DEEP UNDERGROUND NEUTRINO EXPERIMENT

Brookhaven National Laboratory

U.S. DEPARTMENT OF ENERGY

WIB Firmware Development for DAT ColdADC QC

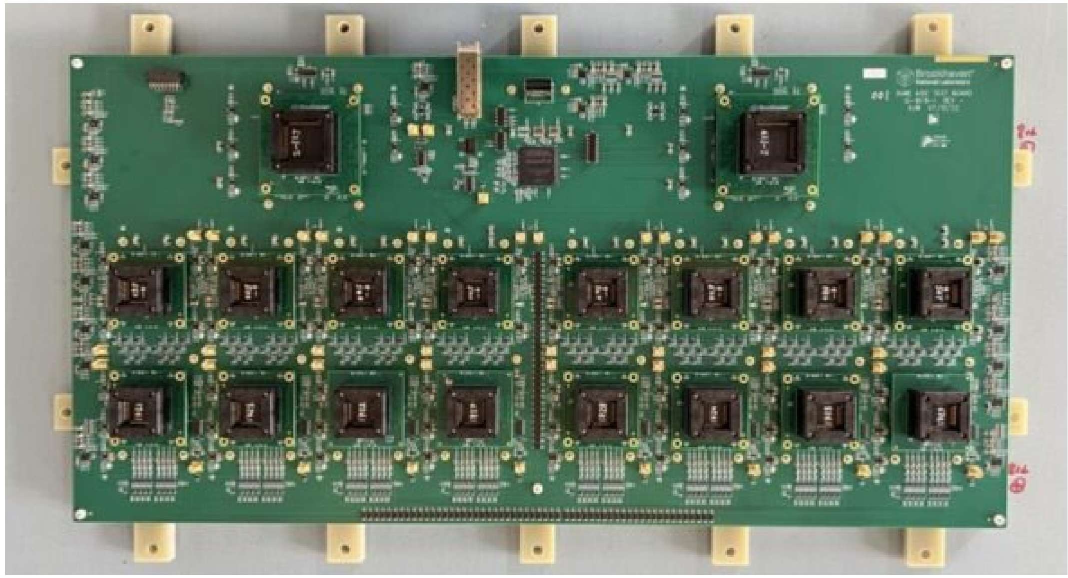
Jillian Donohue on behalf of the BNL CE group

12/19/2023

@BrookhavenLab

Outline

- ColdADC QC requirements
- Averaging firmware
- Histogram firmware
- Current to-do's



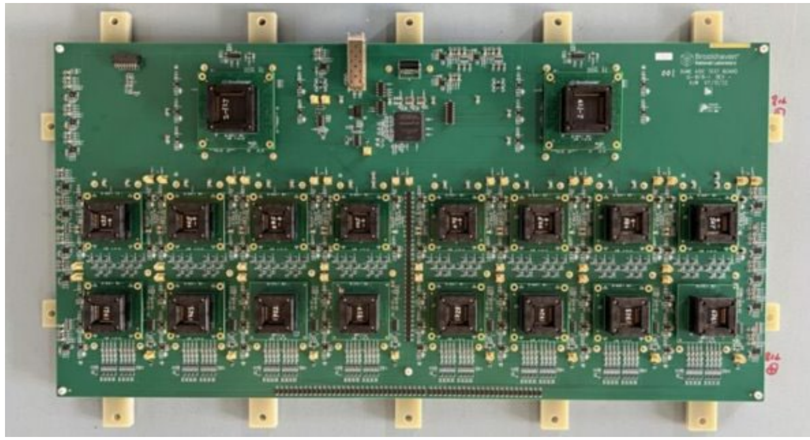
ColdADC QC requirements

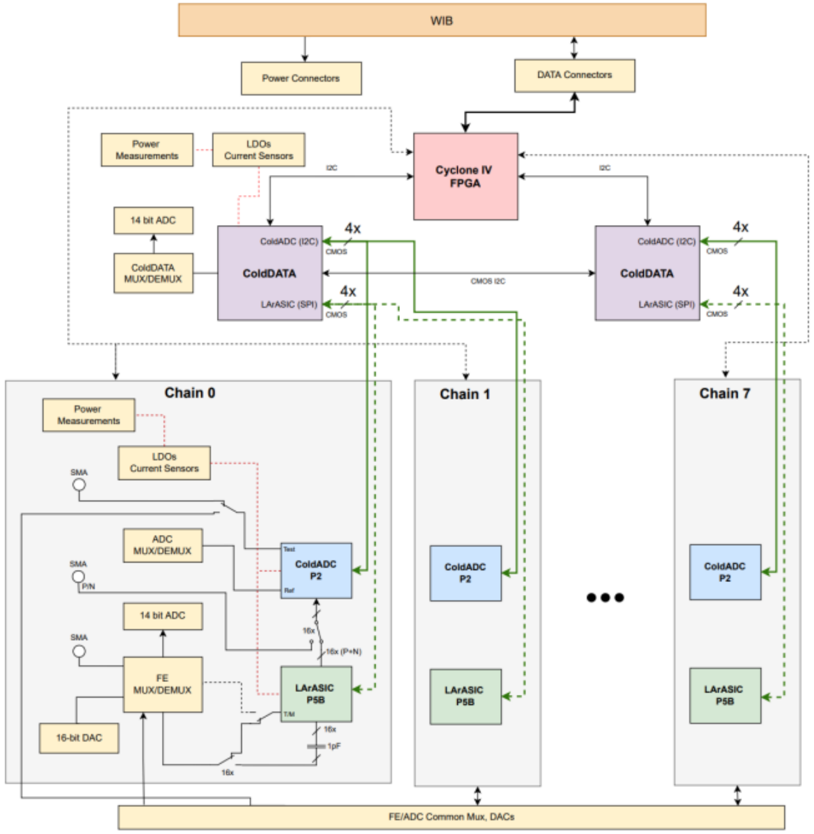
- Requirements:
  - Sampling frequency ~2 MHz
  - Number of ADC bits: ≥ 12
  - Total power (of all ASICs): < 50 mW/channel
  - Noise contribution ≪ 1000 e- (negligible compared to LArASIC)

- Specifications:
  - Crosstalk: < 1%
  - Differential Nonlinearity (DNL): Absolute value < 1
  - Integral Nonlinearity (INL): < 1(12-bit ADC unit)
  - Equivalent Number of Bits (ENOB) > 10.3
  - Overflow protection: When input signal exceeds the upper or lower ADC range, the output should be fixed at the maximum or minimum value.

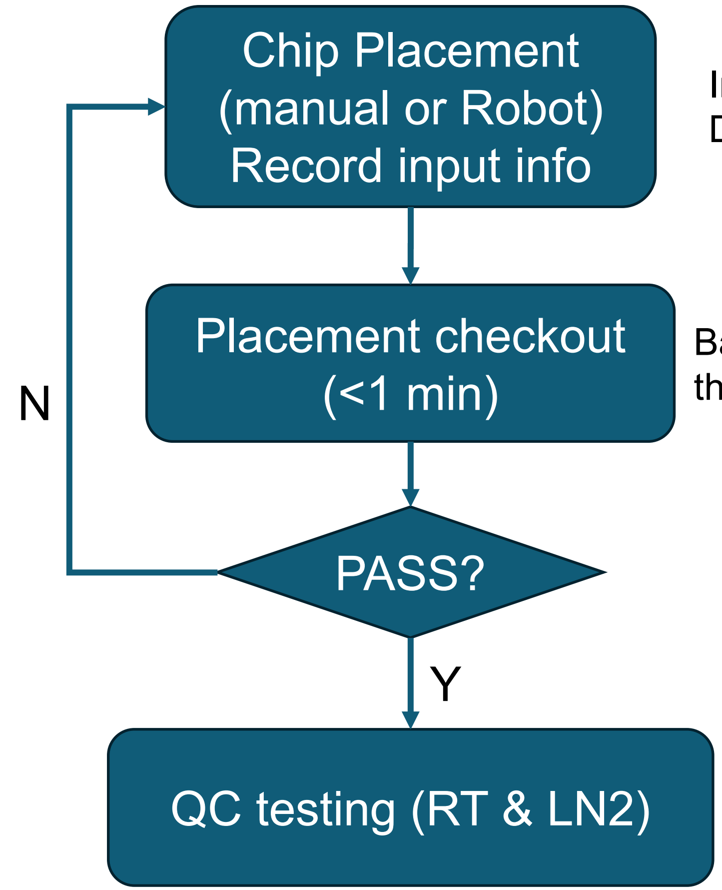
DAT overview

- Tests 1 FEMB's worth of ASICs
  - 2 COLDATA ASICs
  - 8 COLDADC ASICs
  - 8 LArASICs
- Appears like an FEMB to the WIB
- Monitors all ASIC power
- Tests all ASIC analog & digital IO





ColdADC QC Procedure and Test Items based on DAT



Input info: Date, Time, Location, Tester, DAT SN, Socket SN, Chip ID, and etc.

Basic functionality checkout to assure any chip is aligned in the socket, no short, no open pin

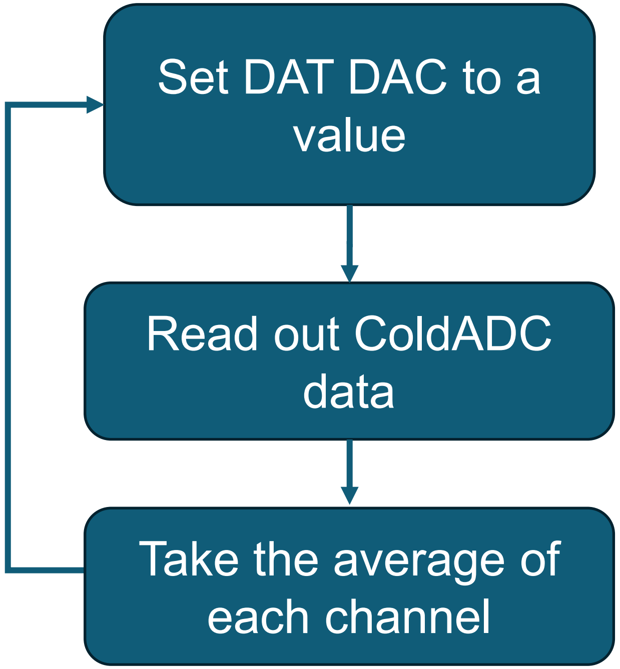
- Power cycle at least 5 times with different configs
- Test I2C communication (Write/read registers)
- Check power-on reset (verify default register values)
- Measure reference voltages
- Measure ring oscillator frequency
- Perform auto calibration; check weights
- Measure open channel noise (new DAT)
- **Measure DNL/INL (slow ramp or sine wave input)**
- Check overflow protection
- Measure ENOB (optional, performance to be evaluated)
- Crosstalk (optional, performance to be evaluated)

