Mu2e-II DAQ Thoughts

29-Aug-2018 Trigger & DAQ Working Session

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 as many DAQ thoughts as I could come up with in a few days, hopefully our discussion will help make the thoughts coherent.
  - All corrections welcome!

Implications (1 of 2)

- ~2x more detector channels, and ~5x more pulses on target, for ~10x higher data rate.
  - Current expected Mu2e-I data rate from front-ends is 38GBps
- More detector channels and more background implies bigger event sizes (maybe ~3x?)
  - Mu2e-I expected event size is 200kB

Implications (2 of 2)

- Reduced OFF Spill periods (to no OFF Spill time?) implies less advantage to large front-end buffers for 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.
- Compare scenario cost:
  1. Large CRV buffers and software trigger
  2. Small CRV buffers and hardware trigger

Streaming vs Triggered

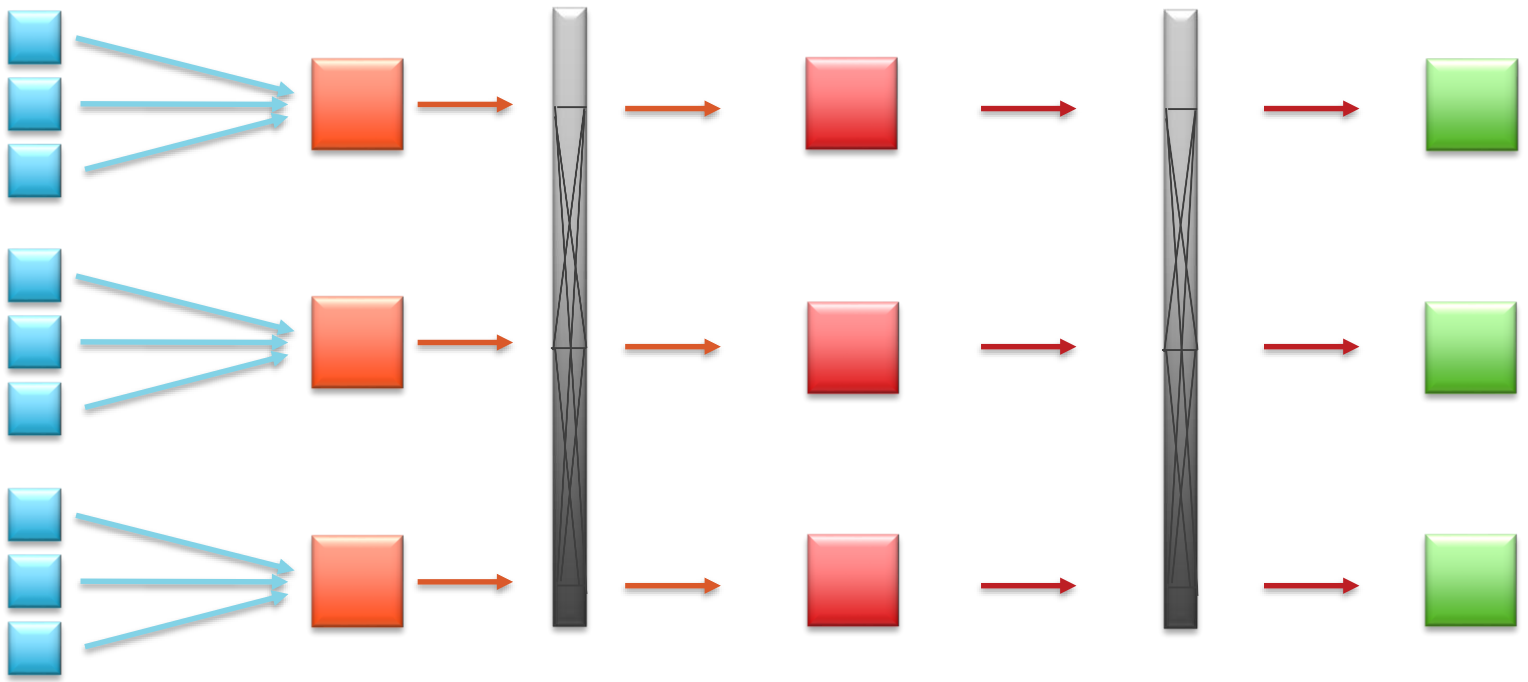
- Important upfront decision as to which detector subsystems are triggered.
- Same as Mu2e-I?
  - Stream all Tracker and Calorimeter data
  - Software Trigger for CRV based on Tracker and Calorimeter
- Alternatives?
  - Stream Calorimeter Data
  - Hardware Trigger for Tracker and CRV based on Calorimeter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

Radiation Implications

- Radiation levels at the detector will be higher than Mu2e-I
  - 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

