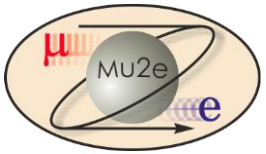
Fermilab

# Development of a Portable Mu2e TDAQ System




Ryan Rivera – Mu2e Trigger & DAQ Level 2 Manager

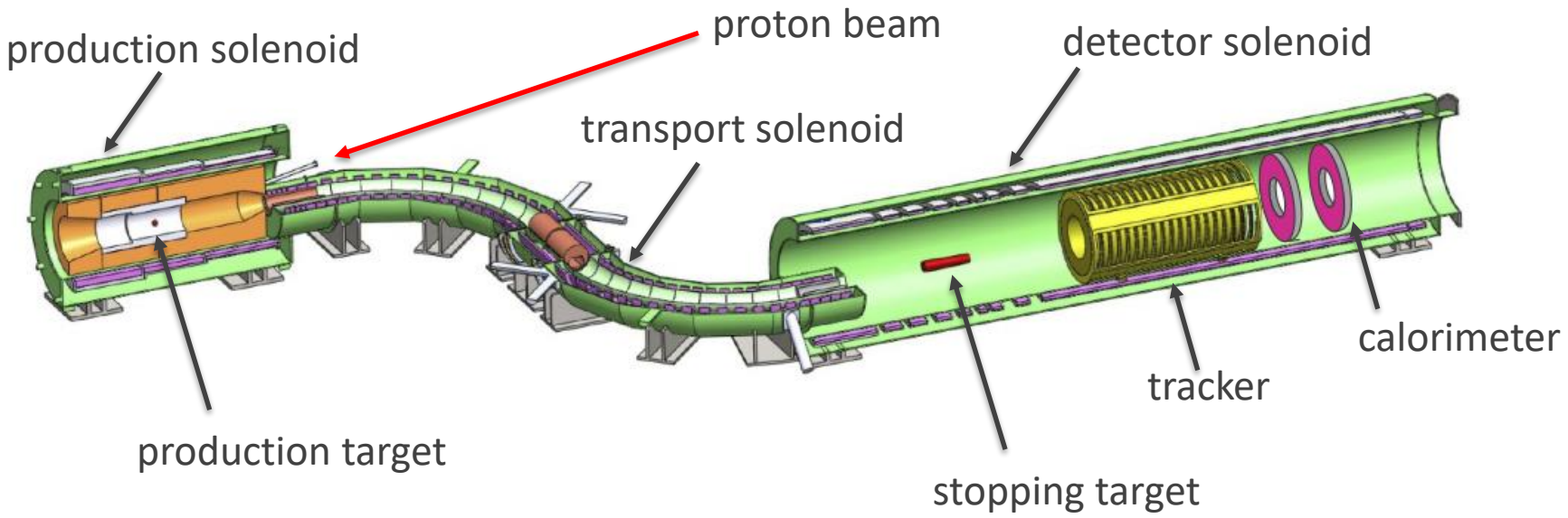
August 04, 2021

# Outline

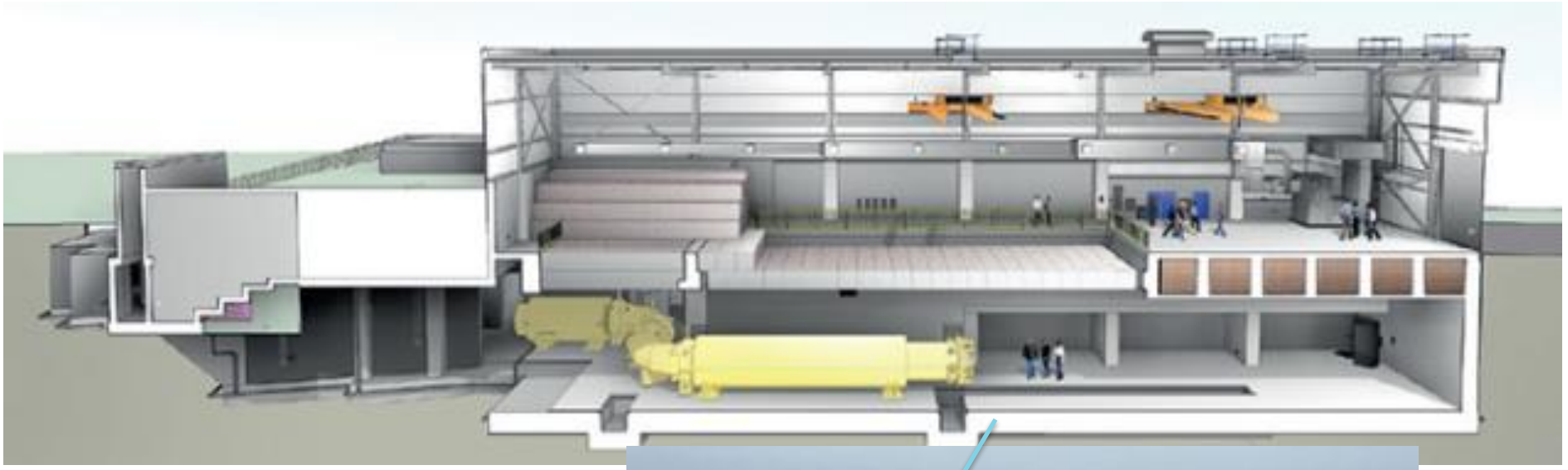
- Introduce Mu2e and TDAQ
- Overview of TDAQ current status
- Details of Timing Distribution
- Student opportunity for Portable Timing System

# What is Mu2e?

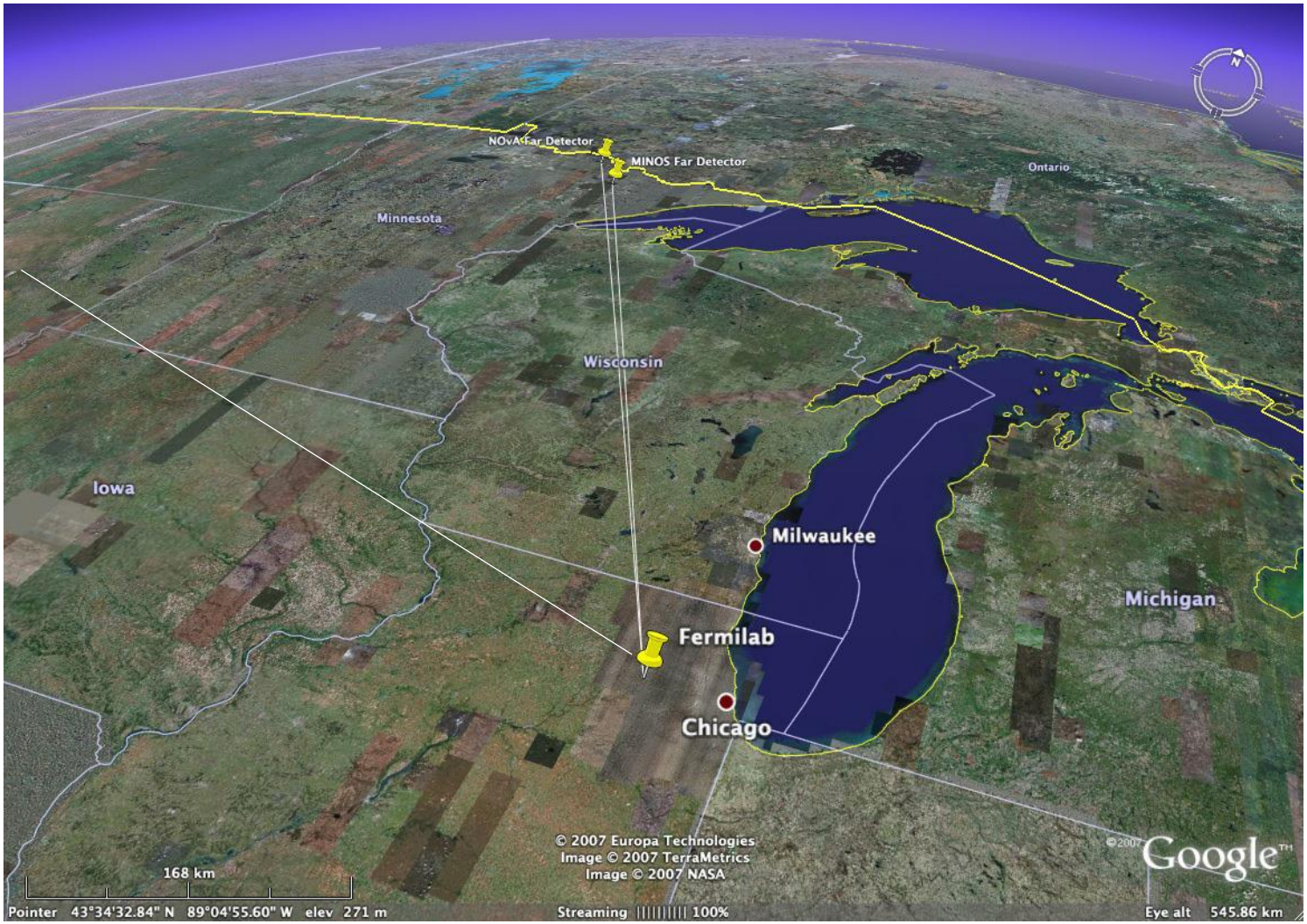
- An experiment at Fermilab (near Chicago) to probe physics beyond the Standard Model.
  - To observe muon-to-electron conversions
  - Observing it, or not, opens the next physics theory doors
- Challenging! Few events per $10^{17}$ stopped muons



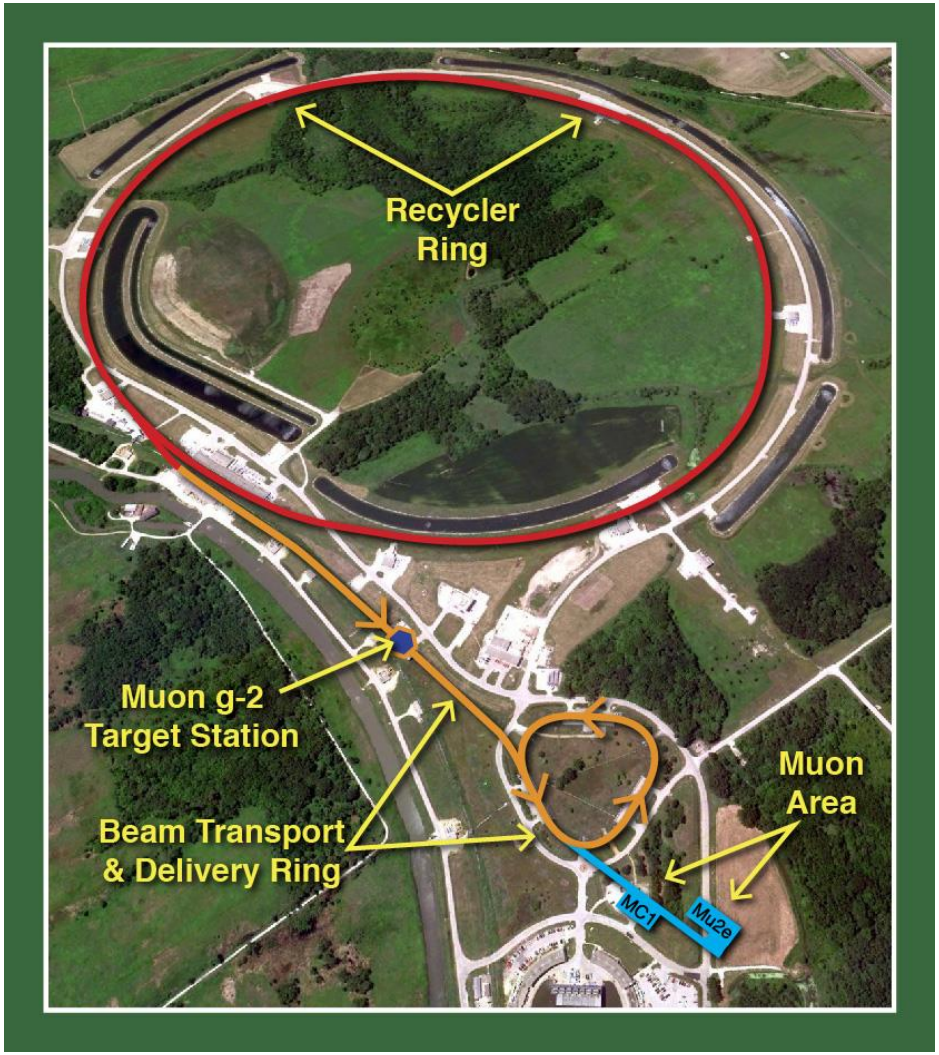
# Where is Mu2e?

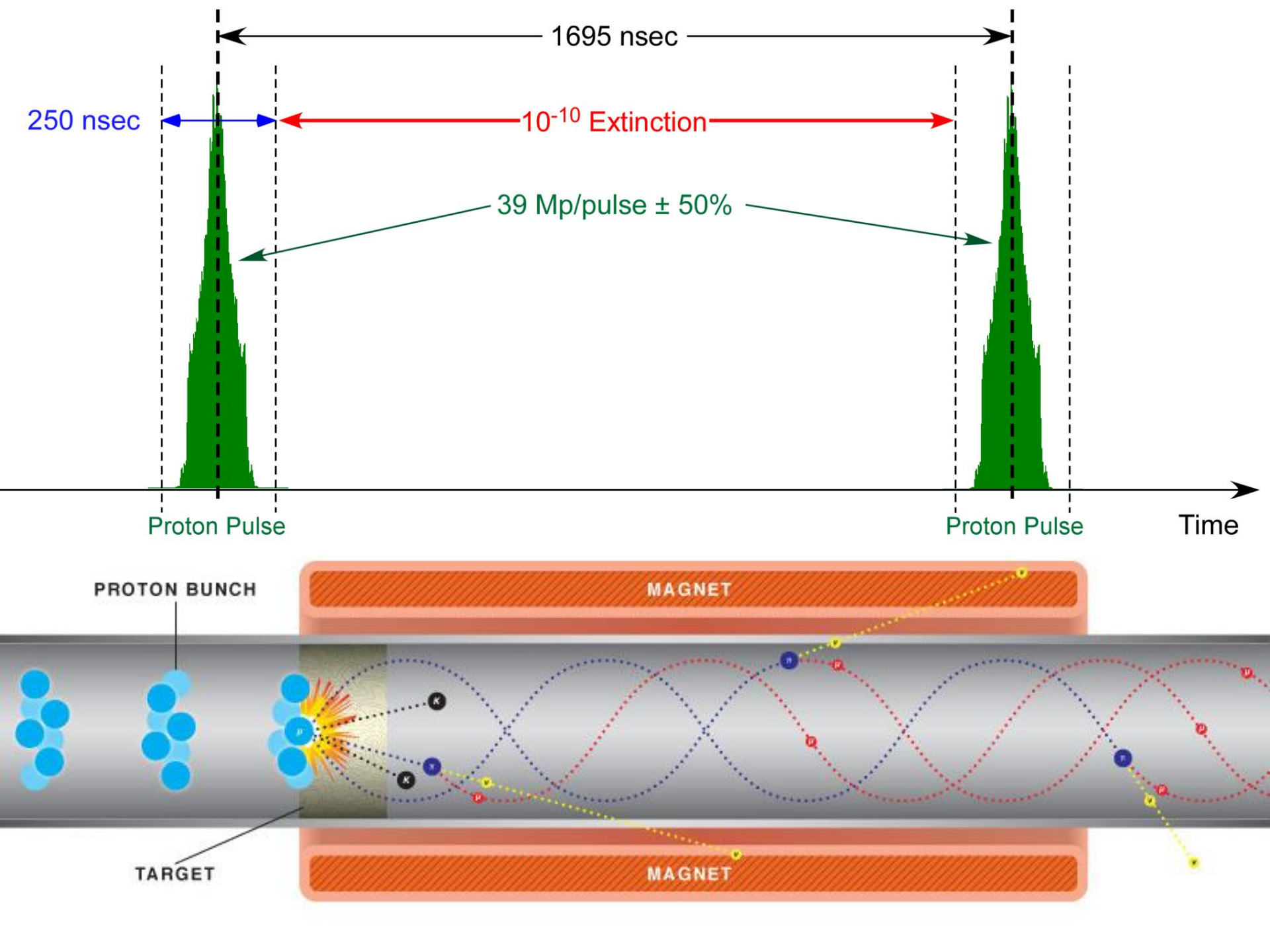


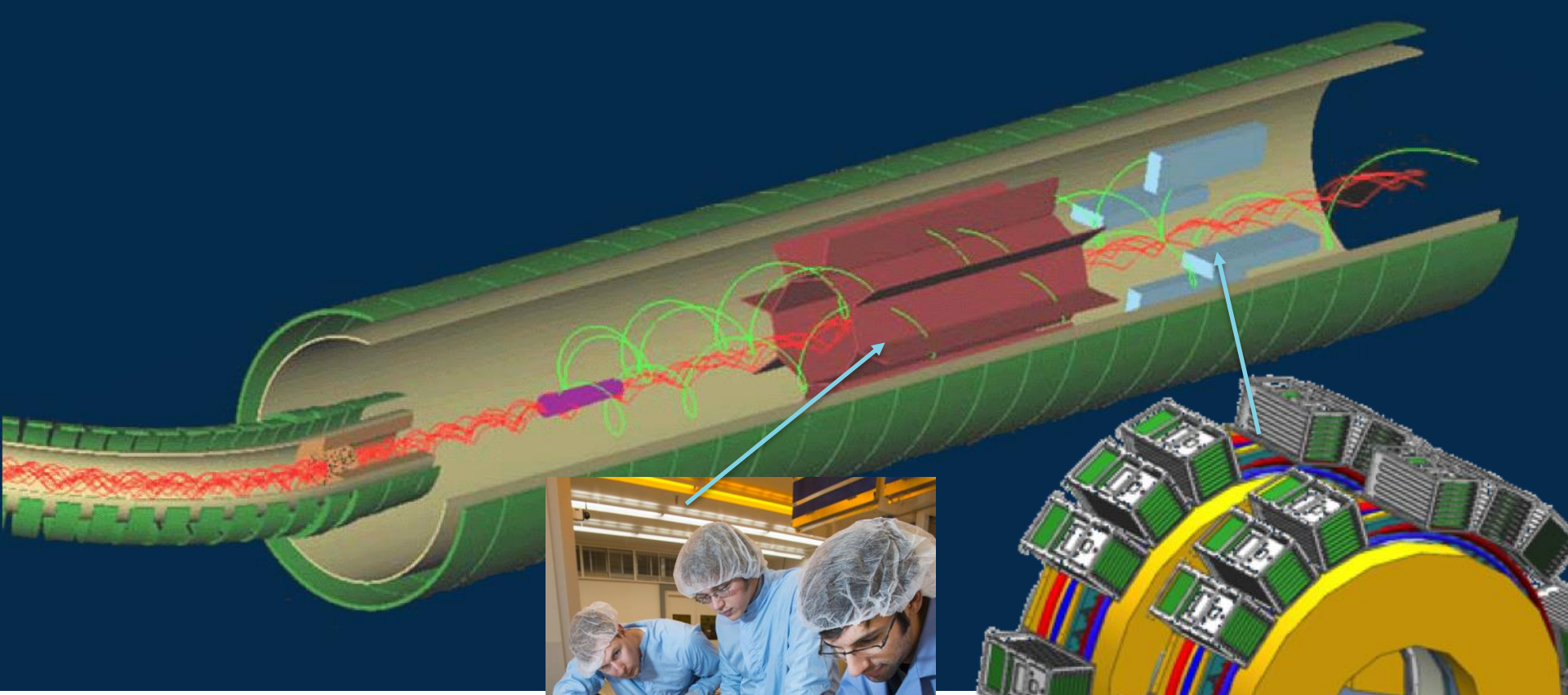
# Where is Mu2e?

