The ATLAS Muon-to-Central Trigger Processor Interface (MUCTPI) Upgrade

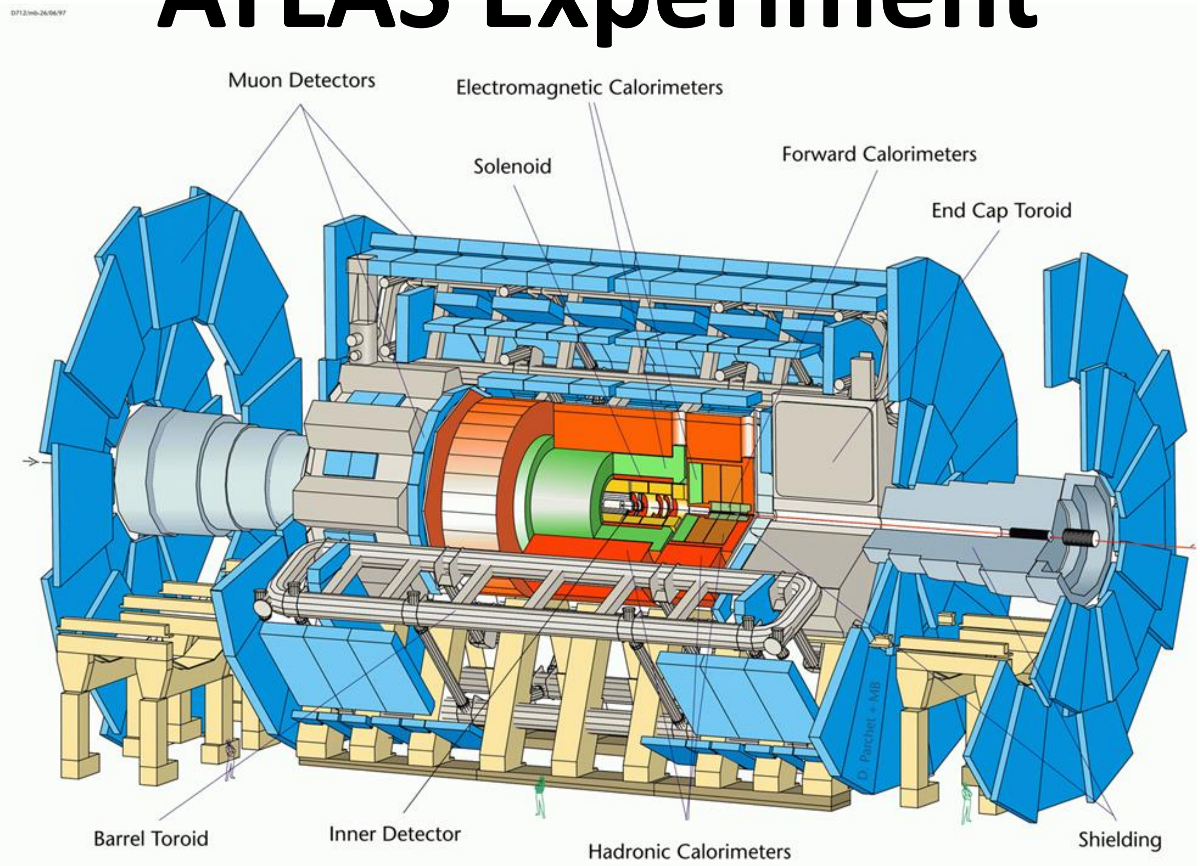
- Introduction
- MUCTPI Upgrade
- MUCTPI Upgrade and Run Control
- Summary

On behalf of:

A. Armbruster[a], G. Carrillo-Montoya[a], M. Chelstowska[a], P. Czodrowski[a], P.-O. Deviveiros[a], T. Eifert[a], N. Ellis[a], G. Galster,[a,b] S. Haas[a], L. Helary[a], O. Lagkas Nikolos[a], A. Marzin[a], T. Pauly[a], V. Ryjov[a], K. Schmieden[a], M. Silva Oliveira[a], R. Spiwoks[a], J. Stelzer[a], P. Vichoudis[a], T. Wengler[a]

