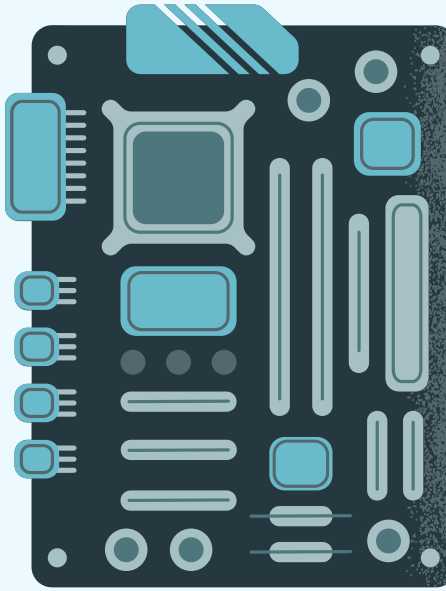
Universitat Politècnica de València

An OS-based Software Architecture for reliable front-end data readout in the Hyper-Kamiokande Experiment



A. Gómez Gambín, F. Ameli, F.J. Mora, J.F. Toledo, F.J. Ballester

25th IEEE Real Time Conference, Elba, Italy

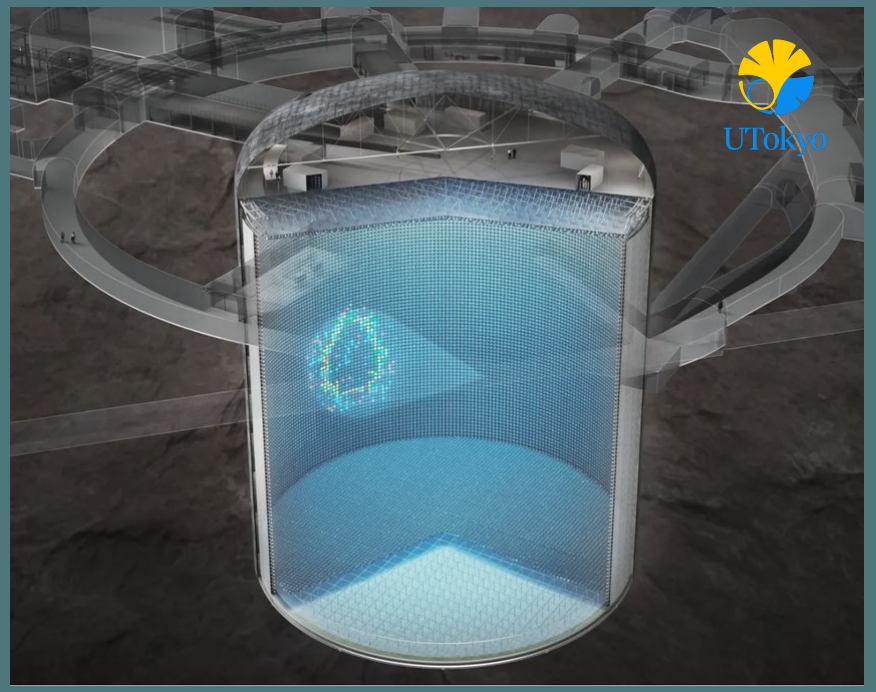
Financiado por la Unión Europea
NextGenerationEU

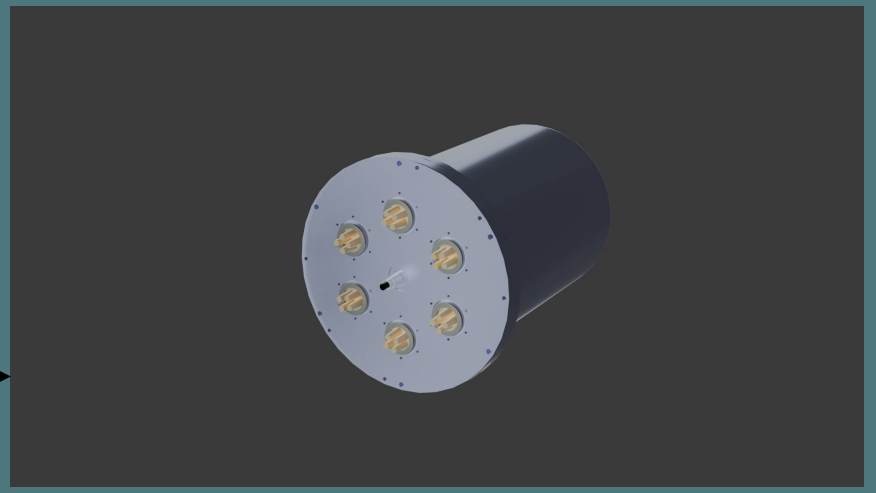
GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES

Plan de Recuperación, Transformación y Resiliencia

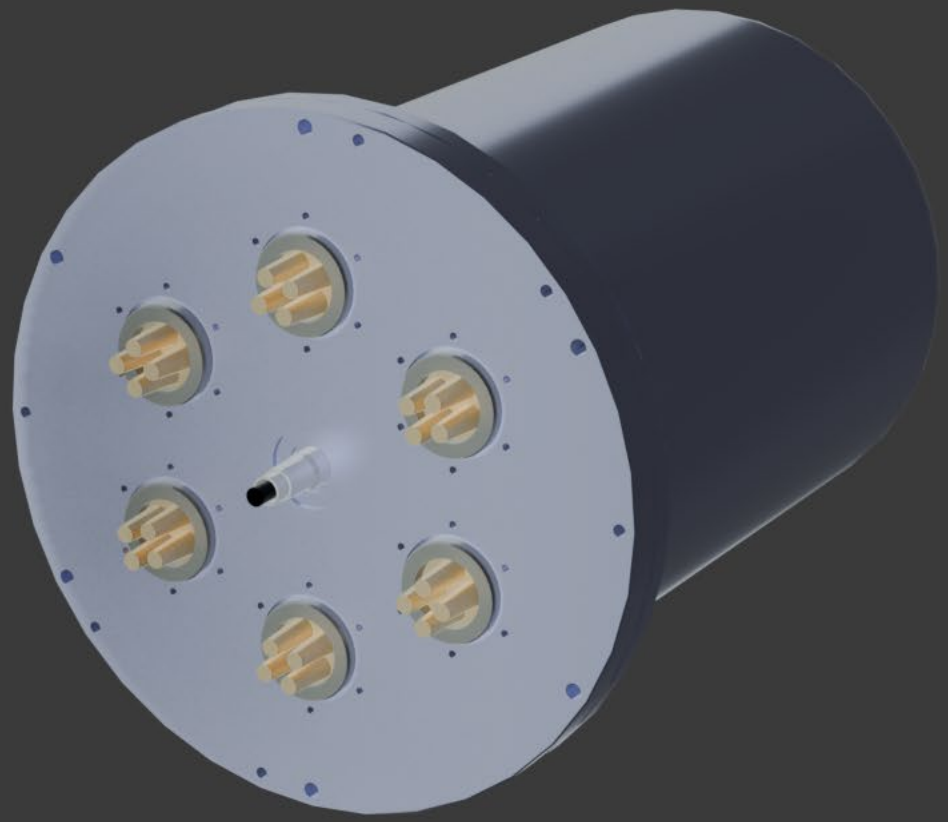
Introduction

- **New challenges arise in the design of Hyper-kamiokande with respect to Super-Kamiokande**
  - 4 times as many PMTs in a 69m diameter, 74m height detector
  - Larger detector implies longer wires, twice as long
  - Hammamatsu R12860, more sensitive PMTs
- **Design choice:** front-end electronics must be as close as possible to the PMTs
- **Consequence:** malfunctioning modules neither replaceable nor reparable without a significant effort!
  - **Design** with **reliability** and **durability** as main focus
  - **Ability to** manage and configure remotely the electronics
  - **Aggregate data** and **send it reliably** through high speed-links to the dome
- **Solution proposed:** In-vessel Software-FPGA mixed approach



