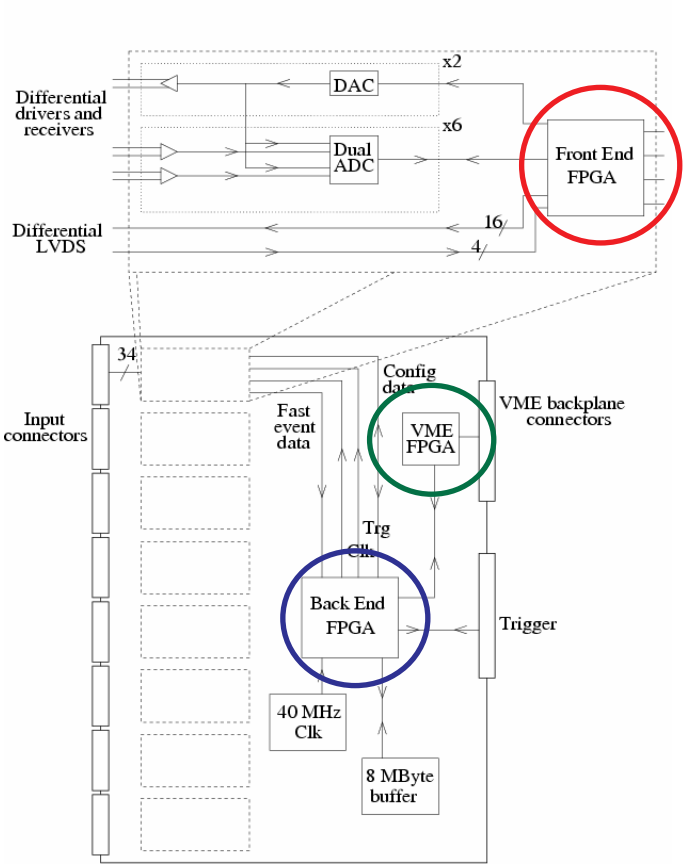
DAQ/Online Status

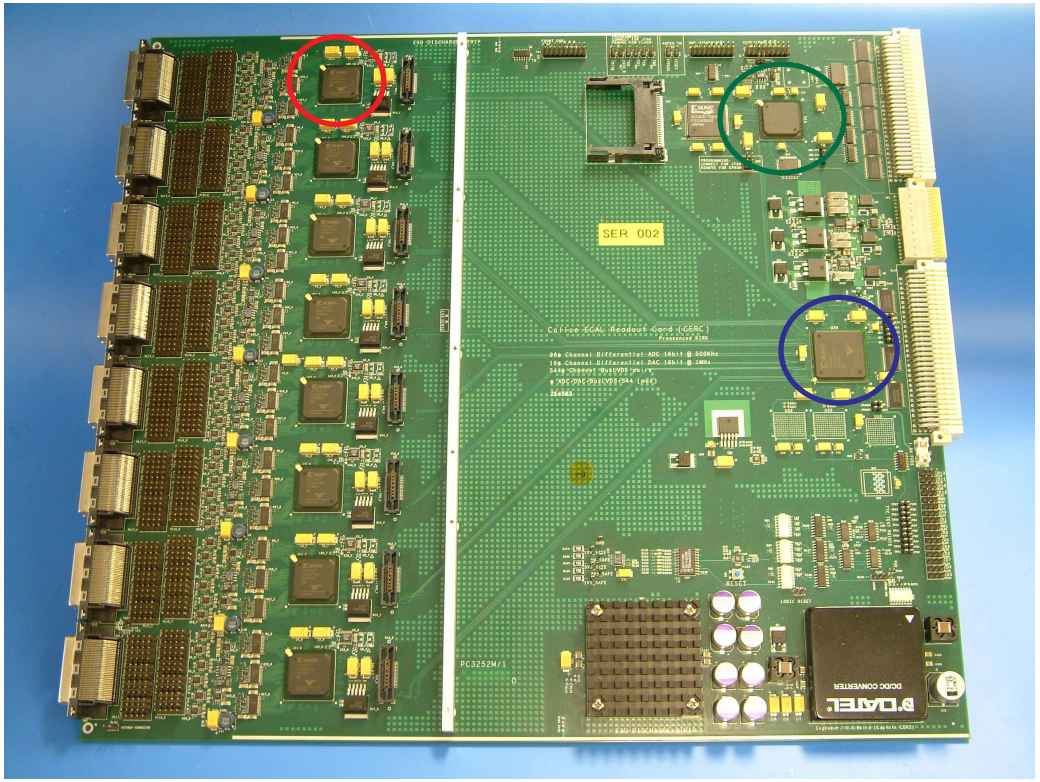
Paul Dauncey

Imperial College London

DAQ overview

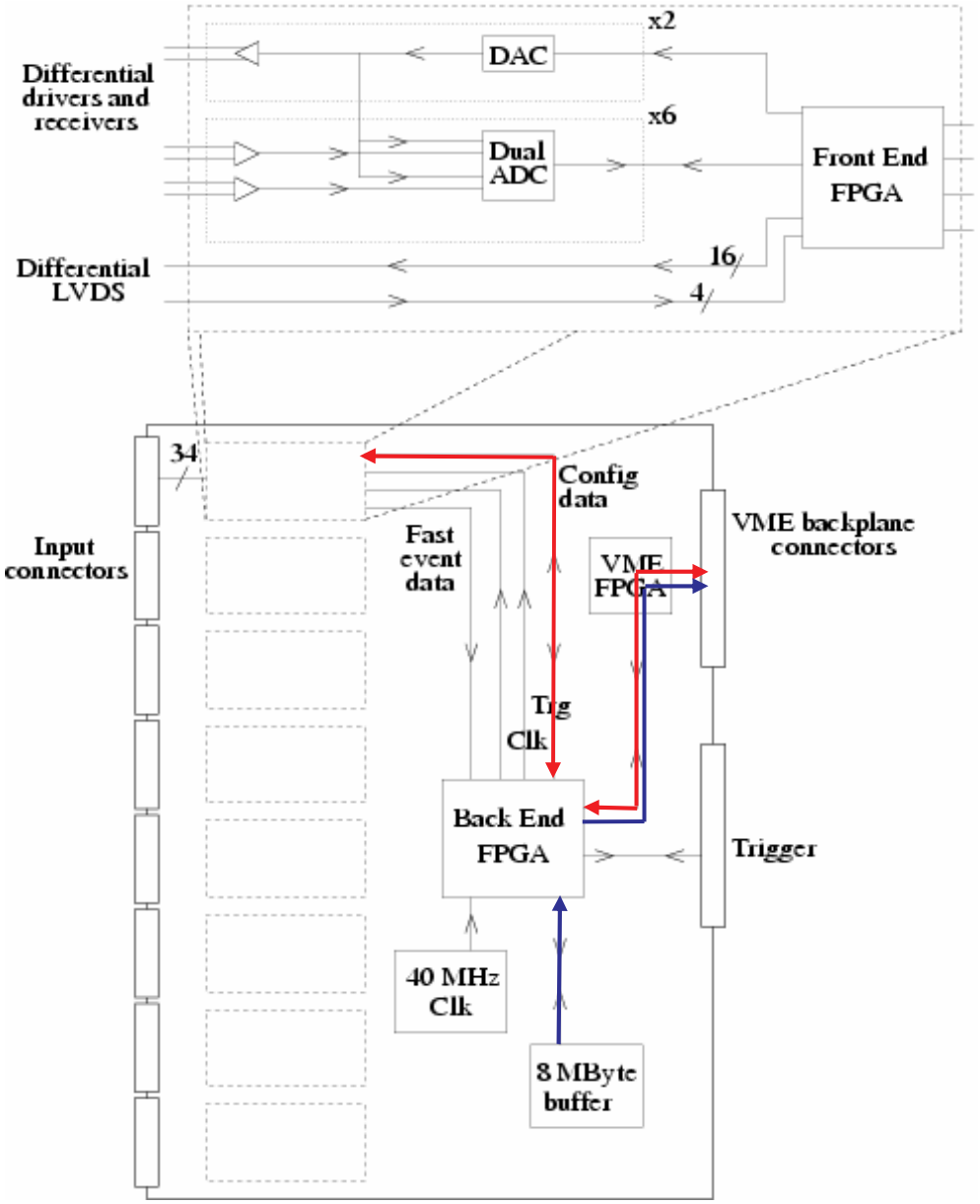
- Mainly based on Calice Readout Card (CRC) VME board
  - Modified from CMS silicon tracker readout board
  - Does very front end control, digitisation and data buffering