a) CERN, Switzerland, b) NBI Copenhagen, Denmark

ATLAS Experiment

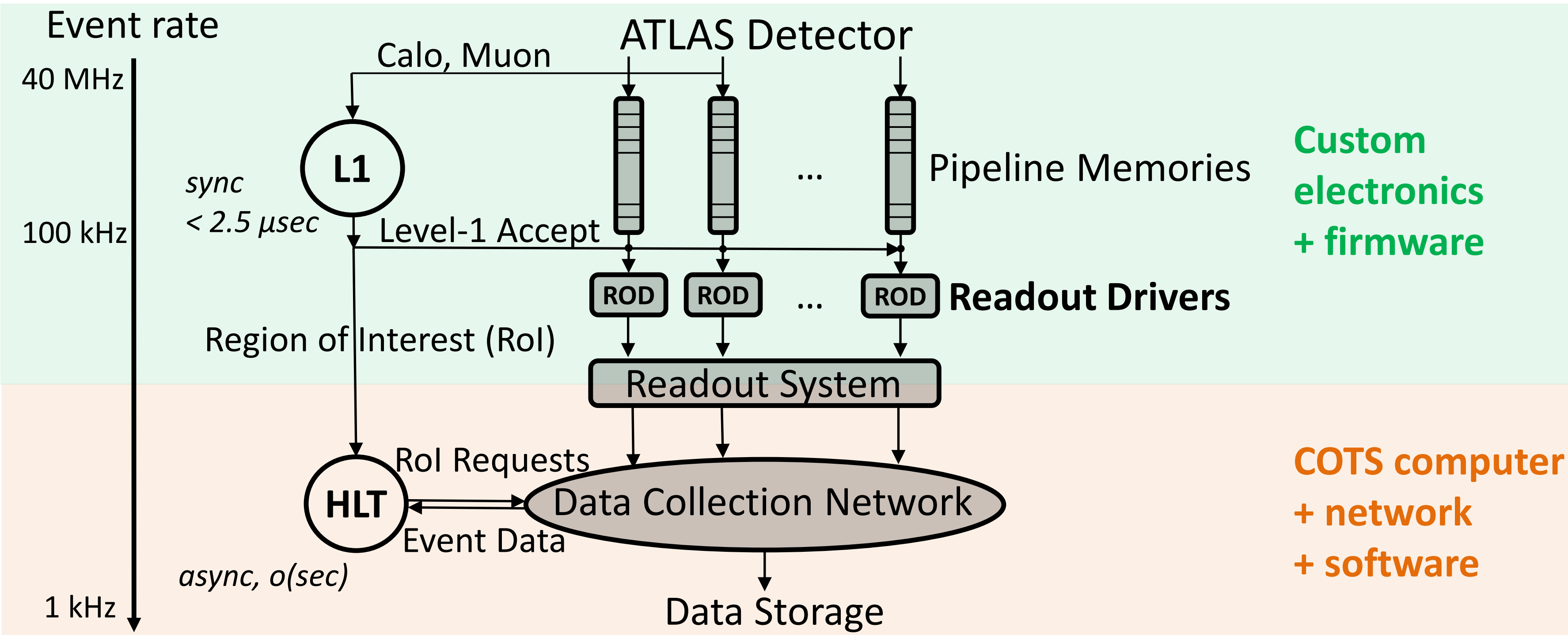


General-purpose experiment at the **Large Hadron Collider (LHC)** at CERN:
- proton-proton collisions at a centre-of-mass energy of 13 TeV
- about 25 collisions (pile-up) per bunch crossing (**BC**) every 25 ns (40 MHz)

⇒ 10^9 interactions per second potentially interesting to physics: need trigger system in order to select events which are interesting to physics and which can be recorded at a reasonable rate

→ For Run 4 of the High-Luminosity LHC (HL-LHC, starting 2025) we expect 140 collisions every BC

