Fermilab

U.S. DEPARTMENT OF ENERGY | Office of Science

Mu2e-II DAQ Thoughts

14-Sep-2020 Trigger & DAQ

Mu2e-II Workshop

Ryan Rivera – Mu2e TDAQ L2

Introduction

- What are the requirements for the Mu2e-II DAQ?
- Mu2e-II will have more beam on target and higher granularity detectors.
- Assumptions:
  - Power and cooling limitations are solved by money
  - Installation around 2030
  - Control and Synchronization of the detector will work itself out, this talk focuses on Trigger and Data Paths
- This talk introduces some DAQ thoughts, hopefully the presentations to follow and our discussion will help make the thoughts coherent.

Implications (1 of 2)

- ~2x more detector channels, and ~5x more pulses on target, for ~10x higher data rate (if background remains the same).
  - Current expected Mu2e-I data rate from front-ends is 40GBps
- More detector channels and more background implies bigger event sizes (maybe ~3x?)
  - Mu2e-I expected event size is 200KB
- Tape capacity for Mu2e-I is 7PB/year
  - Might assume 2x increase for Mu2e-II to 14PB/year

- Necessary rejection for Mu2e-II is ~3000:1
  - 600KB events @ 3MHz ➔ 560MB

Implications (2 of 2)

- Reduced OFF Spill periods (to no OFF Spill time?) implies less advantage for large front-end buffers streaming data
  - In Mu2e-I, have second of downtime to play catchup
  - In Mu2e-II, steady event rate (could buffer just to handle event to event variation, not large accelerator time structures)
- No large front-end buffers at CRV would imply need for low-latency trigger decision for CRV.
  - Low latency trigger decision implies an FPGA trigger layer.
- Consider the cost of these scenarios:
  1. Large CRV buffers and software trigger
  2. Small CRV buffers and hardware trigger

