ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA068

# PandABlocks OPEN FPGA FRAMEWORK AND WEB STACK

C. J. Turner, M. Abbott, T. Cobb, I. Gillingham, I. S. Uzun,
Diamond Light Source Ltd, Oxfordshire, UK
G. Thibaux, Y. M. Abiven, Synchrotron SOLEIL, France

*Abstract*

PandABlocks is the open source firmware and software stack that powers PandABox, a Zynq SoC based "Position and Acquisition" platform for delivering triggers during multi-technique scanning. PandABlocks consists of a number of FPGA functional blocks that can be wired together at run-time according to application specific requirements. Status reporting and high speed data acquisition is handled by the onboard ARM processor and exposed via a TCP server with a protocol suitable for integration into control systems like "EPICS" or "TANGO". Also included in the framework is a webserver and web GUI to visualize and change the wiring of the blocks. The whole system adapts to the functional blocks present in the current FPGA build, allowing different FPGA firmware be created to support new FMC cards without rebuilding the TCP server and webserver. This paper details how the different layers of PandABlocks work together and how the system can be used to implement novel triggering applications.

## INTRODUCTION

This paper describes the firmware and software stack powering PandABox [1]; a development project, which is the result of a collaboration between Diamond Light Source [2] and Synchrotron SOLEIL [3] that was started in 2015. PandABox is a multipurpose platform for multi-technique scanning and feedback applications. It has a flexible and modular framework allowing it to be configured for numerous custom applications. The components of the PandABlocks project, described in this paper, are open source and hosted on GitHub [4].

## FPGA ARCHITECTURE

A customised Linux XiLinx [5] kernel with busy-box [6] is deployed on the ARM dual Cortex-A9 [7] as shown in Fig. 1. Specific types and quantities of different IP blocks, both standard and customised, can be loaded on the FPGA to individually be activated or instantiated by a user. Instantiated blocks representing different components can then be wired together by the user to achieve overall functionality for specific applications. These connections are shown as white lines in Fig. 1. The Processing system communicates with these IP blocks via the AXI Interconnect register interface for the ARM processor.

Individual IP blocks for use on PandABox are defined in a configuration file and are instantiated when building the system. Once built and installed, the number and type of blocks are fixed on the system with the routing/ signal flow fully configurable between them at run time. The flexibility of the IP blocks configuration at build time is to facilitate the modularity of the PandABox device itself. It is possible to install different types of FMC cards on the board and thus a method of accommodating these different configurations is necessary. This also gives the added benefit that the general configuration of the whole system is flexible. Custom blocks can be written and instantiated along with combinations of the standard supplied blocks.

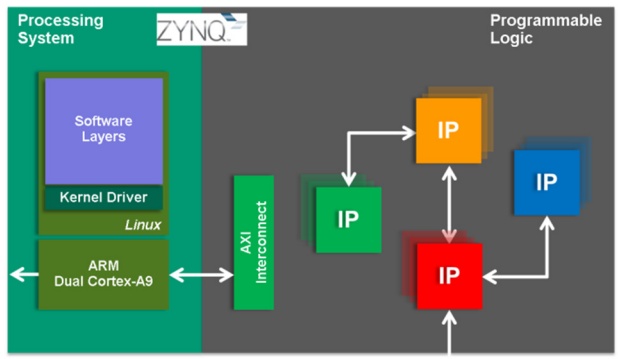


Figure 1: FPGA architecture.

## SIMULATION FRAMEWORK

Once an IP block is written, its functionality can be tested with the PandABlocks simulation framework. The simulation framework is made up of a number of different components; the input sequence file, python block simulation, output vector file, and documentation. The sequence file is a list of inputs, commands and expected outputs to be executed sequentially at given time intervals. The python code emulates the behaviour of the blocks and tests the expected behaviour as defined in the sequence file. This is useful for testing edge cases without using the actual hardware. From the simulation, an output vector file is generated which is used as input for the FPGA simulator, running the VHDL/ Verilog code. The simulation environment also provides the possibility of generating signal diagrams for each test sequence described by the sequence file; an example of one of these graphs as generated for a pulse block is shown in Fig. 2.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA068



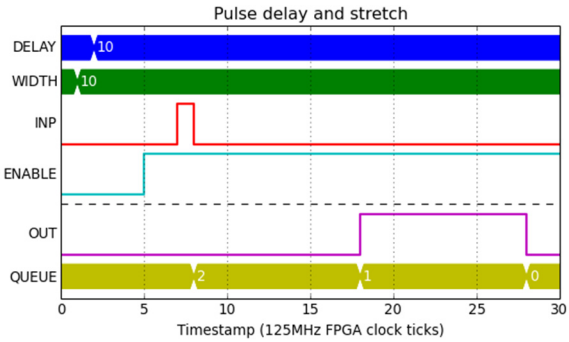


Figure 2: Generated signal diagram showing the output of a simulated pulse block. The output (OUT) is delayed by 10 counts and is stretched to have a width of 10.

