Emerging Technologies and Development Methodologies for High-Throughput Data Acquisition Systems

Candidate:
Alberto Perro

Supervisors:
Renaud Le Gac
Olivier Leroy
Paolo Durante

Jury:
Mossadek TALBY
Pascal VINCENT
Cynthia HADJIDAKIS
Vincent TISSERAND

PhD Dissertation - Aix-Marseille University, CPPM/IN2P3/CNRS - 3rd July 2025

Index

Physics at LHCb

LHCb wide physics' program includes **flavour physics,** heavy ions, and searches for New Physics using **precision measurements** and **rare decay processes**.

During **Run 1 and 2 (2010-2018)** LHCb collected **~9 fb^{-1}**, delivering high precision in flavour physics, setting strong limits for New Physics and complementing direct searches from ATLAS and CMS.

In **Run 3 and 4 (2022-2033)**, LHCb aims to collect **50 fb^{-1}**, greatly increasing the sample size for B and D final states. LHCb will focus on CP violation tests due to their strong experimental theoretical constraints.

Topics investigated include:

- CP violation in B mesons
- Rare Decays
- Heavy Ions
- Hadron Properties focusing on B and D mesons
- CP violations in the *charm* sector
- Quantum Chromo-Dynamics
- Exotic Hadrons

Physics at LHCb: High Luminosity

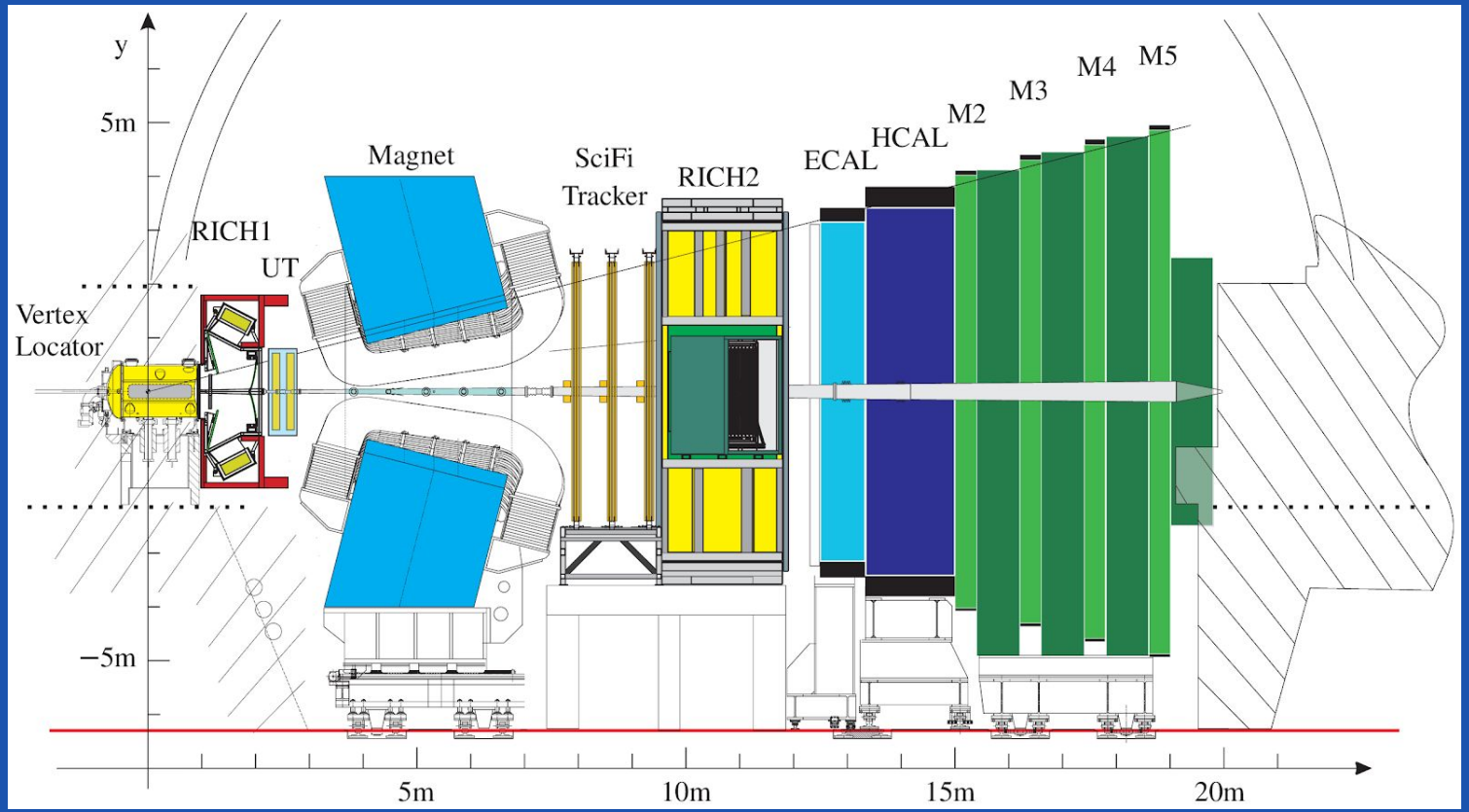
From Run 5 (2036-2041), the upgrade to the accelerator will deliver up to **300 fb^{-1}** to LHCb. This will require a major upgrade of the detector, called **LHCb Upgrade II.**

The number of proton-proton interactions per bunch crossing is expected to reach **40**, **8 times higher** than the current conditions.

This will pose significant challenges:

- Accurate identification of tracks and vertices
- Enhanced radiation damage to the detectors
- Increased data throughput towards the data acquisition system

The LHCb Experiment (2022-2033)



Online DAQ System Overview (2022-2026)

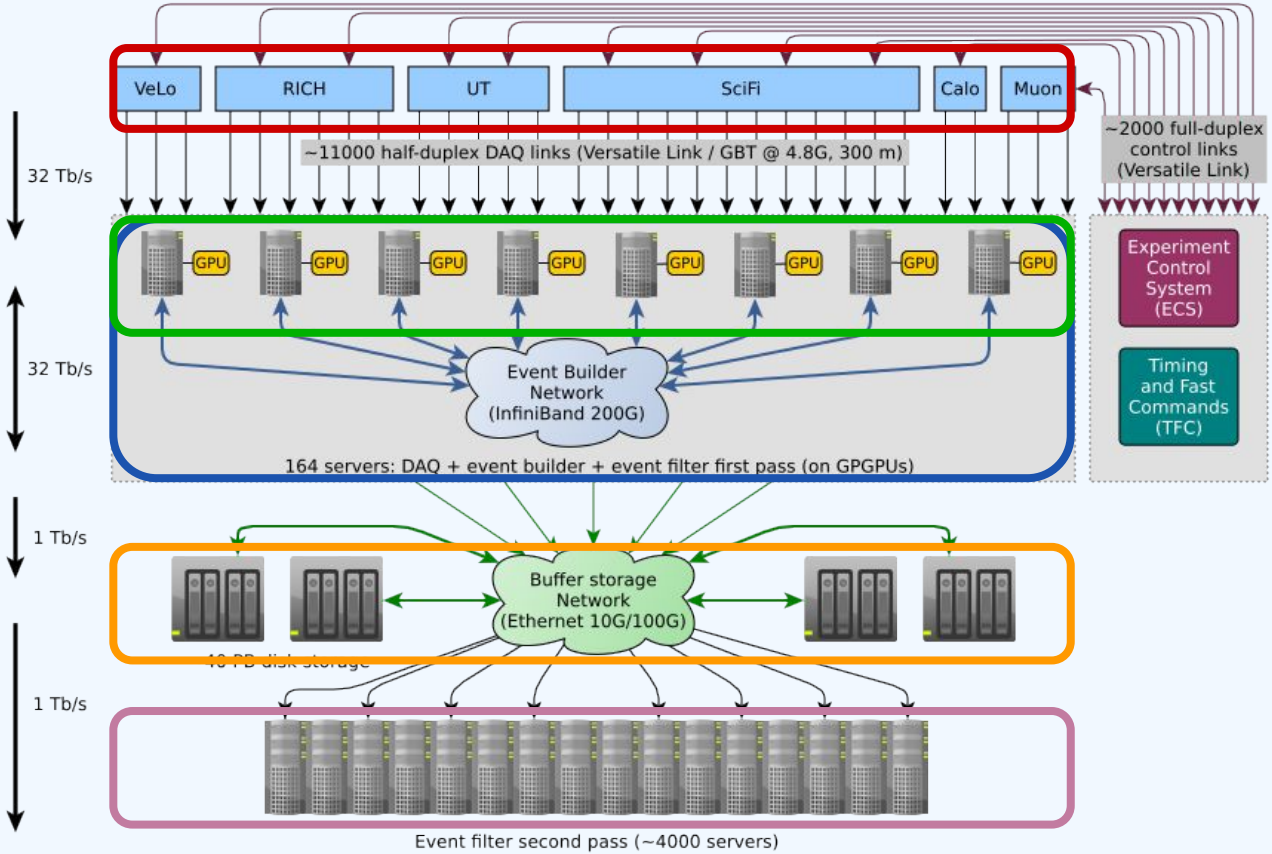
Front-End Electronics

DAQ FPGA Cards

Event Builder
and
High Level Trigger 1

Storage Buffer

HLT2



Front End Electronics

Front-End Electronics (FEE) is the first step in the DAQ chain.

FEE it is **tailored to the specific detector**, usually adopting Application Specific Integrated Circuits (**ASICs**) to acquire and digitize the sensors' signals.

Once digitized, the data is sent via high-speed links towards the Back-End.

In Run 3, FEE are handling the full **40 MHz** bunch-crossing rate. Event data transmission is achieved using the radiation resistant **GBT** chipset.

The **GBT protocol** simplifies back-end design and integrate control and monitoring features for the FEE.

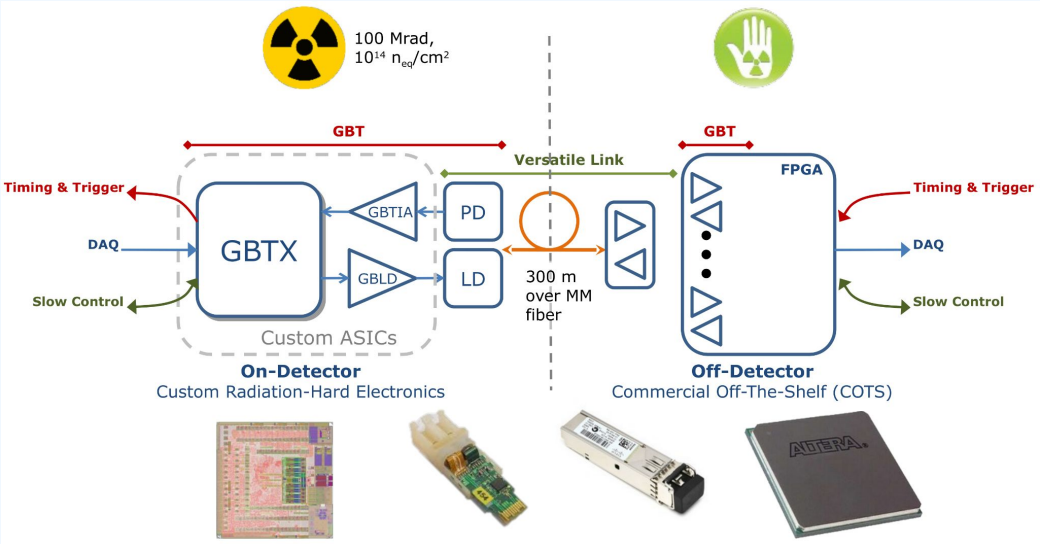
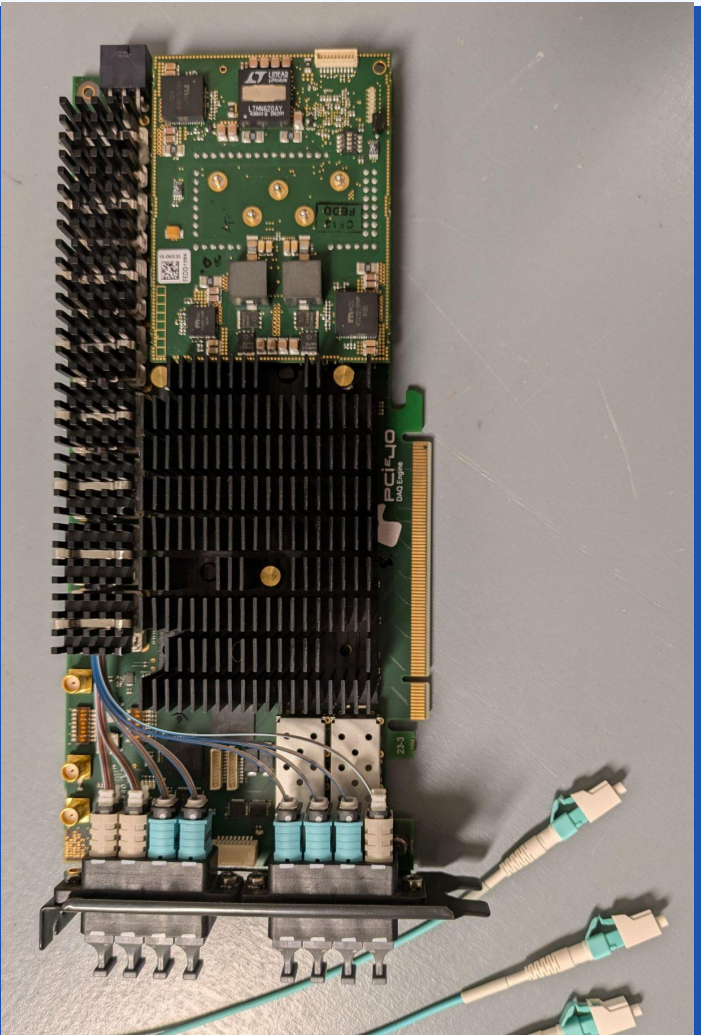


Diagram of the GBT communication system.

10.5170/CERN-2009-006.342



PCIe40 FPGA Cards

- 1.2M Logic Elements
- 48 GBT Radiation Hard Optical Links @ 4.8Gbps
- Two SFP+ modules
- Two PCIe Gen3 x8 interfaces
- Designed by CPPM, Marseille

520 cards are used in LHCb, subdivided in three flavors:

- SODIN: Readout supervisor, source of real-time control commands and entry point for the LHC bunch clock.
- SOL40: It interfaces to the slow and fast control of the FEE and to the fast control of the DAQ cards.
- TELL40: It acquires data from the FEE, processes it, and delivers it to the host through the PCIe interface.

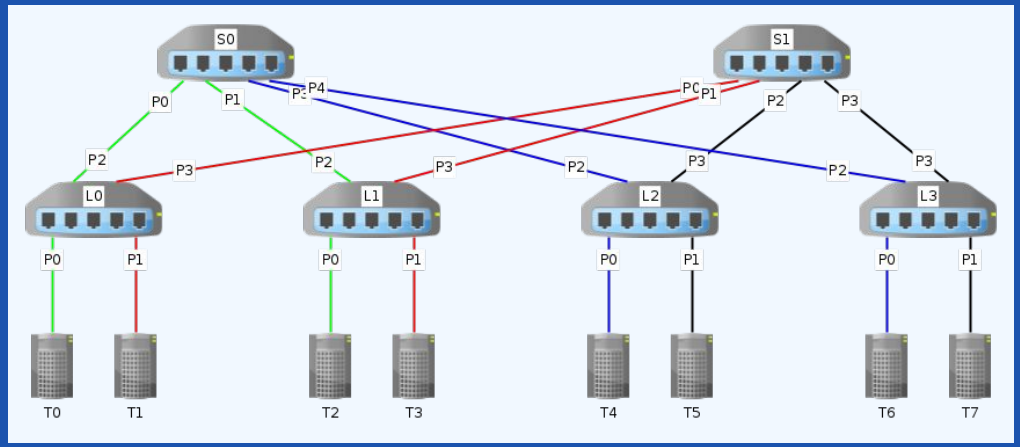
Event Builder Farm

With a full data rate of 32 Tb/s, ~300 nodes are required (300 * 107 Gb/s).

Each node hosts two PCIe40 cards (TELL40)

Fragments are scattered over all the EB nodes, so a network is necessary to perform the collection.

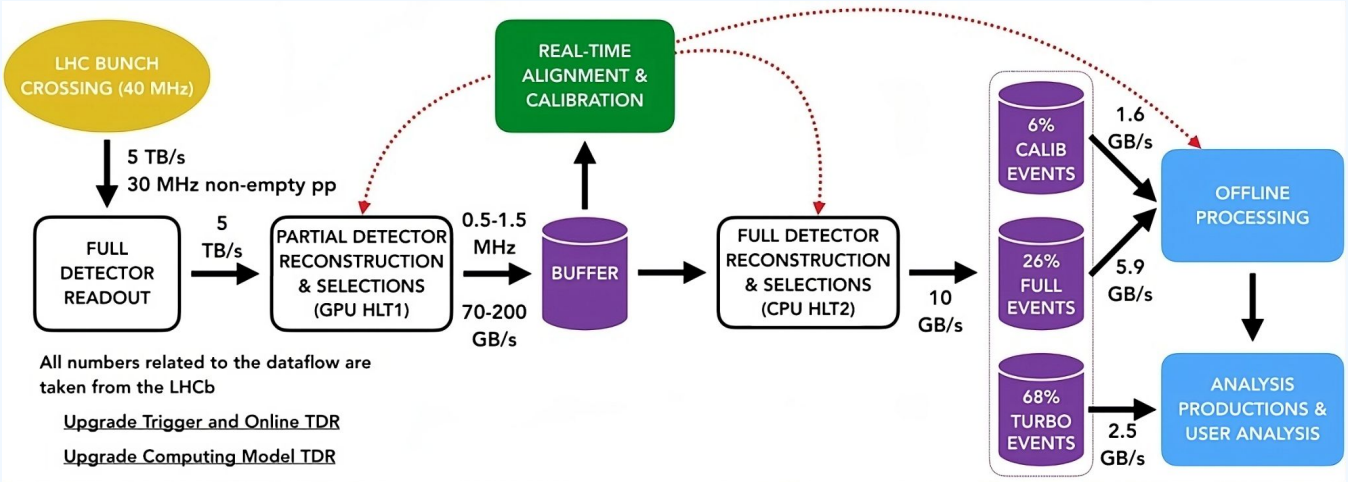
The network is built in a fat-tree topology over InfiniBand HDR (200 Gb/s) fabric.



Example of a Fat-Tree Network Topology

Reconstructed events have a size of ~100 kB, while individual fragments are O(100) B. It is necessary to pack fragments in packets to make efficient use of the network. *Multiple Fragment Packets* have a configurable packing factor (default 30'000).

Trigger Layout in Run 3



- Events are processed by the High Level Trigger 1 using GPUs hosted in the EB servers. HLT1 performs partial reconstruction, reducing the event rate by a **factor 30**.
- Data is **buffered onto disks** to allow collection of data for Real-Time Alignment and Calibration.
- Alignment and Calibration data is provided to the High Level Trigger 2 that performs full reconstruction using a farm of 250'000 CPU cores.
- A second buffer stores the HLT2 output. This way, the system can run independently from the main offline storage.

