PIONEER DAQ Development

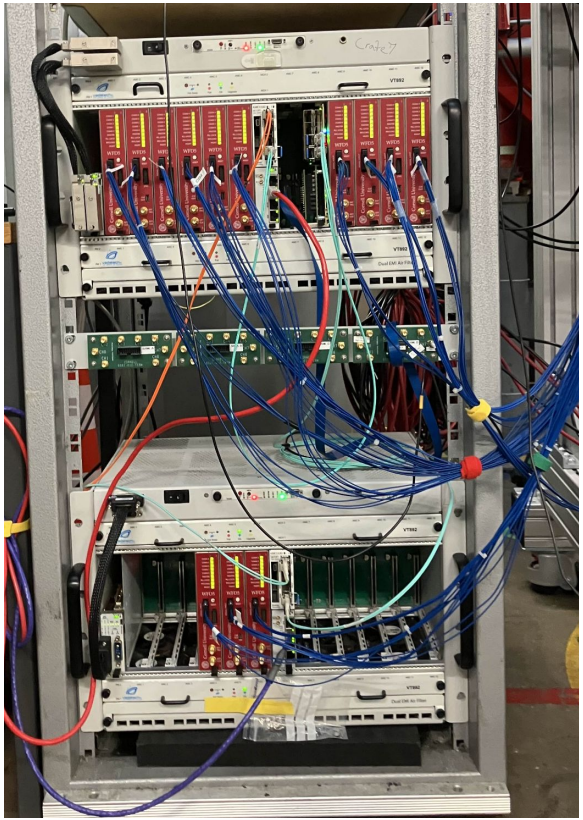
Jack Carlton
University of Kentucky
October 16th, 2025

g-2 DAQ modified - Usage for LYSO testbeam

- Iteration on previous LYSO test beam

Run	Crates	Channels Used	Event Rate (Hz)	Data Rate (MB/s)
2023 PSI LYSO Test Beam	1	~50	~400	~30
2025 PSI LYSO Test Beam	2	~60	~2500	~200

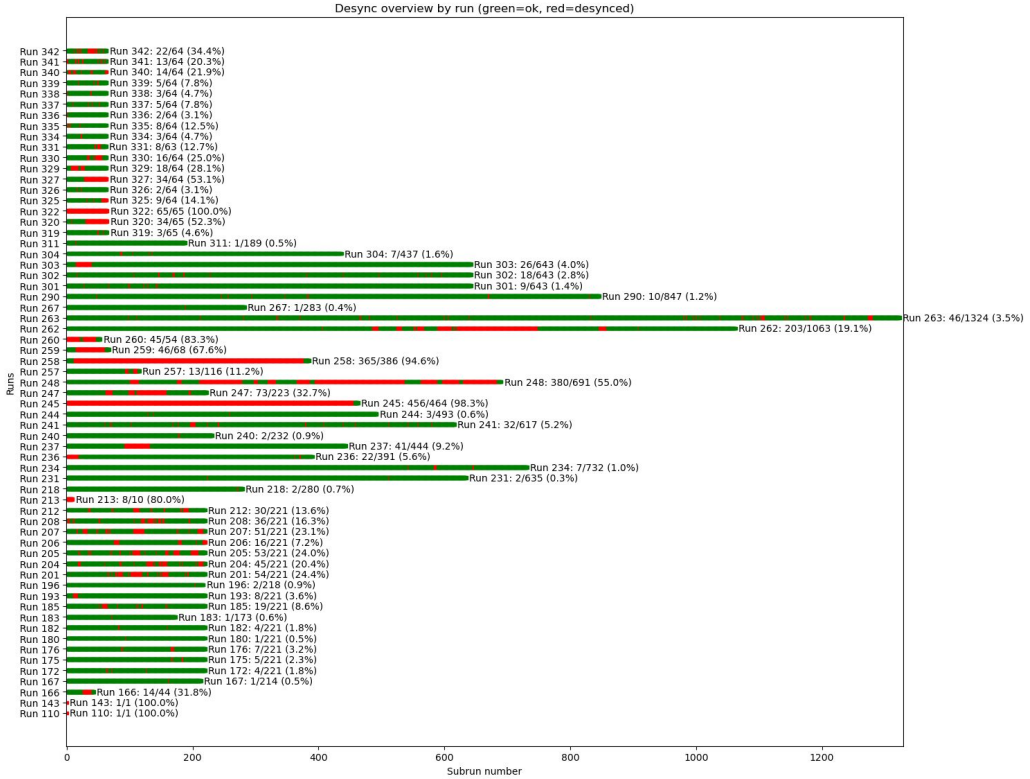
- Rate limited by HDD write speed ~250 MB/s
  - For small scale experiments, could be solved by writing directly to SSDs (>1GB/s write speed)
    - Limited by SSD space (≤8TB for consumer electronics)
- Desyncs between crate data caused (mostly) recoverable issues
  - ~2% of events across all data runs are desynced



