Istituto Nazionale di Fisica Nucleare
Sezione di Pavia

Corso di formazione su Trigger e Data Acquisition

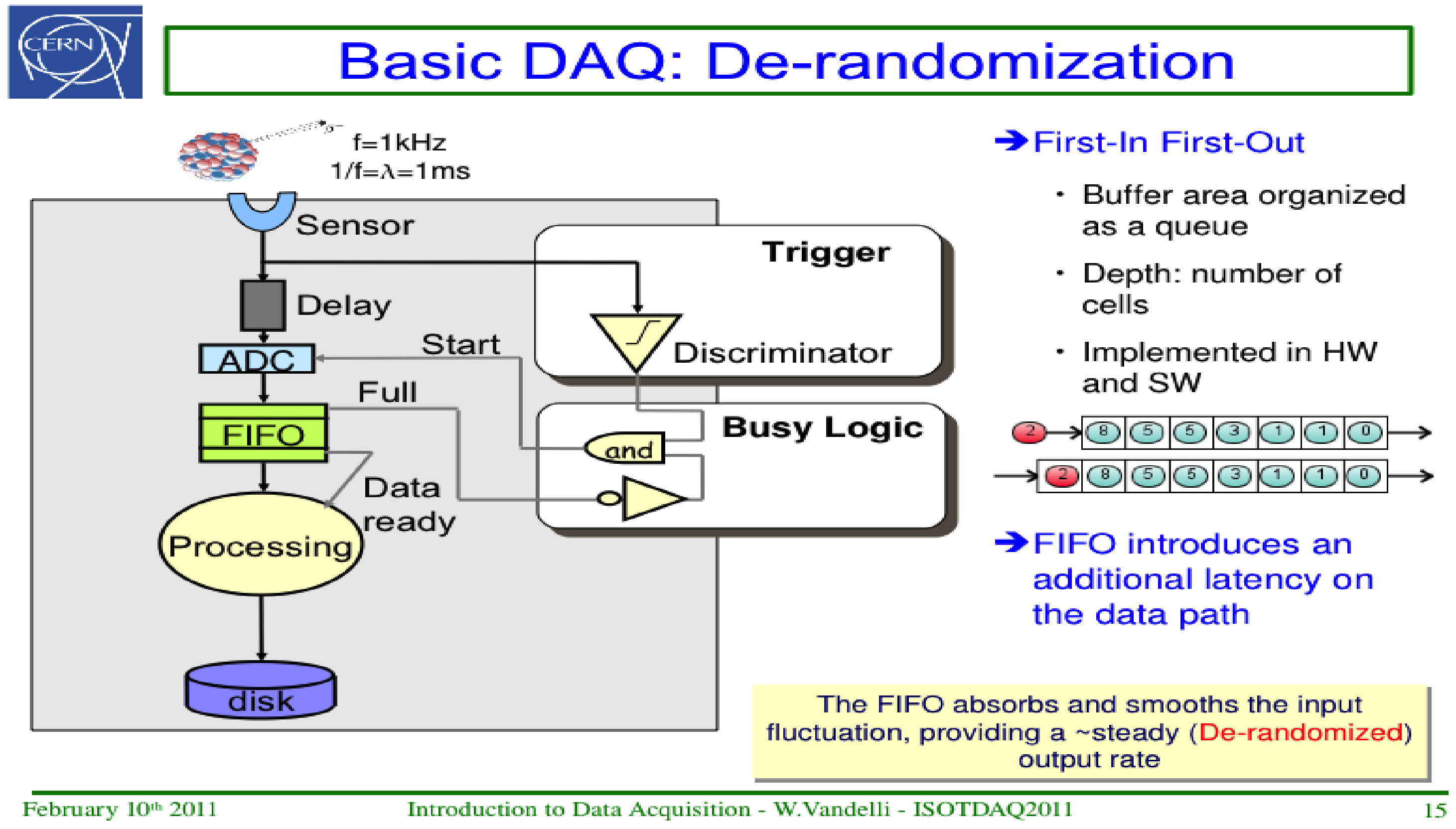
TDAQ design:
da testbeam a esperimenti medium size

Roberto Ferrari
INFN - Pavia

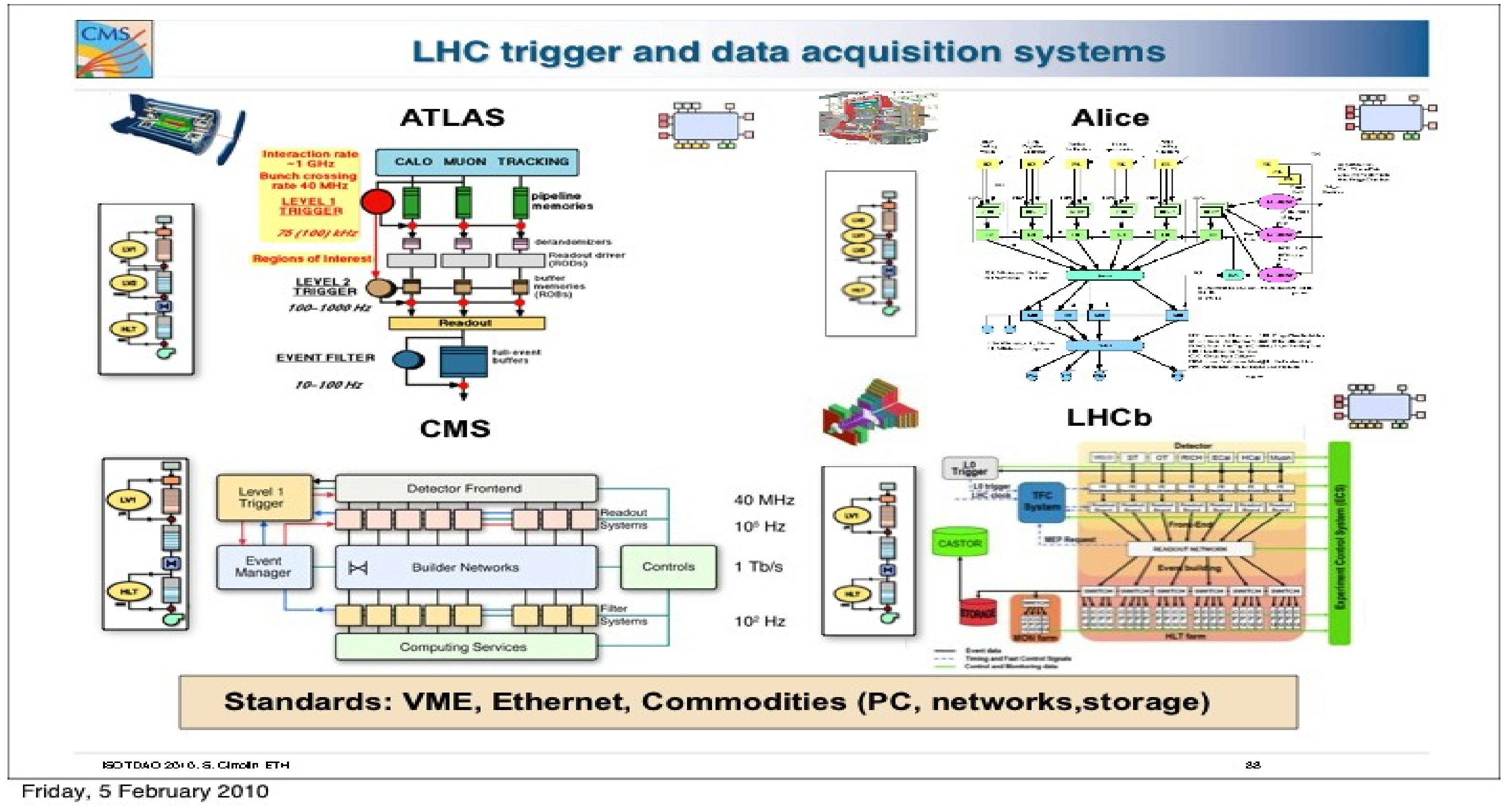
Napoli, 10 ottobre 2023

Trying to move …

from here:

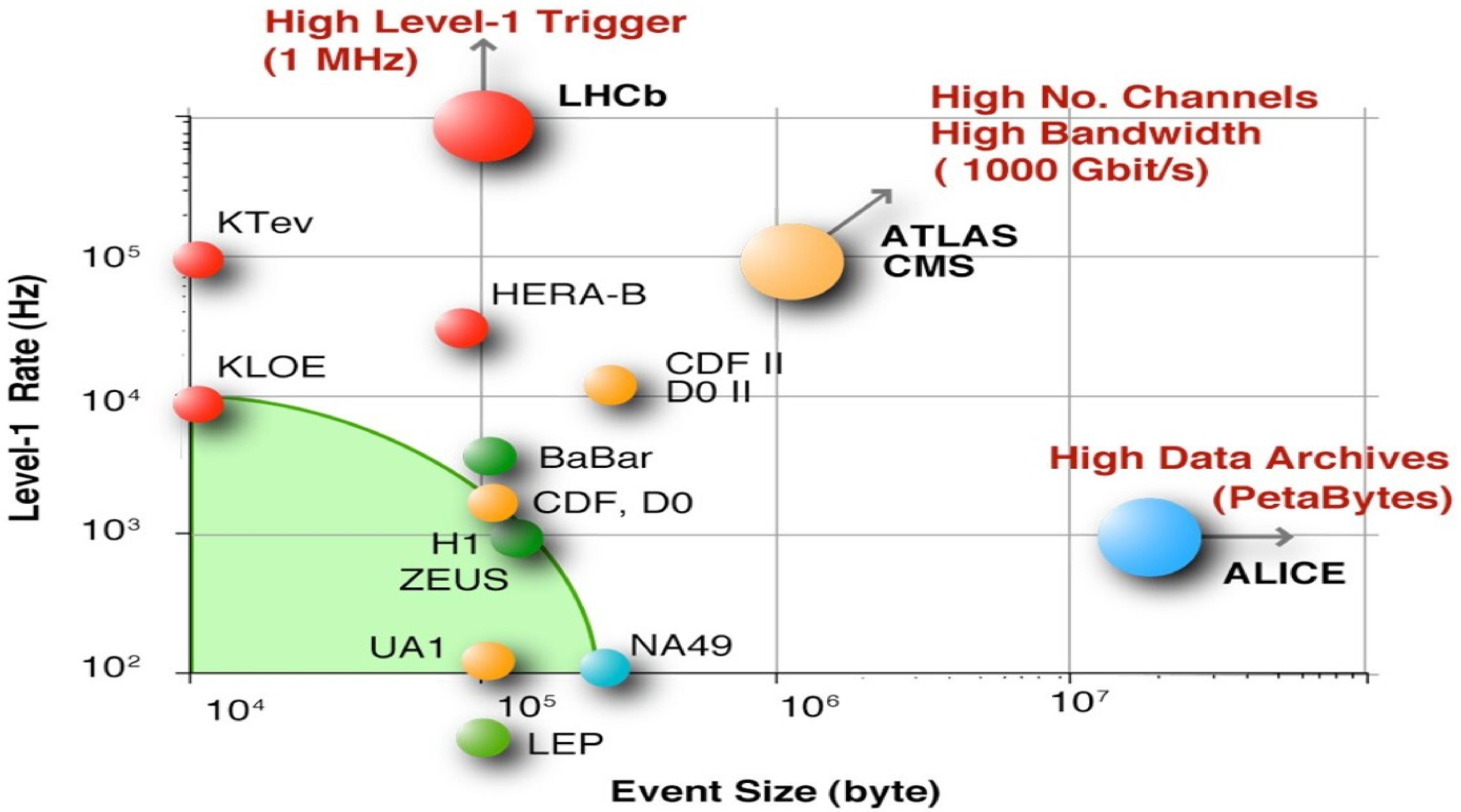


to here:



Trigger & DAQ in HEP

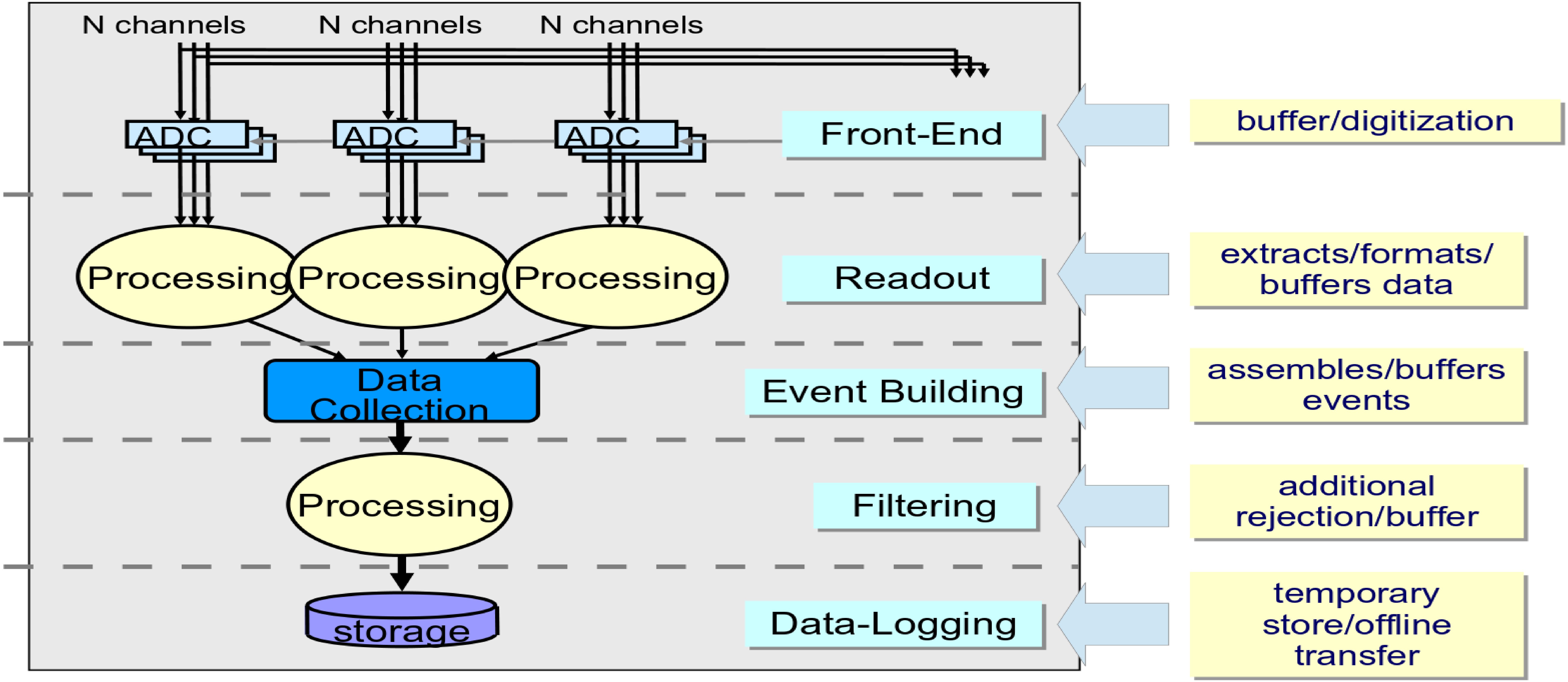
Trigger
rate



DAQ
ev. size

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

disclaimer

- Too many slides → many only as reference
- Likely too naive for the audience

Apologies for that !

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

however, 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

Coincidence circuit [wikipedia]:

“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

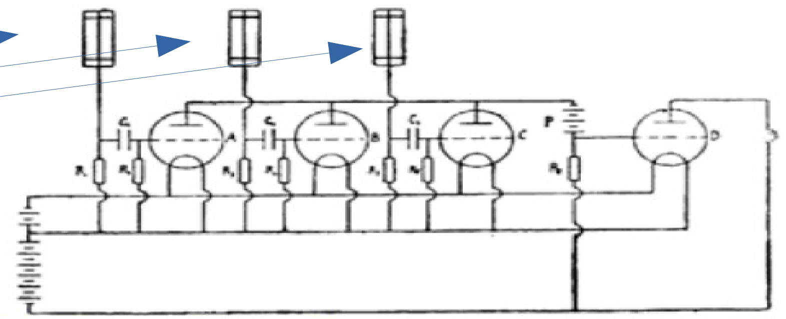


Fig. 17 – Il circuito di Rossi per rivelare coincidenze di raggi cosmici che arrivano sui contatori Geiger (i rettangoli in alto dello schema)[19].

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

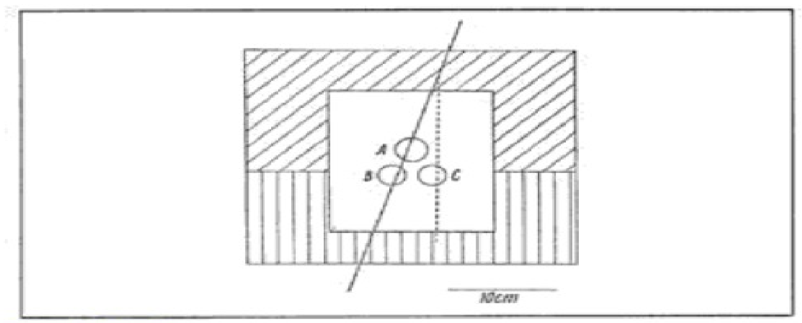


Fig. 18 – L'uso del circuito di Rossi per rivelare una coincidenza tripla che, nella disposizione in figura dei tre contatori, mostra la produzione di una radiazione secondaria (linea tratteggiata) da parte della radiazione primaria (linea continua)[20].

simplest case: 2-signal coincidence

a simple trigger system

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

Veto (anti-coincidence)

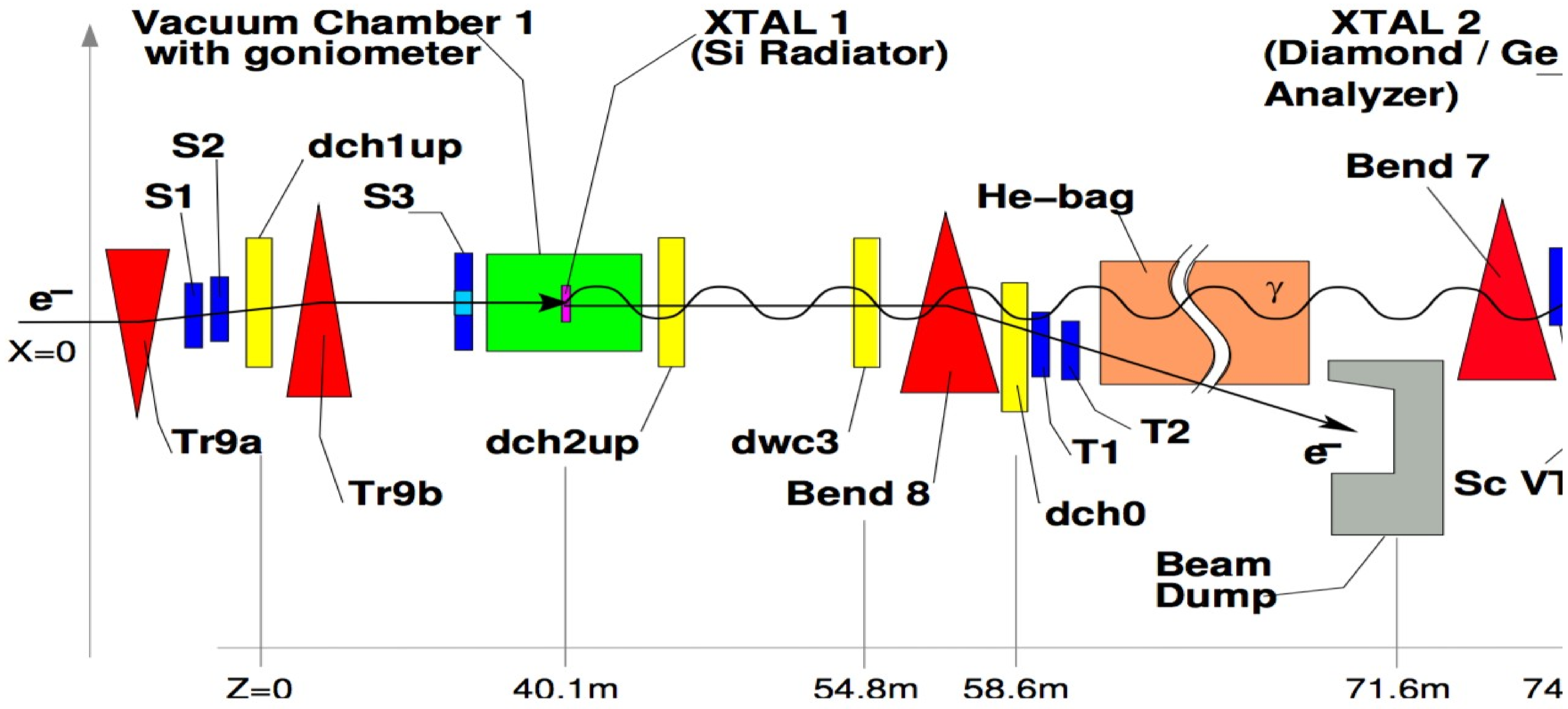
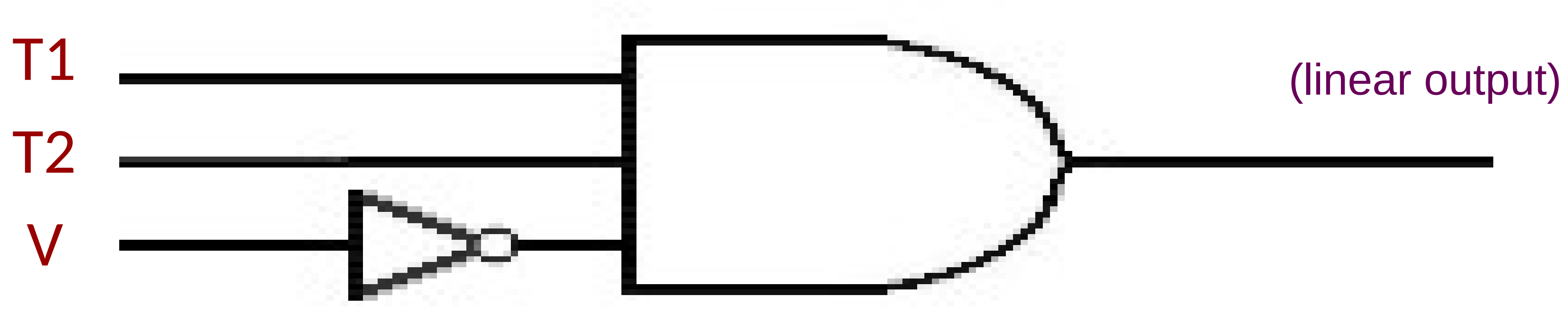


Fig. 1. Setup of the Na59 Experiment

any issue ?

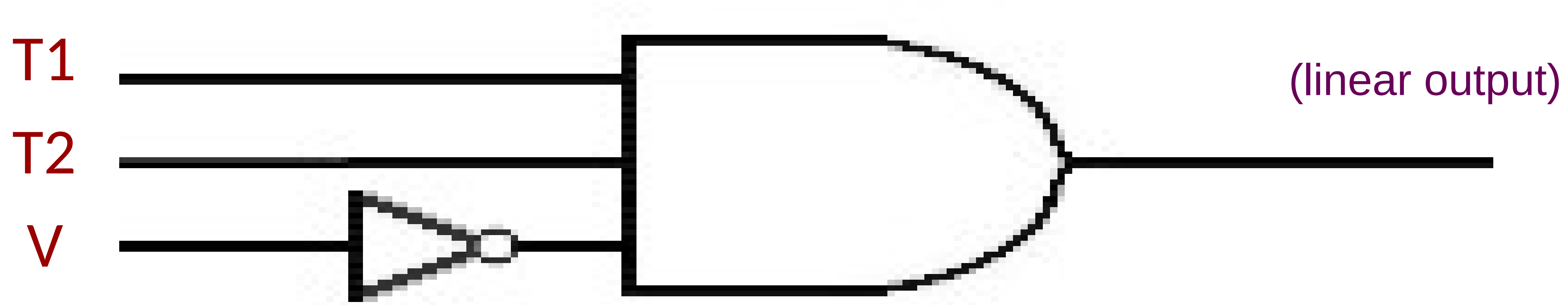
T1, T2, V : logic pulses (“0/1” values)



(anti-)coincidence with veto
→ easy !

any issue ?

T1, T2, V : logic pulses (“0/1” values)

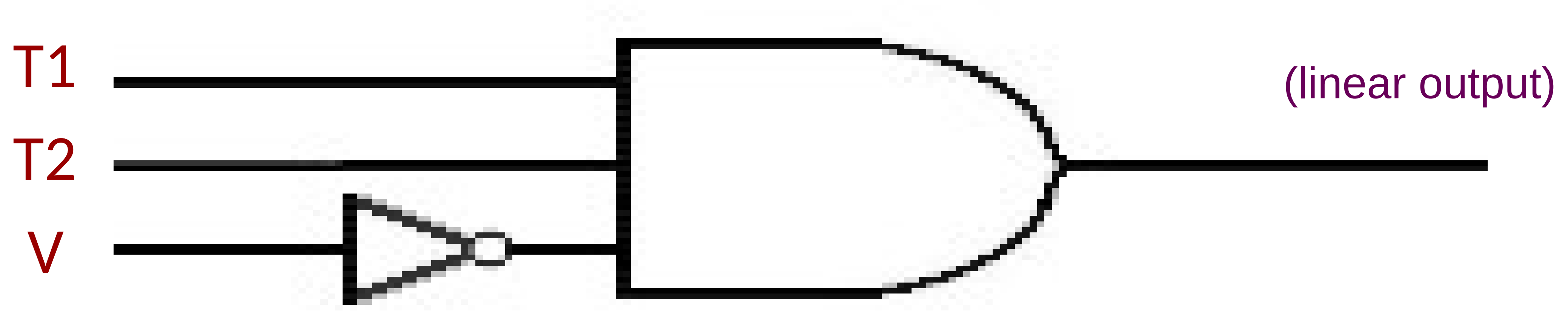


(anti-)coincidence with veto

→ easy !

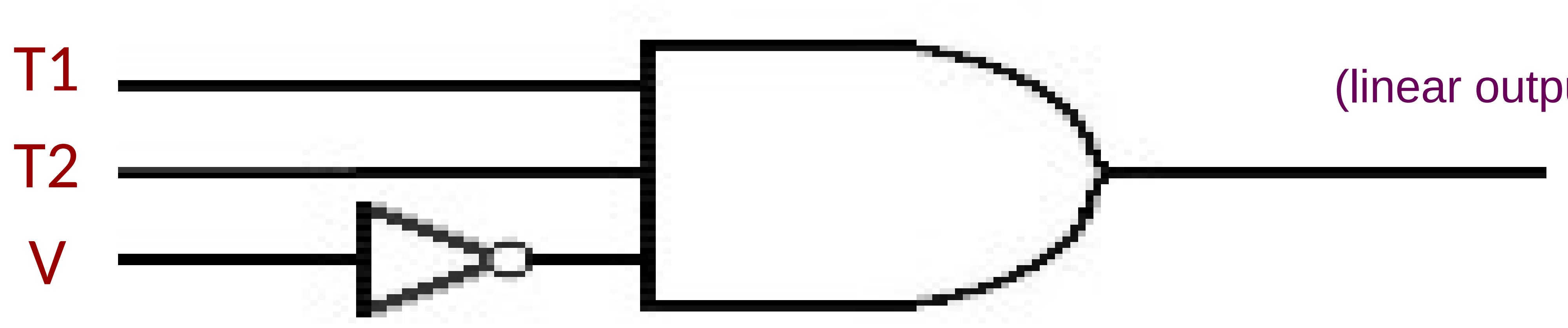
sure it works ?

(anti-)coincidence with veto



Not really !

(anti-)coincidence with veto



(linear output)

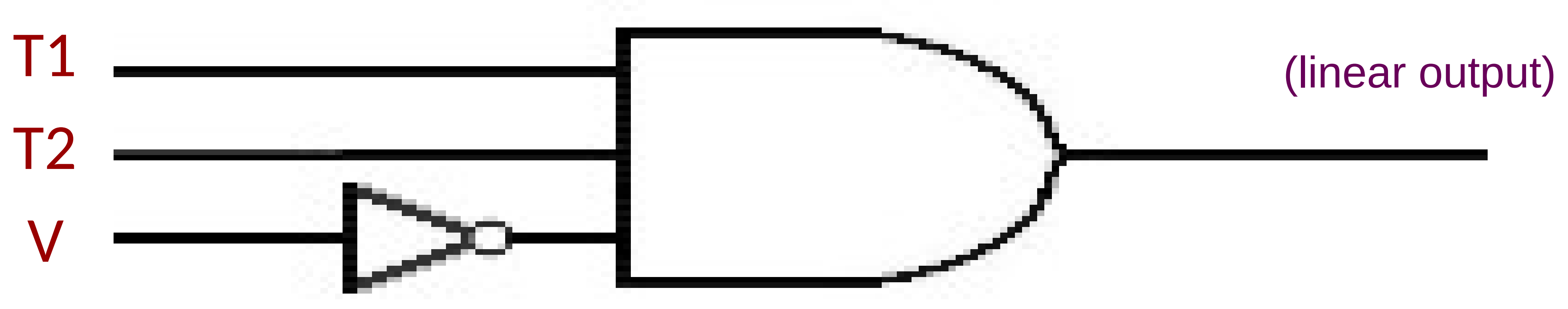
Not really !

Output signals will both:

a) jitter

b) fluctuate in duration

(anti-)coincidence with veto



Not really !

