

Signale, Neutronendetektor und RFSoc

Signale aus einem Neutronendetektor mit einem RFSoc verarbeitet

Jörn Plewka
Christian Jacobsen

Technikum – Helmholtz-Zentrum
hereon

Darmstadt, 19.03.2024

Agenda

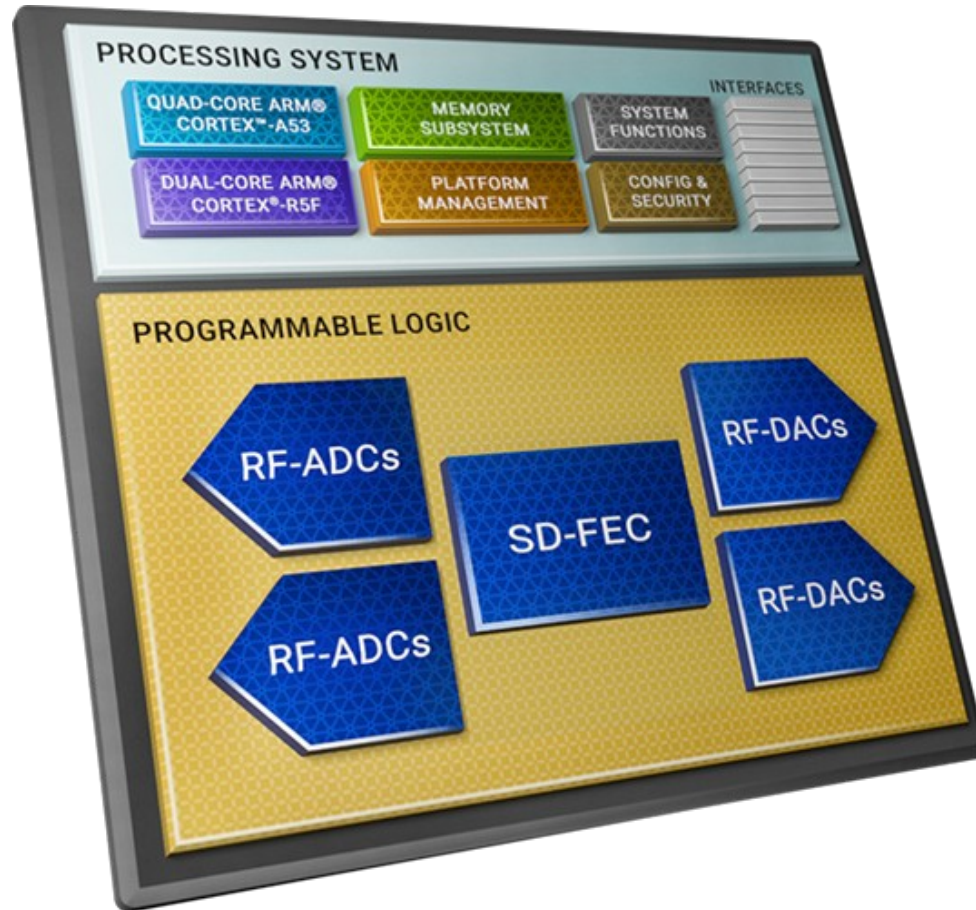
- **Vorstellung AMD RFSoc**
- **Rückblick Detektor**
- **Umstellung auf RFSoc**
- **Python / Jupyter**

RFSoc

Vorstellung AMD RFSoc



AMD RFSoc

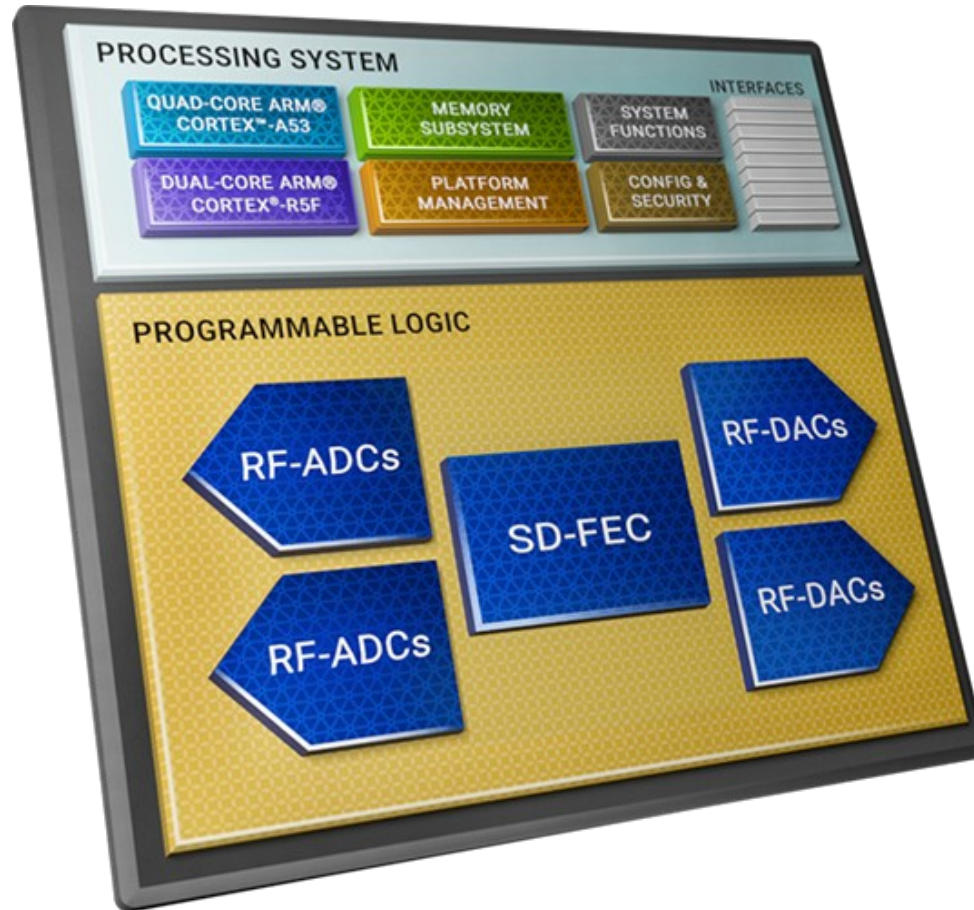


- Zynq Ultrascale+ Architektur
 - ADC / DAC integriert – kein JESD204 Overhead
 - Vielzahl von Kanal- und Geschwindigkeitskombination (3 Generationen)
 - **8x 4GSPS ADC / 8x 6GSPS DAC**
 - **8x 5GSPS ADC / 16x 9GSPS DAC**
 - Quad-core Arm® Cortex®-A53 MPCore™ 1.3GHz
 - Dual-core Arm Cortex-R5F MPCore 533MHz
-
- Selection Guide:
<https://www.xilinx.com/content/dam/xilinx/support/documents/selection-guides/zynq-usp-rfsoc-product-selection-guide.pdf>

Bild: <https://www.xilinx.com/products/silicon-devices/soc/rfsoc.html>

AMD RFSoc

Beispiel Anwendungen (AMD)



- 5G and LTE Wireless
- Remote-PHY for Cable Access DOCSIS 3.1
- Phased Array Radar/Digital Array RADAR
- Test & Measurement
- Satellite Communications

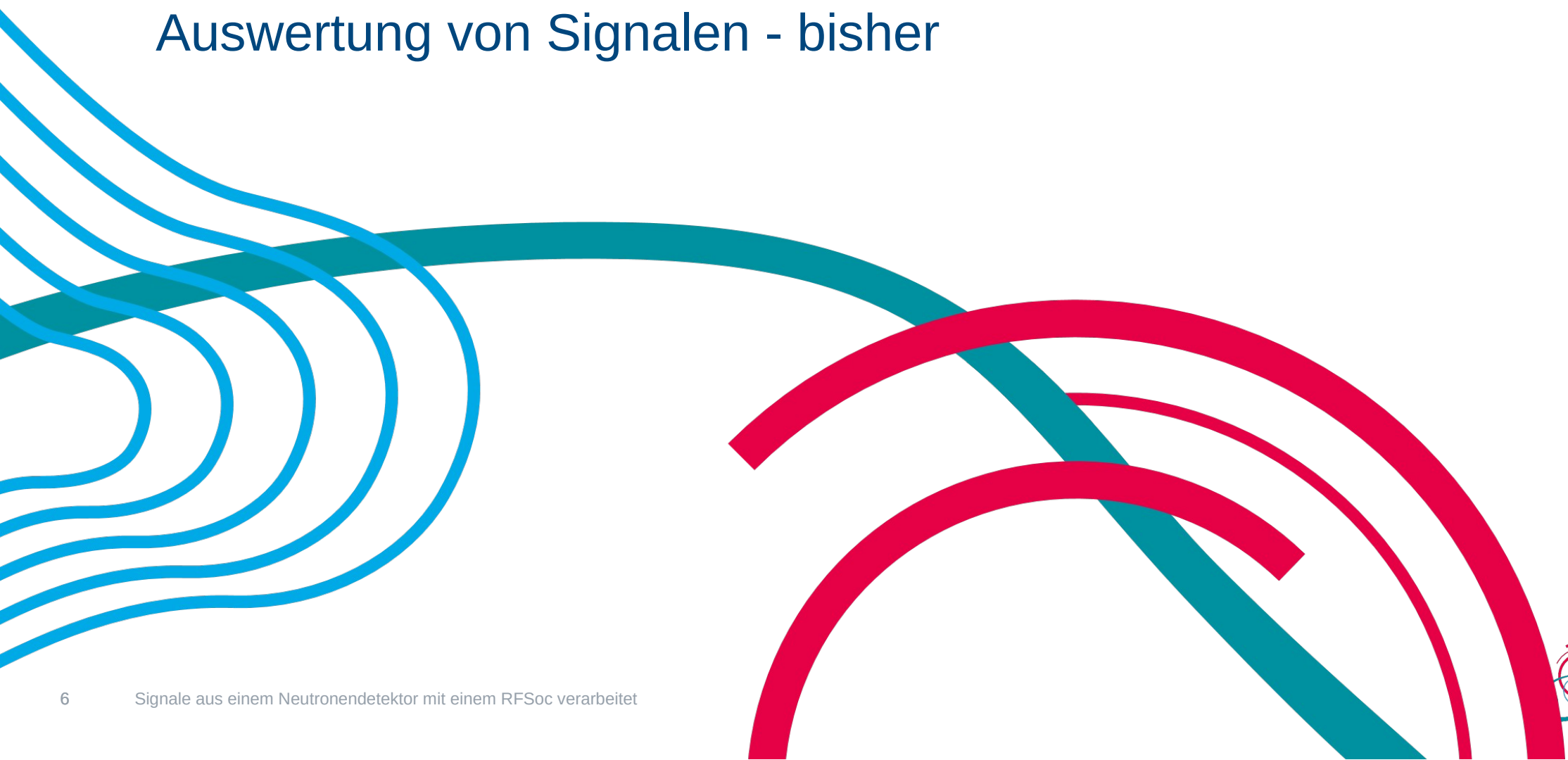
Quelle:

<https://www.xilinx.com/products/silicon-devices/soc/rfsoc.html#applications>

Bild: <https://www.xilinx.com/products/silicon-devices/soc/rfsoc.html>

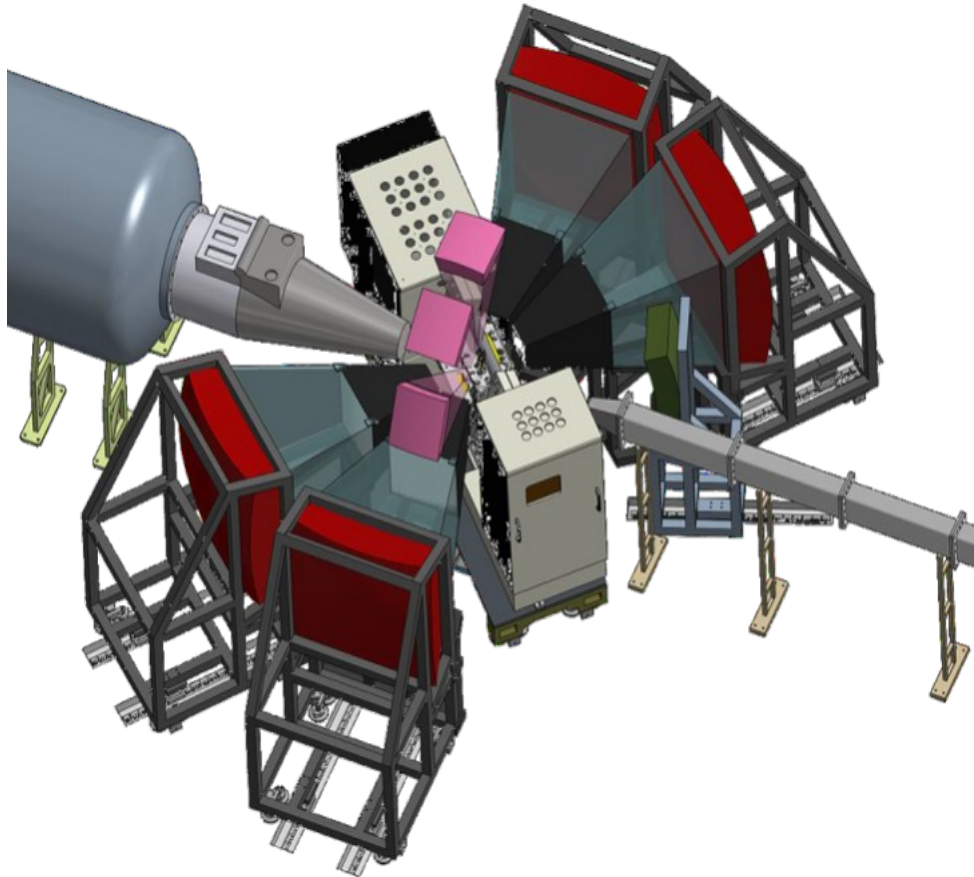
Detektor

Auswertung von Signalen - bisher



BEER @ESS

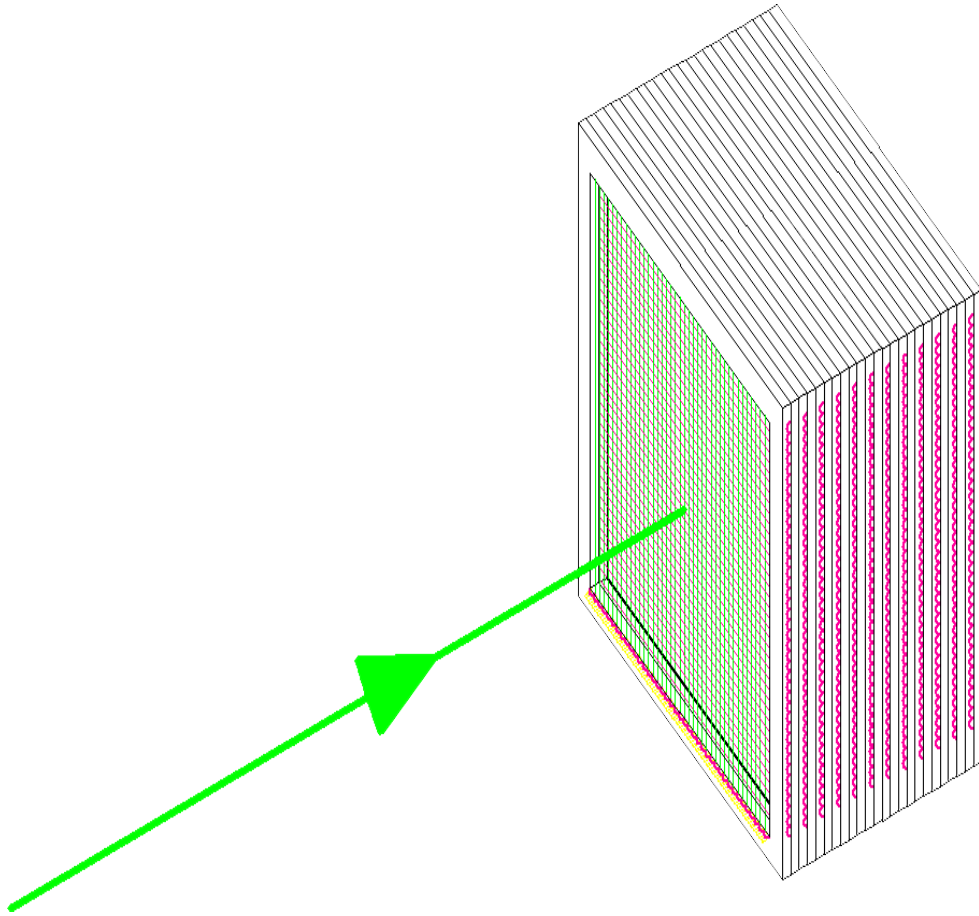
Modell Vollausbau



- BEER = Beamline for European Materials Engineering Research
- Vollausbau:
 - 4x 1m² Detektor
 - SANS Rohr mit Detektor
 - ARC-Detektoren
 - kl. Detektor neben Strahlrohr
 - Beammonitore

1m² Detektor

Aufbau

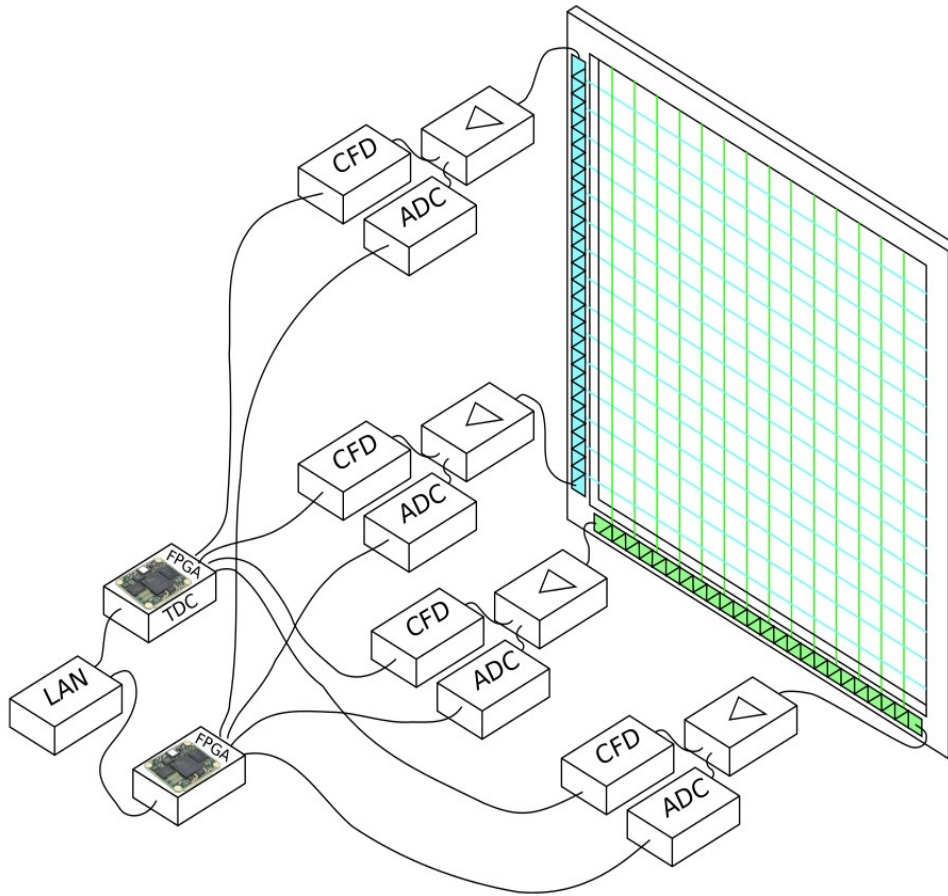


Grundkonstruktion:
Multi-Wire Proportional Chamber (MWPC)

