Hardware Triggered Scanning: Hardware

Philip Taylor, Emma Arandjelovic

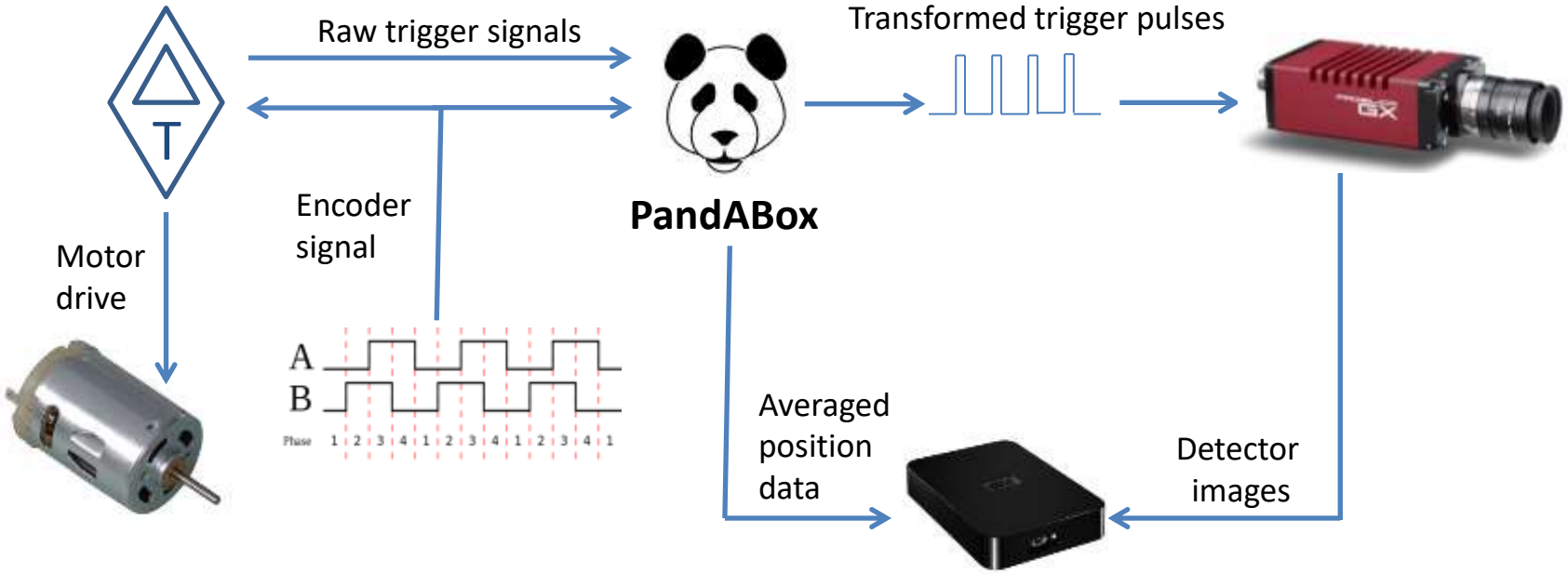
Observatory Sciences Limited

Overview