## SOFTWARE ARCHITECTURE

The general use case is to configure the PandABox via the web GUI. The experiment can then be executed with control and monitoring through a standard EPICS [8] areaDetector driver, which also receives and records data. It is possible to create custom applications which can be loaded onto the PandABox for reusability. EPICS Channel Access client applications like EDM or CS-STUDIO can then be created to interact with these specific configurations. This allows only exposing a subset of controls and configuration settings to the user via EPICS as required for the particular experiments.

The different layers outlined in Fig. 3 are described in the following sections.

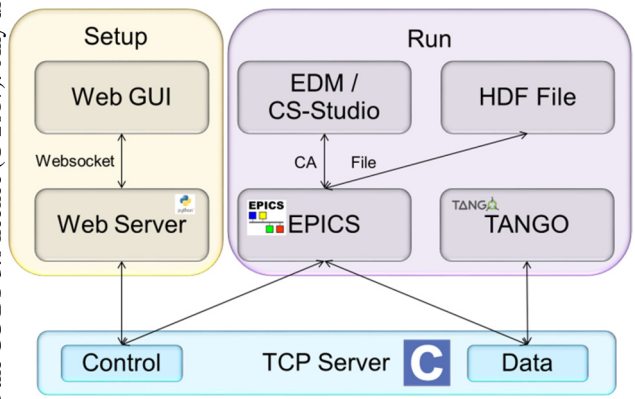


Figure 3: PandABlocks software layers.

### *TCP Server*

There are two communication ports used to communicate with the TCP server: Data and Control. Data is read from the Data port which is received from the FPGA via the DMA using the kernel driver. The Data port publishes socket endpoint with ASCII, BASE64 or BINARY data frame encoding. The Control port is used to interface with the FPGA over the registers and publishes socket endpoint with simple ASCII command response protocol. The block structure implemented on the FPGA is defined in a configuration file.

### *Configuration Web GUI*

A browser based graphical user interface using Diamond's MalcolmJS library [9] allows the user to flexibly wire blocks together in an easy to use graphical environment. This enables users to construct applications for particular experiments in a convenient and intuitive way. It also allows configuration of individual blocks and provides a visual representation of register states and values on the device. Figure 4 shows a screen shot of the web GUI with an input encoder block, position compare block, and TTL output block wired together. The blocks are in the development space in the left pane and their corresponding configuration, when selected, appears on the right. The blocks instantiated on the left are able to be dynamically moved on the screen and also wired and rewired at run time simply by clicking on inputs and outputs to join them together. The right pane displays the currently selected block's registers and configuration parameters.

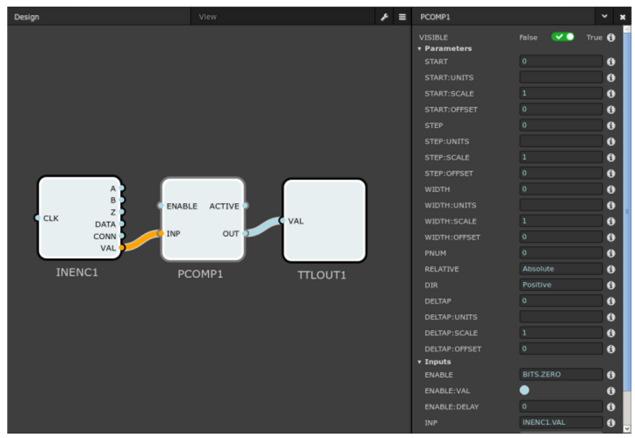


Figure 4: Screen shot of PandABlocks web GUI showing single input encoder block.

### *Webserver*

The webserver is a cut-down Malcolm [10] instance, which is a middlelayer framework implementing behaviour for continuous scans.

The use of Malcolm allows PandA support written for the webserver to be added to other unrelated Malcolm-based systems.

Configuration loading and saving support is implemented at the webserver layer which makes setting up the hardware to desired configurations and reloading these both user friendly and convenient. Configuration files are stored in JSON format, enabling easy version control.

### *EPICS Interface*

The EPICS interface is a Generic AreaDetector [11] interface which allows the acquisition of data from the PandABox. The AreaDetector structure allows NDArrays to be used and passed through the AreaDetector plugin framework; an example of this is writing data to HDF files. Each signal of received data is stored in its own NDAtribute and can be plotted with the NDAttrPlot tool. A simple example of this is shown in Fig. 5, which plots

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA068

the output of a counter block. The data is stored in an NDAttribute called COUNTER1.OUT and is plotted against the capture time stamp.

