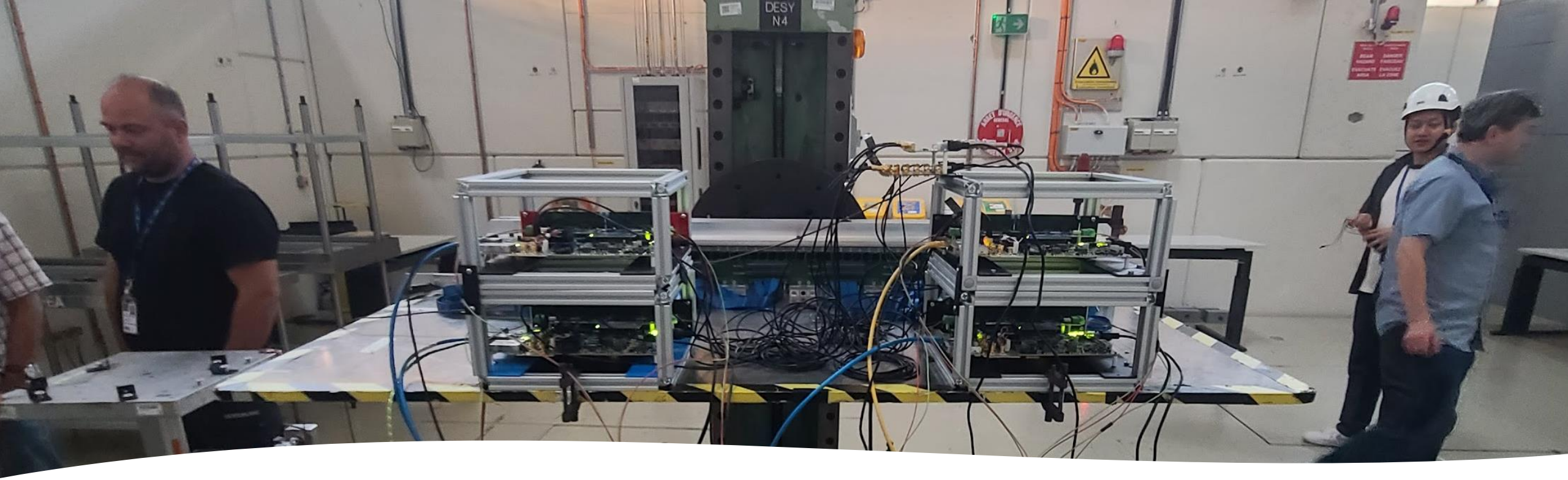


Using the HGCROC readout in test beams

Tristan Protzman

Lehigh University

NSF

LEHIGH UNIVERSITY

Outline

- HGCROC overview
- Hardware setup
- Operations
- Offline processing
- Test beam experience

- Notes:
  - Figures and information comes from H2GROC3 data sheet. Several other versions exist
  - I wrote these having not used the device in 3.5 months. To the best of my memory and notes the procedures are accurate, but I will update as we find issues

HGCROC Overview

- Custom ASIC for SiPM readout
  - Silicon version exists as well
- Developed for CMS High Granularity Calorimeter
- 72 analog channels
  - Additional 4 channels for common mode noise measurement and 2 for MIP calibration
- Digitizes input at 40 MHz, LHC bunch crossing rate
- Designed for triggered operation, “hacked” into streaming readout for test beams

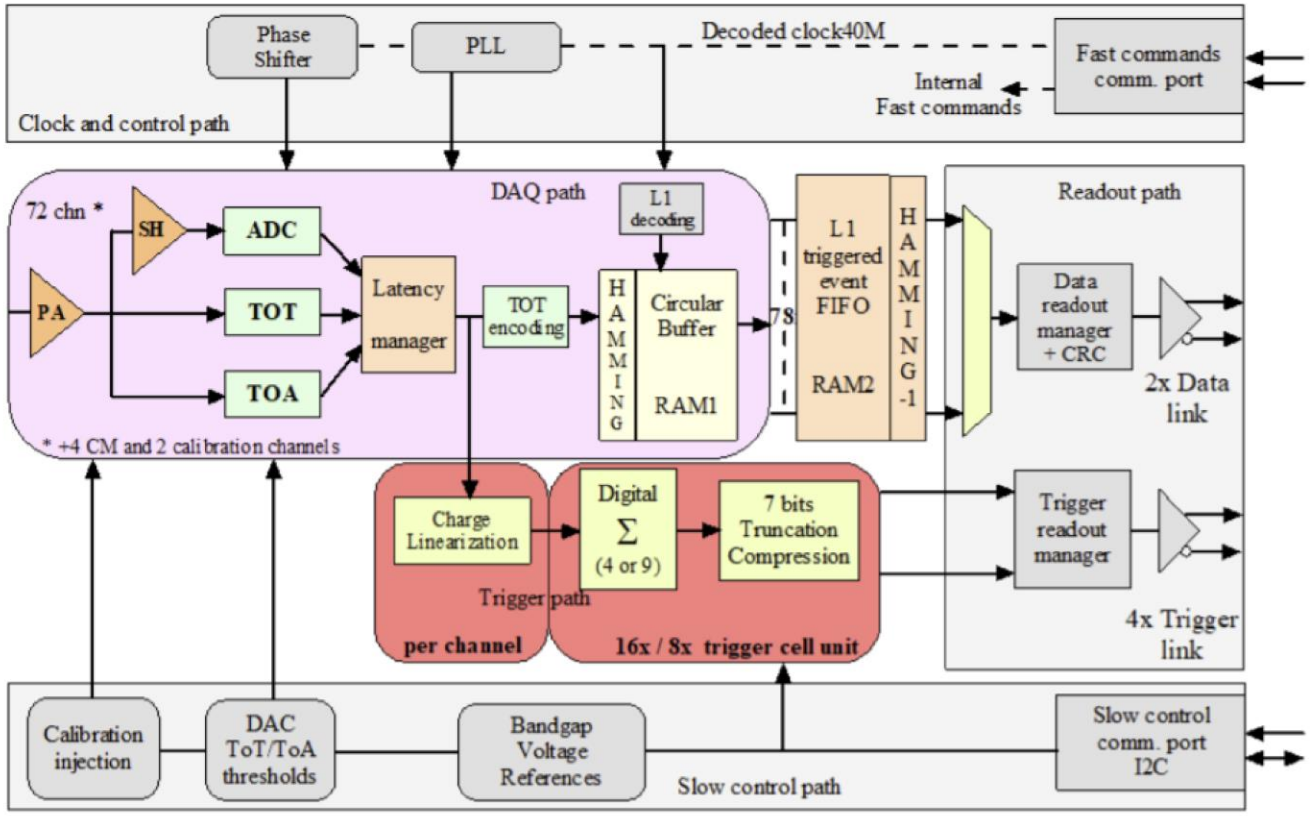


Figure 1.1: The block diagram for the HGCROC3 ASIC

Analog front end

- Low noise current conveyor input
  - Vdac – Channel-wise gain matching
  - Ibi, Ibo, dacb – Half and channel-wise current compensation
- Gain between 0.025 and 0.375, tunable over 4 bits

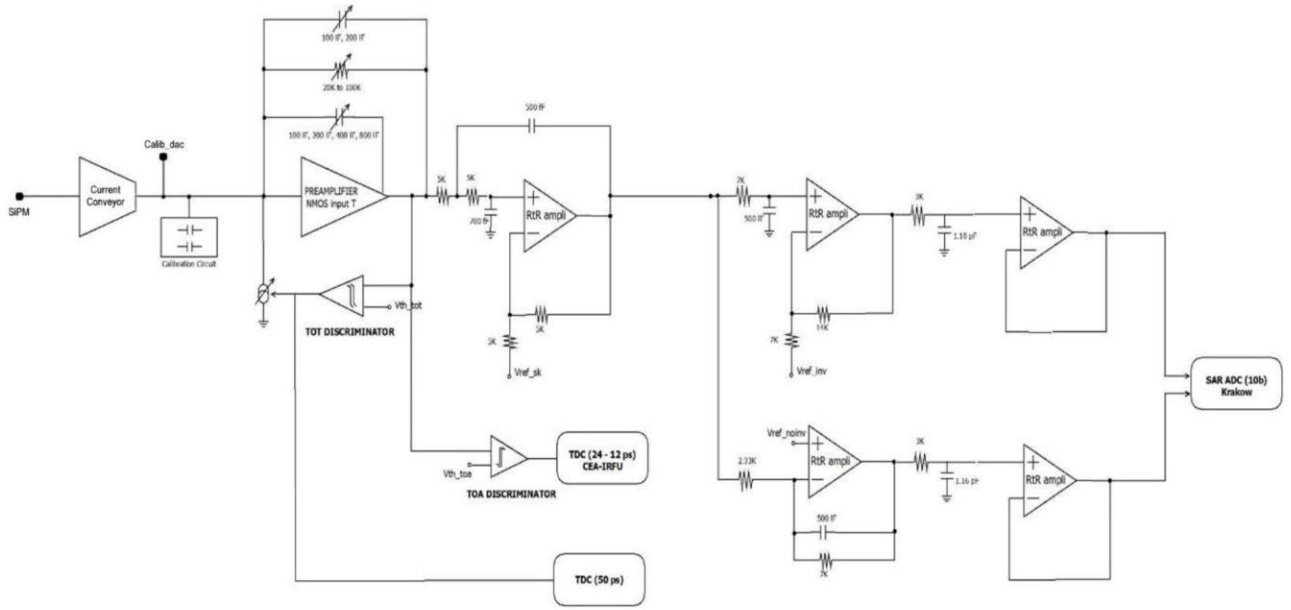
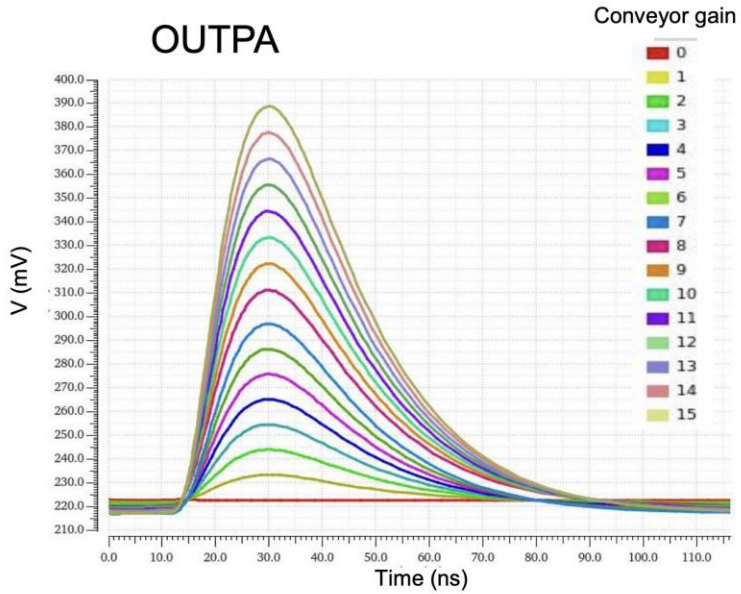


