Trigger/DAQ design:
from test beam to medium size experiments

ISOTDAQ 2020
Escola Tècnica Superior d'Enginyeria
Universitat de València
18 January 2020

Roberto Ferrari
Istituto Nazionale di Fisica Nucleare

Roberto Ferrari
Istituto Nazionale di Fisica Nucleare

Oh my !

Roberto Ferrari
Istituto Nazionale di Fisica Nucleare

Oh my !

Yet another f…[1] Italian ! [2]

Roberto Ferrari
Istituto Nazionale di Fisica Nucleare

Oh my !

Yet another f…[(1)] Italian ! [(2)]

(1) fanatic ... fantastic … ?

(2) about 13.5 lectures (out of 30) covered by Italians

Students' homework

Debate one of the following hypothesis:

Students' homework

Debate one of the following hypothesis:

1) baseline: statistical fluctuation or new physics ?
(to be submitted to Nature)

Students' homework

Debate one of the following hypothesis:

1) baseline: statistical fluctuation or new physics ?
(to be submitted to Nature)

2) romantic: what about "Italians do it better" ?
(to be submitted to Vanity Fair)

Students' homework

Debate one of the following hypothesis:

1) baseline: statistical fluctuation or new physics ?
(to be submitted to Nature)

2) romantic: what about "Italians do it better" ?
(to be submitted to Vanity Fair)

3) last but not least (for the paranoic/complottist ones):

Students' homework

Debate one of the following hypothesis:

1) baseline: statistical fluctuation or new physics ?
(to be submitted to Nature)

2) romantic: what about "Italians do it better" ?
(to be submitted to Vanity Fair)

3) last but not least (for the paranoic/complottist ones):
what about the famous Mafia-Pizza-Spaghetti-TDAQ connection ?
(will go anonymous on the dark web)

On the other hand …

… please, take care !

you can't afford such a demanding environment

you can't afford such a demanding environment
without specific training …

you can't afford such a demanding environment
without specific training …

*… about the **Italians' way***

Luckily

on the web

there are plenty of survival kits

Example 1: basic course (mild concepts)

Example 1: basic course (mild concepts)

A Short Lexicon of Italian Gestures

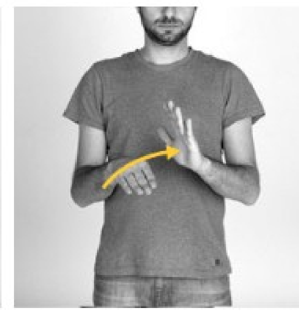
For Italians, it comes naturally. But what do they mean when they talk with their hands? Many things. Roll over the images to learn a few classic gestures. Related Article »

Perfect!

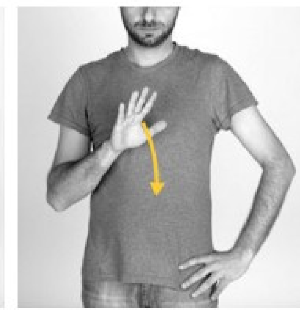
What in God's name are you saying?

Nothing.

Someone talks too much.

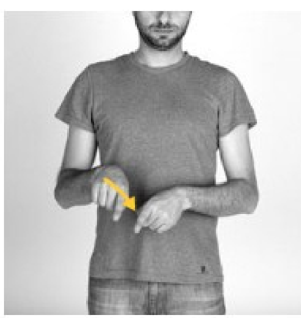


Get out of here.



Slow down or keep calm.

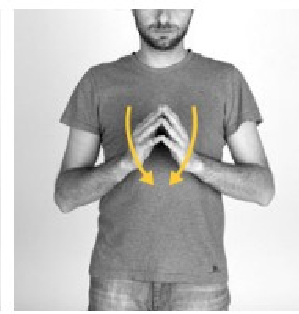
I don't care.



Those two get along.

It wasn't me or I don't know.

Don't worry, I'll take care of it.

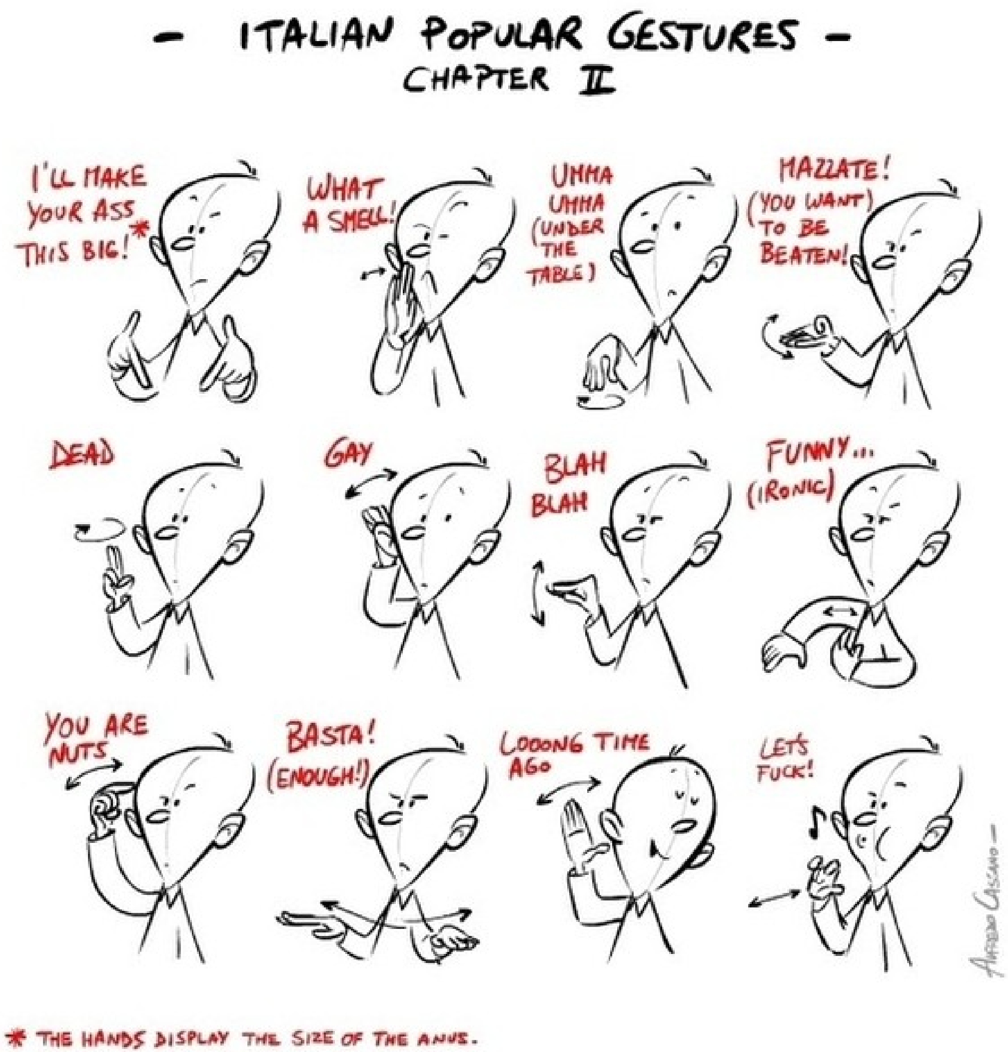


Why in God's name did you/I do it?

To be afraid.

Example 2: advanced course (includes sensitive concepts)

Example 2: advanced course (includes sensitive concepts)



Please take care:

Please take care:

be careful while doing practice!

(expecially for the advanced course)

more seriously ...

(1)

→ hope to give you something sensible ←

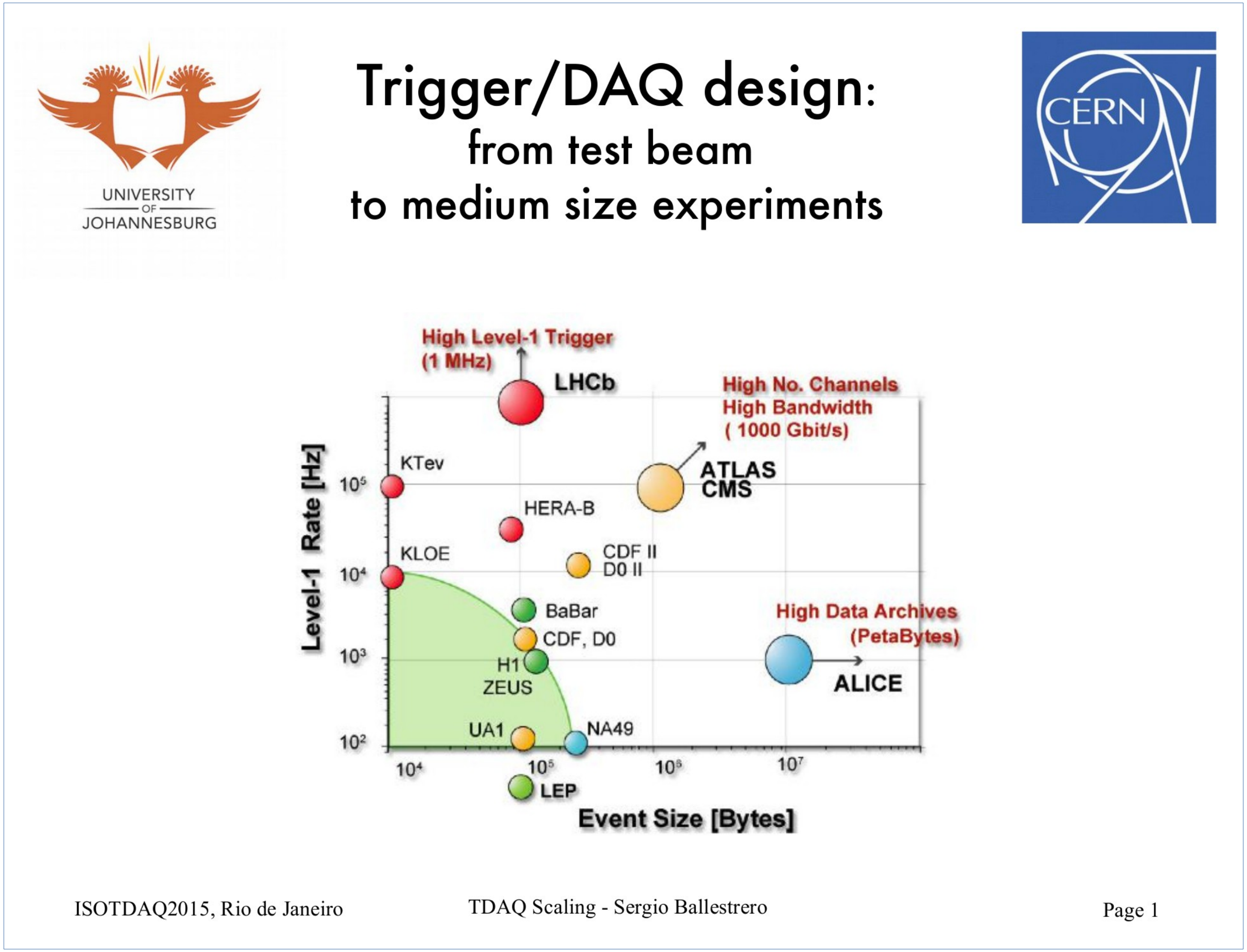
(2)

→ but, please, don’t take anything at face value ←

just aiming at enlightening some critical issues

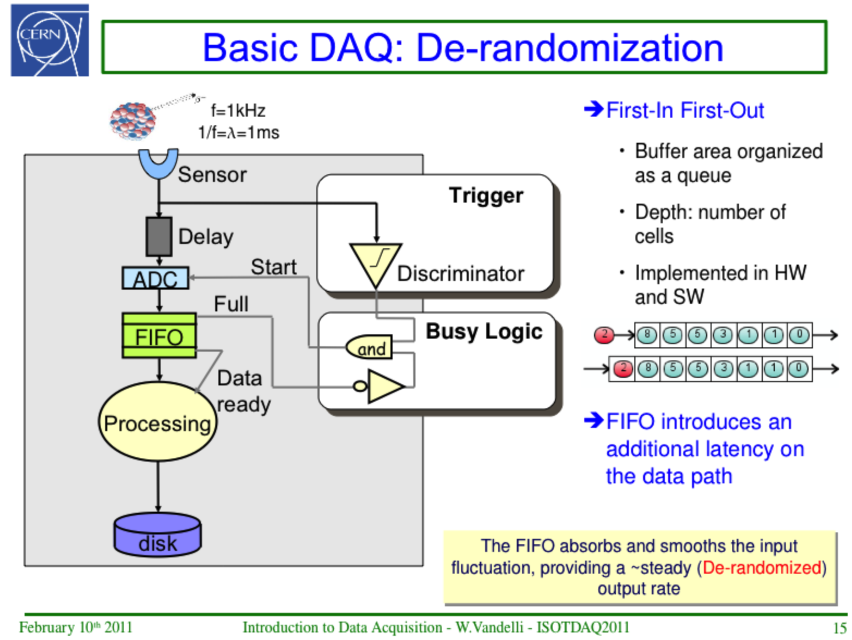
- not meant to be exhaustive (no way!) -

credit to Sergio Ballestrero
most material from his talk at ISOTDAQ 2015

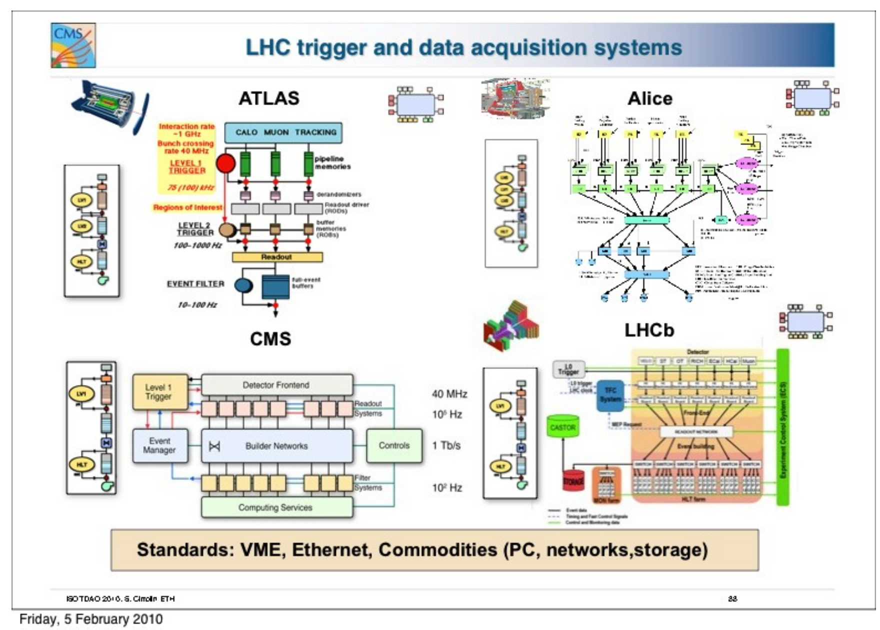


Trying to move …

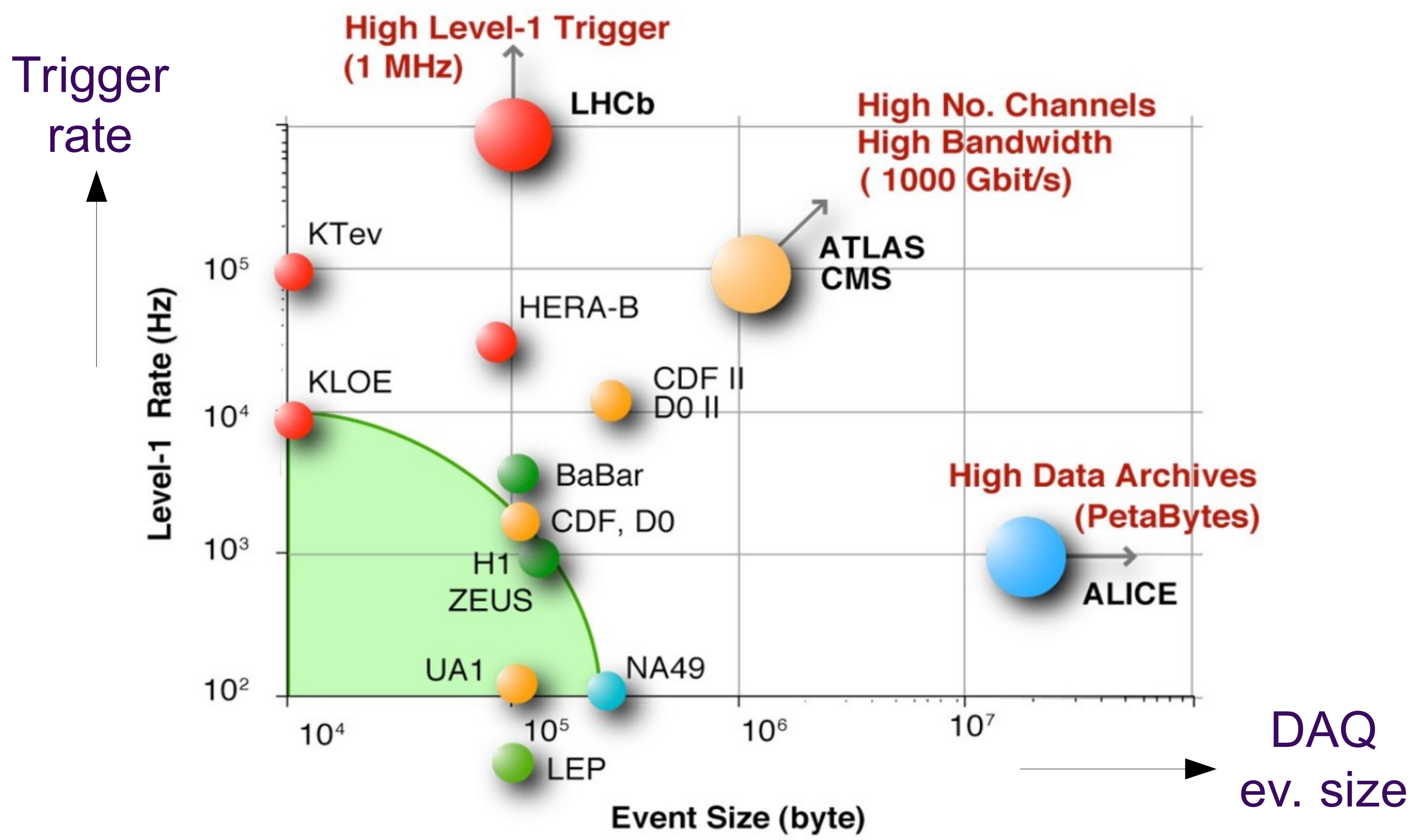
from here:



to here:



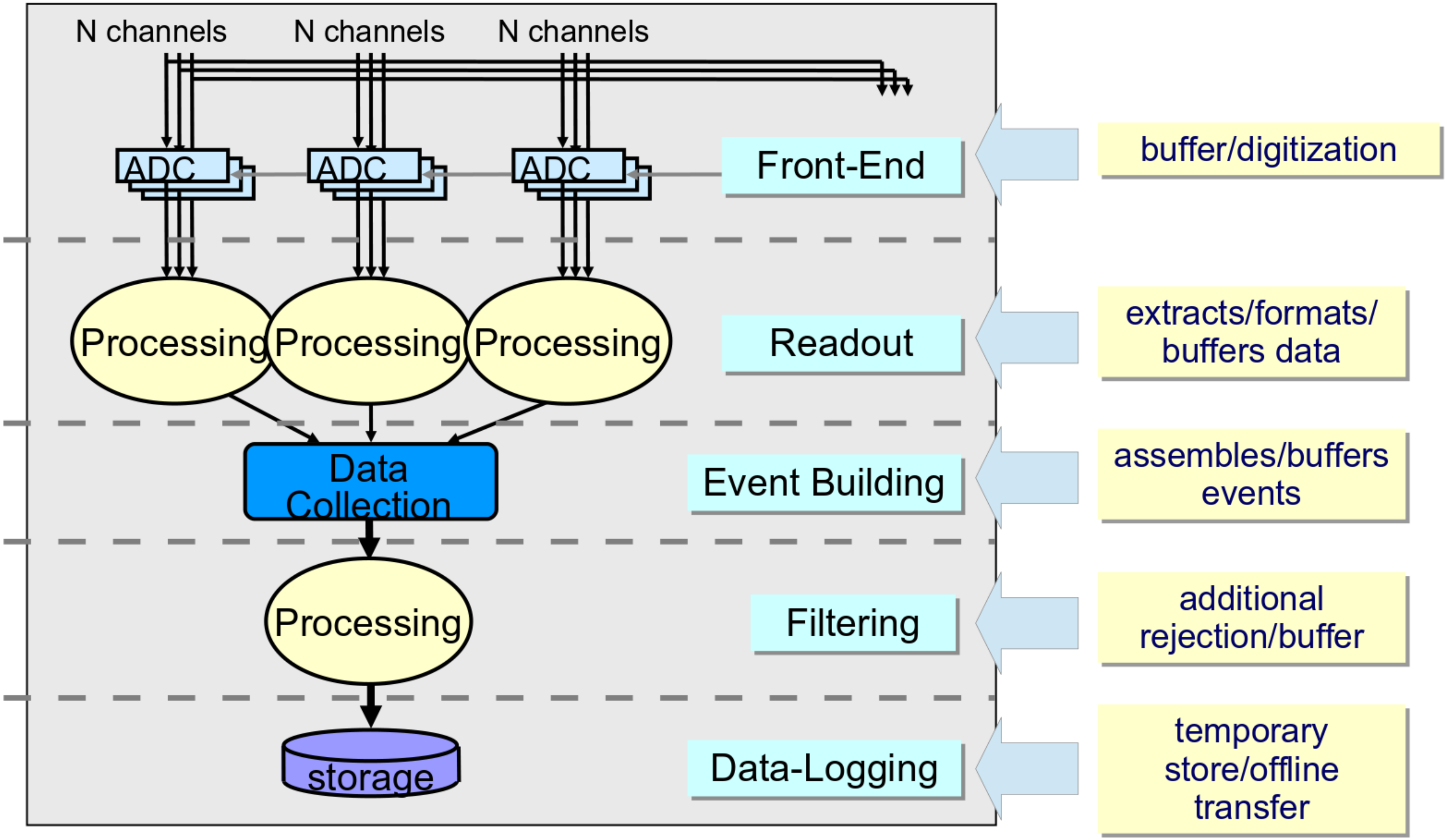
Trigger & DAQ in HEP



different issues → different solutions

no magic, unique solution for all cases

medium/large DAQ: constituents



breakdown into 5 steps ...

