Introduction to Data AcQuisition

ISOTDAQ 2024: 14th International School of Trigger and Data Acquisition

Hefei, 19th Jun 2024 Andrea.Negri@unipv.it

Acknowledgment

- Lecture inherited from Wainer Vandelli
  - Material and ideas taken from: Roberto Ferrari, Clara Gaspar, Niko Neufeld, Lauren Tompkins, ...
- Errors and flaws are mine

Introduction

- Aim of this lesson is to introduce the **basic DAQ concepts** avoiding as many technological details as possible
  - The following lectures will cover these aspects
- Focus on High Energy Physics
  - But key concepts are common to other areas
- w/ links to the agenda
  - the lectures
  - the labs

Outline

- Introduction
  - What is DAQ?
  - Overall framework
- Basic DAQ concepts
  - Digitization, Latency
  - Deadtime, Busy, Backpressure
  - De-randomization
- Scaling up
  - Readout and Event Building
  - Buses vs Network
- Data encoding

What is DAQ?

[Wikipedia]

- **D**ata **A**c**Q**uisition (**DAQ**) is
  - the 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 PC
- Components:
  - **Sensors**: convert physical quantities to electrical signals
  - **Analog-to-digital converters**: convert conditioned sensor signals to digital values
  - Processing and storage elements

What is DAQ?

[Wikipedia]

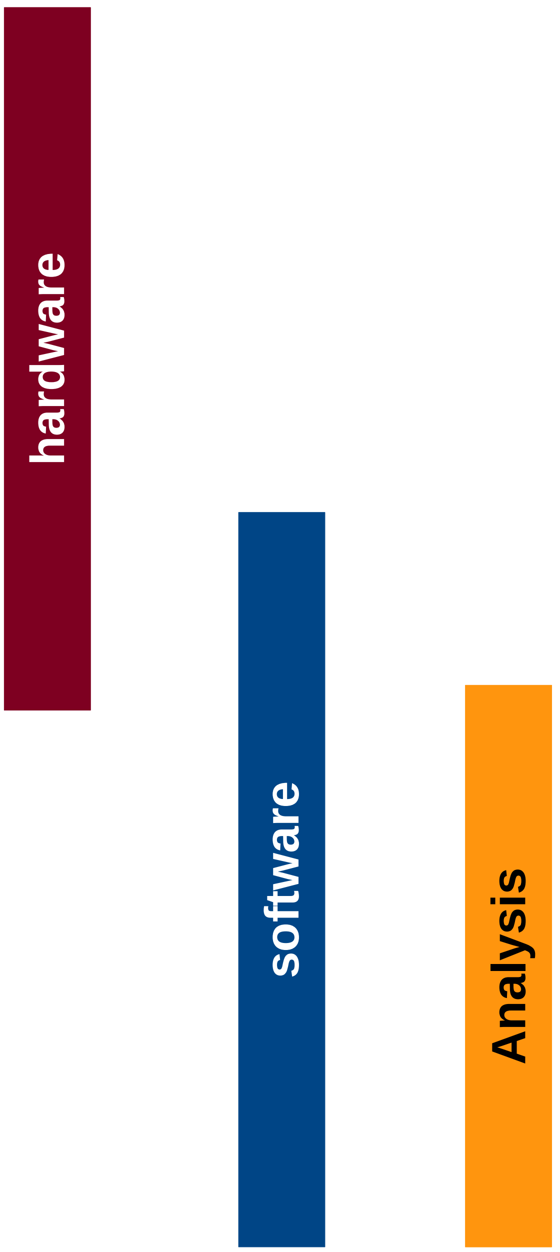
- **D**ata **A**c**Q**uisition (**DAQ**) is
  - the process of **sampling signals**
  - that **measure** real world physical conditions
  - and **converting** the resulting samples **into digital** numeric values that can be manipulated by a PC
- Components:
  - **Sensors**: convert physical quantities to electrical signals
  - **Analog-to-digital converters**: convert conditioned sensor signals to digital values
  - Processing and storage elements

What is DAQ?

- DAQ is an **heterogeneous** field
  - Boundaries not well defined
- An **alchemy** of
  - physics
  - electronics
  - computer science
  - networking
  - hacking
  - experience
- Money and manpower matter as well

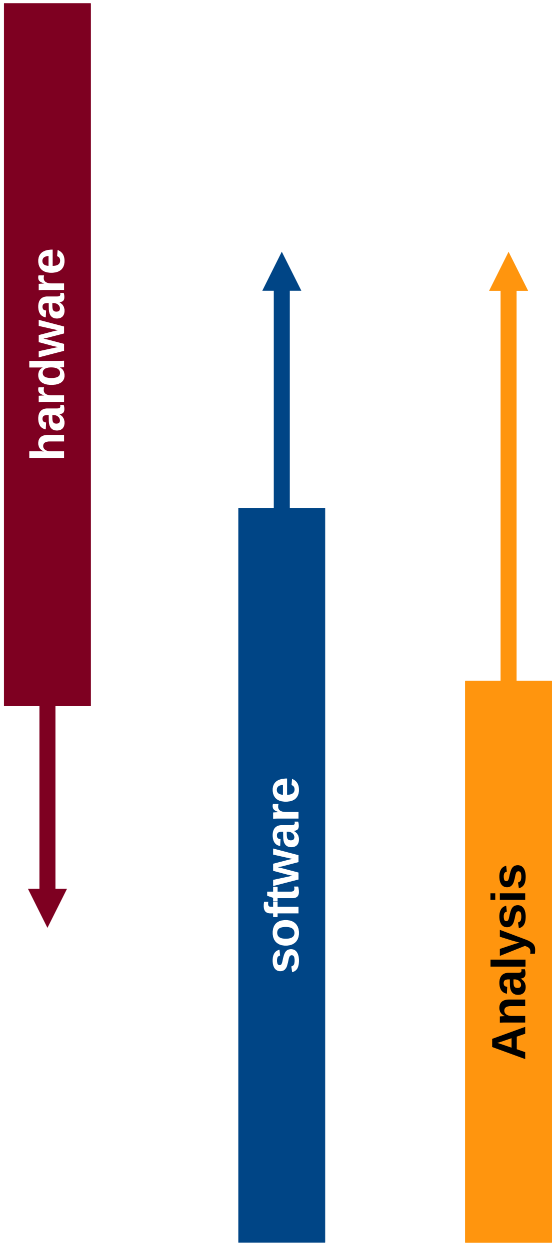
It’s all about physics

- Experiment/detector
  - Produces **physics**
- Data AcQuisition
  - Extracts **physics** from detector
- (Offline) Analysis
  - Extracts **physics** from data



It’s all about physics

- Experiment/detector
  - Produces **physics**
- Data AcQuisition
  - Extracts **physics** from detector
- (Offline) Analysis
  - Extracts **physics** from data

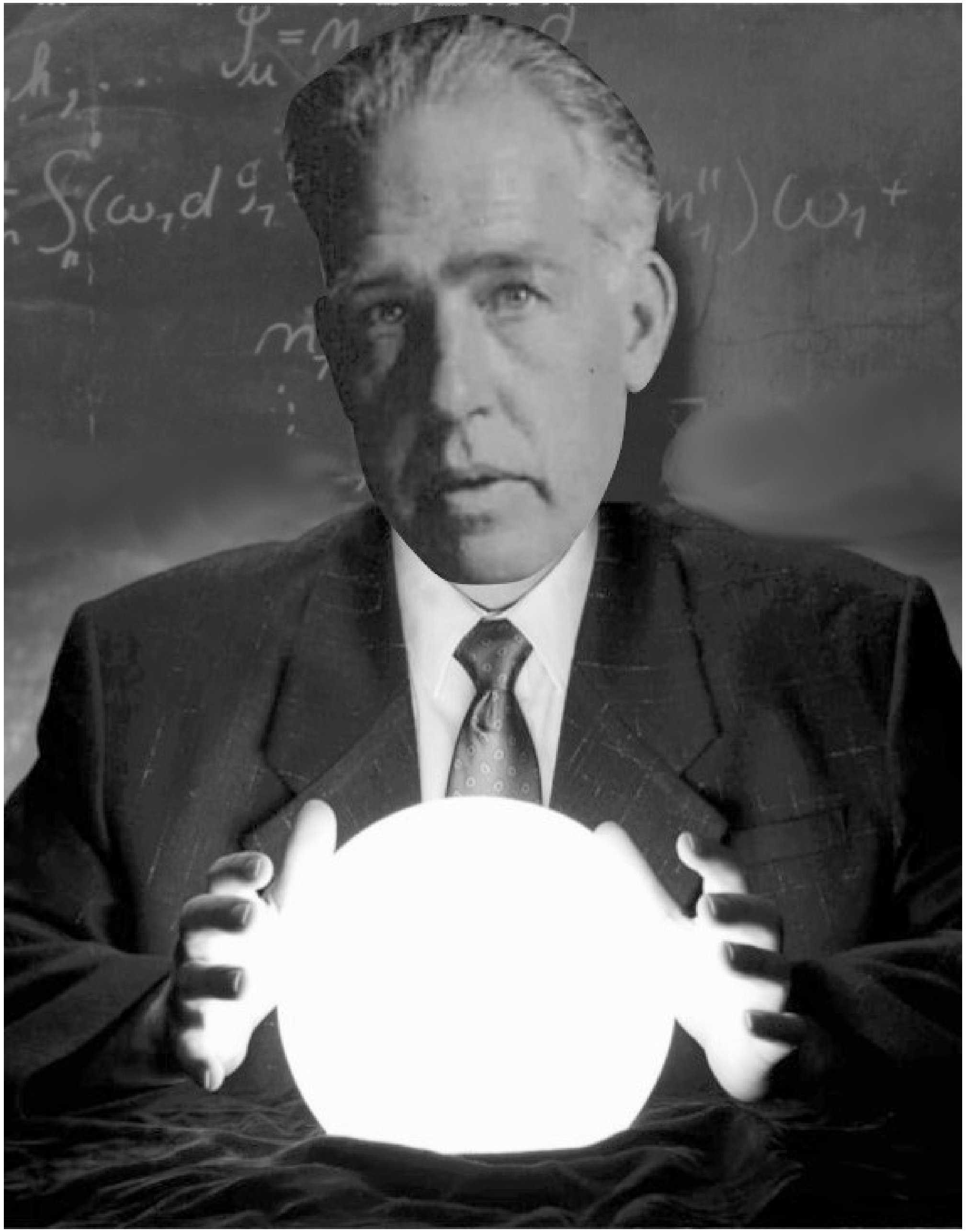


The role of tecnology

- Tecnology and Computing are **enablers** of physics programs
  - The **physics** goals **depend** on tecnology and innovation
- *Particle physicists must monitor technological trends and make innovation*
  - *Expecially true in DAQ field*
- *“Modern computing architectures and emerging technologies are changing the way we do particle physics”*
  [Snowmass 2022 report]

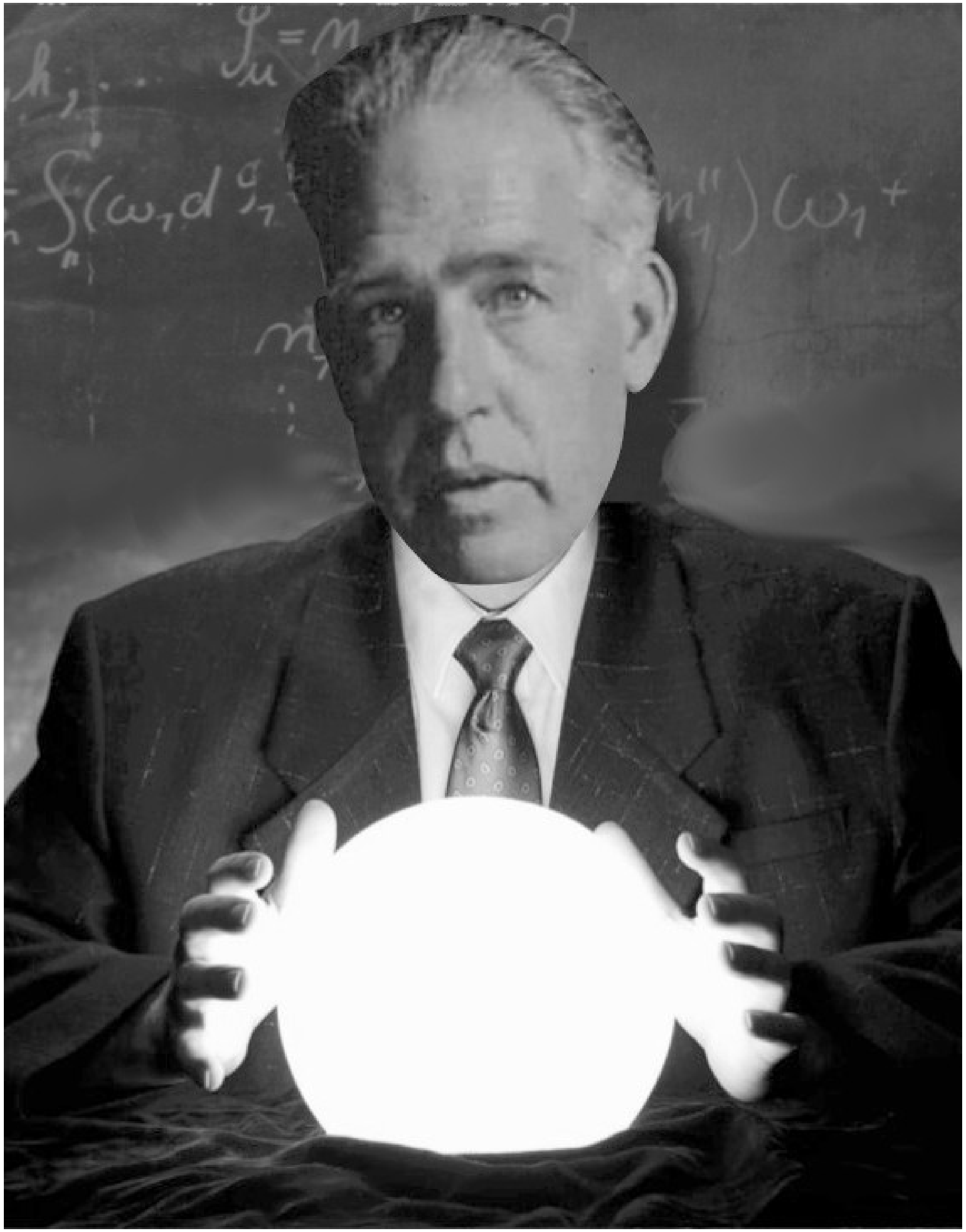
Predictions

- *"Prediction is very difficult, especially about the future."*
  - Niels Bohr



Predictions

- *“Prediction is very difficult, especially about the future.”*
  - Niels Bohr
- *“There is no reason anyone would want a computer at home.”*
  - Ken Olsen,
    Founder of DEC, 1977
- *"Apple is already dead"*
  - N. Myhrvold, former
    Microsoft CTO, 1997



Predictions HEP

- *“Thanks to the Moore law, in 2007 our event selection farm will be based on 8 GHz CPUs”*
  - ATLAS TDR, 2003

- *“Machine learning was essentially not a part of the 2013 Snowmass report”*
  - SnowMass report 2022

What about trigger?

Something interesting

- Main role of DAQ
  - process the signals generated in a detector
  - and saving the **interesting** information on a permanent storage
- What does it mean interesting?
  - When does this happen?
- We need a trigger

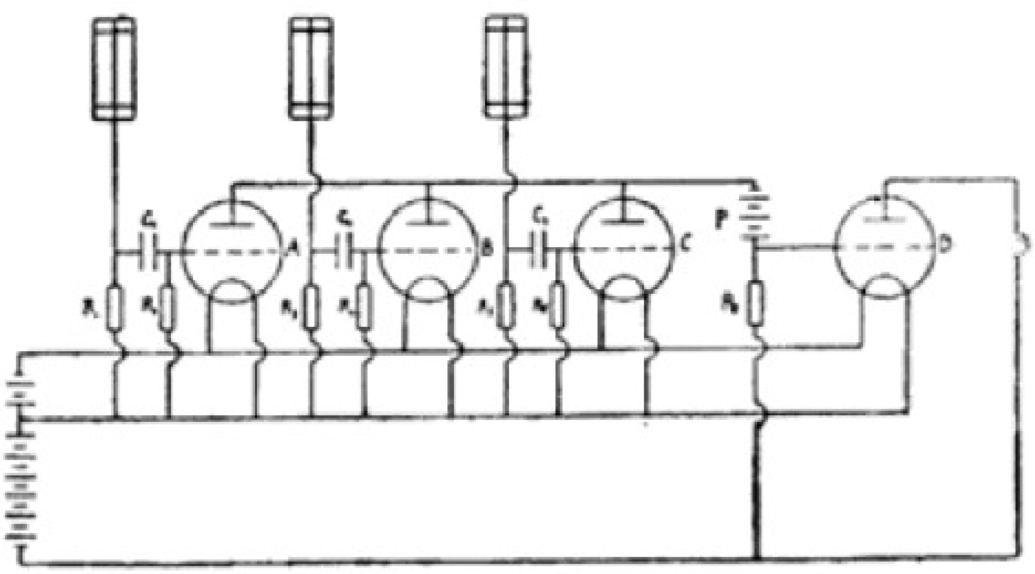
Trigger

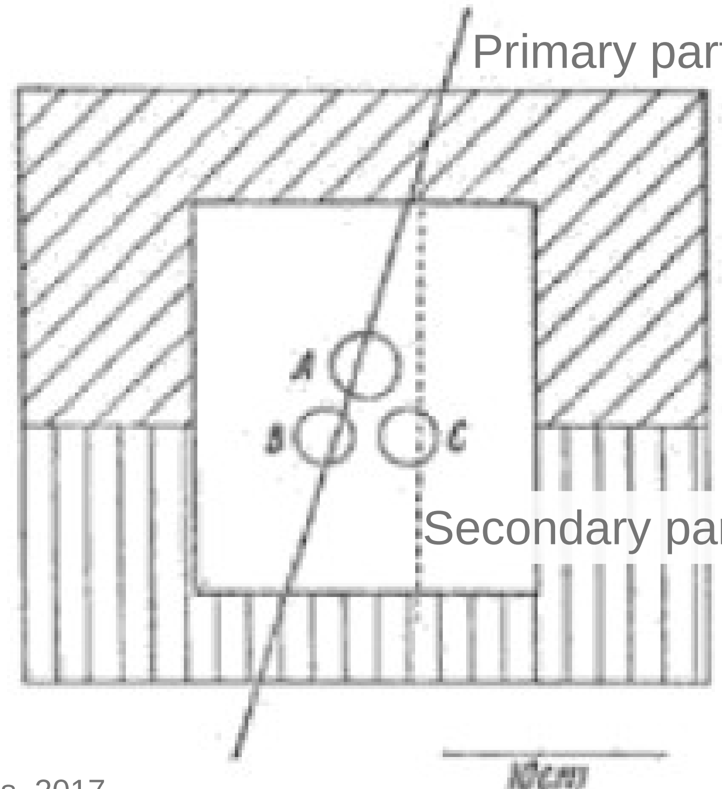
- *"Method of Registering Multiple Simultaneous Impulses of Several Geiger Counters"*

  Bruno Rossi, Nature 1930

  – Online coincidence of three signals

Geiger counters





Astronomia e Fisica a Firenze: dalla Specola ad Arcetri, Firenze Universiry Press, 2017

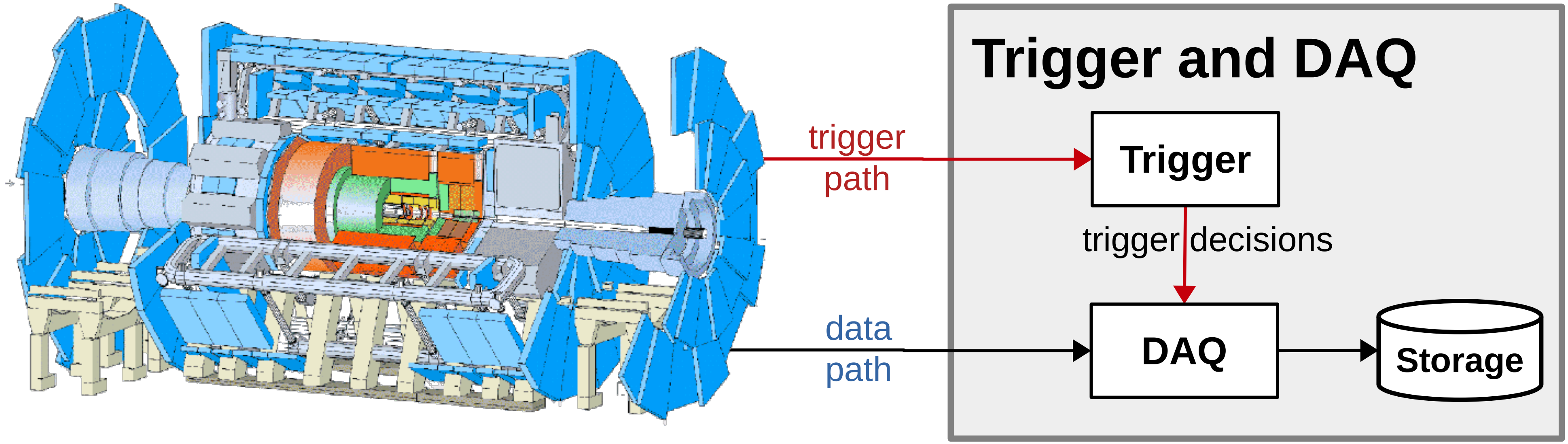
Trigger

- Either selects interesting events or rejects boring ones, in **real time**
  - **Selective**: efficient for "signal" and resistant to "background"
  - **Simple** and **robust**
  - **Quick**

- With minimal *controlled* **latency**
  - time it takes to form and distribute its decision
- The trigger system generates a prompt **signal** used to start the data-acquisition processes
  - To be distributed to front end electronics

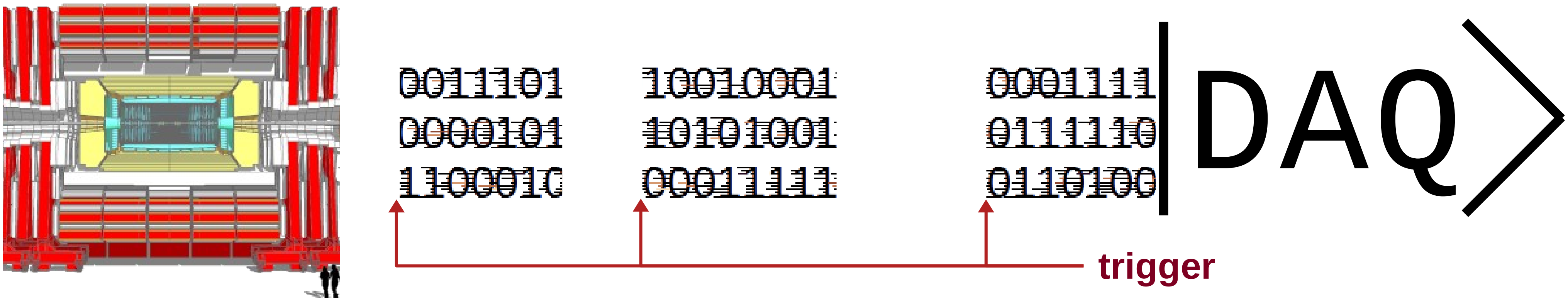
Double paths

- **Trigger** path
  - From dedicated detectors to trigger logic
- **Data** path
  - From all the detectors to storage
  - On positive trigger decision



Trigger(less)

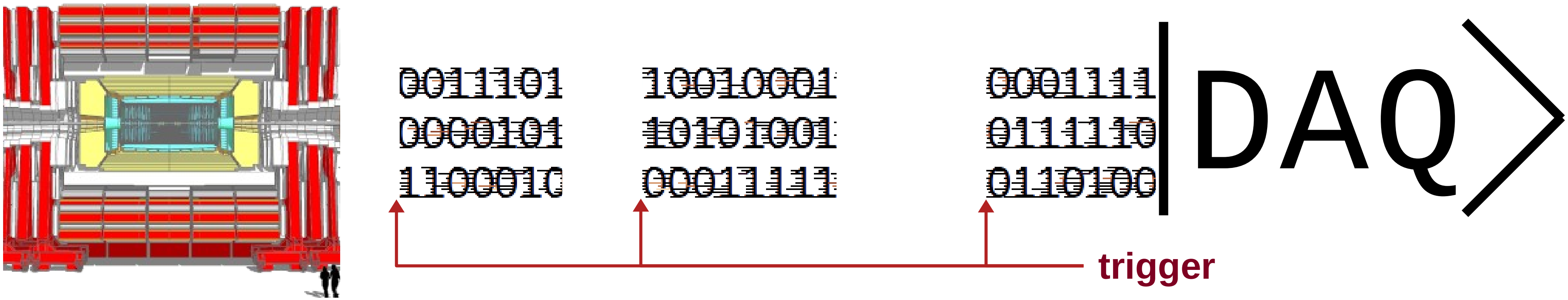
- **Triggered**: data is readout from detector only when a trigger signal is raised



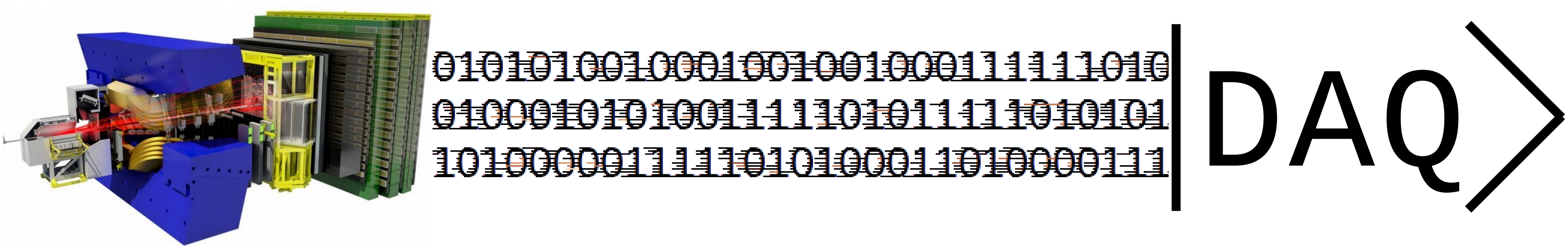
- **Triggerless**: the detector push data at its speed and the downstream daq must keep the pace

Trigger(less)

- **Triggered**: data is readout from detector only when a trigger signal is raised



- **Triggerless**: the detector push data at its speed and the downstream daq must keep the pace



trigger@isotdaq2024

- *Introduction to trigger*
  - **Gokhan** Unel
- *Trigger HW*
  - **Dinyar** Rabady
- *Timing in DAQ*
  - **Mehmet** Ozgur Sahin

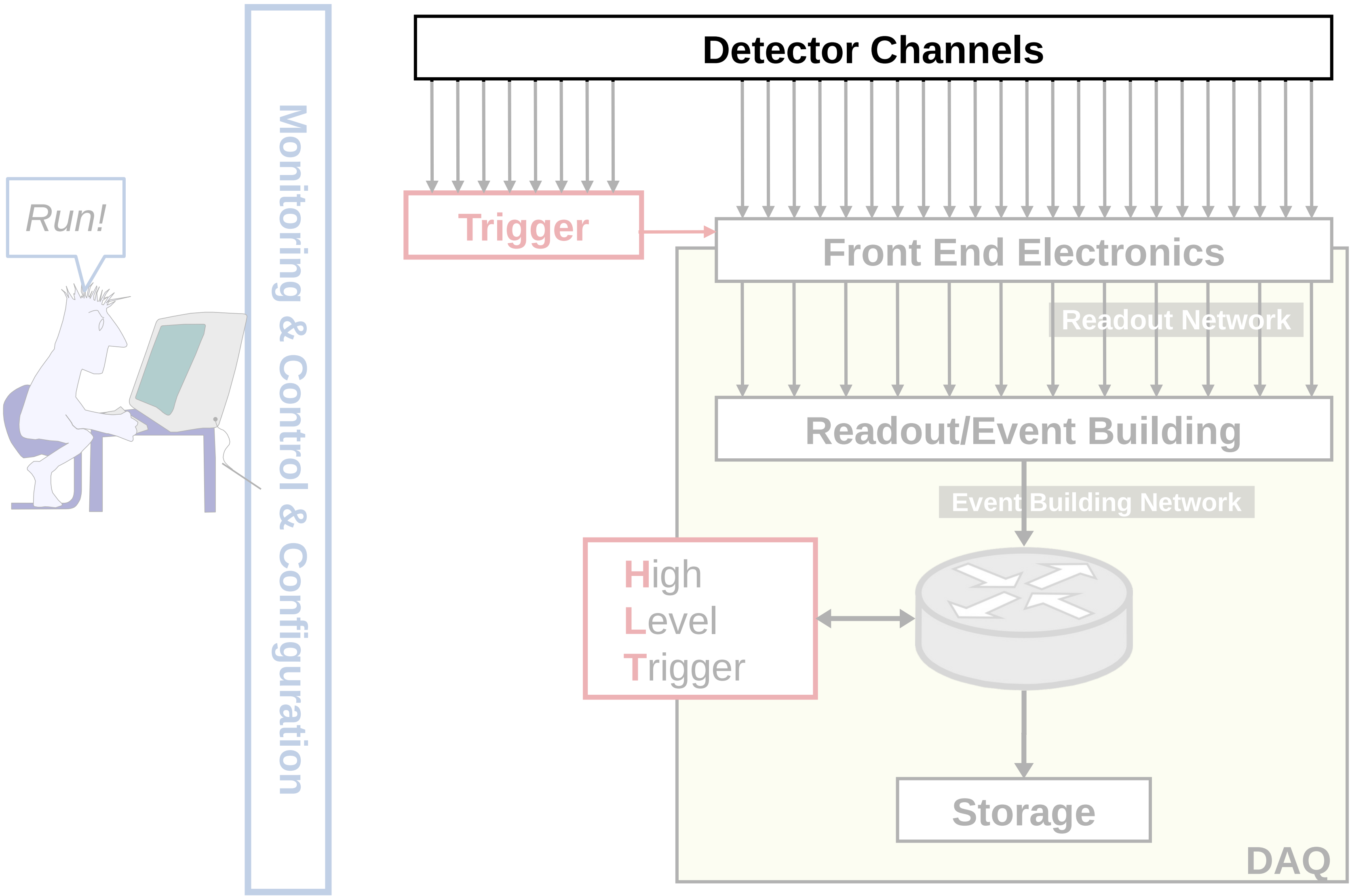
DAQ duties

- Gather data produced by detectors
  - **Readout**
- Form complete events
  - **Data Collection** and **Event Building**
- Possibly feed other trigger levels
  - **H**igh **L**evel **T**rigger
- Store event data
  - **Data Logging**

Data Flow

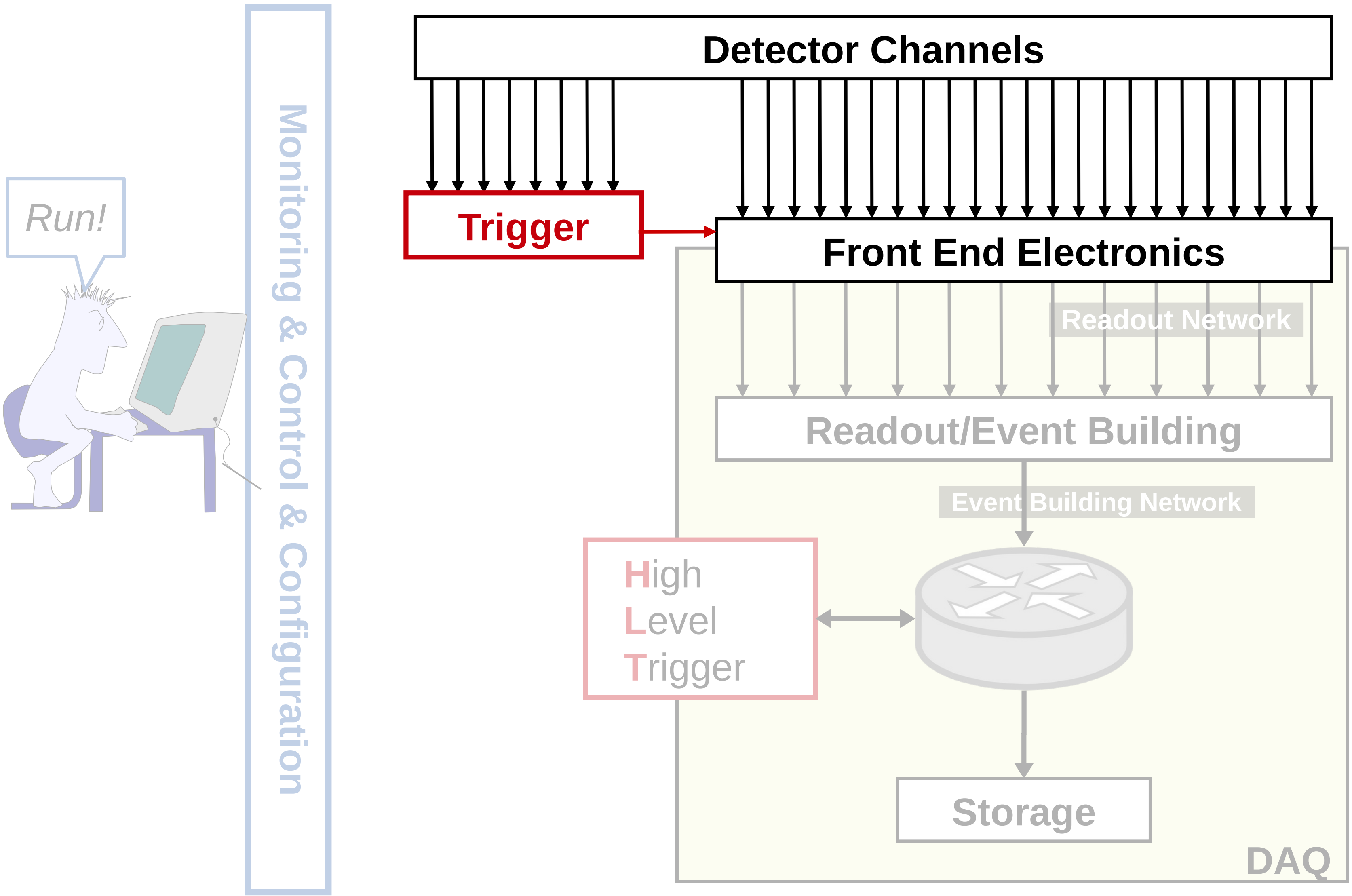
- Manage the operations
  - **Run Control**, **Configuration**, **Monitoring**

T-DAQ



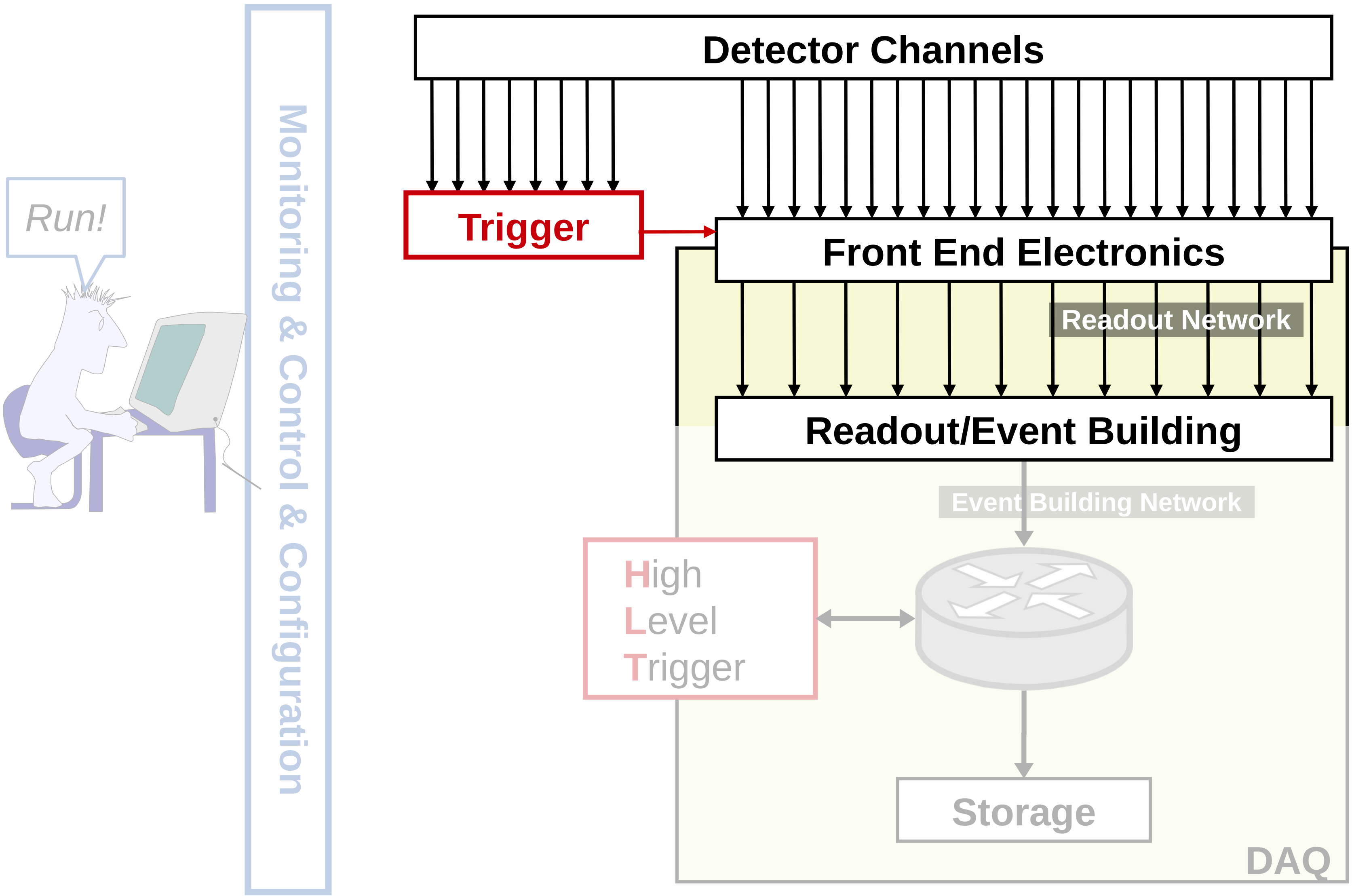
Run!
Monitoring & Control & Configuration
Detector Channels
Trigger
Front End Electronics
Readout Network
Readout/Event Building
Event Building Network
High
Level
Trigger
Storage
DAQ

T-DAQ



Run!
Monitoring & Control & Configuration
Detector Channels
Trigger
Front End Electronics
Readout Network
Readout/Event Building
Event Building Network
High
Level
Trigger
Storage
DAQ

T-DAQ



Run!
Monitoring & Control & Configuration
Detector Channels
Trigger
Front End Electronics
Readout Network
Readout/Event Building
Event Building Network
High
Level
Trigger
Storage
DAQ

DAQHW@isotdaq2024

- *Detector Control Systems*
  - **Paris** Moschovakos
- *Introduction to detector readout*
  - **Gokhan** Unel
- *Frontend electronics & ADC's*
  - **Suerfu** Burkhant
- *Optical Links*
  - **Paolo** Durante

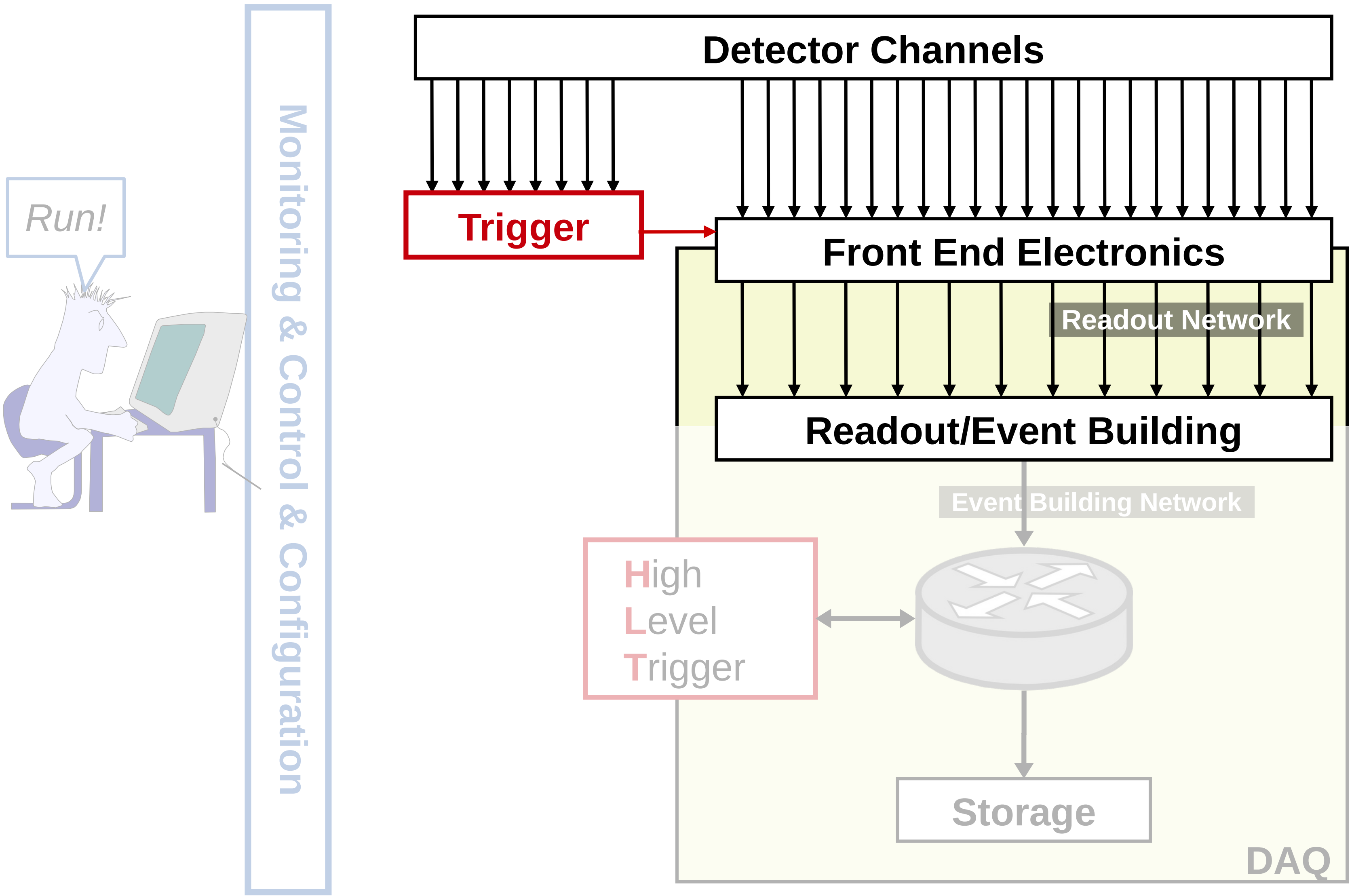
FPGA@isotdaq2024

- FPGAs are ~~becoming~~ the bred&butter of TDAQ
  - Signal processing, data formatting, parallelizable tasks (pattern recognition), machine learning, ...
- *Introduction to **FPGAs***
  - **Hannes** Sakulin
- *Advanced **FPGA** programming*
  - **Mauricio** Feo
- *Precision Time-to-Digital Conversion based on **FPGA***
  - **Jinhong** Wang

FPGA@isotdaq2024

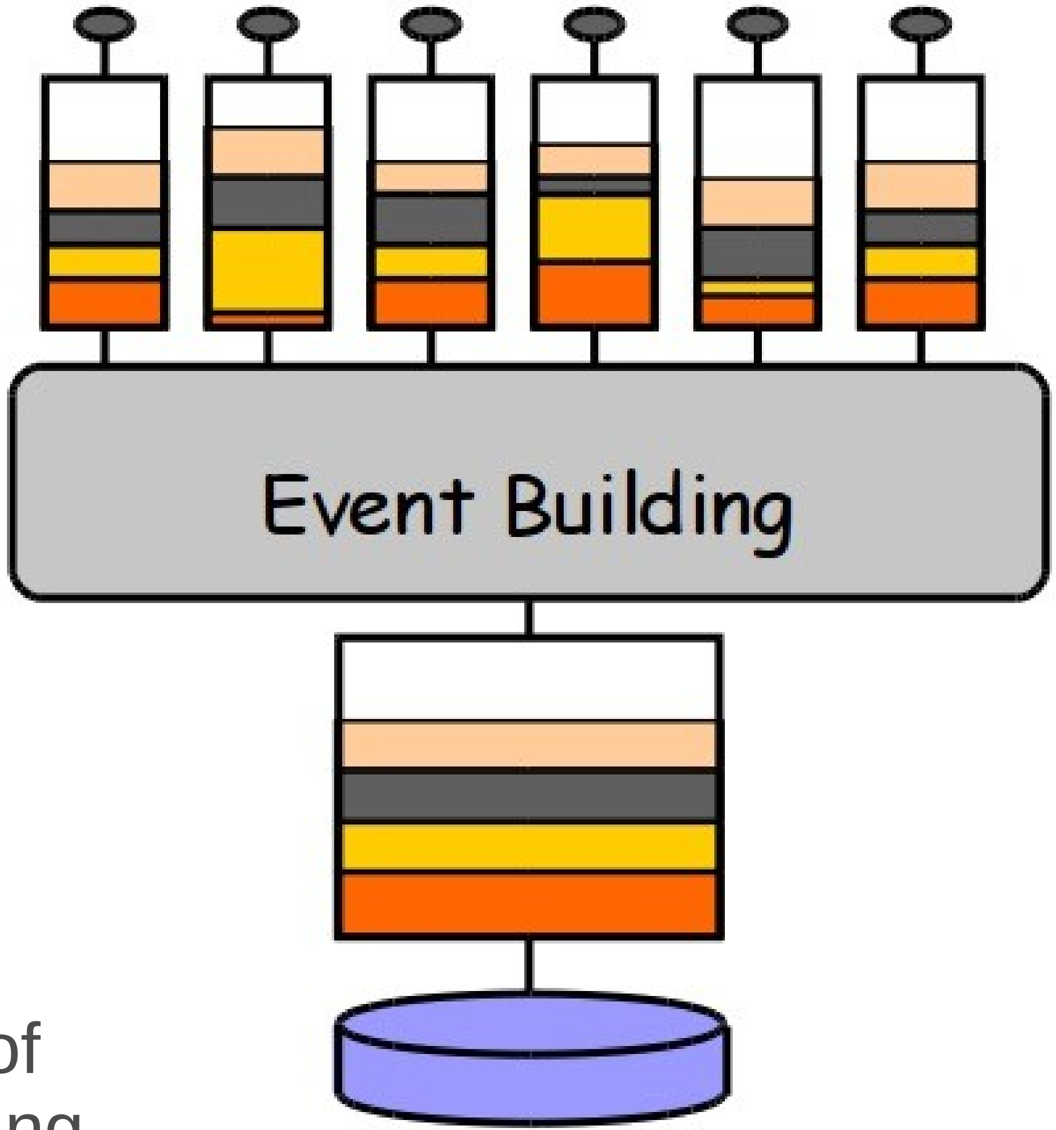
- FPGAs are ~~becoming~~ the bred&butter of TDAQ
  - Signal processing, data formatting, parallelizable tasks (pattern recognition), machine learning, ...
- ***FPGA** programming*
  - Lab 5
- *System on Chip (SoC)* ***FPGA***
  - Lab 13

T-DAQ

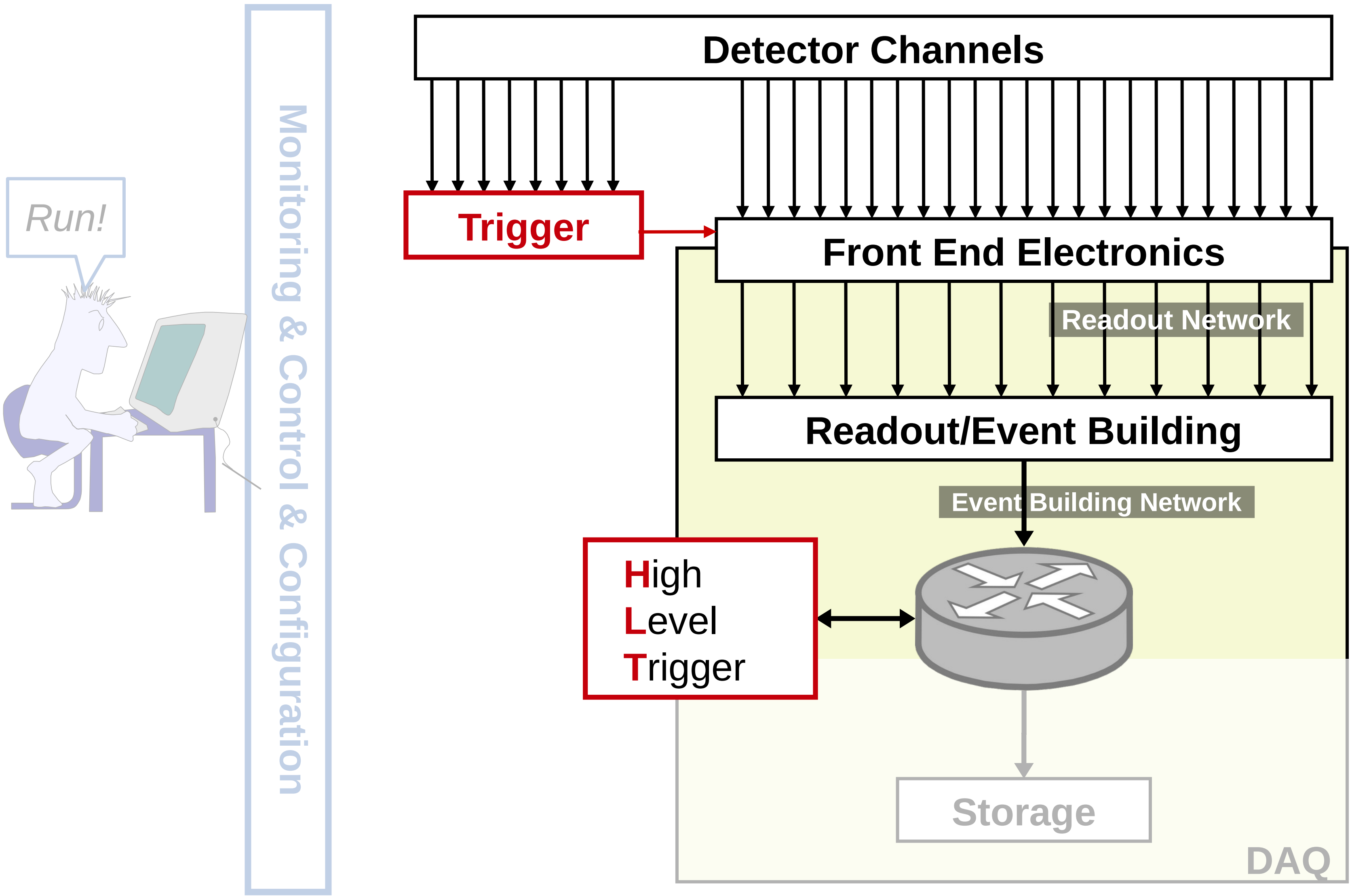

Run!
Monitoring & Control & Configuration
Detector Channels
Trigger
Front End Electronics
Readout Network
Readout/Event Building
Event Building Network
High
Level
Trigger
Storage
DAQ

Event Building

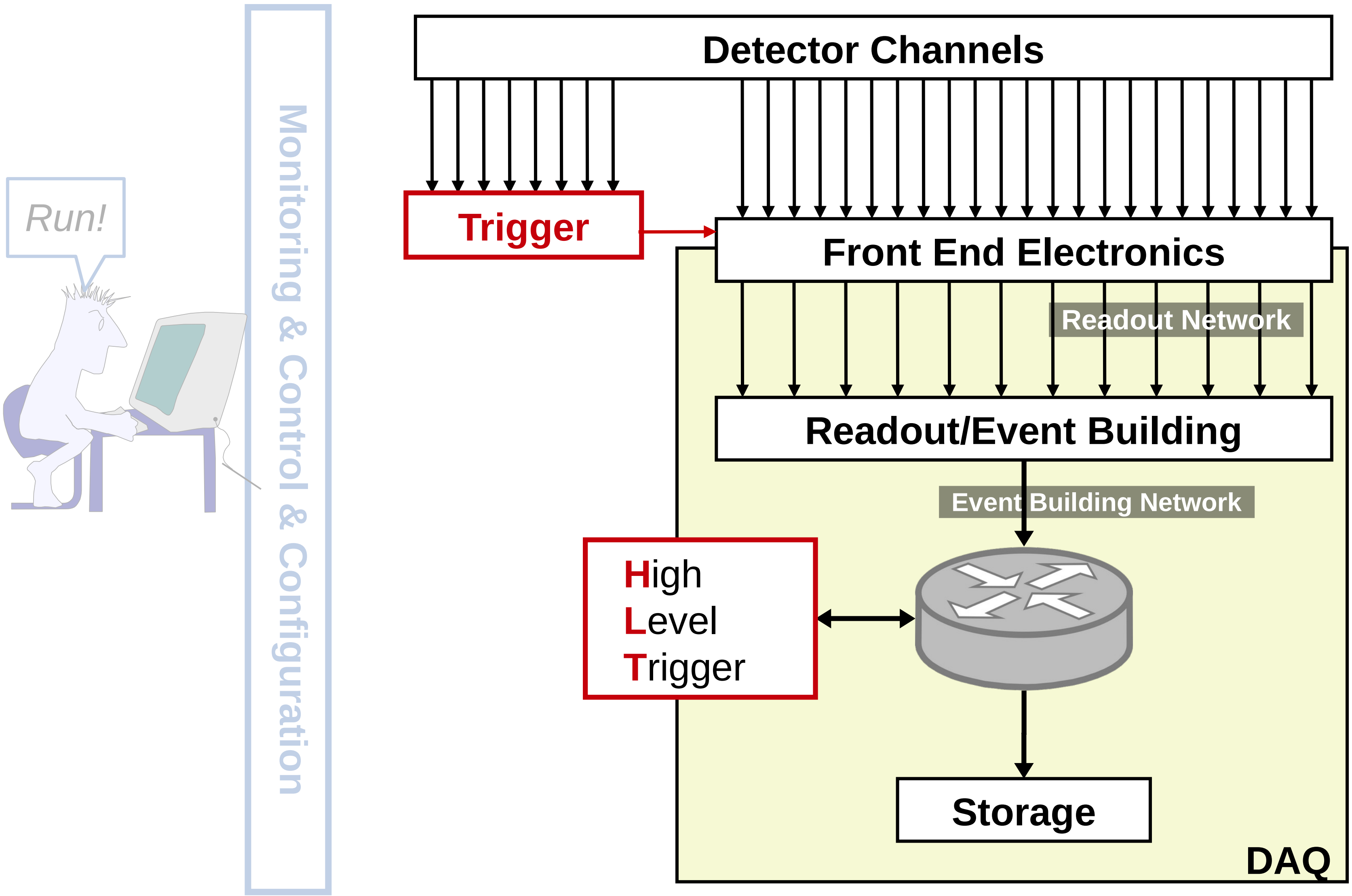
- Associate all **data** corresponding to the **same event**
  - E.g.: to the same bunch crossing
- Or collect all data corresponding to the same time frame (ALICE) or the same LHC orbit (CMS)
  - But you still need to associate all the data of the same bunch crossing



