Data AcQuisition "essentials"

Enrico De Filippo

A short introduction to Data Acquisition systems

- **DAQ definition**
  - Basic concepts (What is a DAQ ?)
  - Front End Electronics (FEE), triggers,
  - Digitization, Readout, dead time. . . . . .
- **Buses (DAQ Hardware)**
- **Analog vs. digital DAQ**
- **Networks and data transport**
- **DAQ Software**

Data acquisition systems

A DAQ ig generally designed to fulfill requirements of a single experiment, a single device or devices operating in a laboratory and DAQ design depends on many factors: number of channels, expected rates, detectors, front-end electronic, physics that we want to study, etc.

All these aspects can brought to completely different technical solutions, also in order to solve the same kind of problems.

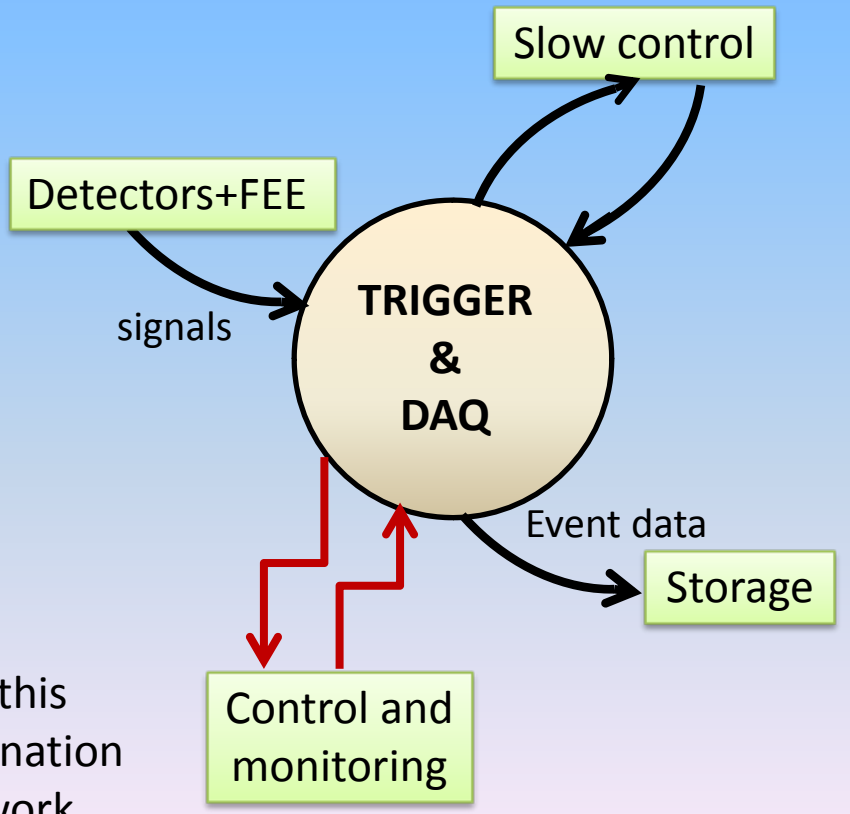
Thus technologies and complexity can be greatly different for various DAQs but we want to look for same basic aspects that are common to all data acquisitions. This gives perhaps the possibility to see what are the some current trends in this field.

What is a DAQ ?

The basic motivation of a DAQ is to **process** signals generated by a detection's apparatus **writing** in a permanent way the most **interesting correlated information (events)** onto data **storage** for data analysis.

KEY-WORDS

Front end electronic (FEE)
Data selection (Triggers)
Codifiers (ADC/QDC/TDC/SADC)
Data Readout from codifiers
Slow control upon FEE and DAQ
Buses for readout and data transfer
Synchronization and Event Building
Data Networking
Permanently Writing data (Storage)
On-Line Analysis and monitoring



Credits: I have adapted many ideas and materials for this lecture from W. Vandelli (CERN) and ISOTDAQ (Internation School of Trigger and Data Acquisition) lectures and work with the CHIMERA detector.

DAQ SCALING

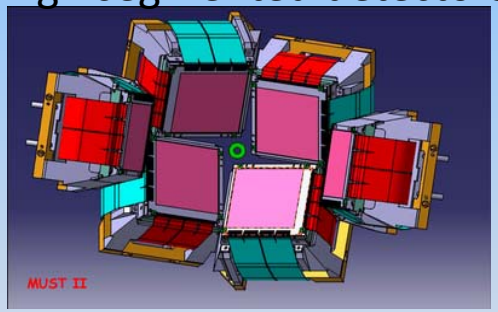
Low and intermediate energy physics

Chimera DAQ parameters	
Number of crates	7
Number of boards	**>100**
Number of parameters (channels)	**~6000**
Acquisition rate	1÷ 3 KHz
Average event size	1÷ 3 Kbytes
Average transfer rate	1 ÷ 3 Mbyte/s
Average data production/beam week	500 –1000 Gbytes

Chimera @LNS

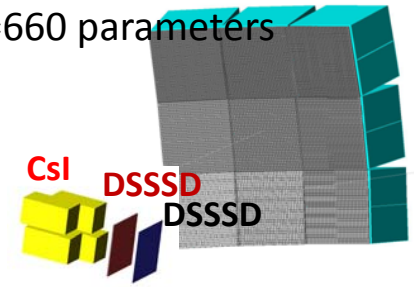
4π detector CHIMERA

High segmented detectors

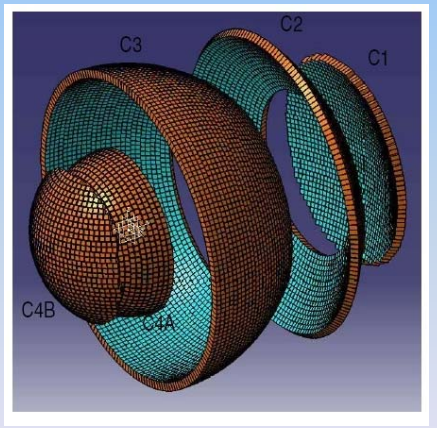


Must2@GANIL
Number of parameters:
(E,t): $576*N_{tel}$

132 channels each cluster
≈660 parameters



Ancillary detectors: the **FARCOS** project; it can be coupled with a 4π or other detectors



4π FAZIA project: 9974 modules; $>10^4$ channels (see G. Poggi talk)

The DAQ role in the framework of an experiment

The CHIMERA detector

Analog signals

Front-end electronics

Logic and analog signals

TRIGGER

OK

DAQ

"Storage"

On-line analysis

Off-line analysis

Theory, Simulations +
Detector's Filter

Physics
Results

Data analysis

TDAQ: Trigger and DAQ

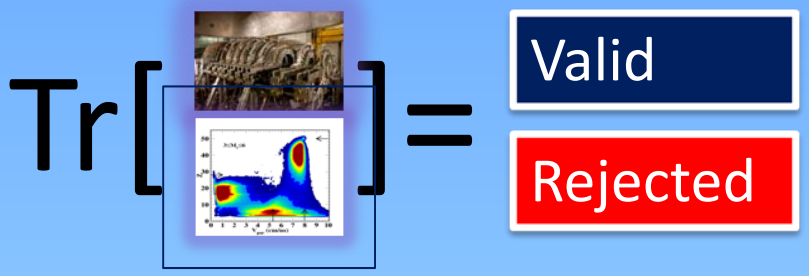
TRIGGER: the trigger make a selection starting from the signals coming from the front-end electronics (**hardware trigger**) thus deciding if the event have to be **readout** or **rejected**.

First level trigger **L1:** is based generally on hardware (discriminators, charged particle multiplicity, etc). It is generally fast (100ns-1μs).

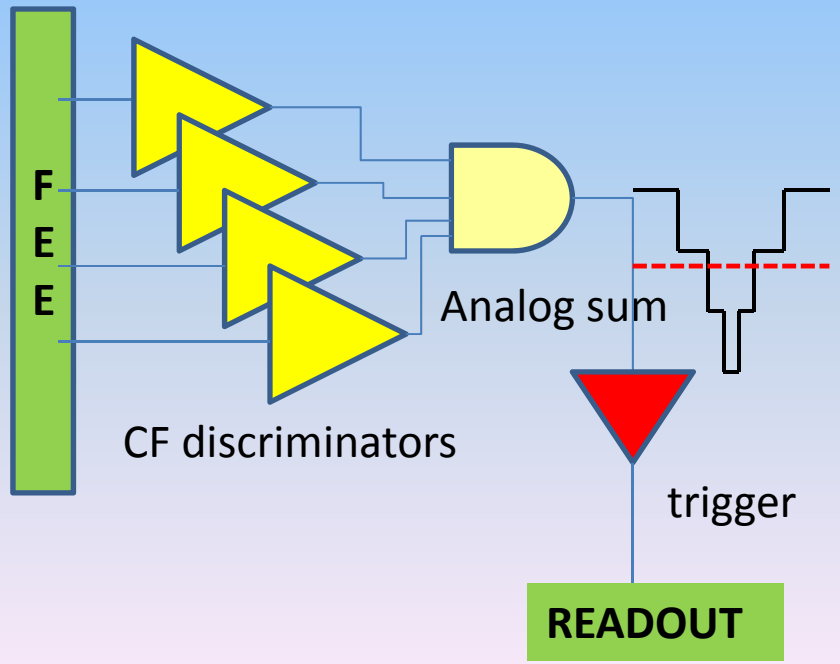
Second level trigger **L2:** for example particle identification, energy calibration, etc. (CPUs, FPGA, etc). (milliseconds)

Third level trigger **L3-Ln:** in high energy physics, event reconstruction, physics processes (computer's farms). (10^{-2}- 10^{0} sec).

