The SoC-based Readout Drivers of the ATLAS Pixel/IBL detector

Oldřich Kepka
Institute of Physics, Prague

On behalf of ATLAS Pixel group

13/06/2019, SoC workshop CERN

FZU Institute of Physics of the Czech Academy of Sciences

ATLAS Pixel Readout System

4-Layer Pixel Detector, Innermost Layer (IBL) installed in 2015

Off-detector electronics

- **9U VME Cards types: Readout Drivers (ROD), Back of Crate (BOC)**
- Legacy SiROD and SiBOC board replaced between 2015-2017
  - Limited bandwidth (1.04 Gbps at the SLink)
  - Calibration: limited VME bus speed, no parallelism
  - Limited control and recovery mechanisms via VME

- New IBL-BOC and IBL-ROD developed for IBL
  - Now used for the entire Pixel system (117 BOC/ROD cards)
  - 4x higher bandwidth needed for IBL closest to the beam
  - **Communication via gigabit ethernet**
  - **SoC PowerPC embedded microprocessor to facilitate control**

Readout electronics

ROD

1x Virtex-5 Master FPGA with PowerPC 440 (C5VFX70T-FF1136)
3x Spartan-6 FPGA**:**

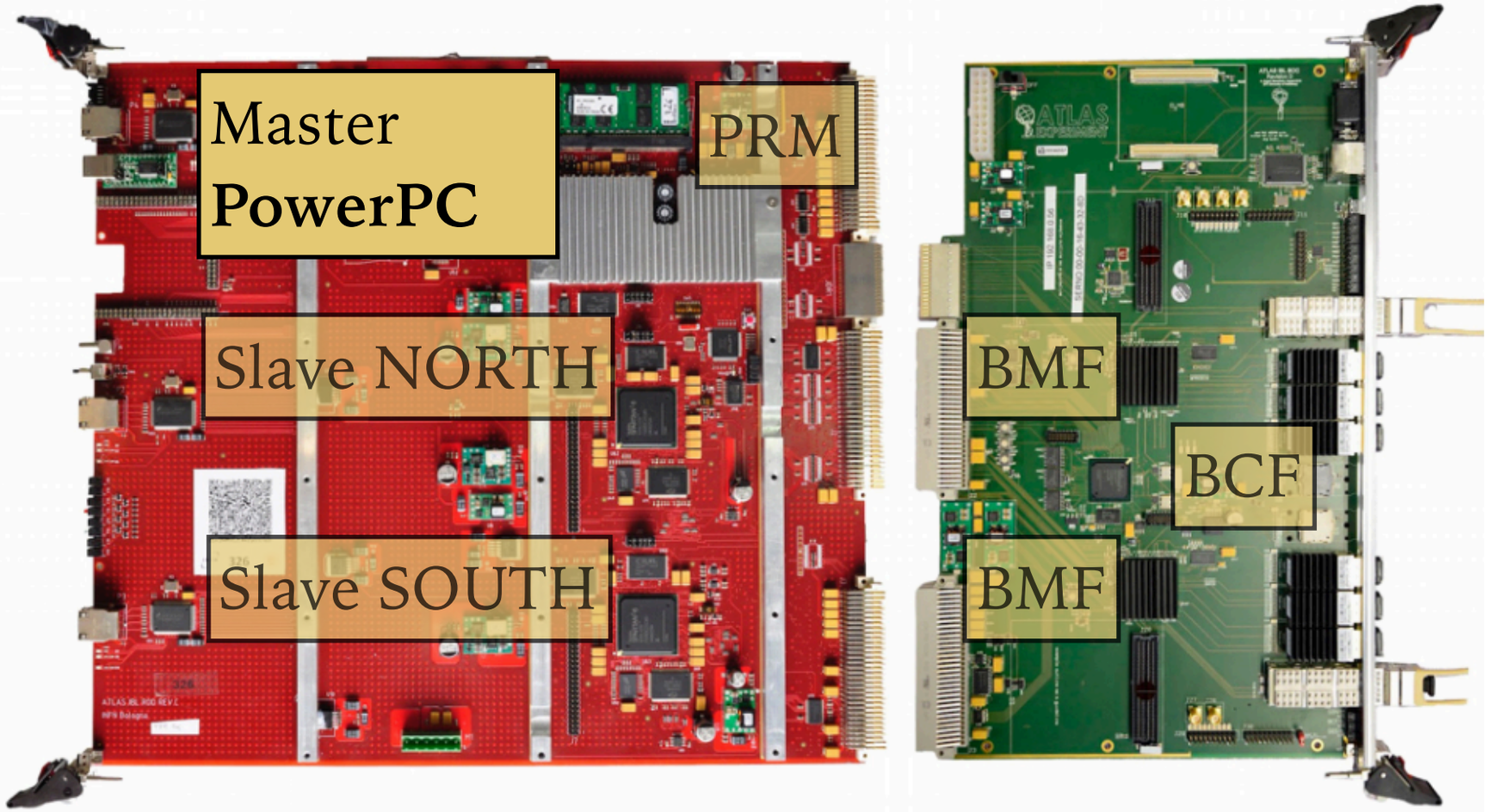
- **Program-Reset Manager (PRM)** - FPGA and Flash memory programming via VME (XC6SLX45FGG484)
- **2x Slaves** (XC6SLX150FGG900) - data processing

