Communication Models for Run Control with ATCA-based Systems

- Introduction
- Architecture models and survey
- Outlook

I apologize for the bias towards my work in the ATLAS Level-1 Central Trigger (L1CT) on the Phase-1 Upgrade of the Muon-to-Central-Trigger-Processor Interface (MUCTPI)

Definition: Run Control

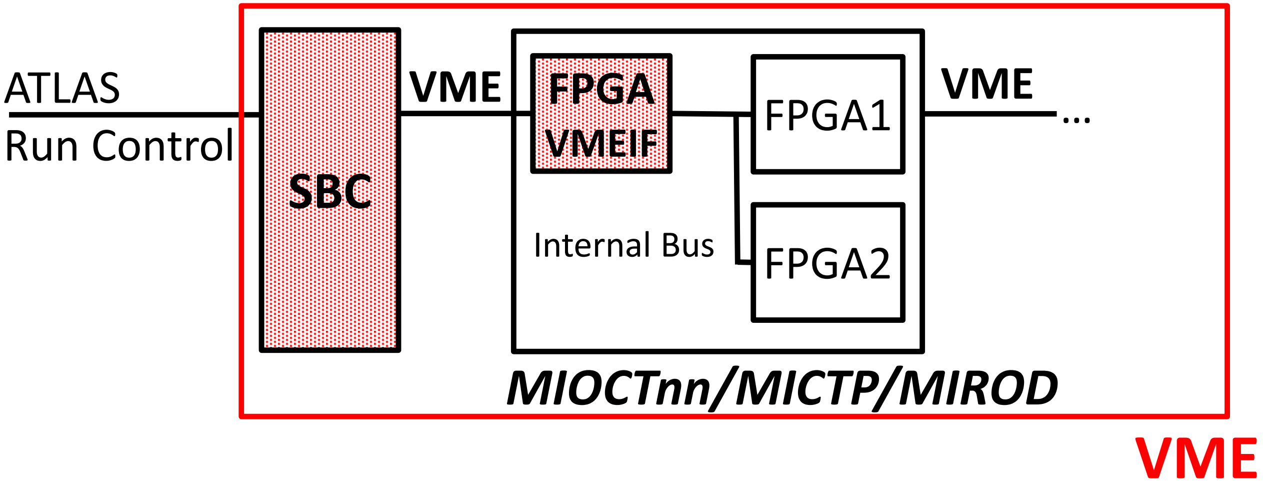
Run Control = Trigger/DAQ control communication:

- 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.

It is NOT:

- **no slow control: voltages, currents, temperatures, etc. (→ IPMI)**
- **no event data, *except for monitoring of selected event data,* (→ Readout Links)**

