X2O ATCA board control solution

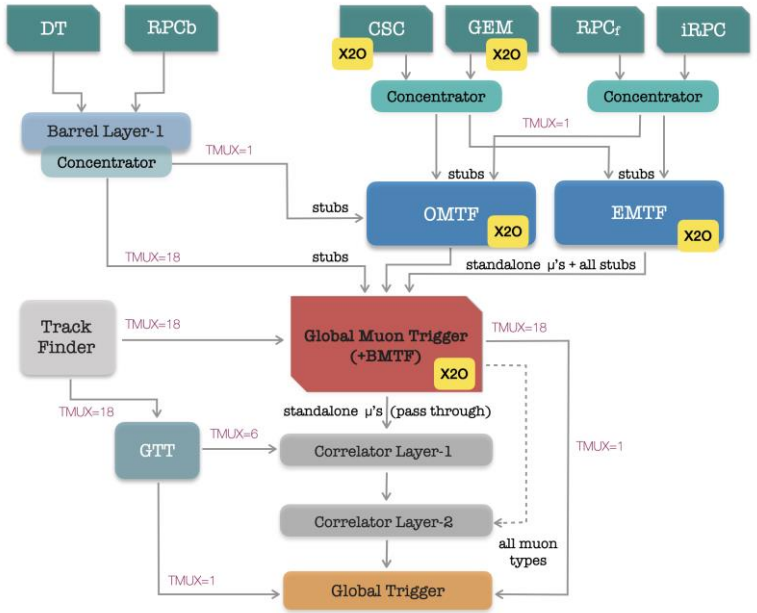
Aleksei Greshilov (UF)
Alexander Madorsky (UF)
Michail Bachtis (UCLA)
Jacobo Konigsberg (UF)
Darin Acosta (Rice)

2021-11-23

Phase-2 Upgrade

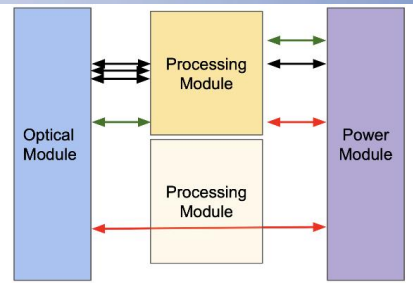
This presentation describes the control system for X2O platform that will be applied for upcoming Phase-2 Upgrade of L1 Trigger subsystems:

- EMTF
- OMTF
- GEM
- GMT
- CSC (DAQ system upgrade)



