

Recent DAQ Developments at DESY, FEA

Tomas Vanat, tomas.vanat@desy.de
SEI-Tagung-2022

Recent DAQ Developments at DESY, FEA

Outline

01 DTS100G

- Project overview
- Prototype commissioning
- Use case

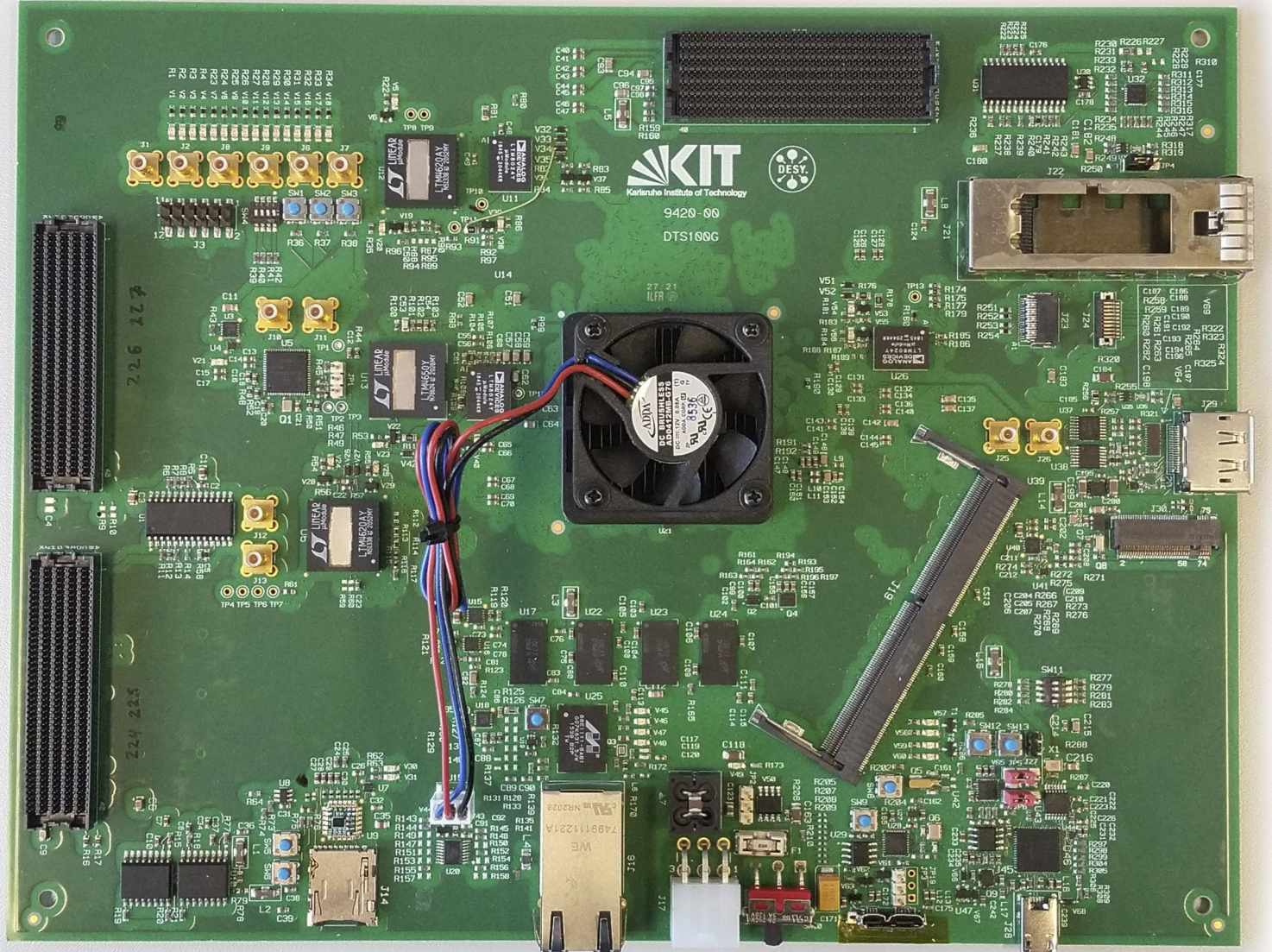
02 Caribou

- Overview of the framework and its capabilities
- Implementing a new detector

03 dSiPM

- Overview of the detector
- DAQ implementation using Caribou

DTS100G



DTS100G

Project overview

- A versatile DAQ board for high data throughput using up-to-date technologies
- A project within the HGF framework in collaboration with KIT (IPE group)
 - Design by DESY FEA (Manfred Zimmer, Artur Boebel, Igor Shevyakov, Tomas Vanat)
 - Production by KIT IPE (Matthias Balzer, Nick Karcher, Oliver Sander, Michael Schleicher and others)
- The first project with 28Gb/s (GTY) links at FEA
- PCB designed in Mentor Xpedition, utilizing parallel design by multiple users
- Based on Xilinx Zynq UltraScale+ MPSoC
- Capable of running embedded Linux (Yocto, Petalinux)

DTS100G

Available peripherals

PL (FPGA) peripherals

- 100Gb/s QSFP28 slot (4 GTY links)
- 100Gb/s FireFly connector (4 GTY links)
- FMC+ with 2×4 GTY and 4×4 GTH links
- 2× FMC HPC connectors with 2×4 GTH links
- 6× SMB GPIO connectors
- 8GB DDR4 RAM
- UART (over FTDI USB)