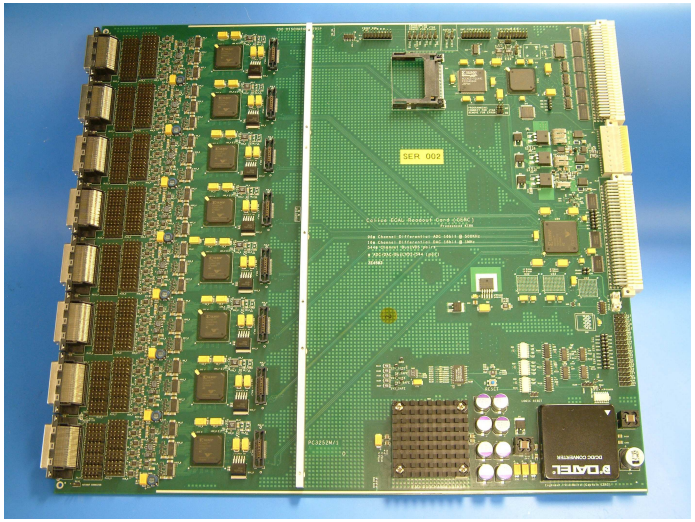


CRC readout

- Two VME data paths
  - Serial path: slow
    - Read and write directly to all FPGAs
    - Used for all configuration data loading and readback
    - Also used for temperature and power monitoring
  - Vlink path: fast (i.e. VME block transfer)
    - Read only from 8MByte memory (via BE)
    - Used for event data only



