Journal of Physics: Conference Series **219** (2010) 022046 doi:10.1088/1742-6596/219/2/022046

Lossless Compression of ATLAS Tile Calorimeter Raw Data

Vakhtang Tsiskaridze on behalf of the ATLAS Tile Calorimeter
Iv. Javakhishvili Tbilisi State University, 1 Chavchavadze ave., Tbilisi, 0128, Georgia
E-mail: vakhtang.tsiskaridze@cern.ch

Abstract. Recording and storing the Tile Calorimeter data at 100 KHz frequency is an important task in ATLAS experiment processing. At this moment Amplitude, Time and Quality Factor (QF) parameters are calculated using Optimal Filtering Reconstruction method. If QF is considered good enough, these three parameters are only stored, otherwise the data quality is considered bad and it is proposed to store raw data for further offline analysis. Without any compression, bandwidth limitation allows to send up to 9 channels of additional raw data. Simple considerations show that when QF is bad due to the shape differences between standard pulse shape and current signal (e.g. when several signals overlap), all channels are likely to report bad QF while the contained data may still be valuable. So, the possibility to save just 9 samples is insufficient and we have to compress the data. Experiments show that standard compression tools such as RAR, ZIP, etc. cannot successfully deal with this problem because they cannot take benefit of smooth curved shape of the raw data and correlations between the channels. In the present paper a lossless data compressing algorithm is proposed which is likely to better meet existing challenges. This method has been checked on SPLASH events (run 87851, contains 26 SPLASH events) and proved to be sufficient to save ALL channels data using the existing bandwidth. Unlike the common purpose compressing tools the proposed method exploits heavily the geometry-dependent correlations between different channels. It is important to note that the method relies on the only assumption that the registered signal shape is smooth enough and it does not require exact information about the standard pulse shape function to compress the data. Thus this method can be applied for recording pilled-up or unexpected signals as well.

1. Introduction

The ATLAS detector consists of four major components:

- inner tracker - measures the momentum of each charged particle;
- calorimeter - measures the energies carried by the particles;
- muon spectrometer - identifies and measures muons;
- magnet system - bending charged particles for momentum measurement.

The ATLAS Calorimeter itself consists of two sub detectors:

- Electromagnetic Liquid Argon Calorimeter (LAr);
- Hadronic Tile Calorimeter (TileCal).

1.1. ATLAS Hadronic Tile Calorimeter

The Hadronic Tile Calorimeter (TileCal) [1] is divided into two barrel and two extended barrel parts. Each part is divided into 64 wedge-shaped modules (Figure 1).