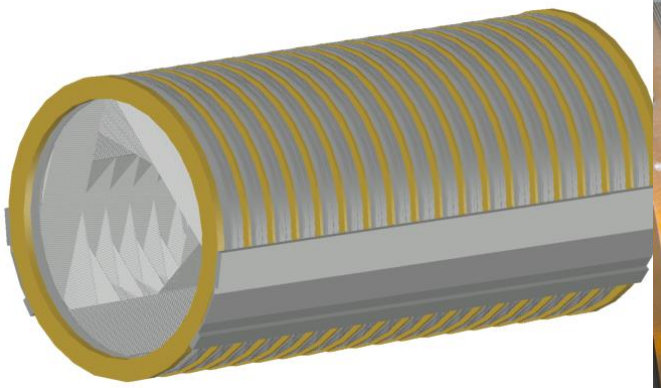


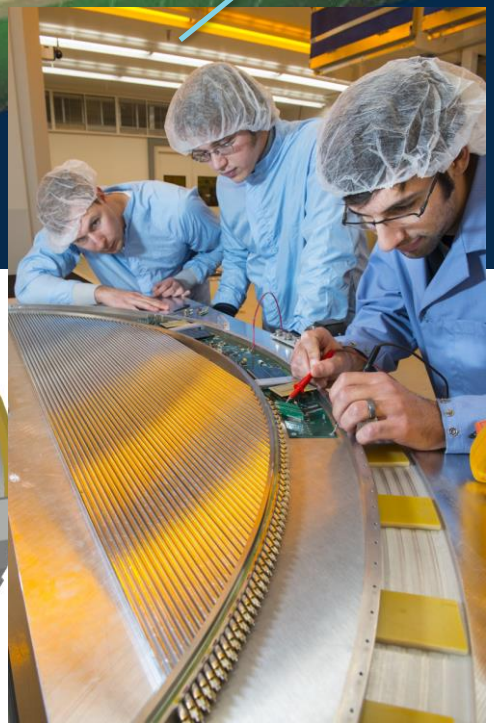
# How does Mu2e work?

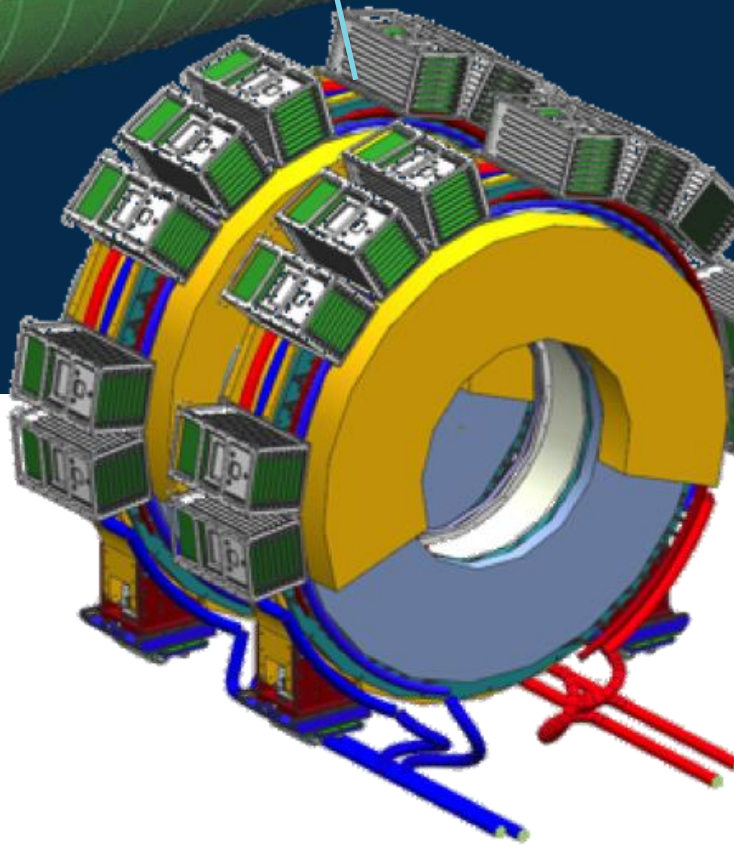



1695 nsec
250 nsec
10-10 Extinction
39 Mp/pulse ± 50%
Proton Pulse
Proton Pulse
Time
PROTON BUNCH
MAGNET
TARGET
MAGNET

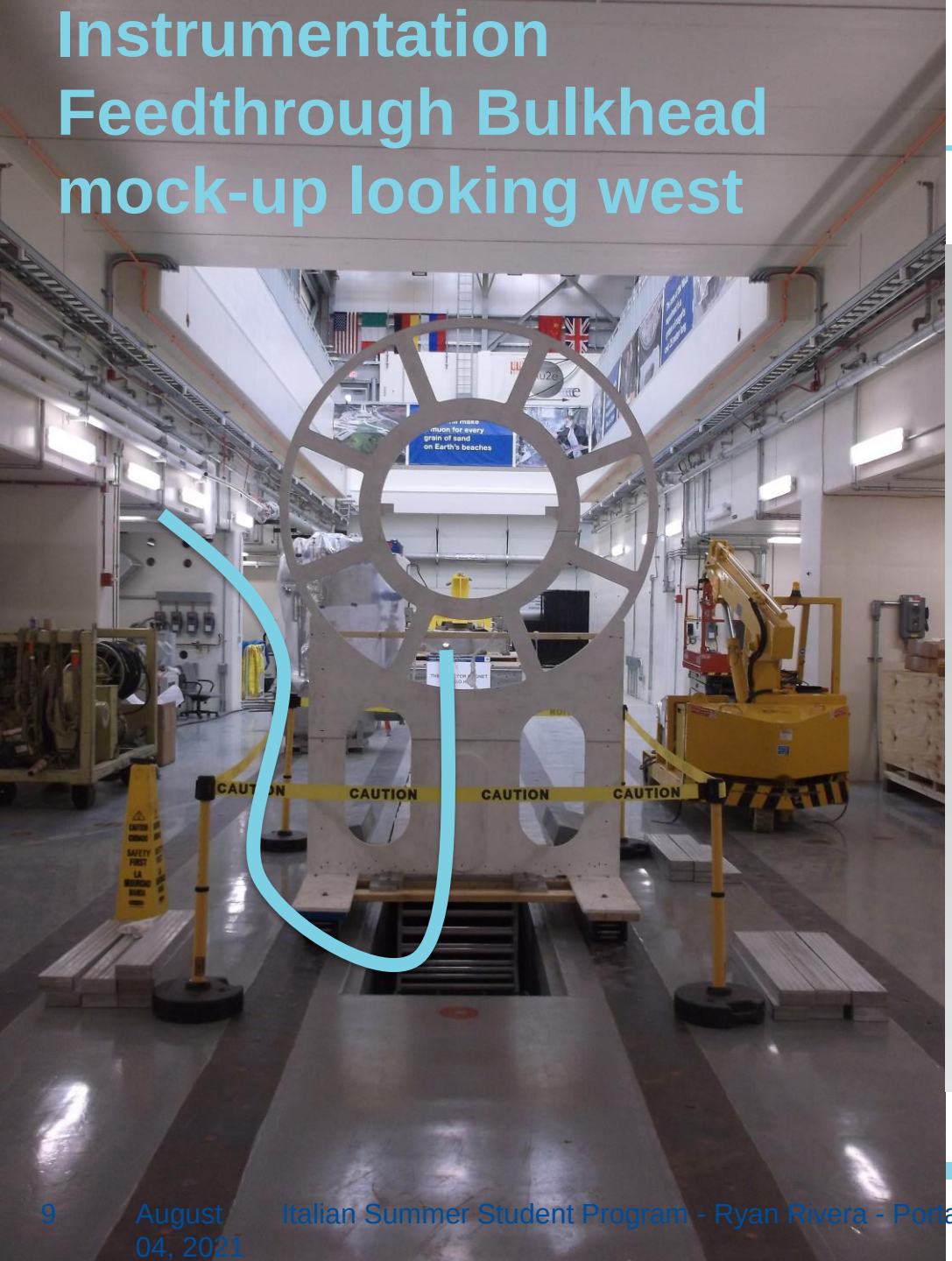


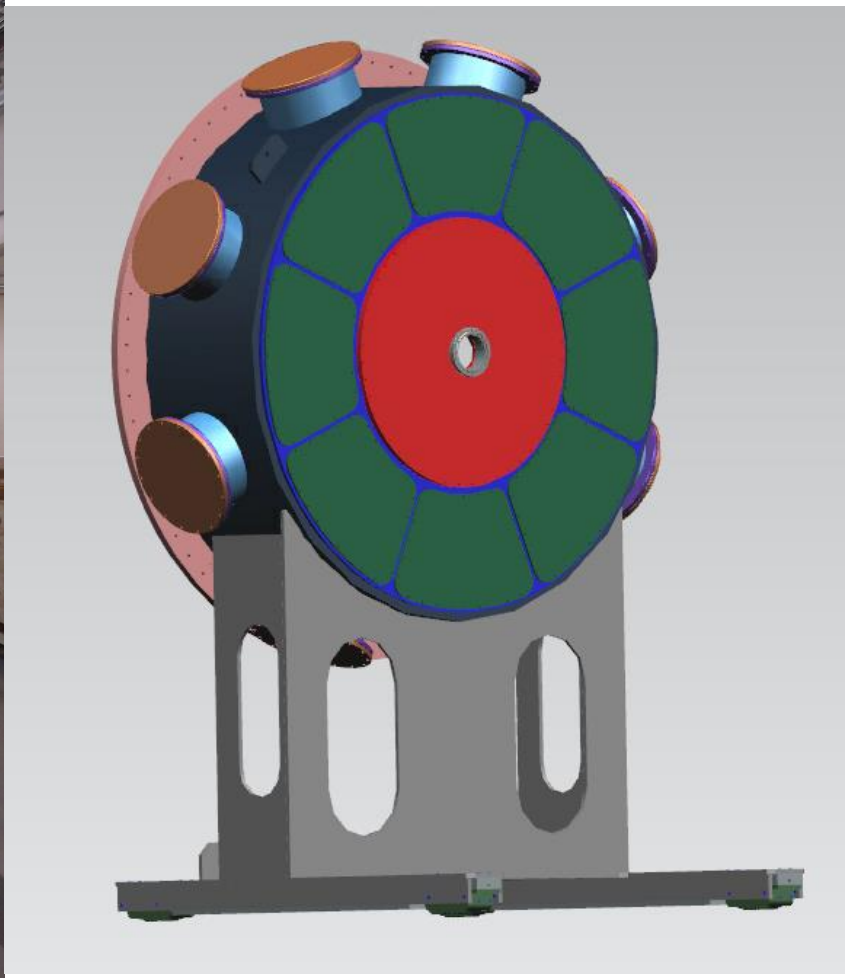






# Instrumentation Feedthrough Bulkhead mock-up looking west

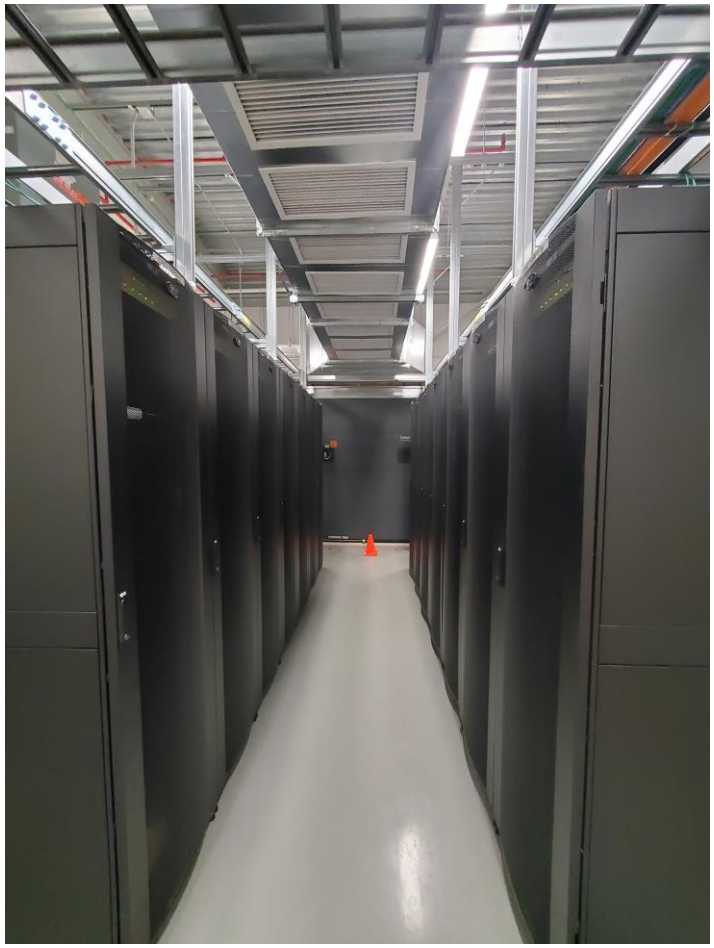


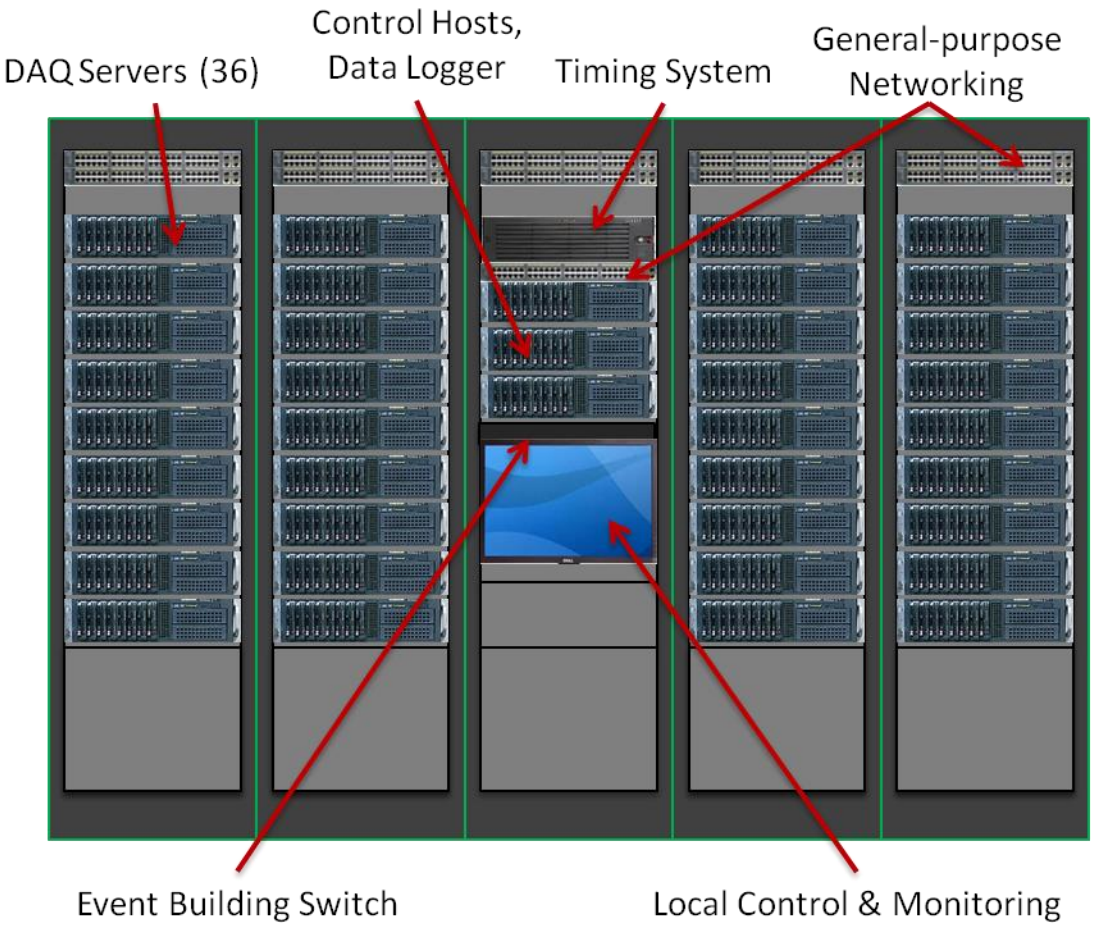





Fermilab

# Racks in DAQ Room





# TDAQ Scope

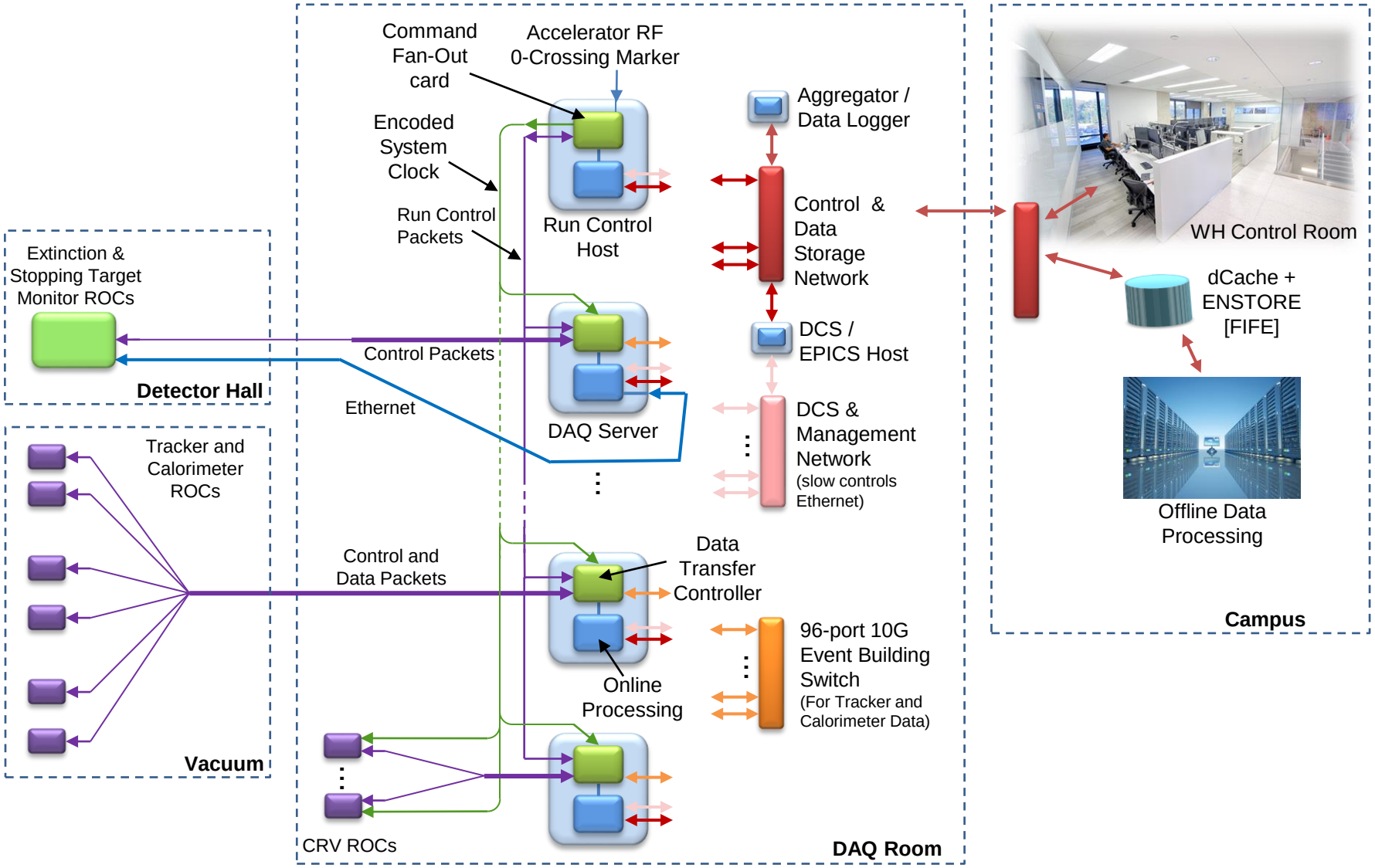
- Trigger & Data Acquisition Includes
  - Optical links between detector and DAQ (bi-directional, control and data)
  - DAQ Servers (detector interface, event building, online processing)
  - Timing System
  - Detector Control System (slow controls)
  - Control room
  - All associated software

- Does not include
  - Detector electronics (digitizers and readout controllers )
  - Safety systems

