

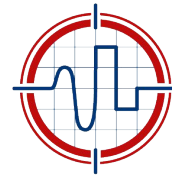
# Trigger Architectures and hardware

---

D. Rabadý, CERN

ISOTDAQ 2022: 12<sup>th</sup> International School of Trigger and Data Acquisition

University of Catania, Catania, 17 June 2022



**ISOTDAQ**

International School of Trigger  
and Data Acquisition



# What's on the (trigger) menu today?

---

## Trigger architectures and hardware

- From simple, home-made trigger systems... to highly complex, multi-level triggers
- Dead time
- First-level trigger systems
- High-level trigger systems

# Reminder

---

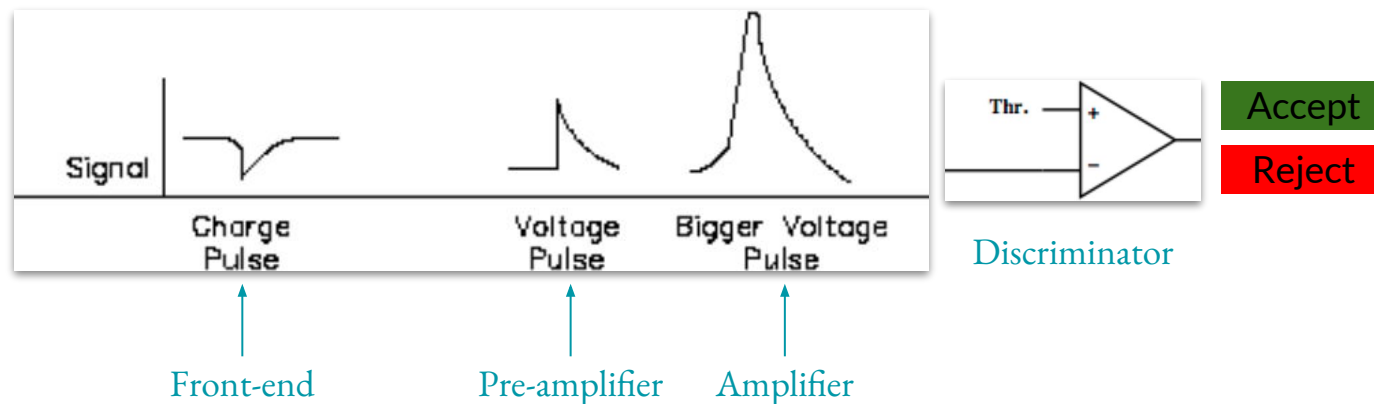
## Trigger basic requirements

- Need **high efficiency** for selecting processes for physics analysis
- Need **large reduction of rate** from unwanted high-rate processes
- **Robustness** is essential
- **Highly flexible**, to react to changing conditions
- System must be **affordable**

# The simplest trigger systems

**Source:** Use the signals from the detector front-ends

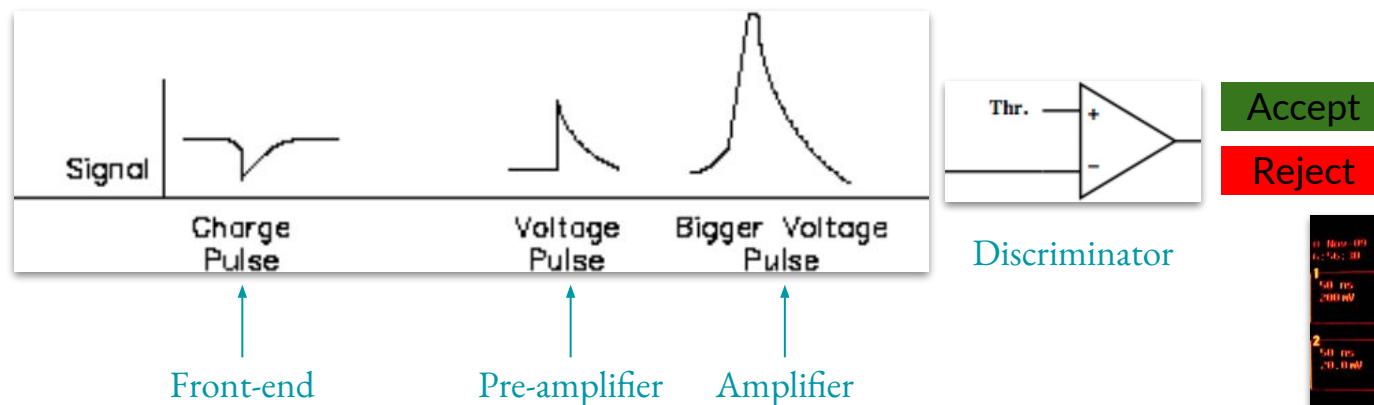
- **Binary:** Tracking detectors (pixels, strips)
- **Analog:** Tracking detectors, time of flight detectors, calorimeters, ...



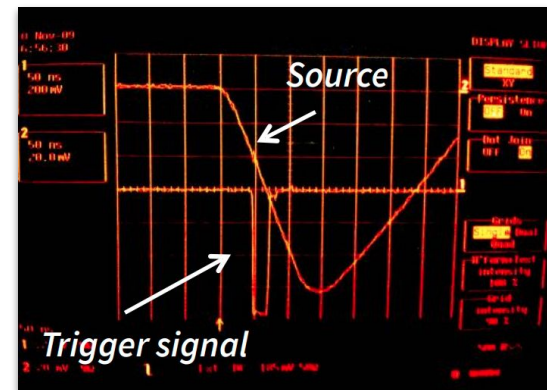
# The simplest trigger systems

**Source:** Use the signals from the detector front-ends

- **Binary:** Tracking detectors (pixels, strips)
- **Analog:** Tracking detectors, time of flight detectors, calorimeters, ...



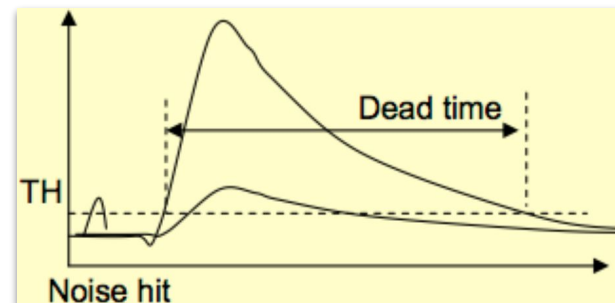
- Most trivial trigger algorithm:  $\text{Signal} > \text{Threshold}$ 
  - Apply the lowest possible threshold
  - Find best compromise between hit efficiency and noise rate



# Detector signals characteristics

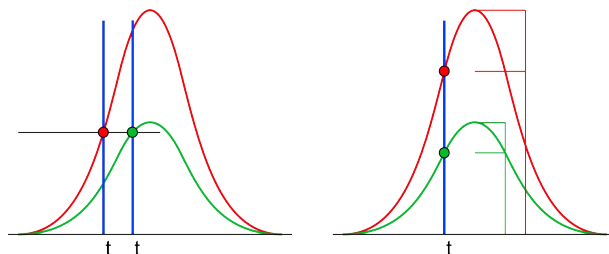
## Pulse width

- Limits the effective hit rate
- Must be adapted to the desired trigger rate



## Time walk

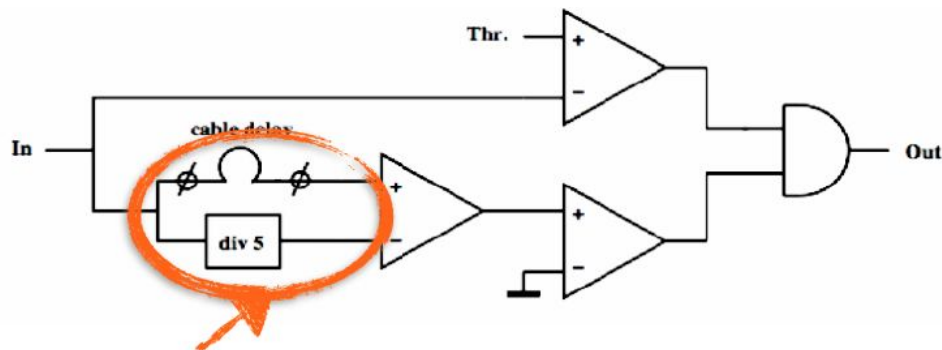
- The threshold-crossing time depends on the signal amplitude
- Must be minimal in good trigger systems



Time walk can be suppressed by triggering on the **total signal fraction**

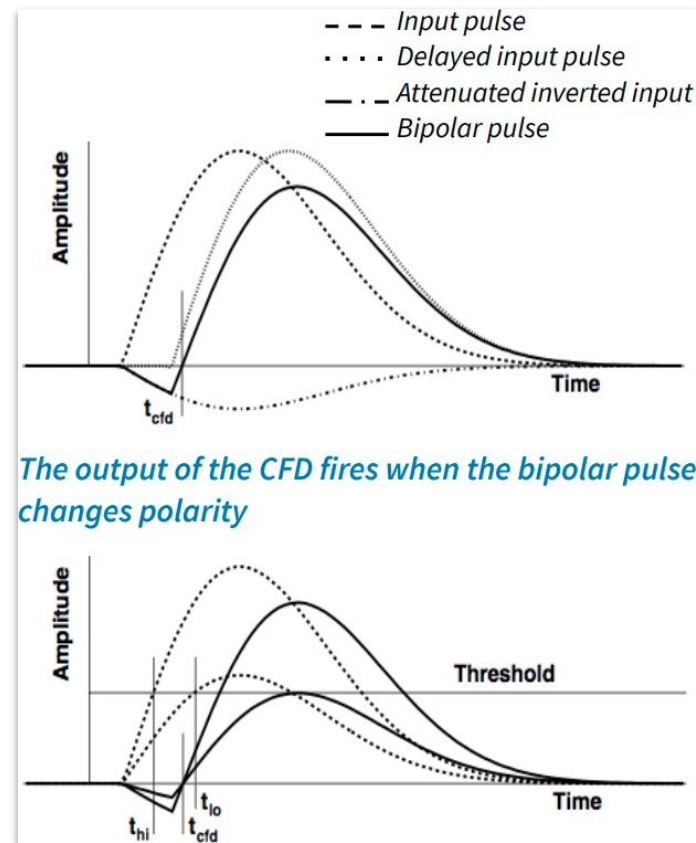
- Applicable on same-shape input signals with different amplitude
  - Scintillator detectors and photomultipliers

# The constant fraction discriminator



Attenuation + configurable delay

If delay too short, the unit works as a normal discriminator because the output of the normal discriminator fires later than the CFD part.



