Summer Student
Lecture Program 2022

Electronics, Trigger and Data Acquisition. 2/3

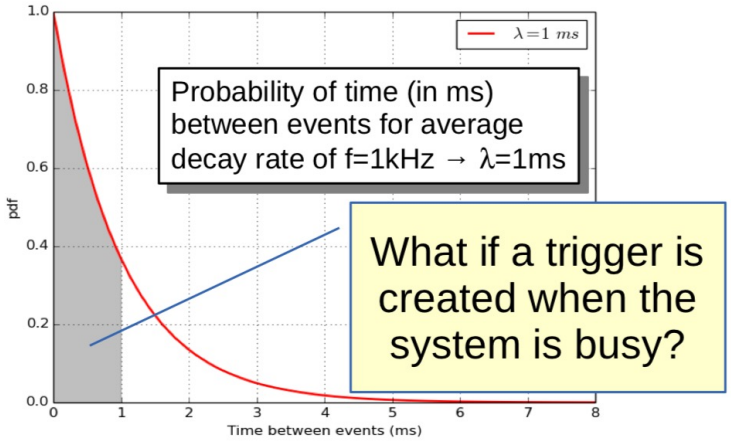
E. Meschi – CERN EP Dept. – CMS Experiment

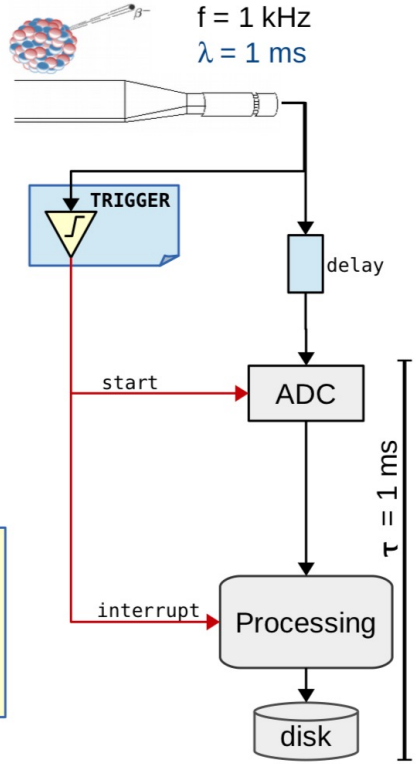
Credits:
Past SSLP ETD lecture series
EM: lectures on DAQ/Trigger at U.Padua 2018-2020
ISOTDAQ: International School of Trigger and DAQ
https://indico.cern.ch/event/928767/
Material from various papers and books (bibliography at the end)

- Trigger and DAQ system concepts
- From signal to physics through examples
- Timing
- Data transport, links, buses
- Queues and Event building
- On-line data processing

Sampling a physics process

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

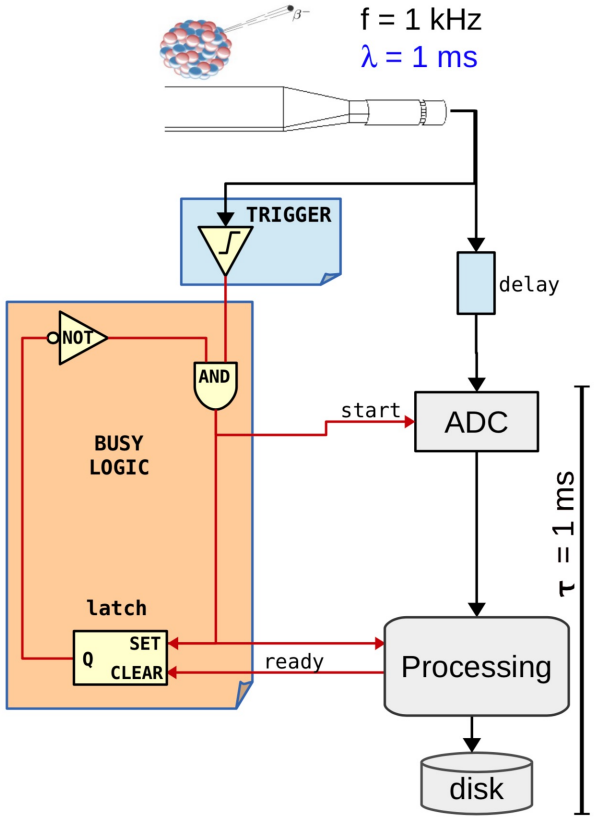




What happens if
New trigger arrives while system is busy ?
a) Each new trigger is accepted and ”restarts” the process
-> **paralysable**
b) No new trigger is accepted until the process is complete
-> **non-paralysable**

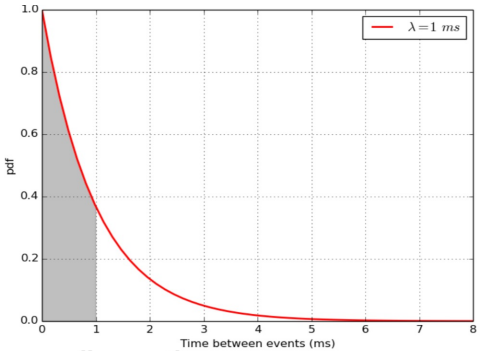
DAQ and Trigger with busy logic

- **Busy logic** avoids triggers while the system is busy in processing
  - E.g.: AND port and a latch
- Latch (**flip-flop**):
  - a bistable circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)



Deadtime

- Which (average) DAQ rate can we achieve now?
  - Reminder: w/ a clock trigger and τ = 1 ms the limit is 1 kHz



- Definitions
  - **f:** average rate of physics phenomenon (input)
  - **ν:** average rate of DAQ (output)
  - **τ: deadtime**, the time the system requires to process an event, without being able to handle other triggers
  - probabilities: P[busy] = $\nu\,\tau$; P[free] = 1 - $\nu\,\tau$
- Therefore:

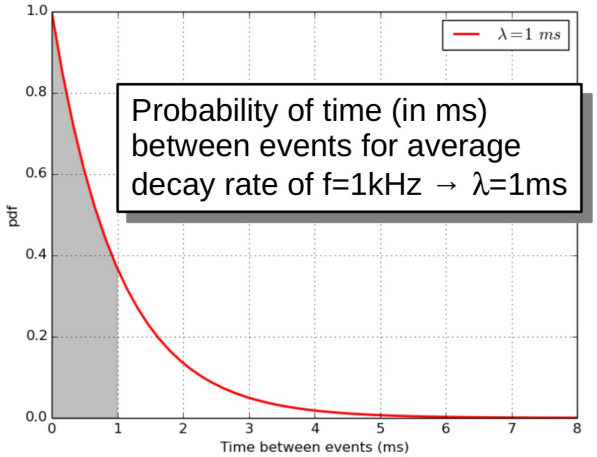
$$\nu = f\,P[free] \;\Rightarrow\; \nu = f(1-\nu\,\tau) \;\Rightarrow\; \nu = \frac{f}{1+f\,\tau}$$

Deadtime and Efficiency

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

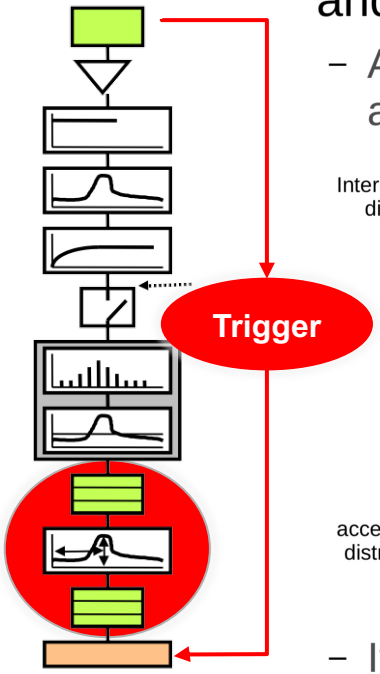
- So, in our specific example

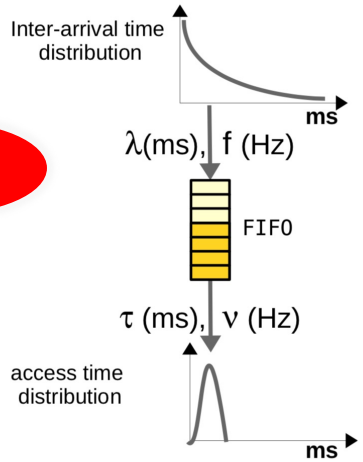
$$\left| \begin{array}{l} f = 1\,kHz \\ \tau = 1\,ms \end{array} \right. \rightarrow \left| \begin{array}{l} \nu = 500\,Hz \\ \epsilon = 50\,\% \end{array} \right.$$

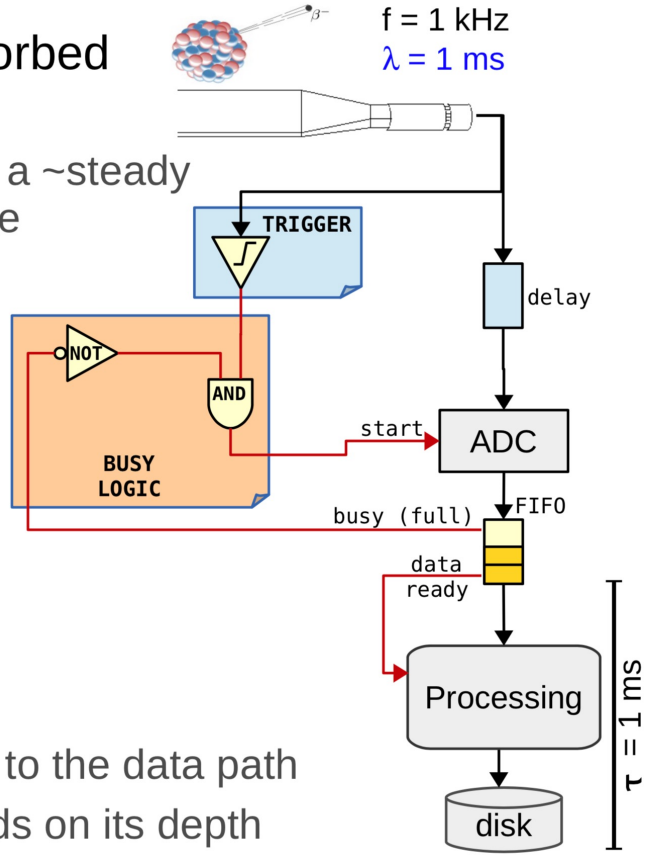


Derandomization

- Input fluctuations can be absorbed and smoothed by a queue
  - A First In First Out can provide a ~steady and **de-randomized** output rate

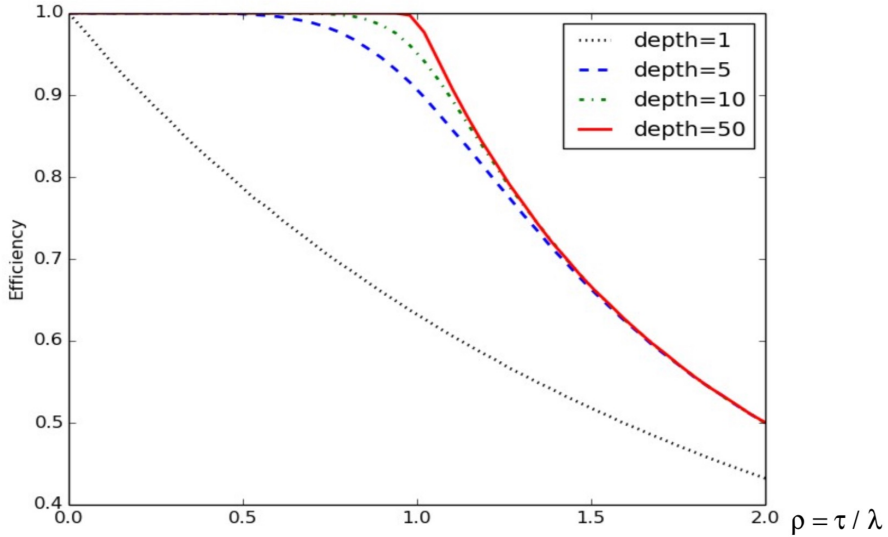


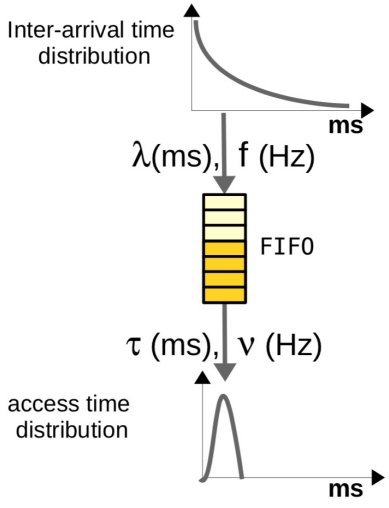




  - It introduces additional latency to the data path
  - The effect of the queue depends on its depth

A bit of queueing theory

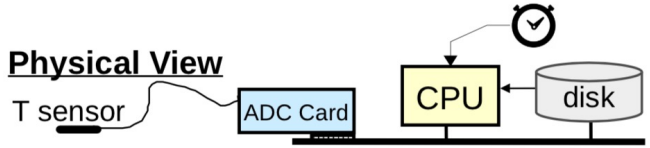




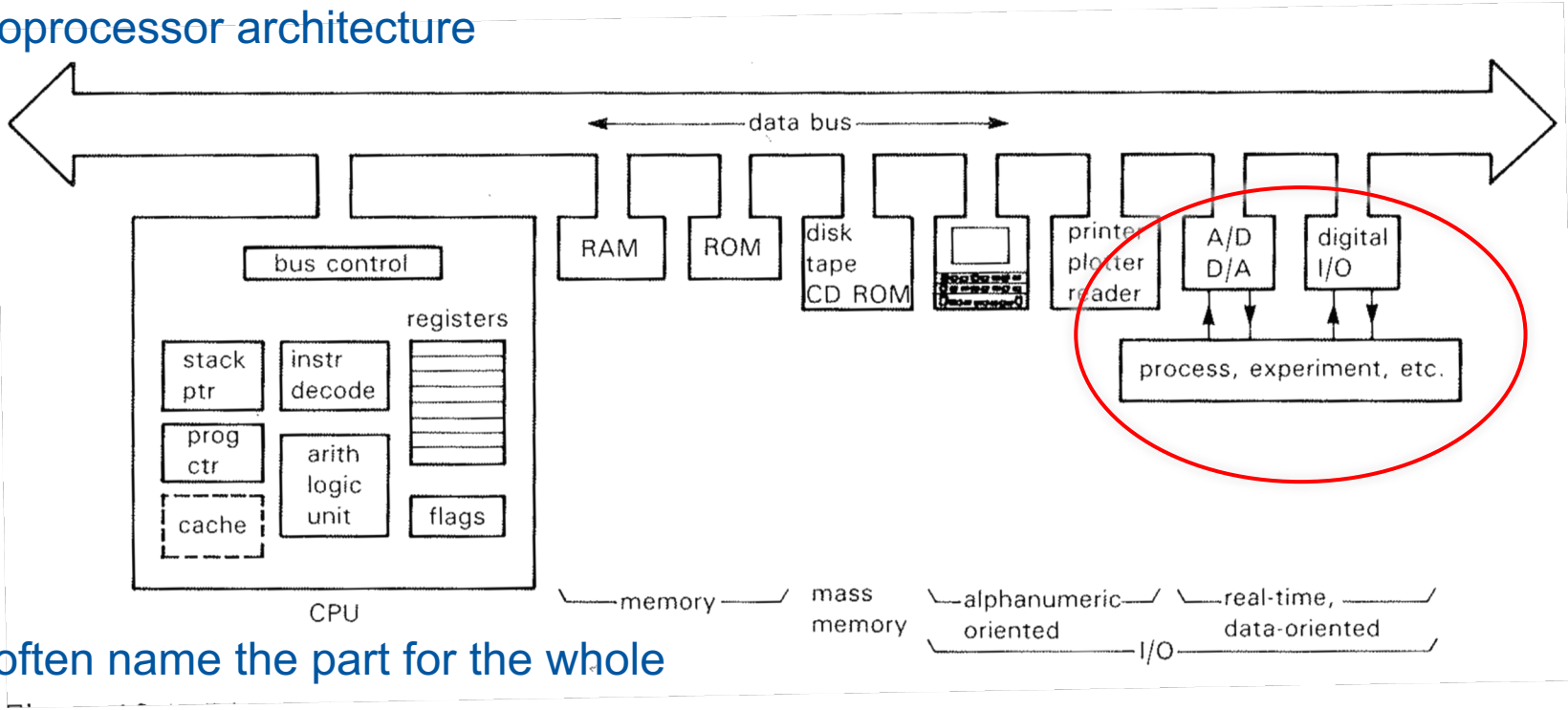
- Efficiency vs traffic intensity ($\rho = \tau / \lambda$) for different queue depths
  - **$\rho > 1$**: the system is overloaded ($\tau > \lambda$)
  - **$\rho << 1$**: the output is over-designed ($\tau << \lambda$)
  - **$\rho \sim 1$**: using a queue, high efficiency obtained even w/ moderate depth