Figure 1.2: Analog channel architecture



Preamp output for 1 pC charge

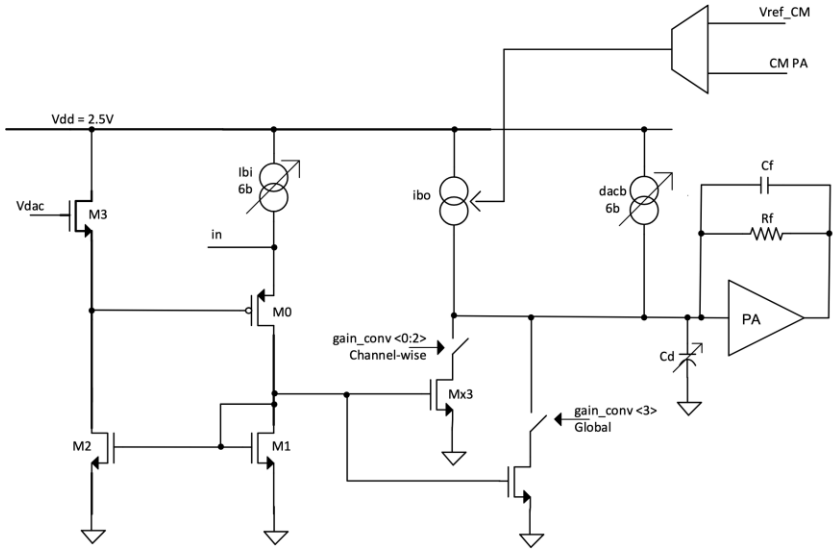
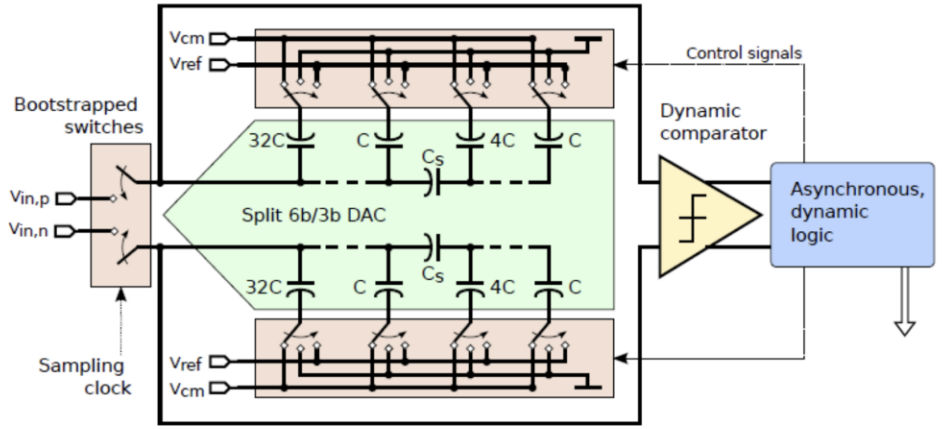
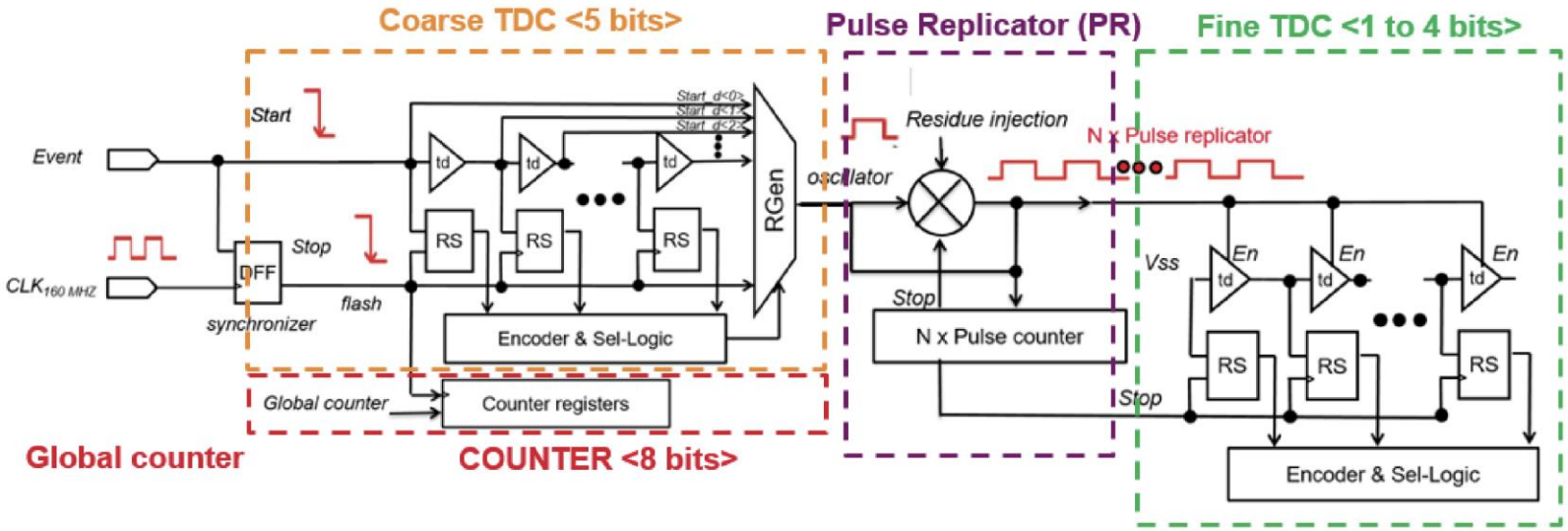


Figure 1.4: Current Conveyor architecture

Digitization

- ADC
  - 10 bit SAR ADC
  - 8 bit register for setting pedestal
- ToA
  - 10 bit, 25 ps resolution TDC
- ToT
  - 12 bit, 50 ps resolution TDC
- Signals from ADC and TDCs have different processing times, this is handled by the align block with 11 clock cycle latency





Registers

- The ASIC parameters are defined in 90 subblocks each with up to 32 8 bit registers
- Register values are defined over the I2C interface and divided into three classes

- Global
  - Affects all 72 ASIC channels
- Half
  - Each ASIC has two “Halfs”, channels 0 through 35 and 36 through 71
- Channel
  - Some registers exist at the per channel level

3.3 **“Global analog”** I2C parameters

Register # 0			
Bit	**Name**	**Default**	**Description**
0	ON_dac_trim	“1”	“1” = enable DC trimming bias
1	ON_input_dac	“1”	“1” = enable input DAC bias
2	ON_conv	“1”	“1” = enable conveyor bias
3	ON_pa	“1”	“1” = enable preamp bias
4	Idac_tot<0>	“0”	x
5	Idac_tot<1>	“0”	
6	Idac_tot<2>	“1”	
7	Gain_conv<3>	“1”	Conveyor Gain: <3> = 0.2

Register # 1			
Bit	**Name**	**Default**	**Description**
0	ON_rtr	“1”	“1” = enable shaper amplifiers bias
1	Sw_super_conv	“1”	“1” = enable preamp fast output bias
2	Idac_inv<0>	“0”	x
3	Idac_inv<1>	“0”	
4	Idac_inv<2>	“1”	
5	Dacb_vb_conv<0>	“0”	Conveyor current bias
6	Dacb_vb_conv<1>	“1”	
7	Dacb_vb_conv<2>	“1”	

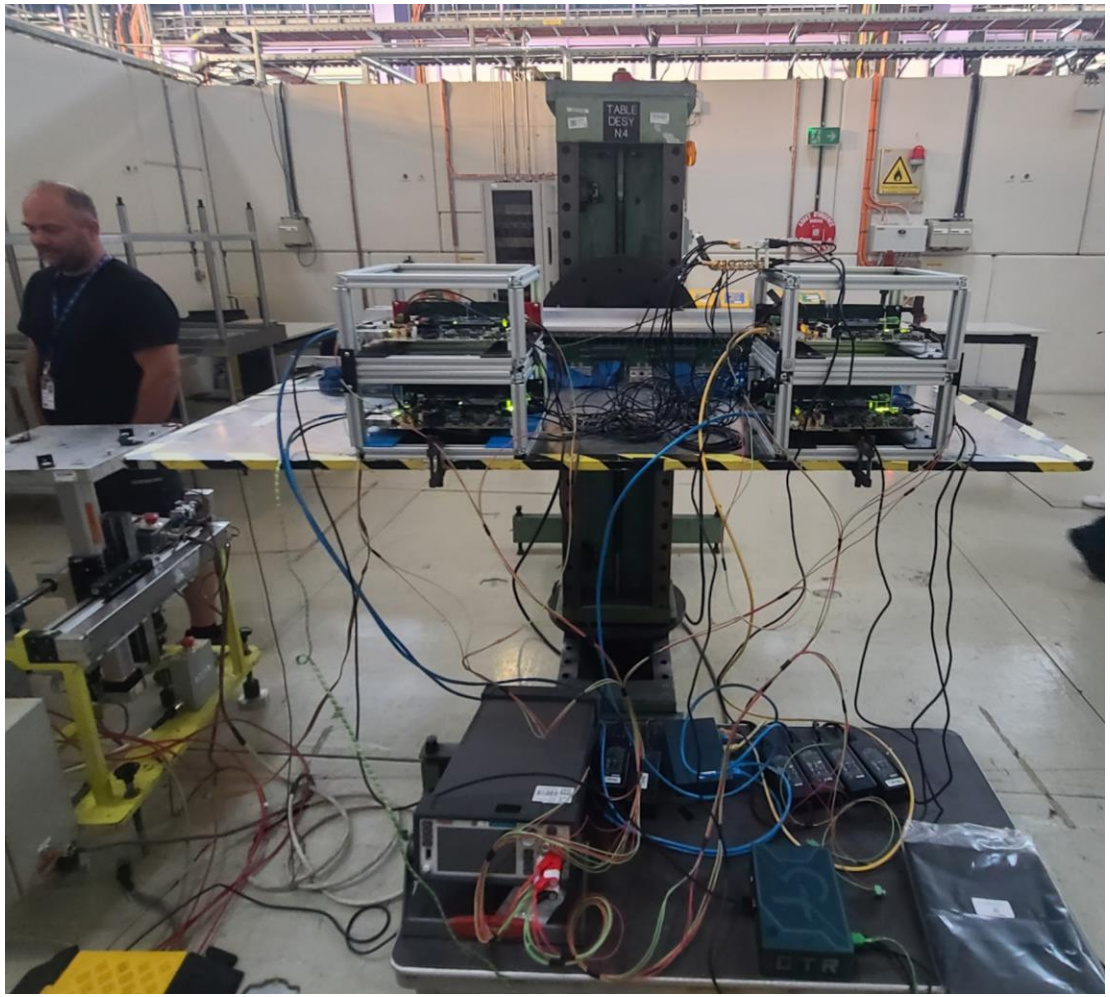
Operating modes

- Different output based on the total measured charge, except…
- We lock it into mode 4
  - ADC, ToT, and ToA for the current sample
  - Kinda makes it look like streaming readout system

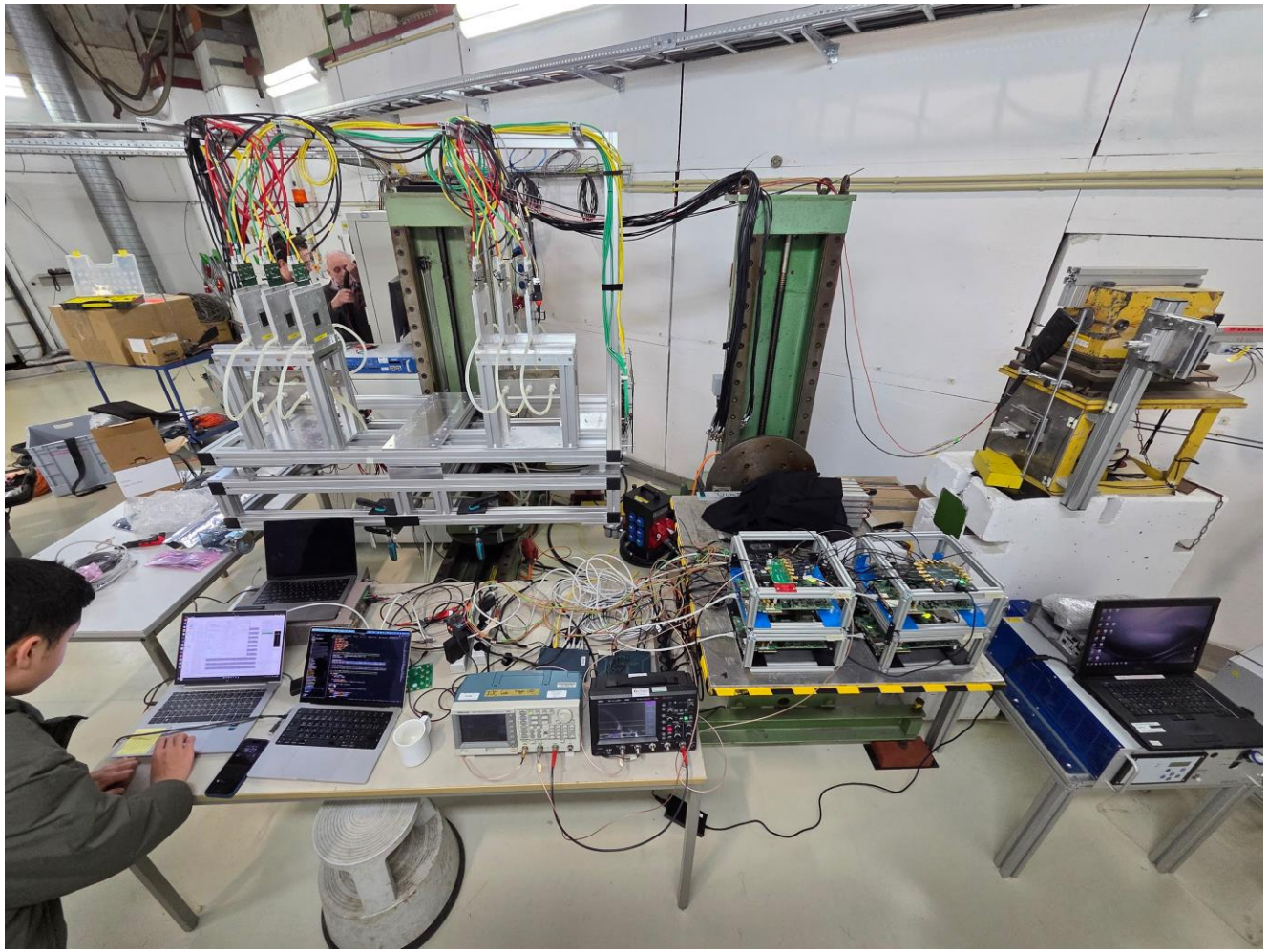
	ADC (BX-1)	ADC (BX)	TOT (BX)	TOA (BX)	Charge collection	Data type
1	x	x		x (=0)	Q < TOA_thr AN	Normal
2	x	x		x	Q < TOT_thr AN	Normal
3	x		x	x	Q > TOT_thr AN	Normal
4		x	x	x		“Characterization”

