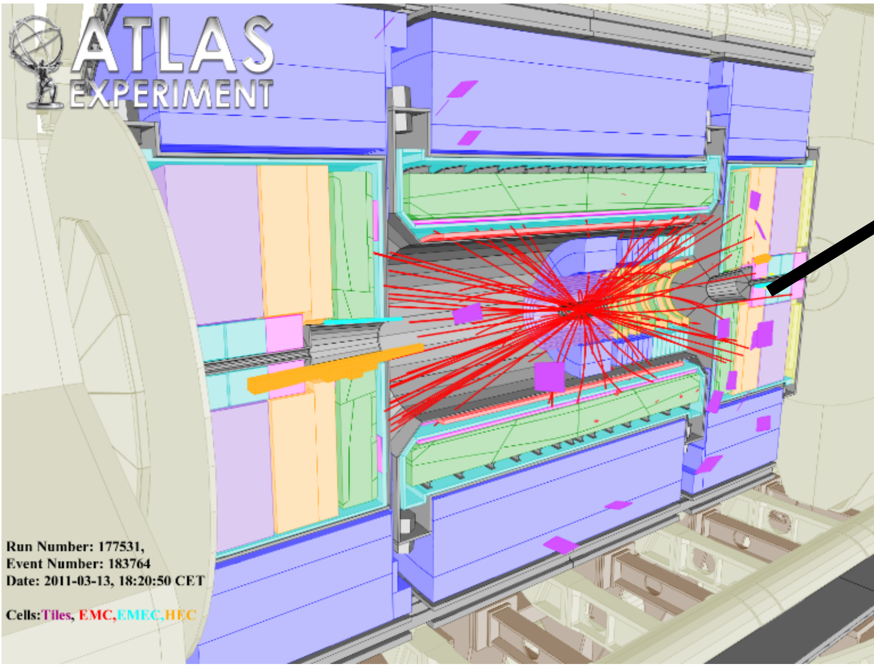
Trigger and Data Acquisition

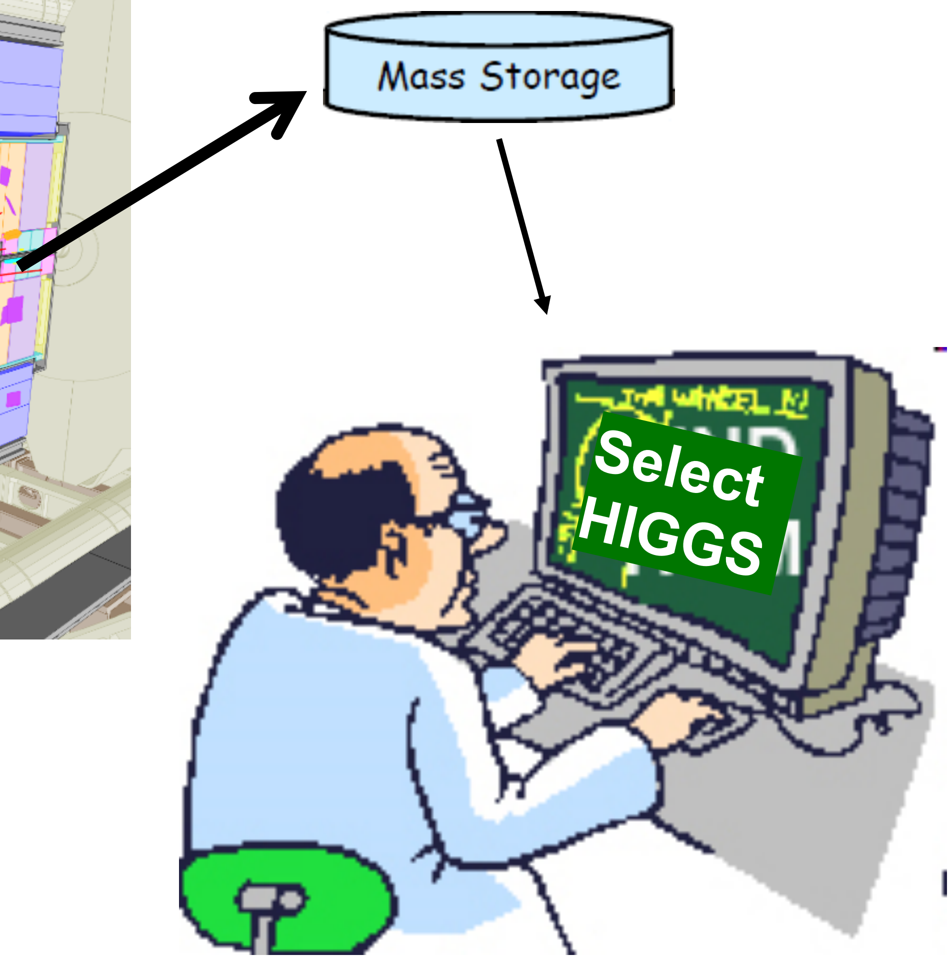
Monika Wielers
Rutherford Appleton Laboratory

- Lecture 1
  - Introduction to data acquisition (DAQ) systems
- Lecture 2
  - DAQ and trigger systems at the LHC
- Lecture 3
  - Trigger selections

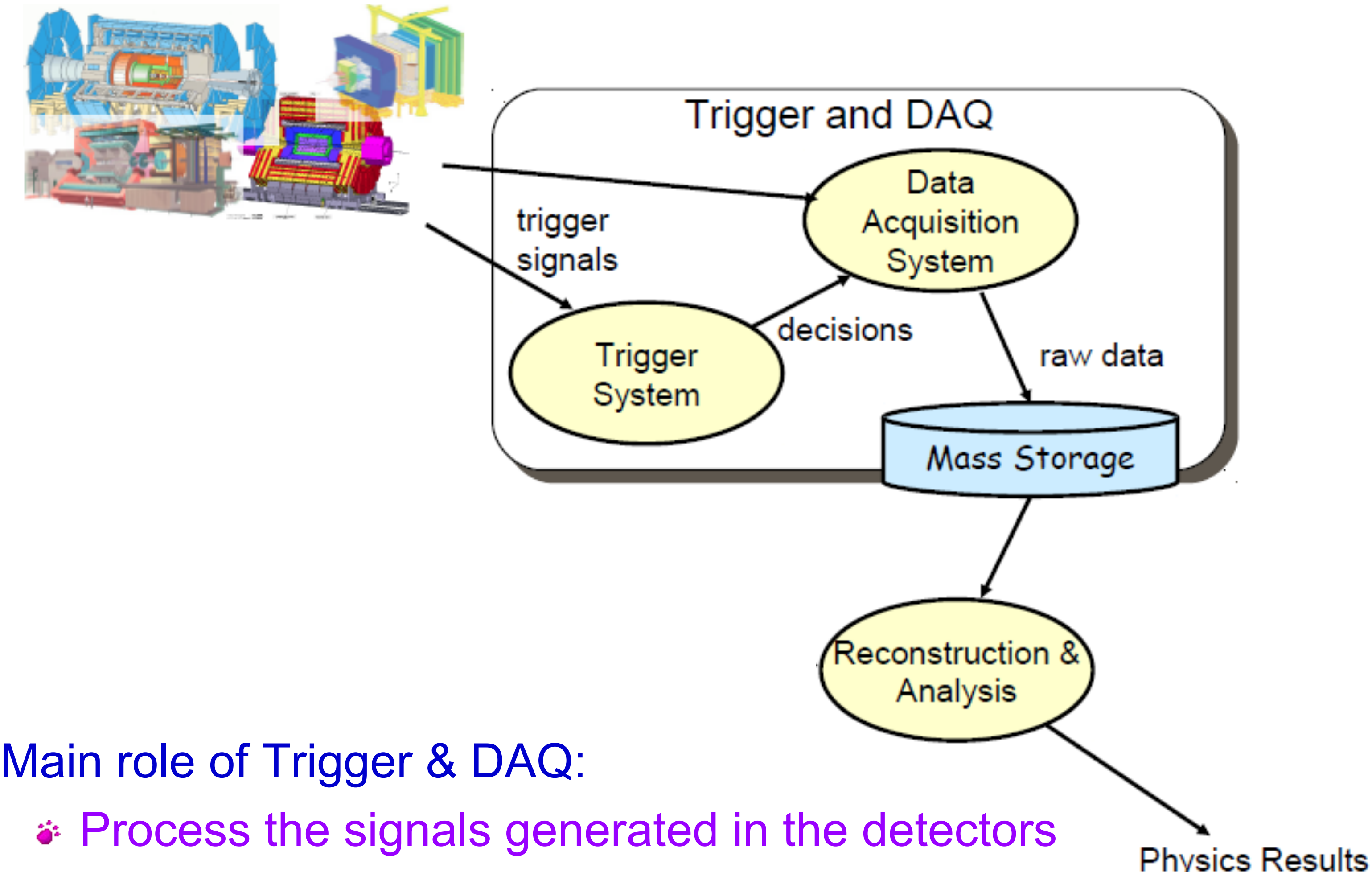
What is it about...

- How to get from … to





Or



- Main role of Trigger & DAQ:
  - Process the signals generated in the detectors
  - Select the ‘interesting’ events and reject the ‘boring’ ones
  - Save interesting ones on mass storage for physics analysis
- Heartbeat of the experiment!

DAQ

- Abbreviation for Data Acquisition System
- Wikipedia:
  - Process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer.
- In HEP it consists mainly of electronics, computer science, networking and quite some physics
- Tasks
  - Gathers the data produced by the detectors (Readout)
  - Forms complete events (Event Building)
  - Possibly feeds (several) levels of deciding to keep the collision (called typically event in the following)
  - Stores event data (Data logging)
  - Provides control, configuration and monitoring facilities

Trigger

- That's one

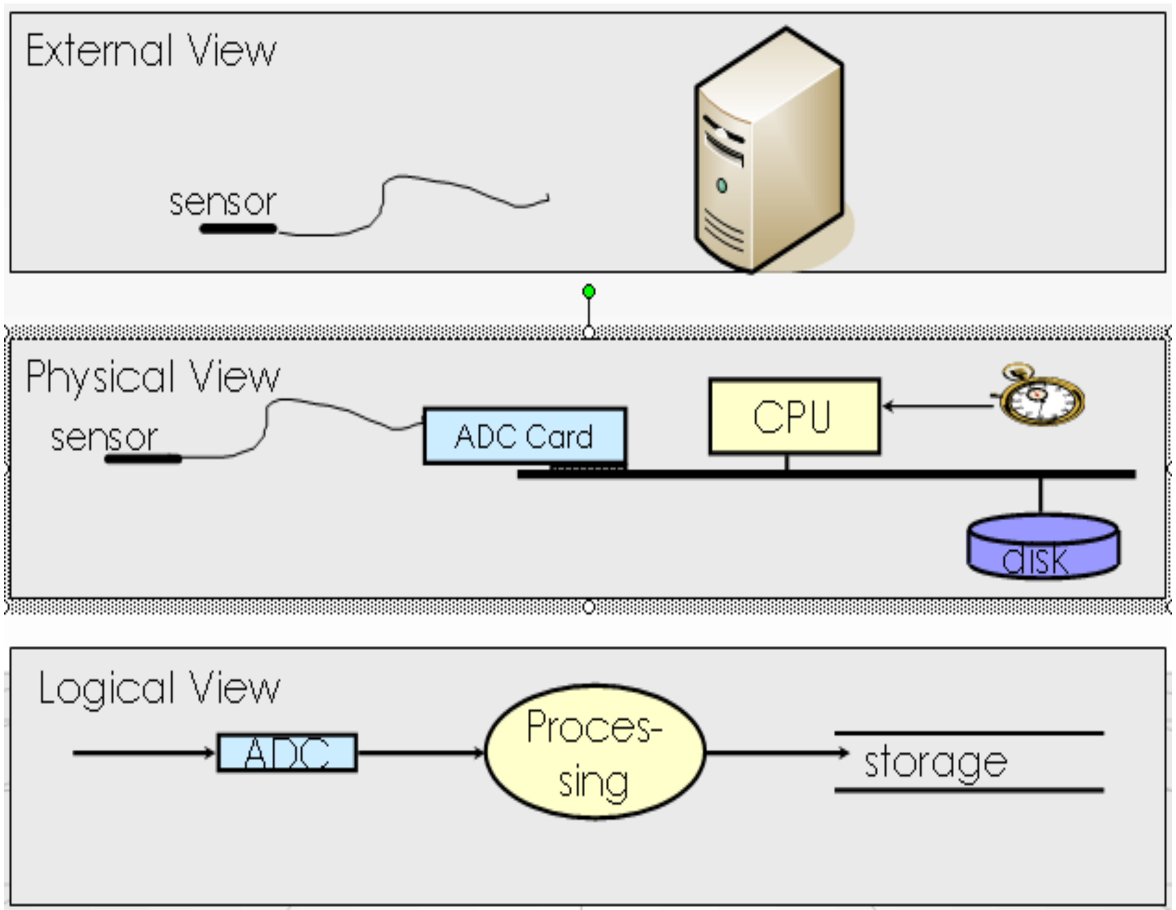
- But that's not what we want to discuss here
- Trigger = in general something which tells you when is the "right" moment to take your data
- Trigger = process to very rapidly decide if you want to keep the data if you can't keep all of them. The decision is based on some 'simple' criteria
- This can happen in several stages, if needed

- Note, DAQ and Trigger often are not two separate issues, but are 'interwoven'

Goals of this lecture

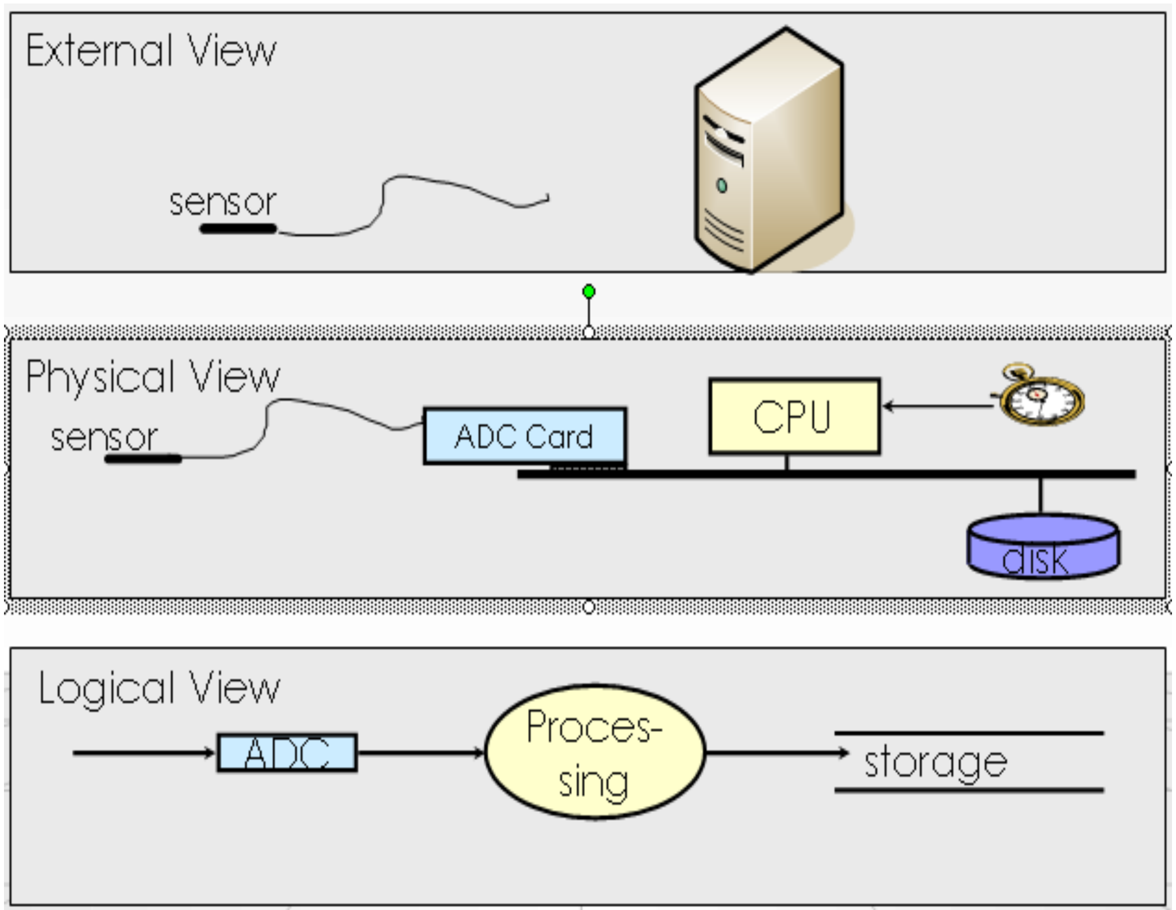
- Understand how data acquisition is devised
  - Start with simple example and then get more complex
- Introduce the terms you will hear when you hear about data acquisition in a HEP experiment
- All this might be a bit technical but might help you later during your Ph.D. and it is actually also quite some fun!

Trivial DAQ (periodic trigger)



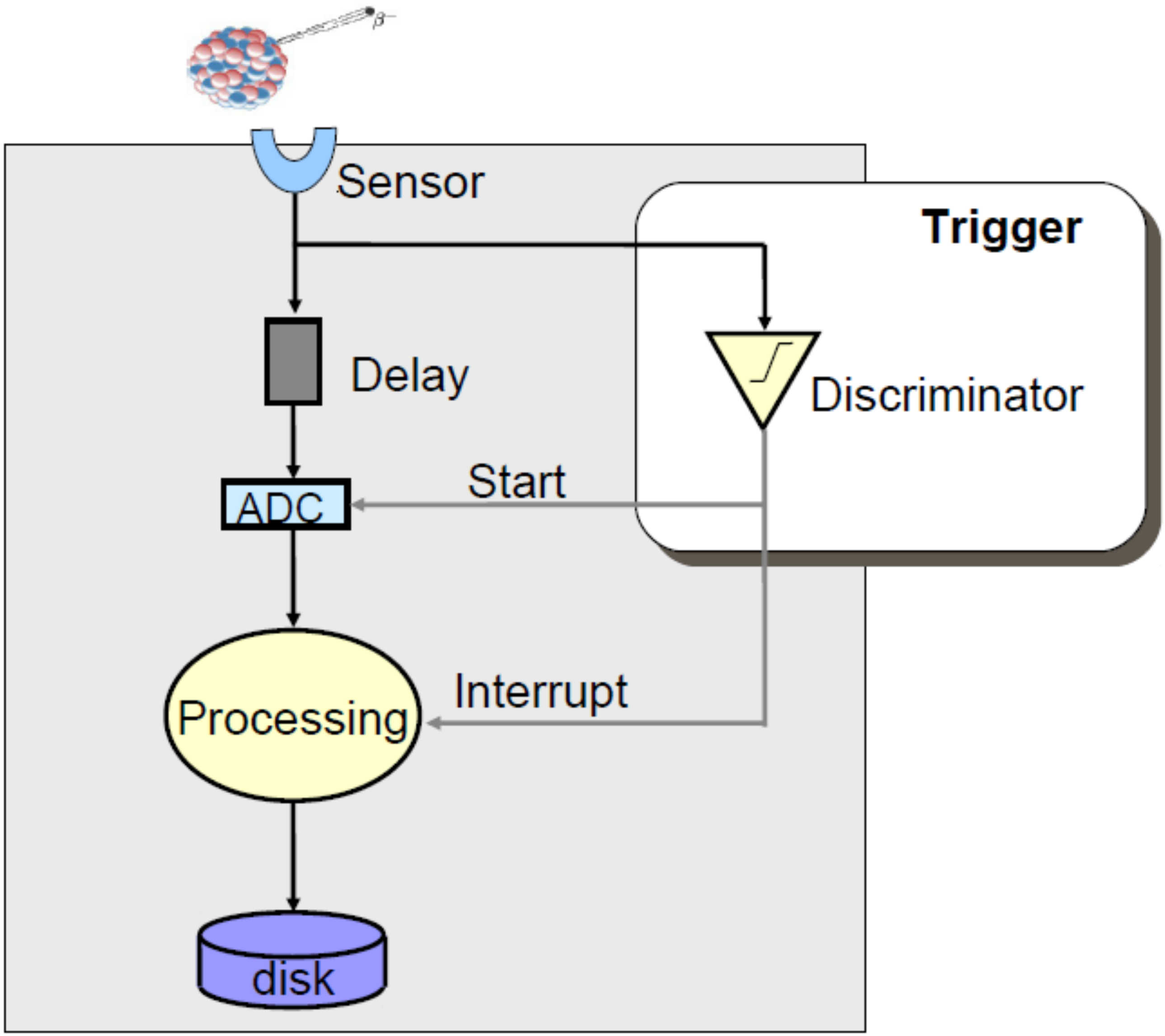
- Measure temperature at a fixed frequency
- ADC performs analog to digital conversion (digitisation)
  - Our frontend electronics
- CPU does readout and processing

Trivial DAQ (periodic trigger)