CPUs and data buses

The CPU does the rest…



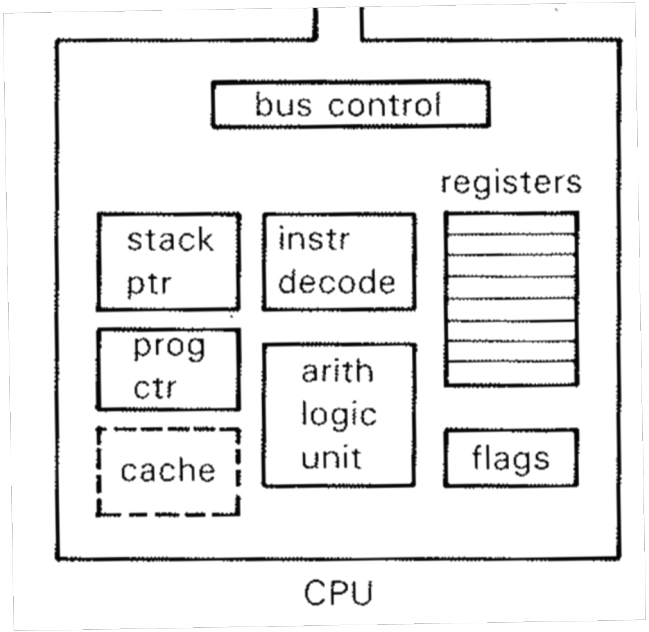
Microprocessor architecture



We often name the part for the whole

CPU (a simplified view)

- Does computation on **words**
  - Sequences of bits (32,64,128…)
- Fetches **instructions** from **memory** (the “program”)
- Instructions are bit codes corresponding to an operation (part of an “instruction set”)
  - They are first decoded and then fed to an **Arithmetic and Logical Unit** (ALU) that performs the operation on **data** contained in **registers** (e.g. add, complement, compare, shift, move…)
  - A **program counter** keeps track of the current location in the program being executed
- Data (as instructions) are fetched (usually) from **memory** over a **bus**
  - A **bus controller** handles the communication with memory and other I/O peripherals (such as a DAQ board, for example)

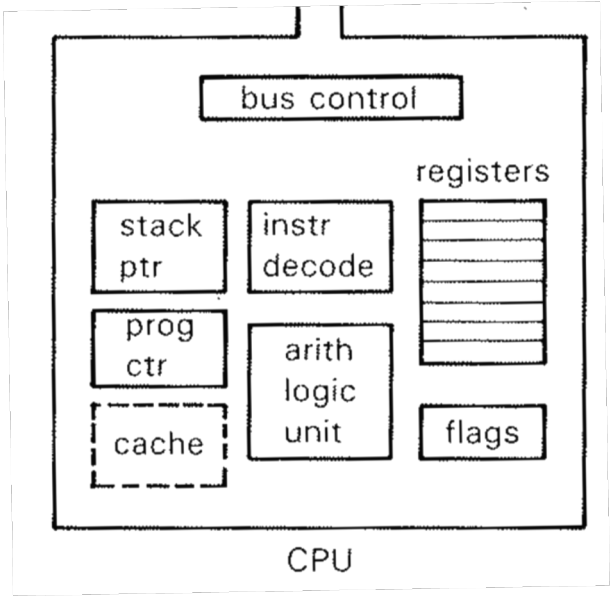


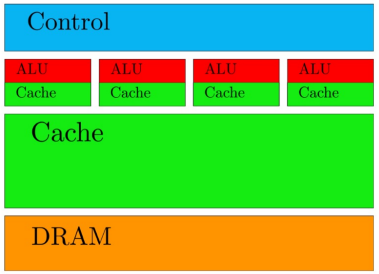
CPU (a simplified view)

- Modern CPUs have a more or less large "**cache**" – fast memory that contains recently or frequently accessed data for quick retrieval (not requiring access to the memory bus)

The **stack** is a portion of memory that works like a LIFO: there are two fundamental instructions PUSH and POP to move data to and from the stack, and a **stack pointer** always pointing at the top – more on this (maybe) later

This simplified view is common (give or take few parts) to many different architectures, **including those specific to DAQ** and trigger (the CPU could be just an "intelligent bus master")





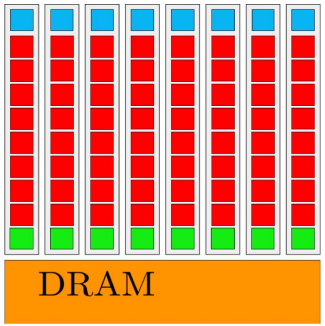
CPU

- Large caches (slow memory accesses to quick cache accesses)
- Powerful ALUs
- Low bandwidth to memory (tens GB/s)

In CMS:

- One event per core, thanks to independency of events
- Memory footprint a issue

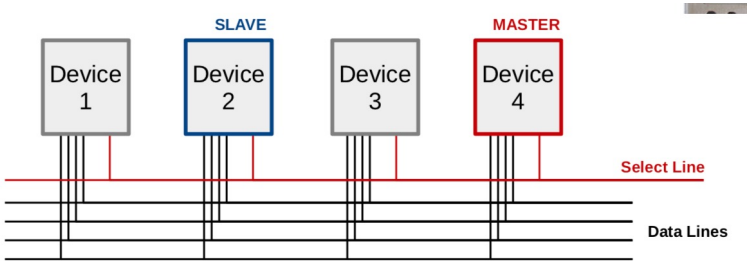
- Many Streaming Multiprocessors execute kernels (aka functions) using hundreds of threads concurrently
- High bandwidth to memory (up to 1TB/s)
- Number of threads in-fly increases with each generation
- In CMS:
  - unroll and offload each event's combinatorics to many threads in parallel



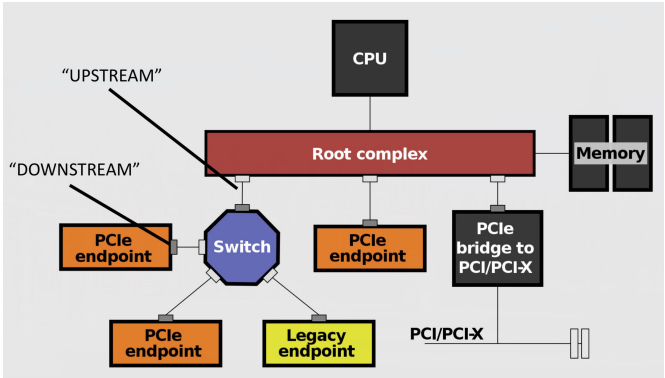
GPU

Back to DAQ

- In complex experiments, many channels are received by multiple electronic boards interconnected by a data **bus** – a computer may not be the most convenient form factor for this, we use **modular electronics**
  - In an architecture **similar to the one** discussed before – at least one particular element on the BUS is a CPU
  - It is common to **use the bus** (e.g. VME) to collect **data from multiple boards** in a single portion of memory



Parallel (e.g. VME)
Shared lines



Serial (e.g. PCIe)
Point-to-point connections

A note about parallel vs. serial

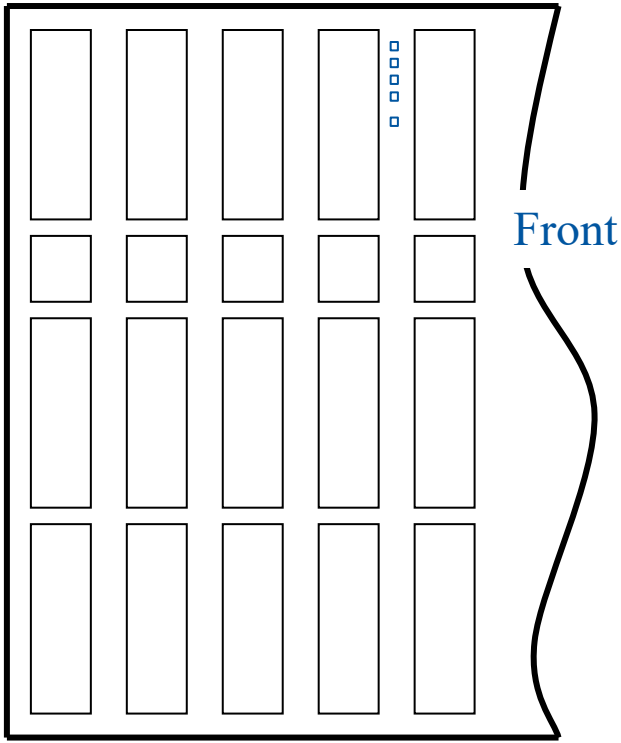
Parallel Buses Are Dead! (RT magazine, 2006)

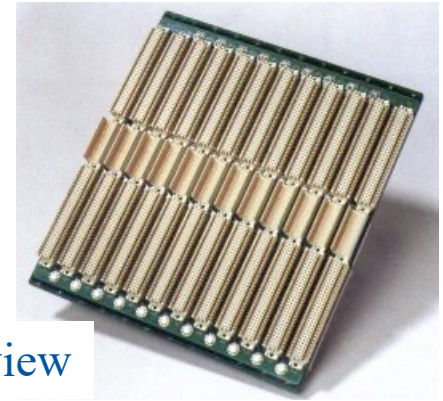
What is wrong about “parallel”?
- You need lots of pins on the chips and wires on the PCBs
- The skew between lines limits the maximum speed

What is wrong about “bus”?
- Speed is a function of the length (impedance) of the lines
- Communication is limited to one master/slave pair at a time (no scalability)
- The handshake may slow down the maximum speed
-

All parallel buses are dead. All? No!
- There is lots of legacy equipment
- VMEbus is still used heavily
  (military / research)

Parallel: VMEbus





Front view

Automatic 6U VME64x backplane

VME64x P1 Connector

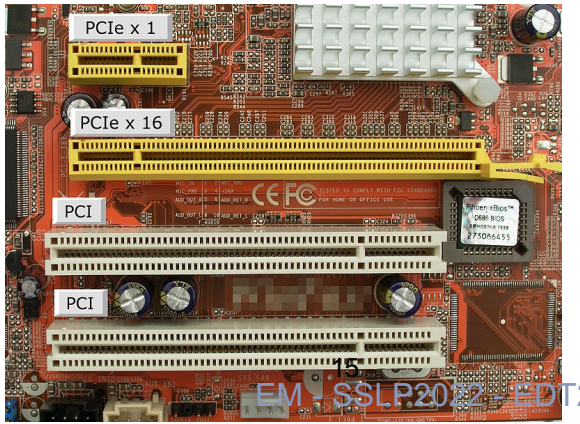
Pin	Signal Name Row z	Signal Name Row A	Signal Name Row B	Signal Name Row C	Signal Name Row d
1	MPR	D00	BBSY*	D08	VPC
2	GND	D01	BCLR*	D09	GND
3	MCLK	D02	ACFAIL*	D10	+1V
4	GND	D03	BG0IN*	D11	+V2
5	MSD	D04	BG0OUT*	D12	RsvU
6	GND	D05	BG1IN*	D13	-V1
7	MMD	D06	BG1OUT*	D14	-V2
8	GND	D07	BG2IN*	D15	RsvU
9	MCTL	GND	BG2OUT*	GND	GAP*
10	GND	SYSCLK	BG3IN*	SYSFAIL*	GA0
11	RESP*	GND	BG3OUT*	BERR*	GA1
12	GND	DS1*	BR0*	SYSRESET*	+3.3v
13	RsvBus	DS0*	BR1*	LWORD*	GA2*
14	GND	WRITE*	BR2*	AM5	+3.3V
15	RsvBus	GND	BR3*	A23	GA3*
16	GND	DTACK*	AM0	A22	+3.3V
17	RsvBus	GND	AM1	A21	GA4*
18	GND	AS*	AM2	A20	+3.3V
19	RsvBus	GND	AM3	A19	RsvBus
20	GND	IACK*	GND	A18	+3.3V
21	RsvBus	IACKIN*	SERCLK	A17	RsvBus
22	GND	IACKOUT*	SERDAT*	A16	+3.3V
23	RsvBus	AM4	GND	A15	RsvBus
24	GND	A07	IRQ7*	A14	+3.3V
25	RsvBus	A06	IRQ6*	A13	RsvBus
26	GND	A05	IRQ5*	A12	+3.3V
27	RsvBus	A04	IRQ4*	A11	LI/I*
28	GND	A03	IRQ3*	A10	+3.3V
29	RsvBus	A02	IRQ2*	A09	LI/O*
30	GND	A01	IRQ1*	A08	+3.3V
31	RsvBus	-12V	+5V Standby	+12V	GND
32	GND	+5V	+5v	+5V	VPC

www.interfacebus.com L. Davis

An example of **parallel** bus
Besides modular electronics, it was the base of Sun-2 computer arch. in the 80s

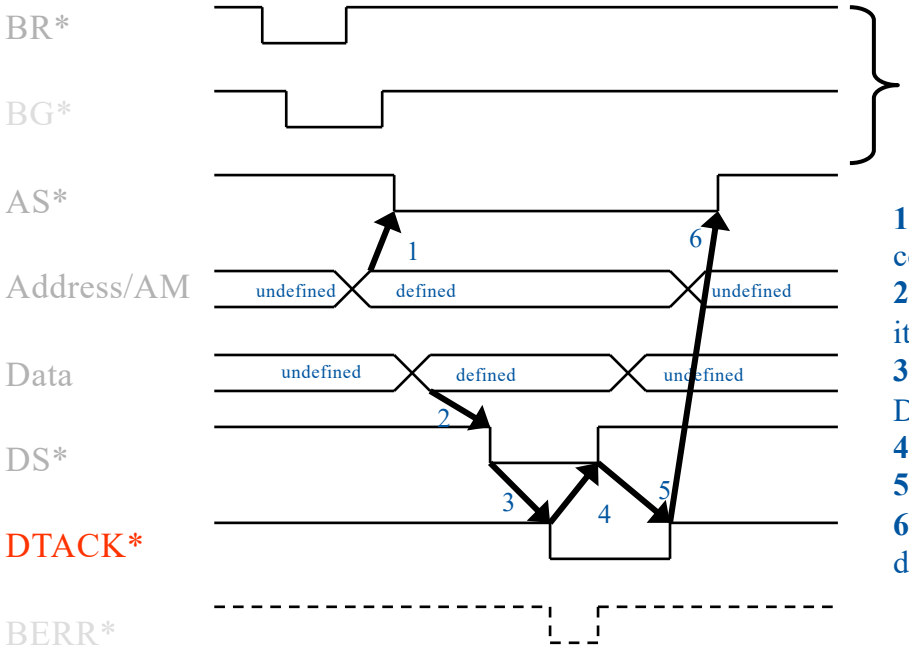
Serial: PCIe (aka PCI Express)

- Not a bus any more but a point-to-point link
- Data not transferred on parallel lines but on one or several serial lanes
  - Lane: One pair of LVDS lines per direction
  - Clock rate: 2.5 GHz (PCIe2.0: 5 GHz, PCIe 3.0: 8 GHz, PCIe 4.0: 16 GHz)
  - 8b/10b encoding (from PCIe3.0: 128/130b encoding)
  - 250 MB/s (PCIe 1.0) raw transfer rate per lane
  - Devices can support up to 32 lanes



Example: VME write

Example: (Simplified) write cycle

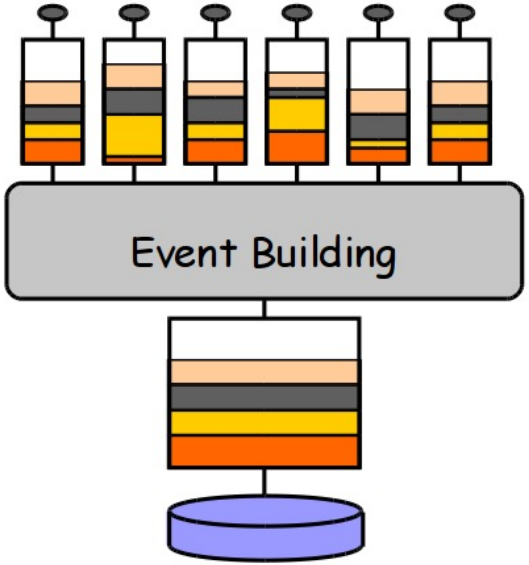


1: Master drives address and AM code. Then it asserts AS
2: Master puts data on the bus. Then it asserts DS
3: Slave latches data and drives DTACK
4: Master removes DS
5: Slave removes DTACK
6: Master releases Address, AM and data lines. Then it releases AS

Color code: Master - Slave - Arbiter

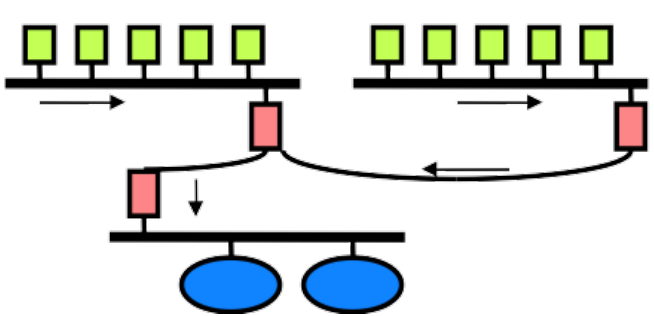
Event Building

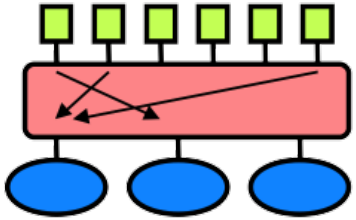
- In large experiments consisting of many different sub-detectors, read out is performed by different boards and by multiple CPUs (for example in multiple VME crates)
  - We want to combine all the portions corresponding to the same “event” in the memory of a single CPU for processing and eventually storage
- Need a mechanism to associate all data corresponding to the same event (e.g. a bunch crossing in a collider, a gamma ray shower in a telescope array…)
  - This process is called **event building**

