Hall B Magnet FastDAQ

Brian Eng
Detector Support Group

August 27, 2018

Overview

- Issues with current configuration
- What existing deployed looks like and how it performs
- Current development code and performance
- What worked and didn't
- Conclusion

Timestamp Jitter

- Data is read from cRIO ADC modules and written to 2000-element EPICS arrays (which automatically add a timestamp) every 200 ms
  - 2000-element array is due to bug in NI EPICS Server not being able to set a larger array size than default
- Calculating time delta between sample timestamps
  - < 100 ms considered duplicate
  - > 300 ms considered a miss
- Ideally this should be exactly 200 ms

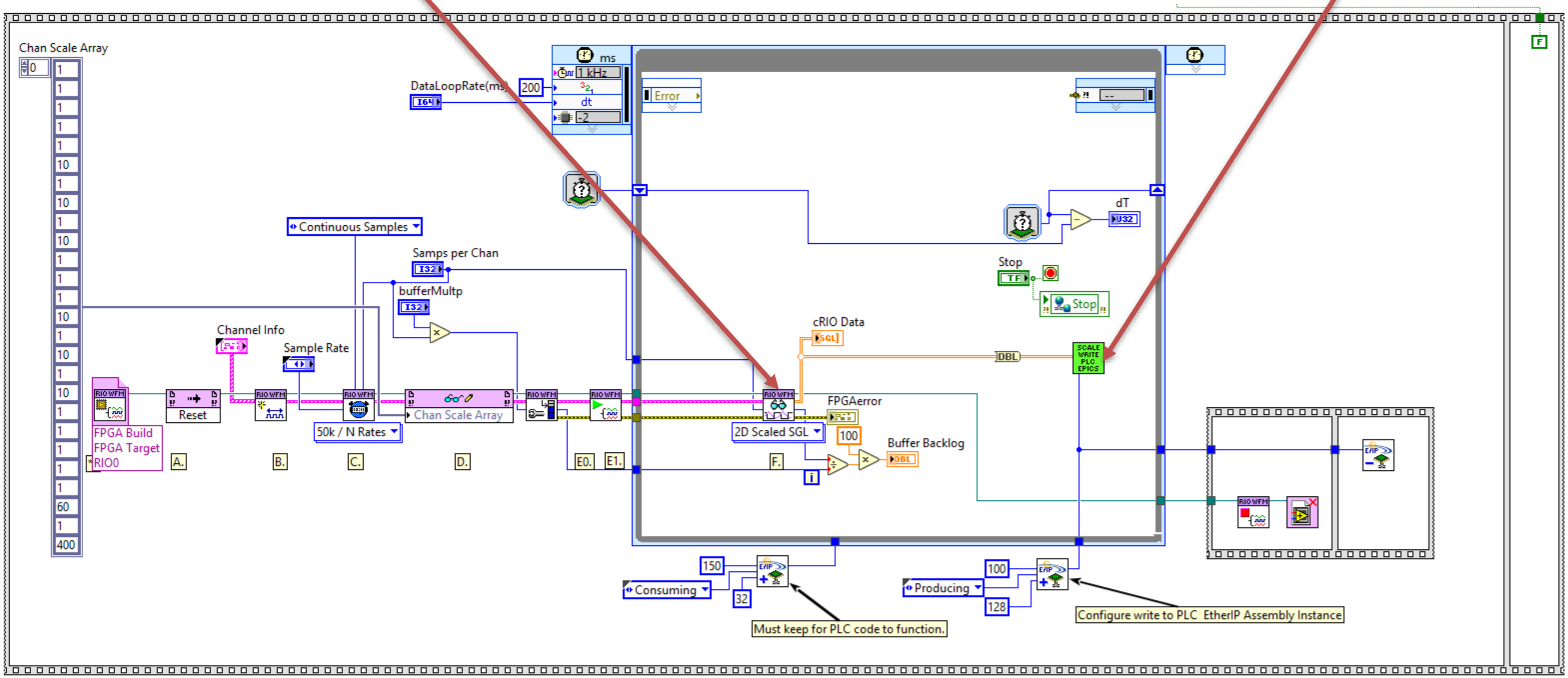
Deployed Code

- Single timed while loop with sequence
  - Each sequence runs at 100 ms
- All calculations (min, avg, max) are done in loop
- Not all ADC channels are used
  - Array manipulation to remove unused channels
- Writing to EPICS and PLC done in loop