Test beam setups

Test beam setup



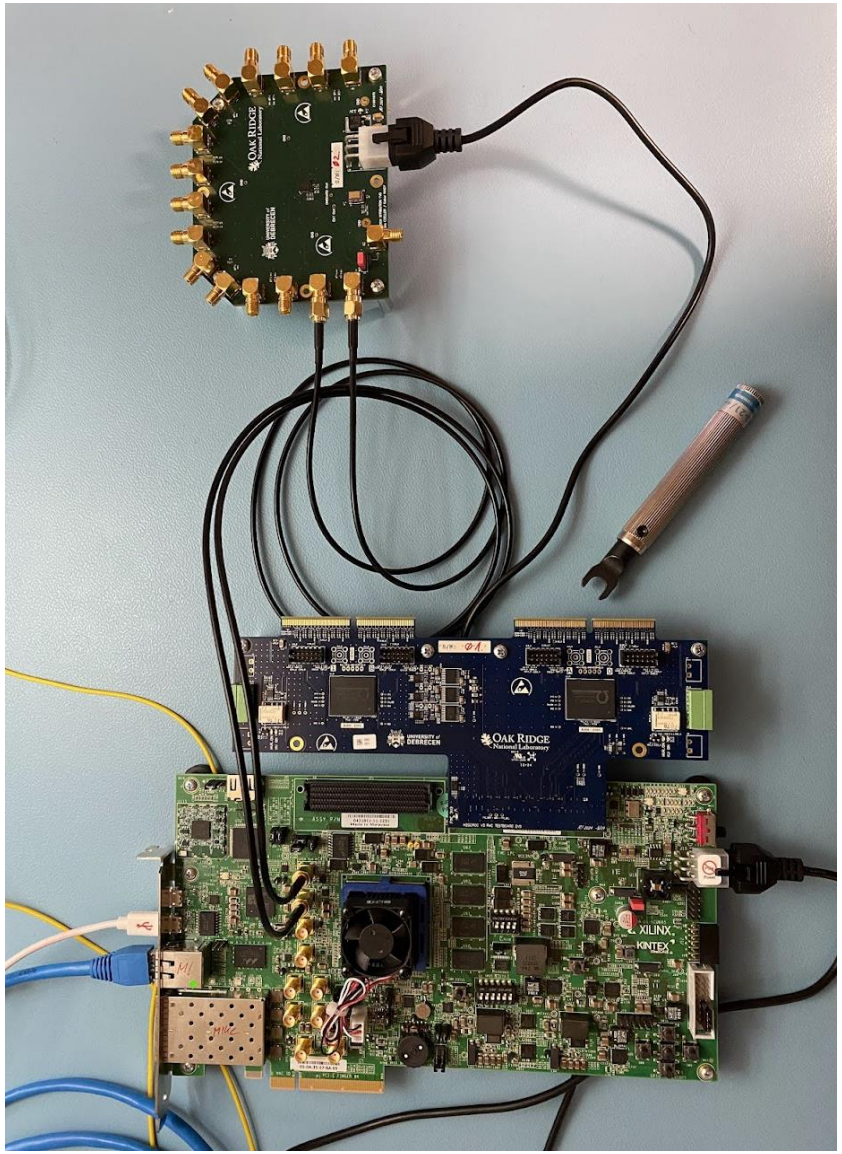
LFHCal @ CERN PS



EEEMCal @ DESY II

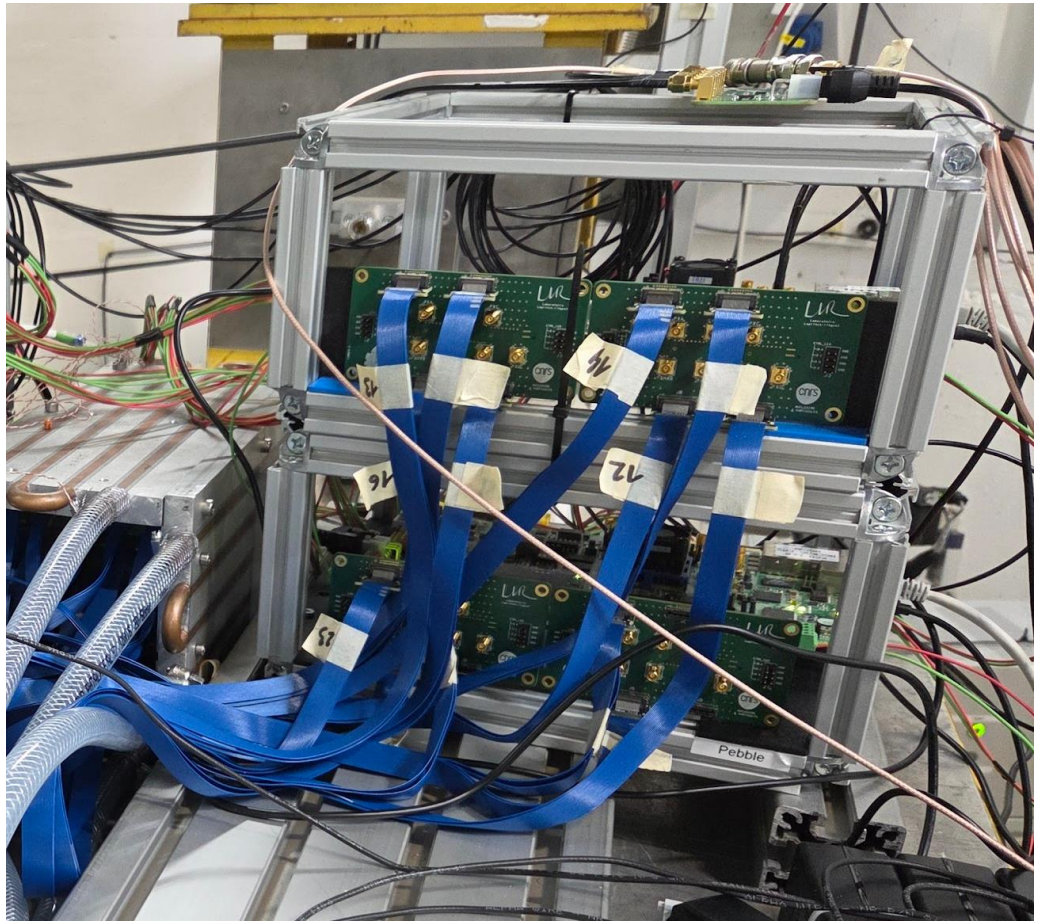
Basic setup

- Xilinx KCU
  - FPGA to interface between between DAQ/slow controls and ASIC
- Protoboard
  - Contains 2 ASICs, breaks out IO
- Clock
  - 40 MHz dual edge clock
  - Common to all KCU used in test beam
- Trigger board (Not shown)
  - Duplicates and distributes trigger to all KCU



Backplane

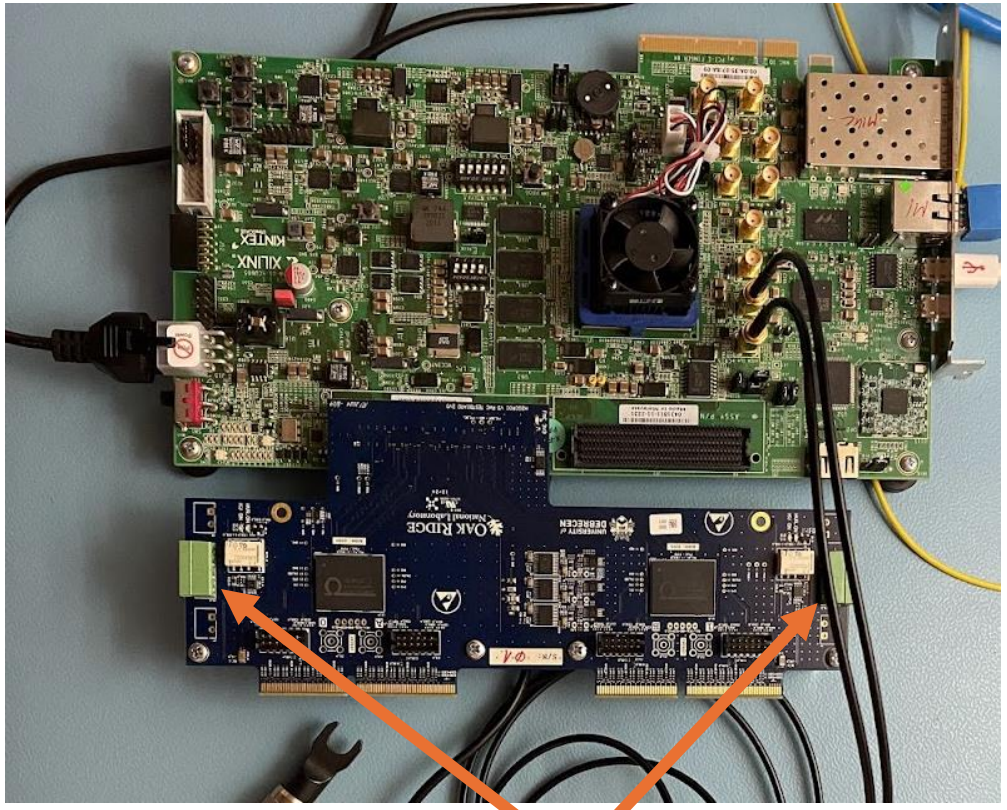
- Protoboard is adapted to the specific instrumentation via backplanes
- Common pitfall – likes to fall off
  - Easy to catch and fix, but keep in mind
  - Has ability to screw in place… because of some frequent switching we didn't take full advantage of this :)



EEEMCal @ DESY II

Bias

- Bias is externally supplied
- Can be switched on/off from DAQ software
- There are provisions for channel by channel adjustment, but I haven't seen an implementation for this yet

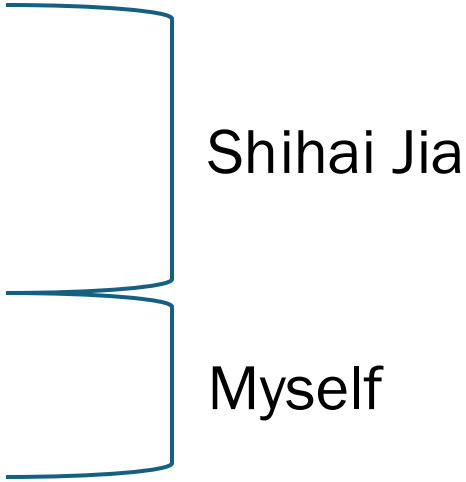


Network

- In DESY, we made use of two computers and two networks
- “DAQ/Slow Controls” Network
  - Linux laptop served as DAQ and control computer
  - Laptop kept in control room
  - Ethernet connection between laptop and each of the 4 KCUs
    - Kept separate from wider internet
- “FPGA Programming” Network
  - Windows PC with remote desktop software for reprogramming FPGAs
  - Kept in IR near detector
  - Mini-USB cable connect computer to each

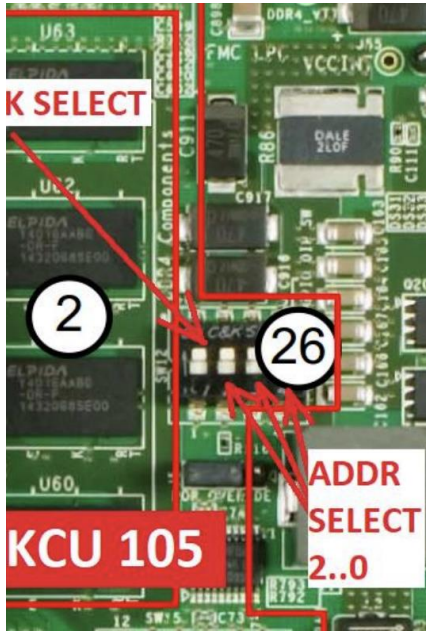
Software

- FPGA Programming: Vivado
- DAQ: H2GDAQ
- Configuration: H2GConfig
- Calibration: H2GCalib
- Online Monitoring: H2G Online Monitoring
- Decoding: H2G Decode