# TDAQ Subprojects

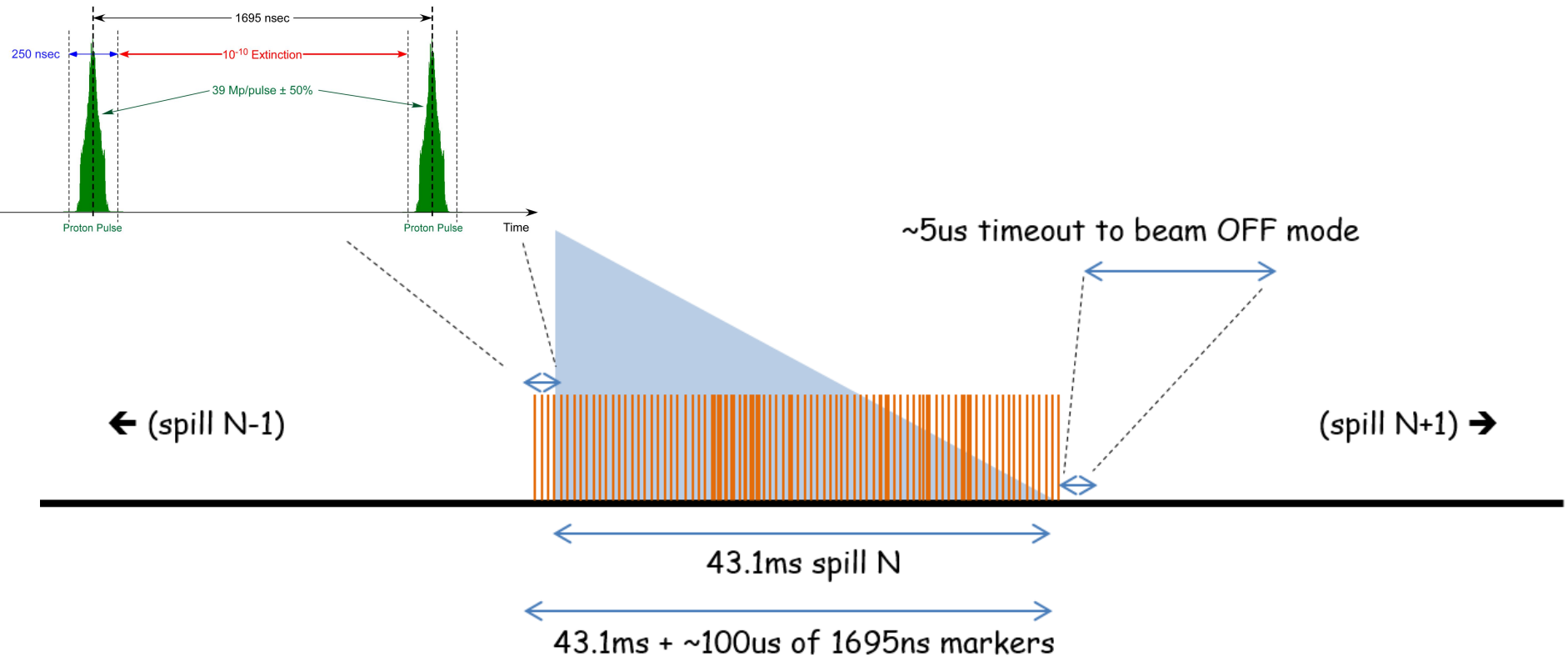
- Management
  - Organization, Schedule, Cost Estimates, QA, Risks, ES&H
- System Design & Test
  - Requirements, System Architecture, System Test
- Data Acquisition
  - Data Readout, Timing System
- Data Processing
  - Online Computing and Data Filters
- Controls & Networking
  - General-purpose Networking, Slow Controls, Control Room

# TDAQ Topology

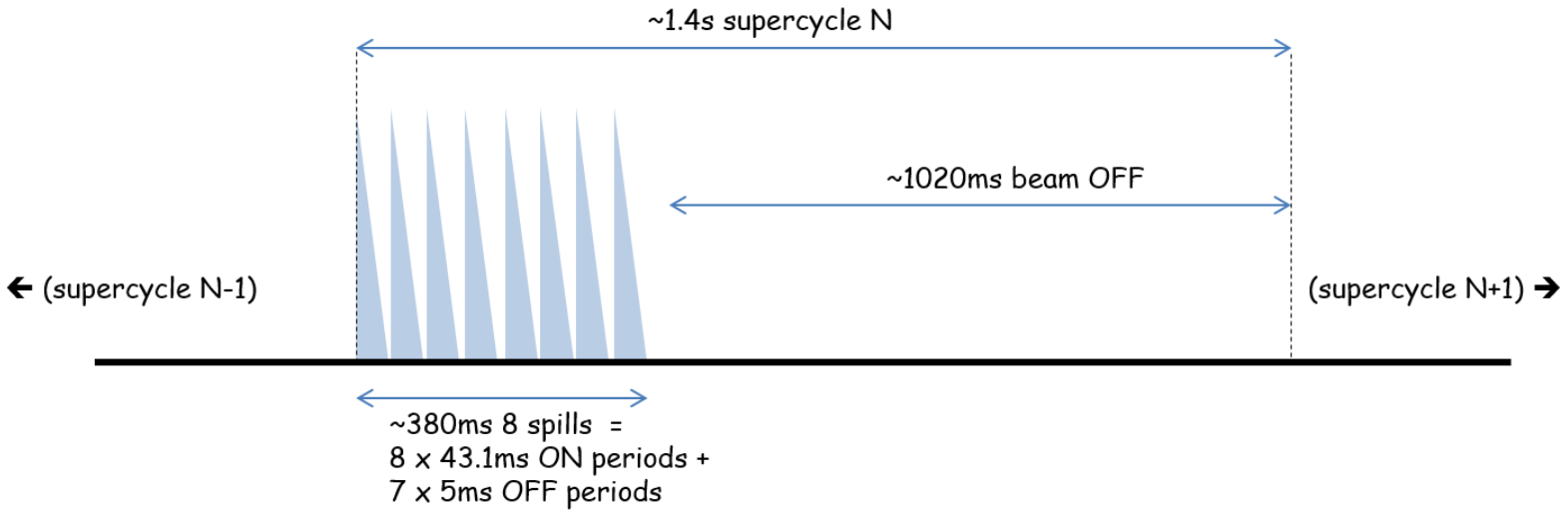


# Mu2e Accelerator Spill

- 25K proton pulses delivered during 43ms spills



# Event Counts per Cycle



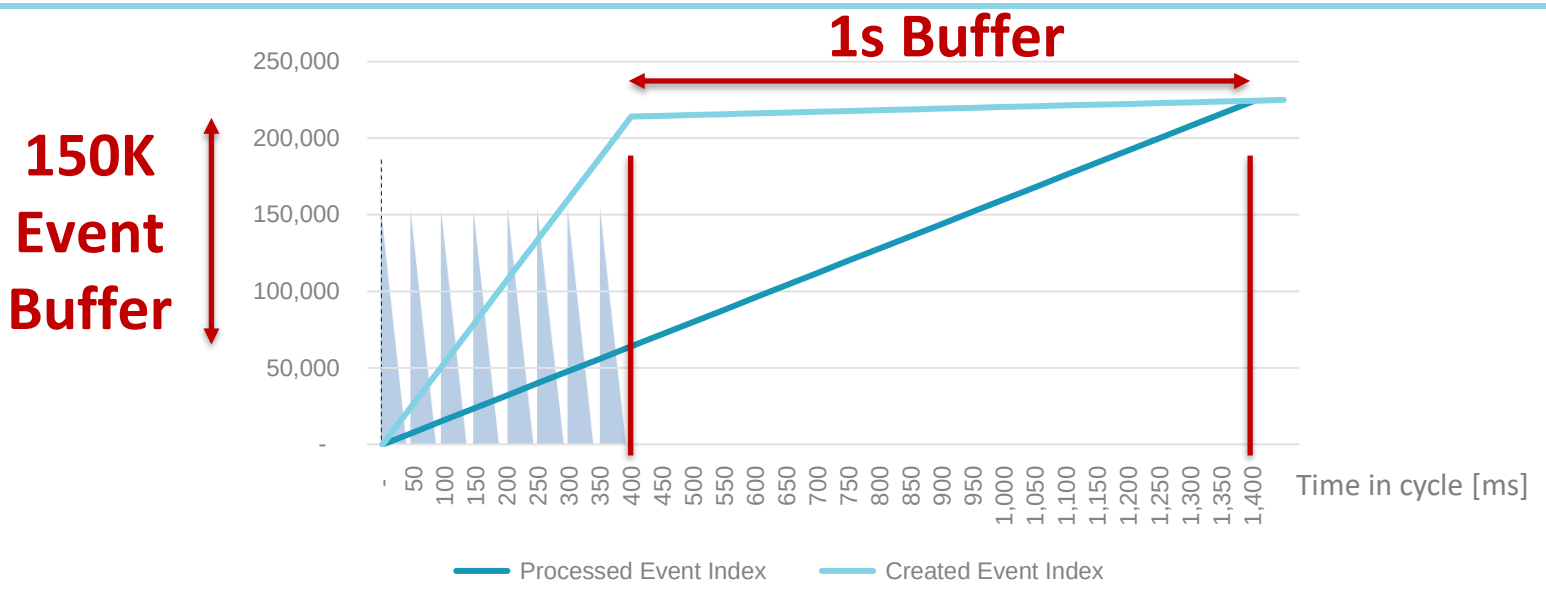

On-spill (0.4s) ➔ 8 * 43.1ms / 1.695µs =
203.4K On-spill events/cycle

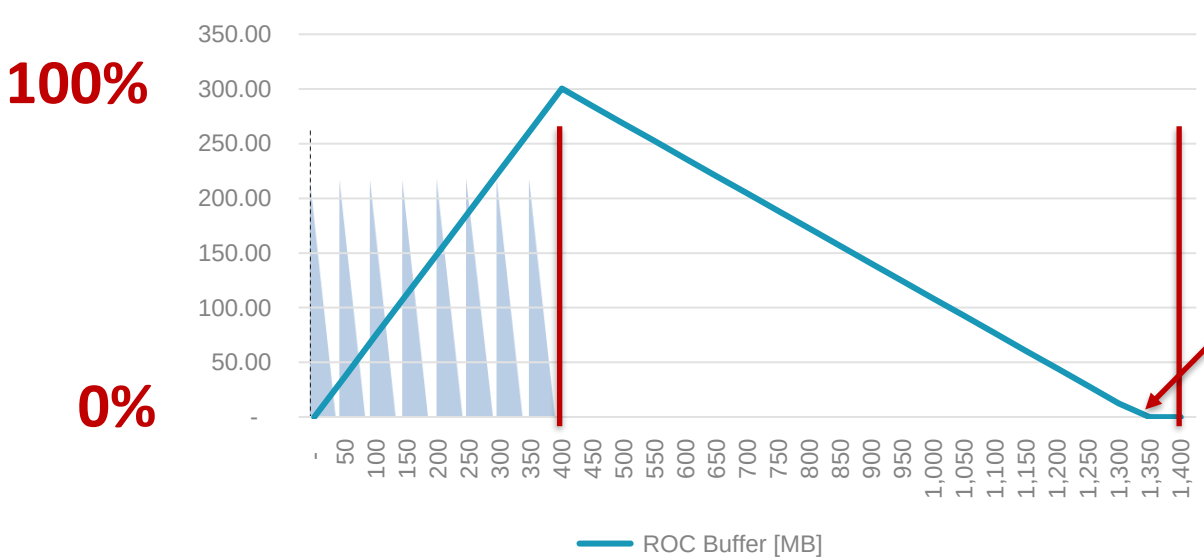
Off-spill (1s) ➔ (7 * 5ms + 1020ms) / 100µs =
10.5K Off-spill events/cycle

# Event Building Design

- **Assumption** that ON-Spill is overwhelming, and OFF-Spill is quiet.
- **Design goal** was to take advantage of OFF-Spill quiet time.
- **Approach** was to invest in *large* front-end buffers to smooth out data transfer over full accelerator cycle.
  - Front-ends required to have at least 1 second buffer.
  - Data from 0.4 seconds is transferred over full 1.4 seconds.
- **Result**:
  - Tracker and Calorimeter ROCs have 0.5 GB each
  - CRV ROCs have 1s buffers in FEBs (4 GB/FEB?).
  - PRE-Switch DAQ FPGAs have 2 GB each
  - POST-Switch DAQ FPGAs have 2 GB each

# ROC Buffer and Event Index during cycle




Assumed 2MB <FE event size>

Empty by end of 1.4s cycle

# TDAQ Current Status

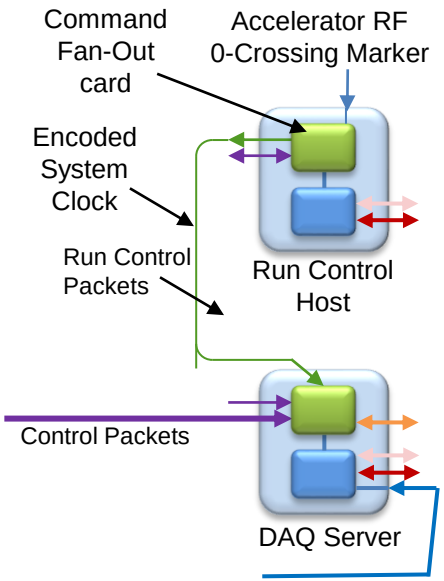
- Trigger & DAQ team is in debug and optimization phase:
  - balancing vertical and horizontal slice support while delivering on demonstrations.
  - Chain-of-10 DTCs demonstrated in FY21
  - Full-scall Hardware Event Building not yet demonstrated (not needed for KPP)
- Collaboration has been active (though we could always use more help – like Italian students!):
  - Trigger, DCS/EPICS, Vertical Slice Tests, DQM
- Still need to buy servers
  - needed for full-scale hardware event building
- Detector timing calibration and verification tools will be valuable!

# Portable Mu2e TDAQ

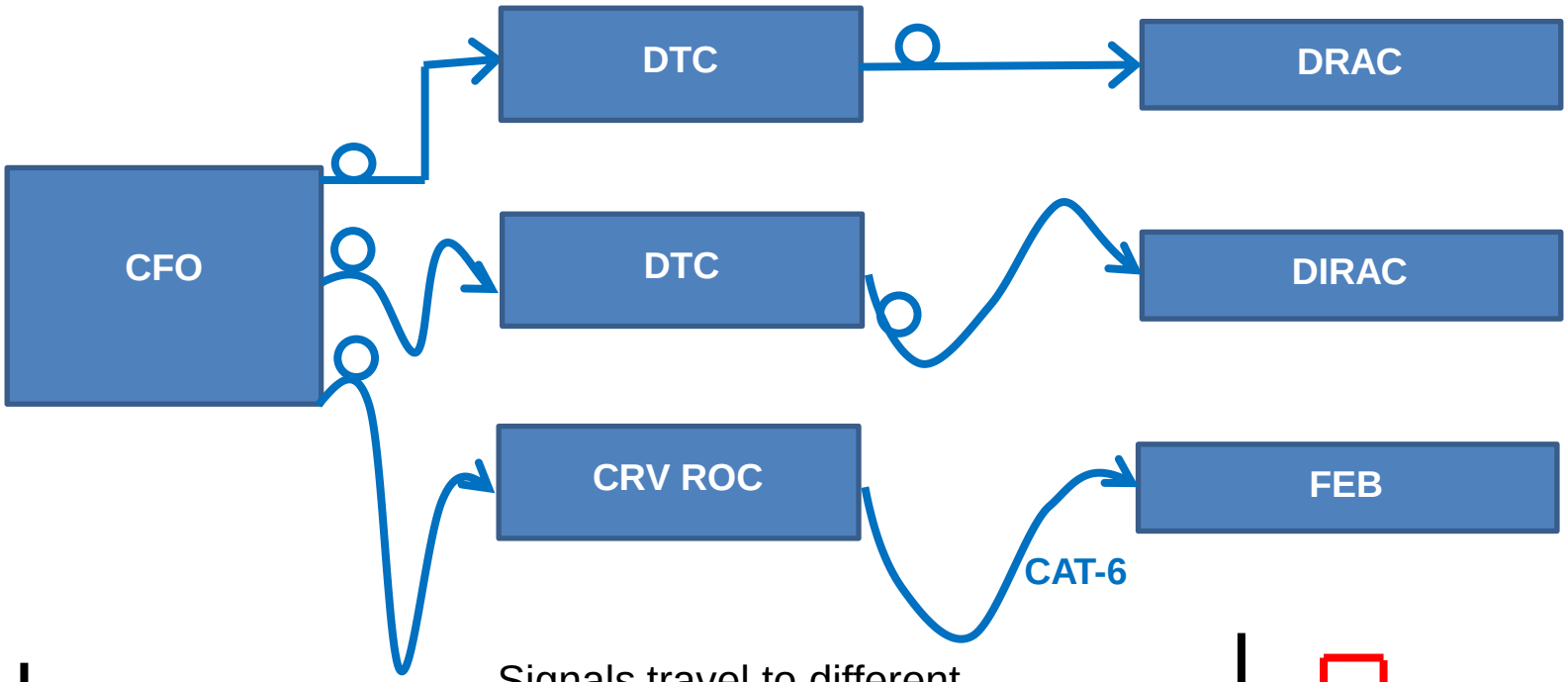
- The concept is a mobile timing verification unit that is the golden standard for Mu2e timestamping of data
  - Components:
    - Detector
    - Readout linked to TDAQ
    - Mechanics to keep it safe and make it user friendly
- Could be an ideal project for mechanical and electrical engineers to work together
- How does Mu2e timestamping work?

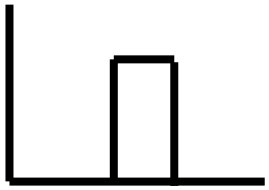
# Top of the Mu2e Timing Tree

- Top of the timing tree is the Command Fan-Out module (CFO)
  - PCIe FPGA card in a TDAQ server
- **Inputs** to CFO
  - RF0 signal ← from Accelerator
  - Run Plan to specify how to collect data and/or calibrate
- **Outputs** from CFO
  - Mu2e system clock:  40 MHz (25 ns)
  - Start-of-event-window marker
  - Heartbeat packet (16 bytes) to specify the detail of each Event Window



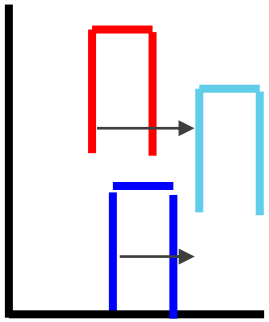
# What does Event Window synchronization mean?



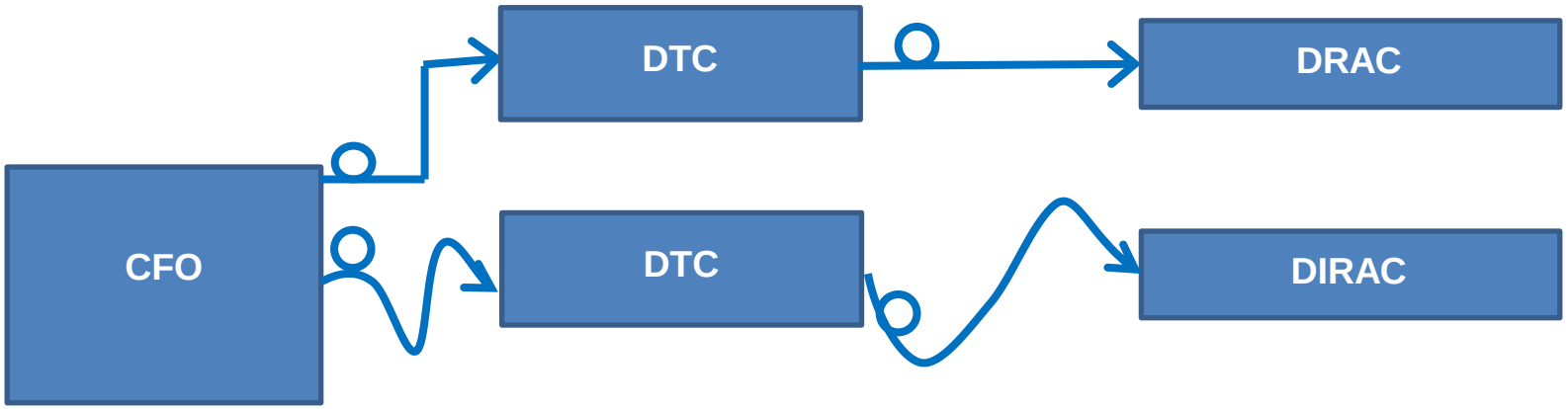
Event Window is defined at CFO.

Signals travel to different boards and through fibers/cables of different lengths.



Must line up Event Window at timestamping front-ends.

