Istituto Nazionale di Fisica Nucleare

European Organization for Nuclear Research

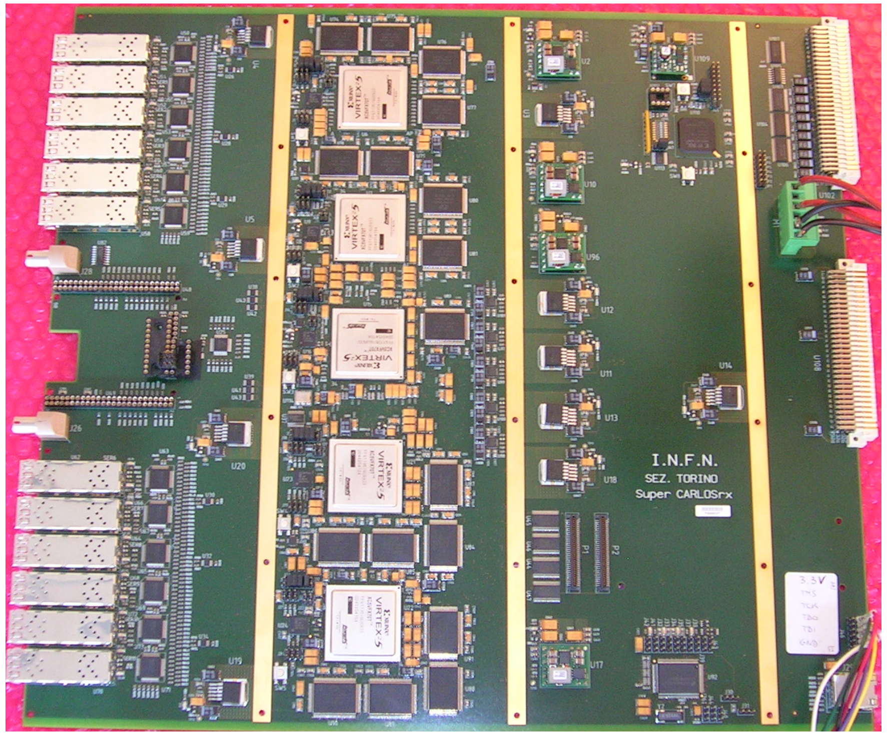
Readout boards for ALICE and ATLAS experiments

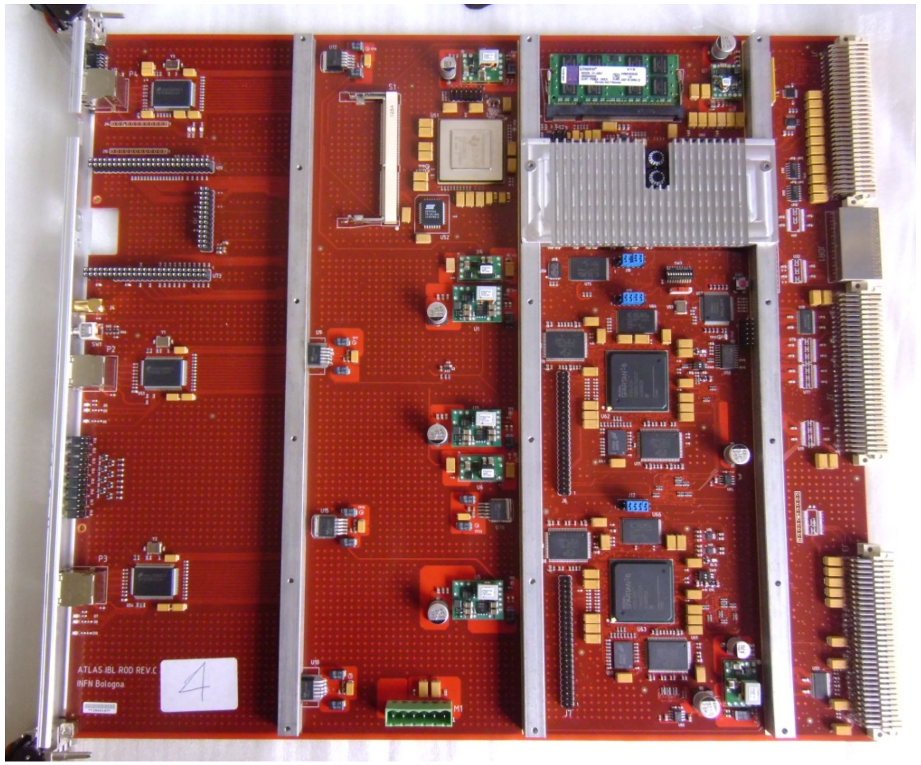
Davide Falchieri

Torino Workshop: 28 November 2013

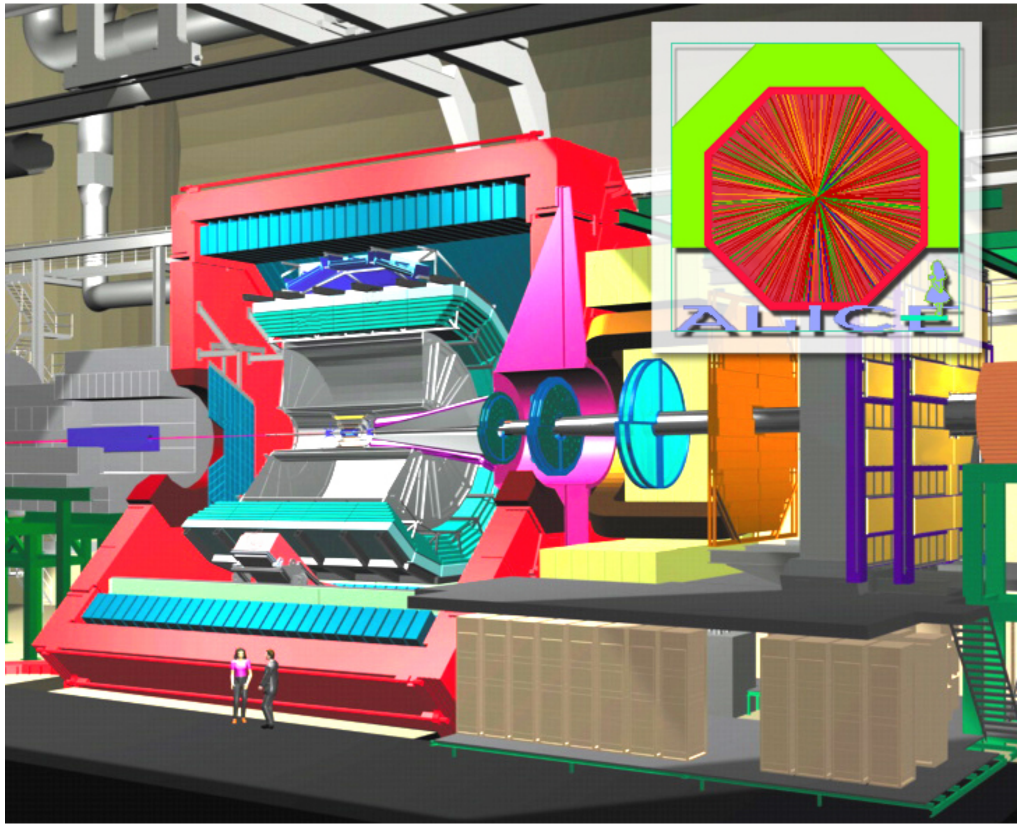
Outline

- Case study 1: ALICE SDD readout board
- Case study 2: ATLAS IBL readout board





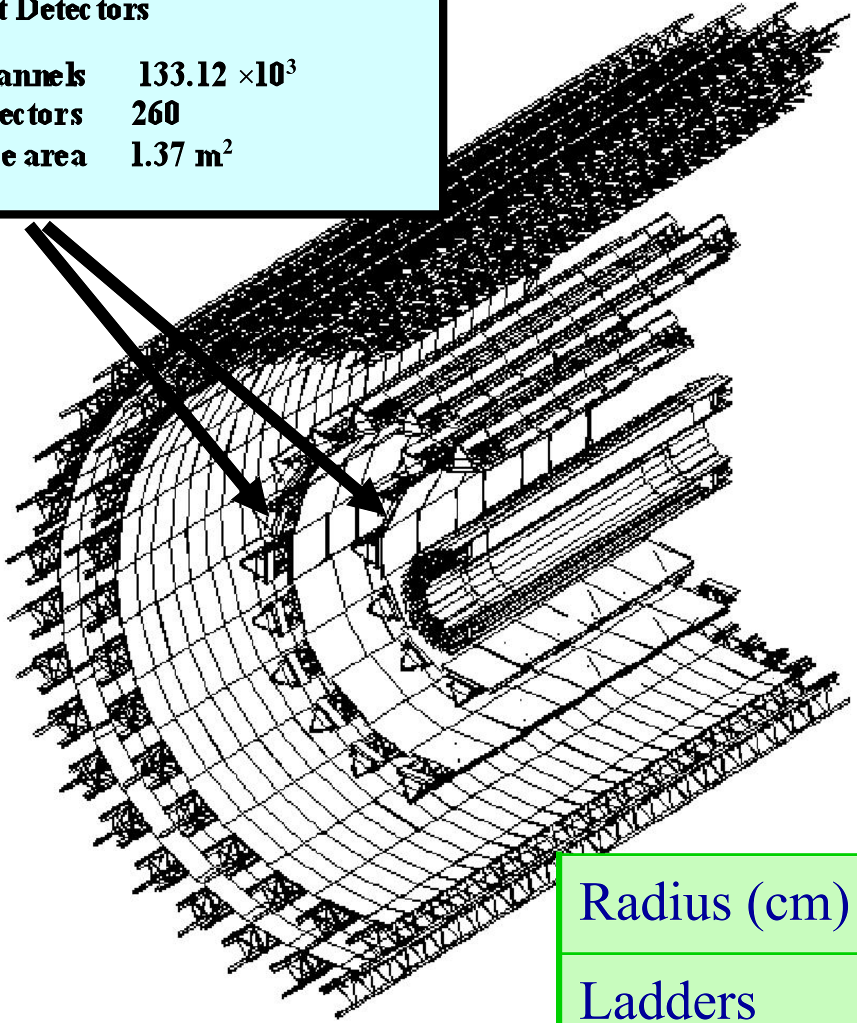
Case study (1):
readout board for ALICE SDD

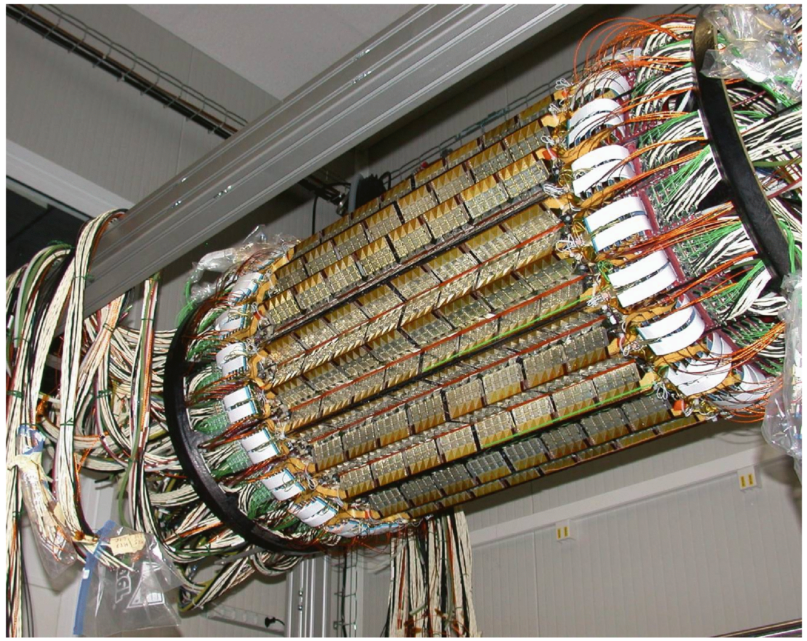


SDDs equip 2 out of 6 cylindrical layers of the ALICE ITS

Silicon Drift Detectors

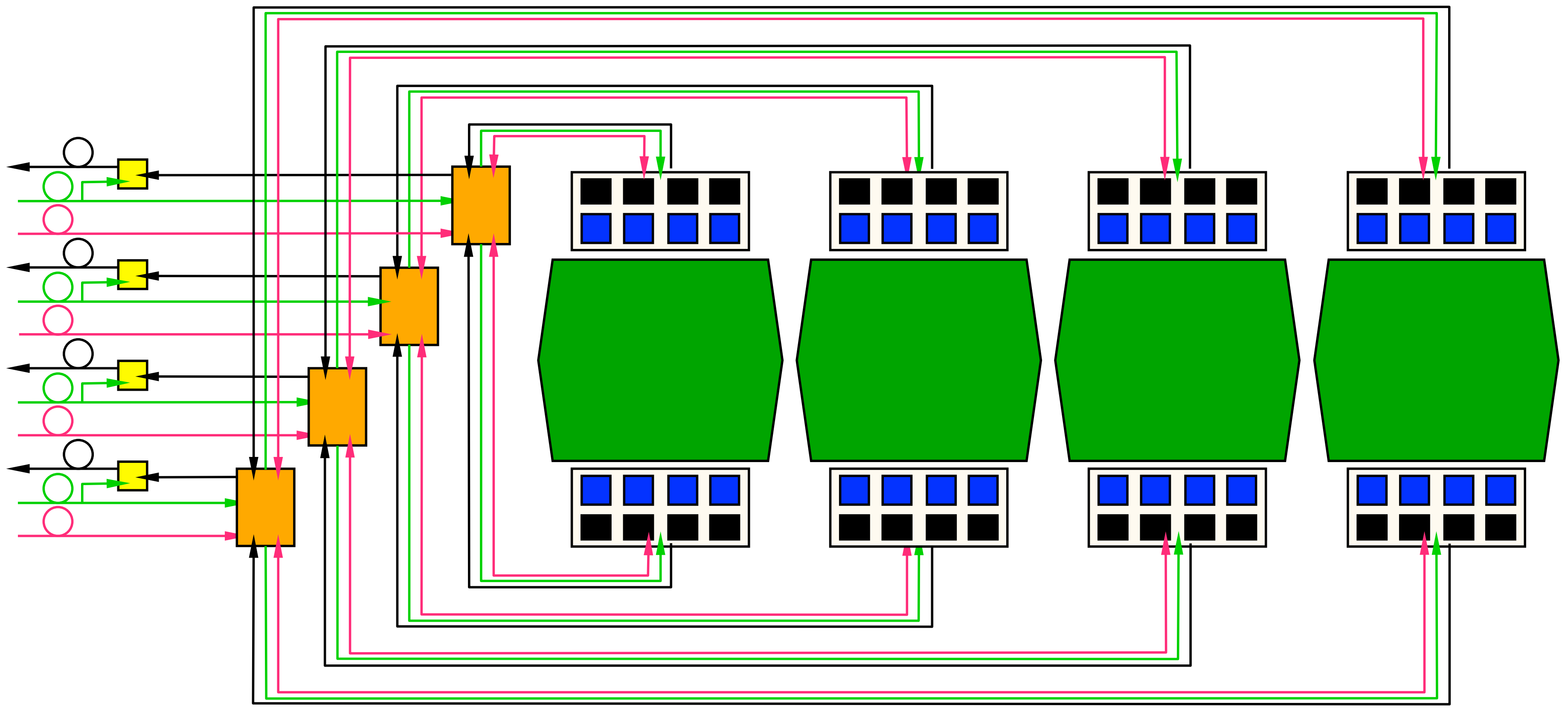
Tot. No. Channels	133.12×10^3
Tot. No. detectors	260
Tot. sensitive area	1.37 m^2





	Layer 3	Layer 4
Radius (cm)	14.9	23.8
Ladders	14	22
SDDs per ladder	6	8

SDD readout architecture (each half-ladder)

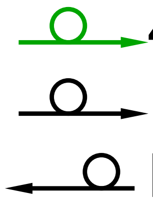


🟦 PASCAL

🟥 AMBRA

🟧 CARLOS (data compression)

🟨 GOL (Gigabit Optical Link) + QPLL


40 MHz clock

Programming & monitoring

Data output & monitoring

SDD front-end and readout electronics

Design specifications

- dynamic range: up to 8 MIPs
- noise: 250 e^-
- readout time < 1ms
- power consumption: <5 mW/channel
- chips thinned to 150µm

PASCAL (64 channels)

- Preamplifier ($\tau \sim 40$ns, RC-CR^2 shaping)
- Analog memory (64 ×256 cells)
- 32 10-bit linear ADC (1 every 2 channels)

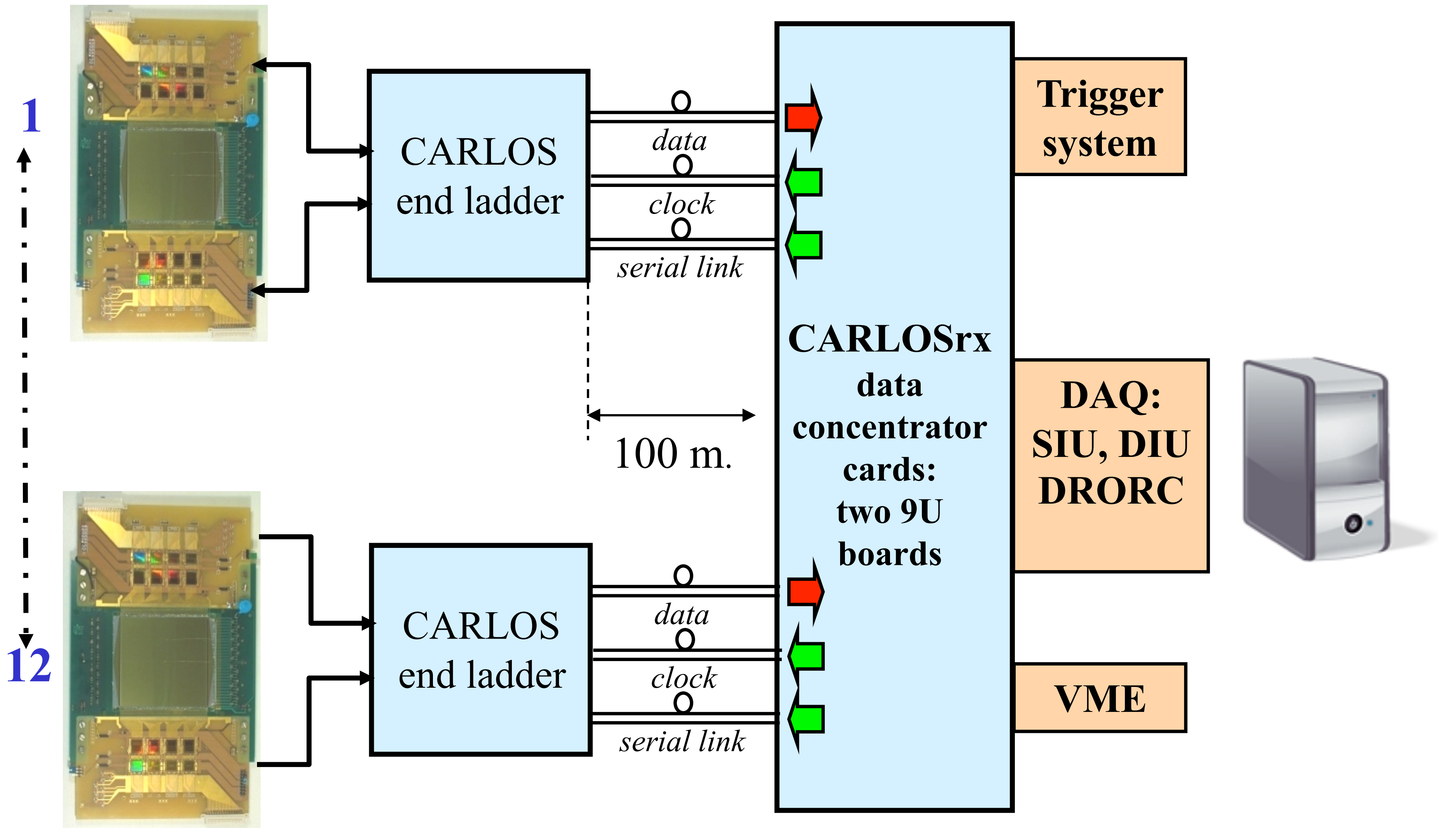
AMBRA (64 channels)

- Four 16 kB buffers
- Baseline equalization
- 10 to 8-bit compression

CARLOS (1 for 8 AMBRAs)

- Zero suppression and Compression of data from 1 SDD with a 2D – 2-Threshold algorithm (programmable parameters)
- Interface with AMBRAs, GOL and CARLOS-rx (FPGA based board, in counting room, which links to DDLs)
- FEE monitoring (SEU) time-multiplexed with data on the 16-bit output data bus
- Protections against radiation effects (parity check)

SDD readout chain



260 SDD detectors

24 CARLOSrx boards

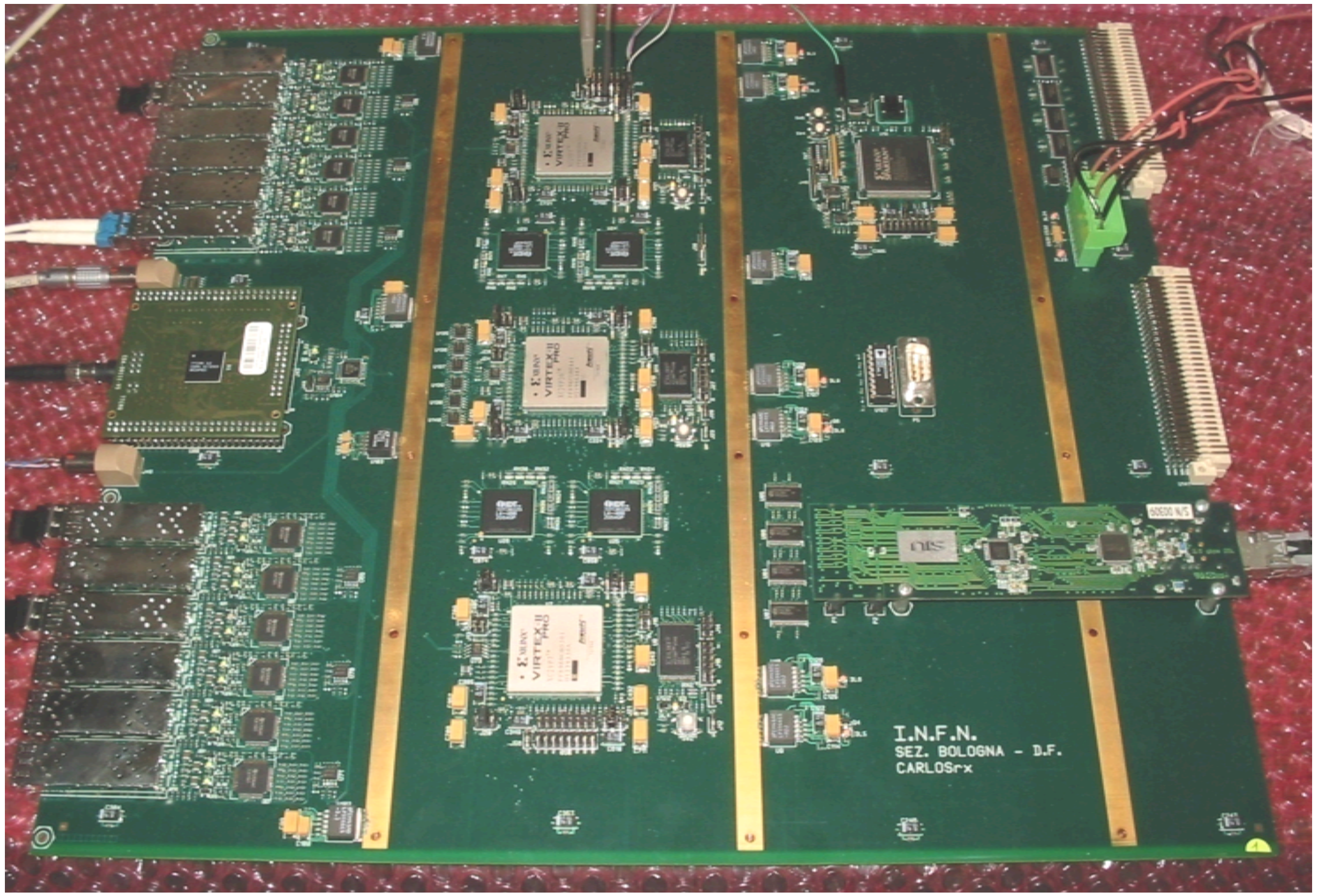
CARLOSrx readout board (2005)

