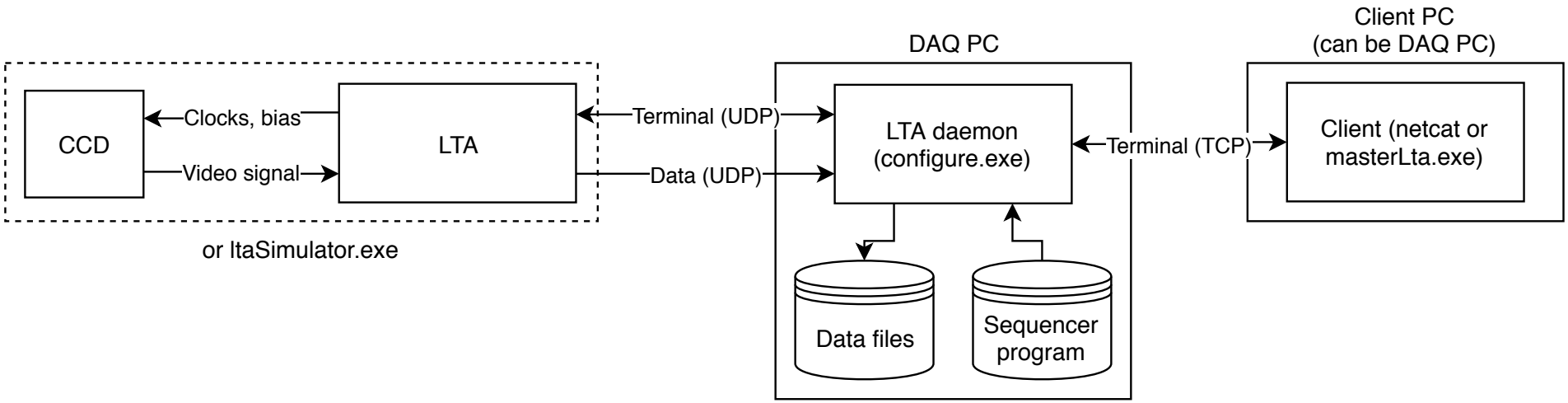
SENSEI DAQ overview

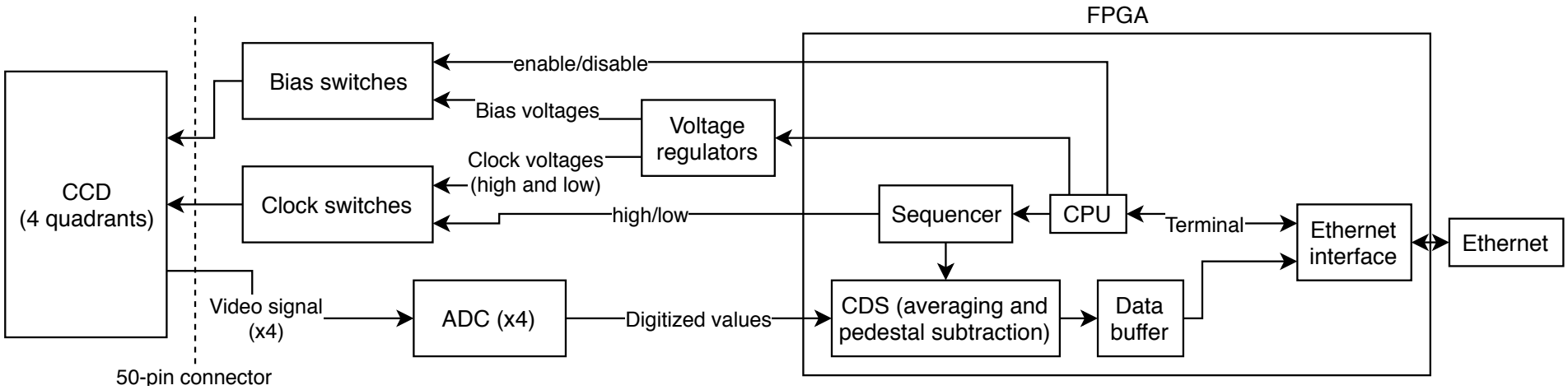
Sho Uemura

Overview



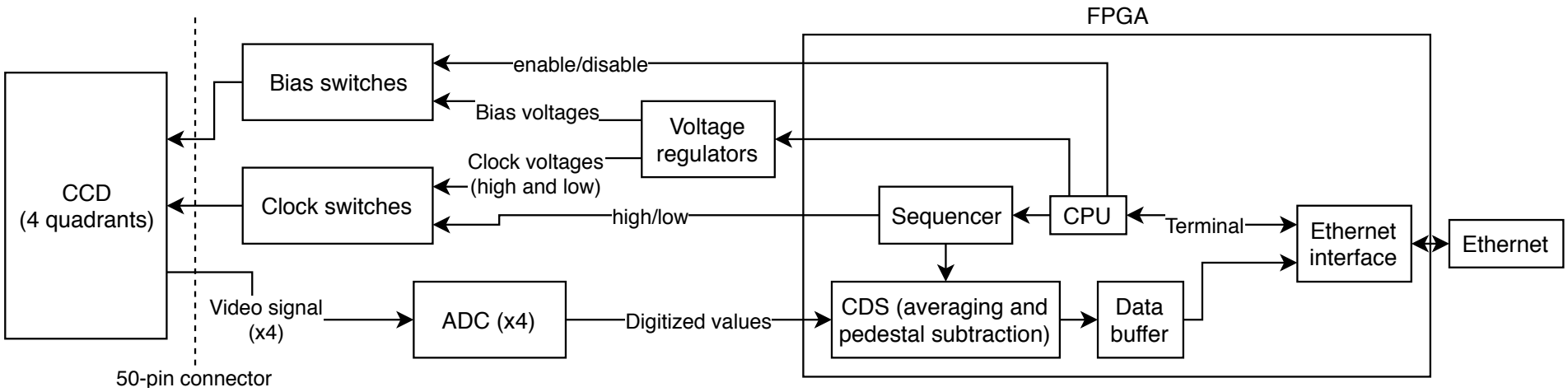
- The LTA is a single board that replaces the Monsoon system
- The core of the DAQ software is the “daemon” which communicates to the LTA over the network and writes data to disk: ltaDaemon repo
- The daemon is controlled over the network by shell commands: e.g. “lta set vdd -22” is aliased to “echo set vdd -22|nc localhost 8888”

LTA



- LTAv2 is expected to be the production version
- FPGA firmware contains a MicroBlaze soft CPU with embedded software
- Not shown: clock distribution and synchronization (for multi-LTA setups — tested but not yet used)