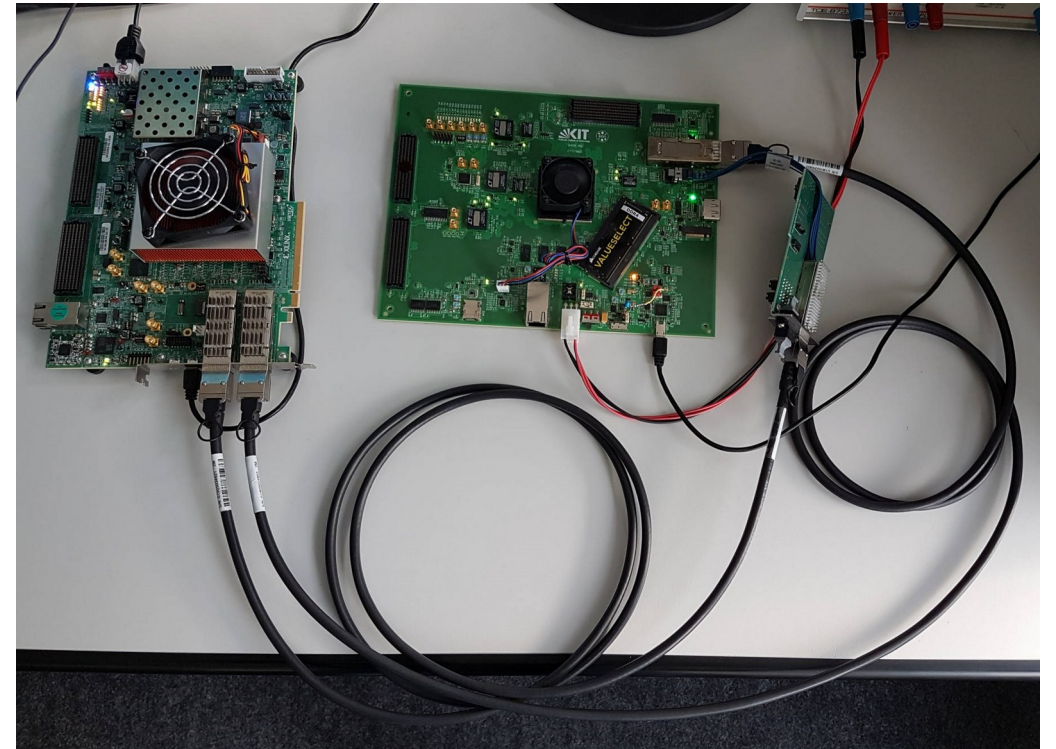
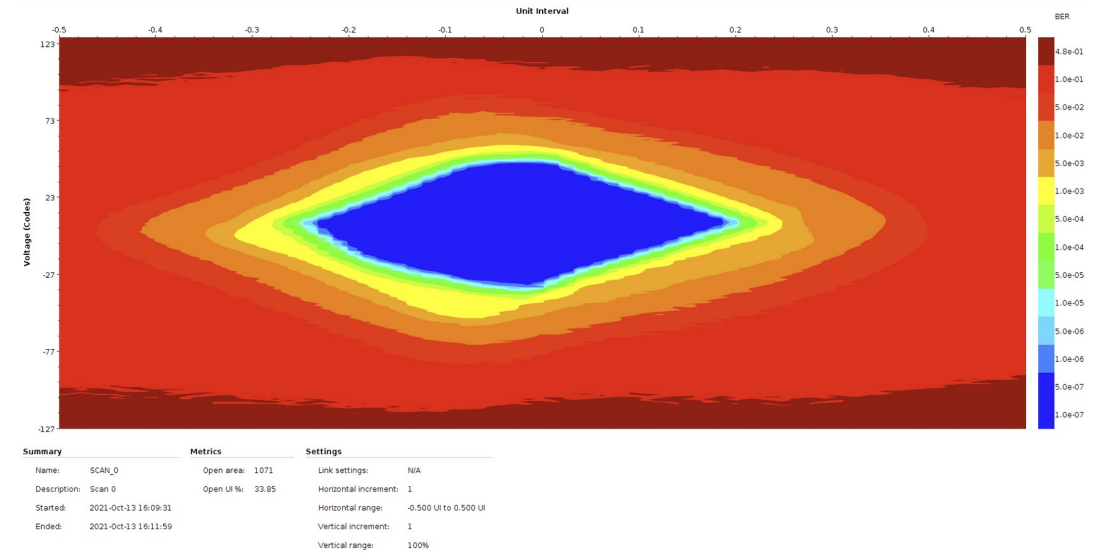
PS (ARM CPU) peripherals

- SoDIMM slot for up to 16GB DDR4 RAM module
- SD card (bootable)
- QSPI flash memory (bootable)
- M.2 SATA disk slot
- 1000BASE-T Ethernet interface
- USB 3.0 host
- DisplayPort
- 2× UART (over FTDI USB)

DTS100G

Commissioning and testing

- UDP packets over 100GBASE-CR4 Ethernet
 - Transfer between DTS100G and Xilinx VCU118
- Reed-Solomon Forward Error Correction (RS-FEC) enabled
- 2 full-duplex 100G lines
- 64 hours run time
- Large UDP packets (8-9 kB) with ASCII encoded text
- $>3 \times 10^{11}$ packets sent on each line
- Effective data rate was 99.3 Gb/s
- Zero errors detected



DTS100G

Use case

Electron Capture $^{163}\text{Holmium}$ (ECHO) experiment at KIT

- PL IP cores for signal digitization and processing
- Controlled from PS running Linux OS (Yocto)
- Interface to the experiment through ADCs and DACs on an FMC mezzanine card

