Modular Hardware, Firmware, and Control-System Integration: Lessons from Commissioning the Fast Beam Interlock System at ESS

S. Pavinato, S. Gabourin, E. D'Costa, M. Carroll, R. Silveira, A. Nordt

***Abstract*—The Fast Beam Interlock System (FBIS) at the European Spallation Source (ESS) is a distributed, safety-critical system that collects status information from multiple accelerator subsystems and, when unsafe conditions arise, stops both the machine and the proton beam within microseconds. Throughout ESS commissioning, from integration tests to full beam operation, FBIS was developed using a modular approach spanning hardware, firmware, and control-system interfaces. Standardized mTCA crates and cPCI platforms enabled hardware components to be shared and reused across different interlock and diagnostic systems, streamlining development and maintenance. Each parametrized firmware version is validated with rigorous lab tests mirroring simulation methodologies, ensuring robust system performance before deployment, and efficient handling of diagnostics and event data during operations. Facility timing ensures precise alignment of diagnostic data across FBIS crates. This contribution summarizes architectural and practical lessons from deploying FBIS, highlighting how modular design in hardware and firmware enabled flexible adaptation, reliable operation, and a modular control-system layer throughout machine commissioning.**

***Index Terms*—Fast Beam Interlock System, Machine Protection, Modular Design, FPGA Verification, EPICS, History Buffer, Factory Acceptance Testing, Commissioning**

I. Introduction

THE European Spallation Source (ESS), located in Lund, Sweden, is one of the largest science and technology infrastructure projects currently being built. Its pulsed linear proton accelerator will deliver an unprecedented proton beam power of 125 MW per pulse (5 MW average) to a rotating tungsten target, generating intense neutron pulses through the spallation process [1].

If this beam energy is released without control, equipment damage unfolds on microsecond timescales. Protecting the machine is therefore not an operational concern but a primary design constraint, one that demands decision cycles measured in microseconds. To meet this requirement, the ESS Machine Protection System of Systems (MPSoS) relies on the Fast Beam Interlock System (FBIS): a distributed, FPGA-based system that continuously monitors status signals from across the accelerator and intervenes to halt beam and machine operation before damage can occur.

The FBIS collects data from approximately 250 Sensor Systems distributed along the linac, from slow PLC-based systems (managing vacuum, magnets, and insertable devices) to fast FPGA-based systems (managing radio frequency control and beam instrumentation). The FBIS is designed to stop beam within 5 μs of an interlock request reaching the actuators in the Normal Conducting Linac (NCL) section, and within 15 μs in the Super Conducting Linac (SCL). These requirements are derived from beam-induced damage timescales: in the NCL, material can melt within 1–3 μs, setting a protection target of 3–5 μs; in the SCL, the damage timescale extends to approximately 40 μs, allowing a reaction time of 10–15 μs.

This paper summarizes the architectural lessons and engineering outcomes from deploying FBIS across multiple commissioning phases at ESS, from the first beam to the Ion Source in 2021 through to the Beam on Dump phases in 2025–2026. It describes how a modular design philosophy, applied consistently across hardware platforms, firmware parametrization, verification methodology, diagnostic systems, and control-system integration, enabled a small team to build, test, deploy, and maintain a complex safety-critical system that remained fully operational throughout commissioning. Aspects of this work have been reported at conference level in prior publications [3]–[6]; this article integrates and extends those contributions.

All authors are with the European Spallation Source ERIC, Lund, Sweden.

II. System Architecture

A. Overall Structure

The FBIS architecture is fully redundant and designed to fulfill ESS Protection Integrity Level 2 (PIL 2) requirements, an internal machine-protection classification based on principles from the IEC 61508 functional safety framework. It is organized around two primary hardware component types deployed along the 600 m linac and in the Target building: the Signal Conversion Unit (SCU) and the Decision Logic Node (DLN). For the final configuration, 25 SCUs and 7 DLNs are installed, each implemented as a fully redundant pair of FPGA-based boards. The full system layout is shown in Fig. 1.