LHCb Upgrade II Requirements

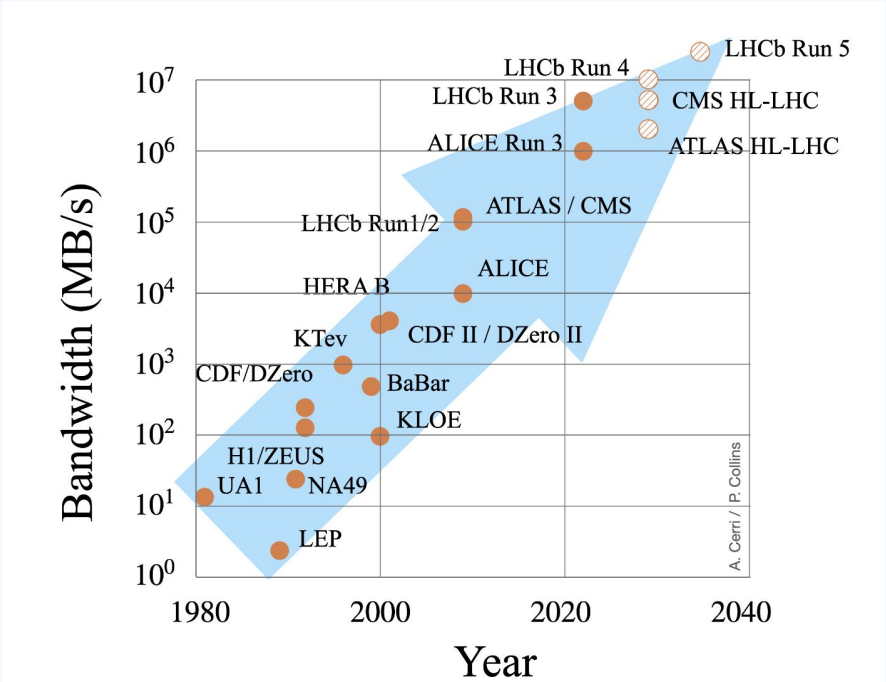
At 1.5×10^{34} $cm^{-2}s^{-1}$ peak luminosity, the LHCb detector will handle **~2000 charged particles** per bunch crossing. This will create significant challenges for **radiation hardness, tracking performance,** and **data acquisition**:

- The RICH will upgrade its system by introducing the FastRICH ASIC. This chip will introduce a **new complex data protocol** based on Aurora, which should improve link utilization and reliability. This protocol will then be decoded by the Back End.
- Numerous detectors (VeLo, RICH, ECAL) will introduce **precision timestamping** with resolutions of O(10) ps to manage the increased multiplicity.
- The ECAL will be redesigned to withstand **extreme radiation doses up to 1 MGy**. New ASICs are under development.

LHCb Upgrade II Requirements

The upgraded DAQ system will have to process 200 Tb/s of raw detector data.

- **GPUs** and **FPGAs** are essential for efficient data processing; **High Level Synthesis** (HLS) may simplify porting algorithms to FPGAs.
- New sub-detectors will use **faster lpGBT links**, doubling throughput but halving length. This necessitates splitting the EB farm between surface and underground.
- Enhanced timing requirements will drive significant upgrades to the **timing distribution** system.
- Transition from Infiniband to **Ethernet** and commercial off-the-shelf (COTS) hardware is under evaluation to improve cost-effectiveness.



Evolution of DAQ Throughput in HEP Experiments

Readout FPGA Architecture in LHCb

PCIe40 Hardware

A single readout board to implement **data acquisition, timing,** and **control**.

PCIe Gen 3.0 2x8 can achieve **100 Gb/s** full-duplex bandwidth.

The LHCb detector utilizes 10’818 links, with an average of 24 links per card.
Up to **48 optical links** organized in modules of 12. The links are connected to the FEE using the GBT protocol. GBT is capable of **4.8 Gb/s** towards the back-end.

Low-jitter components are used to achieve the current timing requirements.

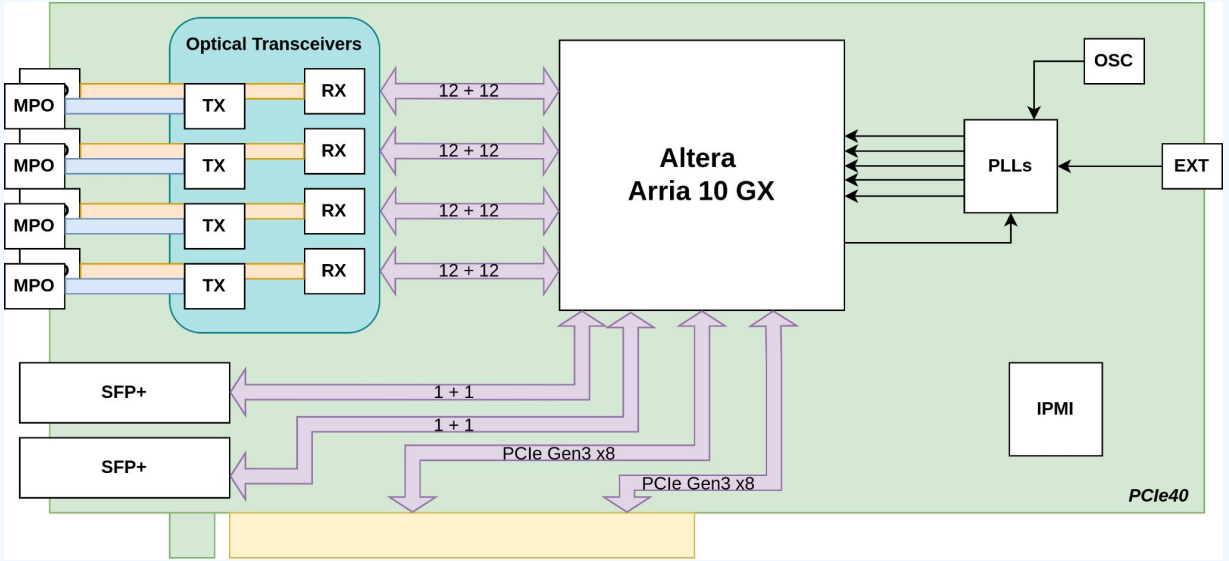
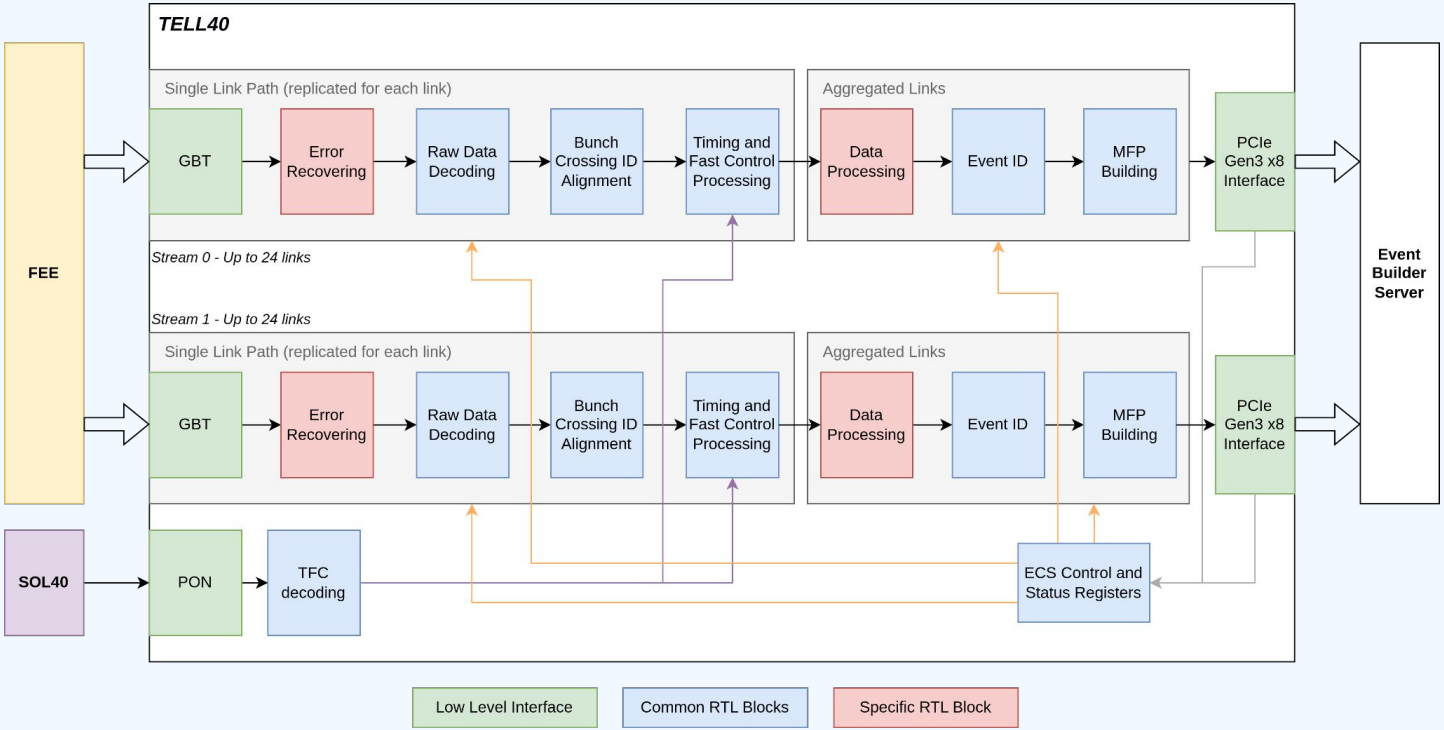


Diagram of PCIe40 Hardware Components

TELL40 Gateware



TELL40 offers a **generic gateware platform** with many common components (in green and blue). Subdetectors have to customize the **error recovering** and **data processing** blocks to their needs. This leads to **more than 30** different gateware configurations to be tested, maintained, and compiled.

Key Takeaways

The architecture has been successfully deployed and it is used now in production.

1. The **limited testing framework** relies only on full simulations with very limited stimuli. Unit tests are not implemented, resulting in **poor coverage**. This leads to labor intensive debugging, risking the **loss of beam time** during data taking.
2. All development relied on **proprietary tools** and components. This **hinders portability** between devices and vendors, while restricting simulation possibilities.
3. PCIe is still a widely adopted interface, but recent trends indicate a shift away from it in HPC due to **power, cooling,** and **space constraints.**
4. **Manpower is fairly limited**, counting fewer than 10 members in the core team, supporting all sub-detectors developers.

Future Hardware: PCIe400

Subdetector upgrades will focus on **timing** and **high-speed links**.

The new lpGBT protocol doubles the current bandwidth at **10 Gb/s**.

Timing constraints in the picoseconds level do not allow the reuse of the PCIe40.

A new board has been designed using the latest FPGA technology.

The new board will feature PCIe Gen 5 x16, 400 Gb/s Ethernet and **46 optical links** capable of **up to 26 Gb/s.**

Moreover, 32 GB of **high bandwidth memory** enable new possibilities in data processing and reconstruction.

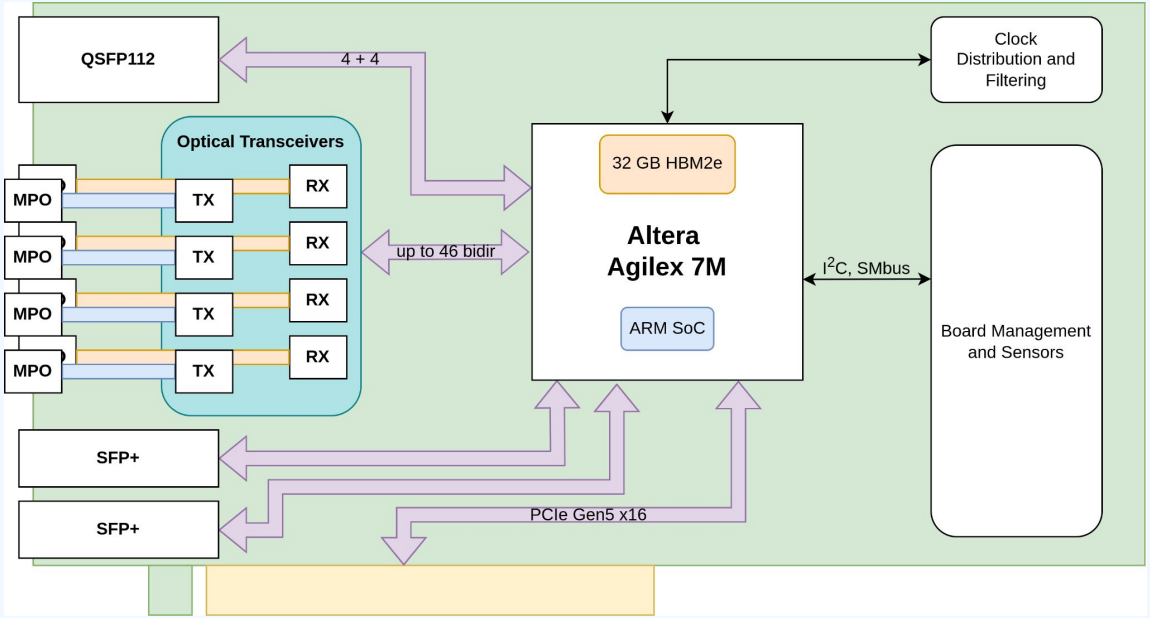


Diagram of PCIe400 Hardware Components

Future Gateware Design

This transition to a new FPGA is a laborious process due to technological differences between devices.

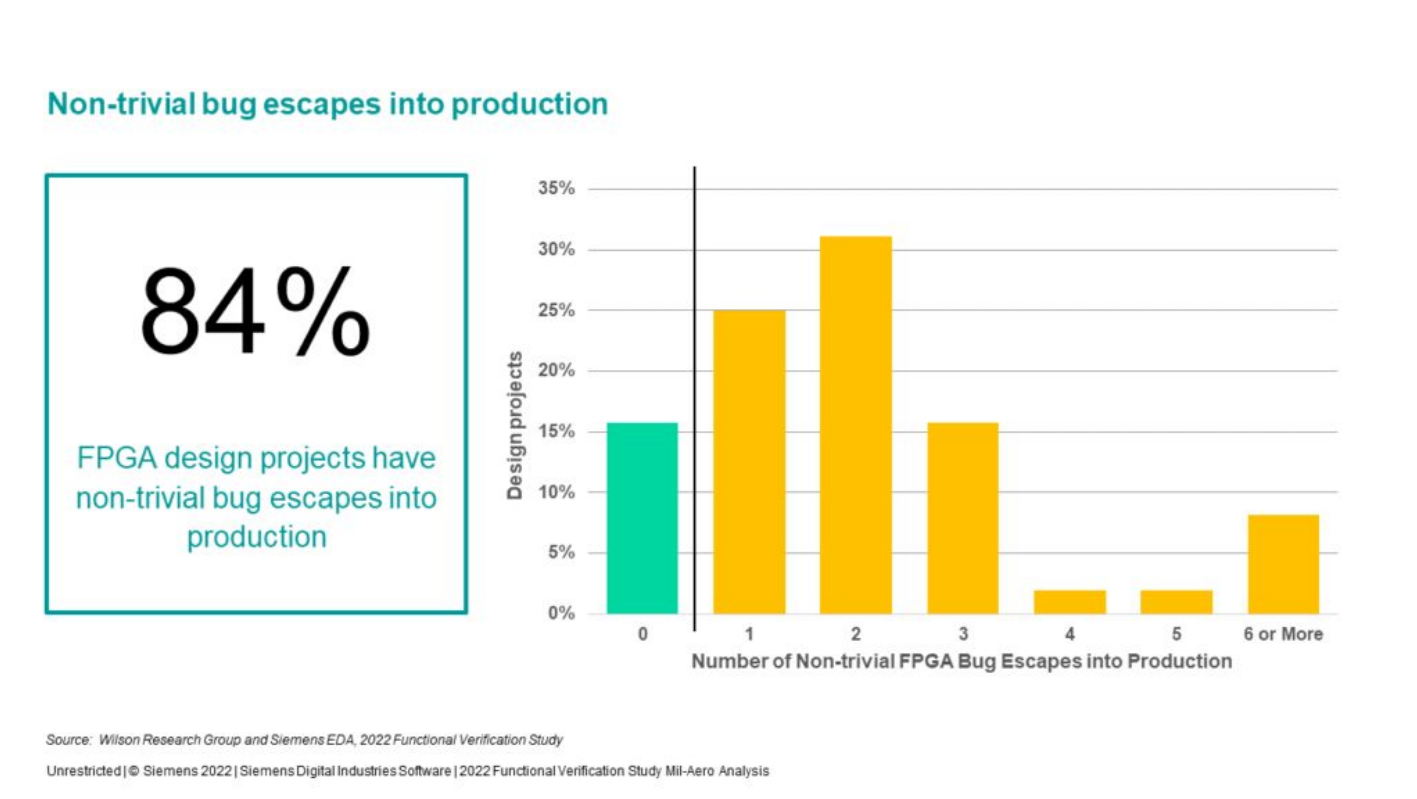
The codebase has to be reworked to implement the new features. Guidelines have been developed to help the development process:

- Developers should embrace the **test driven development** methodology. Tests should be developed first, based on the application requirements before implementation. This **reduces test bias** and **speeds up closure**.
- The code should **not rely on any proprietary core** or **tool** if not strictly necessary. Open source simulators can be used to test the gateware, while the use of a common core library can simplify porting and testing.

Methodologies for Gateware Development

Verification, IP Reuse, and High Level Synthesis

The Need for Verification



Bugs in the LHCb readout contribute to detector inefficiencies.

Verification by Simulation

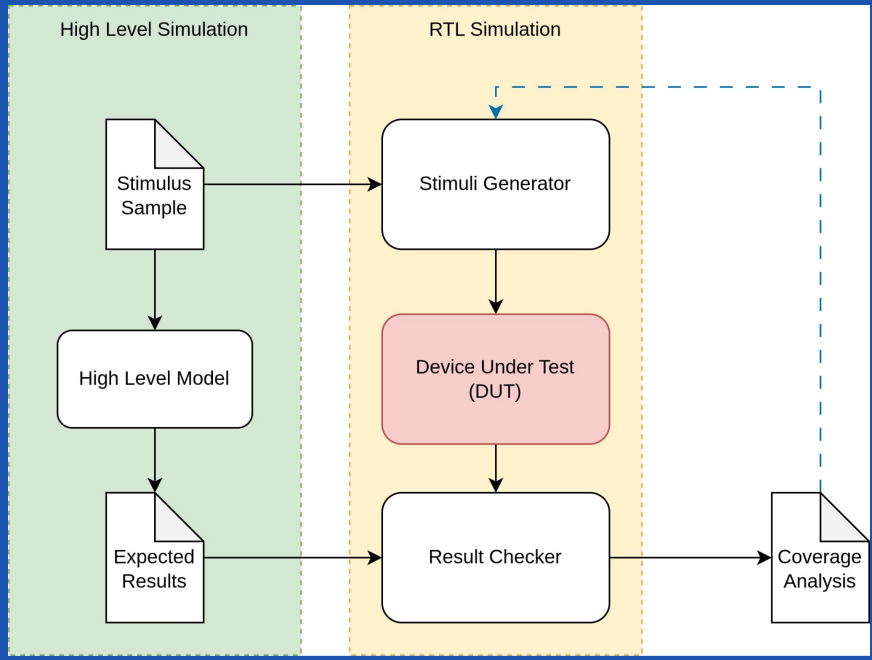
Simulation is the traditional method for verifying functionality.

The most effective abstraction is RTL simulation:

- Works directly on synthesizable components
- Cycle-accurate behaviour tracking
- Device independent

Functional Verification consists of a Device Under Test (DUT), a stimuli generator, and a result checker.