BOC

3x Spartan-6 FPGA:

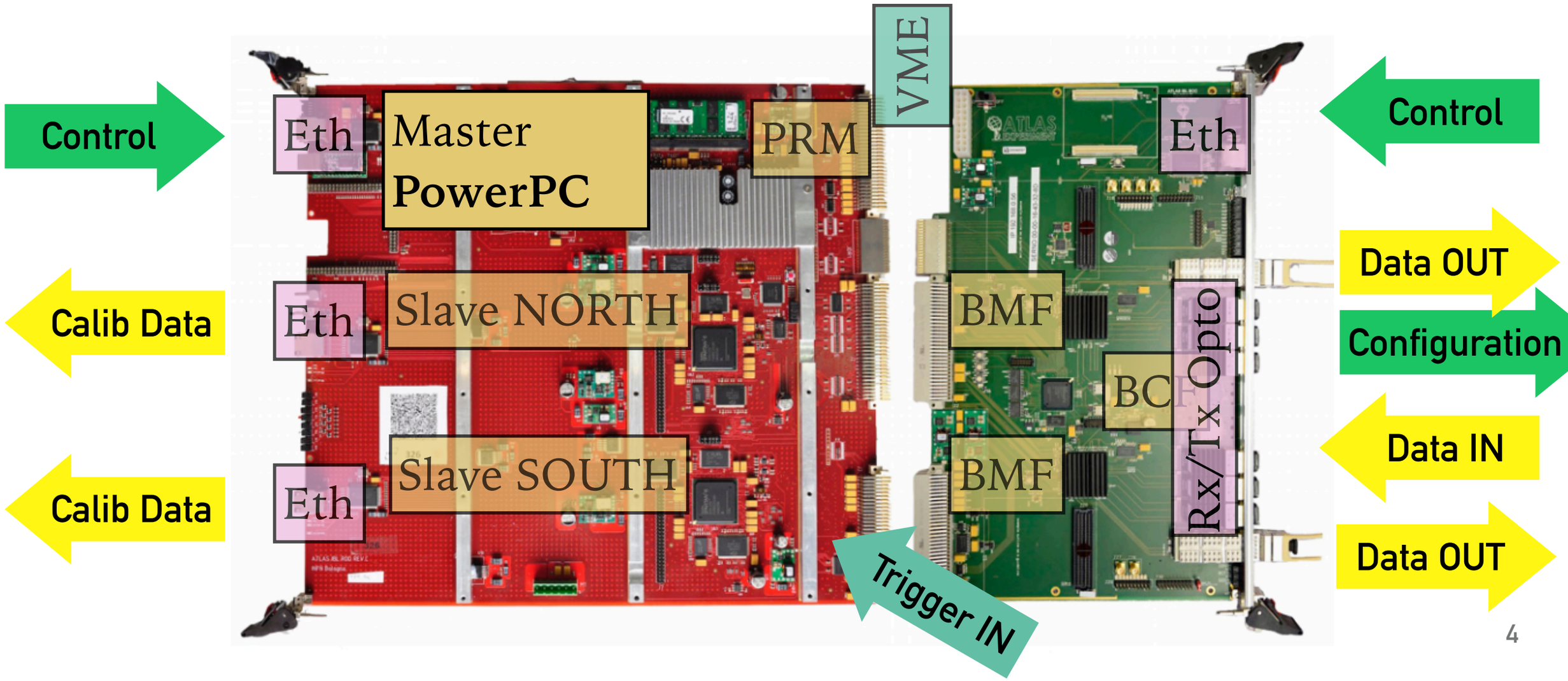
BCF: Boc Controller FPGA - low level control