T-DAQ


Run!
Monitoring & Control & Configuration
Detector Channels
Trigger
Front End Electronics
Readout Network
Readout/Event Building
Event Building Network
High
Level
Trigger
Storage
DAQ

T-DAQ

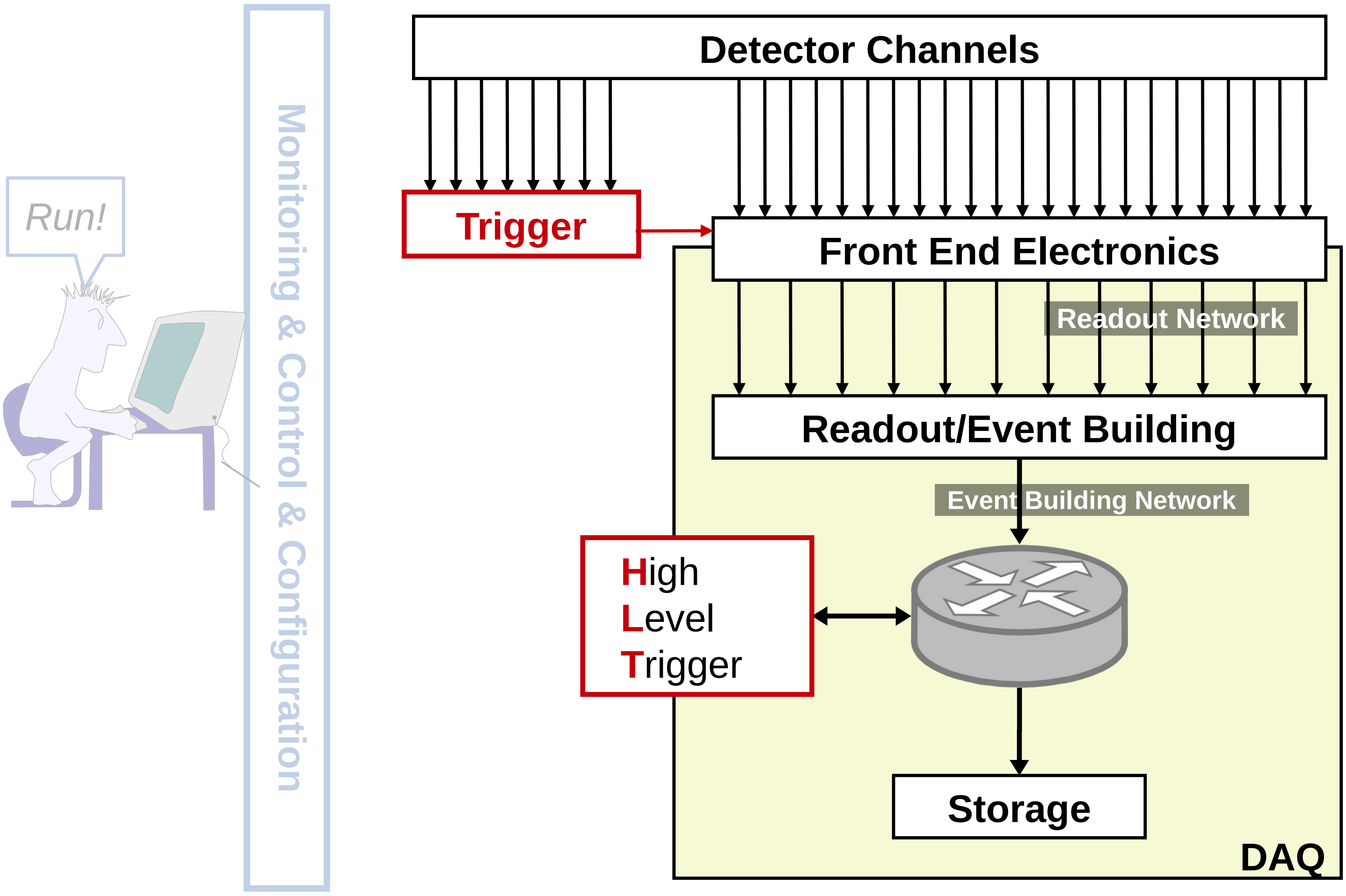


Run!
Monitoring & Control & Configuration
Detector Channels
Trigger
Front End Electronics
Readout Network
Readout/Event Building
Event Building Network
High
Level
Trigger
Storage
DAQ

Storage@isotdaq2024

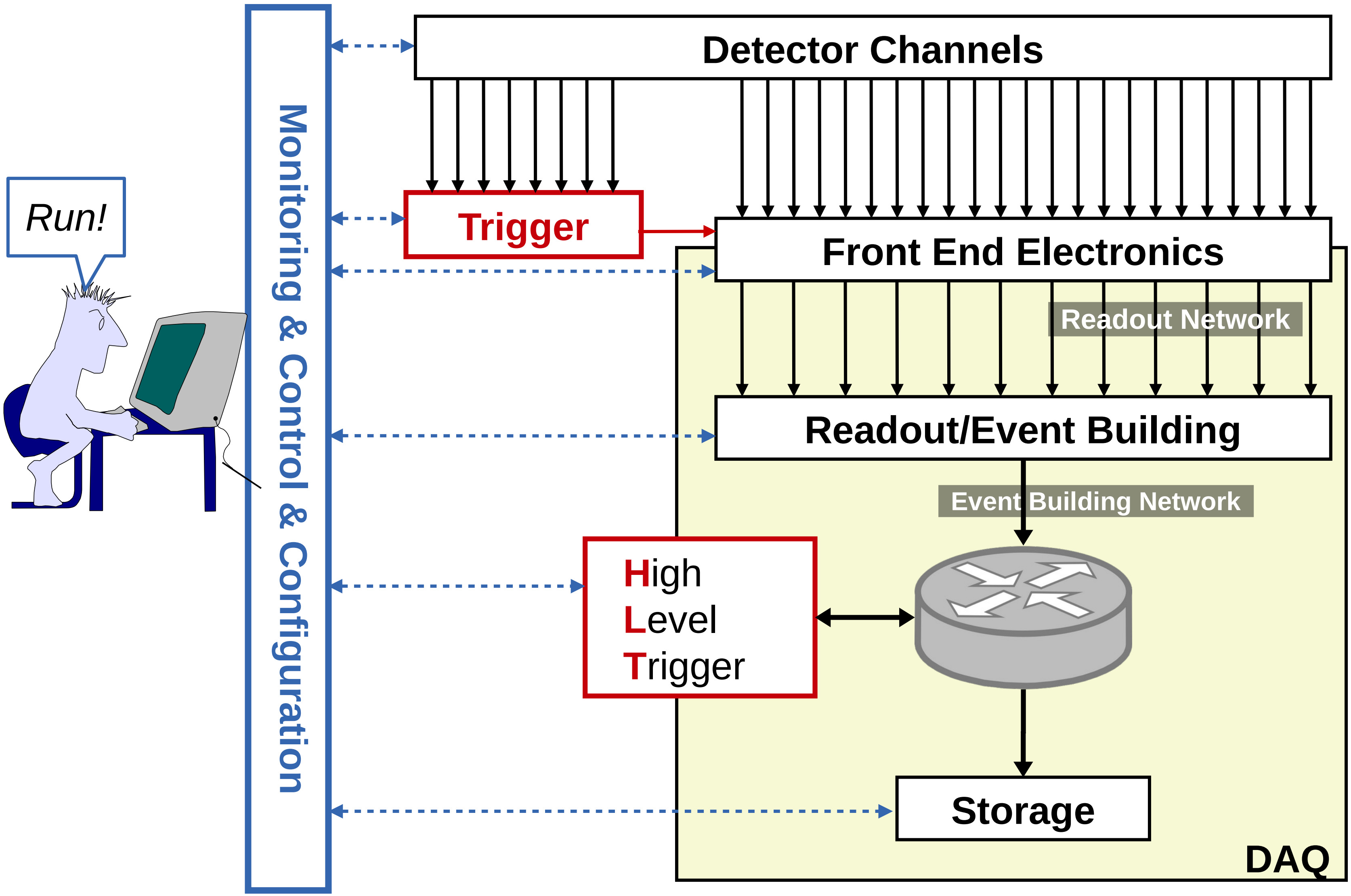
- Storage device technologies gaining importance in HEP
  - Storage data rate increasing with luminosity
  - Distributed file systems being used as data-flow frameworks
    - CMS, ATLAS run 4 (?), ...
- *Storage*
  - **Enrico** Gamberini
- *Storage lab*
  - lab 11

T-DAQ



Run!
Monitoring & Control & Configuration
Detector Channels
Trigger
Front End Electronics
Readout Network
Readout/Event Building
Event Building Network
High
Level
Trigger
Storage
DAQ

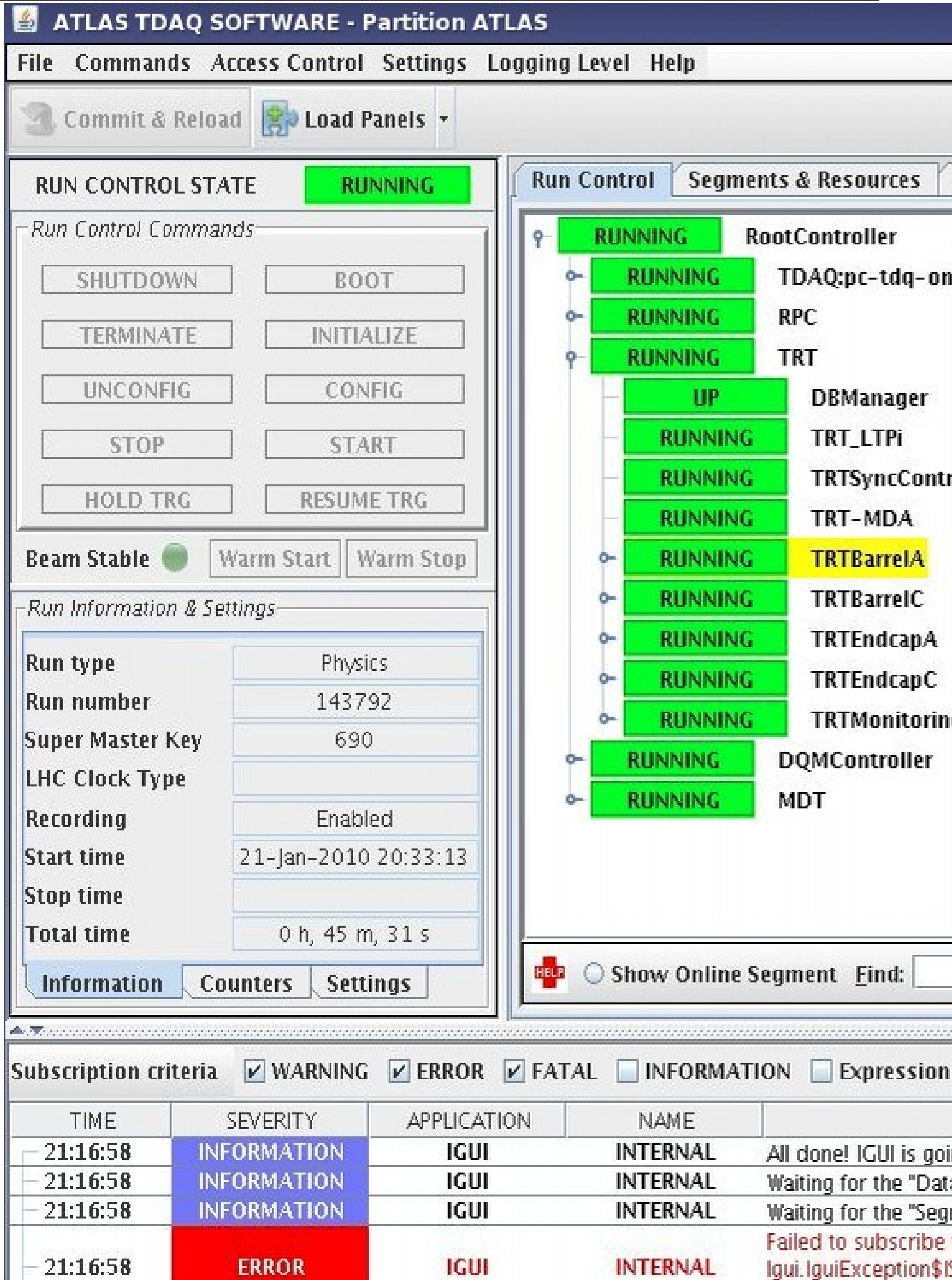
T-DAQ



Run!
Monitoring & Control & Configuration
Detector Channels
Trigger
Front End Electronics
Readout Network
Readout/Event Building
Event Building Network
High
Level
Trigger
Storage
DAQ

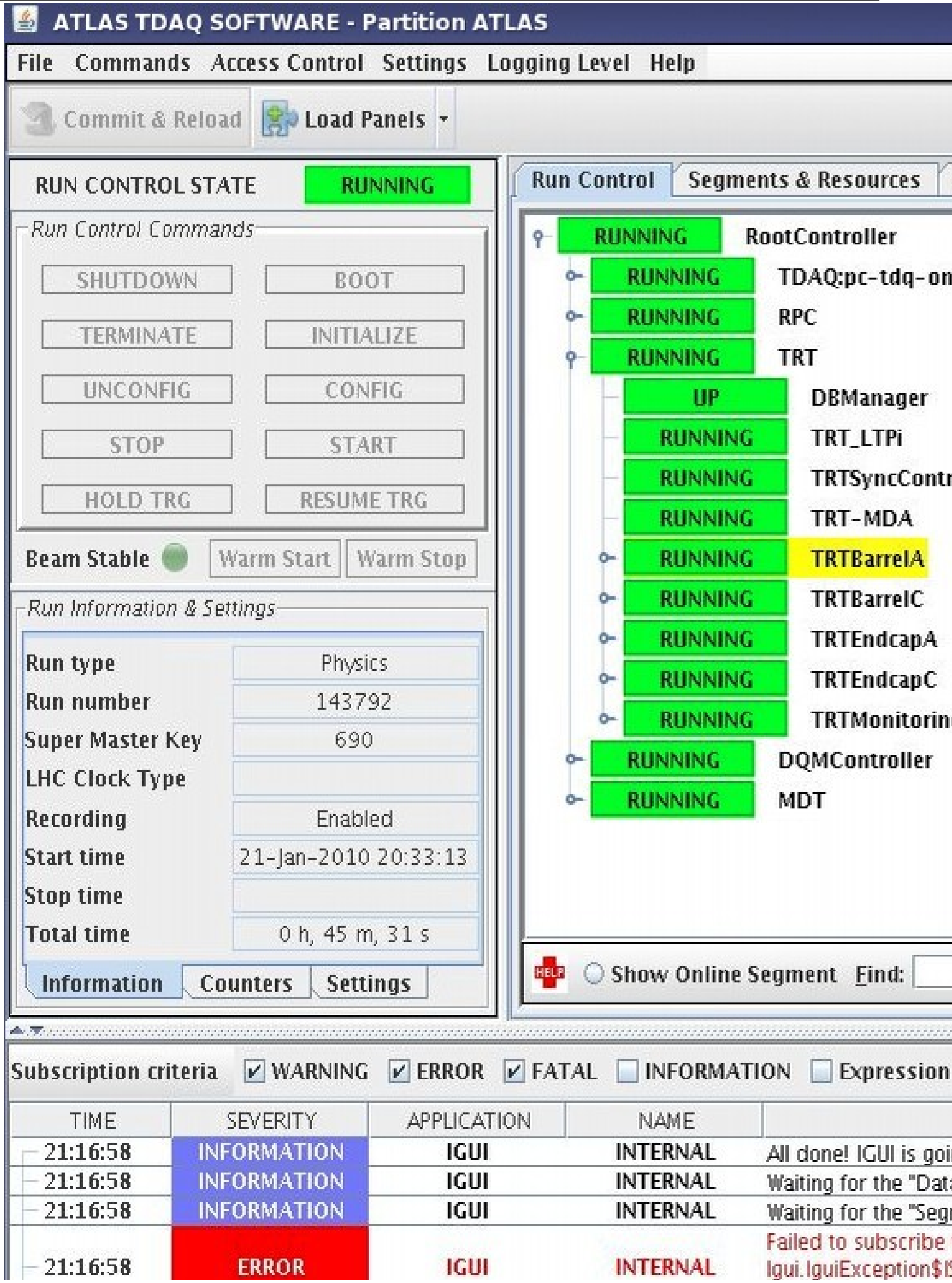
The glue of your experiment

- Configuration
  - The data taking setup
- Control
  - Orchestrate applications participating to data taking
  - Via distributed **F**inite **S**tate **M**achine
- Monitoring of data taking operations
  - What is going on?
  - What happened? When? Where?



The glue of your experiment

- *Control of DAQ. DAQ Online Software*
  - Lab 12
- *Design and implementation of a monitoring system*
  - **Serguei** Kolos
- *Monitoring with Prometheus and Grafana*
  - **Camilo** Carrillo



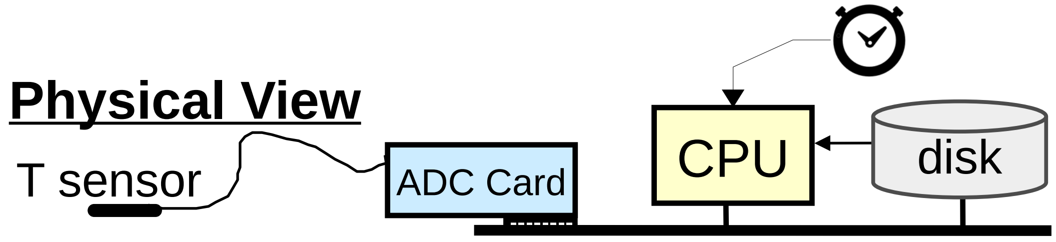
Outline

- Introduction
  - What is DAQ?
  - Overall framework
- Basic DAQ concepts
  - Digitization, Latency
  - Deadtime, Busy, Backpressure
  - De-randomization
- Scaling up
  - Readout and Event Building
  - Buses vs Network
- Data encoding

Via a toy model

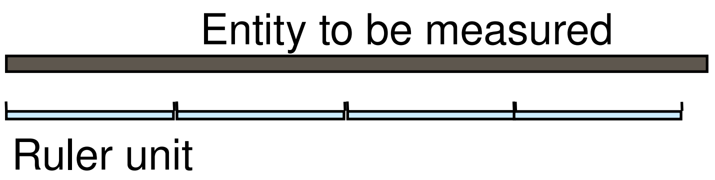
Basic DAQ: periodic trigger

- Eg: measure temperature at a fixed frequency
  - Clock trigger
- CPU does
  - Readout, Processing, Storage
- ADC performs analog to digital conversion, **digitization** (our front-end electronics)
  - Encoding analog value into binary representation



Digitization

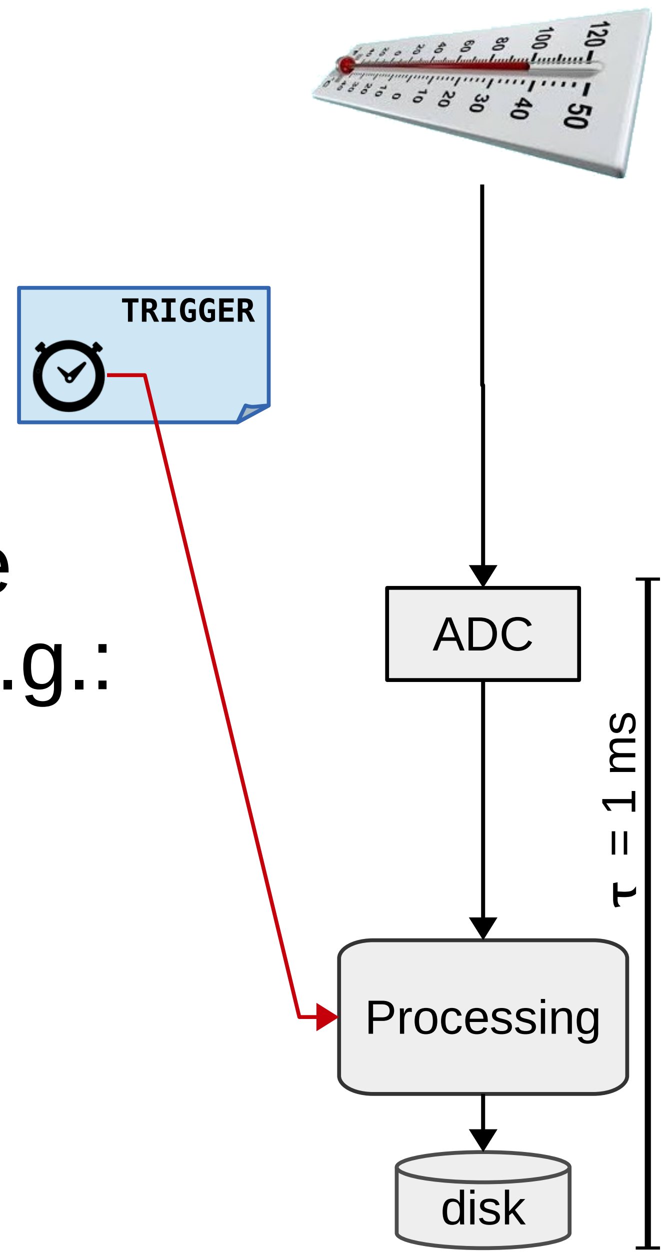
- Encoding an analog value into binary representation
  - Comparing entity with a ruler

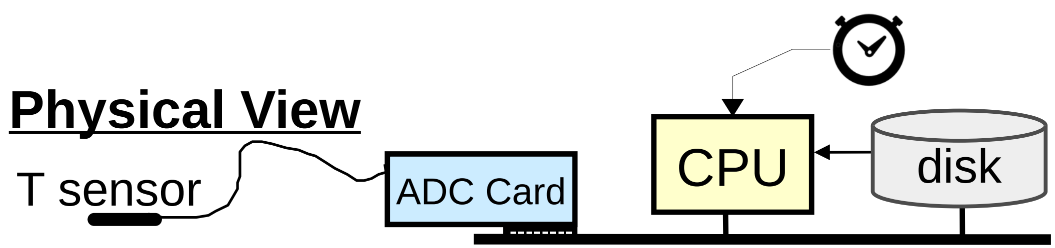


- We will see
  - **ADC**: Analog to Digital Converter
  - **QDC**: Charge to Digital Converter
  - **TDC**: Time to Digital Converter
- *DAQ HW*
  - **Vincenzo** Izzo
- *ADC basics for TDAQ*
  - Lab 8

Basic DAQ: periodic trigger

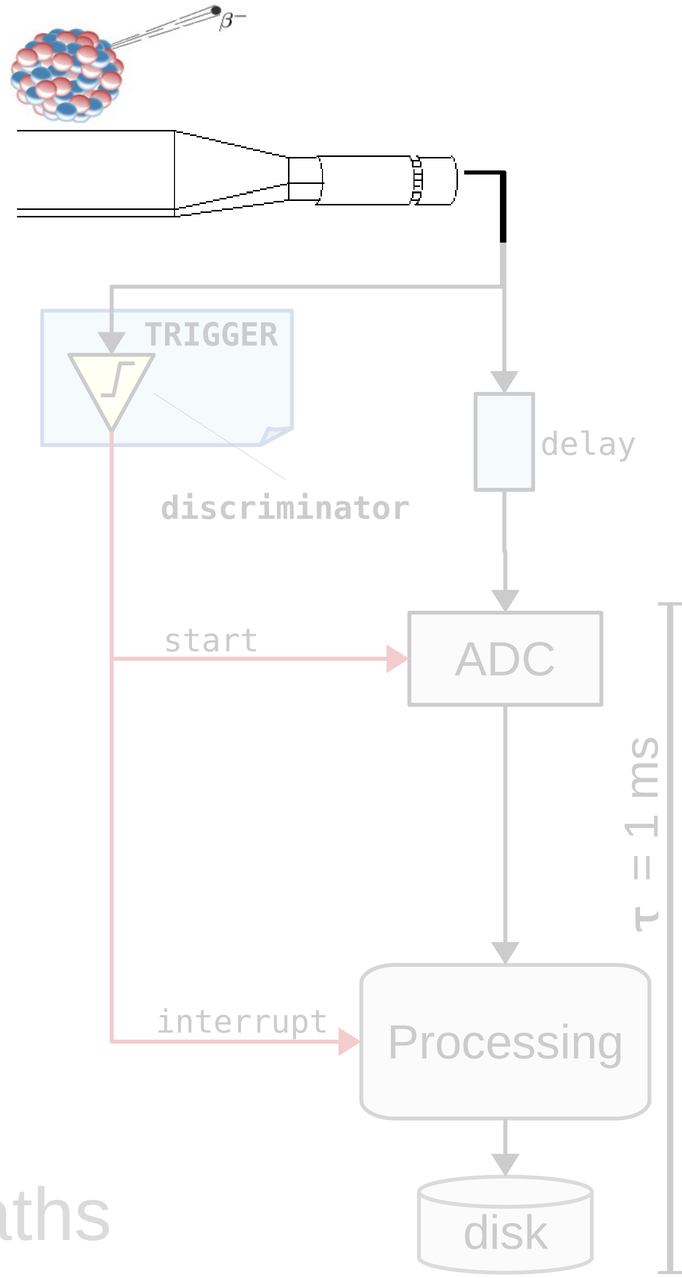
- System clearly limited by the time τ to process an "event"
  - ADC conversion + CPU processing + Storage
- The DAQ maximum sustainable rate is simply the inverse of τ, e.g.:
  - E.g.: $\tau = 1$ ms $\rightarrow R = 1/\tau = 1$ kHz





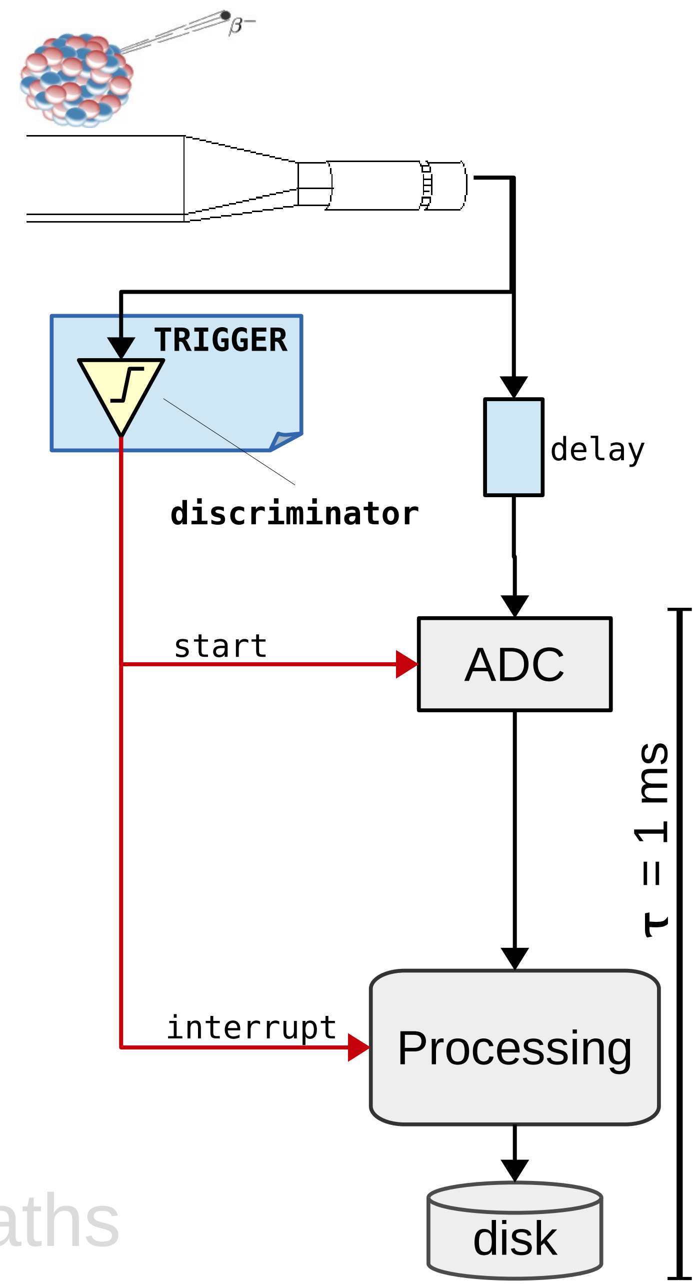
Basic DAQ: “real” trigger

- Events asynchronous and unpredictable
  - E.g.: beta decay studies
- A physics trigger is needed
  - **Discriminator**: generates an output digital signal if amplitude of the input pulse is greater than a given threshold
- NB: delay introduced to compensate for the **trigger latency**
  - Signal split in trigger and data paths



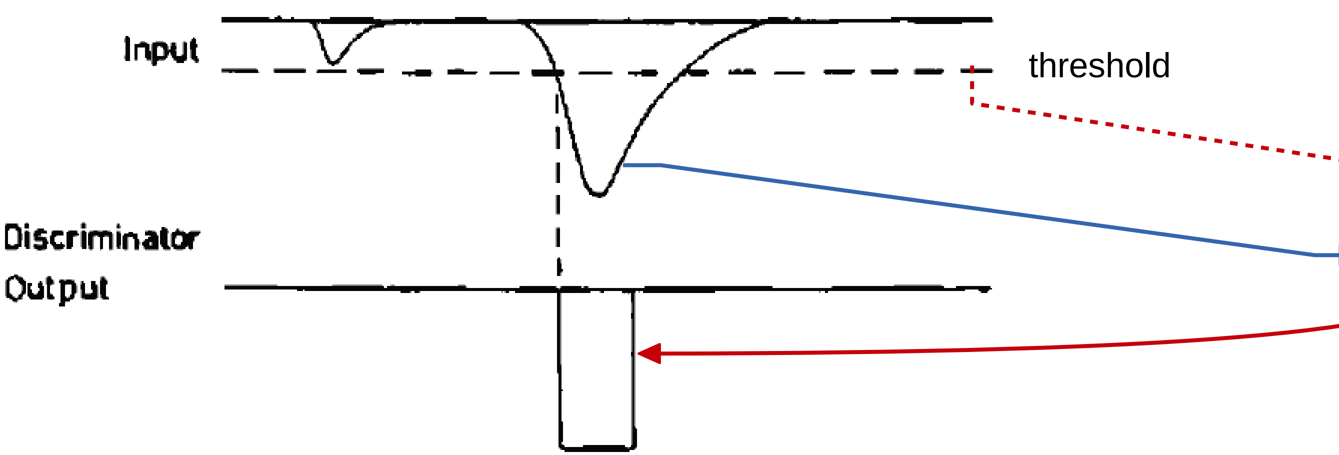
Basic DAQ: "real" trigger

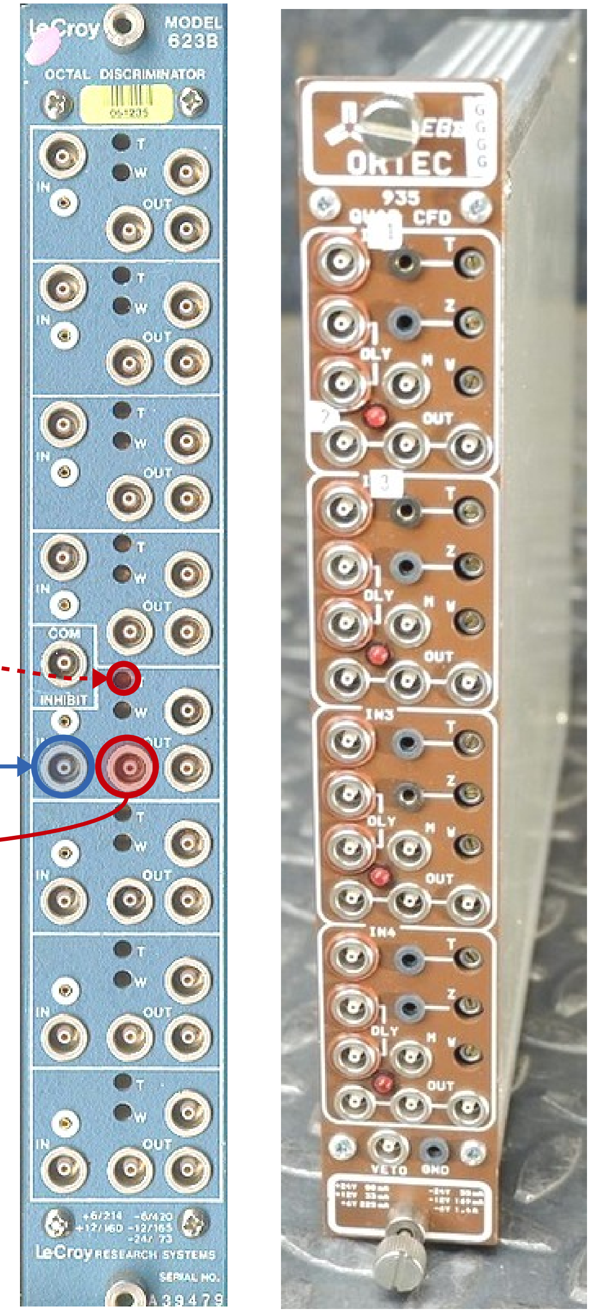
- Events asynchronous and unpredictable
  - E.g.: beta decay studies
- A physics trigger is needed
  - **Discriminator**: generates an output digital signal if amplitude of the input pulse is greater than a given threshold
- NB: delay introduced to compensate for the **trigger latency**
  - Signal split in trigger and data paths



Discriminator

- Discriminator:
  - generates a digital output signal
  - if the amplitude of the input pulse is greater than a given **threshold**

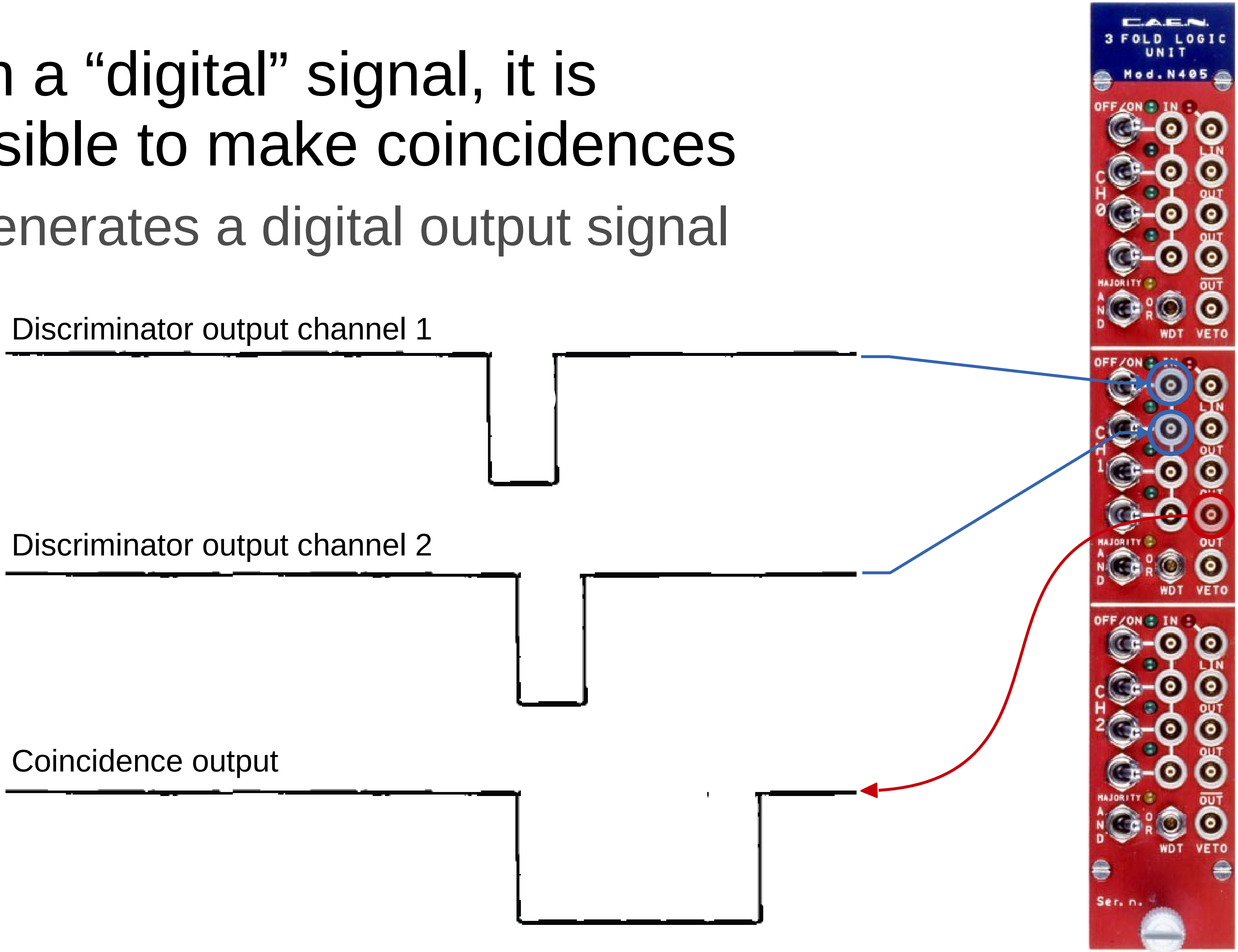




- In lab 2, 3, 4 we will see a couple of NIM discriminators

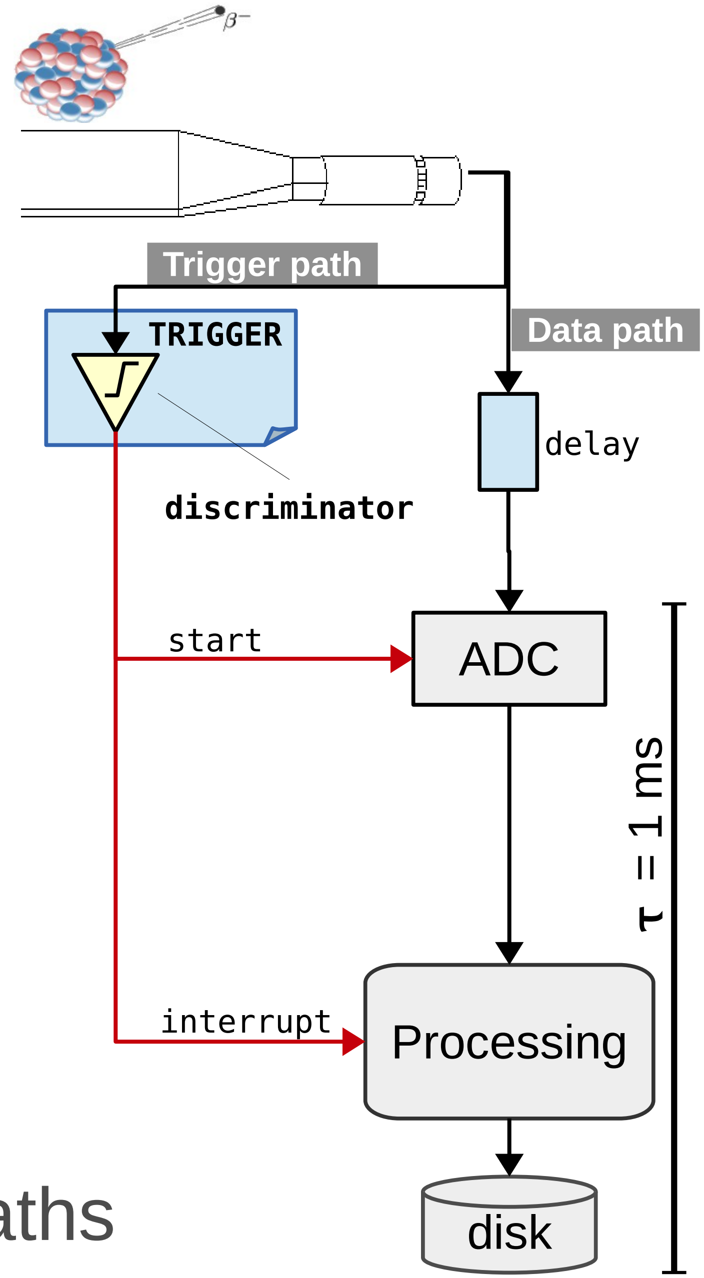
Coincidences

- With a “digital” signal, it is possible to make coincidences
  - generates a digital output signal



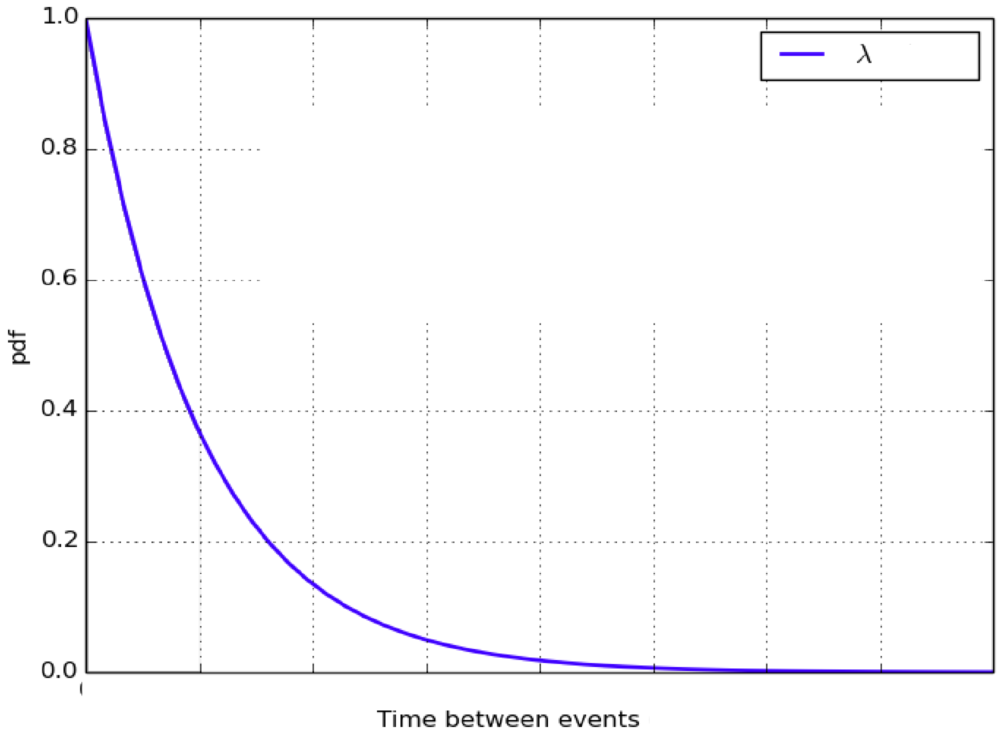
Basic DAQ: “real” trigger

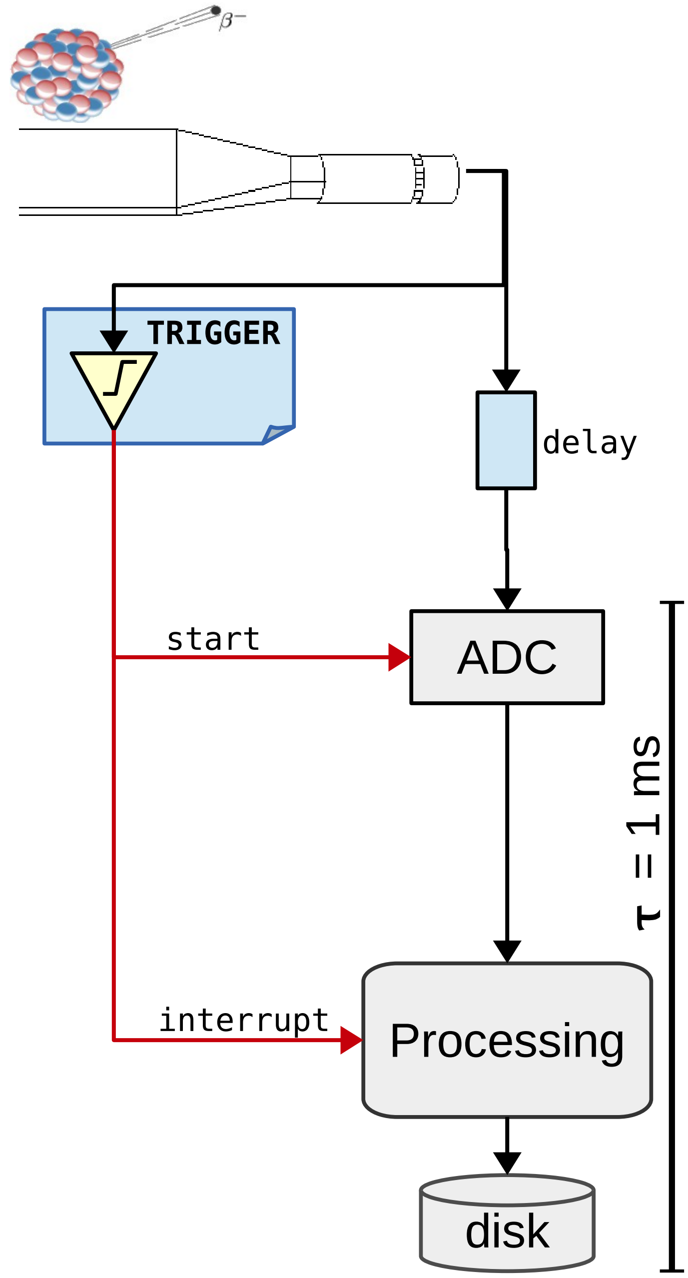
- Events asynchronous and unpredictable
  - E.g.: beta decay studies
- A physics trigger is needed
  - **Discriminator**: generates an output digital signal if amplitude of the input pulse is greater than a given threshold
- NB: delay introduced to compensate for the **trigger latency**
  - Signal split in trigger and data paths



Basic DAQ: “real” trigger

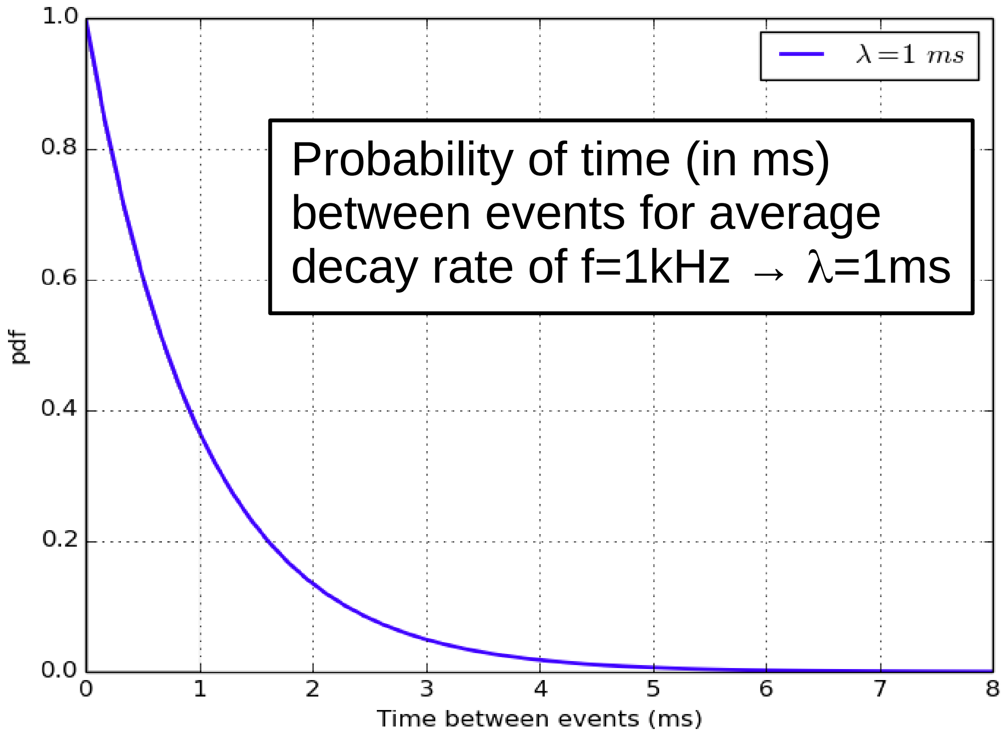
- Stochastic process
  - Fluctuations in time between events
- Let's assume for example
  - physics rate f = 1 kHz, i.e. λ = 1 ms
  - and, as before, τ = 1 ms

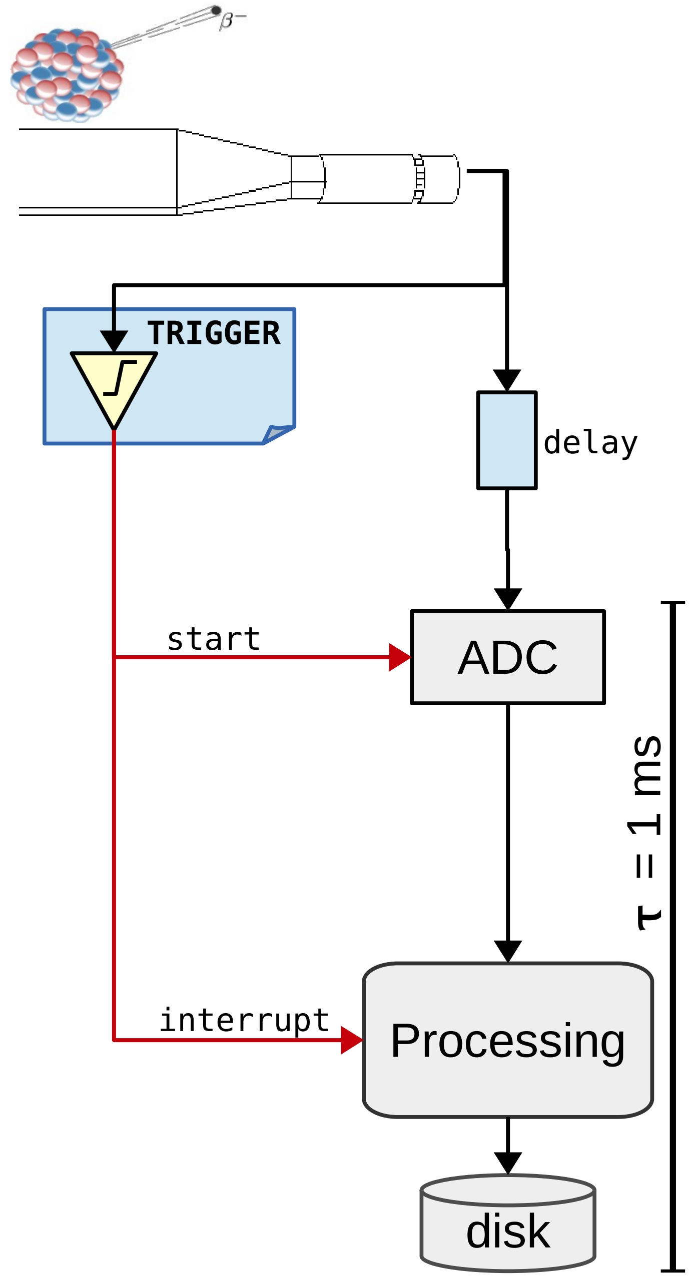




Basic DAQ: “real” trigger

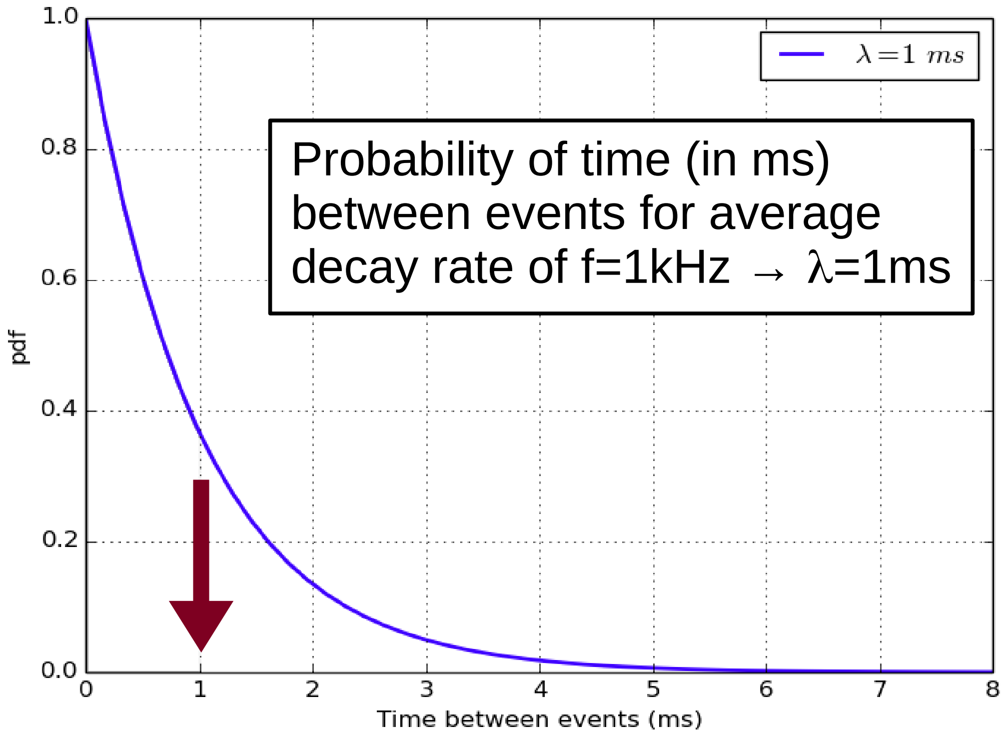
- Stochastic process
  - Fluctuations in time between events
- Let's assume for example
  - physics rate f = 1 kHz, i.e. $\lambda = 1$ ms
  - and, as before, $\tau = 1$ ms