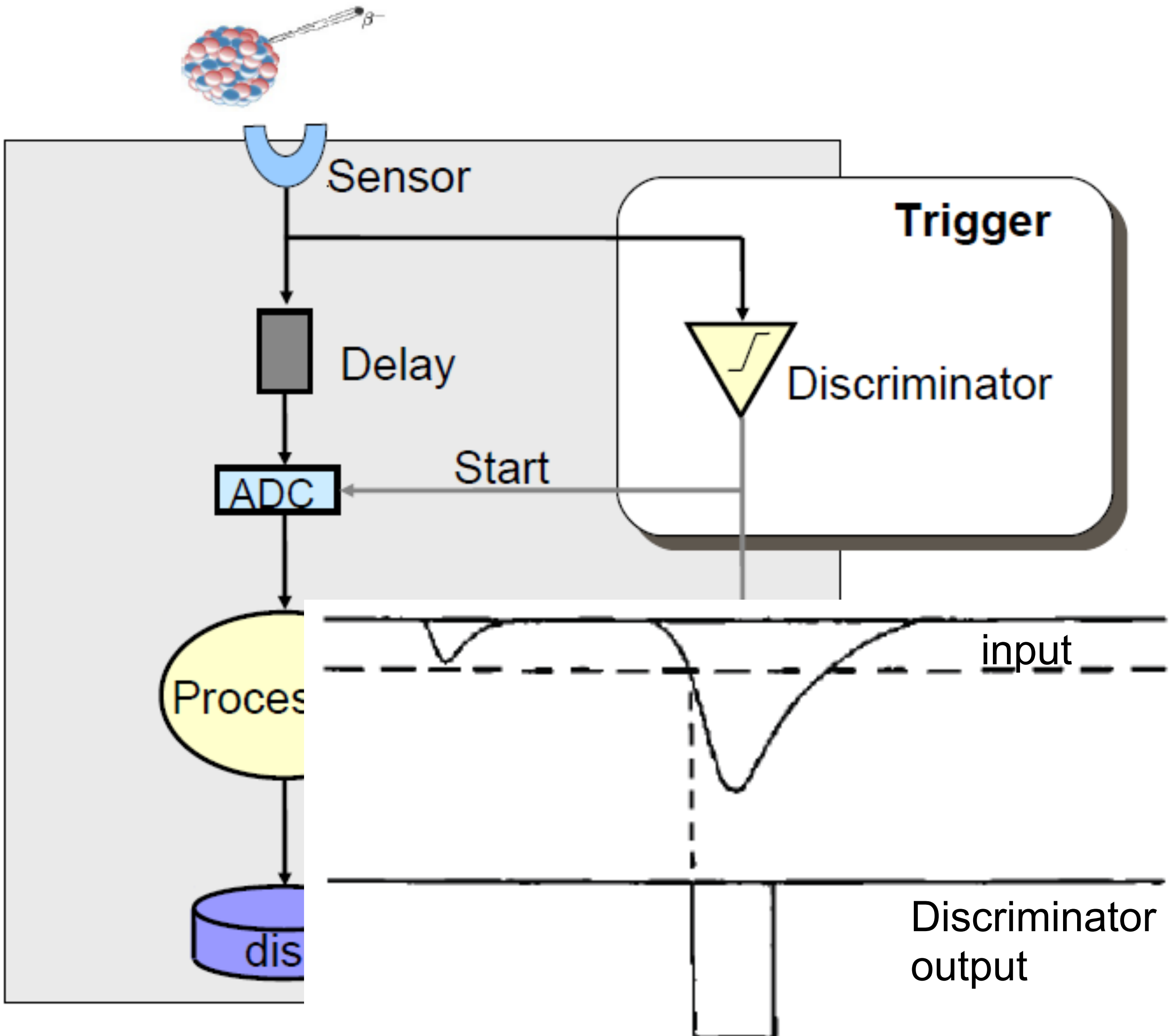
- Measure temperature at a fixed frequency
- The system is clearly limited by the time to process a measurement (or event)
- Example τ=1ms to
  - ADC conversion +CPU processing +Storage
- Sustain maximal ~1/1ms=1kHz ***periodic trigger*** rate

Trivial DAQ with a trigger



- Example: measure β decay properties
- Our events are asynchronous and unpredictable
  - Need a physics trigger
- Delay compensates for the trigger latency

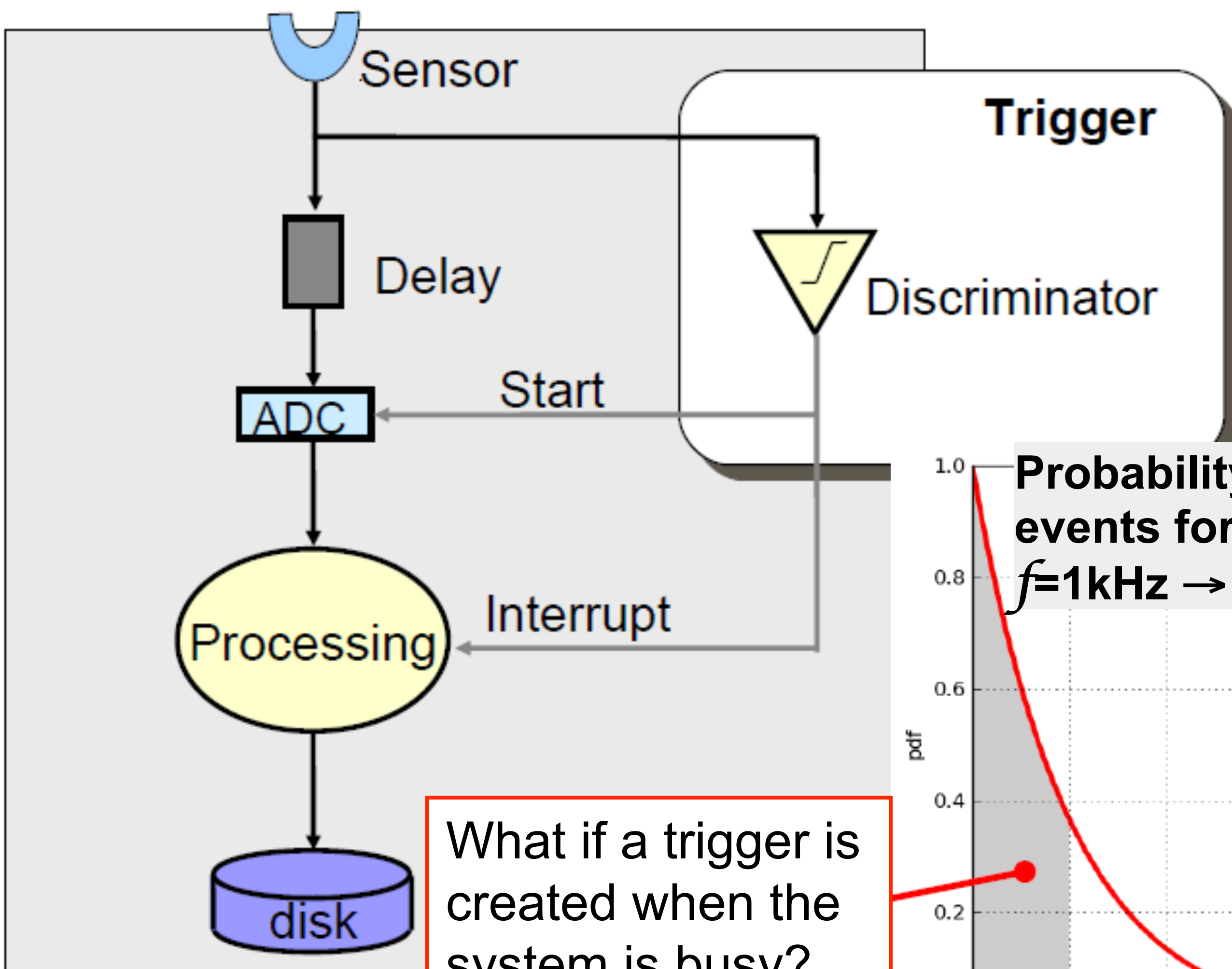
Trivial DAQ with a trigger



- Example: measure β decay properties
- Our events are asynchronous and unpredictable
  - Need a physics trigger
- Discriminator: generate an output signal only if amplitude of input pulse is greater than a certain threshold

Trivial DAQ with a trigger

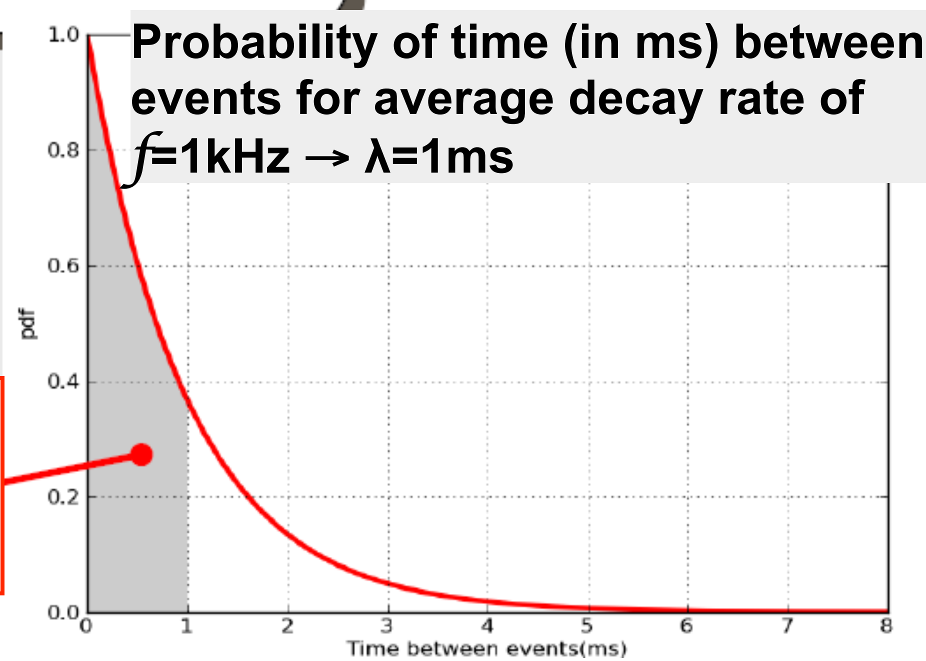
f=1kHz
$\lambda=1/f$ = 1ms



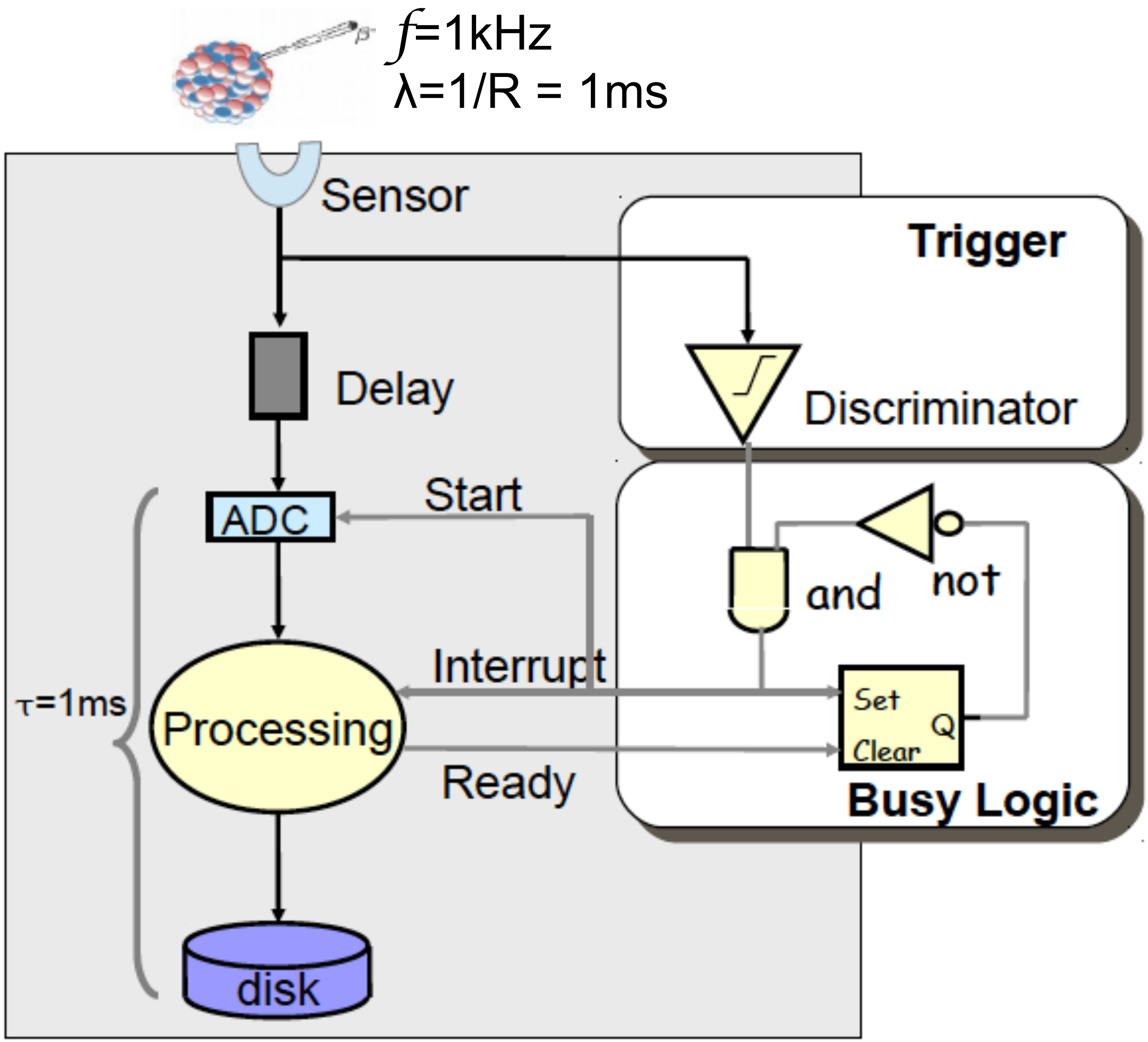
What if a trigger is created when the system is busy?

- Example: measure β decay properties
  - Stochastic process
  - Need a physics trigger

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



Trivial DAQ with a trigger



- Busy logic avoids triggers while processing
- Which (average) DAQ rate can we achieve now?
- Reminder: τ=1ms was sufficient to run at 1kHz with a clock trigger

Deadtime and Efficiency

- Definitions
  - Average rate of physics phenomenon (input): f
  - Process rate: λ=1/f
  - Average rates of DAQ (output): ν
  - Deadtime: τ
    - Time the system requires to process an event, without being able to handle other triggers
  - Probability that DAQ is busy: P[busy] = ν τ
  - Probability that DAQ is free: P[free] = 1 - ν τ
- Therefore
  - $\nu = f \cdot P[\text{free}] \;\Rightarrow\; \nu = f(1 - \nu \cdot \tau) \;\Rightarrow\; \nu = \frac{f}{1 + f \cdot \tau} < f$
  - Efficiency: $\varepsilon = \frac{N_{saved}}{N_{tot}} = \frac{1}{1 + f \cdot \tau} < 100\%$

Deadtime and Efficiency

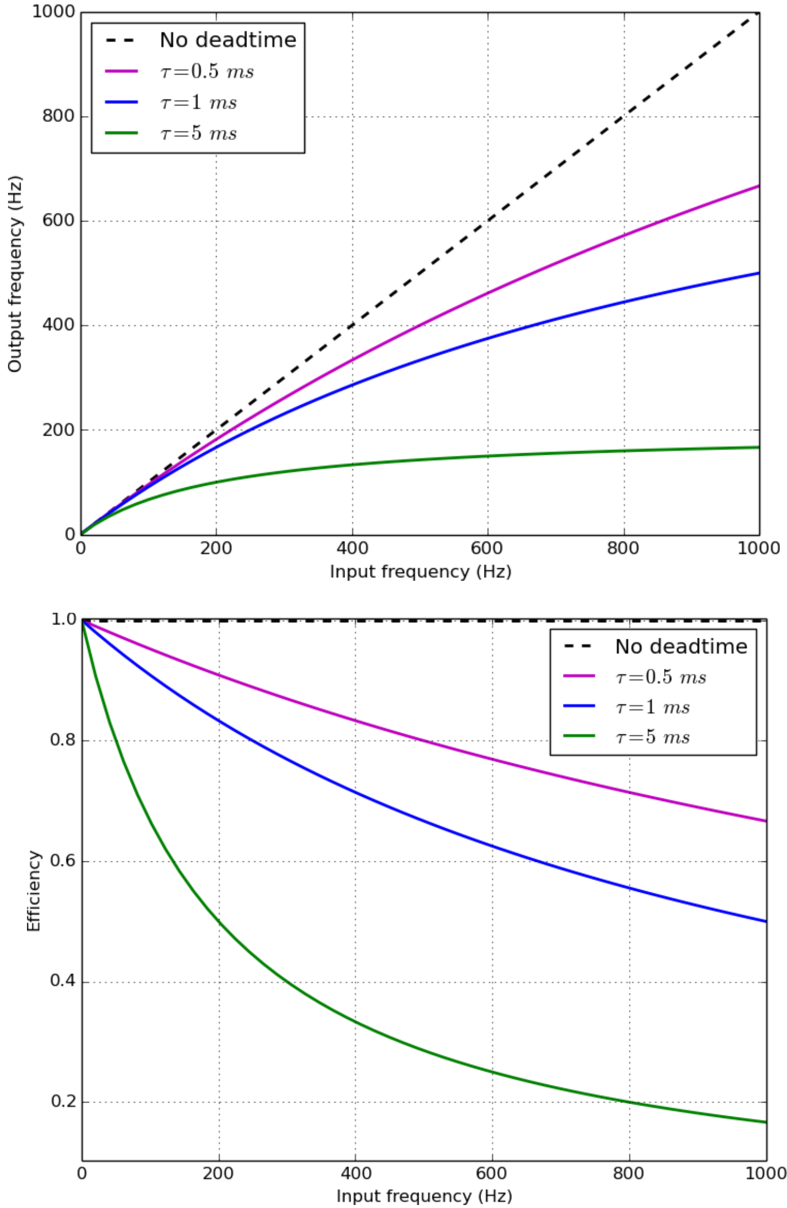
- Due to stochastic fluctuations
  - DAQ rate < physics rate $\nu = \frac{f}{1+f\tau} < f$
  - Efficiency always < 100% $\varepsilon = \frac{1}{1+f\tau} < 1$
- In our example: f=1kHz, τ=1ms
  - ν = 1kHz / (1 + 1kHz · 1ms) = 500Hz
  - ε = 1 / (1 + 1kHz · 1ms) = 50%

Deadtime and Efficiency

- Due to stochastic fluctuations
  - DAQ rate < physics rate $\nu = \frac{f}{1+f\tau} < f$
  - Efficiency always < 100% $\varepsilon = \frac{1}{1+f\tau} < 1$
- In our example: f=1kHz, τ=1ms
  - ν = 1kHz / (1 + 1kHz · 1ms) = 500Hz
  - ε = 1 / (1 + 1kHz · 1ms) = 50%
- To have higher efficiency → $f \cdot \tau \ll 1$
- e.g. f=1kHz, ε>99%

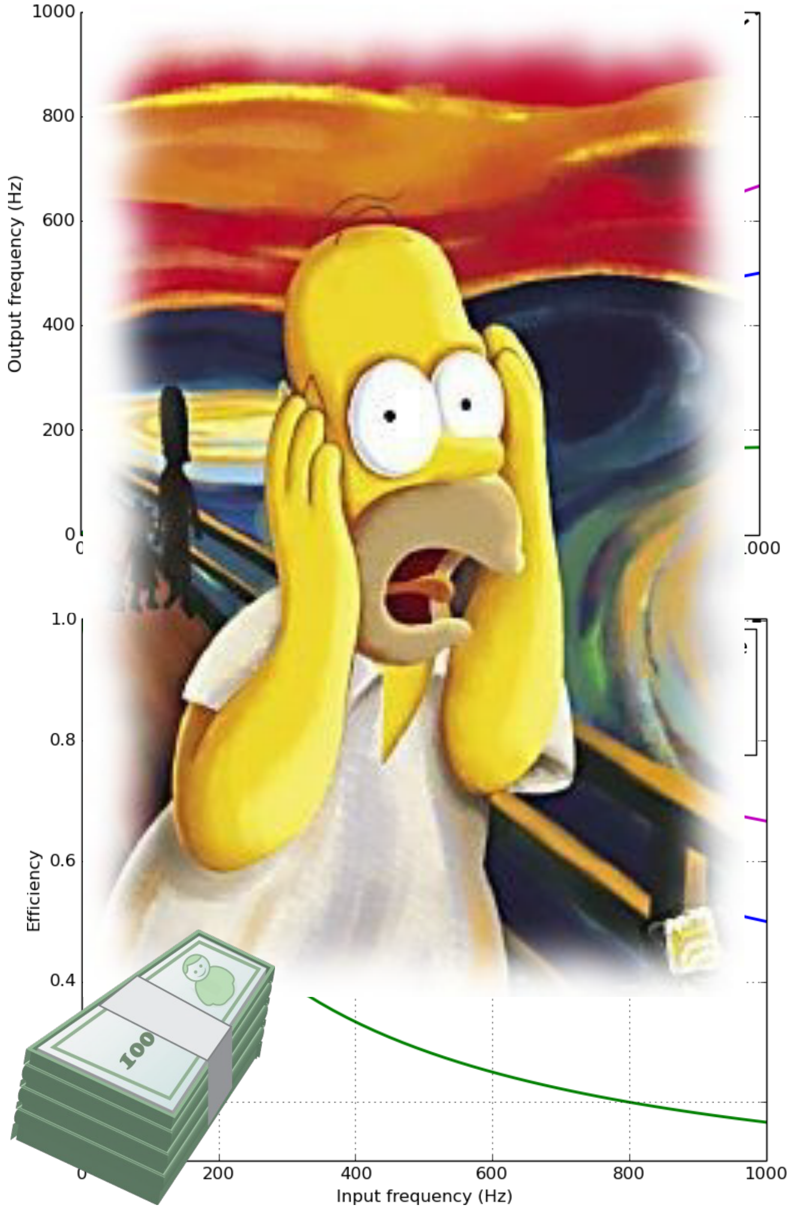
  → $\tau = 1/f\,(1/\varepsilon - 1)$ = 0.1 ms

