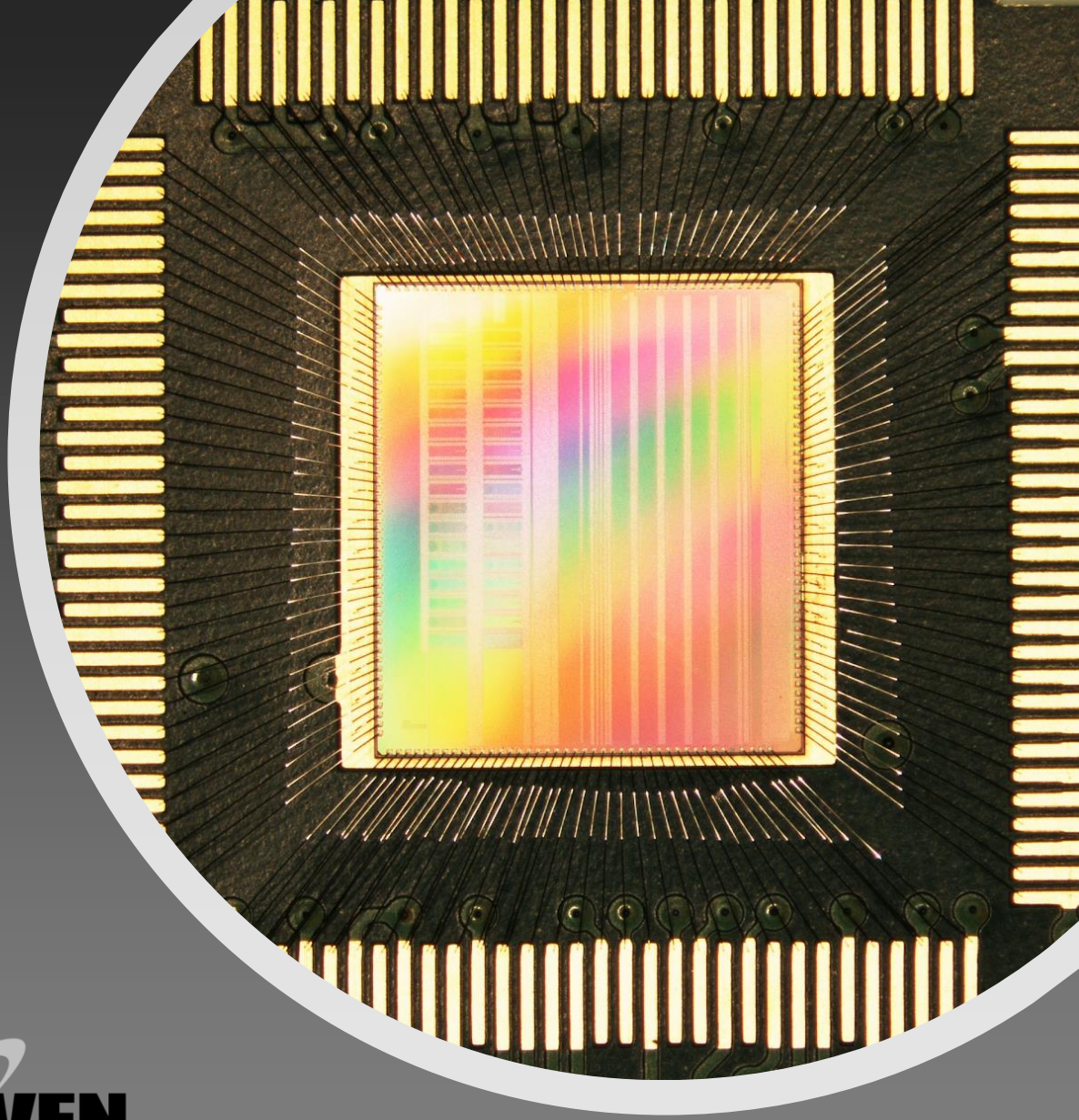
Evaluation and Characterization of 16-Channel ADC ASIC at Cryogenic Temperature (77 K)