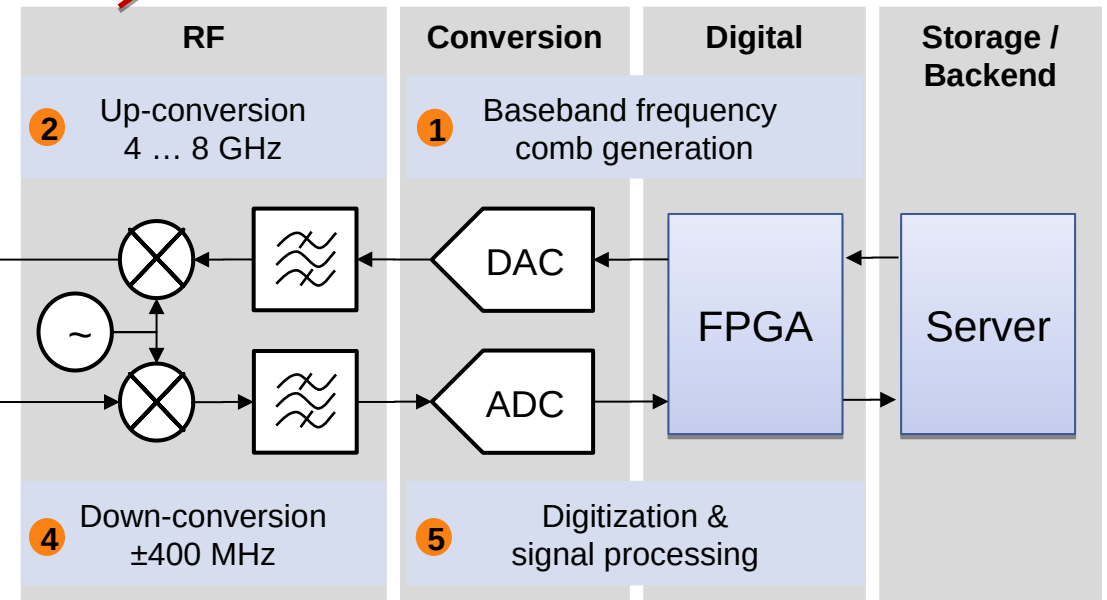
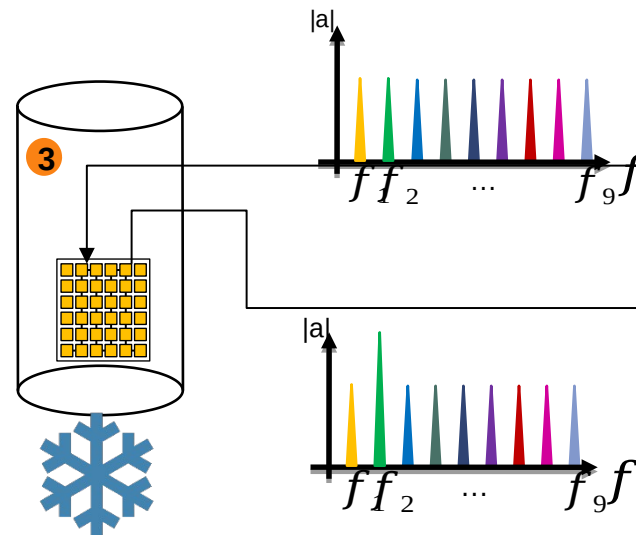
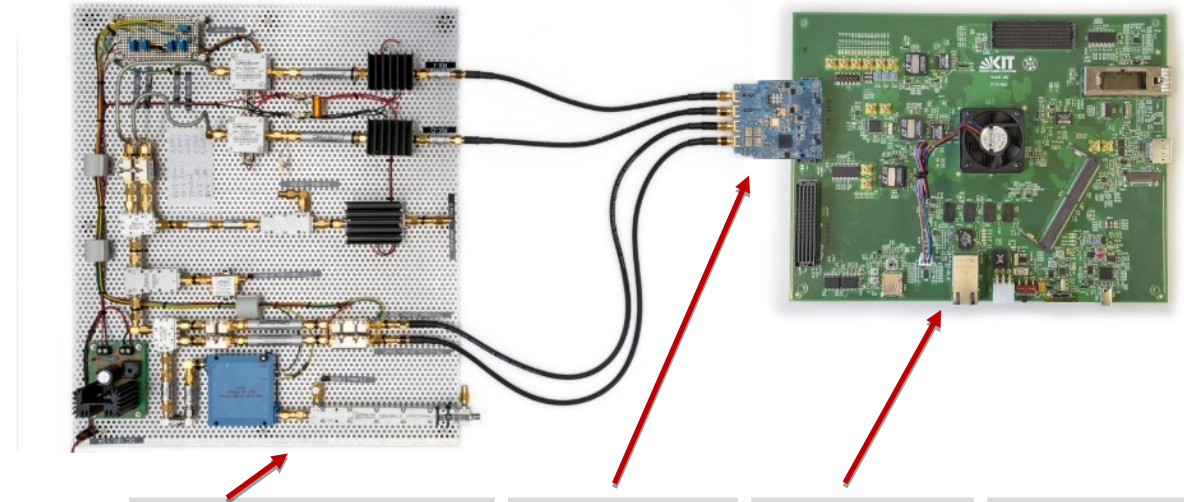
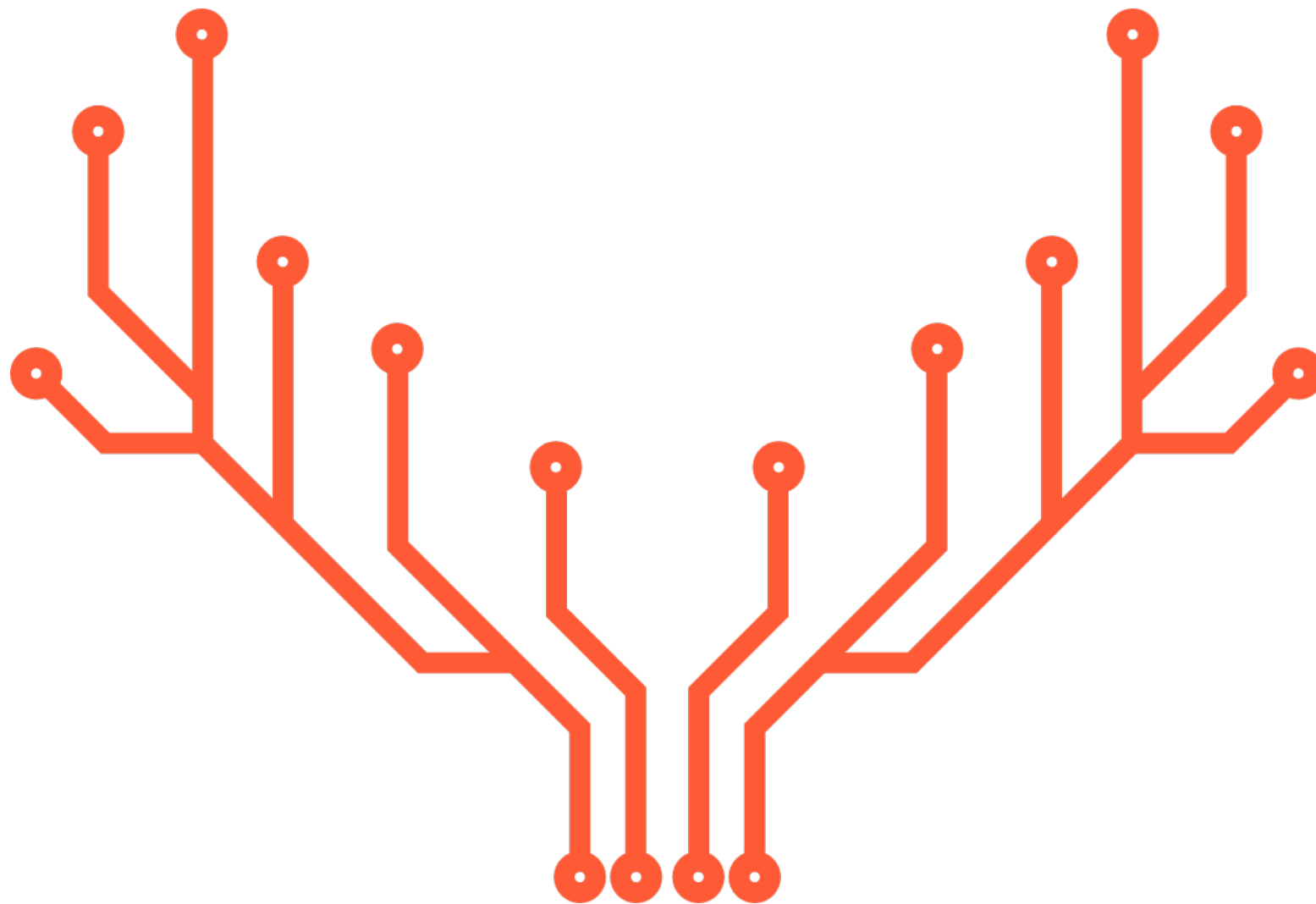