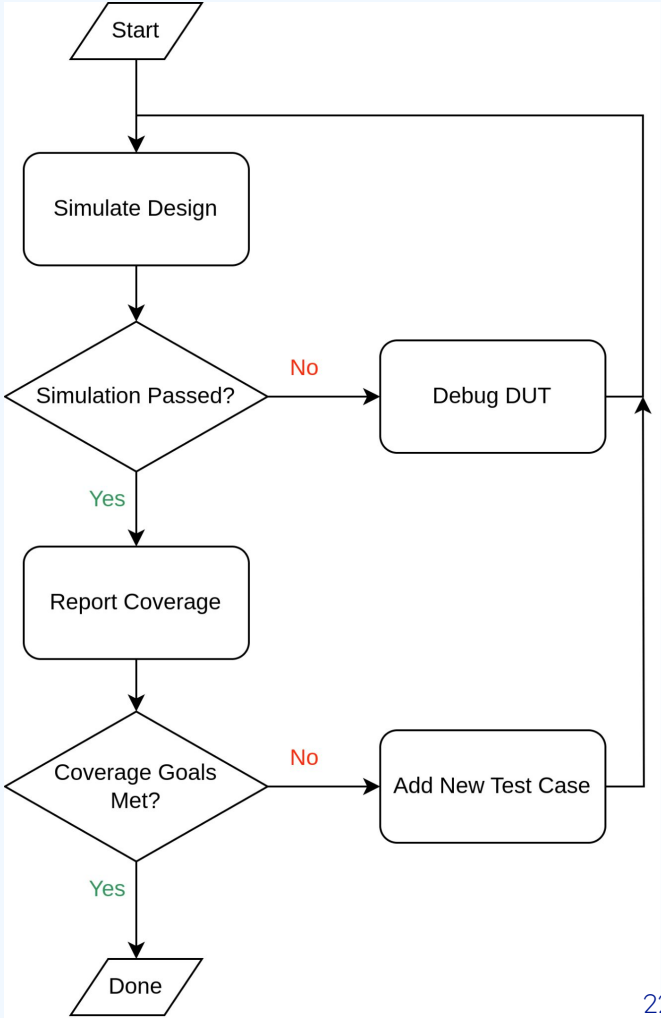
Functional coverage metrics **measure** whether specific scenarios, corner cases, design requirements, and other critical conditions of the DUT have been **observed, validated,** and **tested**.



A typical simulation setup

Constrained Random Testing

- **Manually creating test cases is impractical**. This method lacks portability, reusability, and may miss hidden bugs.
- Existing tests using **fixed datasets** (e.g. Monte Carlo data) **do not provide enough coverage** to fully verify DUT behavior.
- Pseudo-random stimuli generators **automatically produce test cases** from specifications, dramatically increasing coverage.
- Automatic checkers track which test cases have been exercised and report their pass/fail status.



Transaction Level Modeling

Components interactions typically involve **interfaces**, with signals divided into **data** and **control** paths.

Interface interactions are defined by **protocols**, consisting of specific **transactions**.

Transaction Level Modeling abstracts the implementation specific details focusing on the communication details.

Developers can focus on design functionality instead of low-level signals.

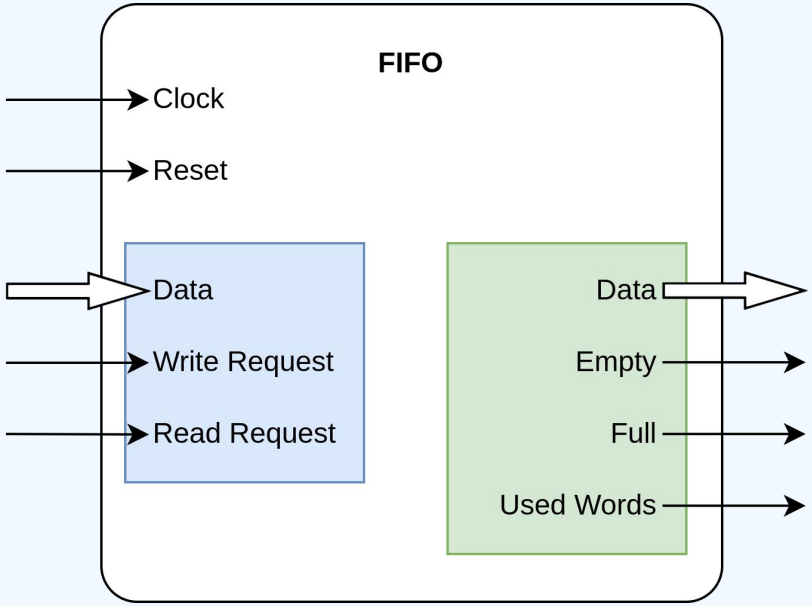


Diagram of FIFO ports and interfaces

Verification Frameworks

Verification features have been consolidated into reusable frameworks.

In VHDL-based designs, three open-source frameworks are available and have been evaluated:

- Pure VHDL-2008
- Checkers and Loggers
- **Advanced TLM interfaces (VVCs)**
- Randomization
- Scoreboards

- VHDL-2008 and TCL
- Checkers and Loggers
- **Randomized Coverage**
- **Scoreboards**
- Detailed reporting
- Test scripting (tcl)

- Python and VHDL-2008
- Checkers and Loggers
- **Simulation Management**
- TLM components
- **Python test scripting**

Formal Verification

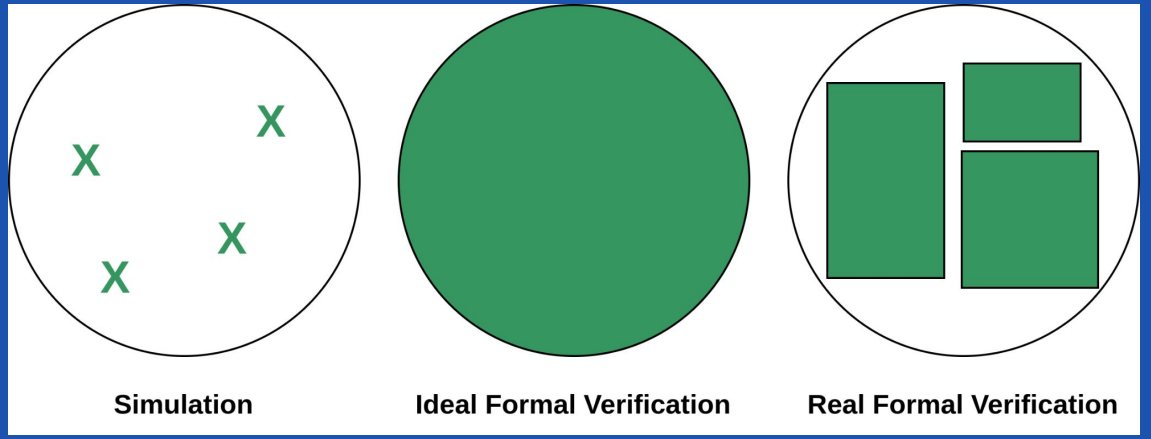
Simulations are **limited** and cannot exhaustively test every scenario.

Formal Verification (FV) translates tests into propositional logic formulas and checks their satisfiability using **mathematical methods**.

FV offers strong guarantees by **proving** that a design **meets its specifications** and is **free from bugs**.

However, FV is NP-complete, so **computational demands increase exponentially** with design complexity.

As a result, **complete proofs** for all possible cases **are often impractical** in real-world designs.



Simulation only offers spot coverage, while FV can cover wide areas of the state space.

Applied Formal Verification

FV tests are written using the **Property Specification Language** (PSL).

PSL has two kinds of properties:

- **Assumptions** define the input constraints to the DUT
- **Assertions** define the allowed behaviour of the DUT

Using language features, a FV testbench can be implemented in **just a few lines** (e.g. a FIFO simulation TB requires ~200 lines, while the FV is just 20 lines).

However, due to the NP-hard nature of the problem, **verification closure could be unreachable**. To optimize this:

- Focus on small designs
- Avoid data path modelling
- Avoid multiple clock domains

Formal Verification should complement conventional testbenches.

```
assume reset;
assert always reset |=> empty;
assert always (not reset and
               write_req and
               not read_req and
               not full) |=>
(write_ptr = prev(write_ptr)+1);
```

Extract from a FIFO Formal Verification Testbench

Common Core Library

The use a common core library allows developers to reuse IPs:

- Improves verification by using already verified code
- Eliminates code duplication and speeds up development by offering ready-made components
- Enhances portability by being vendor independent

I designed the library *colibri* to tackle these points by offering **vendor-independent, fully verified, open source** components for FPGA designs.

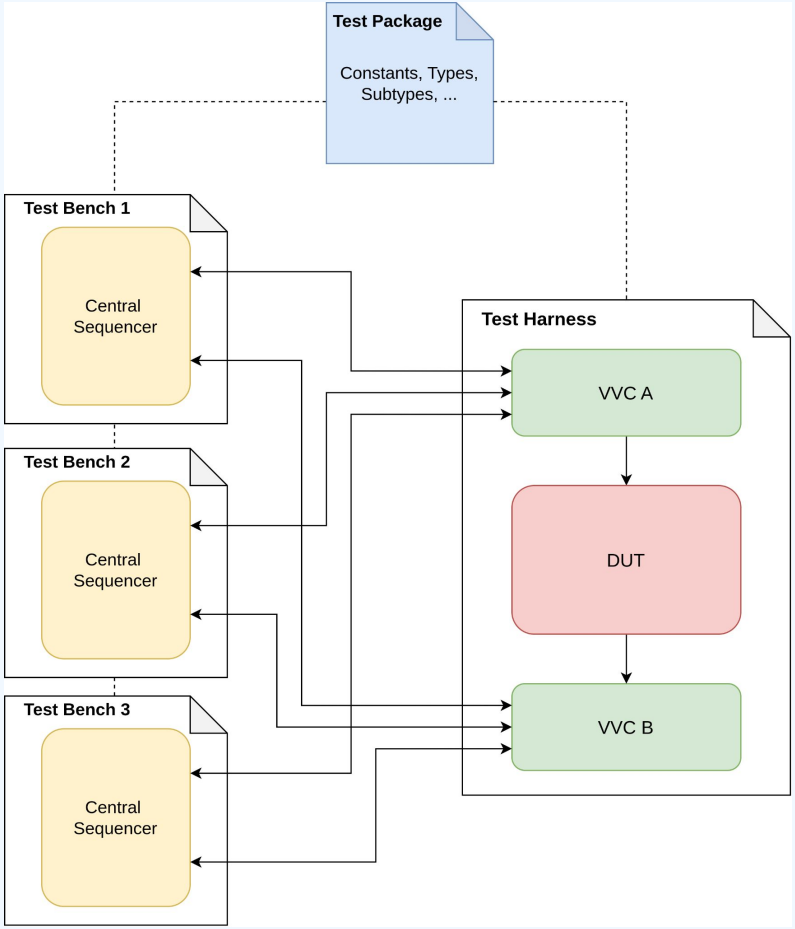
Verification

Every component in the library has an associated simulation testbench which implements constrained random testing.

For complex designs, colibri implements the UVVM modular architecture:

- A **test package** which contains common constants, functions, procedures used across the testbench.
- A **test harness** entity where components such as DUT and TLM interfaces are instantiated and connected.
- **One or multiple testbench files** which control the test using a single sequencer process, which dispatches commands via the UVVM network.

Components suitable for FV have dedicated testbenches.



Verification architecture for complex designs

Continuous Integration

Continuous Integration pipelines are run on the GitLab runners provided by CERN IT, available to all developers at CERN.

Each automated pipeline gives an immediate feedback to developers, simplifying the code review process.

Pipeline status also improve the community's trust in the project, demonstrating its reliability and quality.

Style Checking

Each contribution must adhere to a standardized coding style.

This is enforced in CI by using the VHDL Style Guide tool.

This ensures an homogenous codebase, facilitating onboarding of new collaborators and integration into user projects.

Simulation

Automated testbenches are run using VUnit and the NVC open source simulator.

Simulation output is formatted in a Gitlab-compatible XML format. This integrates the feedback directly in the pipeline, enhancing error tracking and coverage analysis.

Formal Verification

Components suitable for FV have dedicated testbenches which are run using the open source toolchain based on SymbiYosys.

Pass/Fail feedback is integrated in the pipeline reports.

High Level Synthesis

HLS allows developers to translate **C and C++ code directly into hardware**.

The advantages are:

- Higher abstraction of the design specification
- **Lower language barrier** compared to HDL
- **Faster development cycle** thanks to emulation

This requires significantly **smarter tools** to convert the code into efficient hardware implementations.

Intel OneAPI FPGA Toolkit offers a new approach to this methodology using SYCL C++ .

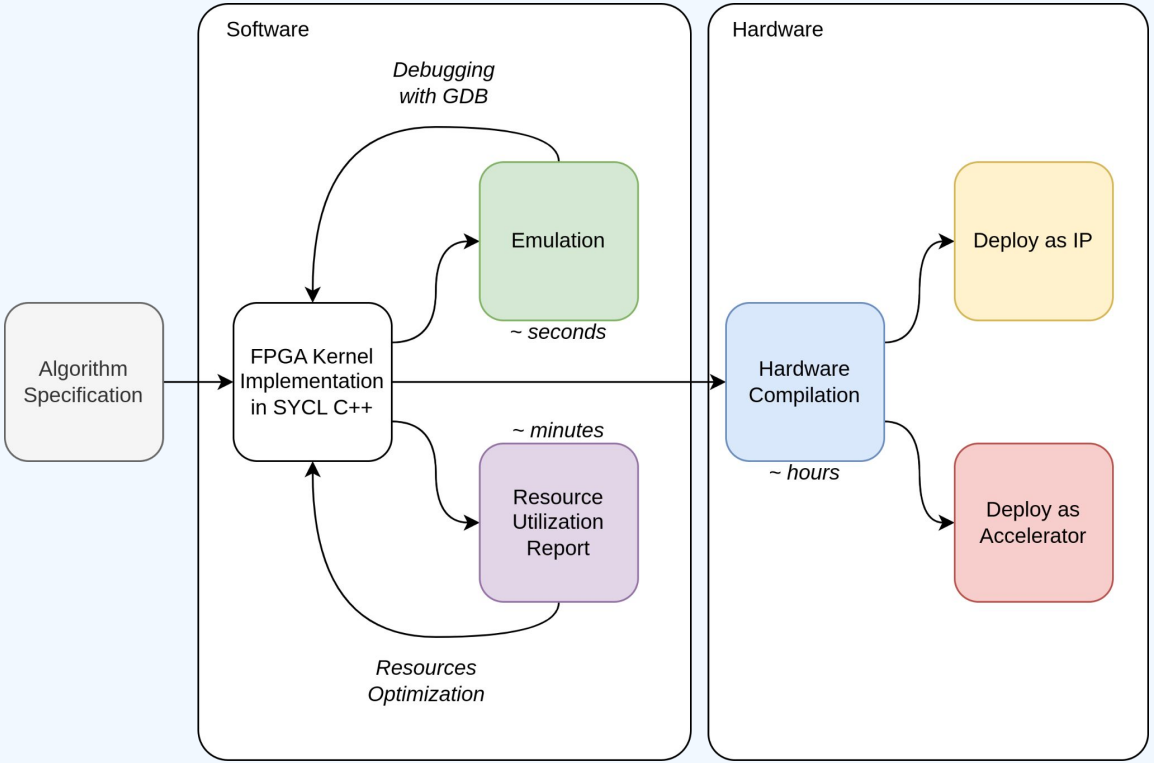
Intel OneAPI FPGA Toolkit

OneAPI offers a full framework based on the SYCL C++ abstraction layer:

- Similar to GPU programming in CUDA
- Single source for CPU, GPU, and FPGA implementation
- Emulation on CPU reduces the debugging time
- Resources utilization reports to reduce footprint

This happens all before hardware compilation.

The user can choose if the FPGA is used as accelerator via PCIe or the code is a standalone IP.



Development workflow using Intel OneAPI FPGA Toolkit

Case Studies

HLS for HLT Acceleration

The oneAPI framework enables the use of FPGAs as **compute accelerators** over PCIe.

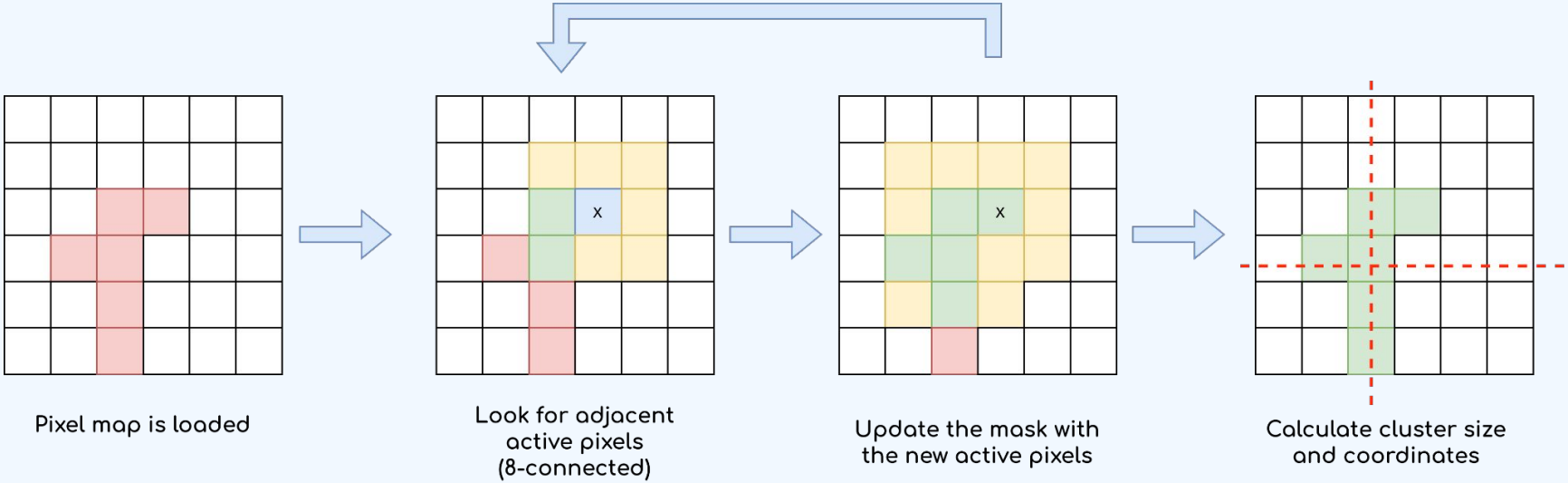
HLT1 algorithms are **embarrassingly parallel problems**, which maps well to the GPU architecture used in Run 3.

However, many algorithms involve **bitwise operations** which are well suited for FPGAs.

This work evaluates the capabilities and the maturity of the Intel oneAPI framework for HLT acceleration.

VeLo Clustering Algorithm

Active pixels coming from the frontend are grouped in 208 banks corresponding to different independent parts of the detector.
The clustering algorithm starts from an **active pixel** (candidate) and **looks for adjacent active pixels**, updating cluster size and weighted averaged coordinates when it finds one.
When there are no remaining adjacent active pixels, the definitive cluster information is returned.



The algorithm presented is highly parallelizable thanks to the **physical independence** of **sensors** and **events**.
The current GPU implementation utilized a block and thread approach parallelizing on candidates, sensor banks, and events.

FPGA Implementation

The code was ported in less than a month, since it was already in SYCL and required minor modifications. However, it was **performing poorly**.