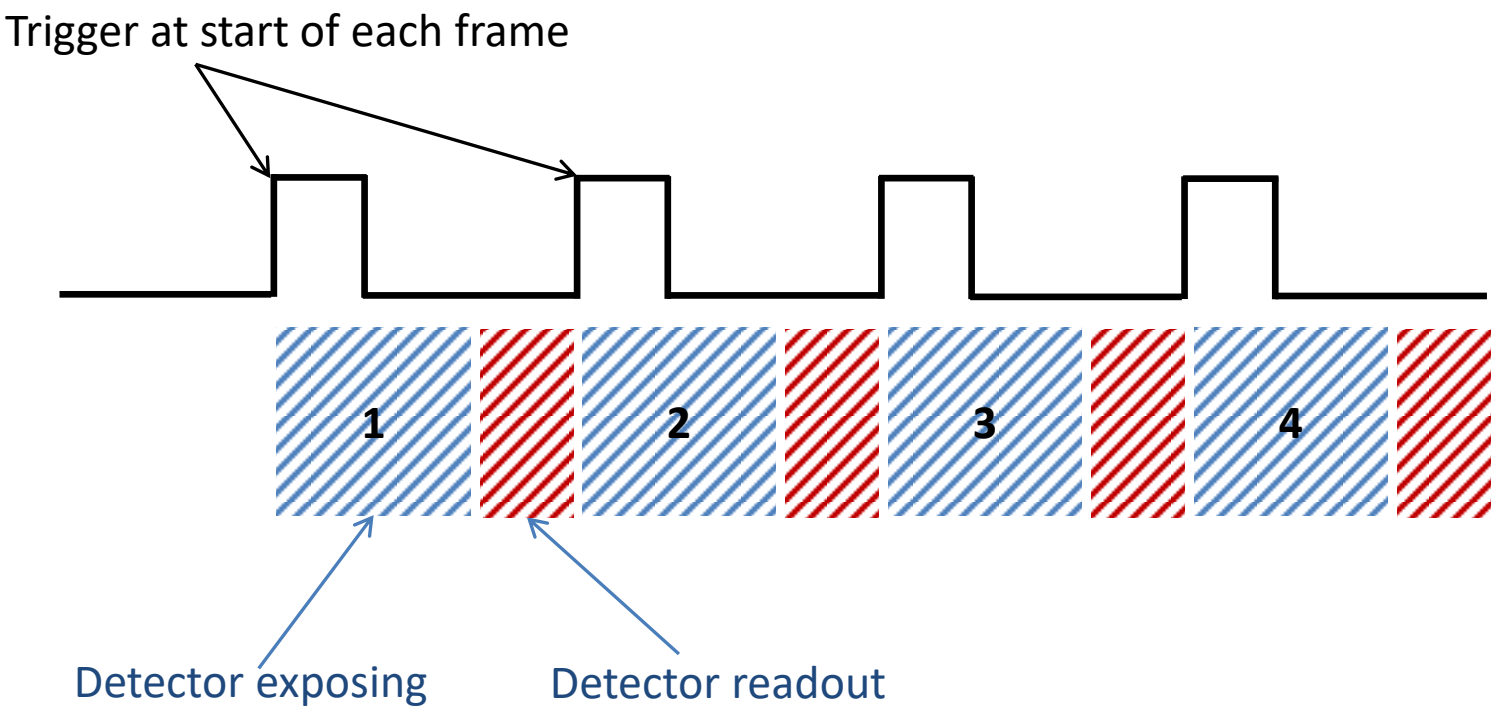
Motion trajectory control

Flexible triggering, and fast position capture

Data capture

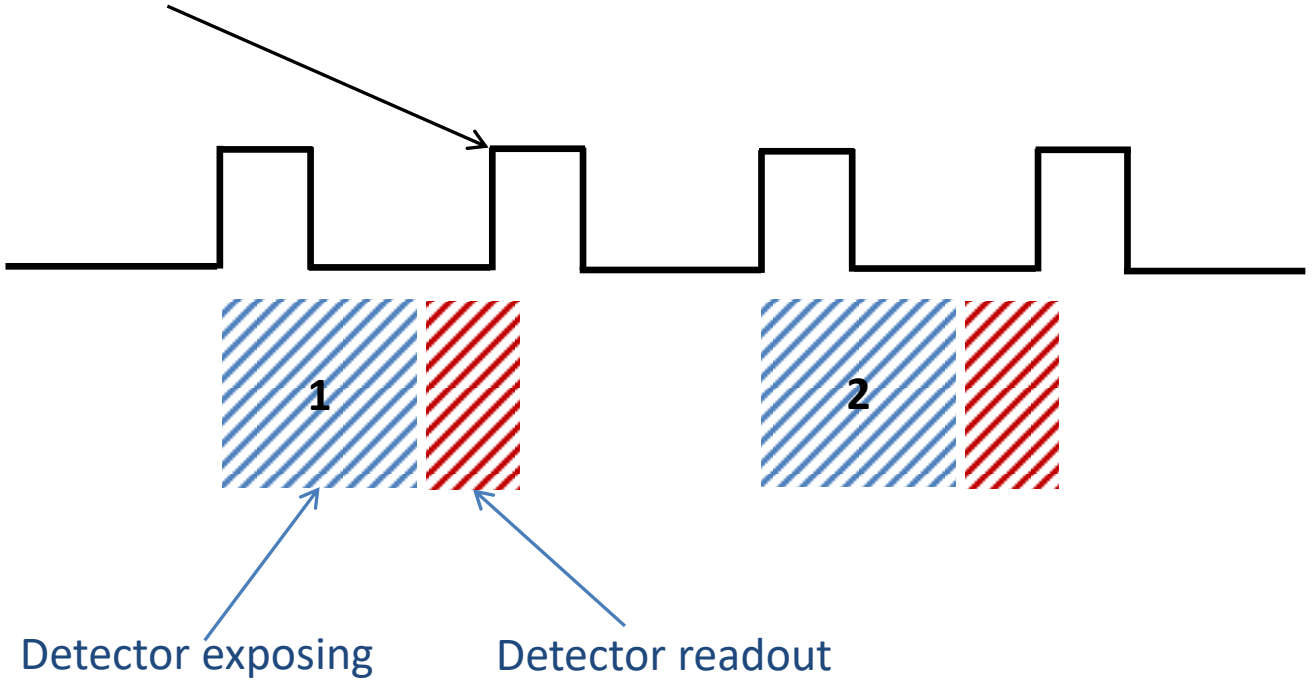