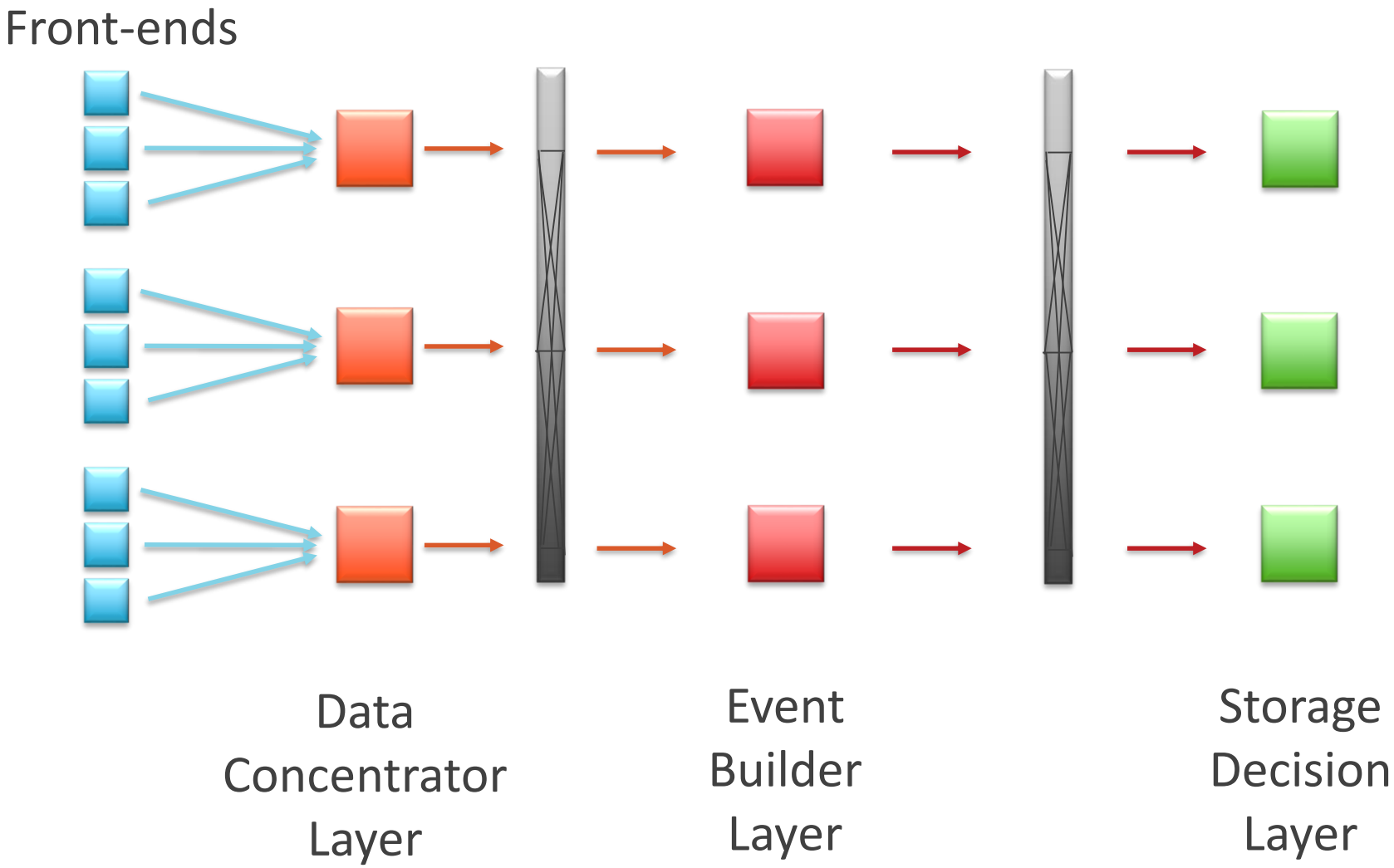
Streaming vs Triggered

- Important upfront decision as to which detector subsystems are triggered.
- Same as Mu2e-I?
  - Stream all Tracker and Calorimeter data
  - Software Trigger for CRV based on Tracker and Calorimeter
- Alternatives?
  - Stream Calorimeter Data
  - Hardware Trigger for Tracker and CRV based on Calorimeter
  - High-level Software Trigger for storage decision

Radiation Tolerance Requirements

- Radiation levels at the detector will be higher than Mu2e-I
  - Mu2e-II comparable to Calorimeter level of CMS phase-II?
- For Mu2e-I, using the VTRx was a primary constraint
  - We had to change the DAQ topology as a result
- Mu2e-II likely will not want to design their own rad-hard links, so we will be at the mercy of CMS/Atlas (again)
  - This should be worked out as soon as possible.