Performance optimization required a **pipelined distributed design**:

- Dedicated kernels read and write to the host memory contiguously
- Jobs are distributed to worker kernels via dedicated dispatching kernels
- A synchronization kernel synchronizes the whole chain to avoid data corruption

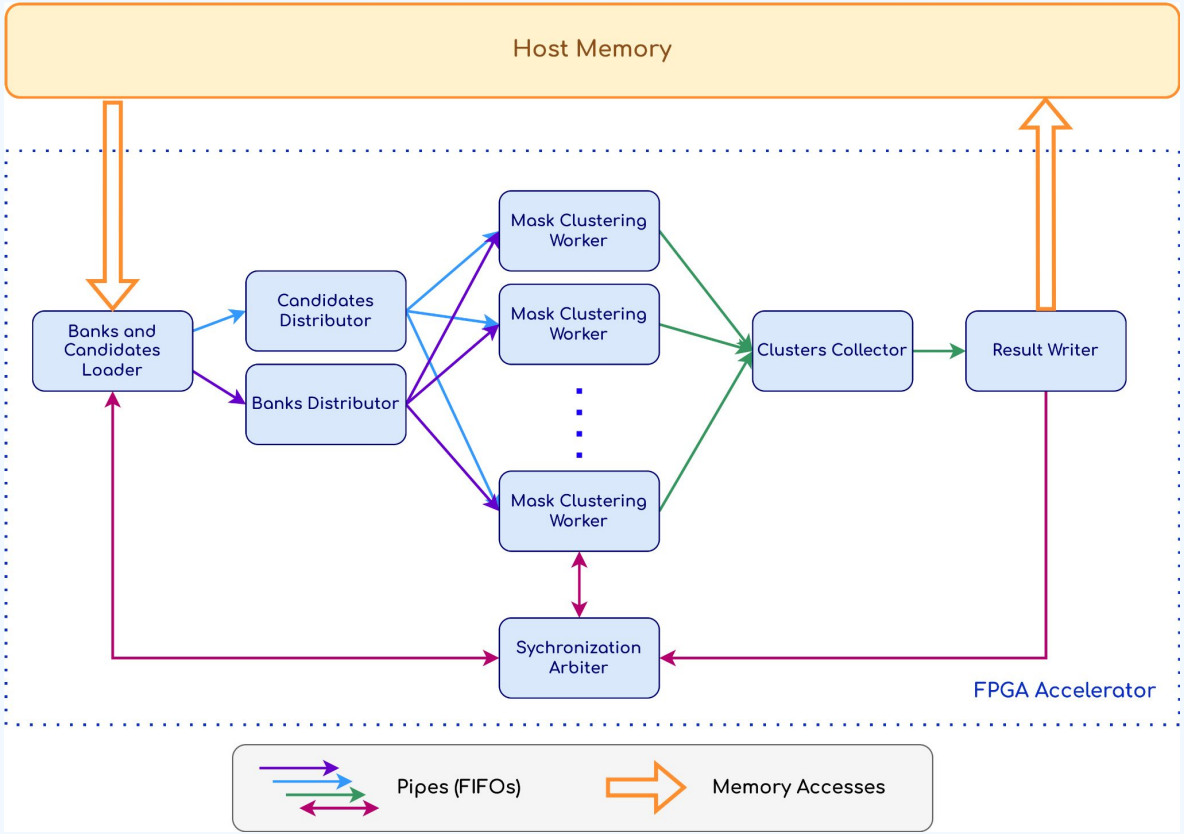


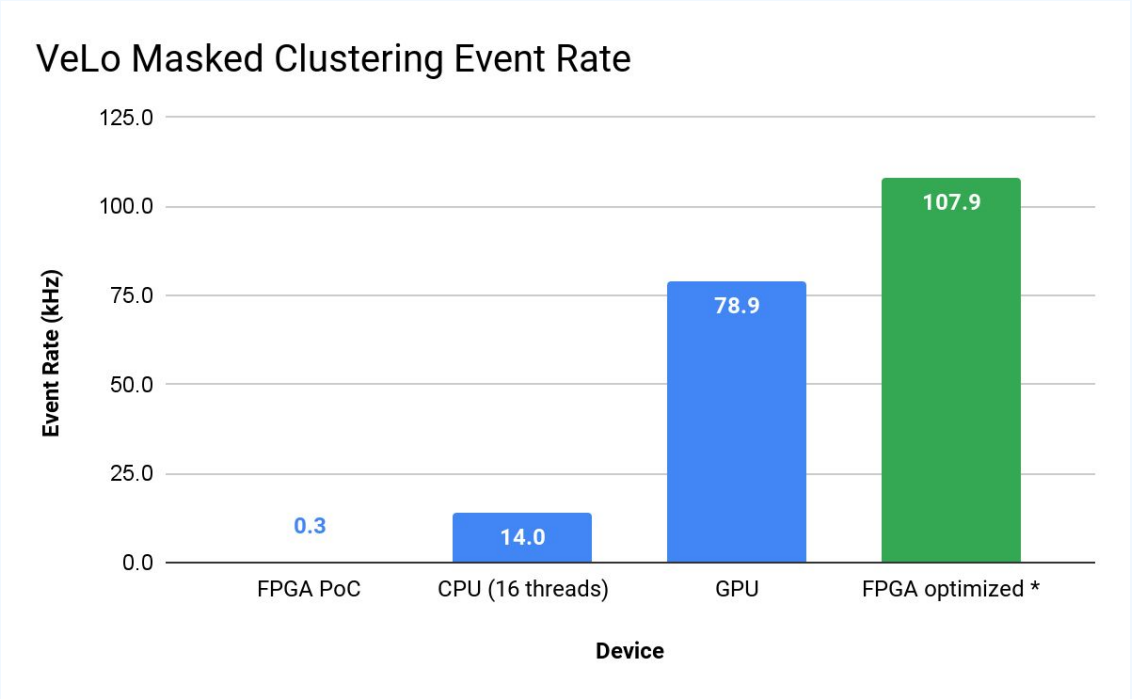
Diagram of the optimized pipelined design

Performance Results

Host memory accesses required **specific design choices** both in the kernel and in the host software to reach the advertised PCIe throughput (single pointer access, contiguous data).

Since this algorithm is **compute intensive**, the multiple workers approach was the **most performant**.

However, a unresolved bug in the oneAPI compiler causes the optimized version to crash the host when writing. Tests were run with a dummy writer kernel to avoid the issue.



Benchmarks were run on a Bittware IA-840F FPGA Accelerator Card, an NVidia RTX A5000 GPU, and an AMD Epyc 7502 CPU.

Bug Finding with Formal Verification

FV was used to identify **sporadic data corruption** in the RICH TELL40 gateware.

The compressor block was the first target of the tests.

This block is responsible for processing 4 85-bit data streams, representing raw data coming from PMTs, and generate packets on a 256-bit data stream.

The block is treated as a black box and the testbench only specifies the behavior of its input and output interfaces in PSL.

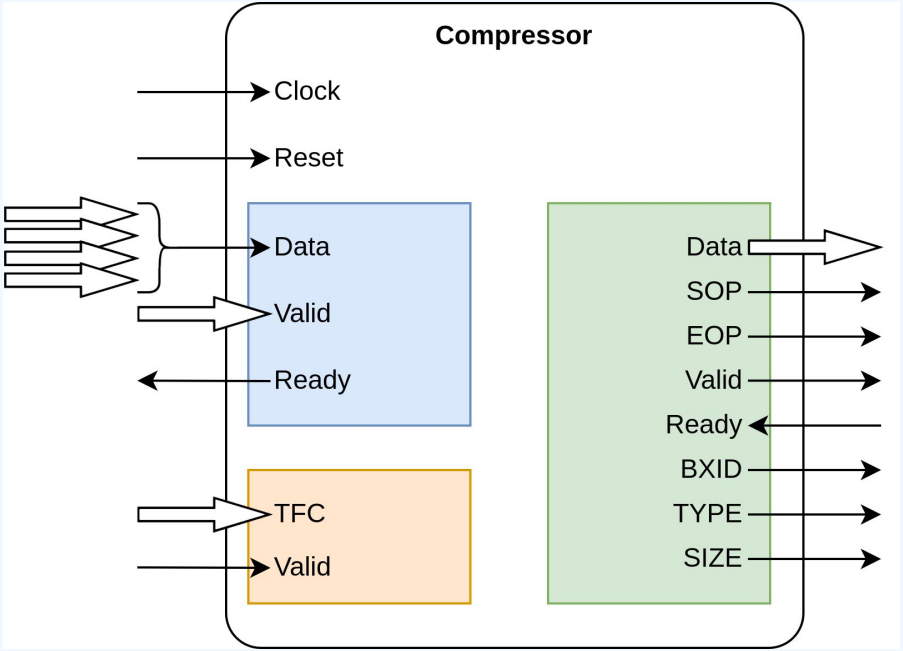


Diagram of the Compressor Ports

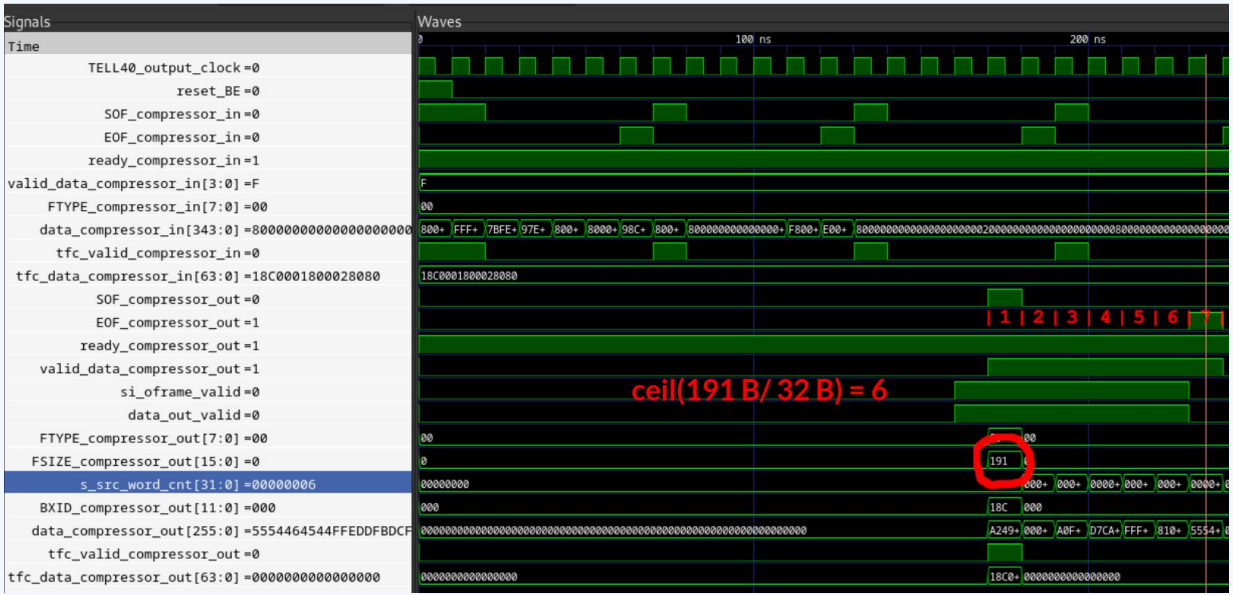
Bug Finding with Formal Verification

Modeling specifications in PSL requires **time**, **expertise**, and **practice**.

A key challenge in FV is ensuring the test harness fully covers the intended specification.

For the RICH system, test design **took a few days**. The FV tool quickly identified a counterexample that reproduced the observed issue. This bug **appears only with stable beams**, losing hours of beam time for traditional debugging.

Without finding this bug, the entire experiment would have been **unable to collect data**.



The bug consists in a mismatch between the size value and the number of actual data words sent.

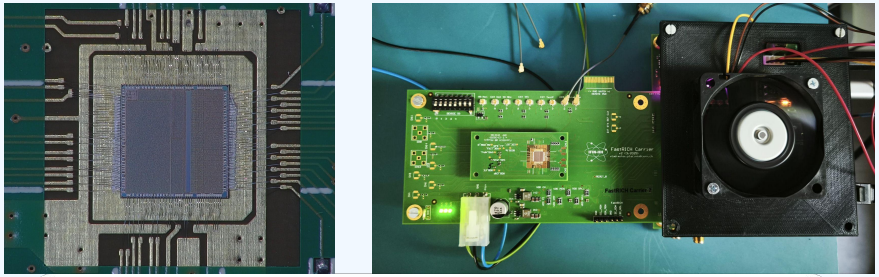
FastRICH ASIC Decoding

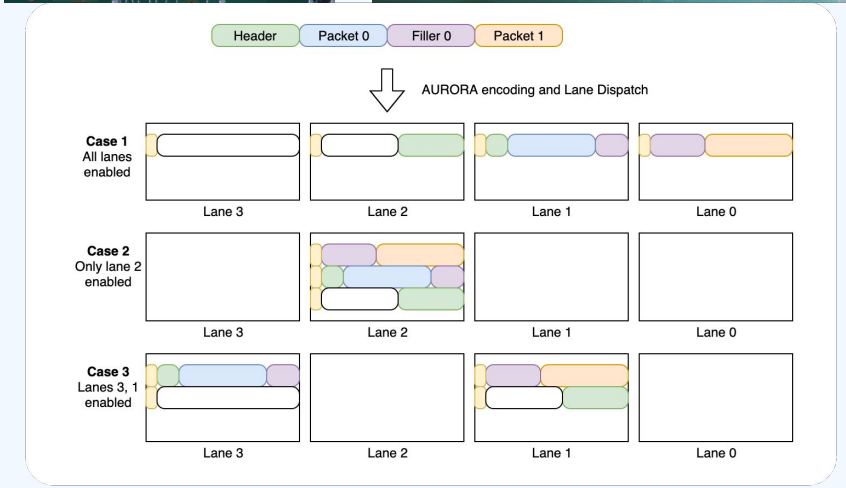
FastRICH employs a protocol based on Aurora, a link-layer protocol that provides **encapsulation**, **DC balancing,** and **multi-lane channels**.

The multi-lane feature is exploited to **optimize the link usage** based on the **occupancy** of different sub-detector regions.

The ASIC can be configured to output 1 to 4 lanes, with three possible lane widths: 8, 16, 32 bits.

Serialized data is then sent to the Back-End over lpGBT links.





The FastRICH ASIC and its testbench (top) and Aurora Lane Dispatching (bottom). Images courtesy of FastRICH

Decoding Aurora

Data is received by the Back-End and the lpGBT encapsulation layer is removed.

Slices of the lpGBT words (**eLinks**) are used as Aurora lanes with their configured width and number.

The Back-End has to do the following steps to **strip away the Aurora encapsulation**:

1. Convert the lane width to Aurora word
2. Align to the word boundary
3. Remove DC balancing
4. Align the lanes
5. Decode the Aurora frames

This Aurora decoder has to be replicated **for each ASIC**, while some of its components are replicated **for each lane**.

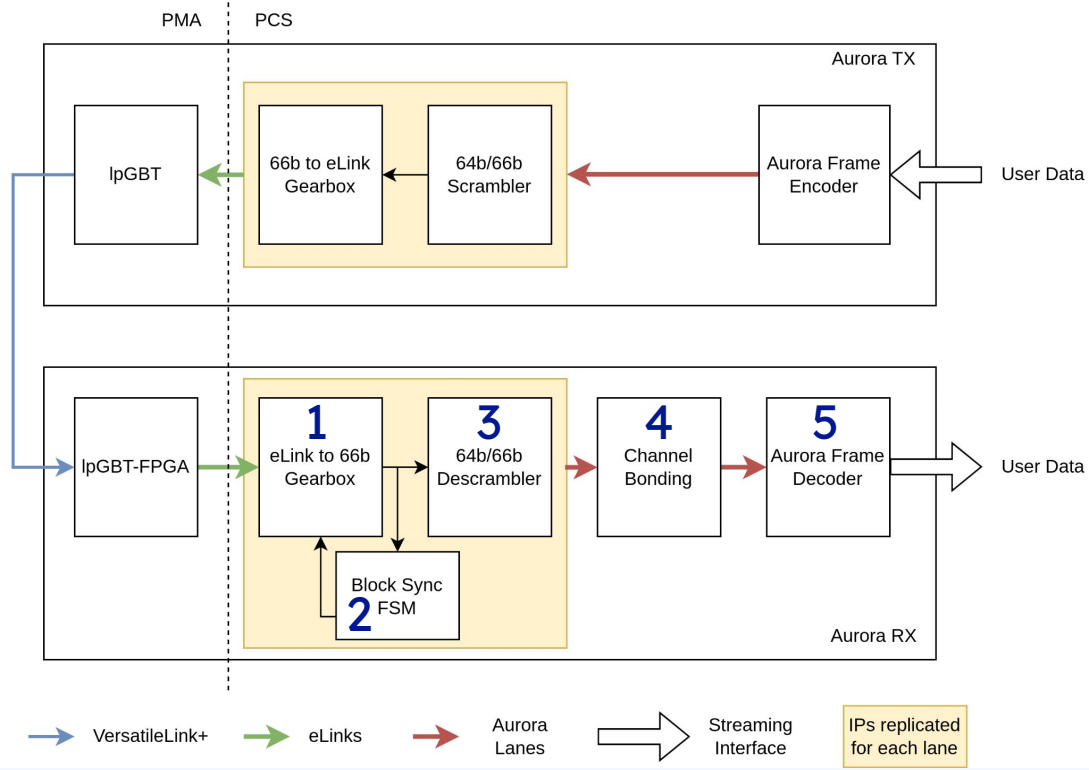


Diagram of the Aurora Physical Coding Sublayer

Design and Verification

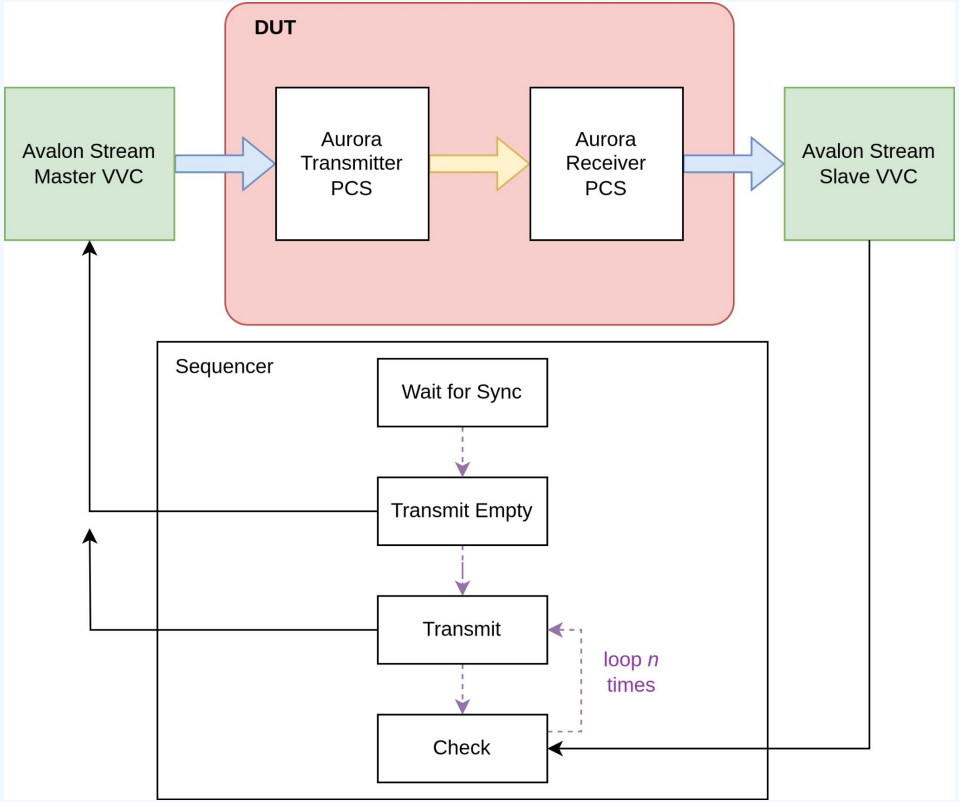
Both Aurora Transmitter and Receiver components have been designed from scratch using only open source components from colibri.