6 SDD modules

busy

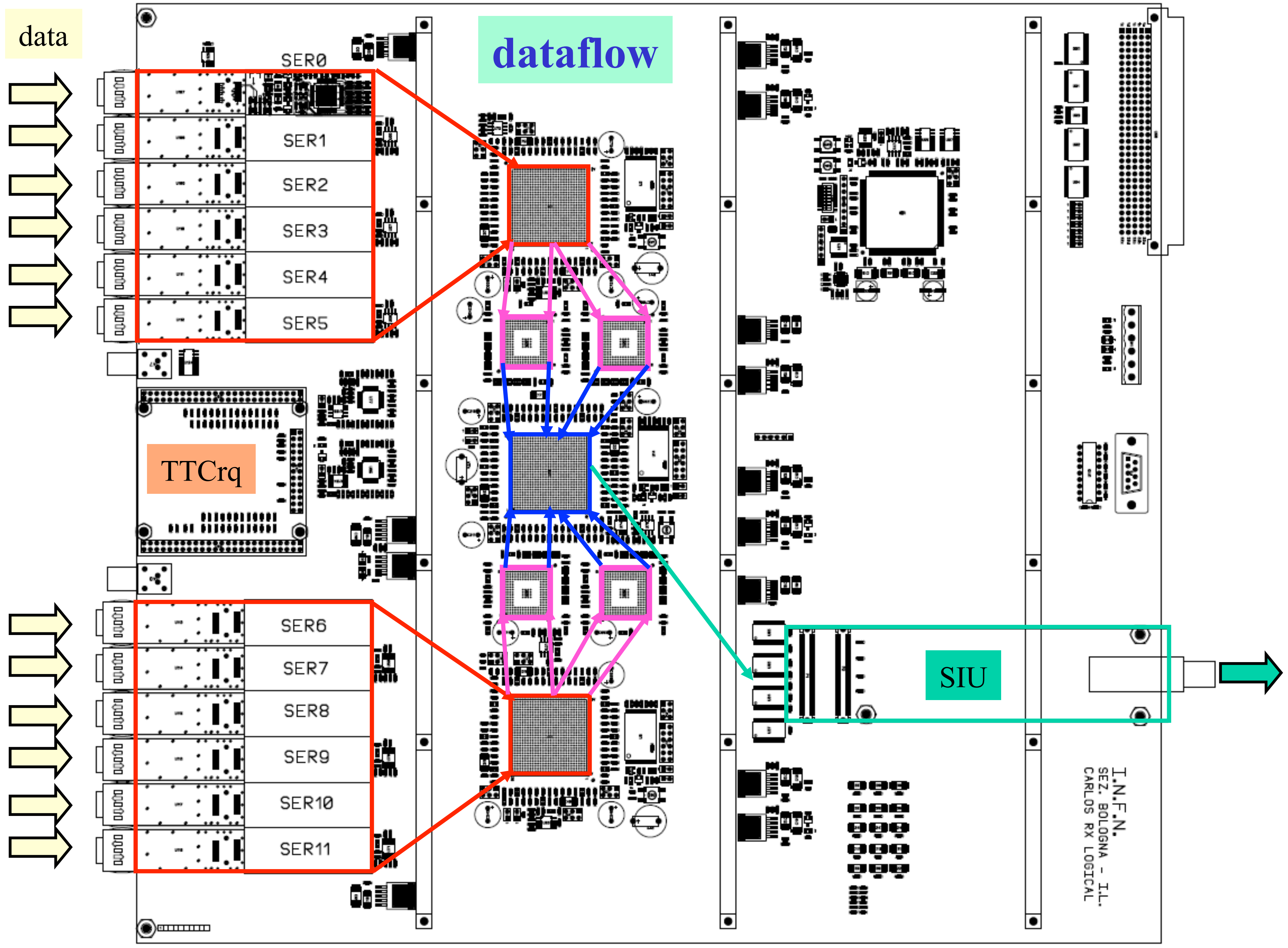
TTCrq

6 SDD modules

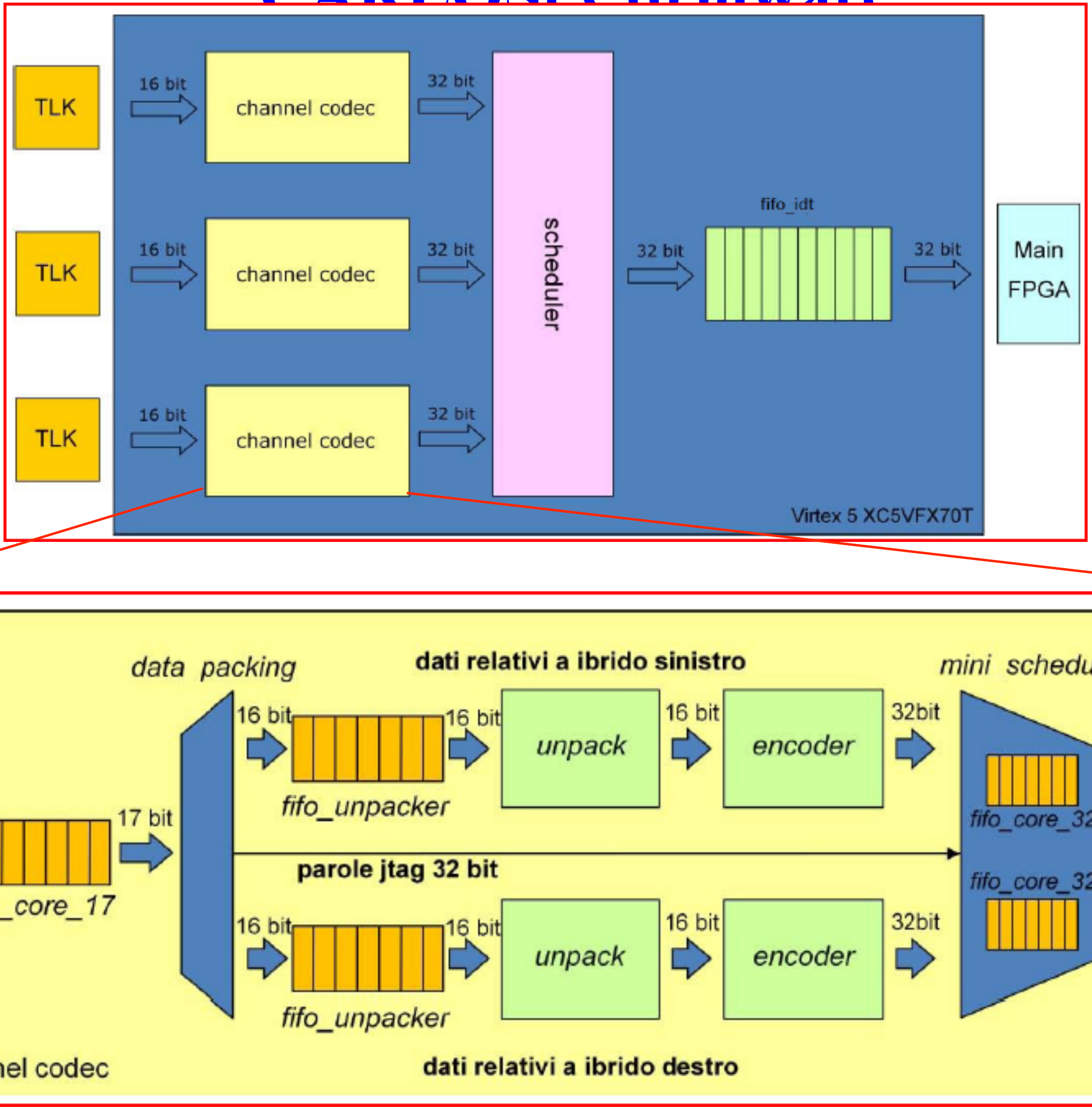


DDL

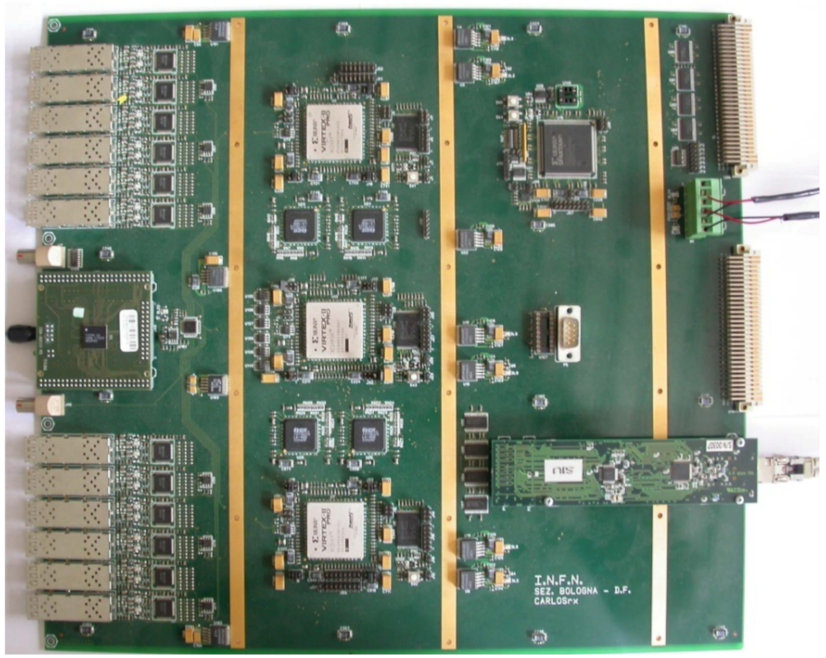
Input bandwidth: **12 x 800 Mbit/s**
Output bandwidth: **2 Gbit/s**
On-line data decoding, formatting and re-encoding with a different format

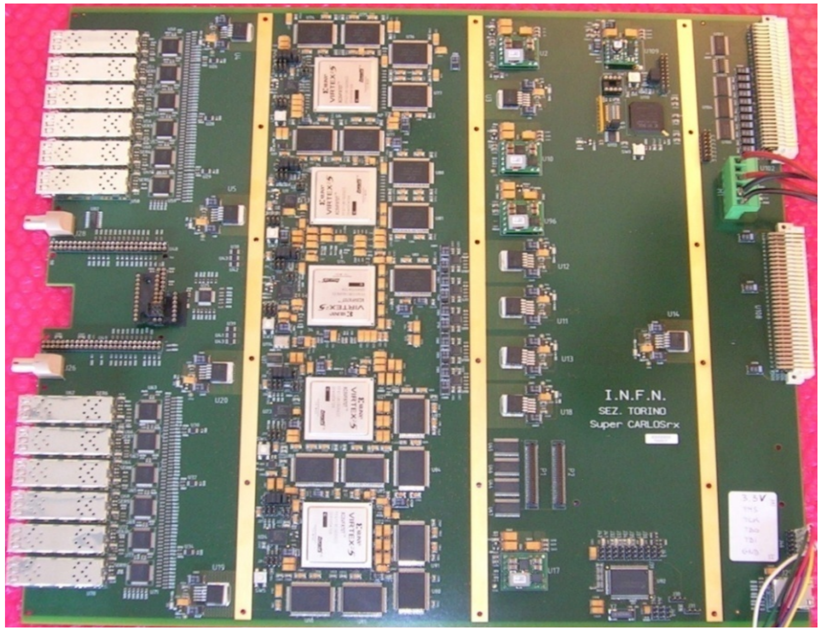

data
dataflow
SER0
SER1
SER2
SER3
SER4
SER5
TTCrq
SER6
SER7
SER8
SER9
SER10
SER11
SIU
I.N.F.N.
SEZ. BOLOGNA - I.L.
CARLOS RX LOGICAL

CARLOSrx firmware



From CARLOSrx to SuperCARLOSrx





Production: 2005-2006

• HW design: Davide Falchieri
• Installed in CR4 in 2007
• 3 Xilinx VirtexII-PRO FPGAs

Production: 2011

• HW design: Luca Toscano, Francesco Rotondo
• Installed in CR4 in 2013
• 5 Xilinx Virtex5 FPGAs

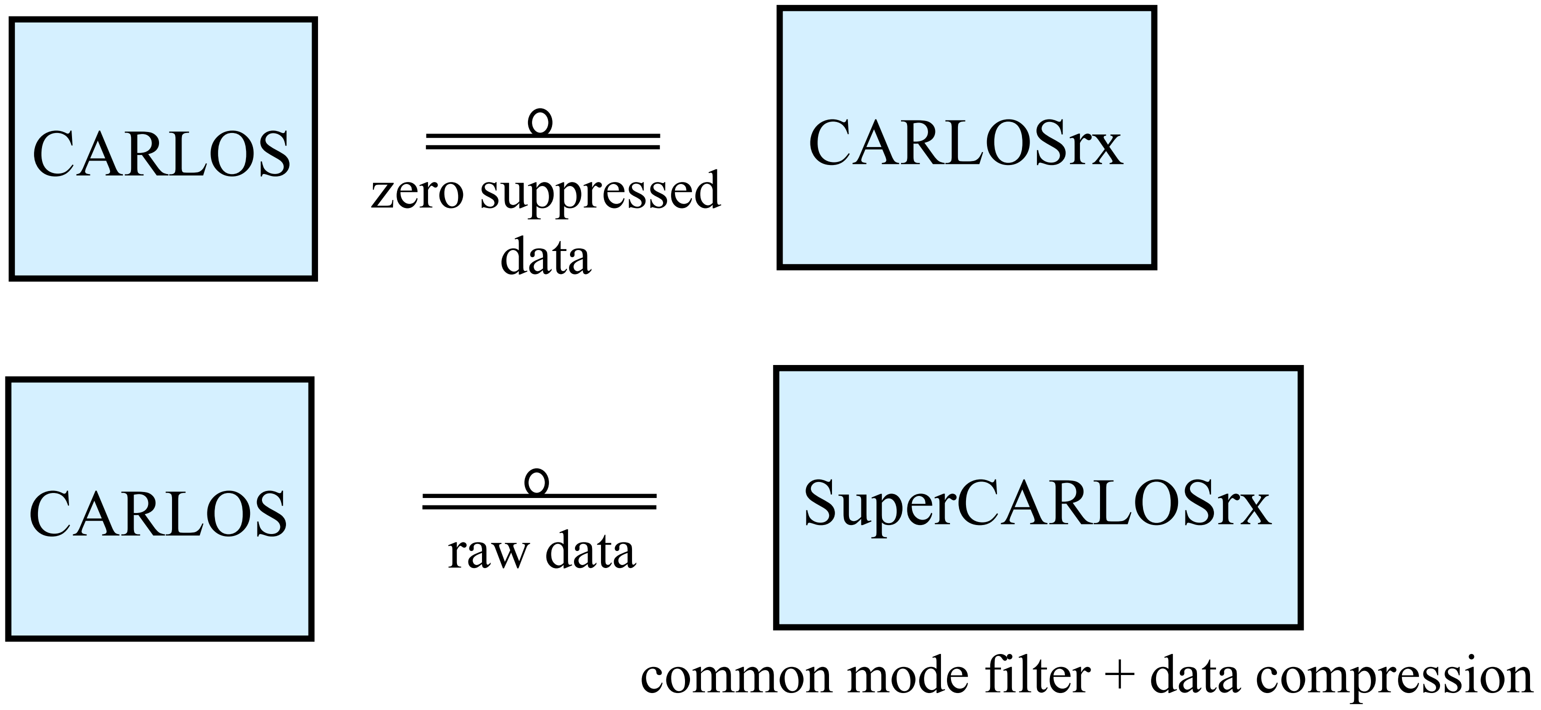
Why SuperCARLOSrx ?

Several reasons:

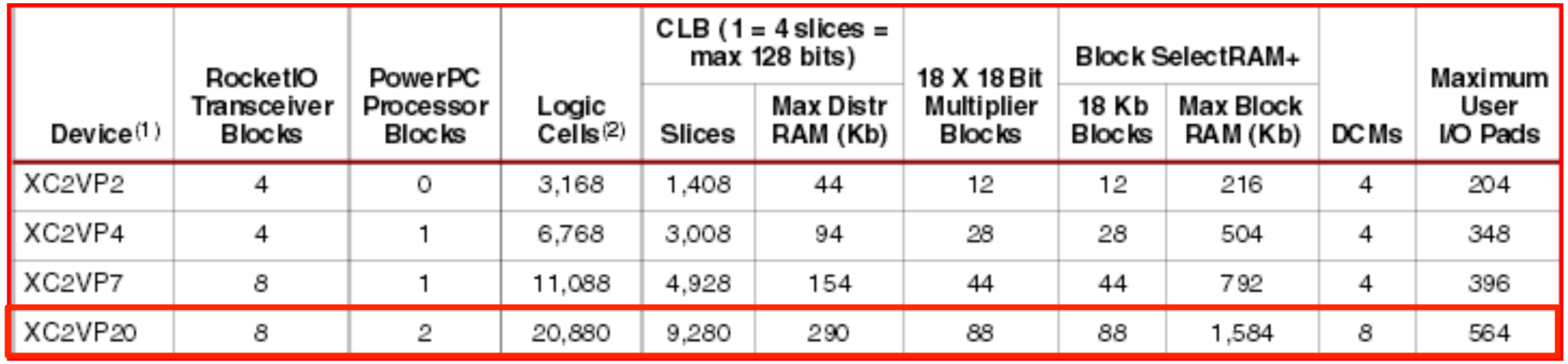
• Virtex-II PRO devices no longer supported by ISE after release 10 (Xilinx defines this family as a mature and discontinued product);

• Resources usage on the 3 Virtex-II PRO was around 60%: not much space for adding new firmware features

• A new algorithm (common mode subtraction) had to be inserted in the data path and simply CARLOSrx had no enough resources for the job



From Virtex-II PRO to Virtex5

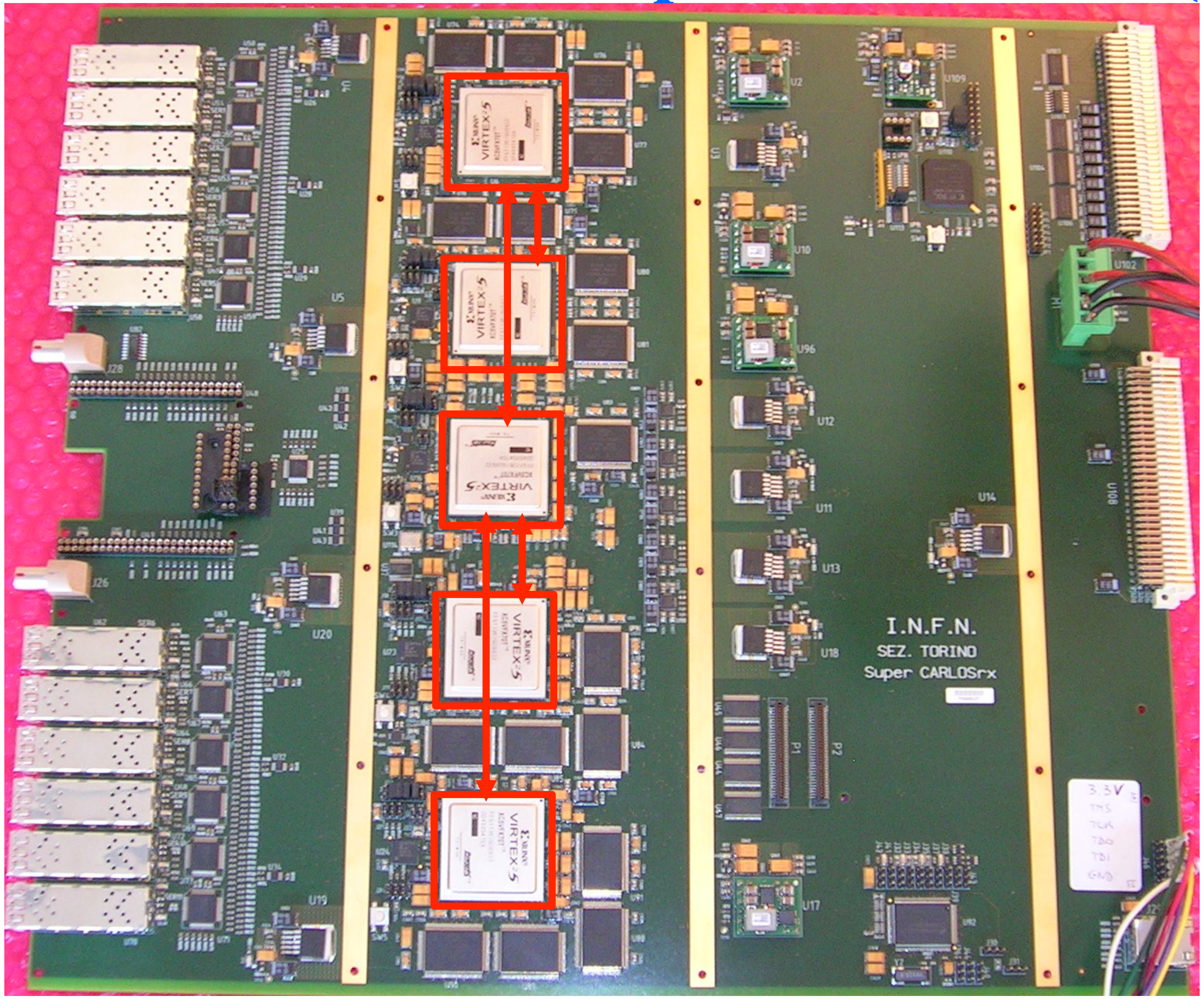


				CLB (1 = 4 slices = max 128 bits)			Block SelectRAM+			
Device[1]	RocketIO Transceiver Blocks	PowerPC Processor Blocks	Logic Cells[2]	Slices	Max Distr RAM (Kb)	18 X 18 Bit Multiplier Blocks	18 Kb Blocks	Max Block RAM (Kb)	DCMs	Maximum User I/O Pads
XC2VP2	4	0	3,168	1,408	44	12	12	216	4	204
XC2VP4	4	1	6,768	3,008	94	28	28	504	4	348
XC2VP7	8	1	11,088	4,928	154	44	44	792	4	396
XC2VP20	8	2	20,880	9,280	290	88	88	1,584	8	564

Device	Array	Slices	DSP48E	Block RAM (Kb)	PowerPC	RocketIO	I/O banks	User I/O
FX30T	80x38	5120	64	2448	1	8	12	360
FX70T	160x38	11200	128	5328	1	16	19	640
FX100T	160x56	16000	256	8208	2	16	20	680
FX130T	200x56	20480	320	10728	2	20	24	840
FX200T	240x68	30720	384	16416	2	24	27	960

Virtex-5 FPGA slices are organized differently from previous generations. Each Virtex-5 FPGA slice contains four LUTs and four flip-flops (previously it was two LUTs and two flip-flops)

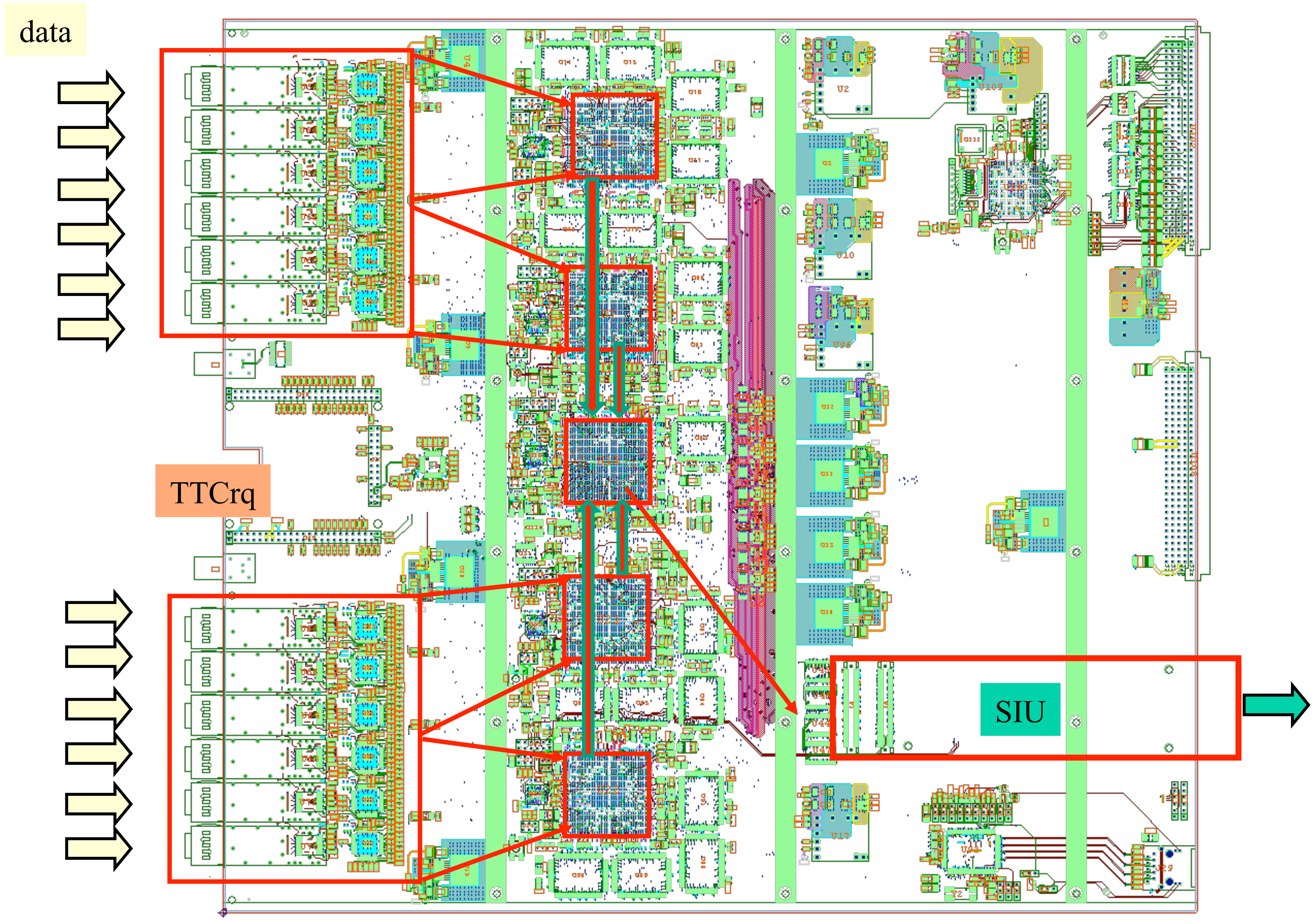
From CARLOSrx to SuperCARLOSrx (2011)



