Intelligent FPGA-based Data Acquisition System

Igor Konorov

Institute for Hadronic Structure and Fundamental Symmetries (E18)

TUM Department of Physics

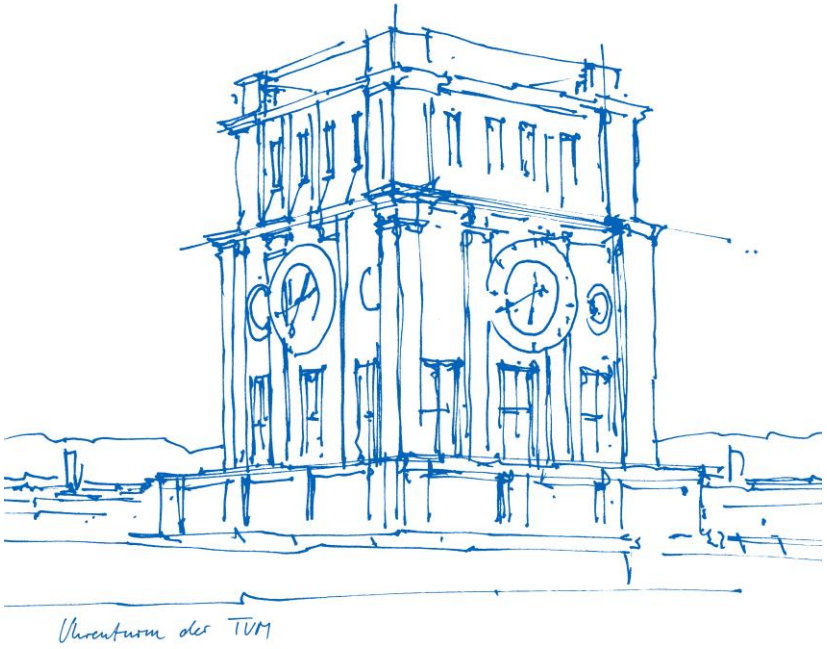
Technical University of Munich

Advanced Workshop on FPGA based

System-on-Chip for Scientific Instrumentation

and Reconfigurable Computing

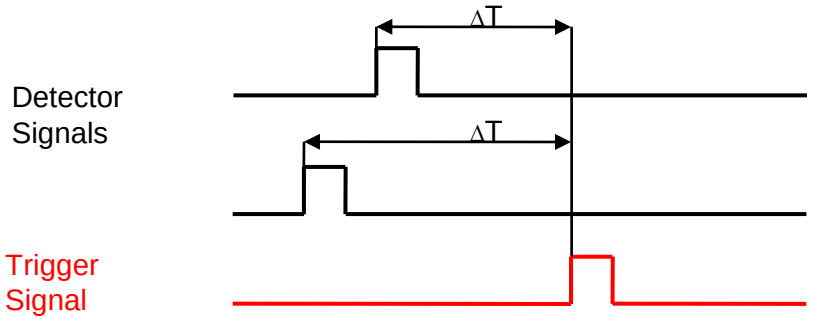
ICTP Trieste



DAQ Elements

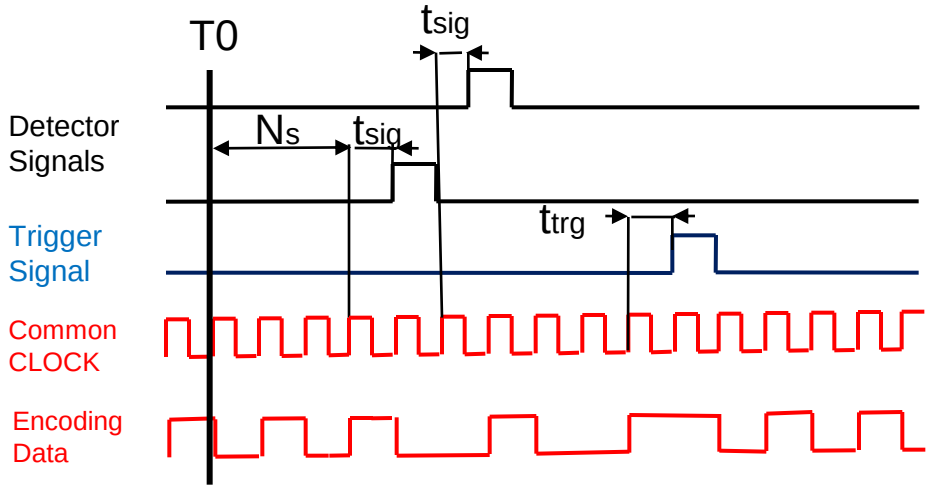
- Front-end electronics, detector specific
  - Conversion of detector analog signal to digital form
  - Derandomization
  - Data processing: signal detection, extraction of signals' parameters **Time** and/or **Amp…**
- Trigger Logic
  - reduce amount of stored data
  - define time when something interesting happen
- Trigger Distribution system => Time Distribution System
- Slow Control System
  - Control and monitoring of PS, Gas system, Temperature, Humidity,…
  - Programming of Front-ends
- Acquisition System => Event builder
  - Data acquisition – moving data from FE to PCs
  - Data flow control
  - Real time Software
  - Run control

Few words about time measurements



Classical method:

- TRIGGER is a reference
- SIGNAL time is measured respectively to TRIGGER signal



Alternative method for big experiments:

- Distribute CLOCK , why clock?
  - Easier to distribute with very low jitter
- Measure absolute time respectively to CLOCK phase

$T_{sig} = N_s\, T_{clk} + t_{sig}$

$T_{trg} = N_t\, T_{clk} + t_{trg}$

Clock and Data are encoded and transmitted from single source to multiple destinations

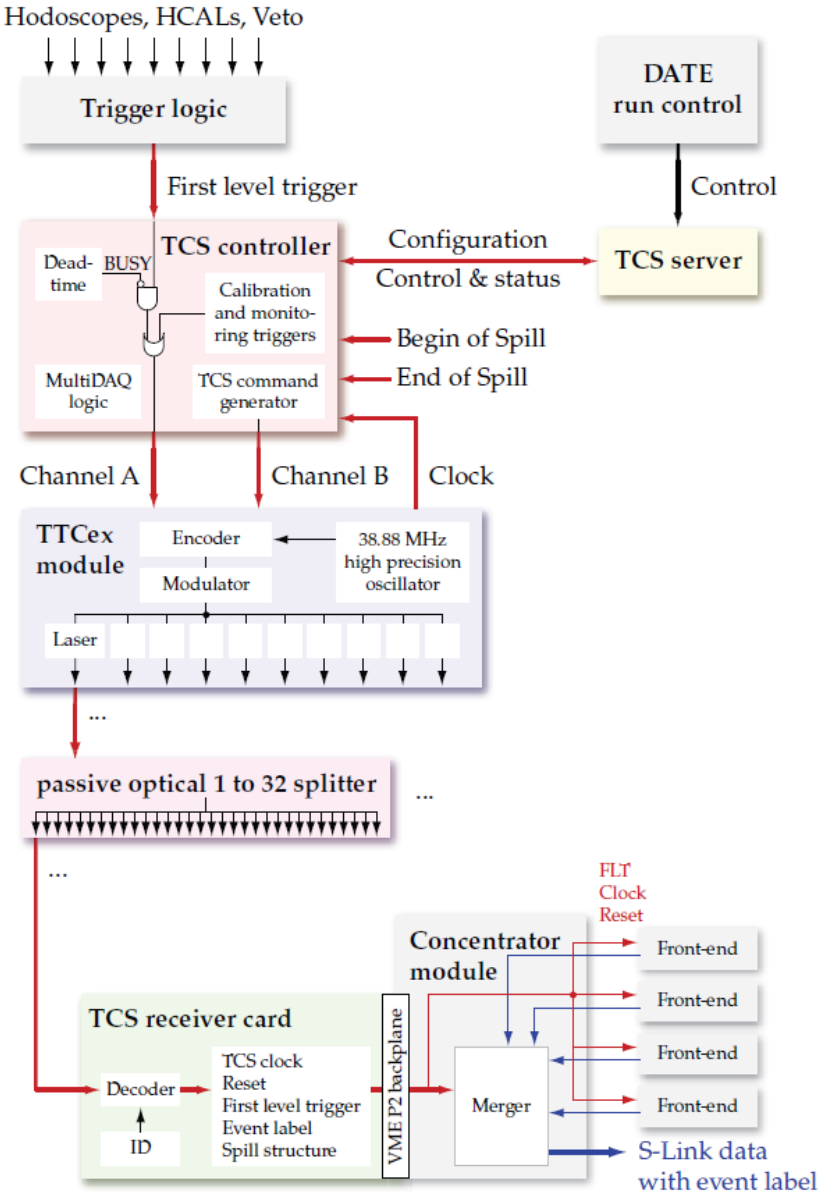
NA48, LHC->TTC, COMPASS->TCS

Time Distribution System

Trigger Control System

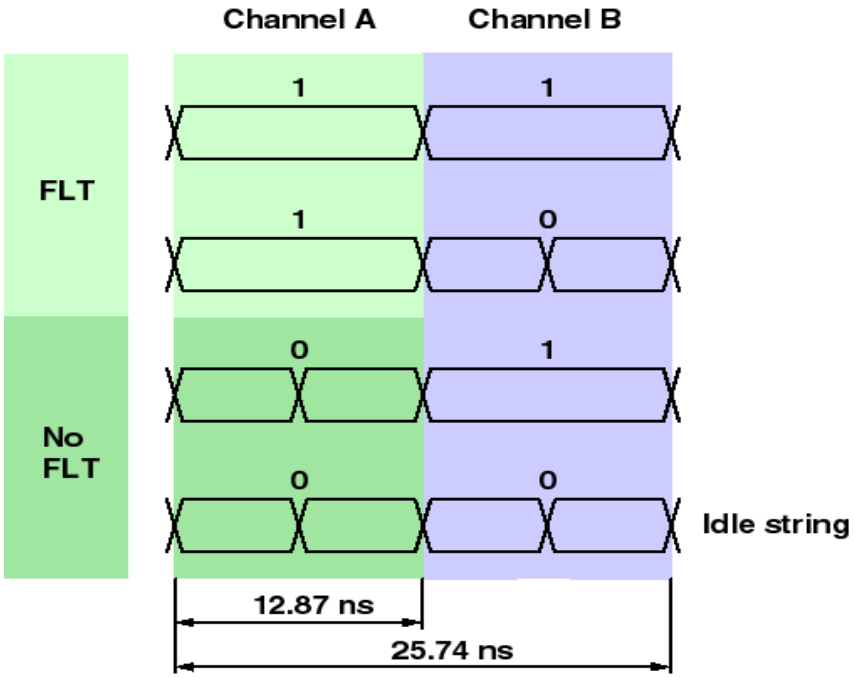
Features:

- Optical network
- Star-like topology with single source and multiple destinations
- Active fan out 1:16
- Passive fan out using optical splitters 1:32
- Unidirectional transmission
- Speed 155 Mbaud
- Two independent channels A and B
- A – trigger
- B – commands and trigger Number
- TCS controller and TCS receiver are implemented in FPGA



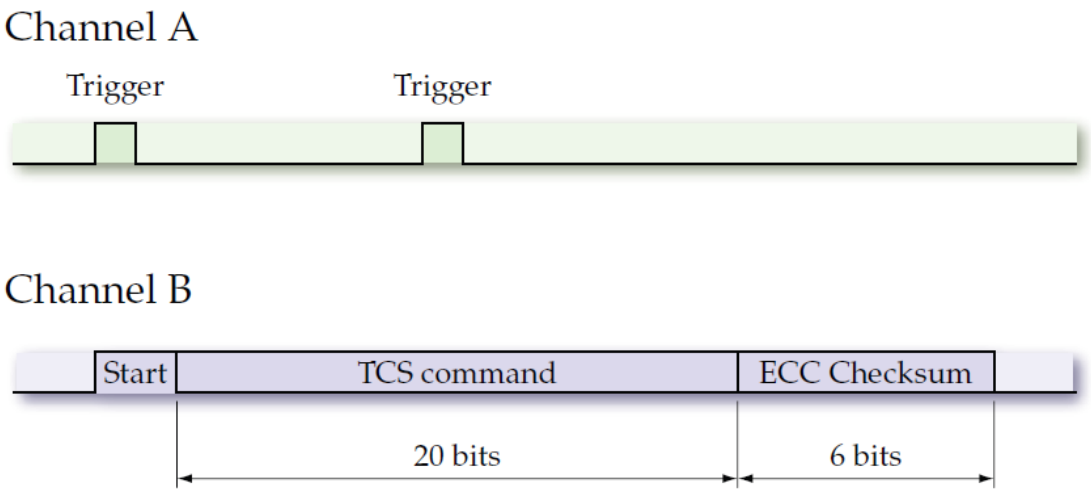
Encoding

Biphase mark and Time-Division-Multiplexed encoding



- **155.52 Mbaud link speed**
- **77.76 Mb/sec can be used**
- **2 independent channels**
- **speed of one channel 38.88 Mb/sec**
- **channel A for FLT(first level trigger)**
- **channel B for data**

Command Format



Broadcast command:
- **reset, beginning of spill, end of spill**
- **event number , spill number, trigger type**
- **pre trigger command for calibration/monitoring trigger**

Addressed command:
- **enabling/disabling TCS receivers**
- **configuration of TCS receiver**

