The New Generation of Intelligent FPGA-based DAQ Architectures

Igor Konorov

Institute for Hadronic Structure and Fundamental Symmetries (E18)

Technical University of Munich

Department of Physics

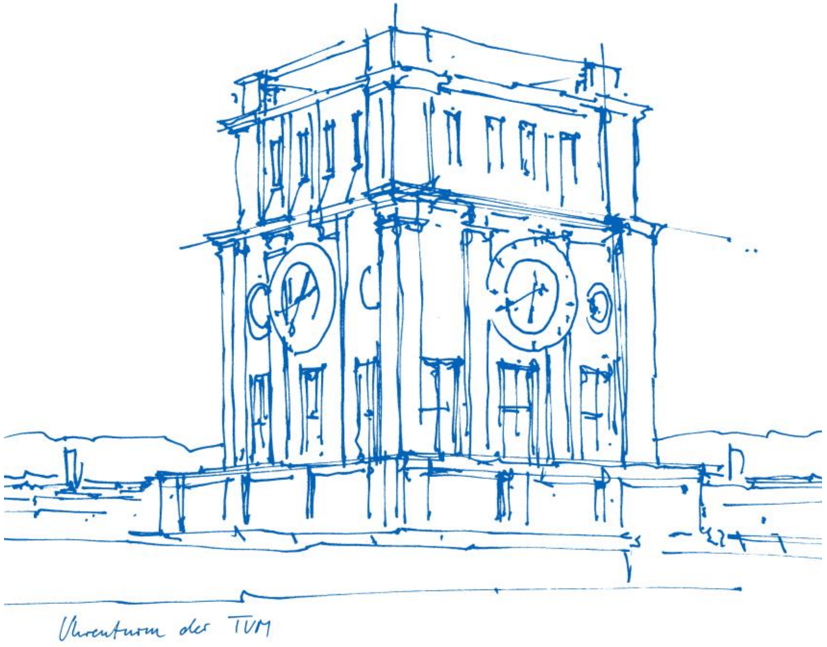
Joint ICTP-IAEA School on FPGA-based SoC

and its Applications for

Nuclear and Related

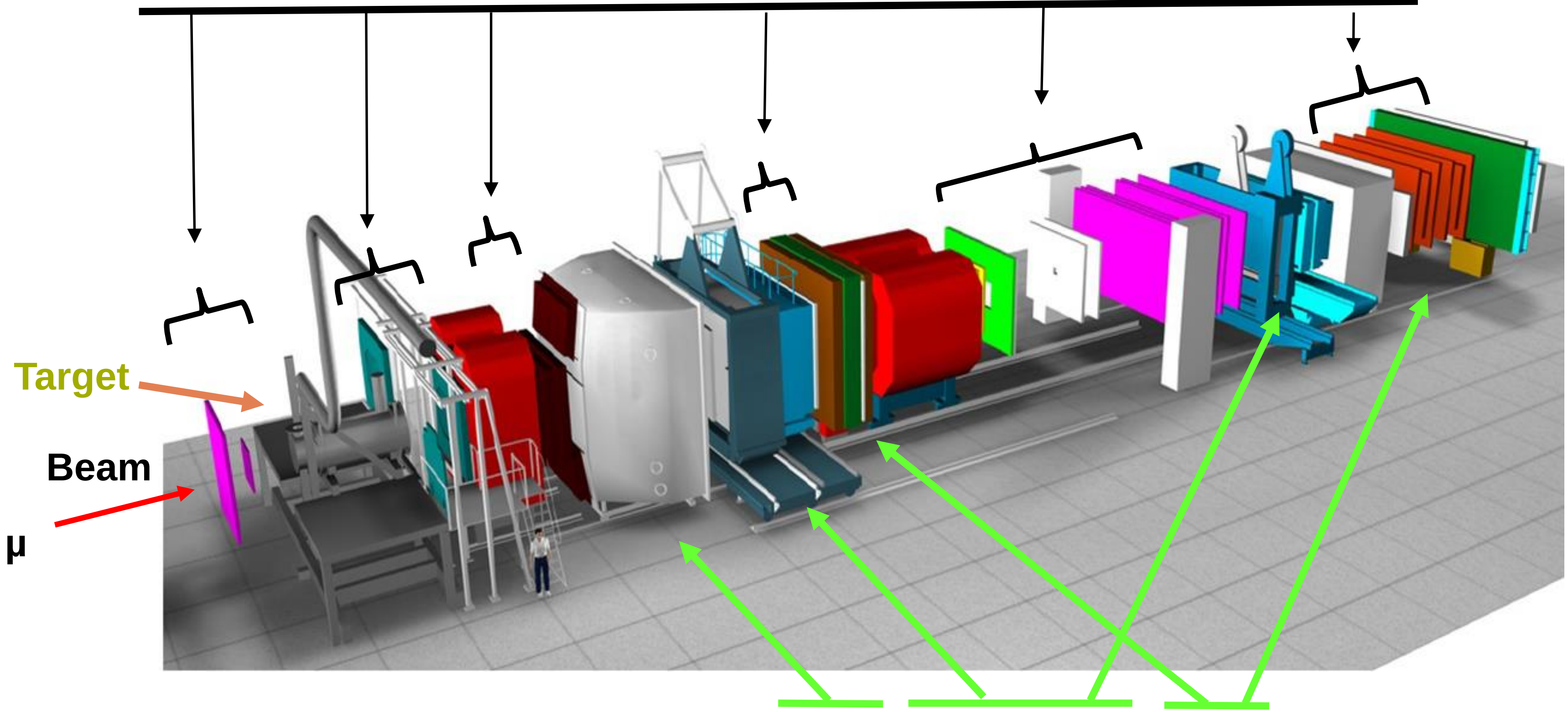
Instrumentation

25 January – 19 February



Particle Physics Experiment (COMPASS)

Tracking detectors : coordinates of charge particles => particle trajectories



Target

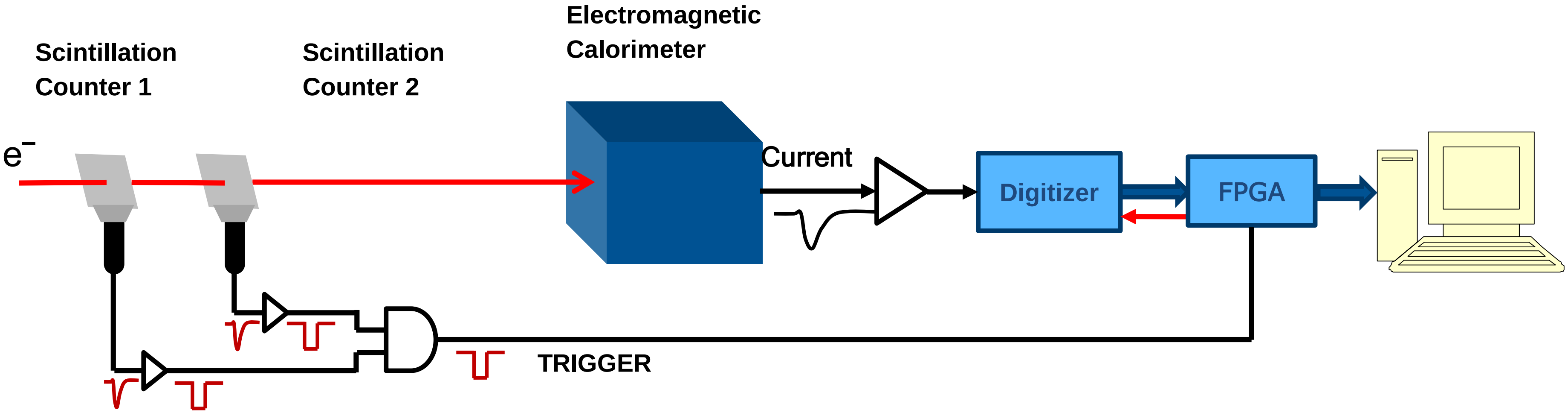
Beam

μ

Particle identification detectors : RICH, Calorimeters, Muon Detectors

300 000 detector channels

Experiment : Electron Energy Measurement



TRIGGER – define time when detector signal to be measured

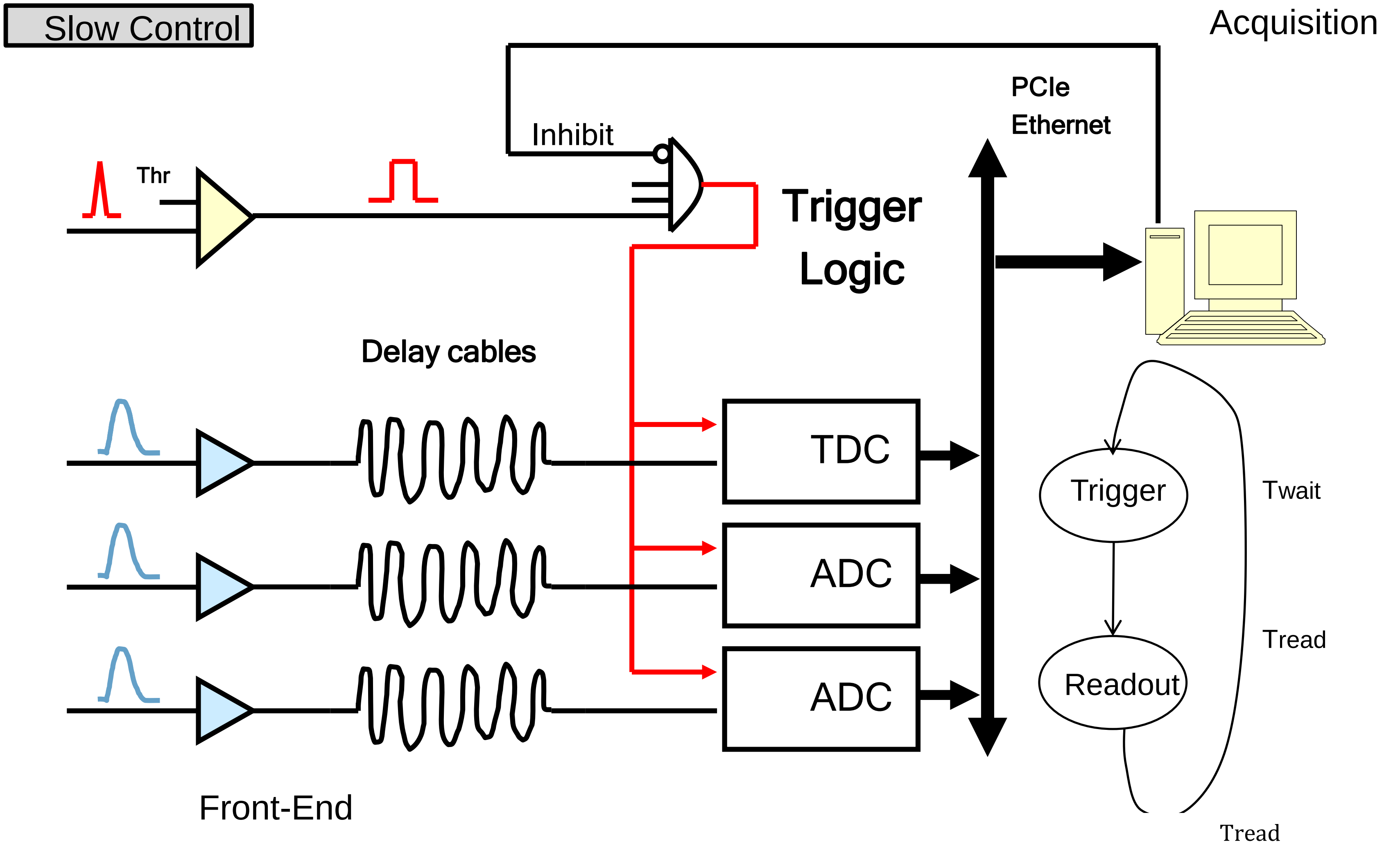
Why read only when trigger and not continuously ?