The output of the CFD fires when the bipolar pulse changes polarity

# Trigger logic implementation

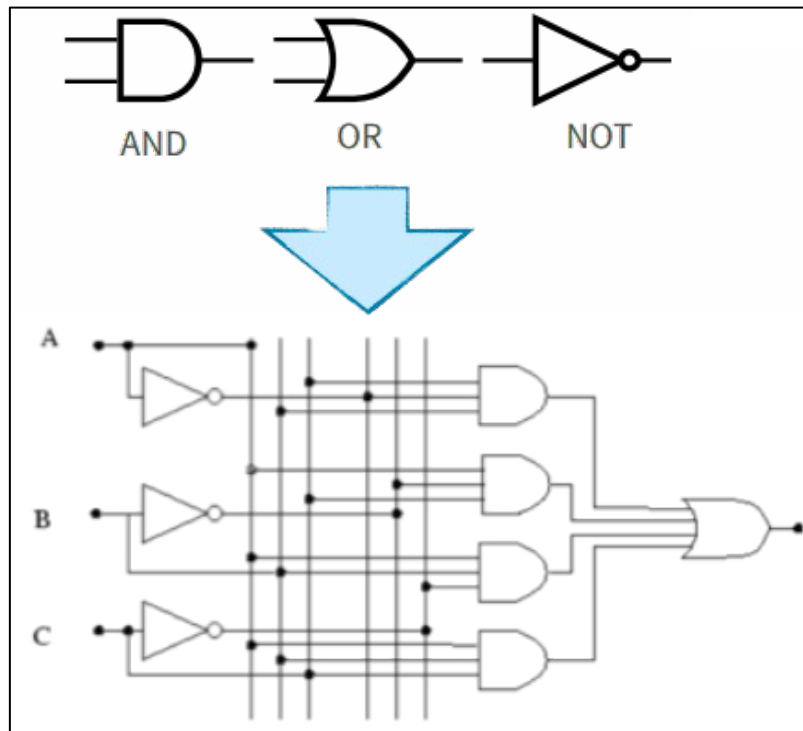
In digital domain, any decision logic can be expressed as **sequence of boolean operations**:

- **Combinatorial**

- Summing, decoders, multiplexers, ...
- Data **propagates as a wave** through the logic

- **Sequential**

- Flip-flops, registers, counters, ...
- Operations happen at **well defined times** and in a **well defined order**



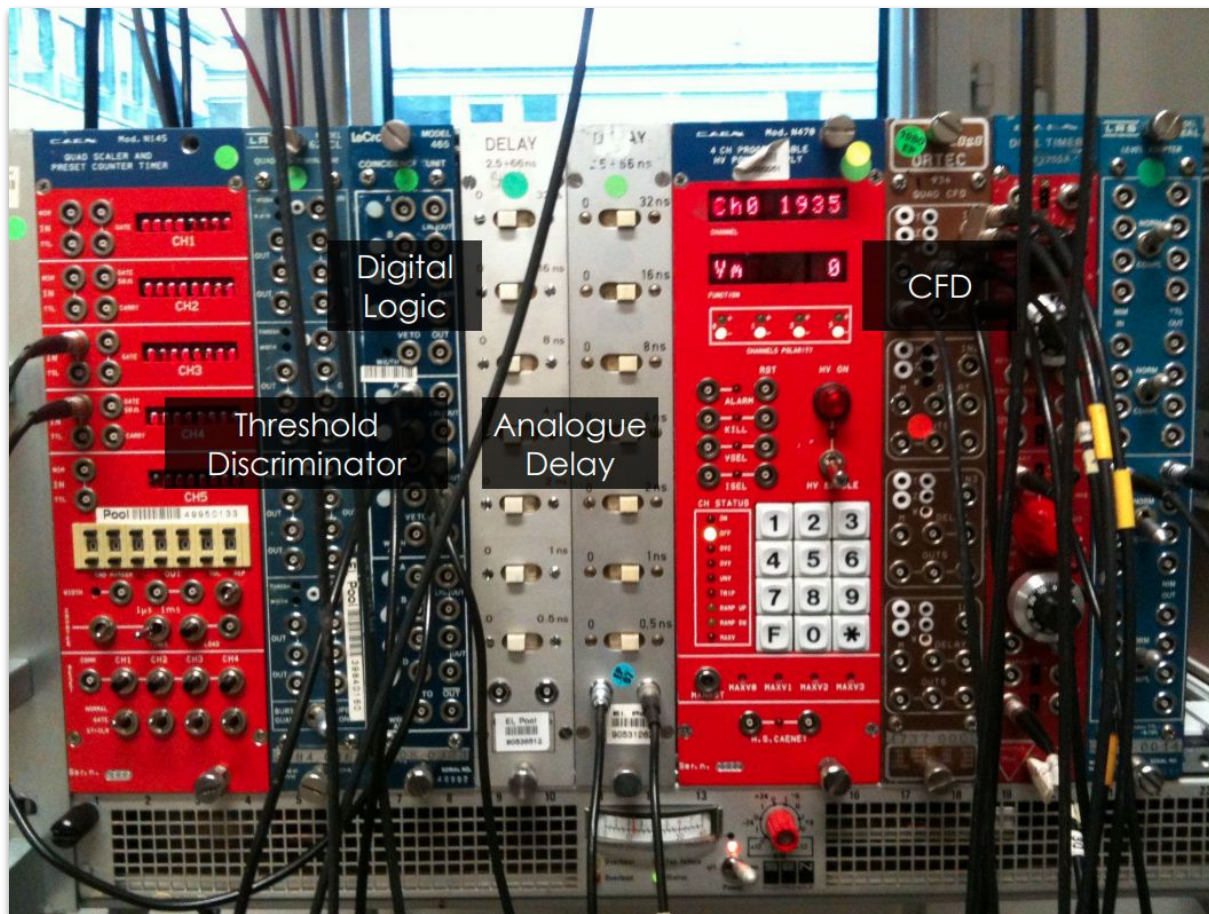


# Building your own trigger system

A simple trigger system can start with a **NIM crate**

- Common support for electronic modules
- Standard impedance, connections, logic levels (negative!)





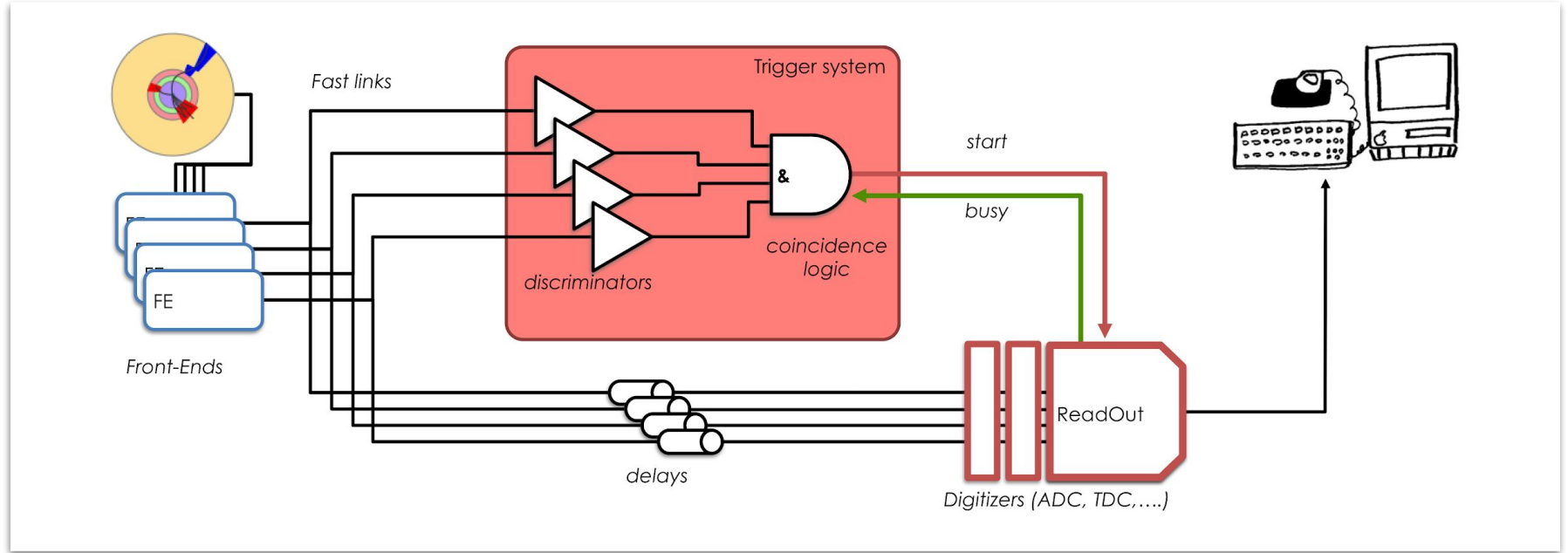
# Back to trigger requirements:

High efficiency

Low dead-time

Fast decision

# A simple trigger system



Incoming event rate can temporarily exceed processing rate  
Trigger signals are rejected if "busy" signal is high

# Deadtime

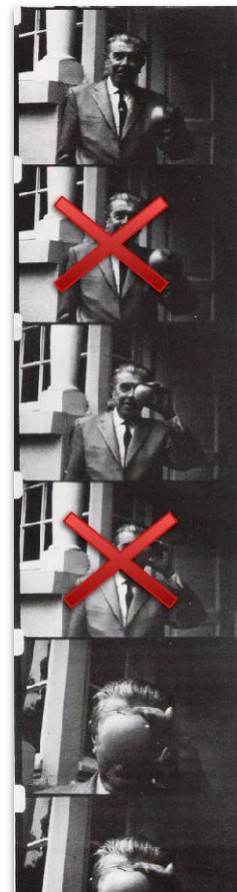
The key parameter in high speed TDAQ systems design

- The fraction of the acquisition time when no events can be recorded.
  - Typically of the order of few %
- Reduces the overall system efficiency

Arises when a given processing step takes a finite amount of time

- Readout dead-time
- Trigger dead-time
- Operational dead-time

**Example:** Sending data from some detector modules requires a certain time. If an event is being read out no others can be accepted, even if it is interesting.



# Maximising data-recording rate

$R_{in}$  = Trigger rate (average)

$R_{out}$  = Readout rate

$T_d$  = processing time of one event

Fraction of lost events  $R_{out} \cdot T_d$

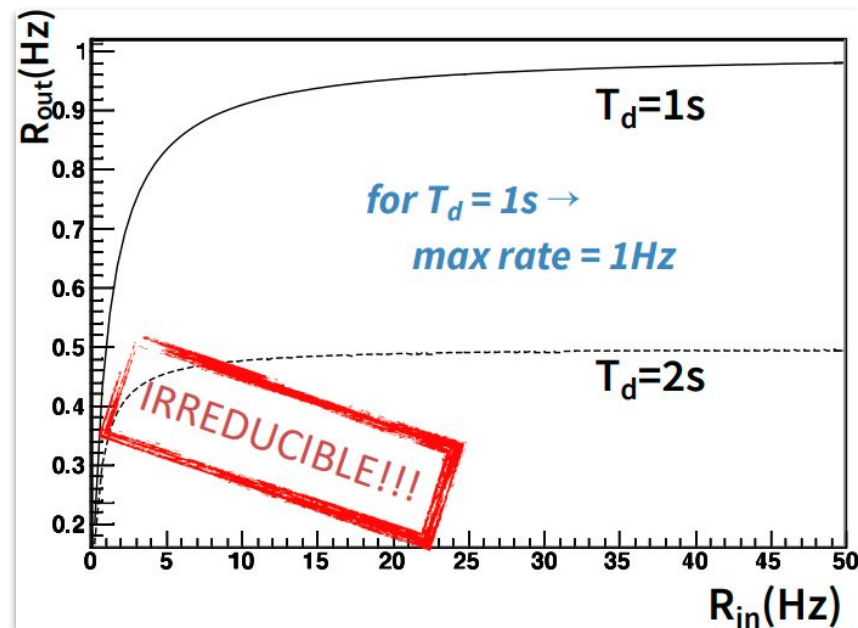
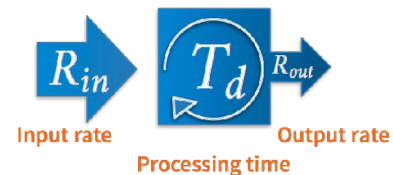
Number of events read  $R_{out} = (1 - R_{out} \cdot T_d) \cdot R_{in}$

