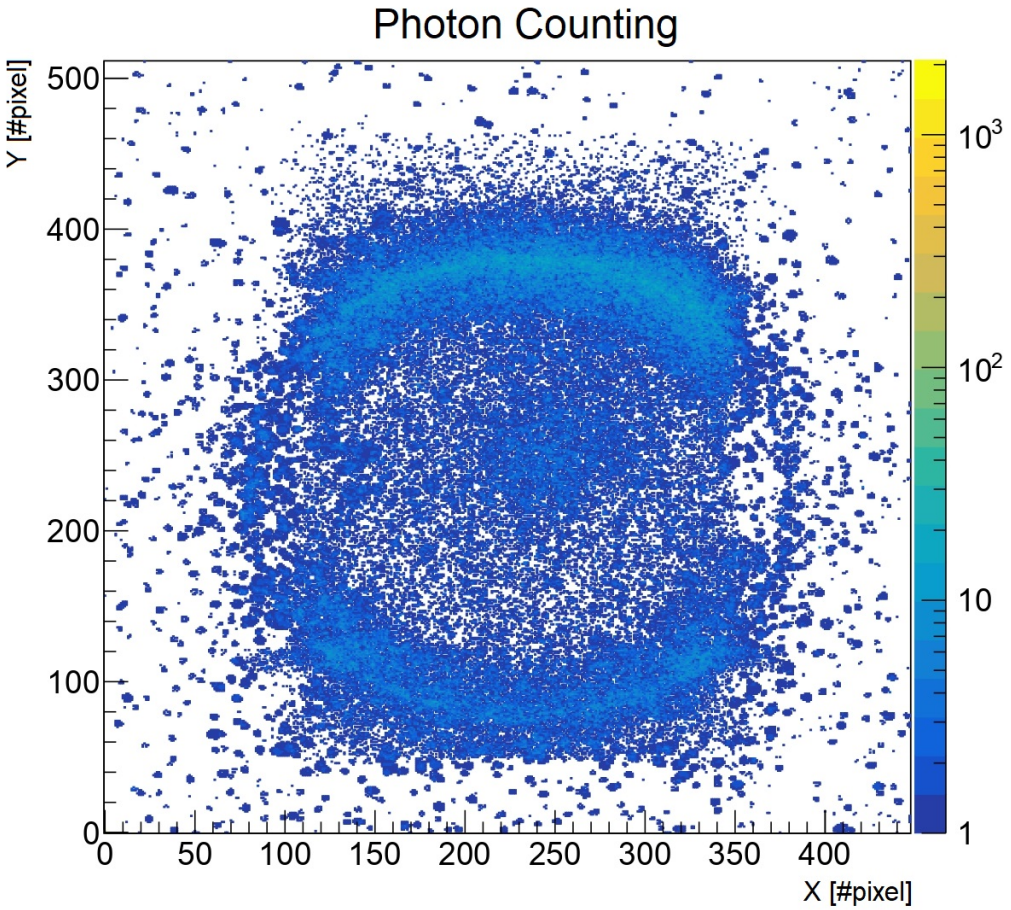


Figure 4.8: Hitmap from a data acquisition with a proton beam. A cumulative Cherenkov ring can be seen.

pattern consistent with previous laboratory characterization. A picture showing the Cherenkov rings is presented in fig. 4.8

The following results refer to the 2024 test-beam, since the 2025 test-beam data are still under analysis. To quantify this observation, a ring fit procedure was implemented, treating the ring centre coordinates (x, y) and the radius (R) as free parameters to minimize the distance between each detected hit and the fitted ring geometry. The resulting spatial reconstruction demonstrated excellent correlation with predictions derived from Geant4 simulations. The mean radius extracted from the experimental data was measured to be $R = 8.62 \pm 0.01$ mm. This measurement was in close agreement with the simulated radius of $R_{sim} = 8.66 \pm 0.01$ mm.

The analysis determined that the mean number of photons detected per event was 15 ± 1. This yield was also consistent with the expectation derived from the Geant4 simulation, which predicted 13 ± 1 photons per event.

Due to output data issues occurred during data taking in the 2024 test-beam, the timing system based on Cherenkov radiators could not been used for timing measurements. So the time resolution of the 4DPHOTON detector was estimated internally by relying exclusively on the timing information from the detected Cherenkov rings themselves. The intrinsic timing resolution of the detector was calculated resulting in $\sigma_{\text{timing}} = 283 \pm 7$ ps.

Those results were published in [59].

4.3. X-Ray Imaging with Timepix4

X-ray imaging has proven to be an essential tool across a wide range of scientific, industrial and medical fields, providing non-destructive insight into the internal structures of the object being studied, without altering it. The introduction of Timepix4, a highly advanced hybrid pixel detector, could significantly improve the capabilities of X-ray imaging, offering high spatial resolution, energy discrimination and precise timing measurements.

This section presents two distinct applications of Timepix4 in X-ray imaging: first, its use in cultural heritage conservation (subsection 4.3.1), where it is employed to reveal the composition of artworks without causing any damage, and second, its integration into the PEPI Lab at INFN Trieste for micro-CT[7] imaging (subsection 4.3.2), enabling high-resolution scans of samples at the micron level.

4.3.1 Cultural heritage applications

A research project, titled “Beyond state-of-the-art hybrid pixel detector system for X-ray spectral imaging in cultural heritage applications” has been funded by the PRIN[8] 2022 research funding program.

[7] Micro-CT (Micro-Computed Tomography) is a high-resolution imaging technique that uses X-rays to create detailed, 3D images of the internal structure of small objects at the micron scale.

[8] “Progetti di Ricerca di Interesse Nazionale”, a national funding scheme issued by the Italian Ministry of University and Research (MUR) to support fundamental research projects

The core objective of the project is the development and construction of a hybrid pixel detector system designed for X-ray imaging, based on the Timepix4 ASIC coupled with a Cadmium Telluride (CdTe) sensor. The prototype system is primarily intended for applications in the cultural-heritage domain. The project's goal is to develop a fully integrated experimental setup capable of performing colour X-ray imaging—specifically Energy-Resolved Radiography (ERR) [60], that allows the acquisition of projection images in multiple energy windows, enabling qualitative and quantitative material discrimination in two dimensions, and Multi-Energy Computed Tomography (MECT) [61], that extends this approach to three dimensions, providing volumetric maps of material composition and elemental distribution within complex objects.

Such a system will provide a non-invasive, non-destructive three-dimensional characterization, enabling the determination and the spatial mapping of materials and elemental composition within an object, using a conventional polychromatic X-ray source.

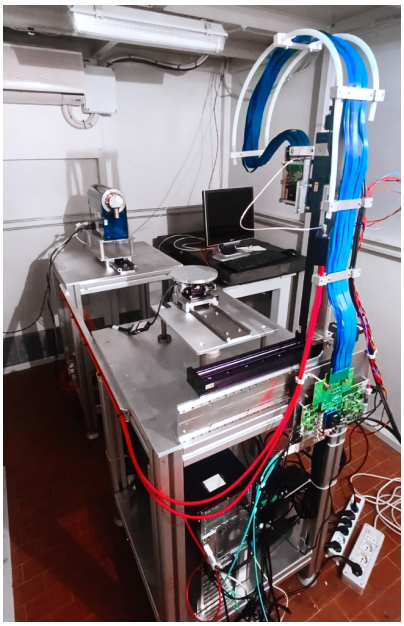


Figure 4.9: Experimental setup at the Physics Department of the University of Bologna.

During the last year, the setup has been built and tested at the Physics Department of the University of Bologna. A picture of the setup can be seen in fig. 4.9. In the image, the X-ray tube can be seen at the back, positioned above the second table. Closer to the foreground, a rotational stage is installed to allow the sample under investigation

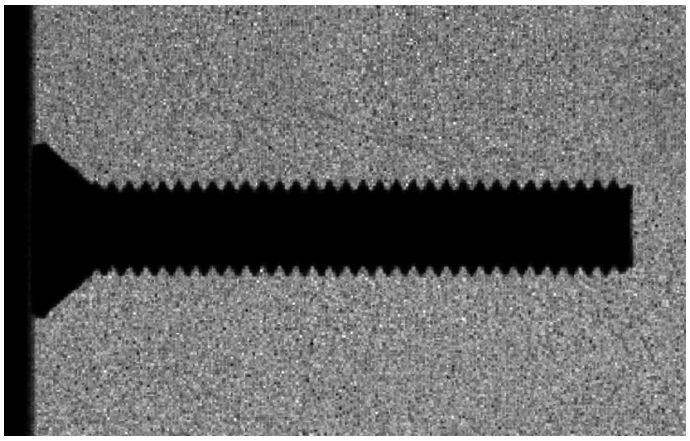


Figure 4.10: The first radiograph obtained with the setup. The screw profile can be seen, with the detailed threads along the length.

to be rotated during acquisition.

Two additional translation stages, aligned along the X and Y axes, enable precise movements of the Timepix4 detector in both directions, facilitating both measurements as well as system alignment and calibration. The Timepix4 ASIC is visible at the end of the blue cable, which connects it to the IDAQ board. The whole setup is connected to the computer located beneath the table, which can be operated from outside the control-room.

Over the past few months, an initial series of preliminary tests has been carried out to prepare the complete experimental setup. These activities involved both the mechanical components of the system (moving stages) and the configuration, synchronization and data-acquisition chain (Timepix4 and IDAQ read-out system). Dedicated studies were performed to assess the linearity and stability of the system response, investigating its dependence on both the acquisition time and the X-ray photon emission rate of the primary source. In addition, flat-field acquisitions and radiographs were conducted as part of the setup validation procedure, allowing the identification of potential non-uniformities and ensuring the correct operation of the system. Overall, these tests confirmed the proper functioning and reliability of both the mechanical stages and the data-acquisition system. An example of a radiograph of a screw, obtained during this characterization phase, is shown in fig. 4.10.

Over the next few months, ad-hoc control scripts will be implemented, to allow all the instrumentation to be controlled from a single interface. Following their deploy-

ment, the initial data-acquisition runs will be performed. These studies will form the basis for evaluating the system's performance in realistic cultural heritage scenarios.

4.3.2 Micro-CT at INFN Trieste

The PEPI (Photon-counting Edge-illumination Phase-contrast Imaging) Lab is a flexible compact multi-modal imaging setup designed and developed at INFN Trieste. Its aim is to perform micro CT with high resolution using the edge-illumination (EI) technique [62] and a chromatic detector. CT is done in two non conventional modalities: spectral (XSI) and phase contrast (XPCI) [63].

The setup of the PEPI Lab is composed by an X-ray source, a sample mask, a sample stage, a detector mask and a detector, each one mounted on mechanical stages that allows for independent motion of all the elements. X-ray source, detector and all the motion stages must be controlled and managed outside the experimental hutch, since it is not possible to access the PEPI Lab during acquisition for safety reasons.