- Not feasible to measure continuous data flow
- Not feasible to transmit such amount of data
- Not feasible to sore such amount of data

DAQ Tasks

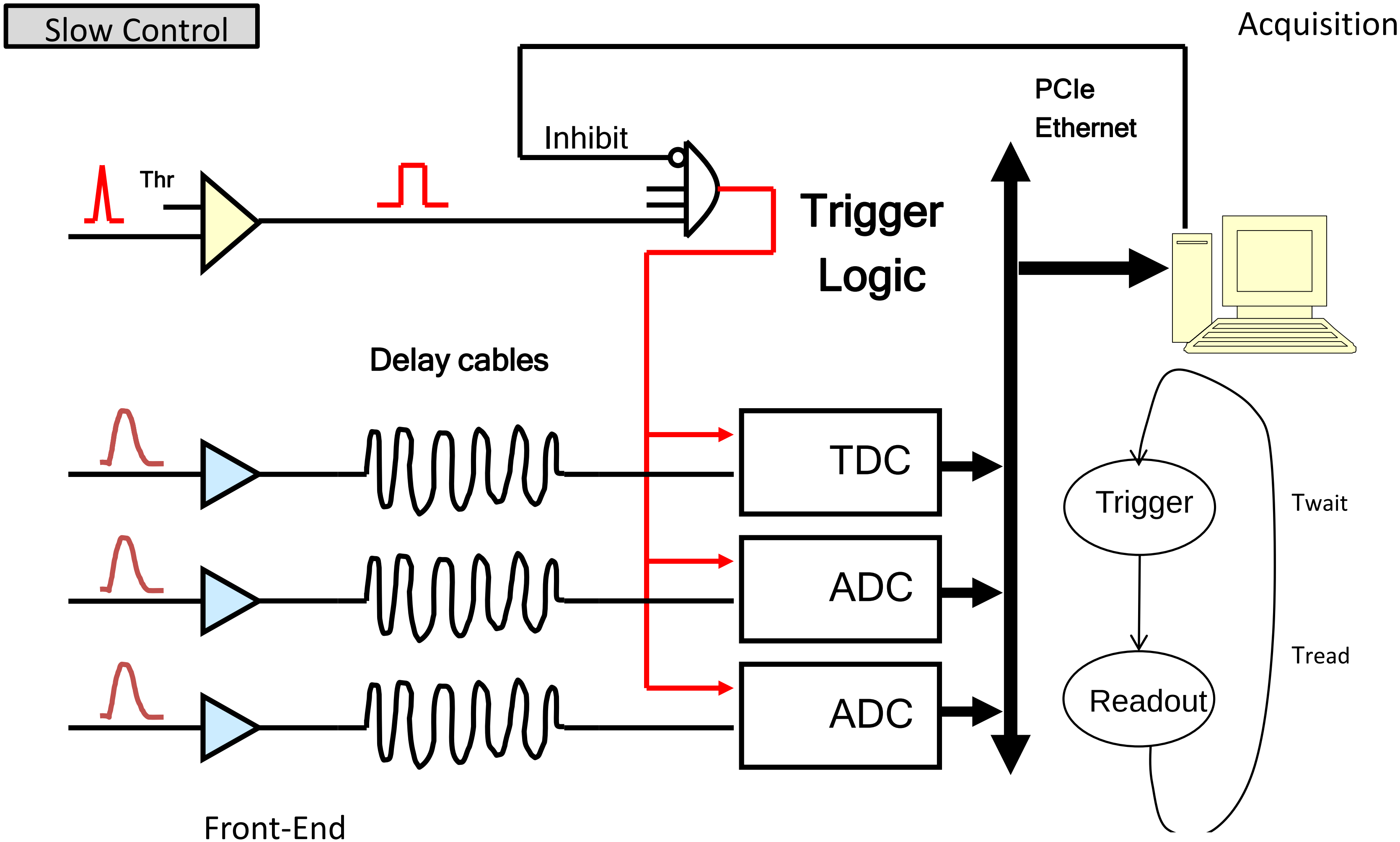
- Time reference system ; system CLOCK, Trigger together with event ID
- Acquire data from detectors – readout by Front-Ends
- Collect data and Event Building if multiple FEE cards
- Data storage
- Configuration and Monitoring. Is everything OK?

DAQ Architecture in Particle Physics



$$\text{Tdead} = \frac{\text{Tread}}{\text{Twait} + \text{Tread}}$$

DAQ Architecture in Particle Physics



Slow Control

Acquisition

Inhibit

Thr

PCIe

Ethernet

Trigger Logic

Delay cables

TDC

ADC

ADC

Trigger

Readout

Twait

Tread

Front-End

$$\mathrm{Tdead} = \frac{\mathrm{Tread}}{\mathrm{Twait} + \mathrm{Tread}}$$

Data Taking Efficiency

Probability for uncorrelated events described by Poisson distribution

$$q(t) = e^{-t/\lambda}$$

λ – average time between events

A rule of thumb:

Dead Time = $\frac{Tbusy}{\lambda}$

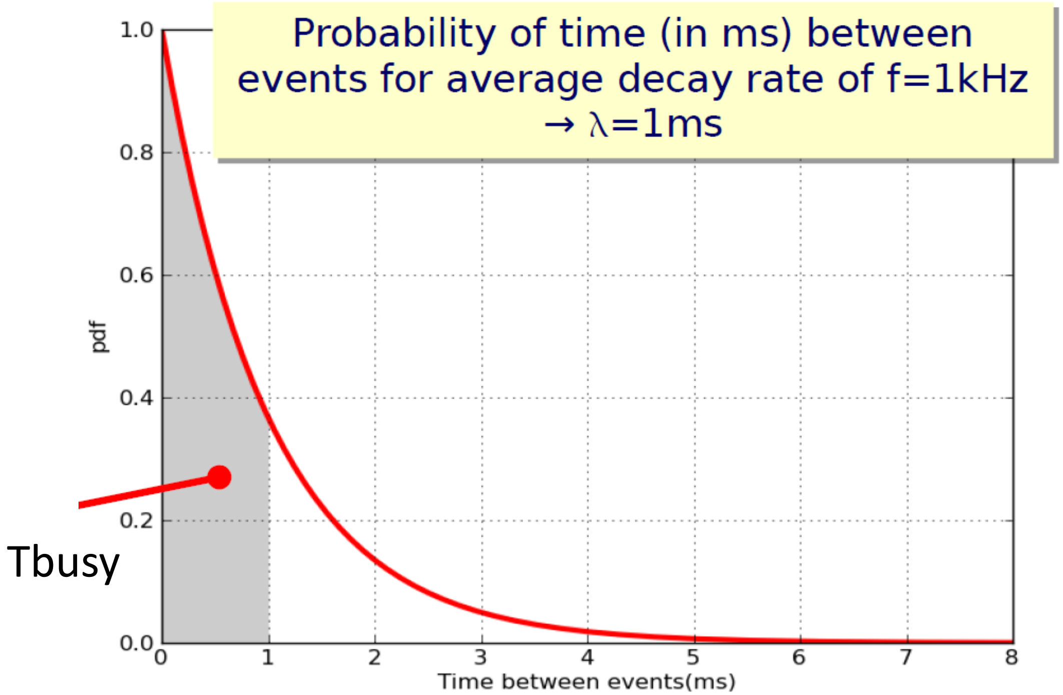
Tbusy – DAQ busy time

Example:

1kHz => λ = 1ms

Tbusy = 50 useconds

DeadTime = 0.05/1 = 0.05 or 5%



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

Tbusy

Data Taking Efficiency

Probability for uncorrelated events described by Poisson distribution

$$q(t) = e^{-t/\lambda}$$

λ – average time between events

A rule of thumb:

Dead Time = $\frac{Tbusy}{\lambda}$

Tbusy – DAQ busy time

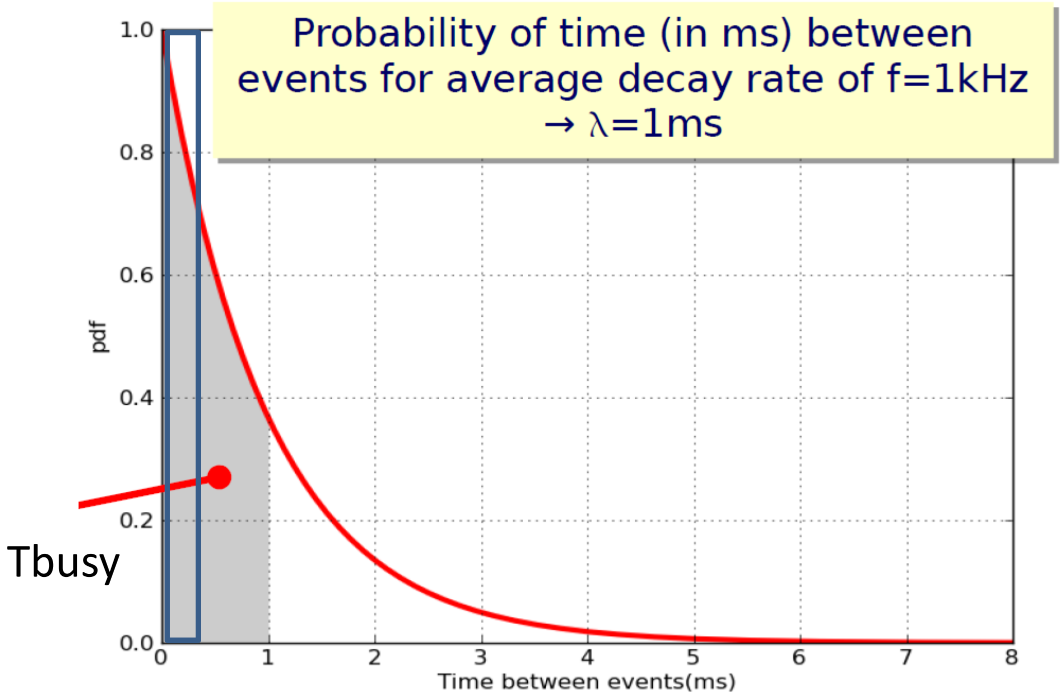
Example:

1kHz => λ = 1ms

Tbusy = 100 useconds

DeadTime = 0.2/1 = 0.2 or 20%

20% of events are lost !!!



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

Data Taking Efficiency

Probability for uncorrelated events described by Poisson distribution

$$q(t) = e^{-t/\lambda}$$

λ – average time between events

A rule of thumb:

Dead Time = $\frac{Tbusy}{\lambda}$

Tbusy – DAQ busy time

Example:

1kHz => λ = 1ms

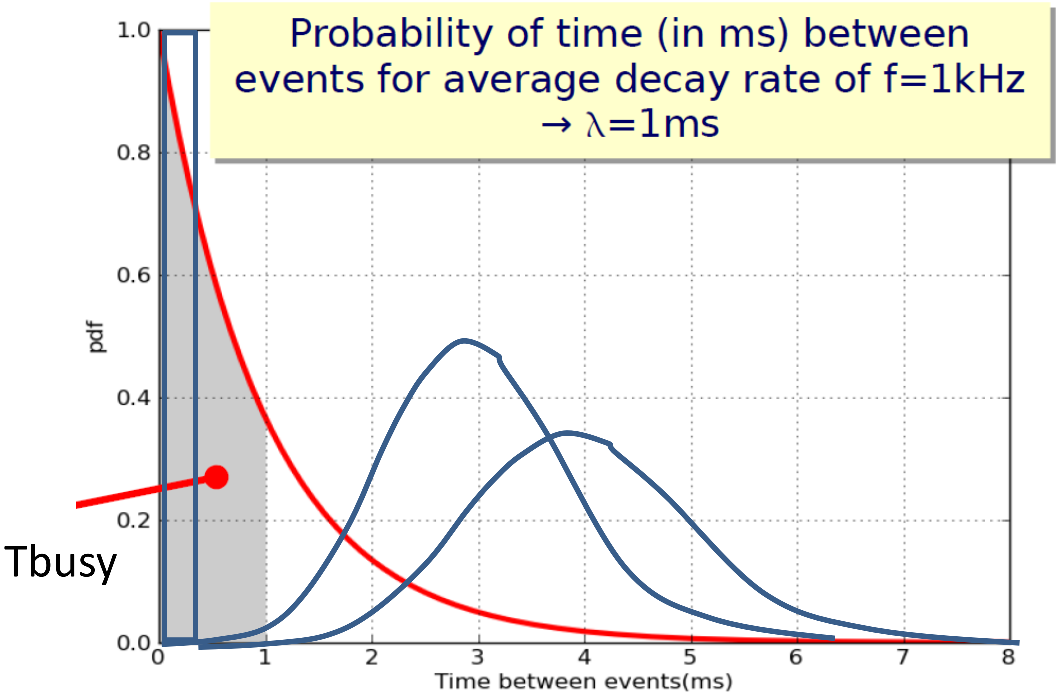
Tbusy = 100 useconds

DeadTime = 0.2/1 = 0.2 or 20%

20% of events are lost !!!

