Intro to Triggering and Data Acquisition

Samuel Silverstein, Stockholm University

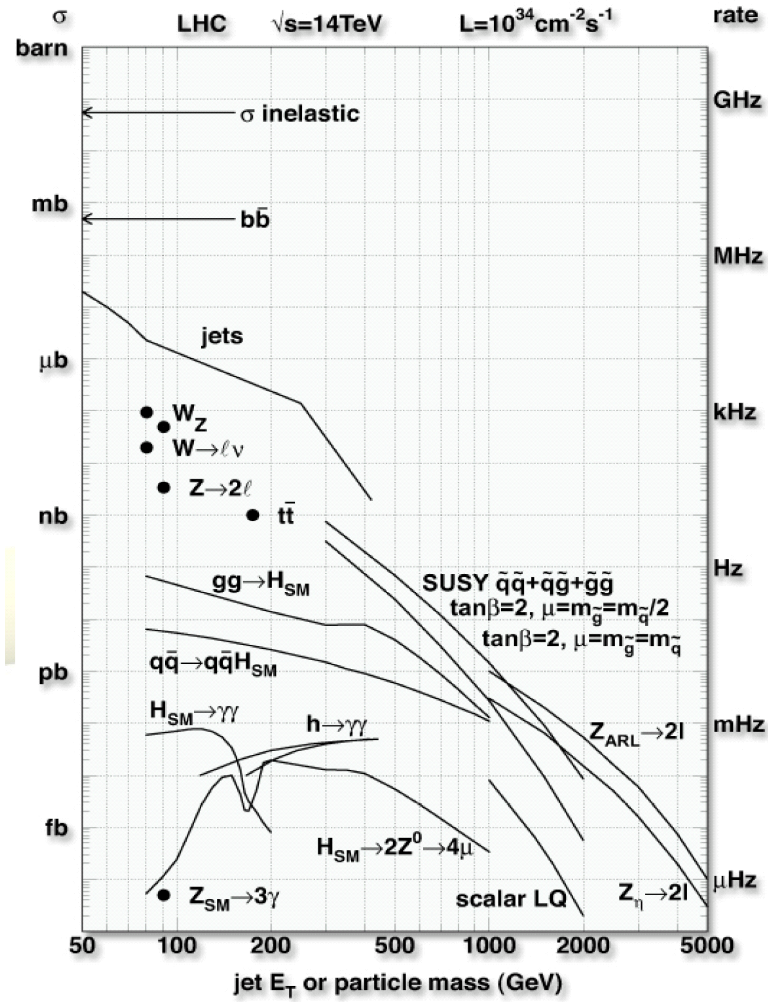
- Trigger/DAQ basics
- Collider TDAQ
- Signal processing
- Dead time

In this trigger/DAQ series:

- Three segments
  - Trigger/DAQ introduction, concepts
  - TDAQ architectures
  - New developments
- Exercises
  - Programmable logic (lab)
  - Dead-time simulation (homework)

Introduction

Colliders produce a lot of data!



- Physics at LHC:
  1 interesting event in ~10^6-10^{13}
- Need a high collision rate:
  - 40 MHz bunch crossing rate
  - 25-50 proton collisions per bunch crossing
- Collision data volume
  - About 1.5 MB/event (ATLAS)
  - ~60 TB/s at 40 MHz!
- <u>Too much data!</u>

Why is it too much?

(In reverse order…)

- Can't analyse it all…
  - Would need a million times more processing
- Can't store it all…
  - 60 TB/s is about 3.6 petabytes/minute
- Can't get it all off the detector*
  - High-bandwidth data links are expensive, take up space and consume power.
  - Practical consequences:
    - Heat dissipation
    - Cables, power, cooling take up space, leaving dead material and "holes" in detector coverage

*N.B. New link technologies making this less true today for *some* systems

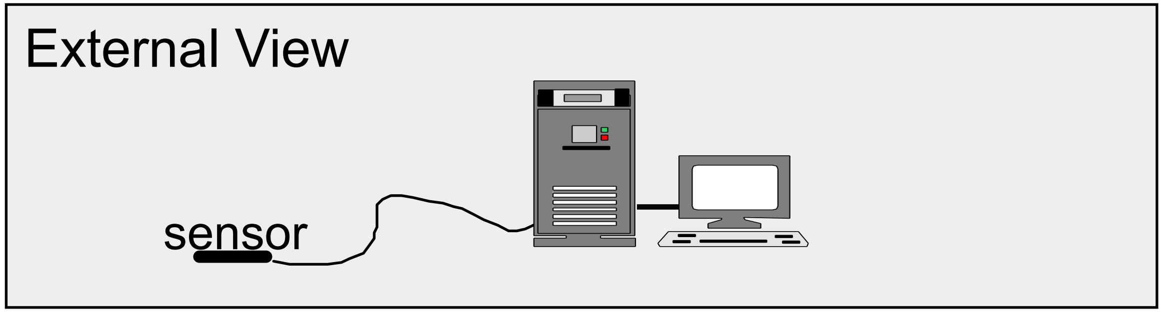
So, the challenge is:

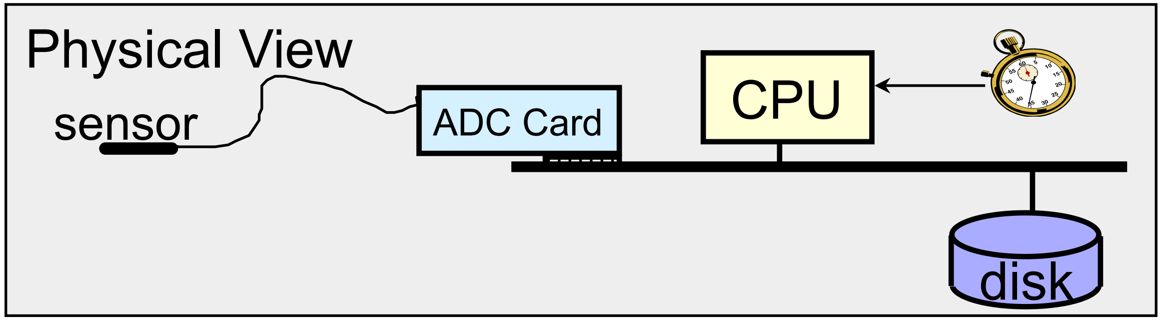
- Throw away 99.999% of the collision data
  - To keep data rates at manageable levels
- But don't throw away:
  - Interesting SM processes
  - New physics predicted by BSM theories
  - Unexpected evidence of new physics
- Doing all of this well is hard!
  - And perfection is practically impossible

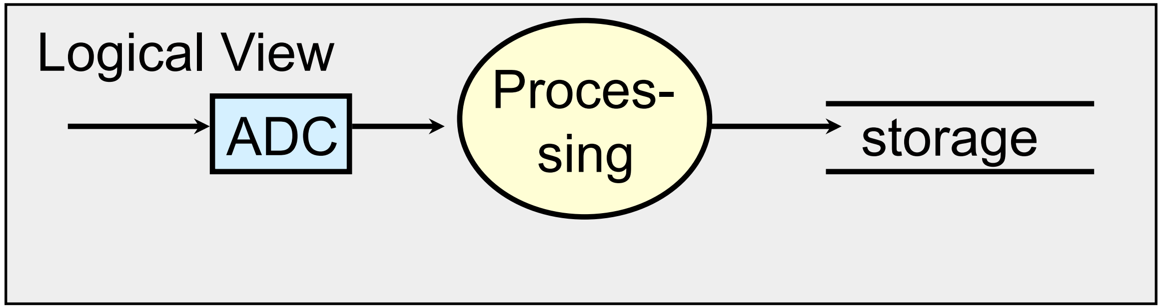
Data acquisition must compromise between physics goals and what is technically achievable!

Data Acquisition Basics

Trivial DAQ example







Trivial DAQ

- How it works:
  - Sensor produces an analog signal
  - ADC periodically converts analog output to digital values
  - CPU reads digital values from the ADC and writes them to disk (readout)
- Problem:
  - If readout rate is much larger than physics rate: lots of uninteresting data to store and analyze.
- Solution:
  - Initiate readout only if there is an interesting signal (trigger)

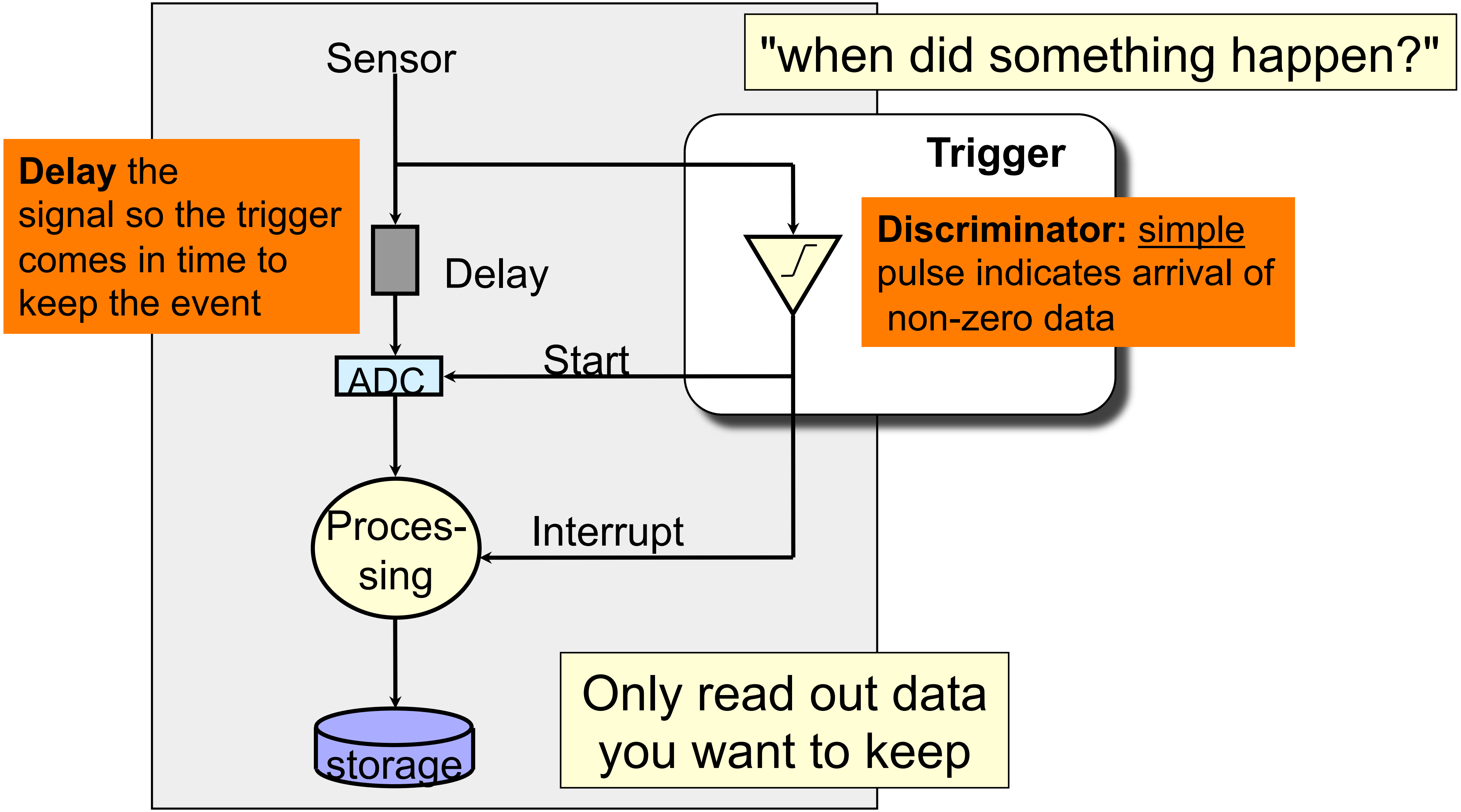
What is a trigger?

Wikipedia: "**A system that uses simple criteria to rapidly decide which events in a particle detector to keep when only a small fraction of the total can be recorded.** "

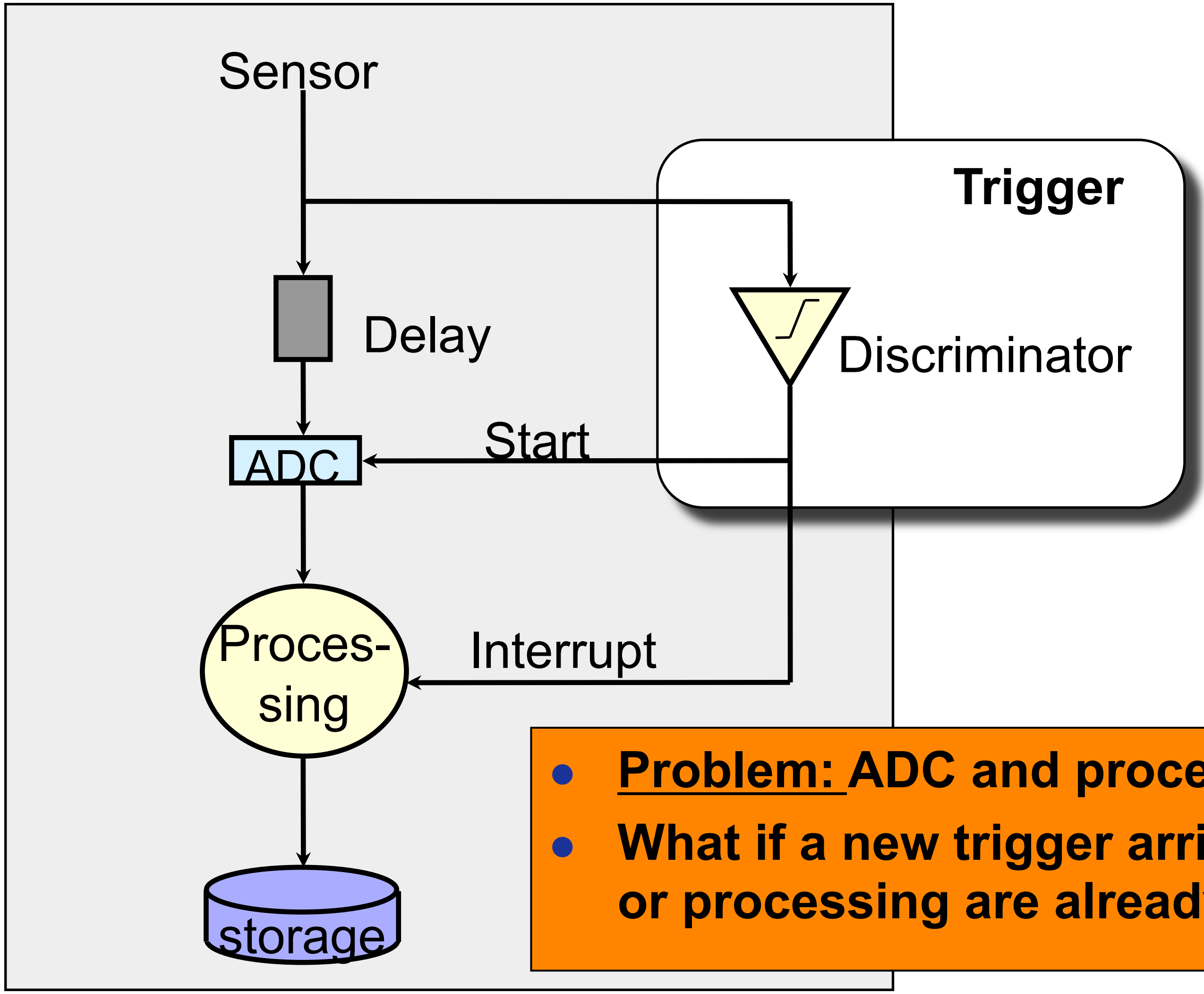
What is a trigger?

- Simple
  - Just need to decide whether to keep the data. Detailed analysis comes later
- Rapid
  - Data you want to keep may be lost if you wait too long to decide
- Selective
  - Need to achieve a sufficient reduction in readout rate, because…
  - “Only a small fraction can be recorded”

Trivial DAQ with trigger



Trivial DAQ with trigger

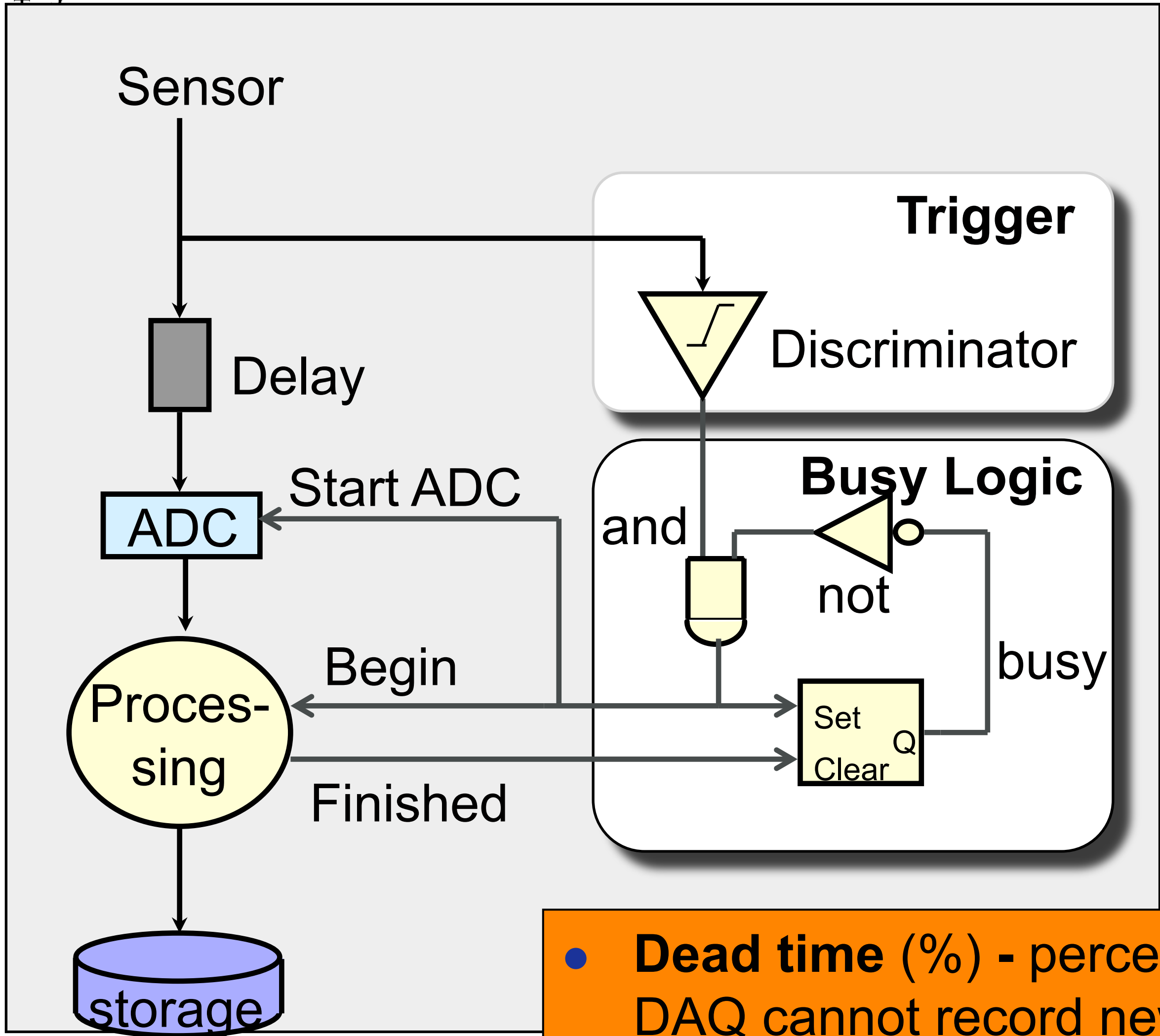


- **Problem: ADC and processing take time!**
- **What if a new trigger arrives when ADC or processing are already busy?**

Flow control

- New triggers can interfere with the readout if a previous event is still being processed
- Solution: flow control
  - ADC/readout send ‘Busy’ signal to temporarily disable the trigger
  - Events occurring during ‘Busy’ period are not recorded (dead time).