Fraction of  
surviving events

$$\frac{R_{out}}{R_{in}} = \frac{1}{1 + R_{in} T_d}$$

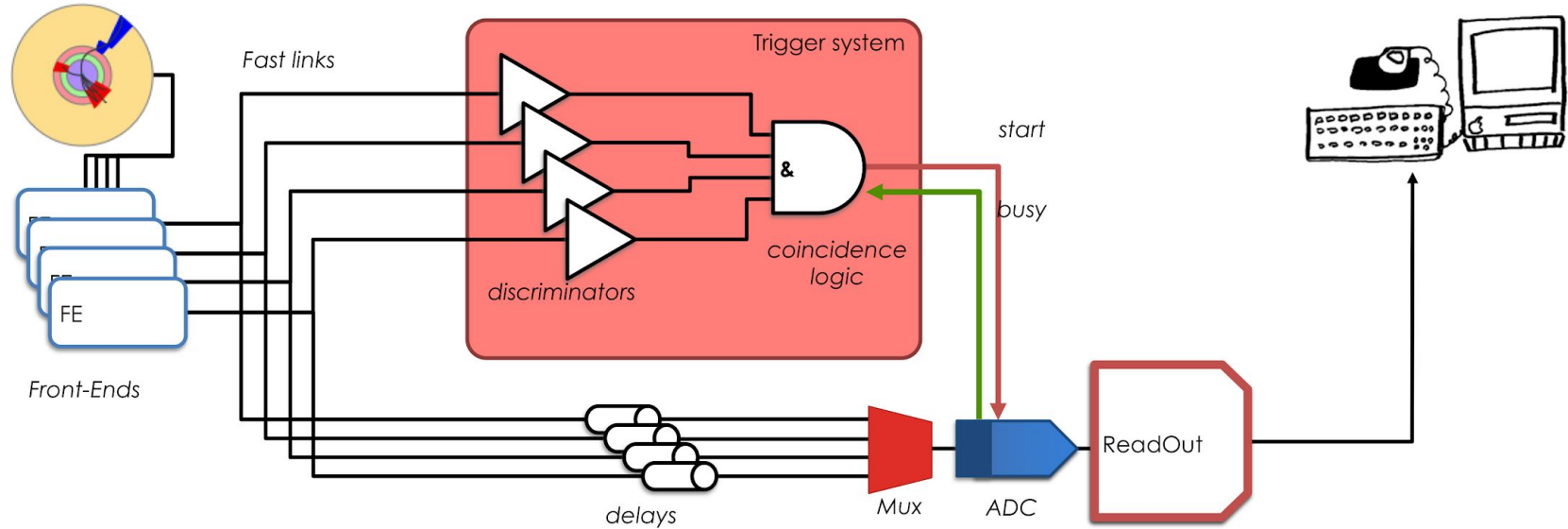
► For instance:  $R_{in} = 1/T_d \rightarrow$  dead-time = 50%

For high efficiency:  $R_{in} \cdot T_d \ll 1$

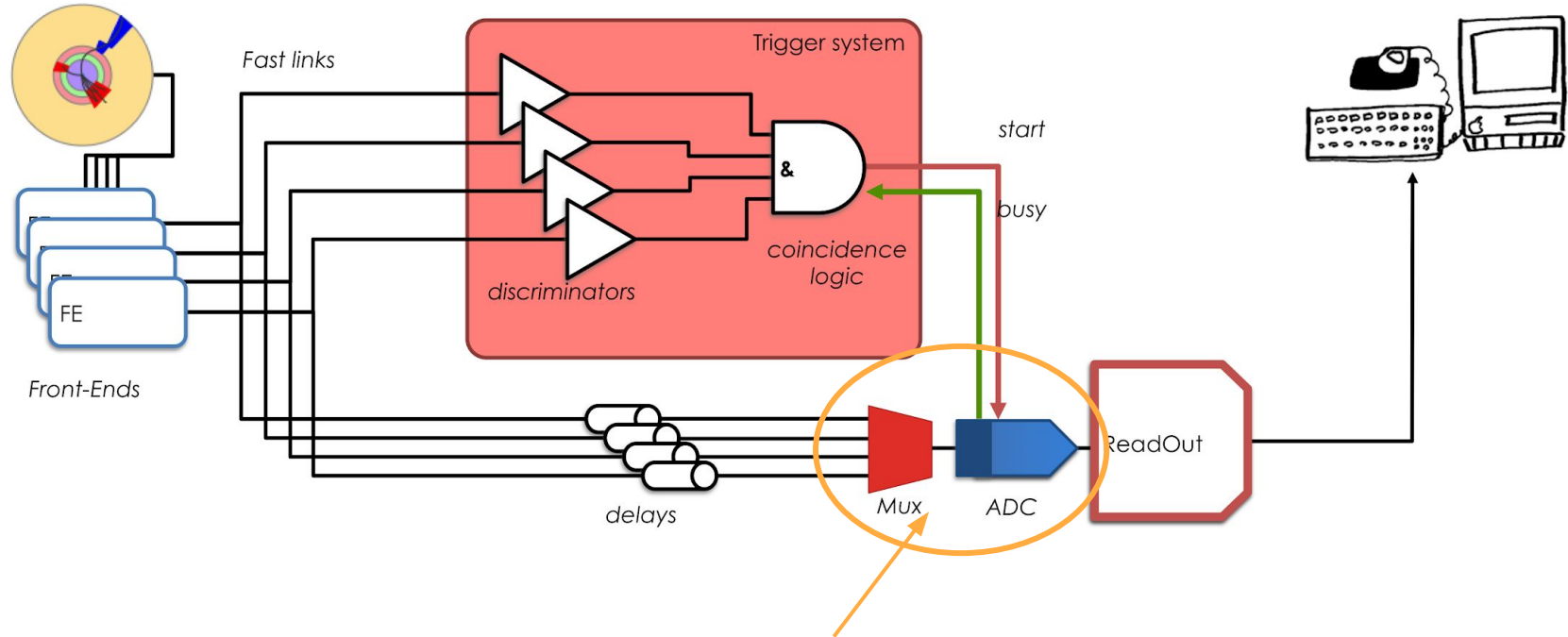




# Deadtime in our simple trigger system



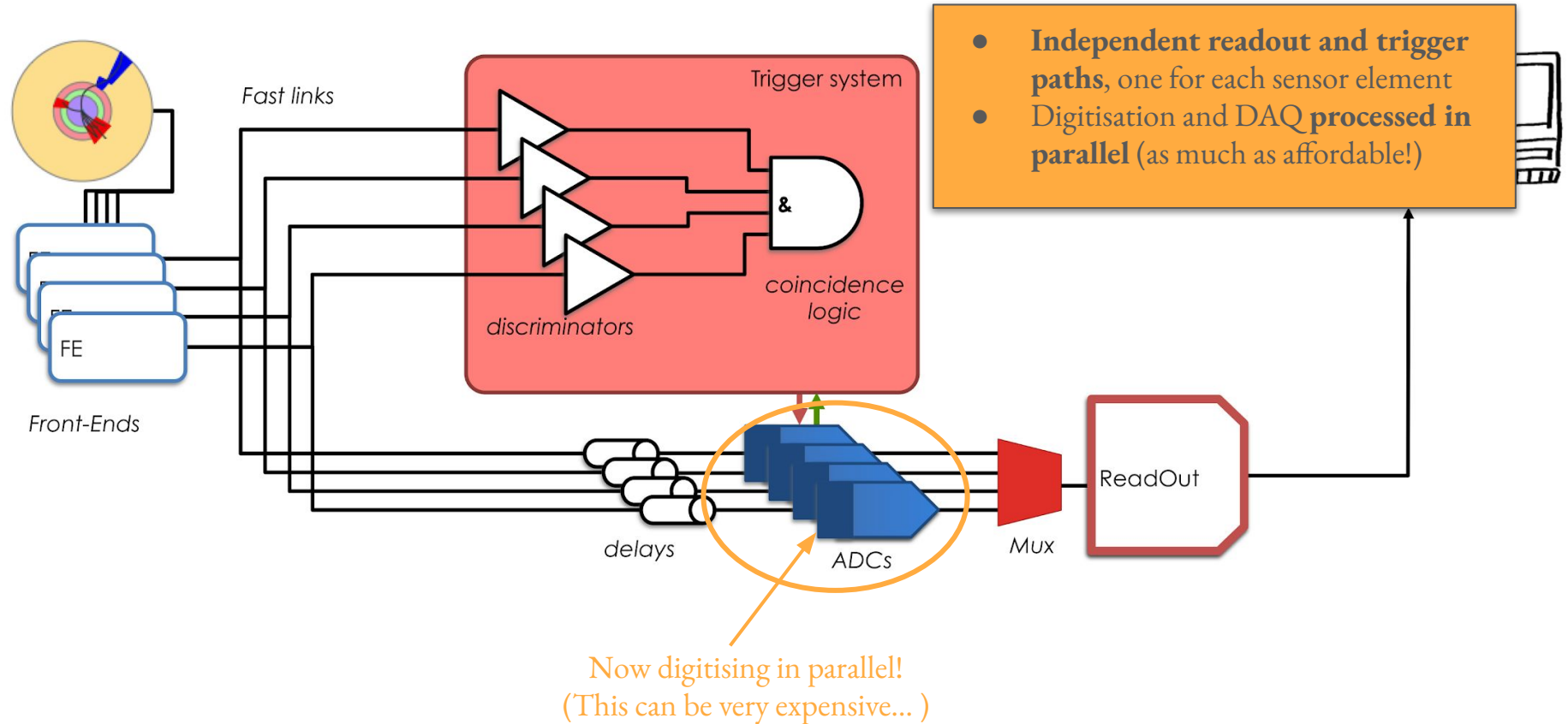
# Deadtime in our simple trigger system



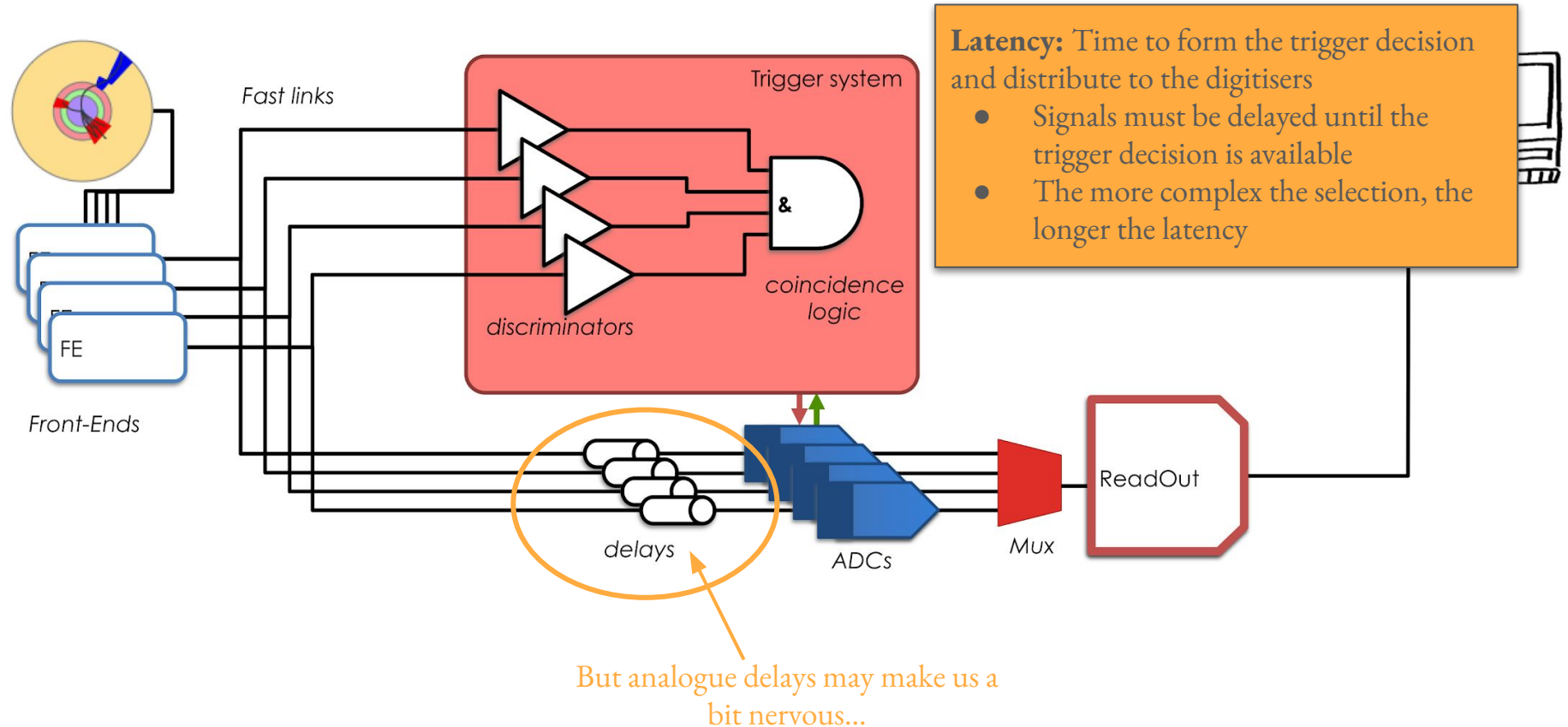
If ADC is a critical step for deadtime,  
this is not a good idea!



