A strategy for managing diverse equipment in the CERN controls group

Javier Serrano, Juan David González Cobas

FOSDEM 2012
Find out about what the world is made of...

**STANDARD MODEL**

**MATTER**

**ATOM**
- Electron
  - Electric charge: -1
  - Responsible for electricity and chemical reactions

**NUCLEUS**
- Neutron
- Proton
- Electron

**QUARK**
- **LEPTONS**
  - **Electron**
    - Electric charge: -1
    - Responsible for electricity and chemical reactions
  - **Electron Neutrino**
    - Electric charge: 0
    - Rarely interacts with other matter
  - **Muon**
    - A heavier relative of the electron
  - **Muon Neutrino**
    - Created with muons when some particles decay
  - **Tau**
    - Heavier still
  - **Tau Neutrino**
    - Not yet observed directly

- **QUARKS**
  - **Up Quark**
    - Electric charge: 2/3
    - Protons have 2 up quarks
    - Neutrons have 1 up quark
  - **Down Quark**
    - Electric charge: -1/3
    - Protons have 1 down quark
    - Neutrons have 2 down quarks
  - **Charmed Quark**
    - A heavier relative of the up
  - **Strange Quark**
    - A heavier relative of the down
  - **Top Quark**
    - Recently observed
  - **Bottom Quark**
    - Heavier still

**ANTIMATTER**
- Each particle also has an antimatter counterpart ... sort of a mirror image.
...and what other worlds might be made of!

- Positron
- Antiproton
- Antineutron
- Antimuon
- Antitau
- Anti-up quark
- Anti-down quark
- Anti-strange quark
- Anti-charm quark
- Anti-bottom quark
- Anti-top quark
Better microscopes for biologists and other scientists

- millimetre: $10^{-3}$
- micron: $10^{-6}$
- angstrom: $10^{-10}$

- Cell
- Bacterium
- Water Molecule
- Protein

- INFRARED
- ULTRAVIOLET
- "SOFT" X RAYS
- "HARD" X RAYS

Javier Serrano, David Cobas et al. (CERN)
Better microscopes for biologists and other scientists
E.g. Mouse brain to study Alzheimer
Better microscopes for biologists and other scientists

E.g. Nucleosome core
Fight tumors more effectively

Hadron therapy

Proton Therapy

© 2006 Midwest Proton Radiotherapy Institute.
Fight tumors more effectively
Better energy deposition than X-rays and traditional radiotherapy
A simple synchrotron

- Radiofrequency cavity
- Bending magnet
- Focusing magnet
- Vacuum chamber
- Collisions
- Injection
- Protons (p)
LEP superconducting cavity
CLIC cavity

Javier Serrano, David Cobas et al. (CERN)

Diverse equipment at CERN

FOSDEM 2012
Drift Tube Linac (DTL)
A simple dipole electromagnet
LHC cryodipole

The 15-m long LHC cryodipole

Superconducting Coils
Beam Pipe
Heat Exchanger Pipe
Helium-II Vessel
Superconducting Bus-Bar
Iron Yoke
Non-Magnetic Collars
Vacuum Vessel
Radiation Screen
Thermal Shield

Spool Piece Bus Bars
Quadrupole Bus Bars
Protection Diode
Auxiliary Bus Bar Tube
Instrumentation Feed Throughs
ATLAS in reality

Javier Serrano, David Cobas et al. (CERN)

Diverse equipment at CERN
Jet Event at 2.36 TeV Collision Energy

2009-12-14, 04:30 CET, Run 142308, Event 482137
CERN Accelerator Complex

CERN's accelerator complex

- LHC (Large Hadron Collider)
- SPS (Super Proton Synchrotron)
- PS (Proton Synchrotron)
- AD (Antiproton Decelerator)
- CTF3 (Clic Test Facility)
- CNGS (CERN Neutrinos to Gran Sasso)
- ISOLDE (Isotope Separator OnLine DEvice)
- LEIR (Low Energy Ion Ring)
- LINAC (Linear ACcelerator)
- n-ToF (Neutrons Time Of Flight)

