INTEGRATION OF AN MPSoC-BASED ACQUISITION SYSTEM INTO THE CERN CONTROL SYSTEM

E. Balcı*, I. Degl'Innocenti, M. Gonzalez Berges, S. Jackson, M. Krupa
CERN, Geneva, Switzerland

Abstract

Future generations of Beam Instrumentation systems will be based on Multiprocessor System on Chip (MPSoC) technology. This new architecture will allow enhanced exploitation of instrumentation signals from CERN's accelerator complex, and has thus been chosen as the next platform for several emerging systems. One of these systems, for the HL-LHC BPM (High-Luminosity LHC Beam Position Monitors), is currently at a prototyping stage, and it is planned to test this prototype with signals from real monitors in CERN's accelerators during 2023. In order to facilitate the analysis of the prototype's performance, a strategy to integrate the setting, control and data acquisition within CERN's accelerator control system has been developed.

This paper describes the exploration of various options and eventual choices to achieve a functional system, covering all aspects from data acquisition from the gateware, through to eventual logging on the accelerator logging database. It also describes how the experiences of integrating this prototype will influence future common strategies within the accelerator sector, highlighting how specific problems were addressed, and quantifying the performance we can eventually expect in the final MPSoC-based systems.

INTRODUCTION

To gain experience with System-on-Chip (SoC) technology and validate its suitability, a vertical slice of the new HL-LHC BPM system was developed. This vertical slice was designed to address the following requirements:

- configuration of the acquisition parameters;
- signal processing in alignment with the acquisition parameters;
- the ability to trigger acquisitions based on accelerator timing events, manual triggers, and timers;
- storage and display of both the acquisition data and system settings.

The chosen hardware is the development board ZCU208 [1], which includes as main component the Xilinx Zynq UltraScale+ RFSoC Gen3 [2]. The RFSoC is a System-on-Chip integrating a Processing System (PS), with Programmable Logic (PL) and high-resolution fast Analog to Digital Converters (ADC) and Digital to Analog Converters (DAC). The PS includes an Application Processing Unit (APU) with 4 Arm Cortex-A53 CPUs and a Real-Time Processing Unit (RPU) with 2 Arm Cortex-R5 CPUs. The PL includes UltraScale+ field programmable gate array (FPGA) fabric, which can be configured through hardware description language to implement fast parallel signal processing, and it is fundamental in this chip to interface the fast RF converters data stream. The RFSoC was identified as potential candidate for the HL-LHC Beam Position Monitor (BPM) electronics [3]. With the requirements listed above in mind, various components available on the RFSoC are harnessed. The vertical slice consists of the signal processing and acquisition logic within the PL, communication software in the PS, and a Front End Server Architecture (FESA) device running on a Front End Computer (FEC) [4].

In addition, we implemented the necessary communication protocols and conducted lab tests and tests in the Super Proton Synchrotron (SPS) accelerator in order to verify the prototype and quantify its performance. The decisions regarding communication protocols have been made with the understanding that this is a prototype, and this selection may not be final. A new iteration of the prototype, incorporating future recommendations from the sector and group, is planned. The current state of the prototype will be further discussed in the following subsections.

SYSTEM OVERVIEW

Figure 1 provides an overview of the system, highlighting its software and gateware components. The analog signals from BPMs are directed to the ADCs on the RFSoC board. The processing of these signals and the implementation of acquisition logic are carried out within the PL, as detailed in the gateware subsection.

A Linux based Operating System(OS) is prepared with the necessary modifications to the device tree in order to accommodate the interactions with the PS DDR memory, utilizing the Petalinux Tools. It is deployed to the APU of the PS. Decoupling of the operating system and the PL configuration is achieved through configuring the PL while the OS is running, through a start script added as a user application. Which is important in order to ensure that a time consuming task such as building an OS image is avoided for each change in the PL configuration.

Configuration of acquisition parameters and data transfer are managed within the APU. The integration of acquisition data and parameters into the CERN control system, as well as the logging of data, are achieved through a FESA device, which is further elaborated upon in its respective subsection.

