FPGA coprocessing in the Upgrade II ECAL electronics

ECAL Upgrade II Workshop — IJCLab, Orsay

Riley Henderson

Table of Contents

1. Introduction

2. ECAL readout architecture
   - Readout architecture
   - Front-end electronics
   - Back-end electronics

3. FPGA coprocessing
   - Algorithmic possibilities
   - High level synthesis

Table of Contents

1. Introduction

2. ECAL readout architecture
   - Readout architecture
   - Front-end electronics
   - Back-end electronics

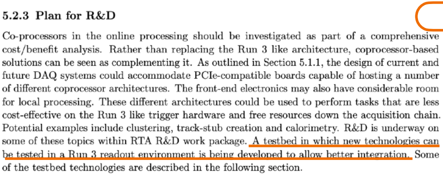
3. FPGA coprocessing
   - Algorithmic possibilities
   - High level synthesis

Introduction — data processing

The ECAL data rate in Upgrade II conditions will be immense!

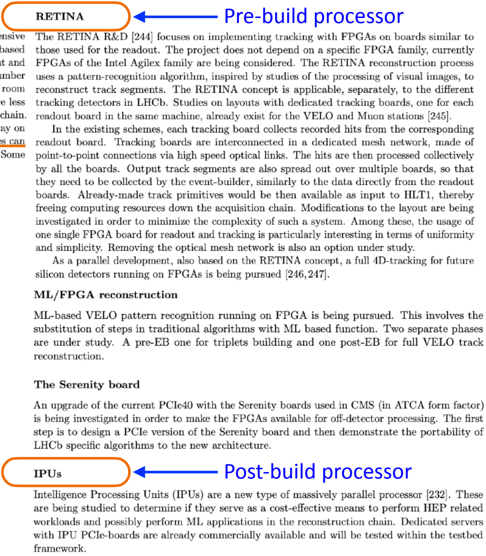
- Under a reasonable set of assumptions regarding the occupancy, granularity, dynamic range, etc.; the estimated full data rate would be $\mathcal{O}(30\text{Tb/s})$.
- Sparsification can bring this down to $\mathcal{O}(4\text{Tb/s})$, but this is still very large.
- Other means of data reduction — potentially within, or following the front-end boards (FEBs) — will be necessary to allow the back-end (BE) electronics to cope.
- If some early stages of reconstruction can be implemented as upstream as possible, this could significantly reduce the load on the subsequent data acquisition (DAQ) and event builder (EB) electronics.

R&D activities planned in the FTDR



5.2.3 Plan for R&D

Co-processors in the online processing should be investigated as part of a comprehensive cost/benefit analysis. Rather than replacing the Run 3 like architecture, coprocessor-based solutions can be seen as complementing it. As outlined in Section 5.1.1, the design of current and future DAQ systems could accommodate PCIe-compatible boards capable of hosting a number of different coprocessor architectures. The front-end electronics may also have considerable room for local processing. These different architectures could be used to perform tasks that are less cost-effective on the Run 3 like trigger hardware and free resources down the acquisition chain. Potential examples include clustering, track-stub creation and calorimetry. R&D is underway on some of these topics within RTA R&D work package. A testbed in which new technologies can be tested in a Run 3 readout environment is being developed to allow better integration. Some of the testbed technologies are described in the following section.



RETINA

Pre-build processor

The RETINA R&D [244] focuses on implementing tracking with FPGAs on boards similar to those used for the readout. The project does not depend on a specific FPGA family, currently FPGAs of the Intel Agilex family are being considered. The RETINA reconstruction process uses a pattern-recognition algorithm, inspired by studies of the processing of visual images, to reconstruct track segments. The RETINA concept is applicable, separately, to the different tracking detectors in LHCb. Studies on layouts with dedicated tracking boards, one for each readout board in the same machine, already exist for the VELO and Muon stations [245].

In the existing schemes, each tracking board collects recorded hits from the corresponding readout board. Tracking boards are interconnected in a dedicated mesh network, made of point-to-point connections via high speed optical links. The hits are then processed collectively by all the boards. Output track segments are also spread out over multiple boards, so that they need to be collected by the event-builder, similarly to the data directly from the readout boards. Already-made track primitives would be then available as input to HLT1, thereby freeing computing resources down the acquisition chain. Modifications to the layout are being investigated in order to minimize the complexity of such a system. Among these, the usage of one single FPGA board for readout and tracking is particularly interesting in terms of uniformity and simplicity. Removing the optical mesh network is also an option under study.