- p (proton)
- ion
- neutrons
- p (antiproton)
- proton/antiproton conversion
- neutrinos
- electron

© CERN 2008
CERN Beams Controls group

Responsible for

- Controls infrastructure for all CERN accelerators, transfer lines and experimental areas
- General services such as synchronization and analog signal acquisition/display
- Specification, design, procurement, integration, installation, commissioning and operation

Supports

- Beam instrumentation, cryogenics, power converters etc.
Beams Controls standard kit

Hardware kit

- analog and digital I/O
- level converters, repeaters
- serial links, timing modules

Software

- Linux device drivers, C/C++ libraries, test programs
Two bus standards

- VME64x
  - 6U, large front-panel space, may use rear transition module
- PICMG 1.3
  - Industrial type PC with the processor on a plug-in board
  - Internal buses PCI Express and PCI

Need for a mezzanine approach

- Functions (e.g. ADC, TDC) are needed for both buses
- Would need twice as many designs, more if additional standards are needed (PXIe, xTCA)
Carriers and mezzanines

Courtesy of VITA: http://www.vita.com/fmc.html
Advantages of the carrier/mezzanine approach

Re-use
- One mezzanine can be used in VME and PCIe carriers.
- People know standards, more likely to re-use or design for it.

Reactivity
- Carrier: place and route a complex FPGA/Memory PCB once.
- Mezzanine: small and easier to route cards, easy assembly.

Rational split of work
’Controls’ can design the carrier, ’Instrumentation’ an ADC mezzanine, ’RF’ a DDS one, etc.
Example of a PCI Express FMC carrier (SPEC)
Example of FMC mezzanine: 100 MSPS 14-bit 4-channel ADC
Another example of FMC mezzanine: 5-channel 1ns TDC
Inside the FPGA: Wishbone

- System becomes pretty complex: System-on-a-chip
- Build up from re-usable HDL cores
- Connect blocks with Wishbone bus
  - open standard
  - simple address/data bus
  - extended with pipelined mode
  - many cores already available

- We developed a design infrastructure
  - **WBGen**: scripts to automatically generate Wishbone slave HDL and documentation
  - **SDWB**: IP blocks with descriptors to enable software re-use
  - **HDLMake**: support to synthesize and simulate designs with distributed sources
  - **Etherbone**: a bridge between Ethernet and Wishbone (made by GSI)
There is an OSHW definition!

Check out http://freedomdefined.org/OSHW

- Inspired by the Open Source definition for software.
- Focuses on ensuring freedom to study, modify, distribute, make and sell designs or hardware based on those designs.
- Now we know exactly what we mean when we say OSHW!
Why we use Open Hardware

Peer review
Get your design reviewed by experts all around the world, including companies!

Design re-use
When it’s Open, people are more likely to re-use it.

Healthier relationship with companies
No vendor-locked situations. Companies selected solely on the basis of technical excellence, good support and price.

Dissemination of knowledge
One of the core missions of CERN.
Open Hardware Repository – ohwr.org

A web-based collaborative tool for electronics designers

- Wiki, News
- File repository
- Issues management
- Mailing list

Fully open access

- All information readable by everyone, without registration

Server made itself of FOSS

- ChiliProject (a fork of Redmine)
- SVN/GIT for version management, integrated in OHR
### FMC Carriers
- VME64x, PCIe, AMC, VXS
- PXIe in the pipeline

### FMC Mezzanines
- ADC’s, sampling speeds: 200 kSPS, 100 MSPS
- TDC and Fine delay (resolution 1 ns)
- Digital I/O: 5 channels, 16 channels

Important: all of these are or will be **commercially** available.
## The best of both worlds

