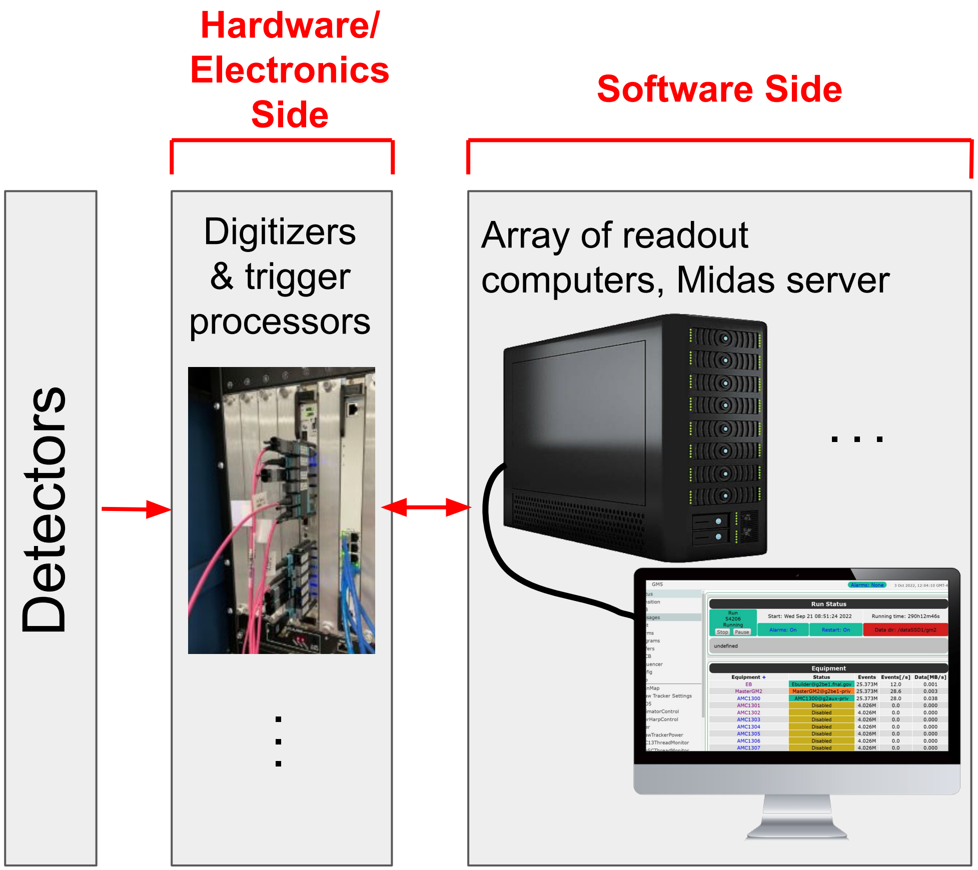
PIONEER DAQ

Jack Carlton
University of Kentucky
June 19th, 2024

Hardware vs. Software Side

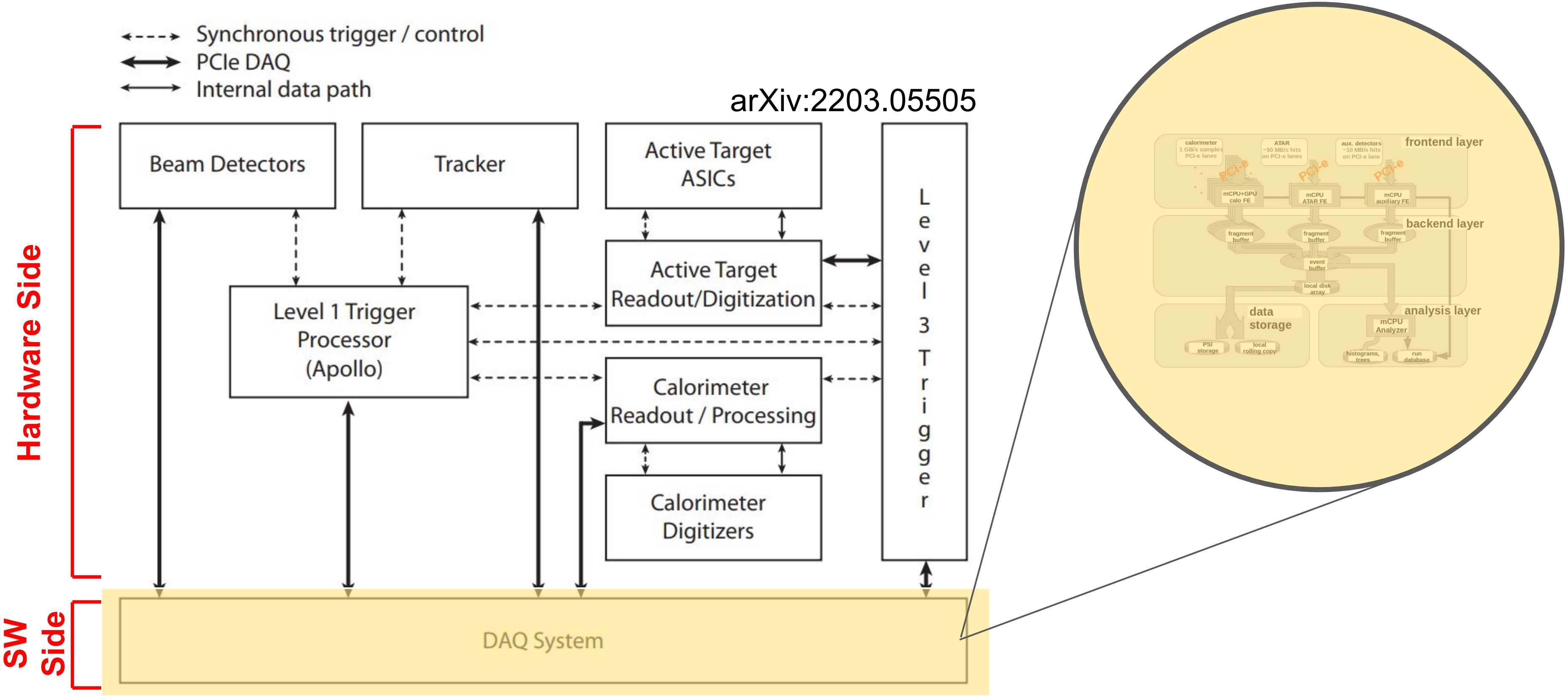
- Usually “DAQ” refers to the “software side” (i.e. MIDAS and related tools)
    - Loosely used for hardware (electronics) side as well

- I like to differentiate between the software and hardware sides



j.carlton@uky.edu

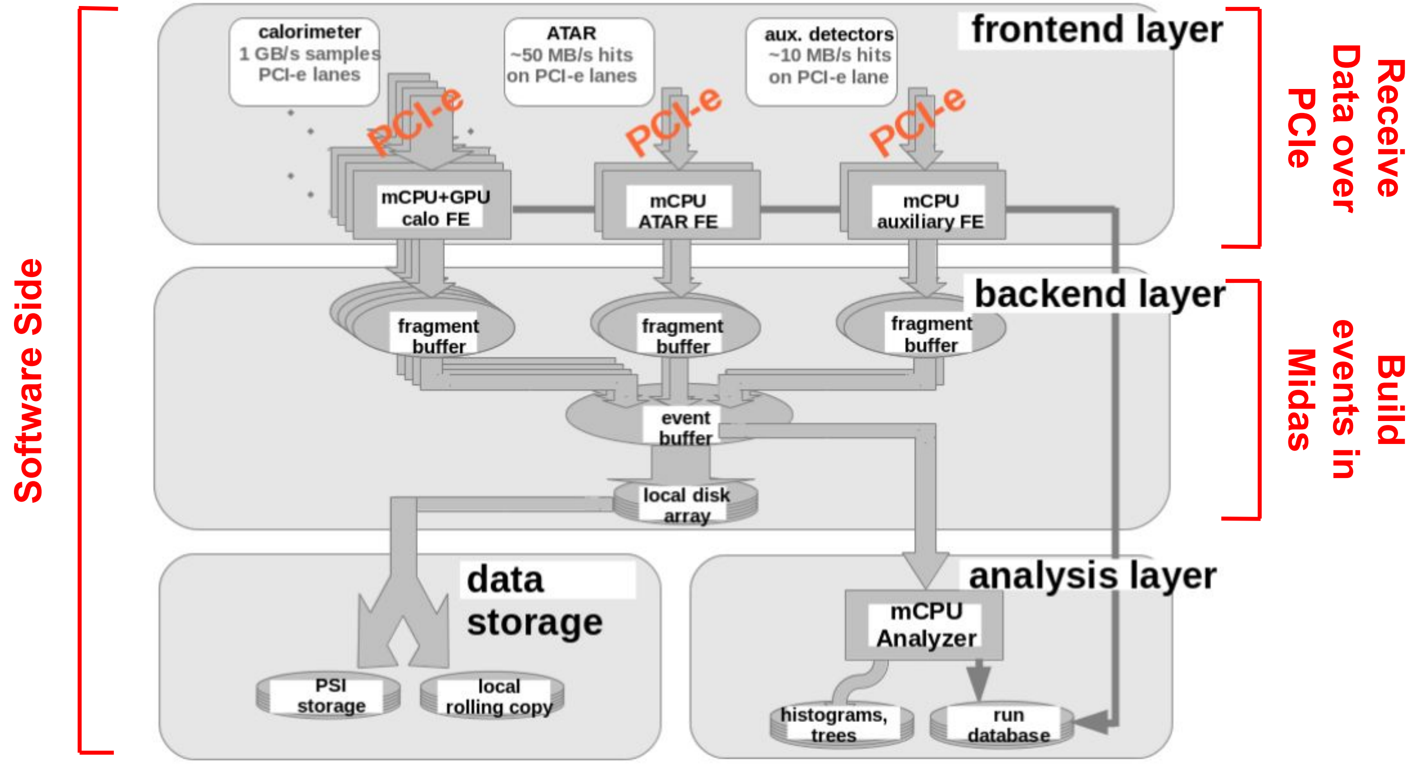
Proposed Data Acquisition (DAQ) Framework



j.carlton@uky.edu

Proposed Data Acquisition (DAQ) Framework

arXiv:2203.05505



Data Rates

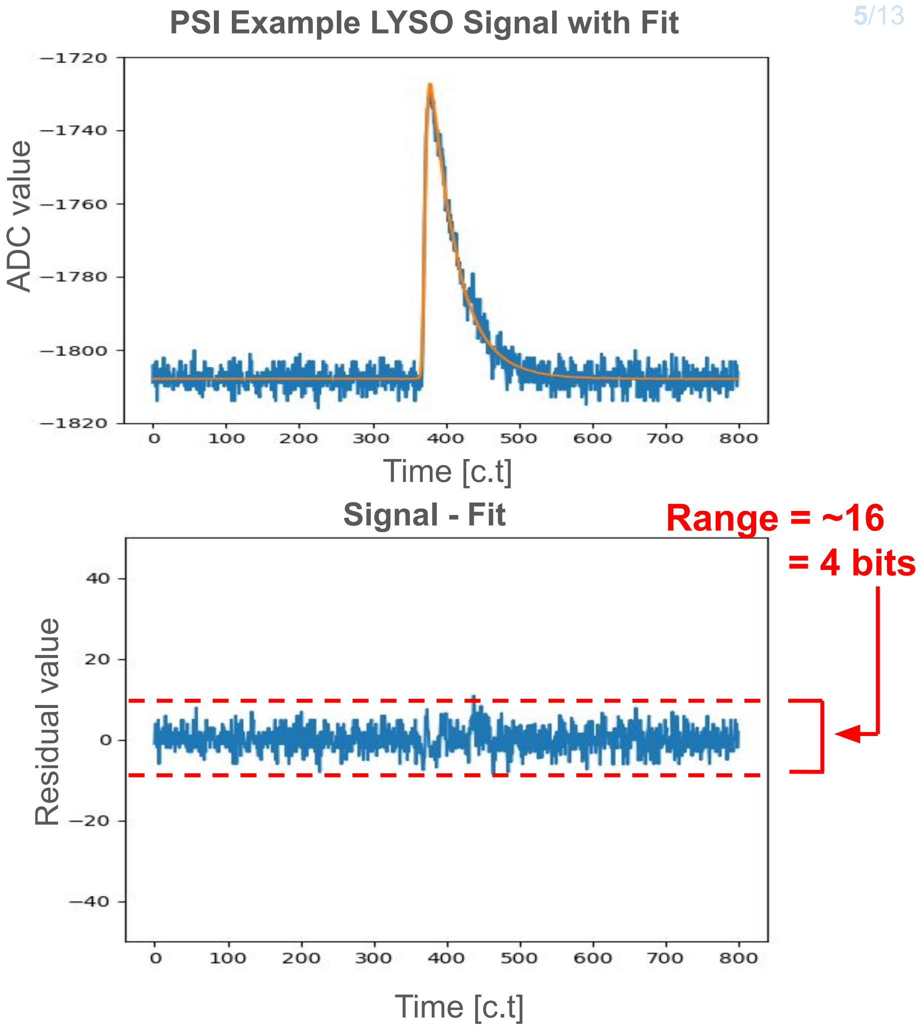
arXiv:2203.01981

triggers	prescale	range TR(ns)	rate (kHz)	CALO ΔT(ns)	CALO chan	CALO MB/s	ATAR digitizer ΔT(ns)	ATAR digitizer chan	ATAR digitizer MB/s	ATAR high thres chan	ATAR high thres MB/s
PI	1000	-300,700	0.3	200	1000	120	30	66	2.4	20	0.012
CaloH	1	-300,700	0.1	200	1000	40	30	66	0.8	20	0.004
TRACK	50	-300,700	3.4	200	1000	1360	30	66	27	20	0.014
PROMPT	1	2,32	5	200	1000	2000	30	66	40	20	0.2

- PIONEER DAQ expects data rate of **~3.5GB/s**
- This is **~100,000 TB/year**
- How do we compress this in real time?
  - Fit data, store fit parameters
  - Compress and store residuals, throw some out
  - Graphics Processing Units (GPUs) used for this operation

Template Fitting

- Can construct a continuous template for our traces $T(t)$
- Can fit traces using template: $f(t) = A \cdot T(t - t_0) + B$
- Storing unfit traces takes ~12 bits per ADC sample
- Storing residuals takes ~4 bits per ADC sample
- By fitting, we can compress the data by a **factor of ~3**