Verified doable on DAT

Brookhaven National Laboratory DUNE

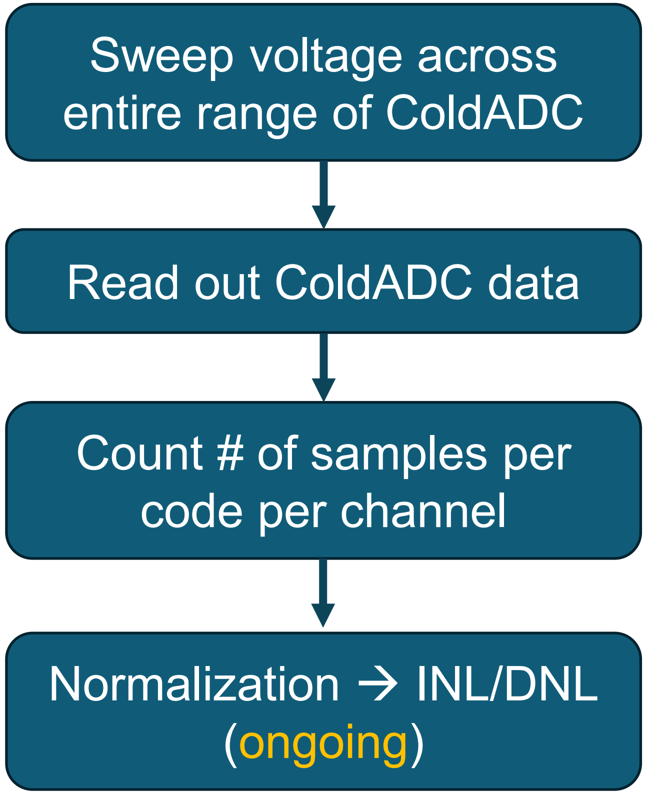
Averager

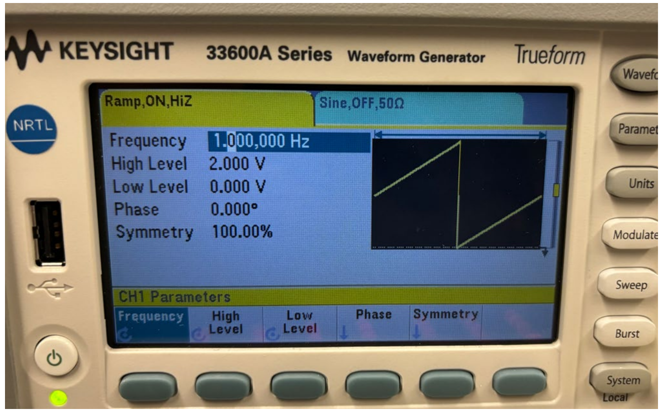
- Informs us on the ColdADC's DC offset, input range, gain, DC noise, & overflow protection
- Inject a known DC voltage via an external DAC into the ColdADC channels to see what code it outputs
  - Need to take an average because noise will be present
- The DAT DACs are capable of producing 2^16 different voltage levels between 0V and 2.5V (reference voltage)
  - To minimize time consumption, sweep DAC range in every 64 codes (2^10 total codes)

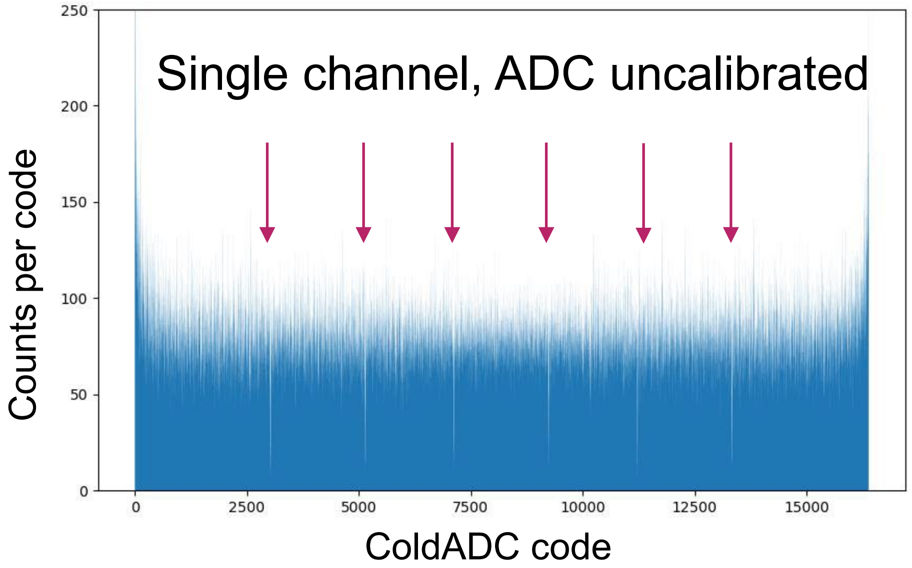


Histogram – INL/DNL

- Looking for INL/DNL, missing codes
- Want ~100 samples/code → ≥1,638,400 continuous samples (about 1 second)
- Sweep with a ramp (or a sine wave) from external waveform generator
  - Sampling rate of 2MHz & 100 samples/code → sweep period of 1 second







Why modify WIB firmware?

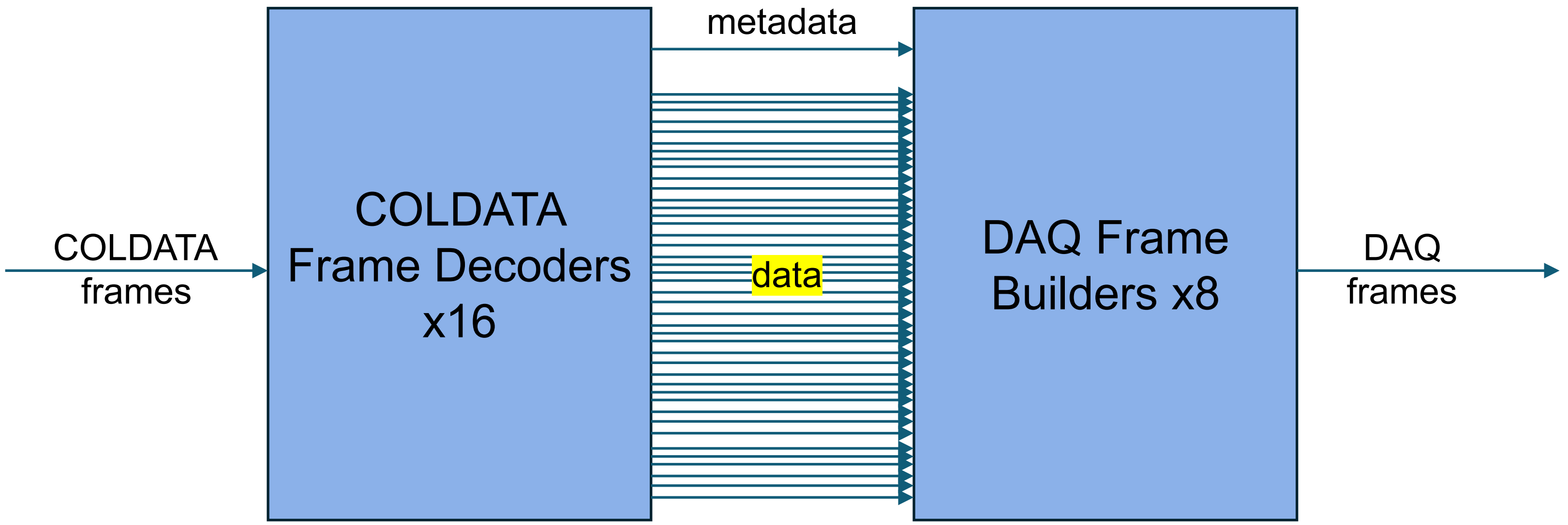
- Linearity test (slow ramp) requires 1,638,400 continuous samples
  - Spy buffer has ~2000 sample limit per channel
- Speed up averager test
  - Non-firmware implementation would require minimum 1024 spy buffer acquisitions

How to access data besides the spy buffer?

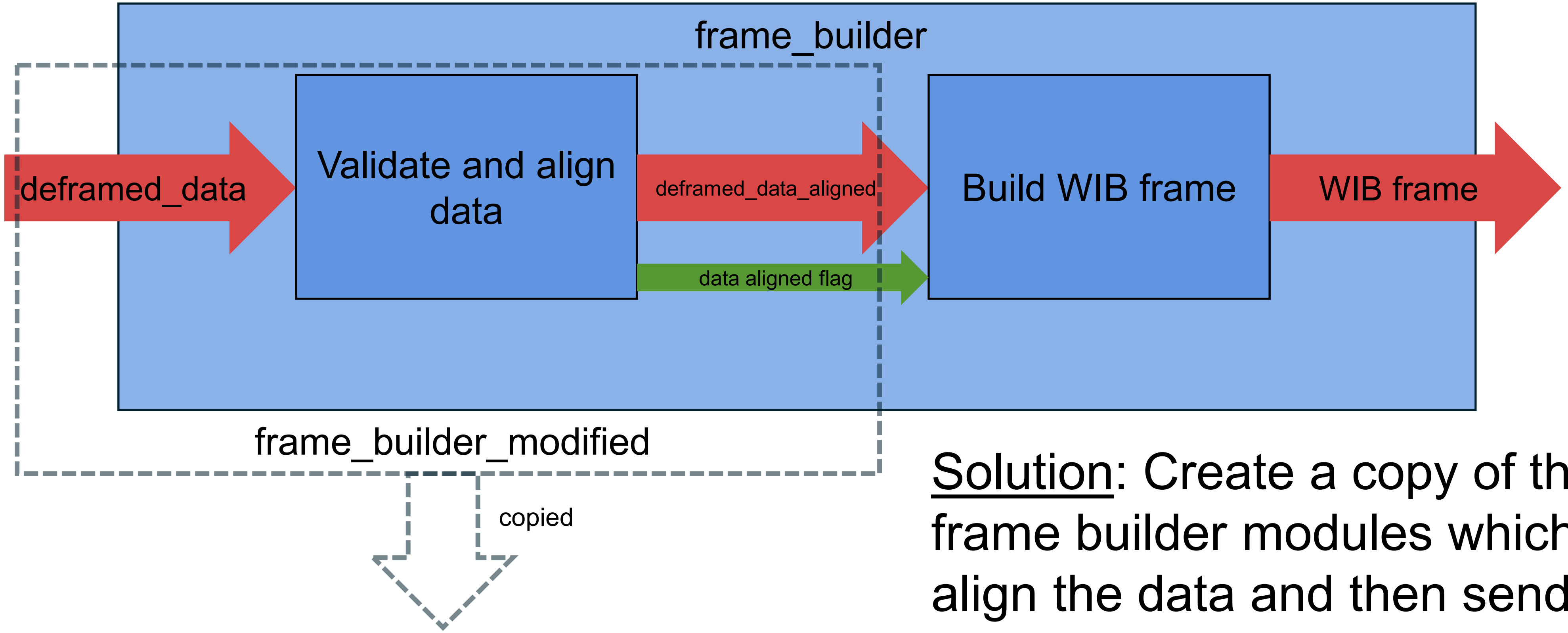
In the firmware, the COLDATA frame decoders extracts channel data and sends them in parallel to the DAQ frame builders, which validate the data and build a WIB frame around them to store in the spy buffers.

The frame builders perform a crucial step in validating & aligning the data so that each sample is only counted once and garbage data isn't counted.

But how to access live data & validate the data without modifying existing firmware modules?

