ALPIDE SystemC Simulations

Simon Voigt Nesbø

Department of Computing, Mathematics, and Physics

October 12, 2017

Western Norway
University of
Applied Sciences

Part 1 - Introduction

Introduction to ALICE, ITS, ALPIDE

ALICE - A Large Ion Collider Experiment

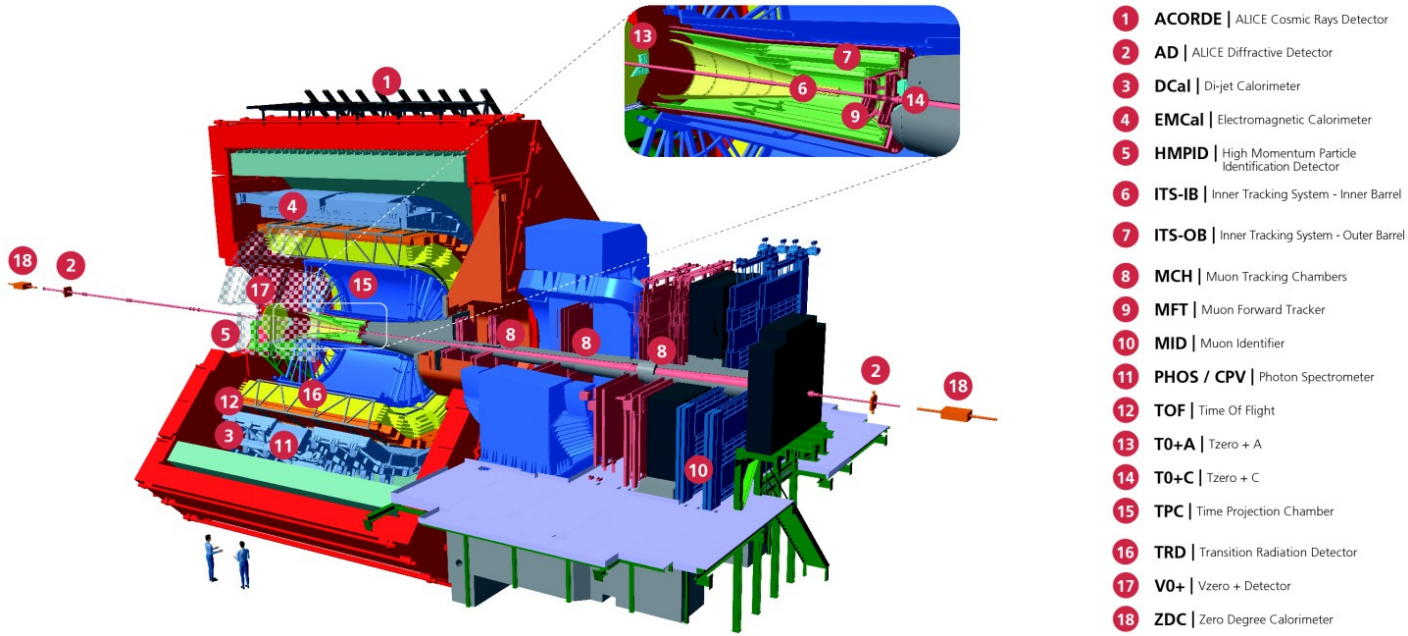


Figure: ALICE Schematics as during RUN3 (after upgrade) [4].

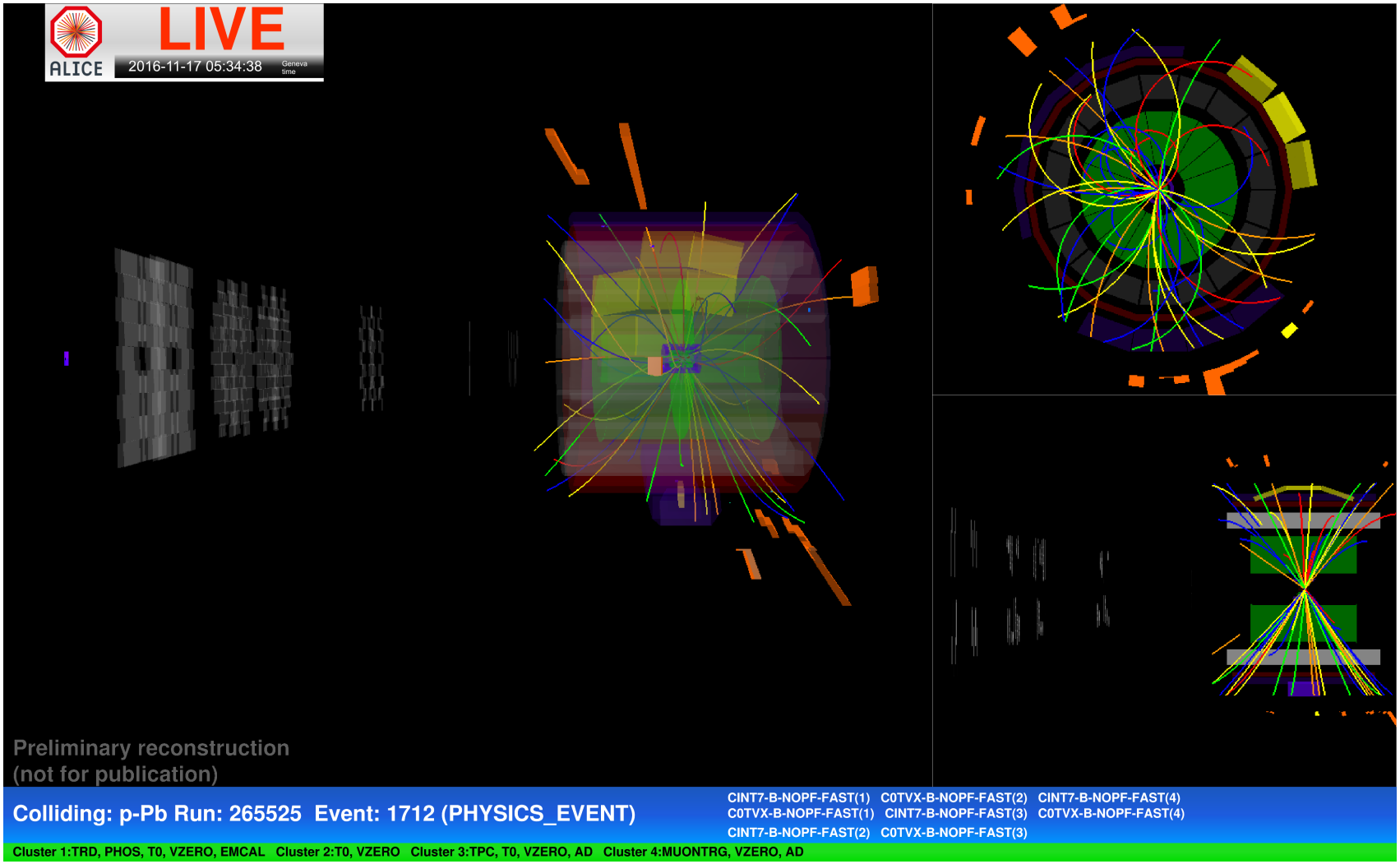
An experiment designed to study:

- Pb-Pb collisions
- Quark-Gluon Plasma (QGP)
- Quantum ChromoDynamics (QCD)

Quark-Gluon Plasma?