As a parallel development, also based on the RETINA concept, a full 4D-tracking for future silicon detectors running on FPGAs is being pursued [246, 247].

ML/FPGA reconstruction

ML-based VELO pattern recognition running on FPGA is being pursued. This involves the substitution of steps in traditional algorithms with ML based function. Two separate phases are under study. A pre-EB one for triplets building and one post-EB for full VELO track reconstruction.

The Serenity board

An upgrade of the current PCIe40 with the Serenity boards used in CMS (in ATCA form factor) is being investigated in order to make the FPGAs available for off-detector processing. The first step is to design a PCIe version of the Serenity board and then demonstrate the portability of LHCb specific algorithms to the new architecture.

IPUs

Post-build processor

Intelligence Processing Units (IPUs) are a new type of massively parallel processor [232]. These are being studied to determine if they serve as a cost-effective means to perform HEP related workloads and possibly perform ML applications in the reconstruction chain. Dedicated servers with IPU PCIe-boards are already commercially available and will be tested within the testbed framework.

See "*Recent testbed results on technologies for future RTA*" by G. Punzi during 106th LHCb week, Dec 2022.

A Vision for Future: primitive-based reconstruction

- There may be ways to increase throughput in post-build processing, but not yet clear.
- Results have shown that it is doable to embed track reconstruction in a compact FPGA system
- Can pre-evaluate primitives (track segments, calo clusters, muon stubs....) during readout (before EB)
  - First step is T-station seeding in Run-4 (Downstream Tracker, under review by U2PG)
- More power available to HLT1
- (Side Note: All this might be based on the same FPGA cards used for the readout)

- Further steps:
  - Consider dropping raw data not associated to the primitives -> reduce B/W by O(10x)
  - Perform reconstruction only ONCE - out of *primitives*, not raw data; preserve HLT1 output to HLT2

- Expected (hoped for) consequences:

1. Reduction of HLT2 workload and data-flow

   => enable more complex detectors

   => enable more selective trigger selections
2. If can do real-time alignment at primitive level => drop buffer between HLT1 e HLT2
3. Greener solution - FPGAs ~50x more energy efficient that CPU/GPUs

=> improve physics and save money - might help in 'scoping' process.

RTA-2 will take work. but we can make it !

See *"Recent testbed results on technologies for future RTA"* by G. Punzi during 106th LHCb week, Dec 2022.

Table of Contents

1 Introduction

2 ECAL readout architecture

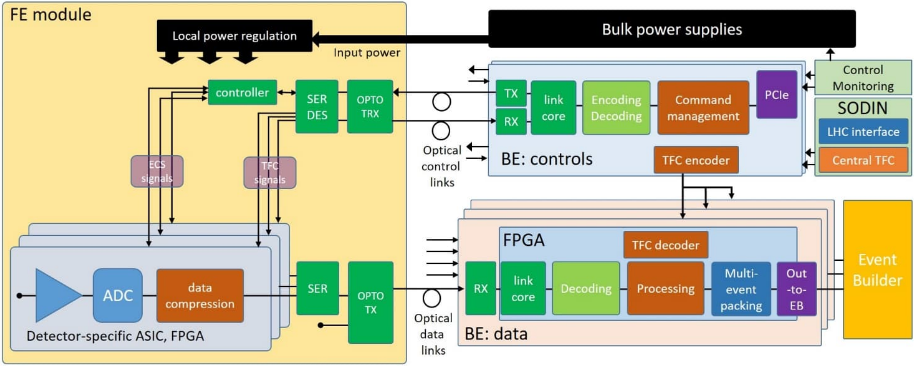
- Readout architecture
- Front-end electronics
- Back-end electronics

3 FPGA coprocessing

- Algorithmic possibilities
- High level synthesis

ECAL readout electronics

- The ECAL readout electronics will feature a range of ASICs and FPGAs of varying sizes, designed to handle a range of different tasks.
- These tasks vary in complexity and many of the implementation details are not yet finalised.
- The general readout architecture is described schematically as follows:

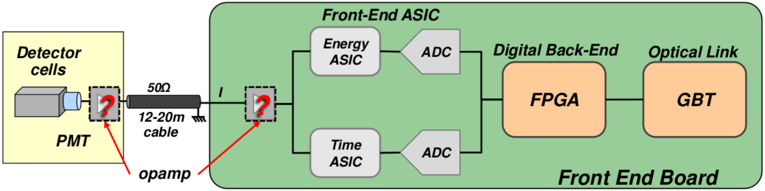


LHCb detector-level readout architecture described here: https://cds.cern.ch/record/2813379.

