Level-1 RPC Trigger in the CMS experiment - software for emulation and commissioning

Michał Bluj[1], Karol Buńkowski[2], **Tomasz Früboes**[2], Artur Kalinowski[2], Marcin Konecki[2], Michał Pietrusinski[2]

CMS Warsaw Group

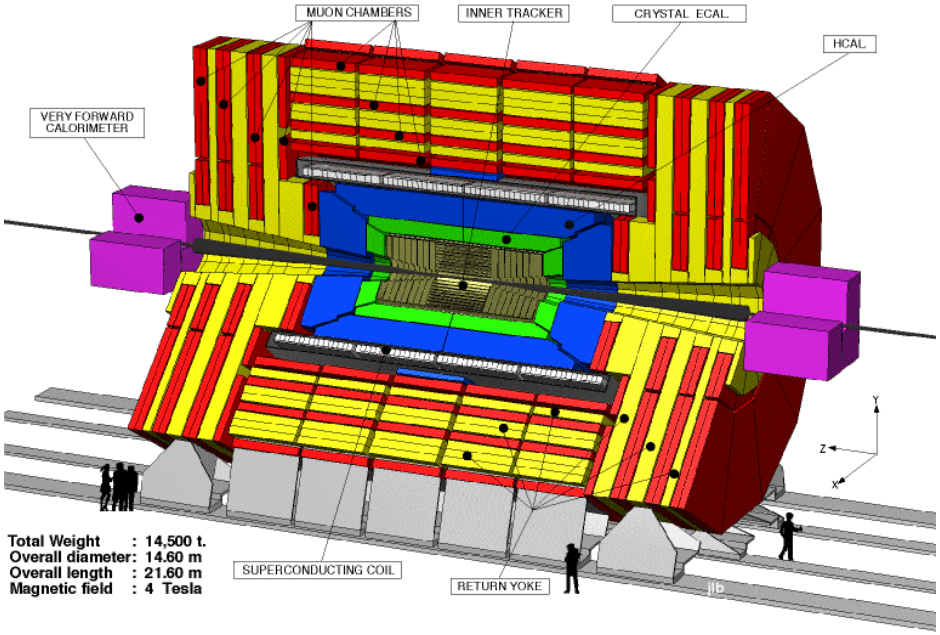
[1] Soltan Institute for Nuclear Studies - Warsaw

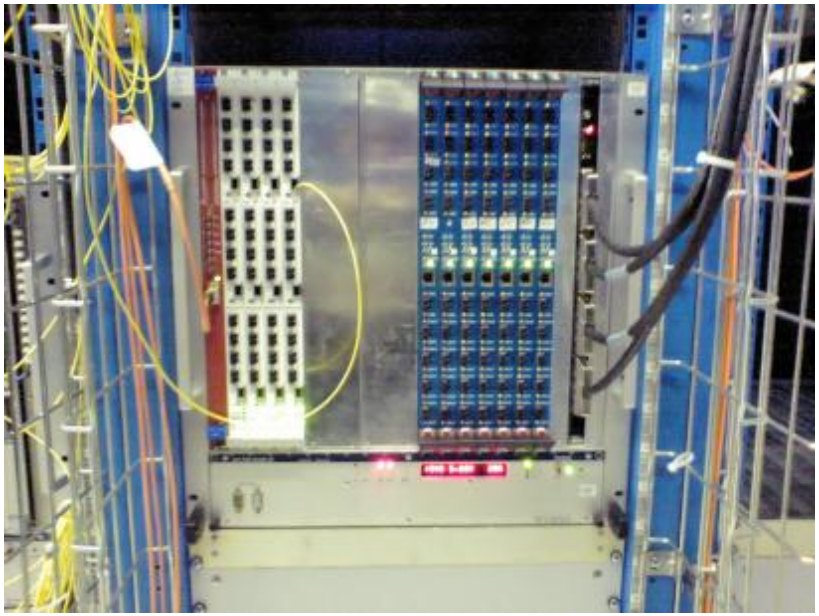
[2] Institute of Experimental Physics - University of Warsaw

September 6, 2007

Outline

Motivation - why test software and hardware?