  → $1/\tau$>10kHz

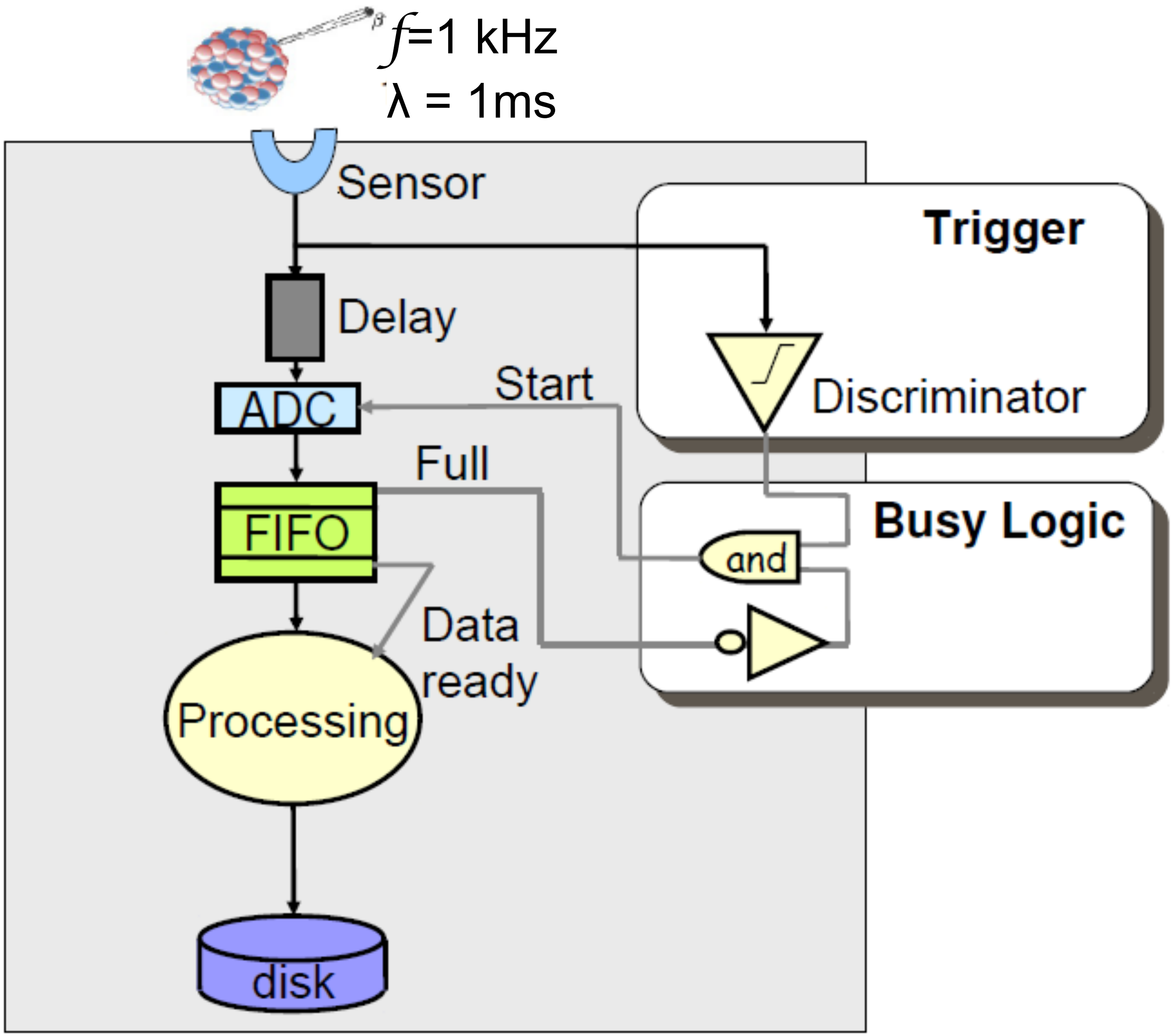


Deadtime and Efficiency

- Due to stochastic fluctuations
  - DAQ rate < physics rate $\nu = \dfrac{f}{1+f\tau} < f$
  - Efficiency always < 100% $\varepsilon = \dfrac{1}{1+f\tau} < 1$
- In our example: f=1kHz, τ=1ms
  - ν = 1kHz / (1 + 1kHz · 1ms) = 500Hz
  - ε = 1 / (1 + 1kHz · 1ms) = 50%
- To have higher efficiency → $f \cdot \tau$<<1
- e.g. f=1kHz, ε>99%
  - → τ = 1/f (1/ε - 1) = 0.1 ms
  - → 1/τ>10kHz
- In order to cope with the input signal fluctuations, we would need to overdesign our DAQ system by a factor 10. hmmmm…

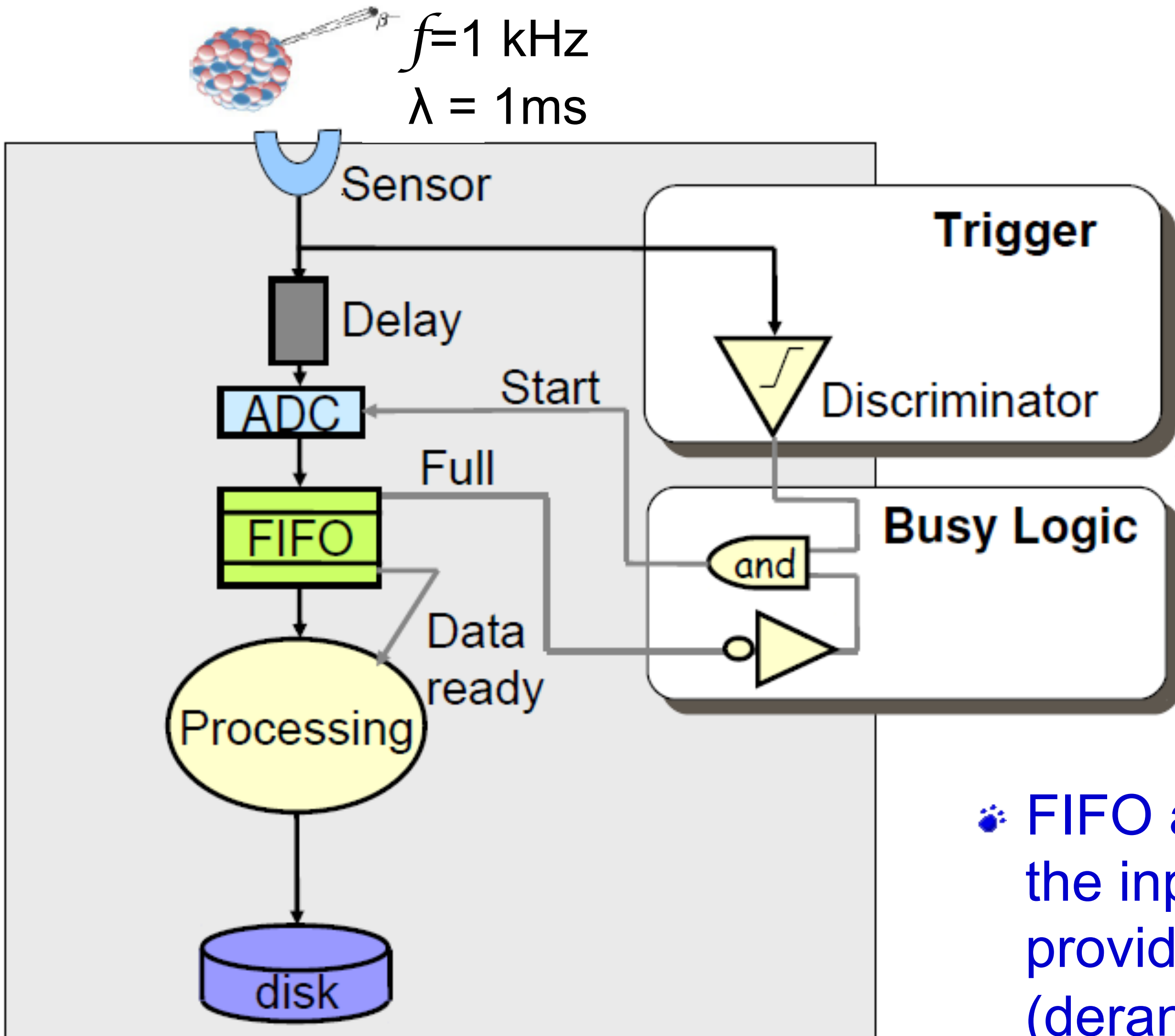


Trivial DAQ with Derandomisation

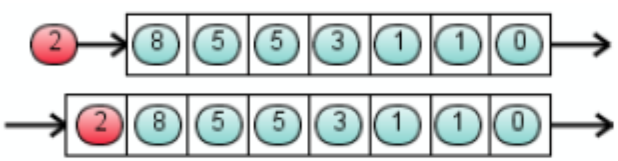


- Buffers are introduced which hold temporarily the data.
- They decouple the data production from the data processing
  ➔Better performance

Trivial DAQ with Derandomisation

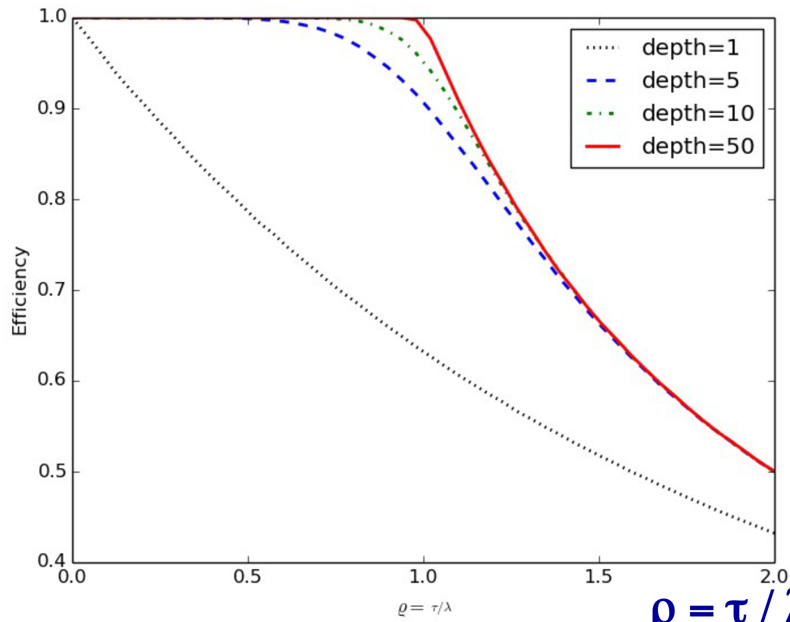


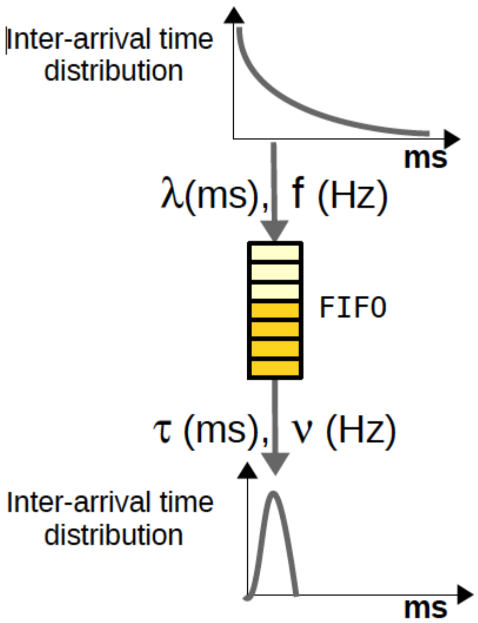
- First In First Out
  - Buffer area organized as a queue
  - Depth: number of cells
  - Implemented in HW and SW



- FIFO absorbs and smoothes the input fluctuations, providing a ~steady (derandomized) output rate
- introduces an additional latency on the data path

De-randomization: queuing theory

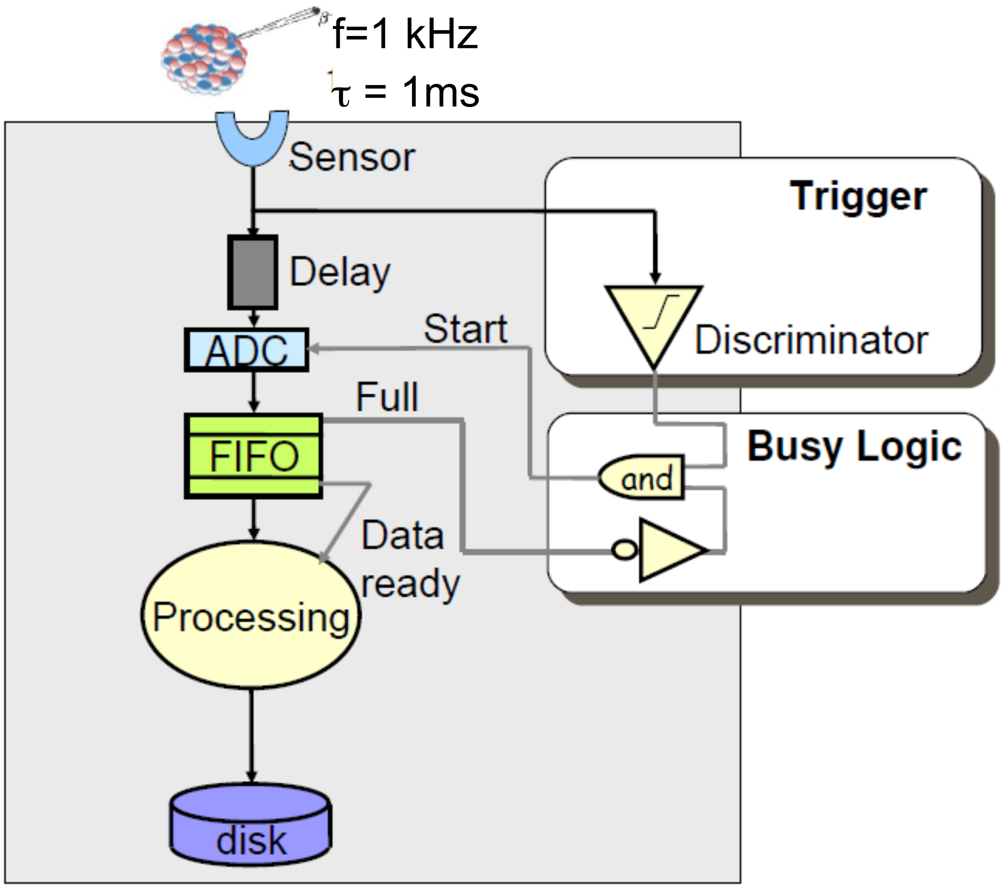




$\rho = \tau / \lambda$

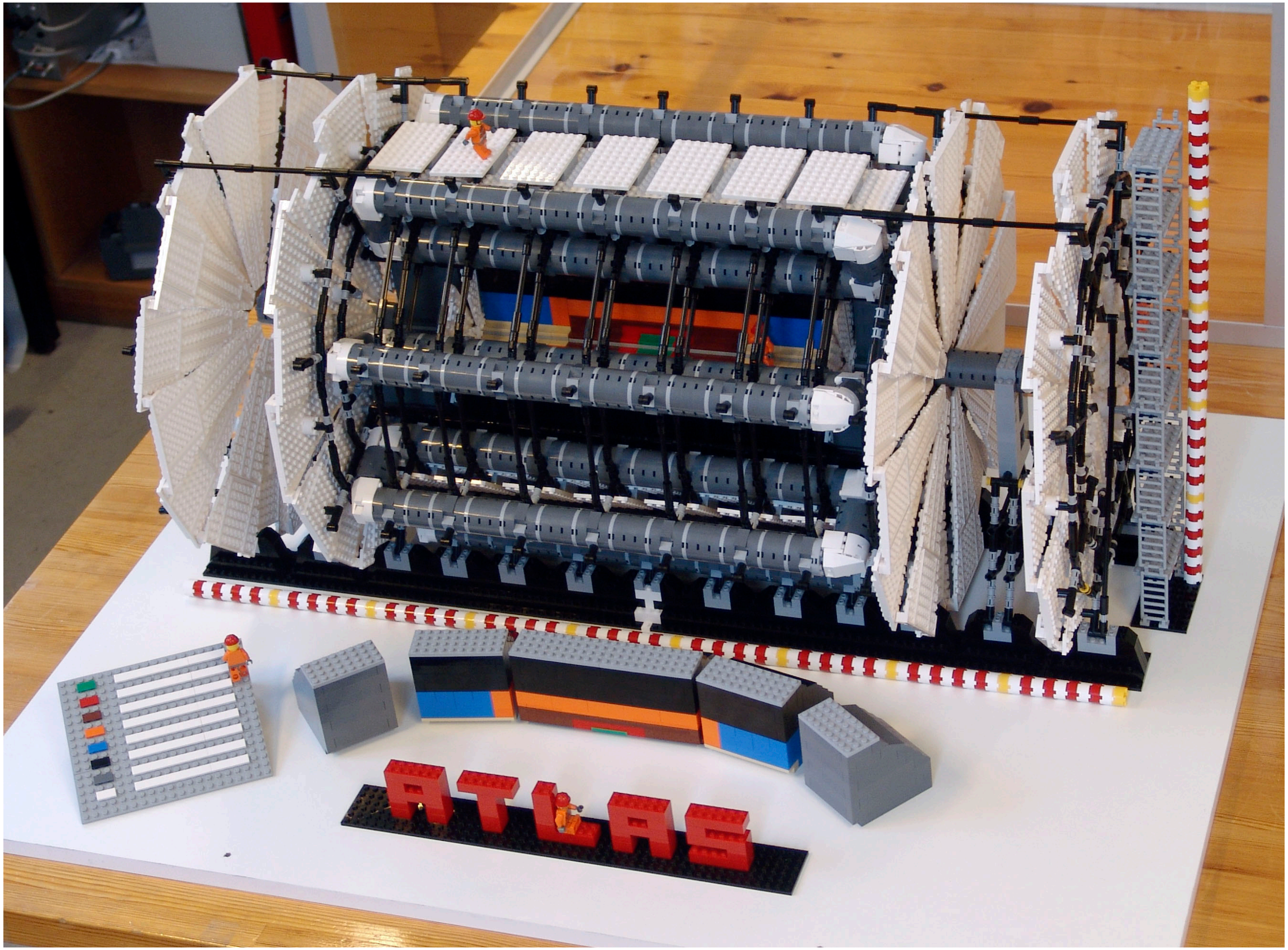
- Efficiency vs traffic intensity ($\rho = \tau / \lambda$) for different queue depths
  - $\rho > 1$, the system is overloaded
  - $\rho << 1$, the output is over-designed
  - $\rho \sim 1$, using a queue, high efficiency can be obtained with moderate depth
- Analytic calculation possible for very simple systems only
  - Otherwise Monte Carlo simulation is required

