System-on-Chip (SoC)

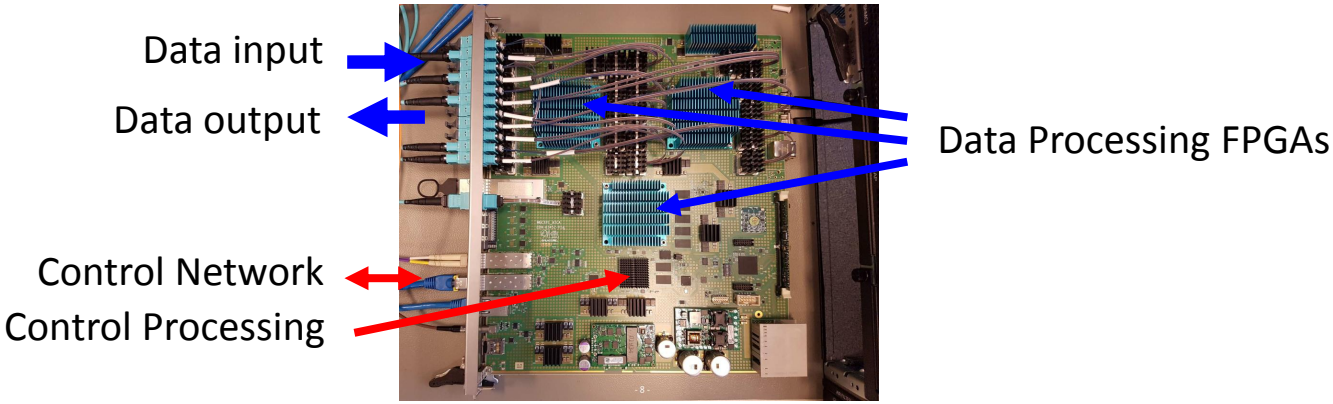
Experience and Recommendations

- **Why do we use SoCs?**
- **What are SoCs actually?**
- **How do we use SoCs?**
- **What message to take away?**

With contributions from many people …

Big thanks!

Control of Electronics Modules (1)



Many electronics modules for trigger and readout in particle physics experiments have a similar structure:

- Several high-end FPGAs for processing, usually 1 to 5.
- Many high-speed links for data input and output, usually ~10 to ~500 links of 1 to 28 GBits/s (planned) .
- Something for control …

Control - send control commands: e.g. start, stop, pause, reset, etc.

Configuration - load configuration data:

Monitoring - collect monitoring data:

- **Hardware related, i.e. related to the clock, power, and optical chips, FPGAs, etc.**

Settings of clock, power, optical chips, firmware configuration of FPGAs etc., usually using industry standards like I2C, SPI, JTAG, etc.

Operational values of hardware, e.g. temperatures, voltages, optical power, etc.

- **Run control related, i.e. related to physics/run (functionality implemented in firmware):**

Control register and/or FPGA memories, e.g. processing settings, look-up tables etc.

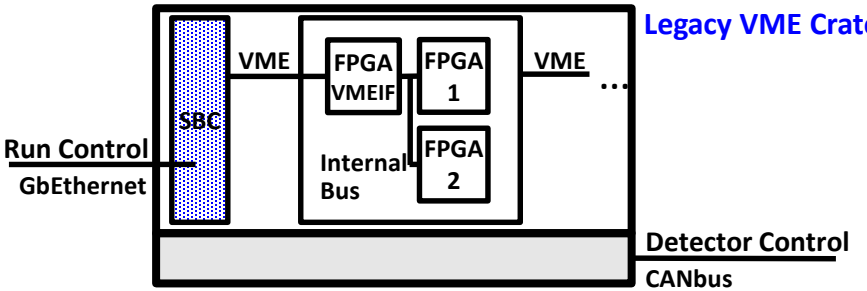
Status registers, counters/rates of physics events, data of some selected physics events, etc.

Control of Electronics Modules (2)

In the past, very popular - VME:

Hardware control: CANbus

Run control: Single-Board Computer (SBC) or VME bridge



Today, many new projects use ATCA:

Control is oriented towards GbEthernet:

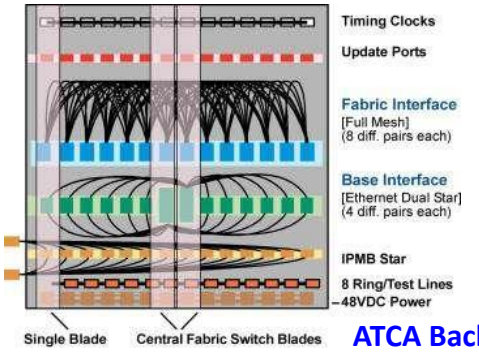
Hardware control:

Blade → IPMI → shelf manager → SCADA*

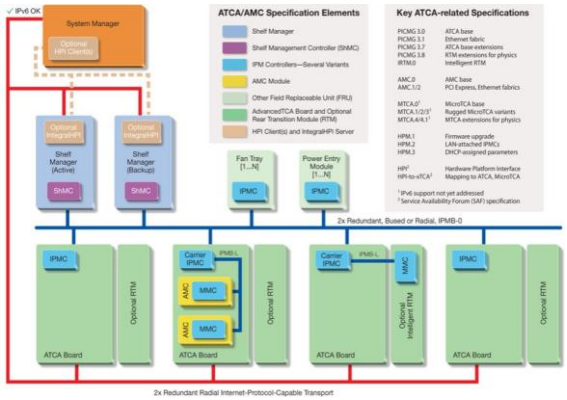
Run control:

Via hub module and base interface, or directly to ATCA blade …

⇒ **Need a new control strategy!**



ATCA Backplane



ATCA Hardware Platform Management

***SCADA = Supervisory Control And Data Acquisition**

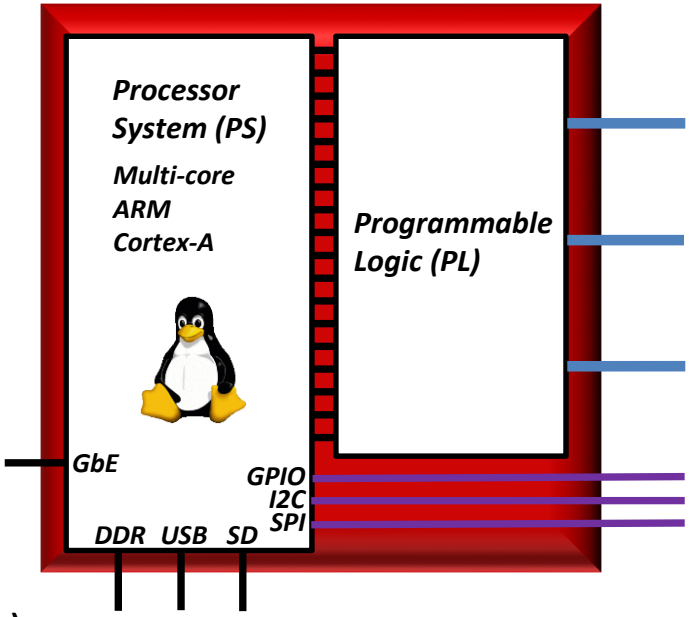
System-on-Chip (1)

System-on-Chip:

Processor system + programmable logic, *"CPU and FPGA"*

This definition of SoC is more restrictive than in Wikipedia

- **Processor system (PS) =** like CPU:
  - Currently all are multi-core ARM processors
  - Has memory and peripherals, e.g. GbEthernet, I2C, SPI, GPIO, etc.
  - Runs software: "bare-metal" application or operating system, e.g. Linux
- **Programmable logic (PL) =** like FPGA:
  - Has logic cells, memory blocks, and I/O links
  - Can interface to the processing FPGAs, e.g. using Xilinx AXI Chip2Chip protocol



→ Very popular SoC: Xilinx Zynq

Zynq SoC ["Zynq"]: ARMv7 Cortex-A9 (32-bit, 1-2 cores)

Zynq UltraScale+ MPSoC ["ZynqMP"]: ARMv8 Cortex-A53 (64-bit, 2-4 cores)

… both with different extent of PL functionality

The Intel Stratix 10 SoC or Arria 10 SoC can also be alternatives …

System-on-Chip (2)

This part if very well known to FPGA designers

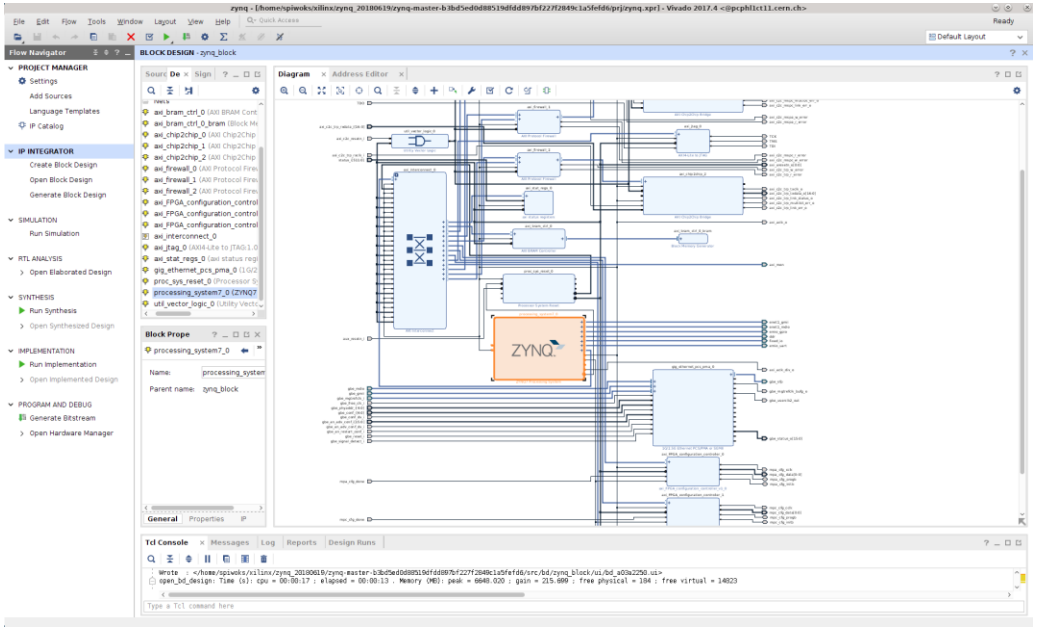
For the Programmable logic (PL):

Use FPGA design tool, e.g. Xilinx Vivado,

Design your logic

using FPGA resources:

- Multiple Gigabit Transceivers (MGT) and LVDS links
- Use them for AXI chip 2 chip: to other FPGAs directly mapped into the memory of the PS
- Ethernet, e.g. 10Gb/s
- Memory blocks
- Logic blocks for trigger/readout algorithms



⇒ Produce bit stream file (for FPGA configuration)
+ board support package (for s/w development)

System-on-Chip (3)

This part if very well known to embedded system designers

For the Processor system (PS):

Current SoCs use ARM-based processor cores

⇒ **Tools required:** toolchain = compiler + system libraries

- **cross compilation** or
- compile natively, e.g. ARM server or on SoC itself

⇒ **Artefacts to be generated:**

- First-Stage Boot Loader (FSBL): for set up of SoC, loading bit stream, and SPL
- Secondary Program Loader (SPL): often U-Boot, if not running a bare-metal application
- Operating system: kernel, device tree, and root file system

Operating system:

- **Linux (very poplular):**
  - **PetaLinux**: Xilinx provided distribution, convenient to start with
  - **Yocto**: free distribution build tool with Xilinx meta layer, better adapted for building your own application software
  - Others: **ArchLinux, CentOS,** …
- **Real-time operating system if reaction time matters, e.g. RTEMS, freeRTOS etc.**

Examples of Use of SoC

A few current and future projects in LHC (mainly ATLAS and CMS)

showing some typical uses of the SoC

This is my selection of projects with SoC high-lighting different aspects of their uses

ATLAS CSC ROD

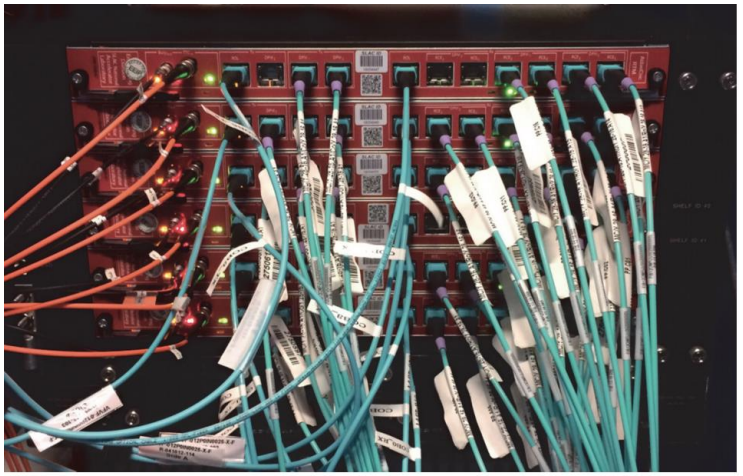
SLAC NATIONAL ACCELERATOR LABORATORY

Readout Driver (ROD) of the Cathode Strip Chamber (CSC) of the ATLAS muon system:

- Provides readout for all 32 CSCs.
- One **ATCA** shelf with 6 **Cluster On Board (COBs)** + Rear-Transition Modules (RTMs)
- Each **COB (= ATCA blade)** is equipped with **5 Reconfigurable Cluster Elements (RCE)**:
  - **4 Data Processing Modules (DPM)**
    - Each DPM has **2 Xilinx Zynq** Z045: receive data, extract features, send data;
      real-time operating system RTEMS, control program uses Sun/RPC and JSON configuration data
  - **1 Data Transmission Module (DTM)**
    - each DTM **1 Xilinx Zynq Z030:** manage trigger and busy signals
      runs ArchLinux, control program uses Sun/RPC and JSON configuration data
  - **24-port 10 GbE Fulcrum Ethernet switch** for network connection
- **The CSC RODs were installed in ATLAS in 2014. They have been running very stably since.**

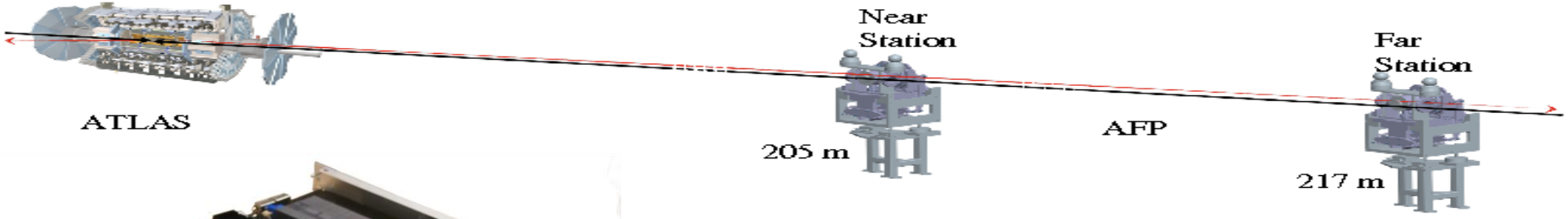
Early adoption of SoC technology

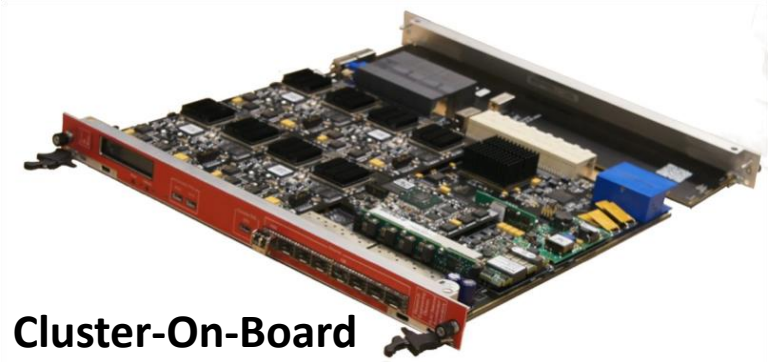
COBs



RTMs GLINKs from CSC chamber and SLINKs to readout

AFP – ATLAS Forward Proton

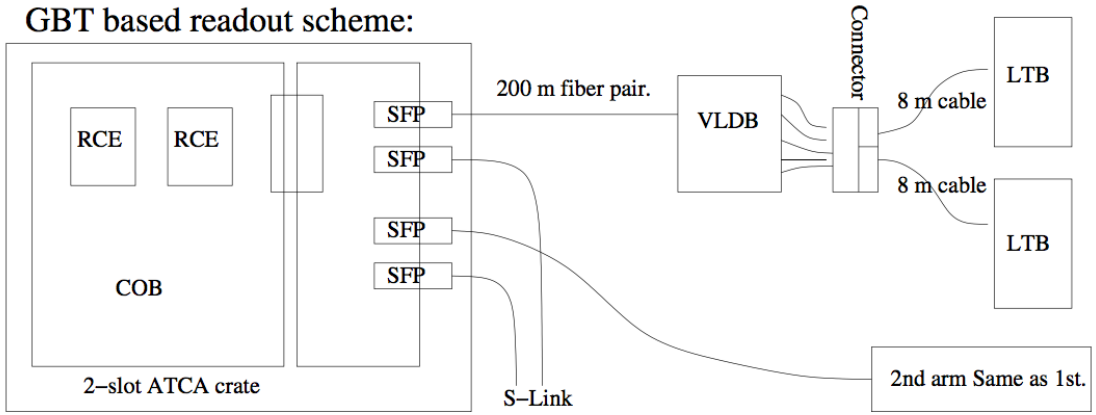




Cluster-On-Board (COB)

