On the Challenge of Keeping ATLAS Tile Calorimeter Raw Data

On behalf of the ATLAS Tile Calorimeter Group

V. Tsiskaridze

Albert-Ludwigs-Universität Freiburg
Iv. Javakhishvili Tbilisi State University

13th ICATPP Villa Olmo, Como 3-7 October 2011
Outline

- Introduction
- Initial Approach (Reco+Frag1)
- Lossless Compression
- DSP Optimization
- Proposed Compression Formats (Frag5)
- Recent updates of Frag1 and Frag5
- Comparative study
- Implementation: Pros and Cons
- Twin Mount Framework
- Validation and High Rate Tests
- Perspectives
- Conclusions
The ATLAS Tile Calorimeter

- Tile Calorimeter – sampling calorimeter made of ~5000 steel/plastic scintillator cells;
- Each cell is connected to two photomultiplier tubes (PMT);
- Passing through the active medium the particles deposit some energy thus producing the light;
- The PMTs detect this light and convert it in electrical signals;
- Each PMT is connected to a read out channel;
- Thus there are ~10000 channels producing signals;
- Long Barrel – 45 channels/module, 128 modules;
- Extended Barrel – 36 channels/module, 128 modules;
- The analogue signals from the PMTs are shaped and amplified in two separate chains (High and Low gain with a gain ratio of 64).
Standard Pulse Shape

- The resulting (undistorted) pulses $s(t)$ are assumed to be proportional to some "standard" shape $g(t)$ arriving at a moment $t = \tau$ called Time, with scaling constant $A$ called Amplitude, and shifted up by some constant value $p$ called Pedestal:

$$s(t) = A \cdot g(t - \tau) + p$$

- The main (non-zero) part of a Standard pulse shape spreads over a time interval of about 150 ns and its maximal value is achieved at point $t = 0$.

- The standard shape itself differs by about 1-2 % for High and Low Gains.

- The measurements show that it also varies slightly from channel to channel.
Piled-up Pulse Shape

- Several pulses coming at different times may sum up forming a piled-up shape.
- While the maximal value (Amplitude) of an undistorted pulse (taking the Pedestal and Gain into account) is proportional to the energy deposited by the particle in the cell, this is not the case for pile-up pulses.
Pulse Sampling

- Each pulse is digitized by means of a 10-bit ADC in 7 time slices sampling the signal every 25 ns. We denote the sampling points as $t_1$, ..., $t_7$ and the resulting samples as $s_1$, ..., $s_7$.

- The only information remaining from the energy deposited by the particles is a collection of seven 10-bit words and one bit indicating gain used (High/Low) which is transmitted to the Back-End.

- The Back-End adds one bit indicating the consistency of data (Good/Bad) received from Front-End for particular channel.
  - Note that this comes to 72 bits in total.
Data Processing Strategy

Data processing on ATLAS Tile Calorimeter (TileCal) consists of online and offline phases. Data selected during online processing are stored for further offline analysis. Online processing is effectuated in the fixed-point arithmetic Digital Signal Processors (DSP). Operation environment limits output bandwidth to 400 (32 bits) words and the processing time to 10 μs at the ATLAS Level1 trigger rate (100 kHz).

The initial data processing strategy consisted in the following:

- Reconstruct online Amplitude and Time;
- Provide reconstructed magnitudes for High Level Trigger responsible for events selection;
- Calculate Quality Factor (QF) and compare it with the predefined QF Threshold;
- For the samples with "good" Quality Factor (i.e. $QF < QF\text{Threshold}$) store reconstructed magnitudes and Quality factor only for further offline analysis;
- For the samples with "bad" Quality Factor (i.e. $QF > QF\text{Threshold}$) store as much raw data as the output bandwidth allows.

Existing implementation was limiting this amount to 8 channels (out of 45) and the appropriate value of the threshold had to ensure fitting within this limit. Thus in case of "bad" QF about 18% of raw data could be stored while in case of "good" QF no raw data was stored at all.
One of the problems of interest was whether we could increase the raw data storage. In the initial approach the output of each module consisted of two fragments:

- **Reco** – containing the reconstructed data from 48 channels;
- **Frag1** – containing the raw data from the selected channels.

Because of straightforward packing the **Frag1** fragment was actually at about 37% empty. Rearranging the information could allow saving a bit more channels, but this still seemed insufficient as far as we intended to save all the raw data.

The starting idea for lossless compression was to use the fact that pulses in different channels are soundly correlated: if there is a light in one cell, then the light is likely to appear also in the cells nearby. Thus the real amount of information to store would be much less than it looks at the first sight and compression may help to improve the performance.

