CMS Pixel Telescope Analysis using the EUTelescope Framework.

Integration into the Generic Telescope Data Analysis Framework

Simon Spannagel
EUTelescope Workshop
DESY, 25.03.2013
Current pixel detector designed for peak luminosity of $10^{34}$ cm$^{-2}$s$^{-1}$

- 2012 peak lumi: $\sim 7.5 \times 10^{33}$ cm$^{-2}$s$^{-1}$, this will be exceeded after LHC Phase I
- Data loss due to buffer overflows at higher luminosity

- More data needs to be sent out
  - Number of optical fibers stays the same
  - Changing from analog to 400MHz digital readout

- Buffer extension on the readout chip
  - Smaller memory cells to fit sensor geom.
  - 80 data buffers, 24 time stamp buffers
  - (present: 32 data, 12 time stamps)
High-rate Beam Test for the PSI46dig ROC

- Qualification of the new CMS pixel chip: efficiencies, high-rate behavior...

- High-rate beam test in the CERN H4IRRAD with high energy protons from SPS at rates up to 250 MHz/cm²

- Operating the ROC as beam telescope with 8 single modules
  - Two geometries: planes perpendicular & planes tilted in xy, yz for proper Lorentz angle

- Telescope consists of ROC PCBs, readout board, and trigger scintillators
Why EUTelescope?

Software requirements for beam tests:

- Detector readout and data conversion/decoding, pulse height calibration
- Cluster and hit formation, alignment of telescope layers
- Track fitting, residual calculation and further analysis

Using existing software, customized to the particular needs:

ILCsoft framework

- common software framework for ILC R&D
- Provides Marlin for modular, event-based data processing
- LCIO data format, GEAR detector geometry description

EUTelescope framework

- Generic Pixel Telescope Data Analysis Framework
- Originally developed for the EUDET pixel telescope (EU FP6 project)
- Under active development and used by e.g. CMS Pixel and Strip Tracker, ATLAS FE-I4 / IBL test beams, DEPFET pixel telescope, Timepix, CLICPix, ...
- Provides processors with advanced algorithms for a variety of tasks
Telescope Data Processing Strategy with EUTelescope
Telescope Data Processing Strategy with EUTelescope

1. Detector Raw Data
2. Format Converter
3. LCIO Raw Data
4. Cluster Search
5. Clustered Data
6. Cluster Selection
7. Hit Candidates
8. Hit Maker
9. Hit Data
10. Alignment
11. Particle Tracks
12. Track Fitter
13. Alignment
14. GEAR File
15. ETA DB
16. Selection DB
17. Pedestal DB
Marlin processor types, EUTelescope data types

Marlin processor types:

- Processor
  - Expects to be fed with LCIO event to be processed
  - Returns after this one event has been processed

- DataSourceProcessor
  - Runs until all data processing is finished
  - Does not return to Marlin after processing one event
  - Needs to call Marlins Processor Manager from within when ready with one event

EUTelescope tracker data types

- Full matrix data:
  - Contains a full matrix of the detector size carrying the full non-zero suppressed data
    
    $\begin{array}{cccccccc}
    0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
    0 & 2 & 4 & 0 & 0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
    \end{array}$

- Sparse data:
  - Contains an arbitrary number of entries of the type (pixel_X, pixel_Y, pulseheight)
    
    $(1, 1, 2), (2, 1, 4), (6, 2, 1)$
The LCIO data format

➢ Run and even based data format:
  ▪ Events are collected in runs
  ▪ All information belonging to one trigger (i.e. event) is stored together

➢ Run header
  ▪ Information about the detector such as the number of planes, the number of pixels in x and y, the recording time of the run, etc.

➢ Event header
  ▪ Event number, run number, timestamp, 
  ▪ Event type:
    ▶ BORE: BeginOfRunEvent marks the first event of a run
    ▶ DE: DataEvent
    ▶ EORE: EndOfRunEvent marks the end of the run

➢ Collections
  ▪ Every event consists of collections containing data from either different detector parts or different processing steps

➢ Use the tool dumpevent to look into LCIO event data
Building the LCIO run header

> Check CMSPixelReader.cc for a simple example.