Legacy model for VME



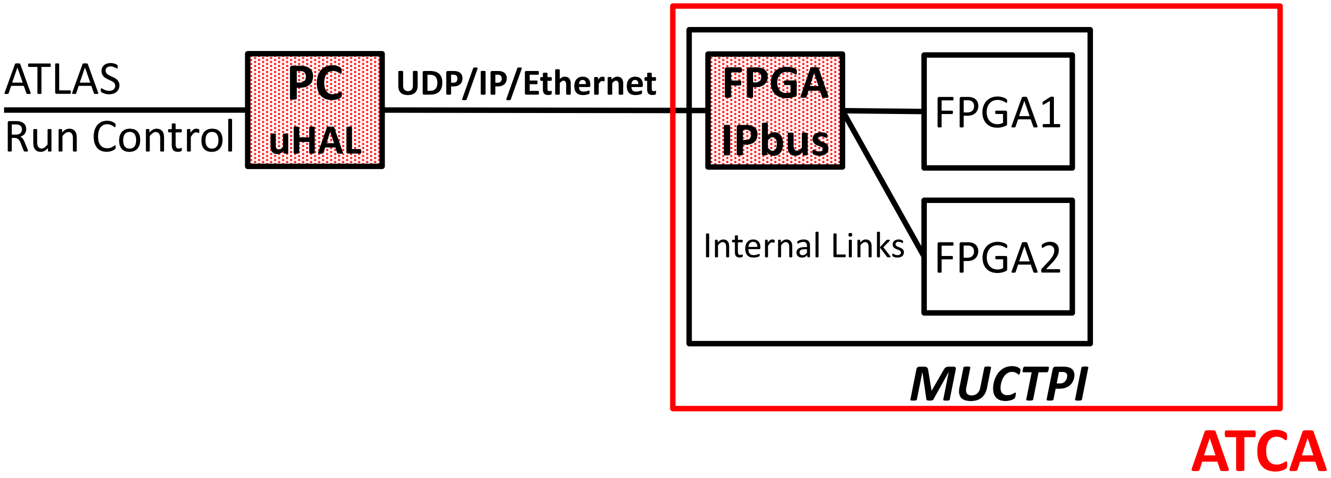
- Hardware modules are based on **VME**
- **Single-board computer (SBC)** communicates on one side with the Run Control via IP/Ethernet
- On the other side the SBC communicates via **VME** with the hardware modules: read/write cycles (single or block)
- Hardware modules have a dedicated **FPGA with VME I/F firmware**, and internal bus to other FPGAs with individual strobe lines

Almost all ATLAS sub-detectors use this model

ATLAS/TDAQ provides support for

- Purchasing of SBCs
- Common VME driver and library

Model 1 for ATCA: IPbus

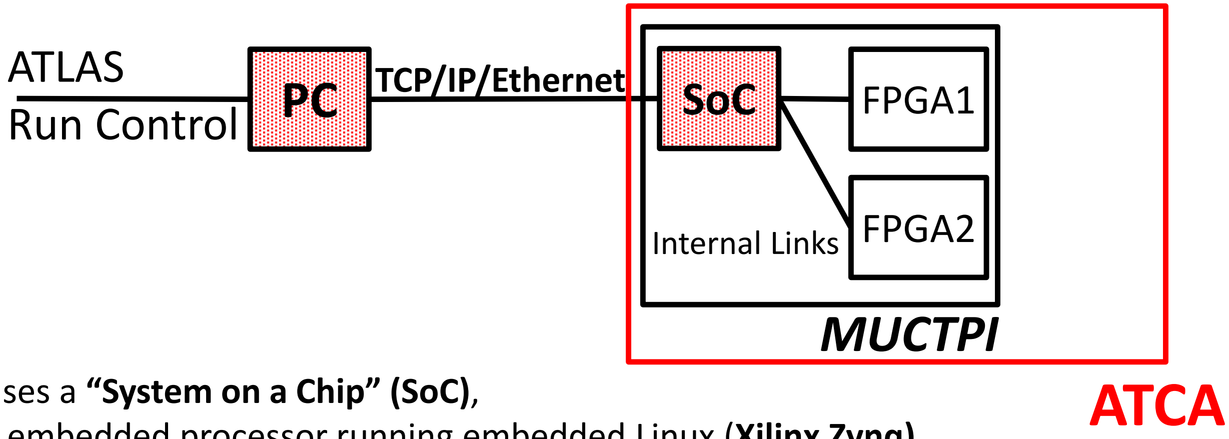


- Provided by CMS: based on firmware and software
- An **FPGA receives UDP packets** and performs read/write transactions with other (processing) FPGAs
- **ATLAS/TDAQ provides the s/w library (uHAL) as part of its releases**
- **In the ATLAS L1CT, we have tested it – it does work:**

  ⇒ We consider it a fall-back solution for the L1CT/MUCTPI

  But: UDP is not a reliable protocol, packets can get lost, and for multiple clients a ControlHub software is needed, written in Erlang …

Model 2 for ATCA: RemoteBus (L1CT)

