Electronics, Trigger and Data Acquisition
part 2

Summer Student Programme 2016, CERN

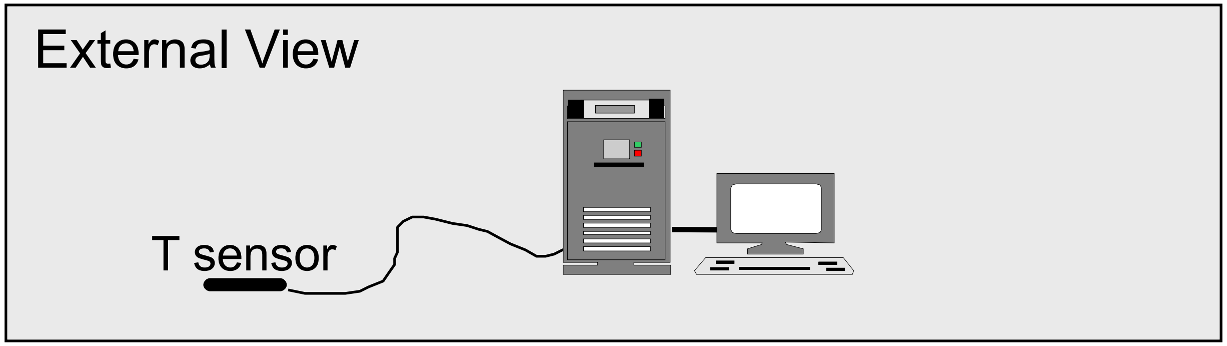
July 12, 2016

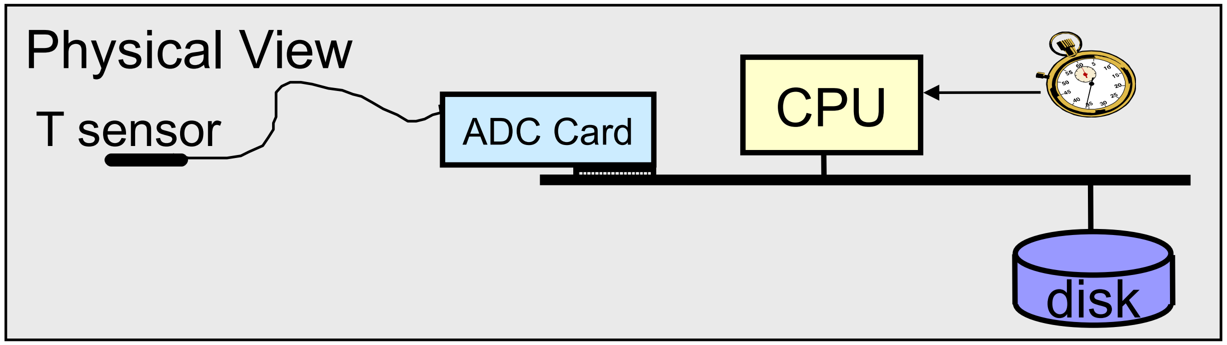
Roberto Ferrari
Instituto Nazionale di Fisica Nucleare

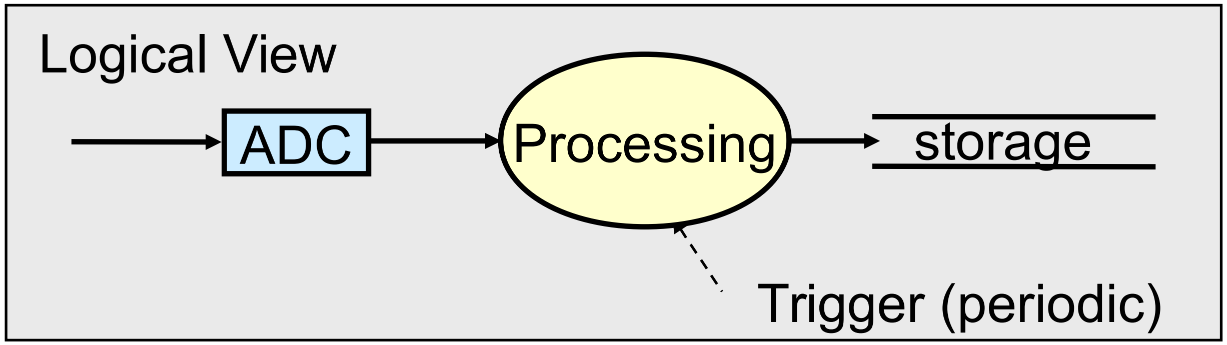
roberto.ferrari@pv.infn.it

Trigger & DAQ

Basic DAQ: Synchronous Trigger (1)

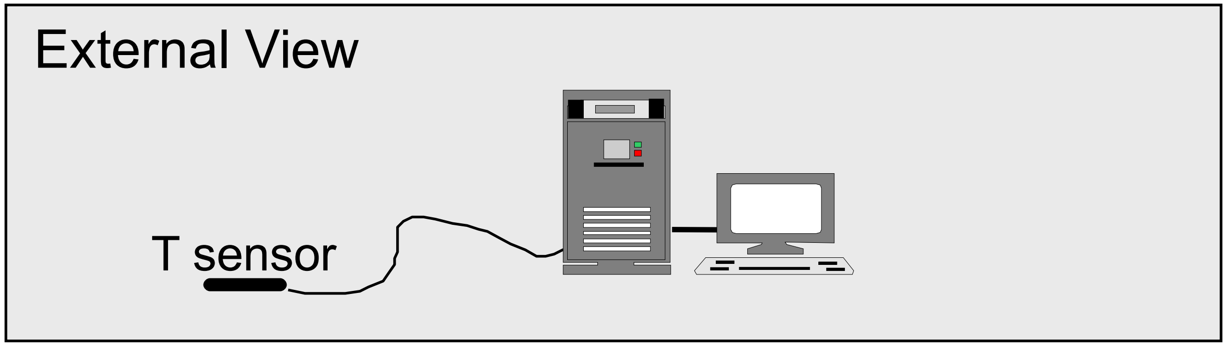


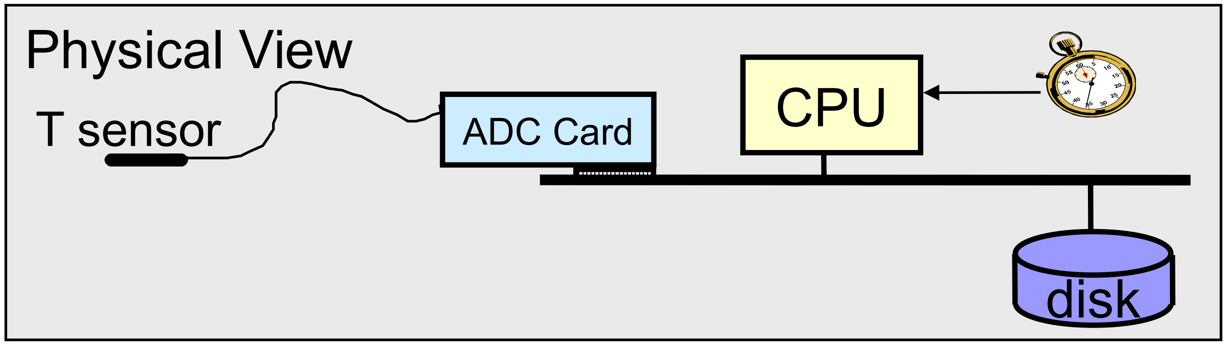


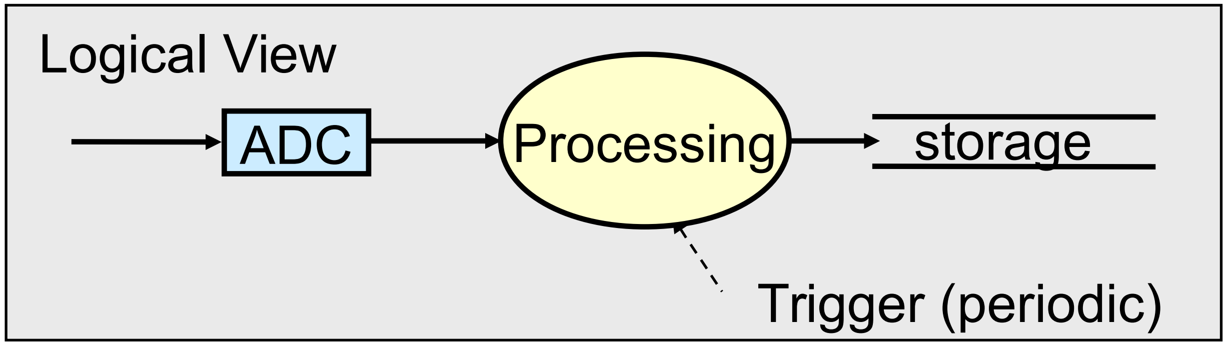


- Measure temperature at fixed frequency
- ADC performs analog-to-digital conversion
  - our front-end electronics
- CPU does readout and processing

Basic DAQ: Synchronous Trigger (2)







- Measure temperature at fixed frequency
- Full sequential → nothing going in parallel
- System limited by time needed to process one "event"
- If τ ~ 1ms for

  ADC conversion

  +CPU processing

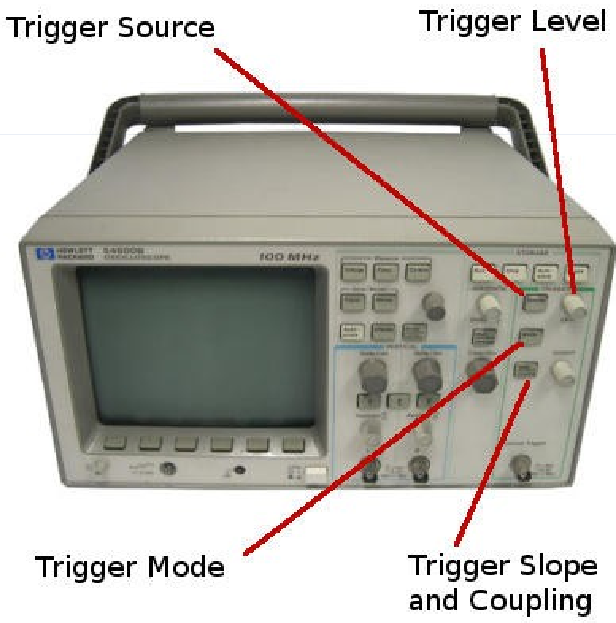
  +storage

  → can sustain up to 1/τ~1kHz of ***periodic (synchronous) trigger*** rate

What does "Trigger" mean?

- *Prompt signal,* built with "as simple as possible" criteria, claiming that, possibly, something interesting took place, initiating the data-acquisition process *["please, look at that"]*
- Keywords: simple, rapid, selective
    - selective = efficient for "signal" & resistant to "background"
- Actual parameters strongly dependent on operating conditions
    - in multi-level trigger system, "next" level way slower and more complex than preceding one

The oscilloscope trigger does exactly this: informs the instrument to initiate the internal signal acquisition and visualization



How Trigger was born

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

Walther Bothe: (1924-1929) offline → online coincidence (logic **AND**) of 2 signals

Bruno Rossi: "Method of Registering Multiple Simultaneous Impulses of Several Geiger Counters" (Nature, 1930), online coincidence of 3 signals (expandable)

"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

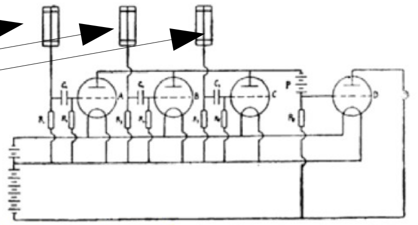


Fig. 17 – Il circuito di Rossi per rivelare coincidenze di raggi cosmici che arrivano sui contatori Geiger (i rettangoli in alto dello schema)[19].