SCUs and DLNs communicate via optical links using SFP modules, gigabit transceivers, and the Xilinx Aurora 64b/66b protocol, referred to as Slink. DLNs share a Global Beam Permit (GBP), a signal asserted when no Beam Inhibit (BI), Regular Beam Interlock (RBI), or Emergency Beam Interlock (EBI) condition is active (indicating that beam production is permitted), through a dedicated optical interface called the Optical Protection Loop (OPL), also using Aurora 64b/66b. When a DLN determines that beam must be stopped, it asserts a Beam Switch-Off (BSO) request on the OPL, at one of three severity levels: BI, RBI, or EBI, and the two SCUs closest to

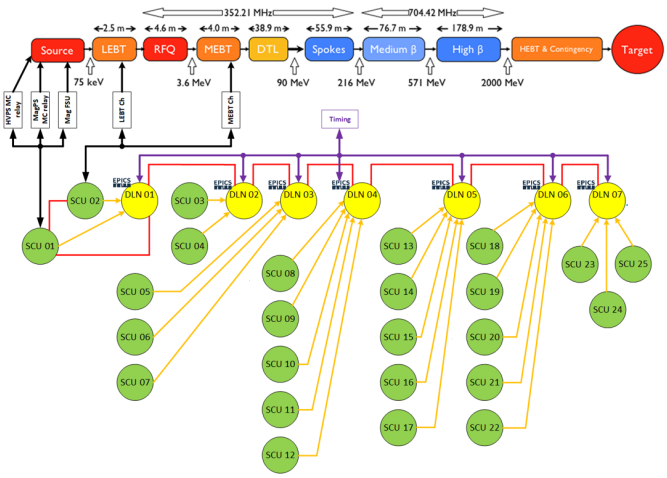

Fig. 1. FBIS architecture along the ESS linac. SCUs (green) connect to DLNs (yellow) via Slinks (orange); the OPL (red) links all DLNs and the two actuator SCUs; actuator connections are shown in black; purple arrows indicate EVR timing synchronization.

Fig. 2. DLN crate (top, mTCA 3U) and SCU crate (bottom, cPCI). In the DLN, the two IFC1410 boards implementing the interlock logic are on the left; the MCH, EVR, and Concurrent CPU are on the right; yellow fibers on the IFC1410 front panels are the incoming Slinks from the SCUs. In the SCU, the two Serializers are at each end; the three RS485 MCs at the centre connect to five BLMs via discrete RS485 signals and Ethernet datalinks; yellow fibers exiting from the bottom of each Serializer are the Slinks to the DLN above.

the Ion Source, which host actuator Mezzanine Cards, read the OPL state and act on the hardware actuators.

B. Signal Conversion Unit (SCU)

The SCU is a concentrator for Sensor System connections, based on a CompactPCI (cPCI) standard chassis with custom electronic cards. Each SCU hosts up to 12 Mezzanine Cards (MC) on which Sensor Systems connect. Two additional cards, called Serializers, each host a Xilinx MPSoC Zynq Ultrascale+ processor to manage backplane connections to MCs and the Slink optical communication to the DLN.

Several MC types were designed to interface with the variety of Sensor Systems at ESS. One MC type manages PLC-based Sensor Systems, using loop currents for discrete signals and PROFINET for the datalink (the digital communication channel for status and diagnostic data), with the host interface implemented entirely in VHDL [2]. Another MC type interfaces with fast electronic Sensor Systems, principally FPGA-based, using RS485 differential signals for discrete inputs and Manchester-encoded or proprietary Ethernet protocols for the datalink.

The two SCUs installed closest to the Ion Source also host specific MCs that interface with five hardware Actuators: upon receiving a BSO request from the DLNs, these actuators physically interrupt beam production. They comprise the two NCL choppers (LEBT and MEBT), the Ion Source magnetron RF, the fast shutdown unit, and the timing pulse inhibit; in emergency conditions the Ion Source high-voltage power supply is also removed. The Serializer processor runs Xilinx Petalinux, hosting a C software daemon that manages monitoring, control, and crate management. The SCU crate is shown at the bottom of Fig. 2.

