ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MODR004

DISTRIBUTED I/O TIER AS A REFERENCE PLATFORM FOR HARNESSING SYSTEM-ON-CHIPS IN CERN'S CONTROL SYSTEM: GATEWARE DESIGN, BUILD SYSTEM, AND SOFTWARE SERVICES

A. Arias Vázquez*, V. Amoiridis, G. Daniluk, A. Pinho, B. Guncic, M. Lettrich, F. Vaga
CERN, Geneva, Switzerland

Abstract

The Distributed I/O Tier (DI/OT) project was initially launched to develop a common, modular hardware platform for custom electronics at the lowest layer of the CERN control system. With the adoption of the AMD Zynq UltraScale+ MPSoC for the high-performance System Board, DI/OT has also become a reference platform for integrating System-on-Chip (SoC) technology into CERN's control system. This paper presents two key aspects of DI/OT's role as a SoC reference platform: (1) tools and methodologies to streamline end-application development and (2) integration of DI/OT into CERN's control system as a Front-End platform. The first aspect includes a user-friendly build system and reference design that enable seamless integration of custom FPGA IP cores and Linux device tree entries while providing the reference design with essential DI/OT functionality and monitoring interfaces for local crate peripherals. This build system also automates synthesis and low-level software compilation to generate a complete bootable binary. The second aspect covers a fail-safe and reliable SoC boot mechanism, network booting of FECOS (a Debian-based CERN Linux image for Front-End Computers), and integration with standard monitoring services.

INTRODUCTION

The use of System-on-Chip (SoC) platforms has grown rapidly within the Accelerator and Technology Sector (ATS) at CERN, driven by their ability to combine real-time logic, embedded processing, and flexible I/O in a compact form factor. This trend is expected to intensify in the lead-up to Long Shutdown 3 (LS3) of CERN's accelerator complex, as new systems demand tighter integration, higher performance, and lower latency at the front end.

The DI/OT project [1] was the first SoC-based platform developed specifically for integration into the CERN control system. Its early adoption of the Zynq MPSoC, combined with a modular hardware design and a complete software stack, established it as the reference platform for subsequent SoC developments in ATS.

However, as more SoC platforms emerge, common challenges are being faced independently: managing gateware complexity, ensuring safe boot and configuration, integrating with control system services, and supporting standard workflows. This fragmentation highlights the need for a unified SoC framework where the DI/OT project becomes the reference.

* alen.arias.vazquez@cern.ch

DI/OT PLATFORM OVERVIEW

The Distributed Input/Output Tier (DI/OT) project provides a centrally supported modular hardware kit for building custom electronics systems in the lowest tier of the control stack, directly interfacing to accelerator equipment. The basic kit is illustrated in Fig. 1. The modules are housed in a 3U Europa crate featuring an in-house designed, passive backplane compliant with the CompactPCI Serial (CPCI-S.0) standard. Having a fully passive backplane prevents potential radiation-related issues in this critical component [2].

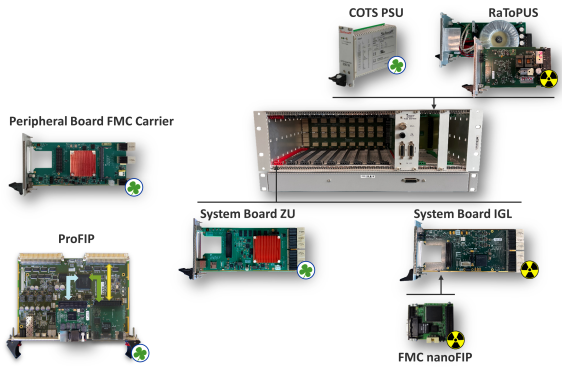

Figure 1: Distributed I/O tier hardware kit [1].

As shown in Fig. 1, the leftmost slot of the crate hosts the system board, while a star topology in the backplane allows high-speed point-to-point communication between the system board and the peripheral boards.

There are two variants of the platform: radiation-tolerant and non-radiation-tolerant [3]. While they share exactly the same 3U crate, they differ by the hardware modules hosted inside that DI/OT crate:

- **Radiation-tolerant variant:**
  - Custom-designed power supply (RaToPUS) developed in-house.
  - System board featuring a flash-based Microsemi IGLOO2 FPGA.
- **Non-radiation-tolerant variant:**
  - Off-the-shelf, CPCI-S power supply.
  - System board based on an AMD Xilinx Zynq UltraScale+ MPSoC.