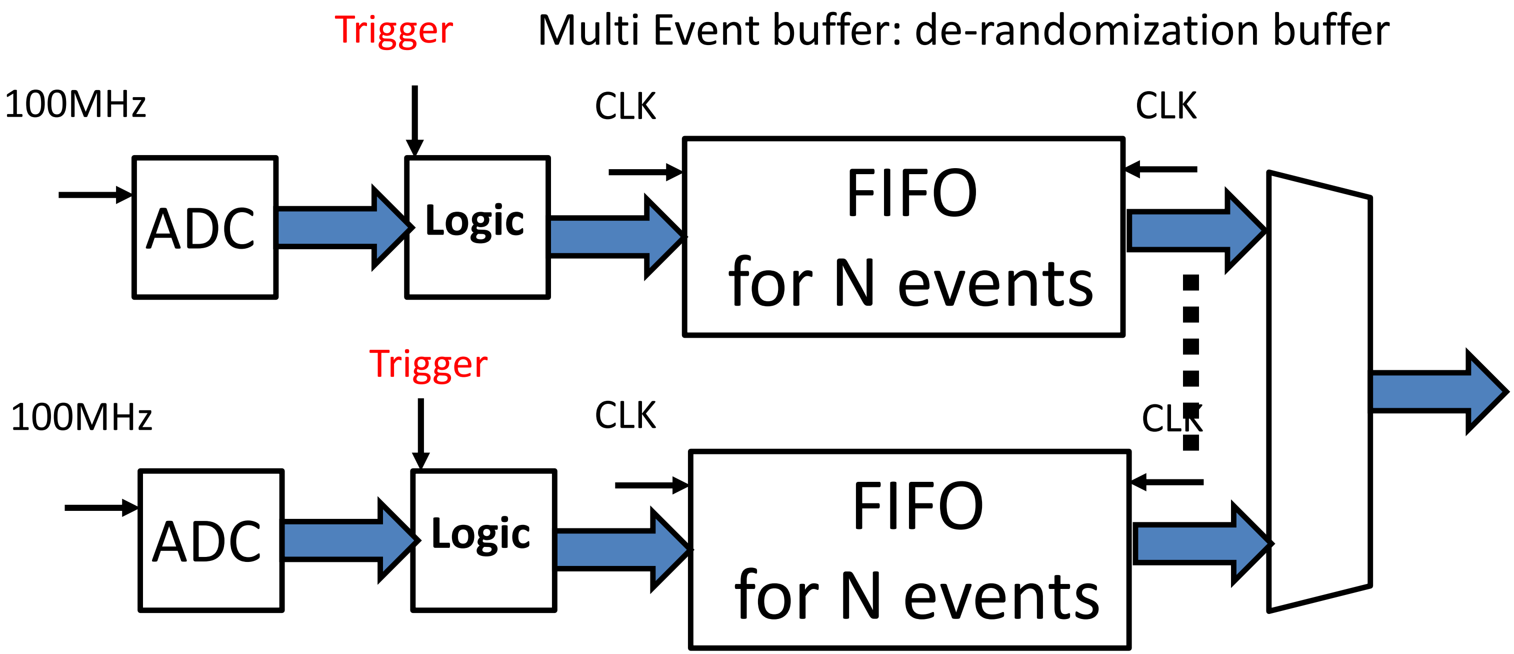
Solution:

Store N events before Readout



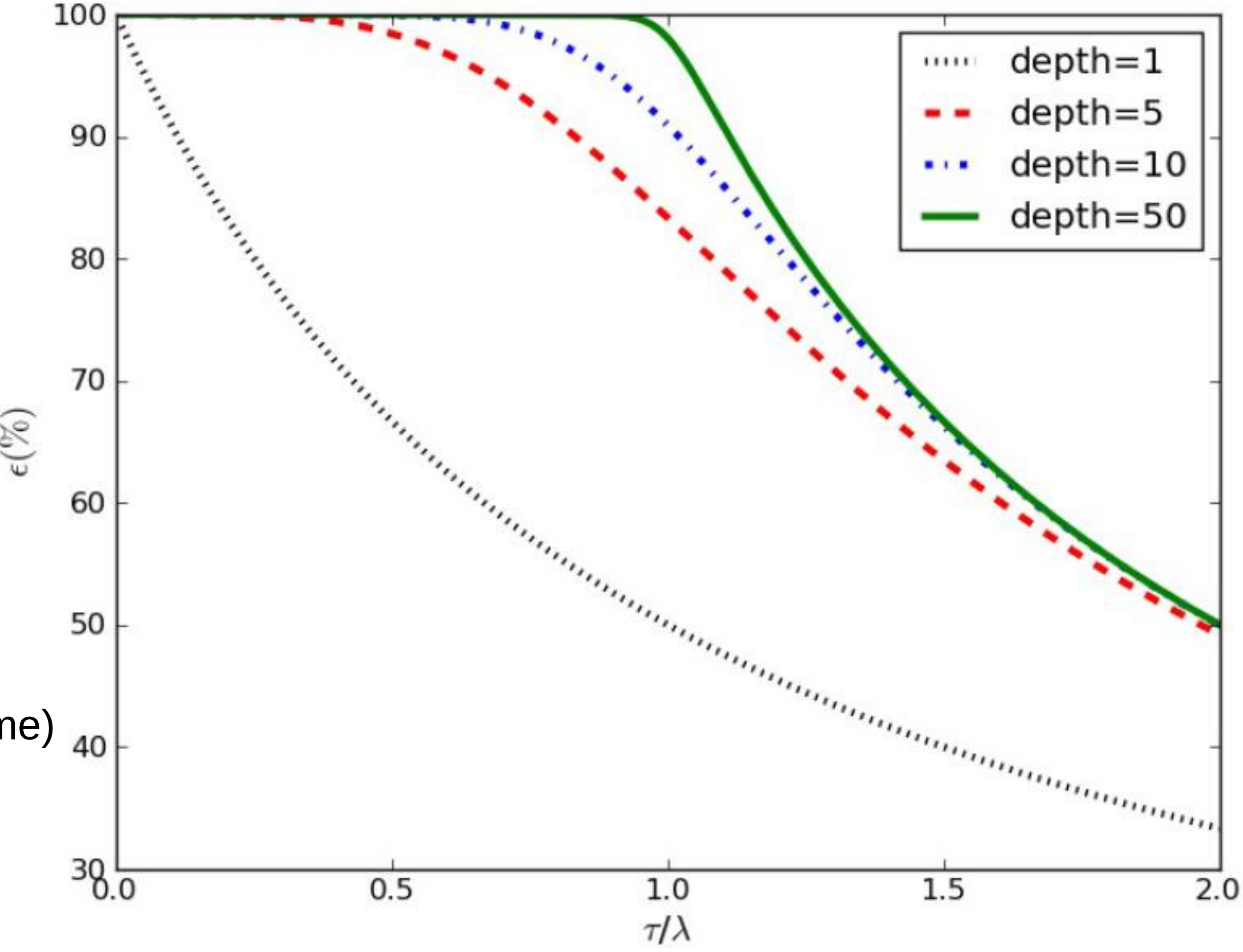
Probability of time (in ms) between events for average decay rate of f=1kHz → λ=1ms

Pipe Line Front Ends



Input : Poisson distribution
Output : more like a Gaussian centered around average value

DAQ efficiency vs FIFO Depth

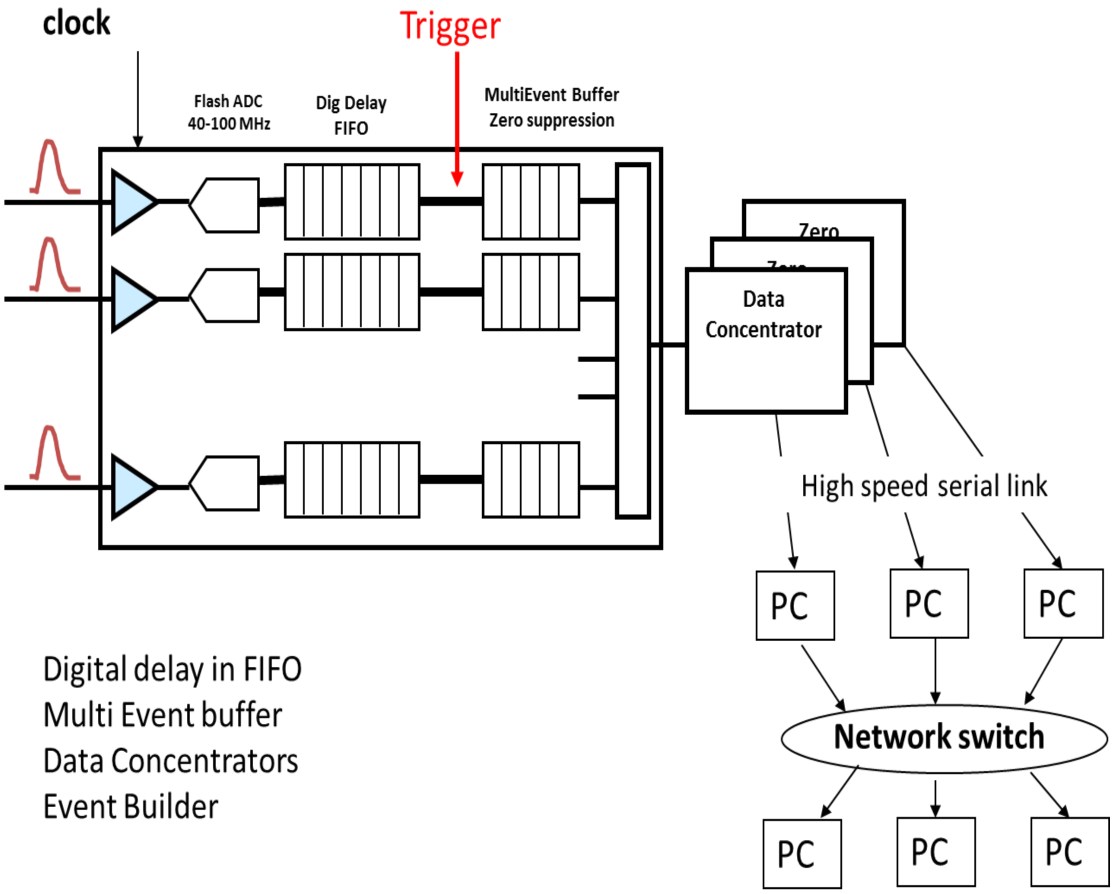


DAQ Efficiency

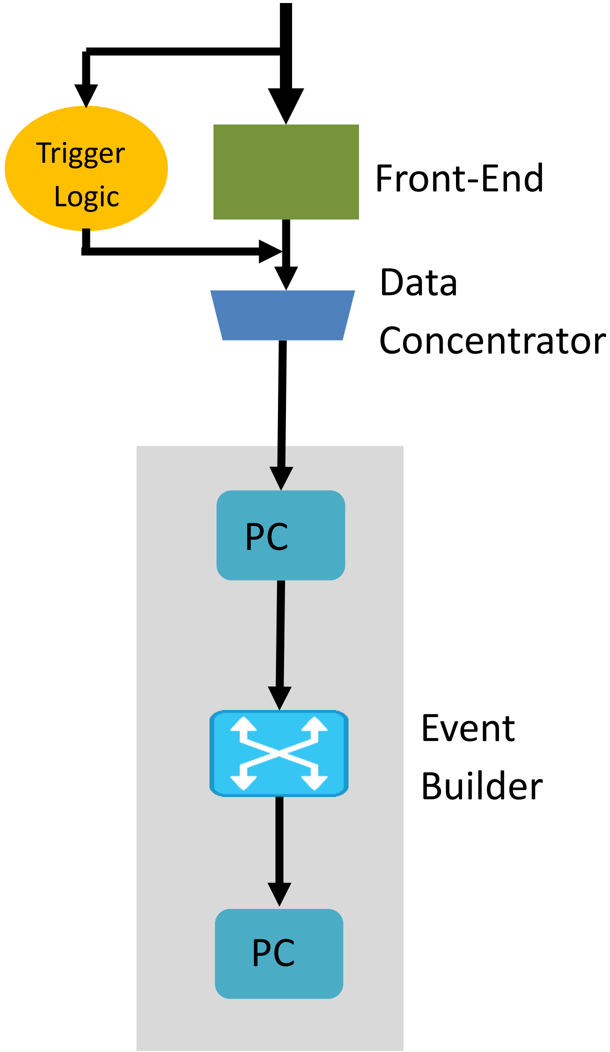
(1 – DeadTime)