C. Decision Logic Node (DLN)

The DLN is responsible for executing the FBIS protection logic. It is built on the mTCA standard [7], using a 3U chassis to house a redundant pair of IFC1410 Intelligent FMC Carrier AMC boards developed by IOxOS Technologies [8]. The DLN also hosts a Timing System Event Receiver (EVR) [9] and a Concurrent Technologies CPU [10] that runs the EPICS control system IOCs.

The protection logic is implemented in VHDL on the IFC1410 FPGAs, organized around parametrized Processing Units (PUs) that compute a unified representation called the Decision Logic Variable (DLV). The DLN receives data from SCUs via the Slink and routes it through parametrized paths to the appropriate PUs, set at synthesis time. The DLV of each PU can trigger three levels of beam switch-off: BI, RBI, or EBI. The DLN crate is shown at the top of Fig. 2.

Each DLN and SCU firmware version is individually parametrized and compiled into a dedicated bitstream before installation. For SCUs, parameters define which MC types are present in each slot and the Slink configuration toward the corresponding DLN. For DLNs, parameters specify which PUs are active, their masking and latching configuration, the Slink configuration, and the applicable Proton Beam Mode (PBM) and Proton Beam Destination (PBD) values: these are read from the timing system at runtime and determine whether a given PU is filtered out (i.e., excluded from contributing to the BI or RBI computation) for the current beam configuration. In both cases, parameters are hard-coded at synthesis and are not changeable at runtime, ensuring deterministic behavior.

III. Modular Hardware Design and Reuse

The choice of standardized hardware platforms proved essential for managing the complexity of deploying FBIS across commissioning phases. The mTCA standard for DLNs and the cPCI standard for SCUs established a common physical

and electrical infrastructure, allowing boards designed for different functional roles to be housed in compatible crates and maintained under a common support structure.

Within the mTCA crate ecosystem at ESS, standardization extends well beyond FBIS itself. More than 300 mTCA systems are deployed across the facility [7], spanning RF, Beam Instrumentation, Machine Protection, and Timing Distribution. The common AMC boards used across all these systems, namely the MicroTCA Carrier Hub (MCH), the Concurrent Technologies CPU, and the EVR, are maintained and updated by the ESS Integrated Control System (ICS) department, of which the Machine Protection System (MPS) group is a part. This organizational alignment means that the MPS team can focus its engineering resources on the two IFC1410 FPGA boards that carry the DLN protection logic and on the custom SCU platform, while benefiting from shared infrastructure, standardized deployment tooling, and the broader ESS FPGA development ecosystem [11].

The hardware tester developed for Factory Acceptance Testing (FAT) exploits this standardization directly: the core of the tester is itself an mTCA crate equipped with the same board types housed in a DLN crate, including custom FMCs and redesigned FPGA firmware for SCU and DLN testers. Drivers integrate the tester firmware with the EPICS layer, and a PLC with appropriate I/O modules, also driven via EPICS, emulates PLC inputs. Since the tester shares the same physical and electrical platform as the operational system, it can faithfully replicate the Aurora Slink and OPL environments seen by the device under test.

The standardized MCH and CPU deployment processes developed across ESS [7] also benefit FBIS directly. Ansible playbooks automate OS installation and EPICS environment setup on CPUs, and GitLab CI scripts handle MCH firmware deployment, reducing manual effort and ensuring consistency between crates. This means an upgraded DLN CPU can be brought to a fully configured state remotely, and the IFC1410 FPGA firmware updated, without requiring manual intervention at the crate.

IV. Firmware Verification: A Unified Simulation and Hardware Methodology

A. Problem Statement

FBIS firmware verification at ESS follows the V-Model, spanning unit testing, integration testing, system testing, and FAT. The first three stages are performed in simulation; FAT is conducted on real hardware. A challenge common to most FPGA verification environments is that simulation and hardware testing typically require separate test infrastructures with limited code reuse, increasing effort and risking divergence between what is verified in simulation and what is validated on hardware.