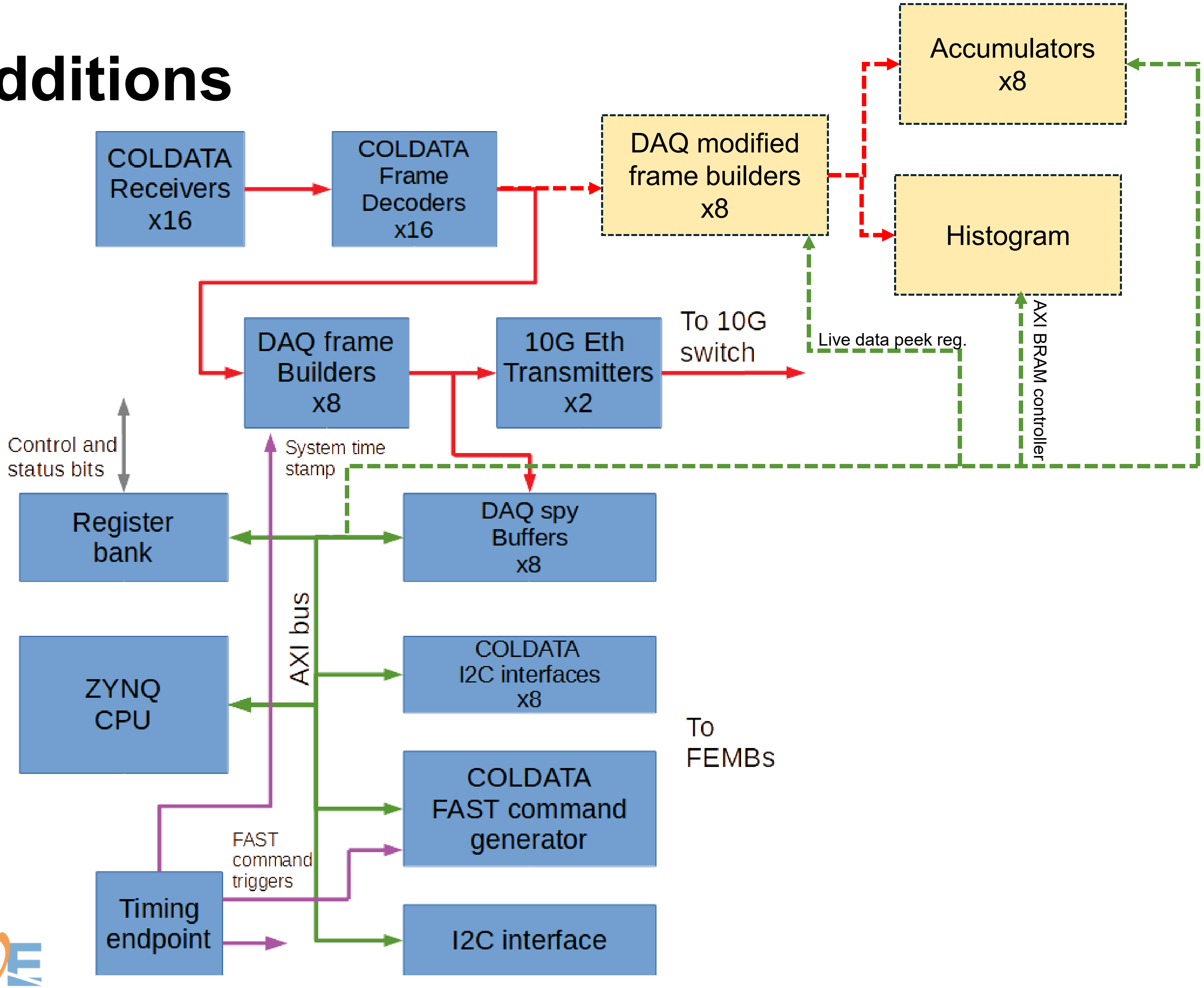


Modified frame builder to access raw streaming data

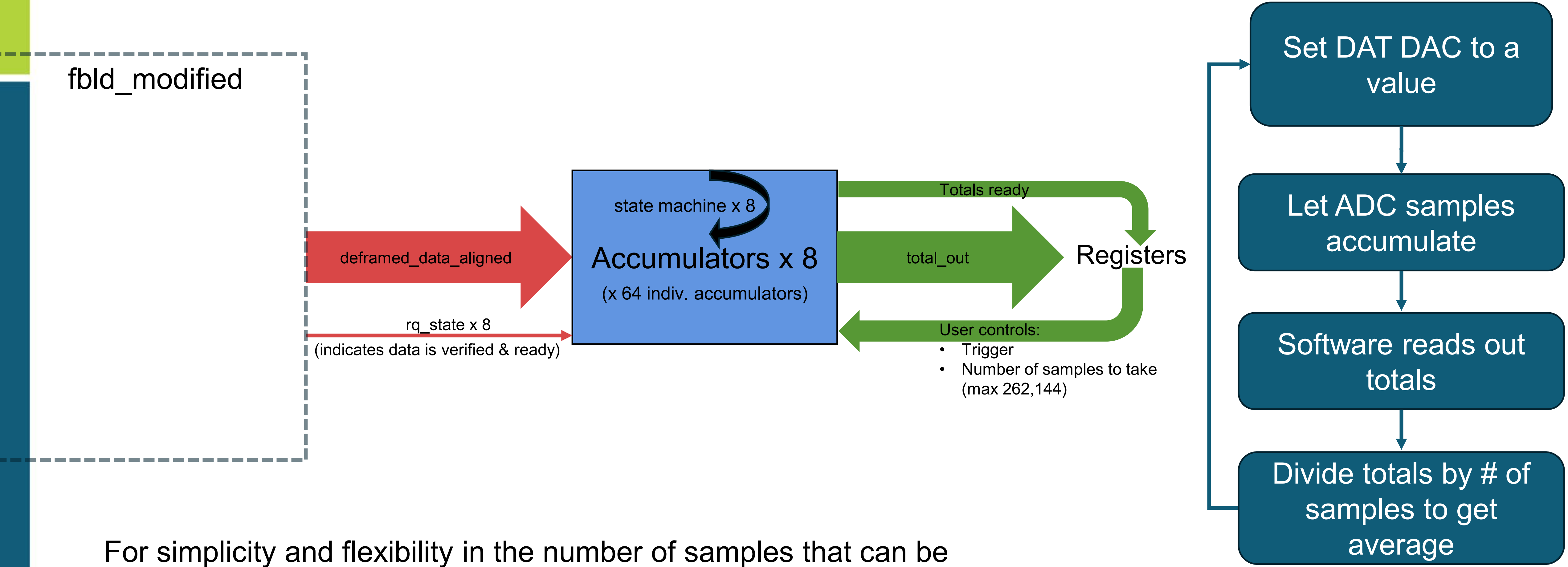


Solution: Create a copy of the DAQ frame builder modules which validate & align the data and then send the data out to our new firmware modules.

Firmware additions

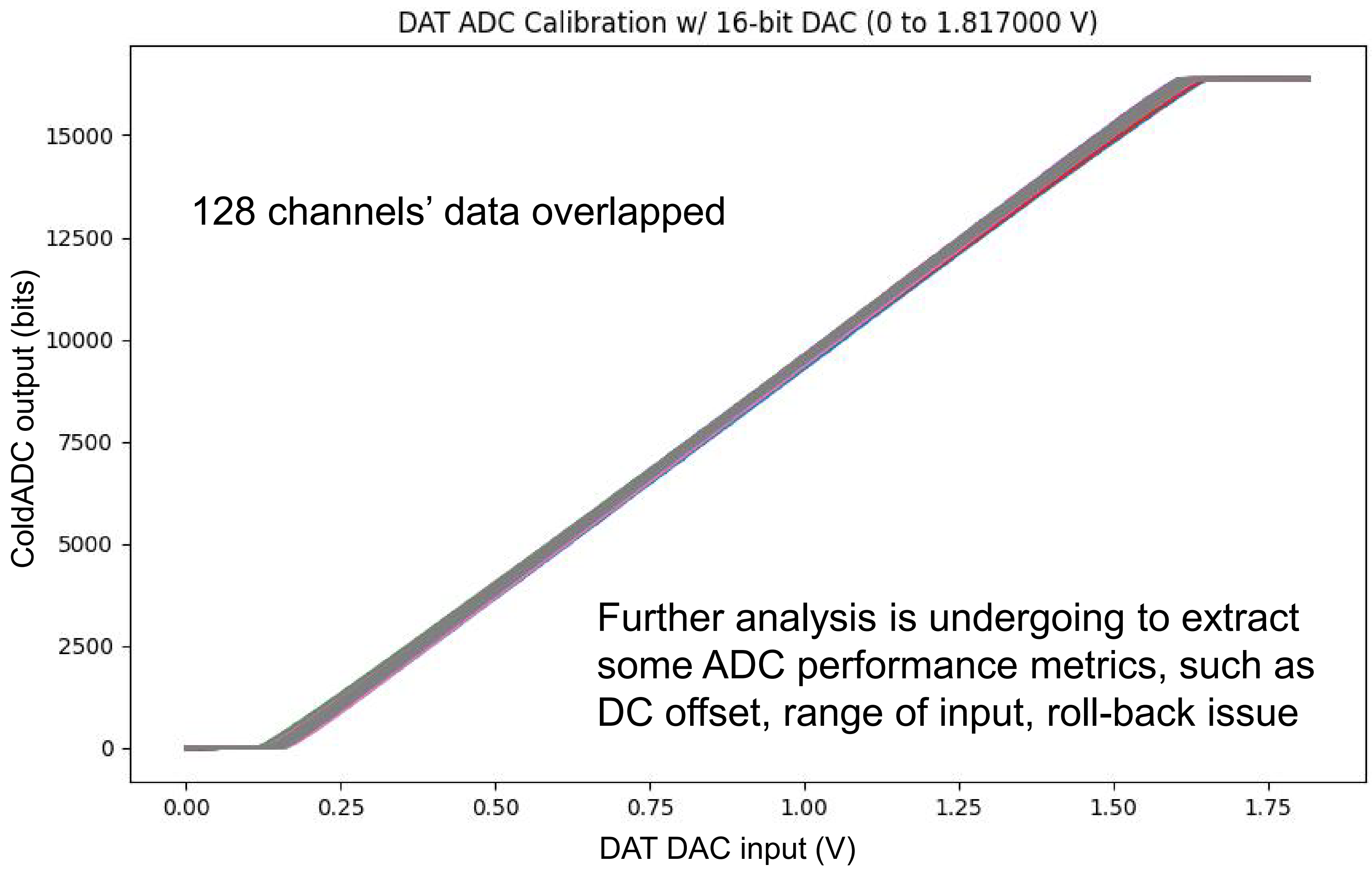


Accumulator (Averager) firmware module

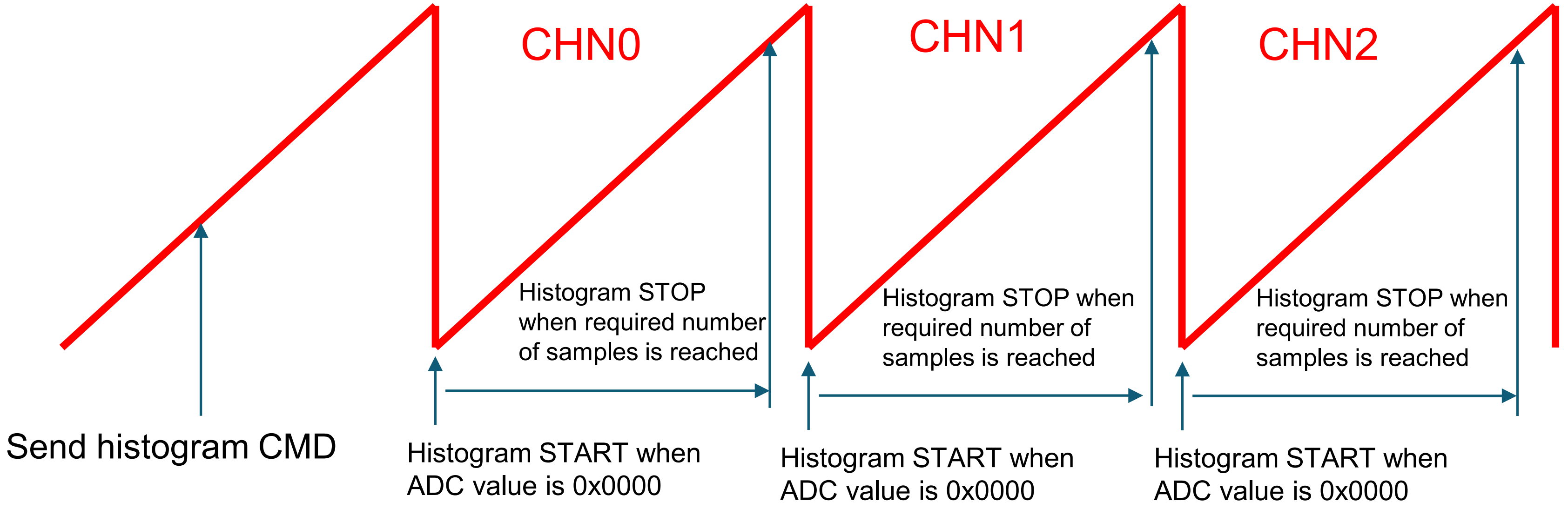


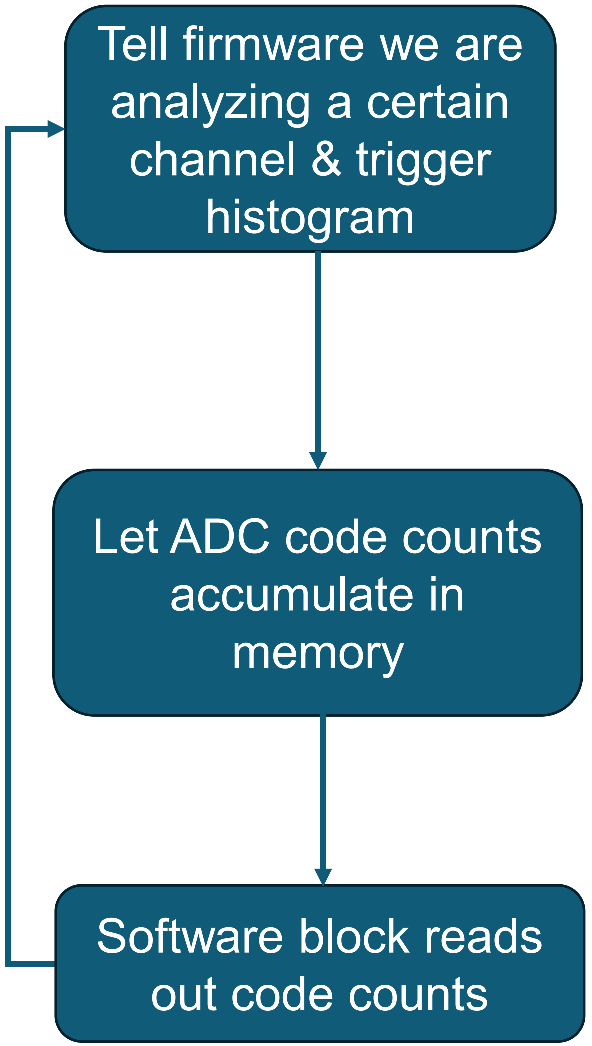
For simplicity and flexibility in the number of samples that can be specified, the firmware only accumulates samples into totals and leaves it to software to divide them into proper averages.