FPGA Firmware

Trigger signals

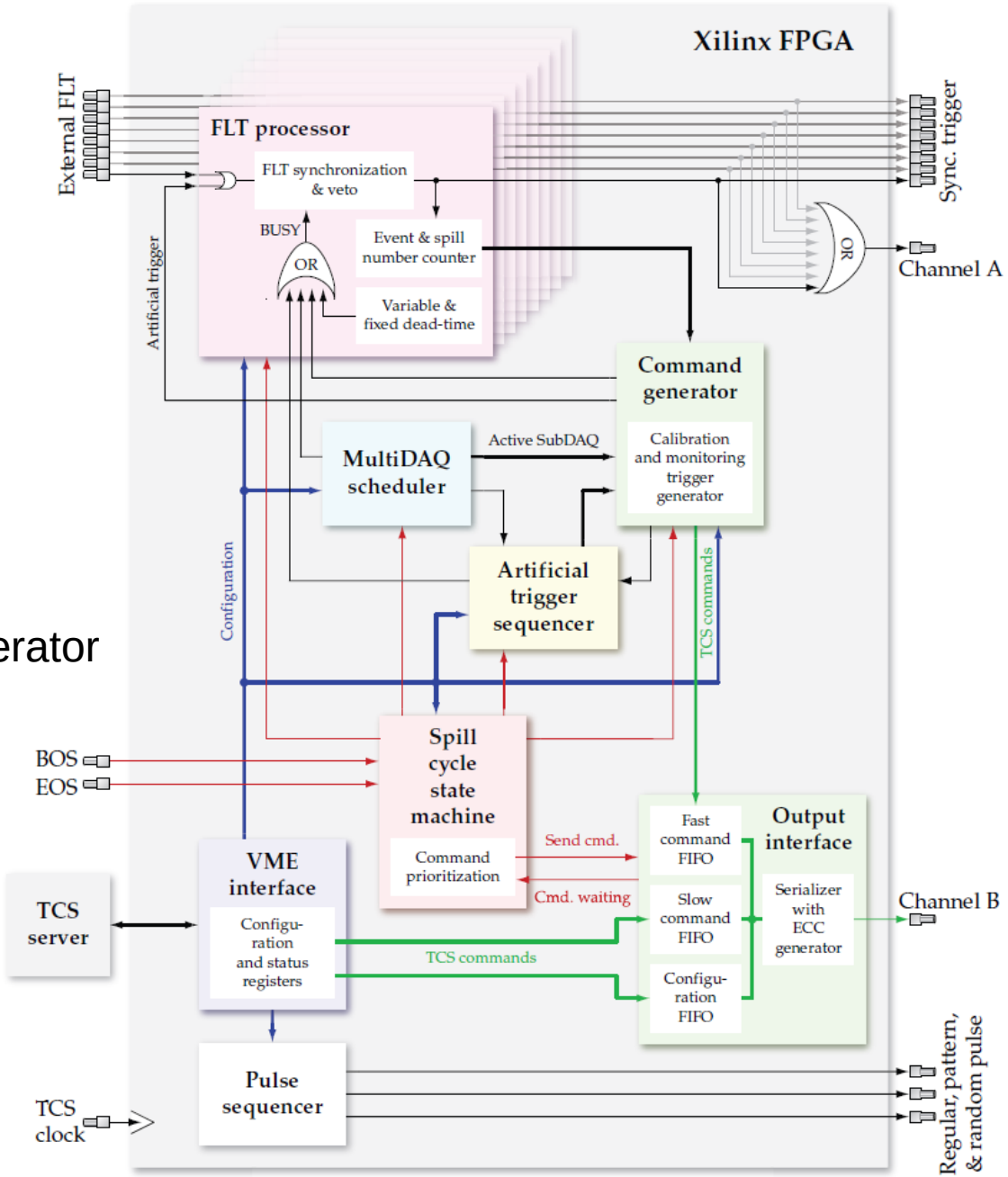
Synchronization with SPS accelerator

- Start Of Spill
- End Of Spill

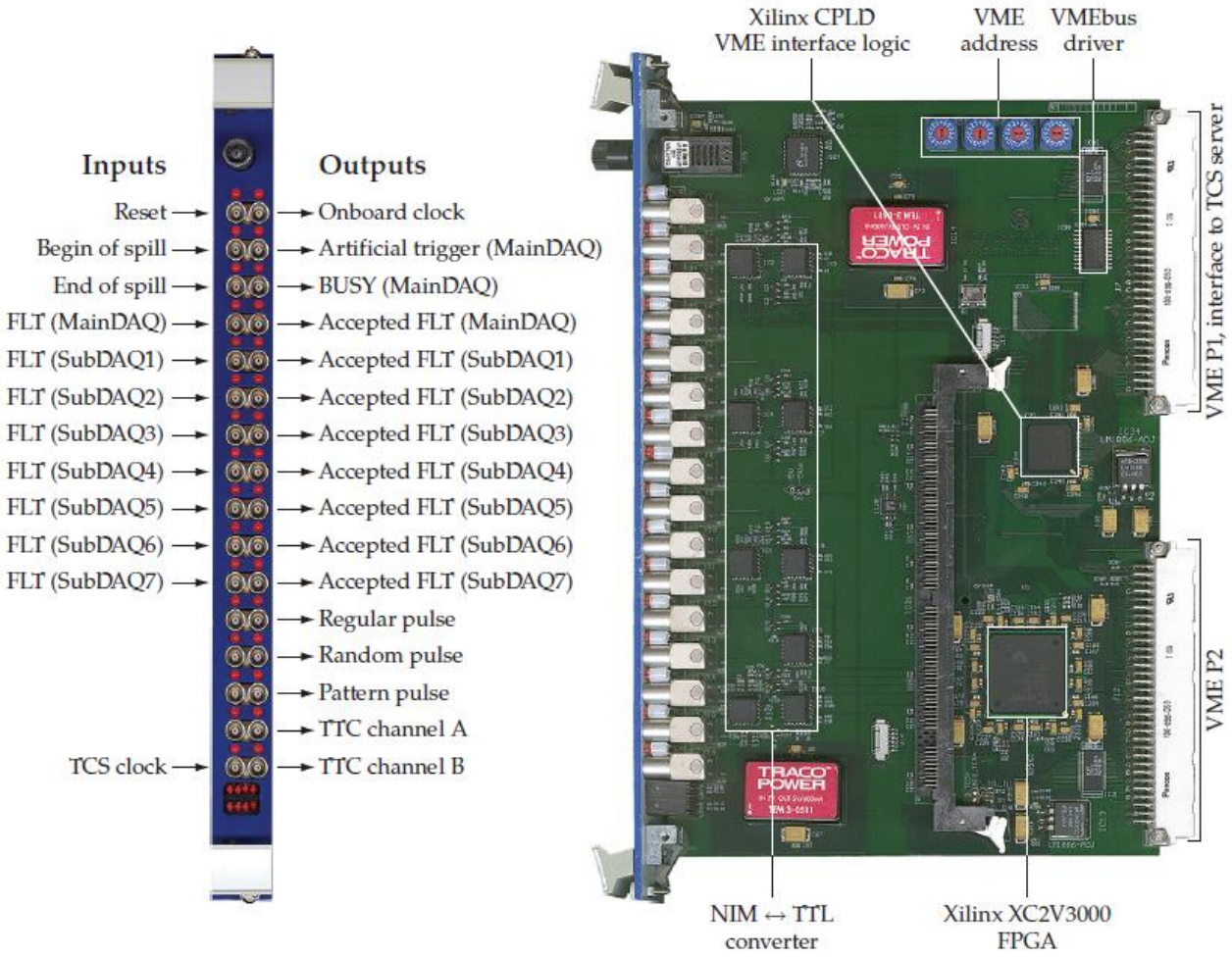
TCS Server

- System Configuration
- Start of Run
- Stop of Run

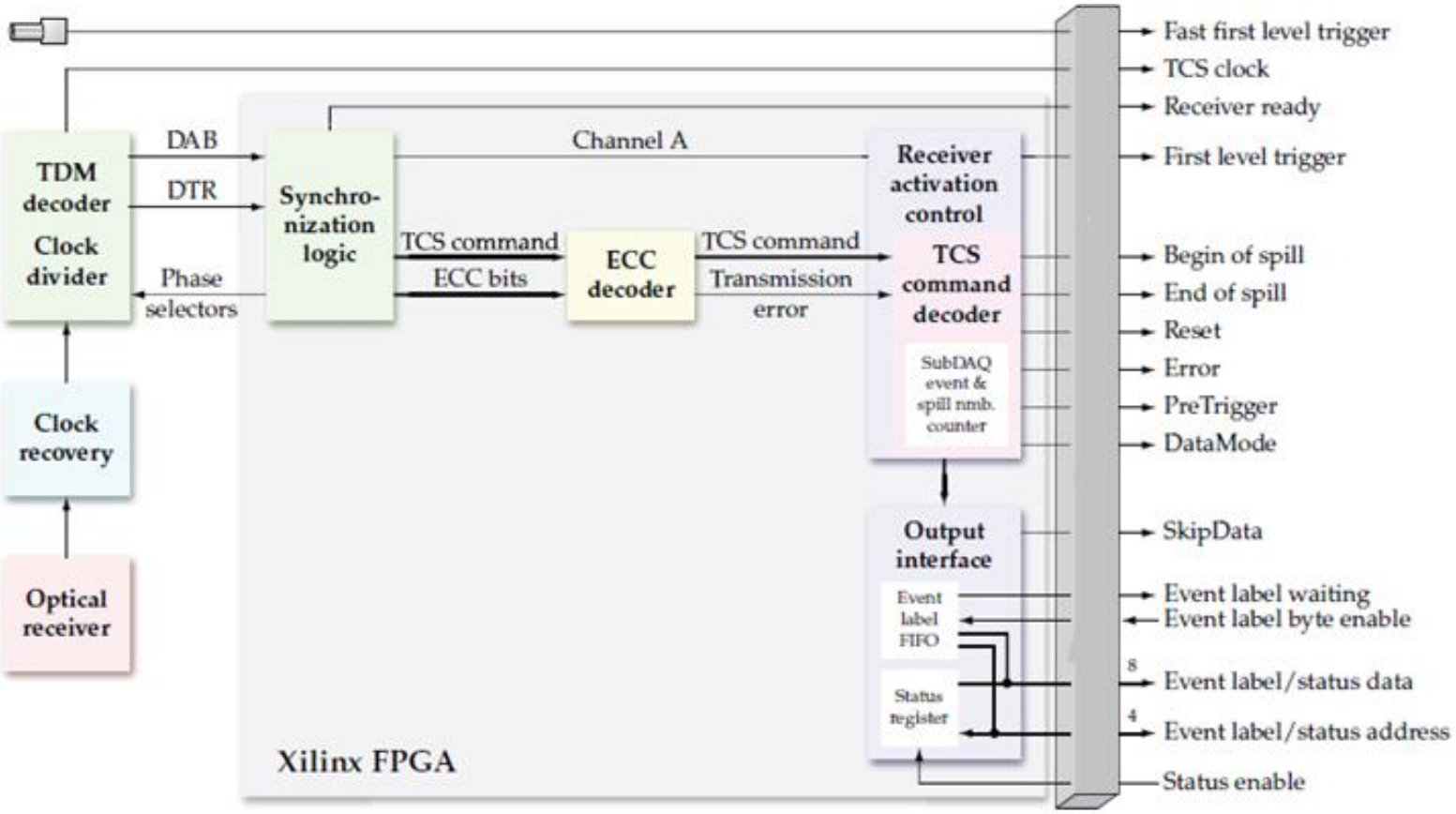
Low jitter TCSystem Clock 38.88 MHz



TCS Controller



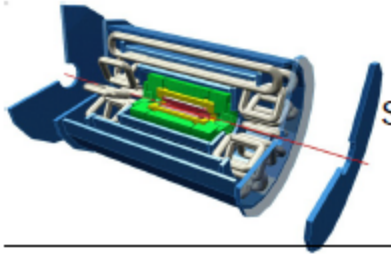
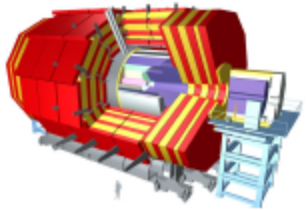
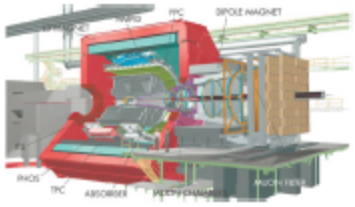
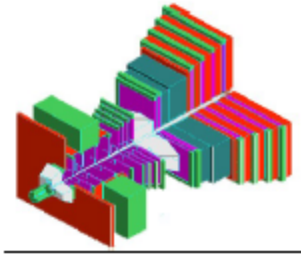
TCS Receiver



EVENT BUILDING

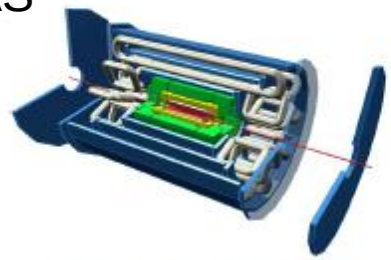
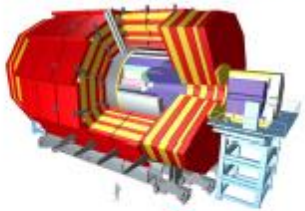
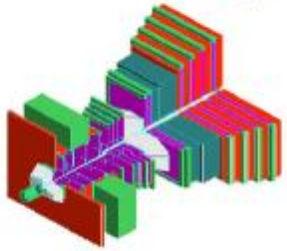
Slide from ISOTDAQ – international school of Trigger and DAQ organized by CERN

Readout Links of LHC Experiments

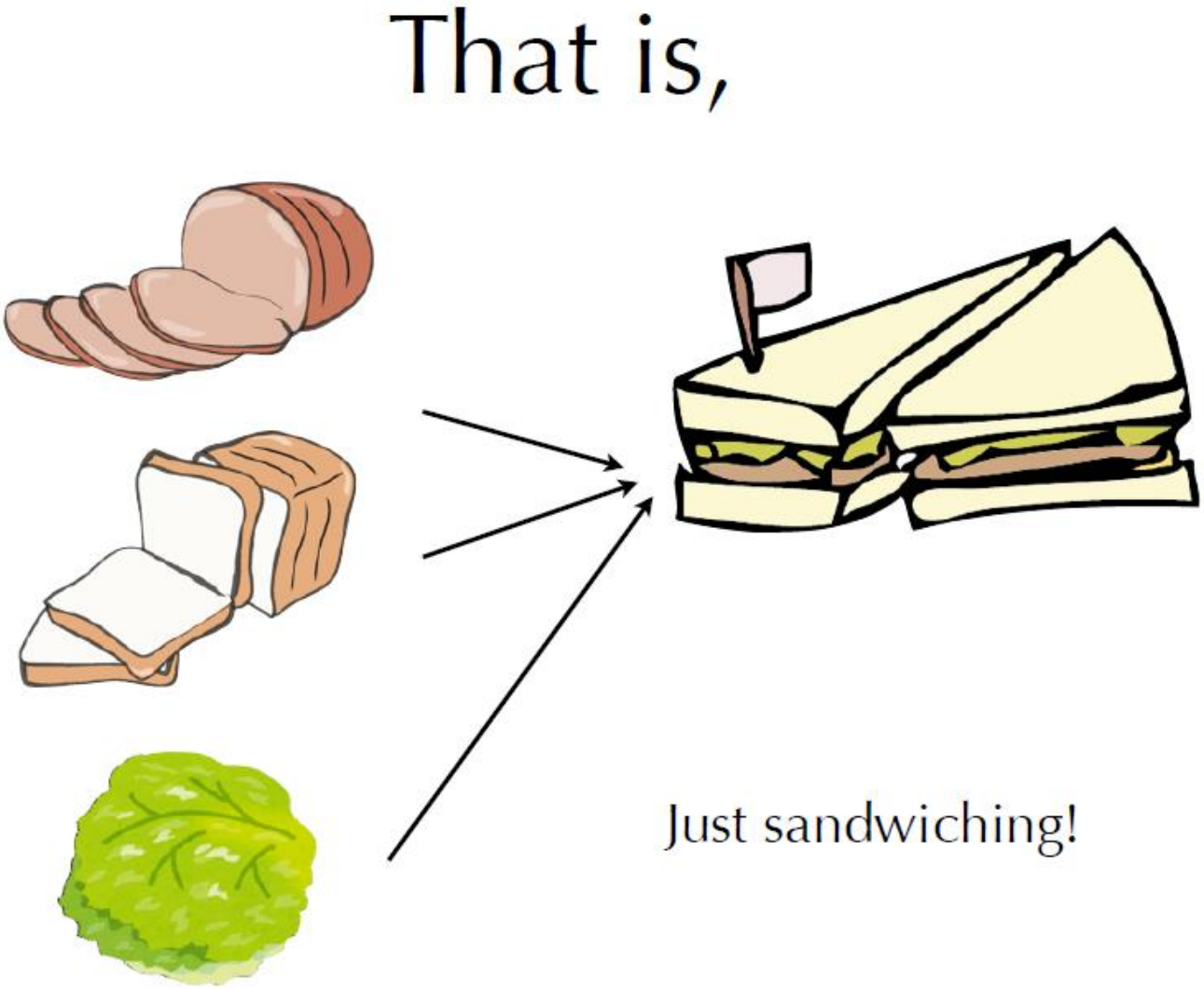
			Flow Control
	SLINK	Optical: 160 MB/s ≈ 1600 Links Receiver card interfaces to PC.	Yes
	SLINK 64	LVDS: 400 MB/s (max. 15m) ≈ 500 links (FE on average: 200 MB/s to readout buffer) Receiver card interfaces to commercial NIC (Network Interface Card)	yes
	DDL	Optical 200 MB/s ≈ 500 links Half duplex: Controls FE (commands, Pedestals,Calibration data) Receiver card interfaces to PC	yes
	TELL-1 & GbE Link	Copper quad GbE Link ≈ 400 links Protocol: IPv4 (direct connection to GbE switch) Forms “Multi Event Fragments” Implements readout buffer	no

