SYNCHRONISING HIGH-SPEED TRIGGERED IMAGE AND META DATA ACQUISITION FOR BEAMLINES

N. De Maio, A. P. Bark, T. M. Cobb, J. A. Thompson
Diamond Light Source Ltd., Didcot OX11 0DE, UK

Abstract

High-speed image acquisition is becoming more and more common on beamlines. As experiments increase in complexity, the need to record parameters related to the environment at the same time increases with them. As a result, conventional systems for combining experimental meta data and images often struggle to deliver at a speed and precision that would be desirable for the experiment. We describe an integrated solution that addresses those needs, overcoming the performance limitations of PV monitoring by combining hardware triggering of an ADC card, coordination of signals in a Zebra box [1] and three instances of areaDetector streaming to HDF5 data. This solution is expected to be appropriate for frame rates ranging from 30Hz to 1000Hz, with the limiting factor being the maximum speed of the camera. Conceptually, the individual data streams are arranged in pipelines controlled by a master Zebra box, expecting start/stop signals on one end and producing the data collections at the other. This design ensures efficiency on the acquisition side while allowing easy interaction with higher-level applications on the other.

INTRODUCTION

The kinds of data rates coming out of a typical modern synchrotron beamline continue to push up against the current limits in experimental complexity, processing power and storage space. Ongoing improvements in automation and computing infrastructure help scientists get past those limits, decreasing the time individual scans of a sample take to set up and carry out.

When it comes to taking data, a number of well-established technologies and beamline designs provide stable frameworks to acquire as much as possible across a number of different devices and channels. Each of these comes with its own requirements with regards to configuration and operation. Each of these may or may not also provide its own native mechanisms to sample data at user-defined regular intervals.

Getting these devices to perform their measurements in a synchronised way poses some unique challenges. If the samples are taken a sufficient time apart, the middleware layer provided by the controls software can, and often does, take on the role of coordinator. But there is a point where the interval between two samples gets too small to guarantee accurate time stamps. Different devices may also expect different kinds of triggers, and every additional layer of software mediating between those makes the delays between the data acquisition system and the hardware more unpredictable. Figure 1 illustrates this problem.

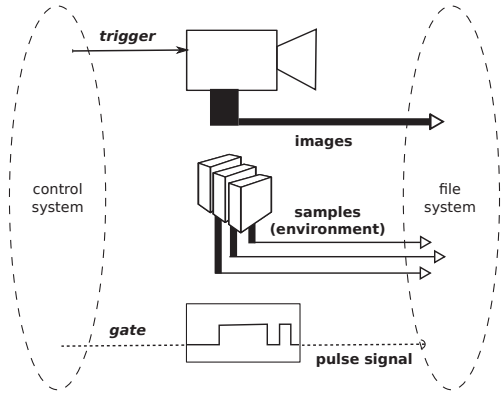

Figure 1: Unsynchronised trigger and data flow during image acquisition.

ARCHITECTURE

The answer to coordinated sample acquisition at small time scales is to use hardware triggering. Using a Zebra box [1], we present a design for a fast data acquisition system intended for sampling rates within the range of 30Hz and 1kHz. It not only coordinates the different hardware triggers involved but also bundles the acquired samples in three parallel pipelines, resulting in three files containing measurements for the same time frame.

Figure 2 gives an overview of the architecture. As Diamond Light Source (DLS) uses EPICS [2] as its control system, we rely on the areaDetector framework [3] and the EPICS implementation of the Zebra driver for configuring and collecting data from the underlying devices. We also rely on areaDetector plugins to post-process this data to some extent and to redirect it to files.

The control signal flow in our model is driven by the Zebra box. In our beamline-specific application, it receives an external trigger signal of its own. That signal causes the Zebra to produce one or more outgoing pulses, which are then forwarded to the connected devices. The incoming pulse need not be another hardware signal. Arm or Acquire commands to the relevant PVs are just as suitable because it is the configuration of the FPGA logic in the Zebra that ultimately decides what kinds of pulses to send out in response, and how many.