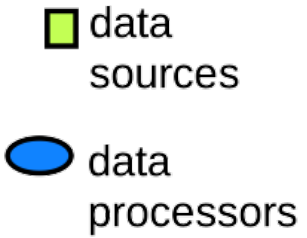


Event Building

- Event Building used to be performed on the bus itself

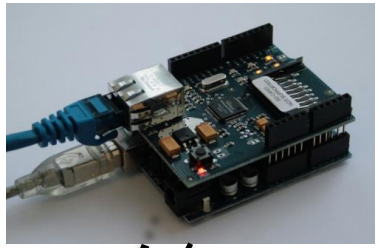


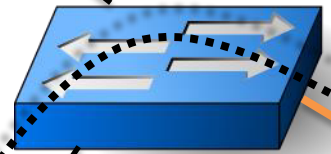




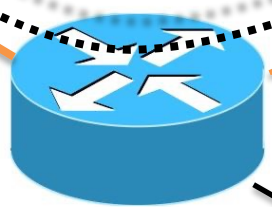
  - But the bus forces the process to be sequential (only one board can “speak” at a time)
  - It is also not infinitely extendible (does not “scale”)
- In all LHC experiments event building is performed by distributed processes through a **switched network**

Networks, switches

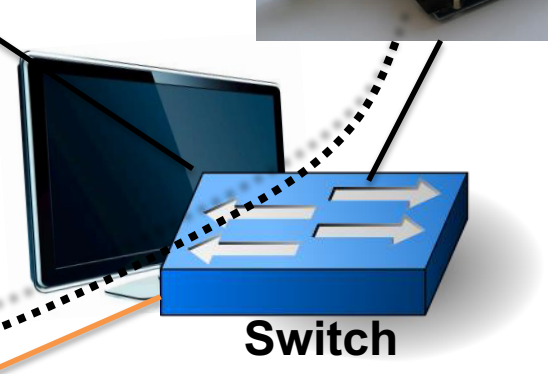




Switch



Router



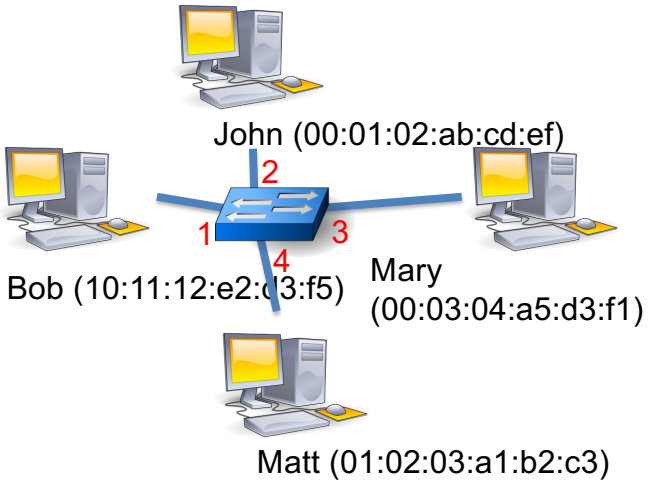
Switch

Wireless AP

Ethernet switch

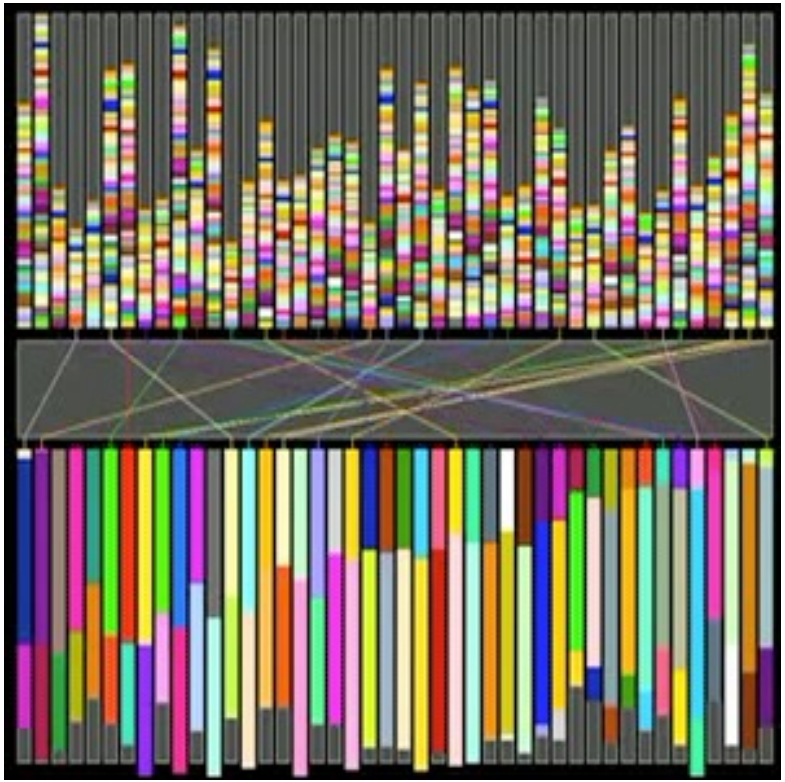
Layer-2 device

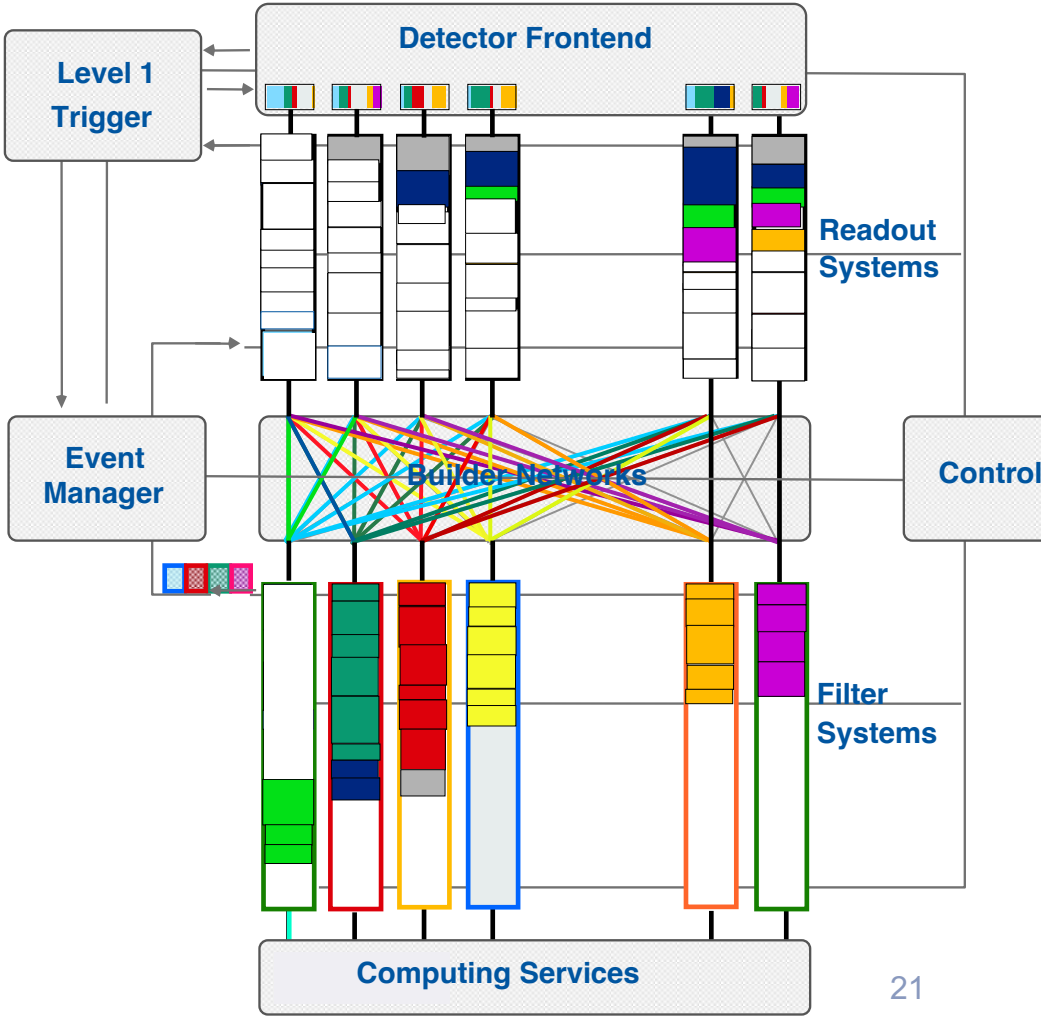
- Switches frames to their destination using the MAC address
- Learns the address associated to each port and stores it in a table



Port	MAC Address
2	00:01:02:ab:cd:ef(John)
4	01:02:03:a1:b2:c3(Matt)
….	…..

Event Building





More in-depth – menu

Back to a real-life (simple) experiment – measuring energy

- Charge to digital converter
- ADC non-linearity

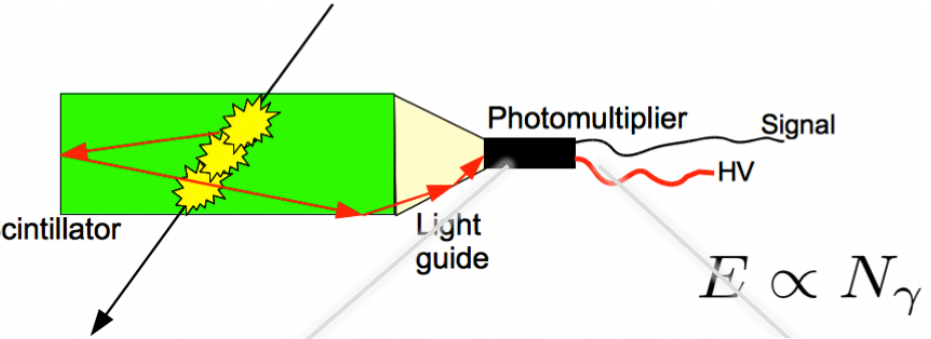
Measuring position with a TDC

Calibration

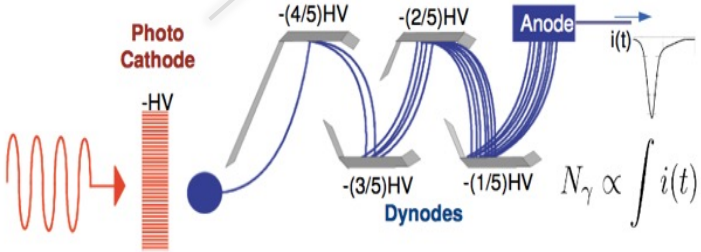
Trigger, latency and deadtime, trigger efficiency

Examples taken from nuclear and (mostly) particle physics

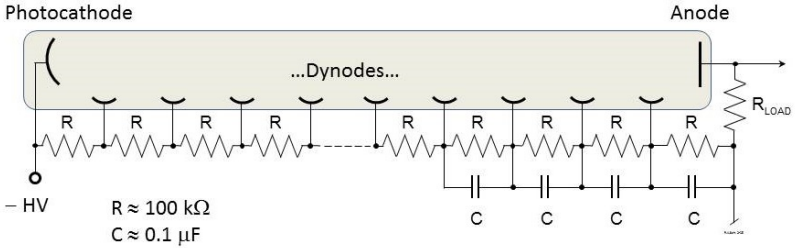
measuring energy with scintillator+PMT



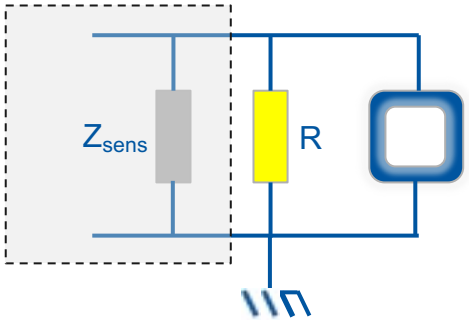
$$E \propto N_\gamma$$



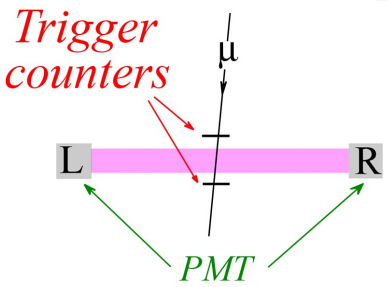
$$N_\gamma \propto \int i(t)\,dt = Q$$

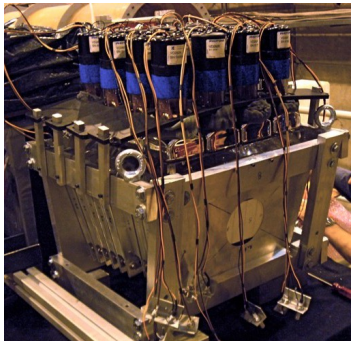


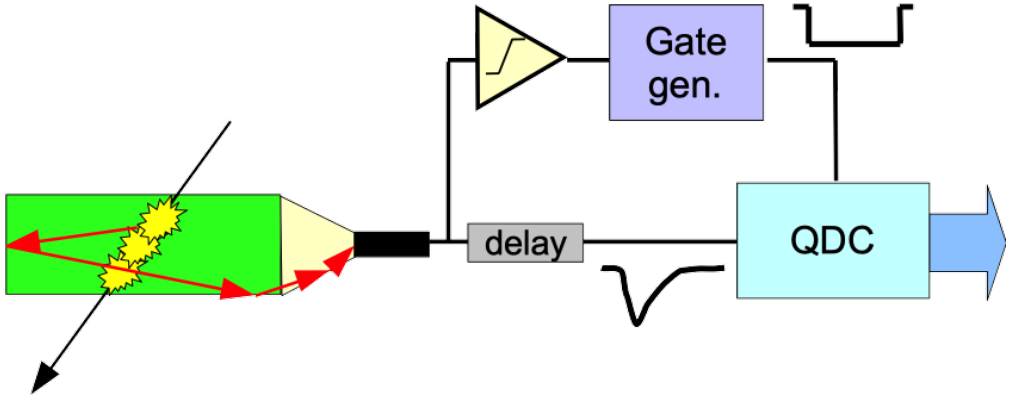
Z_{inst}



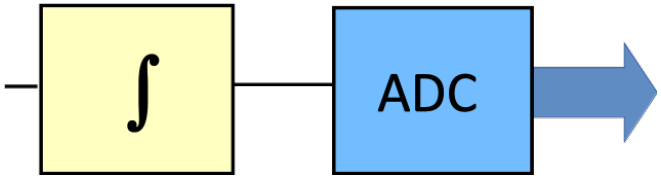
Real ADCs at work



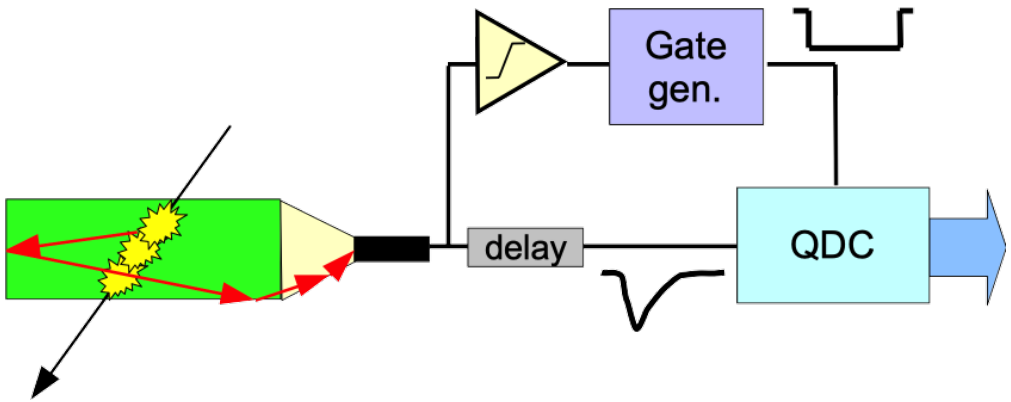




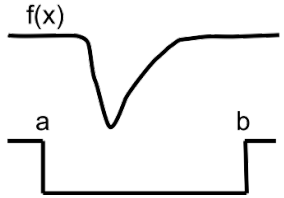
- Real data from a beam test @CERN
- PbWO4 (scintillating) crystal equipped with two PMTs and exposed to e, μ and π beams



- QDC: (gated) charge integrator followed by ADC
  - (fixed-duration) gate opened after trigger
  - digital conversion started after gate closing
  - integrator discharged after conversion