Front-ends



Data Concentrator Layer

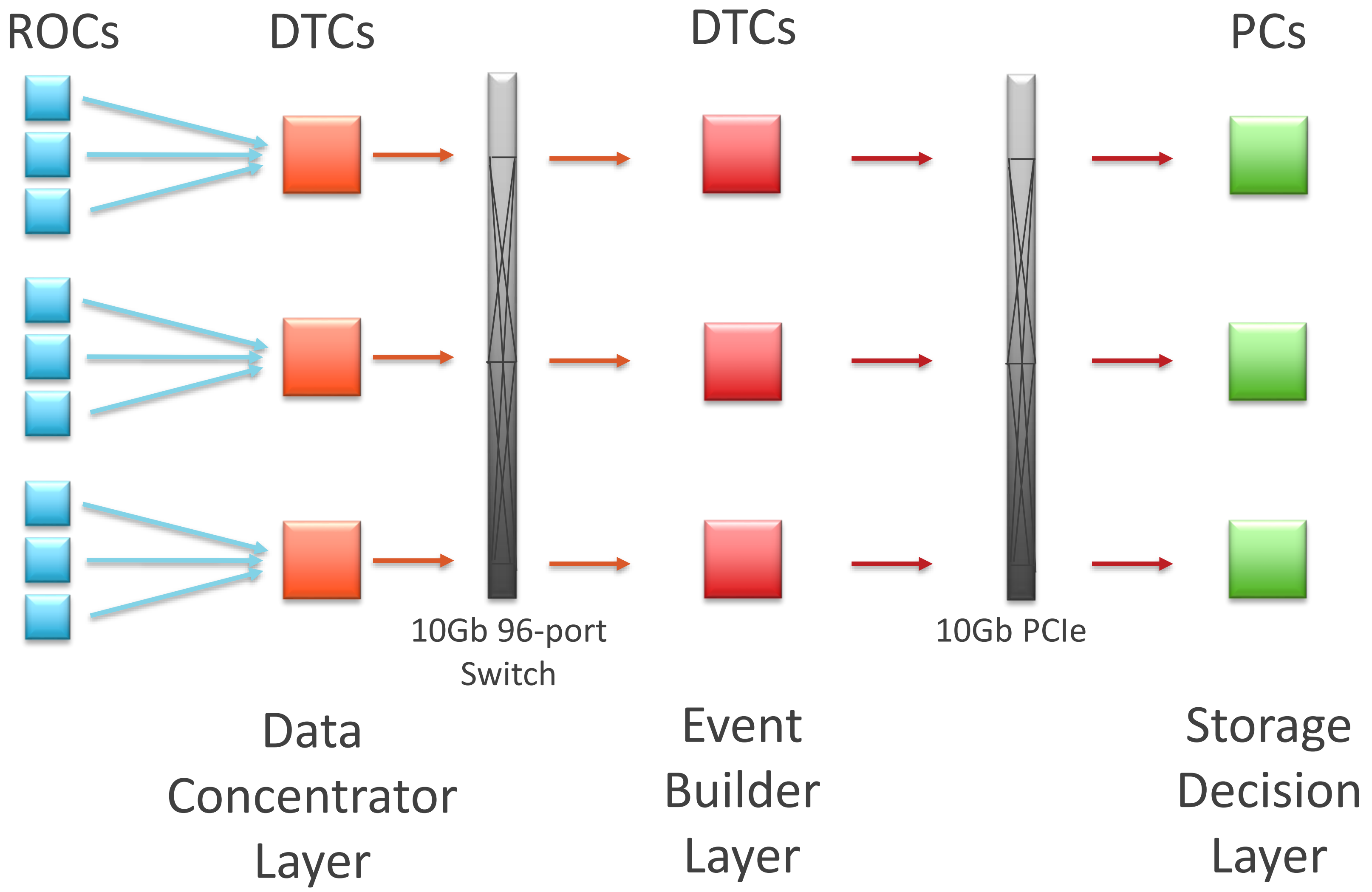
Event Builder Layer

Storage Decision Layer

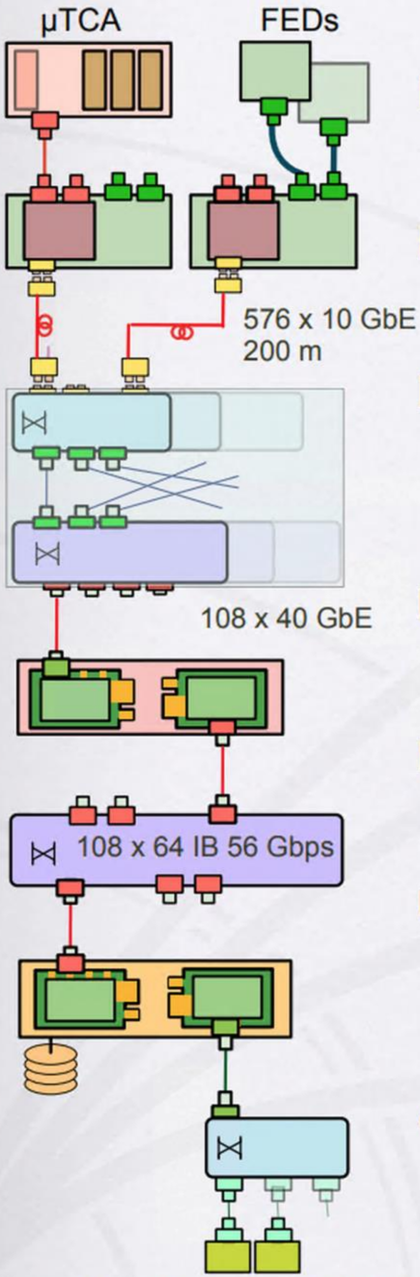
Generic Data Readout Topology

- **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



CMS Event Builder



- Detector front-end (custom electronics)
  - □ ~700 front-end drivers (FEDs) with 1–8kB/fragment at 100 kHz
- Front-End Readout Optical Link (FEROL)
  - □ Optical 10 GbE TCP/IP
- Data Concentrator switches
  - □ Data to Surface