If individual devices expect different pulse shapes as trigger signals, the Zebra box acts as a mediator between incoming and outgoing triggers while ensuring that all outgoing pulses are sent off at the same time. If they are capable of receiving external triggers, they should be set to operate in this way. If they are not, the trigger signal needs to be recorded on one of the data channels so it can be extracted

ISBN 978-3-95450-148-9

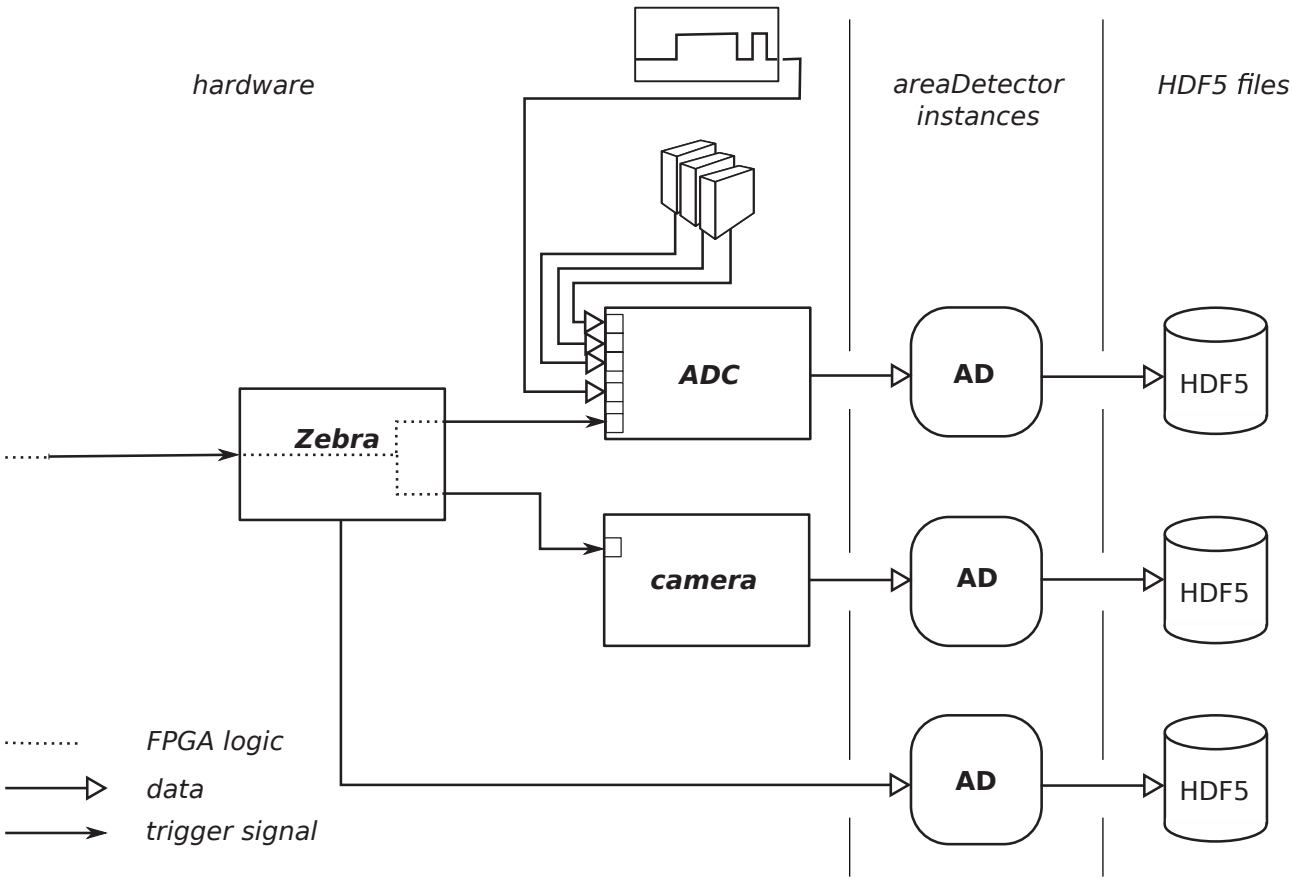

Figure 2: Zebra-driven data acquisition architecture.

later. We describe an EPICS-based way of doing that in our beamline-specific application in the next section.