for TPC Front-End Readout Electronics System

in DUNE Experiment

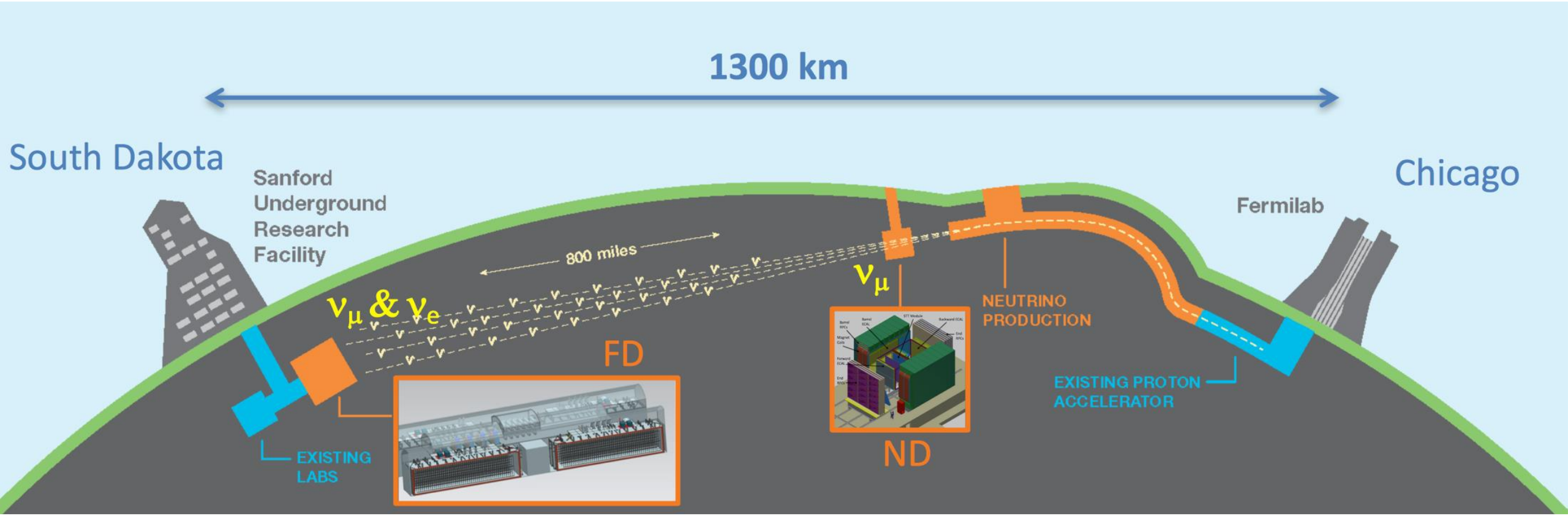


Midterm Presentation
Edoardo Lopriore

U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

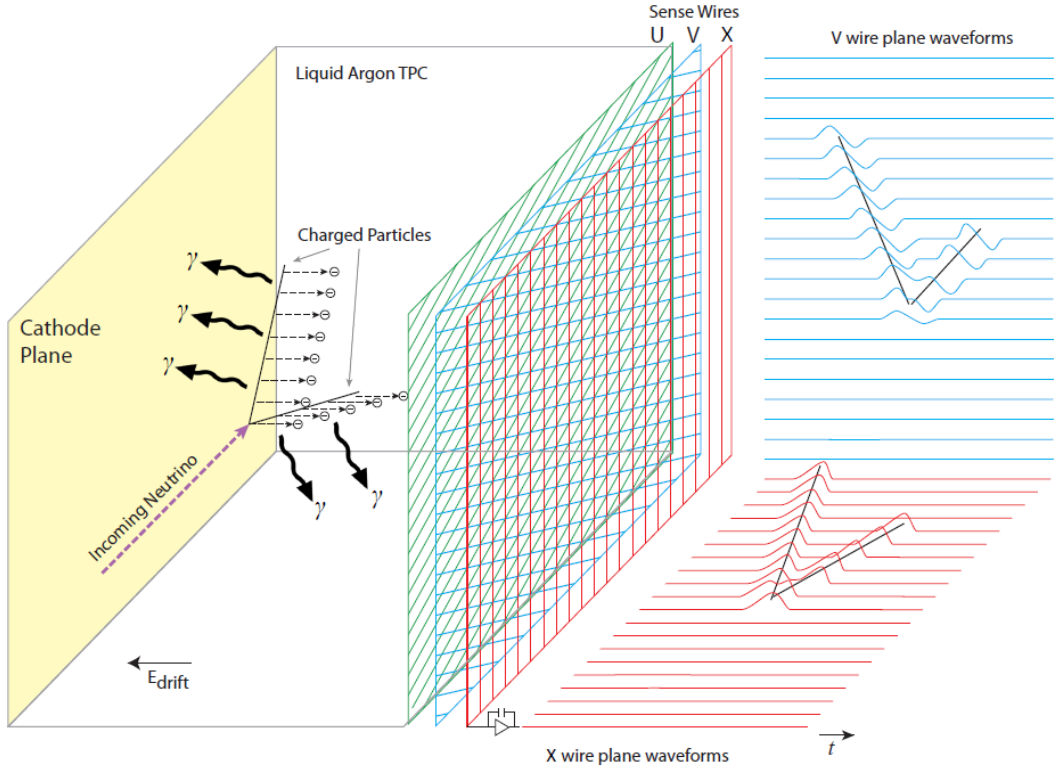
The Deep Underground Neutrino Experiment



Main goals:

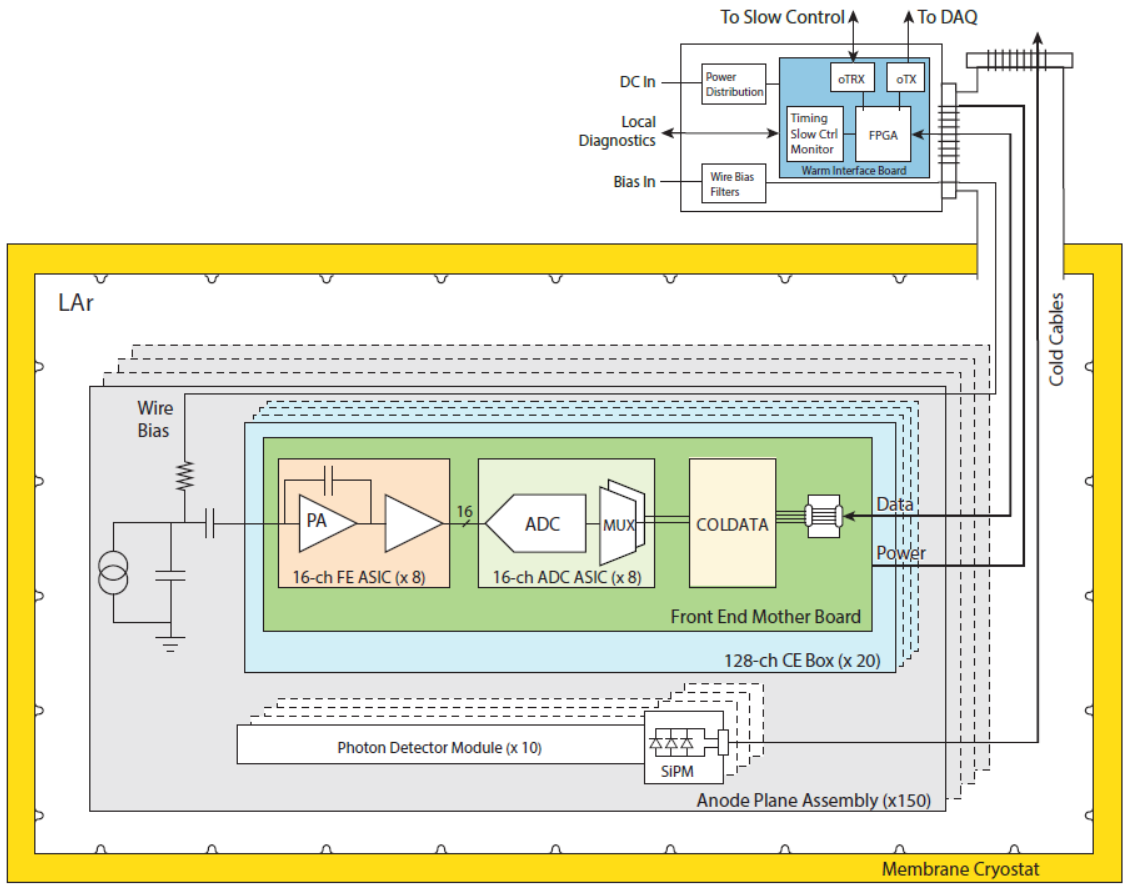
- Leptonic charge parity (CP) violation
- Physics beyond the Standard Model
- Neutrino bursts from supernovae

Liquid Argon Time Projection Chamber: Why Cold Front End Electronics?



