ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA075

FPGA-BASED BPM DATA ACQUISITION FOR LCLS-II

Till Straumann, Sonya Hoobler, Jeff Olsen,
Chengcheng Xu, Andrew Young, SLAC, Menlo Park, CA, USA

Abstract

The LCLS-II facility currently under construction at SLAC will be capable of delivering an electron beam at a rate of up to almost 1MHz. The BPM system (and other diagnostics) are required to acquire time-stamped readings for each individual bunch. The high rate mandates that the processing algorithms as well as data exchange with other high-performance systems such as MPS (machine-protection system) or bunch-length monitors are implemented with FPGA technology. Our BPM-processing firmware builds on top of the SLAC "common-platform" [1–3] and integrates tightly with core services provided by the platform such as timing, data-buffering and communication channels.

INTRODUCTION

A new BPM data acquisition system has been built for the LCLS-II XFEL which is capable of acquiring beam-position readings at a sustained bunch rate of up to 1MHz while resolving individual bunches. It can be configured for different types of BPM pickups (striplines, cavities) which require different processing algorithms. Due to the high beam rate these algorithms must be implemented in FPGA logic.

Some systems (e.g., machine-protection, MPS) intrinsically operate at the full beam rate whereas most diagnostics (BPMs among them) capture time-stamped readings into deep on-board buffers. Capturing can be triggered and configured in a globally synchronized way in co-operation with the timing system. The buffers are then read-out by software and can be correlated with other diagnostics by aligning time-stamps off-line. This facility is called *beam-synchronous acquisition* or BSA.

The system builds on top of the *SLAC Common Platform Architecture* [1–3] which is an ATCA carrier hosting a FPGA with a framework of core firmware blocks (communication protocol stack, JESD204, timing, machine-protection, deep BSA data buffers etc) which are common to several high-performance systems (BPMs, bunch-length monitors, machine-protection etc.).

BPM is considered a specific *application* on top of the common-platform. Application-specific AMCs (e.g., high-speed digitizers) can be mounted on the carrier and application-specific FPGA logic is integrated with the common-platform firmware.

In the following we shall describe the BPM-specific aspects of this system.

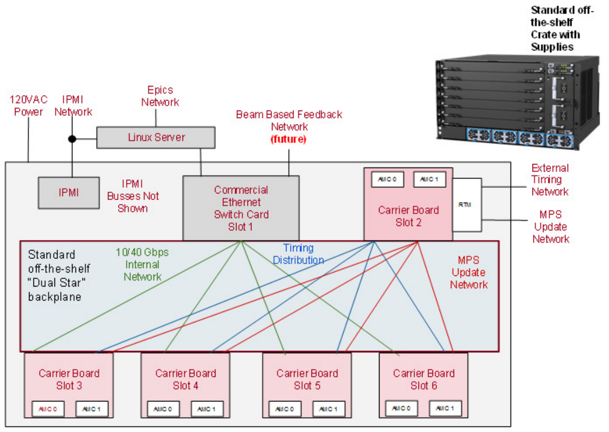

Figure 1: Block diagram of an entire ATCA shelf/crate which can host multiple AMC carriers each supporting different applications.

HARDWARE

Most of the BPM data acquisition hardware consists of common-platform components which can be shared with other subsystems (see also [1–3])

- COTS ATCA shelf with cooling, power-supplies, shelf-manager.
- COTS 10GigE switch blade (in slot 1).
- Timing distribution and MPS concentrator blade with RTM (rear transition module).
- Embedded linux server PC with connectivity to the 10GigE switch as well as the control-system LAN.

or are at least standardized

- Common platform carrier board.
- RTM with diagnostic connections.
- AMC modules (where applicable).

Figure 1 shows the topology of a complete ATCA shelf hosting multiple carriers and shared resources. The BPM subsystem we describe here can support up to two BPMs on a single carrier (i.e., more BPM carriers could be added to a single shelf).

Stripline BPM

Stripline BPMs measure position by detecting the amplitudes of the signals induced at two stripline pickups located at opposite walls of the beam pipe and computing the difference of these amplitudes (normalized to the sum of the amplitudes) [6]. Differential drift of gains in the two signal paths affect this “small difference of big numbers” type of measurement.