Simplified L1 trigger in CHIMERA based on particles Multiplicity

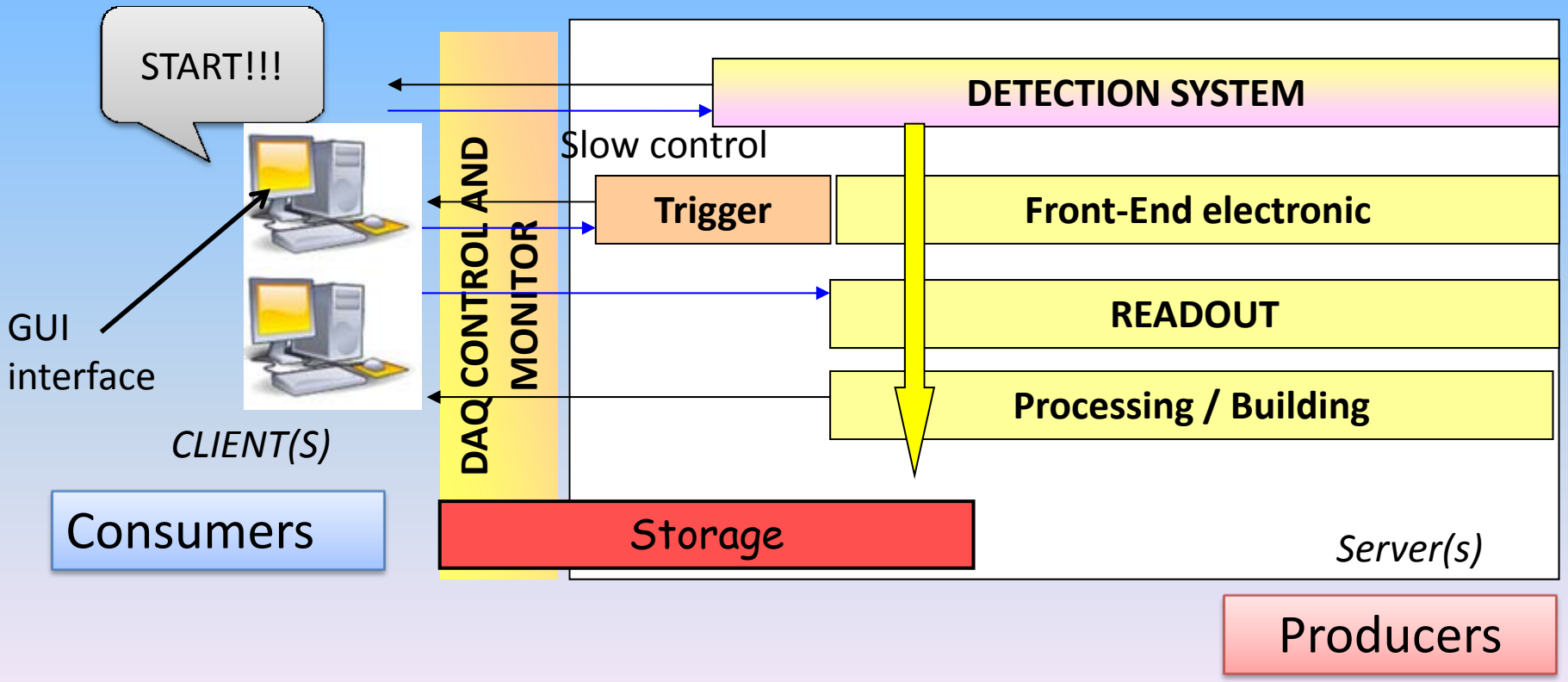


Trigger choice depends on **detector and physics**



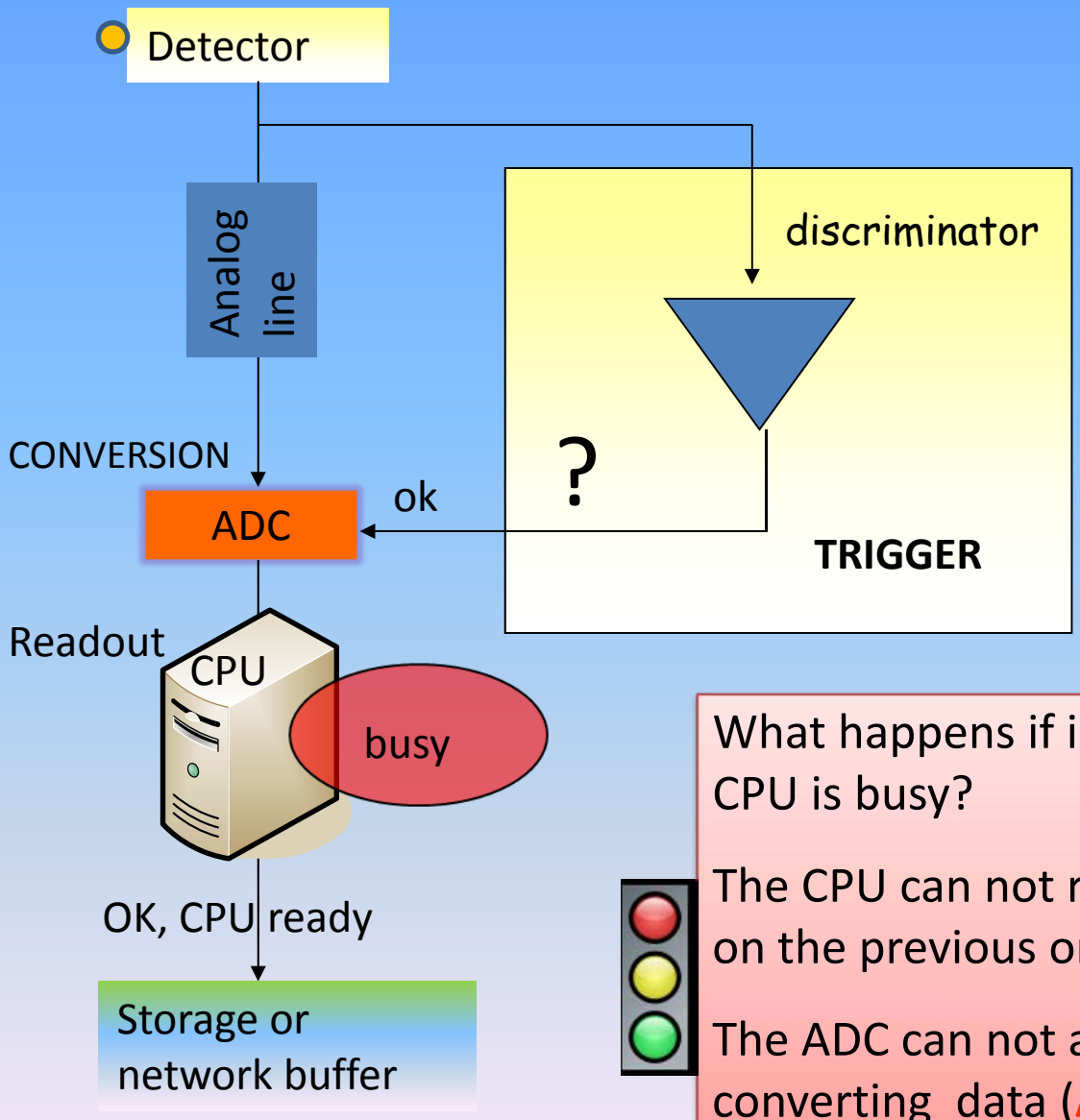
DAQ

FEE : front-end electronic: from preampliers to codifiers: produces digitized information
READOUT: Read data from codifiers in front-end buses; can include the event-building
Event Building : event reconstruction, assembles buffer events
DATA Transfer (Networking): Data transfer to various DAQ server or Consumers client
Storage, control and on-line monitoring
GUI: graphical interface: access, setup to slow controls and on-line data analysis



An elementary DAQ (with something wrong)

Radiactive Source: random process



What happens if is there a new trigger when the CPU is busy?

The CPU can not read other data while working on the previous one (**CPU busy**).

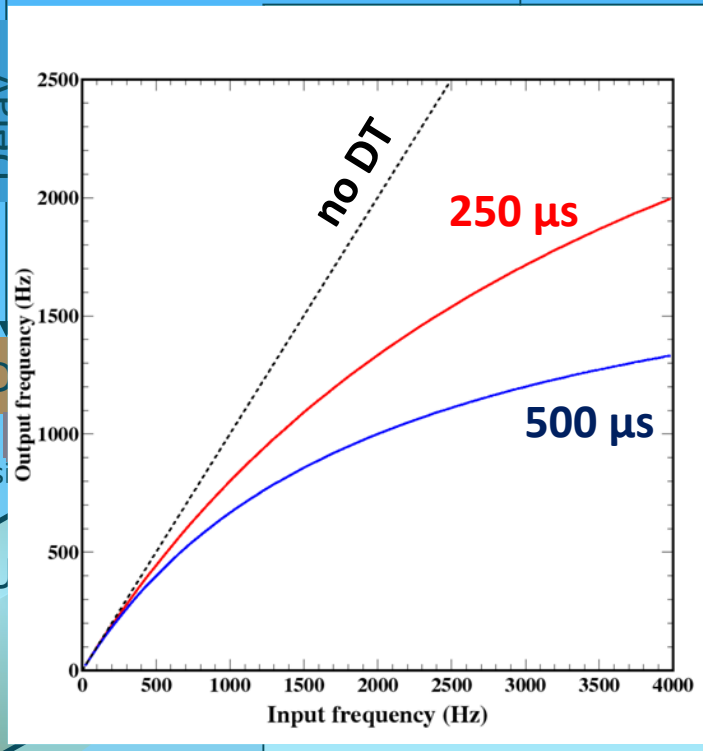
The ADC can not accept a new trigger while converting data (**ADC busy**)

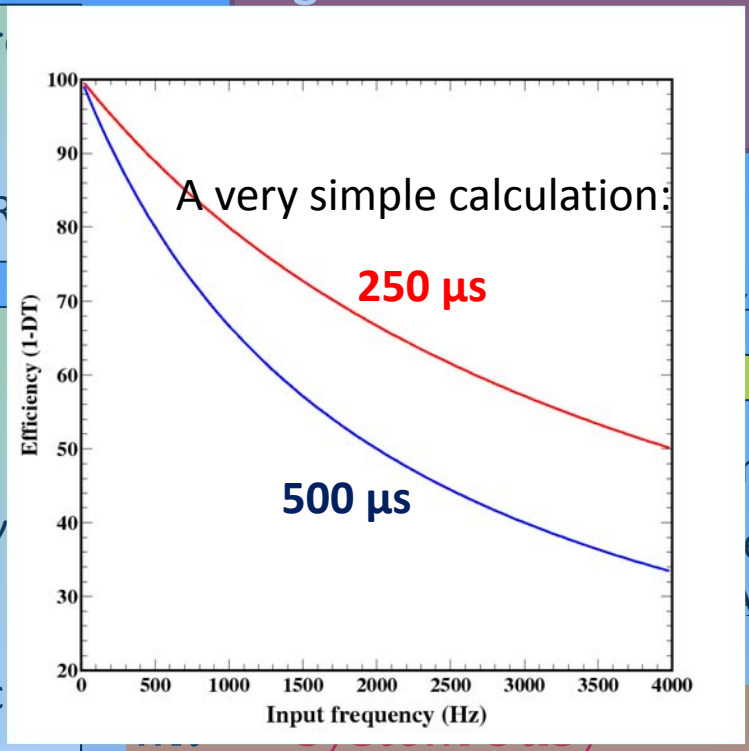
An elementary DAQ with busy logic: the DEAD TIME

Detector

f = 1 KHz 1/f= 1 ms (input rate)

Now the system is protected by a busy logic but . . . . . not all





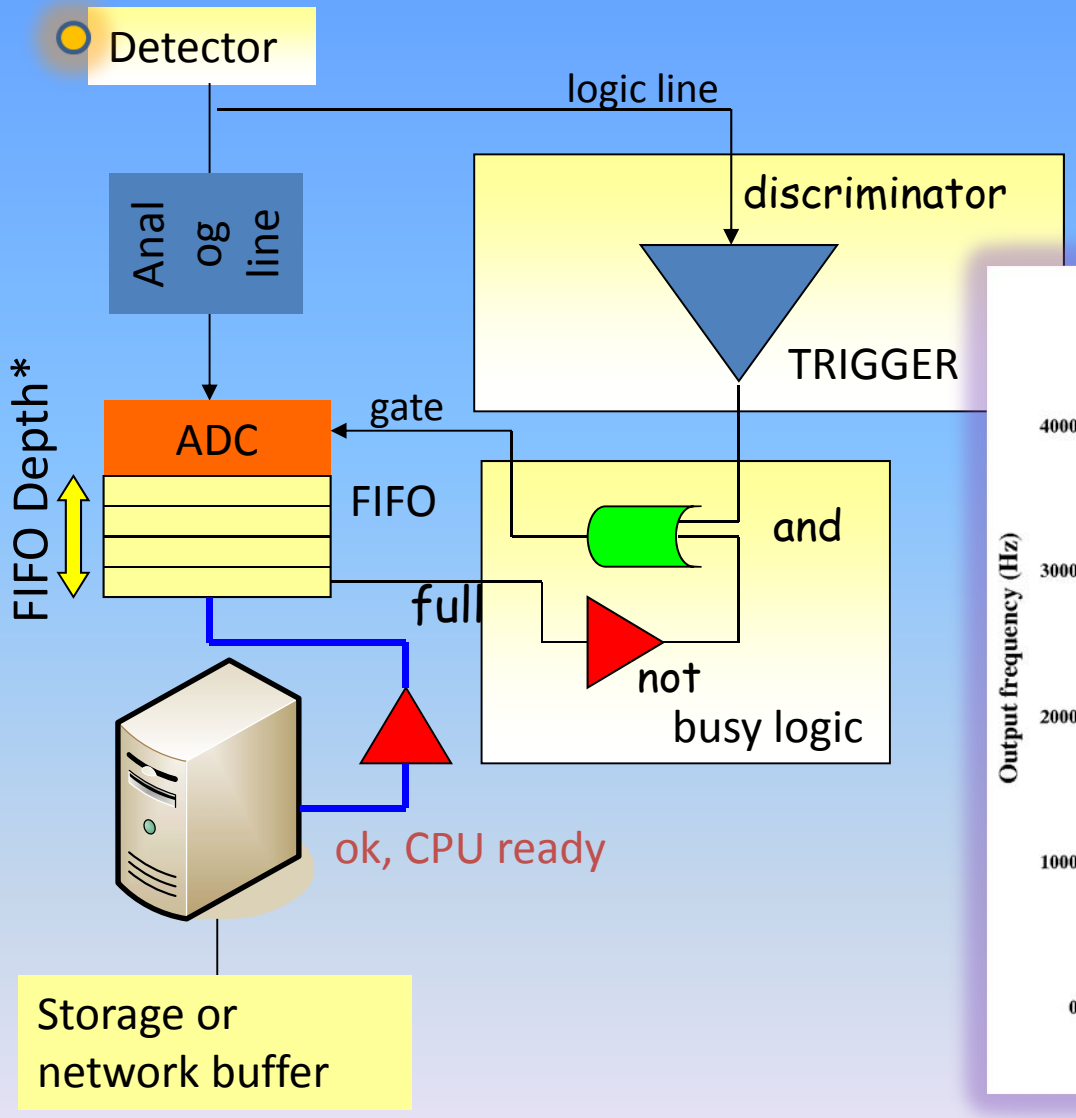
Storage or network buffer

DeadTime – operational is the ratio between input triggers and the accepted ones.