- State of matter where quark and gluons “float around freely” in a plasma
- Extremely high densities and temperatures required.
- The Universe was in a QGP state the first few milliseconds after the Big Bang

Typical ALICE Event



Long Shutdown 2 Upgrade of ALICE

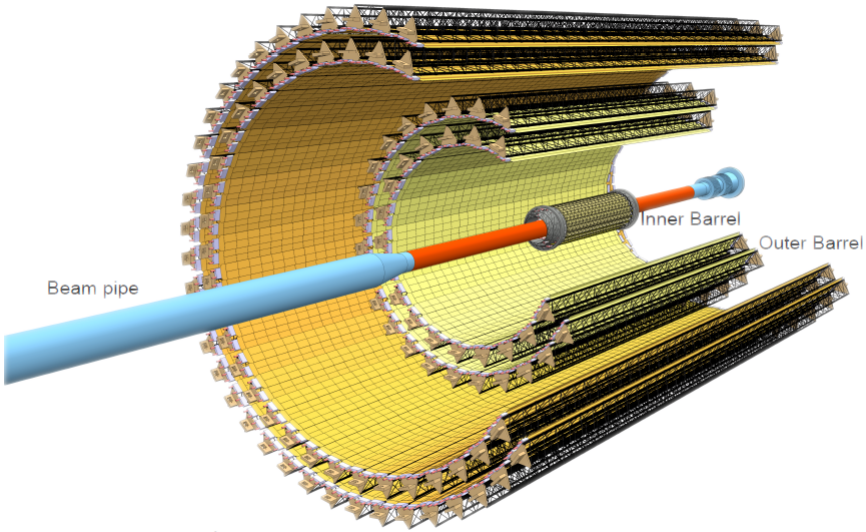
ALICE will undergo major upgrades during the Long Shutdown planned in 2018 (LS2). Data collection is planned to resume in 2020.

Upgrade objectives:

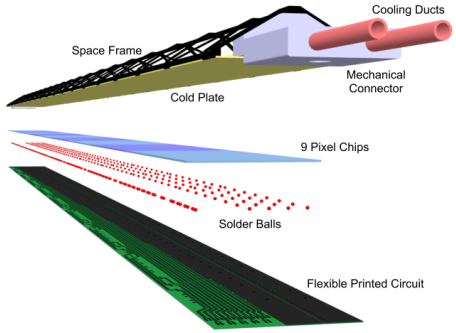
- Physics goals:
  - QGP (still)
  - Higher luminosity $\rightarrow$ More statistics $\rightarrow$ Greater accuracy
- Higher event rates
  - 50 kHz Pb-Pb (currently 8 kHz)
  - 200 kHz pp (currently 8 kHz, readout limited to 500 Hz)
- Minimum-bias triggered or continuous “sampling”
  - Collect every event, not limited to those deemed “interesting”

Inner Tracking System LS2 Upgrade

Upgraded ITS detector



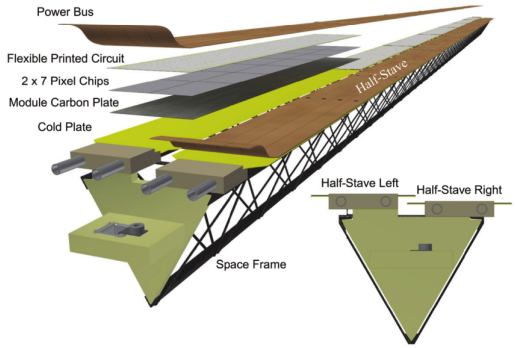
Inner Barrel Stave



Layer configuration:

- 7 layers of Silicon Pixel Detectors
  - Inner Barrel - 3 layers
  - Outer Barrel - 4 layers (2 middle, 2 outer)
- All layers based on the ALPIDE chip

Outer Barrel Stave

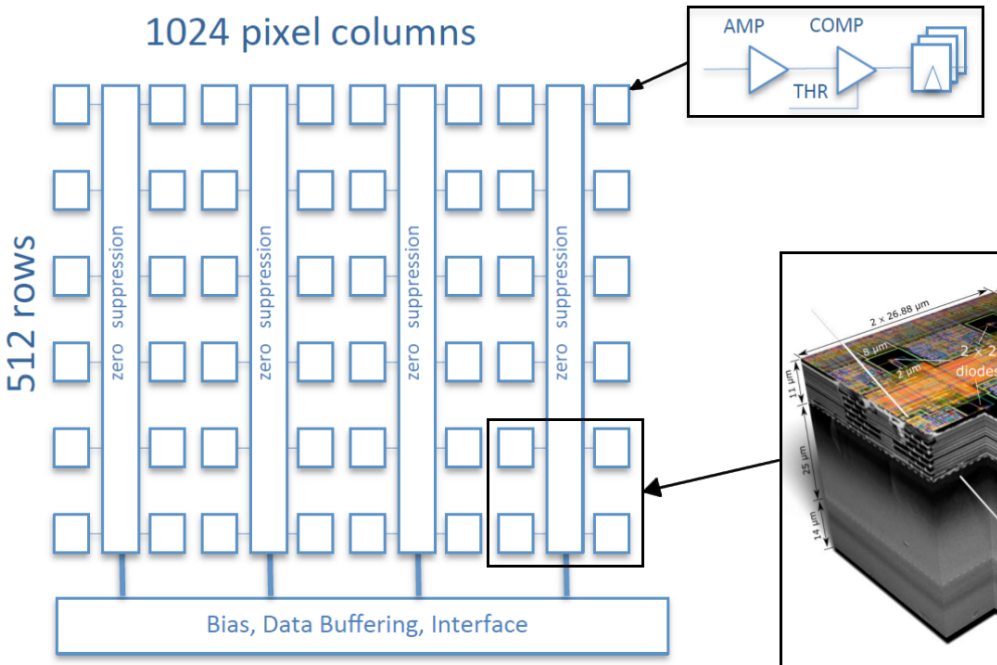


Maximum event rate (2x ALICE design goal):