<table>
<thead>
<tr>
<th></th>
<th>Commercial</th>
<th>Non-commercial</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Open</strong></td>
<td>Winning combination, best of both worlds.</td>
<td>Whole support burden falls on developers. Not scalable.</td>
</tr>
<tr>
<td><strong>Proprietary</strong></td>
<td>Vendor lock-in.</td>
<td>?!</td>
</tr>
</tbody>
</table>
CERN Open Hardware License – ohwr.org/cernohl

Provides a solid legal basis

- Developed with Knowledge Transfer Group at CERN
- Inspired by FOSS licences
- Focused on both design and products
- Persistent as LGPL

Practical: makes it easier to work with others

- Upfront clear that anything you give will be available to everyone
- Makes it clear that anyone can use it for free
CERN Open Hardware License – ohwr.org/cernohl

Same principles as FOSS

- Anyone can see the source (design documentation)
- Anyone is free to study, modify and share
- Any modification and distribution under same licence
- Persistence makes everyone profit from improvements

Hardware production

- When produce: licensee is invited to inform the licensor
Example of mechanics licenced with the CERN OHL

Worm farm and rotocaster
Try to use FOSS tools for development

Tools: the last hurdle to sharing

- We already have a forge and a licence.
- Current proprietary CAD tools make it hard to share designs.

Current efforts

What is White Rabbit?

- Ethernet
  - synchronism
  - determinism
What is White Rabbit?

An **extension** to **Ethernet** which provides:

- **Synchronous mode** *(Sync-E)* - common clock for physical layer in entire network, allowing for precise time and frequency transfer.
- **Deterministic routing** latency - a guarantee that frame transmission delay between two stations will never exceed a certain boundary.

**A multi-lab, multi-company project**

- based on Open Hardware and FOSS
- with expanding user base
### Design goals

<table>
<thead>
<tr>
<th>Scalability</th>
<th>Up to 2000 nodes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>10 km fiber links.</td>
</tr>
<tr>
<td>Precision</td>
<td>1 ns time synchronization accuracy, 20 ps jitter.</td>
</tr>
</tbody>
</table>
White Rabbit Synchronism

General architecture
Monitor phase of bounced-back clock continuously.

Phase-locked loop in the slave follows the phase changes measured by the master.
White Rabbit Switch

- Central element of WR network.
- Fully custom design, done from scratch at CERN.
- 10 1000Base-X ports, capable of driving 10+ km of SM fiber.
- 200 ps synchronization accuracy.
System FPGA handles all frame processing.

Implements PTP stack and management functions (SNMP, Spanning Tree) inside a Linux embedded platform.
CERN’s FMC-based Hardware Kit:

- FMCs (FPGA Mezzanine Cards) with ADCs, DACs, TDCs, fine delays, digital I/O.
- Carrier boards in PCI-Express and VME formats.
- All carriers are equipped with a White Rabbit port.
Common clock in the entire network: no skew between ADCs.

- Ability to sample with different clocks via Distributed DDS.
- External triggers can be time tagged with a TDC and used to reconstruct the original time base in the operator’s PC.
We got ourselves a rather diverse hardware ecosystem...
Software for Diverse Equipment

We got ourselves a rather diverse hardware ecosystem... eeehh, sorry,
We got ourselves a rather diverse hardware ecosyst... eeehh, sorry, zoo
We got ourselves a rather diverse hardware ecosystem... eeehh, sorry, zoo

Different species of hardware animals...
We got ourselves a rather diverse hardware ecosyst... eeehh, sorry, zoo

Different species of hardware animals...
- analog I/O (ADCs, DACs, waveform generators)
We got ourselves a rather diverse hardware ecosystem... eeehh, sorry, zoo

Different species of hardware animals...

- analog I/O (ADCs, DACs, waveform generators)
- digital I/O
We got ourselves a rather diverse hardware ecosystem... eeehh, sorry, zoo

### Different species of hardware animals...

- analog I/O (ADCs, DACs, waveform generators)
- digital I/O
- timing boards (FD, TDC)
We got ourselves a rather diverse hardware ecosystem... eeehh, sorry, zoo

Different species of hardware animals...