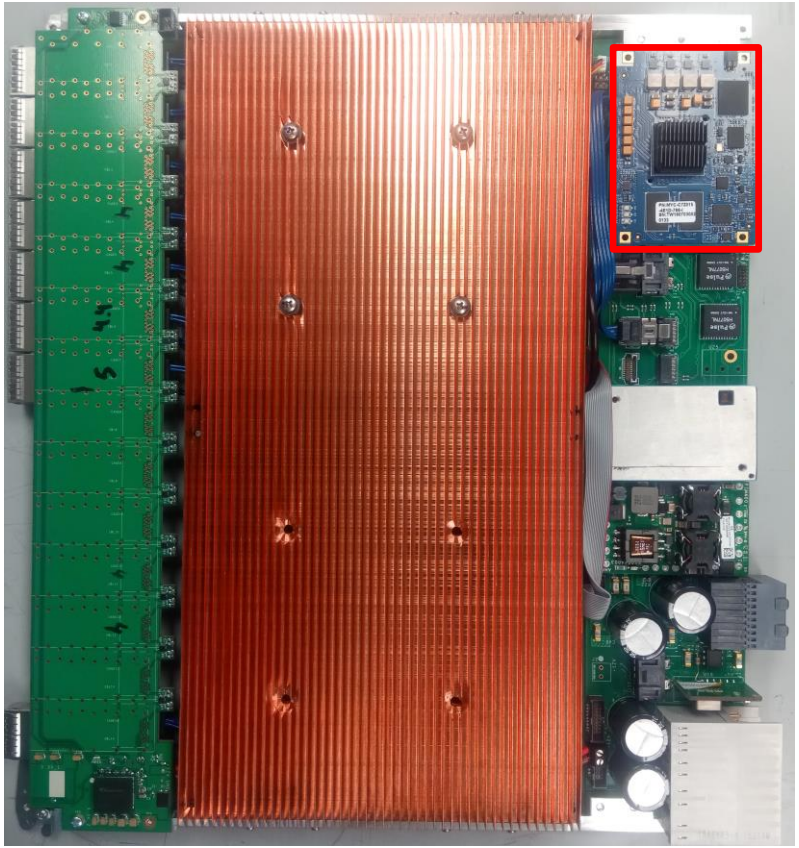
The X2O Platform

- Modular board design allows for different applications
  - Heat-sink mounted [pic]
- Power module
  - ZYNQ 7000 for control, integrated IPMC functions -- UF
- Optical modules: two types
  - w/ 30 QSFP cages, 120 links -- UF
  - w/ 15 QSFP-DD cages, 120 links -- UCLA
  - For high I/O tasks w/ lighter FPGA needs
- Processing modules (max 2 per/blade):
  - Xilinx Kintex KU15P (1760) -- UF
    - High I/O tasks, lighter FPGA
    - e.g. data concentration, detector BE's
  - Xilinx Ultrascale VU13P (A2577) -- UCLA
    - Large resource applications
    - w/ QSFP-DD optics (w/ QSFP works as well)
    - For the Global Muon Trigger (GMT), EMTF and OMTF

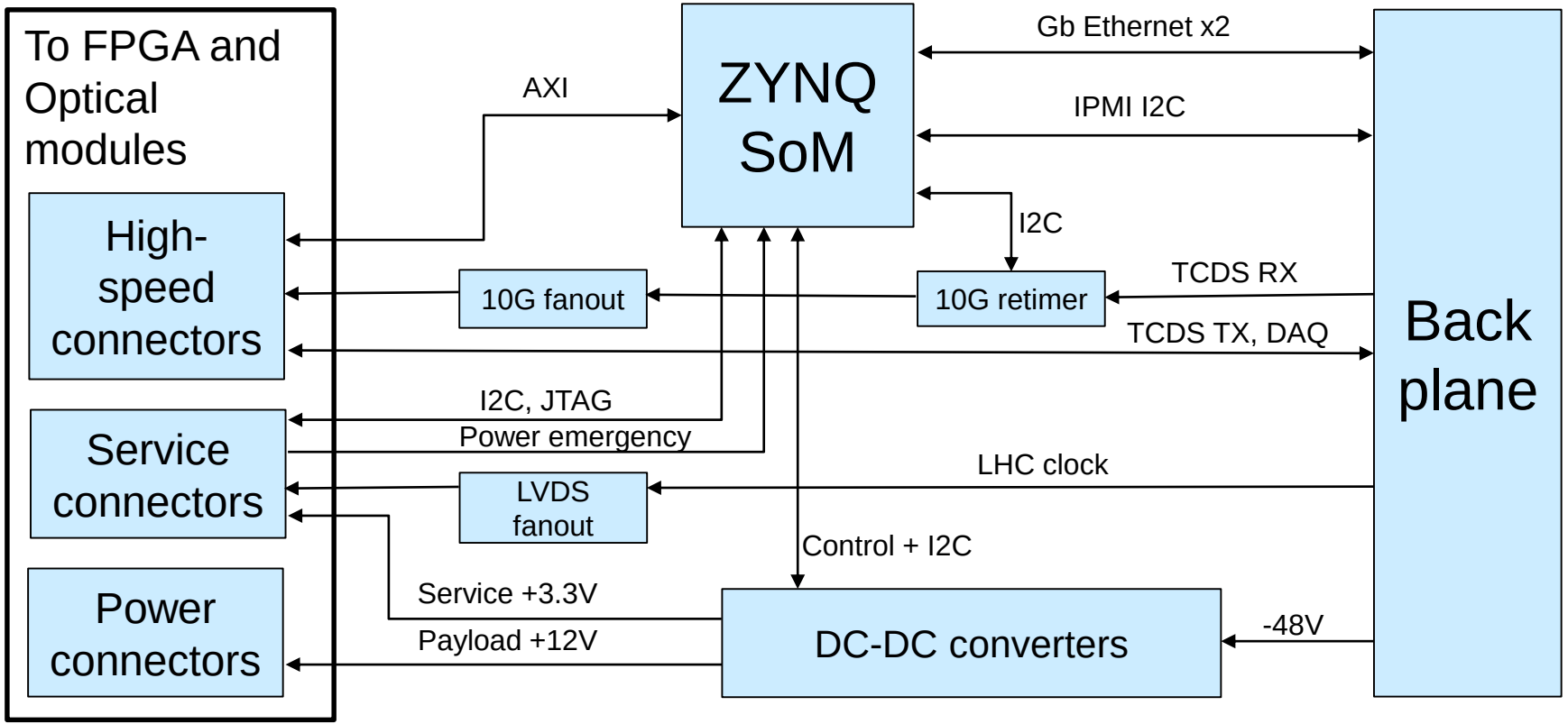


Hardware platform (X2O - Zynq)