- Step 1: Increasing rate
- Step 2: Increasing sensors
- Step 3: Multiple front-ends
- Step 4: Multi-level trigger
- Step 5: Data-flow control

back to square one

A minimal system: what do we need ?

back to square one

Do we really need a trigger ?

back to square one

Do we really need a trigger ?

not obvious … triggerless DAQ systems do exist

back to square one

Do we really need a trigger ?

not obvious … triggerless DAQ systems do exist

even in HEP experiments

back to square one

Do we really need a trigger ?

not obvious … triggerless DAQ systems do exist

even in HEP experiments

e.g.:

a) LHCb upgrade 40 MHz readout

b) DUNE LAr TPC 2 MHz readout

→ *Alessandro's talk on Jan 15th*

but … in most cases, triggering is crucial !

how trigger is born

how trigger is born

Walther Bothe (1924-1929):

offline ⟶ online coincidence (logic **AND**) of 2 signals

Bruno Rossi (Nature, 1930):

"Method of Registering Multiple Simultaneous Impulses of Several Geiger Counters"

→ online coincidence of 3 signals (scalable)!

first modern trigger

https://en.wikipedia.org/wiki/Coincidence_circuit :

"Rossi coincidence circuit was rapidly adopted by experimenters around the world. It was the first practical AND circuit, precursor of the AND logic circuits of electronic computers"

Geiger-Muller counters

Rossi's circuit: coincidence of signals of 3 Geiger-Muller counters

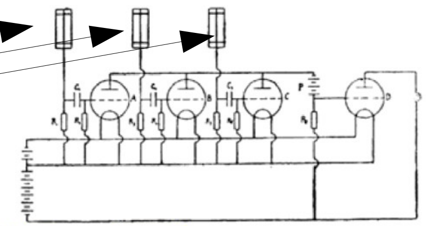


Fig. 17 – Il circuito di Rossi per rivelare coincidenze di raggi cosmici che arrivano sui contatori Geiger (i rettangoli in alto dello schema)[19].

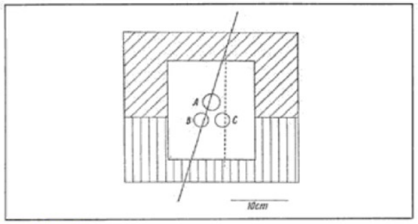


Fig. 18 – L'uso del circuito di Rossi per rivelare una coincidenza tripla che, nella disposizione in figura dei tre contatori, mostra la produzione di una radiazione secondaria (linea tratteggiata) da parte della radiazione primaria (linea continua)[20].

simplest case: 2-signal coincidence

a simple trigger system

Gokhan's talk:

$$N1 = s1 \cdot s2 \cdot \overline{s3}$$

Veto (anti-coincidence)

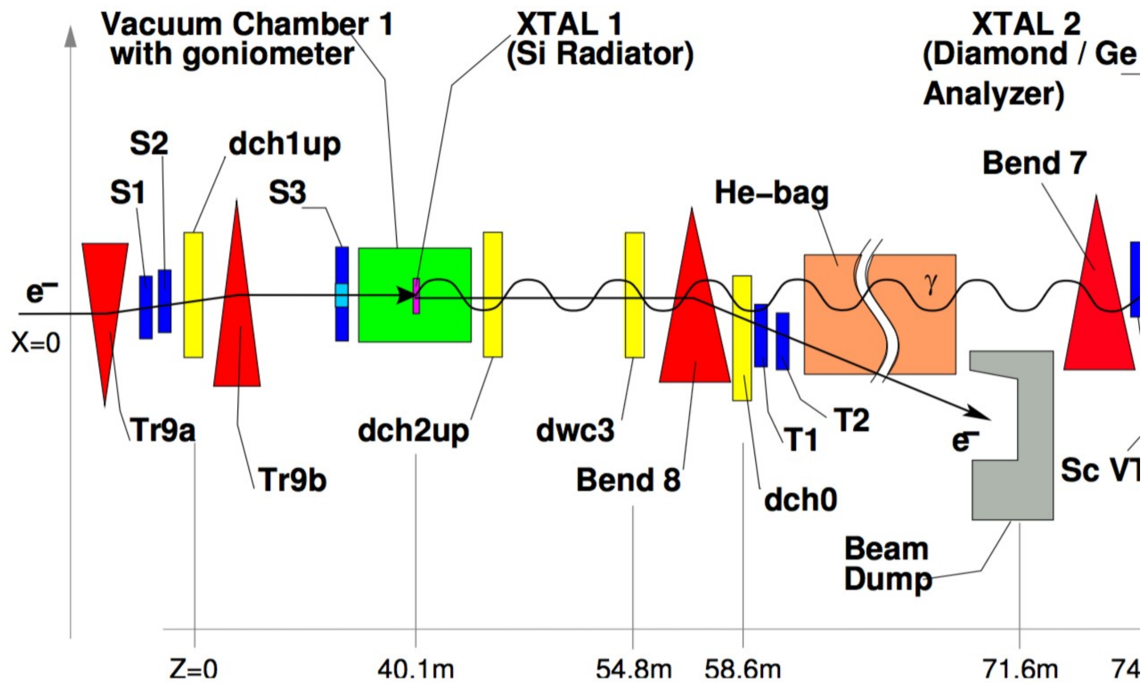
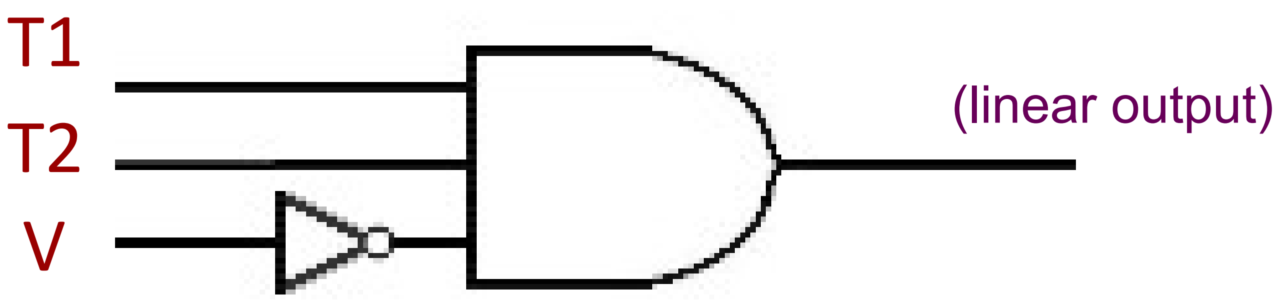


Fig. 1. Setup of the Na59 Experiment

any issue ?

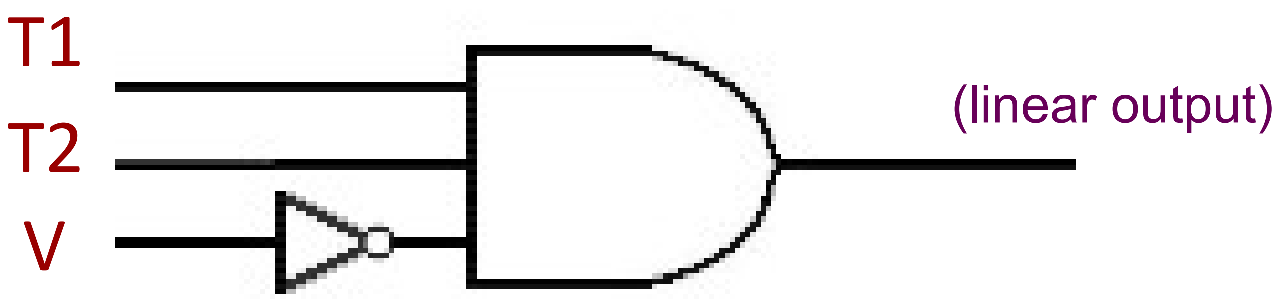
any issue ?



(anti-)coincidence with veto

→ easy !

any issue ?



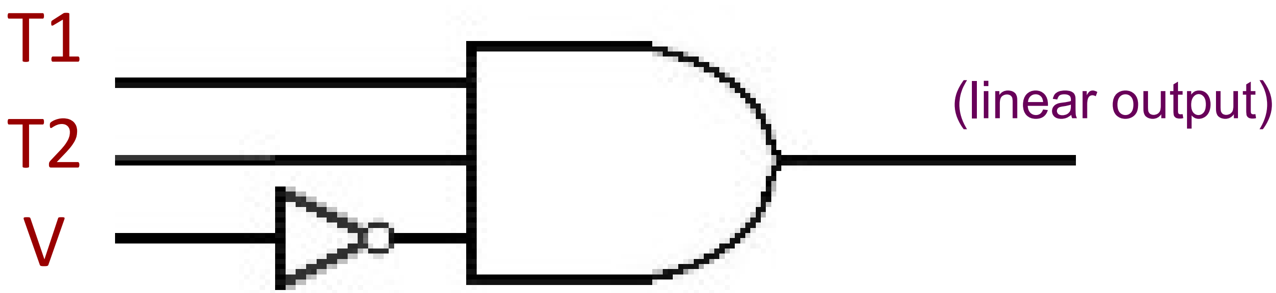
(anti-)coincidence with veto

→ easy !

does it work ?

aahh !

(anti-)coincidence with veto



flawed !

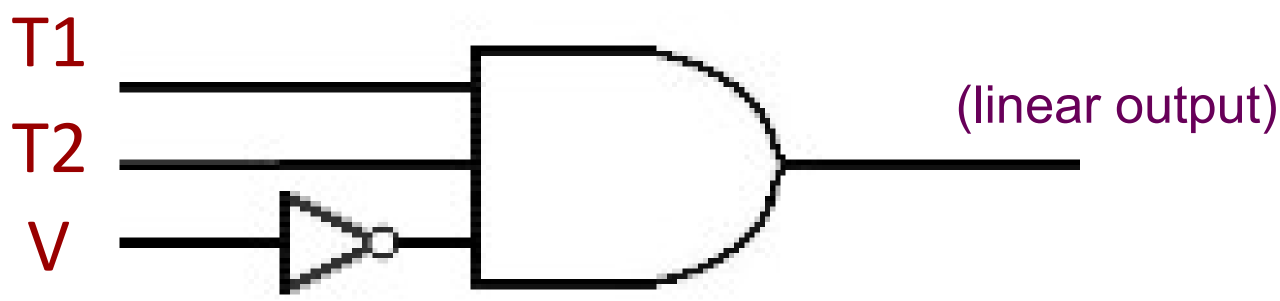
output signal does:

a) jitter

b) fluctuate in duration

why ?

(anti-)coincidence with veto



flawed !

output signal does:

a) jitter

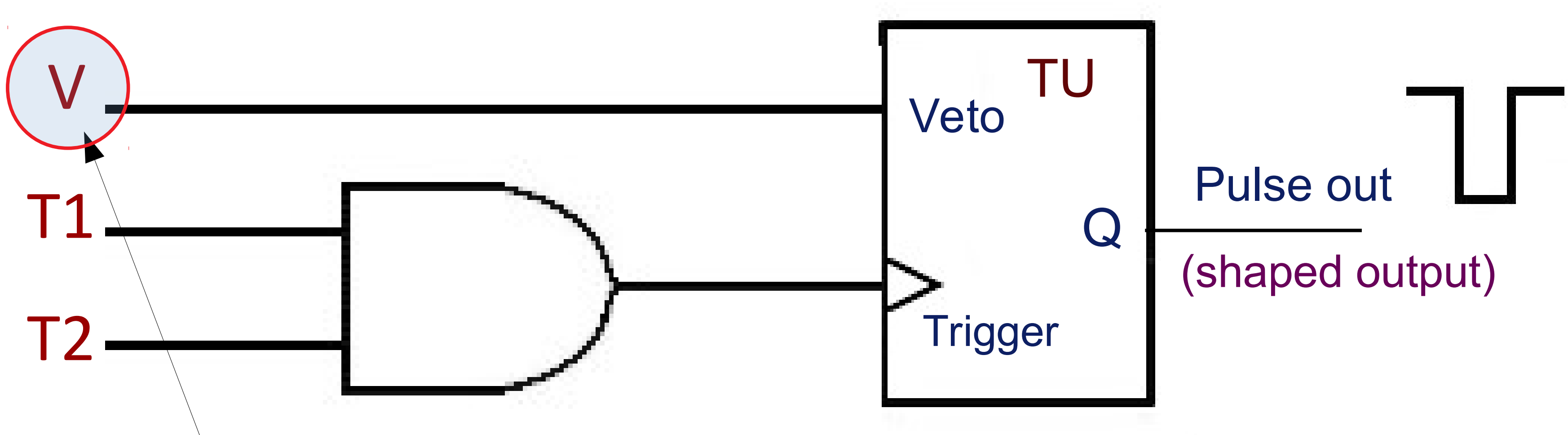
b) fluctuate in duration

because of independent signals from T1, T2, V

Shit !

Forgot noise, pileup …

(anti-)coincidence with veto



can also be a busy signal → busy logic

TU = Timing Unit

= Monostable Multivibrator

= One-Shot Pulse Generator

much better !

first lesson(s)

trigger signal:

1) should be formed

$\longrightarrow$ pulse with predefined duration

2) veto/busy should block pulse generation

3) need both combinatorial (AND, OR, NOT) and sequential logic (TU, FF)

step one: increase rate

Many issues:

→ trigger latency

→ readout latency

→ throughput

→ rate fluctuations (trigger bursts)

→ throughput fluctuations
(correlated noise, …)

step one: increase rate

Many issues:

→ trigger latency

→ readout latency

→ throughput

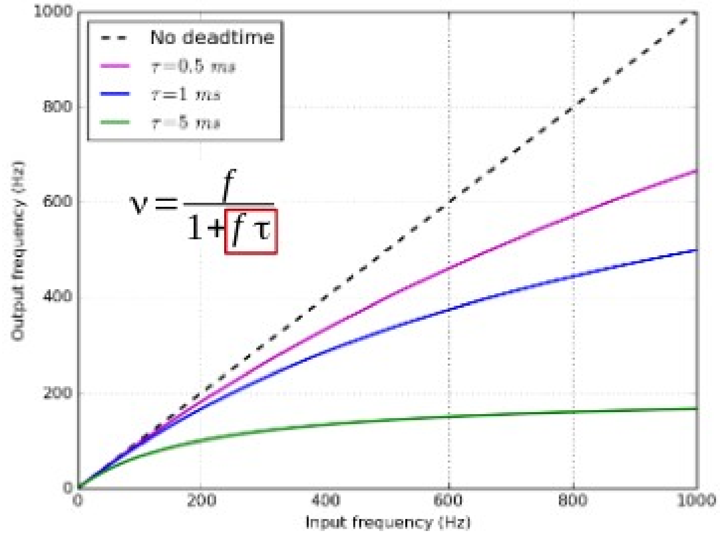
→ rate fluctuations (trigger bursts)

→ throughput fluctuations
(correlated noise, …)

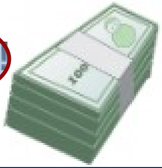
→ dead-time

dead time (from Andrea's introduction)

Deadtime and efficiency



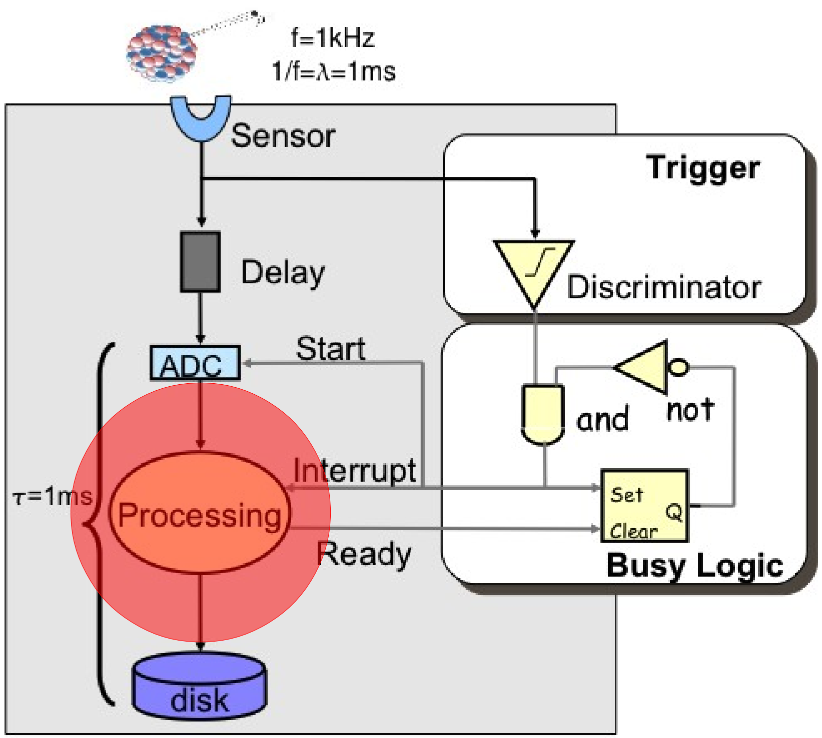
- In order to obtain ε~100% (i.e.: ν~f) $\rightarrow f\tau << 1 \rightarrow \tau << \lambda$
  - E.g.: ε~99% for f = 1 kHz $\rightarrow$ τ < 0.01 ms $\rightarrow$ 1/τ > 100 kHz
  - To cope with the input signal fluctuations, we have to **over-design** our DAQ system **by a factor 100!**



- How can we mitigate this effect?