Trivial DAQ with Derandomisation

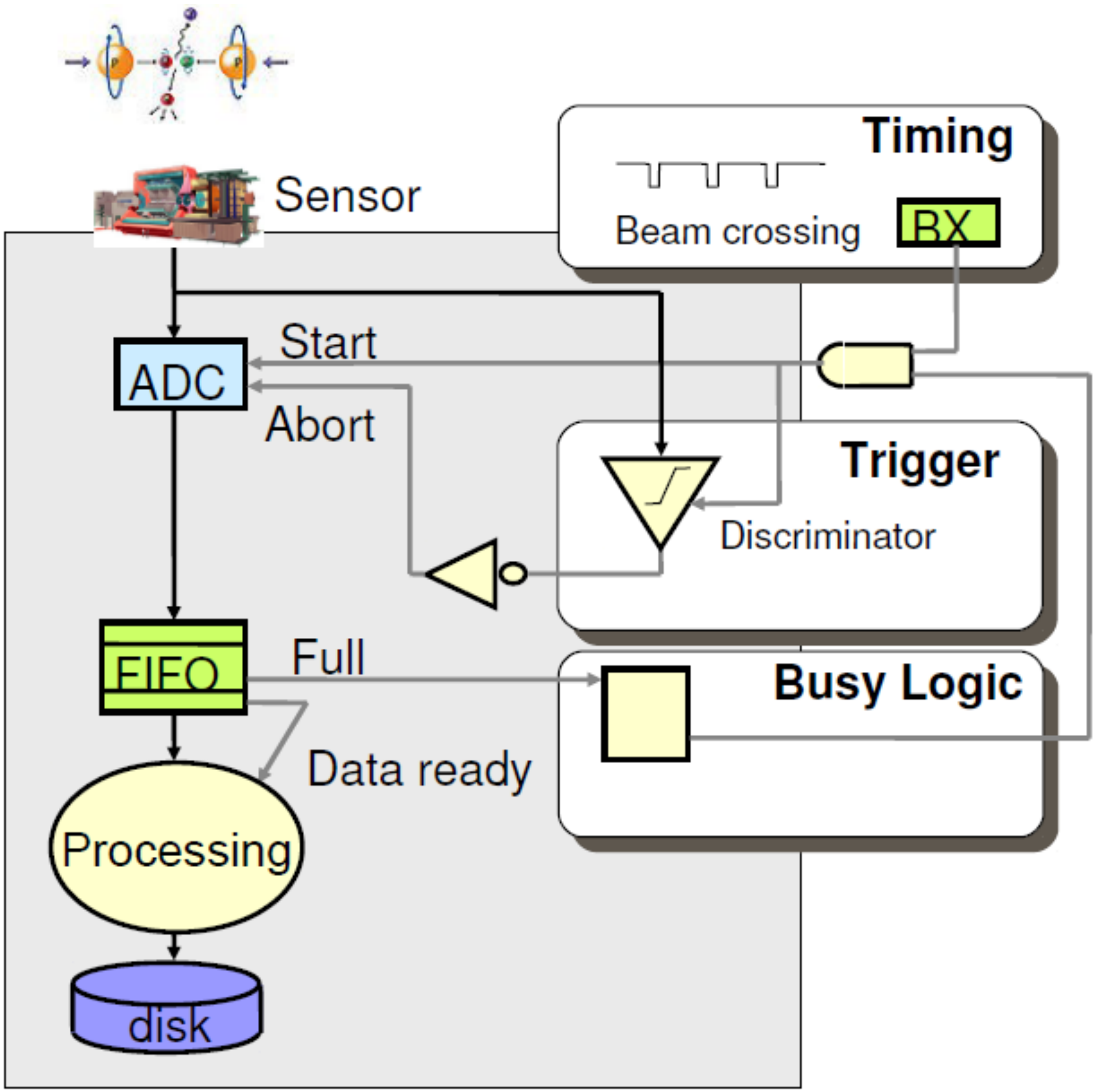


- Almost 100% efficiency and minimal deadtime if
  - ADC is able to operate at rate >>f
  - Data processing and storing operates at ~f
- Minimises the amount of “unnecessary” fast components
- Could the delay be replaced with a “FIFO”?
  - Analog pipelines → Heavily used in HEP DAQs

Let's have a closer look at DAQ at a collider

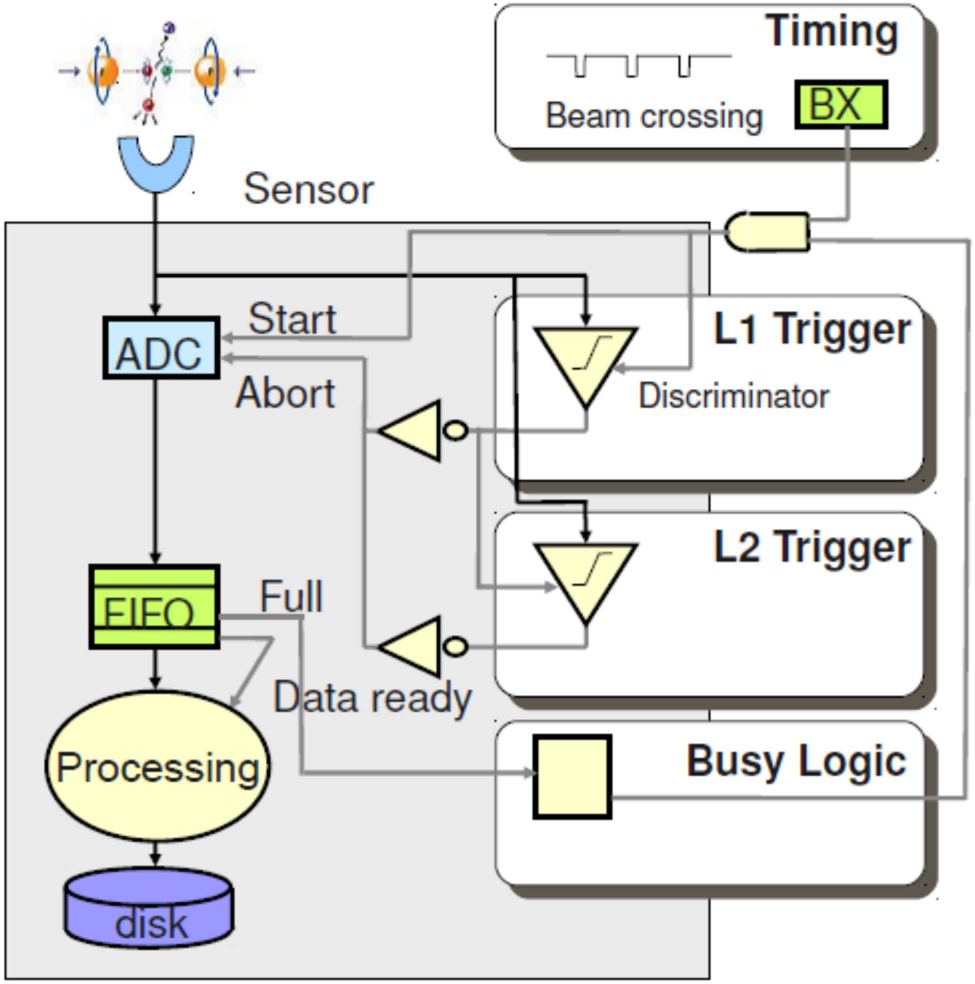


DAQ: Collider mode



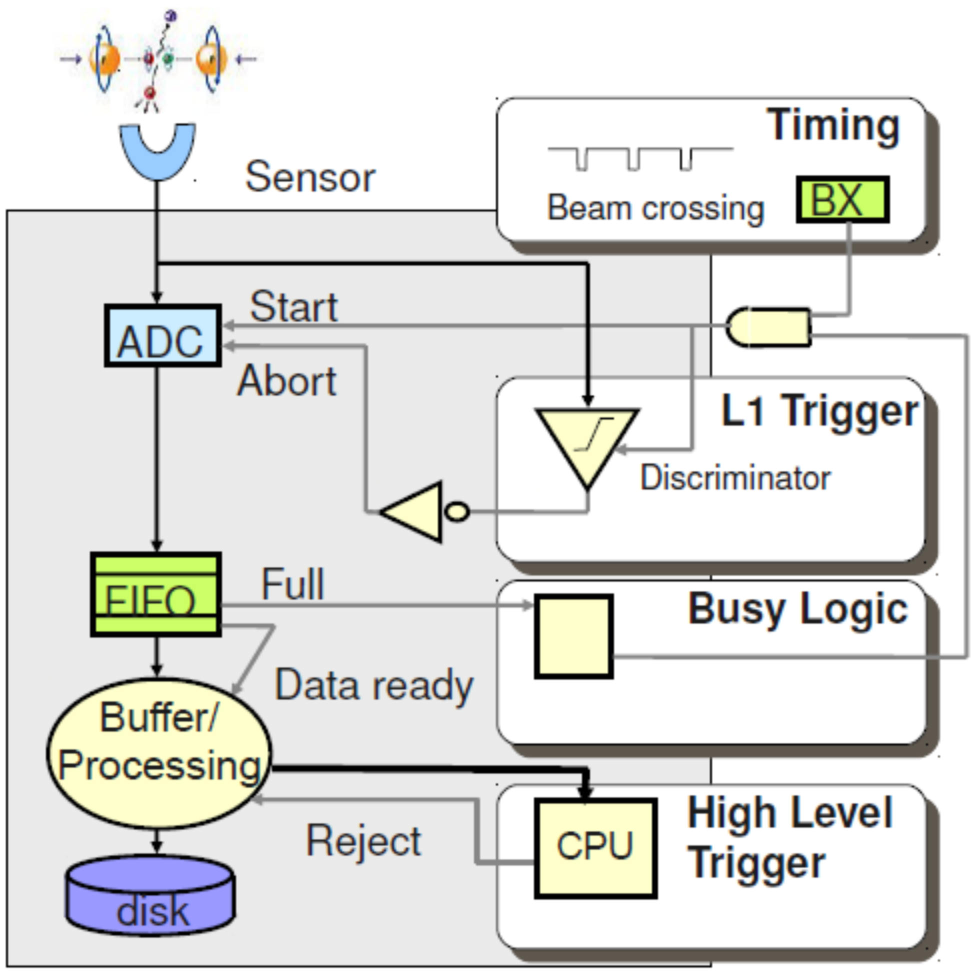
- Particle collisions are synchronous
- Trigger rejects uninteresting events
- Even if collisions are synchronous, the triggers (interesting events) are unpredictable
- Derandomisation is still needed
- No trigger deadtime if trigger latency below time between two beam crossings

Multi-Level Trigger



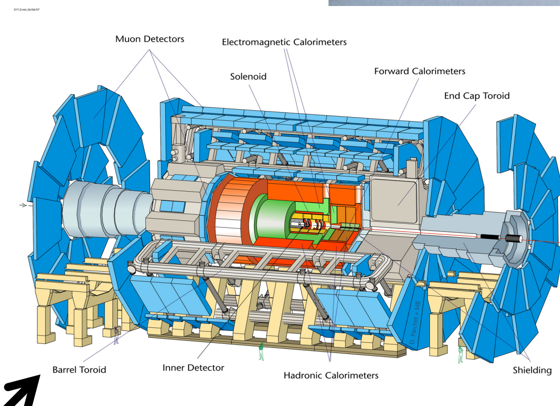
- For complicated triggers latency can be long
  - if $\tau_{trig} > \tau_{BX}$, deadtime>50%
- Split trigger in several levels with increasing complexity and latency
- All levels can reject events
  - with $\tau_{L1} < \tau_{BX}$, trigger deadtime only $\nu_{L1} \cdot \tau_{L2}$

Multi-Level Trigger



- For optimal data reduction can add trigger level between readout and storage (High-level trigger)
- Has access to some/ all processed data

Scaling up

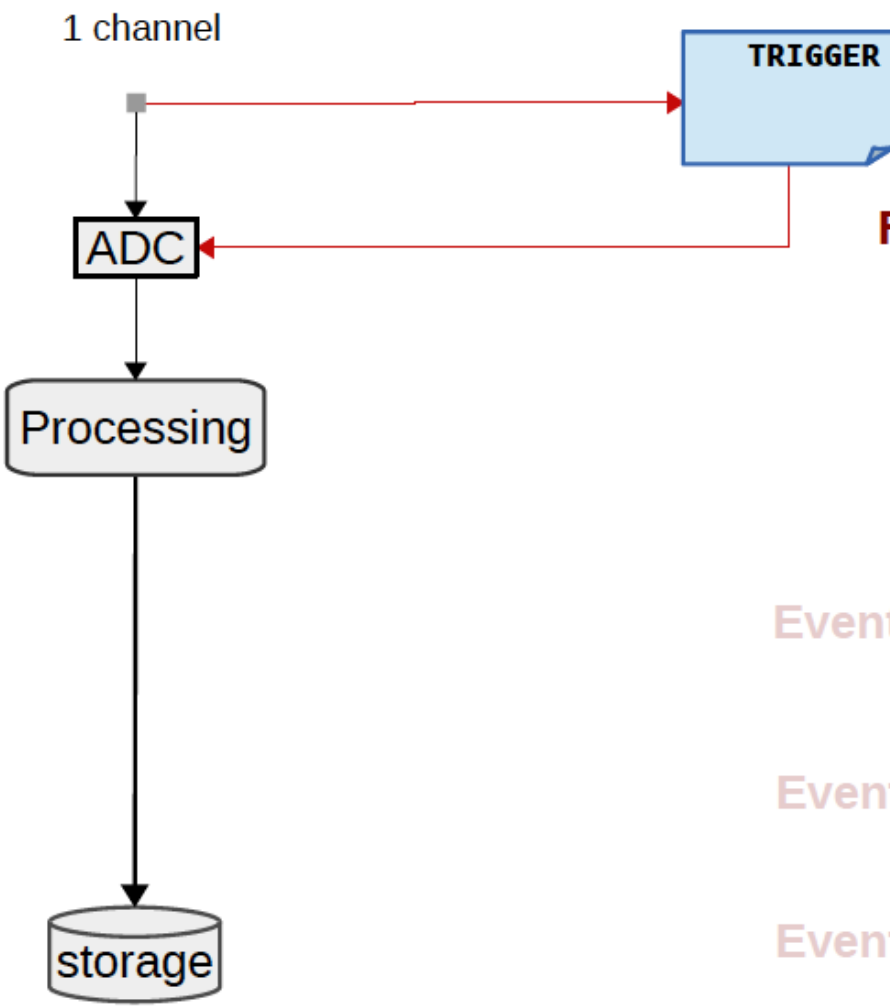


Muon Detectors
Electromagnetic Calorimeters
Solenoid
Forward Calorimeters
End Cap Toroid
Barrel Toroid
Inner Detector
Hadronic Calorimeters
Shielding

sensor

A bit more complicated….

- The increased number of channels require hierarchical structure with well defined interfaces between components



Front-End | data digitization, data buffering

Readout | data extraction, data formatting, data buffering

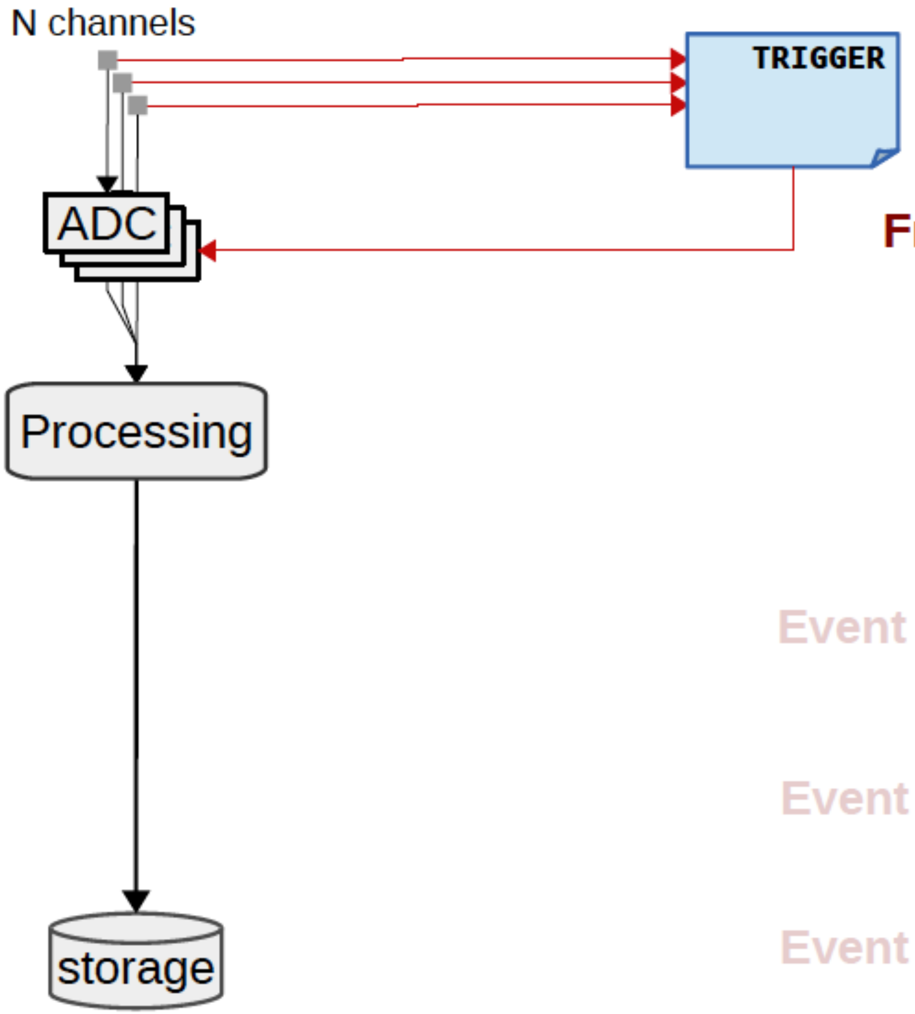
Event Building | event assembly, event buffering

Event Filtering | event rejection, event buffering

Event Logging | file storage, file buffering

A bit more complicated….

- The increased number of channels require hierarchical structure with well defined interfaces between components



Front-End | data digitization, data buffering

Readout | data extraction, data formatting, data buffering

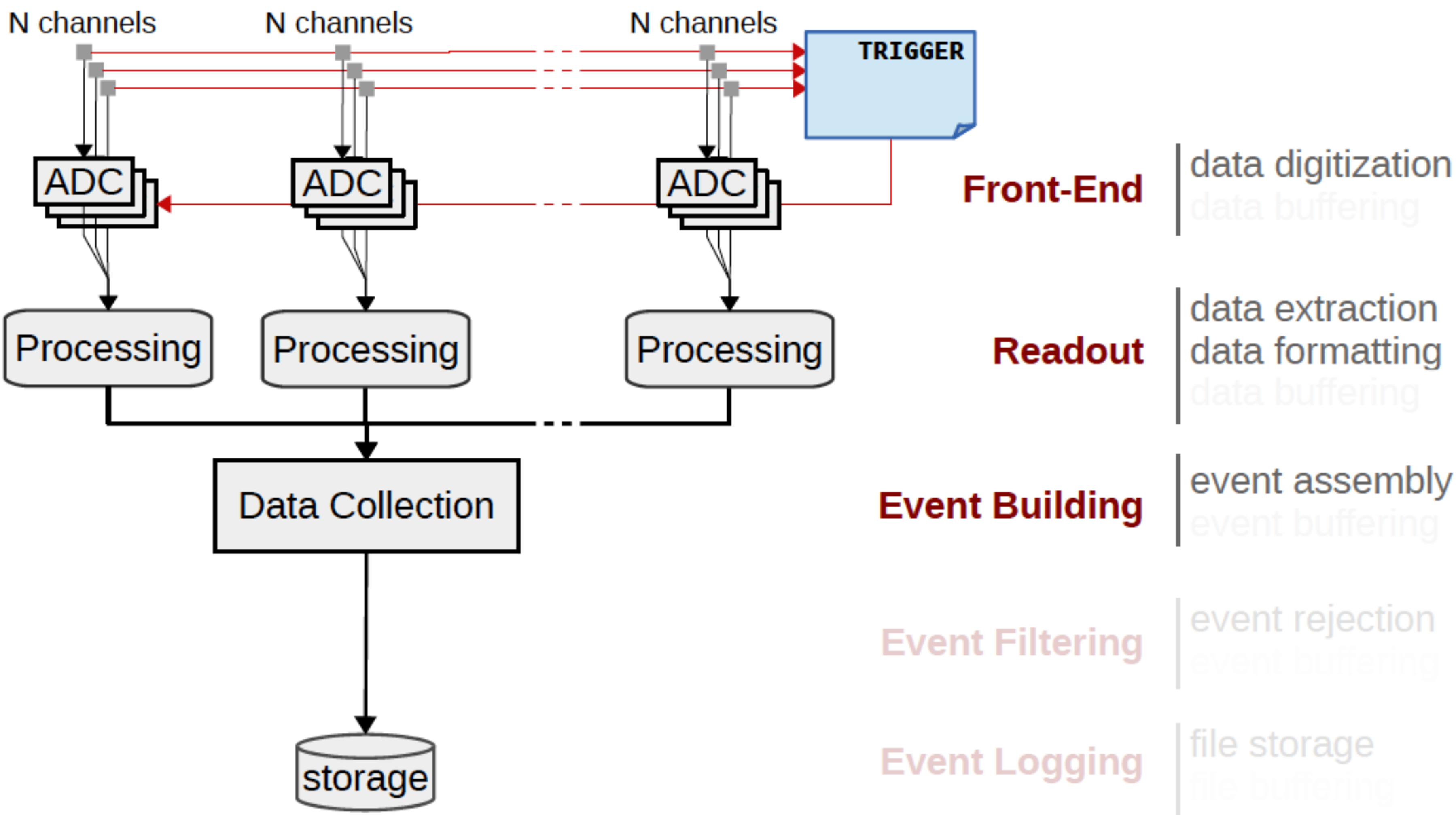
Event Building | event assembly, event buffering