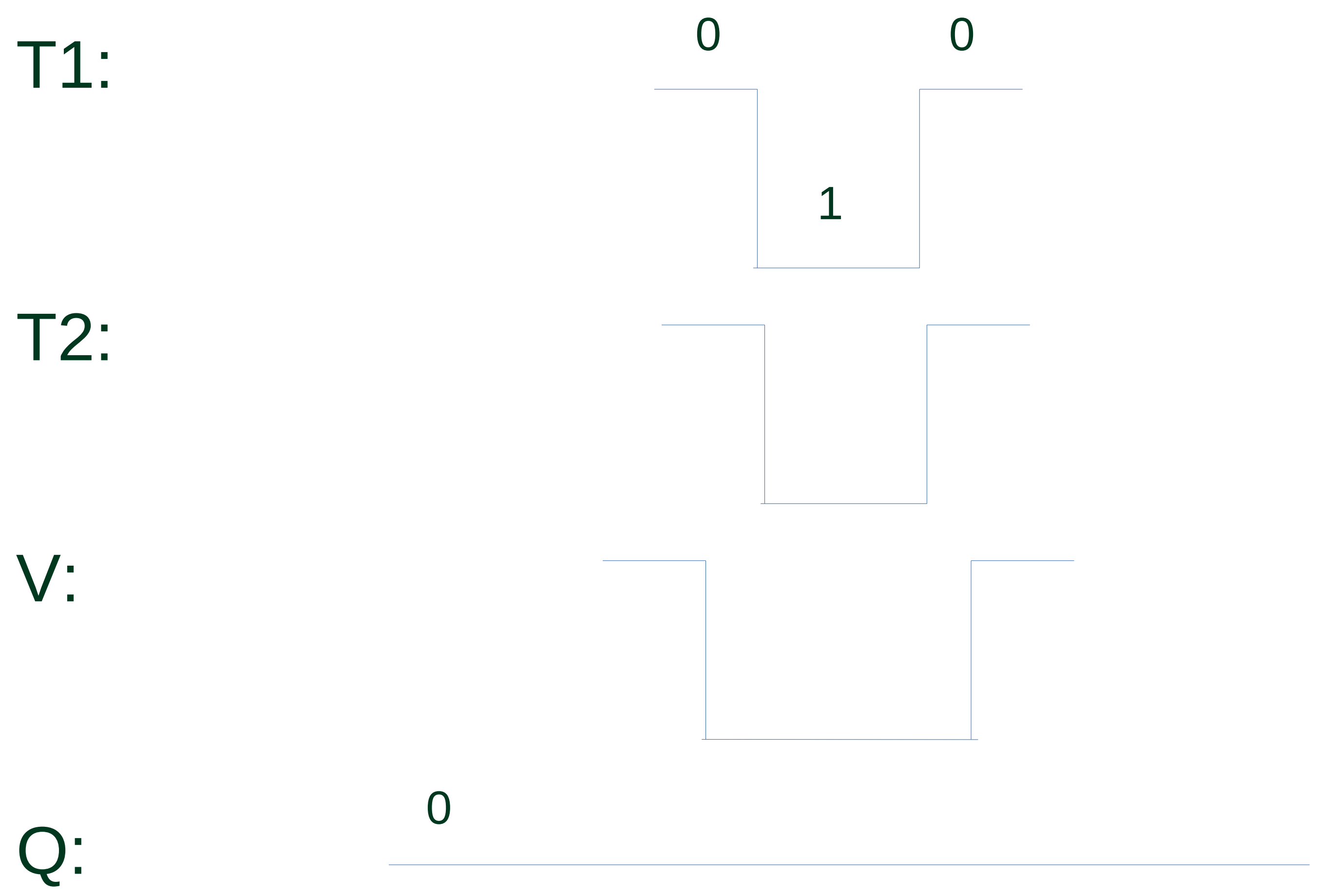
Output signals will both:

a) jitter

b) fluctuate in duration

Why ?

(good) in-time signals



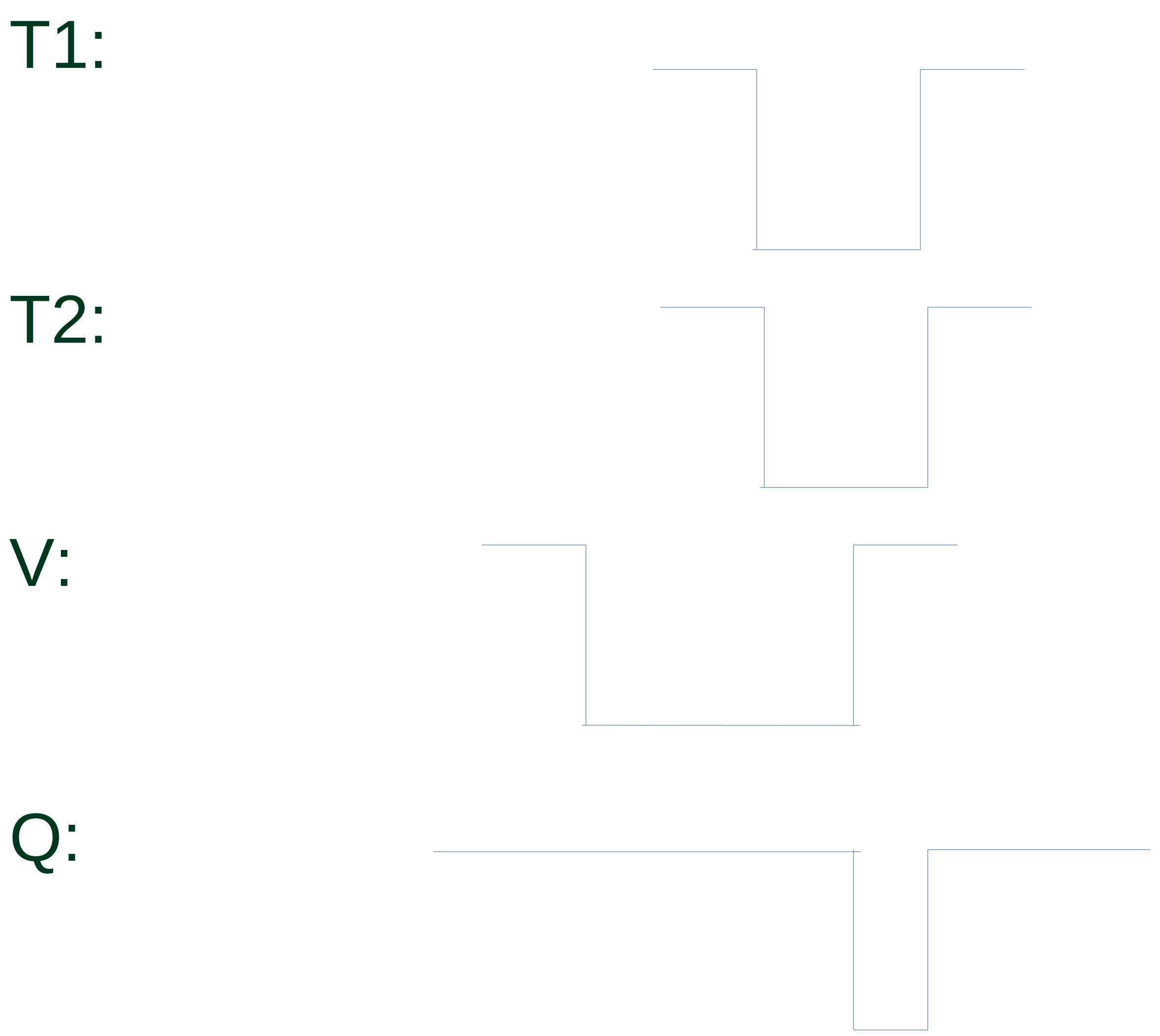
NIM signals:

0 ⇔ 0 V
1 ⇔ - 800 mV

1) T1 and T2 (almost) perfectly in time

2) When needed, V totally overlaps (in time) T1 and T2

(bad) out-of-time signals

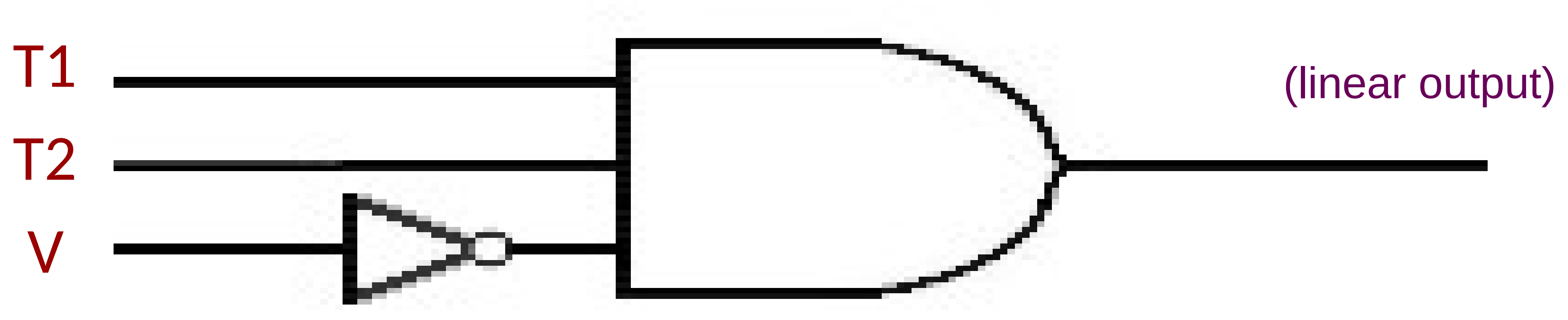


in some cases:

both wrong transition time and wrong duration

(anti-)coincidence with veto

combinatorial logic



Not really !

output signal will both:

a) jitter

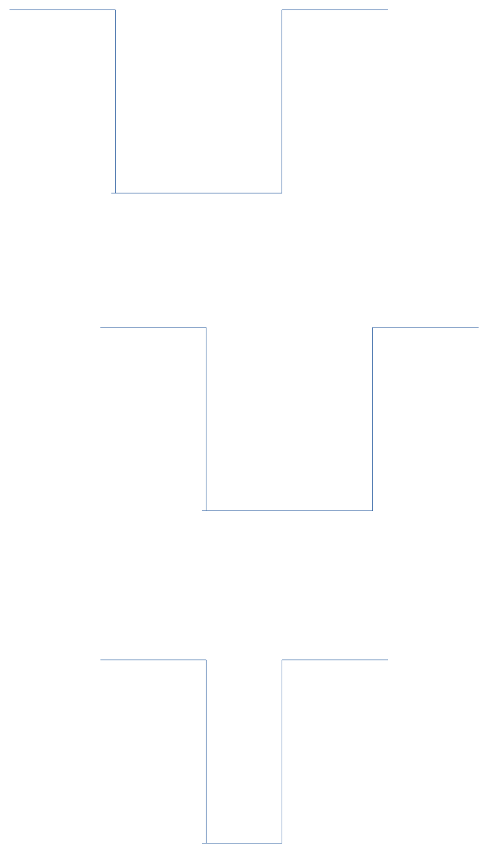
b) fluctuate in duration

because of independent signals from T1, T2, V

S1:

S2:

S1 and S2:



independent → uncorrelated (e.g. random) signals

→ even without veto, output needs to be formed

S...!

Simple random coincidences are sufficient to break your “perfect” trigger

S...!

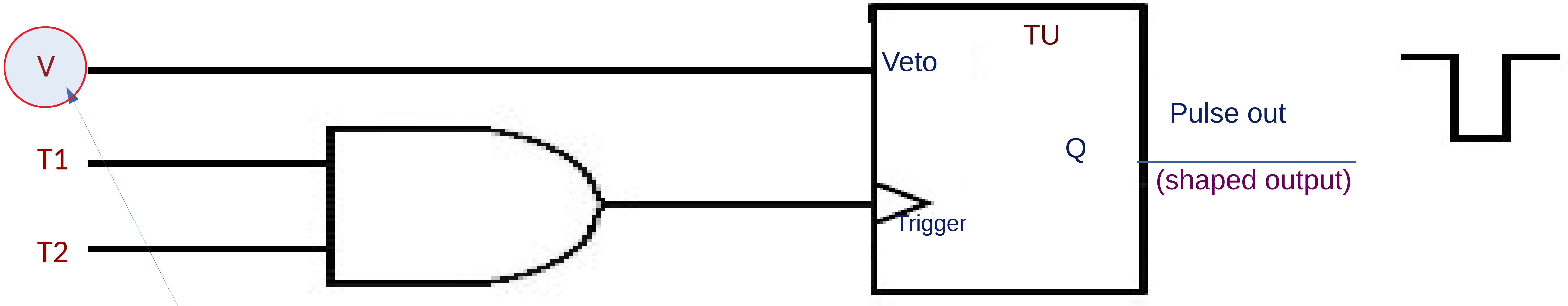
Simple random coincidences are sufficient to break your “perfect” trigger

can’t even blame high rate, pileup, …, locusts

(anti-)coincidence with veto

combinatorial logic

sequential logic



TU = Timing Unit

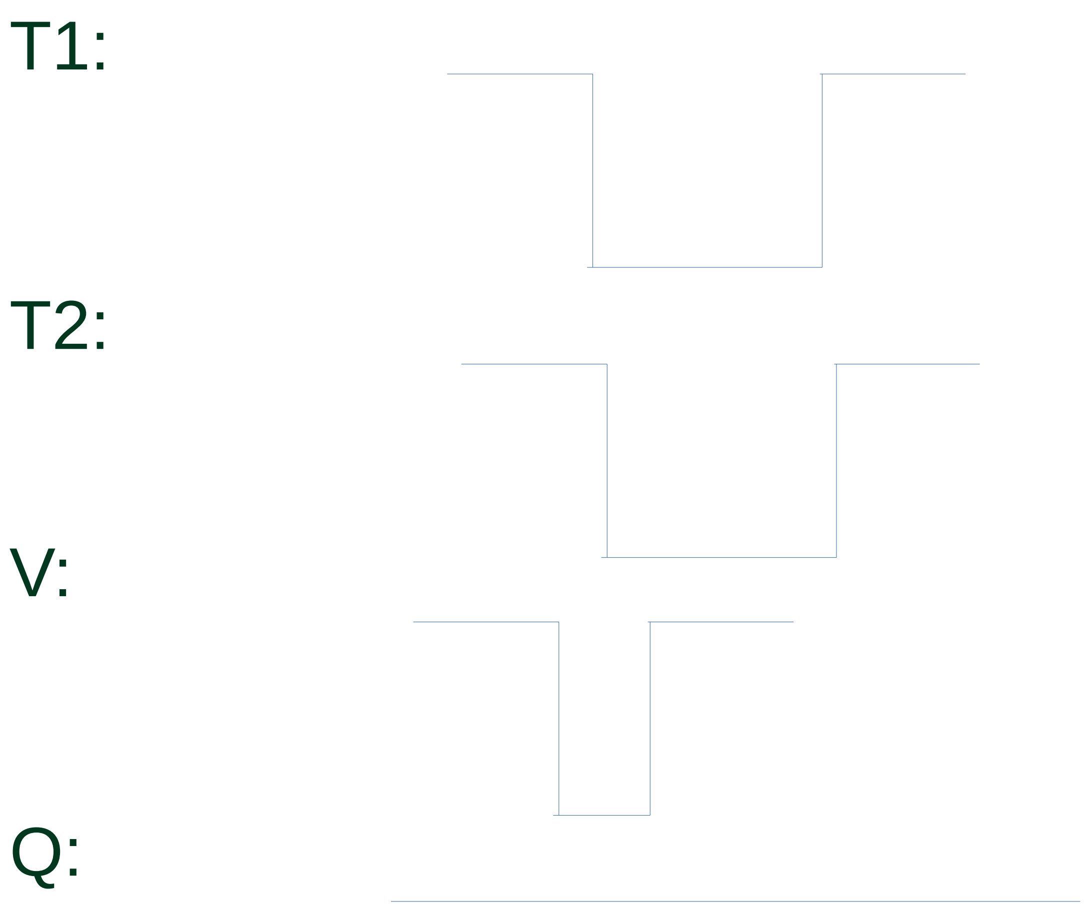
aka Monostable Multivibrator

aka One-Shot Pulse Generator

can also be a busy signal → busy logic

much better !

now



→ V only need to overlap transition region (0→1) of T1 and T2

→ lower dead time introduced by V signals

T1 and T2 → transition time

TU → duration time

T1 and T2 → transition time

TU → duration time

Q: What the relevant information?

first lesson(s)

trigger signal:

1) should be formed!

→ pulse with predefined duration

2) veto/busy should block pulse generation

3) V & T signals must be aligned in time

4) 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)

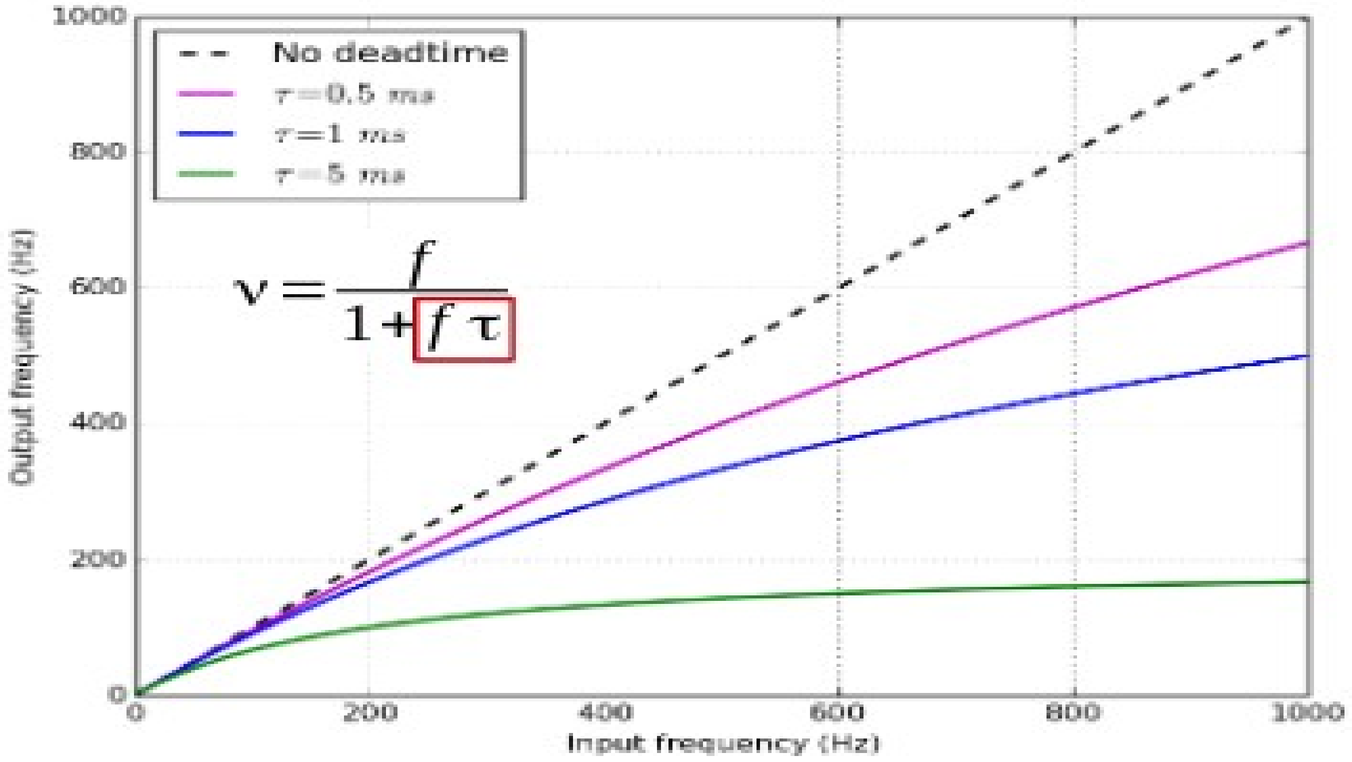
→ throughput fluctuations

(correlated noise, …)

→ dead-time

deadtime (from Andrea’s introduction)

Deadtime and efficiency

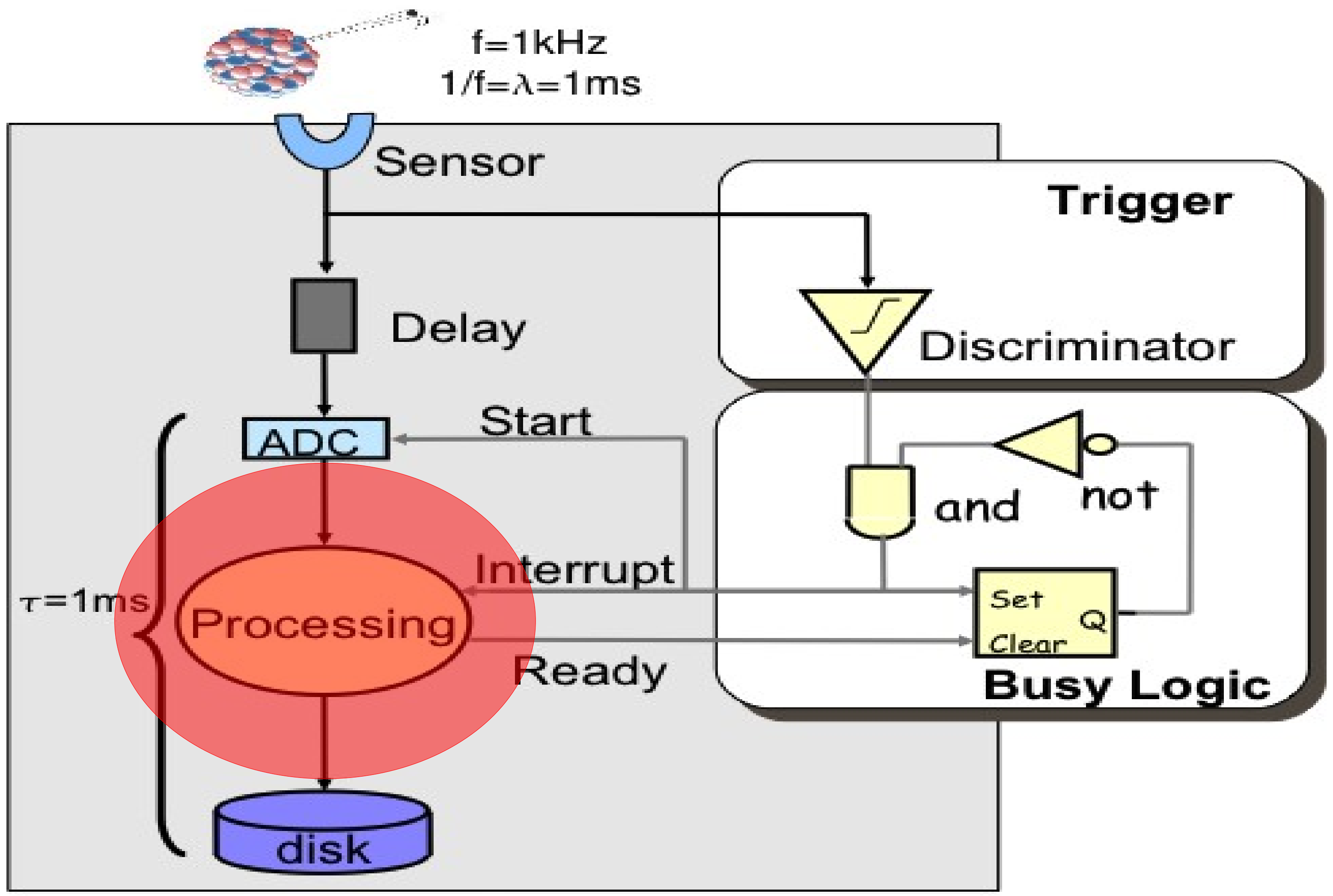


$$\nu = \frac{f}{1+f\,\tau}$$

- In order to obtain ε~100% (i.e.: ν~f) → fτ << 1 → τ << λ
  - E.g.: ε~99% for f = 1 kHz → τ < 0.01 ms → 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?

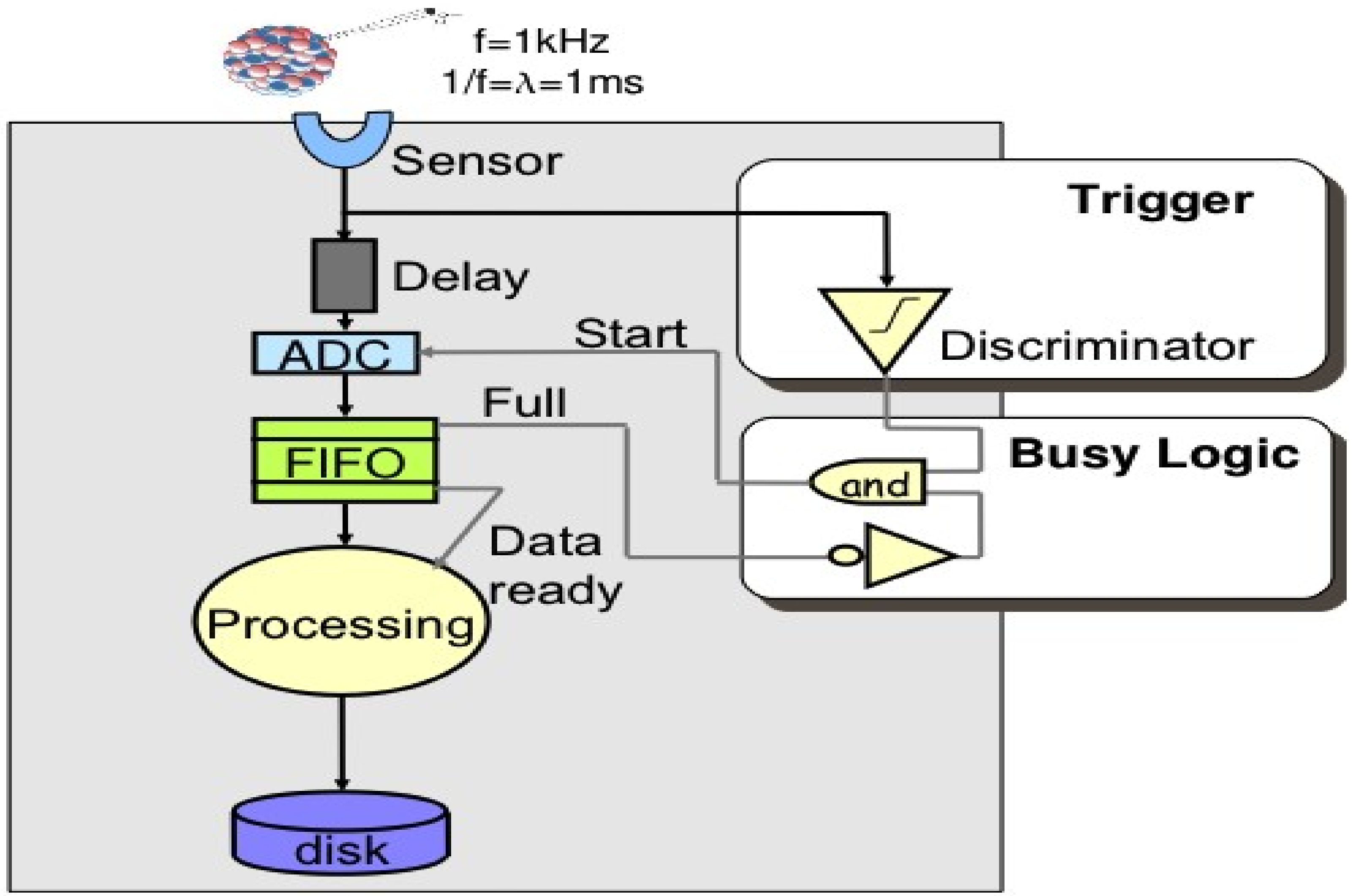
deadtime → de-randomise

- Processing → bottleneck ?



$(f \cdot \tau) \sim 1 \rightarrow$ deadtime $\sim 50\%$

- Buffering → decouple problems



What the impact ?

$(f \cdot \tau) \sim 1 \rightarrow$ deadtime?

FIFO

First-In First-Out memory:

1) independent read/write (sequential) access

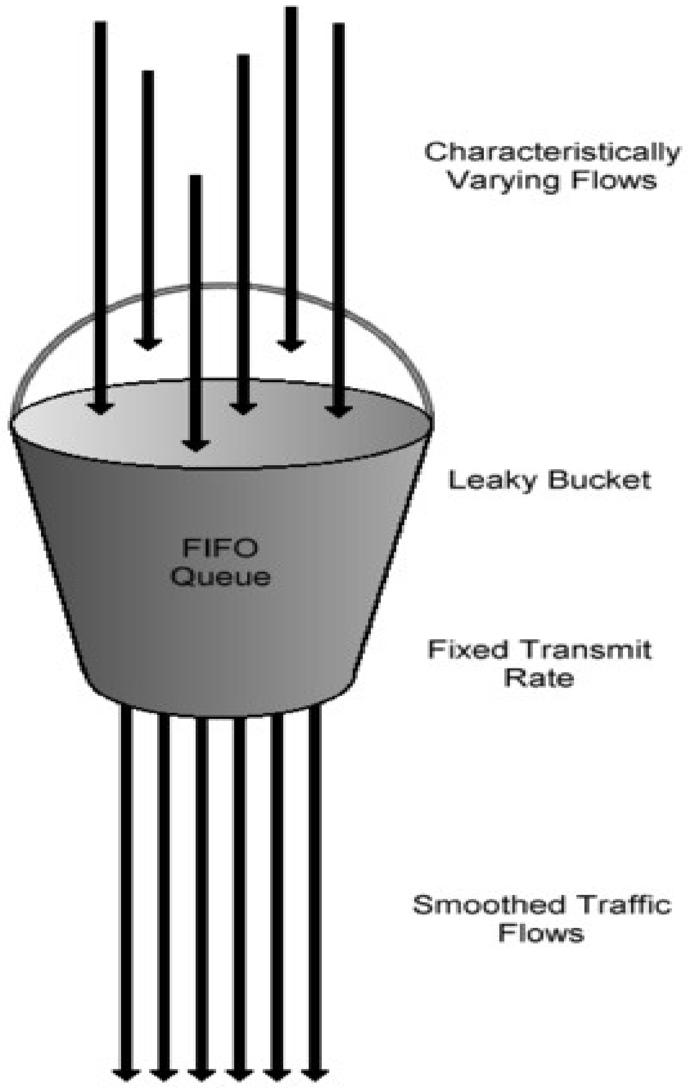
2) may be hardware or over RAM

if RAM better Dual-Port RAM

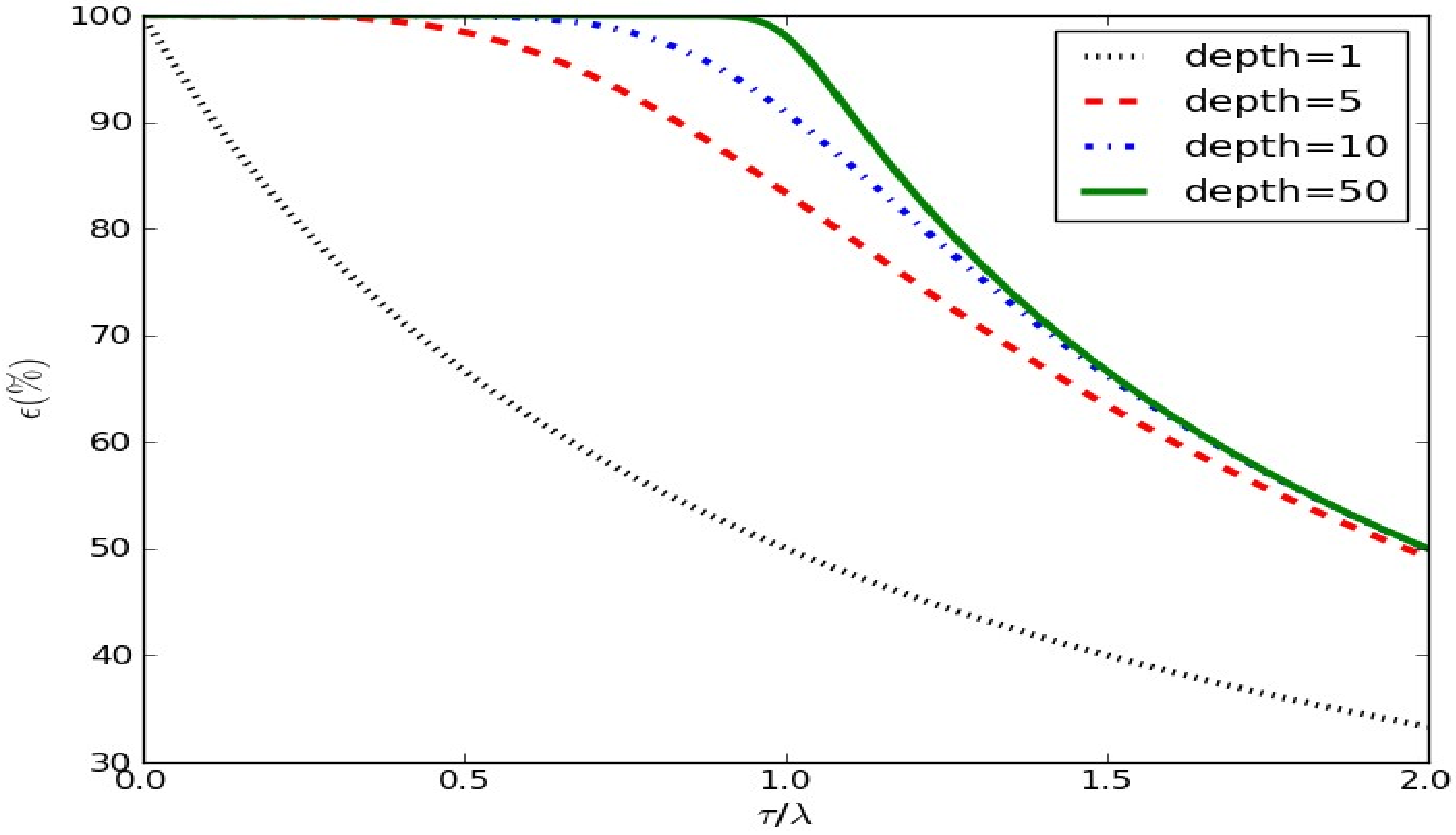
buffering solve all problems ?