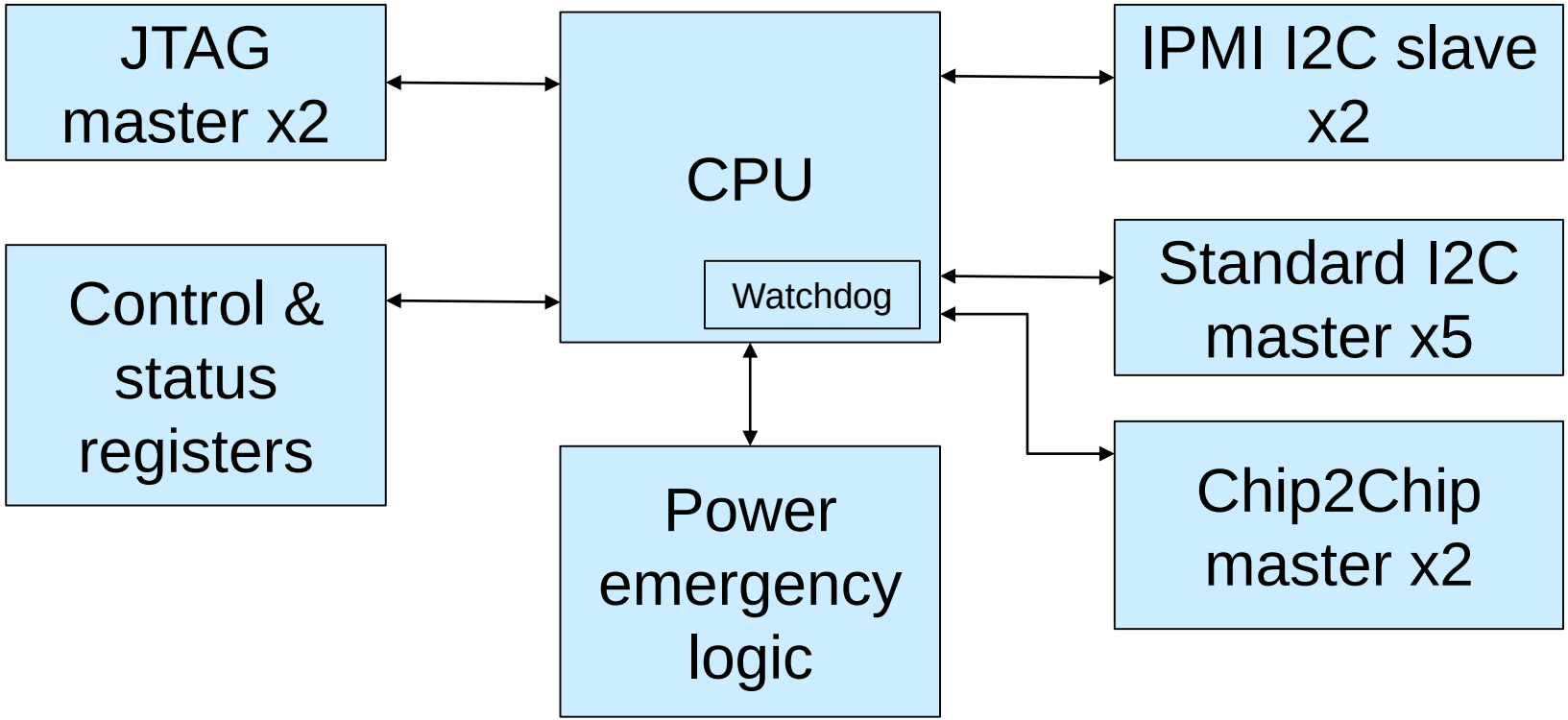
- Device used as control interface and IPMC host:
  - MYC-C7Z015
  - Inexpensive off-the-shelf ZYNQ mezzanine (~$200)
  - RAM: 1 GB
  - Ethernet interfaces
    - One built-in
    - Second as external PHY on host board
  - EMMC: 4 GB
  - QSPI flash: 32 MB
  - SD card on the host board
  - 4 MGT channels for AXI interfaces to payload modules
    - 3.75 Gbps each
  - Occupies minimal PCB real estate



X2O Power Module block schematics



X2O Power Module ZYNQ firmware



X2O Power Module firmware[1] modules

IP Module	Description
IPMI I2C slave	Customized I2C slave transceiver capable of buffering long transfers. Does not pose any timing requirements to IPMC software.
Standard I2C master	Standard I2C master modules used to control I2C devices on Power, FPGA, and Optical modules.
Chip2Chip master	Regular Chip2Chip master, used to extend AXI bus to FPGA modules
JTAG master	JTAG masters are used for firmware downloading and Xilinx Virtual Cable (XVC) debugging of the FPGA modules
Power emergency logic	Payload modules can signal emergency conditions (such as overvoltage) using dedicated lines. Emergency logic then shuts down Payload power in 10 ns or less
Watchdog	The dedicated watchdog hard IP is used to monitor the health of the IPMC software. If IPMC software stops polling the watchdog, the watchdog resets the CPU and shuts down the Payload power.