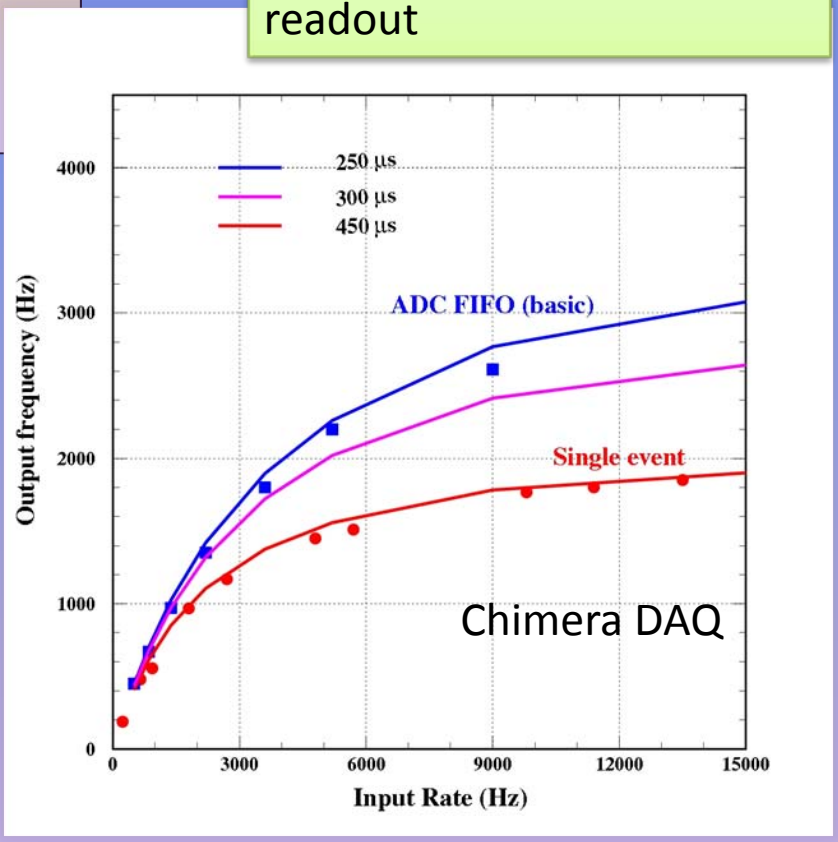
$1 - m\tau$ *system free*

$$f(1 - m\tau) = m \rightarrow m = \frac{f}{1 + f\tau}$$

An elementary DAQ with FIFO (buffer area organized as first-in / first-out)



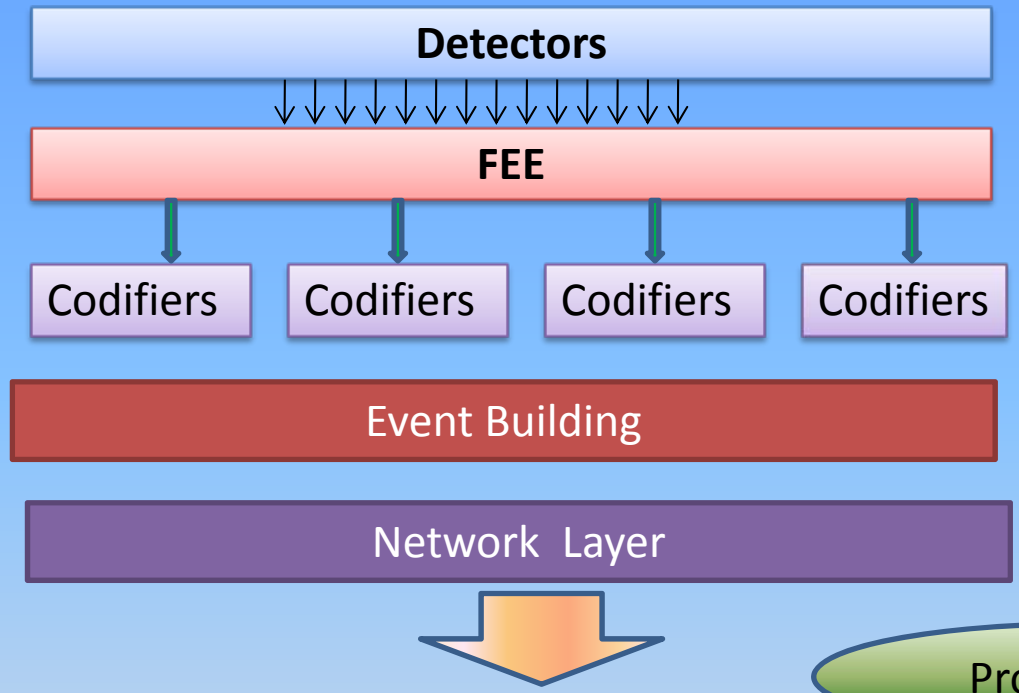
Chimera DAQ: using a minimum FIFO level, depth = 2 (**blue points, double buffer**) during readout



*FIFO:
1) ADC multievent buffer (digital)

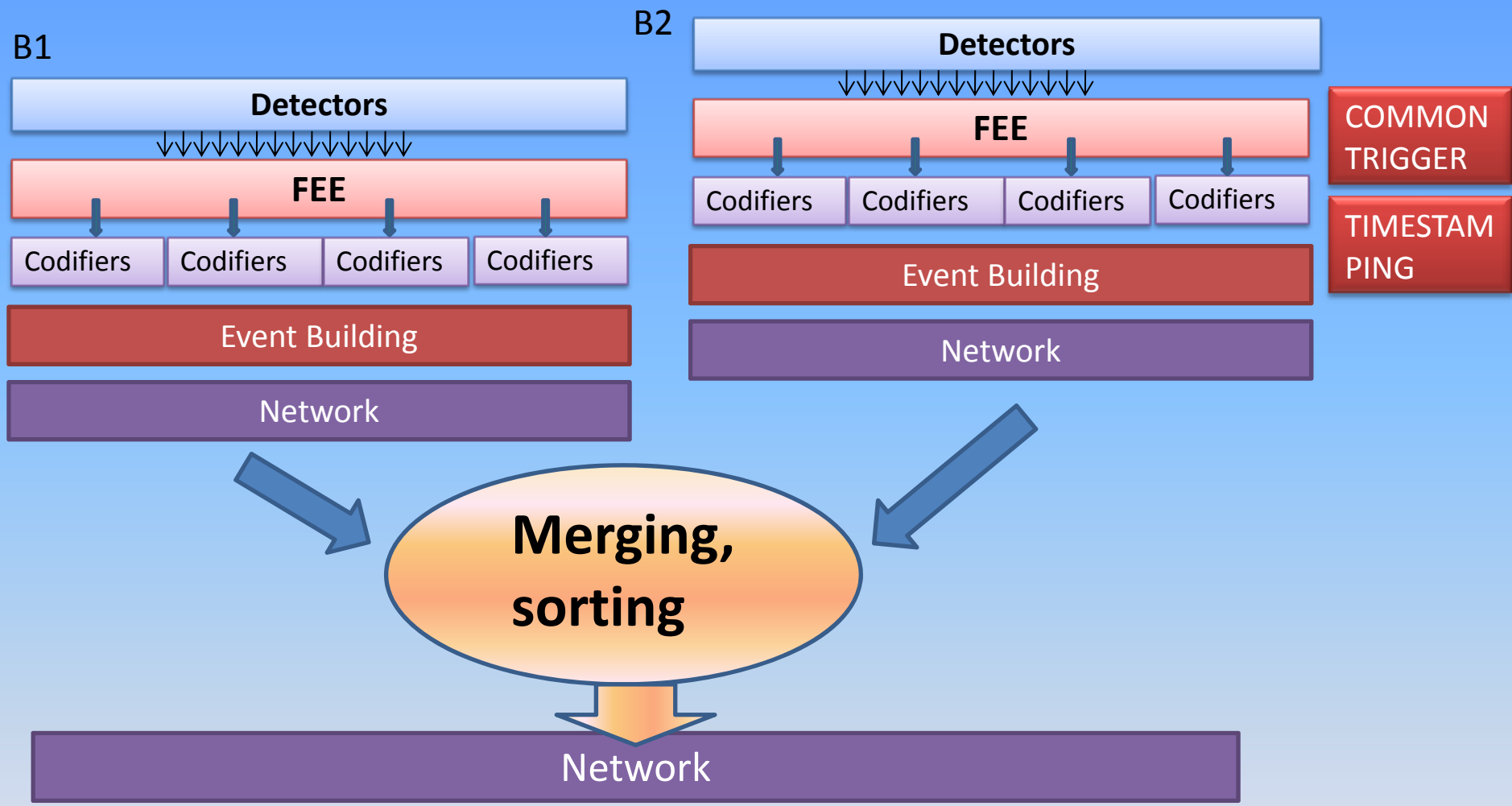
When the complexity of the system increases simple calculations have to be substituted by **Monte-Carlo simulations**

"real world" DAQ: distributed architecture



If there are many channels, DAQ can be divided in blocks (for example all boards of a crate, all crates, a farm of computers to process data). All theses blocks have to be **interconnected**. Synchronization: to form full events from partial ones. **INTERCONNECTION** → **Bus, Networks**

"real world" DAQ: distributed "multi-branch" architecture

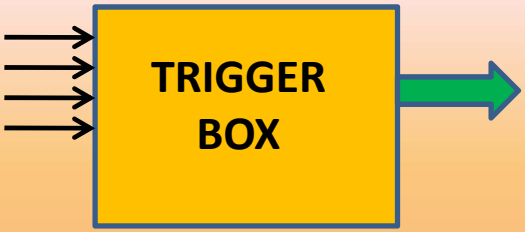


Different DAQs can be put together or a DAQ (for example referring to different detectors systems) can be divided in separate blocks

Real TDAQ: different trigger solutions (depends upon physics requirements)

Common TDAQ at LNS, GANIL, GSI…

Generally First level trigger based on **hardware** (NIM modules) or upon a **TRIGGER BOX** (programmed **FPGA**).

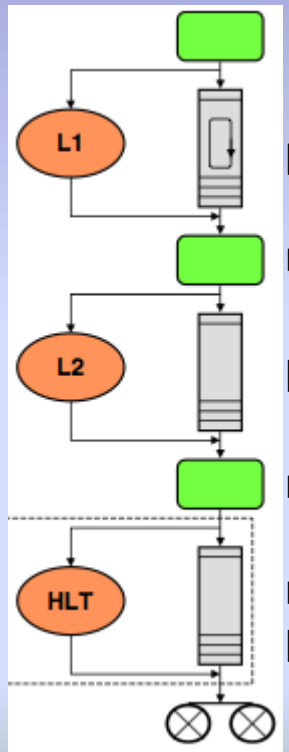


The trigger box is an event selection filter:

- ❑ Event selector
- ❑ Enable/disable channels
- ❑ Manage Dead-time logic
- ❑ Generate "patterns" of selected events
- ❑ Possibility to use divider for a given trigger type
- ❑ Interfaced to a bus (VME, . . .)
- ❑ Slow control (setting)

LHC . . .Cern

Bufferization + **pipelining** techniques (software + hardware architectures).



Agata, ….