2x BMF: Boc Main FPGA used for bitstream decoding/encoding



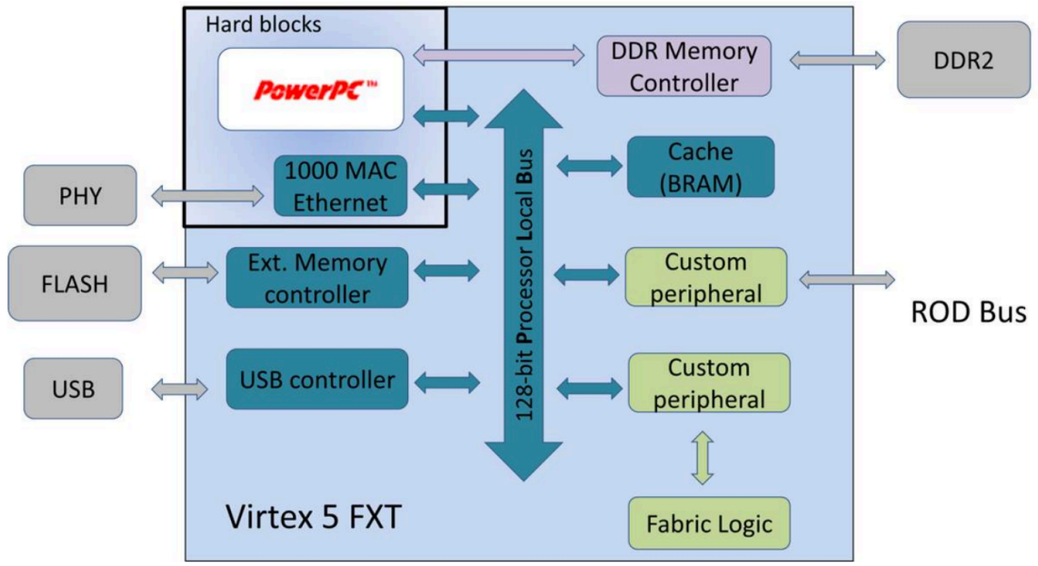
Functionality

- ROD
  - issues commands to be sent to front ends
  - fragment building from incoming data
  - propagate trigger from TTC stream
  - **SoC in the master FPGA controls the readout, performs reconfiguration**
- BOC
  - handling control interfaces to detector
  - processing incoming data sent via optical fibre
  - provides clock to the detector



ROD details

- Older generation Xilinx FPGA also had SoC
- SoC included in Vertex 2 to 5 series with PPC405 to PPC440 32 bit processors
  - Processor removed from Virtex 6 onwards
- 550 MHz CPU @ 2GB RAM
- 8 MB Flash memory
  - Upload Master/Slave SW via VME using PRM



- Tri-Mode Ethernet Media Access Controller (TEMAC) ASIC chip with 1Gbit ethernet
  - Limit usage of VME in favour of network communication

- Development tools:
  - JTAG chain to access for ROD FPGA
    - Accessed via USB (not scalable, very few JTAG programmes in the PIT) or XVC (Ethernet)
  - UART

Legacy Run 2 Master SW

- SW based on xilkernel v5.01a, compiled using Xilinx Software Development Kit
  - Limited debugging capability
- Ethernet communication using LWIP networking stack
  - Lightweight implementation of networking, ICMP/UDP/TCP
  - Small memory requirements, memory shared between different networking layers
  - Used v1.41 and v2.02 official ports to xilkernel by Xilinx [link]

- **Main limitation for Run2 operation was stability**
  - Rare, but very intrusive. Limiting number of reconfiguration actions that could be taken