UF IPMC Software (Linux userspace)

UF IPMC[2] is an Intelligent Platform Management Controller (IPMC) that resides on ATCA boards (X2O) in an embedded ZYNQ[3] device. UF IPMC is responsible for the communication with Shelf Manager.

Built 32-bit ARM Embedded Linux system on ZYNQ module within X2O boards:

- Petalinux kernel version 2021.1
- CentOS 8

Main points:

- Derived from the coreIPM open-source project[4].
- Provides full basic functionality[5] for sensor monitoring.
- Provides easy customization of USER sensors
  - UCLA team has successfully implemented and tested custom Octopus board sensors.
- Supports the use of various sensor readout interfaces, including I2C

NOTE: **A separate IPMC module is not needed**.

Normal CPU usage of UF IPMC process is **2-3%**

```
8c: LUN: 0, Sensor # 9 ("T:QSFPDD")
    Type: Threshold (0x01), "Temperature" (0x01)
    Belongs to entity (0xa0, 0x60): FRU # 0
    Status: 0xc0
       All event messages enabled from this sensor
       Sensor scanning enabled
       Initial update completed
    Raw data: 31 (0x1f)
    Processed data: 31.000000 degrees C
    Current State Mask: 0x60
       At or Above Upper Non-Recoverable Threshold

8c: LUN: 0, Sensor # 7 ("T:2V7_I+ Rmt")
    Type: Threshold (0x01), "Temperature" (0x01)
    Belongs to entity (0xa0, 0x60): FRU # 0
    Status: 0xc0
       All event messages enabled from this sensor
       Sensor scanning enabled
       Initial update completed
    Raw data: 68 (0x44)
    Processed data: 68.000000 degrees C
    Current State Mask: 0x07
       At or Below Lower Non-Critical Threshold
       At or Below Lower Critical Threshold
       At or Below Lower Non-Recoverable Threshold

8c: LUN: 0, Sensor # 5 ("T:V+ Rmt")
    Type: Threshold (0x01), "Temperature" (0x01)
    Belongs to entity (0xa0, 0x60): FRU # 0
    Status: 0xc0
       All event messages enabled from this sensor
       Sensor scanning enabled
       Initial update completed
    Raw data: 29 (0x1d)
    Processed data: 29.000000 degrees C
    Current State Mask: 0x1e
       At or Below Lower Critical Threshold
       At or Below Lower Non-Recoverable Threshold
       At or Above Upper Non-Critical Threshold
       At or Above Upper Critical Threshold
```

ATCA shelf manager recognizing some of our temperature sensors

USER code implementation

- TEMPLATES are provided
- UF IPMC Manual is provided
- Separated USER code space from UF IPMC structure (for easy update)

Power ON

Power OFF

```
void user_module_payload_on( void )
{
  unsigned int payload_read;

  lock(1);
  payload_read = reg_read(devmem_ptr, qbv_on_off);
  payload_read |= 0x20;
  reg_write(devmem_ptr, qbv_on_off, payload_read);
  payload_timeout_init = !bolt;
  power_up_octopus(i2c_fd_snsr[1]);
  power_up_qsfpdd_module(i2c_fd_snsr[1]);
  logger("PAYLOAD", "On");
  power_up_done = 1;
  unlock(1);

}

void
user_module_payload_off( void )
{
  lock(1);
  unsigned int payload_read;
  payload_read = reg_read(devmem_ptr, qbv_on_off);
  payload_read &= ~0x20;
  power_down_octopus(i2c_fd_snsr[1]);
  power_down_qsfpdd_module(i2c_fd_snsr[1]);
  power_up_done = 0;
  reg_write(devmem_ptr, qbv_on_off, payload_read);
  logger("PAYLOAD", "Off");
  unlock(1);

}
```

Sensor enable

Upper non-recoverable threshold exceeded (Power OFF)

Upper critical threshold exceeded (Power OFF)

Upper non-critical threshold exceeded (Fans speed - UP)

Back to normal (Fans speed - DOWN)