- 400 kHz pp
- 100 kHz Pb-Pb

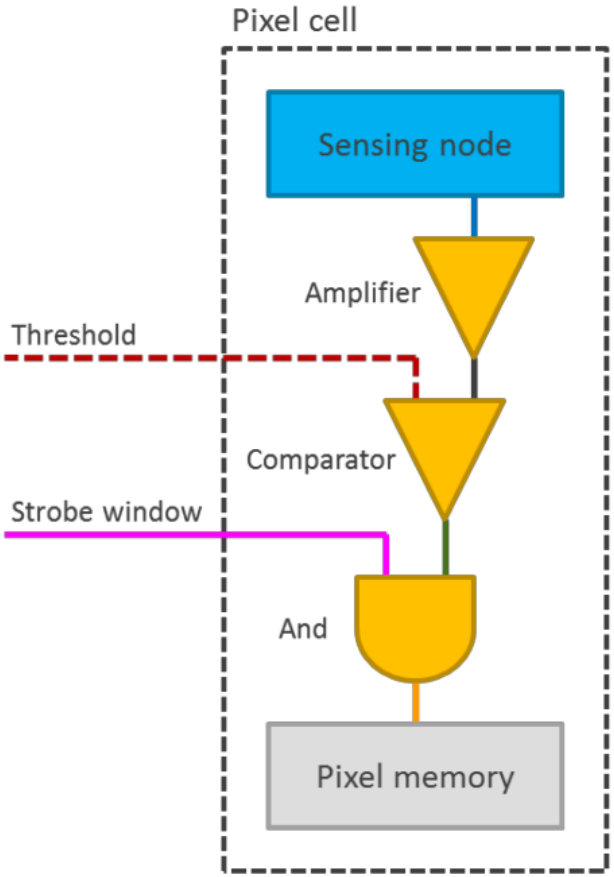
The ALPIDE Chip

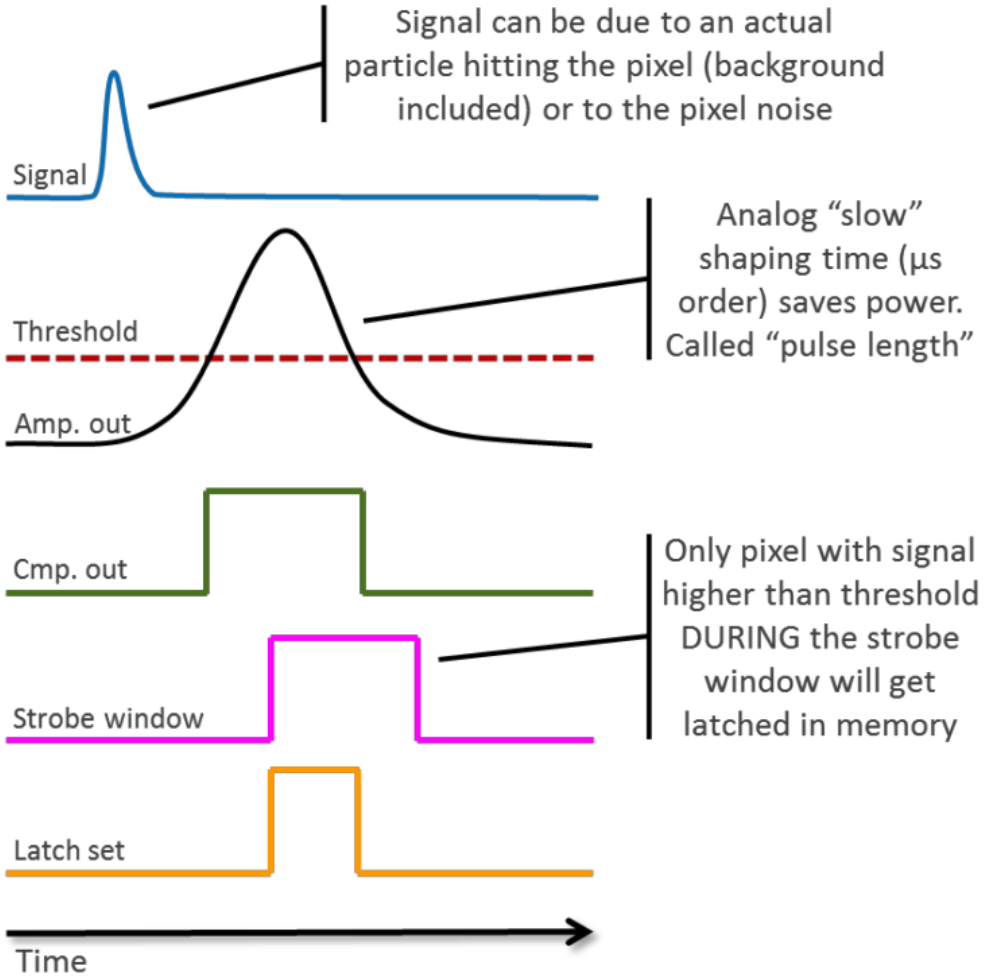
- 1024 x 512 pixels
- 32 regions of pixel columns
- Amplifier and discriminator circuit in each pixel
- 3-event hit memory in each pixel
- Zero suppression: only pixels that are hit are read out



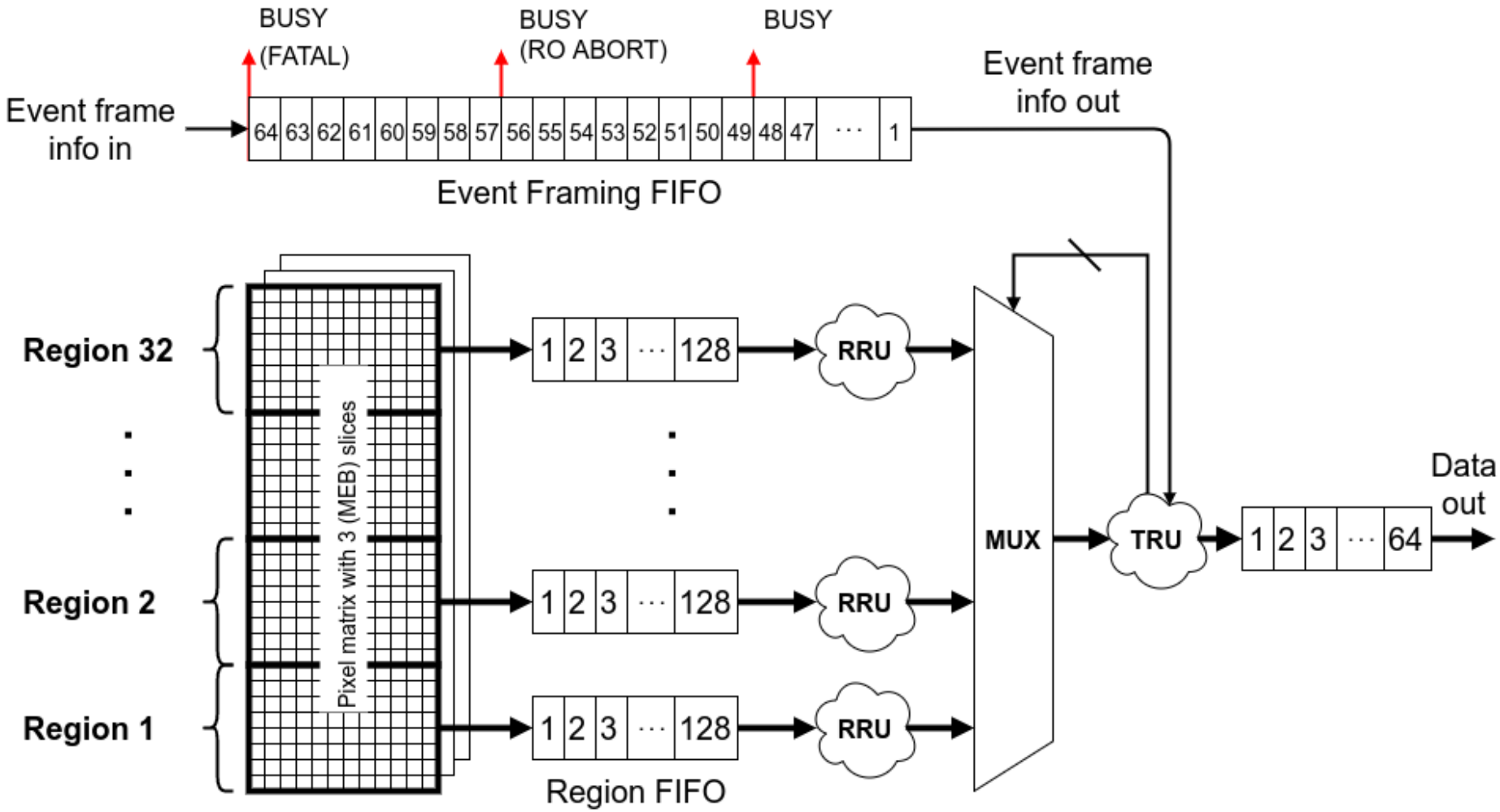
Spatial resolution	$\sim$ 5 µm
Power	$\sim$ 300 nW/pixel
Fake-hit rate	$\sim 10^{-10}$ pixels/event
Max particle hit rate	100 MHz cm^{-2}