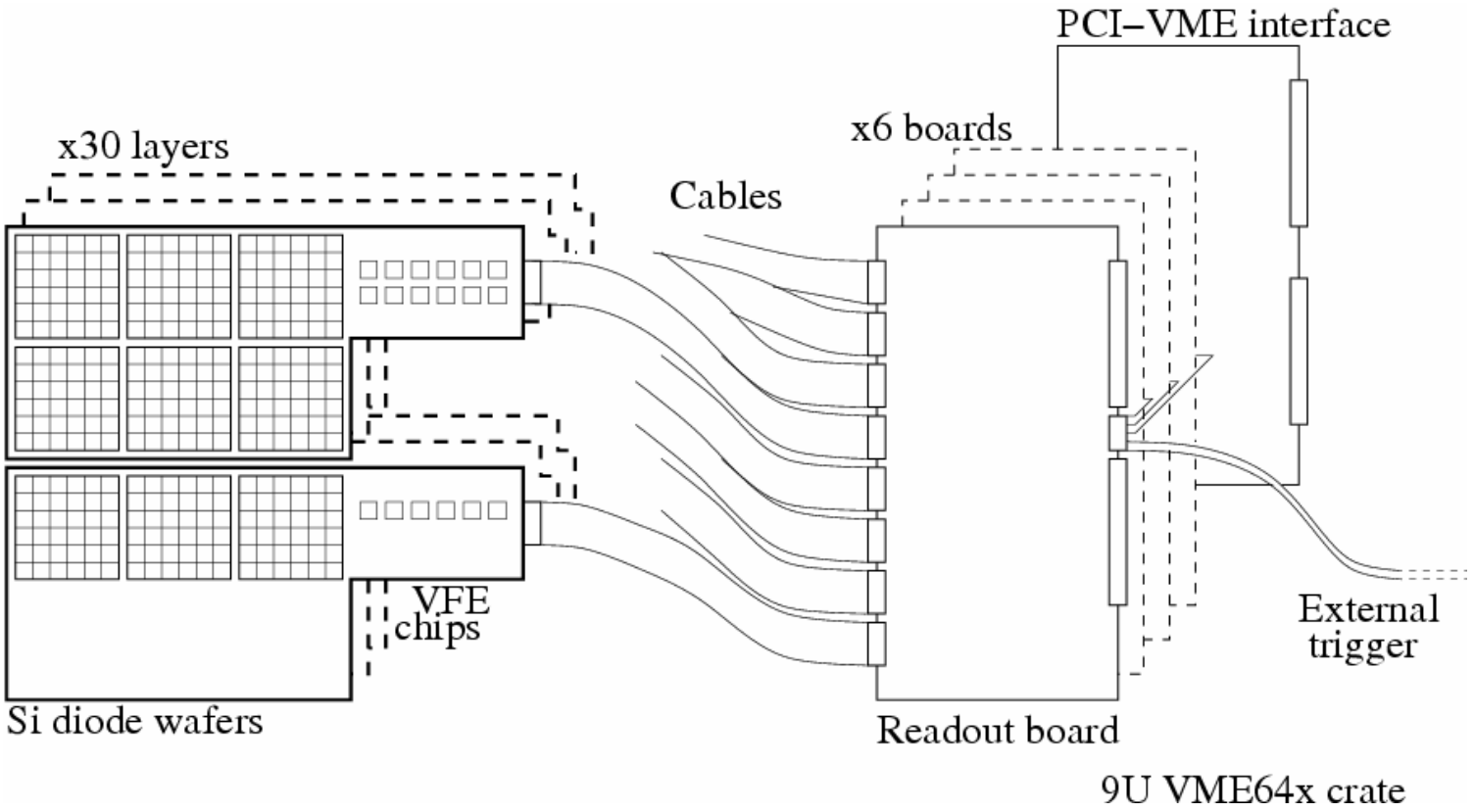
CRC hardware status



- Need 13 CRCs total
  - ECAL requires 6 CRCs
  - AHCAL requires 5 CRCs
  - Trigger (probably) requires 1 CRC
  - Tail catcher requires 1 CRC
- Status
  - 9 exist (2 preproduction, 7 production) and are tested
  - 7 are being manufactured via RAL, delivery in Nov
  - Should have 13 plus 3 spares by end of year
- DHCAL readout still very uncertain
  - Limited by lack of funding; cannot afford system already designed
  - May use CRCs to save money; would need 5 CRCs (like AHCAL) and so would use AHCAL ones, not make additional CRCs
  - No running with DHCAL planned before 2007; ignor for now

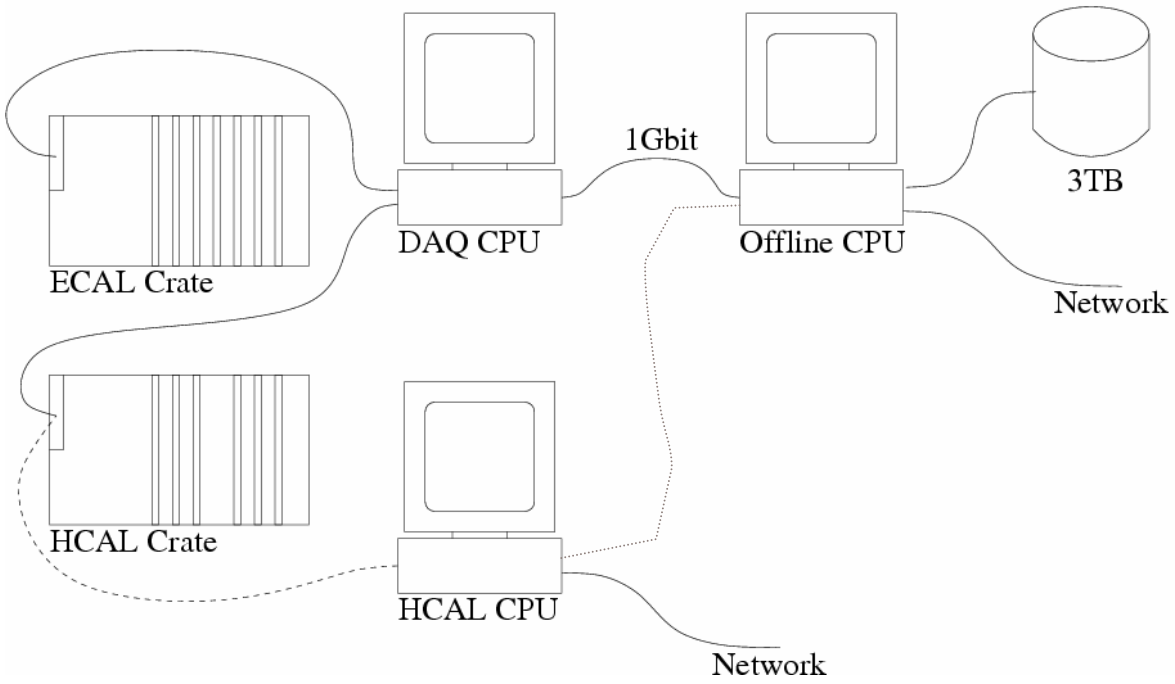
DAQ trigger distribution

- Trigger signal sent to one CRC
- Fanned out on custom backplane to other CRCs in ECAL
- AHCAL/Tail Catcher crate similar; cable connecting backplanes