- Decouples the electrode and cryostat design from the readout design: independent of cable lengths, lower noise
- Signal multiplexing reduces number of cables (less outgassing) and number of feedthroughs: optimum TPC and cryostat configurations

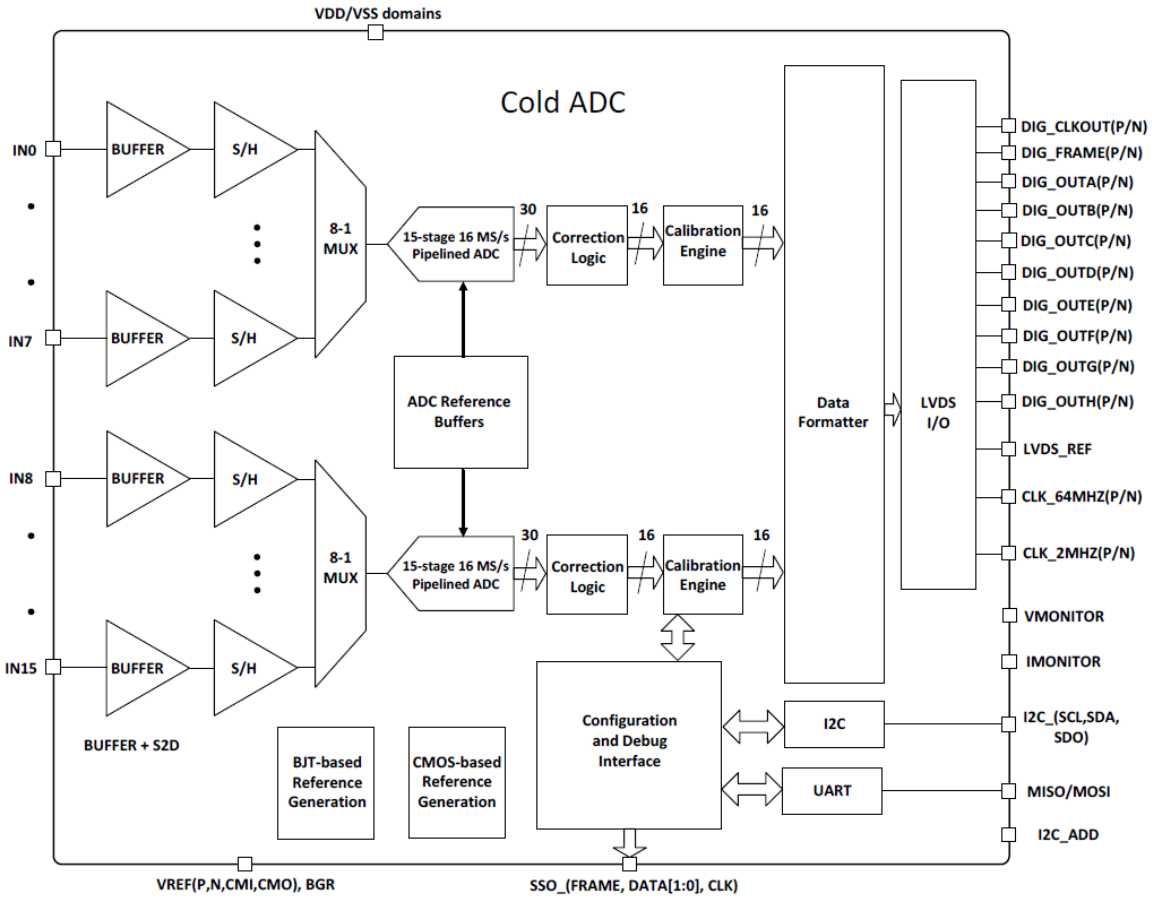
Front End Mother Board



Three ASICs:

- LArASIC: 16 channel preamplifier and pulse shaper
- Cold ADC: 16 channel analog to digital converter operating at 2 MS/s per channel
- COLDATA: 64 channel control and communications

