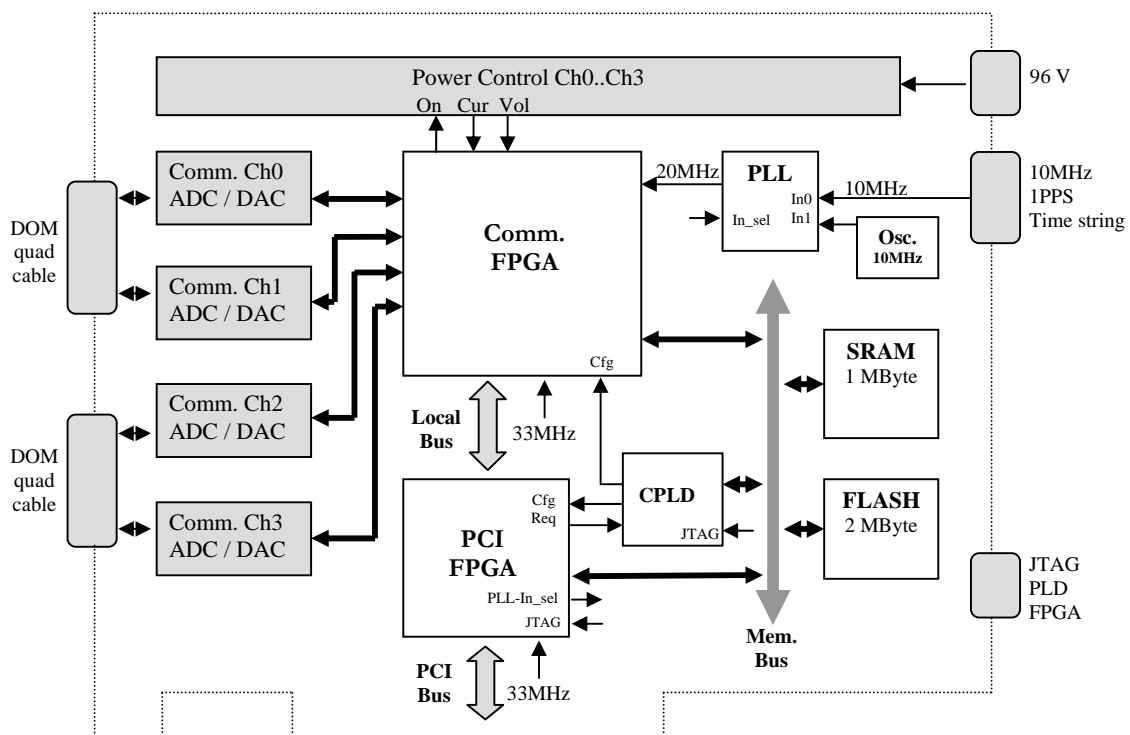


## The DOR Readout (DOR) Card

The DOR card is the interface between up to eight DOMs and the ICECUBE surface DAQ. It provides several critical functions in the DAQ chain. These include:

1. Power management for the DOM
2. Communications to and from the DOM
3. Time calibration
4. PCI Bus control
5. In system firmware update capability

### Block Diagram:



### Power Management:

The power for any of the 4 wire pairs can be switched separately. The following functions apply:

- Power on/off ramping to keep current peaks and voltage related coupling to neighbor wire pairs small
- Wire pair current and voltage monitoring
- Firmware based power off, if the wire pairs current / voltage is outside programmable limits or the cable got unplugged under power

## Communications to and from the DOM:

The DOR is the master in a strict master / slave relationship between the DOR and the DOM. The mode is half duplex, the communications protocol UART like: 1 start, 8 data, and 1 stop bit. In contrast to other encodings a DC-free bit encoding gets used. A logic one is a symmetric bipolar pulse while a logic zero is represented by a quiet line. The advantage of this method is that looking to the received waveforms as provided by the communication ADC, to detect the ones the firmware has to scan for the falling edge only. Such an algorithm requires minimal mathematic operations and therefore does not consume a lot of FPGA-logic cells.