Front-end electronics

Currently, the FEBs will perform the following tasks:

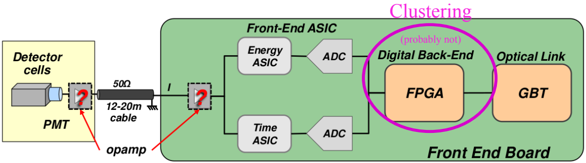
- ASICs + ADCs:
  - waveform sampling ⟶ energy (ICECAL) and timing measurements (SPIDER) — ASICs under development.
  - This will result in 12-bit energy, 11-bit timing measurements.
- FPGAs:
  - Energy measurement calibration.
  - Noise removal — pedestal subtraction and spill-over corrections.
  - Low-level trigger calculation (previously).
  - Sparsification and formatting.



Preliminary readout architecture described at recent electronics workshop.

FPGA coprocessing — FE

- In the FEBs there will likely not be many spare resources available since the FPGAs are quite small and busy.
    - In Run 3, each FEB has 3 FPGAs with 25–150k logic elements and 1–5Mb on-chip RAM.

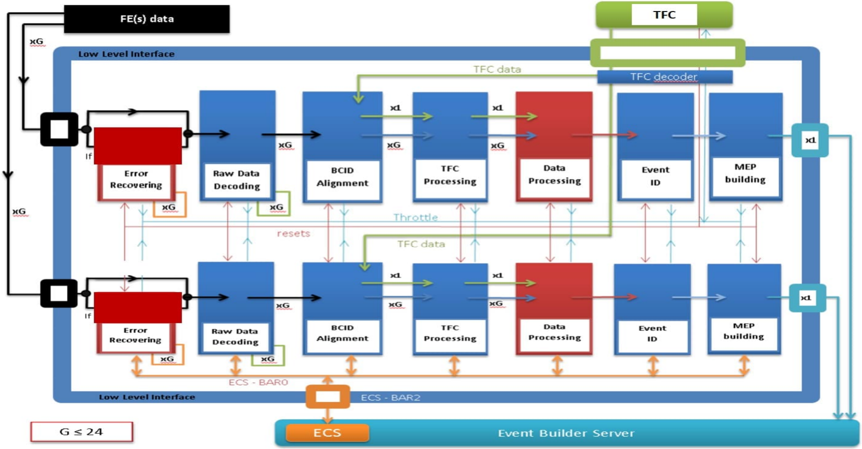


Preliminary readout architecture described at recent electronics workshop.

- The FPGA resources will be quite limited in the FEBs but we may consider implementing some "shortcut" clustering for trivial cases — e.g. when an isolated cluster is fully contained within a single readout block.

Back-end electronics

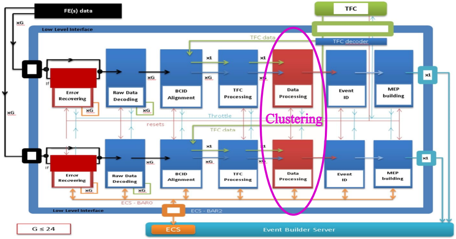
Following the FEBs, the data is sent to the main data acquisition (DAQ) readout boards to be interfaced with the data coming from the rest of the detector.



LHCb back-end readout architecture described here: https://cds.cern.ch/record/2813379.

FPGA coprocessing — BE

- The data processing module in the BE boards could, in principle, be a viable place to implement the clustering.



LHCb back-end readout architecture described here: https://cds.cern.ch/record/2813379.

- It is forseen that the BE DAQ boards will consist of PCIe400 readout boards (an upgrade of the current PCIe40), which will contain:
  - Intel Agilex FPGAs: 2–4M logic elements, 200–400Mb on-chip RAM.

Table of Contents

1. Introduction

2. ECAL readout architecture
   - Readout architecture
   - Front-end electronics
   - Back-end electronics

3. FPGA coprocessing
   - Algorithmic possibilities
   - High level synthesis

FPGA custering algorithms

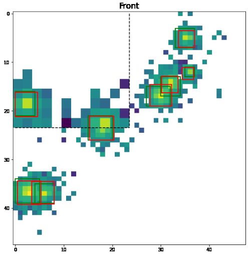
Assuming we have sufficient FPGA resources to implement upstream clustering, what algorithms might we consider?