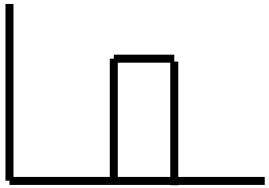
# Pictorially: Event Window synchronization

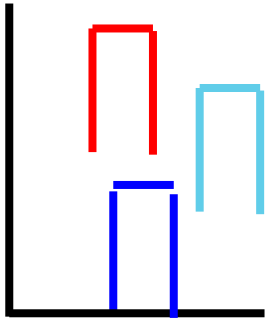


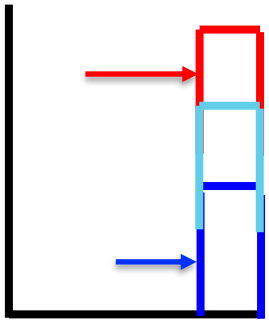

Event Window defined at CFO

Step 1. Measure travel time through different boards and through fibers of different lengths

Step 2: after delays applied at ROC: Event Windows synchronized at timestamping front-ends



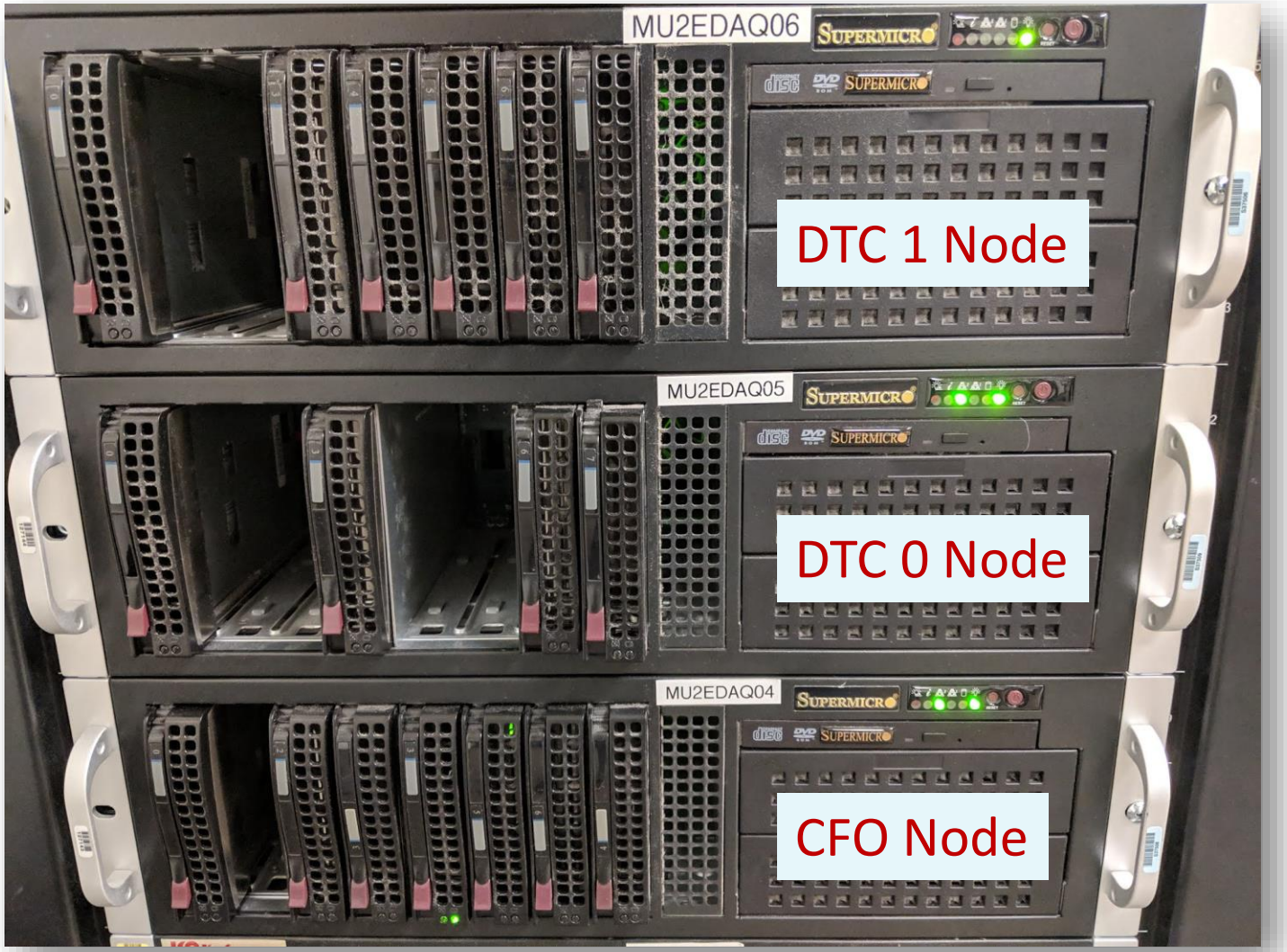




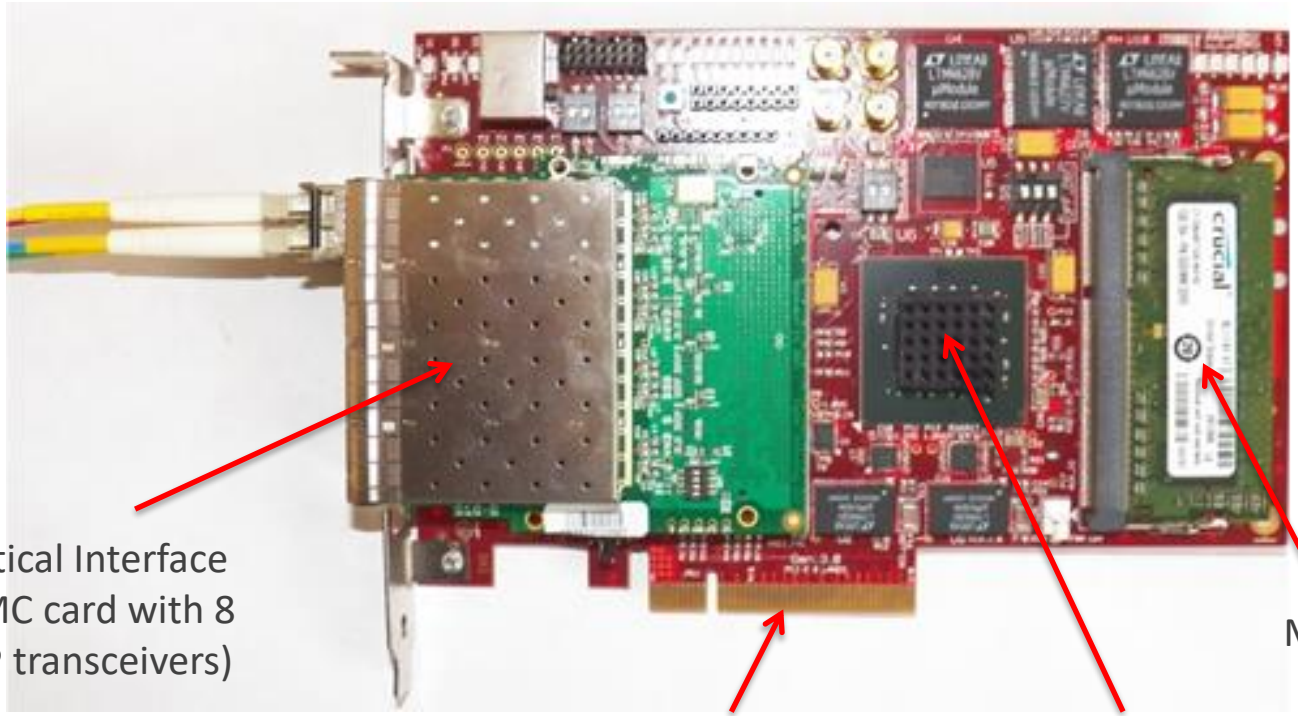
# Event Window Synchronization

- To line up Event Windows the approach is to delay each front-end to match front-end with longest latency.
- How do we determine delay?
  - Calculate from Signal Loopback

# The Test Stand

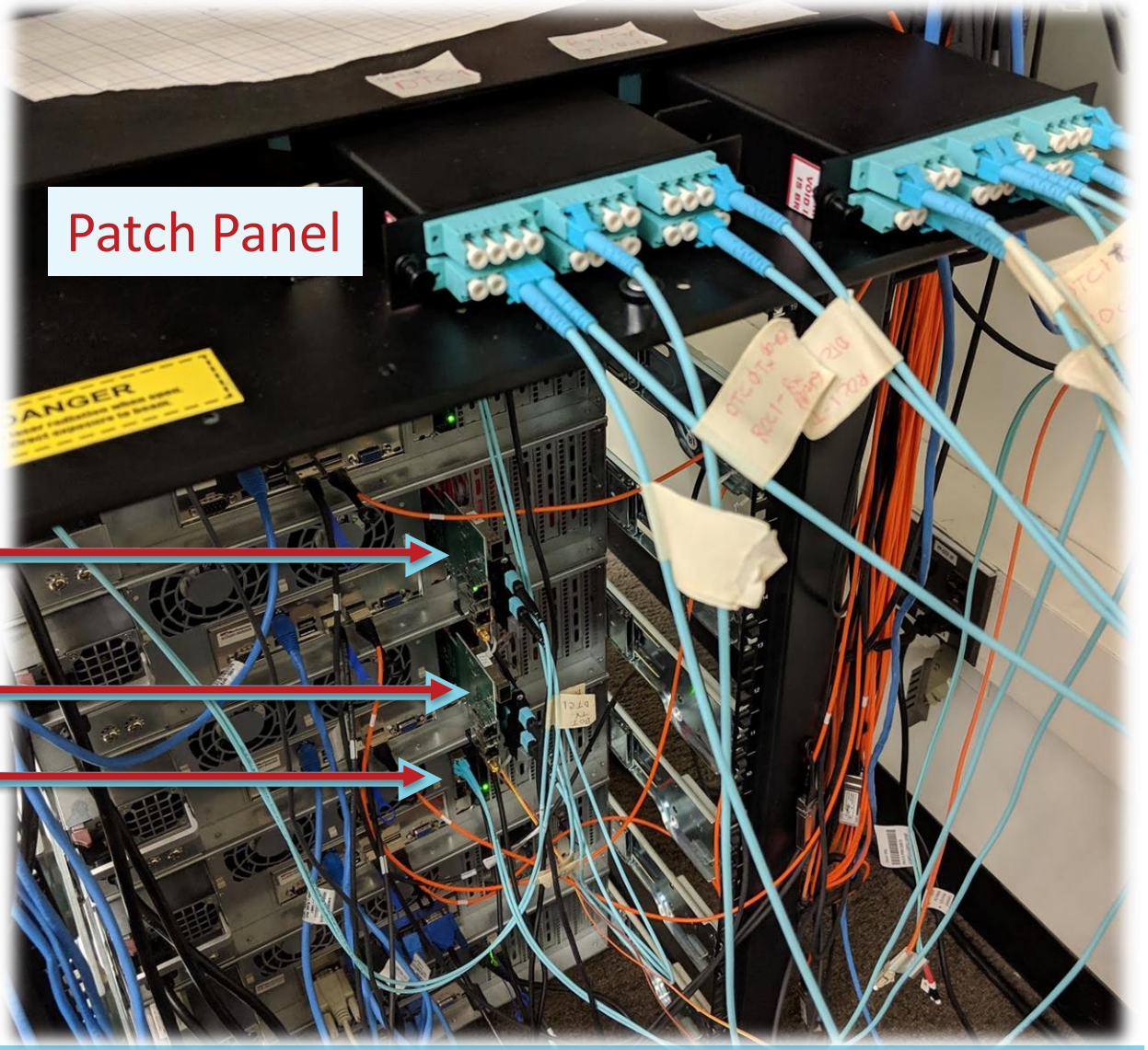


# Data Acquisition

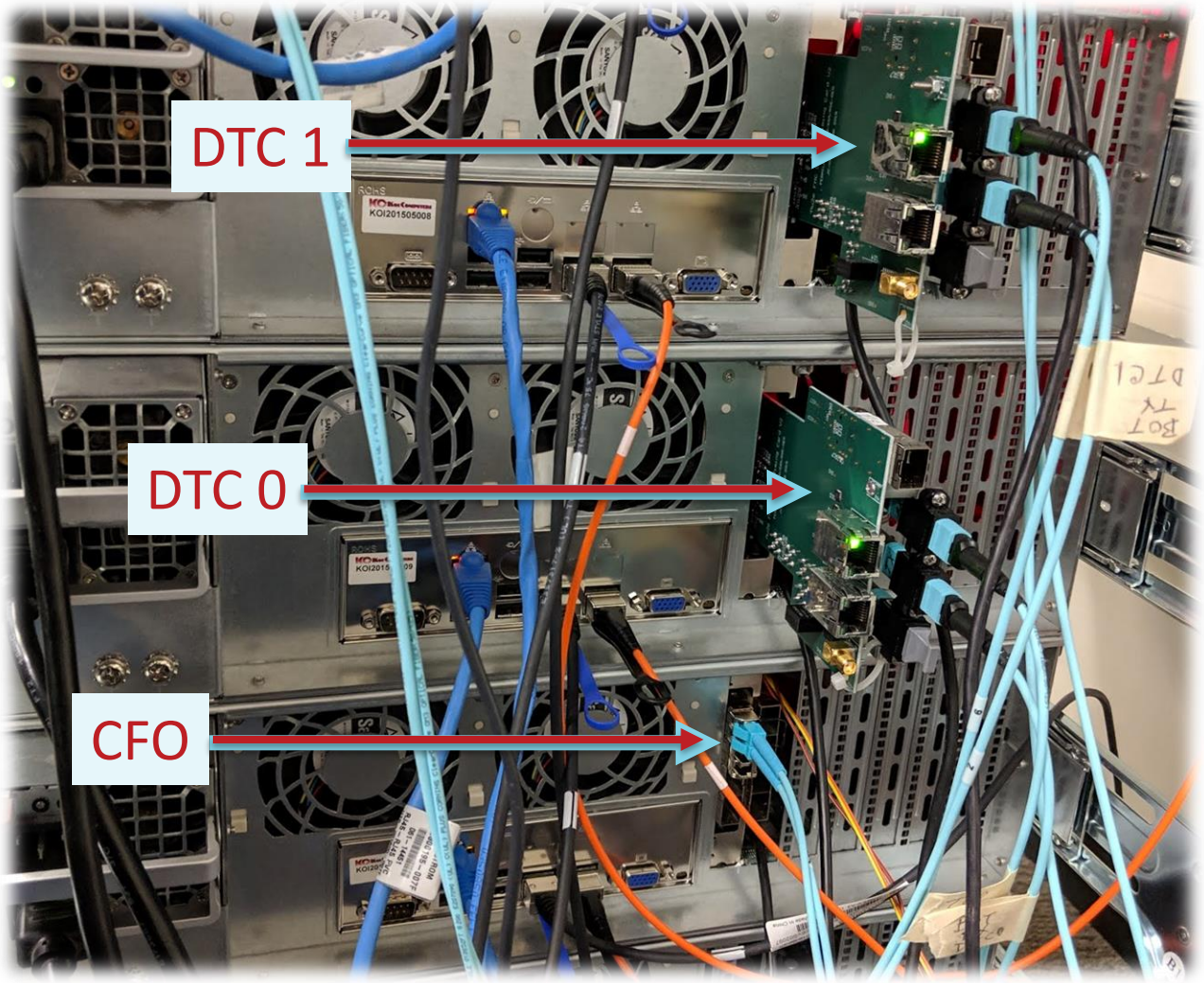



Data Transfer Controller (DTC)

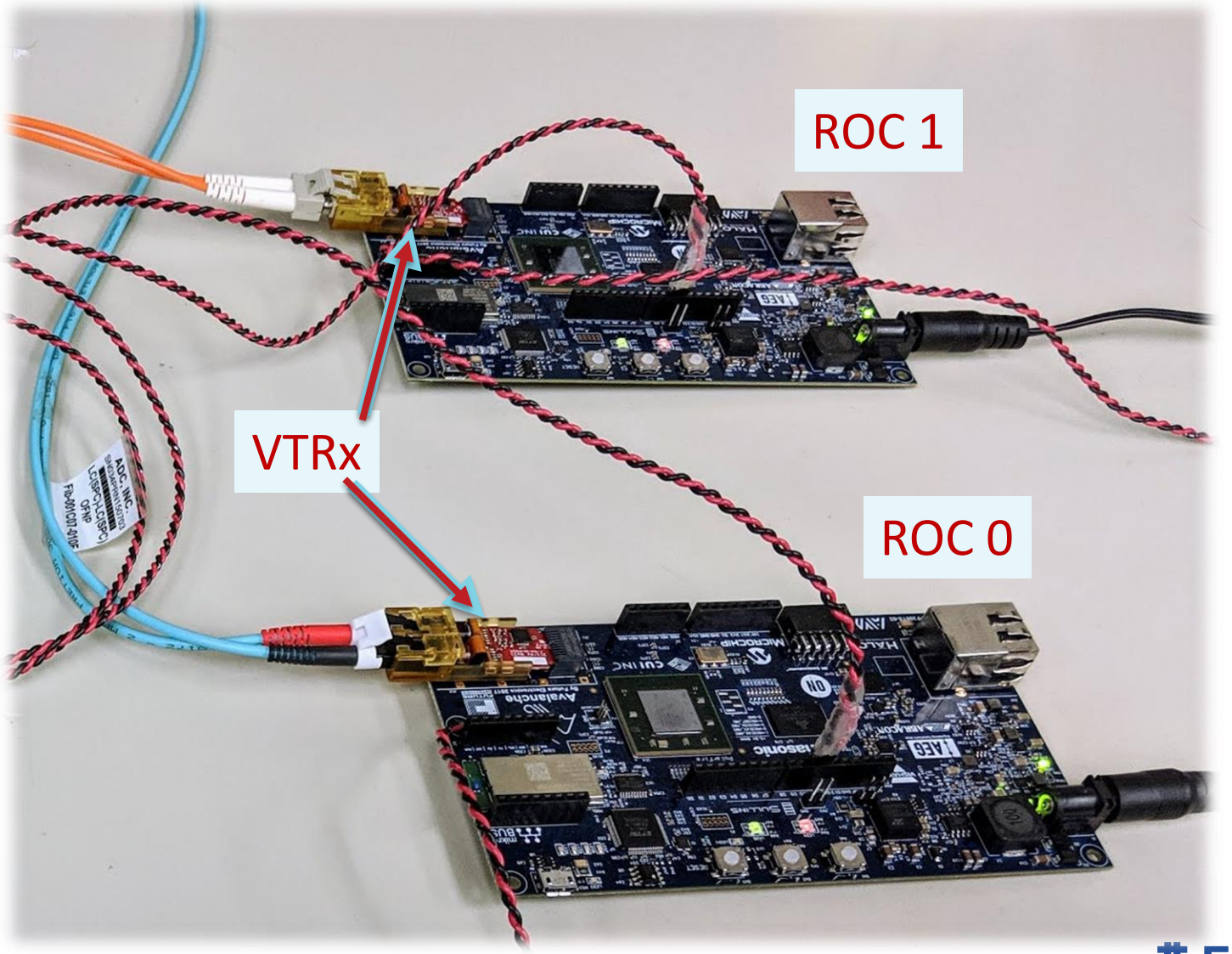
# The Test Stand



# The Test Stand



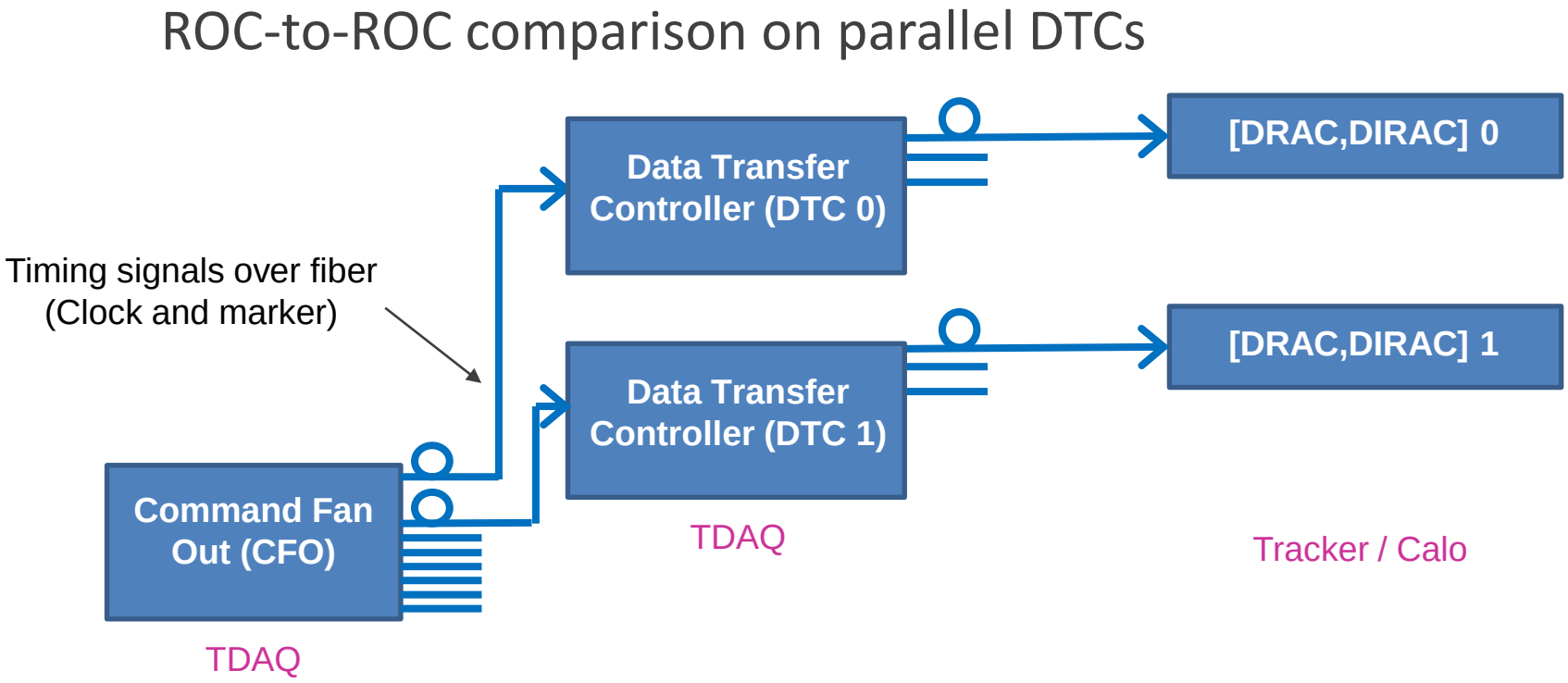
# The Test Stand



# Jitter Measurement Approach

- We want to measure jitter from ROC-to-ROC
  - We use timestamping clock from ROC0 as trigger and we sample timestamping clock from ROC1
  - Timestamping clock for these measurements was 200MHz (5ns periods)