  - □ Aggregate into 40 GbE links
- Up to 108 Readout Unit (RU) PCs
  - □ Combine 4-18 FEROL fragments into one super-fragment

Data Concentrator Layer

- Event Builder switch
  - □ Infiniband FDR 56 Gbps CLOS network
- Up to 64 Builder Unit (BU) PCs
  - □ Event building
  - □ Temporary recording to RAM disk

Event Builder Layer

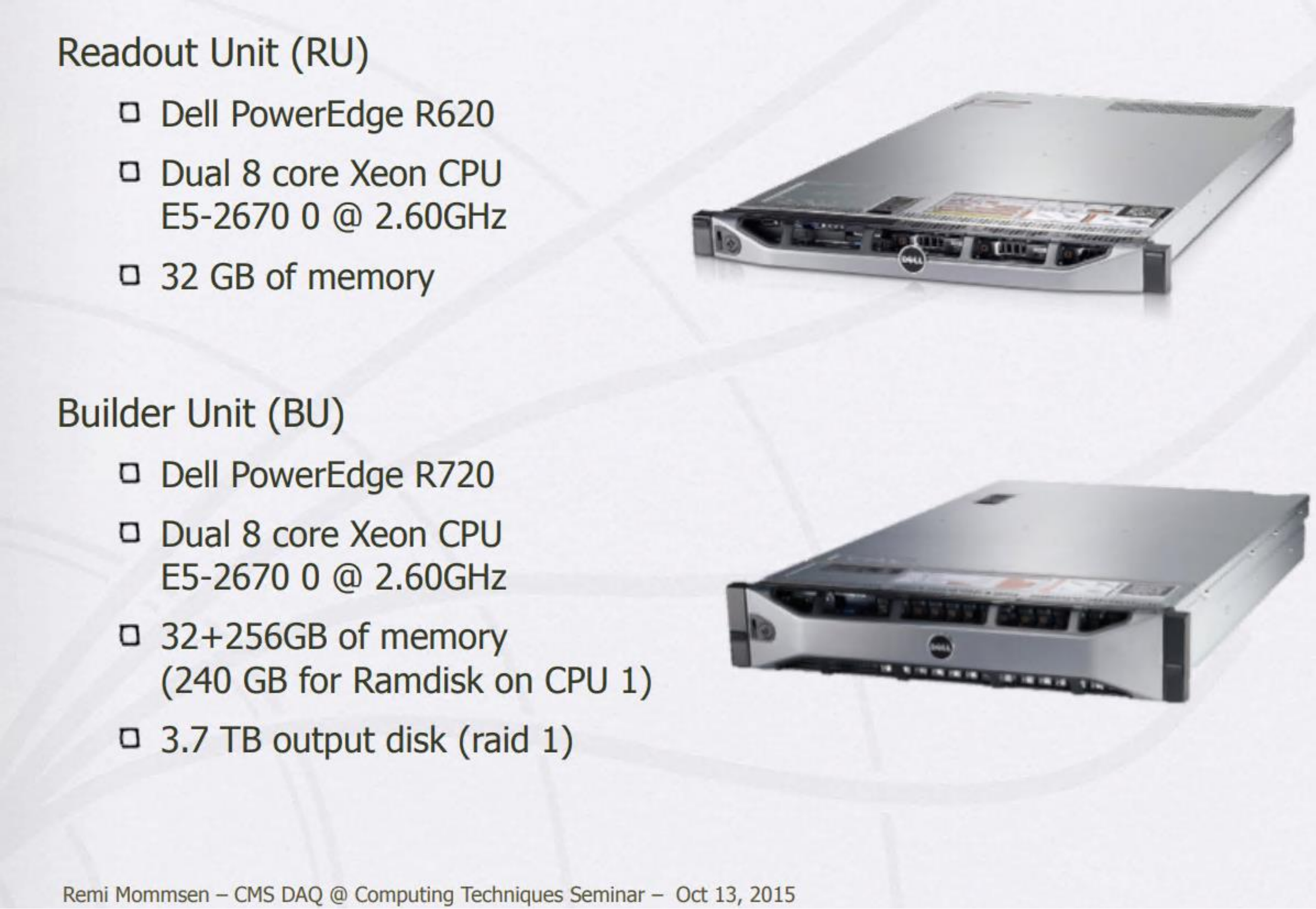
- ~900 Filter Unit (FU) PCs with ~16k cores
  - □ Run HLT selection using files from RAM disk
  - □ Select O(1%) of the events for permanent storage