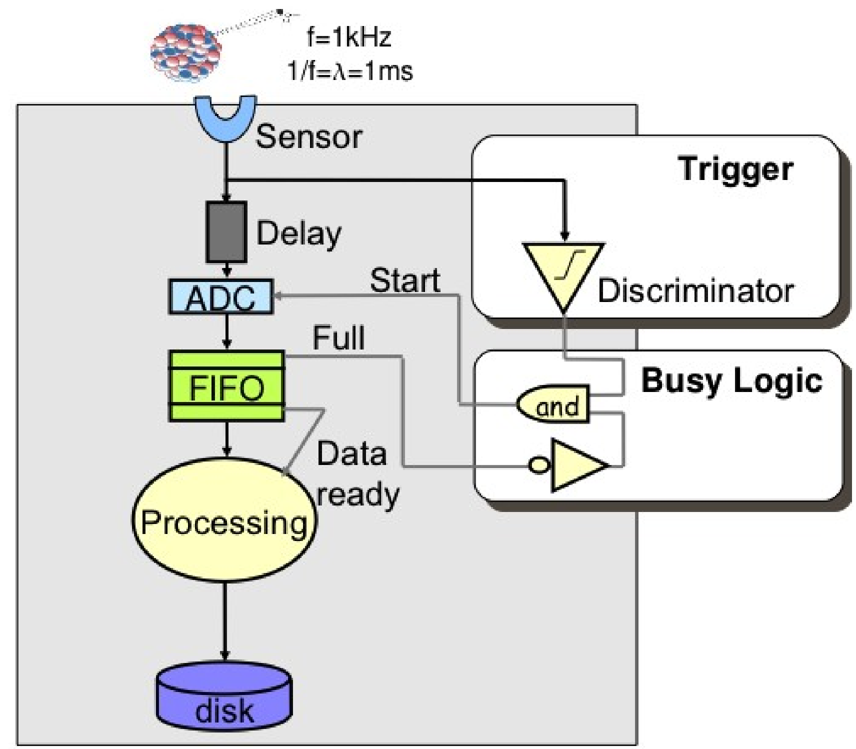
dead time → de-randomise

- Processing → bottleneck ?



(f·τ) ~ 1 → dead time ~ 50%

- Buffering → decouple problems



What the impact ?

(f·τ) ~ 1 → dead time ?

FIFO

First-In First-Out memory:

1) independent read/write (sequential) access

2) may be hardware or over RAM

if RAM better Dual-Port RAM

buffering solve all problems ?

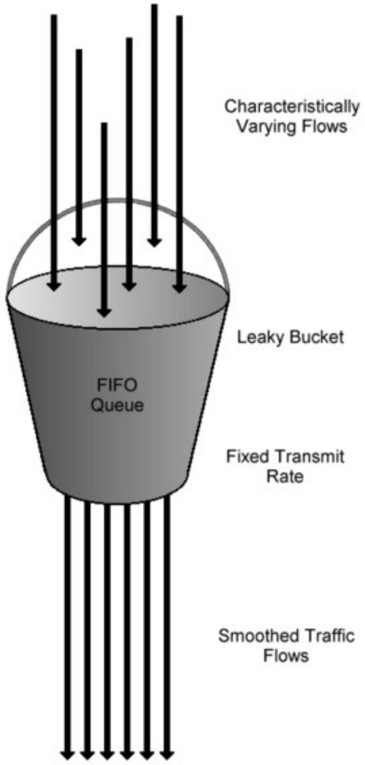
- FIFO

  1) filling at very variable input flow

  2) emptying at smoothed output flow

  $\rightarrow$ the Leaky-Bucket problem

  Q: how often may overflow ?



off-topic: some very basic queueing theory

- N-event buffer ... single queue size N:

P_k : % time with k events in ; P_N = no space available → dead time

$\sum P_k = 1$ [k=0..N]

rate [j→j+1] = $\lambda \cdot P_j$ (fill at rate λ)

rate [j+1→j] = $\mu \cdot P_{j+1}$ (empty at rate μ > λ)

steady state: $\mu \cdot P_{j+1} = \lambda \cdot P_j \Rightarrow P_{j+1} = \rho \cdot P_j = \rho^{j+1} \cdot P_0$ [$\rho = (\lambda/\mu) <\sim 1$]

for $\rho \sim 1 \Rightarrow P_j \sim P_{j+1} \Rightarrow \sum P_k \sim (N+1) \cdot P_0 = 1 \Rightarrow P_0 \sim P_N \sim 1/(N+1)$

$\Rightarrow$ dead time ~ 1/(N+1)

want ~ 1% ⇒ N ~ 100

off-topic: some very basic queueing theory

- N-event buffer ... single queue size N:

P_k : % time with k events in ; P_N = no space available → dead ti

$\sum P_k = 1$ [k=0..N]

rate [j→j+1] = $\lambda \cdot P_j$ (fill at rate

rate [j+1→j] = $\mu \cdot P_{j+1}$ (e rate $\mu > \lambda$)

steady state: $\mu \cdot P_{j+1} = \lambda$ $P_{j+1} = \rho \cdot P_j = \rho^{j+1} \cdot P_0$ [$\rho = (\lambda/\mu)$ <~1]

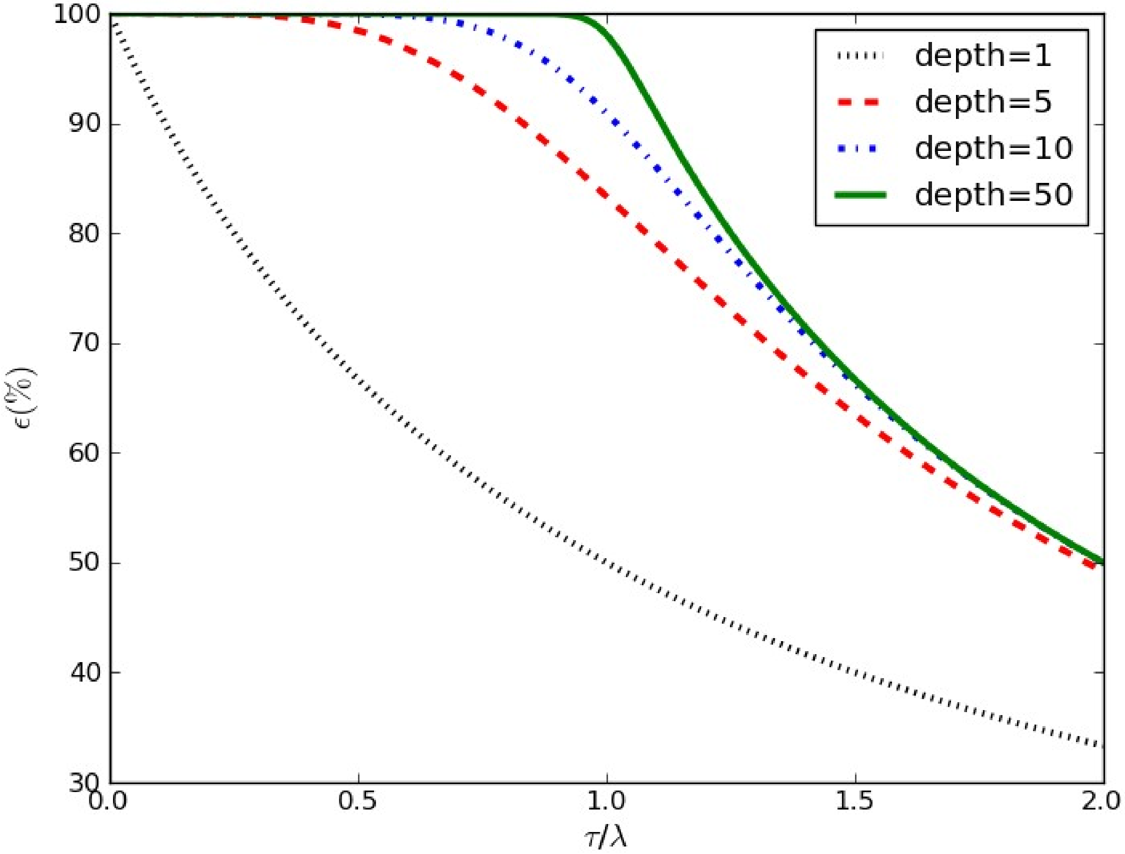
Take care: analytic calculation possible for pretty simple systems only

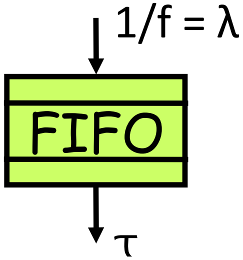
for $P_j \sim P_{j+1}$ ⇒ $\sum P_k \sim (N+1) \cdot P_0 = 1$ ⇒ $P_0 \sim P_N \sim 1/(N+1)$

⇒ dead time ~ 1/(N+1)

want ~ 1% ⇒ N ~ 100

de-randomisation





- DAQ ε ~100% with:
  - τ ~ 1/f
  - “moderate” buffer size
- Two degrees of freedom to play with
- This dead time often managed by trigger system itself (“complex dead time”)

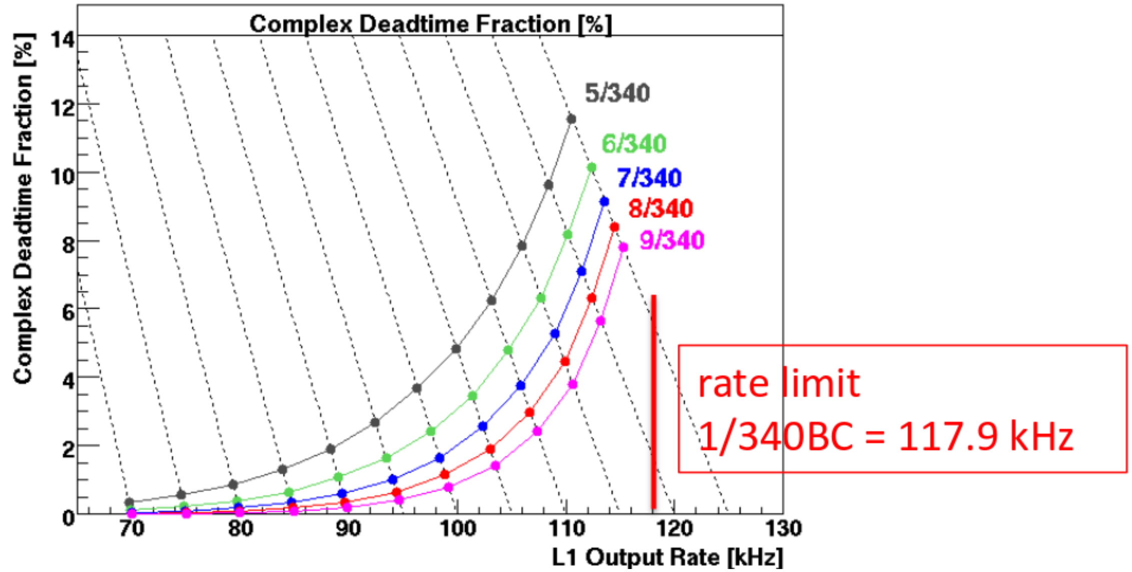
complex dead time

1) Simple dead time: avoid overlapping (conflicting) readout window

2) Complex dead time: avoid overflow in front-end buffers (protection against trigger bursts)

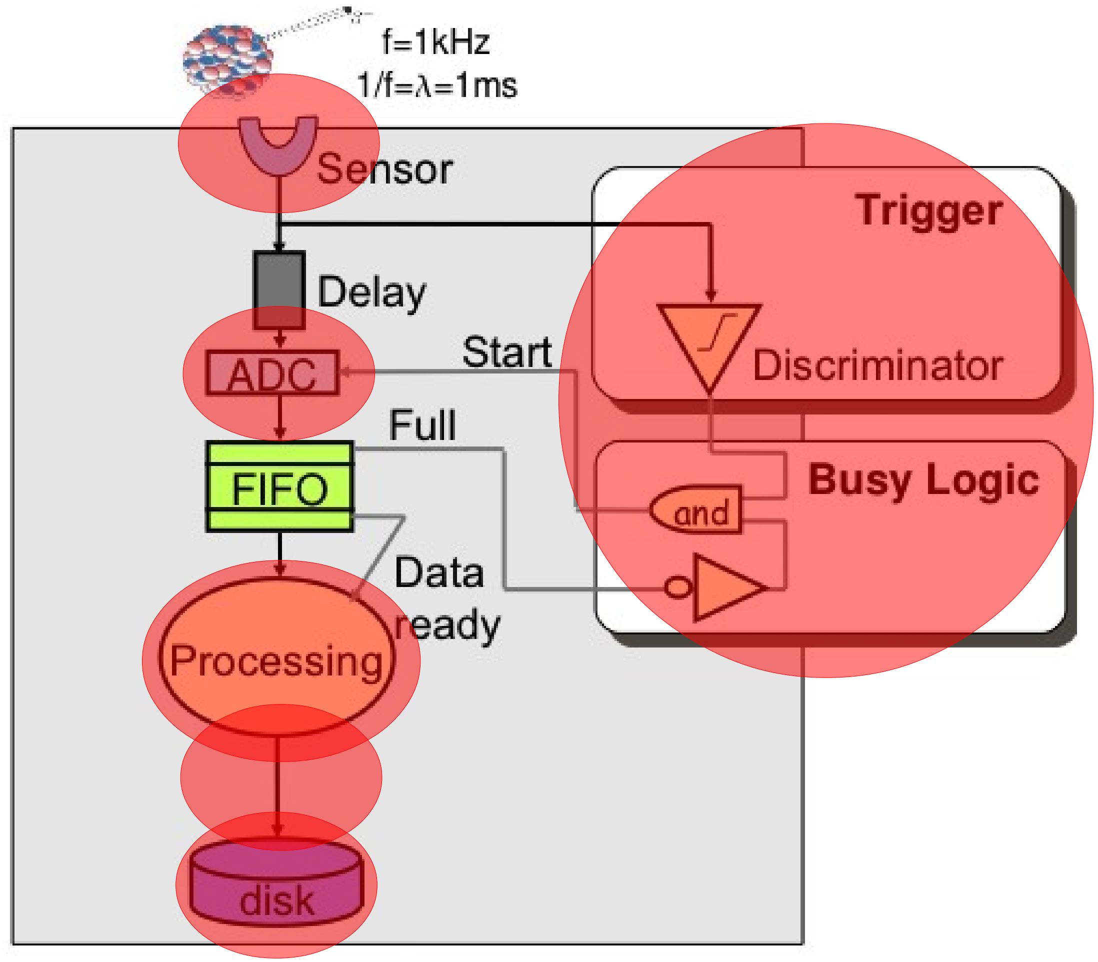
e.g. ATLAS uses simple leaky-bucket algorithms with 2 parameters:

max N triggers (N = FIFO depth) in any (sliding) time window = (N × readout time)



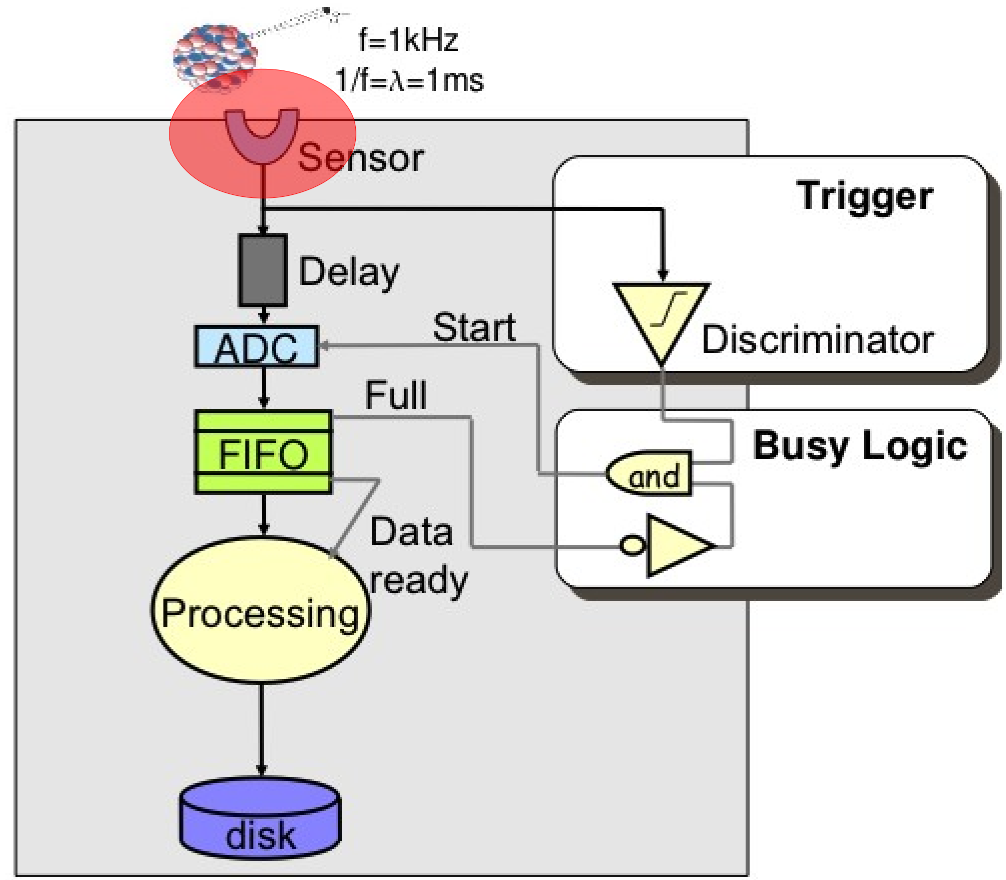
game over ?

many other possible limits even in a simple DAQ



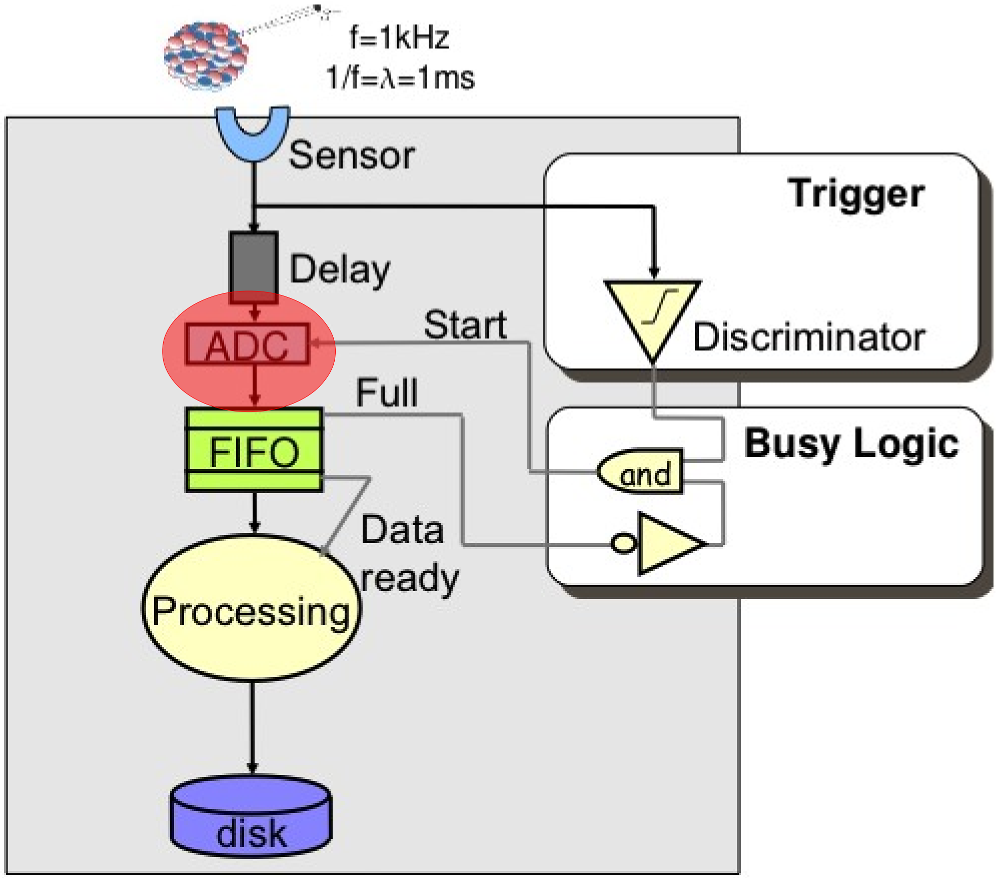
→ sensor

- Sensors limited by physical processes such as:
  - drift times in gases
  - charge collection in Si
- (possibly) choose fast processes
- analog FE imposes limits as well
- split sensors, each gets less rate: "increase granularity"