Event Filtering | event rejection, event buffering

Event Logging | file storage, file buffering

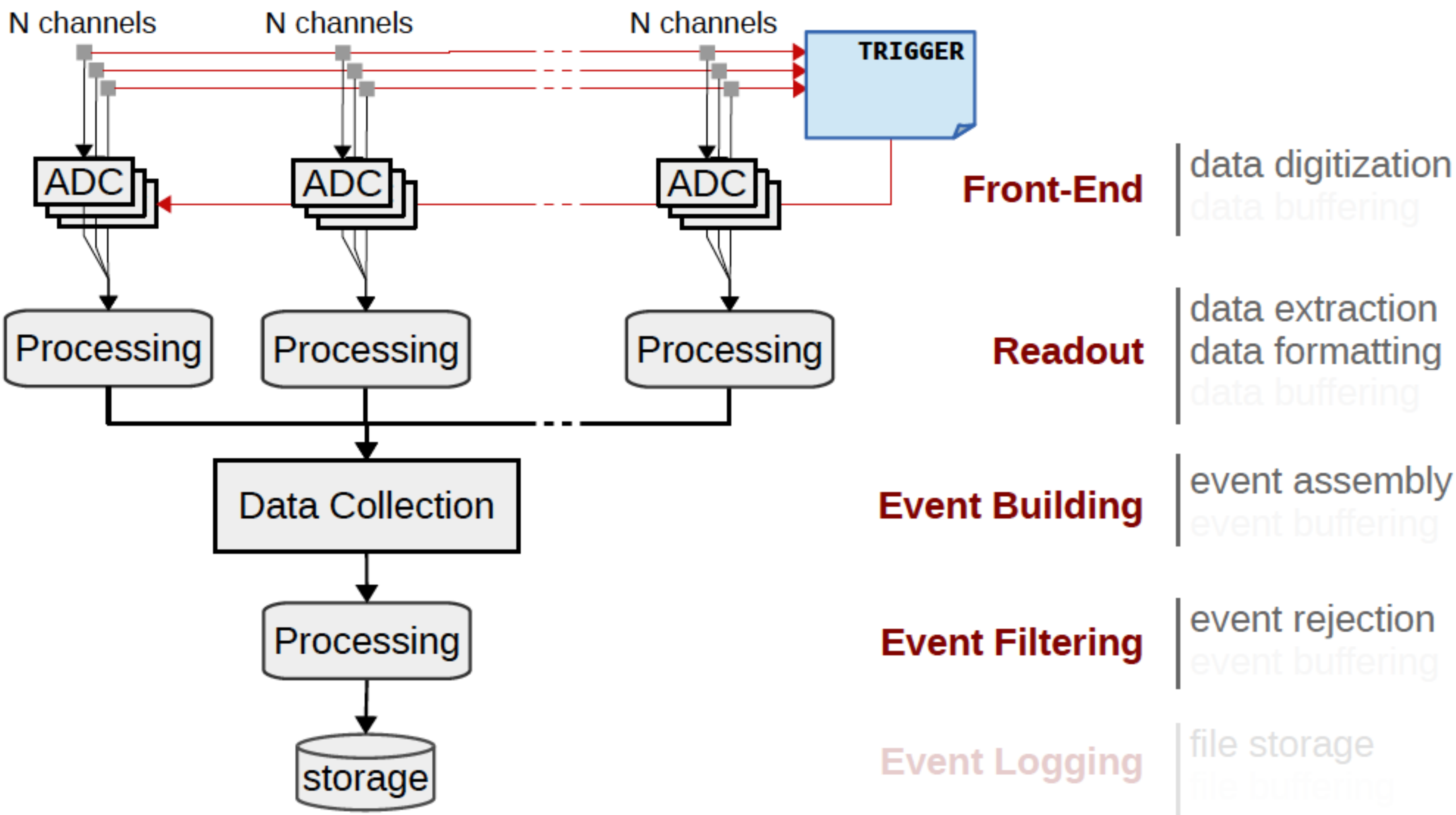
A bit more complicated….

- The increased number of channels require hierarchical structure with well defined interfaces between components



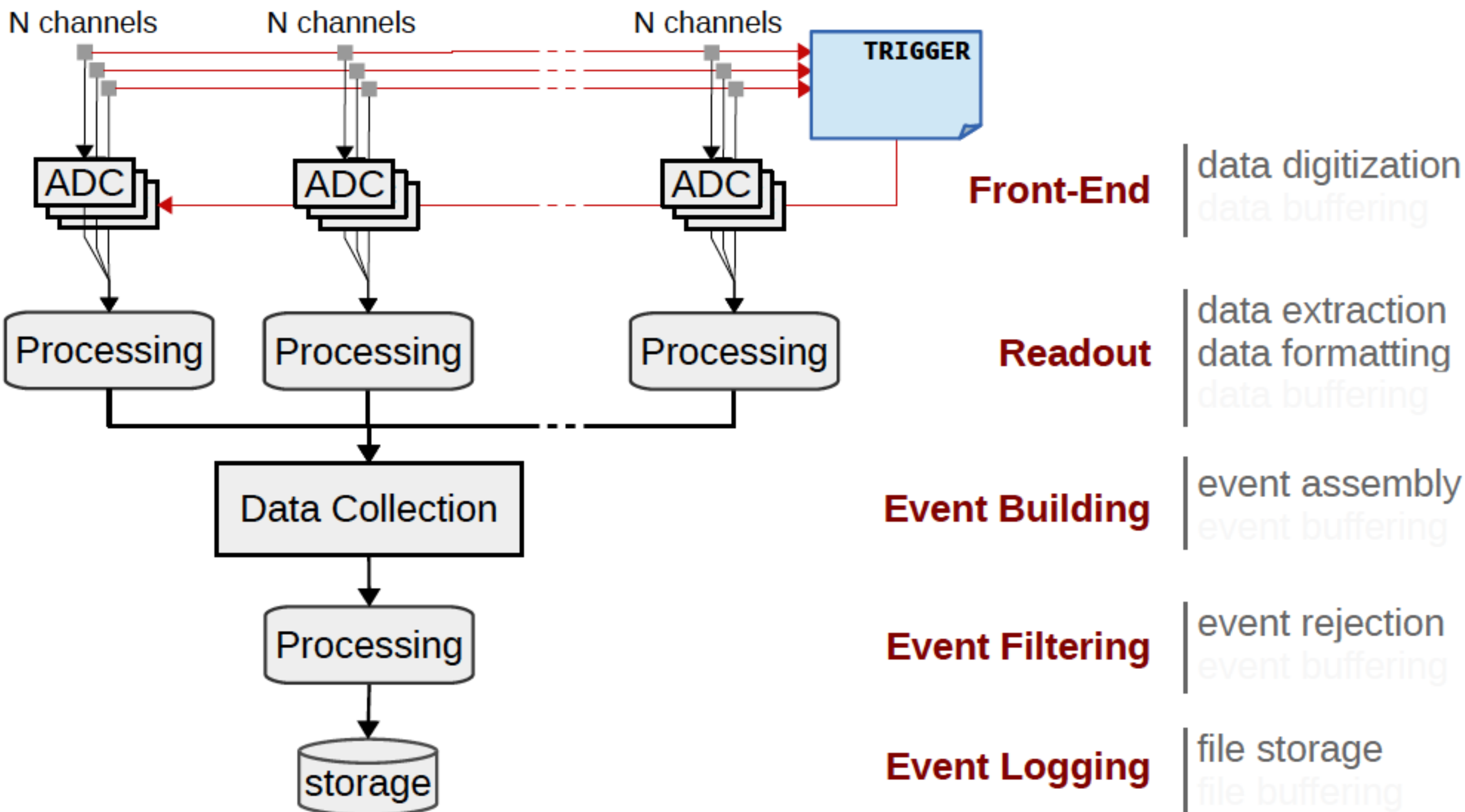
A bit more complicated….

- The increased number of channels require hierarchical structure with well defined interfaces between components



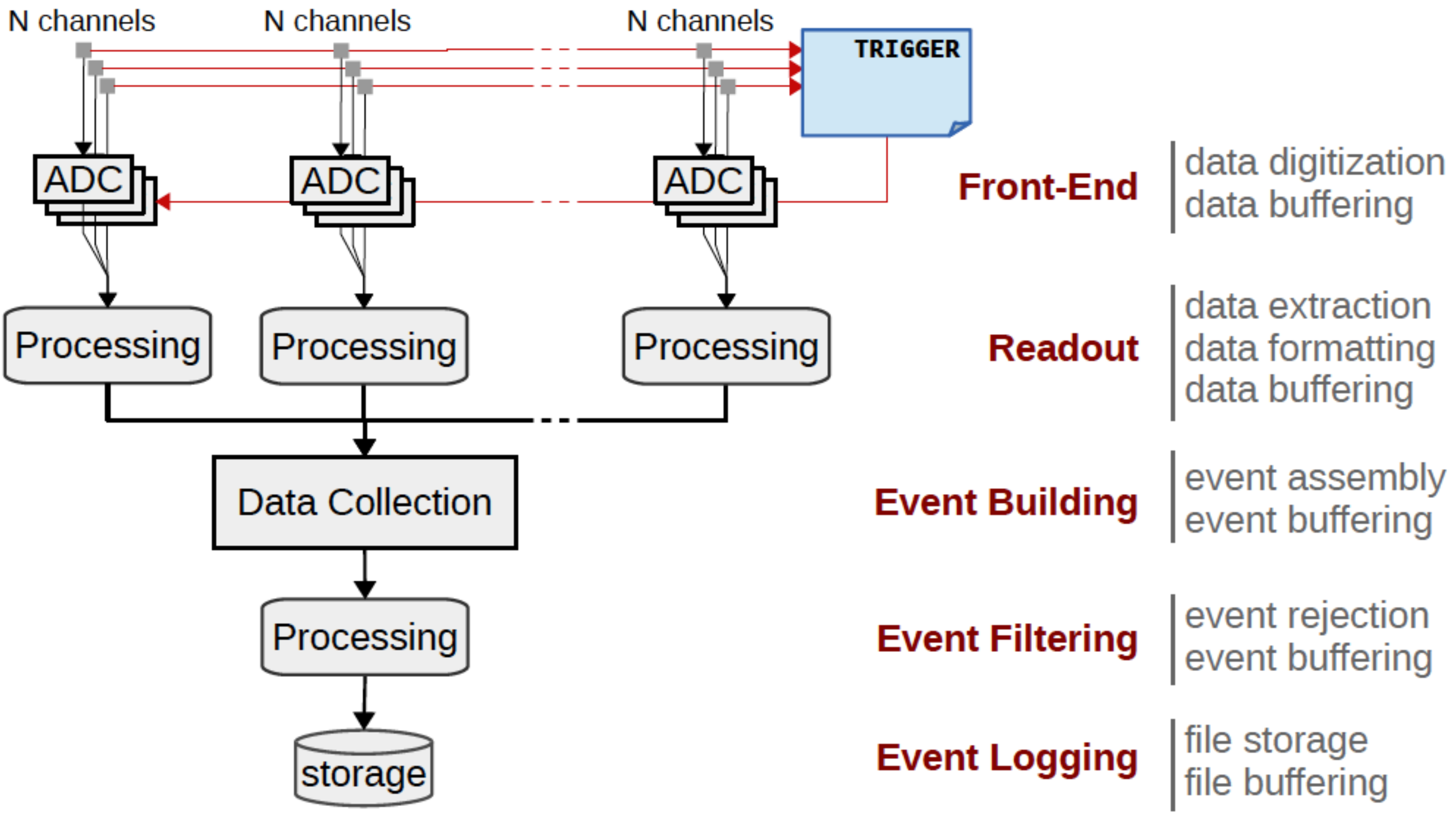
A bit more complicated….

- The increased number of channels require hierarchical structure with well defined interfaces between components



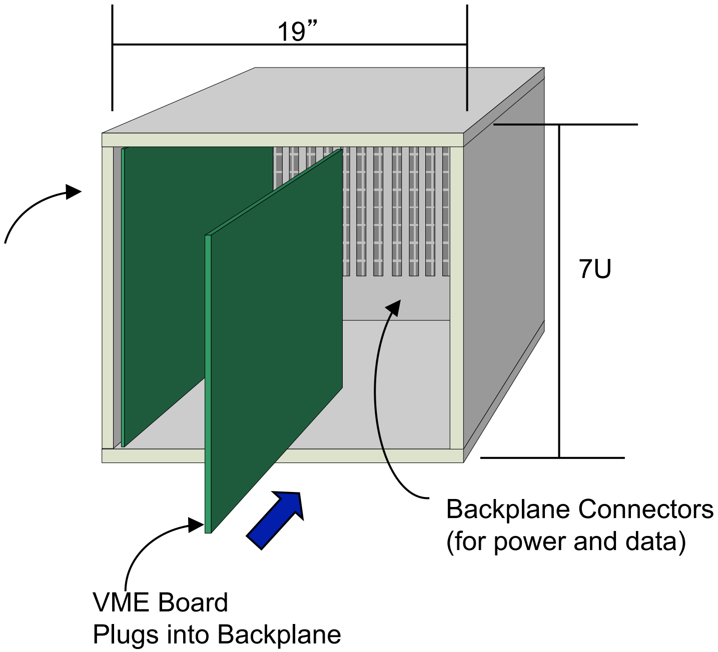
A bit more complicated….

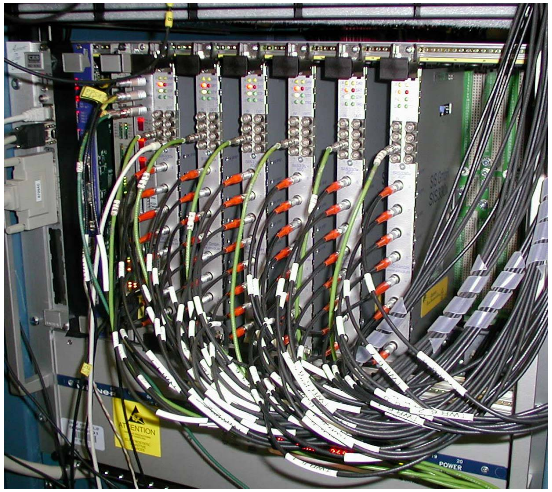
- Buffering usually needed at all levels



Read-out Topology

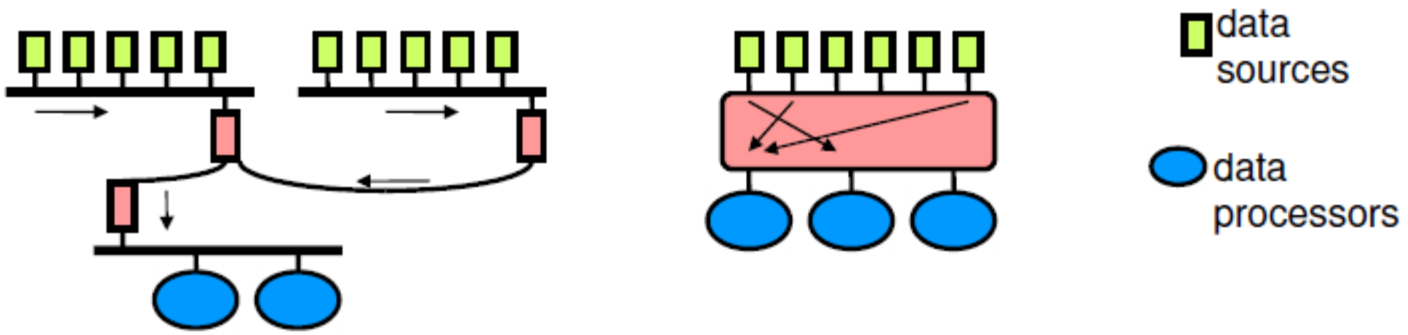
- Reading out = building events out of many detector channels
- We define “building blocks”
  - Example: readout crates, event building nodes, …
- Crate: many modules put in a common chassis which provides
  - Mechanical support
  - Power
  - A standardised way to access the data
  - Provides signal and protocol standard for communication
- All this is provided by standards for (readout) electronics such as NIM or VME (IEEE 1014)





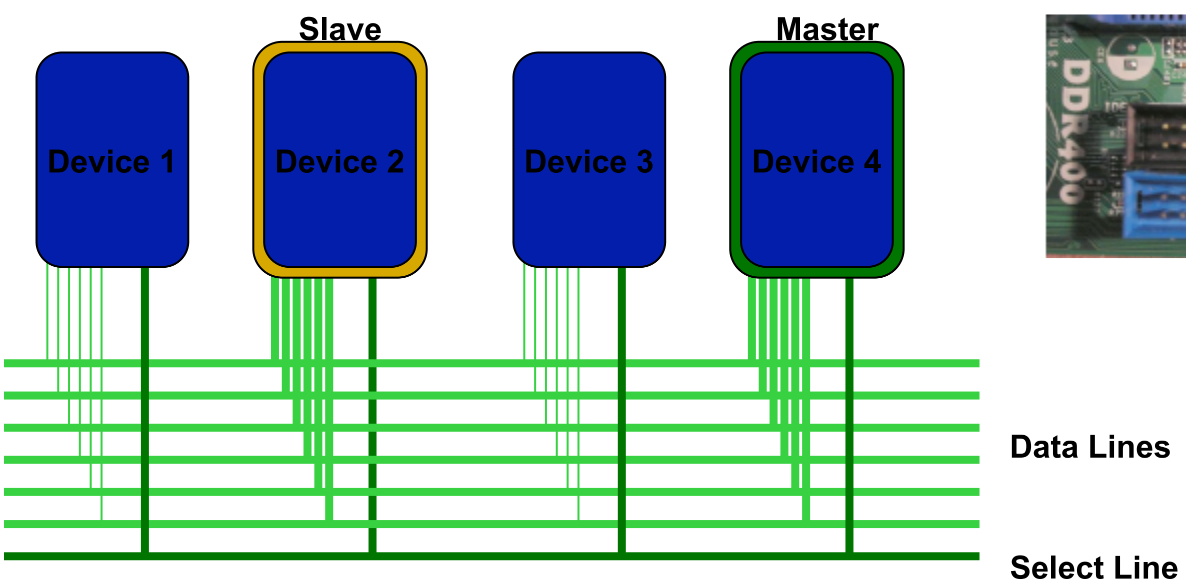
Read-out Topology

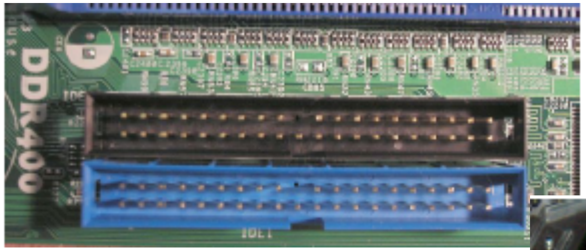
- How to organize the interconnections inside the building blocks and between building blocks?
- Two main classes: bus or network
  - Both of them are very generic concepts



Bus

- A bus connects two or more devices and allows them to communicate
  - Bus → group of electrical lines
- Examples: VME, PCI, SCSI, Parallel ATA, …
- The bus is shared between all devices on the bus → arbitration is required
- Devices can be masters or slaves (some can be both)
- Devices can be uniquely identified ("addressed") on the bus