Deployed Code (Block Diagram)

ADC Read

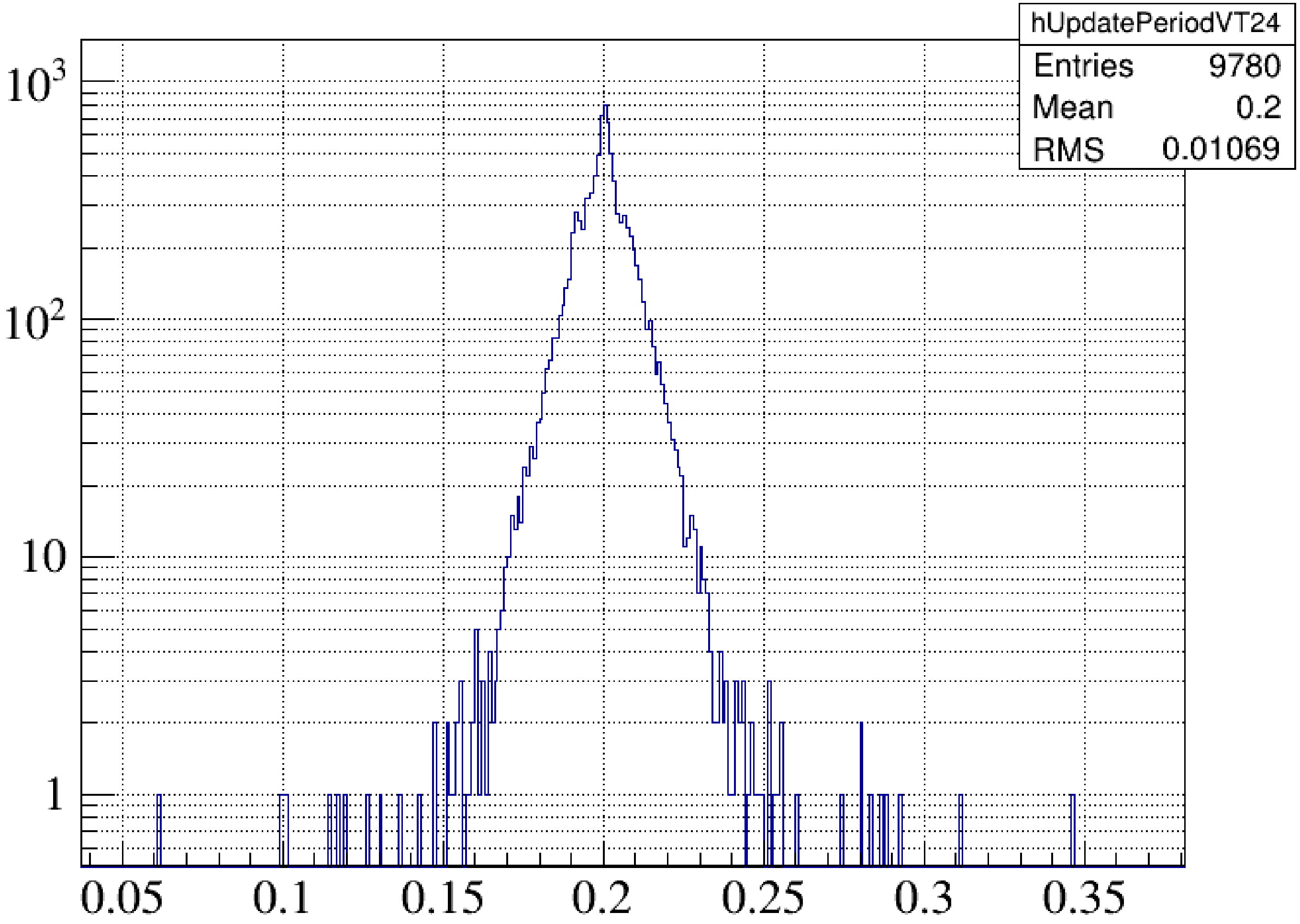
Array manipulations
EPICS & PLC Write