SOFTWARE TRIGGER OR Triggerless

Based on **timestamp (time markers)**

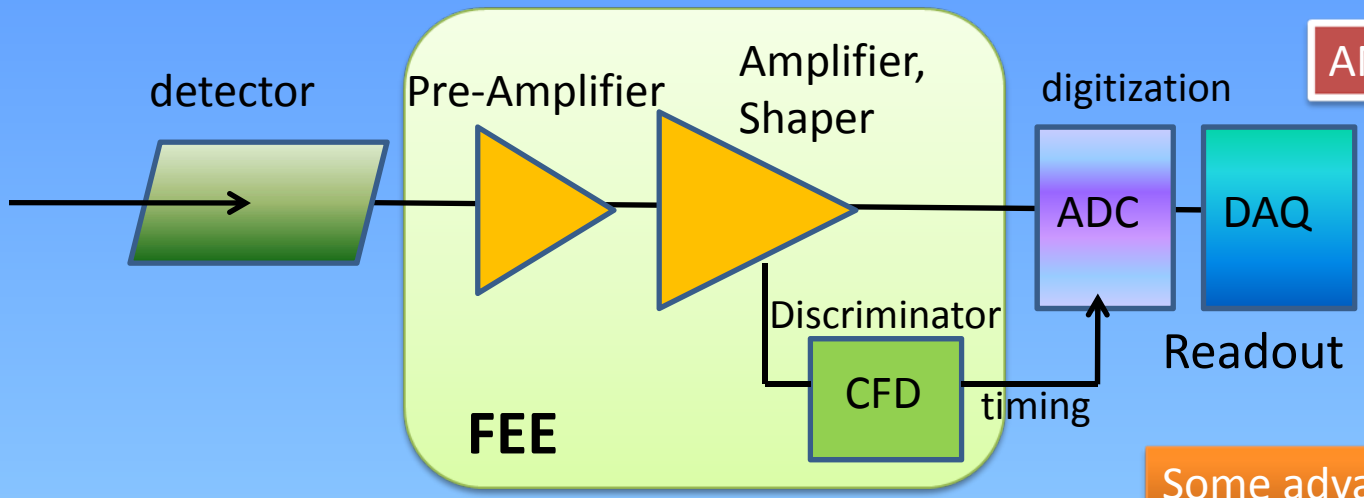
TIMESTAMP

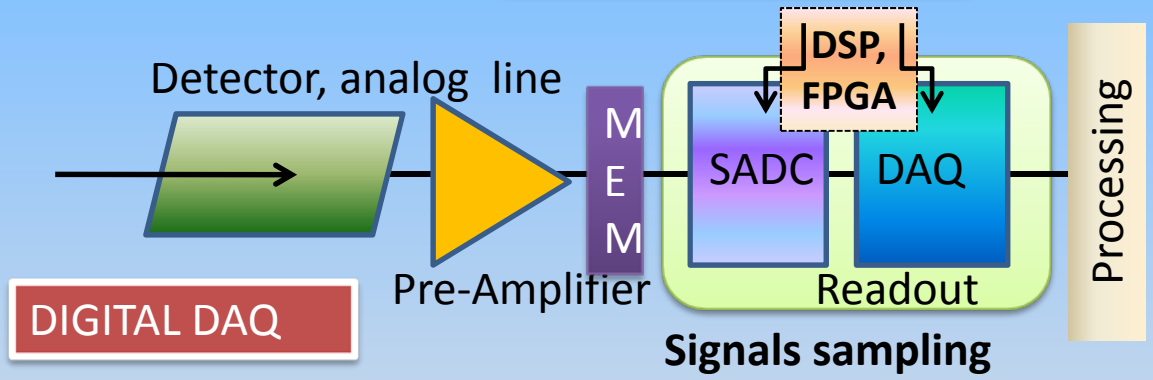
Codifiers

Building, Event reconstruction

Store only selected data

ANALOG vs. DIGITAL DAQ (simplified but realistic view)





Some advantages of digital DAQ

- ✓ Possibility to implement ***optimal filters*** which are hardly realized with analog techniques
- ✓ Insensitivity to pick-up noise, as soon as the signal has been digitized
- ✓ Flexibility and possibility to periodically recalibrate the system
- ✓ Less number of modules → savings and spectrum stability increasing
- ✓ Possibility to implement online computation of algorithms directly on the shape of collected pulses (sample database)

Some dis-advantages of digital DAQ

- ✓ More difficulties with timing measurements
- ✓ Large amount of data rates in the readout bus if all samples are collected especially for large number of channels

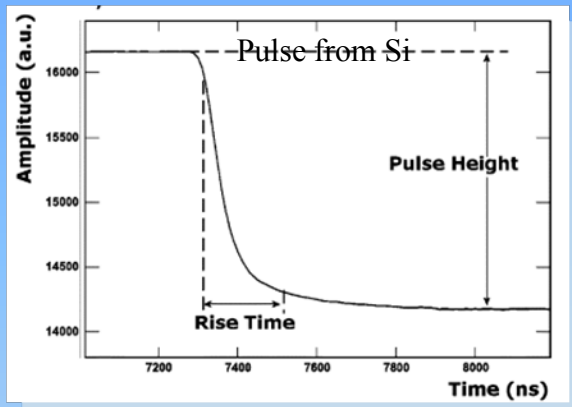
GOAL: determination of pulses Rise Time (τ) as a function of ΔE(Si)

Pulse Handling: pulse processing dire
collected from Chimera preamplifier

Waveforms of 2048 consecutive samples are collected at 10 ns time intervals

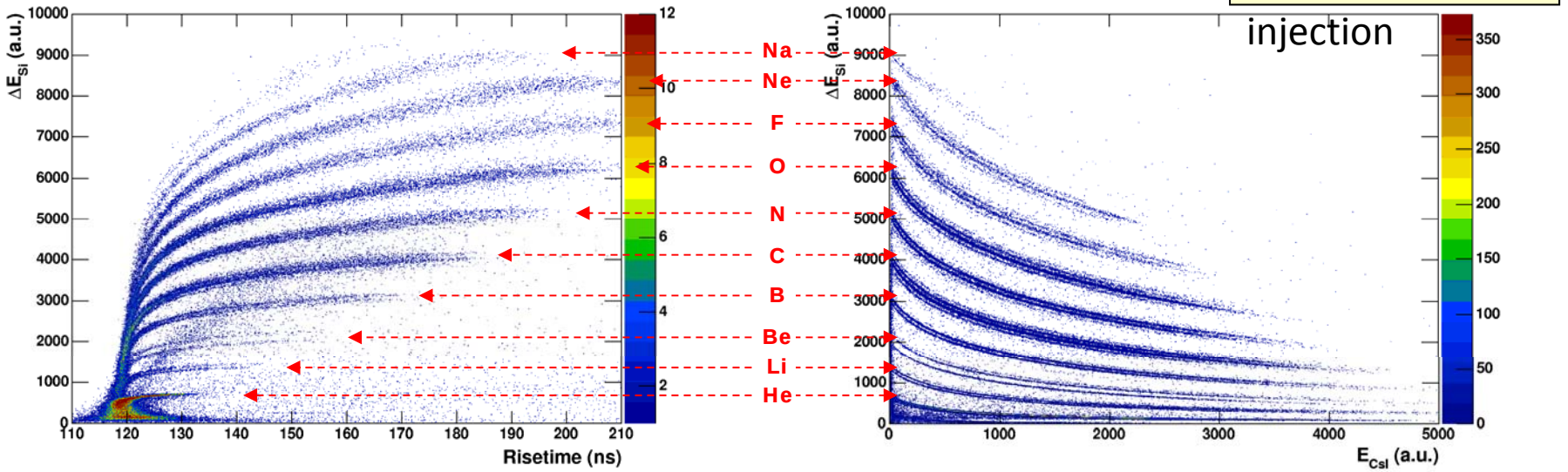
Used Algorithms for Silicons :

- ✓ Estimate of signal baseline
- ✓ Pulse Height (maximum)
- ✓ Rise time calculation (*difference between the times at which the pulse crosses 10% and 90% of the full pulse height relative to baseline*)

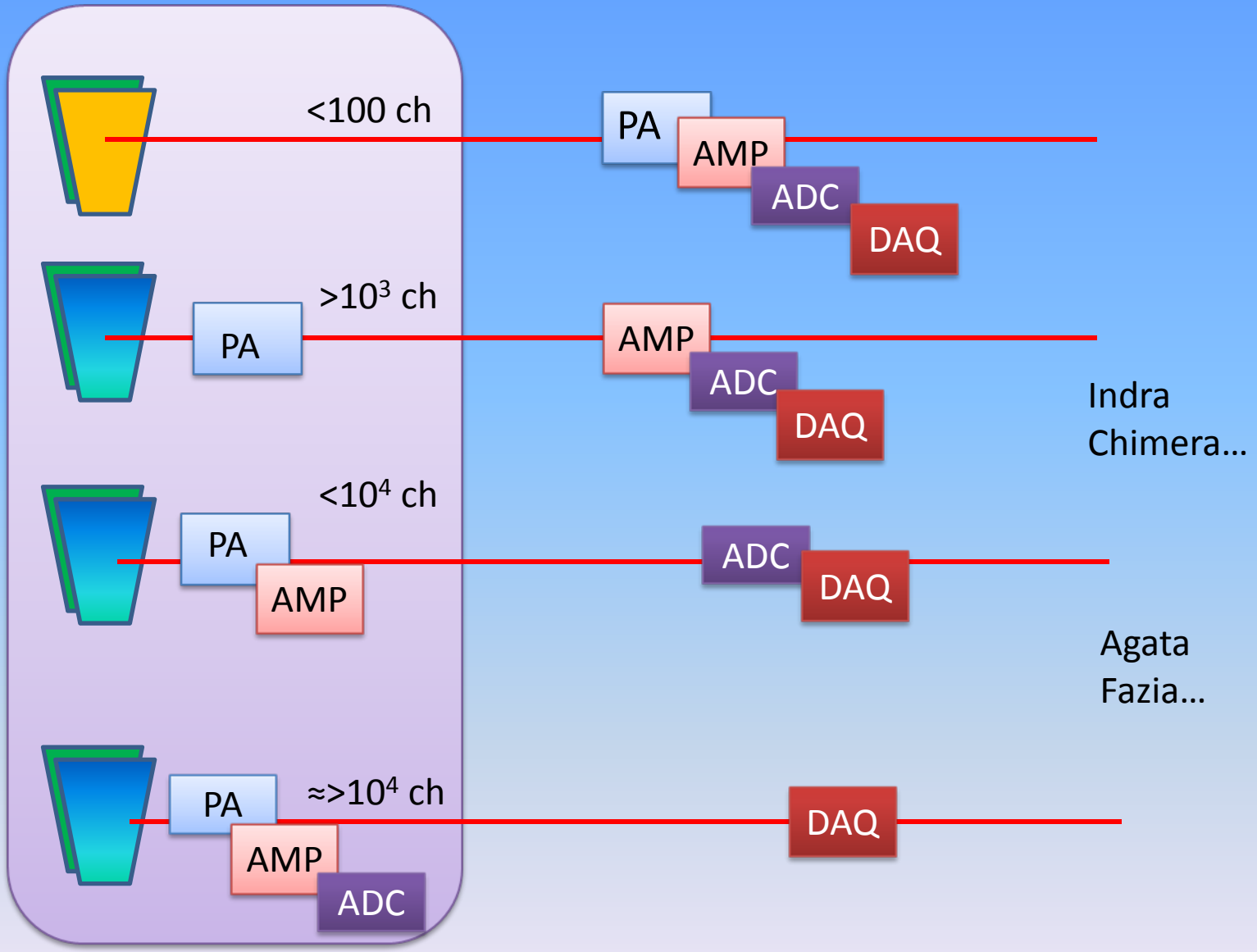


^{20}Ne + ^{12}C @ 20.5 MeV/u - θ_{lab} = 12°

Front side injection



Schematic evolution in nuclear physics front-end (adapted from E. Pollacco)