Accumulator results



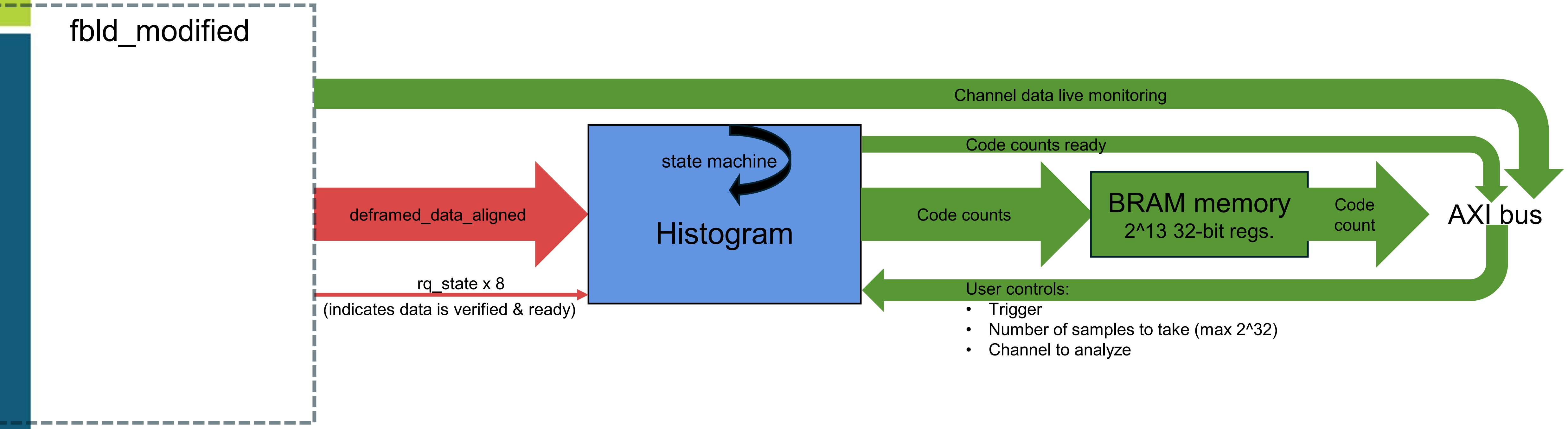
Histogram firmware module





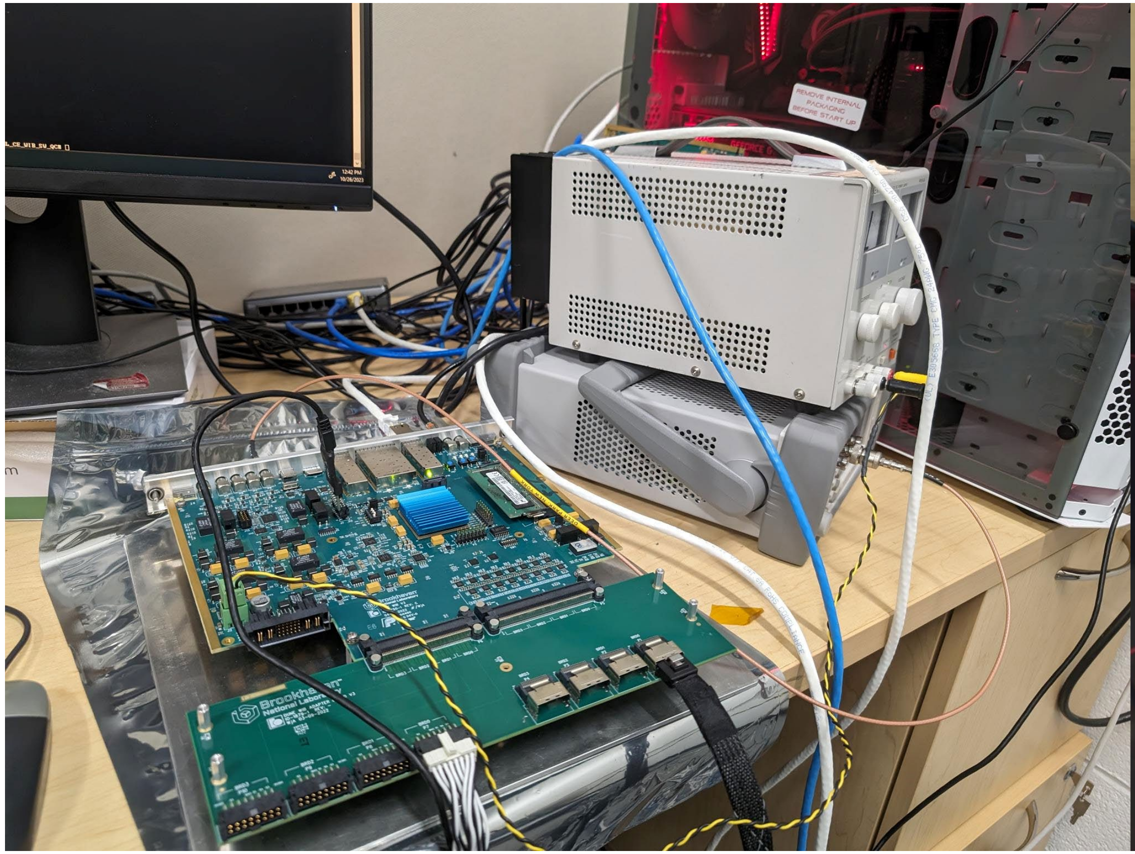
Time consumption of histogram study mainly depends on the period of ramp

Histogram firmware module

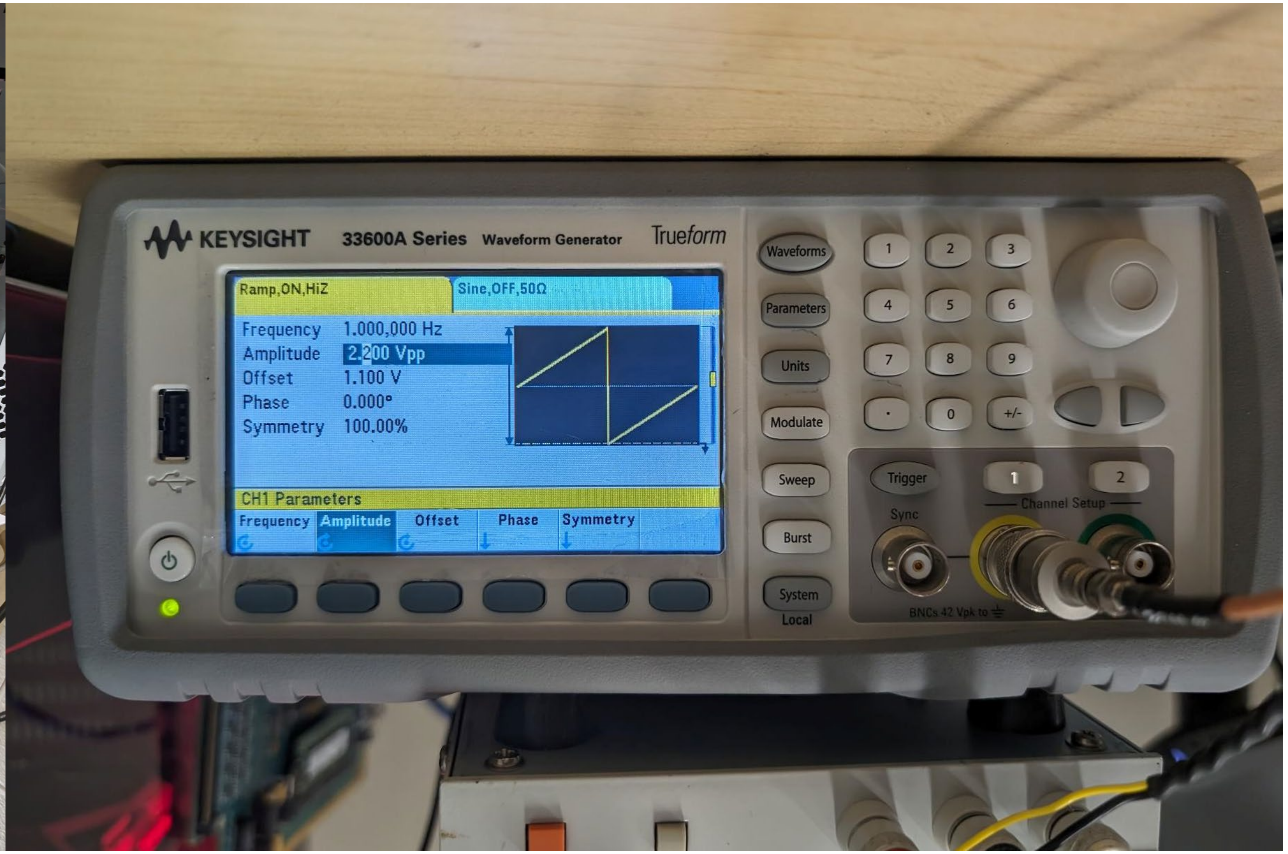


BRAM is AXI-mapped at 0xA00C8000 to 0xA00CFFFF

Histogram hardware setup



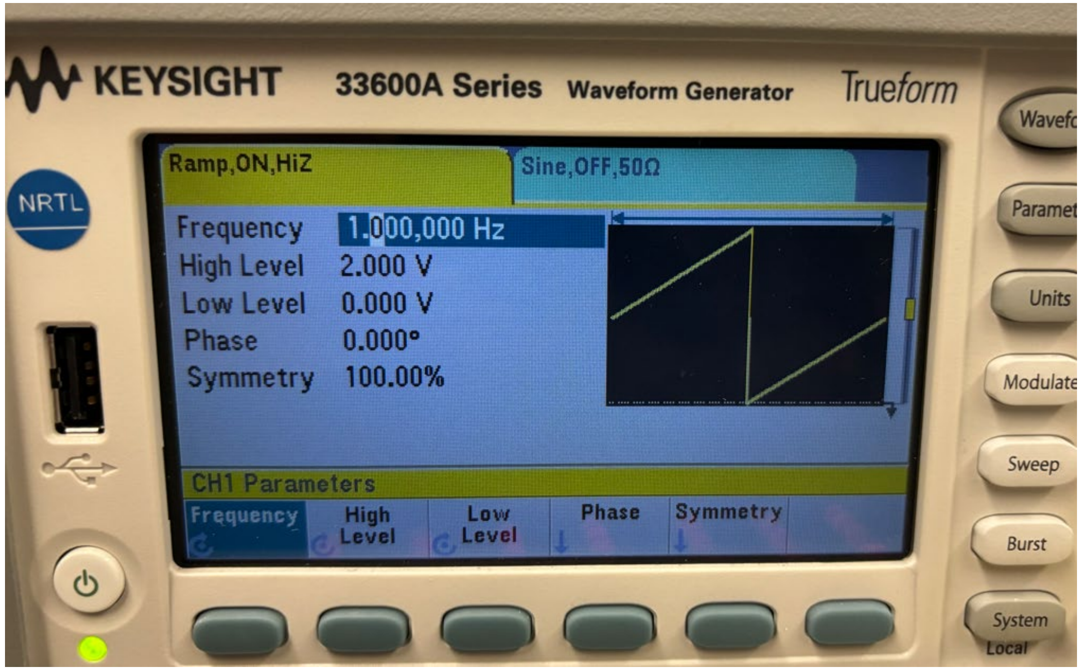
1. Connect a BNC-LEMO connector from one of the signal generator's output channels to P8 on the WIB.



