A Novel Approach to Detector Calibration
Parameter Determination and Detector Monitoring

Eduardo Rodrigues1 (on behalf of the LHCb Vertex Locator group)
1 SUPA, School of Physics and Astronomy, University of Glasgow, Glasgow, G12 8QQ, UK
E-mail: eduardo.rodrigues@glasgow.ac.uk

Abstract. The LHCb experiment is dedicated to searching for New Physics effects in the heavy flavour sector, precise measurements of CP violation and rare heavy meson decays. Precise tracking and vertexing around the interaction point is crucial in achieving these physics goals. The LHCb VELO (VErtex LOcator) silicon detector is the highest precision vertex detector at the LHC and is located at only 8 mm from the proton beams. The high spatial point precision (up to 4 µm single hit precision) is facilitated by a complex chain of processing algorithms to suppress noise and reconstruct clusters. These are implemented in FPGAs, with over one million individually optimised parameters. A novel approach has been developed to optimise the parameters, integrating their determination into the full software framework of the LHCb experiment. While FPGAs are commonly used in HEP, their parameters are typically set by standalone software. However, the VELO complex processing chain would only achieve optimal performance and full sensitivity monitoring through optimisation integrated with the full reconstruction framework of the experiment. This off-line software framework performs a bit perfect emulation of the FPGAs hardware processing, and is fully integrated into the LHCb software, rather than standalone. The software platform facilitates: developing and understanding the behaviour of the processing algorithms; optimizing the parameters of the algorithms that will be loaded into the FPGAs; and monitoring the performance of the detector. This framework has also been adopted by the Silicon Tracker detector of LHCb. This novel approach has been successfully applied to the collision data taken in 2010 by the LHCb experiment. The initial calibration of the VELO detector has been performed using the emulation suite. The software is used to monitor the noise of the detector, pedestals stability, hit reconstruction performance and other detector parameters.

1. The LHCb vertex detector
The LHCb experiment [1] at CERN’s Large Hadron Collider (LHC) is dedicated to the study of CP violation and rare b- and c-flavoured hadron decays. The physics goals of LHCb demand in particular excellent tracking and identification of secondary vertices from long-lived particles.

The LHCb detector is a forward spectrometer located around the interaction point 8 at the LHC. Collision particles first encounter the vertex locator (VELO), a movable silicon vertex detector designed to precisely measure the production and decay vertices of long-lived particles, thereby separating primary from secondary vertices. It is retracted by 29 mm during LHC beam injection, ramping and squeezing. The nominal closed position – with the detector at 8 mm from the proton beams – is only achieved once the LHC declares stable beams.

Each of the two VELO halves consists of a row of 21 silicon micro-strip tracking stations, each with radial ($R$) and azimuthal ($\phi$) measurement capabilities, for a total of 84 sensors.
The VELO is the highest precision vertex detector at the LHC. The present primary vertex resolutions on data are presented in Figs. 1 and 2. Contributing to this outstanding performance is notably the excellent hit resolution as low as 4 \( \mu \)m, as illustrated in Fig. 3: it is better than the binary resolution for all strip pitches. The improvement over binary resolution is achieved by making use of analogue charge readout from the detector. The deposition of charge across several strips, in particular at large track angles (i.e. above 7 degrees) with respect to the silicon planes, produces the improvement of the resolution with respect to the “reference” binary resolution.

![Figure 1. Vertex resolutions in x (red curve slightly above the other curve, upper fit box) and y (blue curve, lower fit box) as a function of track multiplicity in the data.](image1)

![Figure 2. Vertex resolution in z as a function of track multiplicity in the data. The line is the result of the fit to \( y = Z - \text{Const}/n\text{Tracks}^{\text{Power}} + \text{Epsilon} \).](image2)

![Figure 3. VELO hit resolution on data versus the silicon strip pitch for two bins of track projected angle. The curves through the data points represent linear fits.](image3)

2. Data processing and emulation

The analogue signals from the VELO front-end chips are first sent to the so-called TELL1 data acquisition boards where the digitisation takes place; the path is schematically illustrated in Fig. 4. The close to 180,000 VELO sensor read-out channels are read out at 1 MHz, resulting in a typical non-zero-suppressed (NZS) data rate around 200 Gbytes per second; in comparison zero-suppressed (ZS) data amounts solely to roughly 10 Gbytes per second. As a pre-requisite the TELL1 boards first synchronise the data and buffer them as necessary. The subsequent
processing runs on the 4 FPGAs per TELL1 board. Each such board processes data coming from one sensor, i.e. from 2048 (strip) channels.