- Ein Detektor besteht aus 12 Ebenen
- Effizienz einer Ebene wäre zu niedrig
- Zählrate global 1Mcps
 - Zählrate pro Ebene ~100kcps

1m² Detektor

Auswertung

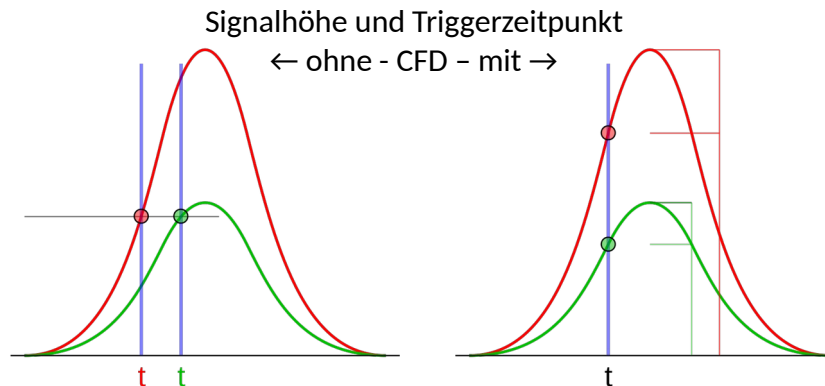


- Delayline: Zeitcodierung des Ortes
- X- und Y-Delayline für jeweilige Drahtebene

- 4x Vorverstärker
- 4x CFD (Constant Fraction Discriminator, s.u.)
- 4x TDC Kanal

16 TDC-Kanäle pro Zynq FPGA realisiert
Statt ADC nur Threshold und weiterer TDC-Kanal

- → 2 Ebenen pro Zynq FPGA



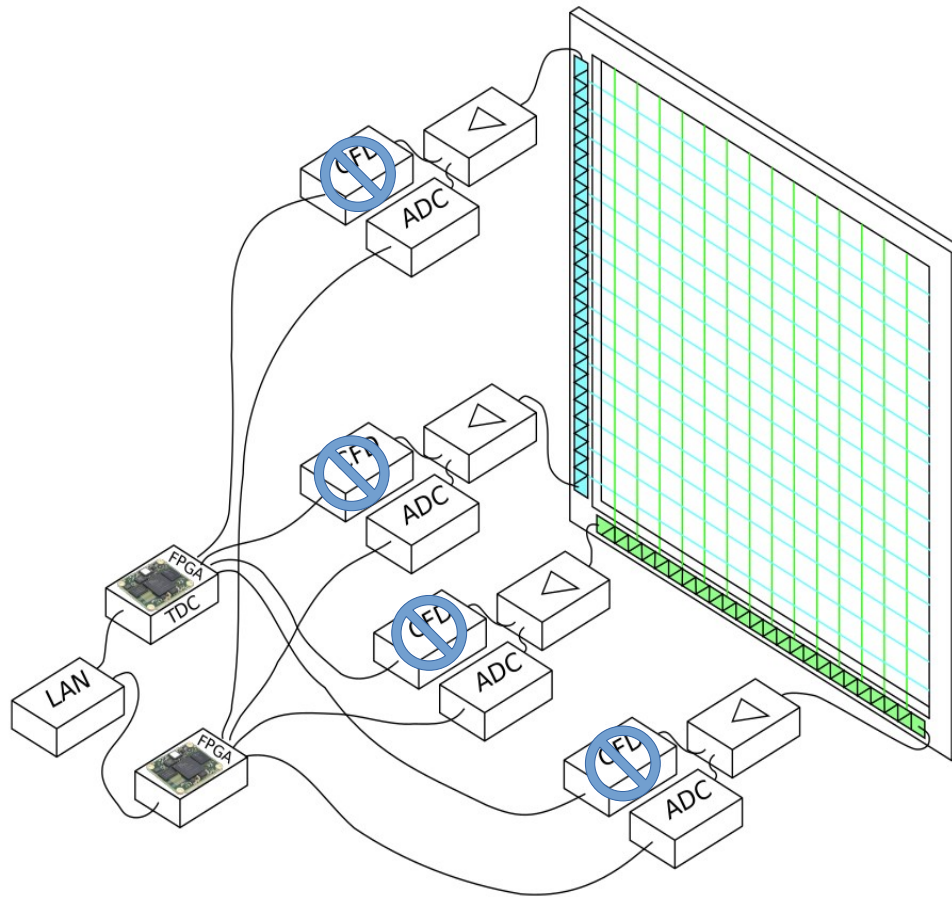
Detektor

Auswertung von Signalen – neue Entwicklung



1m² Detektor

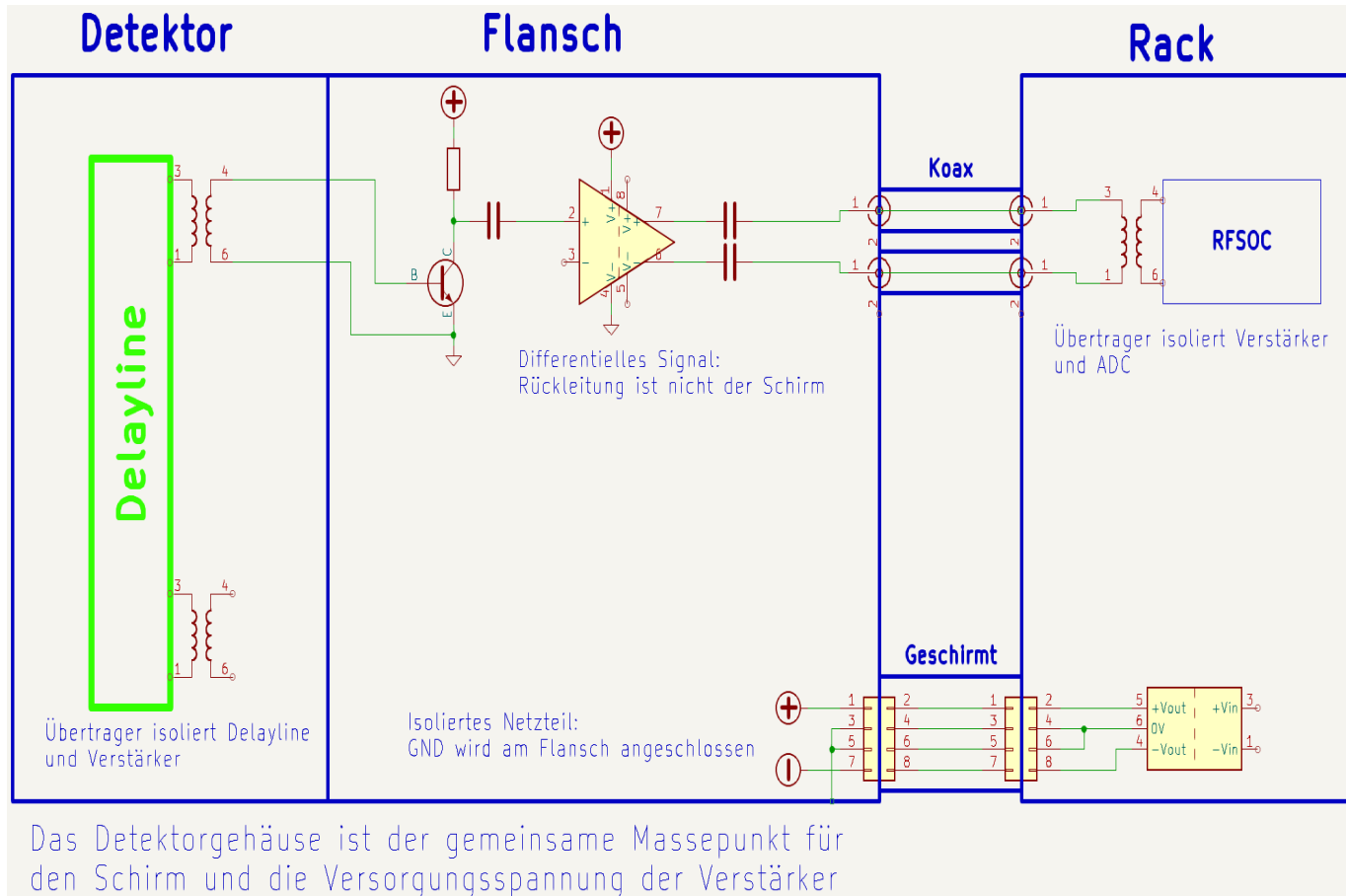
Auswertung RFSoc



- Delayline: Zeitcodierung des Ortes
- X- und Y-Delayline für jeweilige Drahtebene
- 4x Vorverstärker
- 4x ADC Kanäle
- RFSoc FPGA mit 8 ADC-Kanälen
- CFD jetzt in digitaler Logik
- → wieder 2 Ebenen pro RFSoc