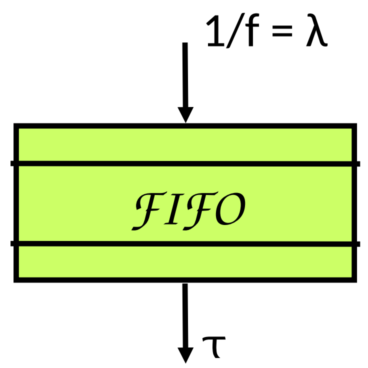
- FIFO (front-end buffers)
  1) filling at very variable input flow
  2) emptying at smoothed output flow
  $\rightarrow$ the Leaky-Bucket problem

  Q: how often may overflow?



de-randomisation





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

deadtime in trigger system

1) Simple deadtime: avoid overlapping (conflicting) readout window

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

→ different subdetector & different front-end elx

→ different algorithm/parameters

ATLAS deadtime @ end of run 2

1) Simple deadtime: 4 LHC BC [i.e. 100 ns] after any LVL-1 trigger

2) Complex deadtime:

2.a) four leaky-bucket algorithms

[two params: bucket size S (in number of events), readout time R (in BC units)]

1) 15 / 370 for LVL-1 Calorimeter and CSC readout

2) 42 / 384 for TRT readout

3) 9 / 351 for LAr readout

4) 14 / 260 for LVL-1 Topo readout

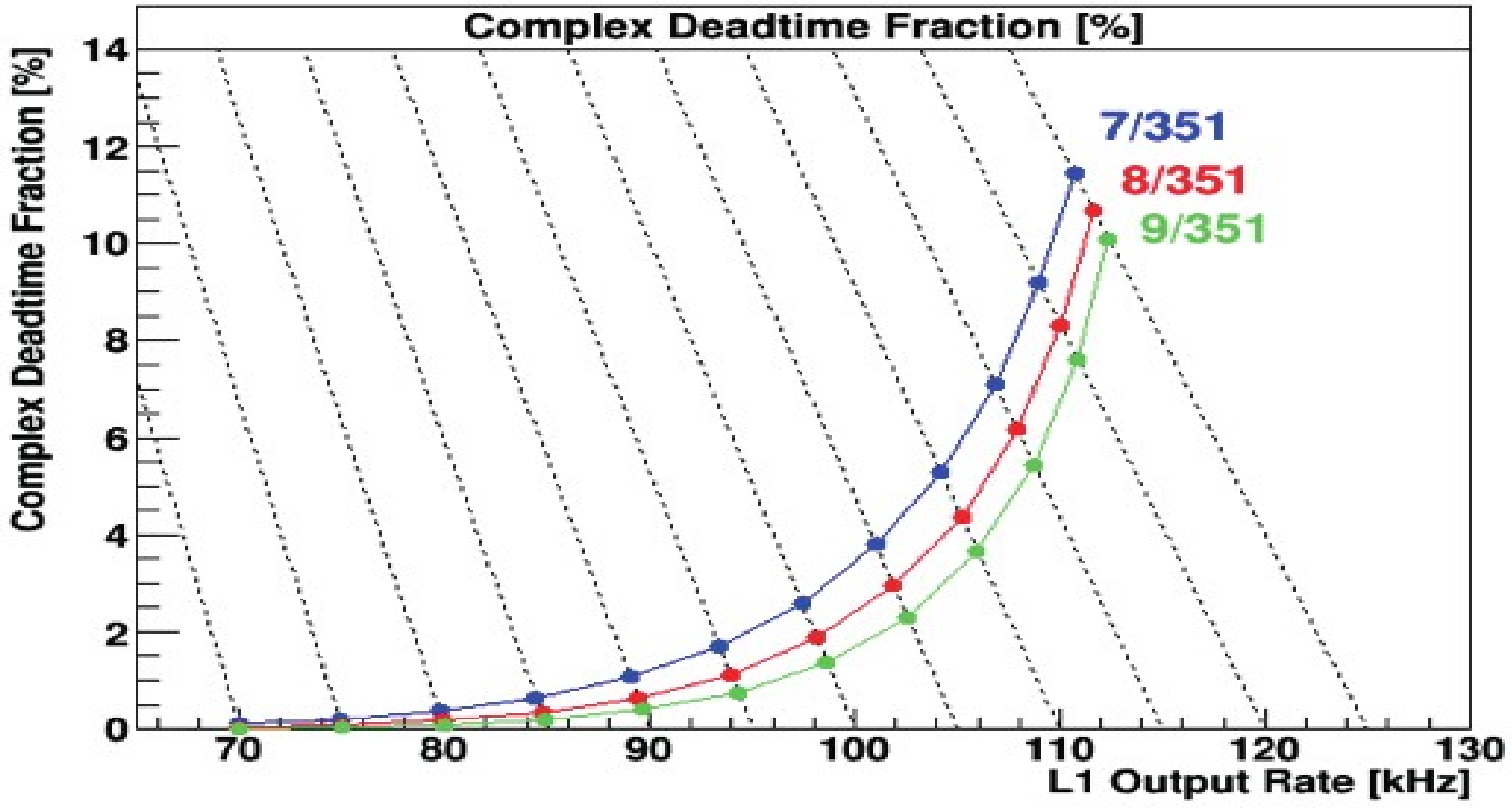
2.b) one sliding-window algorithm

< 16 LVL-1 signals in any 3600 BC sliding window

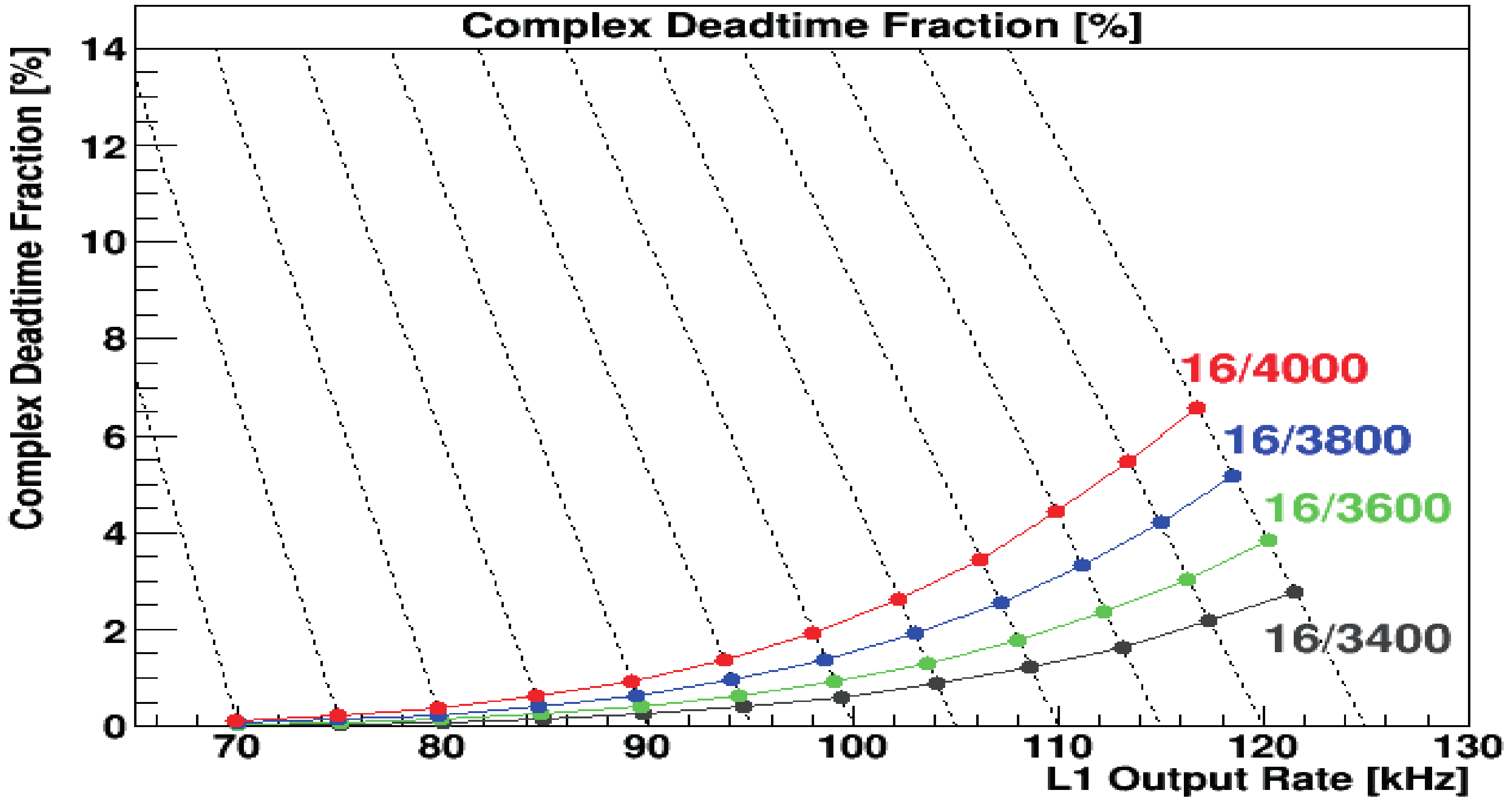
ATLAS deadtime @ end of run 2

Total deadtime @ 90 kHz trigger rate < 2%

Leaky bucket (LAr readout)

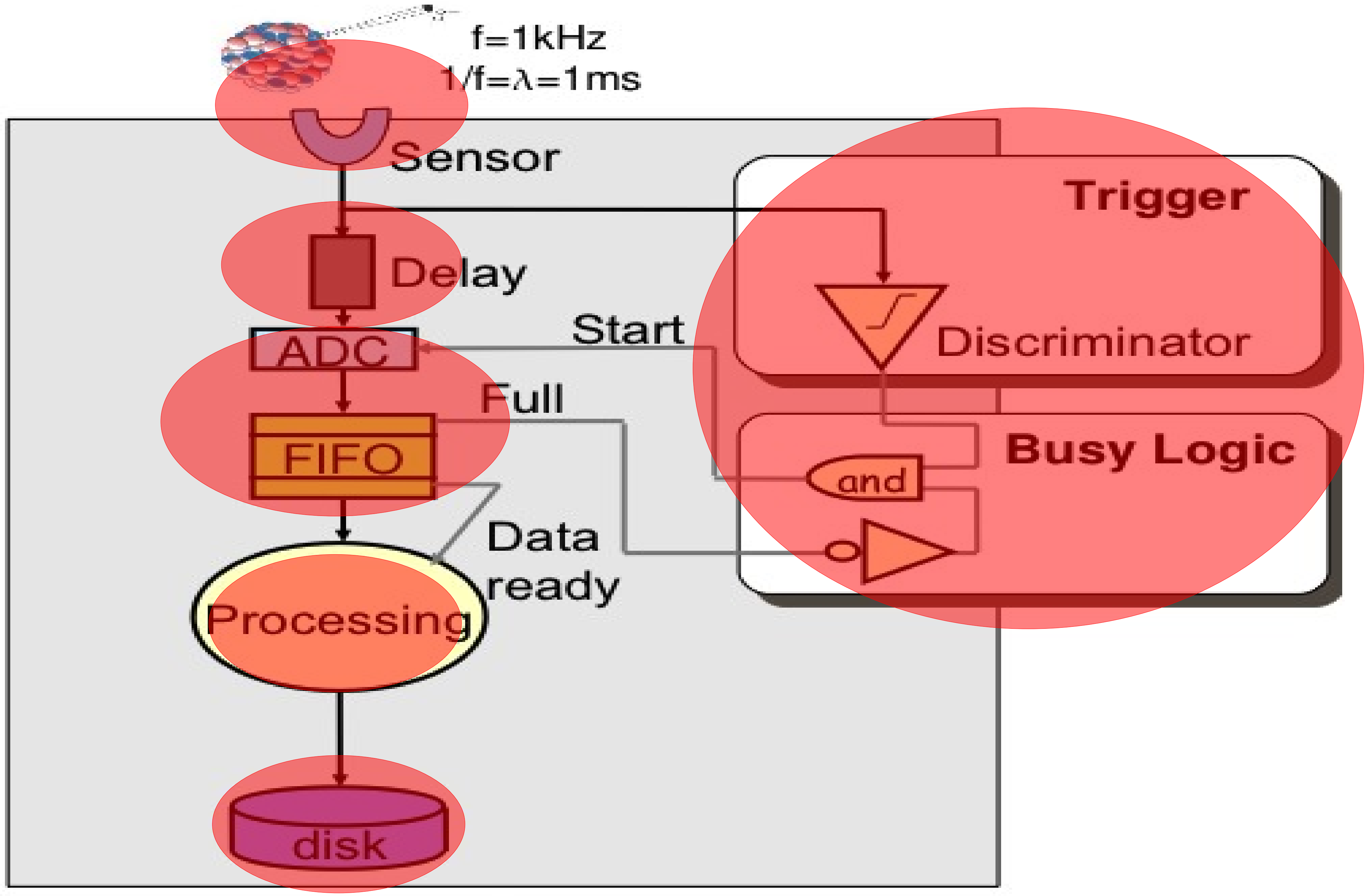


Sliding window (SCT readout)



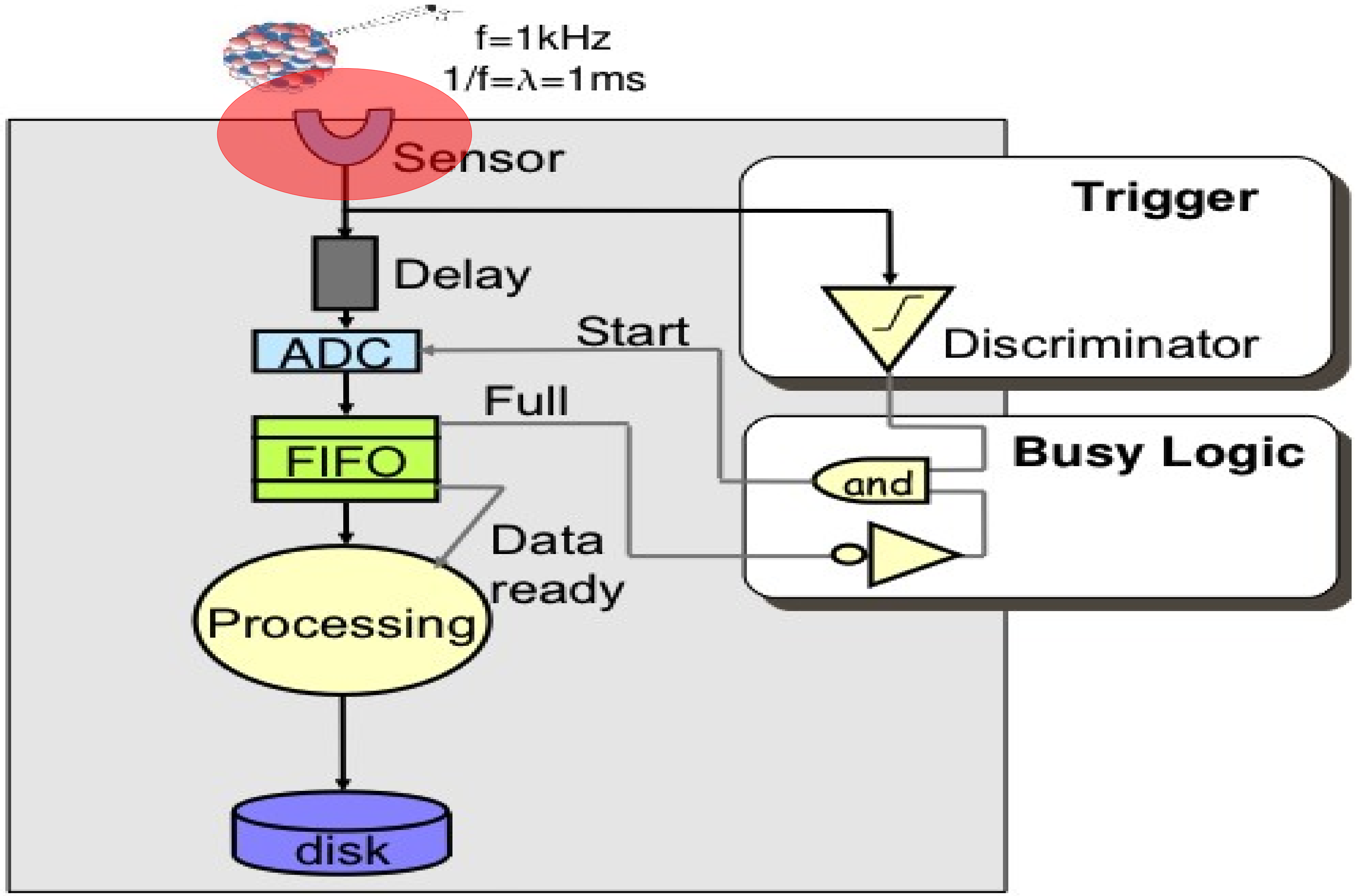
game over ?

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”

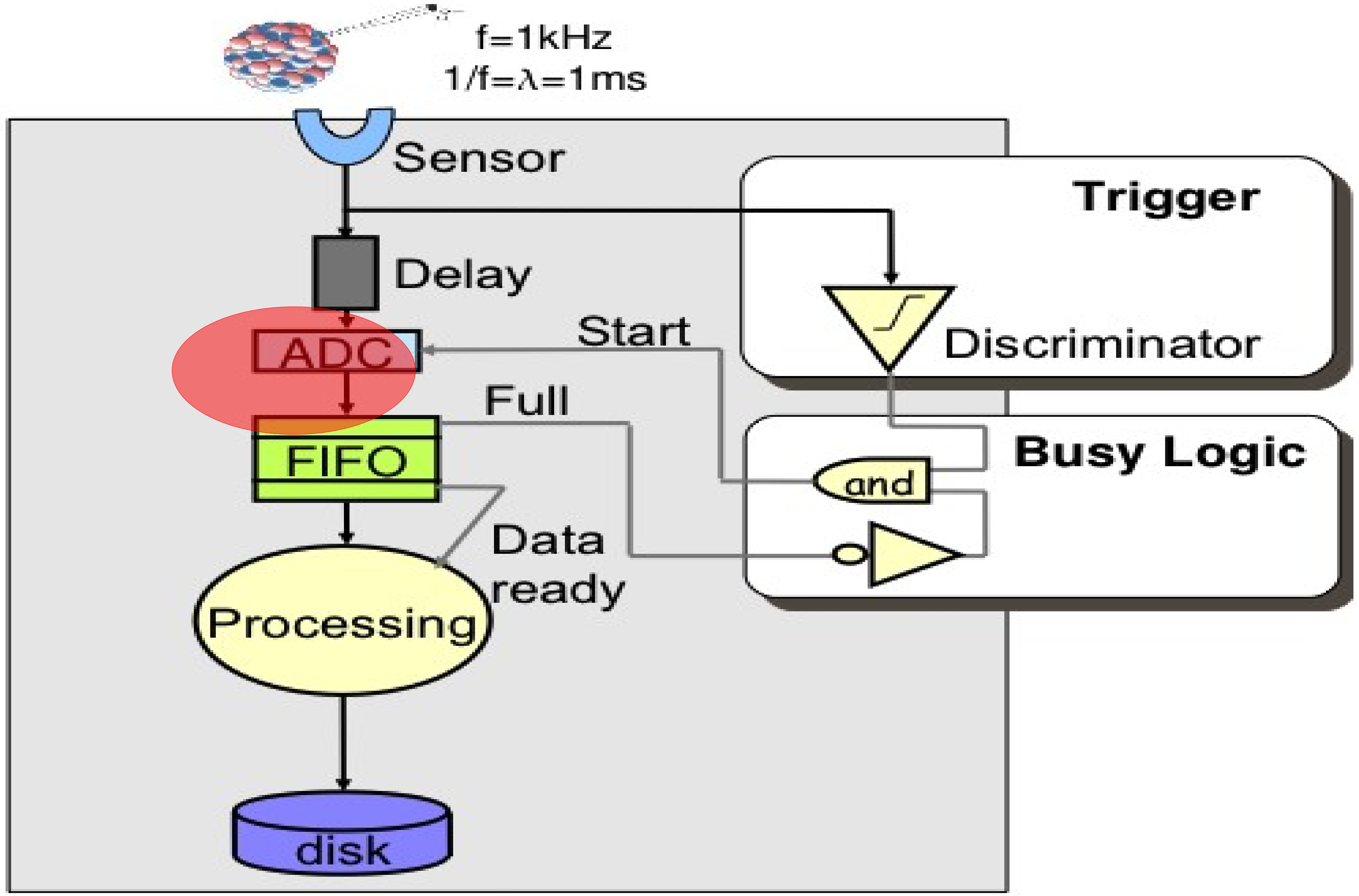


→ ADC

- A/D conversion also limited
- Fast ADC
  - → # of bits (resolution)
  - → power consumption
- Alternatives:

  analog buffers

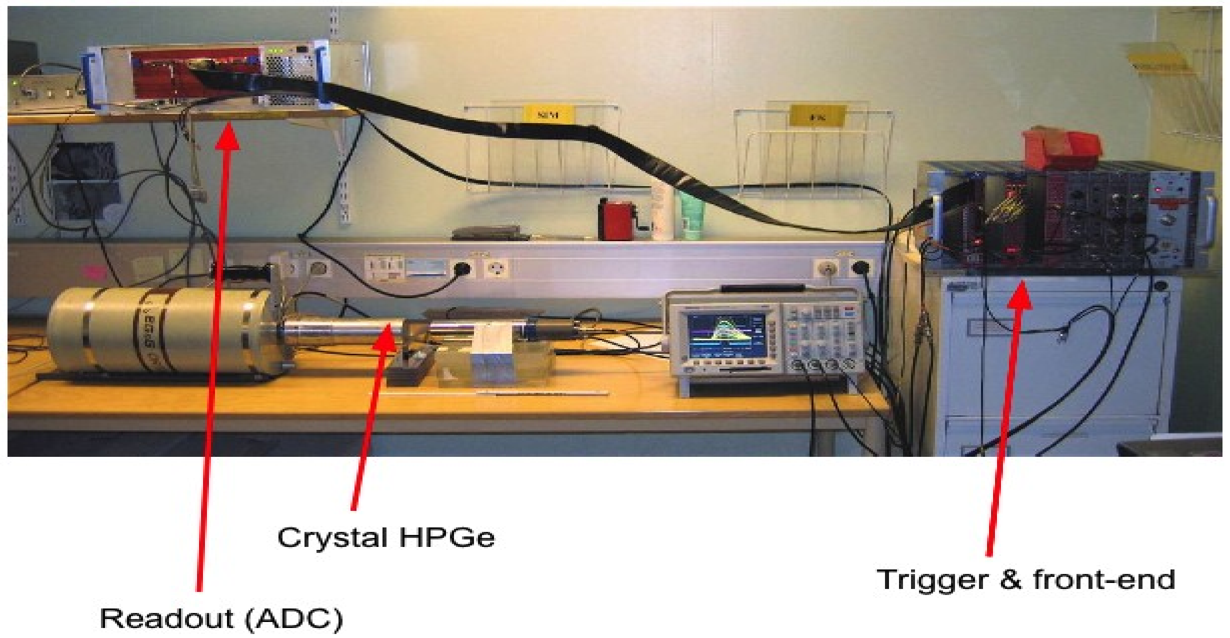
  (e.g. switched capacitor arrays)
- You may need integration (or sampling) over quite some time