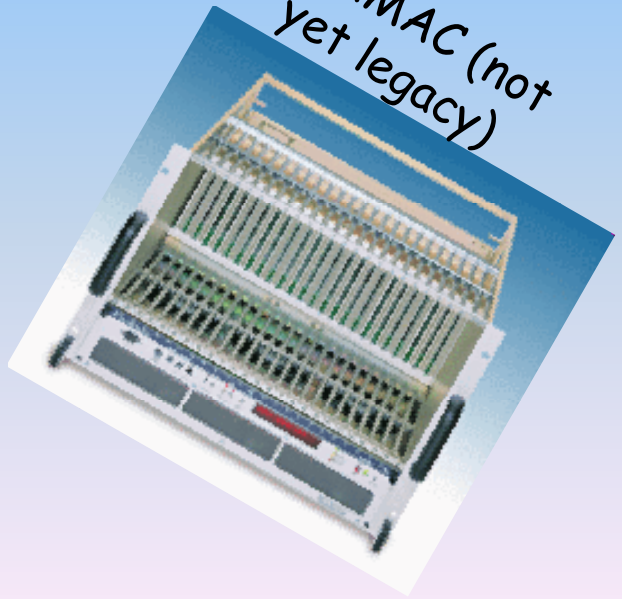


BUSSES

VME

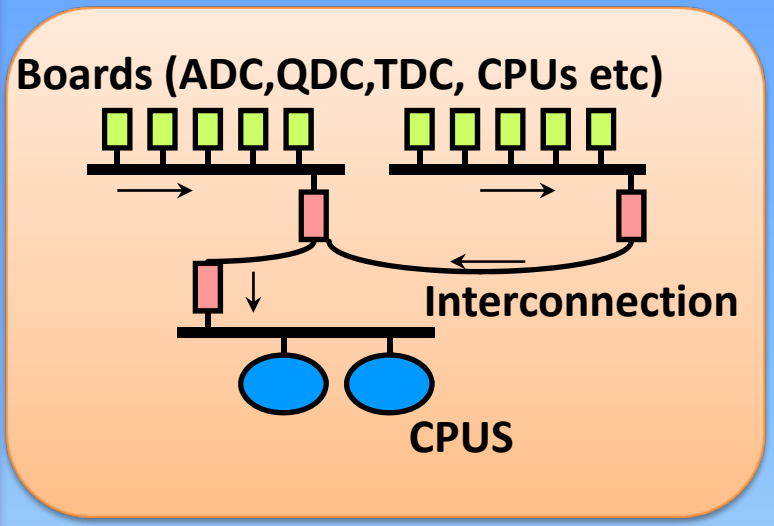
ATCA, µTCA, telecommunications

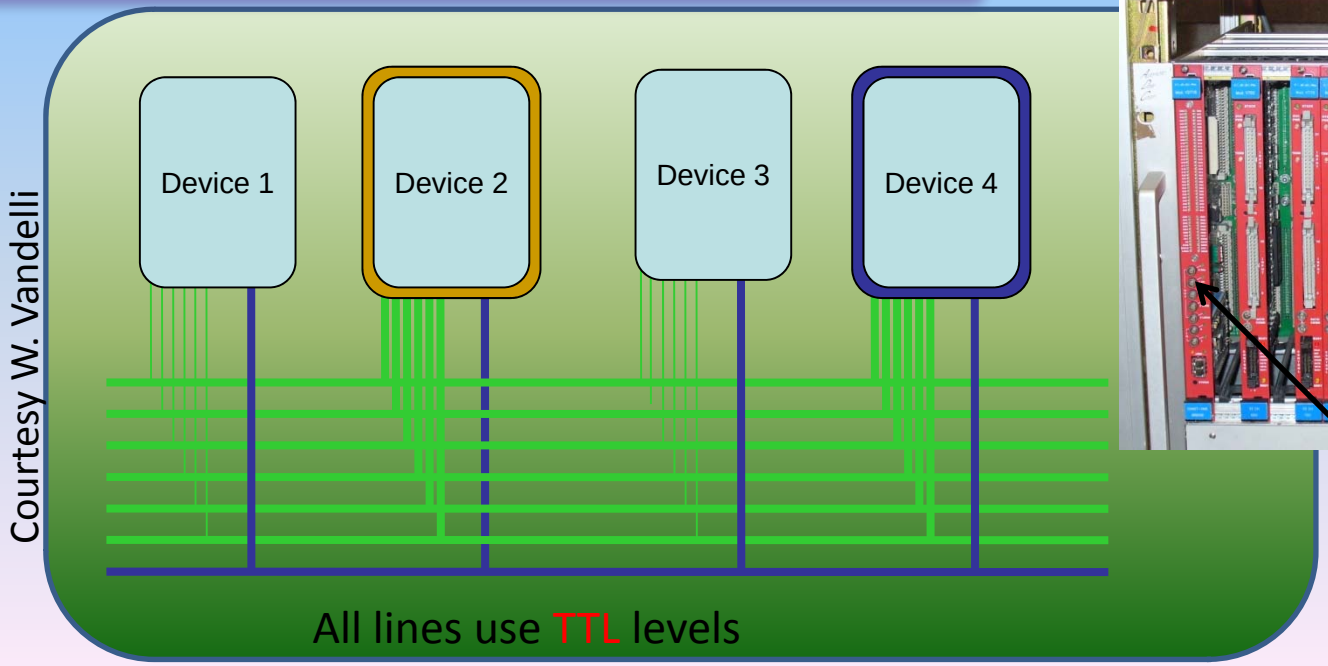
CAMAC (not yet legacy)



Buses

- ➔ Esemples: VME, VXI, CAMAC, PCI, PCIe …
- ➔ Devices are connected by a “shared” bus
- ➔ Devices can be “master” or “slave”
- ➔ Devices can be addressed in a unique way in the bus.
- ➔ Because the bus is shared among different board a bus “arbitrer” is generally necessary (VME)





VME (Versatec Module Europe) basic glossary

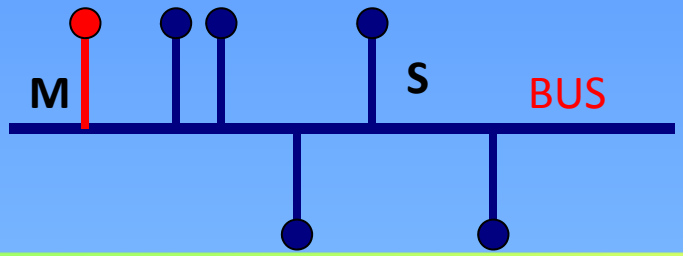
– Master
- A module that can initiate data transfers

– Slave
- A module that responds to a master

– Interrupter
- A module that can send an interrupt (usually a slave)

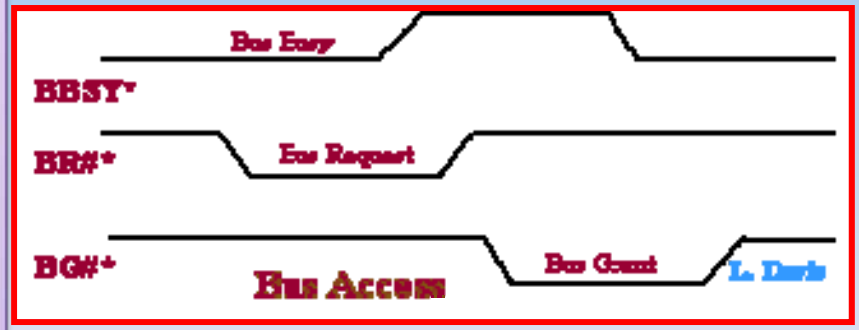
– Interrupt handler
- A module that can receive (and handle) interrupts (usually an embedded CPU)

– Arbitrer
- A piece of electronics (usually a CPU) that arbitrates bus access and monitors the status of the bus. It is generally installed in slot 1 of the VMEbus crate



BUS master-slave: the master manage the VME cycle and assert on the bus the control and address lines.

The data transfer is **asynchronous** : the slave ends the VME cycle with an acknowledge (DTACK) when finished or a bus error (BERR)



Read access to a slave board

a) Master asks access **[BR]** bus request
b) Controller answers ok **[BG]** bus grant
c) Master takes control and puts the bus in busy state **[BBSY]** doing the transition

Chimera VME bus (behind cables. . .) at work

Analog lines (from amplifiers) or TDC start (from CFD discriminators)

Gates and busy lines

Chimera detector readout has actually 7 VME crates (≈ 6000 channels)

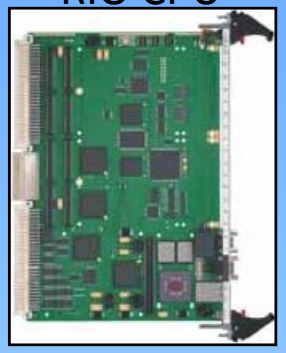
How fast is VME bus ?

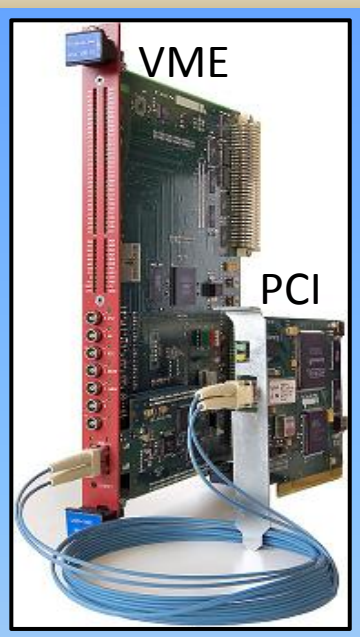
www.vita.com: information on VME BUS

Type	Bus cycle	Max. Velocity (theoretical)
VMEbus IEEE-1014	**Single**	**4 Mbytes/s (D32)**
VMEbus IEEE-1014	**BLT**	**40 Mbytes/s**
VME64	MBLT	80 Mbytes/s
VME64x	2eVME	160 Mbytes/s
VME320	2eSST	320... Mbytes/s

Asynchronous

RIO CPU





Single cycles

Transfer 8, 16 or 32 bits of data (typically) under the control of the CPU on the master (D8, D16, D32) Typical duration: > 1 µs

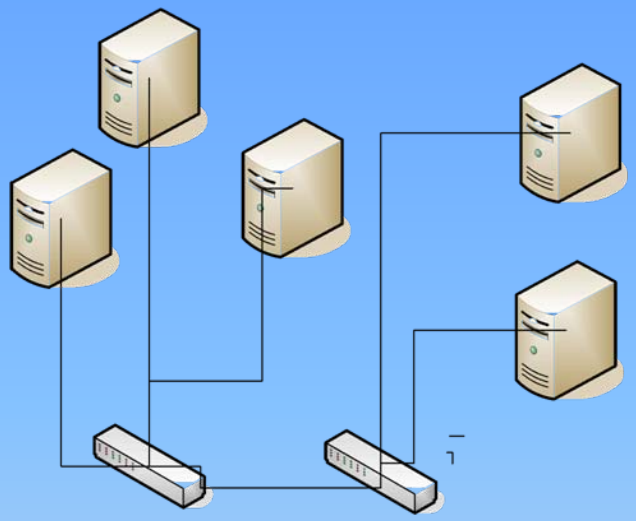
Block transfers (DMA = Direct Memory Access)

Transfer any amount of data (usually 32 or 64 bit at a time) under the control of a DMA controller (CPU independent) (D32BLT and D64MBLT)
Data is transferred in bursts of up to 256 (D32) or 2048 (D64) bytes
Typical duration: 150 ns per data word

Interrupts

Used typically by slaves to signal a condition (e.g. data available, internal error, etc.)
Can have 7 different priorities (Interrupt vs. polling)

READOUT NETWORKS IN A DAQ



INTERCONNECTION IS OBTAINED BY **SWITCHES (it switchs frames to destination using MAC)**

It is possible to connect and exchange data between different "crates" or devices in a network making in parallel the readout of different parts of a multidetector

A network can link also different devices (for example permitting the coupling of different data acquisitions (in a network the connected devices are called "nodes" or "peers" **(no master or slaves)**. In a network devices communicates directly (or through a switch) exchanging data and messages.

ETHERNET is the most used protocol

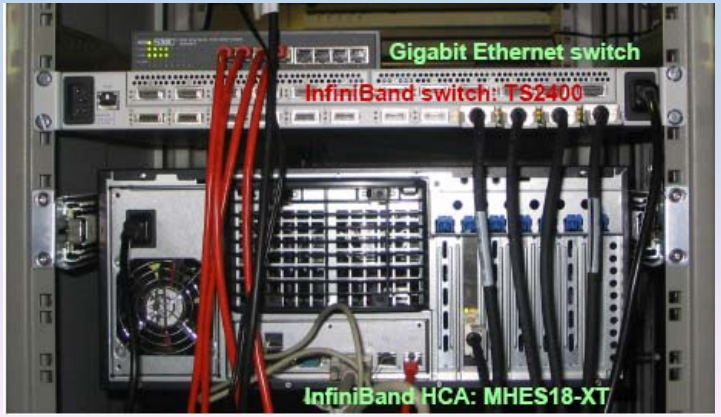
Others high bandwidth (data-transfer/second) , low latency protocols , low distances:

INFINIBAND

MYRINET

. . . . . .