Must keep for PLC code to function.

Configure write to PLC EtherIP Assembly Instance

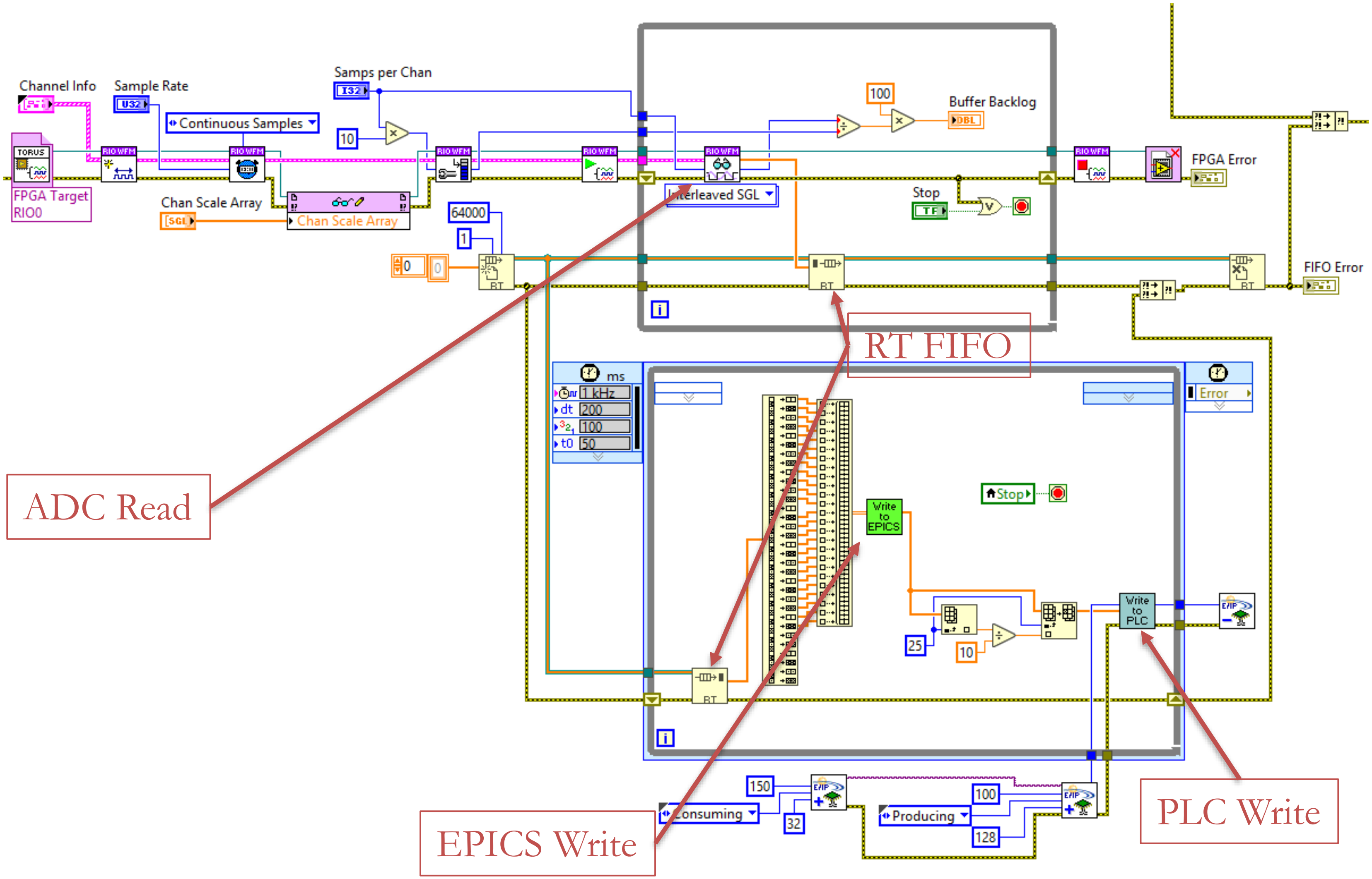
Deployed Code (Jitter)