Trigger rate/DAQ rate capability

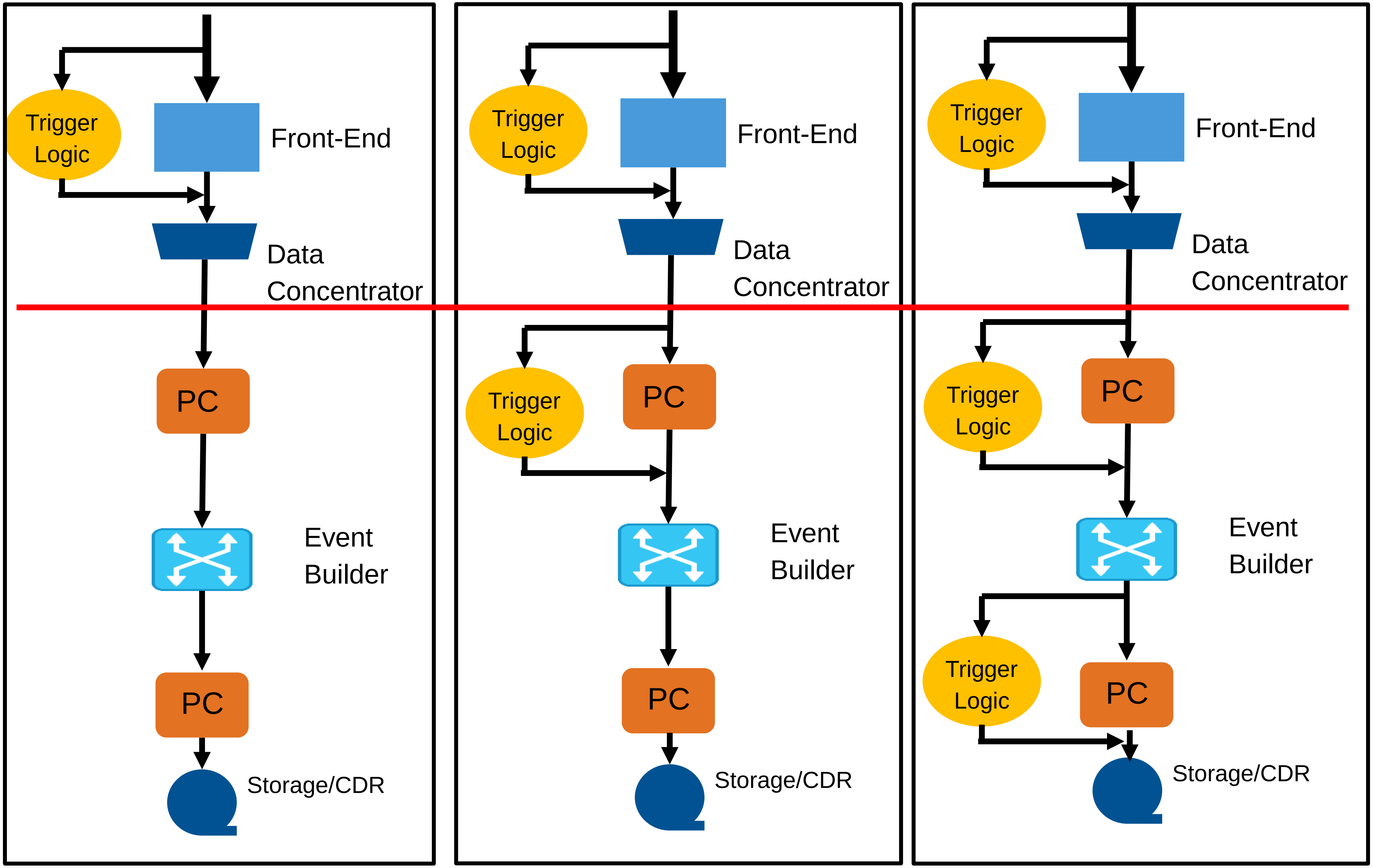
Multi Channel Data Flow



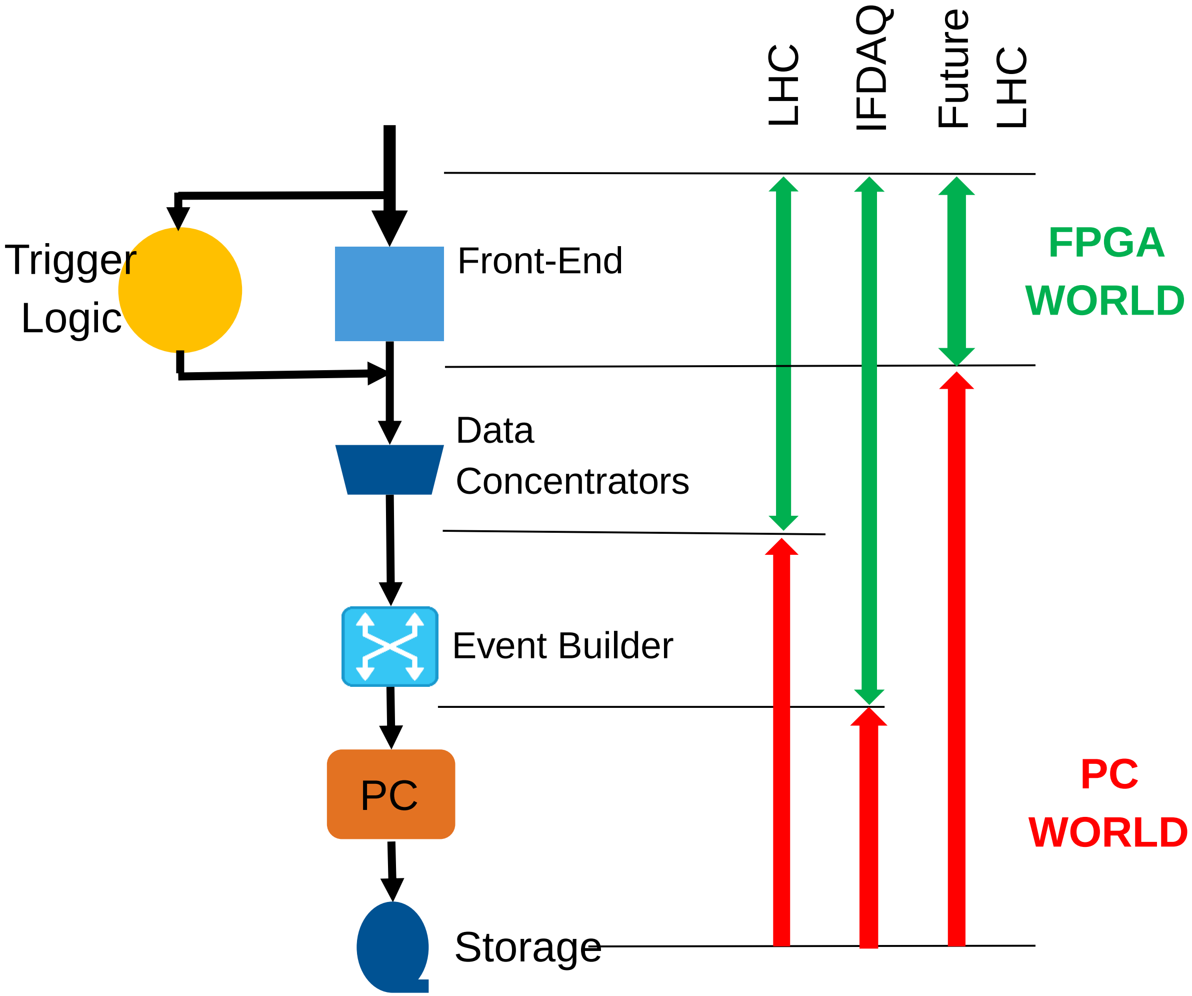
Digital delay in FIFO
Multi Event buffer
Data Concentrators
Event Builder



Multiple Trigger Levels



DAQ Architectures



Concept of iFDAQ

- **Minimize amount of real-time software processes and implement Event Builder in FPGA**

iFDAQ Frame work

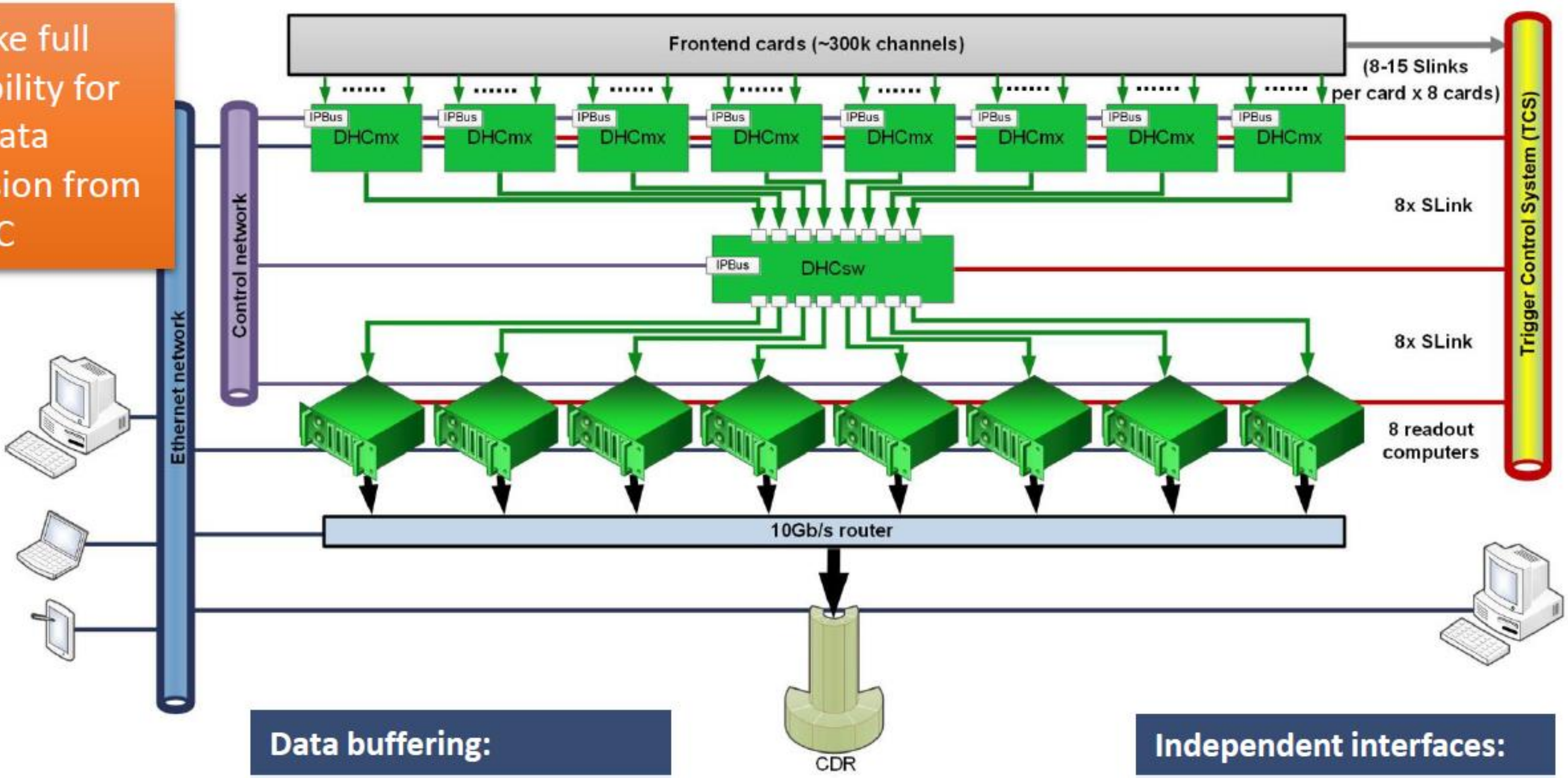
- **Implementing congestion free Event Builder in FPGA**
- **Intelligent data handling**
- **Unified Interfaces**
- **Unified IP Cores**
- **Integrated Digital Trigger (to be impl.)**

Advantages

- **Increased compactness**
- **Increased reliability**
- **Reduced cost**

iFDAQ Architecture

FPGAs take full responsibility for reliable data transmission from FEEs to PC



Data buffering:

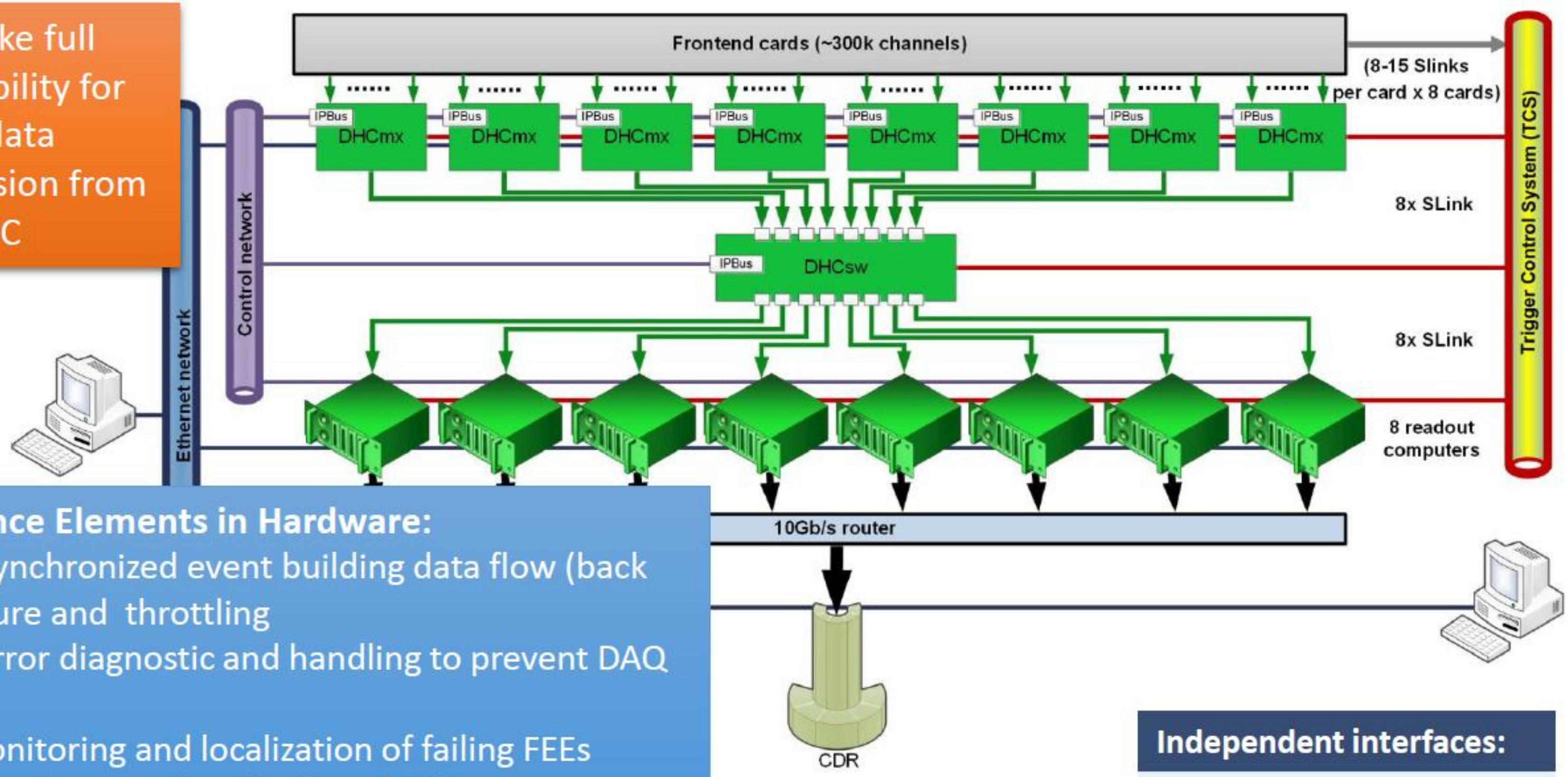
- 4GB RAM on each module => possibility to average data rate over spill cycle
- 1 GB/s sustained rate

Independent interfaces:

- synchronization → **TCS (Trigger Control System)**
- data flow (event building) → **SLINK**
- configuration and data flow control -> **IPbus**

iFDAQ Architecture

FPGAs take full responsibility for reliable data transmission from FEEs to PC



Intelligence Elements in Hardware:

- Self-synchronized event building data flow (back pressure and throttling
- FEE error diagnostic and handling to prevent DAQ crash
  => monitoring and localization of failing FEEs
- Automatic resynchronization of FEEs
  => FEEs can be attached/detached at any time
- Back pressure and throttling mechanism
  => prevention from crashes due to high event and data rate

Independent interfaces:

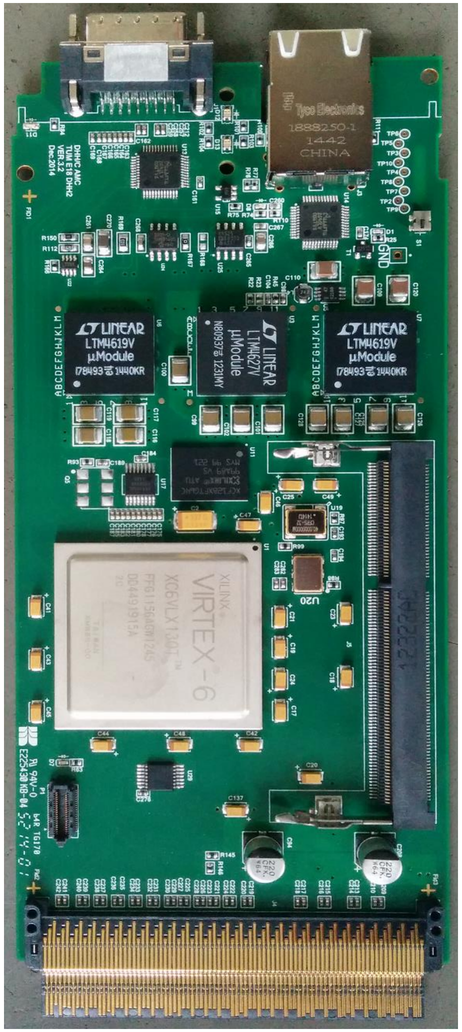
- ○ synchronization → **TCS (Trigger Control System)**
- ○ data flow (event building) → **SLINK**
- ○ configuration and data flow control -> **IPbus**

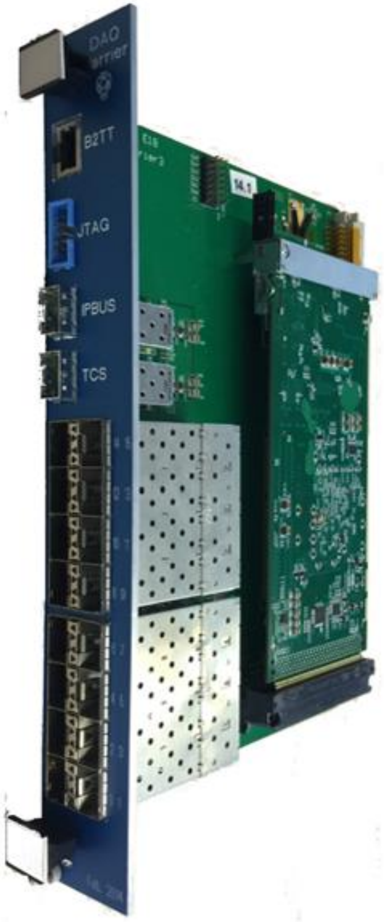
FPGA DAQ Module

FPGA

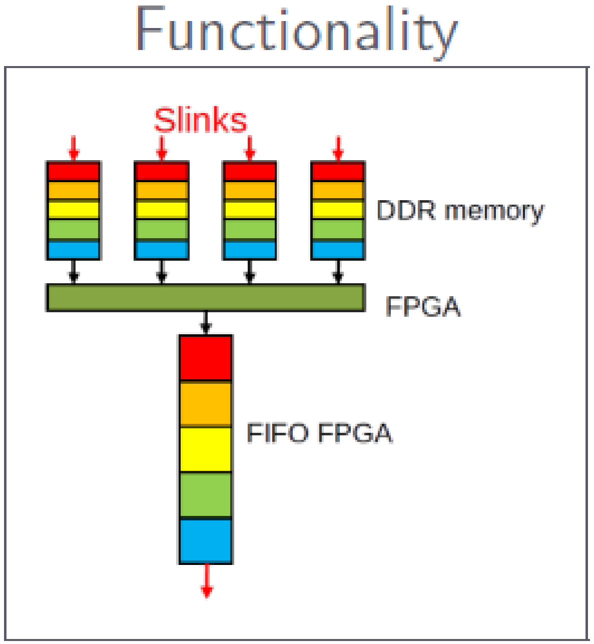
- XC6V130T
- 16 x 6.5 Gbps serial links
- Ethernet UDP
- Time Distribution Input, LVDS