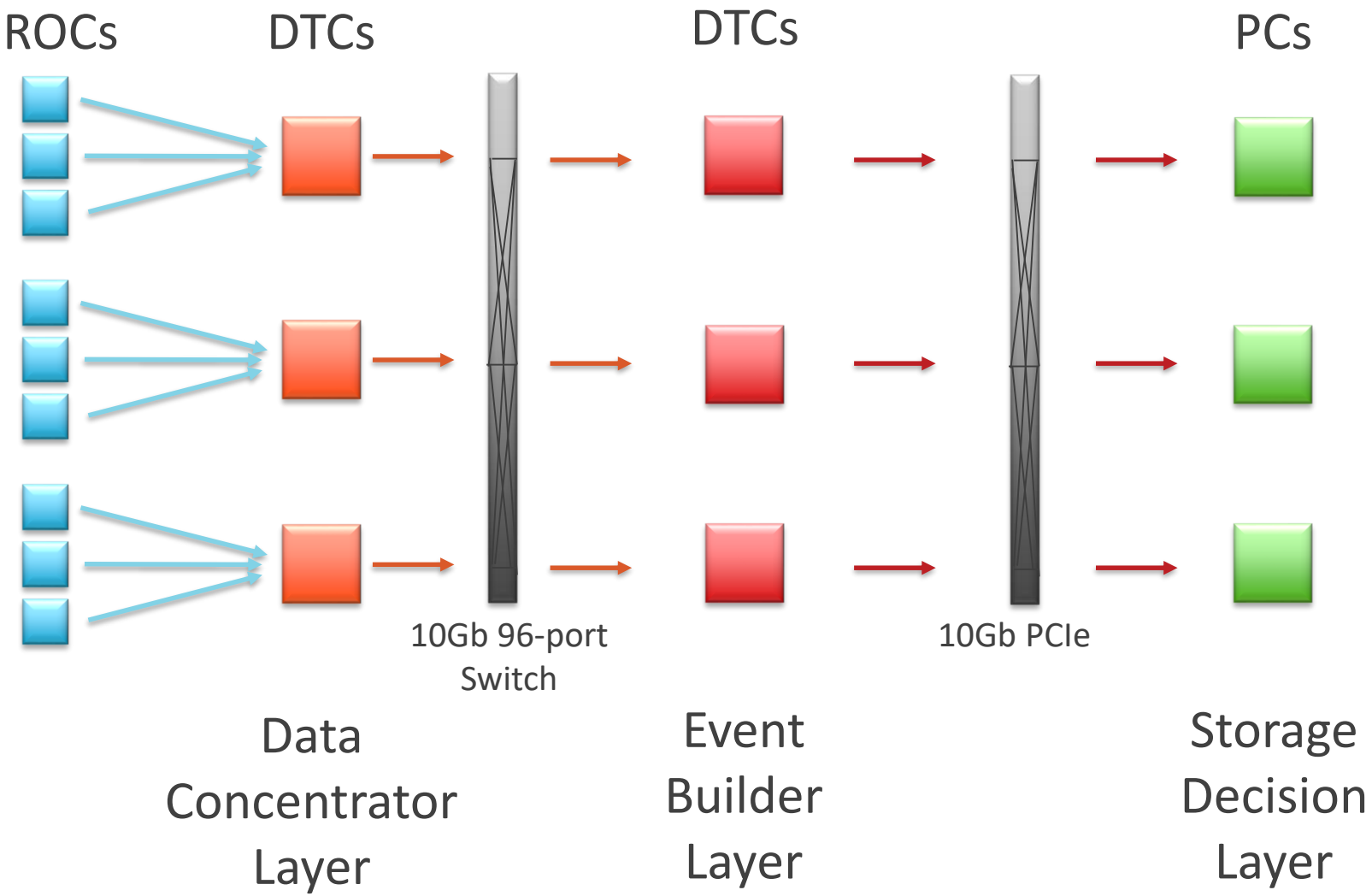
Generic Data Readout Topology



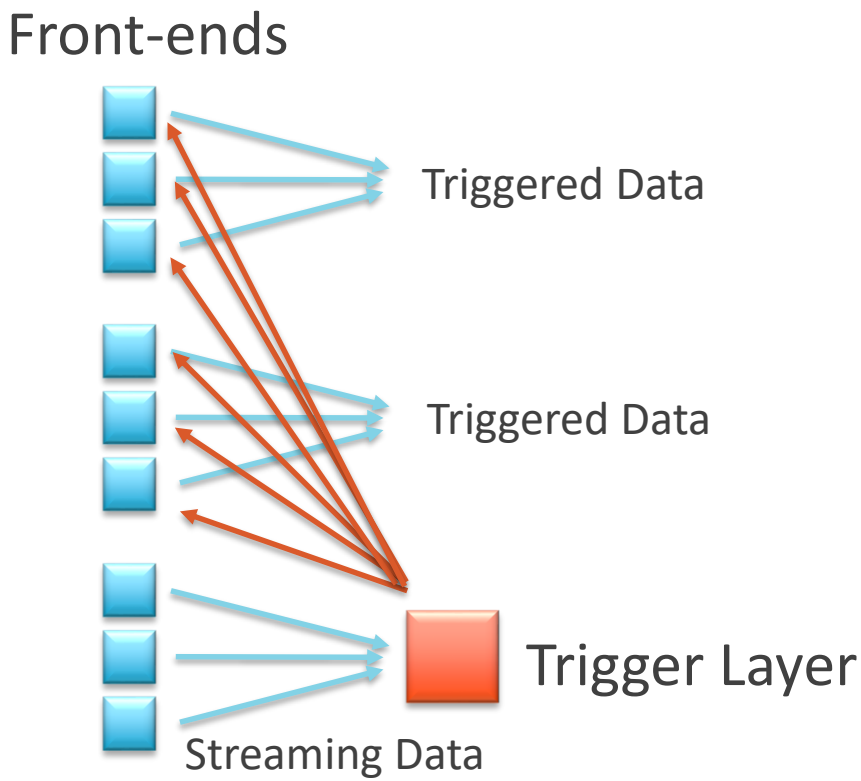
Generic Data Readout Topology

- **Data Concentrator Layer**
  - Aggregate small front-end fragments into larger chunks for efficient event building
- **Event Builder Layer**
  - Data is switched from Concentrator Layer to Event Builder Layer such that full events arrive at Event Builder Layer and are buffered.
    - Preprocessing or filtering could occur
- **Storage Decision Layer**
  - Available decision nodes make high level storage decision on full events retrieved from Event Builder Layer buffer.