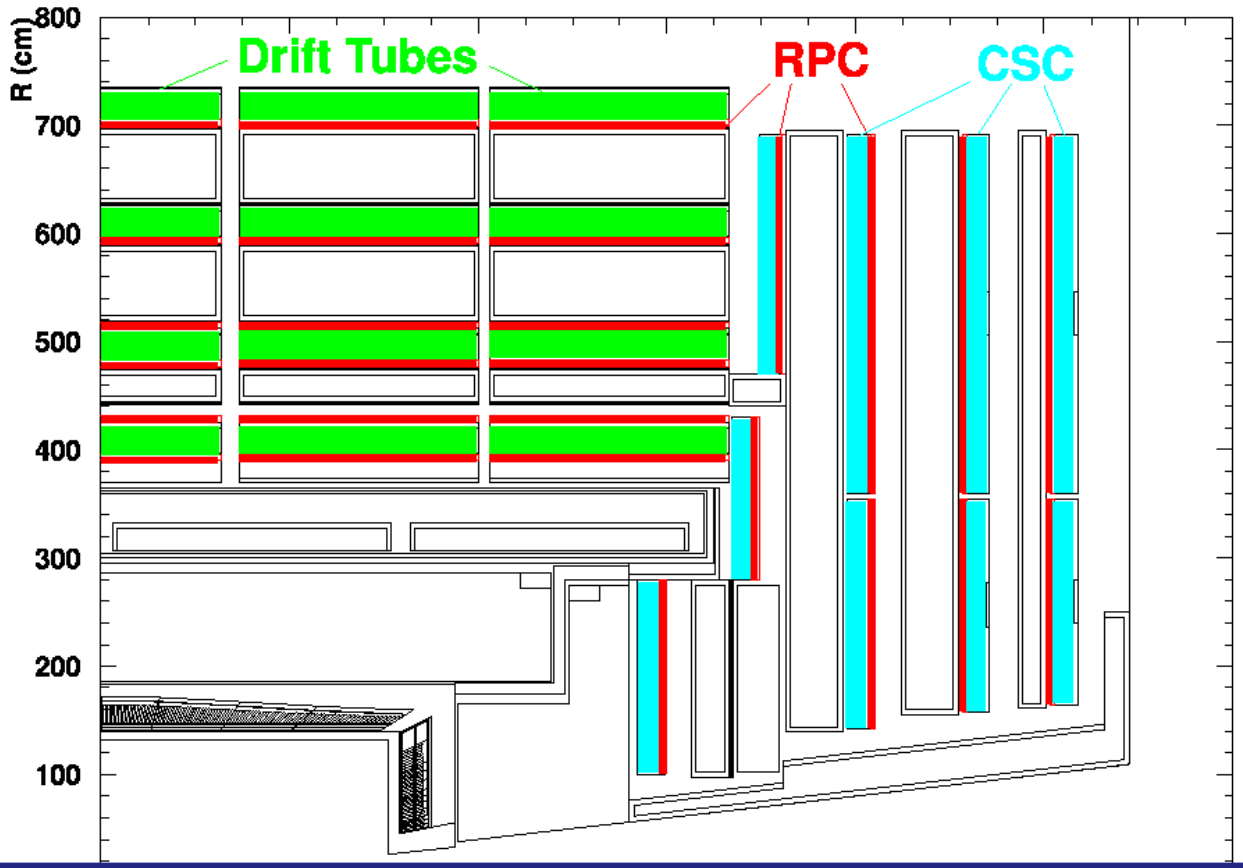




CMS and LHC starting soon...

- By this time hardware must be understood as good as possible
- Hardware implementation of algorithms must be checked
- Emulation software **has to** give the same output as hardware

$\rightarrow$ Comparing results of software and hardware is essential for debugging



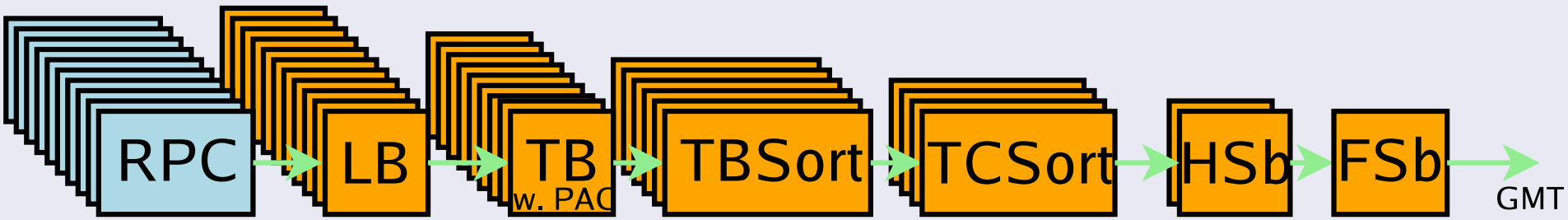
Overlapping muon systems

- DT (barrel) and CSC (endcaps). Duplicated by RPC (barrel & endcaps, dedicated for triggering)
- Three different muon triggers (DT-, CSC- and RPC triggers)