LHC experiments, Run1

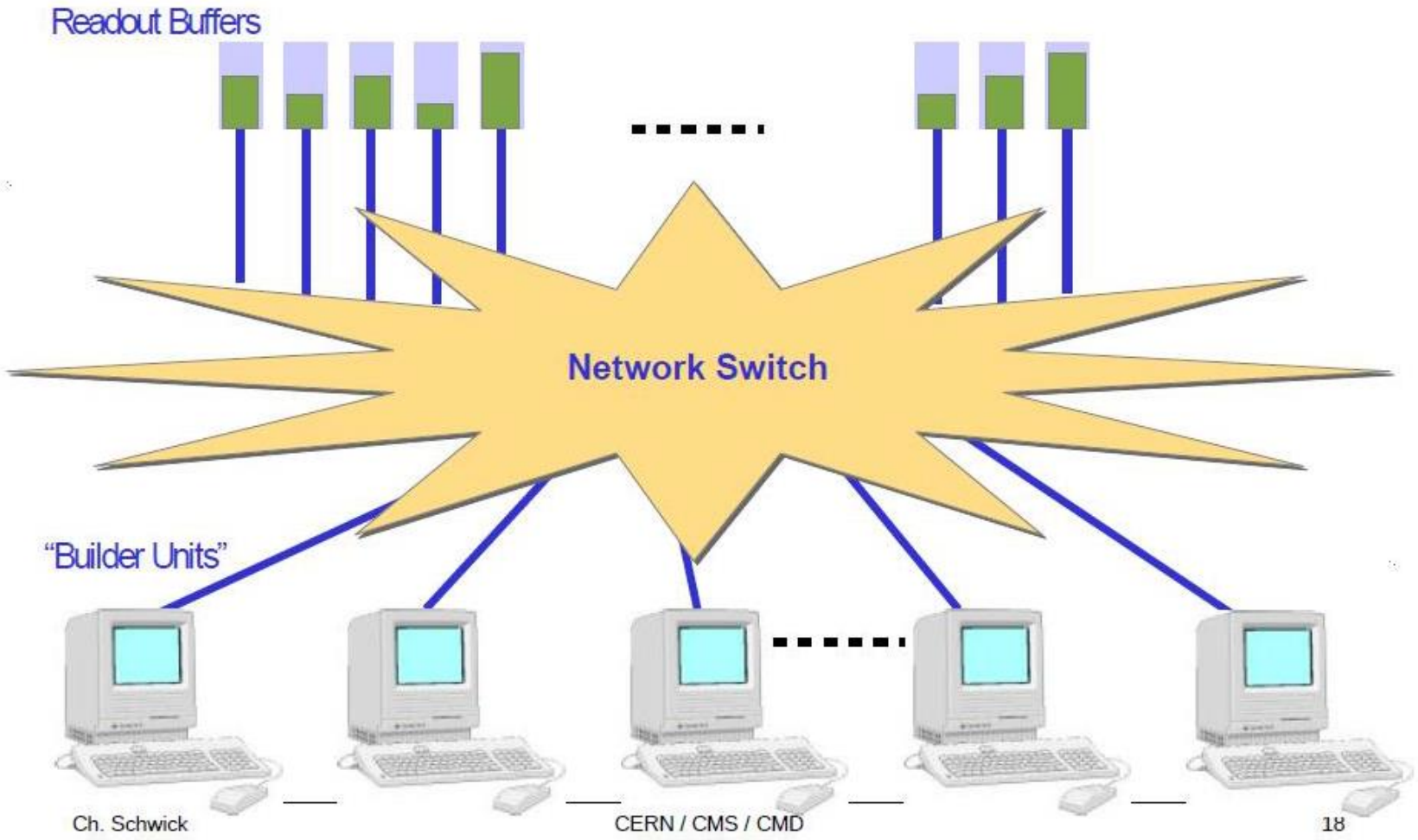
Slide from ISOTDAQ

		No.Levels Trigger	Lvl 0,1,2 Rate (Hz)	Event Size (Byte)	Evt Build. Bandw.(GB/s)	High Level Trigger HLT Out MB/s (Event/s)
ATLAS		3	LV-1 105 LV-2 3x103	1.5 MB	4.5	300 (200)
CMS		2	LV-1 105 Pb-Pb 1500MB/s	1.0 MB	100	300 (200)
LHCb		2	LV-0 106	30 kB	30	60 (2 kHz)
ALICE		4	Pb-Pb 500 p-p 103	70 MB 2 MB	25	1250 (100) 200 (100)

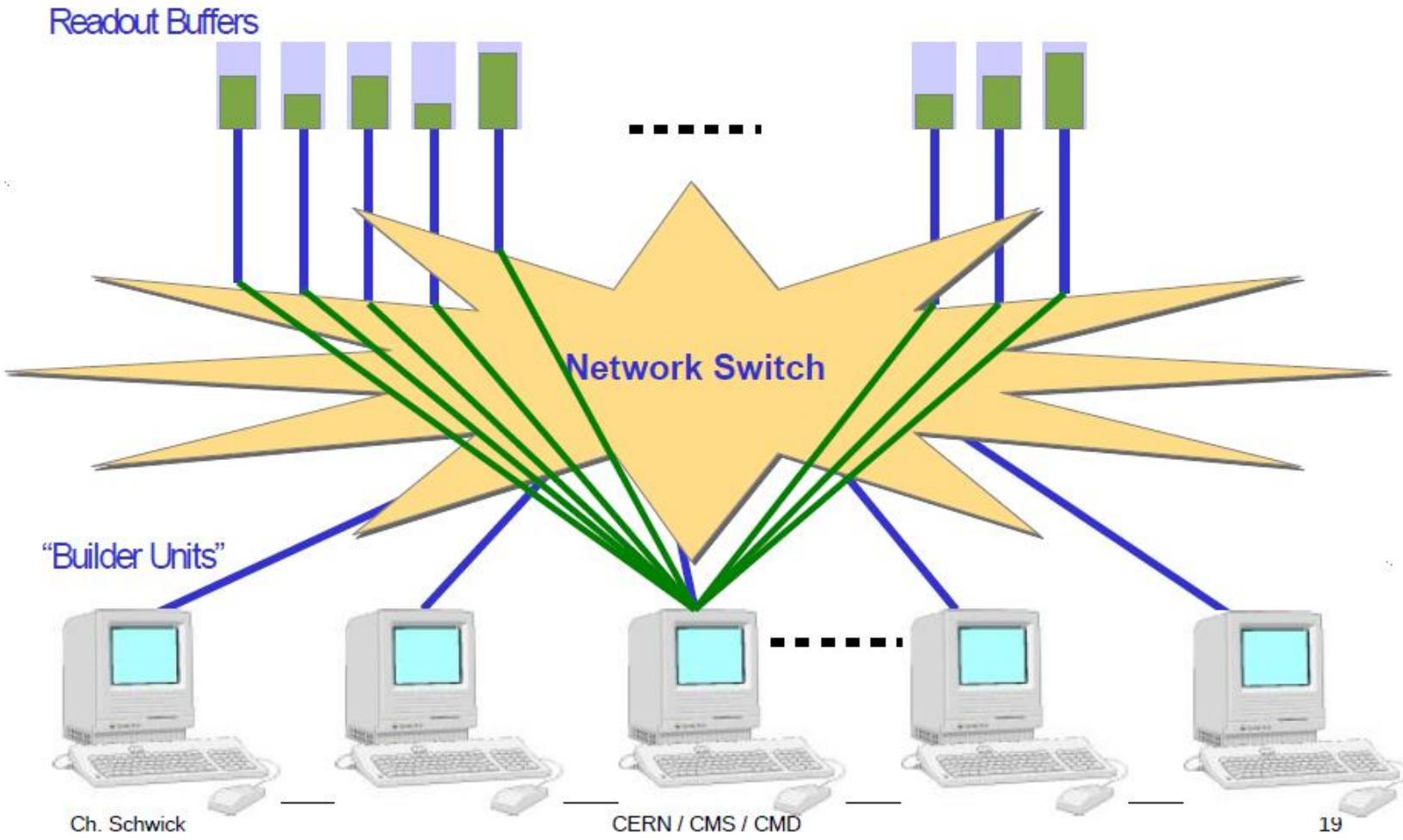
What is Event Building



Event Building Challenges I



Event Building Challenges II



Traffic Pattern Causes Congestion Problem

Commercial Switches for DAQ

Commercial switches aim for high bandwidth and **low latency**

How to use switches

- Shaping Data Traffic – data flow manager
- Employing Switches with back pressure and big internal memory (expensive)
- Employing switches with significantly bigger bandwidth than needed to minimize congestion

Novel iFDAQ Architecture

intelligent **F**PGA-based **DAQ**

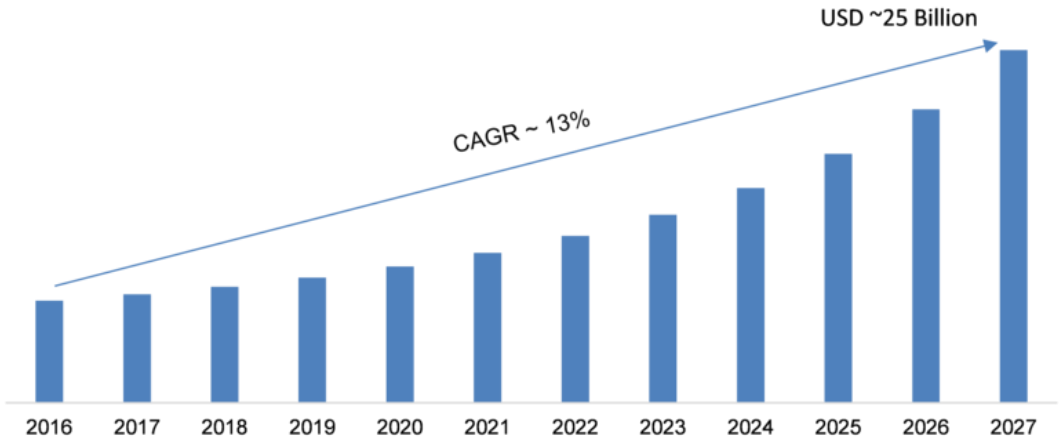
Motivation for iFDAQ

Progress of FPGA technology

Chip	Manufacturer	Technology	Transistor count
Duo-core + GPU Iris Core i7 Broadwell-U	Intel	14 nm	1 900 000 000
22-core Xeon Broadwell-E5	Intel	14 nm	7 200 000 000
Virtex 7	Xilinx	28 nm	6 800 000 000
Virtex Ultra Scale	Xilinx	20 nm	20 000 000 000

- High bandwidth of serial links: 3000 Gbps per single FPGA
- High bandwidth of external memory interfaces 10 GB/s per single FPGA

FPGA Market Expectation



https://www.marketresearchfuture.com/reports/field-programmable-gate-array-market-1019

FPGA technology advantages

- Emerging technology, rapidly extends application fields
- Highly parallel architecture
- Enormous IO bandwidth
- Low cost
- Long development time => Software tools for a moment behind complexity of HW technology

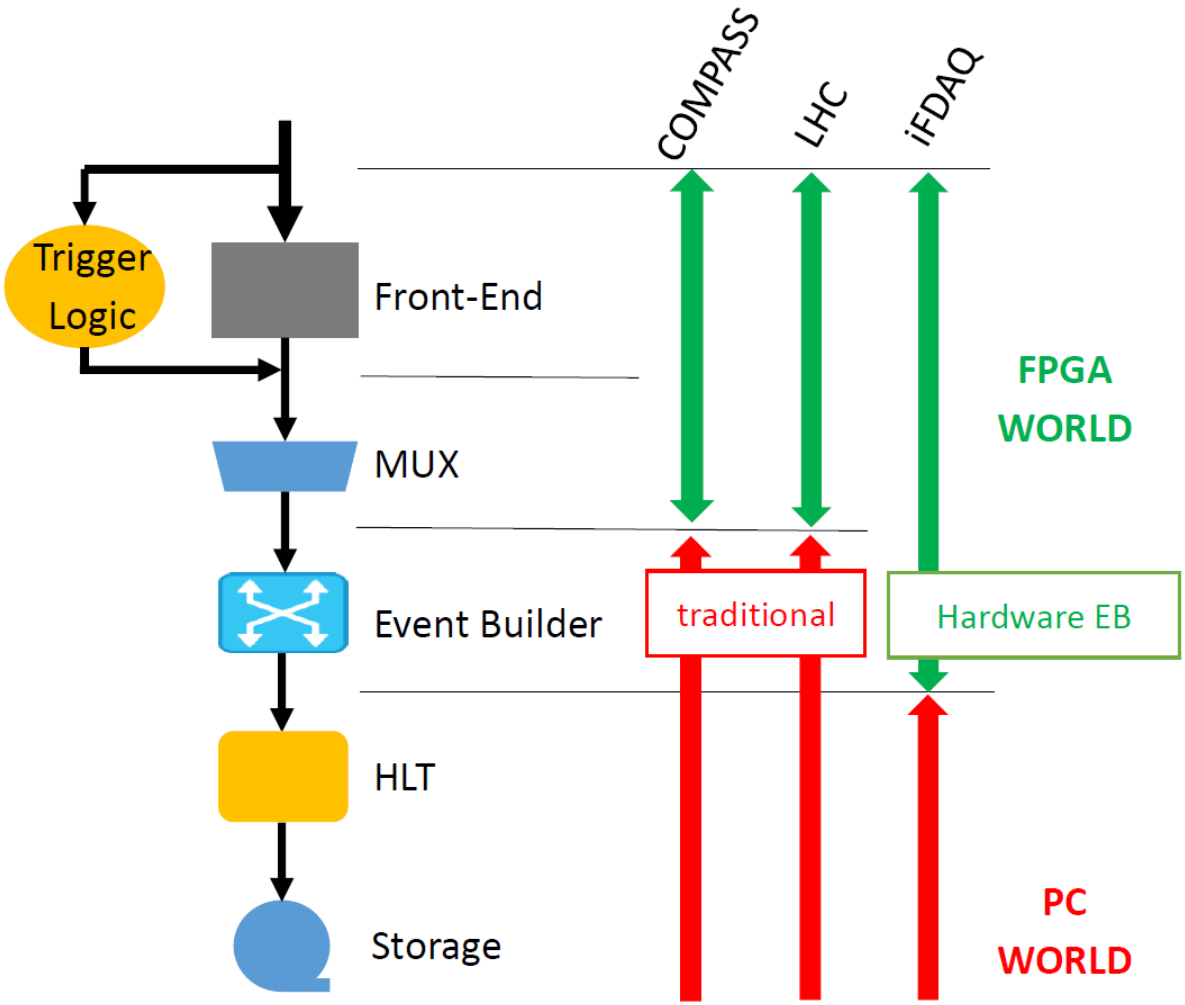
FPGA is ideal technology

for development reliable, high performant, low cost DAQ system

iFDAQ (intelligent FPGA DAQ)