Each component has **dedicated verification testbenches:**

- Small components are validated using **constrained random** stimuli generators and automatic checkers
- The full flow is verified using both constrained random testing together with **transaction level modeling**, which automatically checks for protocol consistency.

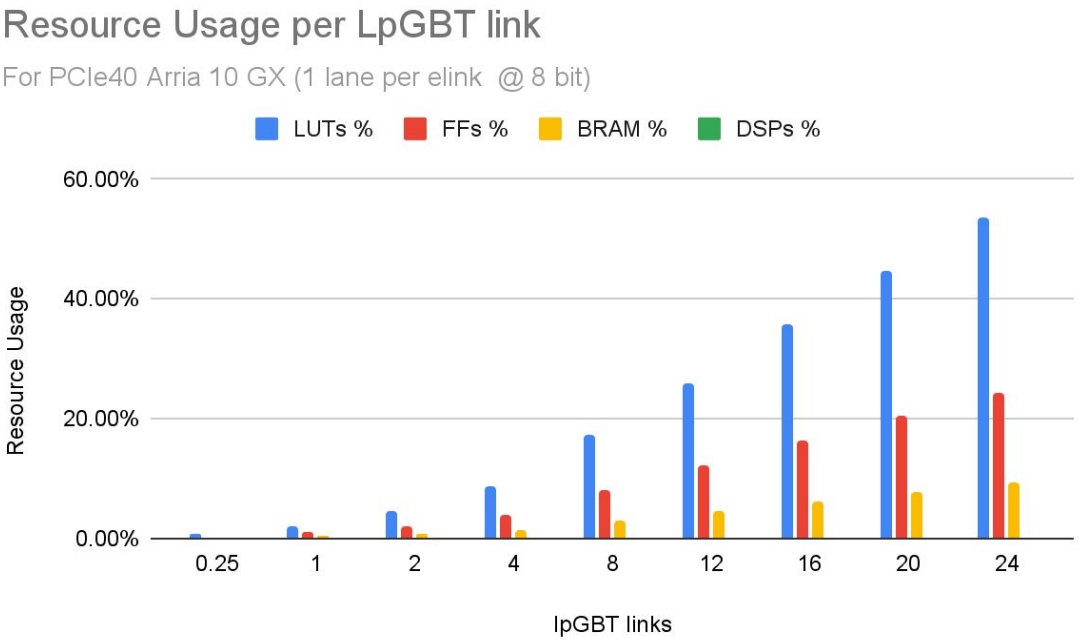
Tests are run in CI using NVC open-source simulator and VUnit framework.

Generics allow the **dynamic creation of tests for all possible configurations** of lanes' number and width.



Testbench layout of Aurora TX and RX in loopback.

Aurora Implementation Results



Aurora could be decoded in the transceiver hw, but this is **already used for lpGBT**. The decoding has to be done in the FPGA, introducing a **significant overhead**.

The original decoder implementation was resource-intensive, **limiting the number of available links** on the PCIe40 board.

By developing resource-optimized components, the gearbox footprint was **reduced by 50%**, allowing more links to be implemented on each board.

The code was successfully ported to Xilinx and Altera devices and it is now being used to test the ASIC.

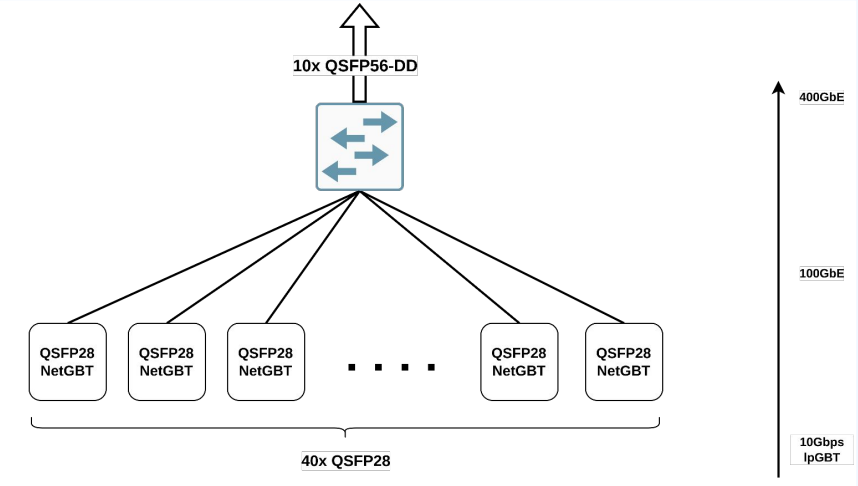
lpGBT to Ethernet Media Converter

Ethernet is a cost-effective and largely diffused technology. This proof of concept evaluates the use of low-cost FPGAs for DAQ systems based on Ethernet instead of PCIe.

The media converter is capable of interfacing with a number of **lpGBT links** and, with little processing, convert them into **UDP/IP streams**.

The Ethernet standard makes the design **highly flexible**:

- Supports **direct connection** to a network interface for testbench and small test beam setups
- **Scalable for large-scale deployments**, where multiple converters can be aggregated using consumer off-the-shelf (COTS) Ethernet switches.



A first draft of a DAQ system using 100GbE-capable NetGBT.

Proof of Concept: Hardware

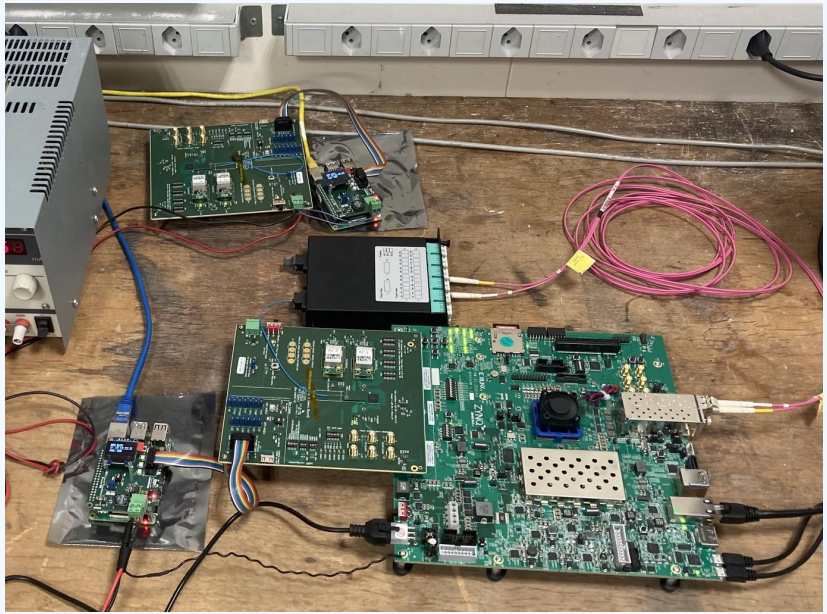
The hardware chosen for the Proof of Concept is an **AMD Artix UltraScale+** AU25P.

This choice was made due to the availability of an **off-the-shelf development kit** which makes all transceivers available.

This board can implement:

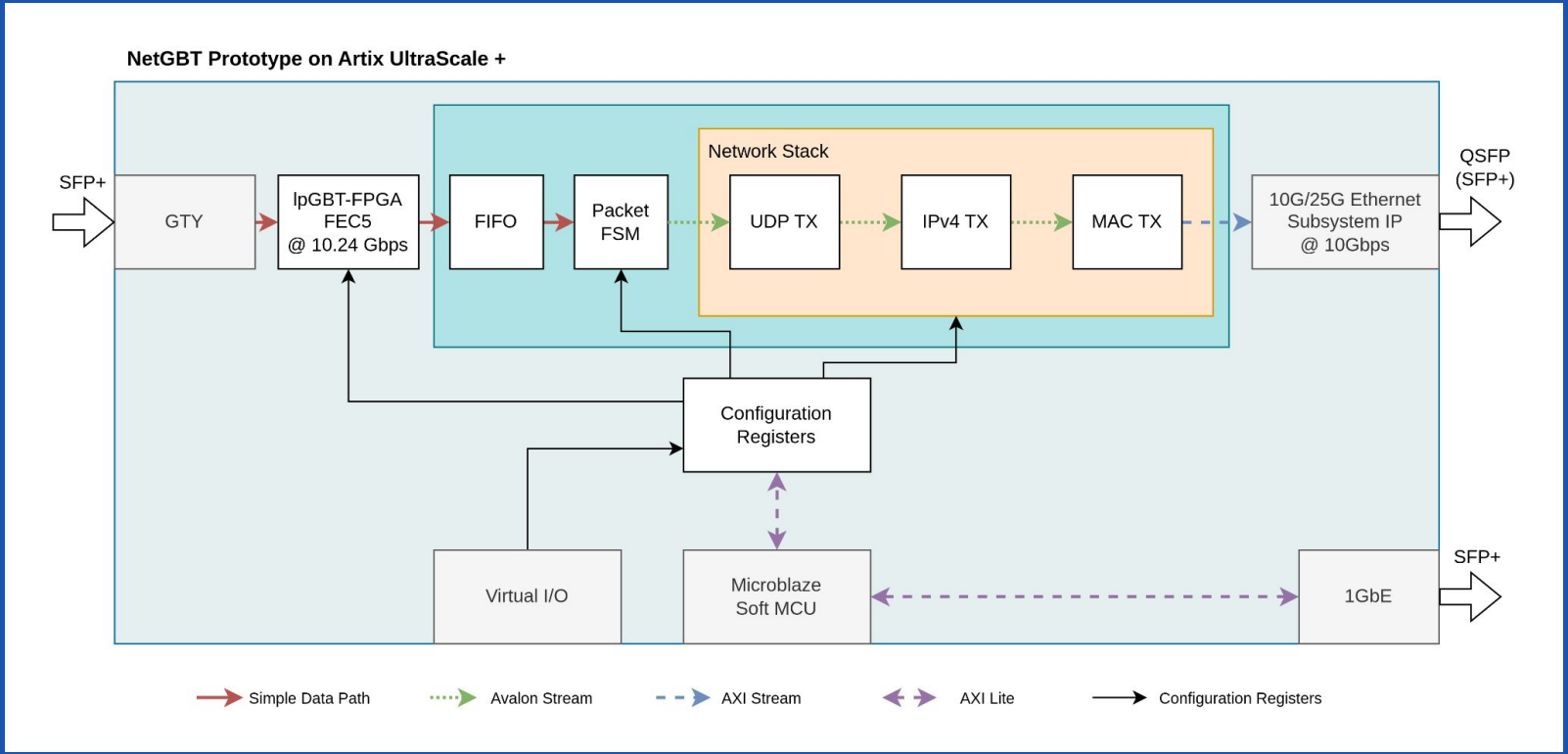
- Up to **4 lpGBT links** @ 10.24 Gbps (8.96 Gbps real)
- Up to **4 Ethernet links** at 10 Gbps
- A dedicated 1 Gbps Ethernet link for configuration

The board however does not offer 25 Gbps transceivers (GTY), which could be used to implement a 100GbE link.



Proof of Concept Setup

Proof of Concept: Gateware



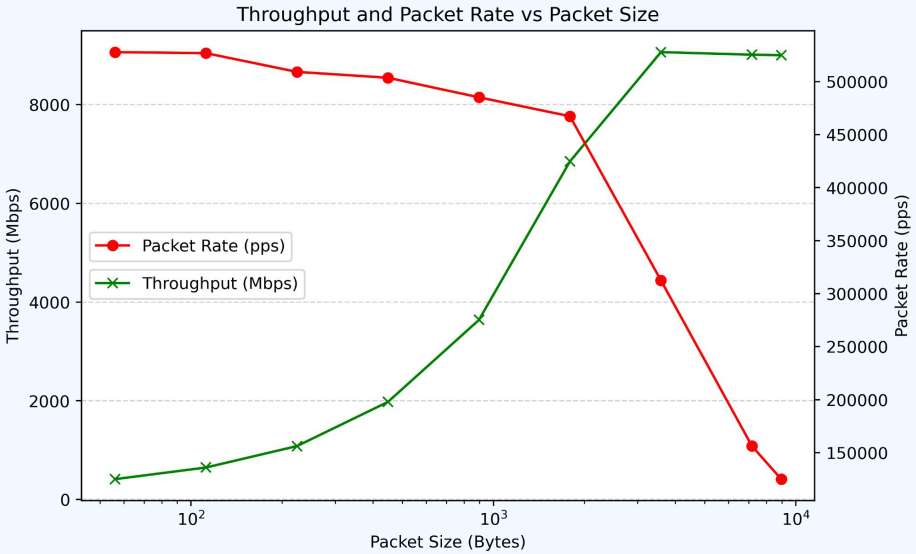
This board (Artix US+) can handle up to 4 lpGBT links, which correspond to a full VTRx+, and a full 4 lane QSFP+ (4x10GbE) on the Ethernet side.

Testbench: Results

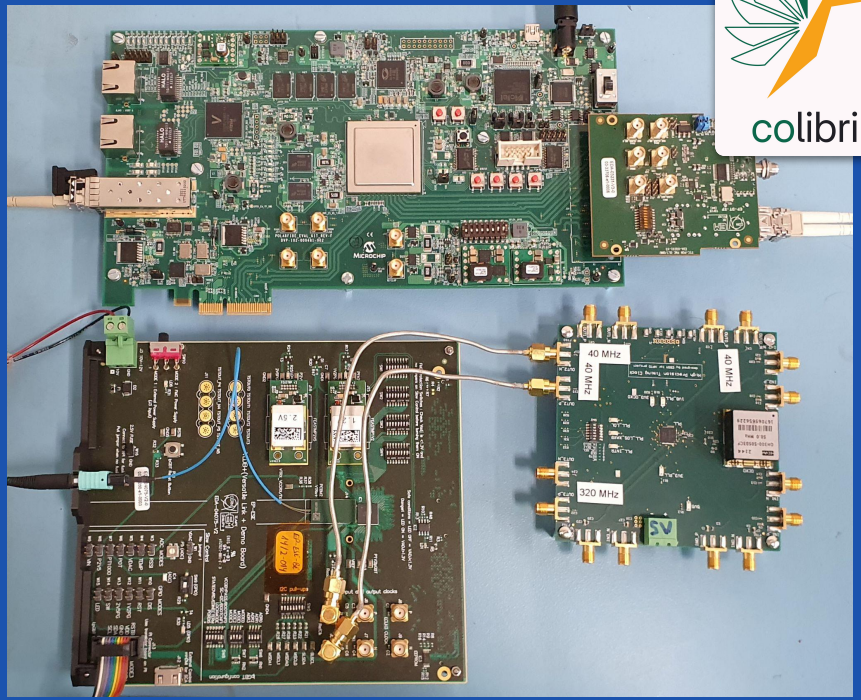
Measurements were taken to evaluate which **packet size** is optimal to **reach peak throughput** and to avoid back pressure.

The NetGBT PoC was connected directly to a **10Gbps** Network Interface via a Direct Attach Cable.

Benchmarks show that the point-to-point connection is saturated when **packet sizes are more than 4 kB**.



Vendor Independence



Credits to Valentin Stumpert of EP-ESE

The gateware has been designed to be **vendor independent** using the ***colibri*** library, enabling testing of different architectures to find which one fits best.

The vendor independence of the design has been proven by the colleagues of CERN EP-ESE, who ported the NetGBT gateware to a Microchip FPGA Development Kit.

colibri supports:

- AMD Vivado
- Intel Quartus
- Lattice Radiant
- Microsemi Libero
- Efinix Efinity
- Gowin EDA
- Aldec Riviera Pro
- Mentor Modelsim
- NVC
- GHDL

Conclusions and Future Directions

Achievements

High Level Synthesis

oneAPI proved to be a modern development toolkit, but it is **not mature enough** for production deployments. Porting algorithms is easy, but achieving **performance requires significant effort**. Bugs are still present in the compilers.

Functional Verification

The use of modern verification frameworks **enhances** the testbench **quality**, **portability**, and extended the test coverage. Methodologies such as Constrained Random Testing and Transaction Level Modeling should be **required for new gateware developments**.

Formal Verification

Even if limited, FV proved to be highly effective in locating corner case issues in **fewer time than conventional tools**. Designers need to practice the PSL language, but the **investment is worth the effort**.

colibri

The use of standardized verified components **greatly reduced development time** for projects such as the FastRICH integration and the NetGBT prototype. Designs were **easily adapted to different FPGAs** thanks to its vendor-independence. colibri is now being used by LHCb, ALICE, and CMS.

Future Directions

LHCb Upgrade II developments will adopt the tools and techniques described as standard.

Functional verification is being implemented in the developments for the PCIe400, while part of the work will update the current tests on the PCIe40.

The colibri library will be integrated in all future projects, streamlining the design, simulation, and verification process.

Formal verification will be extensively used to complete the verification efforts.

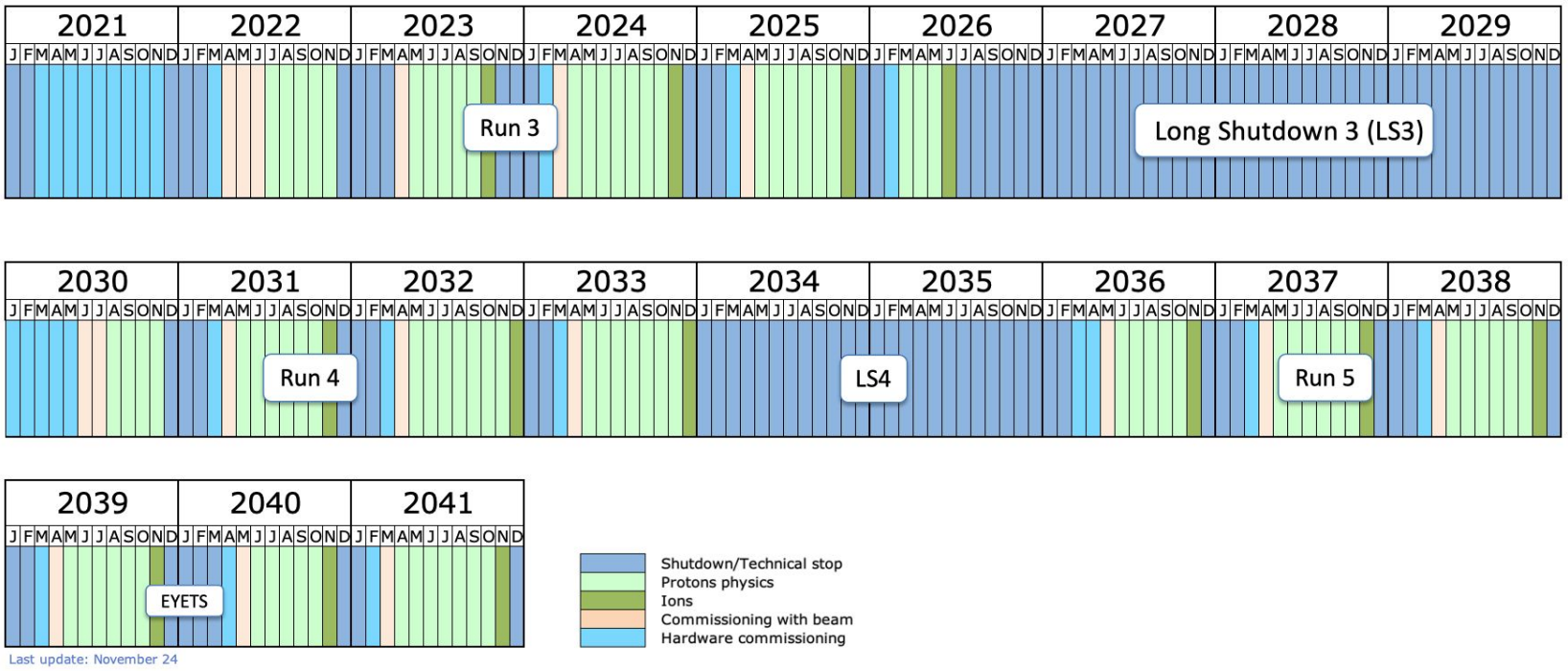
A second prototype of NetGBT will be designed to be able to aggregate more links over higher speed links. Part of the work will cover the design of resource-efficient FE data formats.