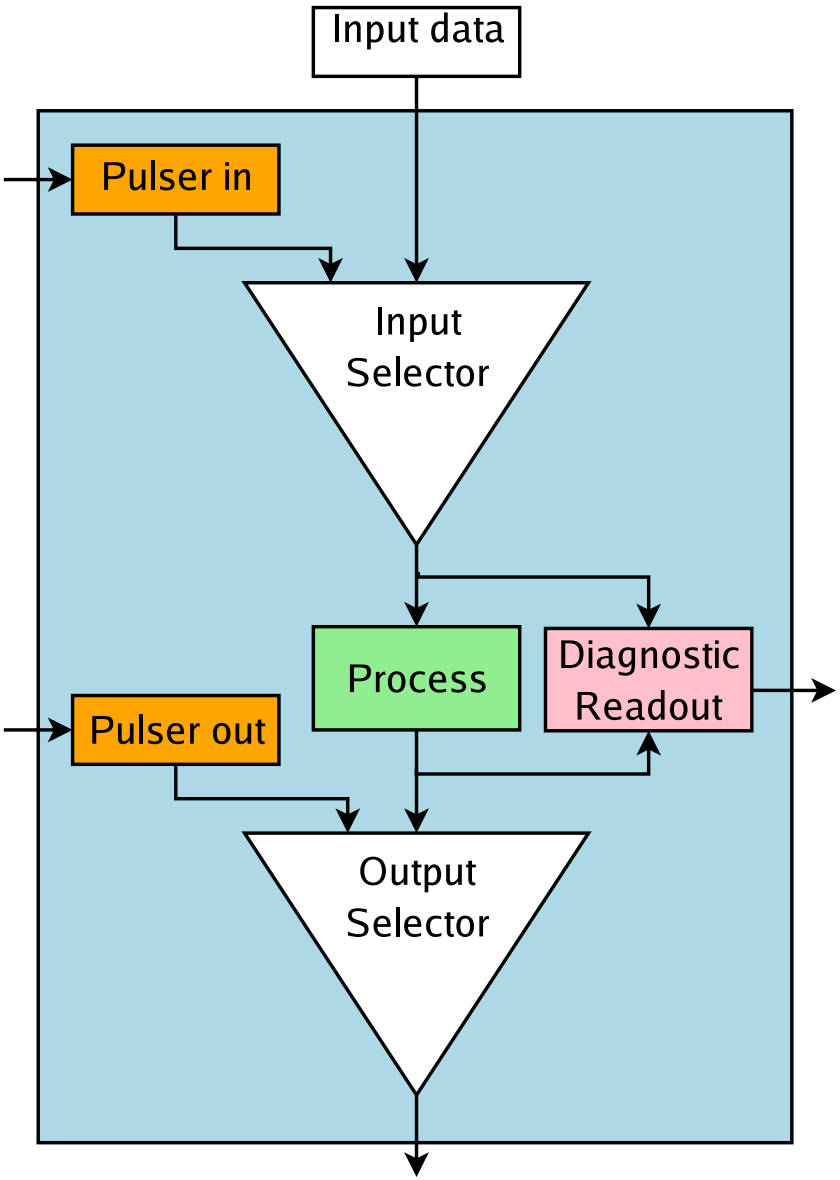
RPC Trigger - How it works

- Uses RPC chambers - located on both barrel and endcaps
- Muon Candidate = spatial and temporal coincidence of hits in RPC chambers
- Search performed by PAttern Comparator chip
- Each muon may be seen by several PAC chips - need of efficient ghostbusting and sorting algorithms
- Muon Candidates sent to Global Muon Trigger
- System implemented in FPGA