Standard compression tools (like **RAR** or **ZIP**) are of little good here, as they cannot benefit the specific structure of the data and dependencies between the channels.
Lossless Compression

The first version of lossless data compression comprised the pedestal compression only and soon was replaced by more complicated and powerful compression scheme reported on TileCal Week (Rio, Dec 2008) and later presented on CHEP09 (Praga, March 2009).

It was proposed to process the channels in an appropriate order and send the differences between consecutive samples according this order.

The method substantially used the geometry of the Calorimeter as it was sensitive to the channels ordering used during the compression. It needed **NO INFORMATION** about the signal pulse shape and proved to be able to compress piled-up and other non-standard signals from **ALL** channels fitting within the existing bandwidth.

Depending on the range of differences between samples 4 different formats (**ped**, **amp6**, **amp8**, **full**) for data transfer were introduced.
Implementation Requirements

While the proposed algorithm was able to pack all the raw data and fit within the bandwidth limitations, it could not fit within the tight time constraints imposed by the ATLAS Level1 Trigger rate (100 kHz). It became clear that new ideas were needed to fit within the time constraints. To be competitive with the existing approach meeting the following requirements was considered mandatory for new algorithm:

- compression formats should be simple enough to enable software fitting within the DSP Level 1 Trigger time constraints;
- reconstructed magnitudes should be easily accessible for HLT (no unpacking should be needed);
- the energy should be reconstructed with the same precision as in currently used Reco fragment;
- reliable theoretical model should exist allowing evaluate algorithm performance under various circumstances;
- in case of "good" signal in ALL or almost all channels, compression should be able to fit within the existing bandwidth limit;
- it should be possible to compress effectively both piled-up and "unexpected" signals.
Optimization of the DSP Resources

To improve DSP performance:

- all possible precalculations were moved outside the loops;
- specific DSP commands were used to increase the performance, such as built-in support for rounding;
- OF constants were rearranged to tune them for DSP commands possibilities;
- DSP uses a robust **Software-Pipelined Loop** mechanism that can significantly speed up loops without branching. This appeared important resource: eliminating 'if' statements from the loops and replacing them by arithmetical operations allowed to twice speed up the code.

Besides the other benefits, this trick allowed to include \( \text{Sum}(E_t, E_z, E) \) calculation into the DSP code.

Later this technique was imported into **Reco** and **Frag1** fragments.
Lossless Compression: Frag5

To meet the DSP time constraints it was decided to simplify calculations using the "standard" pulse shape as a reference point and to store differences between samples and the reference point rather than differences between consecutive samples.

The third version of the lossless compression algorithm called Frag5 fully benefits of this these improvements.

The closer recorded signal to the "standard" pulse shape assumed by the algorithm, the higher the compression of the data;

Worst case analysis shows that:

- non-physical or piled-up data which can not be described by any pulse shape can be recorded at 72 kHz rate at least;
- "reasonable" signals can be recorded at 80 kHz rate even if all channels simultaneously have very large energy deposits and are significantly out of time.
- in case of "standard" signal in all channels, with timing within $\pm5$ ns, data can be recorded at 95 kHz rate.
Data Compression: How It Works

reco: Amp, Time

raw: $s_1, s_2, s_3, s_4, s_5, s_6, s_7$

$s_1, \Delta_2, \Delta_3, \Delta_4, \Delta_5, \Delta_6, \Delta_7$

$A = \sum a_i \cdot s_i$

$A \cdot T = \sum b_i \cdot s_i$
## Frag5 Formats

<table>
<thead>
<tr>
<th>Format [code]</th>
<th>Size [bit]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>32</td>
<td>Empty channels.</td>
</tr>
<tr>
<td>ped4</td>
<td>40</td>
<td>High Gain; Amp &lt; 16; $</td>
</tr>
<tr>
<td>ped5</td>
<td>48</td>
<td>High Gain; Amp &lt; 16; $</td>
</tr>
<tr>
<td>amp5</td>
<td>56</td>
<td>Amp; Time; $</td>
</tr>
<tr>
<td>amp6</td>
<td>64</td>
<td>Amp; Time; $</td>
</tr>
<tr>
<td>raws</td>
<td>72</td>
<td>Amp; 8-9-10-x-10-9-8</td>
</tr>
<tr>
<td>rawf</td>
<td>80</td>
<td>Amp; 10-10-10-x-10-10-10</td>
</tr>
<tr>
<td>full</td>
<td>80</td>
<td>Saturated Amp; 10-10-10-10-10-10</td>
</tr>
<tr>
<td>dump</td>
<td>88</td>
<td>Amp; 10-10-10-10-10-10-10; (bad OF coefficients)</td>
</tr>
</tbody>
</table>
Outcome

By the end of Feb 2011 a fully functional version of Frag5 was successfully tested, a bit later it was documented.

