# 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

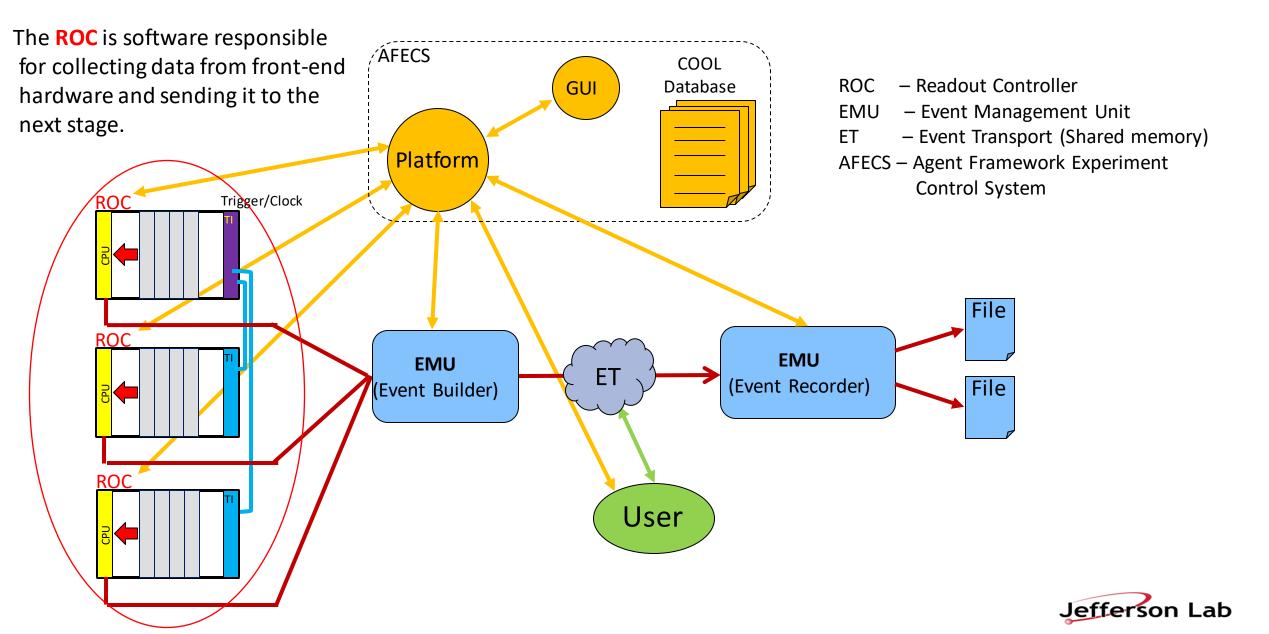


#### Data Acquisition at Jefferson Lab

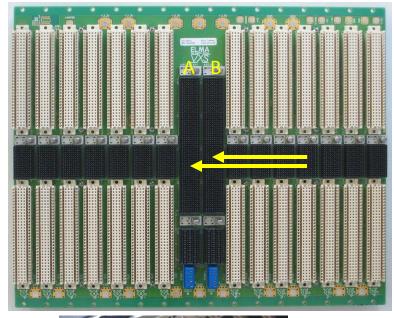
- At JLab we have 4 Experimental Halls, all running with different detectors and physics priorities.
  - Of course, all are having increased demands for the DAQ.
- Experiments are increasingly reliant custom electronics to interface detectors and digitize signals.
  - ASICs and FPGAs are becoming the 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