DAQ with trigger & busy logic



Busy time:
ADC readout +
processing time

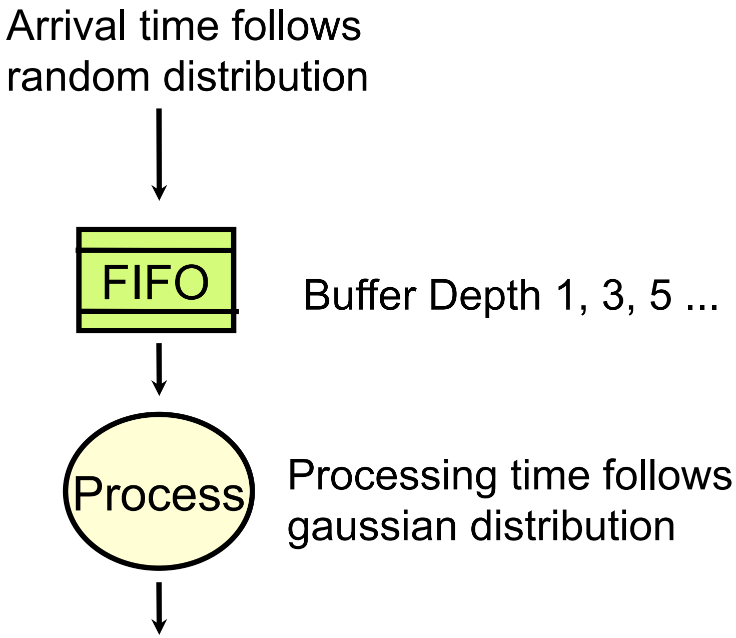
- **Dead time** (%) **-** percentage of time that DAQ cannot record new events

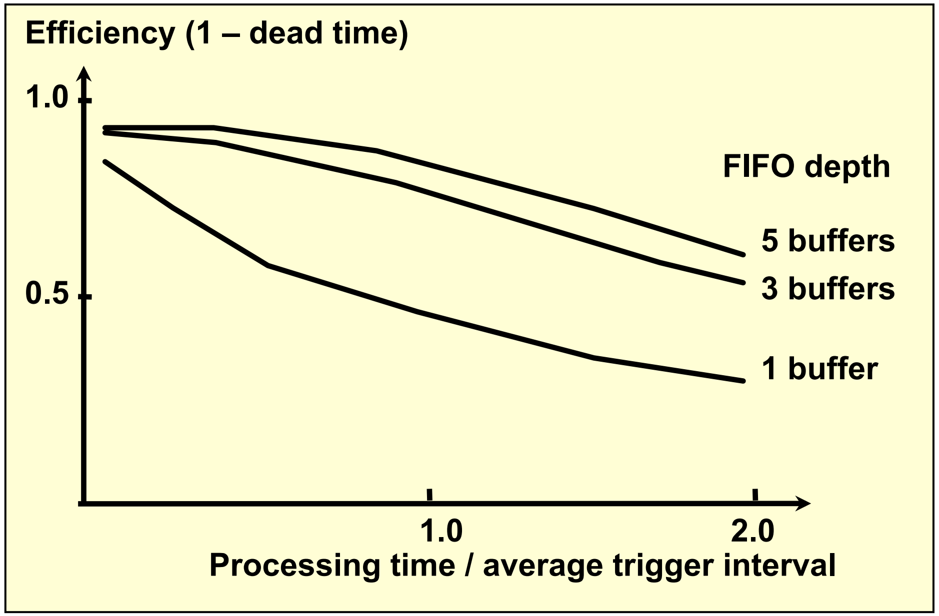
Dead time

- Problem:
  - Triggers occur at random intervals
  - But events take about the same time to read out (Gaussian distribution)
- As trigger rate approaches readout rate, dead time increases
  - And with it, loss of data!
- Solution:
  - Add a derandomising buffer (FIFO)

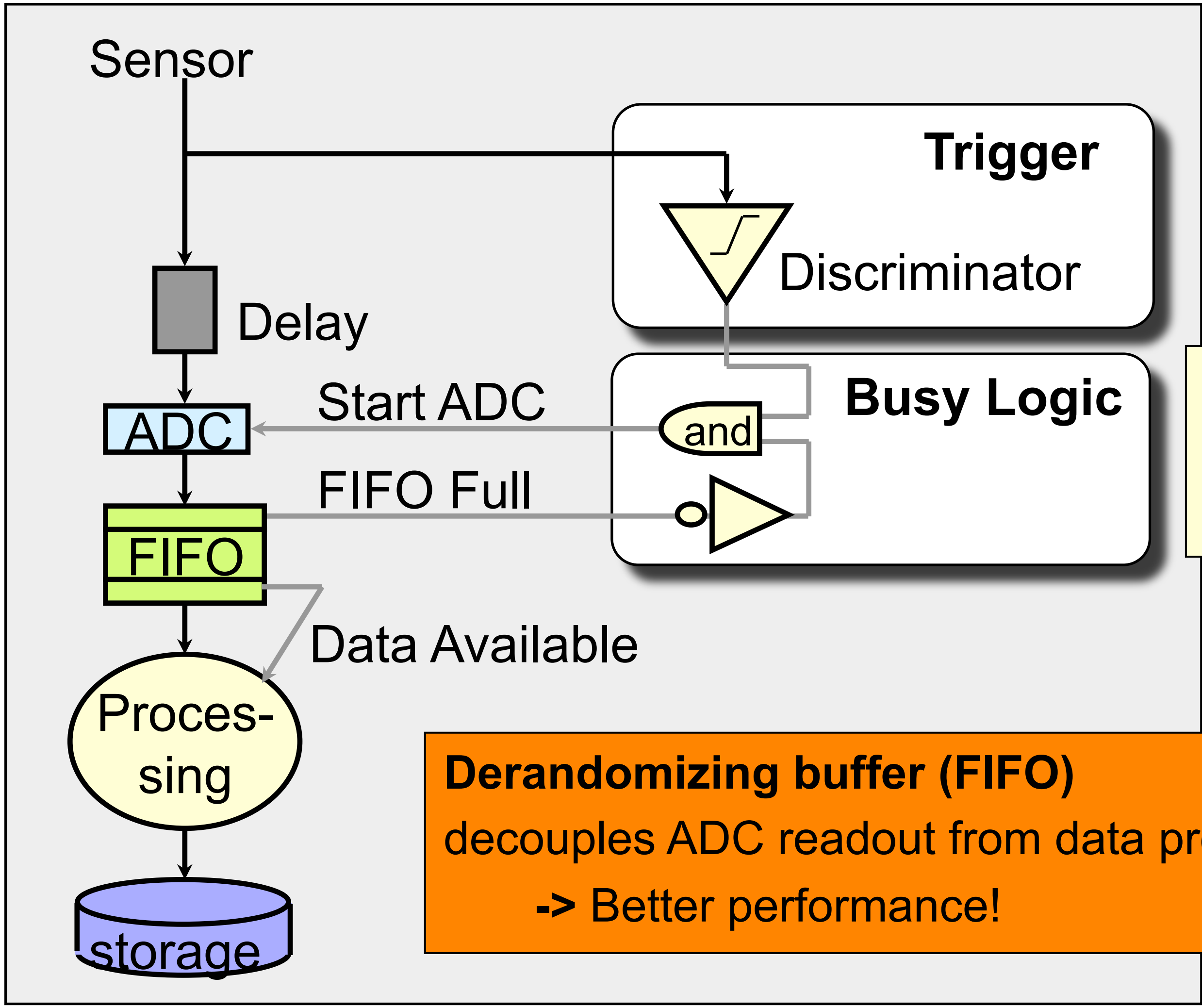
Buffers reduce dead time

- Buffers (FIFO) receive new events as they arrive, send them out when the processor is ready
  - "average out" arrival rate of new events
- Dead time depends on trigger rate, readout time, processing time per event, and FIFO depth





Buffered DAQ



Busy time = ADC

(as long as buffer is not full)

Derandomizing buffer (FIFO)

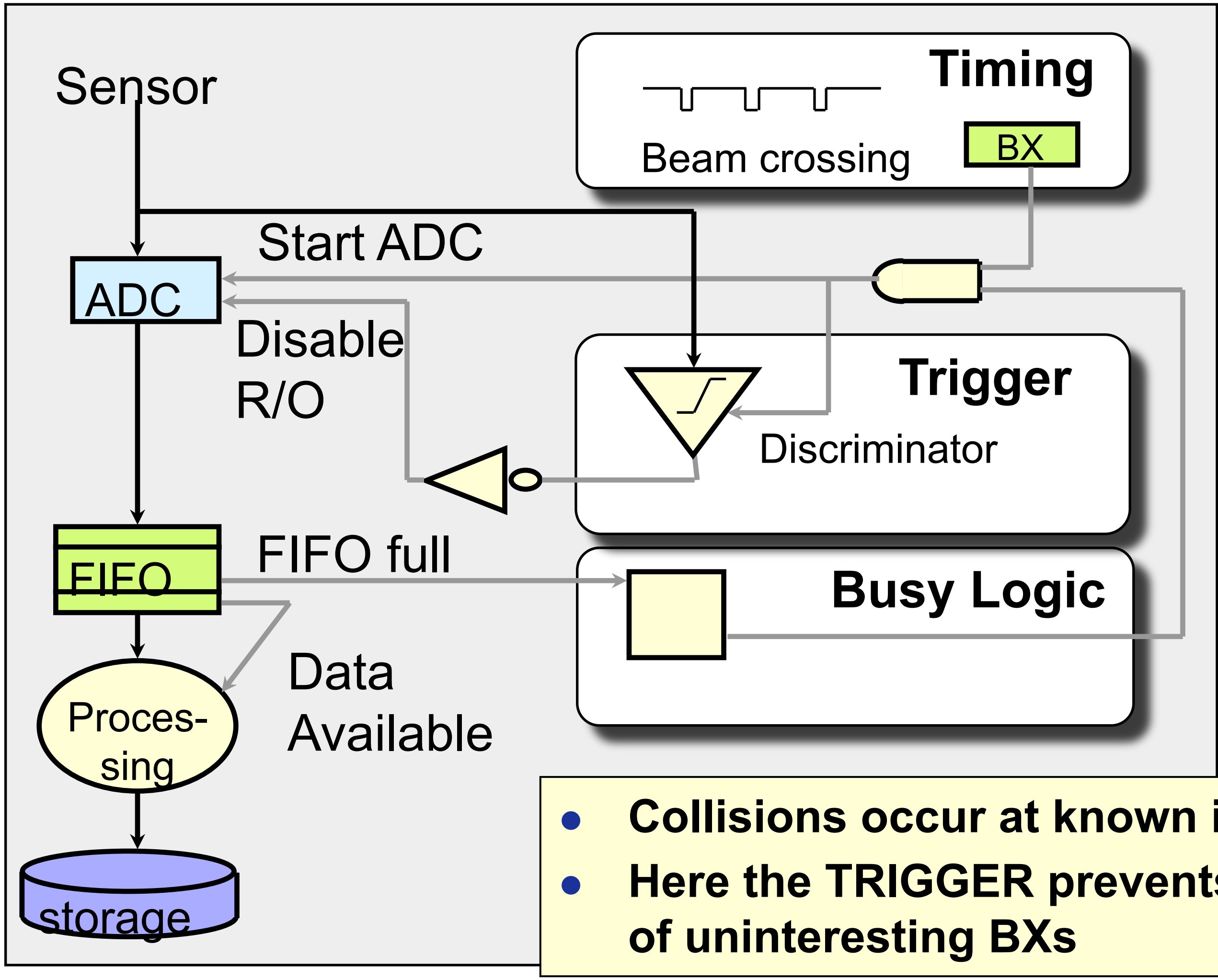
decouples ADC readout from data processing

-> Better performance!

Triggering on bunch crossings

- Particle colliders typically produce collisions at a known bunch crossing (BX) interval
- Trigger system must analyse data from every BX
  - Produce a yes/no trigger decision
- Read out data to DAQ only for selected BXs
  - Data from other BXs are effectively "thrown away".

Trivial collider-type trigger



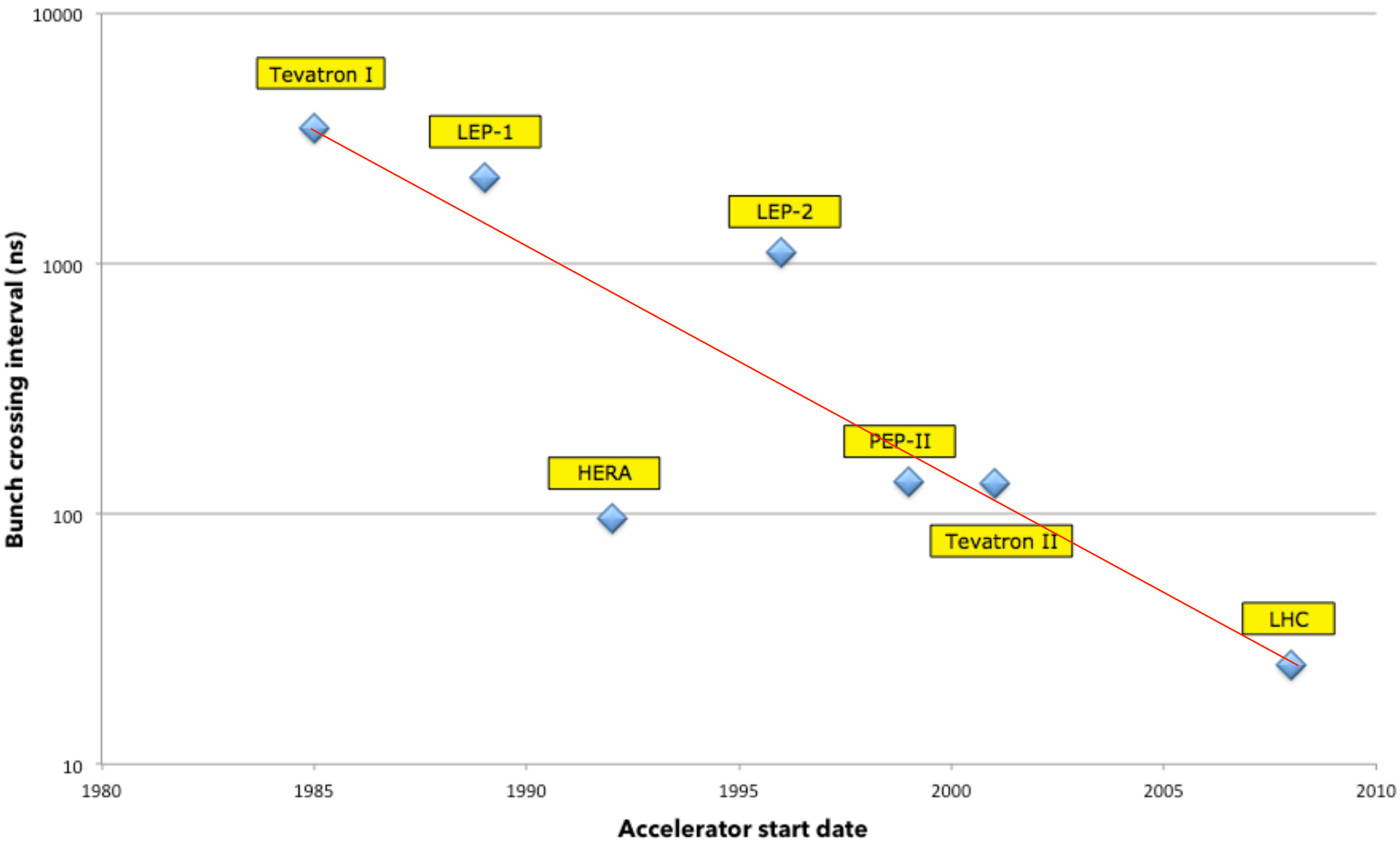
Trigger must look at <u>every</u> BX interval

- **Collisions occur at known intervals (BX).**
- **Here the TRIGGER prevents readout of uninteresting BXs**