Typical Triggering Setup



Triggering Setup: Warning

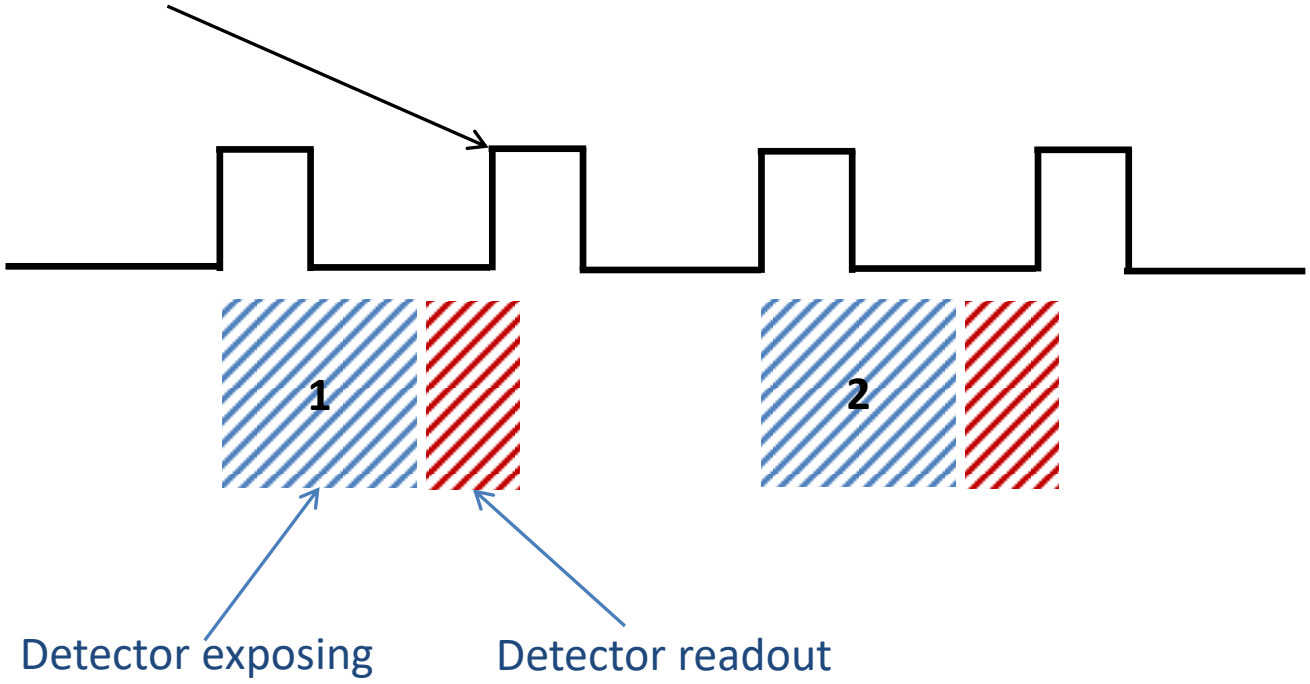
Trigger received when
detector still reading out…