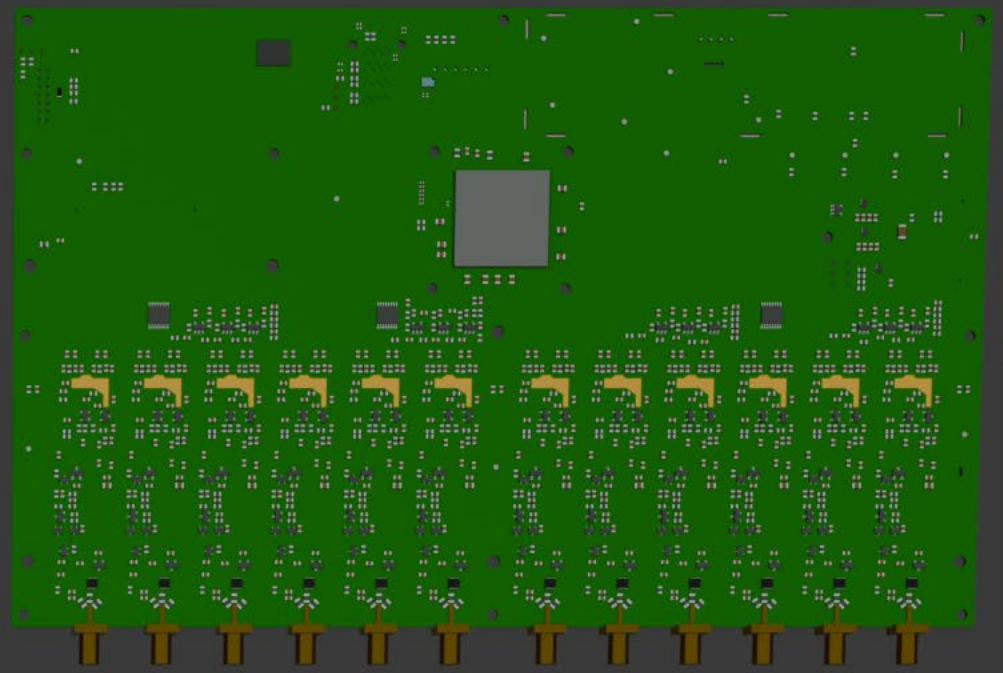


FIRST, A CLOSER LOOK INTO A VESSEL...



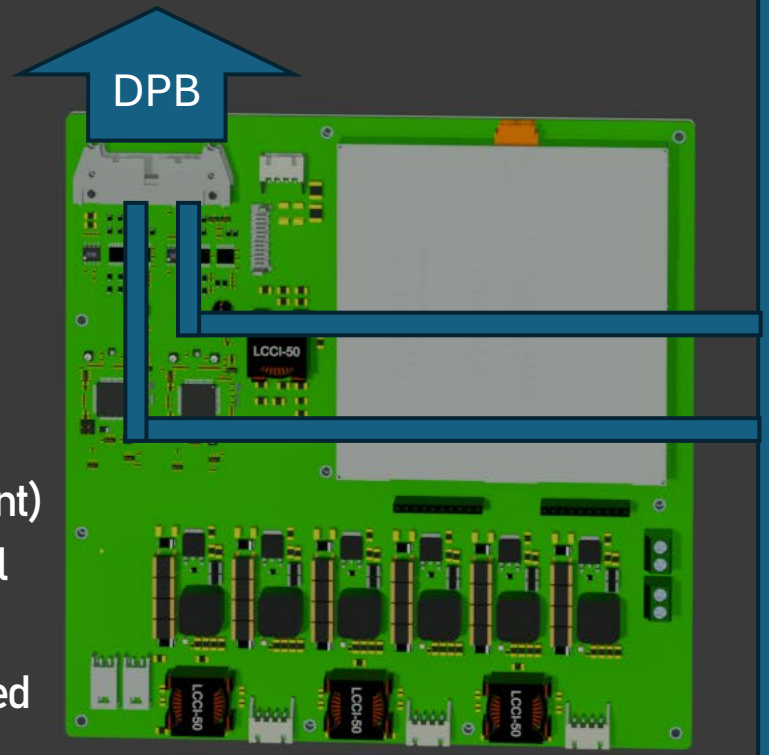
ID DIGITIZER

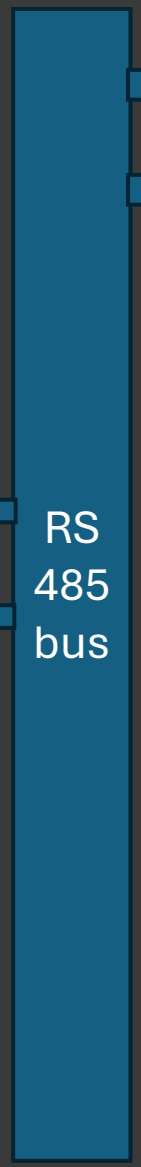
- **Direct interface with the PMTs**
  - Converts charge signal into voltage signal
  - Analog to digital conversion
- **Kintex-7 FPGA:**
  - Microblaze Soft Core running slow control tasks on single thread task
  - Command system to configure data taking parameters
- **36-pin MiniSAS SF-8643 interface with the DPB**
  - Aurora 64B/66B FPGA Core @ 3.125Gbps
  - JTAG programming of the Kintex-7 from the DPB
  - UART Lite FPGA core: serial link for slow control commands
  - Timing and synchronization differential data and clock pairs
- Digitizer for outer detector and calibrator as additional daughter boards