Foto: GSI

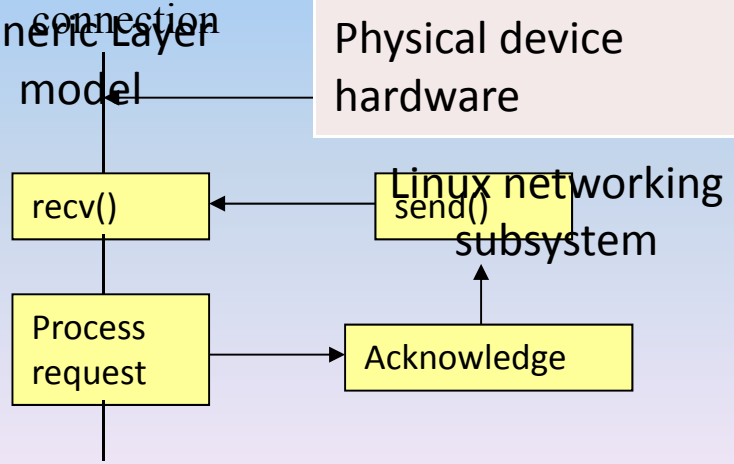


A Ethernet Network: TRANSPORT PROTOCOLS OVER IP

Communication protocols are organized in **LAYERS** model
Transport: socket layers , TPC/IP UDP/IP

Server

Application (firefox)	Application layer
Session (http)	Libreries. User space
Transport (TCP)	Protocol interface
Network (IP)	Network protocols
Data Link(driver)	Device interface
Physical (Ethernet)	device driver (kernel modules)
Generic Layer model	Physical device hardware

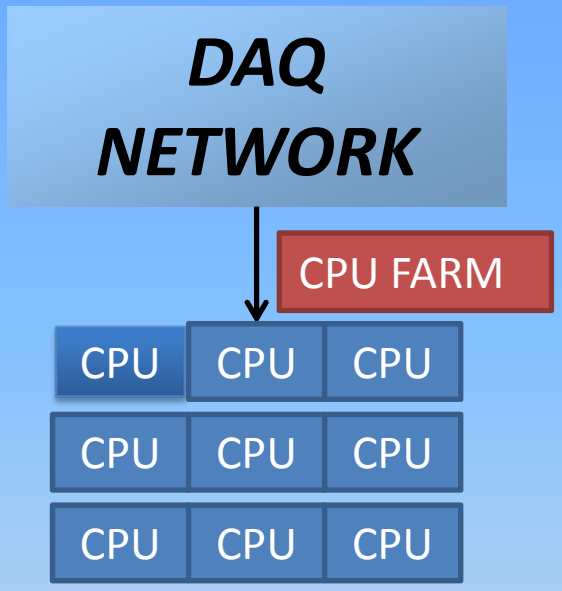


TCP: Transmission Control Protocol. TCP implements a reliable, connection-oriented byte-stream. TCP/IP maintains individual connections between peers and reliable delivers of messages, but has some overhead and can reduce its sending rate as a function of input rate (**STCP** protocol, S=scalable, should overcame this problem)

UDP: User Datagram Protocol. UDP is connectionless protocol and potenzially unreliable: messages could be lost and arrive out of order. A message is transmitted and received in one piece. There is no reduction in data sending rate.

http://www.ibm.com/developerworks/linux/library/l-linux-networking-stack/

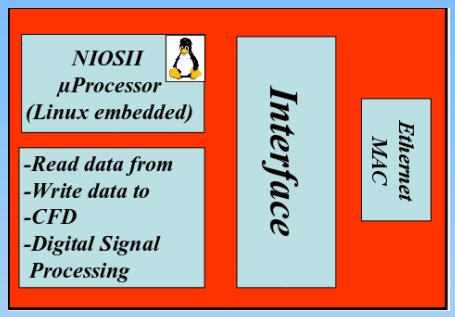
DAQ NETWORK APPLICATIONS: two different ways to process data

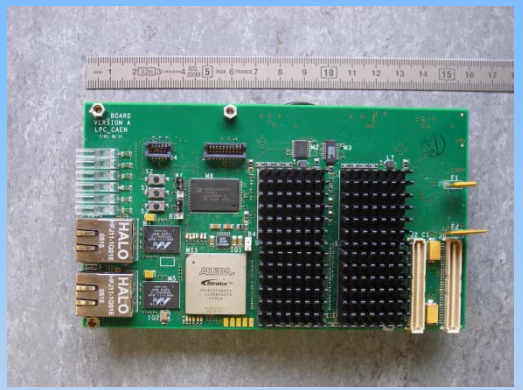


Scaling on a distributed architecture (computer farm).

Chimera, Fazia, Agata . . . .

THE FASTER (Fast Acquisition System on Ethernet Network R&D at LPC (Caen)





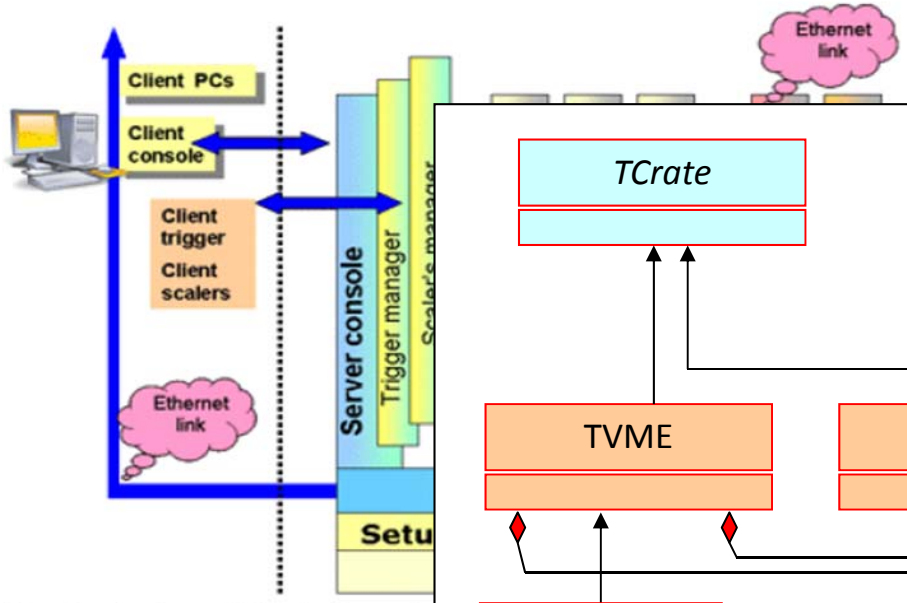
It is a prototype card with four DSP, two Gigabit Ethernet connections, TimeStamping Altera FPGA with a NIOSII processor running Linux

Let the single card do it…

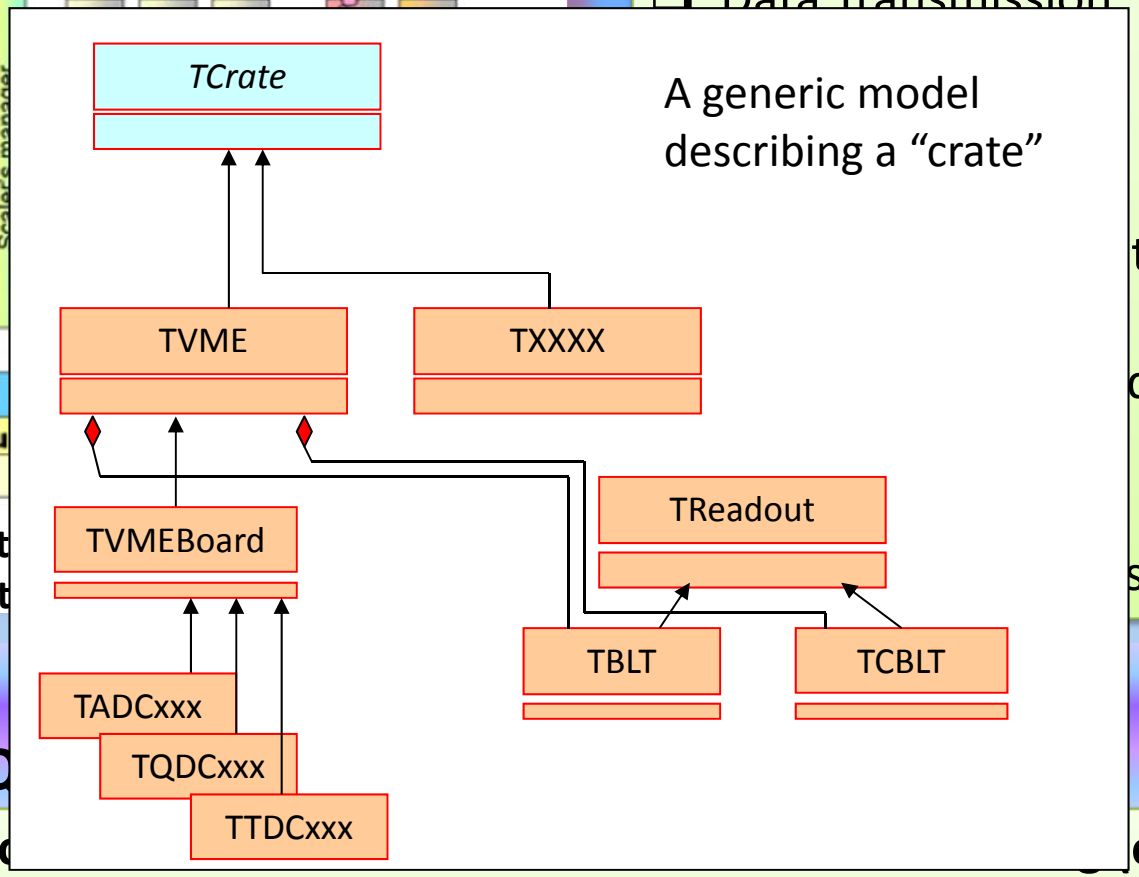
Data acquisition SOFTWARE

SOFTWARE:

- ❑ Data readout
- ❑ Data Transmission



A view of the tasks and soft
KaliMera object library for t



Why an “OBJECT
readout and DAC

CLASSES

A c **object),**
including the object's attributes and the object's behaviour

ABSTRACTION

Designing classes that models the problem at high possible astraction level in order to derive solutions for specific problems from a common basis (code reusing and extending)

Example from CHIMERA DAQ + MBS coupling at GSI

Optical link

chimera

Event builder Chimera

V2718 | TITRIS | 6U VME

V2718 | 9U VME

Timestamp for event synchronization (50 MHz, 48 bit)

UDP/IP

Local online analysis

Titris Bus

MBS: Multi Branch System

RIO3 | TITRIS | TRIVA

Trigger bus

TCP/IP

MBS Chimera Receiver

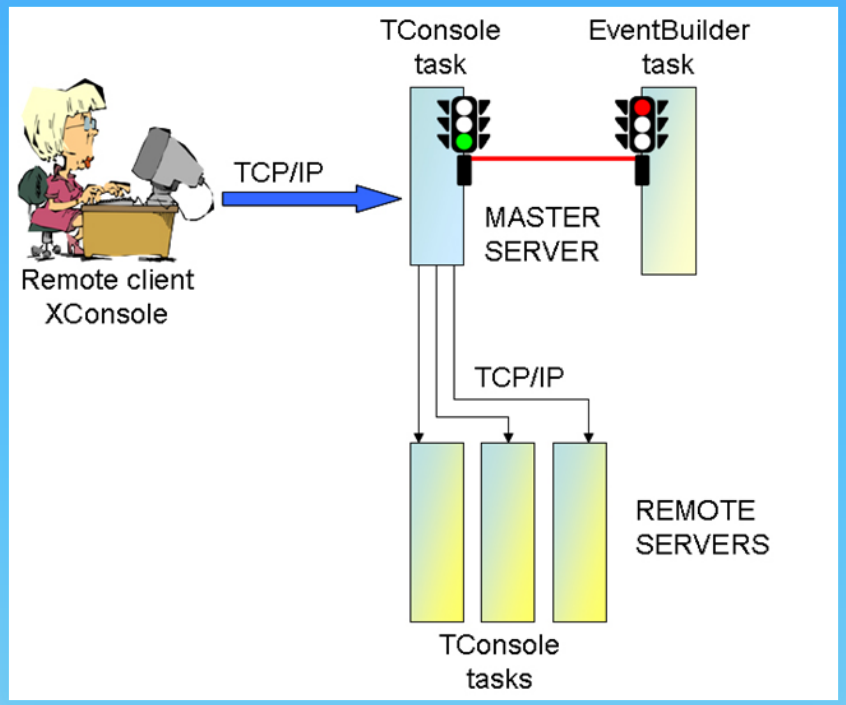
Time ordering, sorting, Data storing
Data to *Global* *on-line analysis clients (GO4)*