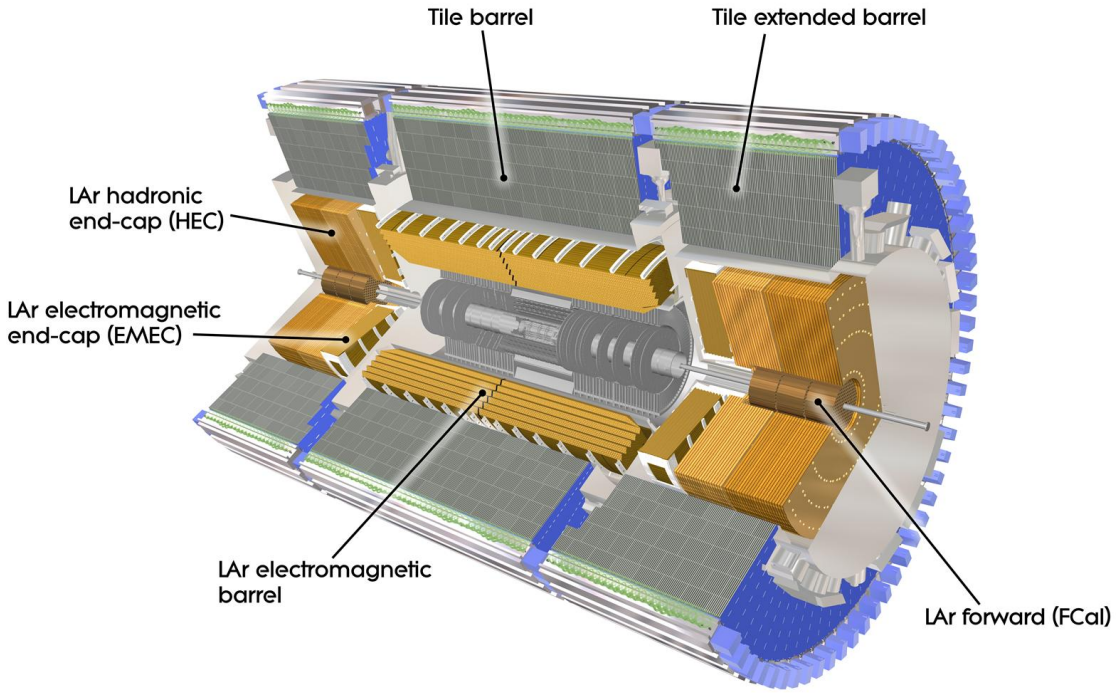


Figure 1. ATLAS Calorimeter

TileCal is a sampling calorimeter made of steel and plastic scintillating tiles which are grouped by cells. The light produced by the particles in cells is routed to photomultiplier tubes (PMTs). Each cell is read-out by 2 PMTs. Each barrel module is read out by 45 PMTs, and each extended barrel module contains 32 PMTs.

The PMTs and the front-end electronics are located inside "drawers" in the base of each module. To each PMT base a 3-in-1 board [2] is attached which amplifies and shapes the PMT output by means of a shaper.

The shaper generates standard pulse shape signal proportional to the energy deposited by the particles in the calorimeter cell. The shaper outputs are connected to the TileCal Digitizer system. The digitizer samples high and low gain signals in each channel with a gain ratio of 64. The system automatically determines which gain to use for a specific pulse, reading out only the high gain data unless an overflows or underflows is detected, in which case the low gain data are used.

Pulse shape is digitized in 7 time slices of 25 ns using 10 bit ADCs and sent to the back-end electronics as well as 1 bit information about the gain (high or low).

Back-end electronics adds additional bad bit indicating some technical problems with the channel. Bad bit may be set for a different reasons, one of them is transferring problem. When the bad is bit set the data cannot be used for reconstruction.

Thus, for each channel, we have 1 bit (Gain) + 1 bit (Bad) + 7 Samples $\times$ 10 bits = 72 bits = 9 bytes of raw data.

1.2. ATLAS Trigger System

The proton-proton Collider at the LHC at CERN will have a bunch crossing rate of 40 MHz [3]. The ATLAS Trigger System has been designed in three levels in order to select only interesting physics events reducing that rate of 40 MHz to the foreseen storage rate of about 100 Hz.

The Level-1 trigger (L1) is a hardware trigger and must reduce a rate from 40 MHz to 100 kHz. On every Level-1 trigger Accept (L1A) each sub-detector passes the data fragment corresponding to this event through the Read-Out Drivers (RODs) to the Read-Out Buffers (ROBs) where it will be stored until the Level-2 trigger requests or rejects it.

The Level-2 (L2) trigger is a software trigger based on Regions Of Interest (ROIs) defined by the Level-1 trigger. It reduces the rate from 100 kHz to 1 kHz. Due to the time limitation (10 μs for taking decision) L2 cannot request the data fragments from all ROBs and confines within the ROIs (requests data fragments from ROBs selected by Level-1).

The Level-3 trigger, a so-called Event Filter (EF) has time limit of 1 ms for decision. It takes into consideration the whole information from the system and reduces the rate from 1 kHz to 100 Hz.

The software based trigger levels (L2 and EF) are known as High Level Trigger (HLT).

1.3. Read-Out Driver (ROD)

The data coming from the Front-End Boards (FEB) are received in the Optical Receivers (OR) located in the front panel of the ROD. There are 8 ORs mounted on each ROD and each one receives data from one drawer of the detector. The bandwidth for each drawer is 640 Mbps.