Gateware

The gateware consists of the RF ADC cores, the logic managing the data stream, the logic selecting the samples to be stored (depending on the acquisition settings), and the state machines controlling the acquisition operations

* elif.balci@cern.ch

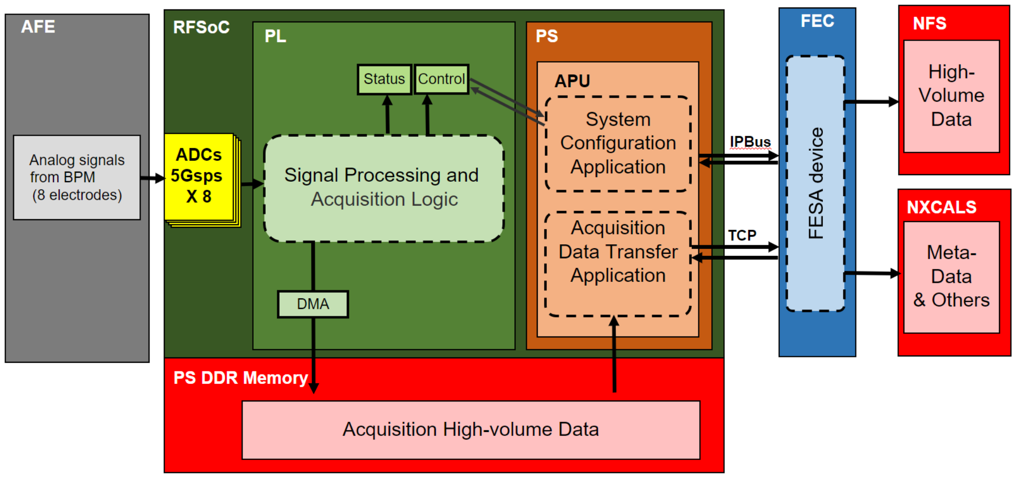
AFE
Analog signals from BPM (8 electrodes)
RFSoC
ADCs 5Gsps X 8
PL
Status
Control
Signal Processing and Acquisition Logic
DMA
PS
APU
System Configuration Application
Acquisition Data Transfer Application
PS DDR Memory
Acquisition High-volume Data
IPBus
TCP
FEC
FESA device
NFS
High-Volume Data
NXCALS
Meta-Data & Others

Figure 1: Box diagram of the prototype, visualizing the components of the system and their interaction.

and the configuration of the Direct Memory Access (DMA) core to store the acquired data. All the 8 ADC cores were configured to sample at the maximum sampling speed, i.e. 8 GSps. The ADCs are divided into two independent and identical 4 ADCs acquisition systems.

The system is designed to make regular acquisitions so it relies on the presence of a periodic trigger which can be an external signal or a synthetic trigger generated within the logic. The parameters of the acquisition are:

- the periodic trigger source (internal or external) and the period in number of clock cycles for the internal trigger;
- the two delays in clock cycles to be applied to the acquisition start trigger and to the periodic trigger;
- the number of periods to be acquired;
- the number of windows in each acquisition (up to 4);
- the size of each window defined as first sample and last sample to acquire after reception of the acquisition trigger;
- the decimation factor (1, 2 or 4).

The memory mapped registers include, together with the acquisition parameters, a set of control registers with software triggers and resets.

When the acquisition start trigger is fired by software, the acquisition control logic exits the idle state and reads the acquisition parameters settings from the memory mapped registers. Then it configures the DMA core to receive the expected number of bytes, and when the core is ready, it enables the storage of the selected samples. When the DMA has finished the transfer, it raises an interrupt to the acquisition logic, which reverts its status back to idle. Other information available in the status register include the DMA status, which contains possible error codes and the number of bytes transferred.

The DMA IP core interfaces the Memory Controller of the PS DDR. The data stream is 128-bit wide and is clocked at 333 MHz. There is a DMA IP core per system of 4 ADCs.

Configuration of the System

The modification of the registers used for storing acquisition parameters, as mentioned in the Gateware section, will be referred to as configuration of the system. To seamlessly integrate the configuration with CERN's control systems, it is imperative to establish a standard means of interaction, which for our system is through a FESA Server. Since these registers, holding the acquisition parameters, are accessible from the PS, the implementation of a communication protocol becomes necessary to allow communication between the computer running the FESA Server and the PS of the SoC.