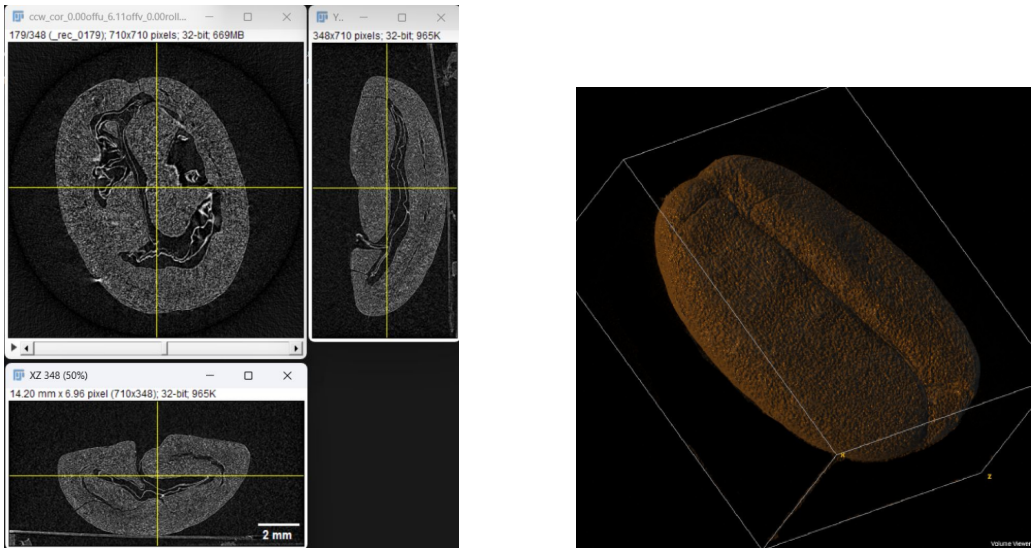
To control the entire environment, a Python framework called PEPIControl has been implemented ad-hoc.

Every instrument has its own dedicated class, that provides the fundamental methods that the user can call to perform simple actions. The detector's class must provide at least 3 methods: `initialize()`, `acquire()` and `terminate()`, to respectively configure the device, acquire data and stop the acquisition.

The goal of the work with the research group at INFN Trieste was to integrate the Timepix4 ASIC into PEPI Lab, both from a hardware as well as software perspective, replacing the current detector and perform some feasibility studies of the firsts μCT scans using Timepix4 bump-bonded to a 500 μm GaAs sensor.

To include DataPix4 into PEPIControl, a new class, called `timepix4_CT`, has been implemented. This instantiates an object with the 3 fundamental methods described above, in addiction to other useful methods and attributes. Then, using `pybind11`, a corresponding Python class, called `Timepix4_CT_py` was created, exploiting the `CMake` function `pybind11_add_module()`.

To acquire data for a CT scan, a custom Python script has been implemented. This connected to the mover responsible for the sample movements and to Timepix4. It initialized the devices and, in a loop, slightly rotated the sample and performed a data



(a) Three isometric views of the coffee bean. (b) Complete coffee bean reconstructed.

Figure 4.11: 3D Rendering of the coffee bean

acquisition for a fixed time. Timepix4 acquired data for 2 seconds for each angle, with 4 frames of 0.5 seconds each and the 16-bit depth counters (the 16-bits frame-based mode is described in details in the dedicated paragraph of subsection 2.2.2). Frame and data acquisition duration could be modified by the user thanks to a configuration file.

When the full rotation scan was over, the data was processed via a custom Python script that decodes the raw binary files and produces one *tiff* image for each data acquisition (summing the 4 frames). Those were then processed to create a complete CT scan of the sample.

One of the firsts samples used for a CT scan with Timepix4 was a coffee bean. The CT was acquired using 1440 projections over 360°, with a slightly off-axis acquisition to modestly increase the field of view; using a detector 512 pixels-wide, reconstructed images of 710×710 pixels were obtained. After the acquisition, the TIGRE [64] software was used to reconstruct the 3D model. The 3D rendering representation of the bean is show in fig. 4.11.

Moreover, exploiting the data-driven acquisition mode, with precise Time-over-Threshold (and therefore energy) measurements of each event, the first full-spectral micro-CT images were acquired, using a Timepix4-based detector equipped with a

1 mm thick CdTe sensor.

The sample was a custom plastic phantom containing 3 small test tubes with three contrast agents commonly used in biomedical imaging: silver, iodine and gadolinium. The CT was performed by acquiring 360 projections over 360°, with an exposure time of 5 s for each angle. After offline data clustering, energy-binned images were processed with a material decomposition algorithm, producing 3D density maps for each individual contrast agent, resulting in a "coloured" tomography, with a different colour for each material [65].

Results about this study and the first full-spectral micro-CT acquired with a Timepix4-based detector will be published soon.

4.4. Beam monitoring system at Fermilab

The development of high-granularity, fast and optimized beam diagnostics has become increasingly important for the experimental program at the Integrable Optics Test Accelerator (IOTA) and the FAST[9] facility at Fermilab [66]. Precise, time-resolved measurements of beam properties are essential both for routine machine operation and for advanced studies. To address these needs, a new beam monitoring system based on the 4DPHOTON detector is planned to be installed on the IOTA ring (fig. 4.12). The overall geometry of the IOTA ring is determined by eight dipole (bending) magnets, consisting of four 30-degree bends and four 60-degree bends. The 4DPHOTON detector will be installed above one of the 60-degree dipole magnets, where it will collect synchrotron-radiation light from circulating electrons, in order to measure both the photons spot size and their arrival time. This configuration will allow the system to operate as a non-invasive, turn-by-turn diagnostic tool.

During the summer of 2025, the 4DPHOTON detector was delivered to Fermilab, enabling the assembly of an initial test setup. For the first stage of commissioning, the detector was being installed outside the IOTA ring, allowing independent characterization and performance validation before its final placement above the dipole. To simulate the expected beam conditions, a LED source controlled by an external pulse generator was exploited. A photo of the detector in its test setup is shown in fig. 4.13.

[9] Fermilab Accelerator Science and Technology

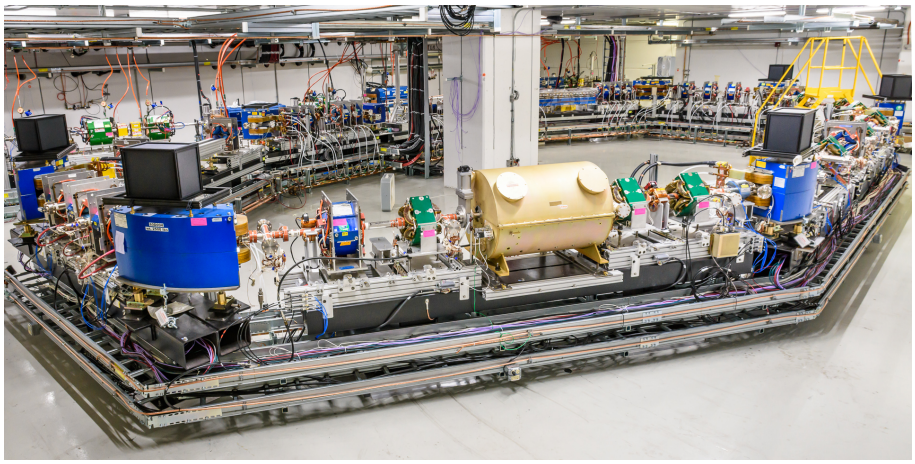


Figure 4.12: The IOTA Ring at Fermilab. In green, the quadrupoles are visible. In blue, on the corners, the bending magnets carry a dark box with the beam monitoring system inside.

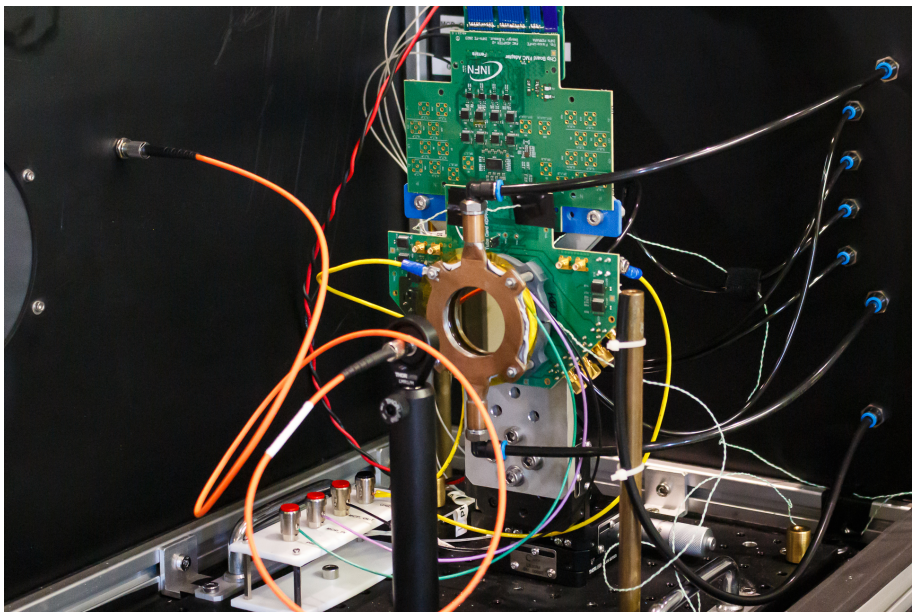
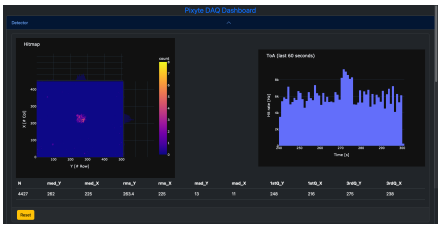
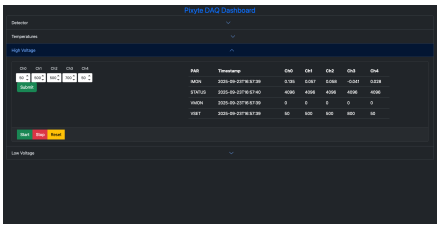


Figure 4.13: The 4DPHOTON detector inside its dark box.



(a) Detector data monitor. On the left, real-time hitmap of the last second of acquisition. On the right, hit-rate monitor of the last minute of acquisition.



(b) High Voltage monitor and controls. On the right, summary table with information per channel. On the left, the user can set the voltages.

Figure 4.14: Screenshots from the custom GUI

A primary goal of the project was to provide an online monitoring tool for the IOTA ring. The DataPix4 visualization and analysis classes were exploited and adapted for this purpose. In particular, the online clustering plots (visible in fig. 3.4) were changed, to plot the cluster centroid and ToA, instead of dimension and charge. Moreover, the online clustering output was integrated with a custom GUI, enabling the user to control all instrumentation —including Timepix4, temperature sensors and power supplies— while simultaneously displaying real-time data from Timepix4 and other devices. Some screenshots from the custom GUI, displaying real-time data from 4DPHOTON detector and monitoring of the high-voltage power supply, are shown in fig. 4.14. This demonstrates how easily the output of the DataPix4 online clustering can be redirected toward custom graphical interfaces. By default, the clustered data are written to file and can be viewed through the built-in online monitoring tools. In this setup, however, the output stream was instead routed to a Unix socket, which was read by a Python-based graphical interface accessible via a web browser for instrument monitoring.