# 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.

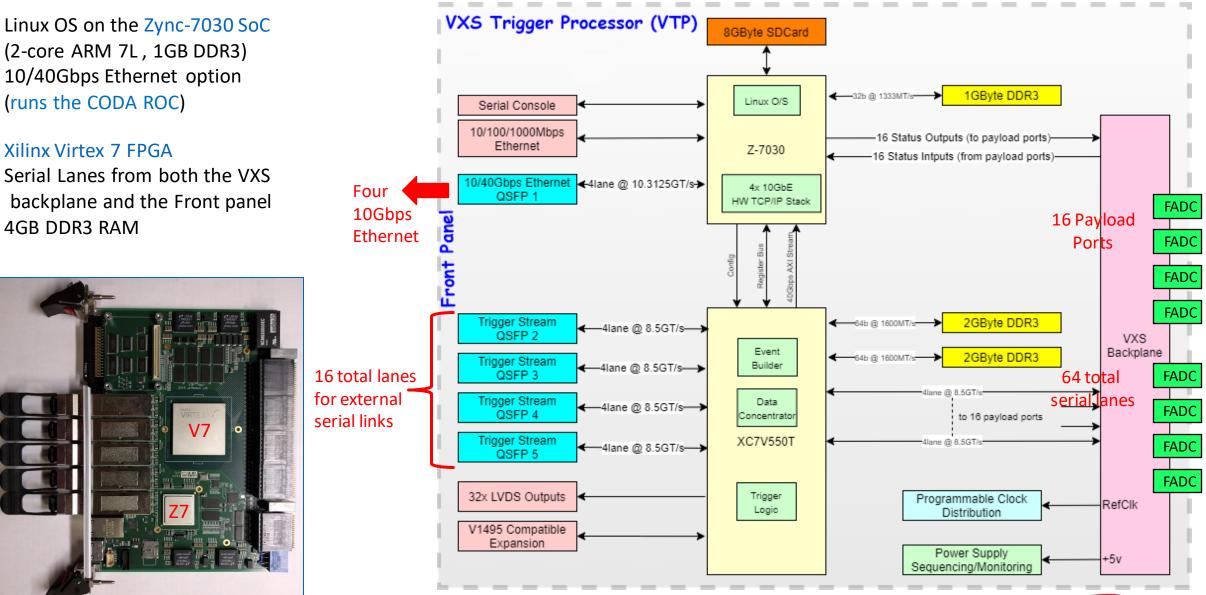


#### VXS Trigger Processor (VTP)

- 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 genral, 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.

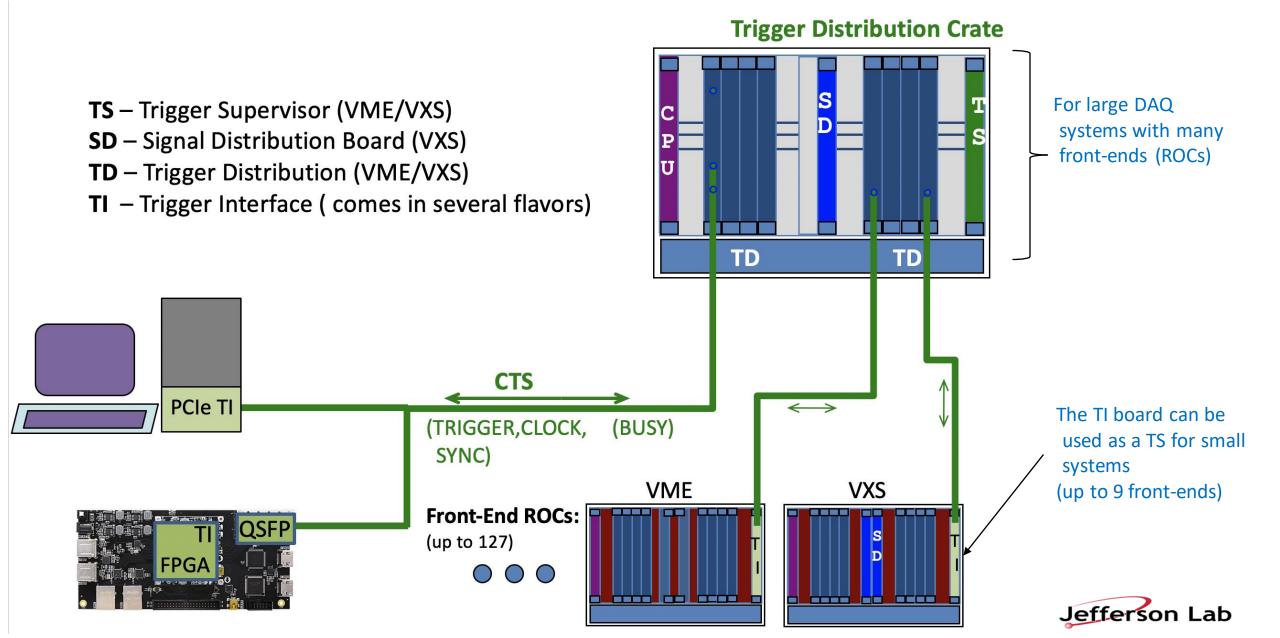


#### JLAB – VTP Board





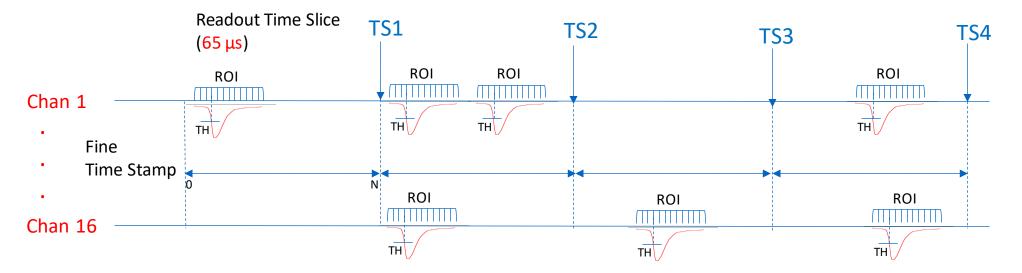
#### JLAB Clock and Trigger Distribution System



### 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 65µs



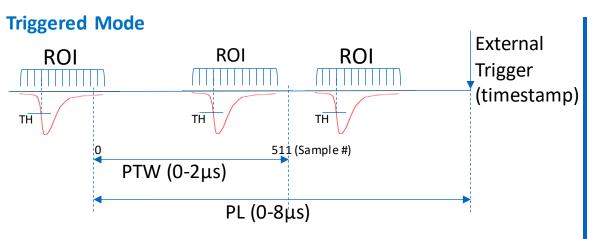


The next revision to the firmware will have an option for full ROI wave forms to be streamed, but this will allow possible dropped hits due to bandwidth limitations