2. Configure a ramp waveform with a period of >1 second from a voltage below the ADCs' range to above their range.

I am using a Keysight 33600A waveform generator.

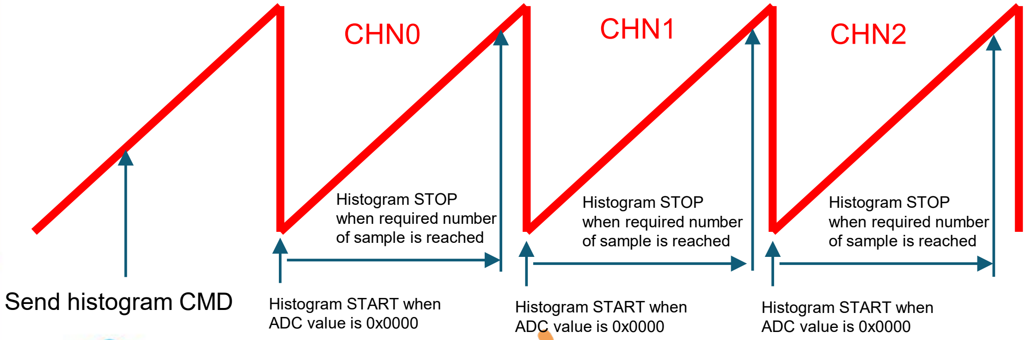
Realtime Histogram Testing and Analysis




```
root@dune-wib:~/BNL_CE_WIB_SW_QC# python3 adc_hist_buf.py 1800000
Linux
WIB histogram test
before running this script: configure the FEMB chips and trigger
ch0 1.65693379
ch1 2.65656448
ch2 3.6566005
ch3 4.65654531
ch4 5.65654928
ch5 6.65654466
ch6 7.65647714
ch7 8.65652029
```

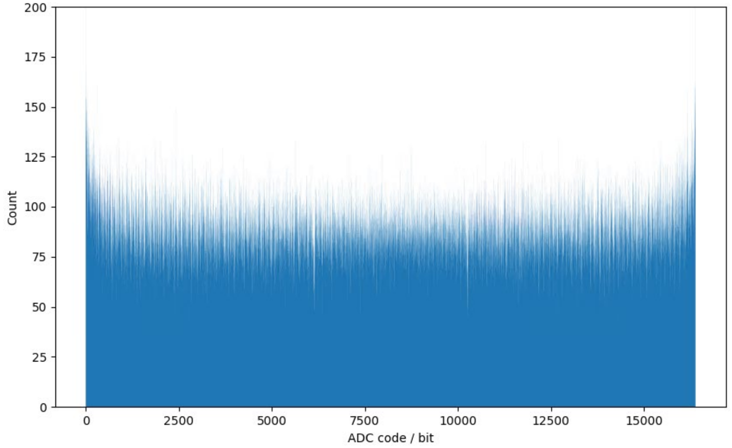
Extra <1s need for find T0 for Ch0

```
ch116 117.65470559
ch117 118.65470591
ch118 119.6546739
ch119 120.6546698
ch120 121.65466333
ch121 122.65456918
ch122 123.65465157
ch123 124.65458828
ch124 125.65456213
ch125 126.65454262
ch126 127.65451939
ch127 128.65455713
root@dune-wib:~/BNL_CE_WIB_SW_QC#
```

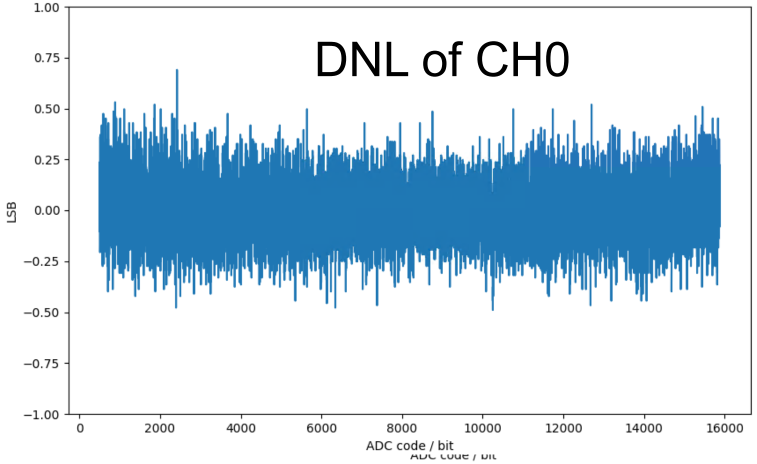


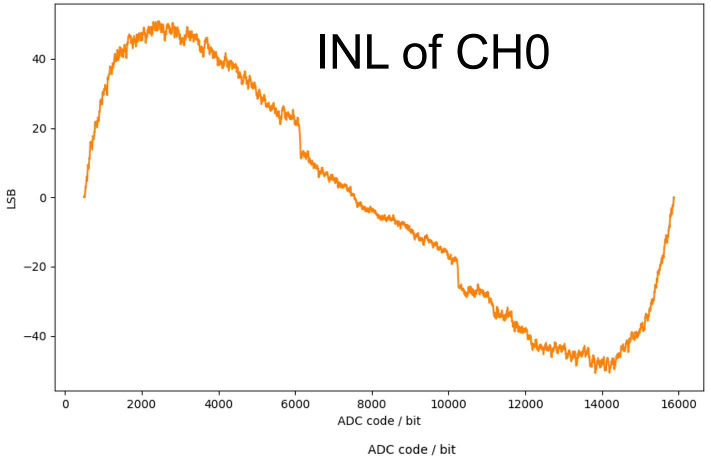
The time consumption ONLY depends on the period of the RAMP.

Normalization (Preliminary, ColdADC calibrated)

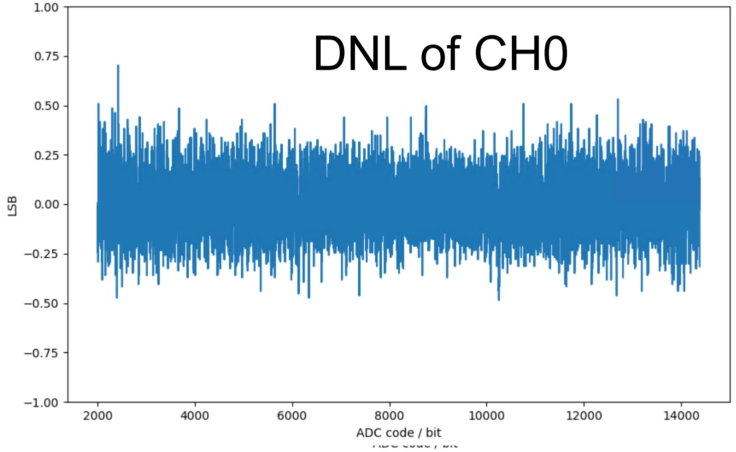


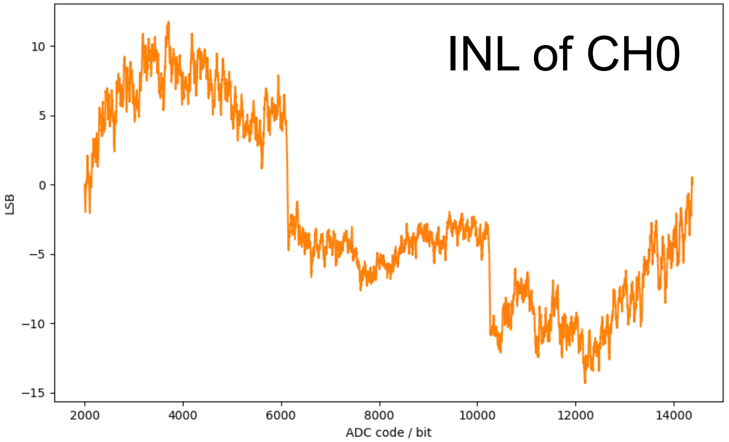
Histogram result of CH0





ADC code [500,15884]

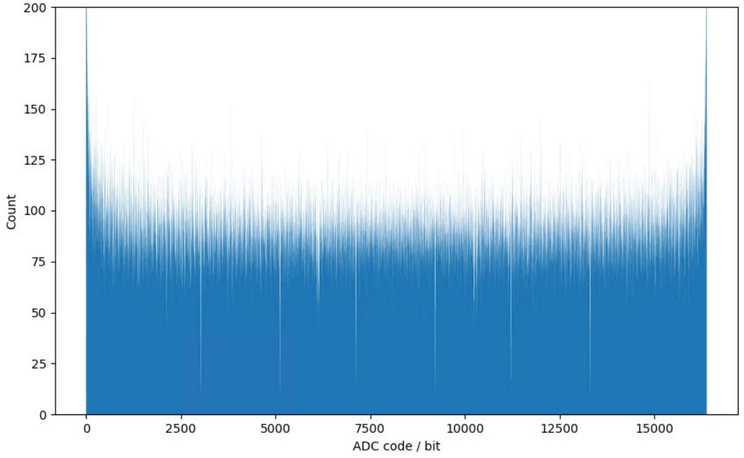




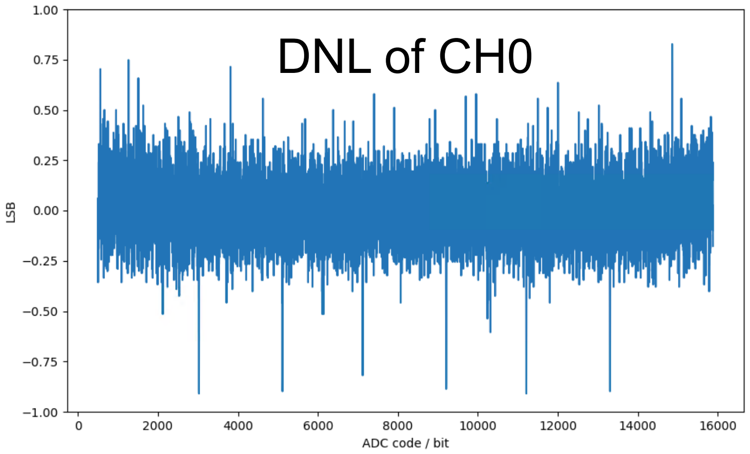
ADC code [2000,14384]

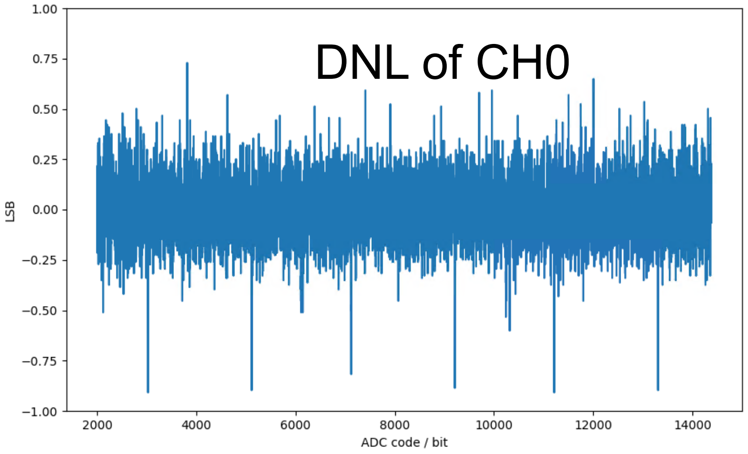
Note: Ramp signal is applied to all 128 channels simultaneously.