Storage Decision Layer

Notes from CMS Run II Data Path

- Triggered CMS data rate of 700 x 4kB x 100kHz = 280 GBps
- Mu2e-II data rate of 38GBps x 10 = same!
  - Just wait by CERN garbage can?
- CMS slides say chose InfiniBand event building switch for cost and reliability
- Readout Unit of Concentration Layer is a PC – seems like an FPGA would be more efficient here.
- High Level Trigger reduces from 100kHz to 1kHz
  - 16K cores. For comparison, Mu2e-I plans to use 800 cores.

CMS Run II PCs (1 of 2)



Readout Unit (RU)

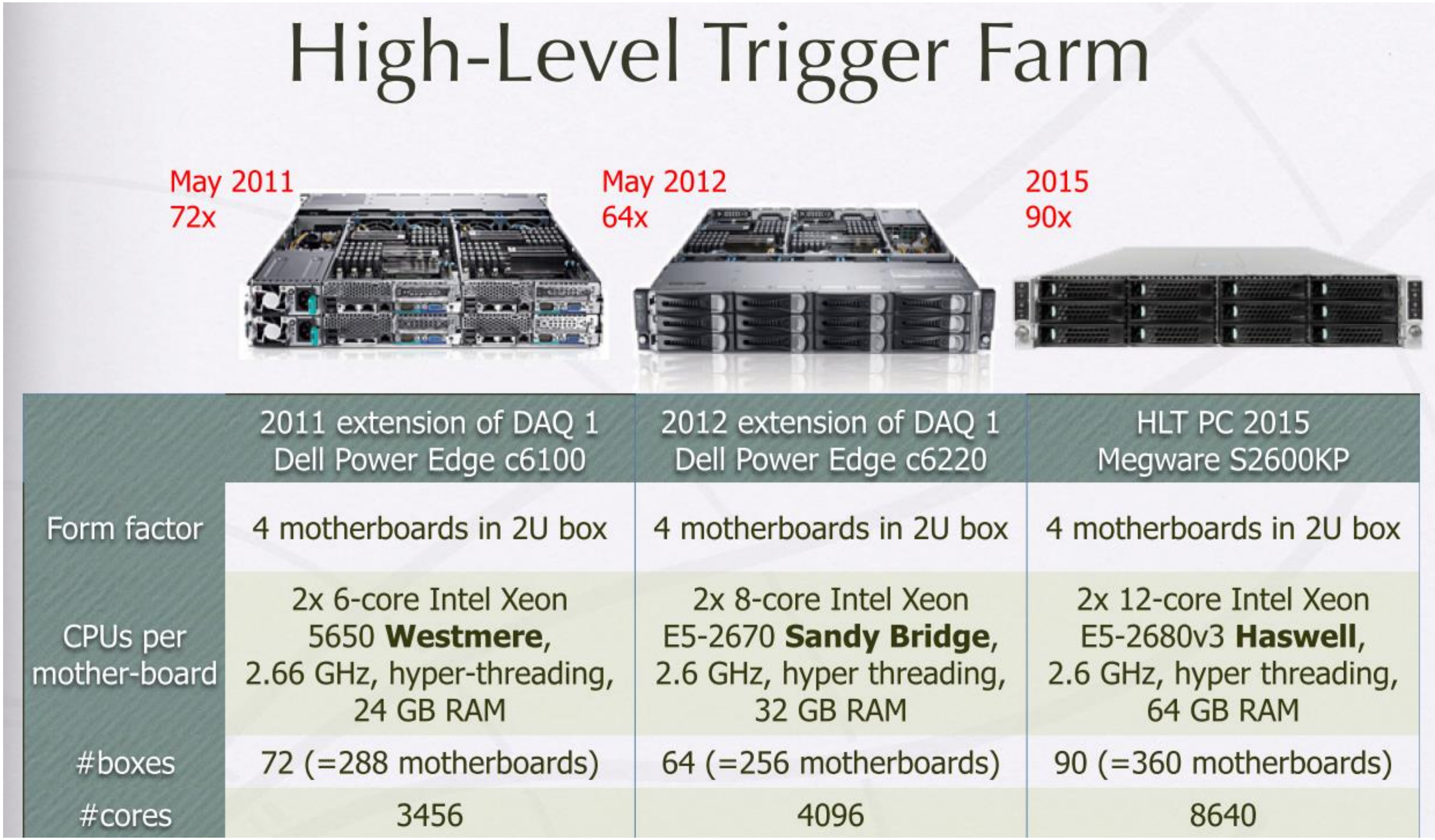
- Dell PowerEdge R620
- Dual 8 core Xeon CPU
  E5-2670 0 @ 2.60GHz
- 32 GB of memory

Builder Unit (BU)