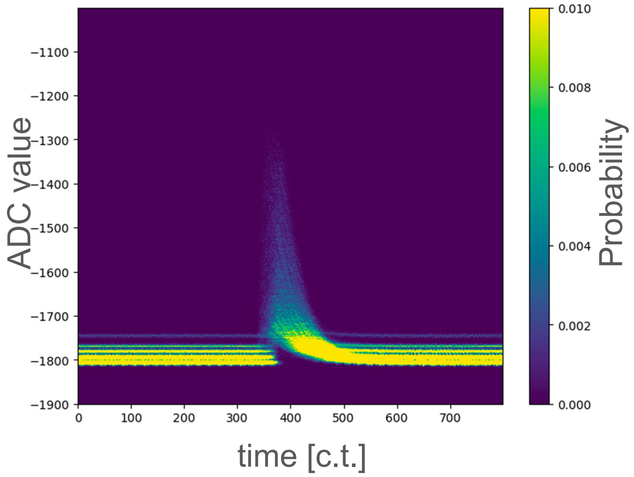
j.carlton@uky.edu

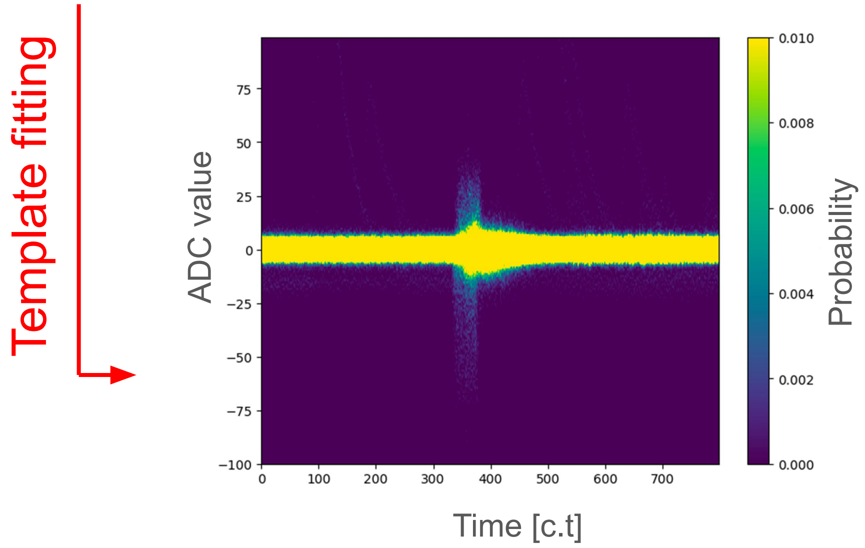
Template Fitting

- Data from PSI test beam

- Each vertical slice corresponds to pdf $p_i(x_i)$

- Template fit drastically reduces spread of data





Theoretical Best Compression

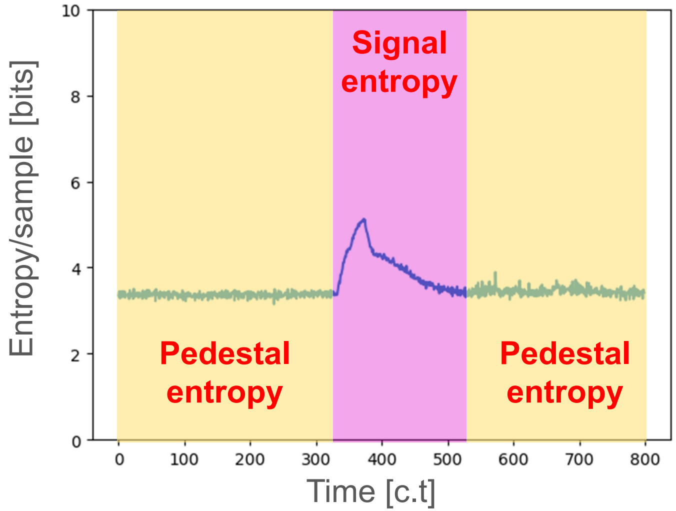
- For lossless compression, the best possible compression rate is the entropy rate
- Entropy rate of pedestal part of signal is **3.4 bits per ADC sample**
  - A perfect fit would reduce signal to pedestal noise
- Best possible data storage rate 3.5 GB/s → **~1 GB/s**
  - Assumes similar noise to PSI test beam data
- Realistically the data storage rate depends how good our fit is
  - Assuming entropy rate of ~5 bits/sample 3.5 GB/s → ~1.5 GB/s

Entropy Rate Formula

$$H(X_i) = \sum_{\text{traces}} p(X_i) \log_2 (p(X_i))$$

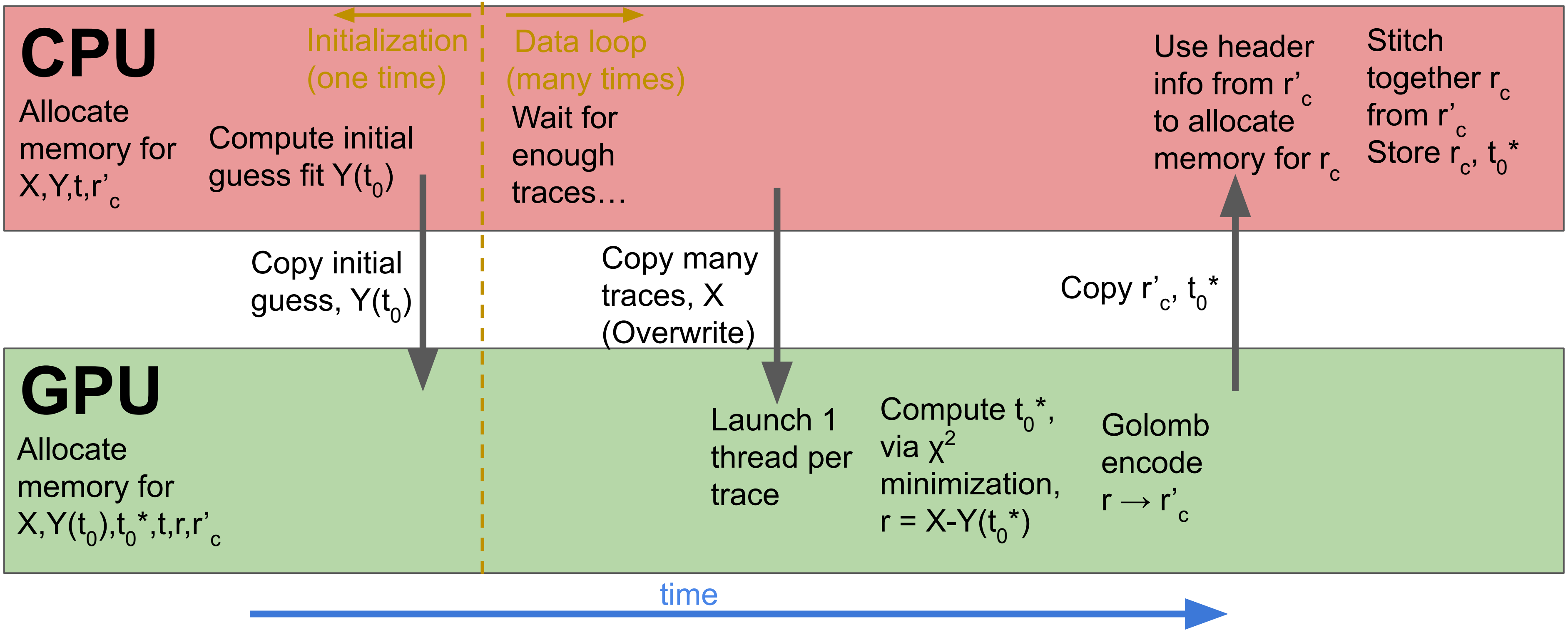
$$X_i \equiv \text{Random variable for } i^{\text{th}} \text{ ADC sample}$$

Entropy Rate of PSI Test Beam Data After Fitting



Real Time Compression Algorithm

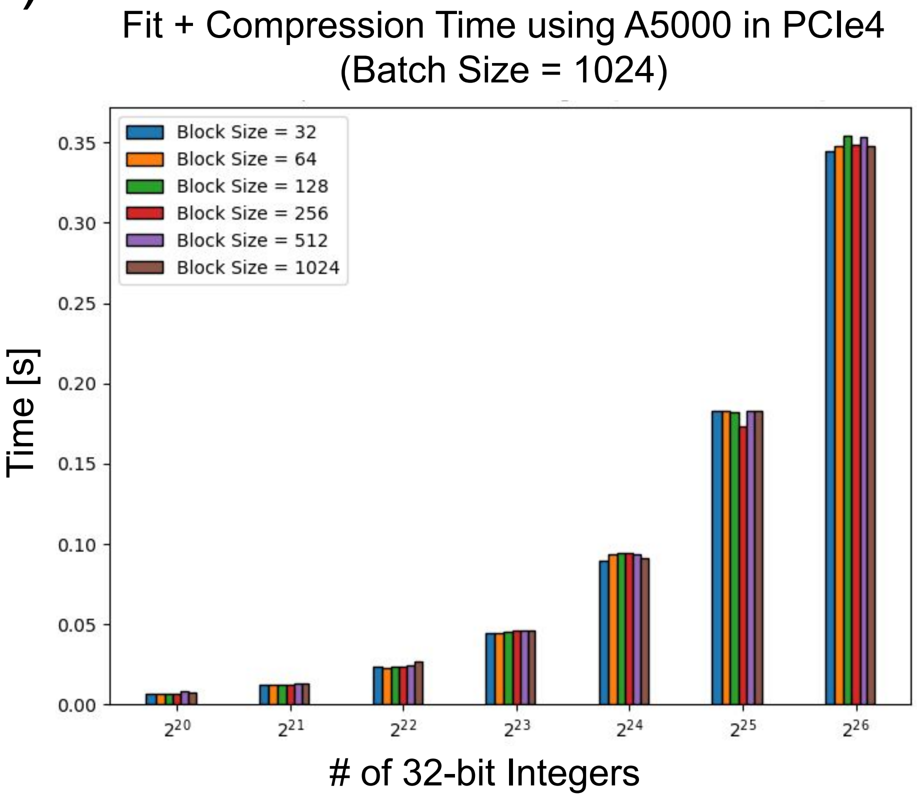
- We choose to let the FE's GPU and CPU handle compression for flexibility



GPU Benchmarking (Timings)

- Block Size:
  - A GPU parameter, number of threads per multiprocessor

- Can compress 2^{26} integers (32-bit) in roughly ⅓ of a second. → ~ **0.8 GB/s** compression rate

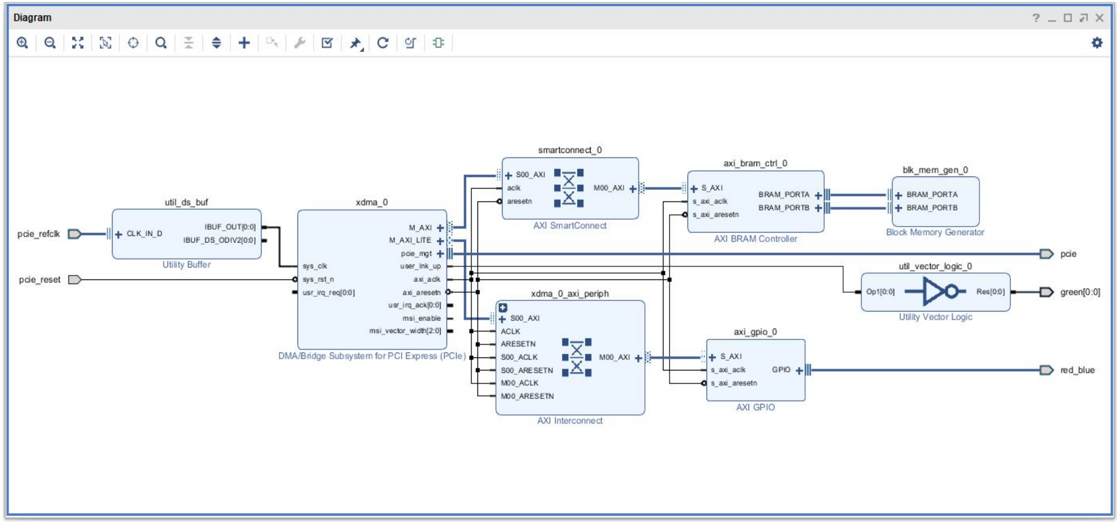


PCIe DMA Data Transfer

- Testing using a PCIe development board
  - Tested on PCIe2 x4

Nereid K7 PCI Express FPGA Development Board

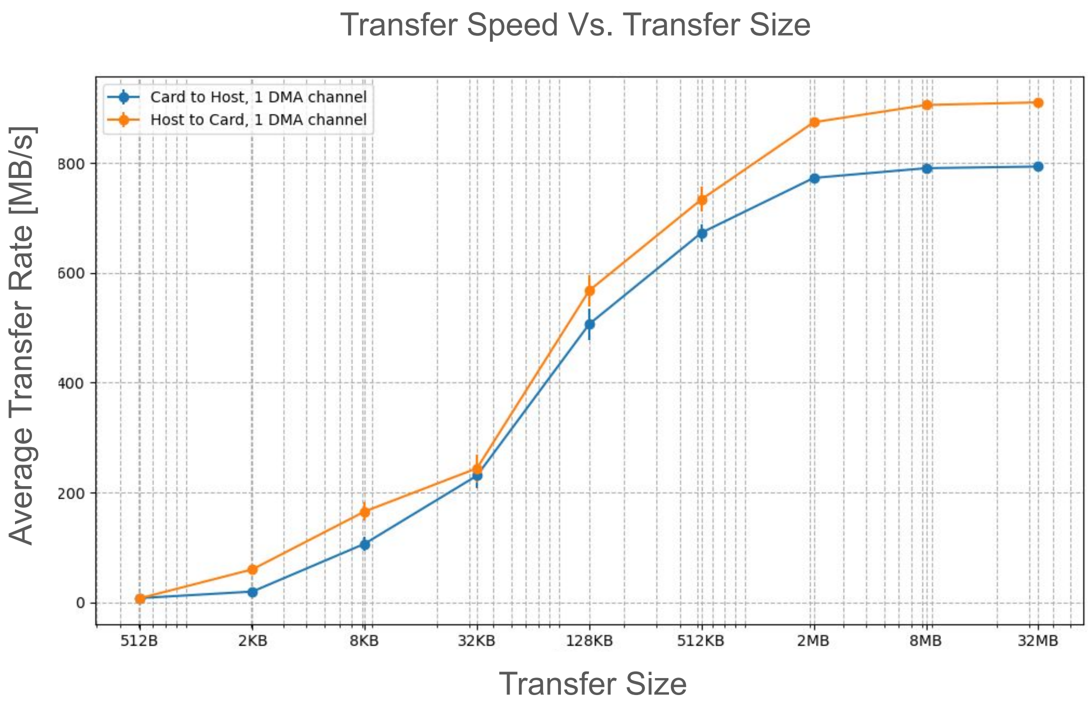
- Using Vivado IP blocks, we can create PCIe DMA design