- analog I/O (ADCs, DACs, waveform generators)
- digital I/O
- timing boards (FD, TDC)
- many others
We got ourselves a rather diverse hardware ecosyst... eeehh, sorry, zoo

Different species of hardware animals...

- analog I/O (ADCs, DACs, waveform generators)
- digital I/O
- timing boards (FD, TDC)
- many others

...of diverse breed
We got ourselves a rather diverse hardware ecosystem... eeehh, sorry, zoo

Different species of hardware animals...

- analog I/O (ADCs, DACs, waveform generators)
- digital I/O
- timing boards (FD, TDC)
- many others

...of diverse breed

- shiny brand new FMC boards
We got ourselves a rather diverse hardware ecosystem... eeehh, sorry, zoo

**Different species of hardware animals...**
- analog I/O (ADCs, DACs, waveform generators)
- digital I/O
- timing boards (FD, TDC)
- many others

**...of diverse breed**
- shiny brand new FMC boards
- legacy COTS and in-house stuff
We got ourselves a rather diverse hardware ecosystem... eeehh, sorry, zoo

Different species of hardware animals...

- analog I/O (ADCs, DACs, waveform generators)
- digital I/O
- timing boards (FD, TDC)
- many others

...of diverse breed

- shiny brand new FMC boards
- legacy COTS and in-house stuff

Some order has to be imposed in the zoo.
Unifying Themes

Wishbone-based drivers covering the FMC family of boards designed around the internal Wishbone bus introducing enumeration of device cores

zio: The Ultimate I/O Linux kernel framework for I/O devices oriented to high bandwidth applications

Javier Serrano, David Cobas et al. (CERN)
Unifying Themes

Wishbone-based drivers

- covering the FMC family of boards
- designed around the internal Wishbone bus
- introducing enumeration of device cores
Unifying Themes

### Wishbone-based drivers
- covering the FMC family of boards
- designed around the internal Wishbone bus
- introducing enumeration of device cores

### zio: The Ultimate I/O
- Linux kernel framework for I/O devices
- oriented to high bandwidth applications
The FMC family of boards

- carriers in PCIe and VME form factors
- simple mezzanines with electronics for ADCs, DACs, DIO and endless other applications
- circuitry in the mezzanine
- FPGA application logic in the carrier

*logic in the FPGA is organized as a set of IP cores interconnected through an internal Wishbone bus*
A typical data acquisition application: carrier

SPEC carrier board (http://www.ohwr.org/projects/spec)
A typical data acquisition application: mezzanine

FMC 100M4ch14b ADC
(http://www.ohwr.org/projects/fmc-adc-100m14b4cha)
Parts of the whole

Block diagram of the FMC ADC application.
Basic \( \text{I}^2\text{C} \) interfacing to the mezzanine board.

Block diagram of the FMC ADC application.
Parts of the whole

- Basic $I^2C$ interfacing to the mezzanine board.
- Wishbone mastering.

Block diagram of the FMC ADC application.
Parts of the whole

- Basic $I^2C$ interfacing to the mezzanine board.
- Wishbone mastering.
- DMA access to DDR3 memory in the carrier board.

Block diagram of the FMC ADC application.
Basic I²C interfacing to the mezzanine board.
- Wishbone mastering.
- DMA access to DDR3 memory in the carrier board.
- Mezzanine-specific control logic (e.g. ADC programming/setup).

Block diagram of the FMC ADC application.
Parts of the whole

- Basic \textsuperscript{1}C interfacing to the mezzanine board.
- Wishbone mastering.
- DMA access to DDR3 memory in the carrier board.
- Mezzanine-specific control logic (e.g. ADC programming/setup).
- Interrupt control.