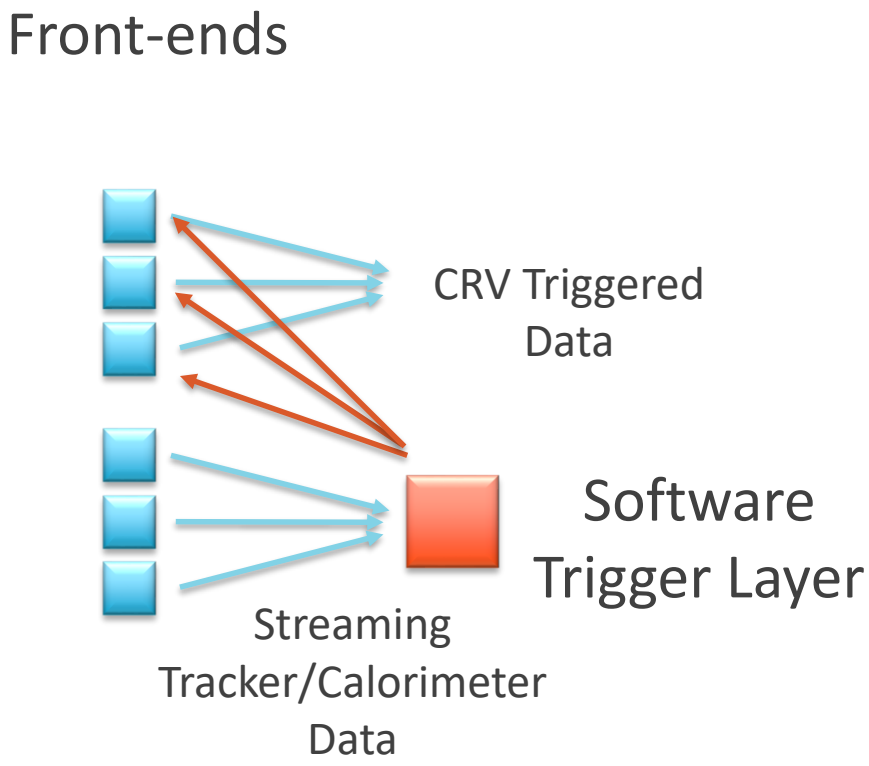
Generic Data Readout Applied to Mu2e-I



Generic Trigger Path Topology



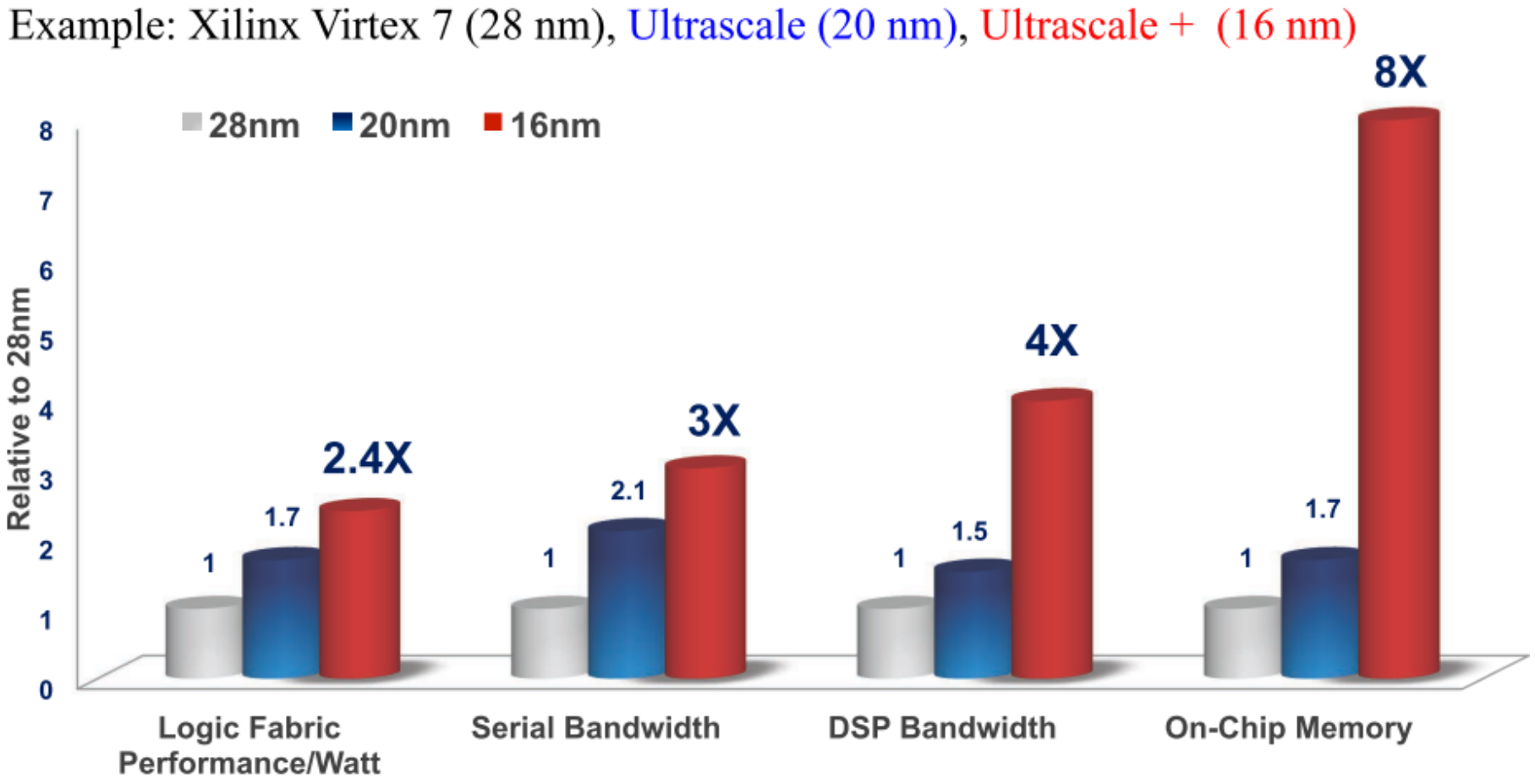
Generic Trigger Path Applied to Mu2e-I



Generic Topology Applied to other experiments

- Other workshop talks will describe how other experiment (e.g. ATLAS and CMS) map to this generic topology.

FPGA scaling



FPGA scaling

Mu2e-I DTC →

	KINTEX-7	KINTEX UltraSCALE	VIRTEX-7	VIRTEX UltraSCALE
Logic Cells (LC)	478	1,161	1,995	4,407
Block RAM (BRAM) (Mbits)	34	76	68	132
DSP-48	1,920	5,520	3,600	2,880
Peak DSP Performance (GMACs)	2,845	8,180	5,335	4,268
Transceiver Count	32	64	96	104
Peak Transceiver Line Rate (Gb/s)	12.5	16.3	28.05	30.5
Peak Transceiver Bandwidth (Gb/s)	800	2,086	2,784	5,886
PCI Express Blocks	1	6	4	6
Memory Interface Performance (Mb/s)	1,866	2,400	1,866	2,400
I/O Pins	500	832	1,200	1,456

FPGA Trend to HLS

- High Level Synthesis is now good enough to rival manual VHDL or Verilog algorithm development.
- Allows physicists to easily understand and develop low and fixed latency FPGA algorithms.
  - Makes emulation easy for offline.
- Debug and verify in a software environment (often 10x faster iterations than firmware simulation tools).
- CMS is heavily investing in HLS approach to FPGA algorithm development.
  - There is a hls4ml collaboration developing machine learning (neural network) tools using HLS.

HLS Code

```