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



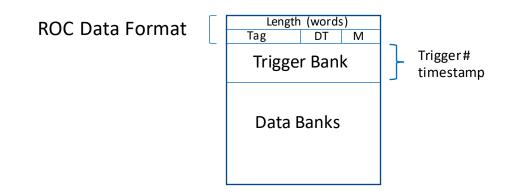
## **FADCs - Triggered vs Streaming**

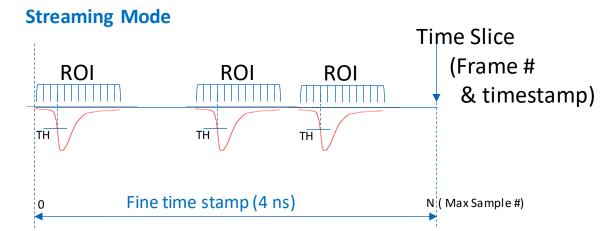


PL: Programmed Lookback PTW: Time window

Data we get on a trigger:

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

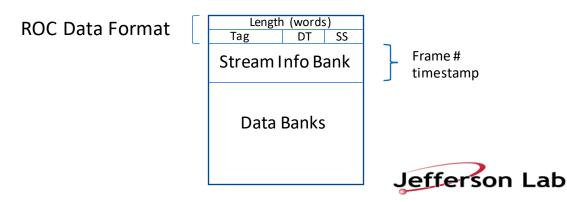




1 Frame = N Clocks (up to 16bits, currently 65536 ns)

#### 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



#### **Firmware Compiler**

#### Primary tools:

Vivado (VHDL & SystemVerilog)

- Needed for fast and/or minimized logic, specialized bus interfaces, specialized IP, multi-clock domain handling

Vivado HLS (C/C++)

- Used for data processing, formatting, and aggregation
- No problem saturating 10G Ethernet ports

Vivado HLS Project - C++ w/special HLS directives to instruct parallelism:

· · · · · · · · · · · · · · · · · · ·	8 - Q	🕸 Debug 👔 Synthesis 🔗 Analysis
Synthesis(solution1) 22 (2+ Dataflow(solution1)	Performance(solution1)	a • [
nthesis Report for 'streaming_eb_hls'		🖻 streaming_eb_hls.cpp 🕱 🧧 🗖
		550 #pragma HLS INTERFACE ap_fifo port=s_mig1_rd_addr
Seneral Information		551 #pragma HLS INTERFACE ap_fifo port=s_mig0_rd_addr 552 #pragma HLS INTERFACE ap_fifo port=s_mig0_rd_addr
Date: Wed Jul 27 16:42:20 2022		553 #pragma HLS INTERFACE ap_stable port=roc_id
Version: 2017.4 (Build 2086221 on Fri Dec 15 2)	1:13:33 MST 2017)	554 #pragma HLS DATAFLOW
Project: streaming_eb_hls		555 static his::streamframe_info_t> s_frame_info_0[16]; 556 static his::streamframe_info t> s_frame_info_1[16]:
Solution: solution1		static his::stream <frame_info_t> s frame_info_[16];</frame_info_t>
Product family: virtex7		<pre>S58 static hls::stream<frame_info_t> s_frame_info_3[16];</frame_info_t></pre>
larget device: xc7vx550tffg1927-1		559 #pragma HLS DATA_PACK variable=s_frame_info_0
erformance Estimates		560 #pragma HLS DATA_PACK variables_frame_info_1 561 #pragma HLS DATA PACK variabless frame_info_2
		562 #pragma HLS DATA PACK variable=s frame_info_3
Timing (ns)		563 static hls::stream <ebif data64="" t=""> s eb info64[4];</ebif>
Summary		564 #pragma HLS STREAM variable=s_eb_info64 depth=64 dim=1
Clock Target Estimated Uncertainty		<pre>565 #pragma HLS DATA_PACK variable=s_eb_info64 566 static hls::stream<ebif data64="" t=""> s ebif data64 partial[4];</ebif></pre>
ap_clk 6.40 6.00 0.80		567 #pragma HLS STREAM variable=s_ebif_data64_partial_depth=64_dim=1
Latency (clock cycles)		568 #pragma HLS DATA_PACK variable=s_ebif_data64_partial
Summary		569 static hls::stream <ddr3_rd_din_t> s_mig0_rd_din[4]; 570 static hls::stream<ddr3_rd_din_t> s_mig1_rd_din[4];</ddr3_rd_din_t></ddr3_rd_din_t>
Latency Interval		5/0 static nts::streamwoors_ro_un_ts s_magiro_uni4; 571 #program HLS DATA PACK variable=s magor d din
min max min max Type		572 #pragma HLS DATA PACK variable=s mig1 rd din
? ? ? ?dataflow		573
E Detail		574 // ddr3 burst reader 575 ddr3 reader<0>(s mig0 rd din[0], s mig0 rd addr[0]);
Instance		5/5 dor3_reader <u>is_migu_ro_dan(u), s_migu_ro_addr(u)); 576 ddr3 reader<u>is_migu ro_dan(1), s_migu_ro_addr(1));</u></u>
	atency Interval	577 ddr3_reader<2>(s_mig0_rd_din[2], s_mig0_rd_addr[2]);
	nin max min max Type	578 ddr3_reader<3>(s_mig0_rd_din[3], s_mig0_rd_addr[3]);
frame info proc 0 U0 frame info proc 0 s	? ? ? ? none	579 ddr3_reader<4>(s mig1_rd din[0], s mig1_rd addr[0]); 580 ddr3 reader<5>(s mig1 rd din[1], s mig1 rd addr[1]);
frame_info_proc574_U0 frame_info_proc574	? ? ? ? none	580 doi's reader-56 (s migi rd dair[1], s migi rd addr[2]);
frame_info_proc575_U0_frame_info_proc575	? ? ? ? none	582 ddr3 reader<7>(s mig1 rd din[3], s mig1 rd addr[3]);
frame_info_proc576_U0_frame_info_proc576	7 7 7 7 none	563
eb_data_writer_0_U0eb_data_writer_0_s eb_data_writer_1_571_U0eb_data_writer_1_571	1 1 1 1 1function 1 1 1 1 1function	<pre>584 // repack data to ensure 64bit writes until last word in frame 585 ebif_writer&lt;0&gt;(s_ebif_data64_partial[0], s_ebif_data64[0]);</pre>
eb data writer 2 572 U0eb data writer 2 572	1 1 1 1 Ifunction	<pre>bol_writeroor(s_bol_doutow_partiat(0), s_bol_datao(0)), S86 ebif writer &gt;(s bbif data64 partiat(1), s ebif data64(1));</pre>
	1 1 1 1 Ifunction	587 ebif_writer<2>(s_ebif_data64_partial[2], s_ebif_data64[2]);
ebif_writer_0_U0 ebif_writer_0_s	1 1 1 1 function	<pre>588 ebif_writer&lt;3&gt;(s_ebif_data64_partial[3], s_ebif_data64[3]);</pre>
ebif_writer_1_568_U0 ebif_writer_1_568	1 1 1 1 function	589 590 // Event writers
ebif_writer_2_569_U0 ebif_writer_2_569 ebif_writer_3_570_U0 ebif_writer_3_570	1 1 1 1 1function 1 1 1 1 1function	390 // EVENT WILLOPS 591 eb data wilter-Ge-(s eb info64[0], s miq0 rd data[0], s miq1 rd data[0], s ebif data64 partial[0]);
ddr3 reader 0 U0 ddr3 reader 0 s	1 1 1 1 Ifunction	592 eb_data_writer <l>(s_eb_info64[1], s_mig0_rd_data[1], s_mig1_rd_data[1], s_ebif_data64_partial[1]);</l>
ddr3 reader 1 562 U0 ddr3 reader 1 562	1 1 1 1 Ifunction	593 eb_data_writer<2>(s_eb_info64[2], s_mig0_rd_data[2], s_mig1_rd_data[2], s_ebif_data64_partial[2]);
ddr3_reader_2_563_U0 ddr3_reader_2_563	1 1 1 1 Ifunction	594 eb_data_writer<3>(s_eb_info64[3], s_mig0_rd_data[3], s_mig1_rd_data[3], s_ebif_data64_partial[3]); 595
ddr3_reader_3_564_U0 ddr3_reader_3_564	1 1 1 1 Ifunction	395 396 // Payload frame info/header processors
ddr3_reader_4_U0 ddr3_reader_4_s	1 1 1 1 1function	597 frame_info_proc<0>(cmsg_hdr_enable, ss_total_streams, frame_len, roc_id, port_enable_mask[0], ais_module_id, cpu_evt_async_en, stream
ddr3_reader_5_565_U0 ddr3_reader_5_565 ddr3 reader 6 566 U0 ddr3 reader 6 566	1 1 1 1 1function 1 1 1 1 1function	598 frame_info_proc <l>(cmsg_hdr_enable, ss_total_streams, frame_len, roc_id, port_enable_mask[1], ais_module_id, cpu_evt_async_en, streams</l>
ddr3 reader 6 566 00 ddr3 reader 6 566 ddr3 reader 7 567 U0 ddr3 reader 7 567	1 1 1 1 Ifunction	509 frame_info_proc<>>(cmsg_hdr_enable, ss_total_streams, frame_len, roc_id, port_enable_mask[2], ais_module_id, cpu_evt_async_en, streami 500 frame_info_proc<>>(cmsg_hdr_enable, ss_total_streams, frame_len, roc_id, port_enable_mask[3], ais_module_id, cpu_evt_async_enable_n
frame_info_router_U0 frame_info_router	0 0 1 1function	000 Trame_into_proc<3>icmsg_nor_enable, ss_total_streams, trame_ten, roc_id, port_enable_mask[3], ais_module_id, cpu_evt_async_en, stream 601
E Loop		602 // Payload frame info router
		603 frame_info_router(port_enable_mask, s_frame_info, s_frame_info_0, s_frame_info_1, s_frame_info_2, s_frame_info_3); 604 }
tilization Estimates		605