Sensor disable

```
/*=============================================*/
/*  QSFPDD Temp Sensor                         */
/*=============================================*/
void read_sensor_temp_qsfpdd(void) {
  lock(1);

  // Wrapper parameters
  u8 i2c_ch = 0x01;

  // Sensor Data Record
  u8 sensor_N = 9;

  if (check_power_up()) {
      // Read the temp
      float temp_f = qsfpddTemperature(i2c_fd_snsr[i2c_ch]);

      // Convert float to byte and get precision
      u8 temp_b = (u8)(temp_f);

      sd[sensor_N].last_sensor_reading = temp_b;
      sd[sensor_N].sensor_scanning_enabled = 1;
      sd[sensor_N].event_messages_enabled = 1;
      sd[sensor_N].unavailable = 0;

      static int first_time = 1;
      static int up_noncrt_assert = 0;

      if (first_time) {
          first_time = 0;
      } else if (sd[sensor_N].last_sensor_reading >= sdr[sensor_N].upper_non_recoverable_threshold) {
          // Transition to M6 for non-recoverable
          unlock(1);
          picmg_m6_state(fru_inventory_cache[0].fru_dev_id);
          logger("WARNING","Non-recoverable threshold crossed for QSFPDD temperature sensor");
          lock(1);
      } else if (sd[sensor_N].last_sensor_reading >= sdr[sensor_N].upper_critical_threshold) {
          // Transition to M6 for upper critical
          unlock(1);
          picmg_m6_state(fru_inventory_cache[0].fru_dev_id);
          logger("WARNING","Critical threshold crossed for QSFPDD temperature sensor");
          lock(1);
      } else if (up_noncrt_assert == 0 && sd[sensor_N].last_sensor_reading >= sdr[sensor_N].upper_non_critical_threshold) {
          // Assertion message for shelf manager
          FRU_TEMPERATURE_EVENT_MSG_REQ msg;

          msg.command = 0x02;
          msg.evt_msg_rev = 0x04;
          msg.sensor_type = 0x01;
          msg.sensor_number = sensor_N;
          msg.evt_direction = 0x01;
          msg.evt_data2_qual = 0x01;
          msg.evt_data3_qual = 0x01;
          msg.evt_reason = 0x07;
          msg.temp_reading = sd[sensor_N].last_sensor_reading;
          msg.threshold = sdr[sensor_N].upper_non_critical_threshold;

          ipmi_send_event_req(( unsigned char * )&msg, sizeof(FRU_TEMPERATURE_EVENT_MSG_REQ), 0);
          up_noncrt_assert = 1;
      } else if (up_noncrt_assert == 1 && sd[sensor_N].last_sensor_reading < sdr[sensor_N].upper_non_critical_threshold) {
          // Deassertion message for shelf manager
          FRU_TEMPERATURE_EVENT_MSG_REQ msg;

          msg.command = 0x02;
          msg.evt_msg_rev = 0x04;
          msg.sensor_type = 0x01;
          msg.sensor_number = sensor_N;
          msg.evt_direction = 0x81;
          msg.evt_data2_qual = 0x01;
          msg.evt_data3_qual = 0x01;
          msg.evt_reason = 0x07;
          msg.temp_reading = sd[sensor_N].last_sensor_reading;
          msg.threshold = sdr[sensor_N].upper_non_critical_threshold;

          ipmi_send_event_req(( unsigned char * )&msg, sizeof(FRU_TEMPERATURE_EVENT_MSG_REQ), 0);
          up_noncrt_assert = 0;
      }
  } else {
      sd[sensor_N].last_sensor_reading = 0;
      sd[sensor_N].sensor_scanning_enabled = 0;
      sd[sensor_N].event_messages_enabled = 0;
      sd[sensor_N].unavailable = 1;
  }

  unlock(1);
}
```

Reaction on Power emergency

- Reaction time to shut down the payload actually matters in some failure on the payload.
- Quick shutdown required for the power emergency, specifically overvoltage.
- Power OK bit check is used for shutting down the payload power via firmware logic, without CPU participation. The reaction time can be as fast as a few nanoseconds.
- UF IPMC software is not involved in this process at all. UF IPMC should eventually let the Shelf Manager know that a failure took place, but there is no timing limitation.
- Shutting down the payload power as a result of the sensor readout via I2C (or another interface) is also possible. Reaction time in that case can be much longer than Power OK bit logic in firmware.

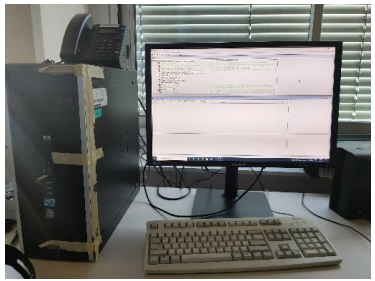
UF IPMC Polaris Compliance test results

UF IPMC tasks:

- Mandatory:
  - 58 (passed)
  - 17 (failed) – not critical
- Optional:
  - 18 (HPM.x) - not implemented
- Debug:
  - 30 (skipped) - not important

ELMA IPMC tasks:

- Mandatory:
  - 42 (passed)
  - 31 (failed)
- Optional:
  - 18 (HPM.x) - not implemented
- Debug:
  - 32 (skipped)