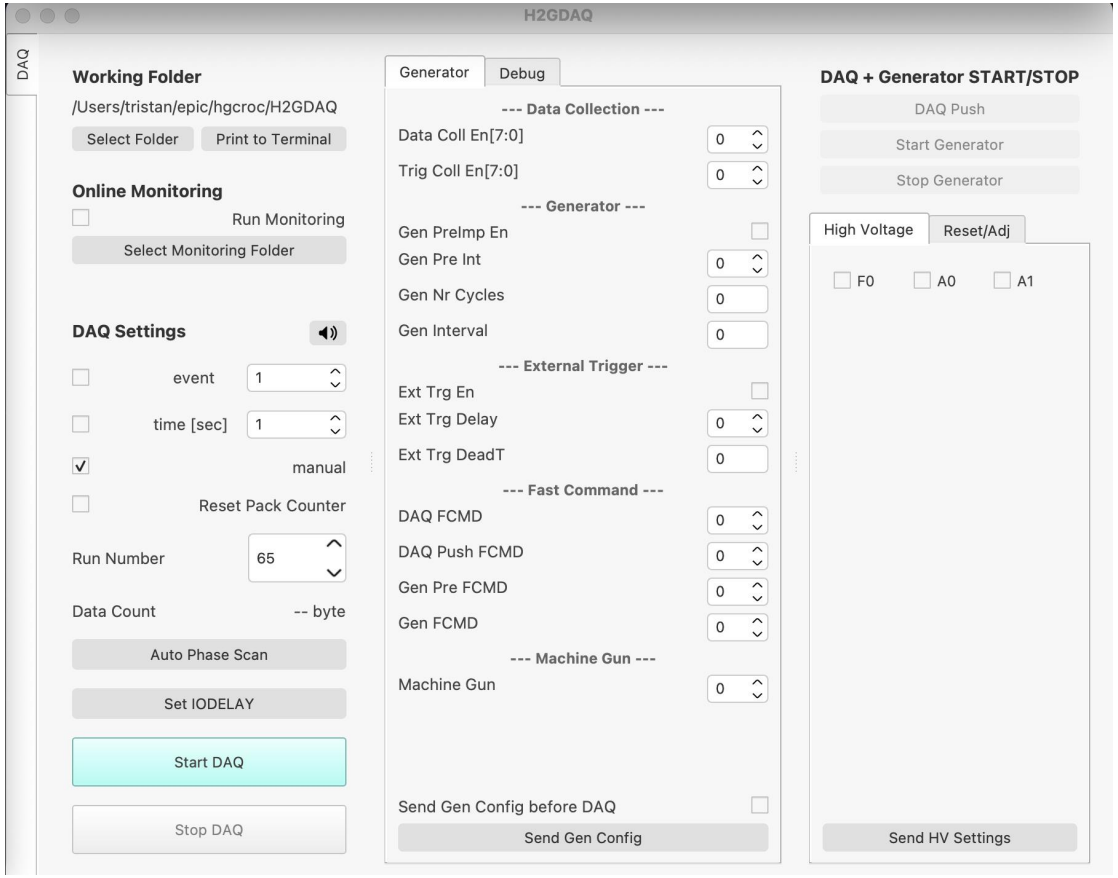
Initial setup

- Set DIP switches
  - Configures FPGA IP address
- Programming FPGAs
  - Over USB, load the desired firmware onto the KCU
  - Different versions exist for internal and external clock
  - This resets the ASIC configuration and switches off the bias (I believe)
  - Reprogramming acts as a hard reset, one step above a full power cycle



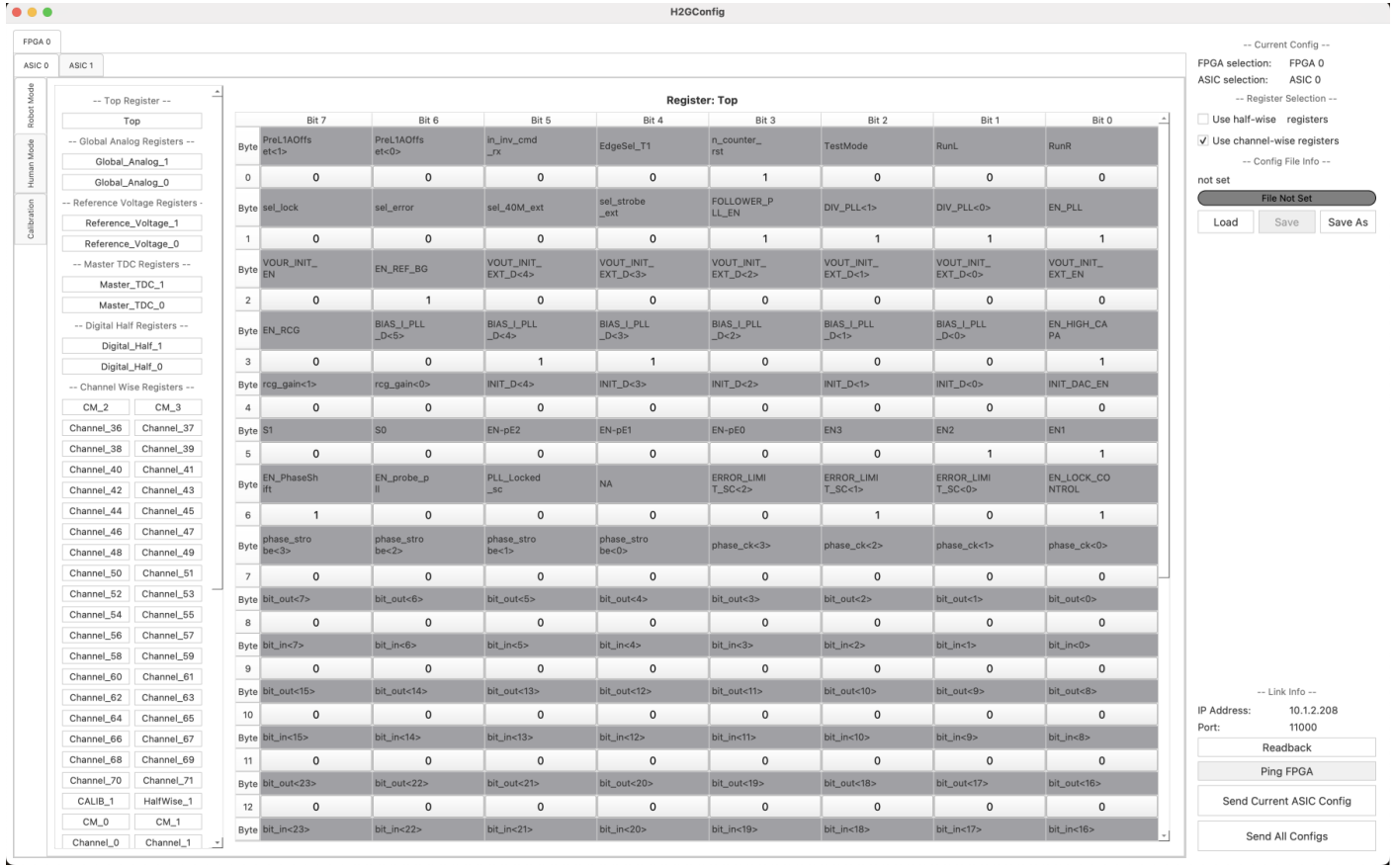
Initial setup continued

- Set IO Delay
  - Done from H2GDAQ
  - Finds the best phase for reading the 1.28 Gbps links
  - Must be done every time the FPGA is reprogrammed
  - Does a soft reset of the ASIC, rerun configuration after



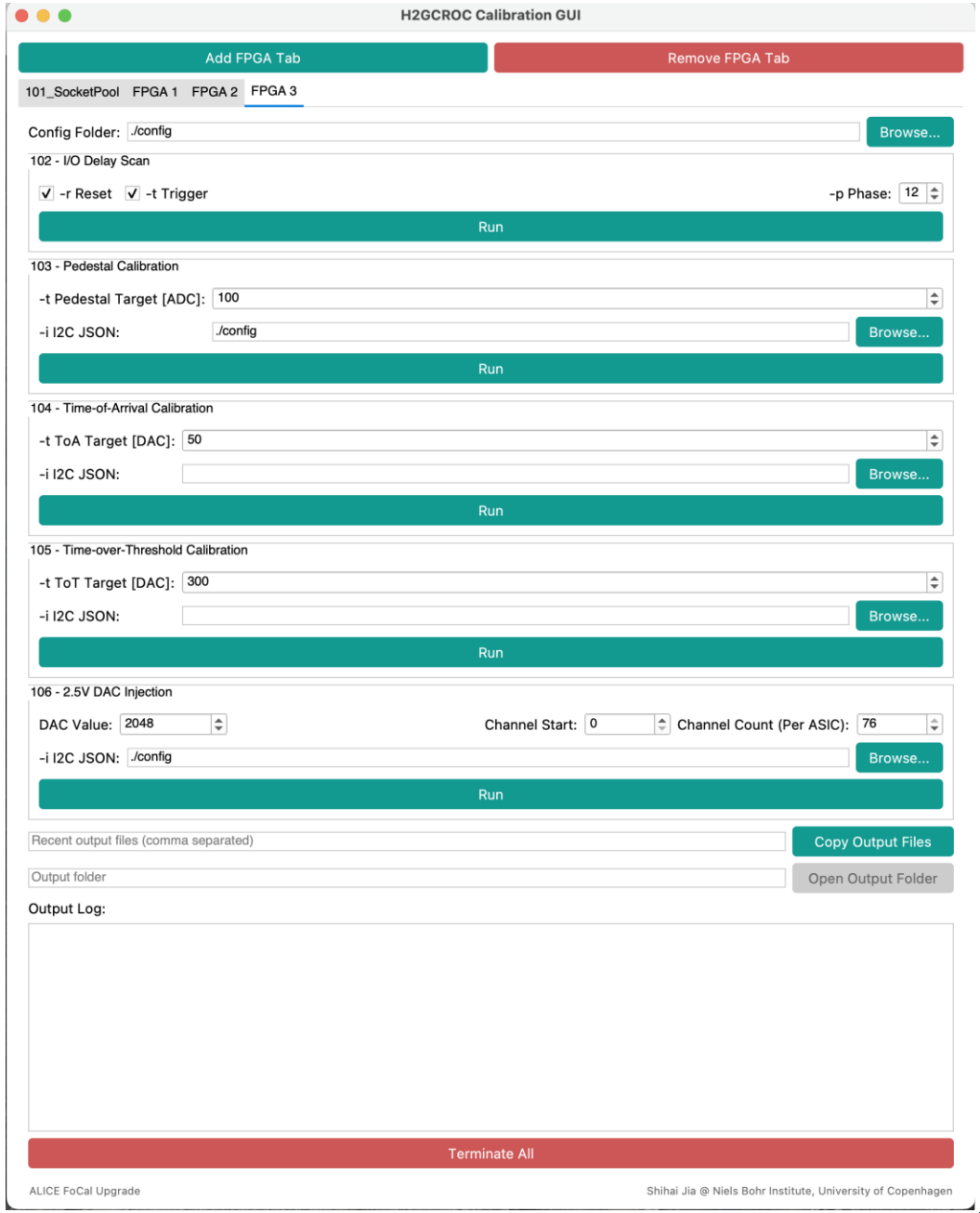
Initial setup continued

- Load initial configuration
  - Done from H2GConfig
  - Configurations can be saved to and loaded from JSON files
  - Can load a single ASIC or all connected ASICs
- Needs a base configuration defining operating mode, gain, etc
- Registers for pedestals and similar will be modified in next step