RPC Trigger - Data flow



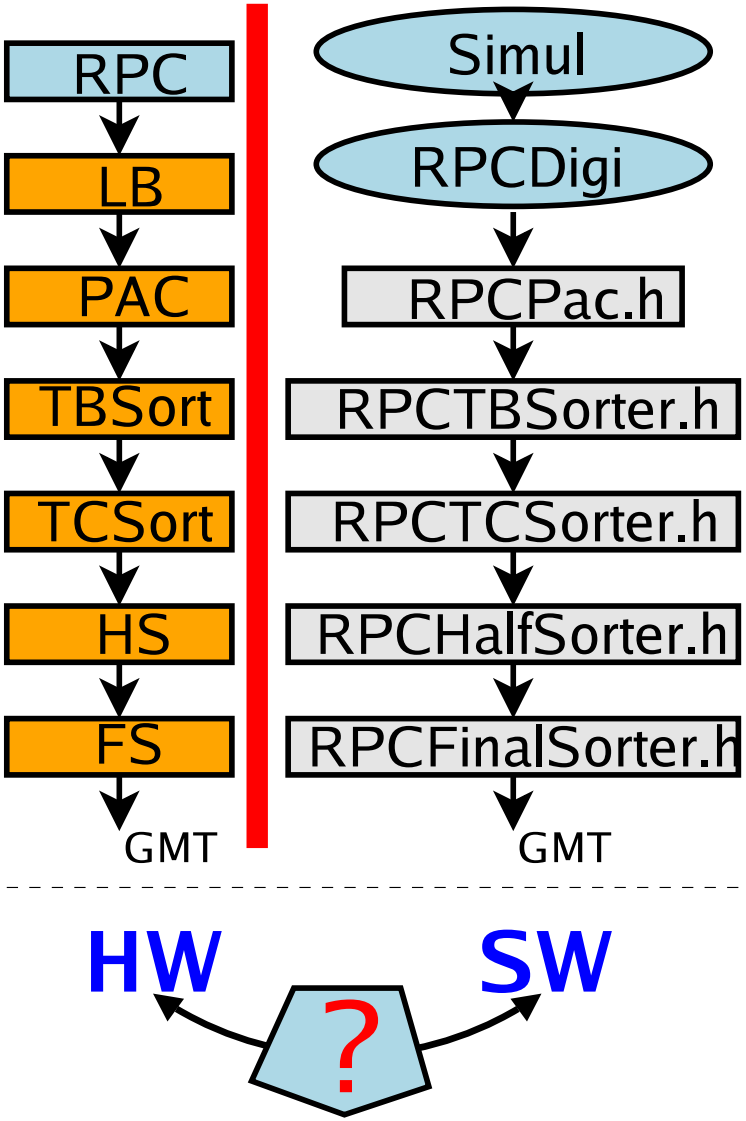
HW architecture for easy debugging



RPC

- Hardware split into different functional devices (PAC, sorters, ...)
- Functional block of the device surrounded by diagnostic layers:
  - Simulation layer - contains pulsers (in and out)
  - Monitoring layer - contains diag. readout
- Possible to inject artificial data and read the response at any stage of system