49      //sum up presamples
50      pedsum_type pedsum = 0;
51      for (int i = 0; i < NUM_PRESAMPLES; i++){
52          pedsum += adc[i];
53      }
54      //find average
55      adc_type pedestal = pedsum / NUM_PRESAMPLES;
56      adc_type peak = 0;
57      for (int i = START_SAMPLES; i < NUM_SAMPLES; i++){
58          if (adc[i] > peak){
59              peak = adc[i];
60          }
61          else{
62              break;
63          }
64      }
65
66      adc_type energy = peak - pedestal;
67      adc_type energy_max_adjusted = ((((energy_max_LSHIFT8 * gain_RSHIFT15) >> 9) *
68                                       inverse_ionization_energy_LSHIFT26) >> 10);
69      adc_type energy_min_adjusted = ((((energy_min_LSHIFT8 * gain_RSHIFT15) >> 9) *
70                                       inverse_ionization_energy_LSHIFT26) >> 10);
71      if (energy > energy_max_adjusted || energy < energy_min_adjusted){
72          failed_energy = 1;//failed
73      }
74      return ((failed_energy<<1) | failed_time);
```

FPGA Algorithm Development

- It's important to realize that FPGA development can take place now – hardware is not needed!
  - Starting now would help decide how many resources are needed, what size FPGA is in the ballpark, and could inform DAQ topology choices.
- Could consider associative memories for pattern matching.
- Could inform custom trigger board design or commercial board selection.