> Once per run one needs to construct the run header of the LCIO file:

```c++
auto_ptr<IMPL::LCRunHeaderImpl> lcHeader(new IMPL::LCRunHeaderImpl );
auto_ptr<EUTelRunHeaderImpl> head(new EUTelRunHeaderImpl (lcHeader.get()));
head->addProcessor(type());
```

> [Add data: setRunNumber, setDescription, setDataType, setDateTime, 
setNoOfDetector, setMinX, setMaxX, setMinY, setMaxY, setDetectorName ... ]

> ProcessorMgr::instance()->processRunHeader(static_cast<lcio::LCRunHeader*>(lcHeader.release()));
Decoding the raw data

Prepare the event infrastructure with the various information needed (see code for examples)

Write the data collection with the decoded raw data:

```cpp
TrackerDataImpl * sparse = new TrackerDataImpl();
EUTelSparseDataImpl<EUTelSimpleSparsePixel> sparseData(sparse);
for(/* all pixel hits */) {
    auto_ptr<EUTelSimpleSparsePixel> sparsePixel(new EUTelSimpleSparsePixel);
    sparsePixel->setXCoord(/* x coordinate */);
    sparsePixel->setYCoord(/* y coordinate */);
    sparsePixel->setSignal(/* pulseheight */);
    sparseData.addSparsePixel(sparsePixel.get());
}
sparseDataCollection->push_back( sparse );
```
Building the LCIO event

Building an LCIO event header for every event decoded from the raw data:

```cpp
EUTelEventImpl *event = new EUTelEventImpl();
LCTime *now = new LCTime(static_cast<EVENT::long64>(_timestamp));
ext->setDetectorName("YourDetector");
ext->setEventType(kDE); // kBORE, kEORE, kDE
ext->setRunNumber(_runNumber);
ext->setEventNumber(_eventNumber);
ext->setTimeStamp(now->timeStamp());
```

Add the previously constructed collection to that event (you can add several collections of course!)

```cpp
event->addCollection(sparseDataCollection,_sparseDataCollectionName);
```

The event needs to be passed to Marlin to get available to the other processors:

```cpp
ProcessorMgr::instance() -> processEvent(static_cast<LCEventImpl*>(event));
```
The CMSPixelTelescope Analysis Chain

CMS pixel processors:
- Data conversion into common LCIO
- Per-pixel pulse height calibration of the pixel hit information

EUTelescope processors/functionality:
- Clustering with Sparse algorithm
- Hit position determination using Center-of-Gravity method
- Hot pixel detection and removal
- Track based alignment of telescope layers using Millepede II (algorithm used by CMS experiment)
- Coordinate System transformations (local/global)
- Track finding and fitting using analytical tracks
- Residual calculation in the Device Under Test telescope plane
Summary and Prospects

- EUTelescope provides advanced algorithms for many tasks in testbeam data reconstruction and analysis

- User implementation: raw data converter, calibration

- More generic integration of CMS data into EUTelescope underway to merge CMS pixel data with events from the AIDA telescopes when used only as DUT

- Further development of the EUTelescope package to e.g. provide different track models and advanced tracking algorithms such as General Broken Lines

- Latest release “EUTelescope 0.8.1” just a week ago
  ...and more to come
Thanks.
**dumpevent output example**

```
== Event : 0 - run: 6692 - timestamp 90822639089 - weight 1 ==

date: 01.01.1970 00:01:30.822639089
detector: unknown
event parameters:
parameter EventType [int]: 2,
collection name: zsdata_m26
parameters:

------------- print out of TrackerData collection -------------

flag: 0x0
   LCIO::TRAWBIT_ID1 : 0
parameter CellIDEncoding [string]: sensorID:5,sparsePixelType:5,

<table>
<thead>
<tr>
<th>id</th>
<th>cellid0</th>
<th>cellid1</th>
<th>time</th>
<th>cellid-fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>[00000012]</td>
<td>00000032</td>
<td>00000000</td>
<td>0</td>
<td>sensorID:0,sparsePixelType:1chargeADC:676,382,1,</td>
</tr>
<tr>
<td>[00000013]</td>
<td>00000033</td>
<td>00000000</td>
<td>0</td>
<td>sensorID:1,sparsePixelType:1chargeADC:684,371,1,</td>
</tr>
<tr>
<td>[00000014]</td>
<td>00000034</td>
<td>00000000</td>
<td>0</td>
<td>sensorID:2,sparsePixelType:1chargeADC:692,379,1,434,1,</td>
</tr>
<tr>
<td>[00000015]</td>
<td>00000035</td>
<td>00000000</td>
<td>0</td>
<td>sensorID:3,sparsePixelType:1chargeADC:208,114,1,</td>
</tr>
<tr>
<td>[00000016]</td>
<td>00000036</td>
<td>00000000</td>
<td>0</td>
<td>sensorID:4,sparsePixelType:1chargeADC:195,0,1,206,</td>
</tr>
<tr>
<td>[00000017]</td>
<td>00000037</td>
<td>00000000</td>
<td>0</td>
<td>sensorID:5,sparsePixelType:1chargeADC:257,9,1,</td>
</tr>
</tbody>
</table>
```

---

Simon Spannagel | CMS Pixel Integration in EUTelescope | 05.03.2013 | Page 17