DAQ hardware layout

- DAQ CPU
  - Trigger/spill handling
  - VME and slow access
  - Data formatting
  - Send data via dedicated link to offline CPU
  - Redundant copy to local disk?
- HCAL PC
  - Partitioning
  - Alternative route to offline PC
- Offline CPU
  - Write to disk array
  - Send to permanent storage
  - Online monitoring
  - Book-keeping

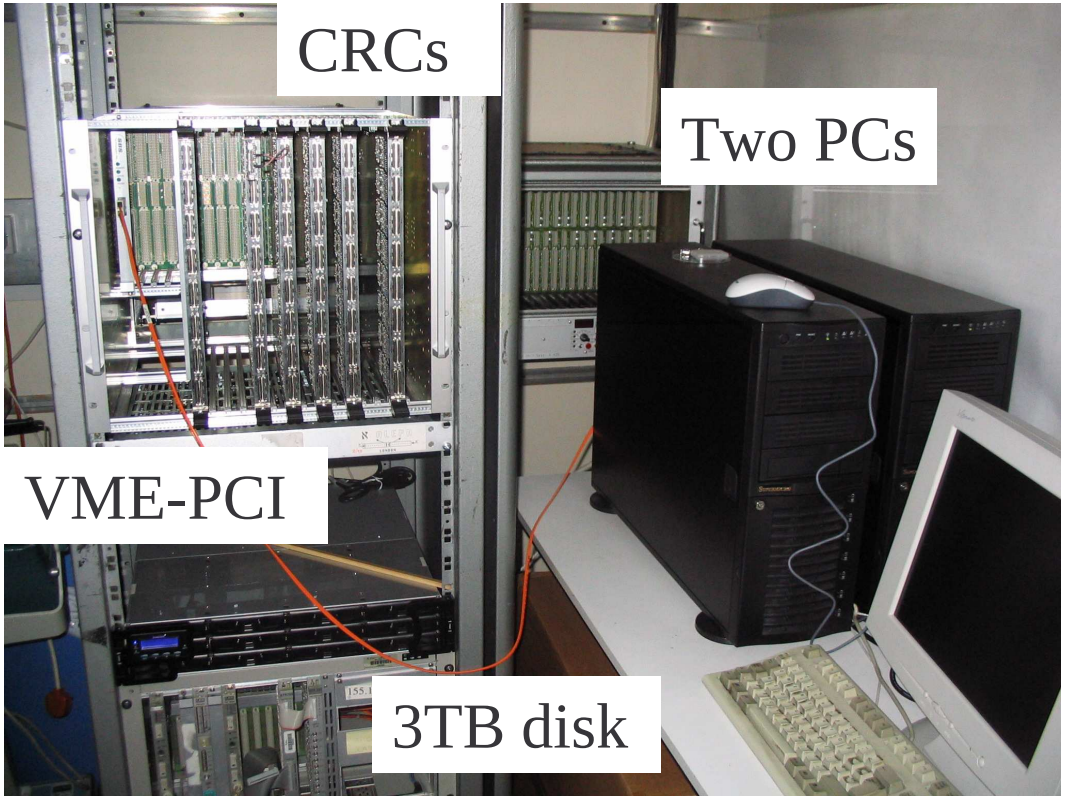


Status of non-CRC hardware

- Two 9U VME crates with custom backplanes needed
  - One for ECAL and trigger
  - One for AHCAL and tail catcher
  - Exist at DESY but no spares (for parallel testing, etc)

- Three VME-PCI bridges needed
  - All purchased and tested
- 100 mini-SCSI cables needed
  - Purchased 70 but not halogen free (needed at CERN)
  - May need to buy more
- Three PCs and disk
  - All purchased and tested

Test station at Imperial



Firmware status

- Three different FPGA firmware designs needed
  - VME: can use CMS version directly; no work needed
  - FE: completely new, but effectively finished
  - BE: two parts to this
    - "Standard" BE: data handling on all CRCs
    - "Trigger" BE: specific for CRC being used for trigger control
- Standard BE firmware is critical path; not complete
  - Can only buffer up to 500 events, but need 2000
  - Can only buffer in 2MBytes of memory, but need 8MBytes
  - Without both of these, data rate will be reduced by factor of four
- Trigger BE firmware needs work also
  - Trigger data (including detection of multi-particle events) can only be read via slow serial path: limits rate to ~20Hz (c.f. 1kHz, not 100Hz)
  - Need to route trigger data into 8MByte memory so can read via fast Vlink
  - Fallback is not to read these data

Slow controls/readout status

- Various slow controls and readout data are collected by DAQ
- CRC slow data
  - Temperatures: 22 different probes over surface of board
  - Power: 5 voltage level measurements of backplane inputs
  - Read out standardly during run: no work needed
- ECAL power and temperatures
  - Plan to read out via stand-alone PC (not yet existing)
  - Will need to interface to DAQ when it appears
- ECAL stage position
  - Stage controlled by stand-alone PC
  - Readout interface to DAQ tested and working
- AHCAL slow data and stage position
  - All centralised in stand-alone PC (running H1 slow control program)
  - Readout and control interface to DAQ tested; needs further work to be complete