CHAPTER 5

Neural Network for Particle Identification

Neural networks are computational models inspired by the human brain, designed to learn complex relationships from data through interconnected layers of simple processing units, or *neurons*. Among their various architectures, convolutional neural networks (CNNs) have emerged as particularly effective for image recognition. By applying convolutional operations, CNNs automatically extract hierarchical spatial features, such as edges, textures and shapes, directly from raw images [67].

In this chapter, a convolutional neural network designed ad-hoc for particle track recognition will be described. The network has been trained on a dataset of tracks generated from particles interacting with a 300 μm silicon sensor bump-bonded to the Timepix4 ASIC. The network design will be discussed in section 5.1, while results will be presented in section 5.4. A quantization of the network, exploiting the AMD Vitis AI environment [68] will be presented in section 5.5. Finally, the integration of the neural network into the DataPix4 framework will be discussed in section 5.6.

5.1. General Structure and Design

The fundamental requirement for the neural network was to classify traces detected by a Timepix4 coupled with a 300 μm silicon sensor. Specifically, distinguishing between four main particle types: Alpha particle, Electron, Muon and Photon. A custom neural network architecture was designed and implemented from scratch, as conventional convolutional neural networks, typically developed for large-scale natural images (e.g. objects, animals or people), proved unsuitable for this application. The particle track images used in this work are significantly smaller in scale, with features as narrow as one pixel and up to tens of pixels in width. Even with extensive fine-tuning, existing architectures could not adequately process such low-resolution input data.

Since Timepix4 can simultaneously measure both the Time-of-Arrival (ToA) and the Time-over-Threshold (ToT) of an event, the chosen input data type incorporates spatial information (pixel coordinates) together with the corresponding ToA and charge values, those directly derived from the ToT measurements, as described in subsection 3.5.3.

In particular, the network was design to accept a 3D vector of shape (2, 50, 50), made of a 2D matrix of ToA and one of charge stuck together, as displayed in fig. 5.1.

The 3D vector will then be divided in its two components thanks to a pre-processing layer of the network. Each 2D matrix is independently processed by a convolutional neural network with the same characteristics, but different weights. Then, the two processed matrices (with dimensions 50x50 and 4 filters) are combined together using a `concatenate` layer, to create a 3D stack with shape 50x50x8. The three following 2D convolutions further process the entire data, then a dense layer (with `softmax` activation) with four output units returns the classification. Since the four classes are independent, the corresponding labels are codified using the *one-hot encoding* method [69]. The dense layer of the network returns a vector of four elements: the position of the greater element indicates the classification and its value indicates the probability. A schematic of the neural network can be seen in fig. 5.2.

Particles' tracks, when using Timepix4 bump-bonded to a 300 μm silicon sensor, are usually composed by a small number of pixels, between 1 and $O(50)$. Those, however, can vary greatly in shape, going from a single pixel, to a big round cluster or a long curly shape. This variability introduced uncertainty in selecting an appropriate kernel

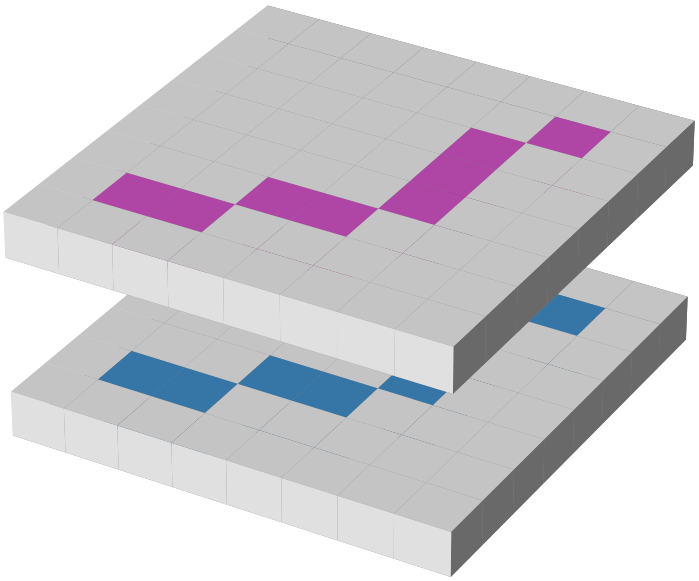


Figure 5.1: The neural network takes as input a 3D vector of shape (2, 50, 50). Here a simplified scheme of the 3D vector, split in a ToA and a charge 2D vectors.

size for the convolutional layers. To accommodate this diversity, multiple convolutional operations with different kernel sizes were implemented in parallel, allowing each particle track to activate the most suitable filter. The resulting feature maps were then combined using a `sum`[1] layer to integrate the information from all kernel scales.

This design, with a series on inception blocks, was inspired by the GoogLeNet, a deep convolutional neural network architecture that won the ImageNet Large-Scale Visual Recognition Challenge in 2014 [71]. The inception block used can be seen in fig. 5.2, on the right. These are 1×1, 3×3, 5×5 and 10×10 kernel size blocks.

Figure 5.3 shows four representative particle's tracks, one from each class, processed through the neural network. Each column corresponds to one of the convolutional operations within the inception block (1×1, 3×3, 5×5 and 10×10). The images illustrate the feature maps obtained by applying each convolution to the original input image, while the final column presents the output of the subsequent `sum` layer. It can be observed that smaller particles, such as photons (typically one or few pixels wide)

[1] The `maximum` layer had slightly better results, but Vitis AI was not capable of quantize it [70]. Among the quantizable layers, the `sum` was the one performing better. This will be explained more in detail in section 5.5.

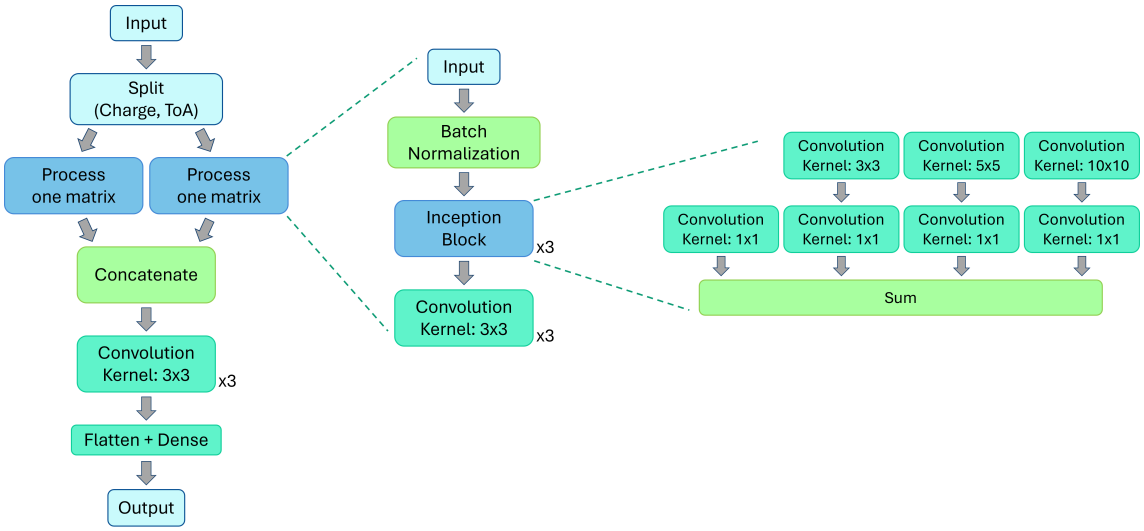


Figure 5.2: Scheme of the Convolutional Neural Network. On the left, the complete network. In the centre, the network used to process a single 2D matrix. On the right the "inception block".

and small alpha particles (compact clusters spanning a few tens of pixels), predominantly activate the 1×1 and 3×3 kernels. In contrast, elongated tracks produced by electrons and muons primarily engage the 5×5 and, occasionally, the 10×10 kernels.

5.2. Dataset

The dataset used for training contains more than 3500 images, with approximately 900 images for each label. The dataset used for validation is composed by approximately 800 images instead. An example of some particles' tracks (charge's matrices) in the dataset are shown in fig. 5.4.

Different particle types leave distinct geometric signatures due to their interactions with the 300 μm thick silicon sensor:

- Alpha particles: release energy rapidly due to their high cross-section, forming dense, round clusters usually spanning one or more tens of pixels.

- Electrons: generally create curved, elongated tracks, often showing a localized peak of energy released where the particle stops.

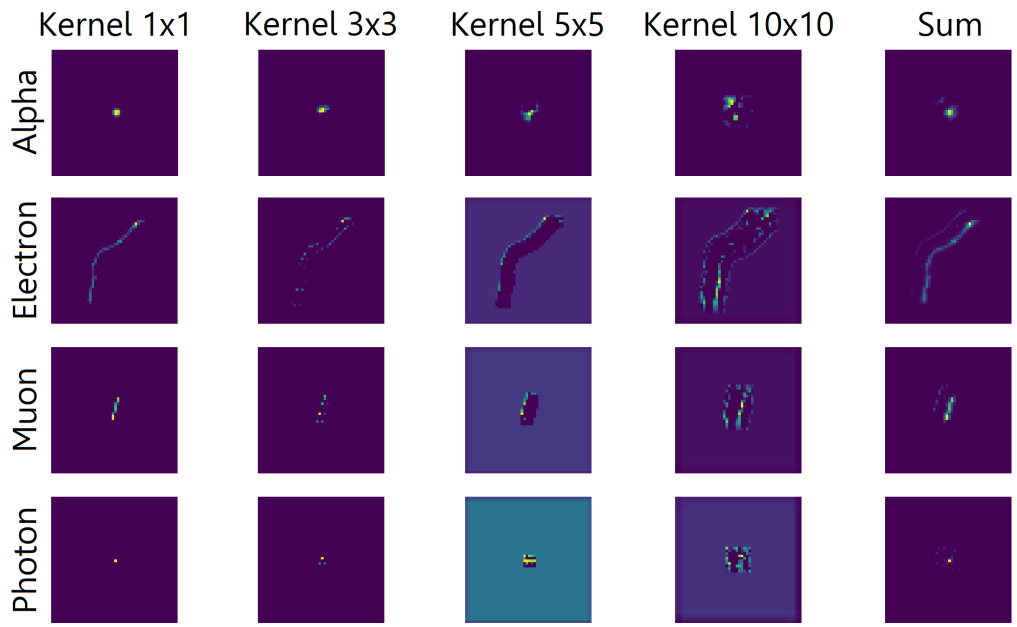


Figure 5.3: Four representative particles (one per class) are shown as processed through the inception block of the network.

- Muons: create elongated and typically straight tracks, often traversing large sections of the sensor while releasing a relatively constant energy value.
- Photons: (X-ray photons) are neutral, massless particles that typically appear as single pixels or, occasionally, pairs of pixels.