Reliability achieved by smart recovery algorithms included in FPGA

Different DAQ Architectures



Concept of iFDAQ:

Minimize amount of real-time processes and push Event Building into FPGA-world

Feasibility due to recent FPGA developments:

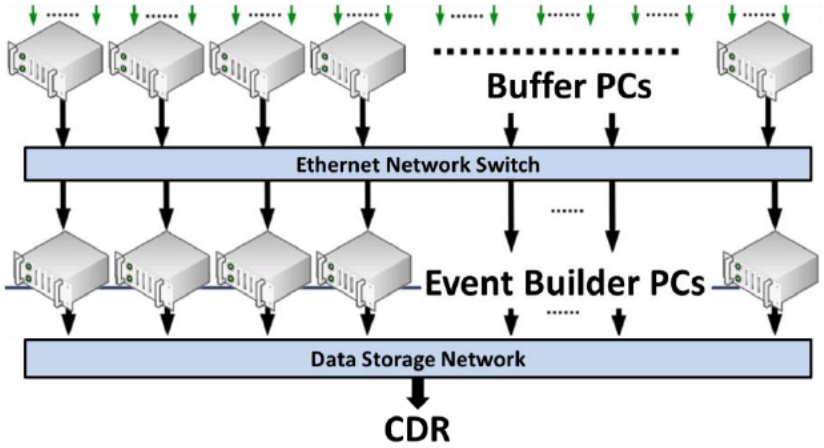
- Increased I/O bandwidth
- Support for high-performance SDRAMs

Expected Advantages:

- Increased Compactness
- Increased Reliability
- Higher Scalability
- Reduced Costs

General Network vs Point to Point

General Network (Ethernet)



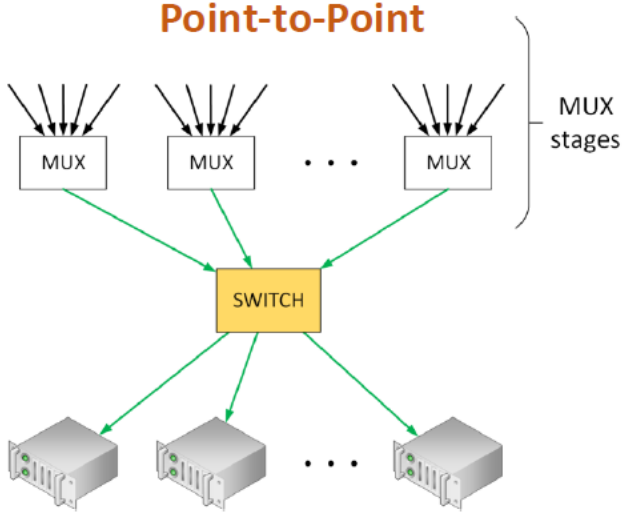
Advantages

- Easy integration of redundancy elements (traffic shaping according to load on nodes)
- Usage of mass-produced components and standards

Disadvantages:

- Throughput limited by EB-network switch
- Inefficient usage of max. bandwith due to:
  - Improper comm. pattern (N senders -> 1 receiver) => network congestion
  - Data overhead due to addressing etc.

Point-to-Point



Advantages:

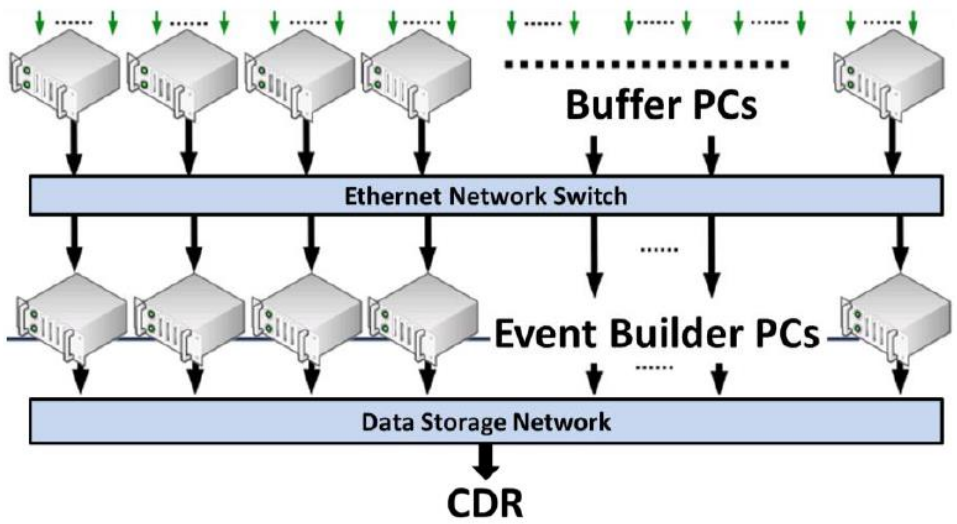
- Independence of network switch
- Efficient usage of link bandwidth (no addressing etc.)
- High reliability

Disadvantages:

- Strong dependence on reliability of network nodes (no rerouting possibility in case of hardware failure)
- No possibility for dynamic network optimization (e.g. load balancing)

Event Building : CPU vs FPGA

CPU



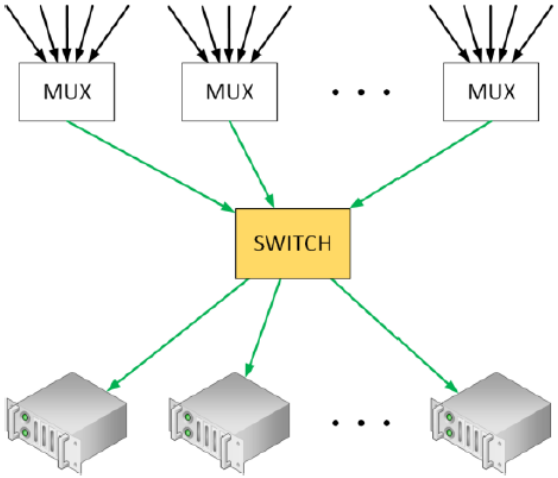
Buffer PCs

- Buffering and scheduling data transmission Event Builder PC
- Replicated over computers to fit performance needs

Event Builder PCs

- Collect event fragments and combines them into complete event
- Replicated over computers to fit through put requirements

FPGA



MUX

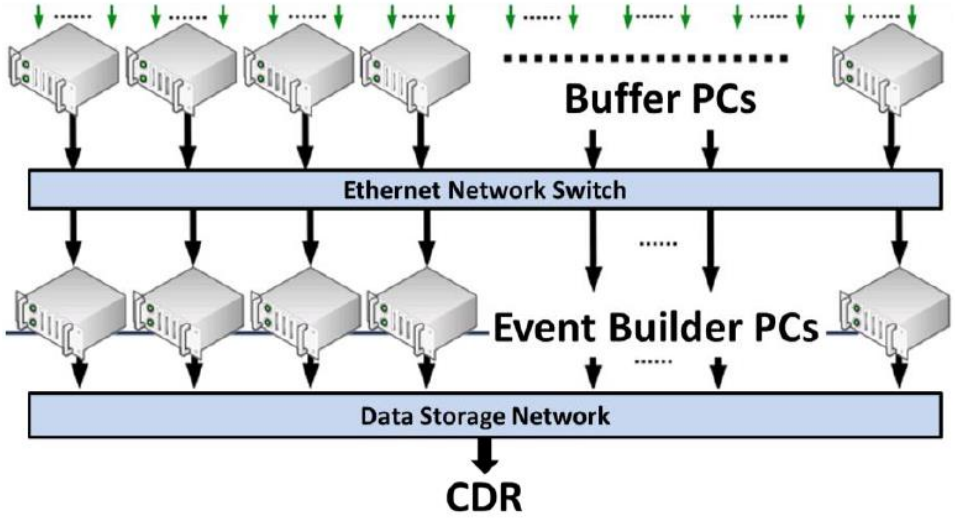
- Reduction of number of links
- Buffering and subevent building for efficient usage of serial interfaces

SWITCH

- Parallel execution of event building processes
- Distribution of complete events to different compute nodes

Event Building CPU vs FPGA

CPU



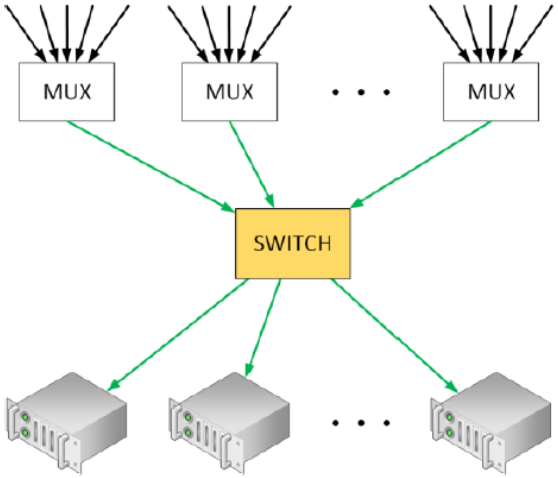
Advantages:

- 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

FPGA



Advantages:

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

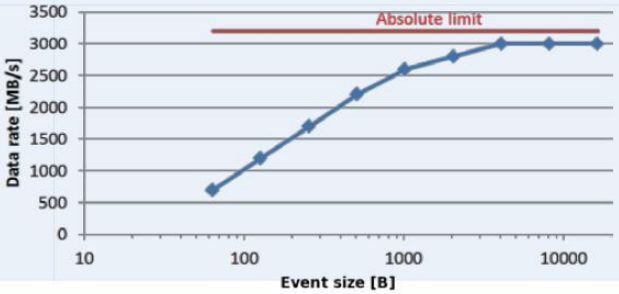
Disadvantages:

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

⇒ **Motivation**

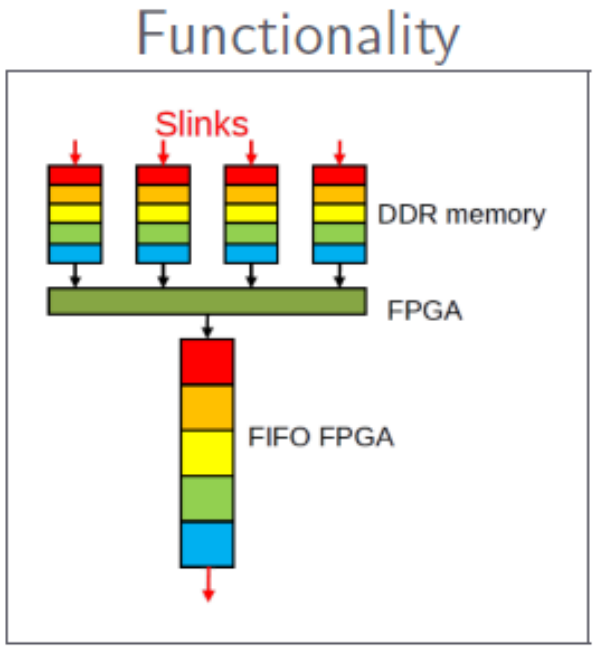
- Minimize real time SW processes
- Development of highly automotized and reliable DAQ

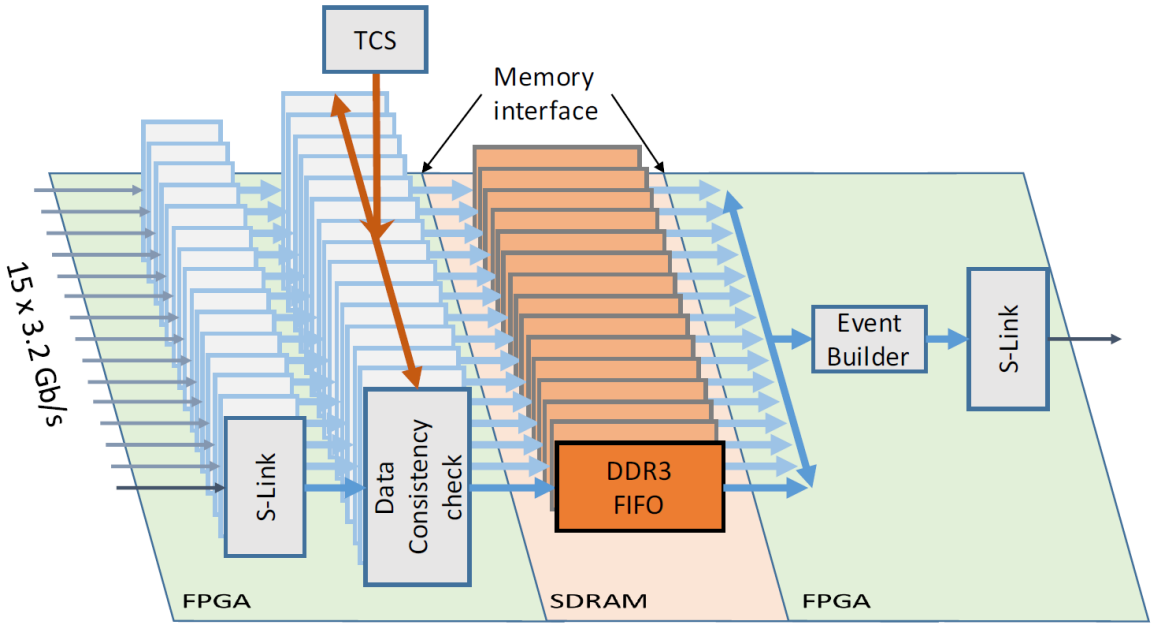
Data Handling Card

AMC module	
○ **form factor:**	AMC standard
○ **FPGA:**	Virtex6 XC6VLX130T
○ **memory:**	4 GB DDR3 SDRAM
○ **firmware:**	• **DHCmx** 12:1 multiplexer [1] • **DHCsw** 8x8-switch
○ **data rate:**	