- Dell PowerEdge R720
- Dual 8 core Xeon CPU
  E5-2670 0 @ 2.60GHz
- 32+256GB of memory
  (240 GB for Ramdisk on CPU 1)
- 3.7 TB output disk (raid 1)

Remi Mommsen – CMS DAQ @ Computing Techniques Seminar – Oct 13, 2015

CMS Run II PCs (2 of 2)



High-Level Trigger Farm

May 2011
72x

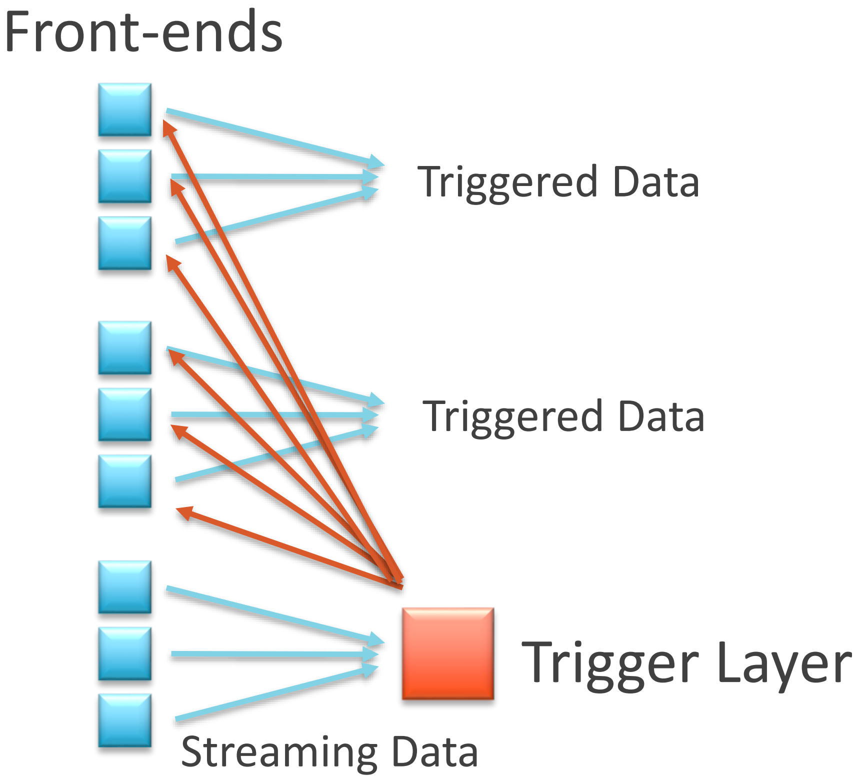
May 2012
64x

2015
90x

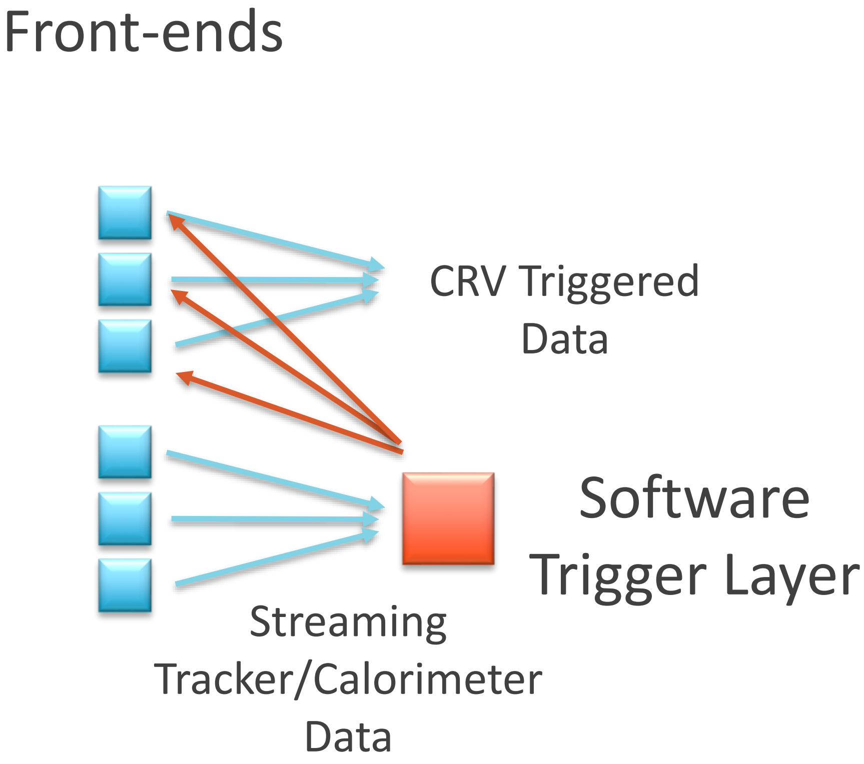
	2011 extension of DAQ 1 Dell Power Edge c6100	2012 extension of DAQ 1 Dell Power Edge c6220	HLT PC 2015 Megware S2600KP
Form factor	4 motherboards in 2U box	4 motherboards in 2U box	4 motherboards in 2U box
CPUs per mother-board	2x 6-core Intel Xeon 5650 **Westmere**, 2.66 GHz, hyper-threading, 24 GB RAM	2x 8-core Intel Xeon E5-2670 **Sandy Bridge**, 2.6 GHz, hyper threading, 32 GB RAM	2x 12-core Intel Xeon E5-2680v3 **Haswell**, 2.6 GHz, hyper threading, 64 GB RAM
#boxes	72 (=288 motherboards)	64 (=256 motherboards)	90 (=360 motherboards)
#cores	3456	4096	8640