The data are digested by a complex chain of processing algorithms to suppress noise and reconstruct (VELO) clusters, the necessary input to track finding. In detail the TELL1 firmware suite on the FPGAs runs the following algorithms that form the data processing chain:

- Pedestals subtraction;
- (Front-end chip cross-talk removal);
- (Cable cross-talk correction);
- Mean common mode noise subtraction;
- Channel reordering;
- (Linear common mode noise subtraction);
- Clusterisation.

In the above all steps in parentheses are possible but disabled at present.

As can be seen from Fig. 5 the raw uncorrected NZS data from a typical Beetle chip exhibits ADC value (pedestal) variations at the level of the 4 analogue links on the chip and also at the channel level. The data processing chain starts by correcting pedestal offsets. The algorithm that determines the per-channel pedestal offset values to be subtracted from the raw data is described in [2]; the pedestals are typically around 512 ADC counts. After pedestal correction the data are well centered around 0 ADC counts, see Fig. 6.

The data are then corrected for mean common mode noise. The subsequent channel reordering phase simply converts the channel numbers to a strip numbering scheme that follows the $R$ and $\phi$ geometry of the sensors. It facilitates the manipulation of the data in the next processing step, the cluster finding procedure.

The clustering algorithm consists of two main modules: the first module is responsible for the detection of VELO hits whereas the second one performs the actual clusterisation and outputs the ZS data. The algorithm is run in parallel in a number of processing units (stream processors) implemented on the FPGA chips (each TELL1 board has 4 of these), with each logical processing unit taking care of a portion of raw NZS data. The processing proceeds as follows. First, the
input pedestal-subtracted data are compared to the seeding (hit) thresholds and possible seeds are determined. Next, the seeding strips are used to perform inclusion of the neighbouring strips and form clusters. All neighbouring strips above an inclusion threshold are added to the seeds.

![Pedestal-corrected data](image1)

**Figure 5.** Typical raw NZS data ADC values for a single Beetle chip with visible variations at link and channel level.

![Decoded raw data](image2)

**Figure 6.** Data corrected for pedestals. All values are well centered around 0 ADC counts.

In total around one million TELL1 parameters are required to run the full system. These processing, or calibration, parameters need to be individually optimised to overall achieve the outstanding detector performance described above.

In a “standard approach”, FPGA algorithm parameters are typically determined via standalone calculations or measurements. But due to (a) a complex chain of algorithms and (b) a very large number of configuration parameters, a novel approach has been developed in view of achieving the best possible performance in a very flexible way. It uses NZS data and relies on a bit-perfect emulation of the hardware processing taking place on the FPGAs. Furthermore, as the firmware on the FPGAs works with integers, a dedicated TELL1 software emulation is necessary in order to mimic the integer operations.

In our approach the determination of the detector parameters has been fully integrated in the data processing framework itself and consequently integrated also in the LHCb software framework. Apart from software maintenance benefits and alike, the approach has the advantage of naturally providing a means to optimise the parameters for the best physics performance given that the whole reconstruction sequence from the raw data to the formation of clusters and tracks can be executed within the framework. The framework is introduced in the next section.

### 3. The Vetra software framework

The Vetra software project [3] has been developed to provide a unique framework for the determination and optimisation of the various types of TELL1 parameters. It is fully integrated in the LHCb software framework, is highly flexible and configurable with Python.

At the heart of the framework is the emulation of the TELL1 algorithms that run on the FPGAs (VHDL code), implemented in C and mimicking the whole data processing sequence. The Vetra processing code can treat the same data formats as possibly output by the TELL1s, *i.e.* mainly:

- Zero-suppressed (ZS) banks: the main input to the reconstruction in the form of encoded reconstructed hits (clusters from hits). Each sensor produces one ZS data bank per trigger;
- Non-zero-suppressed (NZS) banks: full data as read out by a sensor (it’s the main input to the TELL1 emulation);
• Error banks: contain information on possible errors produced by the TELL1 boards.

Mimicking the data processing described in Sec. 2 the TELL1 emulator runs algorithms notably for pedestals subtraction, common-mode noise suppression and clusterisation.

Apart from the standard processing of the various bank types and (online and offline) monitoring jobs, Vetra can also perform more complicated tasks. In detail these are:

• Processing of any type of VELO raw data and;
• Monitoring of relevant quantities such as noise, pedestals, clusters, error banks, tracks, etc. (each bank type decoding has a corresponding monitoring sequence);
• Determination and optimisation of TELL1 parameters and subsequent quality check;
• Cross-check of the bit perfectness of the TELL1 emulation.

Having the ability to process NZS data from the TELL1s with Vetra makes it possible to develop and test new TELL1 algorithms prior to using them in the hardware itself. On top of this the framework can be and has been used in beam tests and in VELO upgrade projects.