5 Xilinx FPGAs: Virtex5 FX70T

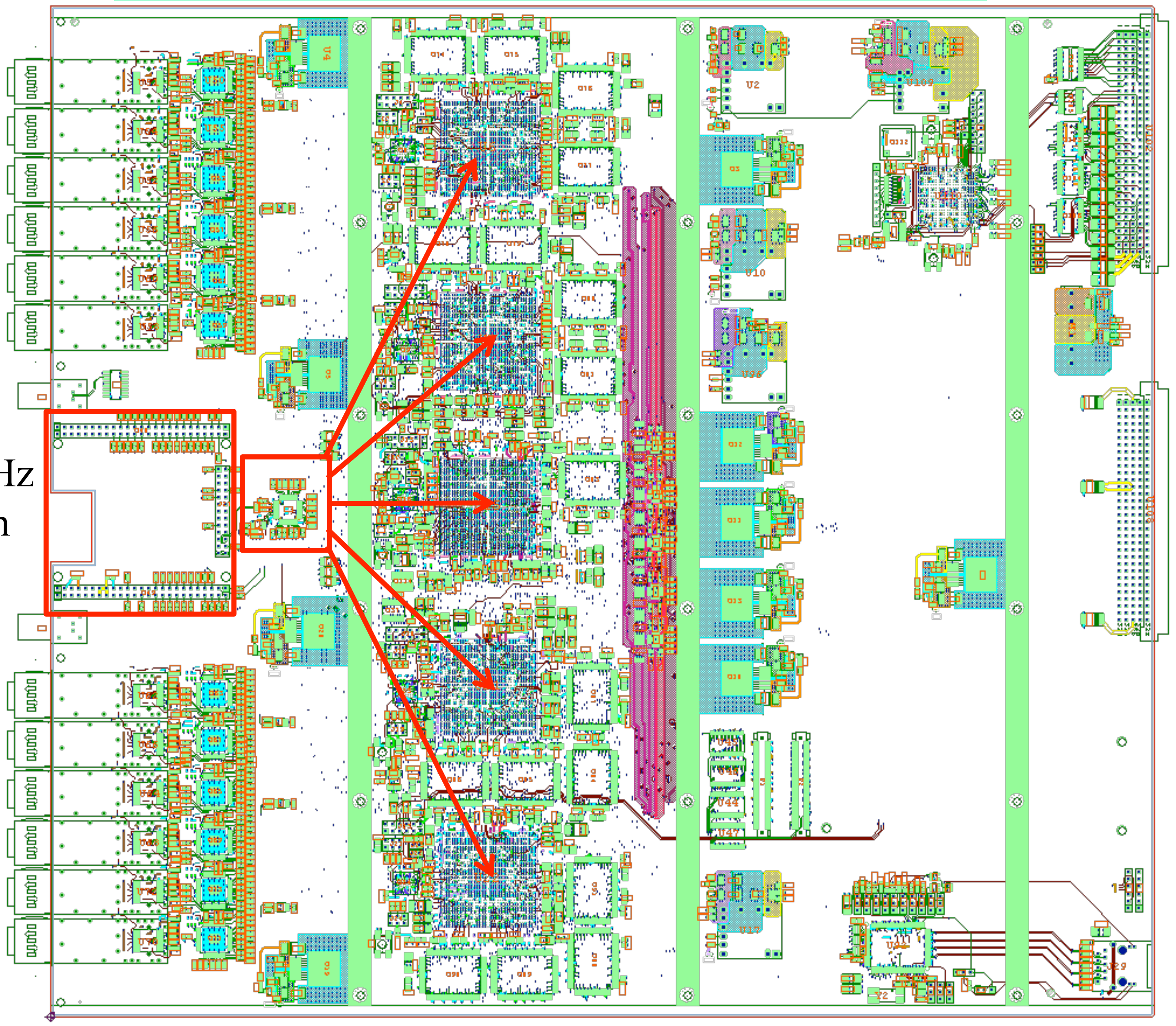
Serial connections running @ 2.5 Gb/s

superCARLOSrx dataflow

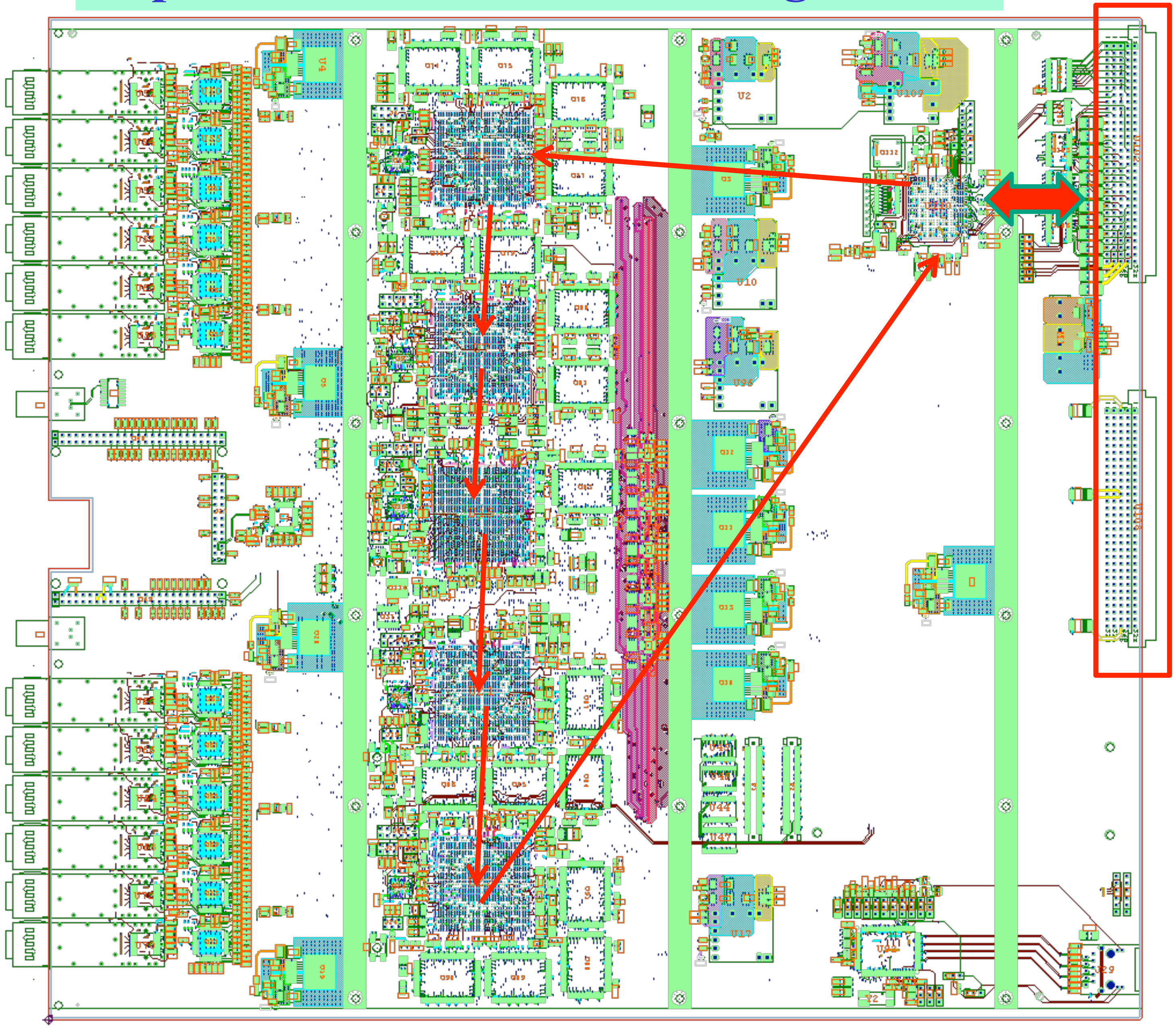


SuperCARLOSrx clock distribution

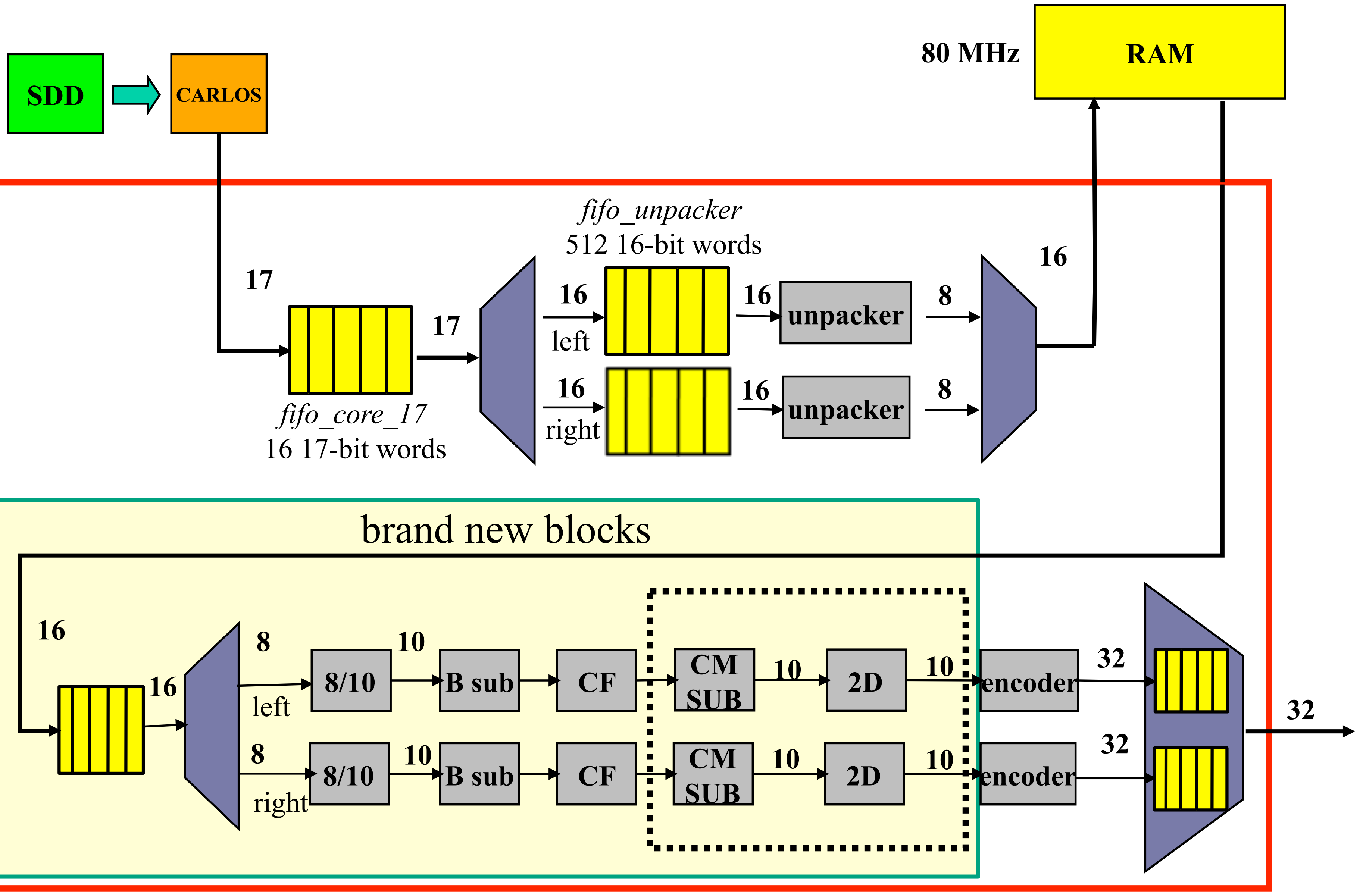
LHC 40 MHz clock from TTCrq

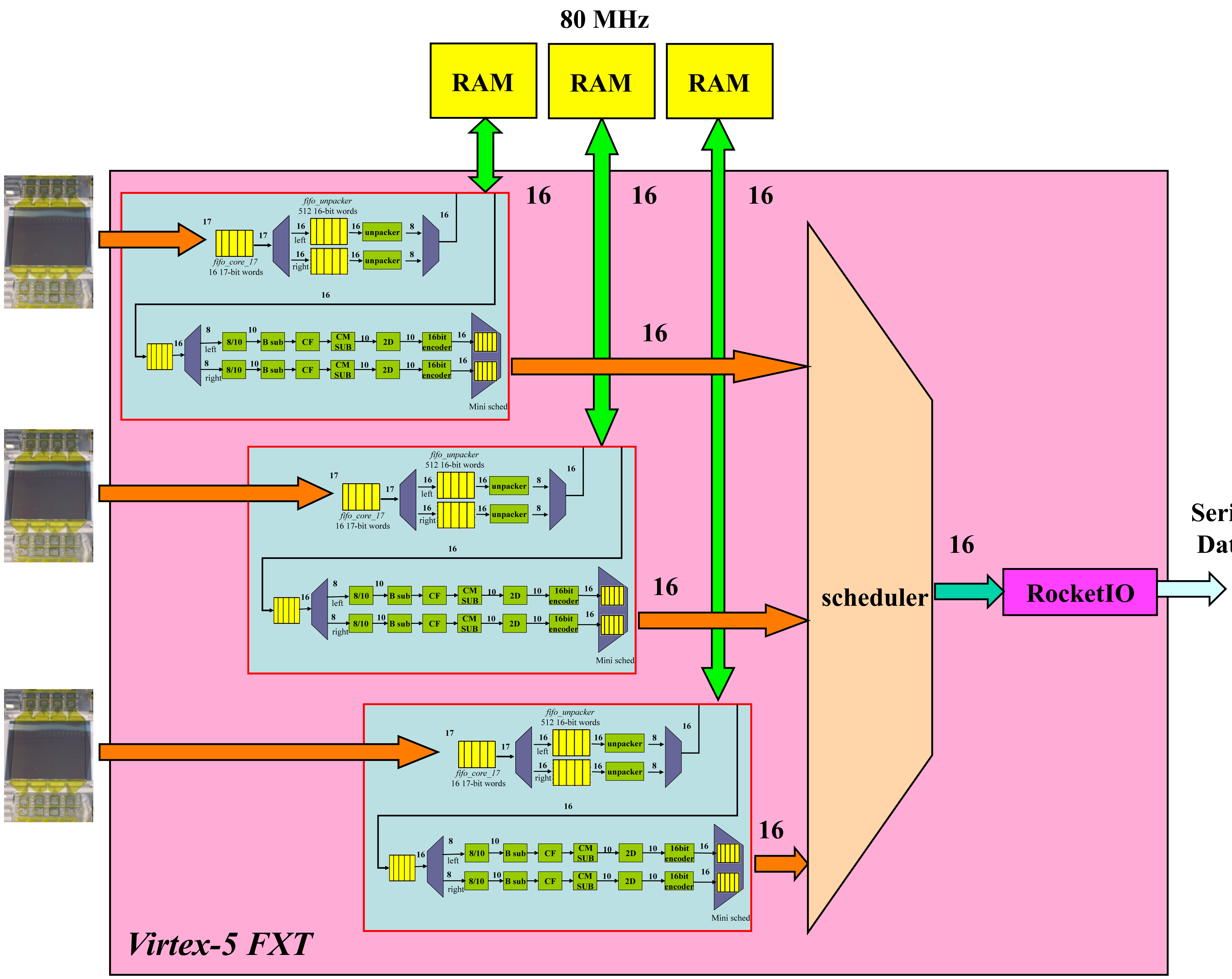


SuperCARLOSrx FPGA configuration

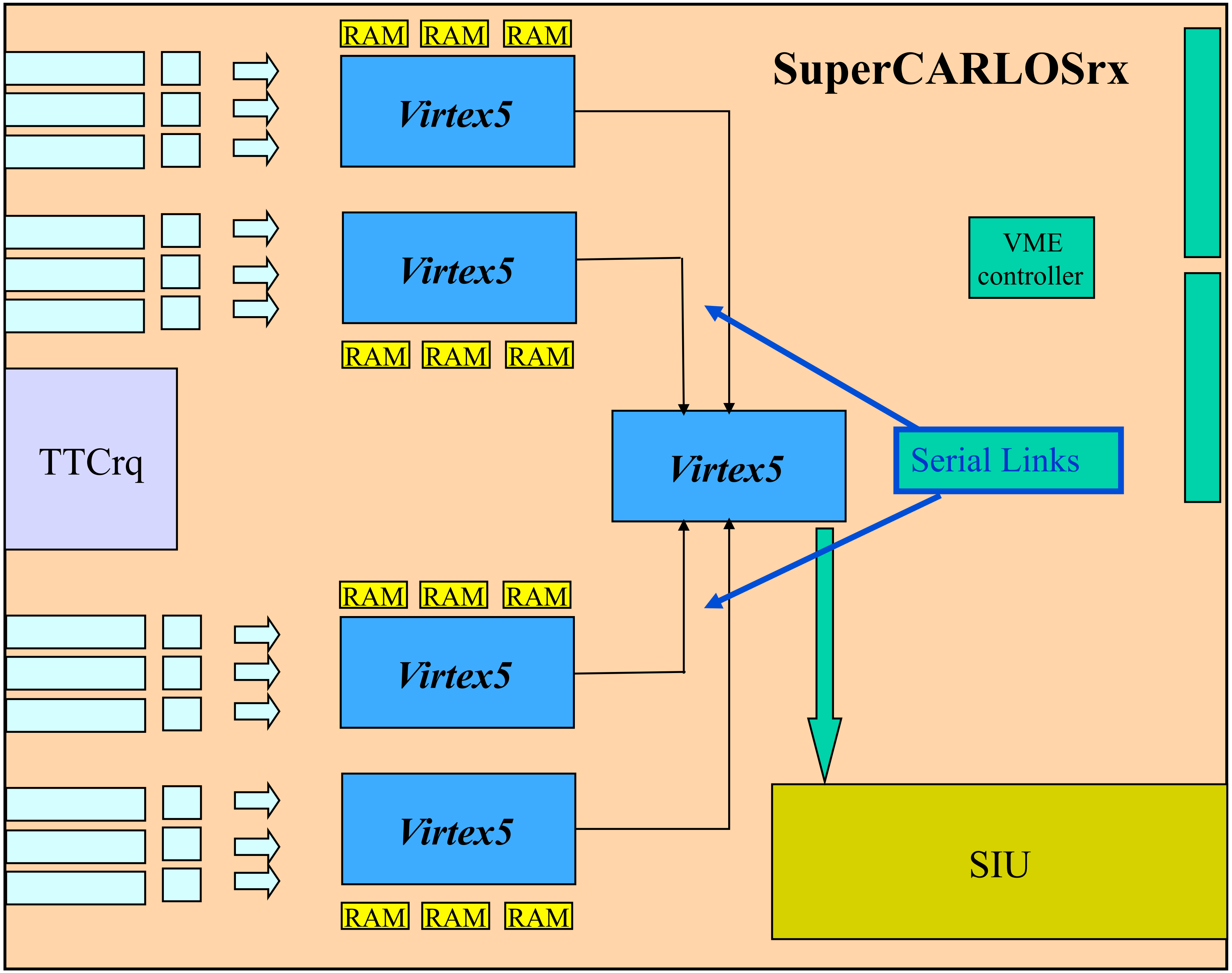


Block diagram for a single module



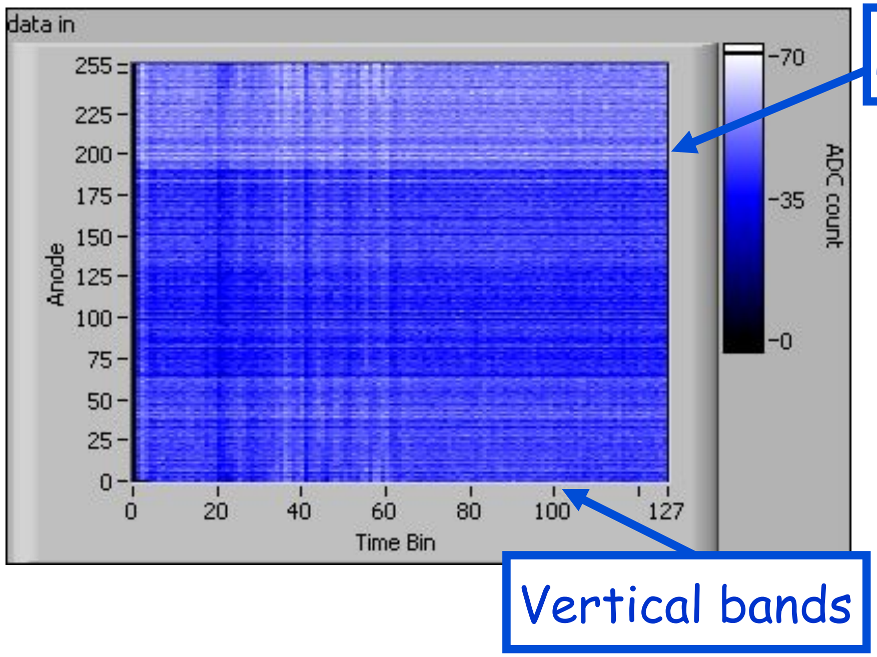


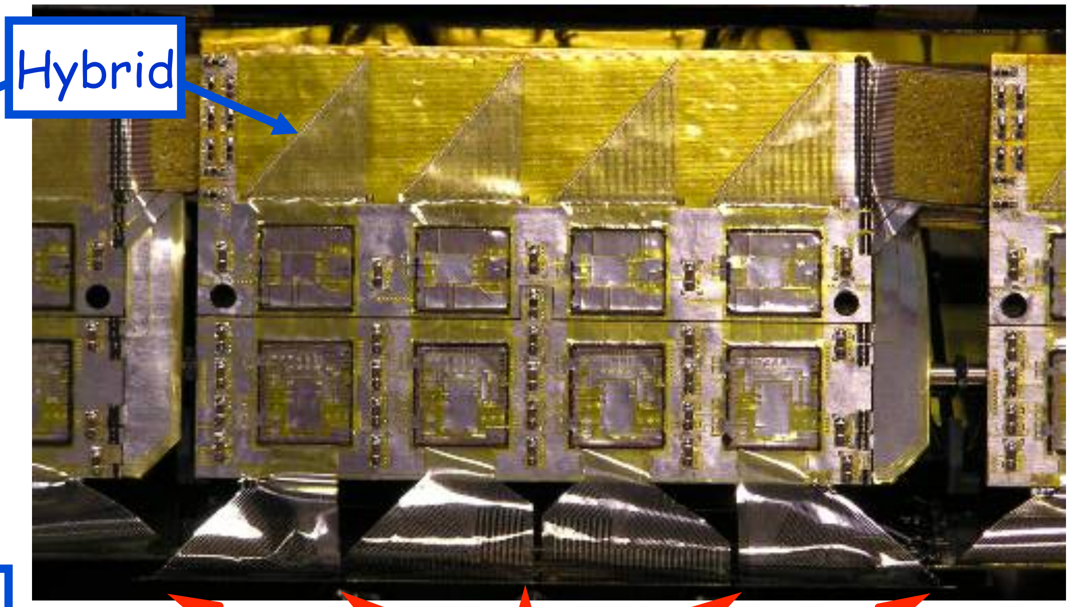
80 MHz
RAM
RAM
RAM
16
16
16
fifo_unpacker
512 16-bit words
17
17
16
left
16
right
16
16
unpacker
unpacker
8
8
16
fifo_core_17
16 17-bit words
16
16
8
left
8
right
10
10
8/10
8/10
B sub
B sub
CF
CF
CM SUB
CM SUB
2D
2D
10
10
16bit encoder
16bit encoder
16
16
Mini sched
16
16
16
scheduler
16
RocketIO
Serial Data
Virtex-5 FXT

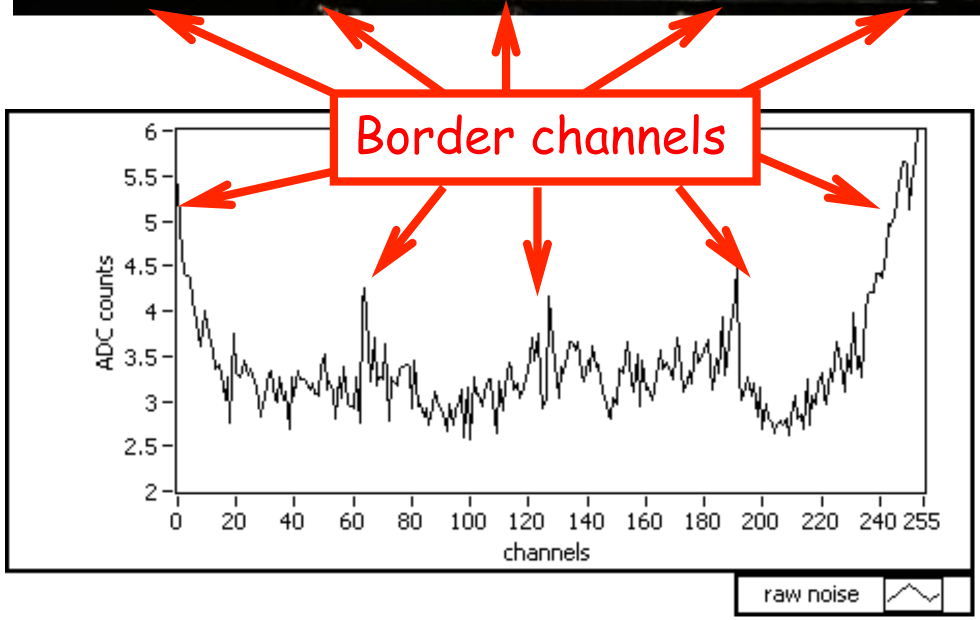

SuperCARLOSrx
RAM
RAM
RAM
Virtex5
Virtex5
RAM
RAM
RAM
VME controller
TTCrq
Virtex5
Serial Links
RAM
RAM
RAM
Virtex5
Virtex5
RAM
RAM
RAM
SIU

Why a common mode noise filter ?