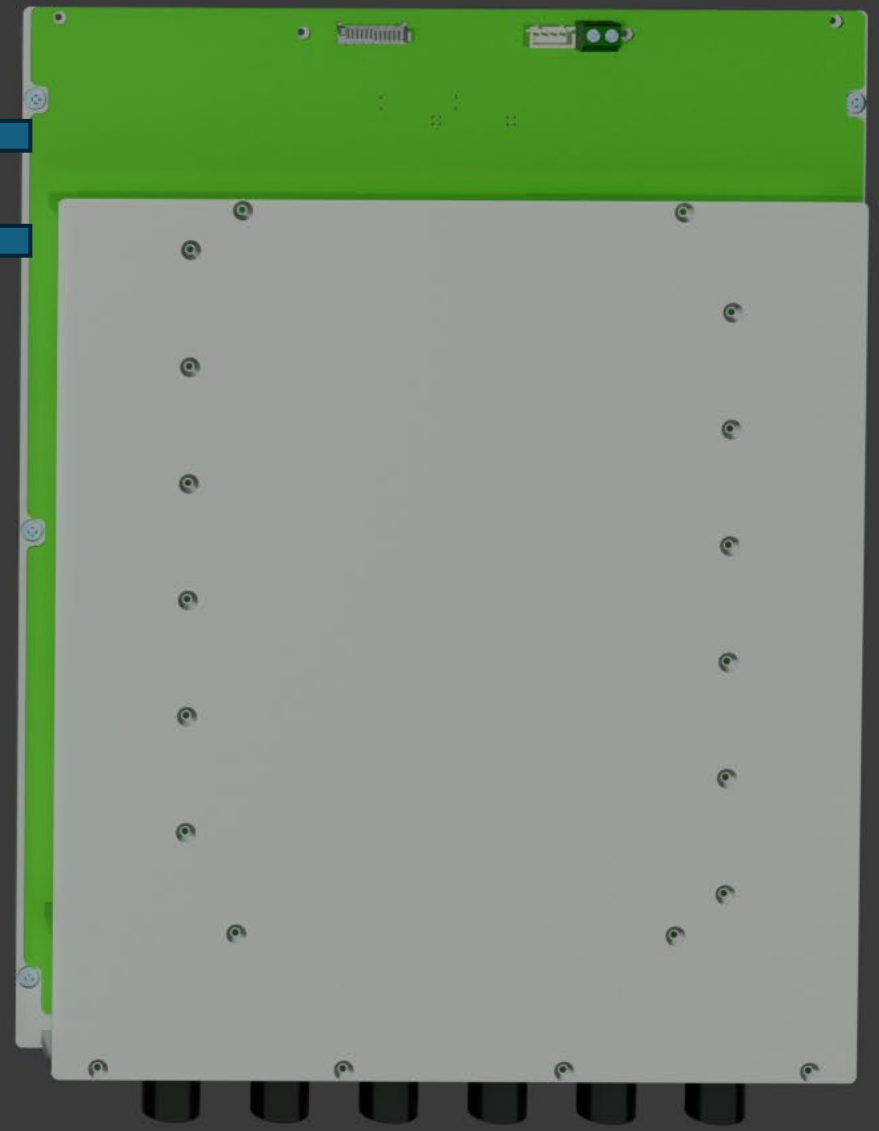


HIGH VOLTAGE AND LOW VOLTAGE BOARDS

- **Power supplies of the vessel elements**
  - **HV:** 2.5kV for the PMTs
  - **LV:** 48V to 12V for DPB, Digitizers and HV
- **MSP430 Texas Instruments microcontroller**
  - 2 per board (main, redundant)
  - Command system to control HV and LV
- Main and redundant RS485 shared buses
- Direct connection to RS485 drivers on the DPB