- ROD stops accepting TCP/IP connection
  - running out of resources, connection sockets are not closed
  - rod continues to reply to ping and UDP
- ROD SW random crashes
  - narrowed down to LWIP memory allocation function for incoming packets on interrupt calls from TEMAC driver
  - reproduced by sending parallel pings to the system
- Network activity **AND** CPU load needed to reproduce the problem

Linux on IBL ROD

- SW based on the official **Xillinx linux kernel** [linux-xlnx]
  - with support for PowerPC 440 and TEMAC (drivers: Generic Xilinx Virtex 5 FXT board support, Xilinx LL TEMAC)
  - Tested kernel version 4.14; foreseen updates kernel versions to comply with security

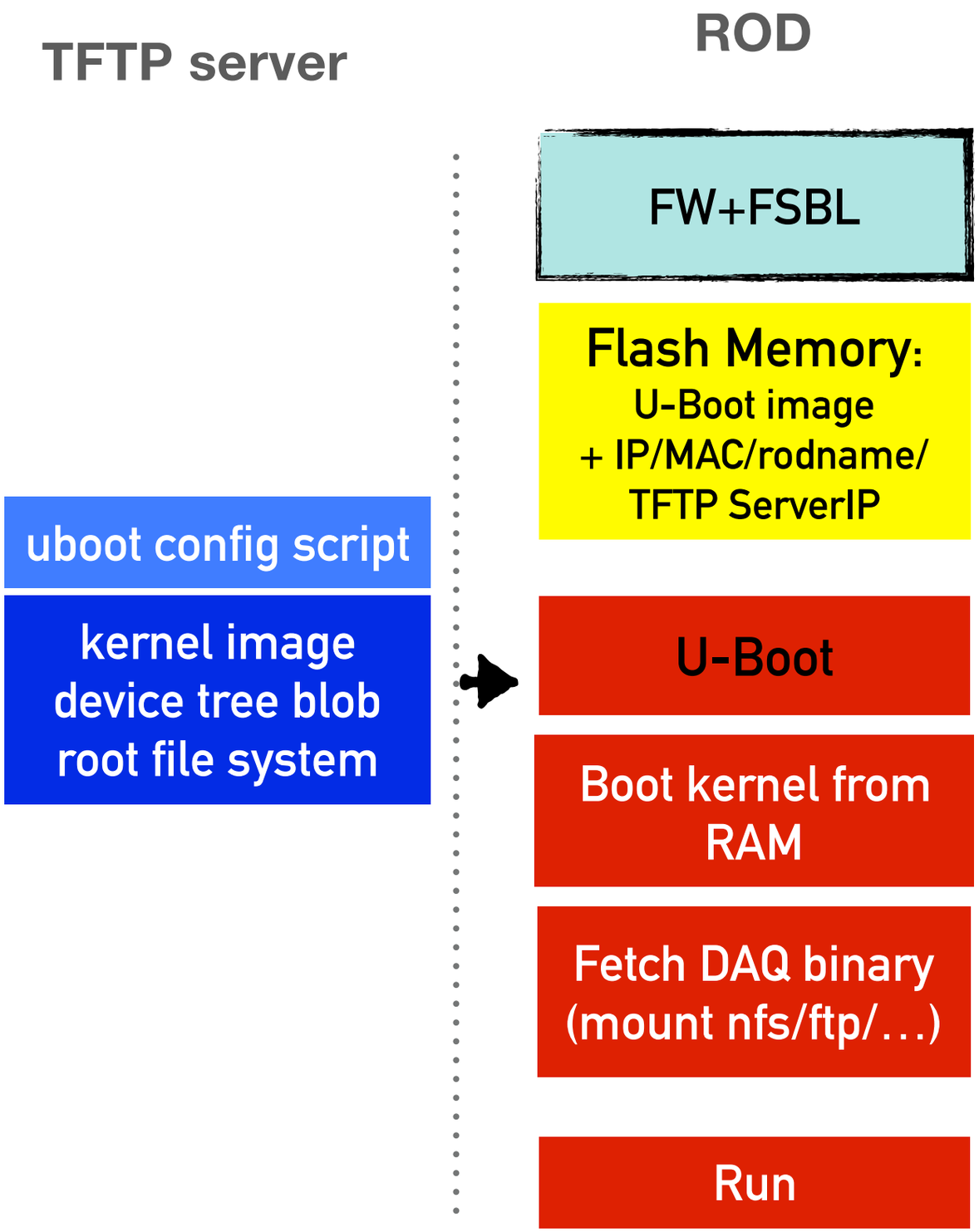
- Cross compiled with **crosstool-ng**

- Using **U-Boot** manager provided by Xilinx [u-boot-xlnx]
  - ROD flash memory limited (8MB), only small u-boot imaged loaded to flash
  - Linux image, device tree blob and root file system loaded via tftp
  - Support for PPC stopped in version u-boot v2017.4

- Buildroot to build **root FS**
  - various applications: standard linux commands (BusyBox), ssh, gdb, iperf, nfs, …