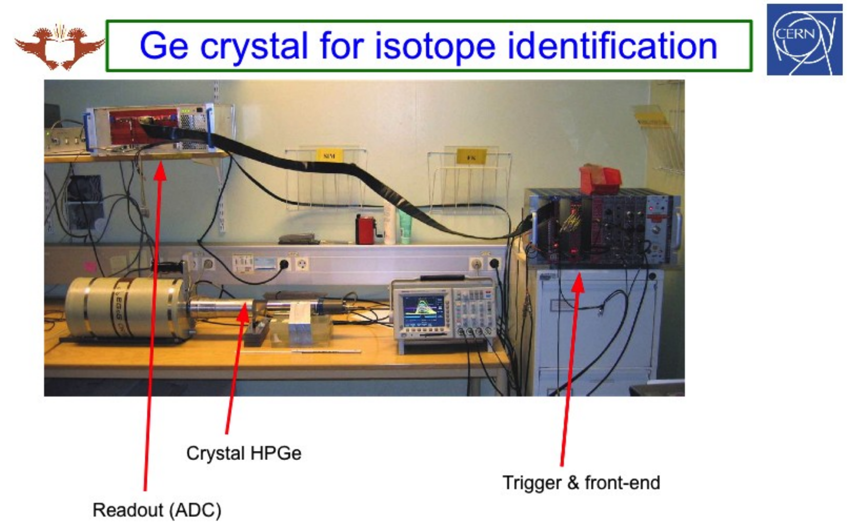
→ ADC

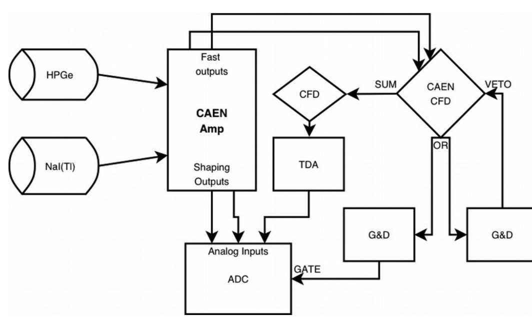
- A/D conversion also limited
- Fast ADC
  - → # of bits (resolution)
  - → power consumption
- Alternatives:
  - analog buffers
  - (e.g. switched capacitor arrays)
- You may need integration (or sampling) over quite some time

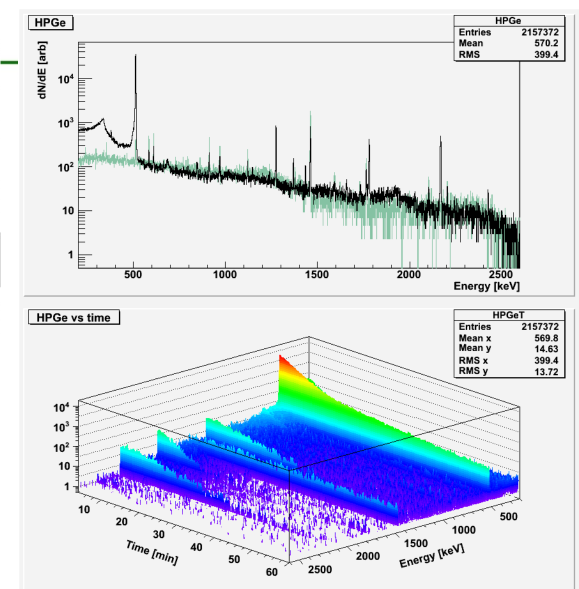


an example

- HPGe + NaI Scintillator
  High res spectroscopy and beta+ decay identification
- minimal trigger with busy logic
- Peak ADC with buffering, zero suppression
- VME SBC with local storage
- Root for monitor & storage
- Rate limit ~14 kHz
  - HPGe signal shaping for charge collection
  - PADC conversion time

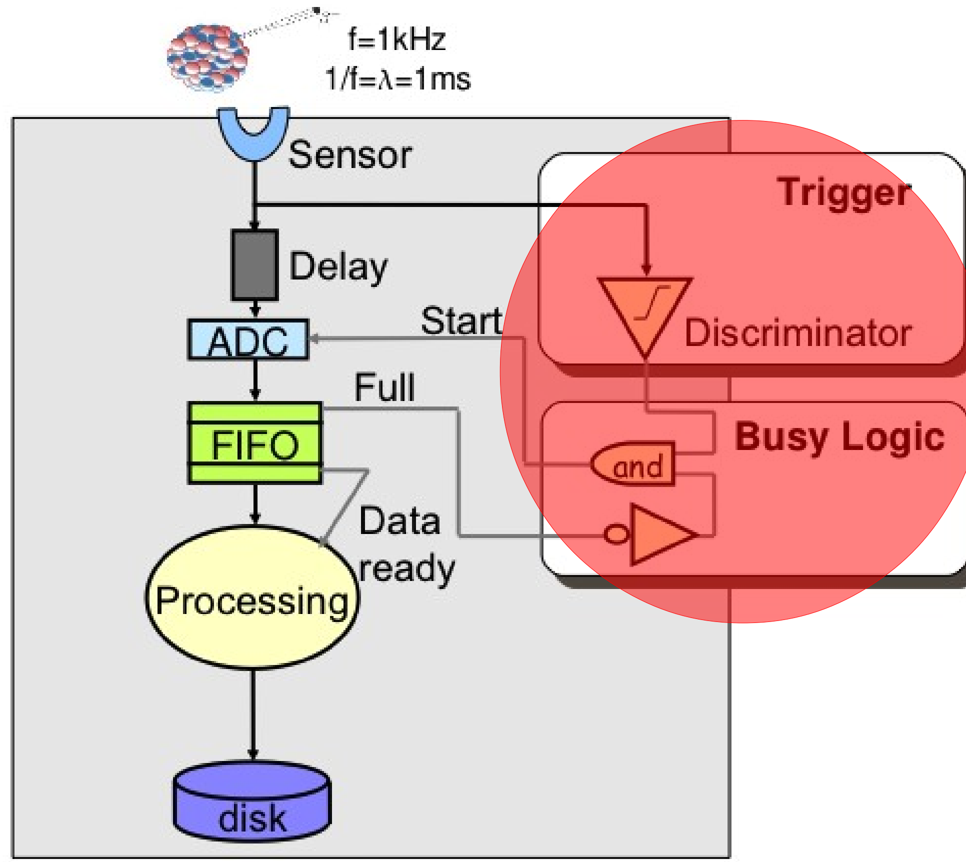






→ trigger latency

- simple trigger: ~fast
- complex trigger logic: not obvious [even when all in hw]
- some trigger detectors may be far away / slow → latency
- trigger signal is one: all information at a single point
  - in one step: too many cables
  - in many steps: delays



→ discrete modules: ~ 5-10 ns delay → tot. latency ≥ 20-30 ns ←

DREAM/RD52 (2006→): a testbeam case

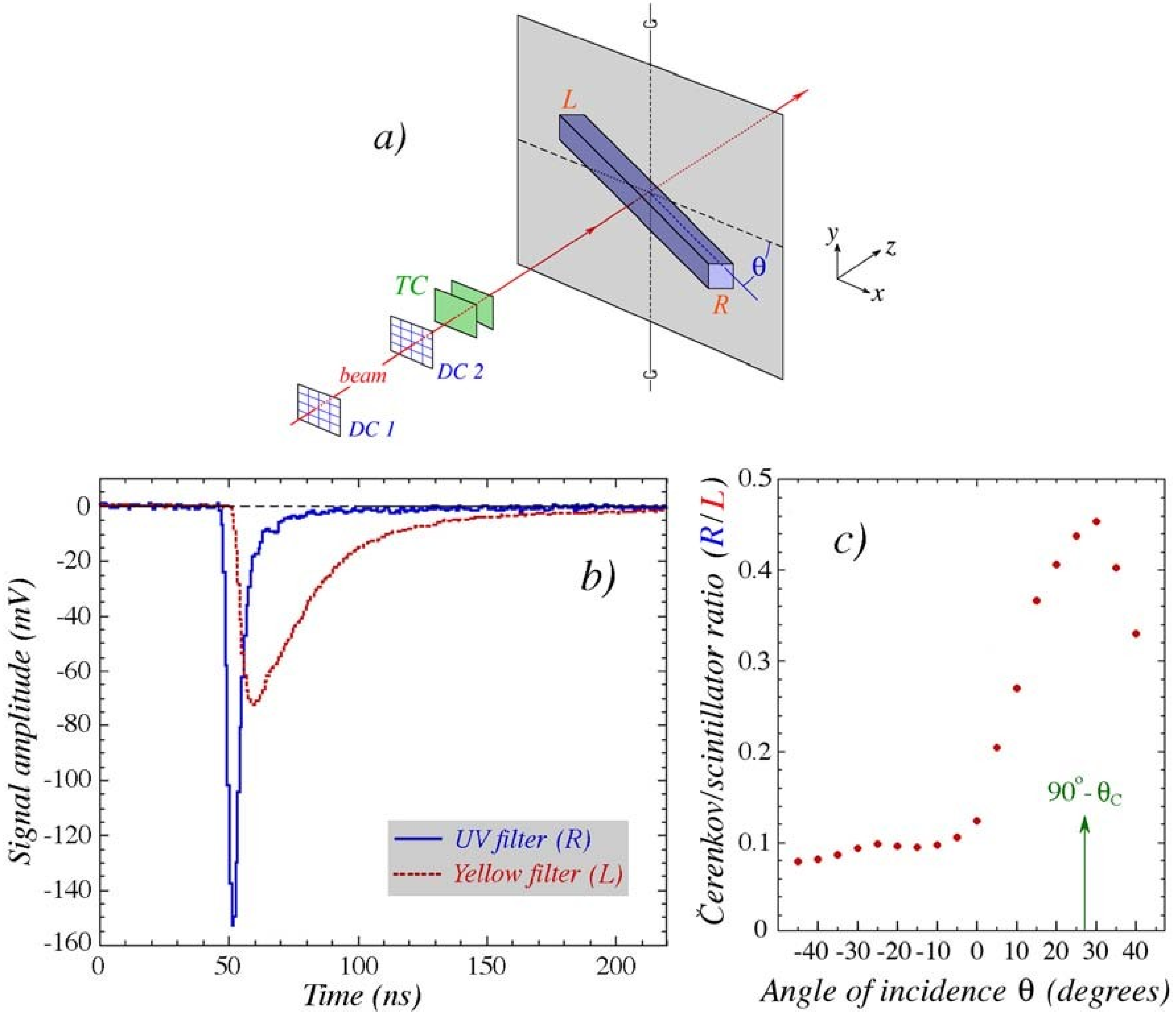
R&D on dual-readout calorimetry, setup:

- Crystals
- Scintillating/cherenkov fibers in lead/copper matrices
- Scintillator arrays as shower leakage counters
- Trigger/veto/muon counters
- Precision chamber hodoscope

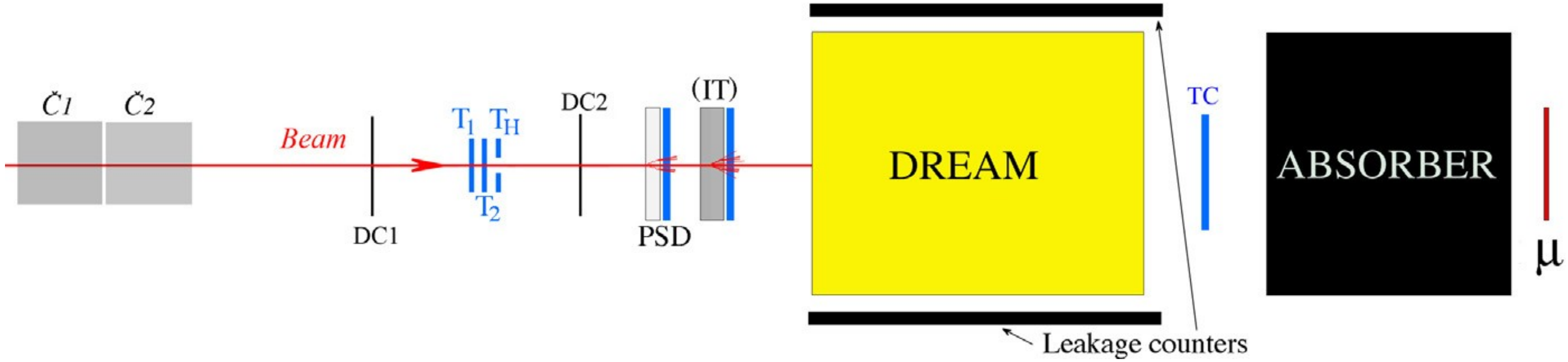
… always evolving

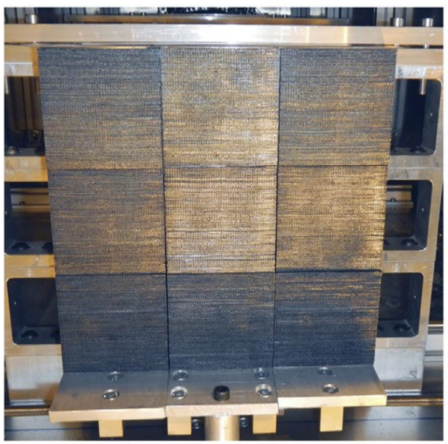
Acquiring: waveforms, total charge, time information

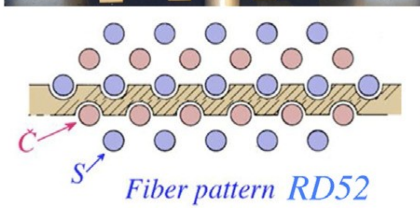
DREAM/RD52: crystal prototype

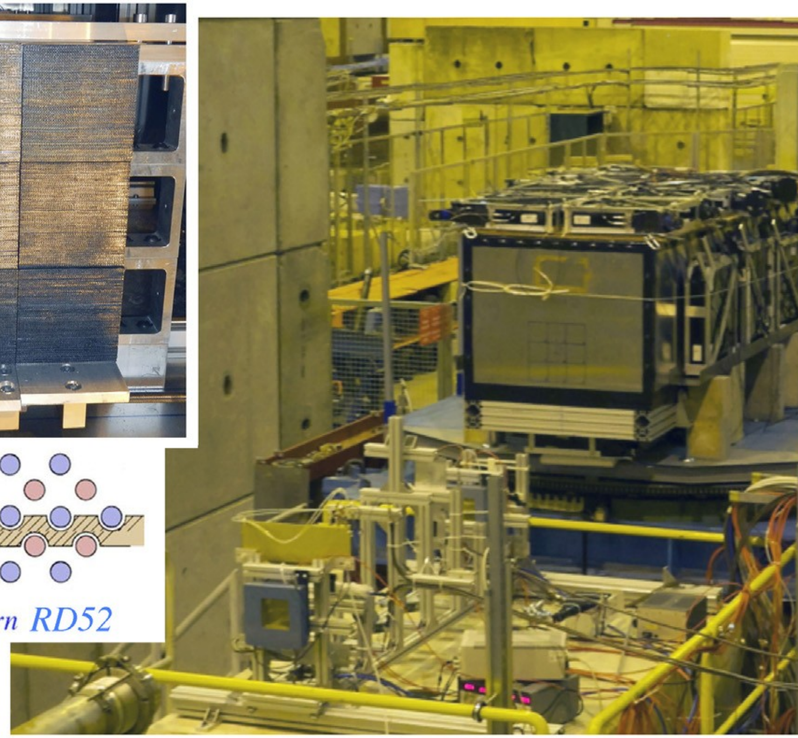


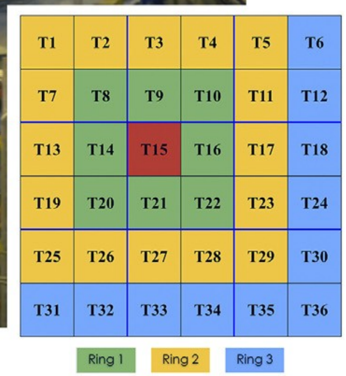
DREAM/RD52: fibre-sampling prototype











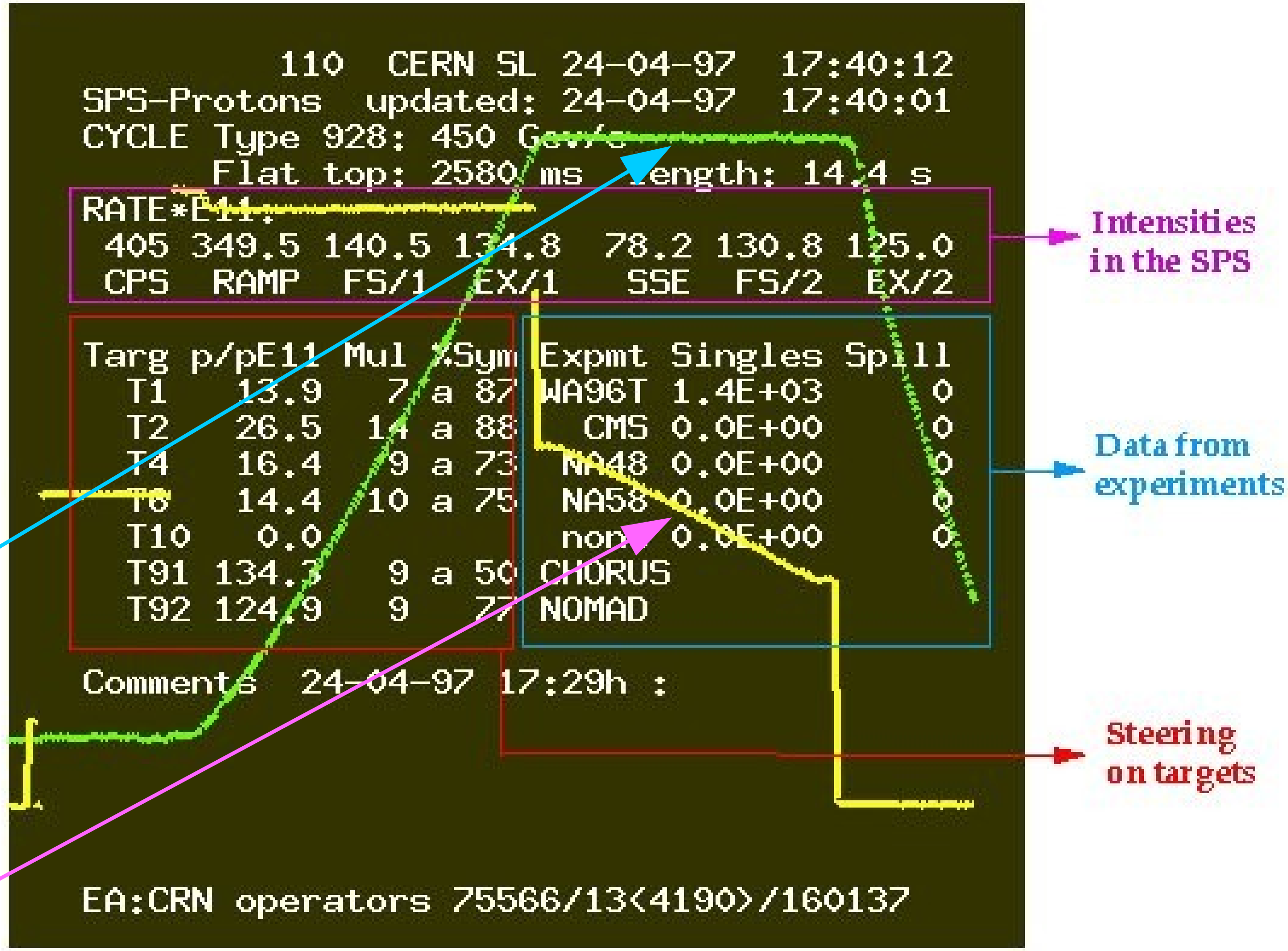
DREAM (2006→): a testbeam case

a possible
SPS cycle

duty cycle:

~2 s / 14.4 s

(flat top)



flat top

slow extraction

Trigger = ($\overline{V}$×T_1×T_2 | ped) → easy !

readout system

1 PC $\rightarrow$ readout of 2 VME crates (via CAEN optical interfaces)
1 PC $\rightarrow$ storage

6 × 32 ch QDCs + TDCs $\rightarrow$ CAEN V792, V862, V775

1 × 34 ch (5 Gs/s) digitizer $\rightarrow$ CAEN V1742
(single event: $\sim$34×1024×12bit)