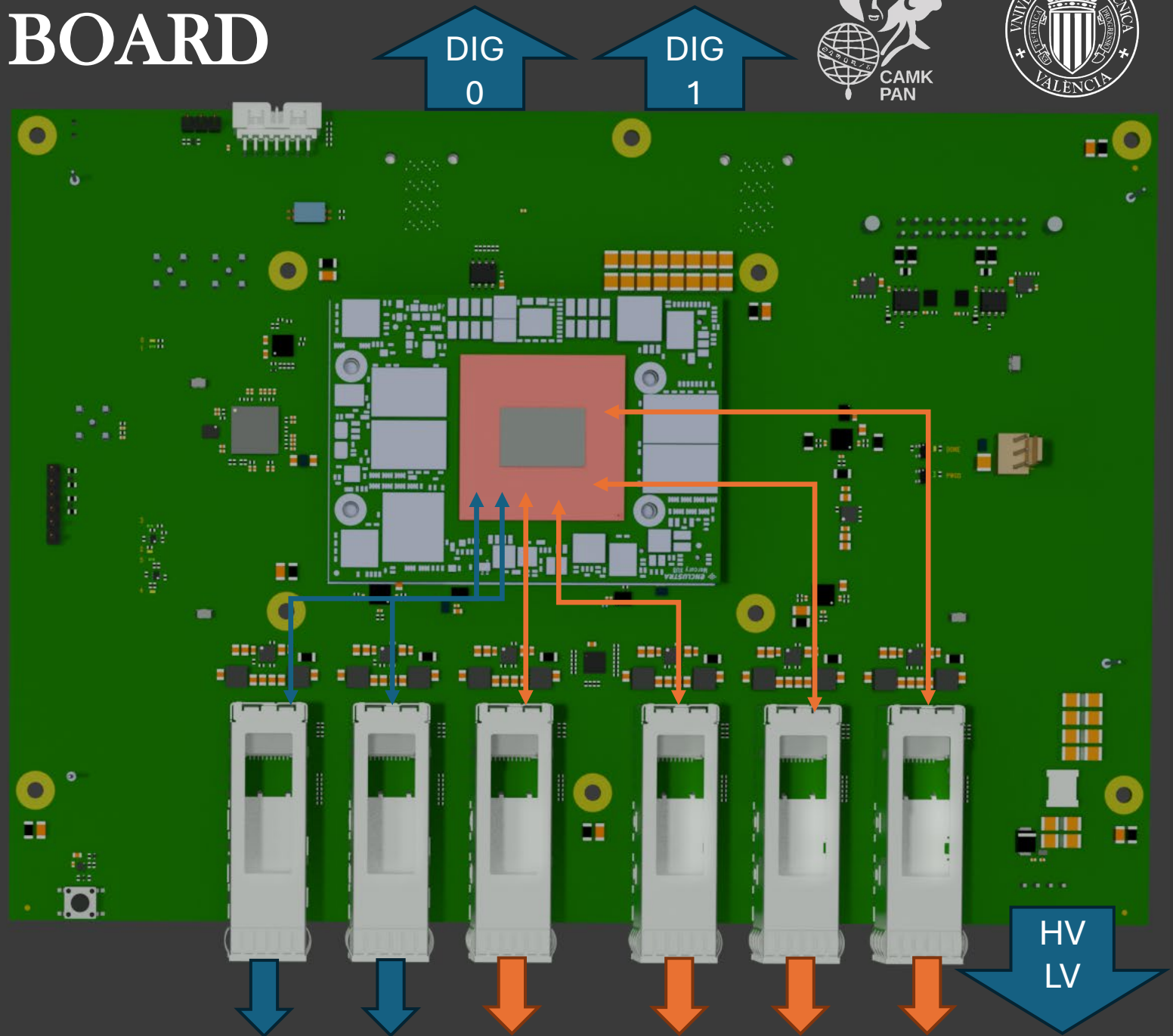




CAEN
Tools for Discovery

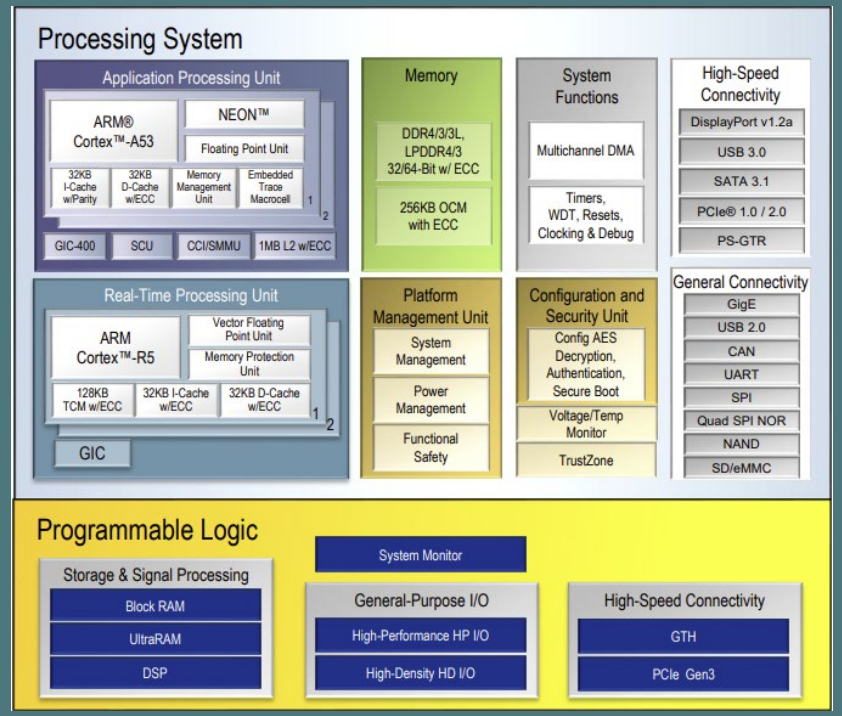
DATA PROCESSING BOARD

- **Hub of sensor and PMT data from the front-end electronics in the vessel**
  - Carrier + SOM Architecture
  - Zynq Ultrascale+ MPSoC
- **High I/O capabilities**
  - 2x **36-pin MiniSAS SF-8643**
  - 6x **SFP modules**
  - 1x D3428-5202 (HV/LV conn)
  - JTAG and UART (debugging)
- **Reliability through redundancy**
  - 2x PS-routed SFPs
  - 4x PL-routed SFPs
  - HV/LV redundant RS485 drivers
  - Data and Timing links redundancy in MiniSAS links