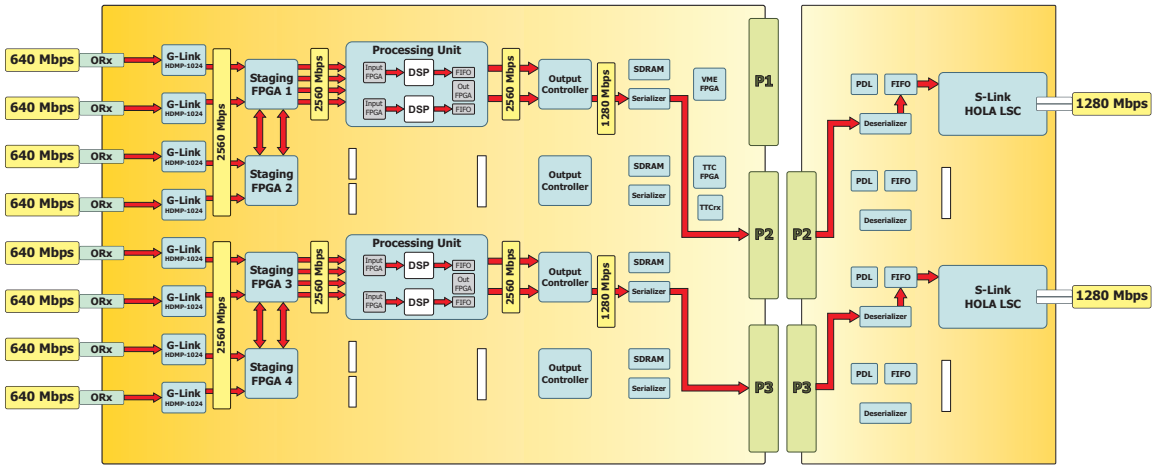


Figure 2. Block diagram for the ROD System

By design, TileCal uses 2 Processing Units (PU) per ROD. Each PU contains 2 Digital Signal Processors (DSP). Thus each DSP processes data from 2 drawers.

Each PU sends data trough the Output Controller (OC) FPGA. OC adds S-Link header and trailer words to the output data and sends this ROD event fragment through the S-Link to the Read-Out Buffer (ROB). Output bandwidth of the S-Link is 1280 Mbps.

2. Problem statement

Due to bandwidth limitations for 100 kHz frequency we cannot transfer the whole raw data as is and send only energy and time reconstructed inside the ROD's DSP using Optimal Filtering (OF) method [4, 5]. We send also the Quality Factor (QF) of the reconstruction. Thus we are dropping

the raw data recording and relying completely on the goodness of the online reconstruction of the Tile Calorimeter signal.

A careful analysis of the Tile Cal data shows that the main assumption of the OF method about linear dependence of the pulse shape of the amplitude is valid as the first approximation only and increasing the precision of the offline energy reconstruction may require storing as much raw data as possible. Keeping the raw data is also indispensable for offline reprocessing as well as for debugging and validation purposes. The proposed approach contributes to resolving this problem.

Let us first calculate the amount of a free space at our disposal during a single transfer.

Output bandwidth for ROD's PU is 1280 Mbps, counting for 100 kHz output limit is 12800 bits = 400 words per 4 drawers.

Of these, 14 words are used for:

- S-Link Begin 1 word
- Frag Header 9 words
- Trailer 3 words
- S-Link End 1 word

Thus, for each drawer we have the limit of $(400 - 14)/4 = 96.5$ words.

An additional 11 words overhead is sent for each drawer:

- Header 3 words
- DQ 5 words
- TotalET 1 word
- Muon 2 words (for 1 Muon)

Finally, for the drawer data itself it remains the limit of $96 - 11 = 85$ words.

For transferring all channels we need:

- Long Barrel: 45 channels $\times$ 9 bytes = 405 bytes $\sim$ 101 words
- Ext. Barrel: 32 channels $\times$ 9 bytes = 288 bytes $\sim$ 72 words

As we see, for long barrels the raw data do not fit within the bandwidth limits.

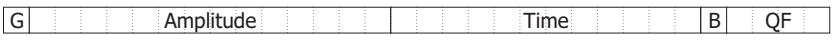
3. Two stage processing

At present, a two-stage processing is considered. On the first stage Amplitude, Time and Quality Factor (QF) parameters are calculated. QF represents the sum of absolute deviations of data from the standard shape (after fitting). The lower the value of QF, the better the quality of reconstruction.

3.1. Good Quality Factor ($QF \leq q_0$)

In the existing approach some boundary value $\mathrm{QF} = q_0$ must be fixed. It is assumed that the data with $\mathrm{QF} \leq q_0$ is good enough to be fully processed. In this case the Optimal Filtering Reconstruction method is considered sufficient and only Amplitude, Time and QF are stored.