Common mode noise







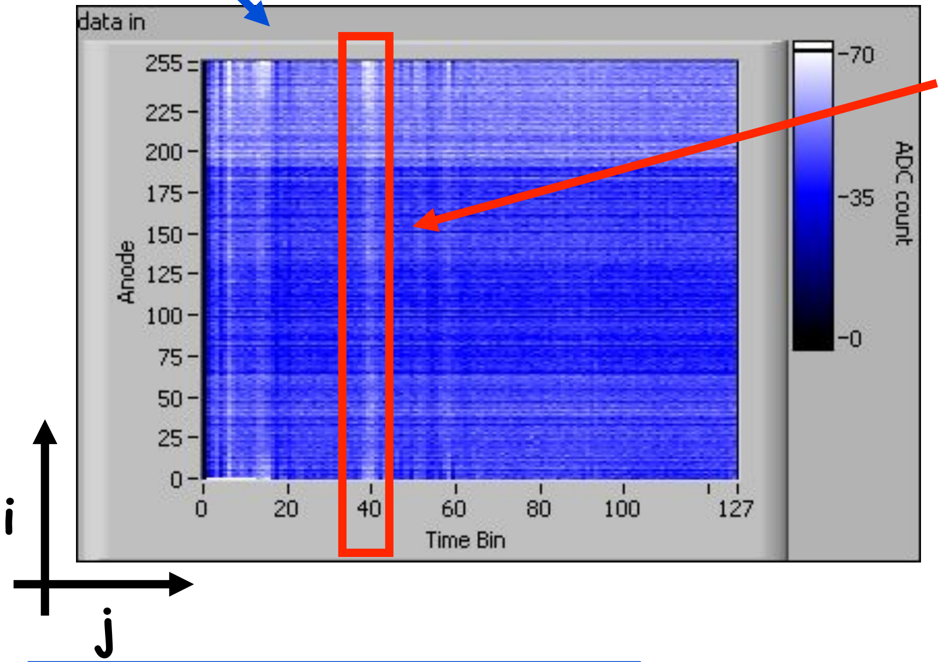
• The common mode noise is a coherent fluctuation of a group of electronic channels induced by external sources

•Vertical bands on Raw Data Plot

• Border channel effect due to microcables proximity

The Algorithm

Single Hybrid



Common mode shift for each time bin

pedestal

$$p_i = \frac{\sum_{j=1}^{TB} a_{ij}}{TB}$$

$$s_j = \frac{\sum_{i=1} (a_{ij} - p_i)}{m}$$

$$m \le n = 256$$

Average over a number of events

no hits and dead channels

Evaluated on-line event-by-event

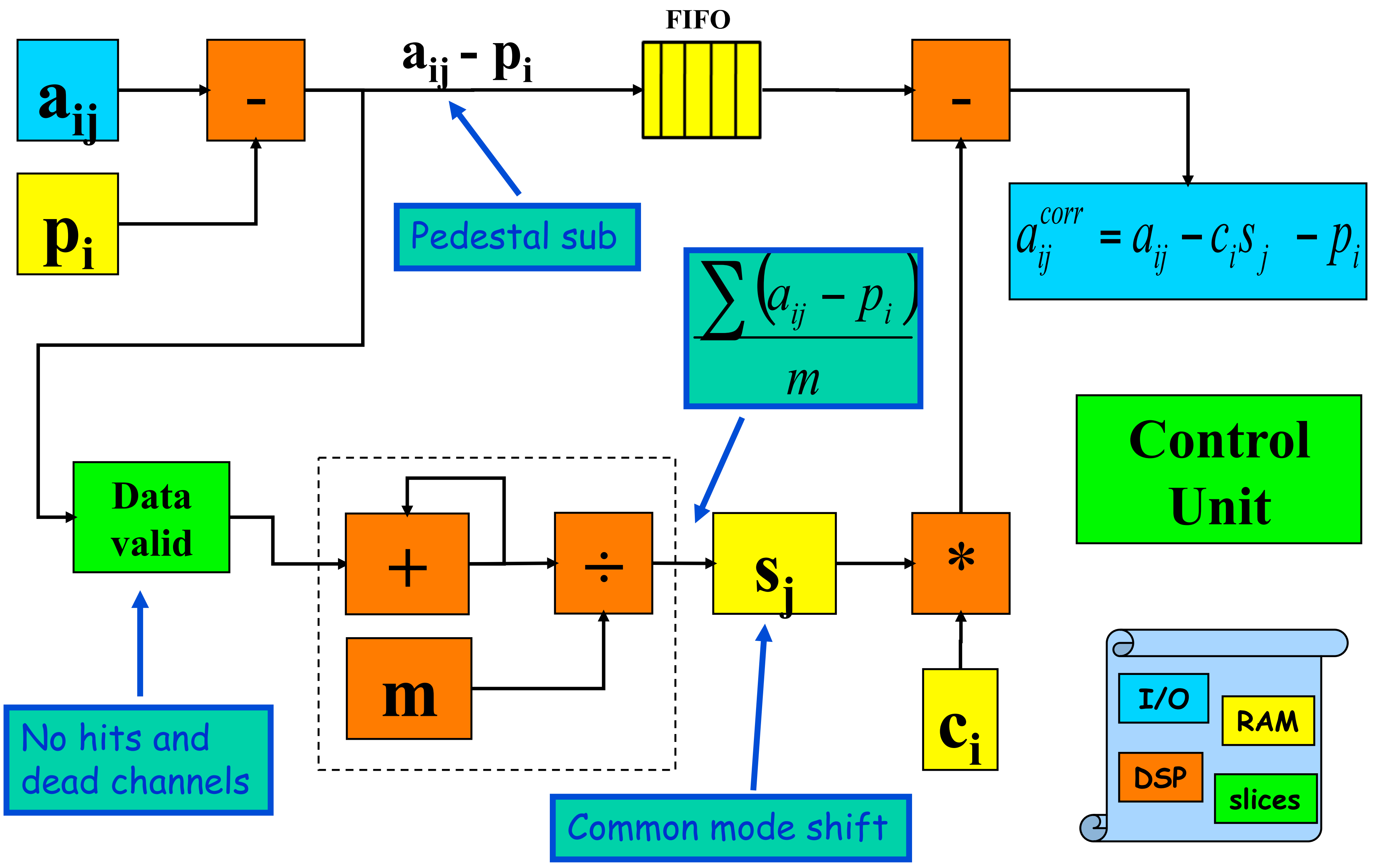
Correlation coefficient for each channel

$$c_i = \frac{\sum_j (a_{ij} - p_i) s_j}{\sum_j s_j^2}$$

$$a_{ij}^{corr} = a_{ij} - c_i s_j^* - p_i$$

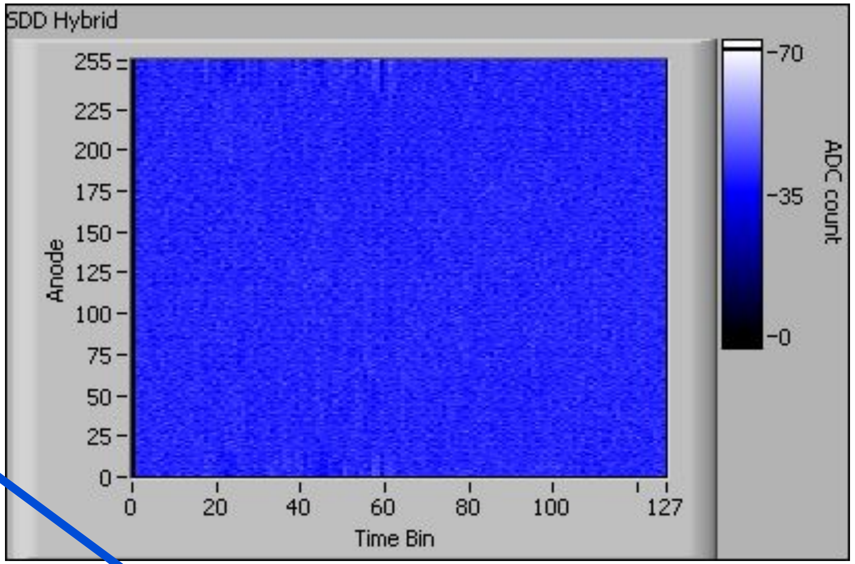
common mode and pedestal subtraction

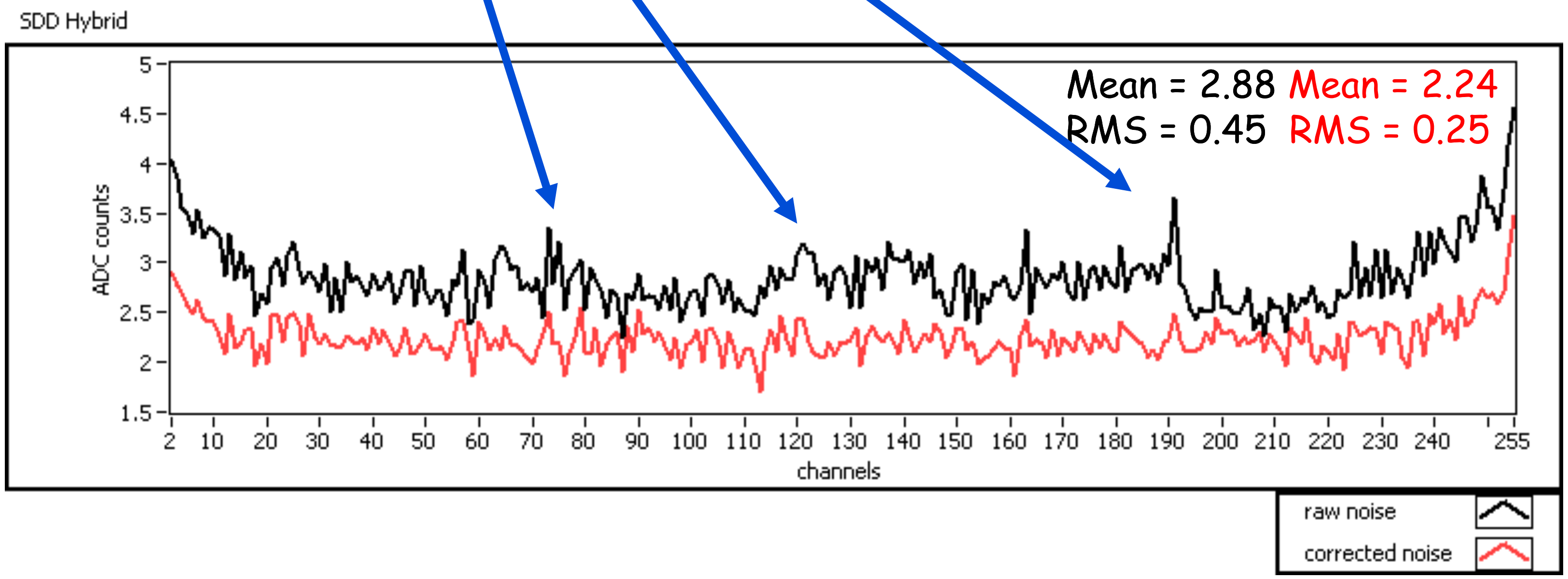
FPGA Implementation



Single Hybrid Test results

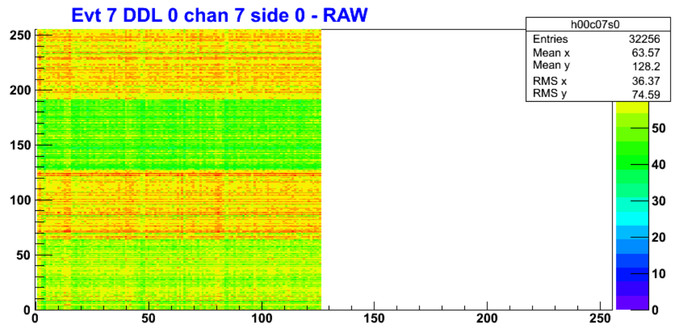
Minimize border channel effects

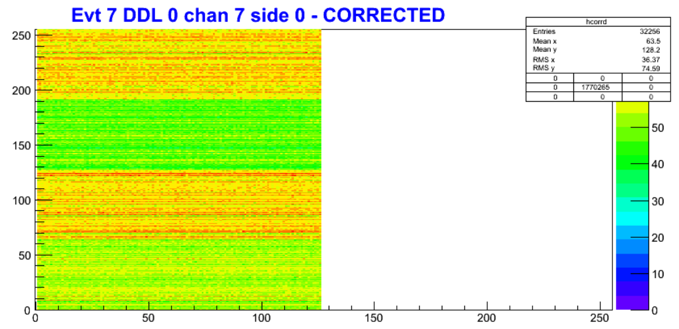


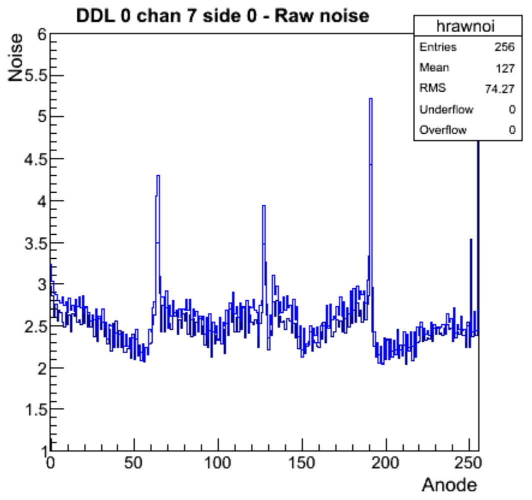


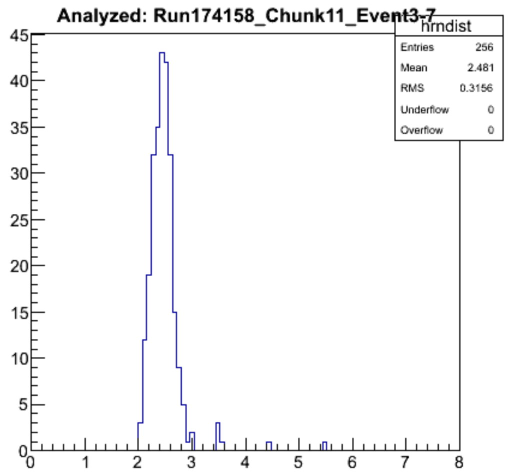
Mean = 2.88 Mean = 2.24
RMS = 0.45 RMS = 0.25

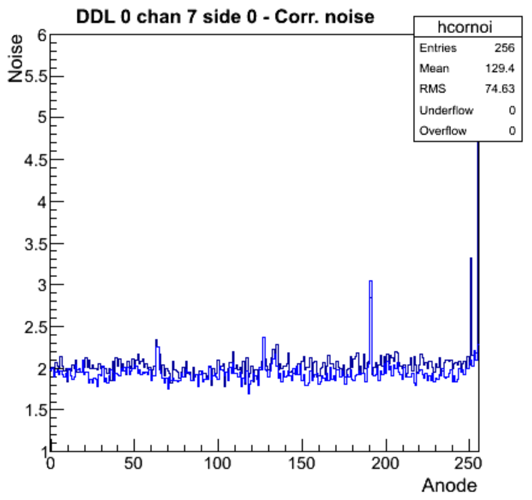
Common mode reduction algorithm tuning

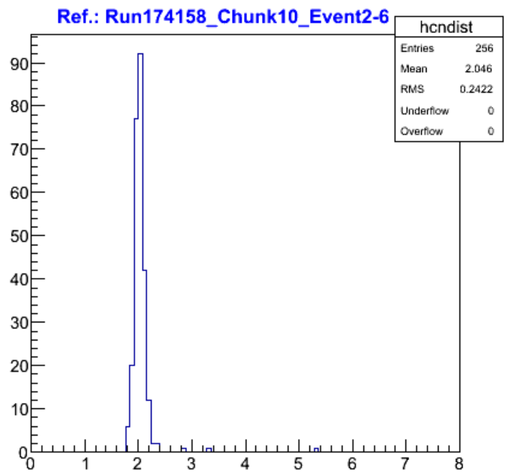










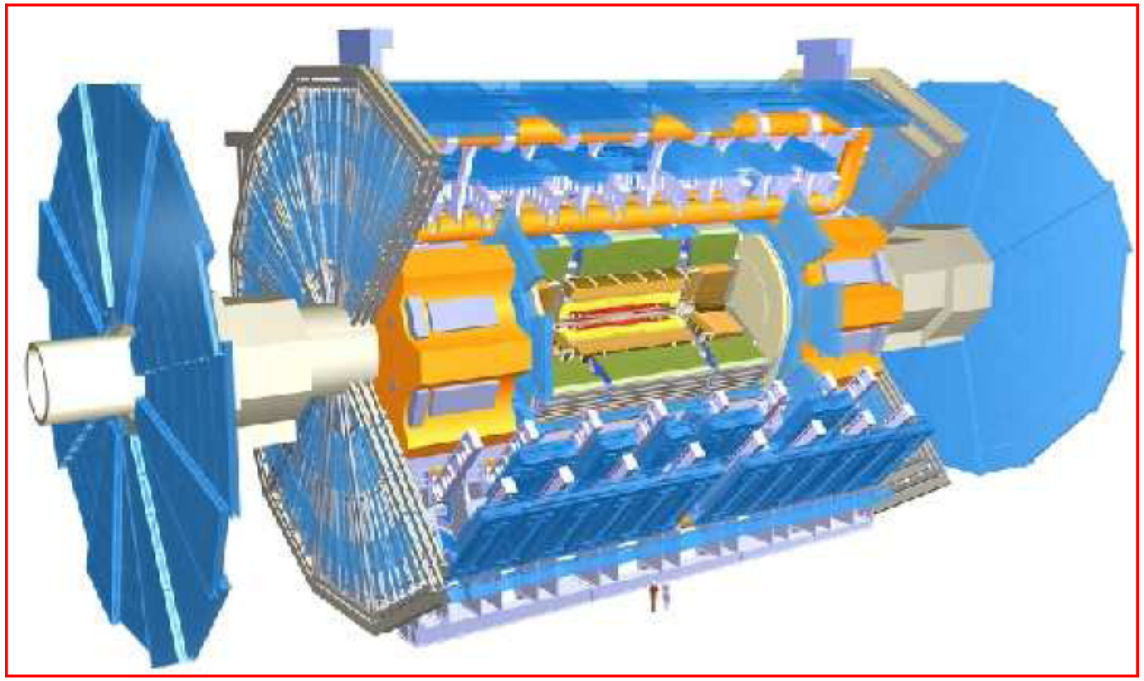


- Software validation is in progress in Torino (Luciano Ramello)
- Hardware coding of the algorithm started

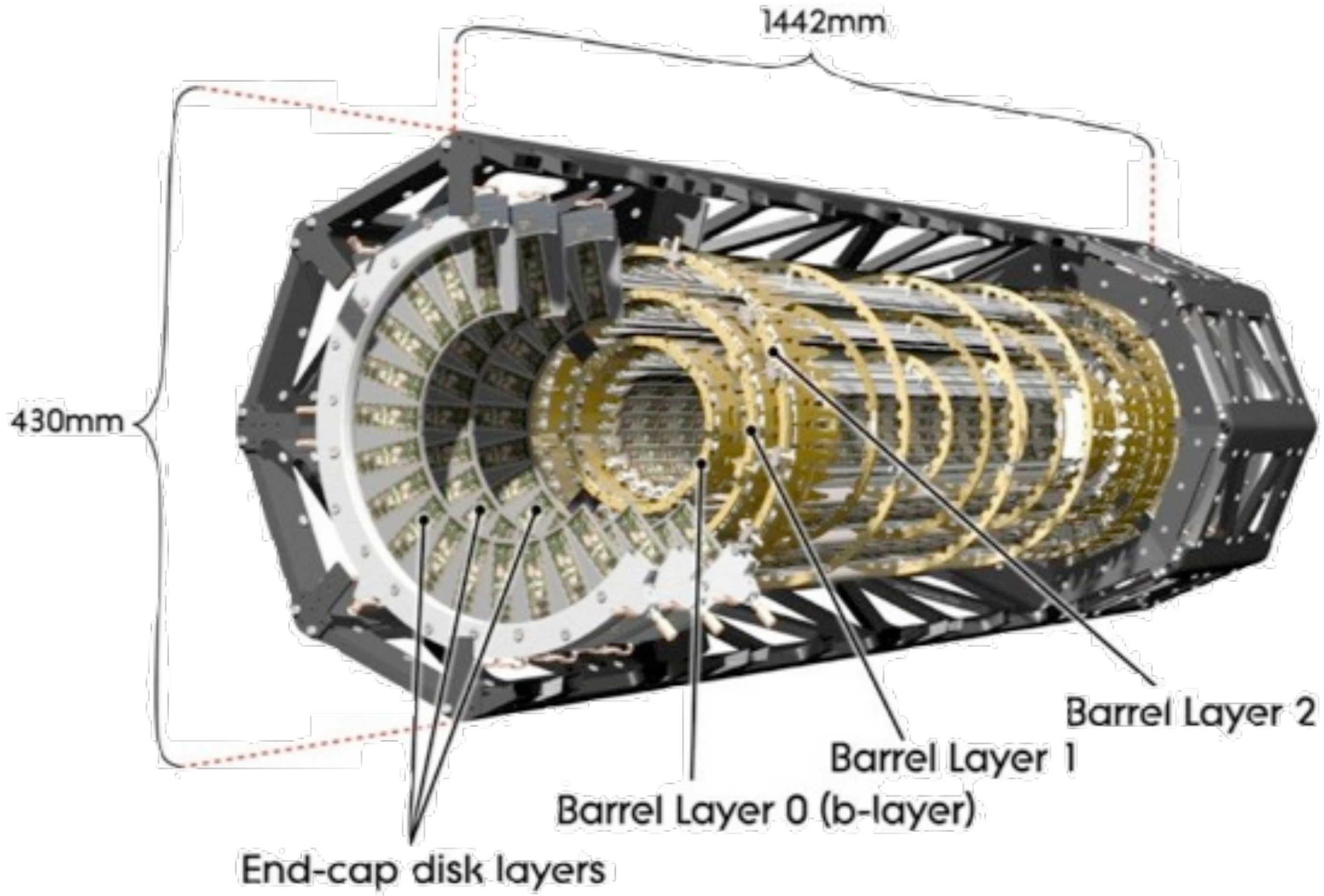
SuperCARLOSrx Summary

- A new HW board is in place
- A lot of new firmware features can be implemented and tested

Case study (2): readout board for ATLAS IBL



Atlas Pixel Detector Overview



Atlas Pixel Detector Overview

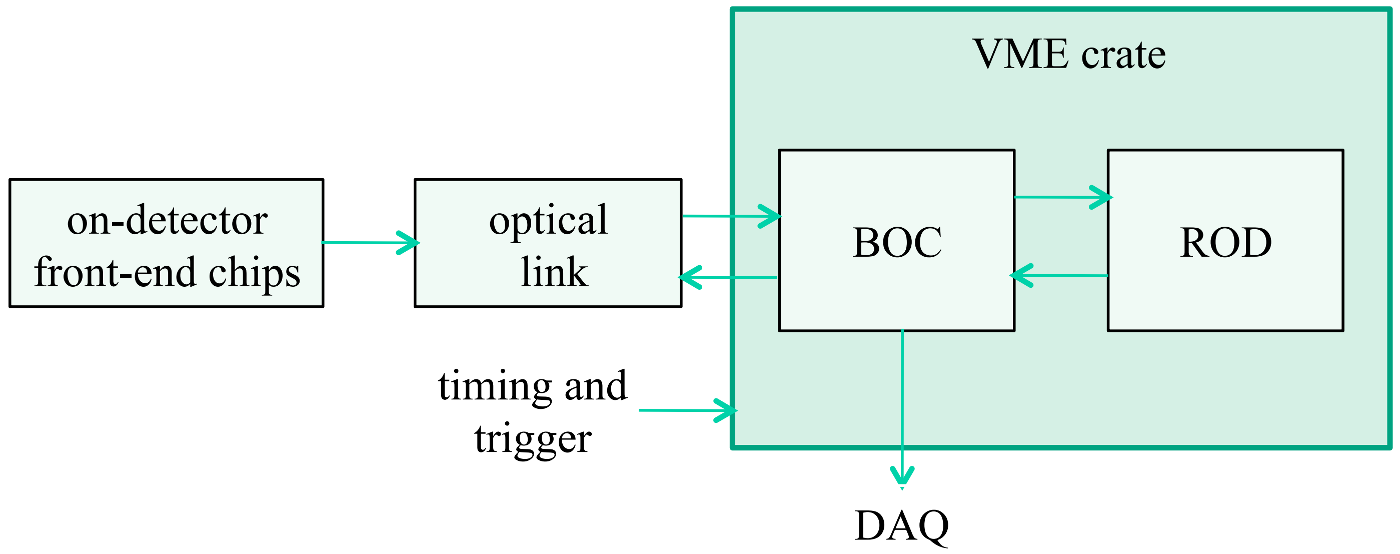
Innermost detector: tracking and vertexing
3 barrel layers - b-layer closer the beam pipe (<r>=5cm)
80M pixels in total (50 um x 400 um)

front-end chip : FE-I3 (2880 channels x chip)
basic unit: module (sensitive region coupled with 16 FE-I3)
1774 modules in total