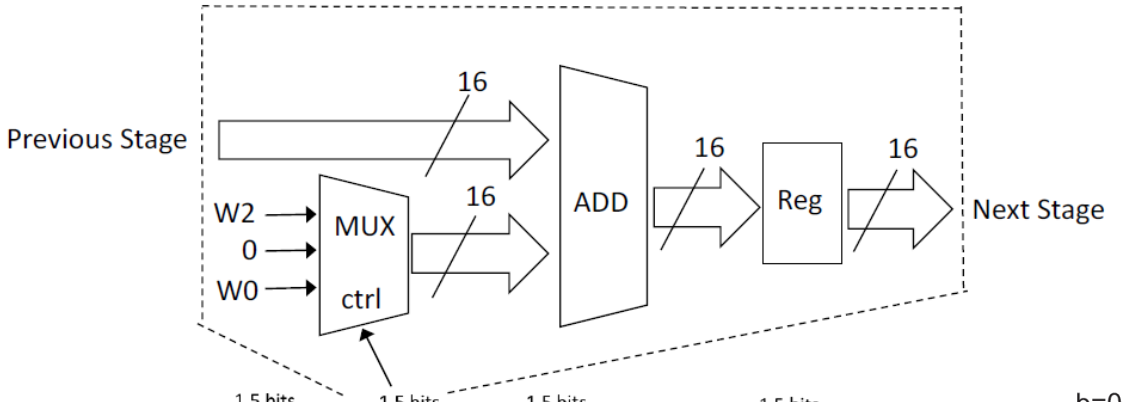
Cold ADC: working principles

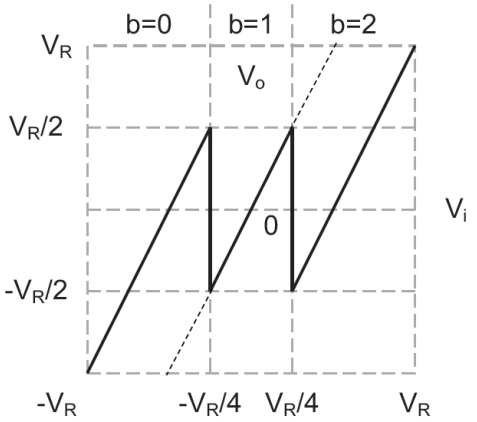


- CMOS Technology at 77 K: higher mobility and lower thermal fluctuations
  - Higher gain, transconductance, speed
  - Lower noise
- Two multiplexed 16 MS/s pipelined ADCs, 15 stages each
- 16 bit ADC, but some bits used for calibration

12 bit, 16 channel ADC

Cold ADC: 1.5 bit Pipelined Architecture



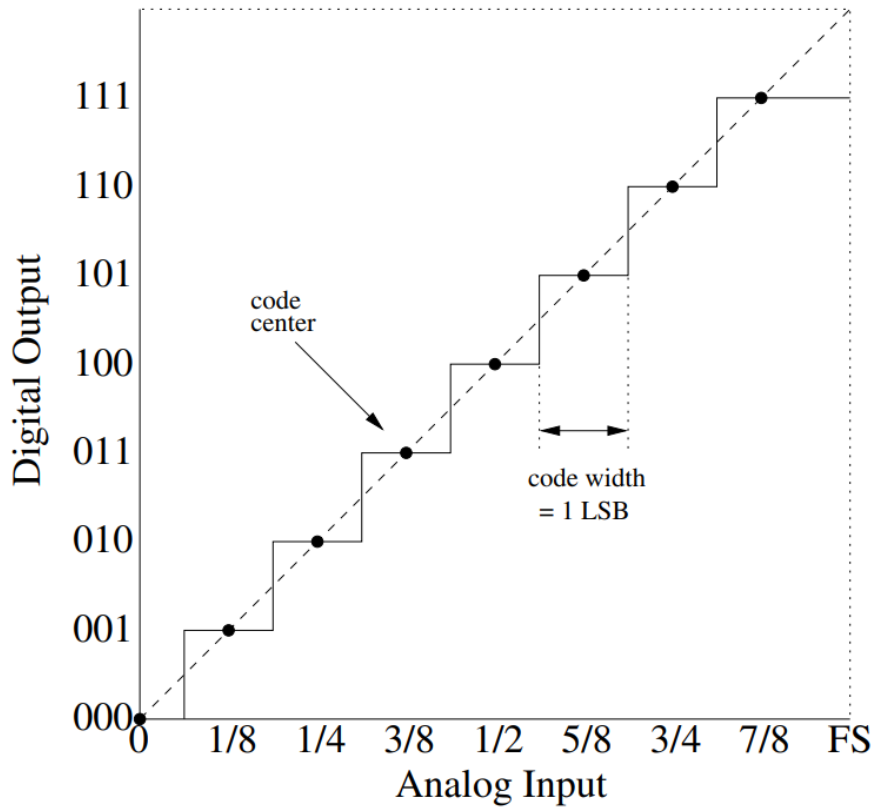


- Each stage makes a three-level coarse decision, selects calibration "weights", outputs digitalization difference
  - Calibration is necessary to obtain high linearity
- Redundancy: "errors" are corrected in sequent stages [1]

[1] Lewis, S., Fetterman, H., Gross, G., Ramachandran, R., & Viswanathan, T. (1992). A 10-b 20-Msample/s analog-to-digital converter. *IEEE Journal of Solid-State Circuits*, *27*(3), 351–358

Characterization: Static Behavior

Ideal 3 bit ADC characteristic



- Differential Non-Linearity (DNL): deviation of the code transition widths from the ideal width of 1 LSB
- Integral Non-Linearity (INL): the distance of the code centers in the A/D converter characteristic from the ideal line

$$LSB = \frac{V_{FS}}{2^N}$$

$$DNL(i) = \frac{binwidth(i) - LSB}{LSB}$$

$$INL(i) = \sum_{n=0}^{i} DNL(n)$$

Static Behavior: Code Density Method

- Input signal with known probability density function (PDF): compare output histogram with ideal PDF to obtain linearity parameters
- Ramp or sine? It is easier to generate a clean sine wave. However, both are to be done for comparison

Sine Wave Code Density

- Necessity of a high number of samples [2]:

$$N_{samp} = \frac{\pi 2^{N-1} z_{\alpha/2}^2}{\beta^2}$$

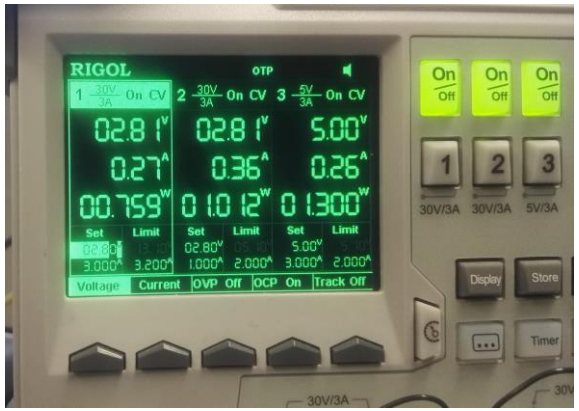
> 4 million samples for high confidence and resolution < 0.1 LSB

[2] J. Doernberg, H.-S. Lee, and D. Hodges, "Full-speed testing of A/D converters," *IEEE Journal of Solid-State Circuits*, vol. 19, no. 6, pp. 820–827, 1984.