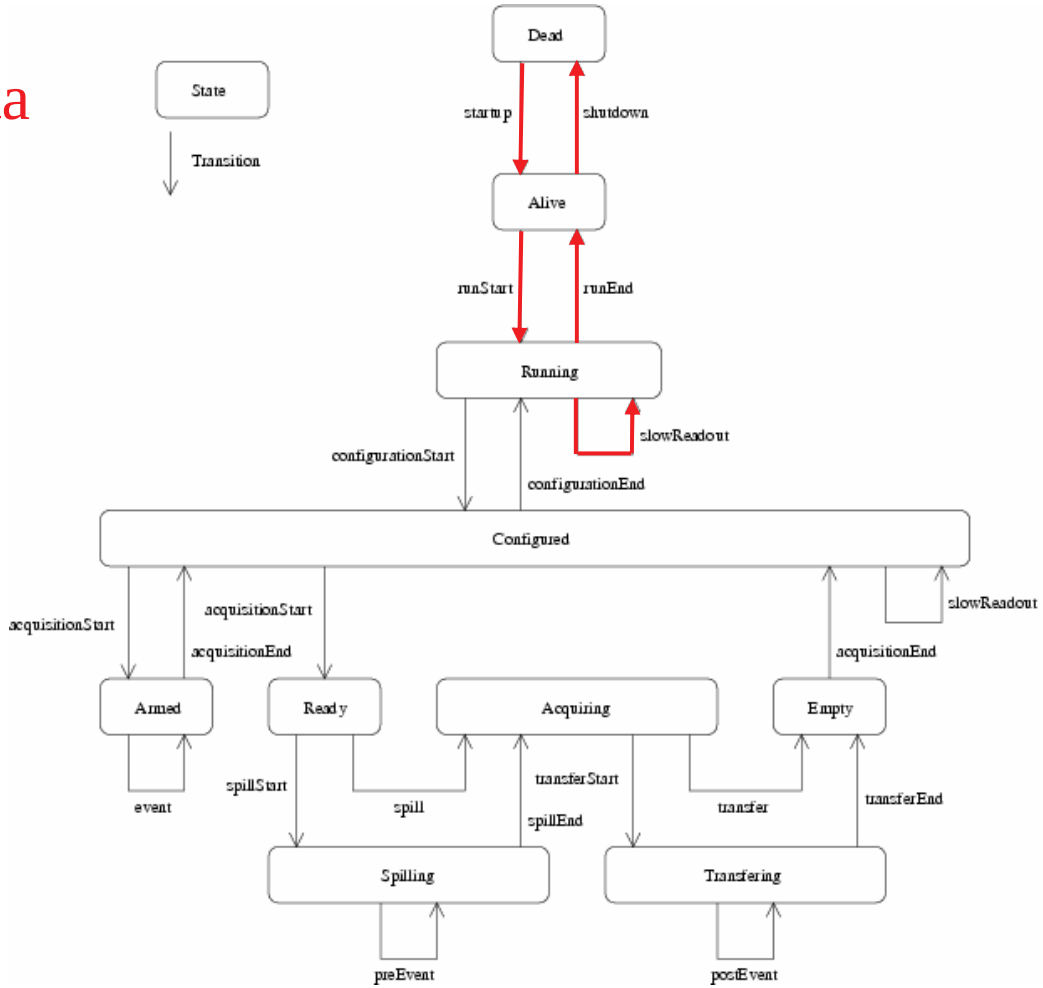
DAQ software status

- DAQ online software is based on a state machine
  - States have well-defined status where system is unchanging
    - CRCs configured, buffers full, etc.
  - Transitions between states cause changes to system
    - Download configuration data, take an event, etc.
  - DAQ pushes hardware round state machine by sending transition indicators
    - “Records”
- Many changes since version used at DESY
  - Firmware changes
  - Change to use of LCIO rather than raw data for analysis
  - Experience from DESY run
- Major rewrite currently in progress
  - Biggest task at present

DAQ state machine

- Nested structure: arbitrary number of loops at each level
- Three main types of run

