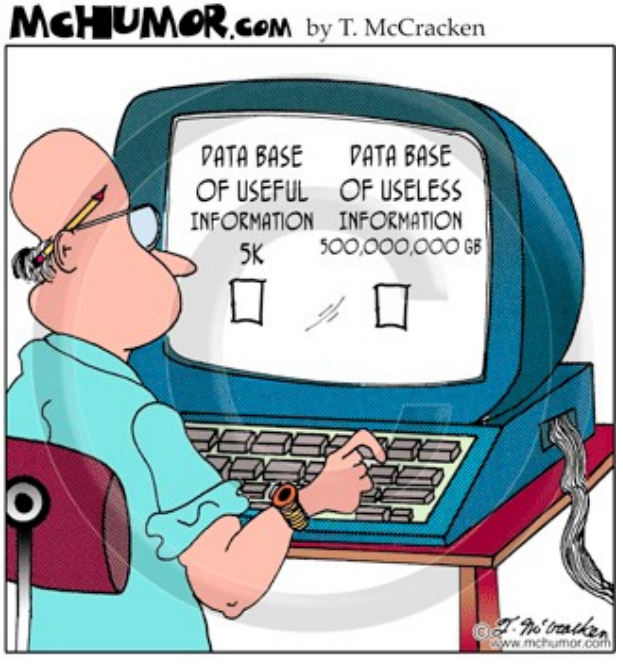
Data compresssion efficiency in silicon detector readout



M. Garcia-Sciveres
Lawrence Berkeley National Lab

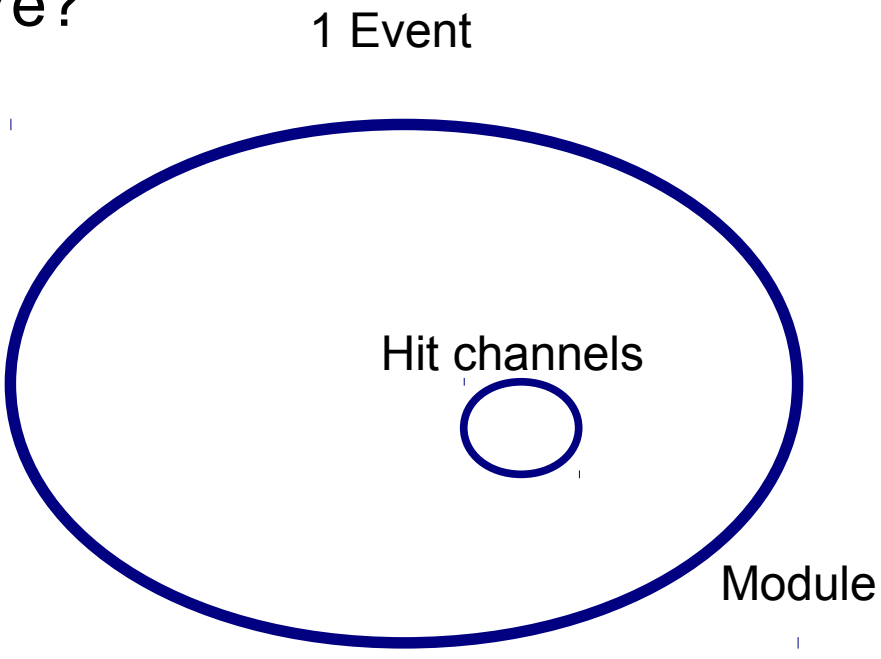
Workshop on Intelligent Trackers
University of Pennsylvania, May 15, 2014

Introduction

- Different detectors have chosen different encoding methods for digital readout.
  - Historical and trial/simulation motivated choices
  - Important consequences for hardware bandwidth
- Wanted to have an objective measure of what is the best one can possibly do (most information packed into least bandwidth)
  - With this one can ask if a particular readout scheme is good enough or has room for improvement
  - First analyzed case of binary strip detectors. Results published
    - **JINST_9_P04021 (link on agenda page)**
  - Now working on pixel detectors. May have results as proceedings

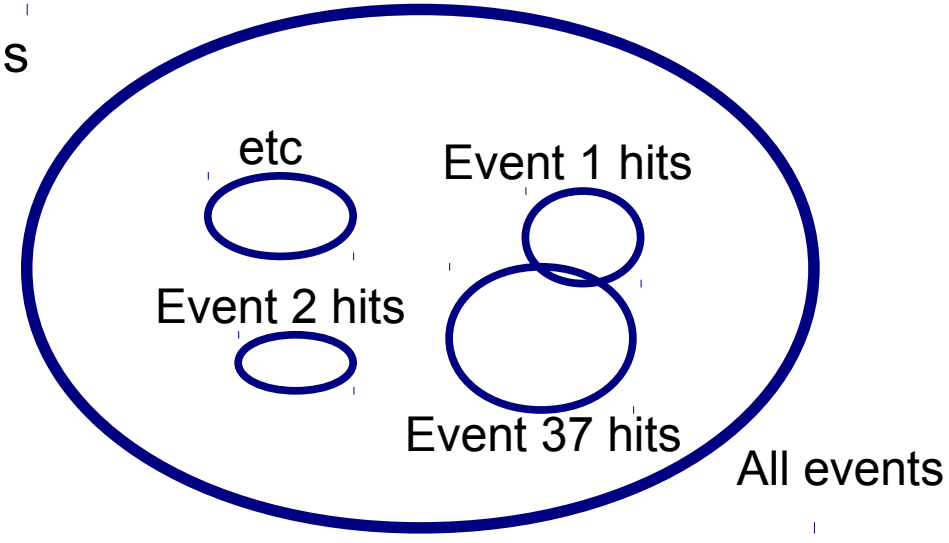
How much Information?