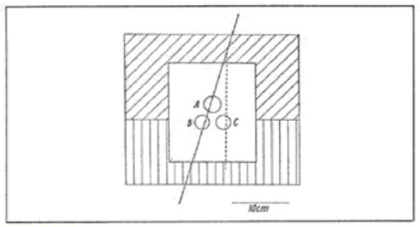
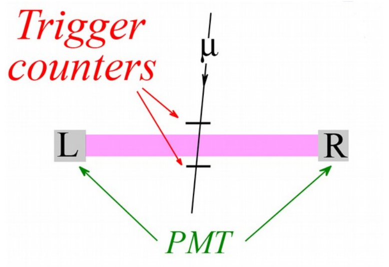
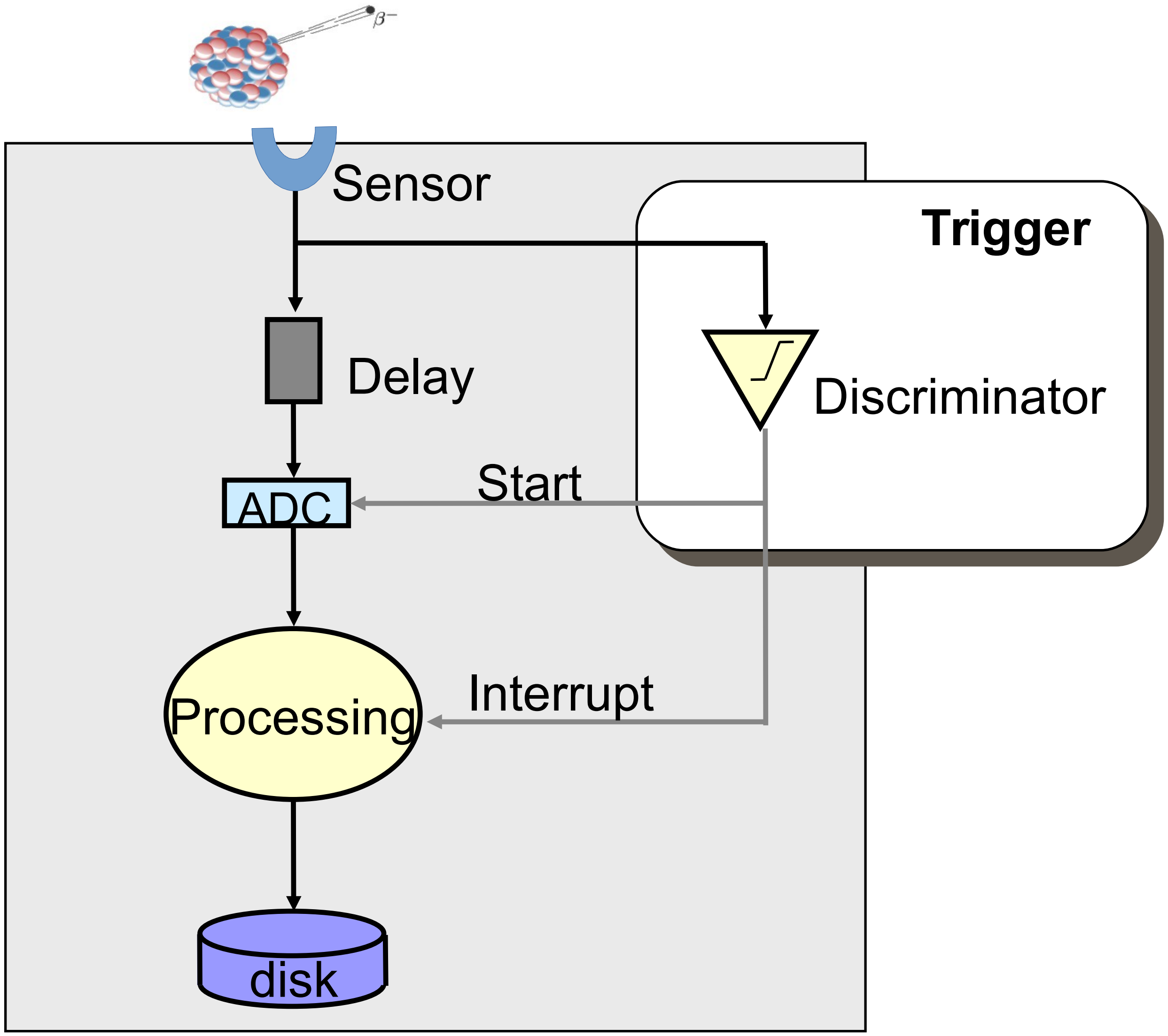


Fig. 18 – L'uso del circuito di Rossi per rivelare una coincidenza tripla che, nella disposizione in figura dei tre contatori, mostra la produzione di una radiazione secondaria (linea tratteggiata) da parte della radiazione primaria (linea continua)[20].



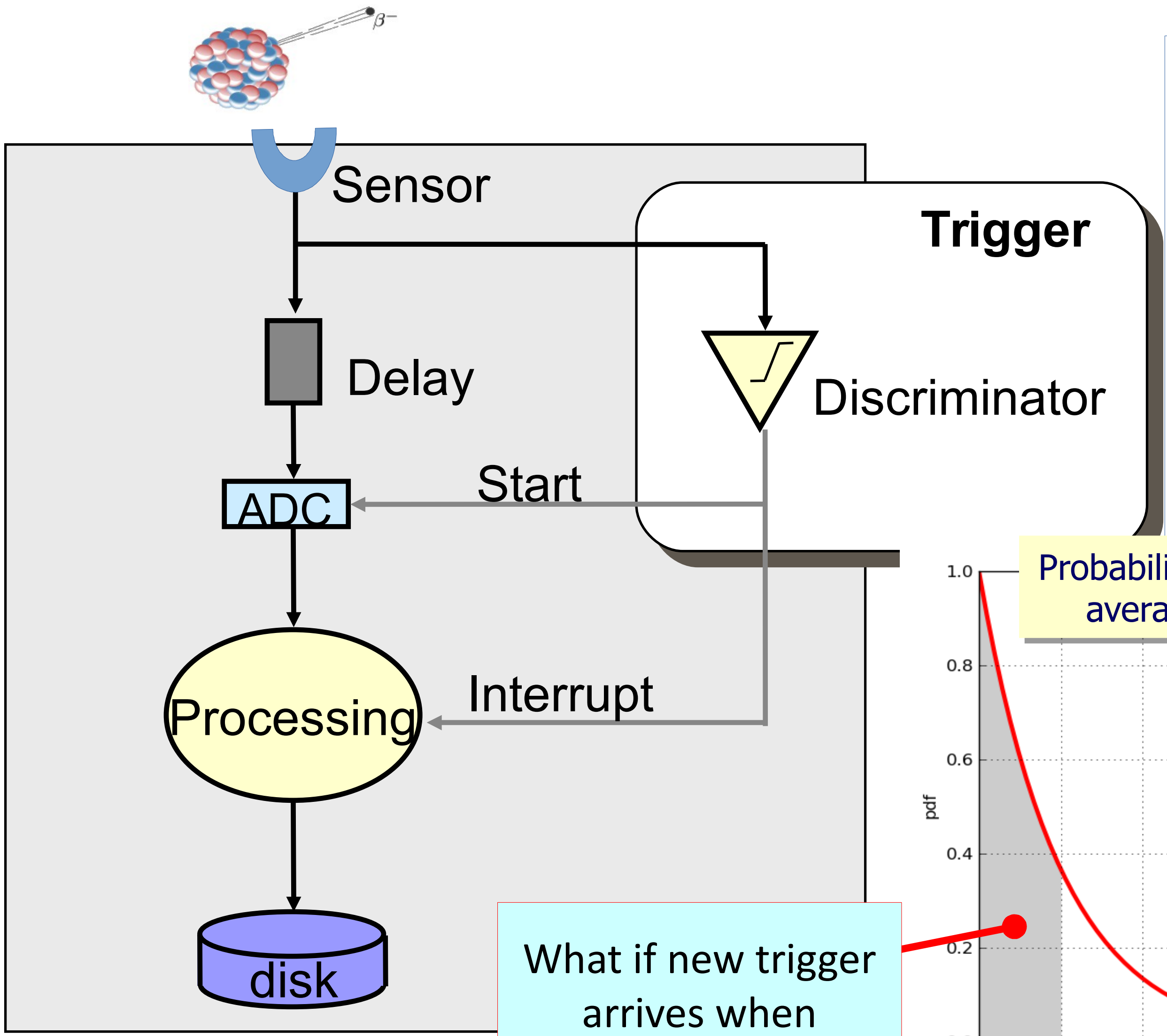
simplest case: 2-signal coincidence

Basic DAQ: Physics Trigger



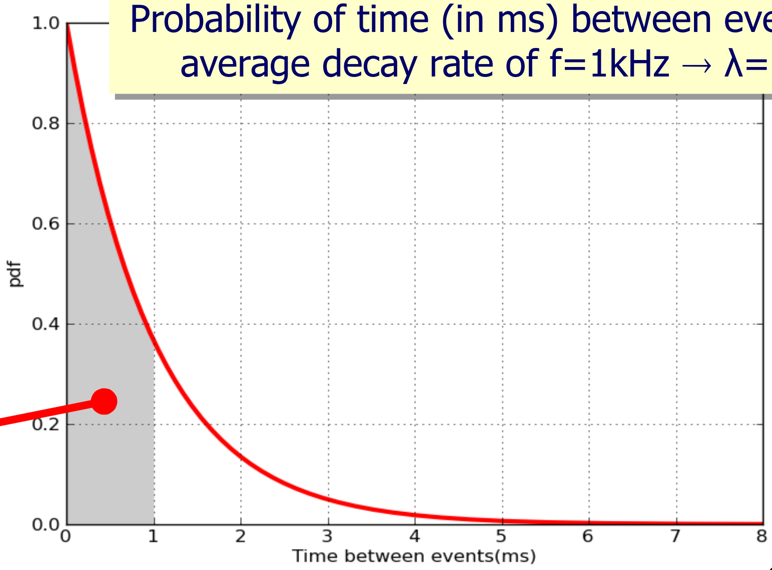
- Measure β decay properties
- Asynchronous and unpredictable events
  - need a **physics** trigger
- Delay compensates for trigger latency
  - time needed to reach a decision
- When system busy (=not ready to react to triggers) → dead time

Basic DAQ: Real Trigger



- Measure β decay properties
- Stochastic (i.e. fully uncorrelated) process
    - fluctuations

Probability of time (in ms) between events for average decay rate of f=1kHz → λ=1ms



What if new trigger arrives when system busy?

Basic DAQ: don't loose any event ?

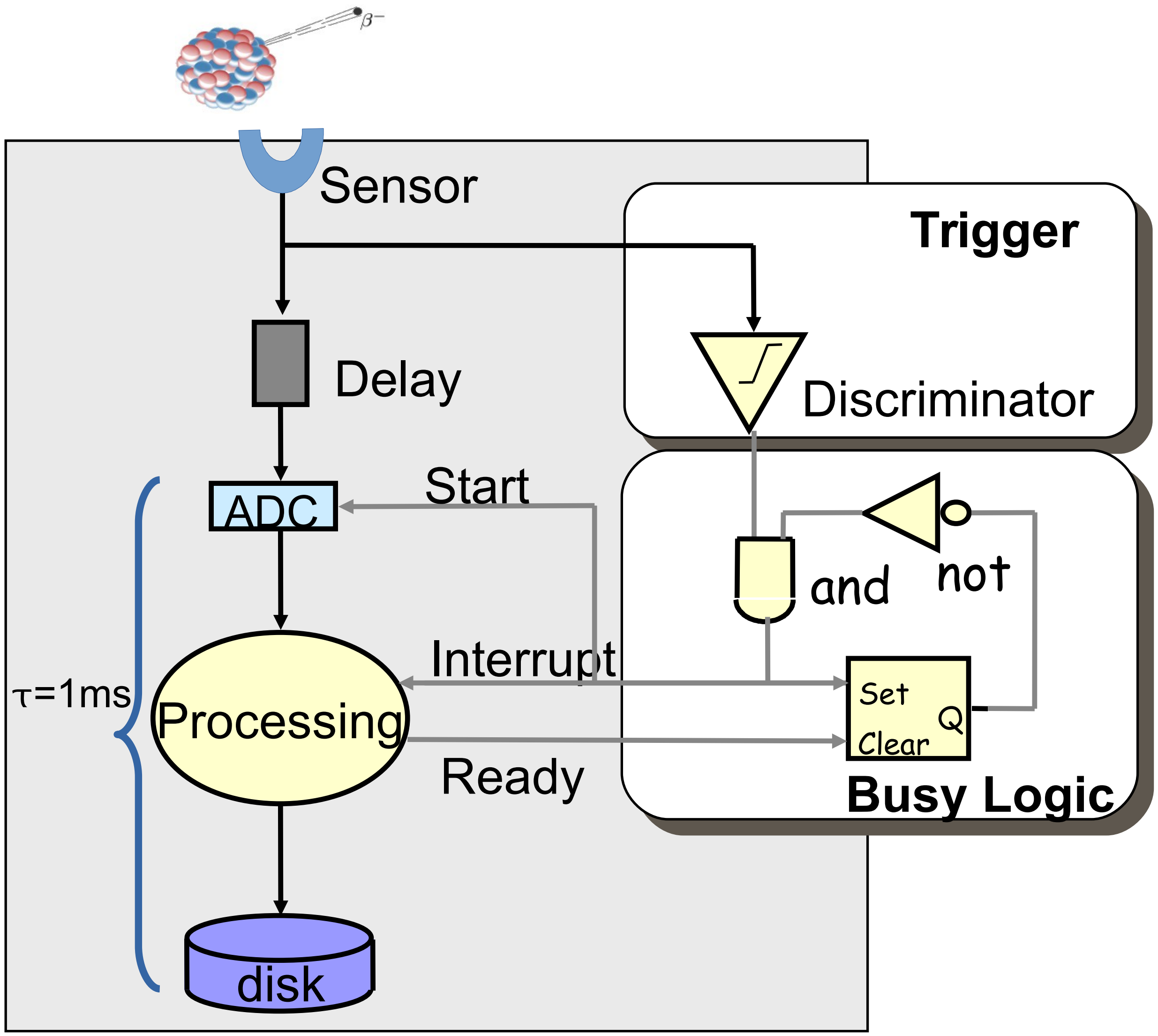
a) Retriggerable DAQ system: any new trigger accepted, each time causing dead-time restart, regardless of DAQ state

$\Rightarrow$ paralysable DAQ

b) Non retriggerable : none new trigger until dead time elapsed

$\Rightarrow$ non-paralysable DAQ

Basic DAQ: Real Trigger & Busy



- Busy logic avoids triggers while processing
- Which (average) DAQ rate can we achieve now?