- Several studies and tests performed on the system demonstrated the feasibility of the digits lossless compression approach.

- It noted that the increasing luminosity and energy of the LHC would pose more and more challenging conditions to the signal reconstruction.

- It considered very important to be ready with realistic upgrades of the DSP that could support lossless compression scheme and encouraged the continuation of studies and validation tests of this scheme with a realistic goal to have it implemented in the system.
Frag1: Threshold Selection

Due to some variations in the pulse shape QF appeared amplitude dependent. This made it impossible using predefined QF threshold for selecting samples for recording.

Several attempts were made to improve QF definition. Finally it was decided to simplify the decision procedure and to record all data over some Bandwidth Threshold ($B_{\text{Threshold}}$) regardless the value of QF. Initially this threshold was set to $S_{\text{max}} - S_{\text{min}} > 5$ ADC counts since the signals above this threshold are differently handled in online and offline reconstruction.

The subsequent runs with increased rate and luminosity showed however that statistics had changed and a number of samples overcoming this threshold exceeded the predefined limit of recorded channels set by current implementation. To retain a tractable amount of recorded samples the threshold was increased to $B_{\text{Threshold}} = 6$ ADC counts but this appeared a temporary way out.

Further increase of luminosity posed the $B_{\text{Threshold}}$ problem anew. Investigating "the 5 vs 6 or new ADC value in future" became one of the current priorities.
Last Changes in Current Scheme

It became clear that the current scheme required considerable modification. No reasonable threshold could be set to select required limited amount of channels for recording. It was decided to change the current Frag1 format and to implement some level of data compression similar to that of Frag5. New Frag1 format is still dropping a signal below the predefined bandwidth threshold but it is compressing all the others using two formats for small and for large signals. Quality Factor (QF) calculations are also ceased for dropped signals as they are time consuming. This means discarding information about the quality of signals below the threshold.
**Frag5: New Format ped3**

**ped3** (32 bits) – new Frag5 format for effective noise compression (used for high gain only). For comparison reason format is chosen in such a way, that estimation of its usage would be possible even without having all raw data. When reconstructed amplitude fits within 8 bits, minimal sample is within interval [16..79] and $S_{\text{max}} - S_{\text{min}} < 8$ ADC, we are sending:

- Amplitude 8 bits
- Minimum 6 bits [16..79]
- Residuals 18 bits = 3 bits $\times$ 6 samples

Information about Code is stored separately in 1 bit, thus adding this format does not affect existing Frag5 structure and is used as an additional layer optimized for noise compression only.

Taking into account, that pedestal level is $\sim 40$ ADC, we can use the estimation, that for noise signal minimal sample is expected to be in interval [16..79]. This means, that all raw samples, dropped by Reco+Frag1, may be compressed by **ped3** format without increasing the current bandwidth.
Frag5 Tests Performed in P1

- **Real case**: Heavy Ion collisions data Pb+Pb $\sqrt{s_{NN}} = 2.76$ TeV, $L_{\text{int}} = 1.35$ mb$^{-1}$.
  - Frag5 uses 87% of bandwidth

- **Worst case scenario**: Laser calibration run, large time jittering $\sim 25$ ns, signal in all channels $\sim 70$ ADC (0.8 GeV) which is unlike to happen in real data.
  - This causes the maximal ROD fragment size to exceed the bandwidth limit. Nevertheless all the data can be transferred at least at 93 kHz L1 Trigger rate.
Comparative Study

Reconstruction:

- **Reco** – uses standard OF online reconstruction algorithm, already validated;
- **Frag5** – uses the same standard OF algorithm, in principle it is also validated.

Recorded events:

- **Frag1**: selected channels only, the amount limited by bandwidth and data format (selected data \(\sim 18\%\) of channels)
  - **Pro**: saves essential information;
  - **Con**: definition of "essential" is bandwidth driven, no physics rationale.
- **Frag5**: all channels (100%)

Impact of increasing luminosity:

- **Frag1**: performance scaling – forced to elevate the bandwidth threshold, thus to loose valuable information;
- **Frag5**: much less affected than any version of Frag1, records ANY signal at 72 kHz and "reasonable" pile-up at 80 kHz rate.