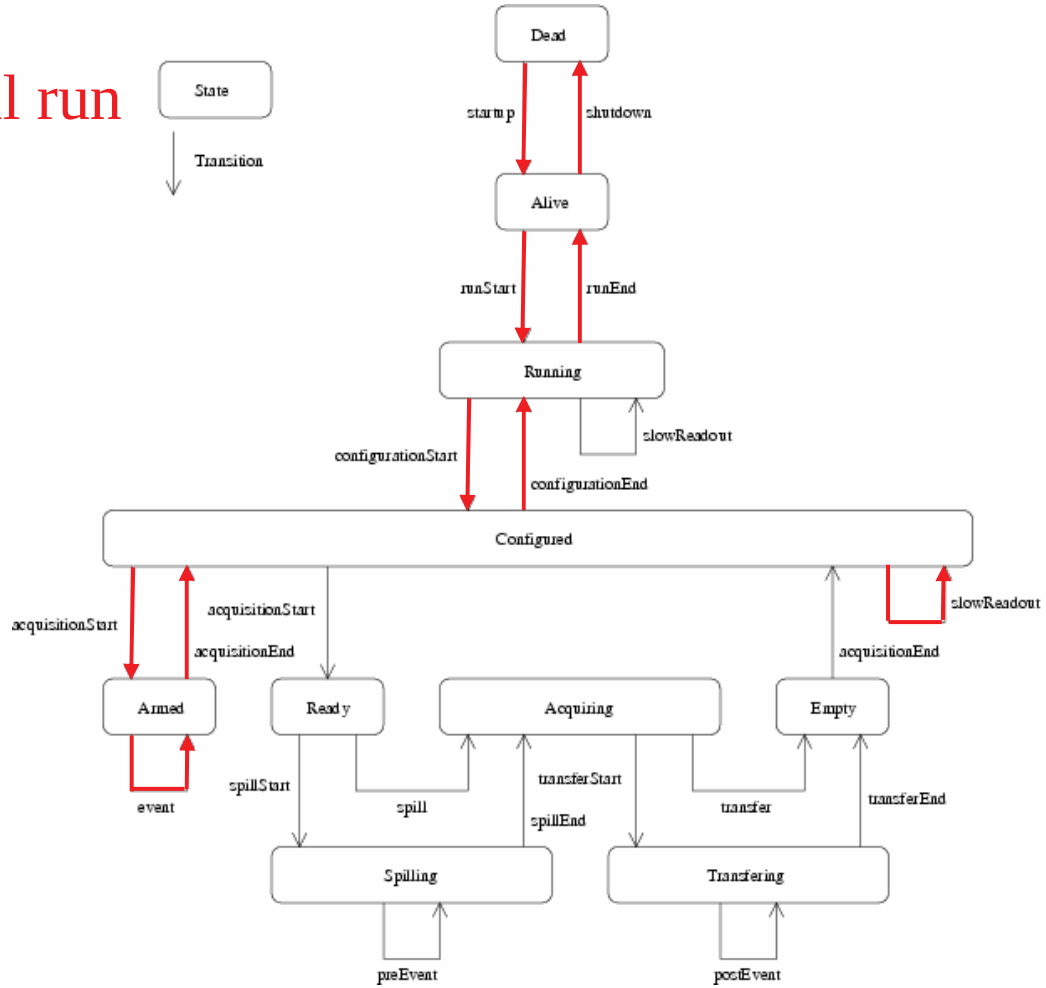
Slow data



DAQ state machine

- Nested structure: arbitrary number of loops at each level
- Three main types of run