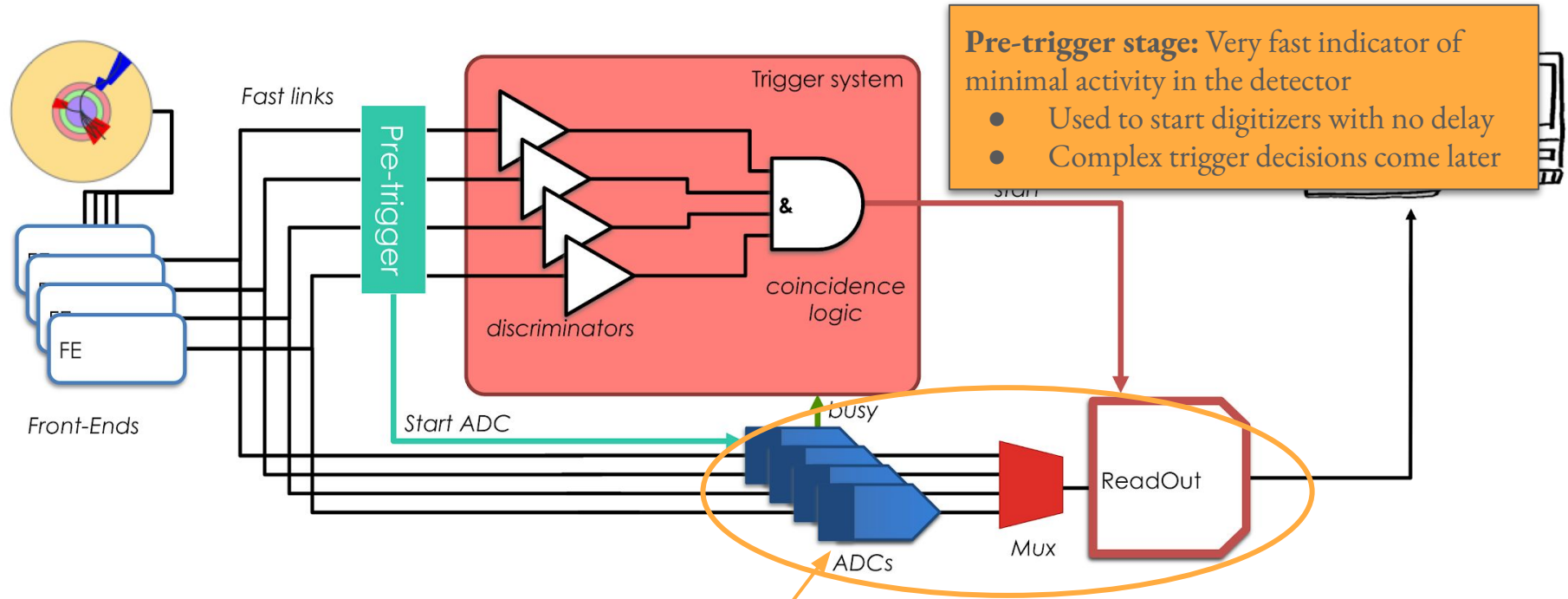
# Deadtime in our simple trigger system



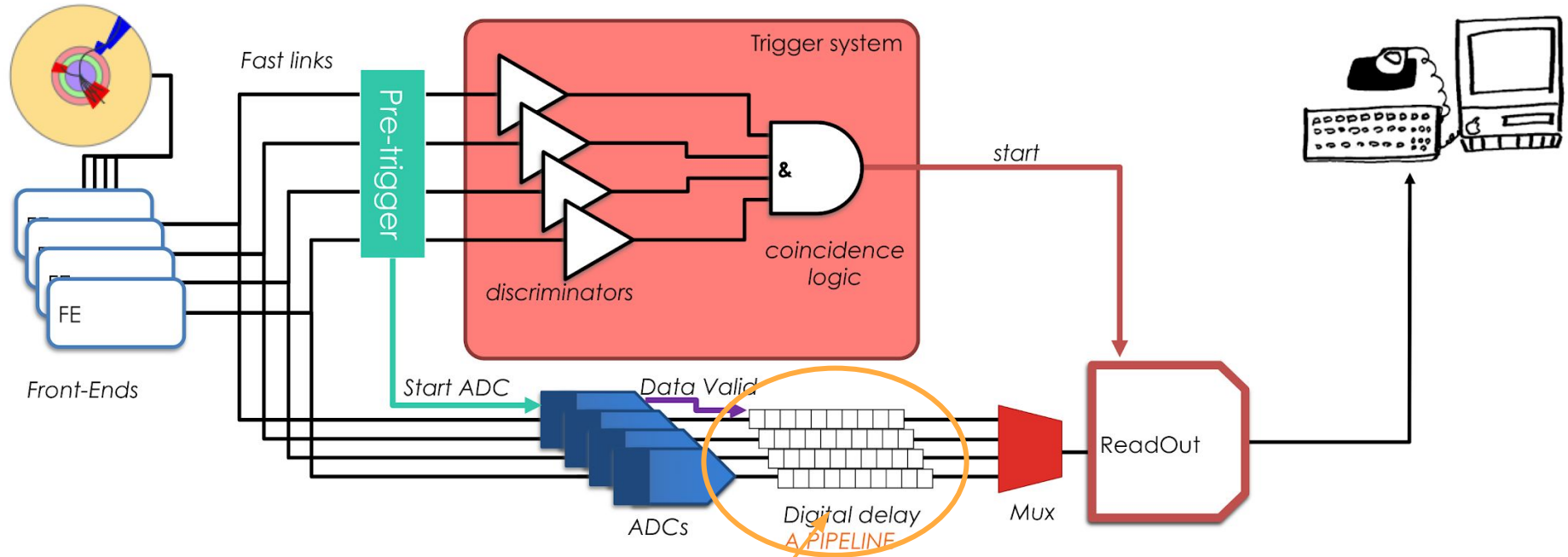
# Latency in our simple trigger system



# Pre-trigger in our simple trigger system

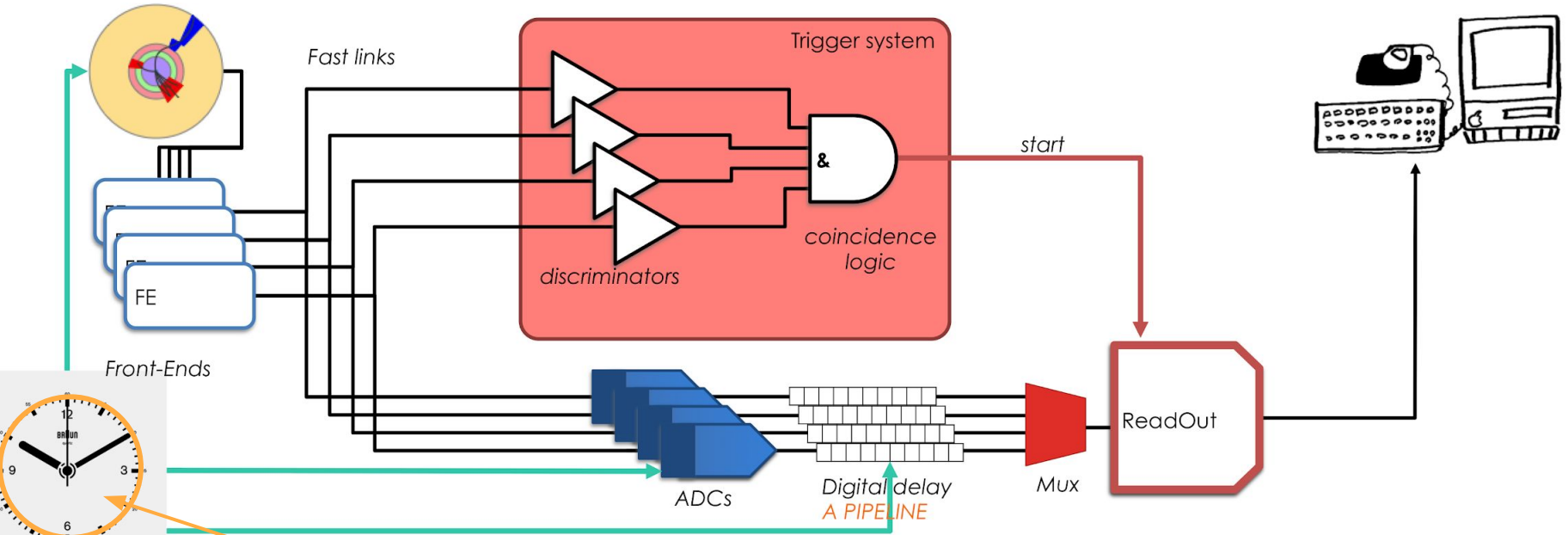


# Pre-trigger in our simple trigger system



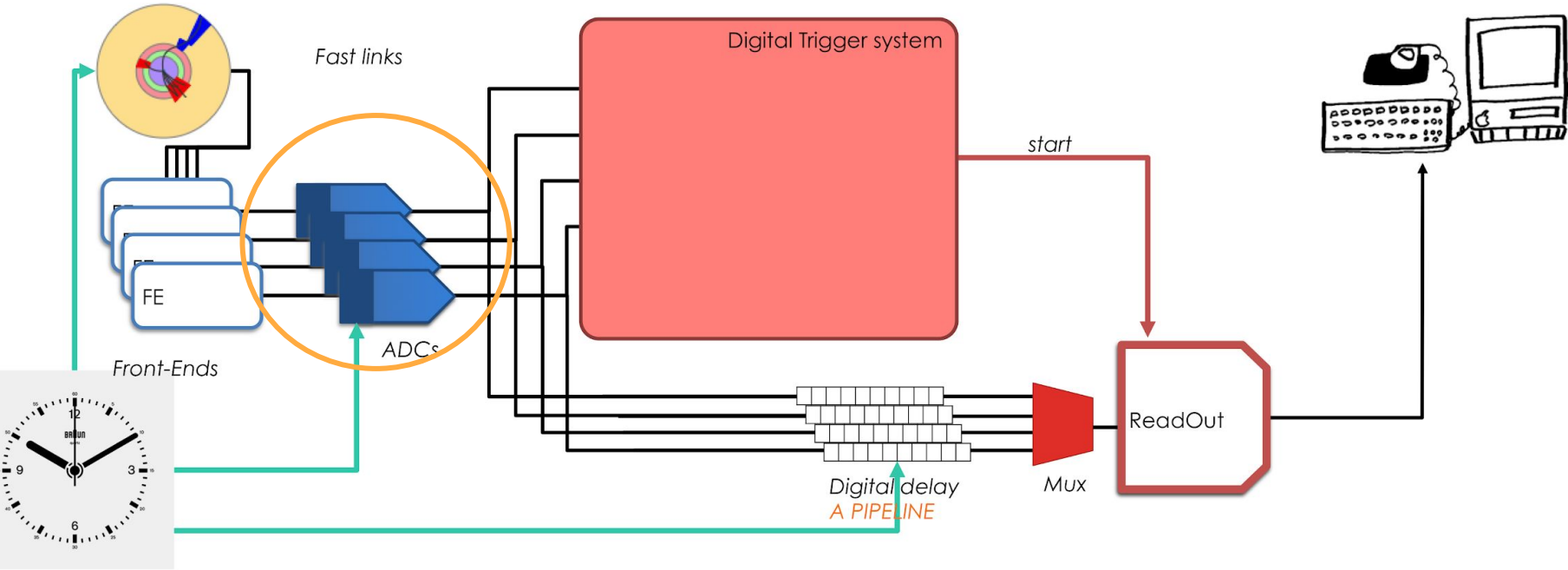
Can store the result of each decision  
in RAM until trigger decides