HLS will continue to be evaluated to assess its suitability for DAQ development.

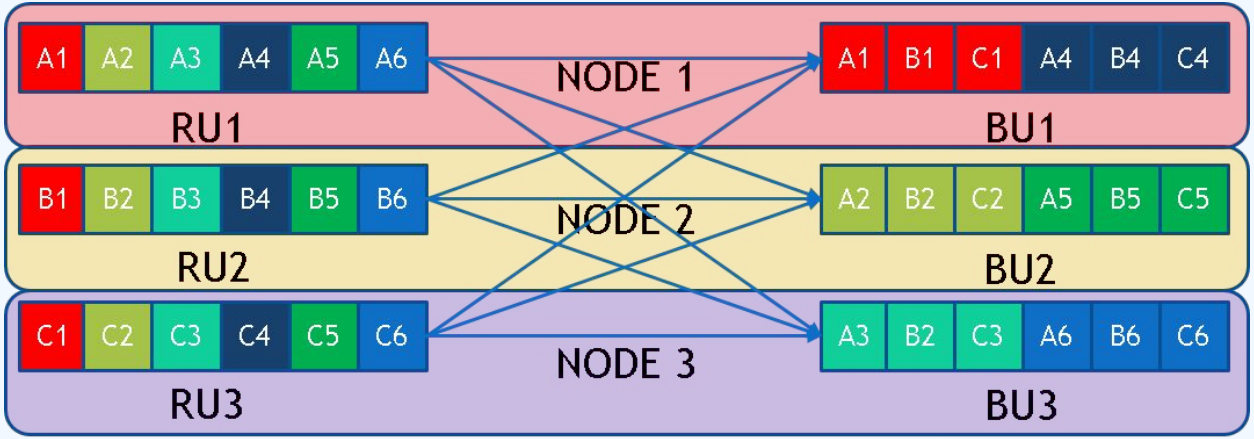
Thank you for your attention

BACKUP

LHC Schedule



The Event Building Process



The Event Builder (EB) collects data fragments from all the sub-detectors and builds full events.

Every Readout Unit (RU) read fragments from DAQ board and send them to the Builder Units.

Every Builder Unit (BU) receive fragments and build full events.

Timing and Fast Control Distribution

The TFC is responsible for:

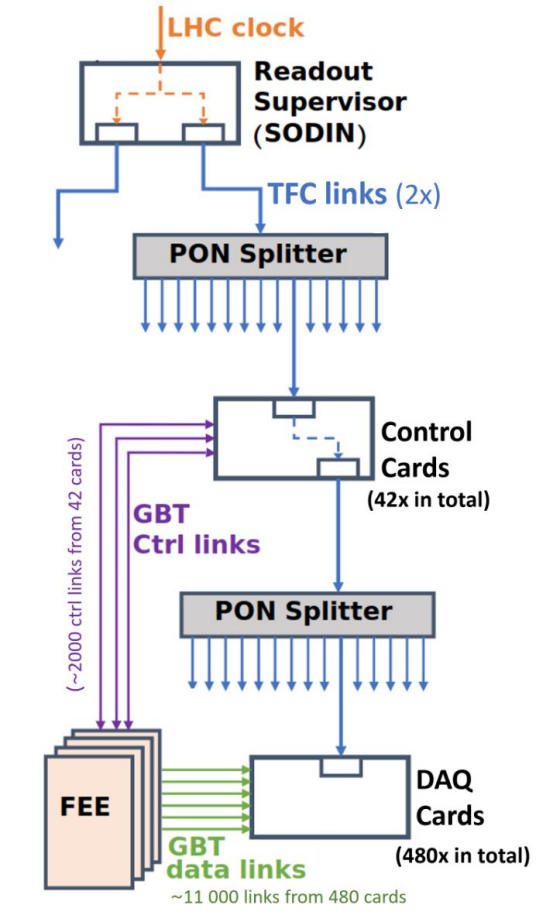
- Distributing LHC bunch clock with fixed phase
- Centrally generating real-time commands and distributing them with fixed latency
- Monitoring all the readout modules in real-time

The TFC Passive Optical Network uses the PCIe40 bidirectional 10G SFP+ modules.

The supervisor SODIN is connected to the control cards SOL40 via splitters.

Clock is recovered and cleaned with fixed phase in the SOL40 PLLs.

FEE are directly connected to the SOL40s via the available GBT links.



10.1109/TNS.2023.3273086

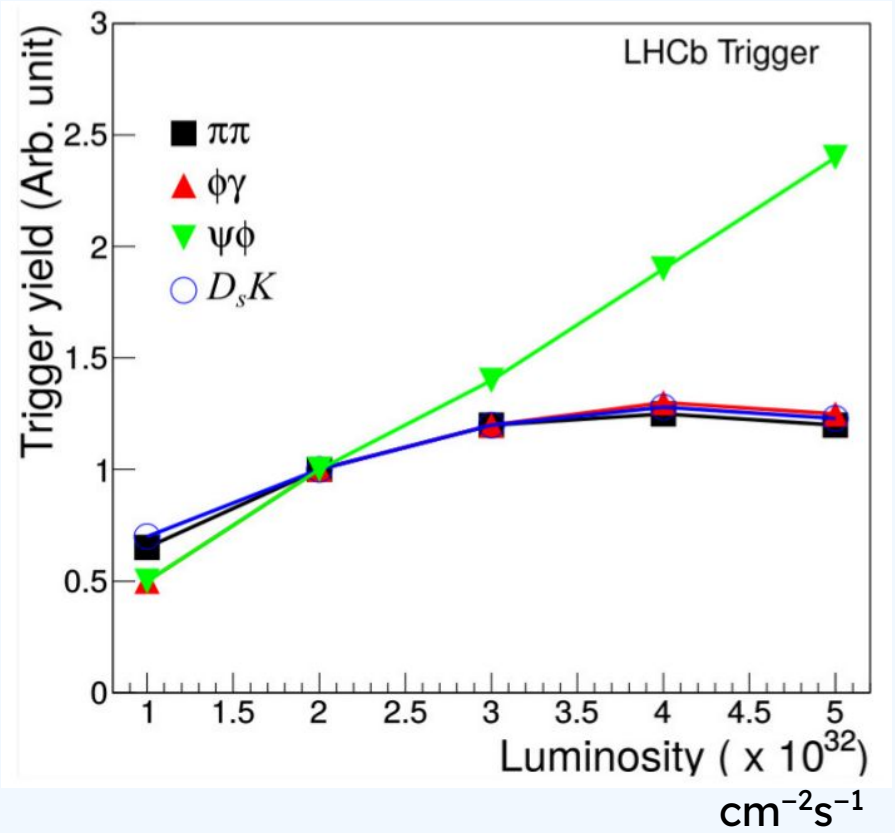
Conventional Trigger Limitations

The first level trigger (L0) is implemented in hardware:

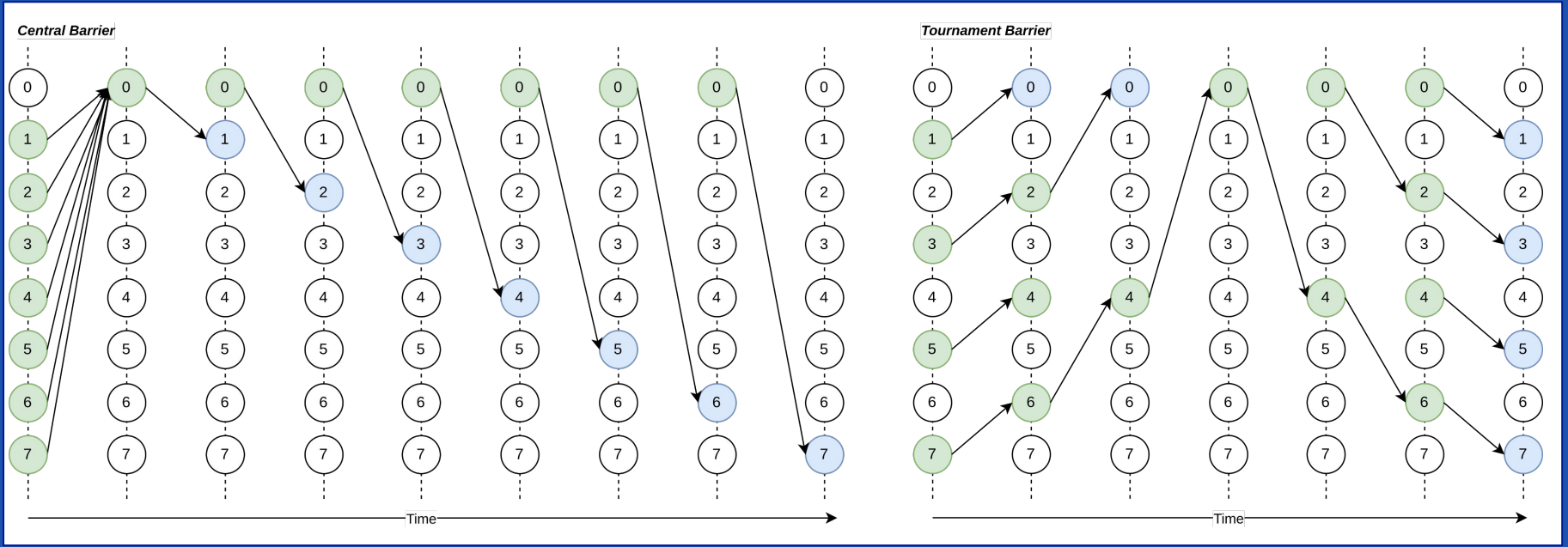
- Selections are made at 40 MHz using Calorimeters or Muon Systems
- Criteria are based on high deposits of transverse energy by charged particles and photons
- Output rate to HLT1 is 1 MHz

Hadronic modes saturate yield in Run II.

An upgrade is needed to run at higher luminosity (5 times the luminosity of Run II).



Synchronization Algorithms



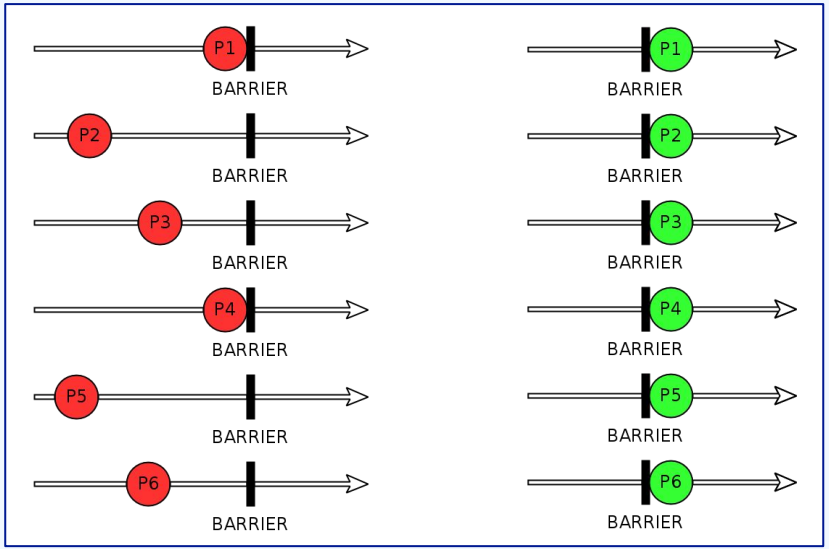
Synchronization

Strong synchronization between all nodes is done at every step of the scheduling.

Synchronization Barrier:

- Processes report they reached the barrier and wait for release
- When all processes have reached the barrier, they are released.

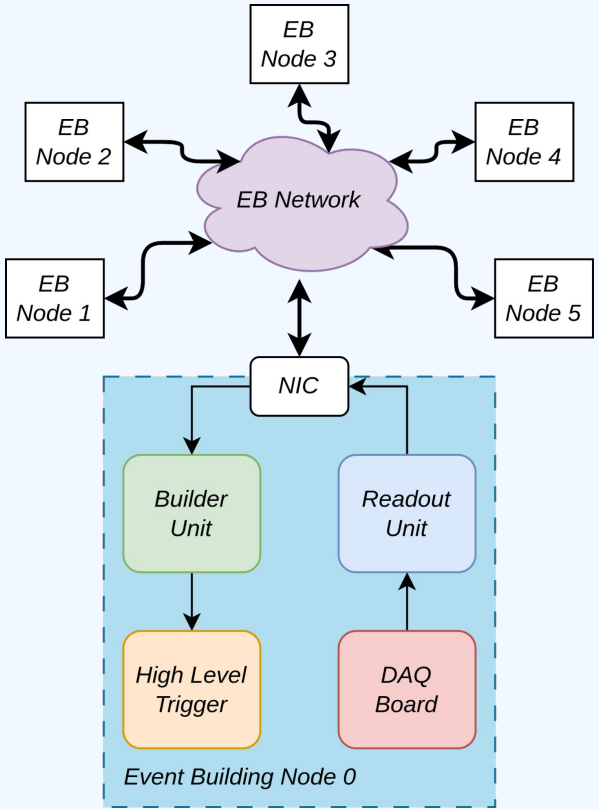
Multiple barrier algorithms are implemented in the communication library.



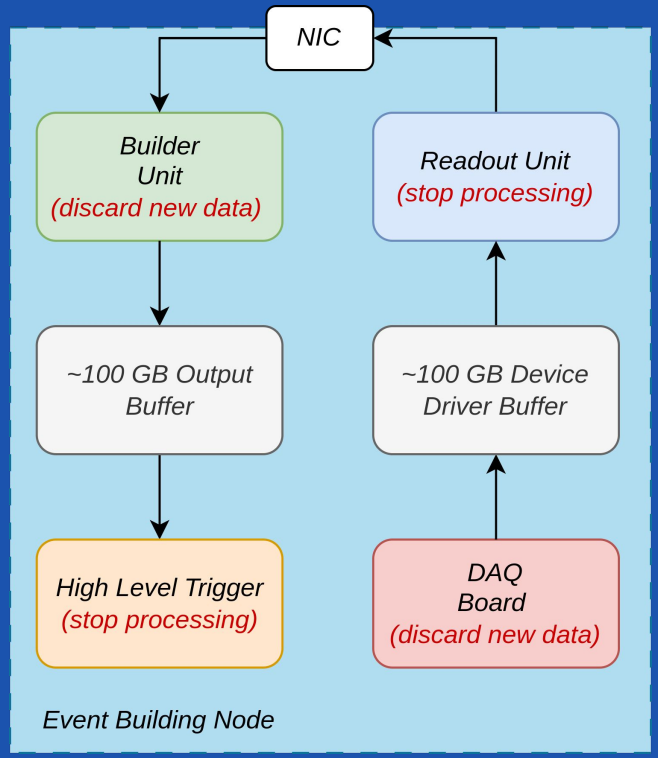
Event Builder Architecture

The key features of the EB architecture are:

- Modular design built in C++
- Large amount of RAM buffers to absorb latency spikes and to shape the traffic
- Use of DMA and Remote DMA to lower memory bandwidth usage
- Dedicated low-level communication library to optimize network performance
- Barrier-based scheduling to guarantee strong synchronisation
- Buffer-isolated critical sections to minimise slowdowns and dead-time



Back-pressure and Fault Tolerance

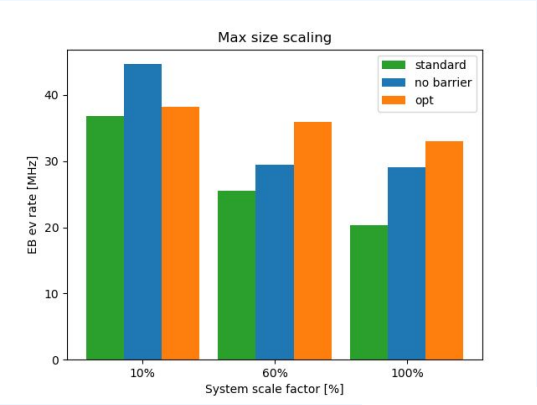


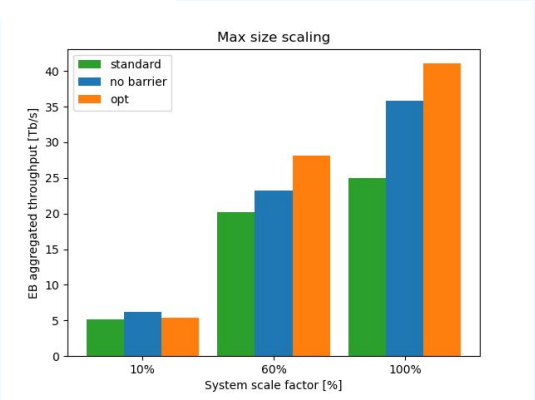
Big RAM buffers at the input and output of the EB are used to reduce latency spikes and back-pressure.

Discard Policies on the Readout Cards and Builder Units also allow to contain back-pressure propagation.

Node-related issues can be solved by moving the EB units to a different host.

Synchronization Performance





With a few nodes, the barrier adds an overhead, reducing the performance.

With the full scale system, the strong synchronization ensures a ~6% higher throughput.

Optimization of the algorithm is crucial to get the maximum performance.

A 40% difference has been measured between the central algorithm and the tournament one.

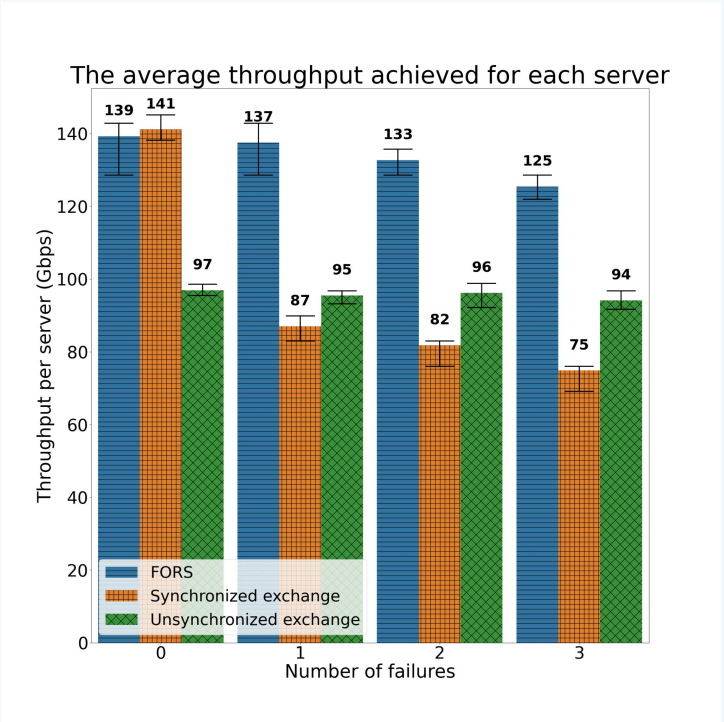
10.1109/TNS.2023.3240514

Network Fault Tolerance

Traffic scheduling requires specific routing to ensure maximum bandwidth.

Link failures can greatly reduce the total throughput by creating congestions.

A fault-adaptive congestion-free routing and scheduling solution has been developed to make efficient use of the remaining bandwidth.