- One event with binary readout. One module.
  - Particles come and go, amplifiers and discriminators do their thing, a hit / no-hit pattern is left behind
- How much information is there?
  - Depends on the hit pattern
  - (so can't say)



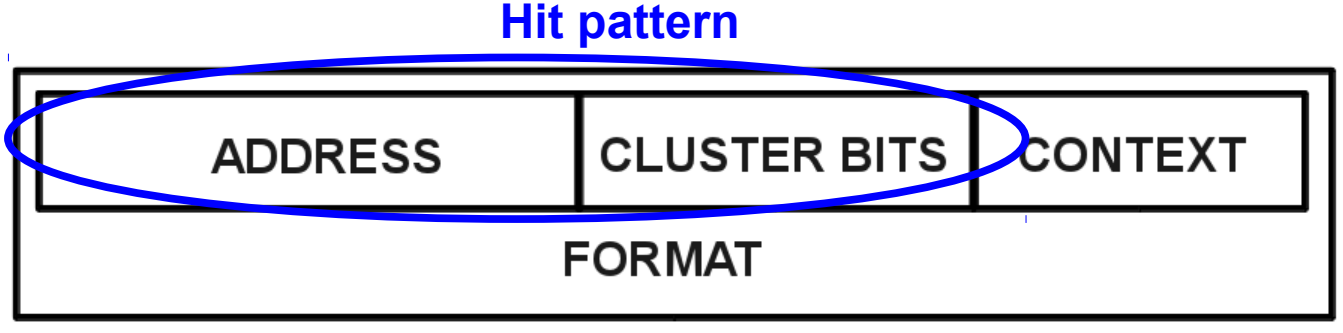
How much Information? (cont.)

- Now consider a large number of events (like a run)
  - The average information per event is defined
- Information Entropy
  - $H = -\Sigma\, p_i \log(p_i)$
  - Sum over all possible hit patterns
  - p_i is probability or frequency of hit pattern i
  - Choose log base 2, so H is in bits
- Let's assume we have H

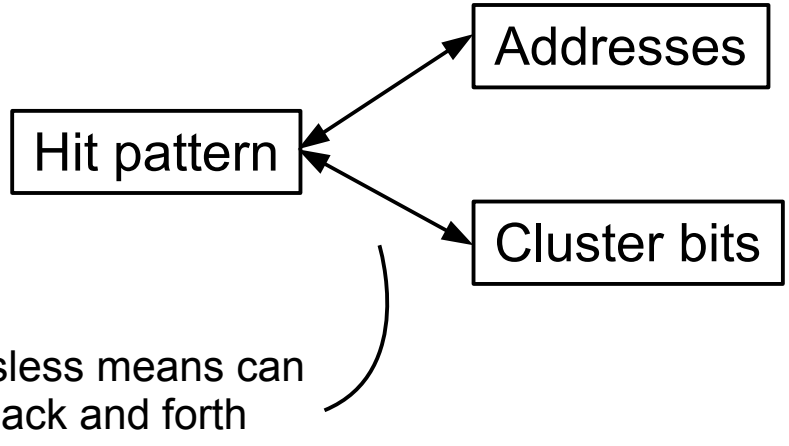


Assume we have calculated H

- Can now define a data encoding efficiency
  - Efficiency =
    (avg. number of bits/event used by your favorite encoding method) / H
  - (Put H in denominator so that efficiency is between 0 and 1)
  - Impossible to do better than H for lossless compression
- But there is more to data transmission than compression of hit information
  - Factorize the problem:



Calculating H in general for binary strips



Assumed random. Random is easy to calculate

Assume fully efficient. $H_{\text{cluster bits}} = N_{\text{cluster bits}}$

$$H_{\text{hit pattern}} = H_{\text{address}} + H_{\text{cluster bits}}$$

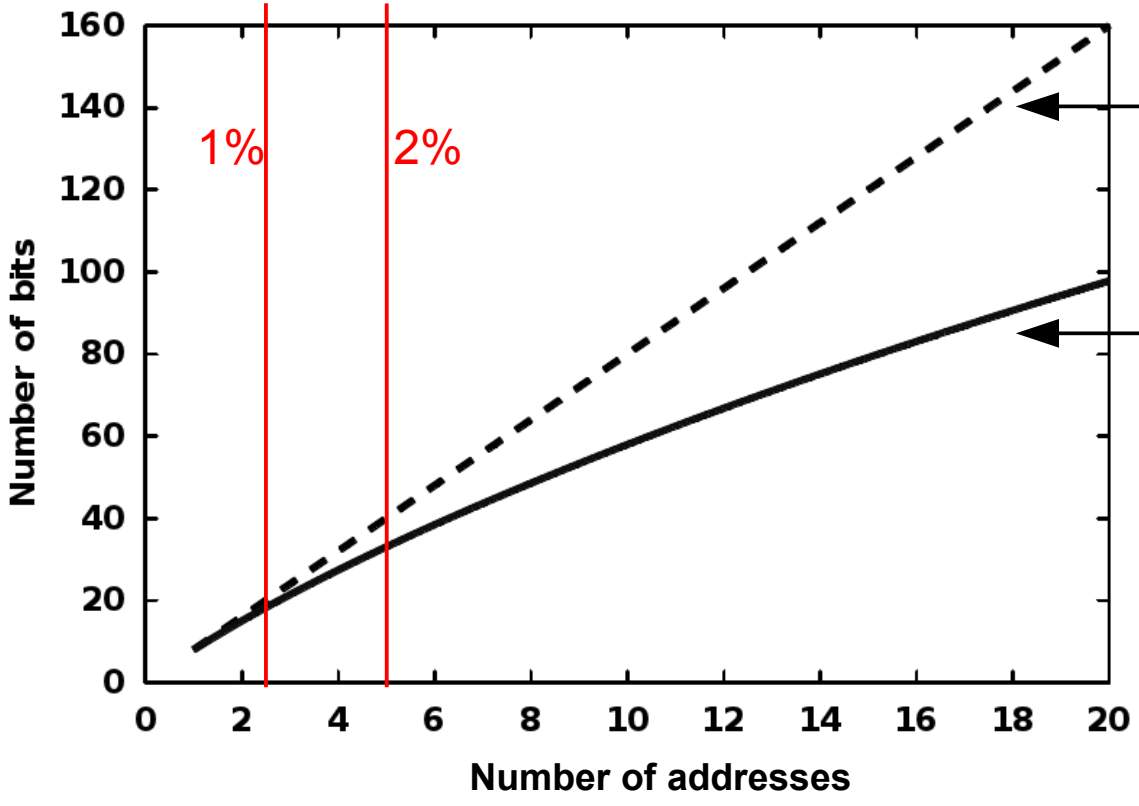
Real pattern not random due to clusters

0	1	1	1	1	1	0	0	1	1	0	0

Addresses approximately random

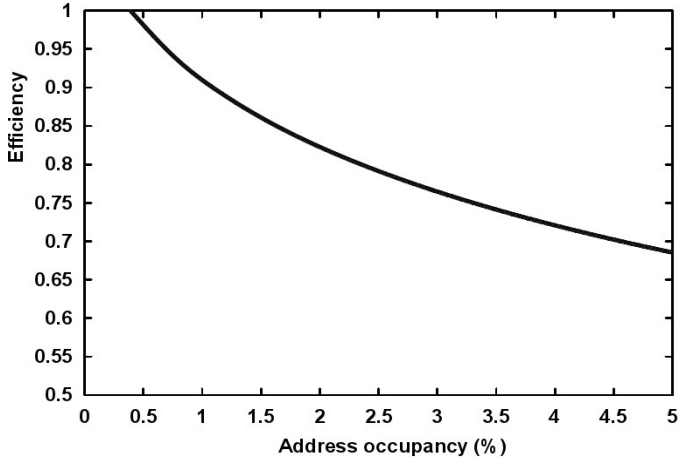
	x				x			x			

Example $H_{address}$ for 256 channels



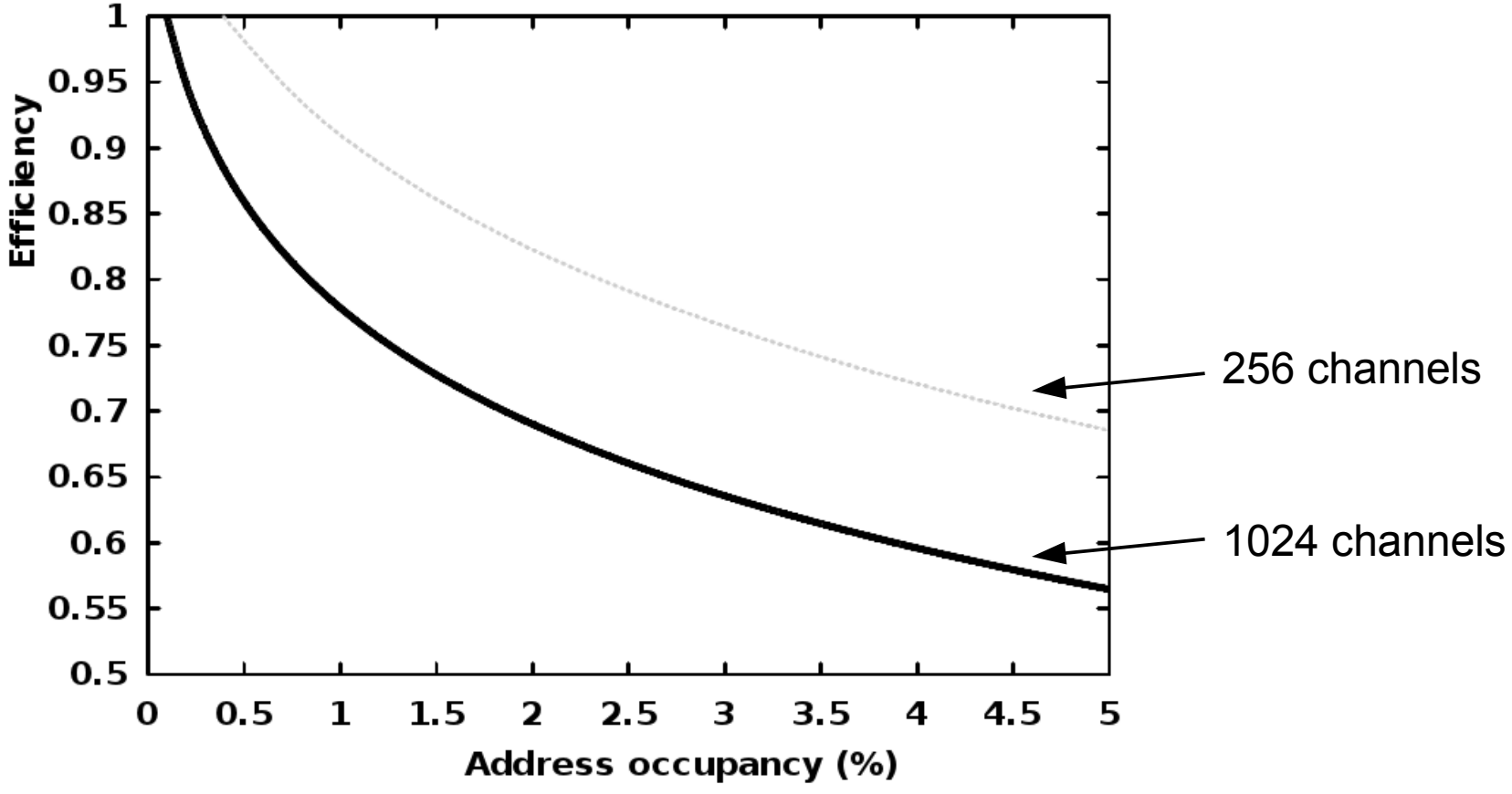
8 bits per address as we are used to

Entropy H
Log_2 (number of ways to pick k out of 256)



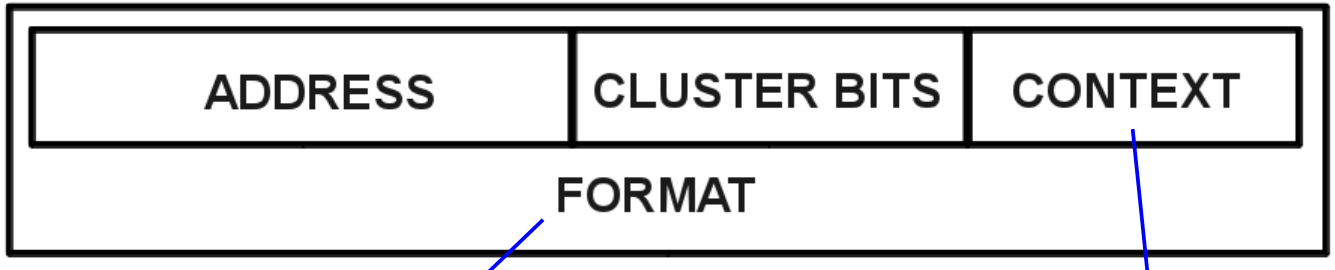
- Seems pretty efficient at 1% or 2% occupancy, right?
- (have to size readout BW for maximum occupancy with margin)

How about 1024 channels?