Rather than replacing established platforms such as PLCs or VME, DI/OT is designed to address two specific but essential use cases:

- Electronics exposed to radiation, where qualification in test facilities and mitigation techniques need to be applied.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MODR004

- Systems requiring custom inter-board connectivity and/or hard real-time data processing, where the programmable nature of the system board and the backplane's flexible copper lanes allow direct streaming (e.g. of ADC data), interlock wiring, and other application-specific configurations.

Combining early adoption of SoC technology with seamless integration into CERN's control system, DI/OT has established itself as the reference platform for current and future SoC-based developments in ATS. This paper specifically describes the non-radiation-tolerant variant, which serves as the foundation for a unified SoC architecture and software framework.

TOOLS AND METHODOLOGIES

The initial focus involves the development of an intuitive build system coupled with a comprehensive reference design, enabling the seamless integration of application-specific FPGA IP cores and corresponding Linux device tree configurations. The reference design incorporates critical DI/OT functionalities and monitoring interfaces for local crate peripherals. Furthermore, the build system fully automates the synthesis of gateware alongside the compilation of low-level software components, culminating in the generation of a complete and bootable system image. To ensure continuous integration and delivery, the entire build and verification process is integrated within a GitLab CI/CD pipeline [4], which includes automated hardware-in-the-loop(HIL) testing to validate design correctness against physical hardware in a controlled environment.

Build System

The DI/OT project employs a modular and scalable build system specifically tailored for heterogeneous projects utilizing AMD Xilinx Zynq Ultrascale+ platforms. Built on a GNU Makefile infrastructure, the build system supports essential operations including `build`, `clean`, `mrproper`, and `distclean`, facilitating project management and artifact control.

Leveraging the CI4FPGA continuous integration framework [5] and standardized Docker images, the system ensures reproducible builds and streamlines development workflows. A Git tagging mechanism introduces safeguards between user modifications and the core platform, preserving platform integrity while allowing flexibility. The build environment automatically alerts users when the Vivado project is outdated relative to the source TCL scripts, ensuring synchronization between design sources and project files.

During compilation, warnings, critical warnings, and errors from the implementation phase are explicitly displayed in the build console output, enhancing user awareness and expediting debugging. Incremental compilation is enabled throughout all build stages, including synthesis and implementation of the gateware, significantly reducing turnaround time.

The build system encompasses all necessary phases to produce a complete bootable image. This includes the generation of gateware bitstreams, Arm Trusted Firmware, First Stage Bootloader, PMU Firmware, U-Boot bootloader, and the Device Tree. To enforce design correctness, any violation of timing constraints triggers an automatic compilation failure.

Device Tree generation is automated directly from the Vivado hardware description file (XSA), while providing a clean and efficient mechanism for users to customize and extend Device Tree configurations and peripheral descriptions. Similarly, the system offers user-friendly methods to customize gateware designs, including modifications to block designs, configuration parameters, and the addition of new design sources.

The build system incorporates a patching mechanism for the firmware components, as illustrated in Fig. 2. This includes:

- First Stage Bootloader (FSBL)
- PMU Firmware (PMUFW)
- Arm Trusted Firmware (ATF)
- U-Boot bootloader

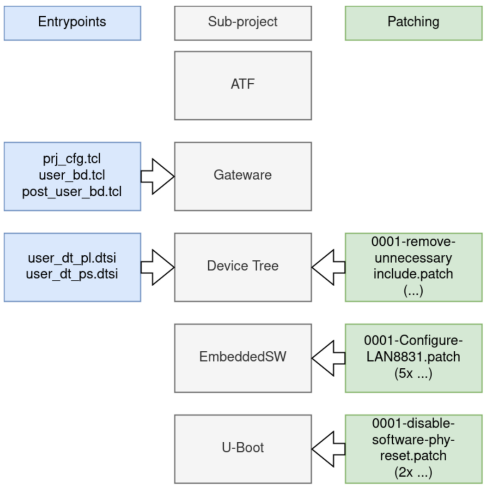

Figure 2: Block diagram of the patching system implemented in the build system.

This patching system enables users to modify and extend firmware functionality while maintaining compatibility with the core platform. Combined with comprehensive documentation, the solution provides a flexible framework for customization without compromising system integrity. The modular design ensures that modifications can be implemented consistently across different hardware configurations and use cases.