Multi-level triggering

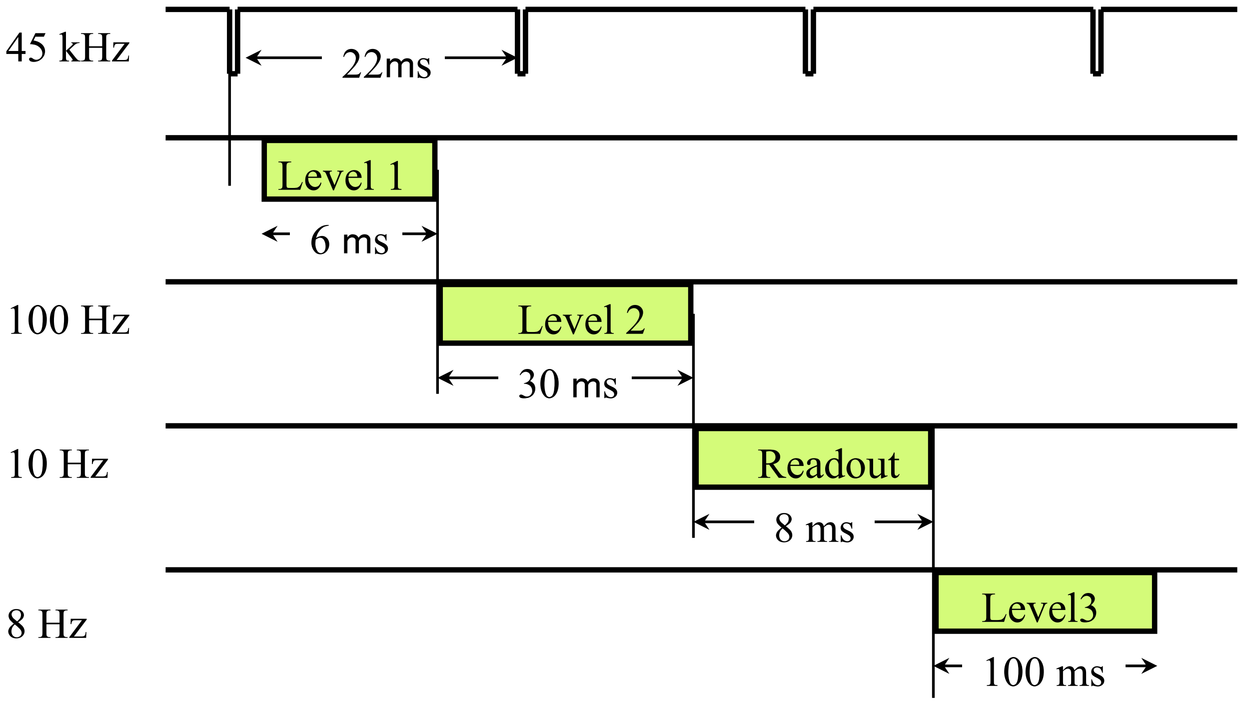
- Collider experiments need to reject most of the events (~99.999% at LHC)
  - But keep the valuable physics!
- Cannot achieve this with a single trigger.
  - Selective algorithms need full detector data
  - Too complex to analyze 40M BXs/second
- Solution: Multi-level triggers
  - Start with simple algorithms, high rates
    - Typically implemented in custom hardware
  - Perform more detailed algorithms on (fewer) events that pass the previous levels
    - Implemented in CPUs
- How to implement a multilevel trigger depends on the BX rate…

Collider BX rate evolution



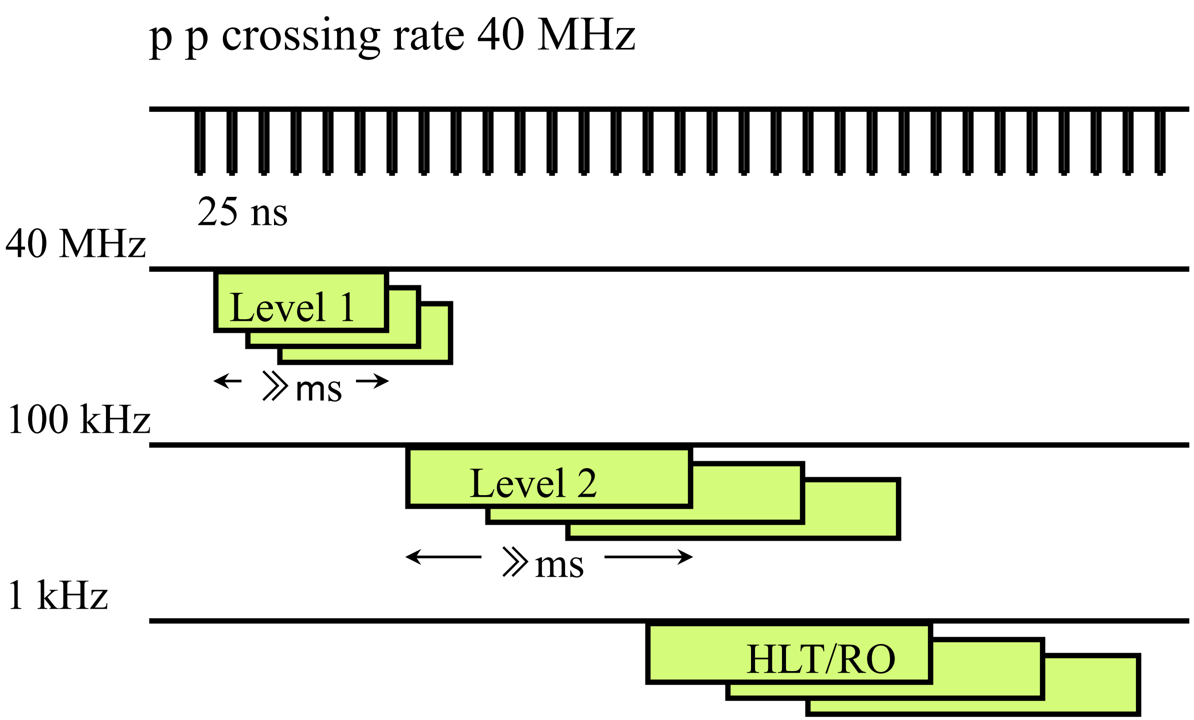
Trigger/DAQ at LEP

e^+e^- crossing rate: 45 kHz (LEP-1)



- **Level 1 trigger latency less than BX interval –> No dead time**
- **No overlapping events at any trigger level**
- **Most trigger/DAQ electronics located off-detector**

Trigger/DAQ at LHC

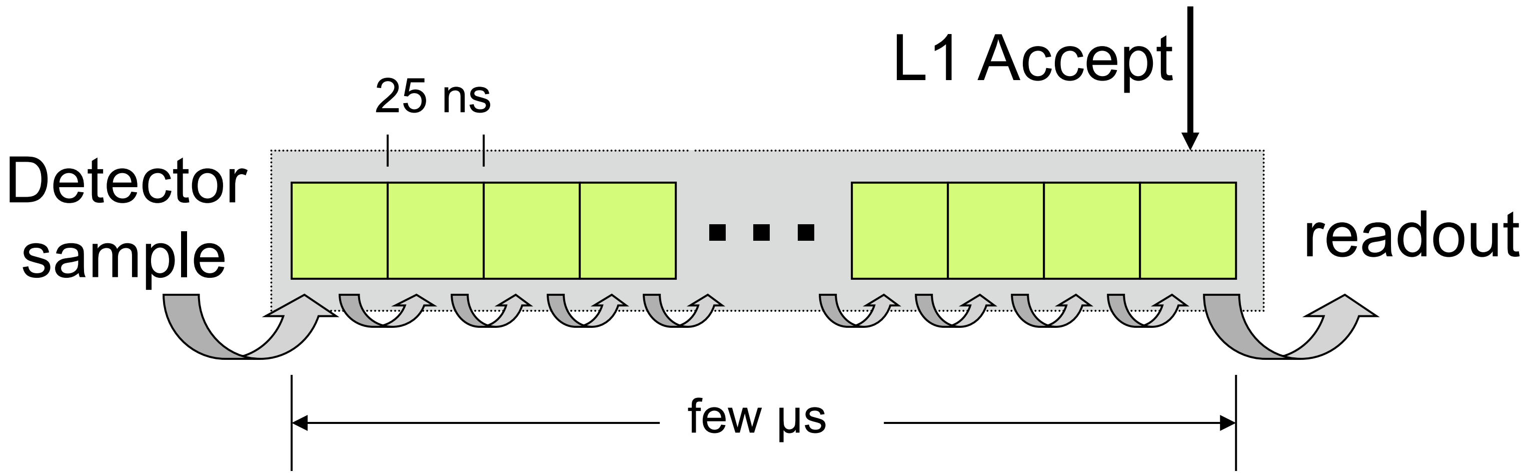


- **Level 1 trigger latency >> bunch interval**
- **Up to 10 μs for complete readout of an event**
- **Need <u>pipelined</u> trigger and DAQ**

Pipelined trigger

- For every bunch crossing:
  - Sample and store all detector data in fixed-length pipeline buffers (~few μs)
  - Send reduced-granularity data to the Level-1 trigger over a low-latency path
  - Level-1 does pipelined processing
    - Many consecutive stages
    - Produces a Level-1 accept decision (L1A) for every BX
- If an L1A is issued for an event:
  - Extract data for that event from the end of the pipeline buffers and read it out to DAQ

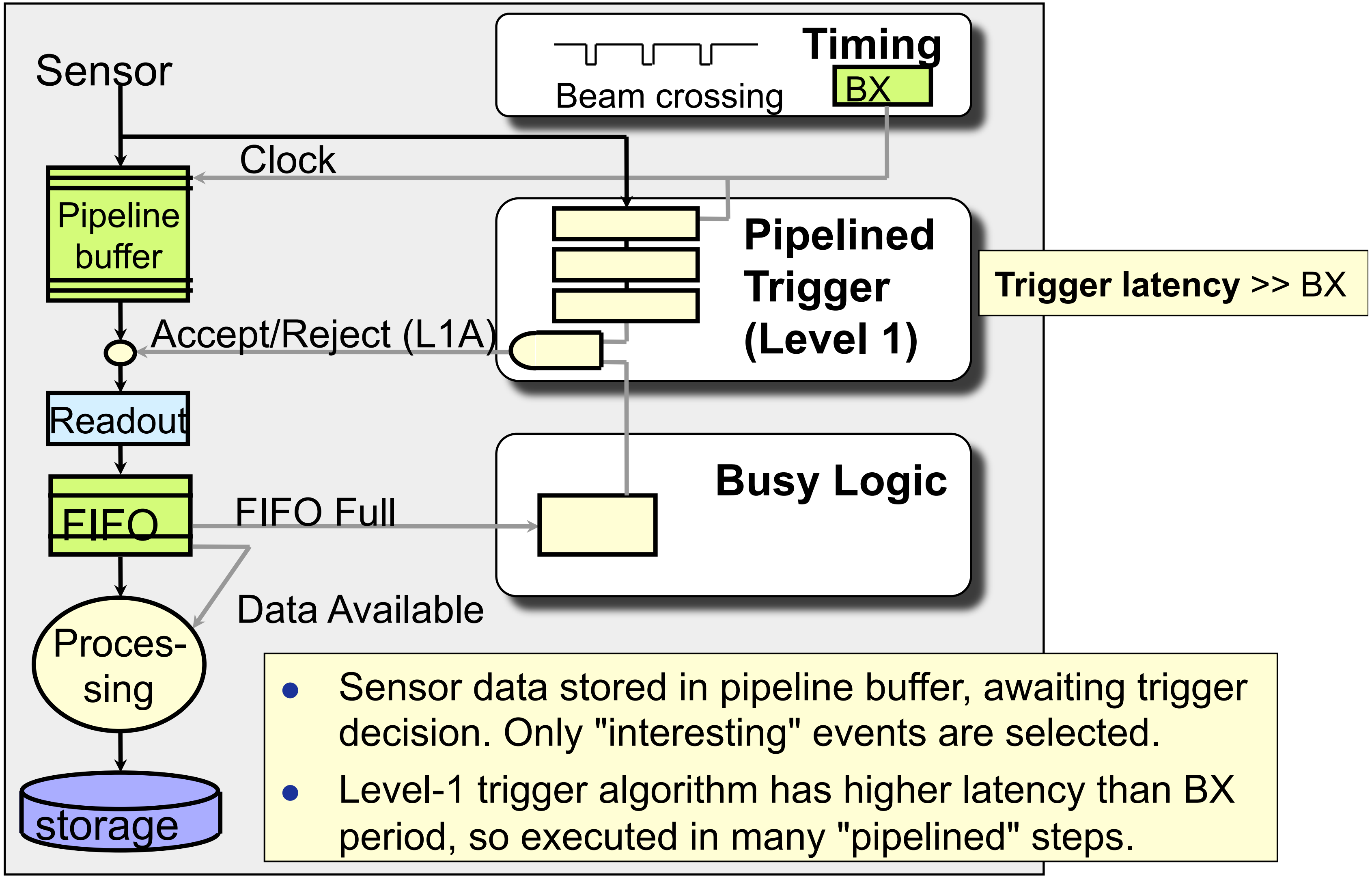
Pipeline buffer



New data enters the buffer every BX

Keep event at end of pipeline if L1A received

Simple pipelined trigger/DAQ

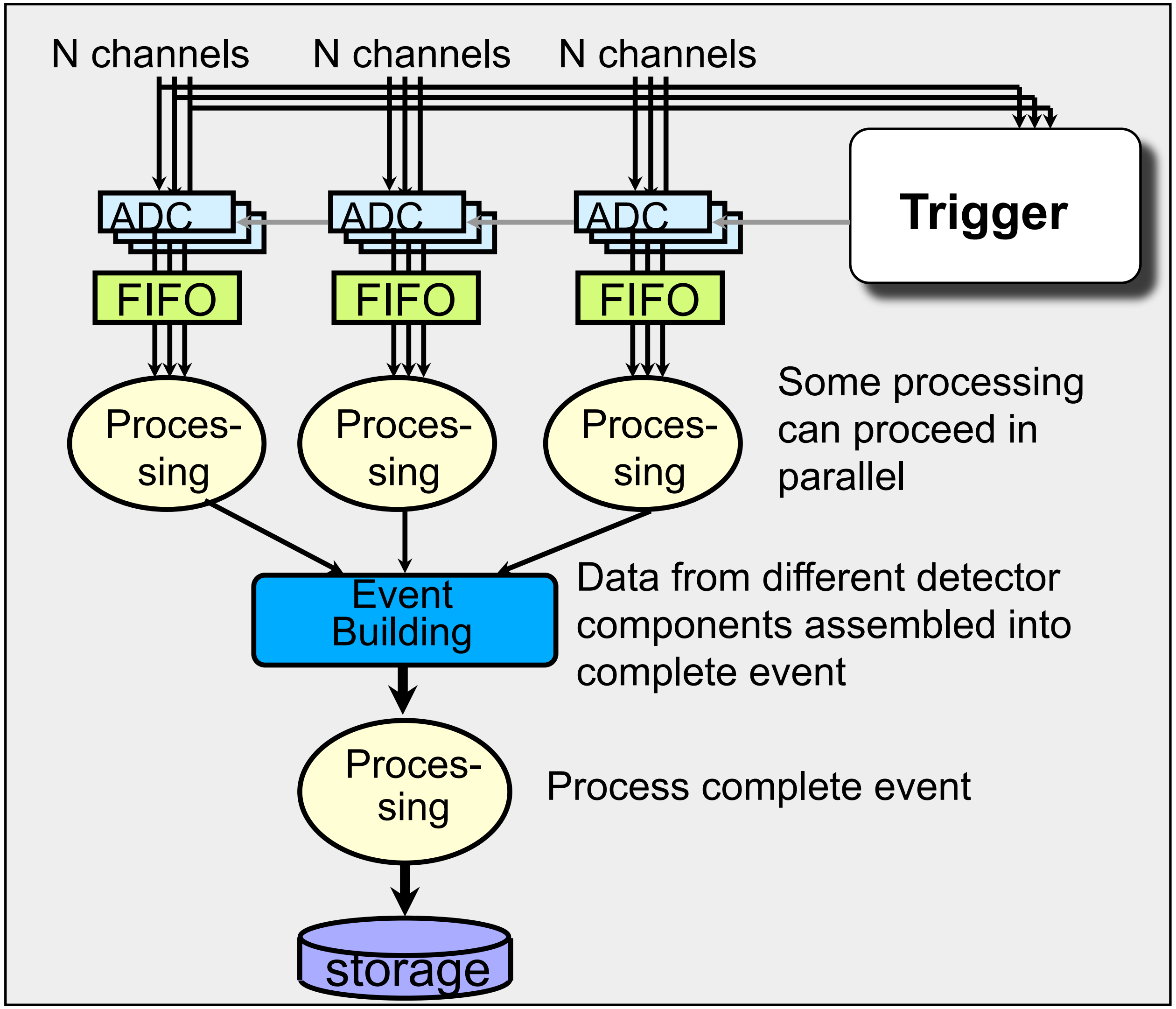


- Sensor data stored in pipeline buffer, awaiting trigger decision. Only "interesting" events are selected.
- Level-1 trigger algorithm has higher latency than BX period, so executed in many "pipelined" steps.