The number of devices controlled in this way is only limited by the number of outgoing connectors on the Zebra box. So it is completely possible to trigger more than one type of detector or ADC card at once - or any other device that meets the requirements outlined in the previous paragraph.

Each of the devices involved has an instance of areaDetector associated with it. They ensure a consistent interface to EPICS and enable the architecture to write generic frames to, typically, HDF5 files. They also provide a well-defined set of parameters to interact with when configuring a specific scan or experimental run.

It is also worth pointing out that the entire design is is self-contained within the EPICS control system. So once the three pipelines are set up and configured, all a larger application needs to do is send start and stop commands to one end and pick up the data files at the other.

IMPLEMENTATION

Besides the areaDetector framework, the EPICS control system also provides drivers for a number of different cameras. These typically derive from areaDetector and use it in the conventional way, to acquire a series of image frames. Although an essential part of our data acquisition system, we did not need to extend any of them to make them work within it. So they will not be discussed further here.

The Zebra box and the ADC card, on the other hand, do not produce image frames but time series of samples across a set of channels. So the interpretation of a frame changes: it no longer represents an image in the traditional sense but a kind of ribbon of sampled values. Because we group several channels together, we continue to acquire two-dimensional data. As such, the areaDetector framework remains a suitable abstraction layer for this kind of measurement. The width of such a frame is fixed and corresponds to the number of available input channels. By contrast, the height has to be set by the user. As we explain below, approaches on what to set it to vary depending on the device.

The hardware triggering presented another, unrelated set of problems. While external trigger signals are a native feature of the Zebra box, our choice of ADC card does not really have this feature in a way that is useful to us. So both devices needed different kinds of drivers and driver extensions, which needed to be in line with the requirement that they both fit into the areaDetector framework. But on top of that, our ADC cards need to have the trigger signal fed in as data so one of the areaDetector plugins can extract it later. We will discuss each of those aspects in turn.

ISBN 978-3-95450-148-9

areaDetector Driver for Zebra

The original Zebra driver, developed at DLS, derives from class asynPortDriver and uses a serial port to communicate with the hardware. It also stores samples from ten different channels (encoder inputs, results of some of the logic blocks) in individual waveforms whenever a trigger is issued. So the time series for all of the channels are already available but there is no time stamp to go with them, and no native way of writing them to file.

Changing the inheritance chain of the Zebra to ADDriver turns it into an areaDetector. As ADDriver itself derives from asynPortDriver, no existing functionality is lost. Instead, the new parent gives the Zebra driver access to all of the areaDetector data structures. Figure 3 shows the details of the old and new parent class relations.

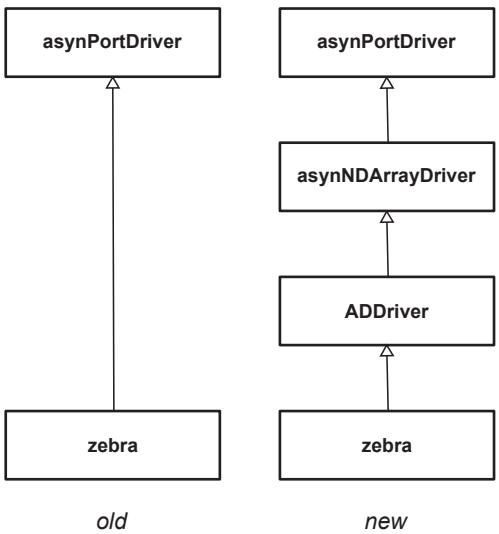

Figure 3: Old Vs. new inheritance tree for a Zebra driver in EPICS.

Every time the ten channels are sampled for the waveforms, we now store the same values as an NDArray in an areaDetector frame. A time stamp is added as the eleventh column. The Zebra's native interpretation of the Arm and Disarm commands as representing the beginning and end of a scan makes them equivalent to the start and stop points of an areaDetector acquisition. We extended the driver to treat them equally as a result. So while we acquire a stack of N frames from the camera during a scan of N points, we also get a stack of N one-line frames from the Zebra.

areaDetector Driver for D-tAcq ADC

DLS developed the original version of the areaDetector driver for the D-tAcq [4] ADC card. It connects to TCP port 4210 to read a continuous stream of data from the ADC, and to TCP port 4220 to send control commands to it. As the ADC samples its channels at a fixed rate, the driver's main task is to collect the samples reliably and store them as lines in a frame.