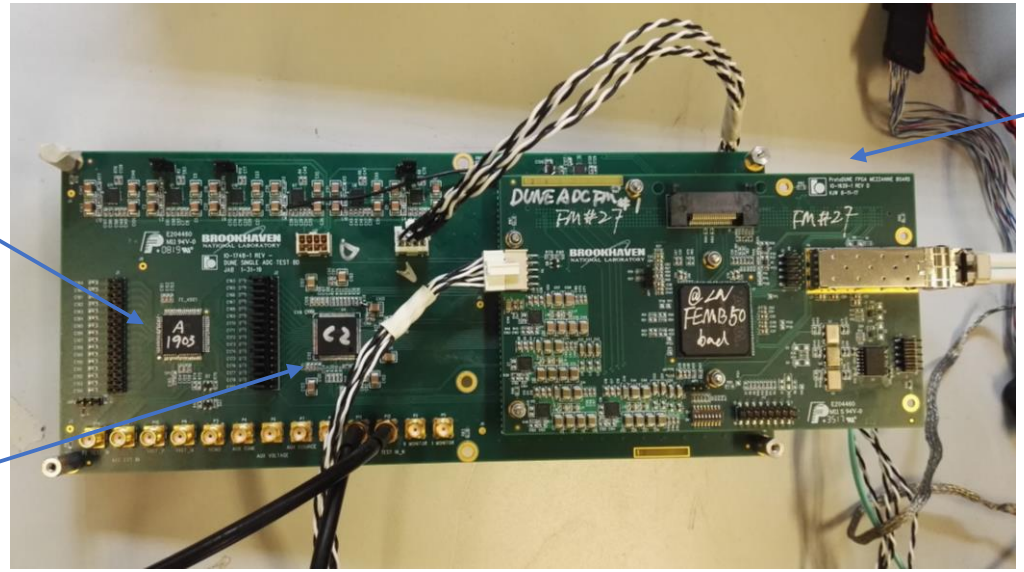
Measurements Set-up

Low distortion sine wave generator
from Stanford Research Systems

Power supply



Supplies the on-board regulators → lower noise



- An FPGA mezzanine reproduces the behavior of COLDATA
- The sine wave generator inputs a signal into the ADC Test Input through SMA cables

Static Behavior: Analysis Implementation

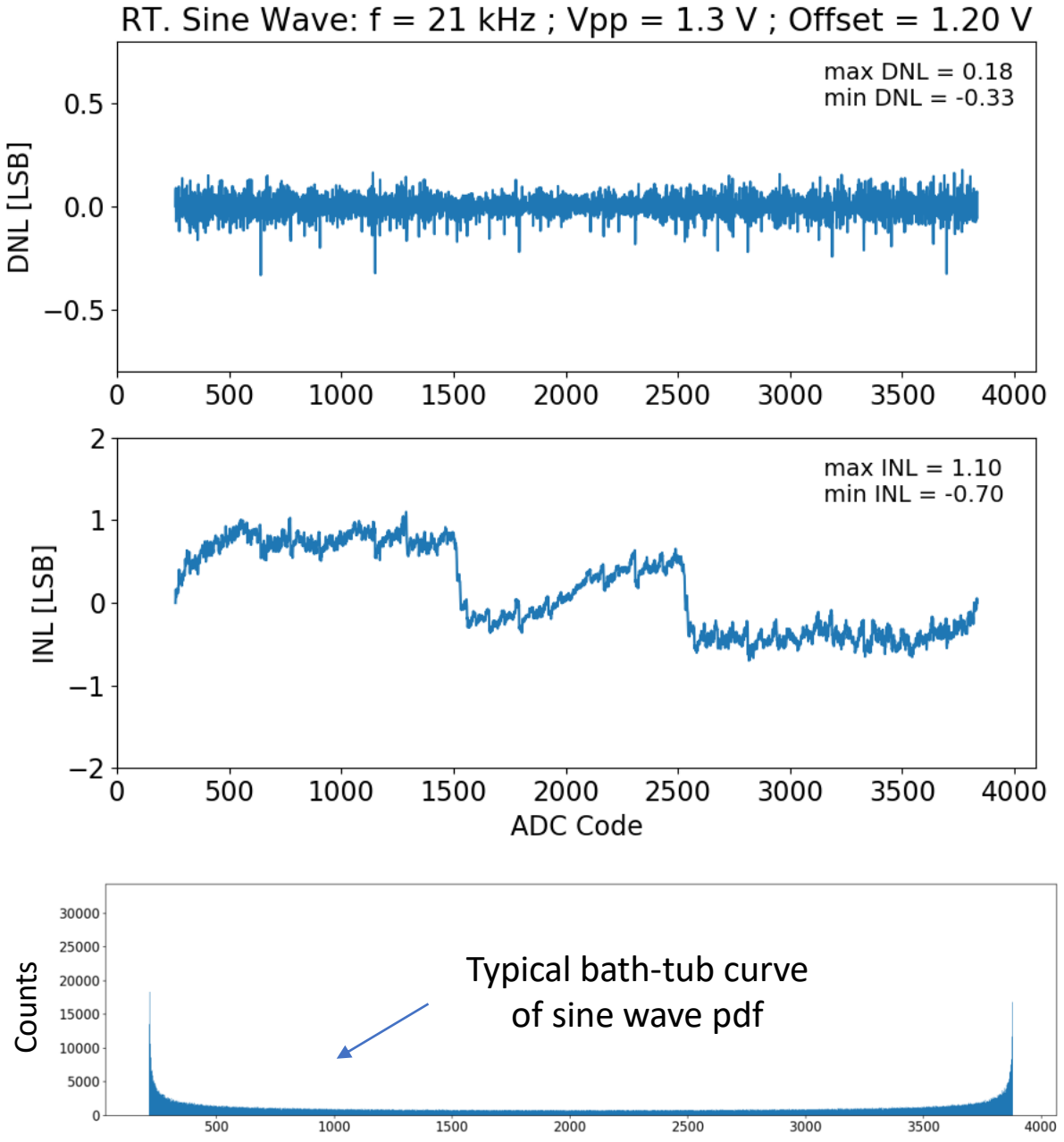
- Some methods for linearity histogram tests are dependent on sine wave amplitude in a complex way (rely on amplitude measurements)
- Cumulative Histogram method: transition voltages directly dependent on amplitude, therefore normalizable. Best way of calculating DNL and INL with sine wave [1]

$$V_j = -A\cos\left(\frac{\pi}{M}\sum_{k=0}^{j} H_k\right) \longrightarrow \mathrm{INL}_j = \frac{V_j - V_1}{1\ \mathrm{LSB}} \qquad \mathrm{DNL}_j = \frac{V_{j+1} - V_j}{1\ \mathrm{LSB}}$$