Reminder: τ=1ms sufficient to fully handle 1kHz synchronous trigger

DAQ Dead Time & Efficiency (1)

Being: τ = DAQ dead time (per event) ; f = average signal rate ; ν = average acquisition rate

$\rightarrow \nu \cdot \tau$ = total DAQ dead time $\Rightarrow$ $(1-\nu\cdot\tau)$ = total DAQ available time

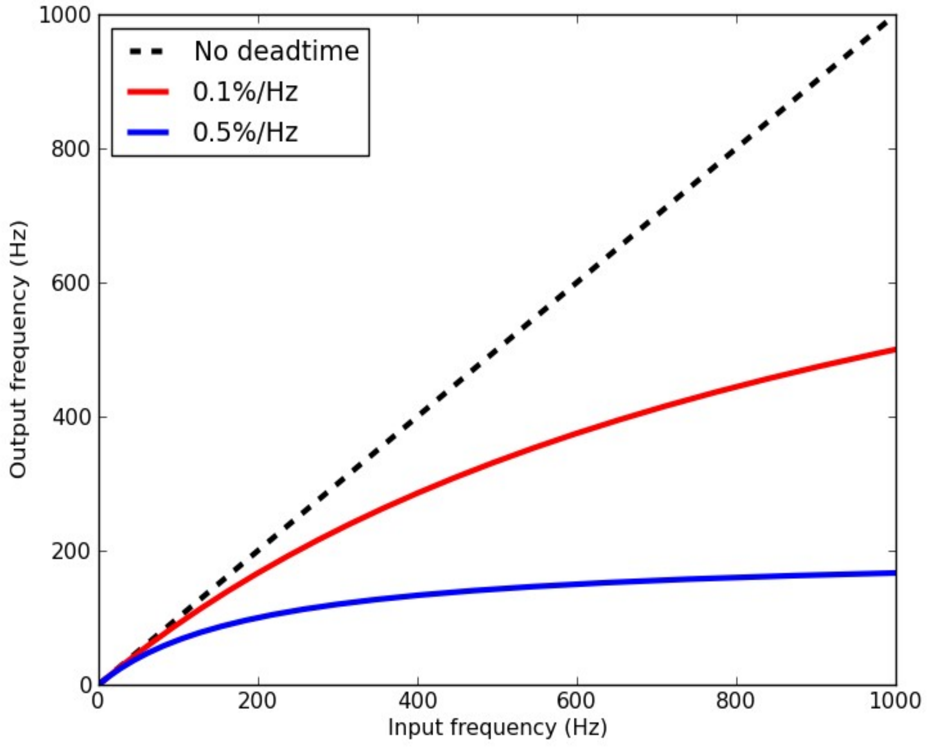
$\rightarrow f\cdot(1-\nu\cdot\tau) = \nu \quad \Rightarrow \quad \nu = f/(1+f\cdot\tau) < f,\ 1/\tau$

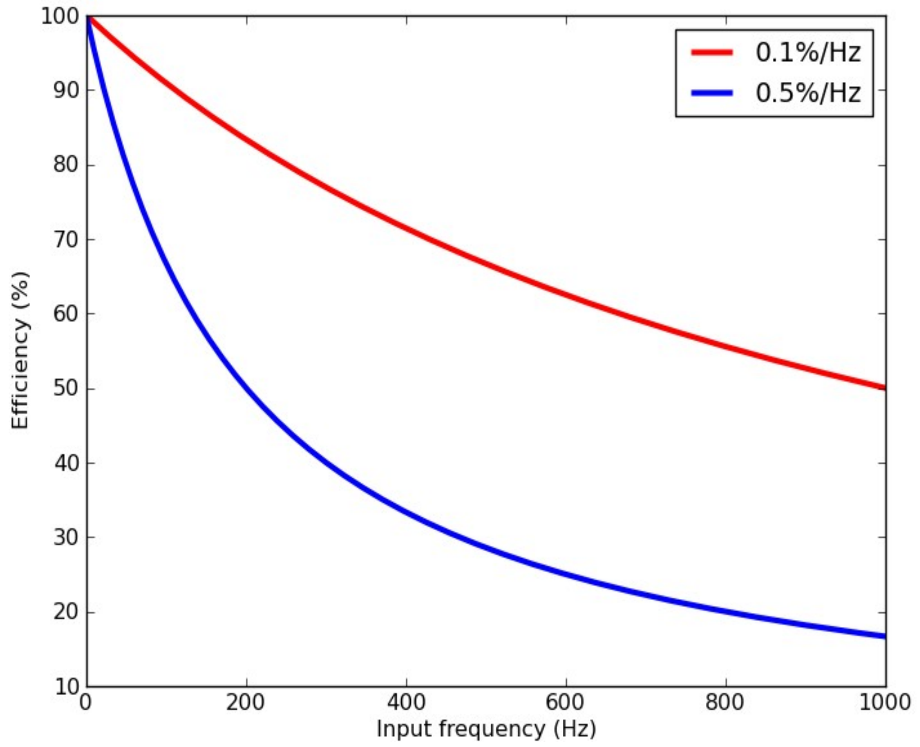
Efficiency $\varepsilon = \nu/f = 1/(1+f\cdot\tau) < 100\%$

Dead time $(1-\varepsilon) = f\cdot\tau/(1+f\cdot\tau) \quad \Rightarrow \quad f\cdot\tau < (1-\varepsilon) < 1$

- Max acquisition speed ($f\rightarrow\infty$) $\nu\rightarrow 1/\tau$
- Due to stochastic fluctuations, efficiency will always be less than 100%
  - in our specific example, τ=1ms, f=1kHz $\Rightarrow$ ν=500Hz, ε=50%

DAQ Dead Time & Efficiency (2)

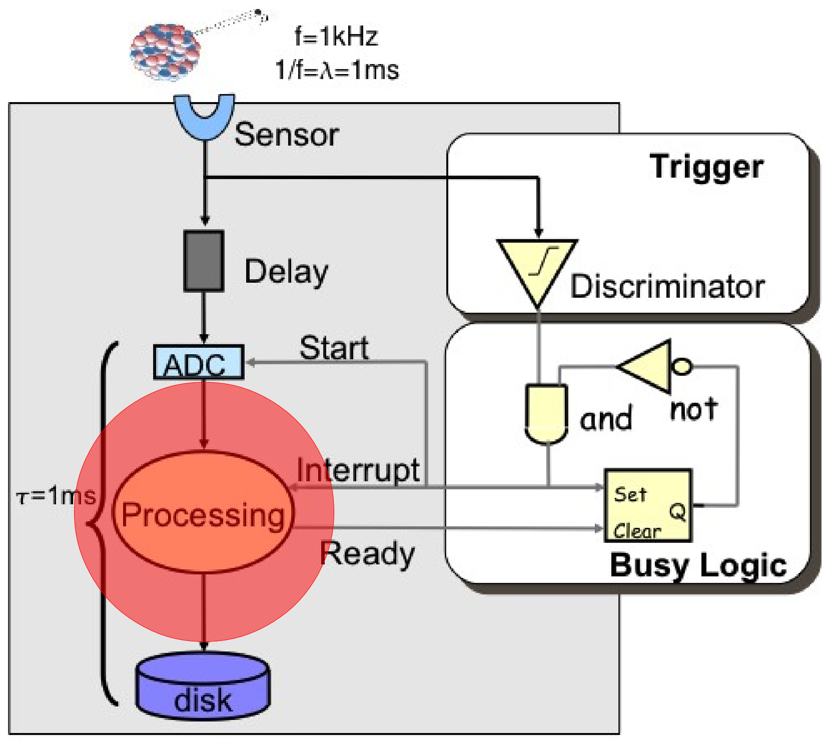




- Want: ν∼f (ε∼100%) ⇒ (f·τ)<<1 ⇒ τ<<1/f
    - f=1kHz, ε=99% ⇒ τ=0.01ms ⇒ 1/τ=100kHz
- In order to cope with input signal fluctuations, we have to over-design our DAQ system by a factor 100. Very inconvenient! Can we mitigate this effect?

Dead Time → de-randomise

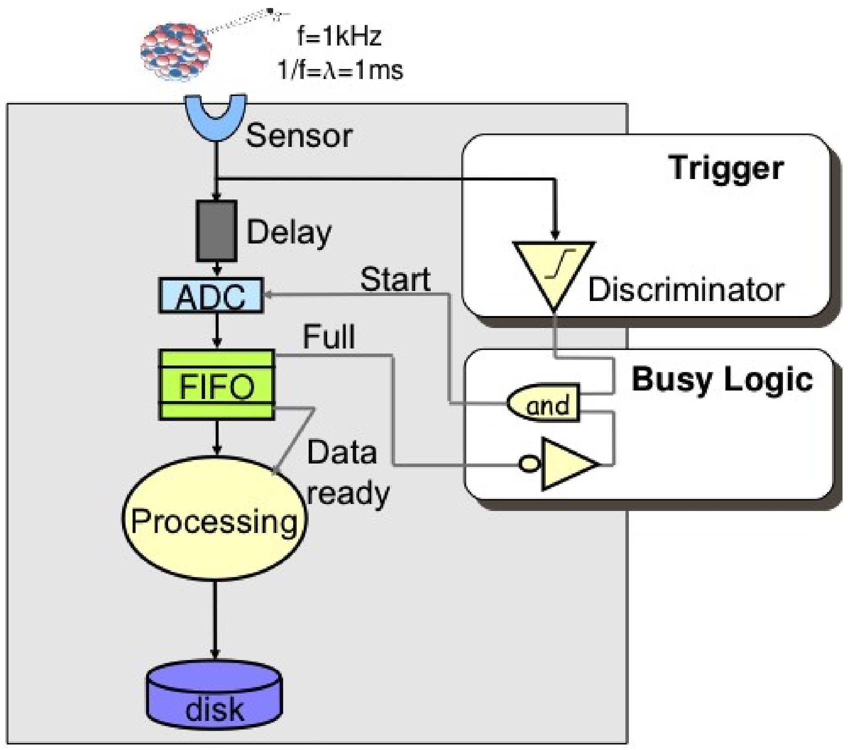
- Processing → bottleneck



Dead time ~ $(1+x)^{-1}$ ~ 50%

[for x = 1/(f·τ) ~ 1]

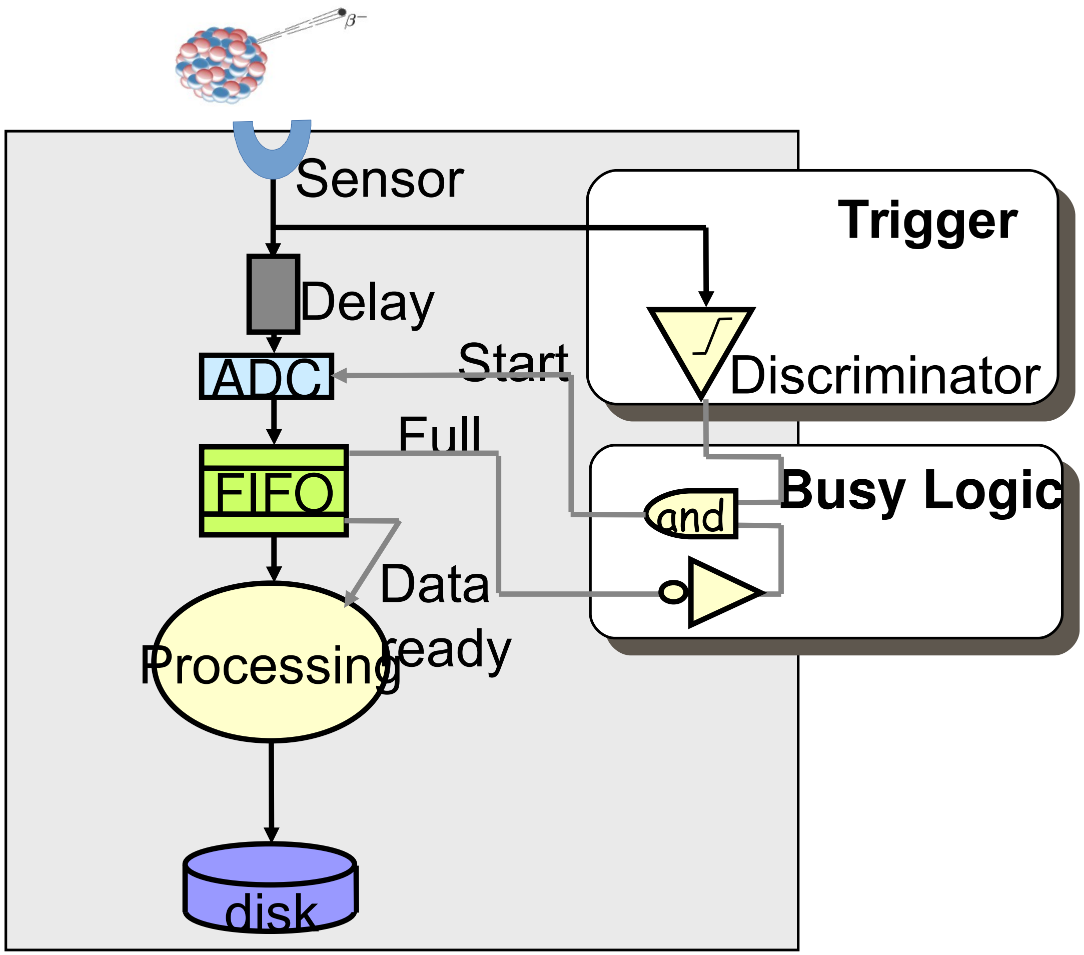
- Buffering allows to decouple problems



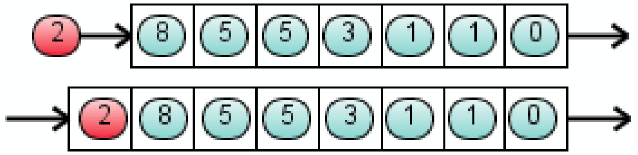
Dead time ~ $(\sum^{0..N} x^j)^{-1}$ ~ 1/(N+1)

