EPJ Web of Conferences **295**, 02028 (2024) https://doi.org/10.1051/epjconf/202429502028
CHEP 2023

The CMS Inner Tracker DAQ system for the High Luminosity upgrade of LHC: from single-chip testing, to large-scale assembly qualification

Mauro Emanuele Dinardo[1,2,*]

[1]Università degli Studi di Milano - Bicocca (Italy)
[2]On Behalf of the CMS Collaboration.

Abstract. To cope with the challenging environment of the planned high luminosity upgrade of the Large Hadron Collider (HL-LHC), scheduled to start operation in the late 2020s, CMS will replace its entire tracking system. The requirements for the tracker are largely determined by the long operation time of 10 years with a peak instantaneous luminosity of up to $7.5 \times 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ in the ultimate performance scenario. In particular, the Inner Tracker (IT) is being completely redesigned featuring a front-end chip capable to deal with hit rates of up to $3.38\,\mathrm{GHz/cm}^2$. The communication between the front-end and the back-end electronics occurs through an optical link based on a custom Low-power Gigabit Transceiver which sends data at 10 and 2.5 Gbps on the uplink and downlink, respectively. The number of pixels has been increased by a factor of 6 with respect to the present detector, resulting in an unprecedented number of channels of about two billion, and covering a pseudorapidity region up to 4. This represents a challenging requirement for the data acquisition system since it needs to efficiently configure, monitor, and calibrate them. A dedicated data acquisition system, written in C++ and based on a custom μTCA board to handle trigger, data, and detector control, equipped with an FPGA, was developed to fully test and characterize the IT modules on a bench and with beam tests. In this note, we will describe the system architecture and its scalability to the final system which will be based on custom back-end boards equipped with FPGAs and CPUs.

1 Introduction

In the coming years, the LHC accelerator will be upgraded to enable instantaneous peak luminosities of $7.5 \times 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, the so-called High Luminosity LHC (HL-LHC) [1], which is expected to run at a center-of-mass energy of 14 TeV and with a bunch spacing of 25 ns. This will allow the CMS experiment [2] to collect an integrated luminosity of the order of $400\,\mathrm{fb}^{-1}$ per year and up to $4\,000\,\mathrm{fb}^{-1}$ during the HL-LHC projected lifetime of ten years. About 200 overlapping pp collisions per proton bunch crossing (referred to as "pileup") are expected for the HL-LHC peak luminosity. The HL-LHC upgrade is accompanied by an upgrade program of the CMS experiment, to maintain the excellent performance of the detector and to allow physicists to fully profit from the HL-LHC potential, despite the challenging

*e-mail: mauro.dinardo@cern.ch

radiation levels and operating conditions. To guarantee high performance, the entire CMS silicon tracking system will be replaced to meet the requirements of radiation tolerance, low occupancy, low material budget, and high data transfer throughput for the Level-1 (L1) track finder. A detailed description of the so-called "Phase-2" upgrade can be found in Ref. [2].

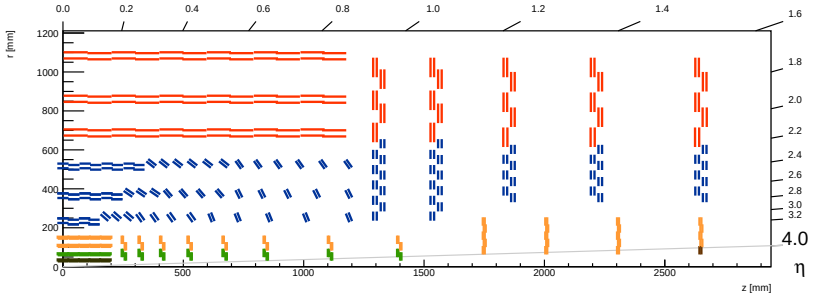

Figure 1. One-quarter of the CMS Phase-2 tracker in the $r - z$ plane. IT modules with planar silicon sensors are shown in green for modules with 1×2 and orange for modules with 2×2 readout chips. Modules with 1×2 readout chips and silicon sensors in 3D technology are shown in brown. OT modules with Pixel-Strip (PS) and Strip-Strip (2S) sensors are shown in blue and red, respectively [2].