Two crate setup used at PSI LYSO Testbeam

g-2 DAQ modified - Desync Issues

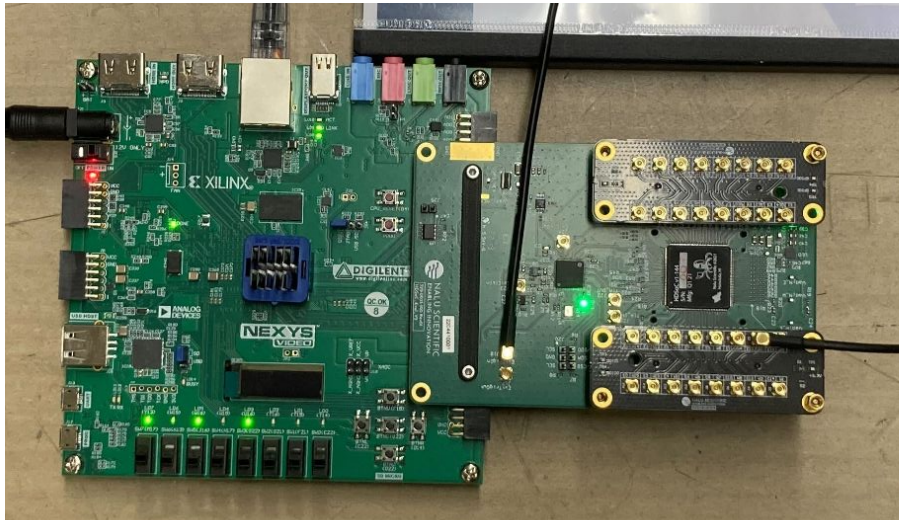
- Desync issues should be fixed
  - Replicated the issue on the UKy teststand
    - Caused by internal ring buffer overflows (“GPU_BUFFER”)
  - Fixed software bug preventing trigger throttling from working properly
    - Trigger throttles apply back pressure to prevent ring buffer overflows
- For the testbeam we used a “band-aid” solution
  - Event builder stops run on desync detected
  - Run is restarted immediately
  - Gives the crates time to recover
  - Downsides: splits data runs, likely not scalable for many crates



All desynced subruns in 2025 PSI LYSO Testbeam

HDSoC DAQ - Status

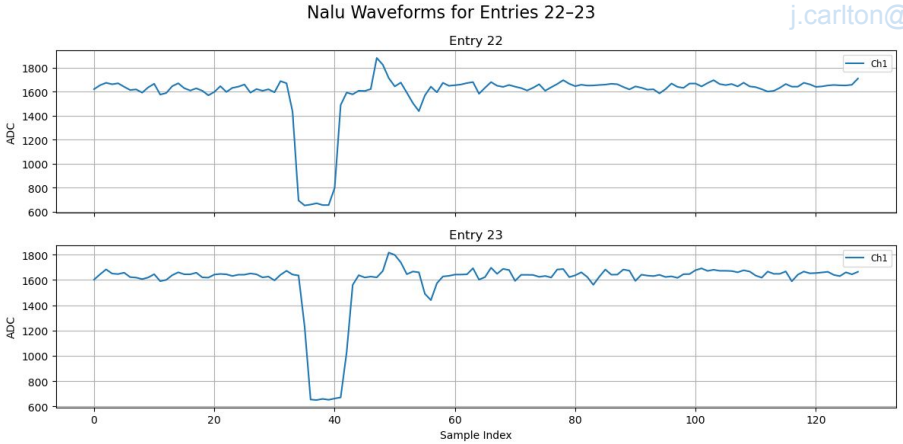
- HDSoC is (was?) a candidate ATAR digitization system
- Midas frontend working
  - Supports internal and external trigger settings
  - Handles HDSoCv1 rev2 max data rate of ~55 MB/s
- Documentation (manual) available
  - Details setup, configuration, etc.
- Unpacker app available
  - Unpacks midas data into ROOT tree, stores in analyzable .root files



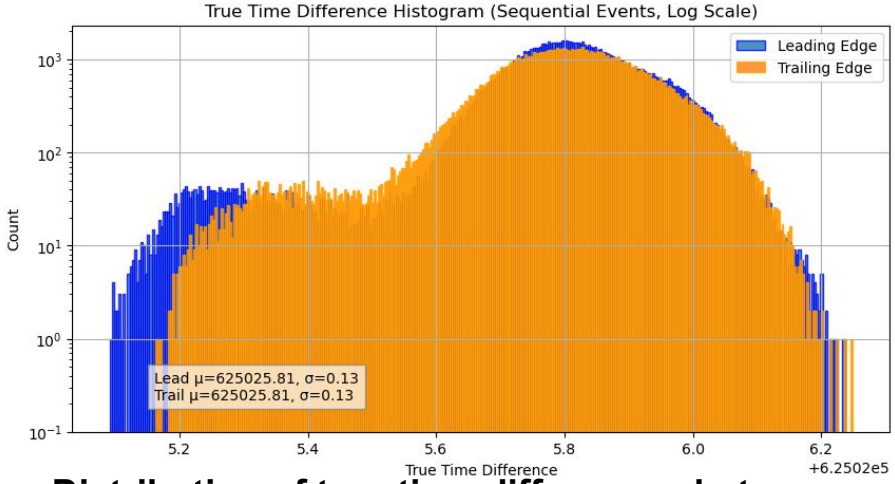
Nalu's HDSoC FMC attached to a Nexys A7 Video Card

HDSoC DAQ - Next Steps

- Low priority for the time being
  - Not actively working on development
  - Development focus shifted to to SAMPic system
- Undergraduate Brennan Edwards doing studies on HDSoC digitized data
  - Timing resolution studies
    - Constructs “pseudo time” of leading and trailing edge
    - Then constructs true time differences between events
    - Similar to g-2 analysis