Triggering Setup: Warning

Trigger received when detector still reading out…

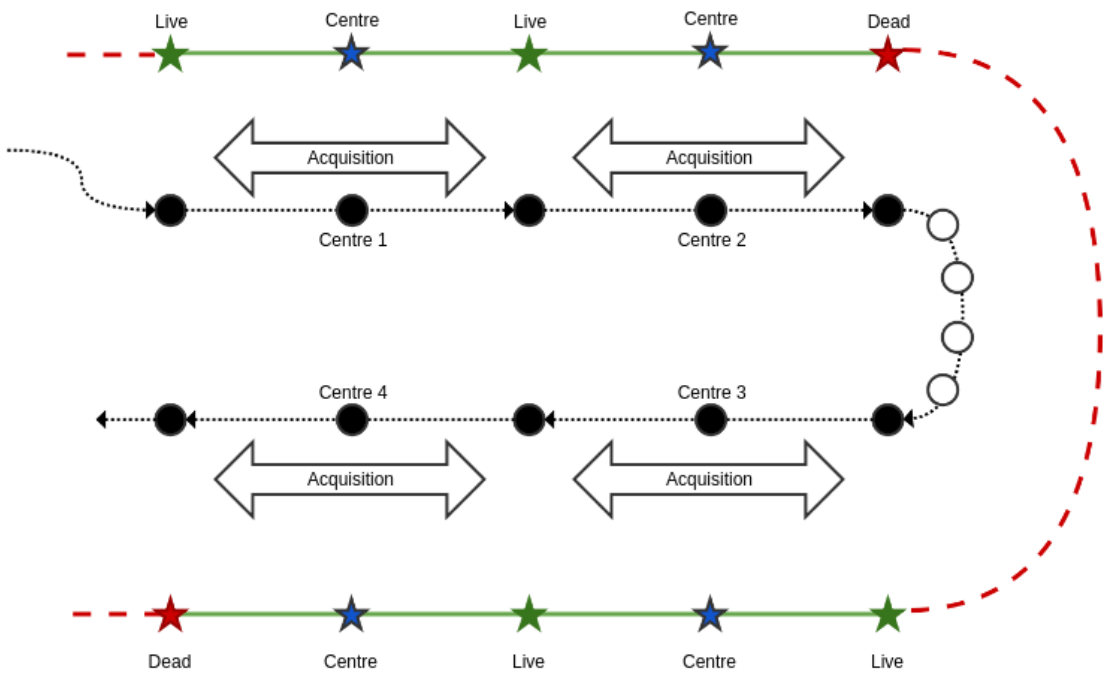
… leads to missing frames!!

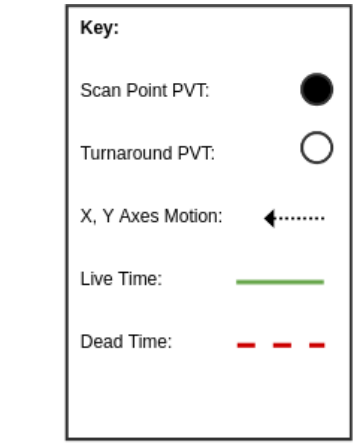


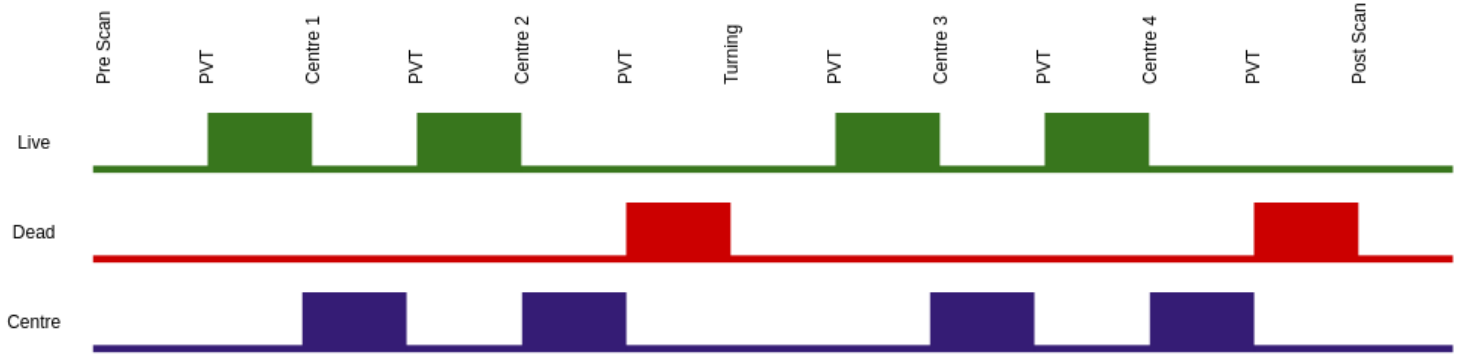
Trigger Signals

- The PMAC has 3 GPIO **output** signals, generated during a scan
- They are fed into the PandA box, which in turn triggers detector acquisition
- There are a wide range of detectors in use at Diamond which may require different trigger signals – the PandA can be configured to output an appropriate signal
- The GPIO signals are:
  1. *Live*: The start of a frame
  2. *Dead*: The start of a period when no frames are acquired
  3. *Centre*: The middle of the current frame (not normally used)

