

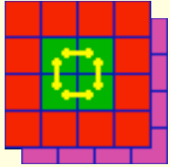
Comments on data compression for DAQ readout

◆ **Acknowledgements**

- ❖ **Most of this is not new, but it isn't written down in one place
(*and isn't all written down*)**
- ❖ **Useful discussions with several people, notably Norman**

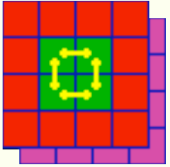
◆ **Introduction**

- ❖ **We will now use one type of ROD everywhere, so ...**
- ❖ **Preprocessor no longer uses PipeLine Bus to RODs**
 - ❖ **This was the bottleneck that dictated the need to do data compression on the PPMs**
- ❖ **Therefore, it is a good time to revisit:**
 - ❖ **Why** we need data compression
 - ❖ **Which** data to compress
 - ❖ **Where** to do it
 - ❖ **How** we might do it



Summary of data read out to DAQ

- ◆ Which data to read out, & number of slices, is variable
 - ❖ Our URD says it must be *at least one slice* of:
 - ❖ Trigger-tower output from look-up tables
 - ❖ Trigger bits sent to CTP
 - ❖ *This allows verification of algorithm processing, and gives details of where and what in the detector caused the trigger*
 - ❖ Much more data available, for up to 5 slices:
 - ❖ PPM trigger-tower raw data
 - ❖ PPM trigger-tower look-up table outputs for >1 slices
 - ❖ CPM trigger-tower input data, and hit-count results
 - ❖ JEM input data (*2x2 trigger towers*), hit-count results, and energy sums
 - ❖ CMM input data, and results (*both crate and system levels*)
 - ❖ The most voluminous items are the first three
 - ❖ **Essential** to be able to read out PPM inputs and outputs separately (*and different number of slices*), since they are the two biggest items
 - ❖ **Could reduce volume if CPM, JEM, CMM had separate control over readout of results and inputs** (*don't always want both ends of data links*)



Compression generalities

◆ Why?

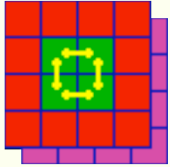
- ❖ DAQ needs to minimise number of readout links, buffers, and event size for storage
 - ✦ This is a 'soft' limit, not a 'hard' one
 - ✦ We might hope to achieve something like a factor of 2 reduction

◆ Which data?

- ❖ Biggest volumes are raw data, lookup-table outputs, and CPM inputs (*latter two are the same thing for $\eta < 2.5$*)
- ❖ **JEM inputs are marginal**, the rest not worth doing

◆ Where?

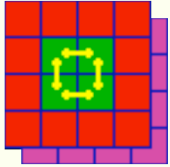
- ❖ As late as possible, on the RODs (*just before S-Links*)
 - ✦ Keeps things simple on the modules
 - ✦ Allows data monitoring and calibration from RODs without having to understand or undo the compression
 - ✦ *NOTE: might want to send some data to ROD for monitoring but not read it out to DAQ*



Compression methods (1)

◆ Zero suppression

- ❖ Simple case of run-length encoding — simply replace 0, 0, 0, ... by $n*0$
- ❖ Use it (*as already foreseen*) for look-up table outputs and CPM inputs, which are mostly zero (*pedestal-subtracted, noise-suppressed, 1 GeV/count*)
- ❖ **Less effective for JEM inputs** (*fewer zeroes due to adding towers in fours*)
- ❖ **Can't be used for raw data** (*pedestal, noise, 0.25 GeV/count*)



Compression methods (2)

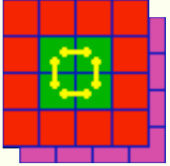
◆ Entropy coding

❖ Huffman coding

- ❖ Was studied extensively for Preprocessor
- ❖ Uses continuously-variable word lengths, with shortest words for most frequent data
- ❖ Not easy for humans to comprehend (*e.g. in event dumps*)
- ❖ If frequency distribution varies, must change code table in order to maintain efficient compression

❖ Something simpler?

- ❖ Most raw-data trigger towers are clustered in a small range around pedestal value (*and JEM inputs, if wanted, are mostly at or near zero*)
- ❖ Could do a reasonable job (*not as efficient as Huffman*) by using short words (*3–4 bits?*) for data near pedestal, and full-length words for rest
- ❖ Must evaluate how constant pedestal values will be; better if range using short words can remain fixed and the same for all towers



Conclusions

- ◆ Only reason left for data compression is to reduce event data to DAQ
- ◆ Only do it for large data volumes:
 - ❖ Trigger-tower raw data
 - ❖ Trigger-tower look-up table outputs
 - ❖ CPM trigger-tower input data
 - ❖ *Perhaps* JEM input data
- ◆ Use zero suppression for look-up table outputs
- ◆ Simple entropy coding for the rest
 - ❖ Huffman coding has disadvantages
 - ❖ Investigate using just two word lengths
 - ❖ Optimise the choice based on expected frequency distribution