  - Fallback version of Pixel DAQ code

- **Custom** but **fully automated** build system of all needed components

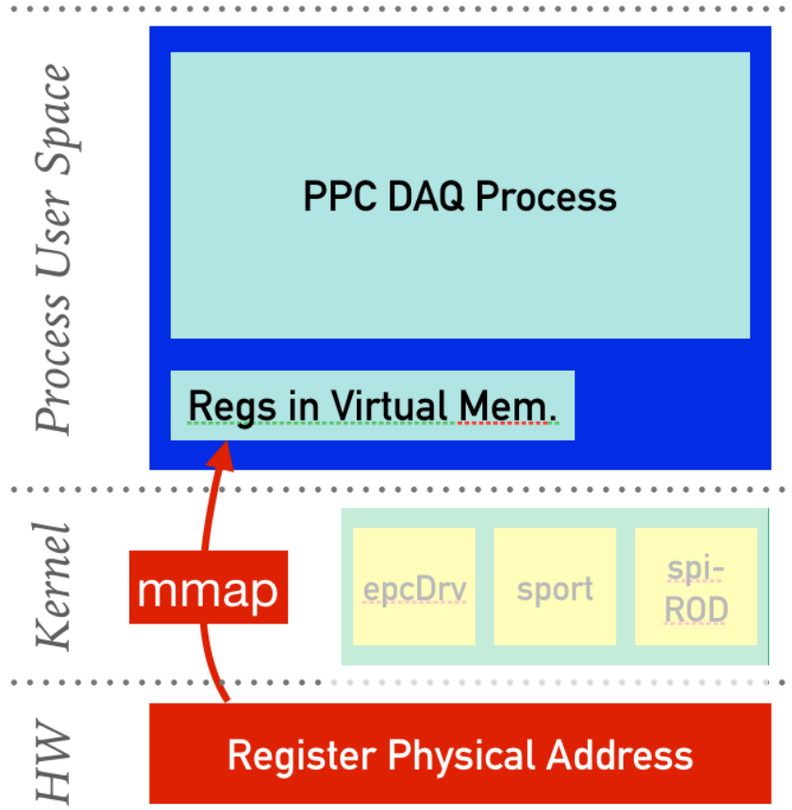
Booting scheme



- ROD network configuration stored in Flash memory together with the U-Boot image
- FSBL copies U-Boot to RAM and runs it
- TFTP server contains kernel/ dtb/rootfs
  - Quick to apply kernel updates
- Pixel DAQ code fetched
  - from outside
  - or fallback Pixel DAQ binary in the rootfs

Pixel DAQ PPC SW

- Single process Run Control application
  - RPC calls from HOST to PPC using custom TCP/IP server
  - Provides monitoring information and command execution
  - **Boils down to reading/writing certain ROD and BOC registers provided by the ROD FW**
- Registers are memory-mapped to the user space
  - Using /dev/mem device



- FW registers exported to c++ include files to keep FW and SW synchronised
- PPC SW is a standalone package cross-compiled using cmake, no dependency on TDAQ
- Continuous integration for SW and FW, gitlab-runners running on Lab machines, registered to gitlab
- Improvements of the PPC DAQ being investigated
  - gcc8 allows to profit from modern tools: ZeroMQ, MsgPack, …