A typical build starts with the user updating or adding custom FPGA IP cores within the Vivado block design, followed by committing changes to the Git repository [4]. Upon triggering the CI pipeline, the system performs incremental synthesis and implementation, generates the hardware description (XSA), and automatically updates the Device Tree.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MODR004

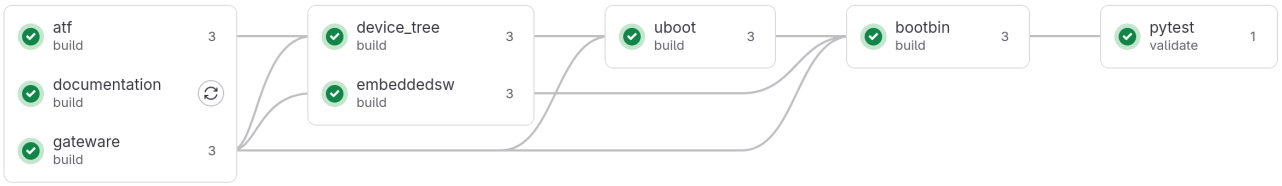

Figure 3: DI/OT build system flow, illustrating gitlab continuous integration from gateware synthesis to HIL [4].

Subsequently, low-level software components, including the trusted firmware and bootloaders, are compiled and patched if necessary. Finally, all elements are assembled into a unified bootable image ready for deployment, as illustrated in Fig. 3.

DI/OT Reference Design

The DI/OT reference design serves as a pre-validated Vivado block design that accelerates development by providing key platform services and reusable IP interfaces. It ensures users can integrate custom logic while preserving compatibility with DI/OT infrastructure and CERN control system requirements.

The block design, shown in Fig. 4 includes the following core components:

- **Processing System**: Default configuration to boot Linux and provide required interfaces.
- **Aurora + Chip-to-Chip (C2C)**: a high-speed serial link based on Xilinx Aurora and Chip-to-Chip protocols. This enables communication between the system board and peripheral boards via the CPCI-S backplane. These links are critical for high-speed streaming application data such as ADC samples or exposing memory-mapped interfaces.
- **AXI I2C PL Master**: An AXI-based I2C controller is instantiated in the Programmable Logic (PL) to interface with local I2C peripherals. These may include temperature sensors, EEPROMs, or crate-level I2C devices. Using a PL-based I2C master enables deterministic control and independent access from the PS-side Linux drivers.
- **Diagnostics Interfaces (PL/PS shared access)**: Diagnostic signals such as system status, board presence, and power supply health are made available to both the Processing System (PS) and Programmable Logic. This dual-access mechanism enables tight integration of monitoring services within both firmware and software domains, ensuring high reliability and observability of the system.
- **Modularity and Extensibility**: The block design is intentionally modular, allowing users to insert their own custom AXI peripherals, DMA engines, or another IP blocks.
- **Device Tree Integration**: Each hardware peripheral instantiated in the block design is automatically exported in the Vivado-generated '.xsa' file. The build system then generates a Linux-compatible Device Tree that includes all base and custom peripherals, enabling user-space access via standard Linux drivers or custom services.

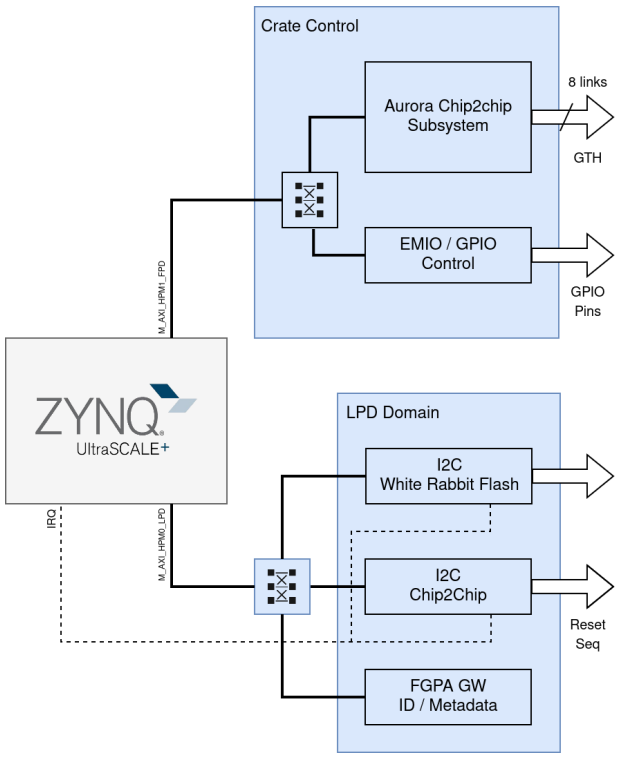

Figure 4: DI/OT gateware reference design.