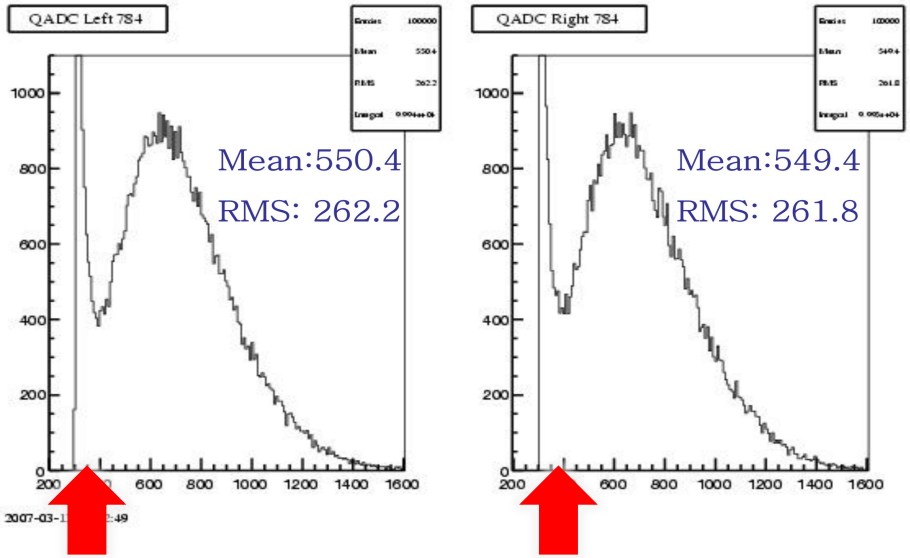
- QDC → Charge to Digital Converter
- Essentially an integration step followed by an ADC
- Integration requires limits → gate



- Relative timing between signal and gate is important: delay tuning
- Gate should be large enough to contain the full pulse and to accommodate for the **jitter**
- Gate should not be too large → increases the noise level
- Gate should not be too large → **the system is dead to further signal**

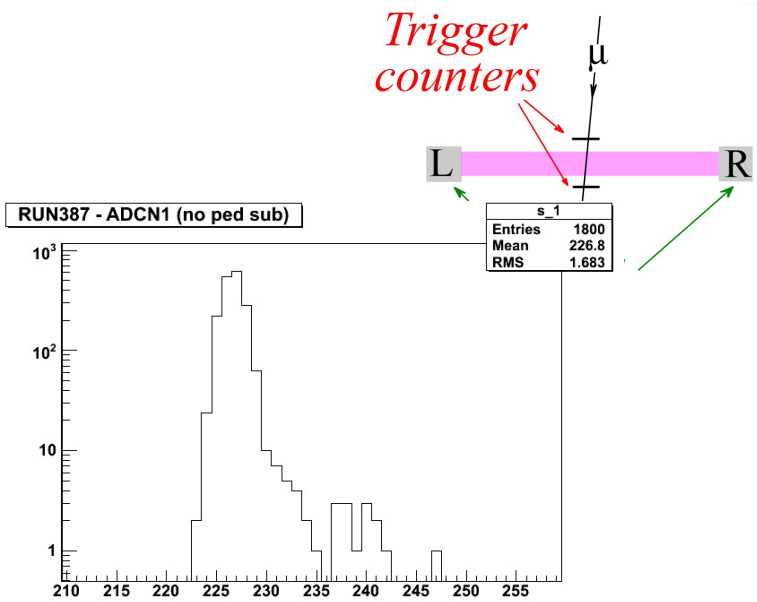
By the way, which are the noise contribution to our charge measurement?

Pedestal subtraction



The result of a pedestal measurement has to be subtracted from our charge measurements

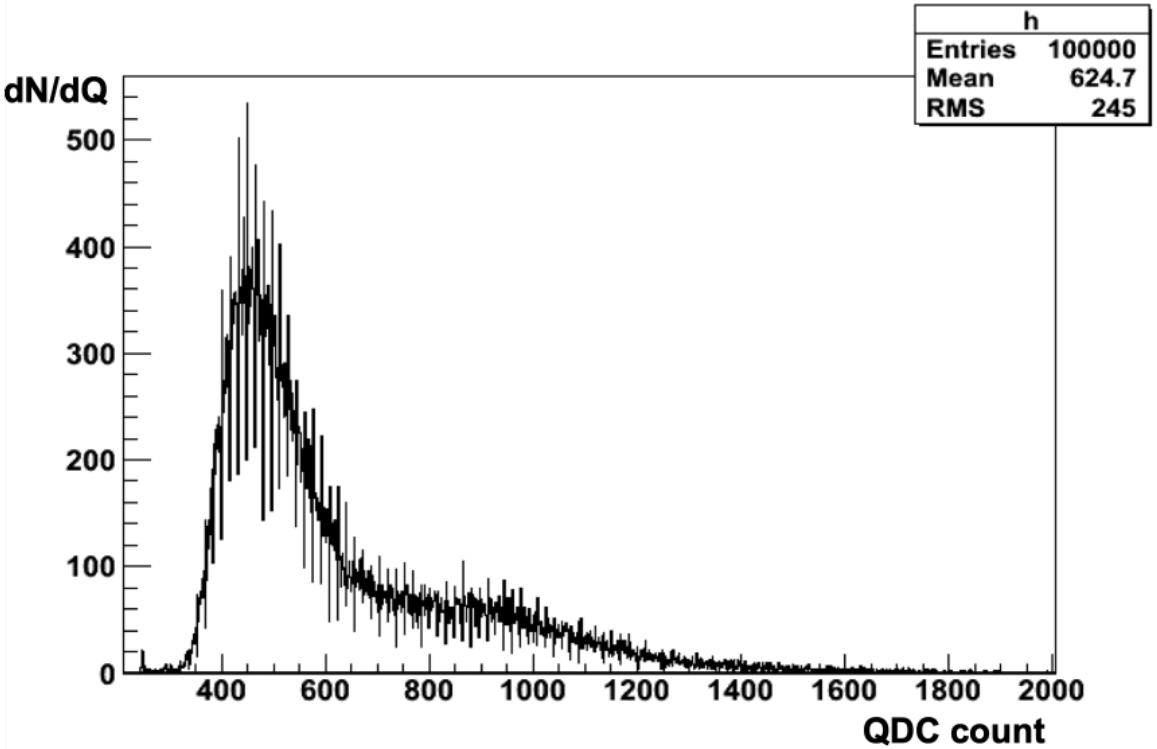
The pedestal is produced essentially by PMT dark current (peak corresponding to zero photoelectrons), thermal noise, …out-of-time particles

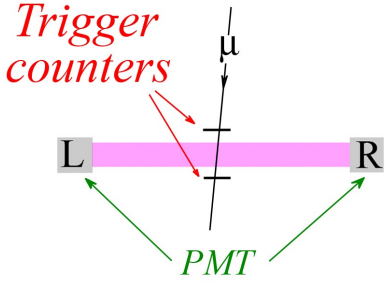


With an out-of-phase trigger we can measure the baseline (pedestal)
The same noise enters our physics measurements and contributes with an offset to the distribution

NOTA BENE: AFTER PEDESTAL SUBTRACTION, THE EFFECTIVE (USABLE) DYNAMIC RANGE IS REDUCED

Read QDC data

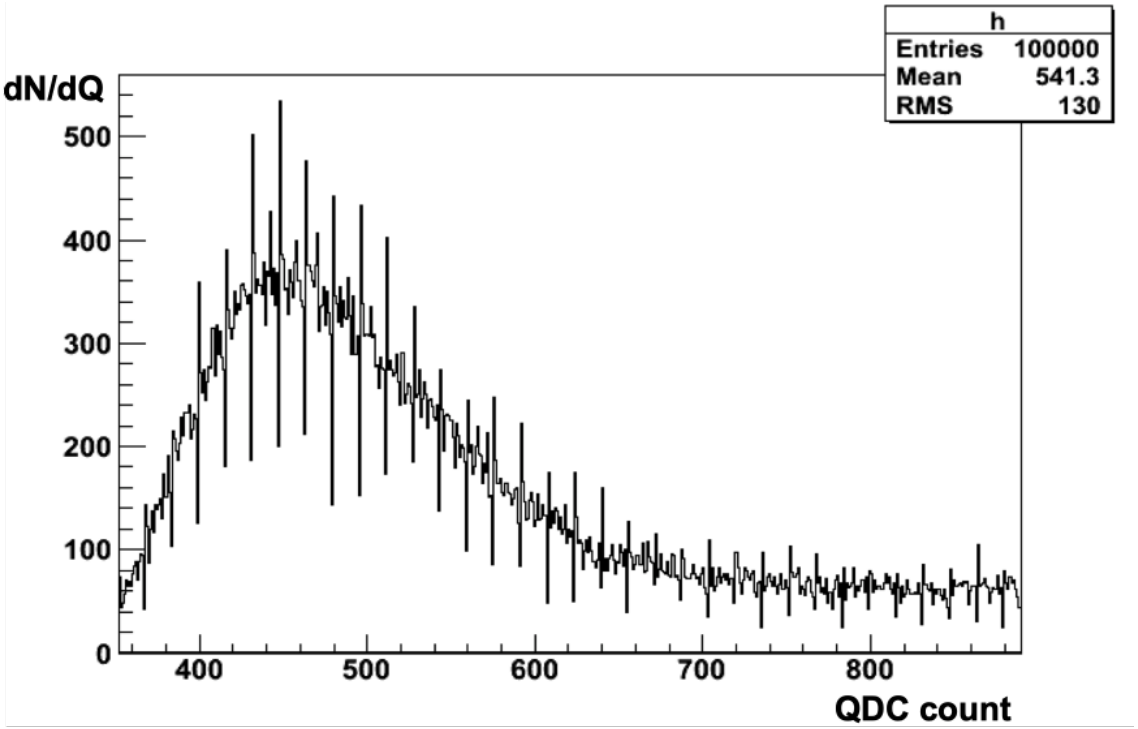


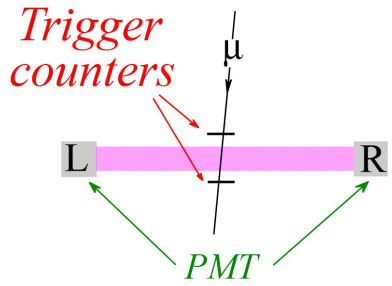


Nice pion-beam charge-distribution for one PMT

But, what are all those little peaks? Just statistical fluctuations? Let's zoom in!

Real QDC data





- Bin with N entries should fluctuate with $\sigma=\sqrt{N}$

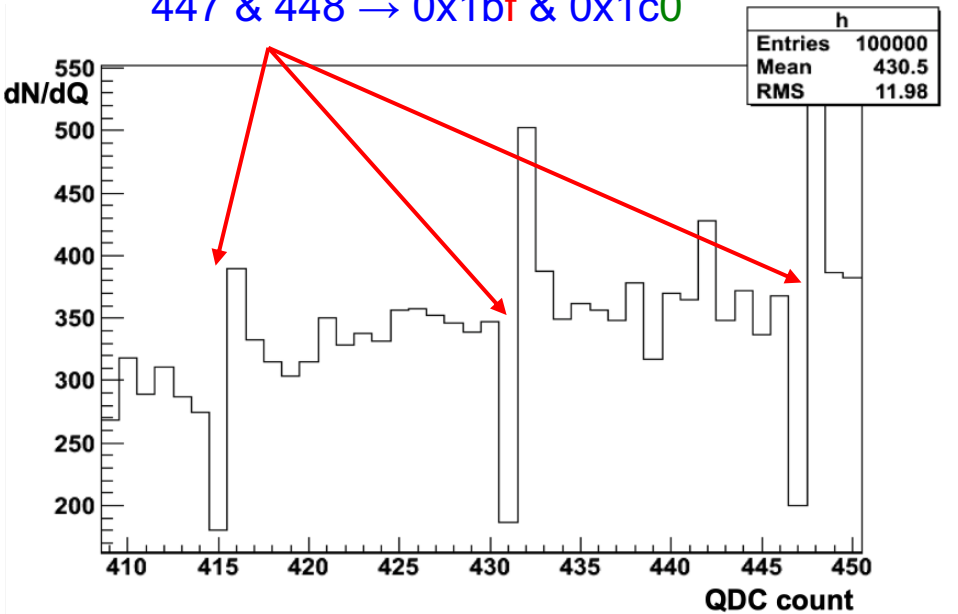
  $\sqrt{360}\sim19 \rightarrow ()\sim10\ \sigma$

- Spikes are regularly distributed
- Some systematic effect must be taking place

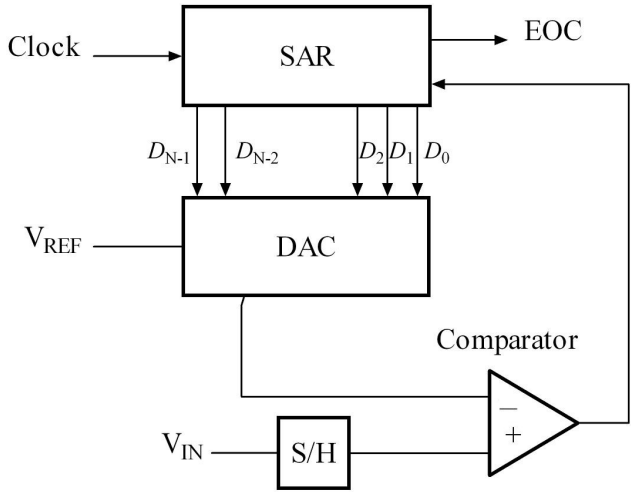
- Zoom in a bit more

Real QDC data

415 & 416 → 0x19f & 0x1a0
431 & 432 → 0x1af & 0x1b0
447 & 448 → 0x1bf & 0x1c0

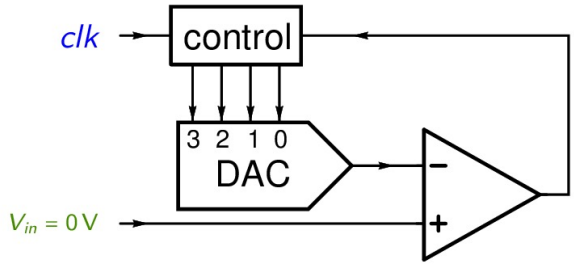


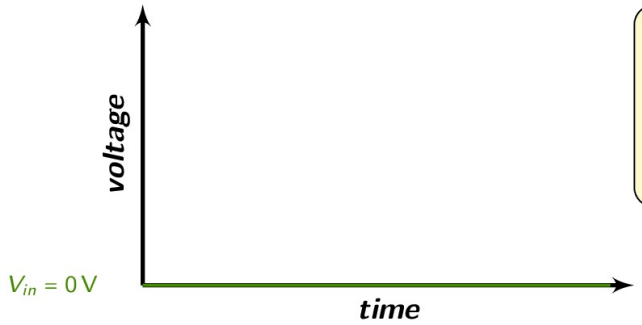
- Results of type xxx0 win over xxxf
- Typical differential non linearity of successive approximation ADCs



Successive approximation ADC

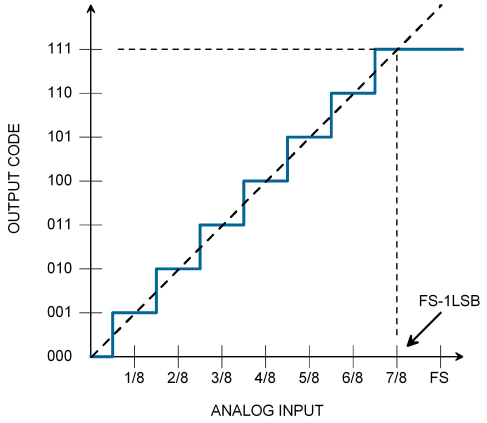
Successive Approximation – example of a 4-bit ADC