1m² Detektor

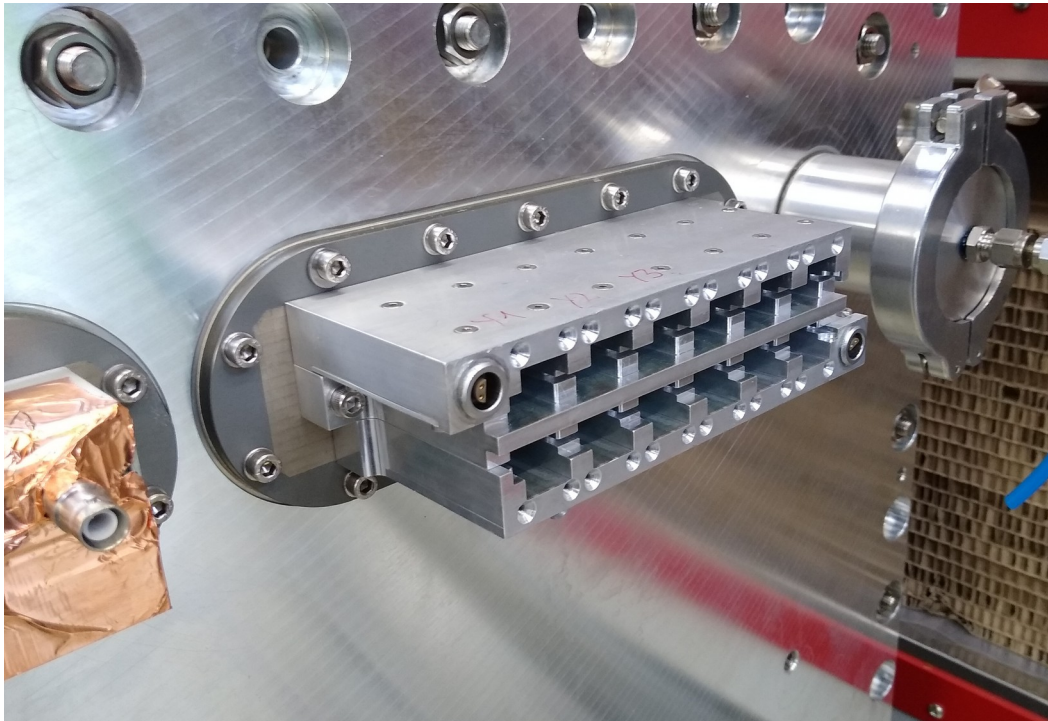
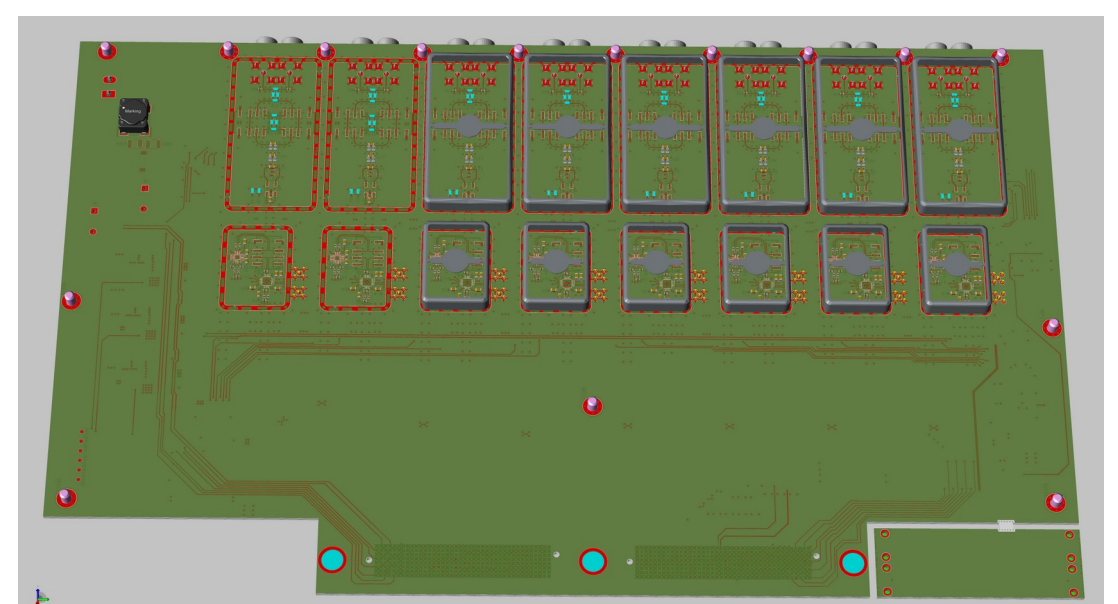
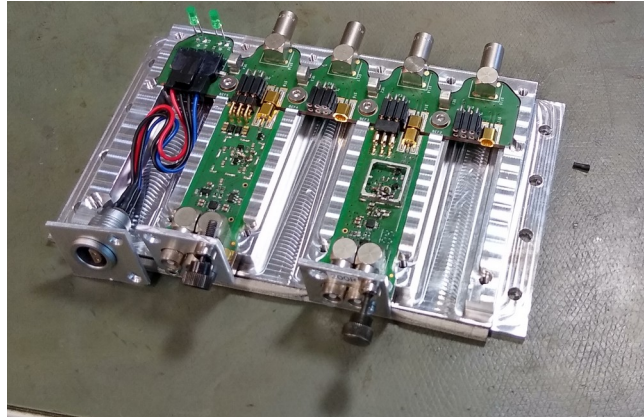
Signalweg und Massekonzept



Ältere Beamlines haben oft große Probleme mit Masseströmen, daher:

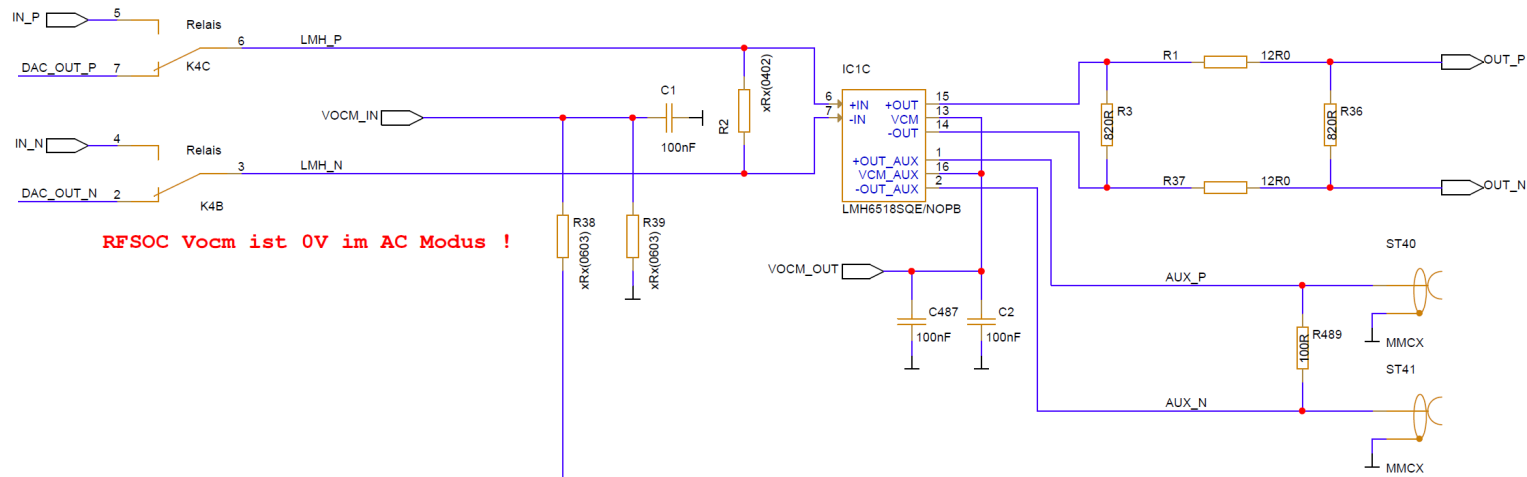
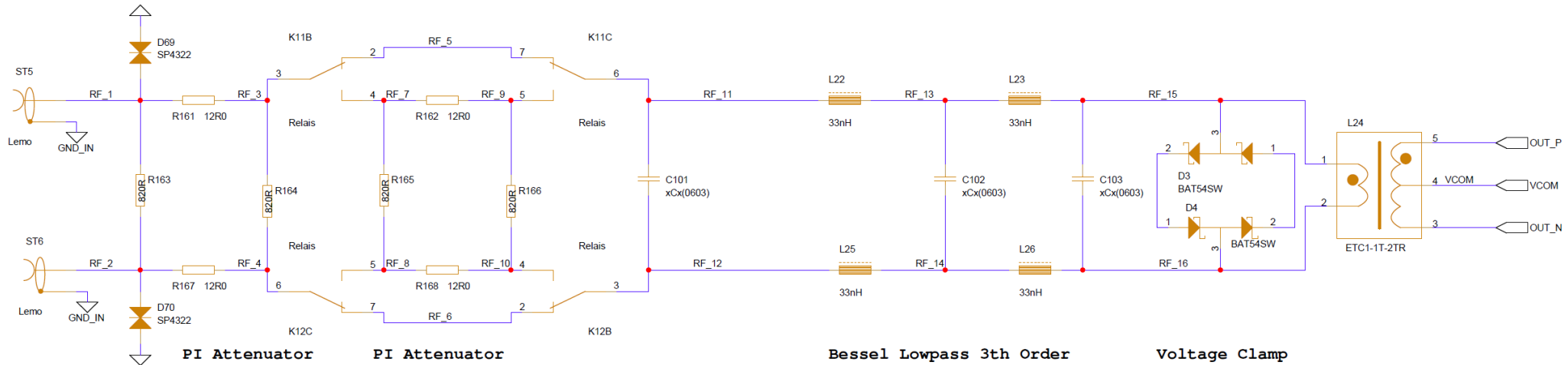
- Übertrager auf beiden Seiten
- Signale werden möglichst früh differentiell
- Flansch mit Steckplätzen für Vorverstärker ist Teil des Detektorgehäuses
- Signal- und Atmosphärenübergang per Leiterplatte im geschirmten Flansch