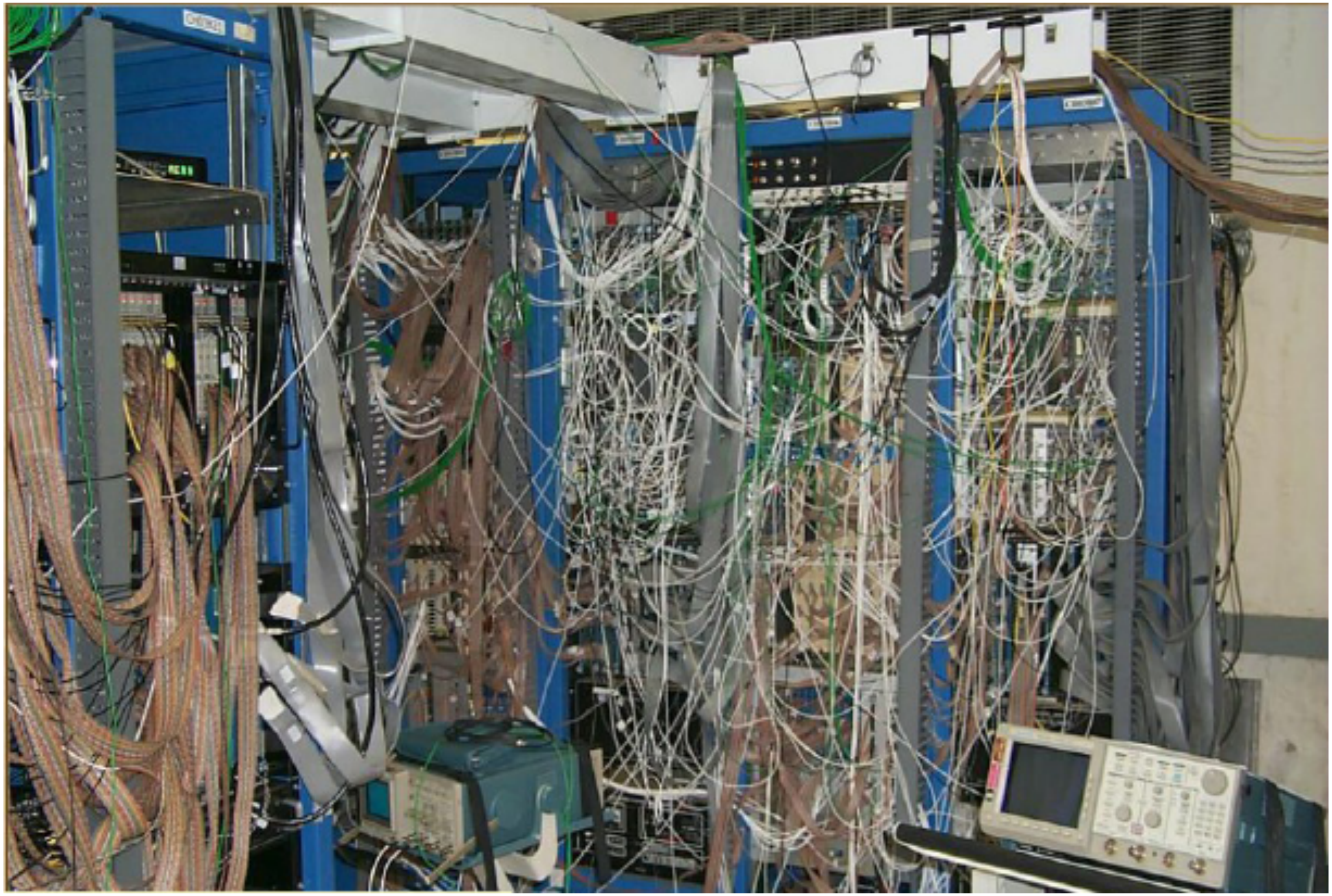


Bus

- Relatively simple to implement
  - Constant number of lines
  - Each device implements the same interface
  - ➔Easy to add new devices
- Scalability issues
  - Number of devices and physical bus-length is limited
  - Each new active device slows everybody down as bus bandwidth* shared among all the devices
  - Maximum bus size (bus width) is limited (128 bit for PC-system bus)
    - Determines how much data can be transmitted at one time
  - Maximum bus frequency (number of elementary operations per second) is inversely proportional to the bus length
- Typical buses have a lot of control, data and address lines (e.g. SCSI cable (Small Computer System Interface)
- Buses are typically useful for systems < 1 GB/s

Bandwidth = amount of data transferred / per unit of time (measured in Bytes/h)

Bus: another limitation

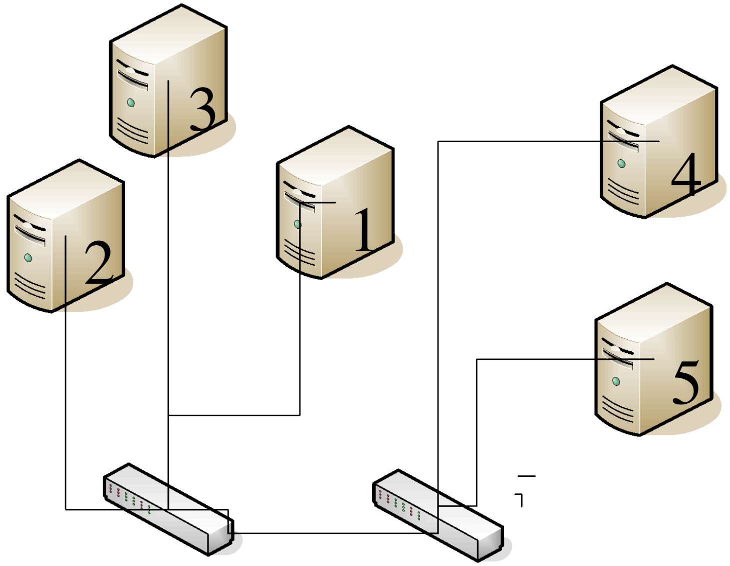


Network based DAQ

- In large (HEP) experiments we typically have thousands of devices to read, which are sometimes very far from each other → *buses can not do that*
- Network technology solves the scalability issues of buses
  - Examples: Ethernet, Telephone, Infiniband, …
  - Devices are equal ("peers")
  - They communicate directly with each other by sending messages
    - No arbitration necessary
    - Bandwidth guaranteed
  - Data and control use the same path
    - Much fewer lines (e.g. in traditional Ethernet only two)
  - On an network a device is identified by a network address
    - Eg: phone-number, MAC address
  - At the signaling level buses tend to use parallel copper lines. Network technologies can be also optical or wireless

Switched Networks

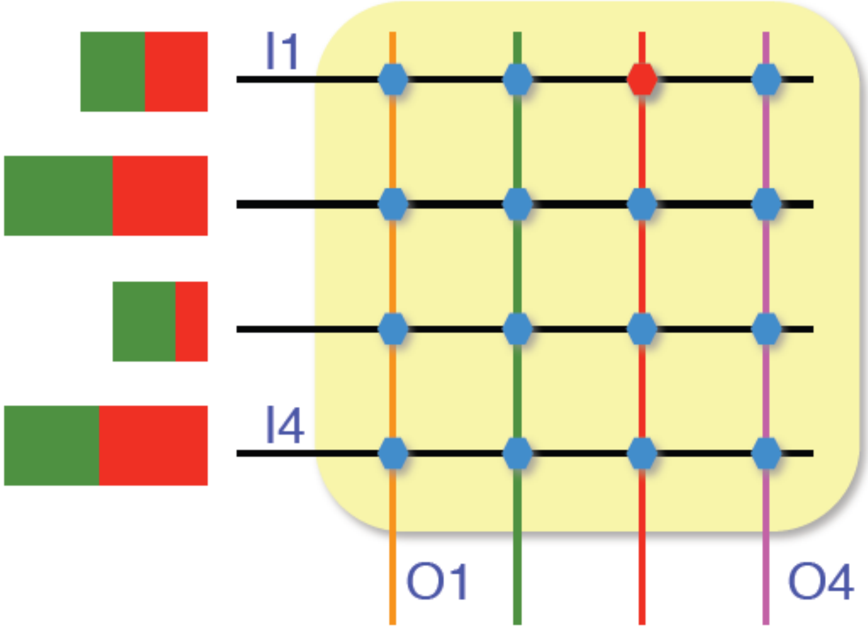
- Modern networks are *switched with point-to-point links*
- Each node is connected either to another node or to a switch
- Switches can be connected to other switches
- A path from one node to another leads through 1 or more switches
- Switches move messages between sources and destinations
  - Find the right path
  - Handle "congestion" (two messages with the same destination at the same time)



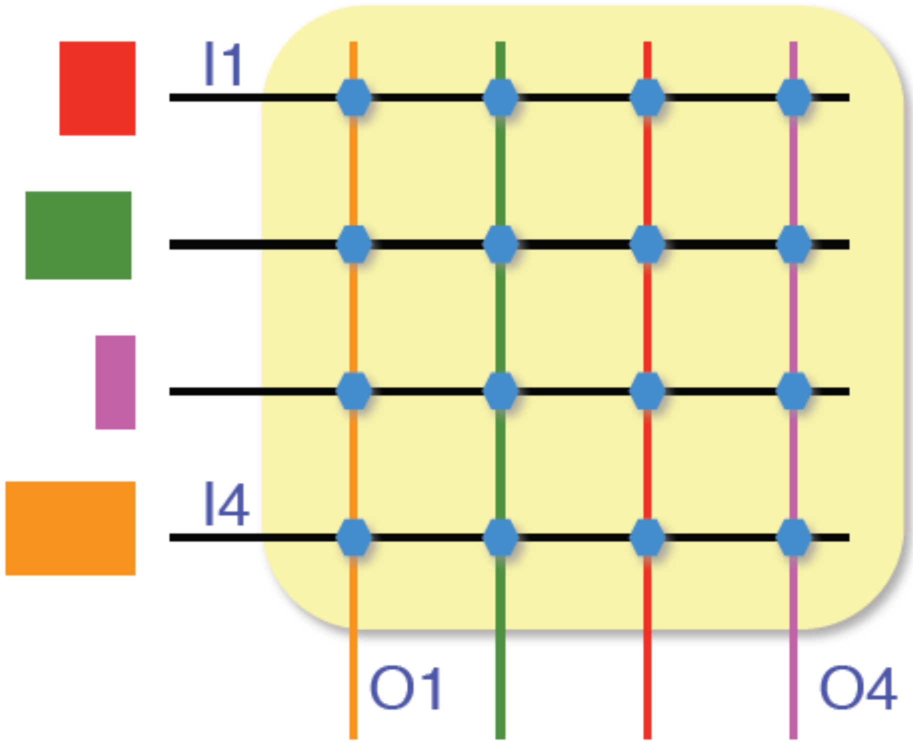
- Example
  - While 2 can send data to 1 and 4, 3 can send at full speed to 5
  - 2 can distribute the bandwidth between 1 and 4 as needed

Switched Network

- Challenge
  - Find the right path
  - Handle "congestion" (two messages with the same destination at the same time)



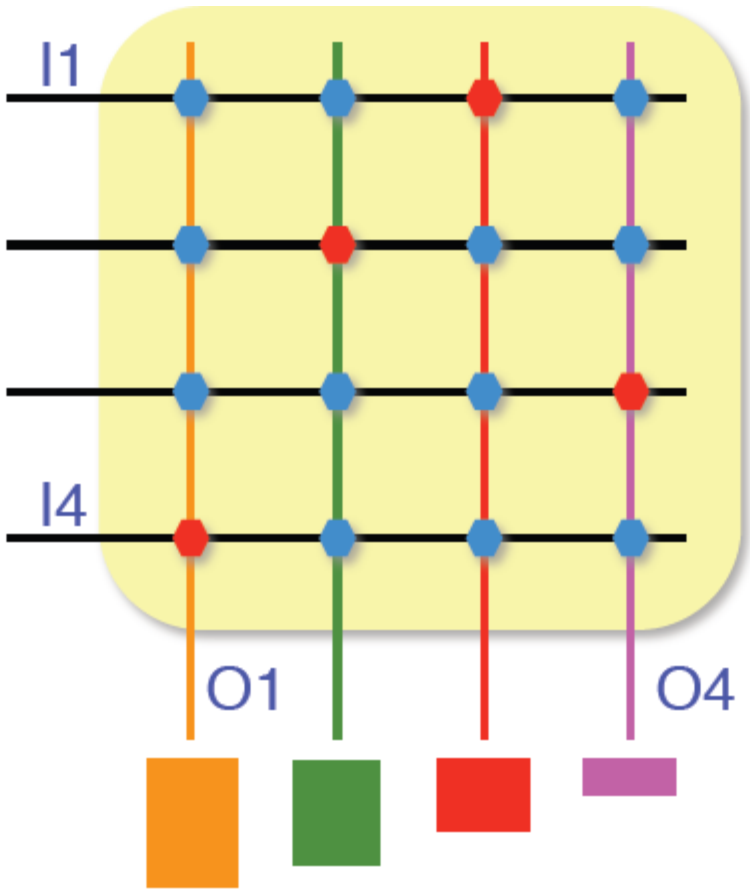
Switch implementation: cross-bar



Collection of switches arranged in matrix configuration

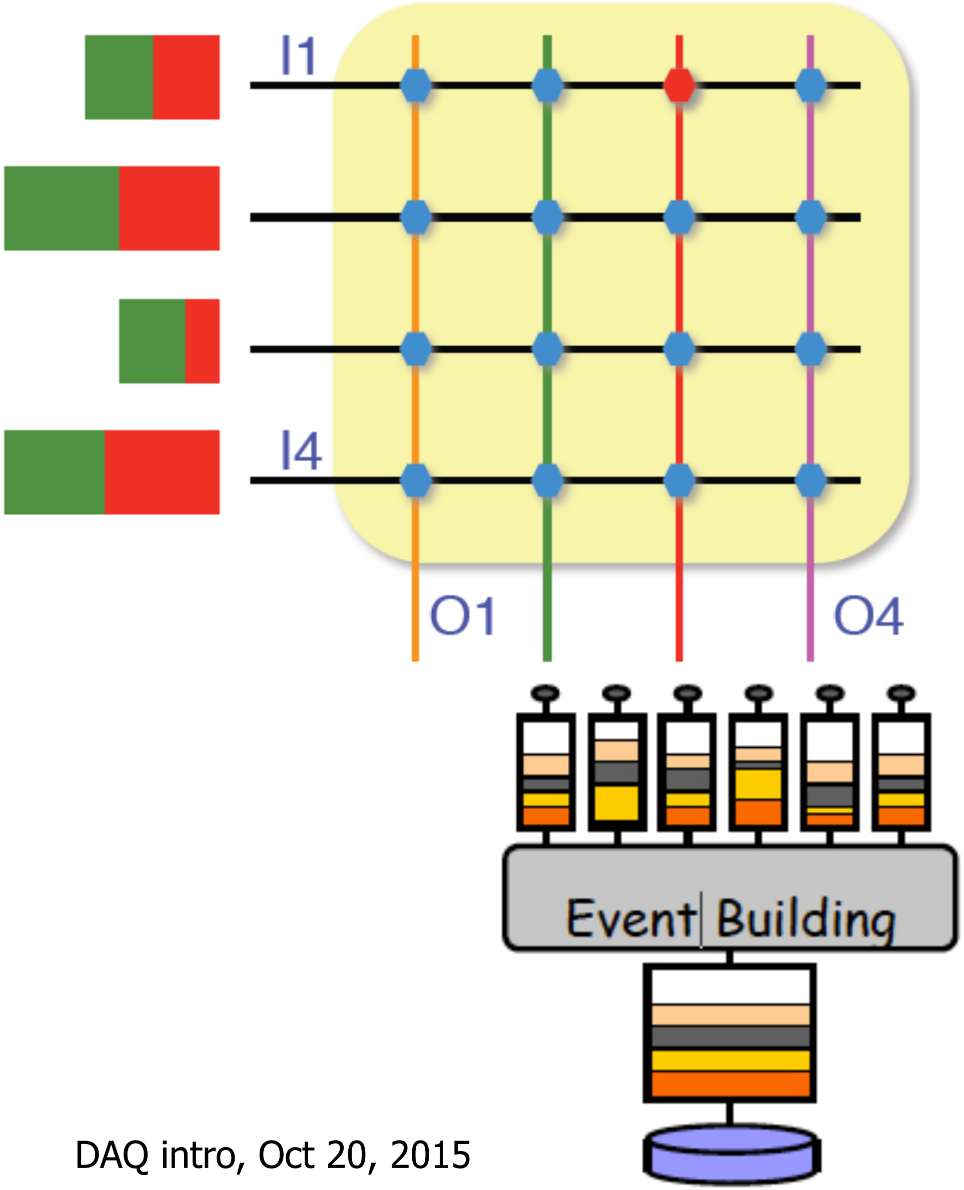
- Paradise scenario:
  - No congestion, since every data package finds a free path through the switch.

Switch implementation: cross-bar



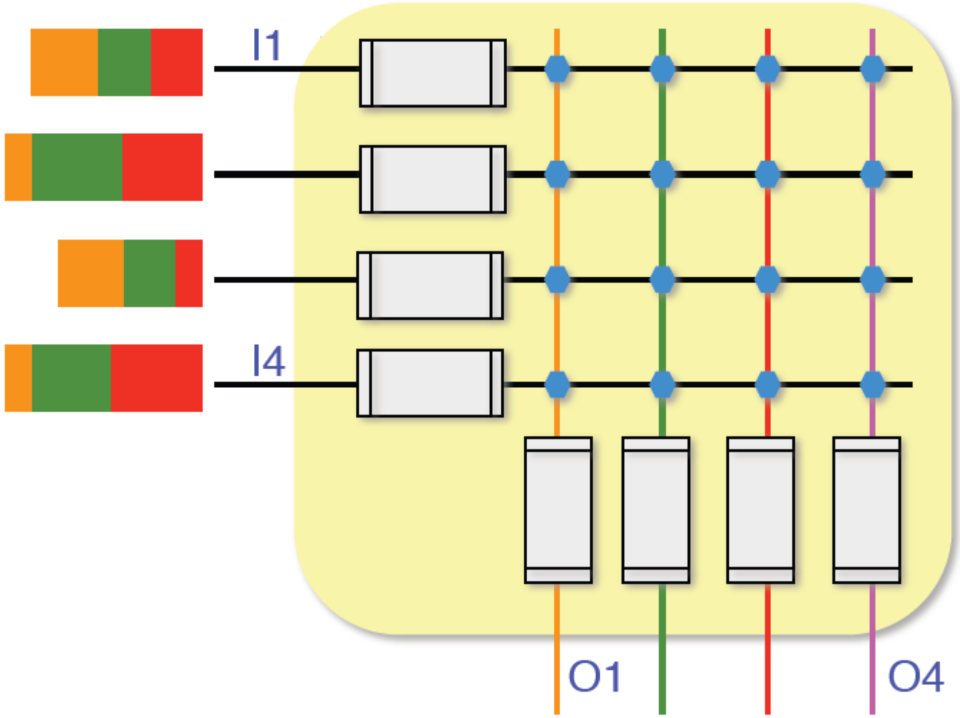
- Paradise scenario:
  - No congestion, since every data package finds a free path through the switch.

Switch implementation: cross-bar



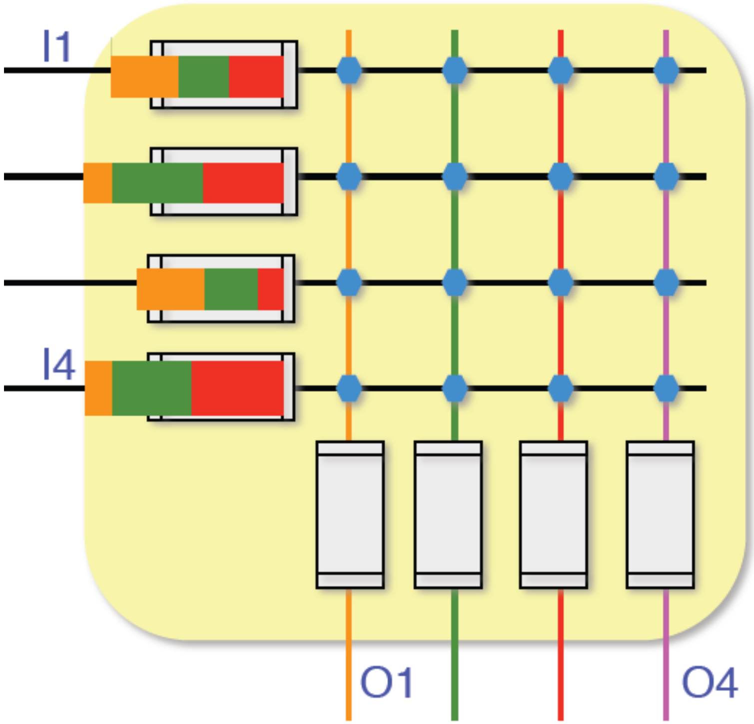
- Nightmare scenario:
  - Only one packet at a time can be routed to the destination. Congestion!
- Event building is an example of such a configuration
- How can we avoid this?

Switch implementation



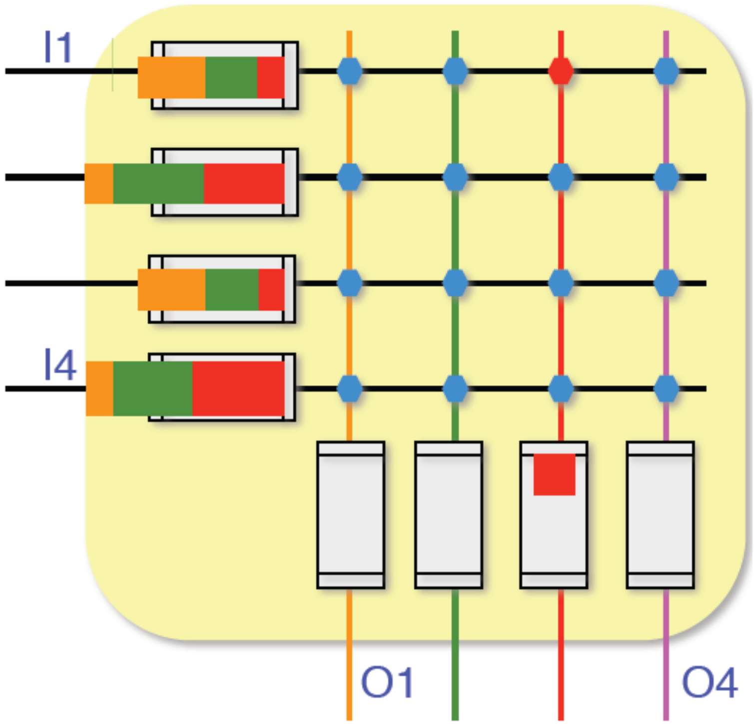
- Use the old "trick"
  - Add buffer
- FIFOs can "absorb" congestion …until they are full.