10.1109/LCN58197.2023.10223324

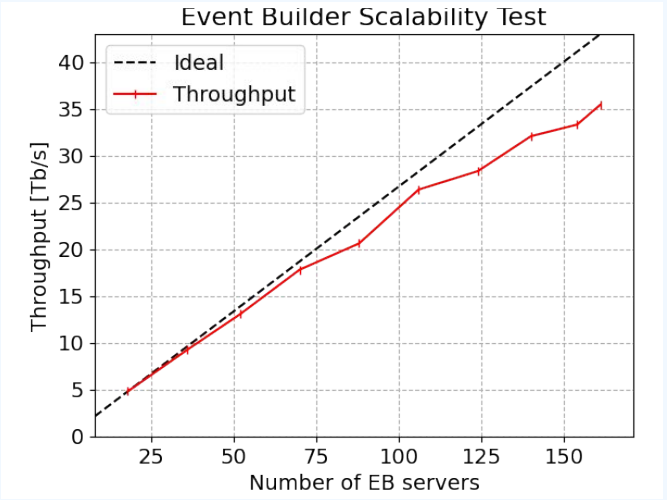
Event Builder: Implementation Results

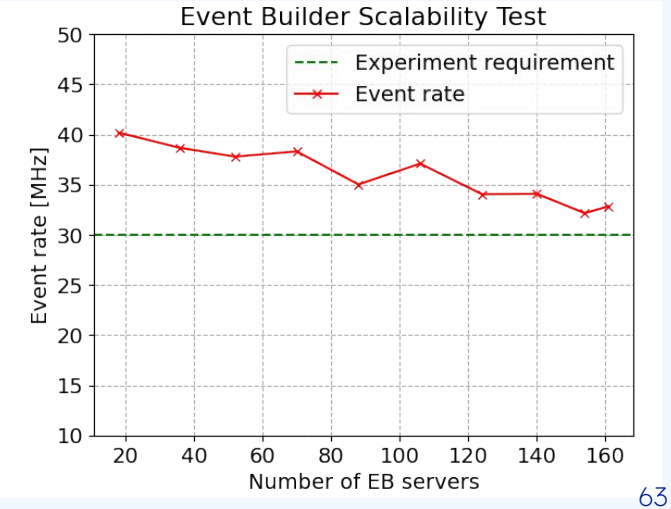
Advantages

- ✓ Removed physics inefficiencies related to the hardware trigger
- ✓ Robust system with respect to network latency spikes
- ✓ Reduced cost thanks to off-the-shelf components and converged architecture

Disadvantages:

- ✗ The trigger provides non physics functionality
- ✗ Tuning required to optimise PCIe communication
- ✗ Converged architectures are less flexible and reliable





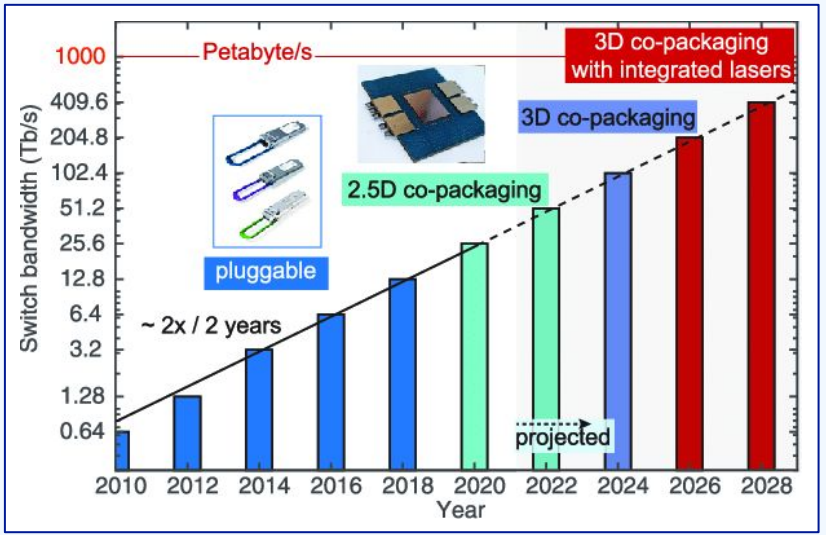
Network Fabric of the Event Builder

The Event Builder in Run 3 uses **InfiniBand HDR** @ 200 Gbps.
This design decision, made in 2020, was based on the performance difference between Ethernet and InfiniBand for this specific use case at the time.

Ethernet, however is **cheaper**, **widely adopted**, and it is evolving at a fast pace.

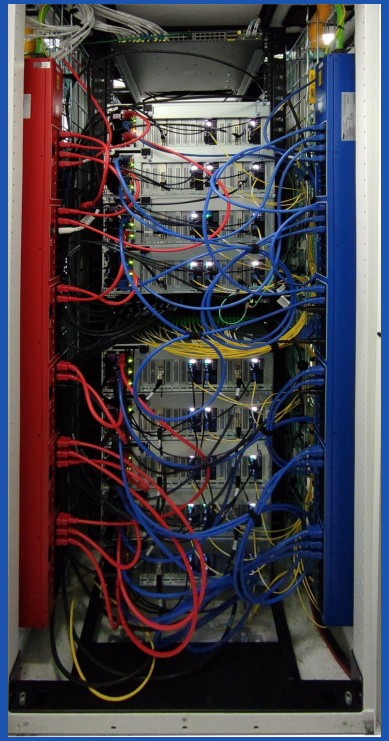
Modern Ethernet ASICs can switch more than **50 Tbps**, with throughput expected to **double every two years**.

Moreover, the current PCIe boards require servers with **sufficient space** to host them, which reduce the available options compared to standard Ethernet NICs.



Event Builder Farm

- 163 EB servers
- 24 racks: 18 EB, 2 control, 4 storage
- 28 40-port IB HDR switches: 18 leaf and 10 spine



HLT1 GPU Scheduling

Events are physically independent and can be reconstructed in parallel.

Parallelism is achieved in three ways:

- Run several sequences in parallel
- Batch multiple events in each sequence
- Exploit data-parallel patterns in each reconstruction algorithm

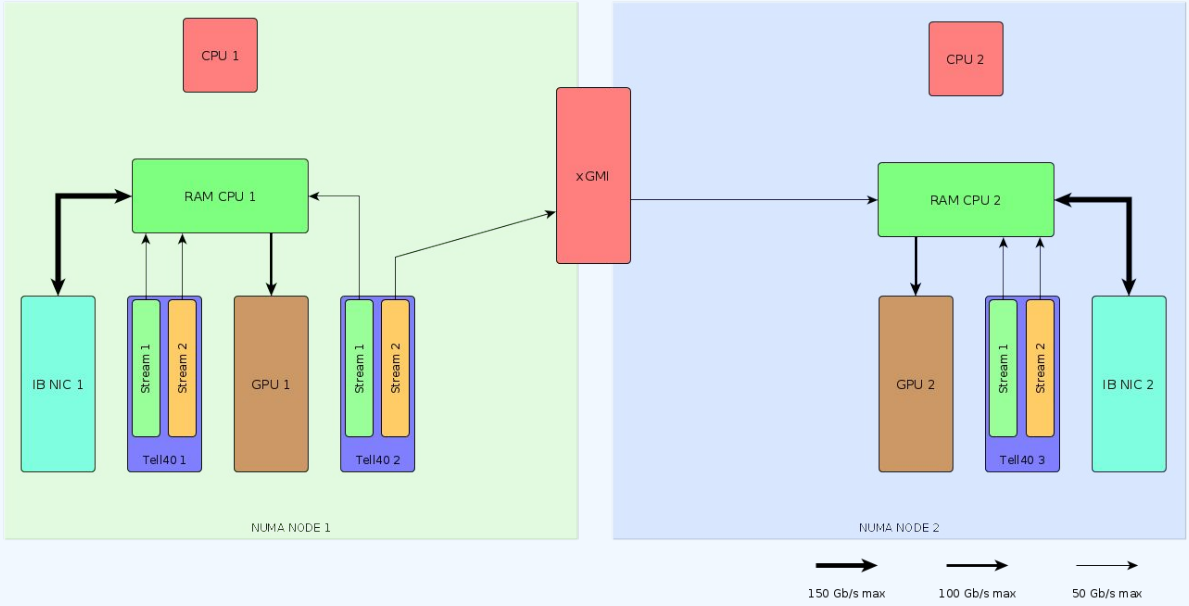
Batching events introduces a scheduling problem.

A multi-event scheduler has been created in order to optimise the execution of the algorithms.

The scheduling heuristics is based on two main principles:

- Spreading algorithms with similar profiles (e.g. data-dependent or compute-dependent)
- Exploit previous sequence knowledge to reduce branching

HLT1 Hardware Architecture

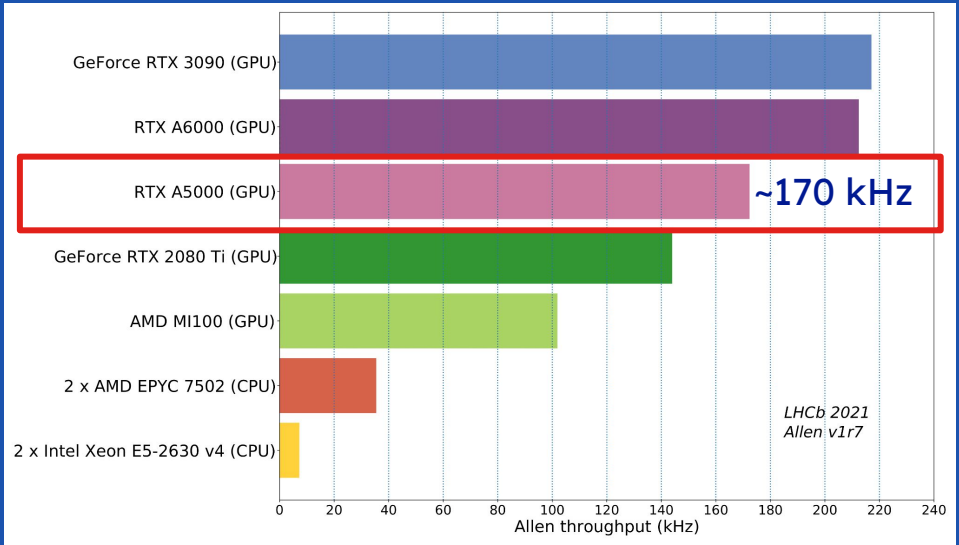


The HLT1 is hosted in the Event Builder farm. (173 servers)

Each server has 3 free PCIe slots to host GPUs, offers sufficient cooling and power, and results in a self-contained system.

This solution results in a cheaper and more scalable architecture than the CPU alternative.

HLT1 Throughput



Throughput is defined as number of events processed in steady-state conditions.

LHCb-FIGURE-2020-014

The requirement of 30 MHz are met with O(200) GPUs, with space in the servers for O(500).

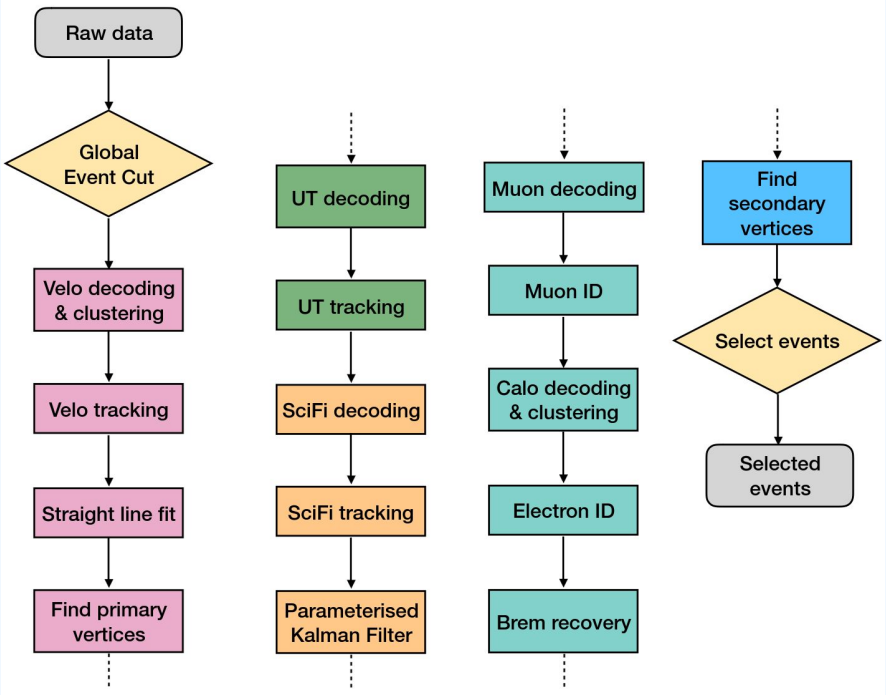
HLT1 scales well with the theoretical TFLOPs available on the GPUs, thanks to its inherently parallelizable tasks.

HLT1 Sequence

Tracking is at the core of the HLT1 reconstruction. It relies on 3 sub-detector systems:

- **VELO** : clustering, tracking, and vertex reconstruction
- **UT**: track reconstruction, momentum estimation, fake rejection
- **SciFi**: track reconstruction, momentum measurements

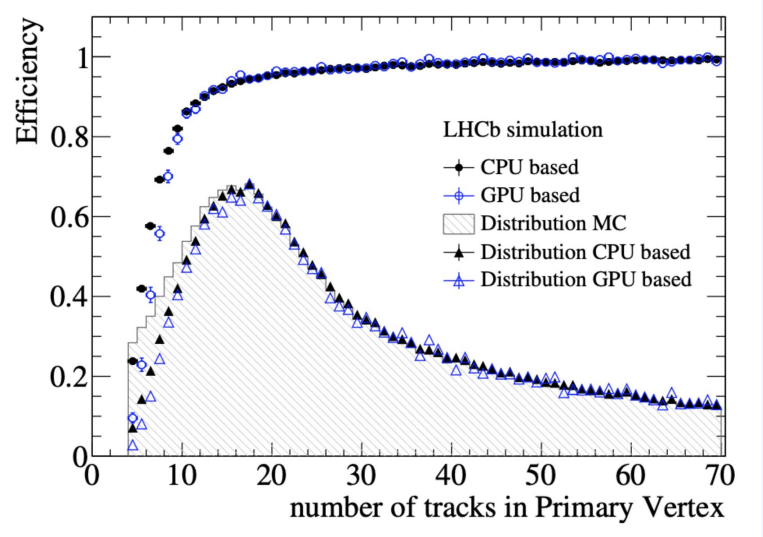
It also uses the muon stations and the calorimeter to do particle identification.



LHCb-TALK-2022-175

HLT1 Performance

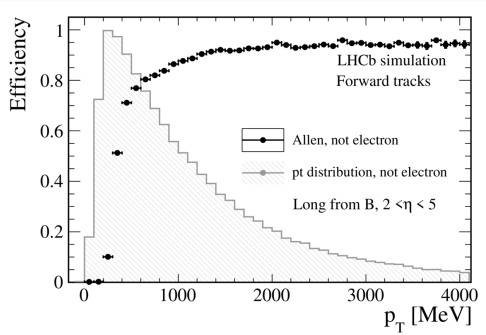
Vertex Reconstruction:
Efficiency > 90% for vertices with >10 tracks



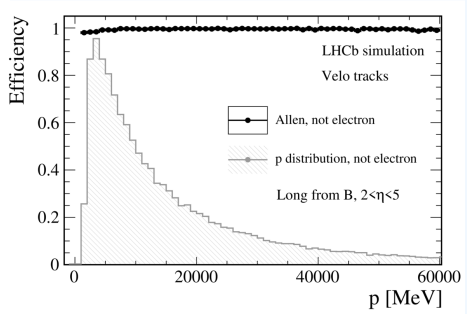
10.1007/s41781-021-00070-2

Tracking Performance:

- **Efficiency > 99%** for VELO, **95%** for high-p forward tracks
- Good momentum resolution and fake rejection



LHCb-FIGURE-2020-014

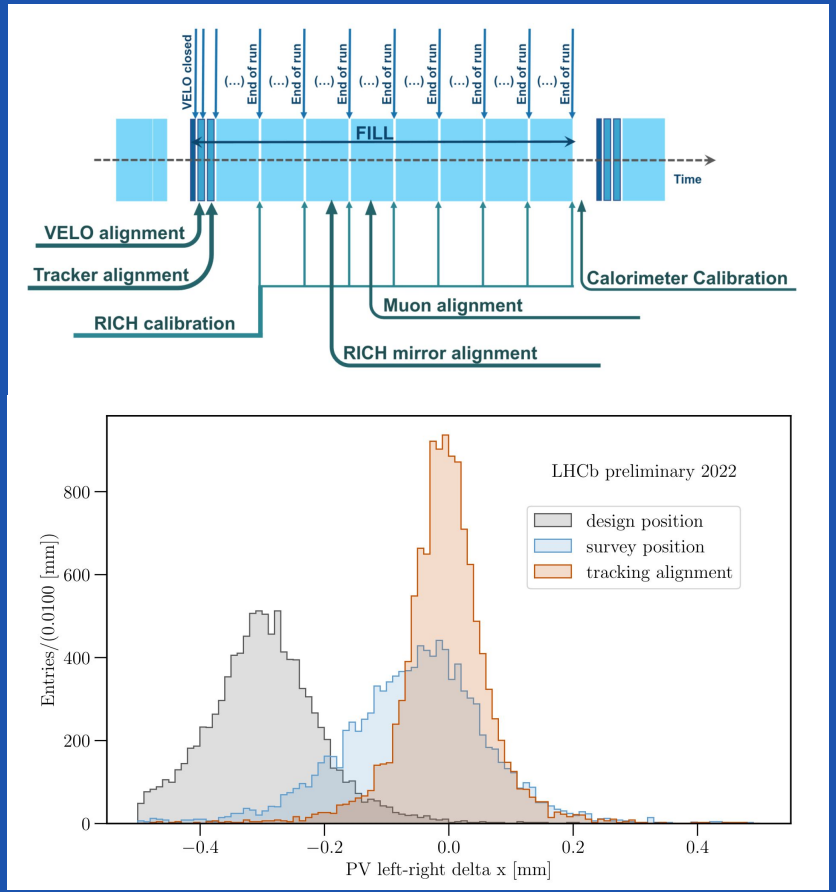


Real-time alignment and calibration

Dedicated trigger lines in HLT1 are used to select events for the alignment procedure.

Data must be fully aligned and calibrated before running the second High Level Trigger (HLT2).

This is managed by storing the events in a buffer of O(30) PB, giving enough time for the alignment and calibration procedure to complete.



LHCB-FIGURE-2022-016

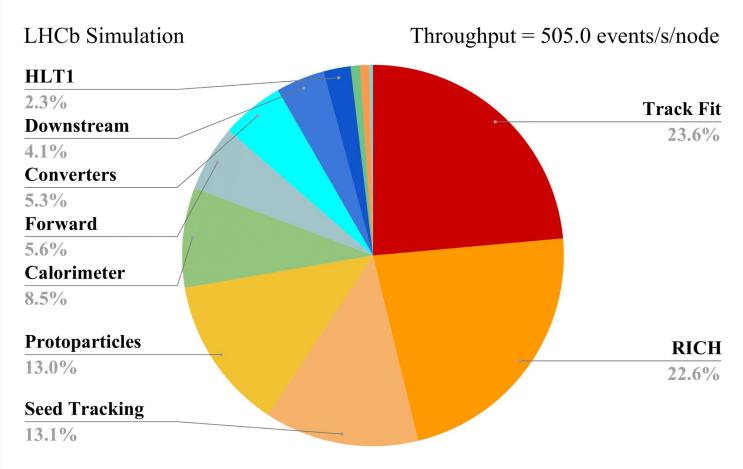
The Second Level Trigger (HLT2)

HLT2 fully reconstruct offline-quality tracks and neutral objects, including particle identification information.