Example: Laser run (signal \(\sim 70\) ADC, time jittering \(\sim 25\) ns):

- **Frag1**: will be forced to raise the BThreshold up to 70 ADC;
- **Frag5**: records all raw data at 93 kHz rate.
Comparative Study

TileROD fragment average size and RMS:

Cosmic run: 10000 events

- Frag1: aver. Size – 370.5 words, RMS – 59.4
  BThreshold = 5 ADC, ~ 11% of raw data;
- Frag5: aver. Size – 319.4 words, RMS – 3.2
  all raw data, reducing size by ~ 15% and RMS in ~20 times.

Physics run 189280: 7 TeV pp collisions, μ ~ 15.5

- Frag1: aver. Size – 357 words, BThreshold = 6 ADC;
- Frag5: aver. Size – 320 words, all raw data, reducing size by ~ 10%;

Frag5 Pros:

- Fragment size reduced by ~ 10% (for BThreshold = 6 ADC)
  the same remains valid for new versions of Frag5 and Frag1;
- More stability in fragment size;
- Helps to reduce ROL/DN payload, improves DN stability.
Maximal Average Load Distribution

Cosmic ray data
- CosmicCalo stream
\[ \sqrt{s} = 7 \text{ TeV} \text{ p-p collisions data} \]
- Express stream;
- \( L_{\text{int}} = 3030 \text{ nb}^{-1} \);
- \( \mu = 15.5 \pm 0.4 \);

Frag1 - Raw data for \( S_{\text{max}} - S_{\text{min}} > 6 \text{ ADC} \)

Frag5 - All raw data

For each of 32 Tile ROD fragments produced by 16 Long Barrel RODs, the moving average of fragment sizes over 16 consecutive events is calculated and the maximum of the 32 is selected.

- Number of events \( \sim 10k \)
- Bandwidth Limit 398 words.
- For 7 TeV p-p collisions sizes of Frag5 are estimated.
Implementation: Pros

- All the raw data can give more handle to cope with MB pileup or unforeseen problems. Allow using during analysis inter-channel dependencies and correlations that are beyond the online approach.
- Performance for high level objects (jets, MET, SUMET) is still under study. There is also a need to enforce efforts on small signals (muons).
- Having all the raw data may appear indispensable for exotic searches.
- Lossless compression simplifies data collection – no need of complicated estimates and threshold adjustments.
- It has already inspired current optimization of Frag1.
- Keeping the raw data makes it possible debugging and validation as well as offline reprocessing. Otherwise we have to rely completely on the online reconstruction performed in harsh time constraints with limited resources of fix-point DSP arithmetic.
- With all the raw data the offline processing is always open to further improvements.

**Is the current approach sufficient and complete?**

**Do we need all raw data?**

**If we can keep all raw data, should we have it?**

**Should we seek for any special physic case to drive the answers?**
We consider three main objections to implementing of Lossless Compression tool: consistency, maintainability, safety.

**Consistency:**

- Is the Lossless Compression approach consistent with data taking adopted in ATLAS experiment, say in Liquid Argon Calorimeter (LAr)?
  They are also dropping significant part of raw data.

Surely, **Frag5** is not consistent with what is used in LAr, but we can look at it from the other end: one can make it consistent implementing Lossless Compression in LAr.

Really, ATLAS Liquid Argon Calorimeter has a very similar DAQ and uses the same Optimal Filtering approach for data reconstruction. The LAr data exceed significantly those of the TileCal and take up almost a half of the ATLAS event size, thus compression here may appear particularly helpful. It is well known that LAr team had applied for additional space to store more data, but the application could not be accepted. So, the need to save more data exits. Lossless compression approach may be applied to the LAr to study the possibility of storing ALL the raw data without increasing the currently used bandwidth and with a minor (if any) increase of the currently used capacity.

In other words, this objection is rather a call to further research and development than an argument against the lossless compression.
Implementation: Cons

Maintainability:

- Lossless Compression code is about ten times longer and more complicated, so its maintenance cost will increase accordingly.

This objection is not "physics driven". We should first ask whether ATLAS experiment needs saving as much data as possible based on its objectives. If the answer is "Yes", then we can start counting the cost. As far as the problem is challenging, we should not expect the solution to be simple. In fact, complexity of Lossless Compression tool matches the complexity of the initial problem. The LHC experiment is the most sophisticated engine ever created by a man. It is not surprising that its maintenance requires extraordinaire efforts. Besides, the development of Frag1, announced implementation of compression formats in the current scheme shows that the problem of maintainability has been somewhat overestimated and surely it is tractable.