- **Single COB with 2 RCE for readout of AFP:**
- ATLAS TTC interface in the RTM.
- SFP connections to VLDB (front-ends) and S-Link.
- 16 ATLAS IBL-like tracker modules.
- 2 Time-of-Flight systems.
- Control program uses Remote Call Framework (C++).

- **The COB-based system is supposed to be installed during LS2.**
- **The same setup has been used for the Inner Tracker Pixel Test Readout.**



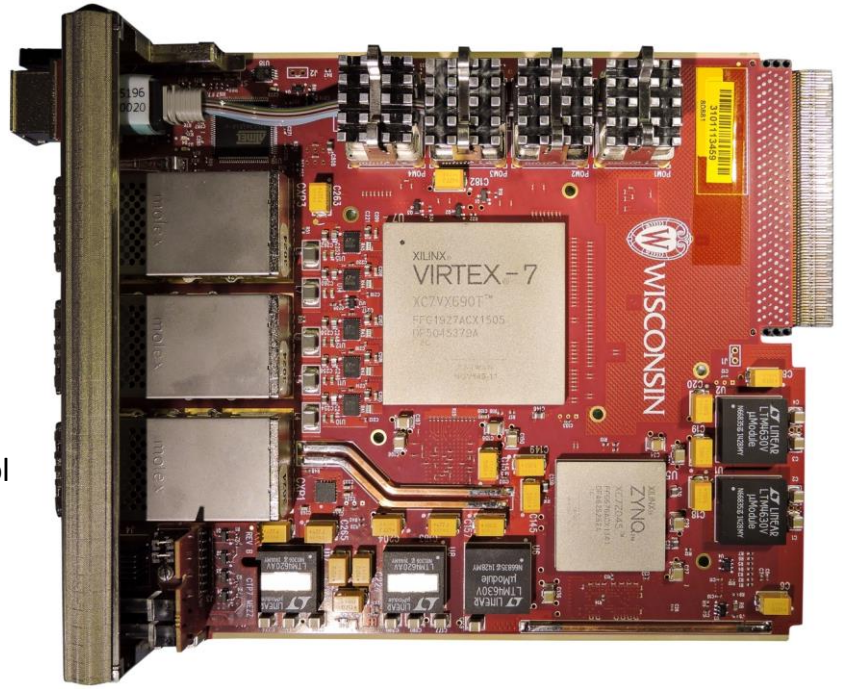
Versatile Link Demonstrator Board (VLDB)

CMS – CTP7

Early adoption of SoC technology for µTCA card

µTCA-based Calorimeter Trigger Processor
First CMS ZYNQ SoC card (Phase-1 upgrade)

- *22 cards (incl. spares)* ***deployed in production in 2015 and running stably*** *since as part of calorimeter L1 trigger chain*
- Also in widespread **use as a Phase-2 demonstrator platforms** in several CMS upgrade projects and multiple institutions*



Key designs points:

- **Virtex-7 690T FPGA with 80 MGTs for data processing**
  - 69RX/48TX 10Gbps optical links, 12RX/TX backplane MGT links
- ZYNQ XC7045 with dual ARM Cortex-A9 CPU used for board control
- **Embedded Xilinx PetaLinux running on the ZYNQ**

Use of the ZYNQ SoC:

- **Provides board configuration support**: booting FPGA, configuring memories, clocks, optics, etc.
- Acts as primary network-connected TCP endpoint on the board during online operation
- **AXI Chip-to-Chip** employed to seamlessly extend the memory space of the ZYNQ SoC to the Virtex-7 FPGA.
- **Xilinx Virtual Cable (XVC)** service running on the ZYNQ replaces a physical cable with a TCP/IP connection to Xilinx toolset
- **Eye Scan Capability of IBERT** ported to ZYNQ: integrated Eye Scan (IES)
- **Application software based on Remote Procedure Call (RPC)**

***CMS Phase-2 Demonstrators:**
Track Trigger: CERN, Cornell, Rutgers
Correlator Trigger: CERN, FNAL, Wisconsin
EMU, HB/EB Cal Readout and Trigger: CERN, FNAL, Princeton, Texas A&M, UCLA, Virginia

CMS – ELM1

“ELM”: Embedded Linux Mezzanine (Phase-2 upgrade)

Key design points:

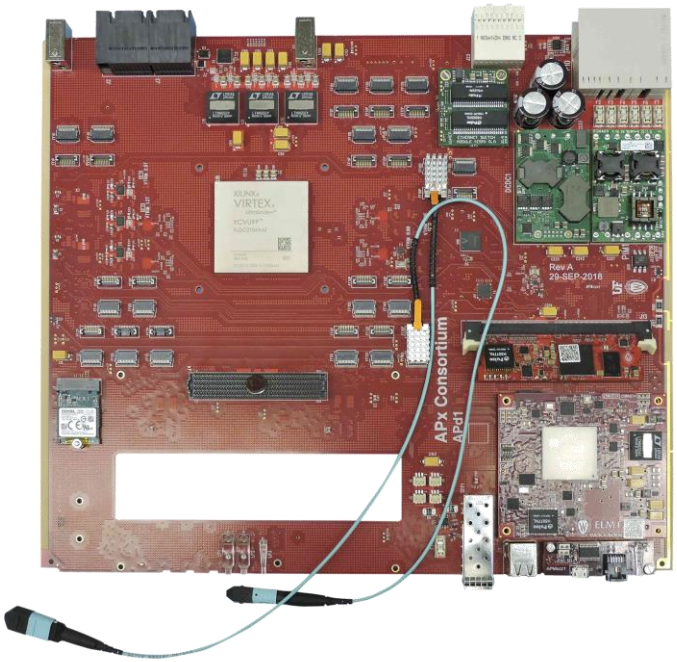
- Control point for **ATCA blades in mezzanine** form factor (84mm x 75mm): **stretched version of COMe mini, with a second MGT connector** added and PC/AT pins repurposed to ZYNQ PL IOs
- Adopted by multiple CMS Phase-2 upgrade back end subsystems*
- **Sub-project of APx consortium** **
- ELM1: 7-Series ZYNQ, XC7Z045-2 (8 GTX MGTs) exists, **ELM2: ZYNQ Ultrascale+ MPSoC is being designed**
- Programmable IO: >24@3.3V, 74@1.8V, 8x 10Gbps MGTs
- Embedded Xilinx PetaLinux running on the ZYNQ

Use of the ZYNQ SoC:

- Extending the concept successfully pioneered with CTP7 design
- Low-latency access point tightly integrated with processing FPGA(s): **MGT-based AXI Chip-to-chip links**
- **PL-based 10GbE TCP/IP Offload Engine (TOE):** fast data transfer to processing FPGA(s) for large memory configuration and test stand DAQ applications
- Interface to PCIe/SSD
- **Application software based on Remote Procedure Call (RPC), using Google Protocol Buffers**

First example of use of the ELM:

- **Advanced Processor demonstrator #1 (APd1) board**: demonstrator card for the CME Phase-2 Trigger upgrade, essentially a “large FPGA (VU9) with 100 links @ 28 GB/s”



***Present CMS Phase-2 Intended Use:**
Calorimeter, Correlator and Muon Triggers
ECAL Barrel, CSC and GEM readouts.

****APx Consortium: University of Florida, University of Virginia, Fermilab, University of Illinois at Chicago, University of Wisconsin-Madison, University of Notre Dame**

ZYNQ-IPMC*

An IPMC is required on each ATCA blade in order to make it work with the ATCA shelf

Key designs points:

- **Open-source non-NDA HW/FW/SW ATCA IPMI Controller**
- Adopted by multiple CMS Phase-2 upgrade back-end subsystems
- Developed at University of Wisconsin, sub-project of APx consortium (see slide 11)
- **Real-time monitoring, diagnostic and control of ATCA board services (control of hw)**
- **244-pin MiniDIMM form factor, ~100 3.3V configurable IOs in ZYNQ PL**
- ZYNQ XC7Z020 SoC
- 16 ADC channels (16-bit) for fast response signal monitoring

Use of the ZYNQ SoC:

- Cortex-A9 @ 600+ MHz backed by 256MB of RAM is a radical upgrade in resources compared to typical IPMC hardware platforms
- **C++/FreeRTOS-based real-time framework**
- Employing a true object-oriented approach to the design without performance penalty
- **HW design to be fully published**: schematics, PCB, artwork, BOM, for independent production
- **SW design to be released under GPL**, no NDA required to use/reference

***IPMC = Intelligent Platform Management Controller**

Other IPMC cards exist, notably the CERN IPMC

gFEX - Global Feature EXtractor (1)

Part of the Phase-1 Upgrade of the ATLAS Level-1 Calorimeter Trigger

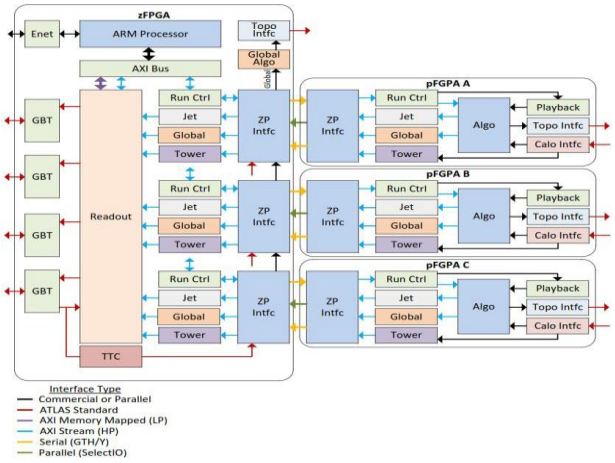
- **Board architecture**
  - **3x UltraScale+ Virtex Processor FPGAs** - U9P
    - Full calorimeter data sent over 300 input fibres to a single gFEX board
    - Run processing algorithms: identify jets, calculate global event observables (Large-R jets, MET), provide trigger objects.
  - **1x UltraScale+ Zynq MPSoC** - ZU19
    - Configuration, Slow-control and Monitoring.
  - **ATCA platform with IPMC full control and monitoring capability**
- **Custom Operating System (OS)**
  - **Using OpenEmbedded/Yocto toolchain to cross-compile Linux kernel; maintain official meta-l1calo layer registered with OpenEmbedded.**
  - Kernel patches and custom recipes written for python packages to be included in custom-developed OS on top of mainline Linux.
  - Undergoing implementation of **ATLAS DCS OPC-UA project**, as well as a recipe for cross-compiling **ROOT6.**