Image courtesy of Nick Karcher et al., KIT

Caribou



Caribou

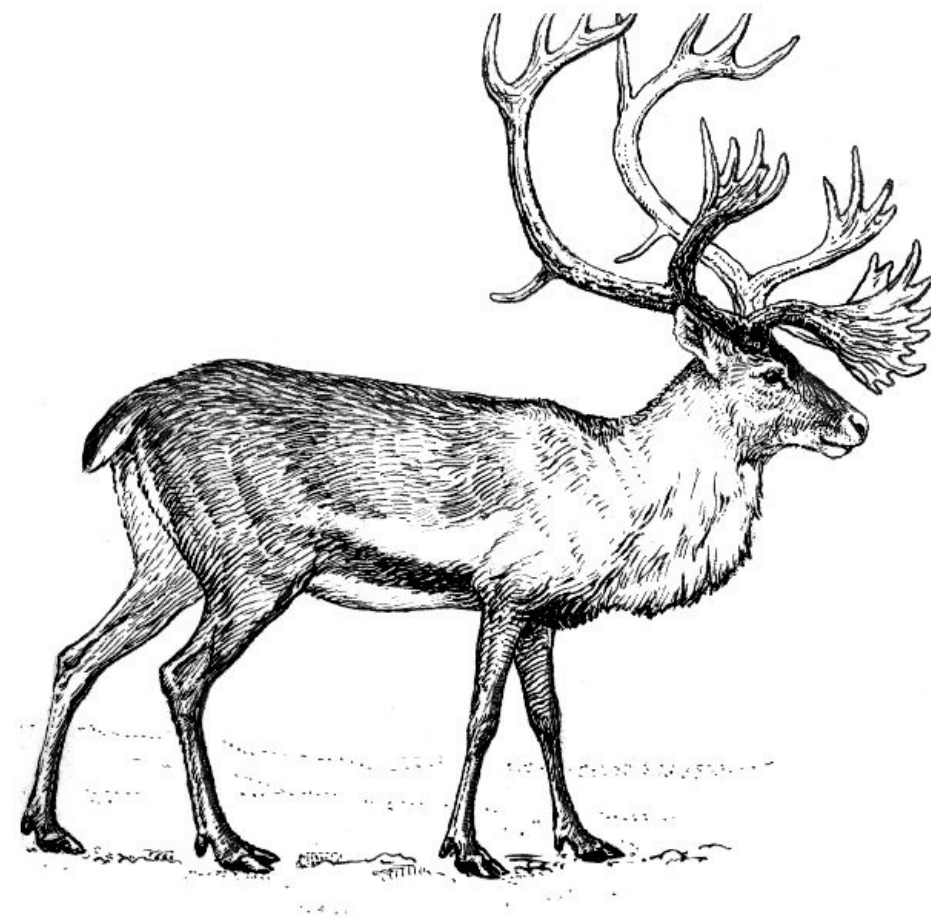
Motivation

- A similar concept of readout, control and power is used in most silicon pixel detectors
 - Differs in voltage levels, number of channels (data/voltage) or protocol
- A new detector-specific DAQ system is usually developed for each new detector (or an existing one is modified)
 - Time-consuming process of HW/FW/SW development
 - No innovative functionality
- A versatile DAQ system can speed-up development
 - Common HW and SW core and interface
 - For detector development and tests

Caribou

Project overview

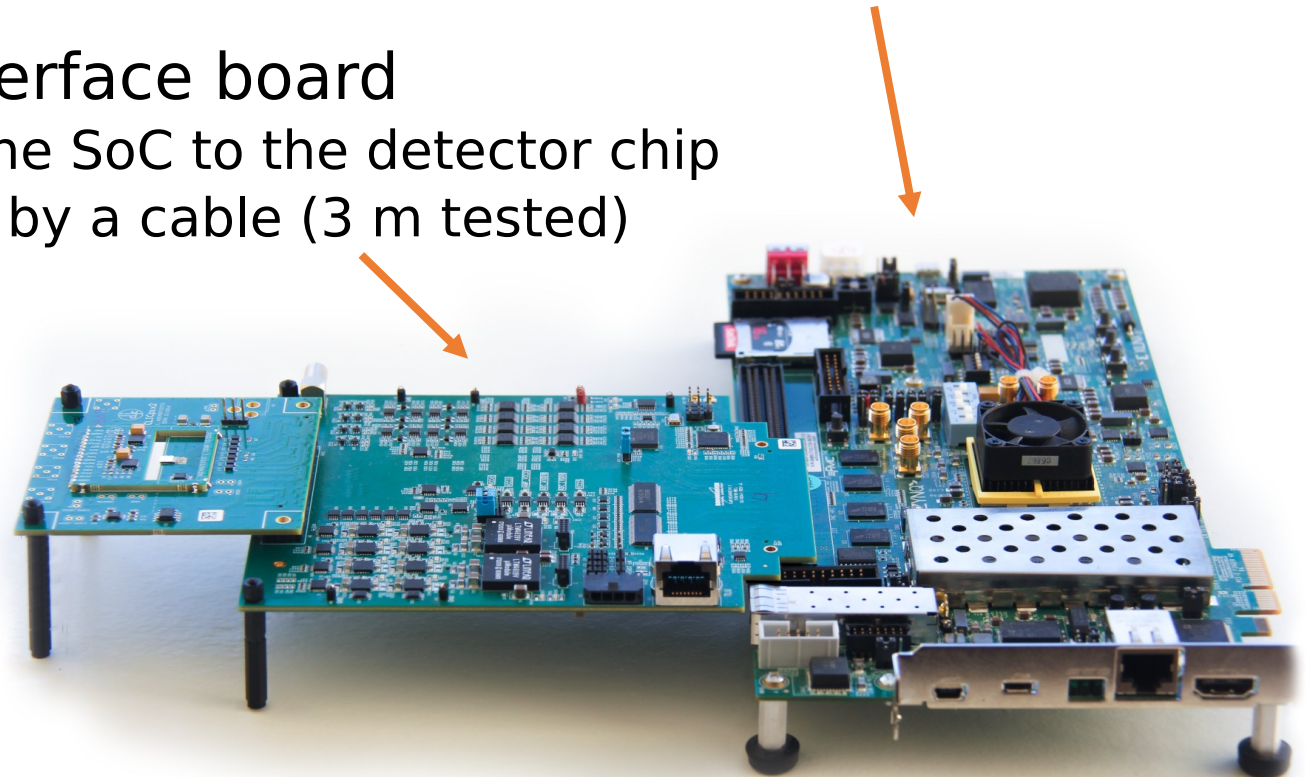
- Open source hardware, firmware and software for laboratory and high-rate beam tests of (silicone) pixel detectors
- Maintained by collective effort of developers from
 - Brookhaven National Lab
 - CERN
 - DESY
- Minimizes device integration effort
- Reduces time to get first data from a new detector



Caribou


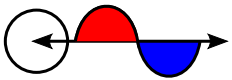
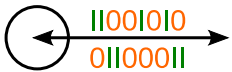