1 × 4 ch (20 Gs/s) oscilloscope $\rightarrow$ Tektronix TDS 7254B

... few VME I/O & discriminator boards

... all in the control room

dataflow

1) Pull mode → FE electronics waiting for PC readout
(self-blocking trigger, re-enabled after readout)

2) Block data transfer → DMA (Direct Memory Access)
data moved by specialised hw (not by CPU)

[Push mode → FE electronics sending data as soon as available]

DAQ

DAQ logic spill-driven (no “real time”, SLC desktops)

in-spill (slow extraction)

poll trigger signal … if trigger present:

a) (block) read all VME boards
b) format & store on large buffers (FIFO over RAM)
c) re-enable trigger

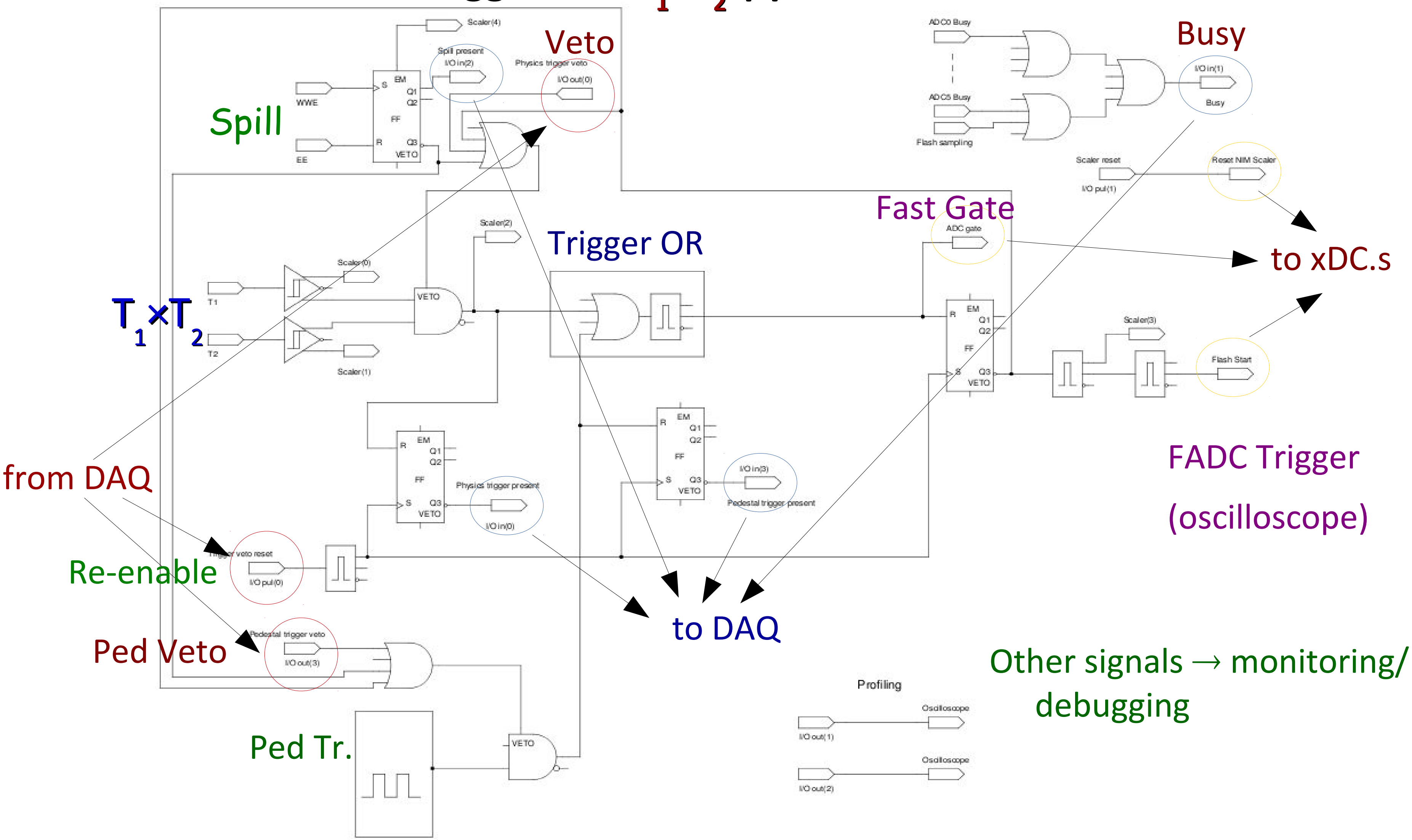
out-of-spill

a) read scope (in case) → event size fixed at run start
b.1) flush buffers to disk (beam and pedestal files) over network
b.2) monitor data (produce root files)

rate ~ O(1 kHz) limited by DAQ readout

spill-driven (asynchronous) trigger

$$\text{Trigger} = \overline{V} \times T_1 \times T_2 \mid \text{ped}$$



trigger system

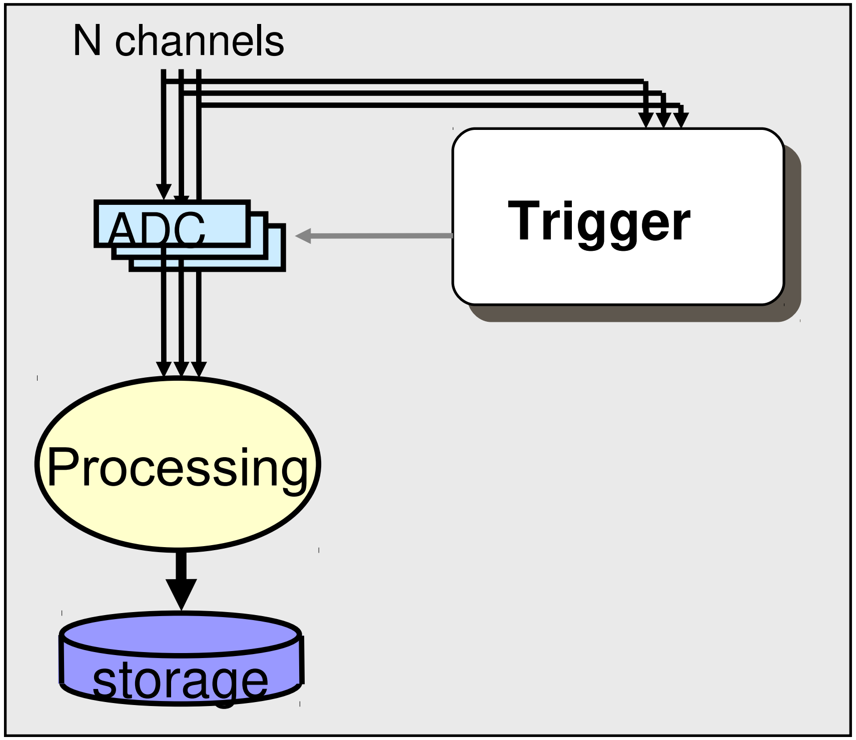
a) crystals w/ fast PMT.s
b) no analog buffering

$\rightarrow$ low-latency trigger

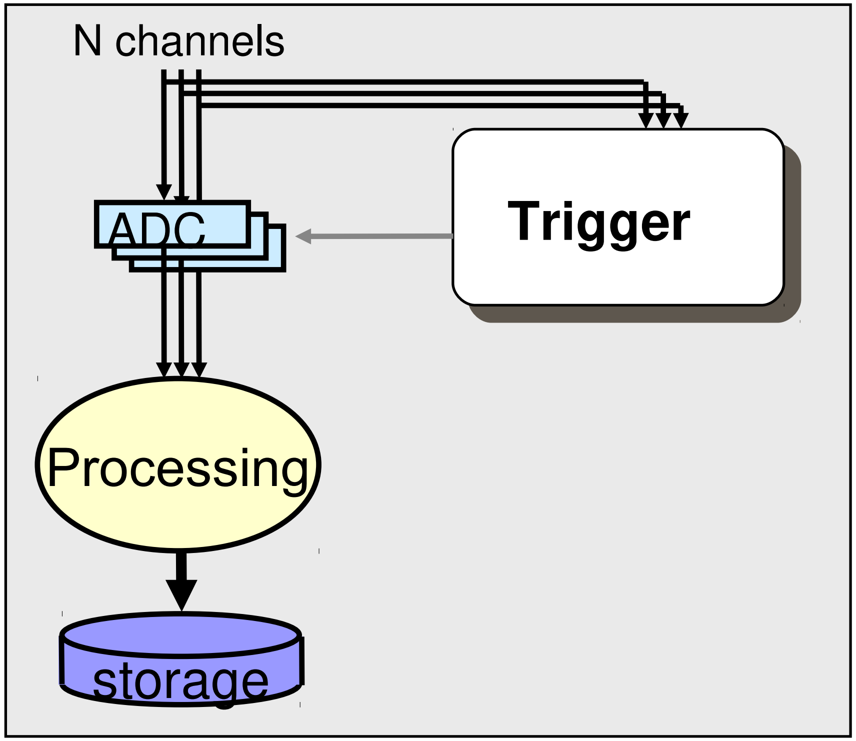
first discrete, then FPGA
(Xilinx Spartan 3AN evaluation board)

step two: increase # of sensors

- More granularity at the physical level
- Multiple channels (usually with FIFOs)
- Single, all-HW trigger
- Single processing unit
- Single I/O

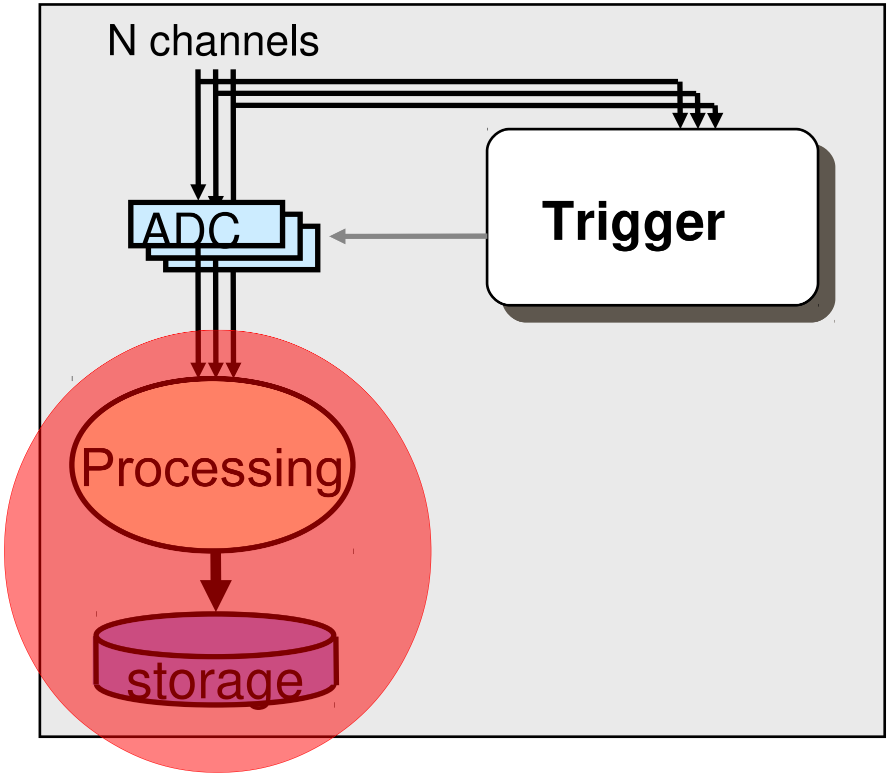


multi-channel, single-PU system



- common architecture in test beams and small experiments
- often rate limited by (interesting) physics itself, not TDAQ system
- or by the sensors

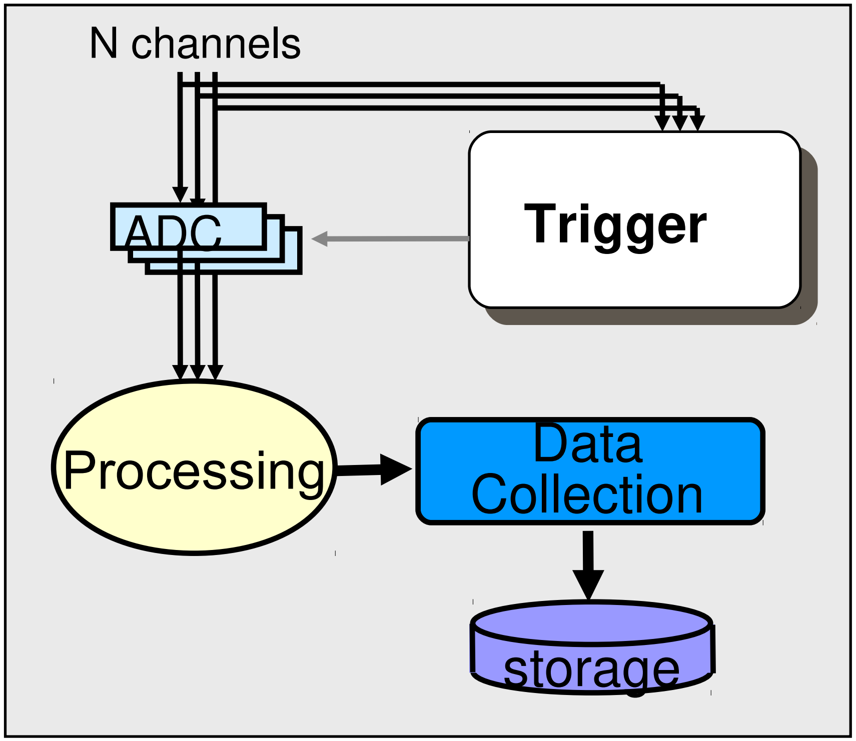
bottlenecks: PU and storage



- a single PU can be a limit
  - collect / reformat / compress data can be heavy
  - simultaneously writing storage
- final storage too:
  - VME up to 50MB/s → 1TB in 6h
    too many disks in a week!

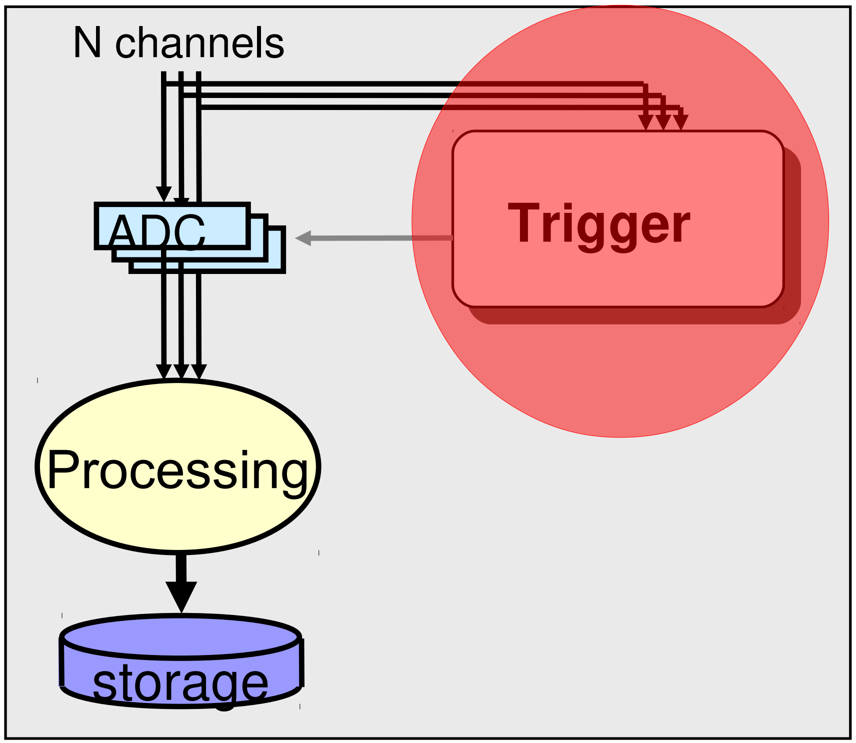
Laptop SATA disk: ~100 MB/s
USB2: ~60 MB/s

→ decouple storage from PU



- data transfer data → dedicated “Data Collection” unit to format, compress and store
- more room for smarter processing or decreased dead time on non-buffered ADCs

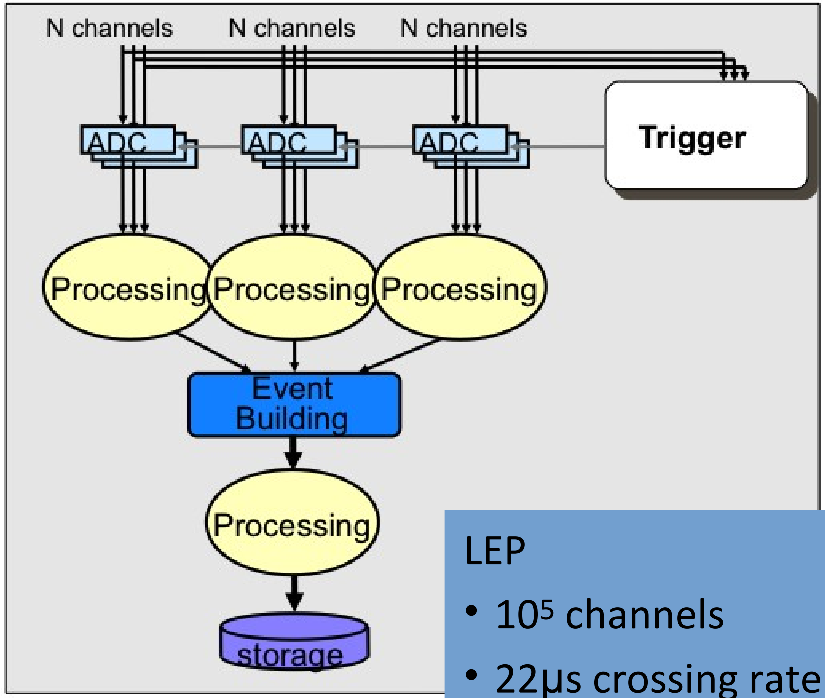
bottlenecks: trigger



- to reduce data rates (to avoid storage issues)
  - → non-trivial trigger
- complexity may already hit manageability limits for discrete logic (latency!)
- integrated, programmable logic came to rescue (FPGA)
  - → latency may go down to O(few ns)

step three: multiple PUs (SBC)

- e.g.: CERN LEP experiments
- complex detectors, moderate trigger rate, very little background
- little pileup, limited channel occupancy
- simpler, slow gas-based main trackers

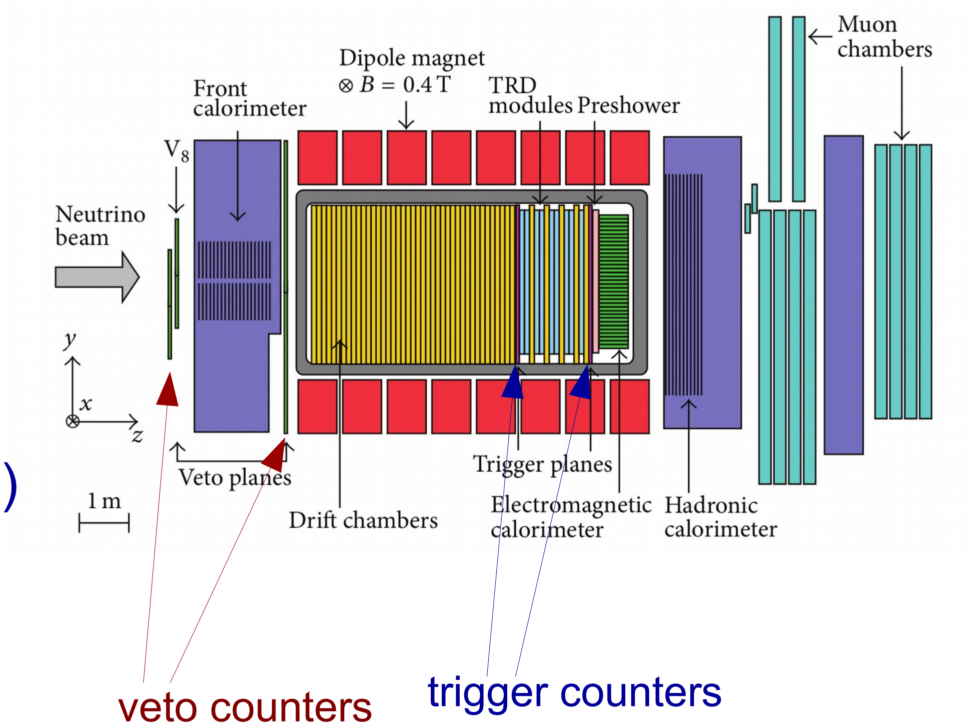


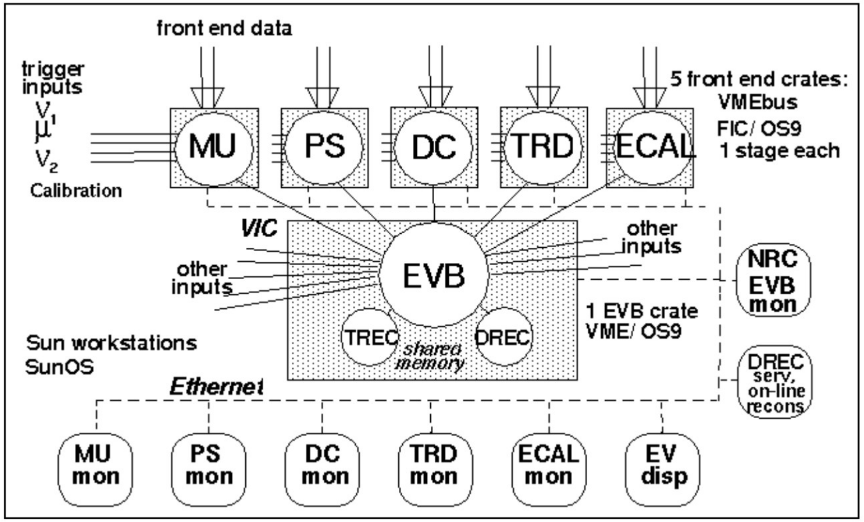
LEP

- 10^5 channels
- 22μs crossing rate
  – no event overlap
- single interaction

NOMAD (1995-1998)

