Bad component automatization

Tamás Álmos VÁMI[1]

[1] Wigner RCP, Budapest

Pixel Offline meeting

Goals

Automatize the bad component detection

Include the SiPixelQuality (bad component list) generation/upload in PCL

Reduce the bad component granularity to ROC level

Reduce the time granularity to lumisection level

For the Phase I detector this project is crucial (stuck-TBM, DCDC conv)

Bad component detection

Urs developed a code for Phase 0, I upgraded to Phase 1

The code computes an average ROC occupancy in detector

The following categories are defined:

• Dead:	ROC occupancy $< 10^{-3}$ ROC detector average
• Hot:	ROC occupancy $> 10\sigma$ from ROC detector average
• Inefficient:	ROC occupancy $< 10\sigma$ from ROC detector average

PCL integration

Suchandra (who developed bad component automatization for Strips) told me that we need to have a workflow which we have to pass to the central PCL people and they will take care of the PCL integration

What we have now:

Occupancy calculation → Bad ROC determination → SiPixelQuality payload → Upload to Prep

We can run on Condor/LXBATCH

TODO: Merge the separate processes + optimize for time

Validation

We have a way to test the results using DQM

In DQM from the bins there is a bad ROC list calculated:
e.g.: http://vocms061.cern.ch/event_display/Data2017/Beam/304/304906/HIZeroBias/DeadROC_offline.txt

Viktor made some modifications to the SiPixelQuality builder:
https://github.com/cms-analysis/DPGAnalysis-SiPixelTools/pull/14
Using which we can create a Quality from the list above.

To make the simulation with the new Quality faster I created a simulation up to RAW: /eos/user/t/tvami/BadComponentAtPCL/PerfectPhaseI_GenNu_13TeV_RAW.root

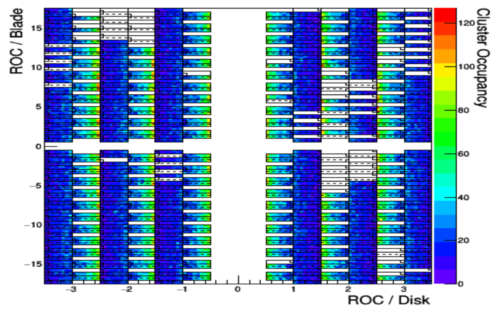
The SiPixelQuality payload is applied in the RAW2DIGI step:

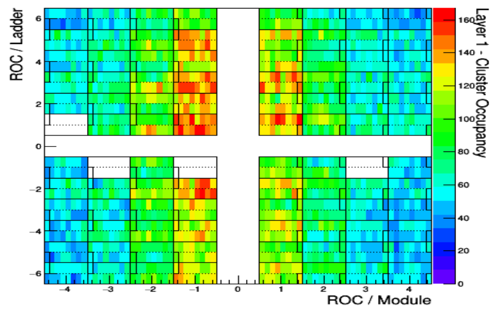
```
process.siPixelDigis.UseQualityInfo = cms.bool(True)
```