In order to overcome this problem a special AMC with the traditional SLAC on-line calibration circuit [4, 5], gain stages, filtering and high-speed ADCs was developed for

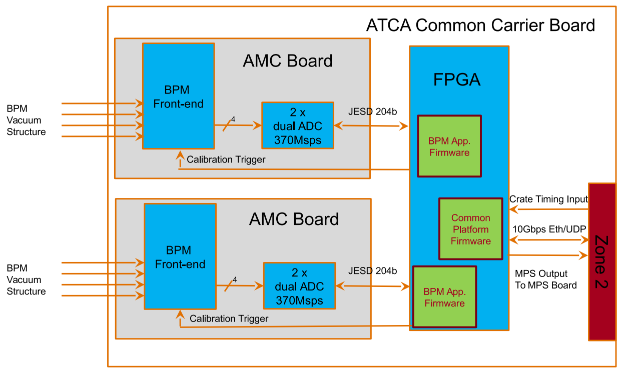

Figure 2: Block diagram of a single ATCA carrier with special-purpose AMCs for stripline BPM support (2 BPMs). The AMC features front-end electronics (gain, filters, attenuators), analog switches and a calibration tone generator as well as high-speed ADCs (jesd204b). These features are controlled from the main FPGA on the carrier.

stripline BPMs. A block diagram of the hardware is shown in Fig. 2.

The switches involved in the calibration process are controlled by BPM firmware which is integrated in the FPGA on the carrier card.

The sampling frequency for stripline BPMs is 370Mhz with an analog system bandwidth of 30MHz centered around 300Mhz (stripline length is approximately 4.5in).

Cavity BPM

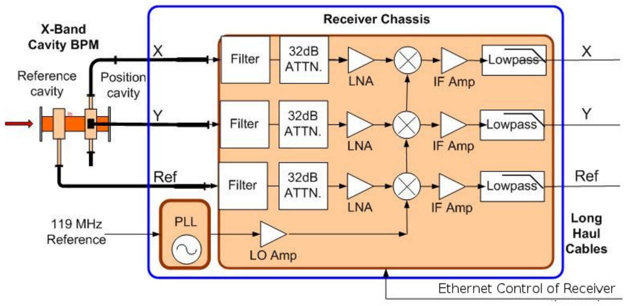

Figure 3: Block diagram of the X-band receiver chassis.

Analog preprocessing of cavity BPM signals (down-conversion from X-band) is implemented upstream of the ATCA system in a separate chassis (Fig. 3) which is installed in proximity to the cavities. Thus, for subsequent data acquisition a "multi-purpose" high-speed ADC AMC (which is also used by other applications) can be employed (see block diagram in Fig. 4).

The cavity BPM (f_0 = 11.424GHz, Q_L = 2000..3000) IF signal (f_{IF} = 40MHz) is sampled at 370Mhz. A 60MHz low-pass filter provides alias-rejection.

FIRMWARE

As a common-platform "application" the BPM firmware leverages the many services offered by the common firmware

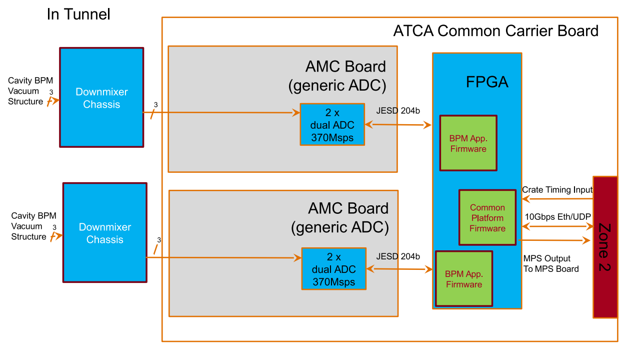

Figure 4: Block diagram of ATCA carrier with general-purpose ADC AMCs for cavity BPM support (2 BPMs). The front-end electronics including X-band mixers are hosted by a chassis which is located physically close to the cavities in the tunnel.

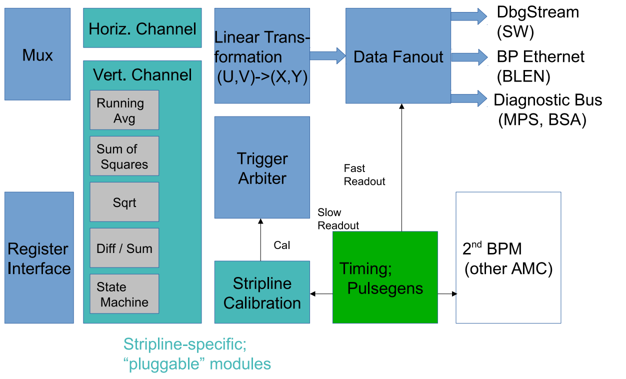

Figure 5: Block diagram of the BPM application firmware. Turqoise blocks are specific for stripline BPMs (detection algorithm) and are replaced by a different module when FW is instantiated for cavity BPMs. Blue blocks are common to all types of BPMs.