Mu2e

Generic Trigger Path Topology



Generic Trigger Path Applied to Mu2e-I



CMS Run-II Trigger Path Notes

- Trigger input data rate is much higher (1000x?) than Mu2e-II potential trigger input data rate
  - Level-1 Trigger reduces event rate from 1GHz to 100 kHz with ~3 microseconds of fixed latency (pipelined)
  - Several FPGA trigger layers:
    - FPGA Layer to generate trigger primitives (my guess: ~20-100 boards, small to medium FPGAs < $10K each)
    - Local Trigger FPGA layer for subset of detector trigger decision accept (my guess: ~1-10 boards with large FPGAs > $10K each)
    - Global Trigger FPGA layer that takes trigger primitive objects as input and generates Level-1 accept (my guess: ~1 board with large FPGA > $10K each)

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

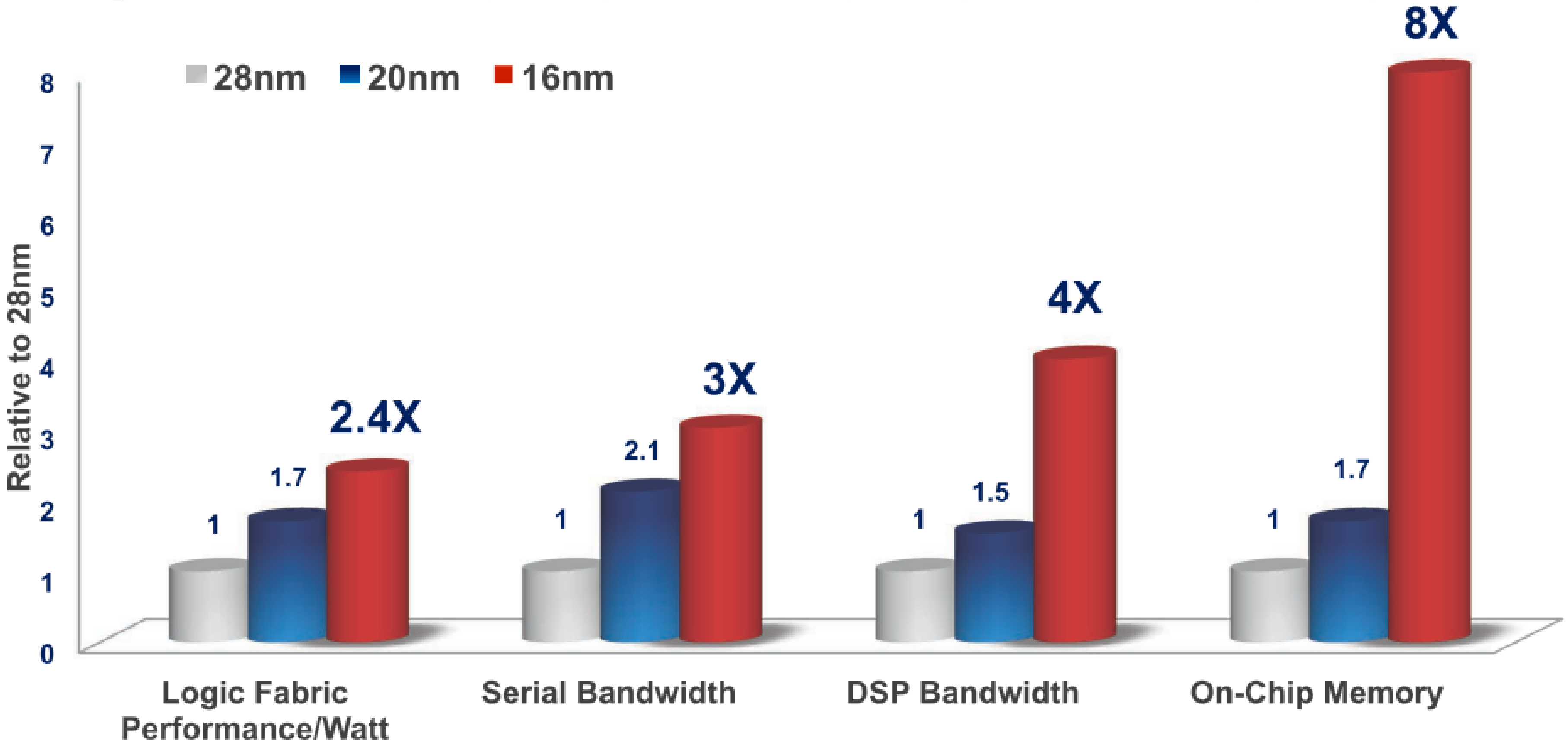
- Altera/Intel – Stratix 10
  - Up to 10 TFLOPS of single-precision floating-point DSP performance.
  - Up to 70% lower power than prior-generation high-end FPGAs
  - Up to 80 GFLOPS/Watt of single-precision floating point power efficiency.
  - Up to 144 full duplex transceivers in a single package.
  - 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**