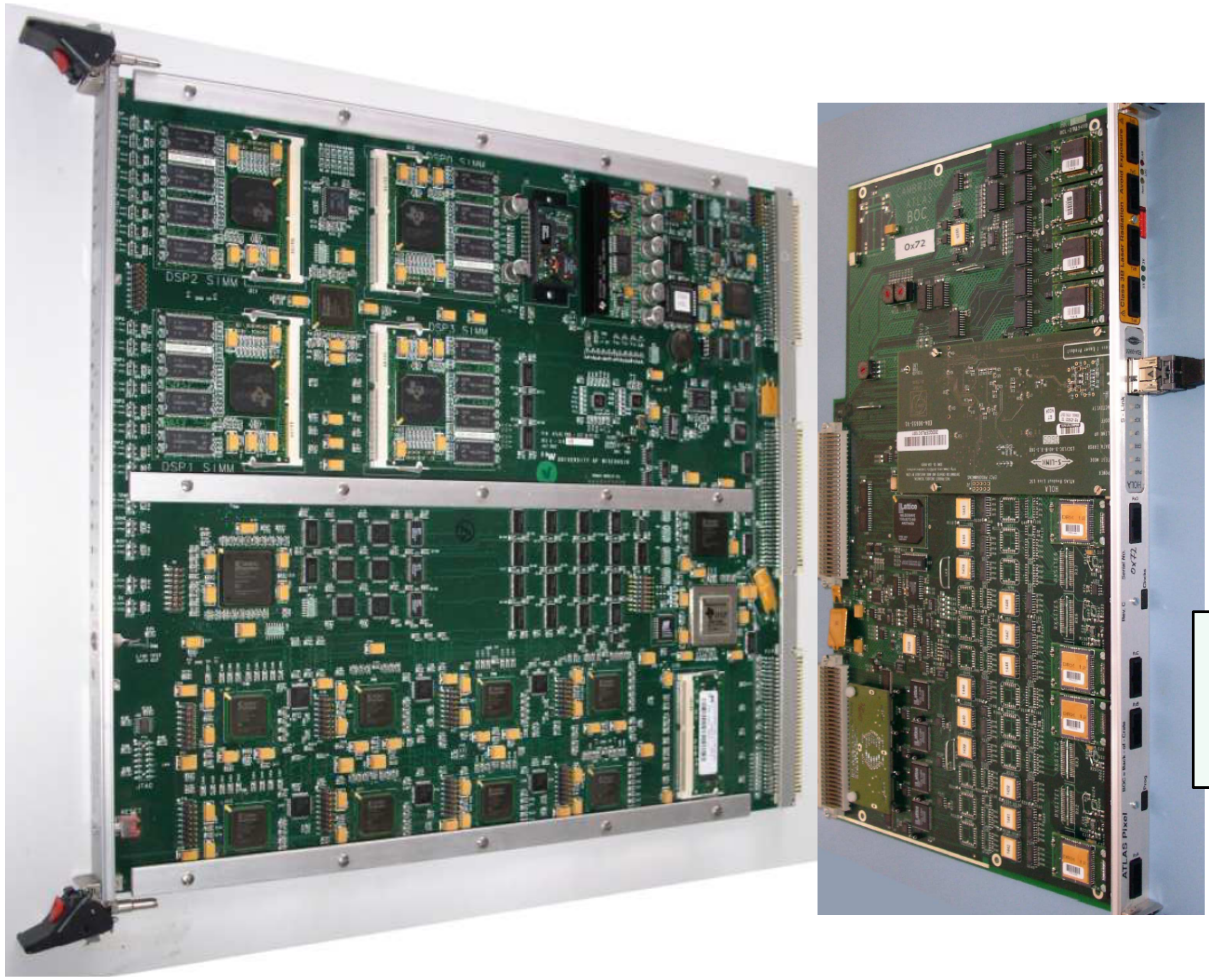
one Module Control Chip (MCC) x module different readout schemes:
b-layer: 2 link @ 80 Mb/s (160 Mb/s)
layer 1: 1 link @ 80 Mb/s
layer 2: 1 link @ 40 Mb/s

Each off-detector readout unit handles up to 160 MB/s
(8 links @ full speed = 1 S-Link)

Readout electronics for the Pixel Detector



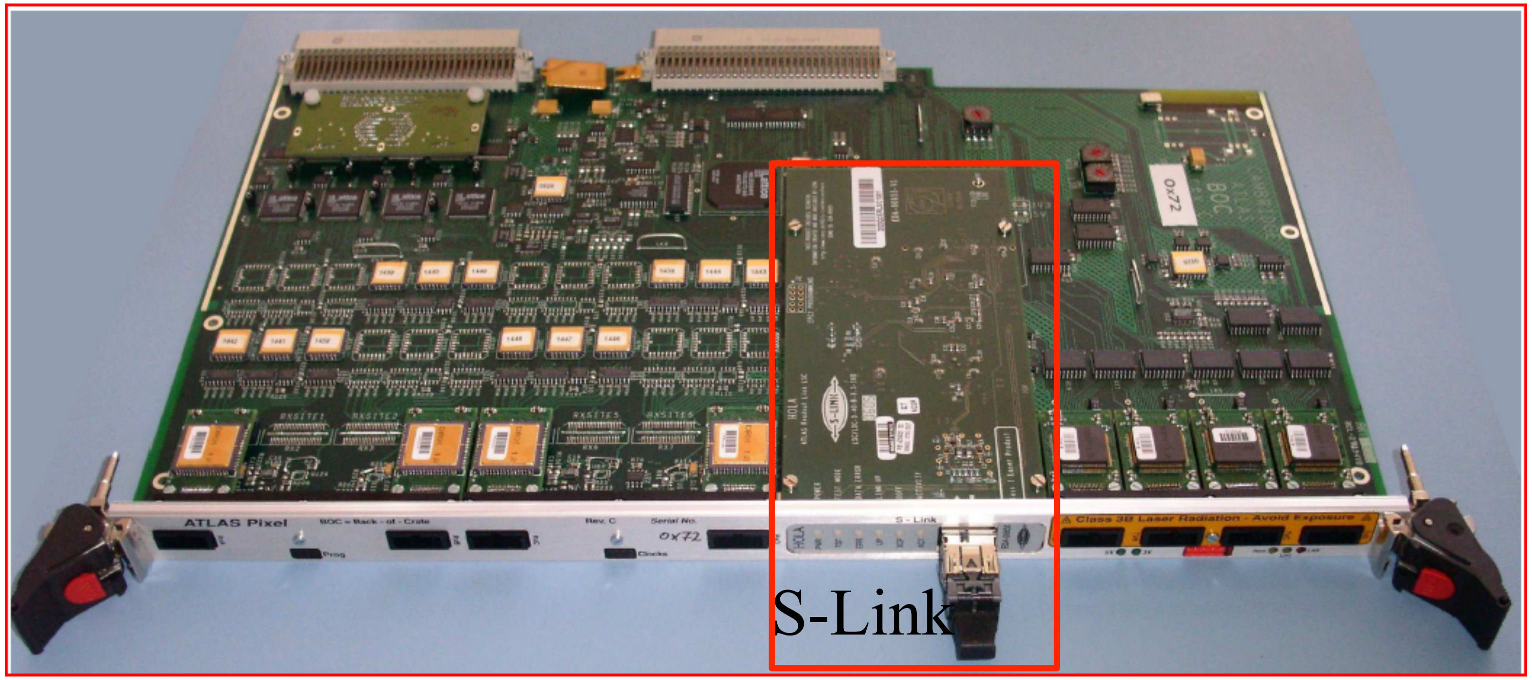
Old BOC-ROD pair



1
S-Link

8
on-detector
front-end chips

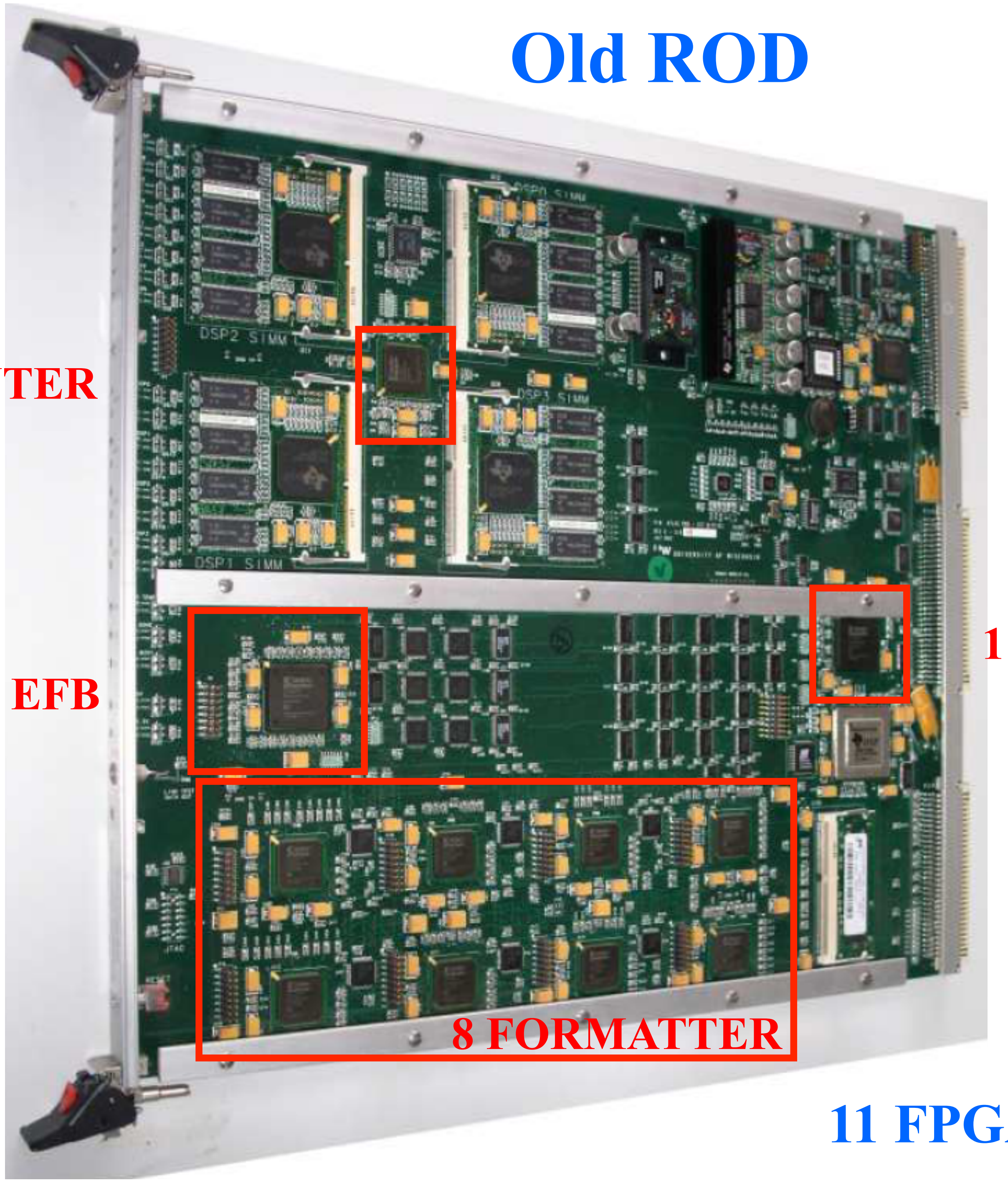
Old BOC



The BOC card serves as the optical interface for the off detector part and provides the complete timing functionality for the ATLAS pixel detector and its readout

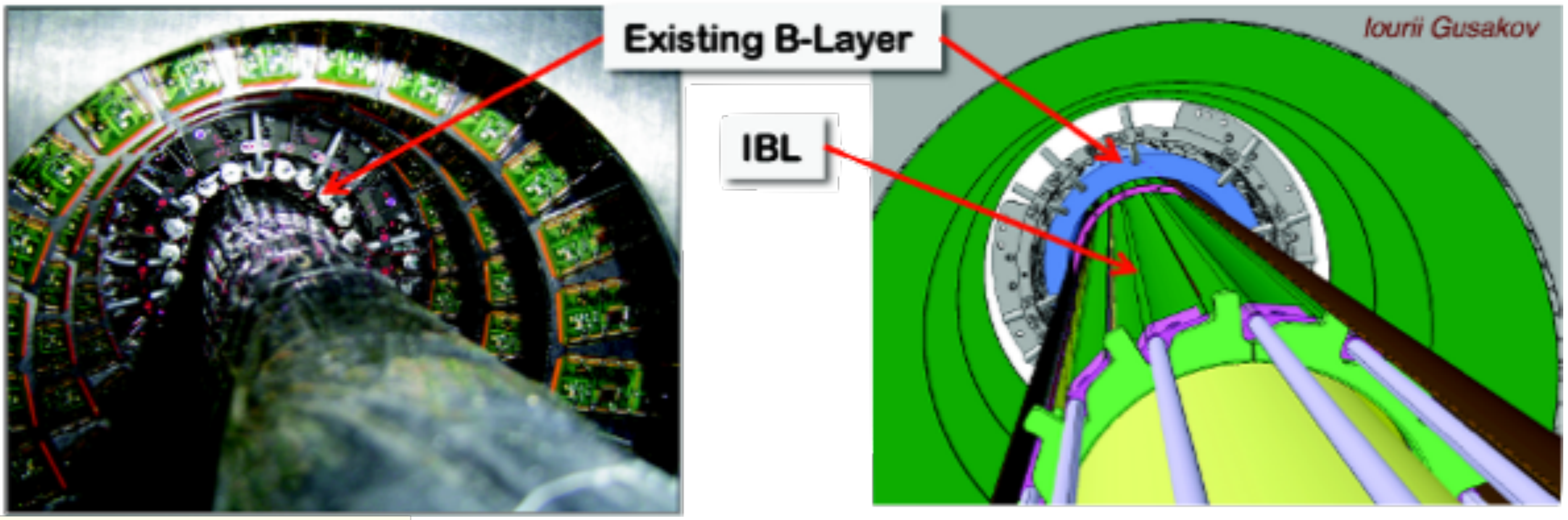
- Receive and distribute the clock signal as provided by the TIM.
- Receive the electrical control signals for the modules from the ROD, convert them to Bi-Phase Mark encoded optical signals, and dispatch these down the clock and control fibres to the detector modules.
- Provide masking, timing, and laser current adjustment functions for the transmitted control signals.
- Receive the optical data packets from the detector modules via the fibre ribbons, convert them to electrical form, and pass them to the ROD.
- Provide a timing adjustment, threshold adjustment, and clock synchronisation for the received optical data.
- Take the data stream from the ROD in parallel form, and provide the S-Link path via the Readout Buffer to the DAQ.

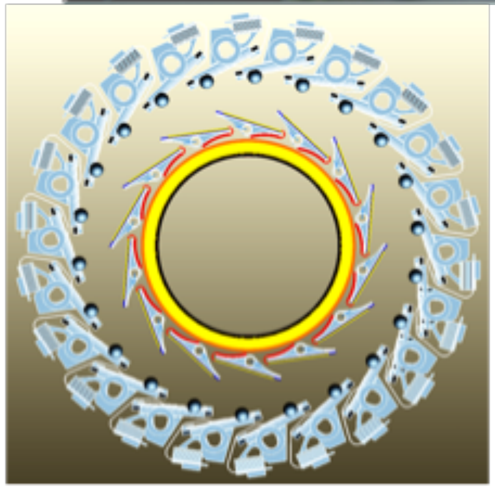
Old ROD

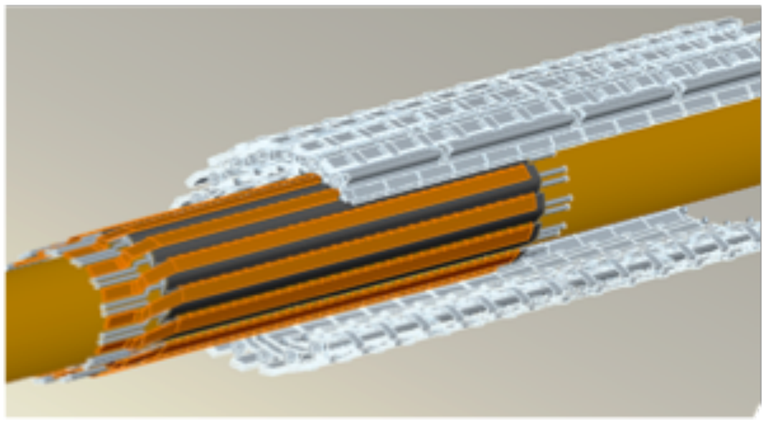


11 FPGAs + 5 DSPs

Pixel Detector Upgrade: Inner Barrel Layer (IBL)







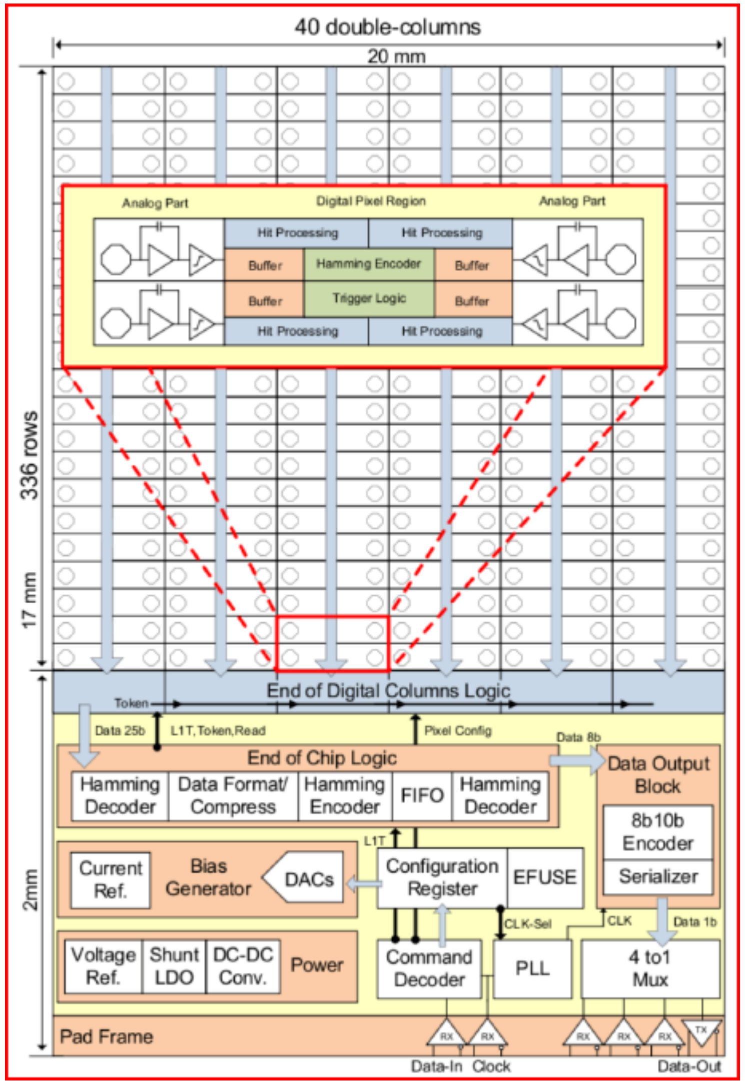
12 millions pixels
Pixel size : 50 x 250 um
<R> = 33 mm
|Z| < 33.2 cm
14 Staves
224 Modules

Major Goals:

- strengthen the tracking capability by increasing both redundancy and precision;
- preserve the performances of the Pixel Detector for effects due to the increased luminosity expected after LHC upgrades (greater pile-up and radiation doses).

New front-end electronics for IBL

A new front-end ASIC, called **FeI4**, has been designed to face the larger occupancy as well as to manage the increased bandwidth expected for IBL.



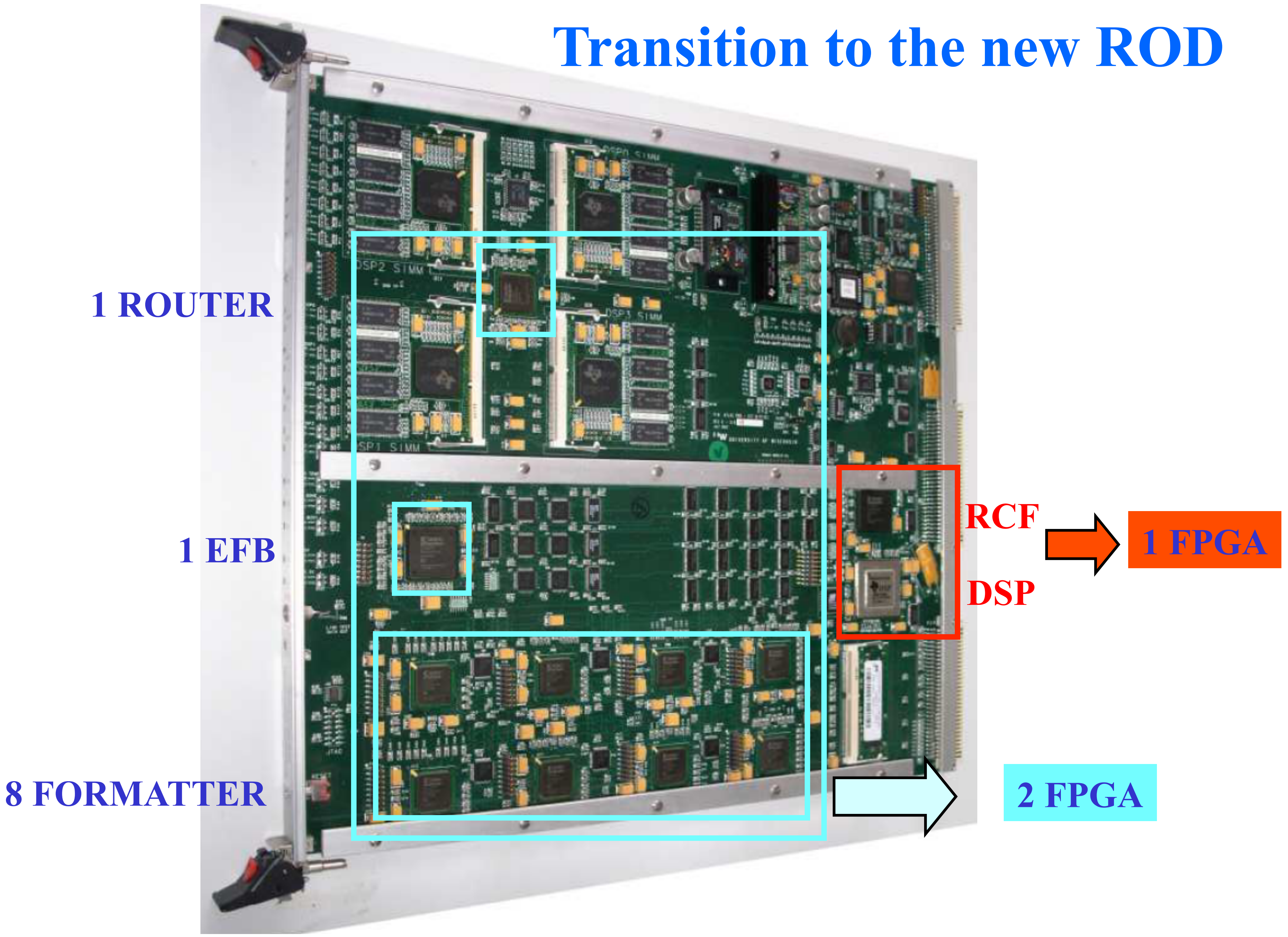
pixel array: 336x80 pixels

Each FeI4 pixel contains an independent, free running amplification stage with adjustable shaping, followed by a discriminator with independently adjustable threshold. The chip keeps track of the firing time of each discriminator as well as the time over threshold (ToT) with 4-bit resolution, in counts of an externally supplied clock, nominally 40 MHz. Information from all discriminator firings is kept in the chip for a latency interval, programmable up to 255 cycles of the external clock. Within this latency interval, the information can be retrieved by supplying a trigger. The data output is serial over a current-balanced pair (similar to LVDS). The primary output mode is 8b/10b encoded with 160 Mb/s rate. The FE-I4 is controlled by a serial LVDS input synchronized by the external clock.