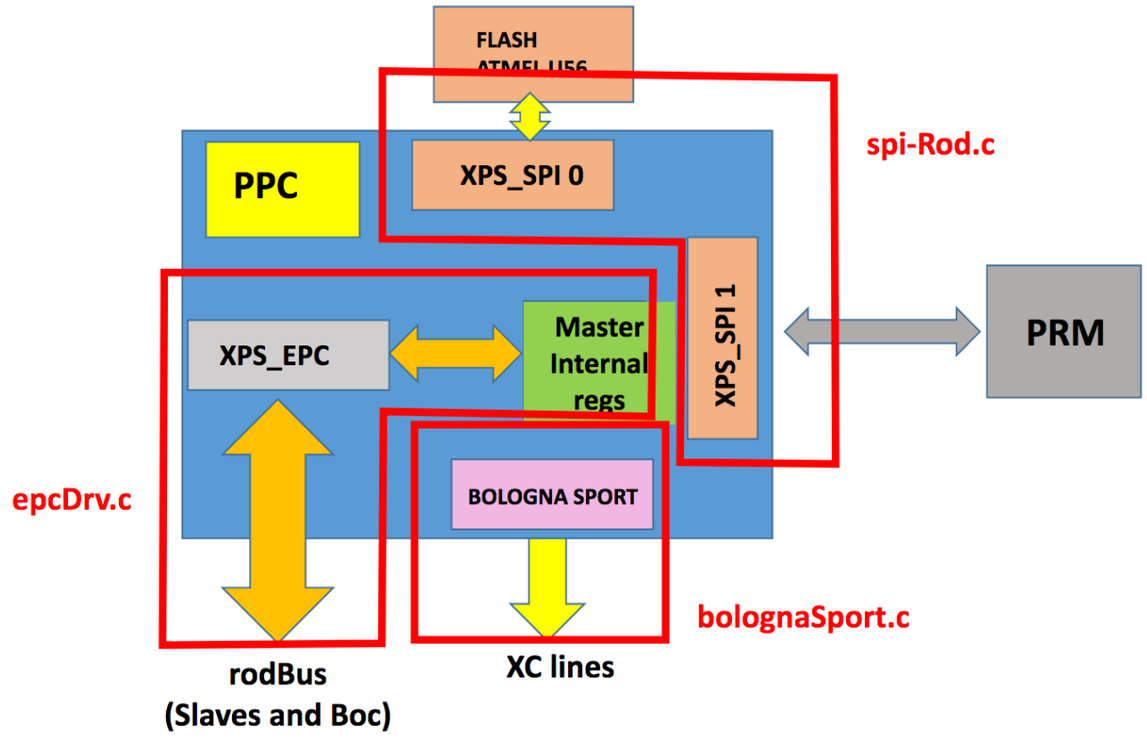
Future plans for FW & SW

- Resources limited on Slave FPGA programable logic
  - Limitation for adding new functionalities and for debugging capabilities
- Developing new mechanism allowing to change Slave FW ‘on-the-fly’
  - Allows different FW for calibration and data taking which are orthogonal
- Fast programming of the Slaves via ethernet using SoC on the master
  - Direct access from PPC to PRM to program slave PROMS
- Dedicated char Linux drivers developed