LTA



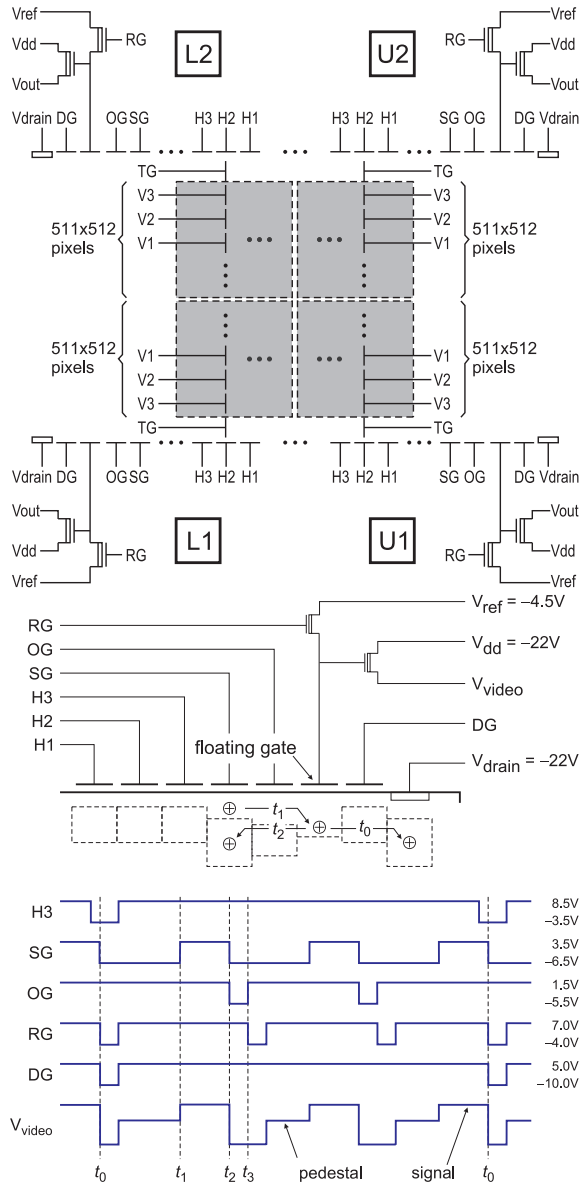
- Current firmware and embedded software: lta-v2-system-firmware, lta-v2-system-software
- “Smart Skipper” rewrite of the firmware allows for more complex sequencer programs (different number of samples for different regions of the CCD), other enhancements
  - VHDL sequencer replaced with a second MicroBlaze CPU
  - “Smart Skipper” firmware and software: lta-v2-sskp-firmware, lta-v2-sskp-ubglobal-software, lta-v2-sskp-ubseq-software
  - Testing this now, if it works we will use it for SENSEI (we don’t care about the “smart” stuff but it should fix some bugs)

Data format

- The LTA transmits raw data to the daemon, just a stream of sample values calculated by the CDS blocks
  - It's possible to lose data in transmission if a UDP packet is lost — if this happens, the output file doesn't have the right dimensions
- LTA daemon outputs a FITS file: one file with 4 HDUs (Header Data Unit), one per quadrant
  - Before last week, the daemon wrote 4 separate FITS files
- Each HDU contains:
  - Header: values of LTA variables (voltage settings, sequencer variables and program, "telemetry" aka voltage readbacks)
  - Image: array of floats, NCOL*NSAMP $\times$ NROW, not baseline-sutracted (0 e^- is not 0 ADU, and the baseline shifts from pixel to pixel)

CCD intro (diagrams from arXiv:1106.1839)

- CCD transfer and readout is controlled by cycling the “clock” voltages
- Vertical clocks move charge in the active area to the serial register: vertical transfer inefficiency can happen in the active area or at the transfer gate
- Horizontal clocks move charge in the serial register to the output stage: horizontal transfer inefficiency (HTI) can happen in the serial register or the output stage
- If you cycle the clocks more times than the physical number of rows or columns, you read out virtual pixels
  - Overscan (OS): $x > 369$ — exposed while they traverse the serial register
  - Vertical overscan: $y > 624$ — exposed during readout as they traverse the active area
  - Prescan: $x < 8$ — cells between the active area and output stage, these should be the same as overscan



Data processing: skipper2root

- skipper2root.exe converts the raw FITS file to processed data: a processed FITS file and a ROOT file
  - The images in the processed data are arrays NCOL $\times$ NROW, baseline-subtracted
  - The ROOT file also contains the headers, can be useful for QA/tuning (`https://github.com/meeg/skipperAna/blob/master/spurious_charge_scan.py`)
- The averaging of the Skipper samples is easy, the baseline subtraction is hard
  - Basic principle: most pixels in the overscan region have 0 e^-
  - Problem: there is some spurious charge in OS (not much), and HTI can lead to significant charge in OS
- With default options: average all the overscan pixels in each row, do linear interpolation between rows to estimate the baseline for every pixel
  - If you have HTI, a few rows will have big negative shifts in the 0 e^- peak
- Smarter options fit the OS distribution (-z, -a) or use a running baseline (-b), but are not default since they can go nuts if the data is bad
- You should look at the options and try them out