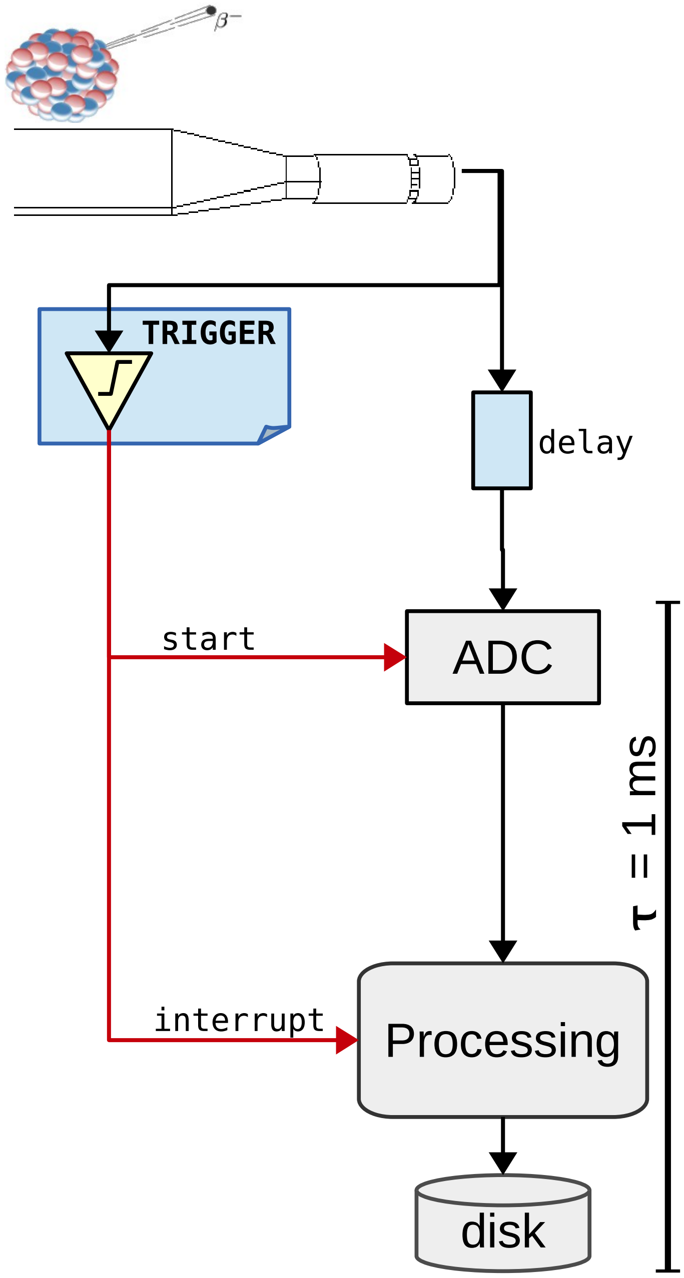




Basic DAQ: "real" trigger

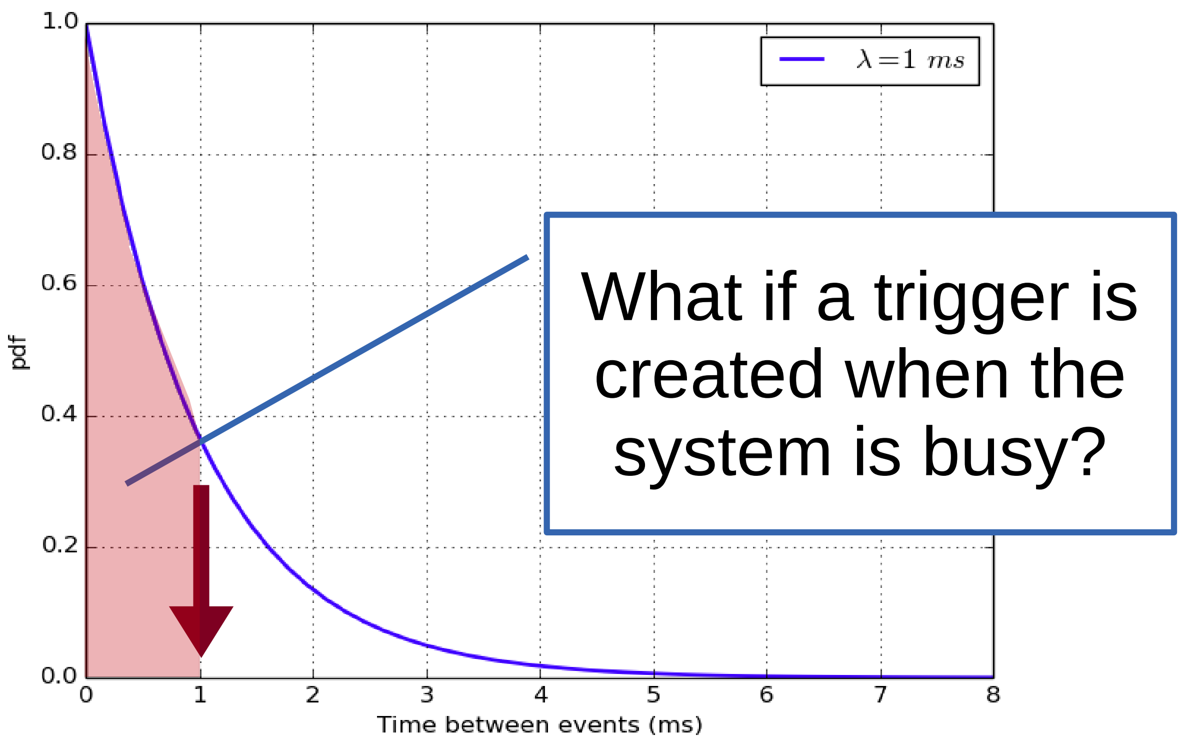
- Stochastic process
  - Fluctuations in time between events
- Let's assume for example
  - physics rate f = 1 kHz, i.e. λ = 1 ms
  - and, as before, τ = 1 ms

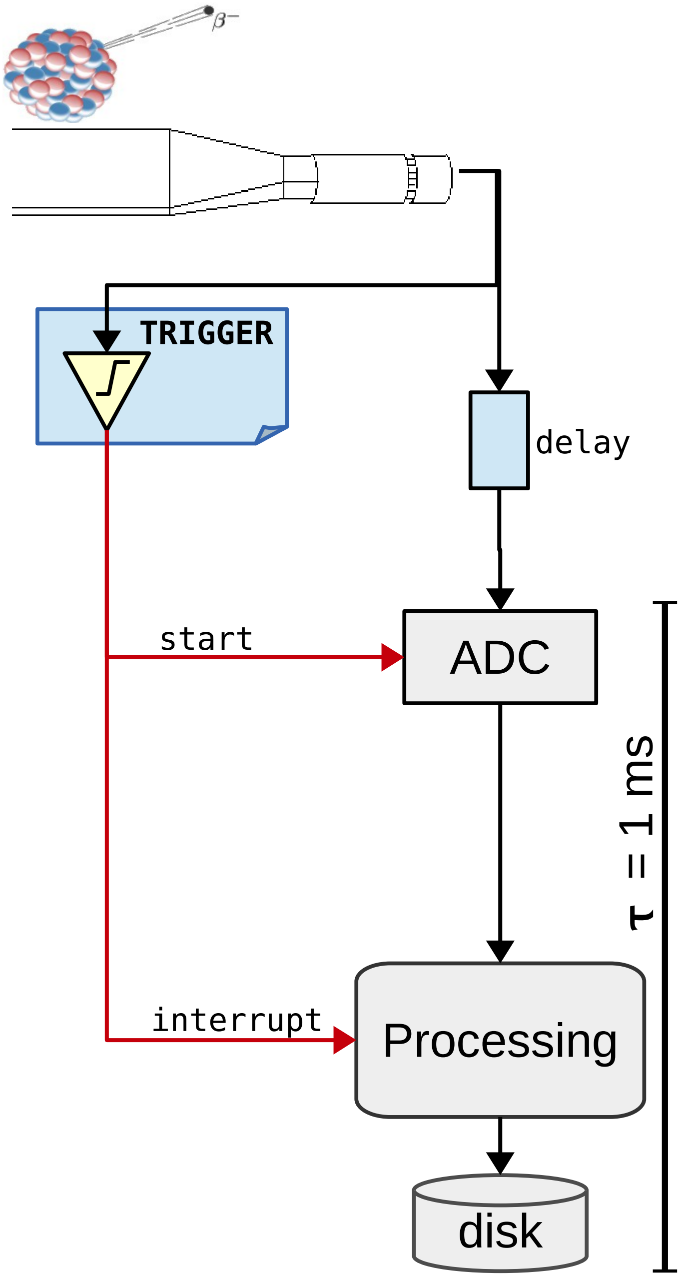




Basic DAQ: "real" trigger

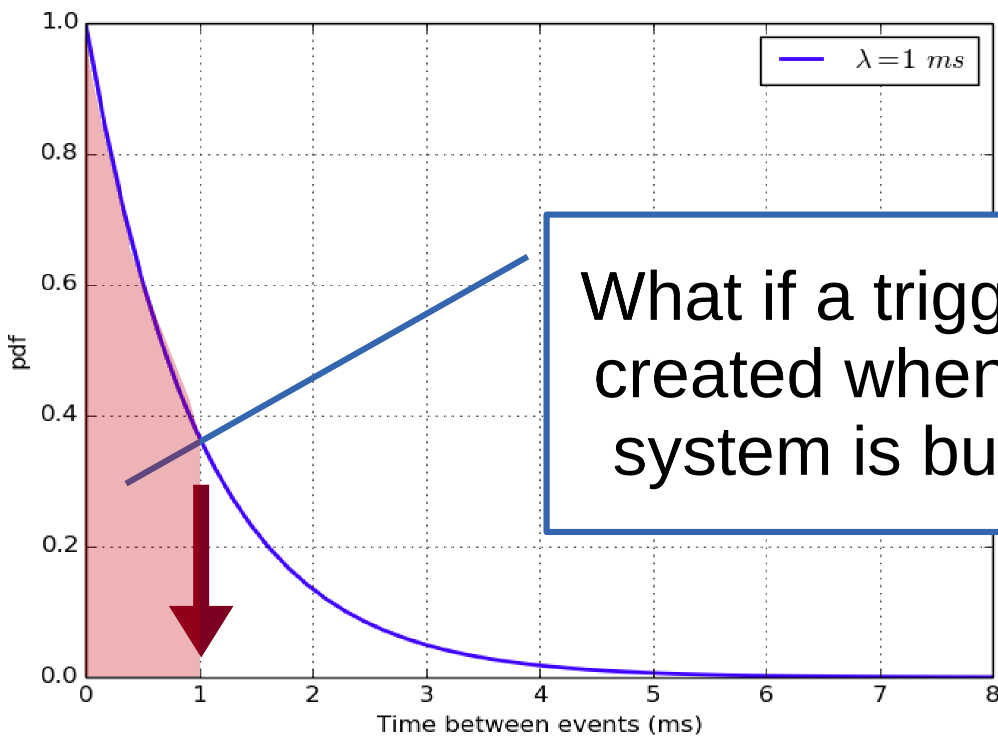
- Stochastic process
  - Fluctuations in time between events
- Let's assume for example
  - physics rate f = 1 kHz, i.e. $\lambda = 1$ ms
  - and, as before, $\tau = 1$ ms



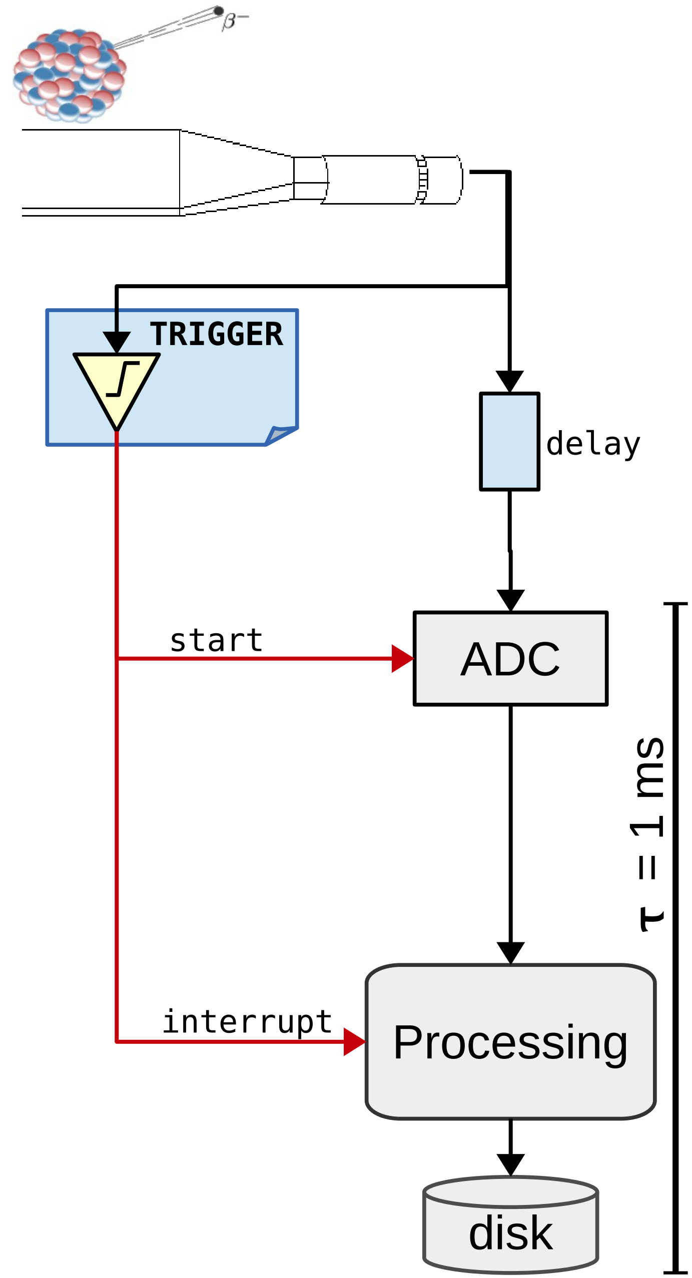


System still processing ...

- If a new trigger arrives when the system is still processing the previous event
  - The processing of the previous event could be screwed up



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

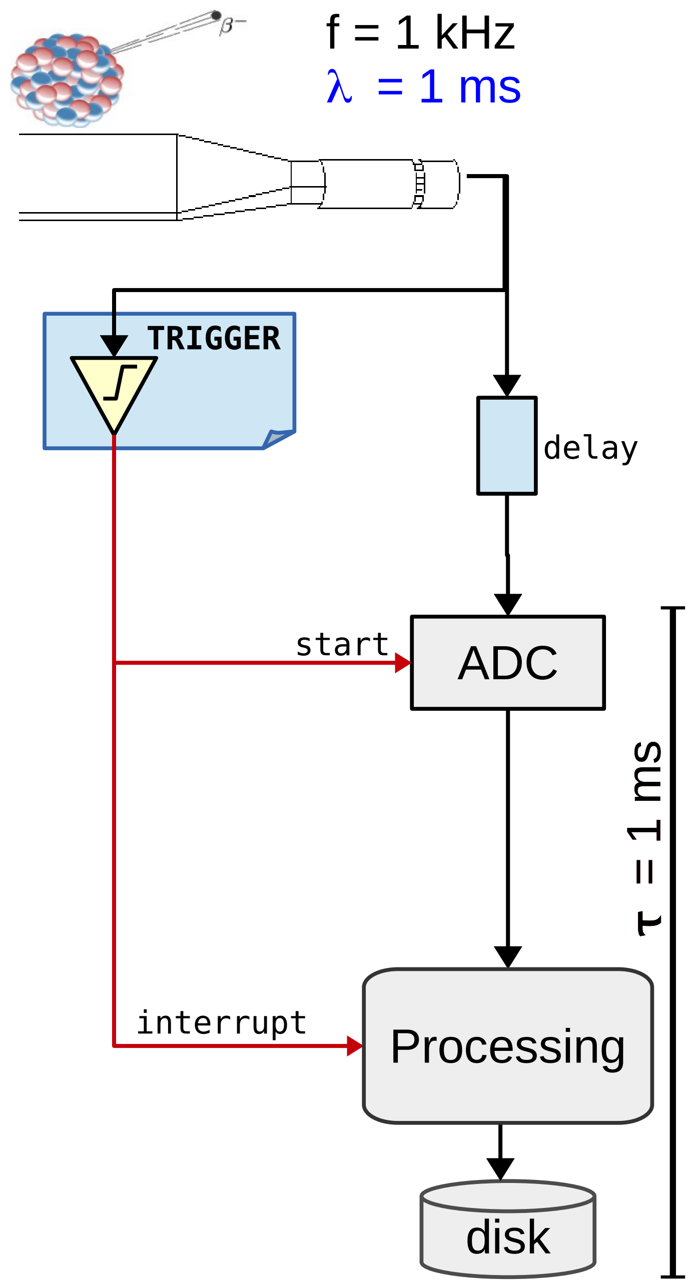


Pause to regroup

- For stochastic processes, our trigger and daq system needs to be able to:
  - determine if there is an "event" (**trigger**)
  - process and store the data from the event (**daq**)
  - w/o getting paralysed by new events
- Need a feedback mechanism, to know if the data processing pipeline is free to process a new event:
  - **busy logic**

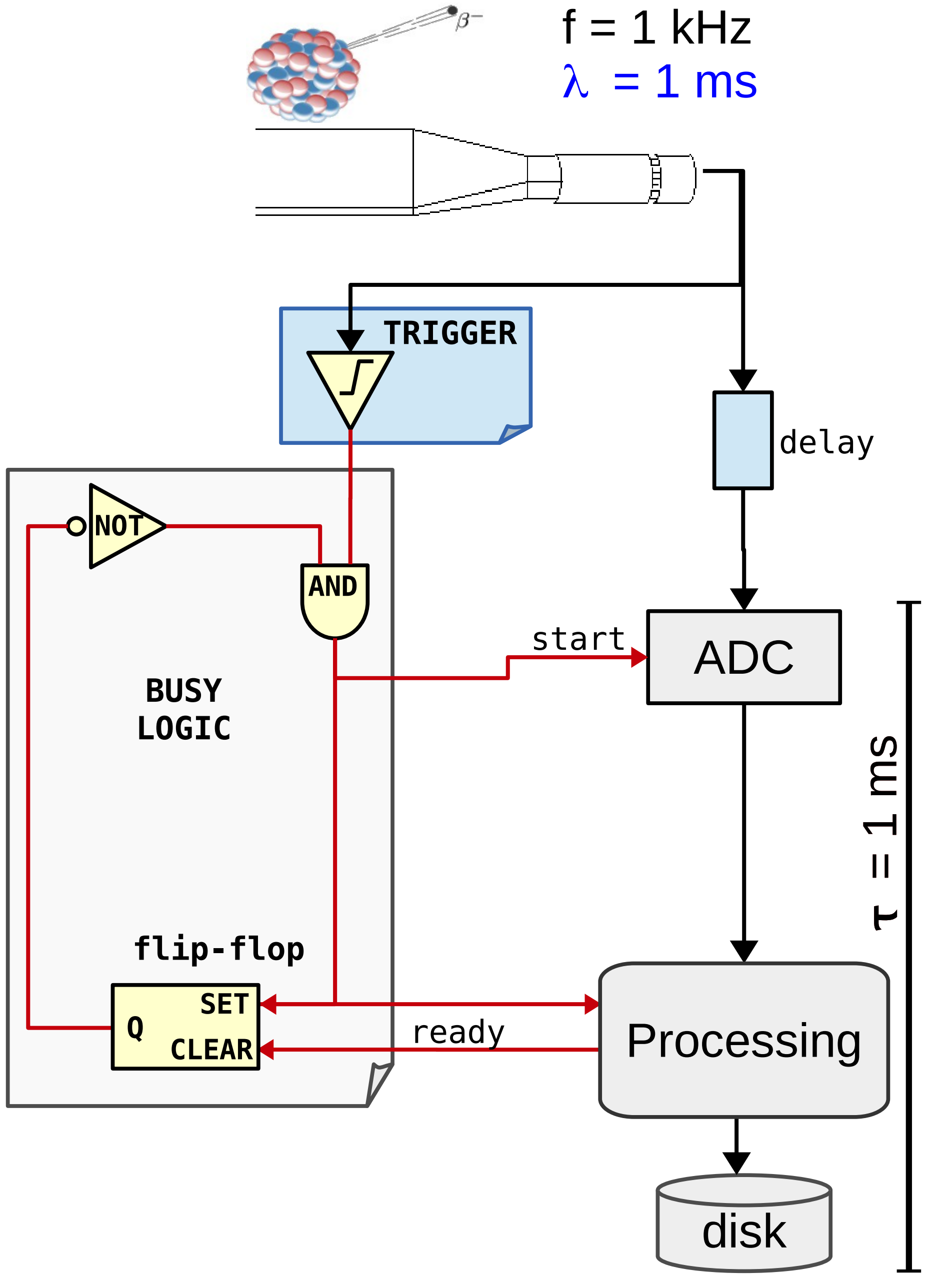
Busy logic

- The **busy logic** avoids triggers while the system is busy in processing
- A minimal **busy logic** can be implemented with
  - an **AND** gate
  - a **NOT** gate
  - a flip-flop (**flip-flop**)
- More in lab 2



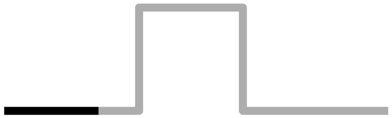
Busy logic

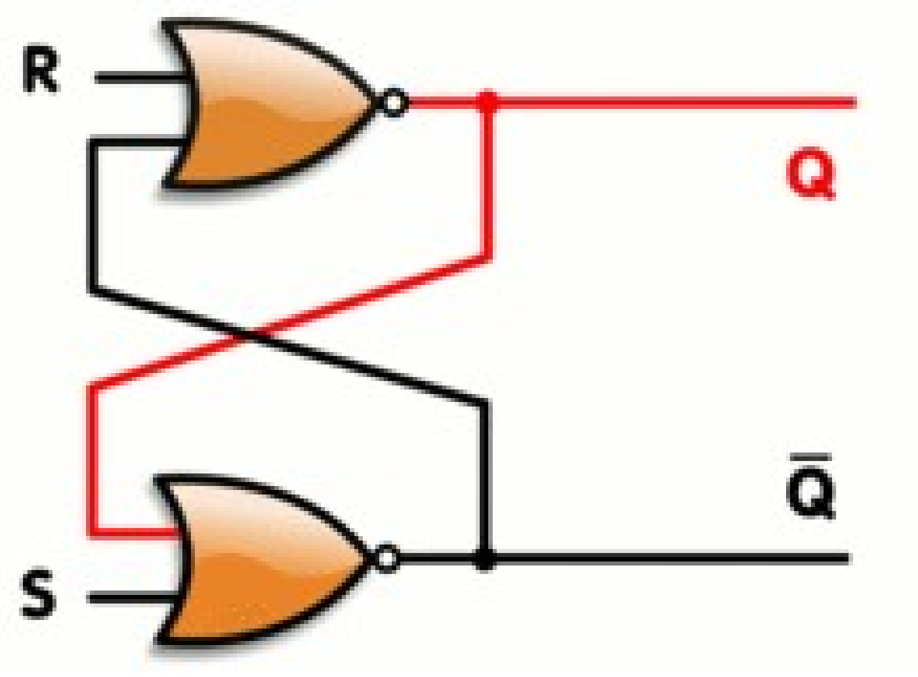
- The **busy logic** avoids triggers while the system is busy in processing
- A minimal **busy logic** can be implemented with
  - an **AND** gate
  - a **NOT** gate
  - a flip-flop (**flip-flop**)
- More in lab 2

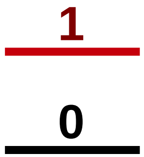


Flip Flop 1/5

- Flip-flop
  - a **bistable** circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)
- Before: stable state, Q up and $\overline{Q}$ down

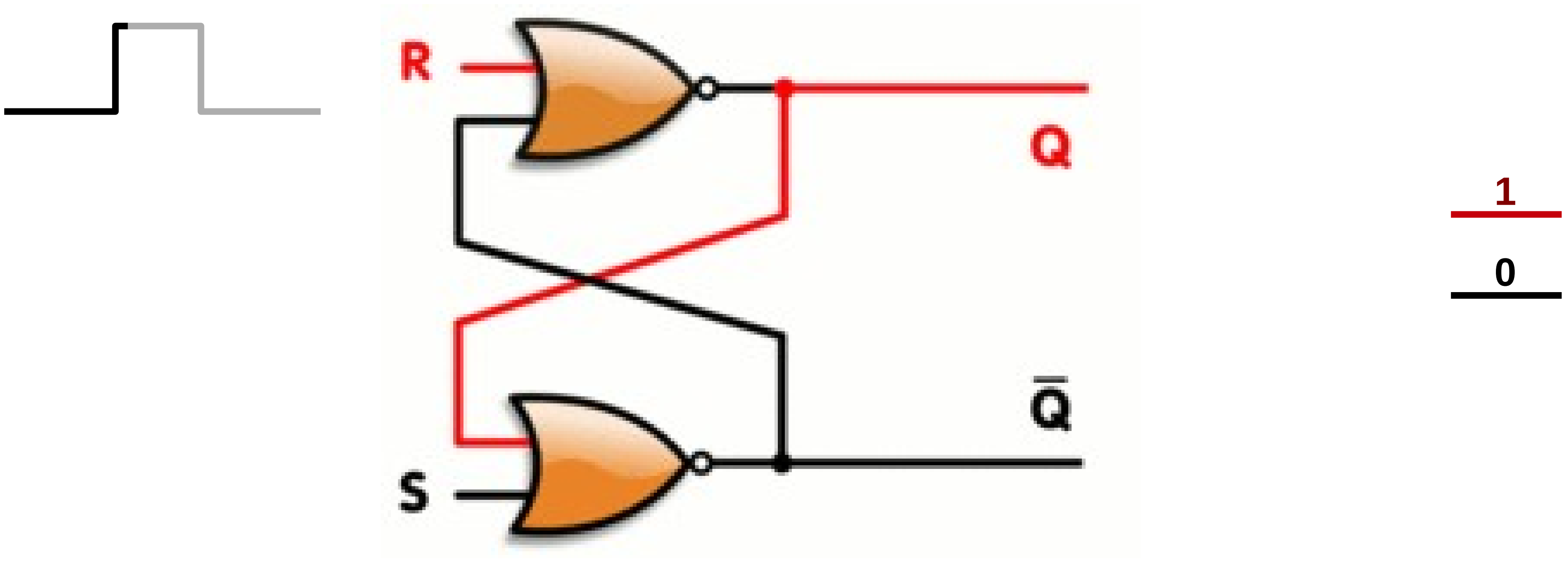






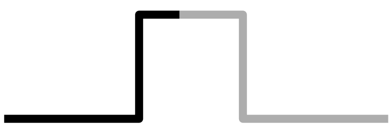
Flip Flop 2/5

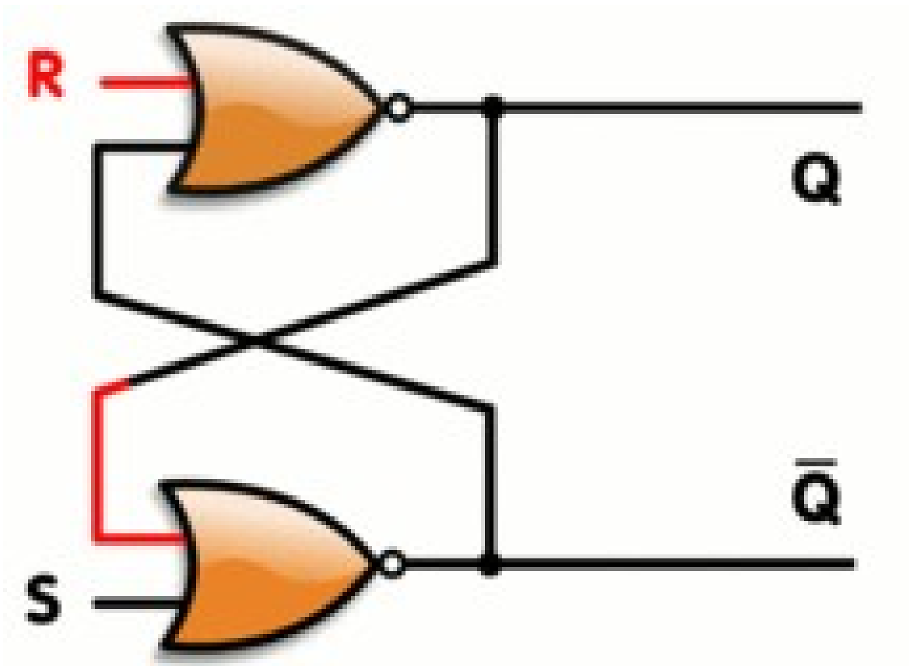
- Flip-flop
  - a **bistable** circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)
- At some point, signal injected in R

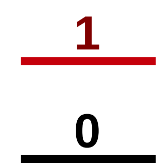


Flip Flop 3/5

- Flip-flop
  - a **bistable** circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)
- At some point, signal injected in R
  - Q switched down and the feedback travels to S

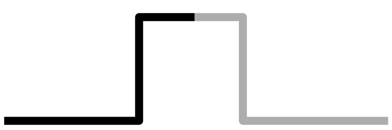


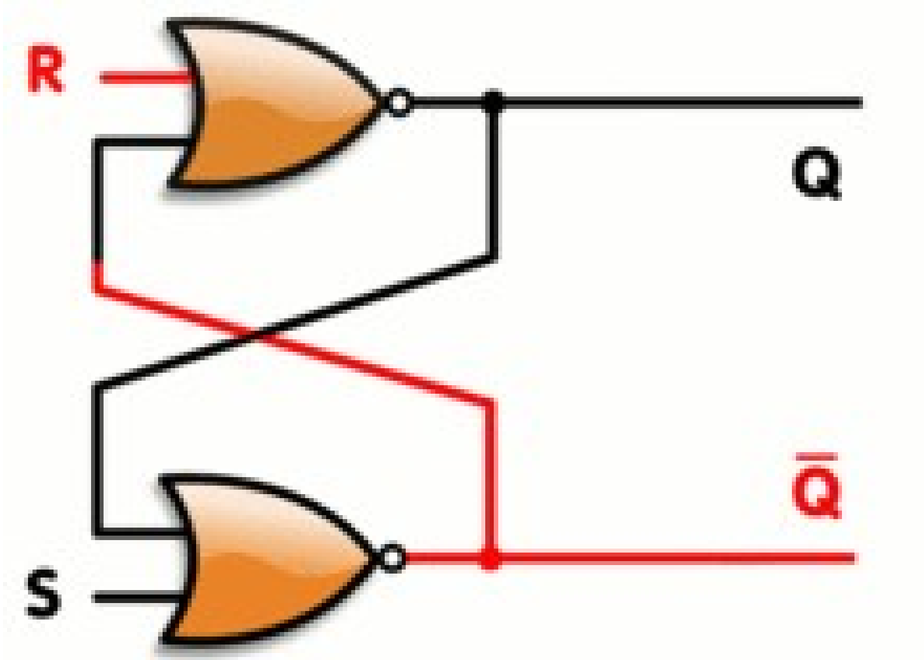


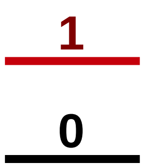


Flip Flop 4/5

- Flip-flop
  - a **bistable** circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)
- At some point, signal injected in R
  - $\overline{Q}$ becomes up and the feedback travels to R

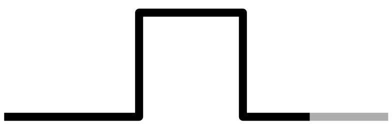


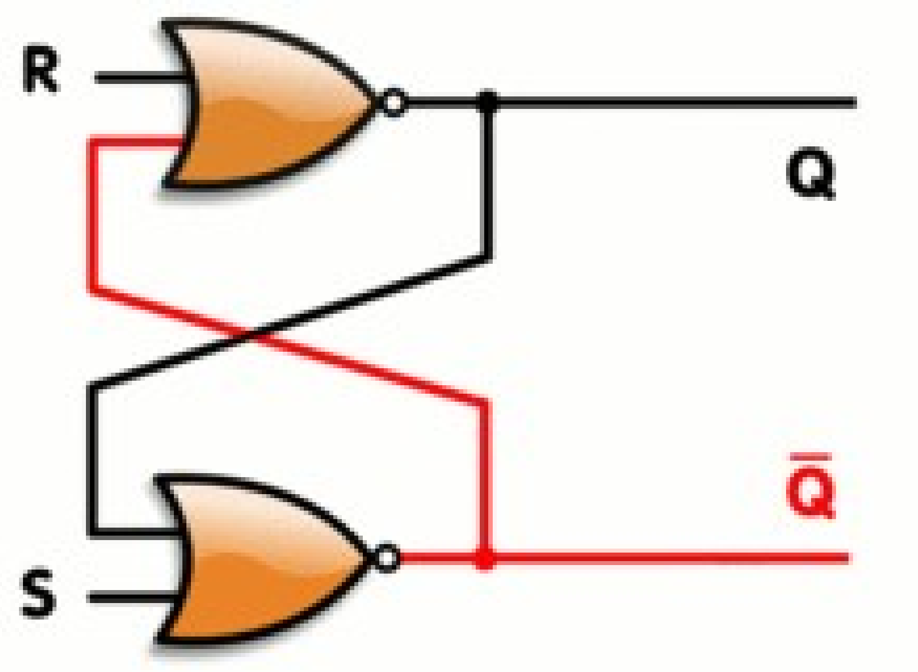


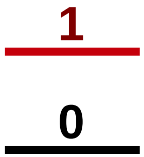


Flip Flop 5/5

- Flip-flop
  - a **bistable** circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)
- After: stable state, Q down and $\overline{Q}$ up:
  - End of pulse

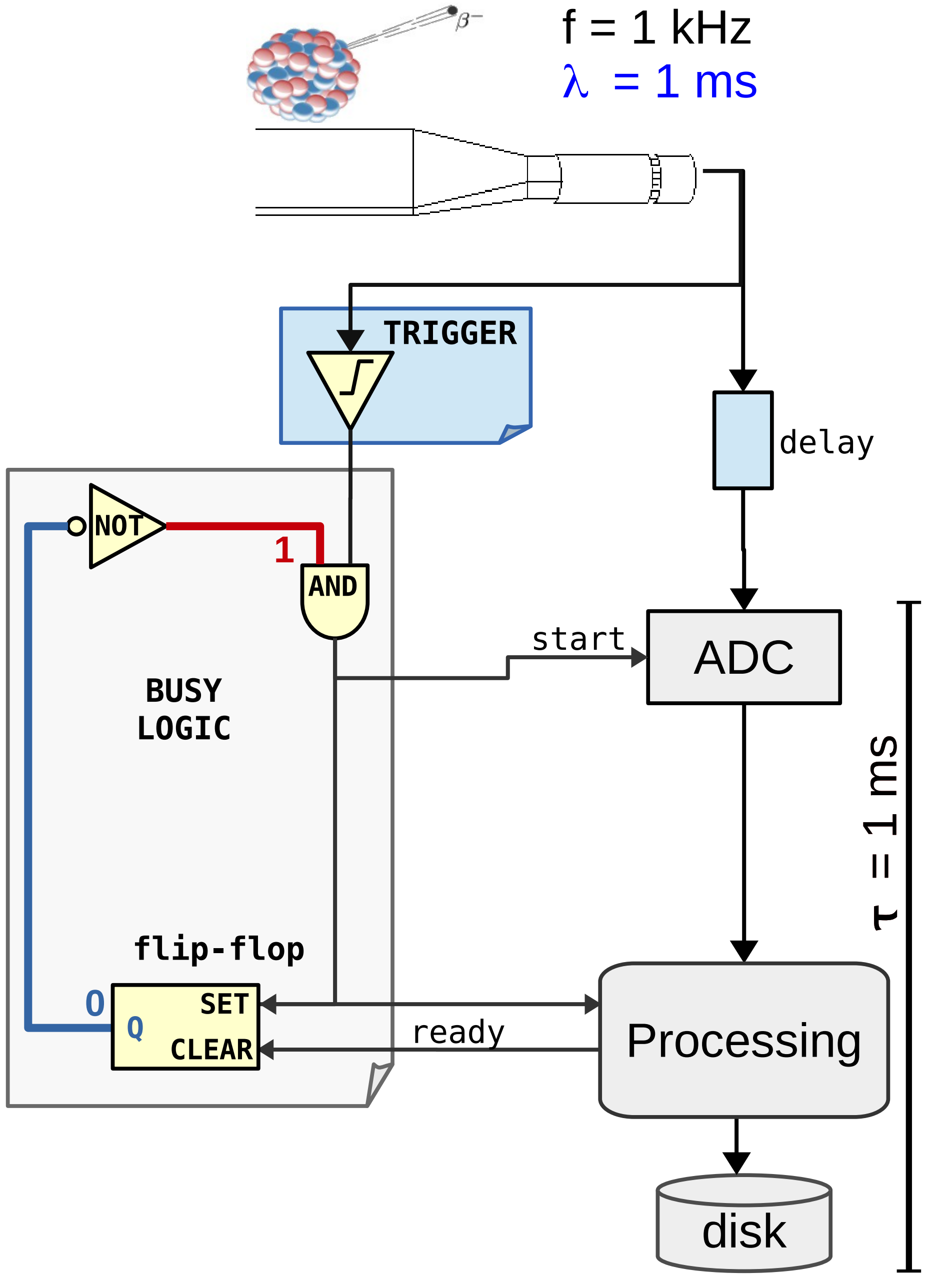






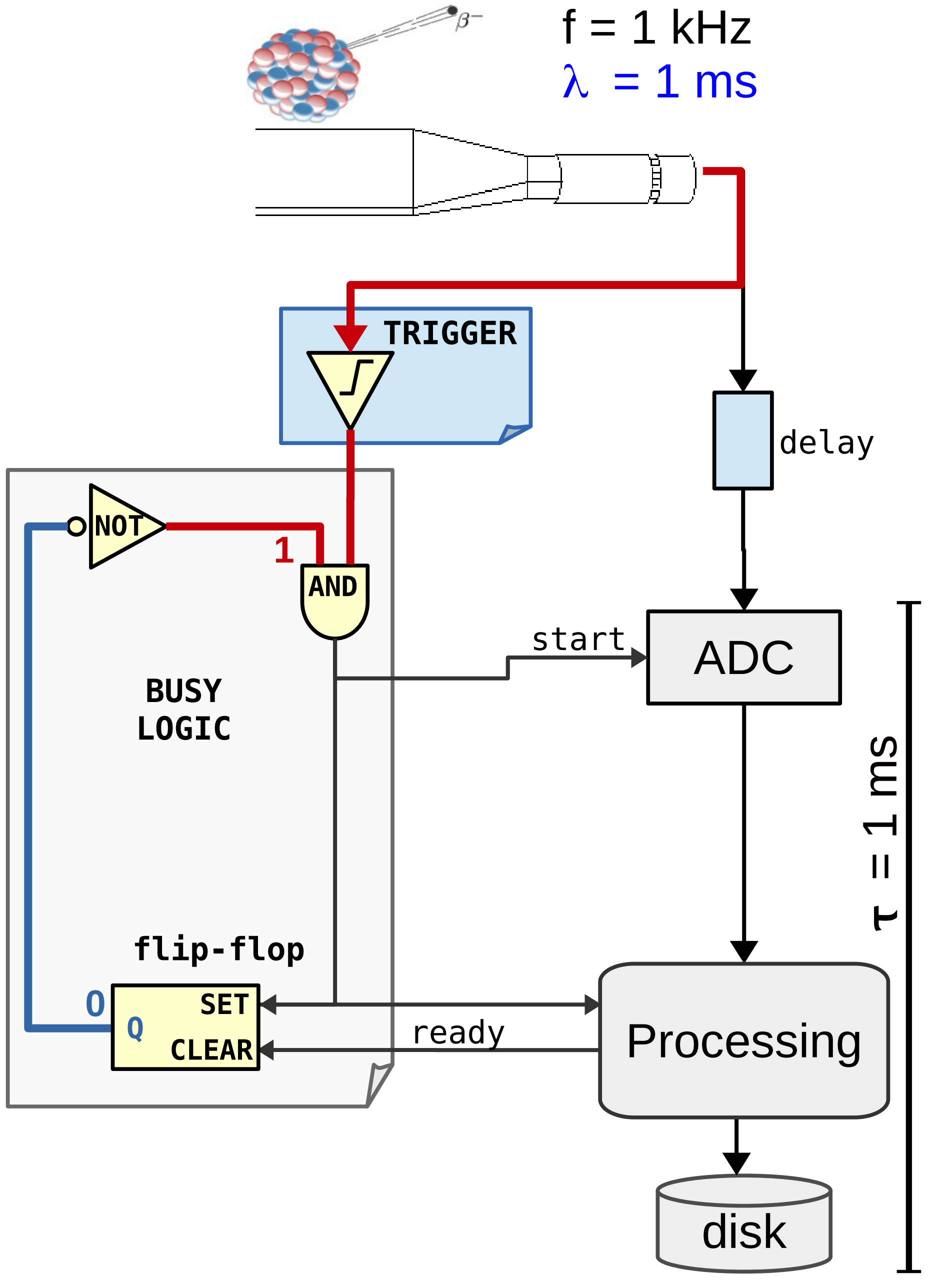
Busy logic

- Start of run
  - the flip-flop output is down (ground state)
  - via the NOT, one of the port of the AND gate is set to up (opened)
- i.e. system ready for new triggers



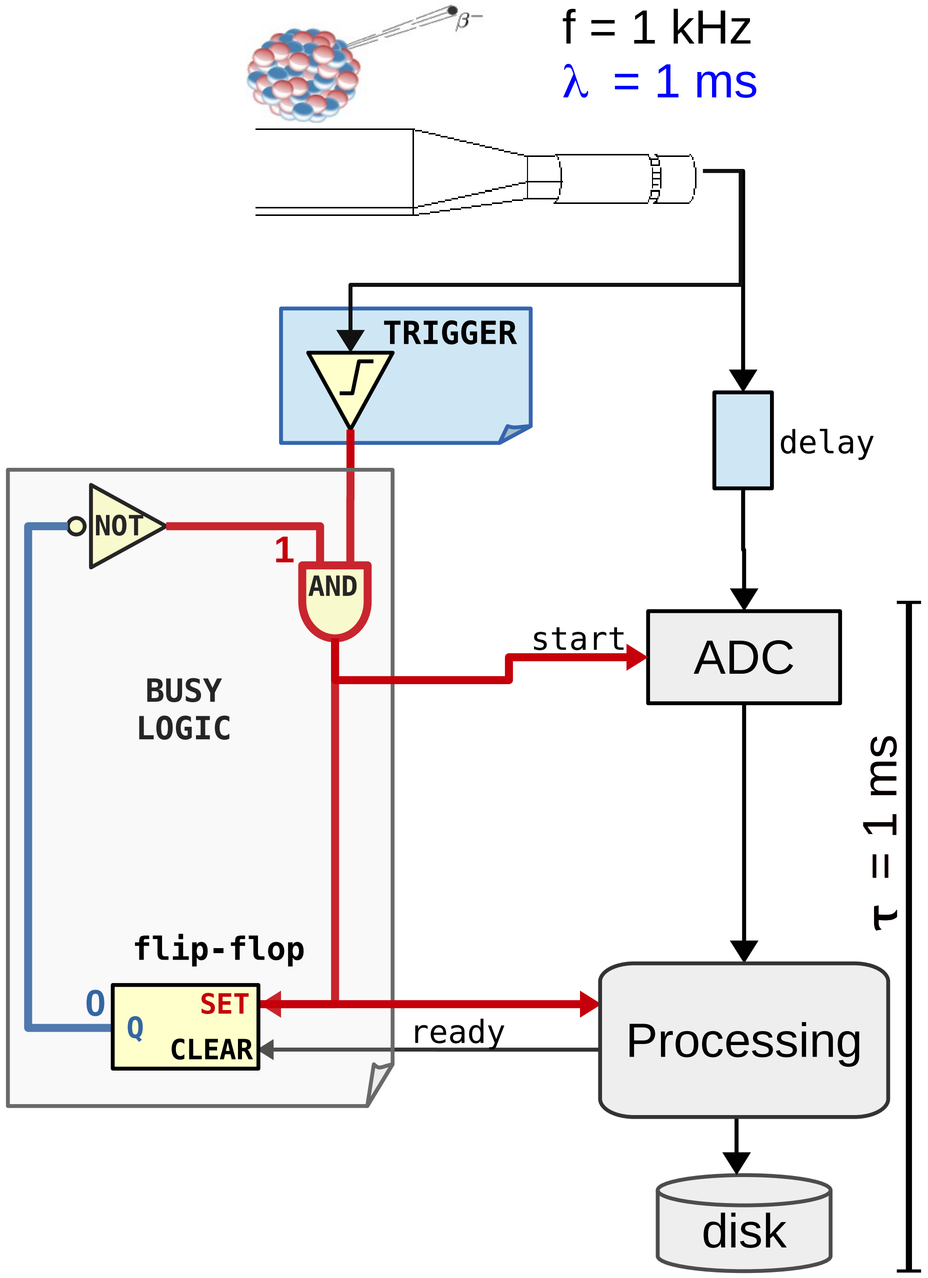
Busy logic

- If a trigger arrives, the signal finds the AND gate open, so:
  - The ADC is started
  - The processing is started
  - The flip-flop is flipped
  - One of the AND inputs is now steadily down (closed)
- Any new trigger is inhibited by the AND gate (busy)



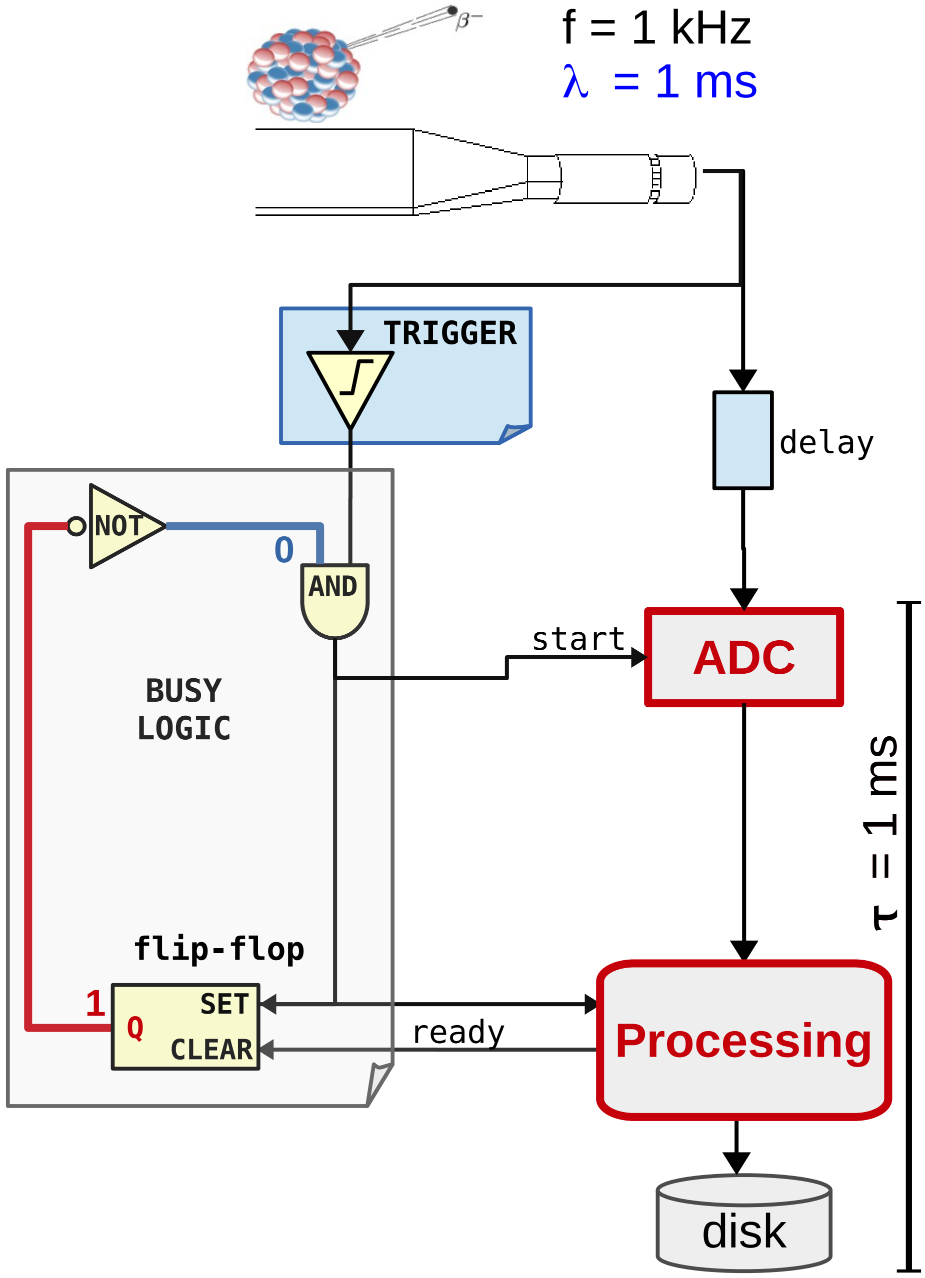
Busy logic

- If a trigger arrives, the signal finds the AND gate open, so:
  - The ADC is started
  - The processing is started
  - The flip-flop is flipped
  - One of the AND inputs is now steadily down (closed)
- Any new trigger is inhibited by the AND gate (busy)



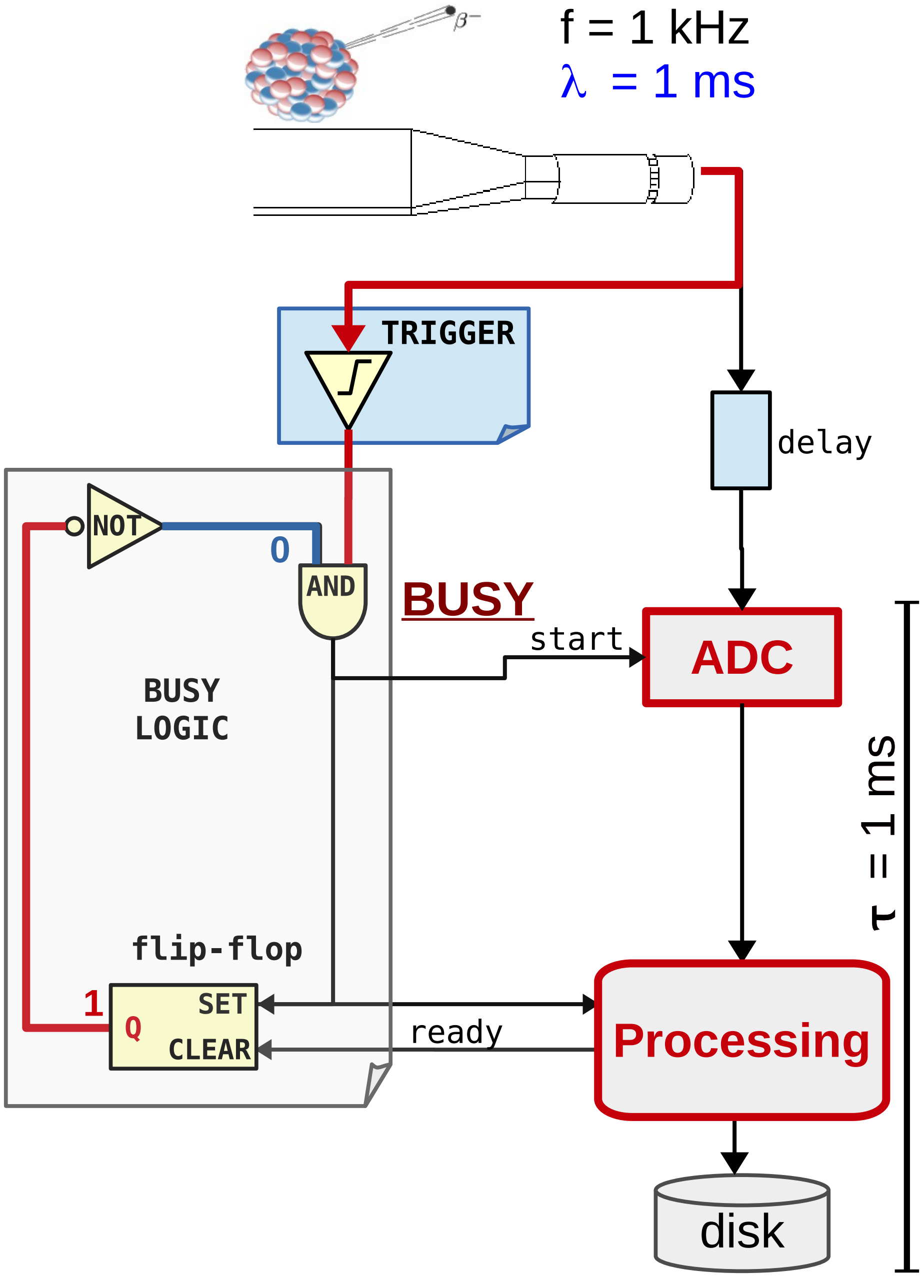
Busy logic

- If a trigger arrives, the signal finds the AND gate open, so:
  - The ADC is started
  - The processing is started
  - The flip-flop is flipped
  - One of the AND inputs is now steadily down (closed)
- Any new trigger is inhibited by the AND gate (busy)