Example block diagram (made in Vivado) for a PCIe FPGA

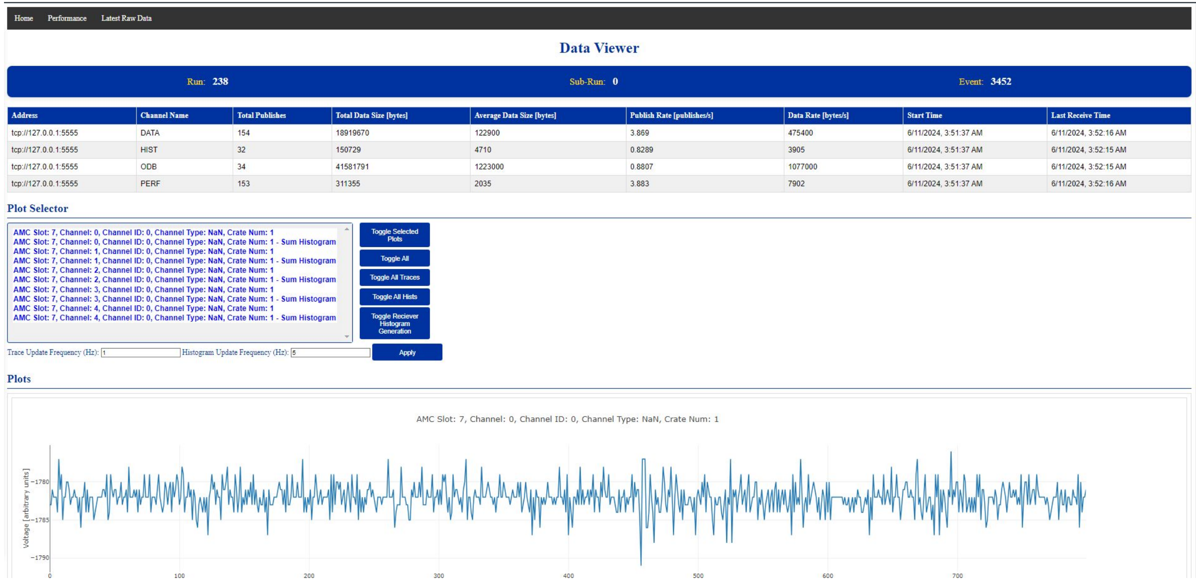
PCIe DMA Data Transfer

- Speeds here are limited by the board's transfer rate
  - Board can only handle 5GT/s (PCIe gen 2)
  - Expect faster for other boards
- Transfer rate ~1GB/s in ballpark of PIONEER rate (3.5 GB/s)
- Better to transfer in large packets



Software Development

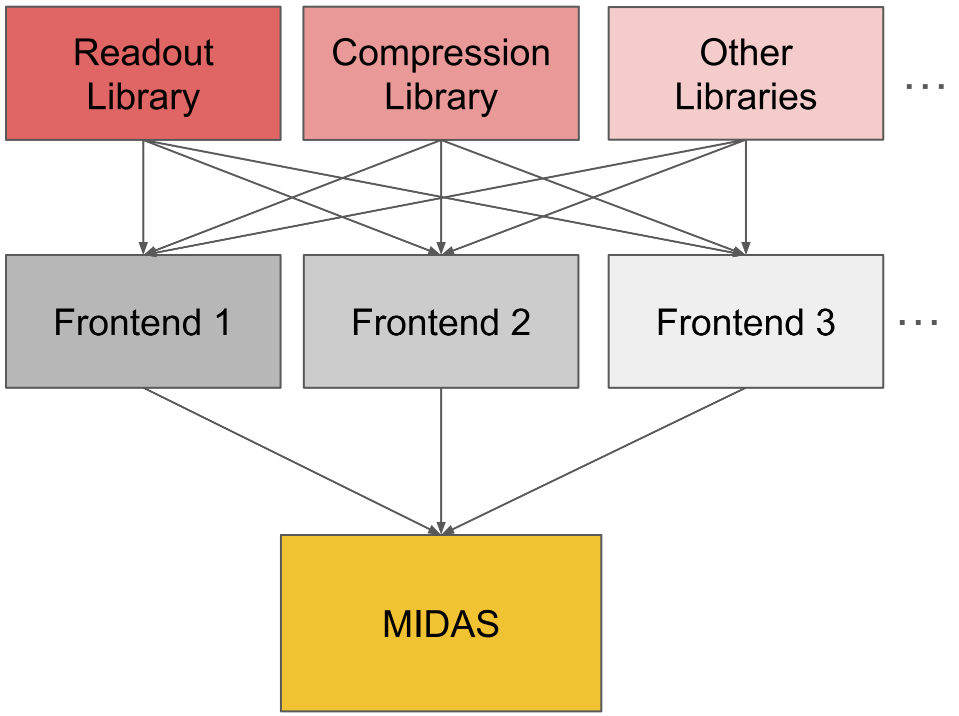
- Developed modular software working around midas
    - Useful for Calo test beam DAQ
    - Detached from Calo test beam DAQ, can be used with PIONEER DAQ
- Examples:
    - Midas Event Unpacker
    - Midas Event Publisher
    - Generalized DQM
    - Computer System Monitor



Generalized DQM Webpage

Software Development Plan

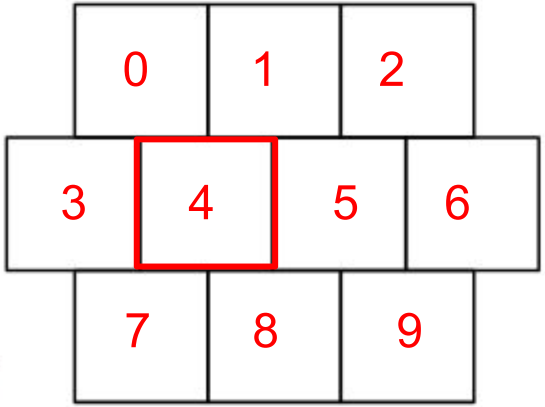
- Continue writing modular software
  - Will make experiment DAQ code much more manageable in the future
- Write PCIe readout libraries usable for PIONEER
- Write compression libraries usable for PIONEER
- Write midas frontend to read data out of FPGA over PCIe
  - Rate test, compression test



Auxiliary Slides

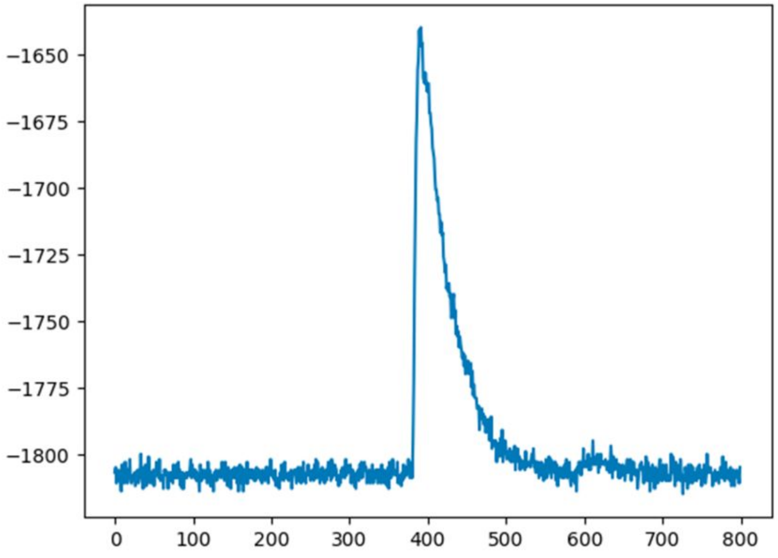
Data Set

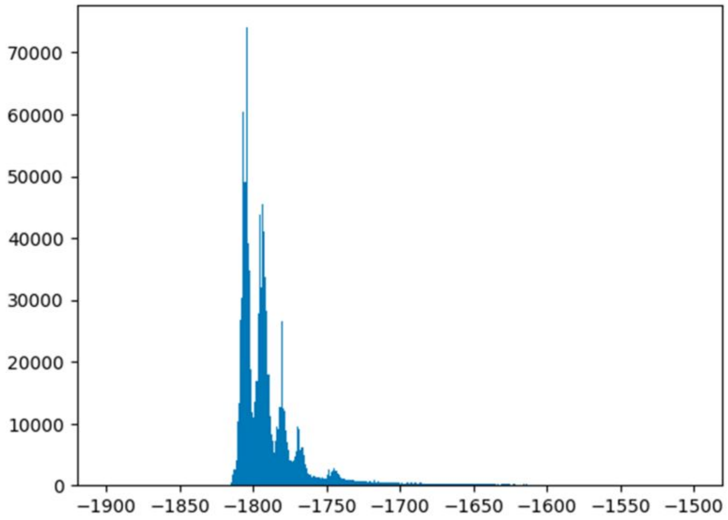
- PSI Test beam, Run 1887
- 70 MeV/c centered on LYSO crystal 4.
- The data only includes lyso channels (no NaI for instance)
- More details on that run are in this elog (https://maxwell.npl.washington.edu/elog/pienuxe/R23/124)



LYSO traces

- Select only LYSO channels and traces with a signal
- No pedestal subtraction, fitting, etc. (yet)





Entropy and Lossless Compression

- For lossless compression, the best possible compression rate is the entropy rate
- To first order, the entropy of an entire trace is:

$$H(X_1, ..., X_n) = -\sum_{\text{traces}} p(X_1, ..., X_n) \log_2(p(X_1, ..., X_n))$$

- X_i is the random variable for the ADC value of the ith sample in the trace with n samples
- If we assume X_i independent, then

$$H(X_1, ..., X_n) = H(X_1) + ... + H(X_n)$$

- By transforming ($X_i \rightarrow$ fit residuals), X_i becomes approximately independent

Citation: Coding and Information Theory Class Notes, Dr. Jay Weitzen, University of Massachusetts Lowell
https://faculty.uml.edu/jweitzen/16.548/classnotes/Theory%20of%20Data%20Compression.htm#:~:text=When%20the%20compression%20is%20lossless,rate%20is%20the%20entropy%20rate

Higher Order Entropy Estimations

- Assume we have N characters (traces) in our alphabet (data set)
- **Zero order:** each character in alphabet is statistically independent

$$H = \log_2(N)$$

- **First order:** each character in alphabet is statistically independent, p_i is the probability of that character to occur

$$H = -\sum_{i=1}^{N} p_i \log_2(p_i)$$

- **Second order:** $P_{j|i}$ is correlation between subsequent characters

$$H = -\sum_{i=1}^{N} p_i \sum_{j=1}^{N} P_{j|i} \log_2(P_{j|i})$$

- **General Model (impractical):** B_n represents the first n characters

$$H = \lim_{n\to\infty} \left[-\frac{1}{n} \sum p(B_n) \log_2(B_n) \right]$$

Citation: Joint Entropy, Wikipedia
https://en.wikipedia.org/wiki/Joint_entropy

Joint Entropy, Mutual Information

$$H(X_1, ..., X_n) \leq H(X_1) + ... + H(X_n)$$

Equality only holds if

$$X_1, ..., X_n \text{ are mutually statistically independent}$$

This means if

$$I(X_1, X_2) = H(X_1) + H(X_2) - H(X, Y) = 0$$

Then we must have X_1 and X_2 be statistically independent

Citation: Proof from stackexchange
https://math.stackexchange.com/questions/3519345/joint-entropy-of-2-independent-random-variables

Joint entropy for Independent Variables Proof

Statement:

$H(X_1, ..., X_n) = \sum_{i=1}^{n} H(X_i)$

Proof (part 1):

$$H(X_1, ..., X_n) = -\sum_{x_1, ..., x_n} P(x_1, ..., x_n) \log_2(P(x_1, ..., x_n))$$

$$= -\sum_{x_1, ..., x_n} P(x_1)...P(x_n) \left(\log_2(P(x_1)) + ... + \log_2(P(x_n))\right)$$

(Note: I am lazy, each $P(x_i)$ represents a different pdf in general)

Citation: Proof from stackexchange
https://math.stackexchange.com/questions/3519345/joint-entropy-of-2-independent-random-variables

Joint entropy for Independent Variables Proof

Proof (part 2):

$$H(X_1, ..., X_n) = -\left(\sum_{x_1} P(x_1)\log_2(P(x_1))\right)\left(\sum_{x_2} P(x_2) \cdot ... \cdot \sum_{x_n} P(x_n)\right)$$

$$- ...$$

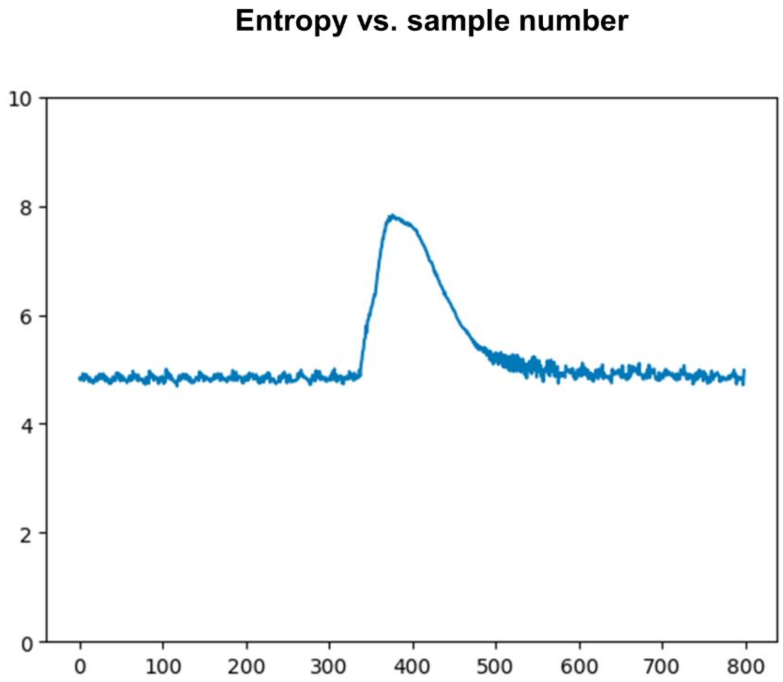
$$- \left(\sum_{x_1} P(x_1) \cdot ... \cdot \sum_{x_{n-1}} P(x_{n-1})\right)\left(\sum_{x_n} P(x_n)\log_2(P(x_n))\right)$$

$$\text{Note } \sum_{x_i} P(x_i) = 1 \text{ and } \sum_{x_1} P(x_i)\log_2(P(x_i)) = H(X_i)$$