1m² Detektor Signalaufbereitung



1m² Detektor

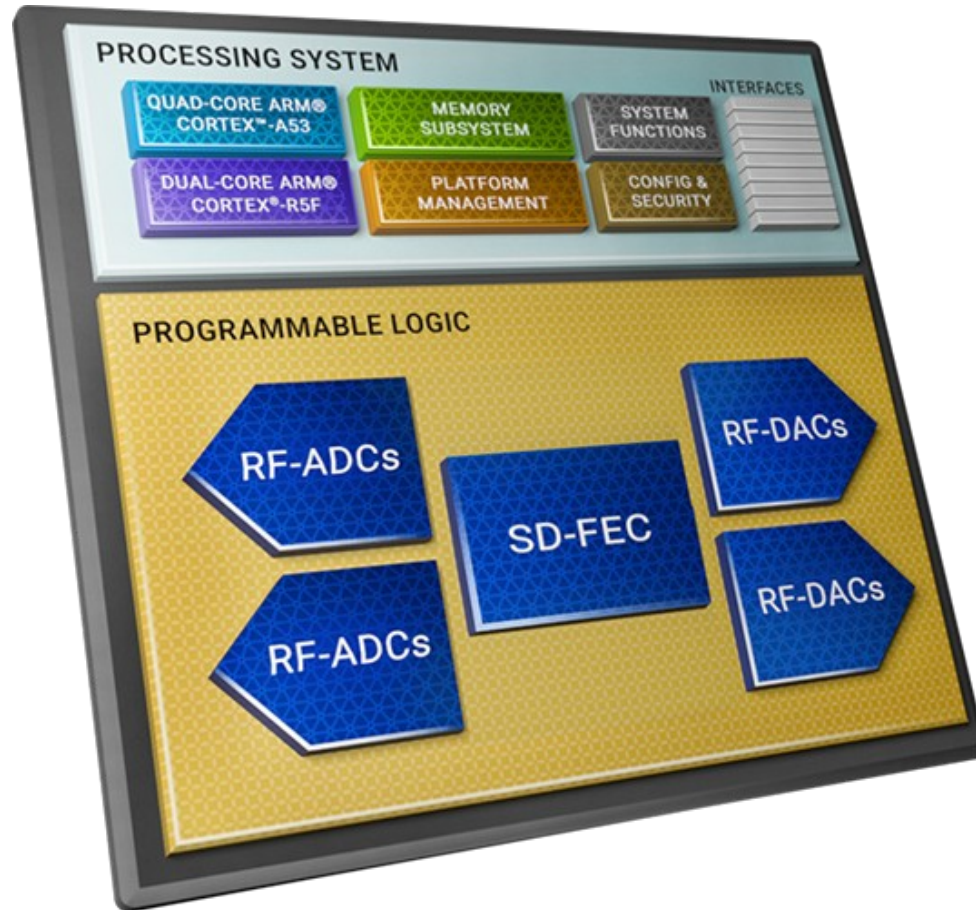
Signalaufbereitung



Frontend Board für ZCU111 Eval Board

1m² Detektor

Auswertung RFSoc



Schneller in die digitale Welt:

- Vorverstärkte Signale direkt abgetastet
- Zeitpunkt des Spitzenwerts/Umkehrpunkts in Logik (CFD)
- Ortsbestimmung per:
Samplenummer, Abtastfrequenz → Zeitdifferenz

Weitere Vorteile:

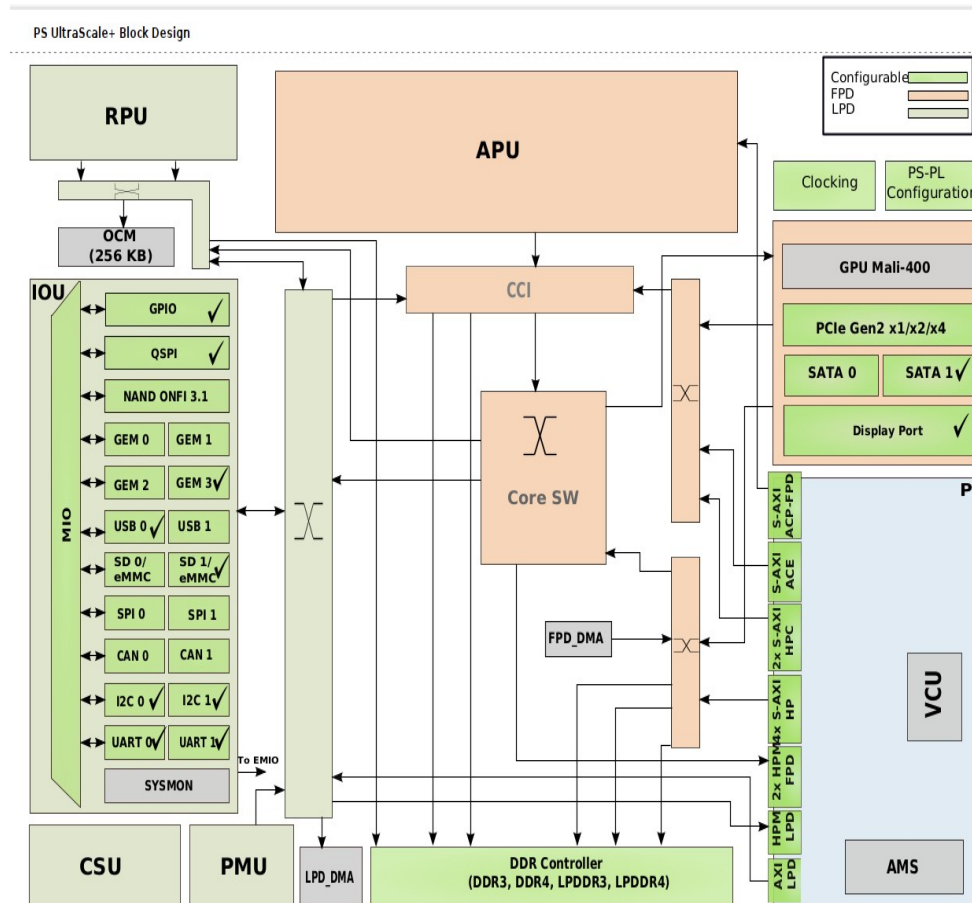
- Testsignale direkt mit DACs erzeugen
- Einstellbare Verstärkung und
- zuschaltbare Dämpfer zur Anpassung
- „Eingebautes Oszilloskop“ auf allen Kanälen

Beschaffung:

- weniger und universellere Bauteile
- RFSoc: Abhängigkeit und Preis

1m² Detektor

Auswertung RFSoc



RFSoc als Fortführung von Zynq und Ultrascale-SoC

- Leistungsstarkes uC-System
 - 64bit Multicore Anwendungsprozessor
 - 32bit Multicore Echtzeitprozessor
 - Verwaltungsprozessoren
 - leistungsstarkes kohärentes Bussystem
 - leistungsfähige, vielfältige Peripherie u.a. mit Speichercontroller und ggf. GPU
 - FPGA als weitere flexible Peripherie
-
- **Sehr schnelle ADCs und DACs mit brauchbarer Spannungsauflösung**

1m² Detektor

RFSoc Analog Block

IP Symbol

ADC Physical Resources

DAC Physic

Show disabled ports

+ s_axi

+ s00_axi

+ s01_axi

+ s02_axi

+ s03_axi

+ s10_axi

+ s11_axi

+ s12_axi

+ s13_axi

+ adc0_clk

+ adc1_clk

+ adc2_clk

+ adc3_clk

+ dac0_clk

+ dac1_clk

+ vin0_01

+ vin1_01

+ vin2_01

+ vin3_01

+ sysref_in

+ dac0_rts

+ dac1_rts

+ adc0_rts

+ adc1_rts

+ adc2_rts

+ adc3_rts

+ s_axi_aclk

+ s_axi_aresetn

+ m00_axi_aresetn

+ m01_axi_aresetn

+ m10_axi_aresetn

+ m11_axi_aresetn

+ m20_axi_aresetn

+ m30_axi_aresetn

+ s00_axi_aclk

+ s01_axi_aclk

+ s02_axi_aclk

+ s03_axi_aclk

+ s10_axi_aclk

+ s11_axi_aclk

+ s12_axi_aclk

+ s13_axi_aclk

m00_axi

m02_axi

m10_axi

m12_axi

m20_axi

m30_axi

vout00

vout01

vout02

vout03

vout10

vout11

vout12

vout13

clk_adc0

clk_adc1

clk_adc2

clk_adc3

clk_dac0

clk_dac1

irq

Component Name

analog/data_converter/rfdc

Basic

System Clocking

Advanced

System Configuration

Preset

Start from scratch

Converter Setup

Converter Setup

Advanced

Changing Converter Setup to Simple will cause current Advanced IP configuration to be lost.