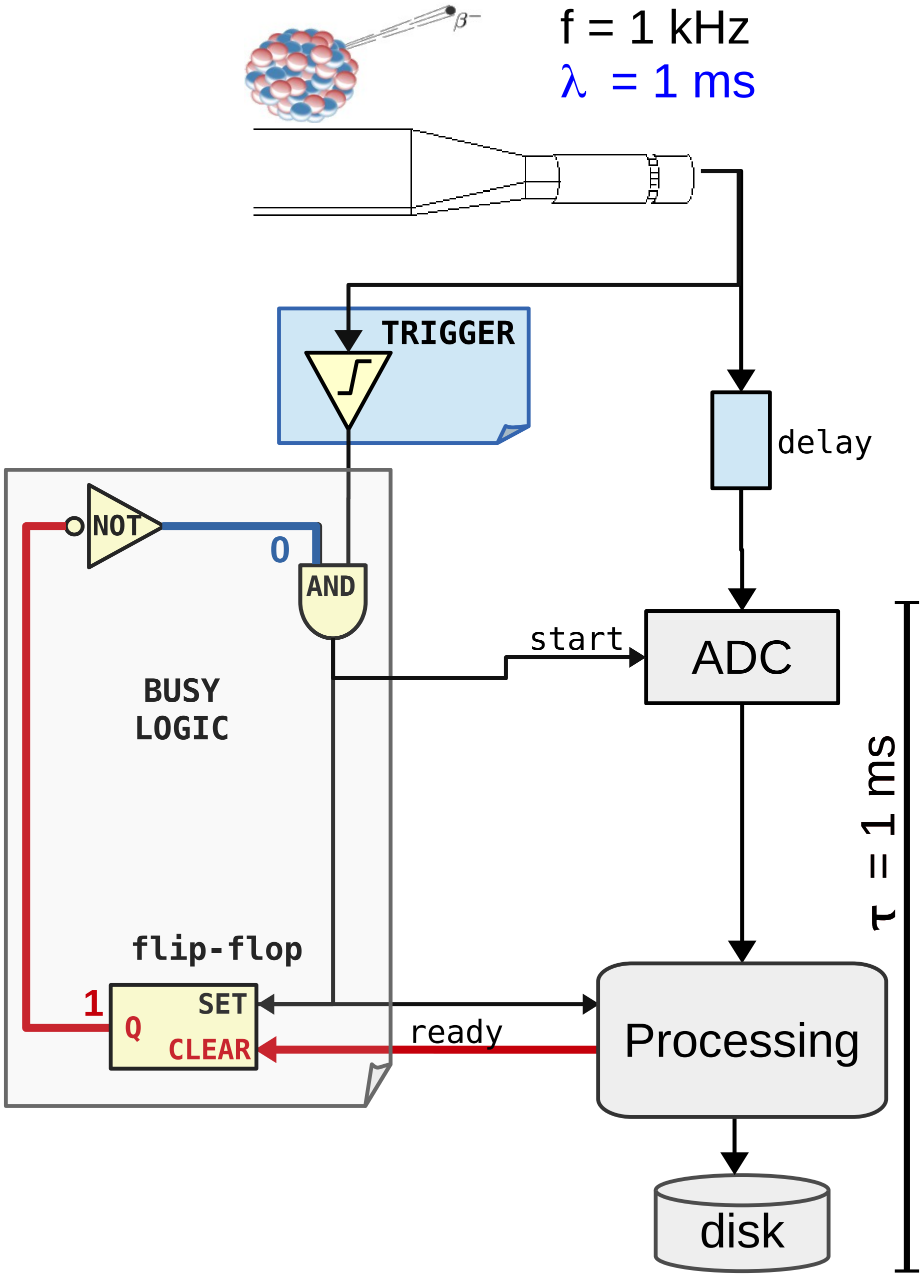
Busy logic

- If a trigger arrives, the signal finds the AND gate open, so:
  - The ADC is started
  - The processing is started
  - The flip-flop is flipped
  - One of the AND inputs is now steadily down (closed)
- Any new trigger is inhibited by the AND gate (busy)



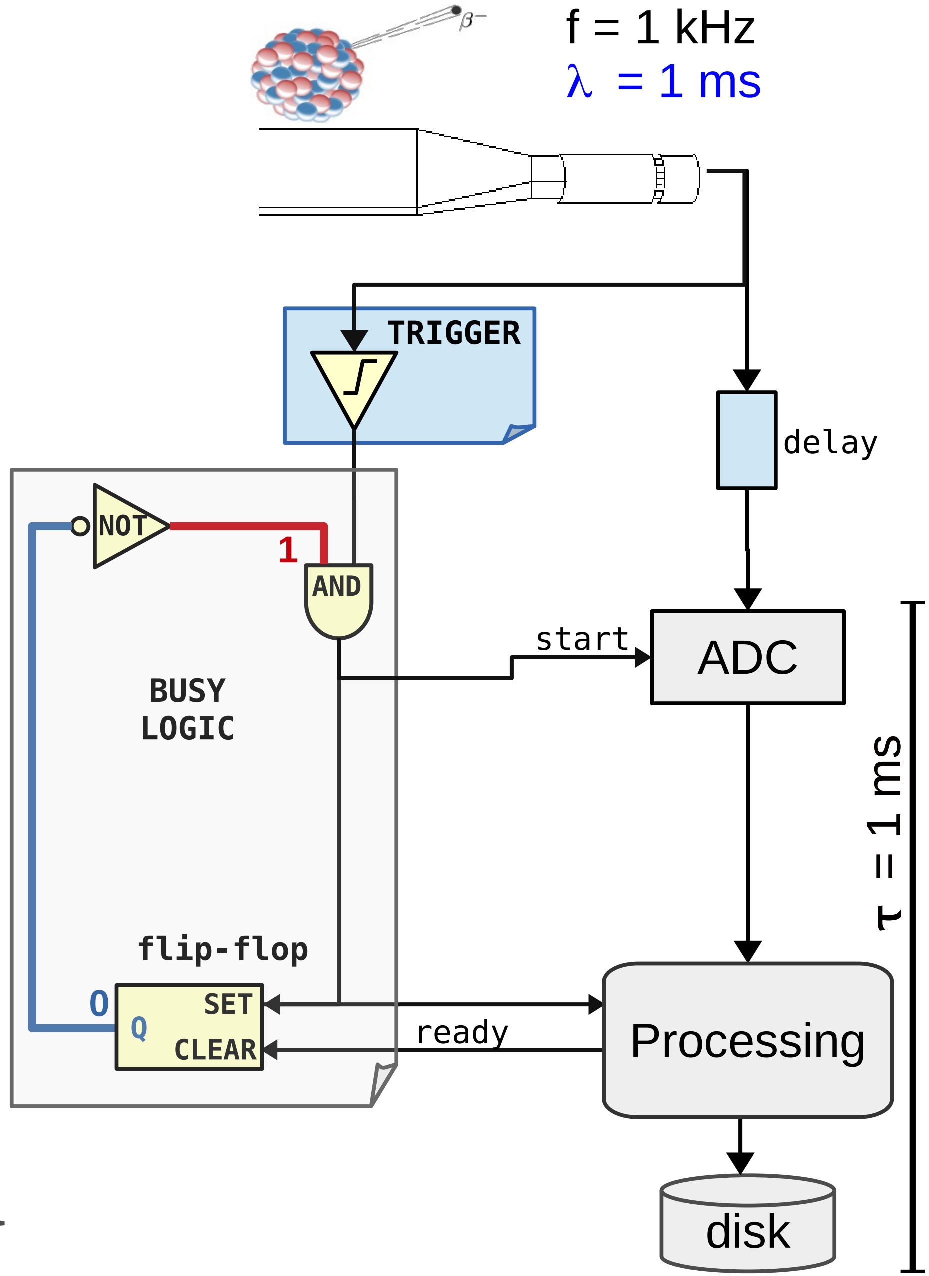
Busy logic

- At the end of processing a ready signal is sent to the flip-flop
  - The flip-flop flips again
  - The gate is now opened
  - The system is ready to accept a new trigger



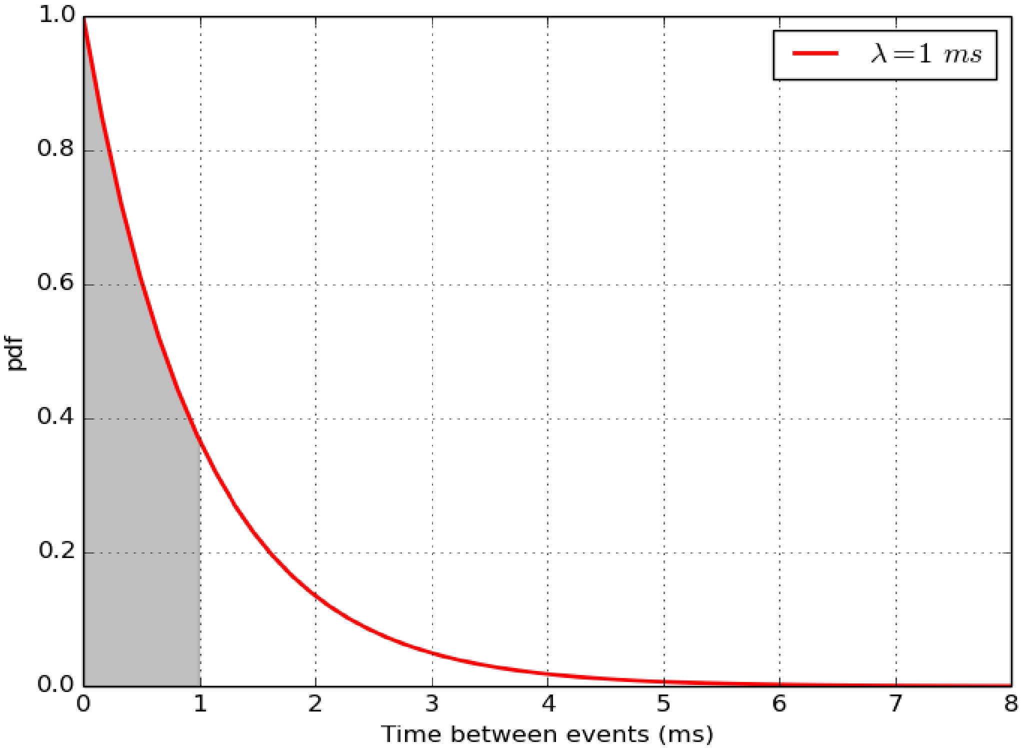
Busy logic

- At the end of processing a ready signal is sent to the flip-flop
  - The flip-flop flips again
  - The gate is now opened
  - The system is ready to accept a new trigger
- i.e. busy logic avoids triggers while daq is busy in processing
  - New triggers do not interfere w/ previous data



Deadtime and efficiency

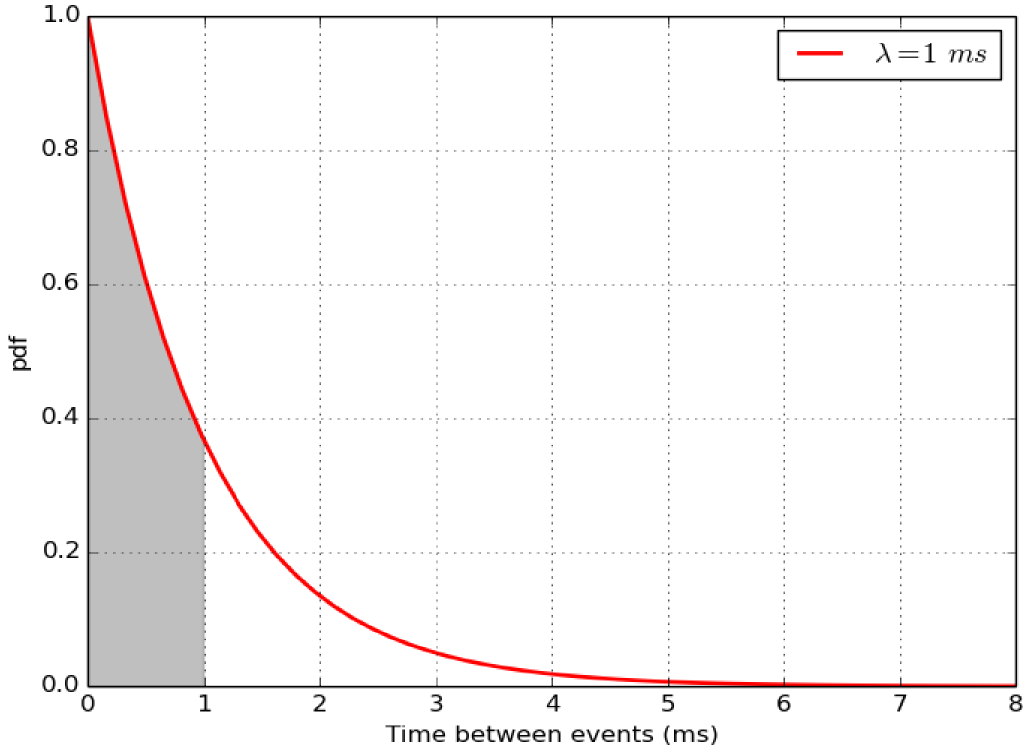
- So the **busy** mechanism protects our electronics from unwanted triggers
  - New signals are accepted only when the system in ready to process them



- Which (average) DAQ rate can we achieve now?
  - How much we lose with the busy logic?
  - Reminder: with a clock trigger and τ = 1 ms the limit was 1 kHz

Deadtime and efficiency

- Definitions
  - **f**: average rate of physics (input)
  - **ν**: average rate of DAQ (output)
  - **τ: deadtime**, needed to process an event, without being able to handle other triggers

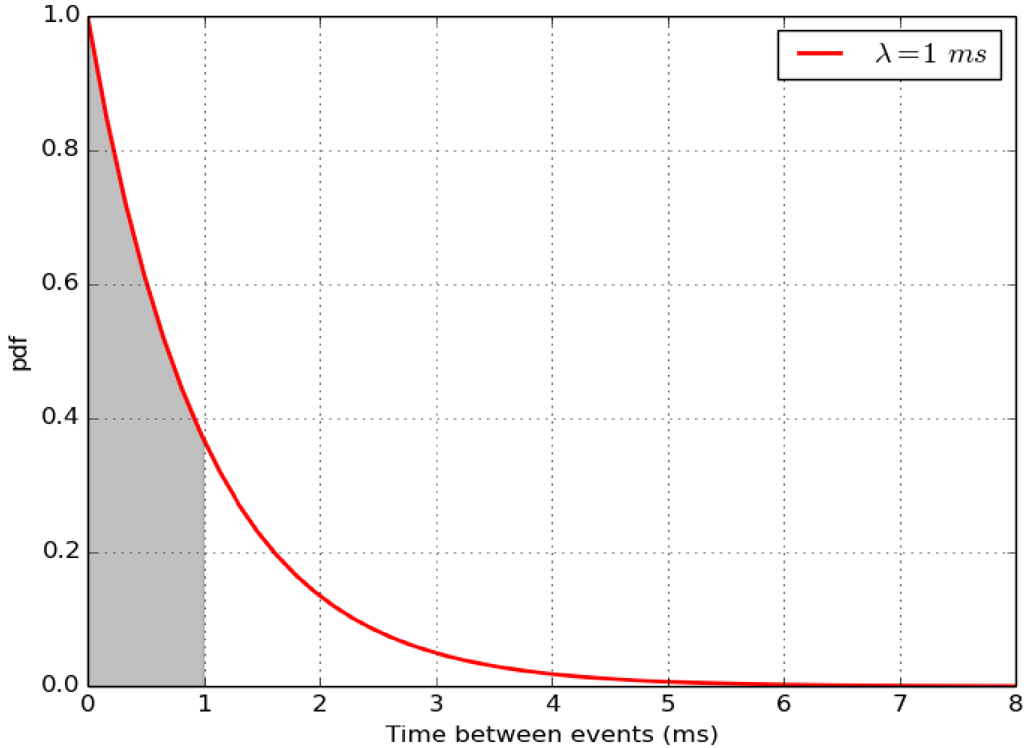


  - probabilities: P[busy] = ν τ; P[free] = 1 - ν τ

- Therefore:

$$\nu = f\, P[free] \Rightarrow \nu = f(1 - \nu\tau) \Rightarrow \nu = \frac{f}{1 + f\tau}$$

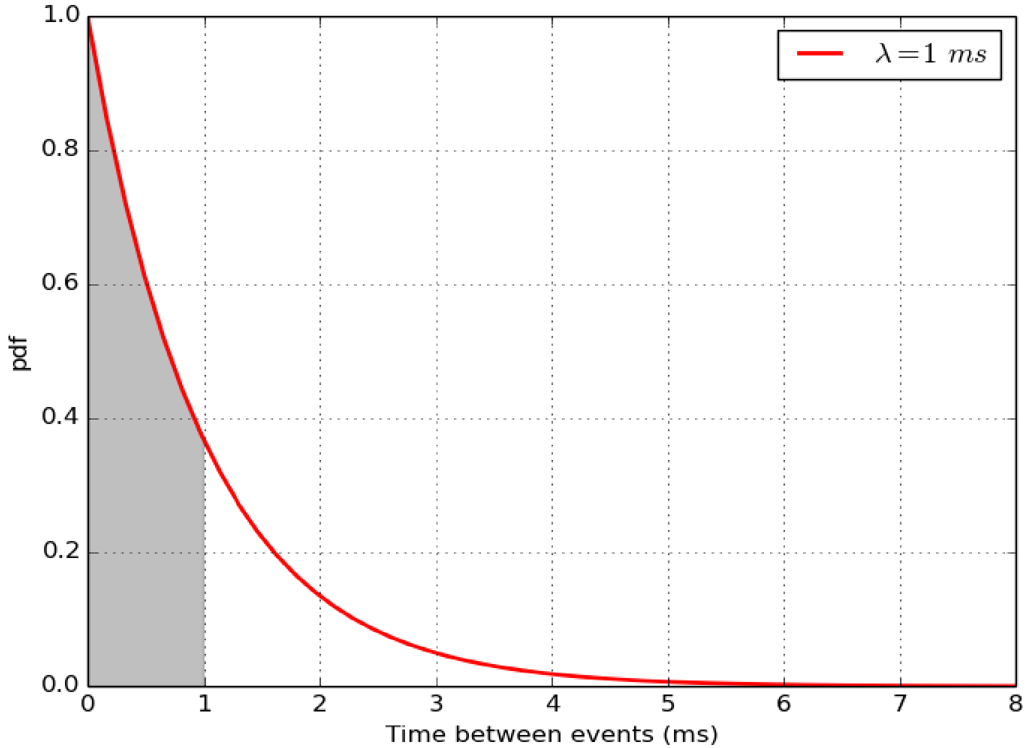
Deadtime and efficiency



- Definitions
  - **f**: average rate of physics (input)
  - **ν**: average rate of DAQ (output)
  - **τ: deadtime**, needed to process an event, without being able to handle other triggers
  - probabilities: P[busy] = ν τ; P[free] = 1 - ν τ
- Therefore:

$$\nu = f\,P[free] \Rightarrow \nu = f(1-\nu\tau) \Rightarrow \nu = \frac{f}{1+f\tau}$$

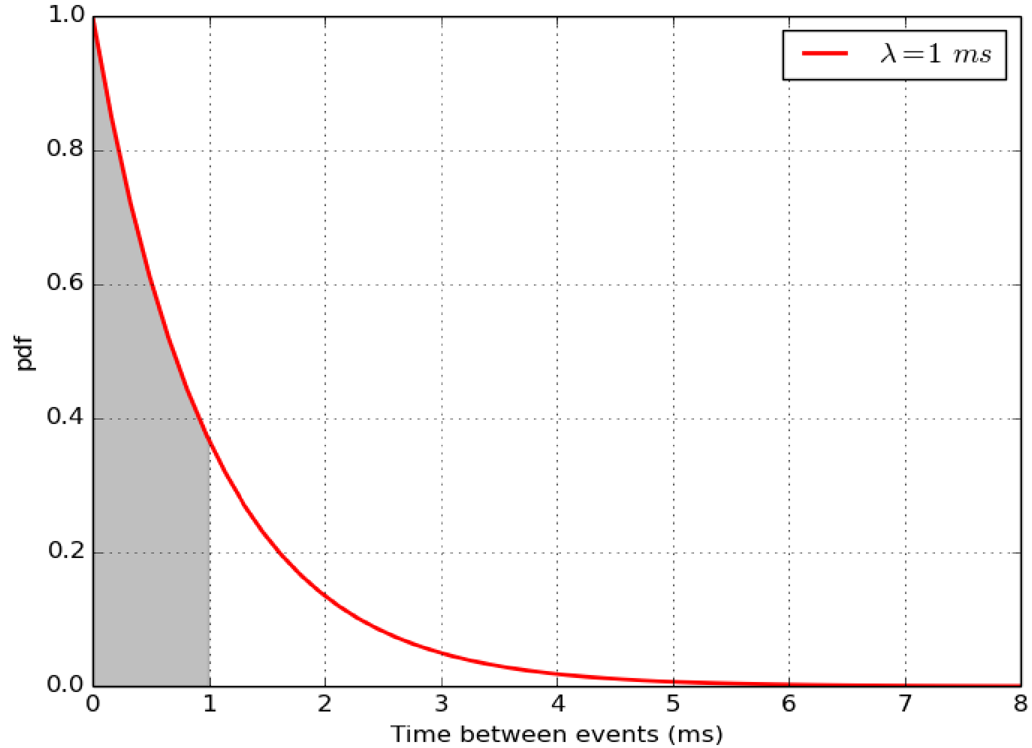
Deadtime and efficiency



- Definitions
  - **f**: average rate of physics (input)
  - **ν**: average rate of DAQ (output)
  - **τ: deadtime**, needed to process an event, without being able to handle other triggers
  - probabilities: P[busy] = ν τ; P[free] = 1 - ν τ
- Therefore:

$$\nu = f\,P[free] \Rightarrow \nu = f(1-\nu\tau) \Rightarrow \nu = \frac{f}{1+f\tau}$$

Deadtime and efficiency



- Definitions
  - **f**: average rate of physics (input)
  - **ν**: average rate of DAQ (output)
  - **τ: deadtime**, needed to process an event, without being able to handle other triggers
  - probabilities: P[busy] = ν τ; P[free] = 1 - ν τ
- Therefore:

$$\nu = f\,P[free] \Rightarrow \nu = f(1 - \nu\tau) \Rightarrow \nu = \frac{f}{1 + f\tau}$$

Deadtime and efficiency

- Due to stochastic fluctuations
  - DAQ rate always < physics rate $\nu = \frac{f}{1+f\tau} < f$
  - Efficiency always < 100% $\epsilon = \frac{N_{saved}}{N_{tot}} = \frac{1}{1+f\tau} < 100\%$

- So, in our specific example
  - Physics rate 1 kHz
  - Deadtime 1 ms

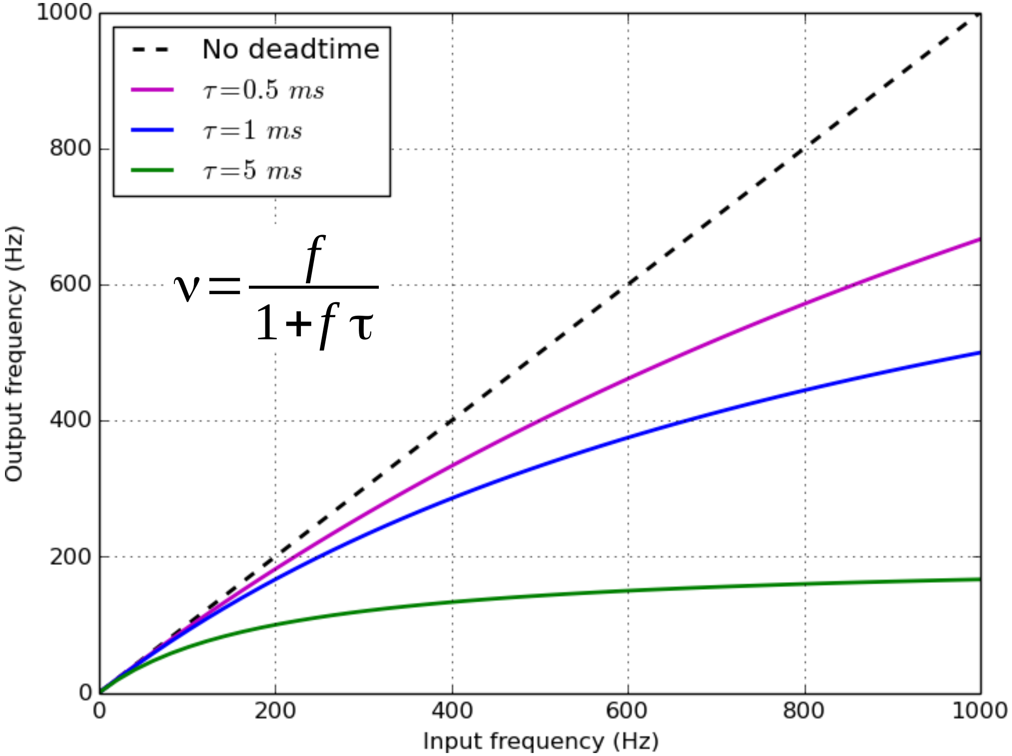
$$\left| \begin{array}{l} f = 1\,kHz \\ \tau = 1\,ms \end{array} \right. \rightarrow \left| \begin{array}{l} \nu = 500\,Hz \\ \epsilon = 50\,\% \end{array} \right.$$

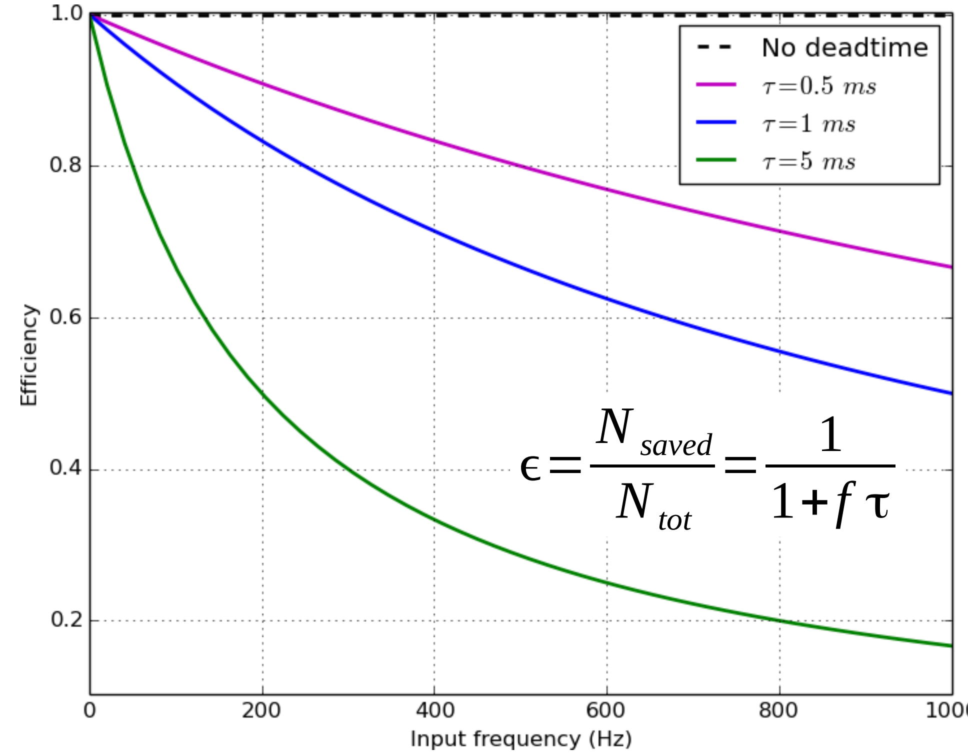
Deadtime and efficiency

- Due to stochastic fluctuations
  - DAQ rate always < physics rate $\nu = \frac{f}{1+f\tau} < f$
  - Efficiency always < 100% $\epsilon = \frac{N_{saved}}{N_{tot}} = \frac{1}{1+f\tau} < 100\%$
- So, in our specific example
  - Physics rate 1 kHz
  - Deadtime 1 ms

$$\left| \begin{array}{l} f = 1\,kHz \\ \tau = 1\,ms \end{array} \right. \rightarrow \left| \begin{array}{l} \nu = 500\,Hz \\ \epsilon = 50\% \end{array} \right.$$

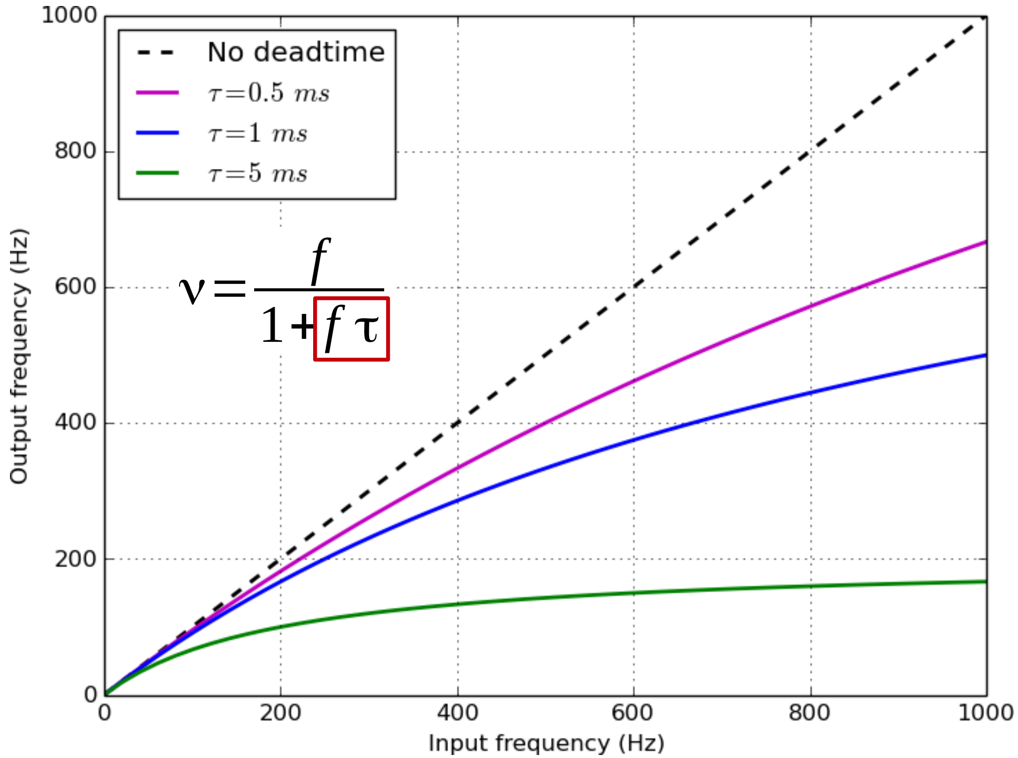
Deadtime and efficiency

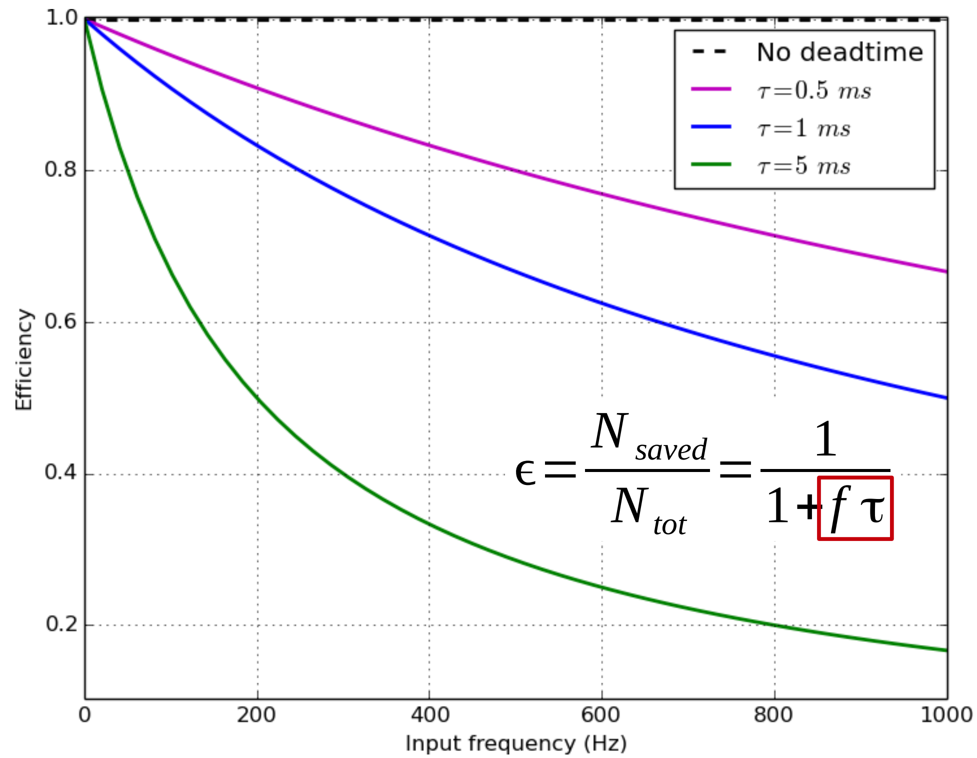




- In order to obtain ε~100% (i.e.: ν~f) → fτ << 1 → τ << λ
  - E.g.: ε~99% for f = 1 kHz → τ < 0.01 ms → 1/τ > 100 kHz
  - To cope with the input signal fluctuations, we have to **over-design** our DAQ system **by a factor 100!**
- How can we mitigate this effect?

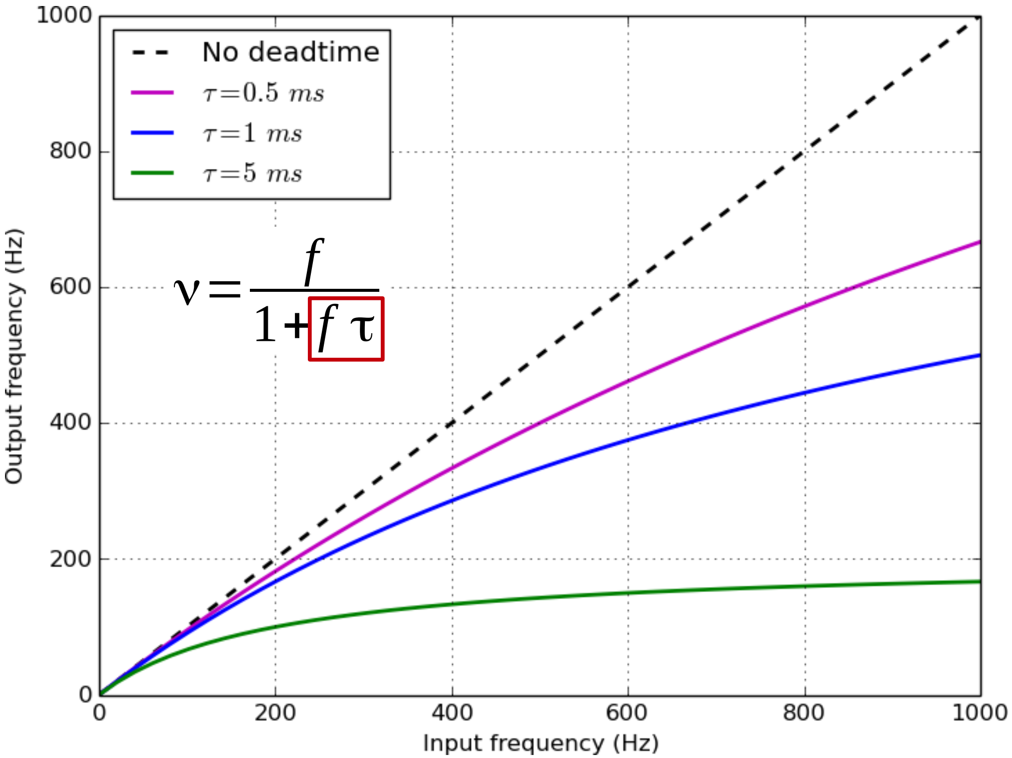
Deadtime and efficiency





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

Deadtime and efficiency

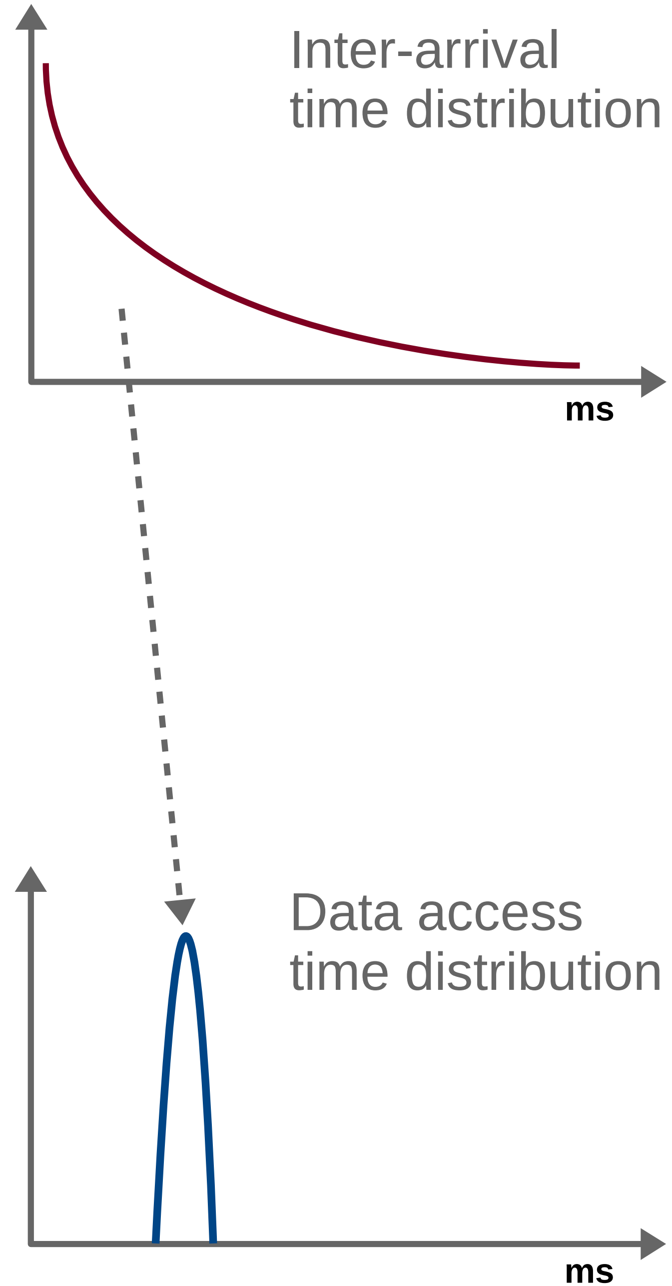


- In order to obtain ε~100% (i.e.: ν~f) → fτ << 1 → τ << λ
  - E.g.: ε~99% for f = 1 kHz → τ < 0.01 ms → 1/τ > 100 kHz
  - To cope with the input signal fluctuations, we have to **over-design** our DAQ system **by a factor 100!**

- How can we mitigate this effect?

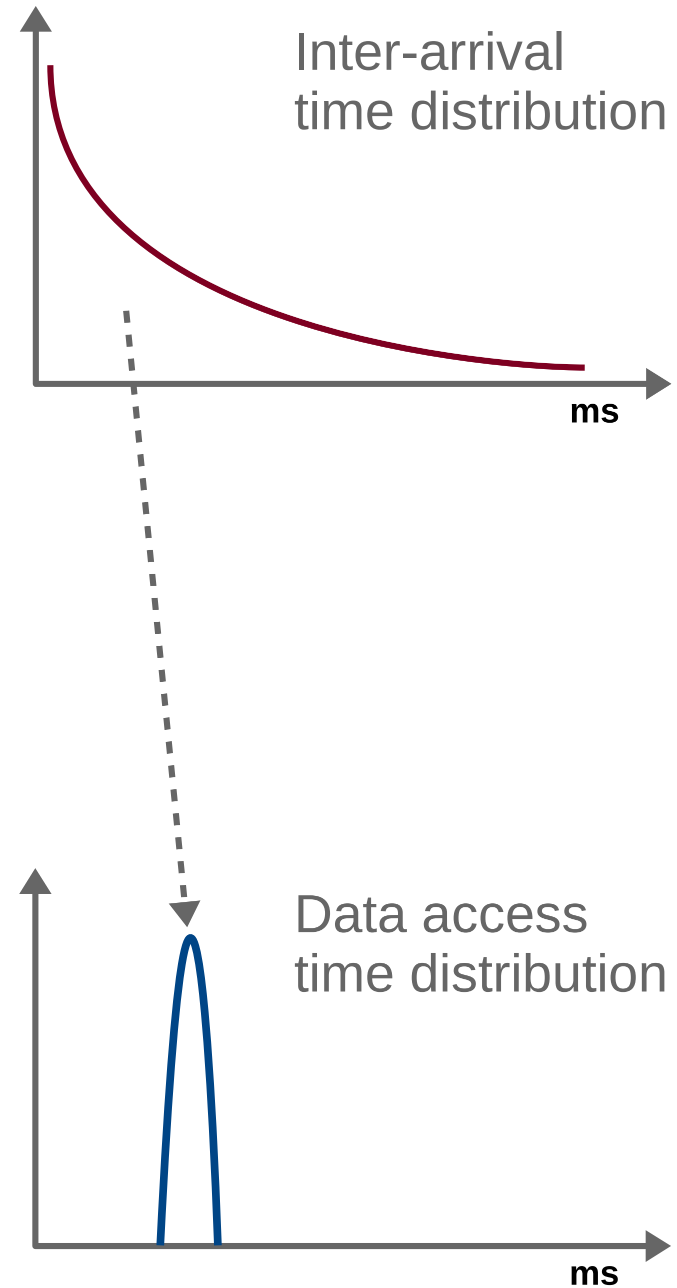
De-randomization

- What if we were able to make the system more **deterministic** and less dependent on the arrival time of our signals?
  - Then we could ensure that events don't arrive when the system is busy
  - This is called **de-randomization**
- How it can be achieved?



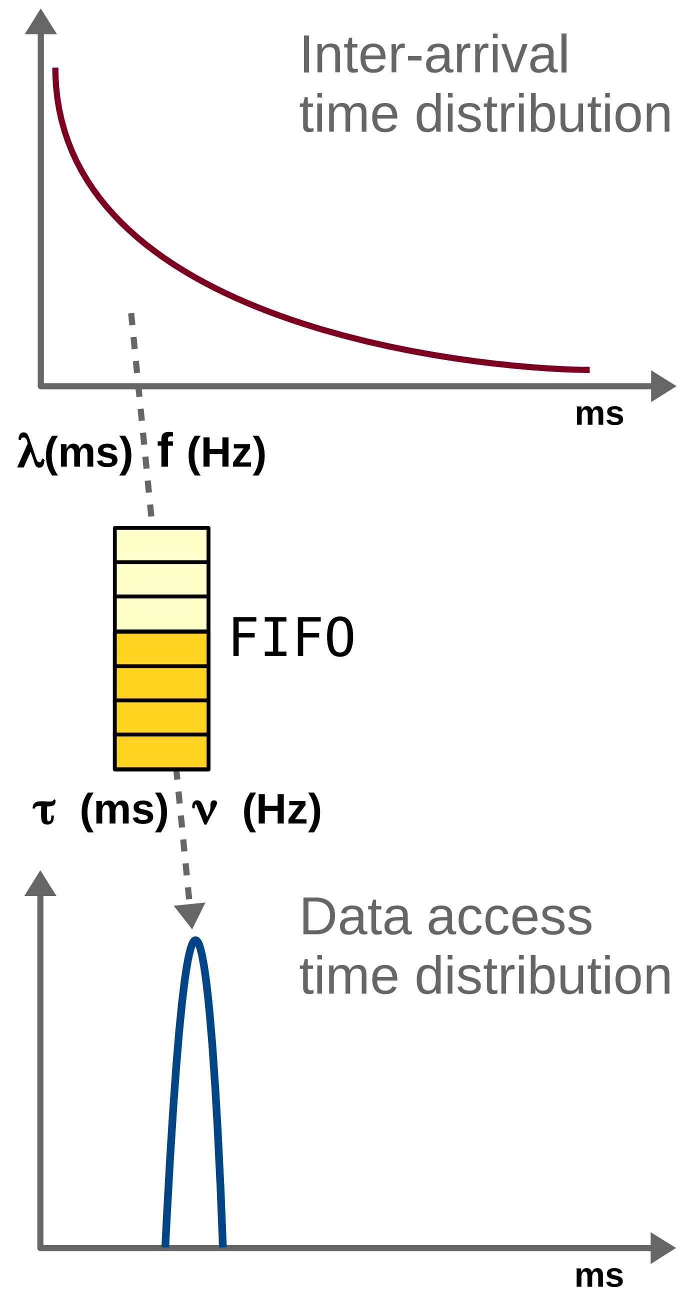
De-randomization

- What if we were able to make the system more **deterministic** and less dependent on the arrival time of our signals?
  - Then we could ensure that events don't arrive when the system is busy
  - This is called **de-randomization**
- How it can be achieved?



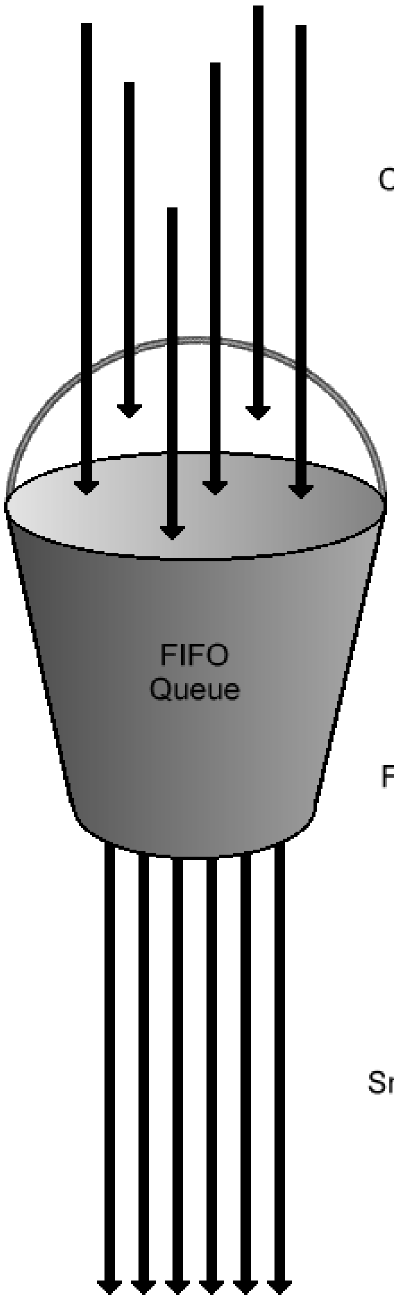
De-randomization

- What if we were able to make the system more **deterministic** and less dependent on the arrival time of our signals?
  - Then we could ensure that events don’t arrive when the system is busy
  - This is called **de-randomization**
- How it can be achieved?
  - by **buffering** the data (having a holding queue where we can slot it up to be processed)

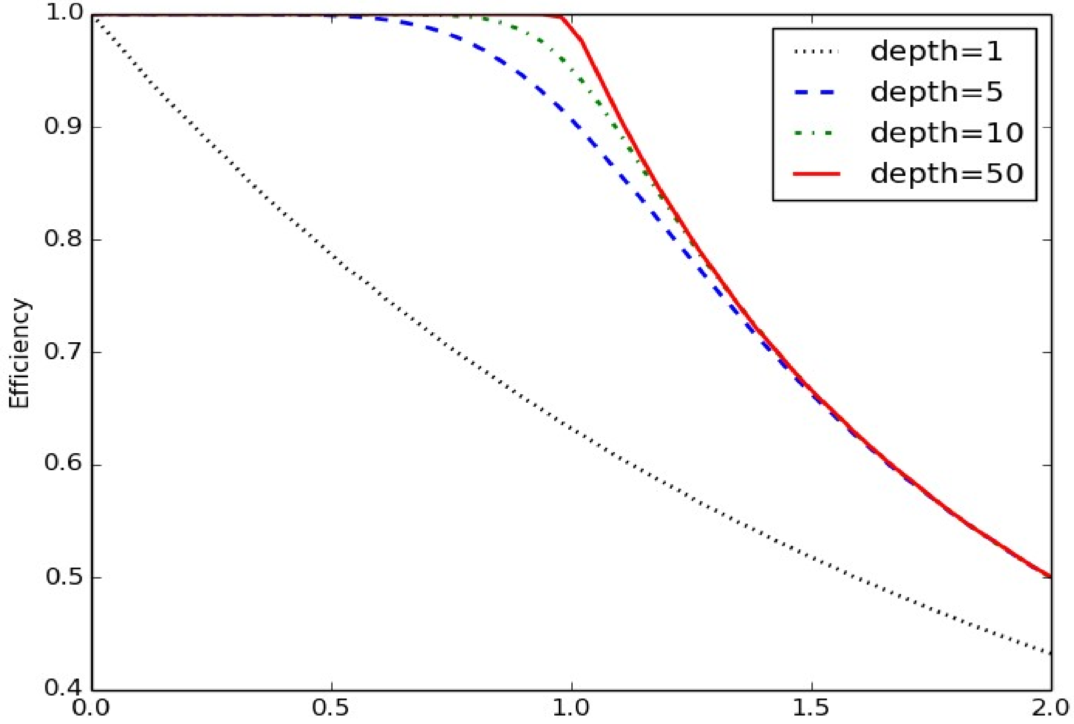


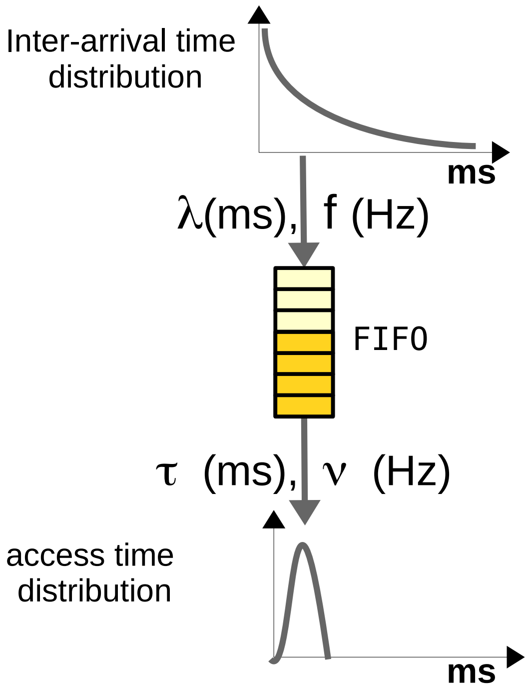
De-randomization

- What if we were able to make the system more **deterministic** and less dependent on the arrival time of our signals?
  - Then we could ensure that events don't arrive when the system is busy
  - This is called **de-randomization**
- How it can be achieved?
  - by **buffering** the data (having a holding queue where we can slot it up to be processed)



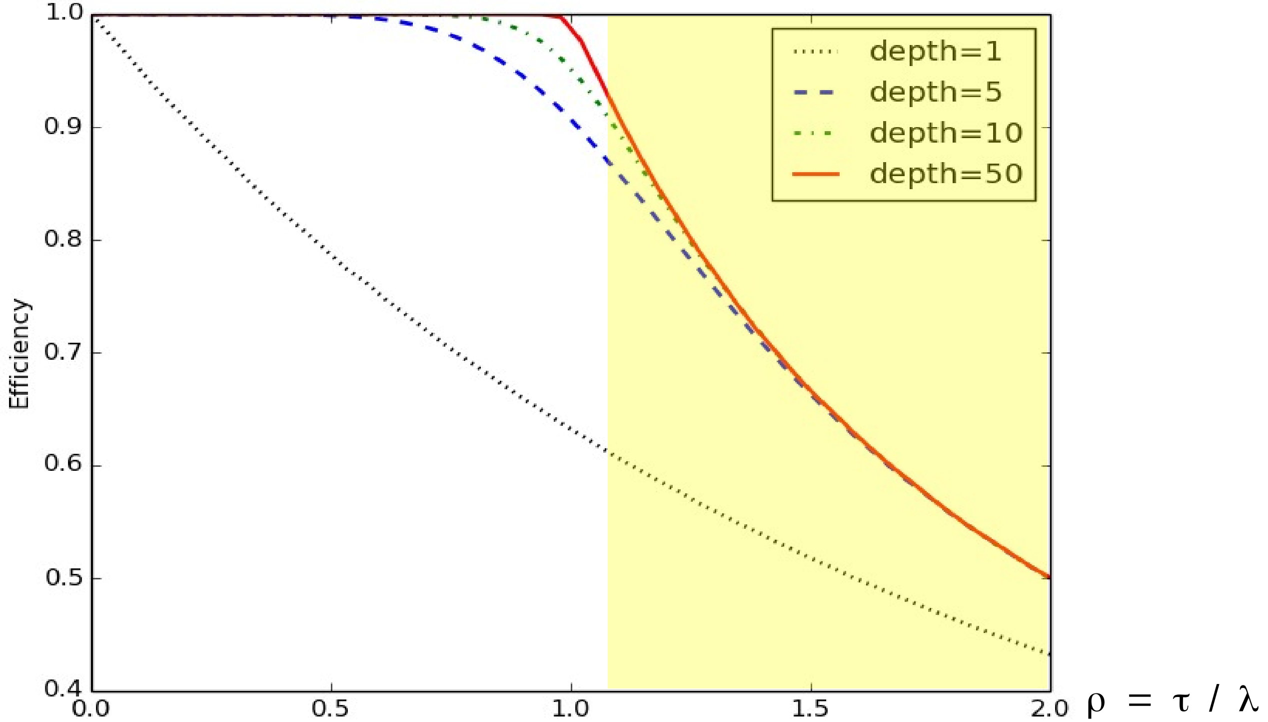
Queuing theory

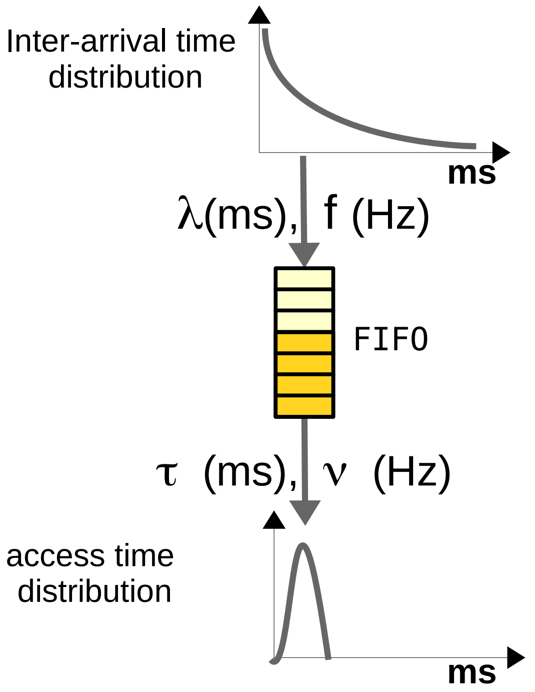




- Efficiency vs traffic intensity ($\rho = \tau / \lambda$) for different queue depths
  - $\rho > 1$: the system is overloaded ($\tau > \lambda$)
  - $\rho << 1$: the output is over-designed ($\tau << \lambda$)
  - $\rho \sim 1$: using a queue, high efficiency obtained even w/ moderate depth
- Analytic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