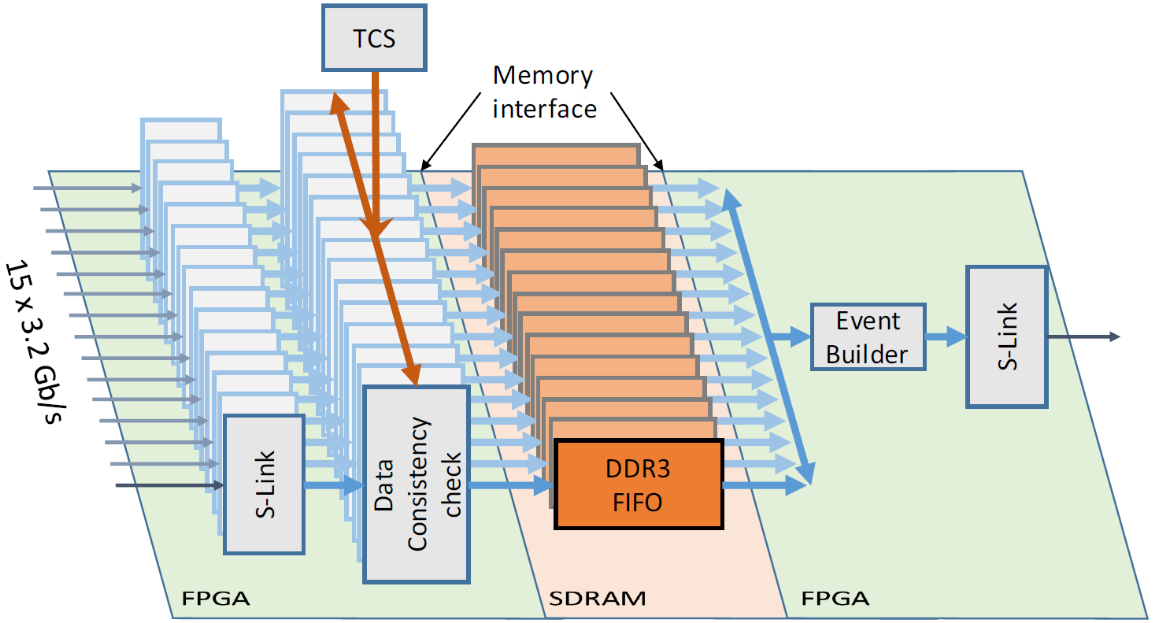
4 Gbyte DDR3 Memory





Data Concentrator Firmware

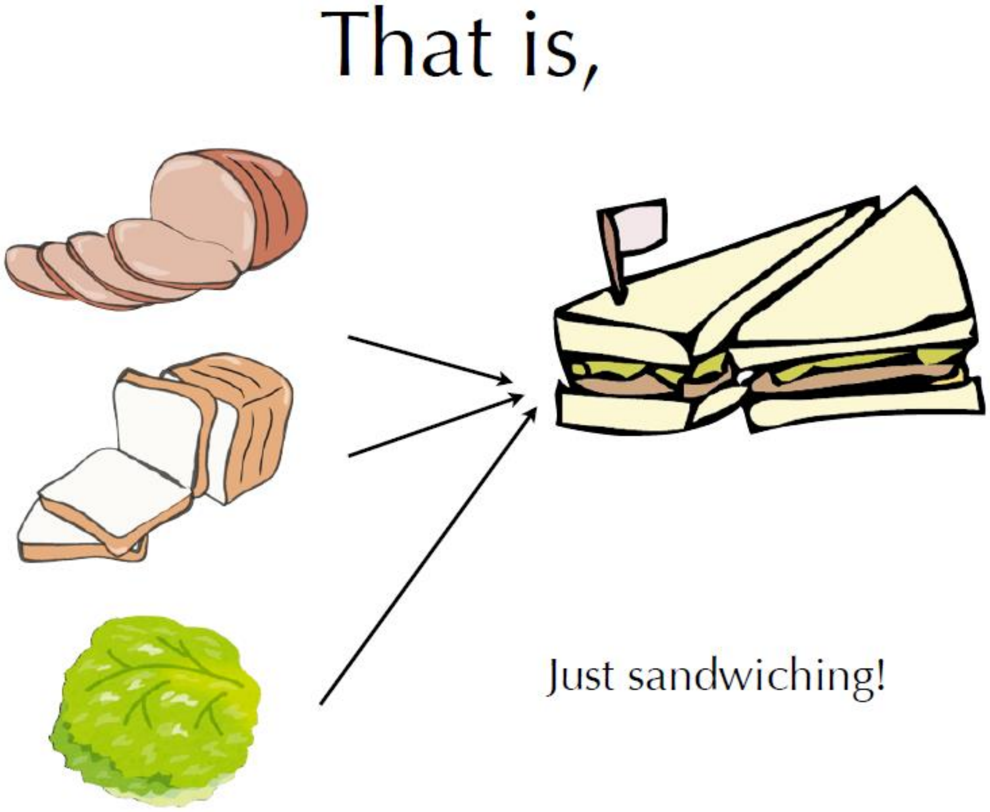




Event Builder

Event Builder,

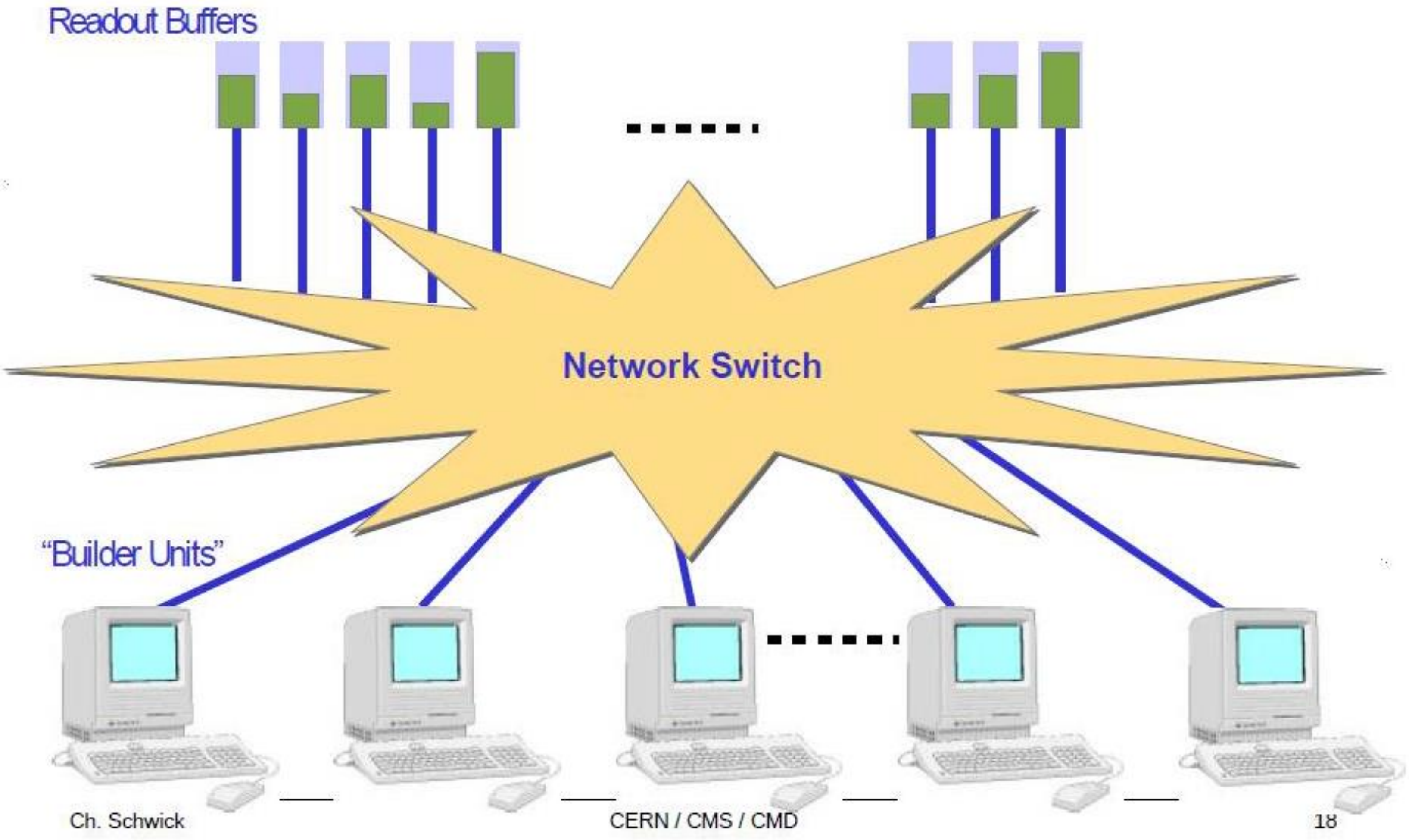
taken from my KEKB colleague Suzuki-san



That is,

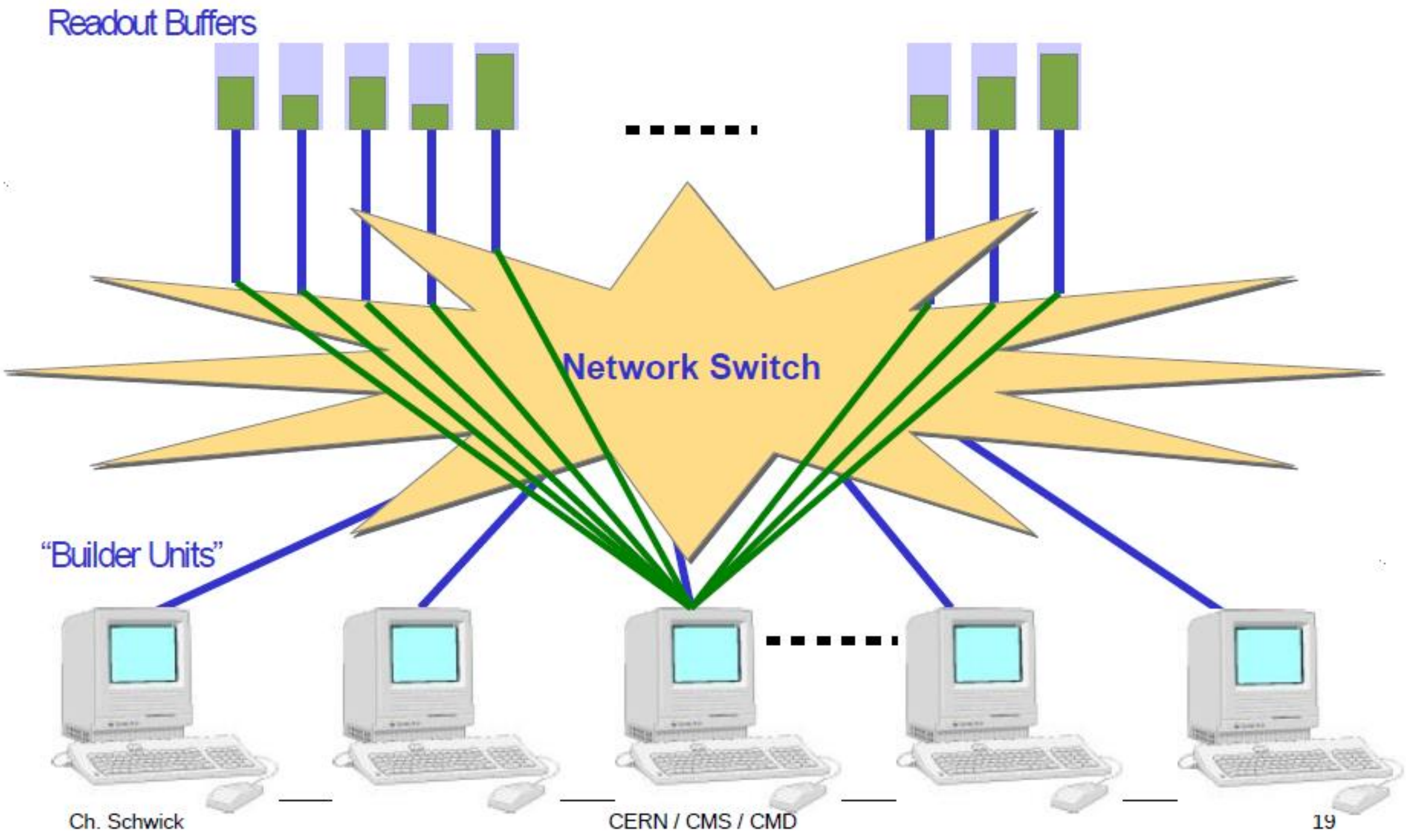
Just sandwiching!

Event Building Challenges I



From CERN ISODAQ School

Event Building Challenges II



From CERN ISODAQ School

Traffic Pattern Causes Congestion Problem

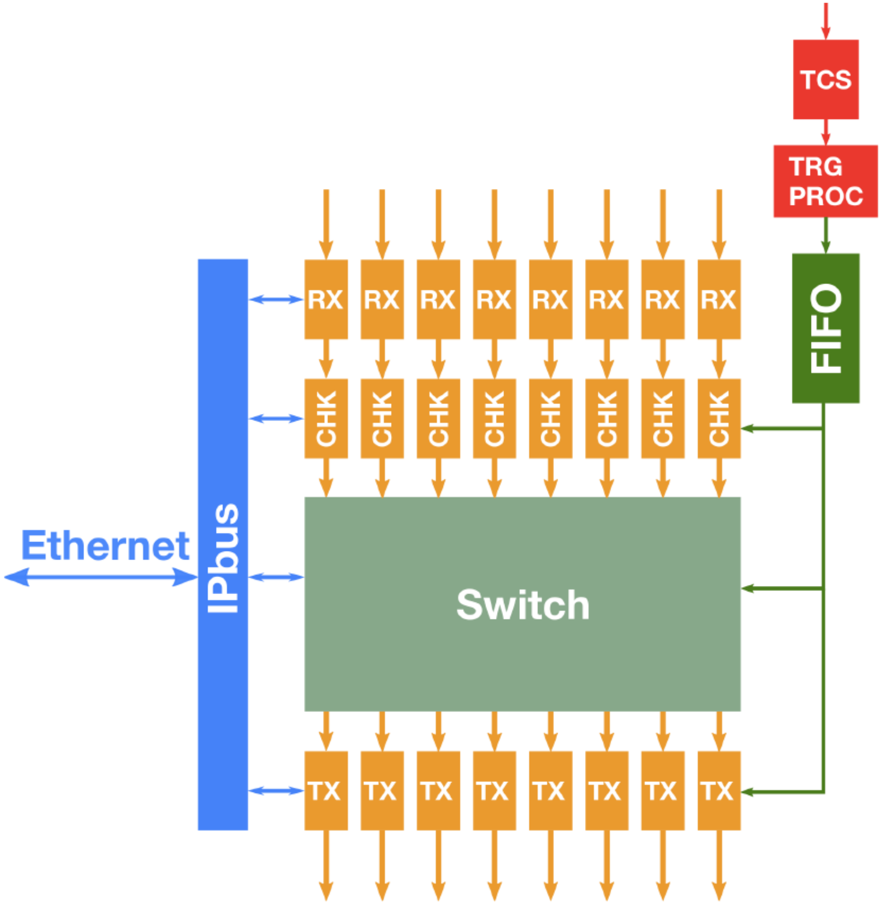
From CERN ISODAQ School

Event Builder Firmware

Control incoming data streams

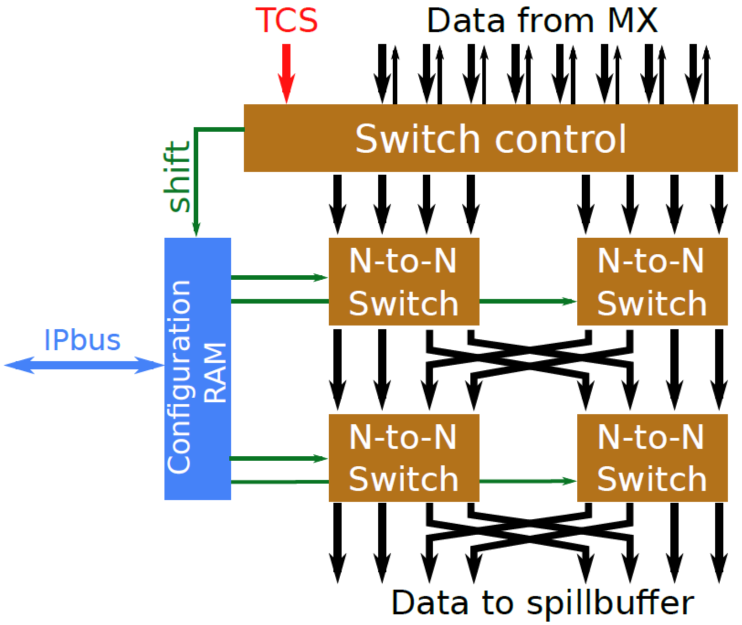
Verification of data consistency

Control of 8 x 8 switch

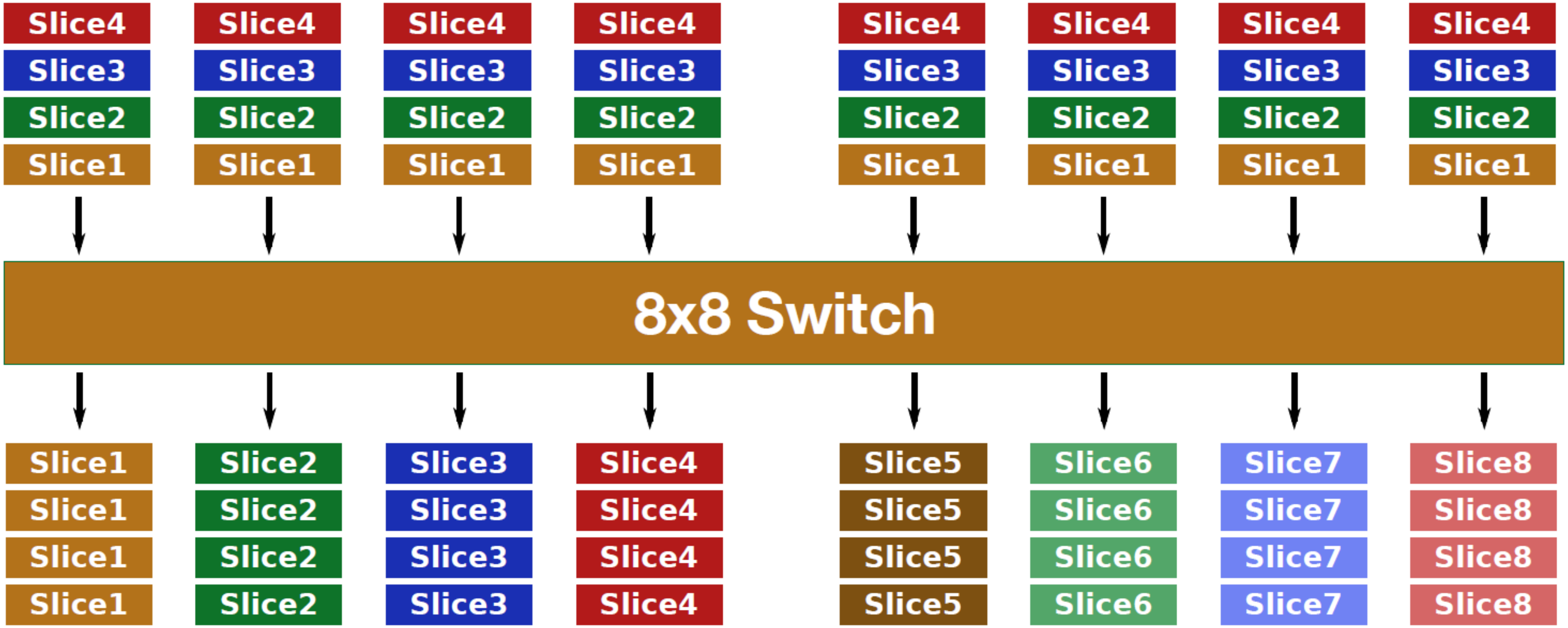


Event Builder Firmware

- Switch control
  - Defines switch interconnections
    - Connects one input to one output
  - Combines N consecutive events in one SLICE
    - Average size of SLICEs
  - After processing one SLICE changes switch interconnections to the next one
    - Information about different interconnections stored in RAM