ALPIDE Pixel Front End and Timing





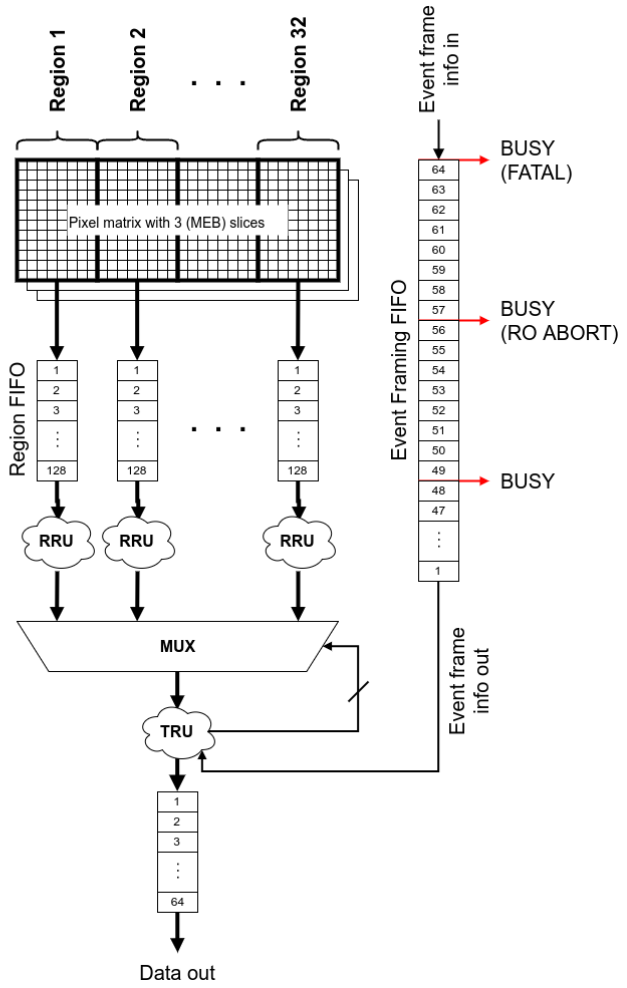
ALPIDE Dataflow Overview



Readout in ALPIDE chip

3 parallel processes in ALPIDE chip:

1. Trigger input
   - ALPIDE receives a trigger
   - Pixel hits stored in one MEB slice
   - A frame word with event ID etc. is pushed onto the framing FIFO
2. Readout from MEB to region FIFOs
   - The RRU's read hits from the pixel MEB into region FIFOs
   - Up to several events can be stored temporarily in the region FIFOs
3. Readout from region FIFOs, data framing and transmission
   - TRU reads out hits for 1 event from the region FIFOs
   - The data is organized in a frame with info from the framing FIFO
   - The data is transmitted off the chip



Busy mechanisms in ALPIDE chip

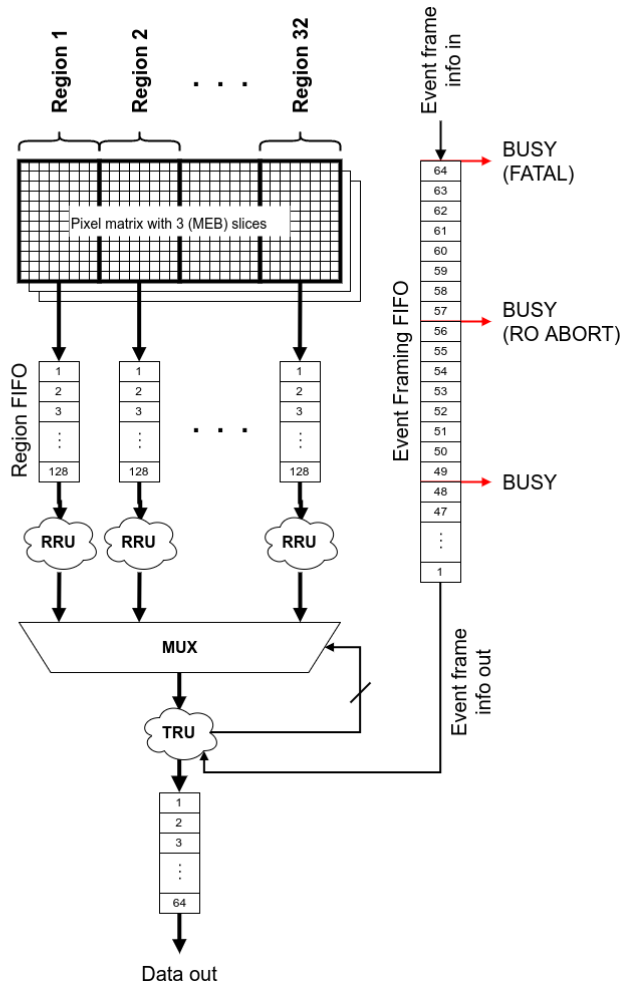
The ALPIDE is **busy** when it is not ready to receive another trigger. When this happens, it will immediately send out a BUSY word on the data stream to the RU.

When is the chip busy?

1. All MEBs are full
2. Framing FIFO reaches critical levels

What happens if we send another trigger when the chip is busy?

- Trigger is ignored, or old data is discarded to make room for new (depends on mode)
- Data is lost in any case
- Event will be marked with a busy violation flag



ITS Detector and Readout

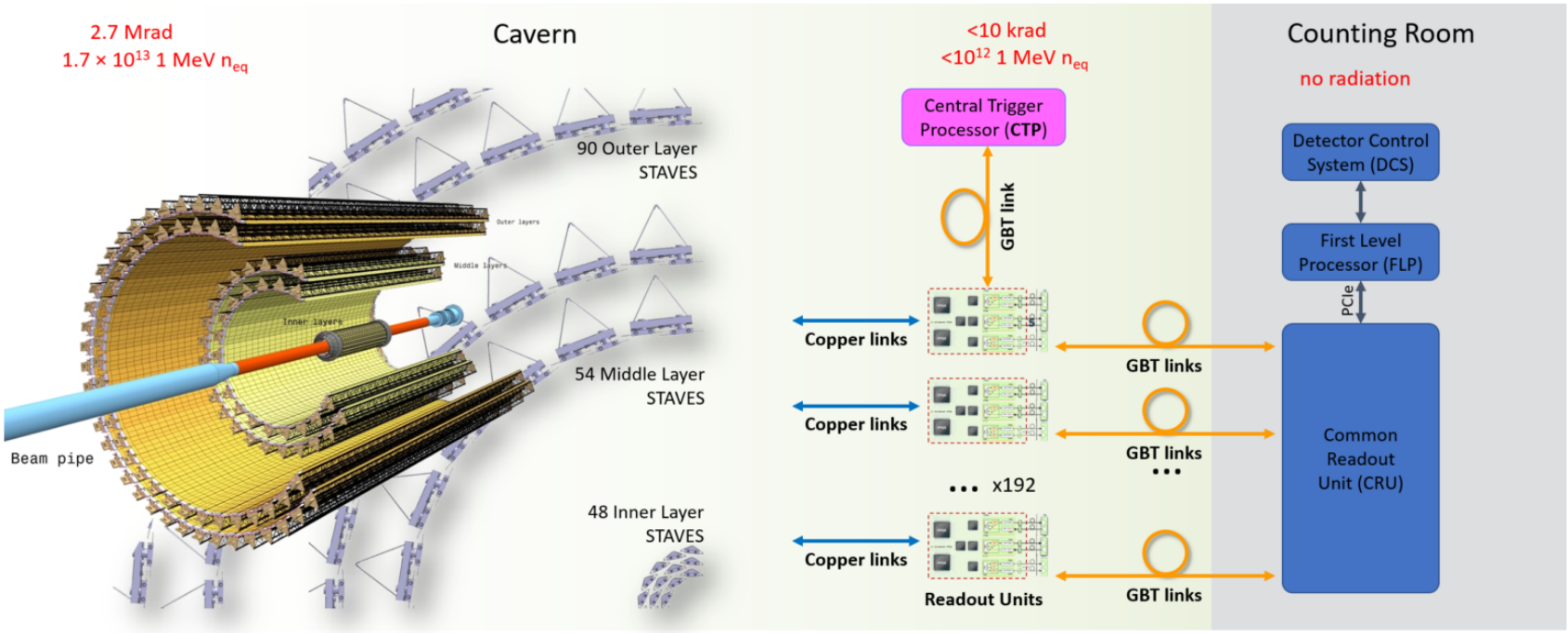


Figure: ITS detector and stave assembly, readout electronics, trigger distribution, and First Level Processors (FLP) [3].