This reference design acts as the foundation for all DI/OT applications. It standardises peripheral access, facilitates inter-board communication, and ensures that custom designs remain aligned with the DI/OT ecosystem and CERN's control system.

INTEGRATION INTO THE CERN CONTROL SYSTEM

The CERN control system orchestrates the operation of the accelerator complex through a distributed architecture. It is organized in three functional layers (Fig. 5):

- front-end computers interfacing directly with hardware.
- middle-tier servers handling data processing and coordination.
- operator interfaces for monitoring and control.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MODR004

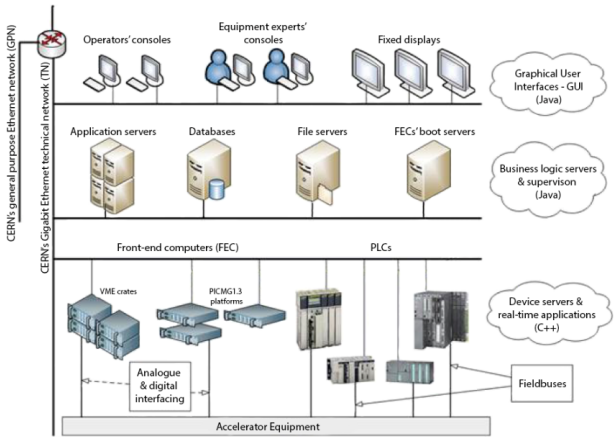

Figure 5: The three-tier architecture of CERN's accelerator control system [6].

This hierarchical design enables deterministic low-level control while providing flexible high-level operation, with all components integrated through CERN-developed software frameworks and communication protocols.

The system has continually evolved to integrate new technologies while preserving operational reliability. Starting with VME-based architectures from the 1980's onwards, it advanced through PCI and PCIe systems, and is now transitioning toward PXIe solutions and System-on-Chip platforms such as DI/OT. This ongoing technological progression has led to improved performance, lower latency, and better fulfillment of critical requirements for the HL-LHC.

Reliable Booting Mechanism

The main idea of the Reliable Booting Mechanism is that it splits the boot process in 2 stages [7]:

- **QSPI golden boot**: The hardware switch on the DI/OT System Board is configured such that whenever there is a power cycle, the board boots from QSPI flash. The QSPI flash holds a BOOT.BIN that is known to work. It also has a BOOT0001.BIN that is bit-by-bit identical with the BOOT.BIN and is there in case the BOOT.BIN gets corrupted, to trigger the Golden Image Search. This BOOT.BIN is responsible to perform basic hardware checks, change the bootmode(by writing to the internal volatile registers), and continue to the next stage of the eMMC user boot.
- **eMMC user boot**: This BOOT.BIN is generated by the DI/OT user and is stored in the first partition of the eMMC. The image contains the user's custom FPGA bitstream and other components (e.g. FSBL) generated and compiled based on their application-specific modifications. The user BOOT.BIN is also generated by the DI/OT build system, from their forked repository.
  The eMMC BOOT.BIN moves to fetch GRUB over the network in order to initiate the FECOS boot process.

The SoC boot process is shown Fig. 6. To configure the SoC, it is necessary to flash the memories with a specific structure which is shown in Fig. 7.

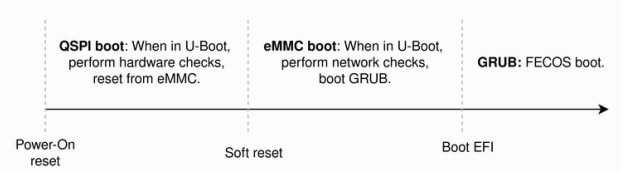

Figure 6: Reliable booting sequence.