After careful consideration of various protocols, the decision was made to implement the IpBus protocol for the prototype [5]. Several factors influenced this choice. Firstly, IpBus is a widely adopted protocol in numerous beam instrumentation projects, which means there is a wealth of expertise within the group [6]. Additionally, well-documented libraries like Ironman and uHal facilitate and accelerate the development process. It's worth noting that IpBus employs a 32-bit addressing scheme, while the SoC board supports 64-bits. However, for the purposes of the prototype, addition of a prefix to the memory addresses pointing to the second half of the memory mitigates this incompatibility and didn't pose any significant issues.

The configuration consists of three key components: a memory definition file that catalogs all control and status registers, a text file specifying the associated ports along with the memory definition file name, and a Python script responsible for running the server based on the Ironman library which is a software toolbox for SoC communications [7]. While the FESA device uses a uHal-based custom library for communication with the Ironman server, the Ironman server operates by block-waiting for IPBus packages from the FESA device and executing the corresponding read or write operation upon receipt.

Data and Status Readout

Data generated by the gateware is initially stored in the PS DDR Memory. To facilitate integration with CERN's control systems and enable logging, a communication application running on the PS is essential. After careful consideration, we opted to implement a very simple protocol based on the Transmission Control Protocol (TCP) for this prototype. Other protocols were considered (e.g. ZeroMQ), but in order to keep code complexity and external dependencies to a minimum, we decided to make a bespoke protocol, thus maintaining control of the protocol and facilitating benchmarking and debugging.

Within the TCP communication, the first elements of each packet serve as headers to denote the package type. These package types encompass various functions, such as read requests, data incoming, heartbeat requests, heartbeat incoming, set DMA address requests, ADC reset requests, ADC shutdown requests, CPU temperature read requests, and exit requests.

In response to ADC reset and ADC shutdown requests, two user applications packaged within the operating system are invoked.

When the FESA device's TCP server initiates a DMA address setting request packet, it includes the proper header for DMA address setting, along with the configured DMA address. Upon receipt, the memory region starting from this address is mapped into the virtual address space of the program, ready for subsequent data readout.

For read requests, the TCP server of the FESA device includes the number of bytes to be read in the packet. The TCP client on the PS then reads the specified number of bytes from the mapped memory, sending them in chunks while including the data incoming header.

Regarding temperature read requests, the physical memory address remains constant and is mapped during the initialization of the TCP client. Upon receiving a packet with this header, the client retrieves the temperature from the already mapped memory and returns it, packaged alongside the data incoming header.

FESA Class

FESA is a framework developed at CERN, facilitating the digital representation of physical devices and offering real-time capabilities for action scheduling and execution [4]. This framework has become the standard for control and data acquisition at CERN accelerator systems. Therefore, to facilitate seamless integration of acquisition data and settings of the prototype into the CERN control system, a dedicated FESA class has been developed to interface with the SoC.

The FESA class developed for this project offers a list of expert settings corresponding to the acquisition parameters of the PS logic. Once these expert settings are configured, the FESA class transfers the information to the SoC's PS, where it is written to the appropriate registers. Only after this step are the acquisition parameters taken into account during the next acquisition cycle.

The FESA class includes a Real-Time (RT) Action for arming the PS's acquisition process by writing to the control register. This RT action is triggered by:

- Manual, (facilitates testing and development);
- User-defined time intervals in milliseconds;
- CERN timing events.

Once the acquisition is armed, the RT action waits for the acquisition and DMA data transfer to complete. It relies on the configuration application to determine when to request the transfer of acquisition data from the PS.

Communication with the configuration application is facilitated through a custom communication library based on the μHAL library which provides a python API for IpBus commands [5]. The custom communication library not only encompasses the list of registers mentioned in the Gateware section, along with comprehensive information on their usage and associated flags, but also utilises the μHAL to send IPBus packages to the PS configuration application. It is packaged as a shared object, designed specifically for the seamless integration with FESA devices.

After confirmation by the configuration application that the acquisition and DMA transfer have concluded, the length of the DMA transfer is determined. The integrated TCP server within the FESA class relies on the configuration parameters of the FESA device to ascertain the physical memory address to be utilised by the DMA. This address is then included in the TCP packet, along with the corresponding header. Upon reception by the TCP client on the SoC, the DMA address is retained for future memory access. It's worth noting that transmitting the DMA address may be required only once, despite the TCP client-server's ability to handle it multiple times.