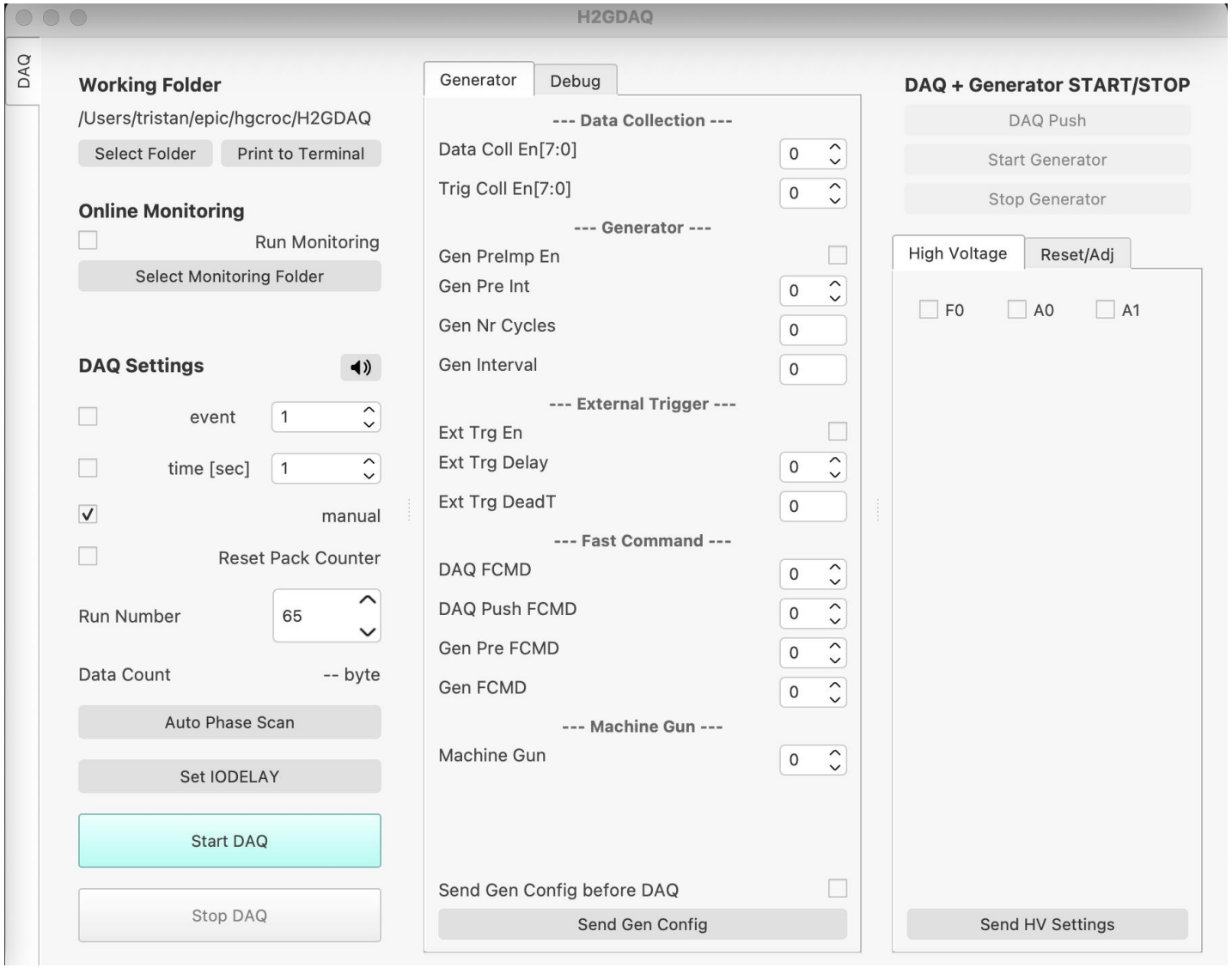


Initial calibrations

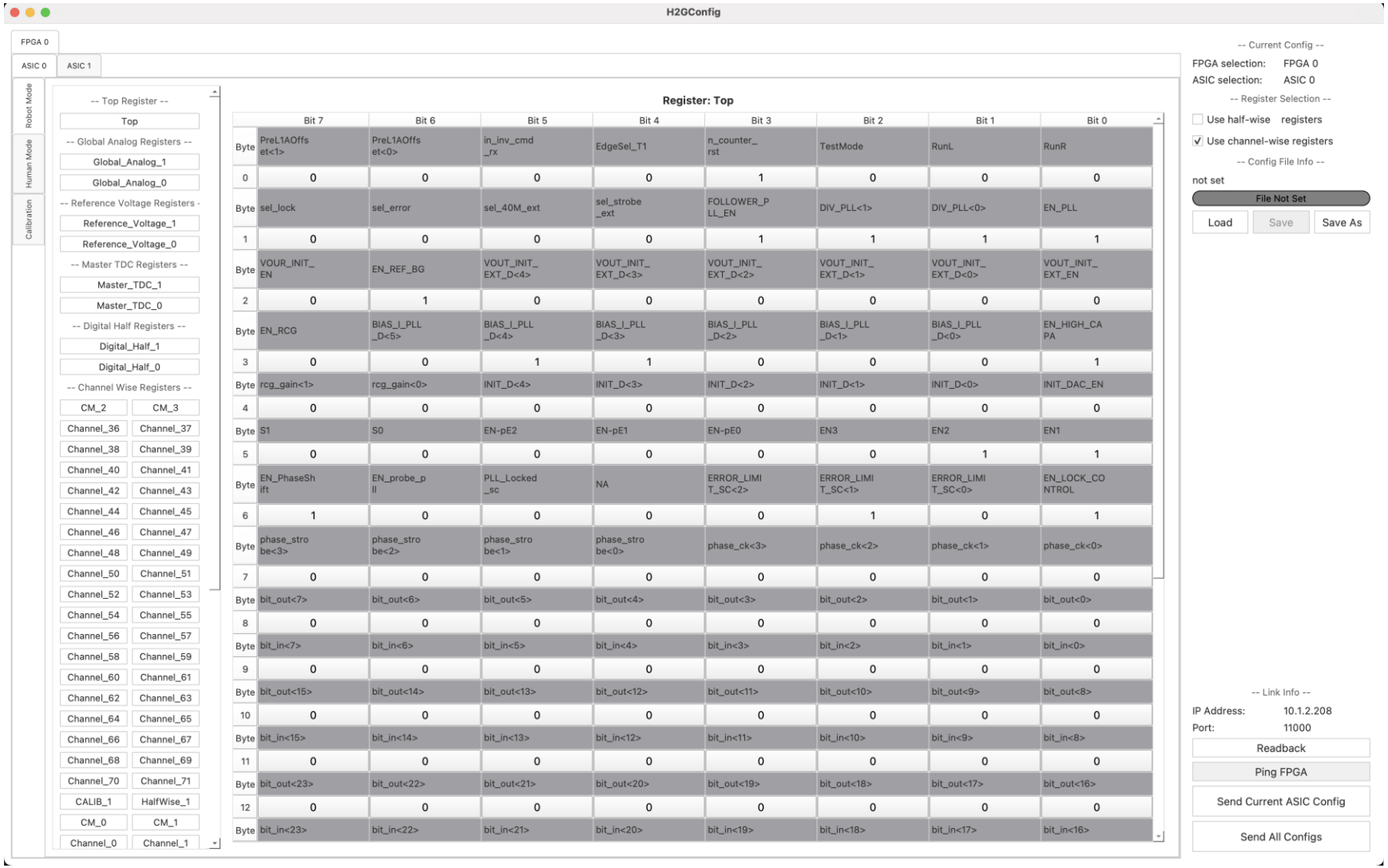
- The target ADC pedestal, ToT threshold, and ToA threshold must be set for the attacked device
  - At DESY, we redid this procedure each time we switched SiPM groupings
  - This can be a black hole for time
- h2gcalib provides a GUI for running these scans
- Shihai is continually working on these, I don't have firsthand experience with them in their current state yet



H2GDAQ

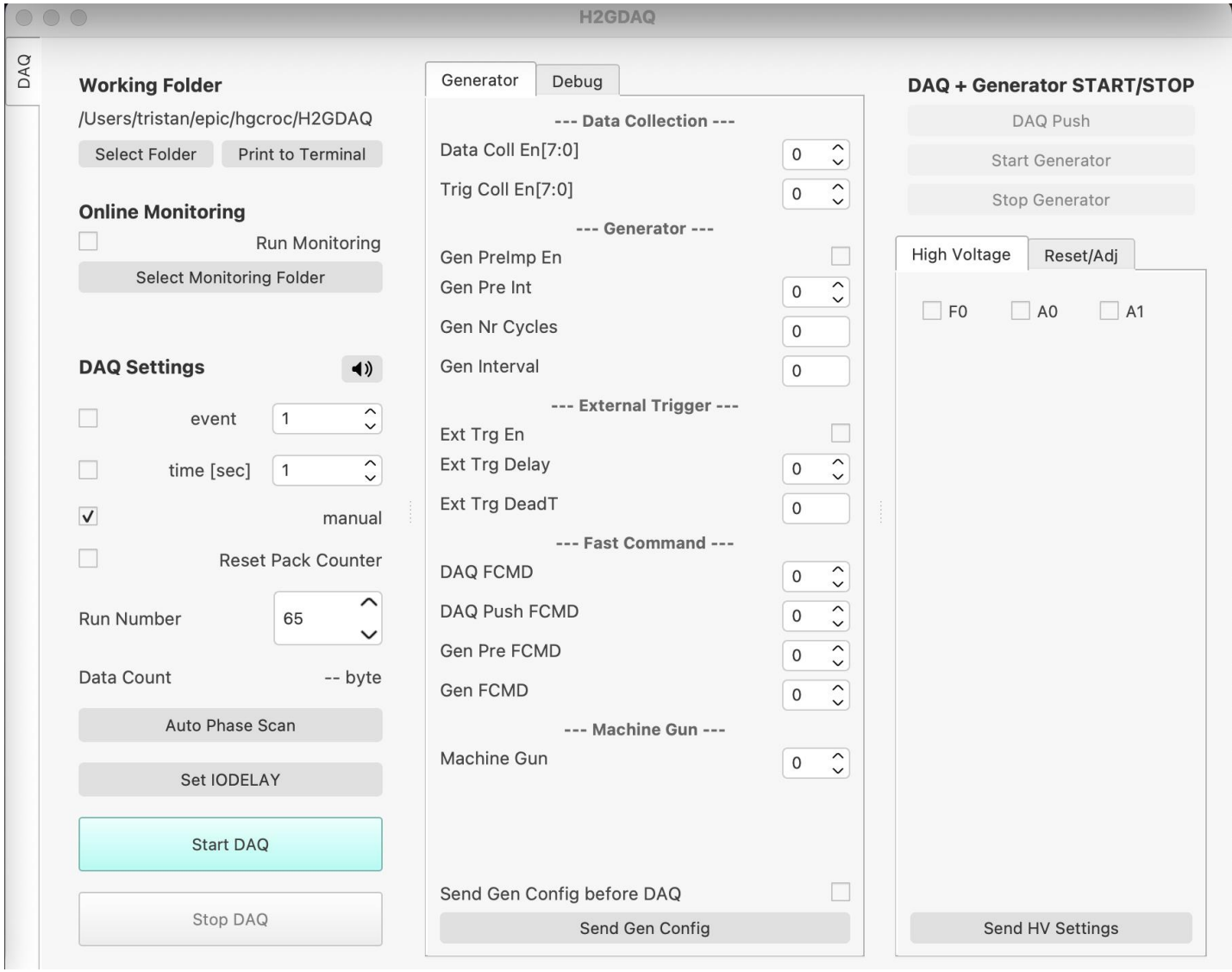

H2GDAQ
DAQ
Working Folder
/Users/tristan/epic/hgcroc/H2GDAQ
Select Folder
Print to Terminal
Online Monitoring
Run Monitoring
Select Monitoring Folder
DAQ Settings
event
1
time [sec]
1
manual
Reset Pack Counter
Run Number
65
Data Count
-- byte
Auto Phase Scan
Set IODELAY
Start DAQ
Stop DAQ
Generator
Debug
--- Data Collection ---
Data Coll En[7:0]
0
Trig Coll En[7:0]
0
--- Generator ---
Gen PreImp En
Gen Pre Int
0
Gen Nr Cycles
0
Gen Interval
0
--- External Trigger ---
Ext Trg En
Ext Trg Delay
0
Ext Trg DeadT
0
--- Fast Command ---
DAQ FCMD
0
DAQ Push FCMD
0
Gen Pre FCMD
0
Gen FCMD
0
--- Machine Gun ---
Machine Gun
0
Send Gen Config before DAQ
Send Gen Config
DAQ + Generator START/STOP
DAQ Push
Start Generator
Stop Generator
High Voltage
Reset/Adj
F0
A0
A1
Send HV Settings