FPGA scaling

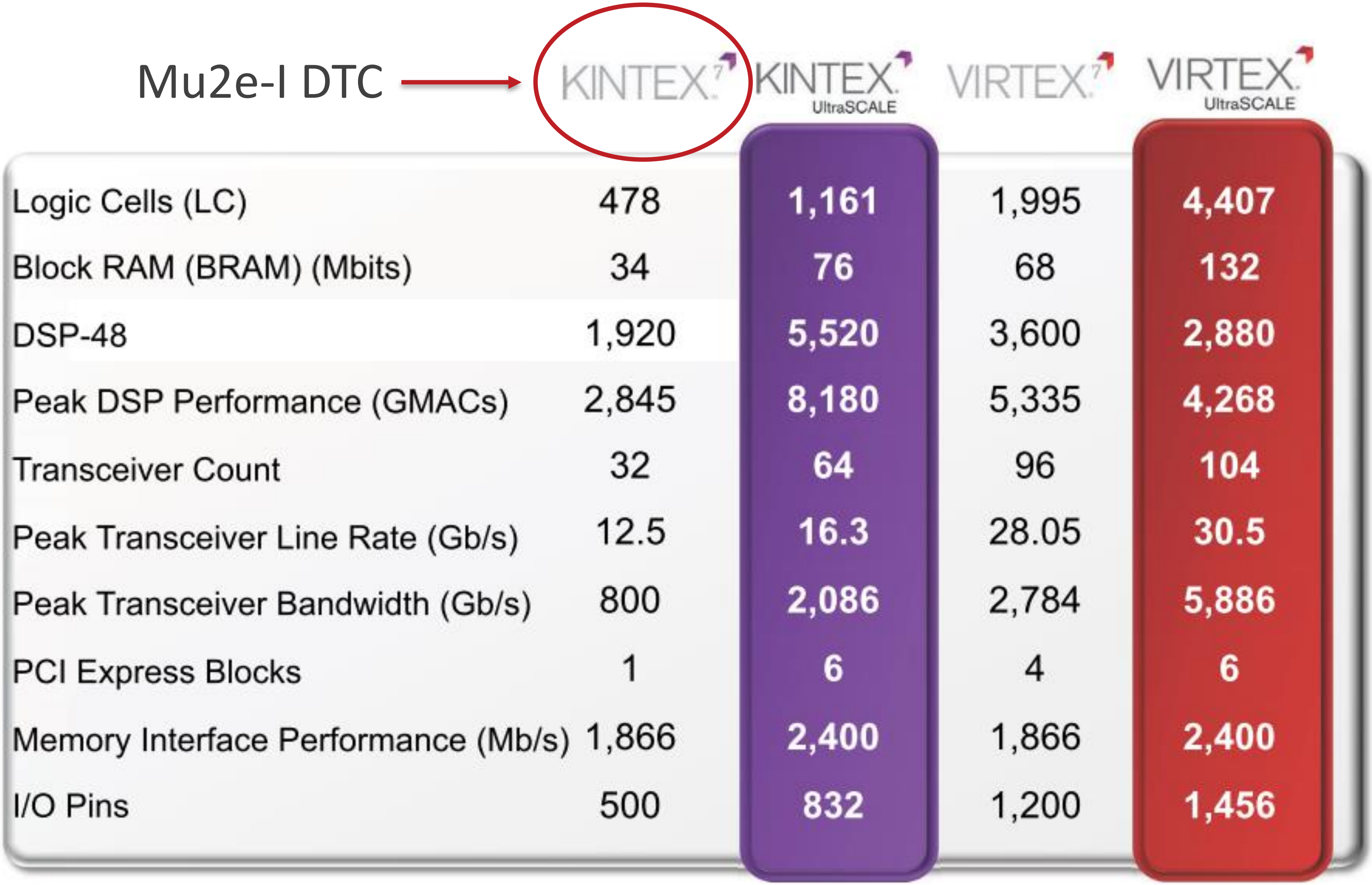
Example: Xilinx Virtex 7 (28 nm), Ultrascale (20 nm), Ultrascale + (16 nm)