Hardware architecture

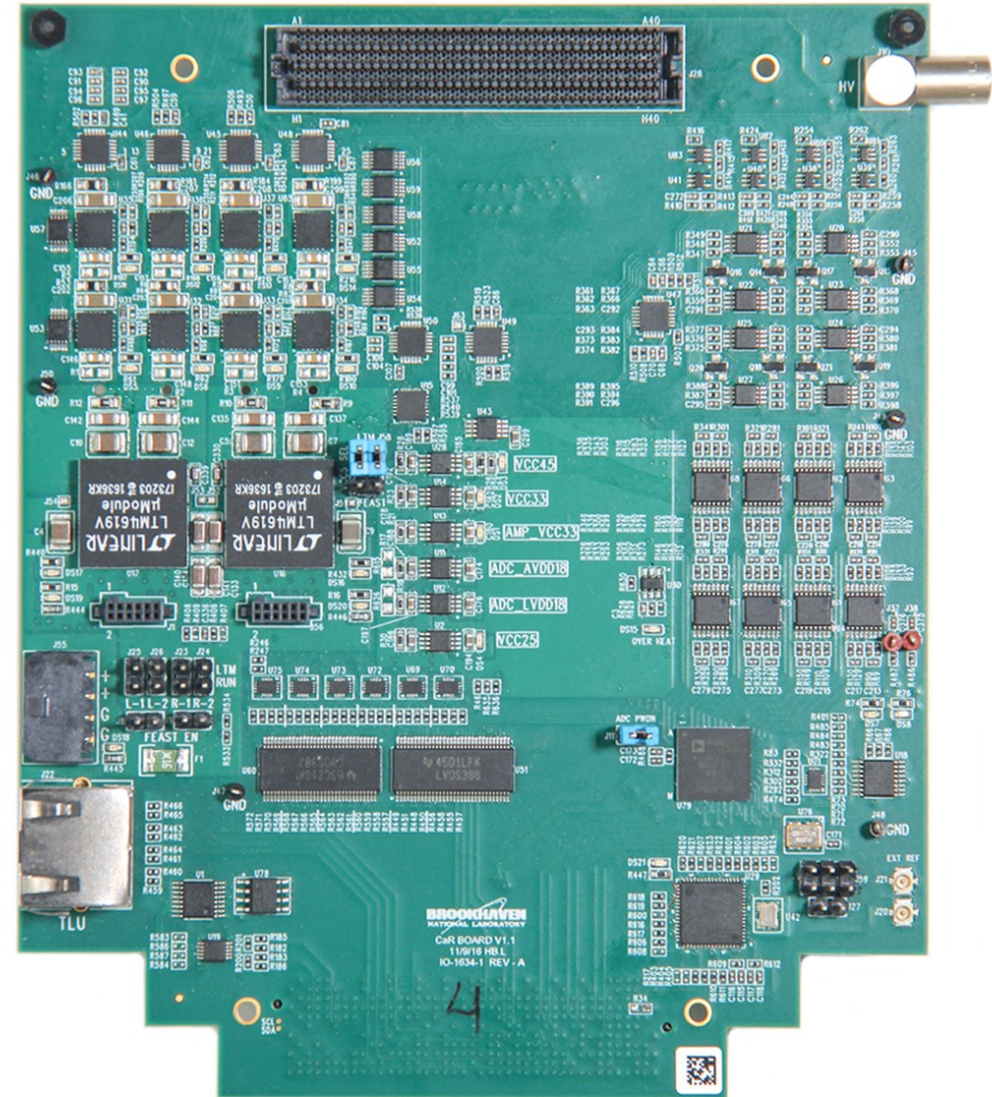
- SoC board (e.g. Xilinx ZC706)
 - An embedded CPU runs DAQ and control software
 - An FPGA runs custom hardware blocks for data processing and detector control
- Control and Readout (CaR) interface board
 - Provides physical interface from the SoC to the detector chip
 - CaR - SoC connection extendable by a cable (3 m tested)
- Application-specific detector carrier board
 - Detector chip and passive components only



Caribou

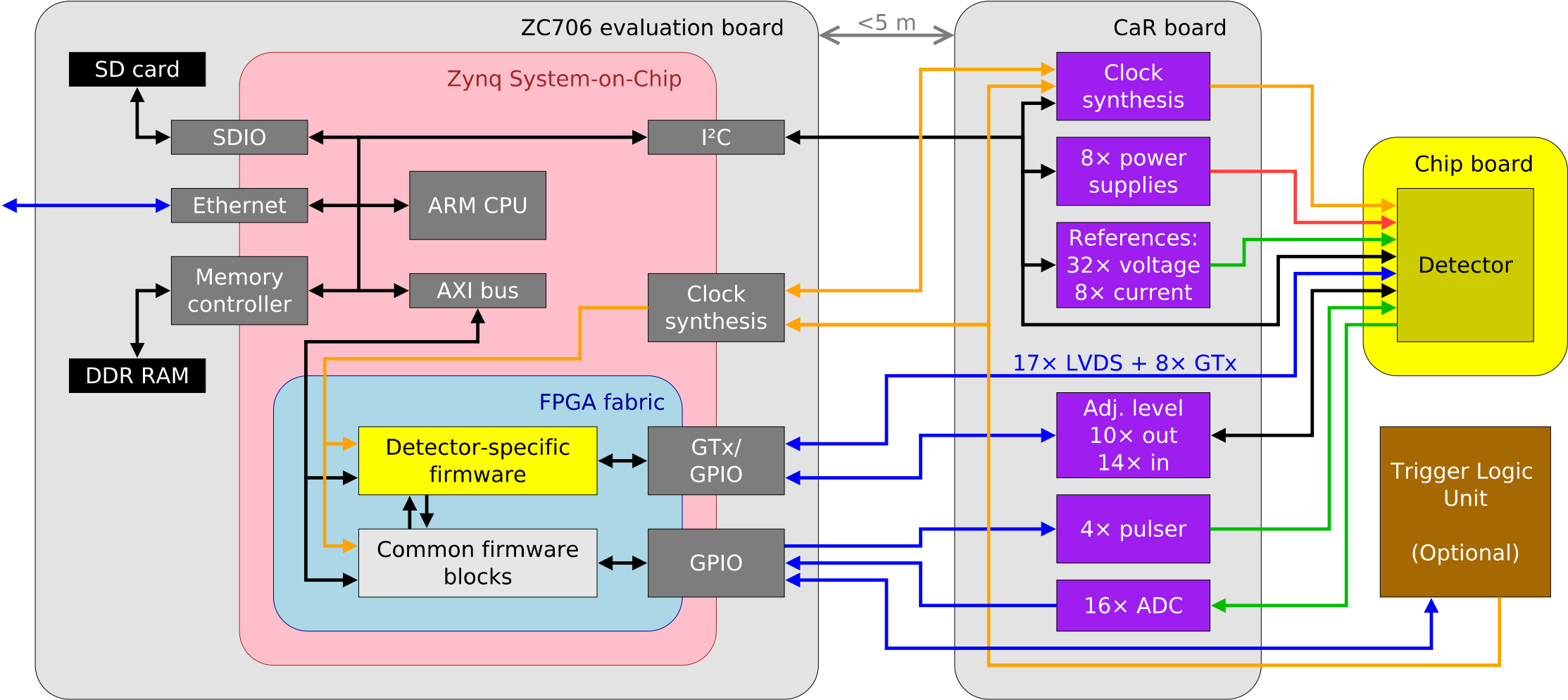
Control and Readout (CaR) board

 Power	<ul style="list-style-type: none">• 8 adjustable power supplies with monitoring• external HV input
 Analog I/O	<ul style="list-style-type: none">• 8 inputs to 12-bit ADC, 50 kSamples/s• 16 inputs to 14-bit ADC, 65 MSamples/s• 4 programmable injection pulsers• 32 adjustable voltage references• 8 adjustable current references
 Digital I/O	<ul style="list-style-type: none">• 8 full-duplex high-speed links (0.8-12 Gb/s)• 17 bidirectional LVDS links (<1.1 Gb/s)• 10/14 output/input links, adjustable level• TLU interface
 Clocking	<ul style="list-style-type: none">• Programmable low-jitter clock generator with external (TLU) reference
 Interface	<ul style="list-style-type: none">• FMC interface to FPGA board• 320-pin SEARAY interface to detector chip



Caribou

Hardware architecture schematic



Caribou

Operating system

Custom Yocto-based Linux distribution

- OpenEmbedded build system
- Using Yocto reference embedded distribution (called Poky)
 - Adding some standard Linux packages (ssh, python, git, NTP etc.)
 - Using community-developed layer for Xilinx Devices
 - Adding custom layers with own software and recipes to build an SD card image (meta-caribou)
- OS boots from SD card
 - One boot partition with FPGA bitstream, boot configuration and Linux kernel
 - Second partition with Linux root filesystem



