
Streaming Readout Development for CODA @ JLab

23rd IEEE NPSS Real Time Conference
August 1-5, 2022

David Abbott

Ben Raydo

FEDAQ Group

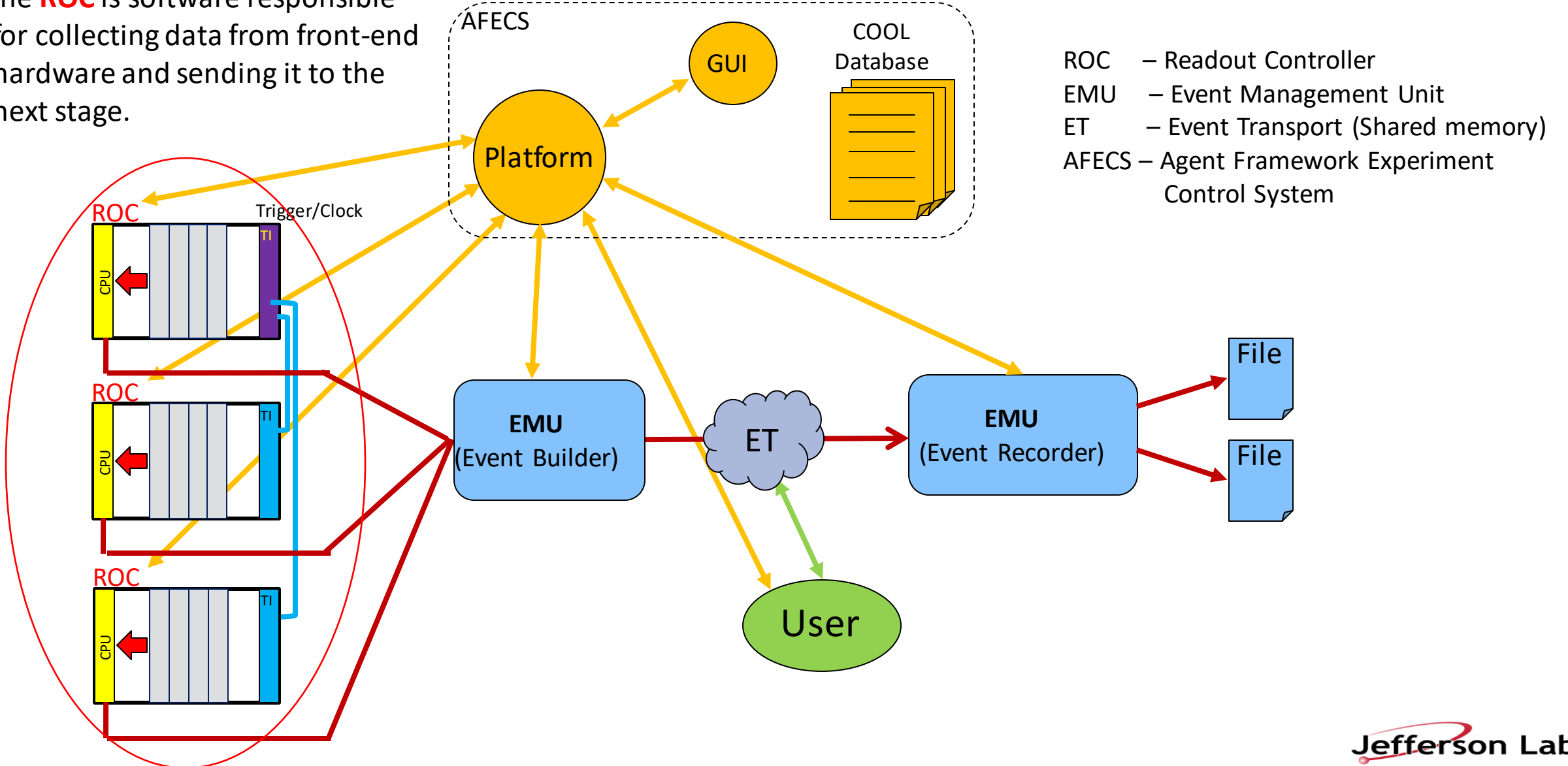
Jefferson Lab – Physics Division

Data Acquisition at Jefferson Lab

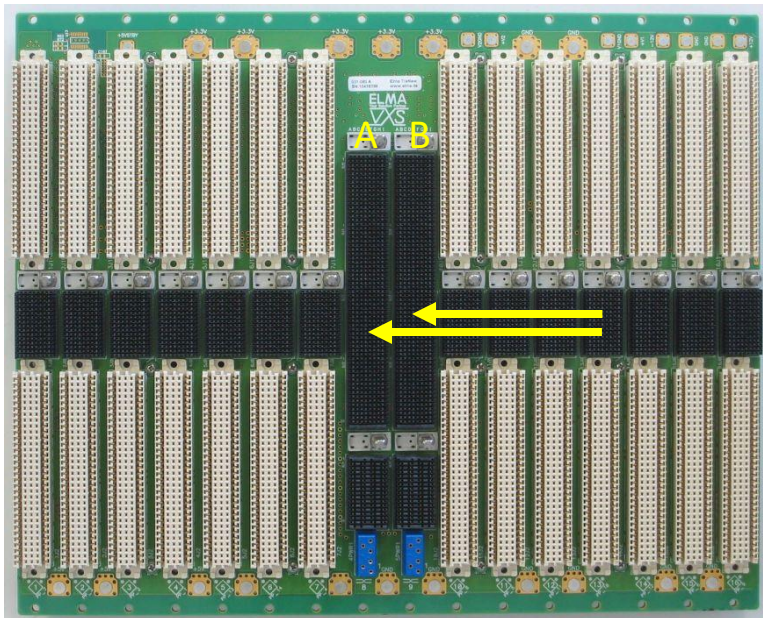
- At JLab we have 4 Experimental Halls, all running with different detectors - and physics priorities.
 - Of course, all are having increased demands for the DAQ.
- Experiments are increasingly reliant custom electronics to interface detectors and digitize signals.
 - ASICs and FPGAs are becoming the norm (and the future) for the front-end.
 - But older hardware is still relevant and useful (particularly for starving budgets)
- Our goal is to support both the traditional **Triggered** model along with the **Streaming** model within one integrated DAQ framework.
 - Leverage existing hardware to implement streaming
 - Add support for new electronics
 - Try to make it as seamless and user friendly as possible

The CODA Data Acquisition Toolkit

The **ROC** is software responsible for collecting data from front-end hardware and sending it to the next stage.



VXS Standard (VITA 41)



- JLab standardized on this technology for the 12GeV Upgrade
 - Originally used for the [L1 trigger](#) data path
- [Dual Star](#) – switched serial backplane (along with original VME)
- Up to 20Gb (4 lanes) from each Payload to the 2 Switch slots (A, B)
- Up to 18 Payload slots are available
- Easy distribution of Trigger, Sync and low jitter clock to all modules in the crate.



VXS Trigger Processor (VTP)

- Relieve the ROC of all the “[Readout](#)” tasks and implement them in the FPGAs.
- [Triggered](#) or [Streaming](#) readout from ALL payload modules in parallel
- In general, the payload modules should have some intelligence/programmability and serial link capability (e.g. FPGA-based).
- The [Software ROC](#) now is primarily responsible only for Configure, Control and Monitoring the **VTP-Based DAQ**.