RF-ADC

RF-DAC

ADC Tile 224

ADC Tile 225

ADC Tile 226

ADC Tile 227

Multi Tile Sync

Converter Band Mode

Link Coupling

Enable Multi Tile Sync

Band

Single

Link Coupling

AC

Converter Configuration

ADC 0

ADC 1

Enable ADC

Invert Q Output

Dither

Bypass Background Calibration

Data Settings

Digital Output Data

Real

Decimation Mode

2x

Samples per AXI4-Stream Cycle

8

Required AXI4-Stream clock: 256.000 Mhz

Mixer Settings

Mixer Type

Bypassed

Mixer Mode

Real->Real

Analog Settings

Nyquist Zone

Zone 1

Calibration Mode

Mode2

The diagram illustrates the internal architecture of the ADC Physical Resources. It shows two main sections: ADC0 and ADC1. Each section contains an ADC block (ADC0 or ADC1) connected to a Mixer block. The Mixer blocks are further connected to DAC blocks (DAC0 or DAC1). The DAC blocks are connected to various output ports (m00, m01, m02, m03). The diagram also shows the connection of the ADC blocks to the DAC blocks via a common bus or signal path.

Component Name

analog/data_converter/rfdc

Basic

System Clocking

Advanced

AXI4-Lite Interface Configuration

AXI4-Lite Clock (MHz)

100

Tile Clocking Settings

Tile	Sampling Rate (GSPS)	Max Fs (GSPS)	PLL	Reference Clock (MHz)	PLL Ref Clock (MHz)	Ref Clock Divider	Fabric Clock (MHz)	Clock Out (MHz)
ADC 224	4.096	4.096	☒	409.600	409.6	1	256.000	256.000
ADC 225	4.096	4.096	☒	409.600	409.6	1	256.000	256.000
ADC 226	4.096	4.096	☒	409.600	409.6	1	256.000	256.000
ADC 227	4.096	4.096	☒	409.600	409.6	1	256.000	256.000
DAC 228	4.096	6.554	☒	409.600	409.6	1	256.000	256.000
DAC 229	4.096	6.554	☒	409.600	409.6	1	256.000	256.000

PLL Summary Settings

Tile	Vco (MHz)	Fb Div	M	R
ADC 224	12288.0	30	3	1
ADC 225	12288.0	30	3	1
ADC 226	12288.0	30	3	1
ADC 227	12288.0	30	3	1
DAC 228	12288.0	30	3	1
DAC 229	12288.0	30	3	1

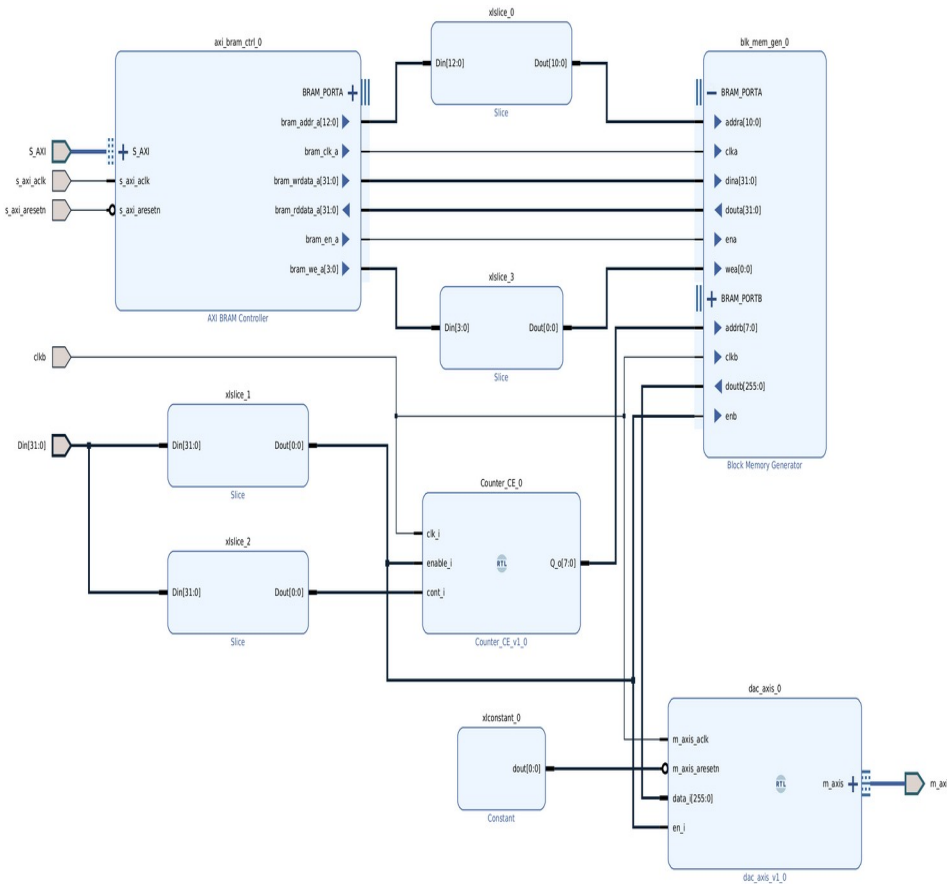
17

Signale aus einem Neutronendetektor mit einem RFSoc verarbeitet

Helmholtz-Zentrum hereon

1m² Detektor

RFSoc Analog Block

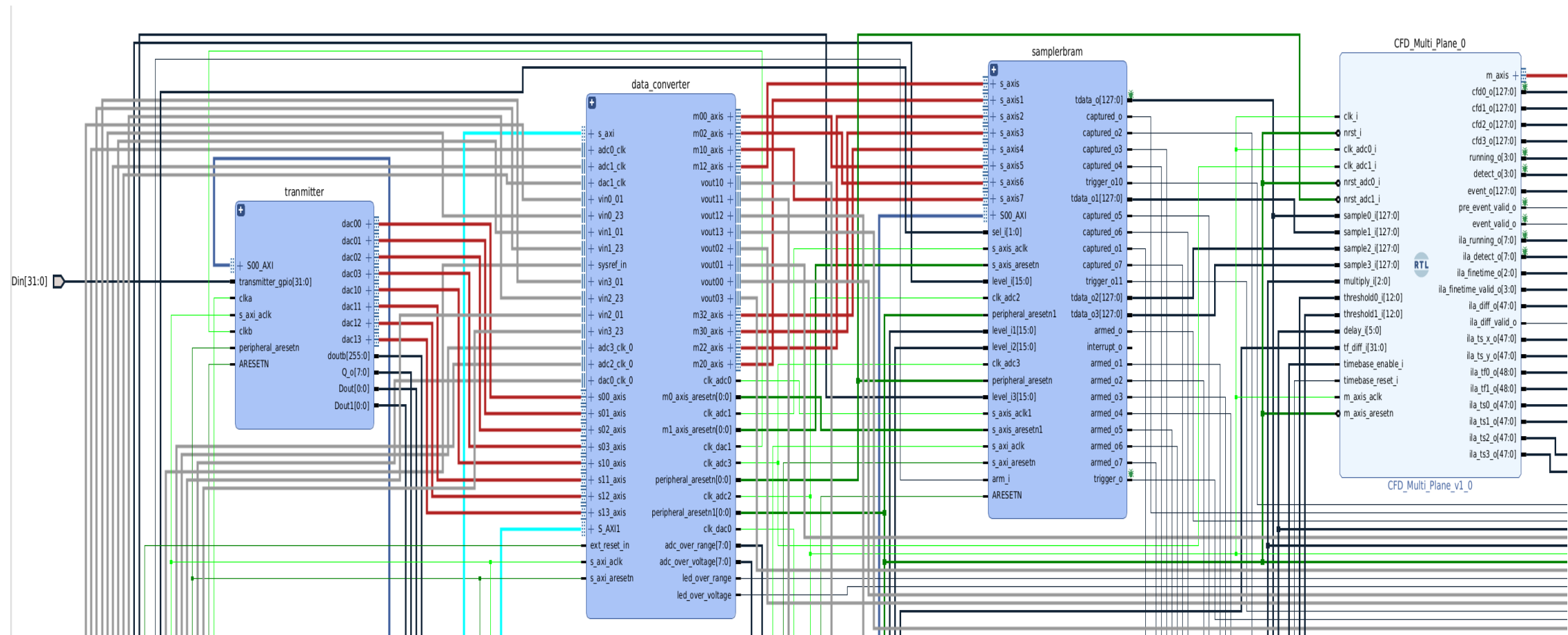


Einfaches Beispiel: Daten per DAC ausgeben

- Dual Ported RAM als Interface
 - Ein Port am AXI-Stream Bus
 - Zweiter Port Daten direkt am DAC, Adresse per Zähler incrementieren