#### Vivado Project - usual mix of VHDL and Verilog custom and Vendor IP:

Edit Flow Tools Window La	***** **** m++ -	l∼ Quick Access												-,	nthesis Out-of-date	
ROJECT MANAGER - vtp_fadc_stream_v7															- Derauk cayou	?
Sources		oject Summary														? 🗆 🖒 🗙
Q ጟ ♦ + 1 0 ● °		Project name:	vtp_fadc_st													
✓ ֎▲ v7_vtp(synthesis) (v7_vtp.vhd		Project location:		do/Projects/	e_fw/Firm	ware/Pr	ojects/v	tp_fadc_stream	n_v/							
> @ vtp_master_bridge_inst : vtp > @ v7clk_per_inst : v7clk_per(s)		Product family:	Virtex-7 xc7vx550tff													
> @ v7sd per inst : v7sd per(sy		Project part: Top module name:	v7_vtp	drast												
> @ streaming_eb_per_inst0 : st		Target language:	VHDL													
> @ v7mig per inst r : v7mig pe	7 11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Simulator language:	VHDL													
> 🐵 v7mig_per_inst_l : v7mig_per	(synthesis) (v7mig	sinulator language:	VHUC													
> 🛞 ebio_tx_per_inst0 : ebio_tx_p																
> @ ebio_tx_per_inst1 : ebio_tx_p		iynthesis							Implement	tation						
> @ qsfp_per_gen[3].v7aurora8t > @ qsfp_per_gen[2].v7aurora8t		Status: 🚽	Out-of-date						Status:		Not sta	rted				
> @ qsfp_per_gen[1].v7aurora8t		Messages: 🧕 🌖	1502 warnings						Messages:		No erro	rs or warn	ings			
> @ qsfp_per_gen[0].v7aurora8t		Active run:	synth_1						Active run:		impl_1					
> @ vxstrg_per_gen[15].v7auror		Part:	xc7vx550tffg19	27-1					Part:		xc7vx55	0tffg1927	-1			
> 🙆 vxstrg_per_gen[14].v7auror	a8b10b_per_gen.vi	Strategy:	Vivado Synthes	is Defaults					Strategy:			mplement				
> 🐵 vxstrg_per_gen[13].v7auror		Report Strategy:	Vivado Synthes	is Default R	ports				Report Str	ategy:	Vivado I	mplement	ation De	fault Re	ports	
> @ wstrg_per_gen[12].v7auron									Incrementa	al compile:	None					
	·····>															
Hierarchy IP Sources Libraries	Compile ( 4 )> =	ORC Violations							Timing							
Properties	? _ 🗆 🖾 ×		Run Impleme	entation to	see DRC r	esults					Run Imple	mentation	to see	timing r	results	
	+ + •					1.5										
	·	Itilization		Post-S	ynthesi	S   POS		nentation	Power							
							Grap	h   Table								
		т-	33	%												
		LUTRAM - 4%														
		FF -	20%													
		BRAM -	29%													
Select an object to see pro	perties	10 -	_	_		74%					Run Imple	mentation	to see	nower r	a filia a	
		GT -	_	_	-		-	L00%			riari inges	in the first first	10 500			
		BUFG -			59%											
		MMCM -	25%													
		PLL-	15%													
		0	25	50	7	5	10	0								
				Estimated	Utilization	n (%)										
d Console Messages Log R	eports Design Runs	×														? _ 🗆
Q																
lame	Constraints	Status		WNS TN	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed
synth_1 (active)	constrs_1		Out-of-date								139431		0	0		
⊳ impl_1	constrs_1	Not started														
Prove of Contract Madule Dune																



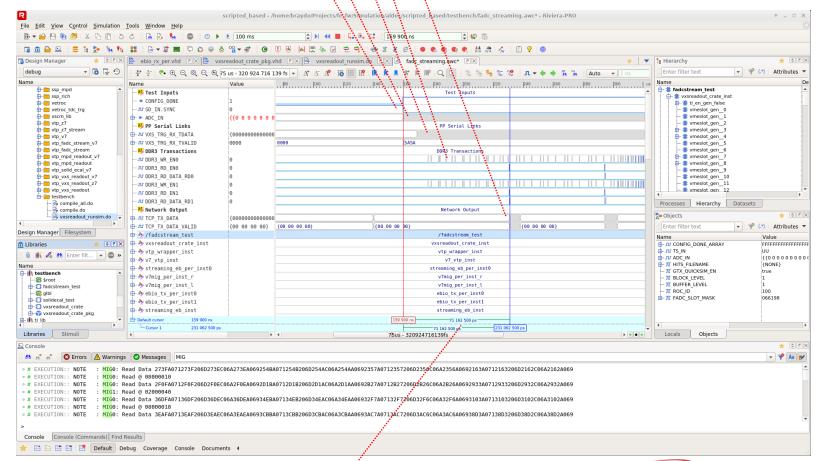
### **Front-end Crate Simulation**

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 65µs for a single crate of 256 FADC channels, but often only takes a few frames to get needed information

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

Jefferson Lab



71µs 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.