Development Code (as of Today…)

- Separate loops for reading ADC and all other functionality
  - Use RT FIFO to pass data between loops
- Update cRIO Waveform Reference Library
  - Can return data as interleaved 1D array (no array manipulations done)
- Array manipulations and calculations done in separate loop
- Writing to EPICS and PLC are separate VI calls
- Using 32-bit floating point instead of 64-bit

Development Code (Block Diagram)



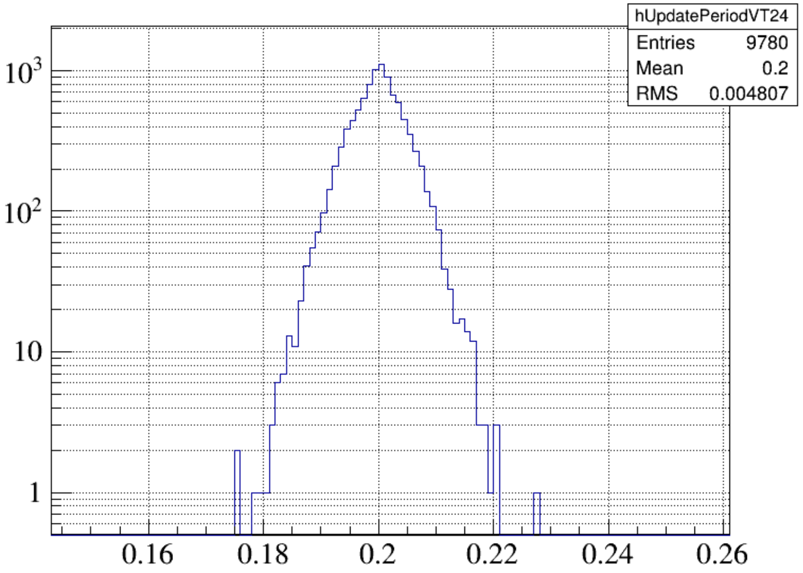
ADC Read

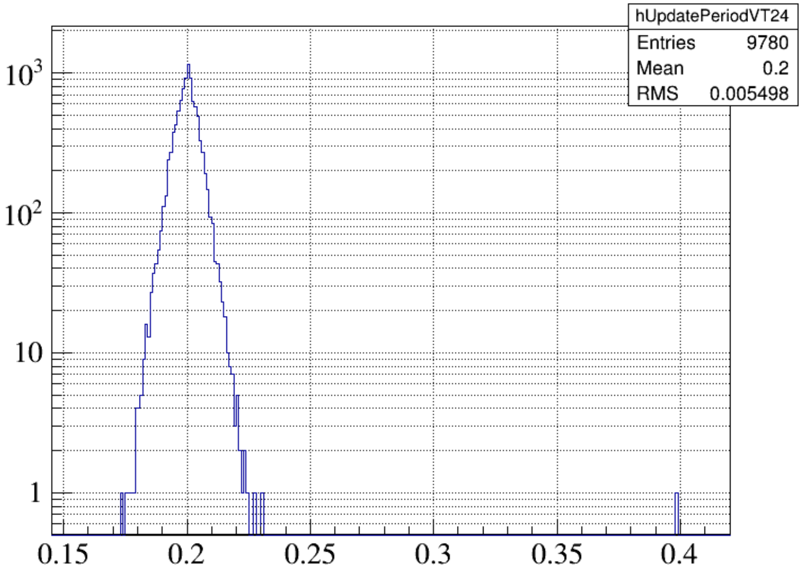
RT FIFO

EPICS Write

PLC Write

Development Code (Jitter)