ATLAS TDAQ System

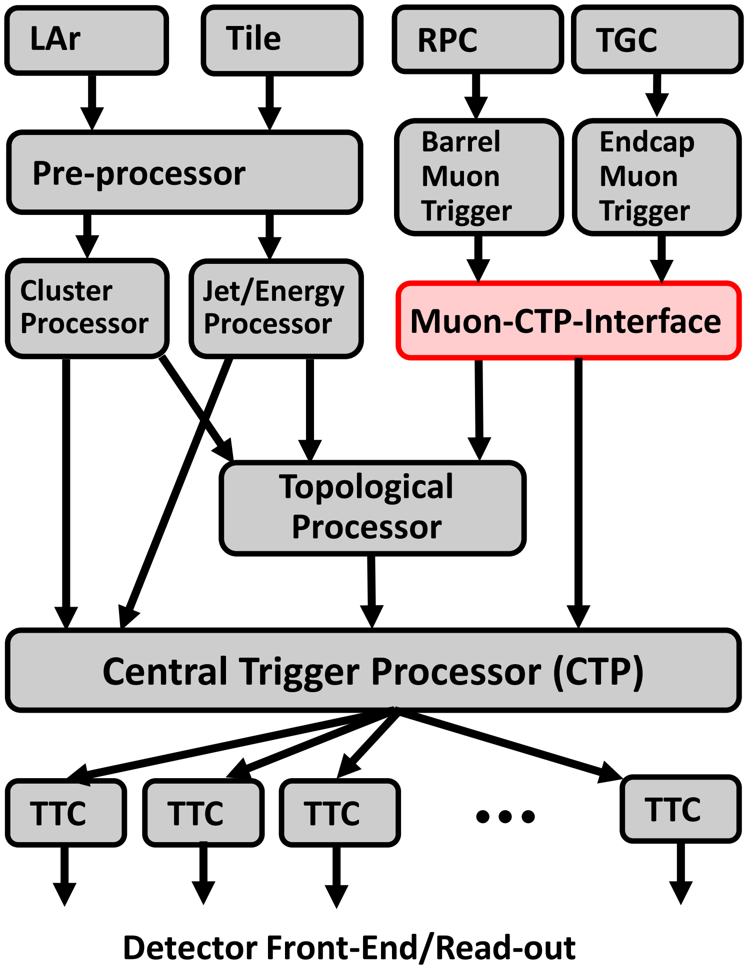


ATLAS Trigger/DAQ system:

- Custom electronics and firmware for Level-1 trigger (L1) and read-out system
  - ⇒ reduction of event rate to maximum of 100 kHz
- Commercial off-the-shelf computers, network and software for High-Level Trigger (HLT) and data acquisition
  - ⇒ reduction of event rate to 1.5 kHz (peak) and around 1 MByte per event (≲ 1.5 GByte/s)

ATLAS Level-1 Trigger System

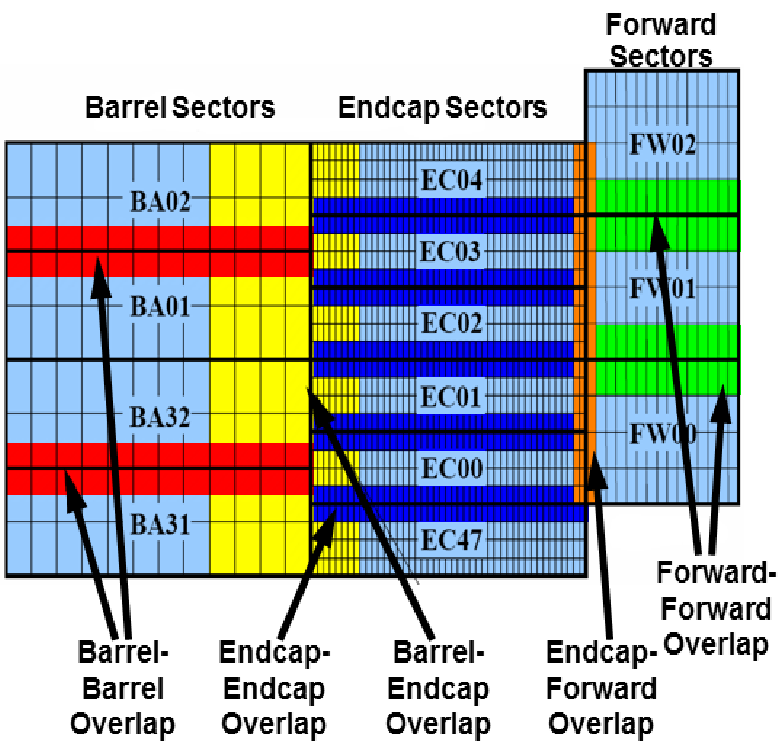
- The Level-1 trigger system processes **reduced granularity** information from **calorimeter** detectors and information from dedicated **muon** trigger detectors
- The trigger information is based on **multiplicities** of **topologies** of trigger candidate objects and **flags** for trigger conditions
- The muon trigger is based on resistive plate chambers (**RPC**) in the **barrel** region and thin-gap chambers (**TGC**) in the **endcap region**
- The **MUCTPI** combines the muon candidate counts from the barrel and the endcap regions
- The **Central Trigger Processor** (**CTP**) combines all trigger multiplicities and flags based on a trigger menu and makes the final Level-1 decision
- The **Timing, Trigger, and Control** (**TTC**) system sends the Level-1 trigger decision back to the detector front-end electronics, which perform (or not) the read out of the event data to the readout system