The CMS Phase-2 tracker detector is divided into two main partitions: the Inner Tracker (IT) and the Outer Tracker (OT) as shown in Fig. 1. The IT is composed of about 4 350 modules, corresponding to an area of approximately 5 m^2 of silicon, with a pseudorapidity coverage up to $|\eta| = 4$. The OT is instrumented with so-called p_T-modules with two closely spaced silicon sensors. By correlating hits on the two sensors, the front-end chips form local track segments called stubs. The modules select stubs consistent with tracks having a transverse momentum $p_T > 2$ GeV and send the information to the L1 track finder at a rate of 40 MHz. Simulations indicate that this design will allow keeping excellent tracking capabilities over a wide pseudorapidity region despite the high pileup. The tracker data acquisition system (DAQ) plays a crucial role to meet these extraordinary requirements. In particular the DAQ will enable constant monitoring of the performance and will help to ensure stability over time.

2 Inner Tracker System

The main challenges for the IT DAQ arise from the high particle rate and number of channels. For the innermost layer, located at a radius of about 3 cm, the rate will be as high as 3.38 GHz/cm^2. A total of 14 000 readout chips with approximately 2 billion readout channels will have to be calibrated and monitored. To address both the radiation resistance and the hit rate requirements, a CMS and ATLAS common development for the pixel readout chip (ROC) was carried out within the RD53 Collaboration [3]. The first prototype, RD53A, was half of the size of the final chip, and it featured three pre-amplifier architectures, among which CMS choose the "linear" one [4]. The second prototype, RD53B, has a design close to the final version. It features a 336×432 pixel matrix of $50 \times 50\,\mu m^2$ pixel size, which ensures an occupancy below 0.1%. The IT sensor modules, with a pitch of $25 \times 100\,\mu m^2$, host 1×2 or 2×2 ROCs. The main characteristics of the ROC are: 65 nm CMOS technology; global threshold with per-pixel 5-bit threshold trimming and zero suppression; Time-over-Threshold (ToT) hit charge measurement with a 4-bit ADC; internal charge injection circuit for self-calibration; dual-slope gain; data compression by binary tree encoding, allowing to

reduce the data volume by about 30%; data stream toward the chip with a custom protocol at 160 Mbps for clock, trigger, commands, and configuration data; data stream from the chip with the Aurora protocol [5] at 1.28 Gbps per lane for hit and monitoring data.

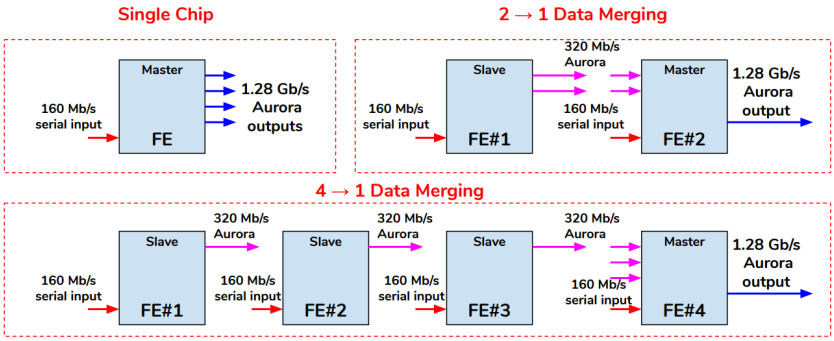

Figure 2. Schematic of the IT on-chip data merging capabilities and the possible combinations.

To reduce the amount of data to be transmitted off-chip, pixels are read out in groups of 4 by adjacent columns, allowing to reduce the number of bits needed to address each of them. This results in a reduction of data volume by about a factor of 2. To handle the large amount of data to be transmitted, each ROC is equipped with four Aurora lanes. This allows addressing the bandwidth requirements for the innermost layer. The chip will implement the capability of merging data from multiple ROCs on the module, reducing the number of electrical lines, e-links, to transmit the data, hence reducing the material budget (Fig. 2).