The Good, Bad, and Ugly

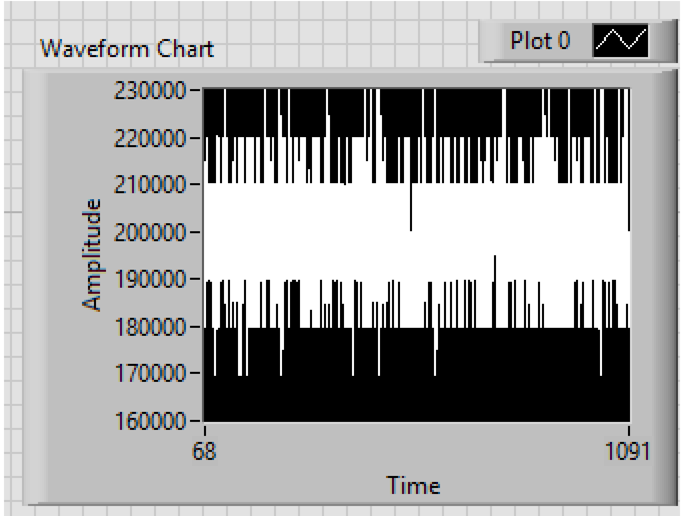
- Moving to 32-bit floating point was big improvement
  - Since data from ADC modules is 32-bit no loss of precision
- Separating out functionality allowed easier measurement of timing of individual functionality
- DAQmx API is much simpler and faster to deploy than FPGA, but worse timing (~13 ms vs ~5 ms RMS).
- Adding timeouts to some functions (e.g. RT FIFO Read) completely killed any jitter gains, 2 peaks at 200 ms and 400 ms

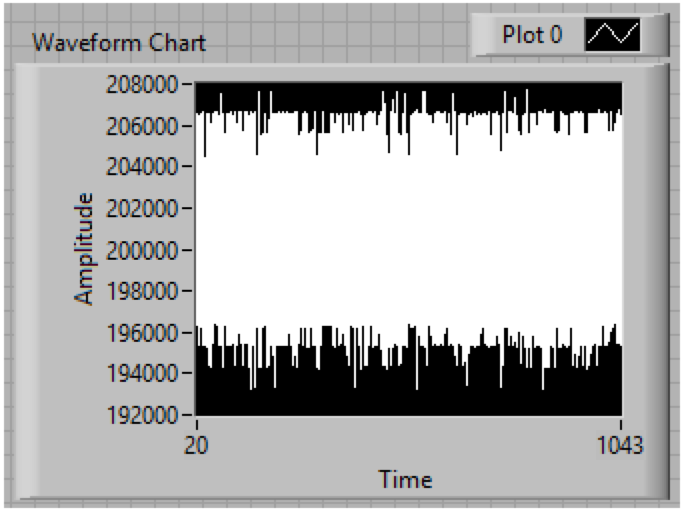
Conclusions

- New code has significantly less jitter, but at expense of completely missing a sample
  - Still much less frequent than current code
    - ✓ Most files have 0 timestamp issues
  - Current code has timing issues on nearly every single file generated (2 GB ROOT files, ~30 min)
- Still a work in progress
- A slower ADC read rate would eliminate all timestamp issues independent of code