→ Please see also talk "ATLAS Level-1 Trigger System at 13 TeV" by L. Helary, on Thursday, May 25

ATLAS MUCTPI

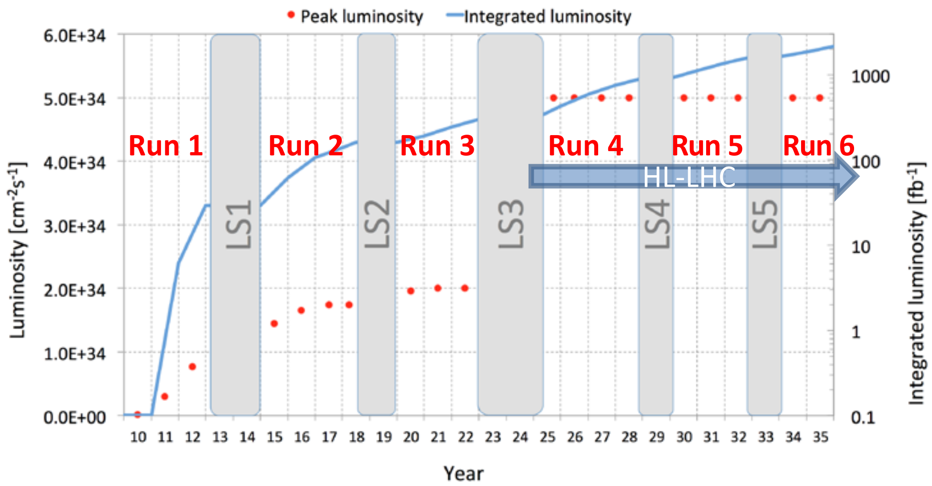
→ previous MUCTPI (currently used in Run 2):



Geometrical coverage of 1 of the 16 boards of the MUCTPI with 4 barrel, 6 endcap, and 3 forward sectors

- The MUCTPI receives **up to two muon candidates** from each of the 208 muon sectors (64 in the barrel region and 144 in the endcap and forward region) for each bunch crossing of the LHC
- The MUCTPI counts muon candidates **for six different p_T thresholds**
- The MUCTPI **avoids double counting** of single muons that are detected by more than one muon sector due to geometrical overlap of the muon chambers and the trajectory of the muon in the magnetic field
  (we call this the "*overlap handling*")

