The CMS APx Platform Using SoC Devices

T. Gorski for the APx Engineering Team
12-June-2019
The APx Consortium

- Pooling of efforts in ATCA Processor hardware, firmware and software development
- Multiple ATCA processors and mezzanine board types
- Modular design philosophy, emphasis on platform solutions with flexibility and expandability
- Reusable circuit, firmware and software elements
- APx application areas in CMS Phase 2 Upgrade: Barrel Calorimeter, Muons, Trigger
• Built:
  • Controller Dev. Board
  • APx Demo #1 (VU9P)

• In Design:
  • APx Test Hub (KU9P)
  • Barrel Calo Proc. Demo. (KU115)
CMS Phase 1 Upgrade (2015)

- 18 CTP7 Card Install (MicroTCA)
  - Part of Calorimeter Trigger upgrade
- XC7V690T FPGA + XC7Z045 SoC
- First CMS install with ZYNQ devices
  - PetaLinux distribution
- Starting point for APx SoC architecture
Custom APx SoC Boards

**ZYNQ-IPMC**

- XC7Z014S/XC7Z020 device
- 244-pin DIMM form factor
- +3.3VMP supply
- 109 PL IOs, 16 ADC channels
- 256MB DDR3

**Embedded Linux Mezzanine (ELM)**

- ELM1: XC7045 device
- 84mm × 75mm form factor
- +12V Supply
- SD/QSPI primary/recovery boot options (IPMC-selectable)
- PL IOs: 24+ @ 3.3V, 74 @ 1.8V
- MGTs: 8
- 512MB DDR3 Memory
ZYNQ SoCs with configurable PL are an excellent fit for diversity of hardware connection types needed by APx ATCA cards.
IPMC SoC HW Context—Power and Sensors

IPMC operates on +3.3V ATCA management power, ELM operates on payload power under IPMC control.
“Traditional” PICMG IPMC function with enhancements for fault monitoring, payload power control, plus JTAG and ELM connectivity

Custom IPMC HW/SW/FW, ZYNQ-7000 based

To be released as open-source HW/FW/SW project

C++/FreeRTOS runtime environment

PL is customized to specific ATCA application

Current status: pre-release operation, APd1 image

Links to existing IPMC talks:

- [https://indico.cern.ch/event/791616/contributions/3399058/attachments/1842549/3021715/xTCA14_ZYNQIPMCUpdate_mpv_v1.pdf](https://indico.cern.ch/event/791616/contributions/3399058/ attachments/1842549/3021715/xTCA14_ZYNQIPMCUpdate_mpv_v1.pdf)
- [https://indico.cern.ch/event/737733/contributions/3156538/attachments/1732685/2801036/IPMCWorkshop_OpenSourceIPMC Solution_mpv_v2.pdf](https://indico.cern.ch/event/737733/contributions/3156538/ attachments/1732685/2801036/IPMCWorkshop_OpenSourceIPMC Solution_mpv_v2.pdf)
ELM in context when payload power to the main board is active. ELM has primary responsibility for payload operation.
### Embedded Linux Mezzanine (ELM)

- Mezzanine form factor to facilitate use across APx ATCA board family
- ELM1: XC7Z045-based
- MGT connections for 10GbE, AXI bridge, Solid State Disks, etc.
- PL-based IP library and custom peripherals support main board operation
- PL peripherals tailored to specific ATCA board requirements
- Key ELM functions in APx use:
  - 1GbE and 10GbE endpoint
  - FPGA bitstream loading and register/memory IO (via AXI bridge)
  - Support device configuration & monitoring (e.g. Refclk Synths, Fireflys, etc.)
- ELM2: XCZU4/5CG-based (in PCB layout)
ELM 10GbE

- 10GbE implemented in ELM SoC PL section

- Use case examples:
  - Deep memory operations (e.g., R/W to large external memory attached to FPGAs)
  - SpyDAQ path (e.g., zero-bias, non-triggered data)

- Using Ethernet stack from SLAC
  - Part of SLAC Ultimate RTL Framework (SURF)
    - [https://github.com/slaclab/surf/](https://github.com/slaclab/surf/)
  - Overview of SURF’s Ethernet Library
  - Firmware-based 5-layer Ethernet stack (PHY, MAC, IPv4, UDP, RSSI)
    - Reliable SLAC Streaming Interface (RSSI) engine is the 5th Ethernet layer and is the reliable communications layer based upon RUDP (Cisco implementation: refer to RFC-908 and RFC-1151)
    - [SLAC’s RSSI Documentation Homepage](#)
ELM 10GbE Block Diagram
APx Bitstream Loading