such as clock generation, register access, communication (UDP, JESD204b, AXI) just to name a few. In fact, the application firmware is just one block embedded into the common platform. It is presented with a number of interfaces:

- Clocking
- ADC sample stream
- timing message bus
- AXI interconnect (various streams as well as AXILite)

The BPM firmware uses the information from the timing message bus to generate several triggers and for obtaining timestamps which are attached to the processed readings.

The stream of raw samples along with triggers are supplied[1]to a "processing module" which implements the detection algorithm for specific types of BPM (stripline vs. cavity) as shown in Fig. 5.

Readings out of the processing module are post-processed (e.g., scaling, timestamps) and eventually forwarded to multiple data sinks:

[1] Via a multiplexer which permits permutations of channel assignments

- Decimated readings (along with the raw sample waveform from which they were computed[2]) are streamed to software.
- Full-rate readings are optionally sent to other HPS (high-performance systems) applications e.g., bunch-length measurement (BLEN) residing on a different carrier. Such messages are distributed via the switched 10G backplane ethernet.
- Full-rate readings are broadcast on an in-firmware "diagnostic bus" where they are e.g., picked up by MPS and BSA (parts of the common platform firmware [3]).

All processing is performed in fixed-point arithmetic with a precision of 18 bits (some intermediate results use a higher number of bits). The reasons (as opposed to a floating-point implementation) are less resource consumption and compliance with the BSA implementation which also uses fixed-point arithmetic for computing various averages.

While for BPM readings 18-bit fixed-point numbers are certainly adequate to cover their dynamic range it is nevertheless more cumbersome than handling floating-point numbers since close attention must be paid to proper scaling during various steps. Also, the end-results must be re-scaled to proper physical units.

All parameters of the algorithms and many diagnostics and slow-readouts are available on a AXILite register interface (the common-platform firmware eventually maps register access to network operations, i.e., the registers are accessible from ethernet).

Stripline BPM

Since stripline BPMs measure the (potentially small) difference of large signals a reasonable SNR is always required for operation and (phase-insensitive) amplitude detection is sufficient.

The stripline-BPM processing algorithm computes the RMS of a programmable number of samples after a trigger event occurs for each pickup electrode (Fig. 5). The acquisition window N is chosen such that the time-domain response has decayed into the noise floor for $i >= N$:

$$RMS = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}\left(x_i - x_{avg}\right)^2}$$

where the average x_{avg} is computed by a running-average filter. Due to the analog system's band-pass response the only contribution to this average is a DC-offset of the digitizer. This method is numerically less sensitive than subtracting the square of the average from the mean square

$$\sqrt{< x^2 > - < x >^2}$$

Position is then estimated by the well-known "difference over sum" algorithm [6] and the results are handed off (together with status information) to the generic processor.

[2] These waveforms are a valuable diagnostic!

Cavity BPM

Cavity BPMs are capable of delivering much higher resolution than their stripline counterparts [7]. Close to the center of the device the signal amplitude becomes very small so that any noise has a direct impact on resolution. Consequently, a phase-sensitive or synchronous detection algorithm is beneficial since it suppresses noise components which are in quadrature to the signal. An RMS-style detector would be inadequate since its sensitivity at very small amplitudes tends towards zero.

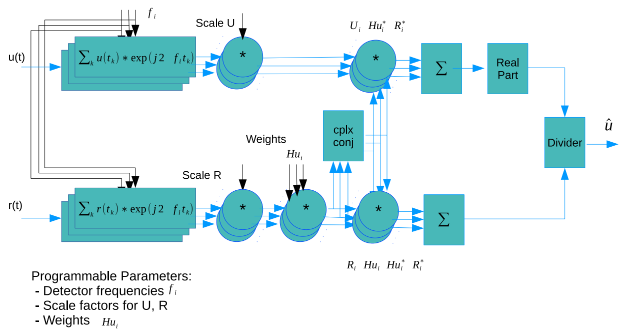

Figure 6: Block diagram of the cavity-BPM detection algorithm (replaces turqoise blocks in Fig. 5). The algorithm implements a complex correlator in the frequency domain. Note that only one axis (u) is shown.