RPC Trigger emulation software



Emulator architecture

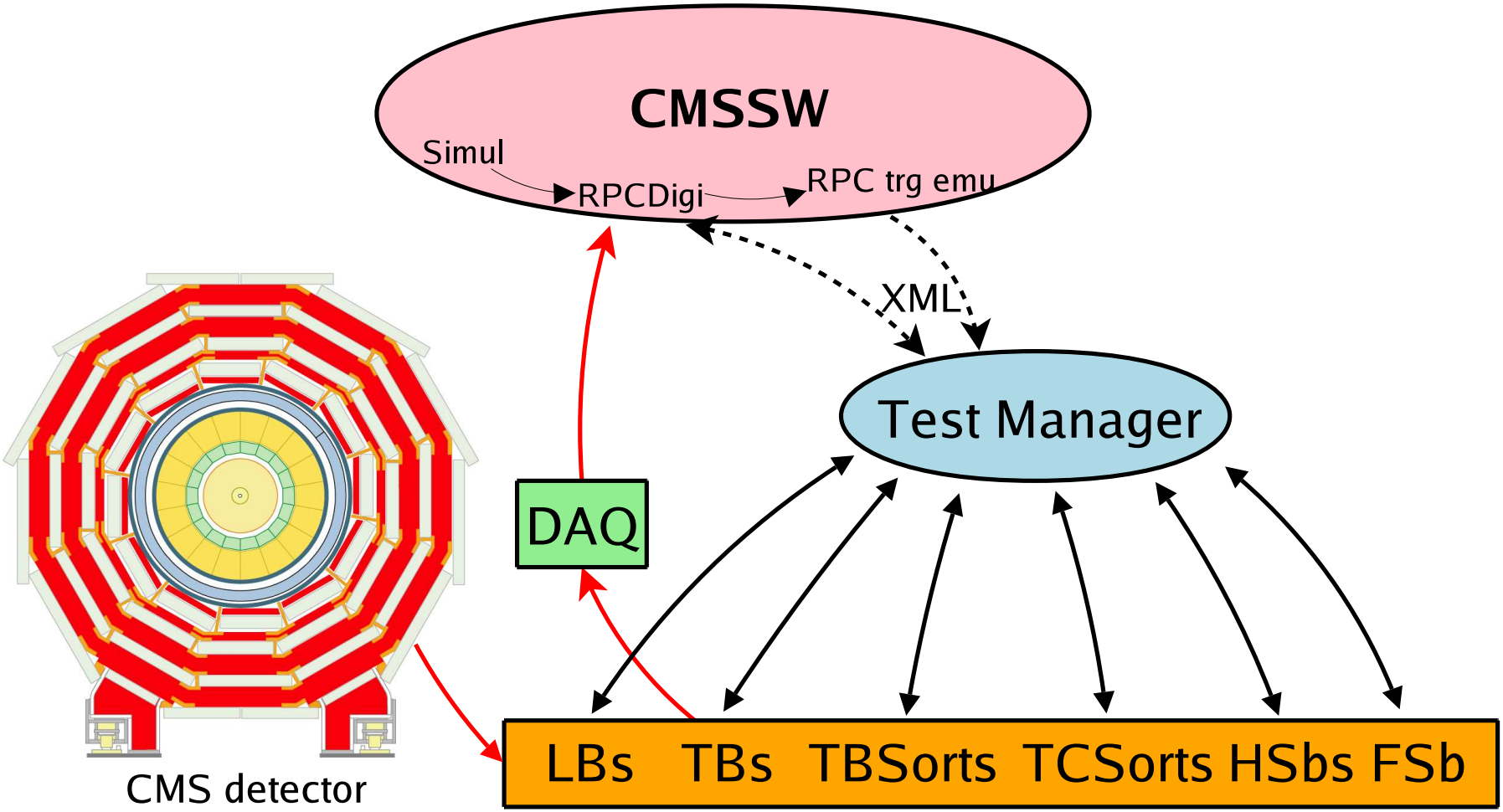
- Part of CMSSW (CMS SoftWare)
- Emulator design follows hardware design
  $\rightarrow$ Each main HW part has its SW counter-object (trigger board, PAC, ...)
- Internal data representation (and algorithms) same as in HW
- State of the emulator (set of muon candidates) easily dumped on any stage of its pipeline

Java software for tests

Java Test Manager

- Central manager written in Java for setup and control of the tests, communication with HW control applications (XDAQ) via SOAP, uses Hibernate framework to communicate with DB
- Uses pulsers and diagnostic readouts of HW
- Reads/stores pulse patterns/muon candidates in XML files:
  - Patterns for pulses are generated with CMSSW
  - Muon candidates from diagnostic readouts compared with emulator
- Allows insertion of fake data (i.e. no-muon-like). Important for connection tests between RPC Trigger and GMT
- Performs event building (data arrives with different times and mixed between bunch crossings)

Software architecture for tests



Test scenarios

Different test scenarios

1. Design/algorithm implementation test - check response of single HW parts (PAC, sorters, ...)
2. Interconnection test with fake data - transmission check to GMT
3. Cosmic test - trigger and record cosmic events

Tests results - implementation & interconnection tests

Implementation tests - small differences between SW & HW spotted and eliminated

- Internal data representation in emulator made more consistent with HW
- Sorting rules changed in emulation (minor, muon sign taken into account)
- Apart from that, no differences between SW & HW spotted
- HW behaves as expected. Software consistent with hardware