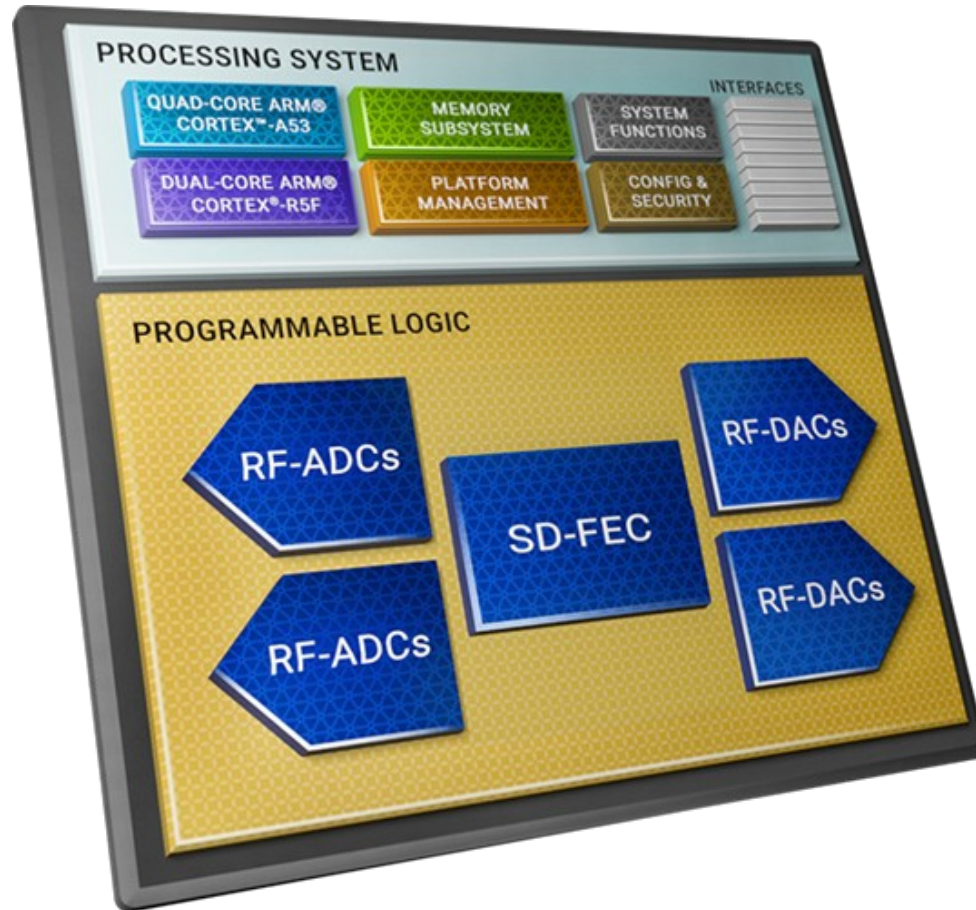
1m² Detektor

RFSoc Analog Block



1m² Detektor

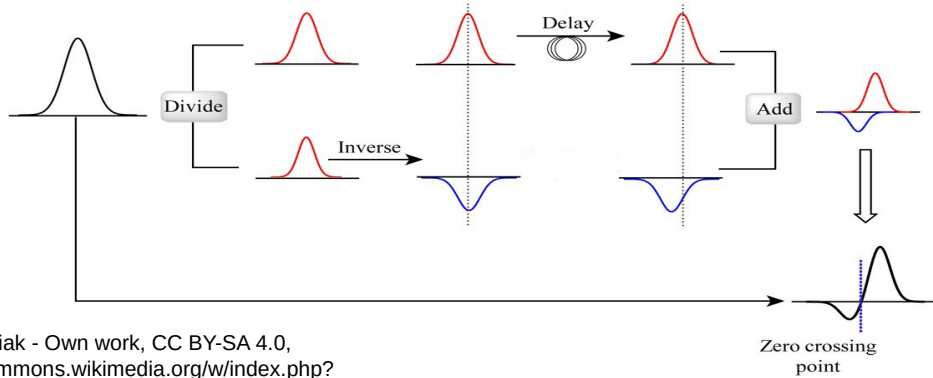
CFD in Logik



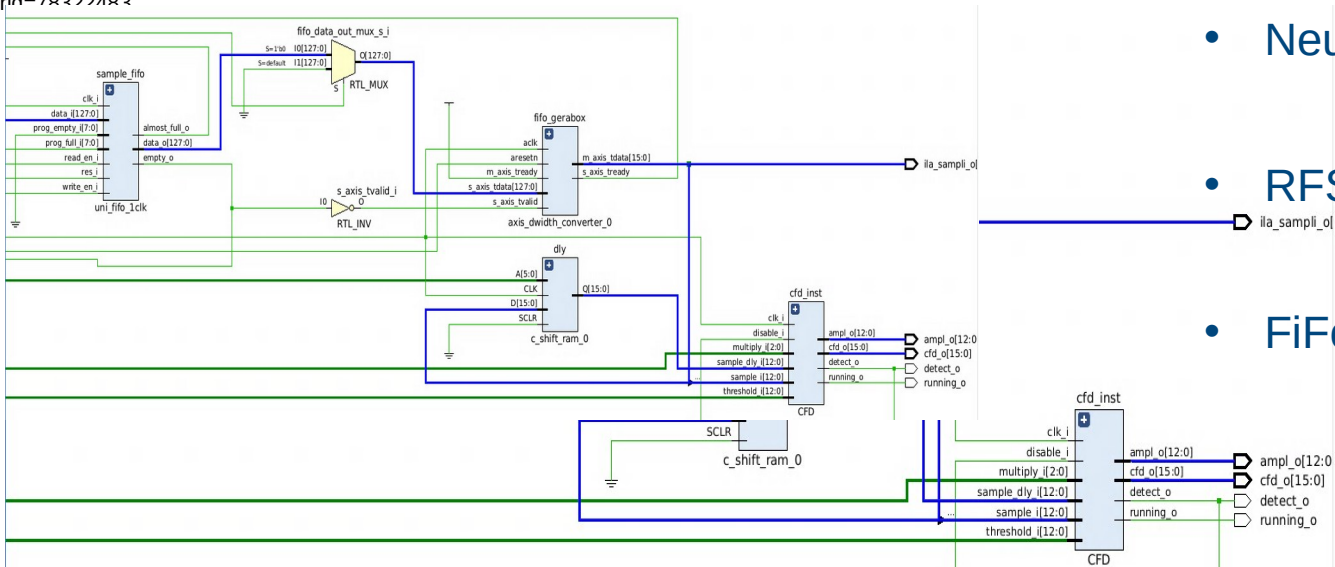
- ADCs einer Ebene (sogar zwei) laufen synchron, daher reicht Sample# für Differenz-Zeit-Ermittlung
- Phasensynchronisation der ADCs auch zu anderen SOC's mehrerer Ebenen nicht nötig, nur gemeinsamer Start (< 100ns) und synchroner Lauf.
- Erstmal geringere Zeitauflösung aber homogene bins ohne Physical Synthesis usw.
- Mehrere Möglichkeiten zur Suche des Umkehrpunkts oder Spitzenwerts, Nachahmung des analogen Algorithmus mit Verzögerung und Differenzbildung aber recht zielführend
- Viel mehr Flexibilität z.B. für die Delays, Optimierung und Diagnose incl. Fernwartung

1m² Detektor

CFD in Logik



By Tdudziak - Own work, CC BY-SA 4.0,
<https://commons.wikimedia.org/w/index.php?curid=78277182>



- Funktionsweise Constant Fraction Discriminator
Signal Splitten, ein zwei Verzögern, den anderen invertieren. Beide wieder addieren.
→ Zeitpunkt Nulldurchgang entspricht Spitzenwert
- Zuvor mit Delay-Platine und OP-Amps diskret aufgebaut
- Neu: realisiert in VHDL
- RFSoc ADC: 128bit@256MHz (8 Sample pro Takt)
- FiFo mit Pre-Trigger

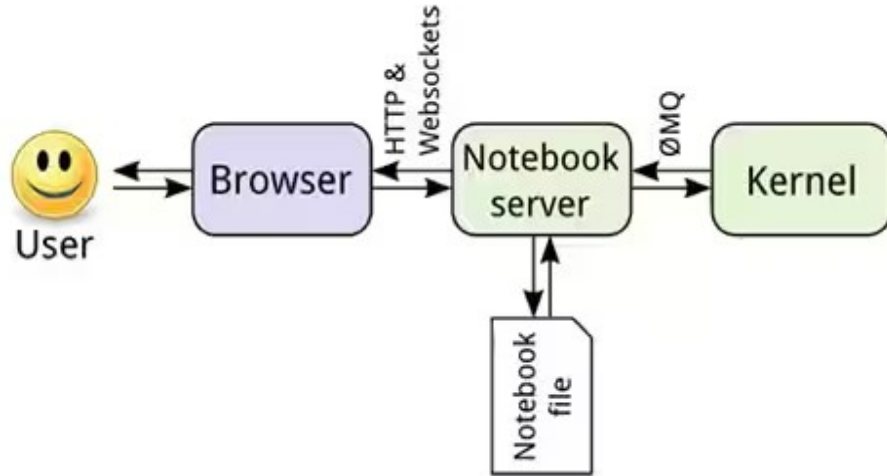
Zynq+Python = PYNQ

Jupyter, Python, Zynq



PYNQ

Python + Zynq



- Python on Zynq
- komfortabler abstrakter und dennoch direkter Zugang auch zur speziellen Peripherie auf dem Embedded-System mit den Möglichkeiten von Python (axi-gpio, axi-dma, axi-spi, bram, ...)
- Python bietet gutes Interfacing zu C - unter der hier wichtigen Vermeidung von Datenkopien
- Browser und vernetzter PC als universelles Bedien- Interface
- OS: Ubuntu, produktiv Petalinux
- Unterstützung vieler Python Bibliotheken im Browser (matplotlib, ...) und eigenem GUI Toolkit