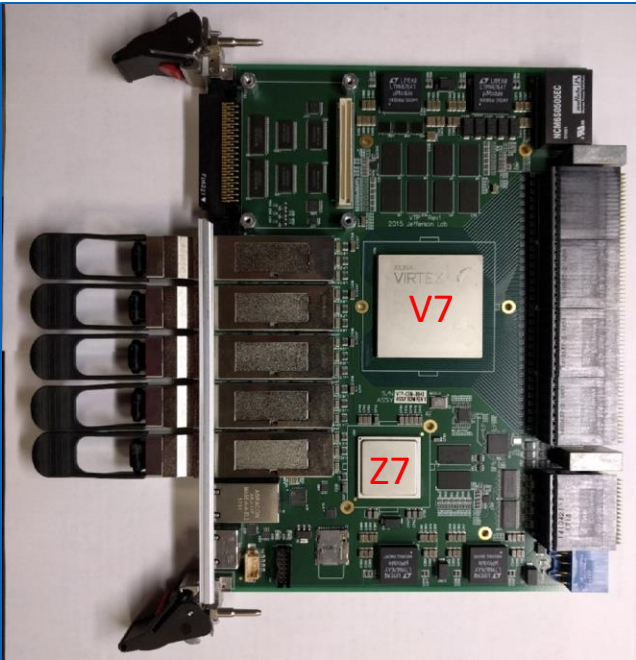
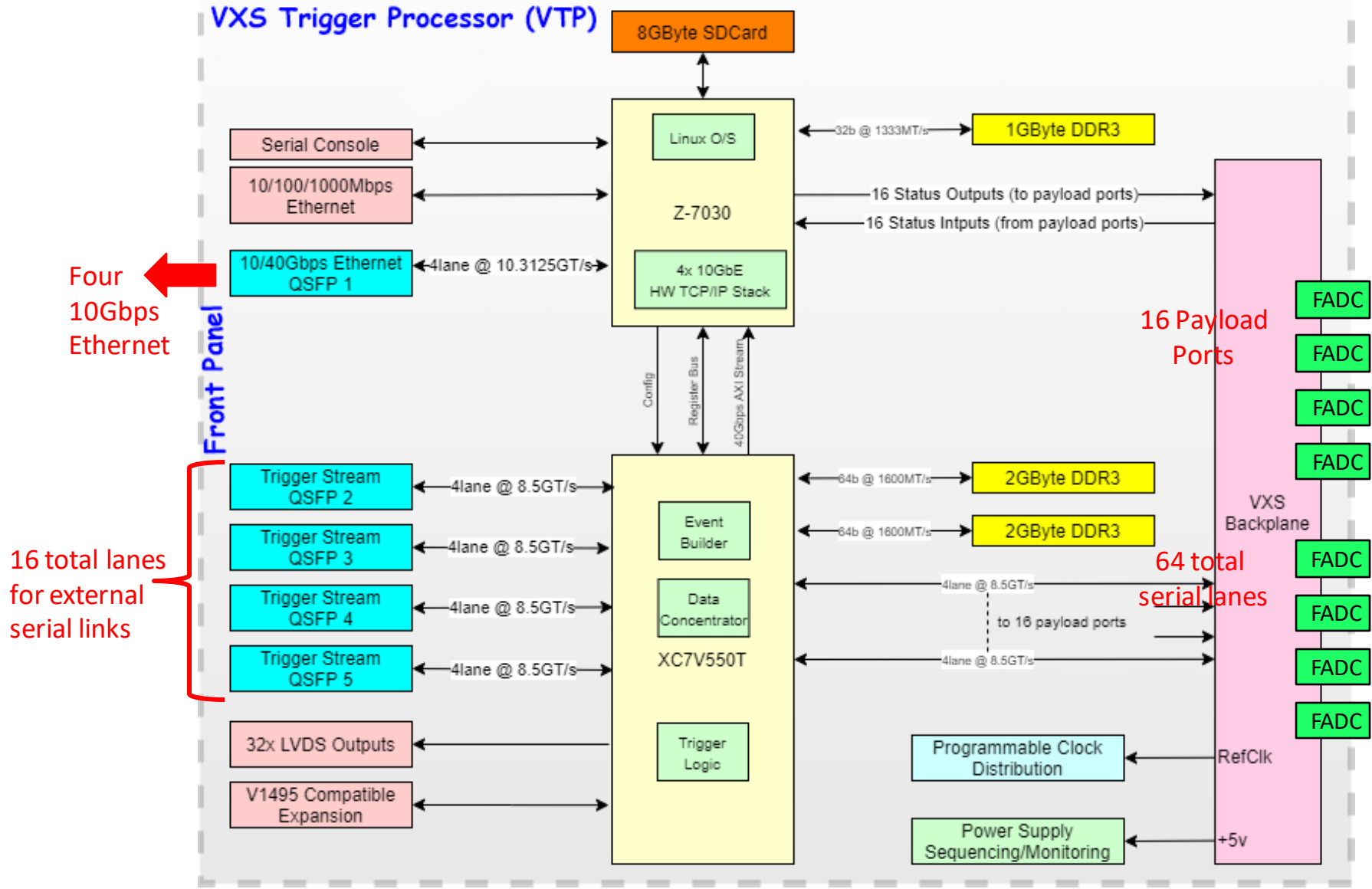
JLAB – VTP Board

Linux OS on the Zync-7030 SoC
 (2-core ARM 7L, 1GB DDR3)
 10/40Gbps Ethernet option
 (runs the CODA ROC)

Xilinx Virtex 7 FPGA

Serial Lanes from both the VXS
 backplane and the Front panel
 4GB DDR3 RAM

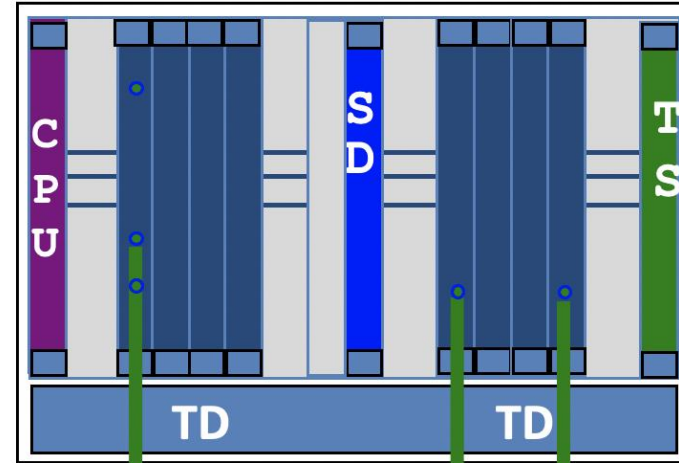
VXS Trigger Processor (VTP)



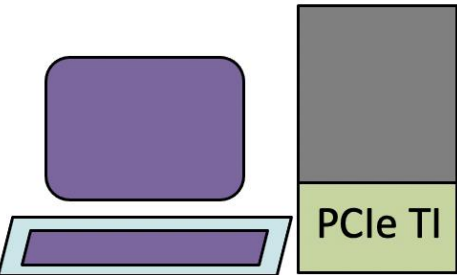
JLAB Clock and Trigger Distribution System

- TS** – Trigger Supervisor (VME/VXS)
- SD** – Signal Distribution Board (VXS)
- TD** – Trigger Distribution (VME/VXS)
- TI** – Trigger Interface (comes in several flavors)

Trigger Distribution Crate



For large DAQ systems with many front-ends (ROCs)