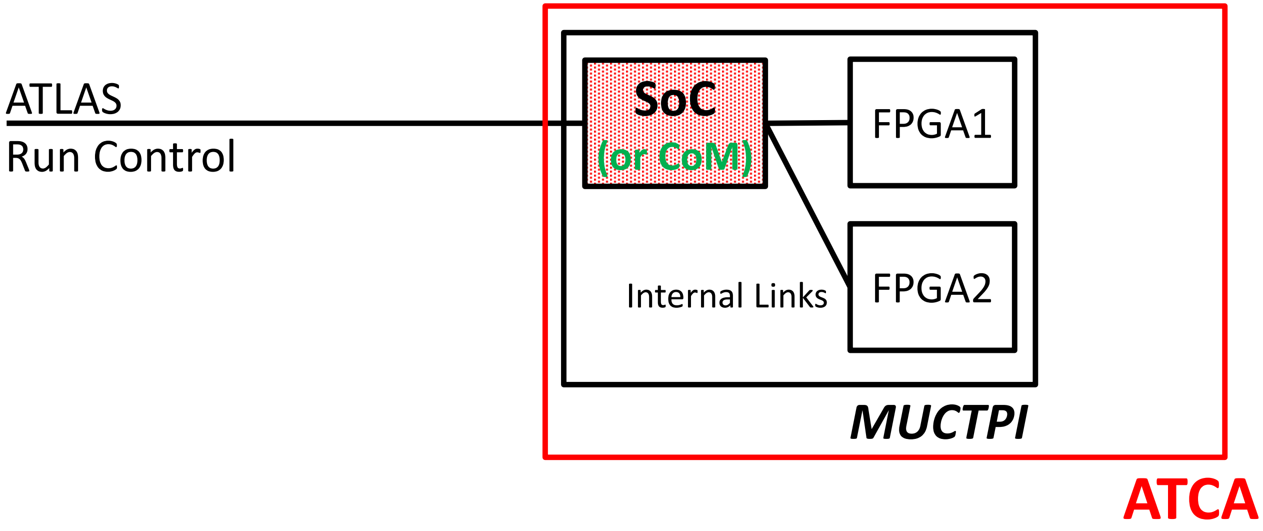


- The MUCTPI uses a **“System on a Chip” (SoC)**,
  i.e. FPGA with embedded processor running embedded Linux (**Xilinx Zynq)**
- Use a **client-server** and **request-response** approach:
  - client = TDAQ controller on PC sends requests
  - server = process on Zynq, receives requests and sends responses
- Use **TCP**: reliable protocol, i.e. no data loss
- Use **synchronous** approach: as before with VME, but allow **multiple clients** and **multi-threaded server**
- Provide several modes of working:
  - Single and block **read/write** functions (as before with VME)
  - **Remote functions** for more complex hardware access (like Remote Procedure Call, RPC), e.g. **I2C, SPI, JTAG,** etc.
  - **Queuing of several requests**: bundle several requests before sending them together ⇒ mitigate latency overhead
- **Extend functionality** by using C++ inheritance for adding more complex functions
- Use **Yocto/OpenEmbedded framework** for building Linux operating system and RemoteBus software

⇒ We use it currently to test a prototype of the new L1CT/MUCTPI

This is an example of using remote procedure call developed by L1CT, other implementations in ATLAS exist

Model 3 for ATCA: TDAQ/Embedded Linux



- The **TDAQ controller runs directly on the SoC***

→ **How difficult to port ATLAS TDAQ s/w? How much effort to maintain? How much effort to fulfil CERN/IT's security requirements?**

In the L1CT, we have started to evaluate the porting ATLAS TDAQ to embedded Linux using the Yocto/OpenEmbedded framework
→ technical student project, started 03/17, so far going quite well …