- Python code fully implemented, generates INL and DNL plot from raw ADC data

Static Behavior: Results (so far)

- Measurements shown are obtained with ADC differential test input at room temperature
- Great results! More than compatible with tests by LBNL and FNAL
- Results with active Sample and Hold Amplifiers (SHA) and Single to Differential Converters (SDC) are in progress, together with cryogenic testing



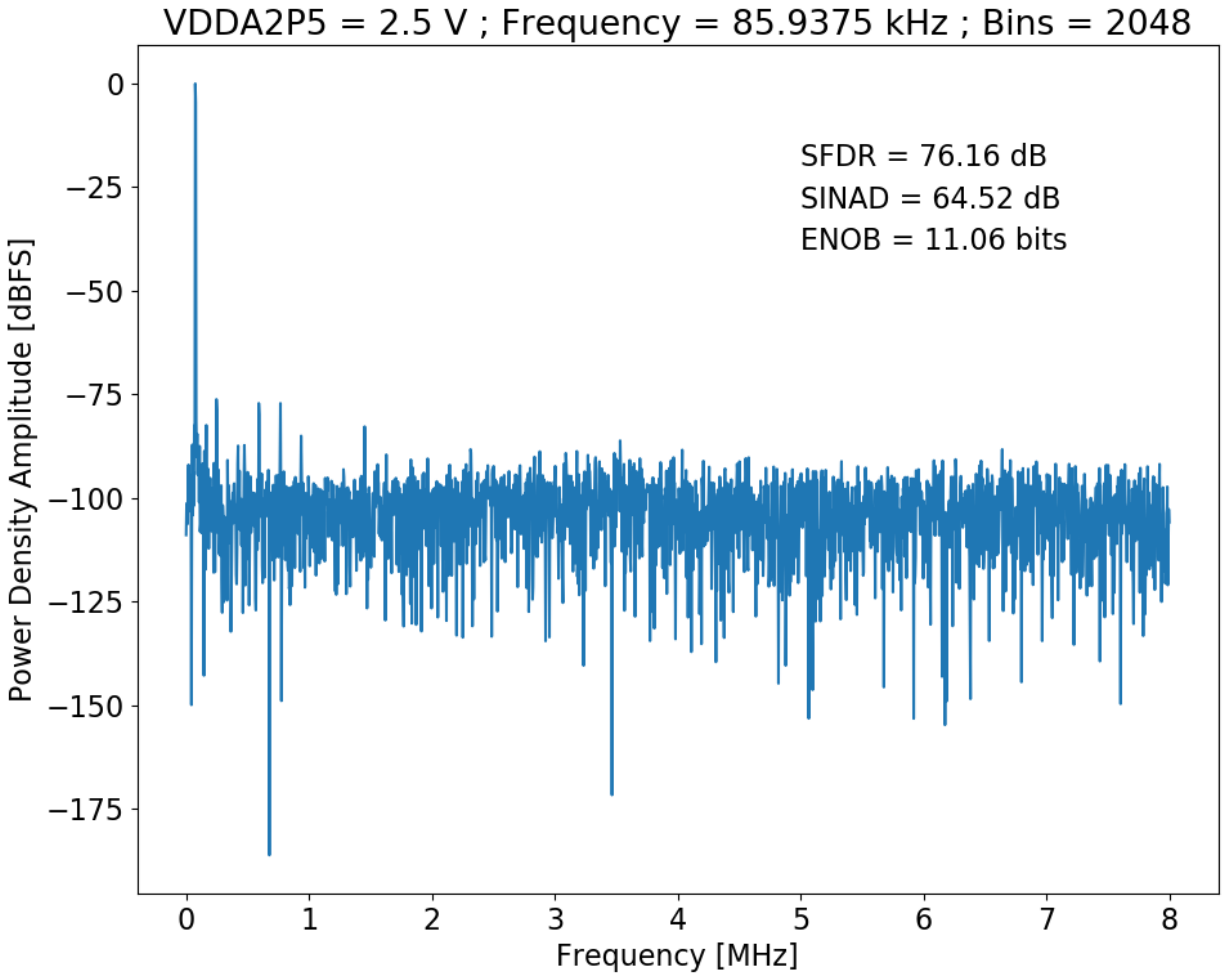
Characterization: Dynamic Behavior

- AC performance is characterized with various parameters, all derived by FFT of sampled sine wave data:
  - Signal to Noise and Distortion Ratio (SINAD or SNDR)
  - Spurious-Free Dynamic Range (SFDR), the amplitude ratio between fundamental and the highest of the other peaks in the FFT spectrum
  - Effective Number of Bits (ENOB), the number of bits of an ideal ADC with the same SINAD as the ADC under test

- Measurement Technique: Coherent Sampling → $f_{in} = \frac{N_{cycles}}{N_{samples}} f_s$
  - Guarantees integrity of frequency spectrum of sampled signal
  - Nsamples should be power of 2, for fastest Fourier transform
  - Ncycles must be mutually prime with Nsamples in order to obtain the desired coherence