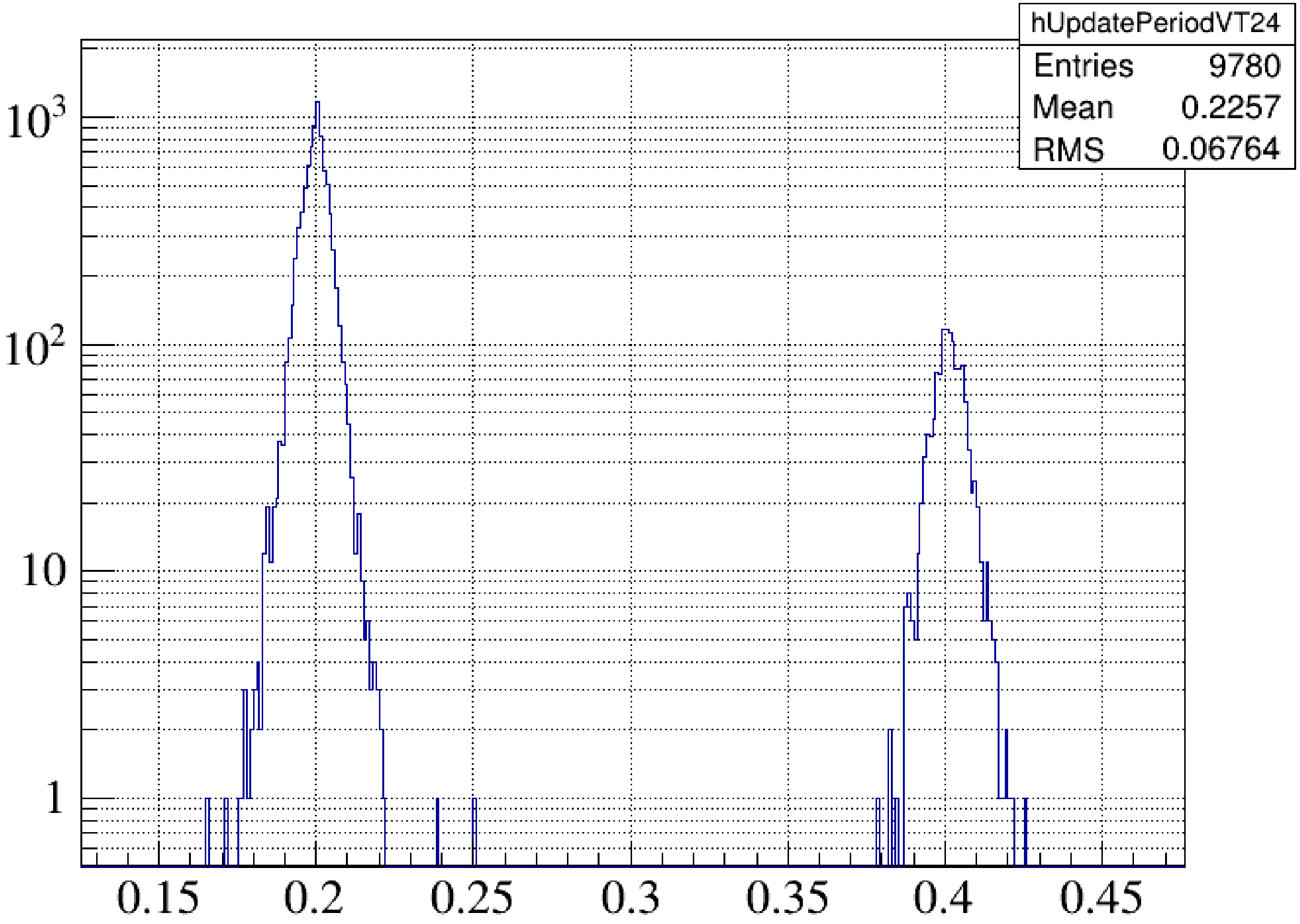
BACKUPS

DAQmx (left) vs FPGA (right)





RT FIFO Read with Timeout


hUpdatePeriodVT24
Entries 9780
Mean 0.2257
RMS 0.06764
10^3
10^2