MUCTPI Upgrade - 1

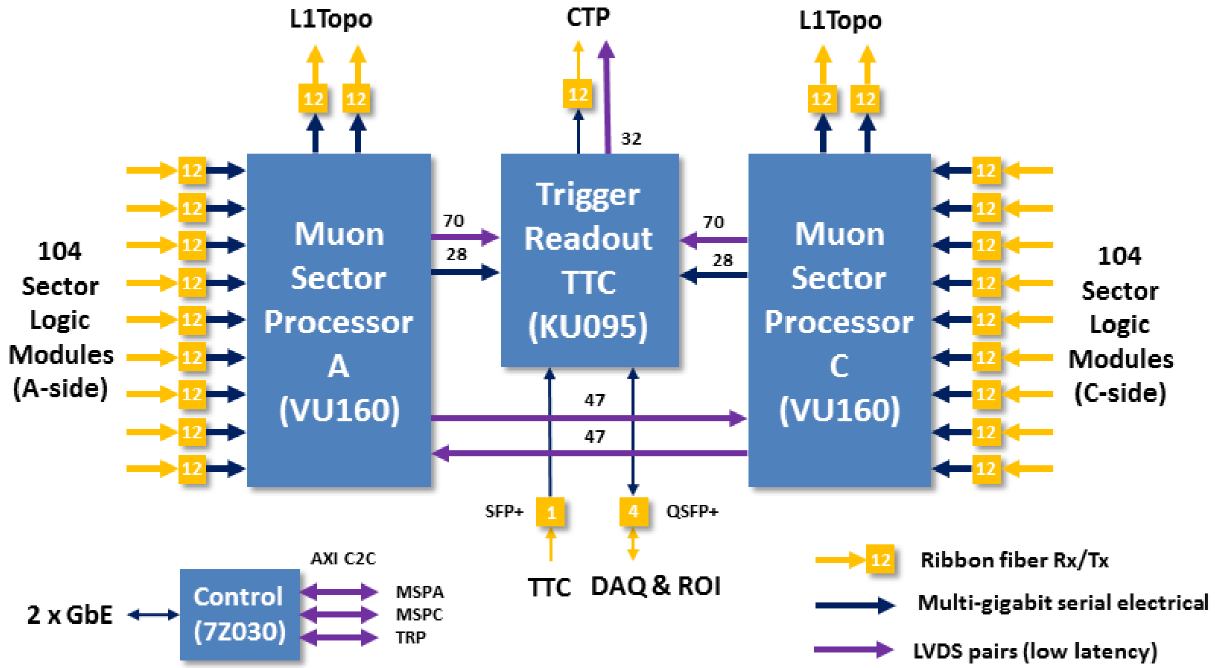


LHC Upgrade Plan

- **MUCTPI upgrade is part of the overall trigger upgrade on the road to the HL-LHC:**
  - Upgrade in line with development of New Small Wheel (NSW) of muon trigger system
- **Required improvements to MUCTPI:**
  - **send full-precision information on muon candidates to topological trigger processor**
  - **replace electrical connections between muon sectors logics and MUCTPI by optical links**: replace bulky and difficult to maintain cables, and allow for new/more information from the sector logic (more candidates, more precise position information, additional flags from algorithms identifying muon candidates)
  - **fit within the same tight latency requirement (8 BC = 200 ns)**
  - **be compatible with the upgrades for Run 4 of the HL-LHC**

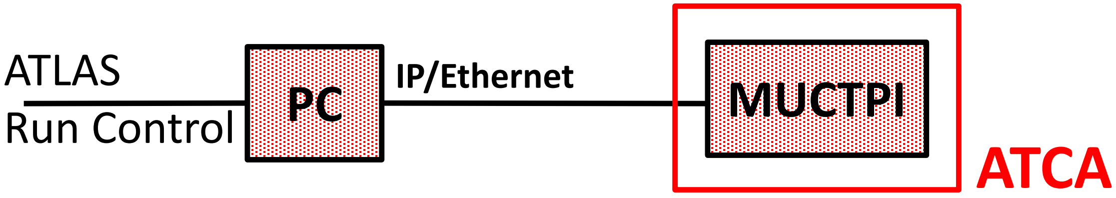
→ Earlier in this session: "Upgrade of the ATLAS L1 Muon Endcap Trigger" by S. Akatsuka

MUCTPI Upgrade - 2



- Implemented as one single **ATCA** blade (before: one VME crate with 18 modules!)
- Receive 208 optical links using fibre ribbons and optical receiver modules (Avago minipods)
- Two state-of-the-art FPGAs (***Muon Sector Processor***, Xilinx Virtex Ultrascale) for processing: overlap handling, counting, and providing muon candidates to Topological Processor
- One FPGA (***Trigger&Readout Processor***, Xilinx Kintex Ultrascale) for total count of muon candidates and readout of trigger information
- One **System-on-Chip** (***Control Processor***, Xilinx Zynq) for control, configuration, and monitoring; **A System-on-Chip (SoC) consists of a processor part (ARM Cortex A9) and programmable logic**

MUCTPI & Run Control



→ Integrate the MUCTPI into the run control system of the ATLAS experiment

Run Control = ATLAS TDAQ control communication (tree of controllers with a finite-state machine):

- Send **control** commands, e.g. start, stop, pause, run calibration etc.
- Load **configuration** data, e.g. lookup-table files, algorithm parameters, etc.
- Collect **monitoring** data, e.g. counters, selected event data, etc.