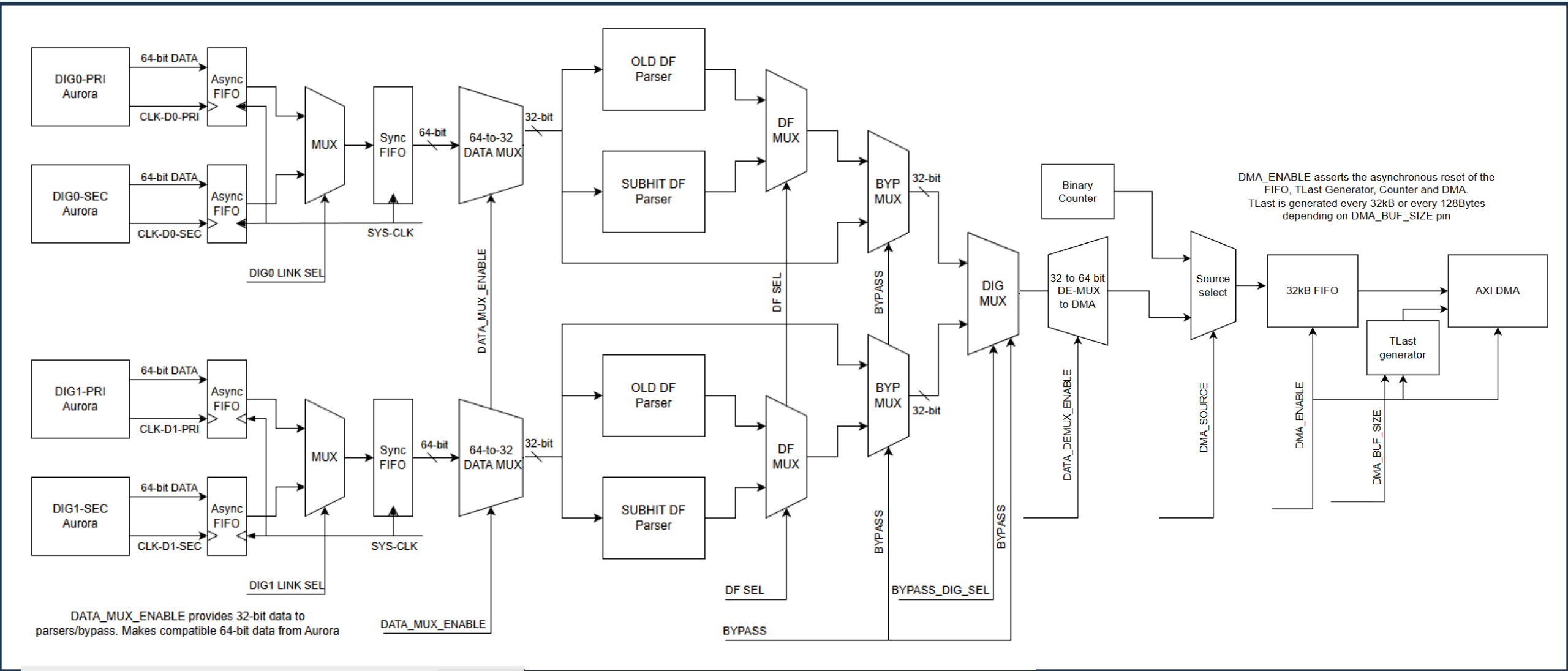


Hybrid CPU-FPGA architecture

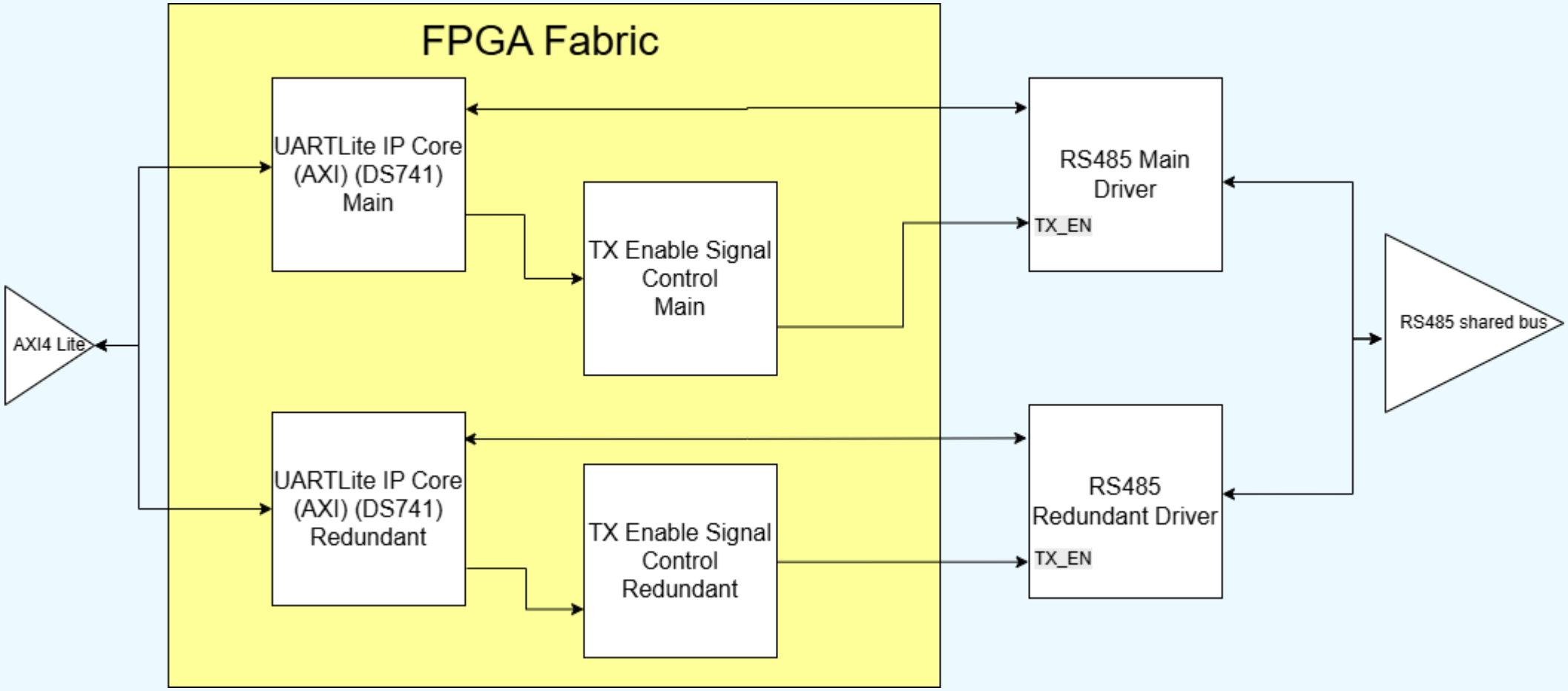
- **Zynq Ultrascale+ MPSoC (XZU4CG)**
- **Flexible FPGA combined with general purpose CPU**
- **Processing System:**
  - CPU Core Cluster (2x ARM A53 + 2x ARM R5)
  - DDR4 Memory controller for data buffering
  - Gigabit Ethernet MAC
  - Multistage bootup from many sources
- **Programmable Logic,** for custom protocols and extended peripherals:
  - Direct data readout from MiniSAS links
  - Additional serial links for slow control tasks
  - Software upgradeability for HV, LV and Digitizers



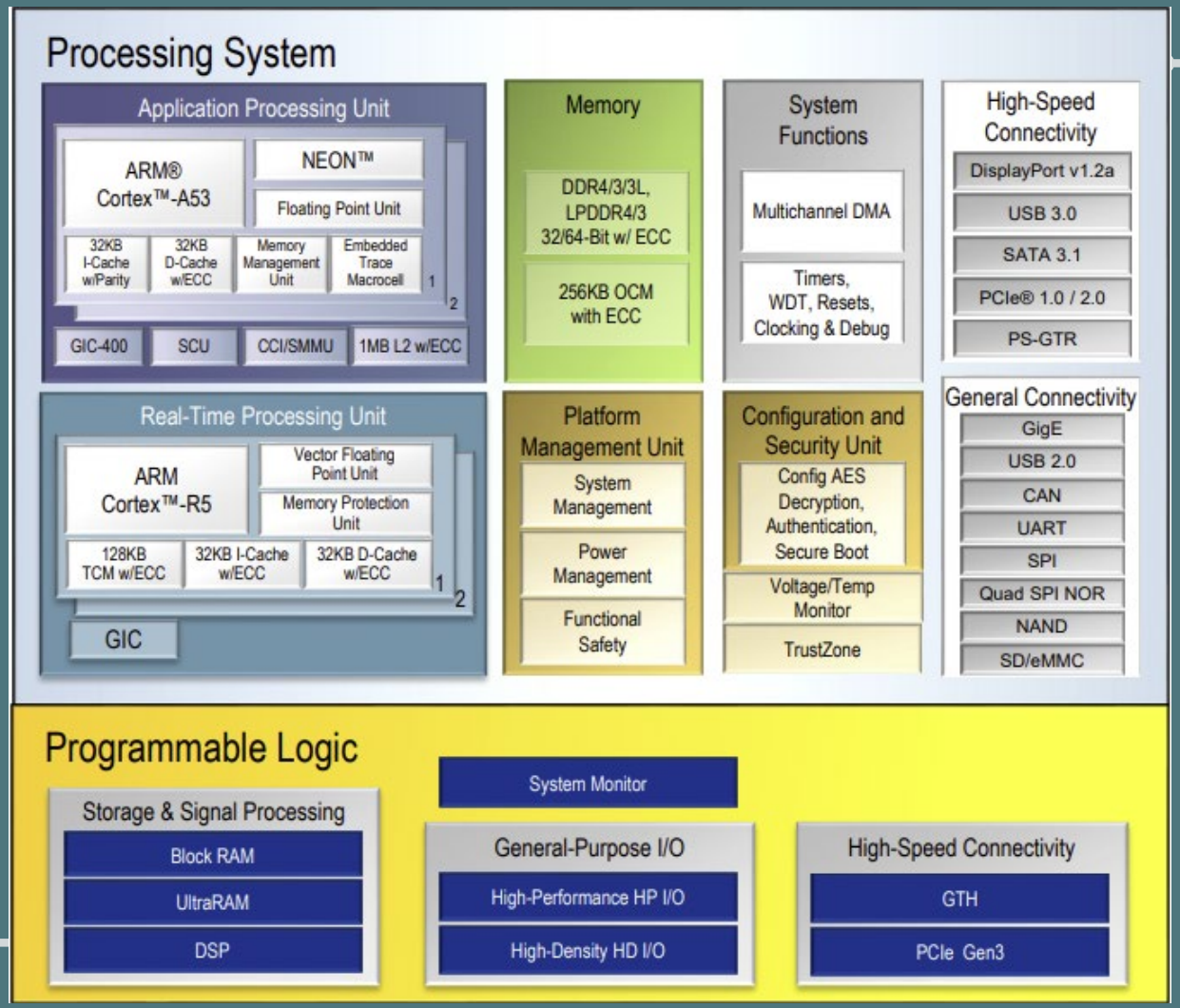
Programmable Logic Data Path:



Programmable Logic HV-LV Path: Controlling the RS485 drivers

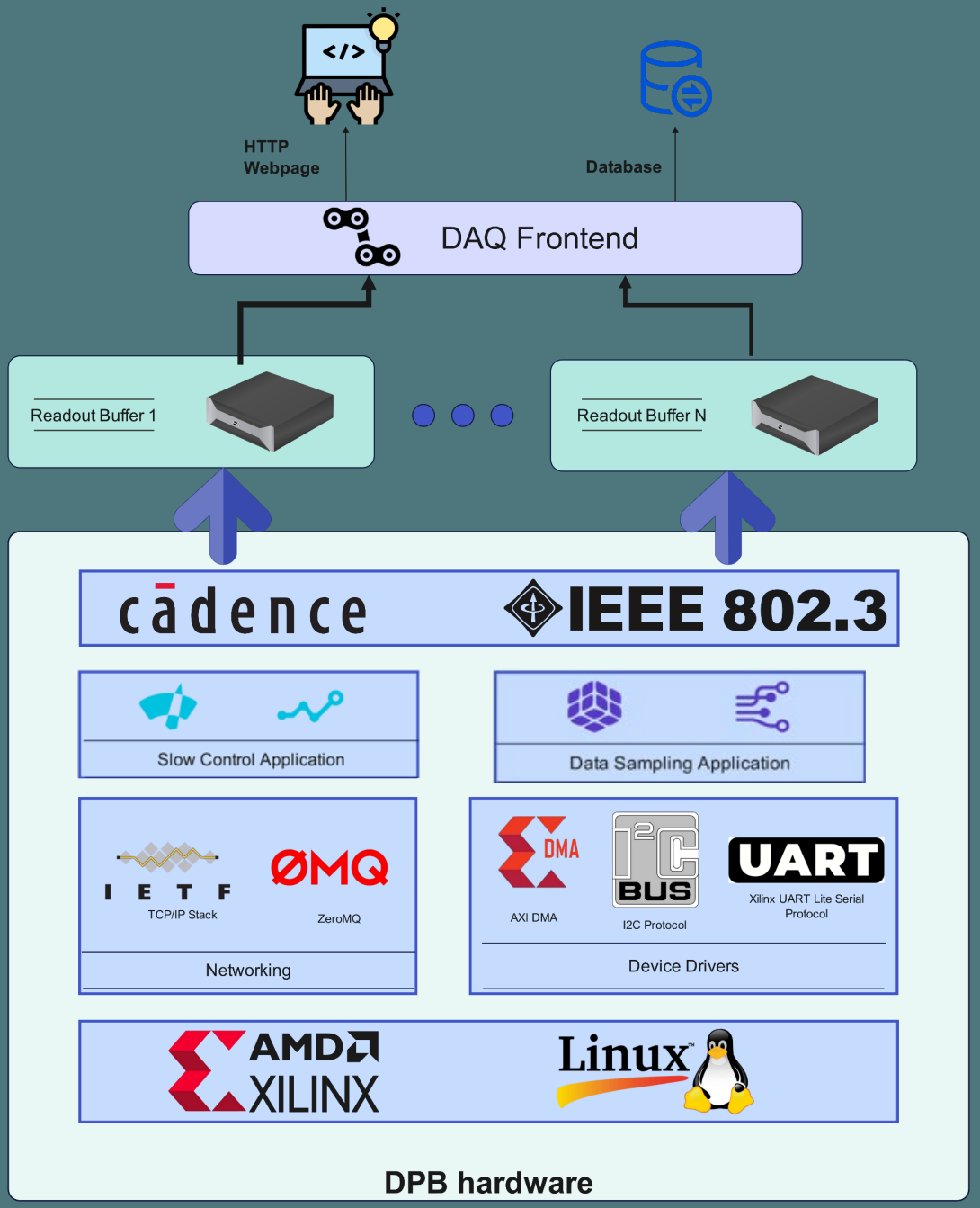


What about software?



CPU OS-based software stack

- **Embedded Linux OS**
  - Petalinux Kernel Image
  - Customised Root File System
  - Mounted as RamDisk, reliability advantages
- **Device drivers**
  - Si5345 PLL kernel drivers, bootup configuration
  - Xilinx AMS Driver
  - QSPI Flash Protection patch
  - DMA Kernel and Userspace drivers
- **Ethernet stack**
  - ZeroMQ: high reliability message-based TCP/IP network library
  - Bonding driver: virtual interface aggregating physical interfaces



Reliability-oriented software

Interoperability with other processes

- **Systemd service, best for reliability:**
  - *.service* file to restart automatically after failure and specify incompatible services
  - Support for cleanup operations if the process fails unexpectedly thanks to continuous monitoring by PID 1
  - Watchdog support: through the *sd_notify* family to avoid deadlocks inside the application's code

```
● dpb-slowcontrolapp.service - Service that runs Slow Control Application for DPB
Monitoring, Alarm and Commands handling
     Loaded: loaded (@;;file://ST1ME-XUB-ACG-1E-D11E/lib/systemd/system/dpb-
slowcontrolapp.service/lib/systemd/system/dpb-slowcontrolapp.service@;;; disabled;
vendor preset: disabled)
     Active: active (running) since Fri 2021-11-19 17:19:42 UTC; 11s ago
   Main PID: 661 (dpb-slowcontrol)
      Tasks: 11 (limit: 1671)
     Memory: 2.2M
     CGroup: /system.slice/dpb-slowcontrolapp.service
             |-661 /usr/bin/dpb-slowcontrolapp 100000 100000 5000000 50000
             |-666 /usr/bin/IIO_MONITOR -a /dev/iio:device0
             `-858 sh -c "ethtool eth1 | grep 'Link detected'"
```

DMA support in userspace

- **DMA is not a separate application-specific peripheral**
  - Other peripherals use DMA to reach higher speeds
  - DMA API in Linux is limited to kernel space
- **Implementation of kernel-userspace interface driver**
  - Driver loadable as module
  - Use ioctl in file_operations to issue and gather DMA transactions
  - Use mmap to share the DMA buffers between kernel and userspace
- **Limitations of current solutions available (dma-proc), solved:**
  - Zynq Ultrascale+ memory map divides DDR space into two parts
  - Memory allocation performed with just one *kalloc* call
  - Low data rate causes stalls and long waiting times
  - Scatter Gather is supported in hardware but not leveraged in software


```
struct file_operations dm_fops = {
    .owner    = THIS_MODULE,
    .open     = local_open,
    .release  = release,
    .unlocked_ioctl = ioctl,
    .mmap     = mmap
};
```

Slow Control Application

Data Sampling Application

ZeroMQ - DMA Zero-Copy Buffers

- **Data readout application threads**
  - DMA transactions thread
  - ZeroMQ thread
- **Pointers queues**
  - Several 128kB (configurable) contiguous memory zones are allocated
  - Q1 contains pointers to buffers not yet filled by new digitizer data
  - Q2 contains pointers to buffers with data ready to be sent to the DAQ RBUs
- **Zero copy**
  - Only pointers are used, no *memcpy()* involved
  - ZeroMQ zero-copy programming guide: http://wiki.zeromq.org/blog:zero-copy

Performance Assesment in slow control tasks

$$\%\,(UART) = \left(1 - \frac{CPU\ Time}{Monotonic\ Time}\right) * 100$$

Execution Time of each task

Task	Execution Runtime
Getting whole system monitoring data	6s
Sending command to the DPB	82.24ms
Sending command to the Digitizer	82.51ms
Sending command to HV/TH	87.28ms
Configure the whole vessel with a full set of parameters	39s
Alarm Triggering after ethernet disconnection	28ms