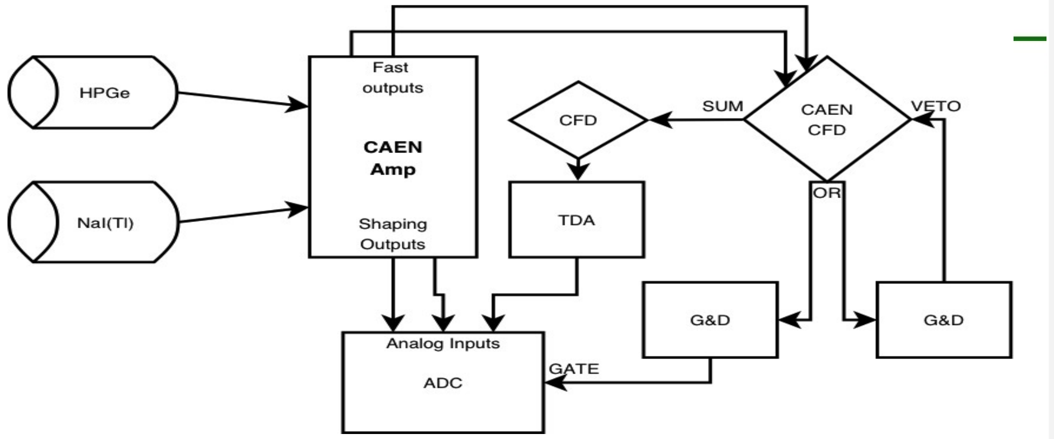


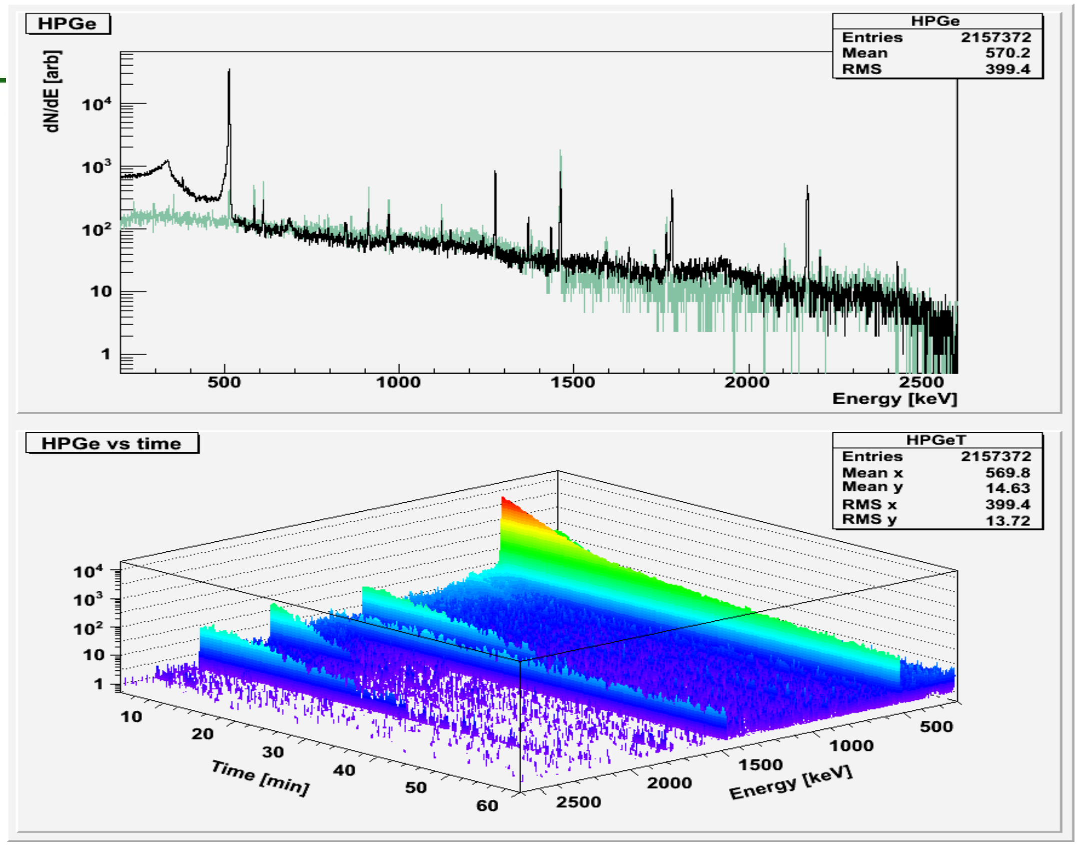
an example

- 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

Ge crystal for isotope identification

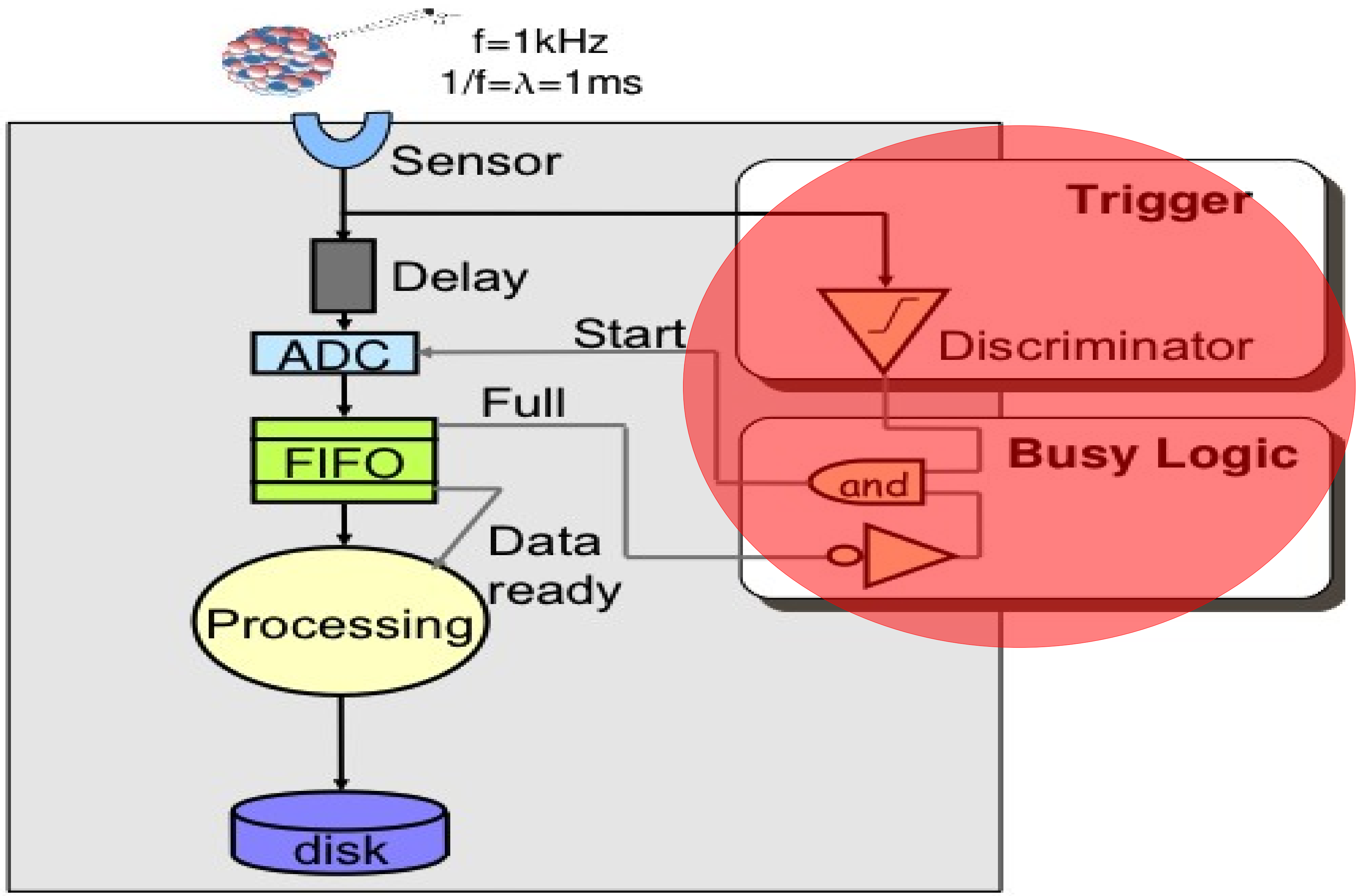






→ trigger latency

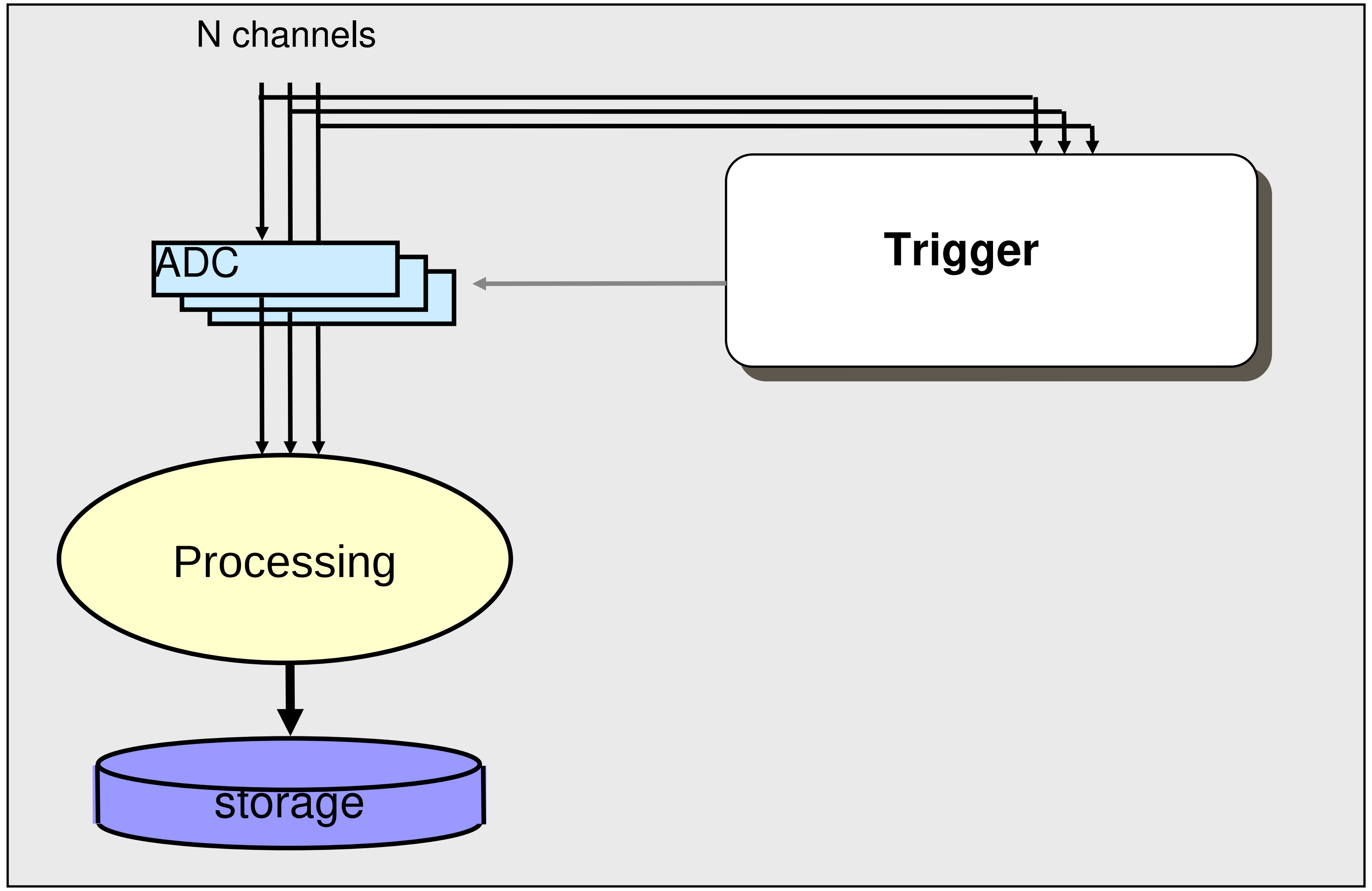
- simple trigger: ~fast
- complex trigger logic: not obvious [even when all in hw]
- some trigger detectors may be far away / slow → latency
- trigger signal is one: all information at a single point
  - in one step:
    too many cables
  - in many steps:
    delays



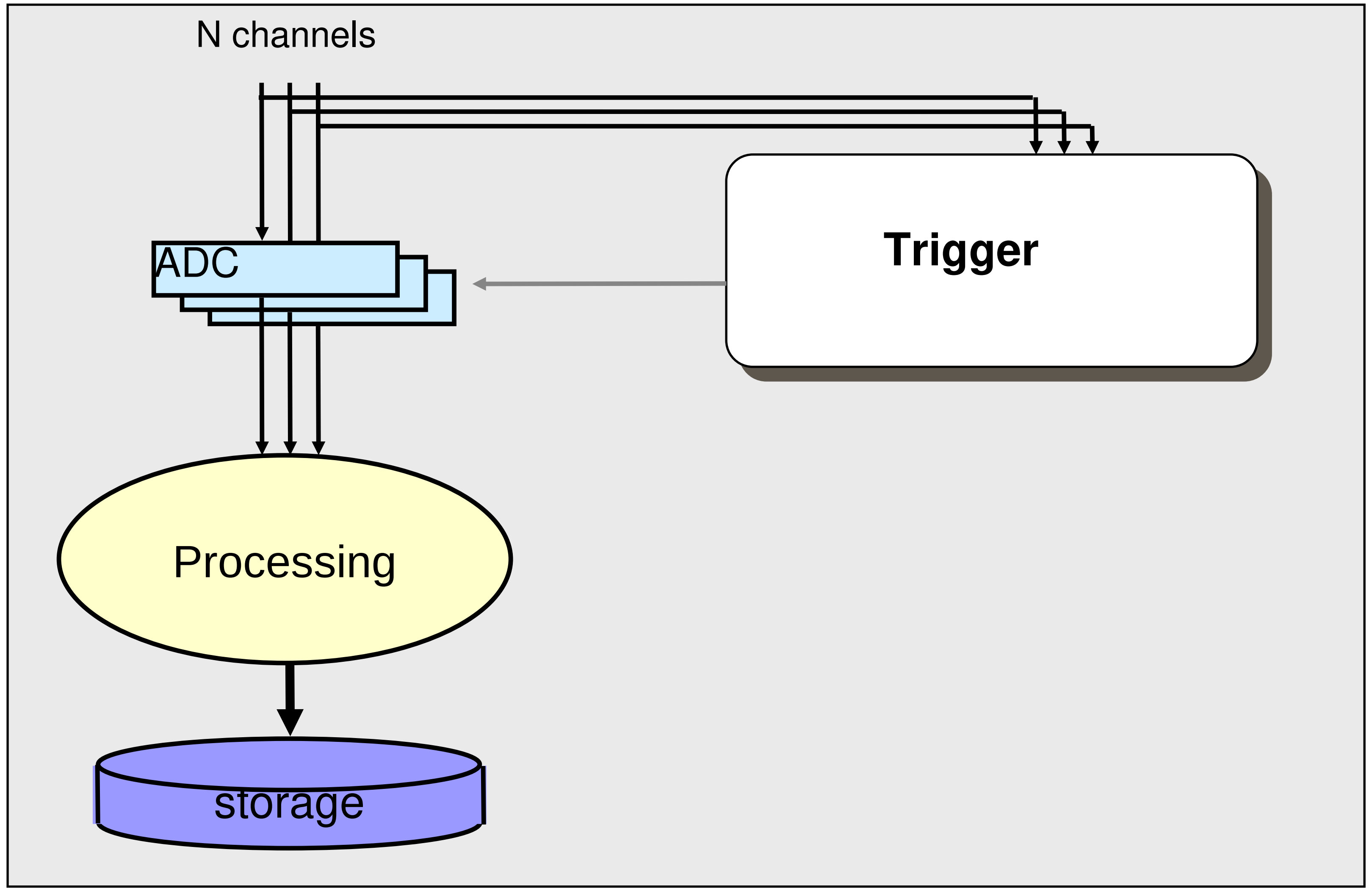
→ discrete modules: ~ 5-10 ns delay → tot. latency ≥ 20-30 ns

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

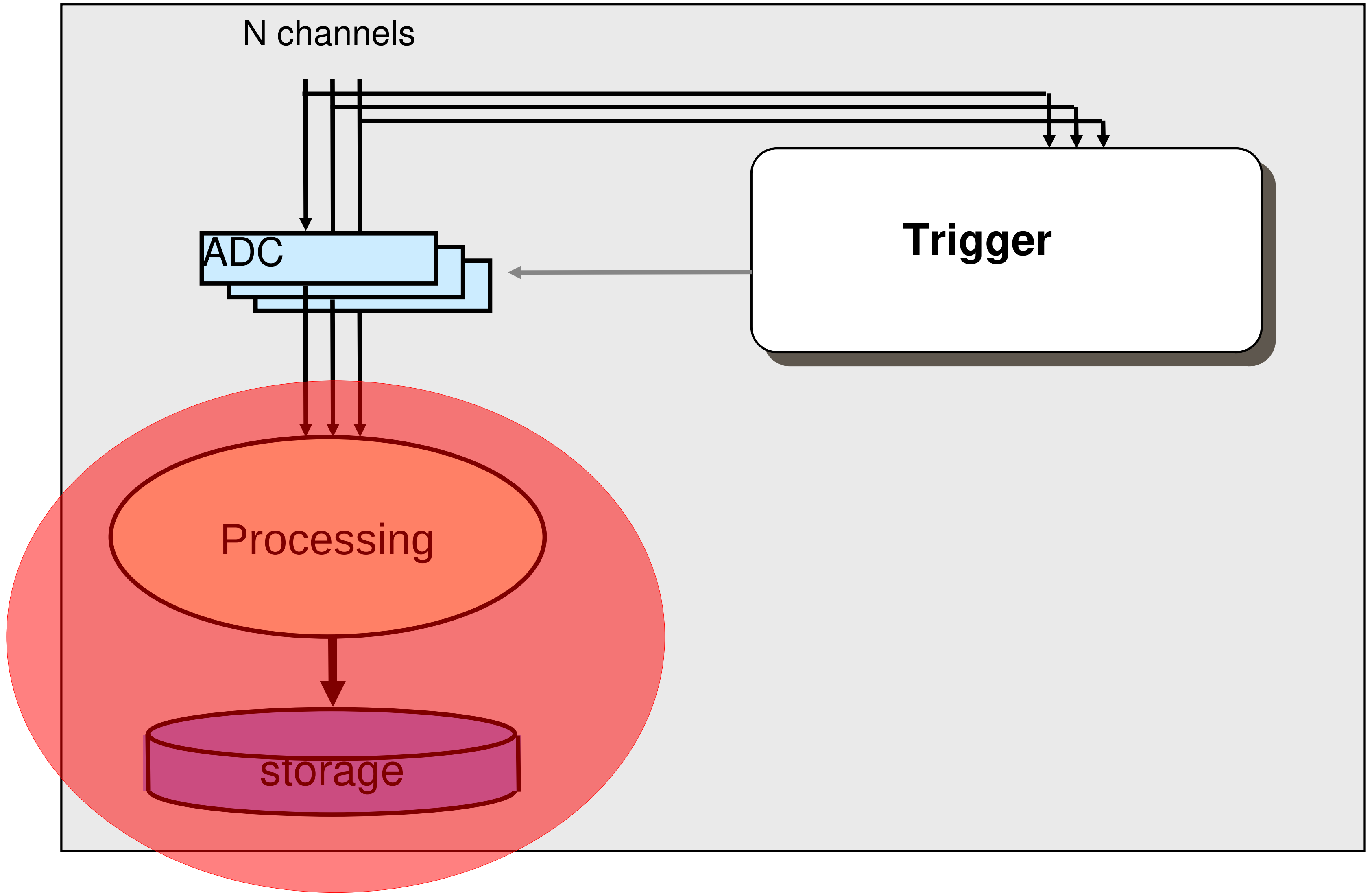


multi-channel, single-PU system



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

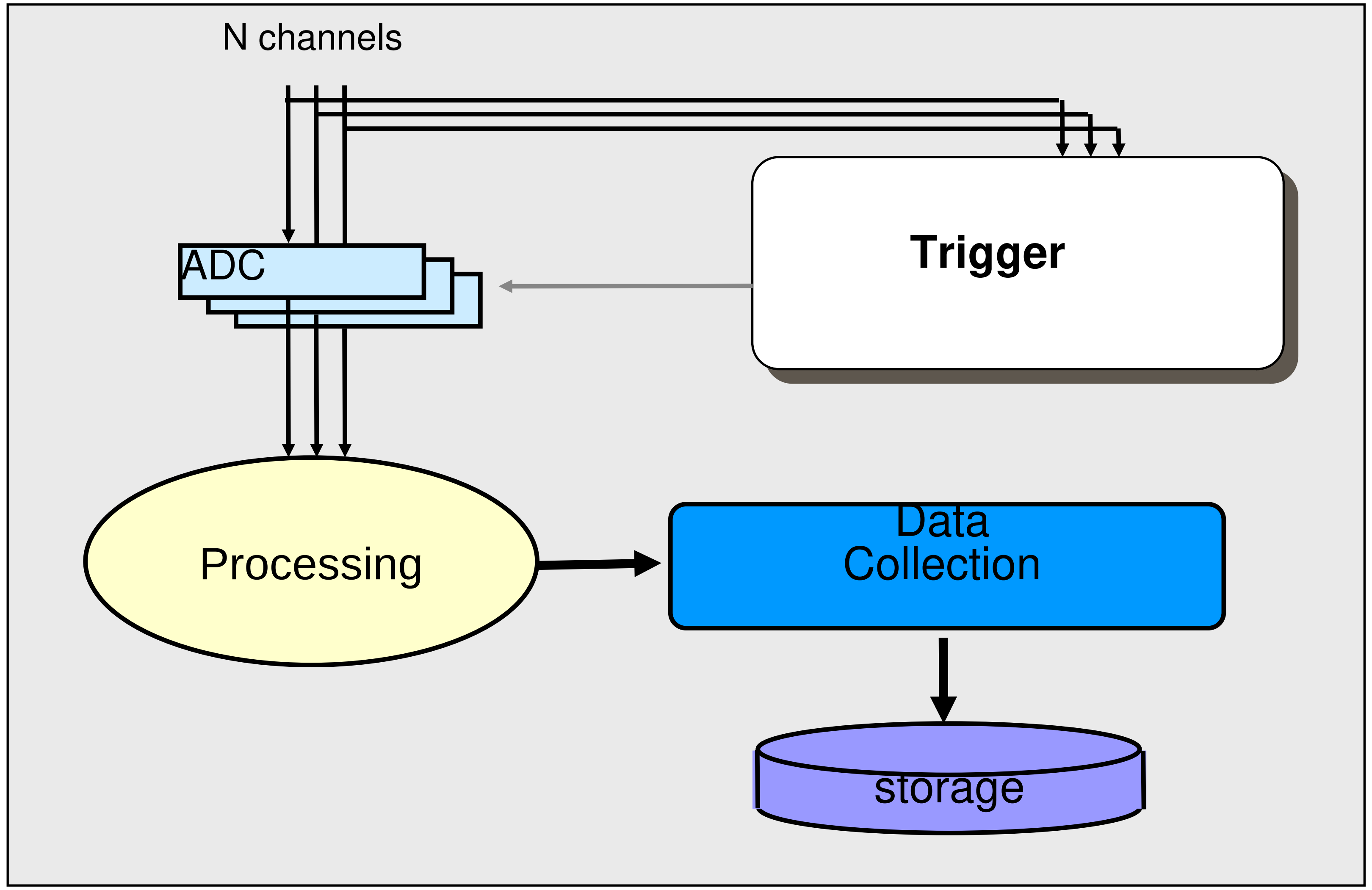
bottlenecks: PU and storage



- 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!

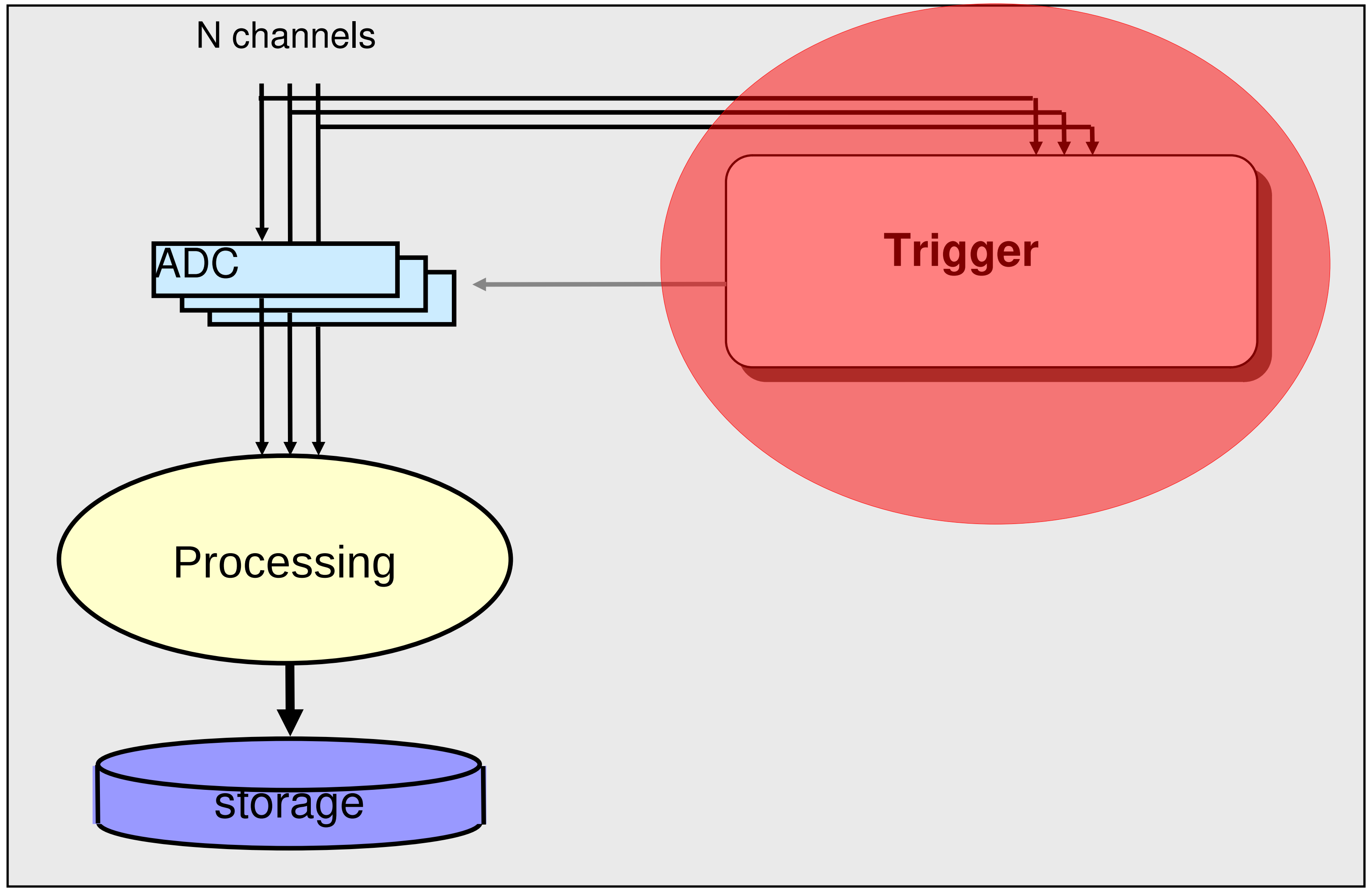
Laptop SATA disk: ~100 MB/s
USB2: ~60 MB/s

→ decouple storage from PU



- data transfer data → dedicated “Data Collection” unit to format, compress and store
- more room for smarter processing or decreased deadtime 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)

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 → Si beam telescope

… always evolving

acquiring: waveforms, total charge, time information

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 → Si beam telescope

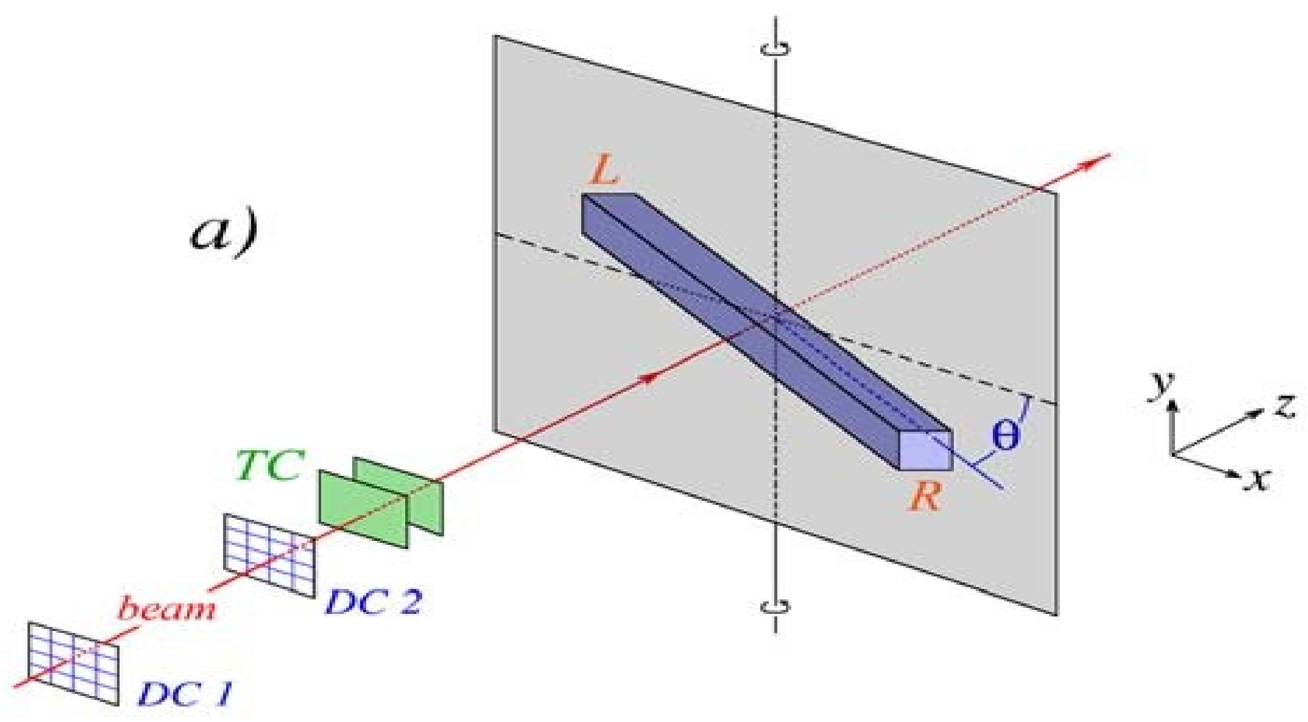
… always evolving

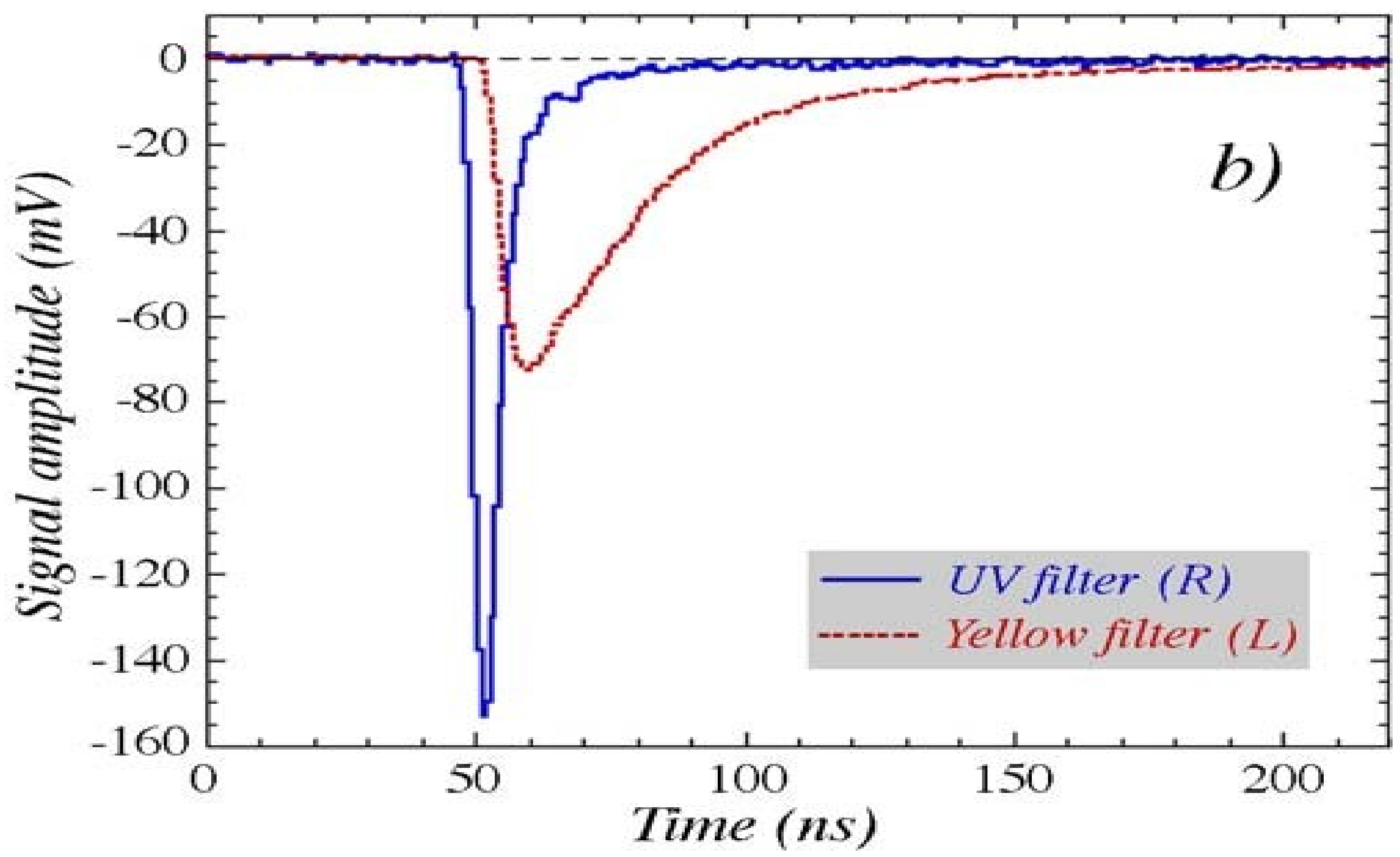
sometime running with 2 or even 3 independent DAQ systems