Switch implementation

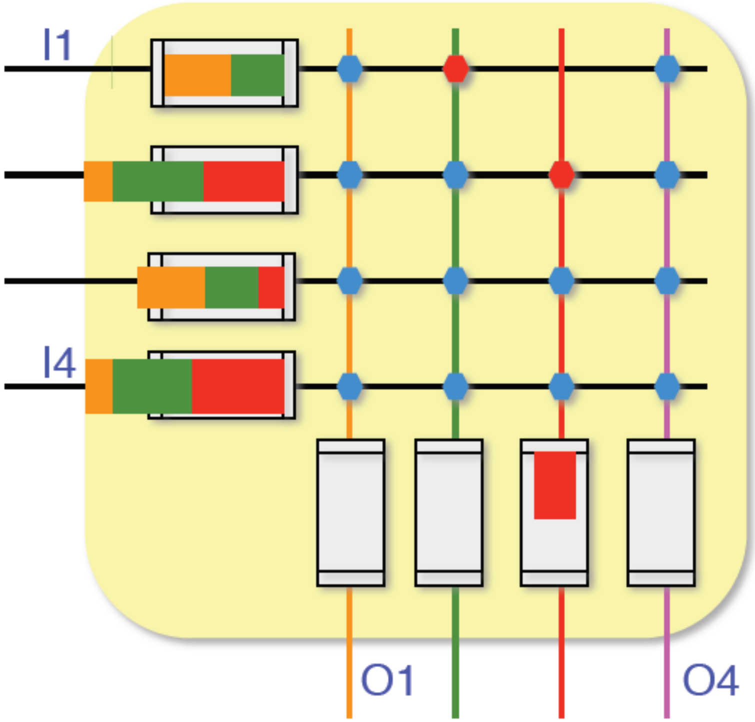


- Use the old "trick"
  - Add buffer
- FIFOs can "absorb" congestion …until they are full.

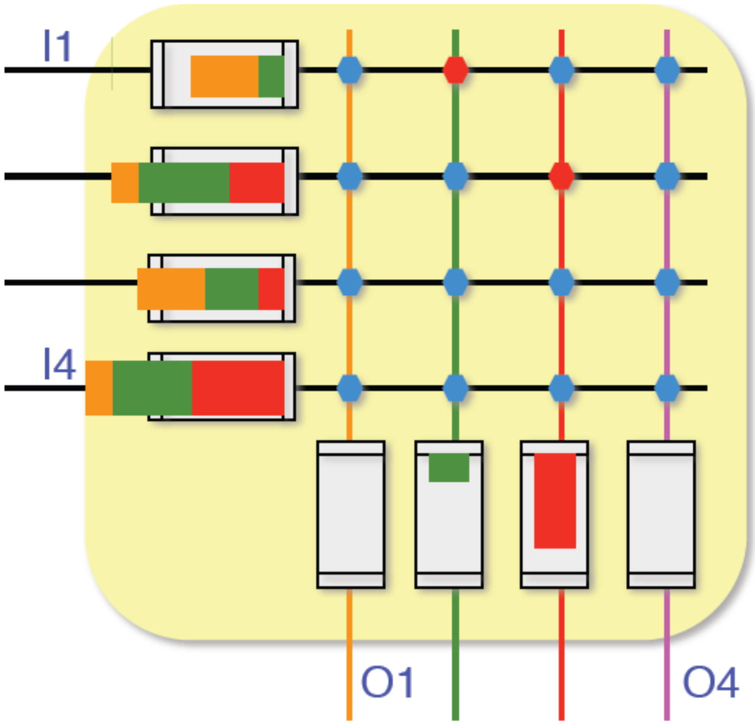
Switch implementation



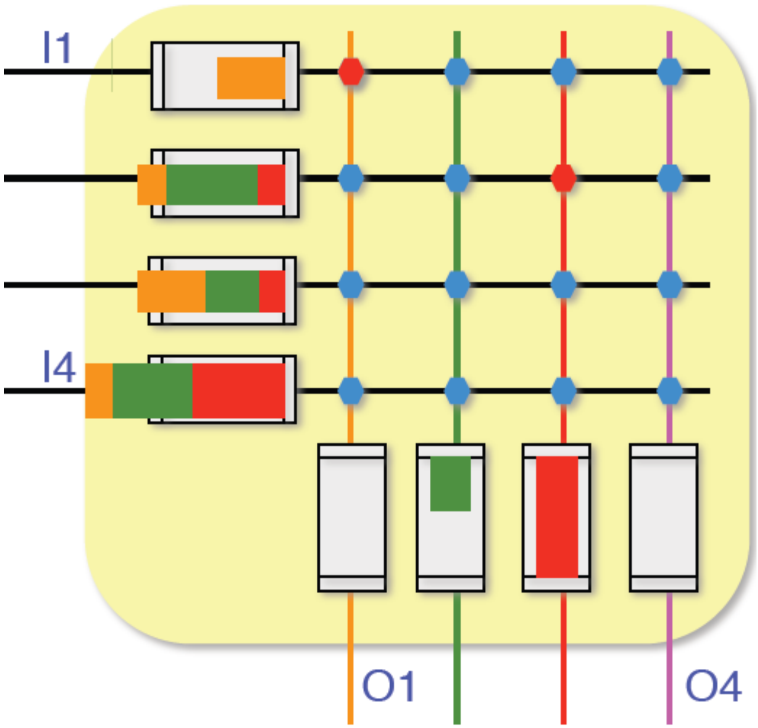
Switch implementation



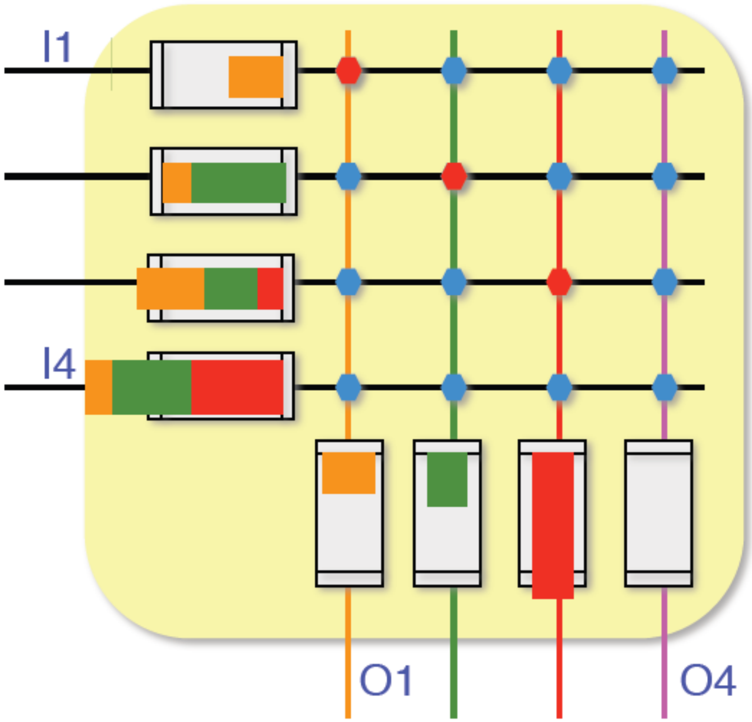
Switch implementation



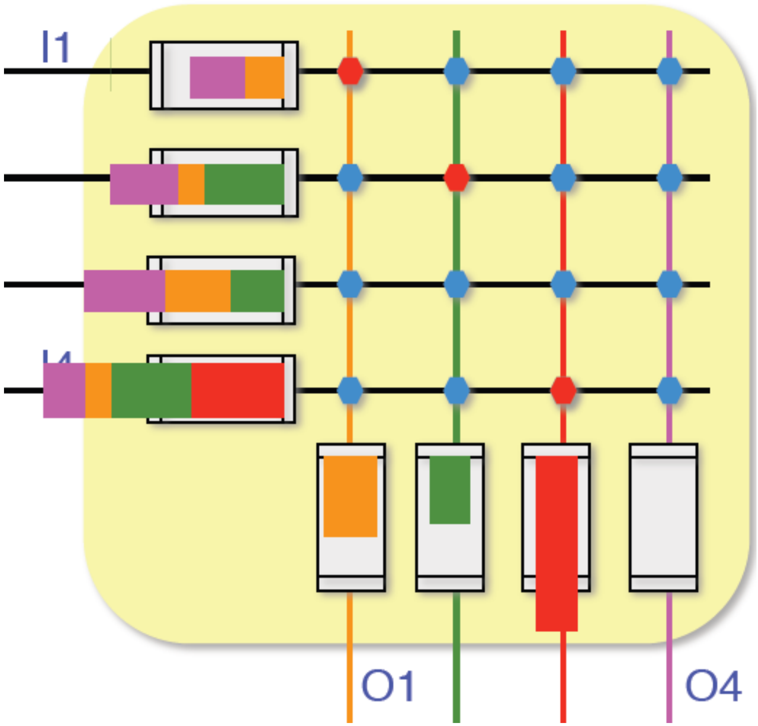
Switch implementation



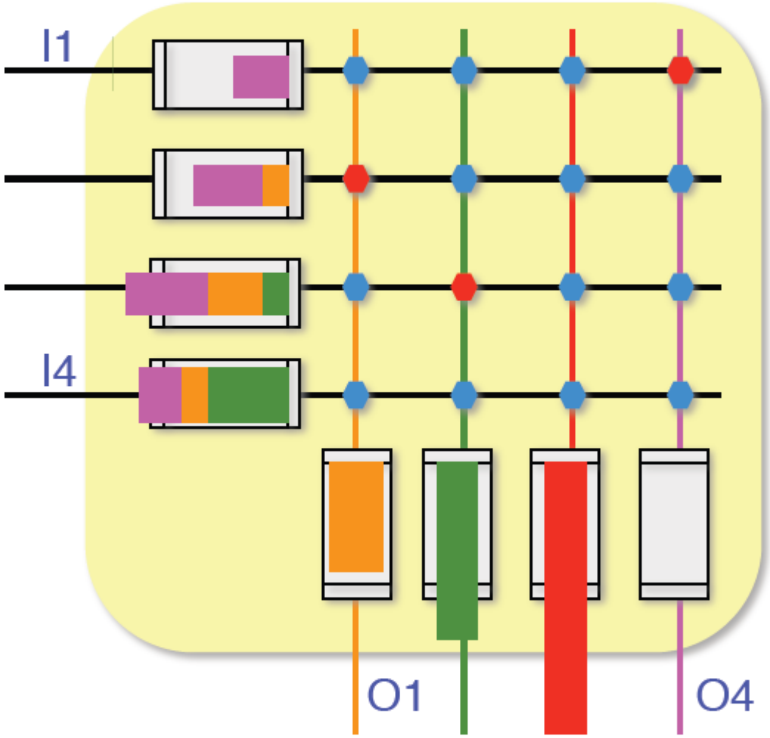
Switch implementation



Switch implementation



Switch implementation



- **Still problematic**:
- Input FIFOs can absorb data fluctuations until they are full. How good it works depends on:
  - FIFOs capacity
  - data distribution
  - Internal speed of the switch
- Traffic: blocking problem remains to some extend

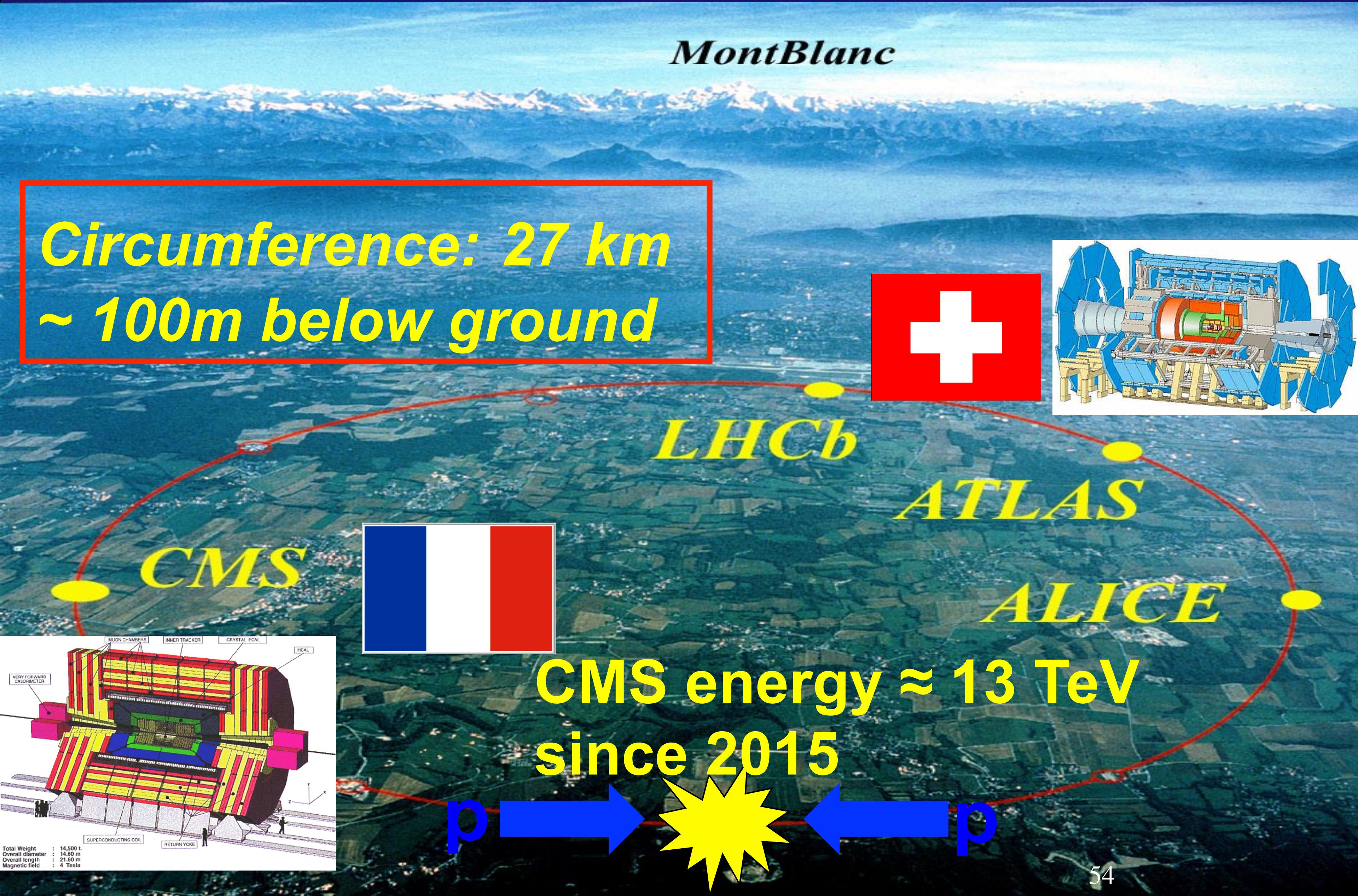
What we have learnt so far

- The principle of a simple data acquisition system
- Introduction to some basic elements: trigger, derandomiser, FIFO, busy logic
- How data is transported
  - Bus versus network

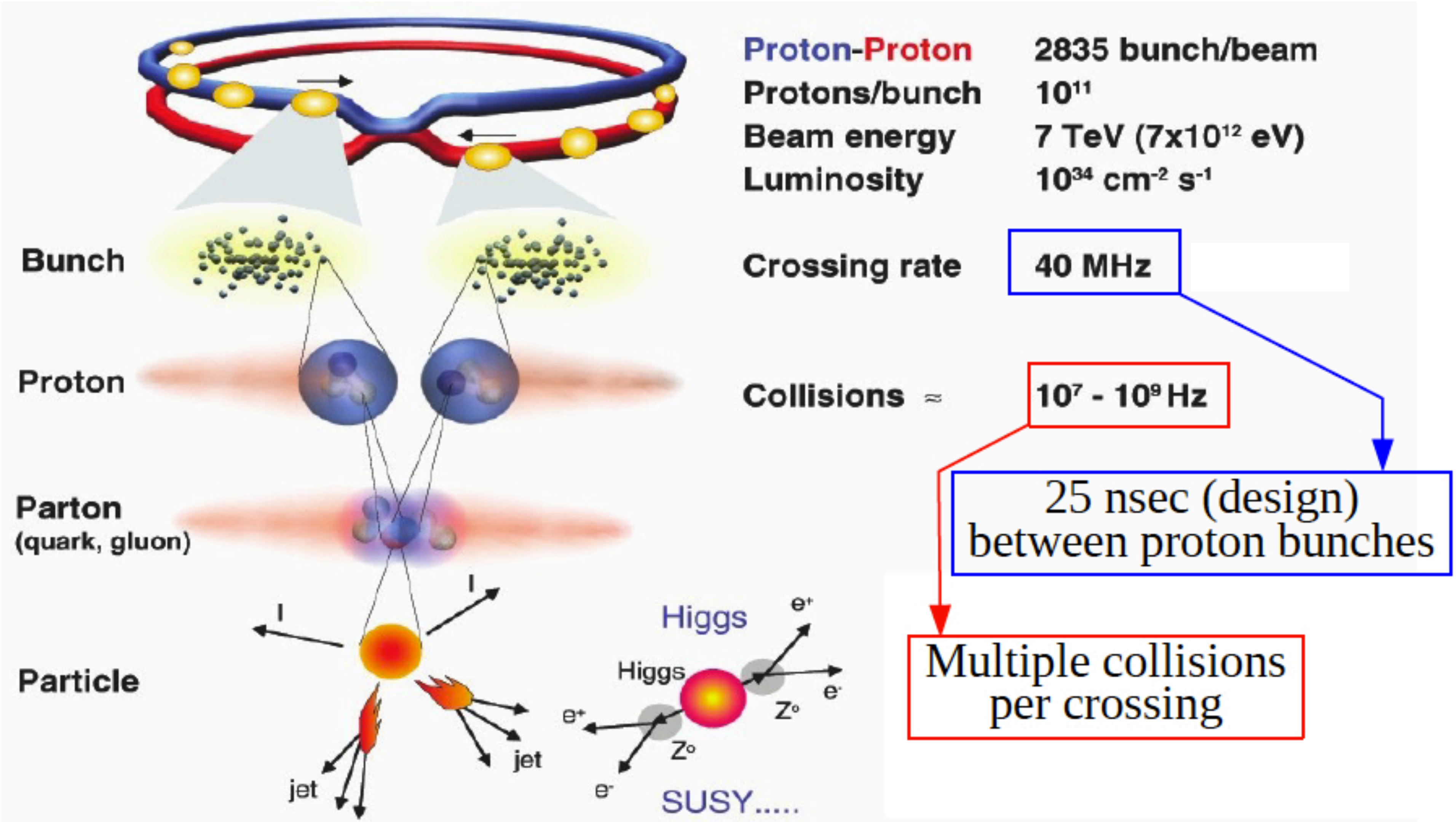
- In the next lecture we will look in more detail at the DAQ used by the experiments at LHC