`Reco` (32 bits)



Gain (G)	1 bit
Amplitude	14 bits
Time	12 bits
Bad (B)	1 bit
QF	4 bits

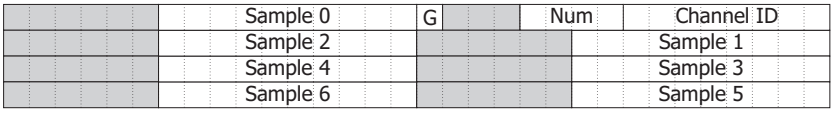
As we have at most 48 channels per drawer, the size of Reco Data is 48 words. If we send only Reco Data, then it rests $85 - 48 = 37$ words for sending some additional information.

3.2. Bad Quality Factor ($QF > q_0$)

In case of QF $> q_0$ the data quality is considered bad and it is necessary to store more detailed information for further studies. This is done during the second stage of data processing when the data of some selected particular channels with bad QF is stored. The `FragType1` (4 words) fragment has been introduced for this purpose.

`FragType1` (128 bits)

Gain (G)	1 bit
Num	4 bits
Channel ID	8 bits
Sample	10 bits



Here Num is a number of samples (in our case it always equals 7) and the Channel ID is an ID of a channel of which raw data are transferred. The above mentioned 37 available words could be used to send additional data for some particular channels with bad QF.

The `FragType1` format is used at present and allows sending up to 37 words/128 bits = 9 channels, but it is subject to changes. We can remove the empty spaces, Gain, Num and even Channel ID data and keep only 7 samples, but even if we change the format, we need at least 70 bits (we may send 37 words/70 bits ∼16 channels).

4. Raw data transfer

If we do not use any compression, at present we send up to 9 channels of additional raw data to the second stage of processing. This means that we have to choose quality bound q_0 large enough to avoid more than 9 channels with bad QF. Simple considerations show that when QF is bad due to the shape differences between standard pulse shape and current signal, all channels are likely to have bad QF. So, the possibility to send just 9 channels in FragType1 (or 16 channels by more compact format) may sometimes appear insufficient.

Experiments show that standard compression tools such as RAR cannot successfully deal with this problem because they cannot take advantage of the smooth curved shape of the raw data and correlations between the channels. Note that if we have bad QF for some channels, we do not need to send Reco Data for them. So, we can free up a 32-bit word per channel for sending raw data.

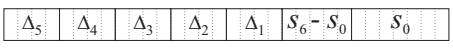
5. Proposed method

In our approach we demonstrate how the data of *all* channels may be stored within the existing 85 words limit. First we gather gain and bad channel bits of up to 48 channels of a drawer ($48+48 = 96$ bits = 12 bytes) into 3 words. Then we introduce the following 4 different formats (`Ped`, `Amp6`, `Amp8`, `Full`) for representing the compressed data.

5.1. Pedestals compression

An overwhelming majority of the channels contain pedestal or very low signal. In this case we do not need to send 10 bits per sample. We compress data by sending the average and deviations (the difference between the samples and the average).

`Ped` (4 bytes)



7+5+4+4+4+4+4 = 32 bits

s_0	7 bits	[0..127]
$s_6 - s_0$	5 bits	[-16..15]
$s_i - a$	4 bits	[-8..7], where $a = (s_6 + s_0)/2$

We consider that the deviation does not exceed ∼8 ADC counts. In our case we are calculating the deviation from the average a between the first and the last samples: $a = (s_0 + s_6)/2$. The expected pedestal values are ∼40 ADC counts. So, for the first sample a 7 bits [0..127] interval is more than sufficient. As far as the deviation does not exceed 8 ADC counts the difference between s_0 and s_6 cannot exceed 16 ADC by absolute value. For storing this difference 5 bits [-16, 15] will be sufficient. For other samples, 4 bits are sufficient, difference will be [-8..7].

Let us recall, that we have to pack data for 45 channels in the volume of 85 words. If we have at least 36% (at least 16 channels) with pedestal data for each drawer (or group of 4 drawers in average), we shall be able to send all other channels even without compression:

$$12 \text{ bytes (Gain \& Bad bits)} + 16 \times 4 \text{ bytes (Ped format)} + (45 - 16) \times 9 \text{ bytes (raw data)} = \\ = 337 \text{ bytes} = 84.25 \text{ words}$$

fits within 85 words limit.

All channels that fit within this format are sent by this way. Easy to see, that the data compressed by this way can be easily uncompressed.