TDAQ for multiple systems

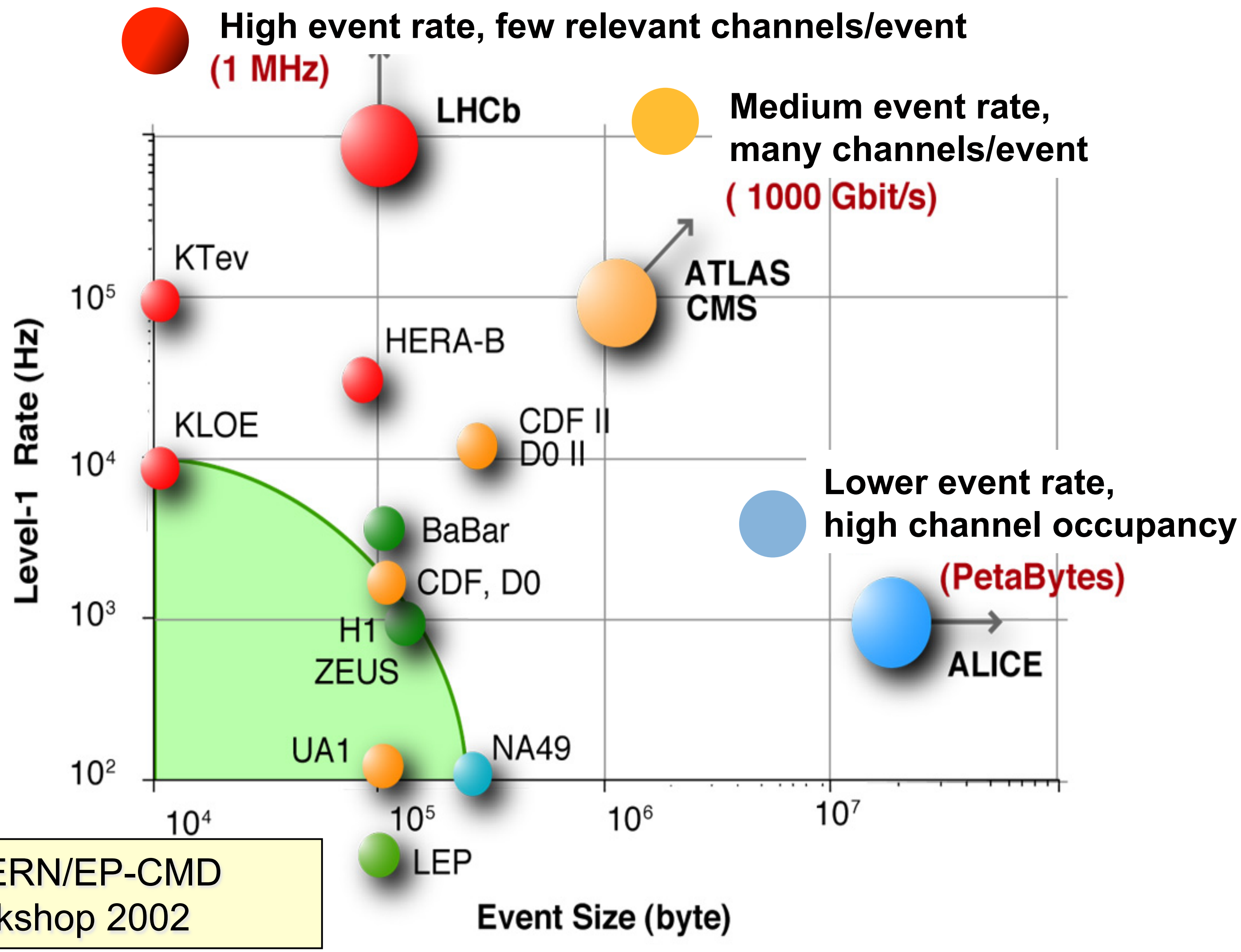
- Collider detectors are collections of detector systems working together
  - Tracking detectors
  - Calorimeters
  - Muon spectrometers
- A single Central trigger initiates readout for all detectors
- DAQ collects and records data from all detectors for each triggered event

Multi-system Trigger/DAQ



Trigger/DAQ architectures

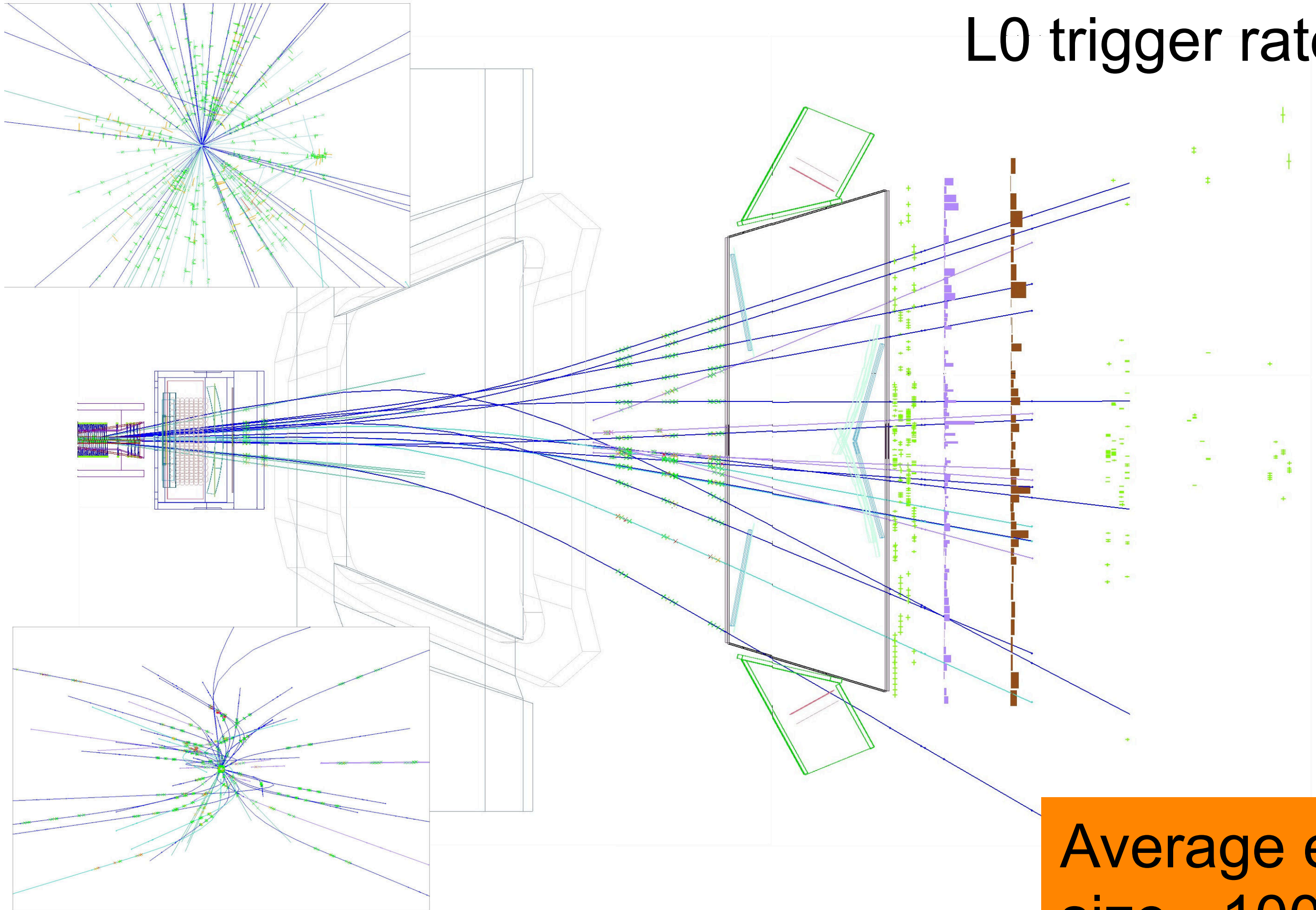
Trigger/DAQ requirements



S. Cittolin CERN/EP-CMD
LECC Workshop 2002

LHCb: High rate, small event

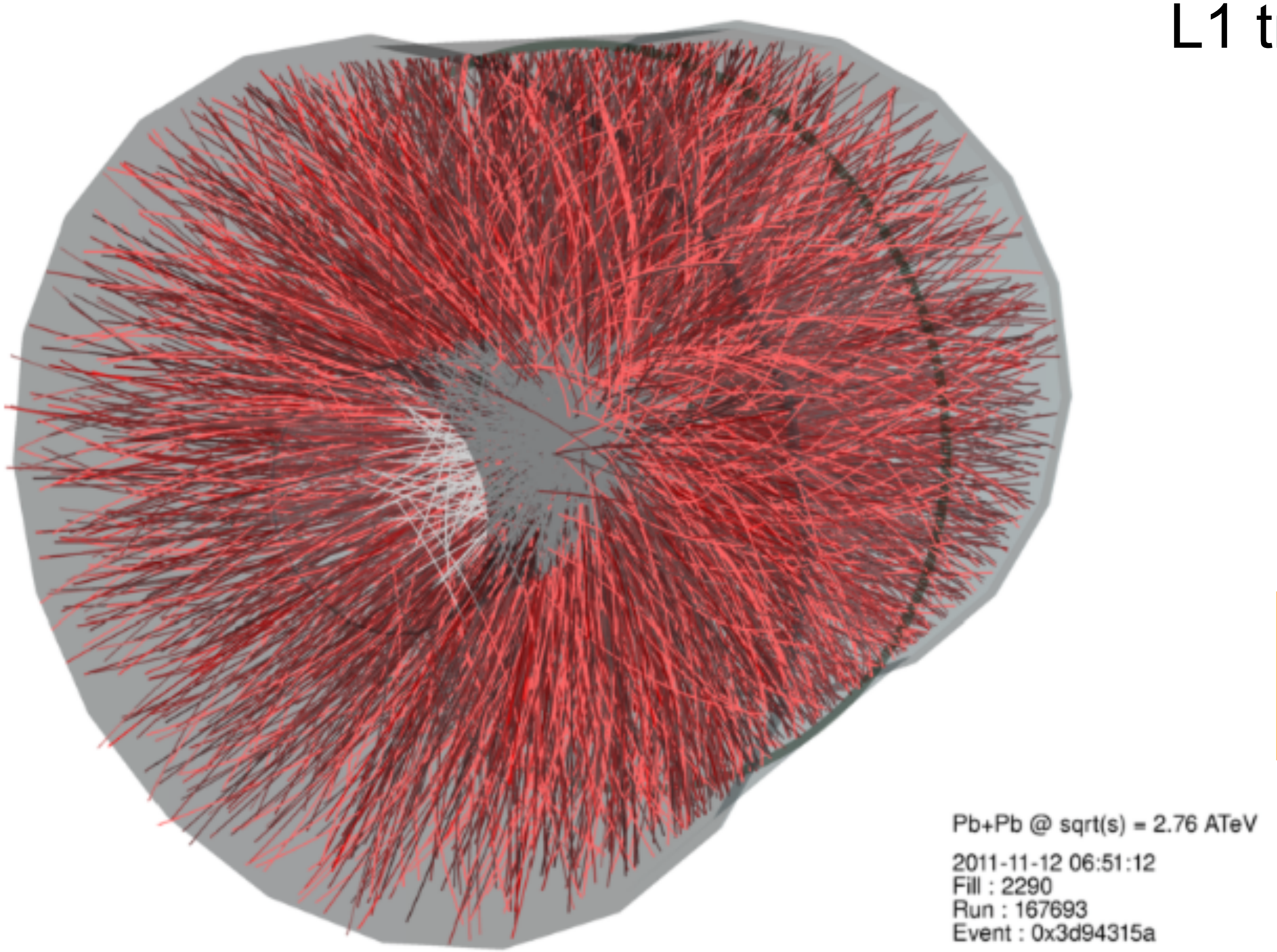
L0 trigger rate: 1 MHz



Average event size ~100kB

ALICE: Low rate, large event

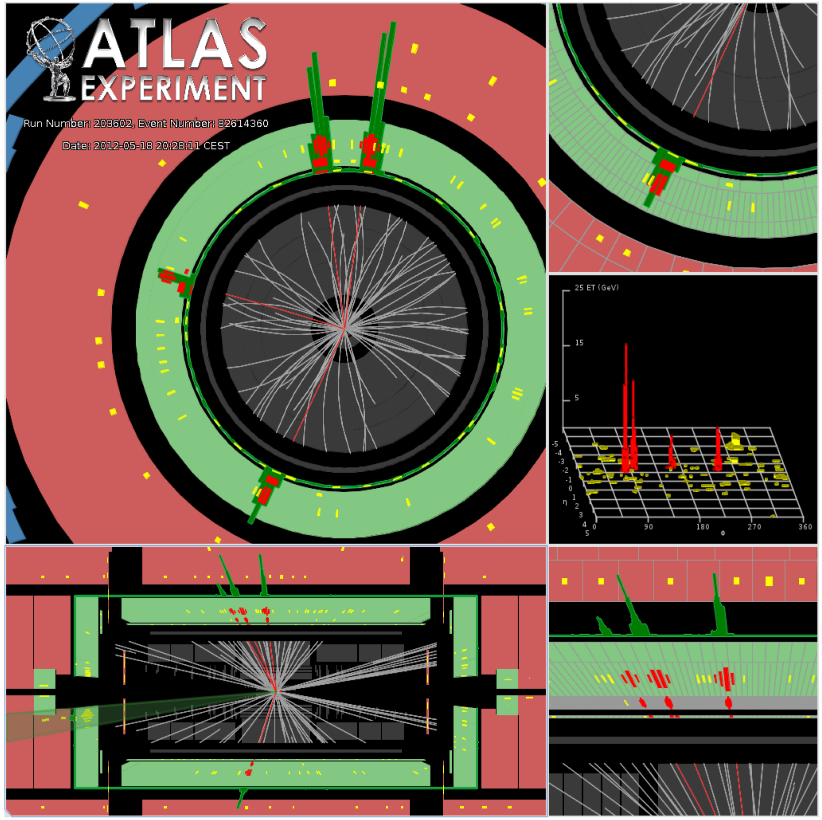
L1 trigger rate: 8 kHz

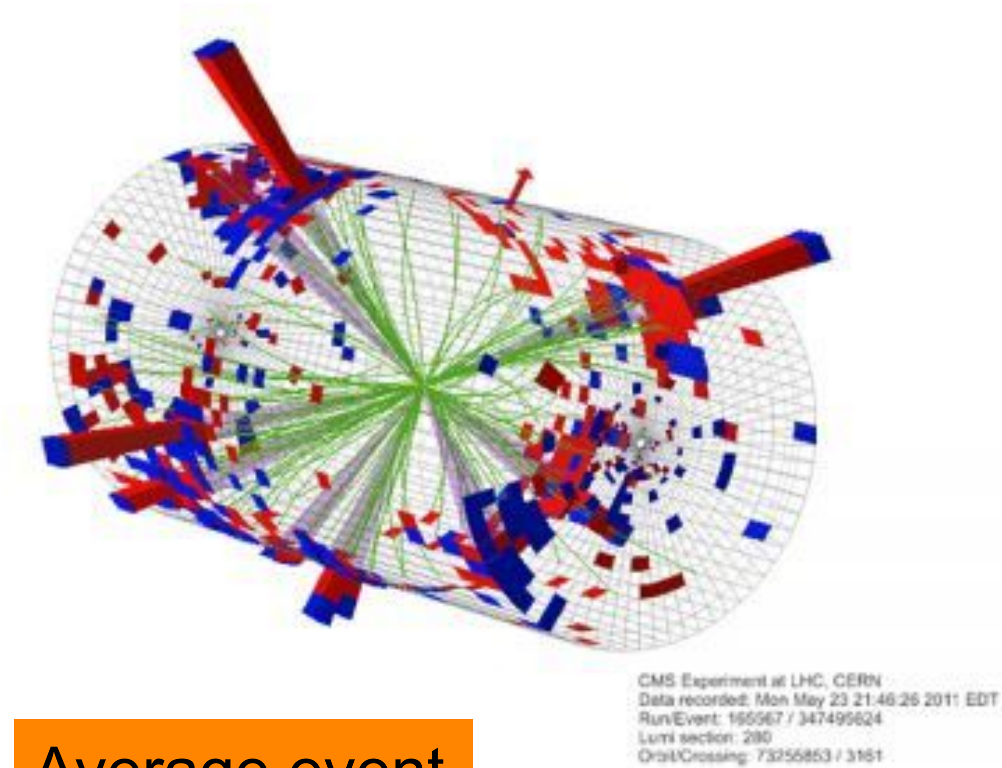


Average event size > 40 MB

ATLAS and CMS

L0 trigger rate: 75-100 kHz





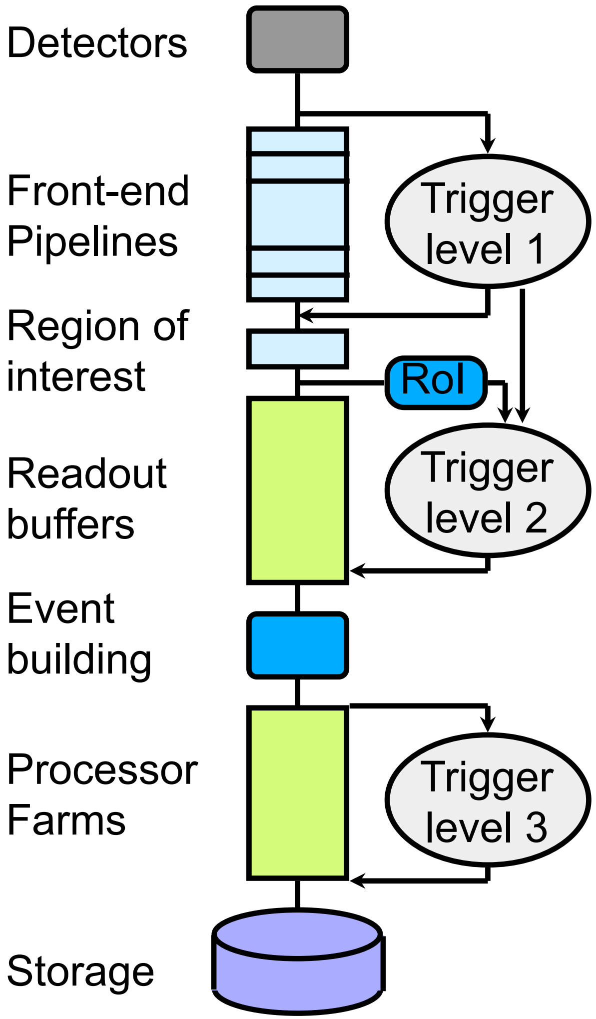
Average event
Size ~ 1 MB

Different DAQ approaches

- Lower event rate, many channels (ALICE)
  - Simpler trigger
    - Naively: something happened ("Activity")
  - Read out entire event
- High event rate, few channels (LHCb)
  - Read out only "active" channels
    - Currently: zero-suppression in DAQ readout
    - Upgrading to so-called "triggerless" DAQ
- Medium event rate, many channels
  - Sophisticated, multi-level trigger
    - More "physics-like" than activity trigger
  - Read out entire event after level-1 trigger