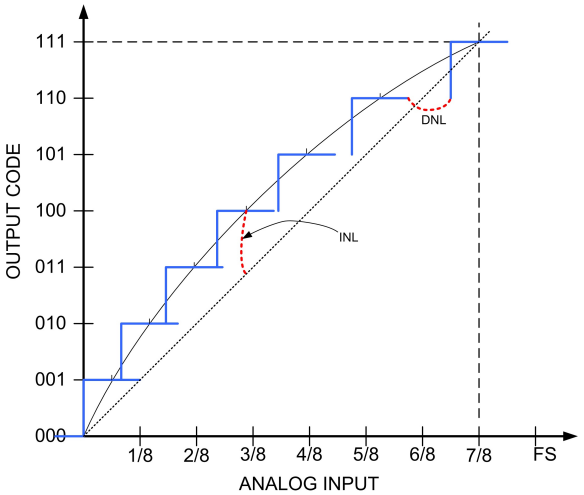


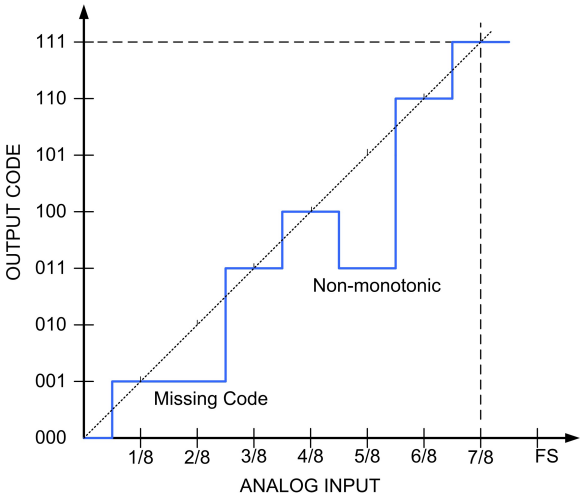


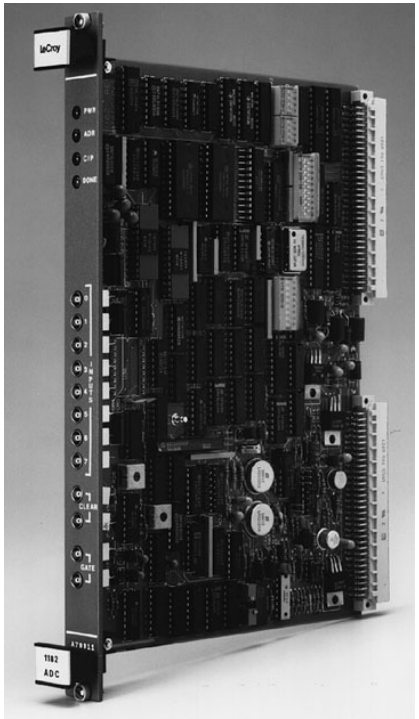
Resolution:	
5 V × 1/2	2.5000 V
5 V × 1/4	1.2500 V
5 V × 1/8	0.6250 V
5 V × 1/16	0.3125 V
...	
5 V × 1/1024	0.0049 V

ADC non-linearities





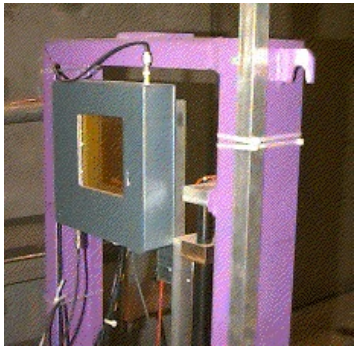




- High Density, 8 Channels
- High Sensitivity, 50 fC/count
- Wide Dynamic Range, 12 Bits
- Flexible Gates, 50 nsec to 2 µsec
- Fast Clear, 650 nsec
- Fast Readout, 5 Megawords/sec
- Short Conversion Time, 16 µsec
- 16 Event Buffer Memory
- Single-Width 6 U, A24/D16 VME Slave

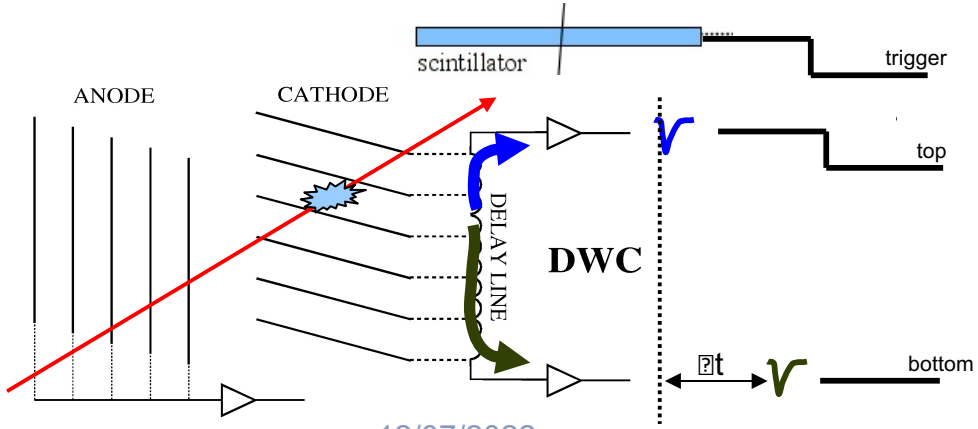
http://teledynelecroy.com/lrs/dsheets/1182.htm

Real life example: TDC

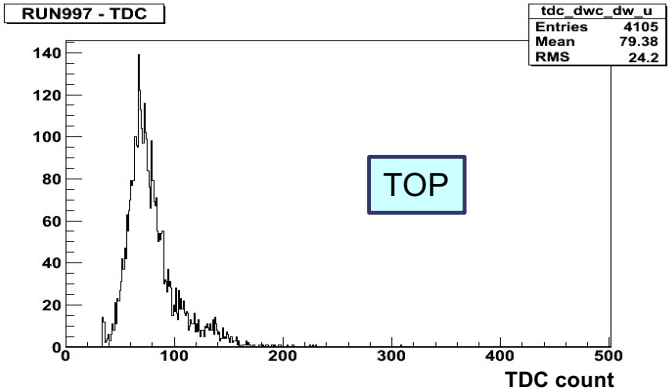


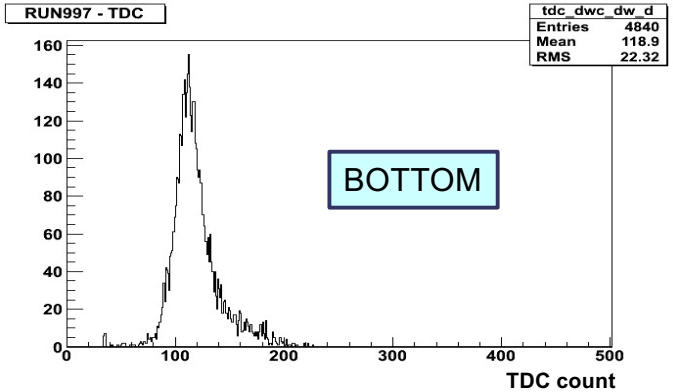
- XDWC: delay wire chambers used on the SPS extracted lines to measure beam profiles
- Two cathode planes provide X and Y positions
- Measurement is based on the delay gained along a delay line

$$y = \alpha \cdot \Delta t + \beta = \alpha \cdot \left(t_{top} - t_{bottom}\right) + \beta$$

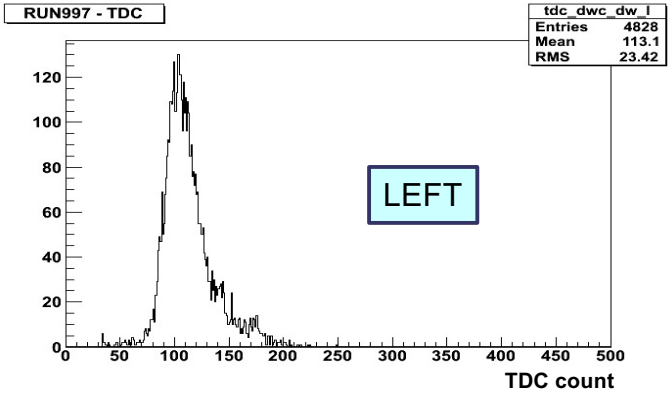


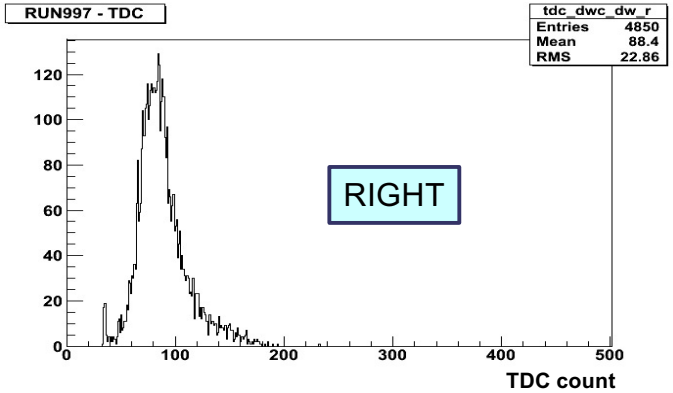
Raw time data



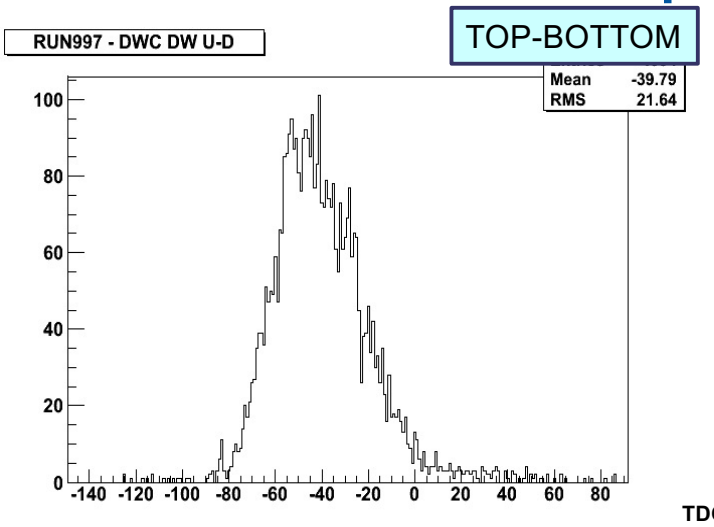


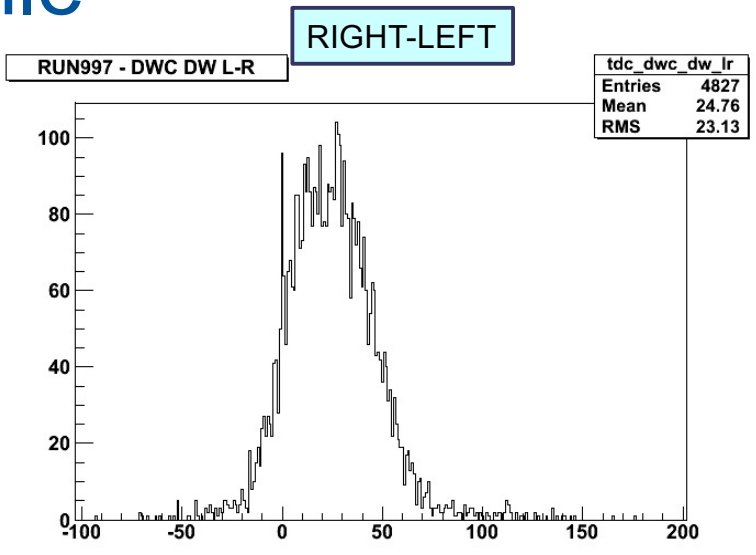
Individual channel distributions





Uncalibrated beam profile





- Beam sizes are still in TDC counts
  - Not very useful, though
- How do we convert this into a known scale (e.g. cm)?

- We have our ADC/TDC counts, but how do we correlate them with energy/position ?
- Need a calibration procedure

Calibration

- The experiments we discussed provide relative measurements. The values obtained via our system are in some (known) relation with the interesting quantity
  - Scintillator

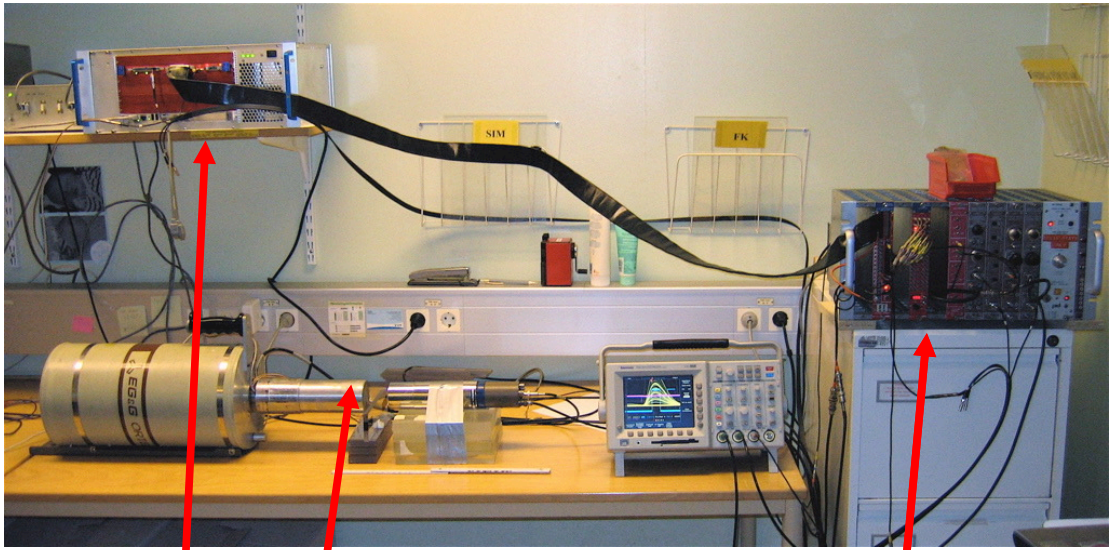
$$Q \propto N_\gamma \propto E$$

  - XDWC

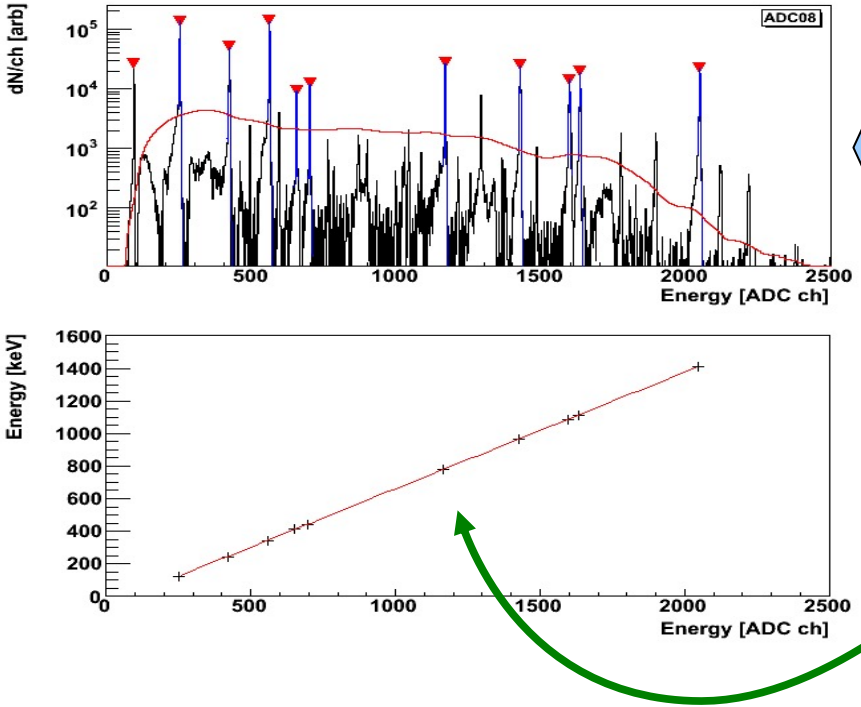
$$y = \alpha \cdot \Delta t + \beta = \alpha \cdot (t_{top} - t_{bottom}) + \beta$$

- Our instruments need to be in order to give us the answer we are looking for
  - We have to determine the **calibrated**parameters that transform the raw data into a physics quantity
  - The parameters normally depend on the experimental setup (e.g. cable length, delay settings, HV settings, …)
- In the design of our detector and DAQ we have to foresee calibration mechanisms/procedures

Ge crystal for isotope identification



Crystal calibration

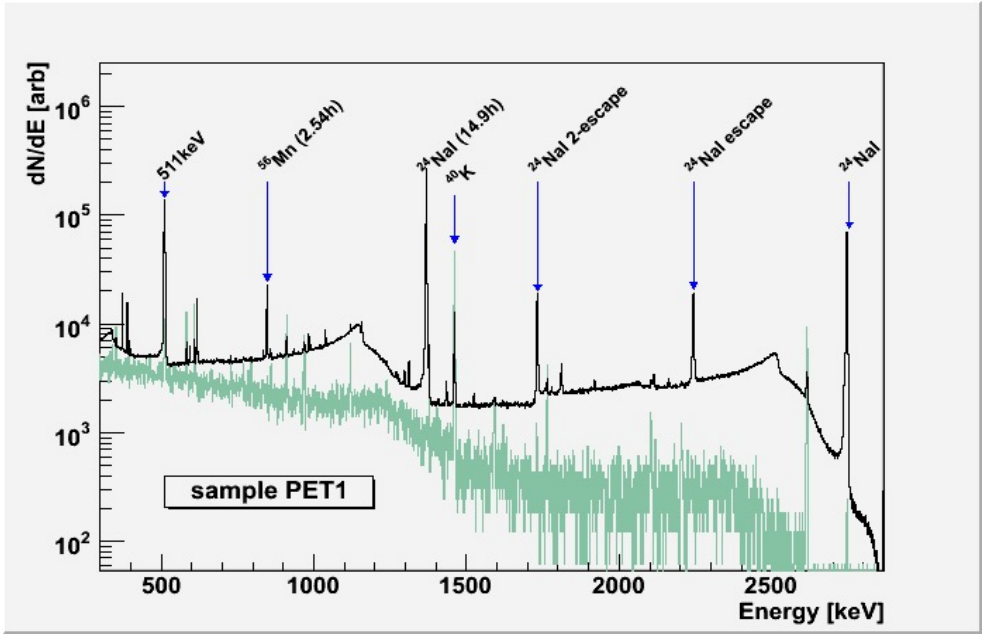


- ^{152}Europium reference source
  - Known γ emission lines
  - Peak find & fit
- Allow for definition of the parameters describing functional relation between ADC count and E

$$Q \propto N_\gamma \propto E$$

- Reality is not perfectly linear
  - Polynomial fits
  - Physical models

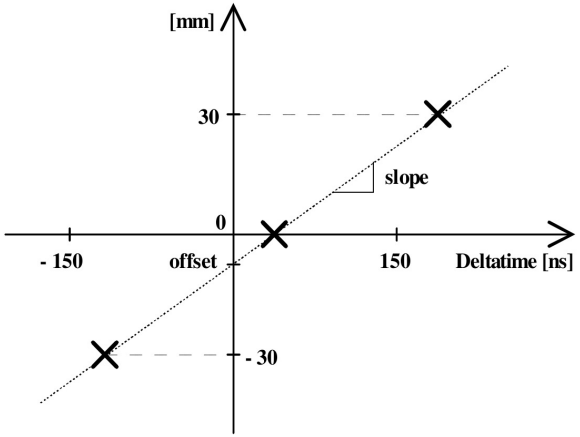
Isotope identification



Calibrated crystal setup can be used to identify isotopes generated in irradiated samples