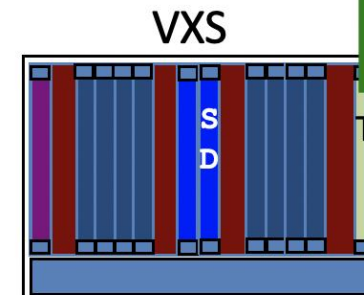
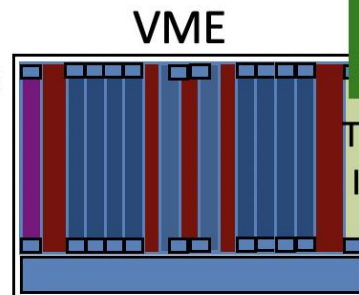


CTS

(TRIGGER, CLOCK, SYNC) (BUSY)



Front-End ROCs:
(up to 127)



The TI board can be used as a TS for small systems (up to 9 front-ends)

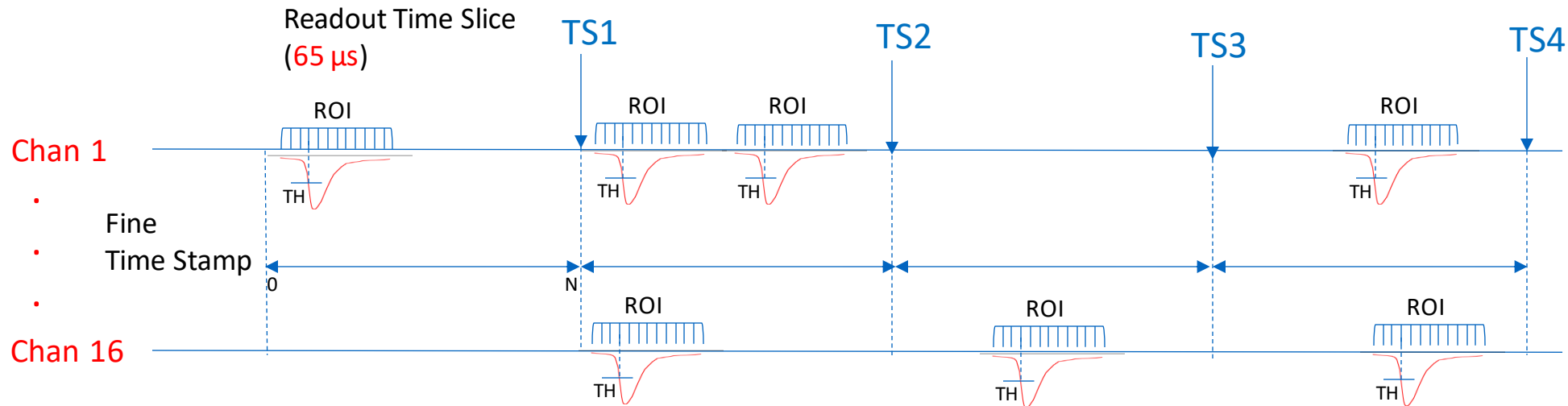
JLAB FADC – Streaming mode

A 250 MHz FADC generates a 12 bit sample every 4ns. That's 3 Gb/s for one channel. 16 channels is 48 Gb/s.

Currently, we identify a threshold crossing (hit) and integrate charge over a ROI and send only a **sum** and **timestamp** for each hit.

Available bandwidth will allow for 1 hit every 32ns from all channels.

A data frame (Time Slice) for all available hits is generated in the VTP every **65 μ s**



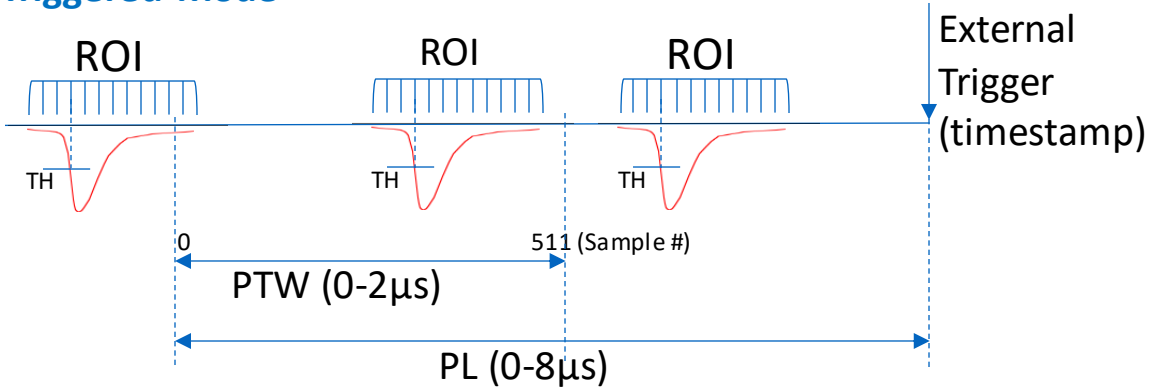
The next revision to the firmware will have an option for full ROI wave forms to be streamed, but this will allow possible dropped hits due to bandwidth limitations

The FADC can still simultaneously operate in triggered mode with an 8 μ s pipeline and 2 μ s readout window.



FADCs - Triggered vs Streaming

Triggered Mode

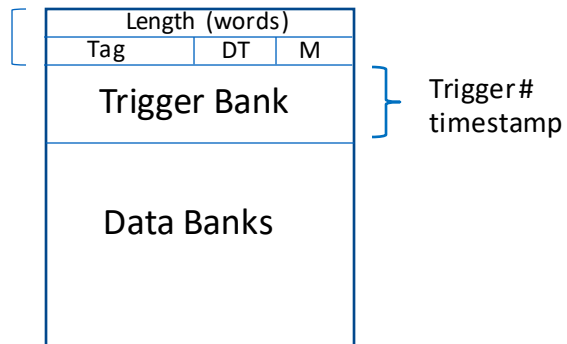


PL: Programmed Lookback
PTW: Time window

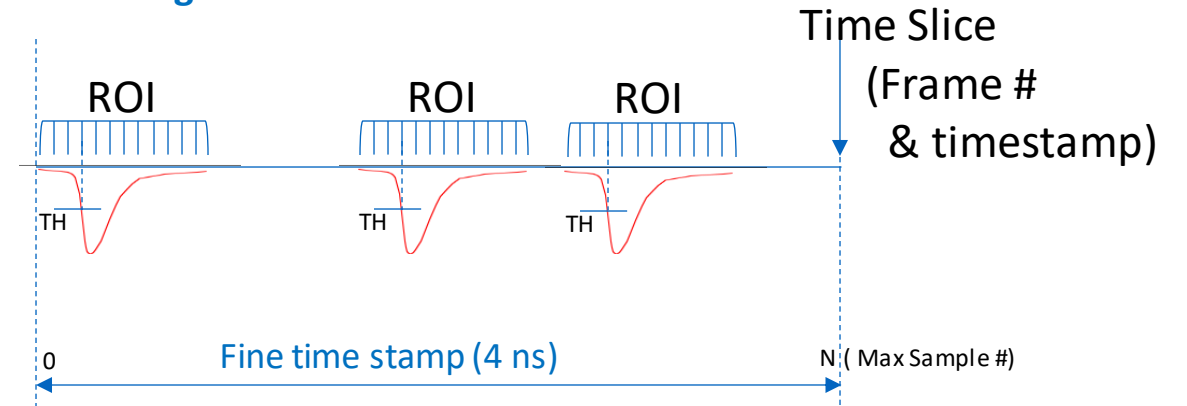
Data we get on a trigger:

- FADC waveform values for the ROI
- Threshold Sample # (hit time)
- Trigger absolute time stamp

ROC Data Format



Streaming Mode

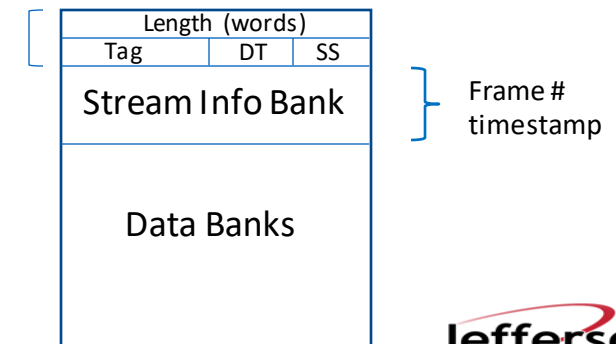


1 Frame = N Clocks (up to 16bits, currently 65536 ns)

Data we get for a Frame:

- Pedestal subtracted sums over an ROI for every hit over threshold
- Threshold sample # fine time stamp for each hit
- Frame # and absolute time stamp for the frame

ROC Data Format

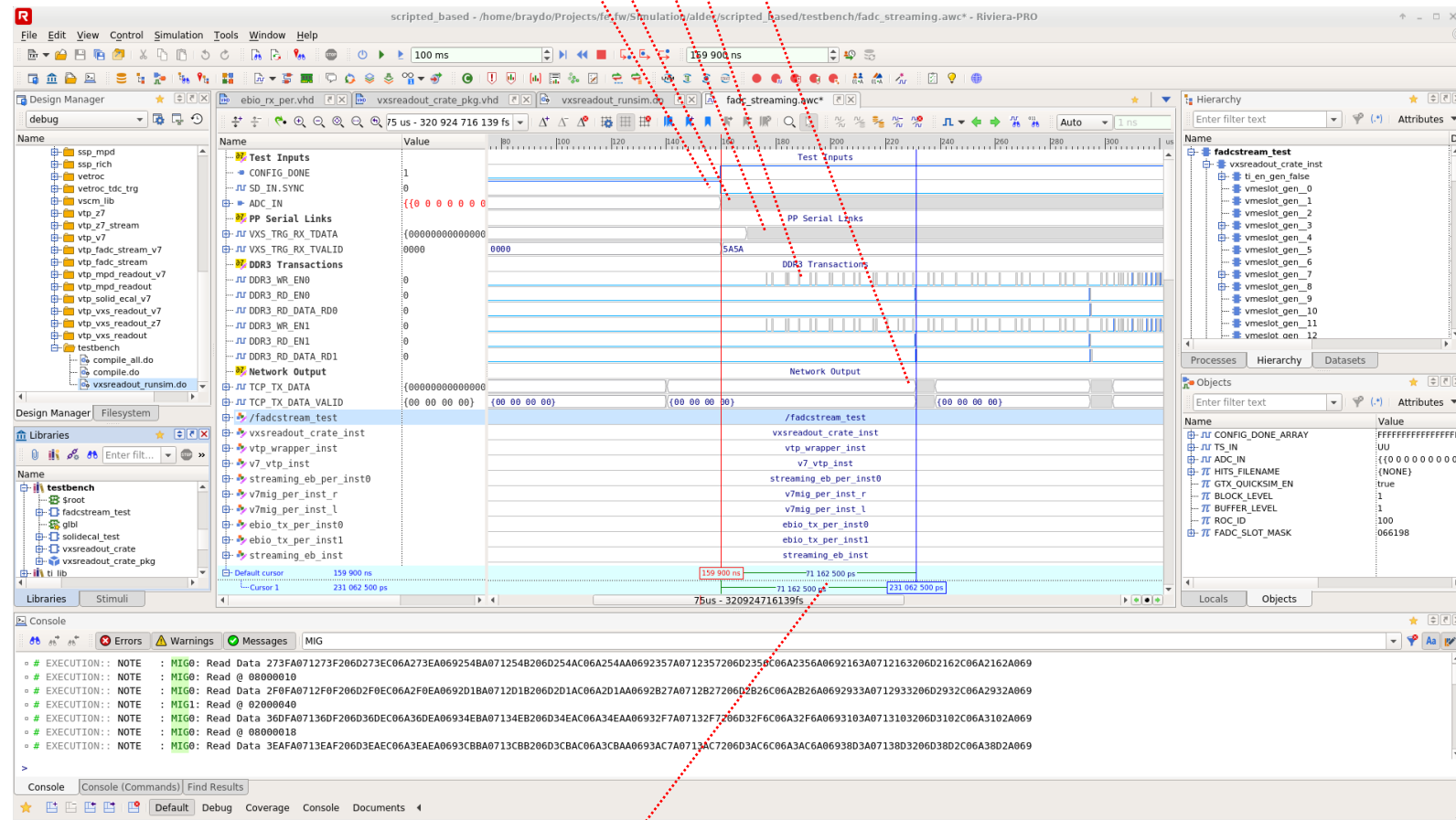


Front-end Crate Simulation

Aldec Riviera used for VHDL, SystemVerilog, C/C++

- Read & write files & network
- Library access (ROOT, EVIO, etc)
- Detailed waveform viewing for low-level debugging
- Accurate latency information
- Have been able to reproduce many failure cases from real system tests
- Simulation isn't fast: few minutes to simulate a $65\mu\text{s}$ for a single crate of 256 FADC channels, but often only takes a few frames to get needed information

1. SYNC released (starts data flow)
2. Analog pulses to FADC250
3. Serialized pulses to VTP
4. Writes/Reads to/from DDR buffer
5. Frame builder writes TCP stream



71 μs latency: analog pulse to network interface

Event Viewer/Processing Tools

EVIO compliant files come from real DAQ and simulation output. Allows standard Jlab DAQ tools and event builders to use these sources from the new streaming data system.

E.g. jevioldmp is checking 'event block' format for correctness:

The screenshot displays the Event Viewer/Processing Tools interface. The main window title is "vtp2_streamv3_1TCP_1000.dat bytes (on indra-s1)". The file path is "/scratch/abbott/d/ata/vtp2_streamv3_1TCP_1000.dat".

Search By:

- Word Value
- Word Position
- Page Scrolling
- Evio Block
- Evio Event
- Evio Fault

Search For: 0xc0da0100

Search Controls:

< >

Start Scan Stop

Done

Block Info:

Total words	13
Header words	8
Id number	-1
Event count	1
Version	4
Has dictionary	false
Is last	true

Color Key:

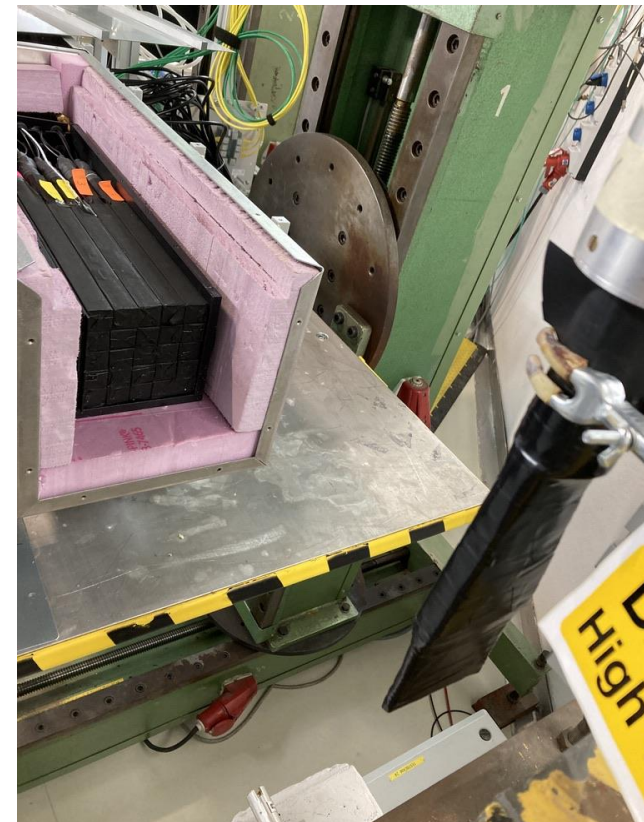
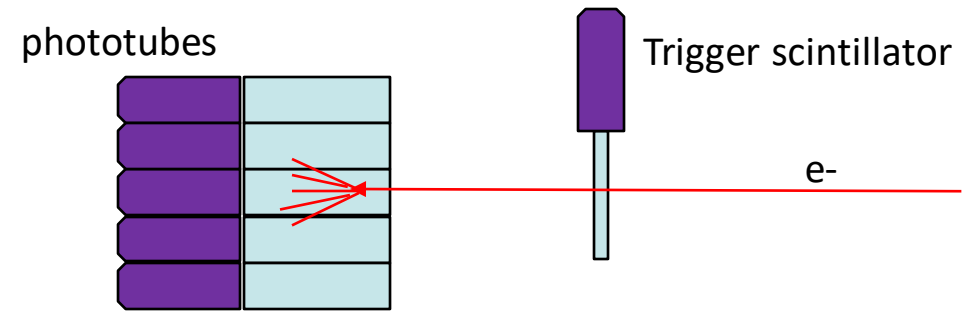
- Block normal
- Event normal
- Block with error
- Event with error
- Evio struct error
- Word value
- Current selection

Hex Dump Table:

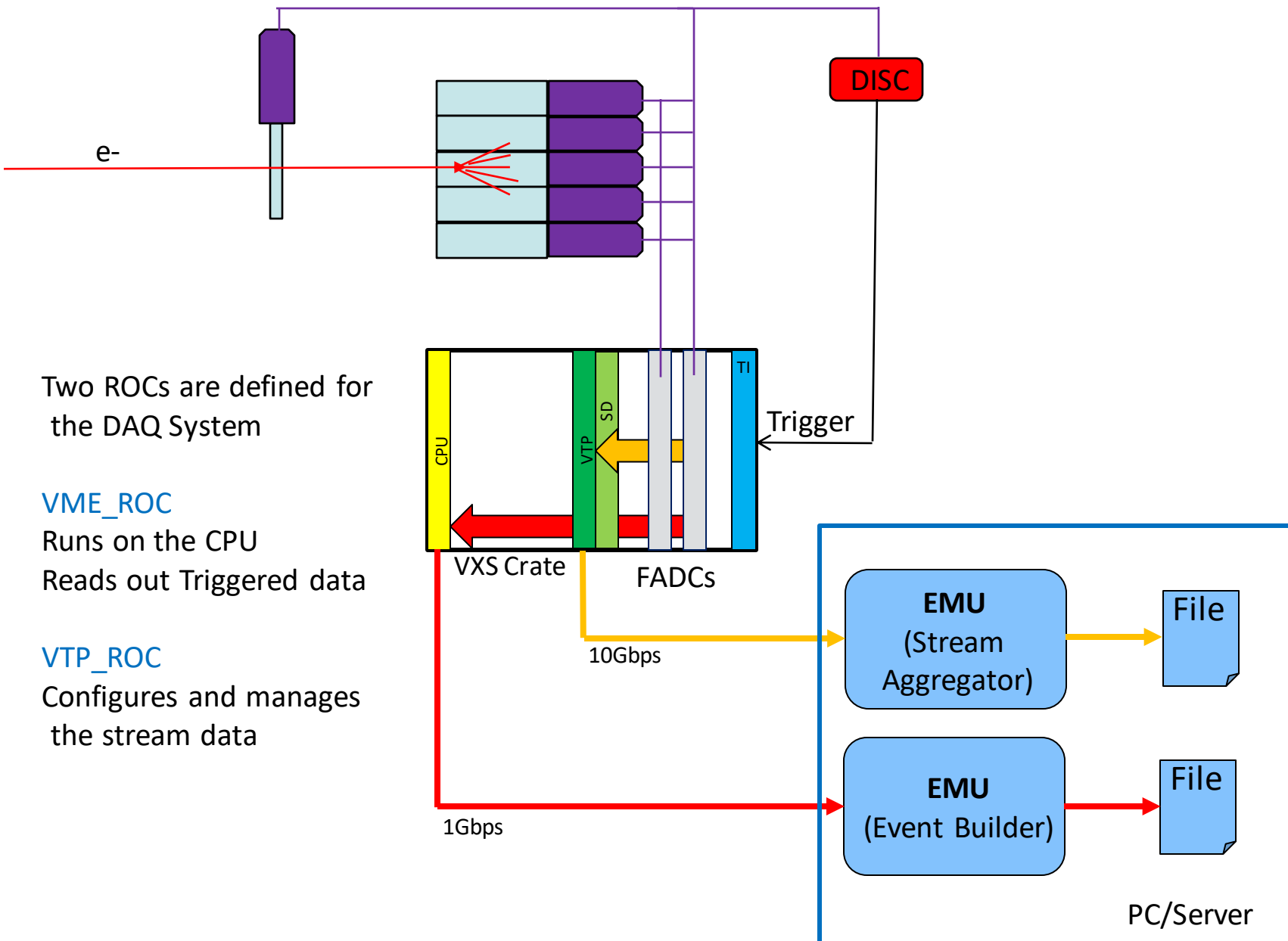
Word Position	+1	+2	+3	+4	+5	Comments
0	0x634d7367	0x20697320	0x636f6f6c	0x00000006	0x00000024	
5	0x00400800	0x00000001	0x00000001	0x00000001	0x00000034	
10	0x0000000d	0xffffffff	0x00000008	0x00000001	0x00000024	
15	0x00001604	0x00000000	0xc0da0100	0x00000004	0xffd10100	Block Header
20	0x000004b0	0x000010b7	0x00000001	0x00000001	0x00000034	
25	0x0000000d	0xffffffff	0x00000008	0x00000001	0x00000024	
30	0x00001604	0x00000000	0xc0da0100	0x00000004	0xffd20100	
35	0x000004b0	0x00000000	0x00000000	0x00000001	0x00000054	
40	0x00002415	0x00000001	0x00000008	0x00000001	0x00000024	
45	0x00002204	0x00000000	0xc0da0100	0x0000240c	0x00241011	
50	0x0000000a	0xff302011	0x31010003	0x00000000	0x00000000	
55	0x00000000	0x41050004	0x00040002	0x00070005	0x0008c000a	
60	0x000f000d	0x0000035e	0x00020f00	0x00109fff	0x00113fa1	
65	0x00119fff	0x10489ff3	0x10493f6f	0x10499fff	0x10ac9fff6	
70	0x10ad3f6d	0x10ad9fff	0x11109fff7	0x11113f70	0x11119fff	
75	0x11749ff6	0x11753f6d	0x11759fff	0x11d89fef	0x11d93f6f	
80	0x11d99fff	0x123c9fff0	0x123d3f63	0x123d9fff	0x12a09ff1	
85	0x12a13f67	0x12a19fff	0x13049ff0	0x13053f61	0x13059fff	
90	0x13689fea	0x13693f5a	0x13699fff	0x13cc9fec	0x13cd3f61	
95	0x13cd9fff	0x14309fef	0x14313f64	0x14319fff	0x14949feb	
100	0x14953f61	0x14959fff	0x14f89fed	0x14f93f5f	0x14f99fff	
105	0x155c9fe9	0x155d3f59	0x155d9fff	0x15c09fe6	0x15c13f5c	
110	0x15c19fff	0x16249fe0	0x16253f5b	0x16259fff	0x16889fea	
115	0x16893f65	0x16899fff	0x16ec9fe5	0x16ed3f5a	0x16ed9fff	
120	0x17509fdf	0x17513f53	0x17519fff	0x17b49fdf	0x17b53f57	
125	0x17b59fff	0x18189fe5	0x18193f5a	0x18199fff	0x187c9fda	
130	0x187d3f59	0x187d9fff	0x18e09fe5	0x18e13f5b	0x18e19fff	
135	0x19449fdc	0x19453f57	0x19459fff	0x19a89fda	0x19a93f60	
140	0x19a99fff	0x1a0c9fe0	0x1a0d3f54	0x1a0d9fff	0x1a709fdd	
145	0x1a713f5c	0x1a719fff	0x1ad49fe4	0x1ad53f56	0x1ad59fff	
150	0x1b389fe9	0x1b393f60	0x1b399fff	0x1b9c9fe0	0x1b9d3f62	
155	0x1b9d9fff	0x1c009fd4	0x1c013f5a	0x1c019fff	0x1c649fdc	
160	0x1c653f60	0x1c659fff	0x1cc89fdf	0x1cc93f5c	0x1cc99fff	