The processing algorithm for cavities depicted in Fig. 6 (for a single axis, u, only) implements a small bank of N programmable Goertzel-DFT computers for each of the u, v (position-) and r (reference-) cavity signals which provide the frequency-domain representations U, V and R, respectively, over a small range of frequencies (which is chosen to cover the cavity response, of course).

E.g., with M input samples u_m

$$U_i = \sum_{m=0}^{M-1} u_m\, e^{jm\,\phi(i)}$$

where $\phi(i)$ is typically an integer multiple of $2\pi/M$.

The signal out of the position-cavities is proportional to the beam offset from the center and its charge whereas the reference signal only depends on the beam charge [7].

Because the reference and position signals are tuned to the same frequency the complex correlations

$$C_u = \textstyle\sum_{i=0}^{N-1} \frac{U_i R_i^*}{R_i R_i^*} \qquad C_v = \textstyle\sum_{i=0}^{N-1} \frac{V_i R_i^*}{R_i R_i^*}$$

yield a good estimate for the beam position. Note that C_u and C_v are *complex* quantities. Their phase-angle represents any difference in phase (e.g., due to cable-length differences) between the position- and reference- channels. This phase must be calibrated e.g., by using movers or steering the beam to a moderate, known off-center position. The position is then estimated as the real-part of the product of the correlations with complex scale factors:

$$\hat{u} = \Re\{A_u C_u\} \quad \hat{v} = \Re\{A_v C_v\}$$

The position estimates $\hat{u}$ and $\hat{v}$ denote unscaled, raw values; x and y are obtained by a linear transformation (scale/offset/rotation) as shown in Fig. 5.

All relevant parameters used by the firmware-algorithm are software-programmable.

SOFTWARE SUPPORT

The BPM data acquisition meets high-performance requirements as well as offering access to a traditional control system. The high-performance features include

- Full-rate streaming of pulse-by-pulse data
- Capture of full-rate data into on-board buffers for subsequent off-line analysis (BSA)

The first use case is entirely covered by in-firmware high-performance systems (such as feedbacks, MPS etc) because the high data rate precludes the use of software solutions. Dedicated communication channels exist in firmware for this purpose ("diagnostic bus" and "backplane (BP) ethernet" in Fig. 5).

The second use-case is covered by the LCLS-2 BSA subsystem which consists of firm- and software components. BSA is a common-platform feature and BPM can leverage this service by using firm- and software libraries.

Thus, the principal role of software is that of the traditional control-system

- Management of parameters for in-firmware algorithms
- Diagnostic and status readout
- Slow readout of position data for operator displays
- Setup, control and readout of BSA

The BPM software only needs to deal with the first three items. BSA is – as already mentioned – covered by libraries. Since a BPM software solution already had been developed for LCLS-1 the LCLS-2 firmware has been designed such that the LCLS-2 ATCA system can be easily integrated into the existing BPM software.

Common Platform Support

All communication between software and firmware is ethernet based and uses a suite of different protocols on top of UDP [1–3] for register access, streaming of data, bulk memory transfer etc.

However, the common-platform software (CPSW) framework (c++) shields the user application from all these details and presents an abstracted view of the firmware entities which can be introspected and manipulated. CPSW is configured for a particular firmware by means of a description of the firmware in YAML [8] format which is produced by the firmware generation process.

All software described as "new components" in the next section (blue boxes in Fig. 7) builds on top of CPSW and does not have to deal with protocols, bit-manipulations etc. as they are often performed by traditional drivers.

E.g., if a particular bit in a register needs to be changed then the exact offset etc. is described in YAML and automatically handled by CPSW which makes this bit available as a "value" with a hierarchical name.

A driver then just looks up the name of the target entity and uses access operations (such as e.g., `getVal()/setVal()`) for controlling it.

Control System Integration

EPICS based BPM-IOC software has been in use at SLAC (LCLS-1) for many years and a wide variety of higher-level (e.g., matlab) applications exist using the EPICS PV interface.