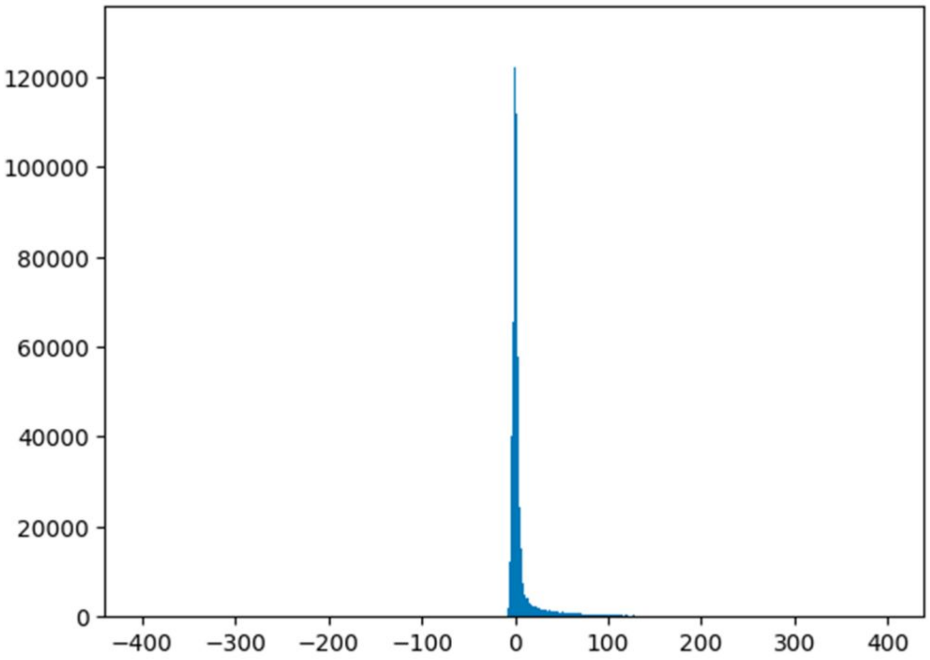
$$= H(X_1) + ... + H(X_n) \blacksquare$$

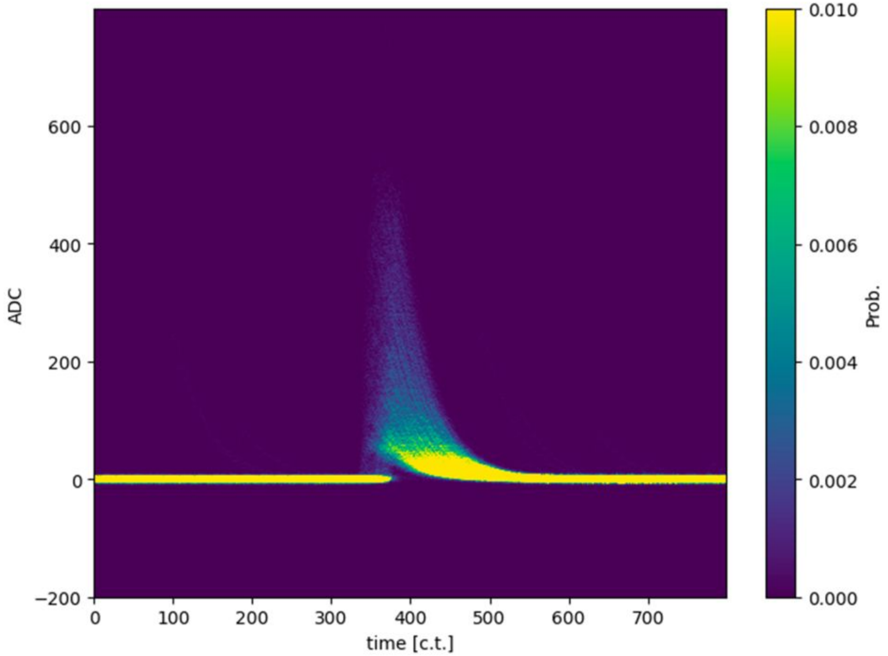
Entropy estimation

- Average entropy per bit: 5.22 bits / sample (compare to 16 bits for a short)
- Samples near waveform edge have lower entropy
- Samples near middle have higher entropy, due to the pulses
- Entropy is nonzero b/c the waveforms are **not** identical: difference pedestals, different pulse sizes

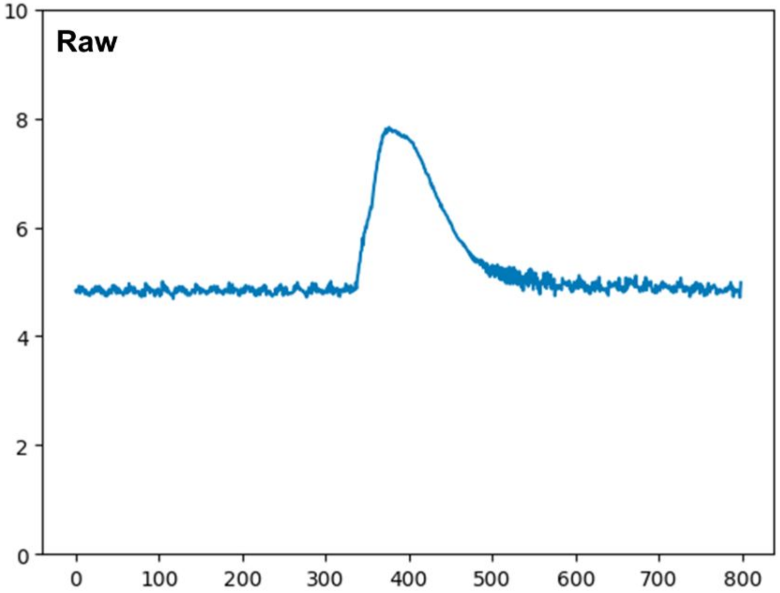


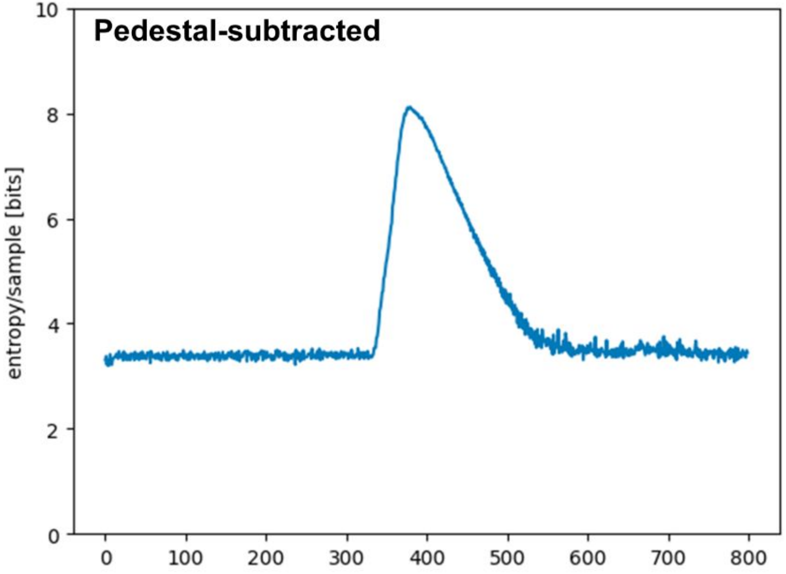
Pedestal subtracted





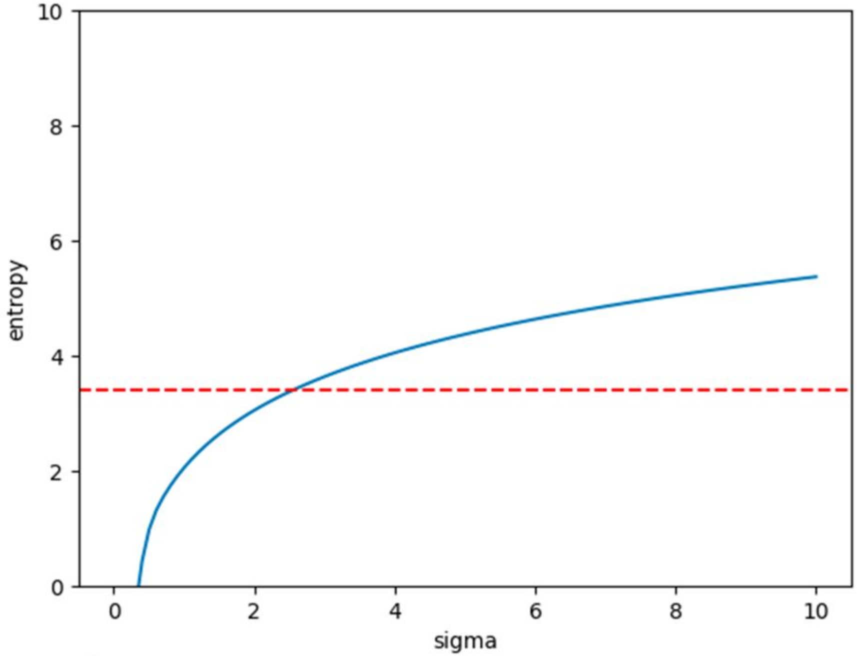
Entropy estimation

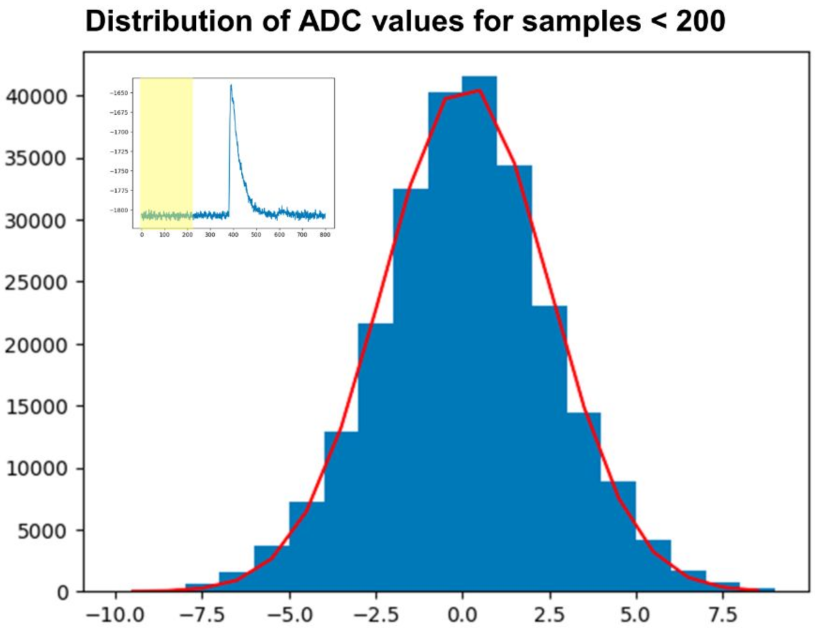




- Entropy reduced for samples near waveform edge: ~3.4 bits
- Average entropy per sample now: 4.05 bits/sample

Discrete Gaussian entropy

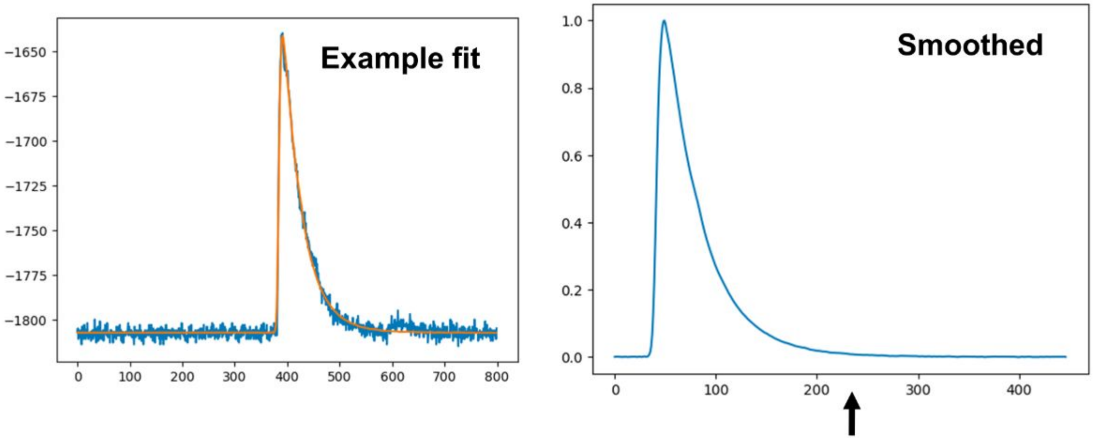


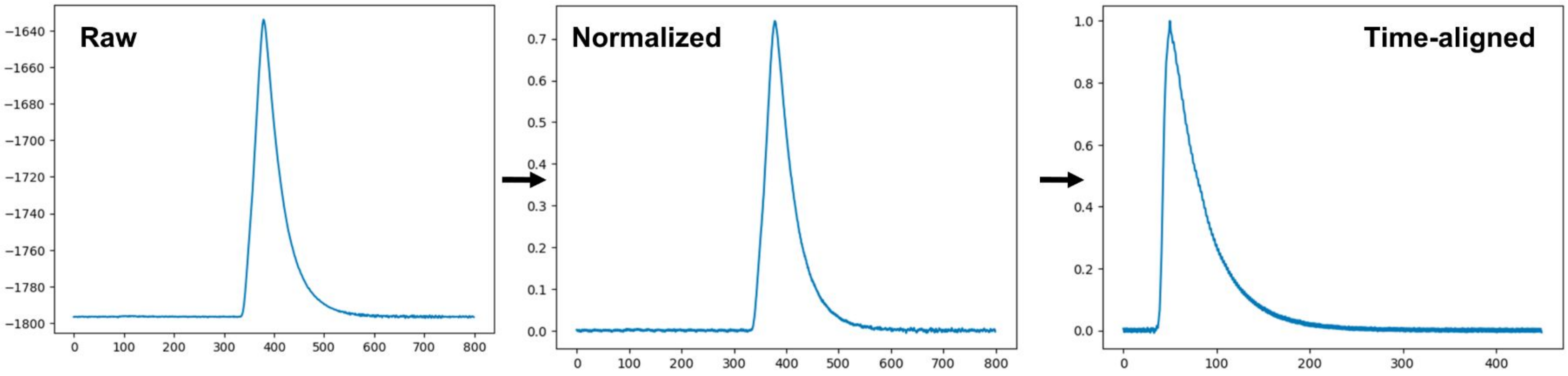


- If we assume gaussian noise: entropy of 3.4 bits -> $\sigma = 2.6$
- If we look at samples < samples number 200 and fit ADC to gaussian: $\sigma = 2.4$