# Our simple trigger system with bunched colliders

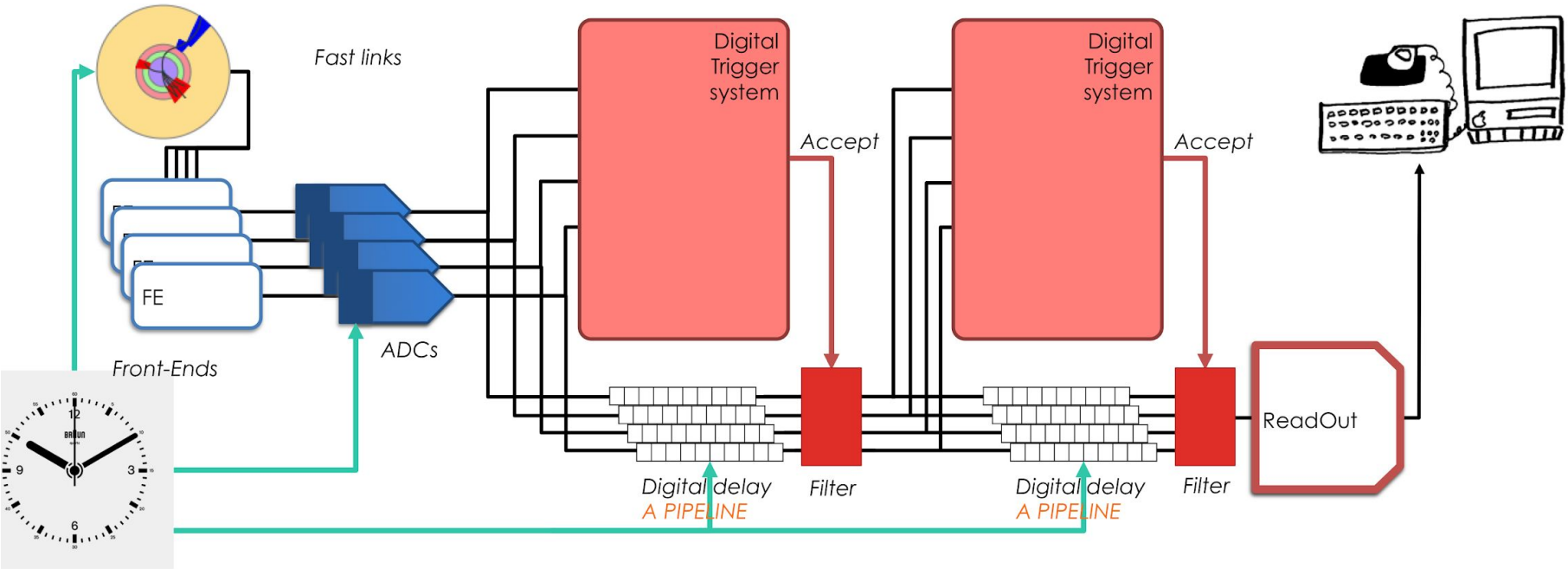


Have an authoritative clock, the collisions themselves!  
No need for a dedicated pre-trigger.

# Our simple digital trigger system



# A trigger system with multiple layers



# Multilayer triggers

---

- Each stage reduces the rate, so later stages can have longer latency
- Complexity of algorithms increases at each level

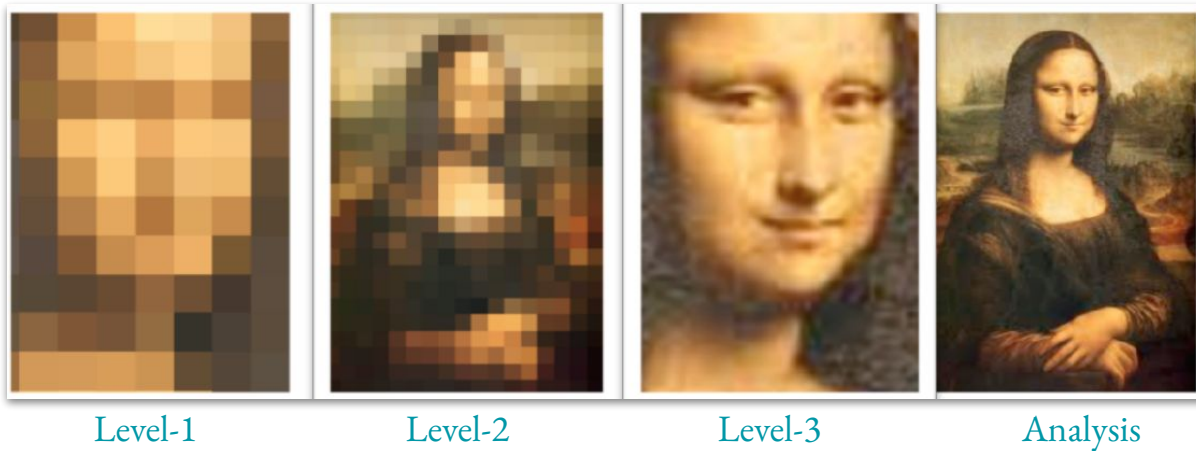
Dead-time is the sum of the trigger dead-time, summed over the trigger levels, and the readout dead-time.



# Multilayer triggers

## Adopted in large experiments

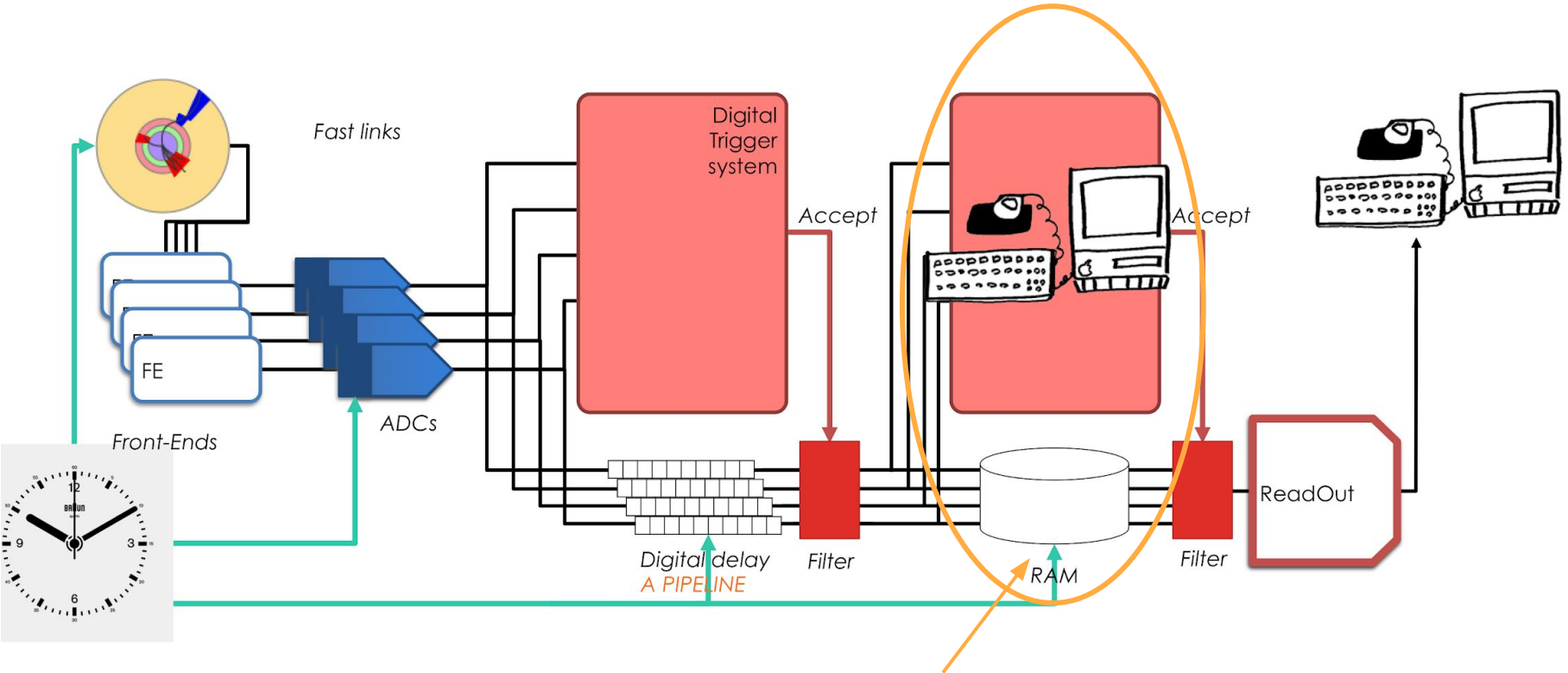
- More and more complex algorithms are applied on lower and lower data rates
  - Earlier levels with short latency, working at higher rates
  - Higher levels with more complex algorithms, but longer latency
  - Efficiency for the desired physics must be kept high at all levels, since rejected events are lost forever