- Moral: The larger the number of channels combined, the greater the compression potential for a given occupancy
  - (compare zipping a small file vs. a large file)
- Single-chip based encoding probably not a great choice

Efficiency with all the parts



A few extra bits, eg. For trigger ID.
Assume we don't compress these

DC-balance, framing and so on.
Need an “entropy” that includes these elements

Number of addresses

$$\varepsilon_2 = \frac{H + E(H) + C + \nu K}{B_2}$$

Total number of bits used by your favorite method

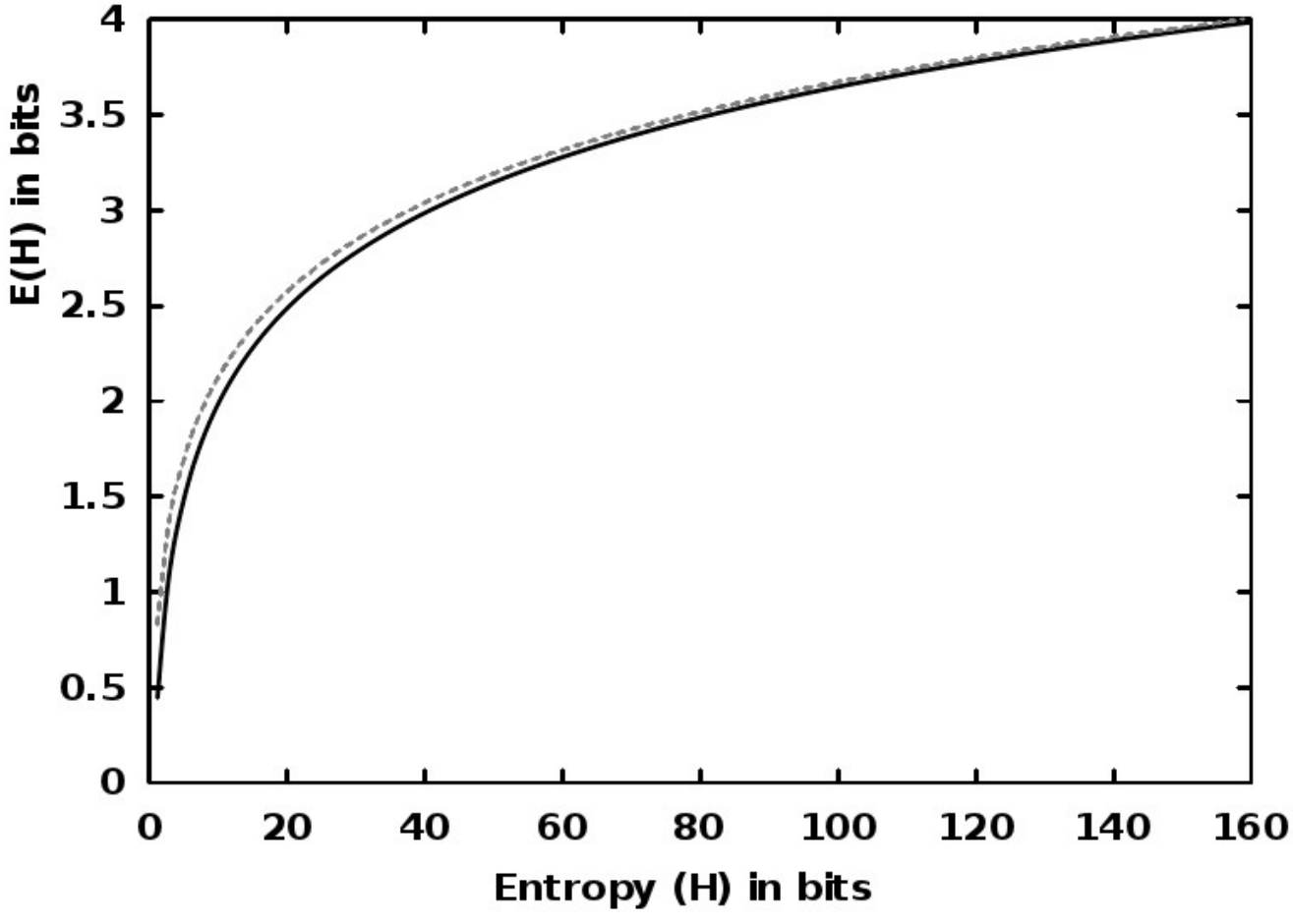
Number of cluster bits

E(H)

- Would like E(H) to be the minimum number of bits one needs to add to achieve DC-balance, framing, and basic error detection.
- No theoretical proof for a lower bound of E(H)
- But can define an example E(H) that is very small, so don't really care if still smaller is possible.
- Example E(H) is generalization of 8b10b
  - 8b10b provides all 3: DC-balance, framing, error detection
- 8b10b: replace all 2^8 8b "symbols" with symmetric 10b symbols (*)
    - Symmetric = half ones, half zeroes.
- General form: replace all 2^H Hb symbols with symmetric (H+E(H))b symbols
  - E(H) is smallest value such that there are at least 2^H symmetric symbols