For a small verification team managing tens of parametrized firmware versions for SCUs and DLNs, the cost of duplicating test logic is significant. In the previous verification setup, simulation relied on VHDL or SystemVerilog testbenches while hardware testing used NI hardware and software tools such as LabVIEW and NI TestStand, a combination that proved impossible to maintain over time for a small team: tools, languages, and skill sets were fragmented across the two domains, and any change to the firmware required parallel updates in environments with no shared infrastructure. The approach adopted for FBIS aims to close this gap by borrowing key concepts from UVM and the Portable Test and Stimulus Standard (PSS) [12], while implementing them in Python, a language common to both the Cocotb simulation framework and the Pytest hardware test framework.

B. Cocotb and Pytest as a Common Foundation

The FBIS verification framework uses Cocotb 2.0 for simulation and Pytest for hardware testing [3]. Both are Python-based, and Cocotb 2.0 was selected precisely because version 2.0 closes much of the gap with Pytest: it adopts a Python-based runner in place of the legacy Makefile mechanism, introduces a `cocotb.parametrize` decorator analogous to Pytest's own parametrization, and supports test marking for label- or condition-based selection.

Both environments share the same `pytest` entry point, which allows them to feed the same GitLab CI/CD infrastructure and give results in JUnit XML format, a uniformity that simplifies audit trails and directly supports the traceability matrix described below. Scenario parameters (input combinations, expected outputs, protocol variants) are stored in Excel, JSON, or YAML files shared between both environments, so the same test scenarios exercise both the simulation model and the real hardware without duplication.

C. UVM-Inspired Modular Structure

The framework adopts a UVM-inspired modular structure. Each testbench is organized around an `Environment` class that instantiates a `Driver`, a `Monitor`, and a `Scoreboard` as independent, customizable components, with a common entry point for test execution enforced across both environments.

The `Driver` and `Monitor` are environment-specific: in simulation the `Driver` applies stimuli to HDL signals using Cocotb coroutines, while in hardware testing it sets EPICS process variables; the `Monitor` reads outputs accordingly from HDL signals or EPICS PVs. The `Scoreboard` is shared across both environments: its concrete implementations depend only on the logic under test, not on the execution environment. A full description of the class interfaces is provided in [3].

D. Specification Coverage and Verification Metrics

Completeness is assessed along three dimensions, described in detail in [3]. Requirements Traceability ensures that every test case maps to a Detailed Design Specification (DDS) item, which in turn traces to a System Requirements Specification (SRS) item; coverage is achieved when each requirement is exercised by at least one passing simulation test and one passing hardware test. Scenario Coverage ensures that every functional behavior described in the specification is exercised: V&V engineers define test scenarios in human-readable formats (such as Excel, JSON, or YAML), accessible to engineers

without low-level coding skills, which are then translated into parametrized Cocotb and Pytest test cases. Code Coverage, gathered by HDL simulators during simulation runs, confirms that the test scenarios drive the firmware implementation sufficiently; any uncovered code is reviewed to distinguish unused logic from gaps in the test suite.

Unlike constraint-random methodologies, the FBIS approach does not pursue functional coverage closure as its primary completeness criterion. Instead, the three dimensions above (traceability, scenario execution, and code coverage) are used in combination to argue that verification is sufficient. Randomized stimulus is used selectively for targeted unit tests, such as stressing data-path components in isolation, but plays no role in system-level coverage arguments.

V. Firmware Diagnostics: The History Buffer

The History Buffer (HB) is a firmware component present in all 64 FPGAs of the FBIS — every SCU Serializer board and every DLN FPGA. Implemented once as a generic VHDL module and instantiated identically across the system, it records timestamped state transitions in a rolling buffer, capturing signal status changes, beam permit transitions, and internal fault flags as they occur. Its design details — record structure, simultaneous trigger handling, event type reduction, glitch filtering, and cyclic data management — are covered in [4]; this section focuses on the operational role it plays and the cross-crate synchronization challenge it introduces.

A. Evolution of the History Buffer

The HB is the primary tool for post-mortem analysis: after a beam stop, experts read the relevant crate HBs to reconstruct the event sequence and identify the root cause. It is also used during Final Integration Testing (FIT) to measure actuator reaction times at microsecond resolution, and its records feed the ESS archiver for long-term machine protection statistics.