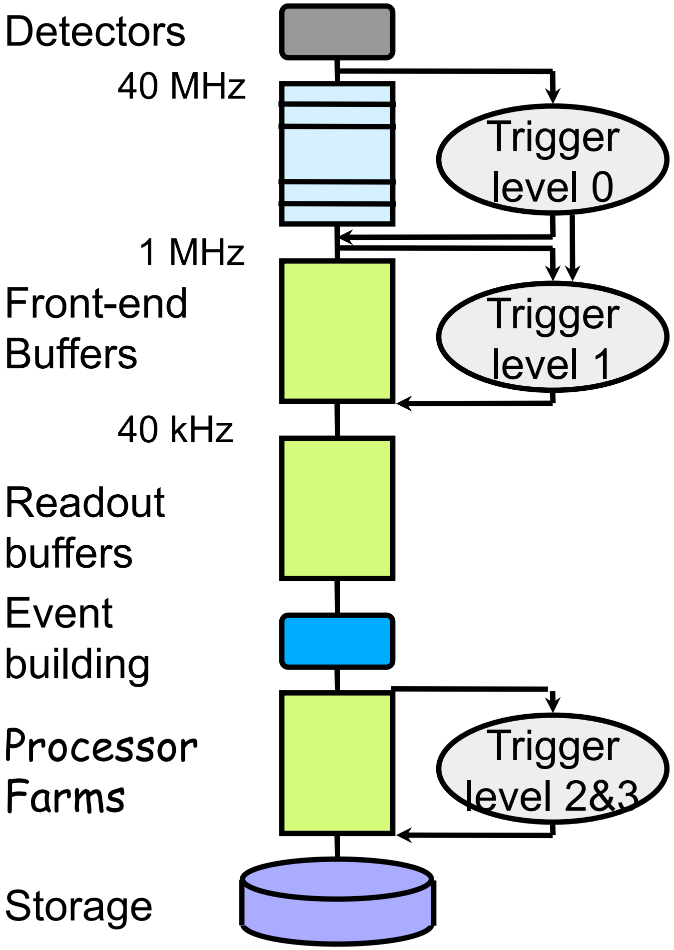
I will *mainly* focus on these… (ATLAS, CMS)

ATLAS trigger



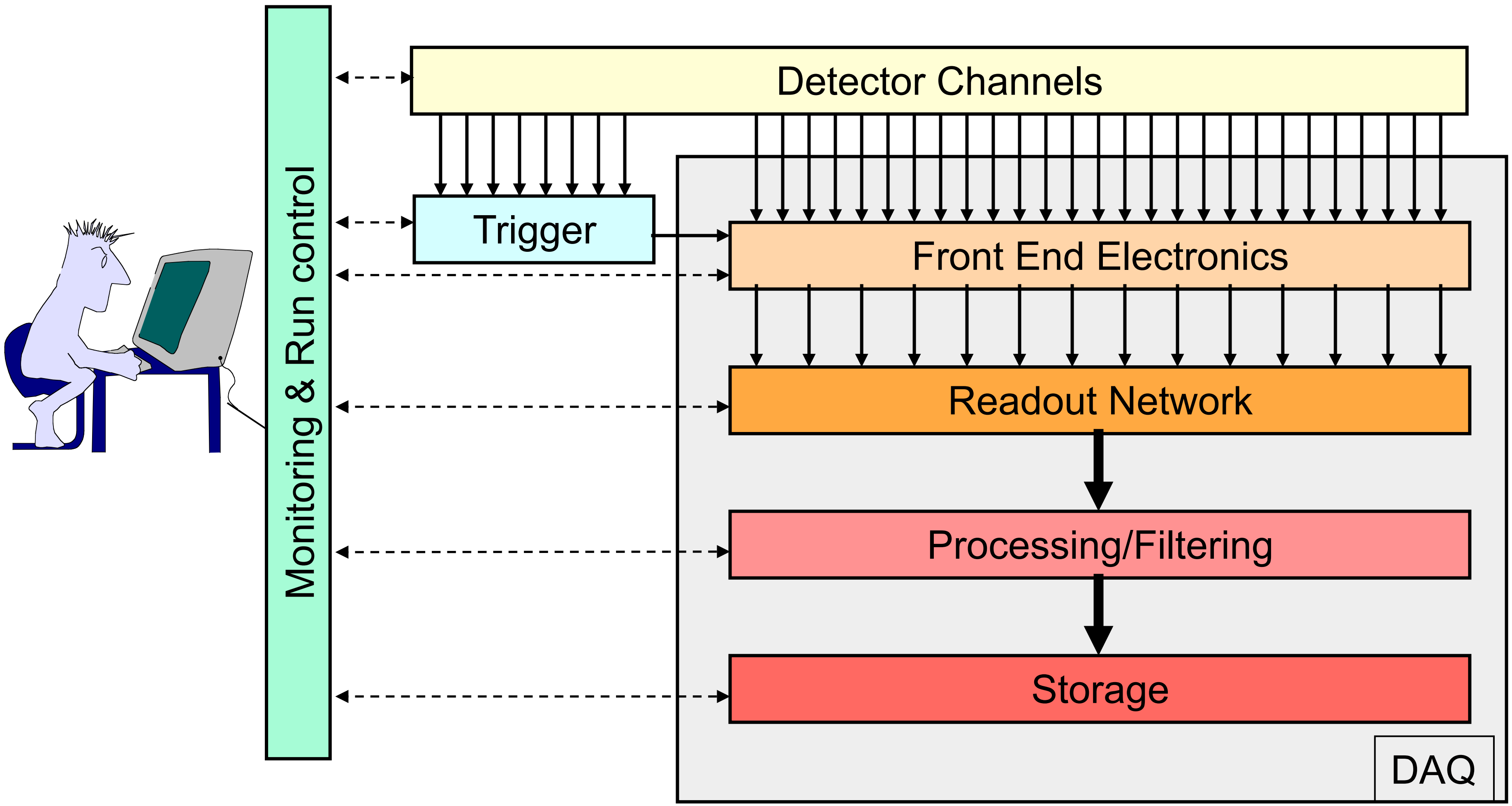
- **Level-1 (3.5 ms) (custom processors)**
  - Isolated clusters, jets, ET in calorimeters
  - Muon trigger: tracking coincidence matrix.
- **Level-2 (100 ms) (processor farm)**
  - Guided by Regions Of Interest (RoI) identified by Level-1
  - Select detector data routed to CPUs by routers and switches
  - Feature extractors (DSP or specialized) perform refined object ID algorithms
  - Staged local and global processors
- **Level-3 (≫ms) (commercial processors)**
  - Reconstruct the event using all data
  - Select of interesting physics channels

LHCb trigger system



- **Level-0 (4 ms) (custom hardware)**
  - High p_T electrons, muons, hadrons
  - Pile-up veto.
- **Level-1 (1000 ms) (specialized processors)**
  - Vertex topology (primary & secondary vertices)
  - Tracking (connecting calorimeter clusters with tracks)
- **Level-2 (≫ms) (commercial processors)**
  - Refinement of Level-1. Background rejection.
- **Level-3 (≫ms) (commercial processors)**
  - Event reconstruction. Select physics channels.

TDAQ and run control



Putting it all together…

Three more concepts today

- Front-end electronics
- Signal processing
- Dead time

Plus…

- Introduction to FPGA lab

Front-end electronics and signal processing

Front-end electronics

- Provide the interface between the detector sensors and trigger/DAQ, including:
  - Sensor signals ➔ digital data
  - Interface with the first level trigger
  - Pipeline buffering and readout to DAQ
- A critical part of the detector design. Design choices strongly influence:
  - Detector resolution and dynamic range
  - Signal/noise ratio
  - Maximum channel occupancy (pileup)
  - Etc.

Front end electronics

Closely related, often on-detector

- Input conditioning
  - Convert detector input signals to a form useful for the trigger, readout
  - Amplifiers, shapers, integrators...
- Sampling and digitization (ADC)

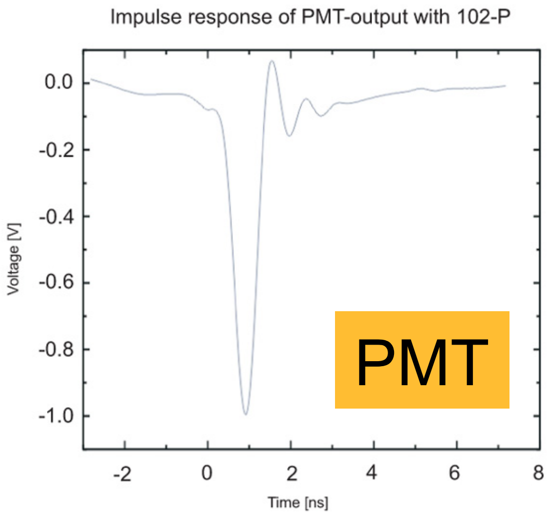
Beginning to migrate off-detector

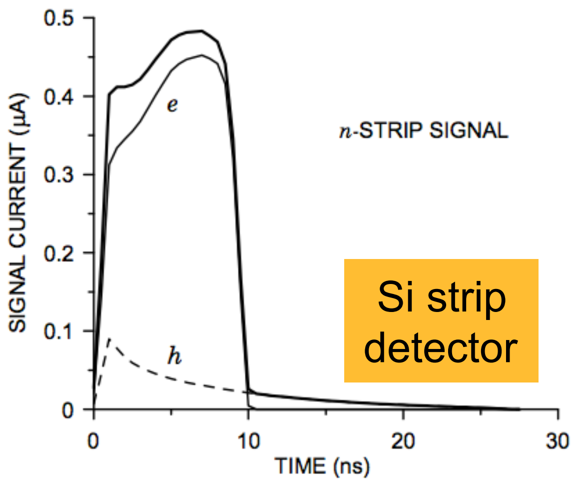
- Buffering and readout

Usually off-detector

- Signal processing (for trigger)
  - Amplitude
  - Timing

Input conditioning/sampling

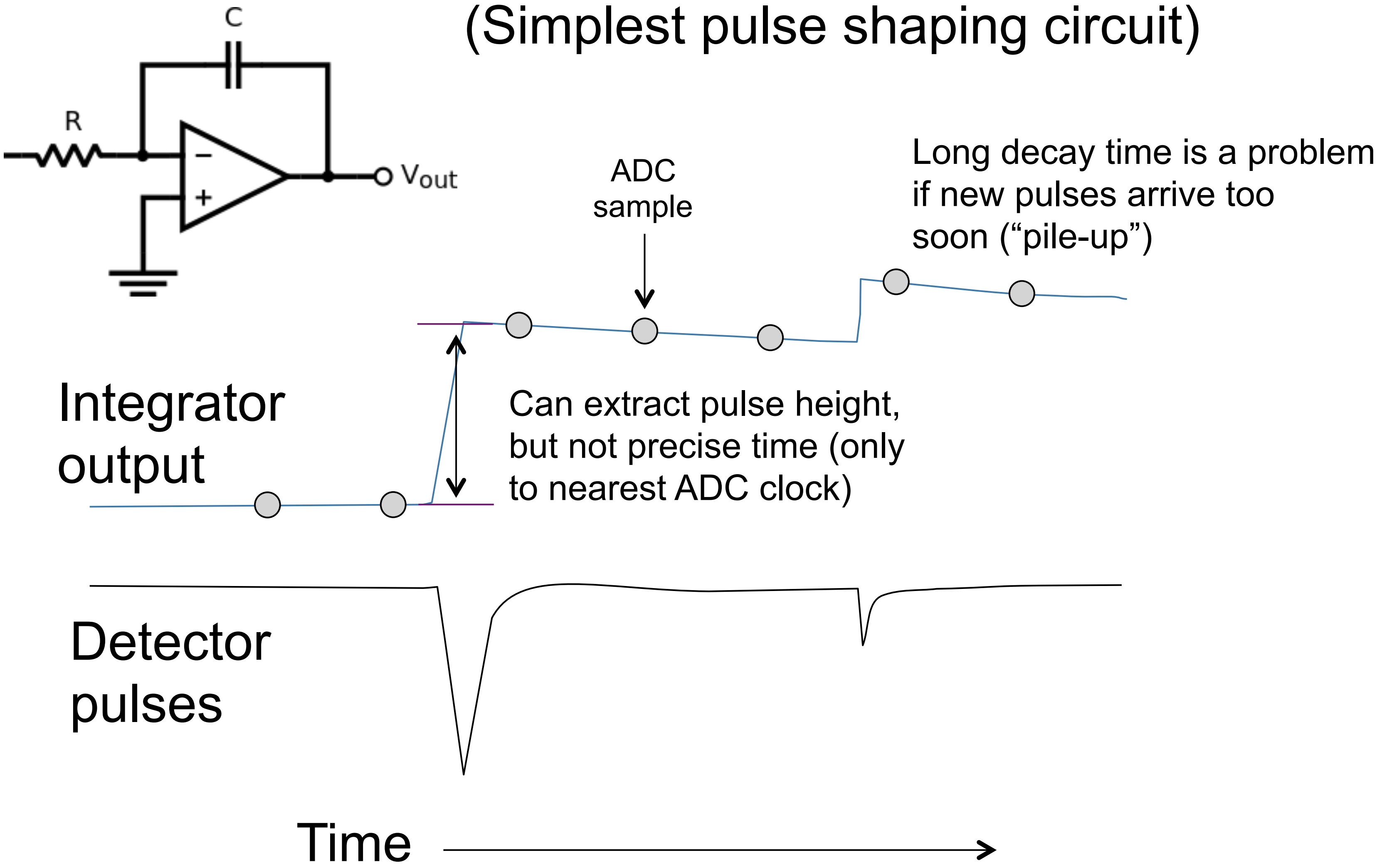




- Raw pulses can be fast and have complex shapes
  - Would fast ADCs to directly measure
  - Expensive, power-hungry, low dynamic range
- A solution is pulse shaping
  - Convert fast pulses into a slower, well-defined shape
  - Ideally amplitude-independent
  - Match to affordable ADC with the desired dynamic range (# bits)

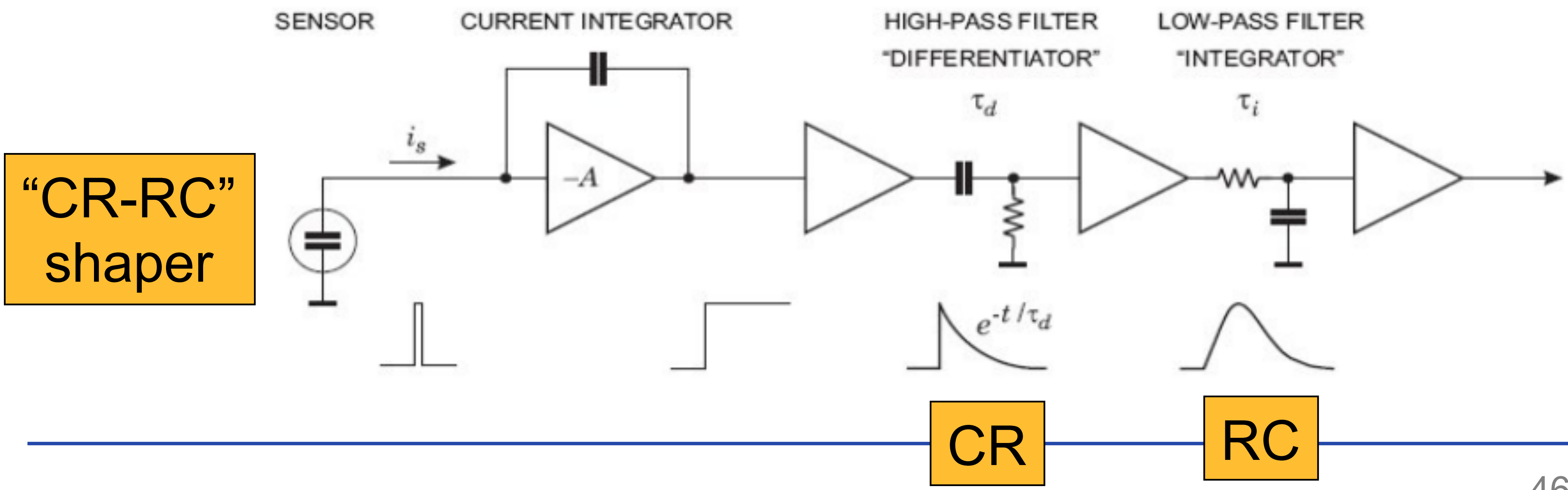
Simple charge integrator

(Simplest pulse shaping circuit)



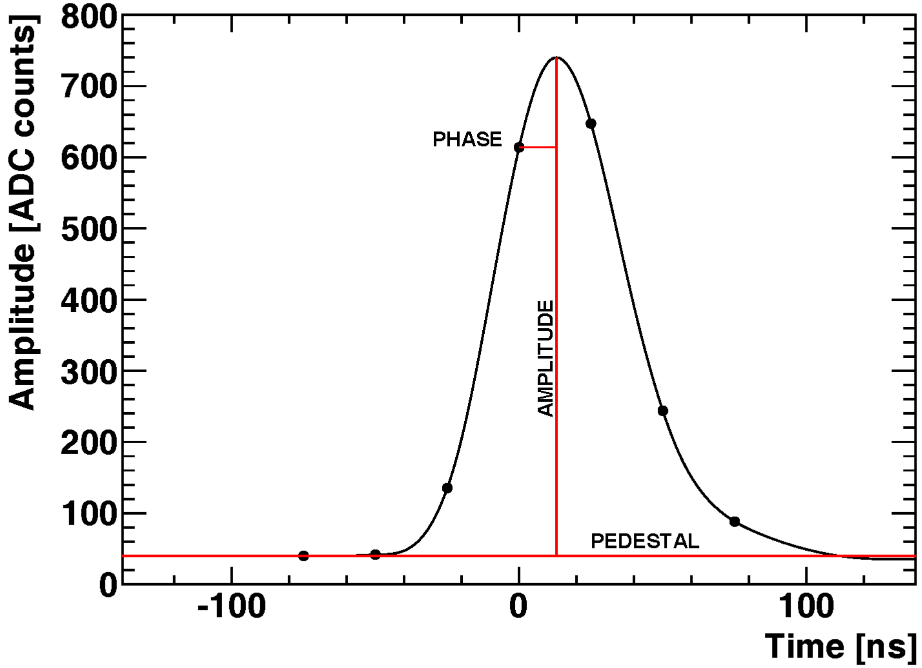
Differentiator/integrator

- Commonly-used shaper in particle detectors
- Differentiator (high-pass filter)
  - Maximum amplitude of shaped pulse
  - Pulse duration (decay time)
- Integrator (low-pass filter)
  - Slows down rise-time of pulse