(*) not quite, but close enough for this analysis

E(H) approximation for real H

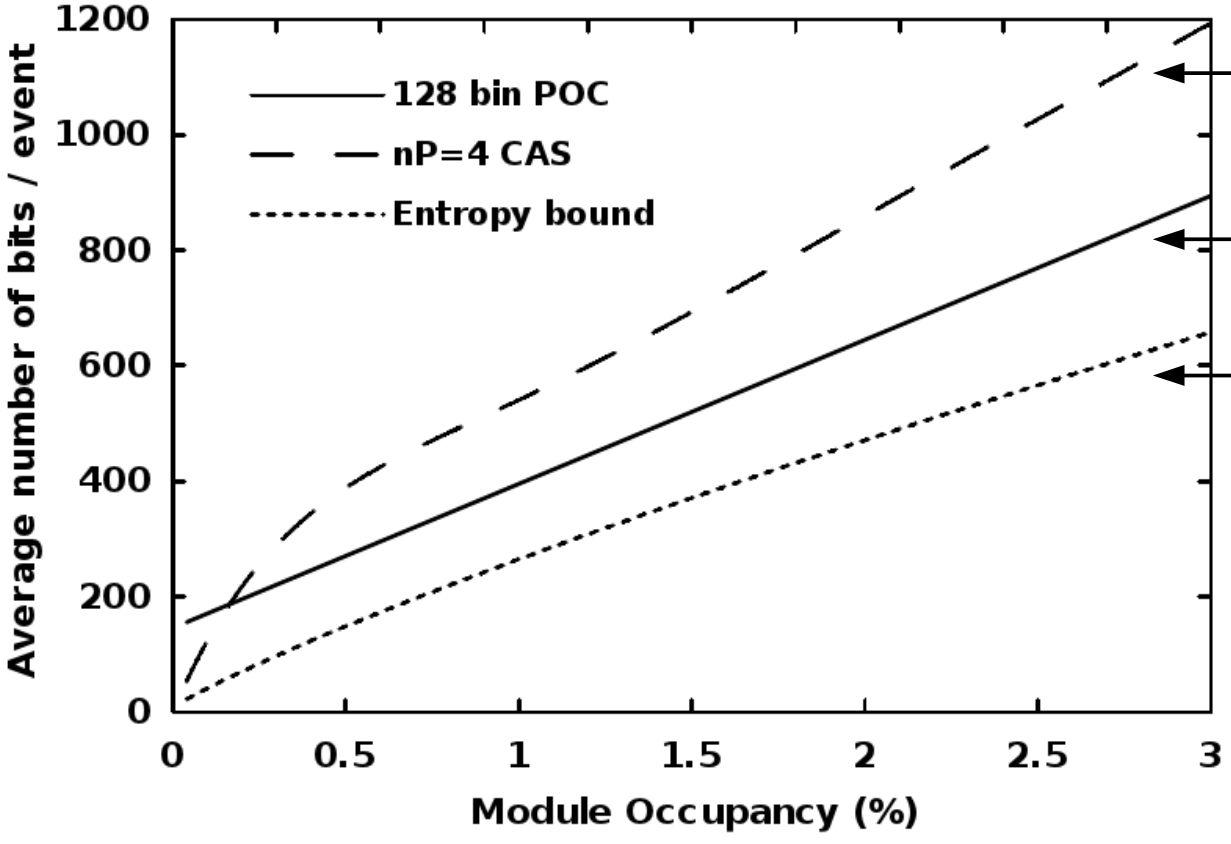


numerical

analytic

Putting sample methods to the test

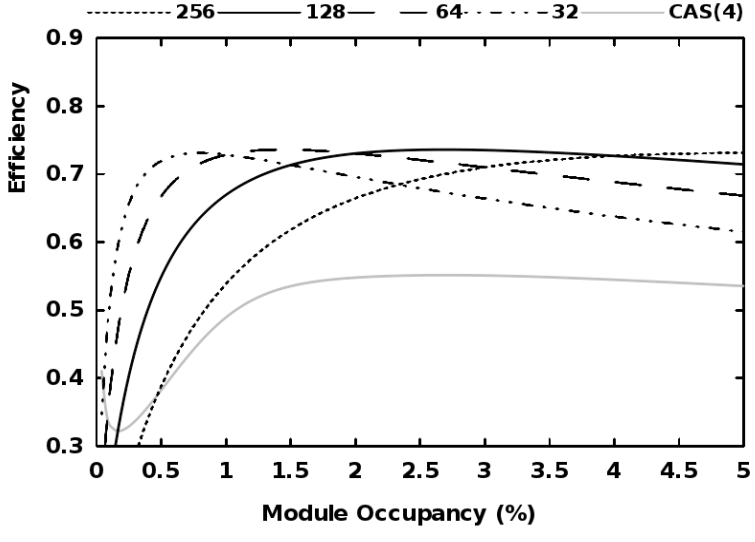
Binary readout of a 2560-channel module



Proposed encoding for ATLAS upgrade strips, with 64/66 transmission format

Pattern Overlay Compression method described in paper

There is clearly money to be made



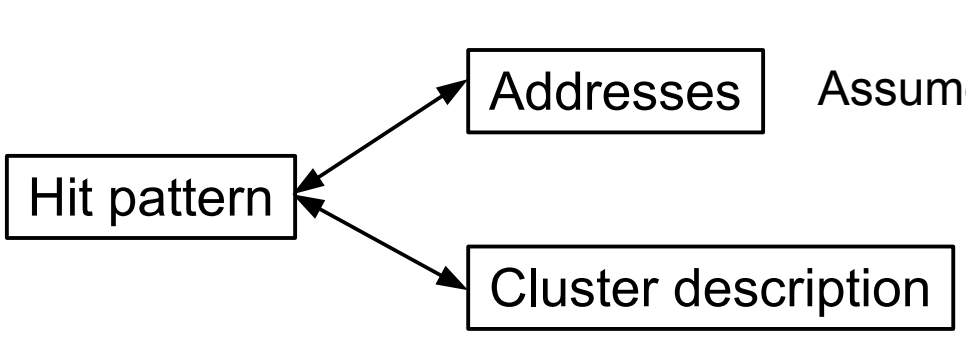
Is it possible to compress more?

- Suppose you want to read out EVERYTHING
  - All beam crossings, all hits
- Maximum possible efficiency involves two questions:
  - 1) how to minimize the entropy per hit?
  - 2) Is lossless readout always required? Lossy compression can use fewer bits than the information entropy
- Only (1) is within scope of this talk
  - Obvious: make bigger modules
  - Less obvious: combine data from multiple crossings within each module to effectively make even bigger modules
  - Even less obvious (outside this meeting): exploit coupled layer correlations (addresses in one layer are not random after all)
  - (also leads to point 2 of course- lossy output just for trigger)

Pixels (work in progress)

- Differences between strips and pixels are not only quantitative
- Pixel channel count is much larger
  (which sounds like problem solved!)
  - But occupancy range of interest is much lower
- Clusters are 2-dimensional
  - Requires dedicated treatment