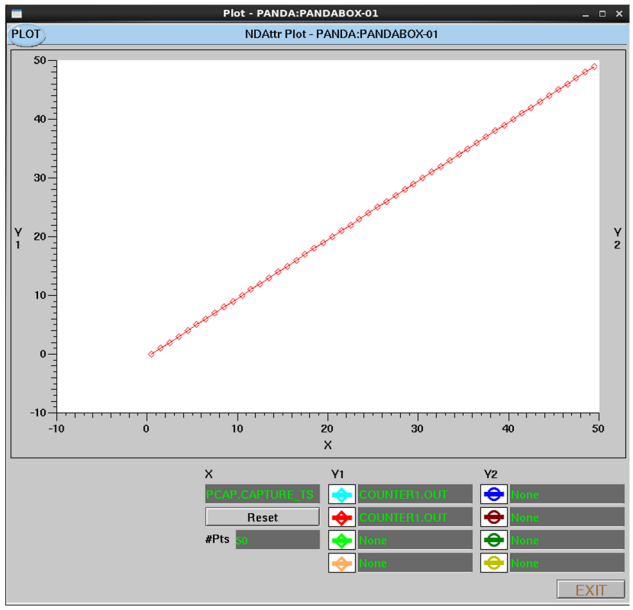


Figure 5: PandABlocks COUNTER block output.

### *TANGO Interface*

The Tango [12] interface will be Similar to the EPICS interface and is currently in development at SOLEIL.

## EXAMPLE APPLICATIONS

Some Proposed use cases are described including a "snake" or "bi-directional raster" scan with time based pulses, and averaging ADC values between position-based pulses.

### *Snake Scan With Time Based Pulses*

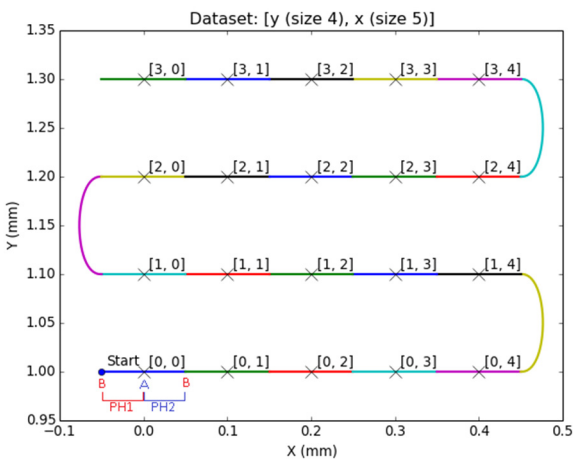


Figure 6: Snake scan with time based pulses.

| Ph1 t | Ph1 Out | | Ph2 t | Ph2 Out | |
|---|---|---|---|---|---|
| (ms) | A | B | (ms) | A | B |
| 100 | 0 | 1 | 100 | 1 | 0 |

Figure 7: Phase times for snake scan with time based pulses.

A snake scan with time based pulses can be set up by triggering position compare blocks at the start of each run. Two position compare blocks are used, one for each direction, a sequencer block is then used to generate the frame pulses, shown in Fig. 7, which trigger the detector on "TTLOUT1", and also generates the capture pulse. This allows the possibility to detect the start of the row and produce time based exposure triggers for the camera and capture in the middle of those frames as seen in Fig. 6. This is done for each row in both the forward and reverse directions. The block wiring described above is graphically shown as displayed on the web GUI screen in Fig. 8.

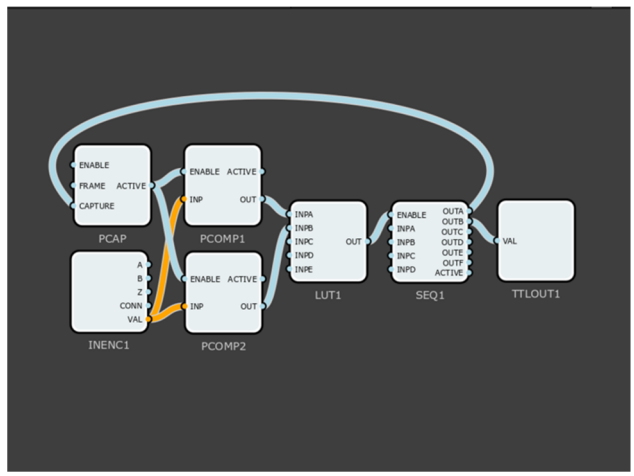


Figure 8: Block setup on web GUI for snake scan with time-based pulses.