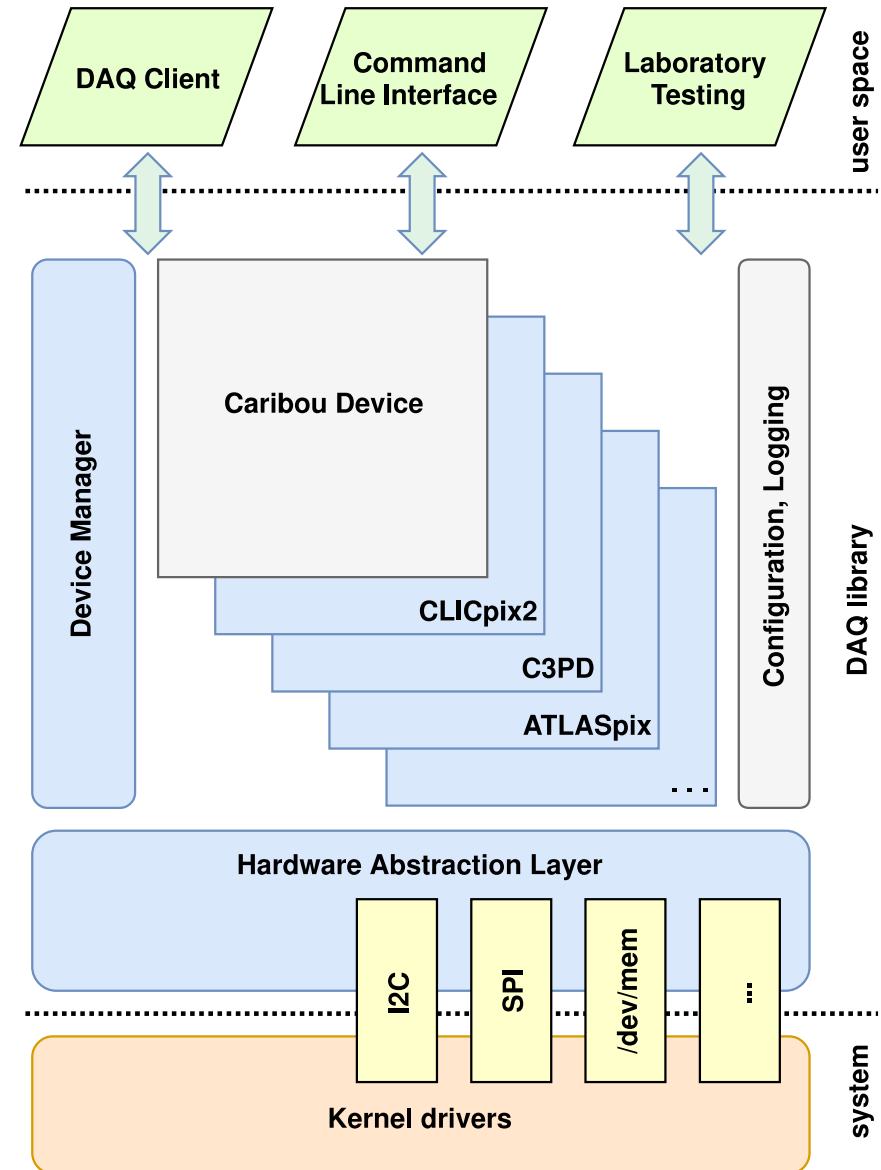
```
pclcd-lab-zynq.dyndns.cern.ch - PuTTY
Using username "root".
root@PCLCD-LAB-ZYNQ's password:
Last login: Fri Jan 11 12:18:03 2019 from 128.141.234.81
root@caribou:~# pearycli
|12:19:07.683| (WARNING) No configuration file provided, all devices will use defaults!
|12:19:07.683| (INFO) Welcome to pearyCLI.
|12:19:07.684| (INFO) Currently 0 devices configured.
|12:19:07.684| (INFO) To add new devices use the "add_device" command.
# add_device CLICpix2
|12:19:15.150| (INFO) Creating new instance of device "CLICpix2".
|12:19:15.151| (STATUS) New Caribou device instance, version peary v0.9+284~g6a04348
|12:19:15.151| (STATUS) This device is managed through the device manager.
|12:19:15.151| (STATUS) Firmware version: 00000000 (0/0/2000 0:0:0)
|12:19:15.154| (INFO) Appending instance to device list, device ID 0
|12:19:15.154| (INFO) Manager returned device ID 0.
# powerOn 0
|12:19:23.289| (INFO) CLICpix2Device: Powering up CLICpix2
# init 0
|12:19:26.625| (INFO) Configuring CLICpix2Device
|12:19:30.670| (ERROR) Cannot lock to external clock.
# exit
|12:19:33.769| (INFO) Done. And thanks for all the fish.
root@caribou:~#
```

Caribou

Peary software

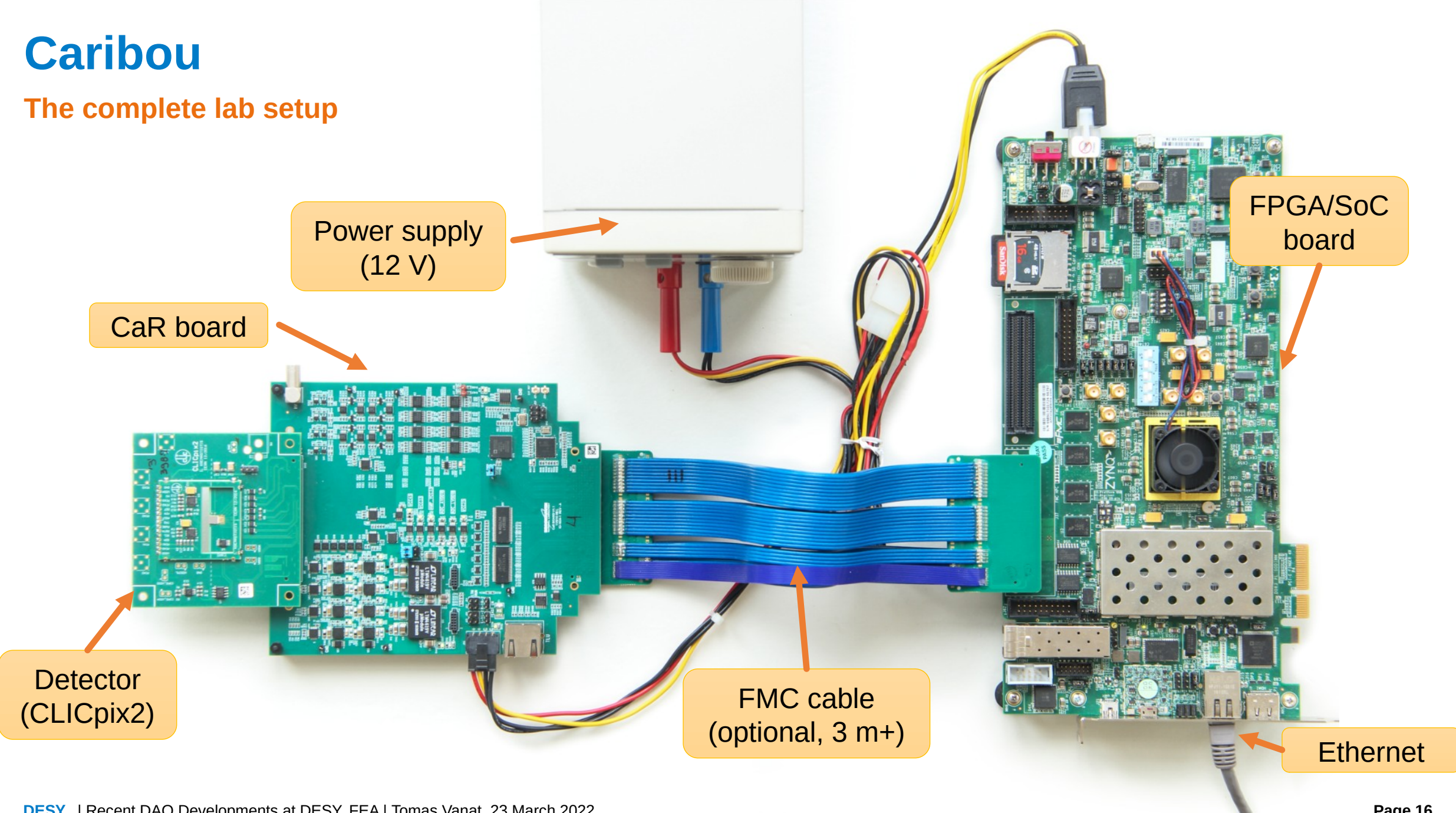
DAQ software framework for Caribou (Peary):