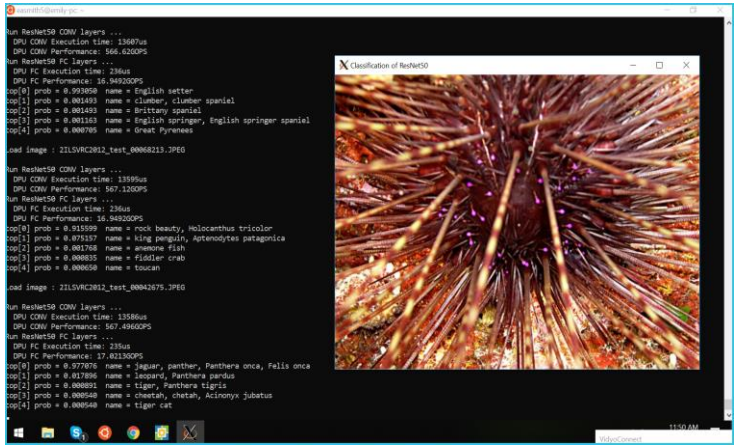
- **Functionality of ZU19**
  - **Configuration:** clocks, FPGAs, golden image, etc. using a well-defined user-facing register map.
  - **Slow-control:** enable read-out fibers through read-out path, set board Run-Mode state, activate/deactivate input/output fibers link by link, read/write QSPI flash, and other registers defined in a user-facing register map.
  - **Slow-control/monitoring via IPBus protocol** and any other communication protocol allowed through software developed in-house: **"ironman", a single-threaded, reactor-based event-driven callback Python framework**; also supports websockets and HTTP streaming.
  - **Monitoring** of the onboard hardware components through **I2C.**

gFEX - Global Feature EXtractor (2)

→ Use of SoC for trigger and readout algorithms

- **Baseline use of ZCU9 Programmable Logic:**
  - Combine global TOB fragments into Global TOBs.
  - Transmit TOBs to L1Topo at 40 MHz.
  - Read out to FELIX on L1A.
- **Plans for on-board trigger-level analysis**
  - PL firmware and on-board Linux OS can increase the DAQ flexibility and capacity in data collection and processing
  - **Could do "real" analysis** (e.g. invariant mass calculations) **for physics-oriented monitoring or calibration-style computations** (e.g. di-jet balance). Depending on the proposed application, these could provide additional information to the trigger.
  - **Proof of principle**: running a neural network (ResNet-50) on the ZCU for classification of images (from the internet):
    Image processing at the level of O(ms), expected to decrease to O(µs) for jet network with 30 x 30 "images" (i.e. gFEX events).
- **Multi-processor system opportunities**
  - Low-power ARM Mali-400 **GPU** may aid in vector calculations, e.g. matrix multiplication for calorimeter image processing.
  - **Real-time processor** for trigger-level analysis (see above), real-time calibration, and object-level monitoring …



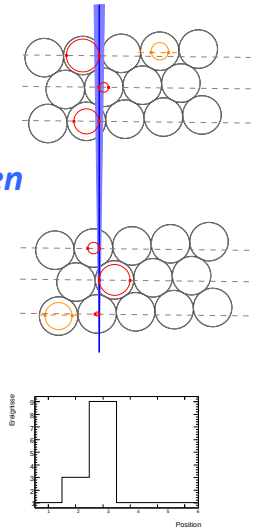


Could do some real physics algorithms with SoC/PS

ATLAS Muon MDT Trigger for HL-LHC

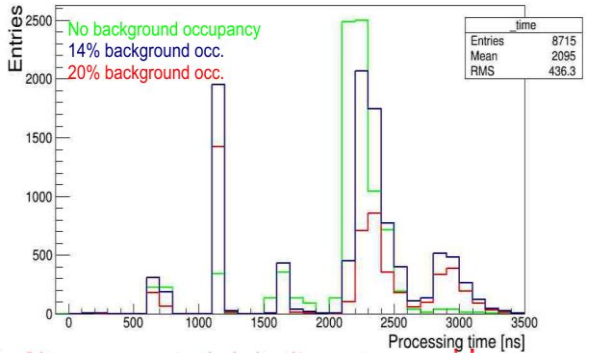
→ Use of Muon Drift Tubes for Trigger at the HL-LHC

MPI Physik, München,
Technische Universität München



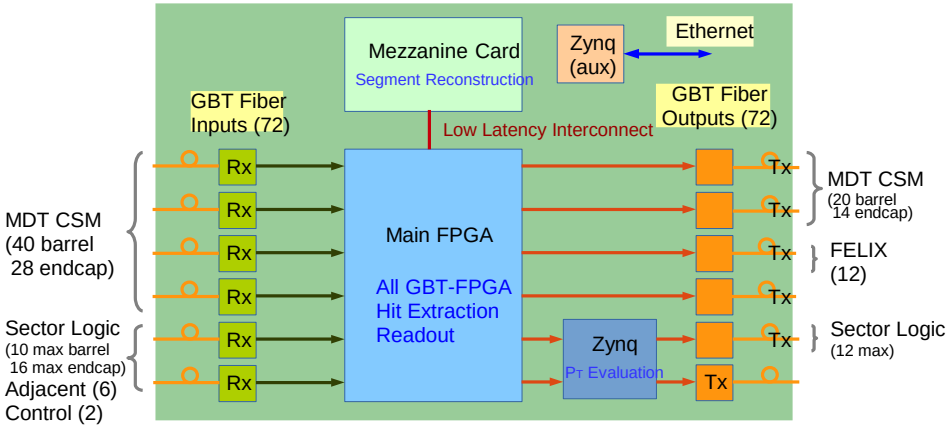
Proof of Principle:
Demonstrator Hardware: Xilinx Evaluation Board ZC706
Zynq-7045, with dual ARM Cortex-A9 @ 800 MHz

- **Processor System:**
  - Track segment reconstruction
  - Based on histogramming (Hough Transformation)
  - ⇒ **The whole processing can be done in ≲ 3 µs**
- **Transfer from Programmable Logic to Processor System:**
  - Using AXI protocol with interrupts:
  - ⇒ **Latency of ~ 0.5 µs**



Follow up:
Move track segment reconstruction to PL (VHDL):
The processing can be done faster than on the PS due to parallelization.