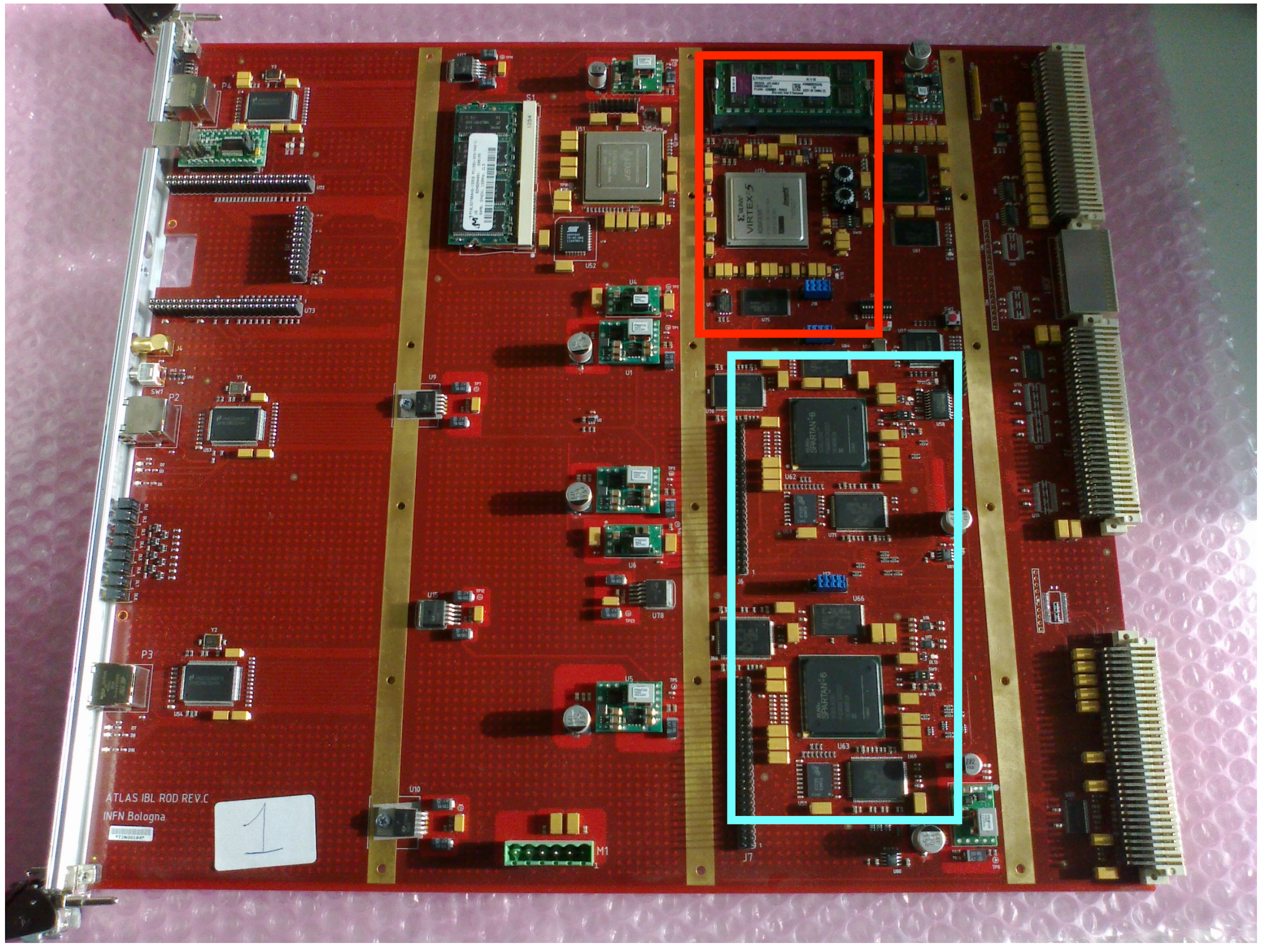
Proposal of a new off-detector readout

When specifying the design of the DAQ chain for the IBL, the first open question was whether the existing ROD was sufficient or if a new one was needed. The existing ROD firmware could be modified to operate with the IBL module data format; however, the hardware of the board is designed to operate with a maximum of eight 160 Mb/s input links (from the modules) to one output S-Link, while, in order to respect the IBL natural modularity, thirty-two 160 Mb/s links to four S-Links have to be handled. This consideration, together with the 4 MB/s bandwidth limitation on the VME bus and the obsolescence of the components led to the decision to design **a new BOC-ROD pair**.

Transition to the new ROD



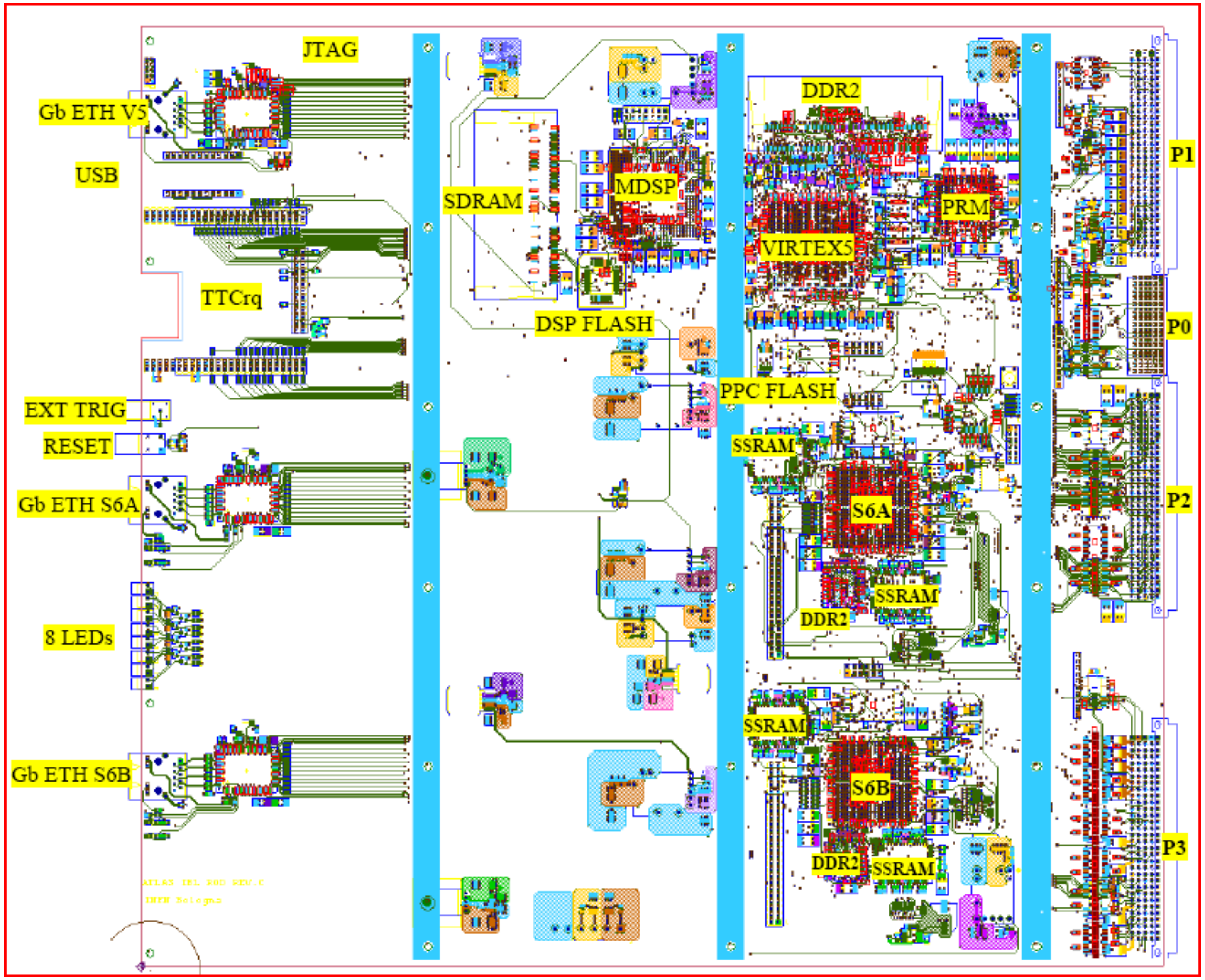
New IBL ROD



1 FPGA with PowerPC inside: Virtex5

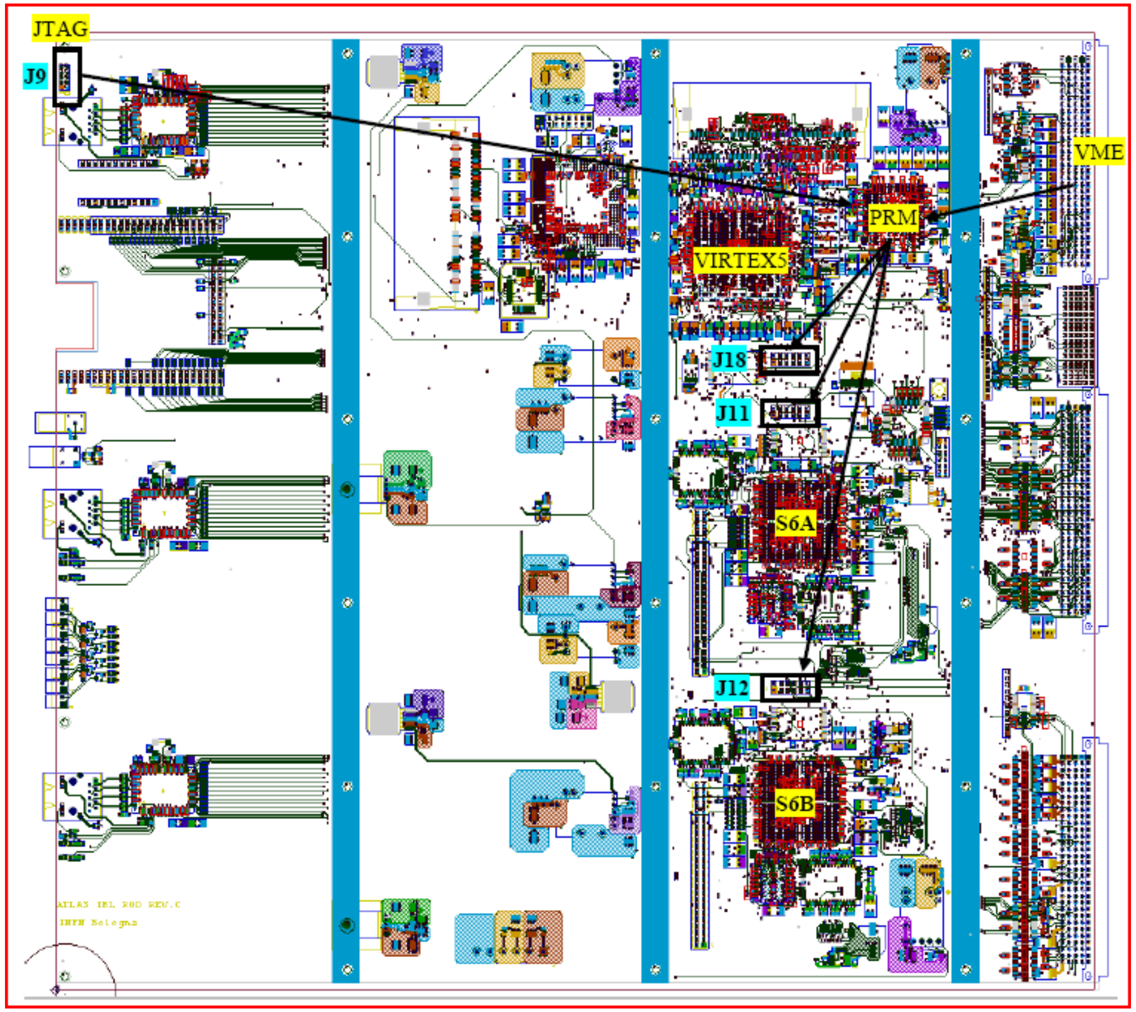
2 FPGAs: Spartan6

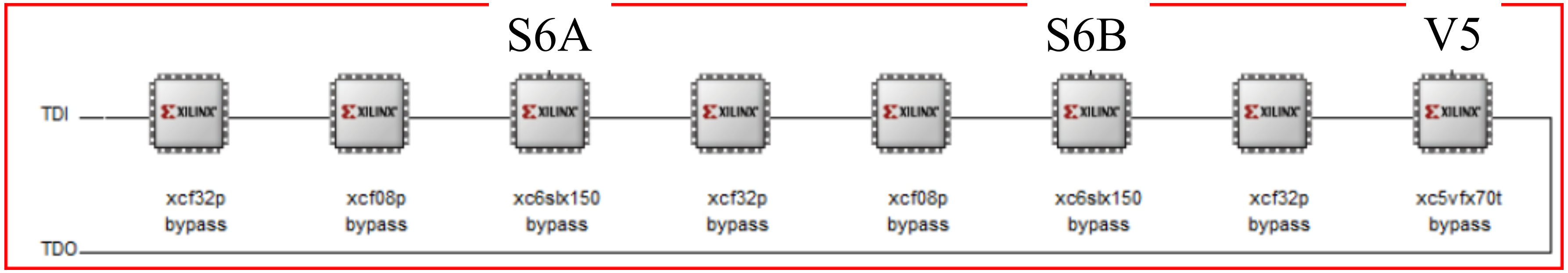
Fewer devices implementing the functionality of 4 old RODs
14-layer PCB


JTAG
Gb ETH V5
USB
TTCrq
EXT TRIG
RESET
Gb ETH S6A
8 LEDs
Gb ETH S6B
ATLAS IBL ROD REV.C
INFN Bologna
SDRAM
MDSP
DSP FLASH
DDR2
PRM
VIRTEX5
PPC FLASH
SSRAM
S6A
SSRAM
DDR2
SSRAM
S6B
DDR2
SSRAM
P1
P0
P2
P3

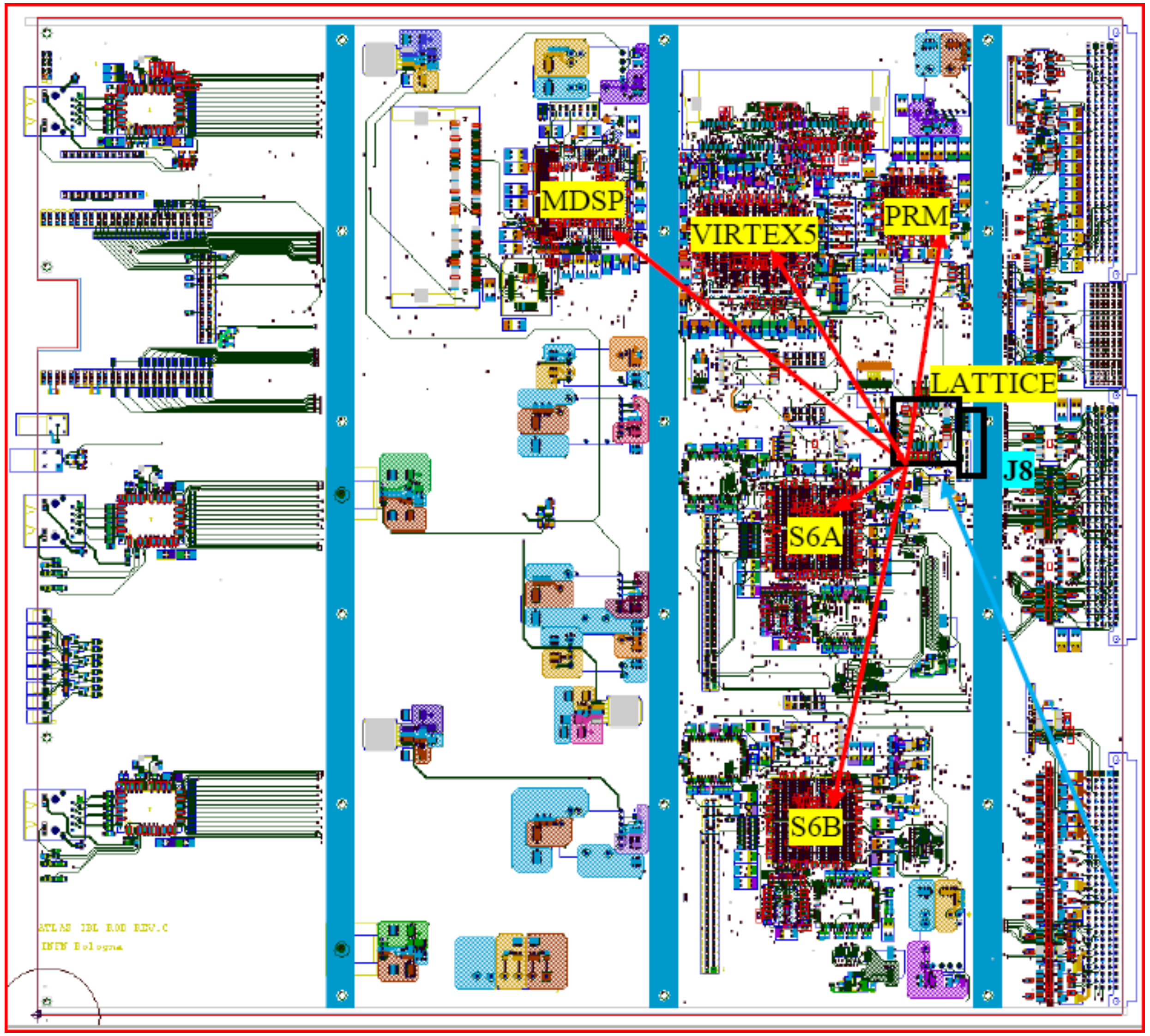
Configuration

The FPGAs can be configured in 2 ways:
- via JTAG from the front panel,
- directly via VME by a VME CPU (we use STAPL files with JAM player)



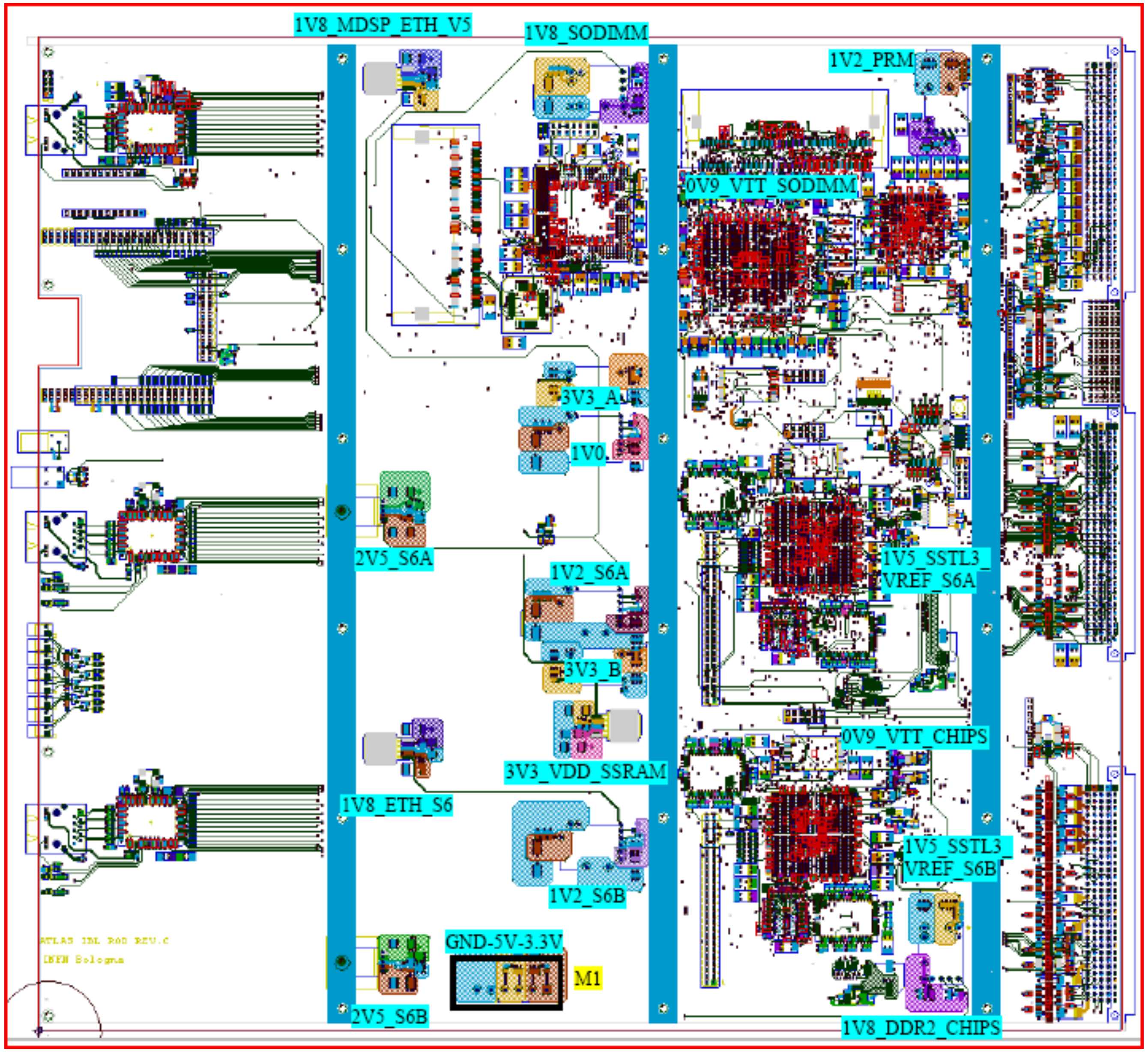


Clock distribution

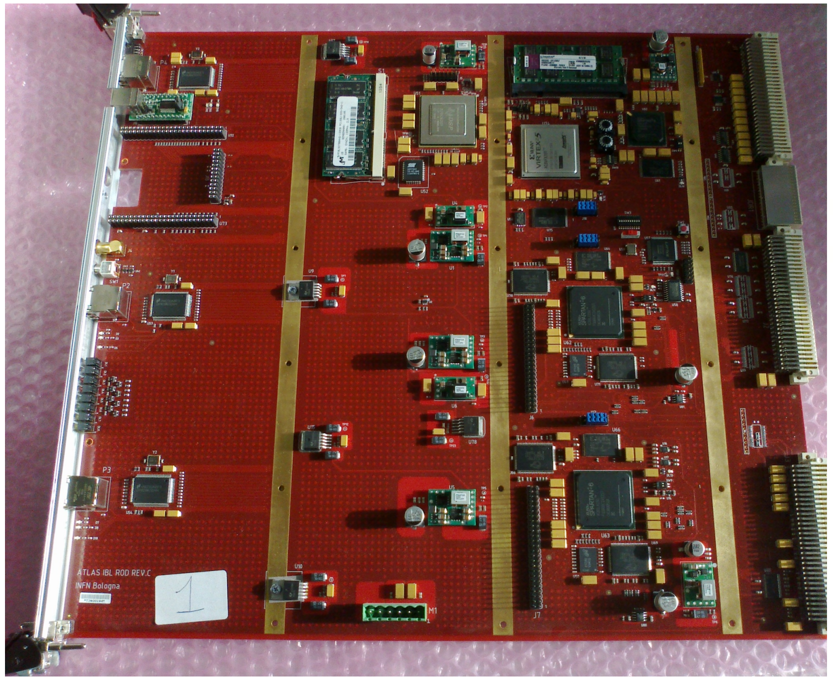


TIM → BOC → ROD (40 MHz clock)