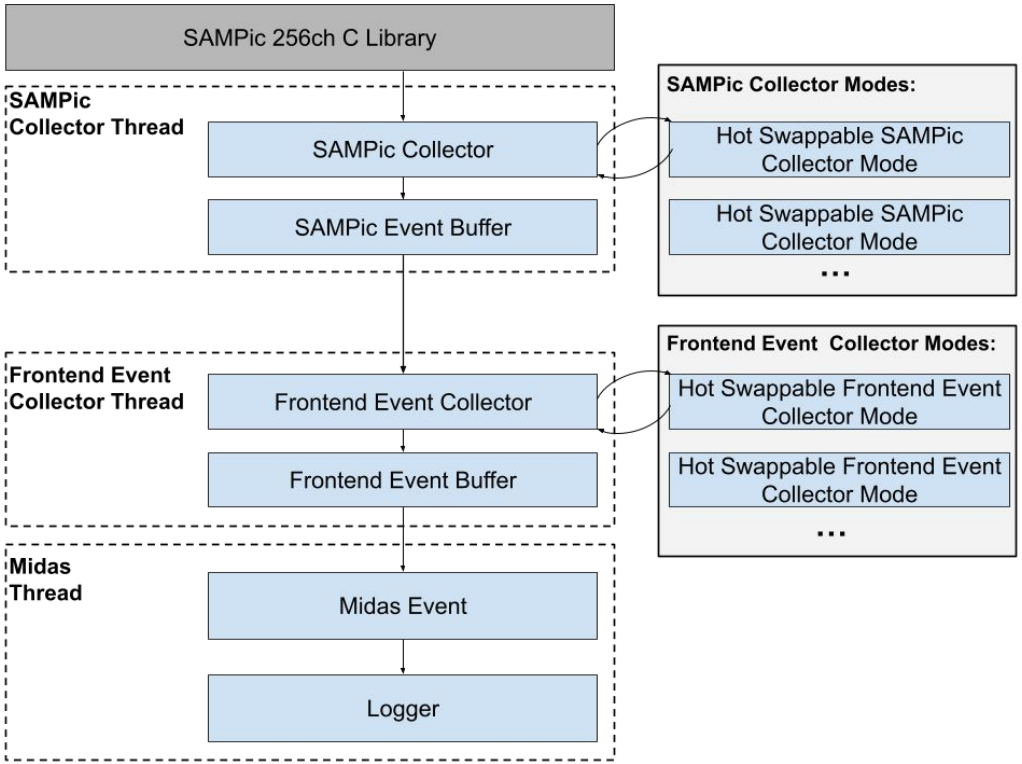
Two consecutive events used to create time difference



Distribution of true time differences between consecutive events [preliminary analysis]

SAMPic DAQ - Status

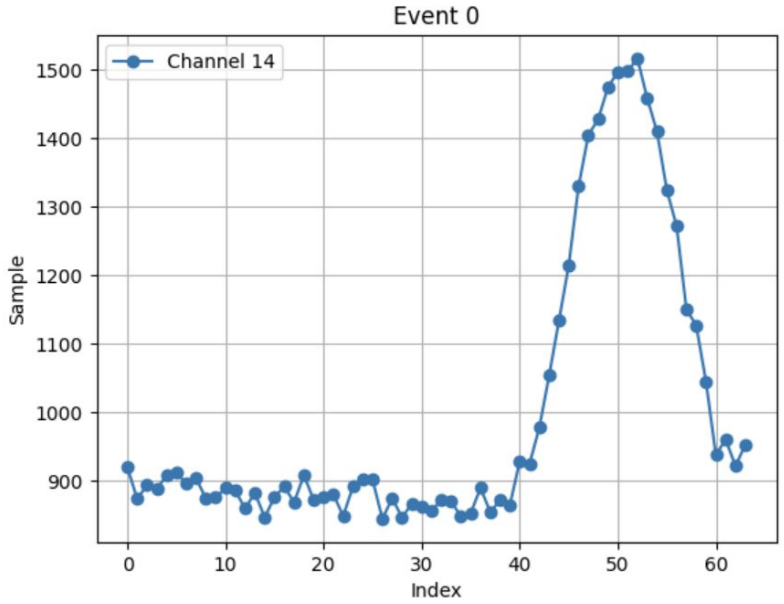
- Working midas integration
  - Converts ODB params → Sampic settings for most settings
  - 1 thread to collect “SAMPic Events”
  - 1 thread to form “SAMPic Events” into “Frontend Events”
    - Frontend events may span multiple sections of “SAMPic Events”
  - 1 thread to handle midas state machine
- Event formation logic is hot swappable for ease of development



Data flow diagram for SAMPic Data

SAMPic DAQ - Next Steps

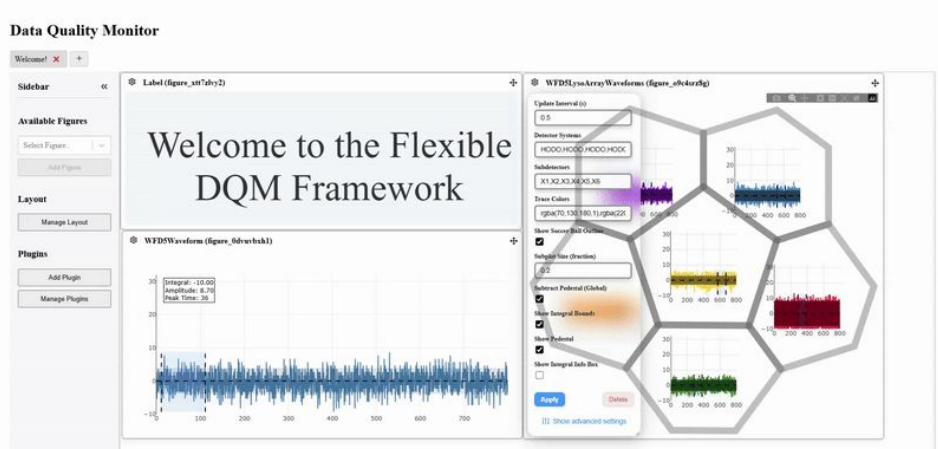
- Rates tests
  - Only tested using simple cases so far (~KB/s rates)
  - Need to test more realistic rates (~10kHz/~100 MB/s)
- Implement more realistic event formation
  - Trigger architecture details needed to finalize event formation design
- Write an unpacker to analyze/debug midas data
  - Can adapt Midas file unpacker app to do this job



