

# SoC TECHNOLOGY FOR EMBEDDED CONTROL AND INTERLOCKING WITHIN FAST PULSED SYSTEMS AT CERN

P. Van Trappen\*, E. Carlier, M. Gauthier, N. Magnin, E. J. Oltedal, J. Schipper  
CERN, Geneva, Switzerland

## Abstract

The control of pulsed systems at CERN requires often the use of fast digital electronics to perform tight timing control and fast protection of high-voltage (HV) pulsed generators. For the implementation of such functionalities, a field-programmable gate array (FPGA) is the perfect candidate for the digital logic, however with limited integration potential within the control system.

The market push for integrated devices, so called System on a Chip (SoC), i.e. in our case a tightly coupled ARM processing system with specific programmable logic in a single device, has allowed a better integration of the various components required for the control of pulsed systems. This technology is used for the implementation of fast switch interlocking logic, integrated within the CERN control framework by using embedded Linux running a Snap7 server. It is also used for the implementation of a lower-tier communication bridge between a front-end computer (FEC) and a high fan-out multiplexing programmable logic for timing and analogue low-level control.

This paper presents these projects where SoC technology has been used, proposes system deployment & booting solutions and discusses possible further applications within distributed real-time control architecture for distributed pulsed systems.

## SYSTEM ON A CHIP TECHNOLOGY

A SoC is an integrated circuit that tightly integrates several components and peripherals of an electronic system, which were once seen as separate chips on a printed circuit board (PCB). The focus of this paper is on SoCs that integrate programmable logic (PL) with one or more computing processors, from now on referenced as the processing system (PS). Most commonly used are ARM® architecture processors which are well known for their use in mobile products because of low cost and efficient power consumption. The PS uses processors that are supported by the free and open-source software (FOSS) Linux mainline kernel which eases the development of software applications.

The programmable logic is composed of the typical FPGA cells and group functionality together in so-called Intellectual Property (IP) cores. These interconnect easily to the PS using various types of the AMBA® ARM® AXI4 protocol buses, enabling e.g. high-performance memory-mapped or streaming data transfer. Custom logic that is implemented in these cells can be clocked at frequencies higher than 100MHz for fast and deterministic behavior.

\* CERN TE-ABT-EC, pieter.van.trappen@cern.ch

This is a strong advantage but FPGAs often suffer from integration problems, i.e. mainly communication-wise, into existing control systems and middle-ware. Now having a closely coupled PS to configure and treat the PL system data makes that integration almost straight-forward. In addition these SoCs embed various additional peripheral cores (e.g. Ethernet, I2C) with mainlined drivers which significantly eases the PCB design while reducing complexity during gateware and software development.

## SWITCH FAST INTERLOCK DETECTION SYSTEM PROJECT

### Project Description

Pulsed kicker magnet systems are powered by high-voltage and high-current pulse generators with adjustable pulse length and amplitude. To deliver this power, fast high-voltage switches such as thyratrons or GTO stacks are used to control the fast discharge of pre-stored energy. To protect the machine and the generator itself against internal failures of these switches, a modular digital Fast Interlock Detection System (FIDS) has been developed for the consolidation of existing systems. A Xilinx Zynq®-7000 SoC has been selected for implementation of the required functionalities [1].

HV switch malfunctioning is referred to as *fast interlocks*, which need to be detectable over the system's full dynamic range. In addition to interlocking, they have often a corrective action such as pulsing other magnets in order to fully deflect the beam and to avoid beam losses, thus to protect the machine.

Before the 2020 restart of CERN's accelerator complex, the FIDS will be installed at four places: at the PS Booster distributor, PS injection, SPS dump extraction and LHC injection installations.

### Implementation

High bandwidth pick-up signals (>100MHz) are sampled from a HV pulse generator, for which full-scale digitizing would not be cost-effective and also not required for the FIDS functionality. Instead fast discrete comparators are used to produce a 1-bit signal that indicate the presence of a current or voltage. This simplification will not limit the FIDS' performance because the signal amplitude itself is generally of no importance. It is important however that the signal-to-noise ratio (SNR) is high for the full dynamic range of the input signals. This was implemented by having the comparator reference follow the generator's pulse forming network (PFN) charge in order to fix the threshold level to e.g. 50% of its flattop. Certain generators operate with PFN charges between 10kV and 80kV and the pick-up signals