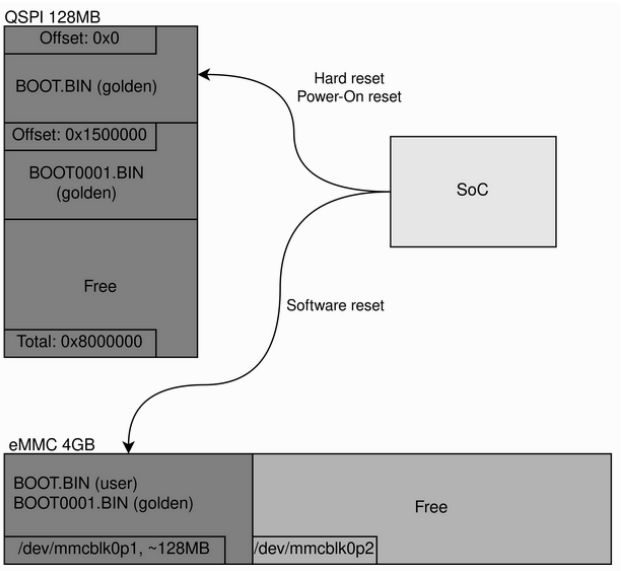

Figure 7: Memory configuration for reliable booting.

The goal is to prevent system failures during boot time, system upgrades, or runtime errors. There is no mechanism to prevent failures in the ROM of the chip or during the FSBL boot phase. Scenarios in which the system cannot connect to the network are not addressed. It is a policy at CERN that all FECs are reset until the network is available to retrieve the operating system.

Network Booting

Figure 8 shows the most common SoC boot process when the system is booting Linux. Xilinx uses U-Boot as its second-stage boot loader (SSBL). U-Boot is loaded into RAM physical address by FSBL and is executed after ATF jumps to it.

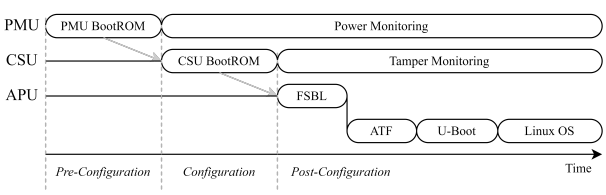

Figure 8: Simplified boot process [8].

U-Boot in the DI/OT project, is mainly responsible for fetching and booting GRUB instead of the Linux kernel in order to adhere to the FECOS boot process. This mechanism allows the same flow to be applied to both SoC and x86

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MODR004

platforms at CERN. It is the GRUB that loads appropriate kernel image depending on the system type. In order to boot FECOS, 3 things need to happen:

- **DHCP request** to get IP.
- **SNTP request** to the NTP server received as part of the DHCP response. This step is required to get the correct time needed for kerberos authentication - thus to alow ssh acces after Linux boots.
- **TFTP request** from the bootserver whose IP address was received during the DHCP request, to fetch the GRUB image.

If these requests succeed, GRUB is fetched and booted. Notably, no device-tree is explicitly provided. This is because, the DI/OT build system described earlier provides the device-tree to U-Boot. In case the *bootefi* [9] command is called without a device-tree argument, U-Boot automatically uses that internal device-tree.

Monitoring Services

DI/OT systems must meet the stringent uptime requirements of the CERN accelerator systems. Seamless integration into CERN's existing monitoring infrastructure is therefore essential, enabling proactive management of system health and availability. When issues are identified, dedicated on-system software allows for detailed inspection of the crate's hardware composition and the execution of administrative actions, such as device resets or power cycles.

The monitoring strategy integrates with standard Linux services running on the FECOS distribution. For observability, DI/OT interfaces with COSMOS, CERN's monitoring platform, which uses collectd [10] to gather metrics. Hardware data is channeled via the Linux HWMON [11] framework by writing custom drivers for designs. This approach allows the standard collectd-sensors plugin to forward metrics directly to COSMOS [12] through sysfs, eliminating the need for custom intermediary code.

Unlike traditional CPU-centric systems, DI/OT is FPGA-centric; the OS primarily serves to interface with the CERN control system and administrate the hardware. Peripheral boards and FMCs therefore do not require OS drivers to function and are not identified via mechanisms like PCI IDs. Instead, they must have an EEPROM containing Field Replaceable Unit (FRU) information at a specified I2C address for identification. Gateware-specific information is exposed via memory-mapped registers accessible from the OS through the UIO framework. While device slots are statically defined in the device tree, their population and identity are confirmed at runtime by reading these EEPROMs. Additionally, GPIO pins are used for presence detection and low-level control.