Safety:

- Is the new code safe? We have now a stable running of the system, and we are not going to jeopardize their present smooth operation without a real physics case.

To meet this objection Twin Mount Framework (TMF) has been proposed.
Twin Mount Framework

To ensure smooth incorporation of lossless compression scheme into the system and to avoid possible (if any) impact on data taking during the transition, it was proposed to make \textit{Frag5} interchangeable on-the-fly with the current scheme \textit{(Reco+Frag1)} using the benefit of the property that \textit{Frag1 BThreshold} is \texttt{configurable on-the-fly}. Essentially, it means creating the following switch:

\begin{verbatim}
if \texttt{BThreshold} = 0 then use \texttt{Frag5} else use \texttt{Reco+Frag1};
\end{verbatim}

This approach called \textit{Twin Mount Framework (TMF)} was successfully developed and tested in the system. It serves to increase the safety, stability and overall performance of data taking:

- it has no impact on current data taking;
- it provides the opportunity of unobstructed development, study and validation of lossless compression (as well as Reco+Frag1, if needed) with a realistic goal to have lossless compression implemented in the system as it is encouraged by the Tile Management Meeting.
High Rate Tests

- High Rate tests have shown, that installation of Twin Mount Framework does not affect performance;
  - i.e. Reco+Frag1 works at the same rate both with and without TMF.
- BUSY in EBC comes from a single module (EBC22);
- Frag5 works "as is" at 100 kHz rate;
- High Rate tests were run without collisions;
  - Note that in case of 7 TeV p-p collisions Frag5 fragments size is even less affected than Reco+Frag1.

<table>
<thead>
<tr>
<th></th>
<th>L1</th>
<th>L1A</th>
<th>Total</th>
<th>EBA</th>
<th>LBA</th>
<th>LBC</th>
<th>EBC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reco+Frag1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120 kHz</td>
<td>120</td>
<td>95</td>
<td>18.8%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.06%</td>
</tr>
<tr>
<td>100 kHz</td>
<td>100</td>
<td>89</td>
<td>8.9%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.15%</td>
</tr>
<tr>
<td>89 kHz</td>
<td>89</td>
<td>82</td>
<td>5.3%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.08%</td>
</tr>
<tr>
<td>84 kHz</td>
<td>84</td>
<td>78</td>
<td>4.5%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.05%</td>
</tr>
<tr>
<td><strong>Frag5</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120 kHz</td>
<td>120</td>
<td>95</td>
<td>18.8%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.011%</td>
</tr>
<tr>
<td>100 kHz</td>
<td>100</td>
<td>89</td>
<td>8.8%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.010%</td>
</tr>
<tr>
<td>89 kHz</td>
<td>89</td>
<td>82</td>
<td>5.2%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.005%</td>
</tr>
<tr>
<td>84 kHz</td>
<td>84</td>
<td>78</td>
<td>4.4%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.003%</td>
</tr>
</tbody>
</table>
Perspectives

The TileCal operation experience and TTM Outcome, as well as the evolution of currently used strategy indicate that using lossless compression is very likely in the nearest future. Current development of the system convinces that the problem with maintenance is tractable. Since compression is already accepted there is no reason to stop in half the way and drag along the problems of uncompressed approach. Whether it happens by directly installing Frag5 with helpful TMF (which seems preferable) or gradually importing its solutions into Frag1, the mission of the first lossless compression tool will be fulfilled.

Not all the resources of the Lossless Compression algorithm are exhausted. If needed, one may return, to the initial idea of ordering the cells, particularly keeping in mind coupling property of PMTs.

Application to LAr is also awaiting its turn. Should it be successful, we will have managed to record ALL the raw data for ATLAS Calorimetry (LAr and TileCal). This will be done without installing additional hardware to upgrade the detectors.
Conclusions

• A lossless compression tool (Frag5) is a fully functional software able to store all the TileCal raw data at 100 KHz rate fitting both within bandwidth and time limitations of ROD DSP. Some details of implementation have been highlighted and results of testing presented.

• Comparative study with respect to the existing operation scheme is performed considering Frag5 as a possible upgrade of the current data reconstruction and storing strategy.

• Evolution and current state of both systems have been traced indicating the importance of compression schemes in the ATLAS Calorimetry data processing.
Backup
Data Reconstruction

Restoring the initial values of **Amplitude** and **Time** from these data is called data reconstruction.