Snake Scan


Live
Centre
Live
Centre
Dead
Acquisition
Acquisition
Centre 1
Centre 2
Centre 4
Centre 3
Acquisition
Acquisition
Dead
Centre
Live
Centre
Live


Key:
Scan Point PVT:
Turnaround PVT:
X, Y Axes Motion:
Live Time:
Dead Time:


Pre Scan
PVT
Centre 1
PVT
Centre 2
PVT
Turning
PVT
Centre 3
PVT
Centre 4
PVT
Post Scan
Live
Dead
Centre

Motion Control Hardware

- Motion control performed by Delta Tau PMAC
- Note: Must be CLIPPER or Geobrick LV IMS II
  (old style Geobricks don't have enough memory)
- Co-ordinate system needed to run motion program

Trajectory Control Overview

- Precise control of the trajectory is needed
- For each step, the user specifies the end position, the velocity at the end point and the time period for that step
- This is called **PVT** mode:
  - **P**osition
  - **V**elocity
  - **T**ime

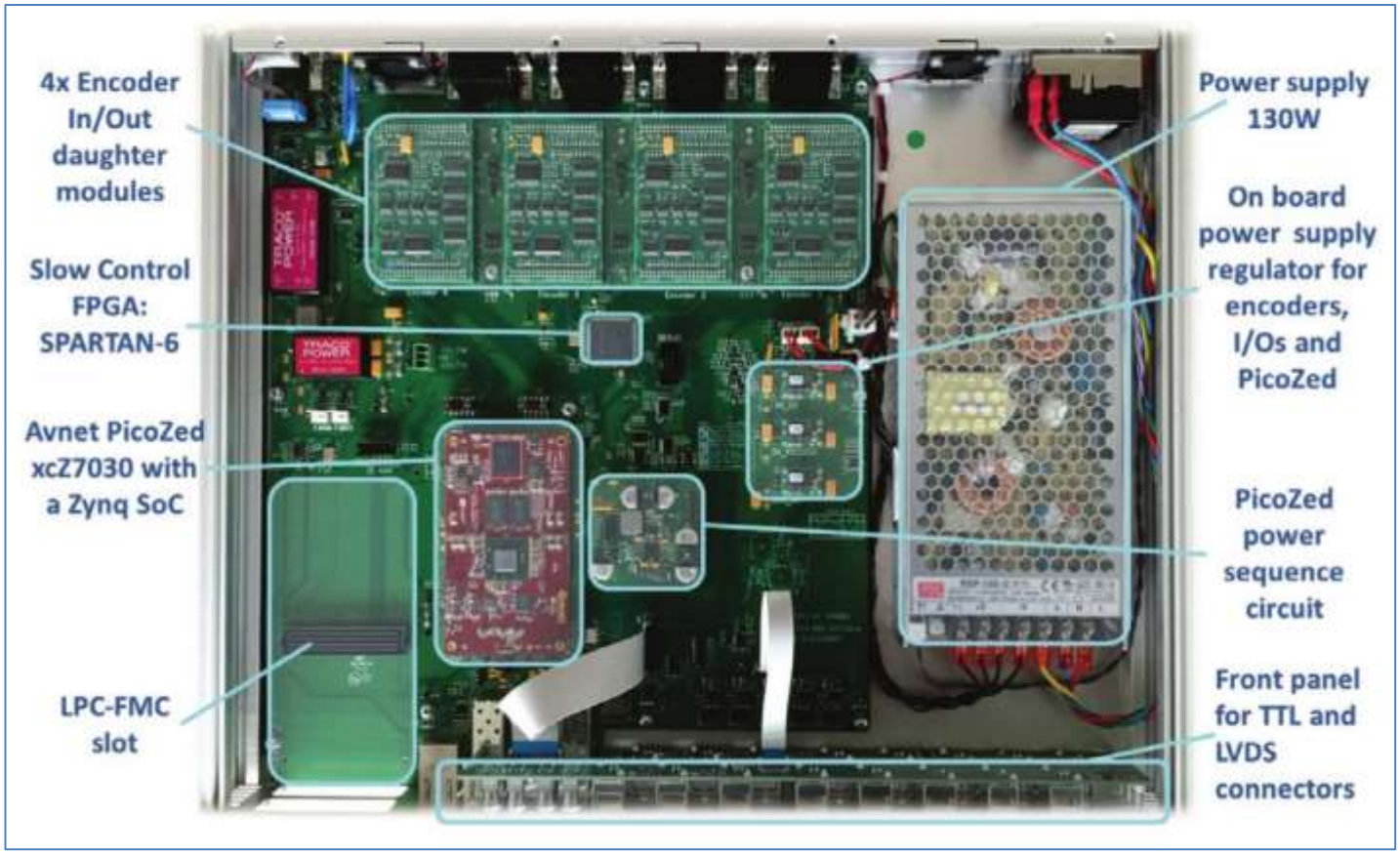
Trajectory Control Sequence

1. Malcolm generates trajectory positions and delta times using the Python module *scanpointgenerator*
2. EPICS writes them to the PMAC via the *pmac* driver
3. The PMAC calculates velocities and performs the scan using PVT mode
4. PMAC User Subroutines, which output GPIO signals, are called as each scan point is reached. 8 routines are available, corresponding to all combinations of the 3 signals: Live + Dead + Centre