Going through potentially 7 DLN HBs to reconstruct the order of events is non-trivial, even as a system expert. For this reason a first cross-crate "super HB," merging the seven DLN HBs around a post-mortem trigger, has been demonstrated operationally [4]; Fig. 3 shows a representative event. The first obvious advantage is to sort inter-DLN events in chronological order. In addition, part of the individual DLN HB records that are not relevant for post-mortem event reconstruction are filtered and not shown. This makes post-mortem root cause analysis accessible not only to experts but also to operators, and therefore accelerates resuming beam operation.

The HB was not systematically tested in early development phases, as it was not considered part of the critical protection logic. When a dedicated operator interface for the HB was introduced during operations, latent bugs surfaced in both the firmware and the control-system layer [4]. This prompted targeted verification using the same Cocotb and Pytest framework described in Section IV, achieving full code and requirements coverage of the HB logic.

Date	ns	Signal	Value	DLN
2025-12-04 23:33:27	452,722,112	PBI-BCM01::FBIS-BP-DL	NOK	1
2025-12-04 23:33:27	452,719,936	PBI-BCM01::FBIS-R-BP-DL	NOK	1
2025-12-04 23:33:27	452,710,976	Local BI	NOK	2
2025-12-04 23:33:27	452,710,976	RFQ-010:RFS-FIM-101:FBIS-Red-BP	NOK	2
2025-12-04 23:33:27	452,710,976	RFQ-010:RFS-FIM-101:FBIS-BP	NOK	2
2025-12-04 23:33:27	452,710,944	Local BI	OK	2
2025-12-04 23:33:27	452,710,944	RFQ-010:RFS-FIM-101:FBIS-Red-BP	OK	2
2025-12-04 23:33:27	452,710,944	RFQ-010:RFS-FIM-101:FBIS-BP	OK	2
2025-12-04 23:33:27	452,710,912	Local BI	NOK	2
2025-12-04 23:33:27	452,710,912	RFQ-010:RFS-FIM-101:FBIS-Red-BP	NOK	2
2025-12-04 23:33:27	452,710,912	RFQ-010:RFS-FIM-101:FBIS-BP	NOK	2
2025-12-04 23:33:27	452,707,488	Local RBI	NOK	1
2025-12-04 23:33:27	452,707,488	Local BI	NOK	1
2025-12-04 23:33:27	452,707,456	PBI-BCM01::FBIS-R-BP	NOK	1
2025-12-04 23:33:27	452,707,456	PBI-BCM01::FBIS-BP	NOK	1

Fig. 3. Super-HB snapshot (newest records at top). BCM01 discrete signals on DLN01 go NOK first, triggering a Local RBI. The RFQ trip appears $\sim$3.4 μs later on DLN02 as a Local BI with a characteristic NOK–OK–NOK bounce (non-latched signal). BCM01 data-link signals, travelling via a slower path, arrive on DLN01 $\sim$12–15 μs after the discrete ones.

B. Synchronization Across Crates

Aligning HB timestamps across physically separate FPGAs is non-trivial: each board runs its own free-running clock, and even identical hardware accumulates measurable drift over time. For DLNs, the solution exploits the EVR infrastructure already deployed in each mTCA crate for beam timing: a one-pulse-per-second GPS signal routed over the backplane mLVDS lines disciplines the DLN firmware time reference, achieving consistent sub-200 ns alignment between DLN HBs, as verified by comparing the post-mortem timestamps recorded across all 14 redundant DLN channels. SCU HBs are aligned to their paired DLN by piggybacking timestamps on the Aurora Slink frames, a solution requiring no additional hardware but bounded in accuracy by the non-deterministic latency of the Aurora core. The resulting precision is sufficient for current operational use; further improvement is required before SCU records can participate in automated cross-system post-mortem analysis.

VI. Control-System Integration via EPICS

A. Architecture

The ESS Integrated Control System is based on EPICS [13], which abstracts data exchange with heterogeneous devices as standardized Process Variables (PVs). Both DLNs and SCUs use StreamDevice support and software daemons as an intermediary layer to integrate FPGAs into EPICS IOCs, with software daemons decoupling client requests from firmware register reads and writes. This flexibility is essential because DLNs (using the TOSCA network-on-chip) and SCUs (using AXI peripheral register access) use different internal protocols. Python libraries in the EPICS ecosystem provide the common mechanism for high-level test scripts, operator interfaces, and control software across all FBIS nodes.