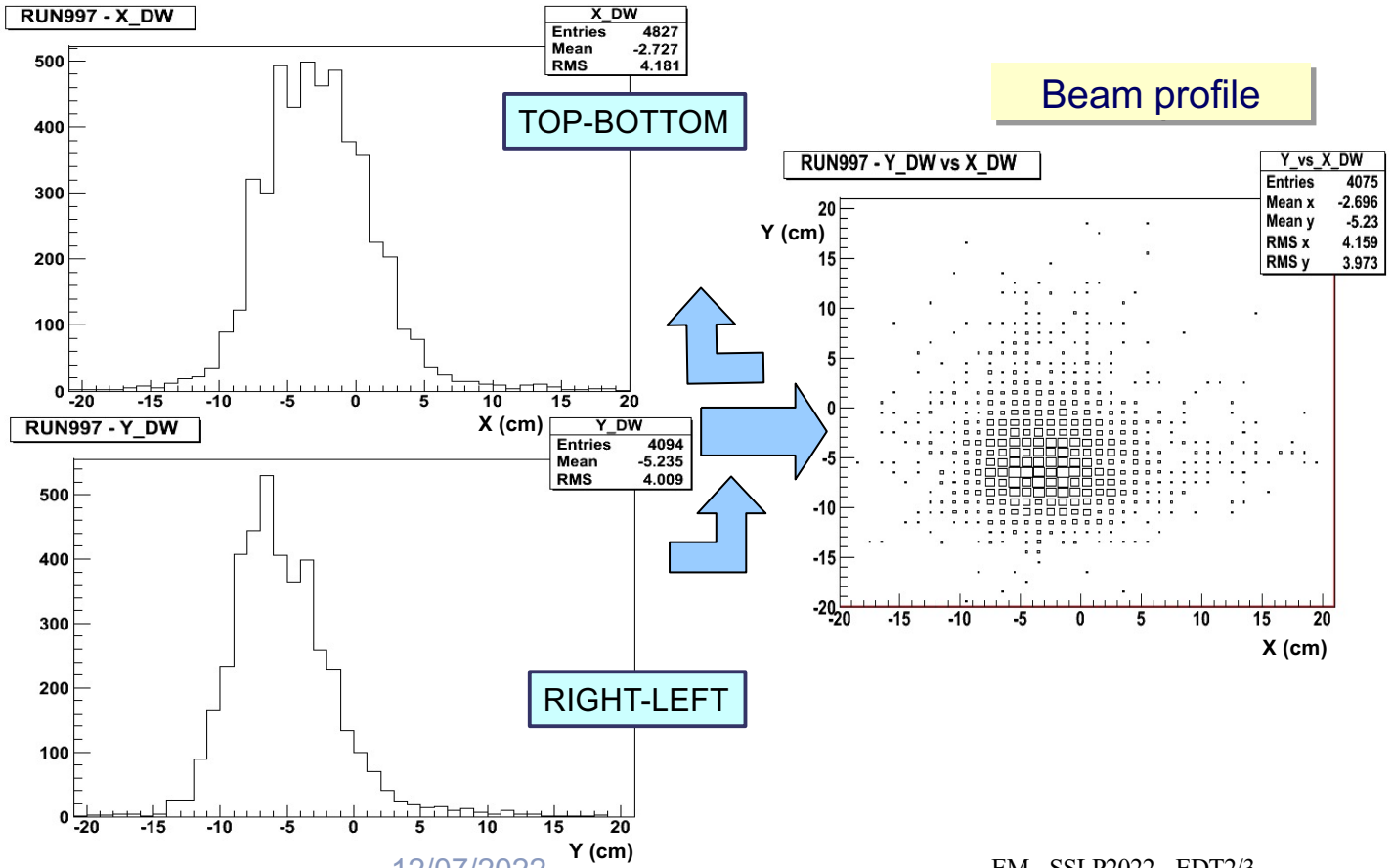
XDWC Calibration

- XDWC chamber have 3 calibration inputs that allow for independent calibrations of X and Y axis with only 3 different sets of data
- The calibration input simulate signals from particles respectively hitting
  - Right-top corner (X=Y=30mm)
  - Center (X=Y=0mm)
  - Left-bottom corner (X=Y=-30mm)
- The calibration data sets are collected with final setup and TDC
- Interpolating the three points in the *t-x* space, the parameters of the calibration equation can be measured



$$x = \alpha t^* + \beta$$

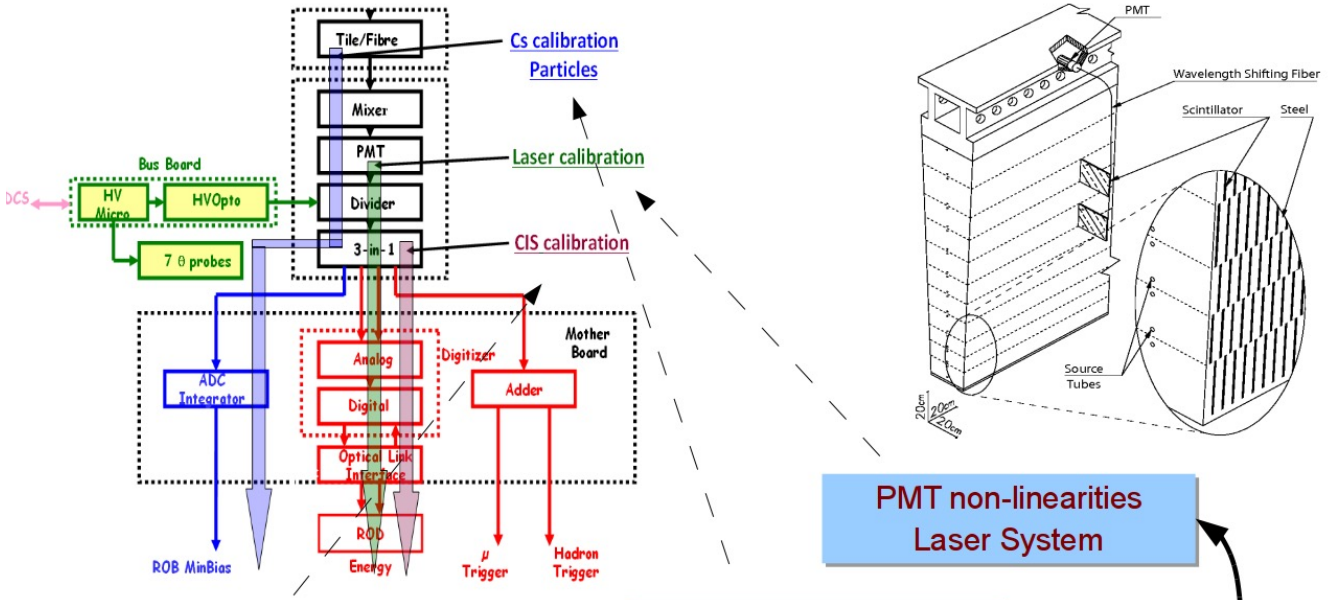
Calibrated XDWC



Calibration: summary

- Our DAQ chain provides us with numbers (raw data) but … what do they really mean ?
- Likely related with physics quantities but with relations (transfer functions) affected by several uncertainties:
- due to physical detection mechanisms
- due to signal processing
- Transfer functions usually parametrised, sometimes based on (look-up) numeric tables
- All system elements need to be calibrated to keep optimal knowledge of all parameters:
  - calibration procedures
  - calibration constants
- Calibration constants change with ageing (mainly due to radiation), beam conditions (electronics may have baseline drifting with pile-up), time … HV, LV, ...
- The design of our detector and DAQ has to foresee calibration mechanisms/procedures
- injection of known signals
- dedicated calibration *triggers* and data streams

Atlas tile calorimeter calibration system

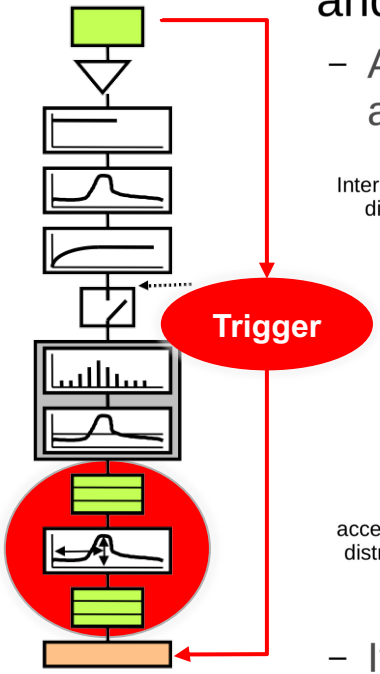


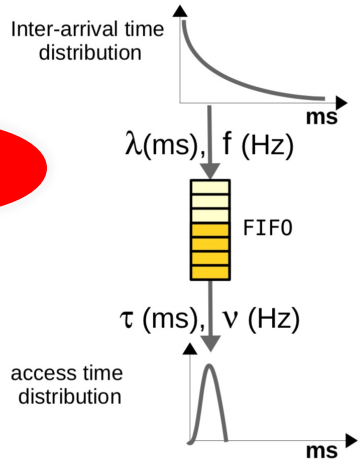
$$E_{channel} = A \cdot C_{ADC \to pC} \cdot C_{pC \to GeV} \cdot C_{Cs} \cdot C_{laser}$$

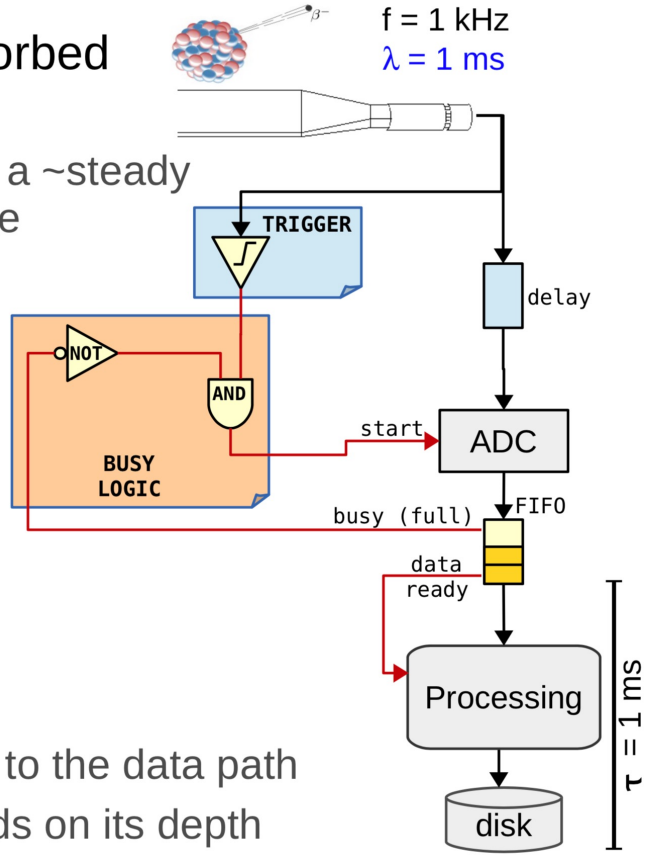
Back to Trigger

Derandomization

- Input fluctuations can be absorbed and smoothed by a queue
  - A First In First Out can provide a ~steady and **de-randomized** output rate

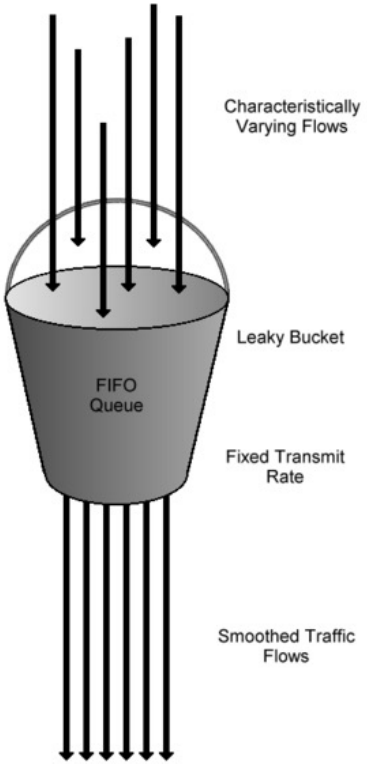






  - It introduces additional latency to the data path
  - The effect of the queue depends on its depth

- Leaky-bucket problem
  - The bucket is filled at variable flow
  - The bucket empties at constant (smooth) flow
  - How often does it overflow ?



Buffer size and deadtime

N-event buffer ... single queue size N, service time τ, service rate – $1/\tau$:

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

$\sum P_k = 1$ [k=0..N]
rate [$j \rightarrow j+1$] = $f \cdot P_j$ (fill at rate *f*)
rate [$j+1 \rightarrow j$] = $\mu \cdot P_{j+1}$ (empty at rate $\mu = 1/\tau > f$)
steady state: $\mu \cdot P_{j+1} = f \cdot P_j \Rightarrow P_{j+1} = \rho \cdot P_j = \rho^{j+1} \cdot P_0$ [$\rho = (f\,\tau) <\sim 1$]

for $\rho \sim 1 \Rightarrow P_j \sim P_{j+1} \Rightarrow \sum P_k \sim (N+1) \cdot P_0 = 1 \Rightarrow P_0 \sim P_N \sim 1/(N+1)$
$\Rightarrow$ dead time ~ 1/(N+1)
want ~ 1% $\Rightarrow$ N ~ 100

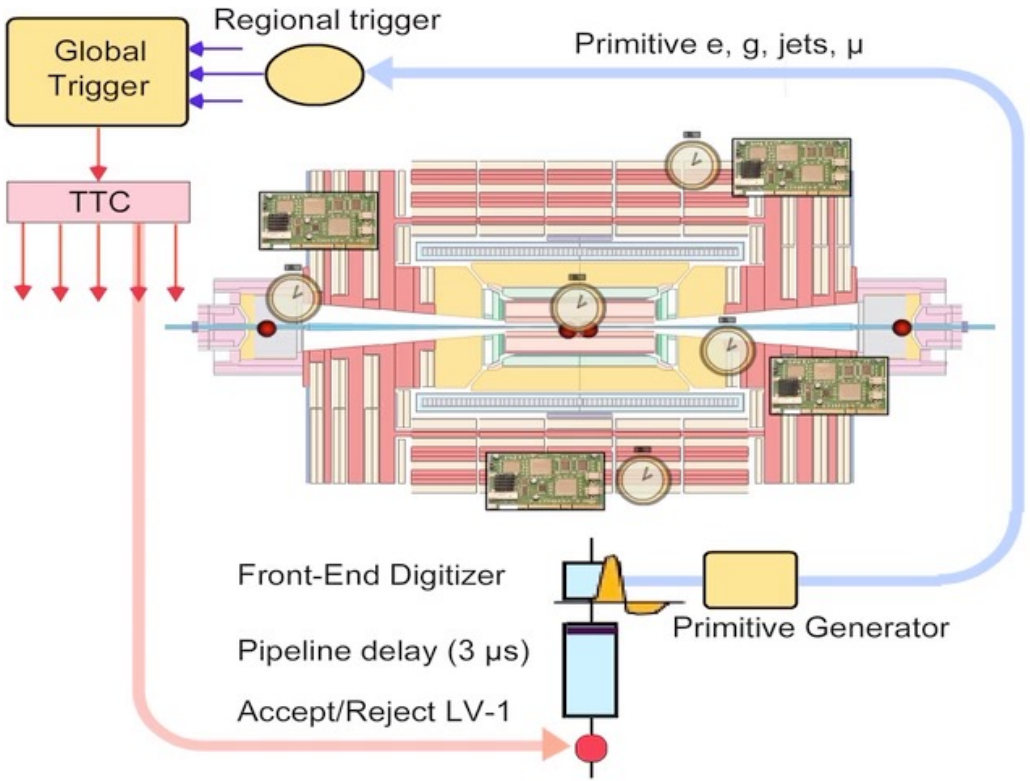
Deadtime: summary

- Deadtime from overlapping readout windows
- Deadtime **to avoid buffer overflow**
  - What happens in case of buffer overflow ?
- Strategies to limit deadtime
  - Size buffers conveniently
    - For collider experiments: f = bunch crossing frequence (40 MHz for LHC), τ = latency **of trigger system** + readout time
  - Suppress trigger “bursts” = “trigger rules”
  - For example not more than M trigger in any (sliding) window of N triggers
    - Trigger rules are a source of deadtime as well (but avoid buffer overruns)
  - Have electronics raise **“almost full” signals** used to **throttle** the trigger