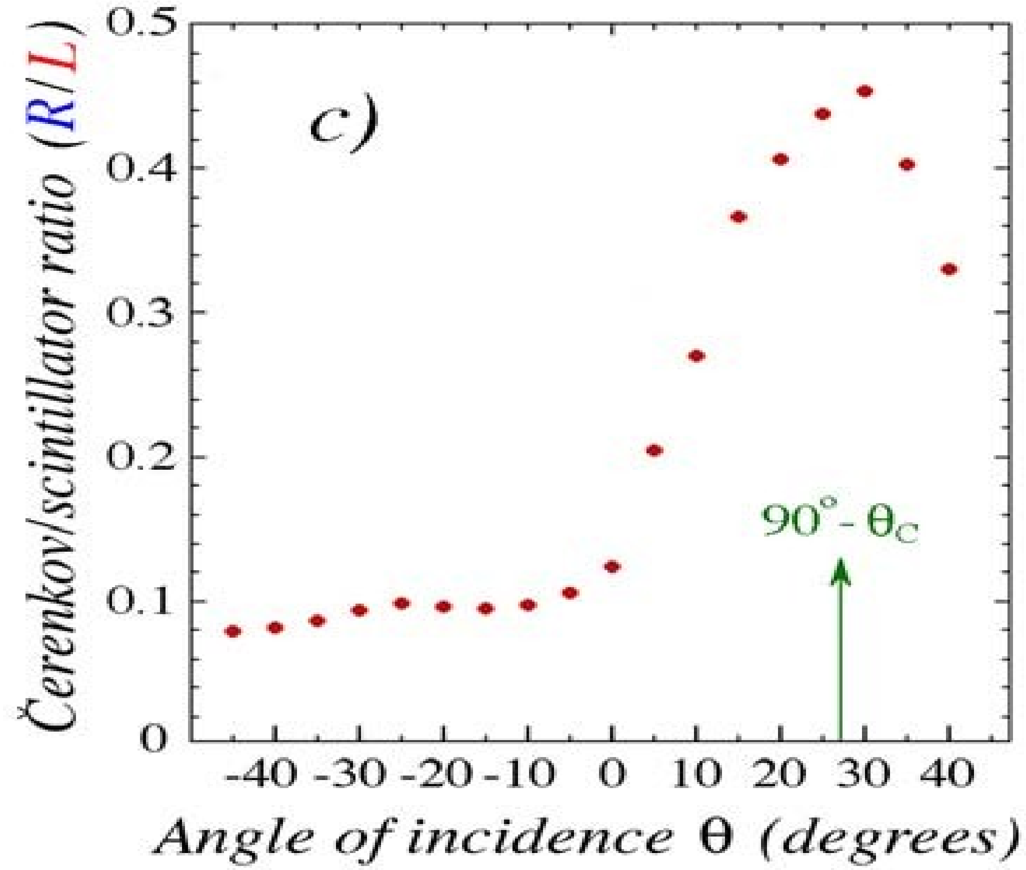
→ trigger and busy signals used for DAQs' synchronisation

→ offline event building

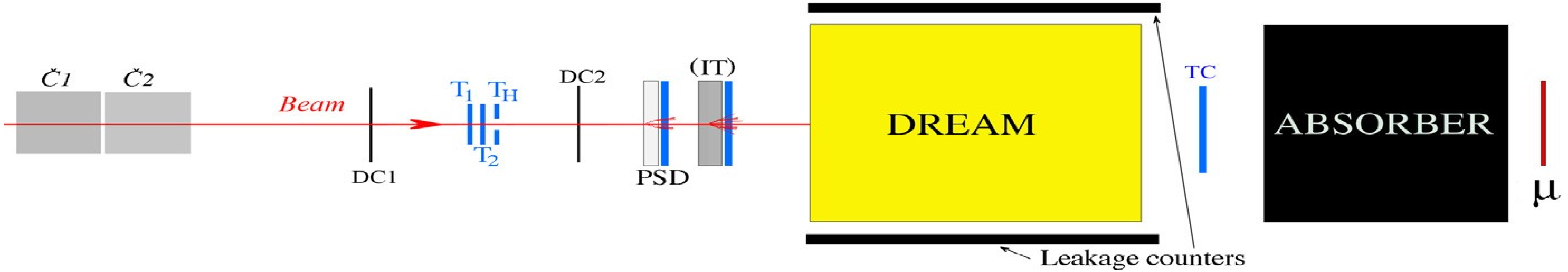
DREAM/RD52: crystal prototype

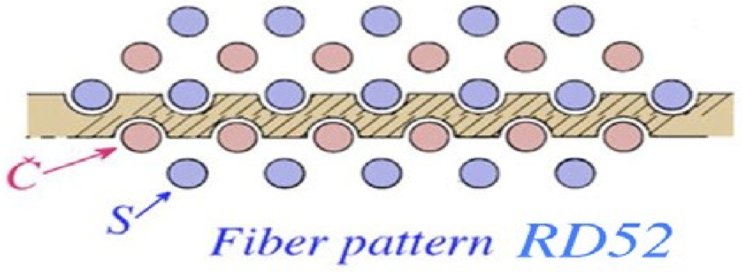


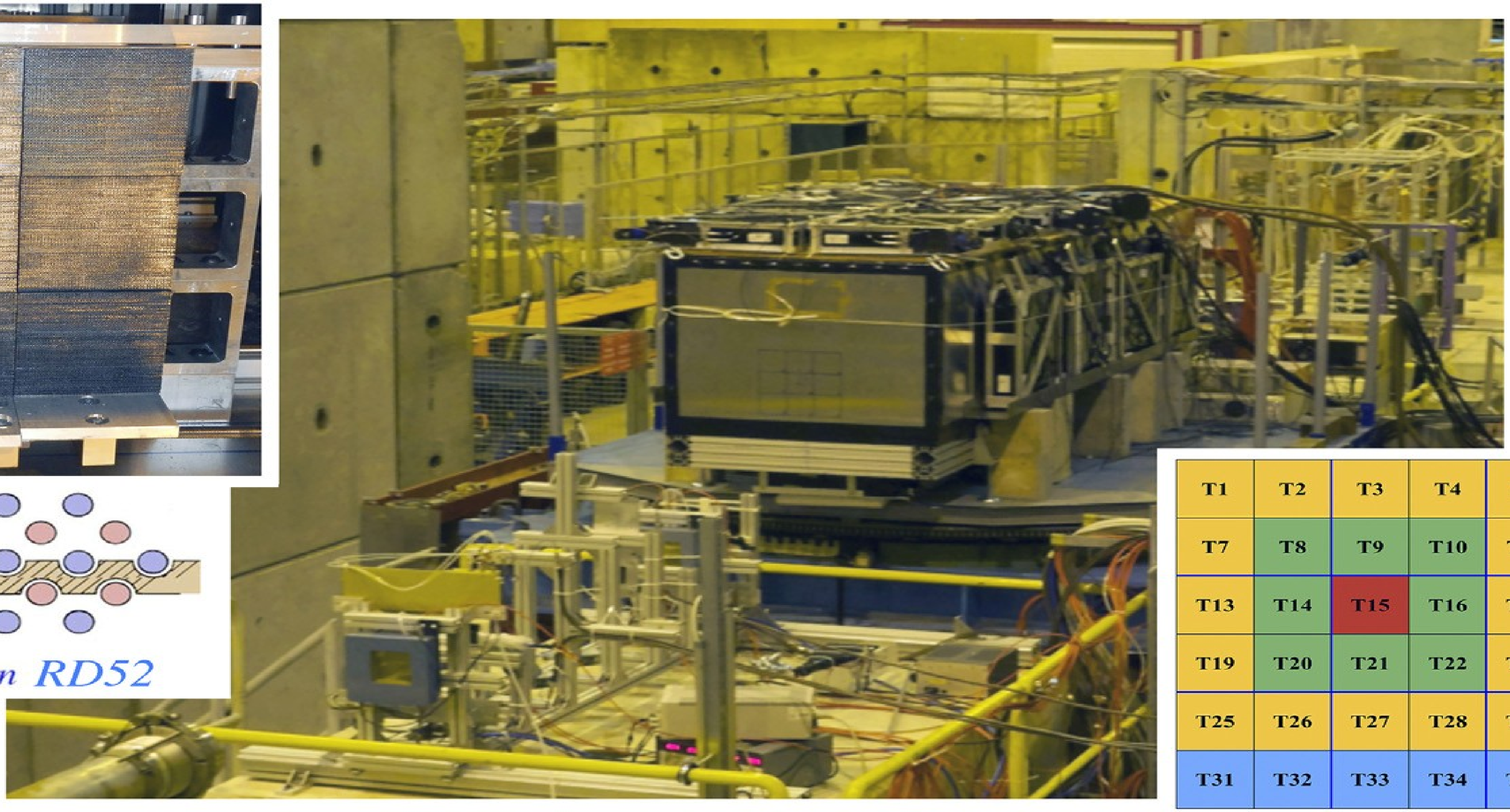




DREAM/RD52: fibre-sampling prototype







T1	T2	T3	T4	T5	T6
T7	T8	T9	T10	T11	T12
T13	T14	T15	T16	T17	T18
T19	T20	T21	T22	T23	T24
T25	T26	T27	T28	T29	T30
T31	T32	T33	T34	T35	T36

Ring 1 Ring 2 Ring 3

DREAM (2006→): a testbeam case

a possible SPS cycle

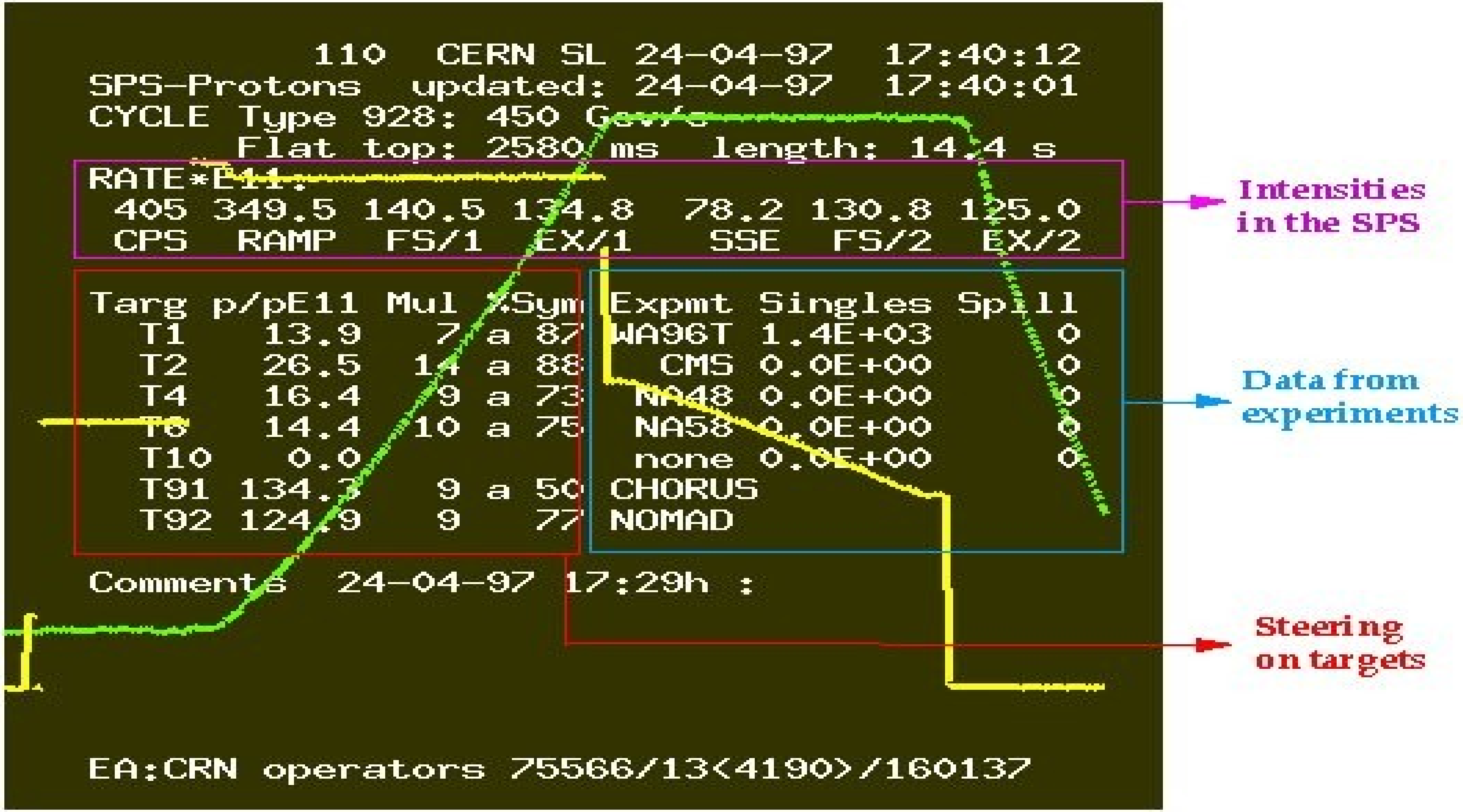
duty cycle:

~2 s / 14.4 s

(flat top)

flat top

slow extraction



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

readout system

1 PC → readout of 2 VME crates (via CAEN optical interfaces)
1 PC → storage

6 × 32 ch QDCs + TDCs → CAEN V792, V862, V775

1 × 34 ch (5 Gs/s) digitizer → CAEN V1742
(single event: ∼34×1024×12bit)

1 × 4 ch (20 Gs/s) oscilloscope → 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]

DREAM DAQ

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

in-spill (slow extraction)

poll trigger signal … if trigger present:

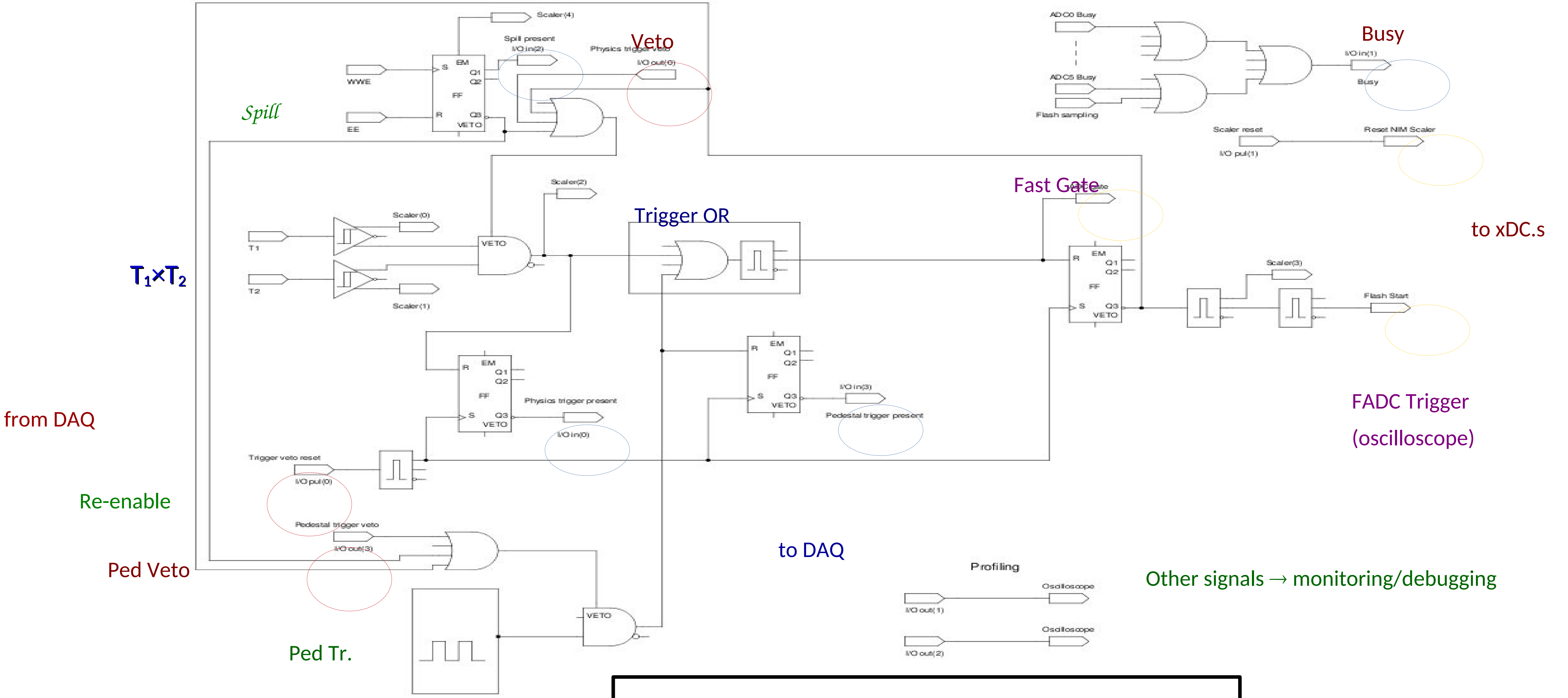
a) (block) read all VME boards
b) format & store on large buffers (FIFO over RAM)
c) re-enable trigger

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



Trigger = $V \times T_1 \times T_2$ | ped

trigger system

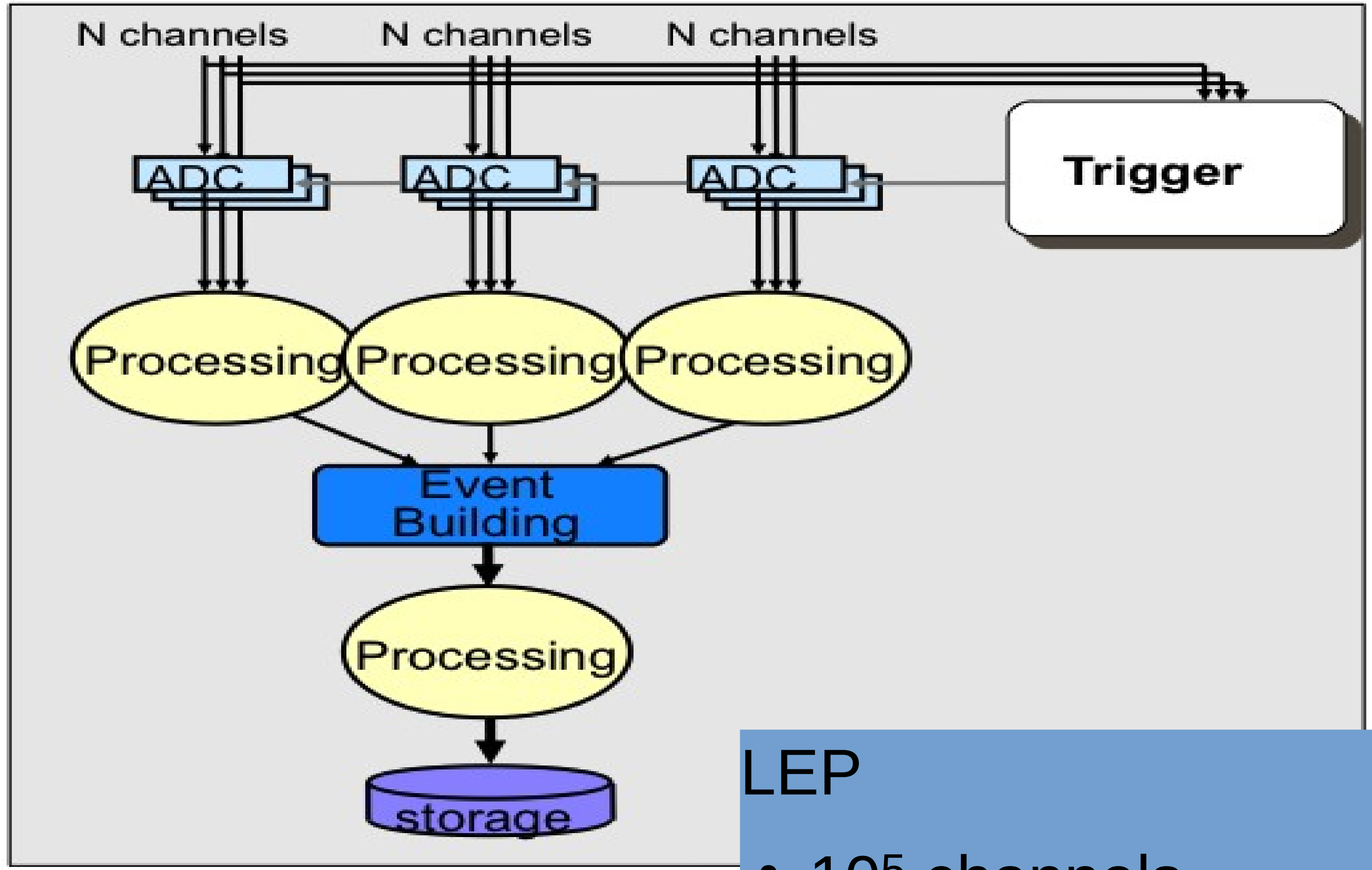
a) crystals w/ fast PMT.s
b) no analog buffering

$\rightarrow$ low-latency trigger

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

step 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

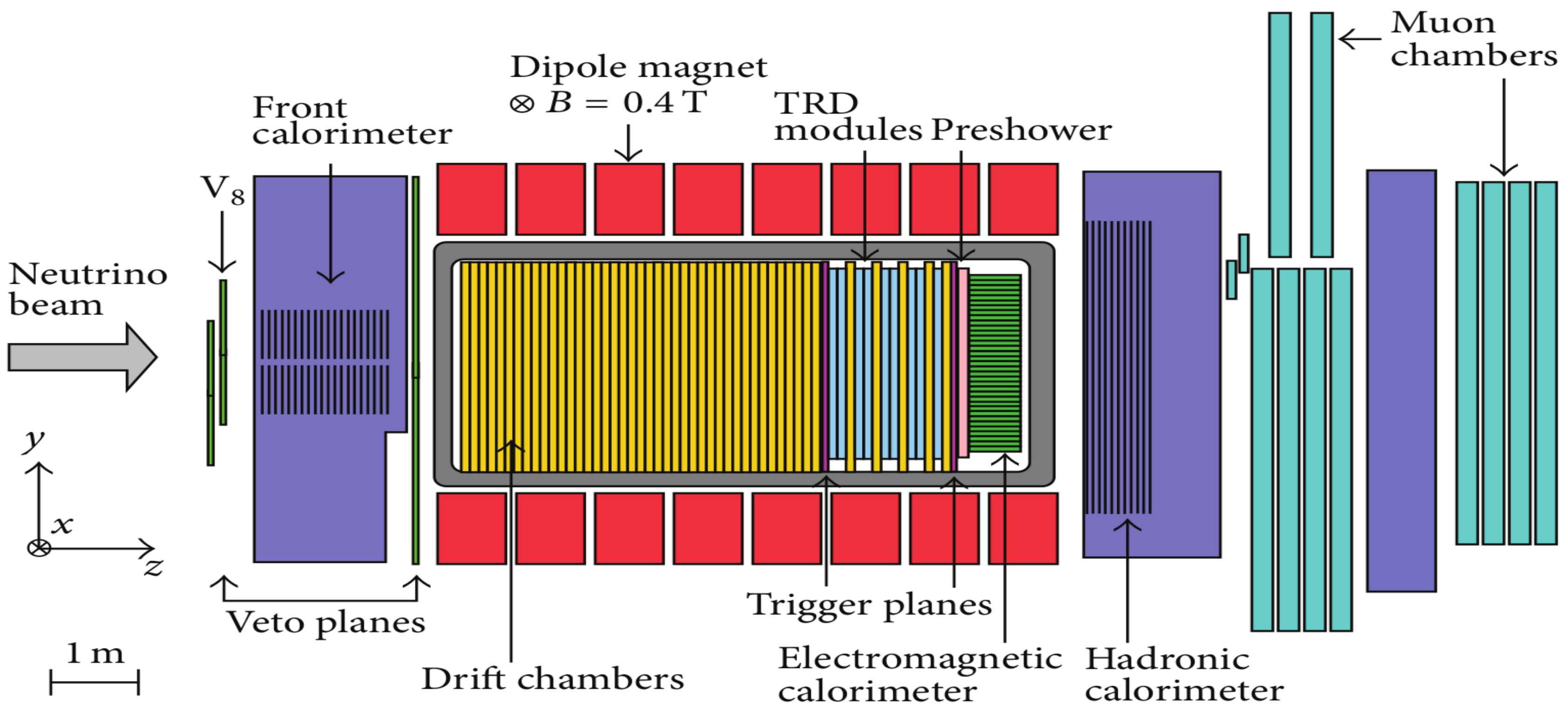


LEP

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

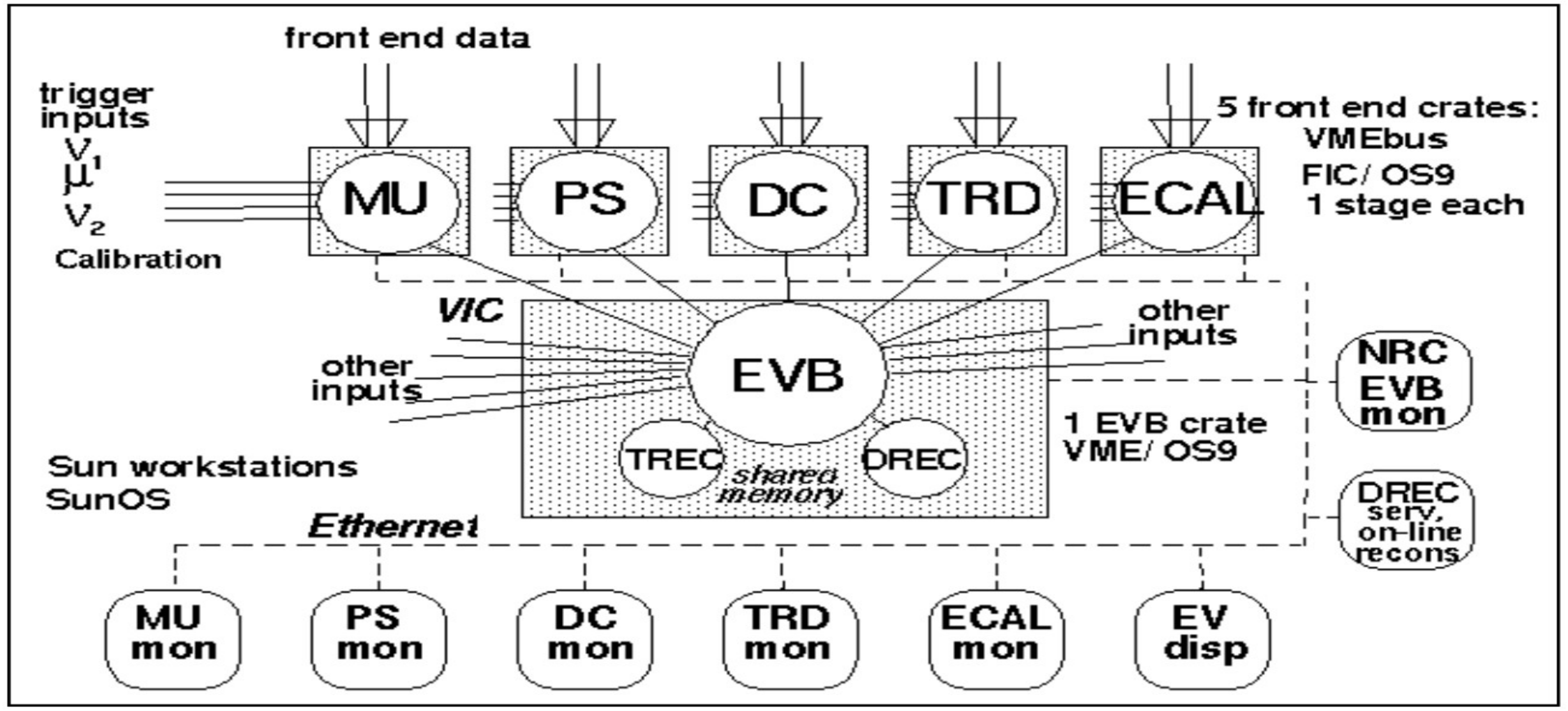
NOMAD (1995-1998)

- Search for $\nu_\mu \rightarrow \nu_\tau$ 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



veto counters

trigger counters



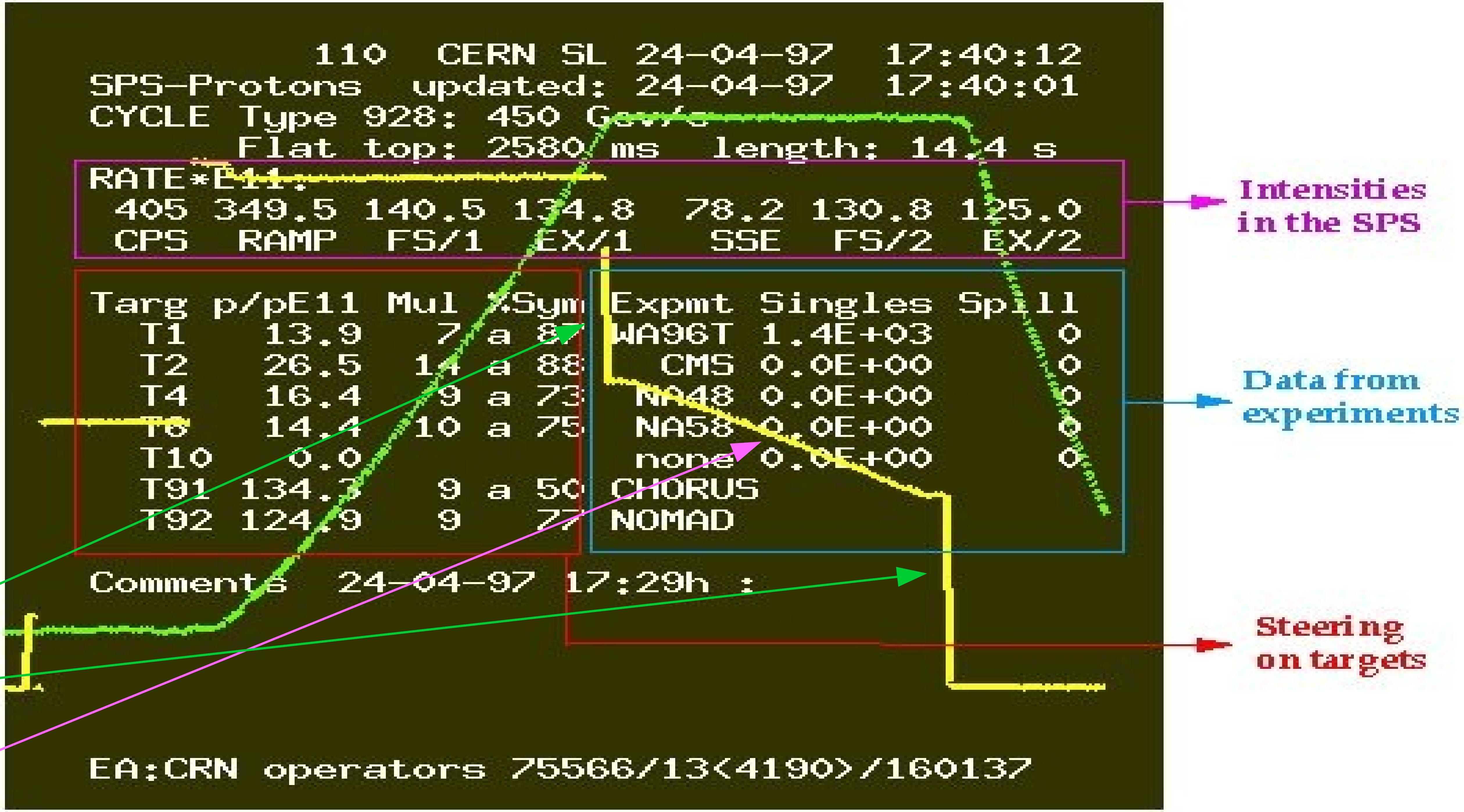
→ DAQ layout

WANF - SPS SuperCycle

14.4 s cycle length

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

1 × 2 s muon spill (slow extraction)



f/s extractions

slow extraction

triggering once more ...