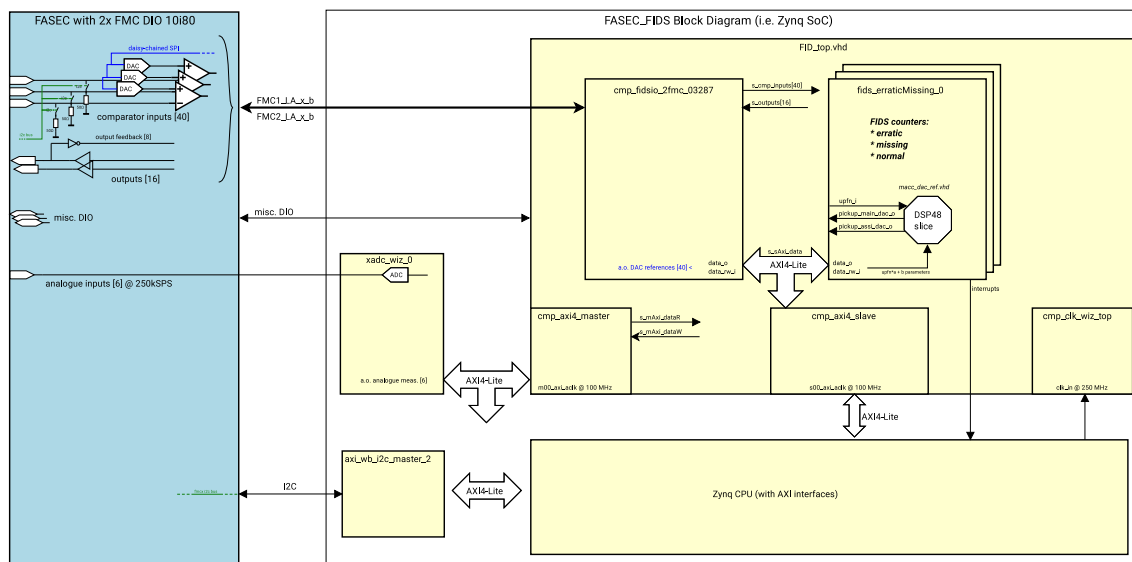


Figure 1: Gateware block diagram.

amplitude has a direct linear relation to this charge. This linear scaling is implemented in the PL while the user sets among others the offset and multiplier through the PS. An additional advantage of the comparator-based approach over digitizers is that additional latency due to ADC conversion and communication is avoided.

### Programmable Logic

All interlock detection and reaction logic is implemented in gateware, using VHDL. The logic is split into functional blocks and designed into separate modules to encourage reuse between projects.

**Hardware** Specifically designed for this project but with reuse by other laboratories in mind, the following hardware has been designed:

- FPGA & ARM SoC FMC Carrier (FASEC), a carrier for two low pin count FPGA Mezzanine Cards (VITA 57) with additional 200 kSPS bipolar analog inputs, Ethernet connectivity and fail-safe functionality [2].
- FMC DIO 10i80, an I/O card with a low-pint count (LPC) connector housing 10 inputs with each two fast differential comparators (propagation delay < 1 ns) and 20 12-bit DAC channels for comparator threshold setting [3].

**Modular Approach** As stated earlier, the FIDS will be installed into a number of kicker system installations that are all different, architecture- and hardware-wise. The required functionality however is the same so the gateware components were designed as independent modules which are then instantiated as required by the installation. All design-files are stored on CERN's Git repositories and the Git submodule functionality is leveraged to reuse this library

of FIDS modules. See Fig. 1 for an configuration template, where the following modules are being used:

- fidsio\_2fmc\_3287: FASEC internal FMC Input/Output management for the FMC DIO 10i80 cards (ref. EDA-03287)
- fasec\_io\_interlocks: FASEC external Input/Output management, including interlocks
- fids\_erraticMissing: normal, erratic and missing shots detection
- fids\_short: magnet or transmission cable short-circuit detection
- fids\_slowVoltageCheck: 12-bit based XADC over- and under-voltage detection

**AXI4 Interface** As shown in Fig. 1, all IPs are currently connected using the AXI4-Lite bus, which is recommended for simple and low-throughput memory-mapped communication. It currently does its job flawlessly but the design is expected to be ported to AXI-Stream for the XADC interface and for a new interface to a PS AXI Direct Memory Access (DMA) unit. That interface will allow streaming all comparator 250MSPS samples to the PS and CERN logging database be used for advanced event diagnostics.