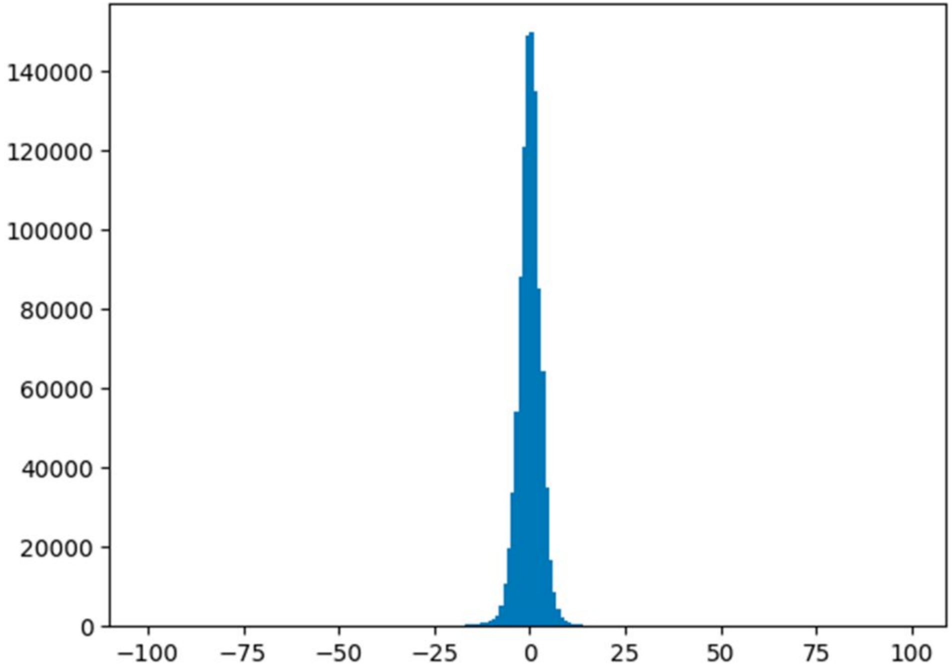
Template fit

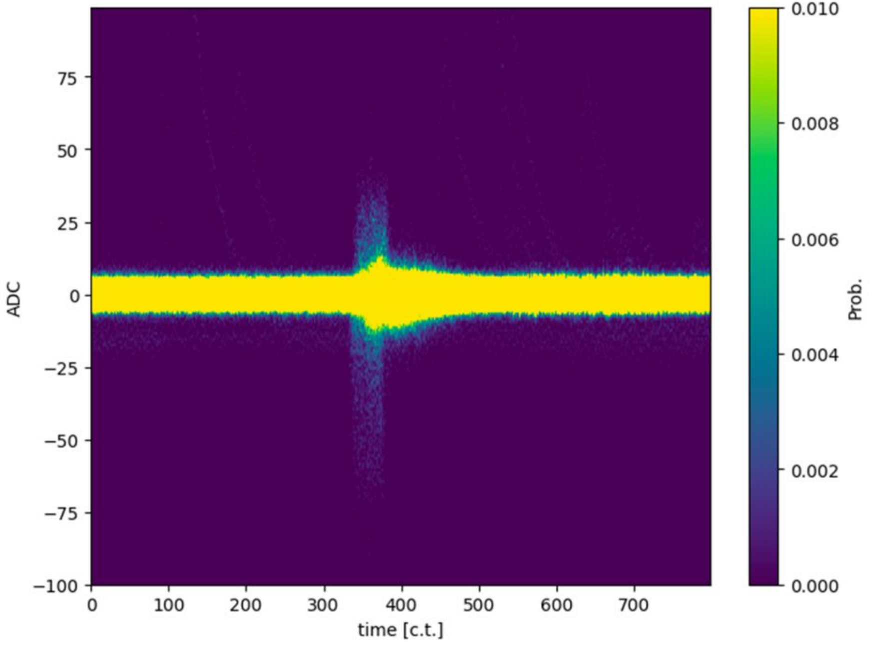
- Constructing a template
  - Normalized all traces
  - Time-align the peak
  - Smooth over adjacent sample
  - Fit with $f(t) = A \cdot T(t - t_0) + C$



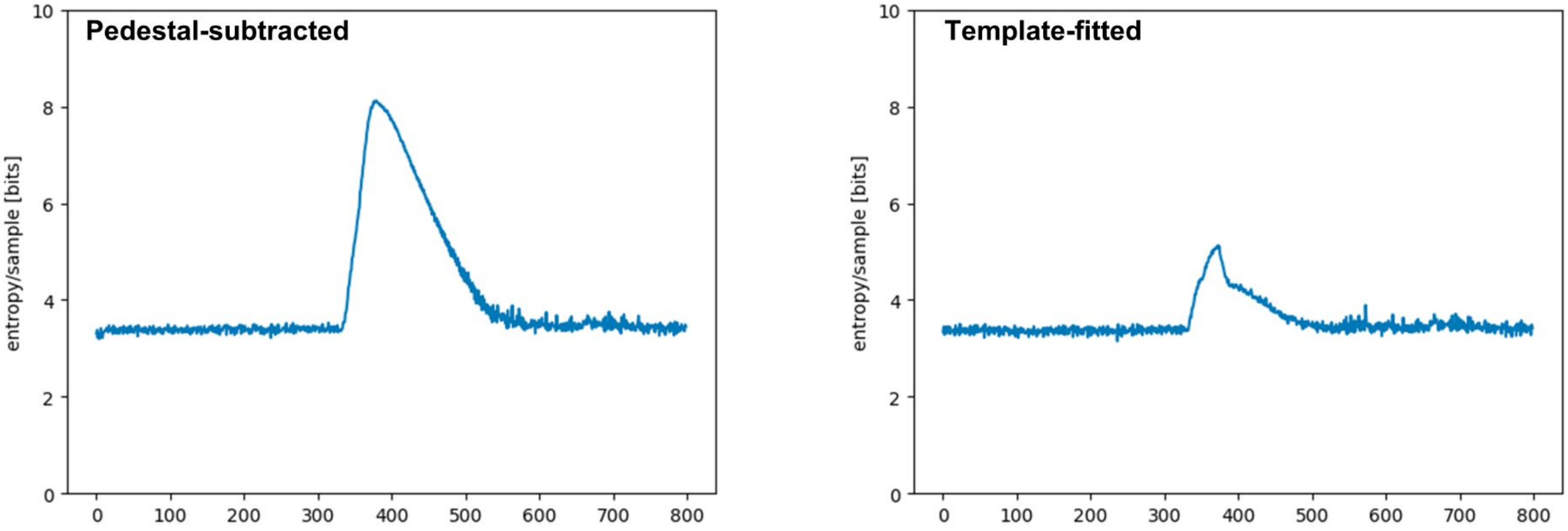


Template fit





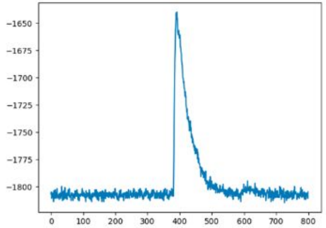
Entropy estimation



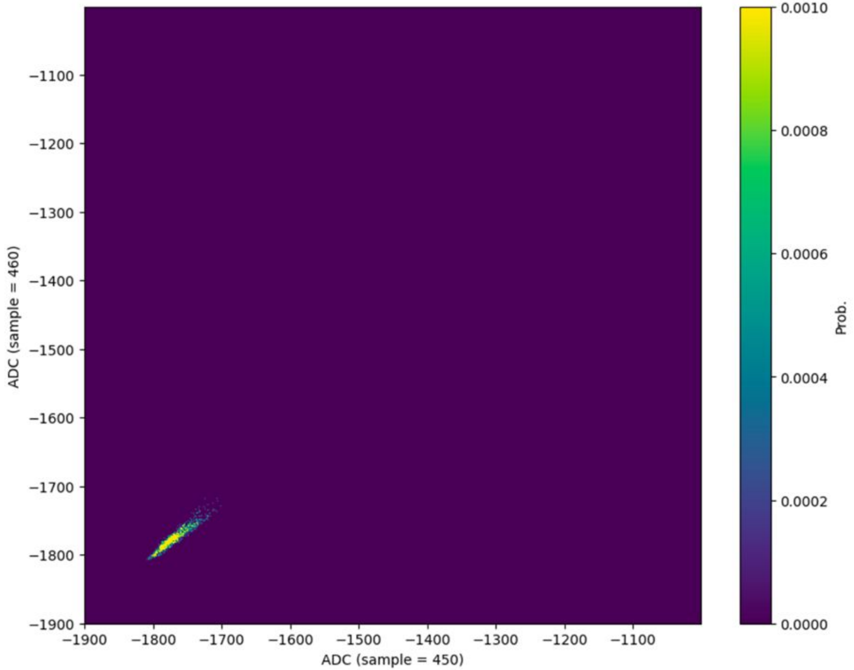
- Baseline hasn't changed much. Makes sense since fluctuations remain
- Peak in middle is reduced, but evidently we can still do better
- Average entropy per sample now: **3.55 bits/sample**

Citation: Entropy estimates of PSI test beam data, Sean Foster (slides)