5.2. Signal compression

The main idea in signal compressing is that we expect the signals in neighboring cells to have arbitrary but similar shapes with more or less equal timing. Similarity means here that one shape may be with some proximity transformed into another by linear transform, i.e. by multiplying by some factor coefficient and adding a shift constant. If this is the case, we suggest storing this linear transform and the differences with respect to the previous channel. Evidently this will require a considerably less storage.

Let us consider two channels (u and s) with samples u_i and s_i respectively ($i = 0, ..., 6$). We are scaling u_i on s_i. First we fix minimum and maximum samples in u. Lets min and max be the indices of the minimal and maximal samples in u. If we have more than one minimal or maximal sample, we choose those with minimal indices. Than we scale linearly the interval $[u_{min}, u_{max}]$ to the corresponding interval $[s_{min}, s_{max}]$. Note, that s_{min} and s_{max} are not necessarily the minimal and maximal values in samples s.

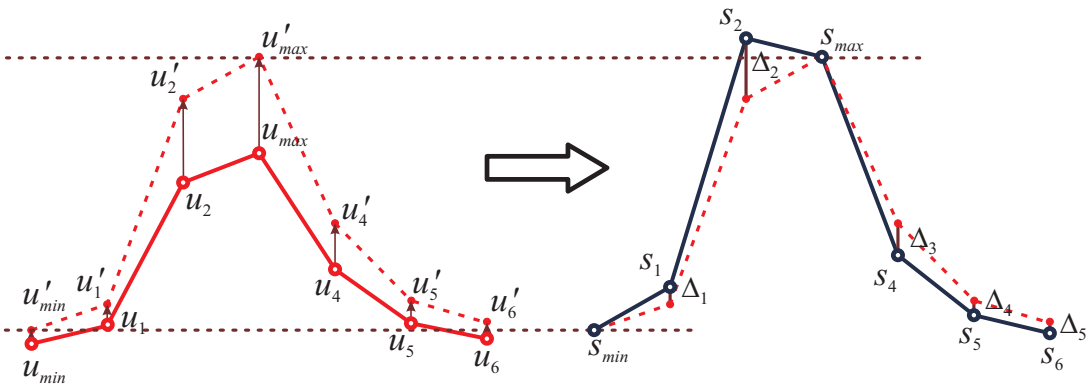


Figure 3. Samples scaling

In this case the scaling function is

$$f(x) = \frac{s_{max}(x - u_{min}) + s_{min}(u_{max} - x)}{u_{max} - u_{min}},$$

so, that $f(u_{min}) = s_{min}$, $f(u_{max}) = s_{max}$.

Let us define $u'_i = f(u_i)$. We are compressing the data storing information on scaling function and difference $\Delta_i = s_i - u'_i$.

To send information about scaling function, we send just s_{min} and s_{max}, because channel u is already sent (i.e. u_{min} and u_{max} can be calculated by receiver). To send s_{max} we need 10 bits, but for s_{min} we expect a value around the pedestal i.e. $\sim$40 ADC counts. So, sending s_{min} may require less than 10 bits storage.

Then we continue with sending differences Δ_i between samples. As far as the values of u_{min}, u_{max}, s_{min} and s_{max} are already sent, we need to send the remaining 5 values of Δ_i, where $i \neq min$ and $i \neq max$.

Depending on difference between samples, we consider two different formats `Amp6` with 6 bits differences ($\pm$32 ADC) and `Amp8` with 8 bits differences ($\pm$128 ADC).

Due to aligning our data formats to byte, for s_{min} in case of `Amp6` we have 8 bits [0..255] and in case of `Amp8` we have 6 bits [0..63].

`Amp6` (6 bytes)

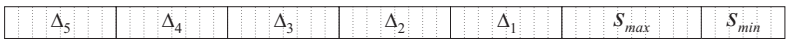
8+10+6+6+6+6+6 = 48 bits

s_{min} 8 bits [0..255]

s_{max} 10 bits [0..1023]

Δ_i 6 bits [-32..31], $i = 1, ..., 5$

`Amp8` (7 bytes)



6+10+8+8+8+8+8 = 56 bits

s_{min} 6 bits [0..63]

s_{max} 10 bits [0..1023]

Δ_i 8 bits [-128..127], $i = 1, ..., 5$

As there is no sense scaling pedestal samples to current one, in both cases we scale previous NON PEDESTAL samples (with ANY QF) on the current samples.