****Alternatively, a CoM ("Computer on Module" ⇒ "PC on ATCA blade") could be used***

Survey: Run Control with ATCA in ATLAS

ATLAS Project	Hardware (SoC, FPGA)	Software/Firmware
gFEX	v2: Xilinx Zynq 7045	Linux (Yocto/OpenEmbedded) + IPbus (software emulation)
	v3: Xilinx Zynq Ultrascale+ MPSoC ZU19EG	Linux (Yocto/OpenEmbedded) + IPbus (software emulation)
		Linux (Yocto/OpenEmbedded) + TDAQ – ***planned***
TOPO	Xilinx Kintex7 325	IPbus (firmware)
jFEX	Xilinx Zynq 7030	IPbus (firmware on Zynq/PL)
		Linux? + software?
eFEX	Xilinx Virtex7 550, 690	IPbus (firmware)
	GBT (via Hub module)	Control traffic with deterministic latency – ***possibility***
MUCTPI	Xilinx Zynq	IPbus (firmware on Zynq/PL) – ***fallback***
		Linux (Yocto/OpenEmbedded) + RemoteBus (L1CT)
		Linux (Yocto/OpenEmbedded) + TDAQ – ***project***

Survey: Run Control with ATCA in ATLAS (cont'd)

ATLAS Project	Hardware	Software/Firmware
CSC ROD	Xilinx Zynq	Linux (RTEMS and ArchLinux) + RPC + JSON (Four daughter boards: RTEMS, one: ArchLinux)
Pixel-Chip teststand	gen1: Xilinx Virtex 4 (PPC405)	Linux (RTEMS) + TDAQ4 (private port to PPC)
	gen3: Xilinx Zynq	Linux (ArchLinux) + TDAQ5 (private port) – ***discontinued*** Linux (ArchLinux) + Remote Call Framework (RCF)
AFP prototype (same as Pixel test stand gen3)	Xilinx Zynq	Linux (ArchLinux) + TDAQ5 (private port) - ***discontinued*** Linux (ArchLinux) + Remote Call Framework (RCF)
NSW Trigger Processor	GBT-SCA (FE ASICs) E-Links (FPGAs)	None
TileCal PreProcessor (ROD)	Prototype: Xilinx Virtex7 + Kintex7	IPbus
FTK Data Formatter (DF) **FTK Level-2 Interface Card (FLIC)**	Xilinx Virtex7	IPbus
Lar LATOME	Altera Arria 10 FPGA	IPbus

Survey: Run Control with ATCA in ATLAS

My observations:

- Many ATLAS projects are using IPbus and people are happy to use it
- Many ATLAS projects are using or plan to use SoC,
  all of which are based on Xilinx Zynq (ARMv7 processors, 32-bit)
  Note: the next generation (Xilinx Zynq Ultrascale+) is based on
  ARMv8 processors (64-bit)
- A few different implementations of RPC-like applications on embedded Linux exist and people are happy to use them

Outlook

- **Several RPC-like solutions:**
  Could the RPC-like applications be unified?
  Could TDAQ provide an RPC stub for the TDAQ run controllers?
  I don't know the answer, but a better way to unite our efforts could be the following:

- **Porting TDAQ to an embedded Linux has definite advantages:**
  - No need for an intermediate layer like IPbus (software & firmware) or RPC-like applications (software)
  - Looks like legacy model of SBC and VME: TDAQ controllers can be written in a similar way
  - Common low-level functionality for inter-FPGA communication, I2C, SPI, JTAG, etc. could be provided in a way similar to ATLAS ROD Crate DAQ (common drivers and libraries)
  - Embedded Linux provides a full operating system which can run many user applications and allows direct interactive access (ssh)

  → In the ATLAS L1CT, we are currently investigating the possibility to port TDAQ to Zynq: technical student project, started 03/17

  → If possible, could a port of the software be maintained by ATLAS TDAQ?

  → What support could possibly be provided by CERN/IT?

- **Investigate the possibility to have CERN CentOS (CC) for ARMv8 processors?**

- **What do other experiments do?**

Let’s build intelligent run control directly into each ATCA blade!