menu for NOMADs:

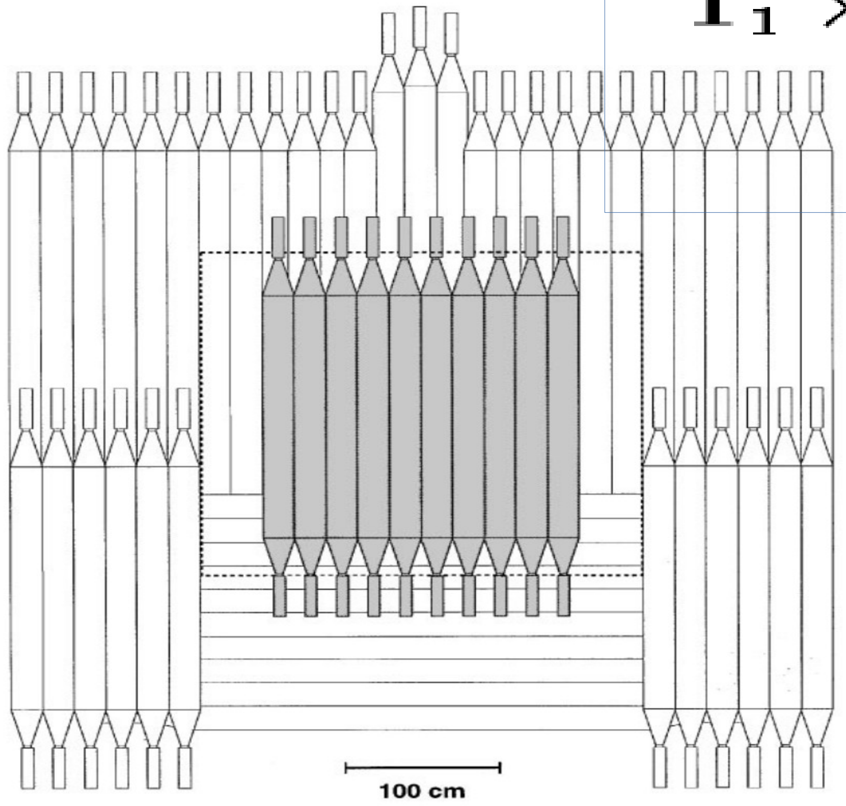
ν-spill triggers

$$\overline{V} \times T_1 \times T_2$$
$$\overline{V_8} \times FCAL$$
$$\overline{V_8} \times FCAL' \times T_1 \times T_2$$
$$\overline{T_1 \times T_2} \times ECAL, \overline{V_8} \times ECAL$$
$$RANDOM$$

μ-spill triggers

$$V \times T_1 \times T_2$$
$$V_8 \times T_2$$
$$V_8 \times T_1$$
$$V_8 \times T_1 \times T_2 \times FCAL'$$
$$V \times T_1 \times T_2 \times ECAL$$

~3m



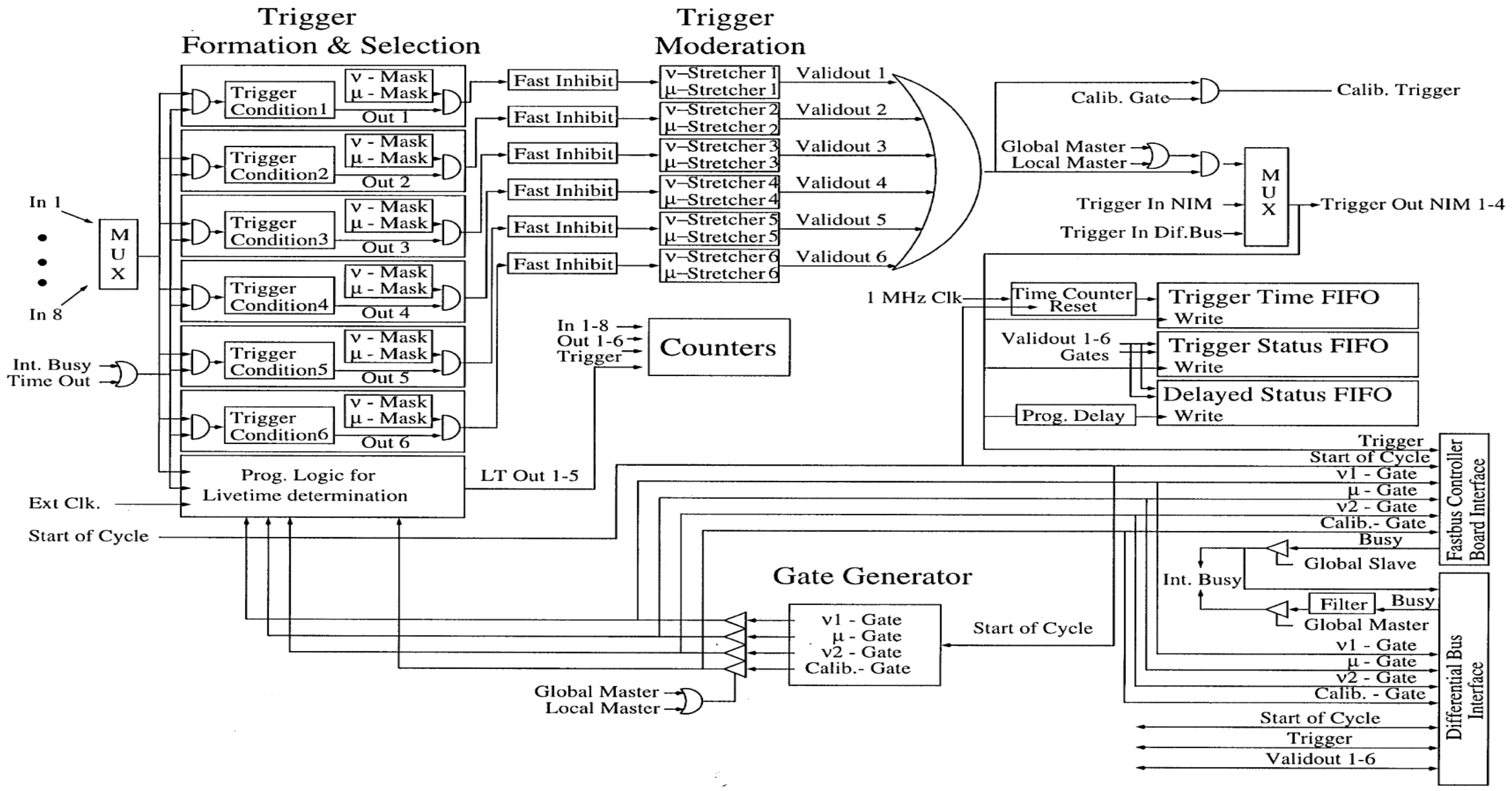
~3m

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)

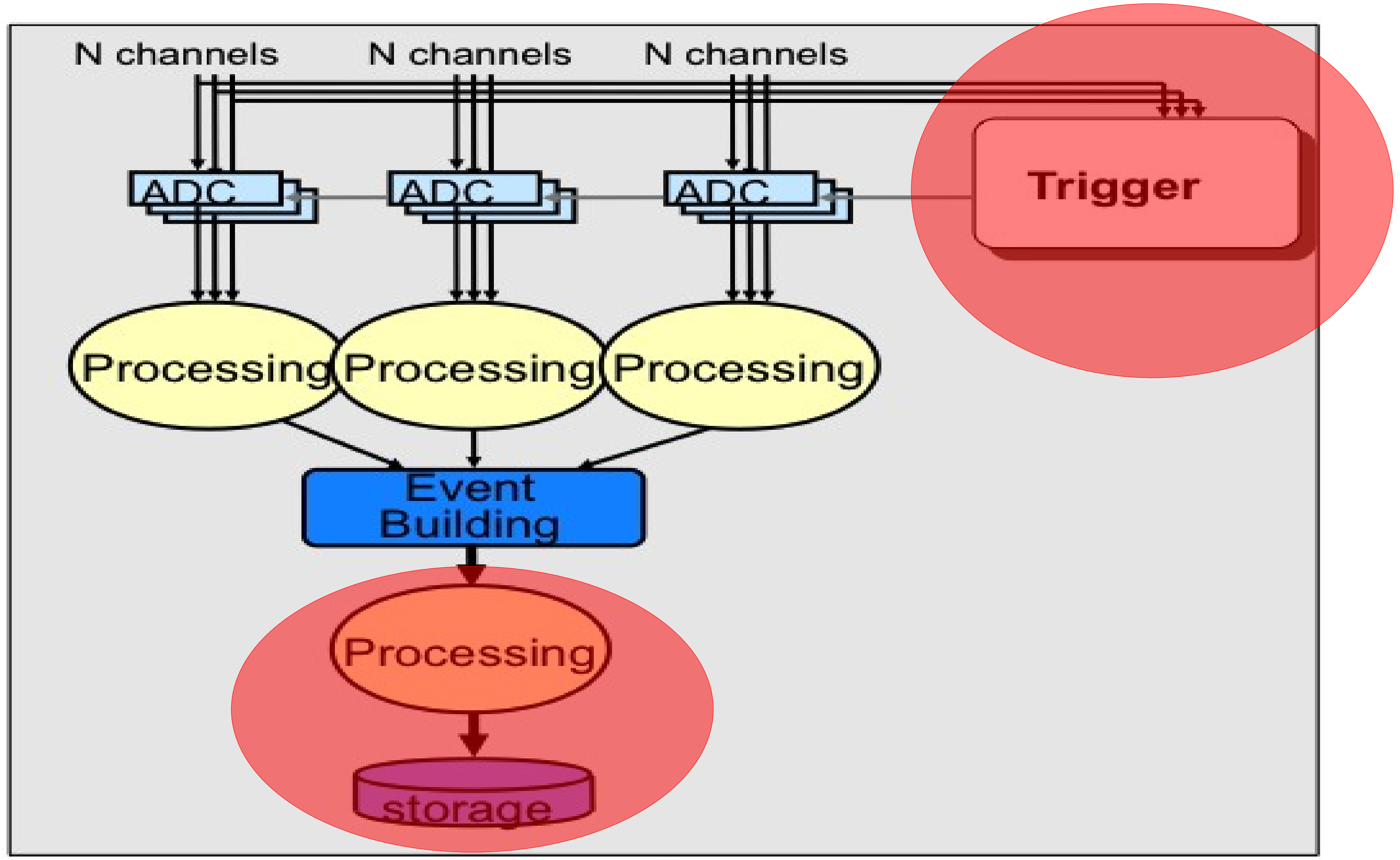
readout sequence

- 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

$\rightarrow$ deadtime in ν spills: $\sim$10% due to digitisation

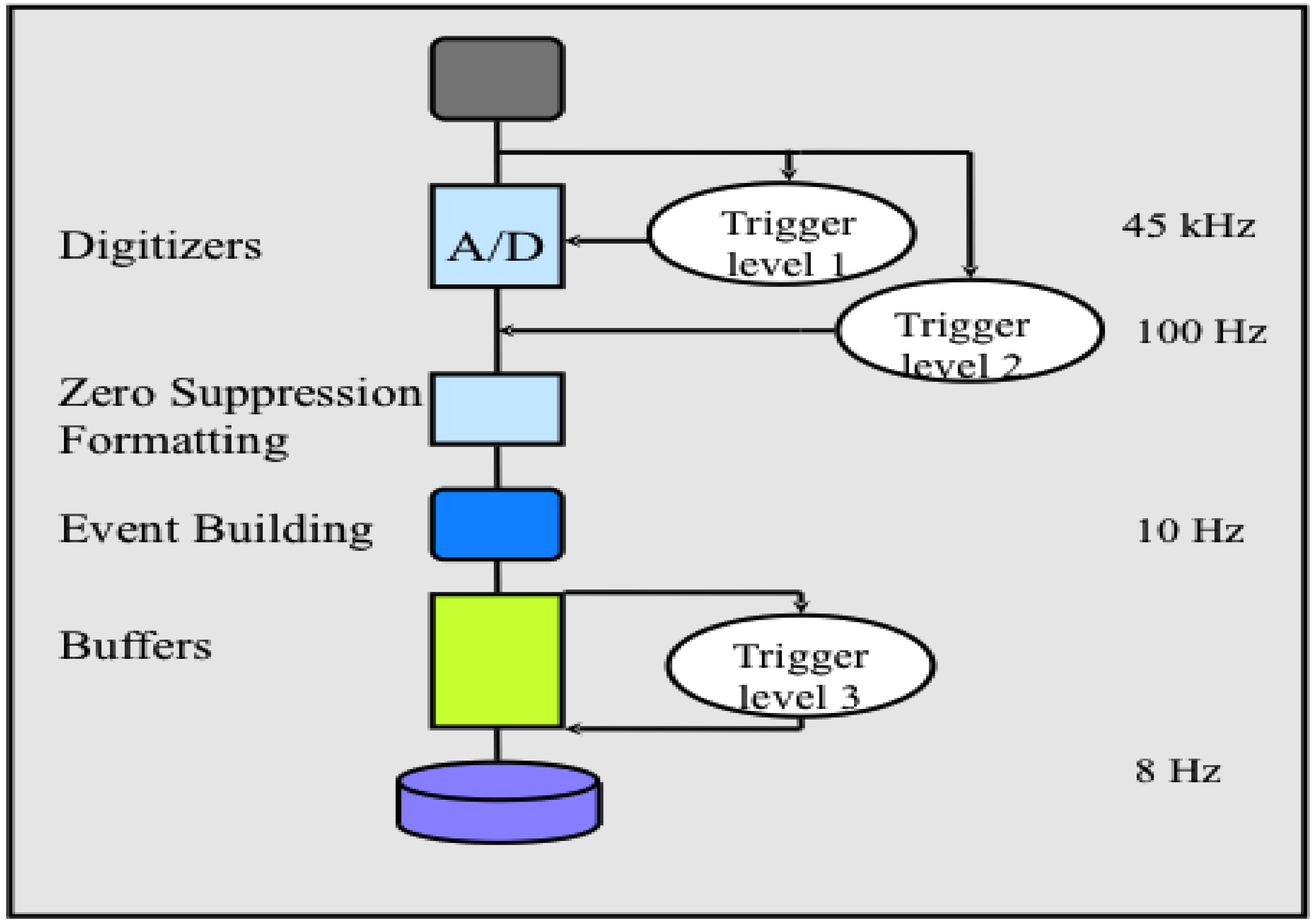
more bottlenecks ?



- 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



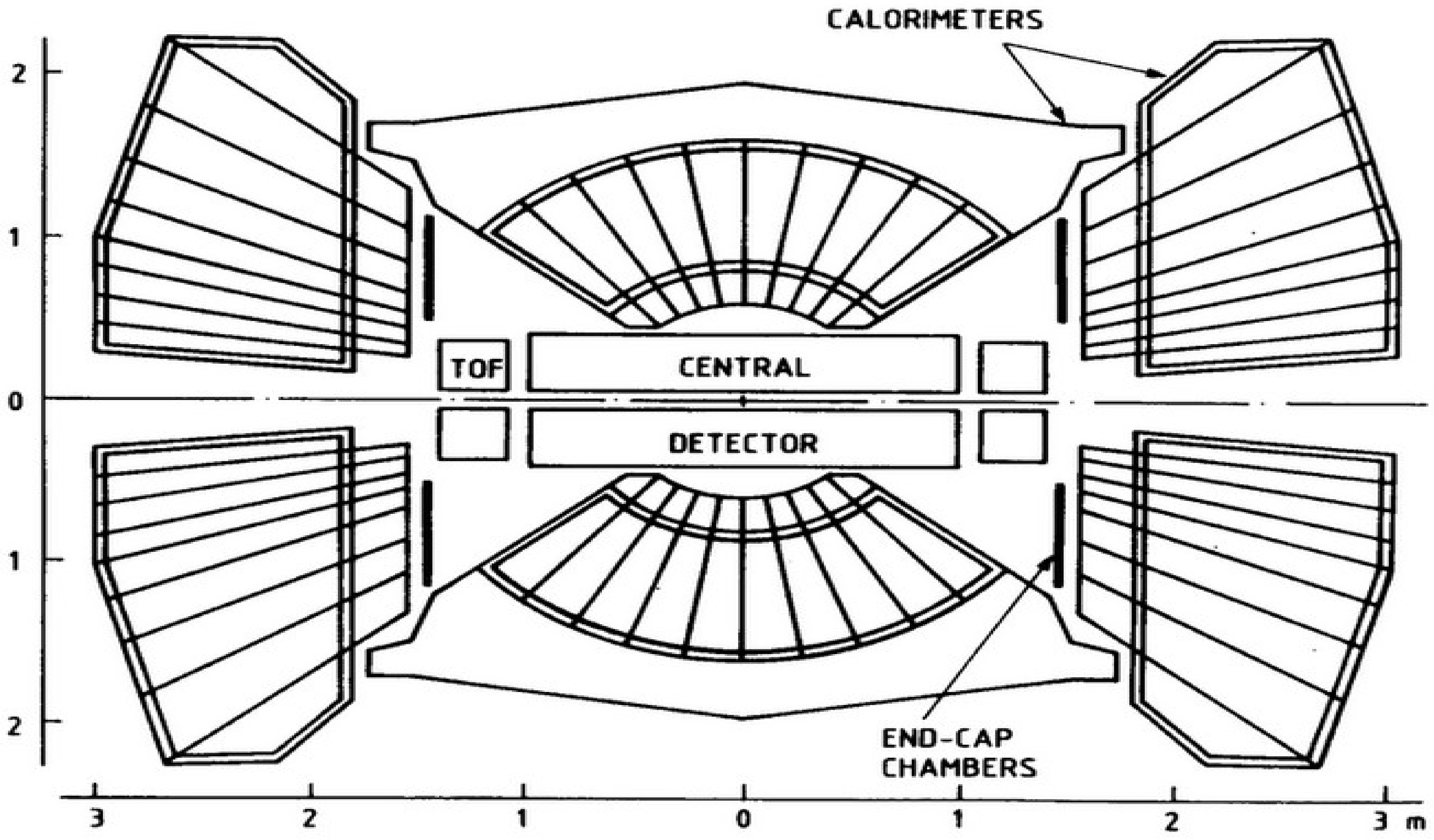
- more complex filters
  - → slower
  - → applied later in the chain

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 $p\bar{p}$ collisions @ CERN $p\bar{p}$ collider:

$\sqrt{s}$ = 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 discovery !!

Upgraded UA2 experiment (1987-1991)

High-lumi $p\bar{p}$ collisions @ CERN $p\bar{p}$ collider:

$\sqrt{s}$ = 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 discovery !!

→ robust theoretical prediction for new physics

Upgraded UA2 experiment (1987-1991)

High-lumi $p\bar{p}$ collisions @ CERN $p\bar{p}$ collider:

$\sqrt{s}$ = 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 discovery !!

→ robust theoretical prediction for new physics

unfortunately … nature was wrong!

Upgraded UA2 experiment (1987-1991)

High-lumi $p\bar{p}$ collisions @ CERN $p\bar{p}$ collider:

$\sqrt{s}$ = 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 discovery !!

Complex trigger signatures:

em, jet and missing E_T

Missing E_T ? How can I trigger on it ?

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:

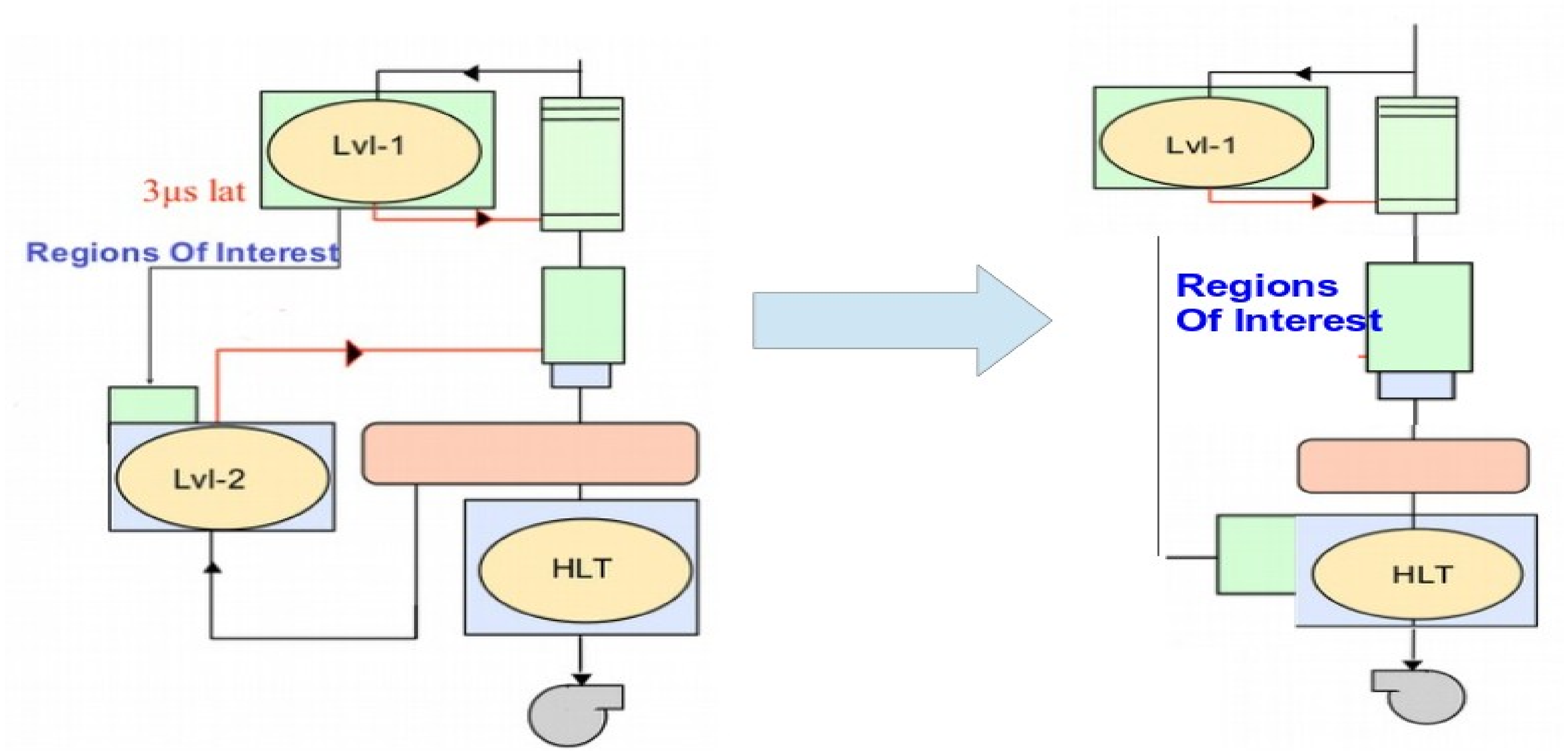
- L1 from on-detector hardware
- L2 over dedicated processors
- L3 over FASTBUS processors (ALEPH event builder)

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 single HLT farm

- preserve Region of Interest but dilute farm separation and fragmentation
- increase flexibly, computing power efficiency

trigger/event-selection latencies

Possible (e.g. ATLAS Run 1) values:

- L1 : O(1 μs in real-time) $\rightarrow$ let say = 1.9 μs
- L2 : O(10 ms) $\rightarrow$ let say = 40 ms
- L3(HLT) : O(s) $\rightarrow$ 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

→ *Latency = Max Latency*

→ over fluctuations bad, will create deadtime

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

→ *Latency = Mean Latency*

→ over fluctuations fine, shouldn't create deadtime

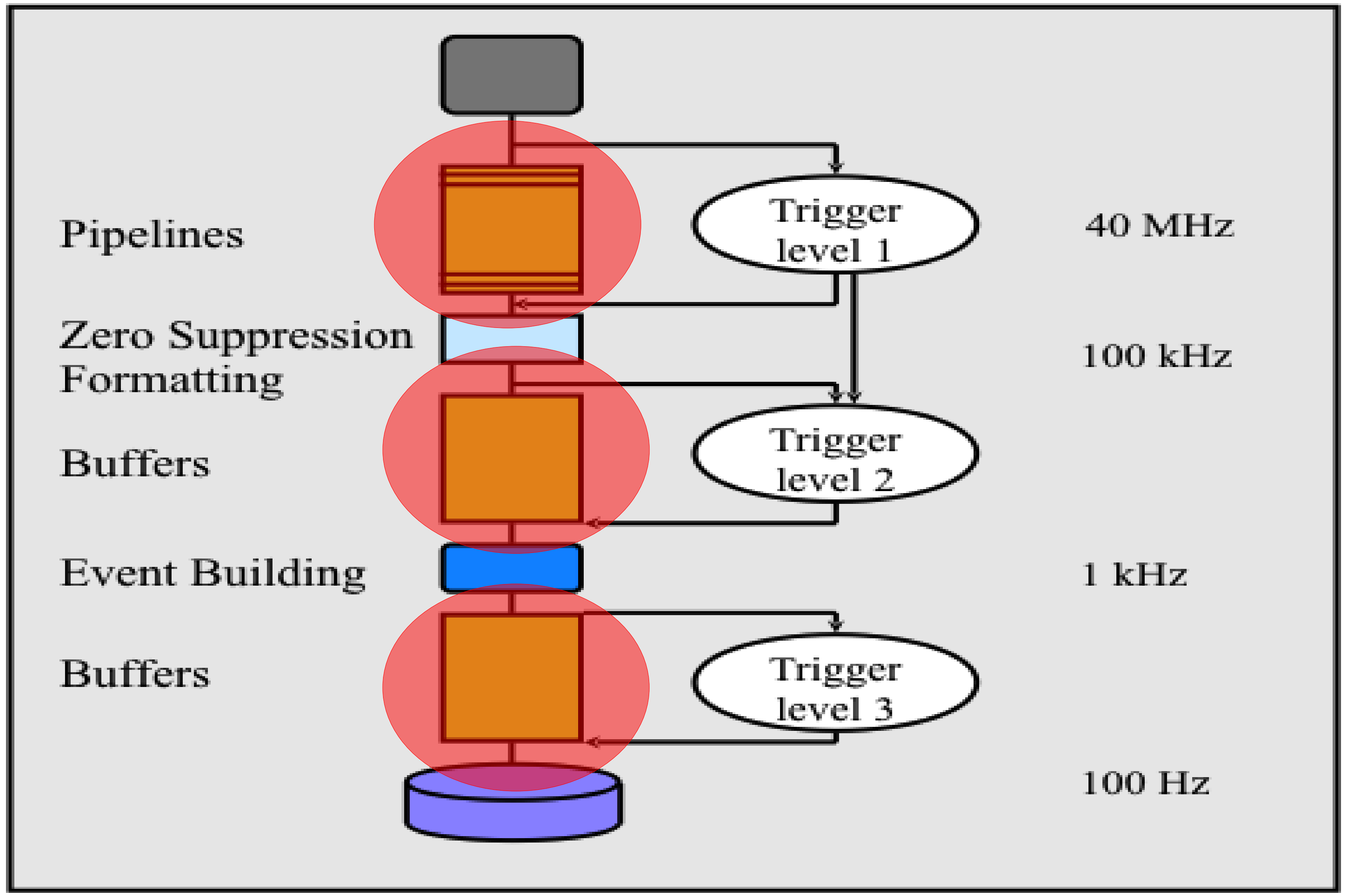
real time o.s. :

very stable time delay in responding to events

standard unix kernels are not real time:

a system call can in principle take any time

step five: dataflow control

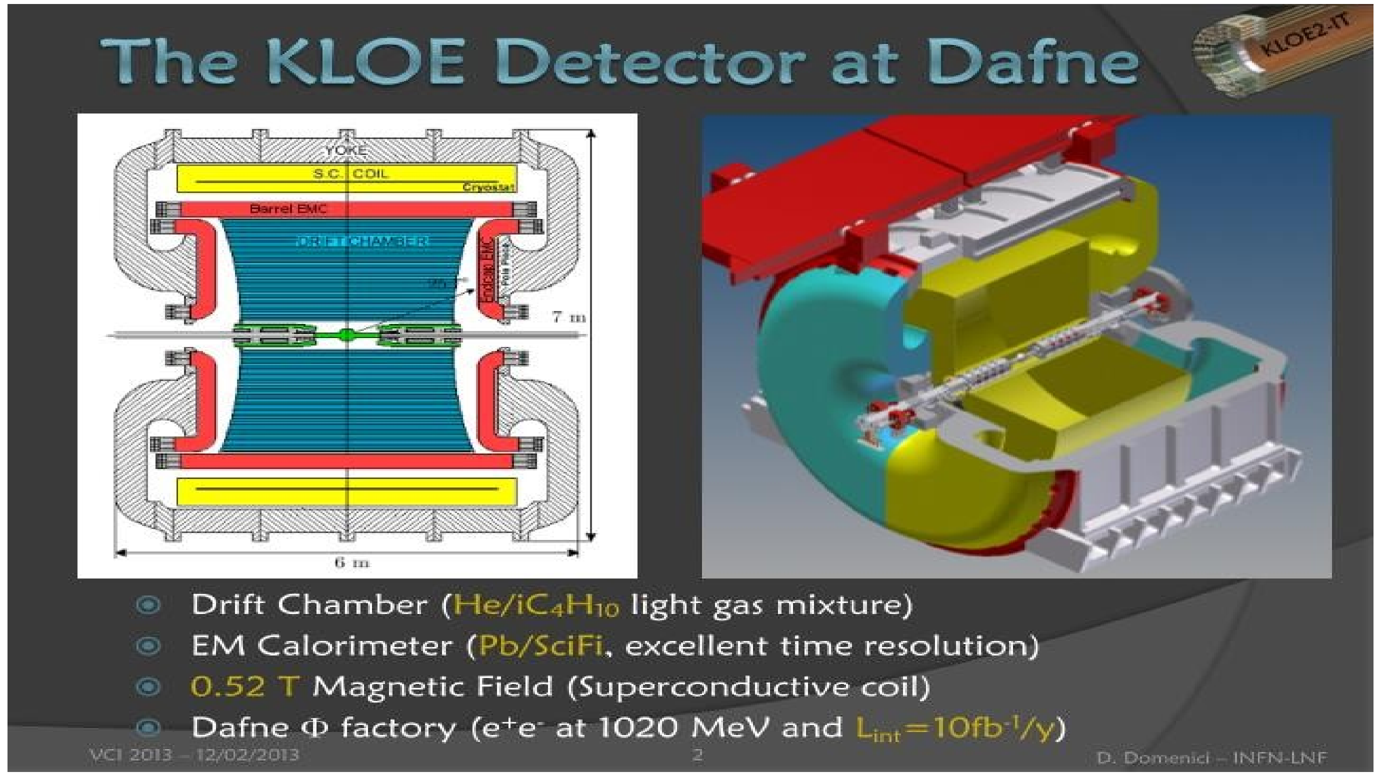


- Buffers:
  - not the “final solution”
  - can overflow due to:
    - bursts
    - unusual event sizes
- Discard
  - either locally
  - or exert “backpressure”
    - ask previous level(s) to block dataflow