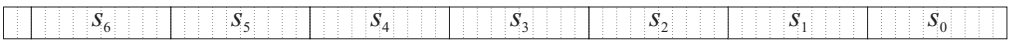
First we try to use `Amp6` format if it is applicable, otherwise we try to use `Amp8` format. If none of these is applicable, we send full data.

Obviously this method is sensitive to the choice of the scaling sample. We use some predefining ordering of channels and choose previous non pedestal samples according to this order.

5.3. Full data

The full format is only used in case, when no other formats described above are applicable.

`Full` (9 bytes)



2+10+10+10+10+10+10+10 = 72 bits

s_i 10 bits [0..1023], $i = 0, ..., 6$

In this case, we send all samples without compression, aligning them to byte.

5.4. Ordering the cells

Some ordering of channels should be predefined before starting compression in such a way, that neighboring channels have similar shapes. Amplitude and pedestal differences may be compensated by linear transform described above.

Experiments show that the proposed method is very sensitive to channels ordering. Here is the ordering that we used during experiments.

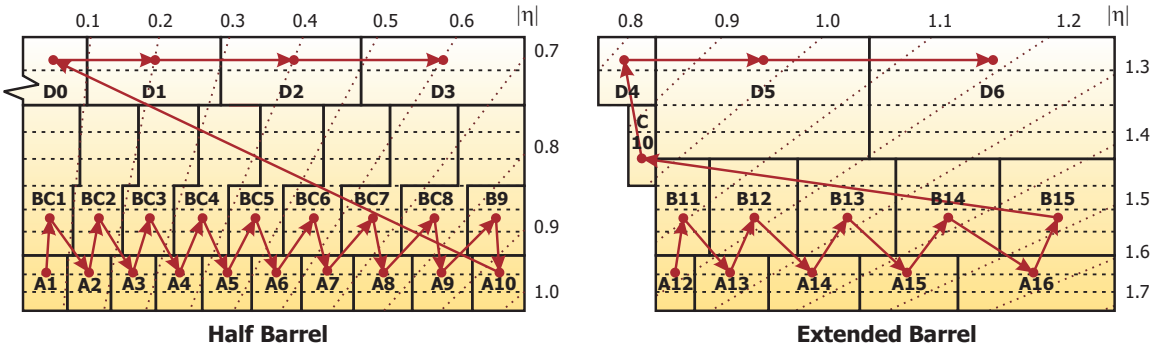


Figure 4. Ordering the cells

For each cell, we place together both PMTs, which are connected to the same cell, because if timing is set correctly, the correlation between these 2 PMTs should be very high. Thus, for sending the data from second PMT, `Amp6` format will probably be sufficient. Next, we try to place neighboring cells in a sequence one after another. The cells sequence for the Long Barrel and the Extended Barrel is different, as shown on Figure 4.

6. Statistics for run 87851

This compression method was tested on SPLASH data [6] run 87851 where there are some events with all channels containing signal. Of course, in real case, we do not expect such a hard situation. Thus, we may consider the results of compression as a confirmation of the efficiency of the method.

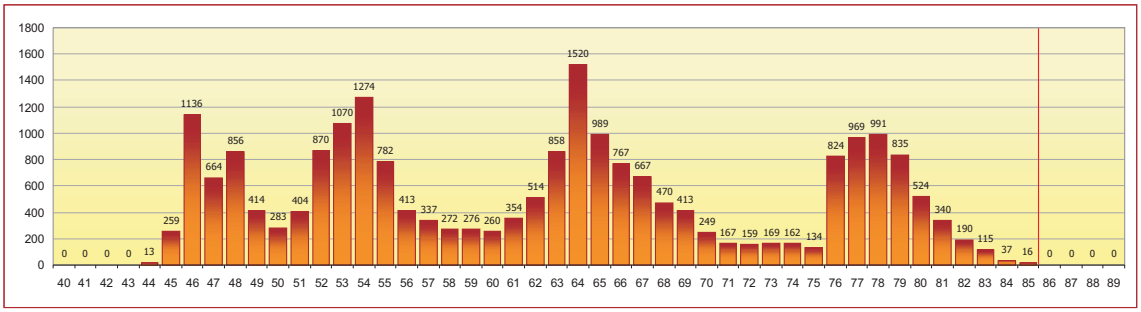


Figure 5. Histogram of compressed data lengths (fits within 85 words)

Figure 5 shows the histogram of compression size of all drawer data for all events of the run. As one can see in all cases it fits within 85 words limit.