### Processing System

The PS has direct access to all PL registers and the Ethernet peripheral is used to forward certain registers to external users (e.g. equipment experts, CERN Control Room). These registers can be used for equipment configuration and diagnostic feedback. Several functionalities have been implemented on the PS, which is running Embedded Linux, based on the Yocto Project® but configured and compiled using the Xilinx Petalinux Tools.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2019). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

**Cheby as Gateway-Software API** The features of a SoC are virtually endless and can lead to a complex system with 100s to 1000s of registers configured. For efficient development a single memory map definition should auto-generate both gateway constants and software headers with the addressing constants and info. A new and FOSS tool developed at CERN, Cheby [4] was used for this purpose.

**FASEC Kernel Module** A Linux kernel driver module has been written to interface the custom gateway to the PS user space. It registers the gateway FIDS memory and has interrupt service routines to, among others, update the active patch-panel over an external I2C bus (i.e. IO status to LEDs, LCD screen).

**Integration with Other Control Systems** Generally a SoC offers a wide variety of interfaces but a raw interface doesn't integrate directly into the CERN Control system. After discussion with the BE-CO Controls Group, the SILECS framework<sup>1</sup> was selected as the best candidate for the job because of an existing code-base and libraries for FESA3 (Front-End Software Architecture) [5] and middleware used at CERN. The so-called SILECS virtual controller, a server daemon, was ported to ARM® architecture for this project. It runs on the SoC and allows register access by using the Snap7 [6] protocol. An important advantage is that this is a native Siemens PLC protocol which integrates nicely with the existing TE-ABT generator slow control PLCs. The SoC server daemon handles several clients to allow configuration and status readback.

**Automated Test Procedure** Once more the power of a SoC was leveraged to produce an automated test procedure for the project card series production (FASEC and FMC DIO 10i8o, see higher). A Keysight 34972A Data Acquisition Switch Unit was used as a programmable signal generator, able to multiplex a bipolar analogue voltage to all I/Os, controlled over LAN by using industry-standard Standard Commands for Programmable Instruments (SCPI). The test procedure consists of Python scripts that run on the SoC itself, controlling internal and external interfaces and performing self-checks using libraries such as pyVISA and unittest. A single card functionality tests now only requires a few minutes and outputs a full test report.

## G-64 PULSE GENERATOR INTERFACE REPLACEMENT PROJECT

### Project Description

**Motivation** The G-64 hardware and software used as an interface between the CERN PS Complex (CPS) pulse generators and the FEC for beam transfer has become obsolete and no longer supported [7]. An single-board controller with a Xilinx Zynq®-7000 SoC, the sbRIO-9607 from National

<sup>1</sup> Communication libraries for interfacing industrial controllers and field-buses within the standard CERN BE-CO infrastructure (SILECS – Software Infrastructure for Low-level Equipment Controllers)

Instruments (NI), was introduced to replace this hardware. This project makes an interesting comparison to the previous one because the sbRIO was purchased off-the-shelf and instead of using the standard embedded toolchain, NI LabVIEW™ was used to some extend for both PL and PS.

**sbRIO Control System** The implemented control system is divided in two parts, the real-time task in the PS and the FPGA configuration in the PL. The real-time task is event driven and waits for events coming from the FEC, in a master-slave scheme because one FEC controls several pulse generators housing the sbRIO as shown in Fig. 2. The real-time task communicates with the FPGA, which can execute many control algorithms in a single cycle of the FPGA clock (40 MHz). The FPGA operates with true parallel execution and has dedicated hardware resources for each task, rather than scheduling tasks and sharing resources, like in processor-based systems.

### HV Pulse Generator Hardware Configuration

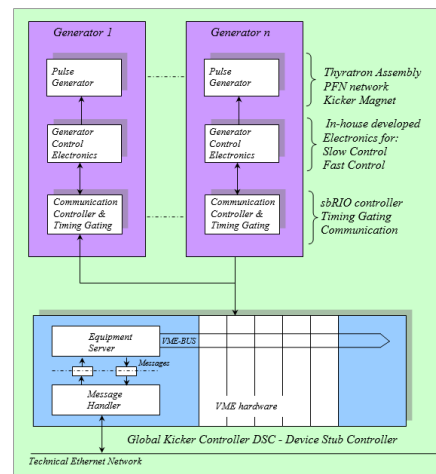


Figure 2: Pulse Generator Control Layout.