Pixel hit pattern



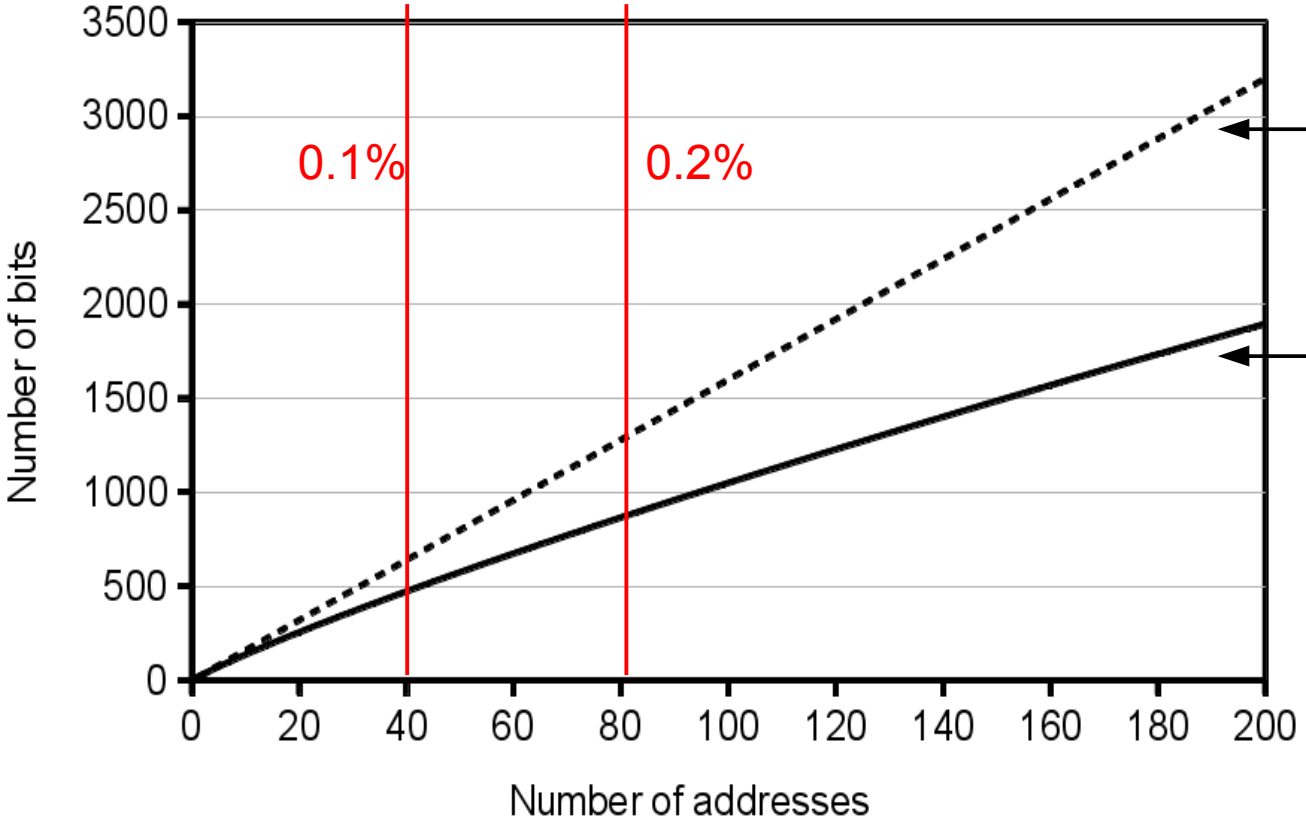
Assumed random as before ✓

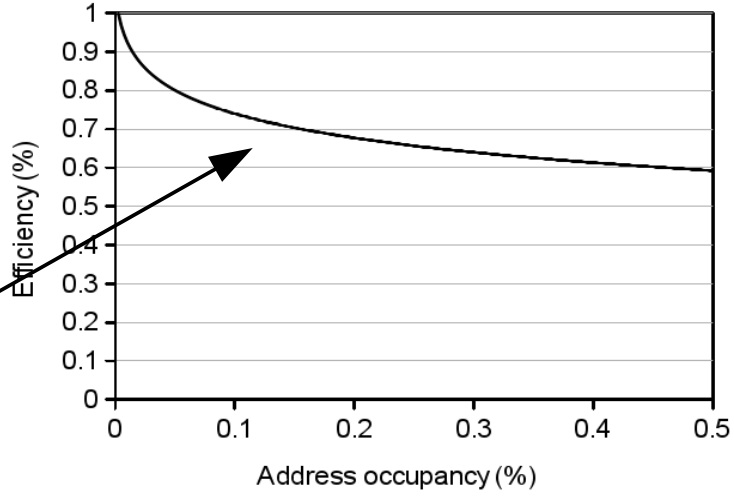
Can no longer make a simple assumption. Variety of cluster shapes possible for a given size.

$$H_{\text{hit pattern}} = H_{\text{address}} + H_{\text{clusters}}$$

$$\sum_{\text{sizes}} \sum_{\text{shapes}} p(\text{size},\text{shape}) \log(p(\text{size},\text{shape}))$$

Example $H_{address}$ for 64k channels



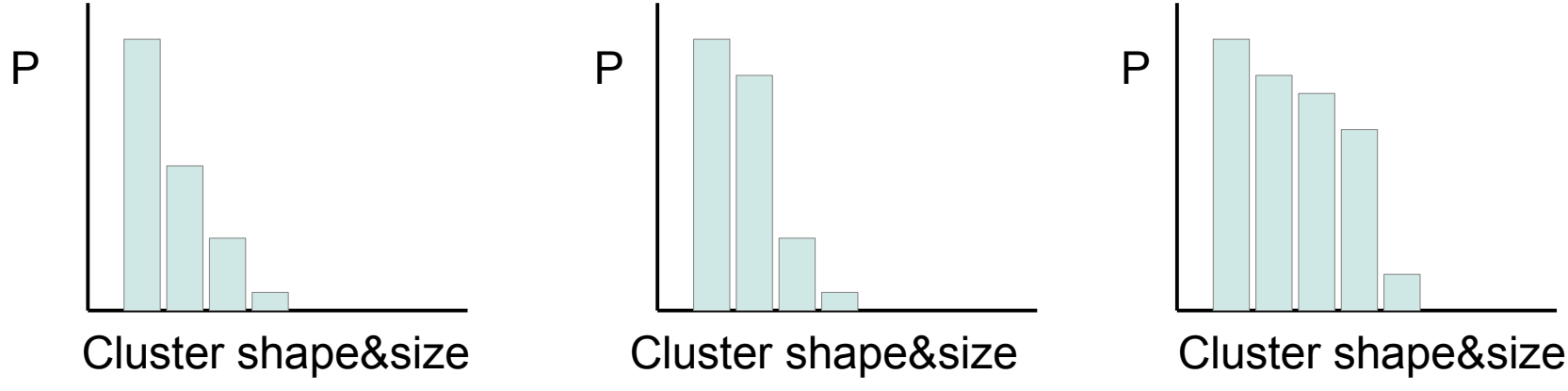


Very similar curve to 1024 strip module with 10X occupancy

Pixel cluster entropy

- Would like a general treatment, not sensitive to detector details
- For this need to find a random distribution (random is easy to calculate)
- Have not succeeded so far
- The problem:
  - There are many possible cluster shapes and sizes
    - For example a 3-pixel cluster has 20 possible shapes
  - But only a few will occur with high probability depending on detector details (including placement in eta)

