Zynq-based Run Control for the ATLAS MUCTPI Upgrade

- Introduction
- Remote-Procedure-Call (RPC)-like Approach
- ATLAS TDAQ Run Control Application
- Outlook
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 (CTP)

• **Use of the SoC:**
  Configuration, control and monitoring of the board:
  - Hardware control of clock, power, and optical modules and configuration of FPGAs
  - Run control of processing FPGAs: read/write status/control registers and memories/LUTs
  - Current prototype with Zynq SoC, build a new prototype with Zynq Ultrascale+ MPSoC
An alternative model: System-on-Chip

**System-on-Chip:**
Processor system + Programmable logic, “CPU and FPGA”

→ Xilinx Zynq:

- **Zynq SoC:** ARM v7 (32-bit, 2 cores)
- **Zynq Ultrascale+ MPSoC:** ARM v8 (64-bit, 4 cores)

- **Processor system (PS):** like CPU, multi-core ARM processors with memory and peripherals, e.g. GbEthernet, I2C, etc., can be run “bare-metal” with Xilinx software or using an operating system, e.g. Linux

- **Programmable logic (PL):** like FPGA, can interface to the processor FPGAs, using Xilinx AXI Chip-to-Chip protocol
Operating System

Xilinx are providing Embedded Linux via the Yocto Project:
- Organised under the banner of the Linux Foundation
- Many hardware and software companies (~ 60) are members

Yocto Project Provides:
- Open-source software
- Complete build environment for different CPU architectures:
  - based on recipes: fetch sources, unpack, patch, configure, compile&link, package
  - recipes can derive from a class (super-recipe)
  - multiple recipes are organised in a layer
- A lot of software, including a reference distribution with a full Linux standard base:
  - ssh, NFS, python, command-line tools, etc.

Xilinx Layer Provides:
- Recipes to build toolchain for ARM: compiler, linker, system libraries (gcc)
- Kernel sources (from Xilinx) + Xilinx drivers
- Recipes to build kernel, device tree (from Vivado), rootfs, and bootloader (U-Boot)

Control application software:
- Two alternative approaches: RemoteBus and Run Control Application
Approach #1: RemoteBus (L1CT)

The MUCTPI uses a “System on a Chip” (SoC) running embedded Linux and application software to respond to requests like Remote Procedure Call (RPC) ⇒ decouple low-level access to processing FPGAs and hardware from high-level run control application

• Use a client-server and request-response approach, TCP, and a 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. read/write 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 framework for building RemoteBus software
MuctpiModule API

Implementation:

- Two base classes, **Client** and **Server**, implemented for communication between any two computers

- Two derived classes, **MuctpiModuleClient** and **MuctpiModuleServer**, were implemented, requests were added for the MUCTPI hardware and FPGA memories: read/write from FPGAs (+ block), read/write I2C, read/write SPI, etc.

- Use with **MuctpiModule** Application Programming Interface API Class (C++):
  - Encapsulates all hardware access, like, registers, memories, I2C, etc.
  - Hardware access can be automatically generated from a register description (.xml)
  - Extend the **MuctpiModule** API using virtual class which allow on to run
    a) **locally on the SoC**, using low-level software, or b) **remotely using RemoteBus**

⇒ **menu, test, and run control programs are transparent to the type of MuctpiModule class, i.e. local or RemoteBus**
The TDAQ run control application runs directly on the SoC
No intermediate PC, nor process, nor “protocol converter”
⇒ Need to port TDAQ software to ARM

• First go: build TDAQ run control software within Yocto framework:
  ⇒ Difficult, required many patches to TDAQ CMake

• Second go: use Yocto to provide toolchain (compiler + libs), and use CMake with toolchain for cross- compilation:
  ⇒ Works much better, only few changes to CMake files, some put into TDAQ, some into top-level CMake file

• Software ported (technical student project):
  − Needed some external packages from WLCG (e.g. Boost, TBB): could be built directly in Yocto
  − Needed to build ~40 packages from ATLAS TDAQ software: no modification to sources, few changes to CMake
    (some will be included by TDAQ, some can be written in a dedicated top-level CMake file)
  − Build additional software from WLCG: ROOT:
    not really made for cross-compilation (yet); but works, and we are discussing with the developers to include cross compilation in the build
Approach #2: Run Control Application

Result:

A working run control application running on the Zynq:

• responds to the run control state transitions
• reads configuration data
• provides monitoring information to Information Service (IS)

• provides ROOT histograms to Online Histogram (OH) service
Outlook

Use of CentOS:

→ *It might help if we can agree on common platform, e.g. CentOS/ARM*

→ *CentOS is gaining speed in the ARM user community, it is being used at CERN’s techlab*

→ *For the Zynq, one could deploy CentOS/ARM (or a stripped down version):*
  - *The kernel is hardware-specific* (for the Xilinx Zynq), because of the drivers and the device tree required by Xilinx and/or custom IP cores in the PL and/or required by different Vivado versions
  - *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)

⇒ *ATLAS AFP ROD (SLAC) have started investigating this approach,*

ATLAS MUCTPI (CERN) are also interested:

Porting of run control application to ARM could become much simpler if used with CentOS, could possibly build run control application natively on CentOS/ARM and use it from NFS, and it could become even more simpler if LCG and TDAQ were built for CentOS/ARM
Yesterday at ACES 2018 workshop, I suggested to create an interest group:

“System-on-Chip for Electronics”

⇒ Mailing list: system-on-chip@cern.ch

⇒ Exchange information, suggestions, questions, ...
⇒ Could organise regular meetings (1-2 times per year?) ...
Build intelligent control into the ATCA blade!