Following this step, the TCP data transfer process is executed, as shown in the Figure 2.

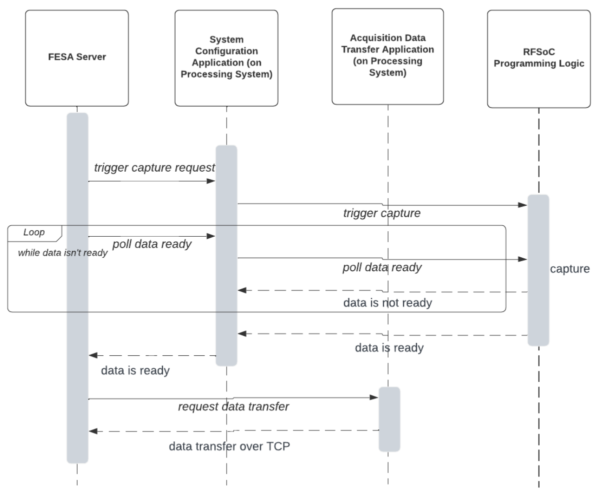

Figure 2: Sequence diagram showing the interaction between the FESA applications, RFSoC server and the PL.

Once the data is received by the FESA Class, it undergoes reordering to determine its association with the corresponding BPM. The FESA Class displays only a selected window of the acquisition data to improve performance while providing the complete data set to the logging software.

Data Logging

When the acquisition data reaches the FESA system, it is subsequently directed to the logging class implemented within the FESA class. In standard operating conditions, FESA data publications can be configured to be logged to NXCALS [8] via the CERN Controls and Configuration Database (CCDB) [9]. However, the BPM HL-LHC scenario presents a challenge as the volume of acquired data exceeds the recommended capacity of the accelerator logging system.

To address this specific challenge, a custom data logging solution was developed. One of the requirements was to facilitate the concurrent logging of acquisition data alongside the expert settings. The HDF5 file format was chosen for this purpose. This selection was based not only on the section's established expertise in generating and utilising HDF5 files but also on the availability of an existing HDF5 analyzer GUI, which greatly facilitates the visualisation of the generated files.

Graphical User Interface

Interaction with FESA devices is made possible through a program known as FESA Navigator. This program provides a list of device settings alongside acquisition data. It offers various built-in features for visualising the acquisition data. However, it is common for beam instrumentation systems to have their own dedicated GUIs. These GUIs are customised to meet the specific needs of beam instrumentation experts, including providing more flexible means for data analysis.

In the past, these GUIs were implemented using the Java programming language and Swing or JavaFX libraries [10]. However, given it's enhanced ability to visualise scientific and engineering data, PyQT, alongside the pyqtgraph library, has become a preferred choice. PyQT's rapid performance is attributed to its use of the Qt graphics libraries, and packages such as NumPy make post-processing much easier [10].

A bespoke BI group library known as 'expert-gui-core' that depends on PyQT, pyqtgraph, and uses the Accelerating Python(acc-py) tools and packages which are made in the Accelerator Department for supporting python development, was selected for the development of the new GUI [11].

The GUI development commenced using our in-house BI GUI manager. This tool streamlines the process of initialising a GUI based on a template, already with access to the 'expert-gui-core' and 'acc-py' packages, and integrated with GitLab. The current version of the GUI is shown in Figure 3.

In the current version an interface for modifying expert settings and visualization of the raw data from the FESA device are provided.

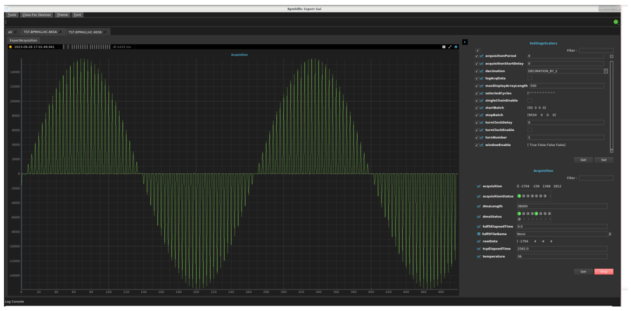

Figure 3: A screenshot from the expert GUI application. The expert settings are listed on the top right corner. The acquisition plots show data from the lab setup where the output of a pattern generator configured to produce sinus waves, is connected to one of the ADCs in order to verify the system.