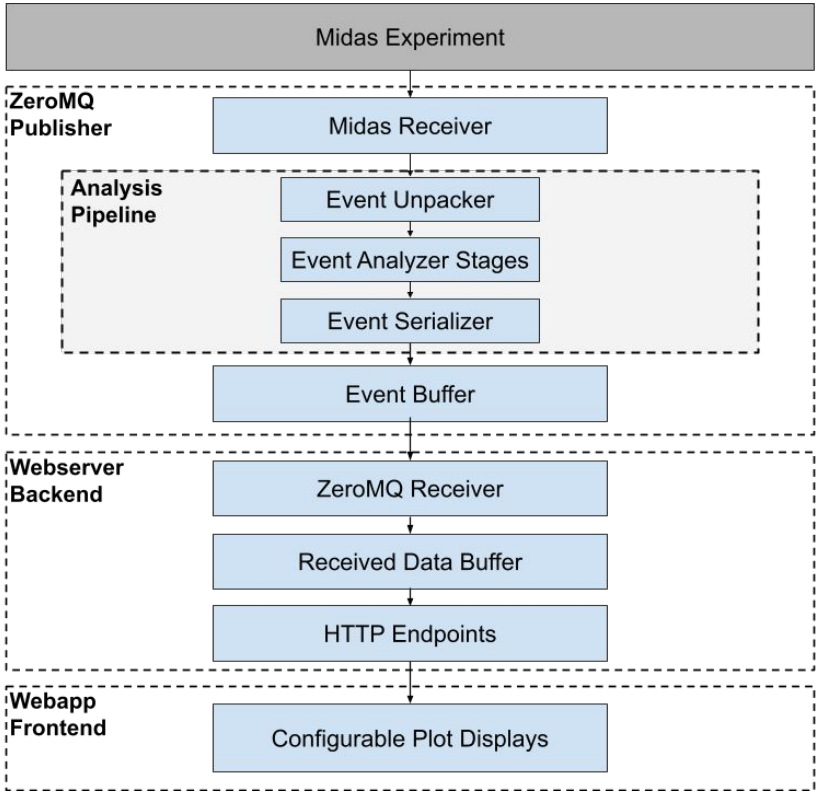
Example Digitized trace from SAMPic System

Supplementary Software - Flexible DQM

- Used in PSI 2025 LYSO Testbeam
  - Worked okay, needed some live bugfixes
  - Configurable to any experiment (with some boilerplate additions)
- Manual, wikis, tutorials, and repos available



Example DQM webapp page view



Data flow diagram for DQM

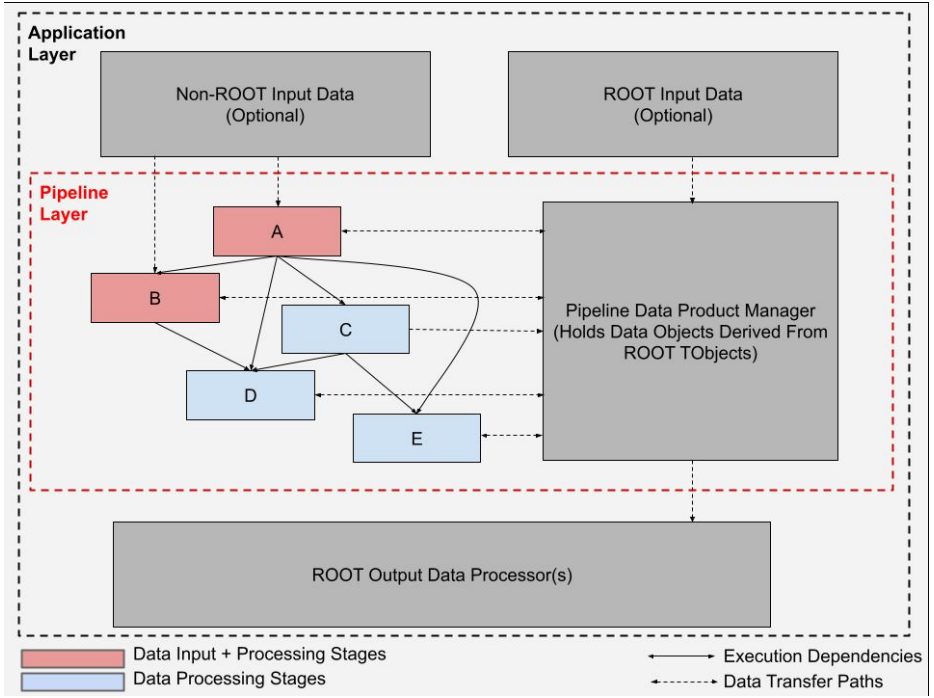
Supplementary Software - Unpacker Application

- Application to convert midas files to ROOT Trees
  - Work in progress
  - Currently on works for HDSoC midas data
  - Plans to add SAMPic midas data support
  - Adaptable to event structure changes
  - Current does not support g-2 modified DAQ midas data, can use sean's unpacker
- Adding new unpackers simplified
  - Boils down to adding new plugins in the analysis pipeline framework

Example HDSoC data ROOT tree formed by the application

Supplementary Software - Analysis Pipeline Framework

- Lightweight, configurable pipeline framework for ROOT data
  - Inspired by Gaudi, but simpler and faster to deploy
  - Built on ROOT, leveraging:
    - Plugin system for modularity
    - Reflection for flexible configuration
    - Custom data products
- “Backbone” of apps:
  - ZMQ publisher (part of DQM framework)
  - Midas file unpacker app
- Documentation available
  - Wiki
  - Repo