H2GConfig


H2GConfig
FPGA 0
ASIC 0
ASIC 1
Robot Mode
Human Mode
Calibration
-- Top Register --
Top
-- Global Analog Registers --
Global_Analog_1
Global_Analog_0
-- Reference Voltage Registers --
Reference_Voltage_1
Reference_Voltage_0
-- Master TDC Registers --
Master_TDC_1
Master_TDC_0
-- Digital Half Registers --
Digital_Half_1
Digital_Half_0
-- Channel Wise Registers --
CM_2
CM_3
Channel_36
Channel_37
Channel_38
Channel_39
Channel_40
Channel_41
Channel_42
Channel_43
Channel_44
Channel_45
Channel_46
Channel_47
Channel_48
Channel_49
Channel_50
Channel_51
Channel_52
Channel_53
Channel_54
Channel_55
Channel_56
Channel_57
Channel_58
Channel_59
Channel_60
Channel_61
Channel_62
Channel_63
Channel_64
Channel_65
Channel_66
Channel_67
Channel_68
Channel_69
Channel_70
Channel_71
CALIB_1
HalfWise_1
CM_0
CM_1
Channel_0
Channel_1
Register: Top
Bit 7
Bit 6
Bit 5
Bit 4
Bit 3
Bit 2
Bit 1
Bit 0
Byte PreL1AOffset<1> PreL1AOffset<0> in_inv_cmd_rx EdgeSel_T1 n_counter_rst TestMode RunL RunR
0 0 0 0 0 1 0 0 0
Byte sel_lock sel_error sel_40M_ext sel_strobe_ext FOLLOWER_PLL_EN DIV_PLL<1> DIV_PLL<0> EN_PLL
1 0 0 0 0 1 1 1 1
Byte VOUR_INIT_EN EN_REF_BG VOUT_INIT_EXT_D<4> VOUT_INIT_EXT_D<3> VOUT_INIT_EXT_D<2> VOUT_INIT_EXT_D<1> VOUT_INIT_EXT_D<0> VOUT_INIT_EXT_EN
2 0 1 0 0 0 0 0 0
Byte EN_RCG BIAS_I_PLL_D<5> BIAS_I_PLL_D<4> BIAS_I_PLL_D<3> BIAS_I_PLL_D<2> BIAS_I_PLL_D<1> BIAS_I_PLL_D<0> EN_HIGH_CAPA
3 0 0 1 1 0 0 0 1
Byte rcg_gain<1> rcg_gain<0> INIT_D<4> INIT_D<3> INIT_D<2> INIT_D<1> INIT_D<0> INIT_DAC_EN
4 0 0 0 0 0 0 0 0
Byte S1 S0 EN-pE2 EN-pE1 EN-pE0 EN3 EN2 EN1
5 0 0 0 0 0 0 1 1
Byte EN_PhaseShift EN_probe_pll PLL_Locked_sc NA ERROR_LIMIT_SC<2> ERROR_LIMIT_SC<1> ERROR_LIMIT_SC<0> EN_LOCK_CONTROL
6 1 0 0 0 0 1 0 1
Byte phase_strobe<3> phase_strobe<2> phase_strobe<1> phase_strobe<0> phase_ck<3> phase_ck<2> phase_ck<1> phase_ck<0>
7 0 0 0 0 0 0 0 0
Byte bit_out<7> bit_out<6> bit_out<5> bit_out<4> bit_out<3> bit_out<2> bit_out<1> bit_out<0>
8 0 0 0 0 0 0 0 0
Byte bit_in<7> bit_in<6> bit_in<5> bit_in<4> bit_in<3> bit_in<2> bit_in<1> bit_in<0>
9 0 0 0 0 0 0 0 0
Byte bit_out<15> bit_out<14> bit_out<13> bit_out<12> bit_out<11> bit_out<10> bit_out<9> bit_out<8>
10 0 0 0 0 0 0 0 0
Byte bit_in<15> bit_in<14> bit_in<13> bit_in<12> bit_in<11> bit_in<10> bit_in<9> bit_in<8>
11 0 0 0 0 0 0 0 0
Byte bit_out<23> bit_out<22> bit_out<21> bit_out<20> bit_out<19> bit_out<18> bit_out<17> bit_out<16>
12 0 0 0 0 0 0 0 0
Byte bit_in<23> bit_in<22> bit_in<21> bit_in<20> bit_in<19> bit_in<18> bit_in<17> bit_in<16>
-- Current Config --
FPGA selection: FPGA 0
ASIC selection: ASIC 0
-- Register Selection --
Use half-wise registers
Use channel-wise registers
-- Config File Info --
not set
File Not Set
Load
Save
Save As
-- Link Info --
IP Address: 10.1.2.208
Port: 11000
Readback
Ping FPGA
Send Current ASIC Config
Send All Configs

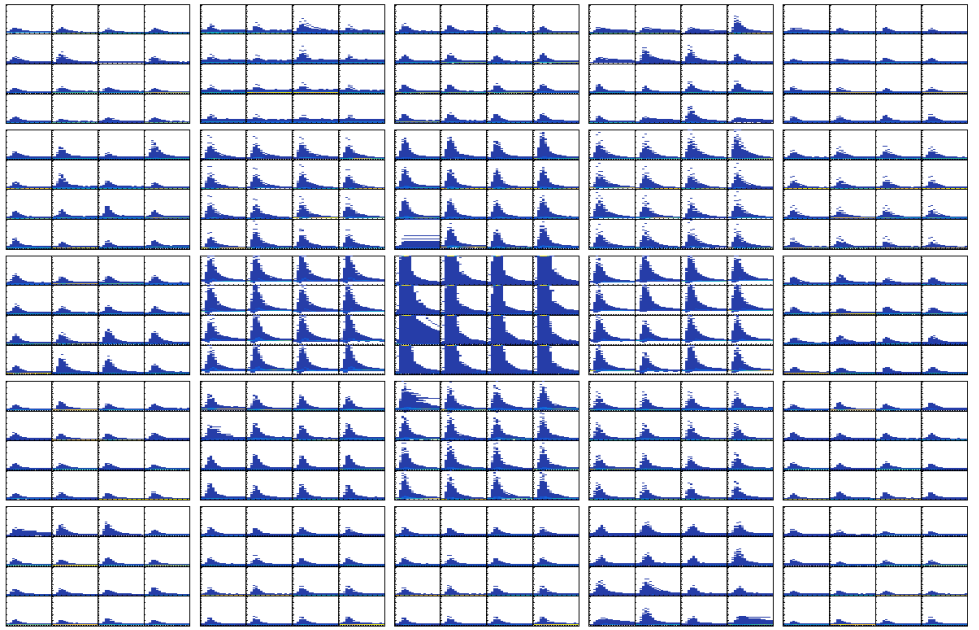
Running the DAQ

- Enable/disable bias from High Voltage tab
- Select external trigger, delay, and number of samples
- Three run modes
  - Number of events
  - Duration
  - Manual start/stop
- Can automatically start the online monitoring
  - Check the box and select the folder with the compiled code

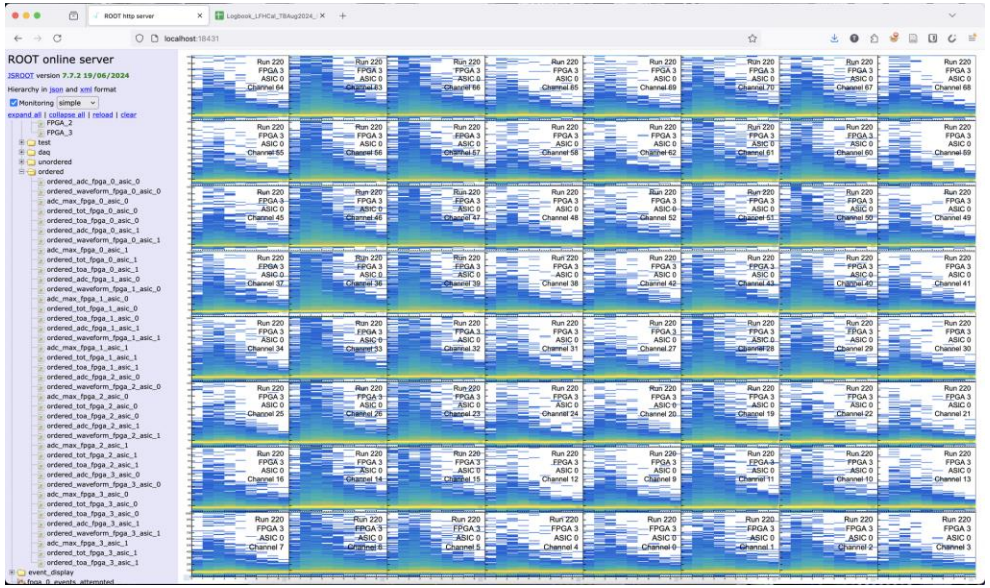


Online monitoring

- Real time decoding of events
- Implemented as webserver accessible from any browser
- Easy-ish to develop detector specific visualizations
- ADC, ToA, ToT spectra
  - Very helpful to make sure pedestals look correct
- Tracks events and packets from each ASIC
- Does **not** do event alignment
  - I could be convinced to add this though



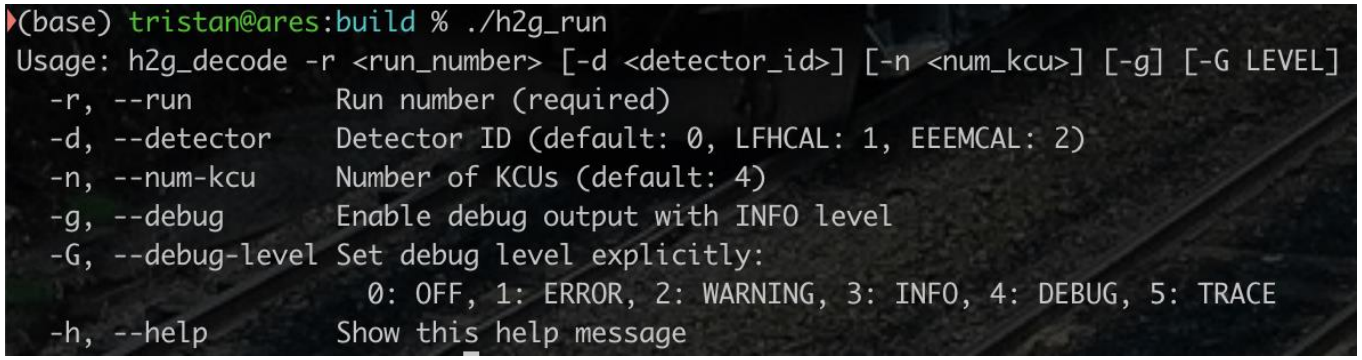
EEEMCal @ DESY II



LFHCal @ CERN PS

Reconstruction

- Output .h2g files can be decoded with the h2g_decode package
- Standalone h2g_run decoder creates a ROOT tree with all data
- Also creates a shared library not dependent on ROOT for integration with other software (i.e. the LFHCal test beam analysis suite)

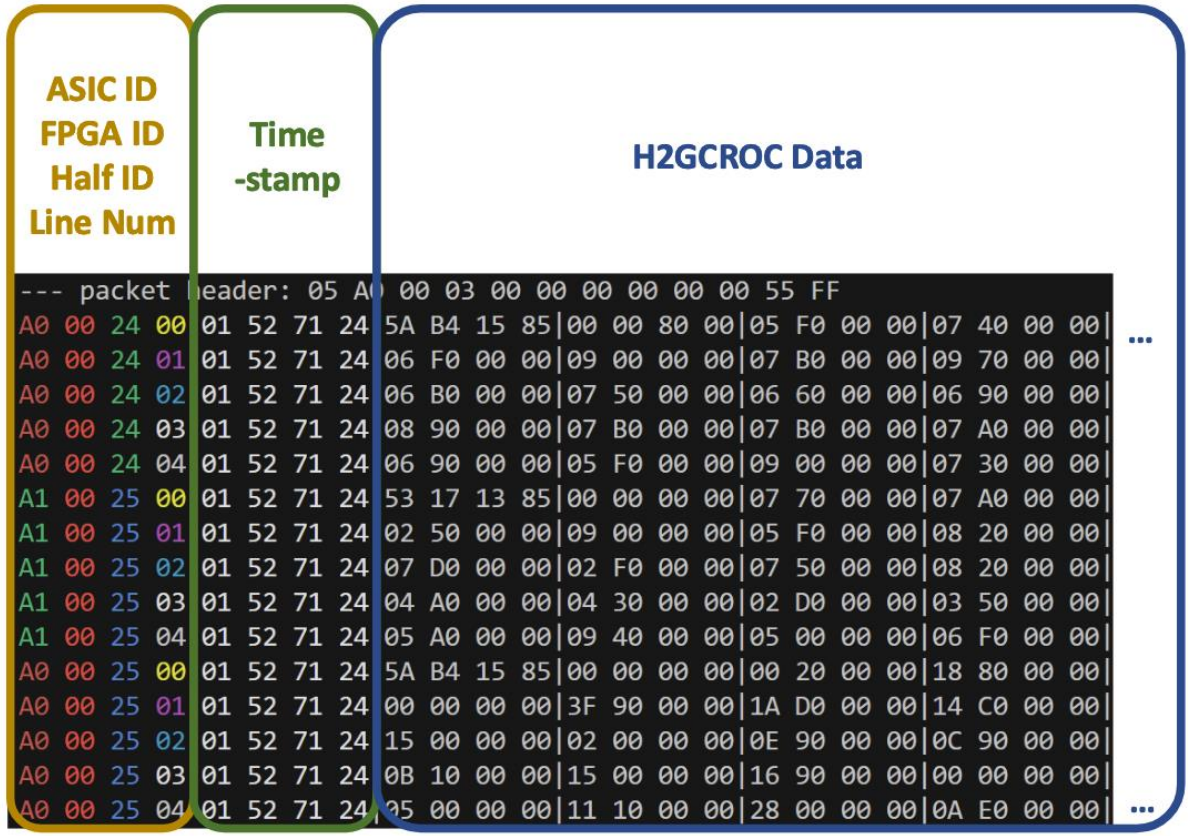


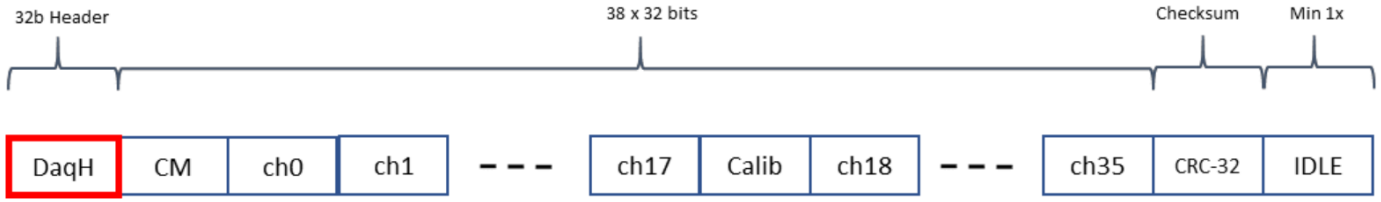
Data format

- Data sent from FPGAs to DAQ computer over UDP

- 1452 byte packets
  - 12 byte packet header
  - 36 “lines” of 40 bytes
    - *4 byte ID*
    - *4 byte timestamp*
    - *32 byte data*