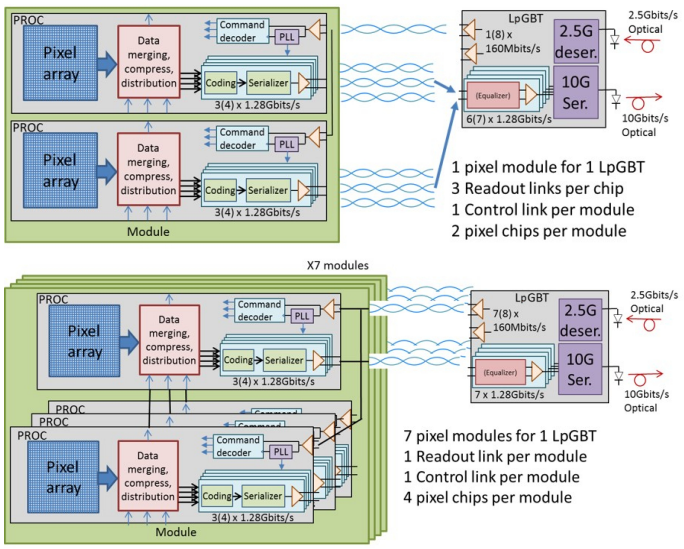

Figure 3. Readout principle of IT modules via e-links. Top: one-pixel module with two pixel chips connected to one LpGBT, using three readout e-links per-pixel chip (inner barrel layer). Bottom: seven pixel modules with four pixel chips per module connected to one LpGBT, using local data merging between chips on the same module used in the outer layers [2].

Data from each module will be transferred to the Low power GigaBit Transceiver (LpGBT) hosted on a dedicated PCB, called "portcard", with up to 6 e-links per module. The data are converted into an optical signal by the Versatile TxRx plus electro-opto transducer (VTRx+), and sent to the back-end electronics at 10.24 Gbps. Clock, fast-commands, and programming are sent from the back-end electronics to the LpGBT via a 2.56 Gbps link, cf. Fig. 3. Both LpGBT and VTRx+ are CERN-developed ASICs for the HL-LHC upgrade.

3 Setup for the Inner Tracker Module Testing

The module testing and characterization requires a dedicated and versatile setup which can be easily installed, transported and connected to any Linux computer. Indeed a minimal test setup is composed by a Linux computer, a custom μTCA Data, Trigger, and Control board (μDTC), and one or more modules. The computer runs the software which communicates via IPbus protocol [7] with the μDTC board that features a Xilinx Kintex 7 FPGA, two FMC connectors, and a DDR3 RAM of 4 Gbit. A commercial Digital I/O FMC board, mounted on the μDTC, provides external trigger and clock signals, and a custom-made FMC board allows to electrically connect the module(s) via a display port cable. The same system can be used for the optical readout by simply exchanging the FMC board. The main tasks of the firmware running on the FPGA are to encode/decode chip commands, send a number of triggers and injections, store data in a memory buffer, handle the external clock and the trigger handshake, and abstract any communication details to support different varieties of front-end hardware. The current system supports both the RD53A and RD53B readout chips [3], which are polymorphically handled internally.

4 Current Status of the Software

The current implementation of the DAQ software called "Ph2-ACF" (Phase-2 Acquisition & Control Framework) [6] is written in C++. The software is developed for module testing, but can meet the requirements of the final system, as explained in Sec. 5. The Ph2-ACF's main components and their relations are shown in Fig. 4, and summarized here:

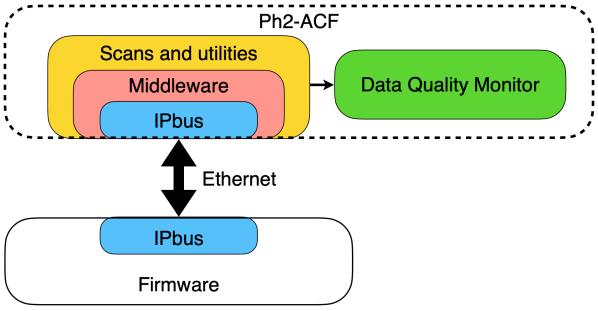

Figure 4. Schematic showing the Phase-2 Acquisition & Control Framework software layers.

- Middleware API layer which wraps the firmware calls and handshakes into C++ functions. It contains object-oriented libraries for sending commands and receiving data, and libraries describing the system components (modules, portcards, μDTCs), called "interface" and "description" classes, respectively;

- Various scans and utilities to read/write configuration files, perform calibrations, and monitor the hardware. These also contain a template data structure, called "Container", which reflects the relations between the different hardware components;
- Data Quality Monitor that collects the data and represents them in a user-friendly way.