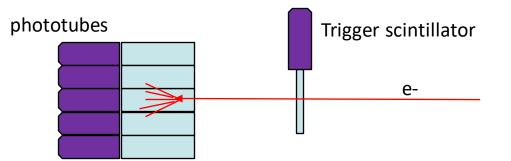
E.g. jeviodmp is checking 'event l	block' format for correctness:
------------------------------------	--------------------------------

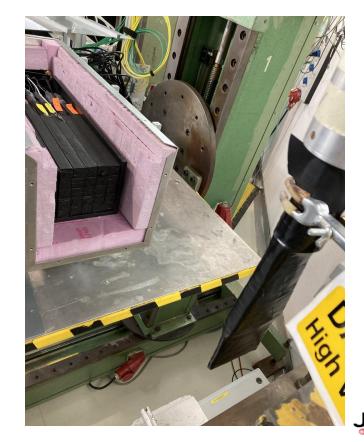
ile									
ne									
			/scratch/ab	bottd/data/vtp2	streamv3_1TCF	P_1000.dat			
Sear	ch By	Word Position	+1	+2	+3	+ 4	+5	Comments	Color Key-
	cirby	0	0x634d7367	0x20697320	0x636f6f6c	0x00000006	0x00000024	▲	Block normal
○ Word <u>V</u> alue		5	0x00400800	0x00000001	0x00000001	0×00000001	0x00000034	E	Event normal
🔾 Word Positio	n	10	0x0000000d	Oxffffffff	0x0000008	0x00000001	0x00000024		
10.000000000000000000000000000000000000		15	0x00001604	0x00000000	0xc0da0100	0x00000004	0xffd10100	Block Header	Block with erro
Page Scrollir	ig	20	0x000004b0	0x000010b7	0×00000001	0×00000001	0x00000034		Event with erro
Evio <u>B</u> lock		25	0x0000000d	Oxffffffff	0×0000008	0×00000001	0x00000024		Evio struct erro
		30	0x00001604	0×00000000	0xc0da0100	0x00000004	0xffd20100		Word value
Evio Event		35	0x000004b0	0x00000000	0×00000000	0x00000001	0x00009054		Current select
o rote roote		40	0x00002415	0×00000001	0×0000008	0×00000001	0x00000024		Current select
Evio <u>F</u> ault		45	0x00002204	0×00000000	0xc0da0100	0x0000240c	0x00241011		
		50	0x0000000a	0xff302011	0x31010003	0×00000000	0x00000000		
Search For		55	0x00000000	0×41050004	0x00840082	0x00870085	0x008c008a		
0xc0da0100		60	0x008f008d	0x0000035e	0x00020f00	0x00109fff	0x00113fal		
		65	0x00119fff	0x10489ff3	0x10493f6f	0x10499fff	0x10ac9ff6		
Search	Controls ———	70	0x10ad3f6d	0x10ad9fff	0x11109ff7	0x11113f70	0x11119fff		
		75	0x11749ff6	0x11753f6d	0x11759fff	0x11d89fef	0x11d93f6f		
<	>	80	0x11d99fff	0x123c9ff0	0x123d3f63	0x123d9fff	0x12a09ff1		
Start Scan	Stop	85	0x12a13f67	0x12a19fff	0x13049ff0	0x13053f61	0x13059fff		
Start Start	эгор	90	0x13689fea	0x13693f5a	0x13699fff	0x13cc9fec	0x13cd3f61		
Do	ine	95	0x13cd9fff	0x14309fef	0x14313f64	0x14319fff	0x14949feb		
]		100	0x14953f61	0x14959fff	0x14f89fed	0x14f93f5f	0x14f99fff		
Block Info		105	0x155c9fe9	0x155d3f59	0x155d9fff	0x15c09fe6	0x15c13f5c		
Total words	13	110	0x15c19fff	0x16249fe0	0x16253f5b	0x16259fff	0x16889fea		
Header words	8	115	0x16893f65	0x16899fff	0x16ec9fe5	0x16ed3f5a	0x16ed9fff		
	-1	120	0x17509fdf	0x17513f53	0x17519fff	0x17b49fdf	0x17b53f57		
	-1	125	0x17b59fff	0x18189fe5	0x18193f5a	0x18199fff	0x187c9fda		
Event count	1	130	0x187d3f59	0x187d9fff	0x18e09fe5	0x18e13f5b	0x18e19fff		
Version	4	135	0x19449fdc	0x19453f57	0x19459fff	0x19a89fda	0x19a93f60		
Has dictionary	false	140	0x19a99fff	0xla0c9fe0	0x1a0d3f54	0xla0d9fff	0x1a709fdd		
	true	145	0x1a713f5c	0xla719fff	0x1ad49fe4	0x1ad53f56	0xlad59fff		
10 1000		150	0x1b389fe9	0x1b393f60	0x1b399fff	0x1b9c9fe0	0x1b9d3f62		
		155	0x1b9d9fff	0x1c009fd4	0x1c013f5a	0x1c019fff	0x1c649fdc		
		160	0x1c653f60	0x1c659fff	0x1cc89fdf	0x1cc93f5c	0xlcc99fff		