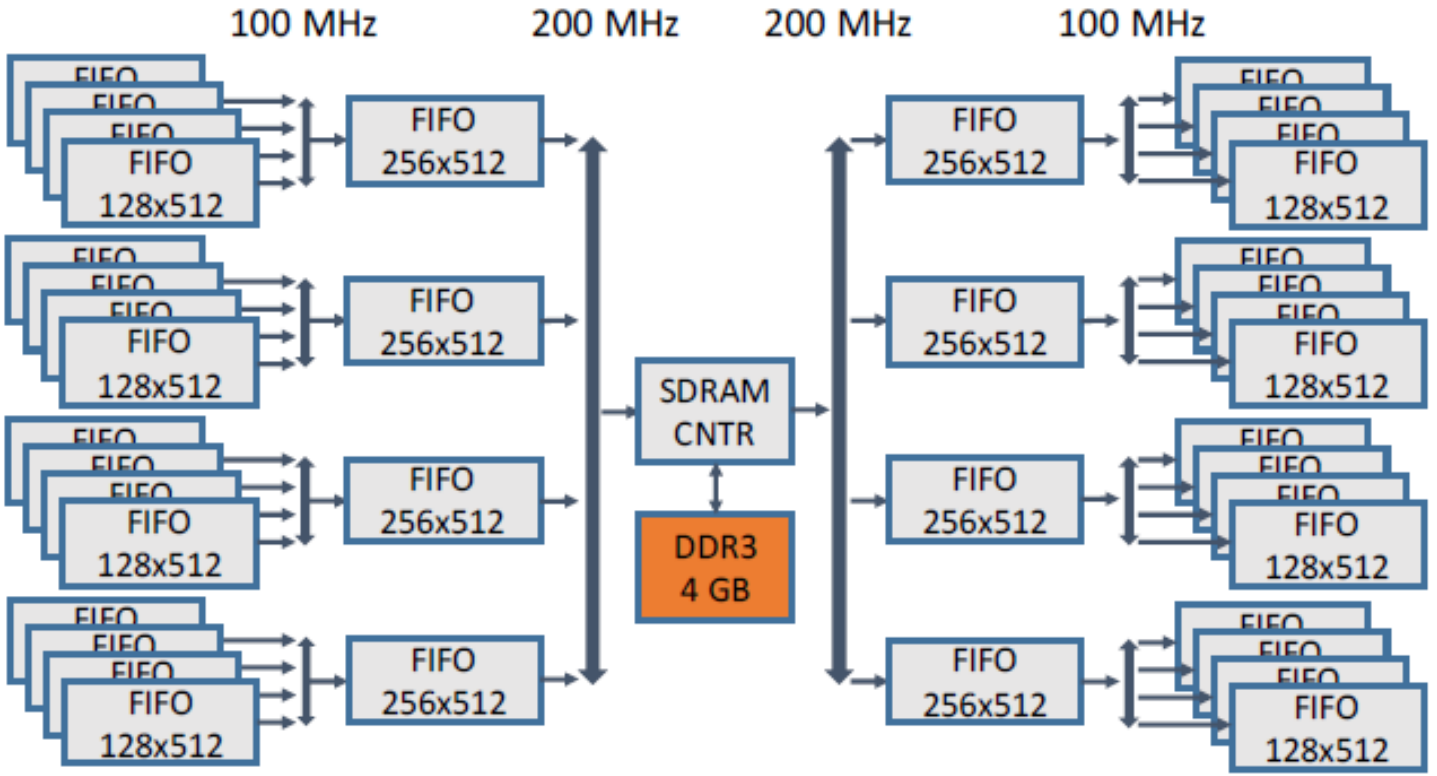
VME carrier card	
○ **form factor:**	6 U VME
○ **interfaces:**	• TCS (Trigger Control System) receiver • 1 Gb Ethernet for control network (IPbus) • 16 serial data links (SLINK) • JTAG for backup programming of FLASH

DHMultiplexer



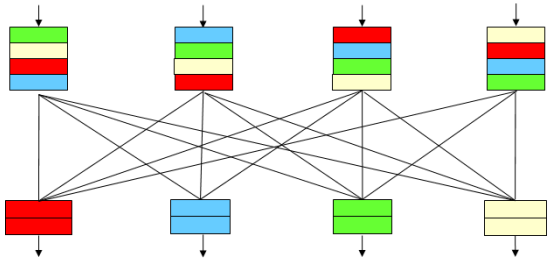


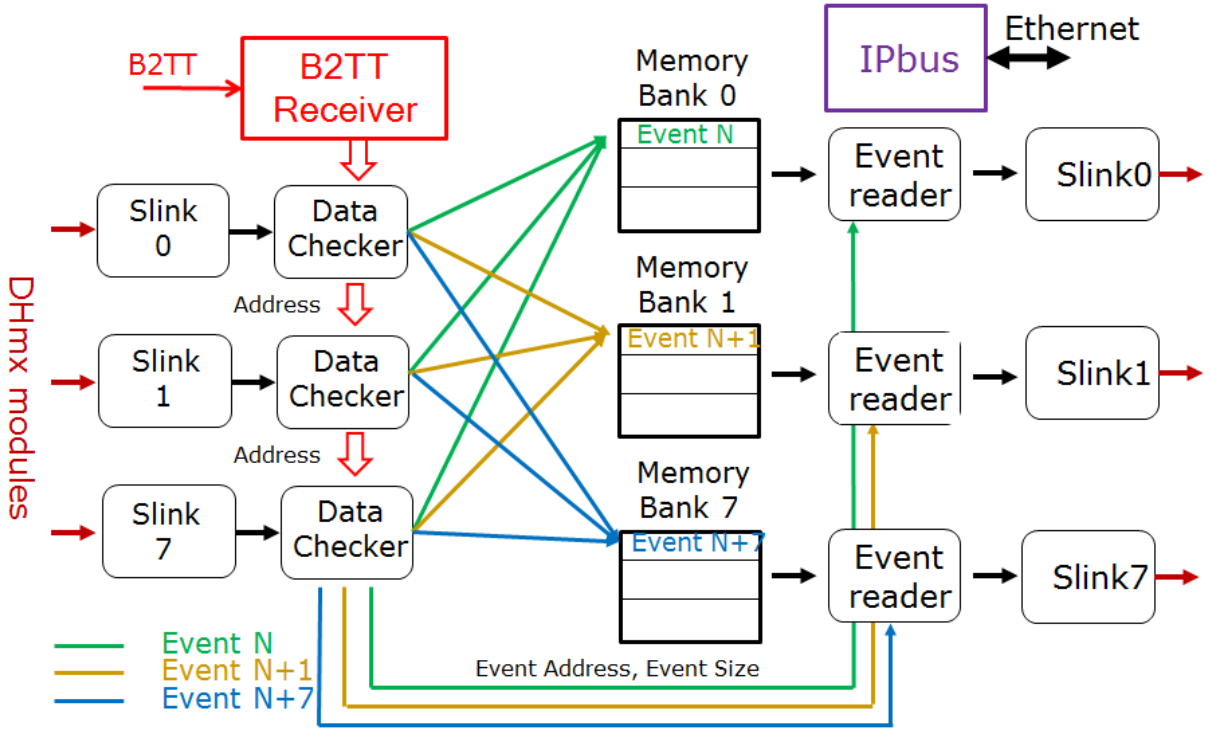
DDR3 memory controller



- 16 independent FIFO like memories blocks of 256 MB each
- 200 MB/s/port write performance
- 3 GB/s throughput

FPGA Switch 8x8(event builder)



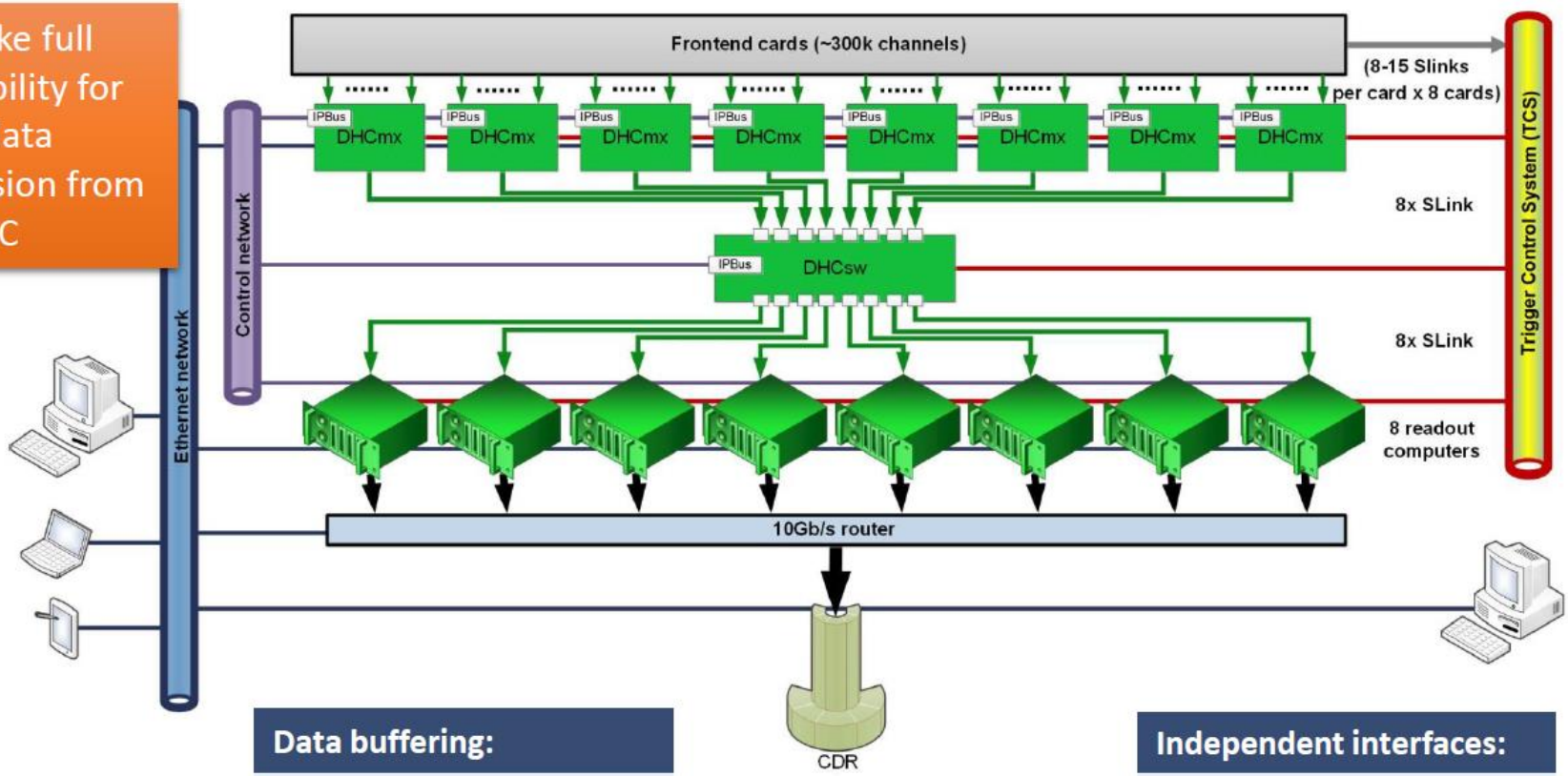


- Events are processed simultaneously and buffered in DDR, no congestion
- Events distributed between outgoing links in round robin manner
- 2.5GB/s (modern FPGA 10 GB/s) throughput

- One FPGA => Event builder for 2.5(10) GB/s throughput

iFDAQ Architecture

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



Data buffering:

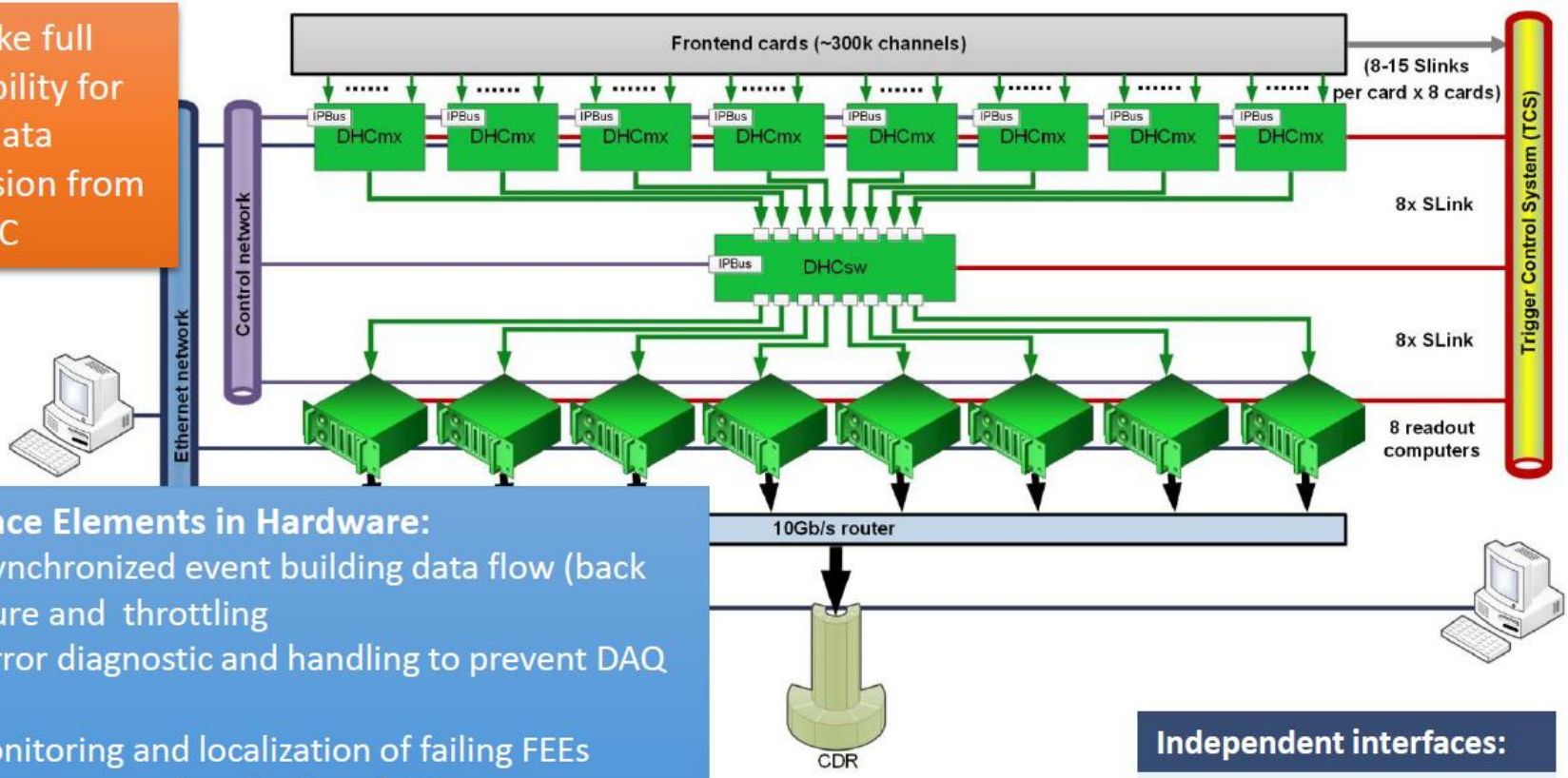
- 4GB RAM on each module => possibility to average data rate over spill cycle
- 1 GB/s sustained rate