Beam Tests

- Recent beam tests with a calorimeter prototype at DESY (*Thanks to Doug Hasell for coordinating this opportunity*)
- 5x5 PbWO₄ Crystal Array (2 cm² face) with 2-5GeV electron test beam
- Jlab 250Mhz FADC boards
 - Triggered data are waveforms read out over VME bus.
 - Stream data are integrated sums and times of all hits over a threshold in the calorimeter regardless of the trigger status.



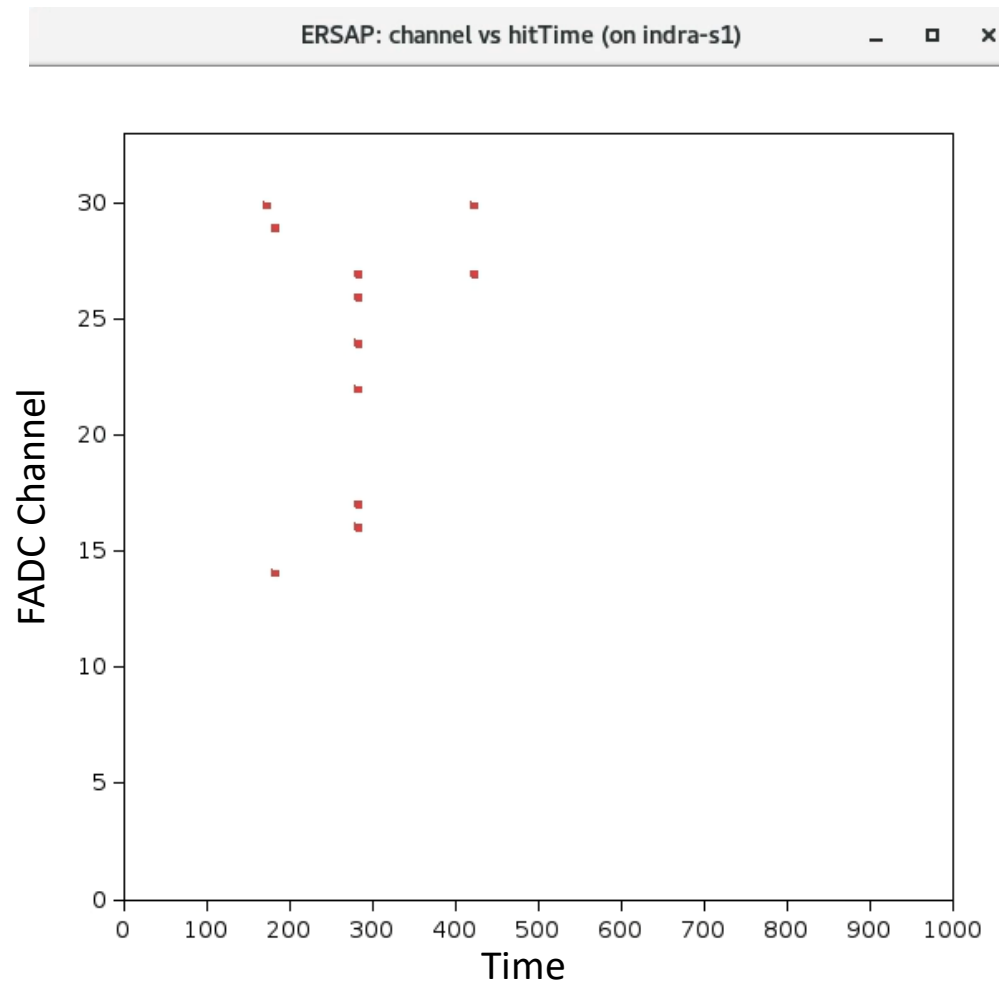
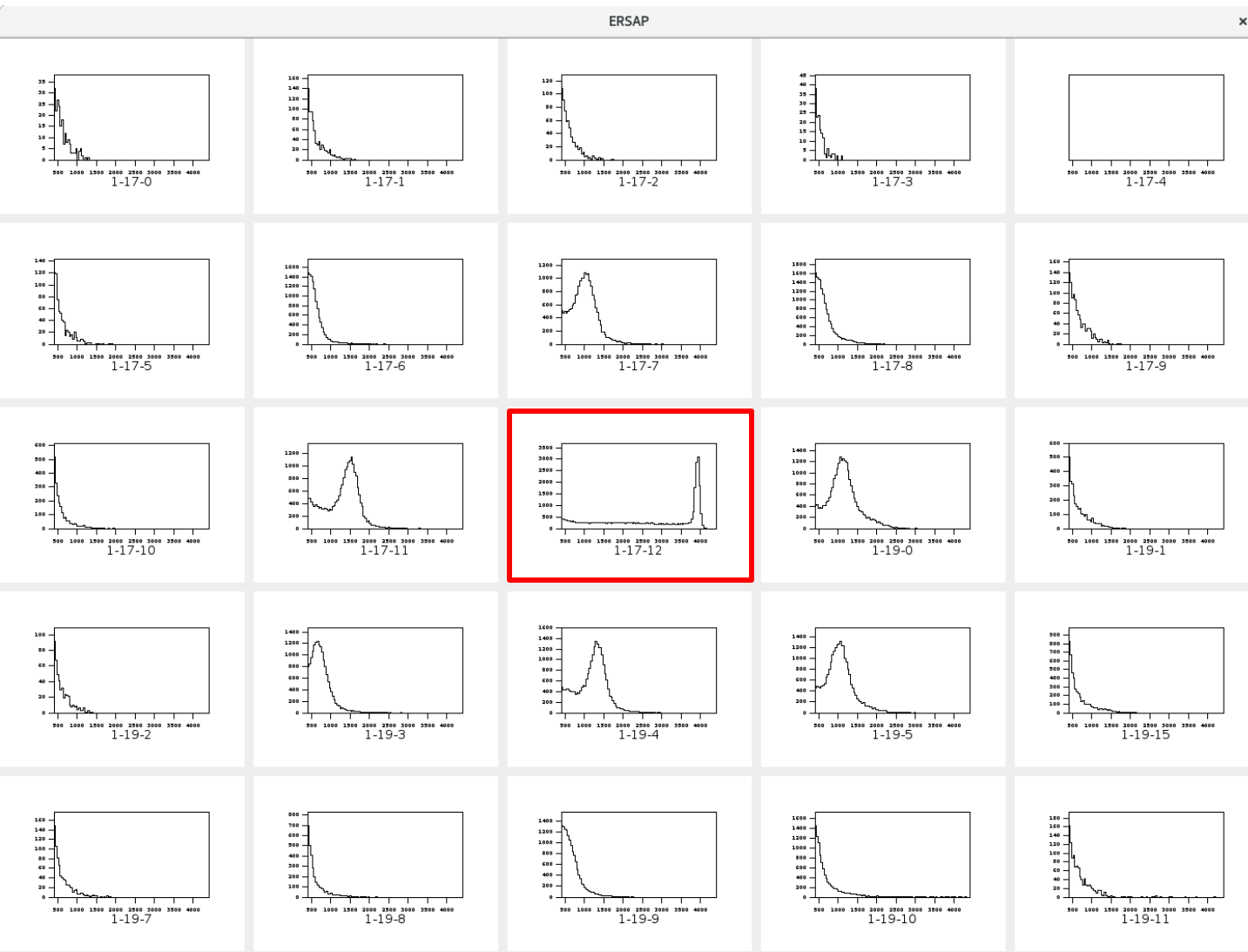
Simple Hybrid CODA System