### *LHC experiments in Run-1*

Exp	#Levels
ATLAS	3
CMS	2
LHCb	3
ALICE	4

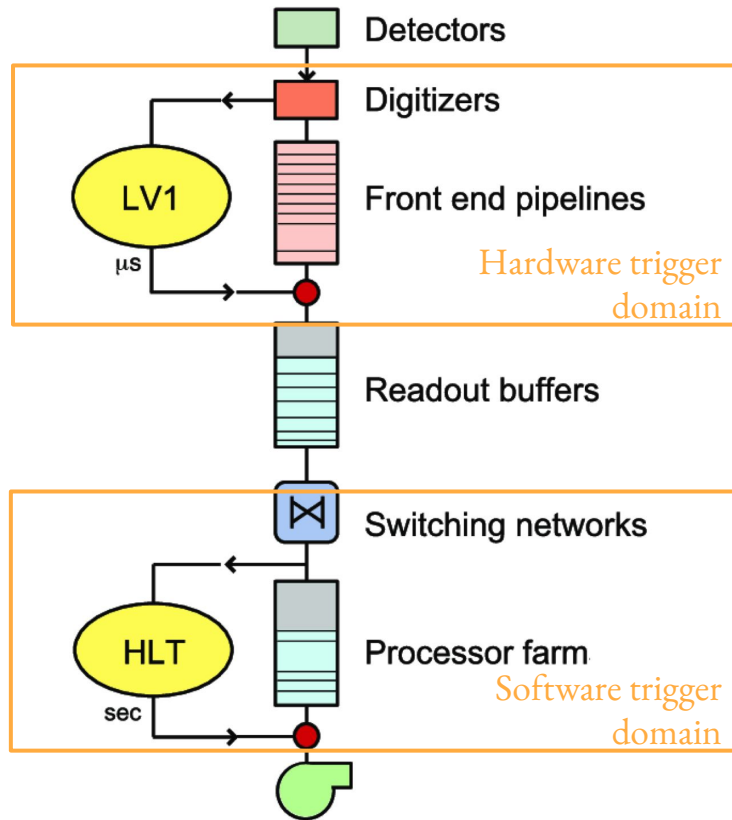
# A trigger system with multiple layers



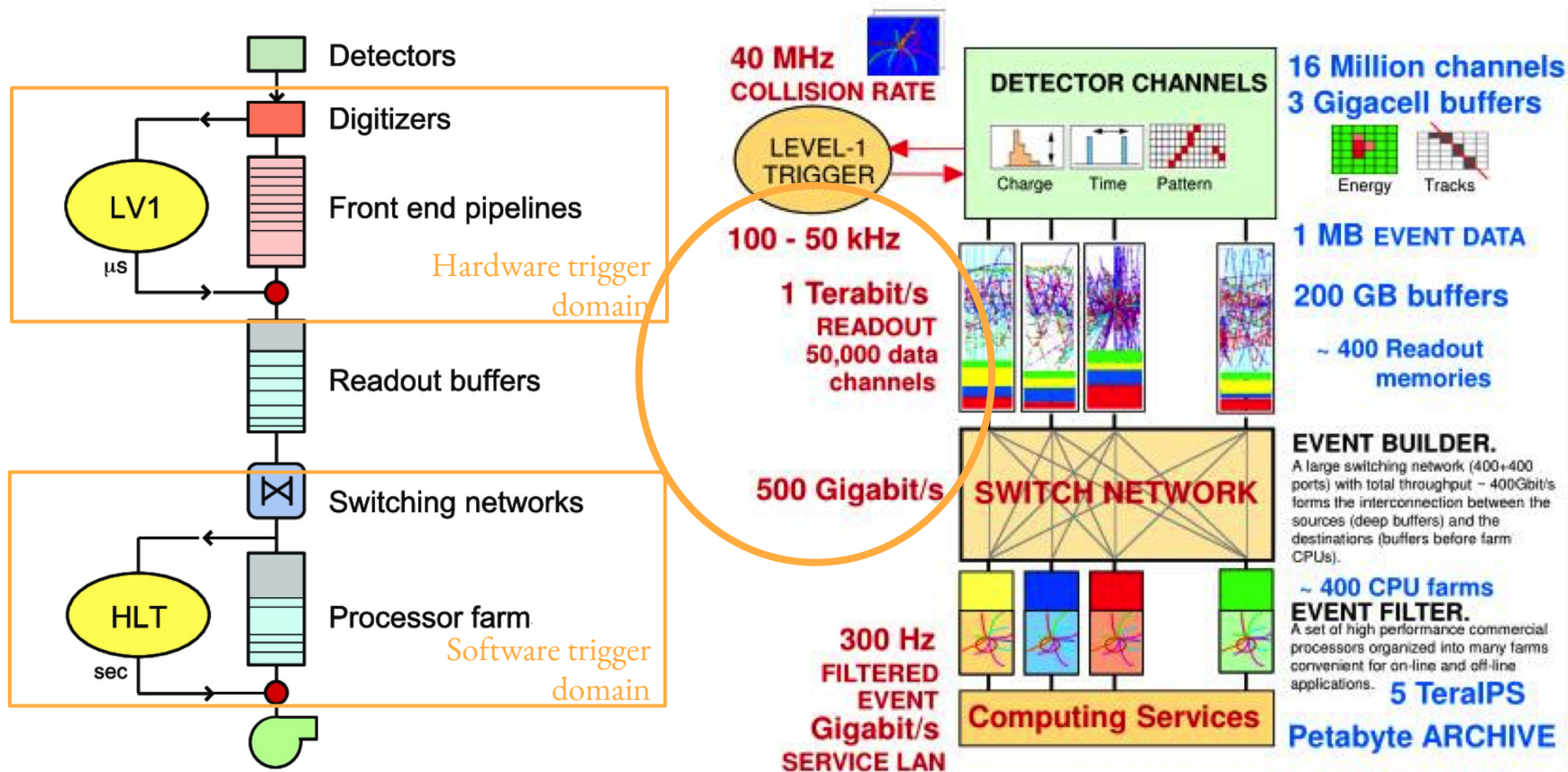
If your input rate is low enough...

# A trigger system with multiple layers

This may have seemed familiar to people with knowledge of e.g. the CMS trigger...



# But "low enough rate" is always relative...



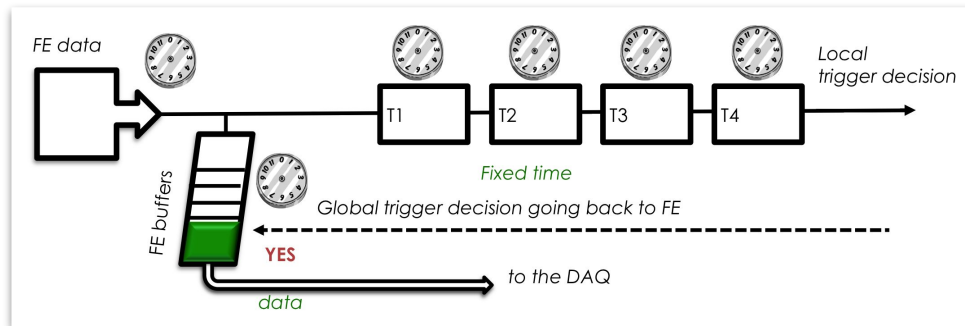
# Synchronous or asynchronous?

**Synchronous:** operates phase-locked with authoritative clock

- Data move in lockstep with the clock through the trigger chain: **Fixed latency**
- The data, held in storage pipelines, are **either sent forward or discarded**
- Used for **triggers in collider experiments**, exploiting the accelerator bunch crossing clock

**Pro's:** dead-time free (just few clock cycles to protect buffers)

**Con's:** cost (high frequency stable electronics, sometimes needs to be custom made); maintain synchronicity throughout the entire system, complicated alignment procedures if the system is large (software, hardware, human...)



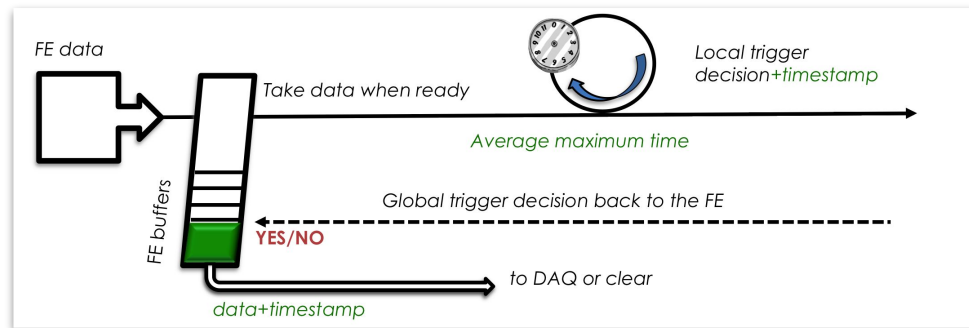
# Synchronous or asynchronous?

**Asynchronous:** operations start at certain conditions (e.g. data is ready)

- Used for **larger time windows**
- Latency known on average (with large buffers to absorb fluctuations)
- If buffer size  $\neq$  dead-time  $\rightarrow$  lost events
- Used also for “software filters”

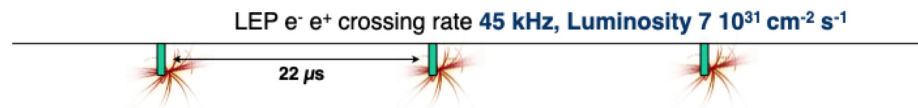
**Pro's:** more resilient to data burst; running on conventional CPUs

**Con's:** needs a timing signal synchronised to the front ends to latch the data, needs time-marker stored in the data, data transfer protocol is more complex)



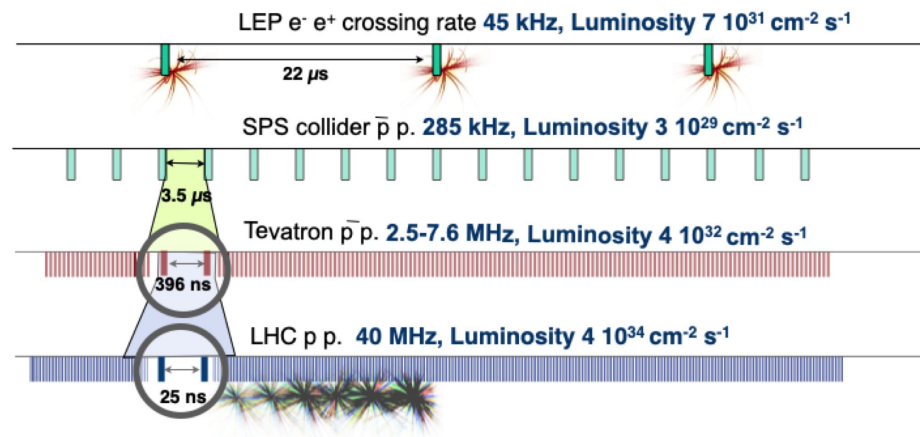
# Timescales

- **At LEP, bunch crossing interval  $22\ \mu\text{s}$ :**  
complex trigger processing was possible  
between BXs



# Timescales

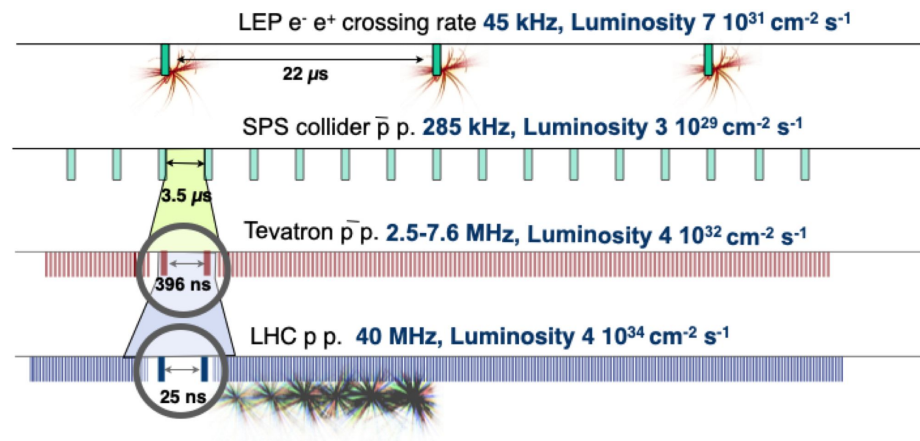
- **At LEP, bunch crossing interval  $22\ \mu\text{s}$ :**  
complex trigger processing was possible  
between BXs
- **Modern colliders chasing statistics**
  - High Luminosity by high rate of BX
  - BX spacing too short for final trigger decision!
  - No mechanism to throttle data







# Timescales


- At LEP, bunch crossing interval  $22\ \mu\text{s}$ : complex trigger processing was possible between BXs
- Modern colliders chasing statistics
  - High Luminosity by high rate of BX
  - BX spacing too short for final trigger decision!
  - No mechanism to throttle data
- Need to **pipeline** our logic!