- APx FPGA config interface: Slave Serial (no FPGA config Flash)
- Config master is custom firmware core in ELM PL (see XAPP583)
- VU9P performance example: uncompressed 76.4MB bitstream in 7 seconds with 100 MHz config clock
- Post load operation: initialize AXI link between ELM and FPGA (Aurora MGT-based)
- Option to load bitstreams over Ethernet (either push or pull) or from local file system

```bash
root@linux:/mnt/persistent# ./fpgaLoader 0 vu9p_link_test.bit
Bitstream read in 0.028 seconds

Bitstream downloaded in 6.995 seconds
FPGA bitstream programmed successfully in 7.025 seconds.
root@linux:/mnt/persistent# ```
APx FPGA Image Bundles

- Single file with related components needed to run the FPGAs
  - FPGA Bitstream
  - Remote Procedure Call (RPC) Service Modules
  - Specialized shell scripts, other ancillary files

- Easy to store, transfer and ‘activate’

- Load all components simultaneously to improve operational consistency and prevent confusion
  - Suitable both for development and production use

- Checksummed zip file or similar, unpacked and activated in single step with `fpgaload` command
APx Remote Procedure Calls (RPC)

- Runtime loadable modules for implementing higher level control functions on FPGAs
- Motivation: keep functions in the relevant layer
  - Subsystem level code in control PC
  - Card-specific reg and memory IO in the ELM SoC
  - Clean API boundary definitions can decouple software/firmware updates
- Design:
  - Familiar call-response interface
  - Client libraries for multiple languages
  - Custom pluggable service modules on cards
- CTP7-era Implementation: custom interface over ProtoBuf/TCP
- ELM-era Implementation: gRPC? msgpack & ZeroMQ? Other? (evaluating options)
User applications need access to device memory

Goals: easy to use, efficient, safe (no corruption from pointer problems, etc.)

Two key system calls: open and mmap

CTP7 era: “memsvc” library
- Uses /dev/mem (raw access to full system memory!)
- Library checks for valid device mapping
- Library provides bus error handling in read/write wrappers (e.g., FPGA not loaded)

APx (ELM) era: Linux Userspace I/O
- Uses /dev/uio* (direct access to individual devices)
- Kernel checks for valid mappings
- Optional library for bus error handling (expensive operation)
Geographic IP Address Assignment

- APx IP Address Assignment Objectives:
  1. **Geographic** – address with the crate/slot, not the card
  2. **Fully automatic** – hot swap without table edits
  3. **Supports multiple endpoints/card** (e.g. IPMC, ELM 1GbE/10GbE)
  4. **DHCP-based** – use established infrastructure to max extent possible

- Prefer DHCP Client IDs over MAC addresses

- Identified mechanism for generating **Geographic Client IDs**:
  - IPMC gets crate/backplane FRU data via Shelf Manager
  - IPMC provides hardware address (slot ID)
  - Crate/Slot info passed to ELM via Service Channel (ELM – IPMC UART link)

- Geographic Client IDs contain 3 pieces of geographic information: **crate, slot, component**. (e.g. “atca-s2e10-46-84-ipmc”)
  - First two items sourced through the crate/card IPMI infrastructure
  - Exact format not determined, we anticipate a standard format will arise
ATCA FPGA/SoC boards represent a complex chain of OS config, FPGA bitstreams and application packages, all working together.

**PetaLinux Distribution:**
- Works very well on the CTP7, still in CMS operation for Run3
- Single-file images (one file contains FW, Kernel, SW), allowing atomic updates and consistent component versions
- Easy upgrade & mgmt., but cumbersome to bake in all apps
- Reboot returns the card to a consistent state every time – zero drift

**CentOS Distribution:**
- Much larger collection of pre-built libraries
- Easy to install individual applications and update configuration in development
- Parts are independent—config and version drift in or between cards can easily occur—no “one reboot” remedy
- Tools and techniques can address config challenges and drift issues, but none are perfect (puppet, PXE, yum, others)
APx is family of ATCA FPGA-based processor cards for CMS Phase 2

Dual SoC model
- ZYNQ-IPMC for power control and sensor monitoring
- ELM (Linux) for direct payload (FPGA) support

Utilizes a set of proven embedded control concepts developed for CMS Phase 1 upgrade (RPC, FPGA loading, etc.), evolving for Phase 2 use

Interested in studying the work of other SoC groups as we refine the APx implementation model