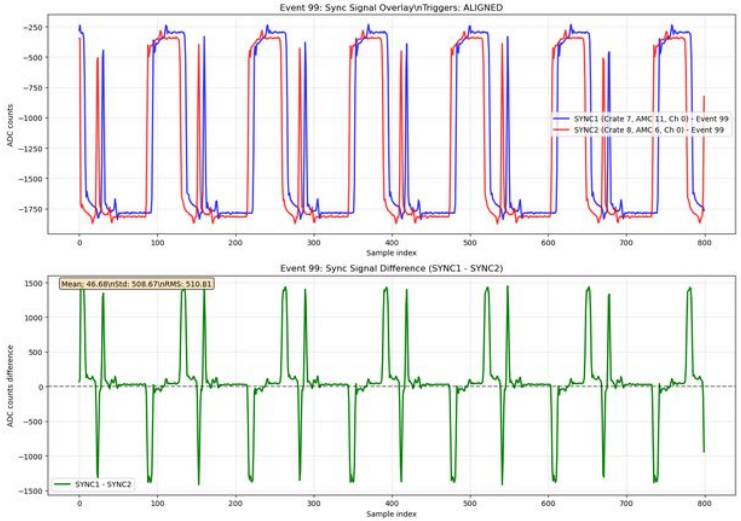


Example data flow diagram for an analysis pipeline

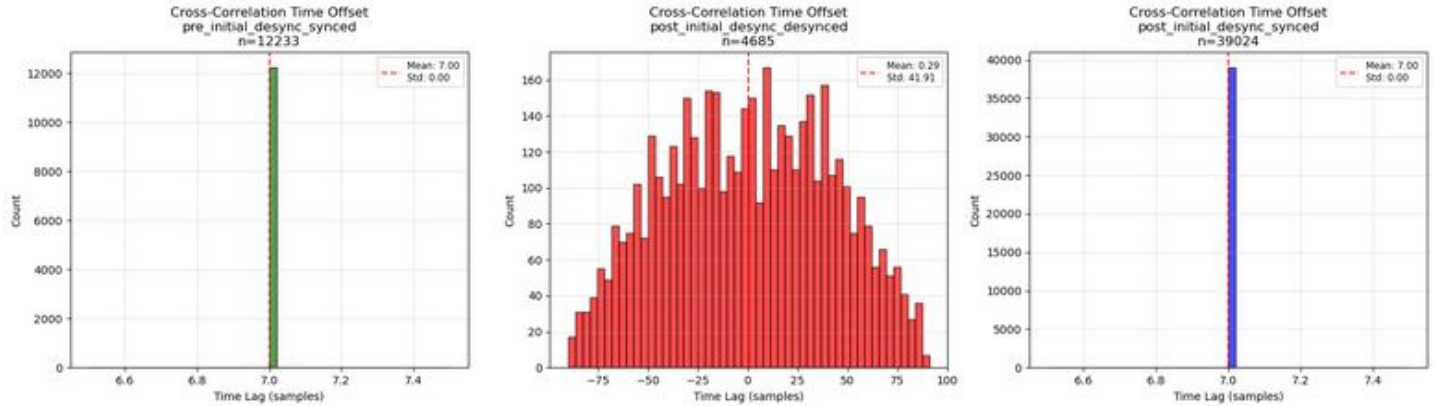
Auxiliary Slides

Study on desynced data

- Elog available
- Clock signal fanned out out into both crates
- Showed the DAQ “recovered” from desyncs properly



Clock signal fanned out in both crates for synced event

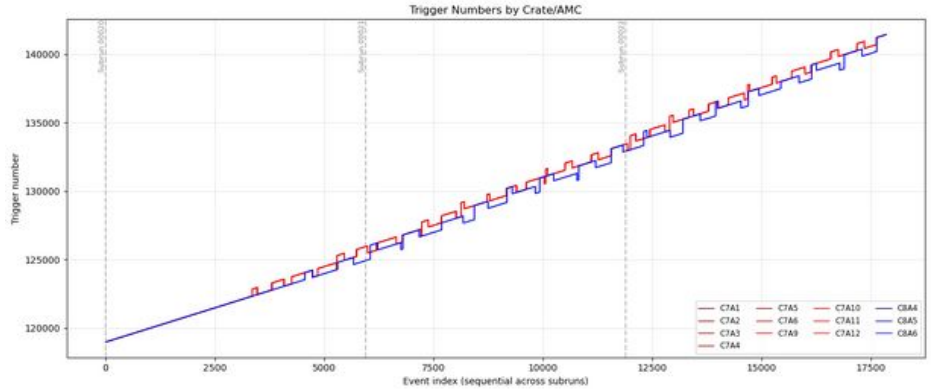


Cross correlated of time offset between both crate clock signals

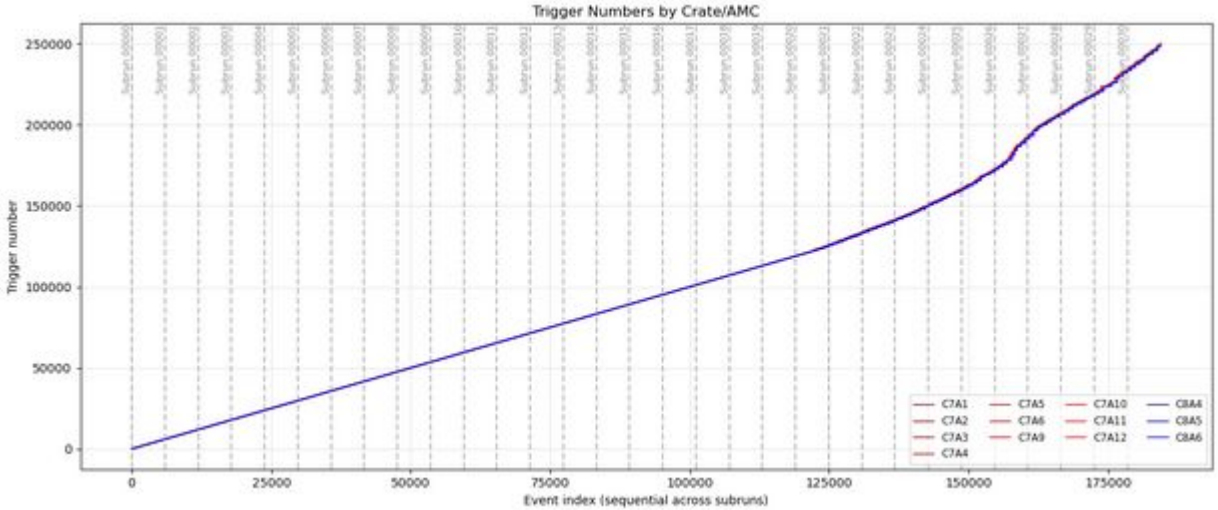
More on desyncs

- Desyncs occurred in multiples of the GPU buffer size (512)