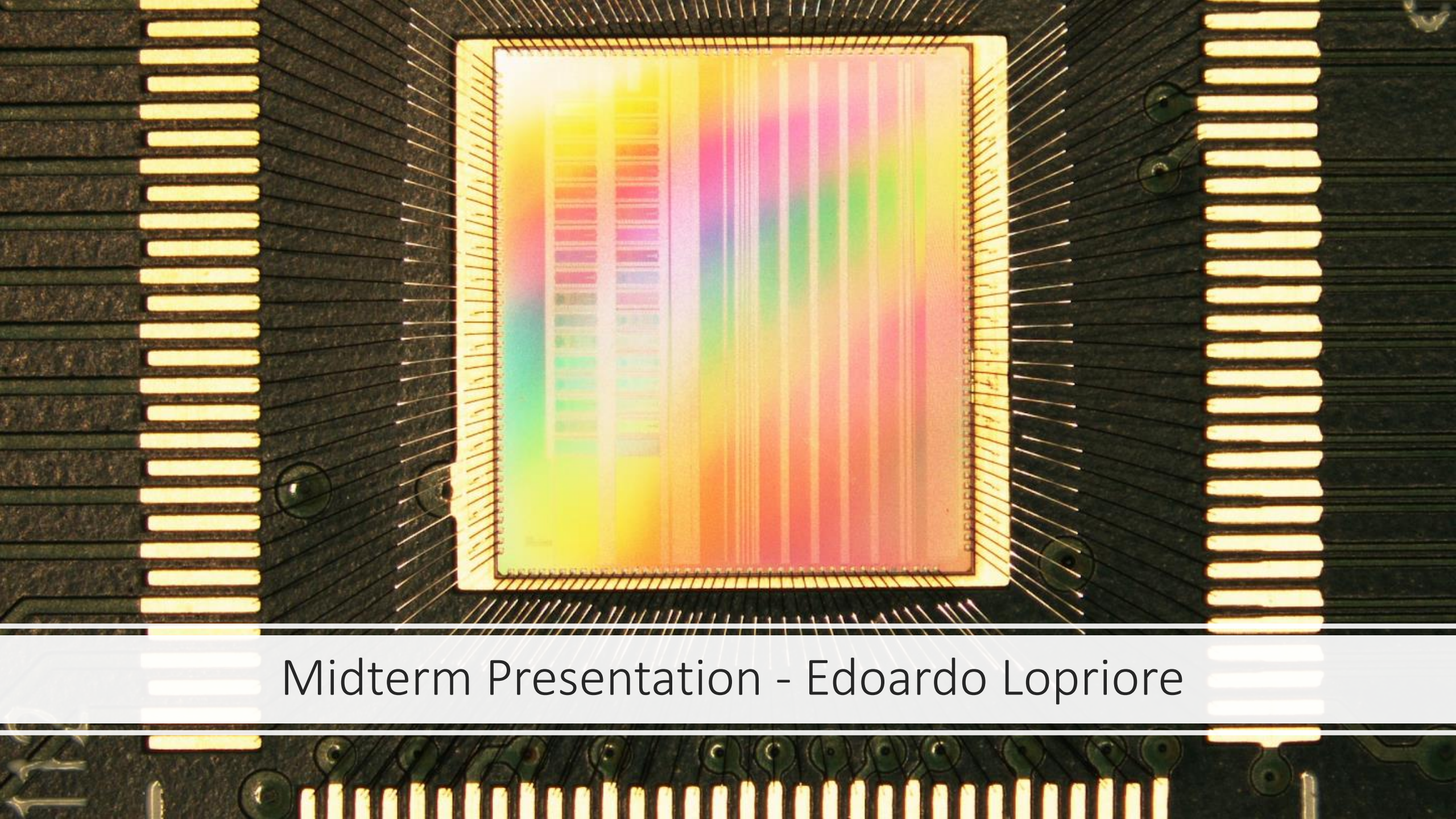
Dynamic Behavior: Implementation and Results

- Python code for FFT analysis and parameter extraction
- Graphic shows the Power Density Spectrum in dBFS (relative to Full Scale)
- Results perfectly compatible with other set-ups



Next Steps:

- Complete characterization of ADC with more active functionalities: study of the different linearity and dynamic performances with SHA and SDC powered on (ongoing)
- Characterization at cryogenic temperature, 77 K (ongoing)
- Testing of full-chain front-end system with active LArASIC, both at room temperature and at cold
- Code for Quality Assurance/Quality Control protocol, to be used for future final Front-End Motherboards



Midterm Presentation - Edoardo Lopriore

Backup Slides

Linearity Code

```
48 ##### ADC_TST_IN Sine Wave #####
49 N = 12
50 Ntot = 4194304
51 freq = 21 #kHz
52 vpp = 1.3 #V
53 offset = 1.20 #V, real input offset (double the sine wave
54
55
56 fn = "D:/ColdADC/ADC_TST_IN/lin" + "_%0.0fk"%freq
57         + "_%0.1fv"%vpp + "_%0.2fv"%offset + ".bin"
58
59 with open (fn, 'rb') as fp:
60     chns = pickle.load(fp)
61     chns = list(np.array(chns)//16)
62
63 chns[0] = chns[0][:Ntot]
64
```

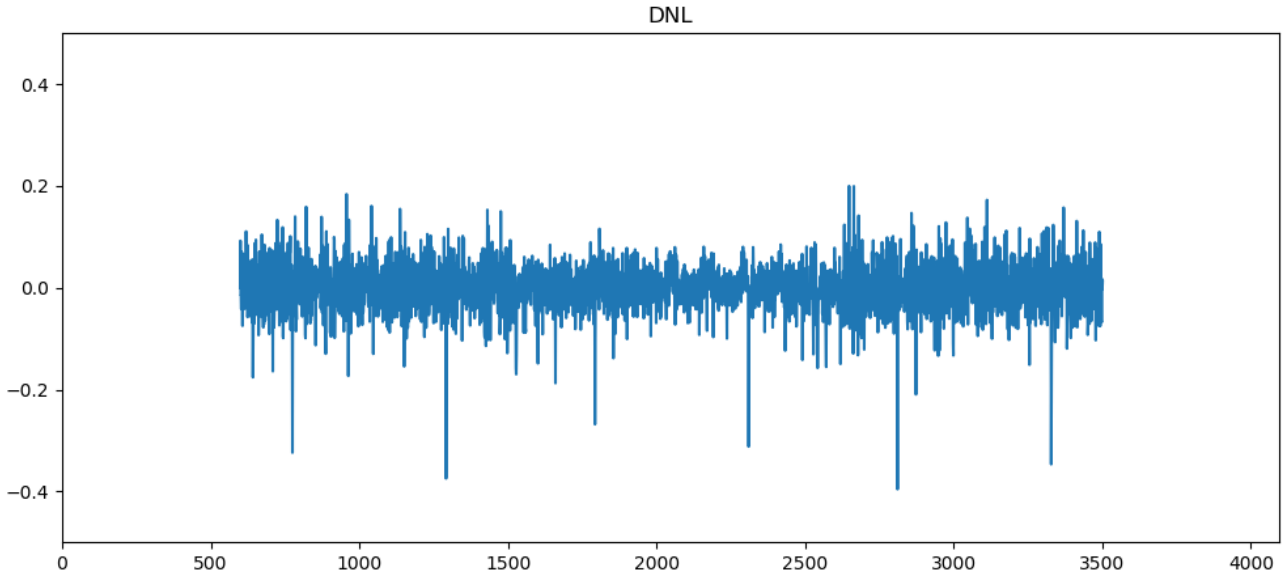
- Takes data from labeled folder, uses channel 0 as example, crops it to the number of samples used (2^22)