[N = buffer depth]

Basic DAQ: De-Randomisation



- First-In First-Out
  - buffer area organized as a queue
  - depth: number of memory cells
  - implemented in HW and SW



- Buffering introduces additional latency on data path

FIFO absorbs and smooths input fluctuations, providing ~steady (de-randomised) output rate

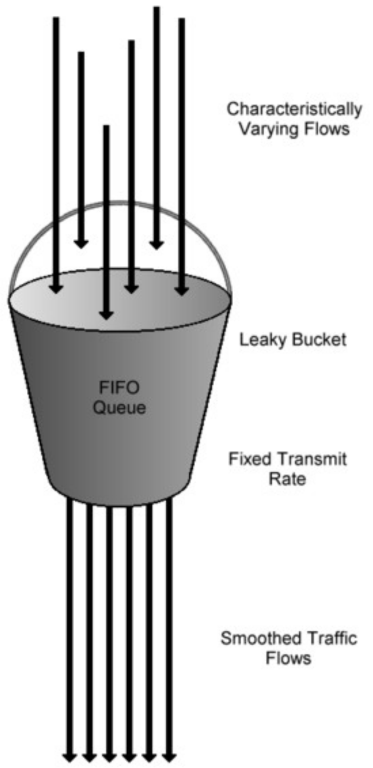
does buffering solve all problems ?

FIFO

- filled with very variable input flow
- emptied at smoothed output flow

  → the Leaky-Bucket problem

Q: how often may overflow ?



Some (Candid) Queueing Theory

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

P_k : % time with k events in ; P_N = no space available $\rightarrow$ dead time

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

rate($j \rightarrow j+1$) = $f \cdot P_j$ (fill at rate f)

rate($j+1 \rightarrow j$) = P_{j+1}/τ (empty at rate $1/\tau$)

steady state: $f \cdot P_j = P_{j+1}/\tau \;\Rightarrow\; P_j = P_{j+1}/(f\tau) = x \cdot P_{j+1}$

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

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

want $\leq$ 1% $\Rightarrow$ N $\geq$ 100

Some (Candid) Queueing Theory

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

P_k : % time with k events in ; P_N = no space available $\rightarrow$ dead tim

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

rate($j \rightarrow j+1$) = $f \cdot P_j$ (fill at rate f)

rate($j+1 \rightarrow j$) = P_{j+1}/τ (em ate $1/\tau$)

steady $P_j = P_{j+1}/\tau \Rightarrow P_j = P_{j+1}/(f\tau) = x \cdot P_{j+1}$

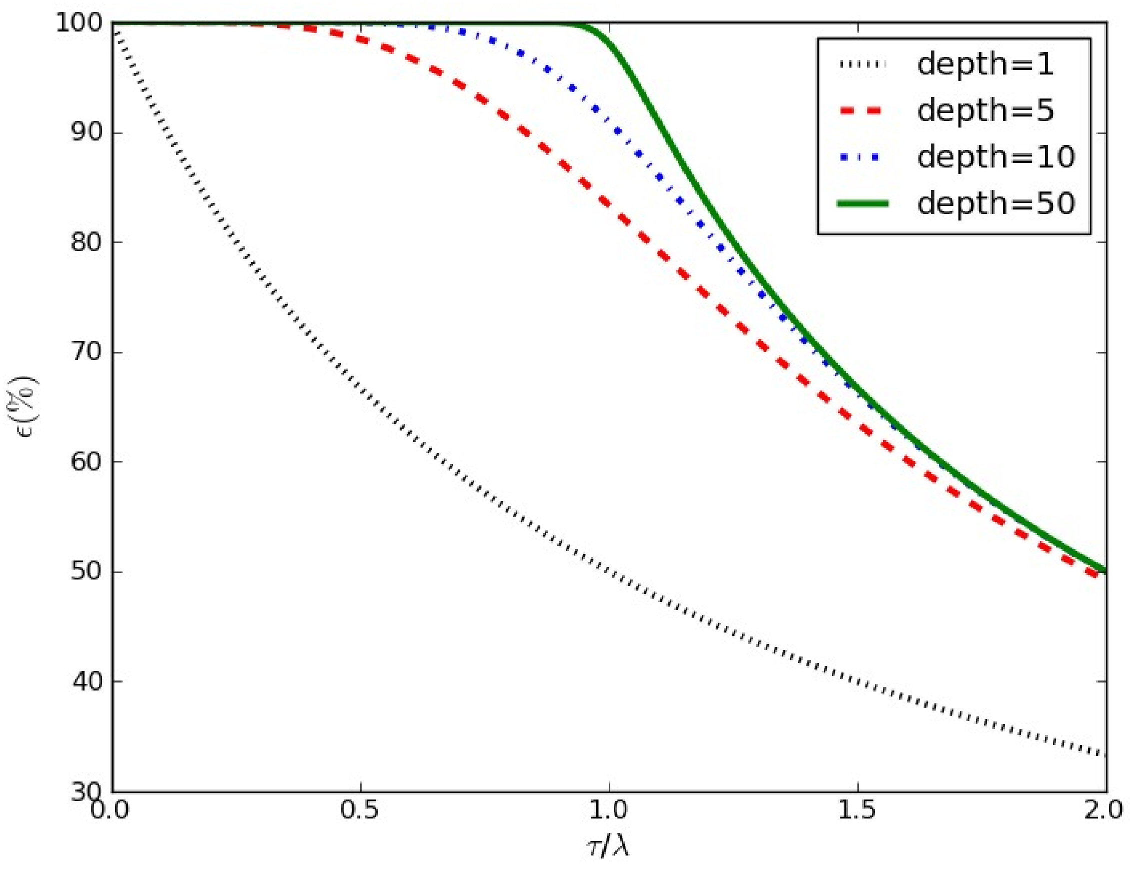
$x \sim 1 \Rightarrow P_j \sim P_{j+1} \Rightarrow \sum P_k \sim (N+1) \cdot P_0 = 1 \Rightarrow P_0 \sim 1/(N+1)$

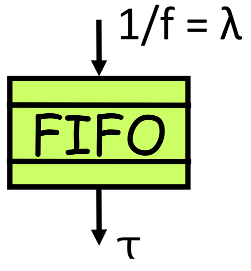
$\Rightarrow$ dead time $\sim 1/(N+1)$

want ≤ 1% ⇒ N ≥ 100

Take care: analytic calculation possible for pretty simple systems only

De-Randomisation: Queueing Theory





- We can now attain a FIFO efficiency ~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")

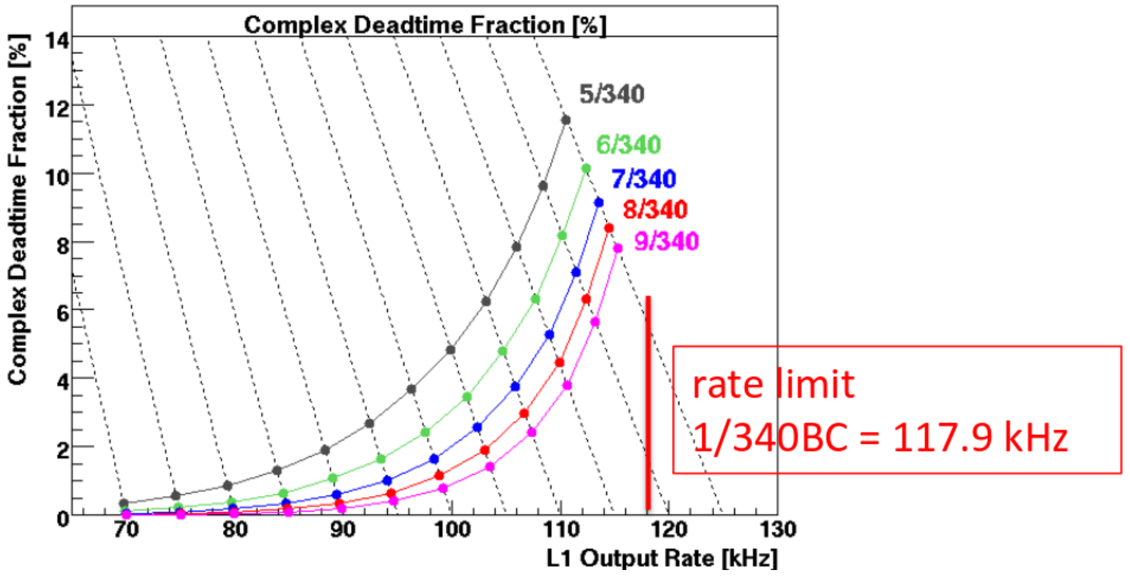
Dead Time: Summary

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

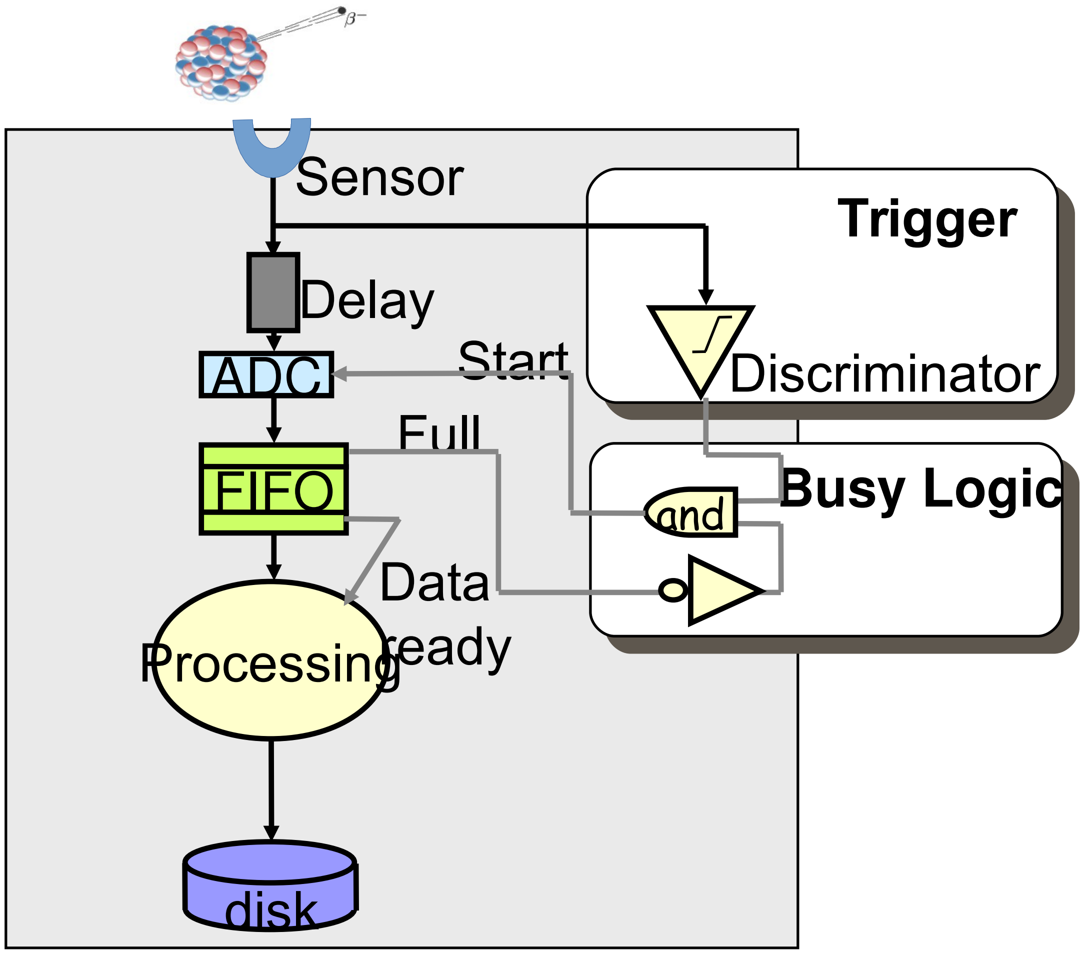
2) Complex dead time: avoid overflow in front-end buffers (protection against trigger bursts)