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

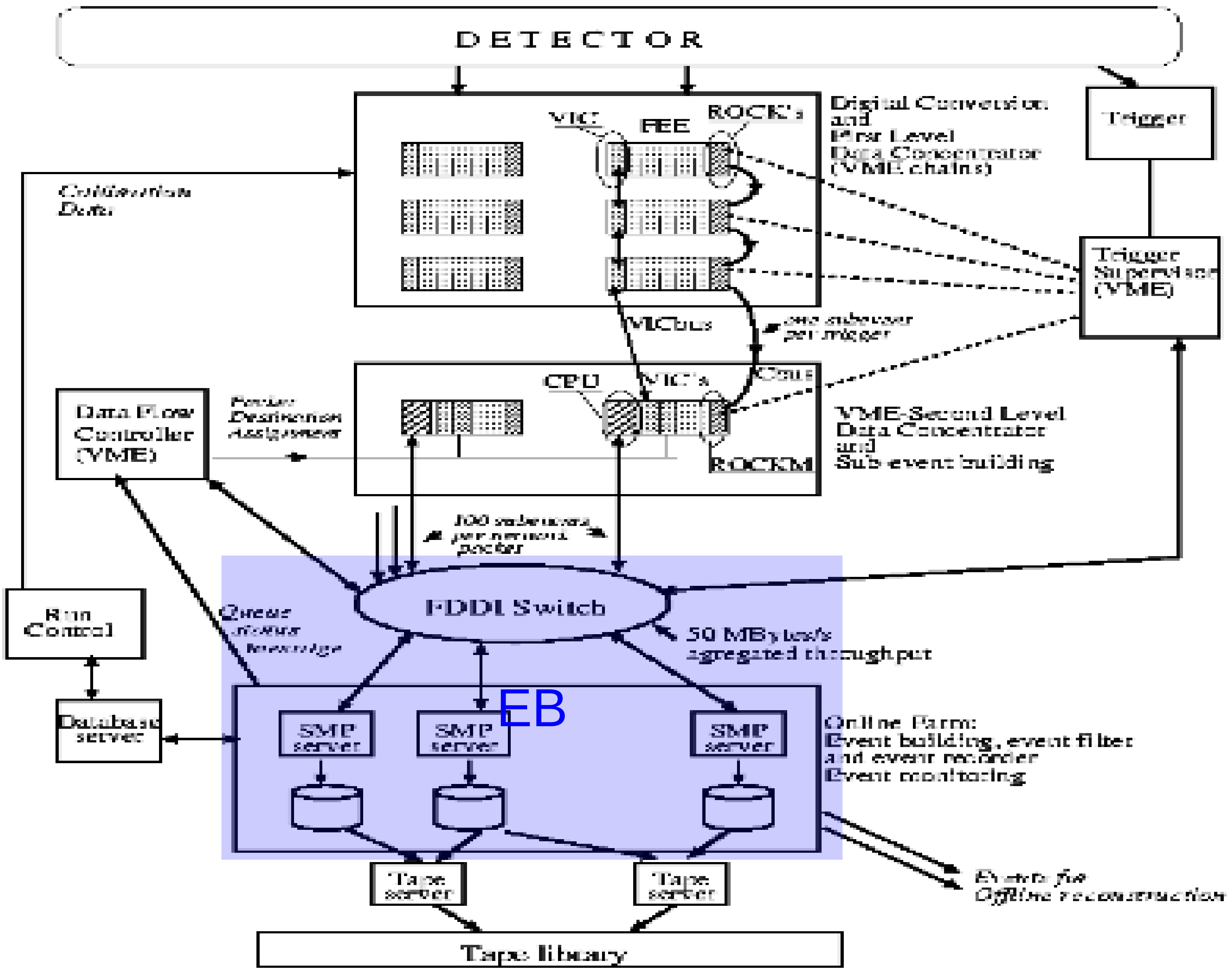
a *push* example: KLOE

- 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
  - 2 μs fixed deadtime
- HLT ~10^4 Hz
  - ~COTS, cosmic rejection only
- 5 kB/ev → 50 MB/s [design]

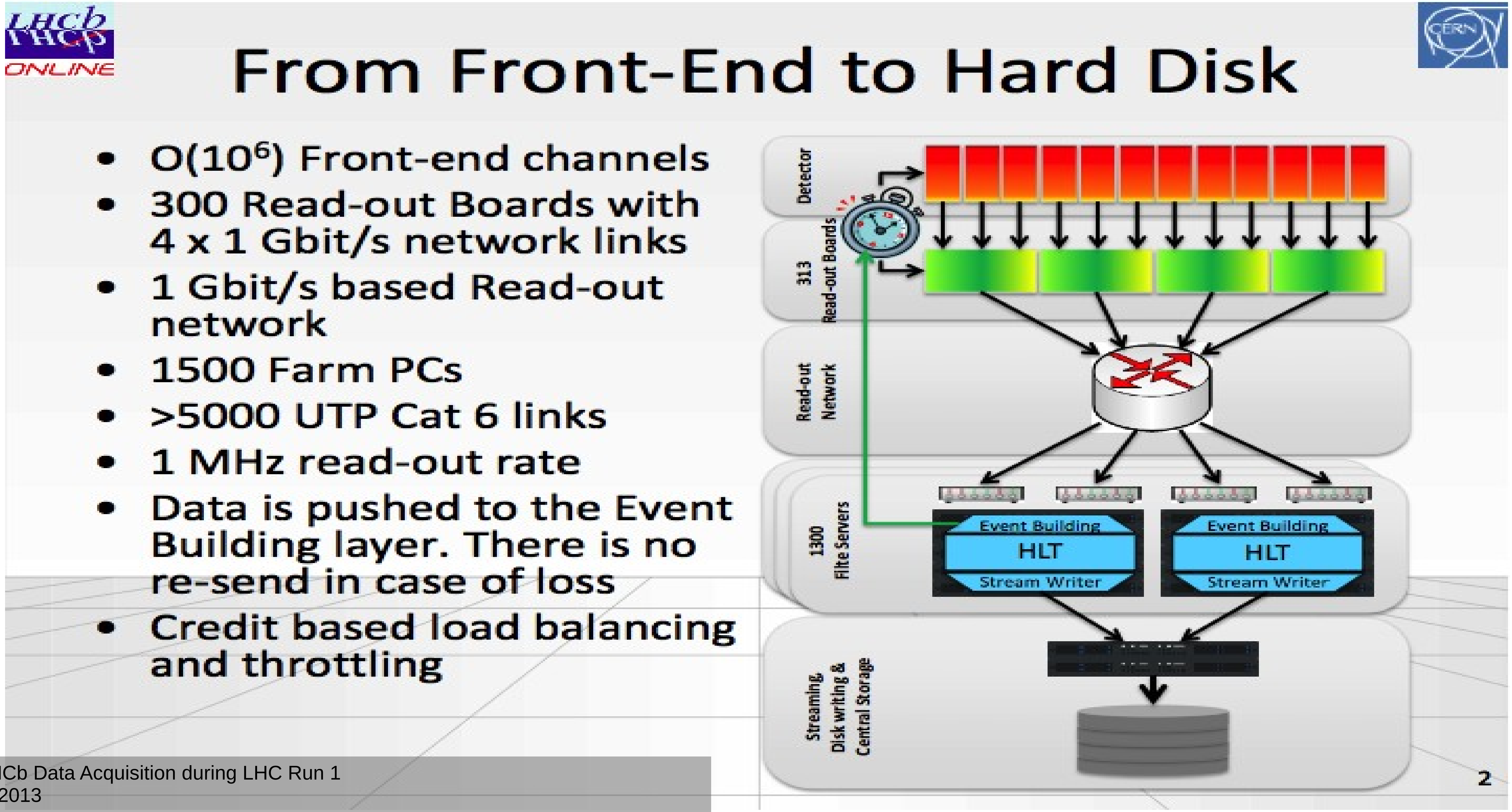
KLOE



- 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



The LHCb Data Acquisition during LHC Run 1
CHEP 2013

more info in "TDAQ for the LHC experiments"

ATLAS TDAQ in Run 2

~ 2 MB events, ~ 50 GB/s network bandwidth, ~ 1.5 GB/s recording throughput

Custom point-to-point links

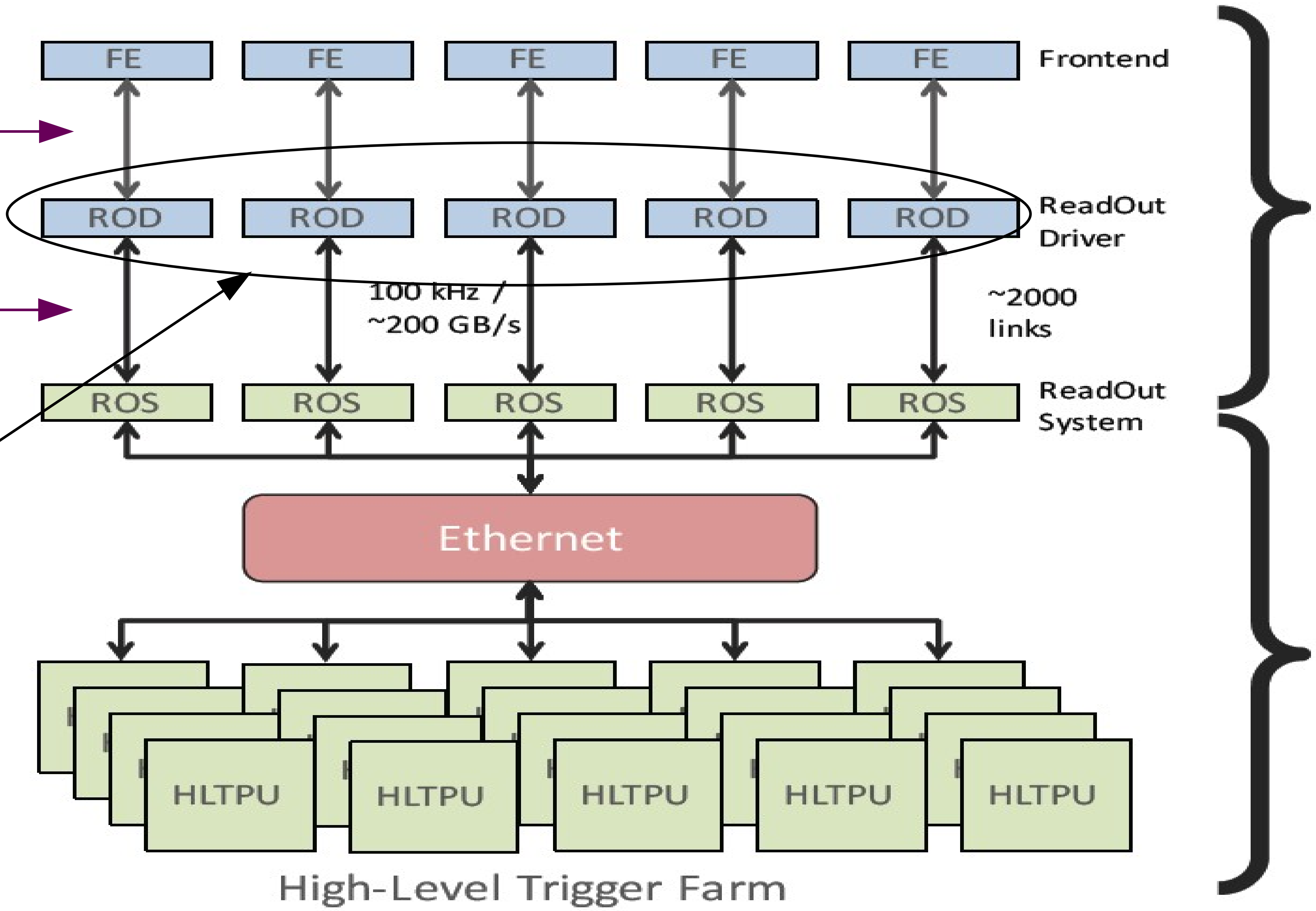
~1000 boards

Point-to-point S-LINKs*

~150 servers

~1500 servers

VME
(detector-specific)



Custom elx components

PCs (COTS)

*S-LINK: CERN Simple Link

Upgrade for Run 3

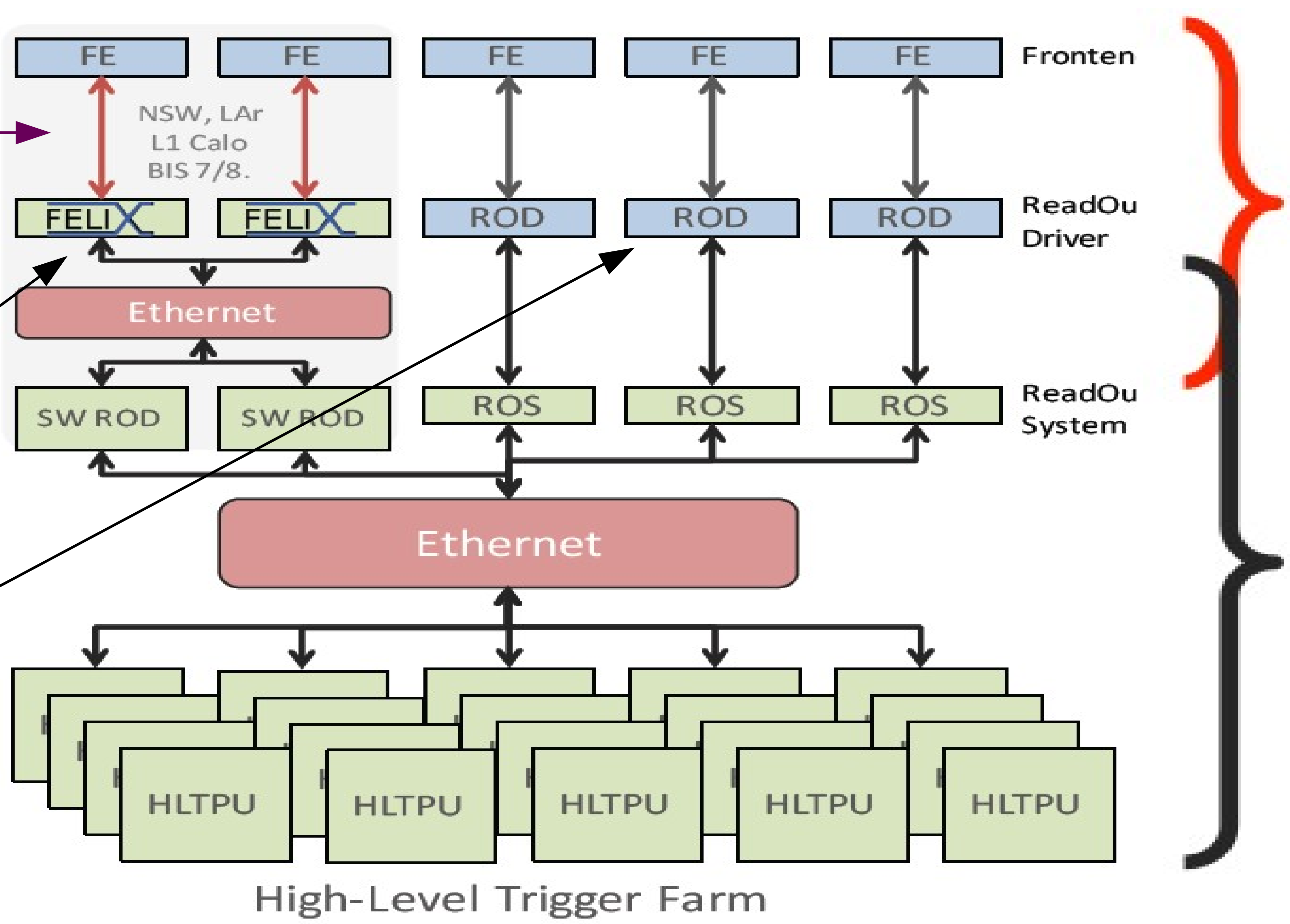
Same requirements as Run 2 but reduced custom components

GBT*/FULL mode links

25-100 Gb/s ethernet

PCIe Gen3
(DAQ-specific)

VME
(detector-specific)



Custom elx components

PCs (COTS)

*GBT: GigaBit Transceiver with Versatile Link

ATLAS dataflow

Push mode from front-end elx up to ROS/swROD system

→ data sent as soon as available

Pull mode from ROS to HLT

→ data requested by HLT as soon as HLT is free

⇒ ROS/swROD must handle all critical dataflow issues

LHC Run 3

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

2013 DAQ@LHC Workshop

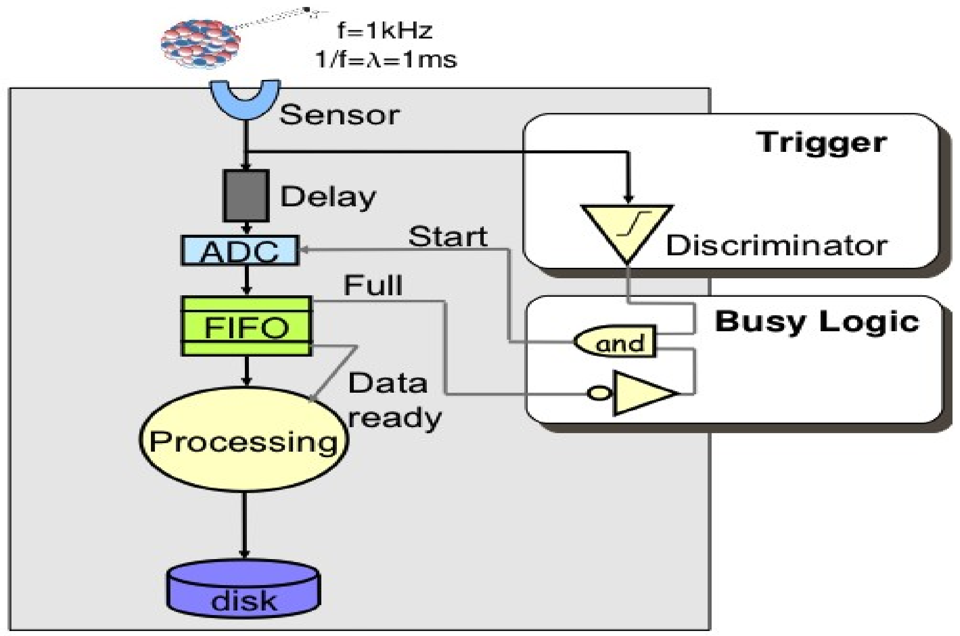
ALICE

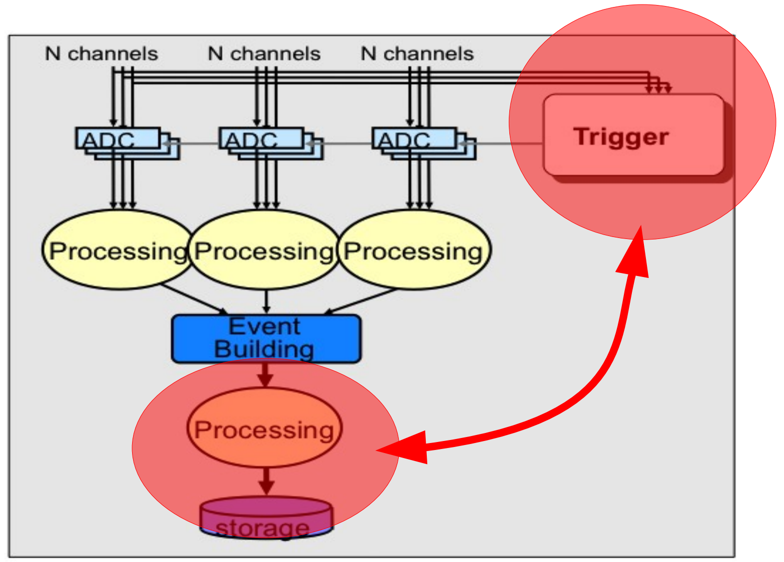
- First level trigger for Pb-Pb interactions 500 Hz → 50 kHz
- 22 MB/event
  - 1 TB/s readout → 500 PB/month
- Data volume reduction
  - on-line full reconstruction
  - discard raw-data
- Combined DAQ/HLT/offline farm
  - COTS, FPGA and GPGPU

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

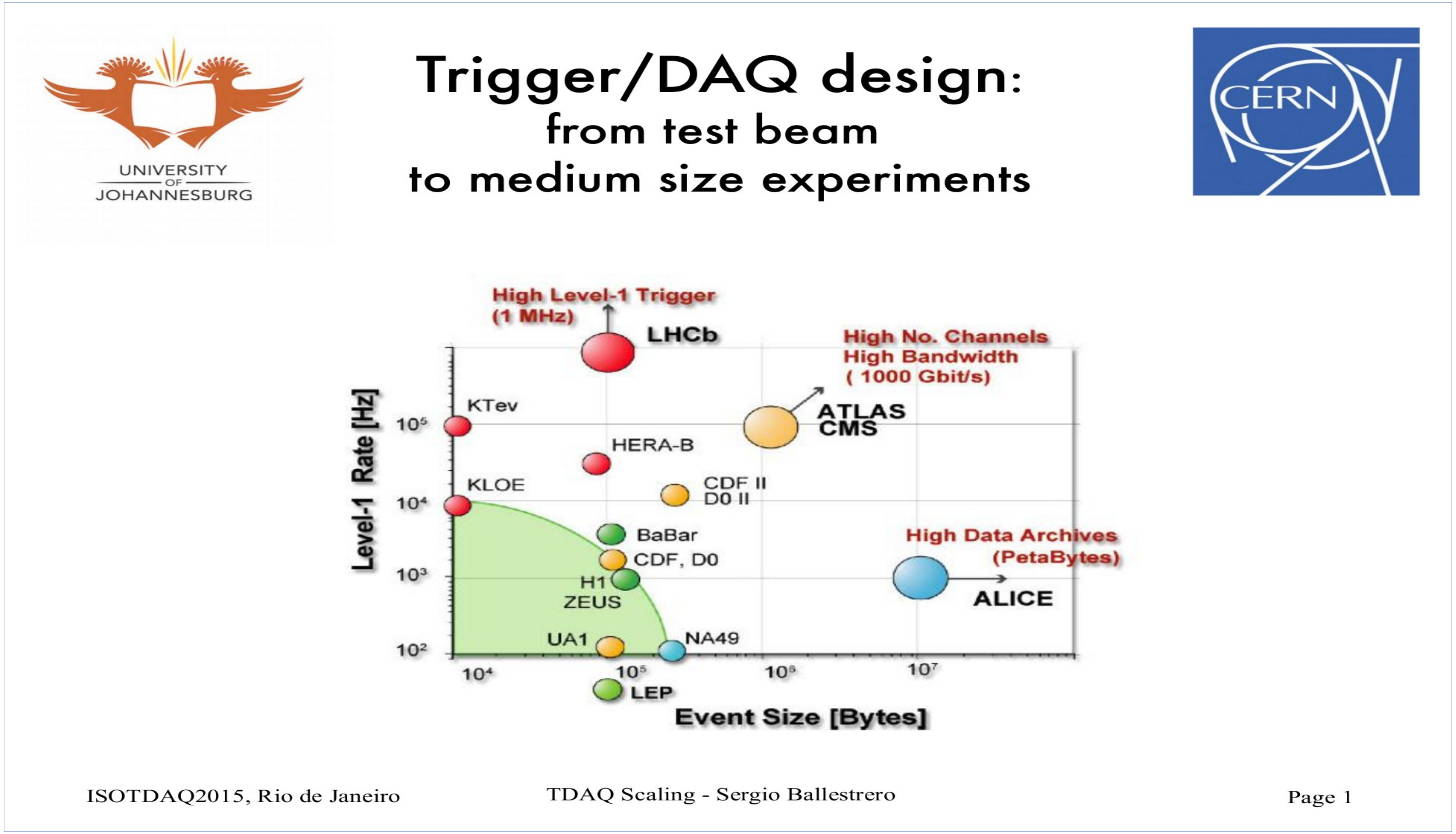
Hw control & recovery

Sw control & recovery

Monitoring

…

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



Lost & Found
(off-topics)