In a few years TileCal data processing passed a long way. To forecast further evolution we should first recall and compare its initial and current state.

Several reconstruction methods were developed within the TileCal collaboration, the most important being:

- **The Fit** - classical 3-parameter fitting of the above equation for $s(t)$ that has been initially used by default for test beam and commissioning cosmic reconstruction, both from real and simulated data;

- **The ManyAmps** - handling piled-up events by successively fitting auxiliary amplitudes coming from adjacent bunch crossings, that has been used by default for full ATLAS reconstruction with simulated data either with or without pile-up.

Both methods require considerable calculations that could not be accepted for online processing as the Level 1 Trigger planned rate (**100 kHz**) imposed severe time constraints on DSP code functionality.
Lossless Compression: V1

**ped** (3 + 32 bits) for pedestal like signal:
- Gain (1 bit) + Bad (1 bit) + Format (1 bit) = 3 bits
- 7 Samples: 7 + 4 + 4 + 4 + 4 + 4 + 5 = 32 bits
- $s_1$ 7 bits [0..127]
- $s_7 - s_1$ 5 bits [-16..15]
- $s_i - ped$ 4 bits [-8..7], $i = 2, 3, 4, 5, 6$, $ped = (s_7 + s_1) \div 2$

**full** (3 + 96 bits) for signal:
- Gain (1 bit) + Bad (1 bit) + Format (1 bit) = 3 bits
- Samples 70 bits = 10 bits $\times$ 7 samples
- Amplitude 15 bits
- Time 11 bits

![Diagram showing time and amplitude intervals with markers for $s_1$, $s_7$, $\Delta_2$, $\Delta_3$, $\Delta_4$, $\Delta_5$, $\Delta_6$]
ped4 and ped5

ped4 (40 bits) is only used for high gain signal (no information about gain needed) when reconstructed amplitude fits within 9 bits and residuals are in [-7..7] interval (i.e. fits in 4 bits). In this case we are sending:

- Code 4 bits
- Amplitude 9 bits
- Sample1 7 bits [0..127]
- Residuals 20 bits = 4 bits × 5 samples

ped5 (48 bits) is only used for high gain, small signal, but higher noise. Sample1 uses all 10 bits and the range for residuals is [-15..15] (i.e. fits in 5 bits). For small signal with small, but higher noise the compression switches to ped5 format. In this case we are sending:

- Code 4 bits
- Amplitude 9 bits
- Sample1 10 bits
- Residuals 25 bits = 5 bits × 5 samples
amp5 and amp6

amp5 (56 bits) format sends time with 0.5 ns precision using 6 bits. If first sample is in [0..127] interval, all 5 residuals are in [-15..15] (i.e. fits in 5 bits) and no overflow/underflow of amplitude, then we are sending:

- Gain (G) 1 bit (0 - low gain; 1 - high gain)
- Code 2 bits
- Amplitude 15 bits
- Time 6 bits
- Sample1 7 bits [0..127]
- Residuals 25 bits = 5 bits × 5 samples

amp6 (64 bits) format almost the same as amp5, Difference is in range for residuals, which for this format is [-31..31] (i.e. fits in 6 bits).

- Code 5 bits
- Residuals 30 bits = 6 bits × 5 samples
**raws and rawf**

**raws** (72 bits) format uses scheme 8-9-10-x-10-9-8, which corresponds to the number of bits used for each sample (sample 4 is not sent). This scheme will cover significant part of "reasonably" strange or piled-up signals and will work at 80 kHz.

![Sample Table]

**rawf** (80 bits) is used in case of very strange or piled-up signal. It sends amplitude and samples 1, 2, 3, 5, 6, 7 (all except sample 4) using 10 bits. Both formats reconstruct sample 4 based on sent information. In case of correct OF coefficients this format is able to transfer any data at 72 kHz rate.