Block diagram of the FMC ADC application.
Drivers for the FMC family

Design concepts

Javier Serrano, David Cobas et al. (CERN)

Diverse equipment at CERN

FOSDEM 2012
Design concepts

- modular structure that reflects the core structure of the firmware
Drivers for the FMC family

Design concepts

- modular structure that reflects the core structure of the firmware
- one-to-one mapping driver ↔ core (usually)
Drivers for the FMC family

Design concepts

- modular structure that reflects the core structure of the firmware
- one-to-one mapping driver ↔ core (usually)
- ability to dynamically load bitstreams by application
Drivers for the FMC family

Design concepts
- modular structure that reflects the core structure of the firmware
- one-to-one mapping driver ↔ core (usually)
- ability to dynamically load bitstreams by application

Nothing new under the FOSS sun
Drivers for the FMC family

Design concepts

- modular structure that reflects the core structure of the firmware
- one-to-one mapping driver $\leftrightarrow$ core (usually)
- ability to dynamically load bitstreams by application

Nothing new under the FOSS sun

- On the whole, the driver for the carrier board acts as a basic firmware loader and a bridge driver (with device enumeration à la PCI) between the host bus (PCIe, VME) and the FPGA interconnection bus
Drivers for the FMC family

Design concepts
- modular structure that reflects the core structure of the firmware
- one-to-one mapping driver $\leftrightarrow$ core (usually)
- ability to dynamically load bitstreams by application

Nothing new under the FOSS sun
- On the whole, the driver for the carrier board acts as a basic firmware loader and a bridge driver (with device enumeration à la PCI) between the host bus (PCIe, VME) and the FPGA interconnection bus
- It will be (we hope) the first Wishbone bus driver in the mainstream kernel
Architecture of the FMC drivers

Identify the carrier board and initialize it.
Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.
Load the application firmware into the carrier FPGA.
Register a Wishbone bus with the kernel.
Enumerate the cores in that firmware.
Register the devices those cores implement and install the drivers associated to them.
Identify the carrier board and initialize it.

Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.

Load the application firmware into the carrier FPGA.

Register a Wishbone bus with the kernel.

Enumerate the cores in that firmware.

Register the devices those cores implement and install the drivers associated to them.
Architecture of the FMC drivers

- Identify the carrier board and initialize it.
- Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.
- Load the application firmware into the carrier FPGA.
- Register a Wishbone bus with the kernel.
- Enumerate the cores in that firmware.
- Register the devices those cores implement and install the drivers associated to them.
Identify the carrier board and initialize it.
Architecture of the FMC drivers

- Identify the carrier board and initialize it.
- Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.
Identify the carrier board and initialize it.

Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.
Identify the carrier board and initialize it.

Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.

- Identify the carrier board and initialize it.
- Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.
Architecture of the FMC drivers

- Identify the carrier board and initialize it.
- Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.
- Load the application firmware into the carrier FPGA.

Identify the carrier board and initialize it.
Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.
Load the application firmware into the carrier FPGA.
Identify the carrier board and initialize it.

Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.

Load the application firmware into the carrier FPGA.

Register a Wishbone bus with the kernel.
Architecture of the FMC drivers

- Identify the carrier board and initialize it.
- Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.
- Load the application firmware into the carrier FPGA.
- Register a Wishbone bus with the kernel.
Identify the carrier board and initialize it.

Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.

Load the application firmware into the carrier FPGA.

Register a Wishbone bus with the kernel.

Enumerate the cores in that firmware.
Identify the carrier board and initialize it.

Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.

Load the application firmware into the carrier FPGA.

Register a Wishbone bus with the kernel.

Enumerate the cores in that firmware.

Register the devices those cores implement and install the drivers associated to them.
- Identify the carrier board and initialize it.
- Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.
- Load the application firmware into the carrier FPGA.
- Register a Wishbone bus with the kernel.
- Enumerate the cores in that firmware.
- Register the devices those cores implement and install the drivers associated to them.
Identify the carrier board and initialize it.

Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.

Load the application firmware into the carrier FPGA.

Register a Wishbone bus with the kernel.

Enumerate the cores in that firmware.

Register the devices those cores implement and install the drivers associated to them.
Identify the carrier board and initialize it.

Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.

Load the application firmware into the carrier FPGA.

Register a Wishbone bus with the kernel.

Enumerate the cores in that firmware.

Register the devices those cores implement and install the drivers associated to them.
Linux Kernel I/O frameworks

In Linux staging area

Comedi

IIO

Drawbacks (for CERN applications)

interfaces are cumbersome (Comedi)

our use cases are far more complicated

Then

zio

comes

Alessandro Rubini and Federico Vaga, main developers

Designed ab initio for upstream integration

See under http://www.ohwr.org/projects/zio

Javier Serrano, David Cobas et al. (CERN)

Diverse equipment at CERN

FOSDEM 2012
Linux Kernel I/O frameworks

In Linux staging area

- Comedi
- IIO

Drawbacks (for CERN applications)

Interfaces are cumbersome (Comedi)

Our use cases are far more complicated

Then...\textit{zio} comes...

Alessandro Rubini and Federico Vaga, main developers

Designed \textit{ab initio} for upstream integration

See under [http://www.ohwr.org/projects/zio](http://www.ohwr.org/projects/zio)

Javier Serrano, David Cobas et al. (CERN)

Diverse equipment at CERN

FOSDEM 2012
Linux Kernel I/O frameworks

In Linux staging area
- Comedi
- IIO

Drawbacks (for CERN applications)
- interfaces are cumbersome (Comedi)
- our use cases are far more complicated
Linux Kernel I/O frameworks

In Linux staging area
- Comedi
- IIO

Drawbacks (for CERN applications)
- interfaces are cumbersome (Comedi)
- our use cases are far more complicated

Then zio comes
- Alessandro Rubini and Federico Vaga, main developers
- Designed ab initio for upstream integration
- See under http://www.ohwr.org/projects/zio
The zio framework is designed to flexibly support the following aspects:
The zio framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
The \texttt{zio} framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
The *ziō* framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
- Buffer management and timing for streaming conversion.
The zio framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
- Buffer management and timing for streaming conversion.
- Resolution,
The zio framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
- Buffer management and timing for streaming conversion.
- Resolution, sampling rate,
The zio framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
- Buffer management and timing for streaming conversion.
- Resolution, sampling rate, timestamping.
The zio framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
- Buffer management and timing for streaming conversion.
- Resolution, sampling rate, timestamping.
- Calibration,
The zio framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
- Buffer management and timing for streaming conversion.
- Resolution, sampling rate, timestamping.
- Calibration, offset
The **zio** framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
- Buffer management and timing for streaming conversion.
- Resolution, sampling rate, timestamping.
- Calibration, offset and gain.
The zio framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
- Buffer management and timing for streaming conversion.
- Resolution, sampling rate, timestamping.
- Calibration, offset and gain.
- Bit grouping in digital I/O.
The zio framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
- Buffer management and timing for streaming conversion.
- Resolution, sampling rate, timestamping.
- Calibration, offset and gain.
- Bit grouping in digital I/O.
- Triggering of acquisition/output.
The zio framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
- Buffer management and timing for streaming conversion.
- Resolution, sampling rate, timestamping.
- Calibration, offset and gain.
- Bit grouping in digital I/O.
- Triggering of acquisition/output.
- Support for DMA.
The \texttt{zio} framework is designed to flexibly support the following aspects:

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
- Buffer management and timing for streaming conversion.
- Resolution, sampling rate, timestamping.
- Calibration, offset and gain.
- Bit grouping in digital I/O.
- Triggering of acquisition/output.
- Support for DMA.