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**

iFDAQ

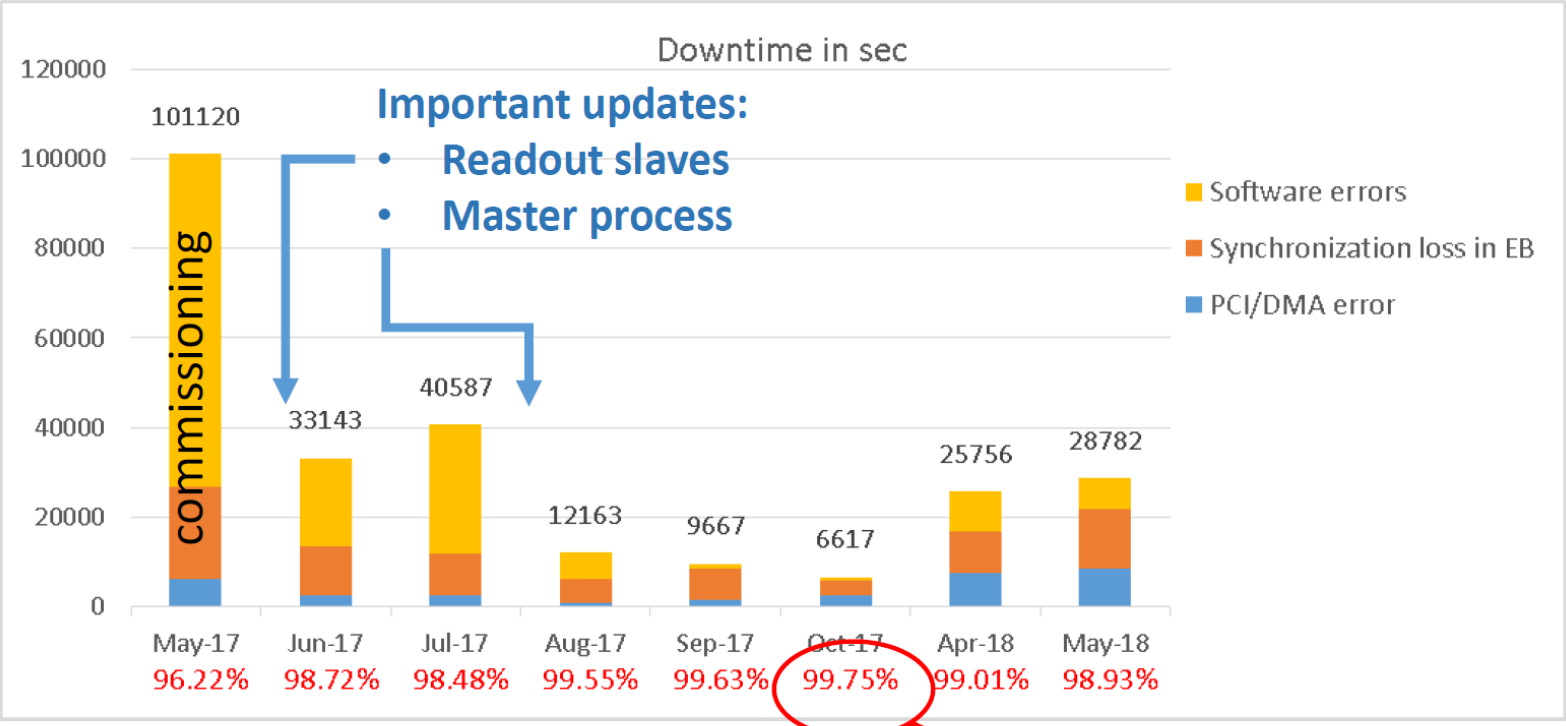
Compact : **Before** : 30 online PCs

Now : one VME 6U crate +

1 rack (8 computers)

Hardware Event Builder

Performance : Up Time in 2017



Effects:

1. Detectors more stable during run than during commissioning phase
2. Upgrades of RCCAR software

Highly reliable

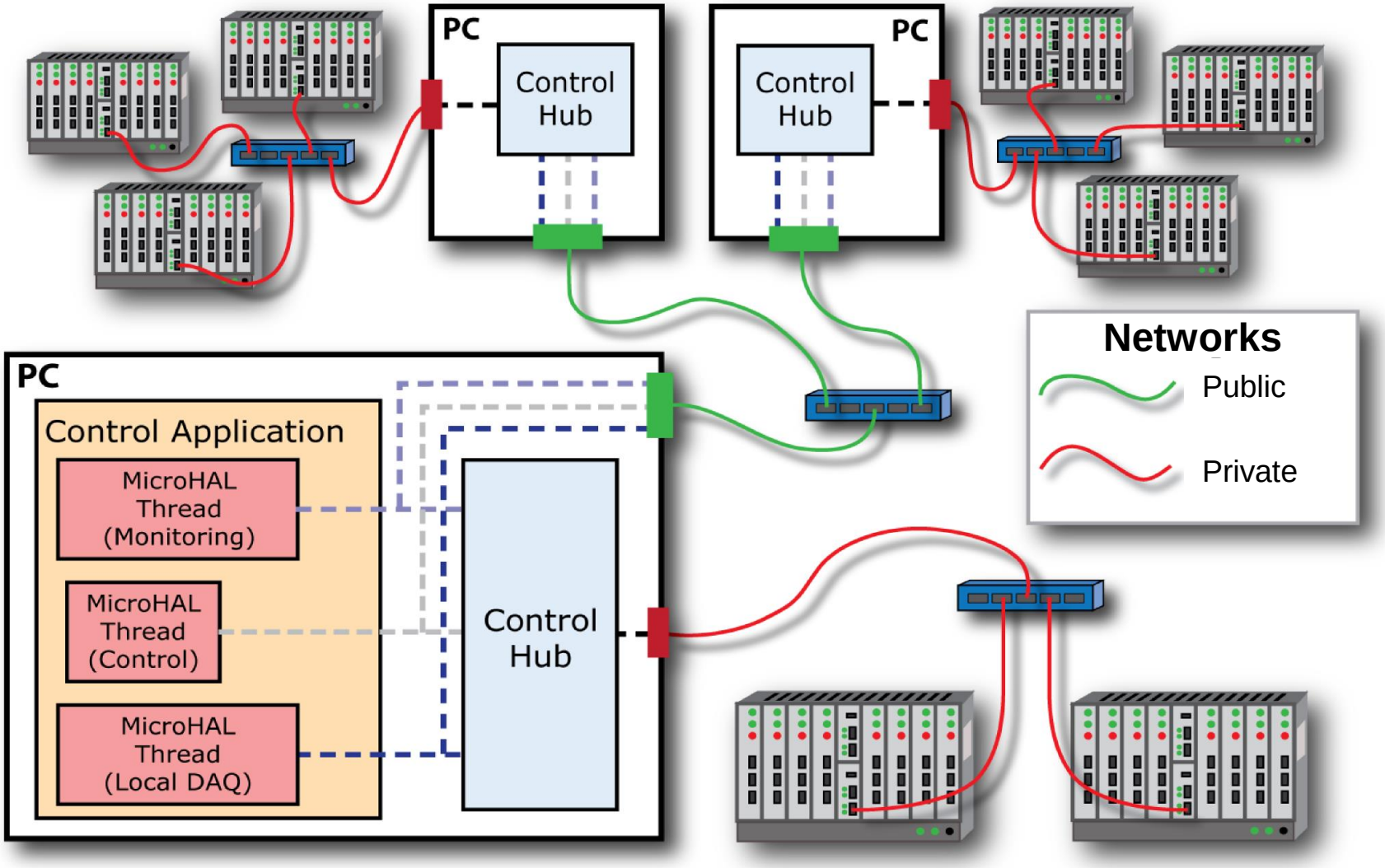
Protocol for slow control in FPGA

Ipbus

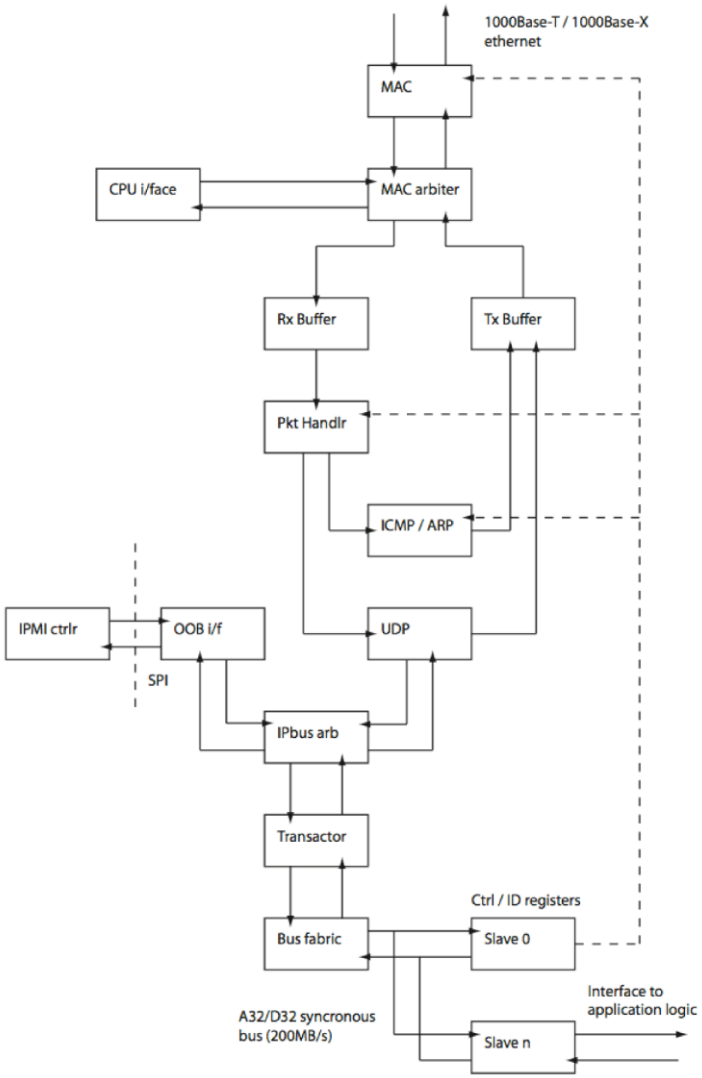
developed for CMS experiment

C.G. Larrea, K. Harder, D. Newbold, D. Sankey, A. Rose, A. Thea, T.Williams,
IPbus: a flexible Ethernet-based control system for xTCA hardware
Journal of Instrumentation 10, C02019 (2015)

IPbus Topology



IPbus FPGA Core and protocol



FPGA core architecture

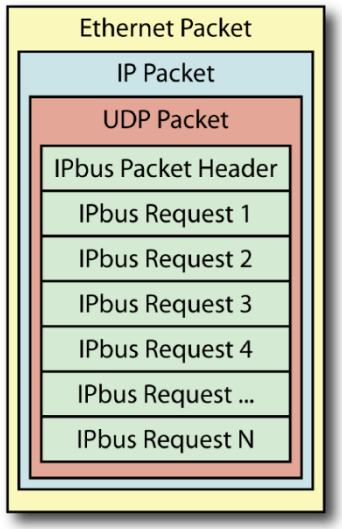
UDP based protocol

Mimics microprocessor interface

- Master runs in software on PC

3 types of transactions

- read
- write
- read-modify-write
  - for accessing bitfields



Ipbus frame

Interface to Slaves

ipbus_package.vhd

```
22 -- The signals going from master to slaves
23   type ipb_wbus is
24     record
25         ipb_addr       : std_logic_vector(31 downto 0);
26         ipb_wdata      : std_logic_vector(31 downto 0);
27         ipb_strobe     : std_logic;
28         ipb_write      : std_logic;
29     end record;
30
31 -- The signals going from slaves to master
32   type ipb_rbus is
33     record
34           ipb_rdata    : std_logic_vector(31 downto 0);
35           ipb_ack      : std_logic;
36           ipb_err      : std_logic;
37           ipb_timeout  : std_logic_vector (15 downto 0);
38     end record;
```

Synchronous microprocessor like interface

- Wishbone-compatible (open bus for systems-on-a-chip)

ipb_strobe signal indicates access to a slave (not address)

Available Slaves

Standard:

- Synchronous R/W register (ipbus_reg.vhd)
- Static register (ipbus_static.vhd)
- RAM (ipbus_ram.vhd)

Non standard:

- I2C (Opencores: i2c_master_top.vhd, …)
- Dynamic Reconfiguration Port for FPGA components (drp.vhd)
- Asynchronous R/W register (ipbus_asynchreg.vhd)
- Block RAM interface (ipbus_bram.vhd)
- FIFO, depth 512, 4096, 65536 words (ipbus_rwfifo.vhd)
- Parallel NOR Flash programmer

Software

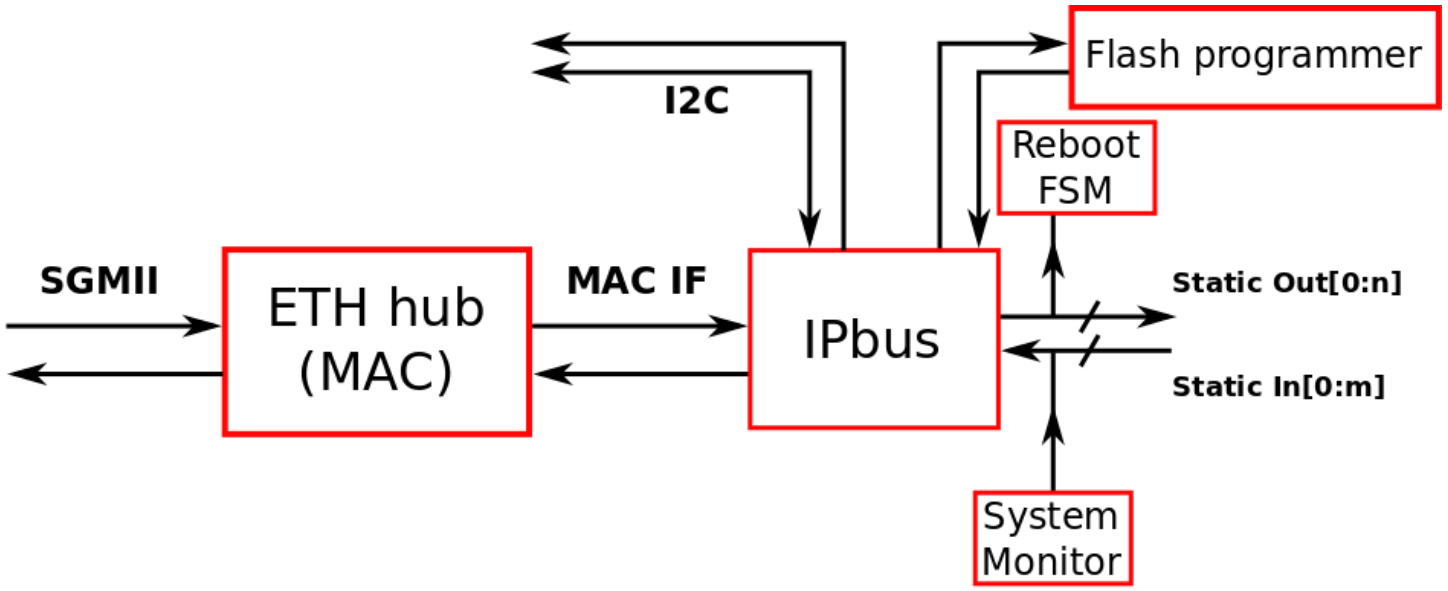
Software available as RPMs for SL6/7 and as source code

- https://svnweb.cern.ch/trac/cactus/wiki/uhalQuickTutorial
- µHAL: c++ framework
- Pycohal: python bindings
- Controlhub: software multiplexer