B. SCU Control System

With all 12 slots filled, an SCU Serializer presents approximately 800 registers organized into read-only monitoring, read/write control (covering signal status and Beam Permit overriding), and write-only reset categories, accessed by the daemon via I2C and GPIOs. The EPICS IOC must know which MC type occupies each slot to decode these generic registers correctly.

SCU HBs are read by a dedicated server program every second and buffered in software; the SCU EPICS IOC reads only new records. This architecture ensures records are not lost even if the IOC polls less frequently than records are generated, provided the total rate stays within the 1024-record rolling buffer.

C. DLN Control System

DLN FPGA registers are accessed via TOSCA, the proprietary network-on-chip provided by IOxOS for the IFC1410 [8]. The DLN IOC organizes registers into two waveform groups: read-once parametrization data (PU maskability, applicable PBMs and PBDs) and periodically-read operational registers (Slink and OPL status, signal values before and after masking and filtering). In the worst-case parametrization, approximately one thousand registers must be read periodically. The DLN IOC also reads EVR timing PVs and writes them into FPGA registers to keep PBM and PBD updated.

D. High-Level FBIS IOC and Operator Interface

A high-level EPICS IOC consolidates all DLN IOC information into operator-friendly representations, introducing the concept of input systems: groups of FBIS signals corresponding to a single Sensor System as perceived by operators. For example, the Personal Safety System (PSS) presents two FBIS inputs; the high-level IOC consolidates these into a single PSS status indicator with a single masking button, shielding operators from underlying hardware redundancy. IOC deployment and monitoring use CE Deploy and Monitor, a Java-based tool that orchestrates Ansible playbooks, enabling remote, standardized administration across the ESS network.

VII. Commissioning Phases and Performance

A. Commissioning Methodology

The FBIS is commissioned in phases aligned with the ESS accelerator installation plan, proceeding through: FAT in the lab before installation; Site Acceptance Testing (SAT) for physical connections; I/O Verification for signal routing; Site Integration Testing (SIT) for protection logic with all connected systems; and Final Integration Testing (FIT) for full-chain verification including beam. At each new commissioning phase, new SCU and DLN firmware bitstreams are generated with updated parametrization, and commissioning activities must be repeated for new connections while verifying that previously connected systems remain unaffected.

B. Factory Acceptance Testing

FAT has benefited most from the unified verification approach. The hardware tester emulates all DLN and SCU inputs and observes all outputs via Python EPICS libraries, enabling Pytest to serve as the hardware validation framework with the same structure, parametrization, and reporting used in simulation. The current FBIS test suite consists of approximately 5000 test cases executed with Cocotb and the Questa HDL simulator, achieving 100% statement code coverage of the interlock logic and providing coverage comparable to the original Hardware-In-the-Loop framework from ZHAW [14]. Verification results are stored in the ESS document management system, under version control, and referenced in system test reports as part of the ESS facility baseline.

C. Site Integration Testing

SIT consists of two main components. I/O tests verify that each signal from each Sensor System is connected to the correct FBIS input and processed by the correct PU type with the correct index; when a signal state changes, the test verifies the expected PU is affected. Functionality tests verify that protection logic operates correctly when integrated with all connected systems, primarily confirming that the GBP is removed in the correct circumstances.

For systems with numerous interfaces, such as RF Local Protection Systems (RFLPS), Beam Position Monitors (BPM), Beam Loss Monitors (BLM), and Beam Current Monitors (BCM), I/O testing is automated with Python and Pytest. The test structure mirrors that used in FAT and in Cocotb simulation: the same modular approach of parametrized scenarios, shared drivers and scoreboards, and JUnit XML reporting applies throughout. This consistency reduces the engineering effort required at each commissioning phase and ensures that the same test logic that validated firmware in simulation can be reused to verify signal routing on site.