Execution time of UART-related functions

	Total Time [ms]	Commands sent	Monotonic clock average [ms]	CPU Time Average [ms]	% of time spent in UART processing
DPB	386.01	76	-	-	-
Digitizer	771.41	241	2.27	0.17	83.8%
Low Voltage	212.76	29	7.26	0.42	94.3%
High Voltage	5202.24	487	7.81	0.46	94.2%

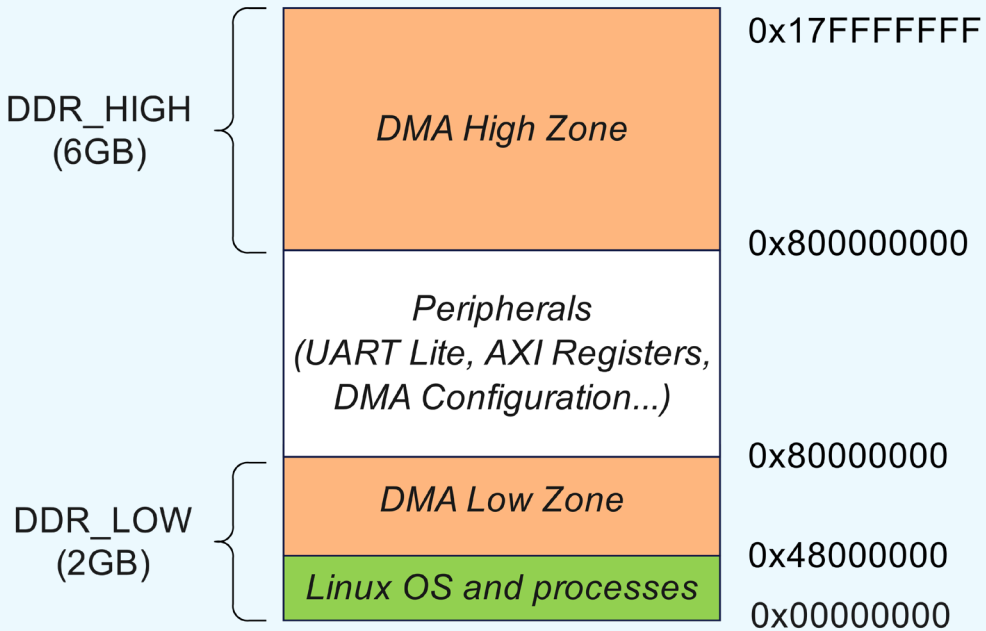
Interoperability with other processes

- **Systemd service, best for reliability:**
  - *.service* file to restart automatically after failure and specify incompatible services
  - Support for cleanup operations if the process fails unexpectedly thanks to continuous monitoring by PID 1
  - Watchdog support: through the *sd_notify* family to avoid deadlocks inside the application's code

```
* dpb-slowcontrolapp.service - Service that runs Slow Control Application for DPB
Monitoring, Alarm and Commands handling
     Loaded: loaded (8;;file://ST1ME-XU8-4CG-1E-D11E/lib/systemd/system/dpb-
slowcontrolapp.service/lib/systemd/system/dpb-slowcontrolapp.service8;;; disabled;
vendor preset: disabled)
     Active: active (running) since Fri 2021-11-19 17:19:42 UTC; 11s ago
   Main PID: 661 (dpb-slowcontrol)
      Tasks: 11 (limit: 1671)
     Memory: 2.2M
     CGroup: /system.slice/dpb-slowcontrolapp.service
             |-661 /usr/bin/dpb-slowcontrolapp 100000 100000 5000000 50000
             |-666 /usr/bin/IIO_MONITOR -a /dev/iio:device0
             `-850 sh -c "ethtool eth1 | grep 'Link detected'"
```

DMA support in userspace

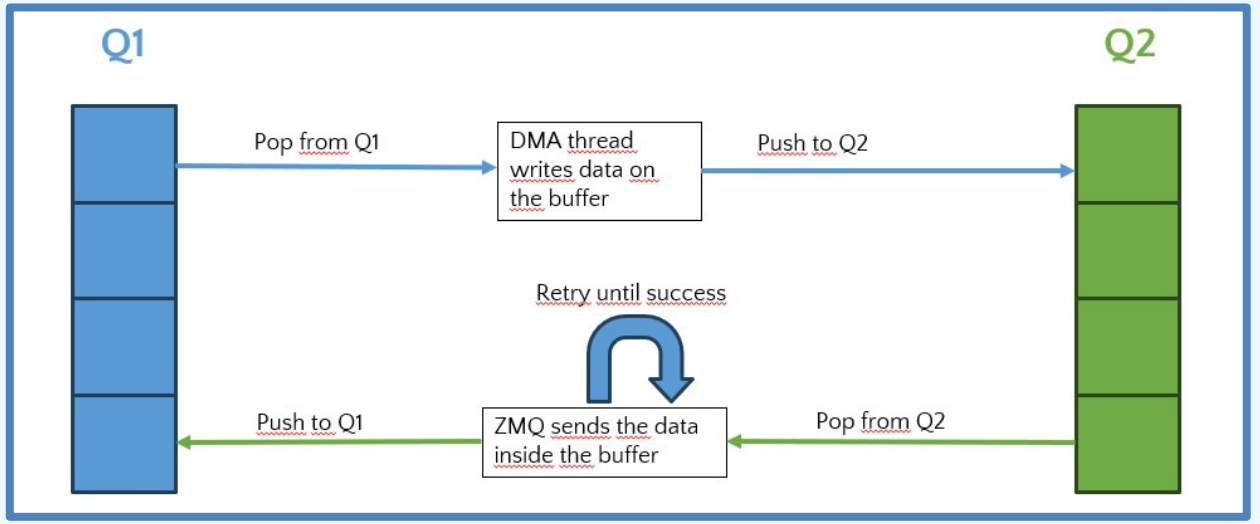
- **DMA is not a separate application-specific peripheral**
  - Other peripherals use DMA to reach higher speeds
  - DMA API in Linux is limited to kernel space
- **Implementation of kernel-userspace interface driver**
  - Driver loadable as module
  - Use ioctl in file_operations to issue and gather DMA transactions
  - Use mmap to share the DMA buffers between kernel and userspace
- **Limitations of current solutions available (dma-proxy), solved:**
  - Zynq Ultrascale+ memory map divides DDR space into two parts
  - Memory allocation performed with just one *kalloc* call
  - Low data rate causes stalls and long waiting times
  - Scatter Gather is supported in hardware but not leveraged in software




```
struct file_operations dm_fops = {
    .owner    = THIS_MODULE,
    .open     = local_open,
    .release  = release,
    .unlocked_ioctl = ioctl,
    .mmap   = mmap
};
```