4. Detector parameter determination and optimisation

The outline of the procedure for the determination and optimisation of the close to one million TELL1 parameters is as follows:

(i) Determine the new parameters running the emulation on a data file with the set of specialised (emulation) algorithms;
(ii) Store the newly calculated parameters in a database;
(iii) Run a confirmation job over another data file making use of the monitoring tools;
(iv) Upload the new parameters onto the TELL1s.

4.1. The VELO conditions database

The emulation can either retrieve the necessary processing parameters from external user options or, by default, from XML files stored on a database – the VELOCOND.db SQLite database. In both cases they are typically the set determined in the last optimisation. The very same information stored in VELOCOND.db is used both in the algorithms running on the FPGAs and in the corresponding Vetra software emulation.

4.2. TELL1 processing parameters training

Among the TELL1 configuration parameters the most important ones are the pedestals and the clusterisation thresholds (cf. Sec. 2). The combination of NZS data with the usage of the TELL1 emulator provides a means of tuning them. The whole procedure therefore relies by construction on the bit perfectness of the emulation, a subject discussed in Sec. 4.4.

As briefly mentioned in Sec. 2, the pedestals are obtained from noise NZS data with an averaging procedure that brings the data to values centered around 0 ADC counts. The determination of the clusterisation seeding and inclusion thresholds is more evolved; it requires a “threshold scan”. First the seeding thresholds are chosen scanning for possible values up to ≈ 20 ADC counts, having as a constraint a rate of clusters formed from noise at most at the 10^{-5} level. Next the inclusion thresholds are determined having in mind the resolution achieved. On the one hand the inclusion thresholds cannot be too high, as it would result in poor resolutions from too large a percentage of 1-strip clusters. On the other hand they cannot be too low, allowing for too many fake “fat clusters” that again could deteriorate the resolution. Typically the seeding thresholds are set to a factor 5-6 above the noise level (the average noise is about 2 ADC counts) whereas the inclusion thresholds are roughly 40% of the seeding values.

Once finalised, the new parameters are stored in the VELOCOND.db database. They can then be used subsequently in software, e.g. in the confirmation job.
4.3. **TELL1 processing parameters confirmation**

The confirmation phase constitutes the second part of the parameters determination procedure. After the training has been done and the VELOCOND.db database updated, the appropriate checking of noise, pedestals, etc. is performed with the monitoring tools available in Vetra. If the confirmation job is successful – the monitoring histograms show the expected results – the new set of calibration parameters gets uploaded onto the TELL1 boards. In parallel they are kept in the database, as said above, ensuring one is at all times able to perform a bit-perfect software emulation of what really happens during the processing of the real data.

4.4. **Emulation bit-perfectness**

This self-consistency check is a condition *sine qua non* for the usability of the emulation in the determination and optimisation of the TELL1 parameters. The check involves the comparison of the real TELL1 ZS banks (the VELO clusters) with the emulated ones produced by the TELL1 emulator starting from NZS banks - they have to be identical. The two “routes” are:

**Normal data processing** : TELL1 boards $\rightarrow$ TELL1 processing $\rightarrow$ ZS banks = clusters;

**Emulation** : TELL1 boards $\rightarrow$ NZS banks $\rightarrow$ TELL1 emulation $\rightarrow$ emulated ZS banks = emulated clusters.

The bit-perfectness check requires in practice the generation of special data in test units on the TELL1 boards. Full control is achieved knowing exactly what data is generated and input to the processing.

5. **Data monitoring**

As already mentioned, the VELO data quality monitoring software utilises the same (Vetra) framework and tools as the ones used for the determination of the necessary TELL1 processing parameters. Regular monitoring of the detector has been done on a daily basis from the start of the commissioning activities to the end of the first collision data taking period in 2010 – the initial calibration of the VELO detector had been performed using the emulation suite. Our novel approach has been successfully applied to monitor the noise of the detector, pedestals stability, hit reconstruction performance and other detector parameters.

6. **Experience from the LHC 2010 run**

A constant monitoring of the detector calibration parameters has proved the stability of the system. On seldom occasions certain parameters had to be retuned and uploaded to the TELL1 boards, notably to cope with increases in data rate due to the steady increase in LHC luminosity. The approach – framework and procedures – described above has been successfully applied to collision data taken in 2010. It proved that the integration in the data processing framework was the choice to make.

The detector has produced high quality data since the start of data taking, as a result of the parameter tuning. It is also worth mentioning that the silicon tracker group responsible for the downstream tracking stations also uses the Vetra software framework.

**References**

[1] The LHCb Collaboration, The LHCb Detector at the LHC, *JINST* 3 (2008) S08005.

[2] T. Szumlak, C. Parkes, Application of the Pedestal Following Algorithm to the VELO Detector, LHCb Note 2009-036, 2009.

[3] T. Szumlak, C. Parkes, Description of the Vetra Project and its Application for the VELO Detector, LHCb Note 2008-022, 2008.