YARR - A PCIe based Readout Concept for Current and Future ATLAS Pixel Modules

To cite this article: Timon Heim 2017 J. Phys.: Conf. Ser. 898 032053

View the article online for updates and enhancements.
YARR - A PCIe based Readout Concept for Current and Future ATLAS Pixel Modules

Timon Heim
Lawrence Berkeley National Lab, USA
E-mail: theim@lbl.gov

Abstract. The Yet Another Rapid Readout (YARR) system is a DAQ system designed for the readout of current generation ATLAS Pixel FE-I4 and next generation chips. It utilises a commercial-off-the-shelf PCIe FPGA card as a reconfigurable I/O interface, which acts as a simple gateway to pipe all data from the Pixel modules via the high speed PCIe connection into the host system’s memory. Relying on modern CPU architectures, which enables the usage of parallelised processing in threads and commercial high speed interfaces in everyday computers, it is possible to perform all processing on a software level in the host CPU. Although FPGAs are very powerful at parallel signal processing their firmware is hard to maintain and constrained by their connected hardware. Software, on the other hand, is very portable and upgraded frequently with new features coming at no cost. A DAQ concept which does not rely on the underlying hardware for acceleration also eases the transition from prototyping in the laboratory to the full scale implementation in the experiment. The overall concept and data flow will be outlined, as well as the challenges and possible bottlenecks which can be encountered when moving the processing from hardware to software.

1. Introduction
In detectors at high energy particle colliders, like the ATLAS experiment [1] at the Large Hadron Collider at CERN, tracking detectors are responsible for measuring the trajectory of particles emerging from the interaction point. The innermost four layers of the ATLAS detector are composed of the Pixel detector [2], which has 92 million silicon pixels distributed over 2192 Pixel modules. These modules are built in a hybrid technique with a dedicated silicon sensor where each pixel is connected in a flip chip process to a custom readout chip manufactured in CMOS technology, which contains the amplifier and digitisation circuitry. The readout system presented in what follows will concern itself with the readout and calibration of these types of readout chips, specifically the FE-I4 readout chip [3] which is used in the latest upgrade of the ATLAS Pixel detector.

The Yet Another Rapid Readout (YARR) system [4] is a DAQ system designed for the readout of the current generation and scalable to the next generation of Pixel detector readout chips. It facilitates a strong emphasis on data processing in software compared to the traditional processing performed in FPGAs, which relaxes the dependency on the used hardware and utilises modern day multi-core CPU architectures. An essential part of this readout system is played by the PCIe bus, which enables the high bandwidth and low latency transfer of raw Pixel module data into the host system’s memory. This has several advantages over traditional architectures. Most of the essential functionality of the readout system is contained in the
high-level software, which is a more familiar domain for the scientists using these devices. In addition the hardware, specifically the FPGA, is only used as a reconfigurable I/O interface, therefore the entanglement between firmware and software is minimal. This enables YARR to be used with different hardware platforms, which are specialised for their field of operation. In what follows the case of a PCIe FPGA card is discussed, which is a promising platform for laboratory usage where only a small number of devices need to be read out at the same time, but the hardware should not be too expensive and be readily available from commercial vendors.

2. Concept
Other existing readout systems for ATLAS Pixel modules use a readout architecture as is depicted in Figure 1a. Particular to that architecture is that the scan engine (or parts of it), which drives the calibration, are not running on the host computer, but in the custom Read-Out-Driver (ROD) connected to the Pixel module. Additionally the first data processing step is also performed on the ROD, typically the type of processing is histogramming which is effectively a lossy data compression. This is usually born out of necessity, because the communication interface between the ROD and host computer does not have enough bandwidth to transfer all of the data or the latency is too high to drive the scan directly from the host computer.

![Diagram](image)

(a) Traditionally used readout architecture.
(b) Readout architecture enabled by YARR.

Figure 1: Comparison of the traditional readout architecture currently shared by various systems for the readout of Pixel modules and the concept implemented in YARR. [4]

This traditional architecture results in several issues:

- In order for the ROD to perform these specific tasks, its hardware is very specialised. In order to perform the histogramming with high enough performance external SRAM is used. To facilitate the complex logic necessary to run the scan engine, processing units like a DSP or a PPC are used. This leads to the readout hardware being highly specialised and is often custom designed, therefore not available from commercial vendors which can lead to supply issues.
- Due to all the functions performed by the ROD, the software running on the host computer is deeply entangled with the firmware, which in return is built around the custom hardware. This results in the high level software only being useable with only one particular hardware platform, but usually multiple hardware platforms exist due to different operation conditions in the laboratory and the detector.
• The scientists operating these readout systems are very comfortable with high-level programming languages like C++, but many parts of the scan execution and processing are buried in firmware, making it less accessible to the users. Over time this leads to the issue that there are less developers available to maintain the firmware. This is amplified by the fact that all readout systems use very different code bases.