Latency

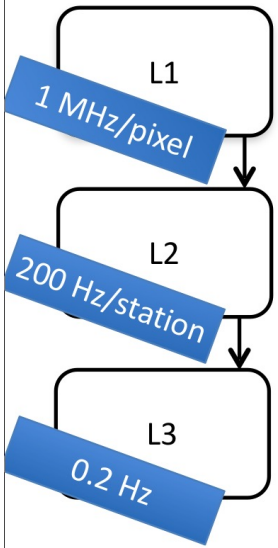
- A synchronous (real time) system must respond within some fixed delay → Latency = Max Latency (constrained) → over fluctuations bad, will create dead time
- An asynchronous system responds as soon as it's available → Latency = Mean Latency (not constrained) → over fluctuations ok, shouldn't create dead time (unless upstream buffers get full)
- Trigger systems for complex experiments are often implemented in multiple levels
- Usually only the first one or two are synchronous
- Synchronous triggers have latencies of the order of μs
  - Front-end buffers e.g. at LHC must be able to hold data for at least order of 40 MHz * 1-5 μs = 40-200 bunch crossings
  - The latency includes the time to process the input, generate the trigger decision, and propagate the trigger signal back to the front-end

Latency, pipeline, trigger distribution



Example: Augier

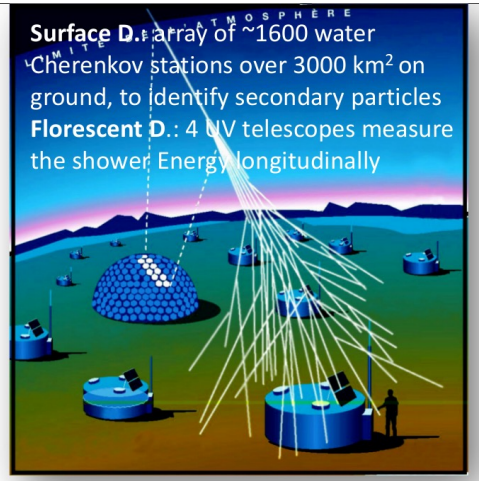
- **Detect air showers generated by cosmic rays above 10^{17} eV**
  - Expected rate < 1/km²/century. Two large area detectors
- On each detector, a 3-level trigger operates at a wide range of primary energies, for both vertical and very inclined showers

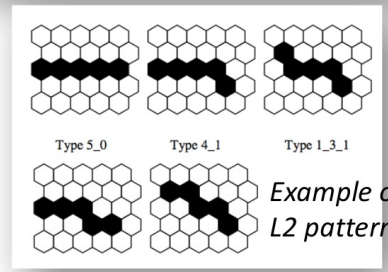


L1: (local) decides the pixel status (on/off)
- ADC counts > **threshold**
- ADC digitizes any 100 ns (**time** resolution)
- ADC values stored for 100 μs in **buffers**
- **Synchronized** with a signal from a GPS clock

L2: (local) identifies track segments
- Geometrical criteria with recognition algorithms on programmable patterns

L3: (central) makes spatial and temporal correlation between L2 triggers

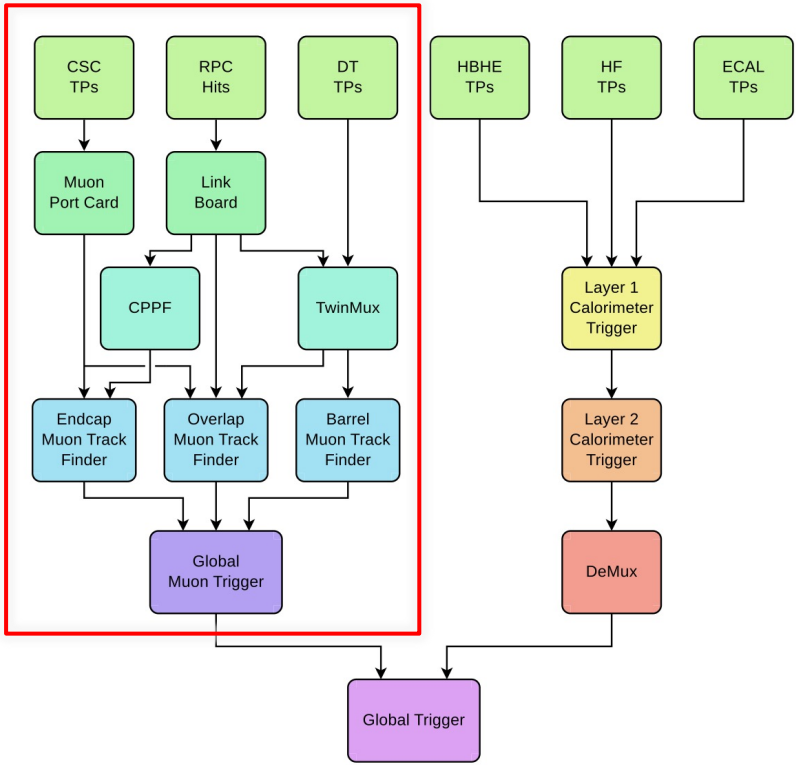


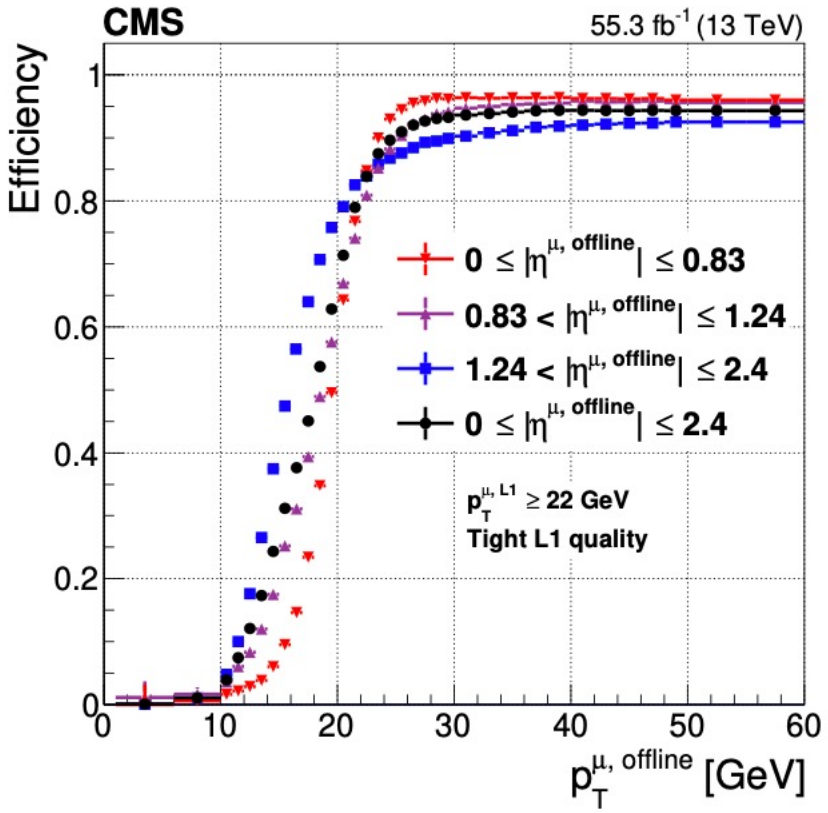


Example of L2 patterns

One event ~ 1MB → 0.2 MB/s bandwidth needed for the DAQ system

Example: CMS muon trigger





Understanding trigger efficiency

$$BR(\ Signal\ ) = \frac{(N_{candidates} - N_{bg})}{\alpha \cdot \varepsilon_{total} \cdot \sigma_{Bs} \cdot \int Ldt}$$

$$\alpha \cdot \varepsilon_{total} = \alpha \cdot \varepsilon_{Tracking} \cdot \varepsilon_{Reco} \cdot \varepsilon_{L1\text{-}Trig} \cdot \varepsilon_{L2\text{-}Trig} \cdot \varepsilon_{L3\text{-}Trig} \cdot \varepsilon_{vertex} \cdot \varepsilon_{analysis}$$

- Must be precisely known (w/ its systematics)
- Independent trigger selections allows cross-calibration

→ need of additional triggers

High-Level Triggers: → pass-through triggers (release selection criteria)

Level-1 Triggers: a) zero or minimum bias samples b) Tag&Probe → trigger on “Tag” and measure “Probe”

Additional topics

More Queueing Theory I

Semi-qualitative discussion:

Transition where queue occupancy decreases by one $(j \rightarrow j+1) = fP_j$

Transition where queue occupancy increases by one $(j+1 \rightarrow j) = P_{j+1}/\tau$

At equilibrium (i.e. distribution in queue is stable):

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

One can then conclude that $P_1 = x \cdot P_0, P_2 = x^2 \cdot P_0, \ldots P_N = x^N \cdot P_0, (x = f \cdot \tau)$

More Queueing Theory II

$$\sum_k^N P_k = 1 \rightarrow P\_0 = 1/\sum_k^N x^k = (1-x)/(1-x^{N+1})$$

incidentally notice that: $< k > = \lim_{N\to\infty} \sum_k^N kP_k = \sum_k^N kx^k \frac{1-x}{(1-x^{N+1})}$

$$\lim_{N\to\infty} \frac{x-1}{(x^{N+1}-1)} \sum_k^N kx^k = 1 - x]_{x<1} \lim_{N\to\infty} \sum_k^N kx^k = (x-1)\frac{x}{x-1}\Big|_{x<1} = f\tau$$

which is a not-so-formal proof of Little's theorem…
For N finite…

$$P_N = x^N \cdot (1-x)/(1-x^{N+1})$$

$$\varepsilon_N = (1-P_N) = (1-x^N)/(1-x^{N+1})$$

$$(x<1): \lim_{N\to\infty} \varepsilon_N = 1$$

$$(x>1): \lim_{N\to\infty} \varepsilon_N = 1/x$$

Combinatorial Logic vs Sequential Logic

Combinational circuits

the steady-state outputs are logical functions of the steady-state inputs only,

→ outputs do not depend on the internal condition of the circuit.

→ combinational circuits can always be constructed with gates.

Examples: - adding or comparing integers

- deciding whether all the conditions for a decision are satisfied

Sequential circuits

when memory of past actions is required → combinational circuits do not suffice.

Example: if we want to detect the second occurrence of a given action, we must somehow register the fact that at some point there was a first occurrence of this action

Synchronous vs Asynchronous

- Synchronous sequential logic:
  - the time at which transitions between circuit states occurs is controlled by a **common clock** signal
  - changes in all variables occur simultaneously
- Asynchronous sequential logic:
  - state transitions occur independently of any clock, and normally depend on the time at which input variables change
  - outputs do not necessarily change simultaneously
- **Clock**
  - a clock signal is a square wave of a fixed frequency
  - it is used to trigger state transitions at fixed times in synchronous circuits

FLIP-FLOPs (and Latches)

A flip-flop is a memory device with two stable states:
→ by an appropriate choice of inputs, either state can be obtained

Flip-flops are needed to execute sequential logic, which requires memory of past actions

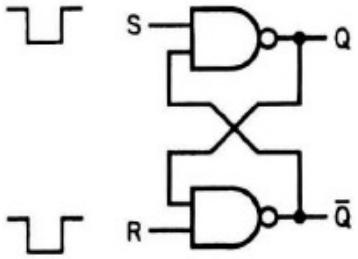
Flip-flops come in many forms:
- we only focus on those obtained by cross-coupling logic gates
- we will assume that all gates have the same gate delay t

Set-Reset FLIP-FLOPs - asyncronous

Consider the asynchronous
RS NAND flip-flop

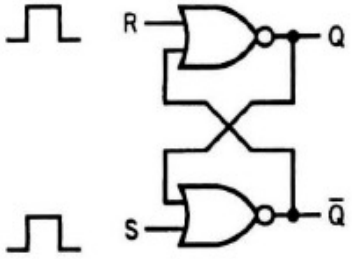
If the S(set) and R(reset)
inputs are inactive (R = S = 1)
→ the outputs Q and Qbar can be in
one of two self-consistent states:
either Q = 0 and Qbar = 1
or Q = 1 and Qbar = 0

active low



(a) RS NAND flip-flop

active high



(b) RS NOR flip-flop

- If S is made active (ie set to 0) → Q will become 1 after a time t

(if it was not already 1) and Qbar will become 0 after a time 2t
After this time, Q will remain at 1 even if S becomes inactive

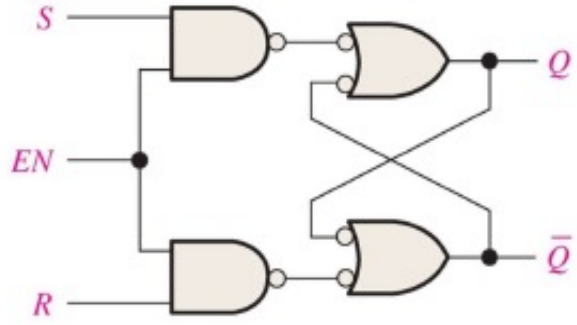
- If R is made active (ie set to 0) for a time longer than 2t → Q can be set to 0

Note:
1) an analogous description is valid for the NOR flip-flop, where
the only difference is that S and R are active when they are equal to 1

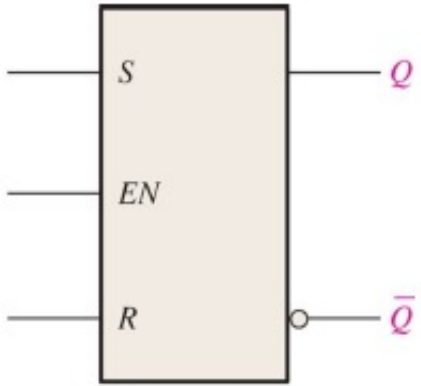
2) NAND FF is active low, NOR FF is active high

Gated Latch

A **gated latch** is a variation on the basic latch
The gated latch has an additional input, called enable (EN) that must be HIGH in order for the latch to respond to the S and R inputs



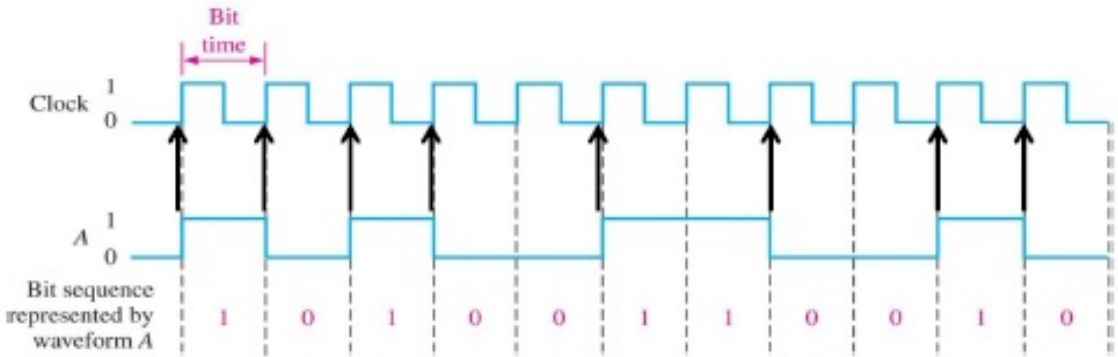
(a) Logic diagram



(b) Logic symbol

Clocks (CLK)

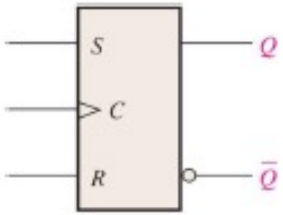
- In digital synchronous systems, all waveforms are synchronized with a clock.
  - The clock waveform itself does not carry information.
- The clock is a periodic waveform in which each interval between pulses (the period) equals the time for one bit.