- Saw periods of “instability” and periods of recovery



Trigger index for each module across one subrun



Trigger index for each module across subruns

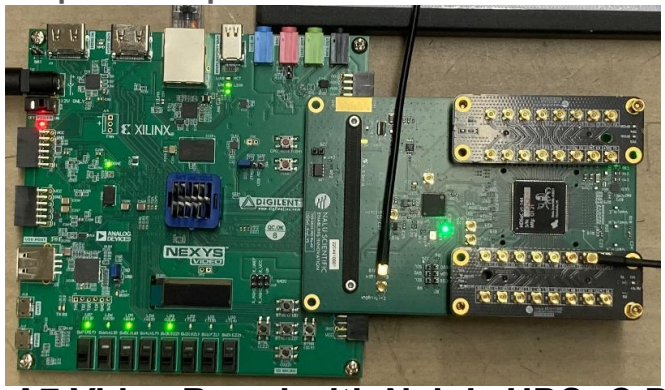
g-2 DAQ modified - Next steps

- Polish the code for trigger throttling before pushing
- Find a solution to HDD bottleneck
  - No bottleneck for:
    - Low rate (<~250MB/s) applications
    - Low storage (<~2TB) applications
  - Potential solution if usage case is under ~16 TB
    - Buy large SSDs (ex. 8TB ~ 500$)
  - Need to revisit g-2 solution for larger applications

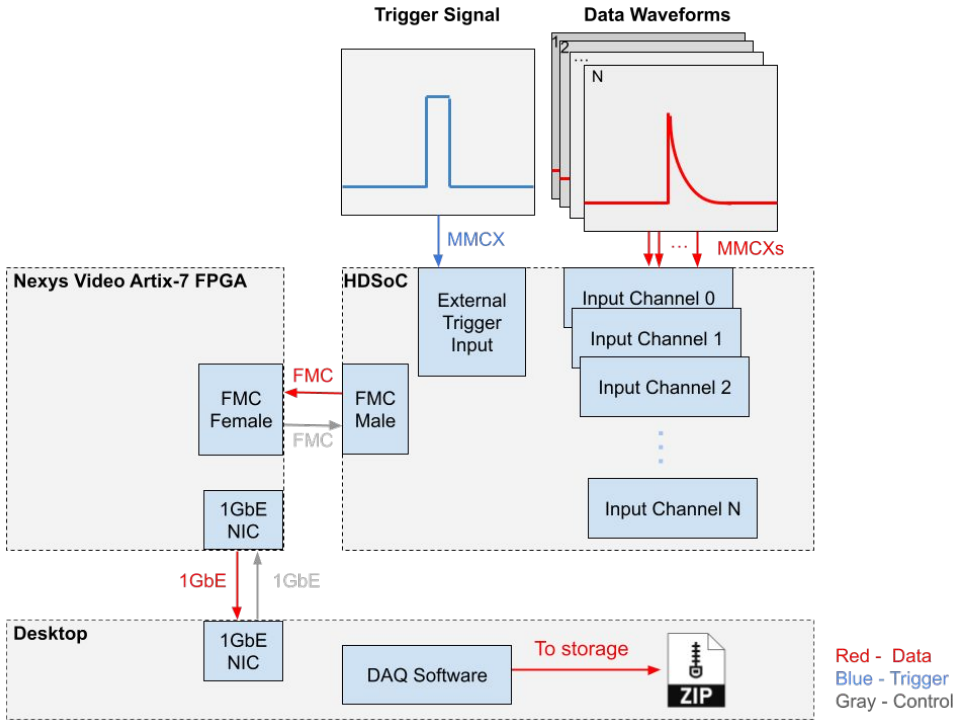
HDSoC DAQ - Hardware

- DAQ is functional and integrated into MIDAS
- Can digitize data rates up to 55 MB/s, event rates up to 30 kHz*

*For specific parameters



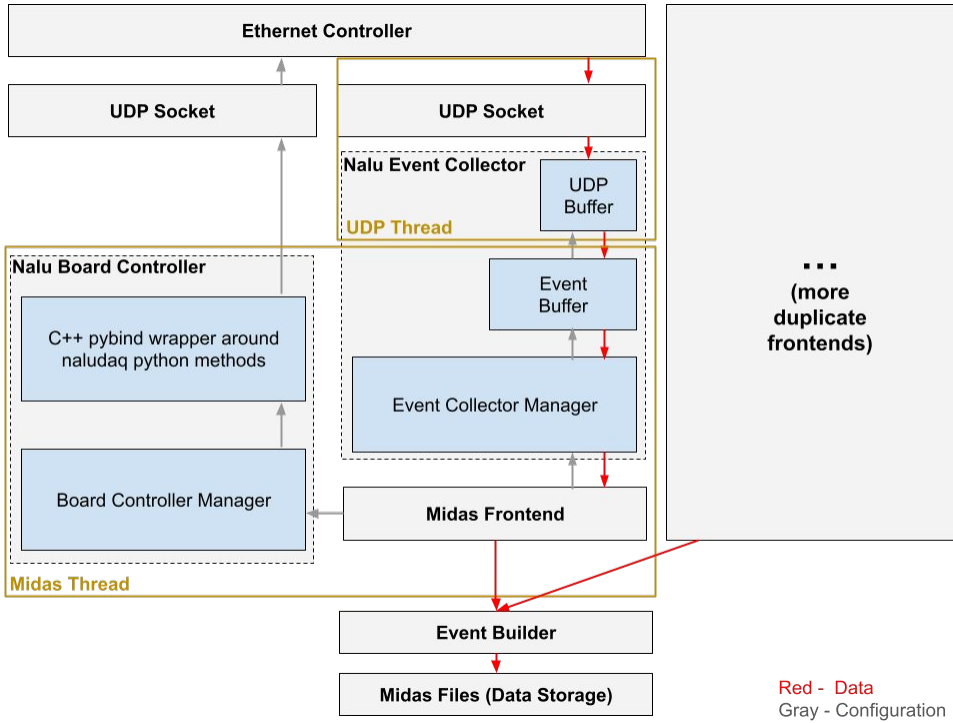
Nexys A7 Video Board with Nalu's HDSoC Digitizer Attached as an FMC Module