Because Timepix4 is a relatively new ASIC and its internal behaviour is difficult to reproduce accurately in simulations, the most straightforward approach for labelling the dataset was to rely on human annotation. An interactive Python script has been implemented, to help the expert user labelling the particles' track. For each track, isolated and processed with the clustering algorithm described in subsection 3.4.2, three plots are presented to the user, showing the values in the Time-of-Arrival, Time-over-Threshold and charge 2D matrices. Below the plots, 4 buttons, corresponding to the 4 classes in the dataset, are present, in addiction to an `exit` button, to leave the program, and an `error` one, that skips the current track. When the user clicks on a button, the track is automatically saved with its label and the subsequent track is displayed. In fig. 5.5, a screenshot of the program interface is presented.

Thanks to expert users, a dataset of ~ 800 tracks, approximately two hundred for

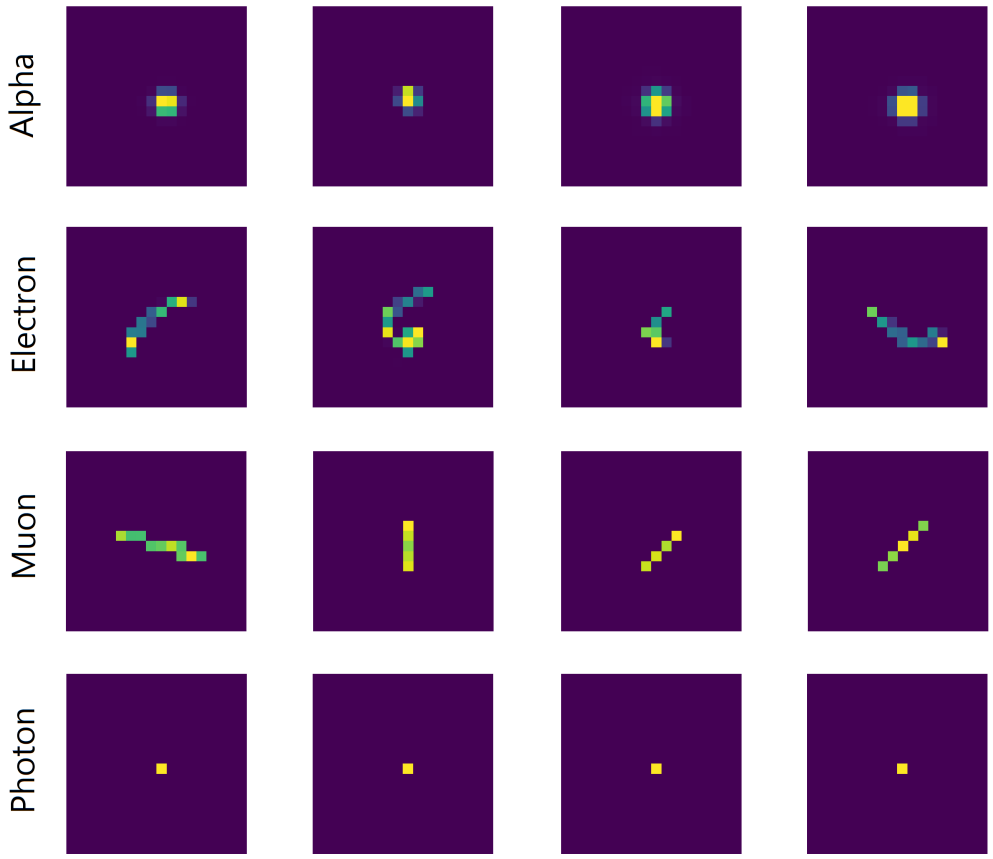


Figure 5.4: Example of particles' tracks in the dataset. The ones displayed are the charges' matrices.

each class, has been created. To increase its dimension, dataset augmentation has been performed, flipping and rotating the images. Moreover, random padding has been added, decentralizing the tracks from the matrix, significantly improving the network's robustness to translations. Finally, a part of ~ 40% of the dataset was chosen to be the *validation* one, while the other tracks remained in the *training* dataset.

5.3. Training

The network was trained using the Adam optimizer [72], with a learning rate of 0.001. The chosen loss function was the Categorical Cross-entropy [73], since the input data needed to be classified into one of several mutually exclusive classes. The training was

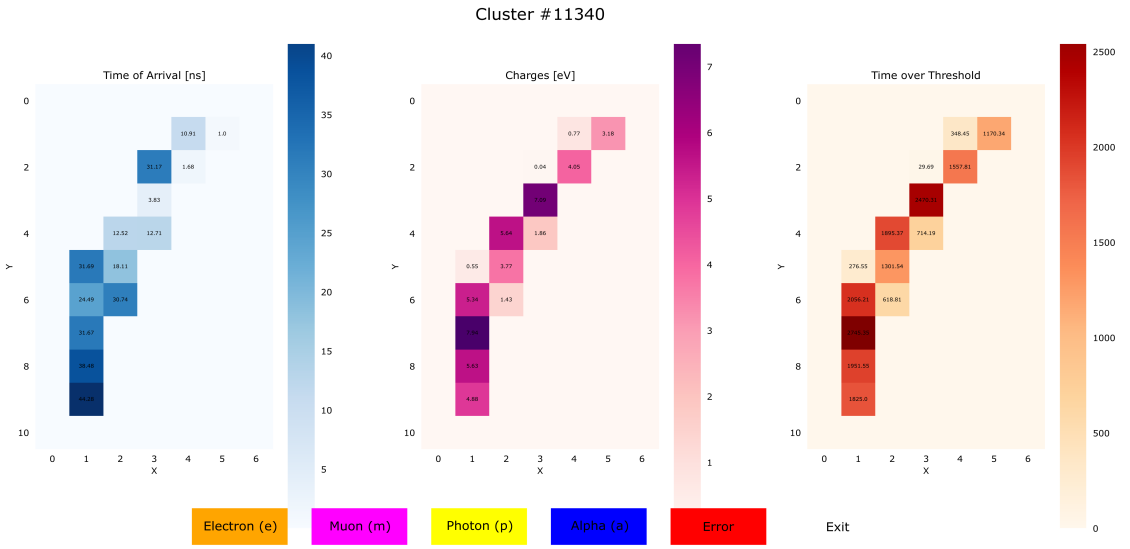


Figure 5.5: Screenshot of the program showing the plots of a track to be labelled.

supervised[2], using the dataset described in the previous section 5.2. The *early-stopping* technique[3] was exploited, monitoring the categorical accuracy on the validation set.

Since the model was implemented in Python3, using the `Tensorflow` library [74], the training was performed using the `fit` method, as shown in the code below.

```
callback = tf.keras.callbacks.EarlyStopping(
    monitor="val_categorical_accuracy", min_delta=0.001, patience=8)
mcp_save = tf.keras.callbacks.ModelCheckpoint(filename + ".hdf5",
    save_best_only=True, monitor="val_categorical_accuracy", mode="max")
history = model.fit(X_train, Y_train, epochs=50, shuffle=True,
    batch_size=32, validation_data=val_dataset, callbacks=[callback, mcp_save])
```

The loss and accuracy history on both training and validation dataset, can be seen in the plots in fig. 5.6. Although the training foresaw 50 epochs, the best results were at the 27th epoch, so the training automatically stopped after 8 epochs[4].

A mild degree of overtraining can be observed, as the validation loss does not show

[2] Various unsupervised training strategies were explored in an attempt to learn meaningful representations directly from the data, without the need for manual labelling. Despite extensive tuning, these approaches failed to produce accurate results and, consequently, will not be taking into consideration in this work.

[3] A regularization technique to prevent over-fitting by halting training when validation performance stops improving.

[4] Due to the `patience` parameter in the `fit` method.

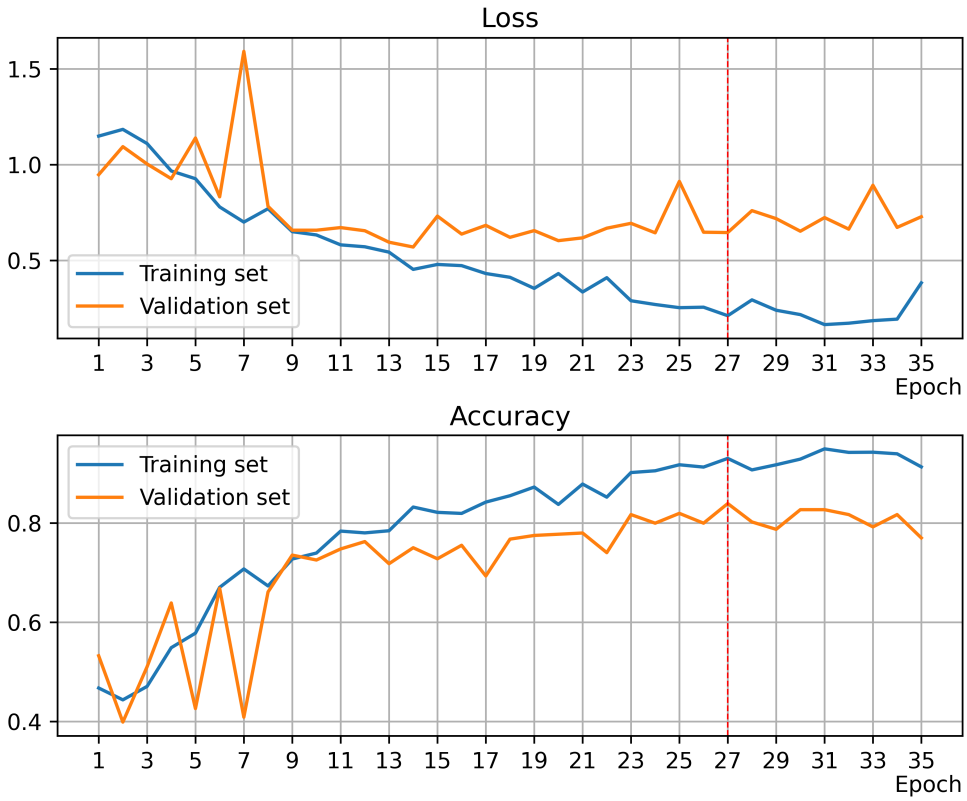


Figure 5.6: Loss and accuracy parameters on training and validation datasets during training.

a significant decrease after approximately the 14th epoch, while the training loss continues to improve. However, the validation accuracy keeps increasing up to the 27th epoch, reaching a maximum value of 0.84, compared to approximately 0.75 at epoch 14. Since the primary evaluation metric for this task is accuracy, the model was selected based on the epoch that maximized validation accuracy rather than minimizing validation loss.

The best accuracy on validation set was 0.84, with a loss of 0.64. Those correspond to an accuracy of 0.93 and a loss of 0.21 on the training set. Despite the presence of mild overfitting, this choice results in a better generalization performance in terms of accuracy, without a significant degradation of validation loss.

The training was performed only on CPU and each epoch took approximately 117 seconds.