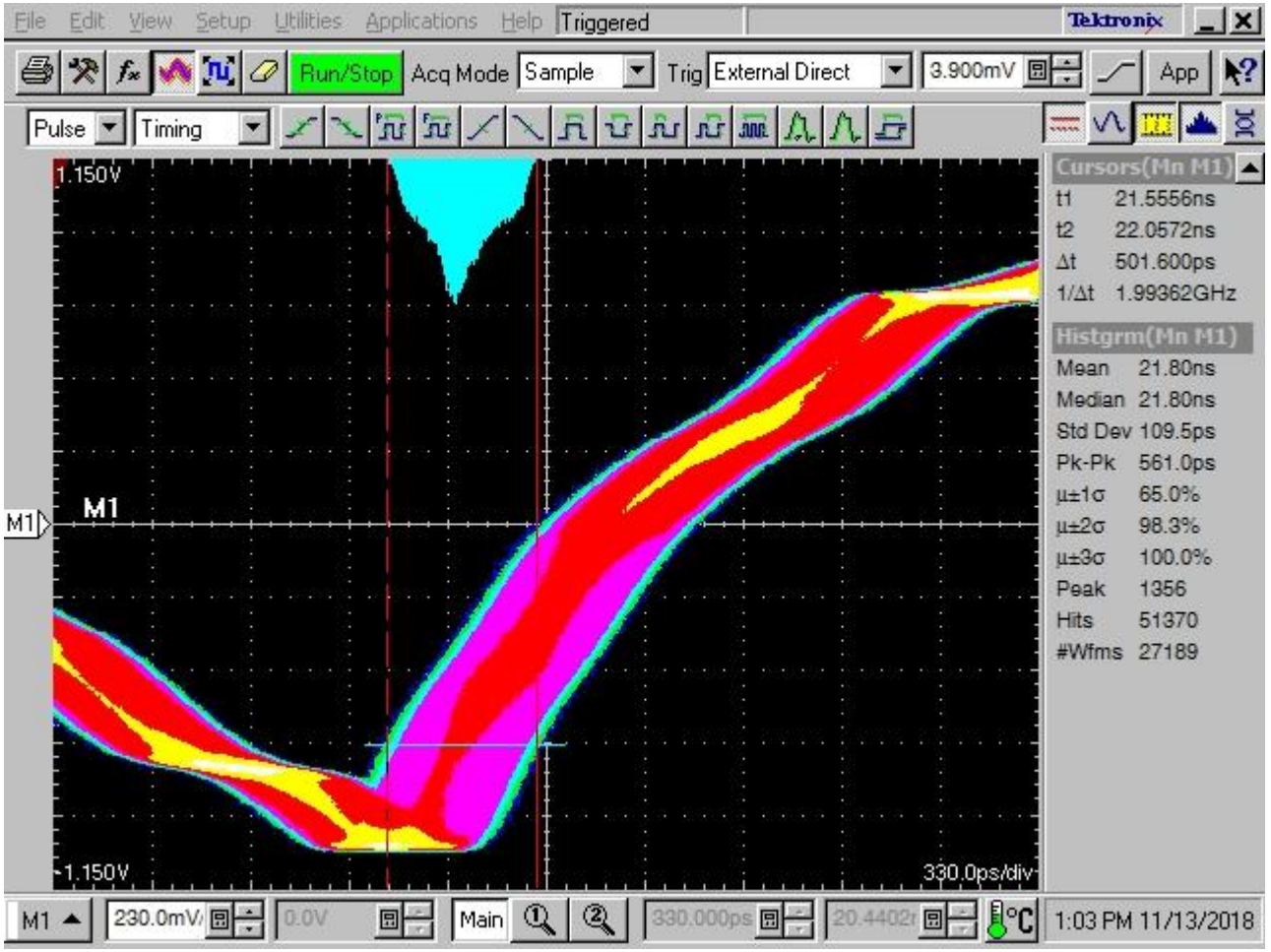
# Jitter Test Topology



# Jitter Test Topology Results

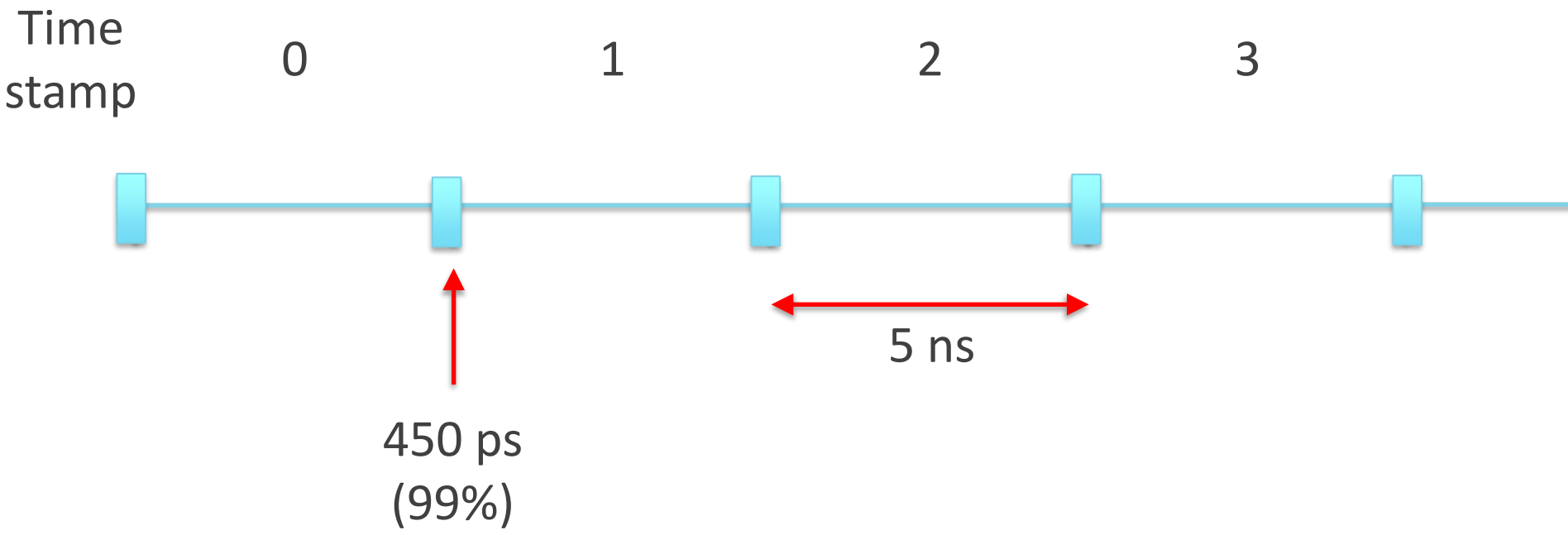
ROC-to-ROC comparison on parallel DTCs

- 27K samples
- StdDev = 109.5ps
- 65% of samples were in a 220ps window.
- 98% of samples were in a 440ps window.
- 100% of samples were in a 561ps window.



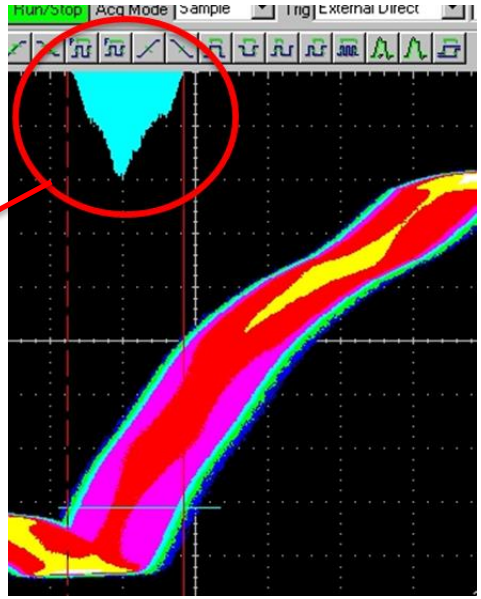
# Effect of jitter

The effect of the jitter is to blur the clock edges after the Event Window synchronization (t=0)

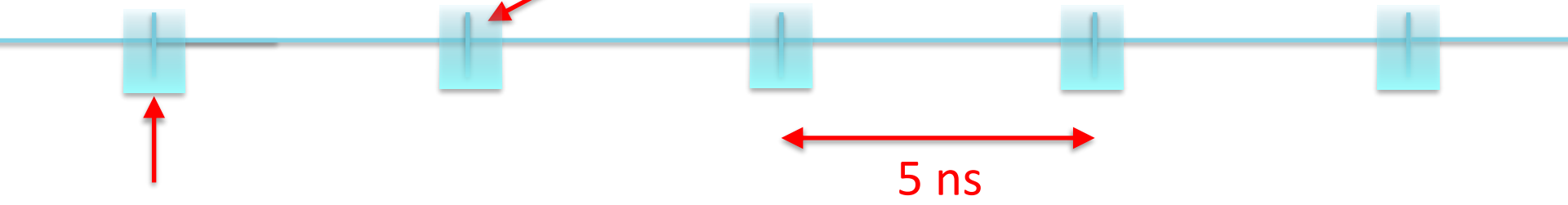


# ROC timestamping

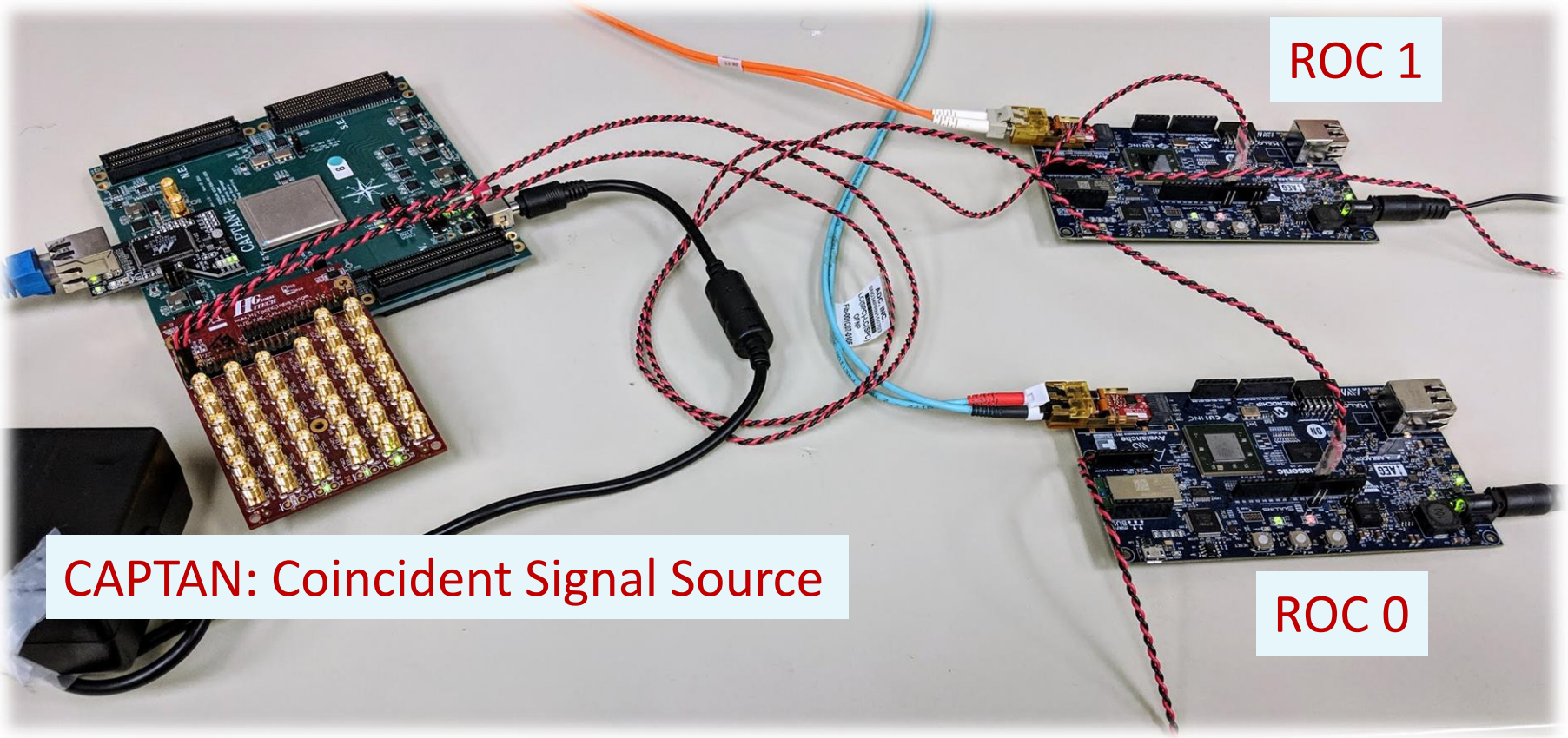
The effect of the jitter is to blur the clock edges after the Event Window synchronization (T=0)



Example:
timestamps @ 200 MHz

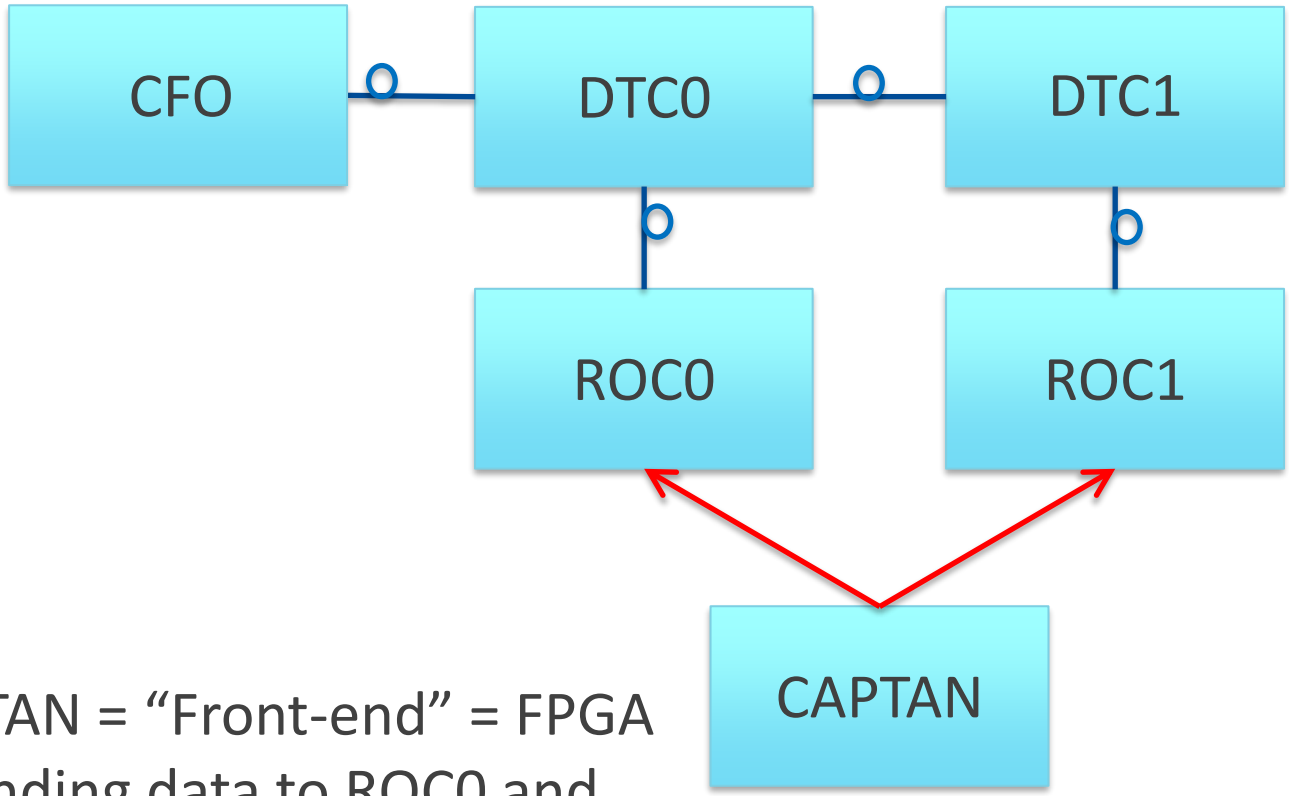


# The Timestamping Test Stand



# Timestamping setup

Idea: timestamp asynchronous data received simultaneously at the ROCs



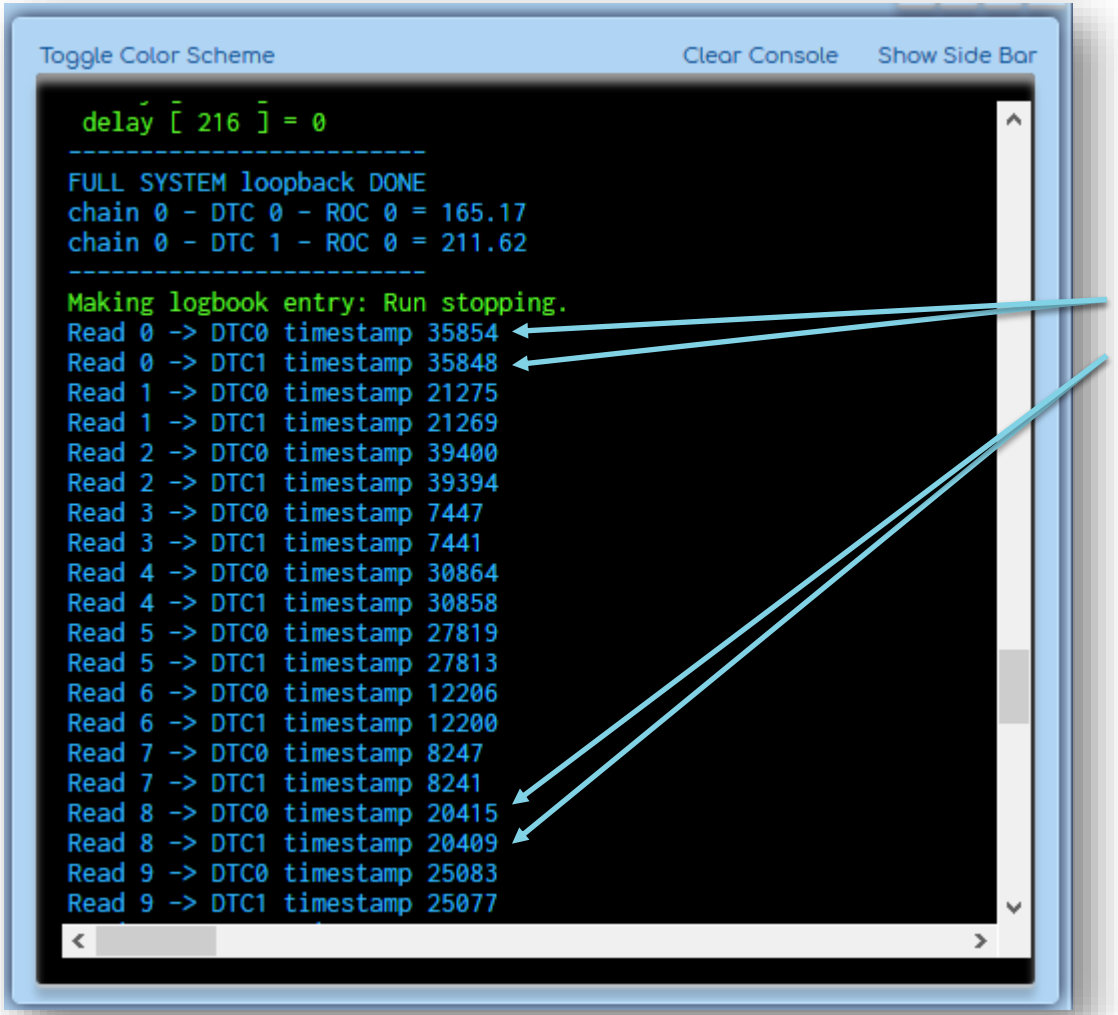

CAPTAN = “Front-end” = FPGA sending data to ROC0 and ROC1 on wires of same length