Conceptual Hardware Diagram for the HDSoC Readout

HDSoC DAQ - Software

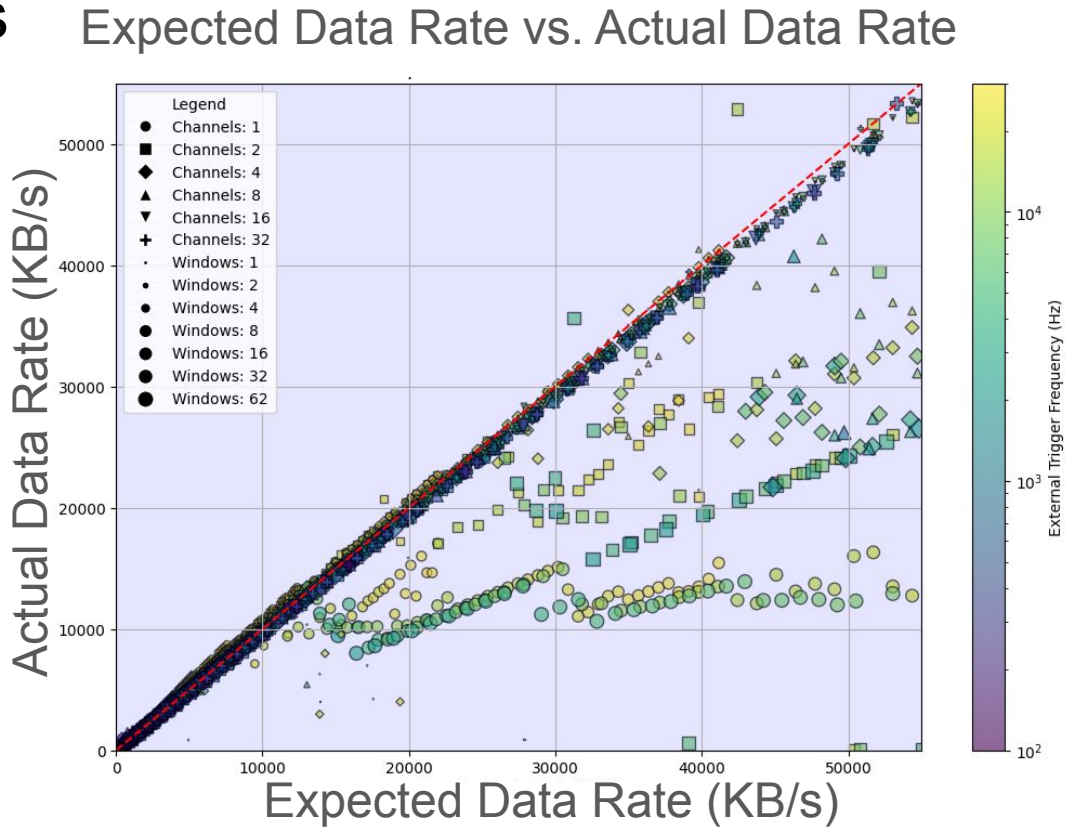
- Wrote a midas frontend that leverages custom libraries created for readout
  - Nalu Board Controller
  - Nalu Event Collector
- Separate branch for rate testing, leveraging custom RP Pico W libraries created for automatic rate testing
  - RP Pico W remote controller
  - RP Pico W board interface