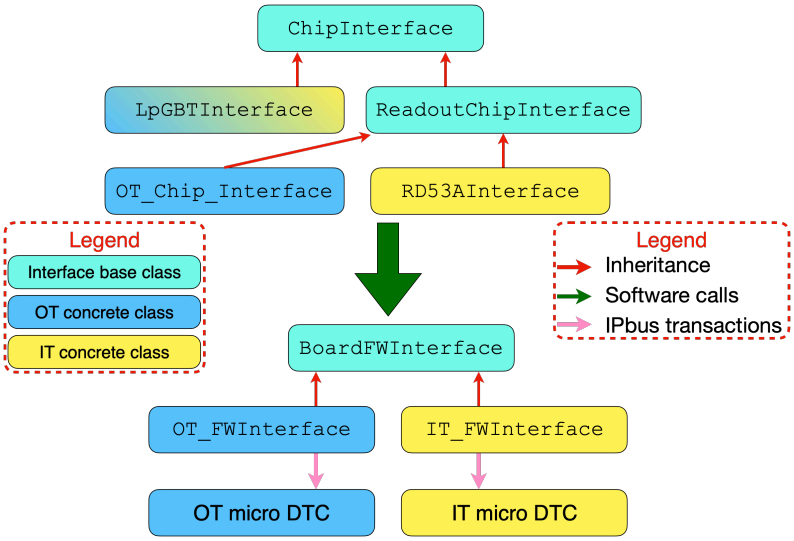

Figure 5. Schematic showing the relations among the different interface classes.

The relations between interface and abstract classes are represented in Fig. 5. The yellow and blue shading represents the concrete implementations for the IT and OT, respectively, and cyan represents the abstract layers. The `ChipInterface` class serves as a base class for all ASICs in the system with direct implementations for the auxiliary chips like the LpGBT. For the readout ASICs, a derived class, called `ReadoutChipInterface`, provides additional functionality only needed for this chip type. All interface classes make calls through the `BoardFWInterface` to the concrete implementation of the firmware class, which performs the software-to-firmware command translation.

The description class relations are represented in Fig. 6, and, just as for the interface classes, the yellow and blue shading represent the concrete implementations for the IT and OT, respectively, They allow for an abstract representation of the hardware hierarchy. This hierarchy is mapped into a `DetectorContainer` structure instantiated and filled at configuration time. The description classes give the type of each layer.

Ph2-ACF exploits multi-threading to perform several functionalities such as run calibrations, read monitoring data (temperatures, voltages, and currents) and decoding of data readout from μDTC memory banks. The hardware race is avoided with the standard `C++` mutex variables and lock guards.

4.1 Calibrations

Ultimately a calibration sequence has to iterate on one or more chip registers, inject charge and send triggers (handled at firmware level), record the response (hit/no-hit, pulse height), run the sequence on a subset of channels and iterate over all subsets. Calibrations need to

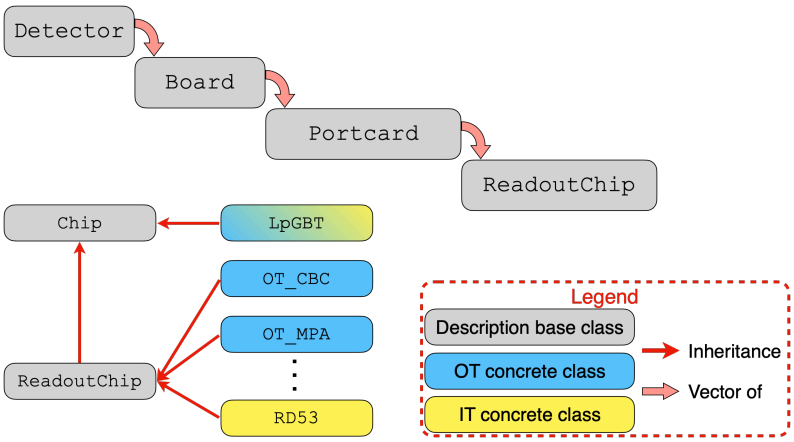

Figure 6. Schematic showing the relation among the different description classes. This relation is mapped into a `DetectorContainer` structure based on `std::vector`.

store information (efficiency, register values, histograms) following the detector's hierarchy. The hierarchy can be copied from the object of type `DetectorContainer` into an object of type `DetectorDataContainer<T>`, featuring at each node an object of type `T`. As shown in Fig. 7 every calibration is a C++ class inheriting from `Tool`, which provides the basic functionalities for looping over the hierarchical structure of the detector.