However, there is a plan to investigate if the PS can be used for p_T evaluation or (almost) complete track finding.



ATLAS - MUCTPI

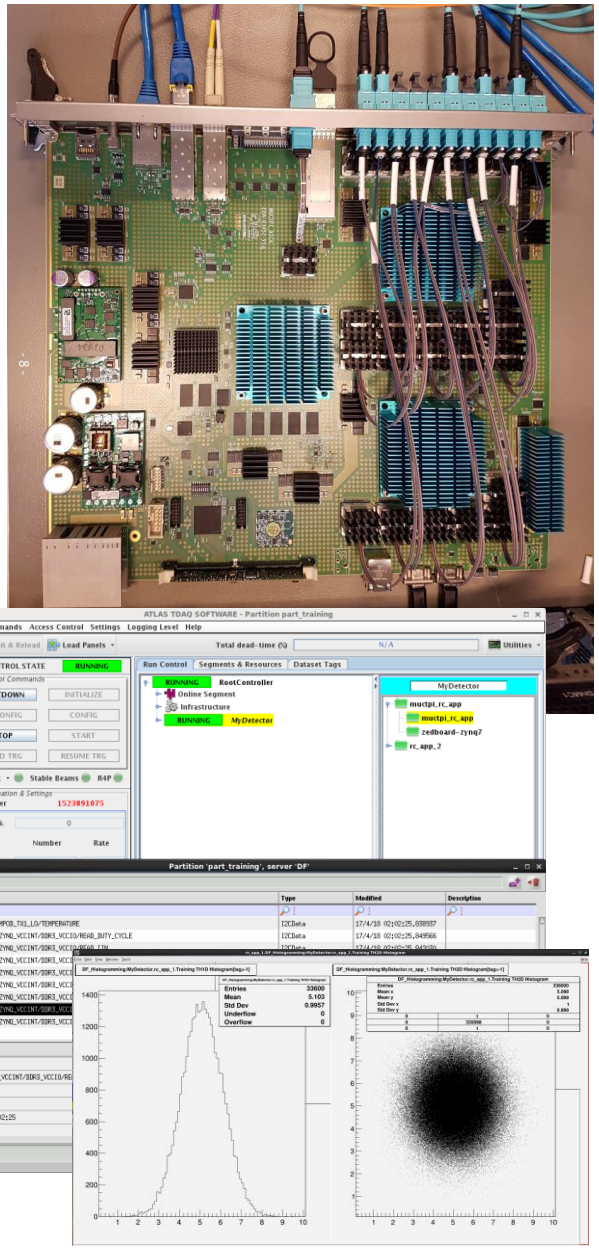
Muon-to-Central-Trigger-Processor Interface

- **Functionality of the MUCTPI:**
  - Data concentration of 208 muon trigger sector inputs.
  - Overlap removal, i.e. avoid double counting of single muons.
  - Provide muon trigger objects to topological trigger and muon trigger object multiplicity to Central Trigger Processor.
- **Use of the SoC:** configuration, control and monitoring of the board:
  - **Hardware control** of clock, power, optical modules + configuration of FPGAs.
  - **Run control** of processing FPGAs: r/w status/control registers and memories/LUTs.
- **Operating System:** Linux built using Yocto and Xilinx meta layer
- **Interactive access and tools:**
  ssh, peek&poke, Xilinx Virtual Cable, FPGA configuration, IBERT test programs, etc.
- **Application software:**
  Port of ATLAS TDAQ to Zynq: run control application + ROOT on Zynq

  Current prototype with Zynq SoC, soon have a new prototype with Zynq Ulrascale+ MPSoC:

  Succeeded to use a ZynqMP (ZCU102) with CentOS root file system and compile run control application + ROOT natively.

Run control application running on the SoC



ATLAS - DCS

Use the SoC to run a "Slow Control Application" (SCADA)

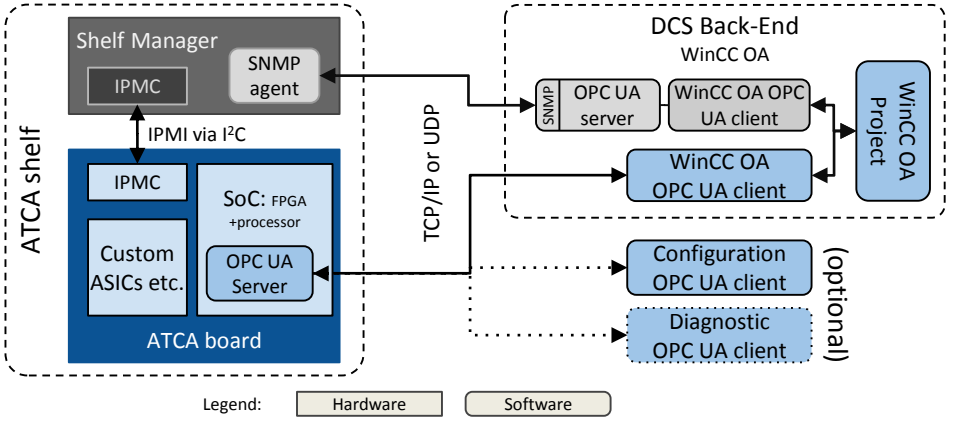
Detector Control System –

Monitor and control ATCA electronics:

- **Control of shelves via shelf manager SNMP interface (Pigeon Point)**
- **Complementary control via SoCs of each blade:**

  Monitor blade health, power control and reset signals.

  No IPMB or shelf manager bottlenecks, no dependency on proprietary software.



Software – OPC UA* based interface:

- **Technology at the basis of CERN JCOP framework**
- **OPC UA server on blade SoC:**

  Industry standard with many commercial and open-source implementations on various platforms.