- Search for $\nu_\mu \to \nu_\tau$ oscillations at the CERN West-Area Neutrino Facility (WANF)
- 2.4×2.4 m^2 fiducial (beam) area
- Two 4 ms spills with 1.8×10^{13} P.o.T. each (ν spills)
- One (2s) slow-extraction spill (μ spill)
- 14.4s cycle duration





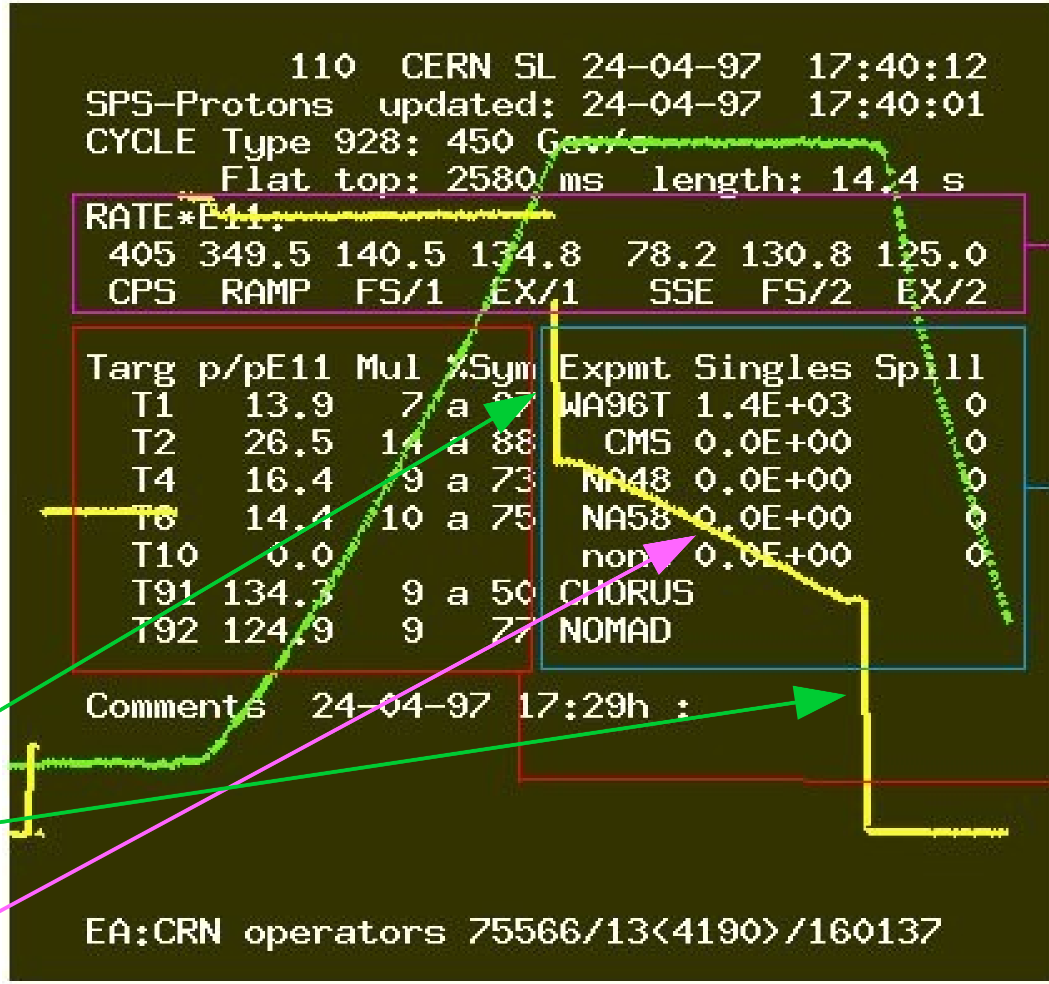
→ DAQ layout

WANF - SPS SuperCycle

14.4 s cycle length

2 × 4 ms neutrino spills (f/s extractions)

1 × 2 s muon spill (slow extraction)



Intensities in the SPS

Data from experiments

Steering on targets

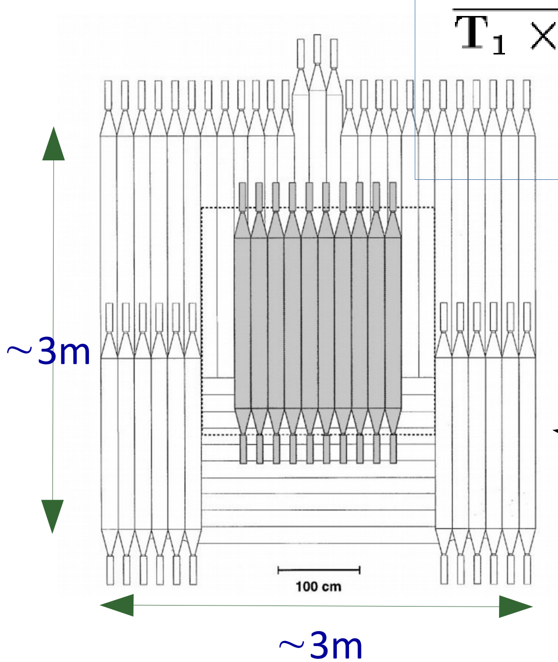
f/s extractions

slow extraction

triggering once more ...

menu for NOMADs:

ν-spill triggers	μ-spill triggers
$\overline{V} \times T_1 \times T_2$	$V \times T_1 \times T_2$
$\overline{V_8} \times FCAL$	$V_8 \times T_2$
$\overline{V_8} \times FCAL' \times T_1 \times T_2$	$V_8 \times T_1$
$\overline{T_1 \times T_2} \times ECAL, \overline{V_8} \times ECAL$	$V_8 \times T_1 \times T_2 \times FCAL'$
RANDOM	$V \times T_1 \times T_2 \times ECAL$

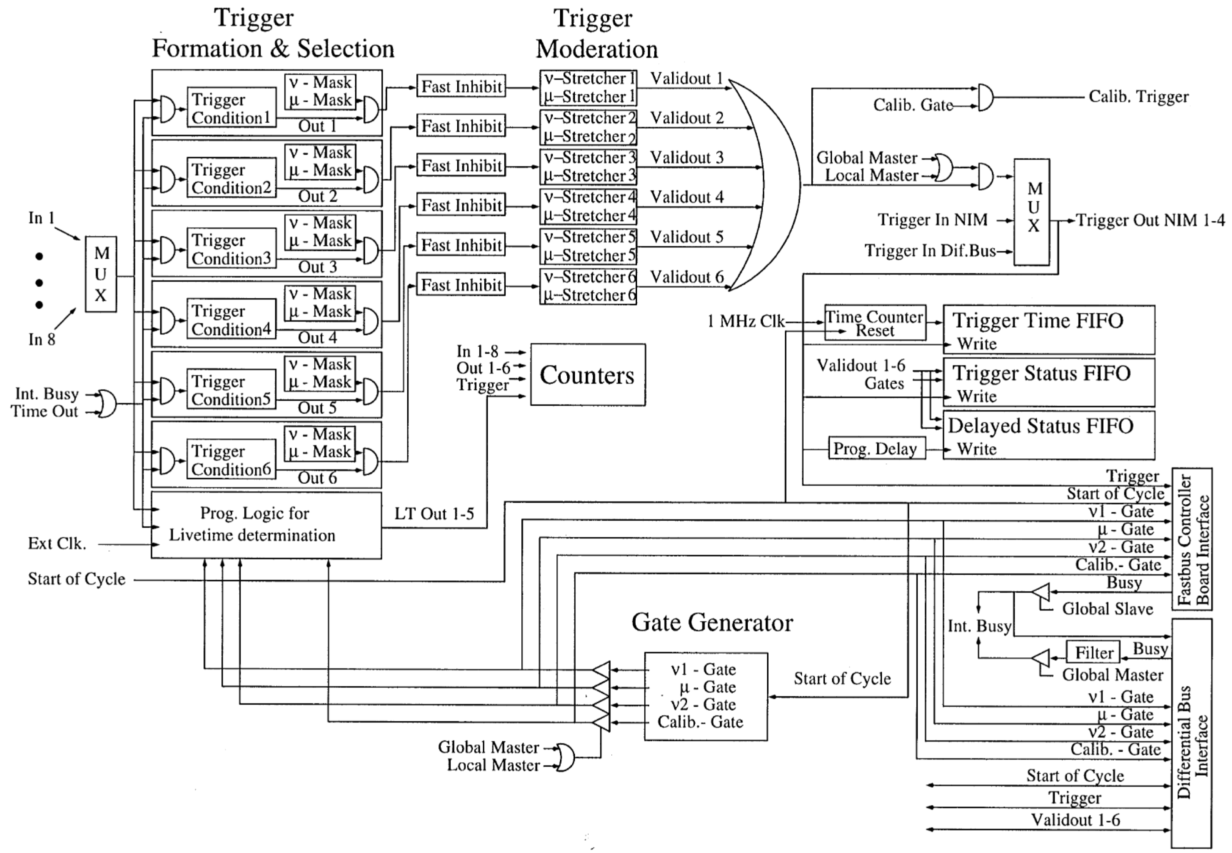


veto counters (central shaded area is V8)

triggering → FPGAs at work

MOdular TRIgger for NOmad (MOTRINO):

6 VME boards providing local and global trigger generation and propagation



DAQ

- FASTBUS digitisers:
  - ~200 (either 64 or 96 channel) xDC boards [x=Q,P,T]
  - O(≥ 2 us) conversion time, 256 event buffers
- VME readout and processing:
  - Motorola 68040 FIC8234 (OS9 real-time system) VME PUs
  - 5 for readout + 1 for event building
- Typically
  - ~4 kHz of neutrino triggers (~15 evts in each 4ms spill)
  - ~30 Hz of muon triggers (~60 evts in each 2s spill)
  - 256-events in off-spill calibration cycles (calibration triggers)

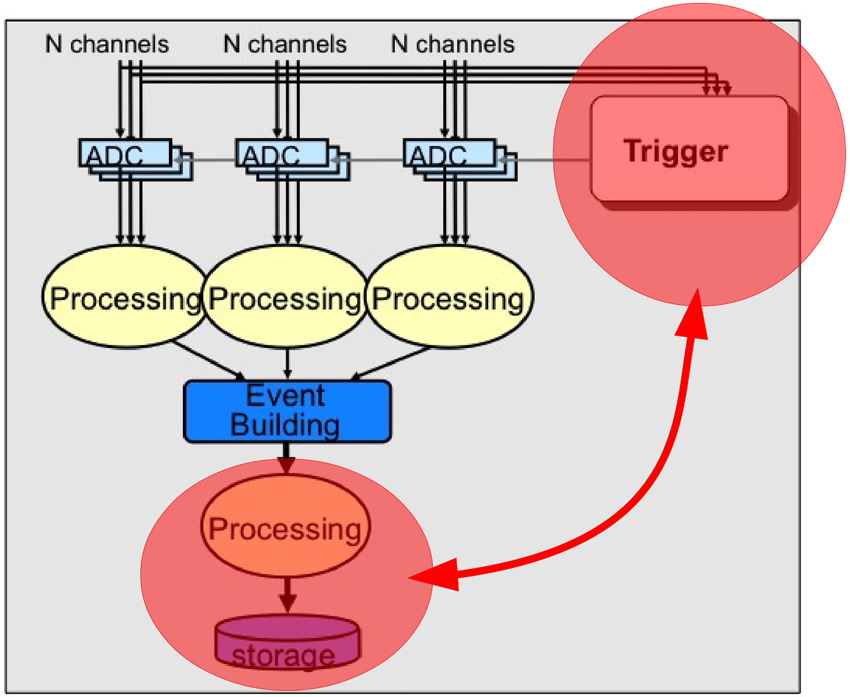
readout sequence

- On-spill on-board buffering
- Off-spill (i.e. off-beam) data transfer and processing
  - on spill (or calibration cycle): on-board event buffering (no way to read event by event)
  - end of spill (or calibration cycle): block transfer to VME
  - then event building + storage

- monitoring and control on SunOs and Solaris workstations

→ dead time in ν spills: ∼10% due to digitisation

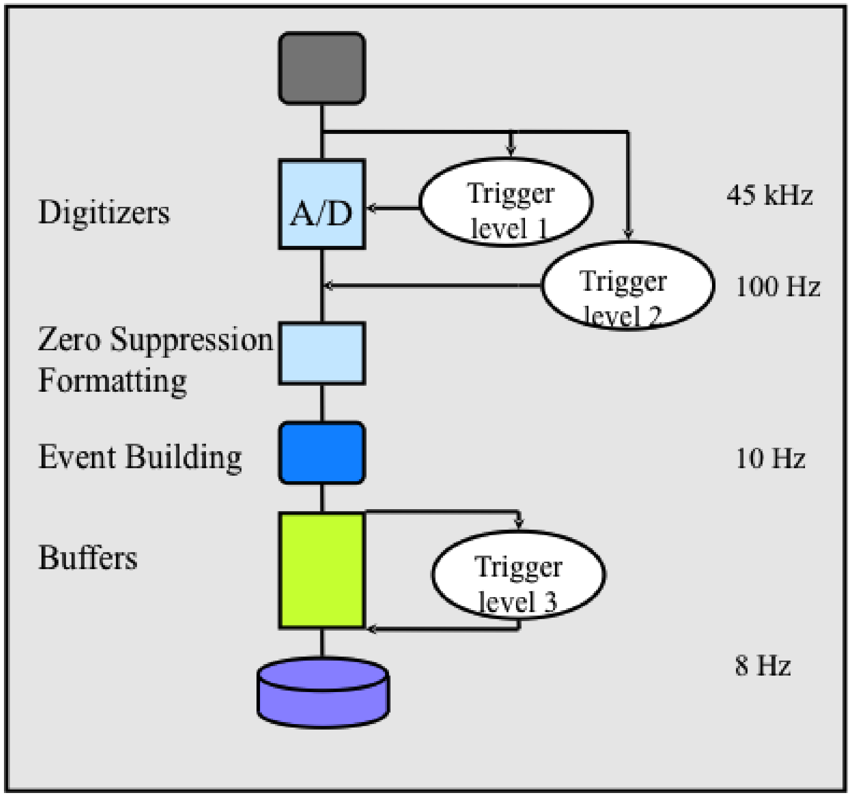
more bottlenecks ?



- trigger complexity ↔ storage
- single HW trigger not sufficient to reduce rate
- add L2 Trigger
- add HLT

step four: multi-level trigger

Typical Trigger / DAQ structure at LEP



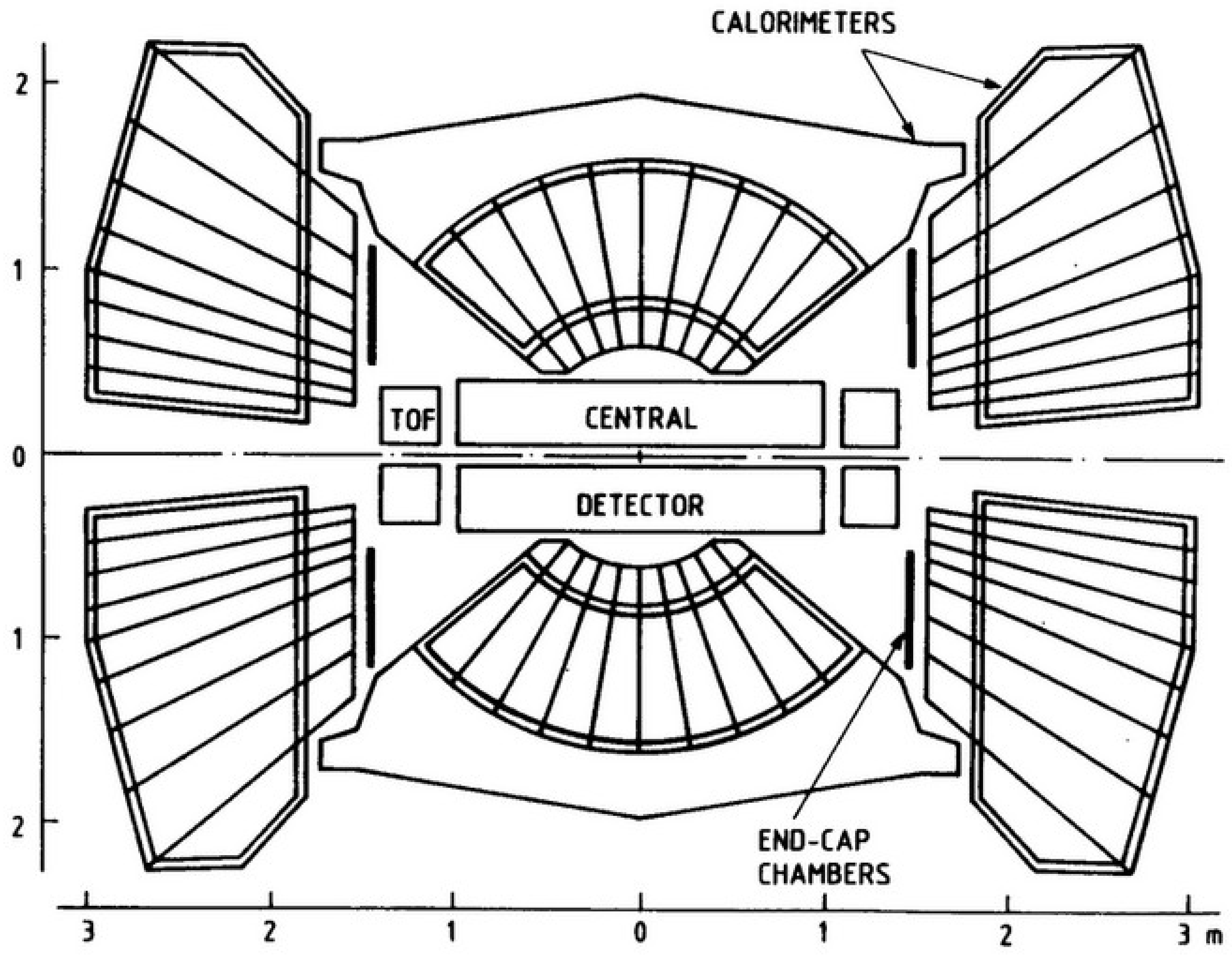
- more complex filters
  - → slower
  - → applied later in the chain

see Trigger lectures

LEP

- 10^5 channels
- 22μs crossing rate
  - –no event overlap
- single interaction
- L1 ~10^3 Hz
- L2 ~10^2 Hz
- L3 ~10^1 Hz
- 100kB/ev → 1MB/s

Upgraded UA2 experiment (1988-1991)



Upgraded UA2 experiment (1987-1991)

High-lumi pp collisions @ CERN SppS:

$\sqrt{s}$ = 630 GeV

L = 5 x 10^{30} cm^{-2} s^{-1} (one order of magnitude increase)

Goal:

W/Z physics

QCD

top quark and SUSY particle discovery

Upgraded UA2 experiment (1987-1991)

High-lumi pp collisions @ CERN SppS:

$\sqrt{s}$ = 630 GeV

L = 5 x 10^{30} cm^{-2} s^{-1} (one order of magnitude increase)

Goal:

W/Z physics

QCD

top quark and SUSY particle discovery

→ robust theoretical prediction for new physics

Upgraded UA2 experiment (1987-1991)

High-lumi pp collisions @ CERN SppS:

√s = 630 GeV

L = 5 x 10^{30} cm^{-2} s^{-1} (one order of magnitude increase)

Goal:

W/Z physics

QCD

top quark and SUSY particle discovery

→ robust theoretical prediction for new physics

… but nature was wrong!