A beam transfer pulse generator consists commonly of a PFN and HV switch, e.g. thyatron, to transfer its charge to a kicker magnet. The pulse generator and generator control electronics are typically configured to service the various sub-components and contain the following functionality:

- Thyatron timing and delays
- PFN reference and acquisition voltages
- PFN charging/discharging protection
- Interlock and thyatron protection
- Power control for all the different units

The sbRIO controller forms the communication layer and is implemented in between the generator hardware control and the FEC. Its basic functions are:

- Receiving control message sent by the FEC (kick strength & module control),

- Returning acquiring status and voltage acquisition from the generator,
- Programmable timing gate for precise pulse control.

This sbRIO-9607 controller integrates a real-time processor running NI Linux Real Time, a user re-configurable FPGA and I/O on a single PCB. This architecture allows tailor-made solutions in this case for the control and acquisition of the CPS pulse generators, the link between the Front-end computer (FEC) and the pulse generator electronics. The FPGA functionality allows adapting to the needs of the different generator installations without changing any hardware (Fig. 3).

### Programmable Logic

The FPGA linked to the generator hardware performs timing control tasks, PFN voltage measurements, reference voltage settings, interlock and status readout and generator control functions. The FPGA returns the generator hardware data to the real-time task, which in its turn returns the data to the FEC for operators and experts control and read-back.

### Toolchain

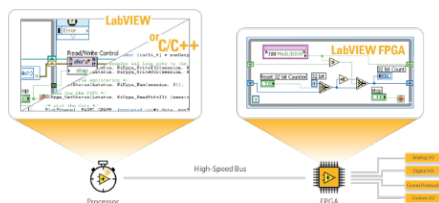


Figure 3: sbRIO programming options.

The FPGA is programmed in NI LabVIEW™ and the real-time task has been programmed first in LabVIEW™, then ported to C++ later on. Choosing LabVIEW™ to program the real-time application has advantages in that its Real-Time Module includes built-in constructs for multithreading, real-time thread scheduling, and many other features specific to building robust, deterministic code. Furthermore, communication tools with the FPGA are fully integrated in LabVIEW™.

C++ is wider adopted, text based and an industry standard which is why later on the code has been ported. LabVIEW™ is now used only for the FPGA logic programming.

### Processing System

NI provides tools to ease the communication between software and gateway. The toolset is composed of two parts, one to compile the NI FPGA project into a bitstream file, the second one to generate a C API to embed in a custom software project. This generated API allows the software to access in read and write directions the registers defined in the FPGA project.

The board runs NI Linux Real-Time, a pre-compiled Embedded Linux version by NI with patches for real-time support. The software developed is cross-compiled and then

executed on the SoC after mounting or copying-over the files.

### Software Design

Concerning the embedded software running on the ARM cores of the Zynq SoC, as seen previously its purpose is to interface the FEC, which is running high level real-time software using FESA [5], with the generator hardware.

**Libraries** The FEC and SoC communication is historically based upon RS-232 protocol, which has been kept in place for the time being to ease the transition. A specific library was developed to build and exchange the different messages depending on the hardware type. It is built around standard Linux libraries, which allow us to use it on both x86-64 and ARM architectures.

Non-standard libraries have been used as well, e.g. boost shared pointer, in this project. These have been cross-compiled for the SoC ARM architecture. The CERN logging library cmw-logger has also been ported to ARM, to allow connecting to the logging system used all across CERN.

**Architecture** The final code was developed using the *Factory Method* design pattern [8]. Each generator hardware has its own FPGA implementation and RS-232 messages. The hardware selection happens at runtime, which loads the desired gateway configuration and messaging implementation. It allows us to develop and maintain a single software program which is able to connect to the different generators available to operation.

The embedded software can receive from FEC three types of commands:

- Set kick and voltage settings for the generator hardware.
- Read after pulse acquisitions including voltages delivered, generator status and interlocks.
- Set generator hardware with a new status command: ON, OFF, STANDBY or RESET.

Once the command is received from the FEC and decoded, the program will read or write the concerned FPGA registers and eventually send back a message to the FEC in case of reading acquisition. The main program is then quite simple, once the correct gateway configuration is loaded it continuously waits for a RS-232 message and executes the associated command.

## OPERATIONAL DEPLOYMENT & BOOTING SOLUTIONS

An important advantage of the presented SoC systems is the availability of an Embedded Linux Operating System (OS), much like the FECs used to control the machine. It allows us to approach the operational deployment from different points of view and have more than one possibility for integration in the CERN Technical Network (TN).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2019). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

We can either consider this hardware as completely independent and standalone systems or as fully connected Linux computer using the same deployment tools as regular FECs. Several flavors are possible depending on projects constraint and external team cooperation.