#### **Beam Tests**

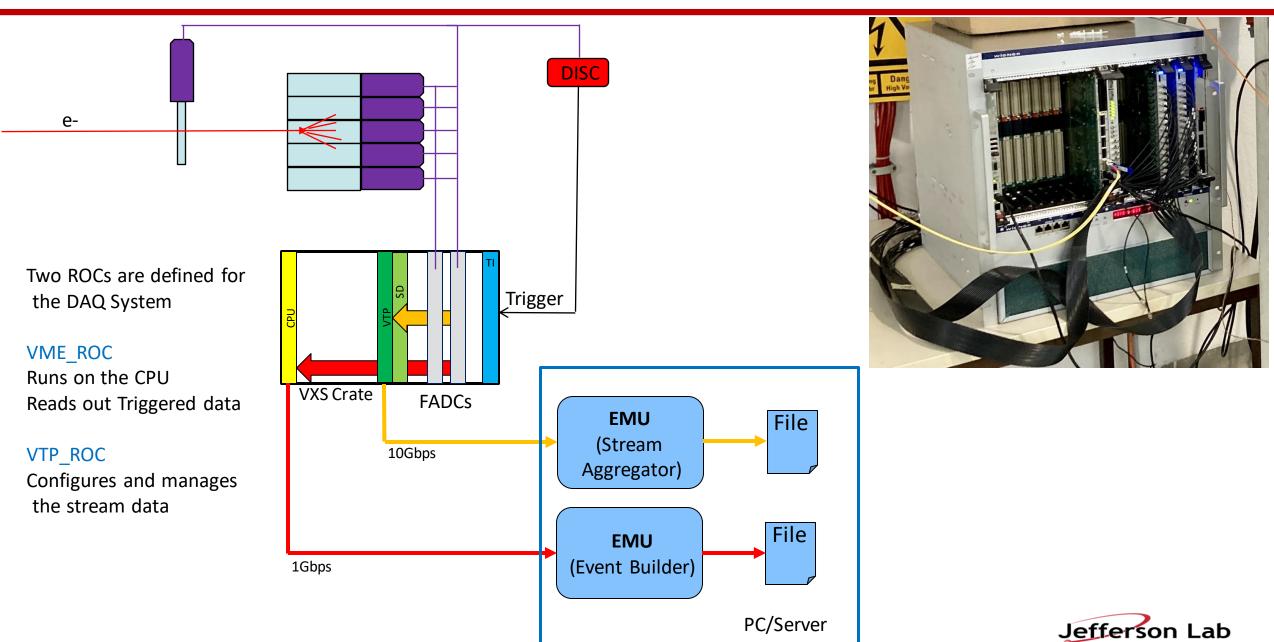
- Recent beam tests with a calorimeter prototype at DESY (*Thanks to Doug Hasell for coordinating this opportunity*)
- 5x5 PbWO4 Crystal Array (2 cm^2 face) with 2-5GeV electron test beam
- Jlab 250Mhz FADC boards
  - Triggered data are waveforms read out over VME bus.
  - Stream data are integrated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