### *ADC Averaging*

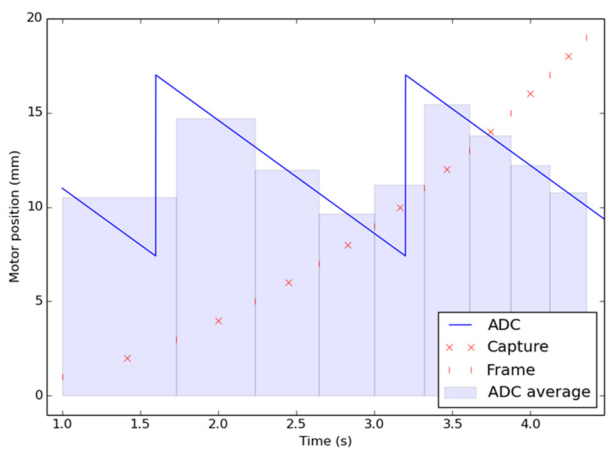


Figure 9: ADC averaging.

Figure 9 shows an application averaging an ADC over irregular frames. This could be used when it is necessary.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA068



to capture the beam current, and as the current isn't stable averaging is required. The frame signals come from the position compare block "PCOMP1", and a second "PCOMP2", is used for the capture signals. The position capture block "PCAP" averages over each frame if it contains a capture signal. The block wiring is shown in Fig. 10.

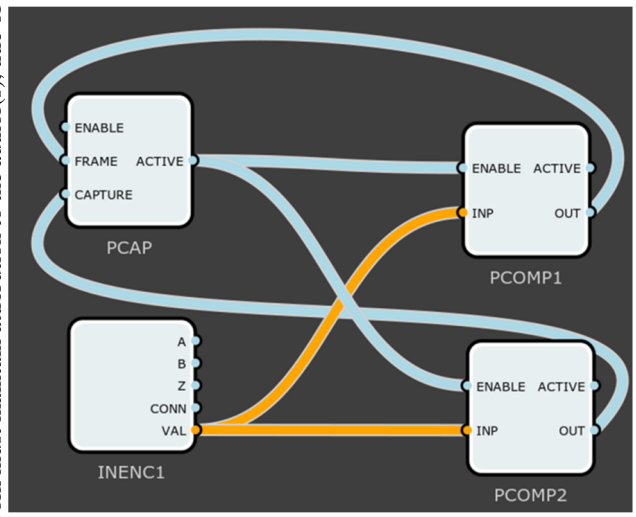


Figure 10: ADC Averaging block setup as seen on web GUI.

## CONCLUSION

The system, deployed on the PandABox device, has been rolled out and deployed at a number of beamlines at Diamond Light Source and is actively in use. There is currently support for the EPICS driver, currently used at Diamond, with the Tango driver still in development.

It is possible to obtain the full web stack and open FPGA framework from github and implement custom experiments and applications 'out of the box' with the standard IP blocks. It is also possible to develop custom blocks and integrate them into applications alongside the standard blocks supplied, which is helped by the simulation framework. The web GUI provides a simple and user friendly interface for configuring individual block parameters, viewing register states and values, and configuring the signal flow between blocks. Full configurations can be extracted and restored in JSON format, enabling version control of experiment configurations.

## REFERENCES

[1] S. Zhang *et al.*, "PandABox: A Multipurpose Platform for Multi-technique Scanning and Feedback Applications", presented at ICALEPCS'17, Barcelona, Spain, May 2016, paper TUAPL05, this conference.
[2] Diamond Light Source, http://www.diamond.ac.uk/.
[3] SOLEIL, https://www.synchrotron-soleil.fr/.
[4] PandABlocks, https://github.com/PandABlocks
[5] Xilinx, https://www.xilinx.com/.
[6] BusyBox, https://busybox.net
[7] ARM, https://developer.arm.com/products/processors/cortex-a/cortex-a9
[8] EPICS, http://www.aps.anl.gov/epics/.
[9] I. Gillingham and T. Cobb, "MalcolmJS: A Browser-Based Graphical User Interface", presented at ICALEPCS'17, Barcelona, Spain, May 2016, paper THPHA184, this conference.
[10] T. Cobb *et al.*, "Malcolm: A Middlelayer Framework for Generic Continuous Scanning", presented at ICALEPCS'17, Barcelona, Spain, May 2016, paper TUPHA159, this conference.
[11] areaDetector, https://github.com/areaDetector
[12] Tango , http://www.tango-controls.org/.