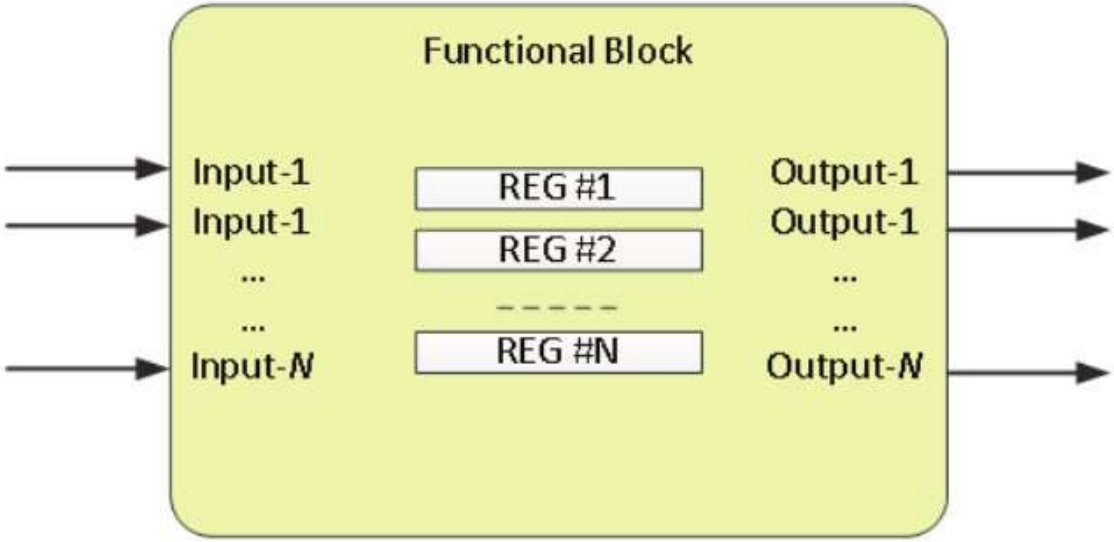
PandABox Overview

- *Position* and *Acquisition* Box
- Flexible, modular platform
- Supports a variety of scanning applications
- Based on a PicoZed module which integrates:
  1. ARM processor running embedded Linux
  2. FPGA firmware providing configurable function blocks

PandA Hardware



Functional Blocks



- Each one performs a specific function, e.g. position capture
- Configured via a number of registers
- Wired up and configured at runtime via a TCP server
- Configurations saved to disk

Available Function Blocks

Function Block	Description
INENC, OUTENC	Input and output encoder signals
TTLIN, TTLOUT	TTL Input and output signals
COUNTER	Up/down pulse counter
CALC	Sums up to 4 position inputs
LUT	Performs logical function on 5 inputs
PCAP	Position capture – options for min, max, sum
PULSE	One-shot pulse delay and stretch
SRGATE	Set Reset Gate – set configurable high or low output
….	….

For the full list and more information, see:
https://pandablocks-fpga.readthedocs.io/en/latest/blocks.html

Configuring PandA

- PandA runs an embedded web server
- Point a browser directly to this for low level configuration such as updating packages
- For normal use, use Malcolm web gui:
  - Test rig: http://localhost:8008/gui
  - Beamlines: http://ixx-control:8008/gui

PandA Configuration GUI