ATLAS uses simply leaky-bucket algorithms with 2 parameters:

max X triggers (X = FIFO depth) in any (sliding) time window = (X*readout time)

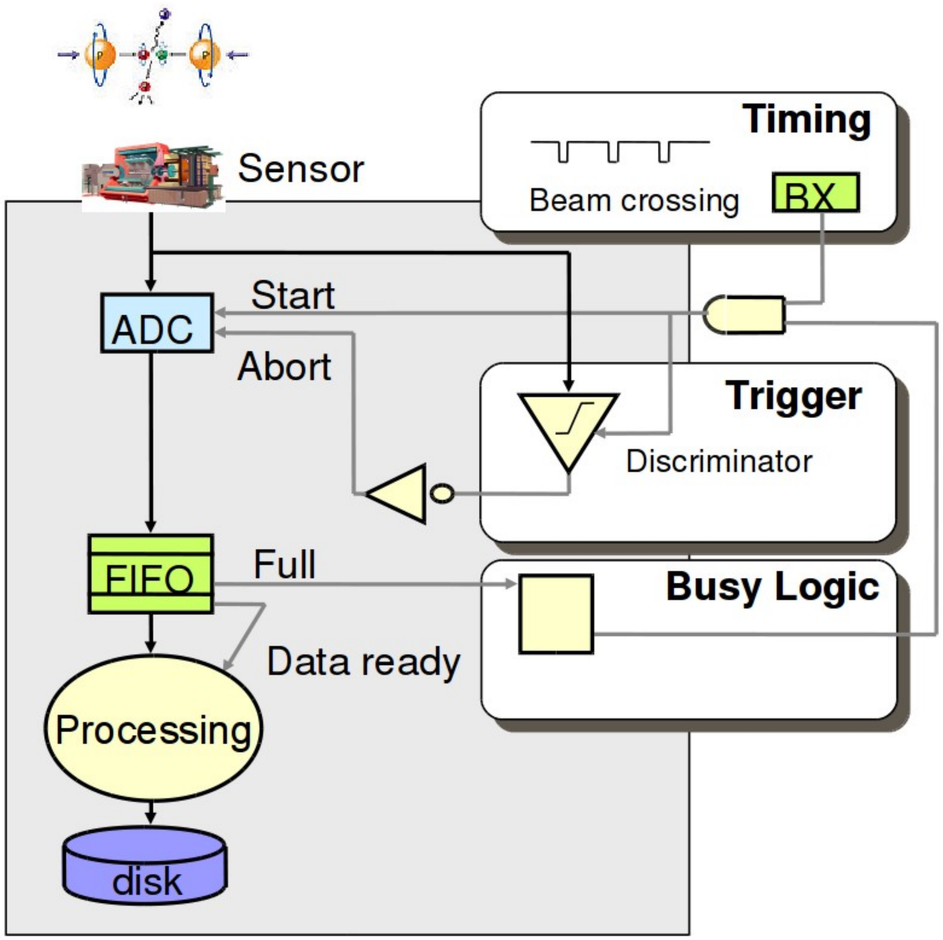


De-Randomisation: Summary



- Almost 100% efficiency and minimal deadtime may be achieved if
  - ADC able to operate at rate >> f
  - data processing and storing operates at ~f
- FIFO decouples low latency front-end from data processing
  - minimize the amount of “unnecessary” fast components
- Could “Delay” be replaced with “FIFO”?
  - analog pipelines → heavily used in LHC DAQs

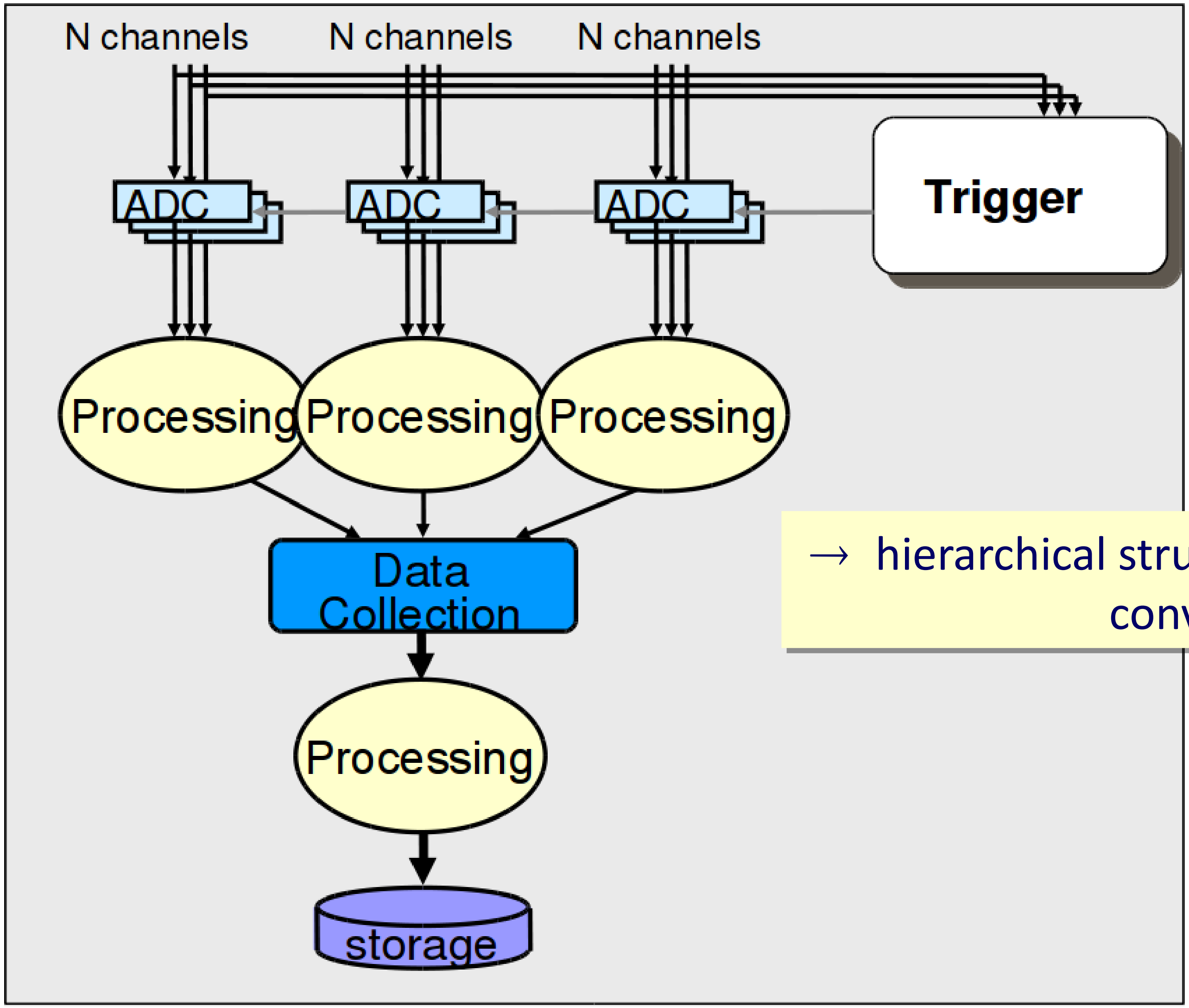
Basic DAQ: Collider Mode



- Synchronous particle collision rate
- Trigger rejects (= does not select) uninteresting events
- Even if collisions are synchronous, triggers unpredictable and uncorrelated
- De-randomisation still needed