PROTOTYPE VALIDATION WITH BEAM DATA

Due to the unavailability for testing purposes of the LHC BPMs, testing the system with a SPS spare BPM was chosen as an alternative. The main implication of deploying the system in the SPS is that, unlike the LHC, the SPS has multiplexed timing. Since the feature of triggering acquisitions based on the timing will anyway be removed during the migration to the LHC, the difference was considered non-essential.

The RFSoC board was installed in a cavern, close to the beam tunnel, isolated from radiation. Four of its onboard ADCs were connected to the four electrodes of a BPM. The board was registered on the CERN's Technical Network (TN), which is isolated from the general internet for security reasons. A FEC was prepared to run the FESA server. After verifying the setup and installation, the data acquisition commenced. Some examples of the waveforms acquired using the SoC are displayed in Figures 4 and 5. These results match the expected behaviour of the system and have helped us validate the full acquisition, processing and storing chain.

PERFORMANCE

The signal processing requirements, as detailed by a previous paper, include a read-out bandwidth requirement of 54 Mbps [3]. In the software components of the system, there are two critical stages where the handling of this substantial amount of data are potential bottlenecks. The first operation, responsible for reading acquisition data from DDR memory and transmitting it over a TCP connection, falls under the responsibility of the TCP Client. The second operation, involving the writing of both the raw data and its reordered version to an NFS disk, is managed by the FESA Class. To assess the prototype's performance on these operations and determine its compliance with the aforementioned requirements, three distinct measurements were undertaken. Firstly, the performance of TCP transactions was evaluated.

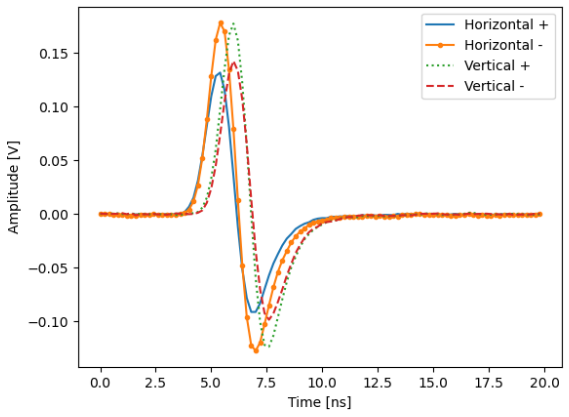

Figure 4: The waveform acquired, from four different electrodes at the passage of an SPS bunch.

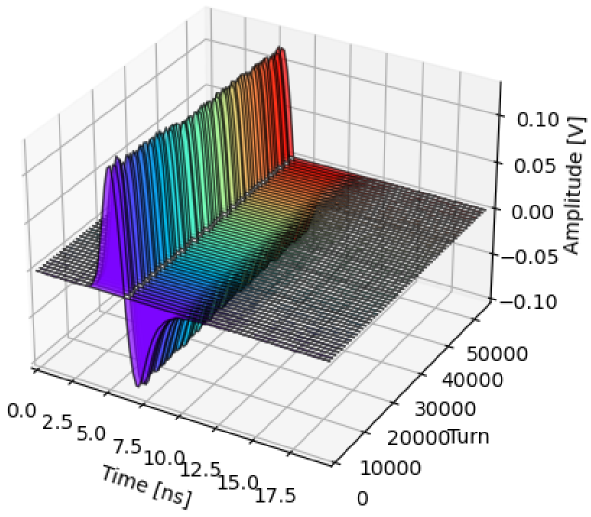

Figure 5: Multiple waveforms acquired turn after turn, from a single electrode at each passage of an SPS bunch. Only one waveform every 1000 turns is plotted.

This assessment involves the TCP Server within the FESA server, which sends a data request packet and subsequently waits to receive all the necessary packets. The second aspect involves saving both the raw data and its reordered counterpart to the NFS file system, referred to as data storage. The final aspect of evaluation encompasses the entire acquisition process, from triggering calibration through the configuration application to waiting for acquisition completion, executing TCP transactions, data reordering, and data storage.