Upgraded UA2 experiment (1987-1991)

High-lumi pp collisions @ CERN SppS:

$\sqrt{s}$ = 630 GeV

L = 5 x 10^{30} cm^{-2} s^{-1} (one order of magnitude increase)

Goal:

W/Z physics

QCD

top quark and SUSY particle discovery

Complex trigger signatures:

em, jet and missing E_T

Upgraded UA2 experiment (1987-1991)

Three-level trigger selection:

L1 from on-detector hardware

L2 over dedicated processors

L3 over FASTBUS processors (ALEPH event builder)

DAQ readout & monitoring:

CAMAC & FASTBUS → VAX/VMS platforms

Upgraded UA2 experiment (1987-1991)

Three-level trigger selection:

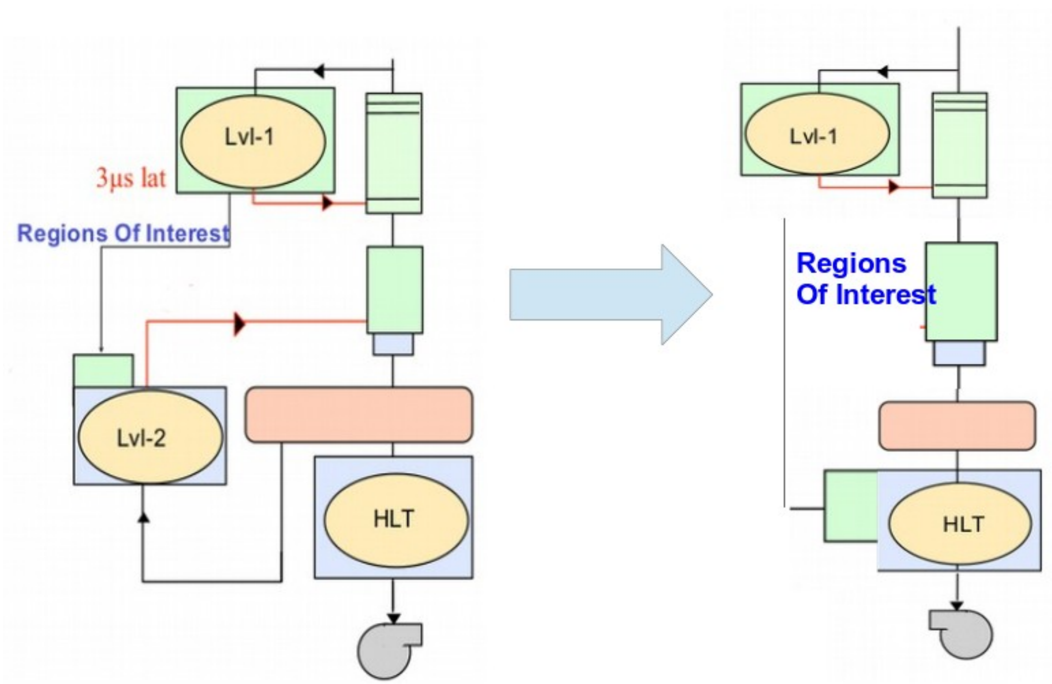
L1 from on-detector hardware
L2 over dedicated processors
L3 over FASTBUS processors (ALEPH event builder)

DAQ readout & monitoring:

CAMAC & FASTBUS → VAX/VMS platforms

No new physics, nevertheless many new/better measurements and observations of SM processes

ATLAS (from run-1 to run-2)



→ Merge L2 and L3 into a single HLT farm

– preserve Region of Interest but dilute the farm separation and fragmentation

– increase flexibly, computing power efficiency

trigger/event-selection latencies

Possible (e.g. ATLAS) values:

- L1 : O(1 μs in real-time) → let say = 2.5 μs
- L2 : O(10 ms) → let say = 40 ms
- L3(HLT) : O(s) → let say = 1 s

Q: do the 3 numbers mean the same thing ?

latency and real-time

real time: system must respond within some fixed delay

$\longrightarrow$ *Latency = Max Latency*

$\longrightarrow$ over fluctuations bad, will create dead time

non-real-time: system responds as soon as it's available

$\longrightarrow$ *Latency = Mean Latency*

$\longrightarrow$ over fluctuations fine, shouldn't create dead time

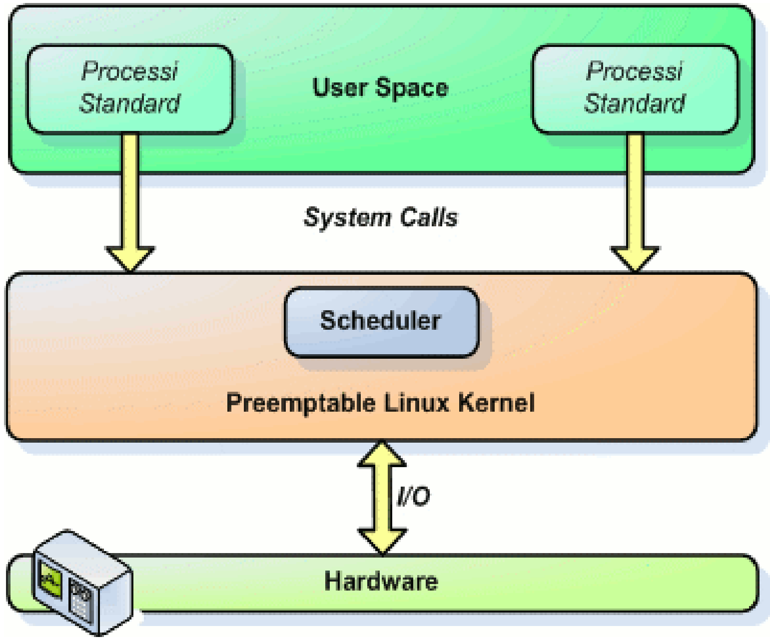
real time o.s. :

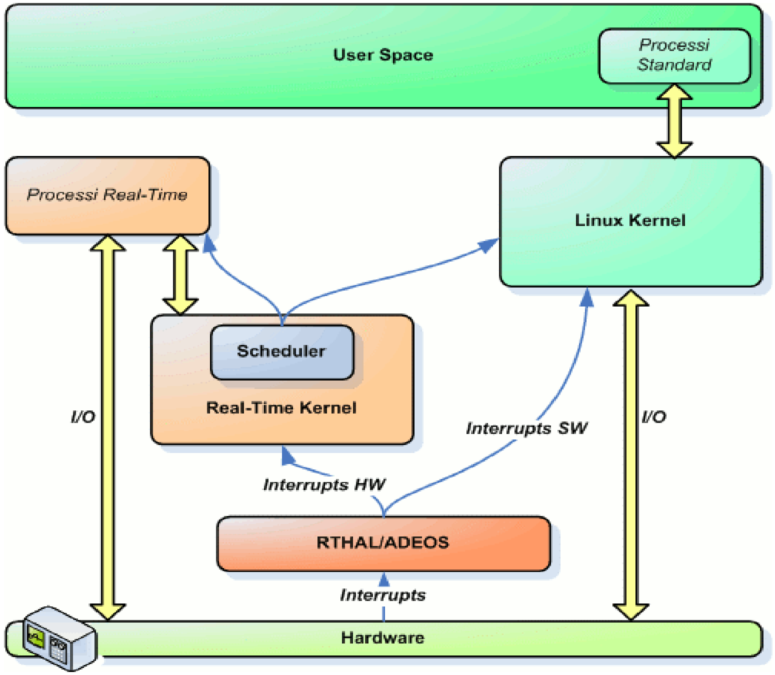
very stable time delay in responding to events

standard unix kernels are not real time:

a system call can in principle take any time

off-topic: real-time linux





Low-latency Ubuntu patch

(soft real time) :

Interruptible linux kernel

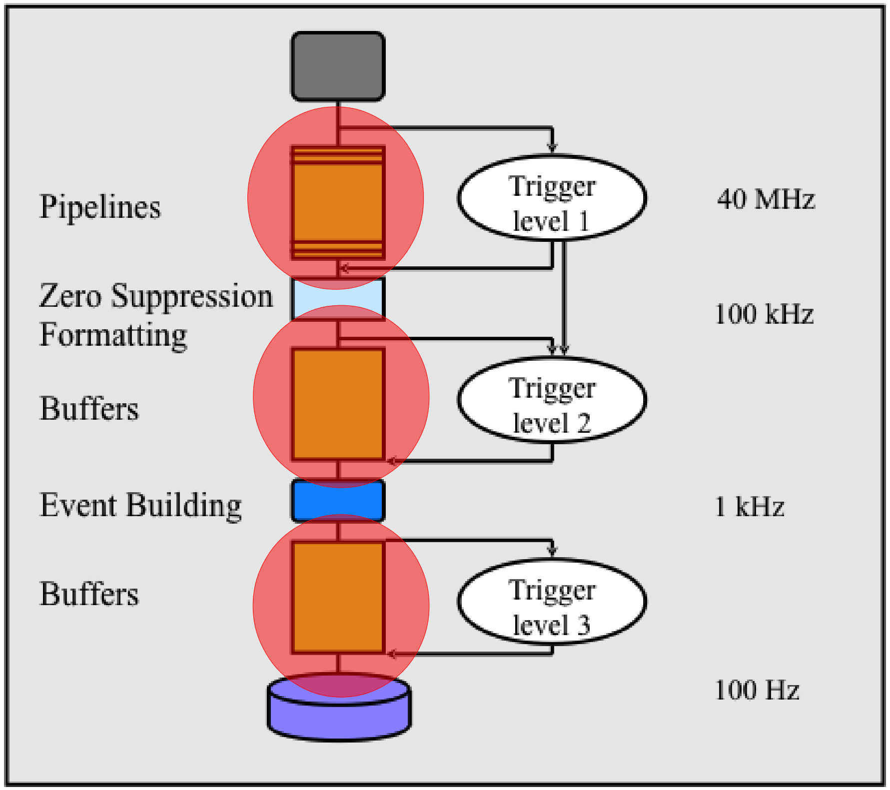
https://help.ubuntu.com/community/UbuntuStudio/RealTimeKernel

RTAI (hard real time) :

linux kernel as high-priority application

https://www.rtai.org/

step five: dataflow control

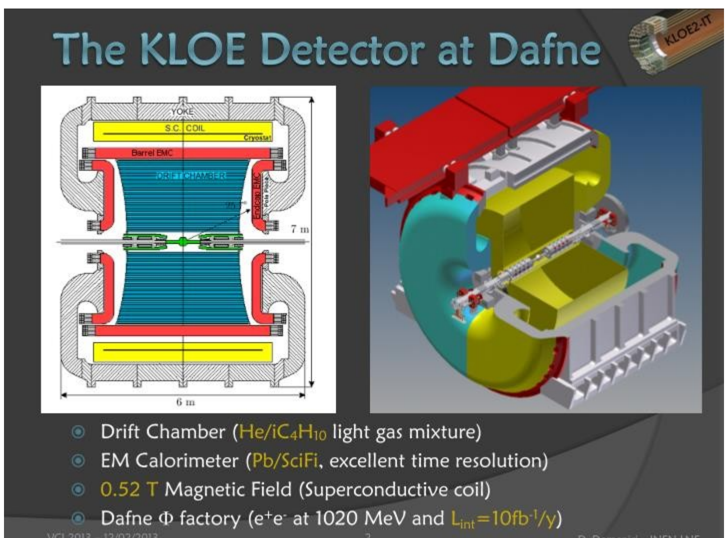


- Buffers are not the <final solution> they can overflow due to:
  - bursts
  - unusual event sizes
- Discard
  - local, or
  - “backpressure”, tells lower levels to discard

Who controls the flow?
FE (*push*) or EB (*pull*)

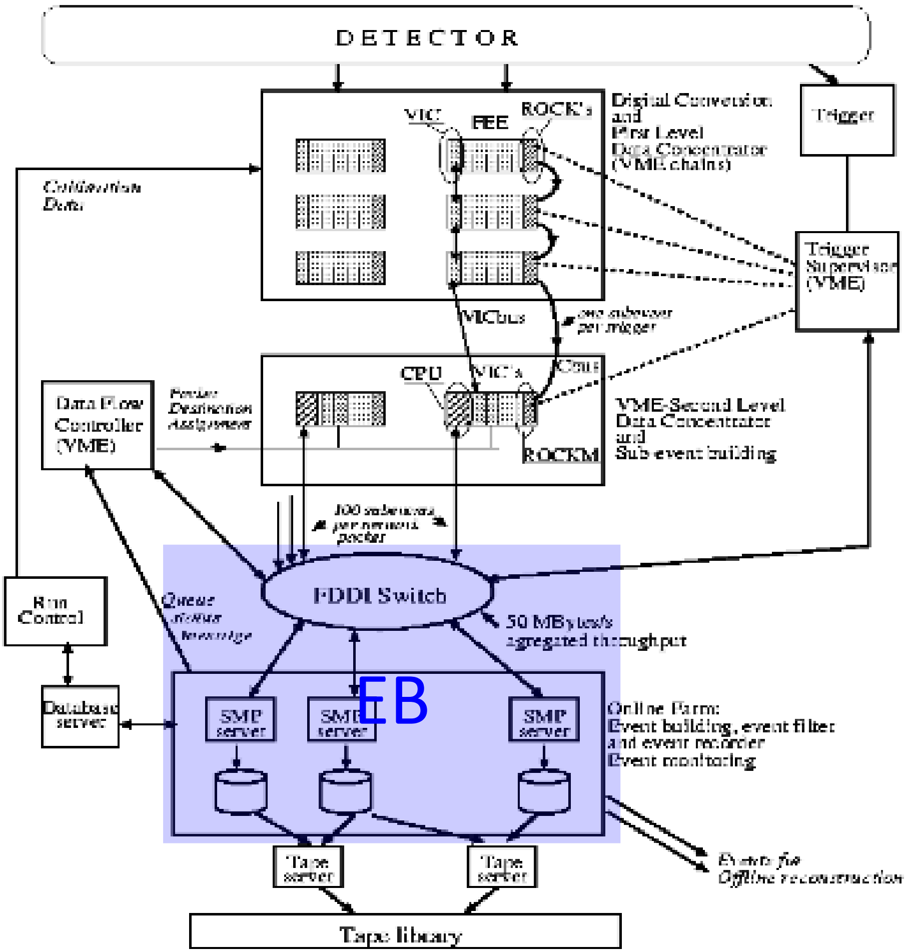
a *push* example: KLOE

- DAΦNE e^+e^- collider in Frascati
- CP violation parameters in the Kaon system
- “factory”: rare events in a high-rate beam



- 10^5 channels
- 2.7 ns crossing rate
  - rarely event overlap
  - “double hit” rejection
- high rate of small events
- L1 ~10^4 Hz
  - 2μs fixed dead time
- HLT ~10^4 Hz
  - ~COTS, cosmic rejection only
- 5 kB/ev → 50 MB/s [design]

KLOE



- deterministic FDDI network
- buffering at all levels (from FE to EB)
- *push* architecture
  vs pull used in ATLAS
  see DAQ Software lecture
- try EB load redistribution before resorting to backpressure

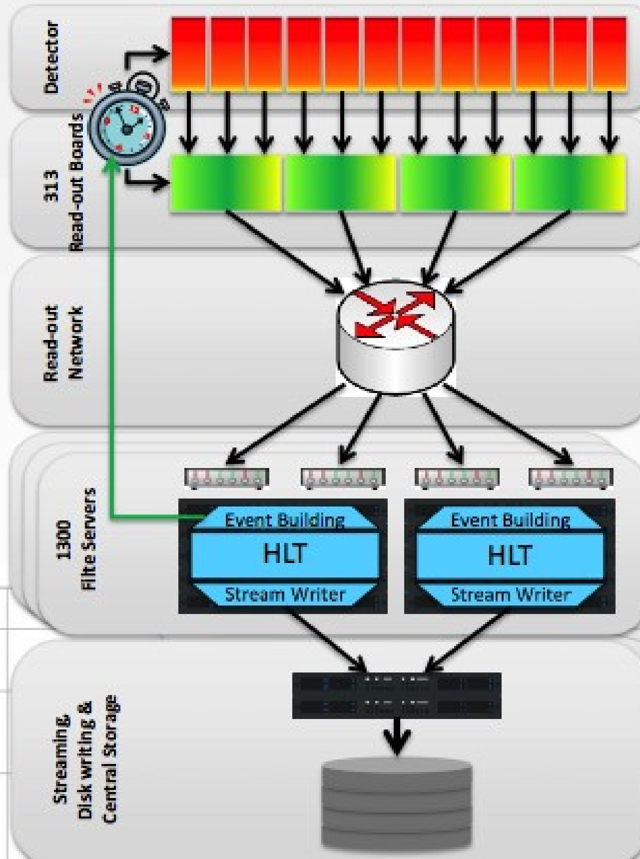
Which LHC experiment has a somewhat similar dataflow architecture ?

LHCb: network is dataflow

LHCb ONLINE

From Front-End to Hard Disk

- $O(10^6)$ Front-end channels
- 300 Read-out Boards with 4 x 1 Gbit/s network links
- 1 Gbit/s based Read-out network
- 1500 Farm PCs
- >5000 UTP Cat 6 links
- 1 MHz read-out rate
- Data is pushed to the Event Building layer. There is no re-send in case of loss
- Credit based load balancing and throttling



2

The LHCb Data Acquisition during LHC Run 1
CHEP 2013

more info in "TDAQ for the LHC experiments"

looking forward to LS2 and beyond

On some long term, all experiments looking forward to significant increase in L1 trigger rate and bandwidth. ALICE and LHCb will pioneer this path during LS2

DAQ@LHC Workshop

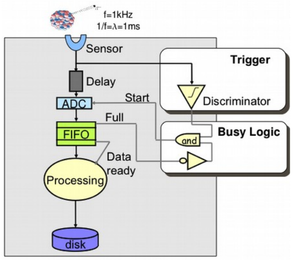
ALICE

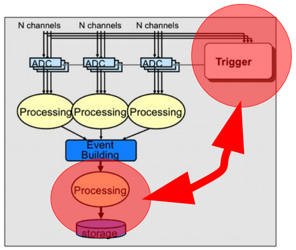
- First level trigger for Pb-Pb interactions **500 Hz → 50 kHz**
- 22 MB/event
  - **1 TB/s readout → <u>500 PB/month</u>**
- Data volume reduction
  - on-line full reconstruction
  - discard raw-data
- Combined DAQ/HLT/offline farm
  - COTS, FPGA and GPGPU

LHCb

- **1 MHz → 40 MHz** readout and event building → trigger-less
  - trigger support for staged computing power deployment
- 100 kB/event
  - on-detector zero suppression → rad-hard FPGA
  - 4 TB/s event-building

trends





- Integrate synchronous, low latency in front end
  - limitations do not disappear, but decouple (factorise)
  - all-HW implementation
  - isolated in replaceable(?) components
- Use networks as soon as possible

- Deal with dataflow instead of latency
- Use COTS network and processing
- Use "network" design already at small scale
  - easily get high performance with commercial components

take care, lot of issues not covered:

Hw configuration

Sw configuration

Hw control & recovery

Sw control & recovery

Monitoring

…

Thank you for your patience …

Lost & Found
(off-topics)

Appendix A: Cables and Transmission Lines

Spoken about signals, amp.s, digitisers, … but ...