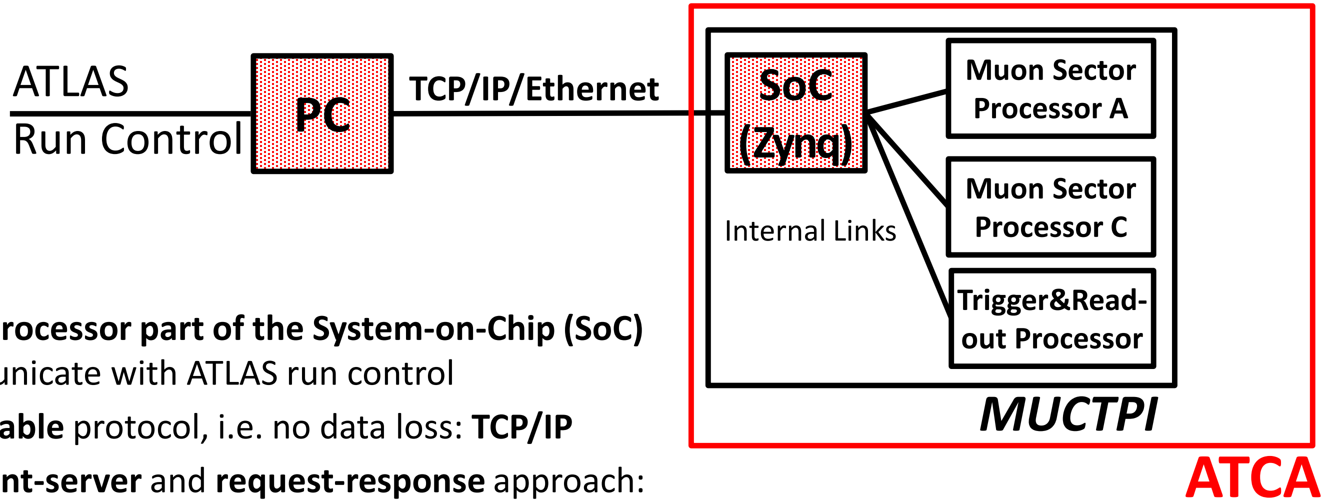
Usually it is NOT:

- no slow control: voltages, currents, temperatures, etc.
- no event data, *except for monitoring of selected event data*

In the following slides: two approaches to run control with the MUCTPI Upgrade

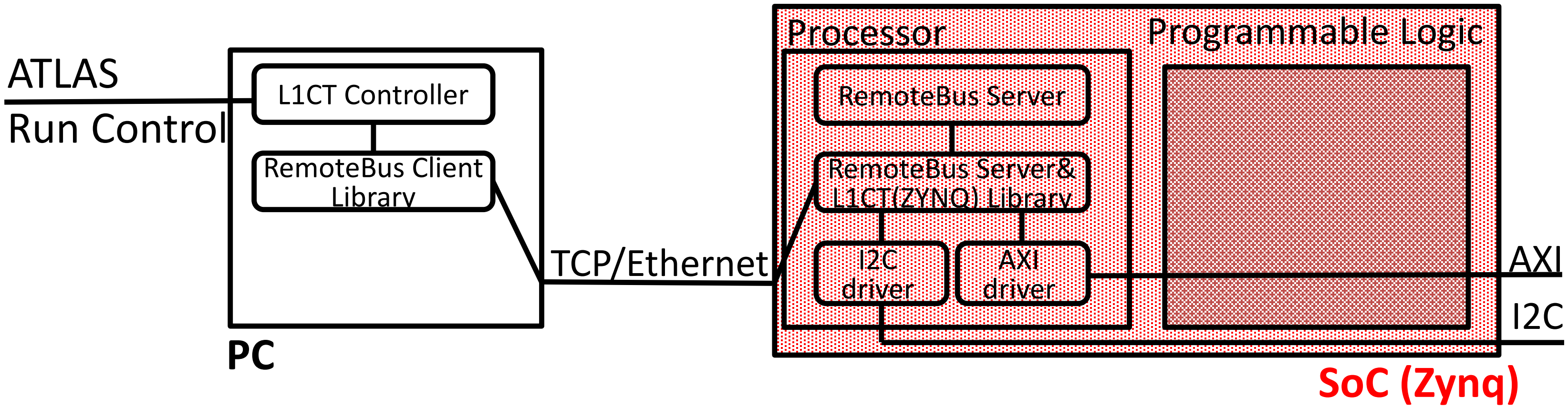
- RemoteBus software for run control
- Port of run control software to embedded Linux