To measure the data rate, a single FESA device is used, communicating with the SoC, which incorporates four ADCs for data sampling. According to these measurements, the performance of the TCP transport is found to be 508.8 Mbps (over an Ethernet link of 1 Gbps) while the data storage rate is 543.2 Mbps. The data rate for the aforementioned entire acquisition process is found to be 13.6 Mbps demonstrating the need for optimization in the final system.

CONCLUSION AND OUTLOOK

This prototype showcases that SoCs offer a variety of components to leverage. While this poses advantages during the development processes of different systems, it also highlights the need for standardisation of the technical choices within the accelerator sector. The experience gained from developing this system provided us with valuable insights into areas that could benefit from improvement. Firstly, we identified the preparation and maintenance of the OS for the PS as a time-consuming and resource-intensive task. It involves setting up an appropriate work environment, requiring system administration skills, and building the OS image. In addition, in a future with many SoC-based projects, standardisation would play a primary part for maintaining these systems. Thus having many different operating systems may become a burden and risk. Considering these reasons, it would be pragmatic to have a standardised OS provided by a central service at CERN.

Running FESA servers directly on a PS of a SoC holds the potential for future advantages, such as eliminating an additional layer of communication, reducing system complexity, and enhancing performance. However, this would require providing support for the SoCs similar to the FECs. If it is necessary to run the FESA servers in the FECs, identifying a standardized communication protocol between the SoCs and the FESA devices becomes essential for the long term maintenance of SoC based systems.

REFERENCES

[1] Xilinx, UG1410 - ZCU208 Evaluation Board User Guide (v1.0), `https://docs.xilinx.com/v/u/en-US/ug1410-zcu208-eval-bd`

[2] Zynq UltraScale+ RFSoC Product Data Sheet: Overview (DS889), `https://docs.xilinx.com/v/u/en-US/ds889-zynq-usp-rfsoc-overview`

[3] I. Degl'Innocenti *et al.*, "HL-LHC BPM electronics development as a case study for direct digitization and integrated processing techniques in accelerator instrumentation", in *Proc. IPAC'23*, Venice, Italy, May 2023, pp. 4657-4660. `doi:10.18429/JACoW-IPAC2023-THPL089`

[4] M. Arruat *et al.*, "Front-End Software Architecture", in *Proc. ICALEPCS'07*, Oak Ridge, TN, USA, Oct. 2007, paper WOPA04, pp. 310–312.

[5] C. Ghabrous Larrea, K. Harder, D. Newbold, D. Sankey, A. Rose, A. Thea and T. Williams, "IPbus: a flexible Ethernet-based control system for xTCA hardware", *JINST*, vol. 10, no. 2, p. C02019, 2015. `doi:10.1088/1748-0221/10/02/C02019`

[6] A. Guerrero, D. Belohrad, J. Emery, S. Jackson, and F. Roncarolo, "Modular Software Architecture for the New CERN Injector Wire-Scanners", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 487–491. `doi:10.18429/JACoW-ICALEPCS2021-TUPV037`

[7] G. Stark, 2015, Ironman's documentation, `https://ironman.readthedocs.io/en/latest/`

[8] J. P. Wozniak and C. Roderick, “NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service”, in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 1465–1469.
doi:10.18429/JACoW-ICALEPCS2019-WEPHA163

[9] Z. Zaharieva, M. Martin Marquez, and M. Peryt, “Database Foundation for the Configuration Management of the CERN Accelerator Controls Systems”, in *Proc. ICALEPCS'11*, Grenoble, France, Oct. 2011, paper MOMAU004, pp. 48–51.

[10] S. Bart Pedersen and S. Jackson, “Graphical User Interface Programming Challenges Moving Beyond Java Swing and JavaFX”, in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 637–640.
doi:10.18429/JACoW-ICALEPCS2019-MOPHA173

[11] P.J. Elson, C. Baldi, and I. Sinkarenko, “Introducing Python as a Supported Language for Accelerator Controls at CERN”, in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 236–241.
doi:10.18429/JACoW-ICALEPCS2021-MOPV040