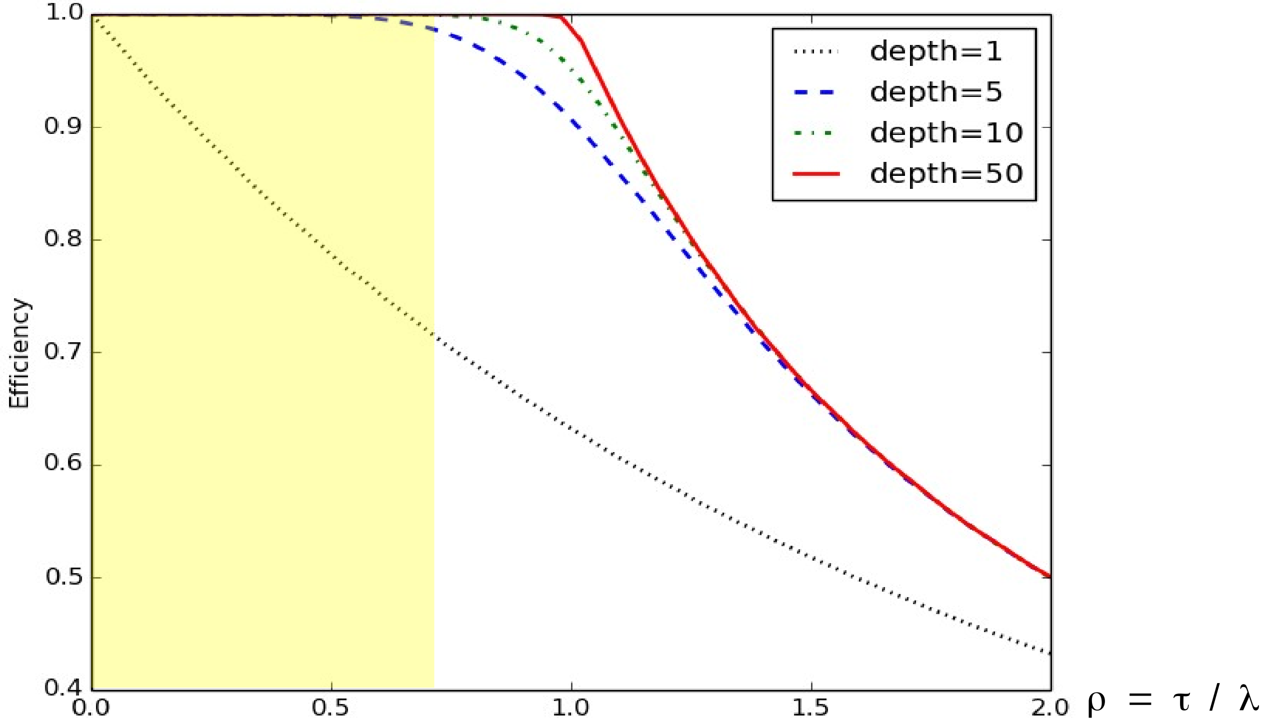
Queuing theory

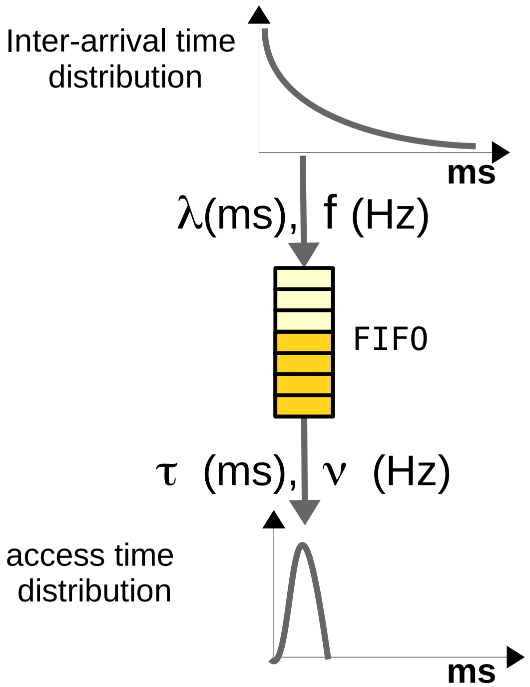




- Efficiency vs traffic intensity ($\rho = \tau / \lambda$) for different queue depths
  - $\rho > 1$: the system is overloaded ($\tau > \lambda$)
  - $\rho << 1$: the output is over-designed ($\tau << \lambda$)
  - $\rho \sim 1$: using a queue, high efficiency obtained even w/ moderate depth
- Analytic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

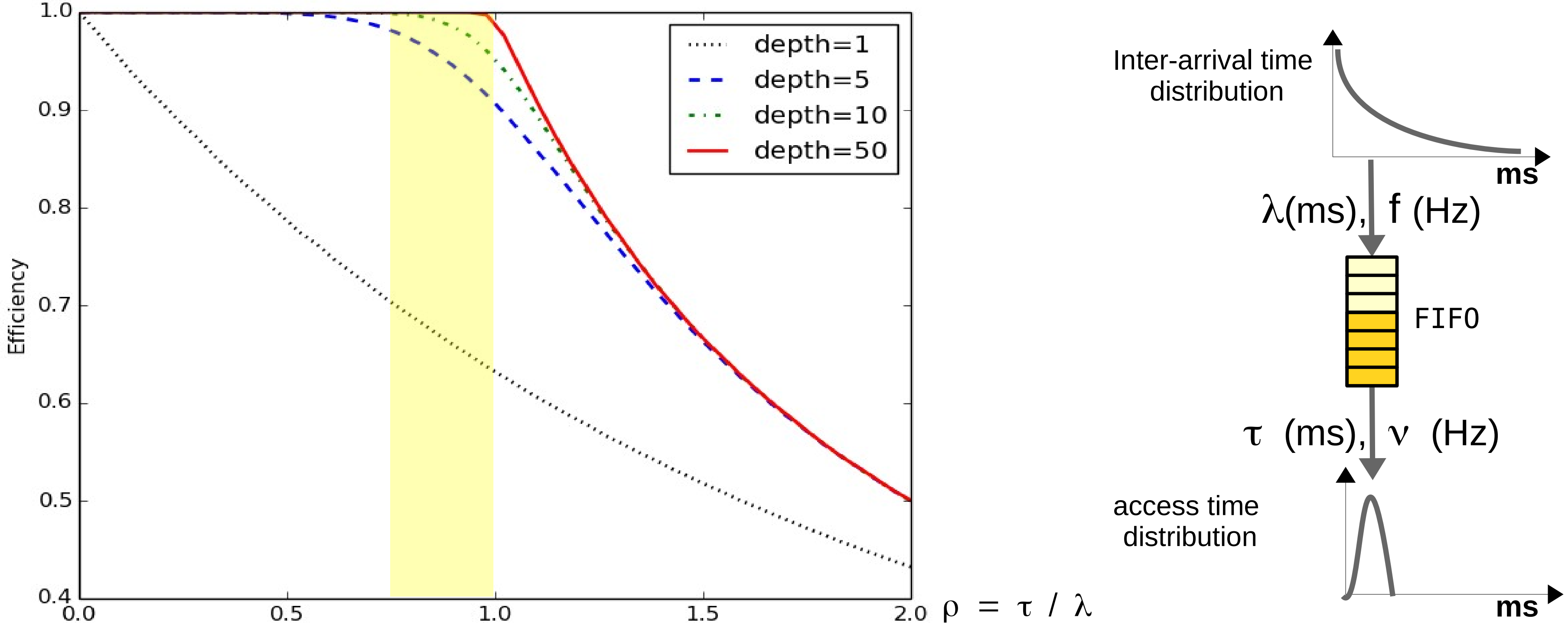
Queuing theory





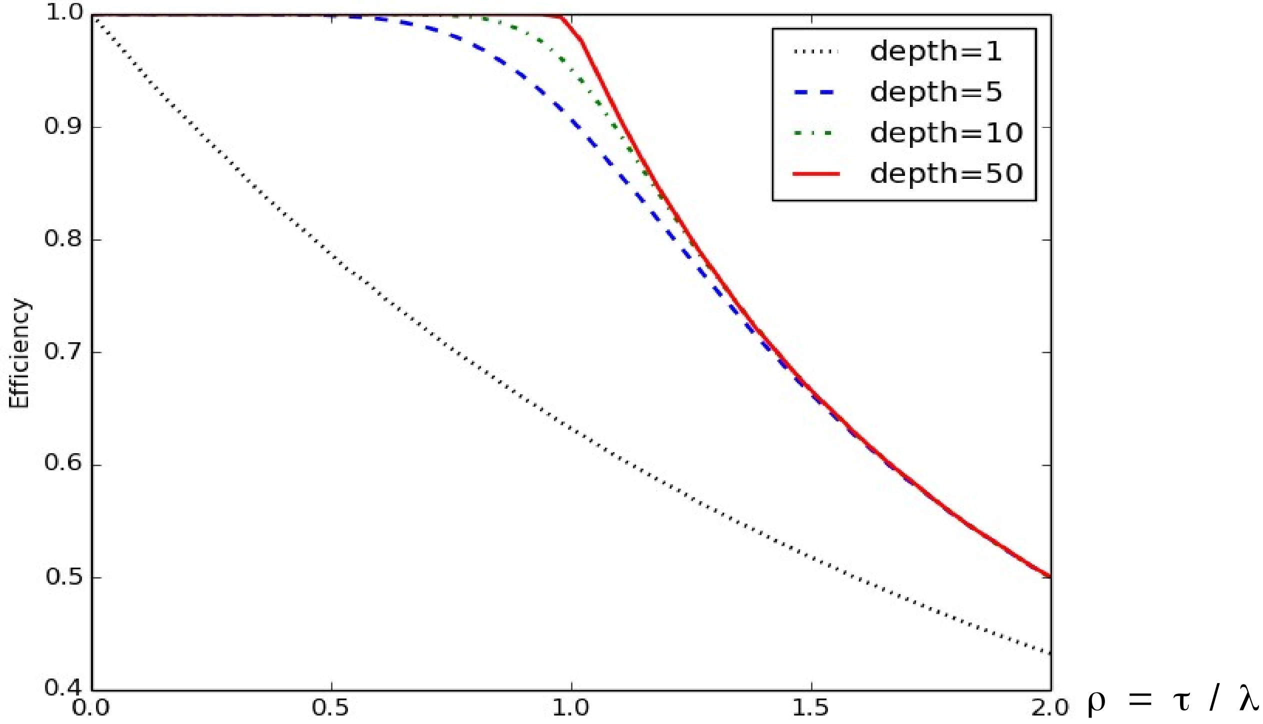
- Efficiency vs traffic intensity ($\rho = \tau / \lambda$) for different queue depths
  - **$\rho > 1$**: the system is overloaded ($\tau > \lambda$)
  - **$\rho << 1$**: the output is over-designed ($\tau << \lambda$)
  - **$\rho \sim 1$**: using a queue, high efficiency obtained even w/ moderate depth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

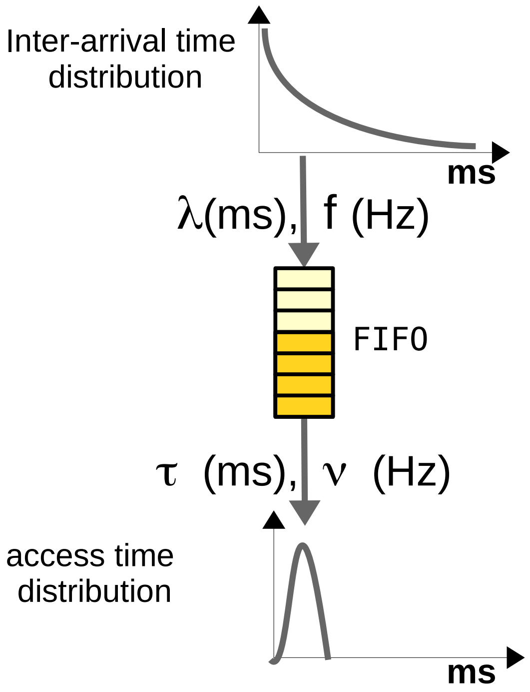
Queuing theory



- Efficiency vs traffic intensity ($\rho = \tau / \lambda$) for different queue depths
  - **ρ > 1**: the system is overloaded ($\tau > \lambda$)
  - **ρ << 1**: the output is over-designed ($\tau << \lambda$)
  - **ρ ~ 1**: using a queue, high efficiency obtained even w/ moderate depth
- Analytic calculation possible for very simple systems only
  - Otherwise MonteCarlo simulation is required

Queuing theory

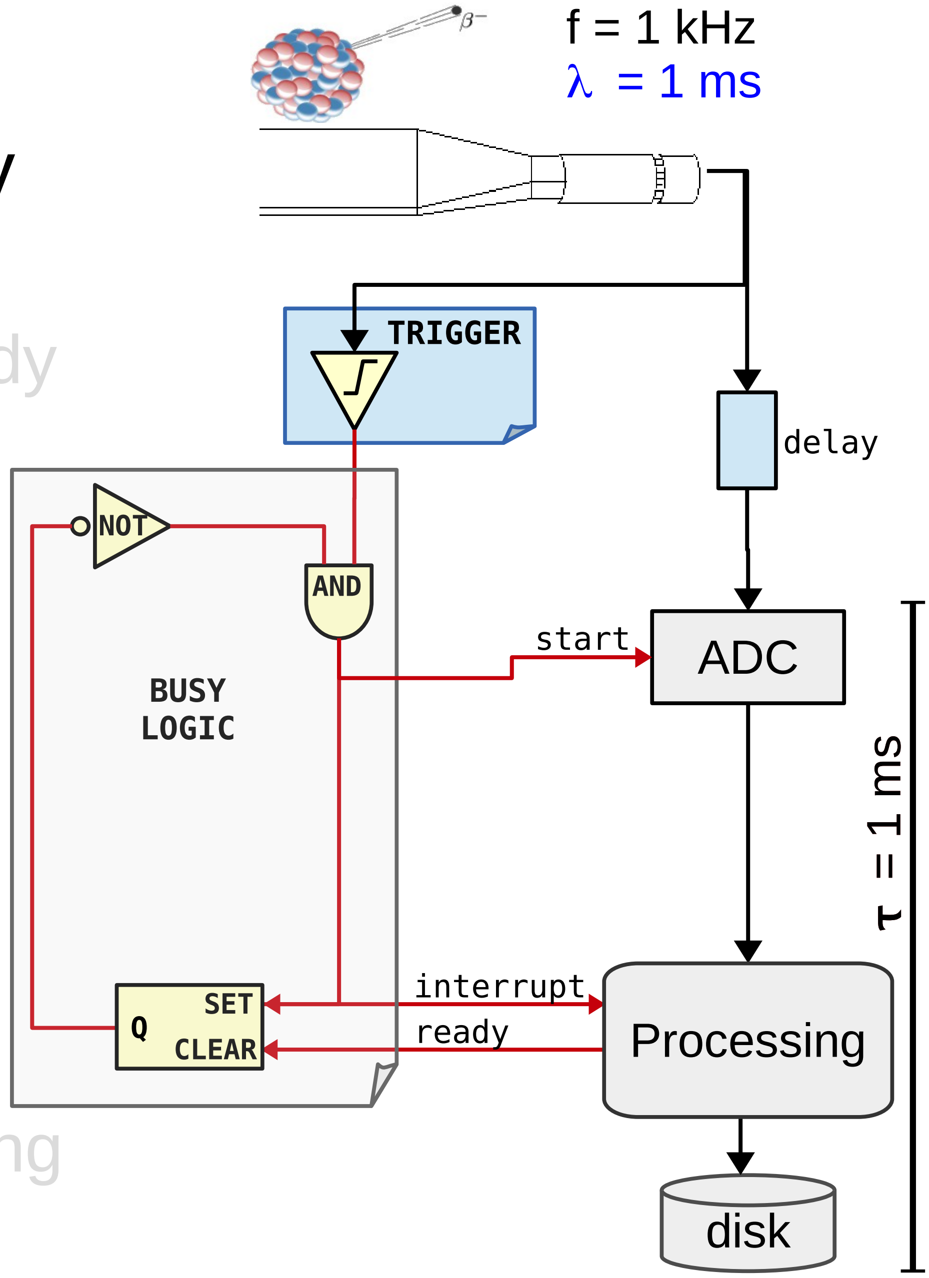




- Efficiency vs traffic intensity ($\rho = \tau / \lambda$) for different queue depths
  - **ρ > 1**: the system is overloaded ($\tau > \lambda$)
  - **ρ << 1**: the output is over-designed ($\tau << \lambda$)
  - **ρ ~ 1**: using a queue, high efficiency obtained even w/ moderate depth
- Analytic calculation possible for very simple systems only
  - Otherwise MonteCarlo simulation is required

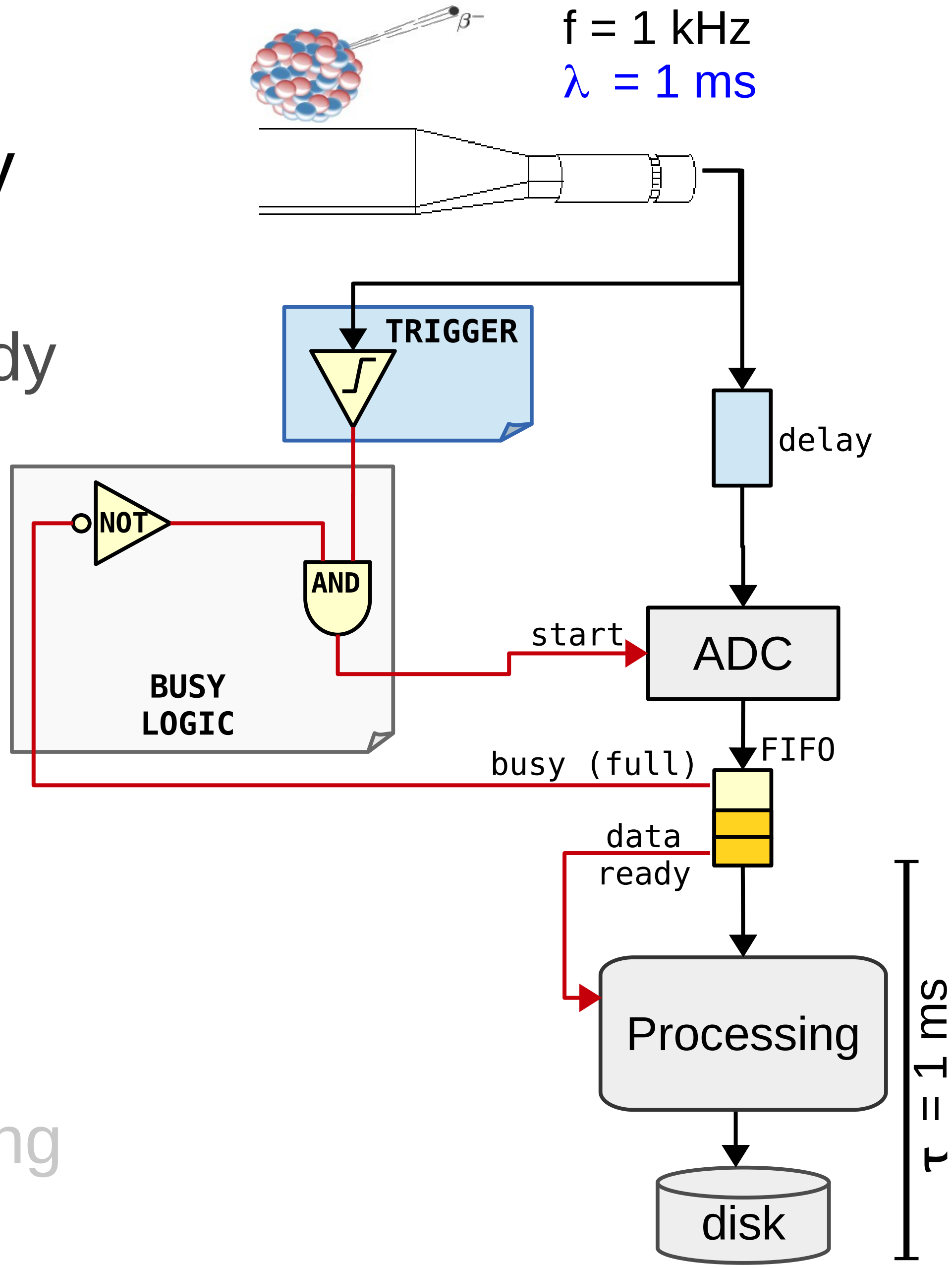
De-randomization

- Input fluctuations can be absorbed and smoothed by a queue
  - A FIFO can provide a ~steady and **de-randomized** output rate
  - The effect of the queue depends on its depth
- Busy is now defined by the buffer occupancy
  - Processor pulls data from the buffer at fixed rate, separating the event receiving and data processing steps



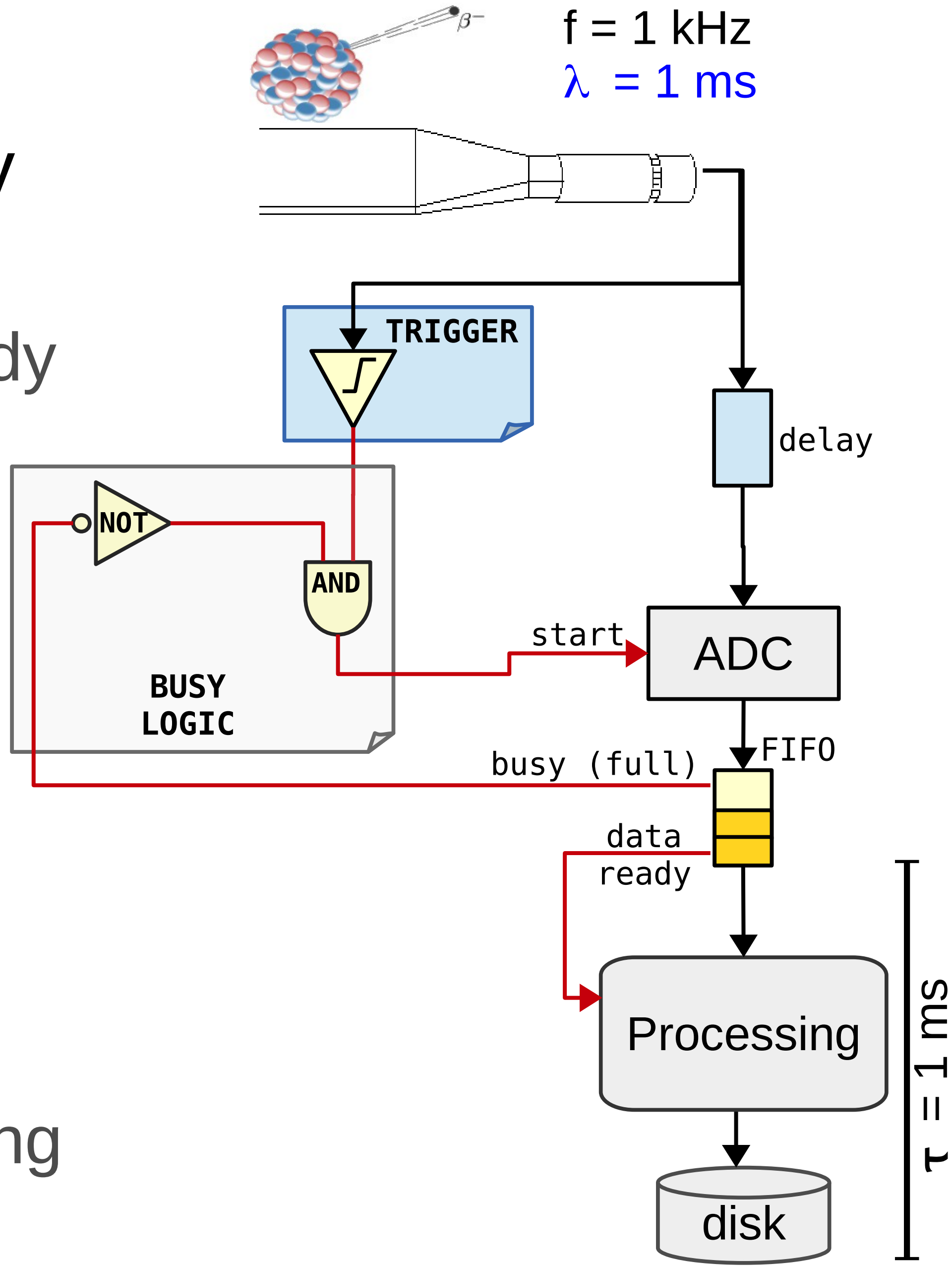
De-randomization

- Input fluctuations can be absorbed and smoothed by a queue
  - A FIFO can provide a ~steady and **de-randomized** output rate
  - The effect of the queue depends on its depth
- Busy is now defined by the buffer occupancy
  - Processor pulls data from the buffer at fixed rate, separating the event receiving and data processing steps



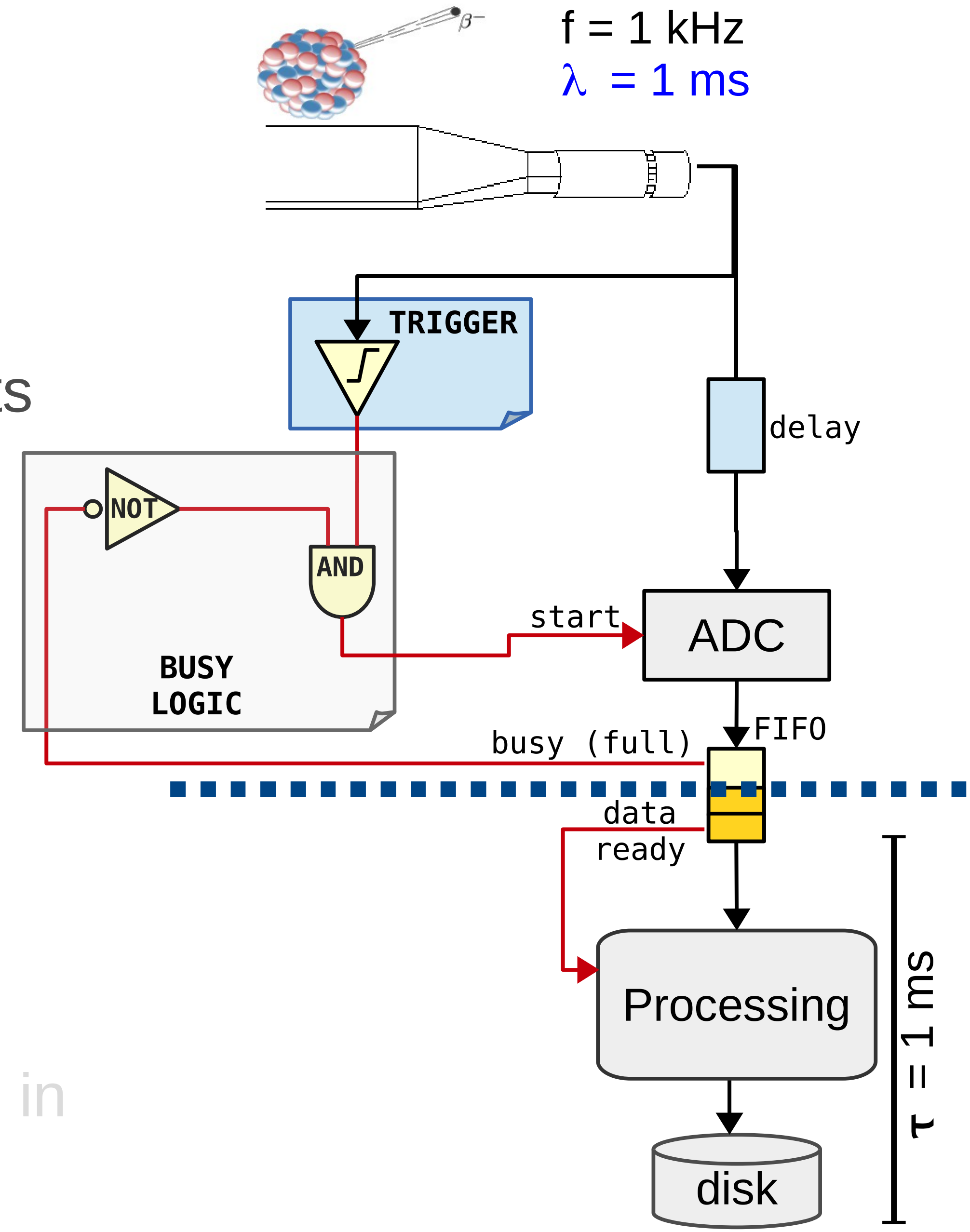
De-randomization

- Input fluctuations can be absorbed and smoothed by a queue
  - A FIFO can provide a ~steady and **de-randomized** output rate
  - The effect of the queue depends on its depth
- Busy is now defined by the buffer **occupancy**
  - Processor pulls data from the buffer at fixed rate, separating the event receiving and data processing steps



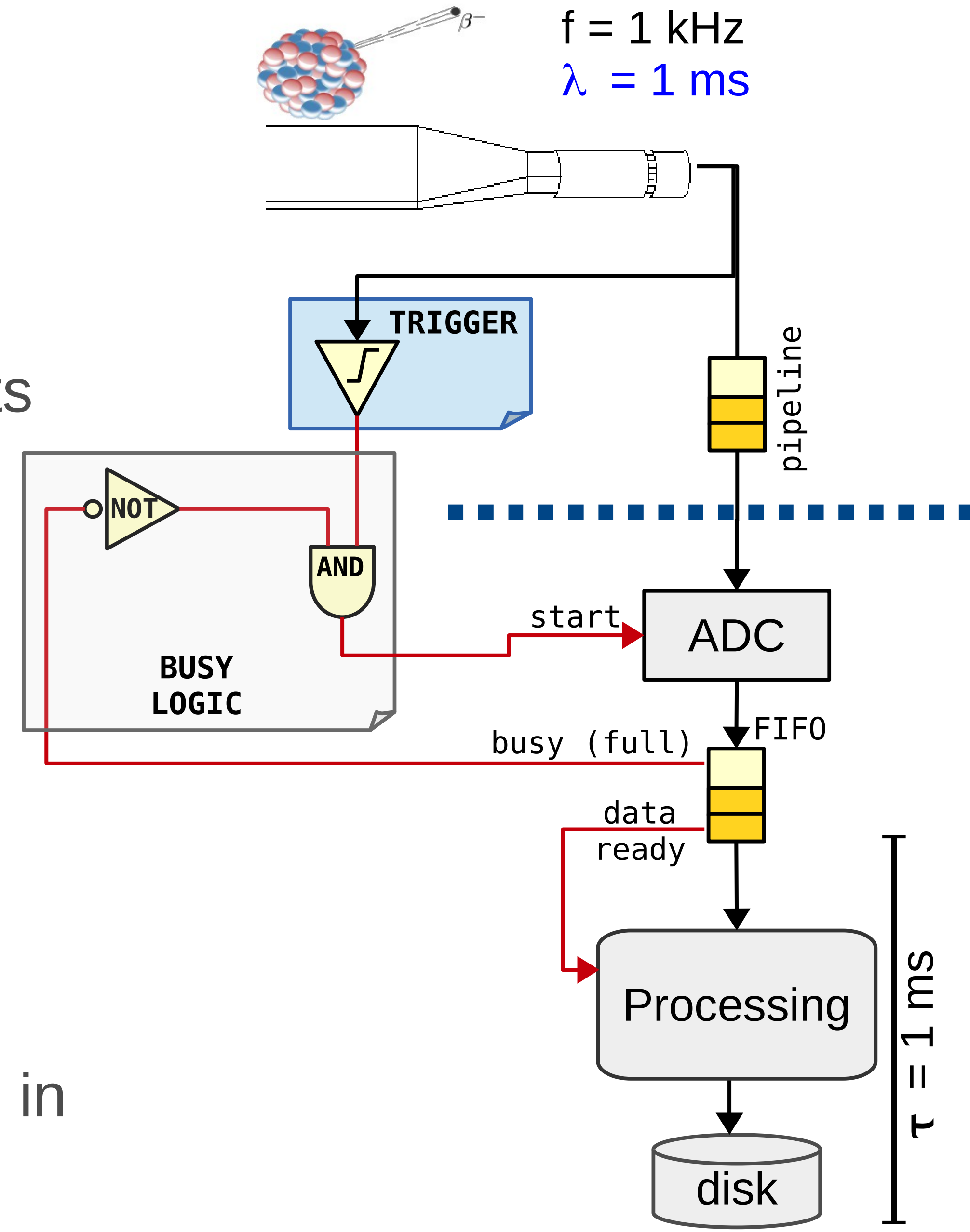
De-randomization summary

- The FIFO **decouples** the low latency front-end from the data processing
  - Minimize the amount of “unnecessary” fast components
- ~100% efficiency w/ minimal deadtime achievable if
  - ADC can operate at rate >> f
  - Data processing and storing operate at a rate ~ f
- Could the delay be replaced with a “FIFO”?
  - Analog pipelines, heavily used in LHC DAQs



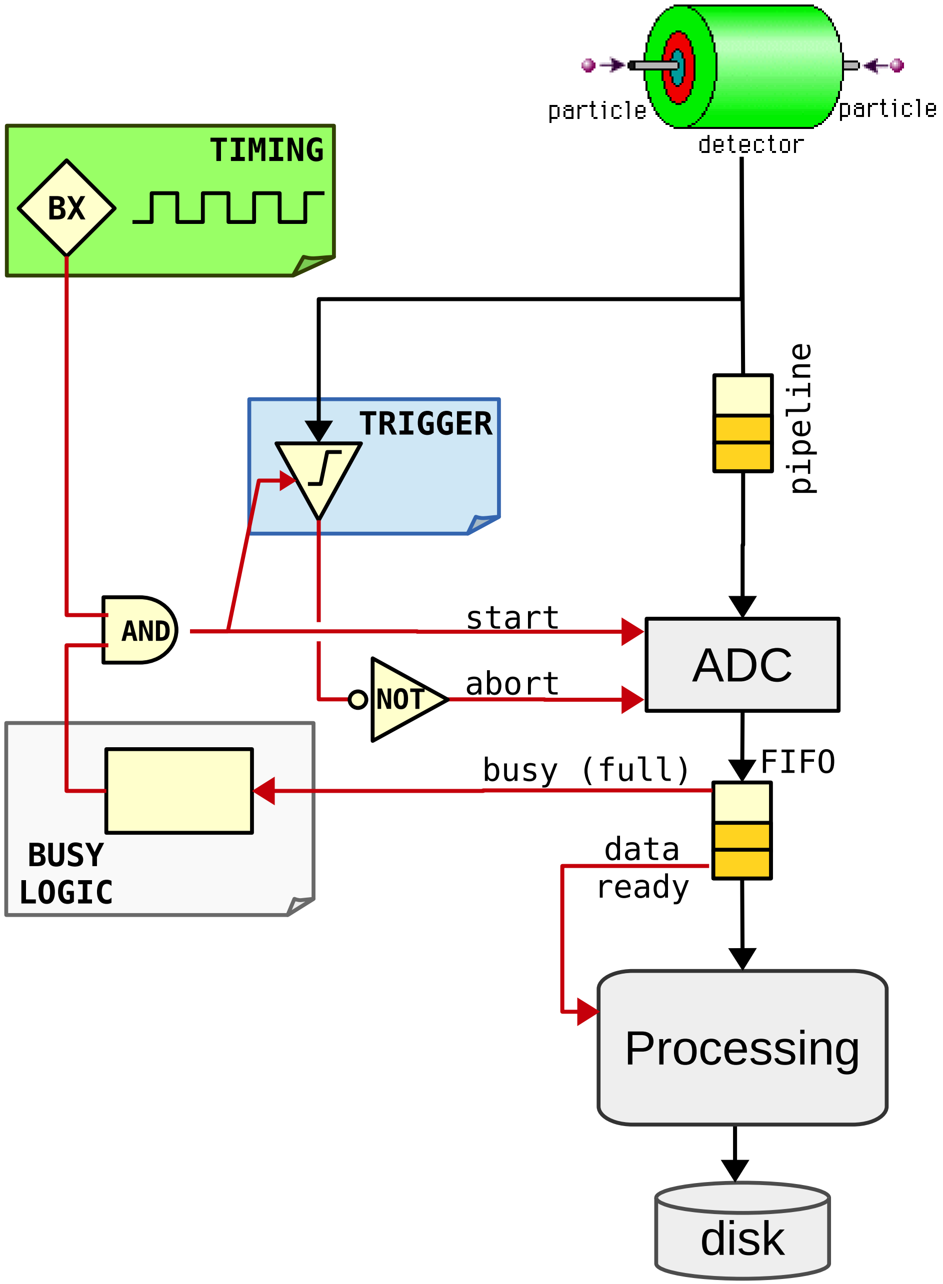
De-randomization summary

- The FIFO **decouples** the low latency front-end from the data processing
  - Minimize the amount of “unnecessary” fast components
- ~100% efficiency w/ minimal deadtime achievable if
  - ADC can operate at rate >> f
  - Data processing and storing operate at a rate ~ f
- Could the delay be replaced with a “FIFO”?
  - Analog pipelines, heavily used in LHC DAQs



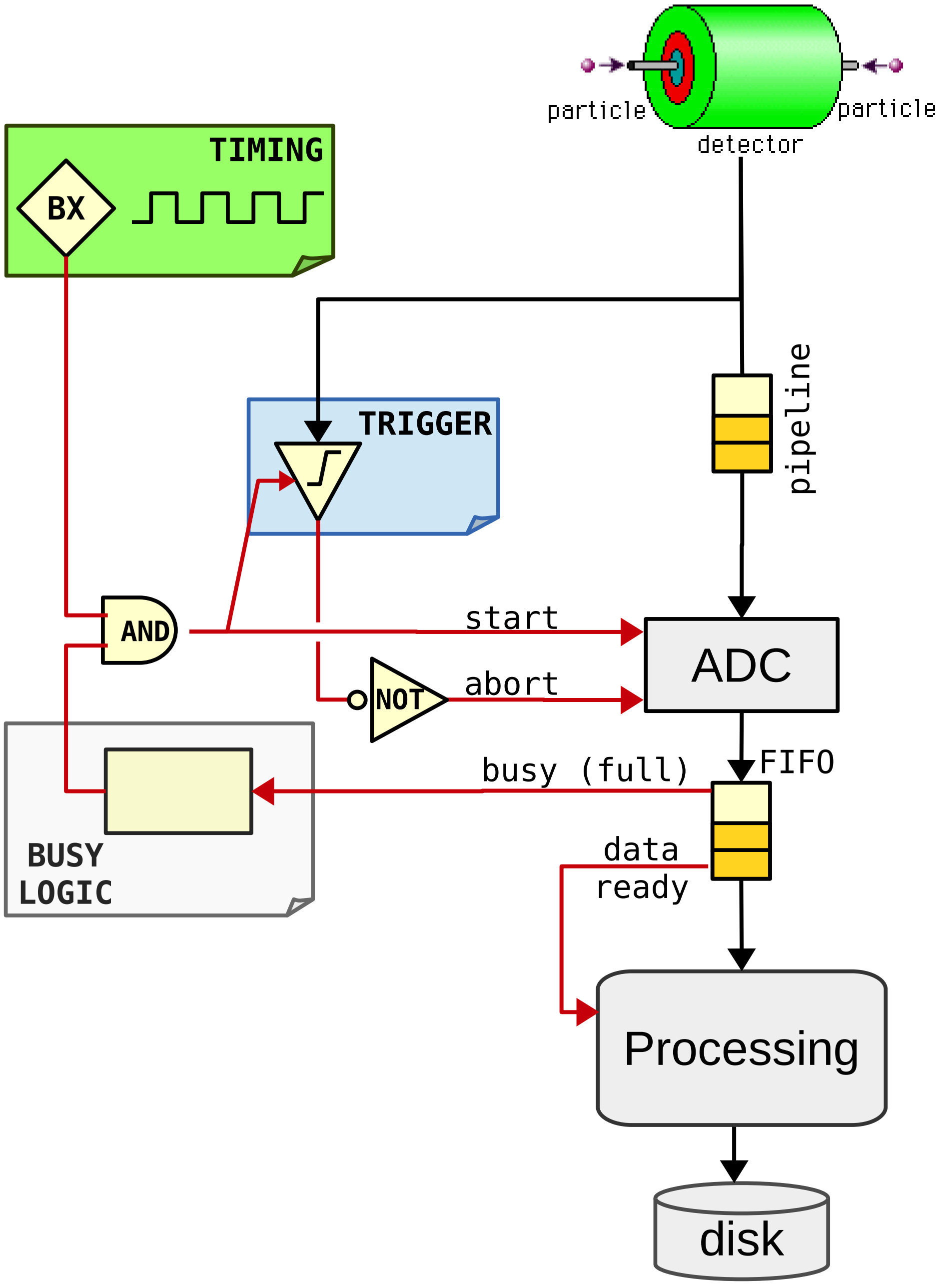
Collider setup

- Do we need de-randomization buffers also in collider setups?
  - Particle collisions are synchronous
  - But the time distribution of triggers is random: good events are unpredictable
- De-randomization still needed
- More complex busy logic to protect buffers and detectors
  - Eg: accept n events every m bunch crossings
  - Eg: prevent some dangerous trigger patterns



Collider setup

- Do we need de-randomization buffers also in collider setups?
  - Particle collisions are synchronous
  - But the time distribution of triggers is random: good events are unpredictable
- De-randomization still needed
- More complex busy logic to protect buffers and detectors
  - Eg: accept n events every m bunch crossings
  - Eg: prevent some dangerous trigger patterns



Outline

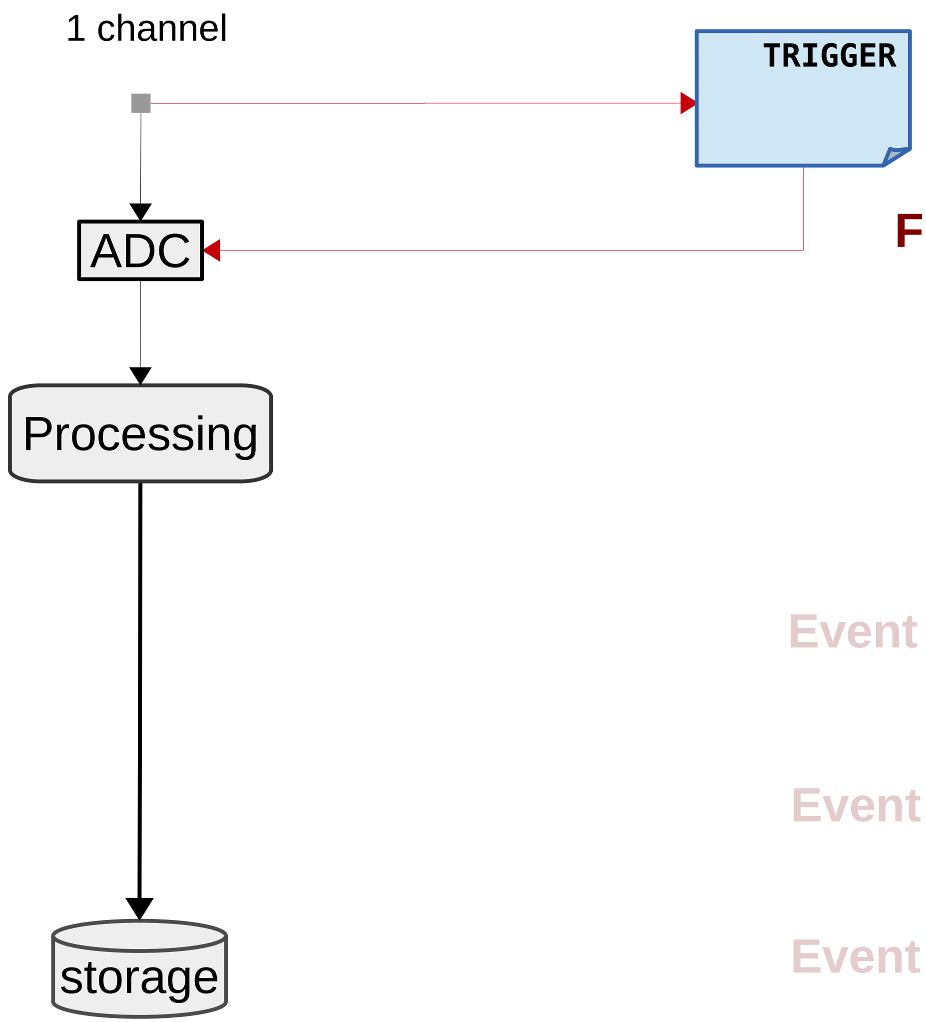
- Introduction
  - What is DAQ?
  - Overall framework
- Basic DAQ concepts
  - Digitization, Latency
  - Deadtime, Busy, Backpressure
  - De-randomization
- Scaling up
  - Readout and Event Building
  - Buses vs Network
- Data encoding

ScalingUp@isotdaq2024

- *TDAQ design: from test beam to medium size experiment*
  - **Roberto** Ferrari
- *TDAQ for the LHC experiments and upgrades*
  - **Francesca** Pastore
- *TDAQ for space experiments*
  - **Valentina** Scotti
- *An introduction to medical imaging devices*
  - **Martin** Lothar Purschke

Adding more channels

- Adding more channels requires a hierarchical structure committed to the data handling and conveyance



Front-End | data digitization, data buffering

Readout | data extraction, data formatting, data buffering

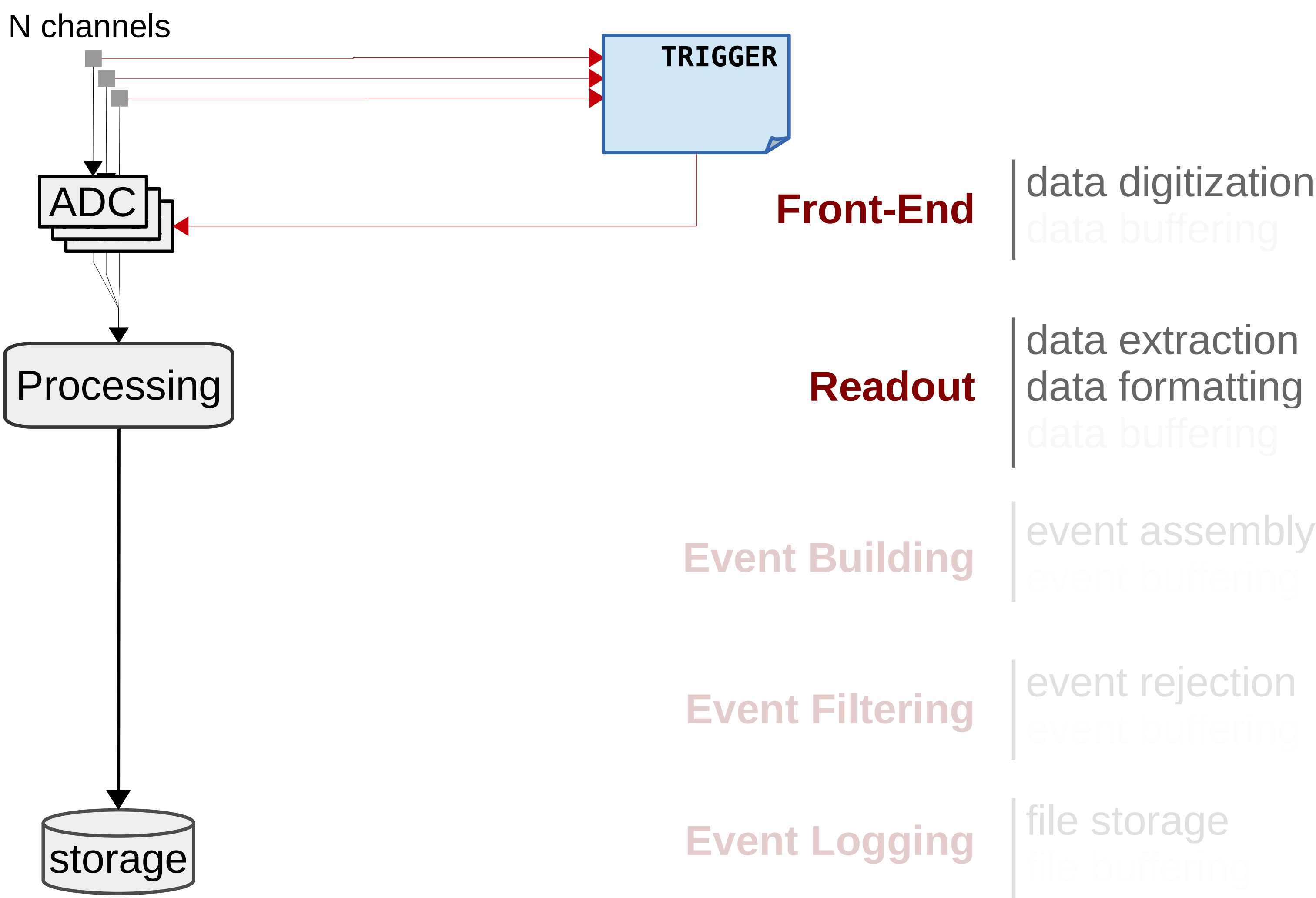
Event Building | event assembly, event buffering

Event Filtering | event rejection, event buffering

Event Logging | file storage, file buffering

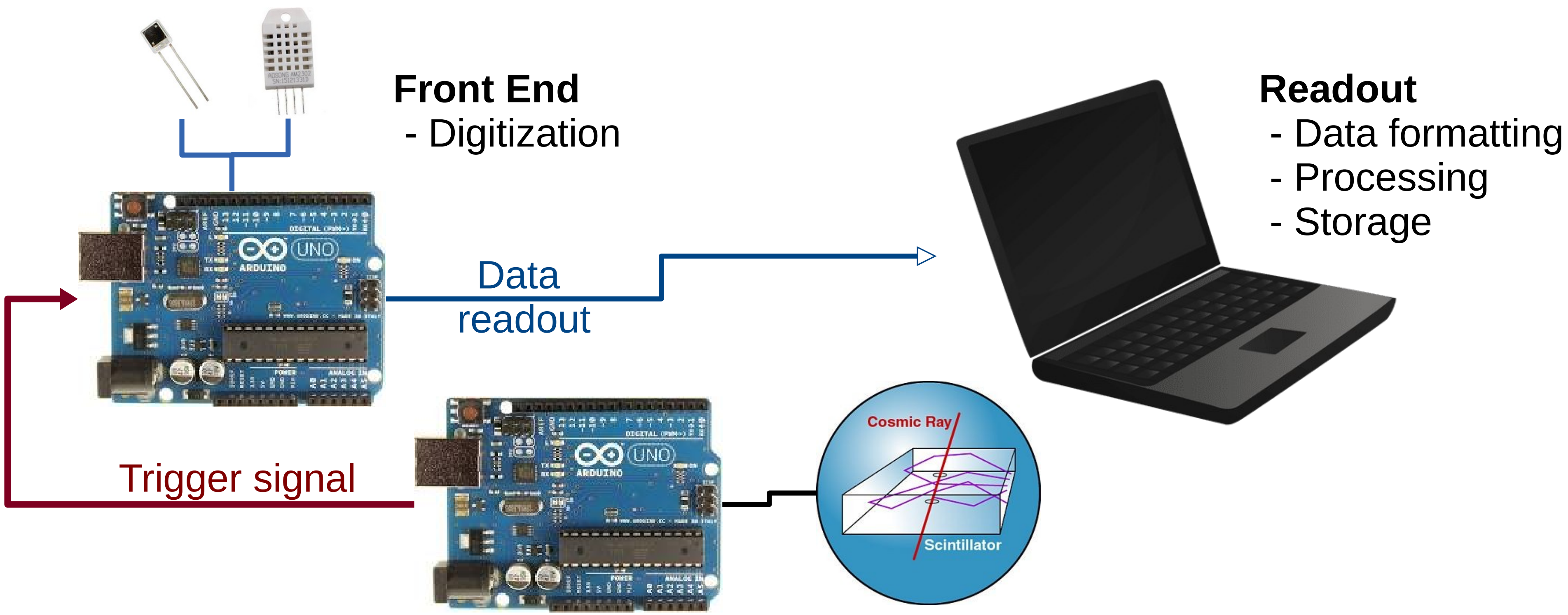
Adding more channels

- Adding more channels requires a hierarchical structure committed to the data handling and conveyance