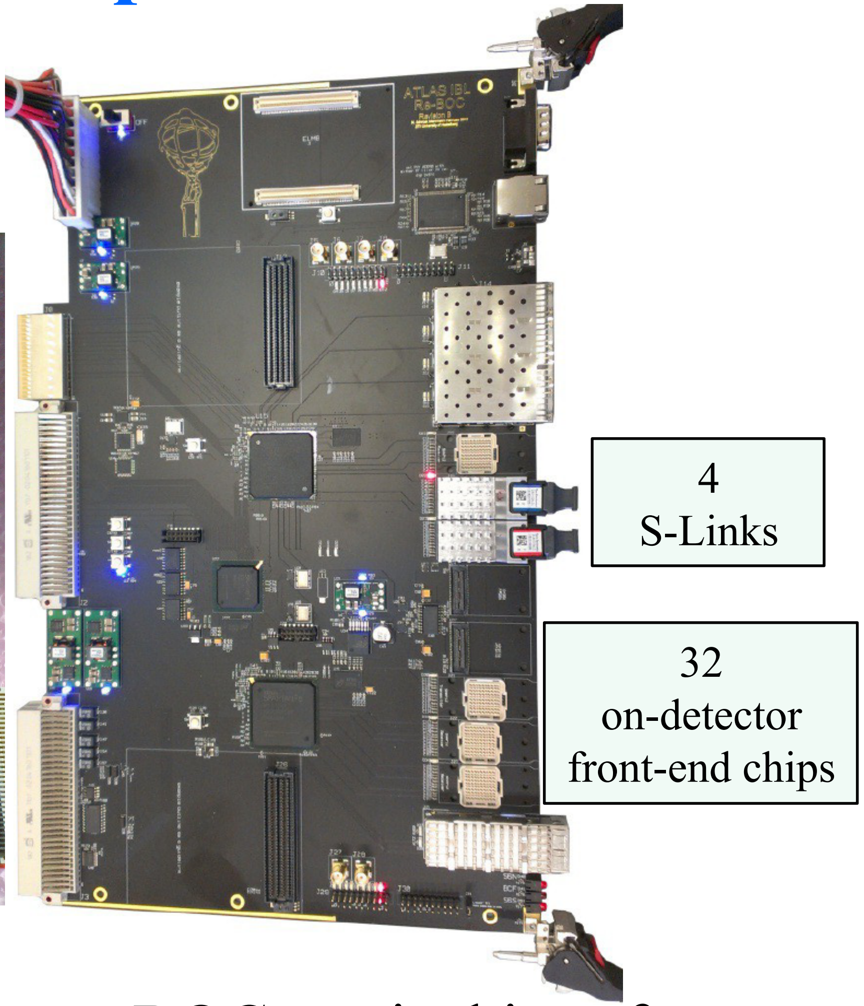
Power supply distribution



New BOC-ROD pair



ROD: data processing



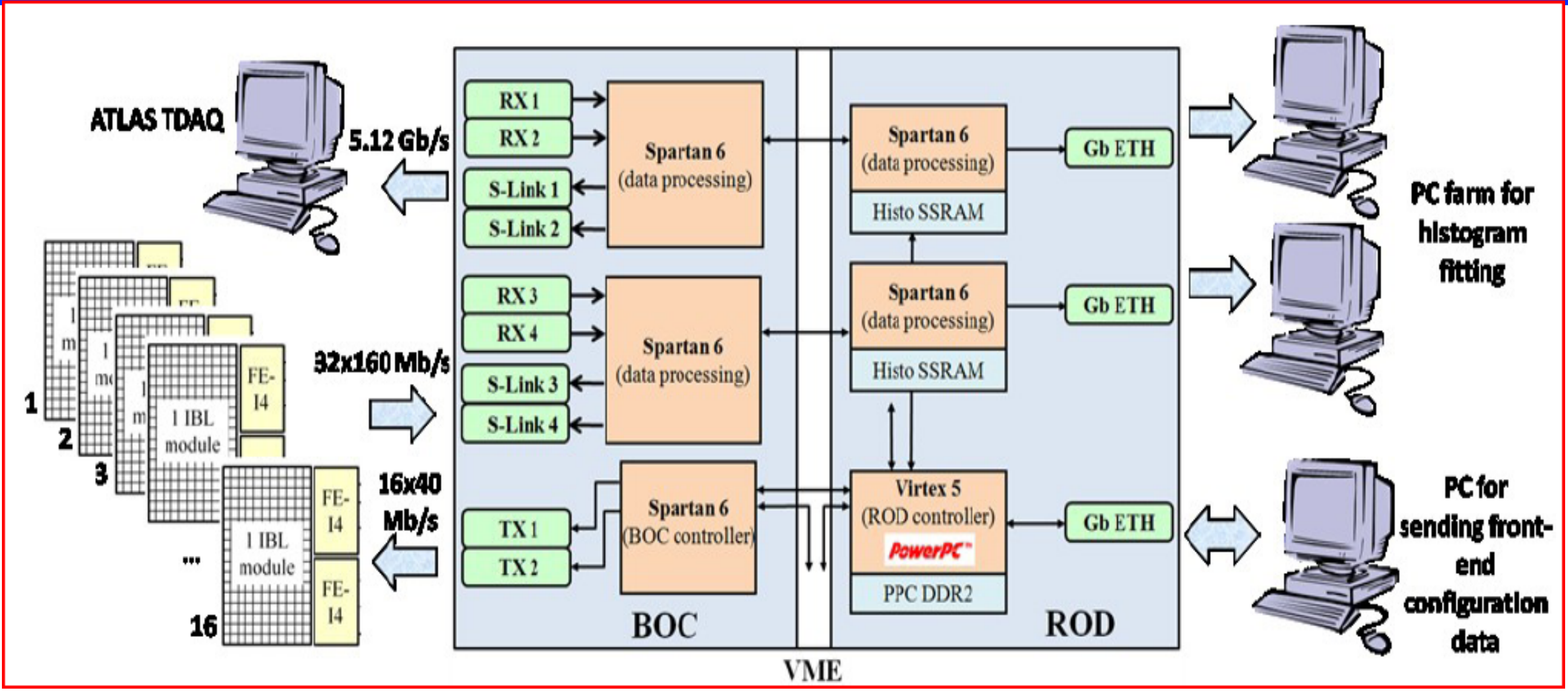
BOC: optical interface

Total I/O bandwidth: 5.12 Gb/s

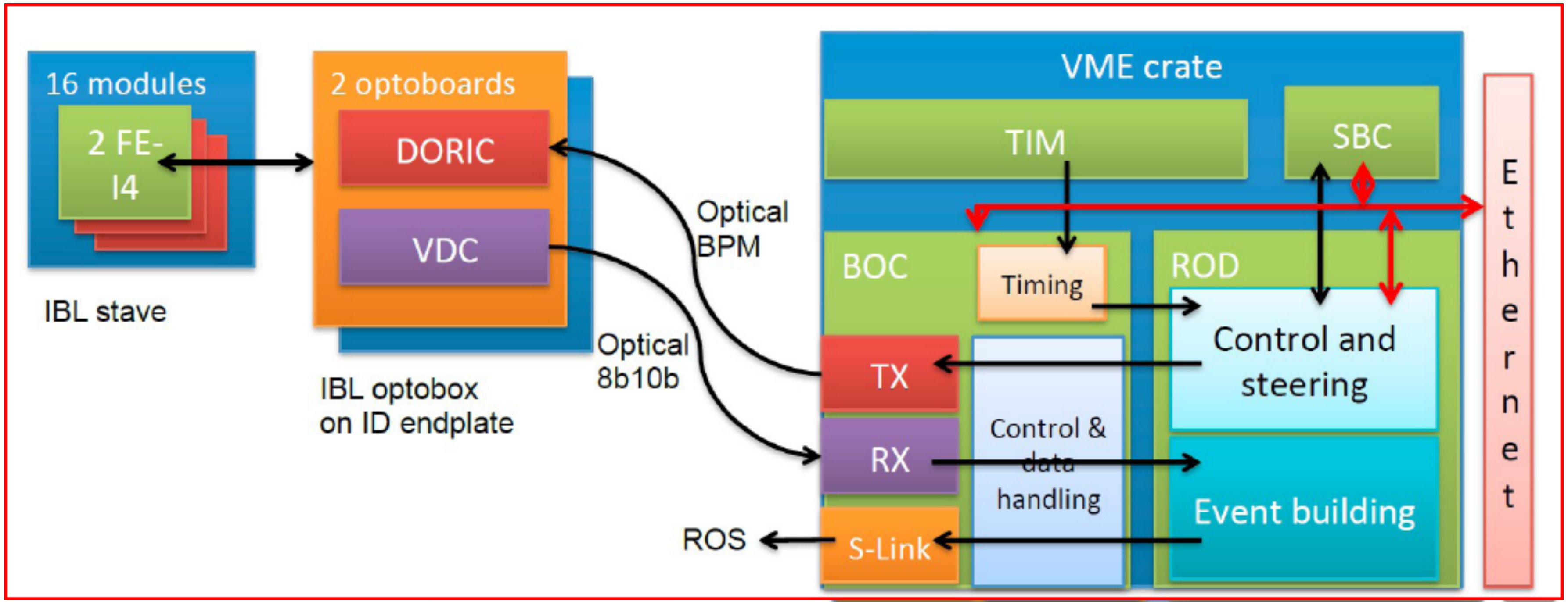
Faster calibration link by using Gb Ethernet instead of the VME bus for data transfer

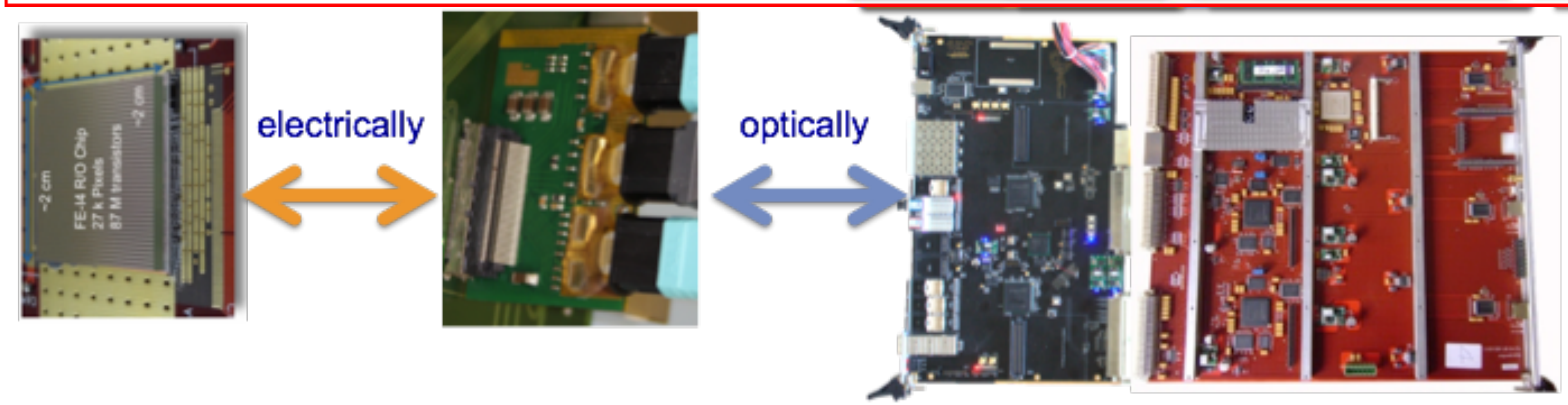
IBL ReadOut system summary

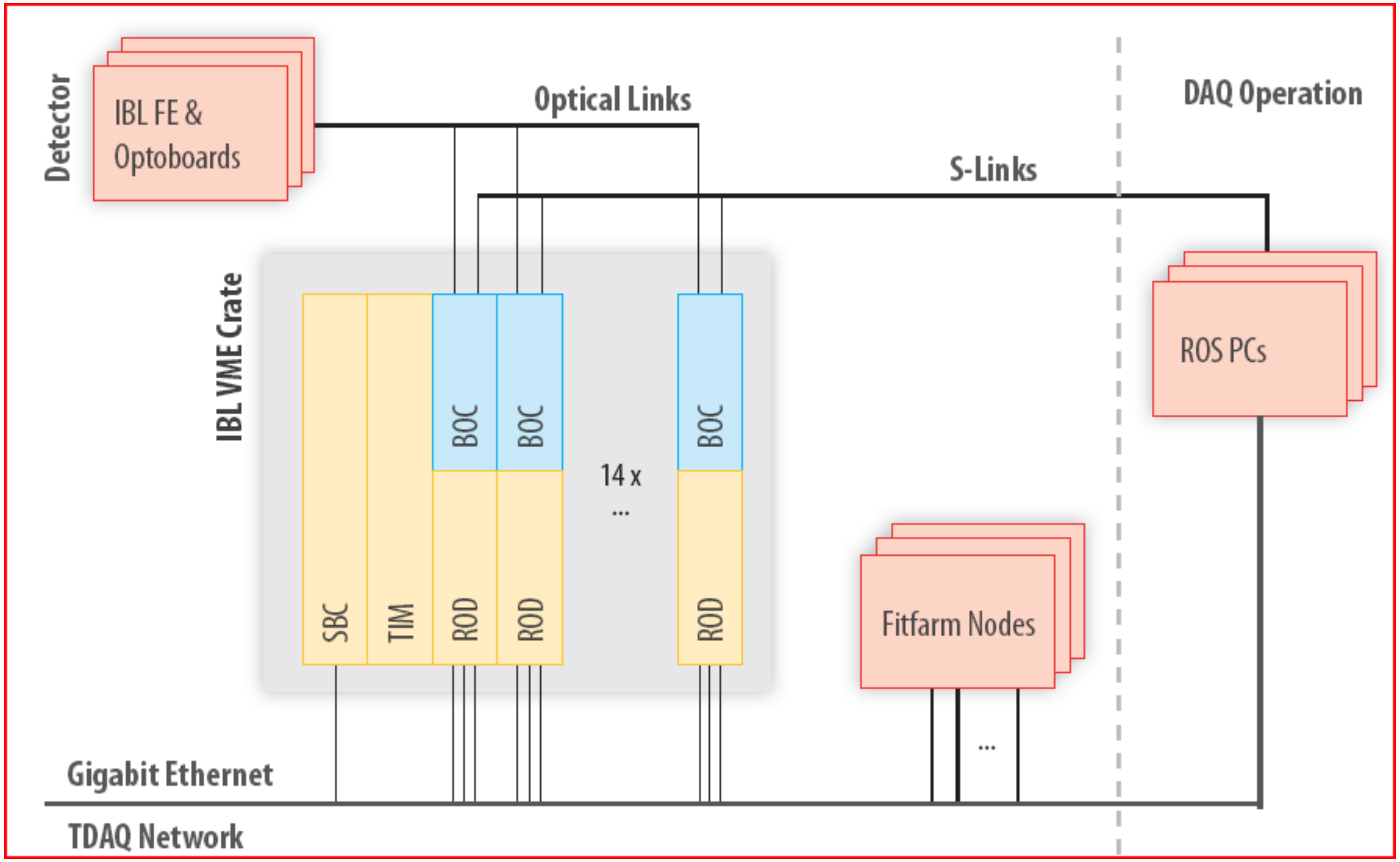
Number of IBL Staves /ROD-BOC pair	14
# DAQ Modules per ROD-BOC pair	16
# FE-I4s chip per ROD-BOC pair	32
Total # of FE-I4s in IBL	448 (32*14)
Number of Pixels per FE-I4	26880
Total # of read-out channels	~12 M


ATLAS TDAQ
5.12 Gb/s
32x160 Mb/s
16x40 Mb/s
1 IBL module
FE-I4
1
2
3
...
16
RX 1
RX 2
S-Link 1
S-Link 2
Spartan 6 (data processing)
RX 3
RX 4
S-Link 3
S-Link 4
Spartan 6 (data processing)
TX 1
TX 2
Spartan 6 (BOC controller)
BOC
VME
Spartan 6 (data processing)
Histo SSRAM
Spartan 6 (data processing)
Histo SSRAM
Virtex 5 (ROD controller)
PowerPC
PPC DDR2
ROD
Gb ETH
Gb ETH
Gb ETH
PC farm for histogram fitting
PC for sending front-end configuration data

ATLAS IBL Readout Structure

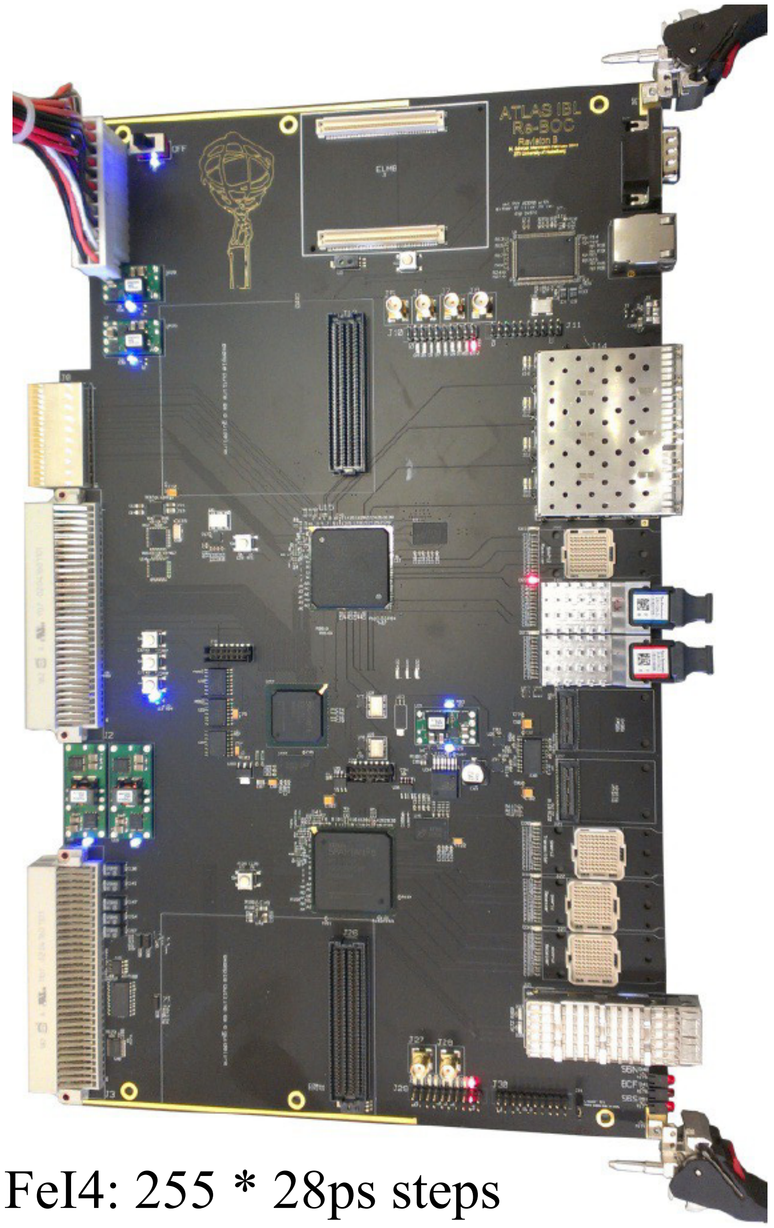





Detector
IBL FE & Optoboards
Optical Links
DAQ Operation
S-Links
IBL VME Crate
SBC
TIM
BOC
ROD
BOC
ROD
14 x
...
BOC
ROD
ROS PCs
Fitfarm Nodes
...
Gigabit Ethernet
TDAQ Network

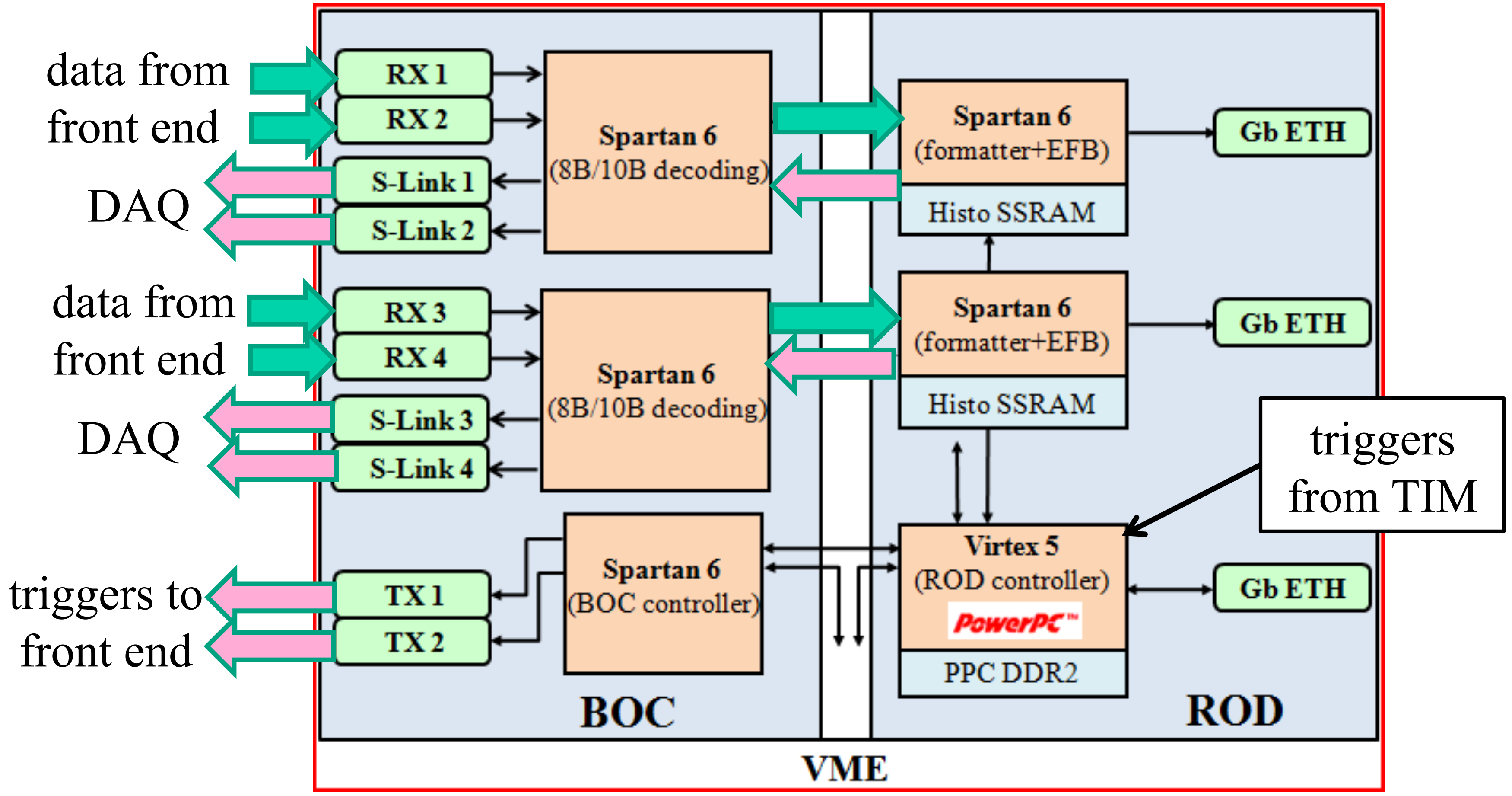
IBL Back Of Crate card

- Timing interface
  - Receive clock from TIM
  - Distribute the clock to detector components and ROD
- Optical interface to/from detector and readout buffers
  - SNAP12 plugins (Tx / Rx)
  - S-Link plugins (QSFP)
- Data en-/decoding for detector communication
  - BPM encoding towards the detector
  - 8b/10b decoding for the detector data to hand over to the ROD
- Monitoring functionalities for detector data

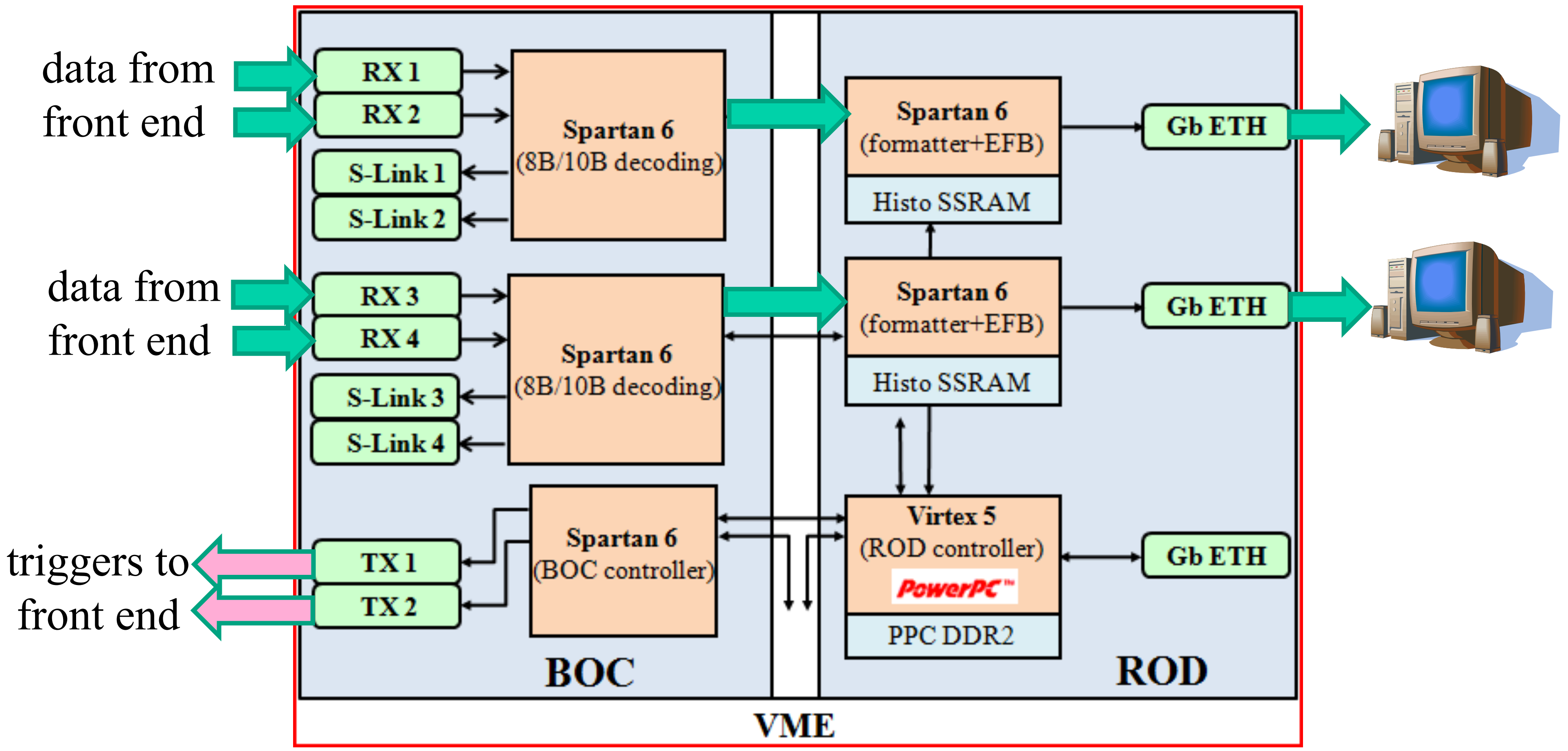


Sampling @ 640 MHz with ISERDES,
GTPs to support 4-Slinks @ 2 Gb/s
Individual output delay on serial lines to FeI4: 255 * 28ps steps