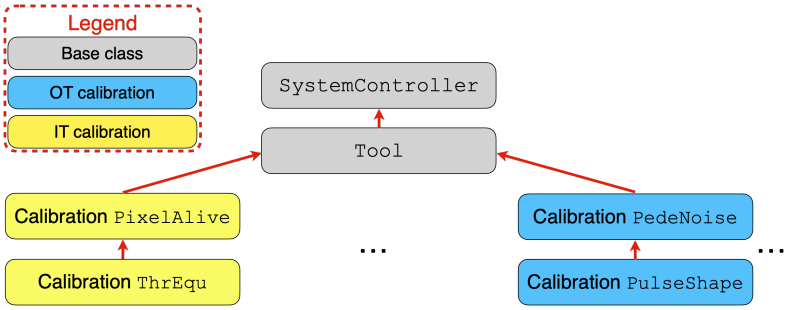

Figure 7. Schematic describing the hierarchy among calibration classes.

Some calibrations can be considered "basic", such as `PixelAlive` which performs occupancy measurements exploiting the internal charge injection circuit of the chip, `SCurve` which measures the threshold and noise, and `Gain` which measures the ToT as a function of the injected charge. Other calibrations are instead extensions of the "basic" ones, and consequently, inherit from them, *e.g.* `ThresholdAdjustment` which adjusts the threshold to a certain value of charge and which uses the occupancy, *i.e.* `PixelAlive`, for the measurement of the threshold position.

4.2 External Devices

The Ph2-ACF software also allows to control external devices, such as power supplies and oscilloscopes. This feature can be useful both to implement the proper turn-on/off sequence,

synchronized with the software, or to perform scans, for example versus a certain voltage, or to monitor electrical quantities. Figure 8 shows the hierarchy of interface and controller classes and their functionalities.

- `ExtDeviceController`: is a TCP/IP server that dispatches commands, such as "TurnOn", "TurnOff", "GetDeviceConnected", "GetStatus", to the external devices;
- `ExtDeviceInterface`: is a TCP/IP client that sends requests to the server to execute commands by the external devices (mainly "TurnOn" and "TurnOff");
- `ExtDeviceMonitorController`: is a TCP/IP client that sends requests to the server to execute commands by external devices (polling external device status, *e.g.* currents and voltages).

Concurrency in accessing the same device by different processes is handled by the TCP/IP client/server protocol. A user who wants to add a new device has to simply provide the concrete class that implements the basic commands of the corresponding abstract class.

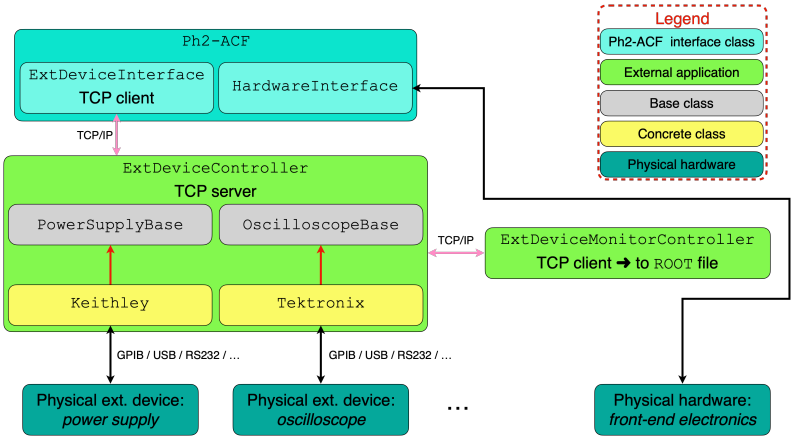

Figure 8. Schematic describing the hierarchy among the different classes handling external devices. The `ExtDeviceController` TCP/IP server receives commands from either the `ExtDeviceInterface` or from the `ExtDeviceMonitorController`, and dispatches them to the proper physical device through the GPIB (USB or RS232) protocol. The `ExtDeviceController` instantiates an object of such a class and handles the TCP/IP communication transparently to the user.

5 A glimpse of the final system

Figure 9 shows a schematic of the final IT readout chain including the ATCA-compliant DTC boards. These will be equipped with a co-processor, System-on-Chip, based on a Intel ×86 Comm-Express or a Xilinx Zynq mezzanine, which will be running a key component of the DAQ, namely "Scans and utilities" from Fig. 4. Each DTC will handle a subset of the IT channels and will feature up to two Xilinx Ultrascale+ FPGAs, and a local memory storage element. Both the IT and the OT readout chains are shown for comparison. The two chains have similar optolinks and back-end boards.