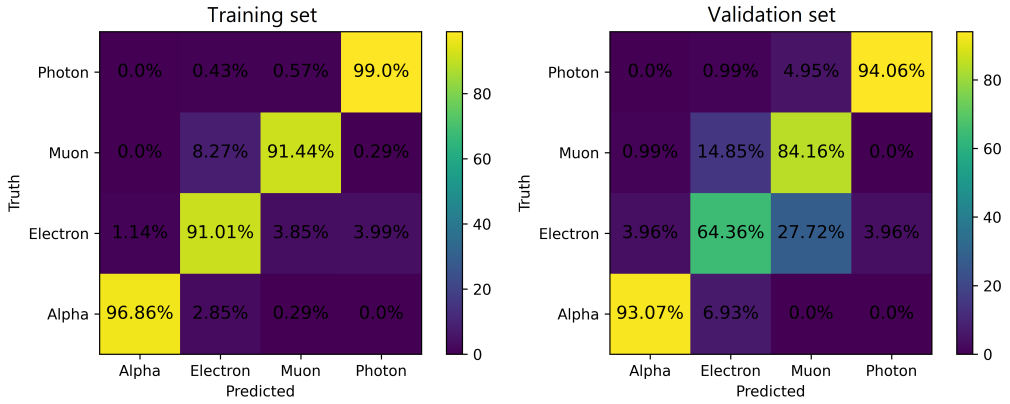


Figure 5.7: Confusion matrices for training and validation datasets.

5.4. Performances

The neural network, after 27 epochs of training, reached an accuracy of 0.84 and a loss of 0.64 on the validation set, as shown in fig. 5.6.

To evaluate the performance of the proposed neural network, two confusion matrices[5] were generated: one based on the training set and another on the validation set. Fig. 5.7 shows the two confusion matrices, with the true label on the y-axis and the predicted label on the x-axis.

The confusion matrix of the training set shows a strong diagonal pattern, indicating that almost all samples were correctly classified. In particular, the network reaches an accuracy of 99% on the photon, probably due to their characteristic shape (cluster of 1 or few pixels). Also, quite all the alpha particles are correctly classified, with an accuracy of almost 97%. Electrons are probably the most difficult particle to classify, due to the variety of their shape, dimension and values. The confusion matrix of the validation set shows a weaker, but still present, diagonal pattern. Although almost all alpha particles and photons are still correctly classified, it seems that the network sometimes struggles to distinguish between muons and electrons.

Figure 5.8 shows an example of the particle-track visualization obtained from a data acquisition of natural radioactivity. The figure shows only a cropped section of the detector's active area. The full acquisition surface is significantly larger, but visualizing

[5] A confusion matrix is a tabular representation that summarizes the performance of a classification model by comparing the predicted class labels with the true class labels for all samples.

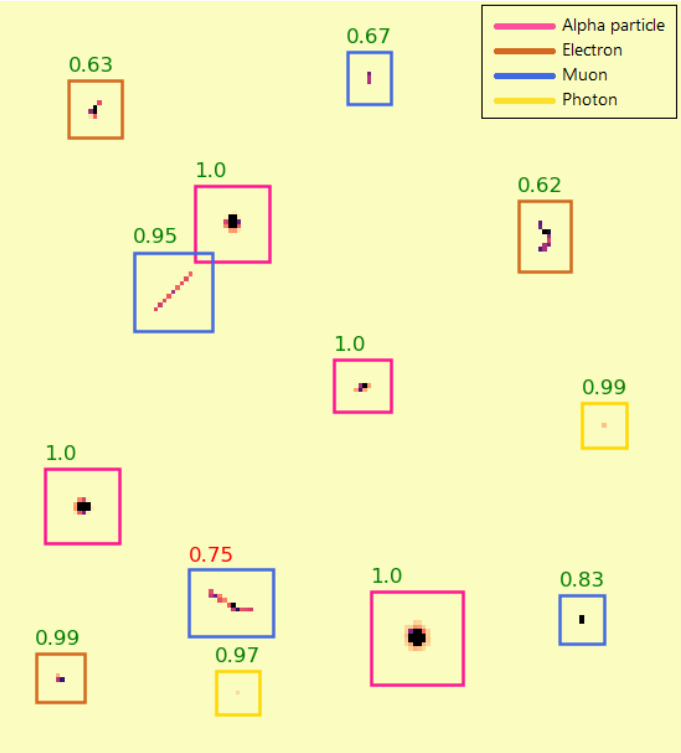


Figure 5.8: Zoomed-in view of the detector's active area showing reconstructed particle tracks classified by the neural network. Each track is enclosed in a colour-coded rectangle, while the number near each rectangle indicates the network's classification certainty (green for correct predictions, red for misclassification).

it in its entirety would render the particle tracks barely distinguishable. A zoomed-in view was therefore selected to highlight the track morphology and the corresponding classification results. The figure displays the charge deposited by each particle on each pixel; for visualization purposes, pixel values were clipped at 10 keV using the `numpy.clip()` function to prevent highly charged tracks from dominating the colour scale and to make low-charge tracks visible.

Each detected track is enclosed within a colour-coded rectangle corresponding to the class assigned by the neural network: pink indicates alpha particles, blue corresponds to muons, brown to electrons and yellow to photons.

Next to each bounding box, a numerical value is displayed, representing the confidence level of the classification predicted by the network for the assigned label. The number is shown in green when the prediction matches the true particle type and in

red when the classification is incorrect. It can be observed, for example, that all alpha-particle tracks are correctly identified with a confidence of 100%, demonstrating the network's strong discrimination capability for this class. Also the two photon are correctly classified with a confidence of 97 – 99%. The only error is an electron's track that has been labelled as muon with a 75% confidence level, maybe due to its straight shape.

In summary, the performance evaluation demonstrates that the neural network achieves reliable classification accuracy across all particle types, in particular alpha particles and photons, with good generalization from training to validation data, as shown in the confusion matrices in fig. 5.7.

5.5. Vitis AI - AI inference on FPGAs

The rapid development of deep learning applications has led to increasing demand for efficient deployment of neural network models on specialized hardware. While high-level training frameworks such as TensorFlow or PyTorch enable fast prototyping and model development, their floating-point implementations are often computationally expensive and power-intensive, especially when executed on embedded or edge devices. To address these challenges, AMD has developed Vitis AI, a comprehensive development environment that optimizes and deploys artificial intelligence workloads on AMD adaptive computing platforms, including FPGAs (Field-Programmable Gate Arrays), SoCs (Systems-on-Chip) and Versal adaptive SoCs [75].

Vitis AI supports models trained in mainstream frameworks (TensorFlow, PyTorch, ONNX and Caffe) and converts them into optimized representations suitable for hardware acceleration. The workflow integrates quantization, pruning, compilation and deployment, allowing developers to transform high-precision floating-point models into low-precision (typically `int8`) fixed-point networks without significant accuracy loss [76].

To increase the performances of the neural network described in this chapter, the corresponding quantized model has been created exploiting the Vitis AI tools. In the following subsections, the quantization process and results will be presented.

5.5.1 PTQ vs. QAT

Quantization is a key step in optimizing neural networks for hardware deployment, as it reduces numerical precision (typically from 32-bit floating point to 8-bit integer) to improve computational efficiency and memory usage. Two main strategies can be adopted: Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT). In PTQ, quantization is applied after the model has been fully trained using floating-point weights. The trained parameters are simply converted to lower-precision values and the model is re-evaluated without retraining. In contrast, QAT incorporates the quantization effects during the training process itself. By simulating low-precision arithmetic in both the forward and backward passes, the network learns to adapt its parameters to the quantized representation. Generally, QAT requires additional training time and computational resources.

In the next sections 5.5.2 and 5.5.3, the two different approaches will be discussed. To train and test the quantized models, the dataset was slightly modified, encoding both the charge and the Time-of-Arrivals of the tracks using `int8` values, instead of `float32` ones.

5.5.2 Post-Training Quantization Performances

To quantize the neural network, the trained model described in section 5.1, 5.3 and 5.4 has been loaded exploiting the Vitis AI Python libraries, as shown in the code below. Then, using the `VitisQuantizer`, it has been quantized into a `int8` model.

```
# Load trained model
with vitis_quantize.quantize_scope():
    q_model = tf.keras.models.load_model(filename, compile=False)
# Create a Vitis  AI Quantizer
quantizer = vitis_quantize.VitisQuantizer(q_model)
# Quantize model
quantized_model = quantizer.quantize_model(
        calib_dataset = train_dataset,
        calib_batch_size = batch_size,
        calib_steps = len(train_dataset)/batch_size,
        include_fast_ft = False
)
```

After the quantization process, the new `int8` model has been compiled using the `categorical_crossentropy` loss function and the `categorical_accuracy` as accuracy metric.

After applying post-training quantization to reduce the model precision from 32-bit floating point to 8-bit integer (`int8`) weights, a substantial degradation in performance was observed.

The quantized model achieved an accuracy of approximately 0.25 on the validation dataset, which corresponds to the expected value of random guessing among the four output classes. In addition, the loss value significantly increased to 31.24, indicating unstable predictions and a poor fit to the data.

In general, the post-training quantization severely affected the representational capability of the network, leading to a complete loss of discriminative power.

Changing the `include_fast_ft` parameter, to add some epochs of fine tuning, did not help changing the behaviour of the neural network. To address this issue, a different approach was adopted: the model was quantized without fast fine-tuning and additional fine-tuning epochs were applied manually using the standard `model.fit()` procedure. This strategy successfully restored the network's behaviour, resulting in an accuracy of 0.79 and a loss of 1.13 after 19 epochs.

In fig. 5.9, the loss and accuracy trends can be seen. After 19 epochs, the best value is reached. In fig. 5.10, the confusion matrices for training and validation datasets. The diagonal pattern remains clearly visible, as observed in the confusion matrices of the `float32` model, indicating that the overall classification capability is preserved after quantization. However, the quantized network appears to have greater difficulty in correctly identifying electrons, while maintaining excellent performance on muons, in fact performing slightly better than the original `float32` model for this class.