Normalization (Preliminary, ColdADC uncalibrated)

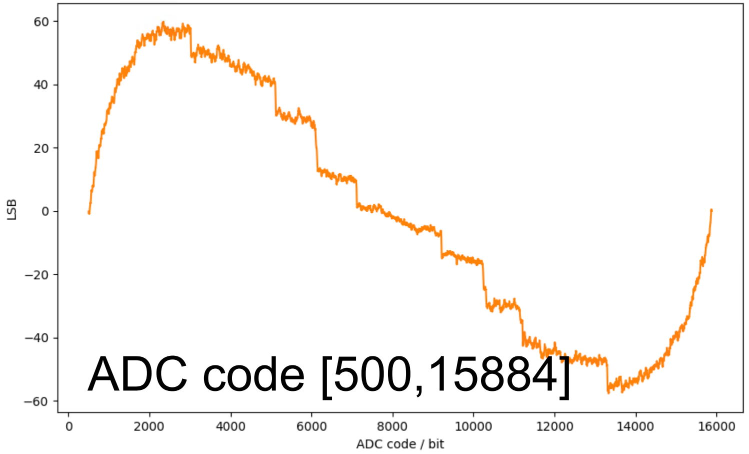


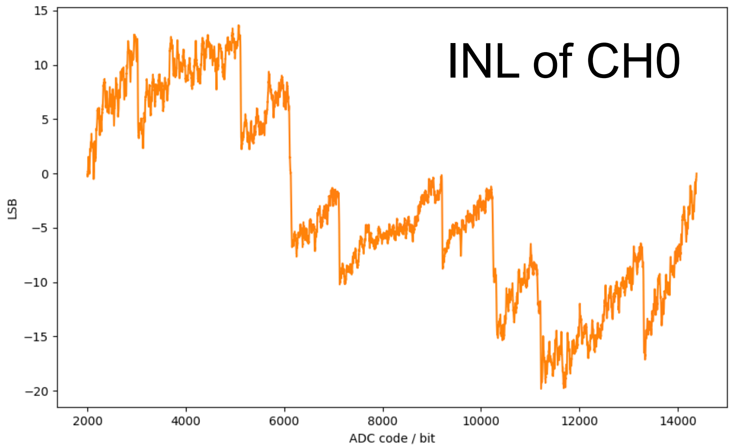
Histogram result of CH0





INL of CH0





ADC code [2000,14384]

Note: Ramp signal is applied to all 128 channels simultaneously.

WIB registers used – config register bank

Address, hex	Bits in register	Parameter name	Description
0xA00C0070	28:10	accum_num_samples	Number of samples to accumulate
0xA00C0070	9:1	accum_total_ch_sel	Accumulator channel readout selector. Channel X's accumulated total will appear in the register [totals] if you write X to this register.
0xA00C0070	0	accum_trig	Triggers accumulators to begin
0xA00C0074	0	hist_trig	Triggers histogram to begin
0xA00C0078	8:0	hist_ch	Channel to take histogram data for
0xA00C007C	31:0	hist_num_samples	Number of samples to count for histogram

Latest production firmware uses config registers up to 0xA00C0044

WIB registers used – status register bank

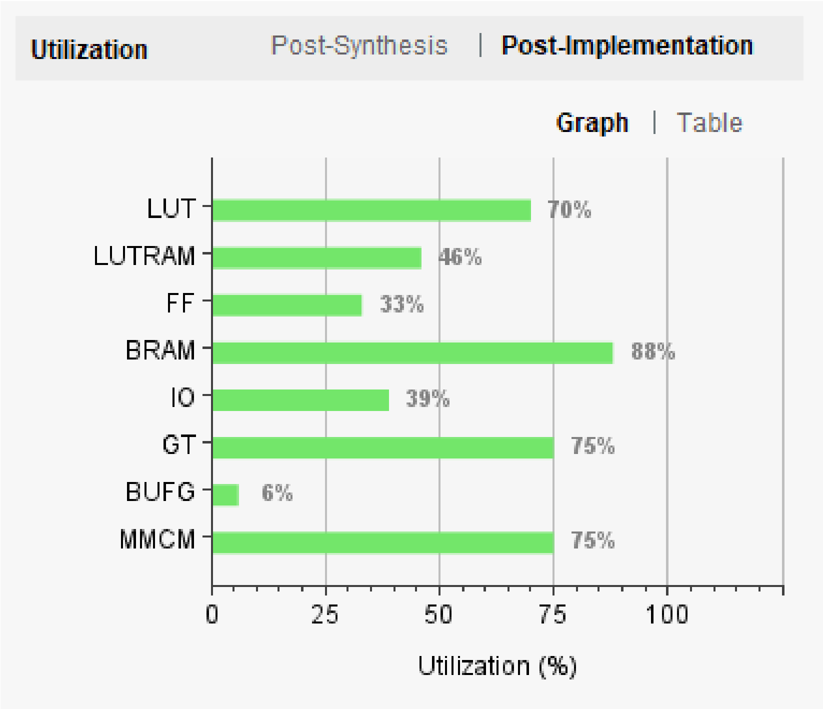
Address, hex	Bits in register	Parameter name	Description
0xA00C00F0	23:10	deframed_data_mon	Allows software to “peek” into the live channel data to aid the histogram test (hist_ch determines which channel)
0xA00C00F0	9	hist_ready	Indicates that the histogram has finished taking samples
0xA00C00F0	7:0	accum_ready	Bit Z indicates that accumulator Z (connected to COLDATA Z) has finished taking samples
0xA00C00F4	31:0	accum_ch_total	Displays the total accumulated in channel accum_total_ch_sel when its accumulator is finished
0xA00C00F8	31:0	hist_out	Was used for peek-by-peek readout of histogram data, but no longer used.

Brookhaven National Laboratory DUNE

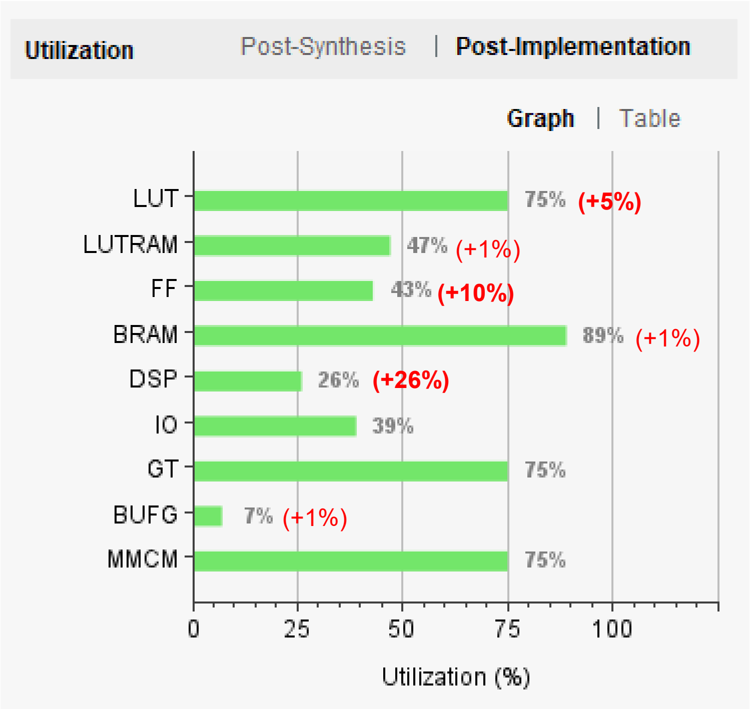
Latest production firmware uses status registers up to 0xA00C00D8

Resource utilization increase

Production firmware:



This firmware:



Includes the addition of the modified frame builder, the accumulator, and the histogram.

To-do list

1. Histogram study with a sine waveform instead of a ramp

2. ENOB study

3. Develop analysis scripts for ADC static performance metrics
   DC-offset, input range, gain, DNL/INL, overflow protection, and etc.

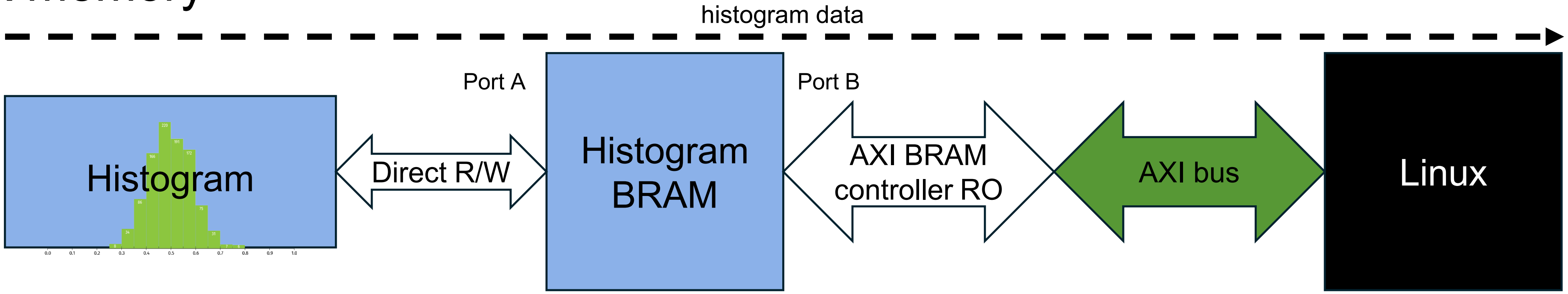
4. Organize a QC procedure – scripts, database, etc.

Summary

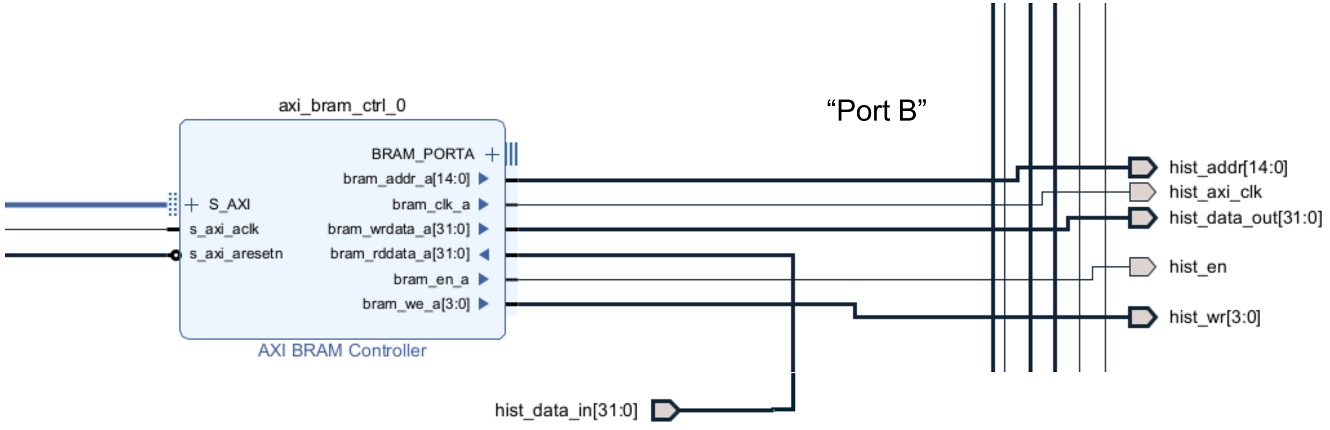
- Histogram development & study with a slow ramp signal is done
- DAT board has been deemed capable for ColdADC QC
  - All necessary QC items have been verified
  - New DAT revision will inject test pulse to each channel independently

AXI memory mapping

To speed up readout of histogram data, I implemented a AXI BRAM controller in the firmware's block design that accesses one side of the histogram dual-port block memory