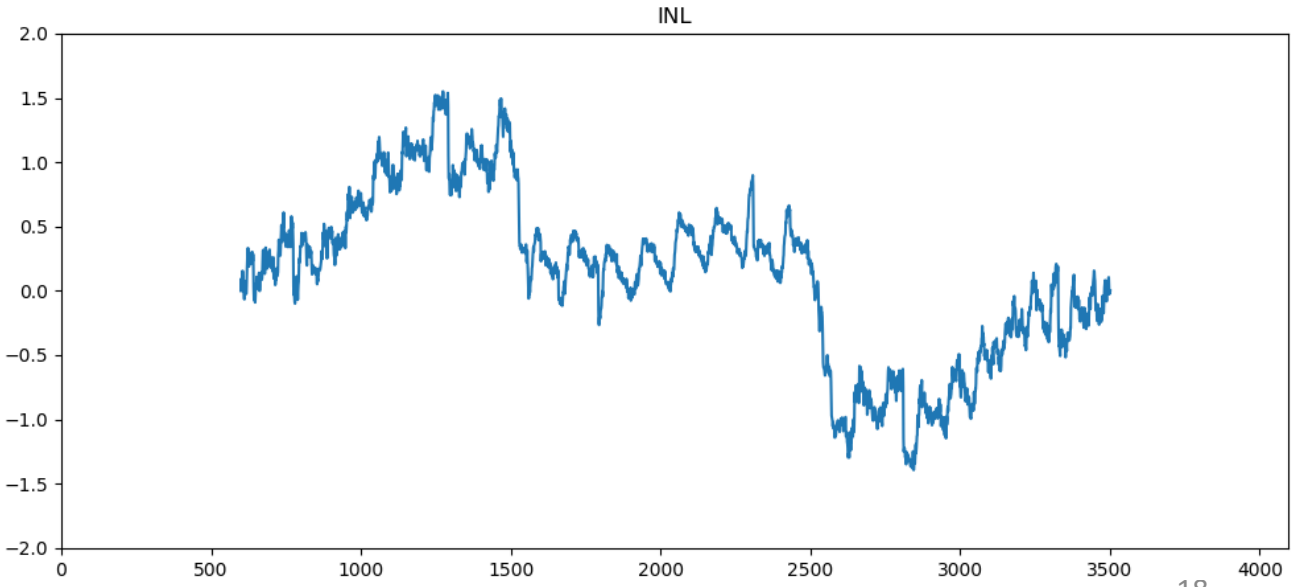
```
64 ## DNL and INL, Cumulative Histogram method ##
65 val, bins = np.histogram(chns[0], bins = 4095, range = (0,4095))
66 nonzero_bins = np.nonzero(val)
67 first_bin = nonzero_bins[0][0]
68 last_bin = nonzero_bins[0][-1]
69 val_nonzero = val[np.nonzero(val)]
70 sum_Hk = np.cumsum(val_nonzero)
71 #normalized transition voltages
72 Vj = - np.cos(np.pi*sum_Hk/Ntot)
73 end = len(Vj)
74 #linearized histogram
75 hlin =  Vj[1:] - Vj[:-1]
76 #first and last bit truncation (can be extended)
77 trunc=50;
78 hlin_trunc = hlin[trunc:-trunc]
79 #LSB calculation as average code width
80 lsb = np.sum(hlin_trunc) / len(hlin_trunc)
81 #DNL, concatenate a 0 at the beginning
82 dnl = np.insert(hlin_trunc/lsb-1, 0, 0.)
83
84 misscodes = np.where((dnl > 1) | (dnl < -1))
85 if(np.array(misscodes).size > 0):
86     print('Number of misscodes found = %d' %np.array(misscodes).size)
87
88 inl = np.cumsum(dnl)
89
```

- Computes DNL and INL with cumulative histogram method
- Displays eventual misscodes

Linearity Results

- Better look at other DNL and INL values
- Run obtained with:
  - Frequency 25 kHz
  - Amplitude 1.1 V
  - Input offset 1.3 V
- Amplitude variation doesn't affect linearity results





AC Analysis Code

- Elimination of DC and Nyquist frequencies
- Eventual eliminations of FFT impurities near fundamental
- Calculation of the various parameters
- Plotting function (not shown)

```
#eliminate DC and Nyquist frequency
trunc = 2
psd = psd[trunc:Ntot-trunc]

#eliminate spurious bins
psd_aux = np.copy(psd)
noise = min(psd)
spurious = 1
while max(psd_aux)>10**(4):
    mx_arr = np.where(psd_aux == max(psd_aux))
    mx = mx_arr[0]
    psd_aux[mx] = noise
    for k in range(1, spurious+1):
        psd[mx-k] = noise
        psd_aux[mx-k] = noise
        if(mx+k < len(psd)):
            psd[mx+k] = noise
            psd_aux[mx+k] = noise

fundamental = max(psd)
NAD = np.sum(psd) - fundamental

SINAD = 10*np.log10( fundamental / NAD )

Vfullscale = 1.553
Vinput = 1.0
ENOB = (SINAD - 1.76 + 20*np.log10(Vfullscale/Vinput)) / 6.02

psd_NAD = np.copy(psd)
psd_NAD[np.where(psd == max(psd))] = 0
SFDR = 10*np.log10( fundamental / max(psd_NAD))

```

Cryogenic Testing Set-up

- Connection used doesn’t work @ 77K, need to stay above LN2 level
- Preliminary set-up, currently in progress
- Some FPGAs don’t respond well, in that case need to change mezzanine