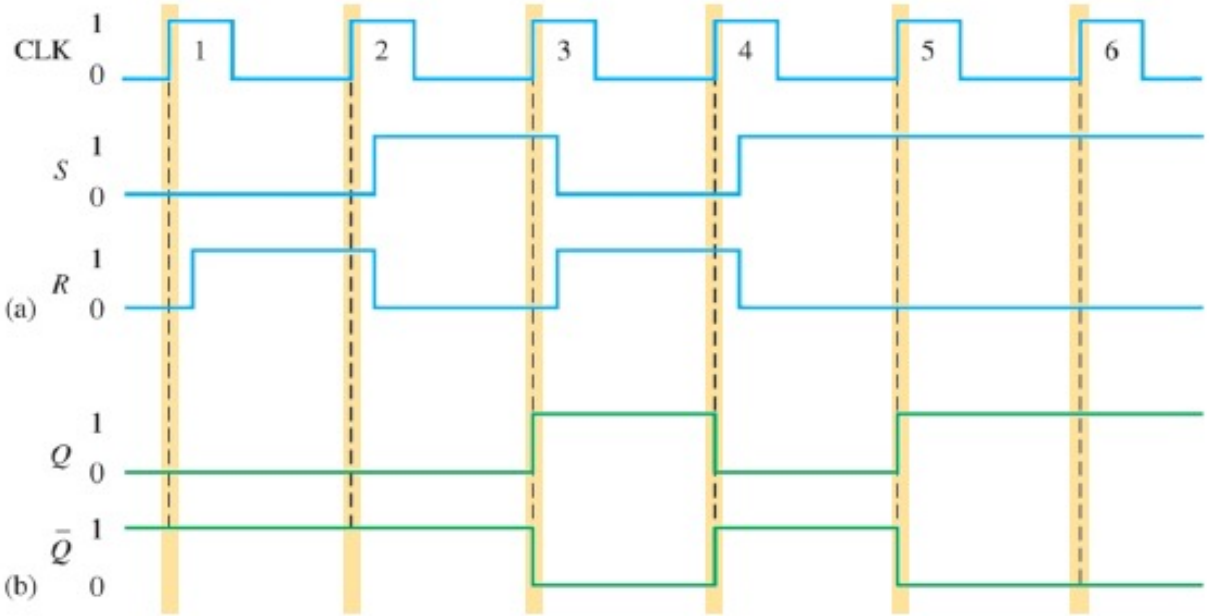


- Notice that change in level of waveform *A* occurs at the rising edge of the clock waveform.

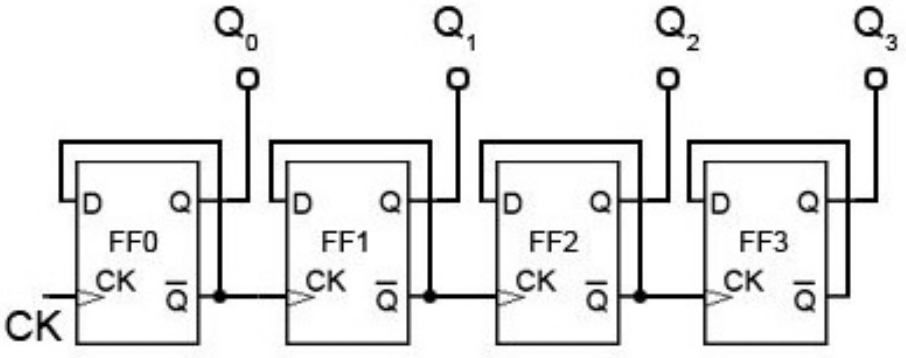
S-R Flip-Flop

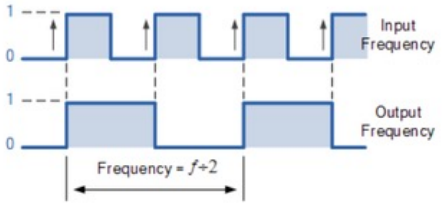
SR-Latch gated with a clock
gives a SR-Flip-Flop

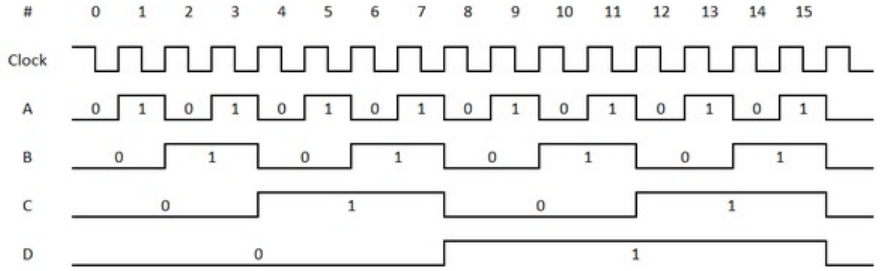




Counter

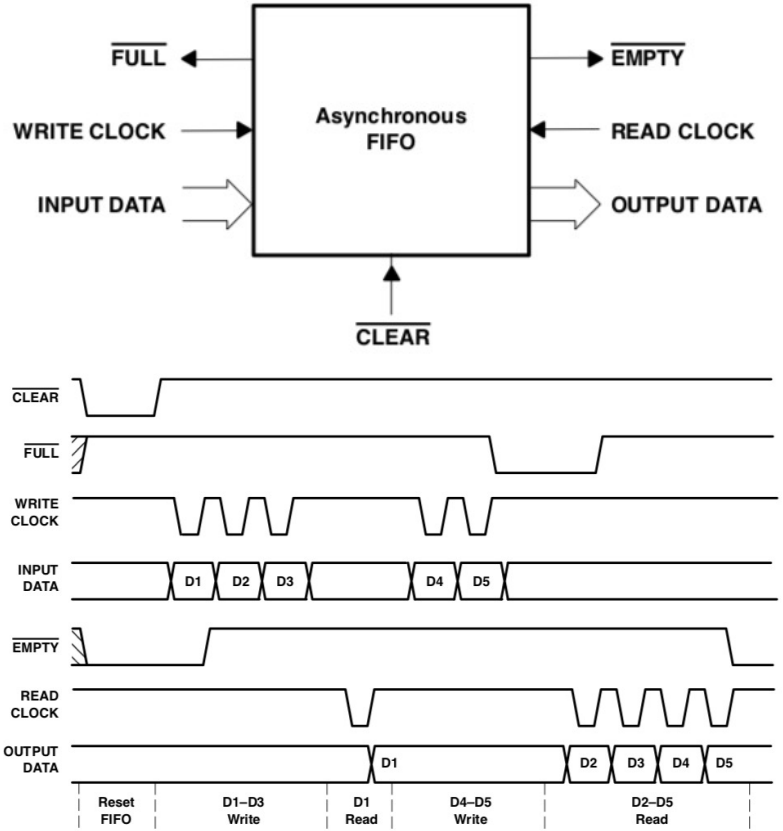

Q0
Q1
Q2
Q3
D
Q
FF0
FF1
FF2
FF3
CK
Q̄
CK


1
0
Input Frequency
Output Frequency
Frequency = f÷2

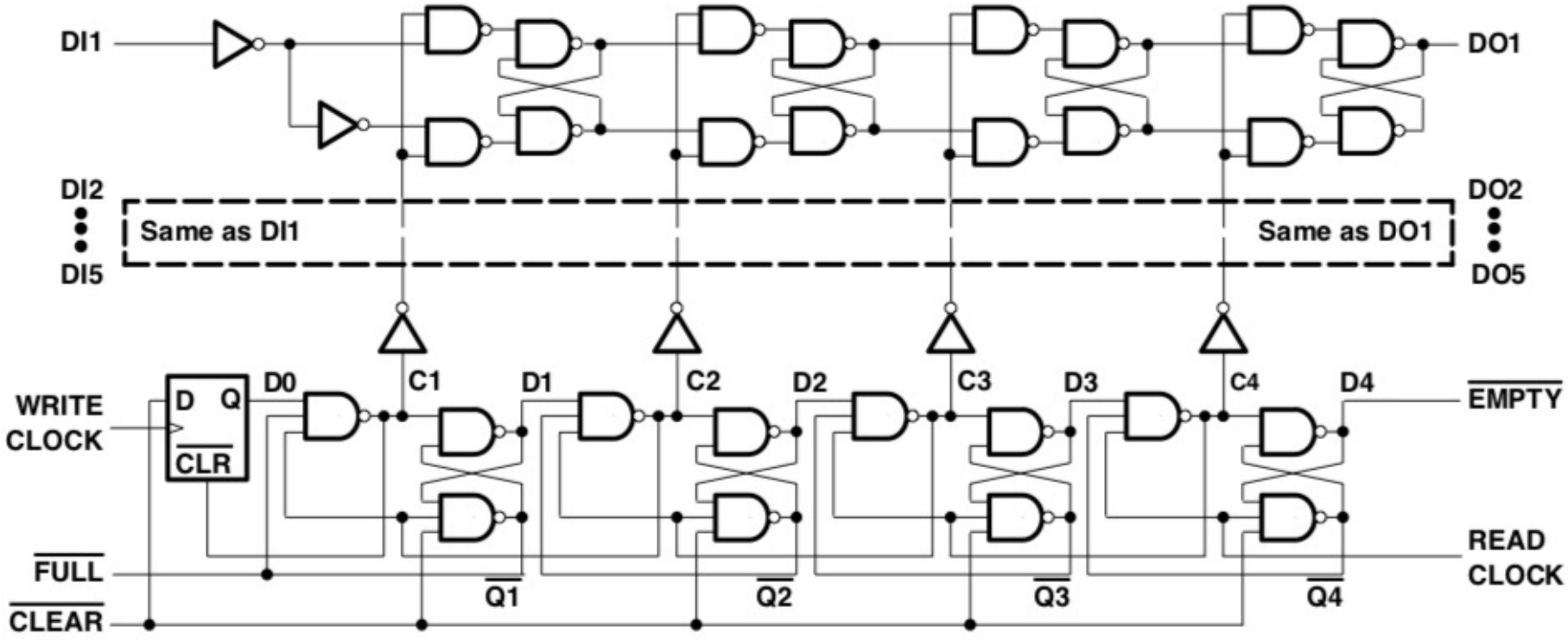

#
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Clock
A
B
C
D

How do I make a FIFO ?

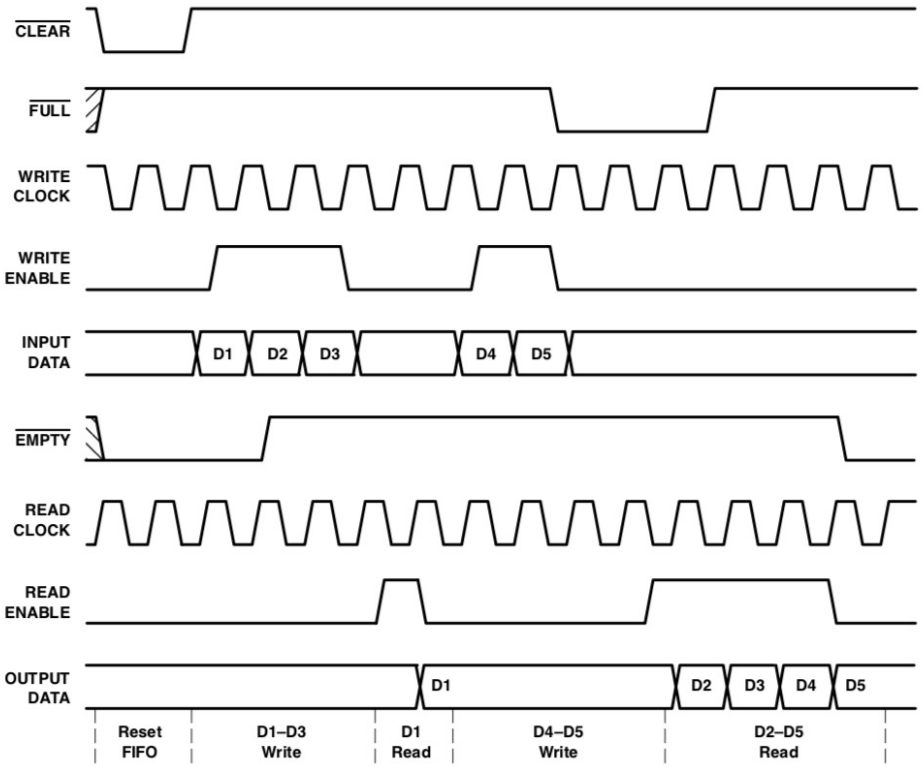
FIFO: asynchronous



Example implementation

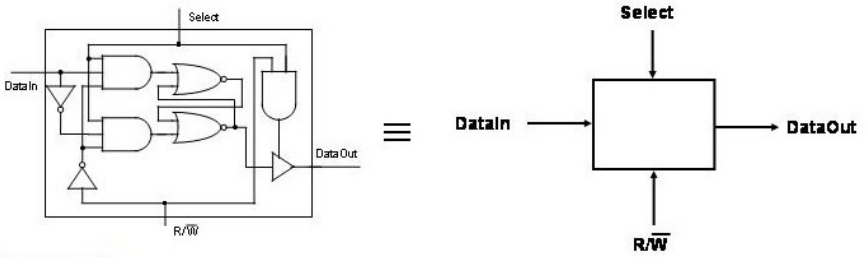


FIFO: synchronous

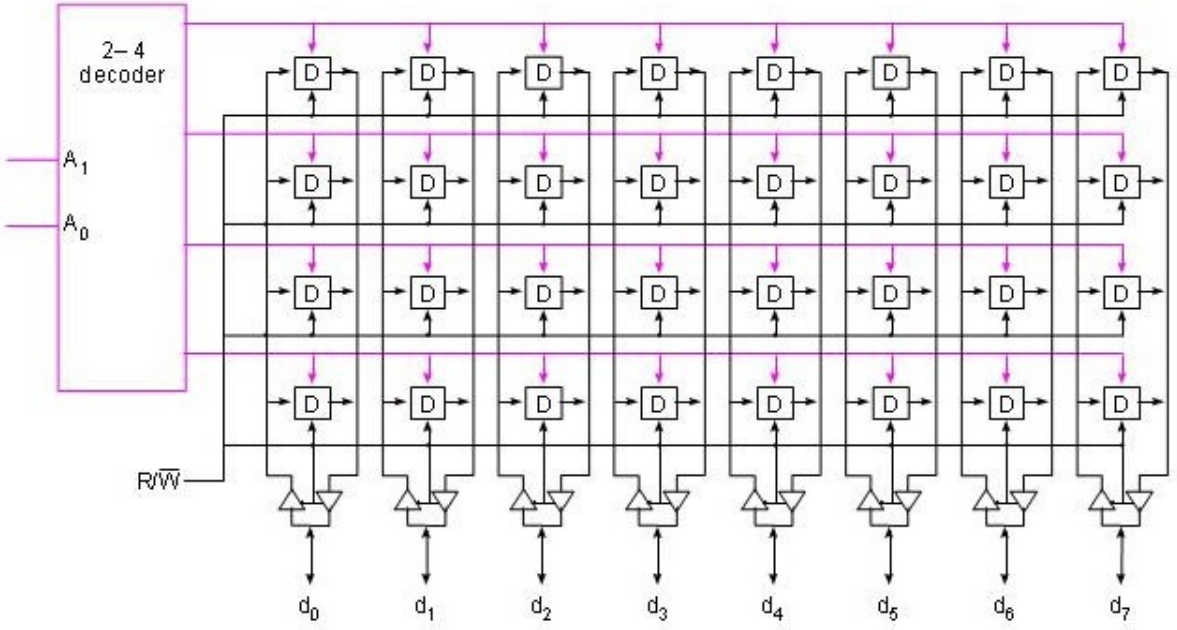


Possibly useful information

Addressing memory (simplified view)



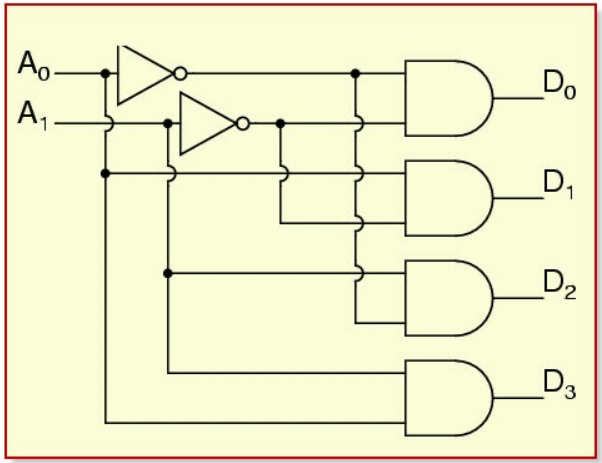
urce: Heuring – Jordan: Computer Systems Architecture and Design

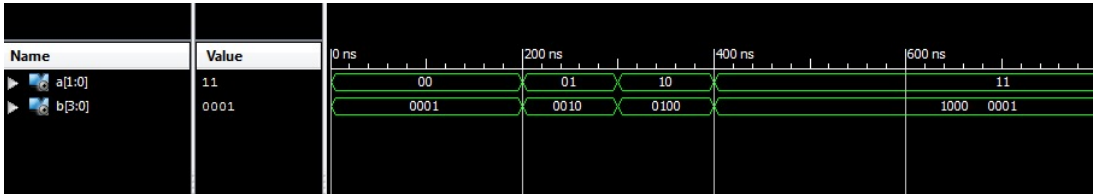


Source: Heuring – Jordan: Computer Systems Architecture and Design

The address space is the total number of unique addresses available to the CPU. Different addressable devices are “mapped” to different unique address ranges for access over the bus

Decoding addresses






```
library IEEE;
use IEEE.STD_LOGIC_1164.all;


entity decoder is
  port(
    a : in STD_LOGIC_VECTOR(1 downto 0);
    b : out STD_LOGIC_VECTOR(3 downto 0)
    );
end decoder;
architecture bhv of decoder is
begin


  process(a)
  begin
    case a is
      when "00" => b <= "0001";
      when "01" => b <= "0010";
      when "10" => b <= "0100";
      when "11" => b <= "1000";
    end case;
  end process;


end bhv;
```