Beam Tests

Calorimeter spectra – Streaming Data
(Electron beam centered on the central crystal)

Time Slice Frame containing a Trigger



Beam Tests cont...

- Rates were relatively low for these tests
- The electron beam varied in current depending on if they were filling the PETRA synchrotron.
- Note the respective data rates on Run Control.
 - Streaming frame and data rates are relatively constant (15.2kHz, ~1.3MB/s)
 - Triggered rates rise and fall as the electron beam comes and goes (reading out wave forms even in triggered mode generates a lot of data)

The screenshot displays the Run Control rcGui-90 interface. At the top, there are control buttons for various functions and a menu bar (control, Sessions, Configurations, Options, Expert, User, Help). The Start Time is 05/05/22 09:08:59 and the End Time is 0.

Run Parameters:
Expid: tpex 0, Session: davetest, Configuration: DESY_Stream_1
Output File: /data/test/DESY_trigger_75.evt.0
User RTV %(config): unset
User RTV %(dir): unset

Run Status:
Run Number: 75, Run State: active, Event Limit: 0
Watch Component: PEB1, Data Limit: 0
Total Events: 28,805, Time Limit (min.): 0

Data Rate Table:

Name	State	EvtRate	DataRate	IntEvtRate	IntDataR...
EB1	active	256.0	1526.7	218.2	1311.3
AG1	active	15234.0	1310.9	15123.1	1293.6
VTP6	active	0.0	0.0	0.0	0.0
ME4	active	295.5	1520.7	224.0	1334.5

Data Rate Graph:
The graph shows Data Rate (KByte/sec) on the y-axis (0 to 2,000) and time on the x-axis. Two data series are plotted: PEB1 (red line) and AG1 (yellow line). The AG1 data rate is relatively constant around 1,300 KByte/sec, while the PEB1 data rate shows significant fluctuations, peaking near 1,800 KByte/sec and dropping to near zero.

Message Log:

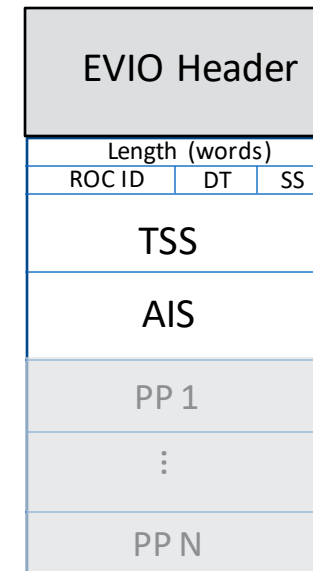
Name	Message	Time	Severity
Gui-90	Configure is started.	08:55:03 05/05	INFO
ms_DESY_Stream_1	Configure succeeded.	08:55:08 05/05	INFO
ms_DESY_Stream_1	Download is started.	09:08:13 05/05	INFO
ms_DESY_Stream_1	waiting for... VTP6,	09:08:22 05/05	WARN
ms_DESY_Stream_1	Download succeeded.	09:08:31 05/05	INFO
ms_DESY_Stream_1	Prestart is started.	09:08:37 05/05	INFO
ms_DESY_Stream_1	waiting for... VTP6,	09:08:46 05/05	WARN
ms_DESY_Stream_1	waiting for... VTP6,	09:08:51 05/05	WARN
ms_DESY_Stream_1	Prestart succeeded.	09:08:53 05/05	INFO
ms_DESY_Stream_1	Go is started.	09:08:55 05/05	INFO
EB1	Emu PEB1 go: waiting for PRESTART event in module EbModule (client msg)	09:08:55 05/05	INFO
AG1	Emu AG1 go: waiting for PRESTART event in module EbModule (client msg)	09:08:55 05/05	INFO
ms_DESY_Stream_1	Go succeeded.	09:08:59 05/05	INFO

“Zero” Suppression

- In the streaming environment we have to manage these the two extremes
 - Empty time frames
 - Too much data for available bandwidth
- For the current Streaming ROC format there is a minimum 72 bytes/frame sent.
 - For 65 μ s frames that comes to ~1.1MB/s “empty” data rate.
- Making time frames longer reduces the overhead. Allow for an adjustable time frame (planning to expand support fo frame size 65 μ s to >1ms).
- EVIO Header (32 bytes) is just for transport to the Aggregator – then stripped.