# Pipelined processing

	19:00	20:00	21:00	22:00	23:00	00:00	01:00	02:00	03:00	04:00
										
										
										
										

# Pipelined processing

	19:00	20:00	21:00	22:00	23:00	00:00	01:00	02:00	03:00	04:00
										
										
										
										

# Pipelined processing

	19:00	20:00	21:00	22:00	23:00	00:00	01:00	02:00	03:00	04:00
										
			<i>Clearly not optimal!</i>							
										
										
										

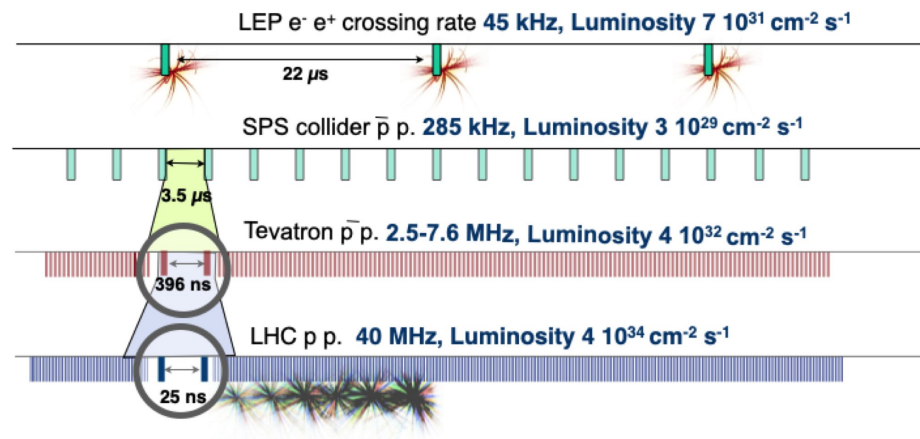
# Pipelined processing

	19:00	20:00	21:00	22:00	23:00	00:00	01:00	02:00	03:00	04:00
										
										
										
										



# Timescales

- At 40 MHz BX rate, a 4 GHz CPU could perform 100 CPU operations (not enough to be useful) before having to pass to the next core
- **What can we use instead?**

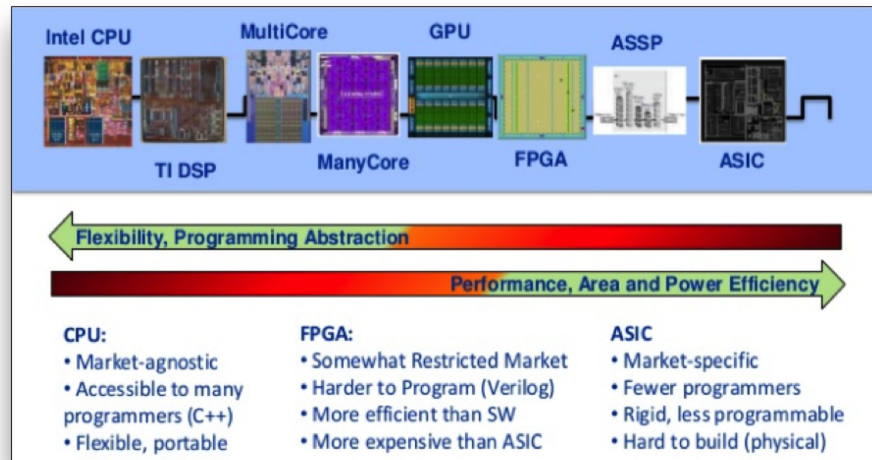


# Programmable devices

## Application-specific integrated circuits (ASICs):

optimised for fast processing,  
design encoded into silicon

## Field-programmable gate arrays (FPGAs): “Programmable ASICs”



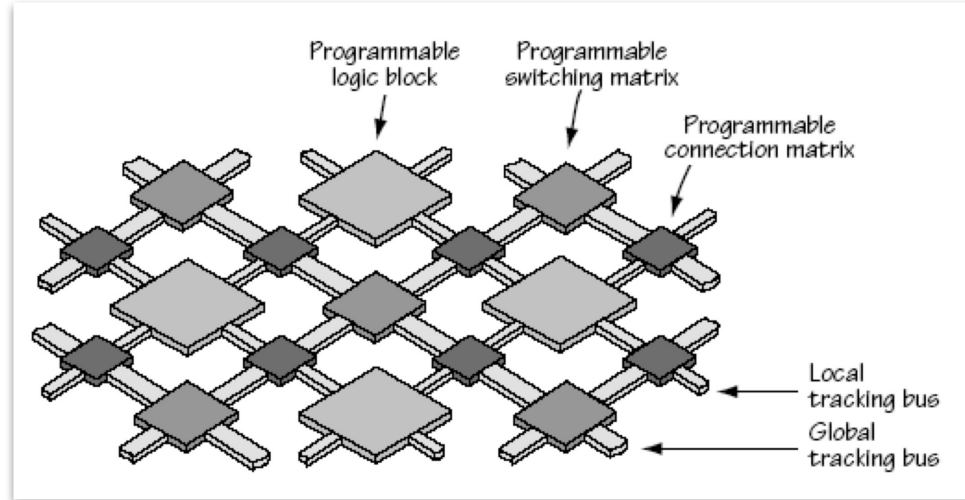


# Field-programmable gate arrays

- Programmable Logic Blocks
- Massive Fabric of Programmable Interconnects

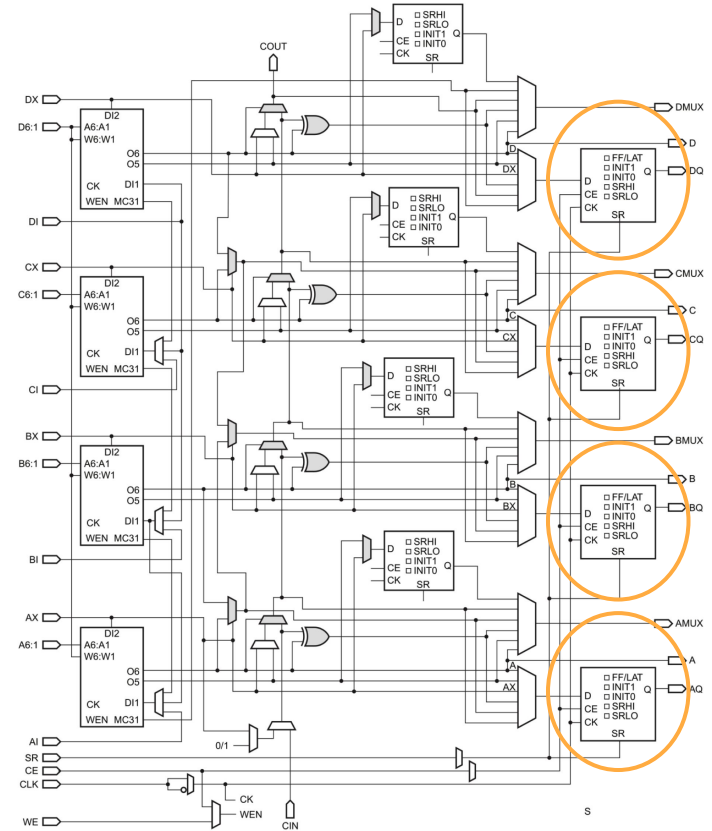
See:

[Introduction to FPGAs](#), Hannes Sakulin  
*Advanced FPGA programming*, Manoel Barros Marin

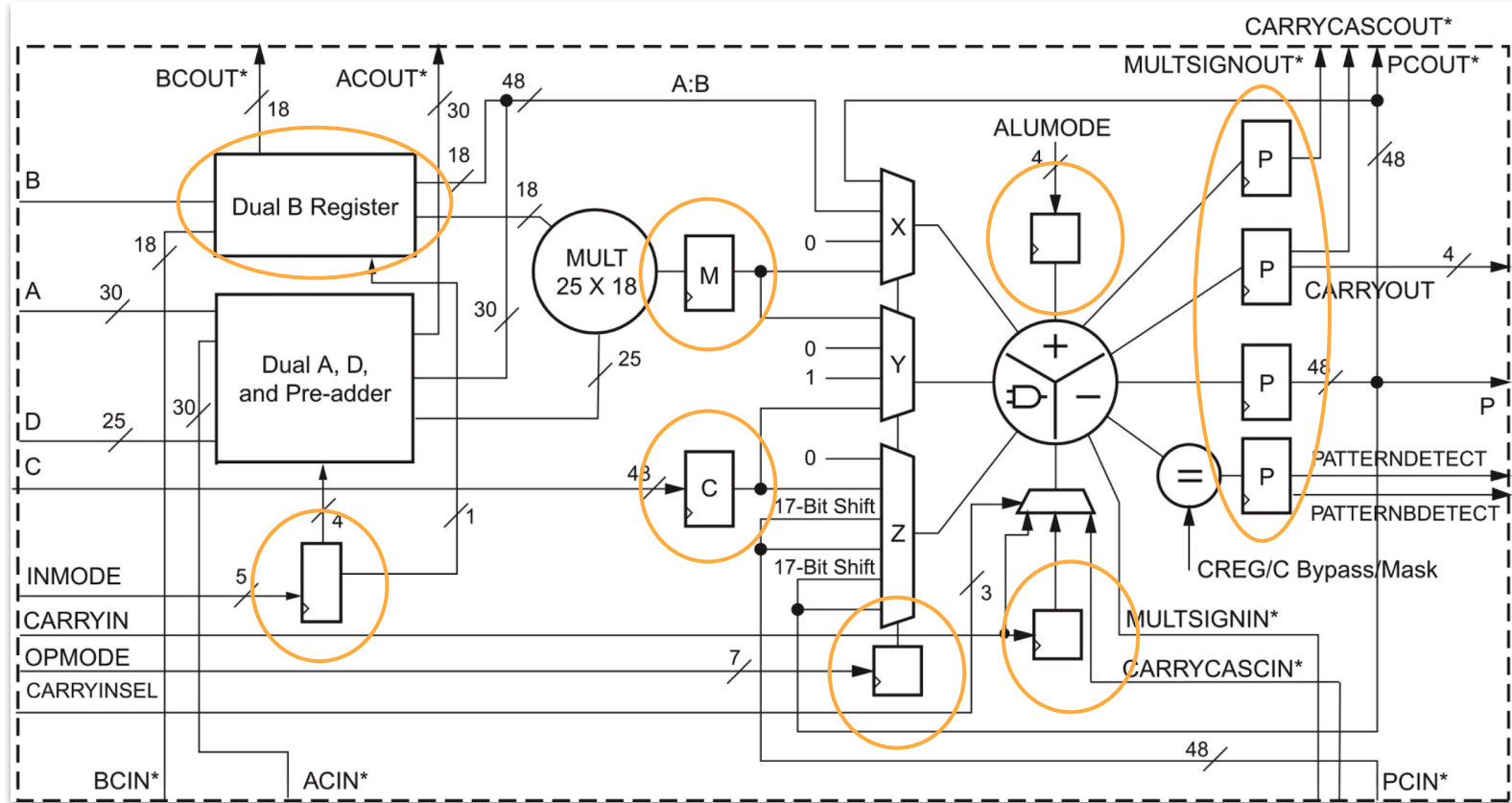


# Configurable logic blocks

- Registers at output of every cell
  - Perfect for pipelined logic!



# Digital signal processing slices



# Example: Xilinx Ultrascale+ FPGAs

- Upwards of 2 million logic cells
- All can be clocked at up to 500 MHz
- Up to  $O(10^{15})$  operations per second
- Upwards of 6000 DSPs
- All pipelined
- Fully programmable
- All good?