RemoteBus - 1



- Use the **processor part of the System-on-Chip (SoC)** to communicate with ATLAS run control
- Use a **reliable** protocol, i.e. no data loss: **TCP/IP**
- Use a **client-server** and **request-response** approach:
  - Client = TDAQ controller on PC, sends requests
  - Server = process on SoC, receives requests, processes them, and sends responses: data or status/error message
- Use **synchronous** approach: as with previous MUCTPI, allow **multiple clients** and **multi-threaded server**
- Provide several modes of working:
  - **Single read/write** from/to memory on the Muon Sector Processor and Trigger&Readout FPGAs, as well as **block read/write** functions (as with previous MUCTPI)
  - **Provide extensibility for user-defined functions,** typically for more complex serial protocols for auxiliary hardware like DC/DC regulators, temperature/voltage monitoring, clock configuration etc. using serial protocols like I2C, SPI, JTAG etc.
  - **Allow queuing of requests**: bundle several requests before sending them together ⇒ mitigate latency overhead due to network transport
- **We named this software "*RemoteBus*"** **(similar to remote procedure call (RPC) and similar to read/write on a computer bus system as with previous MUCTPI which was using VME)**

RemoteBus - 2

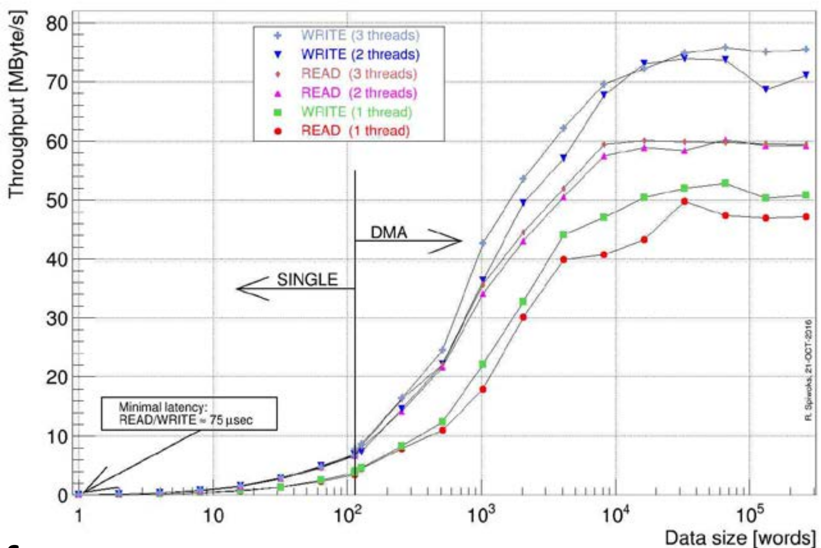


- Every RemoteBus Client (thread) has its **own TCP socket** and its own **RemoteBus Server thread**
- The RemoteBus Server **reads/writes** from/to the other processor FPGAs **using AXI** (Xilinx Protocol for communication between FPGAs) and **executes functions for auxiliary hardware** on the server using I2C, SPI, etc.
- Some requests are pre-defined in base classes, e.g. **READ(N)**, **WRITE(N)**, and additional requests are added depending on the hardware of the server (i.e. the MUCTPI)
- All **parameters** (request or response) are **32-bit data words**, added into the message or retrieved from the message in a **stack-like way**
- Additional request types can be added as functions to the server – like remote procedure call (RPC)
  → use **C++ inheritance** to **extend functionality**
- The **Yocto/OpenEmbedded** development framework is used for creating the Linux operating system, for building the application software (RemoteBus) and for providing all files necessary to boot and run the System-on-Chip;

  The Yocto/OpenEmbedded is a Linux Foundation workgroup, which provides a complete development environment with tools, metadata, and documentation for creating Linux distributions aimed for, but not restricted to, embedded systems