YARR tries to solve these issues by deploying a different readout concept, as shown in Figure 1b. In contrast to the traditional architecture all of the data processing and scan execution are performed by the host computer. This is made possible by the high speed, low latency connection to the PCIe FPGA card. The FPGA firmware is kept simplistic and only contains the minimum amount of logic to communicate with the Pixel Module. The performance of the software, which now contains all of the intelligence of the readout system, is increased by making heavy use of multithreading, enabled by modern multi-core CPUs found in every common desktop computer.

3. Hardware and Firmware

The hardware chosen for the first implementation of YARR is the Simple PCIe Carrier (SPEC) [5] card, shown in Figure 2. It features a Xilinx Spartan 6 FPGA, which is suitable for the readout of the FE-I4 chip, and a local PCIe bus bridge to ease the development of the PCIe FPGA interface. The FMC connector can be used for custom adapter boards to interface with up to 16 Pixel Modules. Important for the choice of this card is its low cost and high availability from commercial vendors.

A block diagram of the firmware is shown in Figure 3. The GN4124 core translates the data received on the PCIe bus to the Wishbone bus, this enables the communication with the rest of the blocks attached as slaves to the Wishbone bus. A bitstream which should be sent to an FE-I4 is written into the TxCore. If a bitstream is thought to be sent periodically then the Trigger Unit can be configured to send it with a programmable frequency for a programmable time or
amount of times. This is typically used to send multiple injection and trigger commands during scans. Data which is sent back from an FE-I4 is aggregated and buffered in DDR3 memory. Once a package of suitable size has been collected, it is transferred with a scatter-gather DMA transfer into the host system’s memory. The performance achieved by the DMA transfer is shown in Figure 4 and saturates around 400 MB/s depending on the computer hardware. This is around 50\% of the maximum bandwidth of the GN4124 and is due to the scatter-gather mechanism which requires fetching of a new host memory address for packages of more than one page (4 kB). However this bandwidth is theoretically sufficient for the readout of up to 25 FE-I4 chips and since only 16 FE-I4 chips can be read out in parallel due to the limited number of I/Os, the achieved performance is deemed to be sufficient.

4. Data Flow
Testing and calibration of the FE-I4 chip is performed with scans. These scans most commonly consist of three parts: one typically being the iteration over a parameter setting, a loop over the pixel matrix enabling only portions of it, and a trigger loop which injects and reads out test charges. As traditional architectures performed execution of these loops partially in hardware, the loop structure and depth was fixed. This limitation is not necessary when all loops are executed in software, therefore YARR implements a dynamic nest loop structure as shown in Figure 5. The advantage of this implementation is that the atomic component of a scan becomes the loop, and scans are simply a collection of loops which can be nested in any desired order.

An overall diagram of the software layout showing the data flow is shown in Figure 6. The loop structure discussed above is running in the scan engine and will iterate over parameter settings and activate portions of pixels, in which it then injects test charges and triggers the

Figure 3: Block diagram of the firmware used for the readout of FE-I4 chips with YARR. Blocks in red are external chips. [4]
Figure 4: Result of a DMA transfer benchmark to determine the minimum package size for efficient DMA transfer. [4]

Figure 5: Dynamic nested loop structure used in YARR for the implementation of scans. [4]

data readout. The raw data fragments are stored in storage containers from where they can be picked up by a data processor. Multiple threads of the data processor can work on different raw data fragments at the same time, they decode the raw data into event data and split it up into one container per FE-I4 being read out. For further processing each FE-I4 has its own histogramming and analysis thread. All communication between threads is based on the exchanged data. To improve performance data is processed as soon as it comes into the host system’s memory and not only when all iterations over a parameter setting are done. For this reason metadata is included in each data fragment, which gives enough information to piece the data in the analysis back together.
Figure 6: Diagram depicting the different stages of the data processing and showing the usage of multi-threading architectures. Each blue box is a different thread, except for the tuning feedback no inter-thread communication is needed, all the information a thread needs is contained in the data objects in form of metadata.[4]

An important feature for calibration is the tuning feedback. During calibration an algorithm in the analysis searches for an optimal parameter setting to, for instance, tune the threshold of all pixels. In order to make the calibration faster, and not just scan over a fixed window of settings, search algorithms can be used to steer the setting towards its optimum, but they require feedback from the analysis to the scan engine. This is the only occurrence of inter-thread communication in YARR and is implemented by a specific loop action which waits for feedback from the analysis before starting a new scan iteration with the new parameter.