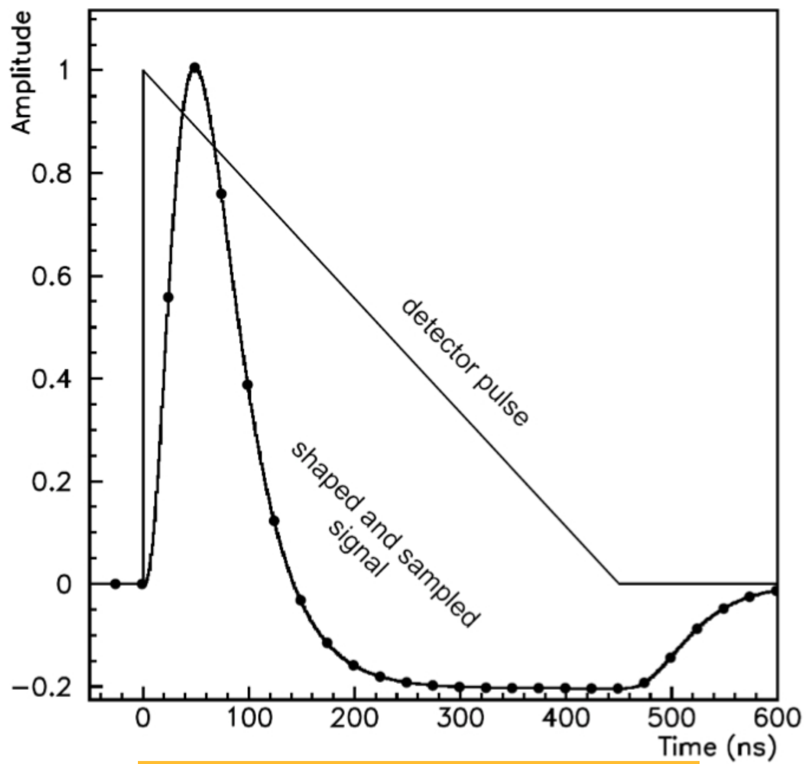
ATLAS calorimeter shapers

Tile (hadronic) calorimeter



Unipolar shaper, makes a short PMT pulse longer

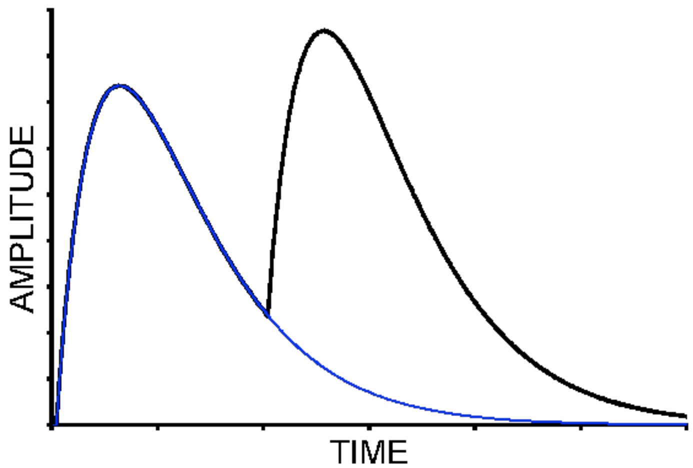
LAr (EM) calorimeter

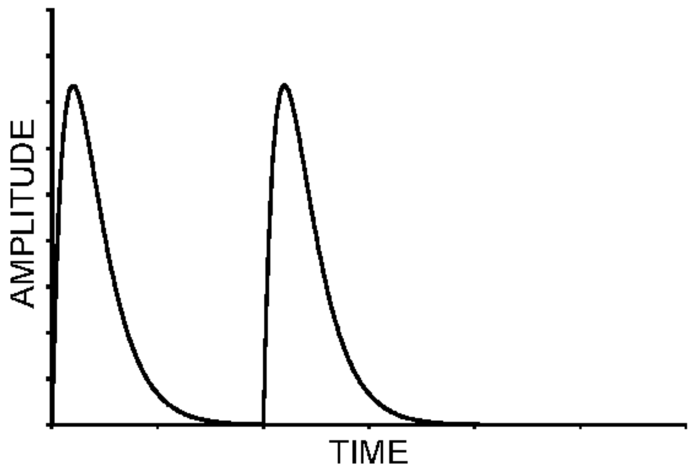


Bipolar shaper, shortens a long ionization signal

Pile-up

- Broad pulses good for digital processing
  - Better amplitude and timing estimates
  - Less sensitive to random nose
- But too broad pulses increase the pile-up rate
- Need to compromise

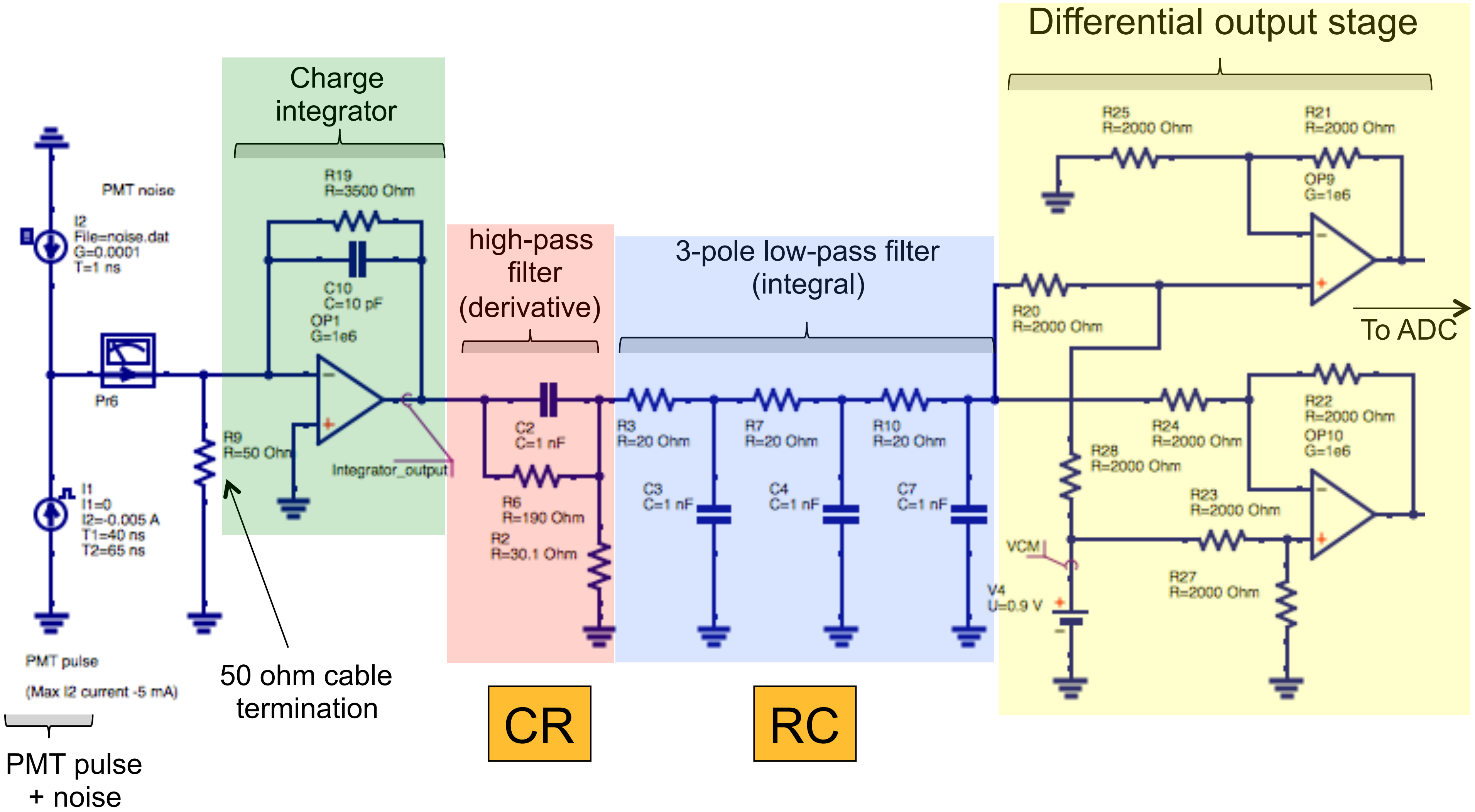




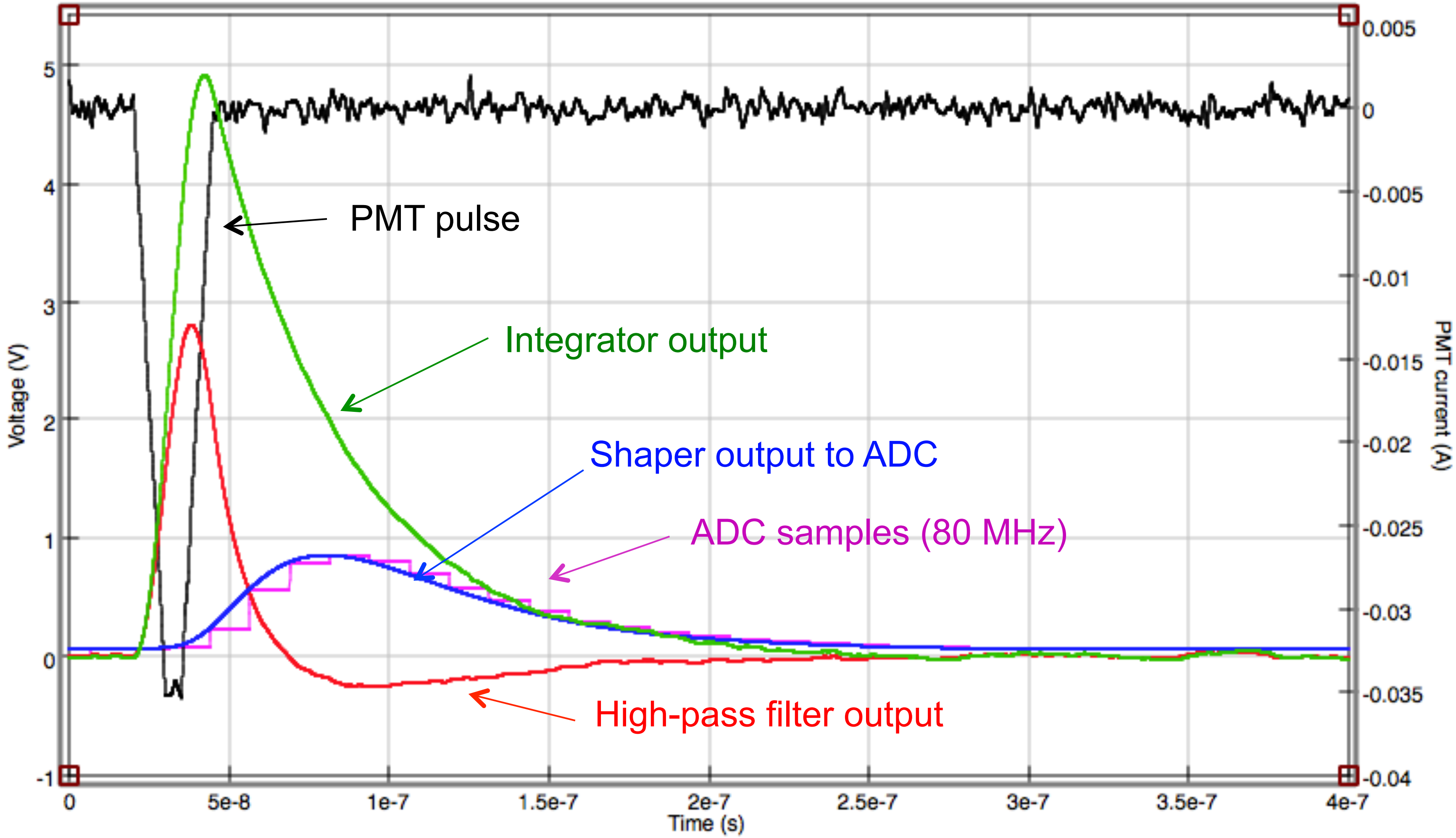
Case study

- The following is a simulation of a slow shaper idea for a hadronic calorimeter
- Design constraints:
  - Fast pulse (PMT)
    - Unipolar shaper (CR-RC)
  - Want a large dynamic range
    - Assumed a 16-bit ADC (13.5 effective bits) with up to 80 MHz sampling rate
  - Low channel occupancy (low pile-up)
    - Pulse can be slow (~150 ns)
    - Oversampling increases # of effective bits