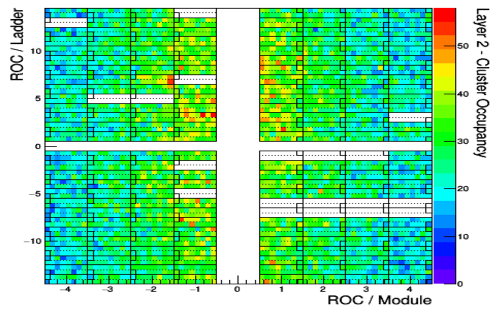
A comparison

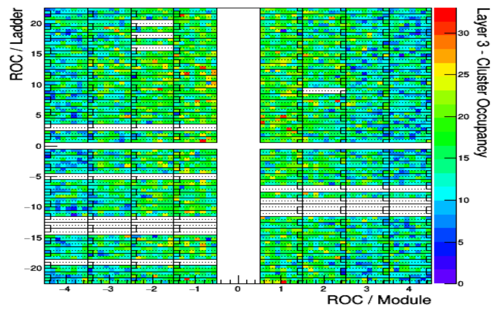
Only including the whole modules here

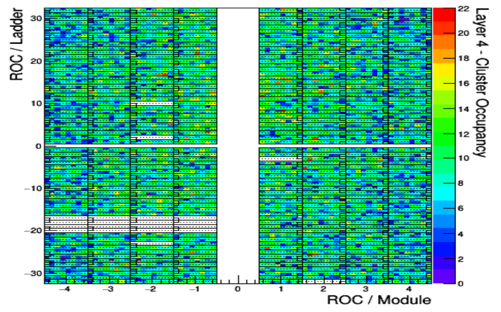
Own code -->





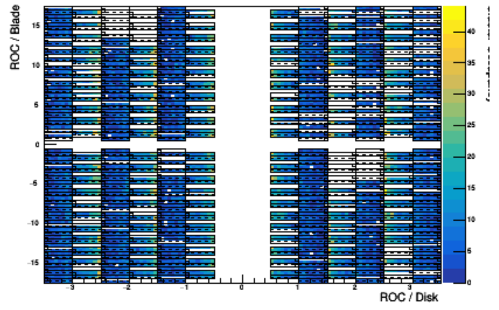


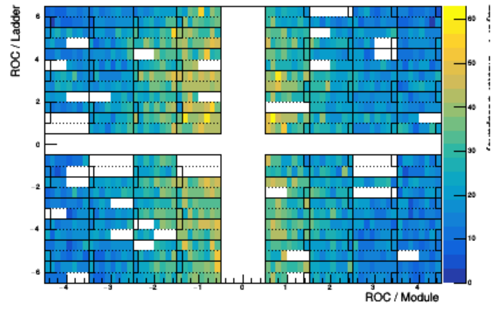


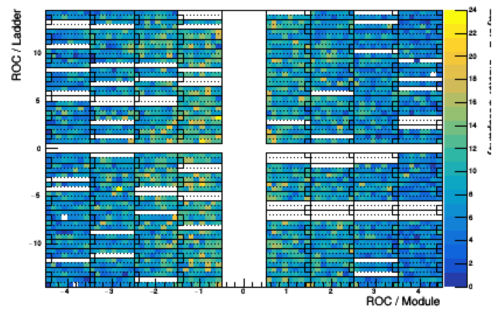


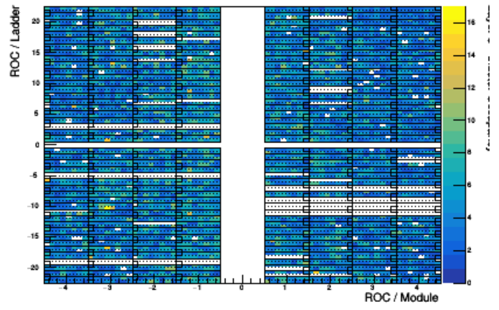
Run=305366

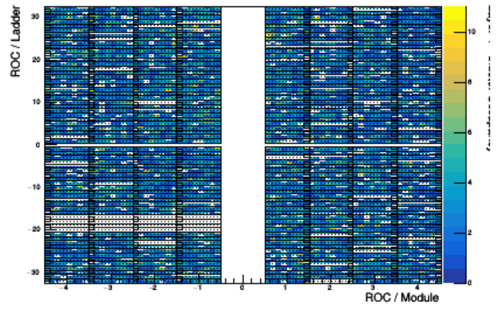
DQM list -->











Run=305366

Everything is ready to make the same thing on ROC level

Lumisection granularity

From Francesco I learnt that DQM already tested how low we can go

Below 8-10 LS there is not enough statistics

AlCaDB is OK with the 10-lumisection based SiPixelQuality

There was no development in the code until so far

I run on a specific number of files equivalent to 10 LS

Conclusion

Automatize the bad component detection	This works well
Include the SiPixelQuality in PCL	The workflow is ready but needs to be merged
Reduce to ROC granularity	This works well (optimization needed?)
Reduce LS granularity	10 LS is the goal. AlCaDB is OK with it. No code development yet

Tongguang Cheng volunteered to finish this project, so he is taking over it