Jefferson Lab

#### Simple Hybrid CODA System

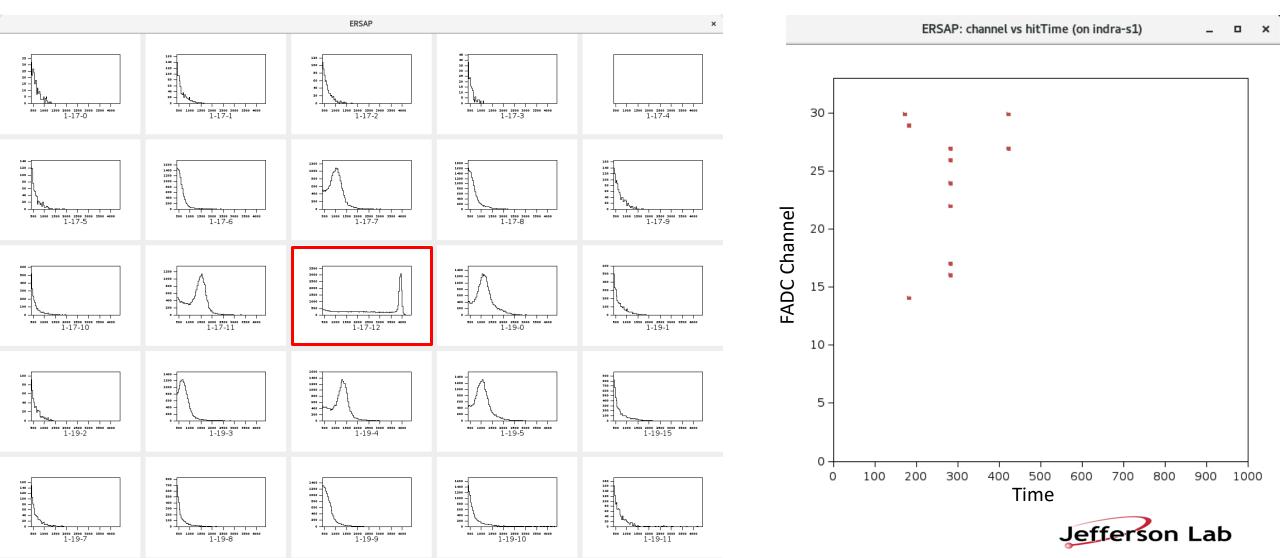


#### **Beam Tests**

Calorimeter spectra – Streaming Data

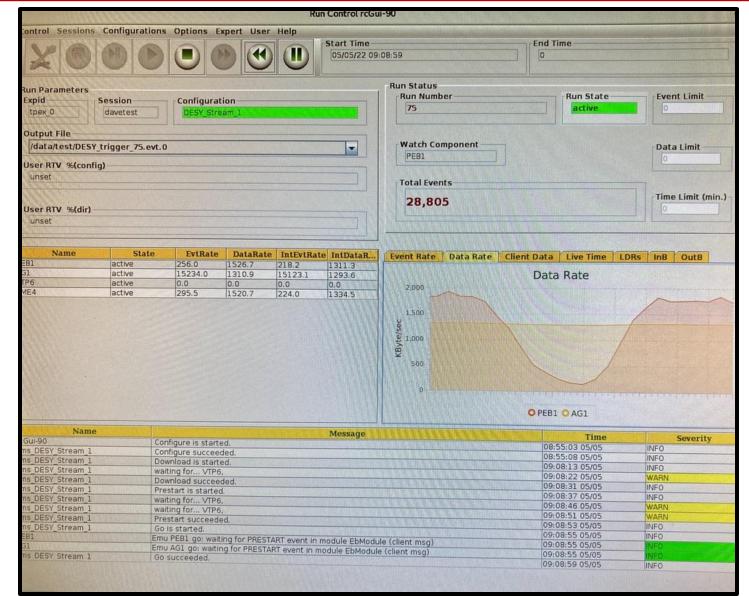
(Electron beam centered on the central crystal)

#### 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)



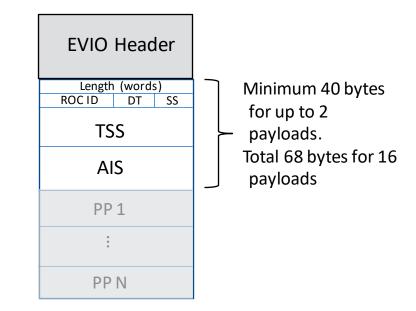


## "Zero" Suppression

- 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 65µs 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 65µs to >1ms).
- EVIO Header (32 bytes) is just for transport to the Aggregator then stripped.

#### Empty Frames for the Beam Tests

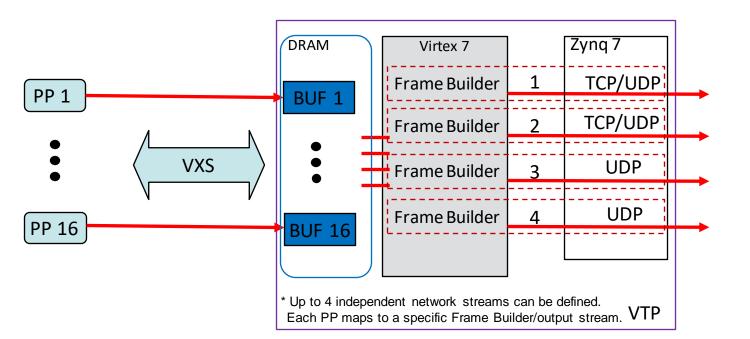
TET (threshold over pedestal)	Empty Frames	Frames with Triggers
10	68.9%	0.084%
50	97.0%	0.64%





### **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: max latency = 131kByte (FrameSize\_max) \* 256 (FrameBuffer\_max) \* 4 (PP\_num) / 9.5Gbps (LinkSpeed) = 113ms 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

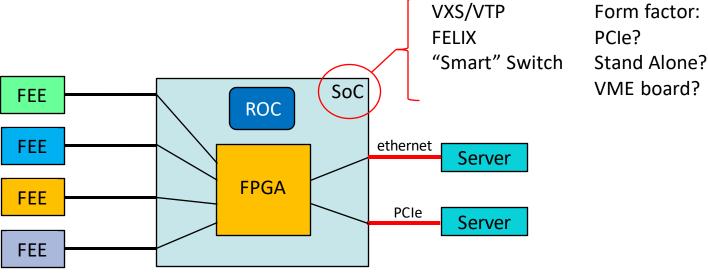
(These tests were done with both VTP and a Server connected through a single switch)



### 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 1<sup>st</sup> 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**