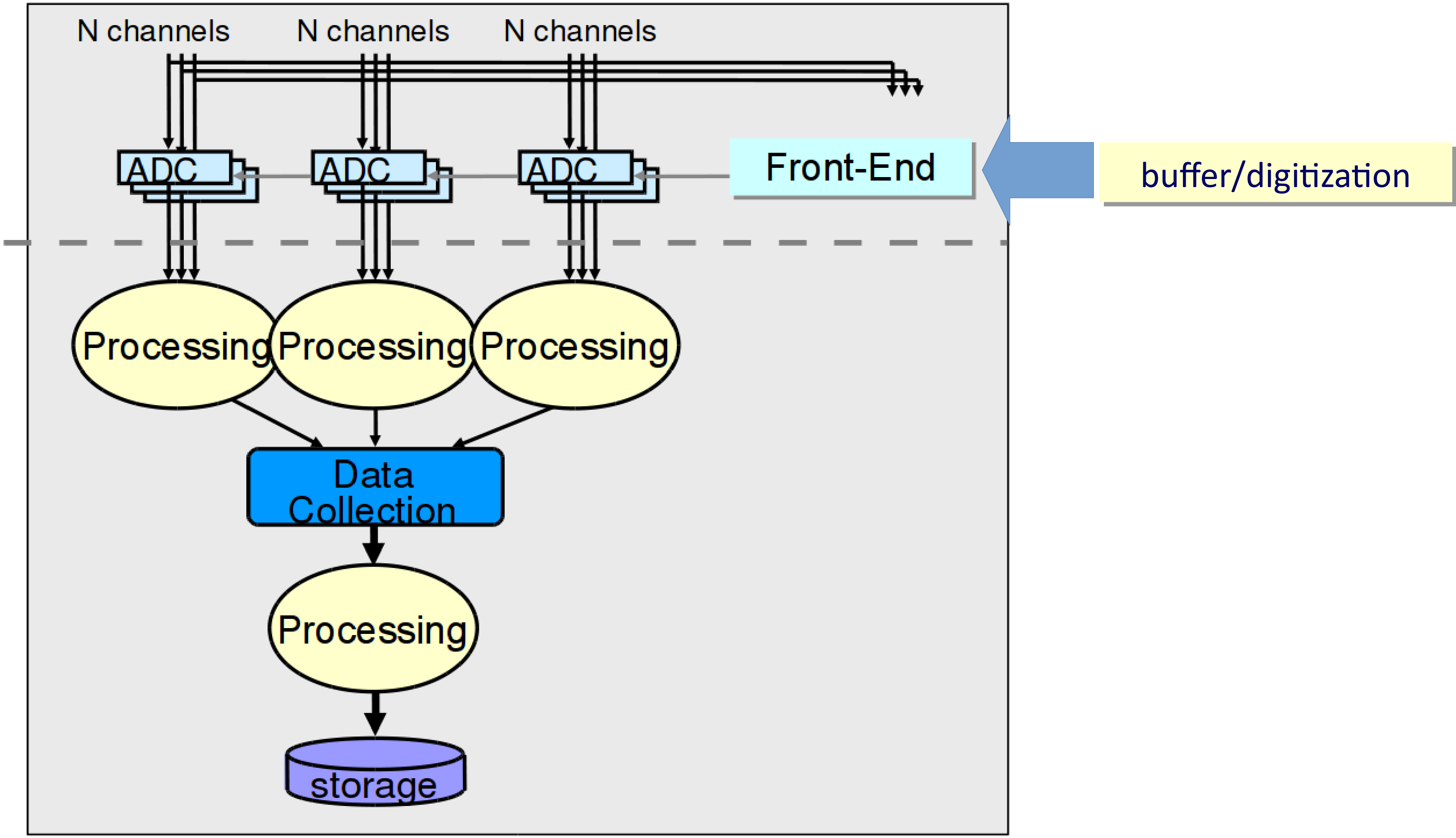
Scaling up: Network & Buses

Basic DAQ: More Channels

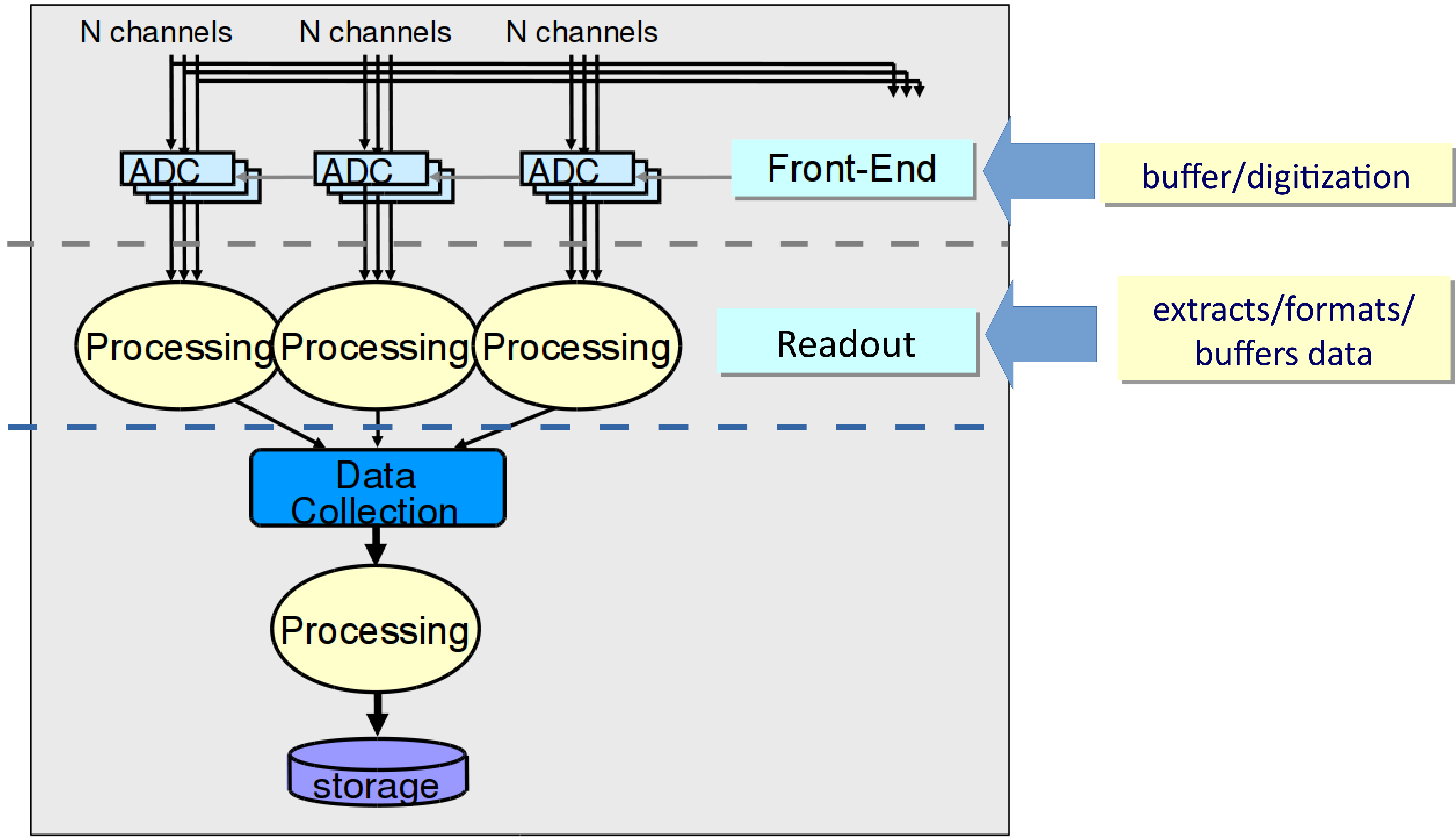


→ hierarchical structure for handling and conveyance

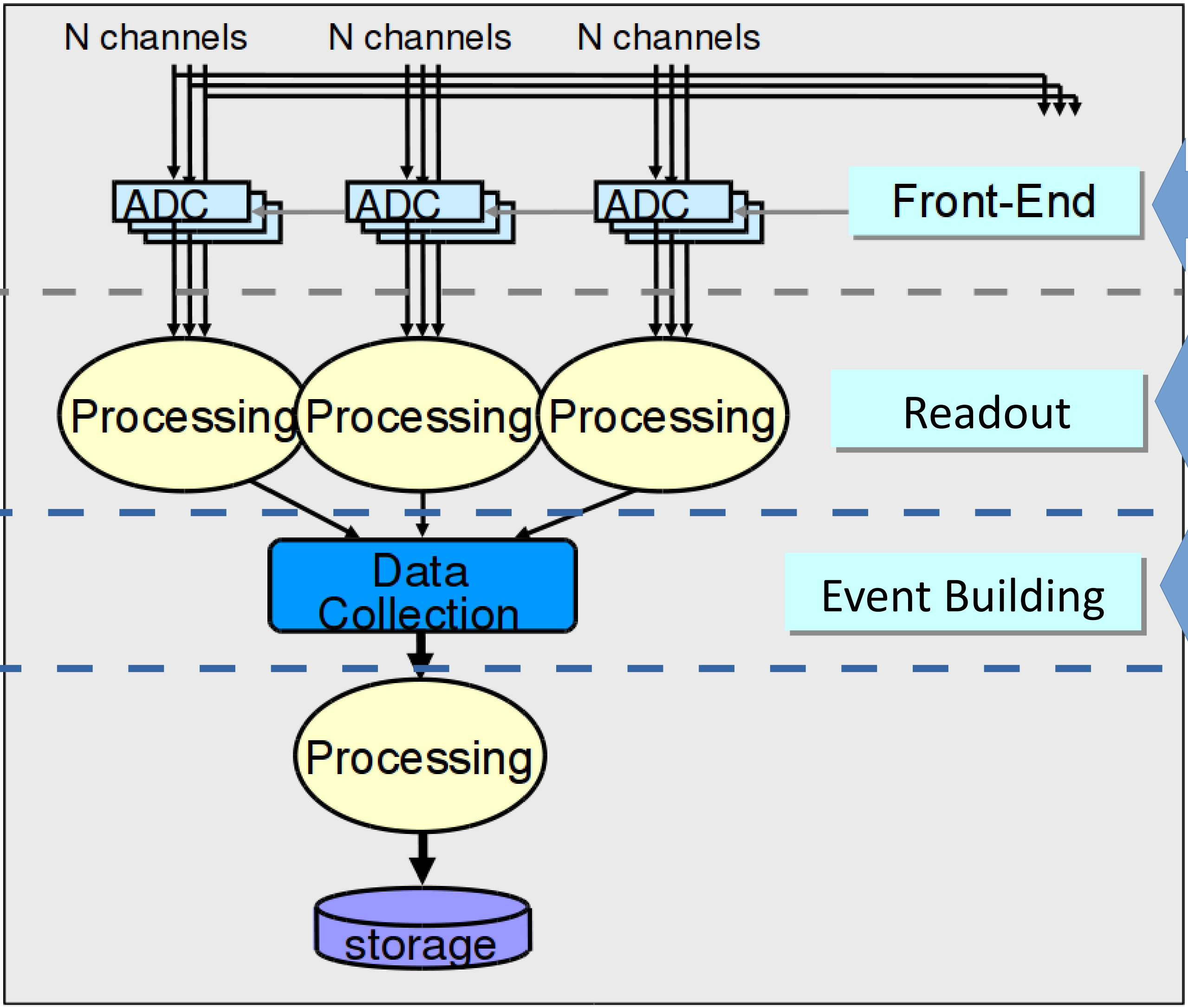
Large DAQ: Constituents

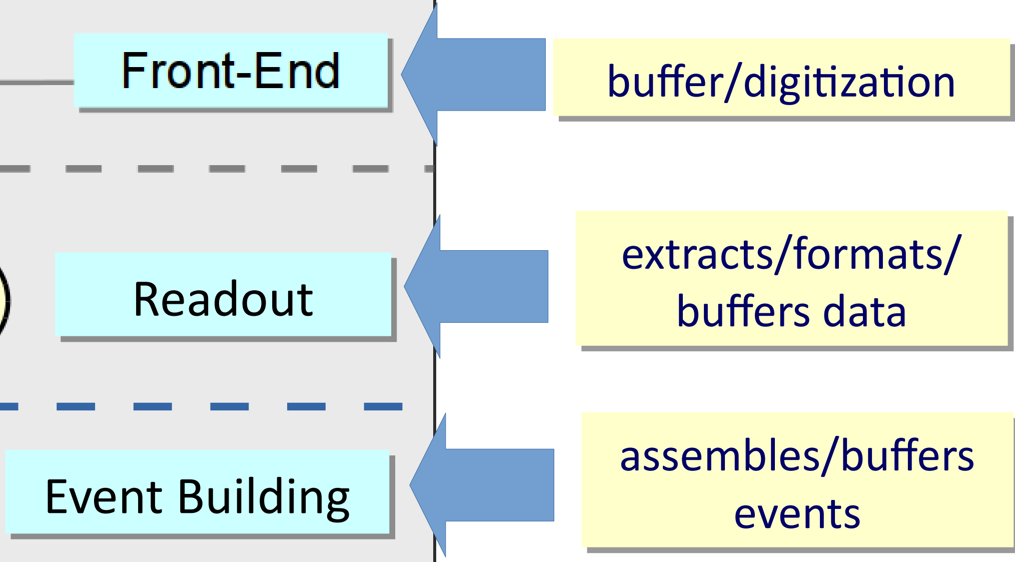


Large DAQ: Constituents

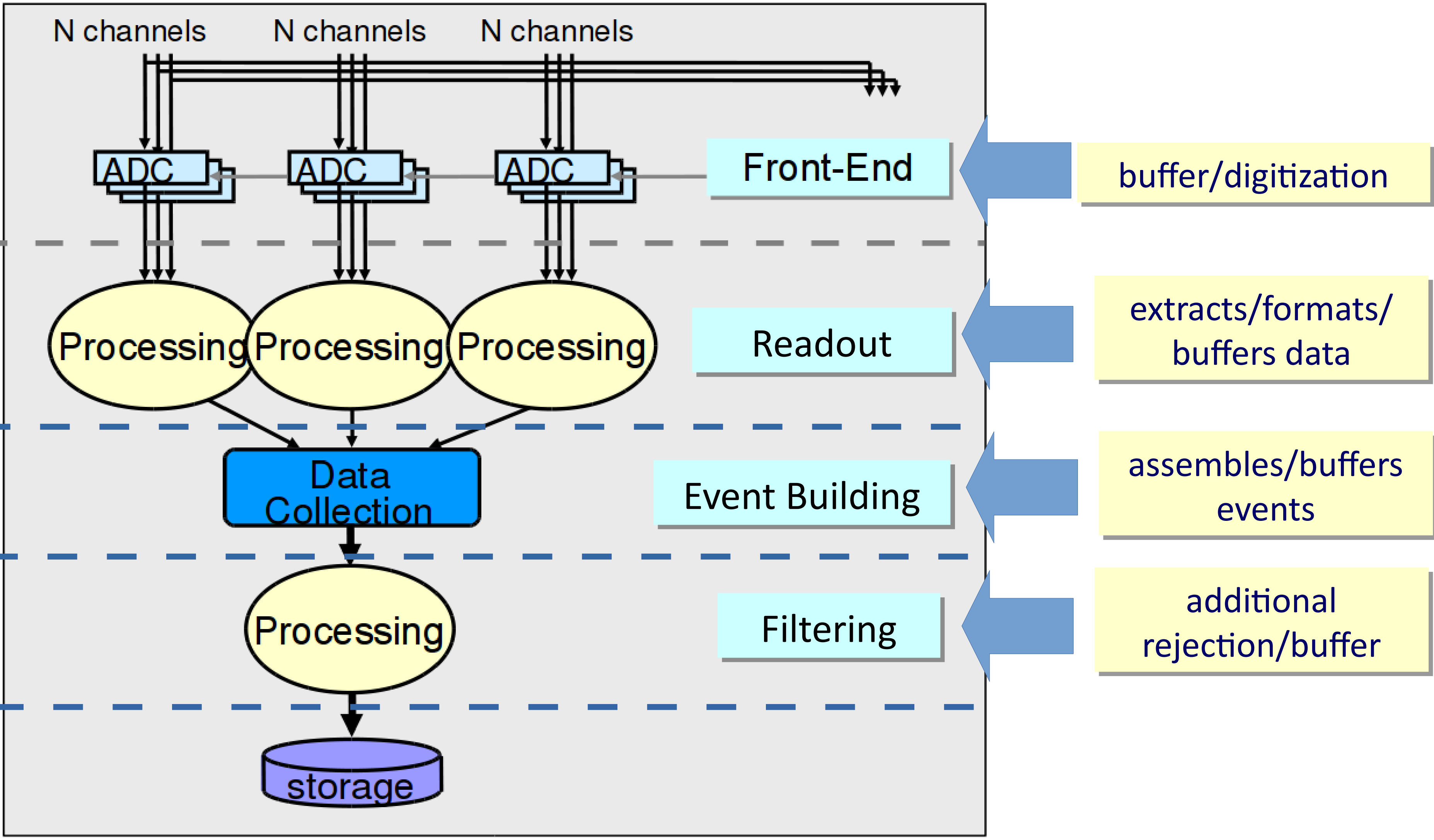


Large DAQ: Constituents

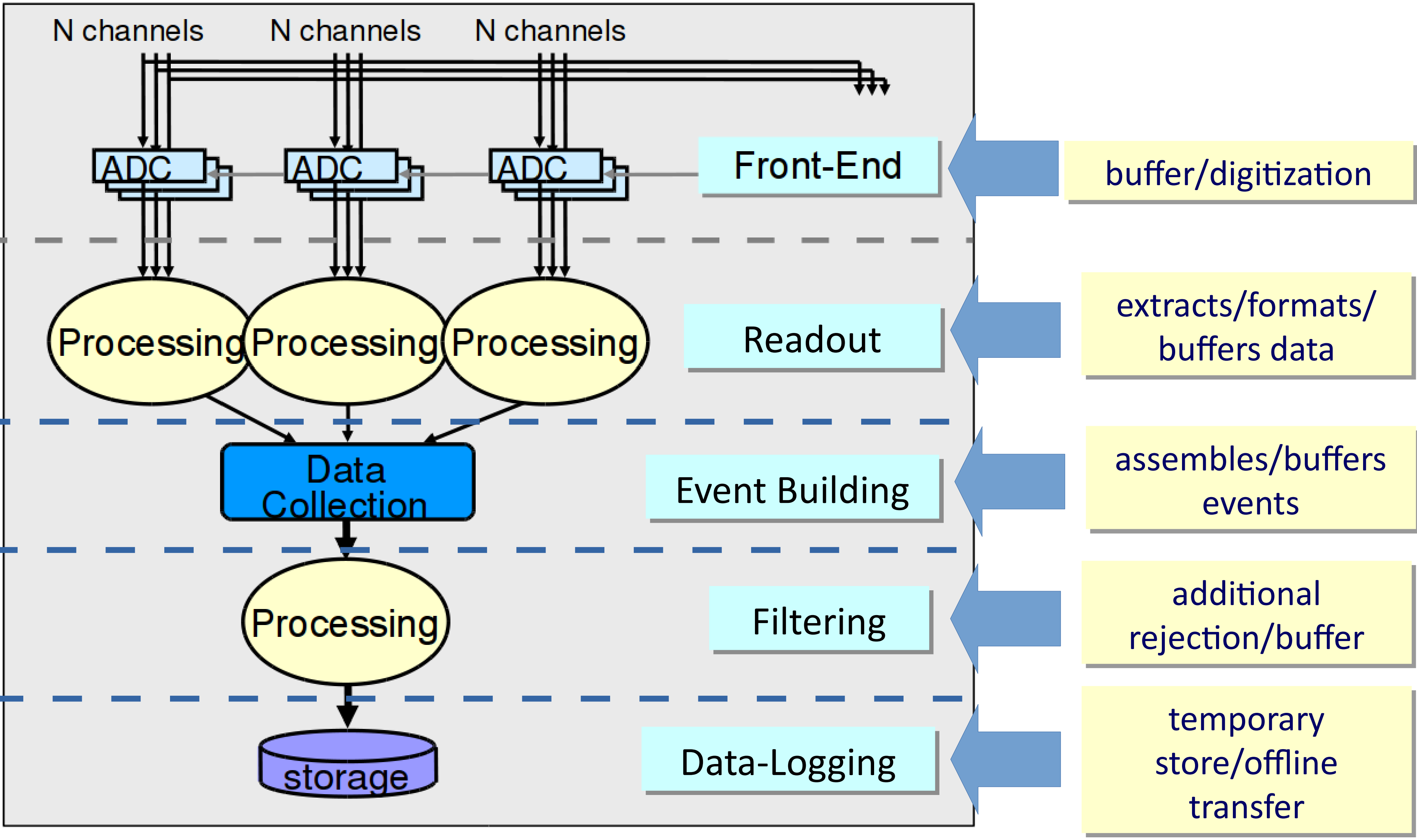




Large DAQ: Constituents

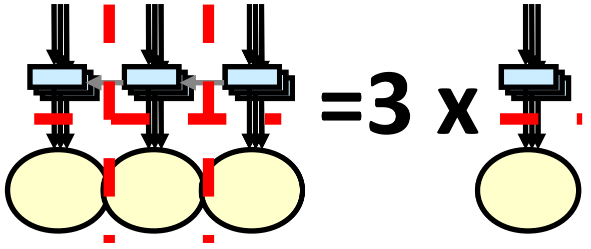


Large DAQ: Constituents

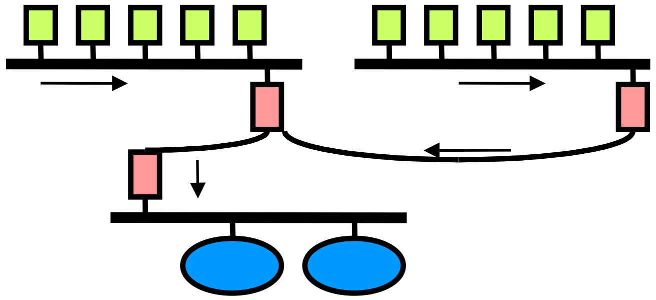


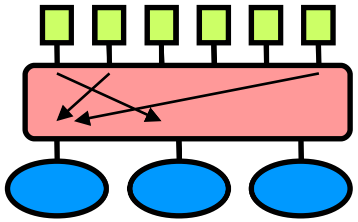
Readout Topology

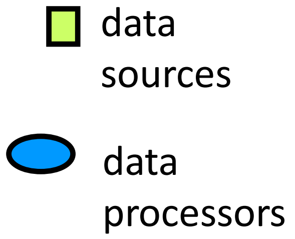
- Reading out or building events out of many channels requires many components
- *Possibly want a modular, scalable system*
- In designing our hierarchical data-collection system, we have better define "building blocks"
  - example: readout crates, event building nodes, …



- How to organize interconnections inside and between building blocks ?
- Two main classes: buses or network

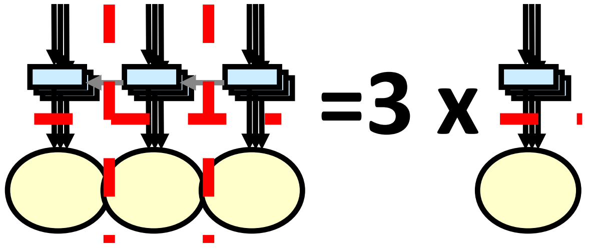




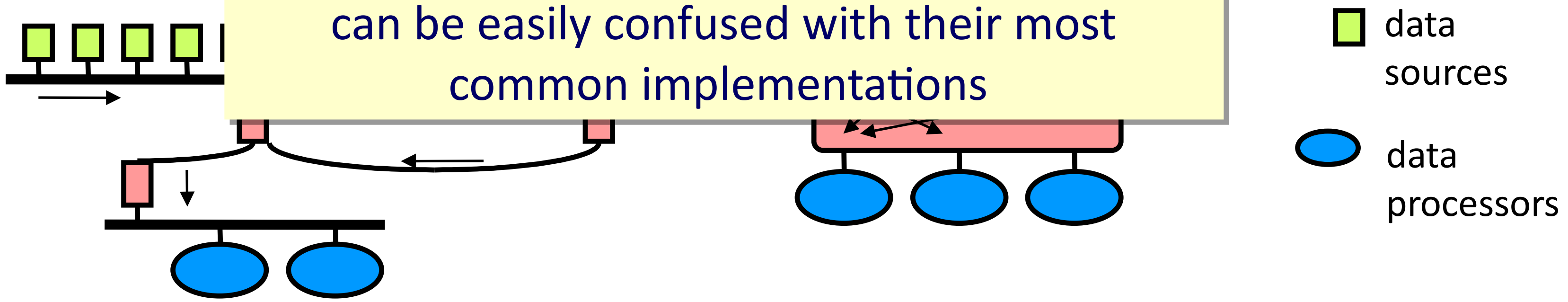


Readout Topology