How peaked is cluster PDF dominates the entropy



Sorted from most probable to least probable

Implementation concept

- How should an actual pixel chip handle the cluster problem?
  - Optimize chip design for a particular cluster PDF?
  - This is what was done in FE-I4, for example
- An interesting option is a chip that dynamically sorts clusters by frequency of occurrence, and assigns Huffman codes to each one
  - Start with a predefined Huffman code table. Eg:
    - 0 corresponds to 1-pixel
    - 10 corresponds to 2-pixel column-wise cluster
    - 110 corresponds to 2-pixel row-wise-cluster
    - 11100 2-pixel diagonal
    - 11101 2-pixel other diagonal
    - Etc.
  - Define new table dynamically to use optimal codes (user can read out and switch to new codes at will)

Remaining subject to study

- Compression of ADC information
- Pixels “like” ADC for
  - Analog interpolation
  - Splitting of merged clusters
  - dE/dx
- ADC distribution is NOT flat, but does have a characteristic falling spectrum
- Resolution requirement for analog interpolation and splitting different than for dE/dx
- Also suited for a Huffman code implementation

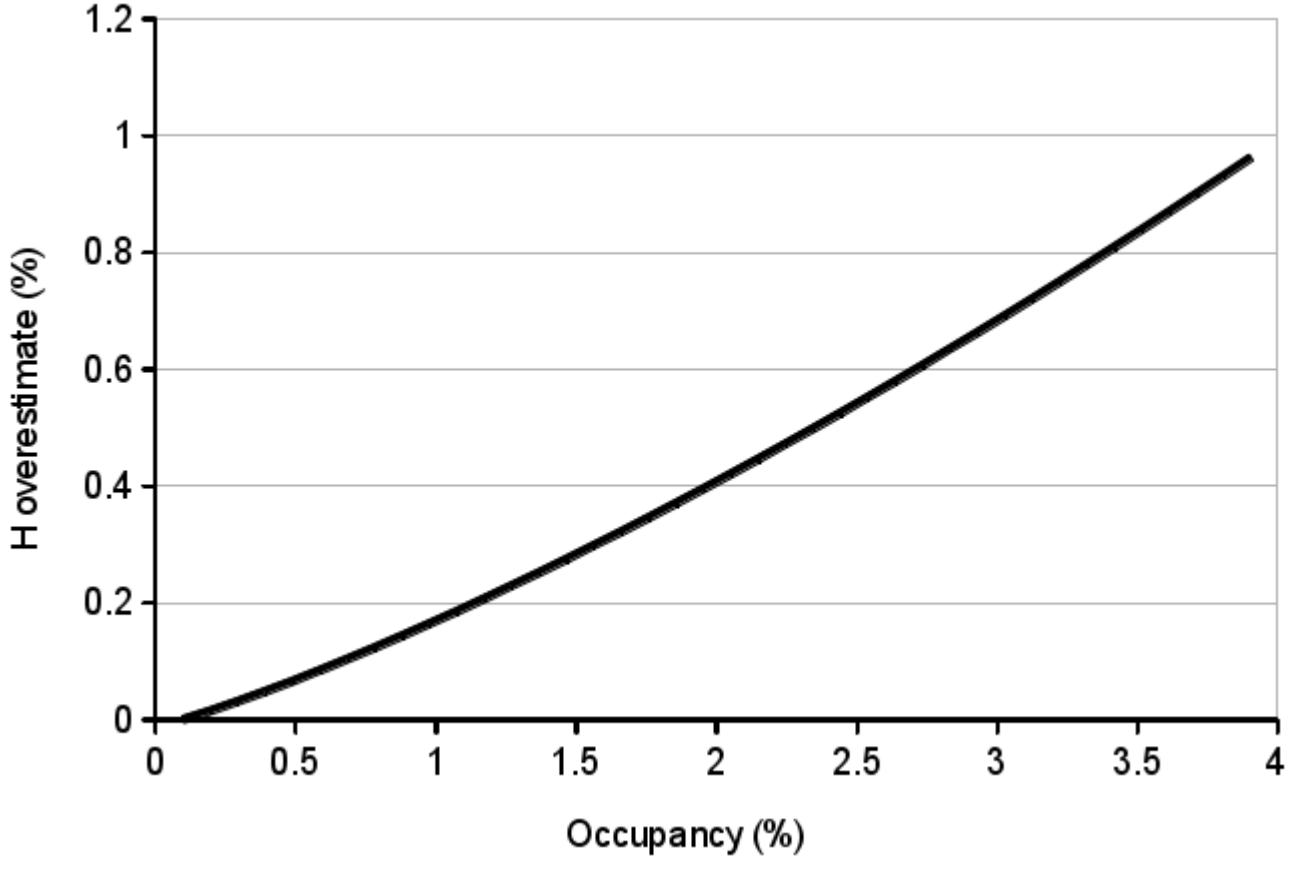
Conclusions

- Objective analysis of data compression efficiency achieved in detector readout is possible. Should be done for all proposed readouts
- When fighting for every last bit of bandwidth, readout efficiency should be maximized
  - Efficiency is equivalent to free, rad-hard, link bandwidth
  - Significant gains w.r.t. currently used/proposed methods possible
- Non-trivial aggregation of data from as many channels as possible is the way to increase efficiency
- Strips and pixels are in similar efficiency regimes despite the large granularity difference, due to accompanying occupancy differences
  - Treatment of clusters is 2-D complicates life for pixels
- Communication between coupled layers (even axial/stereo) could have alternate use to increase LOSSLESS compression efficiency.