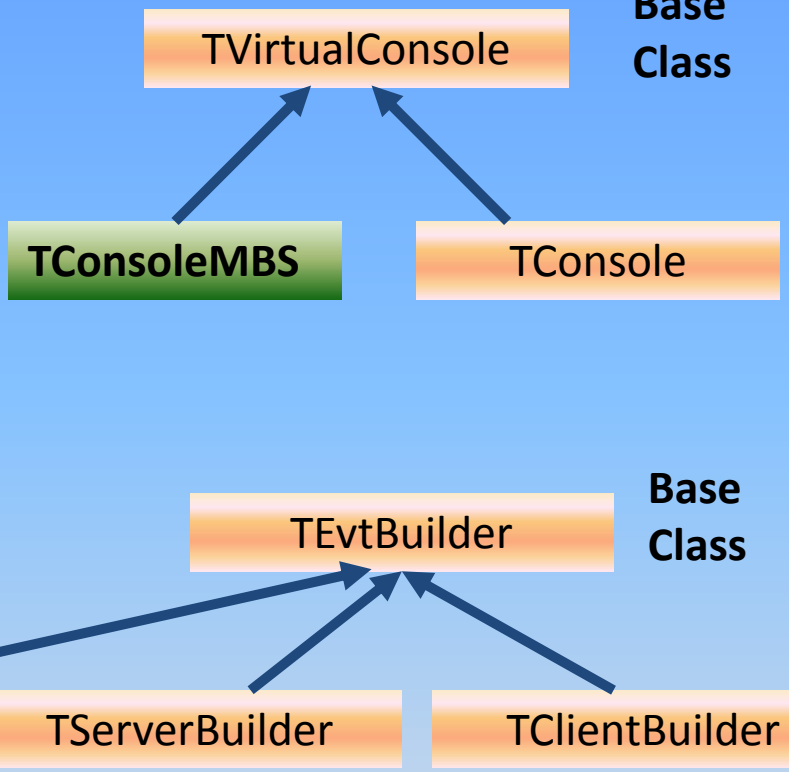
Event builder MBS

Remote DAQ servers

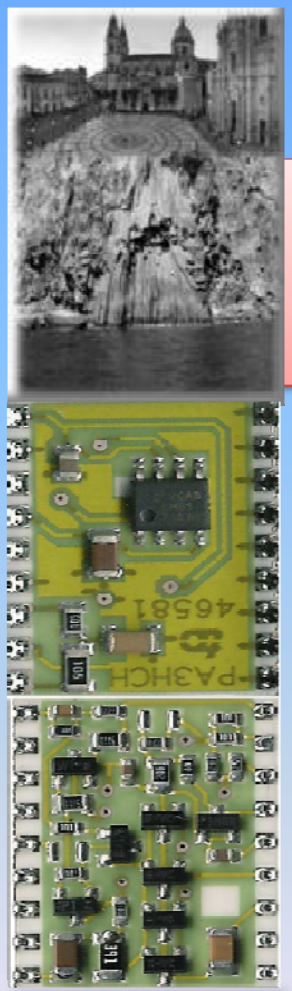
Scalers
Land
TofWall
Krakow Array
MicroBall

Data acquisition SOFTWARE: a simple example from CHIMERA DAQ + MBS coupling





OOP approach creates a framework that in a transparent way permits a sufficient degree of abstraction in order to add or remove components, change or modify readout methods, extend the software implementation following the evolution of the hardware architecture or new needs (like coupling with another DAQ). For this reason OOP is common in software DAQ design
(for example see narval@in2p3.fr) or data analysis and client level (GO4@GSI, Root)

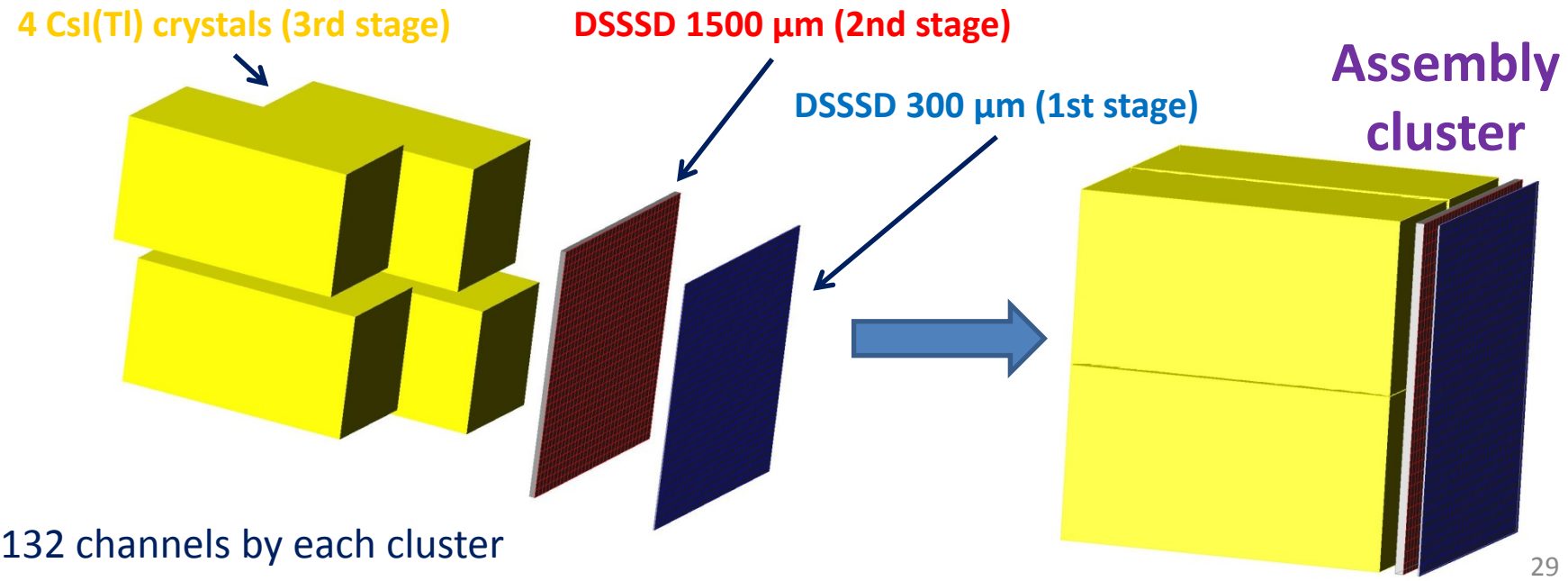


THANKS FOR YOUR ATTENTION

THE FARCOS ARRAY

FARCOS composition

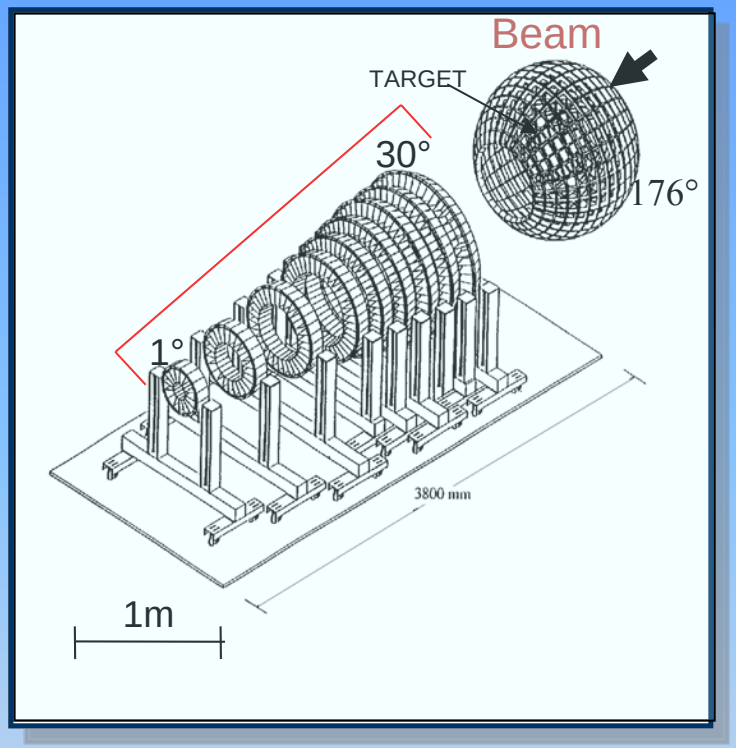
- Based on (62x64x64 mm^3) clusters
- 1 square (0.3x62x62 mm^3) DSSSD 32+32 strips
- 1 square (1.5x62x62 mm^3) DSSSD 32+32 strips
- 4 60x32x32 mm^3 CsI(Tl) crystals (window shape)



132 channels by each cluster

The CHIMERA Detector

1192 Si-CsI(Tl) Telescopes



18 rings in the range $1° \leq \theta \leq 30°$

17 rings in the range $30° \leq \theta \leq 176°$ (sphere)

High granularity and efficiency up to 94% 4π

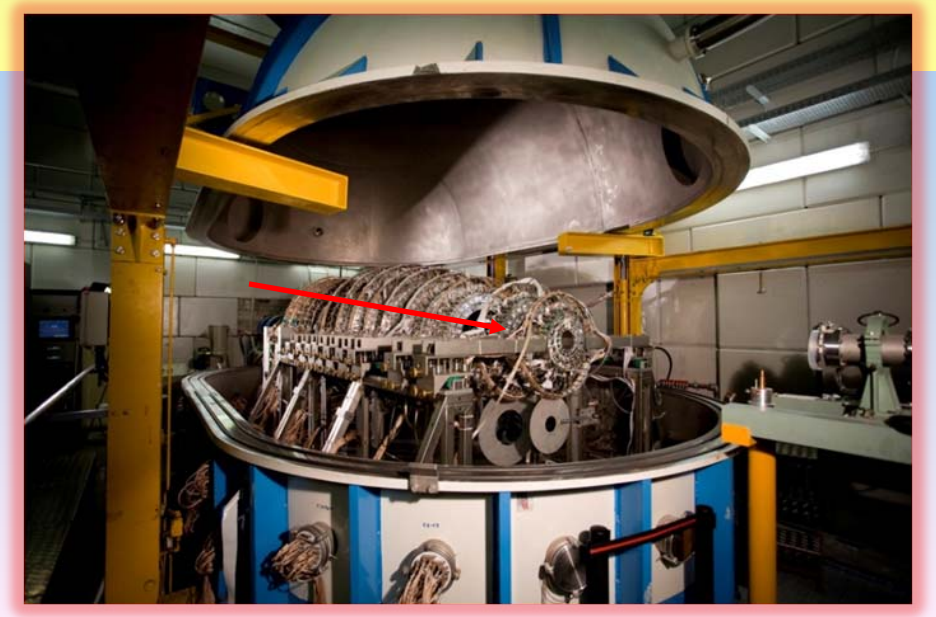
Z identification up to beam charge (ΔE-E)

Z and A identification by ΔE-E up to $Z \leq 9$

Z and A identification in CsI up to $Z \leq 4$

Mass identification with low energy threshold (<0.3 A.MeV) by Time-of-flight

Zeta identification for particles stopping in Si (pulse shape)



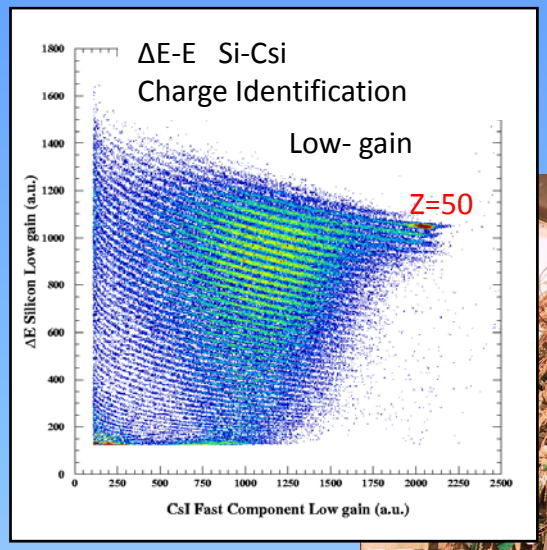
Experimental Methods:

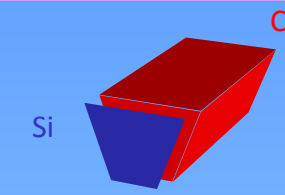
ΔE(Si)-E(CsI(tl)): CHARGE, ISOTOPES

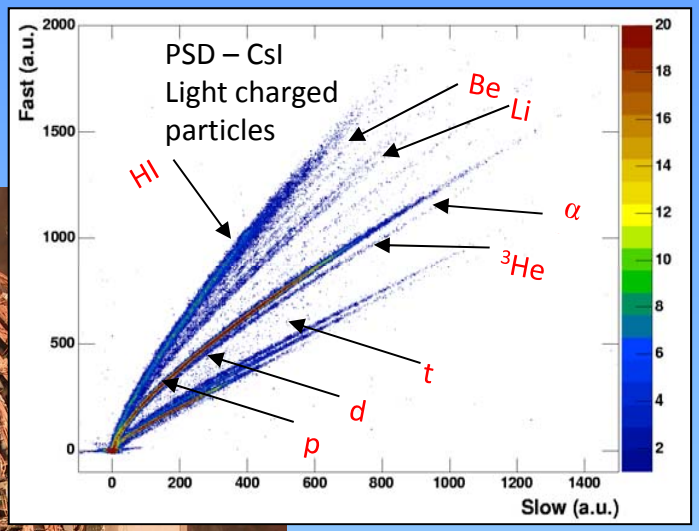
E(Si) –TOF(Si) VELOCITY - MASS

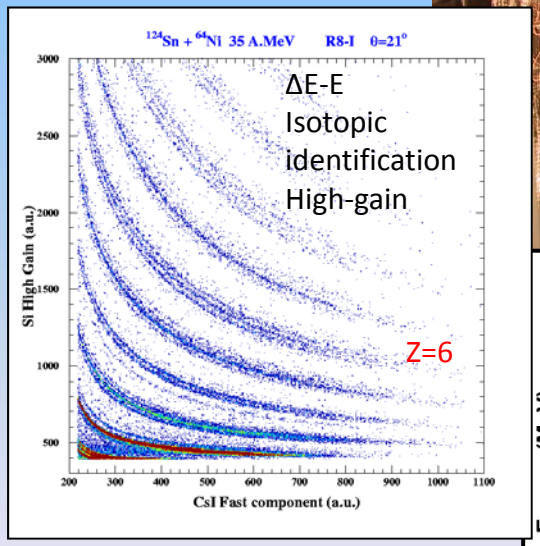
PULSE SHAPE in CsI(Tl)
p,d,t,^{3}He,^{4}He,.$^{6,7,\cdots}$Li,… Z_{light}<5
Pulse shape for particle stopping in Si (CHARGE)

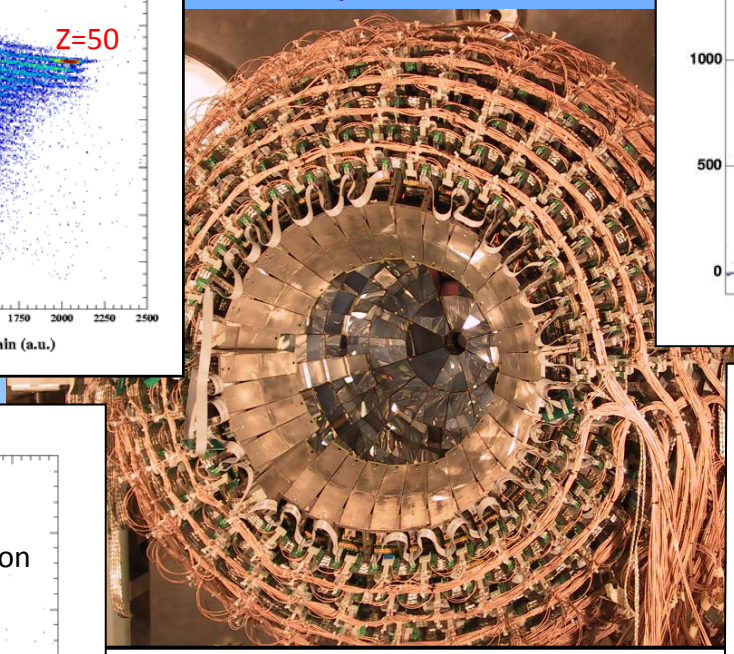
Chimera signals

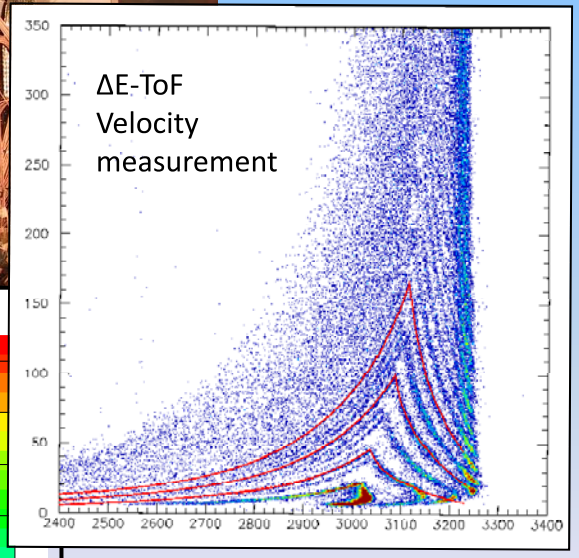


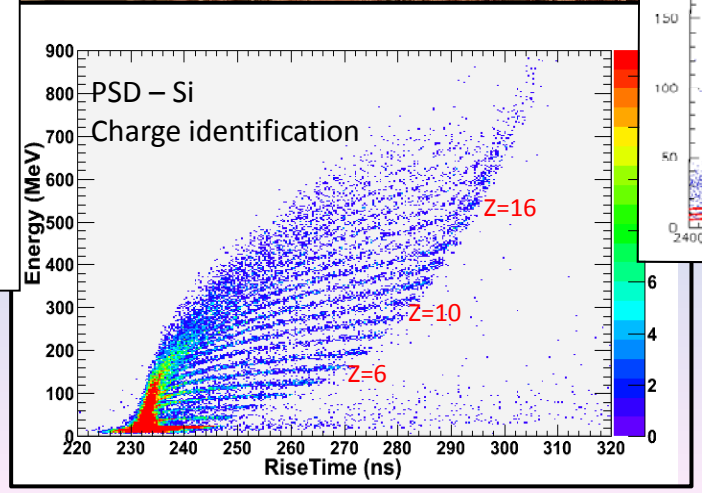




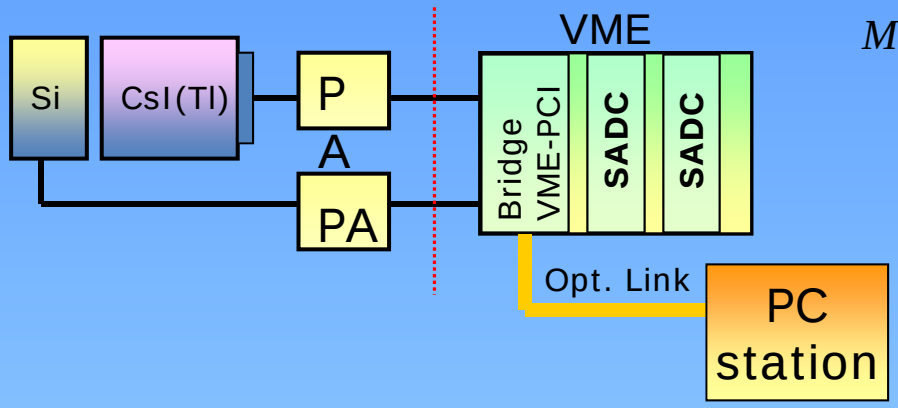








Digital Acquisition on Chimera detectors

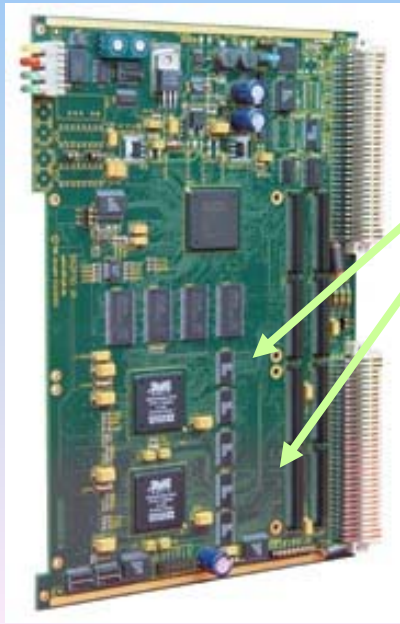


See: M. Alderighi et al., IEEE Trans NS 51, 1475 (2004)
M. Alderighi et al., IEEE Trans NS 53, 279 (2006)

Some advantages:

1) Skipping the electronics front-end.
2) Possibility to implement filters on pulse signal
3) On-line computation of algorithms on the shape of collected pulses

Data shown below have been obtained using a SADC *SIS3150+SIS9300* with 2 DSP 14 BIT, 100 MSample/S, 8 channels (with 2 SIS9300 CMC)



2 DSP – TS101S

BLT32, MBLT64, 2eVME (transfer rate up to 80MB/s)

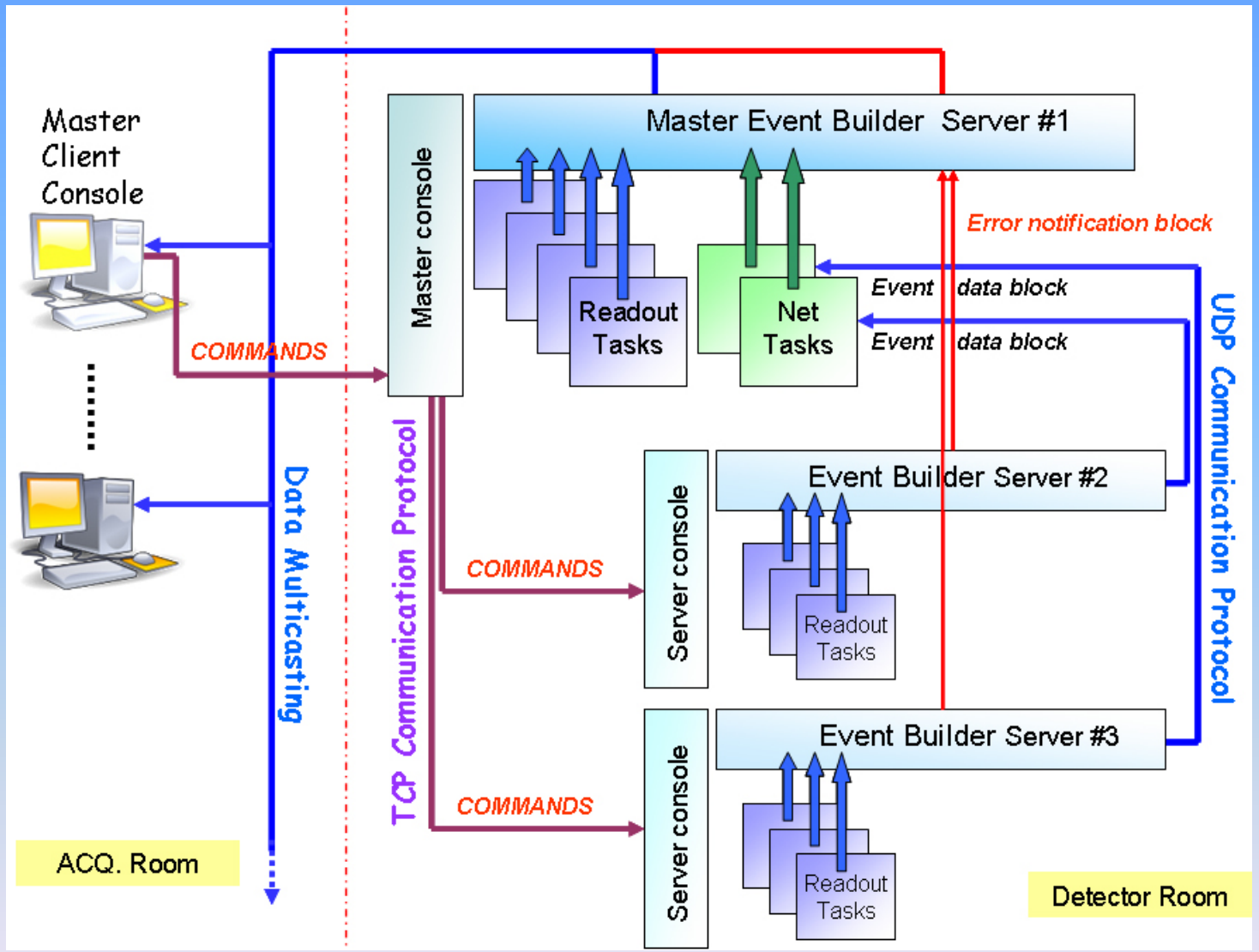
64 MB SDRAM

2 SLOT CMC (for 4+4 channels SIS9300 SADC

See: F. Amorini et al. "On-Board Digital Signal Processing for 4π-Detector Large-Area Telescopes", to be published on: IEEE-TNS Nucl. Sci. Vol 54 Feb. 2007

TS101S TigerSharc

- Superscalar Architecture
- 250 MHz clock
- Up to 4 instructions per clock
- 6 Mbit Internal memory (three 2 Mbit blocks).



Readout is done by a server's cluster (4 nodes). Each one takes care of a part of the detector. The master node is also the general server for event-building.