- ITS is organized in 192 “detector staves”.
- Each stave consists of several ALPIDE pixel detector chips
- Each Readout Unit (RU) is responsible for one stave
- The RUs receive triggers from the CTP over optical GBT link, and distribute triggers to the chips
- The RUs receive data from the ALPIDE chips, and transmits it to the CRU over GBT links

ITS Readout Unit

- Main FPGA: Xilinx Ultrascale (SRAM based)
- Auxiliary FPGA: Microsemi ProASIC3 (Flash based)
- 3 optical links
  - Data output from Readout Unit
  - Trigger input from Central Trigger Processor
- 28 ALPIDE data link inputs

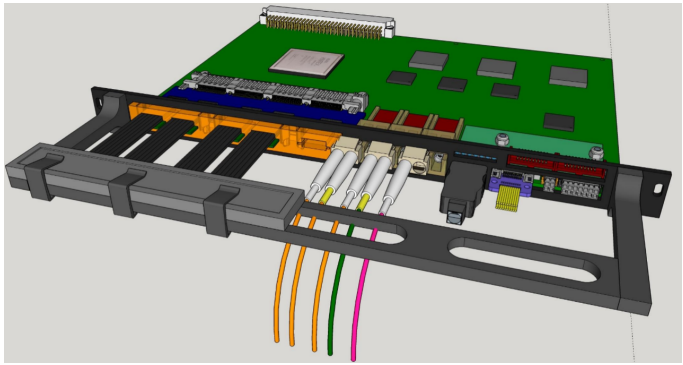


Figure: 3d model of the Readout Unit electronics. PCB designed at University of Utrecht, First prototypes were available in August

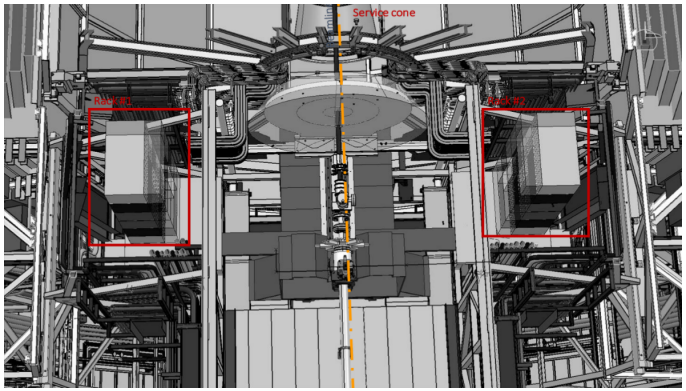


Figure: Position of crates with Readout Units in the ALICE experiment, relatively close to the beam line where it will be exposed to radiation.

Part 2 - Simulations

Simulation models for ALPIDE chip

ALPIDE Simulation Models

Currently there are 4 available simulation models for the ALPIDE chip:

- Full Verilog Model
- Lightweight Alpide Model
- Old SystemC model
- “SystemC Dataflow Model”

Full Verilog Model

The actual model used for chip design
Pros:

- 100% accurate

Cons:

- Very slow to simulate
  - Around 24 hours for 10,000 events
- Restrictions on source code

Lightweight Alpide Model

A more lightweight Alpide model implemented in Verilog/SystemVerilog
Pros:

- Faster than full verilog model
- Command protocol and register interface implemented
- Data output with 8b10 encoding
- Slower than a SystemC model (presumably)

Cons:

- Can not take event inputs
  - Very simple random events are generated in the chip
- Pixel front end, MEBs, and readout is not in the model
- Some restrictions on source code?

Old SystemC Model

A SystemC model that was developed 2-3 years ago, mainly to predict data rates from the different Alpide chips in different ITS layers.
Pros:

- Relatively fast
- Uses MC events generated with aliRoot as input, stored in an XML format

Cons:

- Code predates Alpide-3 chip
  - Hard to see resemblance between logic blocks in Alpide documentation and the implementation in the SystemC model
- Alpide model is part of larger ITS simulation model
- Author is not at CERN anymore, and the code is not maintained
- Very little documentation
- Code is a bit messy

SystemC Dataflow Model

Used to find the optimal way of dealing with busy ALPIDE chips in the RU.

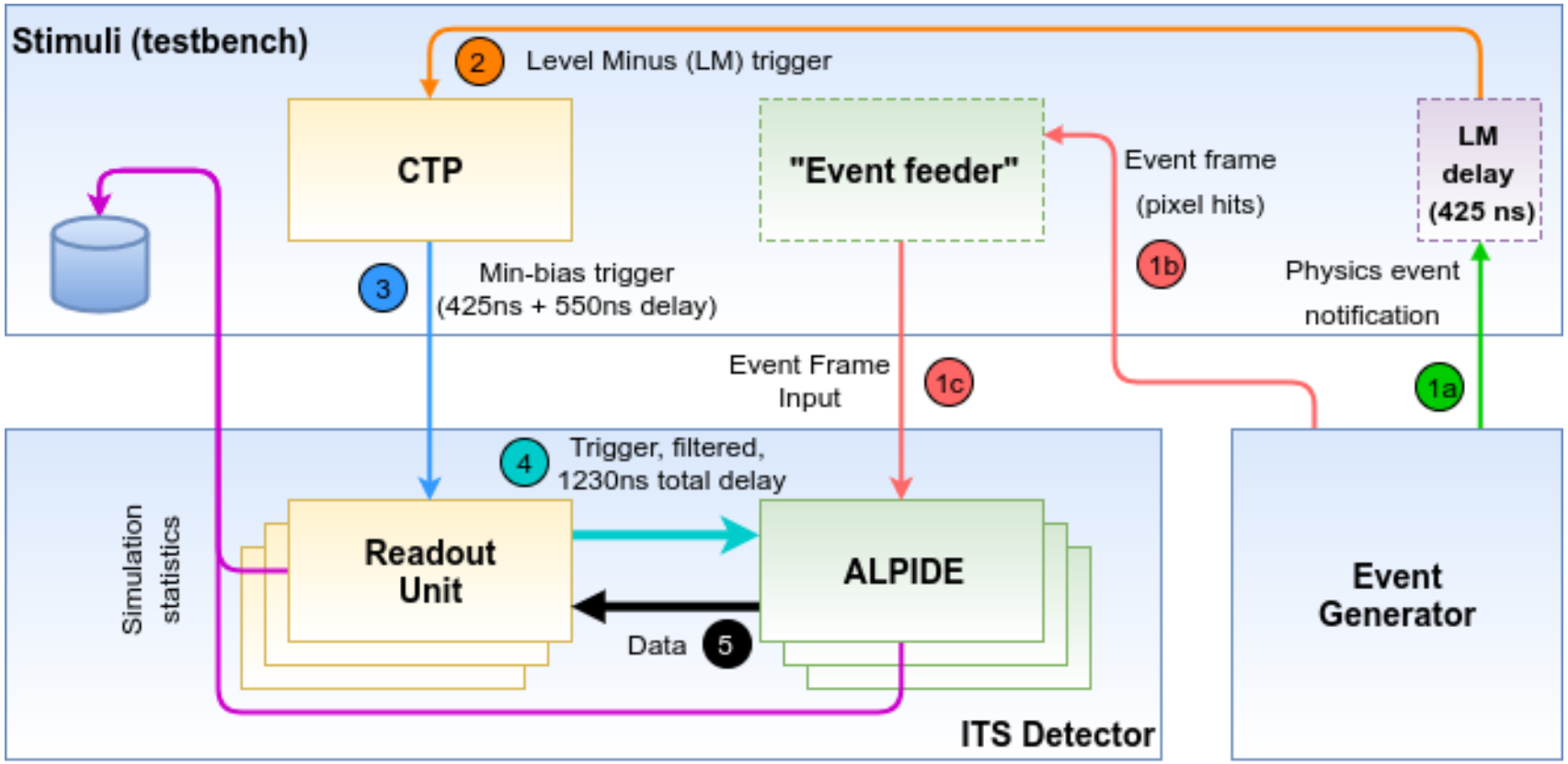
Pros:

- Pretty fast
  - 10,000 events takes around 20 seconds
- Can use MC events generated with aliRoot as input, stored in an XML format (similar to other SystemC model)
- Simple event generator can generate simple events with MC input
- Clear separation between ITS simulation model and Alpide model

Cons:

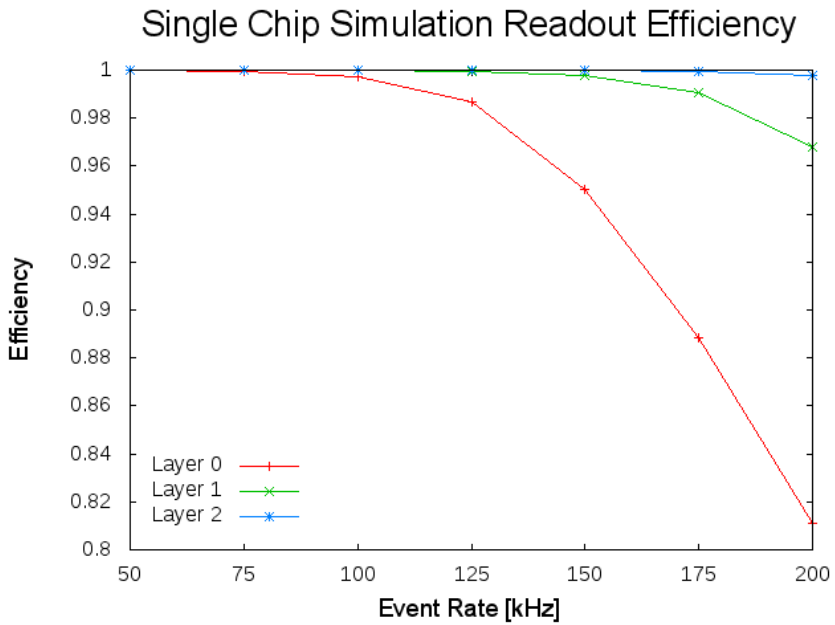
- Less accurate than full Verilog model
- No 8b10 encoding on data output
- No command protocol/register interface, only trigger
- No analog pulse shaping etc, only discriminated hit input (on/off)

SystemC Simulation Model of ITS



Results - Single Chip Simulations

- Figure shows simulation results using the statistical event generator with hit densities expected in the inner barrel.
- Innermost layer:
  - Readout efficiency of $(99.660 \pm 0.002)\,\%$ at 100 kHz event rate
  - Similar to $(99.84 \pm 0.03)\,\%$ obtained with cycle-accurate Verilog model
- Still a work in progress
- Simulations of full ITS detector (including RUs) will be performed in the future



Part 3 - ALPIDE Dataflow SystemC Model

Implementation details for ALPIDE Dataflow SystemC Model

SystemC - Quick Intro

- C++ library for writing hardware simulations in C++
- Introduces concepts such as signals and timing to C++
- C++ classes inheriting from sc_core::sc_module are equivalent to entity in VHDL, module in Verilog
- Can be compiled and run as a standalone program
- Also possible to include in Modelsim simulations etc.

Alpide Dataflow Model Class Diagram

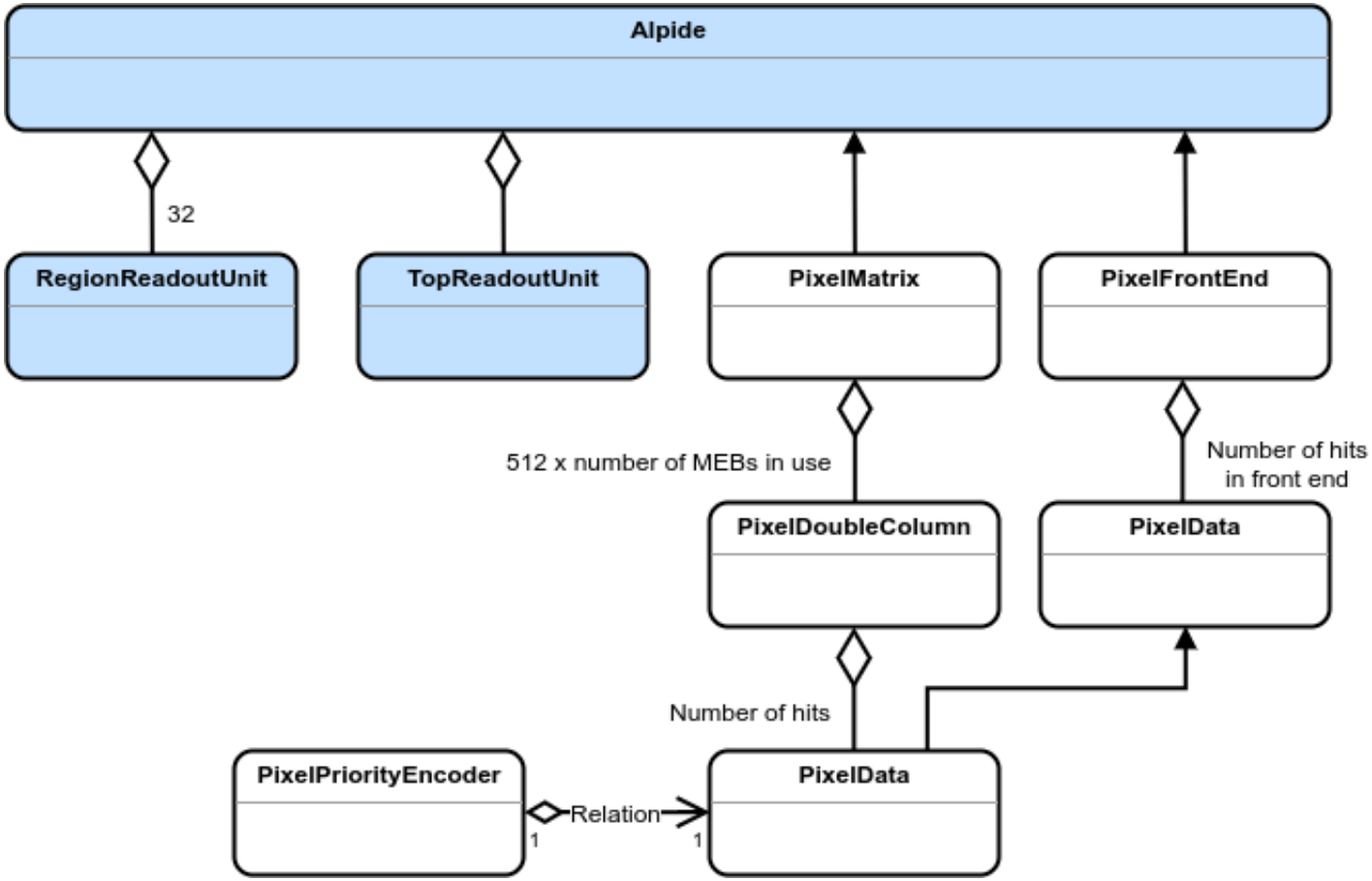
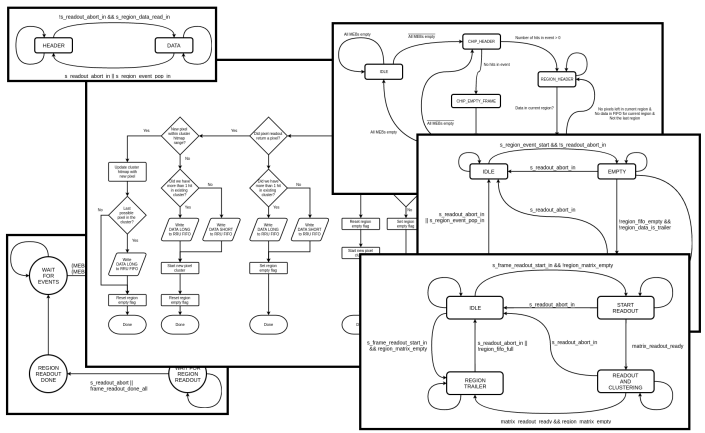