In case of time

Current biggest TDAQ systems used at CERN

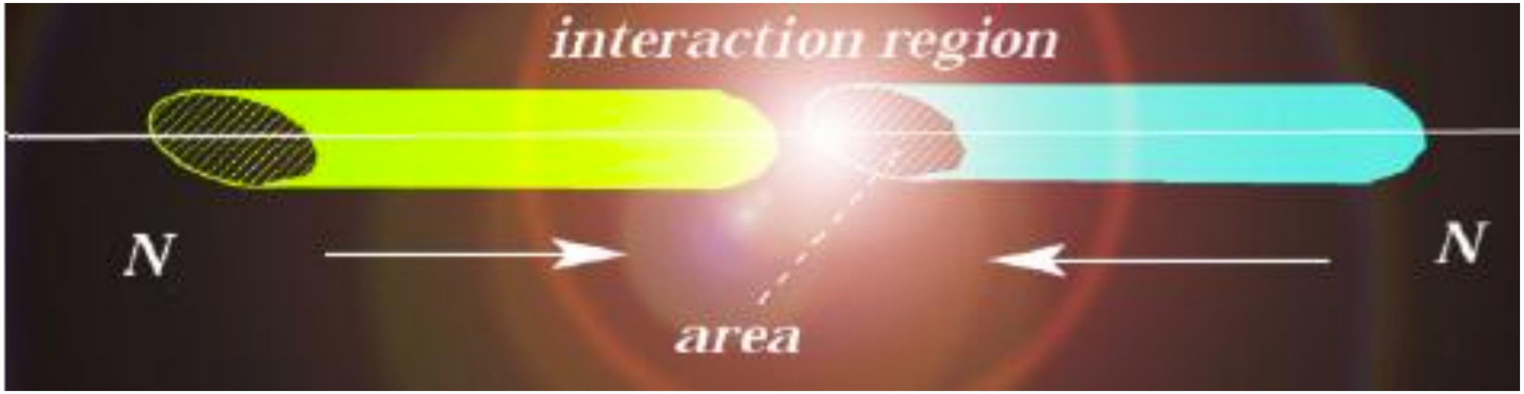


A Few LHC Facts

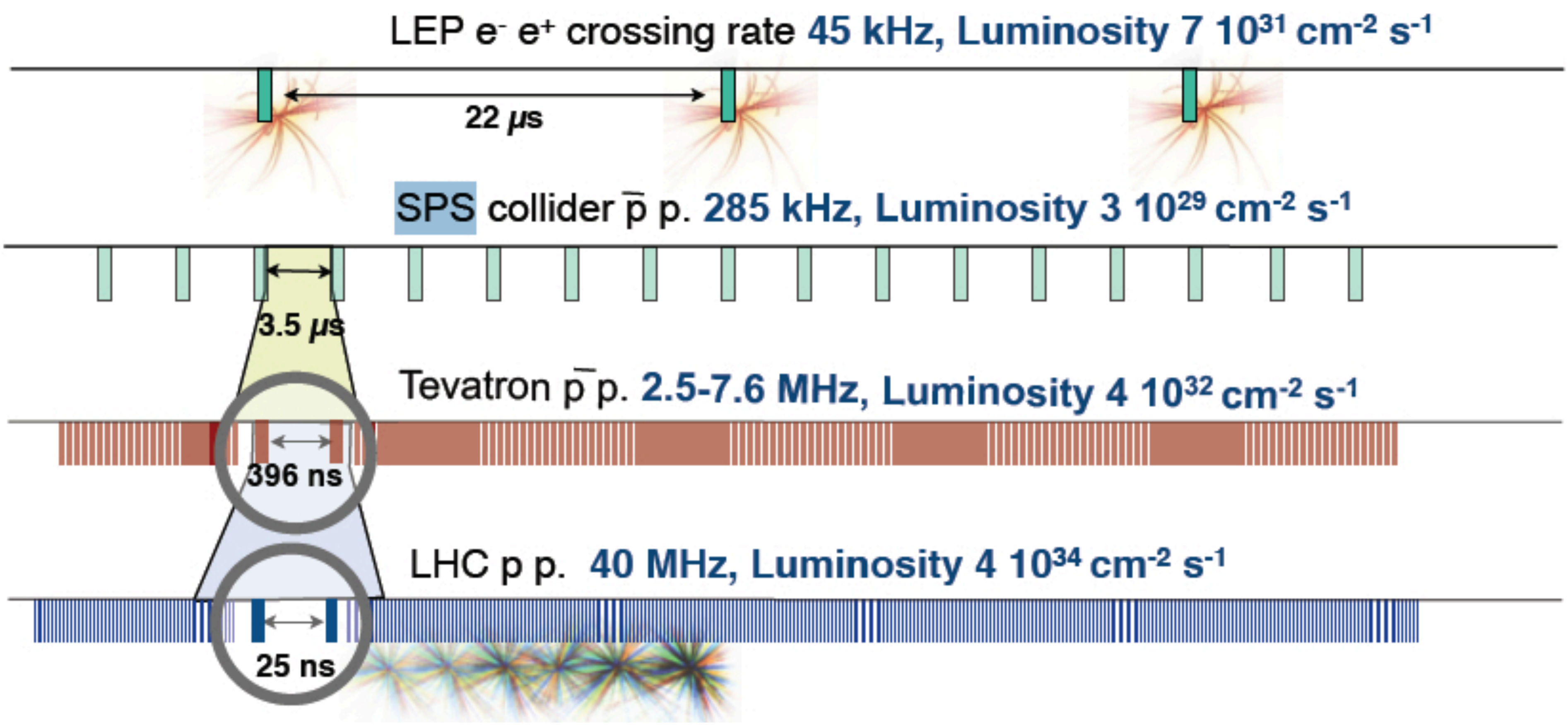


Luminosity

- Definition of luminosity
  - Number of collisions that can be produced per cm^2 and per second.
  - $R = dN/dt = L\ \sigma_p$



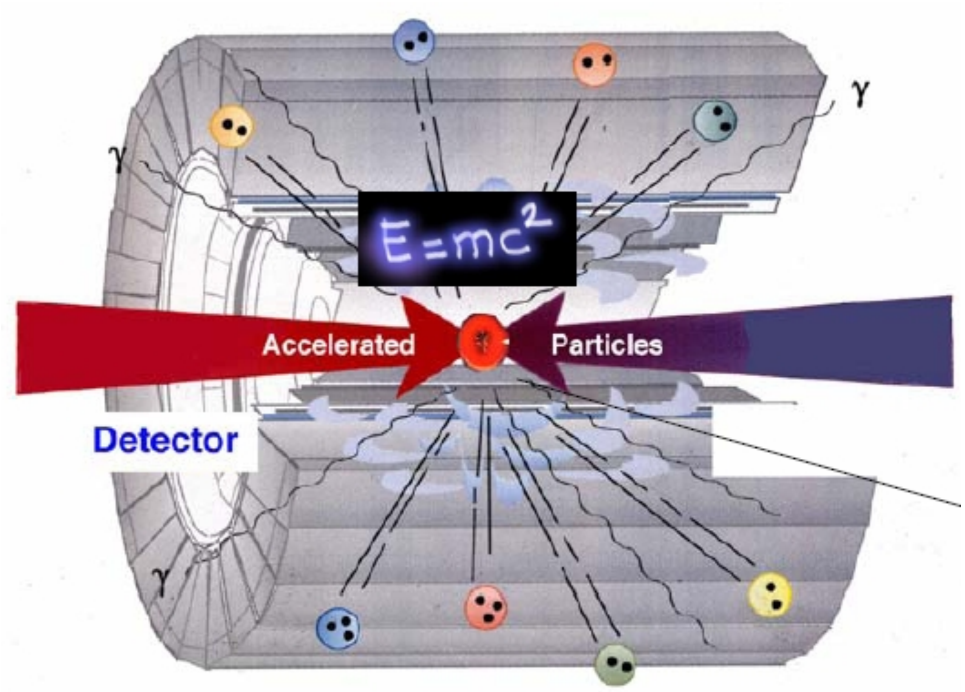
Colliders bunch crossing frequencies

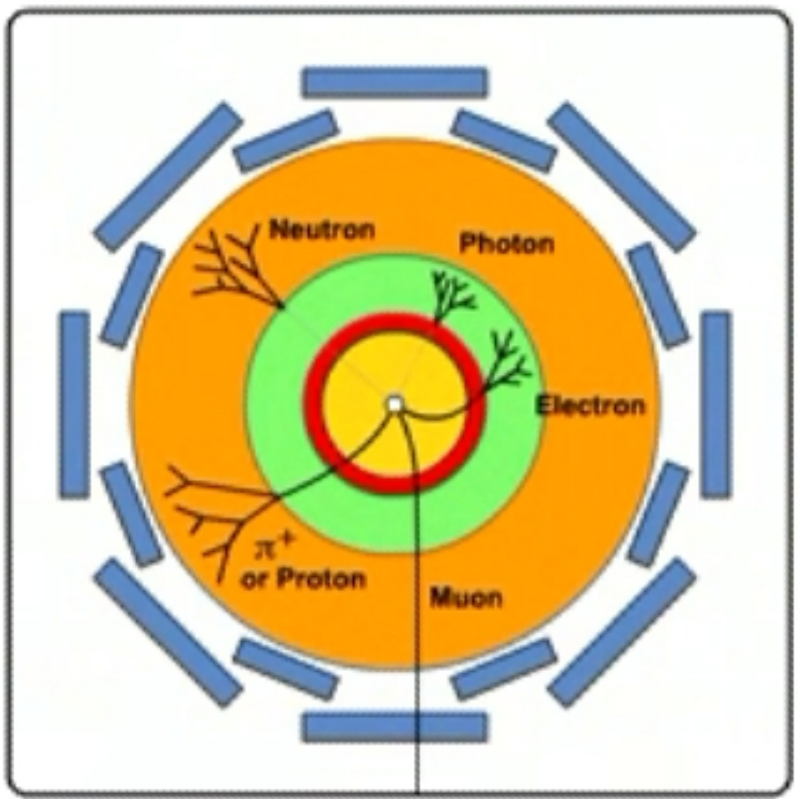


- 25 ns defines an overall time constant for signal integration, DAQ and trigger.

Principle of multi-purpose detector

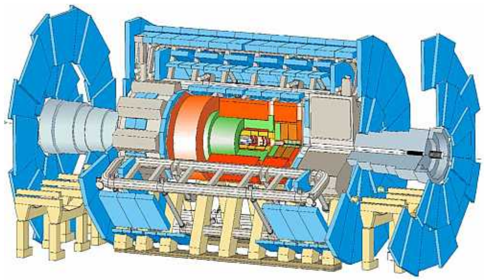
- Detectors built around collision point

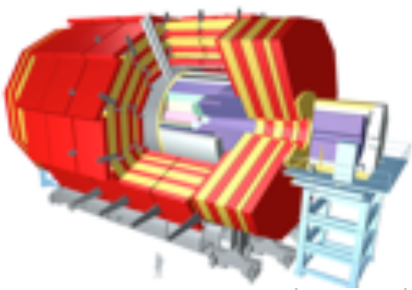


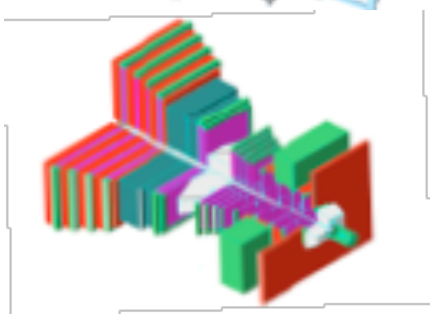


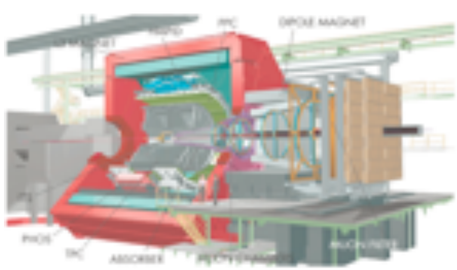
- Several layers of different detectors
  - Separate particle types
  - Measure their energies and direction

The LHC Experiments



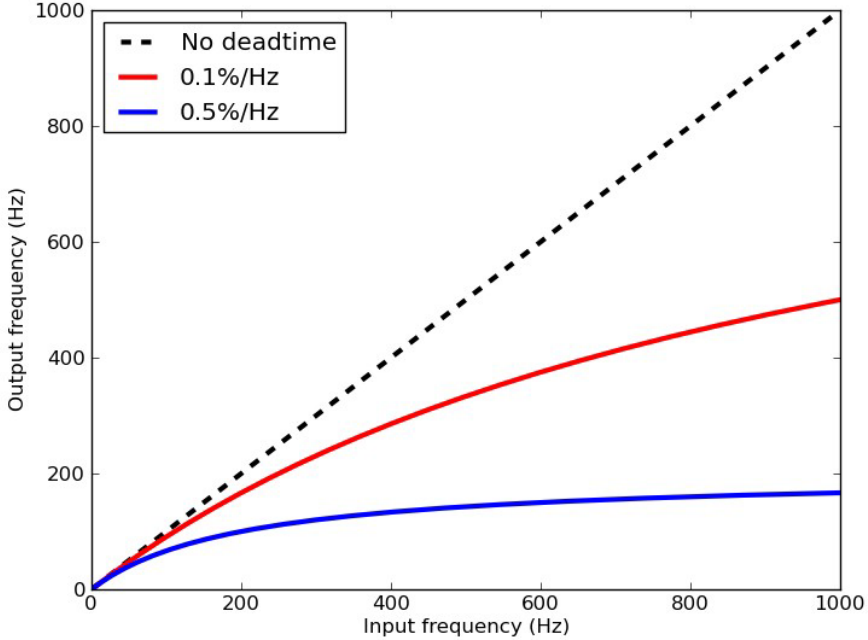


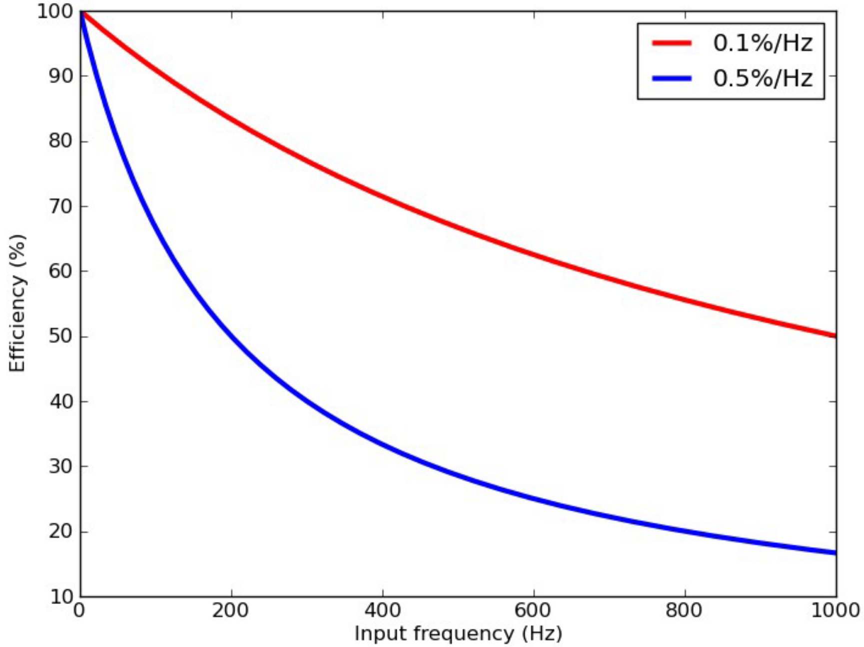




- ATLAS
  - Study of pp and heavy ion collisions
  - Length: 40m, height: 22m, weight: 7000t
  - 10^8 readout channels, event size: 1.5MB
- CMS
  - Study of pp and heavy ion collisions
  - Length: 21m, height: 15m, weight: 12500t
  - 10^7 readout channels, event size 1MB
- LHCb
  - Study of CP violation in B decays
  - Length: 21m, height: 10m, weight: 5600t
  - 10^6 readout channels, event size: 35kB
- ALICE
  - Study of heavy ion collisions
  - Length: 21m, height: 16m, weight: 10000t
  - 10^6 readout channels, event size: 50MB

DAQ deadtime and efficiency

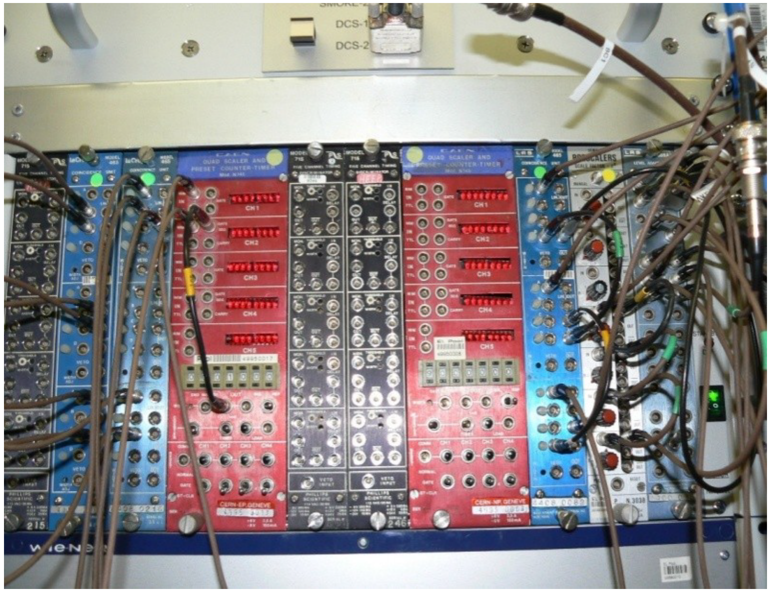


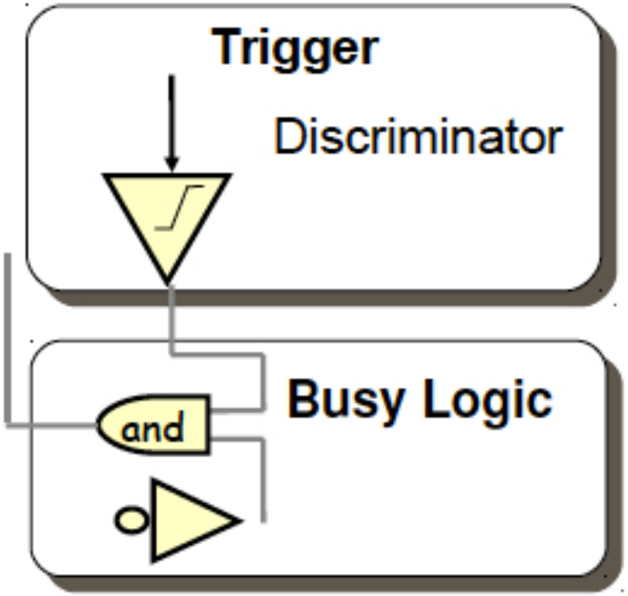


- If we want to obtain $\nu \sim f$ ($\varepsilon \sim 100\%$) → $f\tau \ll 1$ → $\tau \ll 1/f = \lambda$
  - f=1kHz, ε=99% → τ<0.1ms → $1/\tau$>10kHz
- In order to cope with the input signal fluctuations, need to overdesign DAQ system by a factor 10. Can this be mitigated?

NIM

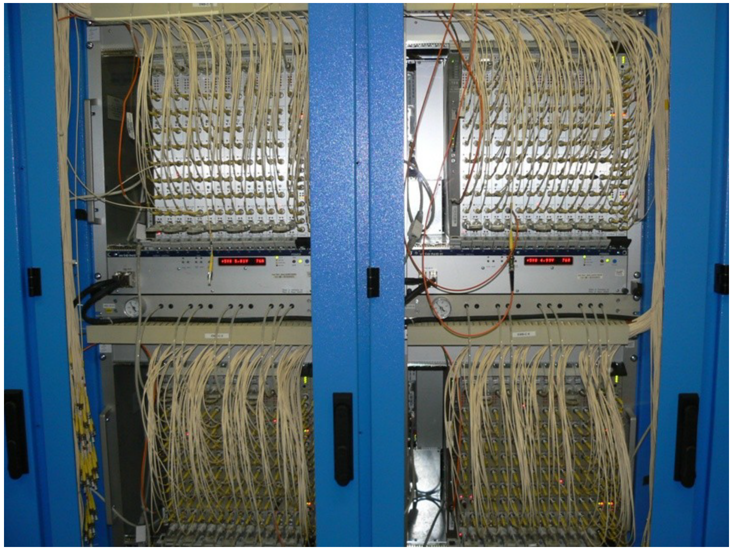
- NIM (1964)
  - "Nuclear Instrumentation Modules"
- NIM modules usually
  - Do not need software, are not connected to PCs
  - Implement logic and signal processing functions
    - Discriminators, Coincidences, Amplifiers, Logic gates, …
- Typically implement basic Trigger and Busy system
- New modules still appear on the market
  - Very diffused in medium-sized HEP experiments
  - Found in counting rooms of LHC exp.

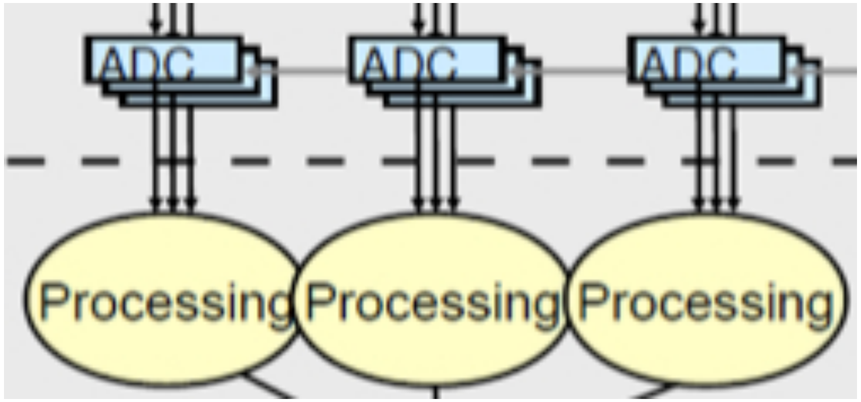




VME

- VMEbus: modules communicate via a "backplane"
  - Standardised way to access data
- Choice of many HEP experiments
  - Relatively simple protocol
  - A 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

Deadtime and Efficiency

- System busy from trigger to end of processing
  - Trigger rate with no deadtime = input rate f per sec.
  - Dead time / trigger = τ sec.
    - Ratio between the time the DAQ is busy and the total time
  - For 1 second of live time = $1 + f\tau$ seconds
  - Live time fraction = $1 / (1 + f\tau)$
  - Real trigger (output) rate ν = $f / (1 + f\tau)$ per sec.
  - Efficiency: $N_{saved}/N_{tot} = \nu/f = 1/(1 + f\cdot\tau)$
    - Note, due to the fluctuations introduced by the stochastic process the efficiency will always be less 100%