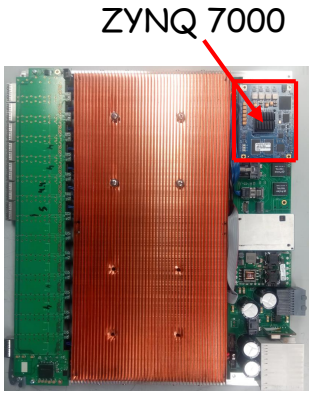


HOST with Polaris tester

IPMI LAN

ATCA Shelf

IPMB



X2O Board

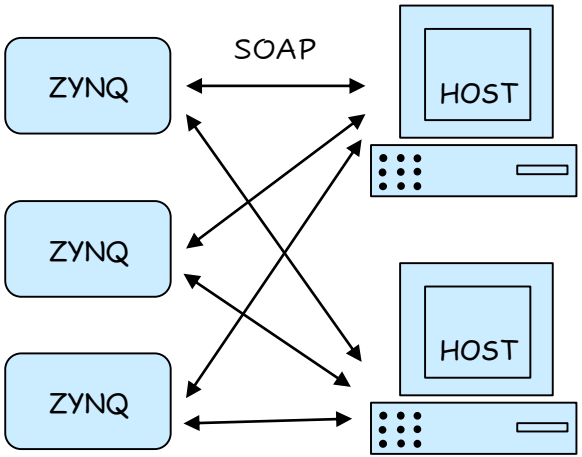
- UF IPMC implements all functionality necessary for normal operation of the device in the ATCA chassis. It was tested in three different ways:
  1. In the CMS-standard ATCA chassis
  2. In the COMTEL ATCA chassis
  3. On the Polaris Compliance test stand in CERN
- All errors detected by Polaris Compliance test stand are expected due to either implementation features, or non-critical functionality not currently implemented in the UF IPMC.
- **No unexplained errors have been detected. None of the detected errors preclude UF IPMC from normal operation in ATCA chassis.**

Thanks a lot to Julian Mendez (CERN EP-ESE) for the assistance with compliance tests!

XDAQ-core on ZYNQ - 7000 device

- XDAQ is the CMS framework and library for system configuration, control and monitoring
- Investigated using XDAQ on ZYNQ. Reasons:
  - Would like to use CMS-standard SOAP messaging for communication with ZYNQ devices
  - Minimal rework is needed in existing SWATCH cells and other software
- XDAQ was successfully compiled for ZYNQ-7000 device
  - Thanks for XDAQ team for a lot of help!
- SOAP communication tested successfully