![Sample Table]
**full and dump**

**full** (80 bits) format is used in case of amplitude overflow or underflow. In this case, instead of amplitude we send only overflow/underflow bit. We also send sample 4 as it is, because one can not reconstruct it correctly using truncated amplitude and 6 other samples.

```
G 0 0 0 0 0 0 1 A  S3  Sample 2  Sample 1
Smp 6  Sample 5  Sample 3  Sample 2
      Sample 7  Sample 6
```

**dump** (88 bits) existence of this format "warranty" that in any case, even if all Optimal Filtering Coefficients (OFC) are zero, we still will be able to store all raw data. If OF coefficients are correct, this format will not appear. The dump format sends gain, amplitude and all 7 samples without any compression.

```
G 1 1  Smp 2  Sample 1  Amplitude
Sample 5  Sample 4  Sample 3  Sample 2
      Sample 7  Sample 6  Smp 5
```
DSP Time Constraints

Several issues were preventing from time-effective implementation of the proposed algorithm:

- implementation required division which was not directly supported by DSP while emulation was time consuming;

- compressed formats did not contain reconstructed magnitudes that were needed online for HLT, and unpacking and reconstructing them on the HLT side was even more time consuming.
Digital Signal Processor (DSP)

Careful study of documentation revealed a robust computer power of the Digital Signal Processor (DSP).

- 2 general-purpose register files (A and B)
- 8 functional units (.L1, .L2, .S1, .S2, .M1, .M2, .D1, .D2)
- 64 registers (32-bit, 32 per register file)

DSP TMS320C64x

- .L – arithmetic, compare, logical, data packing, ...
- .S – arithmetic, shift, logical, data packing, ...
- .M – multiply, bit interleaving, shift, rotation, ...
- .D – load and store, shift, add, subtract, ...

Software-Pipelined Loop

A1 B1 C1 D1
A2 B2 C2 D2
A3 B3 C3 D3
A4 B4 C4 D4

Prolog
Kernel
Epilog
Elimination of 'if' Statements

\begin{align*}
\text{if} \ (a > b) \ &\ u = x; \\
\text{else} \ &\ u = 0; \\
\text{u} &\ = (a > b) \times x; \\
\text{for} \ (ch) \ &\ {\ldots} \\
\quad \text{bool} \ cond = \ldots; \\
\quad \text{if} \ (cond) \ &\ procA(\ldots); \\
\quad \text{else} \ &\ procB(\ldots); \\
\quad \ldots \\
\text{for} \ (ch) \ &\ {\ldots} \\
\quad \text{bool} \ cond = \ldots; \\
\quad chA[i] = ch; \ i += \ cond; \\
\quad chB[j] = ch; \ j += !cond; \\
\quad \ldots \\
\text{for} \ (ch \ in \ chA) \ &\ procA(\ldots); \\
\text{for} \ (ch \ in \ chB) \ &\ procB(\ldots); \\
\end{align*}
SumEt, SumEz, SumE

This trick allowed to find a solution for SumEt calculations

- ATLAS Level 2 Trigger is Region of Interest based and was not designed for global quantities such as MissingEt, but physics demands this!

- Calorimeter RODs (LAr, TileCal) calculates projective sums;
- SumEt was calculated at TileCal with single PMT compensation;
- Was time consuming, could work only up to 80 kHz only;
- Additional request from Level 2 trigger was to include SumEz and SumE calculations and sum up only over noise threshold;
- Algorithm was reimplemented using code speed up techniques; precalculation of sum coefficients allowed to fit Sum(Et, Ez, E) calculation in Frag5 at 100 kHz;
- Later this approach was imported to Reco fragment.
The **Optimal Filtering (OF)** method is based on the use of linear combinations of the signal samples \( s_i \) to obtain the amplitude of the pulse.

The coefficients of these combinations \((a_i, b_i, c_i)\) known as OF weights are chosen in such a way that the impact of the noise to the calorimeter resolution is minimized.

OF was proposed as a possible solution and due to its simplicity in the mathematical formulation was implemented in the core of the Digital Signal Processors (DSPs) in the Read-Out Drivers (RODs) for online energy reconstruction.

\[
A = \sum a_i \cdot s_i \\
A \cdot \tau = \sum b_i \cdot s_i \\
ped = \sum c_i \cdot s_i
\]
It was also proposed to record reconstructed Amplitude and Time together with some parameter describing the quality of reconstruction for further offline analysis.

While the Fit and ManyAmps used for this purpose conventional $\chi^2$, lots of studies were performed on the definition of this parameter called Quality Factor (QF) considering a feasible definition in terms of implementation in the ROD DSP and the goodness in the quality description of the reconstruction.

To meet the requirement of the DSP, instead of considering the square of the residuals, the absolute value was used.

Later QF calculation was improved. Now it almost corresponds to $\chi^2$, except that the pedestal is approximated by the first sample $s_1$.

In case if QF exceeded a designated threshold it was proposed to save selected raw data for further offline analysis. The amount of the raw data was determined by the output bandwidth limitations and the format of the data transfer.
The Read-Out Driver

The Read-Out Drivers are the core of the back-end electronics.