Below is the description of the three different scenarios we are currently investigating:

- **Fully embedded:** The hardware is considered as standalone and works on his own. It won't be connected at all to TN.

Each operational board is prepared upfront, the OS and software will be flashed before on site installation.

Logging data, debug access and software/OS update are only possible by connecting directly to the hardware deployed on site.

This scenario is on one hand the simplest but on the other hand it's also the less convenient to maintain on long term, e.g. in case of bug-fixing we will be obliged to update all boards on site; also OS and software version tracking will require a lot of carefulness.

- **Minimal support:** The hardware is still considered as standalone, meaning the OS and software will remain embedded on-board. However the board will be connected to TN, and a SSH server will remain open. This solution allows us to reduce maintenance work by allowing remote access to programmers. It will be possible to inspect the system remotely in case of debugging and thanks to secure copy through ssh (scp) to install software updates. Also some tasks, such as software update roll-out, can be easily scripted among the deployed devices.

To summarize, this mix requires less maintenance than the previous solution but still encounters some limitations as maintaining and tracking the installed OS and software and also the development or usage of non CERN standard tools.

- **Maximal support:** The hardware is fully integrated with CERN services, it will be TN connected and integrate a TFTP server to boot the OS, SSH access for maintenance purpose, and also NFS and SNMP access. Centralised monitoring and syslog support will complete the picture as make the Soc board look like a common FEC.

This solution would be ideal for equipment groups, however it's the most difficult to realize since it requires significant work upfront to examine the different possibilities to achieve those requirement. Strong support from other teams, especially BE-CO are required to handle common-interest sources such as compiler and kernel sources.

## CONCLUSIONS AND OUTLOOK

The above projects show that a FPGA-based SoC is a powerful addition to an engineer's tool-set, often shortening

PCB design times and providing a wide flexibility for configuration. However such a flexibility comes at a price, i.e. the extended knowledge required for PL and PS in addition to the complexity of the development software packages. Both Xilinx and NI are doing a good job in abstracting some of these with libraries and tools. An advantage of the Xilinx PetaLinux tools are that they are based on common FOSS tools used by all embedded developers, not limited to SoCs. An advantage of the NI LabVIEW™ environment is that most complexity is abstracted away at the price of using a single-vendor proprietary tool.

Recent years have seen an increase of SoC used for low-level control systems, and it is a component that's there to stay and to be further used in the future. Example projects in the making are a CERN-developed *Distributed IO Tier* where the non radiation-tolerant System Board will house a Zynq UltraScale+ [9] and the ESRF Zynq Carrler for RFoWR-based Timing sYstems [10]. Optimised integration within accelerator control infrastructure remains nevertheless a challenge both in term of computing security and of re-validation when equipment safety functionalities are included either at PL or PS level.

## REFERENCES

- [1] P. Van Trappen, E. Carlier, and S. Uyttenhove, "A Fast Interlock Detection System for High-Power Switch Protection", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 367–370. doi:10.18429/JACoW-ICALEPCS2015-MOPGF122
- [2] FASEC on OHWR, <https://www.ohwr.org/project/fasec/wikis/home>
- [3] FMC DIO 10i 8o on OHWR, <https://www.ohwr.org/project/fmc-dio-10i-8o/wikis/home>
- [4] cheby, <https://gitlab.cern.ch/cohtdrivers/cheby>
- [5] L. Fernandez *et al.*, "Front-End Software Architecture", in *Proc. ICALEPCS'07*, Oak Ridge, TN, USA, Oct. 2007, paper WOPA04, pp. 310–312.
- [6] Snap7, <http://snap7.sourceforge.net/>
- [7] D. Calcoen, "The G-64 bus at CERN after 25 years of operation", in *Proc. ICALEPCS'05*, Geneva, Switzerland, Oct. 2005, paper PO1.095-8.
- [8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, 1994.
- [9] G. Daniluk and E. Gousiou, "Low-Cost Modular Platform for Custom Electronics in Radiation-Exposed and Radiation-Free Areas at CERN", presented at ICALEPCS'19, New York, NY, USA, Oct. 2019, paper TUAPP03, this conference.
- [10] CITY on OHWR, <https://ohwr.org/project/city/wikis/home>