- Reading out or building events out of many channels requires many components
- *Possibly want a modular, scalable system*
- In designing our hierarchical data-collection system, we have better define "building blocks"
  - example: readout crates, event building nodes, …



- How to organize interconnections inside and between building blocks ?
- Two main classes: buses or network

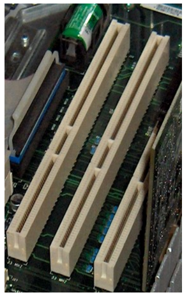


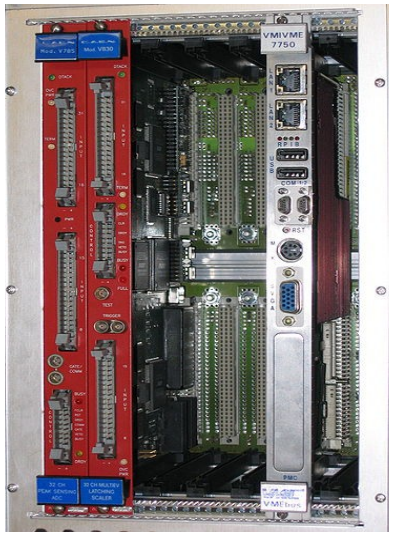
Warning

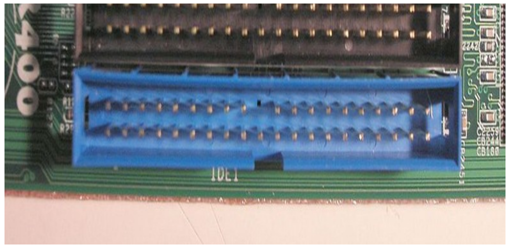
Buses and network are generic concepts that can be easily confused with their most common implementations

Buses

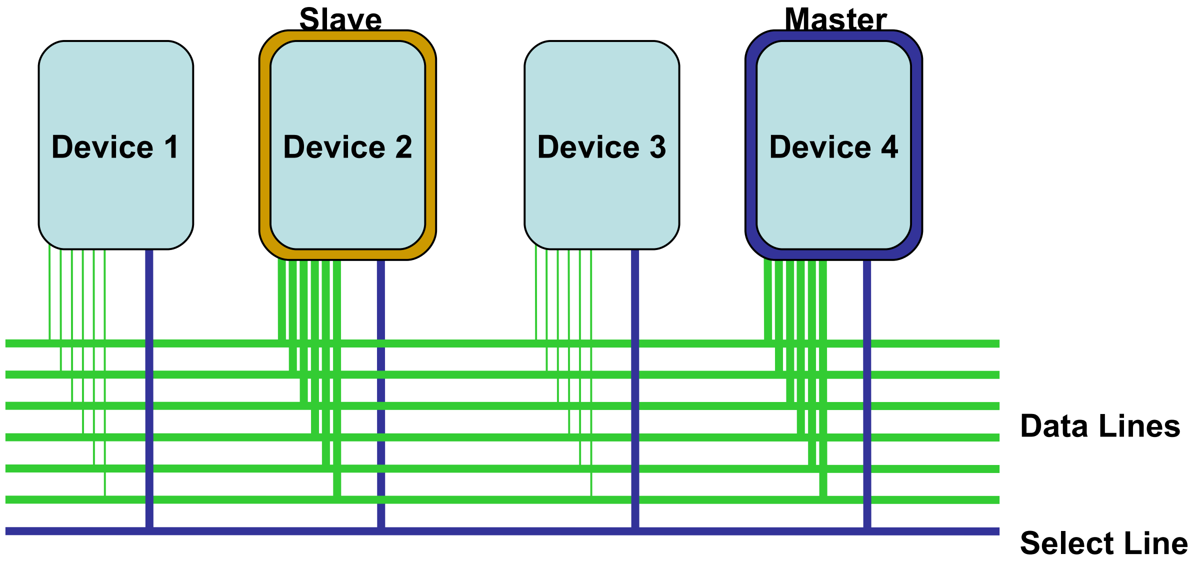
- Examples: VME, PCI, SCSI, Parallel ATA, …
  - local, external, crate, long distance
- Devices connected via shared lines (bus)
  - bus → group of electrical lines
  - sharing implies arbitration





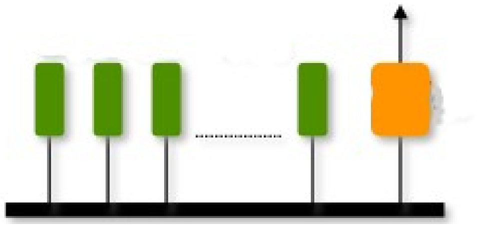


- Devices can be master or slave
- Device can be addressed (uniquely identified) on the bus



Bus Facts

Simple ✔
- fixed number of lines (bus-width)
- devices have to implement well defined hw/sw protocols
  - mechanical, electrical, communication, ...

