Apollo Platform

SoC Integration

D. Gastler on behalf of the APOLLO Platform team

June 13, 2019
OUTLINE

1. MOTIVATION
2. HARDWARE
3. COMMUNICATION
4. ZYNQ SOC
Motivation

We are tasked with contributing to multiple ATCA Blade projects:

- L0MDT (ATLAS)
- Pixel DTC (CMS)
- OT Track Finder (CMS)
- others

Each of these projects are different in the details, but will all need common blade services:

- ATCA Zone1 power interface
- DC-DC converters
- IPMC for shelf negotiations
- Ethernet (+switch) for network access
- Some type of SoC for configuration and slow control.

Our plan is to use a two part blade platform named “APOLLO”.
“Service Module (SM)“:
- 400W power
  (12V to Command Module)
- SoC Zynq
  (70xx demo; US+ final)
- CERN IPMC
- Wisconsin ESM Ethernet Switch

“Command Module (CM)“:
- FPGAs / Zynqs
- Fiber transceivers
- DC/DC+POLs
- MUX logic for I2C
Common ATCA Platform “APOLLO”
CURRENT HARDWARE STATUS
Hardware Status

Service Module
- Boards are back and under testing.
- IPMC, Power, SoC working in shelf.
- No show-stoppers found.

Command Module
- CMS (Cornell, pictured above): Prototype passed tests at Cornell. Board shipped to BU for SM validation
SM / CM Communication
Ethernet
- UW Ethernet Switch
- ATCA, FP RJ45, IPMC, & Zynq
- Independent Zynq RJ45

JTAG
- Normal:
  IPMC ⇒ Zynq
  Zynq ⇒ CM(s)
- One-Chain:
  IPMC ⇒ Zynq+CM(s)

Slow control (async-serial)
- CM(s)/Zynq via muxed IPMC
- Direct from Zynq to CM(s)

High speed links
Sensors
Register access:
- Two MGTs per CM (4 if 1 CM)
- AXI/PCIe (baseline is AXI)
- AXI C2C extends bus from Zynq to CM FPGAs

Ethernet:
- Zynq to ESM via SGMII

CMS TCDS:
- TCDS to Zynq/CM 10 Gbps
- CM(s) TTS++ merging on Zynq
- Fake TTC++ from Zynq to CM(s)
- Local (Si5344) and Backplane clock sources
One specific implementation

Many $I^2C$ devices:
- Temp sensors
- Clock synthesizers
- Power supply components
- Optical transceivers
- FPGAs (die temp, voltages etc)

Readout:
- IPMC: (via SHM) uC curated sensors via $I^2C$ FPGA monitoring via Zynq $I^2C$.
- Zynq: (via ETHERNET) uC cached $I^2C$ sensors via serial FPGA monitoring via AXI (C2C)
Zynq SoC
Zynq SoC

2019 Prototype: ZX1
- 7-series Zynq: XC7Z035 or XC7Z045
- 64 x 54 mm
- Dual-core 32-bit ARM Cortex-A9
- 1GByte RAM
- 8 MGTs (6-10Gbps)

Future versions:
- UltraScale+ Zynq
- Smaller form factor if possible
- 64-bit ARM
- 4GByte RAM
- 8 MGTs
Currently using PetaLinux in bring-up & testing.

A common supported OS would be nice, but the details matter

**Benefits:**
- We don’t have to roll our own
- Updates prepared by someone else
- Expect most software to “just work”

**Drawbacks:**
- Expect to have to abide by many rules to avoid conflict
- Worry about forced updates breaking the board
- PS-PL interface? DeviceTree building?
- Loss of control...

Hopefully this workshop can give a better idea of what this would look like...
SoC: Use and Access

Configuration & Debugging:
- The Zynq will be the path for upgrading FW/SW on the Zynq, CM FPGAs, CM uCs, & other front-ends
- Debugging will use XVC JTAG(s).

Software:
- We expect other users will want to run code on the SoC.
- Initially, we expect these users will run servers and access HW via some API.
- The use of user containers may simplify things, but we don't have the experience yet

Access:
- The SM provides two network connections to the Zynq:
  - FP RJ45 / ATCA networks via switch
  - Rear RJ45 jack directly to Zynq.
- We will require ssh access (with sudo) to the Zynq (via multiple jumps is OK)
- We will require access to ports for XVC, static HTTP, etc (ssh tunneling is OK)
- We will require access to the IPMC for XVC and other services.
Local and remote AXI slaves access via Xilinx AXI & Chip2Chip IP cores

- Slaves are memory-mapped to CPU address space
- MM’d slaves accessed in userspace via the UIO device driver
- Extend the uHAL software, xml address table & API to map AXI slaves UIO(s)
- AXI Slave mapped to MSB(s) of uHAL address.
- Reuse of existing featureful uHAL code with AXI Xilinx IP.
We hope to make full use of both the Zynq PL and PS for serious real-time applications

**Zynq PL (FPGA):**
- e.g. Histogramming prescaled subset of data
- e.g. Pixel threshold calibration
- e.g. Local DAQ (prescaled) with PL filtering

**Zynq PS:**
- We would like the ability to use one of the Zynq cores in RTOS or uC mode. Other user’s CPU needs are unknown. We want resources reserved for real-time
  - Capturing wide range of counters on set intervals
  - Collecting, processing and presenting data (static HTML)
  - Similar real-time access of PL real-time processing for presentation
The “APOLLO” Platform provides flexible ATCA blades where the common services and application specific parts are on separate boards.

Prototype Service and Command Modules have been delivered and are under test.

Service Modules have customizable IPMC & Zynq sensor paths to send monitoring where it is needed.

Use of Zynq SoC Mezzanine running PetaLinux(+RTOS) for configuration, monitoring, and debugging.
BACKUP
Prototypes due soon. Will validate design with CMS prototype CM Plan identical modules for ATLAS L0MDT and CMS tracker
Why not a single PCB?

*This could be done in the future. We believe it is a poor choice now because...*

- Two regions with different requirements and engineering challenges
- Different PCB technologies needed
- Allows splitting design between groups