# Loopback Measurements

```
Info (17:44:53) CFOFrontEndInterface:  Looping back DTC0 ROC0
Debug (17:44:58) CFOFrontEndInterface: LOOPBACK: on DTC 0
Debug (17:44:58) CFOFrontEndInterface:  delay [ 155 ] = 0
Debug (17:44:58) CFOFrontEndInterface:  delay [ 156 ] = 0
Debug (17:44:58) CFOFrontEndInterface:  delay [ 157 ] = 0
Debug (17:44:58) CFOFrontEndInterface:  delay [ 158 ] = 0
Debug (17:44:58) CFOFrontEndInterface:  delay [ 159 ] = 0
Debug (17:44:58) CFOFrontEndInterface:  delay [ 160 ] = 83
Debug (17:44:58) CFOFrontEndInterface:  delay [ 161 ] = 17
Debug (17:44:58) CFOFrontEndInterface:  delay [ 162 ] = 0
Debug (17:44:58) CFOFrontEndInterface:  delay [ 163 ] = 0
Debug (17:44:58) CFOFrontEndInterface:  delay [ 164 ] = 0
Debug (17:44:58) CFOFrontEndInterface:  delay [ 165 ] = 0
Info (17:45:02) CFOFrontEndInterface:  Looping back DTC1 ROC0
Debug (17:45:07) CFOFrontEndInterface: LOOPBACK: on DTC 0
Debug (17:45:07) CFOFrontEndInterface:  delay [ 211 ] = 0
Debug (17:45:07) CFOFrontEndInterface:  delay [ 212 ] = 0
Debug (17:45:07) CFOFrontEndInterface:  delay [ 213 ] = 0
Debug (17:45:07) CFOFrontEndInterface:  delay [ 214 ] = 0
Debug (17:45:07) CFOFrontEndInterface:  delay [ 215 ] = 0
Debug (17:45:07) CFOFrontEndInterface:  delay [ 216 ] = 38
Debug (17:45:07) CFOFrontEndInterface:  delay [ 217 ] = 62
Debug (17:45:07) CFOFrontEndInterface:  delay [ 218 ] = 0
Debug (17:45:07) CFOFrontEndInterface:  delay [ 219 ] = 0
Debug (17:45:07) CFOFrontEndInterface:  delay [ 220 ] = 0
Debug (17:45:07) CFOFrontEndInterface:  delay [ 221 ] = 0
Info (17:45:07) CFOFrontEndInterface: ------------------------
Info (17:45:07) CFOFrontEndInterface: FULL SYSTEM loopback DONE
Info (17:45:07) CFOFrontEndInterface: chain 0 - DTC 0 - ROC 0 = 160.17
Info (17:45:07) CFOFrontEndInterface: chain 0 - DTC 1 - ROC 0 = 216.62
```

First guess is to apply 56/2 = 28 clocks of offset. But path out may not perfectly match latency of path back.

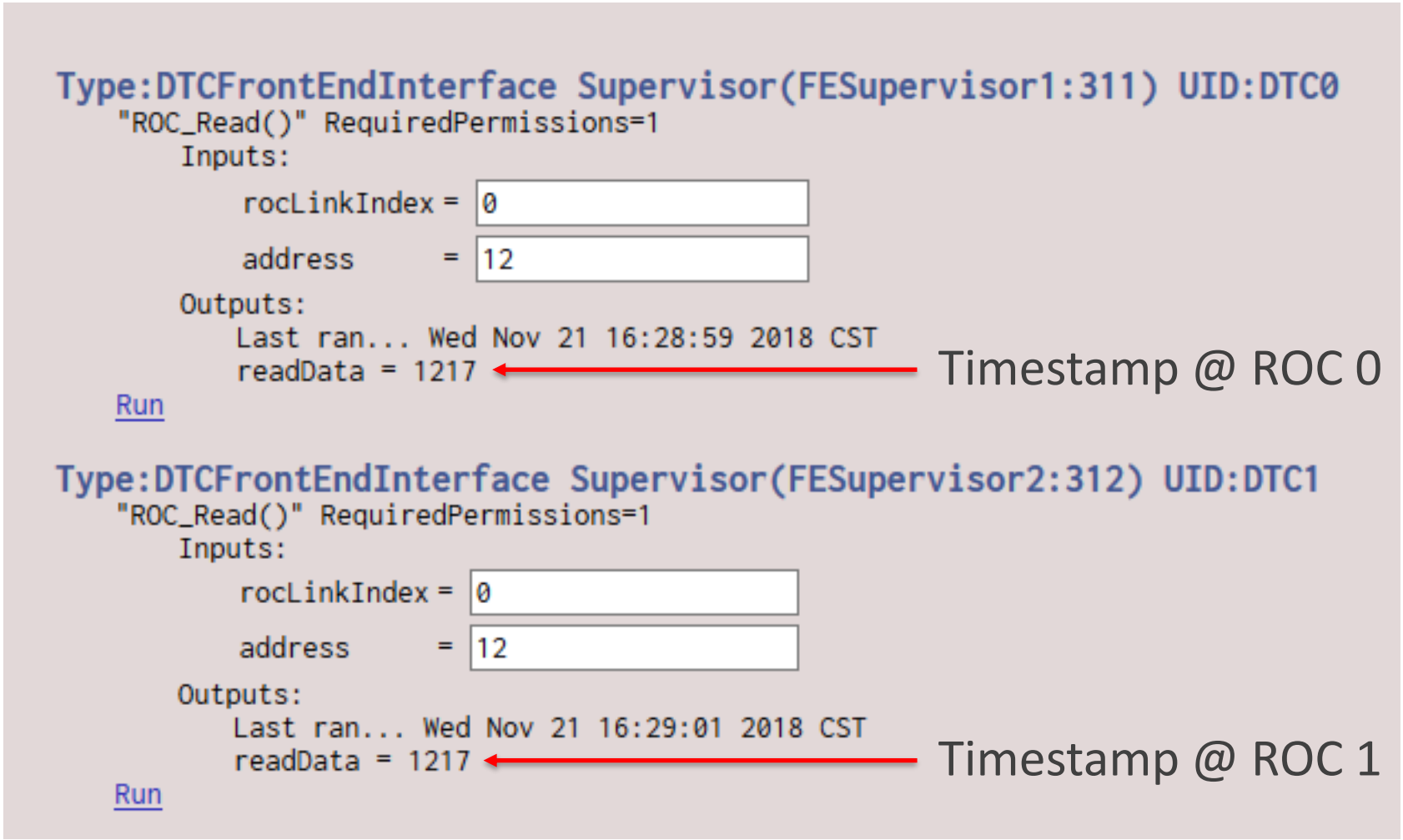
# Timestamping Measurements




Consistent delta of 6.

So apply +6 to DTC0 event window offset (to delay timestamp reset moment).

# Manual Timestamping GUI

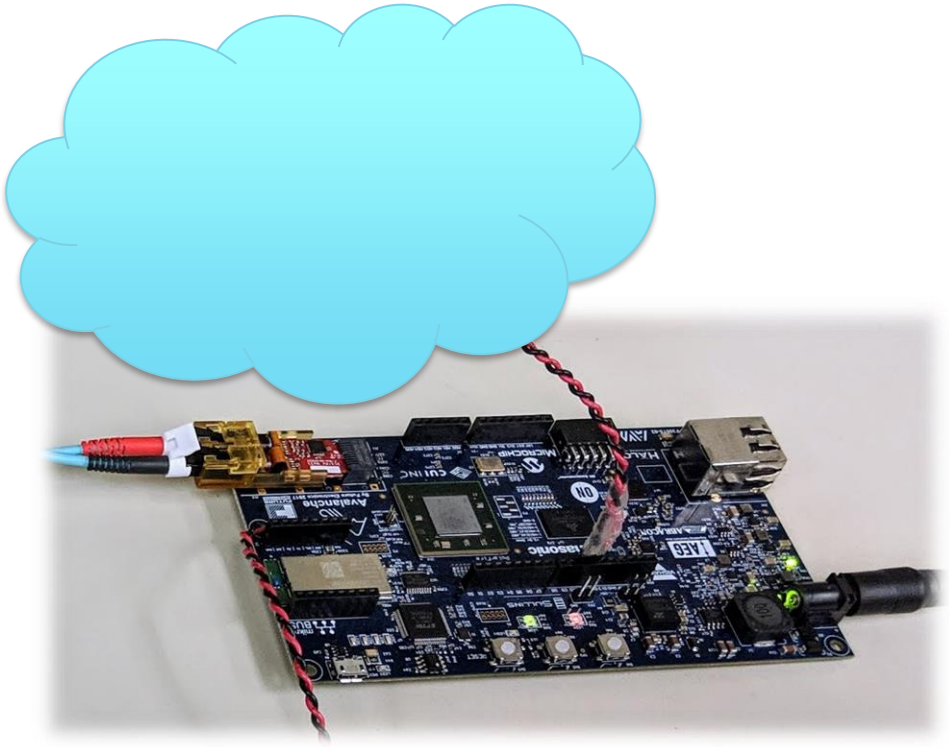


# Value of Portable Timing Verification Unit

- In the real experiment, we would like to confirm the timestamping clocks of two ROCs or two detectors have the same T=0 moment.
- Need a particle or calibration pulse to traverse both detector component and check timestamp.
- With a mobile trusted source for T=0, the two detector components to not have to be physically close!
  - A : B and A : C .. Then B : C

# Design of Portable Timing Verification Unit

- Could be … Prototype ROC in Polar Fire dev kit formfactor
  - Core ROC firmware already developed for form factor
  - Need to define detector and mechanics



# Next Steps

- Choose detector form-factor
  - Consider bias voltage and surface area
- Develop user friendly mechanical package
- In parallel, verify FPGA firmware and software loopback and timestamping
  - Becomes golden standard for Mu2e timestamping during experiment operation!
- Portable timing verification unit could be great summer project for mechanical and electrical engineering students to work on together.

# Backup

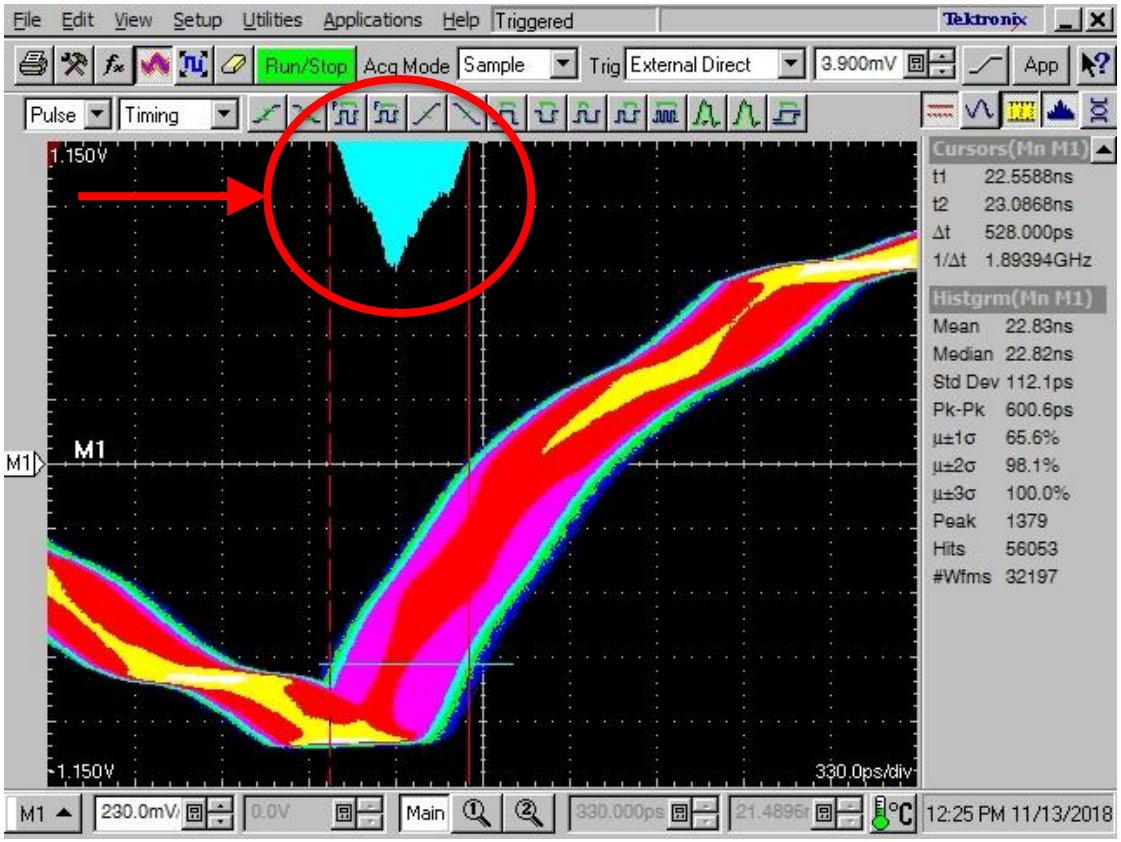
# Loopback measurement: essential detail

Wait a second… Since the loopback measures the average of the distribution below (StdDev ~ 112.1ps), and the CFO measures time in 5 ns bins (200 MHz), why don't we get exactly the same bin every time?

A: ROC Tx (going to DTC) is *asynchronous* with the ROC Rx (coming from DTC)

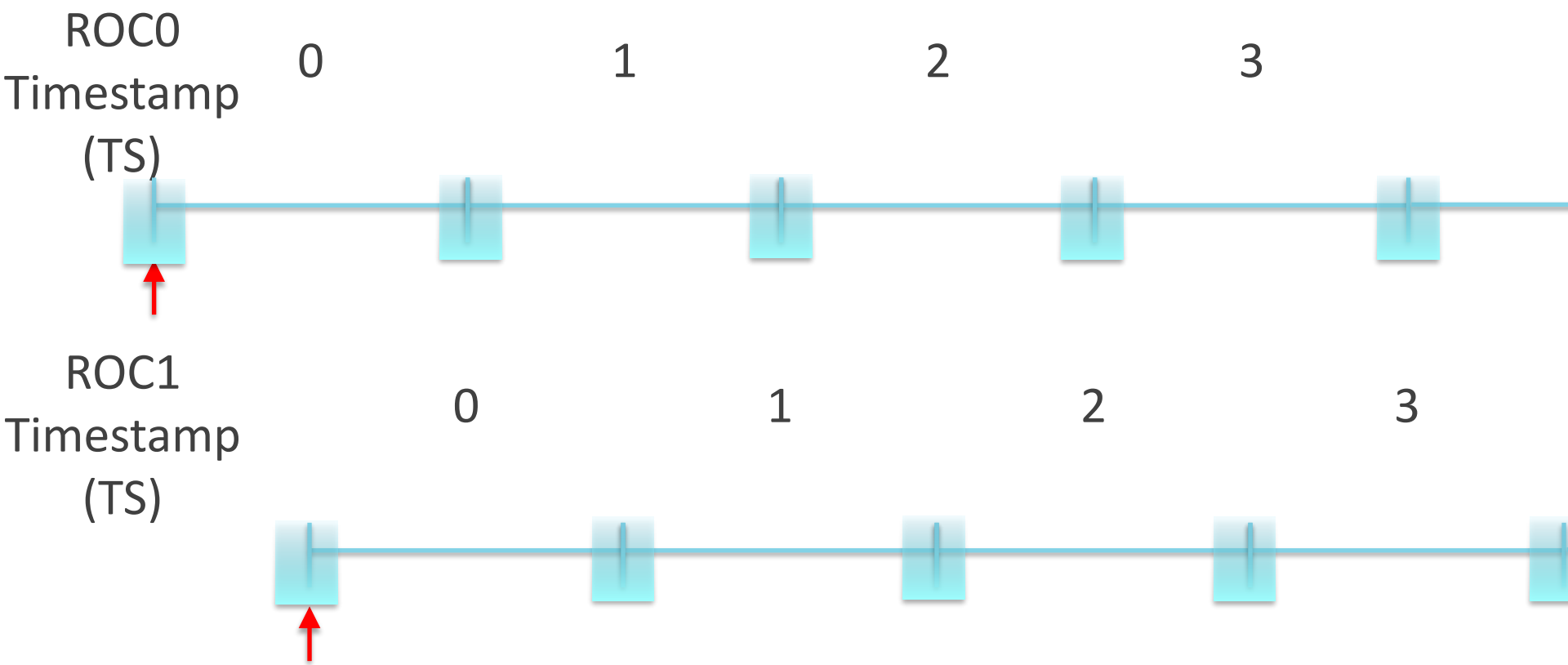
→ The (slight) frequency difference in Tx vs. Rx effectively scans across Rx clock bins

→ Maybe want to explicitly make ROC Tx vs. Rx frequency different so we don't rely on slight differences between clocks



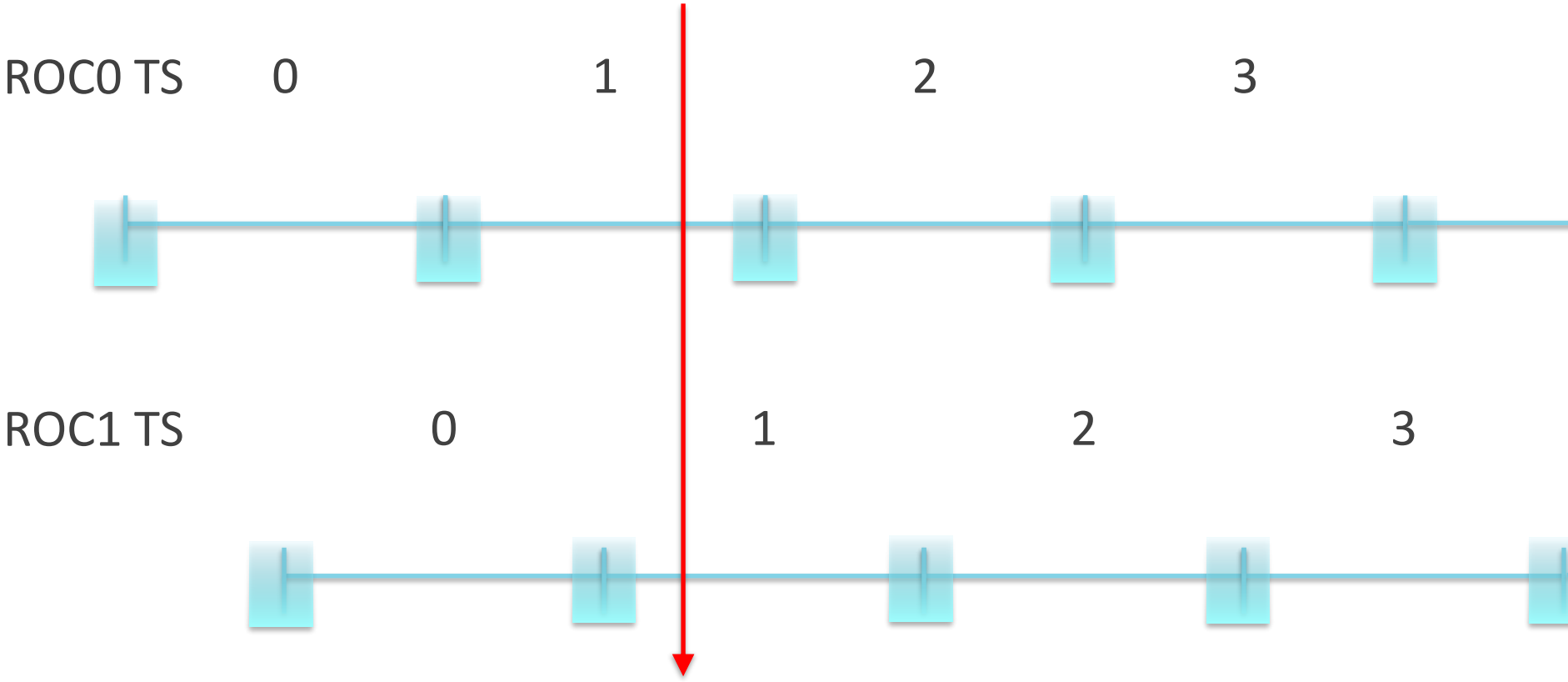

See later for jitter measurements

# Relative timestamping at two unsynchronized ROCs




Due to different fiber lengths, T=0 arrives at different times at ROC0 and ROC1
(Note: at test stand, ROC has 5 ns bins)