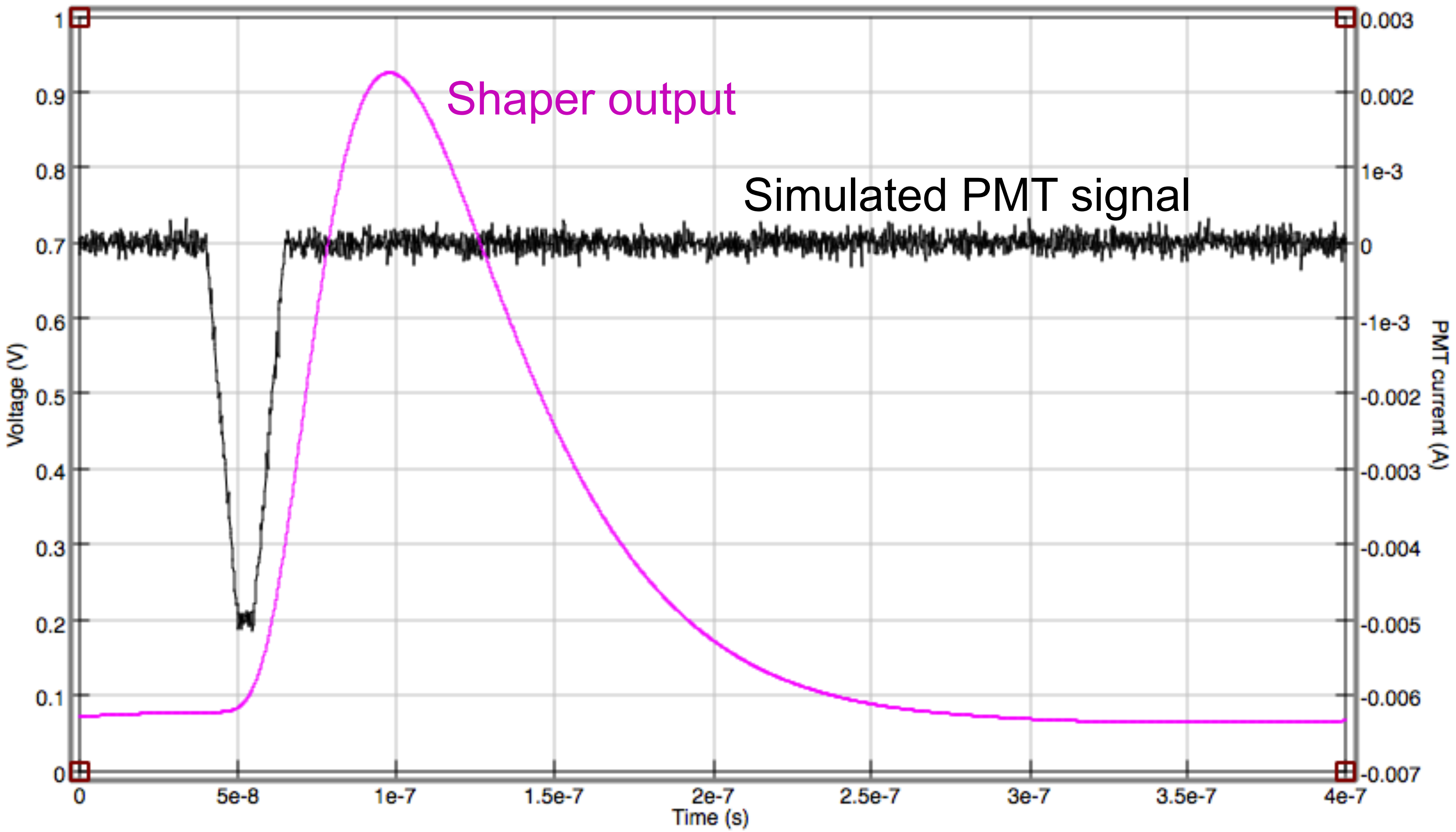
Analog input and shaper



Transient simulation

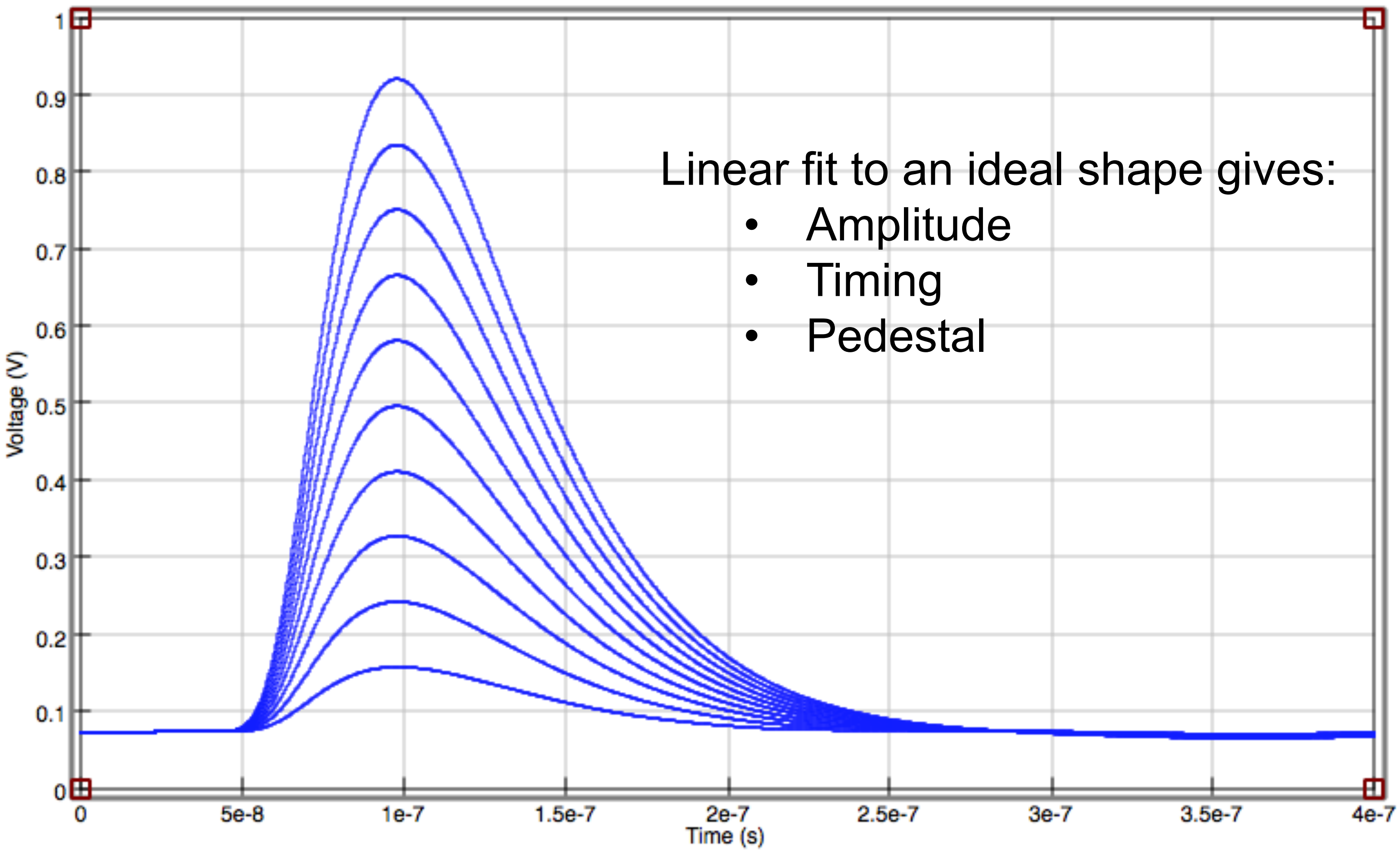


PMT input signal



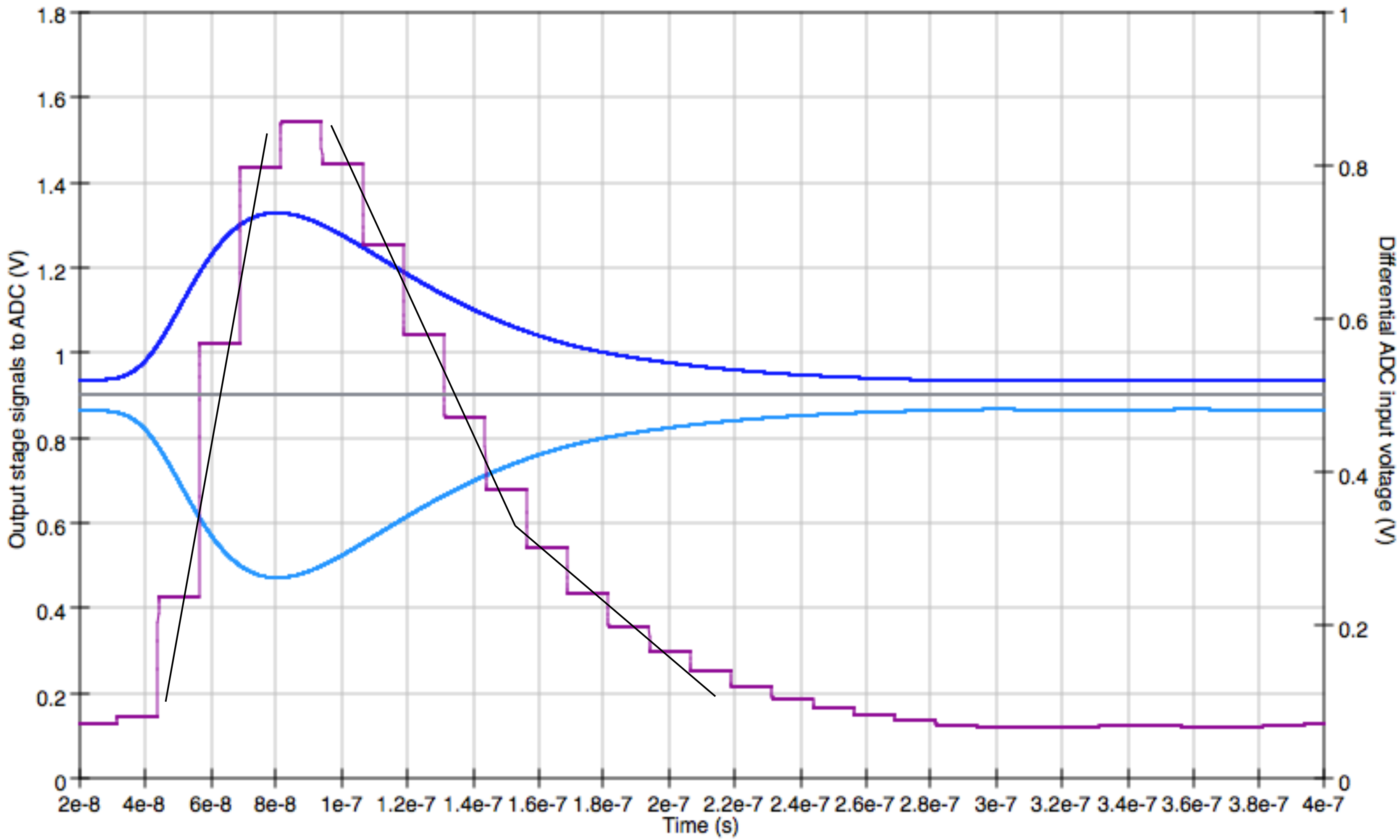
Shaper output

Shape is amplitude-independent



Sampled shape (80 MHz)

Multiple samples on rising and falling edges give good timing estimates



Signal processing

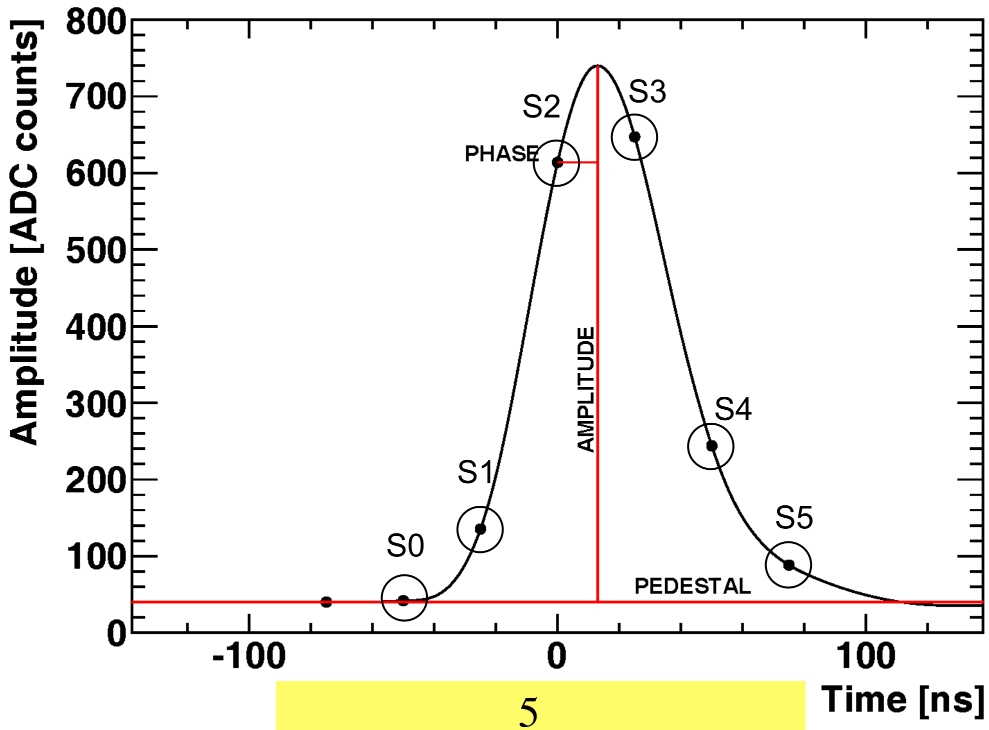
Signal processing

- From sampled signal, need to extract:
  - Pulse amplitude
  - Timing of the pulse
    - Coarse timing (which BX?)
    - Fine timing (ns level)
      - Good for eliminating some backgrounds
- Common approach: <u>digital filter</u>
  - Different approaches
  - I will show finite-impulse-response (FIR) filter with <u>matched</u> coefficients…

FIR filter example

<u>Matched filter</u>:

- Coeffs Cn proportional to normalized sample heights (including noise)
- C0 is a negative number (pedestal subtraction)
- Filter output is phase dependent

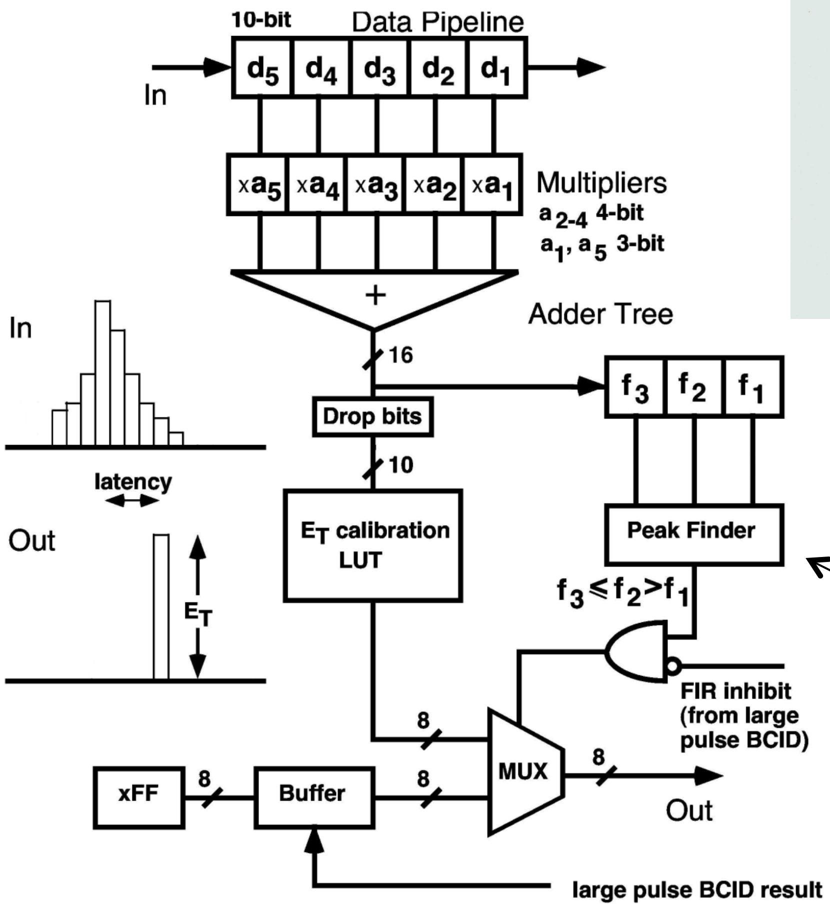


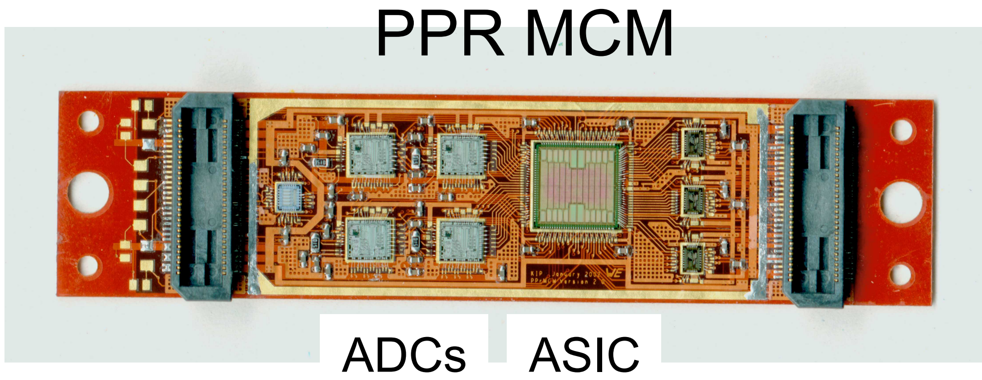
$$F(t) = \sum_{n=0}^{5} C_n S_n(t)$$

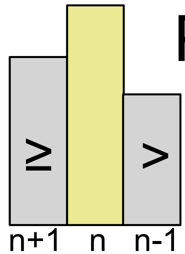
Time extraction: coefficients ~ <u>derivative</u> of the pulse shape