PS: May use another solution depending on CMS developments



XDAQ process running on ZYNQ

```
[root@localhost xdaq_mgt_application]# ./mgtRun.sh
AsyncDNSResolver: fopen(/etc/resolv.conf) failed: No such file or directory
09 Dec 2020 01:02:08.629 [3020849168] WARN  root <> - Not able to resolve hostnames via DNS on host localhost
09 Dec 2020 01:02:08.631 [3020849168] INFO  localhost.p:33333 <> - xdaq version: 05.05.01, compiled on Nov 28 2020 at 02:22:54, 32bit
09 Dec 2020 01:02:08.633 [3020849168] WARN  localhost.p:33333 <> - cannot open file /etc/default.profile, trying next
09 Dec 2020 01:02:08.633 [3020849168] WARN  localhost.p:33333 <> - cannot open file /root/etc/default.profile, trying next
09 Dec 2020 01:02:08.633 [3020849168] WARN  localhost.p:33333 <> - cannot open file /opt/xdaq/share/default/profile/default.localhost.profile, trying next
09 Dec 2020 01:02:08.634 [3020849168] WARN  localhost.p:33333 <> - cannot open file /opt/xdaq/share/default/profile/default.profile, trying next
09 Dec 2020 01:02:08.641 [3020849168] INFO  localhost.p:33333 <> - Loaded profile: /opt/xdaq/etc/default.profile
09 Dec 2020 01:02:08.648 [3020849168] INFO  localhost.p:33333 <> - core::b2innub version: 02.05.02, compiled on Nov 28 2020 at 02:48:57
09 Dec 2020 01:02:08.653 [3020849168] INFO  localhost.p:33333 <> - core::i2outils version: 04.00.00, compiled on Nov 28 2020 at 02:30:40
09 Dec 2020 01:02:08.666 [3020849168] INFO  localhost.p:33333 <> - core::executive version: 04.01.01, compiled on Nov 28 2020 at 02:43:50
09 Dec 2020 01:02:08.690 [3020849168] INFO  localhost.p:33333 <> - core::pthttp version: 04.06.00, compiled on Nov 28 2020 at 02:31:34
09 Dec 2020 01:02:08.702 [3020849168] INFO  localhost.p:33333 <> - core::ptfifo version: 04.03.01, compiled on Nov 28 2020 at 02:37:43
09 Dec 2020 01:02:08.708 [3020849168] INFO  localhost.p:33333 <> - core::xrelay version: 04.01.00, compiled on Nov 28 2020 at 02:47:55
09 Dec 2020 01:02:08.724 [3020849168] INFO  localhost.p:33333 <> - core::hyperdaq version: 05.03.00, compiled on Nov 28 2020 at 02:47:41
09 Dec 2020 01:02:08.725 [3020849168] DEBUG localhost.p:33333 <> - Create application descriptor for class: executive::Application id: 0
09 Dec 2020 01:02:08.771 [3020849168] INFO  localhost.p:33333.executive::Application.lid(0) <> - Log URL not set (was already console)
09 Dec 2020 01:02:08.771 [3020849168] INFO  localhost.p:33333.executive::Application.lid(0) <> - Changed Log level to INFO
09 Dec 2020 01:02:08.790 [3020849168] INFO  localhost.p:33333.pt::http::PeerTransportHTTP.lid(1) <> - Adding header Access-Control-Allow-Origin:*
09 Dec 2020 01:02:08.791 [3020849168] INFO  localhost.p:33333.pt::http::PeerTransportHTTP.lid(1) <> - Adding header Access-Control-Allow-Methods:POST, GET, OPTIONS
09 Dec 2020 01:02:08.791 [3020849168] INFO  localhost.p:33333.pt::http::PeerTransportHTTP.lid(1) <> - Adding header Access-Control-Allow-Headers:x-requested-with, soapaction
09 Dec 2020 01:02:08.791 [3020849168] INFO  localhost.p:33333.pt::http::PeerTransportHTTP.lid(1) <> - Adding expire rule image/png:PT4300H
09 Dec 2020 01:02:08.791 [3020849168] INFO  localhost.p:33333.pt::http::PeerTransportHTTP.lid(1) <> - Adding expire rule image/jpg:PT4300H
09 Dec 2020 01:02:08.792 [3020849168] INFO  localhost.p:33333.pt::http::PeerTransportHTTP.lid(1) <> - Adding expire rule image/gif:PT4300H
09 Dec 2020 01:02:08.792 [3020849168] INFO  localhost.p:33333.pt::http::PeerTransportHTTP.lid(1) <> - Adding expire rule application/x-shockwave-flash:PT120H
09 Dec 2020 01:02:08.792 [3020849168] INFO  localhost.p:33333.pt::http::PeerTransportHTTP.lid(1) <> - Adding expire rule application/font-woff:PT8600H
09 Dec 2020 01:02:08.792 [3020849168] INFO  localhost.p:33333.pt::http::PeerTransportHTTP.lid(1) <> - Adding expire rule text/css:PT4300H
09 Dec 2020 01:02:08.794 [3020849168] INFO  localhost.p:33333.pt::http::PeerTransportHTTP.lid(1) <> - No security policies on this server
09 Dec 2020 01:02:08.835 [3020849168] INFO  localhost.p:33333 <> - emtf::mgtConfigurator version: 01.00.00, compiled on Nov 30 2020 at 22:56:28
09 Dec 2020 01:02:08.843 [3020849168] INFO  localhost.p:33333 <> - Ready.
```

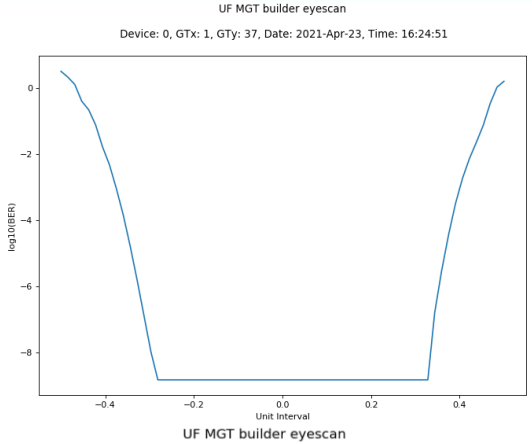
XDAQ MGT Configurator

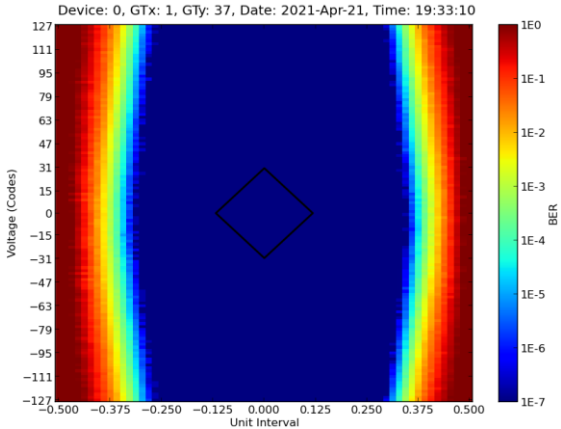
XDAQ MGT Configurator as a rework of FPGA MGT builder[6] tool set (written by A.Madorsky) quickly builds complex configurations with multiple MGTs and supports:

- Running MGT Configurator as separate XDAQ process on ZYNQ modules
- Using SOAP messages to communicate with host machines
- Software is currently used at P5 in EMTF system