Decision Process

1. Which subsystems are streaming?
   a) What are the constraints imposed by rad-hard links?
2. Is it possible to have a low-latency Level-1 trigger with rejection power?
   - Lock an HLS developer and a firmware-system developer in a room for six months and tell them to understand the specs of a hardware trigger layer (what type of FPGA, how much memory) that would do the job.
   - A hardware trigger layer may save money
     - downstream due to data reduction.
     - upstream due to reduced buffer size.
3. How much processing is needed for High Level Trigger?

Overview of TDAQ LOIs for Snowmass 2021

1. **A 2-level TDAQ system based on FPGA pre-processing and trigger primitives**
   - ROCs (create trigger primitives, buffer event fragments), L1 FPGA layer (getting trigger primitives from calo and tracker), and HLT layer (requests event fragments from full detector)
2. **A 2-level TDAQ system based on FPGA pre-filtering**
   - Leverage HLS for FPGA rejection
3. **TDAQ based on GPU co-processor**
   - Using GPUs at HLT (or L0)
4. **A trigger-less TDAQ system based on software trigger**
   - Scale up current system.

Backup Slides

Where are the FPGAs for Mu2e-II?

- At the detector front-ends, need rad-hard ASICS (Maybe already too late to design a new one) or FPGAs.
- Low-Latency trigger
- Data concentration
- Event building
  - Can do custom application specific switching behavior
- High Level Trigger preprocessor/co-processor?
  - Other co-processors? GPUs?

FPGA Landscape

- Altera/Intel – Stratix 10
  - Up to 10 TFLOPS of single-precision floating-point DSP performance.
  - Up to 70% lower power than prior-generation high-end FPGAs
  - Up to 80 GFLOPS/Watt of single-precision floating point power efficiency.
  - Up to 144 full duplex transceivers in a single package.
  - Over 2.5 Tbps bandwidth for serial memory with support for Hybrid Memory Cube.
  - Over 2.3 Tbps bandwidth for parallel memory interfaces with support for DDR4 at 2,666 Mbps.
  - **HLS C++ to RTL**

FPGA Landscape

- Xilinx – Virtex UltraSCALE+
  - Up to 128 33G transceivers deliver 8.4 Tb of serial bandwidth
  - 460GB/s HBM bandwidth, and 2,666 Mb/s DDR4 in a mid-speed grade
  - Up to 60% lower power vs. 7 series FPGAs
  - **HLS C++ to RTL**

HLS Code

```
flag_mask_type filter( //returns flag of if it passed the cut
    //tracker packet data inputs
    tdc_type tdc0, tdc_type tdc1,
    tot_type tot0, tot_type tot1,
    adc_type adc[NUM_SAMPLES],

    calib_constant_type clockstart,
    calib_constant_type panelTDCoffset, calib_constant_type hvoffset,
    calib_constant_type caloffset,
    calib_constant_type energy_max_LSHIFT8,
    calib_constant_type energy_min_LSHIFT8,
    calib_constant_type gain_RSHIFT15,
    calib_constant_type inverse_ionization_energy_LSHIFT26
    )
{
#pragma HLS PIPELINE II=2
#pragma HLS INTERFACE ap_ctrl_hs port=return
#pragma HLS ARRAY_PARTITION variable=adc complete dim=1
```

```
#ifndef DE_DX_HLS_
#define DE_DX_HLS_

#include "ap_int.h"

#define NUM_PRESAMPLES 4
#define NUM_PRESAMPLES_LOG2 2
#define START_SAMPLES 4 //0 indexed
#define NUM_SAMPLES 15
#define NUM_SAMPLES_LOG2 4

typedef ap_uint<16> tdc_type;
typedef ap_uint<8>  tot_type;
typedef ap_uint<12> adc_type;
typedef ap_uint<12 + NUM_PRESAMPLES_LOG2> pedsum_type;
typedef ap_uint<16> calib_constant_type;
typedef ap_uint<8>  flag_mask_type;

//[500,2000]ns / tdcLSB (here it's .03125)
#define LOWER_TDC 16000
#define UPPER_TDC 64000
```