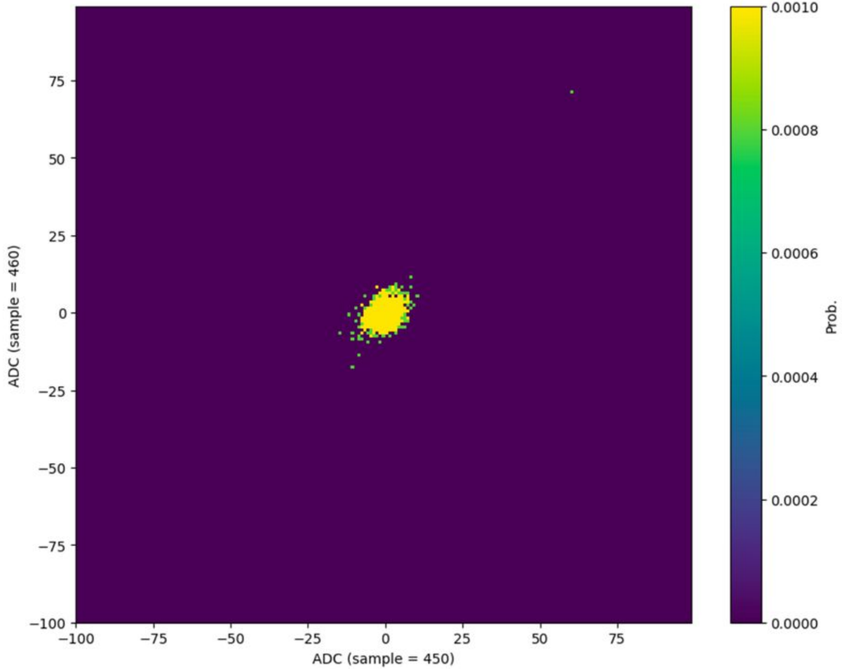
Correlations



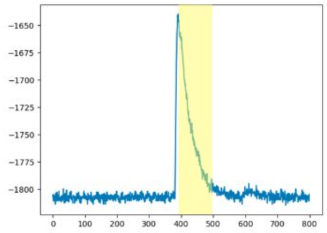
Raw



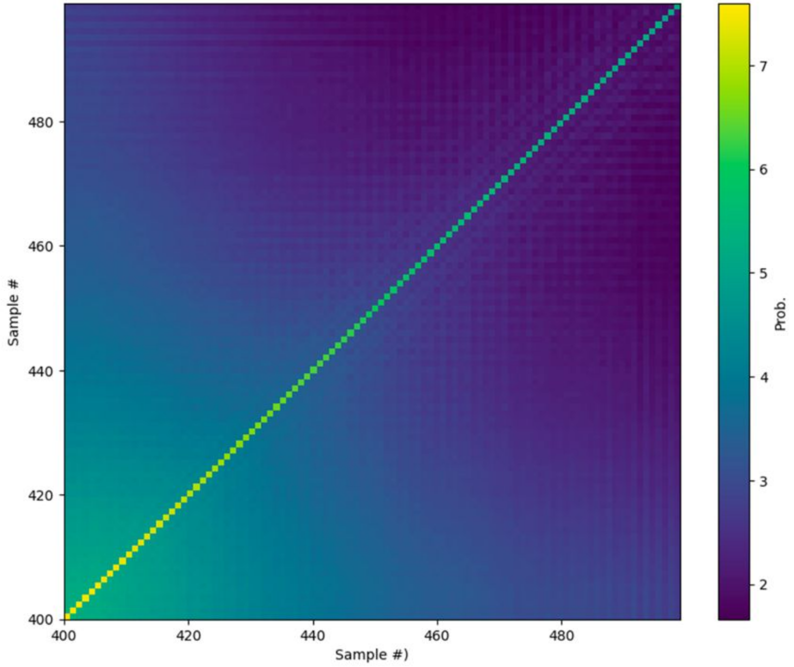
Templated-fitted



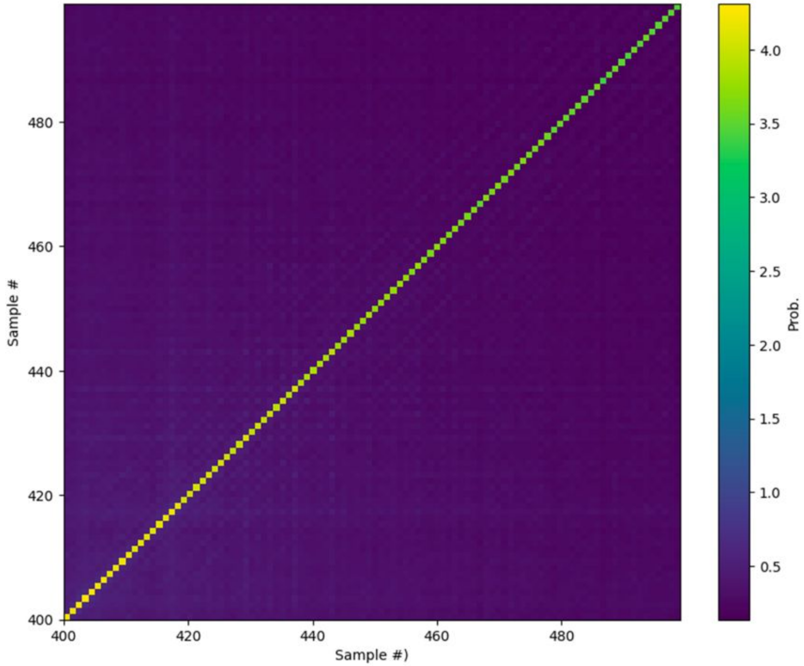
Mutual Information



Raw



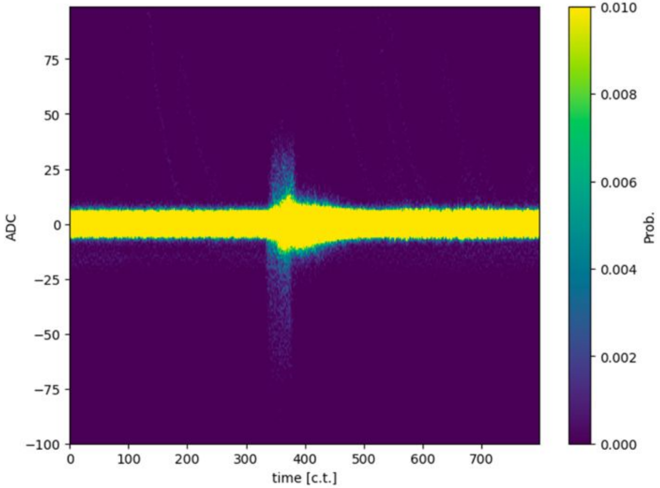
Templated-fitted

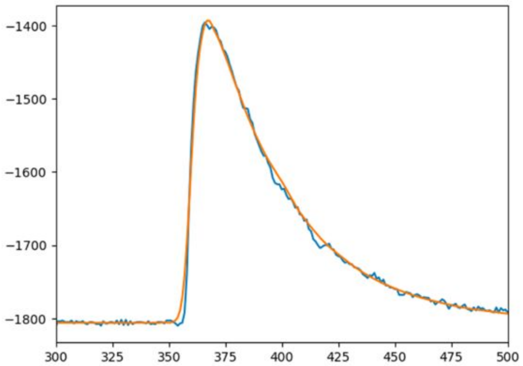


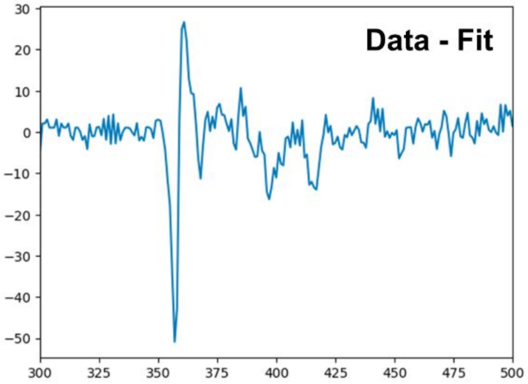
$H(X) + H(Y) - H(X,Y)$ **nonzero means there are still correlations**

Template fitting going wrong

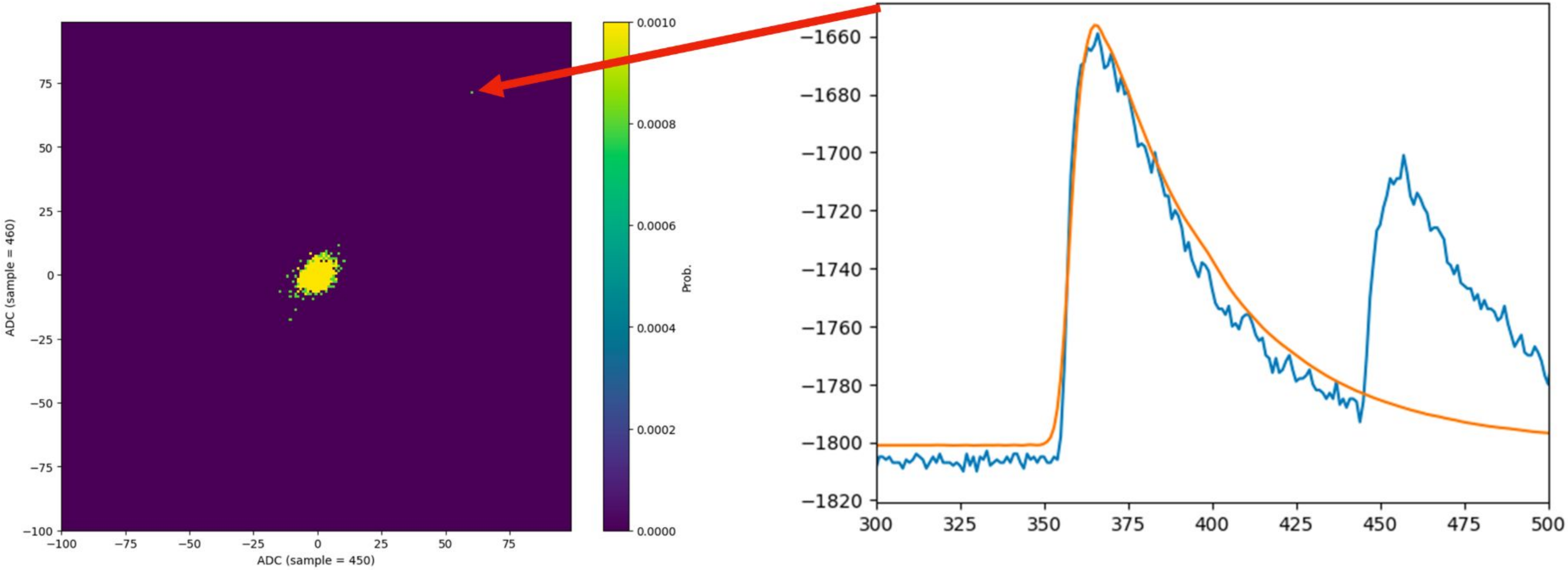
- What's causing the spread at the start of the pulse ~360 c.t. or so? (right plot)
- Seems like my template fit going wrong at the pulse turn-on







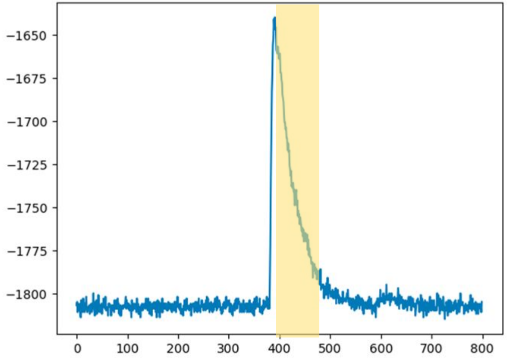
Stray point due to pileup

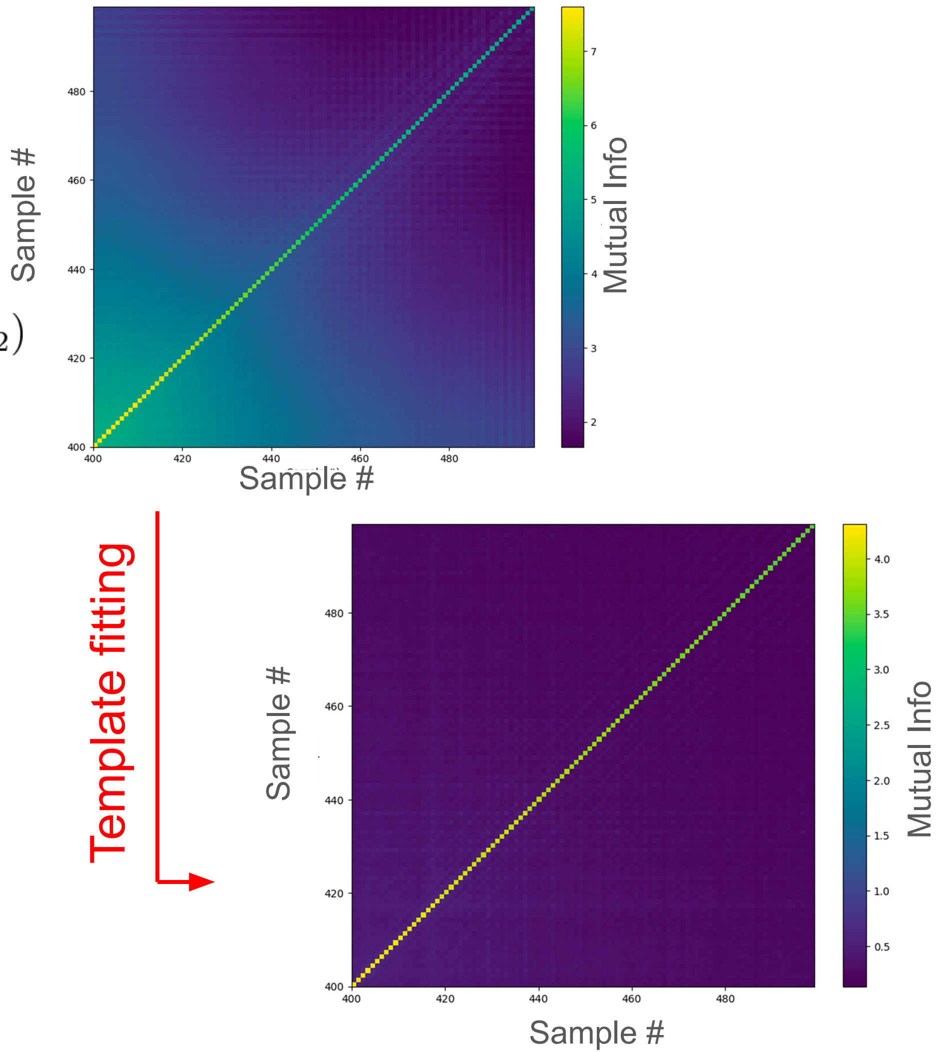


Citation: Entropy estimates of PSI test beam data, Sean Foster (slides)

Mutual Information

- Mutual Information:
  $I(X_1, X_2) = H(X_1) + H(X_2) - H(X_1, X_2)$
- $I(X_1, X_2) = 0 \implies$ no correlation
- Template fitting reduces correlations between subsequent samples





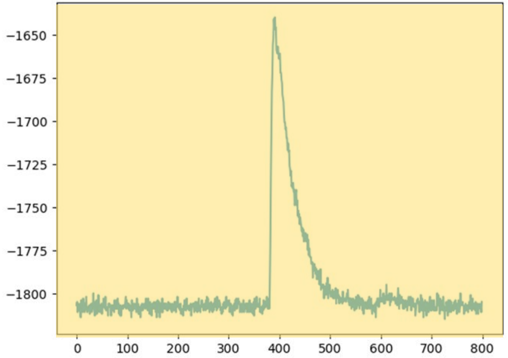
Citation: Entropy estimates of PSI test beam data, Sean Foster (slides)

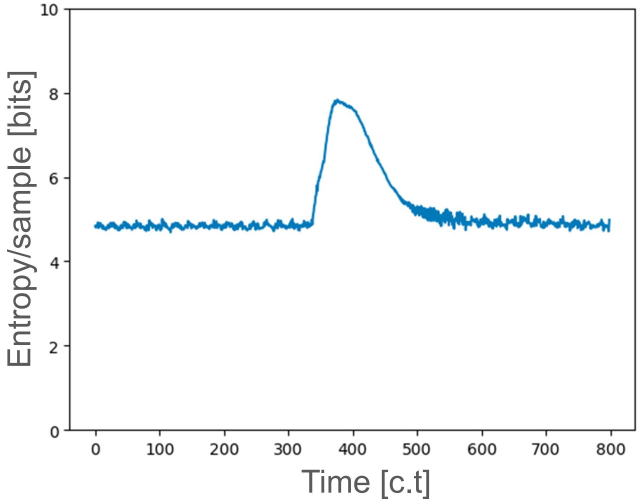
Entropy Estimation

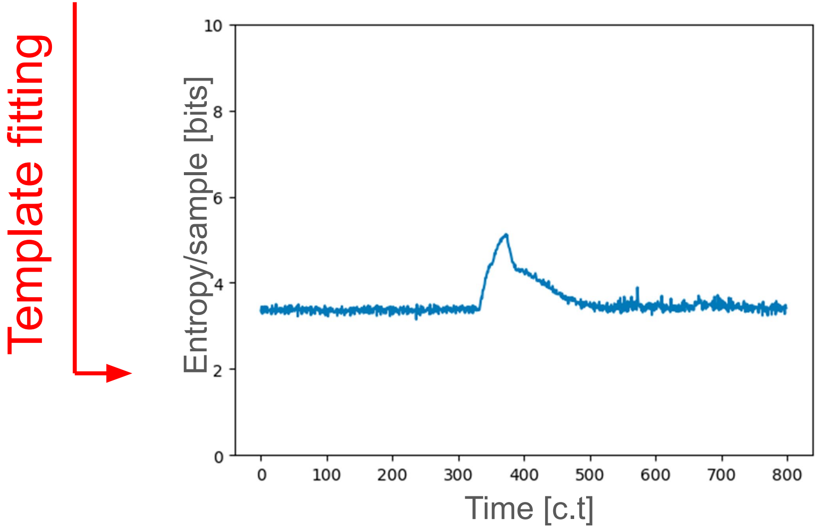
- Average entropy:

$$H_{\text{avg}} = \frac{\sum_{i=1}^{N} H(X_i)}{N}$$

- In this case N = 800
- Before: H_{avg} = 5.22 bits/sample
- After: **H_{avg} = 3.55 bits/sample**
- Some room for improvement(?)