  Runs on embedded Linux.

  Network integration using Ethernet/TCP.

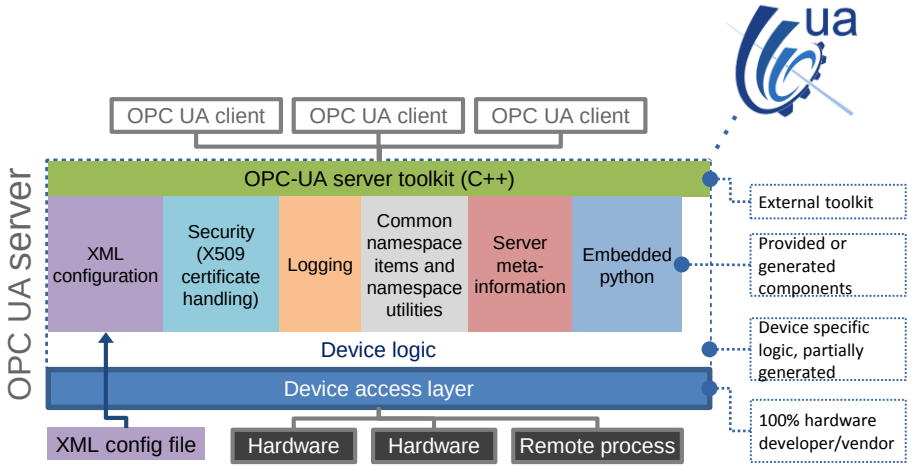
- **Build servers with *quasar*** framework:**

  Makes OPC UA transparent to developer.

  Reduces s/w development (C++, python) significantly by generating code from xml device model and providing useful components.

  Build environment for several platforms (x64 Win+Linux, ARM).

  Client software components to ease SCADA or run control integration.



Run it successfully on a ZCU102 (ZynqMP) using CentOS (MUCTPI)!

*Open Platform Communications Unified Architecture

**Quick OPC UA Server generAtion fRamework

Other projects (very briefly …)

- **CMS Barrel Calorimeter Processor:**
  - Common hardware platform for electromagnetic (EB) and hadronic (HB) calorimeters in the barrel region of CMS.
  - Will use Embedded Linux Mezzanine (ELM).
- **ATLAS Level-0 Muon MDT Trigger Processor:**
  - Plan to investigate if the SoC PS can be used for p_T evaluation or (almost) complete track finding.
- **ATLAS Pixel Data Trigger Controller:**
  - Similar to Level-0 MDT Trigger Processor.
- **CMS HL-LHC Track Trigger:**
  - YUGE (Your Ultrascale Gigabit Evaluator): for evaluation of Ultrascale+ FPGAs and 25 GB/s links.
- **ATLAS TileCal Rear-Extension Module (TREX): VME RTM card, control of hardware**
- **CLICdp: Zynq-based versatile DAQ system for test beam (CaRIBOu)**
  - For testing and characterizing pixel detectors for future CLIC detector (CLICPix) and ATLAS ITK.
- **CERN Accelerator Beam Transfer:**
  - Zynq-based FMC carrier board for Fast Interlocks Detection System.

…

SoCs also used in non-LHC experiments and beam projects

Interest Group & Workshop

- ***Interest group*** ***"System-on-Chip for Electronics":***
  - → ***Mailing list: system-on-chip@cern.ch***
  - → ***Twiki page: http://twiki.cern.ch/SystemOnChip***

- ***Workshop: "System on Chip", 12- 14 June at CERN***
  - → ***Indico registration: https://indico.cern.ch/event/799275***

Summary (1)

There is a lot of commonality of the hardware:

- Almost all projects use or are intending to use
  - Xilinx Zynq SoC, based on ARMv7 Cortex-A9 (32-bit) processor architecture
  - Xilinx Zynq UltraScale+ MPSoC, based ARMv8 Cortex-A53 (64-bit) processor architecture
- Some interest in Intel Stratix 10 SoC:

  ⇒ **Also based on ARMv8 Cortex-A53 (64-bit) processor architecture**

Advantages of SoCs are:

- Provide an efficient, low-latency way to interface hardware and software:
  - Use peripherals, like I2C, SPI, JTAG, etc.
  - Link to other FPGAs via Xilinx AXI Chip2Chip Links
  - Module designers are used to work this way
- Allow one to run a set of low-level test tools:
  - Xilinx Virtual Cable (XVC), IBERT test programs, FPGA configuration, etc.

⇒ ***Consider using an evaluation board: very useful for early testing.***

Summary (2)

Operating system:

- Different solutions: All are open-source! Many are based on Linux.
- ***Promising : use CentOS/aarch64 as common platform***
  (at least for those that do not need real-time features):
  - The kernel is hardware-specific, because of the drivers IP cores in the PL.
  - The decoupling is achieved at the level of the system libraries, i.e. glibc etc.
  - The application software is built on top of CentOS (and system libraries).

  ⇒ ***Consider using a 64-bit SoC.***

Different uses of software on the SoC:

- **"Dumb" device**: mere protocol converter, e.g. Remote Procedure Call (RPC), almost like firmware, vs.
- **"Smart" device**: make use of (a lot of) application software, run directly on the ATCA blade

Other issues to be considered:

- **System administration aspects**: in particular for systems with many SoCs, O(100):
  network organisation, software maintenance, security updates.
  ⇒ ***One of the main topics of the SoC Workshop in June at CERN.***
- **Long-term software support** (>10 years)?
  ⇒ ***Consider using SoC on a mezzanine card.***

Conclusion

- **System-on-Chips are very popular, versatile, and powerful devices.**
- **System-on-Chips are at the very interface between hardware and software.**
- **System-on-Chips are here to stay …**

Build intelligent control into the electronics modules!