In conclusion, the results obtained from the post-training quantization demonstrate that a direct quantization of the `float32` model leads to a severe degradation in performance, regardless of the configuration parameters used within the Vitis AI quantization workflow. However, introducing a short phase of manual fine-tuning after quantization significantly improved the network's behaviour. With approximately 19 epochs[6] of fine-tuning, the model recovered much of its original predictive capabil-

[6]The best validation accuracy on validation dataset was obtained after 19 epochs of fine-tuning; however, even after 5 epochs the model had already reached a stable and reasonably high level of perfor-

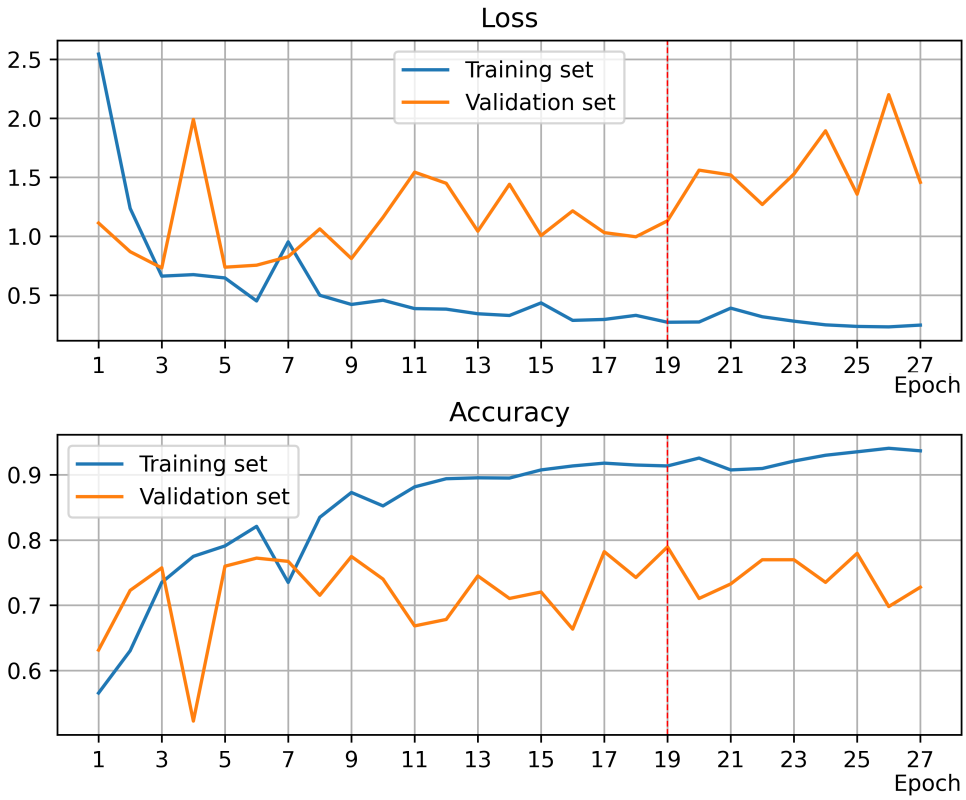


Figure 5.9: Evolution of training and validation metrics during neural network fine tuning after PTQ.

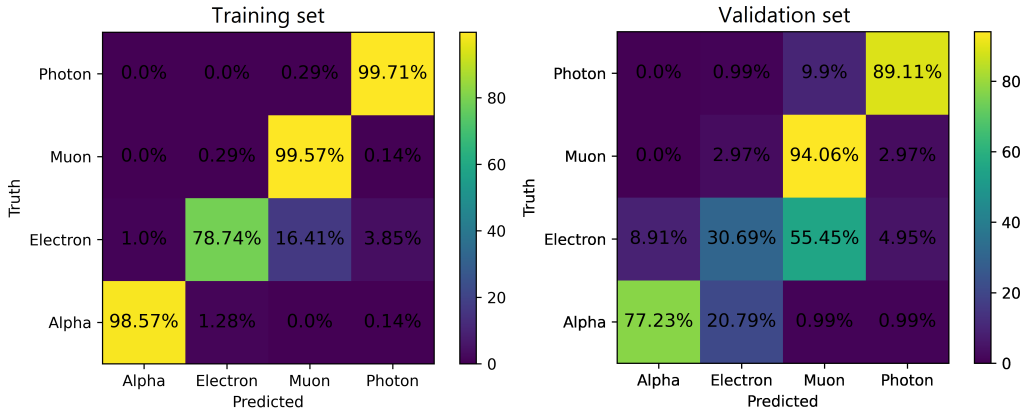


Figure 5.10: Confusion matrices of the network evaluated on the training and validation datasets after PTQ fine tuning.

ity, achieving an accuracy of about 79%. The corresponding confusion matrix confirms this improvement, showing a clear diagonal structure that reflects consistent class discrimination, although still slightly inferior to that of the original `float32` model.

5.5.3 Quantization-Aware Training Performances

While post-training quantization provided a significant improvement thanks to a few epochs of manual fine-tuning, a certain loss of accuracy remained, compared to the original `float32` model. To further minimize this degradation, Quantization-Aware Training (QAT) was employed. QAT incorporates the quantization effects directly into the training process by simulating low-precision arithmetic during both the forward and backward propagation.

To enable Quantization-Aware Training, a standard floating-point model was created from scratch and then converted into a quantization-ready version using the dedicated functions provided by the Vitis AI suite.

```
# Create a new model
model = tpx4NN()
# Create a quantizer
quantizer = vitis_quantize.VitisQuantizer(model)
# Create a QAT model
qat_model = quantizer.get_qat_model(
    init_quant=True,
    input_bit=8,
    weight_bit=8,
    activation_bit=8,
    calib_dataset = train_dataset,
    calib_batch_size=32,
    calib_steps=None,
    include_fast_ft=True,
    fast_ft_epochs=10
)
```

The argument passed to the `get_qat_model()` methods specify that the bit-width for input tensors, weights and activations must be 8-bit integer. Moreover, they specify the training dataset and its batch size. Finally, they enable a short phase of automatic

mance.

fine-tuning after calibration (10 epochs), allowing the model to slightly adjust its parameters to the quantized representation.

The model reached the best accuracy value on the validation set after 14 epochs, with an accuracy of 0.58 and a loss of 2.33. The trends of the two metrics are shown in fig. 5.11.

The confusion matrices in fig. 5.12 show a less evident diagonal structure, in particular when evaluating the validation dataset. Although the overall accuracy remains acceptable, the network fails to correctly classify electrons and muons in more than 50% of the cases. This outcome suggests that starting QAT from randomly initialized weights did not provide a sufficiently strong starting point for the model to adapt to the quantized representation. To address this limitation, a different strategy was adopted: the QAT process was initialized using the pre-trained `float32` model, loaded through the dedicated `Tensorflow` method `tf.keras.models.load_model(filename)`.

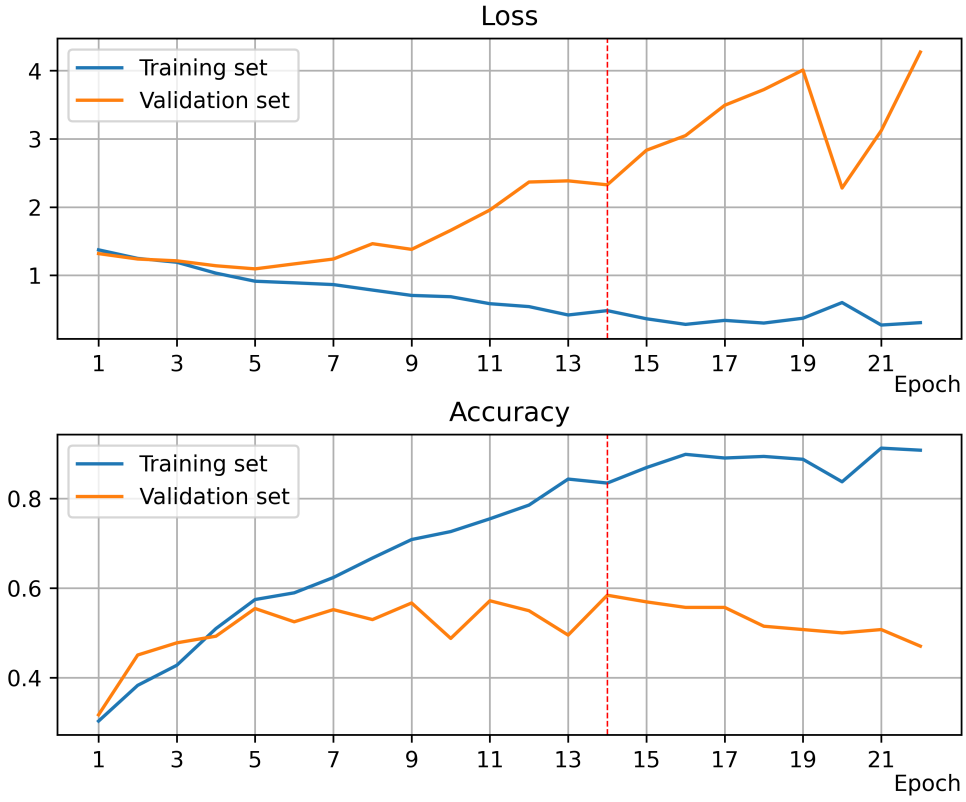


Figure 5.11: Evolution of training and validation metrics during Quantization-Aware Training of the network.

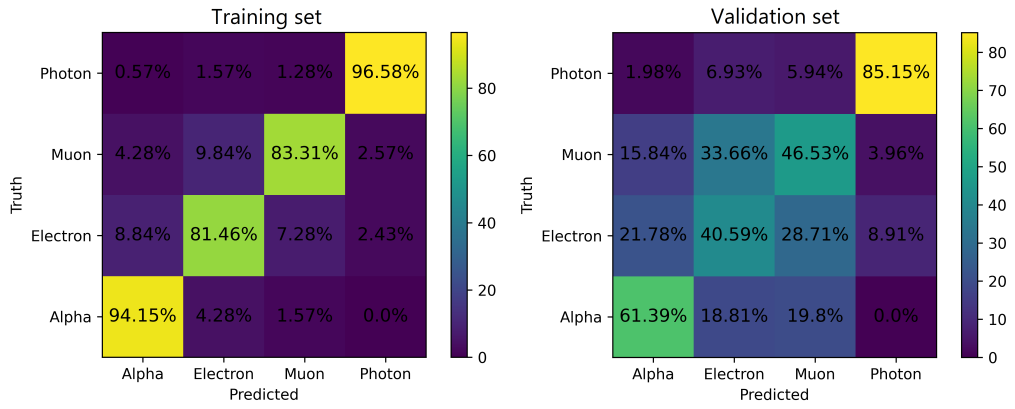


Figure 5.12: Confusion matrices of the network evaluated on the training and validation datasets after Quantization-Aware Training.

As shown in fig. 5.13, the training accuracy rapidly increases, reaching its maximum value after approximately four epochs. A similar trend can be observed in the validation loss, which decreases sharply within the first two epochs and remains consistently low up to around the eleventh epoch, indicating early convergence and stable generalization of the quantized model.

The confusion matrices in fig. 5.14 show a more evident diagonal pattern than the matrices in fig. 5.12. In particular, when QAT is initialized from a newly created model, the network achieves very good training results, with a clear diagonal pattern in the confusion matrix and class accuracies exceeding 80%. However, the corresponding validation matrix shows a significant drop in performance: the model fails to correctly identify electrons and muons in more than 50% of the cases and even alpha particles are classified with only about 60% accuracy. On the opposite, when QAT is performed starting from the pre-trained `float32` model, the training results are slightly less pronounced (in particularly for electrons and muons), but the validation performance improves considerably. In this case, both alpha particles and photons were correctly classified with accuracies above 90%, while electrons and muons remained more challenging but achieved recognition rates exceeding 50%.