Appendix A: 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]
/lib64/libc.so.6(+0x347e0)[0x7fa55f1c07e0]
./my_segf(_Z3bazv+0x14)[0x400aab]
./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]
```

handler

libc

my crash

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 A: Profiling

Take care: optimise your code - first of all - where it really needs → to find where, you may use 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 ...

Appendix B: Some crude queueing theory

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

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

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

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

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

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

for $\rho \sim 1 \;\Rightarrow\; P_j \sim P_{j+1} \;\Rightarrow\; \sum P_k \sim (N+1) \cdot P_0 = 1 \;\Rightarrow\; P_0 \sim P_1 \sim .. \sim P_N \sim 1/(N+1)$

⇒ deadtime ~ 1/(N+1)

want ~ 1% ⇒ N ~ 100

Appendix B: Some crude queueing theory

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

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

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

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

rate [j+1 → j] = $\mu \cdot P_{j+1}$ (empty at rat…)

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

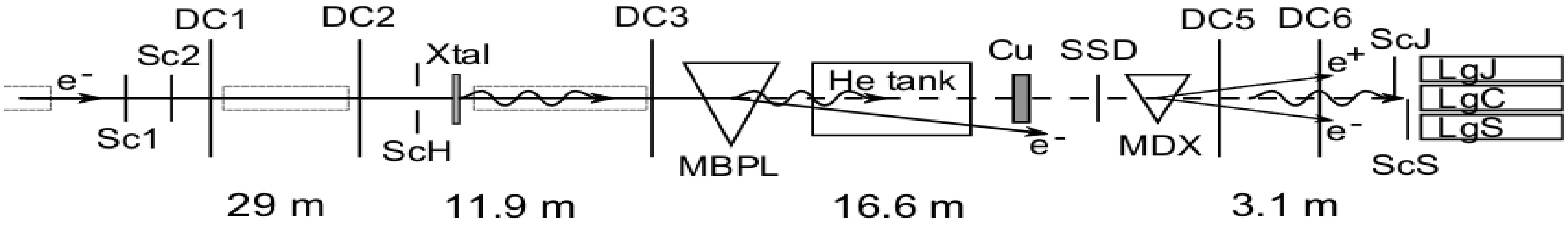
Take care: analytic calculation possible for pretty simple systems only

… 1 ⇒ $P_j \sim P_{j+1}$ ⇒ $\sum P_k \sim (N+1) \cdot P_0 = 1$ ⇒ $P_0 \sim P_1 \sim .. \sim P_N \sim 1/(N+1)$

⇒ deadtime ~ 1/(N+1)

want ~ 1% ⇒ N ~ 100

Appendix C: NA43/NA63

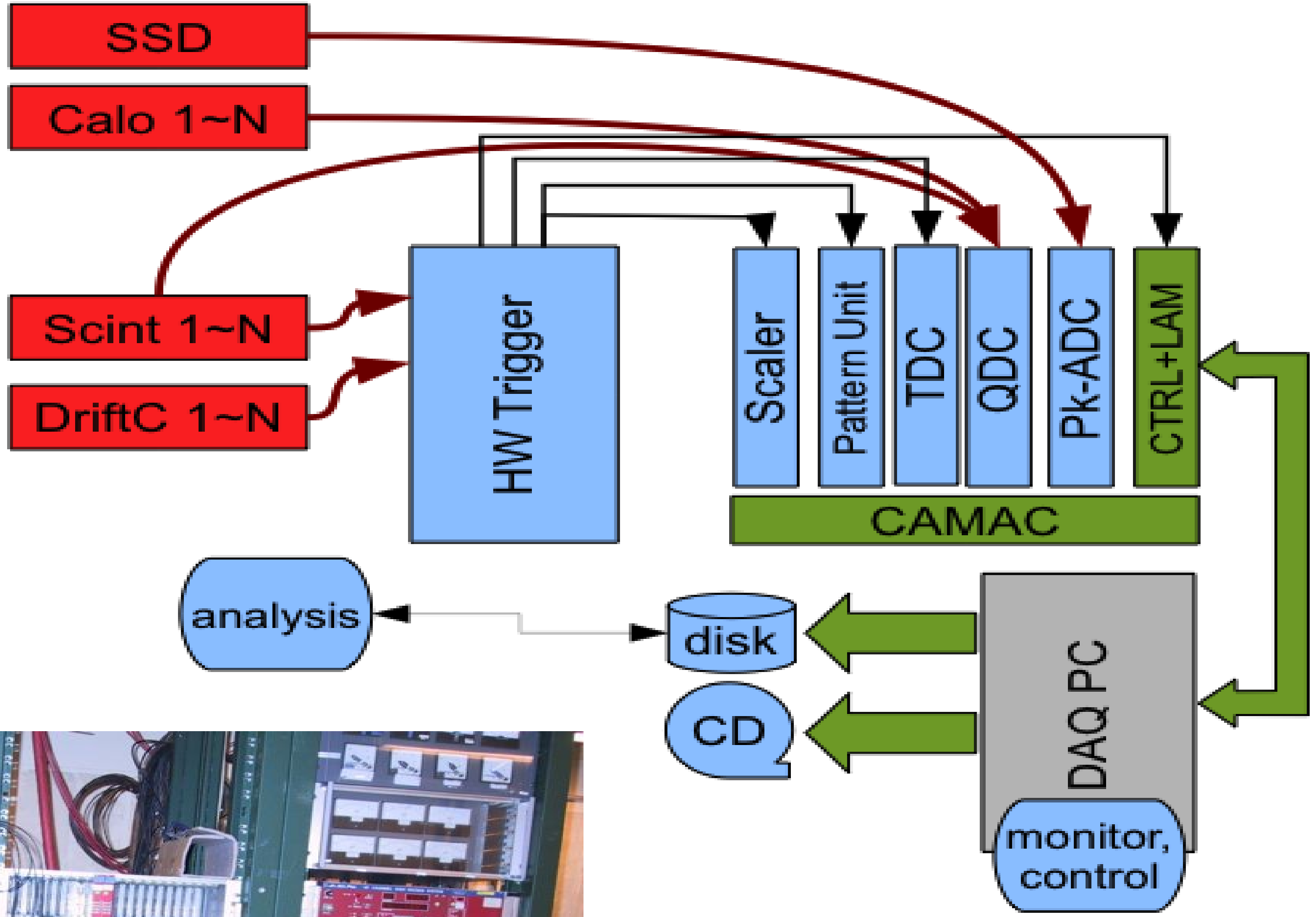


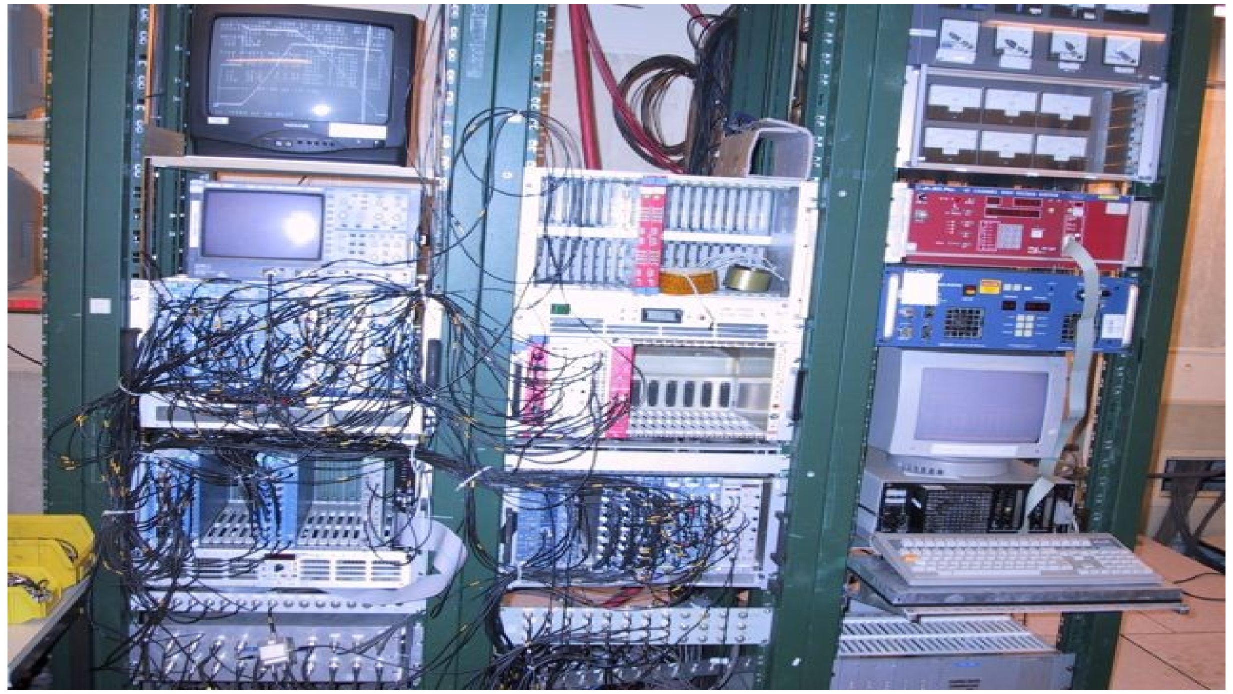
- Radiation processes: coherent emission in crystals and structured targets, LPM suppression…
- 80/120 GeV e- from CERN SPS slow extraction
- 2s spill every 13.5s
- Needs very high angular resolution
- Long baseline + high-res, low material detectors → drift Chambers
- 10 kHz limit on beam for radiation damage

→ 2-3 kHz physics trigger

NA43/NA63

- 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





Appendix D: Trigger qualification

trigger parameters:

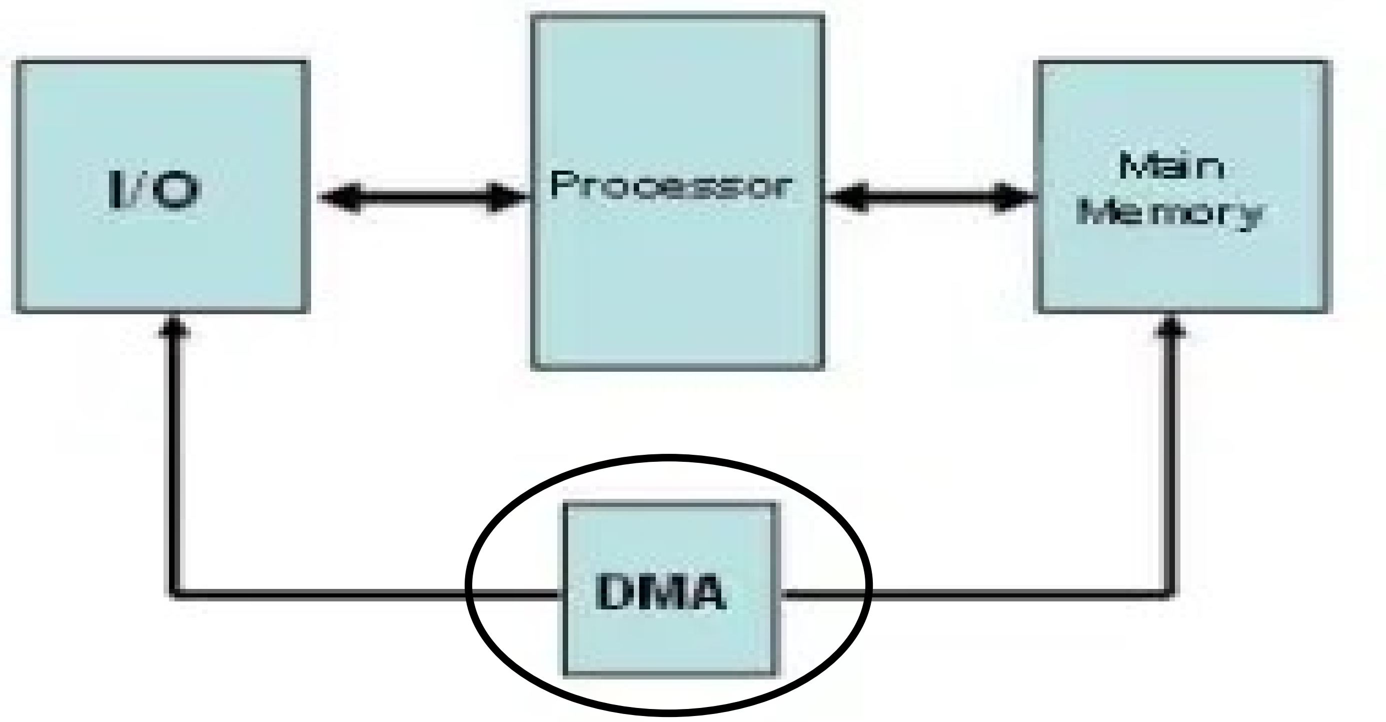
1) (high) efficiency → can't be improved at HLT
2) (high) purity → can be improved at HLT
3) (low) latency → can be compensated for
4) (very low) jitter → can't be compensated for
5) synch/asynch → synch "easier"

Appendix E: block transfer

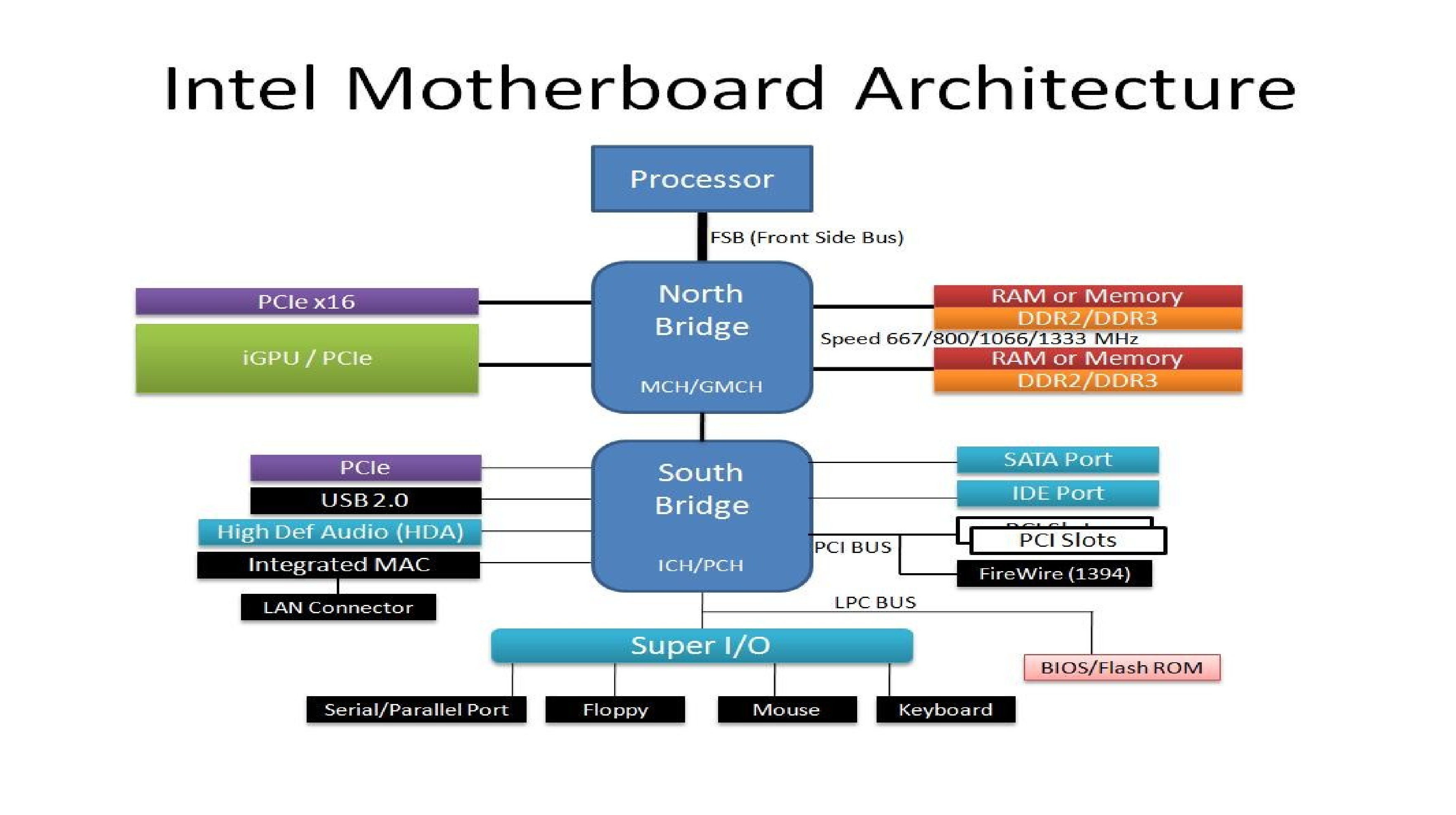
DMA (direct memory access):

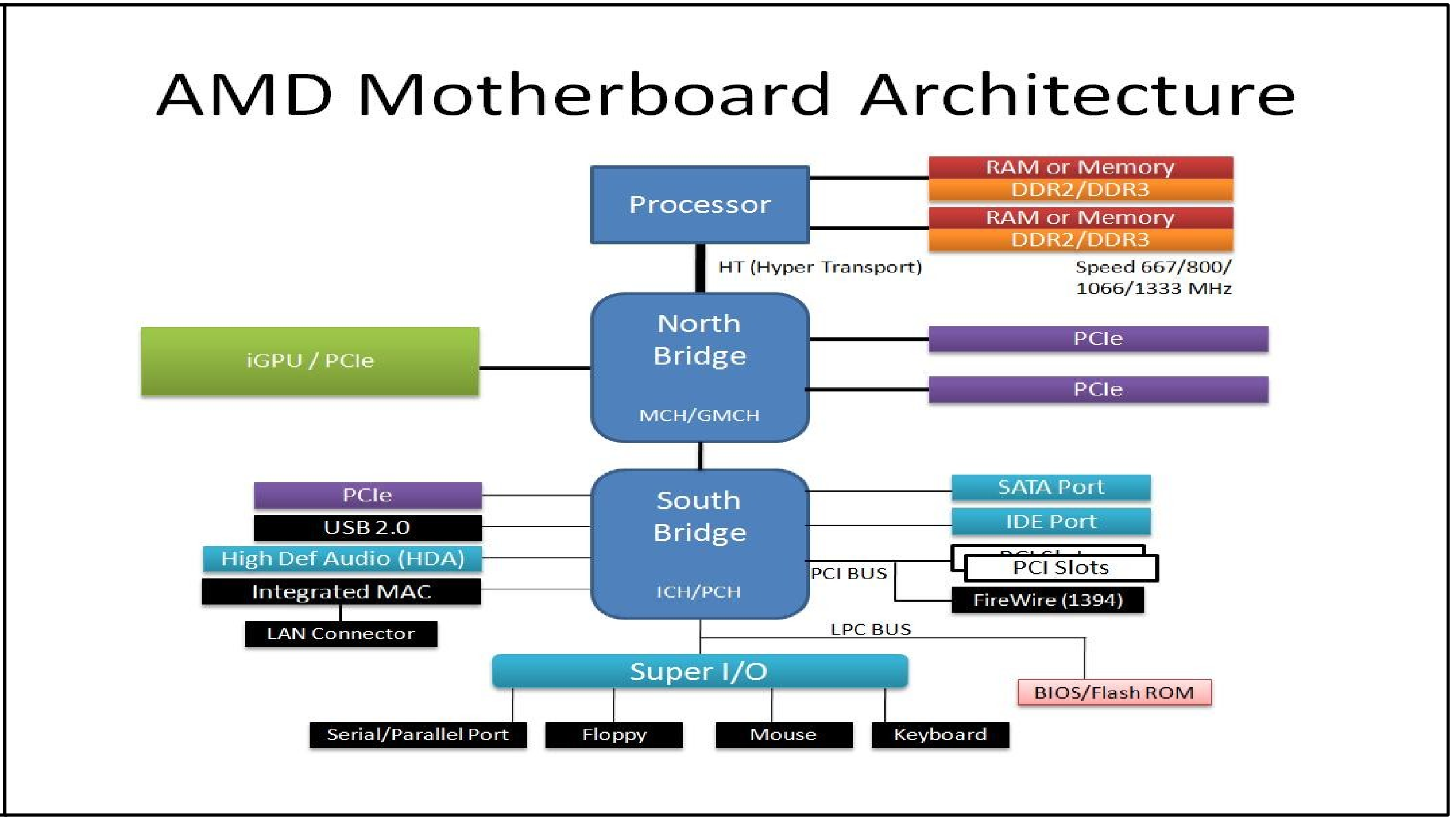
1) load source address (can be FIFO)
2) load destination address (can be FIFO)
3) load size (or until “data-available”)
4) run

needs specialised hardware



main actual implementations

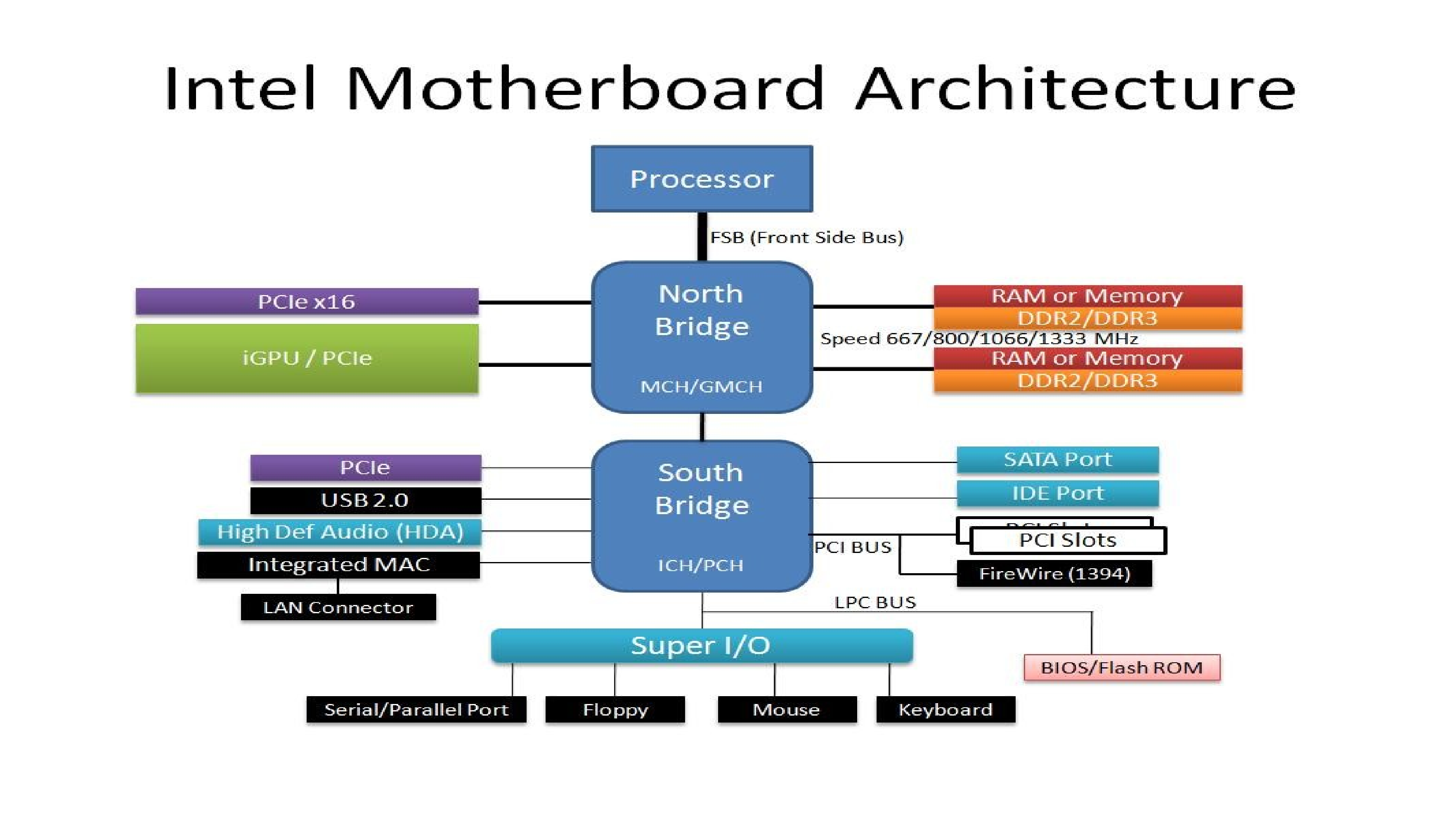


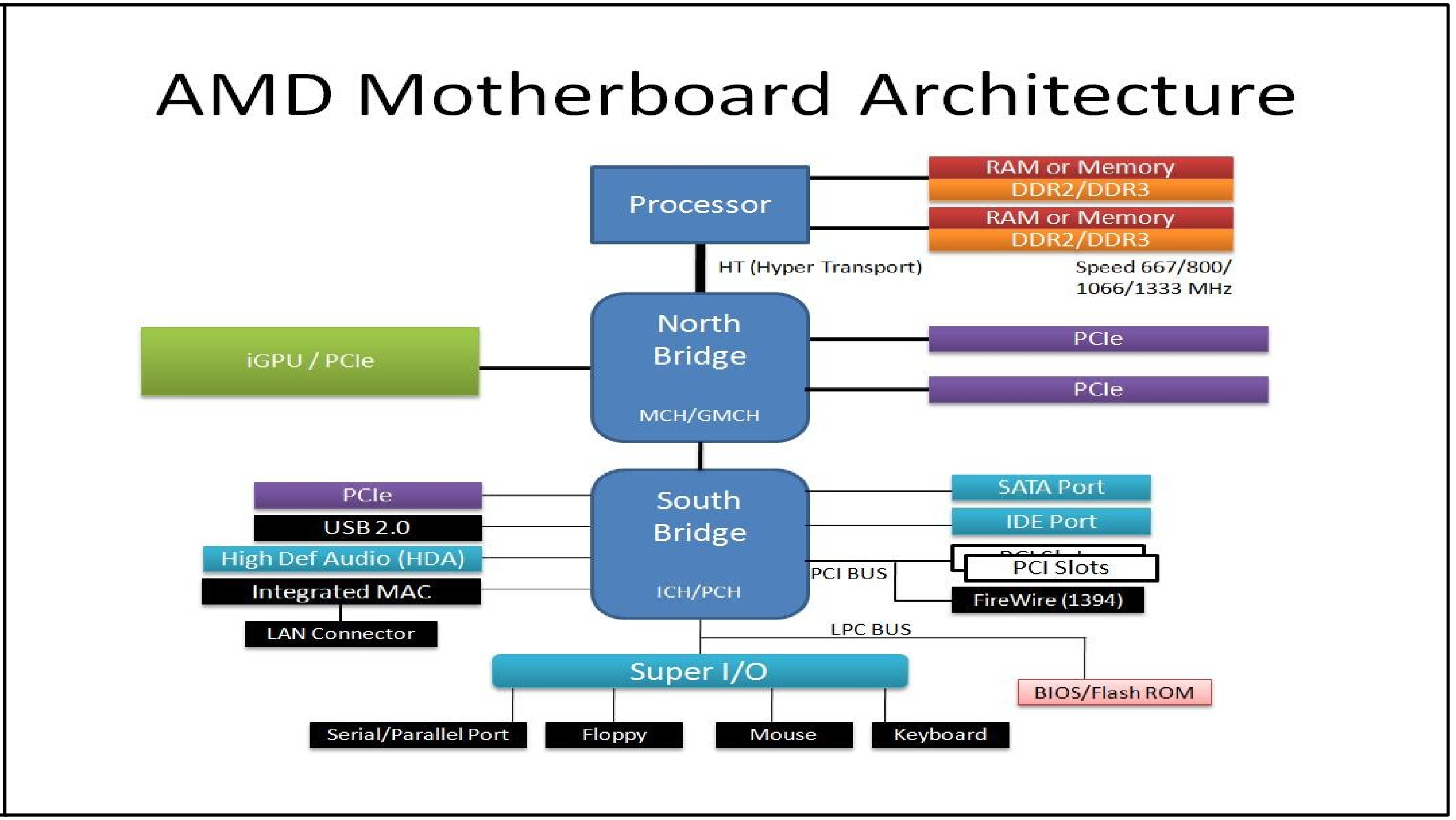


North Bridge: graphics and memory controller hub
South Bridge: I/O controller hub

computer architecture

main actual implementations

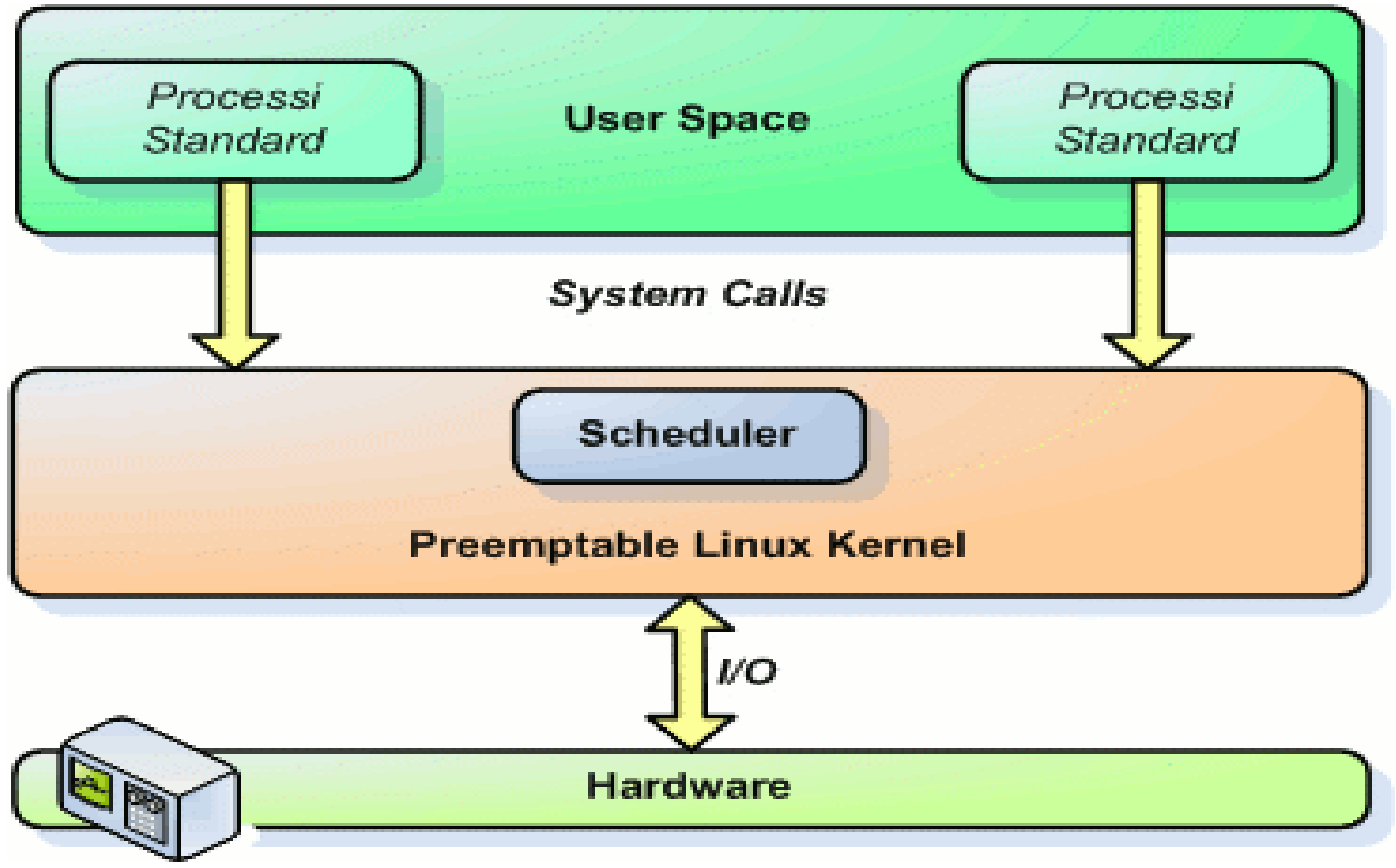


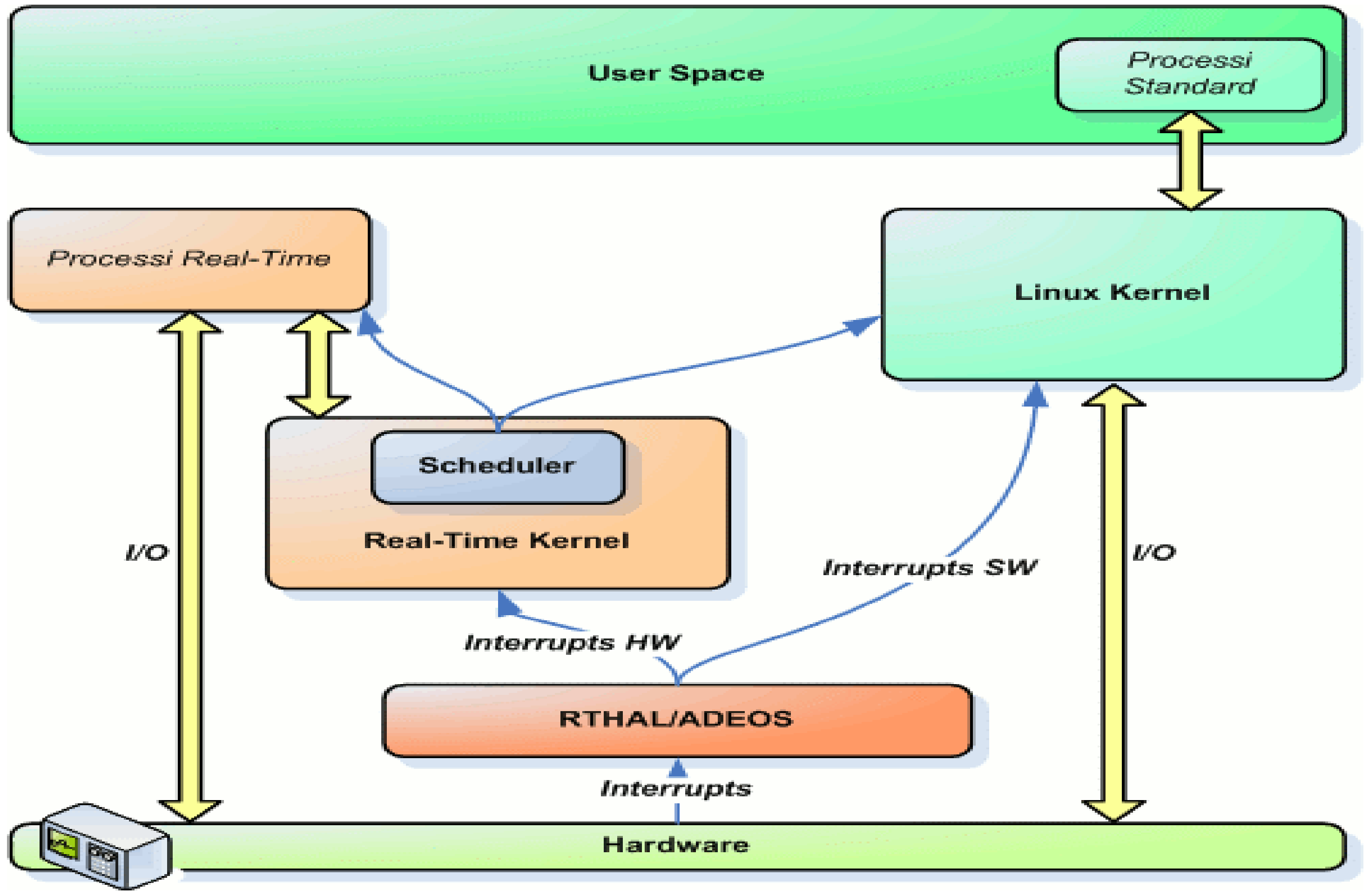


→ is really tuned for data acquisition ?


Well, nobody’s perfect

Appendix F: 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/