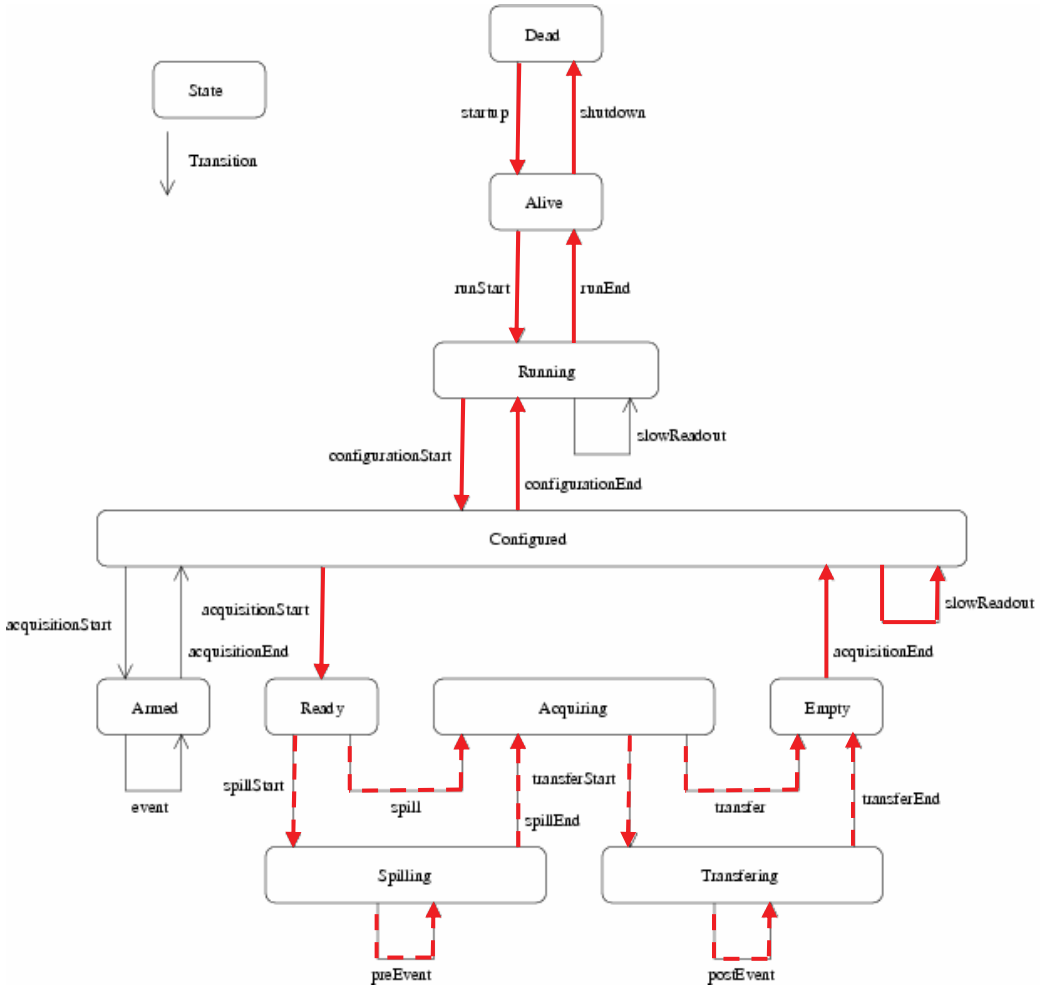
Non-spill run



DAQ state machine

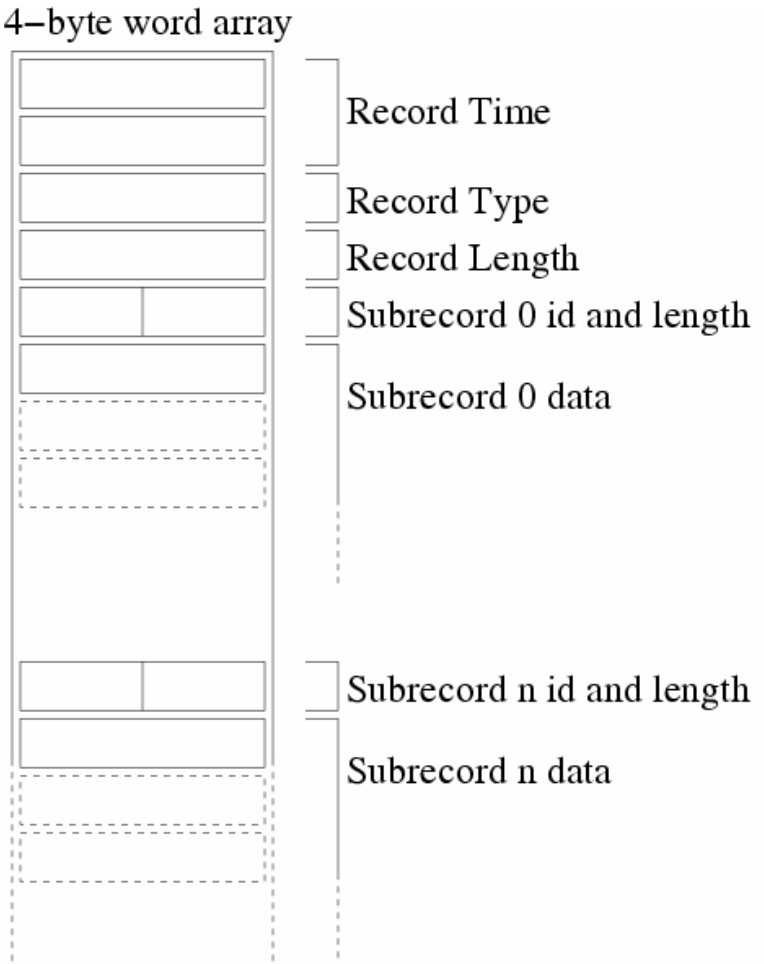
- Nested structure: arbitrary number of loops at each level
- Three main types of run

Spill run



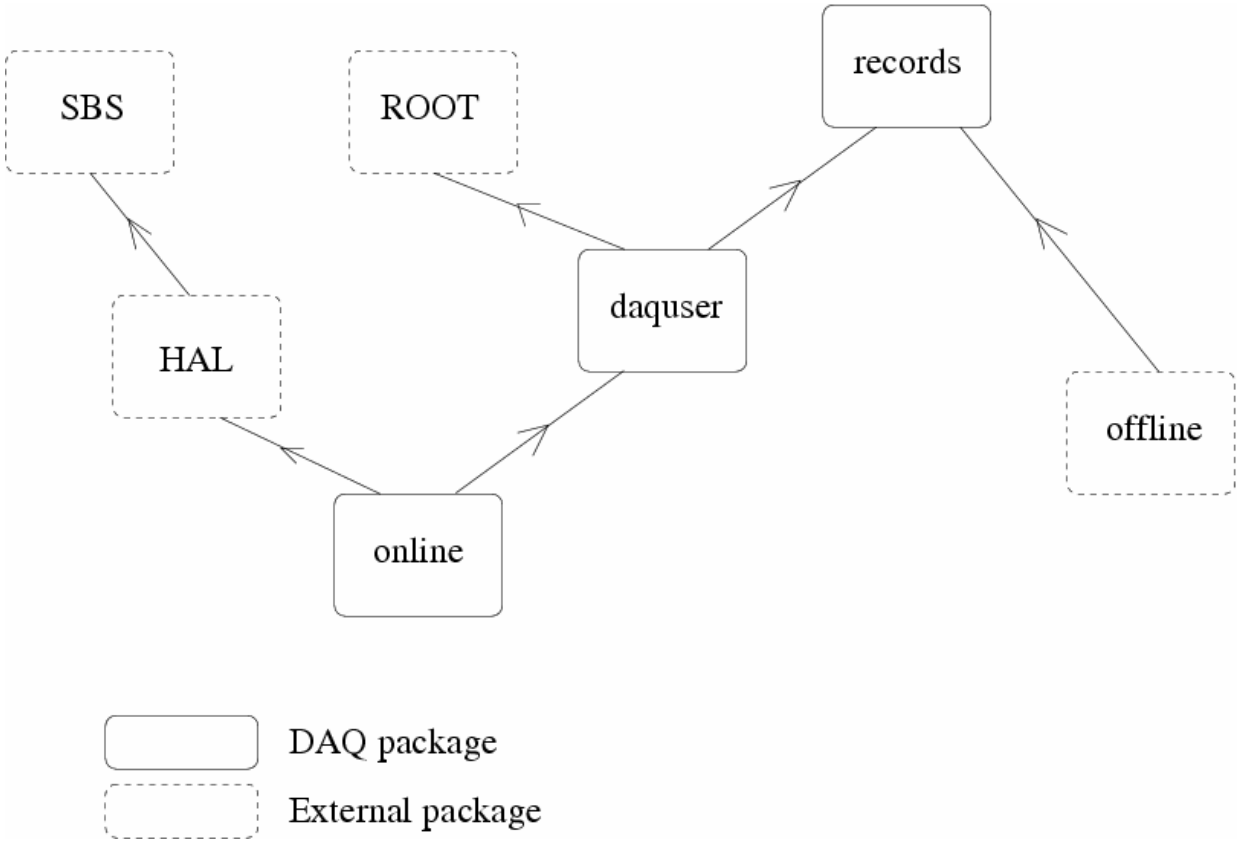
Records and subrecords

- Basic unit of online is a record: two uses
  - Data storage for transport of both upstream and downstream data
  - State machine transition indicator
- DAQ works by pushing records through system to cause transitions
  - Records contain data needed for transition
  - Any data generated by transition is appended to record
  - Final complete record is the raw data and is written to file
- Internal record data structure organised into subrecord objects
  - Map directly to C++ classes
  - Functions provided to access subrecord objects