Data flow during physics runs



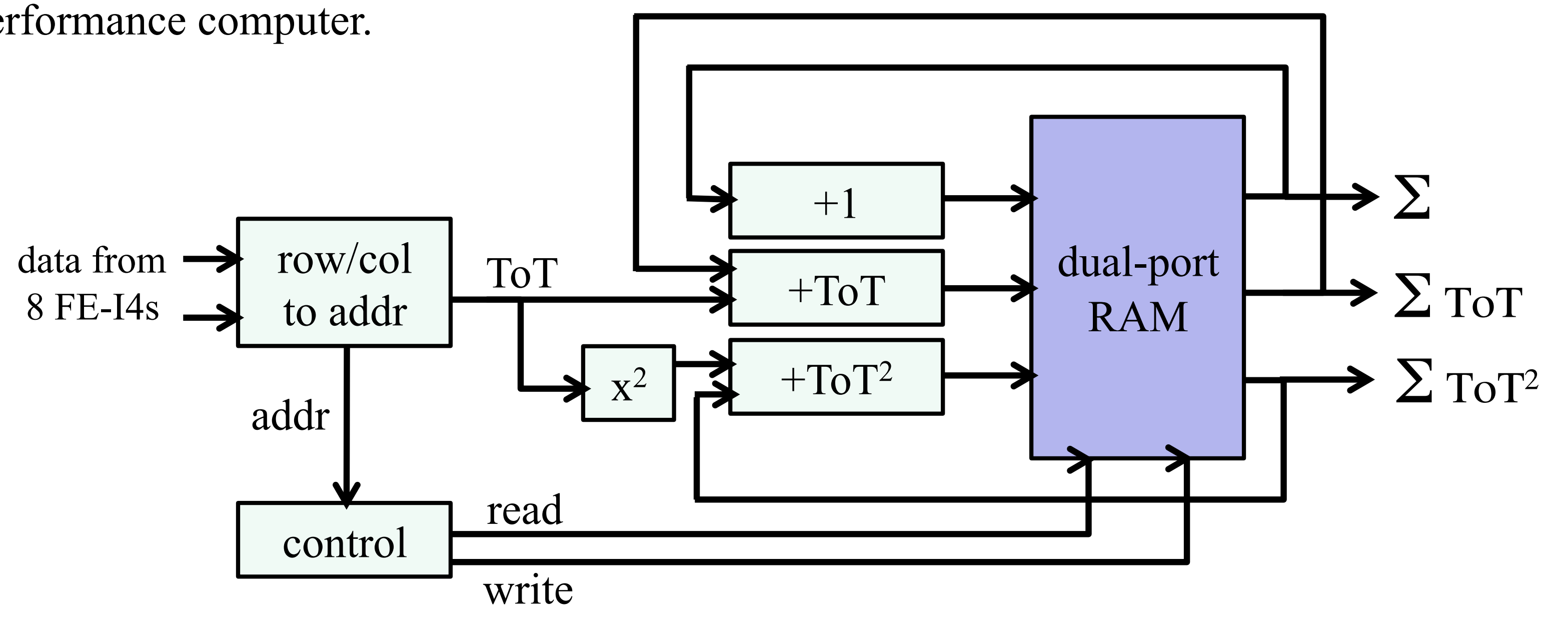
Data flow during calibration runs

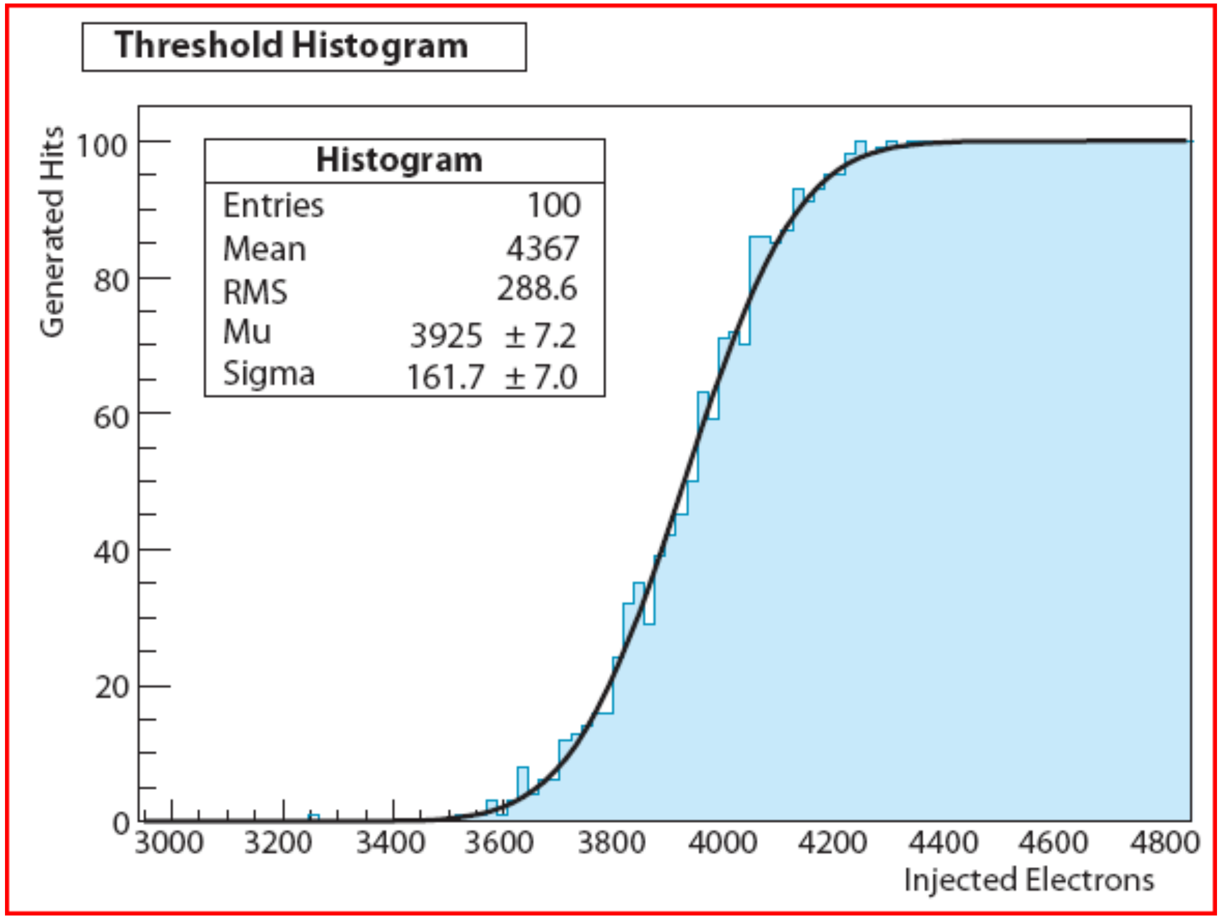


The calibration is controlled by the PPC which drives the calibration scan configuring the FeI4s and sending triggers

Calibration runs

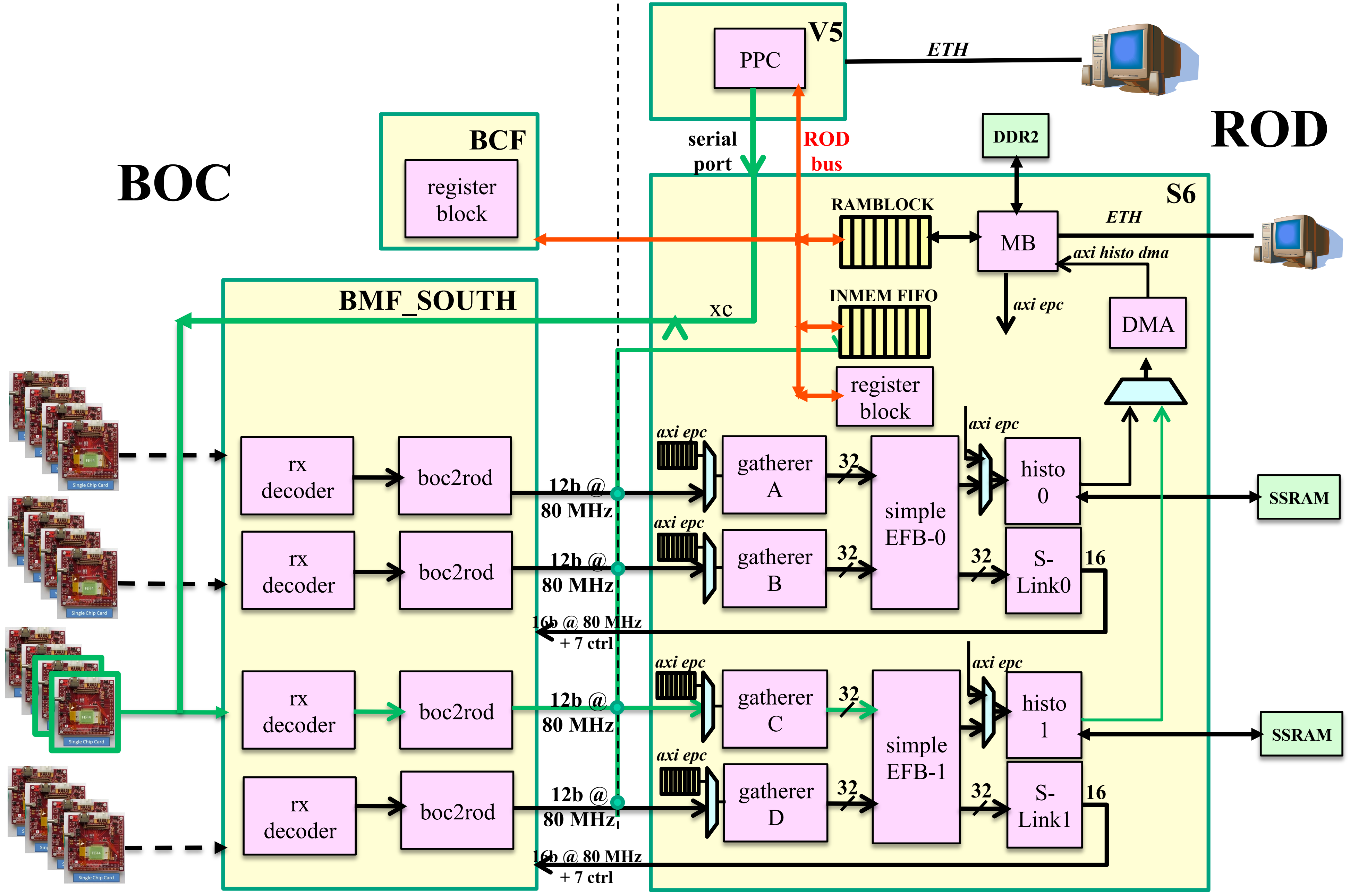
As for the Pixel Detector, the calibration of the IBL detector is performed by repeating relatively short (~100) series of events, recorded while injecting a known charge into each pixel and with different settings of the front-end parameters (e.g. different thresholds or different pre-amplifier feedback currents). This sequence, called **calibration scan**, generates a very large number of events, which have to be analyzed in order to extract the histogram showing how many times each pixel has been fired for a given setting. For this reason, in calibration mode, the data stream coming from the sensor is not sent over the S-Link, but pre-processed in the ROD, where the relevant histograms are produced. At the end of the scan, the histograms are transferred to an external farm via Gb Ethernet for fitting and archiving.
IBL ROD executes only the calibration loops to accumulate the per-pixel occupancies, sums of time-over-threshold (ToT) and sums of ToT^2 parameters. Then histograms are created and saved on-the-fly on RAMs and eventually transferred via Gb Ethernet to an off-line high-performance computer.





A threshold scan determines the threshold μ and noise of a pixel. This is achieved by injecting a defined charge Q multiple times into the pixel and counting the resulting hits. One then iterates over several charge steps (typically 100) by changing the voltage Vcal and creates a histogram

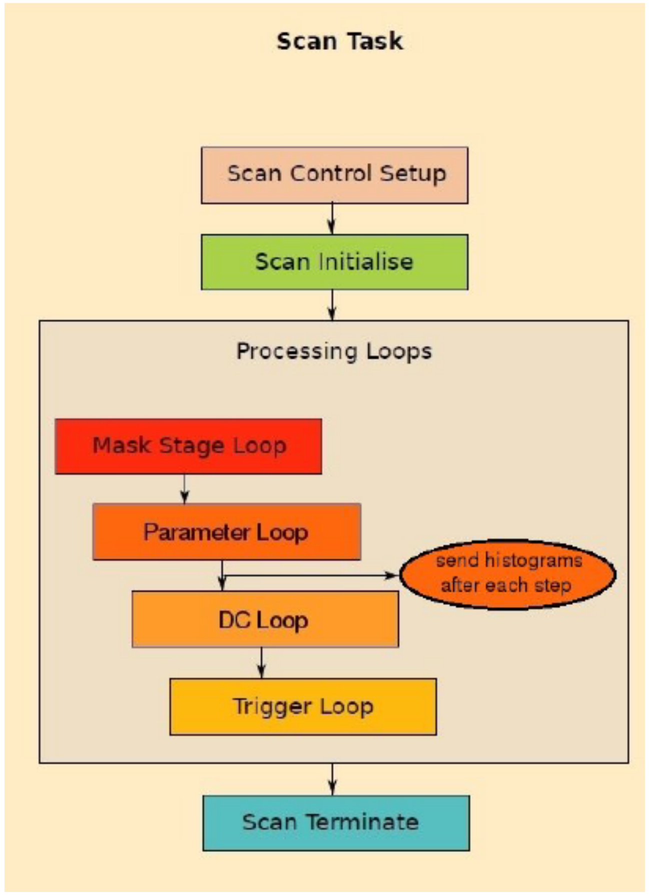
A very quick look to the firmware



A lot of embedded processors

- The **PPC** on the Virtex5 is good for implementing repetitive scan loops (= much easier to code in SW than in VHDL)

- Each of the 2 Spartan6 implements a **Microblaze** core, mainly for dealing with the TCP/IP histograms transfer to the external farm (a custom VHDL core is much more difficult to implement)



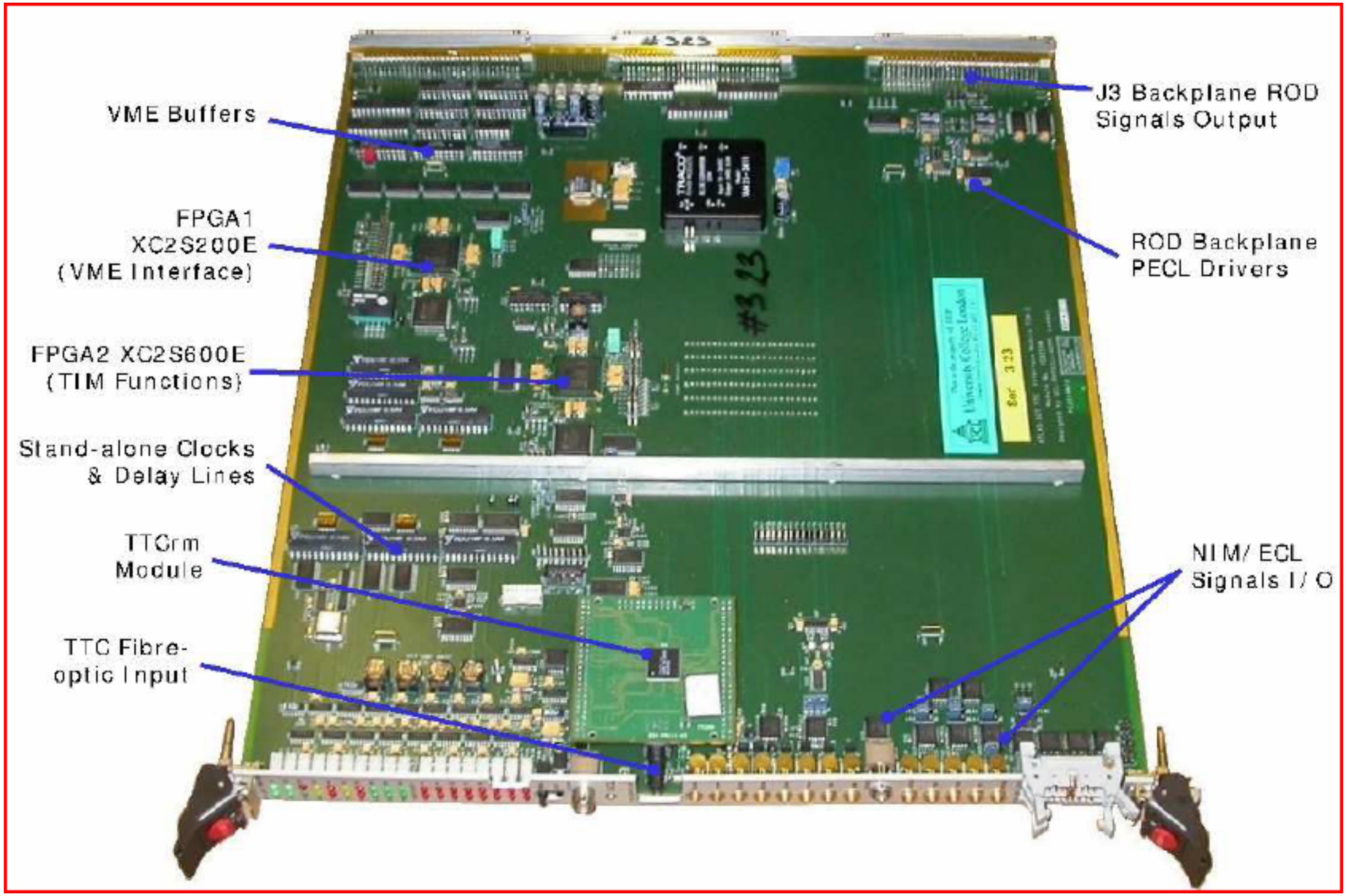
A nice thing of embedded processors is that their behavior can be simulated together with the remaining VHDL blocks in order to find bugs. It was not the same when using TI DSPs and FPGAs …

IBL ROD Summary

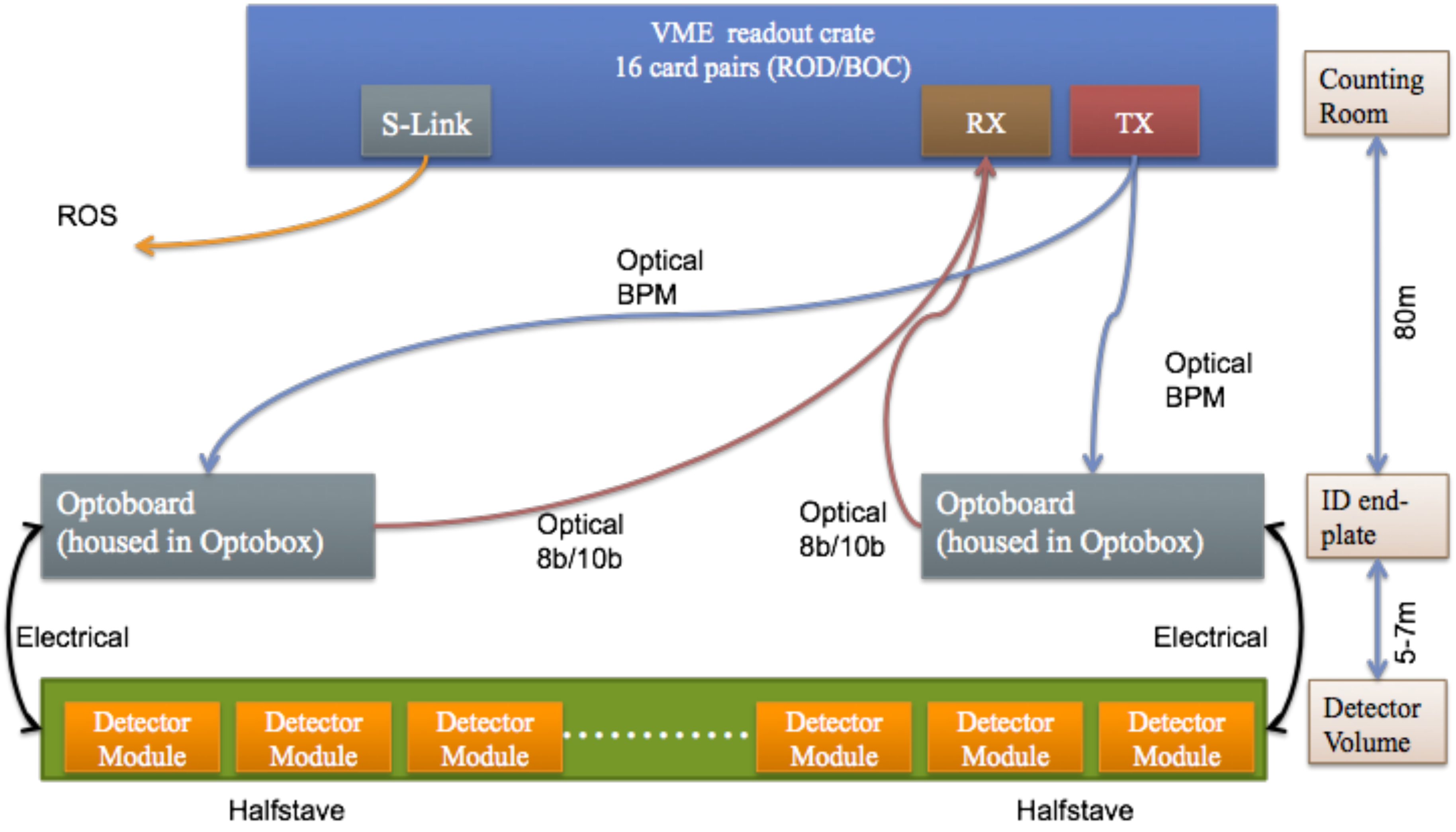
- A new HW board is in place
- A lot of new firmware features can be implemented and tested
- A lot of SW also has to be developed

Backup slides

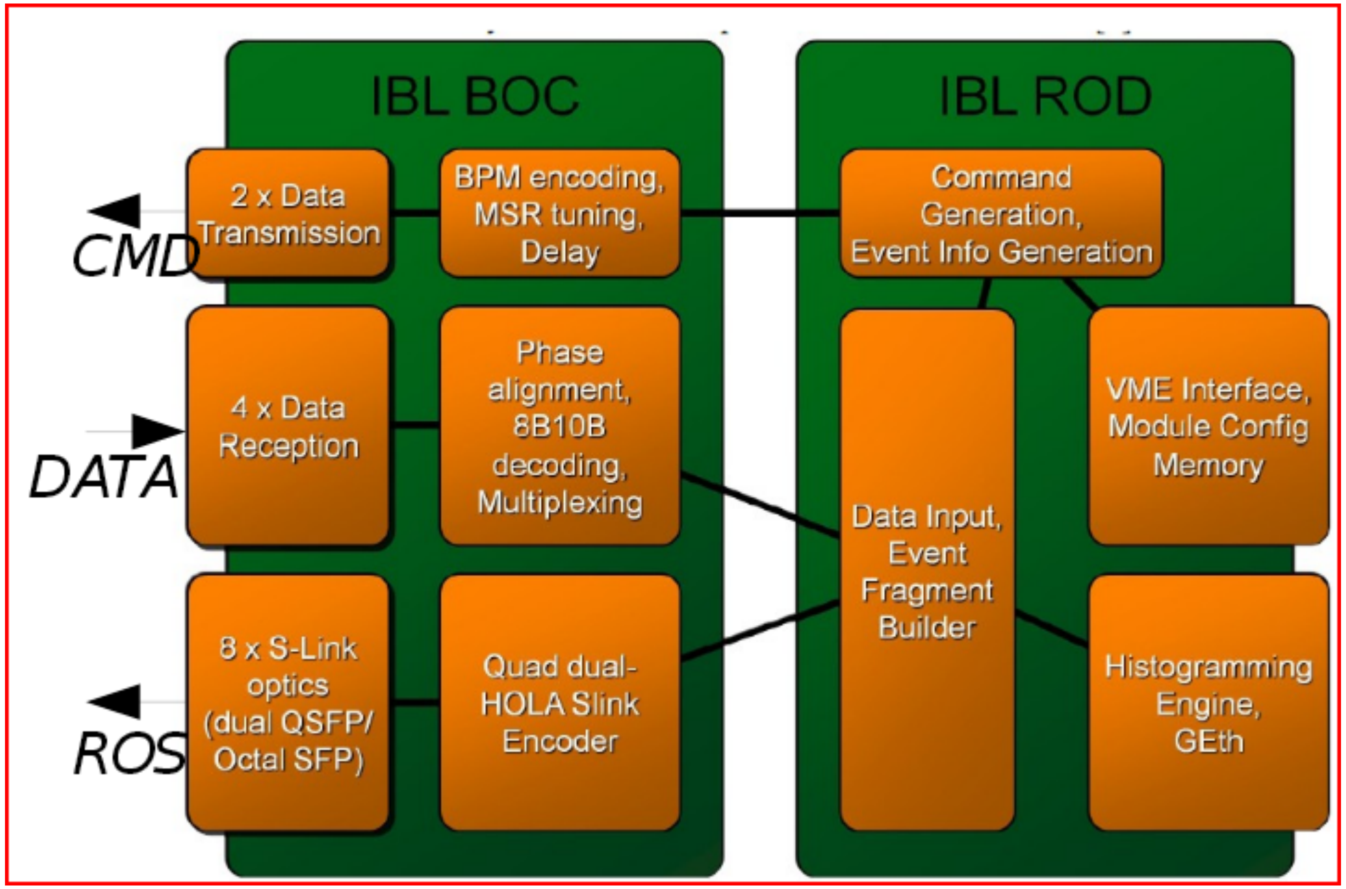
TIM module


VME Buffers
FPGA1 XC2S200E (VME Interface)
FPGA2 XC2S600E (TIM Functions)
Stand-alone Clocks & Delay Lines
TTCrm Module
TTC Fibre-optic Input
J3 Backplane ROD Signals Output
ROD Backplane PECL Drivers
NIM/ ECL Signals I/ O
#323

Readout Scheme Overview



IBL BOC-ROD


IBL BOC
IBL ROD
CMD
DATA
ROS
2 x Data Transmission
BPM encoding, MSR tuning, Delay
Command Generation, Event Info Generation
4 x Data Reception
Phase alignment, 8B10B decoding, Multiplexing
Data Input, Event Fragment Builder
VME Interface, Module Config Memory
8 x S-Link optics (dual QSFP/ Octal SFP)
Quad dual-HOLA Slink Encoder
Histogramming Engine, GEth