epcDrv
- EPC controller for ALL registers of MASTER, SLAVEs and BOC

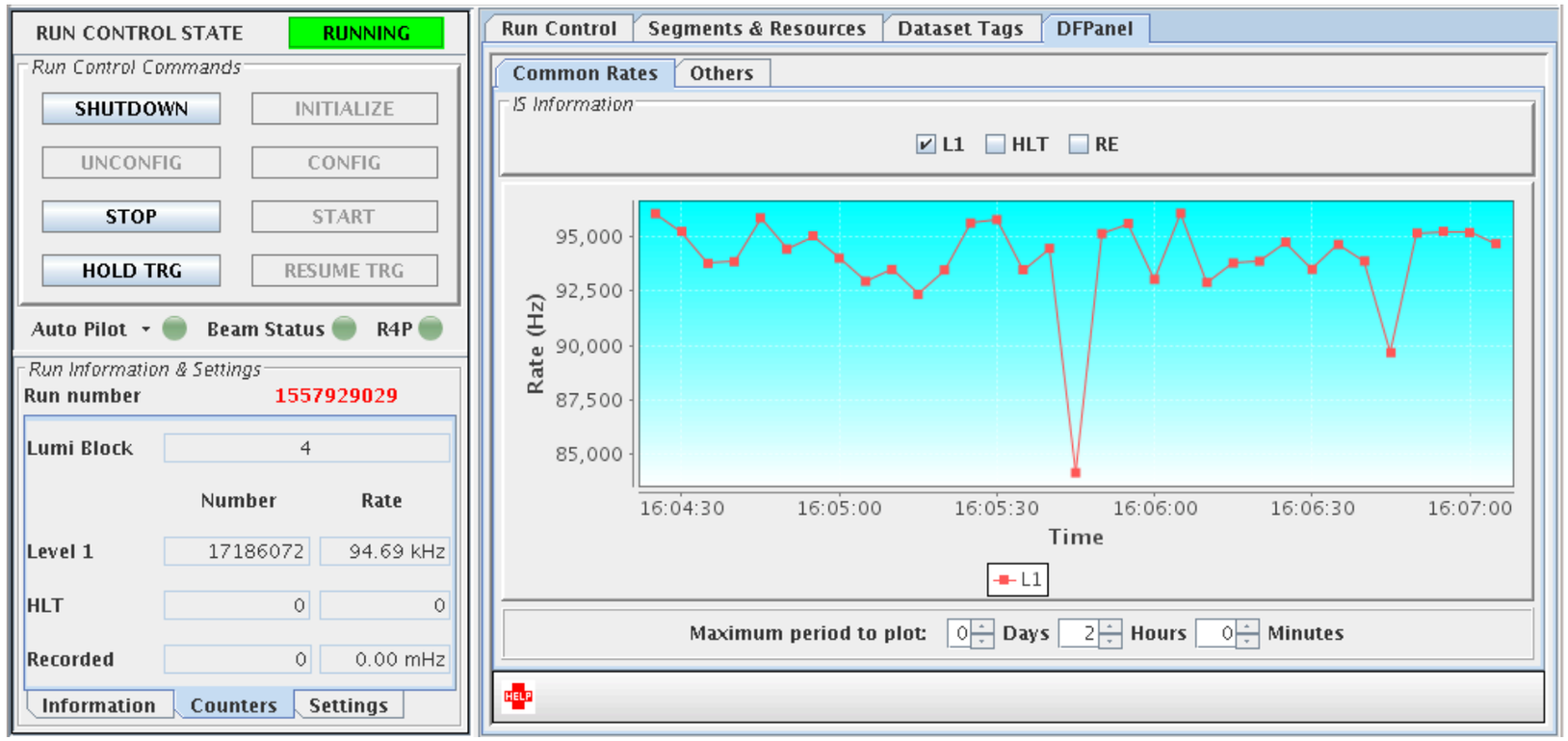
spi-Rod
- SPI dev1 (bus controller for PRM regs)
- SPI dev0 (bus controller for FLASH)

Bologna S-PORT
- for serial commands from PPC



Tests of Pixel DAQ SW

- Existing Pixel DAQ software migrated from XilKernel to Linux
- Successfully tested in the Lab with real detector
  - Reading/writing FPGA registers
  - Sending configuration to real modules
  - Calibration scans
  - Real data taking
- All critical DAQ functionalities working on Linux!



Next steps

- Working on scaling up the system; hoping to deploy in ATLAS by the end of 2018

- ATLAS is on a technical control network (ATCN)
  - Our system is not CERN certified device
  - Ongoing discussion: plan to keep RODs in the private network
  - access via gateway computers

- TFTP server setup needed on gateway machines
  - Accessible from both ATCN and private network
  - A standardised TFTP configuration from IT will be appreciated!

Summary

- 117 IBL RODs will be used until end of Run3 with SoC
- Stability limitations in Run2 related to networking & CPU load
- Transitioned embedded software to Linux
- Successful operation tests in the Lab
- Plan to deploy in ATLAS this year
- More improvements in the FW and SW being worked on during LS2

Backup

fw: RC1 *fw: sp6fmt*

BOC **ROD**