… almost nothing about how signals are transmitted over long distances. ***Is there any issue ?***

Q(1): what is a cable (for a single signal) ?

a couple of ideal conductors (R=C=L=0) ?

Q(2): which speed can it reach ?

Q(3): what's its impedance ?

Q(4): what does it to your signal ?

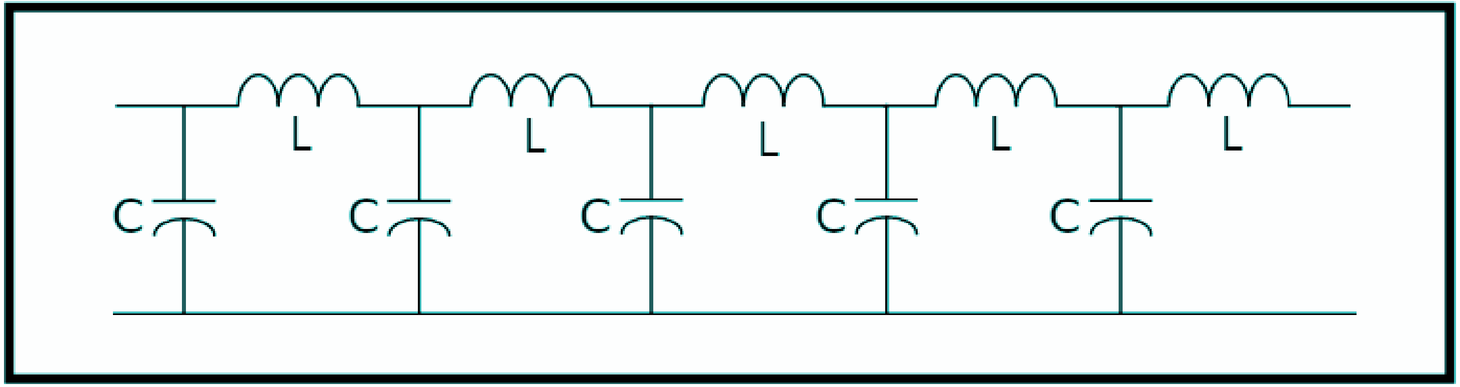
Ok the full line must be properly matched:

Z(out) = Z(cable) = Z(in)

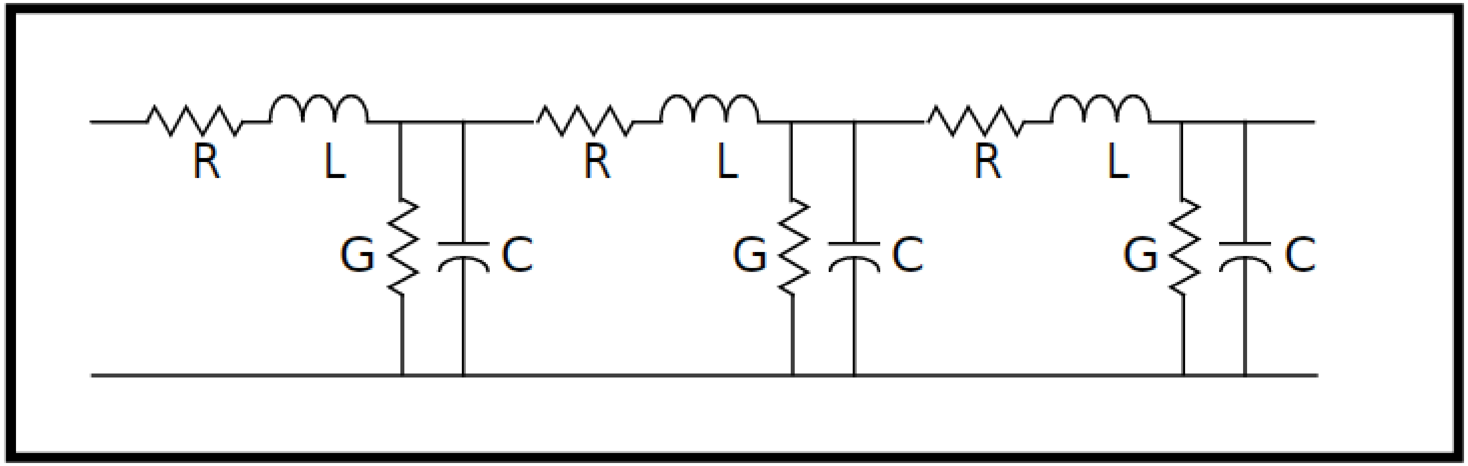
<u>That's all ?</u>

Cables and Transmission Lines

Lossless transmission line:



Lossy transmission line:

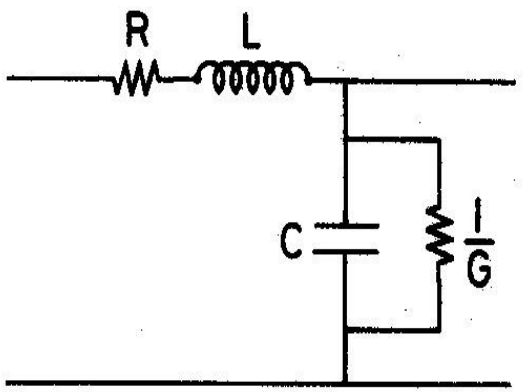


Cables

Cable element (dz):

$$L \approx \frac{\mu}{2\pi} \ln\left(\frac{b}{a}\right) \quad [\mathrm{H/m}]$$

$$C \approx \frac{2\pi\varepsilon}{\ln(b/a)} \quad [\mathrm{F/m}]$$



R depends on the frequency (skin effect)

G should be negligible

$$Z = (L/C)^{1/2}$$

$$v_p = (LC)^{-1/2} = (\mu\varepsilon)^{-1/2}$$

Cables

Equation for standing waves:

$$\frac{\partial^2 V}{\partial z^2} = LC\frac{\partial^2 V}{\partial t^2} + (LG + RC)\frac{\partial V}{\partial t} + RGV$$

solution:

$$\frac{d^2 V}{dz^2} = (R + i\omega L)(G + i\omega C)V = \gamma^2 V$$

$$\gamma = \alpha + ik = \sqrt{(R + i\omega L)(G + i\omega C)}$$

R usually dominated by the skin effect:

$$R(\omega) = r*D/(4*\delta)$$

r = resistance per unit length
D = diameter internal conductor
δ = skin depth ~ 1/√ω

Cable Losses

Neglecting the transconductance G:

$$\alpha = R(\omega)/(2Z_0) \sim c\sqrt{\omega}$$
$$k = \omega\sqrt{RC} = \omega/(\beta c)$$

$$V(z,t) = V_1 \exp(-\alpha z)\exp[i(\omega t - kz)]$$

50-Ohm fast (v = 4 ns/m) CERN-store cables:

04.61.11.F - COAXIAL CABLE 50 OHM - TYPE C-50-6-1
04.61.11.H - COAXIAL CABLE 50 OHM - LOW LOSS - TYPE C-50-11-1

f(-3db, 40 m, cable C-50-6-1) ~ 120 MHz
f(-3dB, 40 m, low loss cable) ~ 640 MHz

Signal Distortions

Time parameter:

$$\alpha \sim \mu \sqrt{f}$$

$$\tau_0 = (\mu z)^2 / \pi$$

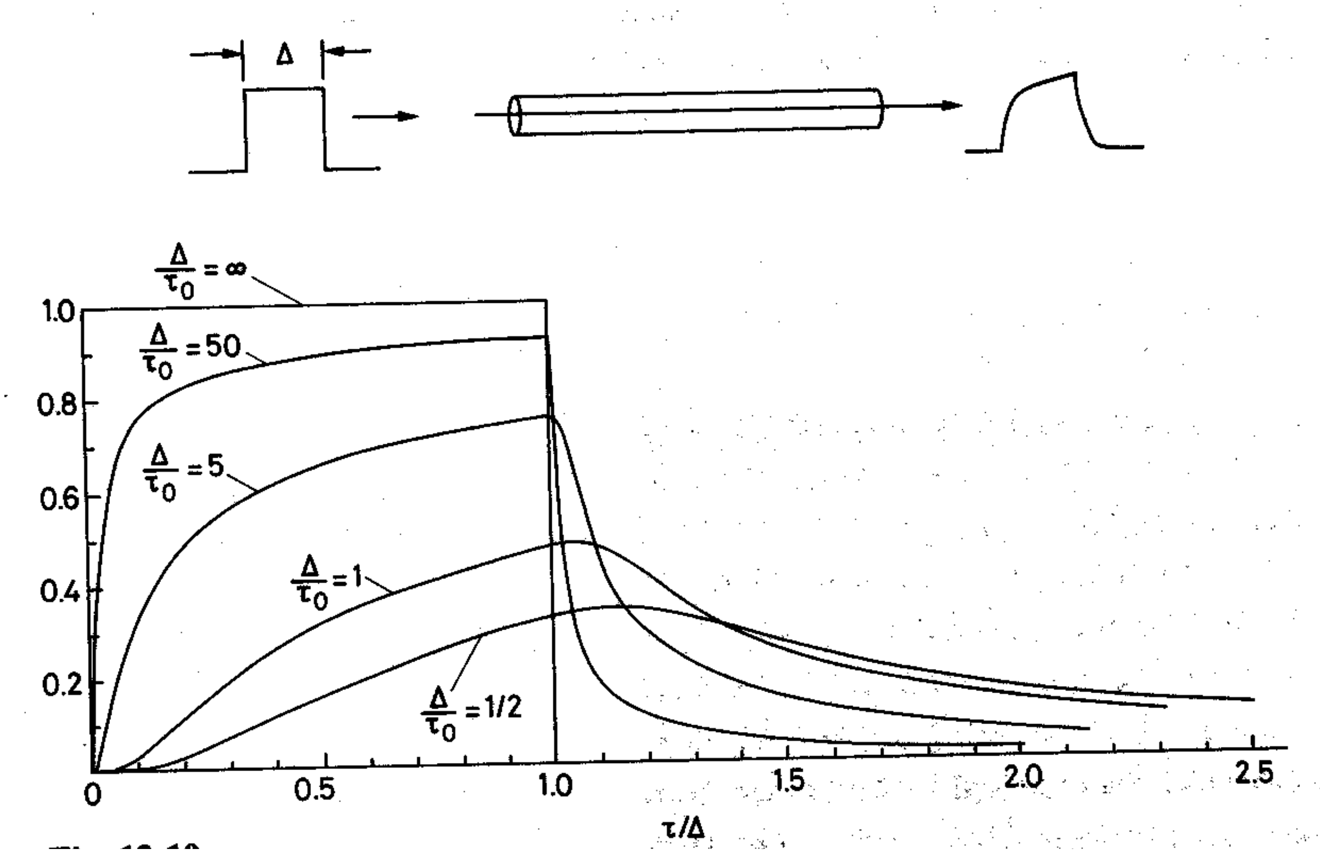
μz ~ 32*E-6 (C-50-6-1), 14E-6 (low loss cables)

$$\tau_0 \sim 320\, ns\, (C-50-6-1)$$

$$\tau_0 \sim 60\, ns\, (low\, loss\, cables)$$

*** Take care: would like $\tau_0 \ll \tau$(signal)

Digital Pulse Distortions

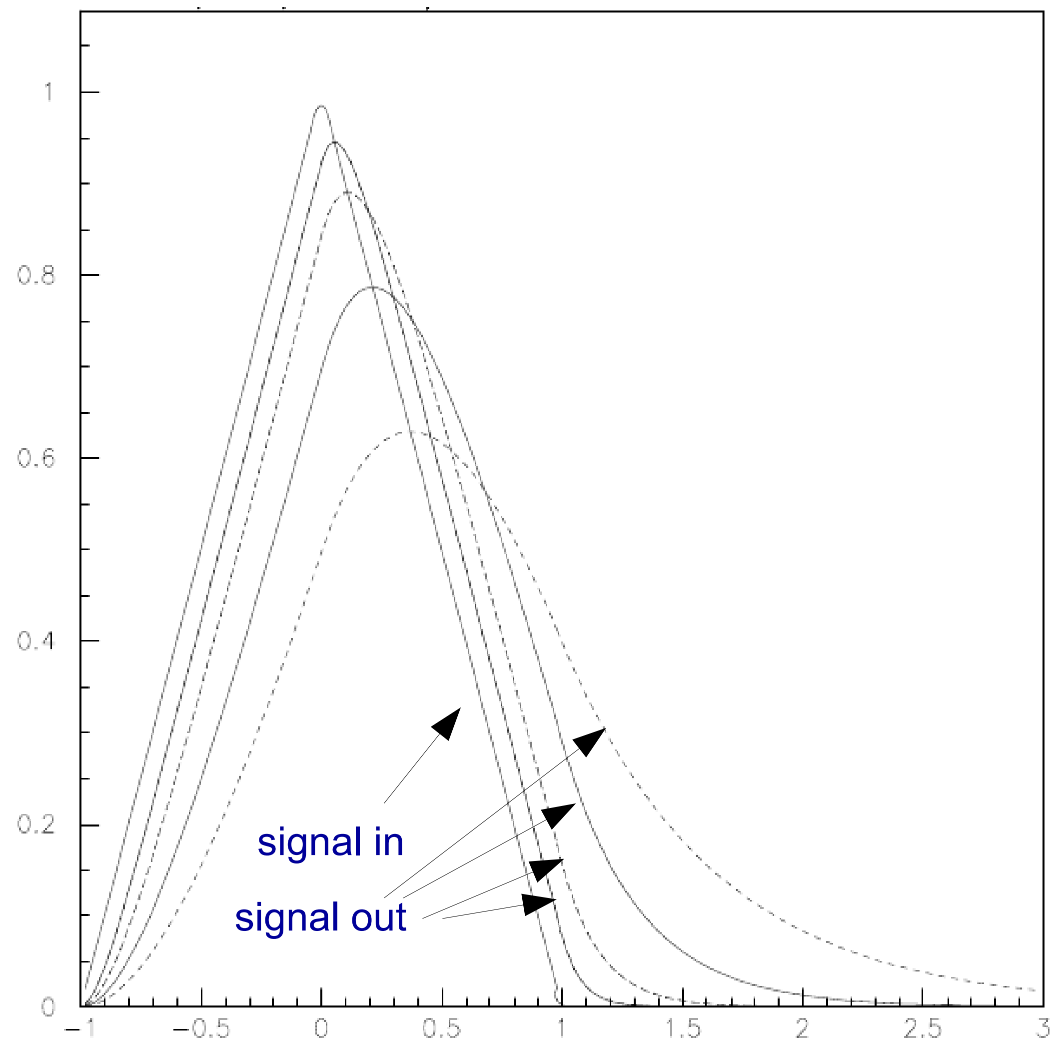


$$\text{for } \alpha \sim a\sqrt{f}$$

$$\tau_0 = (a\,z)^2/\pi$$

Bandwidth Effects – Analog Signals

~1ns analog-signal response for
BW ~ 300, 150, 75, ... MHz



Appendix B: backtrace

Segfaulting ? Have a look at backtrace:

https://www.gnu.org/software/libc/manual/html_node/Backtraces.html

BACKTRACE(3) Linux Programmer's Manual BACKTRACE(3)

NAME

backtrace, backtrace_symbols, backtrace_symbols_fd - support for application self-debugging

SYNOPSIS

```
#include <execinfo.h>

int backtrace(void **buffer, int size);

char **backtrace_symbols(void *const *buffer, int size);

void backtrace_symbols_fd(void *const *buffer, int size, int fd);
```

HowTo

1) file “my_segf.cxx” : install a signal handler to print the backtrace

```
#include <stdio.h>
#include <execinfo.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>

void handler(int sig) {
  void *array[10];
  size_t size;

  // get void*'s for all entries on the stack
  size = backtrace(array, 10);

  // print out all the frames to stderr
  fprintf(stderr, "Error: signal %d:\n", sig);
  backtrace_symbols_fd(array, size, STDERR_FILENO);
  exit(1);
}

void baz() {
 int *foo = (int*)-1; // make a bad pointer
 printf("%d\n", *foo);       // causes segfault
}

void bar() { baz(); }
void foo() { bar(); }

int main(int argc, char **argv) {
  signal(SIGSEGV, handler);   // install our handler
  foo(); // this will call foo, bar, and baz.  Baz segfaults.
}
```

2) compile with -g debug flag on:

```
g++ -g -rdynamic my_segf.cxx -o
my_segf
```

3) get the crash:

```
[roberto@bob-laptop ~]$ ./my_segf
Error: signal 11:
./my_segf(_Z7handleri+0x1c)[0x400a52]        <- handler
/lib64/libc.so.6(+0x347e0)[0x7fa55f1c07e0]   <- libc
./my_segf(_Z3bazv+0x14)[0x400aab]            <- my crash
./my_segf(_Z3barv+0x9)[0x400aca]
./my_segf(_Z3foov+0x9)[0x400ad6]
./my_segf(main+0x23)[0x400afc]
/lib64/libc.so.6(__libc_start_main+0xf1)
[0x7fa55f1ac731]
./my_segf(_start+0x29)[0x400969]
```

4) crash is at (_Z3bazv+0x14) ... the function name is "_Z3bazv" (c++ function name mangling). How to get it ?

5) Demangle it thanks to: http://demangler.com/

6) Take the Answer: baz() → crash is at (baz+0x14)

7) crash is at (baz+0x14) ... open the debugger: gdb my_segf

```
(gdb) info address baz
Symbol "baz()" is a function at address 0x400a55.
```

8) so crash is at address (0x499a55+0x14) … then:

```
(gdb) info line *(0x400a55+0x14)
Line 24 of "my_segf.cxx" starts at address 0x400a65 <baz()+16>
  and ends at 0x400a7c <baz()+39>.
```

9) got it ! That’s not yet the reason but ...

Appendix C: Profiling

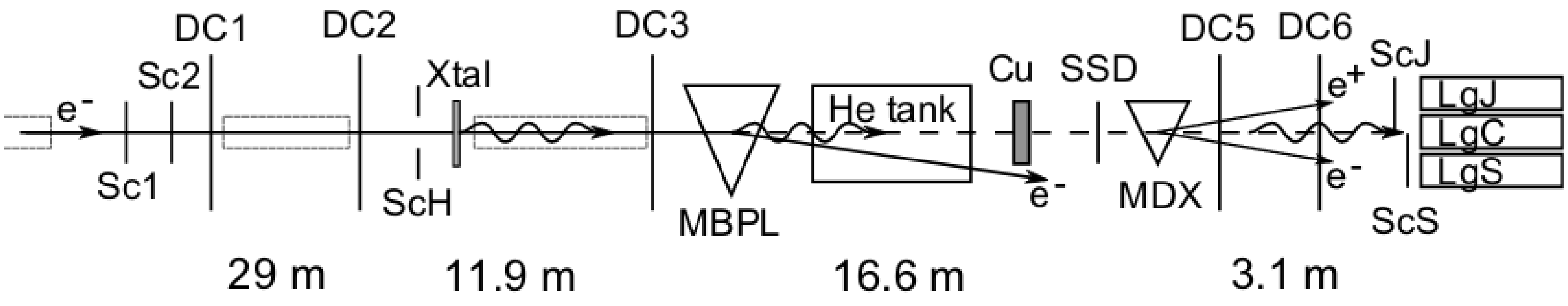
Take care: optimize your code – first of all - where it really needs. To get it, you may use of profiling.

for C/C++ code, look (for example) at this gprof tutorial:

http://www.thegeekstuff.com/2012/08/gprof-tutorial/

Very simple, at least for standalone code ...

another example: NA43/63



- Radiation processes: coherent emission in crystals and structured targets, LPM suppression...
- 80/120 GeV e- from CERN SPS slow extraction
- 2s spill every 13.5s

- Needs very high angular resolution
- Long baseline + high-res, low material detectors
  → drift Chambers
- 10 kHz limit on beam for radiation damage

→ 2-3 kHz physics trigger

NA43/63

- 30-40 TDC, 6-16 QDC, 0-2 PADC
  (depending on measurement)
- CAMAC bus
  1 MB/s, no buffers, no Z.S.
- single PC readout
- NIM logic trigger
  (FPGA since 2009)
  - pileup rejection
  - fixed deadtime