Jupyter

Schnipsel

```
from pynq import Overlay
from pynq import MMIO, Interrupt
from pynq.lib import AxiGPIO
overlay = Overlay('/home/xilinx/hw/pynqdaq.bit', download=True)

sampler_ip = overlay.ip_dict['axi_gpio_sampler']

sampler_enable = AxiGPIO(sampler_ip).channel1
sampler_status = AxiGPIO(sampler_ip).channel2

sampler_enable.write(select<<1 | arm, 0xff)
sampler_status.read()

mmio_transmitter[0] = MMIO(overlay.bram_ctrl_0.mmio.base_addr, overlay.bram_ctrl_0.mmio.length)
mmio_transmitter[0].array[:2048] = pulseX0
```

Load FPGA bitfile

AXI GPIO Ch1

AXI GPIO Ch2

Write to AXI GPIO

Read from AXI GPIO

get reference

Write numpy Array to BRAM

Jupyter

Schnipsel

```
[6]: param_aux = 0
      param_filter = 1
      param_preamp = 0
      param_attenuator = 0x5
      # attenuator max 5

      if True:
          txbuffer = [0]*3

          txbuffer[0] = 0
          txbuffer[1] = ((param_aux&0x01)<<2) + (param_filter>>2)
          txbuffer[2] = (param_filter << 6) + (param_preamp << 4) + param_attenuator

          resp = xfer(txbuffer, mkl_spi, ss=0x1)
          resp = xfer(txbuffer, mkl_spi, ss=0x2)
          resp = xfer(txbuffer, mkl_spi, ss=0x4)
          resp = xfer(txbuffer, mkl_spi, ss=0x8)
```

```
SR 0x24
SR 0x24
SR 0x24
SR 0x24
```

```
# DMA data transver receive channel
```

```
dma = overlay.axi_dma_0
```

```
dma_recv = overlay.axi_dma_0.recvchannel
```

```
output_buffer = allocate(shape=(dma_size*2,), dtype=np.uint64)
```

```
dma_recv.transfer(output_buffer)
```

```
dma_recv.wait()
```

Jupyter

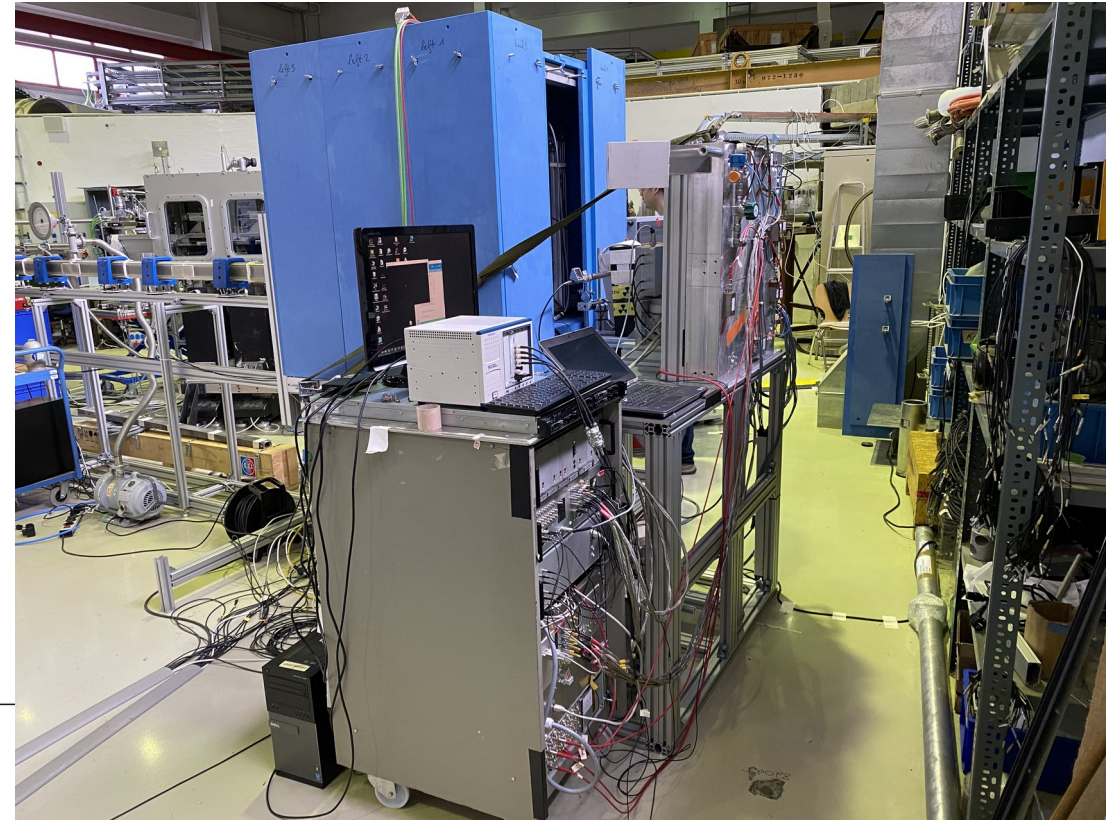
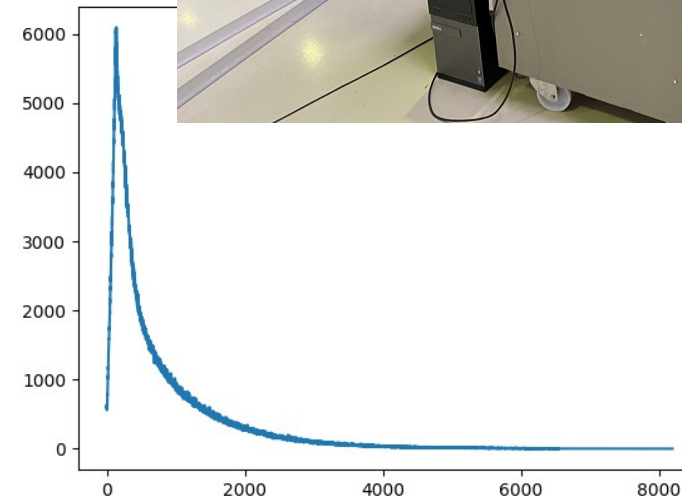
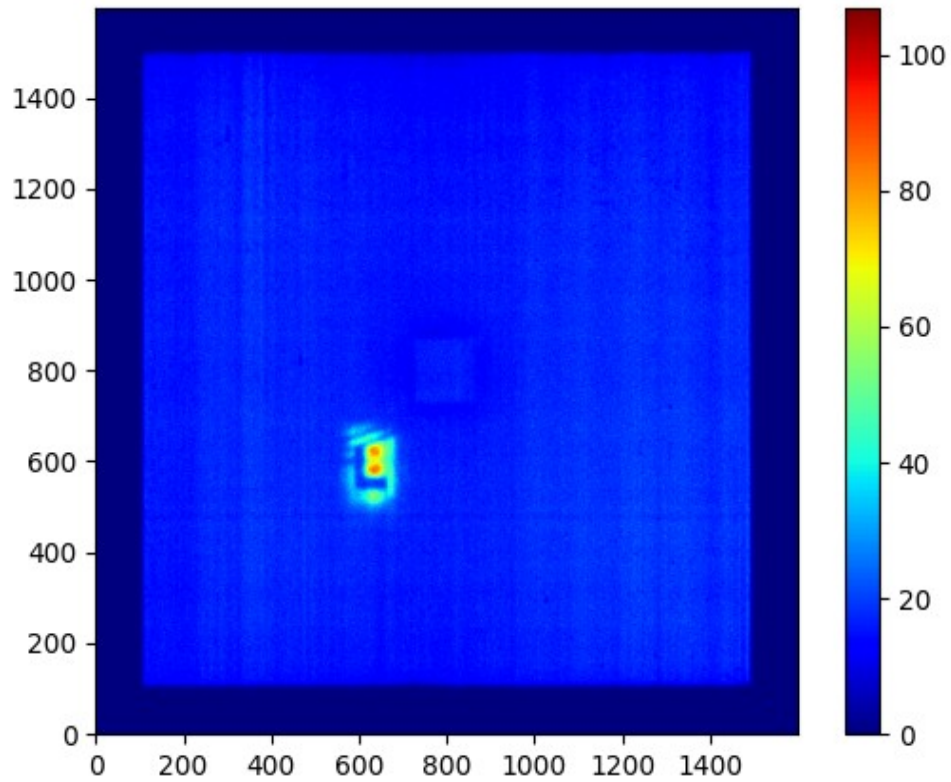
Was hat noch nicht funktioniert

- Gui mit Jupyter für mehrere User
- Charts zeigen Inhalte von Speichern aus der Logik an
- Daten werden von jeder Browser Session aus der HW geholt (DMA)
- Aber auch noch nicht viel Zeit investiert
→ Daten per UDP an anderen PC gesendet und dort visualisiert.

Messzeit Jülich 2023

HBS@BigKarl

- 500x500mm Helium3 Detektor
- „kollimierter“ Strahl
- 2D und Flugzeit



Vielen Dank

www.hereon.de