RemoteBus - Results

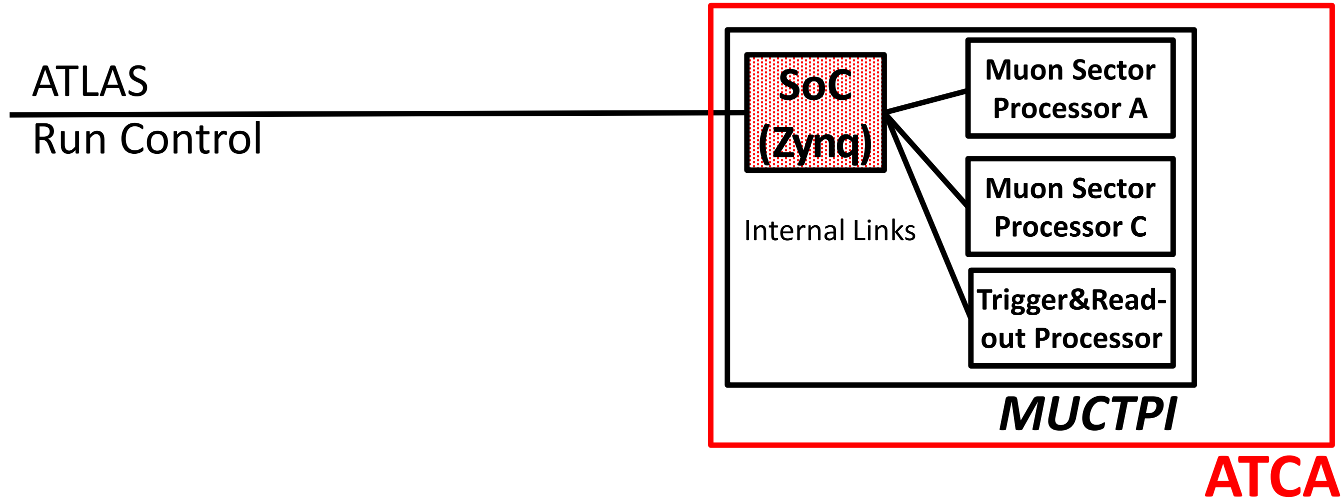
- **Implementation:**
  - **Two base classes "*Client*" and "*Server*"** were implemented for communication between any two computers
  - **Two derived classes "*ZC706Client*" and "*ZC706Server*"** were implemented for the ZC706 (Zynq) evaluation board, requests were added for the hardware of the evaluation board, e.g. I2C, SPI, etc., in particular for reading temperature/voltage monitors and DC/DC regulators




```
> menuZC706Client:
> "BUS[0]/UCD90120A/TEMP" = 27.625000 C
> "BUS[0]/UCD90120A/TEMP" = 27.625000 C
…
```

- **Performance:**
  - The **minimal latency** for a request-response transaction is around **75 µsec**
  - The **bandwidth** is limited by the Ethernet throughput and reaches about **50 Mbyte/s** for 10 kword blocks, this is about 10 times more than the throughput of the previous MUCTPI using VME
- Running **multiple clients** or **client threads** is safe and increases performance
- **RemoteBus will be used for testing the MUCTPI prototype**

Alternative approach: TDAQ/Embedded Linux



“*Push*” the ATLAS run control software directly onto the System-on-Chip:

The **TDAQ controller** runs on the processor part of the SoC (running Linux)

→ **In the ATLAS Level-1 Central Trigger, we have started to evaluate the porting of ATLAS TDAQ software to embedded Linux using the Yocto/OpenEmbedded framework**

Summary

- **New MUCTPI prototype became available begin of May 2017**
- **The run control path has been tested with Xilinx Zynq evaluation boards**
- **RemoteBus software was developed with functions for accessing memory in the processor FPGAs, as well as for auxiliary hardware using I2C, SPI, etc.**
- **A port of the ATLAS TDAQ software to Xilinx Zynq with embedded Linux is under way**
- **The Yocto/OpenEmbedded development framework is used for building the Linux operating system and our application software**

⇒ Trigger electronics are not only becoming fully optical, much denser, and more intelligent for processing but also more intelligent to control …