Interconnection tests

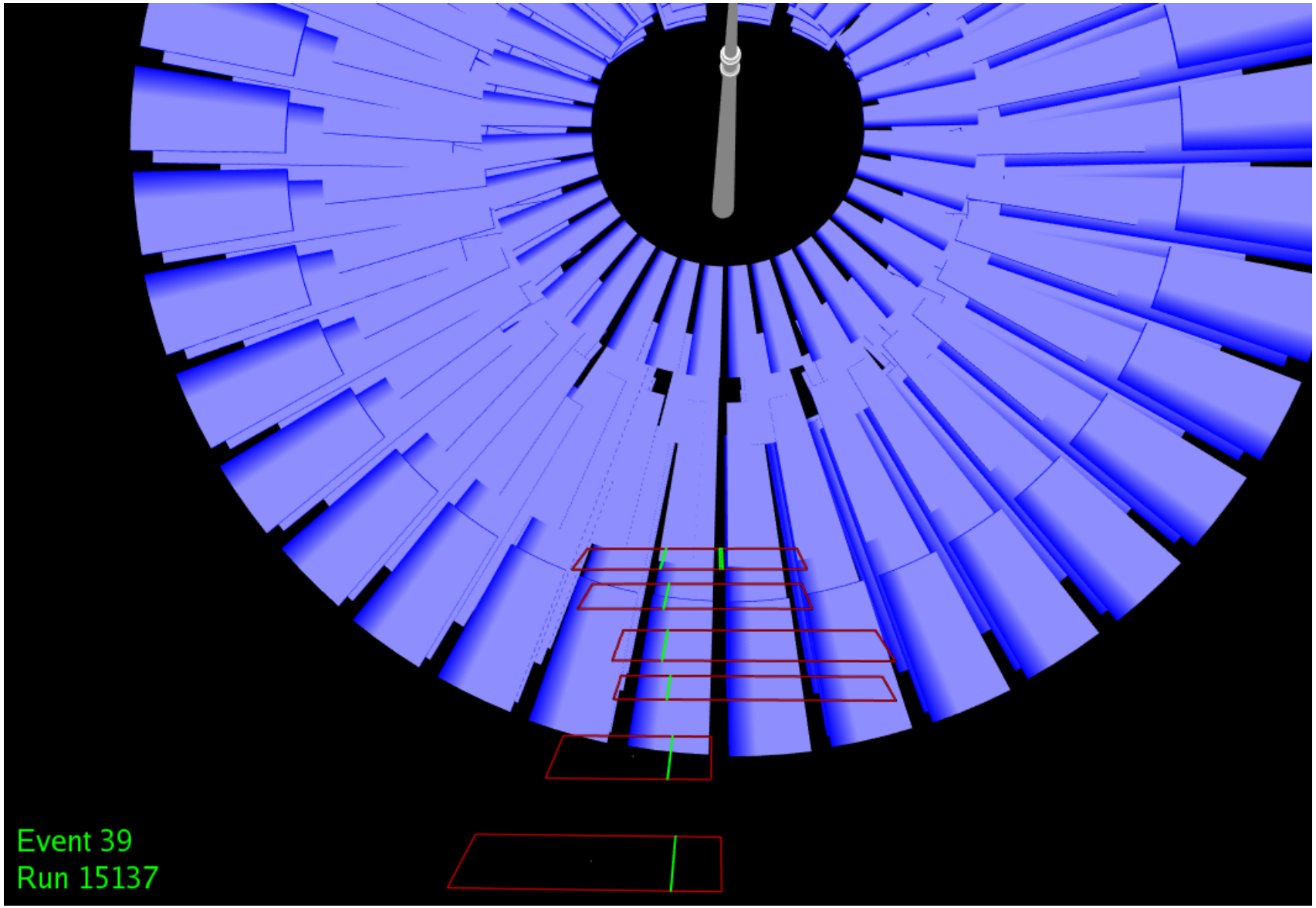
- Muon candidate is sent as a 32 bit number to GMT (each bit tested separately)
- During the tests one bit was found unstable. Fixed by setting different readout time window
- Stability test - sent $\sim 10^9$ events. Connection to GMT found stable