2^14 x 16 bits = 32 KB BRAM



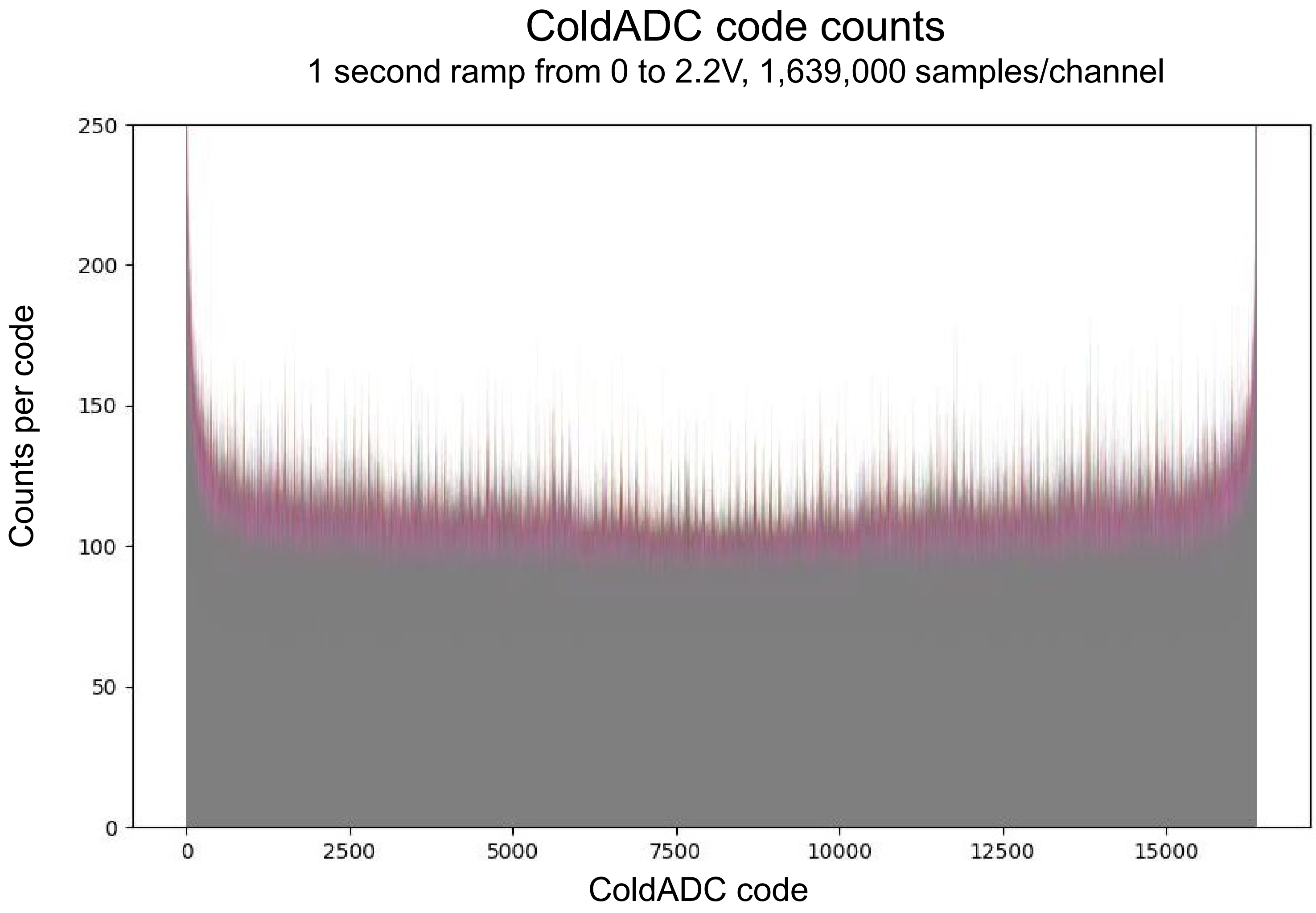
AXI memory mapping

/zynq_ultra_ps_e_0

/zynq_ultra_ps_e_0/Data (40 address bits : 0x00A0000000 [256M] ,0x0400000000 [4G] ,0x1000000000 [224G])

/dbg/debug_bridge_0/S_AXI	S_AXI	Reg0	0x00_A000_0000	64K	0x00_A000_FFFF
/coldata_i2c_dual0/coldata_	S00_AXI	S00_AXI_reg	0x00_A001_0000	64K	0x00_A001_FFFF
/tx_mux_wib_tux_0/S_AXI	S_AXI	reg0	0x00_A002_0000	64K	0x00_A002_FFFF
/coldata_fast_cmd_0/S00_A	S00_AXI	S00_AXI_reg	0x00_A003_0000	64K	0x00_A003_FFFF
/coldata_i2c_dual1/coldata_	S00_AXI	S00_AXI_reg	0x00_A005_0000	64K	0x00_A005_FFFF
/coldata_i2c_dual2/coldata_	S00_AXI	S00_AXI_reg	0x00_A007_0000	64K	0x00_A007_FFFF
/coldata_i2c_dual3/coldata_	S00_AXI	S00_AXI_reg	0x00_A009_0000	64K	0x00_A009_FFFF
/axi_iic_0/S_AXI	S_AXI	Reg	0x00_A00B_0000	64K	0x00_A00B_FFFF
/reg_bank_64_0/S00_AXI	S00_AXI	S00_AXI_reg	0x00_A00C_0000	32K	0x00_A00C_7FFF
/axi_bram_ctrl_0/S_AXI	S_AXI	Mem0	0x00_A00C_8000	32K	0x00_A00C_FFFF
/axi_gpio_1/S_AXI	S_AXI	Reg	0x00_A00D_0000	64K	0x00_A00D_FFFF
/daq_spy_all/daq_spy_0/axi	S_AXI	Mem0	0x04_4000_0000	256K	0x04_4003_FFFF
/daq_spy_all/daq_spy_1/axi	S_AXI	Mem0	0x04_4010_0000	256K	0x04_4013_FFFF
/daq_spy_all/daq_spy_2/axi	S_AXI	Mem0	0x04_4020_0000	256K	0x04_4023_FFFF

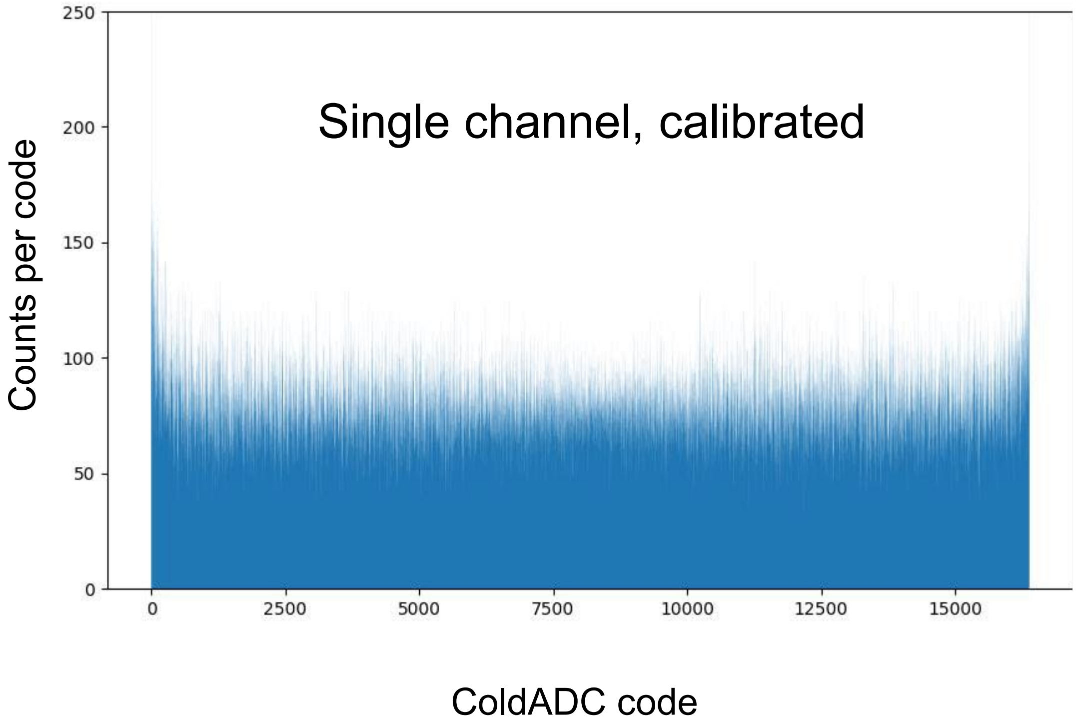
Histogram results

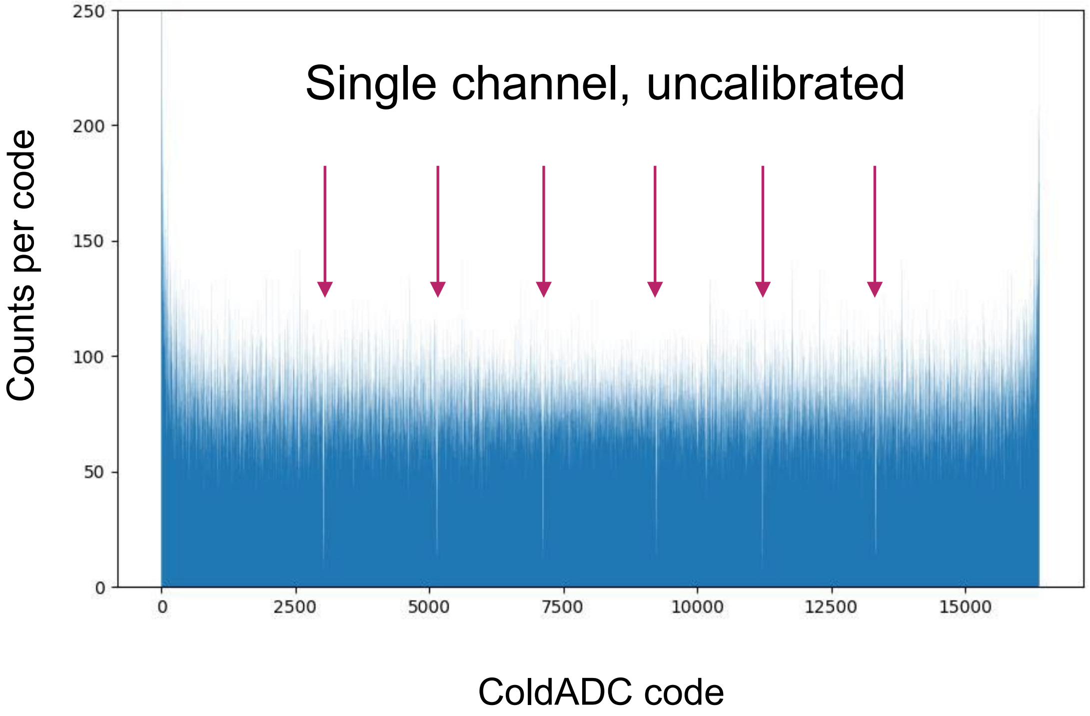


Histogram results

ColdADC code counts

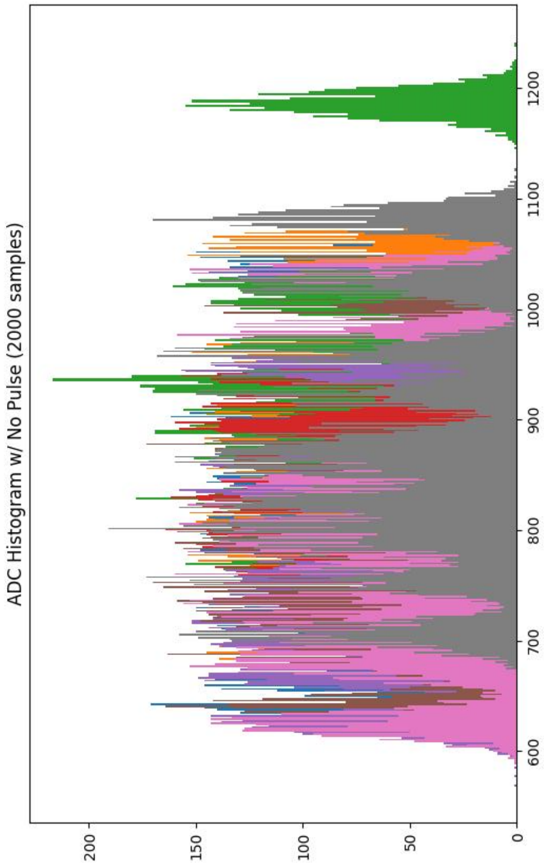
1 second ramp from 0 to 2.2V, 1,639,000 samples/channel

