Pixel readout chip software emulators for the YARR DAQ system upgrade

N Whallon[1], T Heim[2], M Garcia-Sciveres[2], K Potamianos[3], H Oide[4] and S-C Hsu[1]

[1] Department of Physics, University of Washington, 3910 15th Ave. NE, Seattle, WA 98195-1560, USA,
[2] LBNL, 1 Cyclotron Rd., Berkeley, CA 94720, USA,
[3] DESY, Notkestraße 85, 22607 Hamburg, Germany,
[4] INFN, 4 Via Monti Vincenzo, Genova, GE 16156, Italy

E-mail: alokin@uw.edu, timon.heim@cern.ch, mgarcia-sciveres@lbl.gov, karolos.potamianos@cern.ch, Inhideyuki.Oide@cern.ch, schsu@uw.edu

Abstract. The Yet Another Rapid Readout (YARR) system is a DAQ system designed for pixel readout chips including the current generation ATLAS FE-I4 chip and the next generation RD53A chip, which is part of the development of new Pixel detector technology to be implemented in High-Luminosity Large Hadron Collider experiments. YARR utilises a PCI-e FPGA card which acts as a simple gateway to pipe all data from pixel readout chips via the high speed PCI-e connection into the host system’s memory. All data processing is done on a software level in the host CPU(s), utilising a data-driven, multi-threaded, parallel processing paradigm. YARR has recently been upgraded to interface with software emulators of pixel readout chips. These emulators offer many benefits: quick development of DAQ software; expansion of the developer base to users without access to readout hardware; preparation of DAQ software for upcoming readout chips, such as RD53A; implementation of Continuous Integration and unit tests to ensure code quality and maintainability. The design and capabilities of the FE-I4 and RD53A software emulators will be presented.

1. Introduction

Beyond Run 3, the Large Hadron Collider (LHC) will undergo a major upgrade to produce high instantaneous luminosity. This High-Luminosity LHC (HL-LHC) will be able to deliver 3000 fb$^{-1}$ at $\sqrt{s} = 14$ TeV within 10 years after it begins operations.

The increased luminosity will also result in an increased number of pileup interactions and increased radiation doses to detectors. These factors motivate a complete replacement of the ATLAS [1] inner tracker. The new inner tracker, ITk [2], will require a finer granularity than before in order to resolve the increased number of pileup vertices. As such, the readout bandwidth required by the new tracker will increase.

RD53A [3] is a readout chip which meets the criteria of HL-LHC pixel detectors. The YARR [4] data acquisition (DAQ) system is being developed to provide robust data taking and calibration software for RD53A. Before RD53A chips are produced, it is possible to develop the YARR DAQ for RD53A using software emulators. The design and capabilities of pixel readout chip software emulators are described in the following sections.
2. Yet Another Rapid Readout
The Yet Another Rapid Readout (YARR) system is a robust data acquisition (DAQ) system targeting R&D of next generation pixel readout chips, such as RD53A. It is also capable of interfacing with the current generation readout chip FE-I4 [5]. YARR DAQ software runs on a host computer which interfaces with the readout chips via a commercially available PCI-e FPGA card. The custom FPGA firmware implements only a basic buffer and communication blocks. The host computer handles running scan loops, sending commands to the readout chip, and data analysis on the data read back from the readout chip. Since all data processing is handled in software, it is possible to interface YARR also directly with software emulators of readout chips. These emulators need to be able to interpret commands sent by YARR, and produce and send back sensible hit data.

Software emulators of the FE-I4 and RD53A chips are in development primarily as a tool to aid in the development of YARR software. This is especially the case for RD53A, whose specifications are available but which has not yet been produced. The emulators also allow for users without physical readout chips to join the development efforts. Finally, the emulators enable the use of Continuous Integration (CI) for the YARR DAQ software, something which is simply not feasible if hardware is required, and which will ensure code quality and maintainability.

The basic structure of YARR is illustrated in Figure 1.

![Figure 1. The basic strucrure of the YARR DAQ system.](image)

3. Readout Chip Software Emulators
Software emulators of readout chips must be able to interface directly with YARR, correctly interpret the readout chip-specific commands sent by YARR, and return hit data to YARR when trigger commands are received.

The behavior of readout chips depends on values stored in Global Registers (GR) and Pixel Registers (PR). The GRs contain chip-level settings, such as global threshold voltages or pixel masking information, while the PRs contain pixel-level settings, such as local threshold voltage.
or local pixel enable. As such, the software emulators instantiate readout chip objects with virtual GRs and PRs in order to store the configuration of the readout chip. These registers are then configured when the emulators receive the appropriate commands from YARR.

When YARR sends trigger commands, the emulators loop over a virtual pixel array, model hit data, and send this data back to YARR in a format identical to what the real readout chips would send. YARR is then able to do the histogramming, analysis, and plotting with this data.

The basic structure of readout chip software emulators is illustrated in Figure 2. The specifics of decoding commands from YARR, configuring virtual GRs and PRs, and modelling hit data for the FE-I4 and RD53A software emulations are presented in the following subsections.

3.1. **FE-I4 Command Decoding and Register Configuration**

Configuring FE-I4 GRs is straightforward: the FE-I4 emulator receives a `WrRegister` command from YARR, which contains the address of the GR to write to, accompanied by the value to write to that GR. PRs, on the other hand, are configured via a Shift Register (SR), which requires a few extra commands and also relies on values stored in the GR. All PR and SR manipulation is done in double column (DC) loops in order to configure multiple pixels at the same time. How the pixel double columns are looped over is dictated by certain values in the GR. This command decoding and register configuration must all be done to the precise FE-I4 chip specifications by the emulator. A block diagram outlining the decoding is shown in Figure 3.