Switch Operation



Slice4 Slice4 Slice4 Slice4 Slice4 Slice4 Slice4 Slice4

Slice3 Slice3 Slice3 Slice3 Slice3 Slice3 Slice3 Slice3

Slice2 Slice2 Slice2 Slice2 Slice2 Slice2 Slice2 Slice2

Slice1 Slice1 Slice1 Slice1 Slice1 Slice1 Slice1 Slice1

8x8 Switch

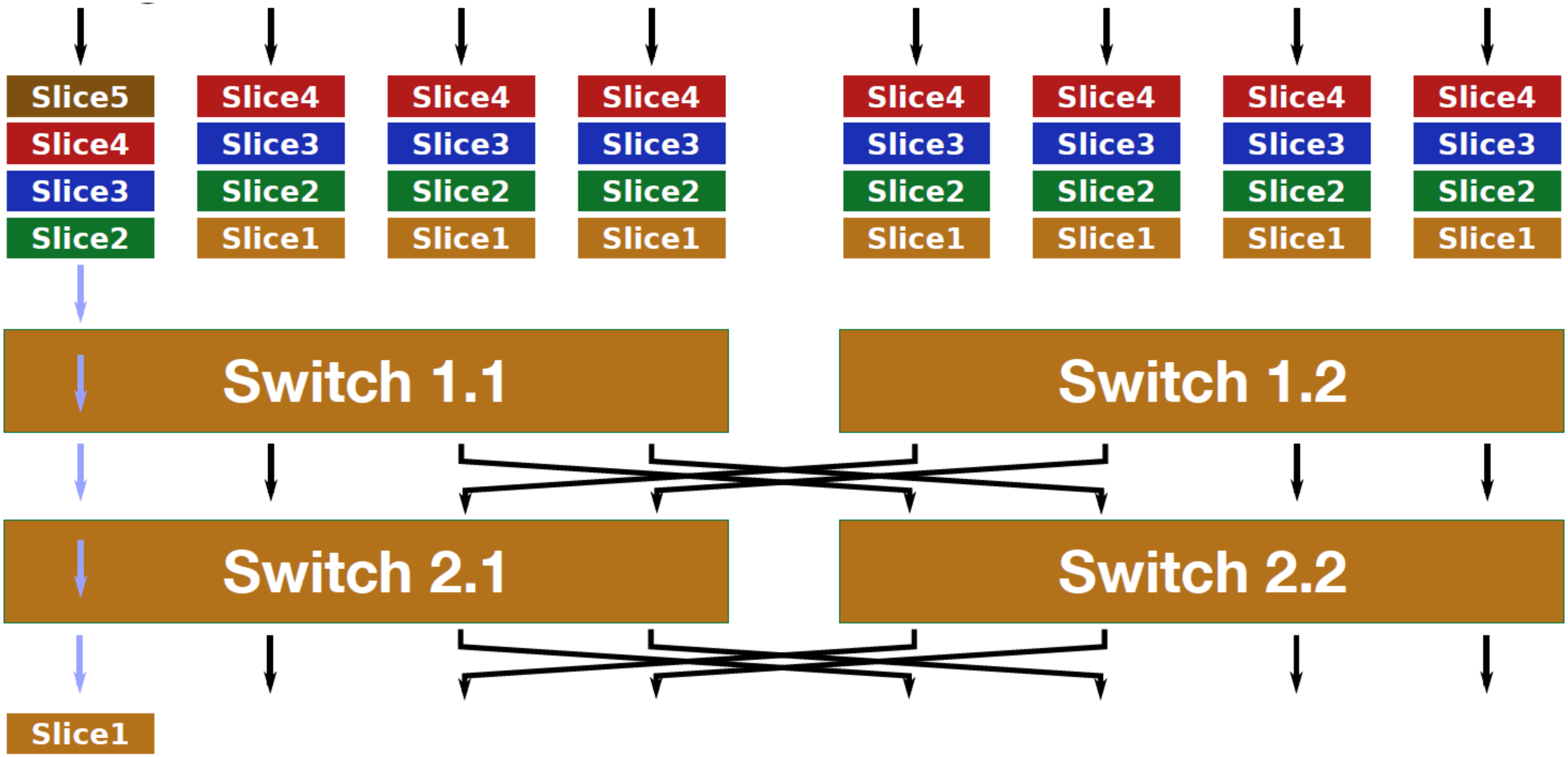
Slice1 Slice2 Slice3 Slice4 Slice5 Slice6 Slice7 Slice8

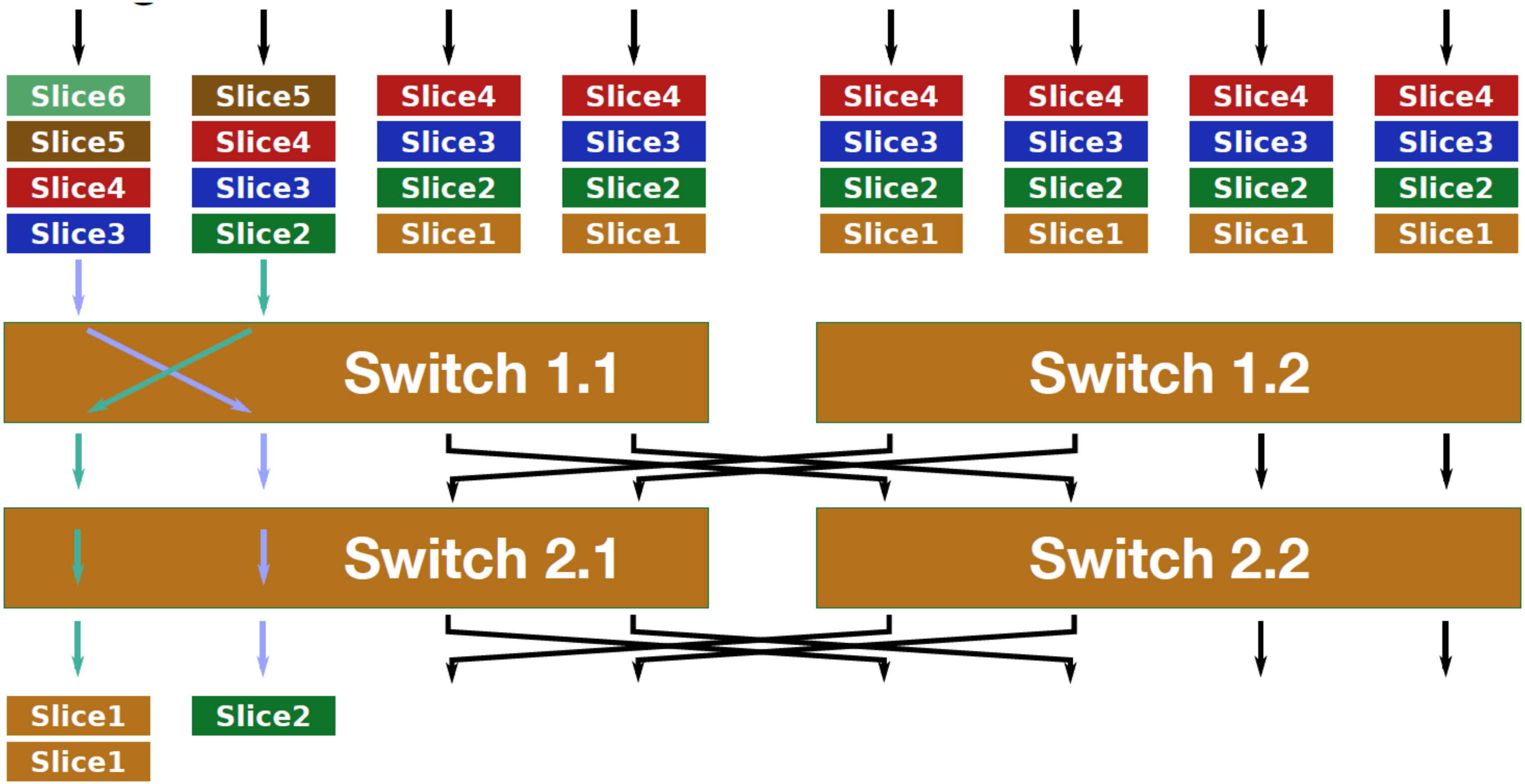
Slice1 Slice2 Slice3 Slice4 Slice5 Slice6 Slice7 Slice8