To manage these interactions, we developed *libdiot*, a C library with Python bindings (*pydiot*), that abstracts sysfs device nodes and addressing. On top of these libraries, we provide applications like *lsdiot* for listing the crate's hardware composition and *diotctl* for executing administrative actions. This foundation also allows equipment groups to develop their own custom applications. The framework currently supports DI/OT system boards v2 and v4 and is being generalized for all SoCs within the Accelerator and Technology Sector at CERN.

OUTLOOK

The near future development roadmap focuses on the following key points:

- **Implementation of CRAZy (Compact Remote Administration for ZynqMP)**: a minimalistic IPMI/RMCP (Intelligent Platform Management Interface/Remote Management Control Protocol) stack running inside the PMU Firmware (PMUFW). This will provide out-of-band management, enabling **remote reset** and **Serial over LAN** console access. This ensures engineers can diagnose and recover from system crashes or boot failures without physical intervention.
- **Remote reprogramming of DI/OT peripheral boards**: Development of a secure mechanism to deploy FPGA bitstreams and/or firmware updates over the network to the entire crate, significantly reducing downtime and streamlining maintenance.
- **Enhanced user support and tooling**: This will integrate standard interfaces for peripheral board monitoring and network-based firmware updates into the *libdiot* and *pydiot* libraries

CONCLUSION

The DI/OT platform has evolved into a reference SoC solution for CERN's control system, bridging the gap between modular hardware and integrated software services. Its adoption of the Zynq UltraScale+ MPSoC, combined with a build system and validated reference design, simplifies the development and deployment of FPGA-based applications.

The build system enables seamless integration of custom IP, automated Device Tree generation, and reproducible, CI-driven image creation. The reliable booting mechanism ensures resilience against power cycles and system failures, while supporting CERN's network boot strategy and FECOS integration.

Through tools like `lsdiot`, `diotctl` and COSMOS integration, DI/OT supports both local and global monitoring, aligning with CERN's operational requirements for high availability and observability.

As SoC adoption grows within the CERN's Accelerator and Technology Sector, DI/OT sets a standardised path for future SoC-based developments, enabling scalable, maintainable, and reliable control system integration ahead of the Long Shutdown 3 and beyond.

ACKNOWLEDGEMENTS

I am grateful to the leaders and members of the SoC project for their technical input to make possible the development of the platform. I also wish to acknowledge the support of the BE-CEM group, with special mention to the

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MODR004

members of my section, BE-CEM-EDL, for their continuous effort, teamwork and dedication in our daily work.

REFERENCES

[1] G. Daniluk *et al.*, "Distributed I/O Tier: from concept to operational readiness – a modular platform for custom radiation-tolerant and radiation-free electronics", presented at ICALEPCS'25, Chicago, USA, Sep. 2025, paper WEBR006, this conference.

[2] T. Gingold *et al.*, "HydRA: A System-on-Chip to Run Software in Radiation-Exposed Areas", in *Proc. ICALEPCS'23*, Cape Town, South Africa, Oct. 2023, pp. 217–221. doi:10.18429/JACoW-ICALEPCS2023-MO4AO02

[3] DIOT - Distributed IO Tier, https://ohwr.org/projects/diot/.

[4] ZynqMP System Board, Boot Image, https://gitlab.cern.ch/be-cem-edl/diot/zynqmp/boot-image

[5] CI4FPGA, https://engineering-software.docs.cern.ch/eda/ci4fpga/.

[6] CERN's accelerator control system, https://be-dep-css.web.cern.ch/sites/default/files/Introduction_to_the_BE-CO_Control_System.pdf

[7] Reliable Booting Description, https://be-cem-edl.web.cern.ch/diot-boot-image/v2.3.0/doc/relboot/reliable-booting-desc.html

[8] Boot and Configuration, https://docs.amd.com/v/u/en-US/ug1085-zynq-ultrascale-trm

[9] Bootefi command, https://docs.u-boot.org/en/v2021.01/usage/bootefi.html

[10] collectd, http://collectd.org

[11] The Linux Hardware Monitoring kernel API, https://docs.kernel.org/hwmon/hwmon-kernel-api.html

[12] F. Locci *et al.*, "CERN Controls Open Source Monitoring System", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 404–408. doi:10.18429/JACoW-ICALEPCS2019-MOPHA085