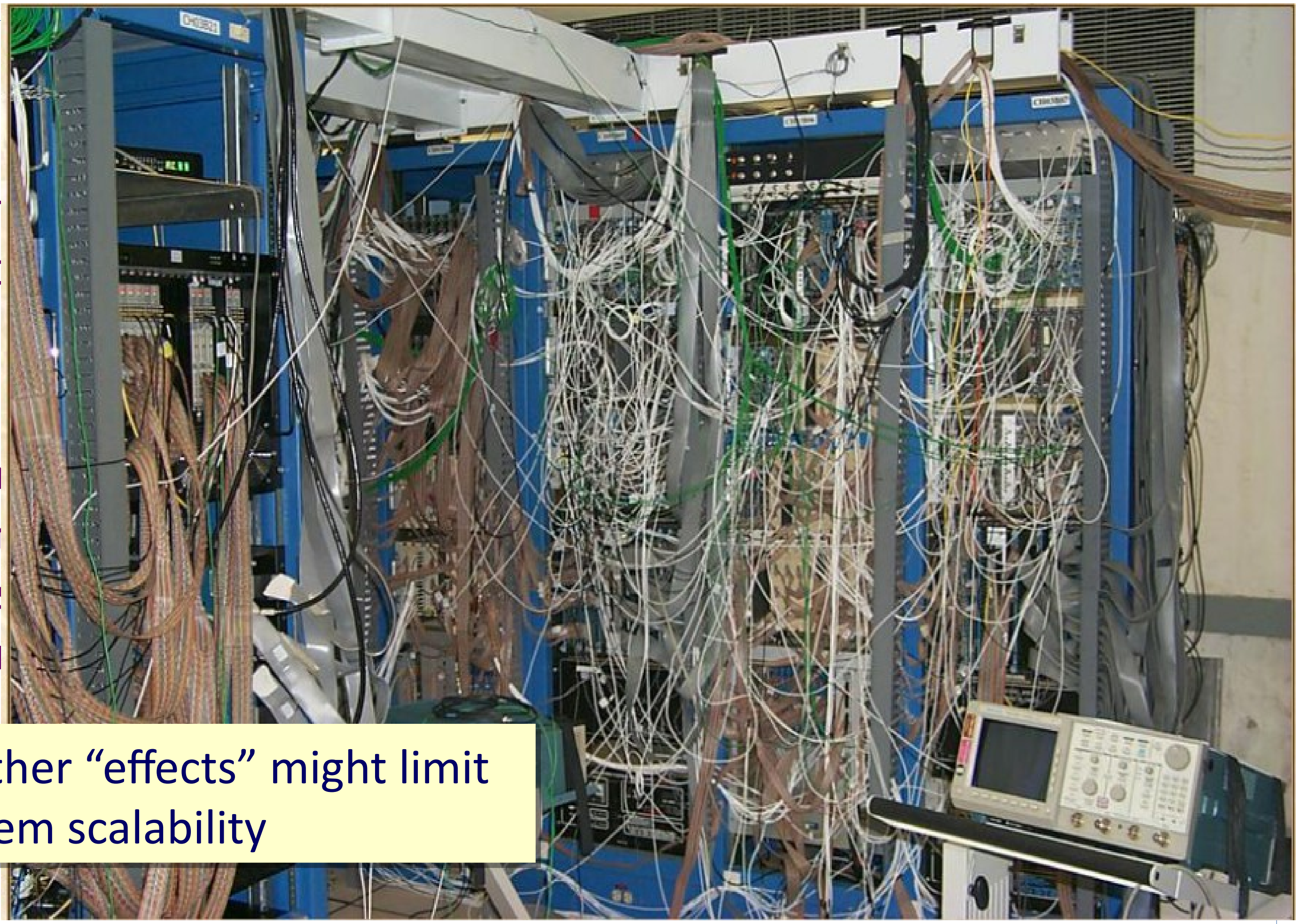


Scalability issues ✘
- bandwidth shared among all devices
- limited maximum bus width
- maximum bus frequency inversely proportional to bus length
- maximum number of devices depends on bus length

Bus Facts

Simple ✔
- fixed number
- devices have
  - mechanic

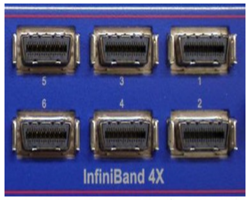
Scalability issues ✘
- bandwidth sh
- limited maxim
- maximum bu
- maximum nu

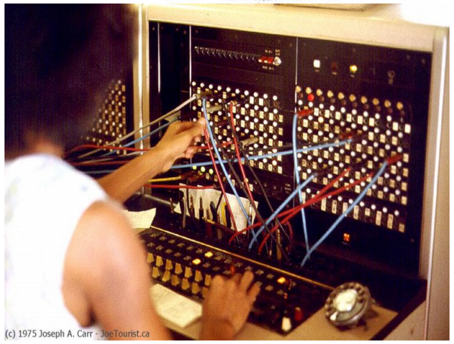


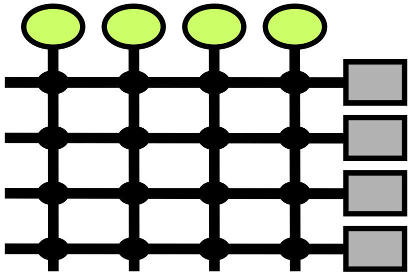
On the long term, other “effects” might limit your system scalability

Network

- Examples: Ethernet, Telephone, Infiniband, …
- All devices are equal
- Devices communicate directly with each other
  - no arbitration, simultaneous communications
- Device communicate by sending messages
- In switched network, switches move messages between sources and destinations
  - find the right path
  - handle "congestion" (two messages with the same destination at the same time)
    - would you be surprised to hear that buffering is the key?



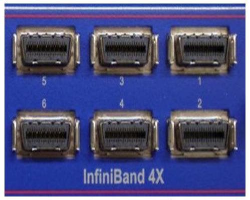


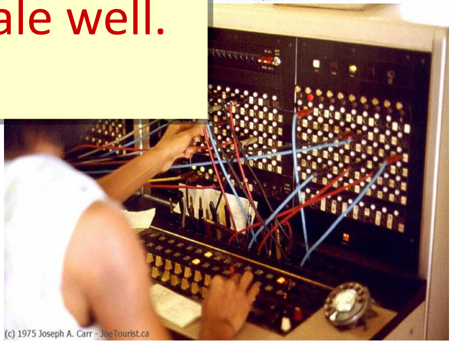


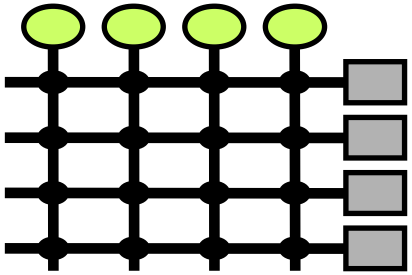
Network

- Examples: Ethernet, Telephone, Infiniband, …
- All devices are equal
- Devices communicate directly with each other
  - no arbitration, simultaneous communications
- Device communicate by sending messages
- In switched network, switches move messages between sourc
  - find the ri
  - handle "congestion" (two messages with the same destination at the same time)
    - would you be surprised to hear that buffering is the key?

Thanks to these characteristics, networks do scale well.
They are the backbones of LHC DAQ systems







Modular Electronics

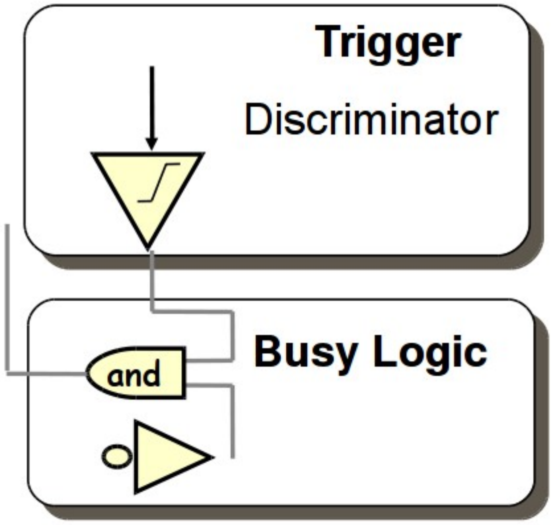
Modular Electronics

- Standard electronics "functions" implemented in well-defined "containers"
  - re-use of generic modules for different applications
  - limit the complexity of individual modules → reliability & maintainability
  - easy to upgrade to newer versions
  - profit from commercially available "functions"
- "Containers" are normally well-defined standards defining mechanical, electrical, … , interfaces
  - "easy" design and integrate your own module
- Historically, in HEP, modular electronics was bus-based
  - currently in a mixed phase …

Allow building your own data-acquisition system just connecting predefined functions → Fast & Efficient

NIM

- NIM (1964)
  - "Nuclear Instrumentation Modules"
  - 50 Ω input/output impedence
  - fast modules may have
    - rise/fall time: ~1 ns
    - duration: ~O(10 ns)
    - input/output delay: few ns
- NIM modules usually
  - do not need software, are not connected to PCs
  - implement logic and signal processing functions
    - discriminators, coincidences, amplifiers, Logic gates, …
  - may also provide HV channels
- Typically implement basic trigger and busy system



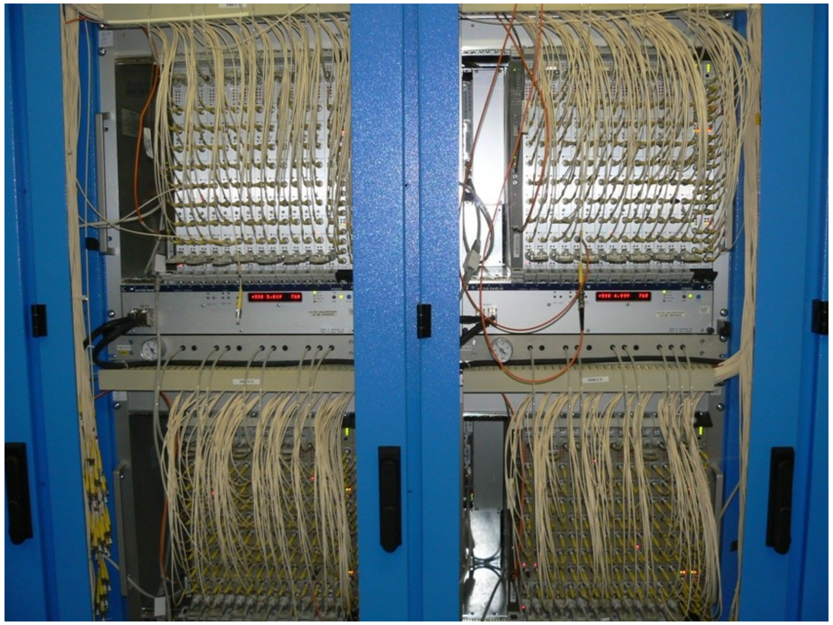
New modules still appear on market

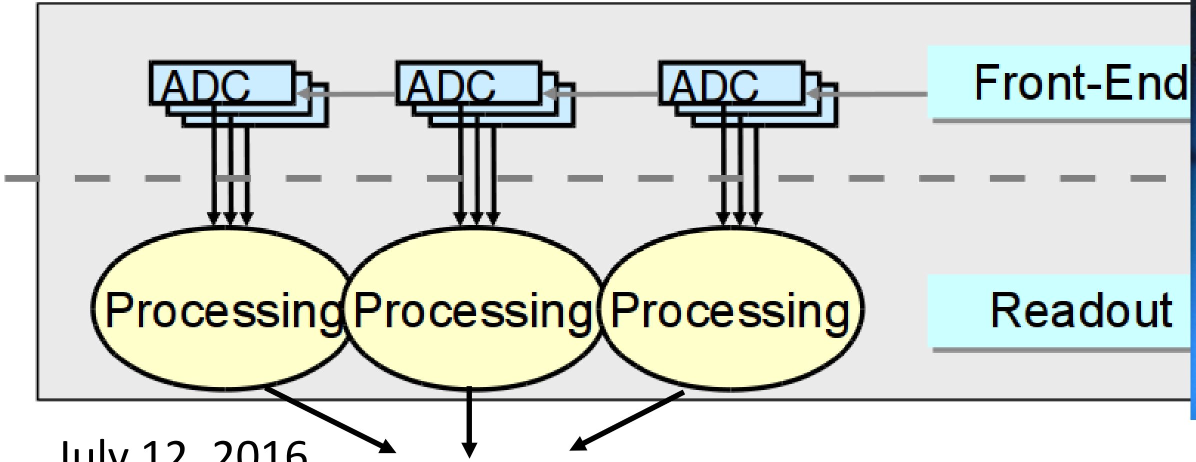
Very diffused in medium-size HEP experiments

Found in counting rooms of LHC experiments

VMEbus

- VMEbus: modules communicate via a "backplane"
  - electrical, mechanical and communication protocols
- Choice of many HEP experiments for off-detector electronics [power and control]
  - relatively simple protocol
  - lot of commercially available functions
- More than 1000 VMEbus crates at CERN





Other (arising) Standards

- PCI-based

- We know buses have limited scalability. Can we have "network-based" modular electronics?
- VXS → essentially VME plus switched interconnectivity
- ATCA and derivatives
  - standard designed for telecom companies
  - high-redundancy, data-throughput, high power density
  - **being used for LHC upgrade programs**

to be continued...