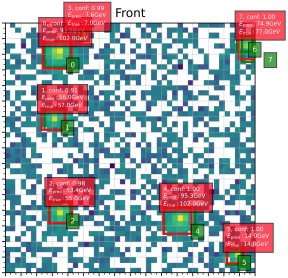
- Currently there are a number of potential approaches to the clustering:
  1. Cellular automaton: the approach used in Runs 1 and 2.
  2. Graph clustering: the new approach used in Run 3.
  3. *Neural network: a completely new approach for Upgrade II, potentially.*
- Translating one of the traditional clustering algorithms into an FPGA coprocessing context is probably the most straightfoward to implement.
- *However*, it remains unclear how to best to incorporate all of the new information available, *i.e.* z segmentation and timing.

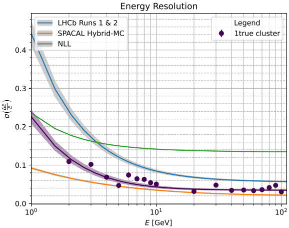
Why neural networks?

Clustering with NNs can solve all of these problems!

- Conceptually, a CNN directly analyses a $(H_{\text{in}}, W_{\text{in}}, C_{\text{in}})$ input array which means *all* dimensions can be naturally taken into account.
- Graph NNs (GNNs) have been shown to be capable of handling irregular geometries and directly identifying clusters which cross the boundary.







Single-electron energy resolution in a SPACAL calorimeter using a pure-CNN clustering model.

HLS studies

- Studying NNs in an FPGA environment is actually quite easy nowadays thanks to high level synthesis (HLS) tools such as `hls4ml`.
- For an 8-bit quantised CNN with 2 Conv2D (4, 4) layers, 1 ReLU, and 1 MaxPooling2D (3, 3) layer, hls4ml gives the following estimates using the Xilinx Vivado® backend:

Input size	**Latency**	**#LUTs (% avail.)**	**#FFs (% avail.)**	**Total #LEs**
$8 \times 8 \times 2$	0.72 µs	3.27×10^5 (18)	0.68×10^5 (1)	3.95×10^5
$16 \times 16 \times 2$	2.0 µs	3.12×10^5 (18)	0.49×10^5 (1)	3.61×10^5
$24 \times 24 \times 2$	4.2 µs	3.21×10^5 (18)	0.57×10^5 (1)	3.78×10^5
$48 \times 48 \times 2$	14.5 µs	3.25×10^5 (18)	0.46×10^5 (1)	3.71×10^5
$8 \times 8 \times 4$	0.72 µs	4.45×10^5 (25)	0.74×10^5 (2)	5.19×10^5

Percentage resource usage quoted for a Xilinx xcu250-figd2104-2L-e FPGA.[1]

- Determining whether such numbers might actually look realistic and/or attractive would require input from people who know more about the readout electronics than myself.

[1] Convolutional layers are not currently implemented in `hls4ml` with the Intel® Quartus® backend.

Summary and conclusions

The Upgrade II FTDR says...

> *"Co-processors in the online processing should be investigated as part of a comprehensive cost/benefit analysis. Rather than replacing the Run 3 like architecture, coprocessor-based solutions can be seen as complimenting it."*

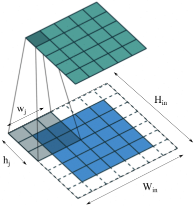
- The ECAL readout is a logical place for this and could provide significant benefit to the latter processing stages (and the overall detector energy budget) by implementing clustering early-on.
- This would represent a significant change to the reconstruction and would be no small task to implement $\implies$ R&D efforts should be ramped up as soon as possible.
- Requires strong collaboration between physics and electronics experts!

Thank for listening!

Backup slides

CNN latency & resource usage

- Internally, a CNN model (for example) consists of a number N_l of 2D convolution and/or pooling layers, which each consists of n_j *filters* of size (h_j, w_j, c_j).



- For a given level of parallelisation (reuse factor), the FPGA latency and resource usage depend entirely on these parameters $\{H_{in}, W_{in}, C_{in}, N_l, n_j, h_j, w_j, c_j\}$:
  - The *latency* is almost perfectly linear in the input size — *i.e.* the total number of readout channels processed.
  - The *resource usage* is essentially fixed for a given filter size.

Various ML approaches

- Yesterday, Alexey Boldyrev presented a study on the use of ML models to, in some sense, complement the existing clustering method.
    - This method uses to ML models to (re)process the signals surrounding the clusters seeds identified with a prior clustering method.
- Here I will instead focus on a pure NN model which naturally utilises all of the available input information and performs all tasks simultaneously — in other words, it would *supersede* the existing clustering methods.
- However, all approaches are worth considering and it is a good time to combine and coordinate efforts on the UII reconstruction studies.