5. Considerations for Future Readout Chips
YARR is not only designed to support the already existing readout chip generation, but also the future generation which is currently being developed by the RD53 collaboration [6] for ATLAS Phase 2 upgrade [7]. The most important change for the DAQ system is the increase in readout bandwidth from 160 Mbps to 5 Gbps being transmitted by the readout chip to the DAQ. On the one hand this requires a new PCIe card with a faster FPGA, the Xilinx Series 7 FPGAs have special transceivers which are capable of receiving at 5 Gbps, but there are only a limited number per FPGA compared to the abundance of generic differential I/Os. The maximum bandwidth by the PCIe bus version 3.0 is 7.8 Gbps per lane with up to 16 lanes per PCIe endpoint.

A possible bottleneck in the software could be the histogramming of the data in computer DRAM memory. For each pixel hit received by the histogrammer, three actions are performed:
the current histogram value is read from memory, the value is summed with input, and the result is written back into the memory. The pixel hits can be assumed to be randomly distributed, i.e. random memory addresses are being read, which is very inefficient in DRAM which is optimised towards reading larger consecutive chunks of memory. This inefficiency might be amplified by the fact that there are multiple histogrammer threads running, all accessing very different regions of memory.

Figure 7 shows a benchmark where an increasing number of threads are histogramming randomly generated pixel hits. The total amount of hits per thread is constant and the average time to histogram a single hit is measured. As can be seen from the results, if more threads access the memory at the same time the average histogramming time per hit increases. It strongly depends on the exact data format of the hit sent by the readout chip, but it can be assumed that position and the digitised charge value require something of the order of 24 bit, which equates to hits arriving every 5 ns at a transmission speed of 5 Gbps. This is around the average time needed to histogram the hit on a standard desktop computer with dual channel memory, when 10 histogramming threads are running in parallel. It should be noted that this benchmark did not optimise the memory usage. A possible solution to bypass this bottleneck is to perform the histogramming in a GPU, which typically use faster memory interfaces than CPUs.

Figure 7: Histogramming benchmark performed on different types of computers. The total number of hits processed by each thread is kept constant, to simulate an increasing numbers threads accessing the memory at the same time.[4]

6. Conclusion
YARR is a readout system for ATLAS Pixel modules which focuses on software for scan execution and data processing. Thereby it reduces the dependency on DAQ hardware, which reduces cost and increases availability by using commercial-off-the-shelf hardware. The software heavily utilises multi-threaded processing making efficient use of common modern day multi-core CPUs. It has been implemented for the current generation FE-I4 readout chip, but is suitable to be used by the next generation of readout chips currently being developed. The concept used in
YARR is proposed as a core to enable the usage of the software in the laboratory and the final detector, something which has not been possible in the past due to entanglement of software, firmware and, hardware. This broadens the user base which is familiar with the software core and will help with the transition from module testing to detector operation.

References
[1] ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider, *JINST* 3 (2008) S08003.
[2] Aad G et al., ATLAS Pixe Detector electronics and sensors, *JINST* 3 (2008) P07007.
[3] Garcia-Sciveres M et al., The FE-I4 pixel readout integrated circuit, *Nuclear Instruments and Methods in Physics Research Section A*, 636, Issue 1, Supplement, 2011.
[4] Heim T, Performance of the Insertable B-Layer for the ATLAS Pixel Detector during Quality Assurance and a Novel Pixel Detector Readout Concept based on PCIe, University of Wuppertal, 2015, Dissertation, http://www.digibib.net/permalink/468/UBWU-x/HBZ:HT019058001.
[5] http://www.ohwr.org/projects/spec/wiki
[6] http://rd53.web.cern.ch/
[7] ATLAS Collaboration, Letter of Intent for the Phase-II Upgrade of the ATLAS Experiment, 2012, CERN-LHCC-2012-022