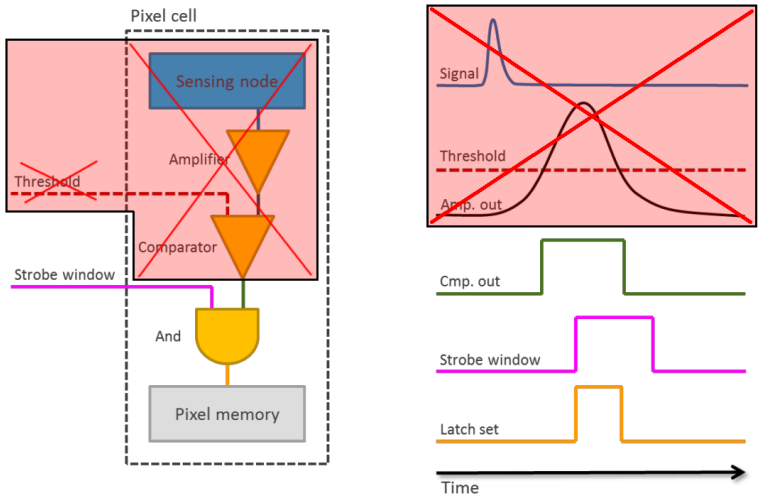
Figure: UML class diagram for Alpide module. SystemC modules in blue.

Alpide Dataflow Model

- Busy mechanisms in the ALPIDE chip is dominated by the readout logic
- The ALPIDE simulation model has an accurate implementation of the readout logic;
  - RRU, TRU, etc.



- Other aspects of the real chip were omitted, or modeled with a lesser degree of accuracy, for instance:
  - No analog pulse shaping
  - No encoding of the outgoing data stream
  - etc.



Alpide Dataflow Model - Using it

- There is a simple “unit test” in the repository, which demonstrates the bare minimum for using the model
- Unfortunately a little out of date - but the principles still apply
- Basic steps:
  - Basically, instantiate an Alpide object
  - Connect a clock signal
  - “Feed” hits to the chip (simple functions for setting pixels in the front end)
  - Send a trigger to the chip
  - Receive data on the output

Alpide Dataflow Model - Example

```
Alpide alpide("alpide",
              0,
              128,
              64,
              dtu_delay_cycles,
              enable_clustering,
              continuous_mode,
              matrix_readout_speed);

AlpideDataParser parser("parser");


// 25ns period, 0.5 duty cycle
sc_clock clock_40MHz("clock_40MHz", 25, 0.5, 2, true);
sc_signal<bool> strobe_n;
sc_signal<bool> chip_ready;

sc_signal<sc_uint<24> > alpide_serial_data;

alpide.s_system_clk_in(clock_40MHz);
alpide.s_strobe_n_in(strobe_n);
alpide.s_chip_ready_out(chip_ready);
alpide.s_serial_data_output(alpide_serial_data);

// Initialize SystemC stuff and connect signals to Alpide here
parser.s_serial_data_in(alpide_serial_data);
parser.s_clk_in(clock_40MHz);
```

Alpide Dataflow Model - Example cont'd

```
// Create an event frame object
   EventFrame e(time_now, time_now+1000, chip_id, event_id++);

   // Create 100 random hits
   for(int i = 0; i < 100; i++) {
     rand_x = rand_x_dist(rand_gen);
     rand_y = rand_y_dist(rand_gen);

     hit_vector.emplace_back(rand_x, rand_y, time_now, time_now+1000);

     e.addHit(hit_vector.back());
   }

   // Feed event frame to Alpide
   e.feedHitsToChip(alpide);

   // Strobe active
   strobe_n = false;

   // Start/run for x number of clock cycles
   sc_core::sc_start(10, sc_core::SC_US);

   // Strobe inactive
   strobe_n = true;

   // Alpide will send out data now
   sc_core::sc_start(10, sc_core::SC_US);
```

The end

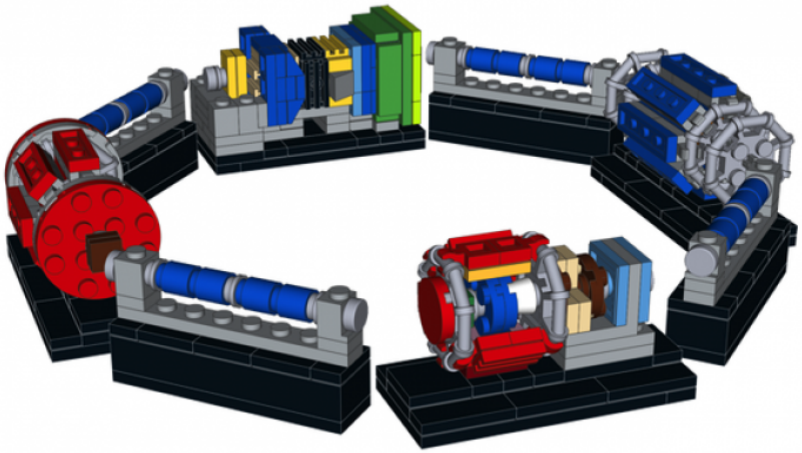


Figure: State of the art modeling tools have been used to design the upgraded LHC experiments [2].

References

[1] via Wikimedia Commons MissMJ [CC BY 3.0 (http://creativecommons.org/licenses/by/3.0) or Public domain].
Standard model of elementary particles, 2017.

[2] Nathan Readioff.
A computer simulation of the miniature lego lhc, complete with four detectors connected with blue dipole magnets, 2016.

[3] J. Schambach, M.J. Rossewij, K.M. Sielewicz, G. Aglieri Rinella, M. Bonora, J. Ferencei, P. Giubilato, and T. Vanat.
ALICE inner tracking system readout electronics prototype testing with the CERN “Giga Bit Transceiver”.
Journal of Instrumentation, 11(12):C12074–C12074, 2016.

[4] The ALICE Collaboration.
Alice-pho-ske-2017-001-5, 2017.
[Online; accessed August 6, 2017].