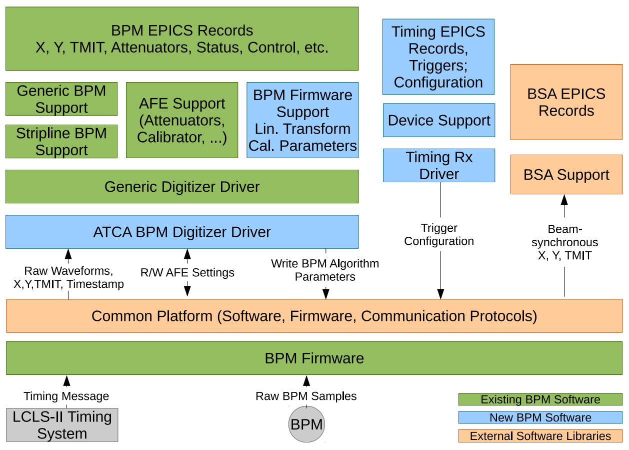
BPM EPICS Records
X, Y, TMIT, Attenuators, Status, Control, etc.
Timing EPICS Records, Triggers; Configuration
BSA EPICS Records
Generic BPM Support
Stripline BPM Support
AFE Support (Attenuators, Calibrator, ...)
BPM Firmware Support Lin. Transform Cal. Parameters
Device Support
Timing Rx Driver
BSA Support
Generic Digitizer Driver
ATCA BPM Digitizer Driver
Trigger Configuration
Beam-synchronous X, Y, TMIT
Raw Waveforms, X,Y,TMIT, Timestamp
R/W AFE Settings
Write BPM Algorithm Parameters
Common Platform (Software, Firmware, Communication Protocols)
BPM Firmware
Timing Message
Raw BPM Samples
LCLS-II Timing System
BPM
Existing BPM Software
New BPM Software
External Software Libraries

Figure 7: Block diagram of the BPM IOC software.

The BPM-IOC software of which a block diagram is shown in Fig. 7 already supports a variety of different digitizers and types of interconnect (VME, PCIe, ethernet, ...) which made integrating a new digitizer straightforward. It was therefore decided to keep the existing software (green blocks in Fig. 7) and treat the slow readout channel of the BPM firmware which provides raw digitizer samples of highly decimated beam pulses as "just another digitizer" (supported by a new driver; one of the blue blocks in Fig. 7). Thus, decimated BPM readings along with raw pickup waveform displays (which are a valuable diagnostic) are available via the existing BPM software. Note that even slow readings can be globally synchronized – by means of the timing system – because the BPM firmware uses a dedicated trigger for such readings.

Full-rate readings are available from the LCLS-II BSA facility (orange blocks in Fig. 7 which is a common (i.e., not BPM-specific) service (consisting of common-platform firmware, low-level and EPICS-level software libraries).

CONCLUSION

The LCLS-II BPM system leverages the services of the SLAC common platform to implement high-speed position acquisition and to interact with other high-performance systems such as MPS or BSA. Slow controls and diagnostics are governed by the existing LCLS BPM software with which the FPGA-based acquisition system easily integrates.

ACKNOWLEDGEMENT

Work supported by U.S. Department of Energy Contract No. DE-AC02-76SF00515.

REFERENCES

[1] J. Frisch *et al.*, “A FPGA Based Common Platform for LCLS2 Beam Diagnostics and Controls,” in *Proceedings of IBIC2016*, Barcelona, Spain, 2016, paper WEPG15.

[2] T. Straumann *et al.*, “New Controls Platform for SLAC High-Performance Systems,” in *Proceedings of PCAPAC 2016*, Campinas, Brazil, 2016, paper THHWPLCO02.

[3] T. Straumann *et al.*, “SLAC’s Common-Platform Firmware for High-Performance Systems,” in *Proceedings, 16th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS’17)*, Barcelona, Spain, 2017, paper THMPL08.

[4] T. Straumann *et al.*, “LCLS Beam-Position Monitor Data Acquisition System,” in *Proceedings, 11th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS’07)*, Knoxville, USA, October 2007, paper WPPB40.

[5] E. Medvedko *et al.*, “Stripline Bpm-Position Monitors for LCLS,” in *Proceedings of BIW08* Tahoe City, USA, 2008.

[6] P. Forck, P. Kowina and D. Liakin, “Beam Position Monitors,” in *CERN Accelerator School: Course on Beam Diagnostics, CERN-2009-005.187*, Dourdan, France, 2008.

[7] R. Lorenz, “Cavity Beam Position Monitors,” in *Proceedings of the 1998 Beam Instrumentation Workshop (BIW98)*, SLAC, Menlo Park, USA, 1998.

[8] O. Ben-Kiki, C. Evans and I. döt Net, “YAML Ain’t Markup Language,” `http://yaml.org/spec/1.2/spec.html`