Slice1 Slice2 Slice3 Slice4 Slice5 Slice6 Slice7 Slice8

Slice1 Slice2 Slice3 Slice4 Slice5 Slice6 Slice7 Slice8

Switch Operation





Slice6
Slice5
Slice4
Slice3

Slice5
Slice4
Slice3
Slice2

Slice4
Slice3
Slice2
Slice1

Slice4
Slice3
Slice2
Slice1

Slice4
Slice3
Slice2
Slice1

Slice4
Slice3
Slice2
Slice1

Slice4
Slice3
Slice2
Slice1

Slice4
Slice3
Slice2
Slice1

Switch 1.1

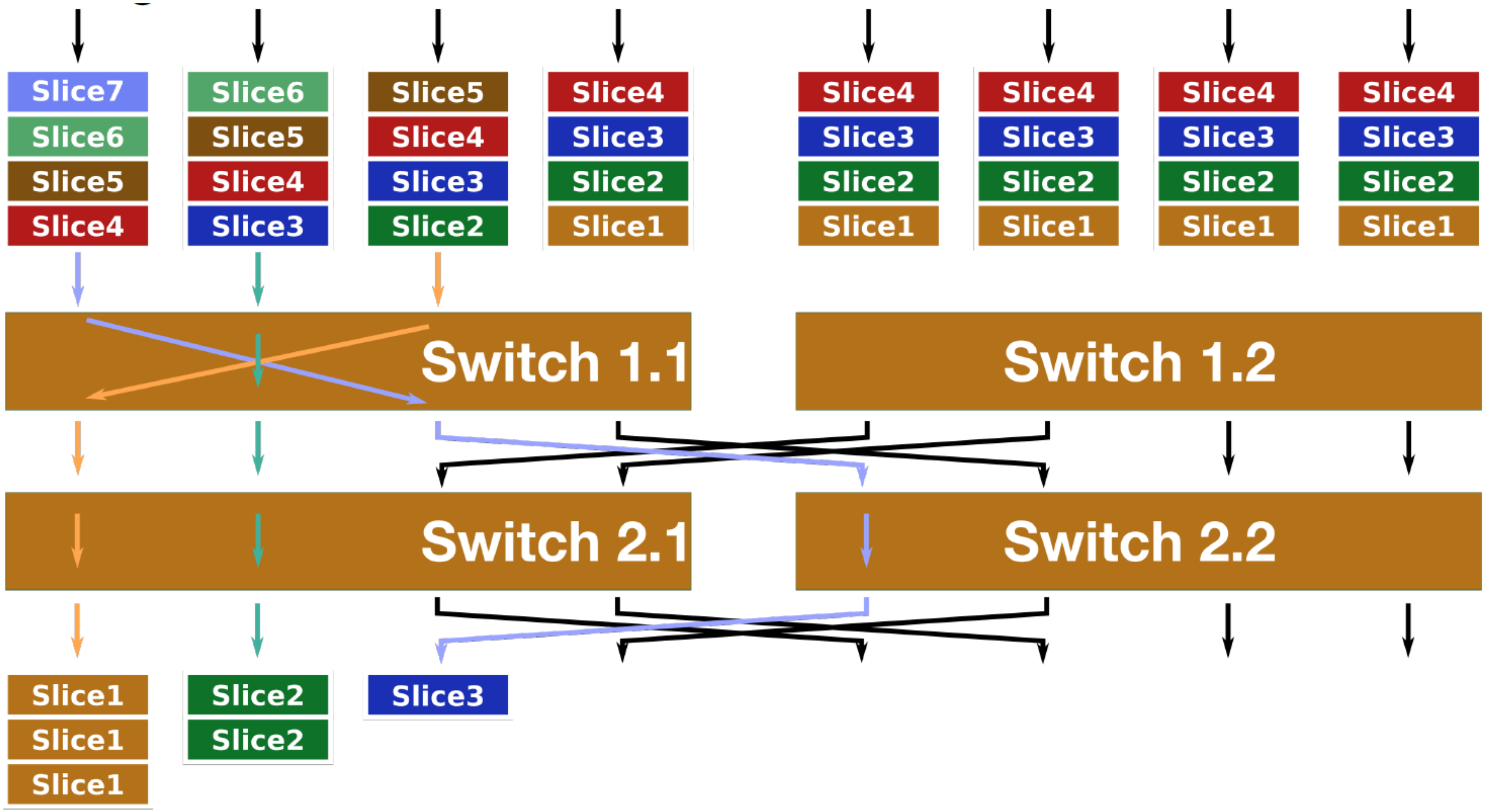
Switch 1.2

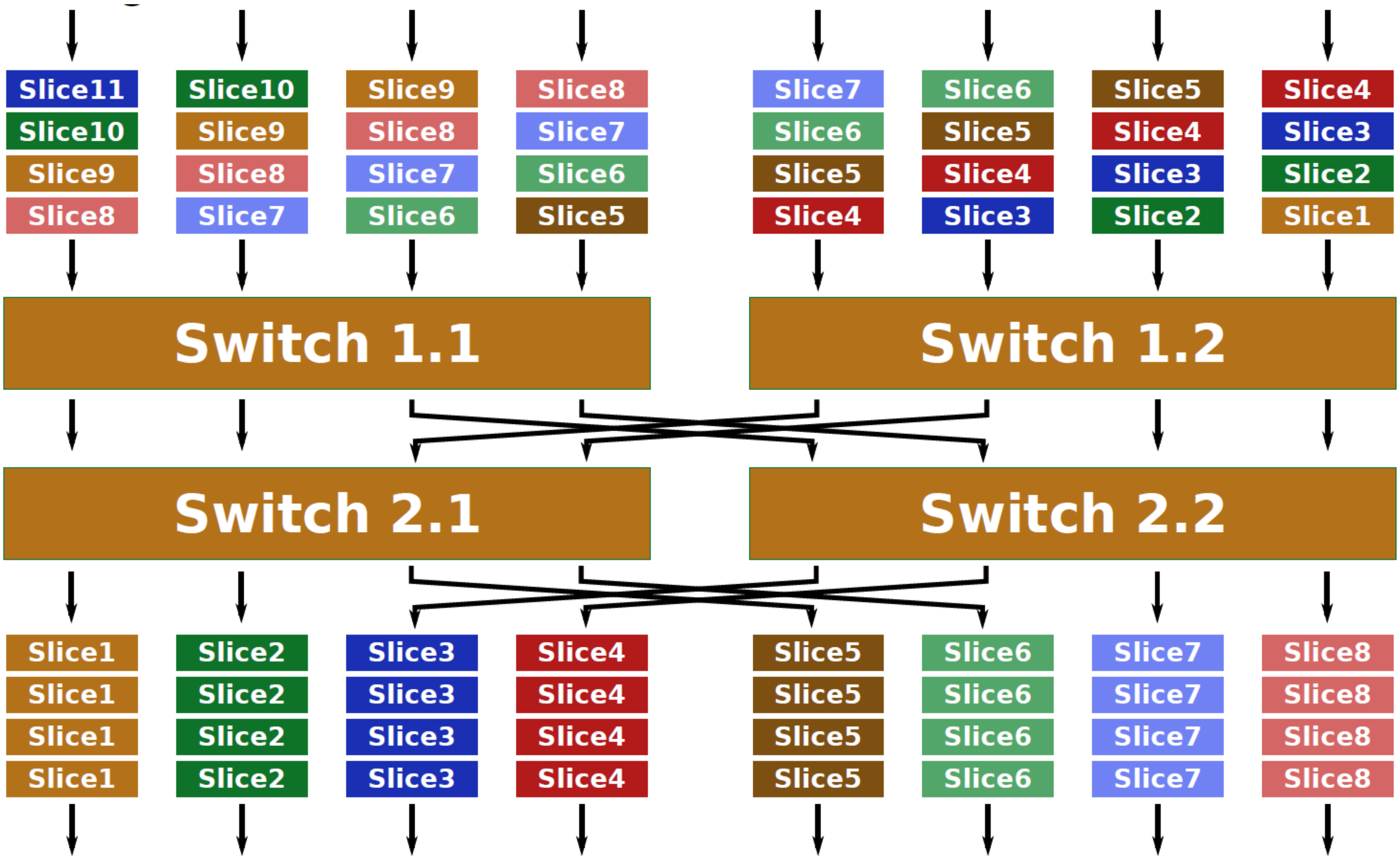
Switch 2.1

Switch 2.2

Slice1
Slice1

Slice2


Slice7
Slice6
Slice5
Slice4
Slice6
Slice5
Slice4
Slice3
Slice5
Slice4
Slice3
Slice2
Slice4
Slice3
Slice2
Slice1
Slice4
Slice3
Slice2
Slice1
Slice4
Slice3
Slice2
Slice1
Slice4
Slice3
Slice2
Slice1
Slice4
Slice3
Slice2
Slice1
Switch 1.1
Switch 1.2
Switch 2.1
Switch 2.2
Slice1
Slice1
Slice1
Slice2
Slice2
Slice3



Slice11 Slice10 Slice9 Slice8 Slice7 Slice6 Slice5 Slice4
Slice10 Slice9 Slice8 Slice7 Slice6 Slice5 Slice4 Slice3
Slice9 Slice8 Slice7 Slice6 Slice5 Slice4 Slice3 Slice2
Slice8 Slice7 Slice6 Slice5 Slice4 Slice3 Slice2 Slice1

Switch 1.1 Switch 1.2

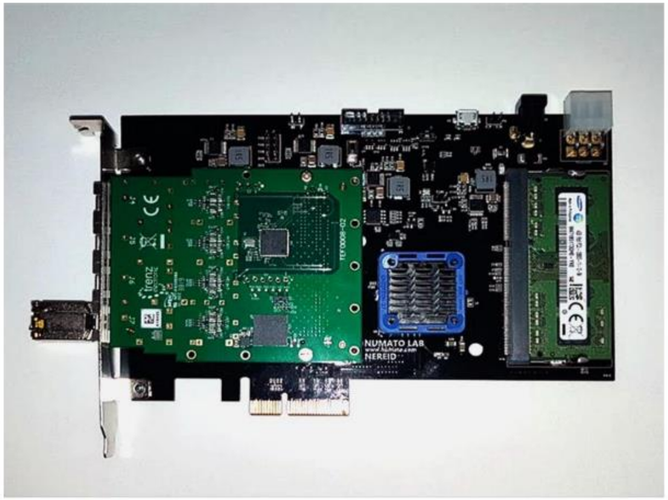
Switch 2.1 Switch 2.2

Slice1 Slice2 Slice3 Slice4 Slice5 Slice6 Slice7 Slice8
Slice1 Slice2 Slice3 Slice4 Slice5 Slice6 Slice7 Slice8
Slice1 Slice2 Slice3 Slice4 Slice5 Slice6 Slice7 Slice8
Slice1 Slice2 Slice3 Slice4 Slice5 Slice6 Slice7 Slice8

Spill Buffer PCIE Card

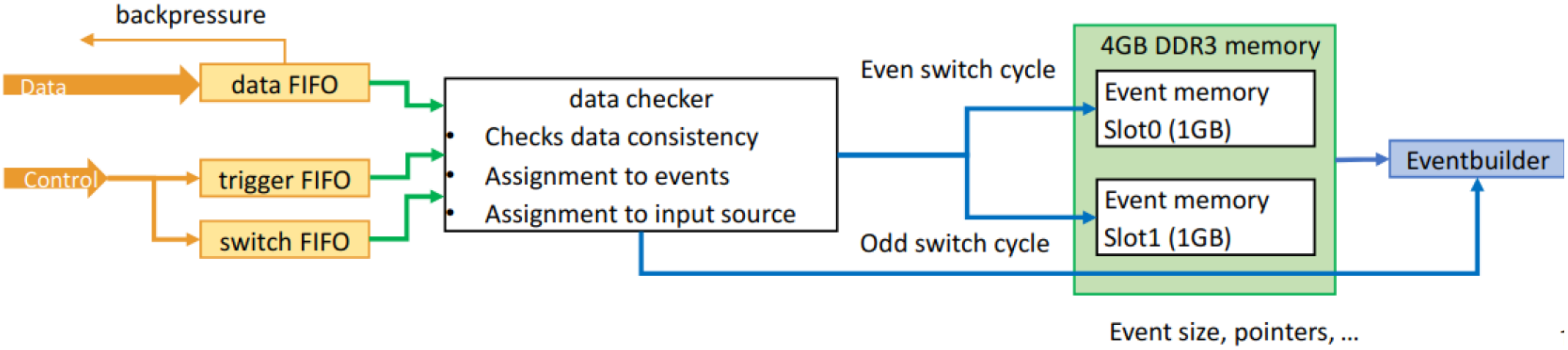
Commercial hardware

- Nereid Kintex 7 PCI Express
- Kintex 7 XC7K160T FPGA
- 4 x PCIe-Gen2 Interface
- 4 GB DDR3 memory



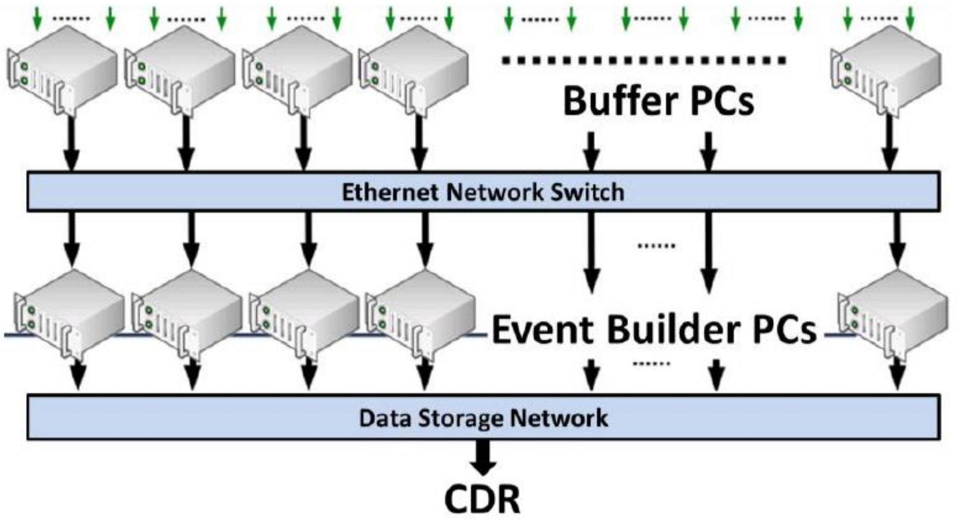
Spill Buffer Firmware

- Single 6.25 Gb/s 8b/10b Aurora interface
  - Data
  - Trigger information
  - Switch configuration
- Events stored in DDR3 memory
- Combination of events according to
  - event number
  - switch configuration
- Built events pushed to PCIe
- Internal bandwidth 3 GB/s

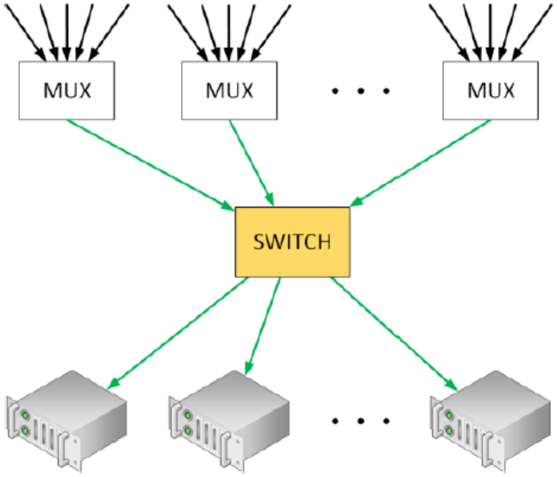


Event Building CPU vs FPGA

CPU



FPGA



Advantages:

- Uses mass-produced components
- Easy integration of redundancy elements
- Availability of libraries and templates

Disadvantages:

- Throughput limited by EB network
- Performance of sequential execution strongly depends on algorithm complexity
- Recovery of crashed processes takes significant time

Advantages:

- Only FPGA allows to build real real-time system
- High scalability
- High reliability
- Low cost

Disadvantages:

- Long firmware development progress: high level simulation tools like System Verilog and OSVVM

⇒ **Motivation**

- Minimize real time SW processes
- Development of highly autamotized and reliable DAQ

iFDAQ

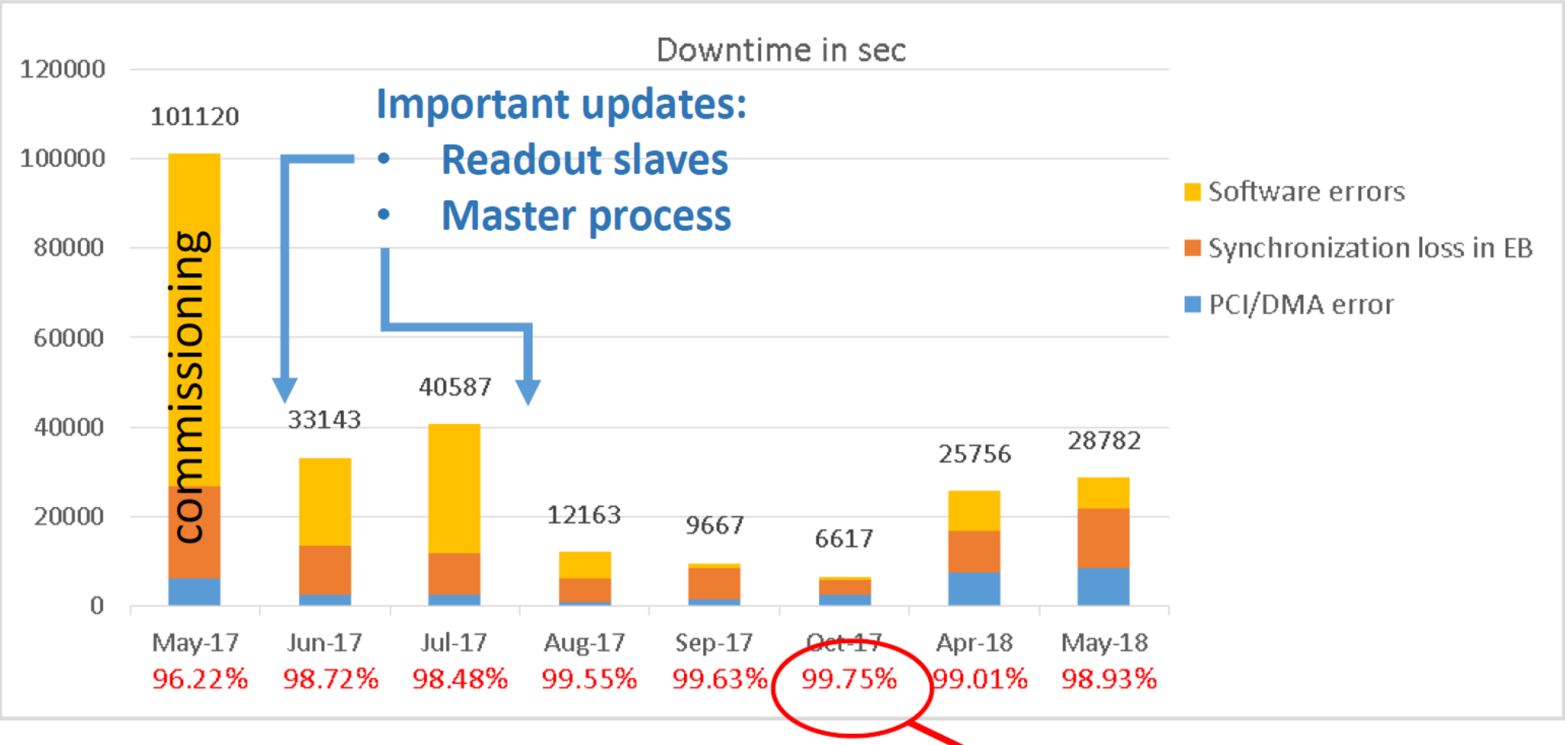
Compact : **Before** : 30 online PCs

Now : one VME 6U crate +

1 rack (8 computers)

Hardware Event Builder

Performance : Up Time in 2017



Important updates:

- **Readout slaves**
- **Master process**

Highly reliable

2. Upgrades of RCCAR software