- 5 lines encode a 36 channel sample
  - One half of a H2GROC
  - 160 bytes divided into 36 channels, common mode and calibration channels, a header, and a CRC check

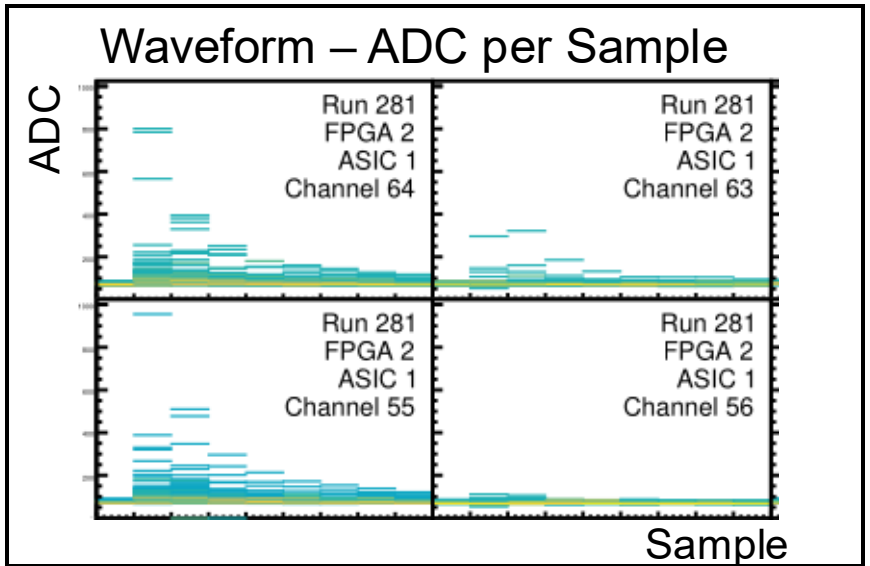


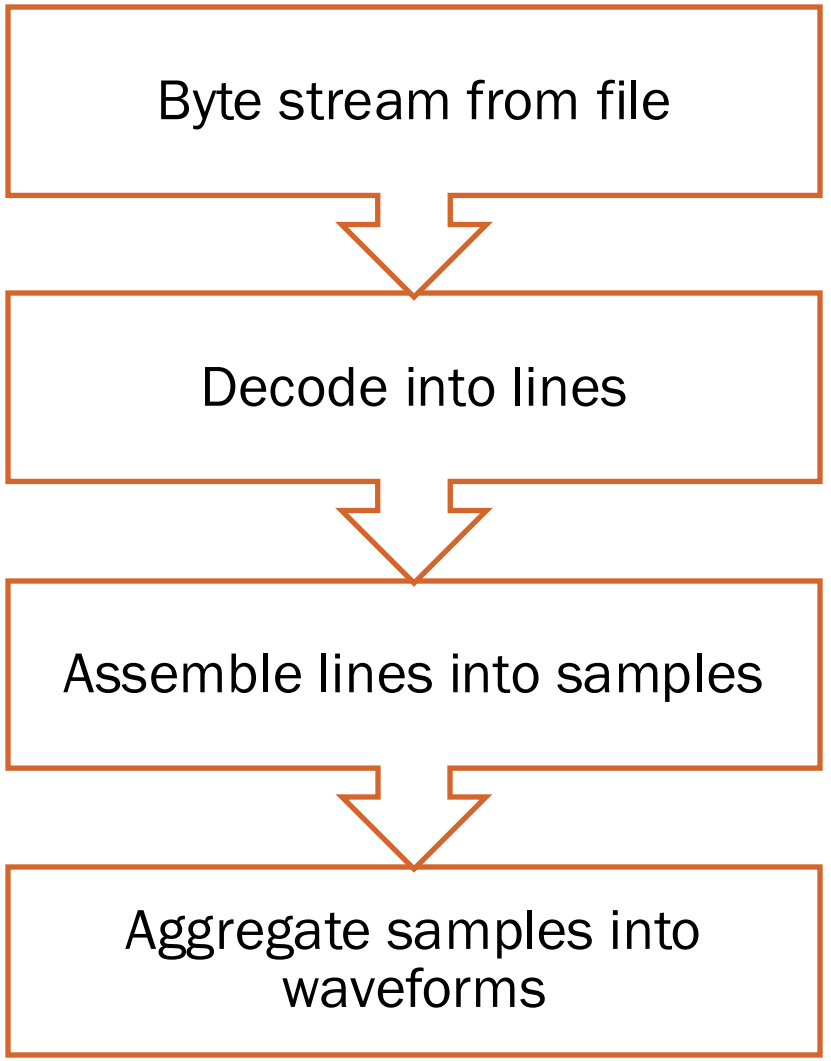


Waveform reconstruction

- Each FPGA is initially processed individually
  - Lines arranged into sample by FPGA timestamp
  - Does not require lines to arrive in order

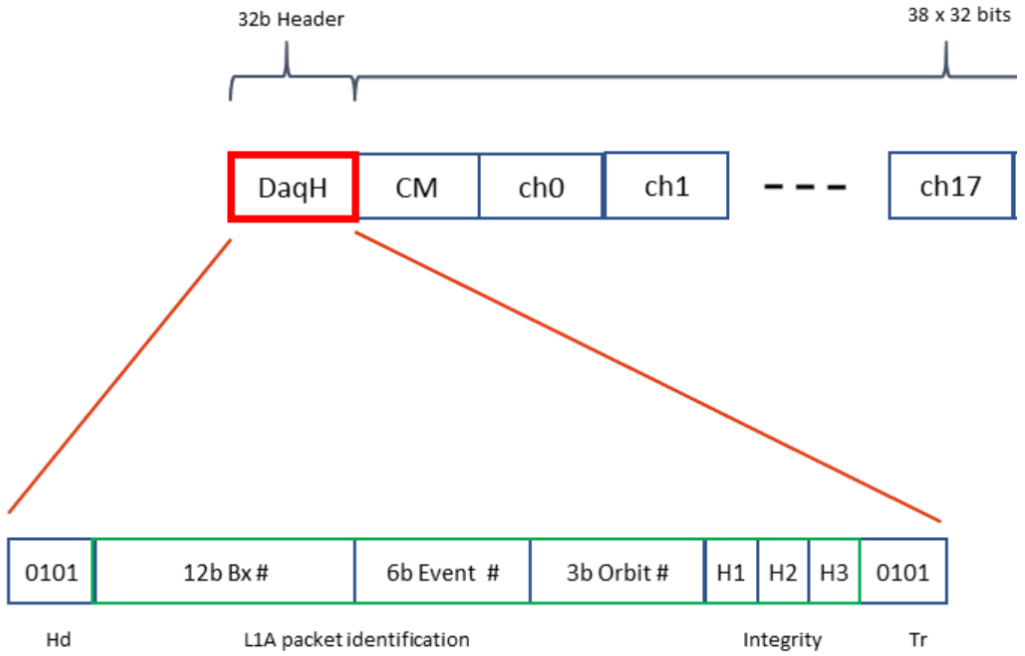
- Processed samples are grouped and aligned into waveforms
  - Samples per waveform configurable from DAQ
  - Relies on knowing timestamp spacing between samples (41 counts)
  - Does not require first sample to arrive first, can shuffle around





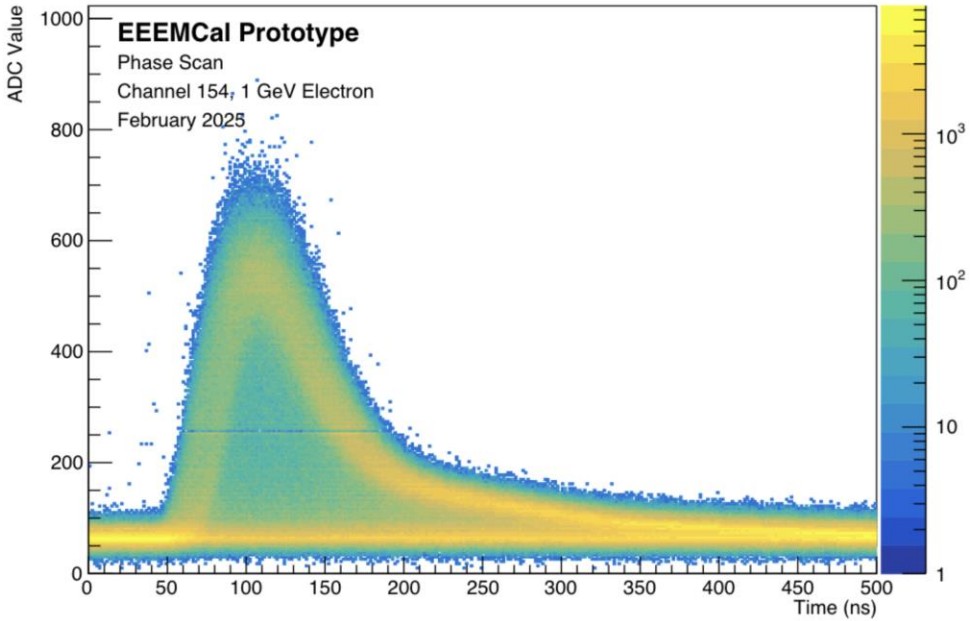
Event alignment

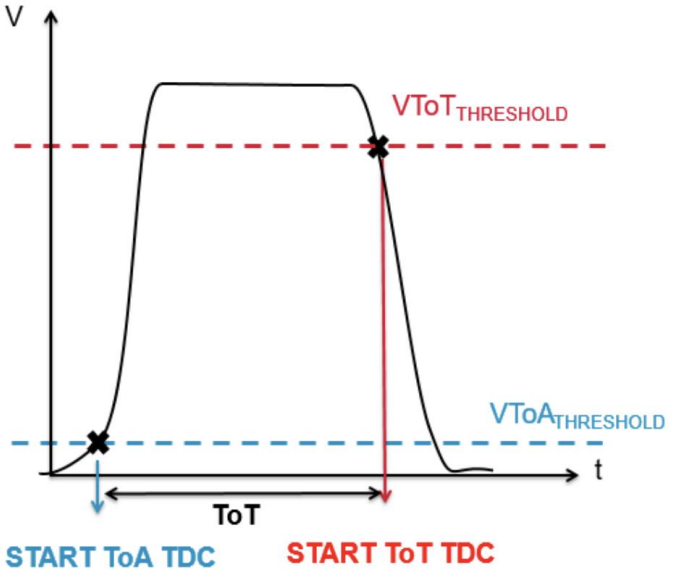
- Waveforms from multiple H2GCROC need to be aligned to process the full event
  - Within a FPGA: Waveform share a common timestamp
  - Alignment between FPGAs requires a different approach
- FPGA streams data to DAQ over UDP
  - Packets out of order, lost
- Each H2GCROC contains an event counter, bunch crossing counter, and orbit counter
- Multiple FPGAs are synchronized by keeping the gap between event numbers equivalent
  - Counters not yet reset synchronously
  - Allows incomplete events to be skipped without losing alignment between FPGAs



Signal extraction

- Two types of signals to process
- ADC
  - Higher sensitivity to low energy events
  - Samples signal every 25 ns
  - Signal from $\mathrm{ADC}_{max} - \mathrm{ADC}_{ped}$, waveform fit, any number of strategies
- ToA/ToT
  - Expands dynamic range by measuring the time the signal is over some threshold
  - One value per “waveform” – most samples are 0
  - If there is a ToT measurement, the ADC measurement is invalid