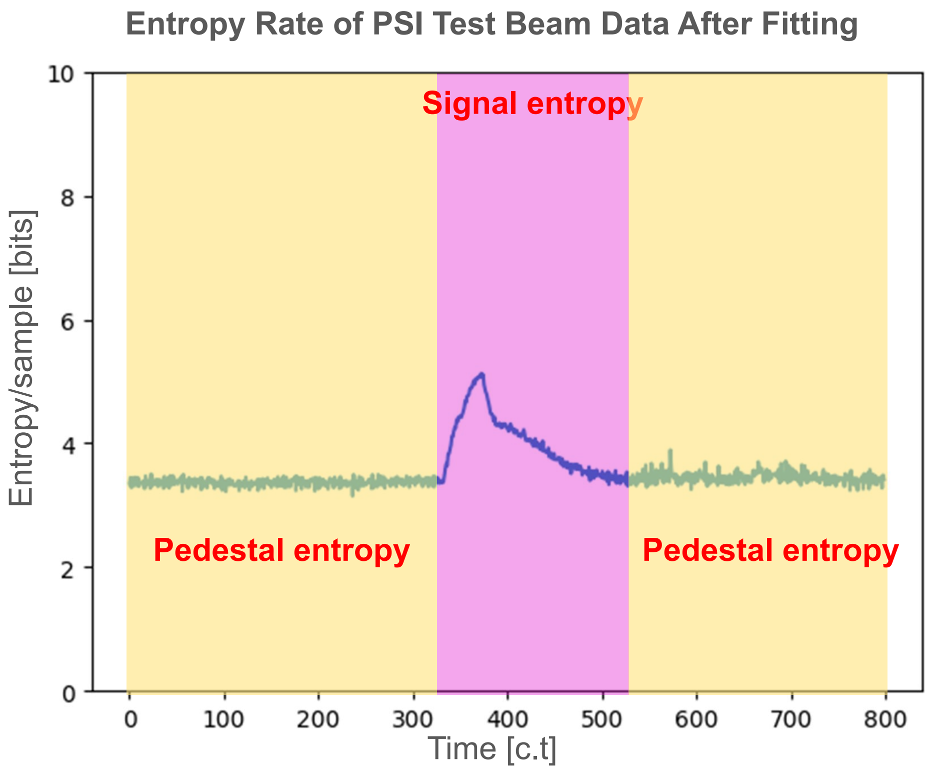






Explanation of Entropy Plot

- The pedestal is easy to fit, so the variance of the pedestal part of the signal is is just the noise of the WFD5s.
  - This is the minimum possible entropy when using this equipment
- The signal is harder to fit and therefore has more variance
  - Entropy of this part of the trace is therefore larger



Theoretical Best Compression Calculation

Assuming data is sent as 12 bit ADC samples over PCIe at a data rate of 3.5 GB/s:

$$\text{Compression Ratio} = \frac{\text{Entropy Rate}}{12}$$

$$\text{Storage Data Rate} = \text{Compression Ratio} \cdot 3.5\ \text{GB/s}$$

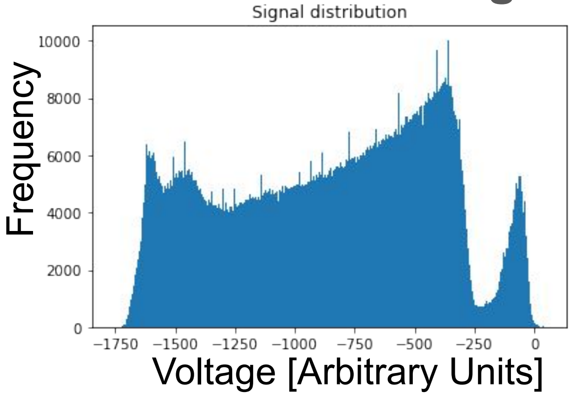
Entropy rate = 3.4 → New Data Rate ≈ 0.99 GB/s

Entropy rate = 5 → New Data Rate ≈ 1.46 GB/s

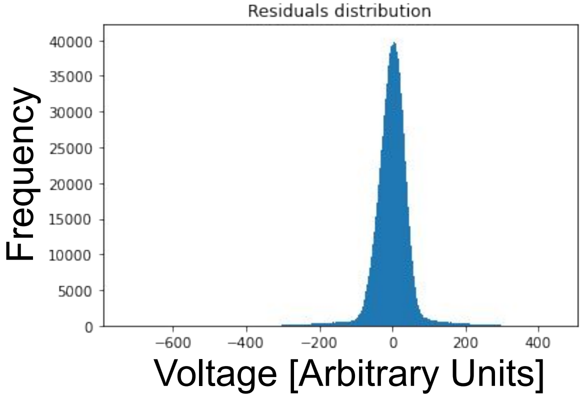
Signal Conditioning

- Want a narrow distribution for compression. Let r_i be the numbers we compress
- Methods tried:
  - No conditioning
  - Delta encoding:
    $r_i = y_{i+1} - y_i$
  - Twice Delta Encoding:
    $r_i = y_{i+2} - 2y_{i+1} + y_i$
  - Double Exponential Fit:
    $r_i = y_i - (A \cdot \exp(at_i) + B \cdot \exp(bt_i))$
  - **Shape Fit**:
    $\mathbf{r_i = y_i - (A \cdot T(t_i - t_0) + B)}$

No Conditioning



Shape Fit



Shape Fitting Algorithm

1. Construct a discrete template from sample pulses
2. Interpolate template to form a continuous Template, *T(t)*
3. “Stretch” and “shift” template to match signal:

$$X[i] = a(t_0)T(t[i] - t_0) + b(t_0)$$

   [Note: a and b can be calculated explicitly given t_0]

4. Compute χ^2 (assuming equal uncertainty on each channel i)

$$\chi^2 \propto \sum_i \{X[i] - a(t_0)T(t[i] - t_0) + b(t_0)\}^2$$

5. Use Euler’s method to minimize χ^2

Lossless Compression Algorithm

- Rice-Golomb Encoding
  - Let x be number to encode
    y = “s”+“q”+”r”
    - q = x/M (unary)
    - r = x%M (binary)
    - s = sign(x)
  - Any distribution
  - Close to optimal for valid choice of M
  - One extra bit to encode negative sign
  - Self-delimiting
  - If quotient too large, we “give up” and write x in binary with a “give up” signal in front

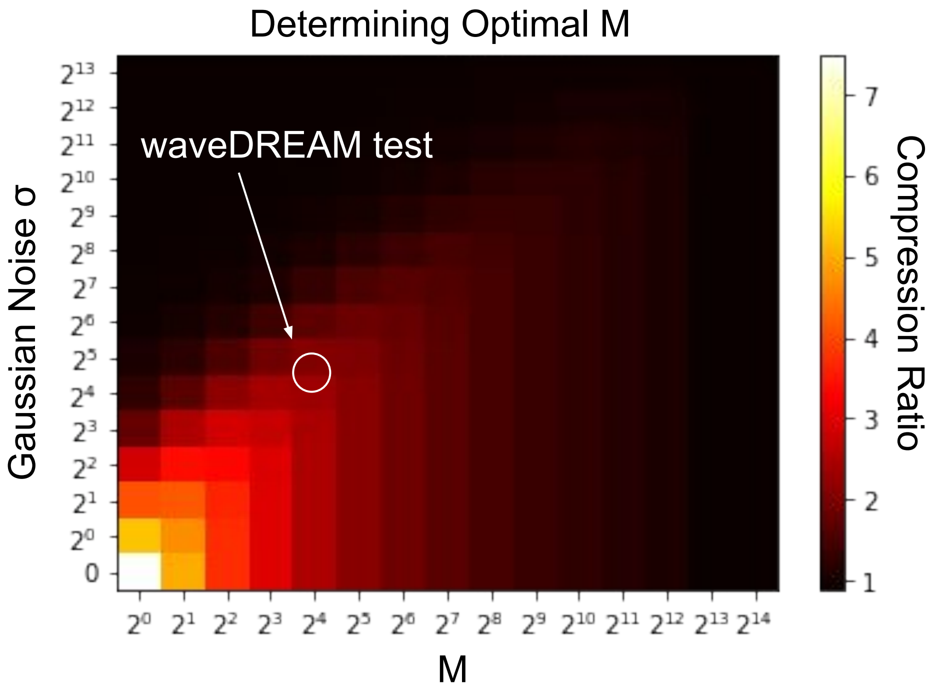
Rice-Golomb Encoding (M=2)

Value	Encoding
-1	011
0	000
1	001
2	1000

Red = sign bit
Blue = quotient bit(s) (Unary)
Yellow = remainder bit (binary)

How to choose Rice-Golomb parameter M

- Generated fake Gaussian data (centered at zero) with variance σ^2
- For random variable X, $M \approx \text{median}(|X|)/2$ is a good choice
  - This is the close to the diagonal on the plot
- $\sigma \approx 32$ for residuals of shape on wavedream data → M = 16 is a good choice

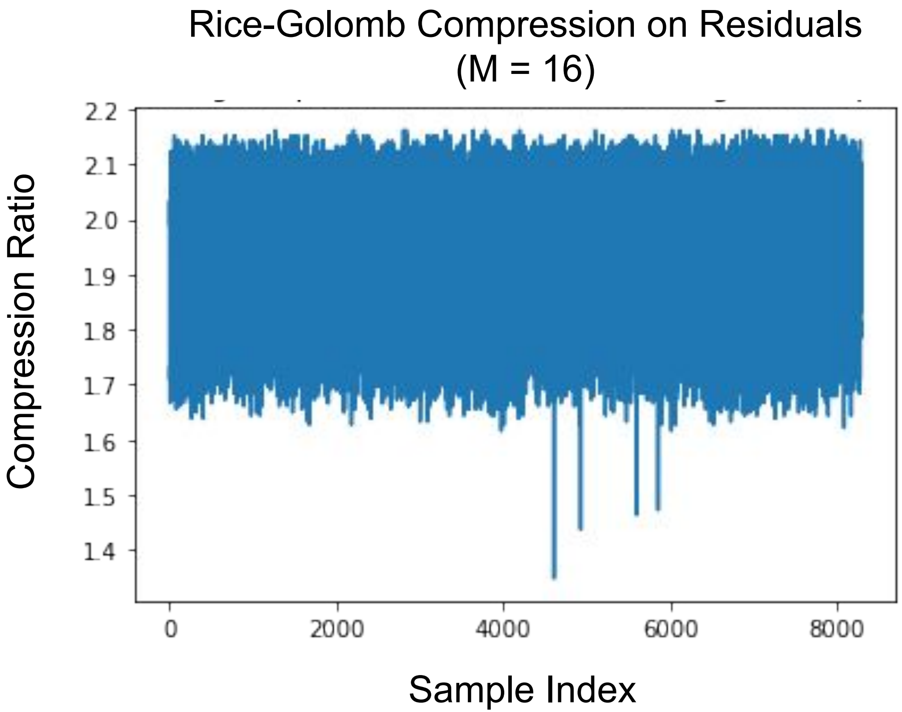


Compression Ratio from Rice-Golomb Encoding

- Lossless compression factor of ~2

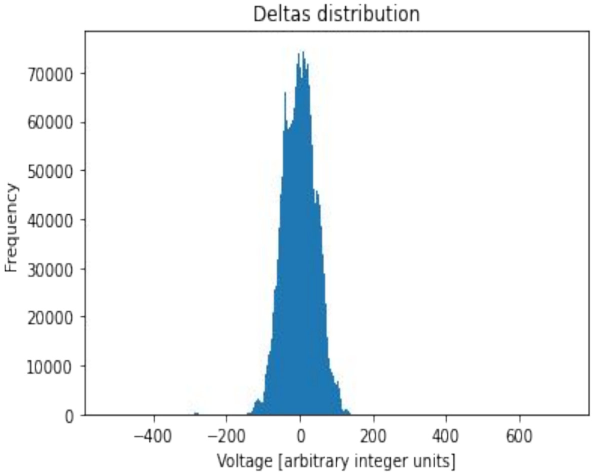
- In agreement with plot from simulated data on last slide

- Data is much noisier than than PSI test beam, so we get a smaller compression factor

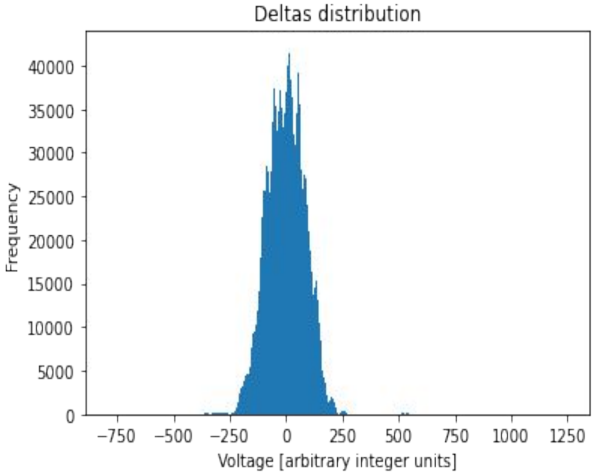


Other Conditioning Distributions

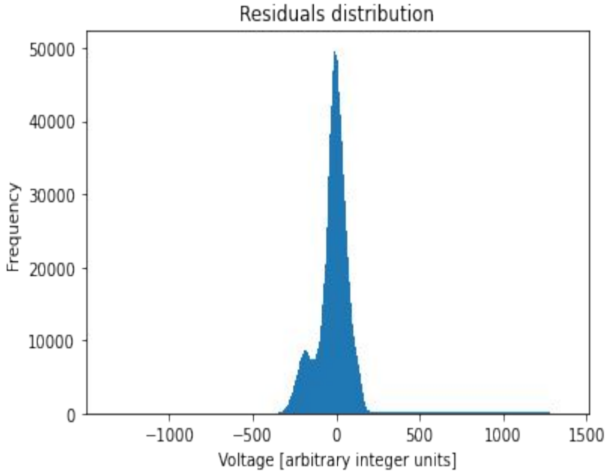
Delta Encoding



Twice Delta Encoding



Double Exponential Fit



Shape Fitting Details

Fit Function

$$X[i] = aT(t[i] - t_0) + b$$

Explicit $a(t_0)$ calc

$$a(t_0) = \frac{\sum_i^N X[i] \sum_i^N T(t[i]-t_0)^2 - \sum_i^N T(t[i]-t_0) \sum_i^N T(t[i]-t_0) X[i]}{N \sum_i^N T(t[i]-t_0)^2 - (\sum_i^N T(t[i]-t_0))^2}$$

Explicit $b(t_0)$ calc

$$b(t_0) = \frac{N \sum_i^N T(t[i]-t_0) X[i] - \sum_i^N T(t[i]-t_0) \sum_i^N X[i]}{N \sum_i^N T(t[i]-t_0)^2 - (\sum_i^N T(t[i]-t_0))^2}$$

Explicit χ^2 calc

$$f(t_0) \equiv \chi^2 \propto \sum_i \{X[i] - a(t_0)T(t[i]-t_0) + b(t_0)\}^2$$

Newton's method

$$(t_0)_{n+1} = (t_0)_n - \frac{f'((t_0)_n)}{f''((t_0)_n)}$$

Threshold requirement

$$|(t_0)_{n+1} - (t_0)_n| < \epsilon \equiv \text{"Threshold"}$$

Citation: Wikipedia
(https://en.wikipedia.org/wiki/Golomb_coding)

Golomb Encoding

- In general, M is an arbitrary choice
- Since computers work with binary, $M = 2^x$ such that x is an integer is a "fast" choice
  - This is called Rice-Golomb Encoding
- Self delimiting so long as the information M is provided

Golomb Encoding Example

Choose M = 10, $b = \log_2(M) = 3$
$2^{b+1} - M = 16 - 10 = 6$
r < 6 → r encoded in b=3 bits
r ≥ 6 → r encoded in b+1=4 bits

Encoding of quotient part	
q	output bits
0	0
1	10
2	110
3	1110
4	11110
5	111110
6	1111110
⋮	⋮
N	111⋯1110

Encoding of remainder part			
r	offset	binary	output bits
0	0	0000	000
1	1	0001	001
2	2	0010	010
3	3	0011	011
4	4	0100	100
5	5	0101	101
6	12	1100	1100
7	13	1101	1101
8	14	1110	1110
9	15	1111	1111

Citation: Wikipedia
(https://en.wikipedia.org/wiki/Huffman_coding)