Empty Frames for the Beam Tests

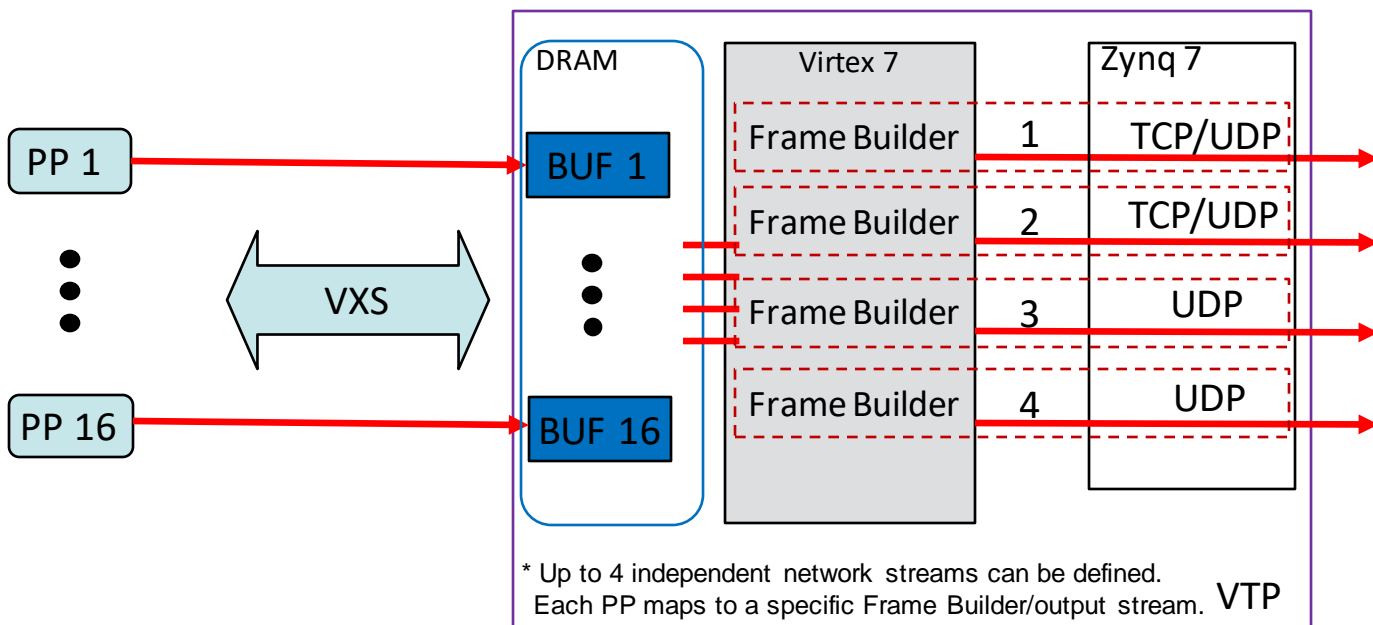
TET (threshold over pedestal)	Empty Frames	Frames with Triggers
10	68.9%	0.084%
50	97.0%	0.64%



Minimum 40 bytes for up to 2 payloads.
Total 68 bytes for 16 payloads

Congestion & Latency Management

- Too much data can be handled both **locally** and **globally**
- Globally “Sync” is used to start and stop all the streams at their source (FADCs)
 - An optional “Busy” generated by any ROC can feed back and inhibit all streams for all ROCs
- Locally, PPs stream hits to a VTP DRAM buffer. The Frame Builders have a frame “fifo”
 - DRAM can hold programmable number of frames (anywhere from 1 to many thousands).
 - When frame buffer is full, the front-end payload card will drop the frame (frame timestamps allow tracking loss).
 - This frame buffer depth constrains the latency of the frame -> network (when UDP is used):
Our typical settings: $\text{max latency} = 131\text{kByte} (\text{FrameSize_max}) * 256 (\text{FrameBuffer_max}) * 4 (\text{PP_num}) / 9.5\text{Gbps} (\text{LinkSpeed}) = 113\text{ms maximum}$
80μs minimum is typical since we expect to operate at low link occupancy



Note:

TCP Performance

~7-8 Gbps per link without frame drops

UDP performance (8000 MTU)

>9.5Gbps per link without frame drops

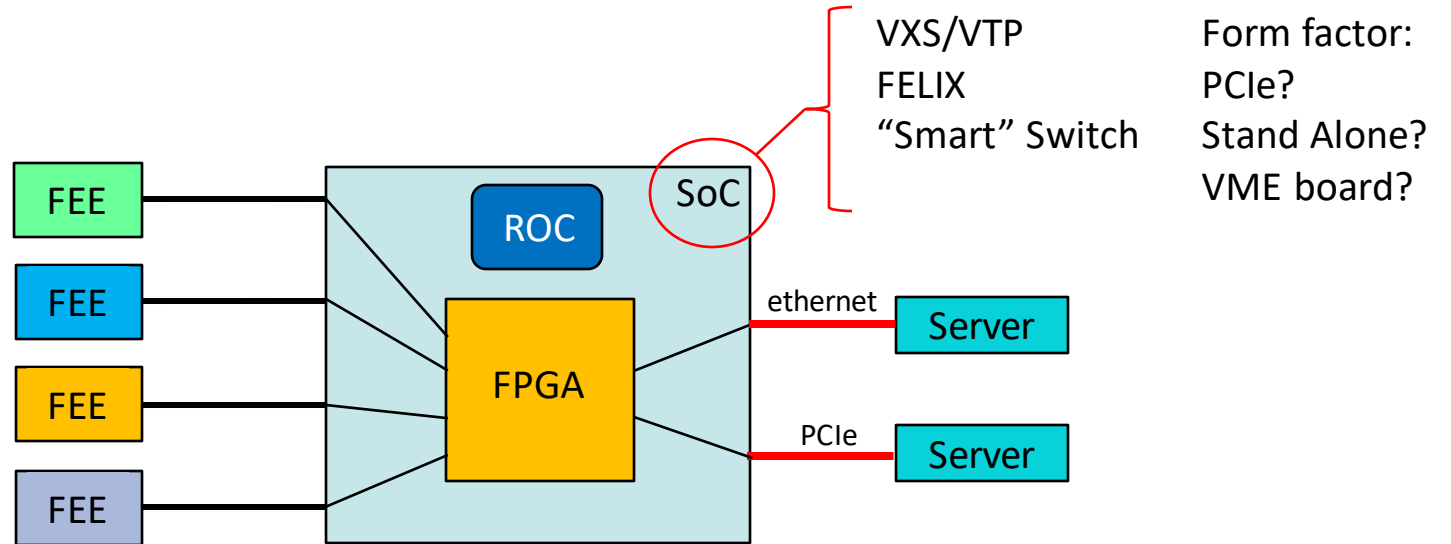
~50% CPU utilization for a single stream

(These tests were done with both VTP and a Server connected through a single switch)

Some general observations...

- The design of our Clock/Trigger/Sync/Busy distribution system is critical to the flexibility and functionality of the CODA Hybrid DAQ.
- The VXS platform works for JLAB (as we have a large inventory of this hardware), but is an impractical and/or financial overhead for small university groups. The same could be said for other solutions like the FELIX/DAM architecture for EIC.
- It seems there is a need for an affordable COTs or alternative "lightweight" solution which supports Streaming that can be made available to the community for development and test systems. Front-end modules with optical outputs ideally would have Ethernet support for easy compatibility with standard PCs.

The critical component to just about any system is a System on a Chip with enough resources to support at least a few Front End electronics [serial link protocols](#) and perform 1st stage hardware stream aggregation. And present the data to the next stage in a standardized way.



Summary

- We have successfully started integration of Streaming support within the CODA software framework and supported hardware.
- The Hybrid DAQ system give us a lot of flexibility to support older hardware within a Triggered system as well as newer hardware that can conform to the the Streaming requirements.
- UDP data transport from the VTP is proving to be reliable and the most efficient method for getting data to backend processing.
- Upgrades to JLAB FADC firmware this Summer will allow for more streaming options including waveforms.
- The ability to take both Triggered and Streaming data simultaneously should provide useful data for Online processing to better define efficient event identification algorithms as part of a high level trigger.
- Integration of other ASIC-based front-end electronics within the CODA streaming environment still needs to be developed.
- CLAS12 DAQ is working to replace a few front-end electronics that don't support streaming. They are making an effort to secure the option to switch to a streaming DAQ. Would be a great test bed that may not require a huge investment.

Timing System Components