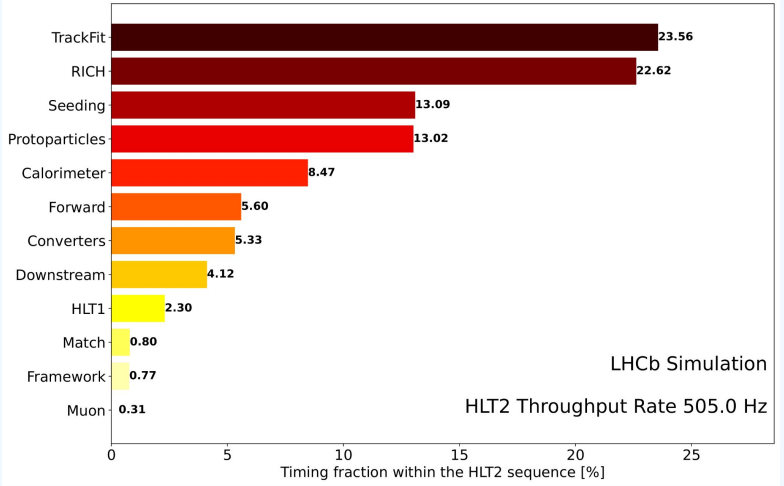
Fixed output throughput limited at 10 GB/s requires an output event rate of ~100 kHz.

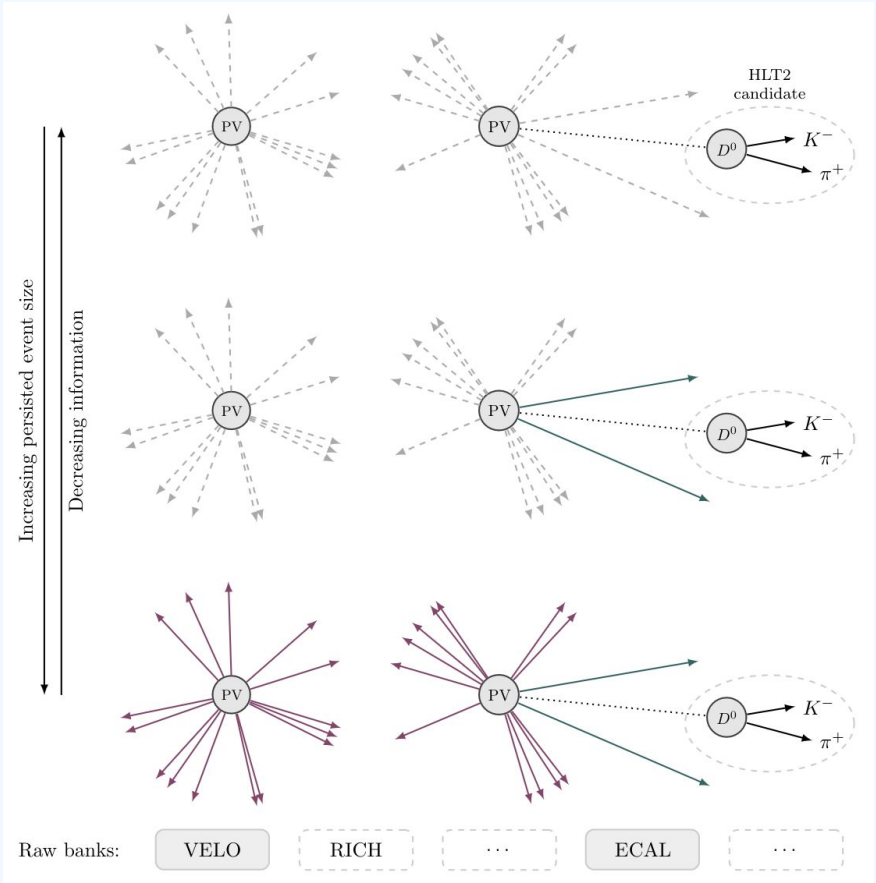
HLT2 runs asynchronously to the LHC, distributing the workload over times where no stable beam is present.

A total of O(1000) trigger lines are used to decide over the persistence of a given event.



LHCB-FIGURE-2022-005





HLT2 Event Persistence

Due to bandwidth limitations, different persistence levels are available:

- **Turbo**: Save raw and reconstructed data only from signal candidate that passed a HLT2 line selection
- **Selective Persistence**: Save selected raw and reconstructed data
- **Full stream**: Save the raw and reconstructed data of the full event

Going from Full Stream to Turbo is a ~10x decrease in size.

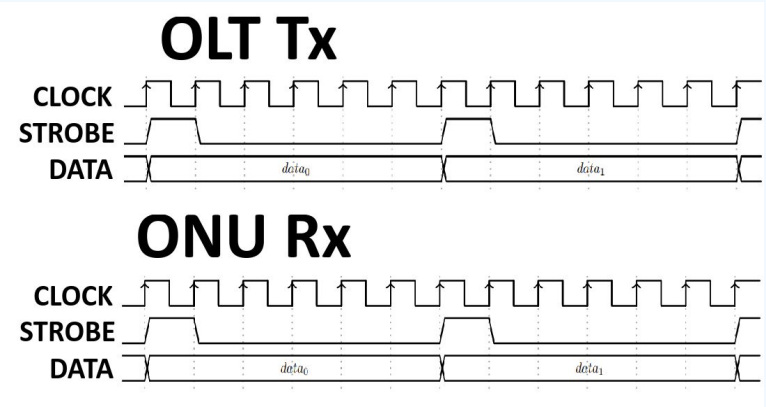
10.1088/1748-0221/14/04/P04006

Clock Recovery

TTC-PON manages clock and data recovery using FPGA transceivers' functionality.

The OLT begin to transmit a new word containing a fixed header when a strobe pulse is provided by the user.

The ONU identifies the header position in the serial stream and generates a strobe pulse phase-aligned with the header.

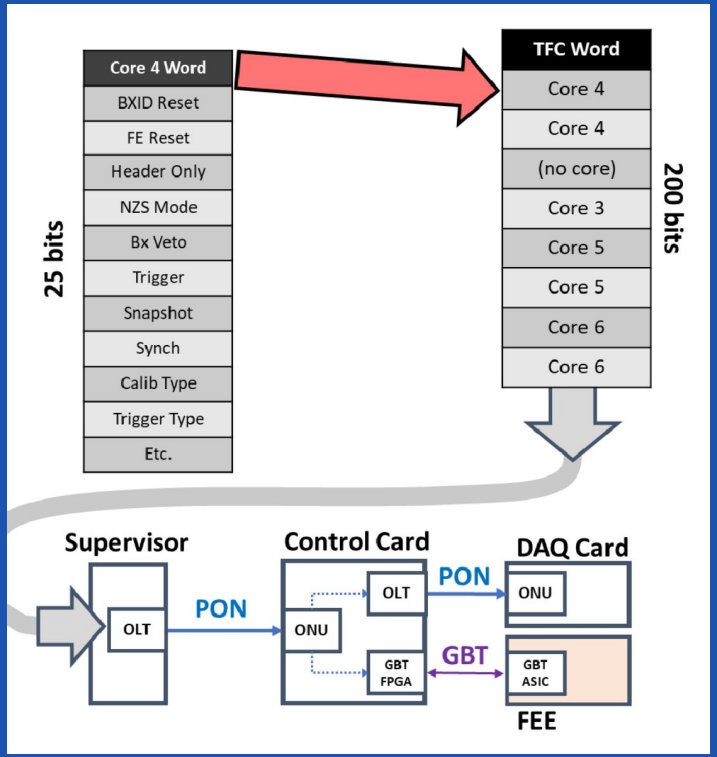


10.1109/TNS.2023.3273086

The waveform might look the same, but:

- In the OLT, strobe drives the data
- In the ONU, strobe is generated from data

The Readout Supervisor (SODIN)



10.1109/TNS.2023.3273086

Central component of the TFC system implemented in custom gateware.

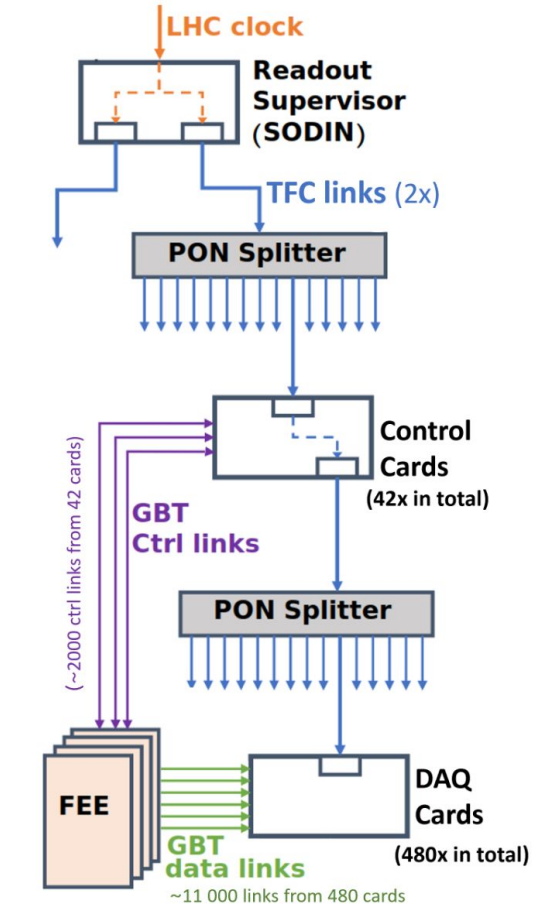
Implements 10 independent cores that control different partitions of the detector simultaneously.

Each core can generate commands towards the DAQ and the FEE, controlling a set of subsystems.

25 bits are available to each partition to forward commands to the control cards.

Control cards relay information to FEE or DAQ sequentially.

Real-Time Monitoring



The TFC endpoints can also send data upstream through the PON interface.

Data is transmitted using a Time Division Multiplexing (TDM) scheme. Each ONU occupies a pre-assigned time slot when transmitting back.

In the worst-case scenario (reading from *all* endpoints in the system), the readout period would be 8us (320 clocks).

An application of this upstream link is throttle management. A throttle signal is transmitted when buffers in the DAQ cards are full. The supervisor then reduces the trigger rate accordingly.

10.1109/TNS.2023.3273086

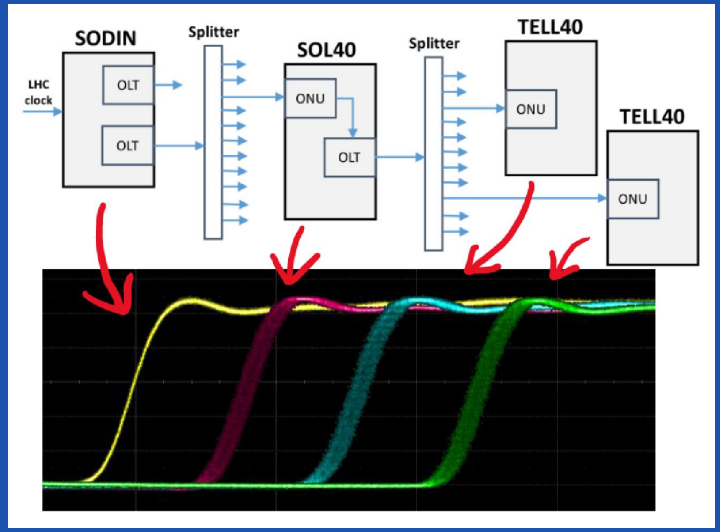
Clock Jitter and Skew Measurements

The requirements of the TFC are:

- **Clock Skew:** Phase difference between the LHC bunch clock and each recovered clock must be < 500 ps
- **Clock Jitter:** The total RMS jitter of each clock must be < 100 ps
- **Bit Error Rate:** The links' BER must be < 1.0×10^{-15}

Measurements reported:

- Maximum clock skew of 254 ps (σ = 76 ps)
- Random Jitter of 13.5 ps (considering 76 ps as max)
- BER < 3.1×10^{-17} (95% C.L.)



10.1109/TNS.2023.3273086

Event Size Model

Sub-detector	fragment size [B]	#tell40 streams	event size [B]	event fraction	MEP size [GB]	MFP size [MB]	RU send size [MB]
Velo	156	104	16250	0.13	0.49	4.69	14.06
UT	100	200	20000	0.16	0.60	3.00	9.00
SCIFI	100	288	28800	0.23	0.86	3.00	9.00
Rich 1	166	132	22000	0.18	0.66	5.00	15.00
Rich 2	166	72	12000	0.10	0.36	5.00	15.00
Calo	156	104	16250	0.13	0.49	4.69	14.06
Muon	156	56	8750	0.07	0.26	4.69	14.06
Total	1000	956	124050	1	3.72	30.06	90.19

* Actual Size depends on Luminosity

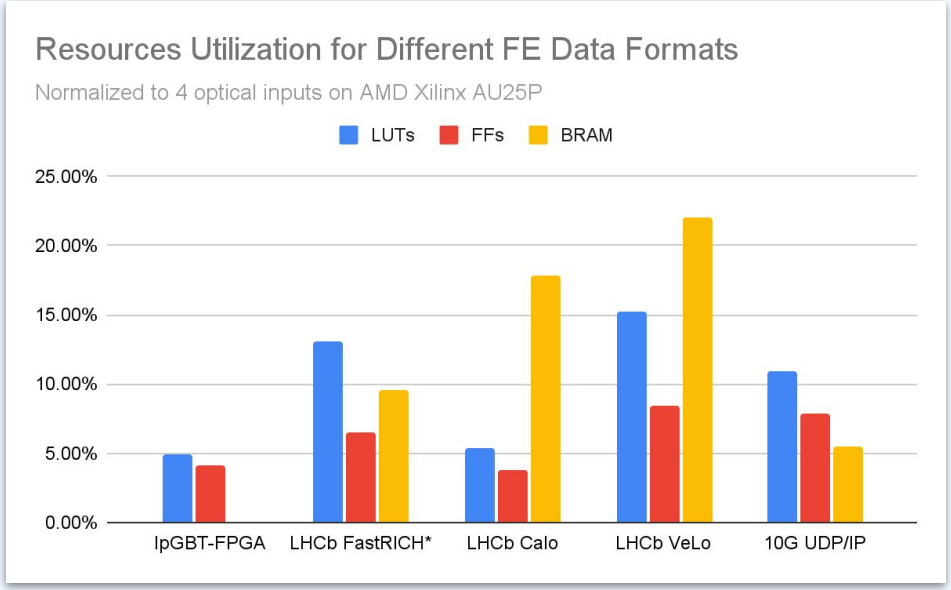
Data Processing

The current gateware does **only packing** of the Front-End data, using a small amount of resources.

We evaluated the footprint of **different data processing** elements on the FPGA to understand how much can be offloaded on the device.

Both simple (CALO) and complex (VeLo) data processing can be fitted on the FPGA.

Key consideration: balance the ratio between resources and transceivers to select the optimal device.



**FastRICH only implements Aurora 64b/66b decoding*

Other HEP Experiments

ATLAS

- New Trigger System based on modern FPGAs to enhance compute
- New FPGA-based PCIe DAQ boards (FELiX) capable of handling links at 25 Gb/s and supporting high precision timing
- New Event Filter with an heterogeneous architecture using accelerators such as GPUs and FPGAs

DUNE

- DAQ shifts from PCIe to Ethernet to reduce costs while maintaining data throughput

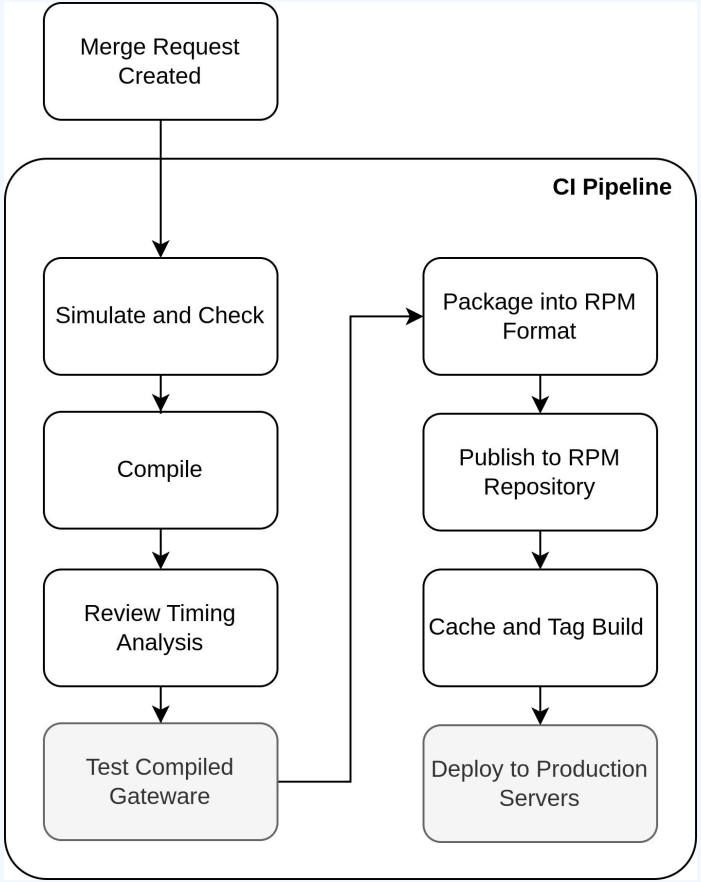
Continuous Integration

An automated pipeline is used to compile the different gateware configurations.

A GitLab repository oversees the different configurations which are imported as submodules.

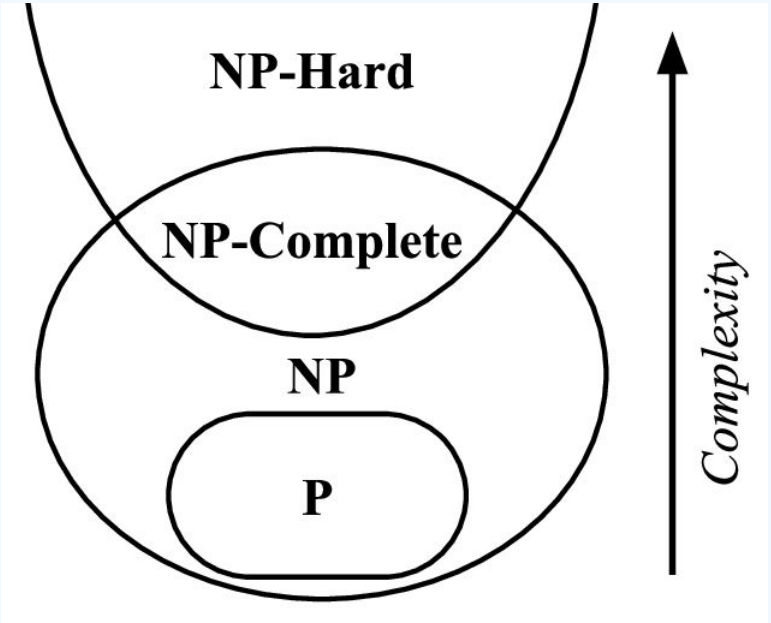
Merge requests trigger the compilation pipeline which will run only the jobs of which sources have been changed.

Dedicated servers execute the pipeline jobs.



Computational Complexity Classes

- **P** are problems that can be solved in polynomial time.
- **NP** are problems which solutions can be verified in polynomial time. Finding the solution however can be hard.
- **NP-hard** problems are *at least* as hard to solve as the hardest NP problem. A solution to NP-hard problems can be used to solve all NP problems in polynomial time.
- **NP-complete** problems are both NP and NP-hard.



Aurora Implementation Results