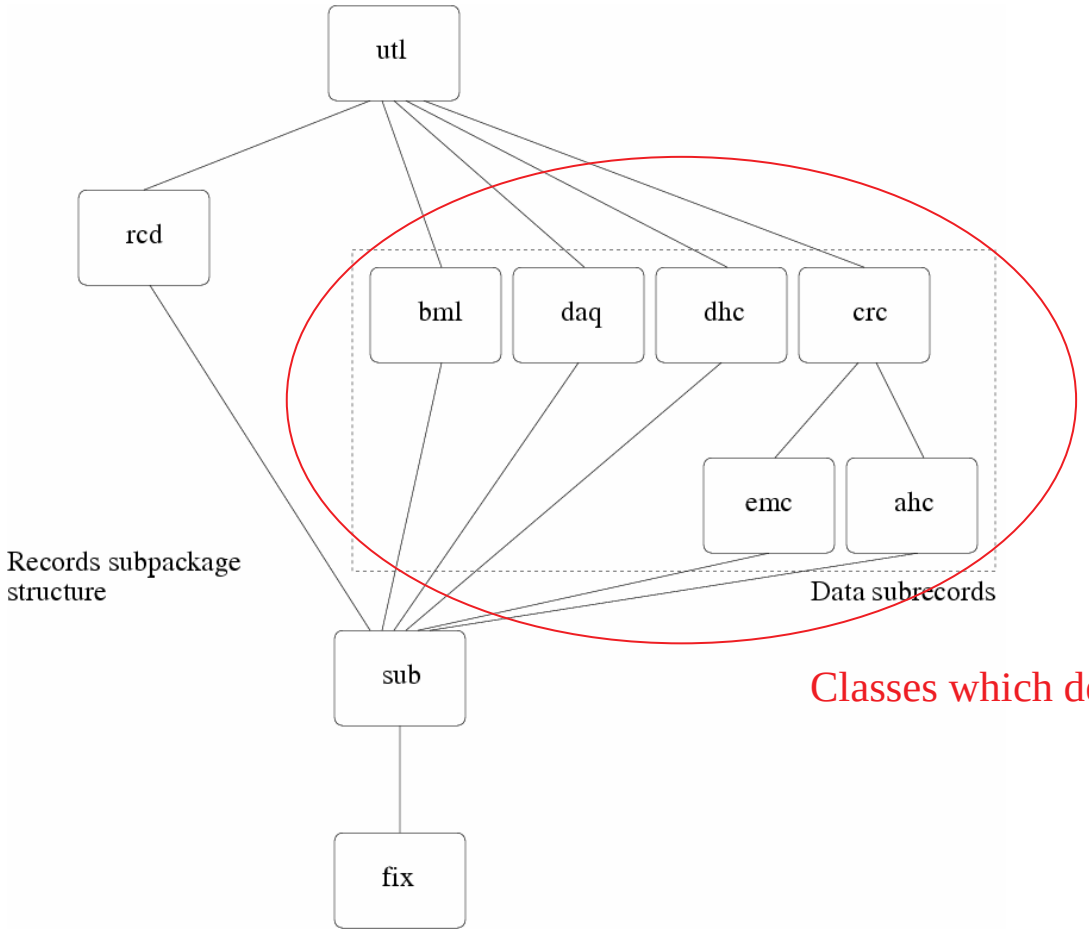
DAQ software packages

- Three major packages
  - records – code for handling records
  - daquser – code for “semi-offline” analysis/monitoring in DAQ system
  - online – true online code



Records software package

- C++ classes to read and access raw data records and subrecords
- Only package needed for “real” offline work; LCIO conversion



Offline use of records

- Straightforward to read raw data files; all code in records package
- Basic read

```
RcdArena arena;
RcdReaderBin reader;
assert(reader.open("myfile"));  // ".bin" is appended
while(reader.read(arena)) {
    // Use record
}
assert(reader.close());
```

- Accessing the subrecords, e.g.

```
SubAccessor accessor(arena);
std::vector<const DaqRunStart*> v(accessor.access<DaqRunStart>());
```

  gets a list of pointers to the DaqRunStart objects

- Need to check records package classes for data in each subrecord
  - XxxRunData, XxxConfigurationData, XxxEventData for each system
  - What each contains is very system-dependent

Work to be done

- Debug future versions of BE firmware, test new CRCs
  - Hope this can be finished by end of year
- Complete major rewrite of online software structure
  - THE major task at present; target is again end of year
- Push maximum trigger rate during spill; currently 2kHz
  - This satisfies basic requirement but would benefit from faster rate
- Push maximum readout rate during transfer; currently 50Hz
  - Requirement is 100Hz; some tricks will be needed to achieve this
- Test parallel access for two PCI cards in one PC
  - PCI bus should not limit compared with two VME buses but need to check
- Test socket access for two PCI cards in two PCs
  - Each reads independently but need to merge records afterwards
- Integrate existing beam line equipment at CERN and FNAL
  - Big uncertainty at present