For example

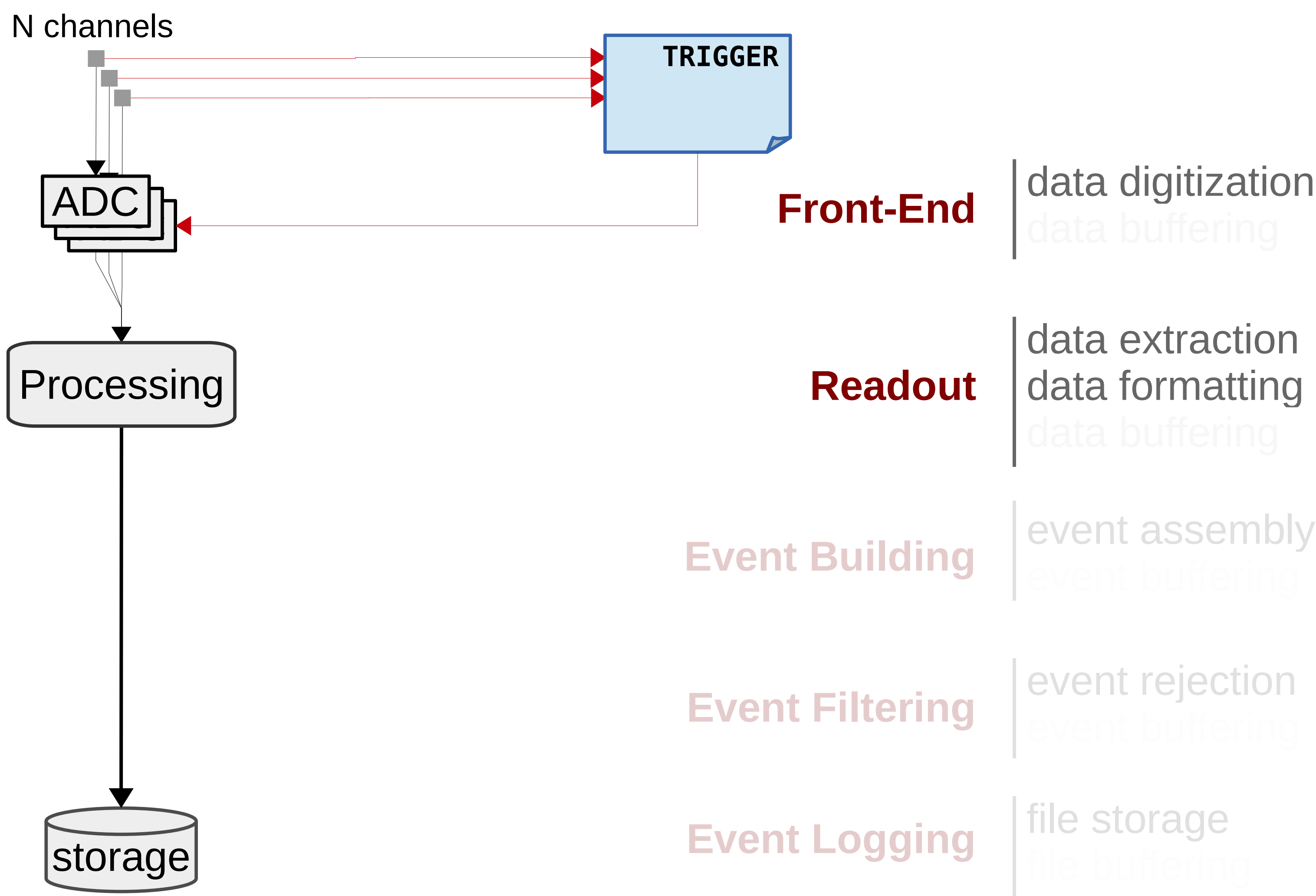
- Minimal setup with Arduino and a PC
  - Arduino has ADCs to read sensors



- *Microcontrollers*
  - **Mauricio** Feo
- *Microcontrollers Exercise*
  - Lab 10

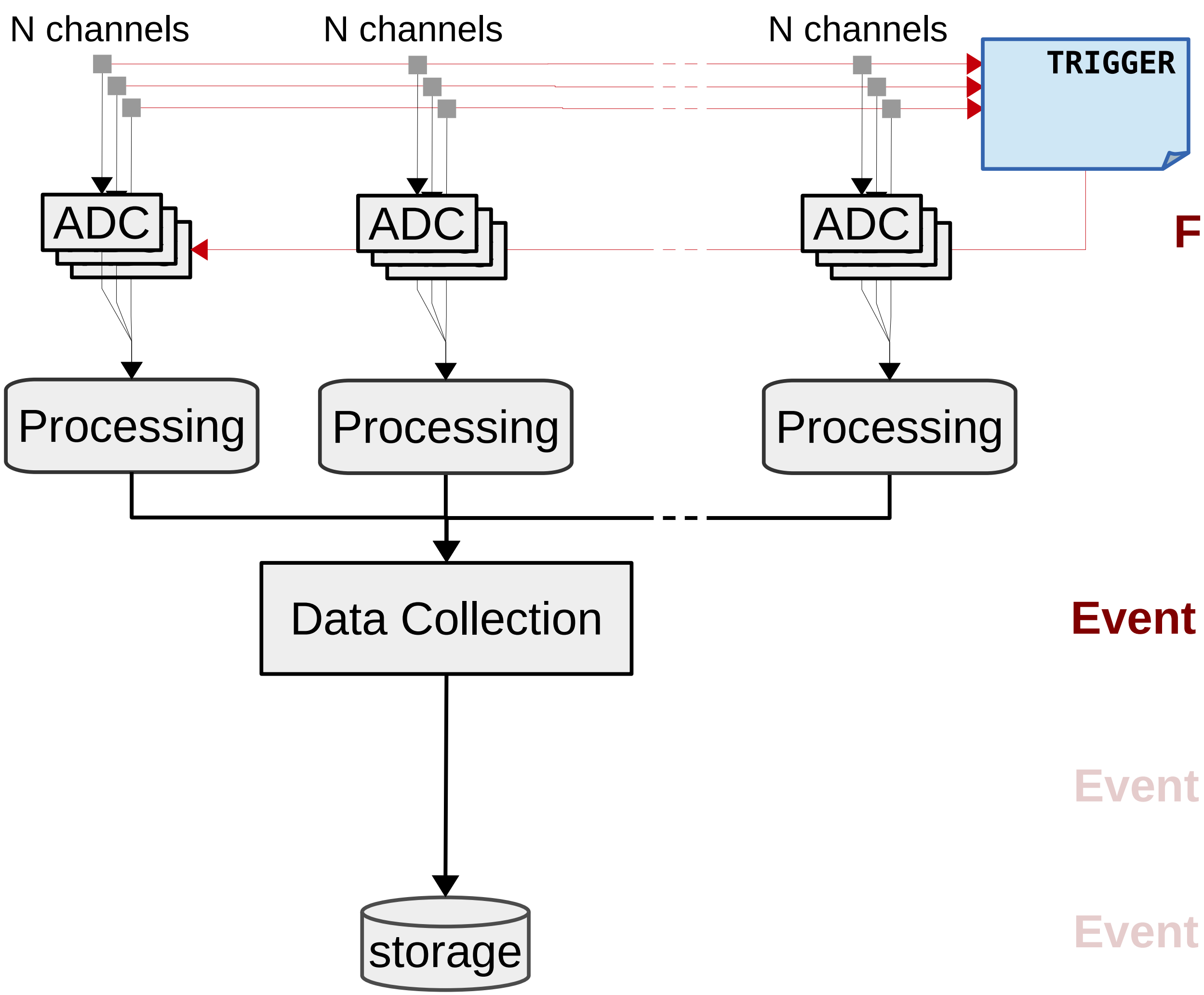
Adding more channels

- Adding more channels requires a hierarchical structure committed to the data handling and conveyance



Adding more channels

- Adding more channels requires a hierarchical structure committed to the data handling and conveyance



Front-End | data digitization, data buffering

Readout | data extraction, data formatting, data buffering

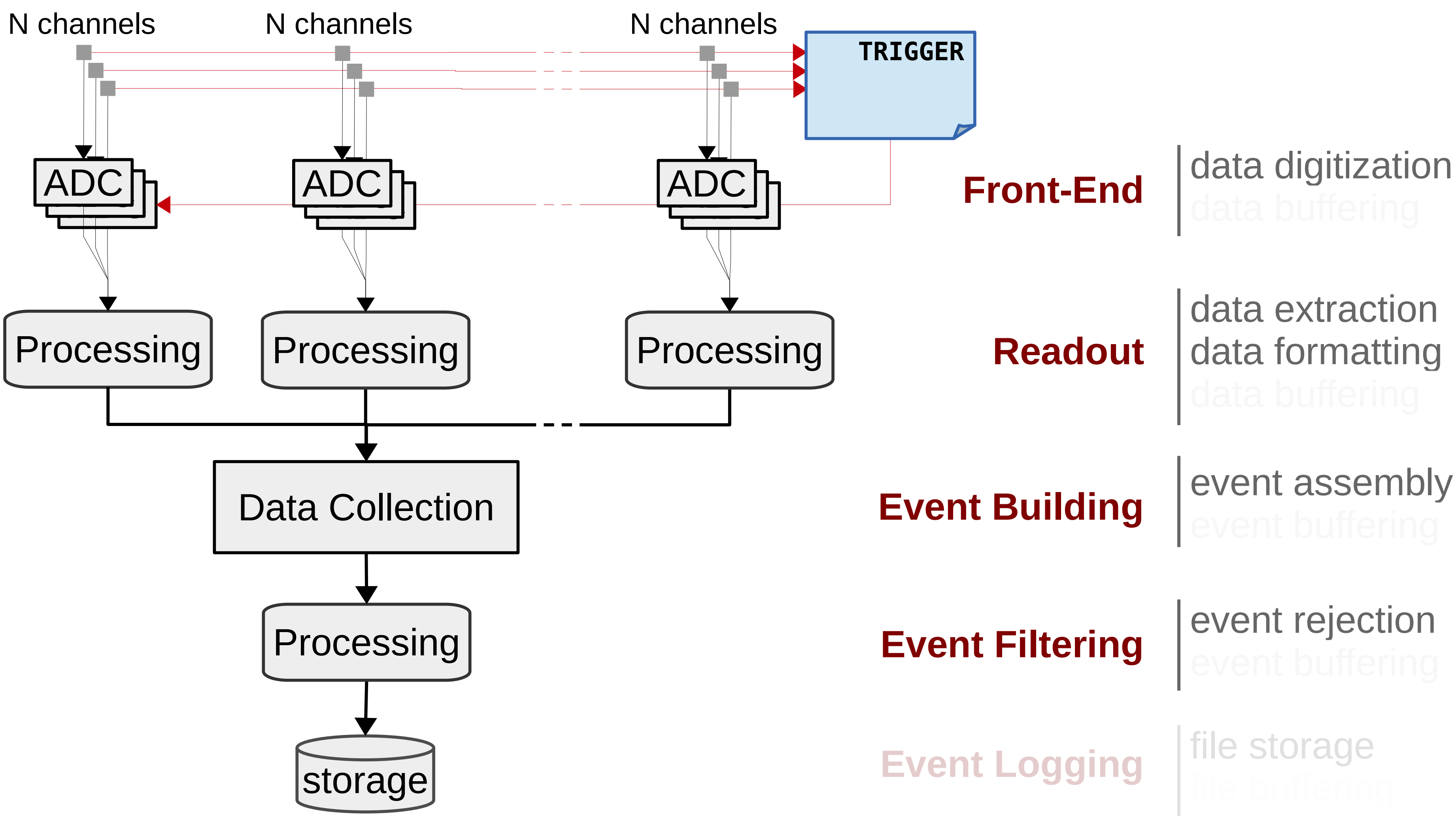
Event Building | event assembly, event buffering

Event Filtering | event rejection, event buffering

Event Logging | file storage, file buffering

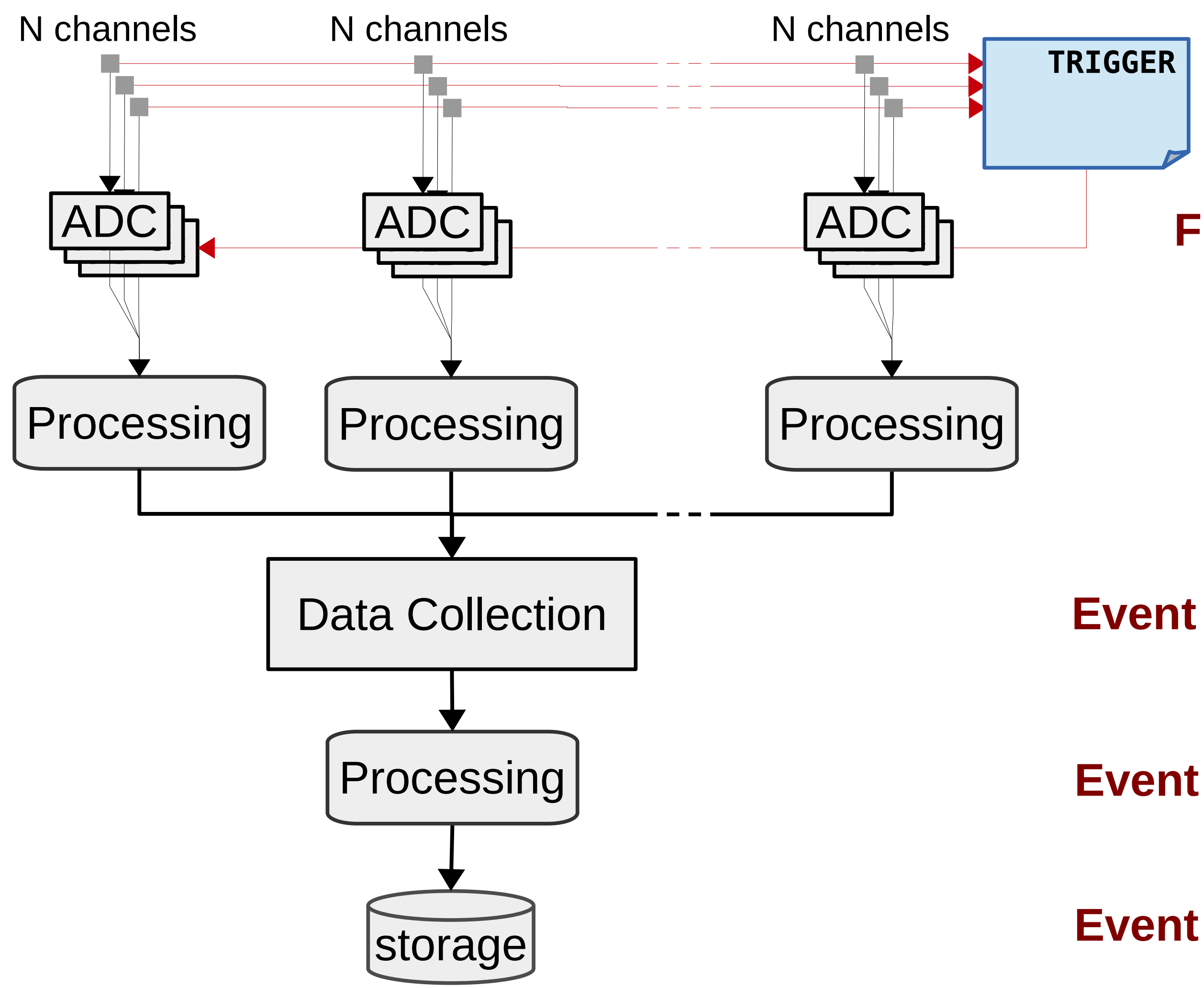
Adding more channels

- Adding more channels requires a hierarchical structure committed to the data handling and conveyance



Adding more channels

- Adding more channels requires a hierarchical structure committed to the data handling and conveyance



Front-End | data digitization, data buffering

Readout | data extraction, data formatting, data buffering

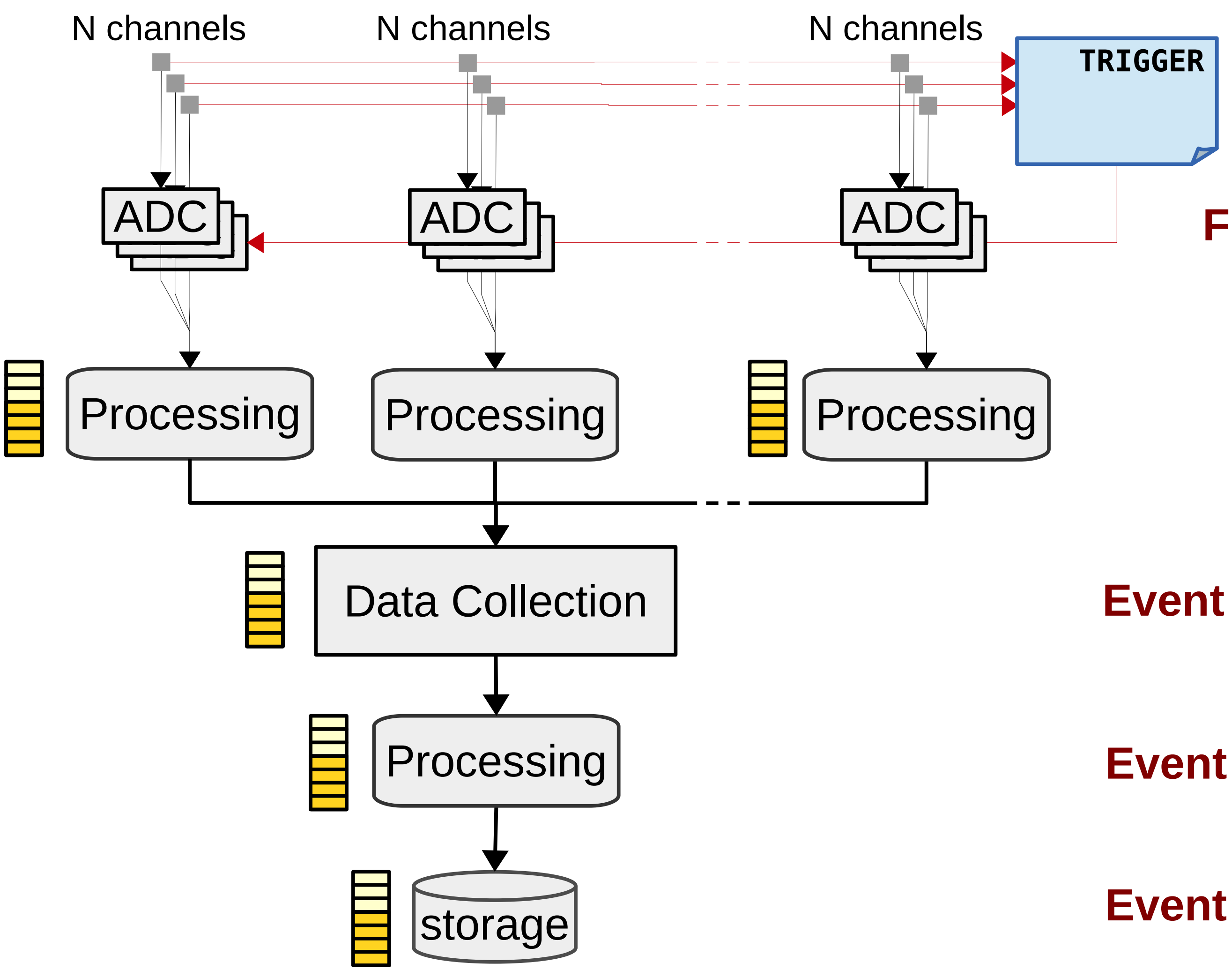
Event Building | event assembly, event buffering

Event Filtering | event rejection, event buffering

Event Logging | file storage, file buffering

Adding more channels

- **Buffering** usually needed at every level
  - DAQ can be seen as a multi level buffering system



Front-End | data digitization, data **buffering**

Readout | data extraction, data formatting, data **buffering**

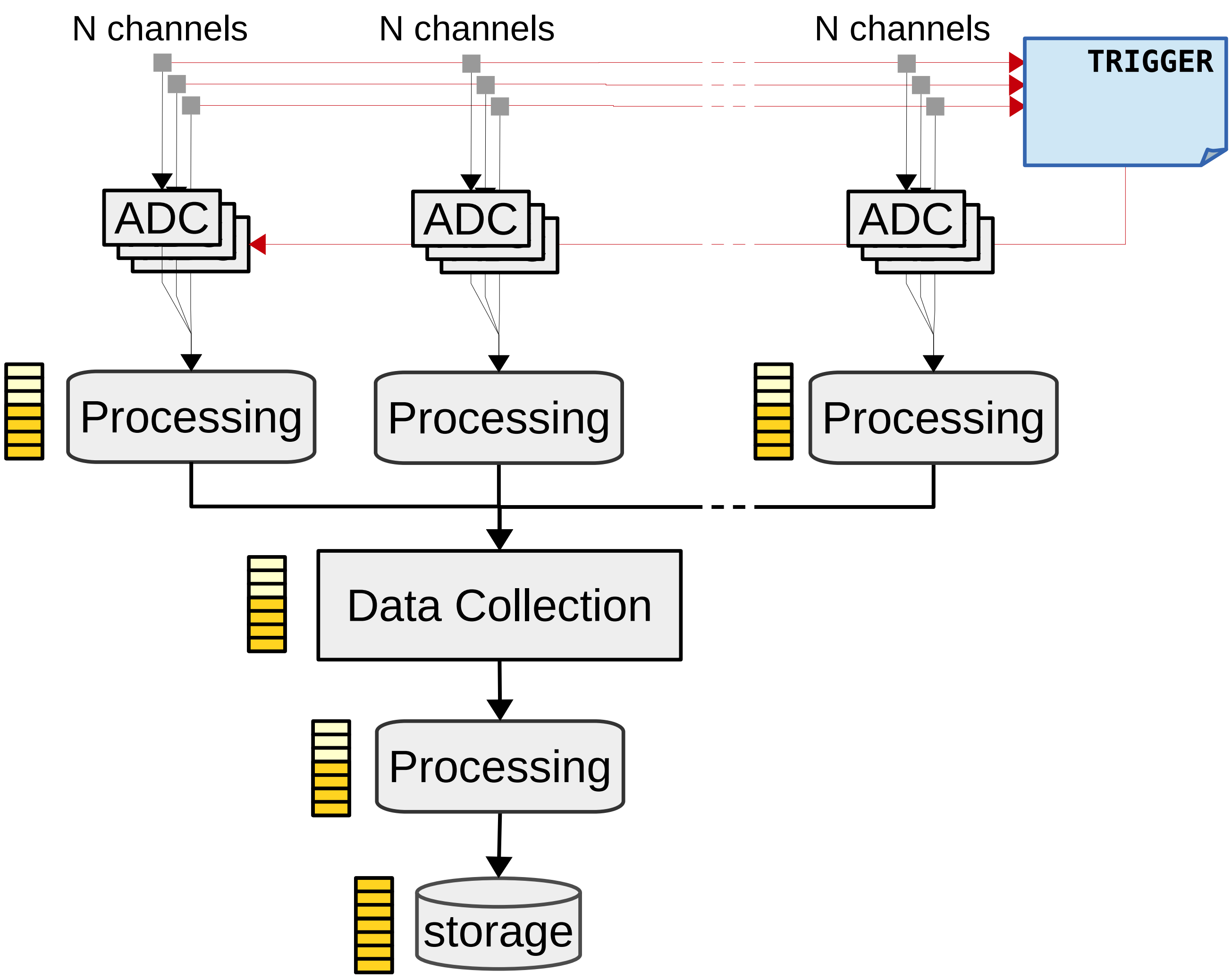
Event Building | event assembly, event **buffering**

Event Filtering | event rejection, event **buffering**

Event Logging | file storage, file **buffering**

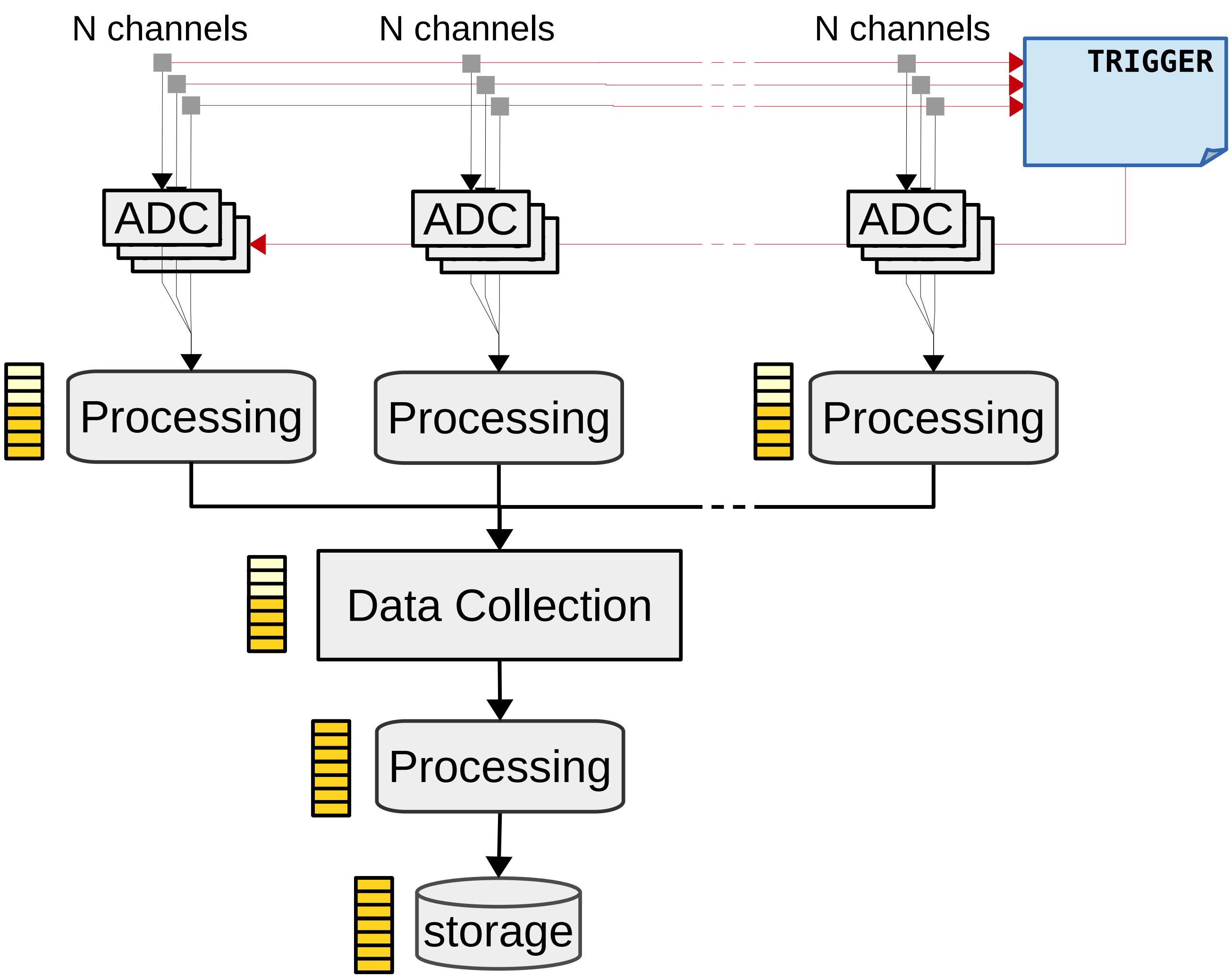
Backpressure

- If a system/buffer gets saturated
  - the “pressure” is propagated upstream (**back-pressure**)



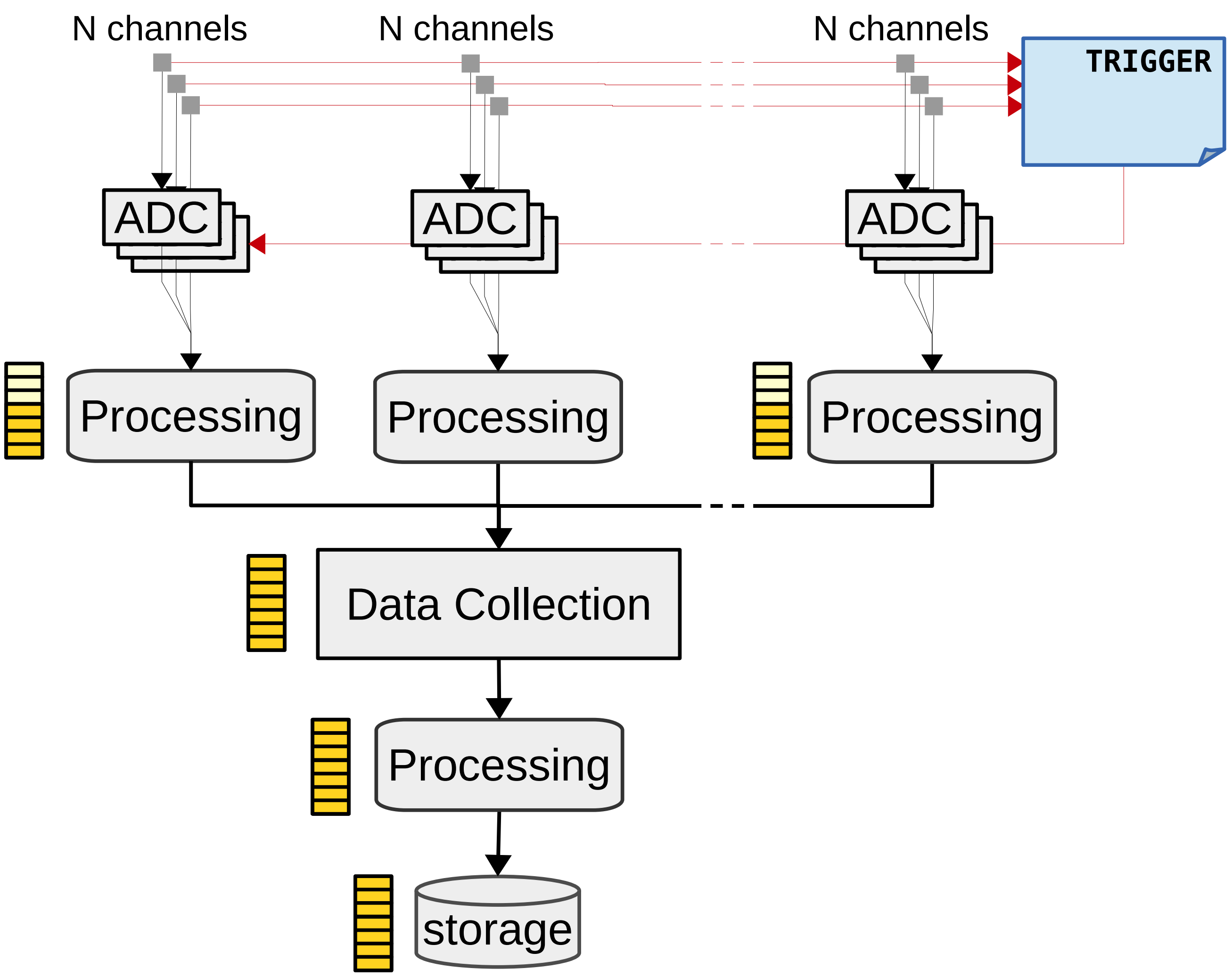
Backpressure

- If a system/buffer gets saturated
  - the “pressure” is propagated upstream (**back-pressure**)



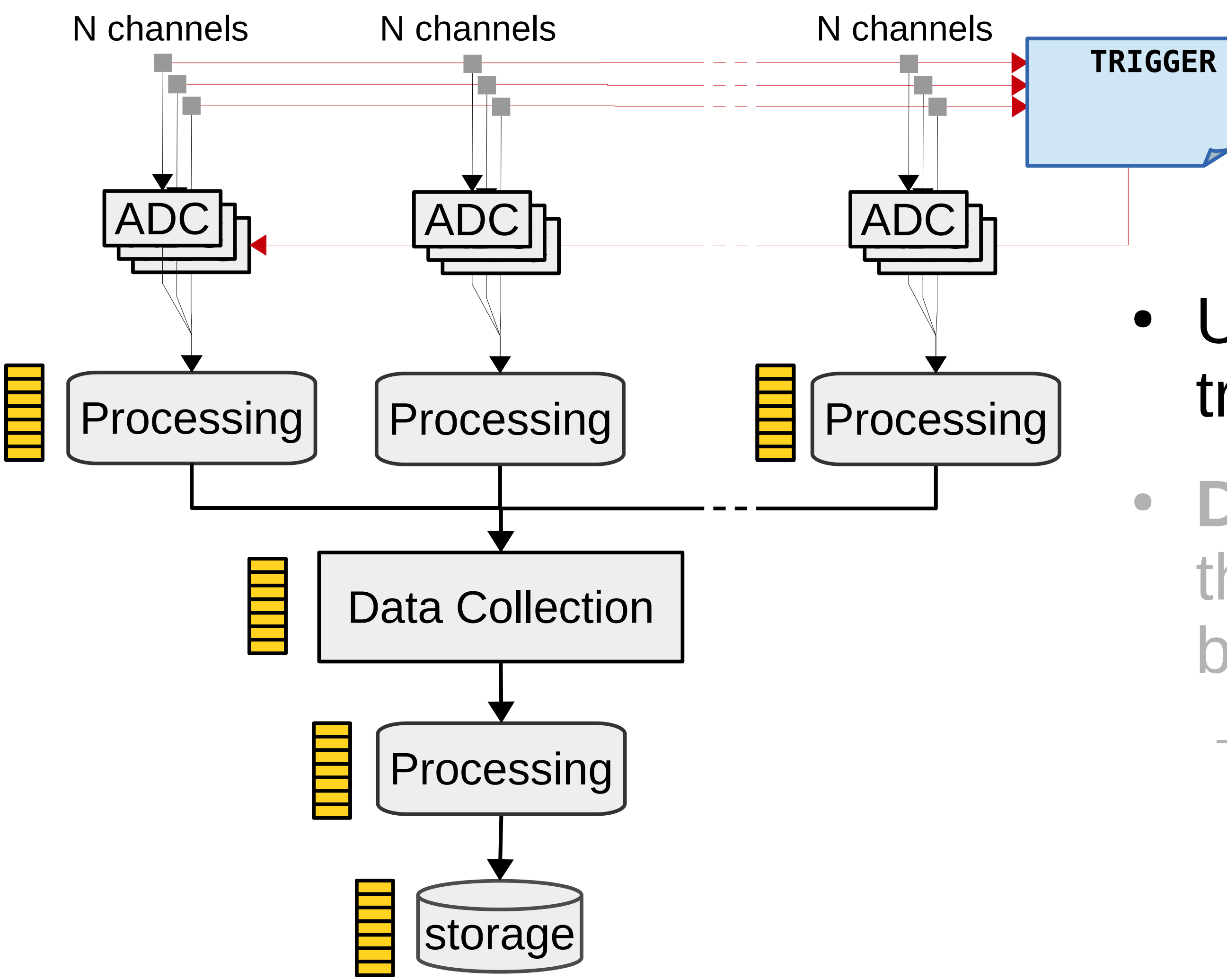
Backpressure

- If a system/buffer gets saturated
  - the “pressure” is propagated upstream (**back-pressure**)



Backpressure

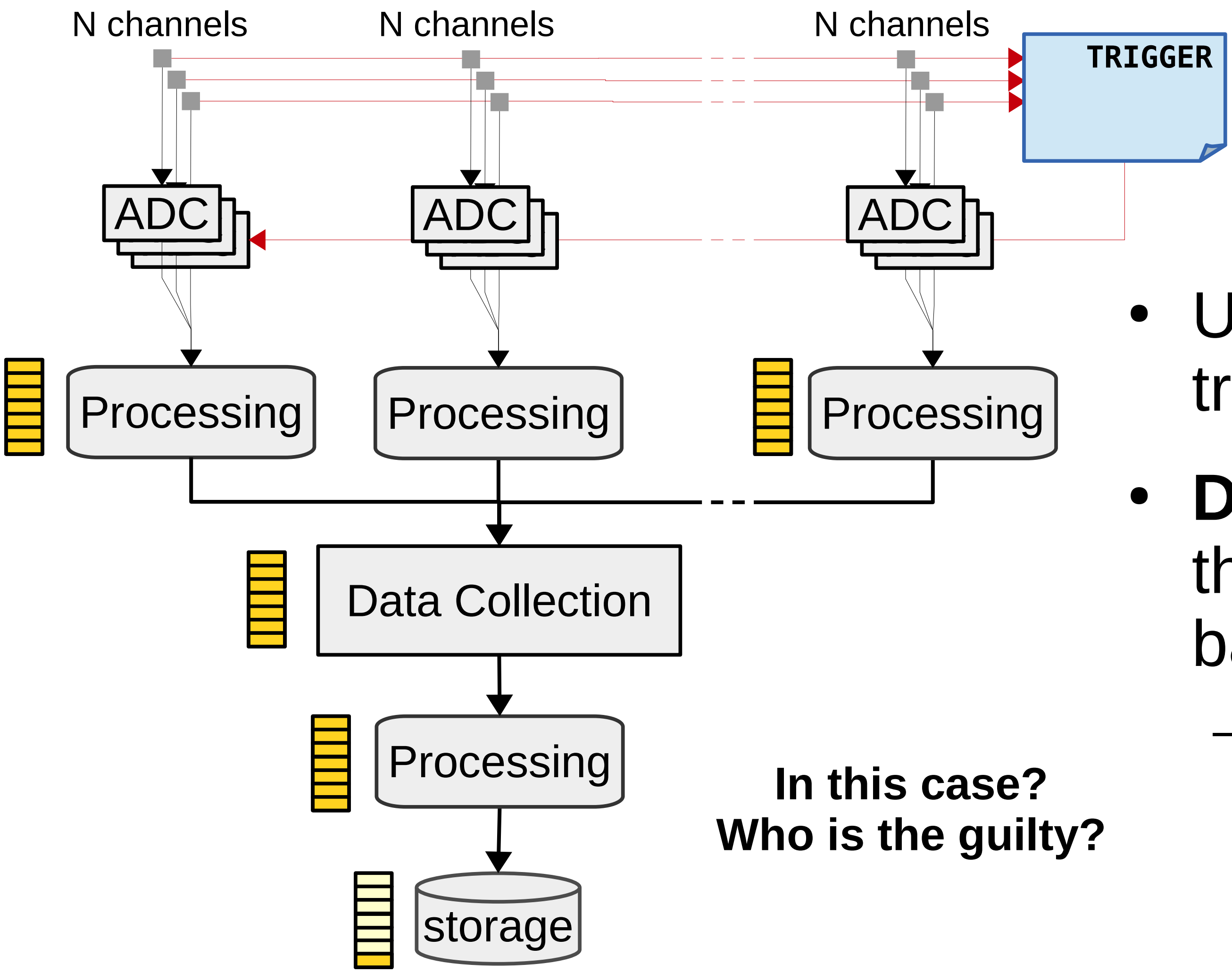
- If a system/buffer gets saturated
  - the "pressure" is propagated upstream (**back-pressure**)



- Up to exert **busy** to the trigger system
- **Debugging**: where is the source of backpressure?
  - follow the buffers occupancy via the monitoring system

Backpressure

- If a system/buffer gets saturated
  - the "pressure" is propagated upstream (**back-pressure**)

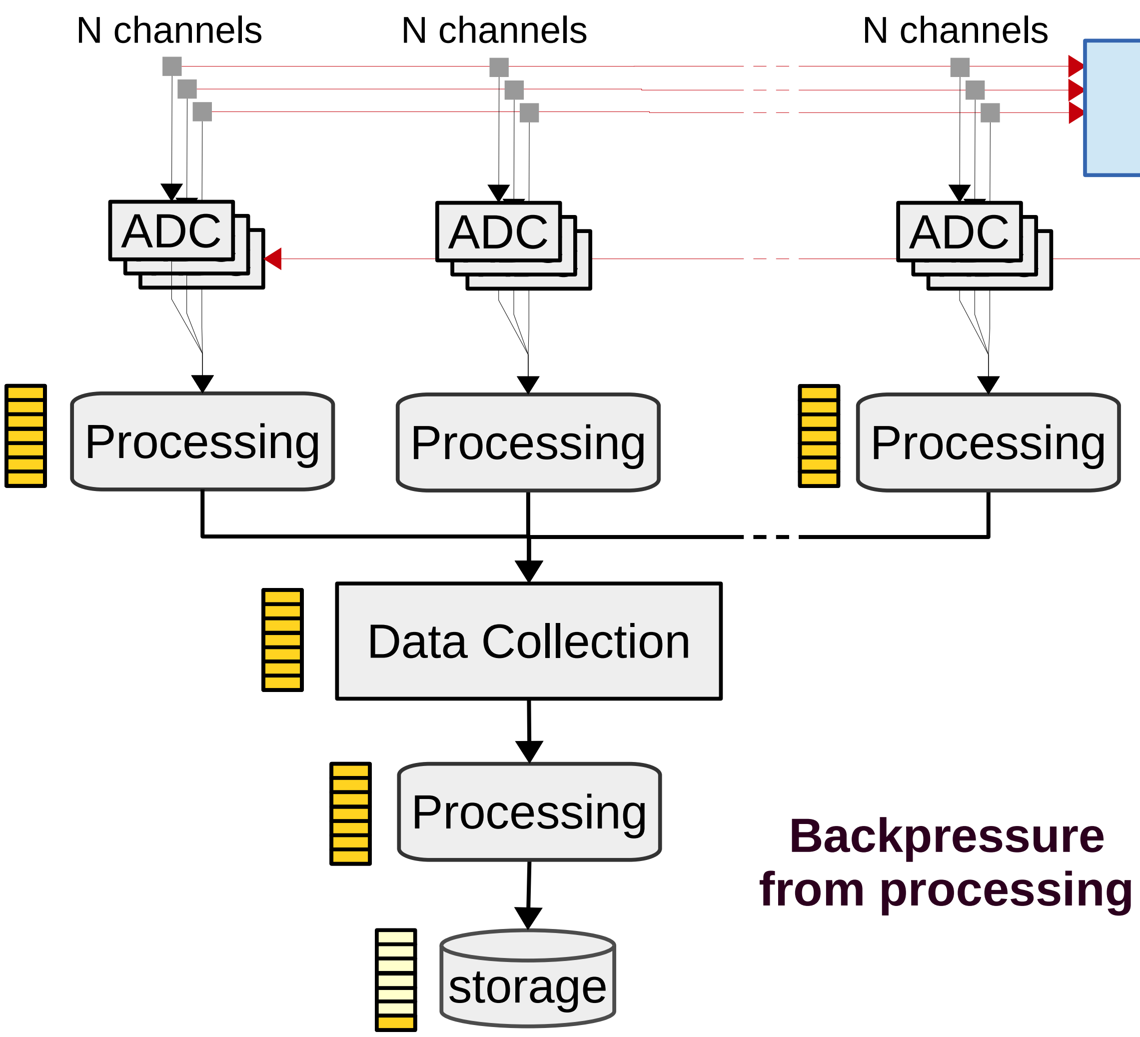


In this case?
Who is the guilty?

- Up to exert **busy** to the trigger system
- **Debugging**: where is the source of backpressure?
  - follow the buffers occupancy via the monitoring system

Backpressure

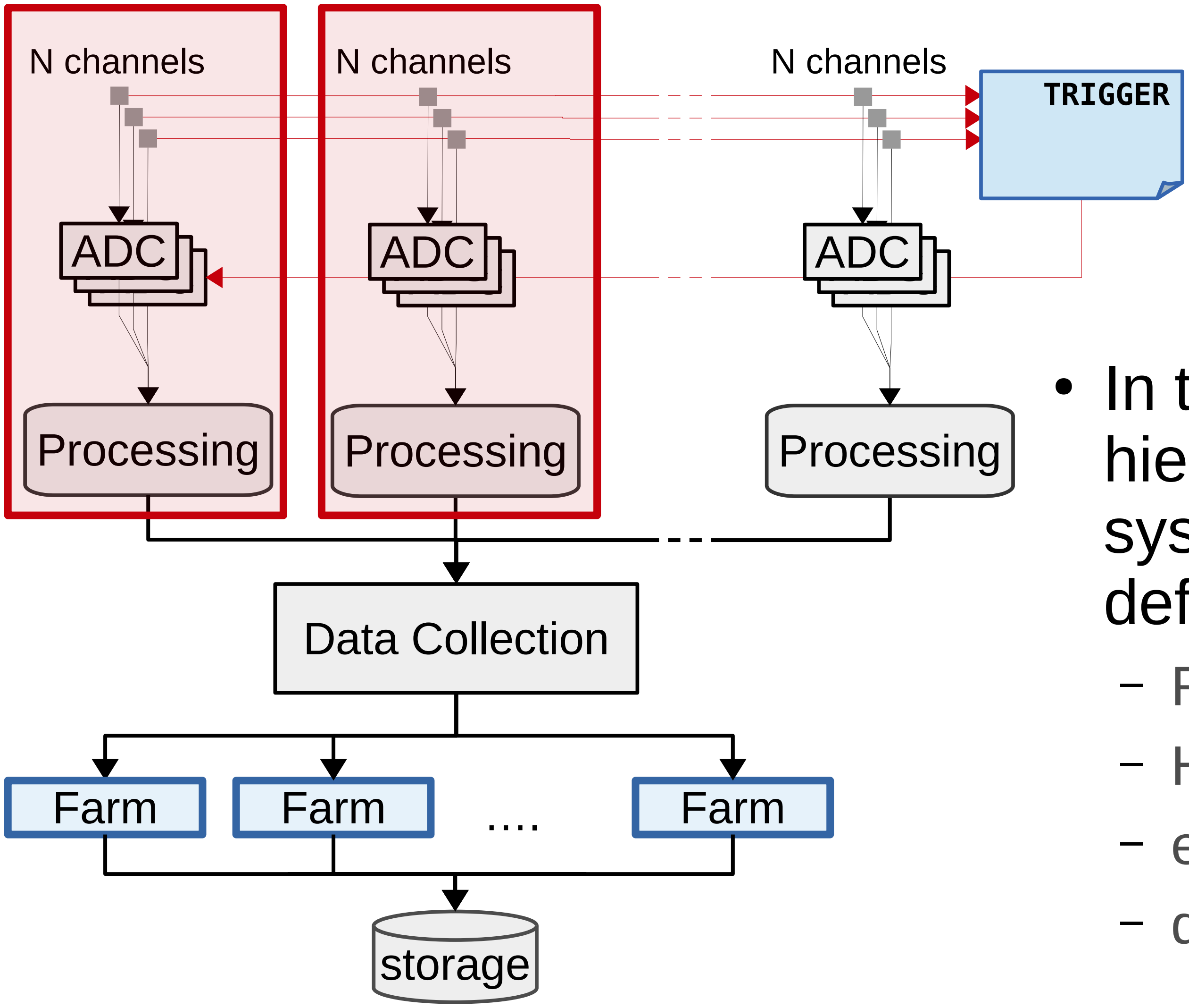
- If a system/buffer gets saturated
  - the "pressure" is propagated upstream (**back-pressure**)



- Up to exert **busy** to the trigger system
- **Debugging**: where is the source of backpressure?
  - follow the buffers occupancy via the monitoring system

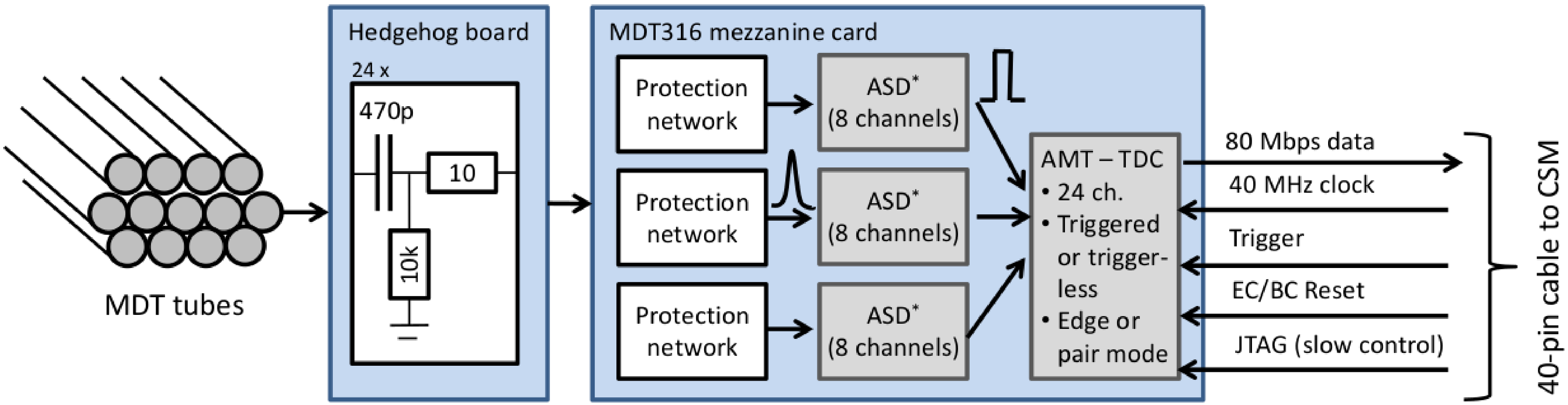
Building blocks

- Reading out data or building events out of many channels requires many components

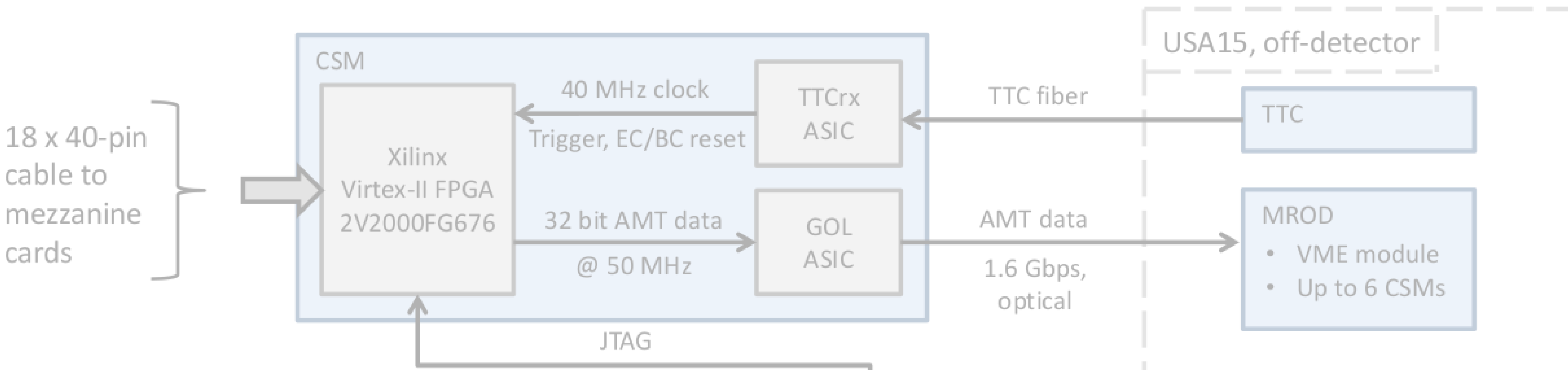


- In the design of our hierarchical data-collection system, we have better define "**building blocks**"
  - Readout crates
  - HLT racks
  - event building groups
  - daq slices

Front End electronics

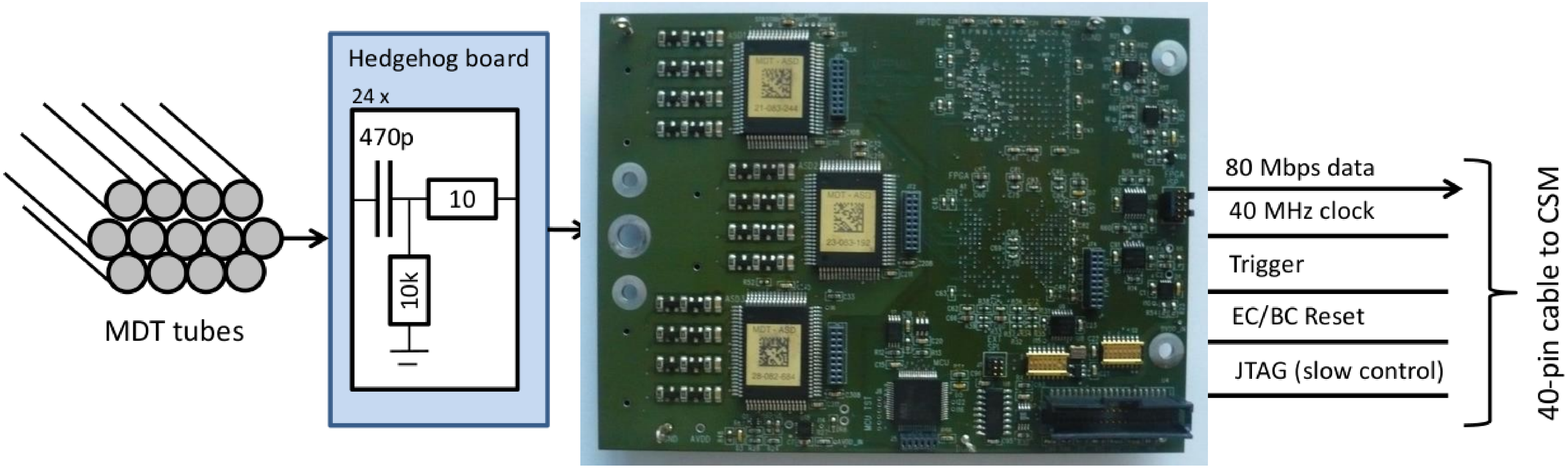


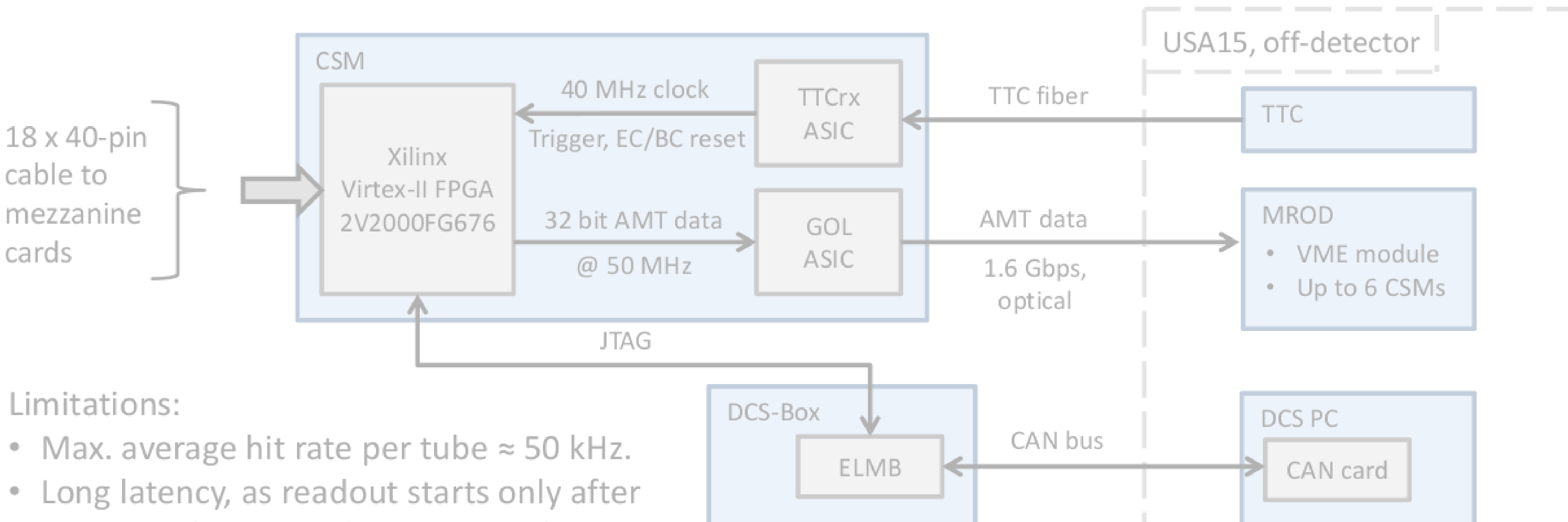
* Amplifier-Shaper-Discriminator: Analog input, digital output.