D. Commissioning Results and Statistics

Earlier FBIS commissioning phases through the Normal Conducting Linac in 2023 are reported in [5]; the deployment and commissioning through the Beam on Dump are described in [6]. Post-mortem analysis of the Beam on Dump commissioning phase (January 2026 to the time of writing, approximately five months) recorded approximately 6 000 trips. Of these, 2 382 were deliberate operator beam inhibits, issued through the control system interface for machine configuration and studies. Among the remaining system-triggered trips, BCMs and RF systems are the two dominant contributors at approximately 49% and 42% respectively. The RF share includes trips generated during conditioning activities. The contributions of BCMs, MPS PLCs, and the Software Interlock System (SIS) are partly driven by PBM/PBD synchronization transients that arise when operators change beam configuration, during which some systems briefly lag the new beam mode or destination setting. BPMs contribute approximately 4%. Aperture monitors and the dump imaging system (AptM/IMG) contribute at the percent level. LEBT and

MEBT choppers, Ion Source PLC, and BLMs are negligible contributors. All counts include entries accumulated during maintenance periods or deliberate functional tests and should be interpreted in that context.

VIII. Conclusions and Outlook

The Fast Beam Interlock System at the European Spallation Source has been deployed and operated successfully through multiple commissioning phases, from the first proton beam in 2021 to Beam on Dump in 2025–2026. The system has remained fully operational throughout, protecting the machine, meeting its latency requirements, and supporting the commissioning process. The modular design approach, applied consistently across hardware platforms, firmware parametrization, verification methodology, diagnostic systems, and control-system integration, has been a key enabler of this outcome: it allowed a small team to manage tens of firmware variants under a single verification framework, reuse the same hardware platform for both the operational system and the FAT tester, share test logic across simulation and hardware environments, and adapt to each commissioning phase without rebuilding infrastructure from scratch.

A less obvious benefit of the mTCA platform choice was organizational: by adopting the same infrastructure as the broader ESS ecosystem, the FBIS team delegated responsibility for common AMC boards (MCH, CPU, EVR) to a dedicated ICS infrastructure group, freeing it to concentrate engineering effort on the FBIS-specific FPGA firmware. The same logic applied to the FAT tester, which was built from the same board types as the operational DLN crate; the investment in tester development was correspondingly small, and its fidelity to the real system was correspondingly high.

The unified firmware verification framework, combining Cocotb for simulation with Pytest for hardware testing through a shared Python infrastructure, has improved test reusability and reduced the effort required to develop and maintain test suites across tens of firmware variants. The most significant efficiency gains have been in FAT. A broader lesson from the verification work is that components perceived as non-critical are still worth testing systematically from the start: the History Buffer, initially undertested, required reactive debugging when an operator interface exposed latent firmware bugs. It has since matured into an operationally valuable diagnostic tool. The super-HB capability across all DLNs has demonstrated the feasibility of multi-crate post-mortem analysis, and a dedicated analysis tool with a Python backend and web frontend is already in use by experts. Operator-facing abstractions proved equally important: the high-level EPICS IOC, which presents FBIS hardware redundancy as single per-system status indicators, substantially reduced the expertise barrier during commissioning and improved system availability.

The current FBIS hardware and firmware configuration is designed to support the full facility scope: it should require no hardware or firmware modifications to protect the Beam on Target phase, the final and highest-power commissioning milestone at ESS. Looking ahead, several software and tooling improvements are nonetheless planned. Post-mortem analysis tooling will be enhanced to merge all FBIS HBs (including SCUs), automatically extract the root cause of beam stops, and provide machine availability statistics. SCU HB synchronization will be improved. The simulation test suite will be expanded to increase Cocotb coverage alongside the hardware-focused test suite. Optical fiber power monitoring will be introduced to detect degradation of the Slink and OPL links before they affect availability. Longer term, migrating the DLN register interface from TOSCA to the AXI4-based ESS FPGA Framework would align FBIS with the broader ESS FPGA ecosystem and remove a dependency on a proprietary bus, improving long-term maintainability and enabling use of the wider ESS toolchain for DLN firmware development. The modular firmware approach also opens the door to exploiting mature external FPGA IP libraries such as SURF (SLAC) [15] and Colibri (CERN) [16], which provide reusable, well-tested protocol and infrastructure cores. Similarly, build-system tools such as HOG [17] and FuseSoC [18] could further streamline firmware project management and synthesis flows across the growing number of FBIS firmware variants.