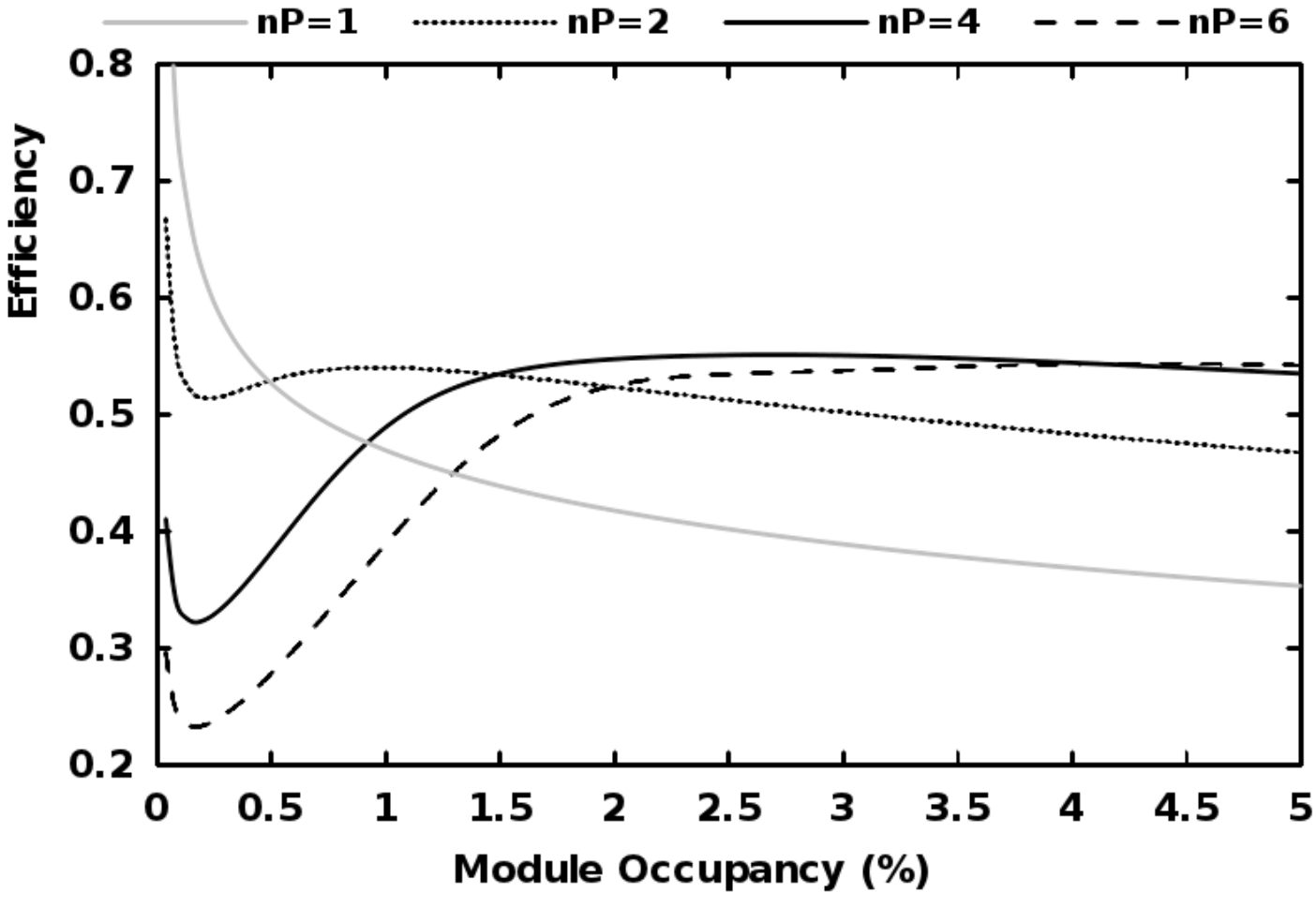
B A C K U P

Address entropy for strips

$\binom{1024}{k}$ Overestimate relative to $\binom{1024-k+1}{k}$

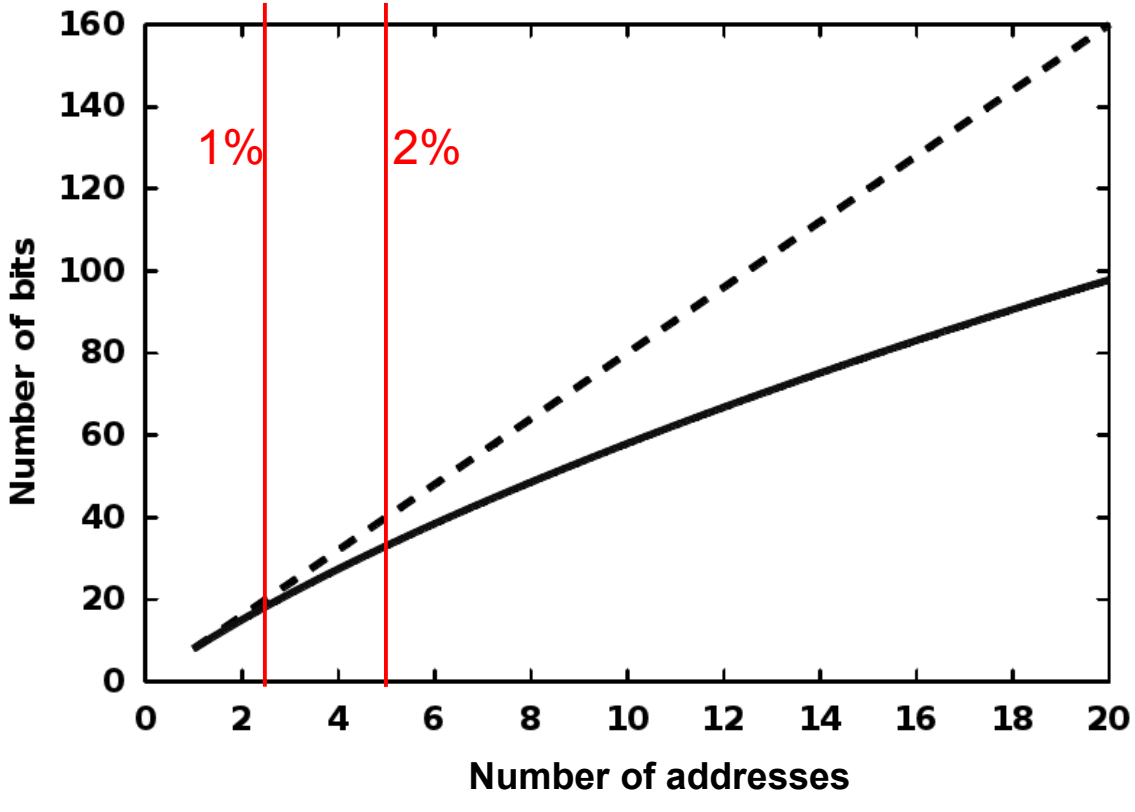


Packet readout



- 2560 channel module, all-inclusive efficiency

Pattern Overlay Compression (1/2)



Q: At what occupancy will this reach ~256 bits?
(the uncompressed pattern size)

A: At 50% occupancy

=> POC concept
Merge together many low occupancy patterns to reach high occupancy.
No compression to be done at high occupancy

Pattern Overlay Compression (2/2)

How to overlay that patterns:

Source pattern A:	0	1	0	0	0	0	0	0
Source pattern B:	0	0	0	0	1	0	0	1
Source pattern C:	0	1	0	0	1	0	0	0
Source pattern D:	1	0	0	0	0	0	0	0
Result pattern:	1	11			11			1
	0	0	0	0	0	0	0	0

How to write result pattern:

1D01A1C0001B1C001B0 or 1011000110010DACBCB

(A,B,C,D= 00,01,10,11)