XML configuration files

- connection.xml: defines connection type (UDP/TCP) and addresses of the hardware
- address.xml: register map

Firmware Layout

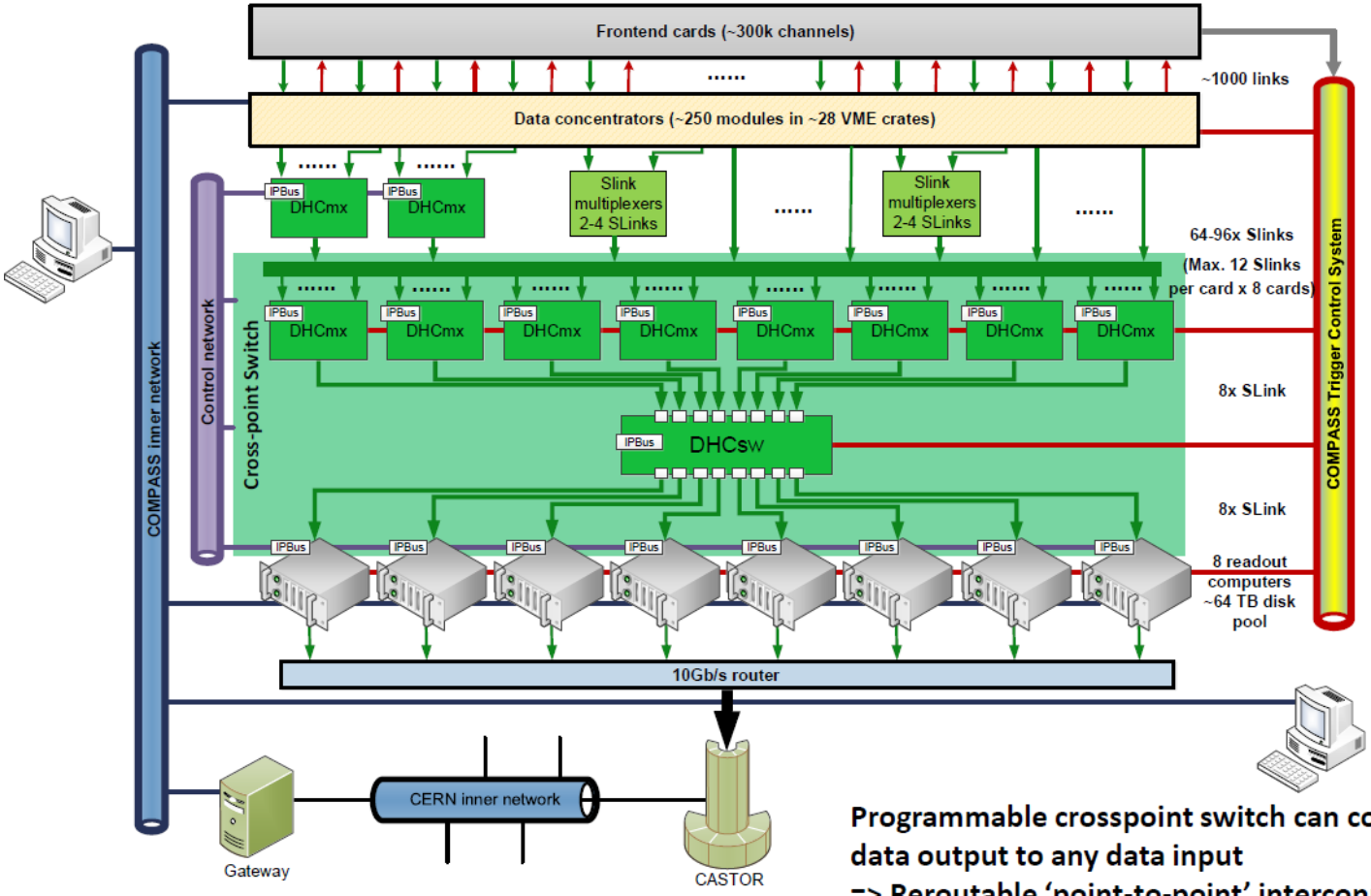


Basic firmware for COMPASS DAQ FPGA boards (Virtex-6)

- monitoring
- rebooting
- flash programming

New Developments

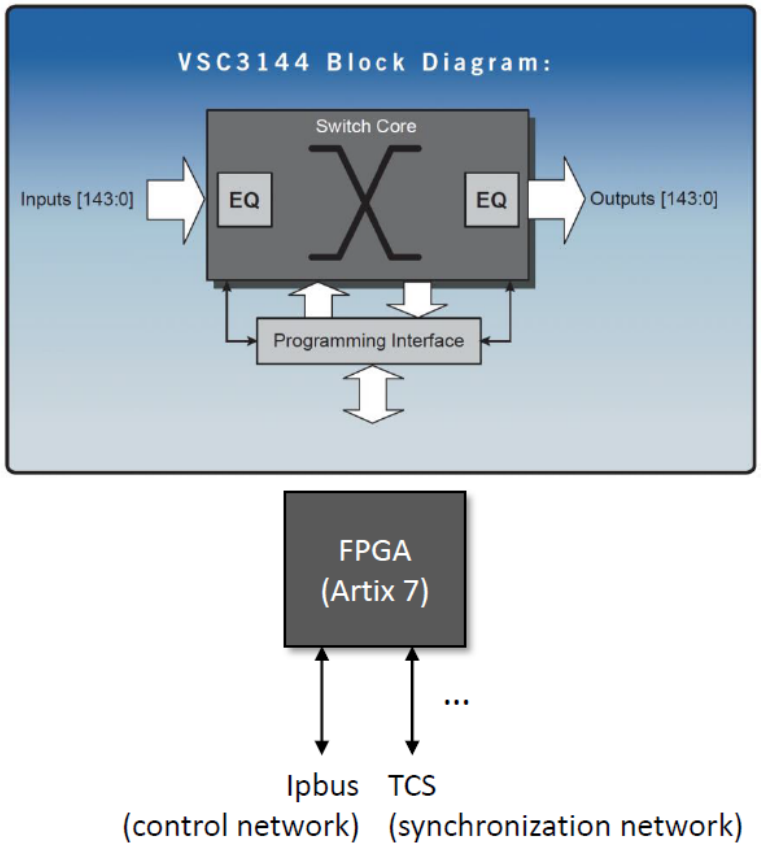
Cross-Point Switch



Programmable crosspoint switch can connect each data output to any data input
=> Reroutable 'point-to-point' interconnections between nodes of EB

Hardware : Vitesse VSC3144

- 144 x 144 strictly non-blocking cross-point switch
- Up to 6.5 Gbps bandwidth per port
- No registers used in data path i.e. asynchronous data path => no restrictions on the phase, frequency, or signal pattern of any input (protocol independent)
- 45mm x 45mm 1072-pin BGA package
- Core programming on port-by-port basis OR simultaneous issuing of multiple queued assignments (low latency: ns)



Crosspoint Switch – Hardware Design

Crosspoint Switch Components

- **interfaces:**
  - 12 x 12 channel CXP transceiver (MPO fiber connectors)
  - Ethernet for IPbus
  - JTAG
  - TCS (Trigger Control System) receiver
- **Switching and Control:**
  - **Vitesse VSC3144-02** – fully configurable 144x144, asynchronous, 6.5 Gbps crosspoint switch
  - **Xilinx Artix-7 FPGA** for switch control and monitoring

- **Interface FPGA – Crosswitch:**
  - 90 MHz, 11-bit parallel data bus
  - Multiple program assignments can be queued and issued simultaneously ⇒ fast programming (<< 1us)

Development System Redundancy

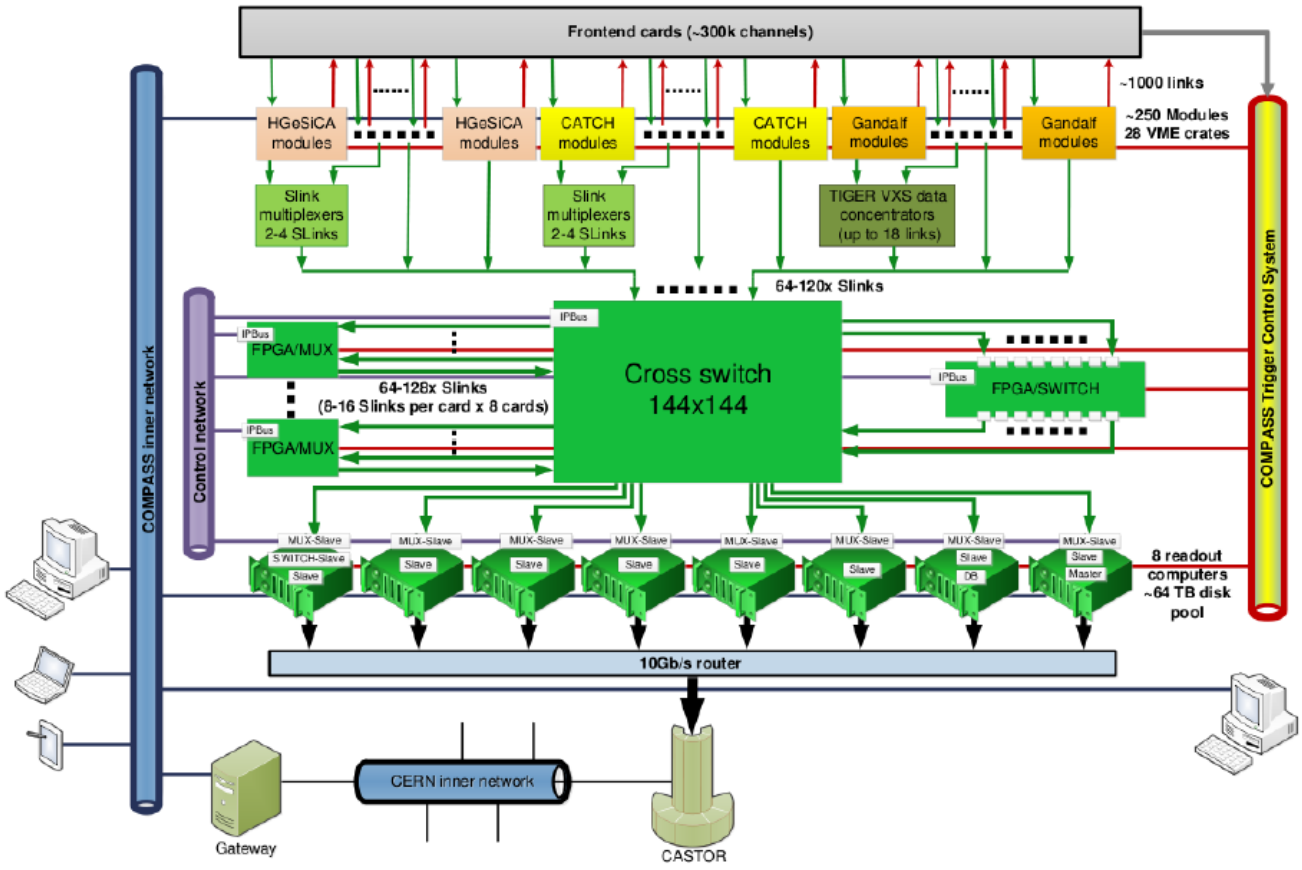
- Automatic identification of malfunctioning hardware parts
- Reconfiguration of crosspoint switch to substitute faulty module by spare one. Executed by software
- Upgrade if Time Distribution System (TCS) to bidirectional PON (passive optical network) with use of Universal Communication Framework developed at TUM for on-the-fly reconfiguration of interconnections
- Minimizing of real-time software processes
  => Direct writing of data onto SSD

Crosspoint Switch

Crosspoint Switch

- connects:
  - FE electronics
  - DHCmx modules
  - DHCsw module
  - Spillbuffers
- purpose:
  - Ease of load balancing
  - System redundancy to compensate hardware failures

⇒ provides fully customizable network topology



New FPGA Module

- Kintex XCKU095T FPGA
- Custom case design

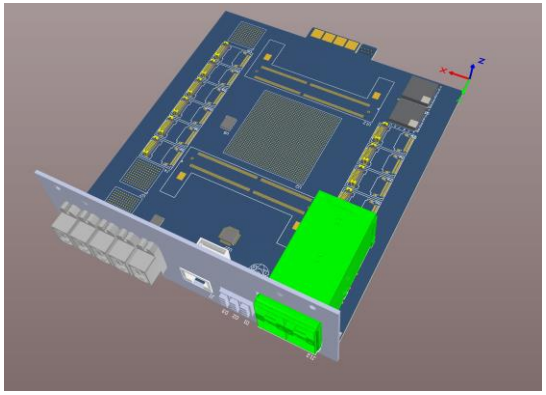
Interfaces

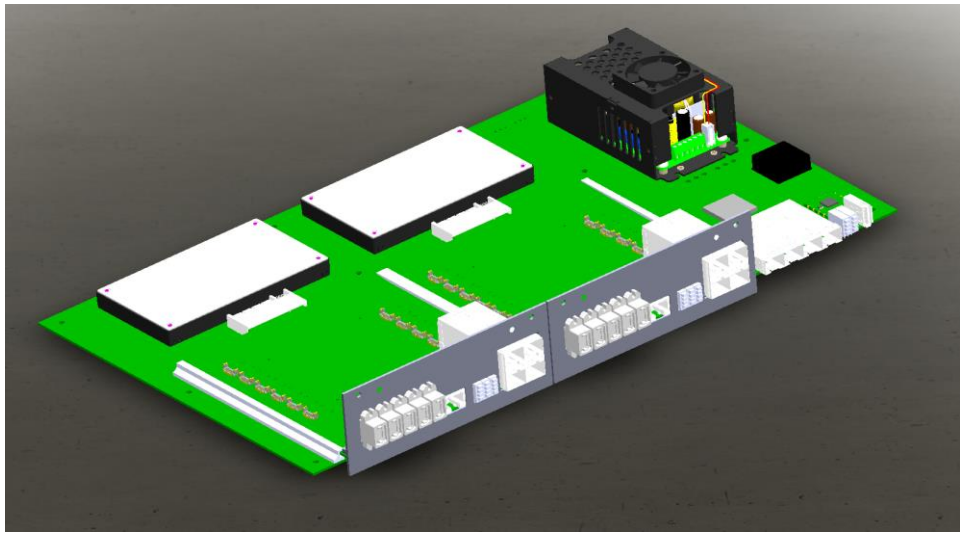
- 60x12Gb/s links
- IPBUS
- TCS

Memory

- 2x16GB DDR4

Module will be available in October 2018





Unified Communication Protocol

Developed by Dominic Gaisbauer

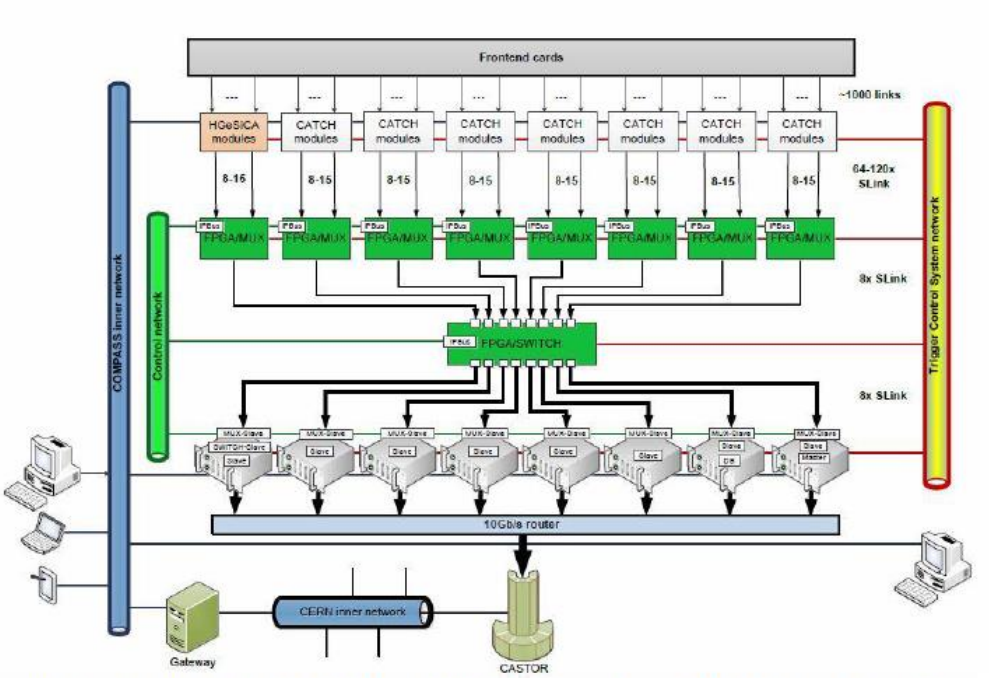
PhD student of TUM

The work received first student award at RT2016 conference

COMPASS Experiment Topology

- One link for TCS messages
- One link for IPBus messages
- Multiple data links
- Underlying principle the same in many other experiments