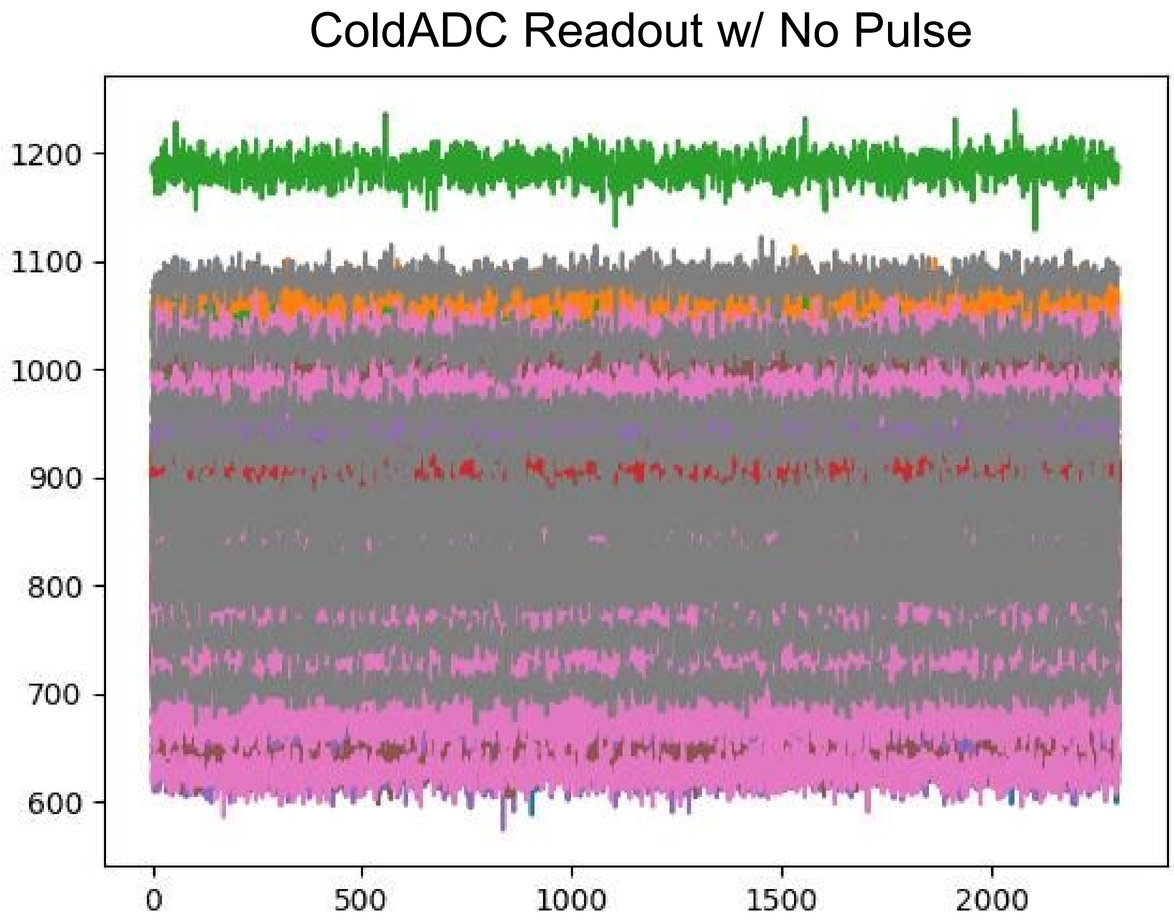




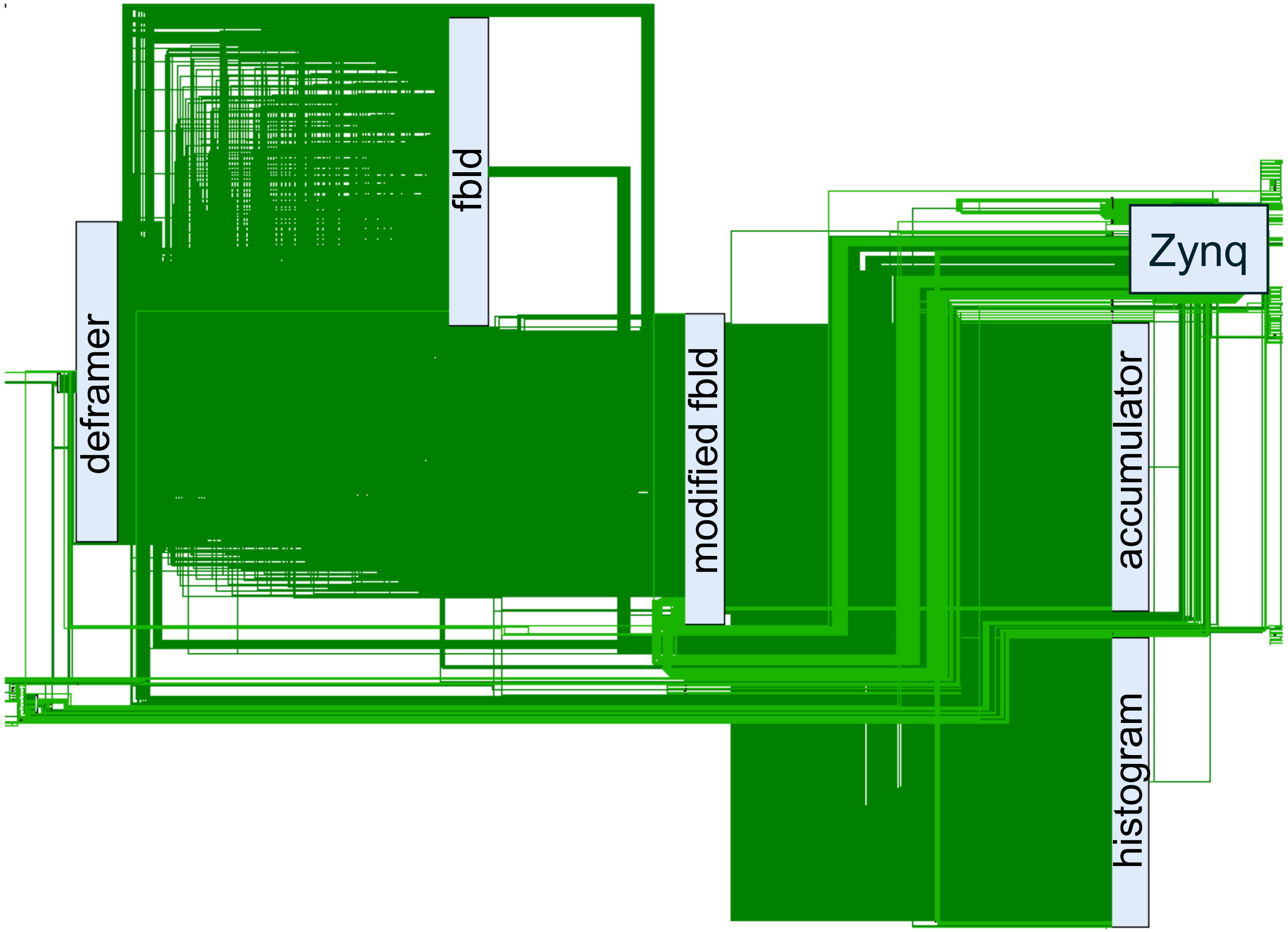
Further analysis is undergoing to extract some ADC DNL/INL

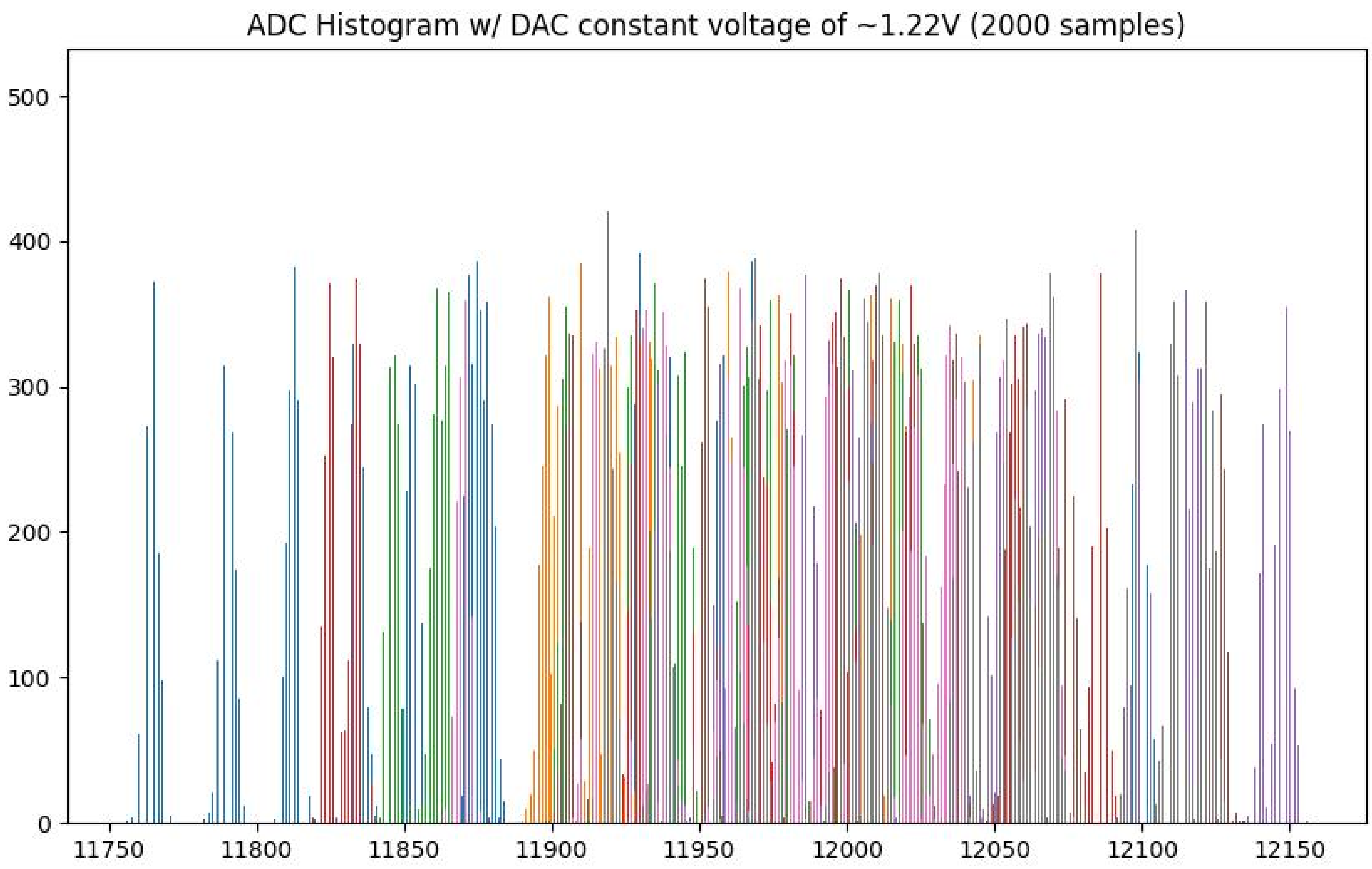
Spy buffer vs. histogram comparison





Backup slide: Vivado firmware schematic





Brookhaven National Laboratory DUNE

Accumulator software (still in development)

Procedure:

1. Set DAT registers so that ADC channels are connected to the ADC DAC
2. Set DAC to 0 LSB
3. Set number of samples to take
4. Trigger the accumulators
5. When accumulators finished, read channel totals out one by one
6. Set DAC to 1 LSB
7. …etc.

Command line:

```
root@dune-wib:~/BNL_CE_WIB_SW_QC# python3 adc_dac_cal.py
```

```
python3 adc_dac_plot.py tmp_data/ADCcal_02_10_2021_06_19_06.bin
```

Brookhaven National Laboratory DUNE

Histogram software (still in development)

Procedure (after hardware setup):

1. Set WIB register so that P8 LEMO input is sent over data cable to the DAT
2. Set DAT registers so that ADC channels are connected to WIB external signal
3. Set channel_to_analyze to 0
4. Wait until live channel monitor register equals 0x0000
5. Trigger histogram
6. When histogram indicates it’s finished, copy the histogram block of memory
7. Set channel_to_analyze to 1
8. …etc.
9. Save the copied data to a .bin file or similar

Command line:

```
root@dune-wib:~/BNL_CE_WIB_SW_QC# python3 adc_hist.py 1639000
```

```
python3 adc_hist_plot.py tmp_data/ADChist_22_09_2021_06_02_19.bin
```

Faster spy buffer decoding using DUNEDAQ C++ software

- Initial WIB Ethernet (HERMES) frame decoding was done with Python (very slow)
- Downloaded source code necessary for decoding frames to WIB
- Compiled .so library file for WIB
- Wrote script for generating a Windows or Linux library file for off-WIB analysis
- Sped up decoding time of one frame from a few seconds to instantaneous