Limitations:
- Max. average hit rate per tube ≈ 50 kHz.
- Long latency, as readout starts only after L1 accept (≈ 8.5 μs after L1 accept).

Front End electronics

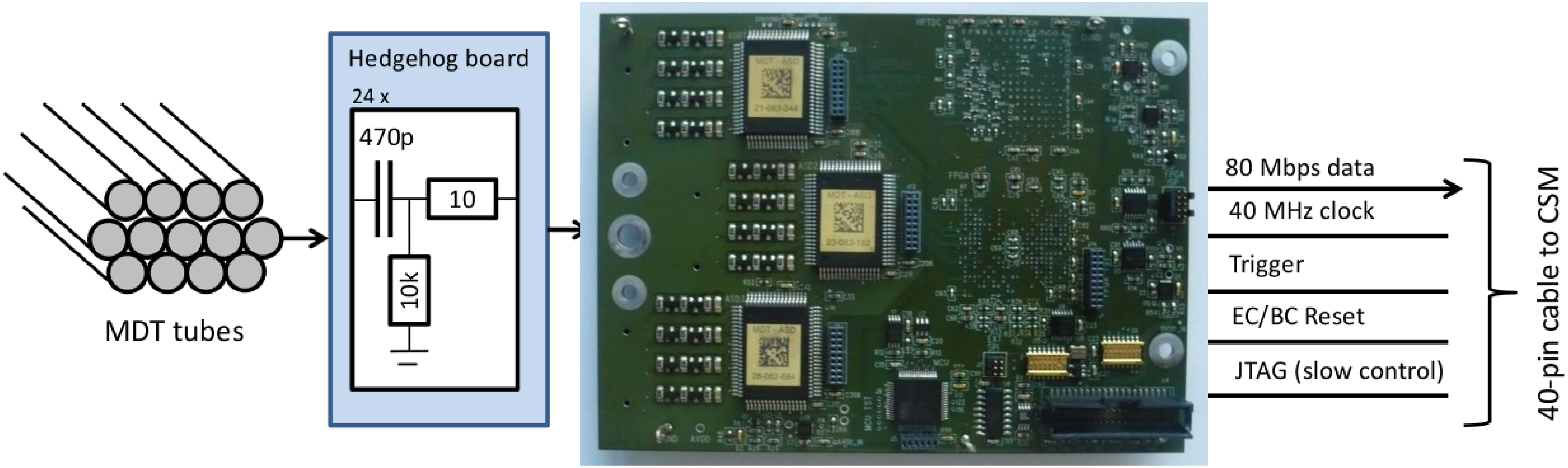


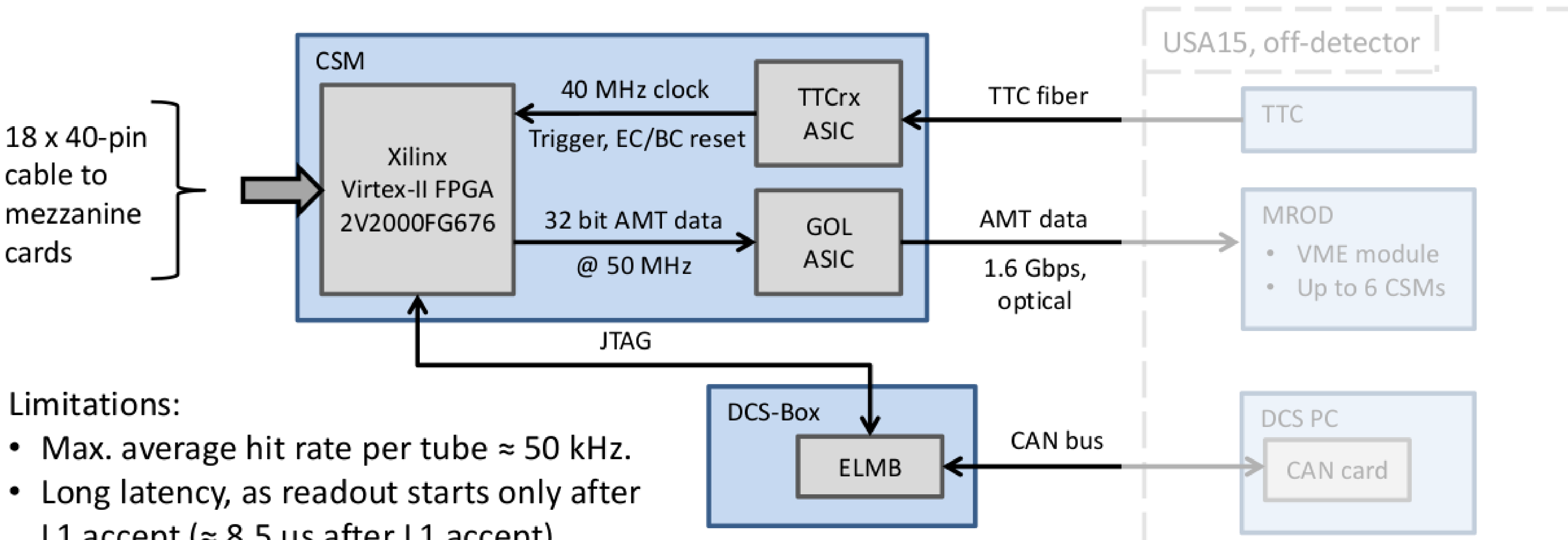


Limitations:

- Max. average hit rate per tube ≈ 50 kHz.
- Long latency, as readout starts only after L1 accept (≈ 8.5 µs after L1 accept).

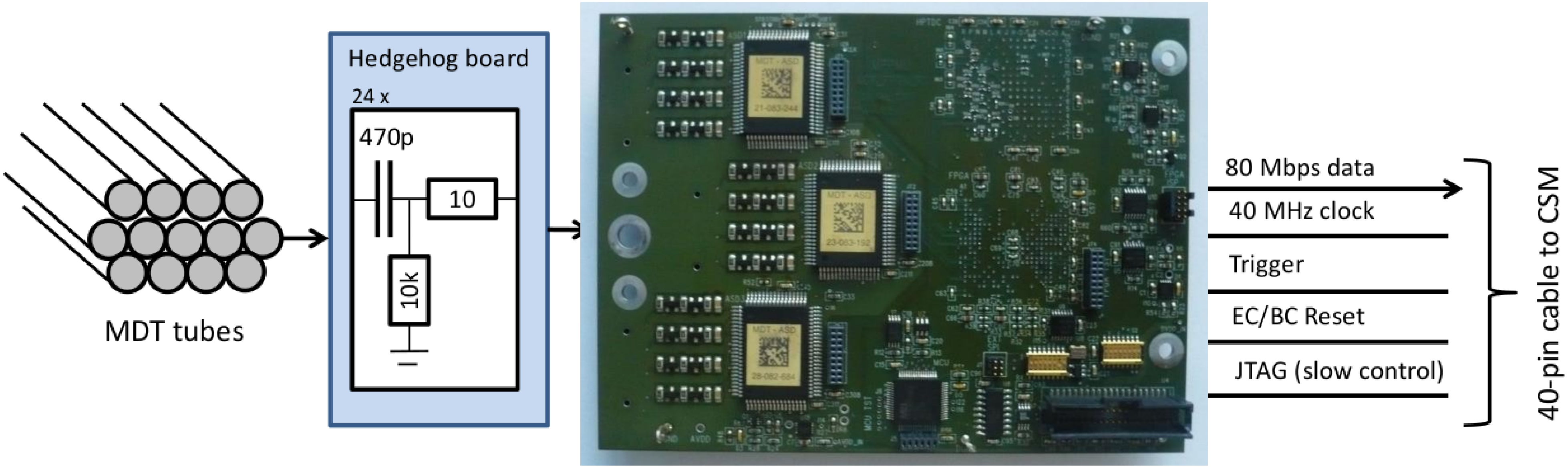
Front End electronics

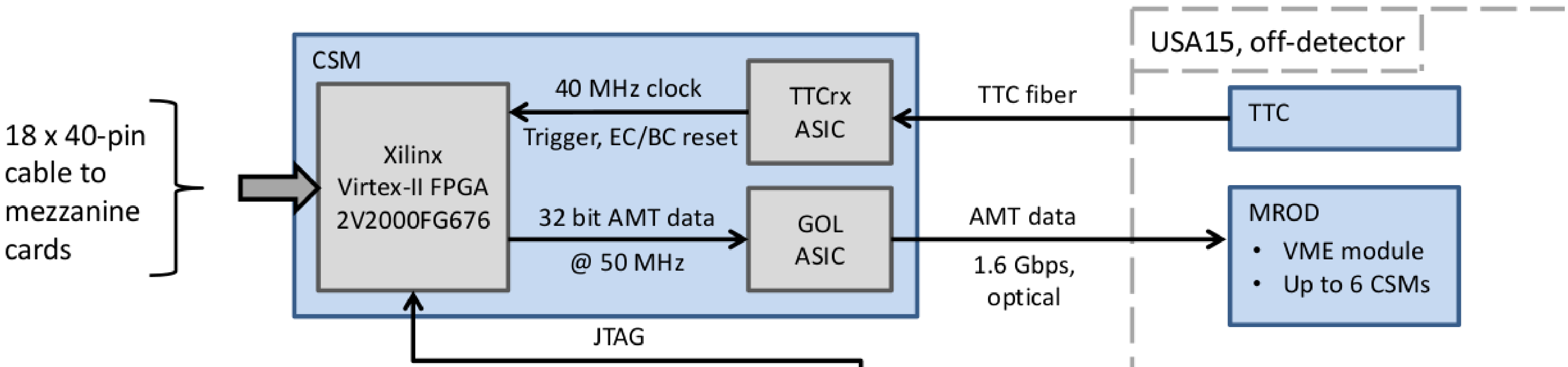




Limitations:
- Max. average hit rate per tube ≈ 50 kHz.
- Long latency, as readout starts only after L1 accept (≈ 8.5 μs after L1 accept).

Front End electronics

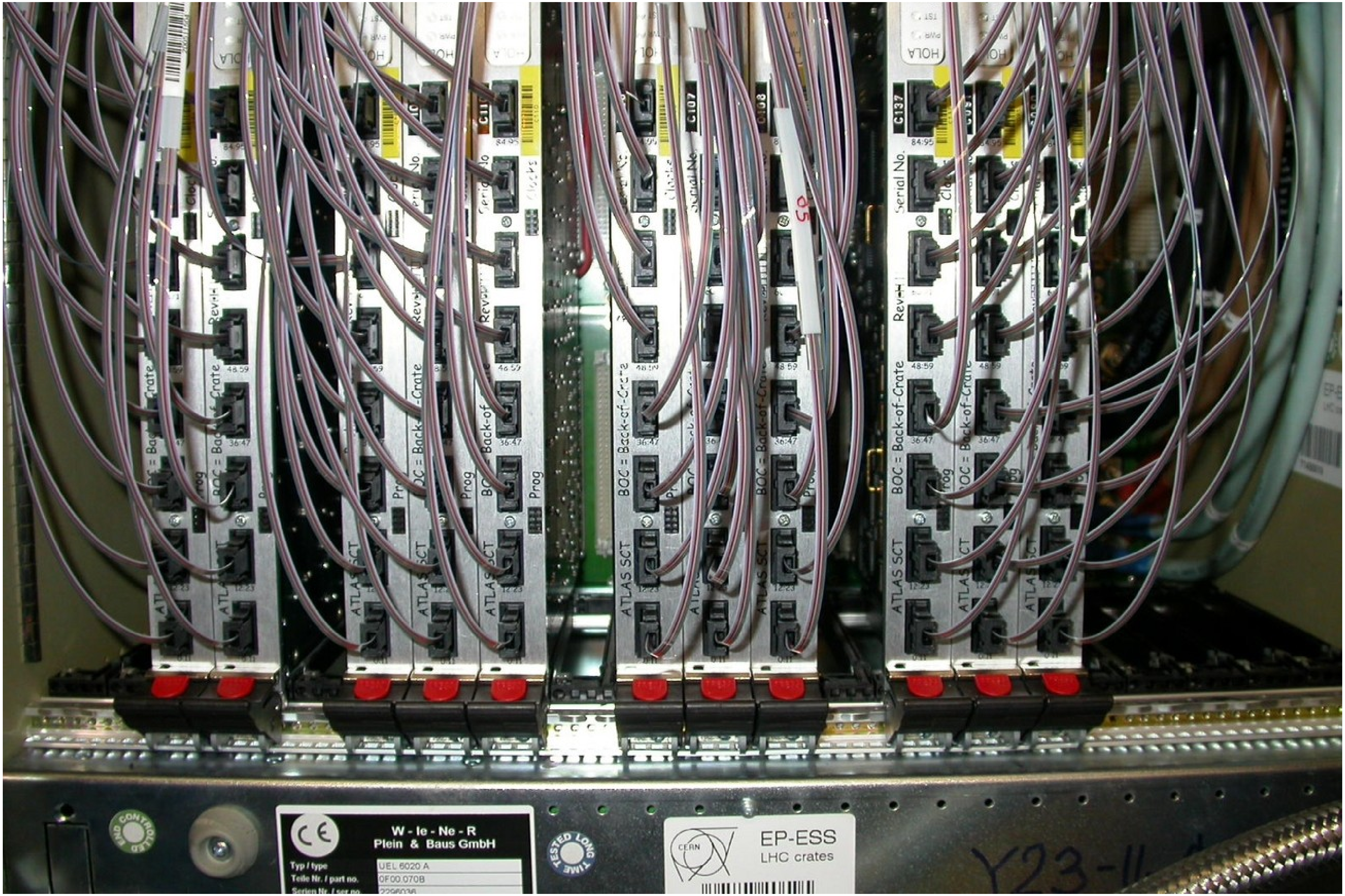




Limitations:

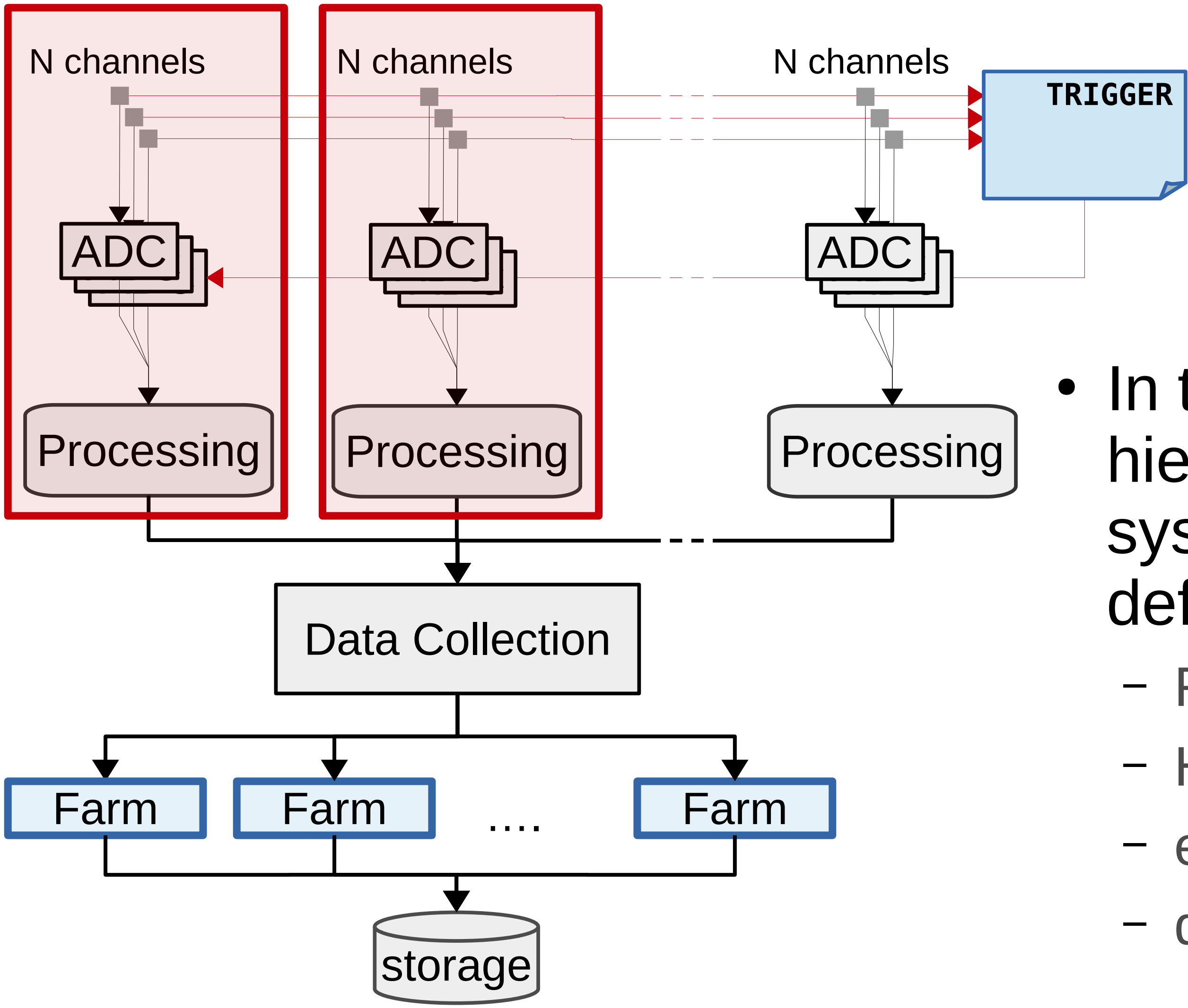
- Max. average hit rate per tube ≈ 50 kHz.
- Long latency, as readout starts only after L1 accept (≈ 8.5 μs after L1 accept).

Readout Boards (Counting Room)



Building blocks

- Reading out data or building events out of many channels requires many components



- In the design of our hierarchical data-collection system, we have better define "**building blocks**"
  - Readout crates
  - HLT racks
  - event building groups
  - daq slices

Farm (@surface)

sw@isotdaq2024

- *LabVIEW*
  - **Adriaan** Rijllart

- *Programming for today's physicist and engineers*
  - **Dinyar** Rabady

- *DAQ software*
  - **Enrico** Pasqualucci

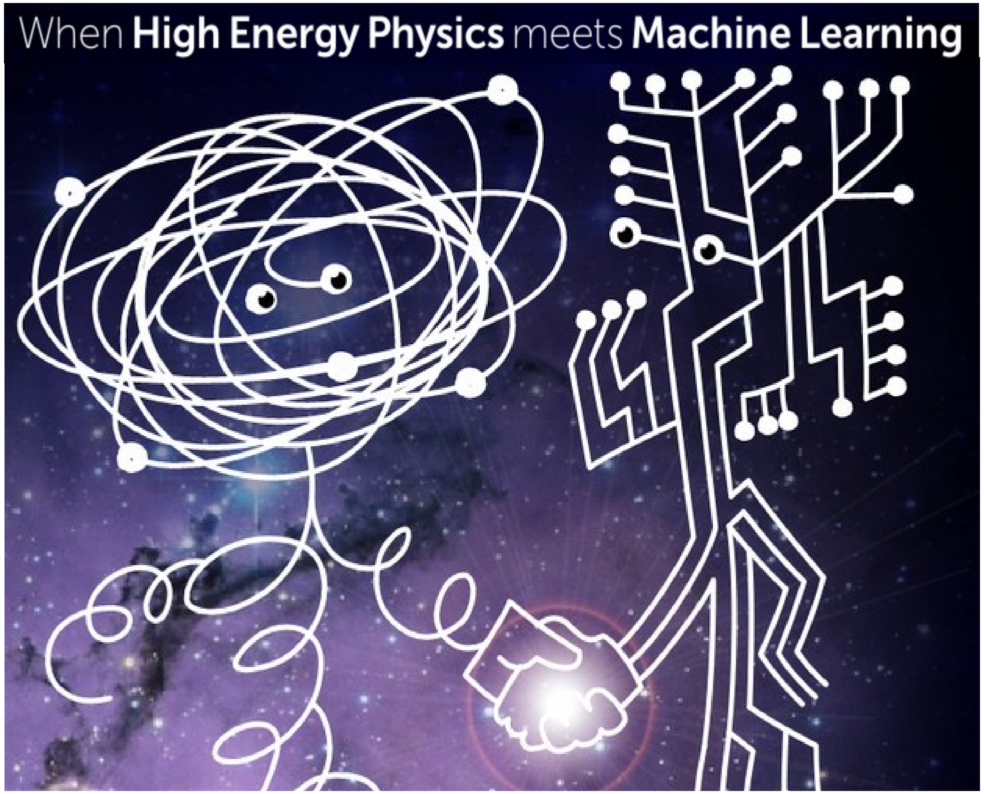
GPU

- Used since a while by Alice, NA62, etc
  - increase processing power for parallelizable tasks
- Part of LHC upgrades
  - LHC-b, CMS, ATLAS
- *GPU in HEP: online high quality trigger processing*
  - **Gianluca** Lamanna
- *Introduction to GPU programming*
  - Lab 14


Hey Chat GPT,
finish this building...
Hey Chat GPT,
finish this building...
Your skills are irreplaceable.
IMPACT
Satisfying jobs in construction and

ML@isotdaq2024

- AI & ML everywhere in HEP
  - NB: not mentioned in SnowMass 2013
- Spreading to Trigger and DAQ
  - Have a look to CHEP23 talks
- *Why you should consider machine learning*
  - **Satchit** Chatterji
- *Machine Learning Applications*
  - **Thomas** Owen James

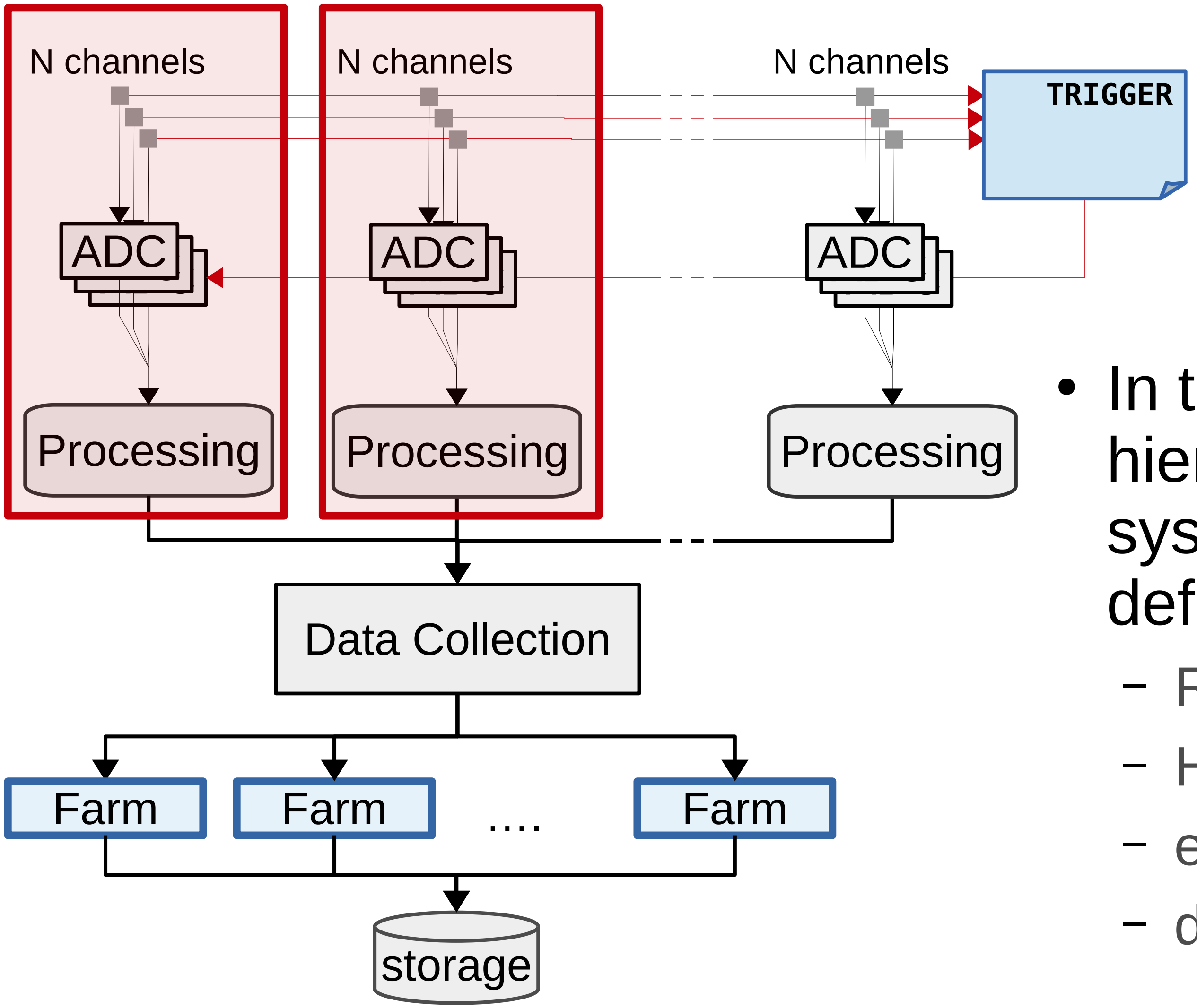


AI/ML@CHEP23

- 48 Talks in 8 sessions; two examples
  - *"Radically different future for HEP enabled by AI/ML"* Kyle Cranmer
    - Few points about TDAQ
  - *"AI and ML for EPIC: an Overview", Cristiano Fanelli*
    - First large scale experiment designed using AI/ML

Building blocks

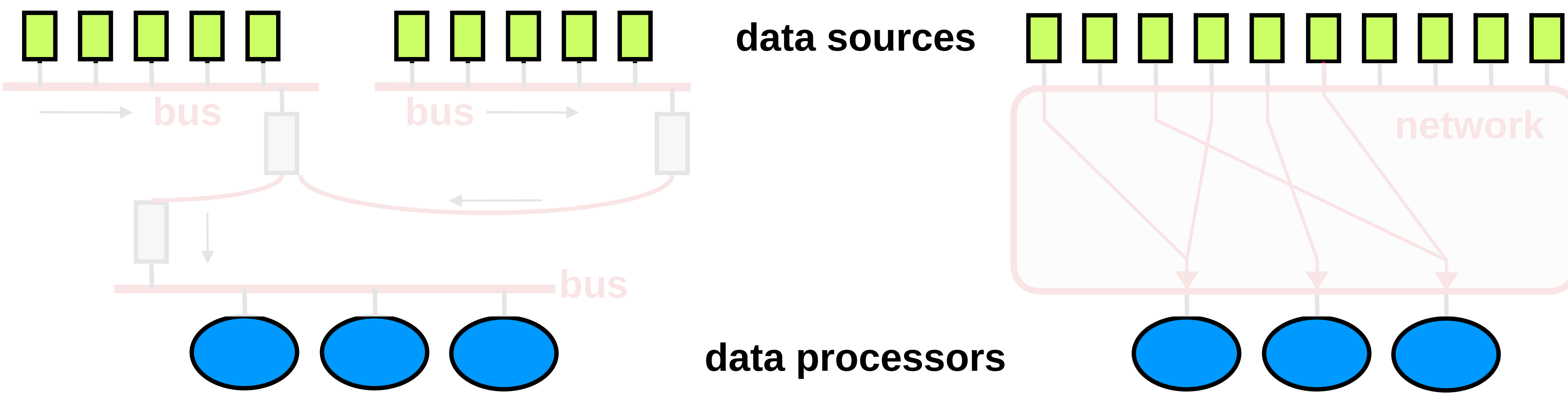
- Reading out data or building events out of many channels requires many components



- In the design of our hierarchical data-collection system, we have better define "**building blocks**"
  - Readout crates
  - HLT racks
  - event building groups
  - daq slices

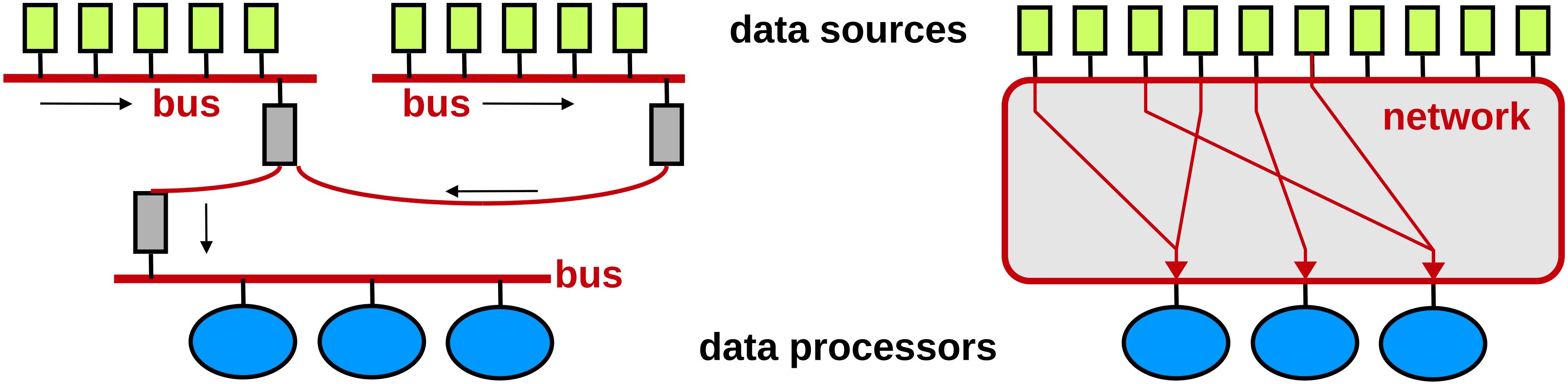
Readout Topology

- How to organize the interconnections inside the building blocks and between building blocks?
  - How to connect data sources and data destinations?
  - Two main classes: **bus** or **network**



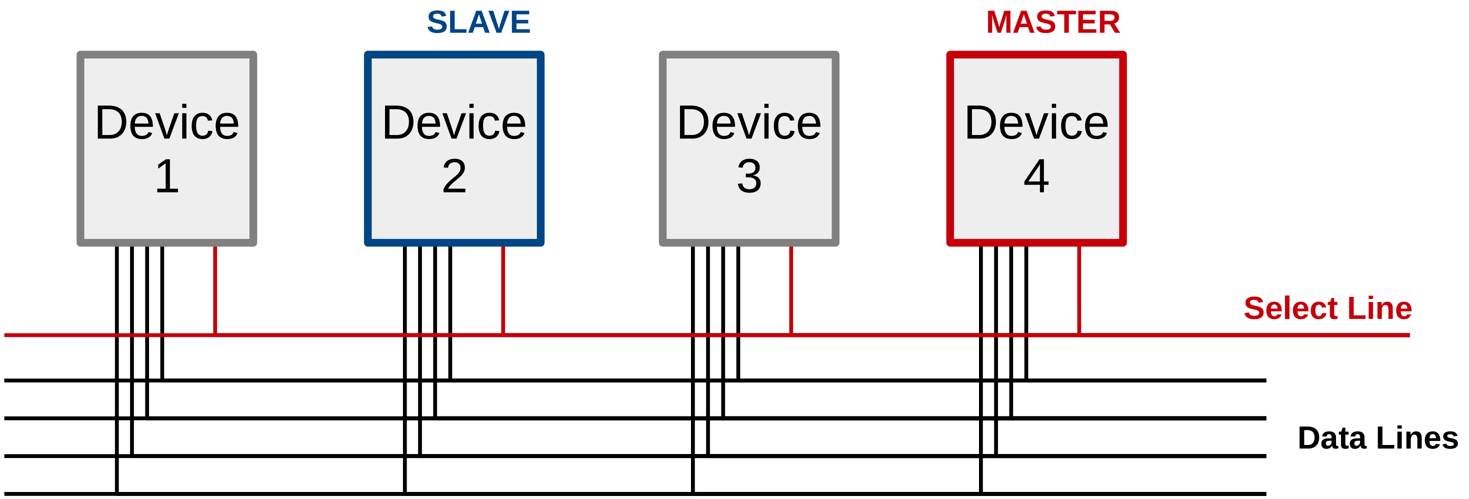
Readout Topology

- How to organize the interconnections inside the building blocks and between building blocks?
  - How to connect data sources and data destinations?
  - Two main classes: **bus** or **network**

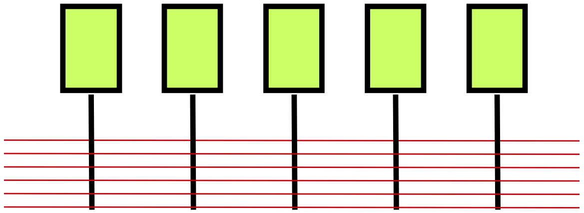


Buses

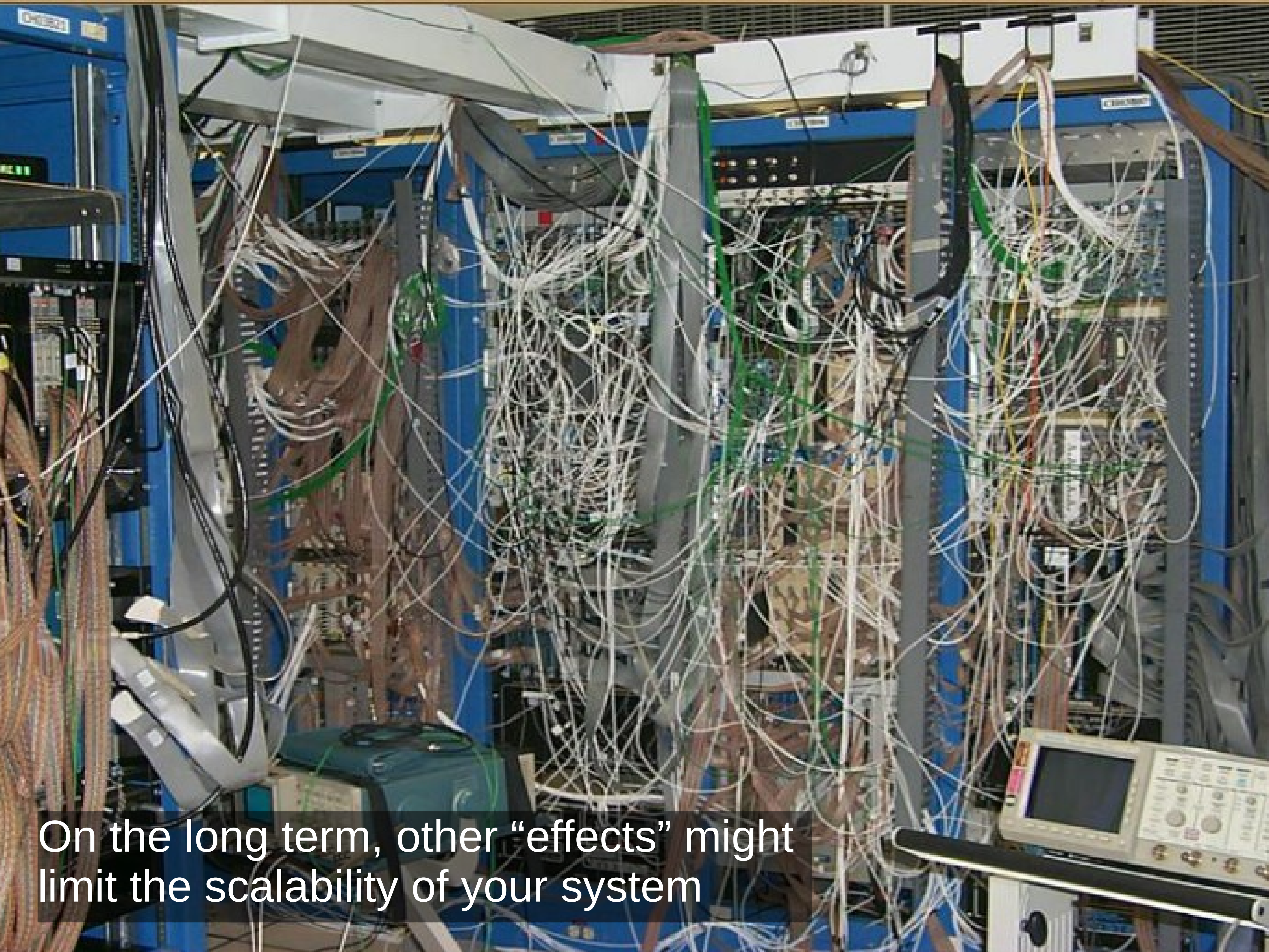
- Devices connected via a **shared bus**
  - Bus → group of electrical lines
- Sharing implies **arbitration**
  - Devices can be **master** or **slave**
  - Devices can be addresses (uniquely identified) on the bus
- E.g.: SCSI, Parallel ATA, VME, PCI …
  - local, external, crate, long distance, ...



Bus facts

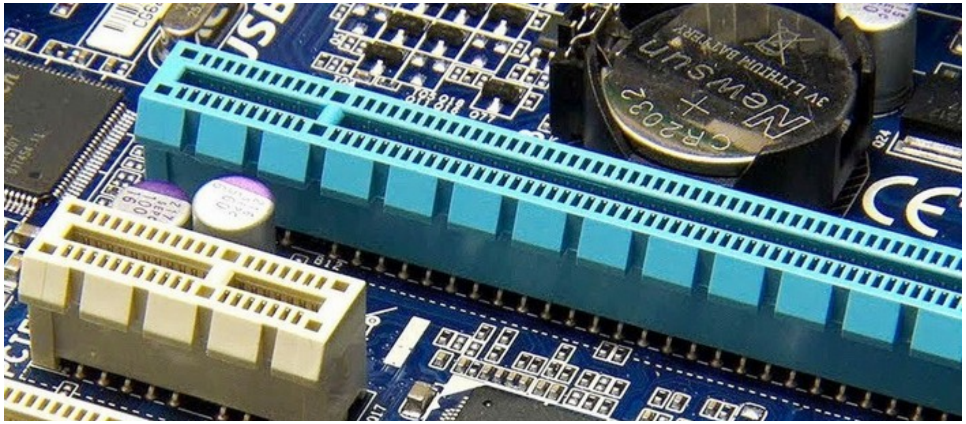


- Simple :-)
  - Fixed number of lines (bus-width)
  - Devices have to follow well defined interfaces
    - Mechanical, electrical, communication, ...
- **Scalability** issues :-(
  - Bus bandwidth is shared among all the devices
  - Maximum bus width is limited
  - Maximum number of devices depends on bus length
  - Maximum bus frequency is inversely proportional to the bus length
  - On the long term, other “effects” might limit the scalability of your system


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

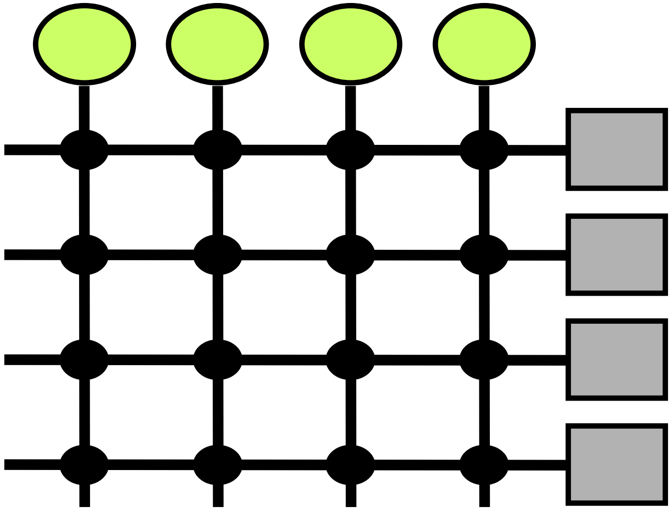
bus@isotdaq2024

- *Introduction to VME*
  - ***Markus** Joos*
- *Modular electronics*
  - **Markus** Joos
- *VME bus programming* [Lab 1]
- µATCA [Lab 6]
- *PCI express*
  - **Paolo** Durante

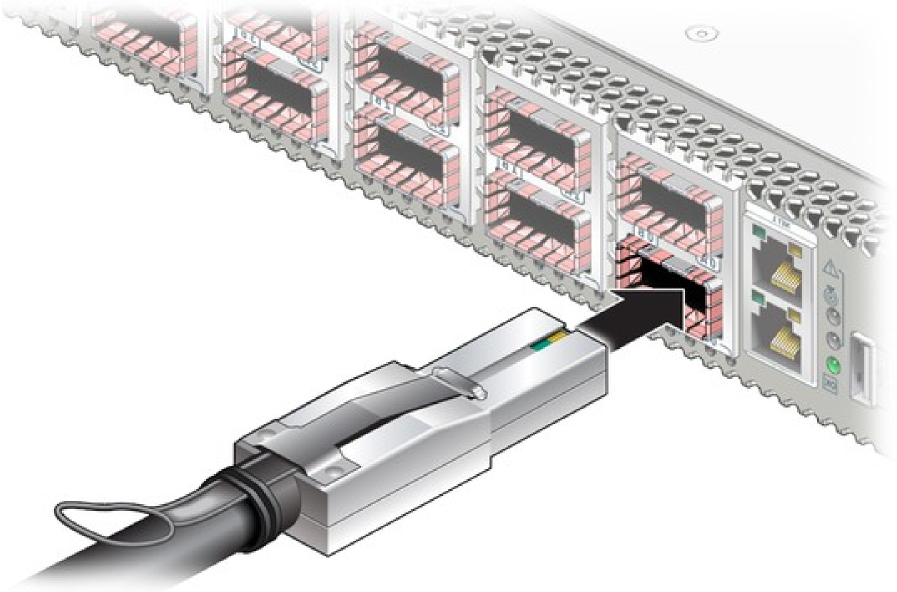


Network

- All devices are **equal**
  - They <u>communicate directly</u> with each other via messages
  - No arbitration, simultaneous communications

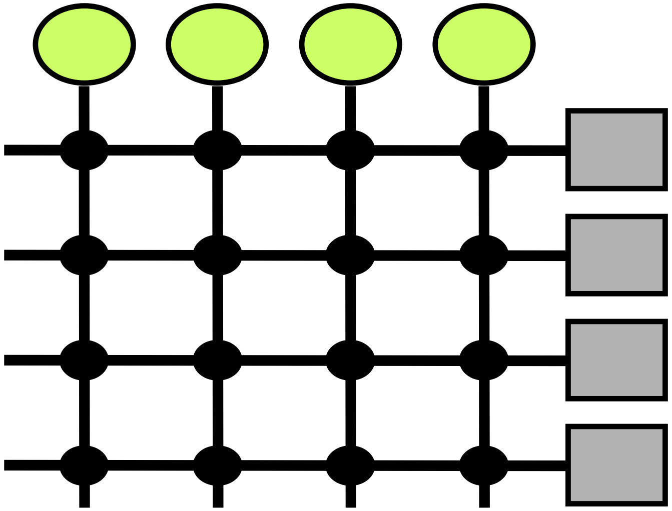


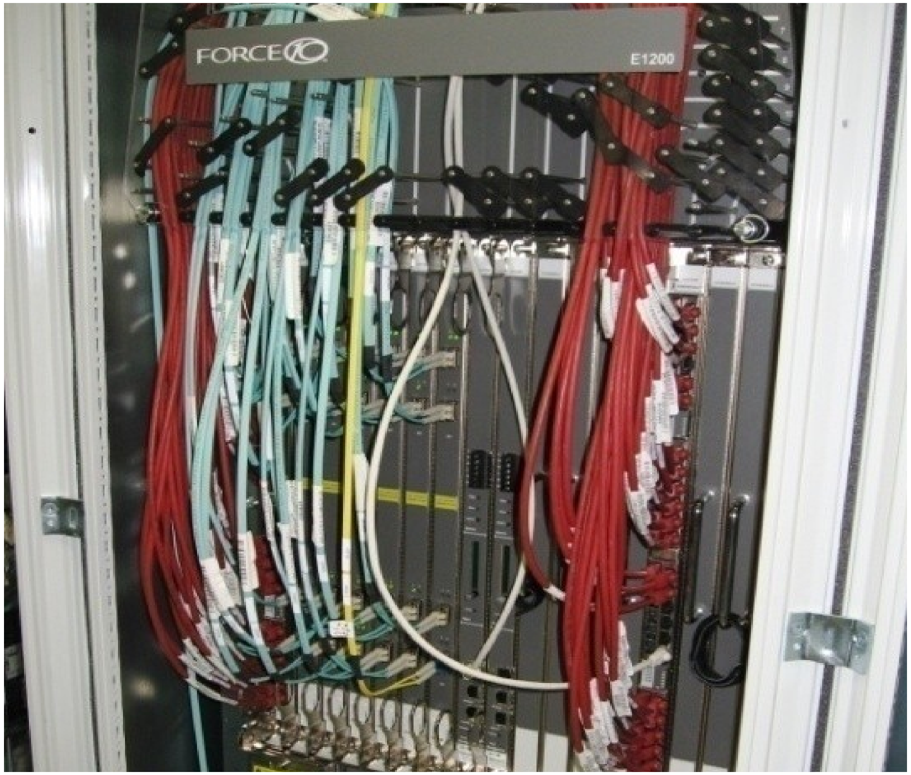
- Eg: Telephone, Ethernet, Infiniband, …



Network

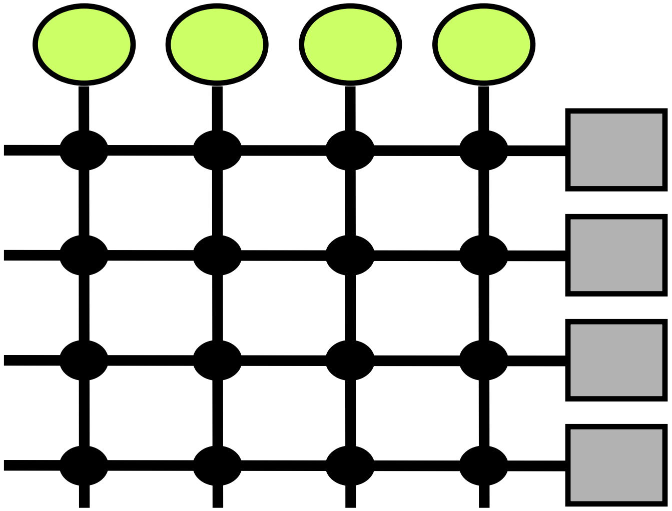
- In switched networks, **switches** move messages between sources and destinations
  - Find the right path
- How **congestions** (two messages with the same destination at the same time) are handled?
  - The key is ...