\textit{Potential to integrate in the main tree.}
**ziō concepts**

- Devices victims of our device drivers, contain channels
- Channels: basic I/O units
- Channel sets: group channels to be triggered (acquire, output) simultaneously
- Buffers for input and output buffering, of course

**ziō abstractions**
**zio concepts**

**zio abstractions**

**devices** victims of our device drivers, contain channels
**ZIO concepts**

**ZIO abstractions**

- **devices**: victims of our device drivers, contain channels
- **channels**: basic I/O units
### zio concepts

### zio abstractions

| **devices** | victims of our device drivers, contain channels |
| **channels** | basic I/O units |
| **channel sets** | group channels to be triggered (acquire, output) simultaneously |
**ZIO concepts**

**ZIO abstractions**

- **devices**: victims of our device drivers, contain channels
- **channels**: basic I/O units
- **channel sets**: group channels to be triggered (acquire, output) simultaneously
- **buffers**: for input and output buffering, of course
**ZIO concepts**

**ZIO abstractions**

- **devices**: victims of our device drivers, contain channels
- **channels**: basic I/O units
- **channel sets**: group channels to be triggered (acquire, output) simultaneously
- **buffers**: for input and output buffering, of course
- **triggers**: cause acquisitions to occur
Next candidates for \( zio \) integration

And for mainstream integration as well...
Next candidates for \((zio)\) integration

And for mainstream integration as well...

CERN-developed OHWR FMC boards
Next candidates for (zio) integration

And for mainstream integration as well...

CERN-developed OHWR FMC boards

- FD (2012 Q1)
Next candidates for (zio) integration

And for mainstream integration as well...

CERN-developed OHWR FMC boards

- FD (2012 Q1)
- 100Msps ADC (2012 Q2)
Next candidates for \((zi\circ)\) integration

And for mainstream integration as well...

CERN-developed OHWR FMC boards

- FD (2012 Q1)
- 100Msps ADC (2012 Q2)
- TDC (2012 Q2)
Next candidates for \((zio)\) integration

And for mainstream integration as well...

### CERN-developed OHWR FMC boards

- **FD** (2012 Q1)
- **100Msp ADC** (2012 Q2)
- **TDC** (2012 Q2)
- \(\ldots\) and so forth.
Next candidates for (zio) integration

And for mainstream integration as well...

**CERN-developed OHWR FMC boards**
- FD (2012 Q1)
- 100Msp ADC (2012 Q2)
- TDC (2012 Q2)
- ... and so forth.

**CERN-developed drivers for good old beasts**
Next candidates for (\texttt{zio}) integration

And for mainstream integration as well...

### CERN-developed OHWR FMC boards

- FD (2012 Q1)
- 100Msps ADC (2012 Q2)
- TDC (2012 Q2)
- ...and so forth.

### CERN-developed drivers for good old beasts

- Struck SIS33xx ADCs
Next candidates for \((\text{zio})\) integration

And for mainstream integration as well...

CERN-developed OHWR FMC boards
- FD (2012 Q1)
- 100Msps ADC (2012 Q2)
- TDC (2012 Q2)
- ... and so forth.

CERN-developed drivers for good old beasts
- Struck SIS33xx ADCs
- Tews TPCI200/TVME200 carries plus IPOCTAL serial boards
Next candidates for \((\text{zio})\) integration

And for mainstream integration as well...

**CERN-developed OHWR FMC boards**
- FD (2012 Q1)
- 100Msps ADC (2012 Q2)
- TDC (2012 Q2)
- \ldots and so forth.

**CERN-developed drivers for good old beasts**
- Struck SIS33xx ADCs
- Tews TPCI200/TVME200 carries plus IPOCTAL serial boards
- timing receivers, White Rabbit, etc.
Let's summarize

Accelerator control systems require managing a complex and diverse zoo of hardware devices. This is a problem many people share. Open Hardware makes for a much more manageable design and production way. It is proven to work in practice, both for a big project like White Rabbit and for the myriad of heterogeneous devices linked to it. The software side cannot simply live without the FOSS model, especially the Linux kernel! Thanks for the device model and the I/O frameworks.
Let’s summarize

- Accelerator control systems require managing a complex and diverse zoo of hardware devices...
Let’s summarize

- Accelerator control systems require managing a complex and diverse zoo of hardware devices... a problem many people share
Let’s summarize

- Accelerator control systems require managing a complex and diverse zoo of hardware devices... a problem many people share.
- Open Hardware makes for a much more manageable design and production way.