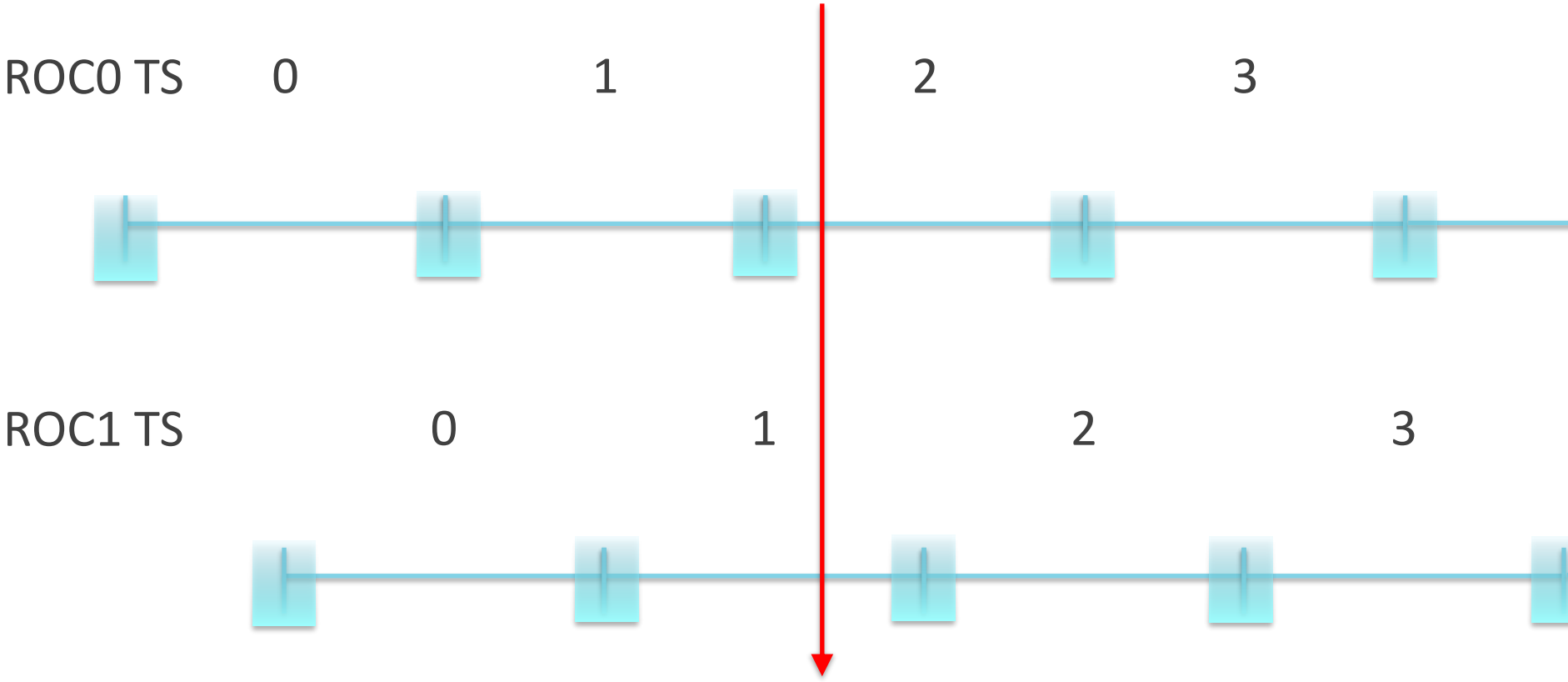
# Simultaneous data at unsynchronized ROCs




Event 1 sent at a random time, received simultaneously at ROCs:

TS(ROC0) = 1 and TS(ROC1) = 1 → TS(ROC1-ROC0) = 0

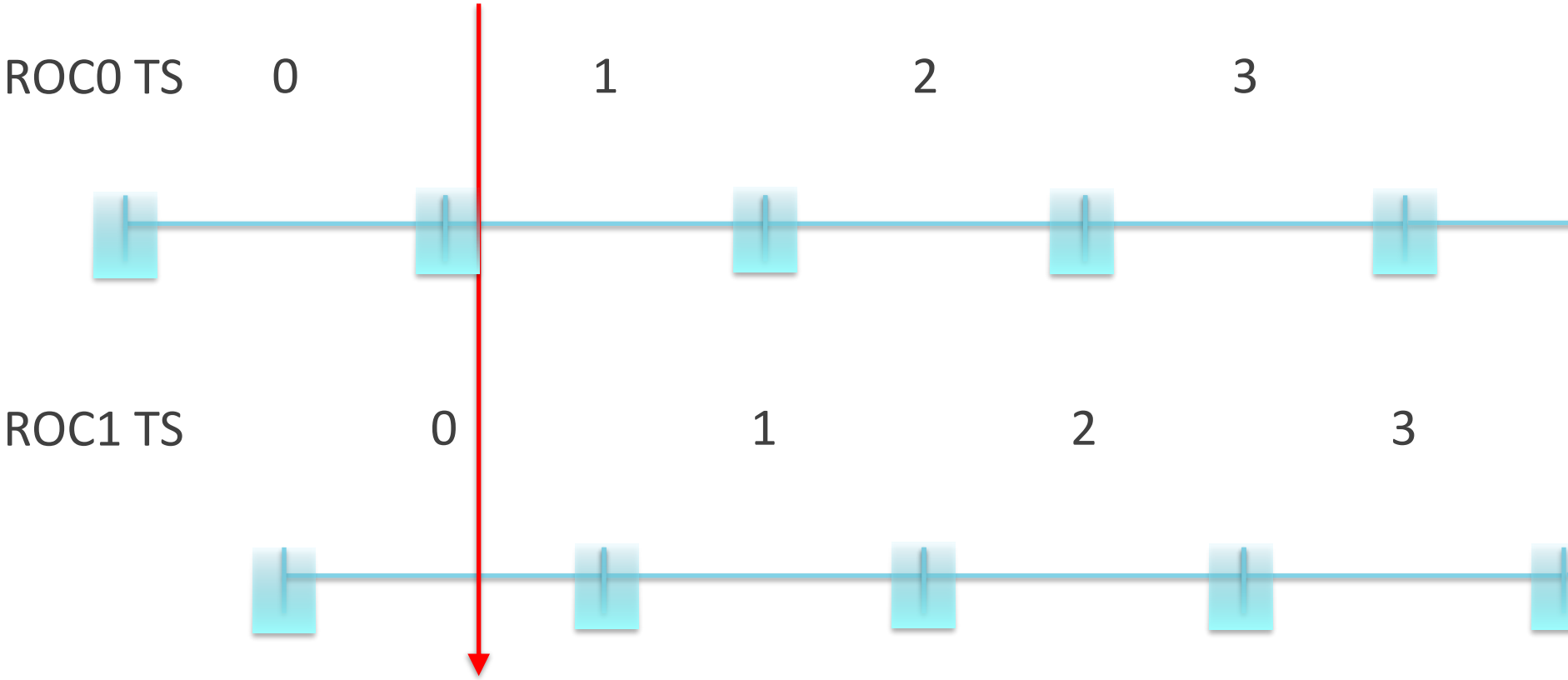
# Simultaneous data at unsynchronized ROCs




Event 2 sent at a random time, received simultaneously at ROCs:

TS(ROC0) = 2 and TS(ROC1) = 1 → TS(ROC1-ROC0) = +1

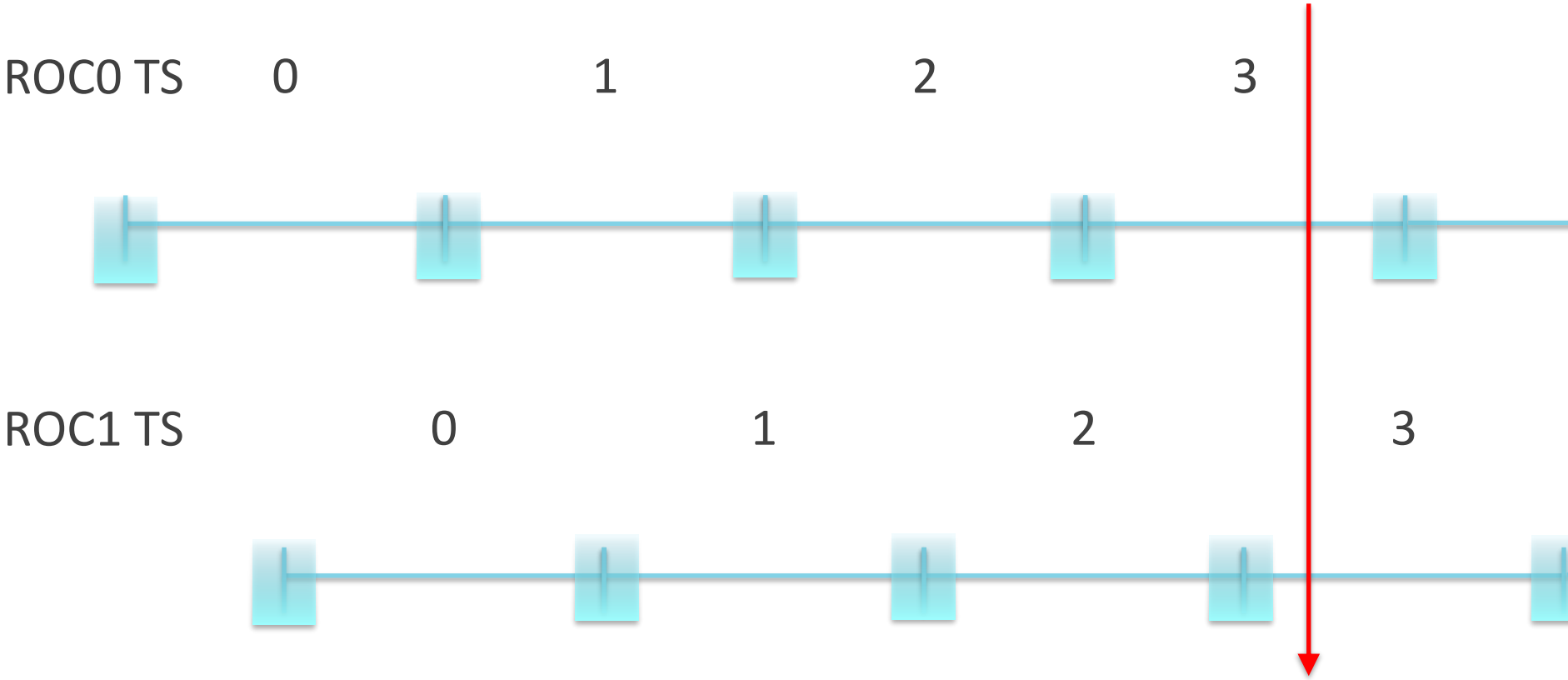
# Simultaneous data at unsynchronized ROCs




Event 3 sent at a random time, received simultaneously at ROCs:

TS(ROC0) = 1 and TS(ROC1) = 0 → TS(ROC1-ROC0) = +1

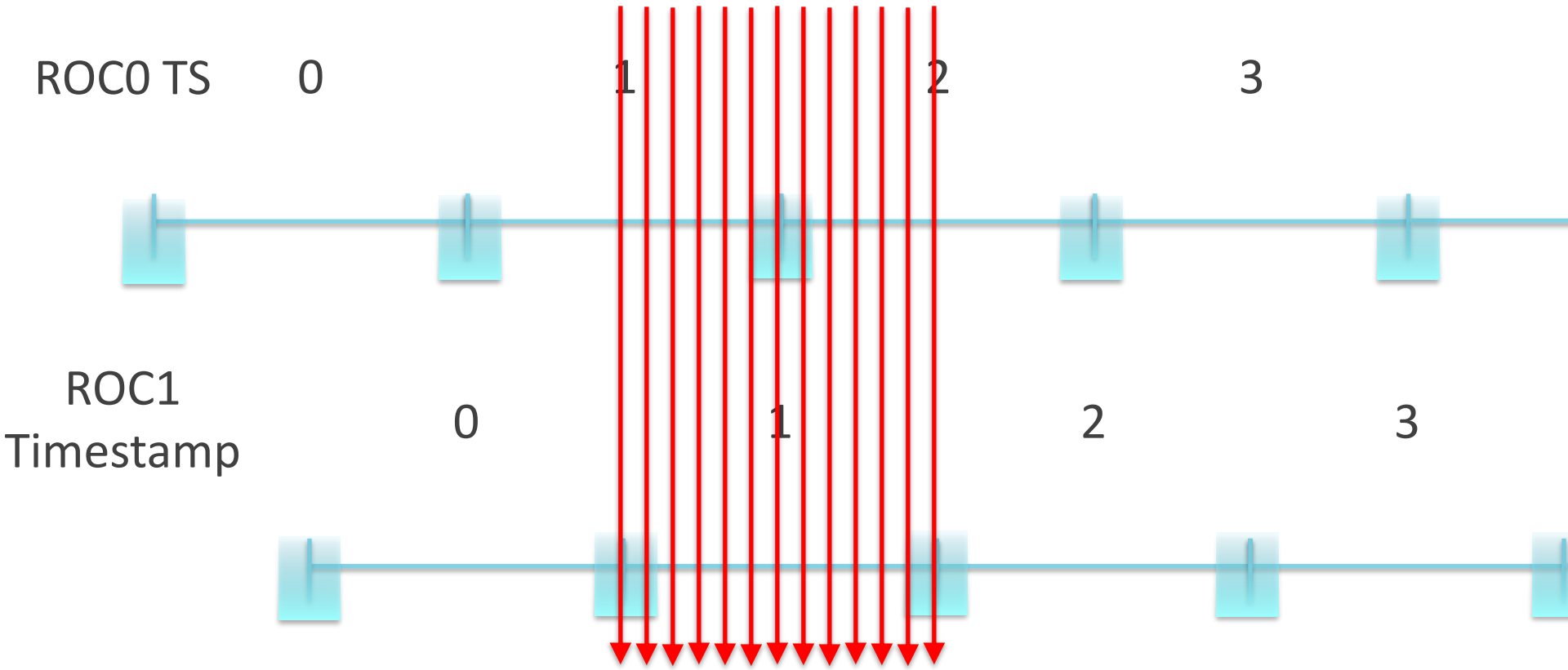
# Simultaneous data at unsynchronized ROCs




Event 4 sent at a random time, received simultaneously at ROCs:

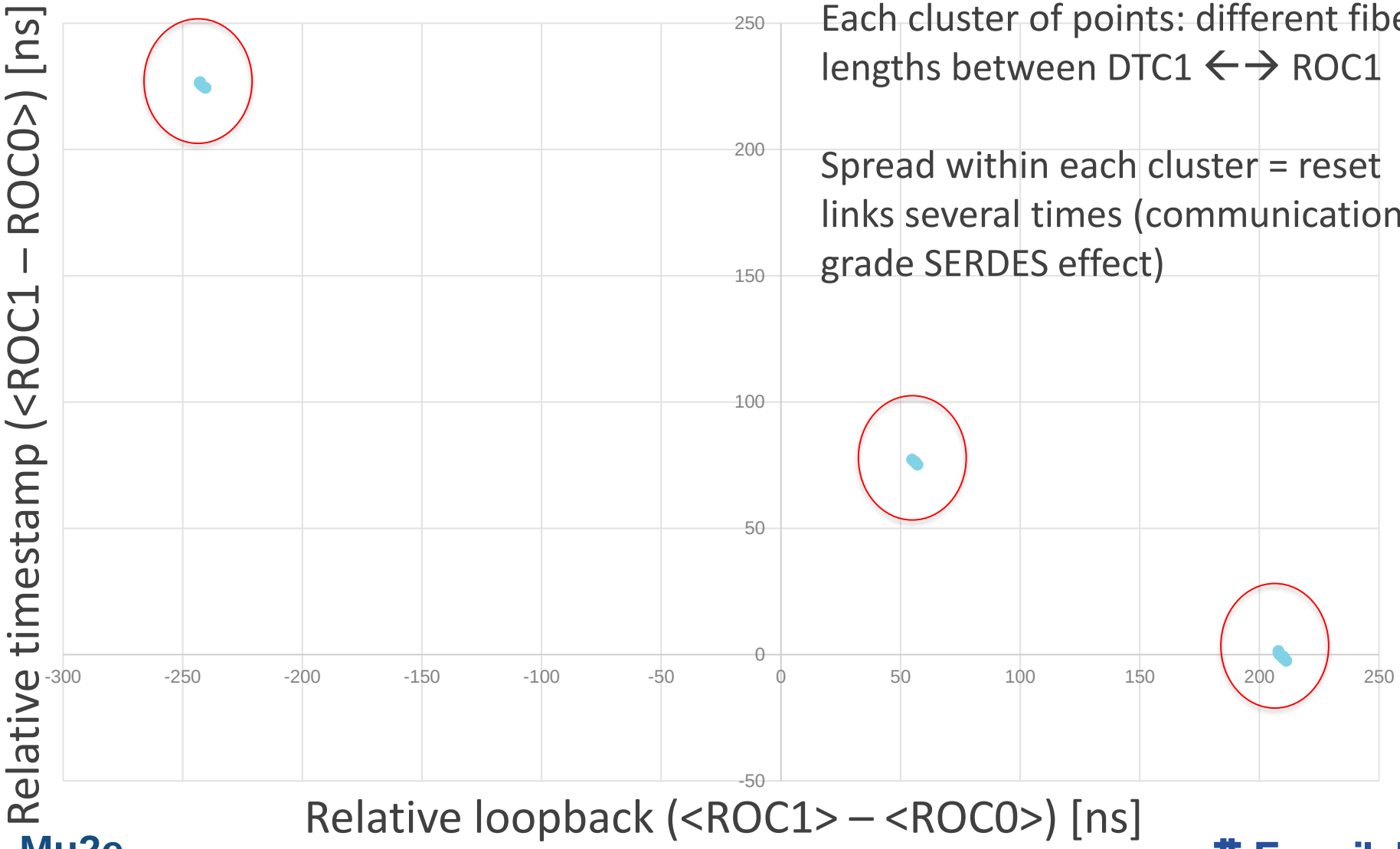
TS(ROC0) = 3 and TS(ROC1) = 3 → TS(ROC1-ROC0) = 0

# Simultaneous data at unsynchronized ROCs




Sending enough random data and histogramming time difference: effectively scans ROC0 timestamp relative to ROC1

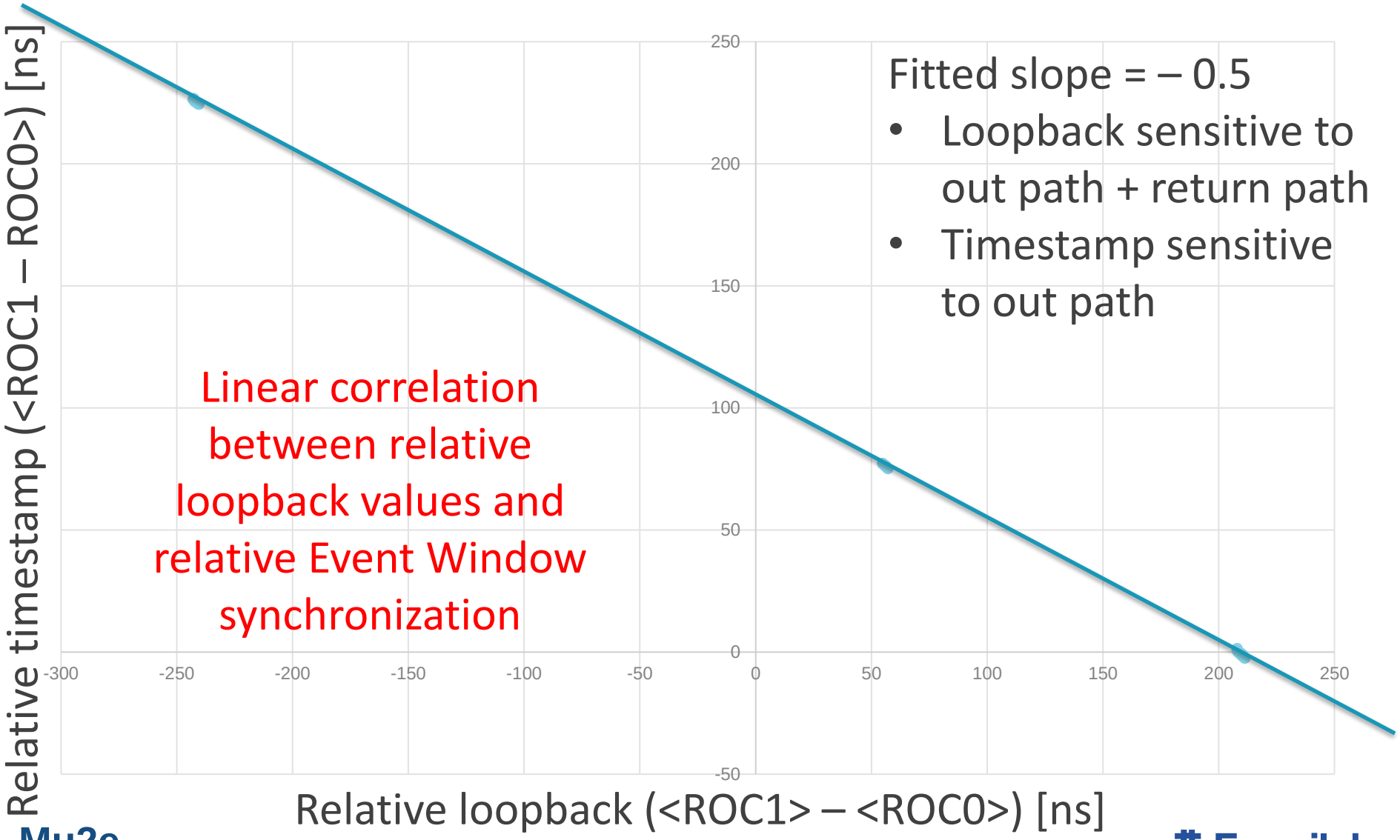
# Relative timestamp vs relative loopback