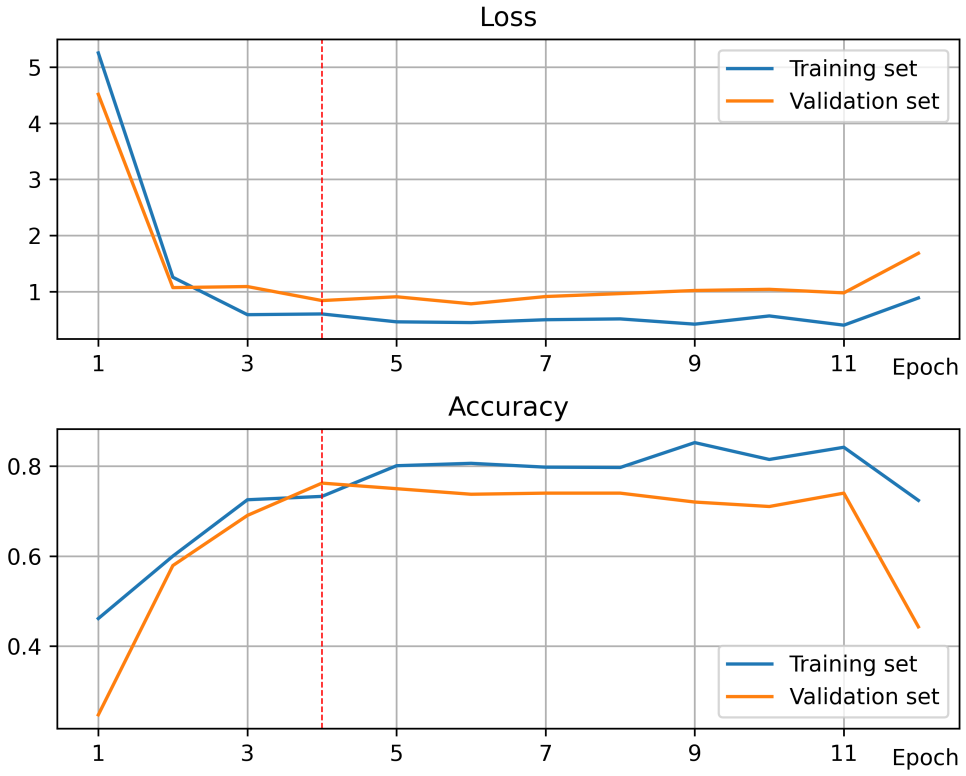


Figure 5.13: Evolution of training and validation metrics during Quantization-Aware Training of the network.

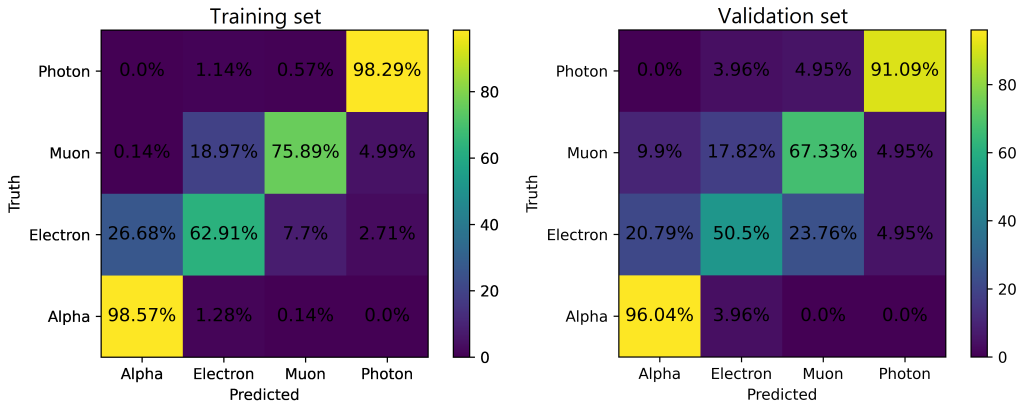


Figure 5.14: Confusion matrices of the network evaluated on the training and validation datasets after Quantization-Aware Training.

5.6. Integration in DataPix4

The neural network described in this chapter 5 will be encapsulated and included in the DataPix4 framework presented in chapter 3. This integration will allow DataPix4 not only to acquire and process raw data from the Timepix4 ASIC but also to perform on-the-fly identification of particle types based on the charge distribution and morphology of the recorded tracks.

A dedicated C++ interface will be developed to handle the interaction between the framework and the neural network. This interface will be responsible for loading the quantized model, preprocessing the data to match the input format expected by the network and executing inference on each detected cluster of pixels. To ensure that the acquisition rate is not compromised, the inference step will be implemented in a separate processing thread, running asynchronously with respect to data acquisition. This design will allow continuous read-out from Timepix4 while simultaneously classifying previously acquired frames.

Conclusion

This thesis presents the design, implementation and application of the DataPix4 framework for the management and operation of the Timepix4 ASIC, a key component of the 4DPHOTON detector. The work addresses the increasing need for software solutions capable of handling high-throughput, high-resolution particle detection systems. DataPix4 (described in details in chapter 3) provides a modular and flexible environment for configuring Timepix4, acquiring and storing data and performing real-time monitoring and basic online analysis. Its architecture supports both slow-control and fast-link communication protocols, enabling adaptability to different control boards and experimental setups. The framework was successfully validated through laboratory characterizations, test-beam measurements and applied physics experiments, confirming its reliability and versatility in a wide range of operating conditions (chapter 4). Moreover, it has been employed across a wide variety of hardware setups, as described in chapter 4, utilizing two different control boards. This demonstrates its flexible design, which can be easily adapted to accommodate changes in the hardware configuration.

The characterization of Timepix4 using DataPix4 allowed precise evaluation of the detector's timing, energy and spatial resolutions. Timing resolutions of 107 ps per pixel, improving to 33 ps with cluster analysis, and energy resolutions better than 1 keV were achieved, demonstrating the effectiveness of the calibration and acquisition

procedures implemented in the software. Furthermore, the 4DPHOTON detector's operation was validated through the observation of Cherenkov rings in test beams, with results consistent with Geant4 simulations.

In parallel, a convolutional neural network was developed to automatically recognize particle tracks from Timepix4 data (chapter 5). The network was specifically designed for the low-resolution, sparse data produced by the ASIC, combining spatial, timing and charge information to distinguish between particle types. Although the integration into DataPix4 is still in progress, this work demonstrates the framework's potential to incorporate machine-learning methods for future real-time data processing and particle recognition.

Overall, the work described in this thesis presents DataPix4 as a comprehensive software solution for Timepix4-based experiments. Its modular structure and the possibility of integration with advanced analysis tools provide a solid foundation for further experimental studies, including high-precision photon detection, X-ray imaging and other applications.

The framework also allows for future upgrades, such as additional online analysis capabilities, support for new control boards and extended machine-learning-based analysis, ensuring that Timepix4 and DataPix4 remain adaptable to evolving research needs.

The work carried out in this thesis represents thus a fundamental part in the research and development for innovative particle detectors designed for the upcoming high energy physics experiment, such as the Upgrade II of the LHCb detector. The need for high performance hardware and software counterpart is in fact highly motivated for a more comprehensive description of the fundamental constituents of the Universe trying to solve the puzzling issues of Standard Model.

Bibliography

[1] The LHCb Collaboration. "The LHCb Detector at the LHC". In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08005. DOI: 10.1088/1748-0221/3/08/S08005. URL: https://dx.doi.org/10.1088/1748-0221/3/08/S08005.

[2] I Belyaev et al. "The history of LHCb". In: *The European Physical Journal H* 46.1 (2021), p. 3.

[3] LHCb Collaboration et al. "The LHCb detector at the LHC". In: *Journal of instrumentation* 3.8 (2008), S08005.

[4] Roel Aaij et al. "The LHCb Upgrade I". In: *JINST* 19.05 (2024), P05065. DOI: 10.1088/1748-0221/19/05/P05065. arXiv: 2305.10515.

[5] *The LHCb Collaboration - CERN Website*. https://home.cern/authors/lhcb-collaboration.

[6] LHCb Collaboration. "LHCb detector performance". In: *International Journal of Modern Physics A* 30.07 (2015), p. 1530022.

[7] Emma Buchanan. "The LHCb vertex locator (VELO) pixel detector upgrade". In: *Journal of Instrumentation* 12.01 (2017), p. C01013.

[8] Thomas Kirn, LHCb Collaboration, et al. “SciFi–A large scintillating fibre tracker for LHCb”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 845 (2017), pp. 481–485.

[9] Marcello Losasso et al. “Tests and field map of lhcb dipole magnet”. In: *IEEE transactions on applied superconductivity* 16.2 (2006), pp. 1700–1703.

[10] A. Papanestis and C. D’Ambrosio. “Performance of the LHCb RICH detectors during the LHC Run II”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 876 (2017). The 9th international workshop on Ring Imaging Cherenkov Detectors (RICH2016), pp. 221–224. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2017.03.009. URL: https://www.sciencedirect.com/science/article/pii/S0168900217303340.

[11] C Abellán Beteta et al. “Calibration and performance of the LHCb calorimeters in Run 1 and 2 at the LHC”. In: *arXiv preprint arXiv:2008.11556* (2020).

[12] A Augusto Alves Jr et al. “Performance of the LHCb muon system”. In: *Journal of Instrumentation* 8.02 (2013), P02022.

[13] Roel Aaij et al. “Allen: A high-level trigger on GPUs for LHCb”. In: *Computing and Software for big Science* 4.1 (2020), p. 7.

[14] T Colombo et al. “The LHCb Online system in 2020: trigger-free read-out with (almost exclusively) off-the-shelf hardware”. In: *Journal of Physics: Conference Series*. Vol. 1085. 3. IOP Publishing. 2018, p. 032041.

[15] The LHCb Collaboration. “LHCb Upgrade II Scoping Document”. In: *Tech. rep.* (2024). URL: https://cds.cern.ch/record/2903094.

[16] R. Calabrese et al. “Performance of the LHCb RICH detectors during LHC Run 2”. In: *Journal of Instrumentation* 17.07 (July 2022), P07013. DOI: 10.1088/1748-0221/17/07/P07013. URL: https://dx.doi.org/10.1088/1748-0221/17/07/P07013.

[17] B. M. Bolotovskii. “Theory of Cherenkov radiation (III)”. In: *Phys. Usp.* 4.5 (1962), pp. 781–811. DOI: 10.1070/PU1962v004n05ABEH003380. URL: https://ufn.ru/en/articles/1962/5/o/.

[18] M Adinolfi et al. “Performance of the LHCb RICH detector at the LHC”. In: *The European Physical Journal C* 73.5 (2013), pp. 1–17.

[19] Edoardo Franzoso and RICH LHCb. “First results on the performance of the upgraded LHCb RICH detectors”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1068 (2024), p. 169689.

[20] The LHCb RICH Collaboration et al. “Performance of the LHCb RICH detector at the LHC”. In: *The European Physical Journal C* C.73 (May 2013), p. 2431. DOI: 10.1140/epjc/s10052-013-2431-9. URL: https://doi.org/10.1140/epjc/s10052-013-2431-9.

[21] Mirco Andreotti et al. “A Fast and Radiation-Hard Single-Photon Counting ASIC for the Upgrade of the LHCb RICH Detector at CERN”. In: *2017 IEEE Radiation Effects Data Workshop (REDW)*. 2017, pp. 1–4. DOI: 10.1109/NSREC.2017.8115435.

[22] M. Fiorini et al. “Single-photon imaging detector with O(10) ps timing and sub-10 μm position resolutions”. In: *Journal of Instrumentation* 13.12 (Dec. 2018), p. C12005. DOI: 10.1088/1748-0221/13/12/C12005. URL: https://doi.org/10.1088/1748-0221/13/12/C12005.