Test results - cosmic tests

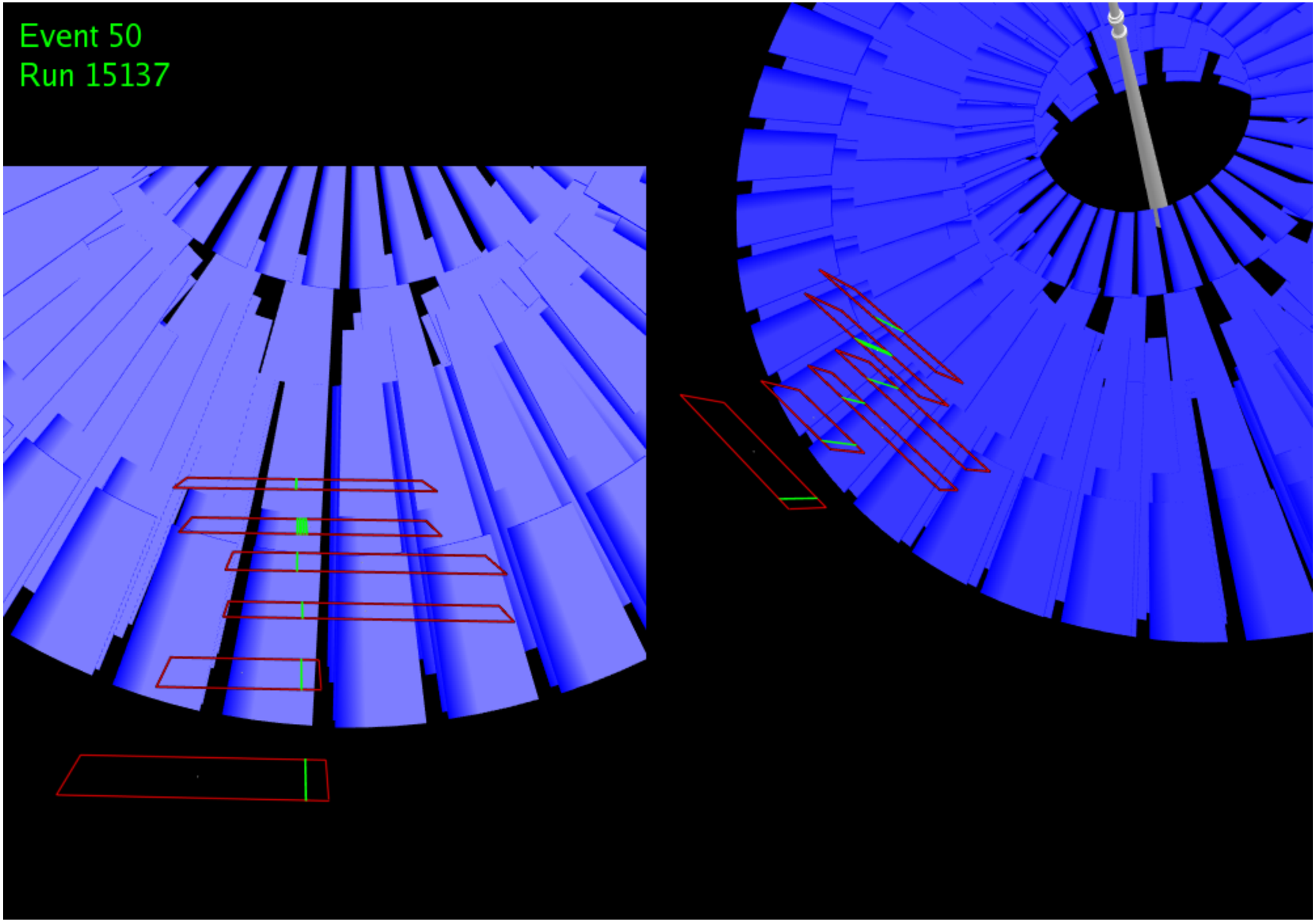
Results - cosmic muons recorded

- Cosmic muons with rate $\sim$2 Hz seen and recorded (rate as expected - detector in the underground cavern)
- PAC used as trigger source
- $\sim$Full chain (readout from chamber, data transmission to trigger, triggering, data recording in DAQ; sorters were skipped) was successfully tested. Everything OK
- $\sim$25 K events recorded
- Test software used simultaneously with standard DAQ. Data from diagnostic readouts will be used to verify connection to DAQ

Test results - cosmic muon



Test results - cosmic muon



Note: image showing same event from different views (only one muon visible)

Summary

Summary

- Hardware designed to allow easy testing of each component (injection of test data/diagnostic readout at every stage of pipeline)
- Emulation software follows HW as close as possible, determines (expected) state of system at every stage
- Minor differences between SW & HW eliminated
- Presented HW & SW architecture allows easy debugging