Though the current priority in the development of the software is characterizing and qualifying electronic components (prototypes and final objects) and checking that the performance of the production components are within the specifications, the software is being written targeting functionalities of the final system by guarantying scalability. To achieve this, the

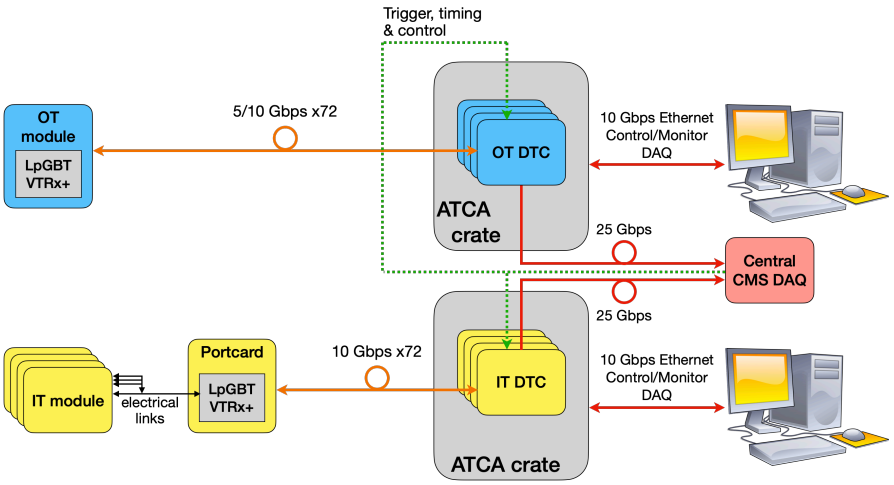

Figure 9. Schematic of the IT and OT readout chains for comparison. The two systems include similar optical readout chain and back-end Data, Trigger and Control boards (DTC).

Ph2-ACF can also be run on a distributed system where the users interact with the the local computer, *i.e.* the Control/Monitor DAQ from Fig. 9, which sends state machine commands through the network to the remote computer(s), *i.e.* the DTC(s). The component of the DAQ, *i.e.* Scans and utilities, running on the remote computer decodes the commands and launches the corresponding calibration, which just needs to implement the basic memember functions: `Configure`, `Start`, `Stop`, `Pause`, and `Resume`. All calibrations have been implemented to use as few resources as possible given the limited resources available on the remote computer. The plotting is performed on the local computer where a Data Quality Monitor object fills the histograms.

6 Summary

In this contribution we presented the overall architecture and capabilities of the CMS Phase-2 Inner Tracker DAQ system for module testing (Ph2-ACF). The software and firmware of this system are still under development, nonetheless, they are already in a mature stage to support users to characterize the modules both on a test bench and with test beams. Ph2-ACF is being designed with the aim of handling a distributed system with a large number of modules, similar to the final system.

References

[1] G. Apollinari et al., *"High-Luminosity Large Hadron Collider (HL-LHC): Preliminary Design Report"*, **DOI:10.5170/CERN-2015-005** (2015).
[2] CMS Collaboration, *"The Phase-2 Upgrade of the CMS Tracker"*, **DOI:10.17181/CERN.QZ28.FLHW** (2017).
[3] RD53 Collaboration, *"Development of pixel readout integrated circuits for extreme rate and radiation"*, **CDS:CERN-LHCC-2013-008, LHCC-P-006** (2013).

[4] The Tracker Group of the CMS Collaboration, *"Comparative evaluation of analogue front-end designs for the CMS Inner Tracker at the High Luminosity LHC"*, Journal of Instrumentation, (2021) **16** P12014, **DOI:10.1088/1748-0221/16/12/P12014**.

[5] Aurora protocol documentation web site: https://www.xilinx.com/products/intellectual-property/aurora64b66b.html

[6] M. E. Dinardo at al., *"The data acquisition system to test and characterise the pixel detector modules of the CMS inner tracker for the High Luminosity upgrade of LHC"*, IEEE NSS MIC (2021, Japan), **DOI:10.1109/NSS/MIC44867.2021.9875778**.

[7] C. Ghabrous Larrea et al., *"IPbus: a flexible Ethernet-based control system for xTCA hardware"*, JINST **10** 02 C02019, (2015), **DOI:10.1088/1748-0221/10/02/C02019**.