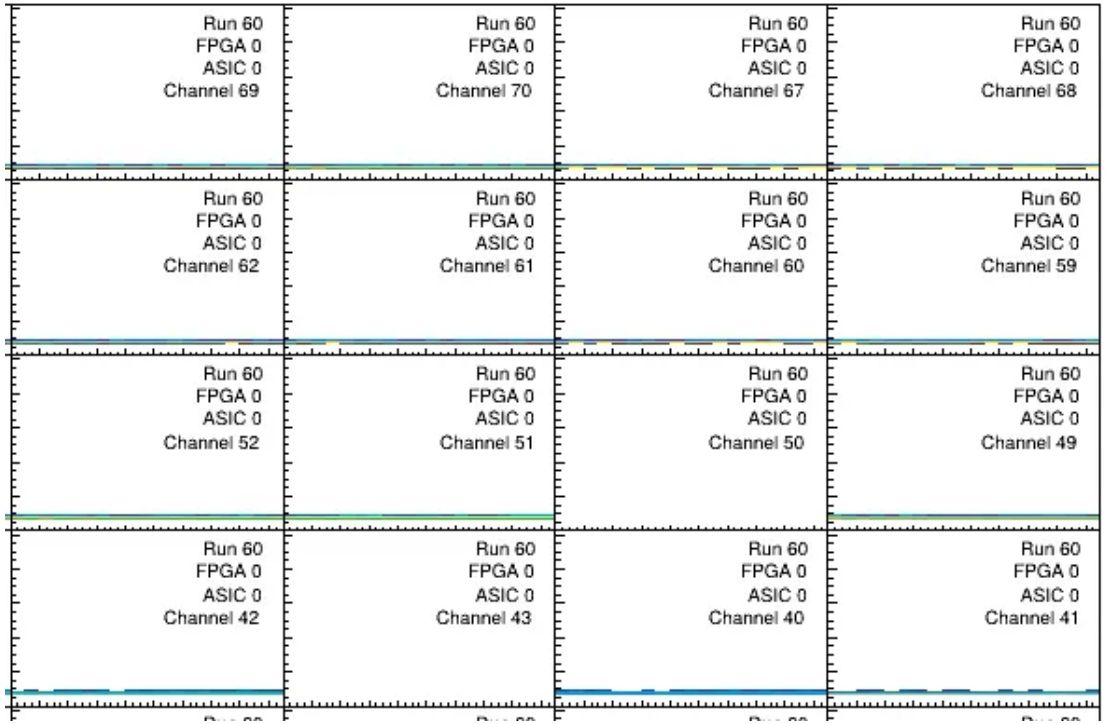




Test beam experience

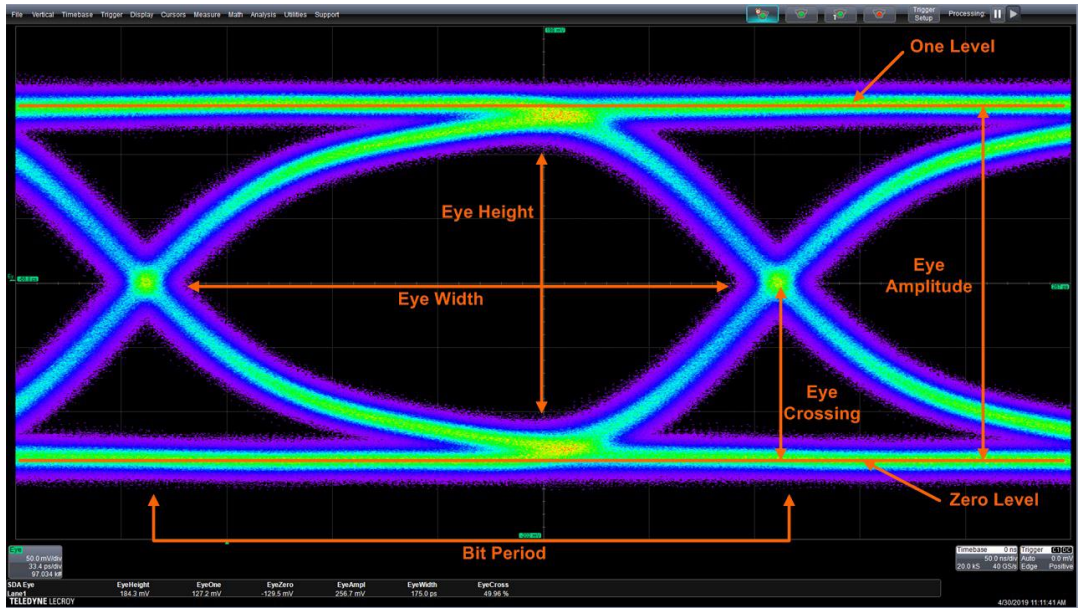
Bad pedestals

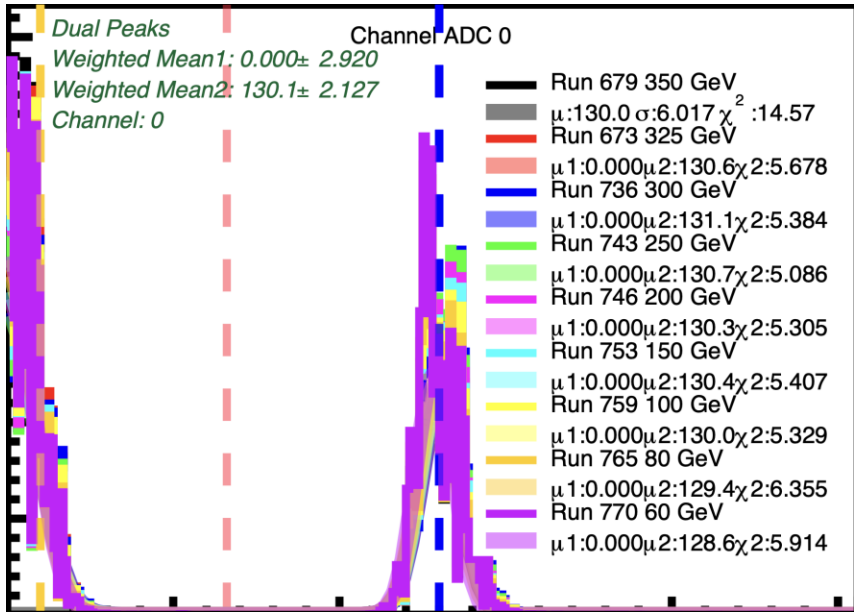
- Lots of trouble setting pedestals to desired values
- Scans seemed to struggle with stability and linearity
- Shihai has been working on improving the calibration routines, should be improved
- We ended up using the “run it a bunch of times and hope one looks good” approach more than I’d care to admit



Split pedestals/bit shifts

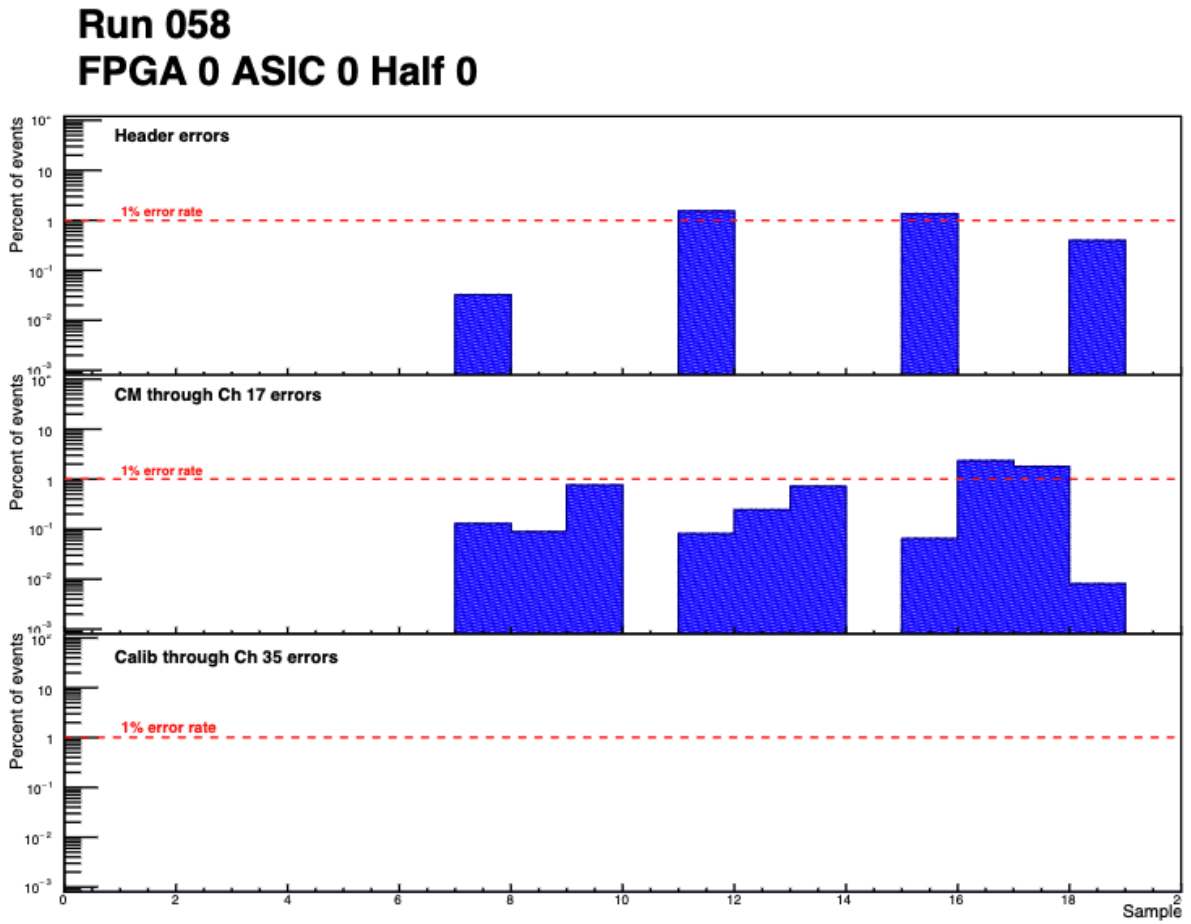
- Sometimes two pedestal values were observed in the ADC spectra
- Likely cause: bit shifts on the 1.28 Gbps links
  - If a marginal phase is selected during the IODelay scan, we may drift into reading during bit transitions
- Solution: Redo IODelay scan
- This will show up on the online monitoring, but *should* not appear in reconstructed events – header will be shifted as well, preventing event alignment





Hamming codes and CRC check

- Decoder checks for hamming errors, but doesn't yet do anything with them
  - Working on allowing events with incomplete waveforms
- Still working on the study, but the seem to start to appear with 1% regularity after ~7 samples at 100 Hz
- I cannot get the CRC check to work
  - Tried several standard implementations and polynomials, need to return to this



EEEMCal @ DESY II

Event alignment

- All that is keeping multiple KCU in sync is a common clock and receiving the same trigger
- Data from each is sent to DAQ over UDP – no guarantee of order, or even arrival
- There are a number of counters which should help us
  - KCU clock, bunch counter, orbit counter, event counter
- I have not found these to be totally reliable, and I don’t have great answers for why not
  - Memory errors in the header?

Event alignment strategy

- Starting with a single KCU
  - Aggregate all data for one sample for each half by KCU timestamp
  - Match halves by KCU timestamp
  - Use KCU timestamps (~20 ticks between samples) and event number (increments once per trigger) to align samples into a waveform
- Matching multiple KCU
  - Use deltas between event number of first sample to create most likely order
  - If we cannot get the event number within an acceptable delta (currently 1), skip events on KCUs to attempt to bring back into alignment
  - Seems to work reasonably on low rate (~< 100 Hz) data
  - Fails miserably at high rates (> 1 kHz)

Event alignment – what would help?

- Ability to synchronously send fast commands
  - If the bunch and orbit counters could be synchronously reset, they should trivialize alignment
- Fewer samples are probably better
  - This was not intended to be a streaming readout
- KCU timestamps are inconvenient at CERN
  - 30 bit counter @ 40 MHz ~= 27 seconds rollover
  - 45 second period for injector complex, some ambiguity about if the counter rolled over or not
  - Timestamps aren't started at the same time, don't roll over at the same time
  - Added a 1 Hz heartbeat packet to help with this, but I haven't had time to make full use of it
- Dealing with bit slips better
  - I haven't had time to fully test this, but I think better handling of bit slips could help significantly
  - Some of the assumptions I'm making totally break if the wrong event numbers are decoded

Questions/future

- Are we using the same DAQ for the 2025 CERN test beams?
  - I know Martin has been adding support to RCDAQ
- I heard the H2GROC3b variant had know issues with memory transfers – are these resolved?
- What are the priorities for the 2024 data?
  - Still working on tying in with LFHCal test beam analysis software – nothing impossible, just taking time
  - I think lots of progress can be made here this week