FPGA MGT Builder extension

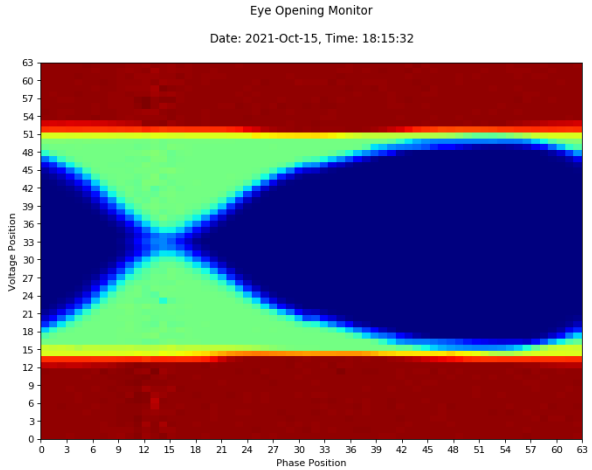
- Implemented a parasitic “Eye Scan[7]” readout of the serial links on the FPGA, applicable to both Phase-2 and Run 3 systems
  - Currently used by EMTF to monitor live links on P5
- Provided a diagram viewer for visualizing data





X2O-Zynq-software[8] (KU15P, VU13P)

- Eye Opening Monitor (EOM) on retimer chip
  - Allows for TSDC quality monitor
- FPGA firmware programmer via JTAG
- Xilinx Virtual Cables (XVC)
- Debug bridge server (UIO interface)
- Synchronous/Asynchronous clock synthesizer configuration
- QSFP monitor
- Debug scripts



References

1. X20 firmware project repository: https://github.com/madorskya/apex
2. UF IPMC github repository: https://github.com/algreshilov88/UF_IPMC
3. UG - 585 ZYNQ – 7000 SoC Technical Reference Manual: https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf
4. coreIPM open-source project: http://www.coreipm.com/
5. IPMI specification: https://www.intel.com/content/www/us/en/products/docs/servers/ipmi/ipmi-second-gen-interface-spec-v2-rev1-1.html
6. FPGA MGT Builder repository: https://github.com/madorskya/mgt_builder
7. 7 Series FPGAs GTX/GTH Transceivers: https://www.xilinx.com/support/documentation/user_guides/ug476_7Series_Transceivers.pdf
8. X2O-Zynq-software repository: https://github.com/algreshilov88/X2O-Zynq-software
9. FRU info storage specification: https://www.intel.com/content/www/us/en/servers/ipmi/ipmi-platform-mgt-fru-infostorage-def-v1-0-rev-1-3-spec-update.html

Backup

UF IPMC Polaris passed mandatory fields (58 tasks)

All required basic functionality for correct IPMC operation is implemented and tested. The following test have passed:

- IPMC state transition commands (M1, M2, M3, M4, M5, M6)
- Monitoring “Criteria Met” conditions within insertion/extraction procedures
- FRU info commands (mandatory header, can be fully completed depending on the Hardware Platform used)
- SDR commands (implemented with Human-Readable .toml format files as Full Sensor Record Type 01h that could be changed at any time without recompilation of project)
- FRU Hot Swap sensor
- IPMB-0 state sensor
- FRU Handle Switch sensor
- Dummy custom sensors (USERS can easily implement their own custom sensors)
- Sensor monitoring functionality
- Power Management commands
- Event Generation functionality:
  - Hot Swap Event messages within IPMC state transitions
  - Hot Swap Event messages within Abnormal Operation Stage
  - Hot Swap Event messages within IPMB-0 state monitoring
- Power faults handling functionality
- Fans Speed Up reaction to the exceeded thresholds within Temperature sensor implementation (USER defined functionality)
- LED commands (commands are present, but not used. In the current implementation on X2O platform doesn’t need to have this functionality. If needed USERs can add it as required)
- Reset functionality (commands are present, but not used. Currently not required in X2O platform implementation).

UF IPMC Polaris failed mandatory fields (17 tasks)

Task count	Description	Reason for error
1	Get SEL command	Not implemented: Not required because UF IPMC is using regular log files
1	Max FRU Device ID in the Get PICMG Properties response is 0	It's sufficient to use FRU Device ID=0
2	Stopping at intermediate IPMC states	Not allowed in UF IPMC
3	Read FRU Data[9] failed – 8 bytes instead of 15	Expected. Only mandatory header (8 bytes) is implemented in UF IPMC
2	Multirecord Info Area is not present in FRU Information	Not implemented
1	Product Info Area is not present in FRU Information	Not implemented
6	Temperature Event Messages fail: USER defined fields	UF IPMC does not allow changing Temperature sensor records via Shelf Manager commands. They should only be modified by updating corresponding SDR configuration files.
1	Watchdog Timer Commands Support failed	~~Not implemented~~ (Fixed by using ZYNQ built-in watchdog system timer)