Backup slides

CERN

- CERN:
  - Conseil Européen pour la Recherche Nucléaire
  - European Organization for Nuclear Physics
- Location:
  - France/Switzerland, near Geneva
- Employees/staff members:
  - Around 2500
- Visiting scientists, engineers, fellows, etc.:
  - Around 12000

Figure: CERN member states

The Large Hadron Collider (LHC)

- The worlds largest machine and particle accelerator
- Circumference: 27 km
- Particles (hadrons):
  - Protons (p)
  - Lead nuclei (Pb)
  - Configurations: pp, p-Pb, Pb-Pb

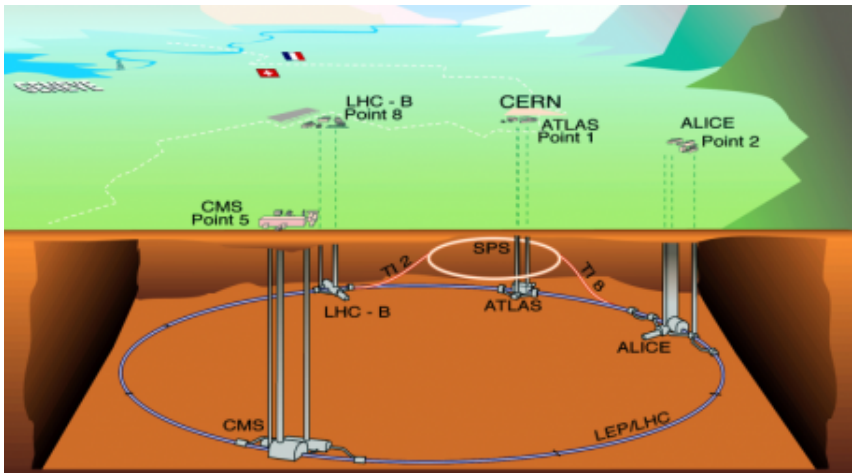
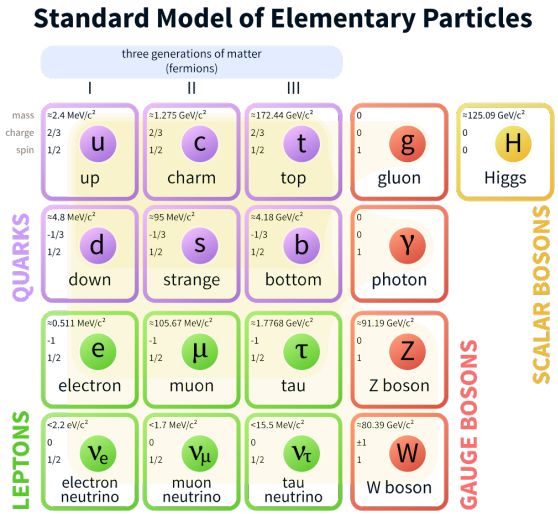


Figure: The Large Hadron Collider

- Particle beam energy: 6.5 TeV per nucleon
- At these energies, colliding hadrons will “break up” into their constituent particles
- Allows particles of the Standard Model to be studied



The Large Hadron Collider (LHC)

- “Bunches” of particles are accelerated in two “beam pipes”, in opposite directions
- The beam pipes cross at 4 “interaction points”
- Collisions between particles in the opposing beams occur at these interaction points
- Large experiments (detectors) located at each interaction point:
  - ATLAS
  - CMS
  - ALICE
  - LHCb

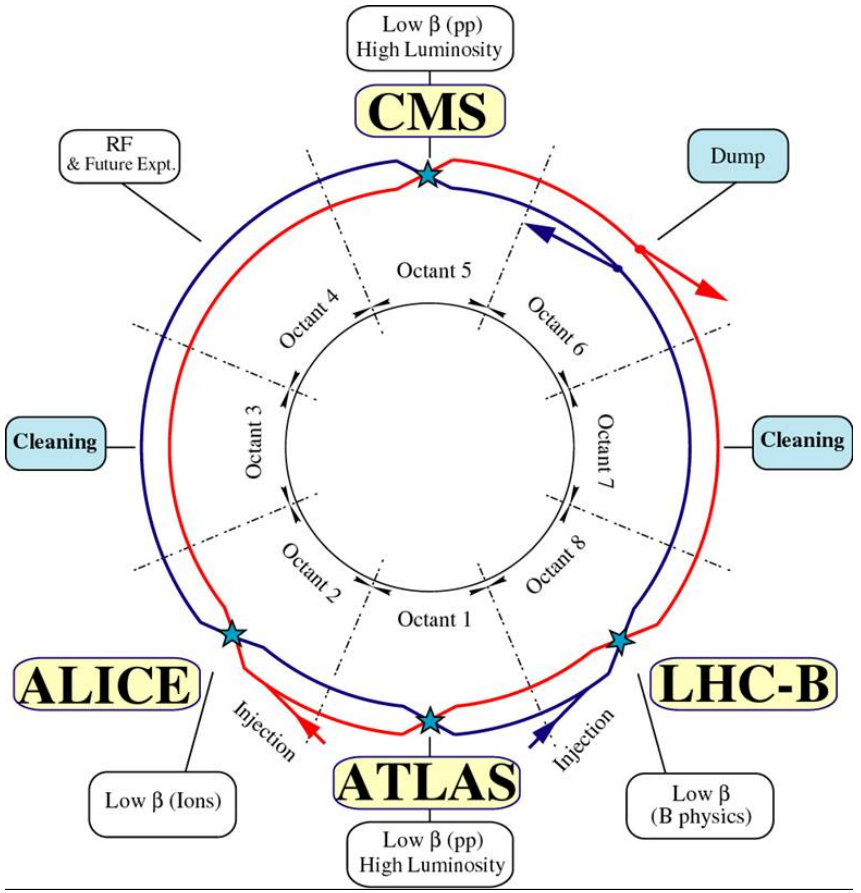


Figure: The two opposing beams at the LHC, in blue and red. Interaction (collision) points marked with stars.

ITS Readout Unit Challenges

- SRAM FPGAs are susceptible to radiation effects
  - Corruption of configuration memory is the main concern
- Flash FPGAs do not have such vulnerabilities (though sensitive to total ionizing dose)
- SRAM based FPGA still used as main FPGA due to:
  - Lack of high speed IOs and resources in flash based alternatives
- Flash FPGA in RU will work as a “backup FPGA” which reprograms the main FPGA in case of corruption

My project

- Design and review activities of Readout Unit
- Firmware design for the Auxiliary FPGA
- Design/implementation of FPGA firmware for trigger handling/distribution and handling of BUSY signals from ALPIDE chips, in the main FPGA
- SystemC simulation efforts for ALPIDE/RU dataflow and trigger/busy handling

Busy Simulations

Handling the busy signals from the ALPIDE chips in the RUs is a very complex problem:

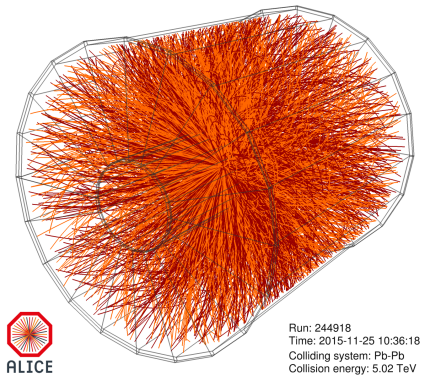
- The trigger signal is shared by all chips in a stave
- A dedicated Busy Module in the RU will decide if it should “hold back” triggers
- The RUs communicate busy status between them, so that they can take coordinated action

To understand the gravity of the problem, and in order to benchmark different busy handling topologies, a simulation model is being designed using SystemC.

SystemC Simulation Model - Event Gen.

Different options:

- Can use Monte Carlo events generated with the ITS upgrade aliRoot framework
- Could probably use real events as input without too much work
- Statistical event generator



Statistical event generator:

- Random multiplicity sampled from min-bias Pb-Pb multiplicity distribution
- Random inter-event times follow exponential distribution