Javier Serrano, David Cobas et al. (CERN)
Diverse equipment at CERN
Let’s summarize

- Accelerator control systems require managing a complex and diverse zoo of hardware devices... a problem many people share
- Open Hardware makes for a much more manageable design and production way
- It is proven to work in practice
Let’s summarize

- Accelerator control systems require managing a complex and diverse zoo of hardware devices... a problem many people share.
- Open Hardware makes for a much more manageable design and production way.
- It is proven to work in practice.
- Both for a big project like White Rabbit and for the myriad of heterogeneous devices linked to it.
Let’s summarize

- Accelerator control systems require managing a complex and diverse zoo of hardware devices... a problem many people share.
- Open Hardware makes for a much more manageable design and production way.
- It is proven to work in practice.
- Both for a big project like White Rabbit and for the myriad of heterogeneous devices linked to it.
- The software side cannot simply live without the FOSS model.
Let’s summarize

- Accelerator control systems require managing a complex and diverse zoo of hardware devices... a problem many people share
- Open Hardware makes for a much more manageable design and production way
- It is proven to work in practice
- Both for a big project like White Rabbit and for the myriad of heterogeneous devices linked to it
- The software side cannot simply live without the FOSS model
- Especially the Linux kernel! Thanks for the device model and the I/O frameworks
Having the values of \( t_1, t_2, t_3 \) and \( t_4 \), the slave can calculate the one-way link delay:

\[
\delta_{ms} = \frac{(t_4-t_1)-(t_3-t_2)}{2}
\]
Millisecond timing clients

Javier Serrano, David Cobas et al. (CERN)

Diverse equipment at CERN

FOSDEM 2012

62 / 71
Millisecond timing
Example: Network Time Protocol (NTP)

Used in general-purpose computers
- Works across the Internet.
- Each client (slave) gets synchronized to one or more servers.

Cannot do better than 1 ms
- Asymmetries in network, switches and routers.
- Non-determinism due to OS scheduler (time tags done in SW).
- Requires strong statistics artillery to average over many measurements.
Microsecond timing clients

Javier Serrano, David Cobas et al. (CERN)

Diverse equipment at CERN
Microsecond timing clients

Javier Serrano, David Cobas et al. (CERN)

Diverse equipment at CERN
Microsecond timing

Example: Precision Time Protocol (PTP, IEEE1588)

Acts on both of NTP’s shortcomings
- Time-tagging can be done in HW.
- Special PTP switches ensure no loss in precision.

Has a hard time doing better than $1\mu s$
- Typical nodes use a free-running oscillator.
- Frequency offset (and drift) compensation generates extra traffic.
Nanosecond and picosecond timing clients

SIMPLIFIED FIELD CONTROL SYSTEM

SET POINTS

ERROR CORRECTION AND CONTROL

RF MODULATOR

REFERENCE RF

RF DETECTOR

LLCC

CAVITY

Desired Field

Real Field

BEAM

Javier Serrano, David Cobas et al. (CERN)

Diverse equipment at CERN
Nanosecond and picosecond timing

Example: White Rabbit

[Diagram showing a system with GPS, System Timing Master, Sync-E switch, and Sync-E node, with connections for clock loopback, Ethernet link, and communication between nodes.]
Monitor phase of bounced-back clock continuously.

Phase-locked loop in the slave follows the phase changes measured by the master.
Nanosecond and picosecond timing

Another example: neutrino oscillation experiments
Nanosecond and picosecond timing

Another example: neutrino oscillation experiments

Diverse equipment at CERN

GPS satellites

excellent long-term stability, but noisy in the short term

short-term stable

\[ \Delta t < 10 \text{ ns} \]

CERN → Gran Sasso

neutrinos

732 km

Javier Serrano, David Cobas et al. (CERN)