Acknowledgment

The authors gratefully acknowledge the contributions of the team at ZHAW (Zurich University of Applied Sciences), comprising the Institute of Applied Mathematics and Physics (IAMP) and the Institute of Embedded Systems (InES), in the conceptualization, hardware and software development, and testing of SCUs and DLNs. The authors also acknowledge IOxOS Technologies for the IFC1410 AMC boards, and the ESS ICS Infrastructure team for the standardized mTCA deployment tools that benefit the FBIS team. The authors used AI-assisted tools for grammar and language editing during the preparation of this manuscript.

References

[1] R. Garoby *et al.*, "The European Spallation Source Design," *Phys. Scr.*, vol. 93, no. 1, p. 014001, Dec. 2017, doi:10.1088/1402-4896/aa9bff.

[2] S. Gabourin, A. Nordt, and S. Pavinato, "Implementation of a VHDL Application for Interfacing Anybus CompactCom," in *Proc. ICALEPCS2021*, Shanghai, China, 2021, doi:10.18429/JACoW-ICALEPCS2021-WEPV041.

[3] S. Pavinato, E. D'Costa, and S. Gabourin, "FPGA Firmware Verification: a common approach for simulation and hardware tests," in *DVCon Europe 2025*, Munich, Germany, 2025.

[4] S. Gabourin *et al.*, "The ESS Fast Beam Interlock System History Buffer for Post-Mortem Analysis and Accelerator Statistics," in *Proc. ICALEPCS2025*, Chicago, IL, USA, 2025, doi:10.18429/JACoW-ICALEPCS2025-THPD018.

[5] S. Pavinato, M. Carroll, S. Gabourin, A. Gorzawski, and A. Nordt, "The ESS Fast Beam Interlock System—Design, Deployment and Commissioning of the Normal Conducting Linac," in *Proc. ICALEPCS2023*, Cape Town, South Africa, 2023, doi:10.18429/JACoW-ICALEPCS2023-TUPDP081.

[6] S. Gabourin *et al.*, "The ESS Fast Beam Interlock System—Design, Deployment and Commissioning till the Beam on Dump," in *Proc. ICALEPCS2025*, Chicago, IL, USA, 2025, doi:10.18429/JACoW-ICALEPCS2025-THPD017.

[7] F. Chicken, J. J. Jamróz, and J. P. S. Martins, "Development of Standard MicroTCA Deployment at ESS," in *Proc. ICALEPCS2023*, Cape Town, South Africa, 2023, doi:10.18429/JACoW-ICALEPCS2023-THMBCMO21.

[8] IOxOS Technologies. [Online]. Available: https://www.ioxos.ch

[9] Micro-Research Finland Oy. [Online]. Available: https://www.mrf.fi

[10] Concurrent Technologies. [Online]. Available: https://www.gocct.com
[11] C. Amstutz *et al.*, "The ESS FPGA Framework and its Application on the ESS LLRF System," in *LLRF Workshop*, 2018, arXiv:1803.09044.
[12] *PSS 3.0 Language Reference Manual*, Accellera Systems Initiative.
[13] EPICS Controls. [Online]. Available: https://epics-controls.org/
[14] ZHAW Zürcher Hochschule für Angewandte Wissenschaften. [Online]. Available: https://www.zhaw.ch/engineering
[15] SURF FPGA Library, SLAC National Accelerator Laboratory. [Online]. Available: https://github.com/slaclab/surf
[16] Colibri FPGA Library, CERN. [Online]. Available: https://gitlab.cern.ch/colibri/colibri
[17] HOG (Hdl On Git), CERN. [Online]. Available: https://gitlab.com/hog-cern/Hog
[18] FuseSoC. [Online]. Available: https://github.com/olofk/fusesoc