Each cluster of points: different fiber lengths between DTC1 ←→ ROC1

Spread within each cluster = reset links several times (communication-grade SERDES effect)

# Linear fit

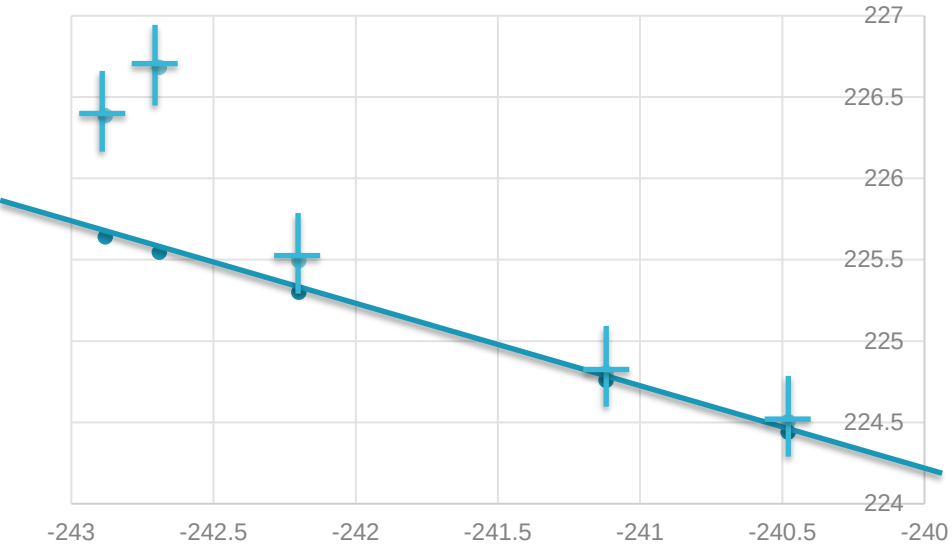


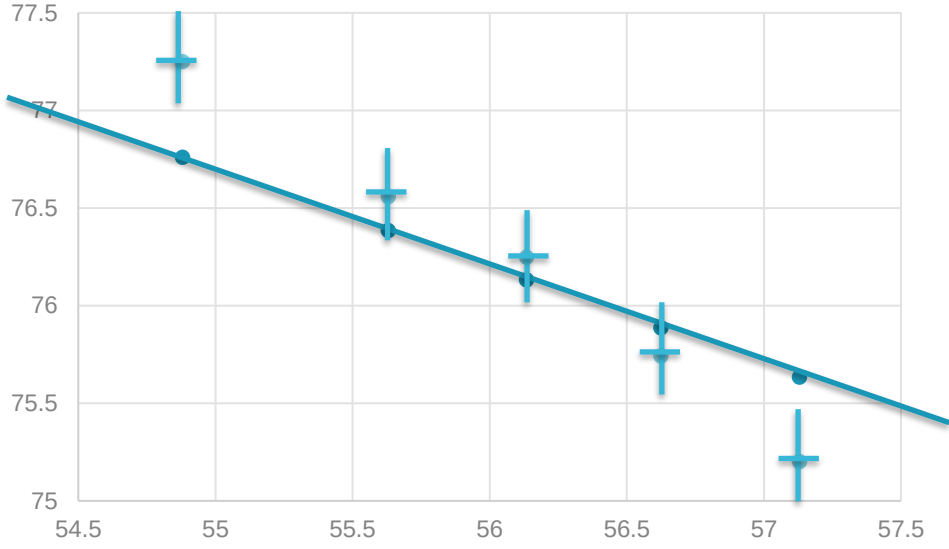

Fitted slope = – 0.5

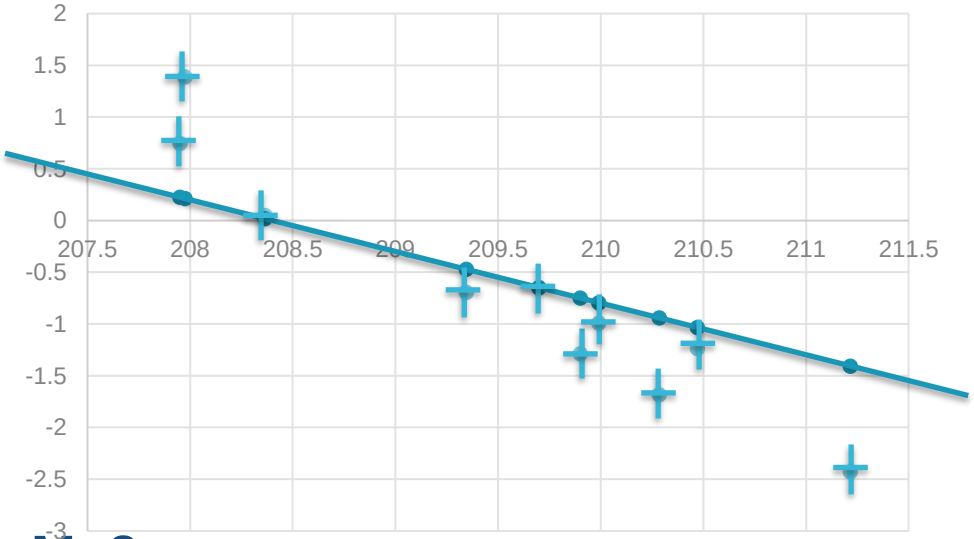
- Loopback sensitive to out path + return path
- Timestamp sensitive to out path

Linear correlation between relative loopback values and relative Event Window synchronization

# Detailed look within each cluster



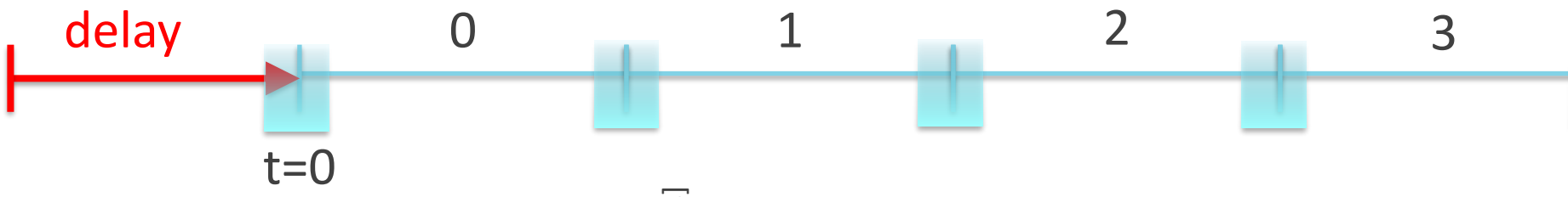




- Blue = data w/ uncertainties
  - 1000 loopbacks (X-axis)
  - 100 timestamps (Y-axis)
- Orange = fit to full data set
  - Slope = – 0.5
  - Intercept = 104.2
- Structure w/in each cluster?

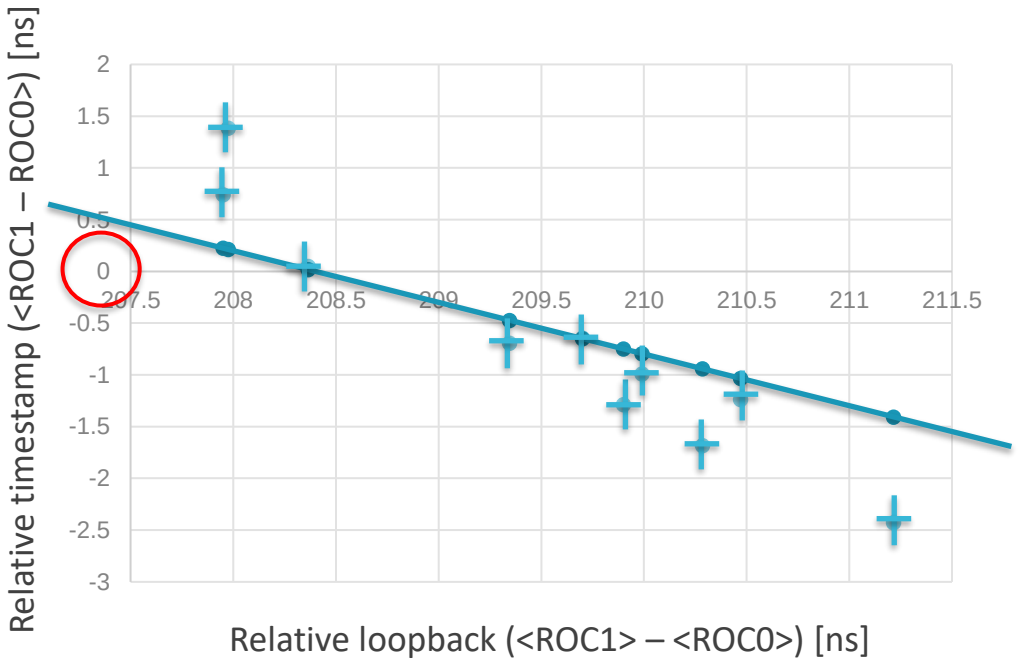
# Plan: proceed with synchronization plan

Use loopback to determine coarse (5ns) + fine (250ps) delays for each ROC to synchronize Event Windows (t=0) to 250 ps



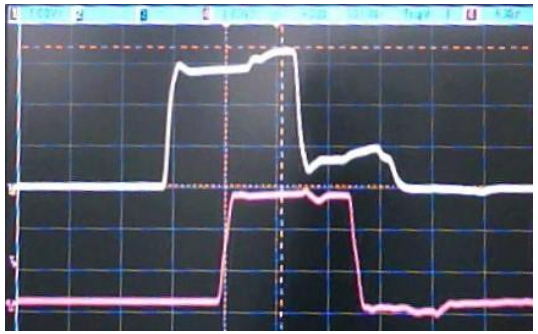

Note: coarse delay applied at ROC for this fiber length

(that's why this relative timestamp intercepts 0…)

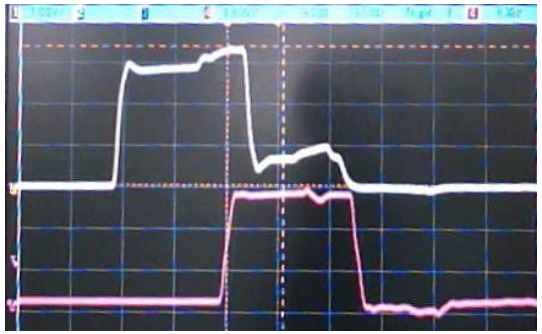


# January 2021: Tracker Event Window Marker Sync

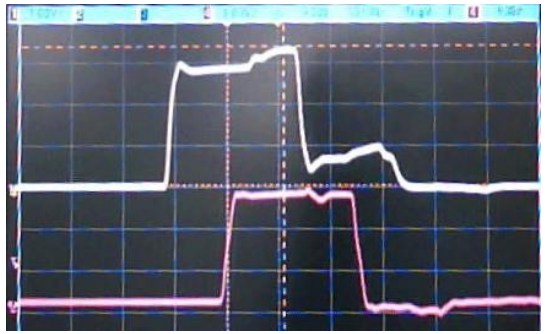
- January 27, 2021: Tracker & TDAQ demonstrated Event Window Marker sync between Polar Fire Avalanche development card ROC and Tracker DRAC.
  - Event window marker fixed relative phase relationship survived all permutations of power down and reset of DTC/DevCard/DRAC!



1. Startup

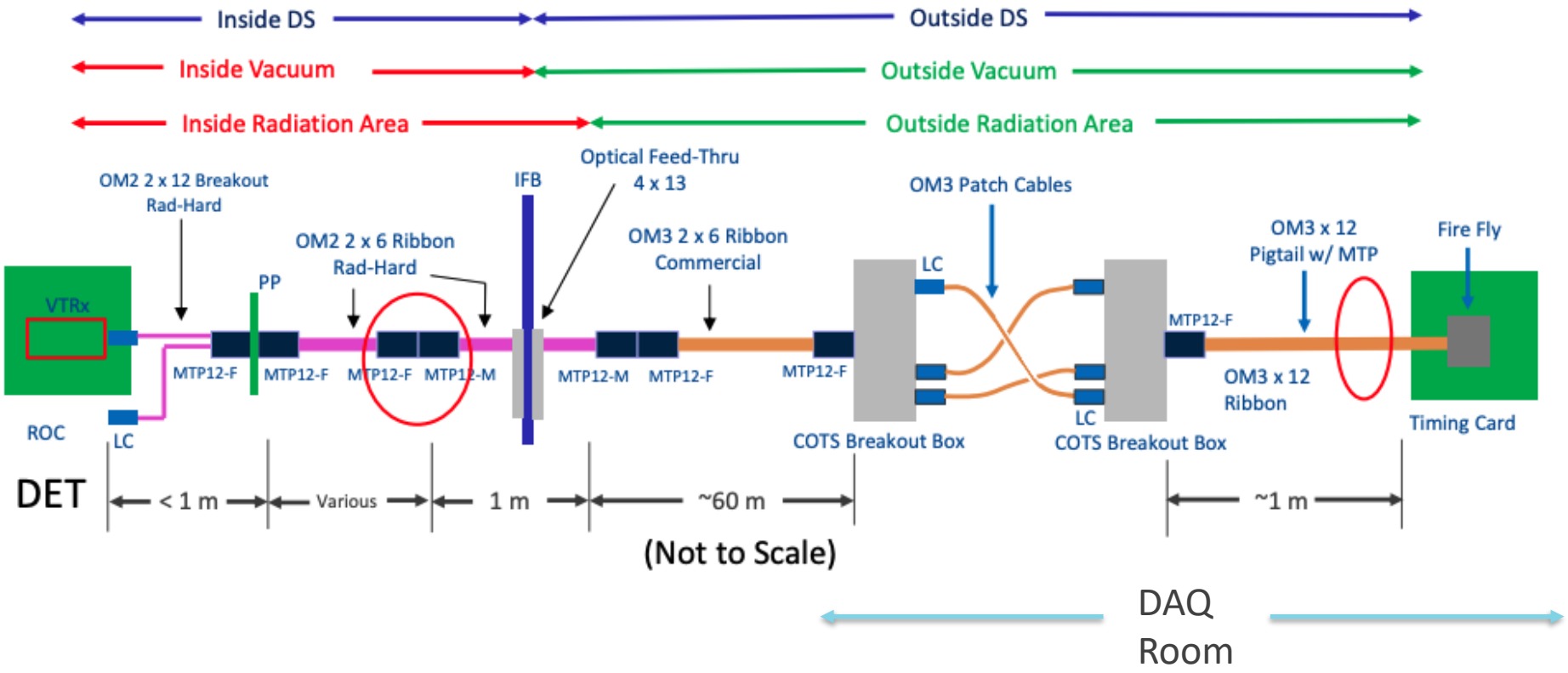


2. Reset DTC (random phase)



3. Send Link Align

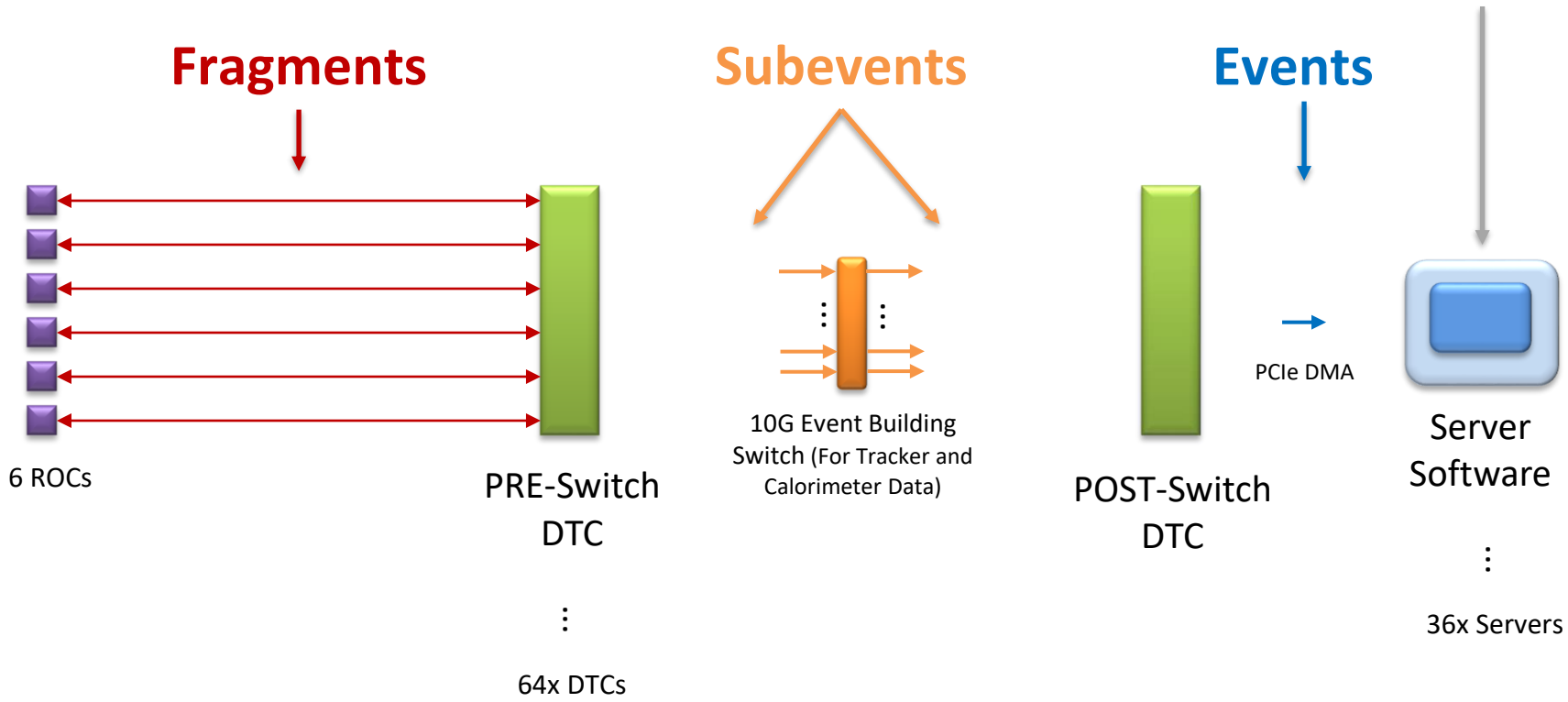
# End-to-End Schematic of TDAQ Fiber Links




- OM2 Rad-Hard fiber expected to be Draka Elite Super Rad Hard fiber
- OM3 fiber expected to be Corning ClearCurve fiber
- Expected total path <100m

# Terminology Topology

Ignoring…
artdaq::Fragment
art::Event
art::EventID (run, subrun, event)



# Terminology

- **Fragment**
  - Complete dataset at one ROC for an event window consisting of a **Data Header** packet and subsequent **Data Payload** packets as specified in docdb 4914. Event windows at ROCs are synchronized using Heartbeat Packets and Markers as described in docdb 18222.
- **Subevent**
  - Complete dataset at one DTC pre-switch (i.e., before the event building switch) consisting of one **Fragment** from each ROC connected to the DTC (up to 6 ROCs allowed).
- **Event**
  - Complete dataset at one DTC post-switch (i.e., after the event building switch) consisting of one **Subevent** from each DTC in the partition.