Using another method, like the DC-free 8b/10b byte encoding, the data throughput can be doubled while maintaining the same bandwidth requirements. It's very likely, that this encoding type gets used in the near future, resulting in a data rate of 2Mbit/s for ICECUBE and 4Mbit/s for ICETOP.

The communications protocol includes the following commands:

- *Data Read Request*: Automatically data fetching from the DOMs in a round robin arbitration schema.
- *Buffer Status*: DOM / DOR data buffer synchronization, the DOR is not sending as long as the DOMs receive data buffer is full. Accordingly the DOR is not asking for DOM data as long as the DOR receiver's buffer is still full.
- *Communication Reset*: to initialize the communication
- *System Reset*: DOM warm boot
- *DOM Reboot*: The DOM reboots, initiated by a higher level software command. Because this means a reload of the DOM firmware, the communication gets temporarily lost and has to be reestablished automatically.
- *TCAL Request*: The software presses a button; the firmware is doing the rest. A round trip TCAL cycle (about 1.3ms) takes place, providing the software with a well defined TCAL data packet at the end.
- *Idle*: If no data transfer takes place the DOMs (A and B) gets pinged all the time to ensure that they are still communicating.

*CRC Error handling*: Data packets having CRC errors did not get written into the receive buffer. The software is responsible for data packets retransmit. Control packets with CRC errors get ignored.

*Hardware Timeout*: If a DOM is not responding for more than 4 seconds a hardware timeout is recognized, all the bandwidth gets dedicated to the still running DOM.

## Time Calibration:

The DOR receives at its RJ45 connector three LVDS signals distributed by the DOM hub Service Board (DSB): 10MHz, 1PPS, serial time string. These signals originate from a GPS receiver.

The 10MHz gets doubled by an onboard PLL. The resulting 20MHz is the global clock driving a 56bit counter, located in the communication FPGA of each DOR card. The Pulse per Second (1PPS) signal triggers a snapshot of the 56bit timer value. The latter

gets recorded together with the serial time string providing a cross reference between the timer value and the UTC time.

At any time the software may initiate a time calibration cycle by setting a bit and either polling on the ready status or by expecting an interrupt. The DOR firmware completely manages the time calibration cycle. At the end a TCAL data packet containing DOR and DOM time stamps and digitized TCAL waveform pulses is available. Both data sets allow transforming the DOM event time into a UTC time.

### **PCI Bus Control:**

The PCI Bus interface is performed by an Altera ACEX FPGA. This FPGA is able to run in 3.3V and 5V systems. Industry PCs backplanes are typically 5V platforms, while newer desktop PCs have 3.3V PCI buses. The PCI Bus interface is of the type master /target 32bit / 33MHz. and bases on the usage of a commercial PCI core (encrypted VHDL). The PCI core got adapted to the DOR specific needs. It is compliant to the "PCI Local Bus Specification revision 2.2".

The PCI FPGA has the following basic functions:

- PCI Bus data transfer in I/O and DMA mode
- FLASH read / write
- FPGA reload by starting a state machine located in the CPLD
- Local Bus control
- Communications clock source select

### **In System Firmware Update Capability:**

The images for both the PCI bus controller FPGA and the communications controller FPGA are located in the 2MB Flash memory. The Flash is divided into 4 pages. While page #3 contains the PCI bus controller image, pages #0...2 are preserved to accommodate the communications controller firmware. Any FLASH sector can be write protected. A jumper has to be set to allow the unprotection of a protected sector. This way the PCI FPGA's image is protected from any accidental overwriting.

The CPLD reads the appropriate flash pages, serializes the bytes and loads both FPGAs. This process happens automatically after power on or PC system reset. The software can initiate the FPGA reconfiguration by accessing a control register of the PCI FPGA. All three Altera devices are connected by a JTAG chain. Via this chain the CPLD gets programmed and the PCI bus controller FPGA gets its very first setup allowing the very first programming of the FLASH. The JTAG chain provides a convenient way for temporarily communications firmware tests as well.

The 'upper corner' of the FLASH contains a unique serial number and individual data like for instance communication ADC noise values gained at the after production test.