Huffman Encoding

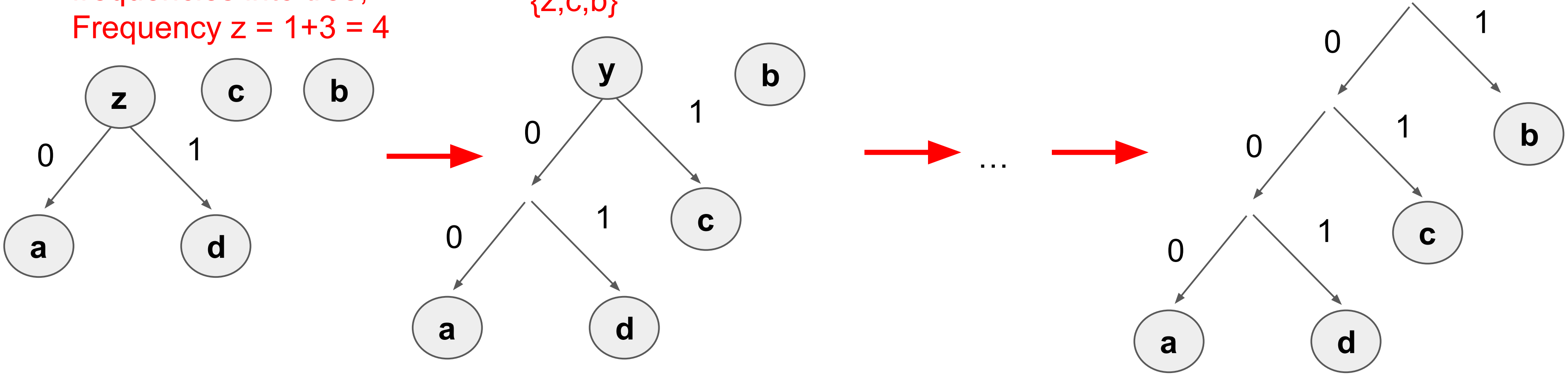
- Requires finite distribution
- Values treated as “symbols”
- Self-delimiting (sometimes called “greedy”)

Huffman Encoding Example

Value	Frequency	Encoding
-1 ≡ a	1	000
0 ≡ b	10	1
1 ≡ c	5	01
2 ≡ d	3	001

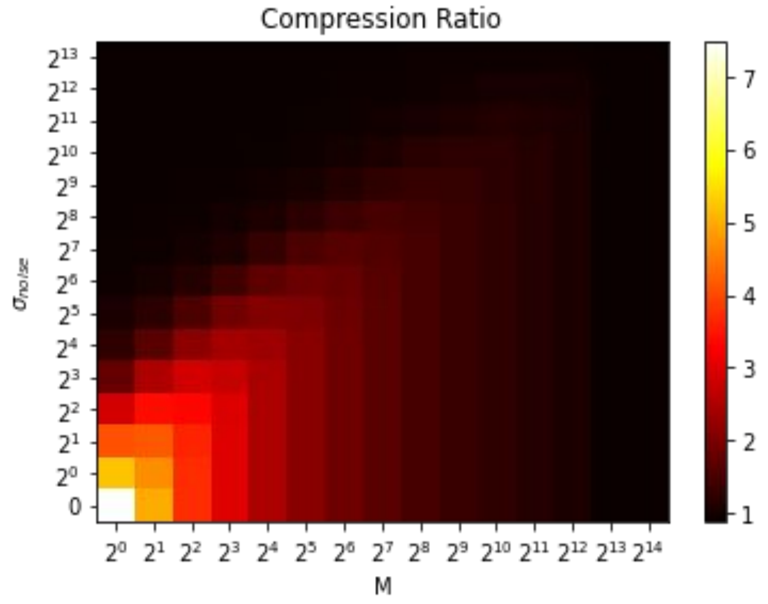
“Combine” two lowest frequencies into tree, Frequency z = 1+3 = 4

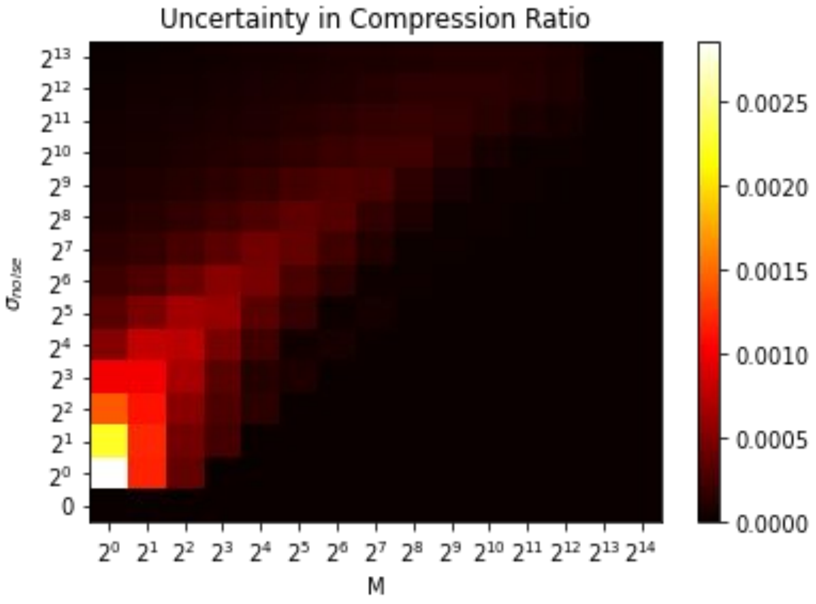
Repeat for set {z,c,b}



Theoretical Uncertainty in Compression Ratio from Gaussian Noise

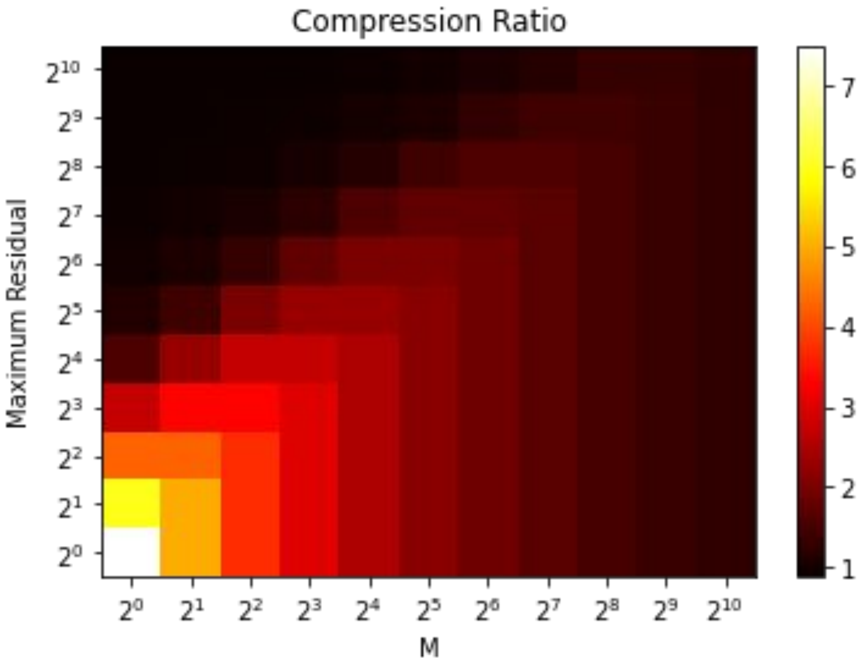
- ~ 0.1% relative error



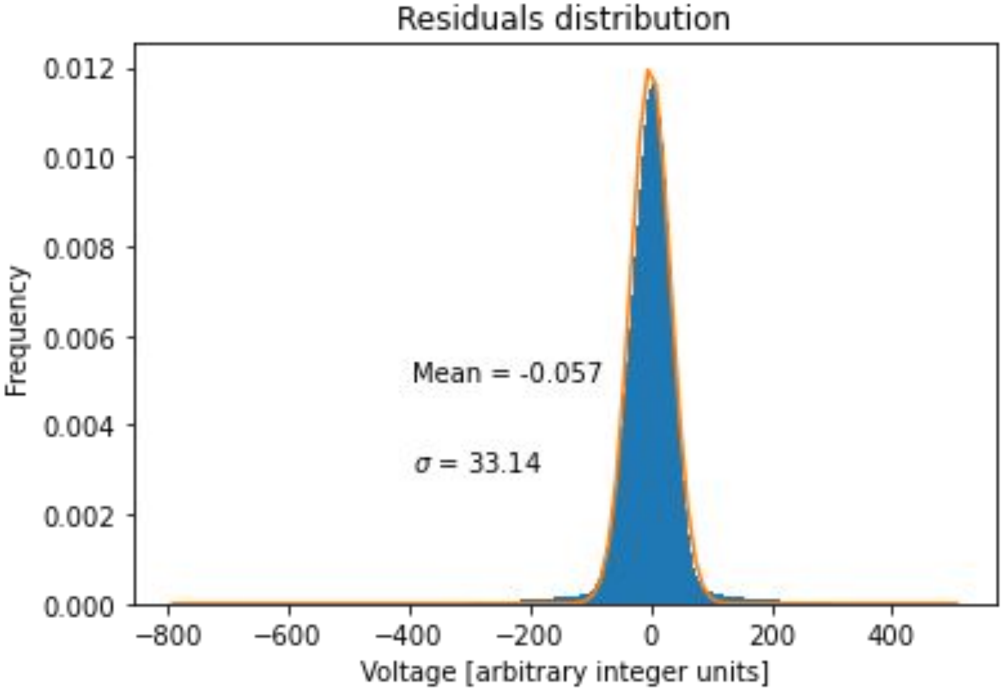


Uniform Distribution of Noise effect on Compression Ratio

- Here instead we use a uniform distribution to generate the noise
- Not much different than gaussian noise, same conclusions really

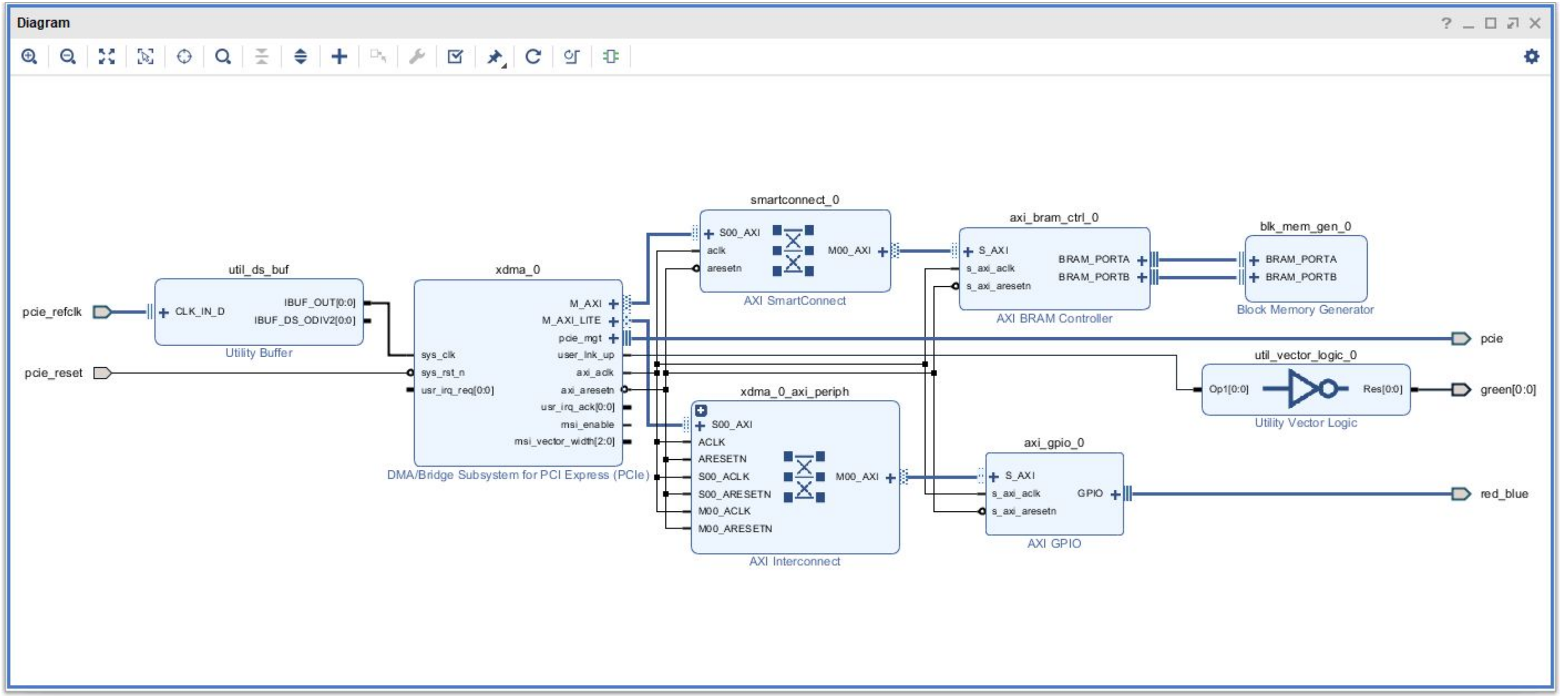


Residuals Distribution and Optimal M



M	Compression Ratio
1	1.04721105
2	1.21287474
4	1.53114598
8	1.92616642
16	**2.09307249**
32	2.02975311
64	1.86037914
128	1.66627451
. . .	. . .

PCIe DMA Block Diagram in Vivado



Example block diagram (made in Vivado) for a PCIe FPGA

PCIe Transfer Speeds for Different Generations

Nereid Test

VERSION	INTRODUCTION YEAR	LINE CODE	TRANSFER RATE	THROUGHPUT				
				x1	x2	x4	x8	x16
1	2003	8b/10b	2.5 GT/s	0.250 GB/s	0.500 GB/s	1.000 GB/s	2.000 GB/s	4.000 GB/s
2	2007	8b/10b	5.0 GT/s	0.500 GB/s	1.000 GB/s	2.000 GB/s	4.000 GB/s	8.000 GB/s
3	2010	128b/130b	8.0 GT/s	0.985 GB/s	1.969 GB/s	3.938 GB/s	7.877 GB/s	15.754 GB/s
4	2017	128b/130b	16.0 GT/s	1.969 GB/s	3.938 GB/s	7.877 GB/s	15.754 GB/s	31.508 GB/s
5	2019	128b/130b	32.0 GT/s	3.938 GB/s	7.877 GB/s	15.754 GB/s	31.508 GB/s	63.015 GB/s
6.0	2021	128b/130b + PAM - 4 + ECC	64.0 GT/s	7.877 GB/s	15.754 GB/s	31.508 GB/s	63.015 GB/s	126.031 GB/s