Hardware implementation

ATLAS trigger preprocessor ASIC



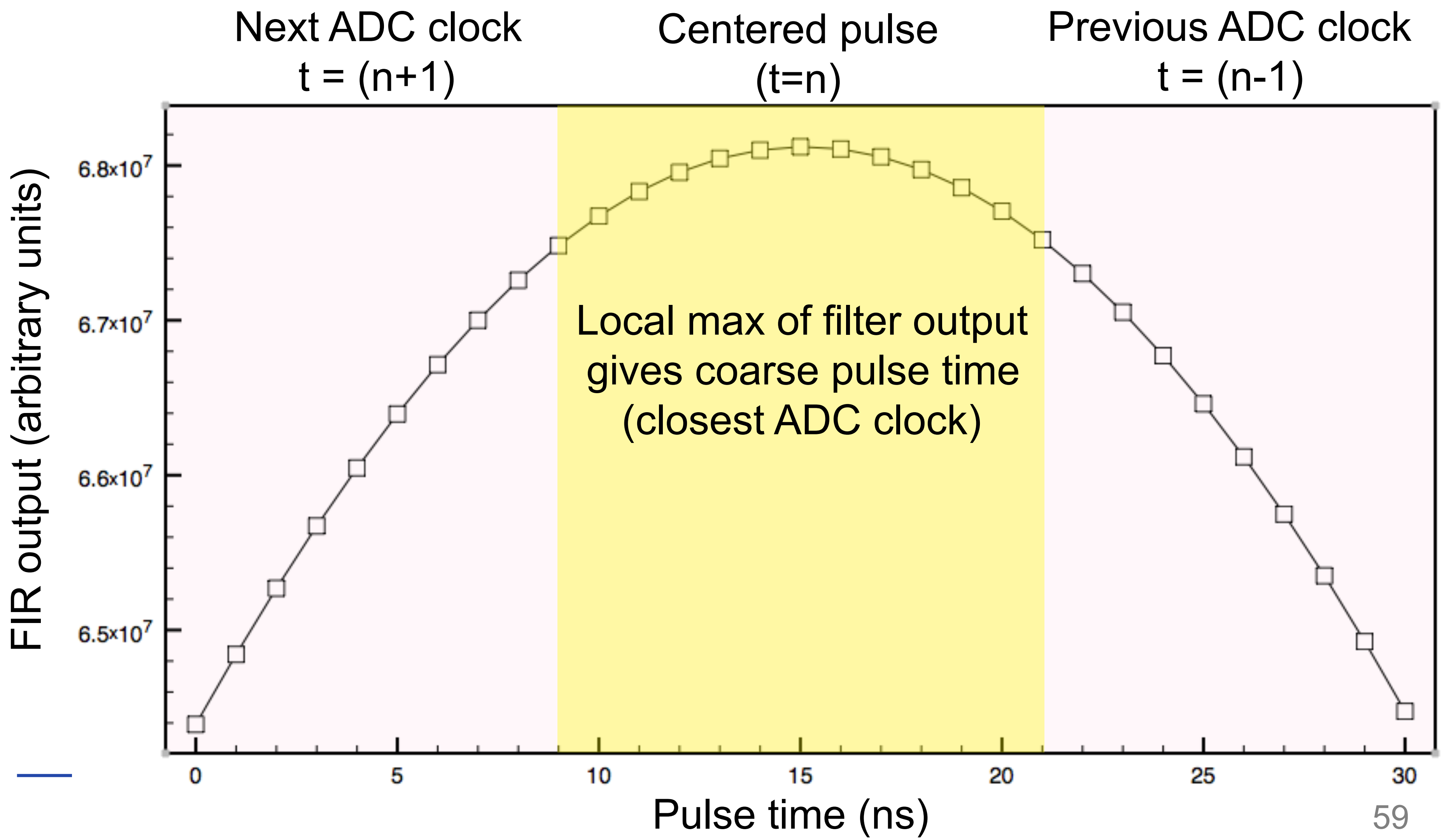




Peak finder identifies BX of pulse

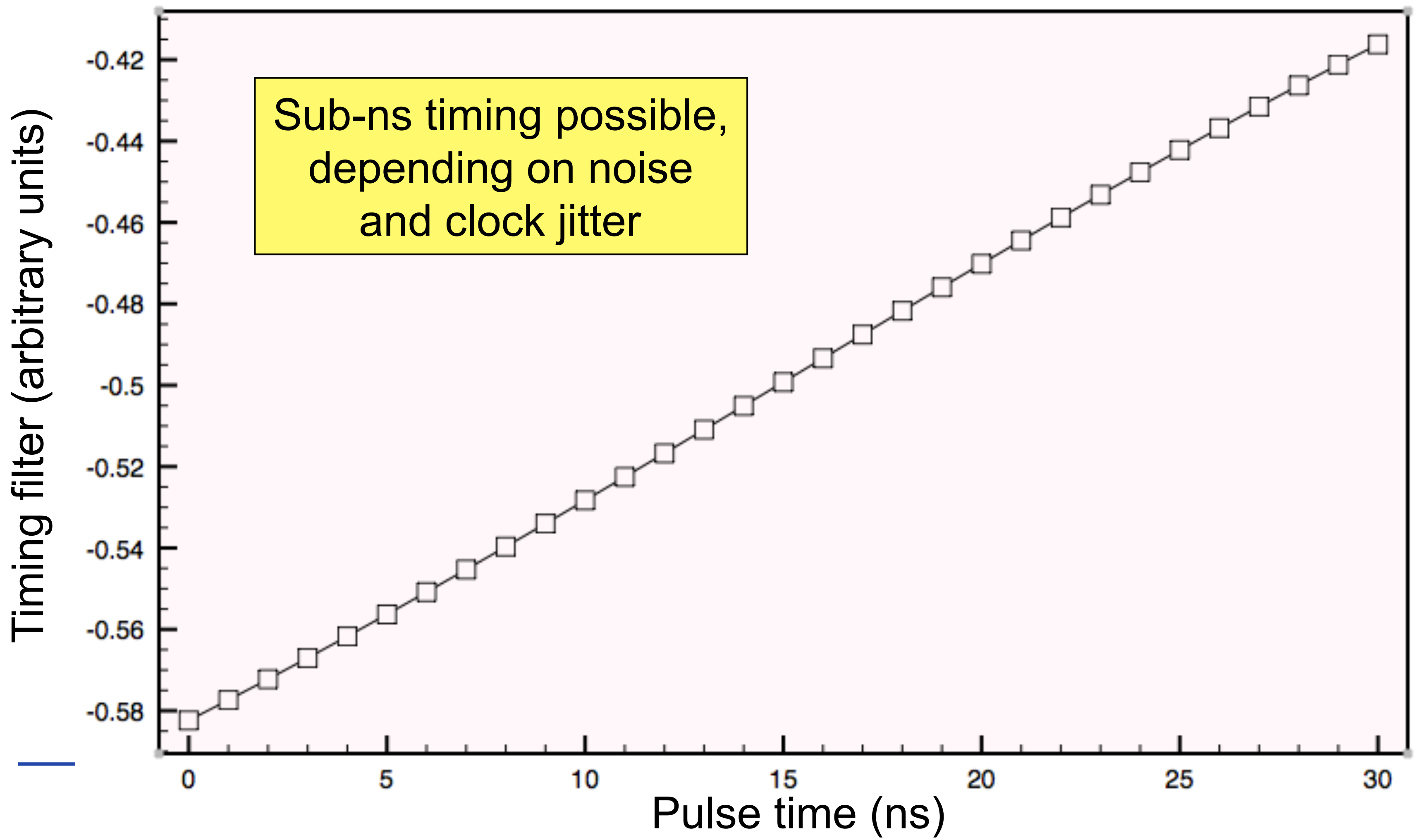
FIR response of shaper sim

15 samples with fixed coefficients



Fine-timing estimate (offline)

Second FIR filter, with coefficients calculated from the derivative of pulse shape at each sample



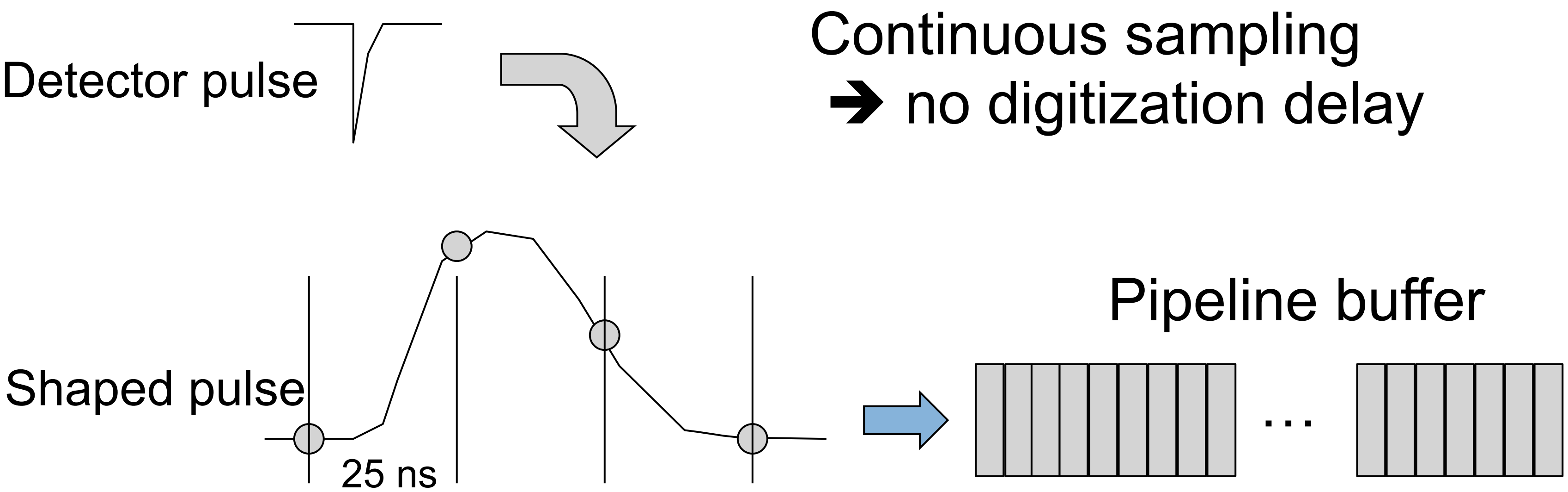
Dead Time

Dead Time

- Fraction of time that system cannot record data
- Sources of dead time:
  - ADC conversion time (less relevant for FADCs)
    - Finite time to sample and digitize an event.
  - Readout dead time
    - Finite time to write ADC samples to R/O buffers
  - “Busy” condition in derandomising buffers
    - Inhibit triggering new events to prevent overflow
  - Operational dead time
    - Detector readiness, etc…

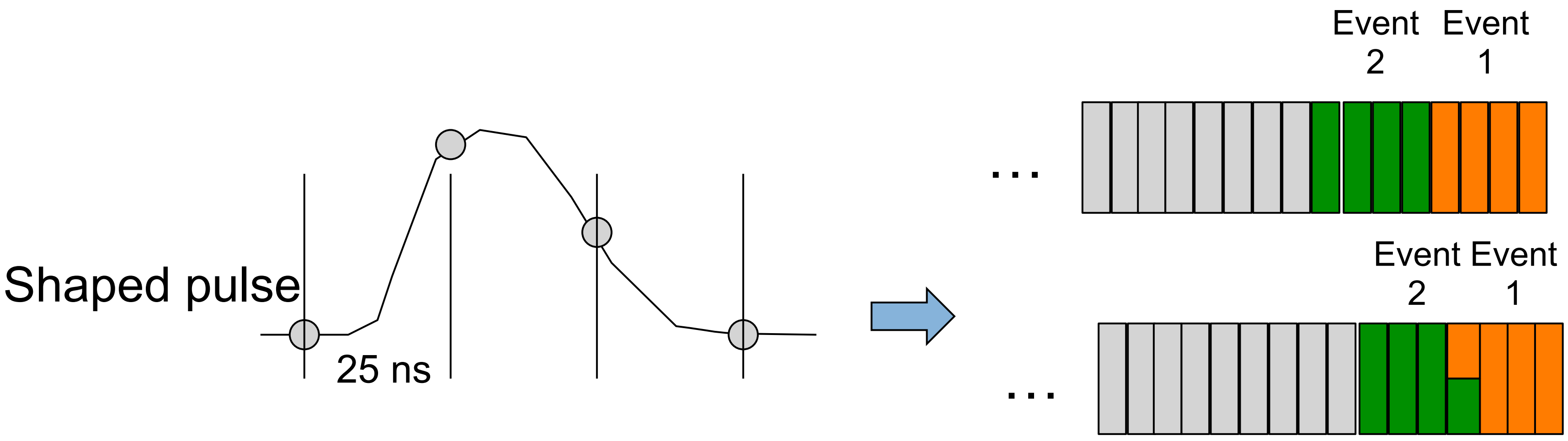
ADC conversion time

- Collider experiments use fast, pipelined ADCs
  - Digitize every BX, store in pipeline buffer
  - ADC normally not a source of dead time

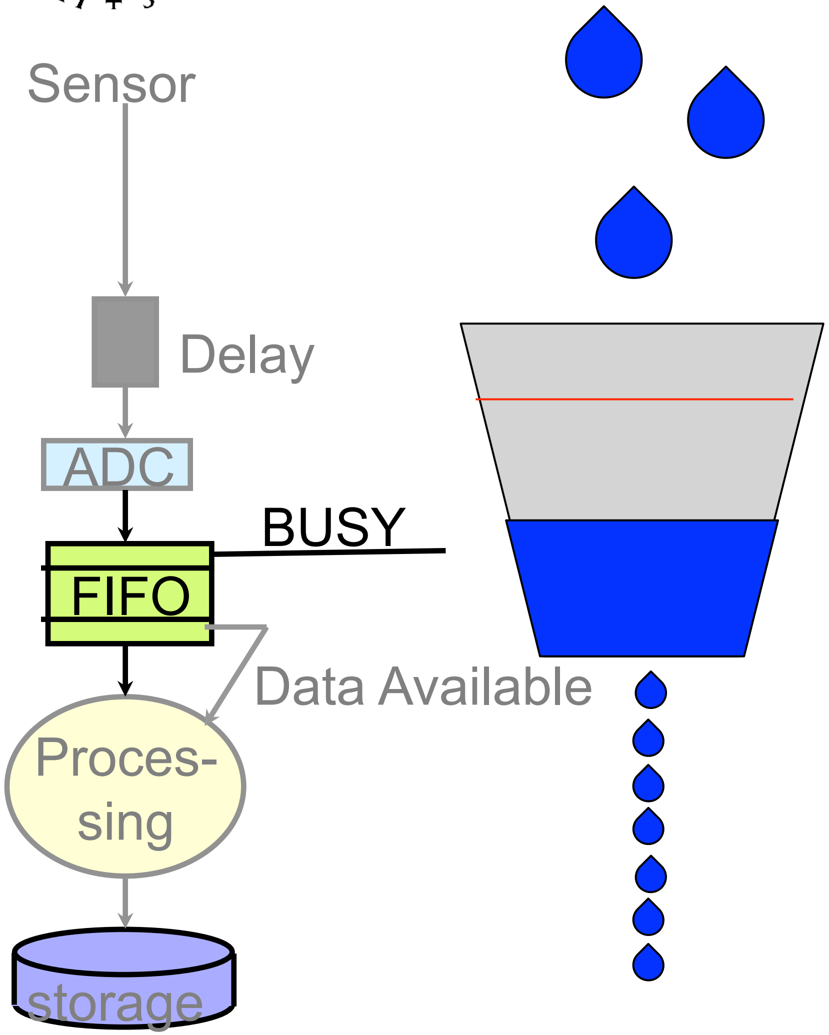


Readout dead time

- Read out multiple samples per event
  - Simpler systems can have problems reading out overlapping events
    - Result: cannot trigger events too close in time
  - Can avoid this with digital readout



"Busy" condition



- "Leaky bucket" paradigm for derandomizing buffers
- Input at random intervals
- ~ Constant output rate
- If "high- water mark" passed, block trigger
- Maximum trigger rate:
  - $R_{max} = 1/T_{readout}$
- Use sufficiently long buffers to minimize dead time below R_{max} .

Calculating the dead time

Analytic solution* assuming a constant time to read out each event:

$$D_N = 1 - \frac{S_N}{1+\rho S_N} \quad \text{where } \rho = R\tau \text{ and}$$

$$S_N = e^{N\rho} \sum_{j=0}^{N-j} \frac{(N-j)^j (-\rho e^{-\rho})^j}{j!}$$

Variables:

- N-1 buffers
- R: Input rate (trigger)
- τ: Readout time

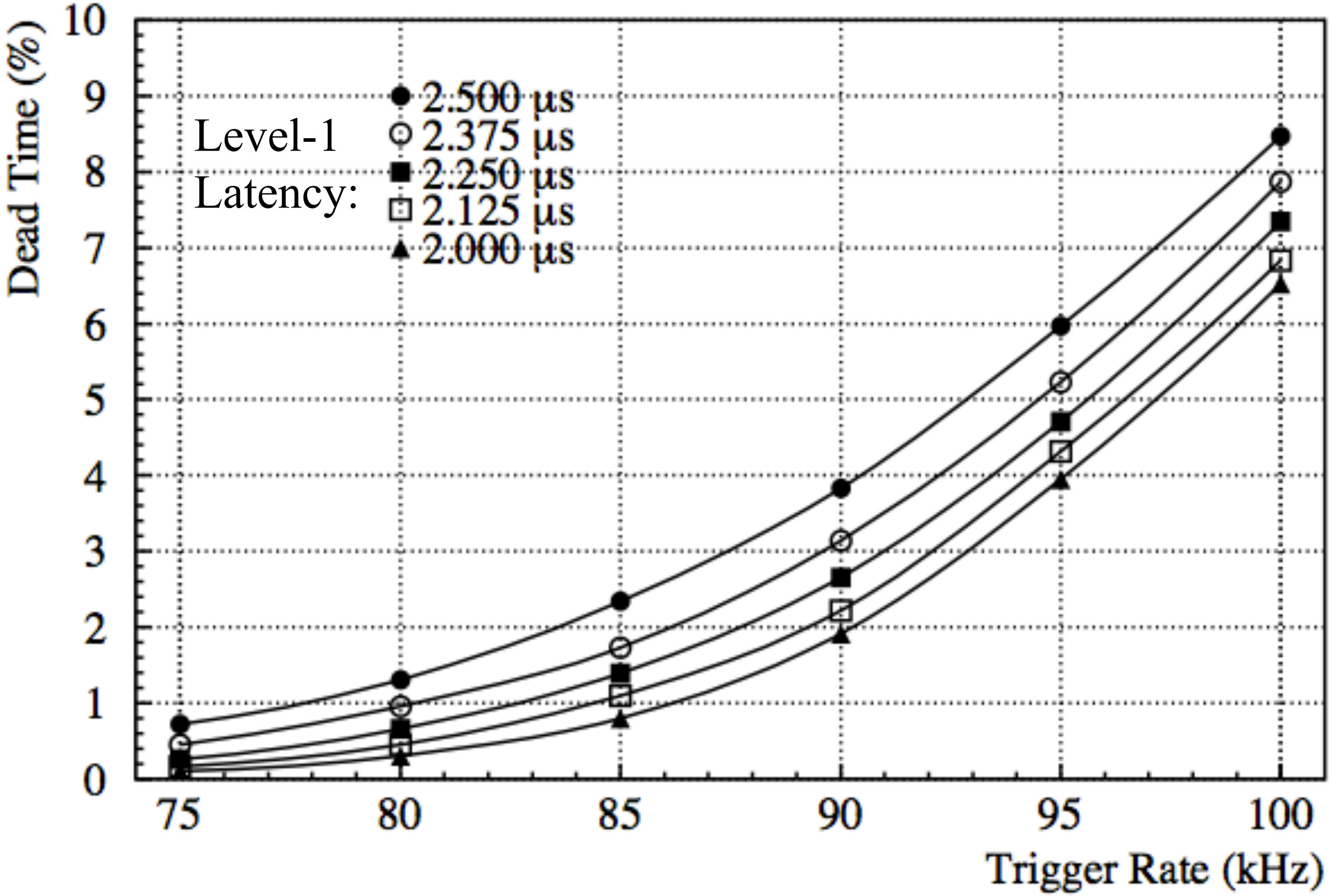
Reality: τ is not always constant, and the dead time model is often more complicated. Monte Carlo simulation provides better accuracy

* Source: G.P. Heath, "Dead time due to trigger processing in a data acquisition system with multiple event buffering", Nuclear Instruments and Methods in Physics Research, A278 (1989) 431-435.

Example: ATLAS LAr calorimeter

- Basic parameters:
  - BX rate: 40.08 MHz (25 ns)
  - Max trigger rate: 75 kHz (upgrade to 100)
  - Readout time (5 samples/event): 10.6 μs
- Other complicating factors:
  - At least five BXs (125 ns) between two Level-1 accepts (L1A) (readout dead time)
  - Not all bunches in LHC filled
    - Nominally 2808 of 3564 (trains of 72 bunches)
    - No L1A expected for empty bunches
  - Analog pipeline buffers are 144 samples long
    - Divided between pipeline and derandomizer FIFO
    - So trigger latency affects FIFO length (& dead time)

Simulated LAr dead time



In my next two lectures…

- Algorithms and architectures
  - More detailed examples of trigger systems, and how they are built
  - The “tools” available for collider detector triggers, and how they can be used
- New directions: SLHC
  - SLHC planning and implications
  - New technologies and architectures
  - TDAQ upgrades for SLHC

TDAQ lab exercise

- Field Programmable Gate Array (FPGA) tutorial
- Specify in VHDL language:
  - Simple Boolean logic gate (e.g. AND)
  - A coincidence counter
- Implement and test in real hardware

Homework

- Write a Monte Carlo dead time simulation of a semi-realistic LHC readout
  - (Based on ATLAS LAr)
  - Use ‘leaking bucket’ algorithm
- Parameters:
  - 40 MHz bunch crossing rate (25 ns)
  - L1 accept rate (random): 75 kHz
  - Readout time per event: 10.6 μs
  - Minimum time between L1 accepts: 125 ns (5 bunch crossings)
- Investigate the questions:
  - How deep must the derandomiser buffer be in order to keep dead time below 1%?

Questions?