[23] M I Babenkov, V S Zhdanov, and S A Starodubov. “Chevron of microchannel plates as position-sensitive detector for beta spectrometers”. In: *AC* 252 (Nov. 1986). DOI: 10.1016/0168-9002(86)90941-1.

[24] B. B. Wiggins et al. “Optimizing the position resolution of a Z-stack microchannel plate resistive anode detector for low intensity signals”. In: *Review of Scientific Instruments* 86.8 (Aug. 2015), p. 083303. ISSN: 0034-6748. DOI: 10.1063/1.4927457. URL: https://doi.org/10.1063/1.4927457.

[25] R. Bolzonella et al. “Development and characterization of hybrid MCP-PMT with embedded Timepix4 ASIC used as pixelated anode”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1082 (2026), p. 170965. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2025.170965. URL: https://www.sciencedirect.com/science/article/pii/S0168900225007673.

[26] *The Medipix4 Collaboration - CERN Website*. https://medipix.web.cern.ch/medipix4.

[27] R. Ballabriga, M. Campbell, and X. Llopart. "An introduction to the Medipix family ASICs". In: *Radiation Measurements* 136 (2020), p. 106271. ISSN: 1350-4487. DOI: https://doi.org/10.1016/j.radmeas.2020.106271. URL: https://www.sciencedirect.com/science/article/pii/S1350448720300354.

[28] X. Llopart et al. "Timepix4, a large area pixel detector readout chip which can be tiled on 4 sides providing sub-200 ps timestamp binning". In: *Journal of Instrumentation* 17.01 (Jan. 2022), p. C01044. DOI: 10.1088/1748-0221/17/01/C01044. URL: https://dx.doi.org/10.1088/1748-0221/17/01/C01044.

[29] *Aurora 64B/66B Protocol Specification*. https://docs.amd.com/v/u/en-US/aurora_64b66b_protocol_spec_sp011.

[30] *Timepix Applications - CERN Website*. URL: https://medipix.web.cern.ch/applications.

[31] *Timepix at the International Space Station*. URL: https://medipix.web.cern.ch/timepix-international-space-station.

[32] Nicholas N. Stoffle et al. "HERA: A Timepix-based radiation detection system for Exploration-class space missions". In: *Life Sciences in Space Research* 39 (2023). Radiation in human space exploration: Detectors and measurements, today and tomorrow, pp. 59–66. ISSN: 2214-5524. DOI: https://doi.org/10.1016/j.lssr.2023.03.004. URL: https://www.sciencedirect.com/science/article/pii/S2214552423000226.

[33] *First 3D colour X-ray of a human using CERN technology*. URL: https://home.cern/news/news/knowledge-sharing/first-3d-colour-x-ray-human-using-cern-technology.

[34] *InsightART helped to analyse the lost Raphael's painting*. https://insightart.eu/2020/09/11/insightart-helped-to-authenticate-the-lost-raphaels-painting/.

[35] *MiniPIX TPX3*. URL: https://advacam.com/camera/minipix-tpx3/.

[36] Viola Cavallini et al. “DataPix4: A C++ framework for Timepix4 configuration and read-out”. In: *Computer Physics Communications* 314 (2025), p. 109658. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2025.109658. URL: https://www.sciencedirect.com/science/article/pii/S0010465525001602.

[37] *User Datagram Protocol*. RFC 768. Aug. 1980. DOI: 10.17487/RFC0768. URL: https://www.rfc-editor.org/info/rfc768.

[38] Nicolò Vladi Biesuz. *A flexible electronics and DAQ system for the Timepix4 and Medipix4 ASICs*. https://agenda.infn.it/event/44098/contributions/258206/.

[39] *Fundamental types - C++ References*. https://en.cppreference.com/w/cpp/language/types.

[40] Rene Brun and Fons Rademakers. “ROOT — An object oriented data analysis framework”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 389.1 (1997). New Computing Techniques in Physics Research V, pp. 81–86. ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(97)00048-X.

[41] “ROOT TTree Data Structure”. In: *ROOT Documentation* (). URL: https://root.cern.ch/doc/master/classTTree.html.

[42] “ROOT TBranch Data Structure”. In: *ROOT Documentation* (). URL: https://root.cern.ch/doc/master/classTBranch.html.

[43] “ROOT TFile”. In: *ROOT Documentation* (). URL: https://root.cern.ch/doc/master/classTFile.html.

[44] R. Bolzonella et al. “Timing resolution performance of Timepix4 bump-bonded assemblies”. In: *Journal of Instrumentation* 19.07 (July 2024), P07021. DOI: 10.1088/1748-0221/19/07/P07021. URL: https://dx.doi.org/10.1088/1748-0221/19/07/P07021.

[45] *std::map - C++ References*. https://en.cppreference.com/w/cpp/container/map.html.

[46] *tkinter - Python interface to Tcl/Tk*. URL: https://docs.python.org/3/library/tkinter.html.

[47] *ttkbootstrap Docs*. URL: https://ttkbootstrap.readthedocs.io/en/latest/.

[48] Riccardo Bolzonella. *Development of an innovative hybrid single-photon imaging detector based on a Timepix4 CMOS ASIC embedded as pixelated anode*. https://iris.unife.it/handle/11392/2586230. 2024.

[49] P. Delogu et al. "Validation of Timepix4 energy calibration procedures with synchrotron X-ray beams". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1068 (2024), p. 169716. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2024.169716. URL: https://www.sciencedirect.com/science/article/pii/S0168900224006429.

[50] Nicolò Vladi Biesuz et al. "Review of INFN activities on characterization and applications of hybrid pixel detectors based on Timepix4 ASIC". In: *Frontiers in Sensors* Volume 6 - 2025 (2025). ISSN: 2673-5067. DOI: 10.3389/fsens.2025.1585385. URL: https://www.frontiersin.org/journals/sensors/articles/10.3389/fsens.2025.1585385.

[51] K. Heijhoff et al. "Timing performance of the Timepix4 front-end". In: *Journal of Instrumentation* 17 (July 2022), P07006. DOI: 10.1088/1748-0221/17/07/P07006.

[52] V. Mazzini et al. "Characterization of a hybrid photon-counting detector based on Timepix4 with a quasi-monochromatic source for spectral X-ray imaging applications". In: *Il Nuovo Cimento C* 4 (2024). DOI: 10.1393/ncc/i2025-25244-5.

[53] P Baldelli et al. "A prototype of a quasi-monochromatic system for mammography applications". In: *Physics in Medicine and Biology* 50.10 (Apr. 2005), p. 2225. DOI: 10.1088/0031-9155/50/10/003. URL: https://dx.doi.org/10.1088/0031-9155/50/10/003.

[54] S. Velardita et al. "First characterization of a GaAs-based Timepix4 detector assembly for X-ray imaging applications". In: *Journal of Instrumentation* 21.01 (Jan. 2026), p. C01023. DOI: 10.1088/1748-0221/21/01/C01023. URL: https://doi.org/10.1088/1748-0221/21/01/C01023.

[55] Tromba G. et al. “The SYRMEP Beamline of Elettra: Clinical Mammography and Bio-medical Applications”. In: *AIP Conf. Proc.* 1266 (June 2010). DOI: https://doi.org/10.1063/1.3478190.

[56] *SPIDR4 Website*. URL: https://spidr4.nikhef.nl/.

[57] *The H8 Secondary Beam Line of EHN1/SPS*. URL: https://sba.web.cern.ch/sba/BeamsAndAreas/H8/H8_presentation.html.

[58] AJ Peters, EA Sindrilaru, and G Adde. “EOS as the present and future solution for data storage at CERN”. In: *Journal of Physics: Conference Series* 664.4 (Dec. 2015), p. 042042. DOI: 10.1088/1742-6596/664/4/042042. URL: https://doi.org/10.1088/1742-6596/664/4/042042.

[59] G. Romolini et al. *Hybrid MCP-PMT characterisation on a testbeam with Cherenkov setup*. 2026. arXiv: 2601.22713 [physics.ins-det]. URL: https://arxiv.org/abs/2601.22713.

[60] A. Castoldi et al. “Energy-resolved X-ray radiography with controlled-drift detectors at Sincrotrone Trieste”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 510.1 (2003). Proceedings of the 2nd International Symposium on Applications of Particle Detectors in Medicine, Biology and Astrophysics, pp. 57–62. ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(03)01679-6. URL: https://www.sciencedirect.com/science/article/pii/S0168900203016796.

[61] Juergen Fornaro et al. *Dual- and multi-energy CT: approach to functional imaging*. Apr. 2011. DOI: 10.1007/s13244-010-0057-0. URL: https://doi.org/10.1007/s13244-010-0057-0.

[62] Robert Speller Alessandro Olivo. *A coded-aperture technique allowing x-ray phase contrast imaging with conventional sources*. Aug. 2007. DOI: https://doi.org/10.1063/1.2772193.

[63] L. Brombal, F. Arfelli, and R.H. et al Menk. *PEPI Lab: a flexible compact multimodal setup for X-ray phase-contrast and spectral imaging*. Mar. 2023. DOI: 10.1038/s41598-023-30316-5. URL: https://www.nature.com/articles/s41598-023-30316-5.

[64] *TIGRE: Tomographic Iterative GPU-based Reconstruction Toolbox*. URL: https://github.com/CERN/TIGRE.

[65] Vittorio Di Trapani, Luca Brombal, and F. Brun. "Multi-material spectral photon-counting micro-CT with minimum residual decomposition and self-supervised deep denoising". In: *Optics Express* 30 (Nov. 2022), pp. 42995–43011. DOI: 10.1364/OE.471439.

[66] Sergei Antipov et al. "IOTA (Integrable Optics Test Accelerator): Facility and Experimental Beam Physics Program". In: *JINST* 12.03 (2017), T03002. DOI: 10.1088/1748-0221/12/03/T03002. arXiv: 1612.06289 [physics.acc-ph].

[67] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521 (2015). DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[68] *Vitis AI*. URL: https://www.amd.com/en/products/software/vitis-ai.html.

[69] Alvah Ivor Bocknor Wisdom Samuels. "One-Hot Encoding and Two-Hot Encoding: An Introduction". In: (Jan. 2024). DOI: 10.13140/RG.2.2.21459.76327.

[70] *Vitis AI - Supported Operations and APIs*. URL: https://docs.amd.com/r/en-US/ug1414-vitis-ai/vai_q_tensorflow2-Supported-Operations-and-APIs.

[71] Christian Szegedy et al. "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[72] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: Dec. 2014. arXiv: 1412.6980 [cs.LG].

[73] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. New York: Springer, 2006. ISBN: 978-0-387-31073-2.

[74] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[75] Xilinx Inc. *Vitis AI: A Unified AI Inference Framework for Xilinx Platforms*. Tech. rep. White Paper, Available at https://www.xilinx.com/support/documentation/white_papers/wp504-vitis-ai.pdf. Xilinx, 2020.

[76] AMD. *Vitis AI User Guide*. Advanced Micro Devices, Inc. 2024. URL: https://vitisai.docs.amd.com/en/gen-1/index.html.