3.2. **RD53A Command Decoding and Register Configuration**

The scheme for configuring registers in RD53A is similar to FE-I4, with the exception of the SR. Instead of using a SR to configure pixels, RD53A has a virtual register which points to the address of specific pairs of pixels. If an RD53A `Write_Register` command specifies this virtual register as the register to write to, the paired value will be written directly to the PRs of the pixel pair pointed to by the virtual register, otherwise the paired value will be written directly to that GR. The command decoding and register configuration scheme used by the RD53A emulator is illustrated in Figure 4.

![Figure 3. The FE-I4 emulator command decoder.](image)

3.3. **FE-I4 Hit Modelling**

After chip configuration, scans typically send a series of commands to inject charge into the digital or analog front-end circuitry of pixels and triggers to read back their response. A simplified pixel analog front-end circuit is shown in Figure 5. A threshold voltage and the voltage resulting from the injected charge are routed into a comparator and the pixel will register a hit if the injected charge is above the threshold.

For the FE-I4 chip, the threshold voltage is determined by a local, per-pixel, threshold value combined with a global, per-chip, threshold value, stored in the PRs and GR, respectively. The
main hit metric is the time-over-threshold (ToT). Figure 6 shows an illustration of this process. The FE-I4 per-pixel feedback current adjustment is not modelled in the emulator and a linear charge to ToT conversion is assumed.

The FE-I4 emulator models hit data by comparing a fixed injection charge to a charge derived from the global and local threshold values stored in the GR and PRs, respectively. Per-pixel Gaussian smearing is applied to the local threshold value and Gaussian noise is applied to the injection charge, with the mean and standard deviation of the noise configurable per-pixel.

3.4. RD53A Hit Modelling
Hit modelling for RD53A is very similar to FE-I4 with the exception of which threshold values are compared to the injection charge. RD53A implements 3 different analog front-end designs, and the RD53A emulator can currently models 2 of the designs, the linear and differential front-end design. The linear front-end design and modelling is identical to that of the FE-I4. A simplified schematic of the differential front-end design is shown in Figure 7.

ToT for the differential front-end design is calculated by shifting the signal from the injection charge and comparing the resulting signal to the original signal inverted and shifted by a different, larger, offset. This is illustrated in Figure 8.

The RD53A emulator applies per-pixel Gaussian smearing to the 2 local threshold values and per-pixel Gaussian noise is applied to the injection charge.

3.5. Software Emulator Scan Results
Interpretation of all commands necessary to run many scans has been implemented into the FE-I4 emulator. Some of these scans include digital scans, analog scans, noise scans, threshold scans, and threshold tunes. As a demonstration of the emulator’s capabilities, threshold scans before and after threshold tunes are shown in Figure 9.
The Gaussian smearing applied to the per-pixel local threshold values and noise is evident in the Gaussian shape of the pixel threshold distributions. After tuning the local and global threshold values, a much more narrow Gaussian threshold distribution can be achieved, centered around a user-determined value. This mirrors the behavior of real FE-I4 chips.

Figure 7. Simplified schematic of RD53A differential analog front-end.

Figure 8. RD53A differential front-end ToT illustration.

Figure 9. Threshold scans before (left) and after (right) applying a per-pixel threshold tune, using the FE-I4 software emulator.

4. Continuous Integration

One major benefit of software emulators is the ability to implement Continuous Integration (CI) for the YARR DAQ software. The CI workflow is illustrated in Figure 10.

When a developer pushes a commit of the YARR DAQ code, a GitLab [6] Runner will automatically check out the YARR repository, compile the YARR software, run a series of test scans using the software emulators, and check the resulting data. If there was an error in compilation, running the scans, or the resulting data does not match what it expected, CI will mark the commit as having failed, and notify the user. The user is then able to see which step failed, correct for the failure, and push a new commit. This system is aimed at increasing code quality and maintainability and would not be feasible if YARR had to interface with physical hardware readout chips to perform the tests.
5. Conclusion
A software emulator which interfaces with YARR has been successfully implemented for FE-I4 and one for RD53A is under development. The emulators are being used to enhance the DAQ software, to prepare YARR to interface physical RD53A chips when produced, and to ensure code quality and maintainability via the implementation of Continuous Integration.

Acknowledgments
The authors would like to acknowledge the support of the Department of Energy Office of Science Graduate Student Research Program which made this work possible.

References
[1] ATLAS Collaboration et al 2008 The atlas experiment at the cern large hadron collider JINST 3 S08003
[2] ATLAS Collaboration et al 2017 Technical design report for the atlas inner tracker strip detector CERN-LHCC-2017-005 ATLAS-TDR-025
[3] Garcia-Sciveres M 2015 Rd53a integrated circuit specifications CERN-RD53-PUB-15-001
[4] Heim T 2016 Performance of the insertable b-layer for the atlas pixel detector during quality assurance and a novel pixel detector readout concept based on pcie CERN-THESIS-2016-085
[5] Zivkovic V et al 2012 The fe-i4 pixel readout system-on-chip resubmission for the insertable b-layer project JINST 7 C02050
[6] GitLab Continuous Integration (GitLab CI) https://docs.gitlab.com/ce/ci/  Accessed October 9, 2017