Device Name	VU9P	VU11P	VU13P
Effective LEs <sup>(1)</sup> (K)	2,485	2,575	3,435
Logic Cells (K)	2,069	2,147	2,863
CLB Flip-Flops (K)	2,364	2,454	3,272
CLB LUTs (K)	1,182	1,227	1,636
Max. Distributed RAM (Mb)	36.1	34.8	46.4
Total Block RAM (Mb)	75.9	70.9	94.5
UltraRAM (Mb)	270.0	324.0	432.0
DSP Slices	6,840	8,928	11,904
PCIe® Gen3 x16 / Gen4 x8	6	3	4
150G Interlaken	9	9	12
100G Ethernet w/ RS-FEC	9	6	8
Max. Single-Ended HP I/Os	832	624	832
GTY 32.75Gb/s Transceivers	120	96	128

# Caveats

---

- Very hard to program efficiently
  - Thinking in a parallel, pipelined-fashion is not intuitive for most people
  - A handful of real experts in CMS
- Efficient use depends on efficiently structured data
- The chip is just the start – needs to be attached to something
- You are also responsible for the infrastructure

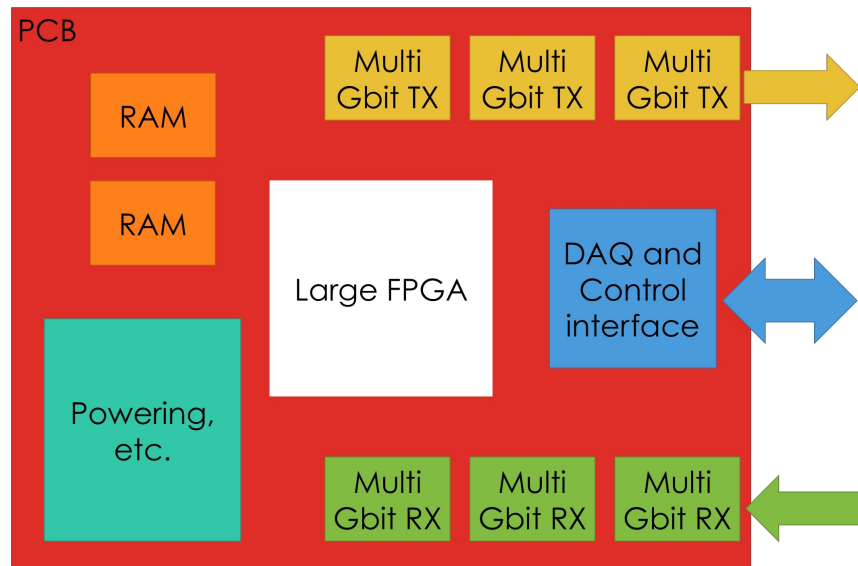
# High-level synthesis

---

- High-level synthesis (HLS) allows you to write your algorithm in (a version of) C++ and "compiles" this to gateware
  - Very complex algorithms now possible in FPGAs, e.g. machine learning
  - Prototyping of ideas significantly easier
- Still requires some knowledge of the underlying chip resources
  - Asking an FPGA to do lots of floating point operations will not get you far
- First production-ready systems appearing in e.g., the CMS L1 trigger
  - Barrel track finder using a Kalman filter for track reconstruction

# Generic hardware

- Once outside interfaces are fixed, all FPGA boards "look the same"
  - Data streamed in
  - Data processed in large FPGA
  - Data streamed out
- **Reusable infrastructure firmware**
  - Virtually all boards for upcoming LHC upgrades not provided "empty" anymore
  - Huge advantage as it allows board users to concentrate on trigger algos
    - And debugging of the infrastructure benefits all...

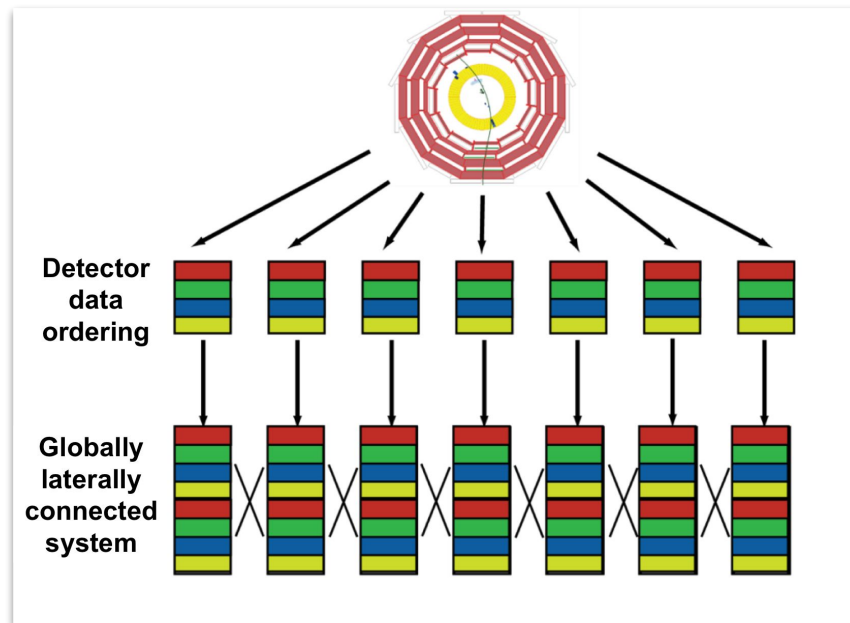


# Trigger system architectures



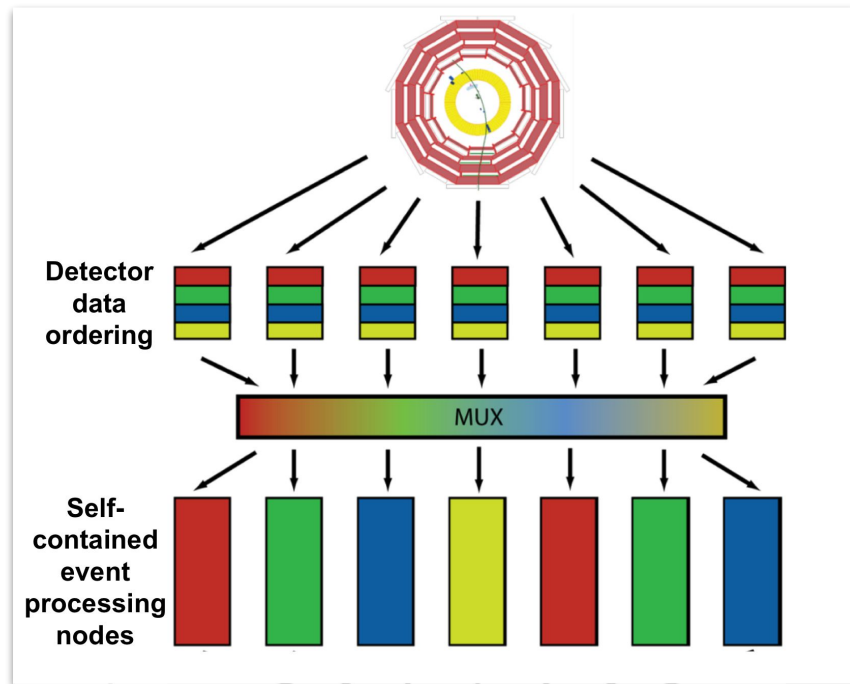
# Conventional architecture

- Each subsystem is regionally segmented
- Each region must talk to its neighbour or data must be duplicated
  - To avoid inefficiency at the borders
- Each region of each processing layer compresses, suppresses, summarizes or otherwise reduces its data and passes it on to the next level which is less regionally segmented



# Time-multiplexed architecture

- Buffer data and stream it out optimized for processing
- Spread processing over time
  - Stream-processing rather than combinatorial-logic
  - Maximise reuse of logic resources
  - Easiest for FPGA design tools to route and meet timing
- Costs you latency, bought back by more efficient processing



# High-level trigger architecture

	Levels	L1 rate	Event size	Readout bandwidth	HLT rate
LEP	2/3	1 kHz	100 kB	few 100 kB/s	~5 Hz
ATLAS	2/3	100 kHz (L2: 10 kHz)	1.5 MB	30 GB/s (incremental event building)	~1 kHz
CMS	2	100 kHz	1.5 MB	100 GB/s	~1 kHz

**LEP:** 40 Mbyte/s, so VME bus sufficient for bandwidth needs

**LHC:** cutting-edge processors, high-speed network interfaces, high speed optical links

Different approaches possible

- Network-based event building (LHC example: CMS)
- Seeded reconstruction (LHC example: ATLAS)

# High-level trigger design principles

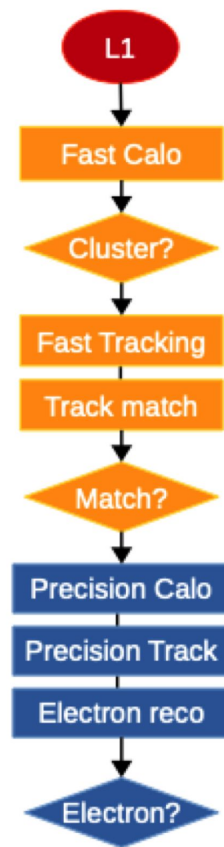
Offline reconstruction too slow to be used directly

- Takes  $>10$ s per event but HLT usually needs  $\ll 1$ s

**Instead:** Step-wise processing with early rejection

1. Fast reconstruction & L1-guided regional reconstruction
  - Trigger-specific or special configurations of offline algorithms
  - L1-guided regional reconstruction
2. Precision reconstruction as full detector data becomes available
  - Offline (or very close to) algorithms
  - Full detector data available

**Stop processing as soon as one step fails!**



# High-level trigger design principles

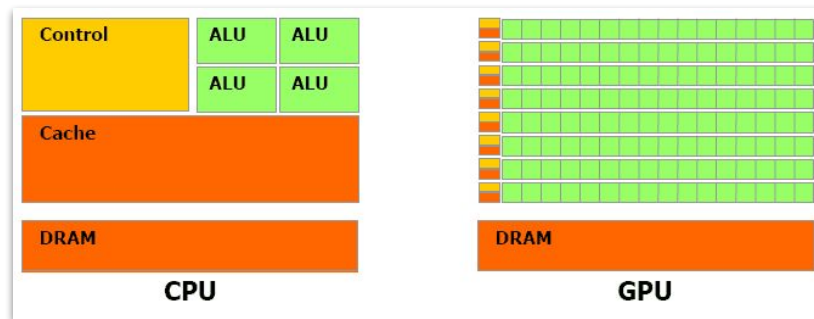
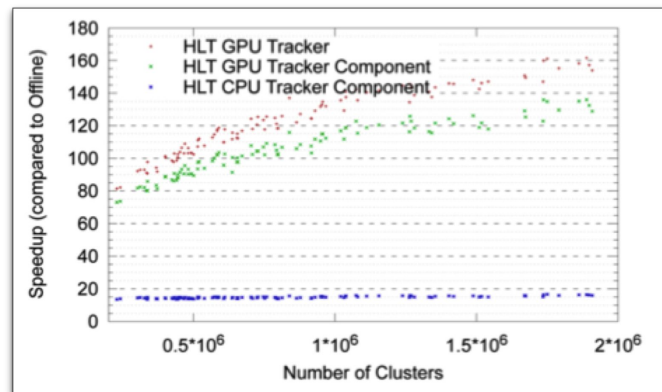
Early rejection reduce data and resources (CPU, memory, etc.)

## Event-level parallelism

- Process more events in parallel
- Multi-processing or/and multi-threading

## Algorithm-level parallelism

- GPUs effective whenever large amount of data can be processed concurrently (although bandwidth can be a limiting factor)



# Conclusions

---

- Triggers have been around for some time
  - ... but they are constantly evolving
  - Keeping up-to-date on developments in industry is mandatory!
- FPGAs are the key tool in the Level-1 trigger
  - Lots of challenges even with modular firmware and the help of HLS
- Working on the trigger can be fun and filled with learning opportunities!
  - Electronics, networking, (heterogeneous) computing, system design, ...