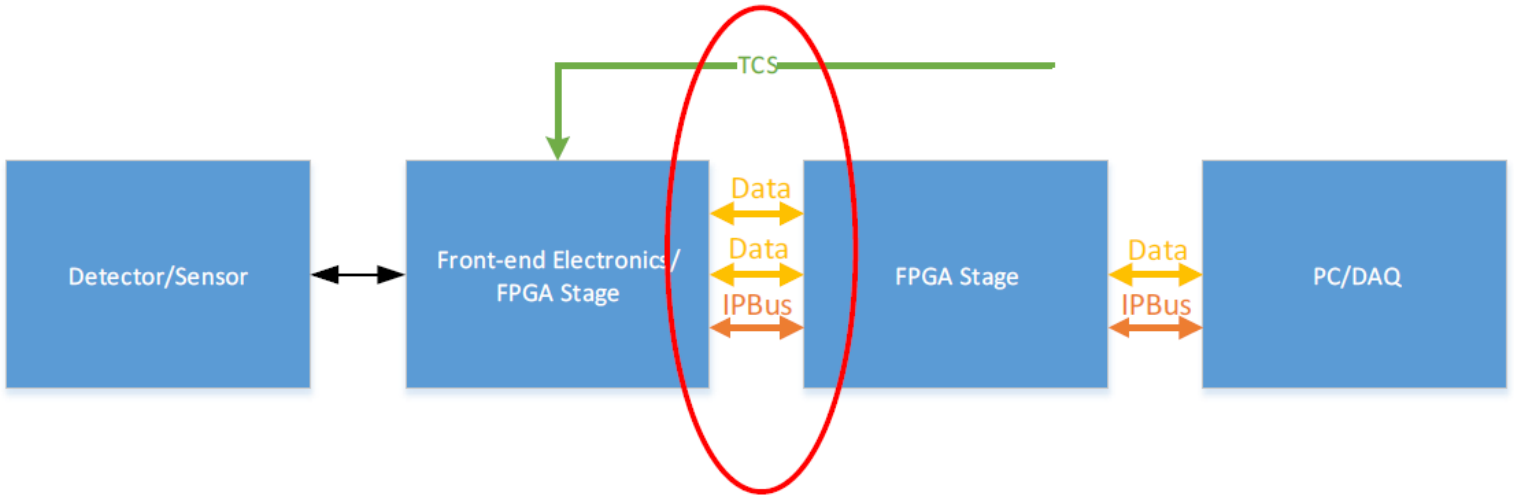
- Why not combing data, TCS, and data in a single link?



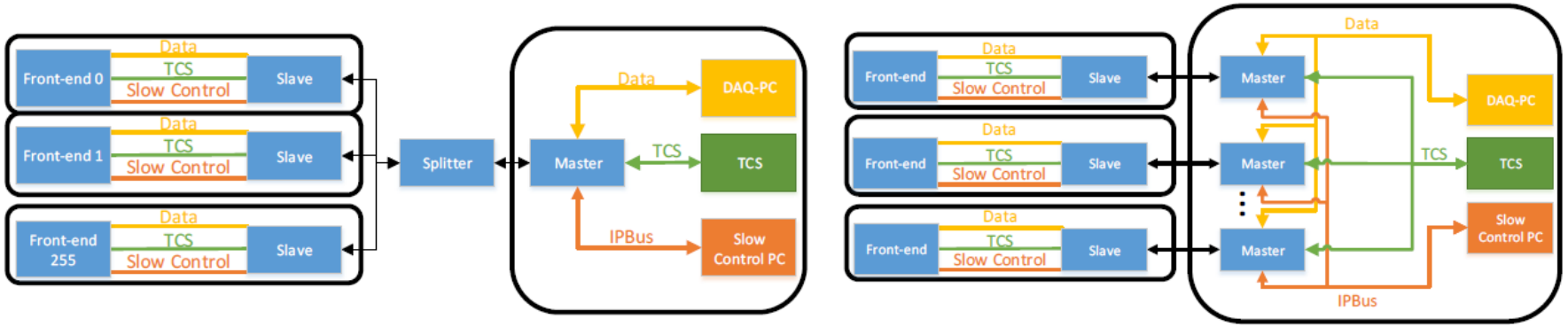
J. Novy et al., FPGA based data acquisition system for COMPASS experiment

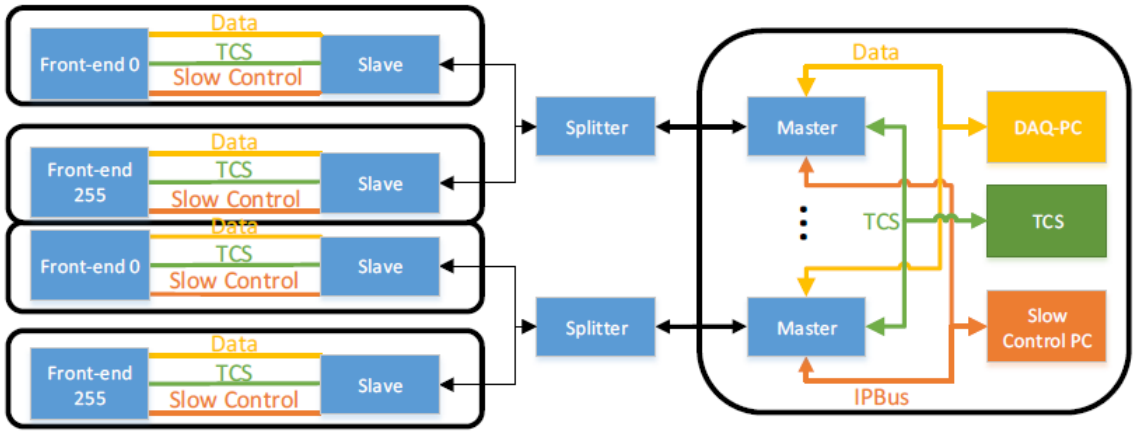
Unified Communication Framework (UCF)

- Originates from the SODA time distribution system
- Single high-speed serial link for data, slow control, trigger, and timing information
- Up to 64 different communication channels
- Unidirectional determined latency for a one channel
- Fully configurable ipcore via single VHDL package file
- Link speed depends only on FPGA resources



UCF – Example Topologies





UCF - Low Layer Protocol

- Backbone of UCF
- Handles communication and initialization
- 8b/10b encoding scheme
- Internal data width of 16b/20b (32b/40b) for the protocol
- 10b K-characters for control and synchronization
- Fixed phase synchronization by sequence of two defines K-characters (x”BCDC”)
- Unique IDs and IPs by FPGA DNA

UCF – Priority Handling

UCF – User Interface and Configuration

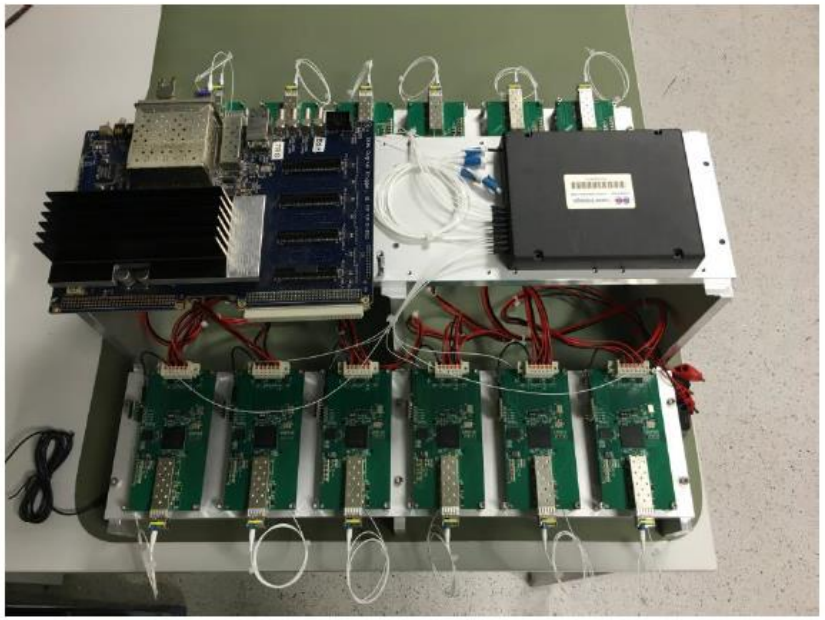
- All channels can be addressed via the standardized ARM AMBA AXI interface
- Leads to easy integration and connection to other ipcores
- Easy configuration of all important parameters within one file:
  - Link speed
  - Topology
  - Data width
  - Device type
  - …

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
--! ConfigUCF is the configuration file that has to be modified by the user according
--! to his requirements
package ConfigUCF_ucf is
constant strDeviceType          : string                                    := "Virtex6";
constant stdActivateP2P         : std_logic                                 := '1';
constant stdCardPurpose         : std_logic                                 := '1';
constant intTransceivers        : integer                                   := 1;
constant intInterfaces          : integer                                   := 1;
constant intCylcesMin           : integer                                   := 8;
--!Data width settings for User
constant intDataWidthUsr        : integer                                   := 32;
constant intByteWidthUsr        : integer                                   := 4;
constant intSerials             : integer                                   := 1;
type     dna_type                 is array (integer range <>) of std_logic_vector(63 downto 0);
constant DNAs :                   dna_type(intSerials downto 0) :=((  x"0000000000000000"),
                                                                    (x"00C0426EBA540890"));
```

UCF – Tests and Measurements

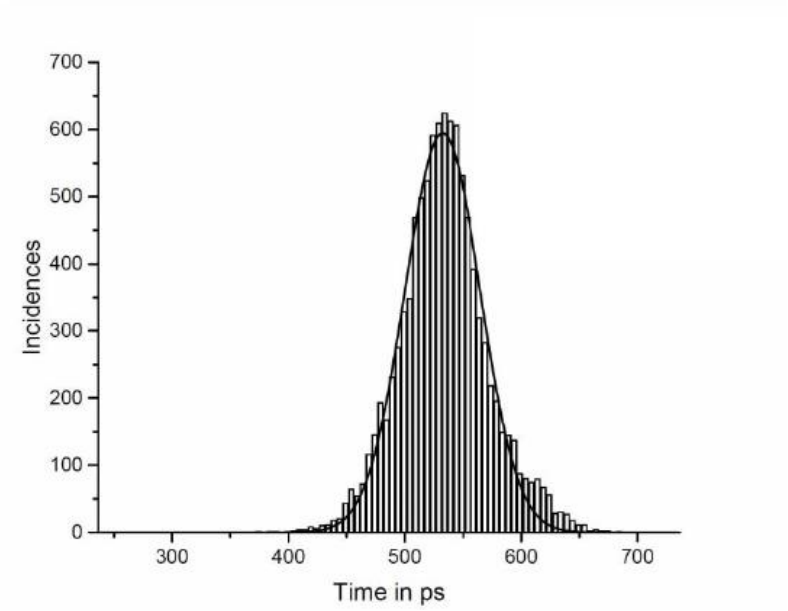
- Star-like topology with 12 slaves and 1 master
- 1.25 Gbit/s link speed
- Spartan 6 FPGA as slave and Virtex 6 as master
- Switching time of 16 µs
- Long term stability test with 99 % link utilization for two weeks
- JTAG chain possible with up to 100 kHz frequency
- IPBus over UCF implemented

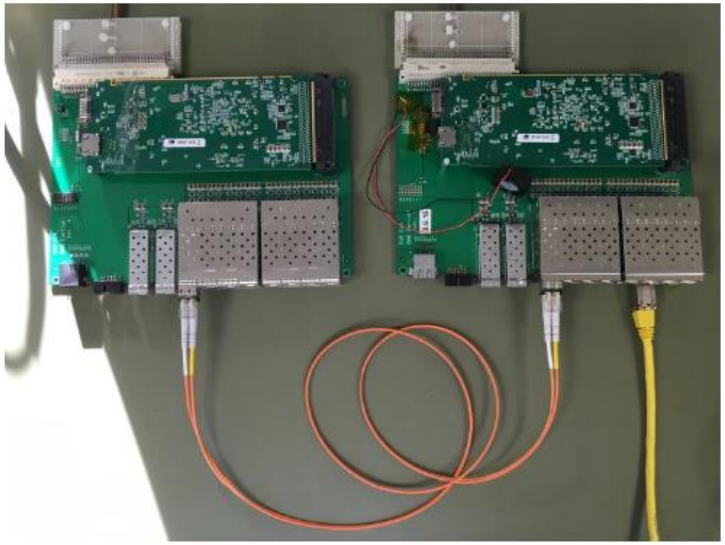
Transmission Time [µs]	Efficiency [%]
25000	99,93
10000	99,84
1000	98,42
500	96,90
100	86,20



UCF – Tests and Measurements

- Point-to-Point topology with 1 slave and 1 master
- 2.5 Gbit/s link speed
- Virtex 6 as slave and master
- Recovered clock jitter of 23 ps
- Long term stability test with 99 % link utilization for two weeks
- IPBus over UCF implemented





UCF - Conclusion

- Developed ipcore providing unified communication of up to 64 channels via a single optical link
- One channel has deterministic latency in one direction from master to slave
- Standard ARM AMBA AXI interface for user
- 1:N or multiple 1:1 connection possible
- 98-99 % link utilization efficiency depending on the architecture
- 16 µs switching time in star-like topology
- 23 ps jitter of the recovered clock
- Implemented JTAG over UCF with 100 kHz frequency
- Implemented IPBus over UCF
- Employed in Belle2 and PENeLOPE experiments

THANK YOU