- Hardware Abstraction Layer (HAL)
 - Interface between SW and HW
 - Allows handling peripherals as objects in C++
- Functions to control CaR board
 - Set/measure voltages, capture ADC, ...
- Device management
 - Multiple devices/detectors in parallel
- Command line interface (CLI)
- Client interface for integration with a superior DAQ
- Compatible with 32-bit and 64-bit systems



Caribou

The complete lab setup

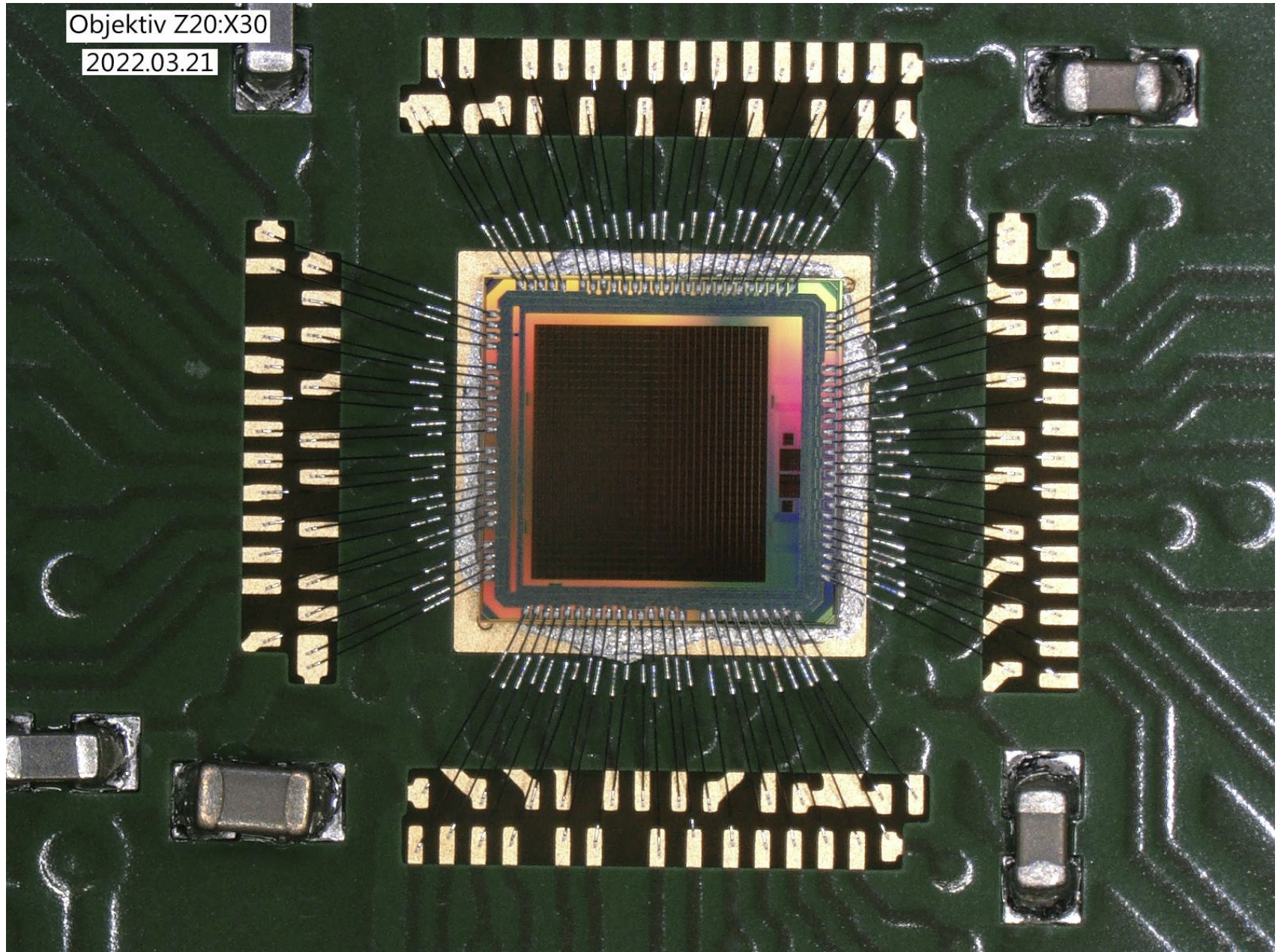


Caribou

How to implement a new detector

- Hardware:
 - Design and produce a chipboard – route chip signals to suitable pins on CaR connector
- FPGA firmware:
 - Create or modify FW blocks:
 - to handle specific control signals of the detector
 - to receive detector data and push it to FIFO
- Software (Peary):
 - Define mapping of:
 - User-friendly detector-specific names to default names of CaR board peripherals (e.g. `Vthreshold`→`BIAS_3`)
 - Names of detector-specific registers in the detector and FPGA to their physical addresses
 - Define detector-specific functions (`configure`, `startDAQ`, ...)