Conceptual Software Diagram for the HDSoC Readout

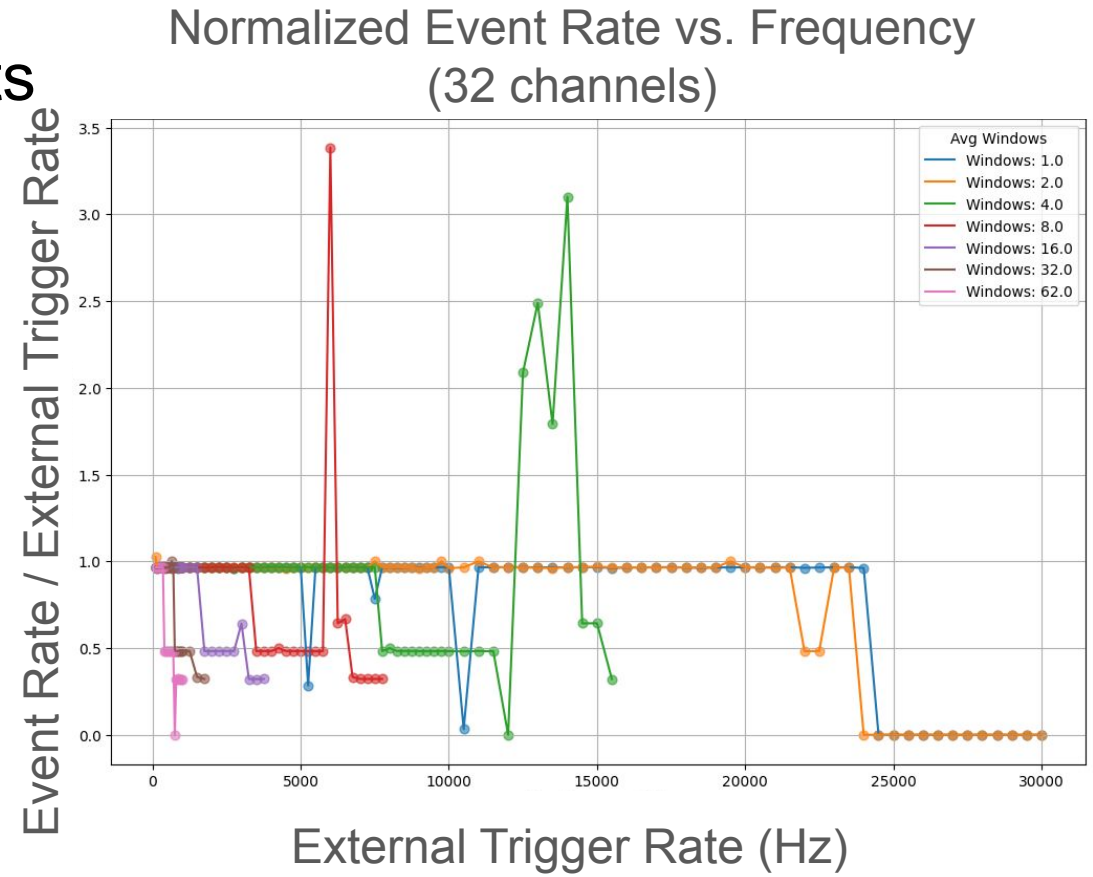
HDSoC DAQ - Rate Tests

- Majority of input parameters → performance as expected
- Outliers where we underperform
  - Expect good performance under 55 MB/s
  - Looking for cause of performance drops



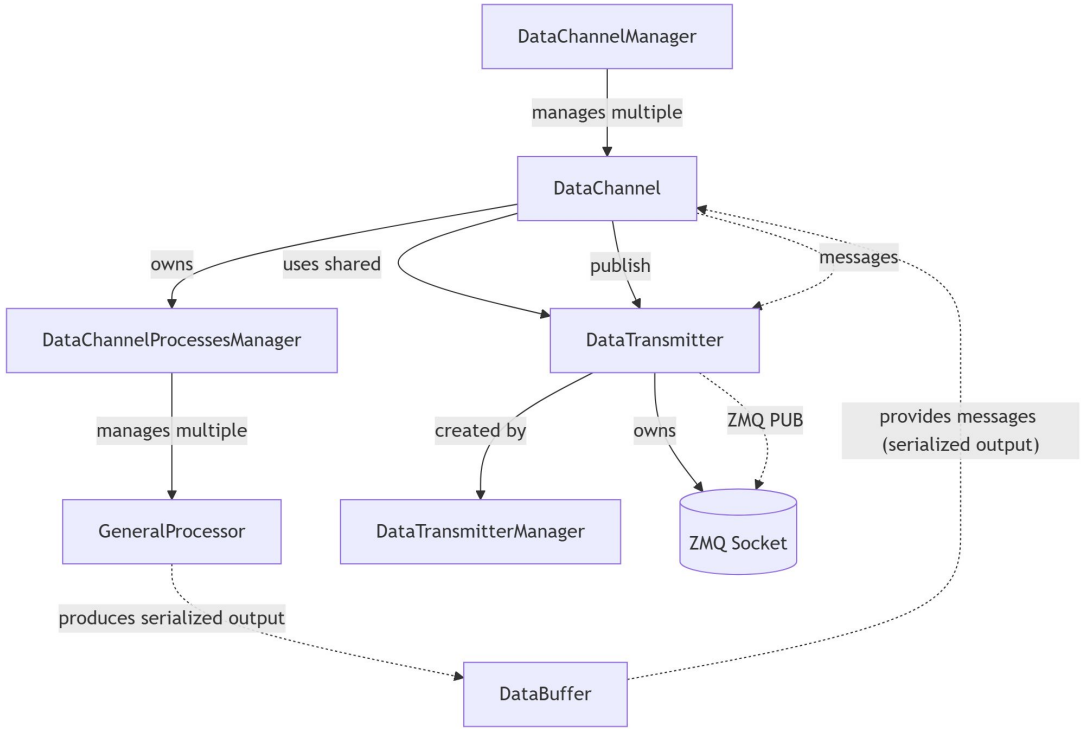
HDSoC DAQ - Rate Tests

- For 32 channels (all active)
- 1 window = 32 12-bit ADC samples
- 1 Gsps
- Can take 32 traces length 64 ns at rates ~20kHz reliably
- Events begin dropping near 55 MB/s threshold



ZeroMQ Publisher Application

- Configurable software to publish data over ZeroMQ
- Main branched configured to use midas
  - Not technically necessary, data source can be anything
  - Uses my midas receiver library
- Uses C++ Package Manager (CPM) to clone (most) dependencies from gitub and build them
  - Less work setting up environments for user



Midas Receiver Library

- Library gives methods to launch a thread that listens for midas events in an embedded, configurable way.
- Used by ZeroMQ receiver to receive events
- Settings for how to receive data
- Draw back:
  - Singleton, cannot have two receivers in one application (midas doesn’t like this)

Field	Meaning / Function
host	Hostname or IP of the MIDAS experiment server; empty string ("") connects to the local host.
experiment	Name of the MIDAS experiment; empty string uses the default experiment for the host.
bufferName	Name of the shared memory event buffer (e.g., "SYSTEM", "BUF001"); determines which event stream is read.
clientName	Logical name for this receiver client; appears in the MIDAS client list (/mhttpd).
eventID	Event ID filter; EVENTID_ALL subscribes to all event types in the buffer.
getAllEvents	True receives all events blocking, false is nonblocking
maxBufferSize	Maximum number of events in circular buffer managed by receiver thread
cmYieldTimeout	Yield interval (ms) for midas communication manager. Smaller means it “talks” to midas more
transitionRegistrations	List of (Transition, Priority) pairs registering interest in specific MIDAS state transitions. Receiver updates run # and errors on transitions.
TR_START	Run start transition; fired when a data-taking run begins.
TR_STOP	Run stop transition; fired when a run is stopped normally.
TR_PAUSE	Run pause transition; data taking temporarily halted without ending run.
TR_RESUME	Run resume transition; resumes data-taking after a pause.
TR_STARTABORT	Triggered on aborted start or initialization failure