The TileCal reconstruction algorithms are implemented in Digital Signal Processors (DSP) placed in the RODs.

There are 2 processing units in a ROD, each of them containing 2 DSPs.
The TileROD Fragment

The overall output bandwidth for 4 modules at 100 kHz rate is 12800 bits = 400 words of 32 bits.

- Of these 2 words are used as S-Link Begin and S-Link End.

Thus for TileROD Fragment limit is 398 words.

- Of these 12 words are used for Header and Trailer.

Finally for each module we have (400 – 14) / 4 = 96.5 words (per module)

Each module sends information about Data Quality (digital errors, ...) 9 words.

Each module needs at least 3 words for fragment header.

Thus 84 words are left for sending 48 channels, in average ~ 56 bits per channel.
As it has already been noted, the input data contain 72 bits per channel.

In initial approach the ROD output for each module consisted of two fragments:

- **Reco** – contained the reconstructed data from 48 channels;
- **Frag1** – contained the raw data from the selected channels.

**reco** – reconstructed data packed into one 32 bit word:

<table>
<thead>
<tr>
<th>G</th>
<th>Amplitude</th>
<th>Time</th>
<th>B</th>
<th>QF</th>
</tr>
</thead>
</table>

- 1 bit – Gain (G);
- 15 bit – **Amplitude** (format 11.4 binary);
- 11 bit – **Time** (format ±6.4 binary);
- 1 bit – Bad bit (B);
- 4 bit – Quality Factor (QF).
Frag1 Fragment

Frag1 format is used to send raw data for the selected channels. With this approach we are able to send in average 7 channels of 45 (16%).

Frag1 raw data format for channel:

<table>
<thead>
<tr>
<th>Sample 1</th>
<th>G</th>
<th>Num</th>
<th>Chan ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample 7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 1 bit – Gain (G);
- 3 bit – Num: number of samples (always 7);
- 6 bit – Chan ID: channel ID;
- 70 bit = 10 bit × 7 samples;

Obviously only 63% of bandwidth was actually used. Rearranging the information for better fitting was rejected as time consuming.
Frag1: Threshold Selection

Due to some variations in the pulse shape QF appeared amplitude dependent. This made it impossible using predefined QF threshold for selecting samples for recording.

Several attempts were made to improve QF definition. Finally it was decided to simplify the decision procedure and to record all data over some bandwidth threshold regardless the value of QF.

In offline analysis the threshold of $5 \text{ ADC}$ counts is used to choose between the two reconstruction algorithms: if $S_{\text{max}} - S_{\text{min}} > 5 \text{ ADC}$ then iterative algorithm is used, otherwise the none-iterative one. So setting bandwidth threshold in Frag1 to $5 \text{ ADC}$ could easily motivate why we are dropping the data below the threshold: for these events the same algorithms are used both in online and offline analysis, so no need to keep the raw data.

The subsequent runs with increased rate and luminosity showed however that statistics had changed and a new threshold $\text{Th} = 6 \text{ ADC}$ counts was introduced to retain tractable amount of recorded samples.

Investigating "the 5 vs 6 or new ADC value in future" issue became one of the current priorities.

To the moment it has been decided to change current Frag1 format implementing some level of data compression similar to that of Frag5.
New Frag1 Fragment

New version of Frag1 format is dropping signal below threshold and compressing signal above threshold using two formats for small and for large signals.

- **fmt1** (48 bits) is used for small signals with $S_{\text{max}} - S_{\text{min}} < 16$ ADC counts but still beyond bandwidth threshold.

$$\text{Diff}_k = \text{Sample}_k - \text{Sample}_\text{min}.$$ 

- **fmt2** (80 bits) is used for $S_{\text{max}} - S_{\text{min}} \geq 16$ ADC.

<table>
<thead>
<tr>
<th>Chan ID</th>
<th>Sample min</th>
<th>Diff 7</th>
<th>Diff 5</th>
<th>Diff 6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Diff 4</td>
<td>Diff 3</td>
<td>Diff 1</td>
</tr>
</tbody>
</table>

where $\text{Diff}_k = \text{Sample}_k - \text{Sample}_\text{min}$. 

<table>
<thead>
<tr>
<th>Sample 3</th>
<th>Sample 2</th>
<th>Sample 1</th>
<th>G</th>
<th>Chan ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample 6</td>
<td>Sample 5</td>
<td>Sample 4</td>
<td>Sample 3</td>
<td>Sample 7</td>
</tr>
</tbody>
</table>