The sampling rate of the D-tAcq ADC far exceeds the frame rates we expect to see during our scans. Since we have no way of turning the data stream on and off, the driver's main focus is on capturing everything in sensible chunks for further combined processing in the downstream plugins. One of those chunks corresponds to a frame of raw data. This turns the user-defined frame height into more of a performance parameter: because everything runs in continuous mode, the length of a scan is defined in the file writer plugin instead.

The fact that there is no way to connect a native external trigger signal to the device was one of the most difficult conceptual challenges to overcome. In the end, we achieved it not within the device driver itself but by using a specialised areaDetector plugin. Using the trigger signal recorded on one of the data channels, it turns the massively over-sampled raw data into a series of one-line frames after the fact.

Using areaDetector's Reframing Plugin

The key to extracting the ADC samples from the raw data stream at the appropriate rate lies in the reframing plugin for areaDetector developed at DLS. It needs to be told which channel the triggers were recorded on in the raw data frame, and what the threshold value for a trigger pulse was for the application at hand. It then uses this information to downsample the incoming data by picking out those lines that have a trigger pulse in them. Each time a trigger is found, it reads out a user-defined number of lines - one in our case - into a new frame, creating a second data stream. It is this post-processed stream that is handed over to the downstream plugins. Figure 4 explains the exact relation between the two streams.

Every time a raw frame reaches the reframing plugin, there is a burst of re-sampled frames going to the downstream plugins, followed by a pause until the next raw frame arrives. So there are some practical considerations that need to be observed when setting up the core driver and plugins. As mentioned, the frame height in the raw data stream influences performance. During our tests, it became apparent that a size that corresponds to about one frame per second strikes the best balance between filling up the queue for the reframing plugin and overwhelming those of the downstream plugins during bursts. Even so, the queue sizes need to be far larger than for other areaDetector applications. A value around 1000 lets them cope with bursts every second or so with a comfortable margin.

The reframing plugin also expects a threshold voltage to be able to identify trigger pulses from the raw data frames. Since the pulse comes from a Zebra and we trigger on a rising edge, a value of 3V has turned out to be appropriate for this purpose in the early tests.

INTEGRATION

While it is perfectly possible to use the pipeline manually, its real power comes from controlling the endpoints using either Python scripting or a data acquisition package - GDA [5] in our case.