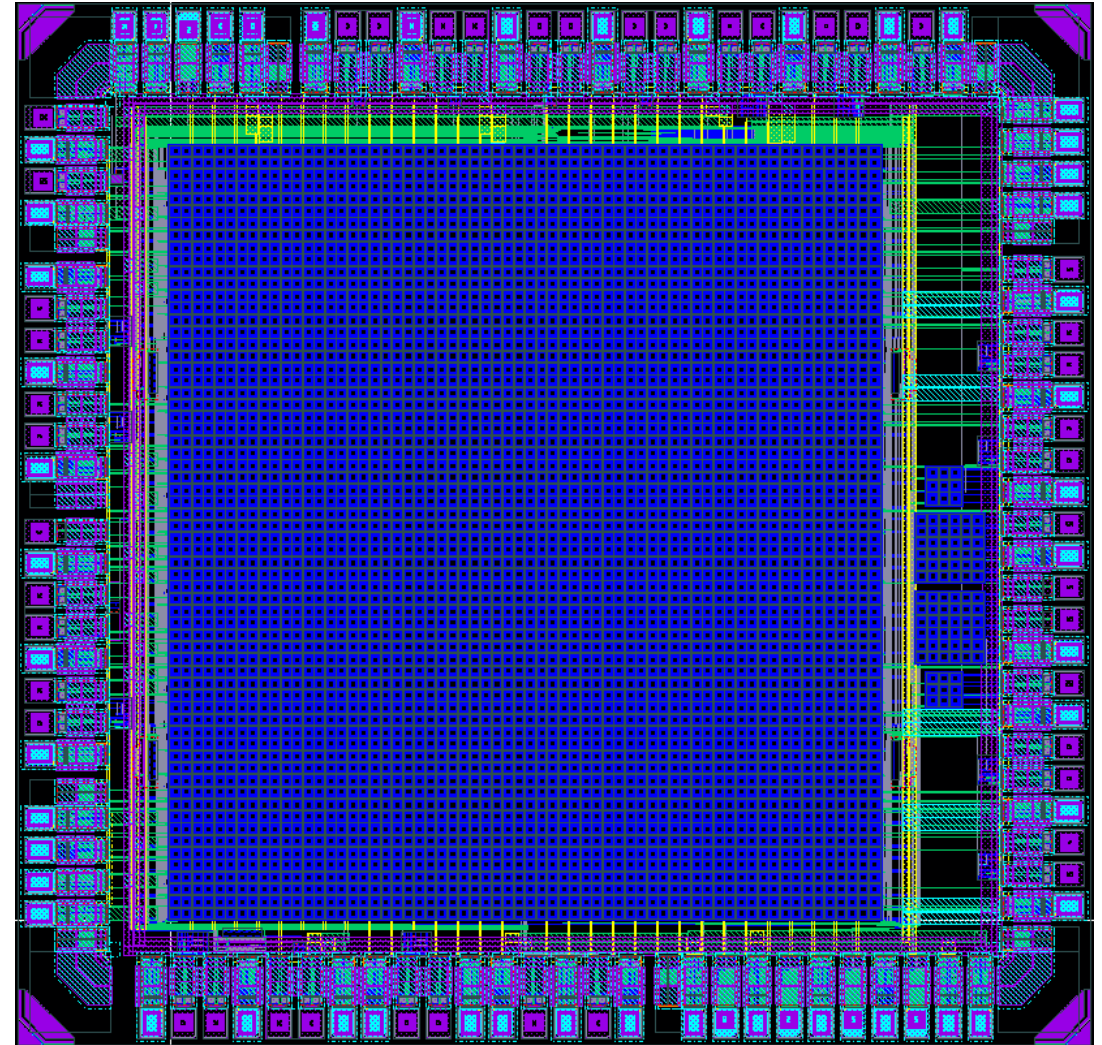
dSiPM



dSiPM

Project Overview

- A Digital (d) Silicon (Si) Photomultiplier (PM)
- Designed at FEC – Microelectronics group
 - Designer: Inge Diehl
- One pixel = 4 SPADs (Single-photon avalanche diode)
- A hit information can be read out from each pixel individually
- 32×32 pixels
- Digital readout interface, including timing information
 - 4×716 Mb/s data + 716 Mb/s timing
- First samples arrived in mid-February 2022



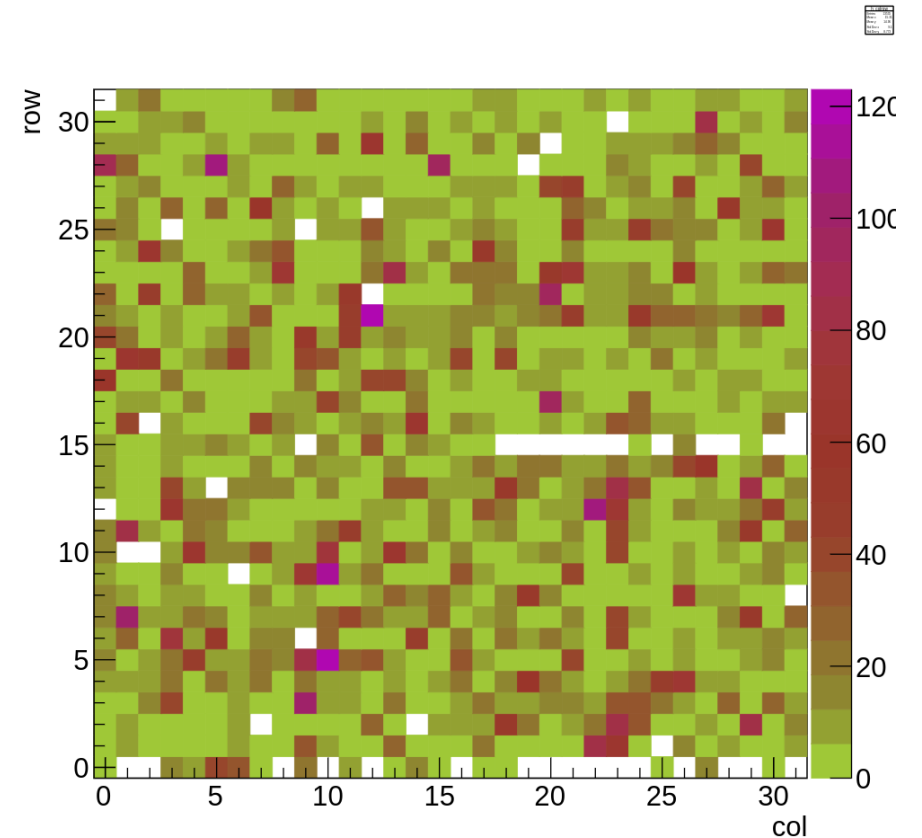
ca. 3400 x 3300 μm^2

Image courtesy of Inge Diehl

dSiPM

DAQ implementation and first tests

- Readout using Caribou:
 - Design a Caribou-compatible chipboard (~1-2 weeks)
 - Implement FPGA code to
 - Load configuration to the chip (~3 days)
 - Provide chip-specific signals during DAQ (~2 days)
 - Receive, align and store the chip data (~3 days)
 - Map CaR peripherals to the chip IOs (~0.5 day)
 - Create the basic functions to control the chip (~3-4 days)
 - Debug and improve :-)
- New DAQ functions are still being implemented, as we are getting familiar with the detector behavior



Summary

Recent DAQ Developments at DESY, FEA

- We are able to design up-to-date hardware with high data throughput
- Successfully designed a fast DAQ card in cooperation with KIT
 - Card prototype produced and tested
 - A few minor issues discovered on the prototype will be fixed in the final version
- Using Caribou for fast DAQ development
 - Ideal for prototype detector testing and characterization
 - We are able to make the detector working within a few weeks