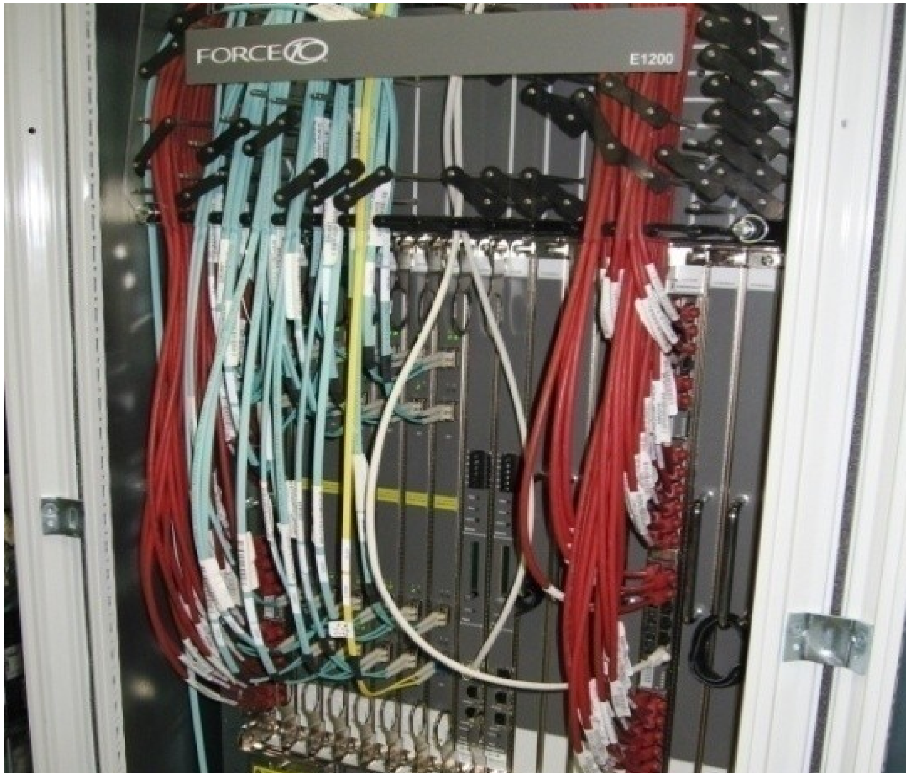




Network

- In switched networks, **switches** move messages between sources and destinations
  - Find the right path
- How **congestions** (two messages with the same destination at the same time) are handled?
  - The key is .... **buffering**

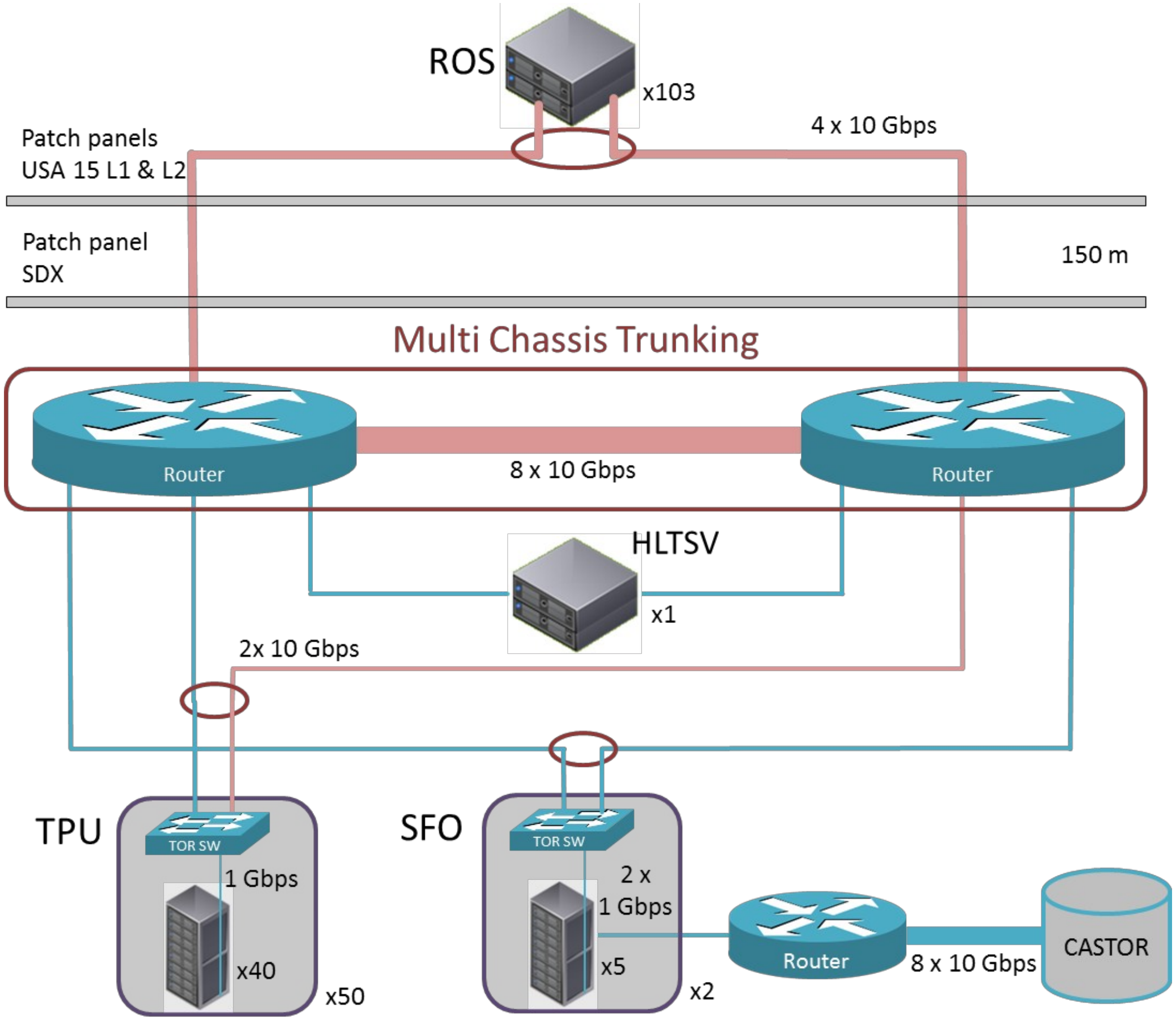




Network

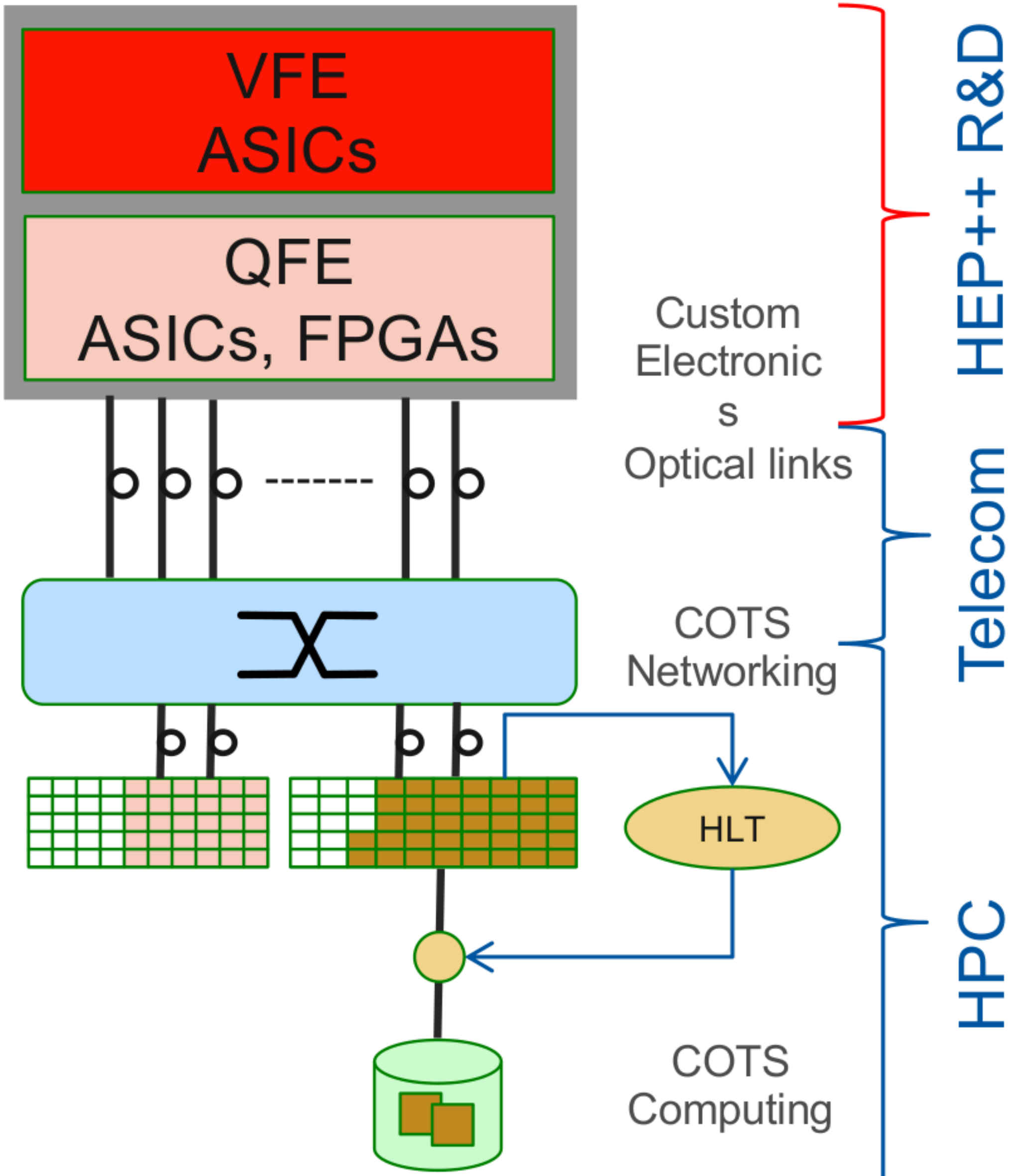
- Networks scale well (and allow redundancy)
  - They are the backbones of LHC DAQ systems

- *Networking for data acquisition systems*
  - **Petr** Zejdl
- *Networking for data acquisition systems*
  - Lab 9



COTS

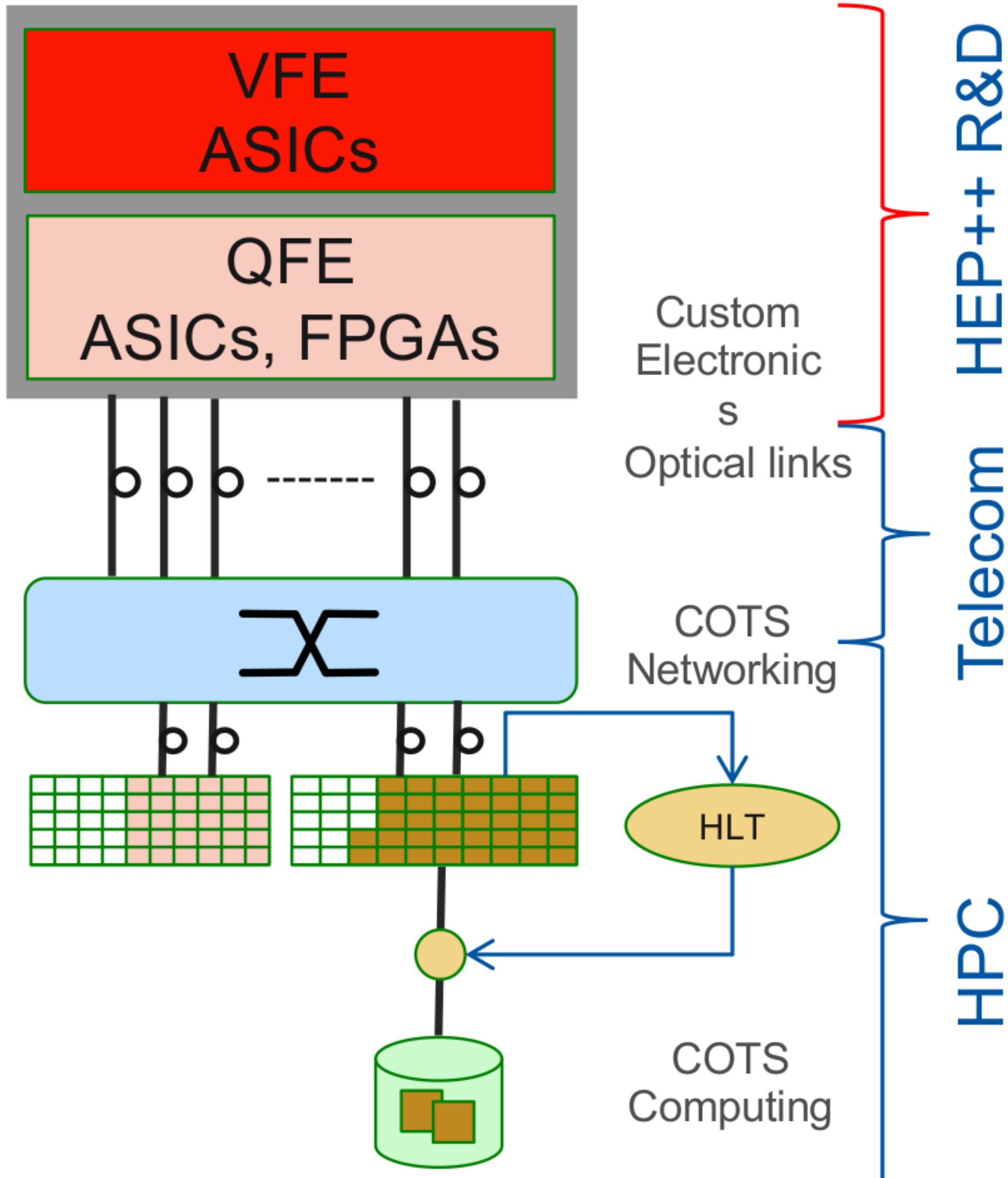
- Very Front End
  - does the analog part
  - ADC, low-level calibration, zero suppression, lossless compression
  - low-power, rad-tolerant
- Quasi Front End
  - medium scale aggregation, local reconstruction, ”lossy” compression, transition to standard protocol on optical links



E. Meschi, Summer Student Lectures 2022

COTS

- **C**ommodity **O**f **T**he **S**helf
- COTS switched networks
  - provide further aggregation, up to and including event building
- COTS servers
  - with co-processors (GPU, FPGA)
  - do the final selection



E. Meschi, Summer Student Lectures 2022

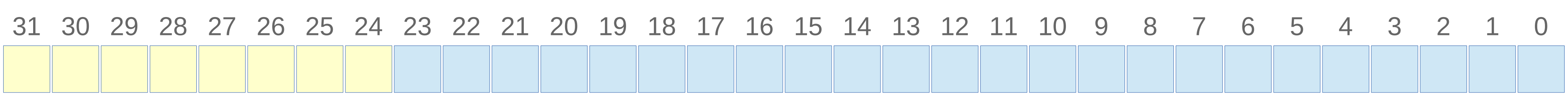
Outline

- Introduction
  - What is DAQ?
  - Overall framework
- Basic DAQ concepts
  - Digitization, Latency
  - Deadtime, Busy, Backpressure
  - De-randomization
- Scaling up
  - Readout and Event Building
  - Buses vs Network
- Data encoding

00000004 00000001 0000c89c aa1234aa 00003227 0000001c 04000000 00793c29 00000001 00000000
00000000 50753e27 0ab16f70 00097a2b 00000000 00033dac 00000063 920117d5 00000aa8 00000081
00000018 00020000 40000000 00000000 00000000 00000000 00000000 00000000 00000000 00020000
00000000 dd1234dd 0000002d 00000009 04000000 00210000 00000002 00000000 92011d7f 00000001
ee1234ee 00000009 03010000 00210000 00033dac 920117d5 00000aa8 00000081 00000000 2003e766
2013e282 201490d2 9c122017 ef322018 9d562023 dfa22039 c2224000 2040aa82 2041c3a2 204282b3
20489082 2057efb2 205a8616 2063cce2 2066aee2 2068a0c2 20768ff7 99522077 de72207b d8224000
00000000 00000000 00000002 00000015 00000001 d04326b2 dd1234dd 0000002d 00000009 04000000
00210001 00000002 00000000 92011d80 00000001 ee1234ee 00000009 03010000 00210001 00033dac
920117d5 00000aa8 00000081 00000000 2004af72 2010a3f2 20128ec2 2017c212 202083c2 9ec22025
c6c22026 a3022034 afb74000 20488602 2053c7c2 20548512 95829672 2063c2e2 e512ee02 20648fb2
2074a5e2 2075d5b2 207aa892 ad32207b ed72ee32 00000000 00000000 00000002 00000015 00000001
3de510d4 dd1234dd 00000031 00000009 04000000 00210002 00000002 00000000 92011d80 00000001
ee1234ee 00000009 03010000 00210002 00033dac 920117d5 00000aa8 00000081 00000000 20109ef2
2011ee42 efc22012 93222013 e2822014 97022017 e182201b e0222025 eaa22027 cab22028 80d3202a
84b22035 c5c2ccb2 2036ebc2 20389672 20508002 95a22051 d3172056 9ee22057 ef42205b cee2eca2
2060ad62 2061c4a2 2063ddb7 20649542 00000000 00000000 00000002 00000019 00000001 f631054a
dd1234dd 00000029 00000009 04000000 00210003 00000002 00000000 92011d80 00000001 ee1234ee
00000009 03010000 00210003 00033dac 920117d5 00000aa8 00000081 00000000 2027d422 203088a2
2031d692 20369542 2037ed92 20409c92 ace22044 9a822046 a9e22047 d3422048 8fb2204a 8a12204b
e172205b c4872060 8f822065 ea222067 c3f24000 00000000 00000000 00000002 00000011 00000001
aeaa0e15 dd1234dd 00000039 00000009 04000000 00210004 00000002 00000000 92011d80 00000001
ee1234ee 00000009 03010000 00210004 00033dac 920117d5 00000aa8 00000081 00000000 2006af12
2017eb47 201a8e76 2025e6d2 20268fa2 a292202b dff74000 2040a152 20469122 20529182 2060aea2
2061c4c2 d722d942 2063c5e2 2064a772 206aa152 206bc322 c7c22070 89d22072 8ad22073 c0b7800f
c187c1a7 c1f7c227 c287c2c7 c2e7c3a7 c3c7800f c3f7c417 c497c4d7 c547c5b7 c5e7c637 c657c677
c6b7c727 c767c7a7 00000000 00000000 00000002 00000021 00000001 a1feebf3 dd1234dd 0000002d
00000009 04000000 00210005 00000002 00000000 92011d80 00000001 ee1234ee 00000009 03010000

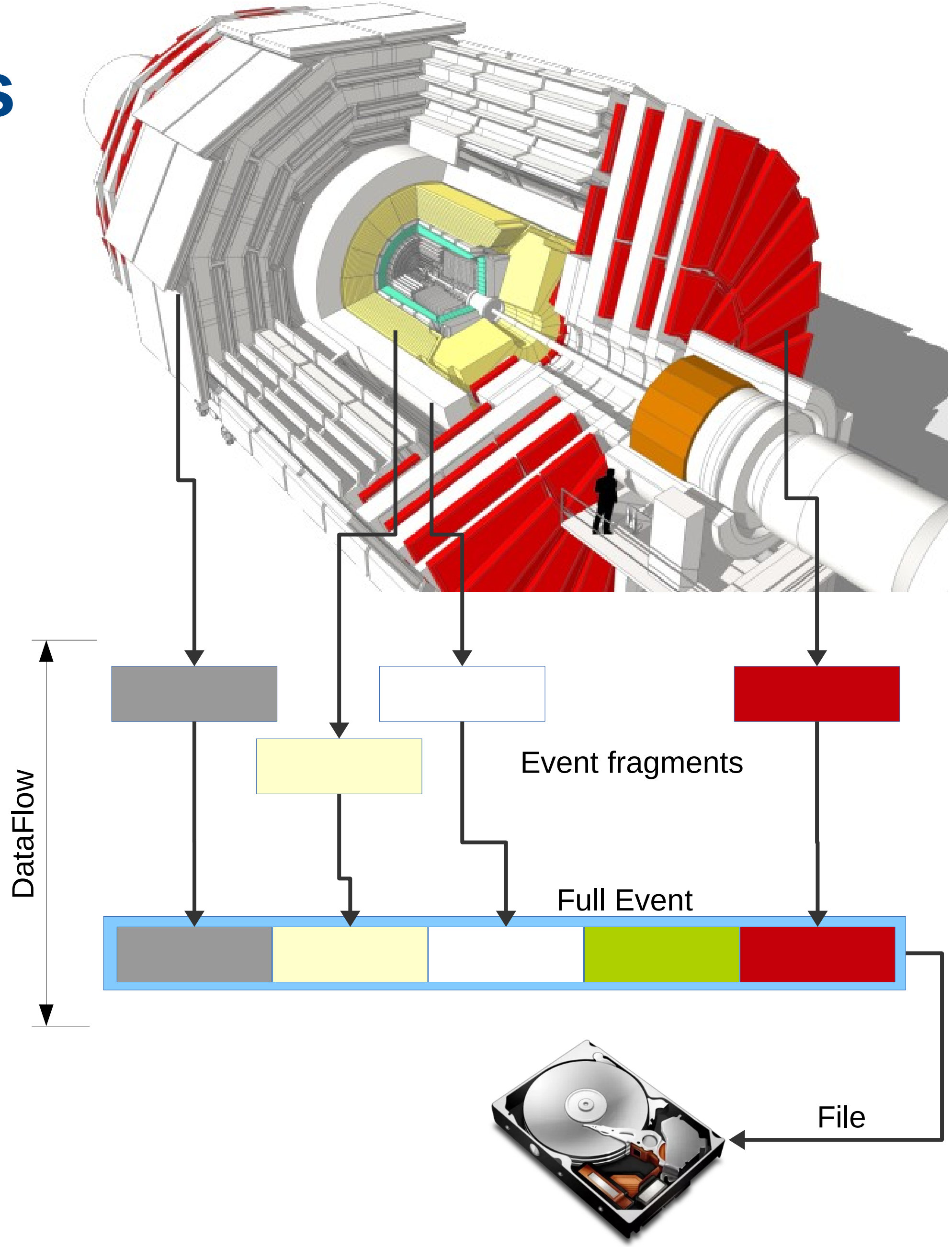
Data Encoding

- Data encoded in **digital** format
  - Arrays of words of fixed size: 2, **4**, 8 bytes
- The quantum of information must contain
  - A digital value + an unique channel identifier
- Example
  - Drift chambers: channel ID and TDC counts
  - Calorimeters: channel ID and ADC counts
- For example, one can split a word in two
  - e.g. n bits for module id, 32-n bits for TDC/ADC counts
  - Number of used bits depends on ADC/TDC range



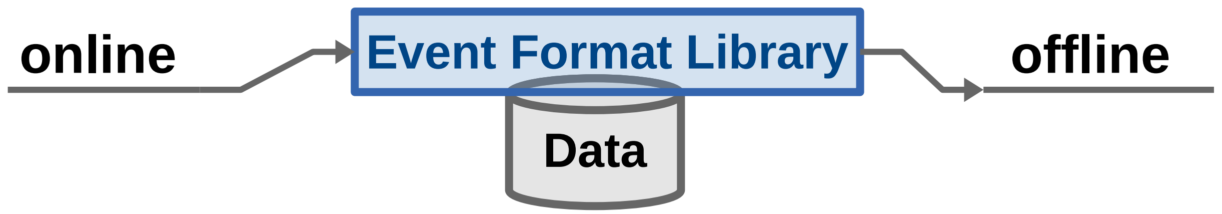
In case of multiple subdetectors

- Several data **fragments**
  - from different parts of the detector (**sources**)
- **flowing** via buses and networks from readout system to event filter to data storage
  - to be **assembled** together in the event builder
  - to be **stored** on self consistent files



Event Format

- Necessary to define an **event format**
  - How event data is encoded, stored and decoded
- It is the core of your experiment
  - The bridge between **online** and **offline** worlds
  - Online for shipping data among data-flow components and for storage
  - Offline to access and decode the data for analysis
- The library implementing the format must be unique and shared between online and offline

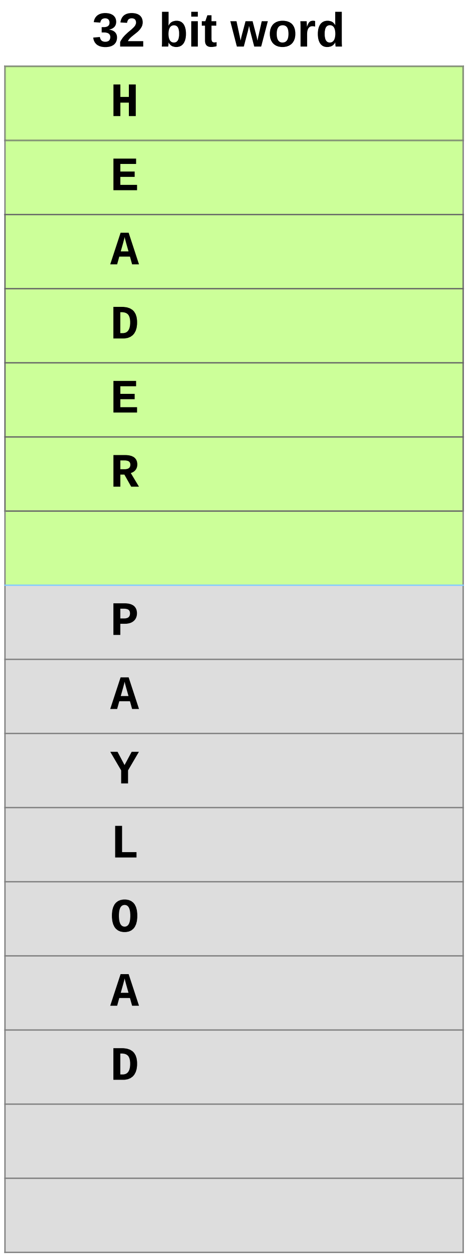


Event Format

- Identify every chunk of data, w/ a **source id**
  - Both during data taking and offline
- Associate data to the proper **bunch-crossing**
  - to collect all fragments belonging to the same event
- Keep track of the event format **version number**
  - That may evolve during experiment lifetime
- Possibility to easily extend the format
  - e.g.: adding sub-detectors
- w/ some redundancies
  - For debugging purpose

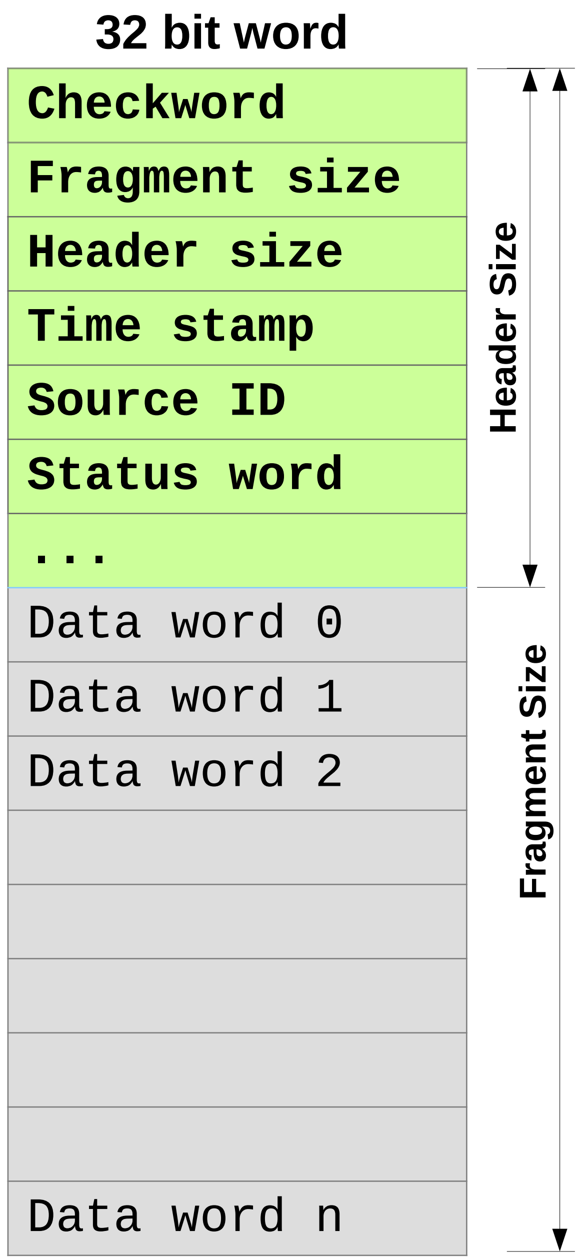
Header and payload

- Each data fragment composed by
  - A **payload**: the actual detector data
  - An **header**: that describes the payload
  - In some cases a **trailer**
- Header structure
  - **Checkword**: begin of frag. (0xEE1234EE)
  - **Fragment size**: where actual data ends
  - **Header size**: where actual data starts
  - **Time/bunchID**: timestamp
  - **Source ID**: where data is coming from
  - **Event ID:** event counter
  - **Error/status word(s)**: truncations, bad detector status, missing elements, ...



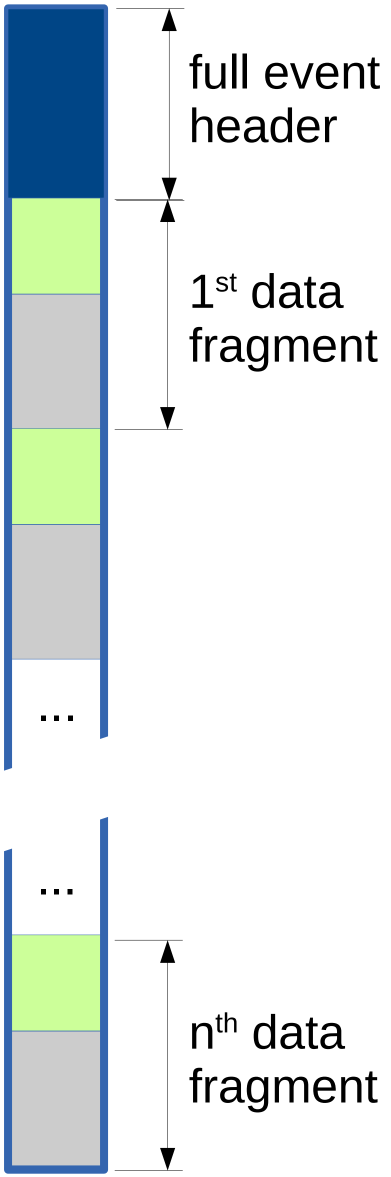
Header and payload

- Each data fragment composed by
  - A **payload**: the actual detector data
  - An **header**: that describes the payload
  - In some cases a **trailer**
- Header structure
  - **Checkword**: begin of frag. (0xEE1234EE)
  - **Fragment size**: where actual data ends
  - **Header size**: where actual data starts
  - **Time/bunchID**: timestamp
  - **Source ID**: where data is coming from
  - **Event ID:** event counter
  - **Error/status word(s)**: truncations, bad detector status, missing elements, ...



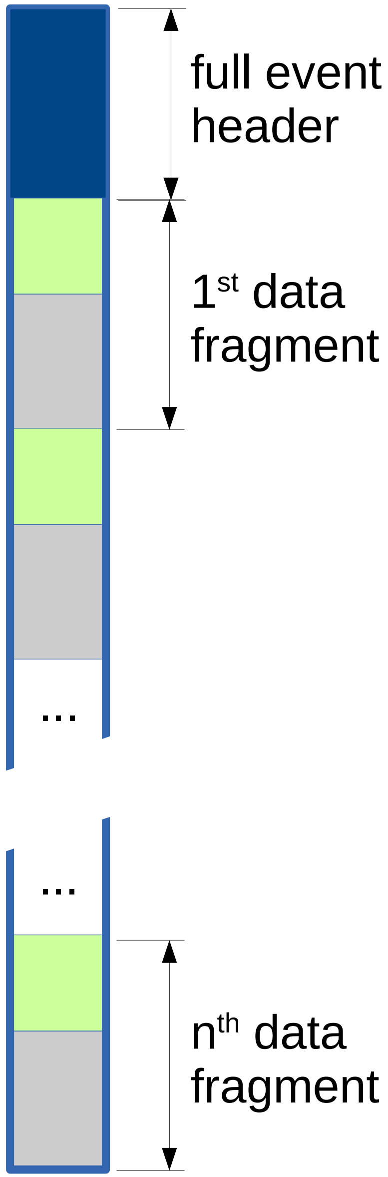
Full event

- A full **event** is a collection of **fragments**
  - There could be intermediate containers
- A full event is composed by
  - A **payload**: the “array” of data fragments
  - An **header**: that describes the event and is the portal to the collection of fragments
- Application reading a file must be able to
  - Find the 1st full event header
  - Navigate among the fragments
    - NB: fragment size word in each header
  - Up to the next event or the end of file



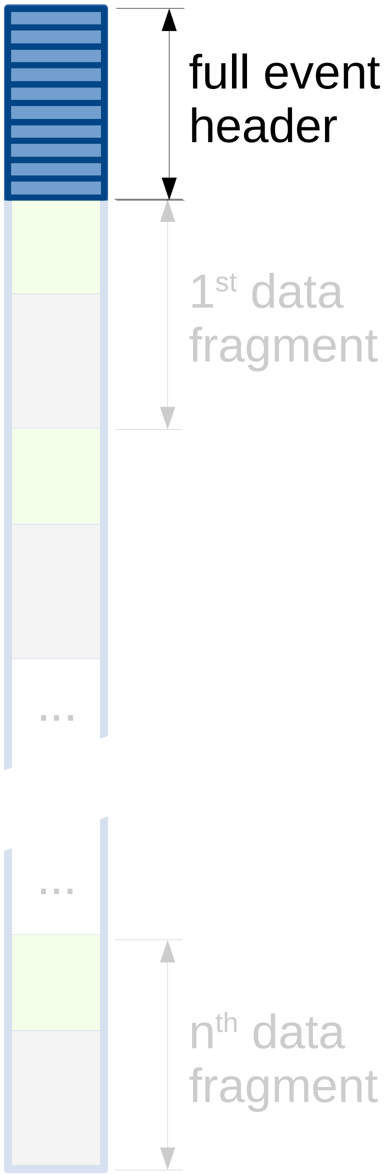
Full event

- A full **event** is a collection of **fragments**
  - There could be intermediate containers
- A full event is composed by
  - A **payload**: the “array” of data fragments
  - An **header**: that describes the event and is the portal to the collection of fragments
- Application reading a file must be able to
  - Find the 1st full event header
  - Navigate among the fragments
    - NB: fragment size word in each header
  - Up to the next event or the end of file

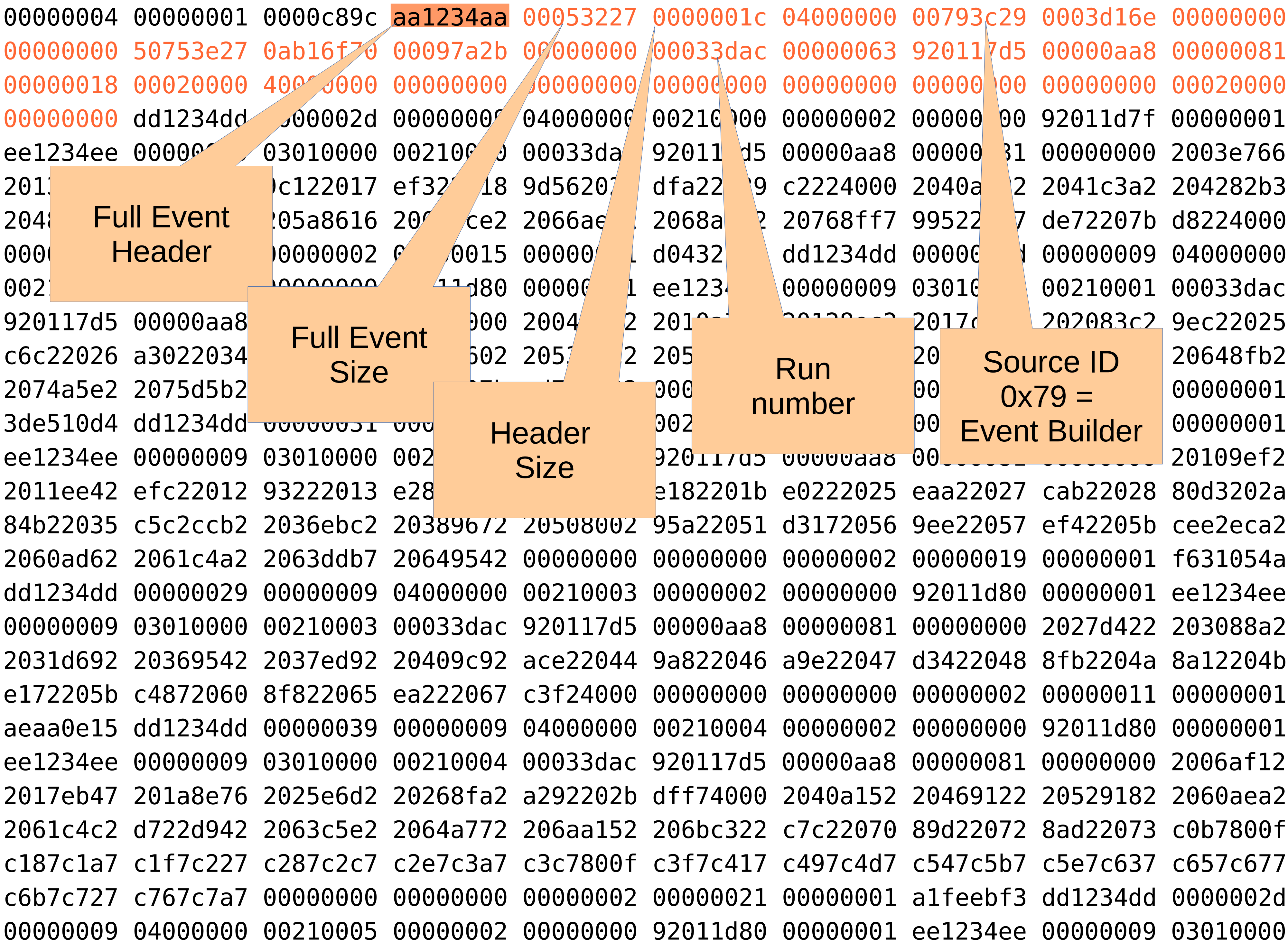


Full event header

- **Checkword**: begin of frag (e.g.: 0xAA1234AA)
- **Fragment size**: where actual data ends
- **Header size**: where actual data starts
- **Time/bunchID**: timestamp
- **Run number**
- **Event classification**
- **Error words**
- Array of offset (one for each fragment)
  - Implemented only if random access is required
  - Otherwise, just navigate from fragment to fragment



00000004 00000001 0000c89c aa1234aa 00003227 0000001c 04000000 00793c29 00000001 00000000
00000000 50753e27 0ab16f70 00097a2b 00000000 00033dac 00000063 920117d5 00000aa8 00000081
00000018 00020000 40000000 00000000 00000000 00000000 00000000 00000000 00000000 00020000
00000000 dd1234dd 0000002d 00000009 04000000 00210000 00000002 00000000 92011d7f 00000001
ee1234ee 00000009 03010000 00210000 00033dac 920117d5 00000aa8 00000081 00000000 2003e766
2013e282 201490d2 9c122017 ef322018 9d562023 dfa22039 c2224000 2040aa82 2041c3a2 204282b3
20489082 2057efb2 205a8616 2063cce2 2066aee2 2068a0c2 20768ff7 99522077 de72207b d8224000
00000000 00000000 00000002 00000015 00000001 d04326b2 dd1234dd 0000002d 00000009 04000000
00210001 00000002 00000000 92011d80 00000001 ee1234ee 00000009 03010000 00210001 00033dac
920117d5 00000aa8 00000081 00000000 2004af72 2010a3f2 20128ec2 2017c212 202083c2 9ec22025
c6c22026 a3022034 afb74000 20488602 2053c7c2 20548512 95829672 2063c2e2 e512ee02 20648fb2
2074a5e2 2075d5b2 207aa892 ad32207b ed72ee32 00000000 00000000 00000002 00000015 00000001
3de510d4 dd1234dd 00000031 00000009 04000000 00210002 00000002 00000000 92011d80 00000001
ee1234ee 00000009 03010000 00210002 00033dac 920117d5 00000aa8 00000081 00000000 20109ef2
2011ee42 efc22012 93222013 e2822014 97022017 e182201b e0222025 eaa22027 cab22028 80d3202a
84b22035 c5c2ccb2 2036ebc2 20389672 20508002 95a22051 d3172056 9ee22057 ef42205b cee2eca2
2060ad62 2061c4a2 2063ddb7 20649542 00000000 00000000 00000002 00000019 00000001 f631054a
dd1234dd 00000029 00000009 04000000 00210003 00000002 00000000 92011d80 00000001 ee1234ee
00000009 03010000 00210003 00033dac 920117d5 00000aa8 00000081 00000000 2027d422 203088a2
2031d692 20369542 2037ed92 20409c92 ace22044 9a822046 a9e22047 d3422048 8fb2204a 8a12204b
e172205b c4872060 8f822065 ea222067 c3f24000 00000000 00000000 00000002 00000011 00000001
aeaa0e15 dd1234dd 00000039 00000009 04000000 00210004 00000002 00000000 92011d80 00000001
ee1234ee 00000009 03010000 00210004 00033dac 920117d5 00000aa8 00000081 00000000 2006af12
2017eb47 201a8e76 2025e6d2 20268fa2 a292202b dff74000 2040a152 20469122 20529182 2060aea2
2061c4c2 d722d942 2063c5e2 2064a772 206aa152 206bc322 c7c22070 89d22072 8ad22073 c0b7800f
c187c1a7 c1f7c227 c287c2c7 c2e7c3a7 c3c7800f c3f7c417 c497c4d7 c547c5b7 c5e7c637 c657c677
c6b7c727 c767c7a7 00000000 00000000 00000002 00000021 00000001 a1feebf3 dd1234dd 0000002d
00000009 04000000 00210005 00000002 00000000 92011d80 00000001 ee1234ee 00000009 03010000


00000004 00000001 0000c89c aa1234aa 00053227 0000001c 04000000 00793c29 0003d16e 00000000
Full Event Header
Full Event Size
Header Size
Run number
Source ID 0x79 = Event Builder
84b22035 c5c2ccb2 2036ebc2 20389672 20508002 95a22051 d3172056 9ee22057 ef42205b cee2eca2
2060ad62 2061c4a2 2063ddb7 20649542 00000000 00000000 00000002 00000019 00000001 f631054a
dd1234dd 00000029 00000009 04000000 00210003 00000002 00000000 92011d80 00000001 ee1234ee
00000009 03010000 00210003 00033dac 920117d5 00000aa8 00000081 00000000 2027d422 203088a2
2031d692 20369542 2037ed92 20409c92 ace22044 9a822046 a9e22047 d3422048 8fb2204a 8a12204b
e172205b c4872060 8f822065 ea222067 c3f24000 00000000 00000000 00000002 00000011 00000001
aeaa0e15 dd1234dd 00000039 00000009 04000000 00210004 00000002 00000000 92011d80 00000001
ee1234ee 00000009 03010000 00210004 00033dac 920117d5 00000aa8 00000081 00000000 2006af12
2017eb47 201a8e76 2025e6d2 20268fa2 a292202b dff74000 2040a152 20469122 20529182 2060aea2
2061c4c2 d722d942 2063c5e2 2064a772 206aa152 206bc322 c7c22070 89d22072 8ad22073 c0b7800f
c187c1a7 c1f7c227 c287c2c7 c2e7c3a7 c3c7800f c3f7c417 c497c4d7 c547c5b7 c5e7c637 c657c677
c6b7c727 c767c7a7 00000000 00000000 00000002 00000021 00000001 a1feebf3 dd1234dd 0000002d
00000009 04000000 00210005 00000002 00000000 92011d80 00000001 ee1234ee 00000009 03010000

00000004 00000001 0000c89c aa1234aa 00053227 0000001c 04000000 00793c29 0003d16e 00000000
00000000 50753e27 0ab16f70 00097a2b 00000000 00033dac 00000063 920117d5 00000aa8 00000081
00000018 00020000 40000000 00000000 00000000 00000000 00000000 00000000 00000000 00020000
00000000 dd1234dd 0000002d 00000009 04000000 00610000 00000002 00000000 92011d7f 00000001
ee1234ee 00000009 03010000 00610000 00033dac 920117d5 00000aa8 00000081 00000000 2003e766
2013e282 201490d2 9c122017 ef322018 9d562023 dfa22039 c2224000 2040aa82 2041c3a2 204282b3
20489082 2057efb2 205a8616 2063cce2 2066aee2 2068a0c2 20768ff7 99522077 de72207b d8224000
00000000 00000000 00000002 00000015 00000001 d04326b2 dd1234dd 0000002d 00000009 04000000
00610001 00000002 00000000 92011d80 00000001 ee1234ee 00000009 03010000 00610001 00033dac
920117d5 00000aa8 00000081 00000000 2004af72 2010a3f2 20128ec2 2017c212 202083c2 9ec22025
c6c22026 a3022034 afb74000 20488602 2053c7c2 20548512 95829672 2063c2e2 e512ee02 20648fb2
2074a5e2 2075d5b2 207aa892 ad322 [illegible] 00000000 00000000 00000002 00000015 00000001
3de510d4 dd1234dd 00000031 0000 [illegible] 00610002 00000002 00000000 92011d80 00000001
ee1234ee 00000009 03010000 0061 [illegible] 920117d5 00000aa8 00000081 00000000 20109ef2
2011ee42 efc22012 93222013 e2822014 97022017 e182201b e0222025 eaa22027 cab22028 80d3202a
84b22035 c5c2ccb2 2036ebc2 20389672 20508002 95a22051 d3172056 9ee22057 ef42205b cee2eca2
2060ad62 2061c4a2 2063ddb7 20649542 00000000 00000000 00000002 00000019 00000001 f631054a
dd1234dd 00000029 00000009 04000000 00610003 00000002 00000000 92011d80 00000001 ee1234ee
00000009 03010000 00610003 00033dac 920117d5 00000aa8 00000081 00000000 2027d422 203088a2
2031d692 20369542 2037ed92 20409c92 ace22044 9a822046 a9e22047 d3422048 8fb2204a 8a12204b
e172205b c4872060 8f822065 ea222067 c3f24000 00000000 00000000 00000002 00000011 00000001
aeaa0e15 dd1234dd 00000039 00000009 04000000 00610004 00000002 00000000 92011d80 00000001
ee1234ee 00000009 03010000 00610004 00033dac 920117d5 00000aa8 00000081 00000000 2006af12
2017eb47 201a8e76 2025e6d2 20268fa2 a292202b dff74000 2040a152 20469122 20529182 2060aea2
2061c4c2 d722d942 2063c5e2 2064a772 206aa152 206bc322 c7c22070 89d22072 8ad22073 c0b7800f
c187c1a7 c1f7c227 c287c2c7 c2e7c3a7 c3c7800f c3f7c417 c497c4d7 c547c5b7 c5e7c637 c657c677
c6b7c727 c767c7a7 00000000 00000000 00000002 00000021 00000001 a1feebf3 dd1234dd 0000002d
00000009 04000000 00610005 00000002 00000000 92011d80 00000001 ee1234ee 00000009 03010000

1600
fragments

```
00000004 00000001 0000c89c aa1234aa 00053227 0000001c 04000000 00793c29 0003d16e 00000000
00000000 50753e27 0ab16f70 00097a2b 00000000 00033dac 00000063 920117d5 00000aa8 00000081
00000018 00020000 40000000 00000000 00000000 00000000 00000000 00000000 00000000 00020000
00000000 dd1234dd 0000002d 00000009 04000000 00610000 00000002 00000000 92011d7f 00000001
ee1234ee 00000009 03010000 00610000 00033dac 920117d5 00000aa8 00000081 00000000 2003e766
2013e282 201490d2 9c122017 ef322018 9d562023 dfa22039 c2224000 2040aa82 2041c3a2 204282b3
20489082 2057efb2 205a8616 2063cce2 2066aee2 2068a0c2 20768ff7 99522077 de72207b d8224000
0 0000 0000000 [illegible] 000001 d04326b2 dd1234dd 0000002d 00000009 04000000
0 0002 0000000 [illegible] 0000 [illegible] 00000009 03010000 00610001 00033dac
9 0aa8 0000008 [illegible] 04af [illegible] 20128ec2 2017c212 202083c2 9ec22025
c6c 26 a3022034 afb7400 [illegible] 53c7c 548512 95829672 2063c2e2 e512ee02 20648fb2
2074 5e2 2075d5b2 207aa892 a 207b ed72ee 00000000 00000000 00000002 00000015 00000001
3de510d4 dd1234dd 00000031 000 0009 04000 000 00610002 00000002 00000000 92011d80 00000001
ee1234ee 00000009 03010000 00610002 00033dac 920117d5 00000aa8 00000081 00000000 20109ef2
2011ee42 efc22012 93222013 e2822014 97022017 e182201b e0222025 eaa22027 cab22028 80d3202a
84b22035 c5c2ccb2 2036ebc2 20389672 20508002 95a22051 d3172056 9ee22057 ef42205b cee2eca2
2060ad62 2061c4a2 2063ddb7 20649542 00000000 00000000 00000002 00000019 00000001 f631054a
dd1234dd 00000029 00000009 04000000 00610003 00000002 00000000 92011d80 00000001 ee1234ee
00000009 03010000 00610003 00033dac 920117d5 00000aa8 00000081 0000000 2027d422 203088a2
2031d692 20369542 2037ed92 409c92 ace22044 9a822046 a9e22047 3422048 8fb2204a 8a12204b
e172205b c4872060 8f82206 7 c3f24000 00000000 0000000 2 00000011 00000001
aeaa0e15 dd1234dd 0000003 9 04000000 00610004 0000000 0 92011d80 00000001
ee1234ee 00000009 03010000 00610004 00033dac 920117d5 00000aa8 00000081 00000000 2006af12
2017eb47 201a8e76 2025e6d2 20268fa2 a292202b dff74000 2040a152 20469122 20529182 2060aea2
2061c4c2 d722d942 2063c5e2 2064a772 206aa152 206bc322 c7c22070 89d22072 8ad22073 c0b7800f
c187c1a7 c1f7c227 c287c2c7 c2e7c3a7 c3c7800f c3f7c417 c497c4d7 c547c5b7 c5e7c637 c657c677
c6b7c727 c767c7a7 00000000 00000000 00000002 00000021 00000001 a1feebf3 dd1234dd 0000002d
00000009 04000000 00610005 00000002 00000000 92011d80 00000001 ee1234ee 00000009 03010000
```

Fragment Header

0x61: MDT Barrel side A (module 2)

Run number

data

Trailer

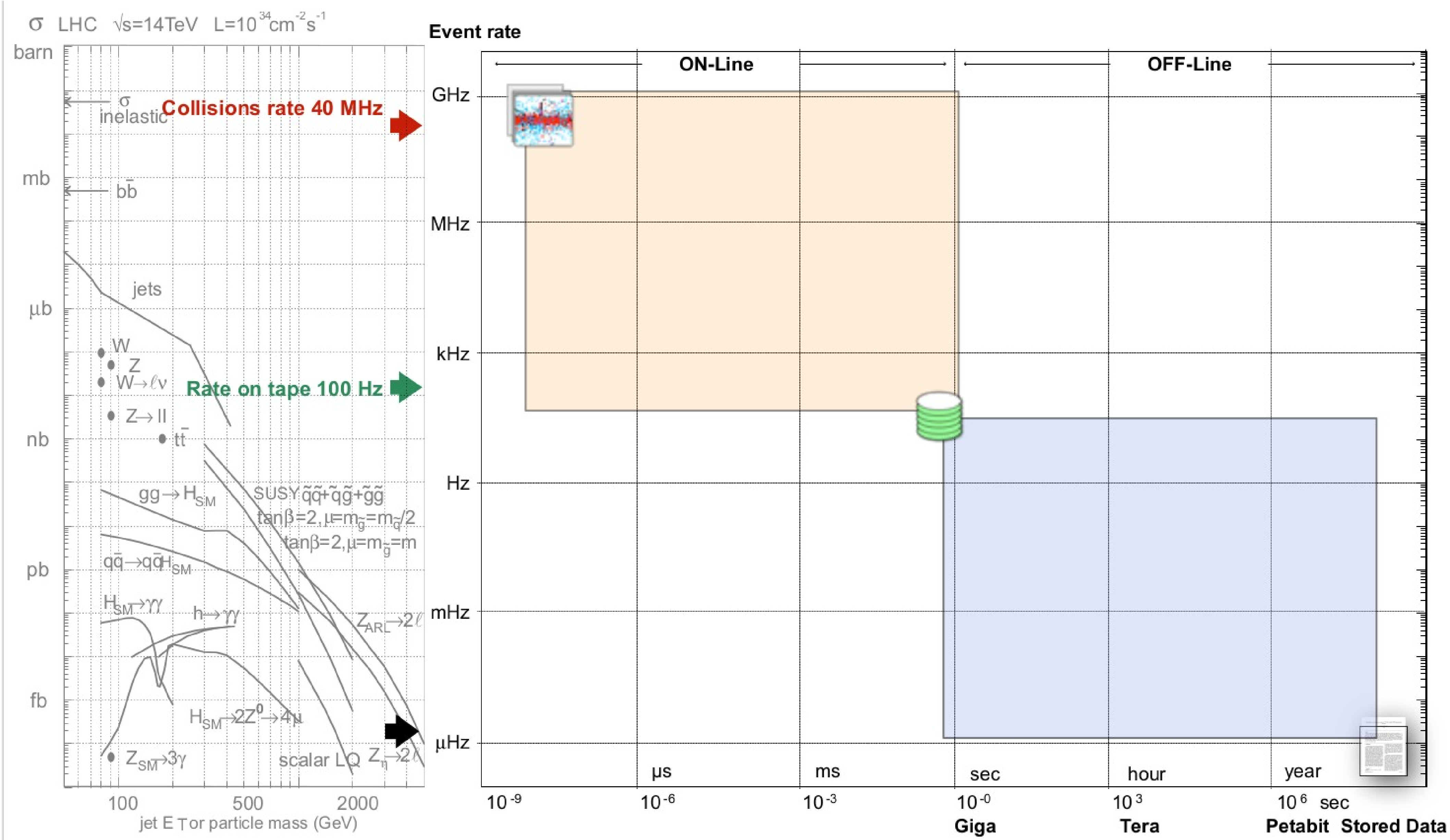
Offset	Word hex	Word dec	Description
0x00000000	**0xaa1234aa**	2853319850	[full event marker]
0x00000001	0x00019b63	105315	fragment size (words)
0x00000002	0x00000069	105	header size (words)
0x00000003	0x05000000	83886080	version: 5.0-0.0
0x00000004	0x007c0000	8126464	source_id: TDAQ_HLT, module=0 (opt=0)
0x00000005	0x00000001	1	number of status words
0x00000006	0x00000000	0	status[0]
0x00000007	0x00000000	0	check sum type
0x00000008	0x5654a93f	1448388927	bunch cros. time in seconds
0x00000009	0x017c5569	24925545	bunch cros. time, additional nanoseconds
0x0000000a	0x00003a51	14929	global event identifier LS
0x0000000b	0x00000000	0	global event identifier MS
0x0000000c	0x00000000	0	run type
0x0000000d	0x00045fb4	286644	run number
0x0000000e	0x00000050	80	lumi block
0x0000000f	0x78000045	2013265989	lvl1 identifier
0x00000010	0x00000001	1	bunch cros. identifier
0x00000011	0x000000a0	160	lvl1 trigger type
0x00000012	0x00000001	1	compression type
0x00000013	0x000401b4	262580	uncompressed payload size
0x00000014	0x00000030	48	number of lvl1 trigger info words
0x00000015	0x00020000	131072	lvl1 trigger info[0]
0x00000016	0x80000000	2147483648	lvl1 trigger info[1]

Trends

TDAQ rules 25 years ago

- Clear separation between **online** and **offline**
  - Analisys and calibration done offline
  - Online code must be more robust than offline one
    - HLT algorithms “derived” from offline
- First level trigger in **hardware**
  - Impossible to read-out the whole detector @40 MHz
- Data flow based on the concept of **event**
  - Aggregation of data from a given bunch crossing
- **Raw data** on disk
  - Do not touch the raw data online

ISOTDAQ 2013: S. Cittolin

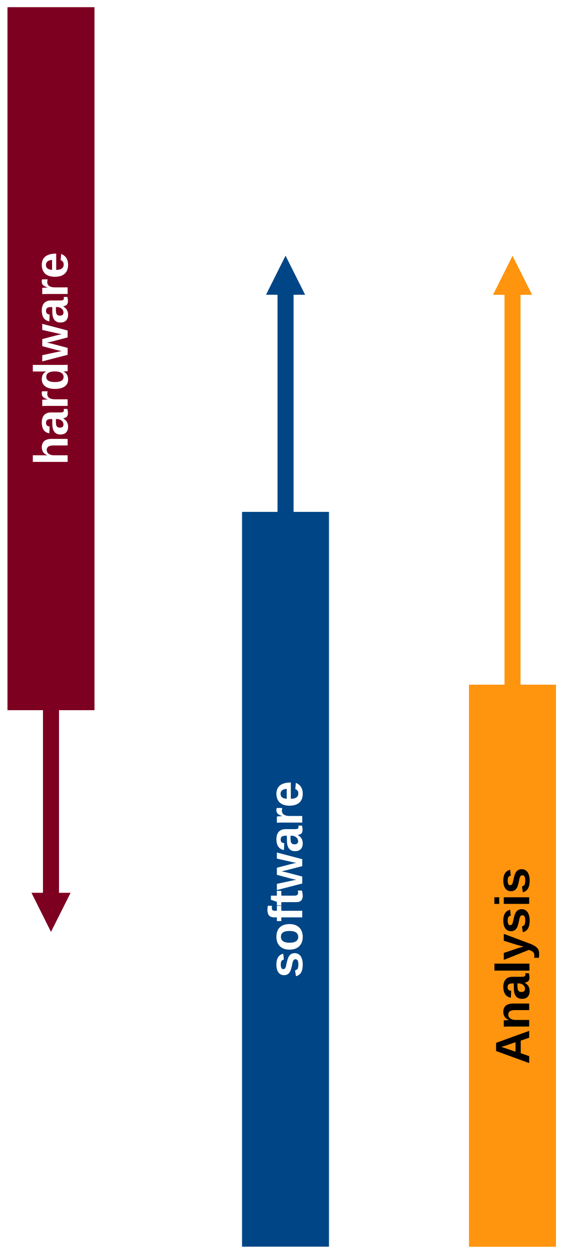


Trends

- **More data**
  - Physics goals require more statistics
- **Triggerless** DAQ or reconstruction in **HW**
  - Alice/LHCb vs ATLAS/CMS
- Less and less **raw data** on disk
  - Some amount still mandatory for trigger validation
- Adopt **commodity** solutions whenever possible
  - FPGA, GPU, file system, containers, …
- Fall of the **online**/**offline** wall
  - Physics analisys begins in T/DAQ

Trends

- Experiment/detector
  - Produces **physics**
- Data AcQuisition
  - Extracts **physics** from detector
- (Offine) Analysis
  - Extracts **physics** from data



DAQ concepts

isotdaq2024

- An **heterogeneous** agenda
  - **30** lectures
  - **14** labs
- DAQ and Trigger hardware
  - ADC, TDC, electronics, FPGA, μcontrollers, network, buses
- Software
  - General programming skills, run control and monitoring, data flow, GPU, machine learning, ...
- DAQ system design
  - From lab, to test beam, to LHC and upgrades

Conclusion

- **Hands on** school
  - Labs are the core of this school
- Unique **opportunity** to see and touch different technologies
  - To understand what you can do with them and where to start
  - The less you know about a lab the more you must play
- Unique **opportunity** to discuss with the experts
  - Don't be shy
  - **Bother the experts**