As long as the high-level application knows about PVs, it

ISBN 978-3-95450-148-9

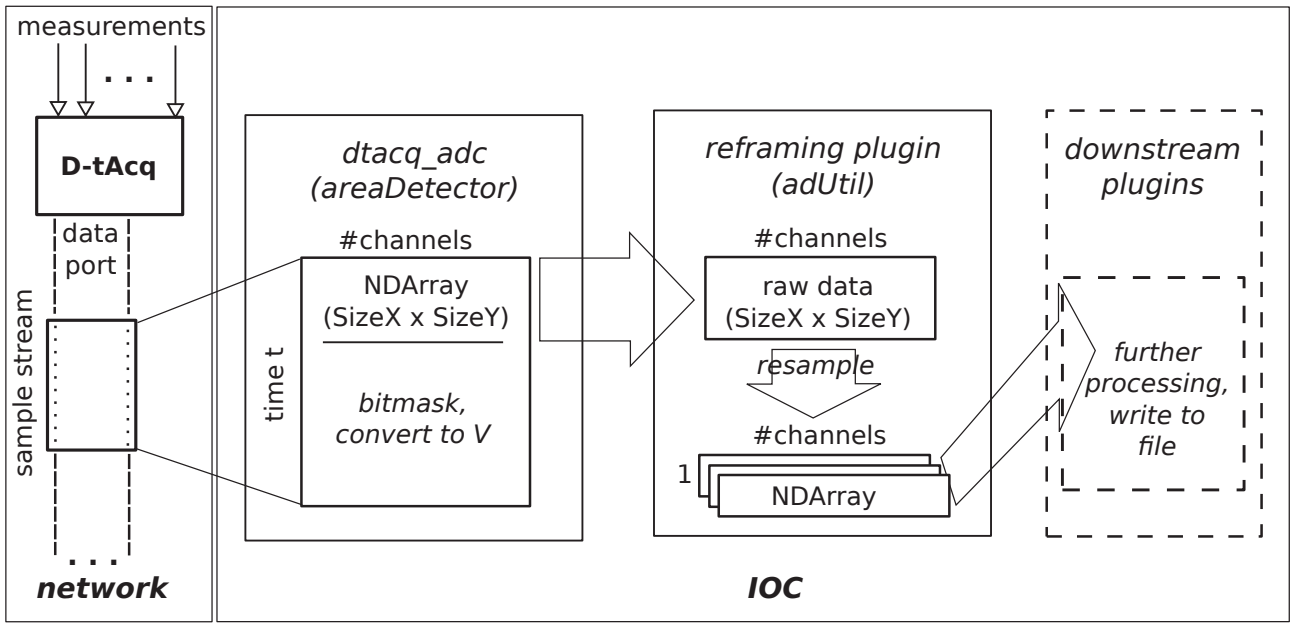

Figure 4: Data flow in the areaDetector plugin chain. Unusually, the raw data stream is discarded after the downsampling step in the reframing plugin and a stream of new frames of a different size is passed on to the downstream plugins.

can configure and monitor the lower-level devices using caput, caget and camonitor. It can also aggregate the results into NeXus files that provide a single entry point into the collected data from the three pipelines. Because it knows about the rest of the experimental setup, it can start and stop acquisitions and file captures at the appropriate times.

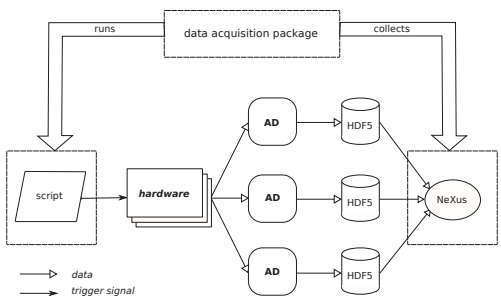

Figure 5: Role of high-level data acquisition package.

Figure 5 shows how GDA or a similar application brackets the lower-level components during a scan, acting in a kind of management role. This approach turns the entire integrated data acquisition system into a single building block within the beamline infrastructure.

CONCLUSION

Using hardware-based trigger signals in combination with the Zebra box's programmable FPGA logic ensures exact timing and synchronised acquisition of data and meta data at higher sampling rates. EPICS drivers derived from areaDetector provide a well-defined interface to configure the system as a whole, all the way down to its individual components. Structuring the controls and data flows in three parallel pipelines allows us to integrate such a system into the rest of the beamline software infrastructure in a modular way. While there are some performance parameters to keep in mind when interacting with the 'pipeline-within-a-pipeline' chain of areaDetector plugins, using them is essential and ultimately adds to the overall flexibility when choosing individual hardware components. Fine-tuning and testing of the complete system is ongoing and we expect to deploy it to one of our high-throughput tomography beamlines at some point next year.

ACKNOWLEDGEMENT

The authors would like to acknowledge that the original Zebra driver was developed by Tom Cobb, DLS, the D-tAcq driver by Adam Bark, DLS, and the areaDetector reframing plugin by Edmund Warrick, DLS.

REFERENCES

[1] T.M. Cobb, Y.S. Chernousko, I.S. Uzun, "ZEBRA: A Flexible Solution for Controlling Scanning Experiments", Proceedings of ICALEPCS 2013.

[2] EPICS collaboration website: `http://www.aps.anl.gov/epics/`

[3] areaDetector documentation: `http://cars9.uchicago.edu/software/epics/areaDetector.html`

[4] D-tAcq website: `http://d-tacq.com`

[5] GDA website: `http://www.opengda.org`

ISBN 978-3-95450-148-9