6.1. Example of successful compression of the raw data with a signal in ALL channels

The following example (Figure 6) illustrates the case when the QF in *all* channels is considered bad and all channels contain signal.

In this example we are sending 48 channels of long barrel (with those 3 channels which are not connected to PMTs).

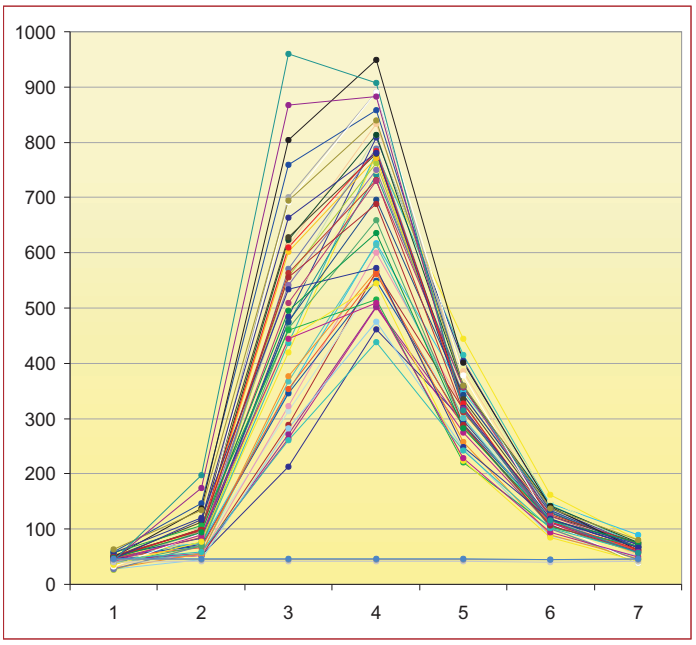


Figure 6. Example of raw data with signal in ALL channels

7. Time constraints

While the problem of fitting the bandwidth seems to be solvable, at the moment it remains an open problem whether the algorithm can fit the time restrictions of data processing at 100 kHz frequency. The answer to this question will affect the QF bound value q_0, which determines whether to use second stage algorithms or not. Obviously, the lower the value of q_0, the higher the required performance of the compression algorithm.

Some latest preliminary considerations allow hoping that the admissible value of q_0 may be very low, but this topic is currently out of the scope of the present paper.

8. Further improvements

Note that the compression algorithm does not use any explicit information about the pulse shape function and one could get additional benefit by taking this shape into consideration. It also seems promising to use a tree driven ordering of cells instead of a linear one. There are also some other improvements that can increase the compression rate.

9. Conclusions

A data compression algorithm is proposed for the Raw Data recording (in case of bad QF). The method used is especially suitable for the existing detector design and hardware configuration. The algorithm has been tested for heavy input flow and proved to be capable to fit into the existing bandwidth bounds. The determination of the optimal value for Quality Factor bound q_0 determining switching to the second stage algorithms remains open and is under study.

Acknowledgments

Author would like to thank Alexander Solodkov for his support and helpful suggestions. Author would also like to thanks Carlos Solans Sanchez and Alberto Valero Biot for detailed discussions on existing data formats and bandwidth limitations.

References

[1] ATLAS Tile Calorimeter Technical Design Report, CERN/LHCC 96-42.
[2] K.Anderson, J.Pilcher, H.Sanders, F.Tang, S.Berglund, C.Bohm, S-O.Holmgren, K.Jon-And, G.Blanchot, M.Cavalli-Sforza, Bi-gain Front-end electronics for ATLAS Tile Calorimeter, Fourth Workshop on Electronics for LHC Experiments, Rome 1998, p. 79.
[3] C. Solans, ATLAS TDAQ Software for TileCal Commissioning, Research Report, University of Valencia, 2007
[4] E. Fullana *et al.*, Optimal Filtering in the ATLAS Hadronic Tile Calorimeter, ATLAS Note ATLTILECAL-2005-001.
[5] J. Poveda *et al.*, Validation and Performance of the TileCal Optimal Filtering Reconstruction Algorithm, ATLAS Note.
[6] P. Krieger, First LHC Beams in ATLAS, ATL-GEN-SLIDE-2009-020.