ZeroMQ - DMA Zero-Copy Buffers

- **Data readout application threads**
  - DMA transactions thread
  - ZeroMQ thread
- **Pointers queues**
  - Several 128kB (configurable) contiguous memory zones are allocated
  - Q1: contains pointers to buffers not yet filled by new digitizer data
  - Q2: contains pointers to buffers with data ready to be sent to the DAQ RBUs
- **Zero copy**
  - Only pointers are used, no *memcpy()* involved
  - ZeroMQ zero-copy programming guide: http://wiki.zeromq.org/blog:zero-copy



Performance Assesment in slow control tasks

$$\%\ (UART) = \left(1 - \frac{CPU\ Time}{Monotonic\ Time}\right) * 100$$

Execution Time of each task

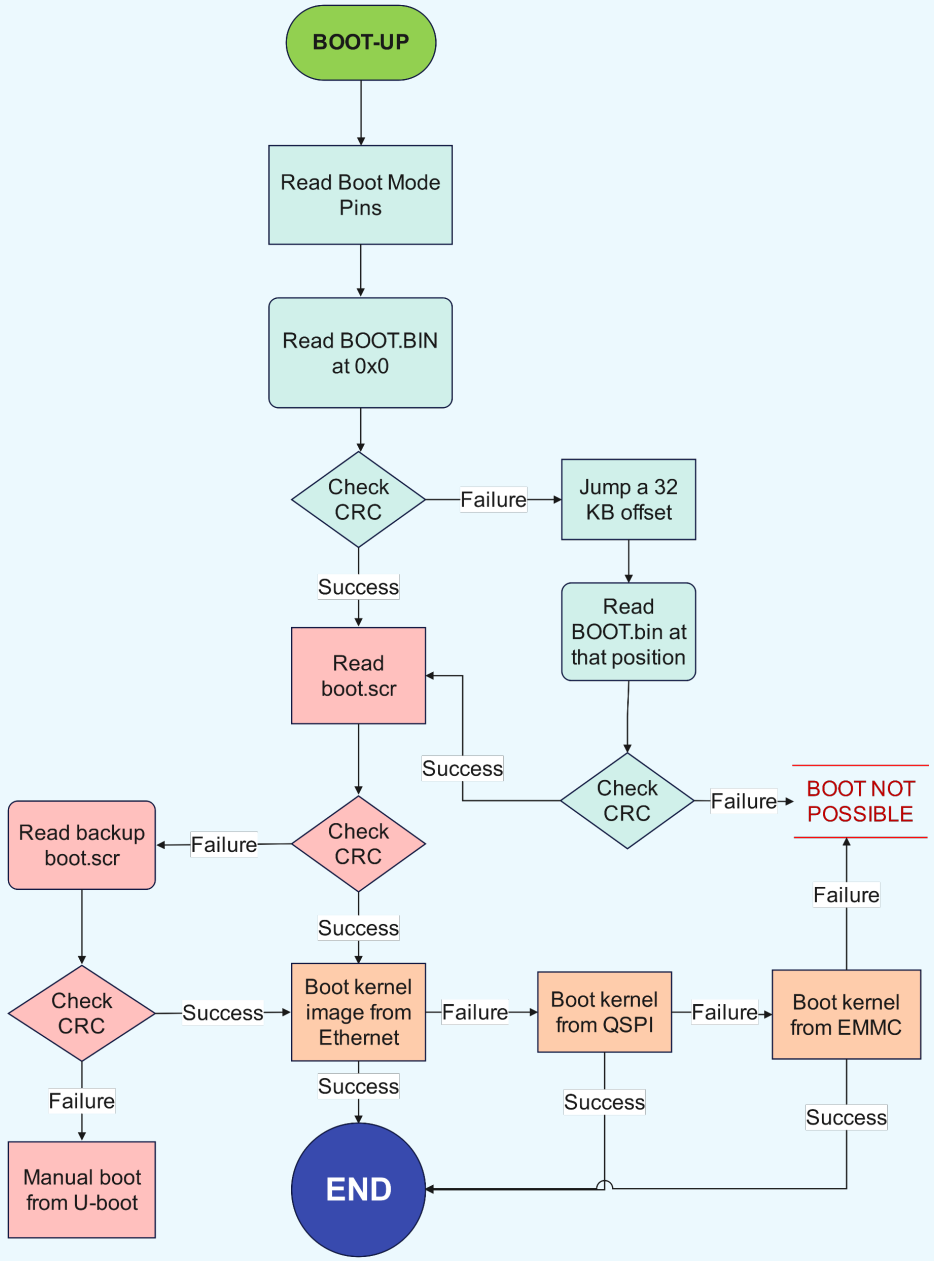
Task	Execution Runtime
Getting whole system monitoring data	*6s*
Sending command to the DPB	*83.24ms*
Sending command to the Digitizer	*82.53ms*
Sending command to HV/LV	*97.39ms*
Configure the whole vessel with a full set of parameters	*39s*
Alarm Triggering after ethernet disconnection	*20ms*

Execution time of UART-related functions

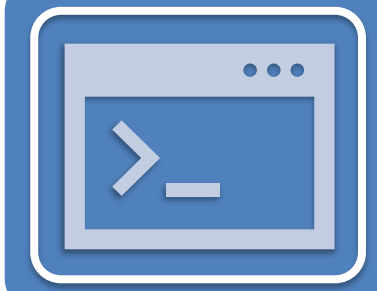
	Total Time (ms)	Commands sent	Monotonic clock average (ms)	CPU Time Average (ms)	% of time spent in UART processing
DPB	*106.01*	*76*	-	-	-
Digitizer	*771.41*	*241*	*2.27*	*0.37*	*83.8%*
Low Voltage	*212.76*	*29*	*7.26*	*0.42*	*94.3%*
High Voltage	*3201.24*	*407*	*7.81*	*0.46*	*94.2%*

Reliable Bootup Procedure

- **Multistage bootup**
  - **Stage 1:** Boot.BIN loading (FSBL, PMU, Bitstream)
  - **Stage 2:** boot.scr execution (U-boot script)
  - **Stage 3:** kernel and rootfs lookout
- **Redundant sources to perform stage 2 and stage 3**
  - Boot script is copied 8 times into the QSPI
  - Kernel bootup can be performed from Ethernet, QSPI and eMMC
  - Bitstream load can be hot-loaded from ethernet
- **Ethernet bootup logging**
  - U-boot normally outputs to stdout (UART serial port)
  - Netconsole can be configured to output through ethernet link a logging console
  - In Hyper-Kamiokande, UART is not connected to DAQ.

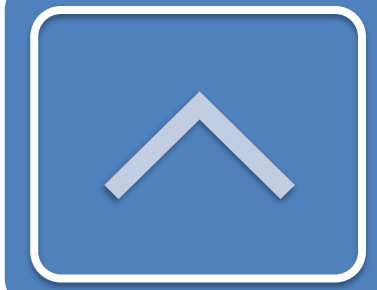


Conclusions



Reliable software stack

- Custom Rootfs mounted in RAM Disk
- Full control over ethernet link



Upgradeability

- Update from DAQ for all electronics boards
- Protected QSPI zone in case of failed update

Failover mechanisms

- Automatic network interface switching
- Bootup procedure from redundant sources