Mu2e

FPGA scaling

Mu2e-I DTC

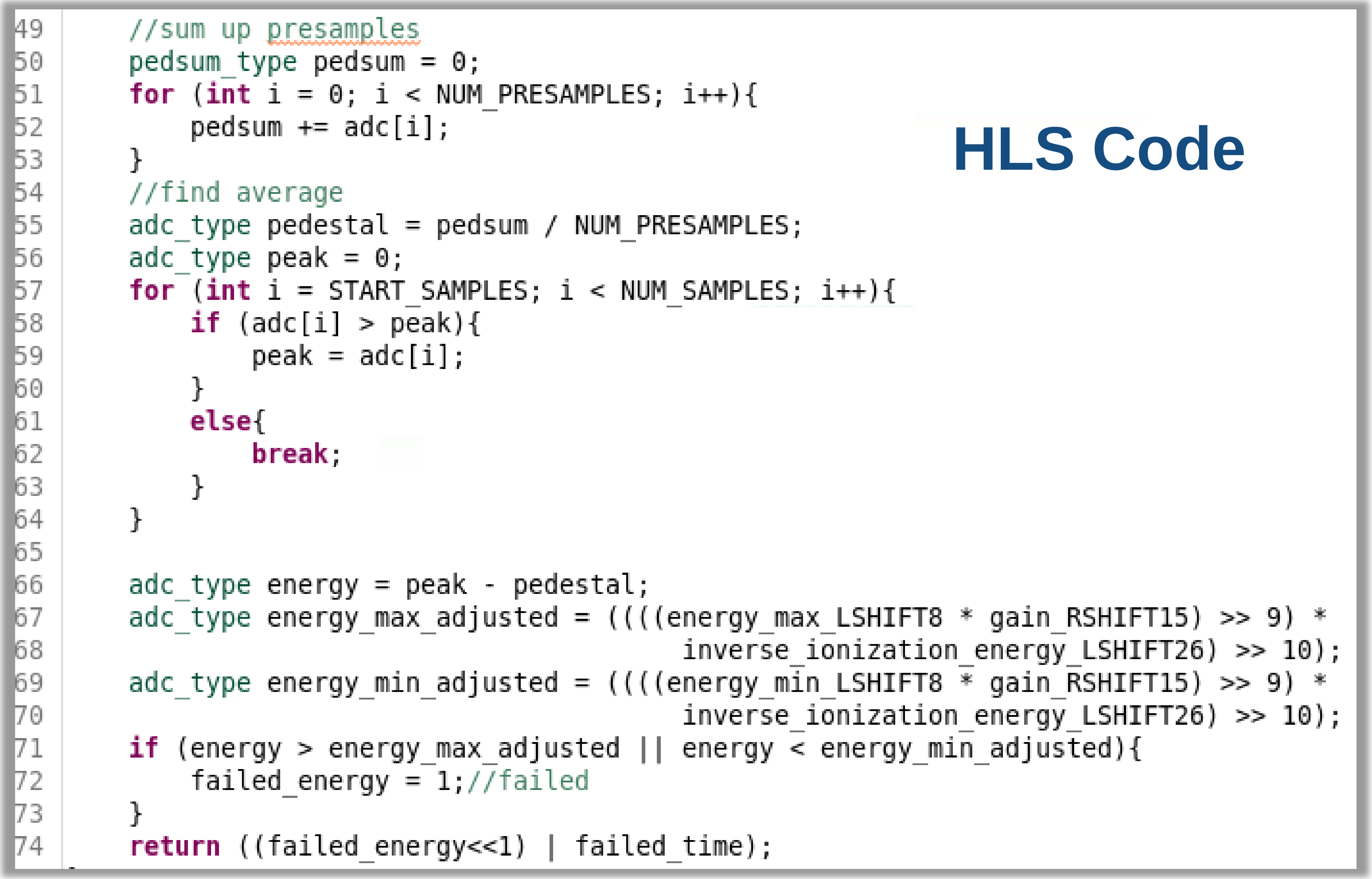


	KINTEX 7	KINTEX UltraSCALE	VIRTEX 7	VIRTEX UltraSCALE
Logic Cells (LC)	478	1,161	1,995	4,407
Block RAM (BRAM) (Mbits)	34	76	68	132
DSP-48	1,920	5,520	3,600	2,880
Peak DSP Performance (GMACs)	2,845	8,180	5,335	4,268
Transceiver Count	32	64	96	104
Peak Transceiver Line Rate (Gb/s)	12.5	16.3	28.05	30.5
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



```
    //sum up presamples
    pedsum_type pedsum = 0;
    for (int i = 0; i < NUM_PRESAMPLES; i++){
        pedsum += adc[i];
    }
    //find average
    adc_type pedestal = pedsum / NUM_PRESAMPLES;
    adc_type peak = 0;
    for (int i = START_SAMPLES; i < NUM_SAMPLES; i++){
        if (adc[i] > peak){
            peak = adc[i];
        }
        else{
            break;
        }
    }

    adc_type energy = peak - pedestal;
    adc_type energy_max_adjusted = ((((energy_max_LSHIFT8 * gain_RSHIFT15) >> 9) *
                                        inverse_ionization_energy_LSHIFT26) >> 10);
    adc_type energy_min_adjusted = ((((energy_min_LSHIFT8 * gain_RSHIFT15) >> 9) *
                                        inverse_ionization_energy_LSHIFT26) >> 10);
    if (energy > energy_max_adjusted || energy < energy_min_adjusted){
        failed_energy = 1;//failed
    }
    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

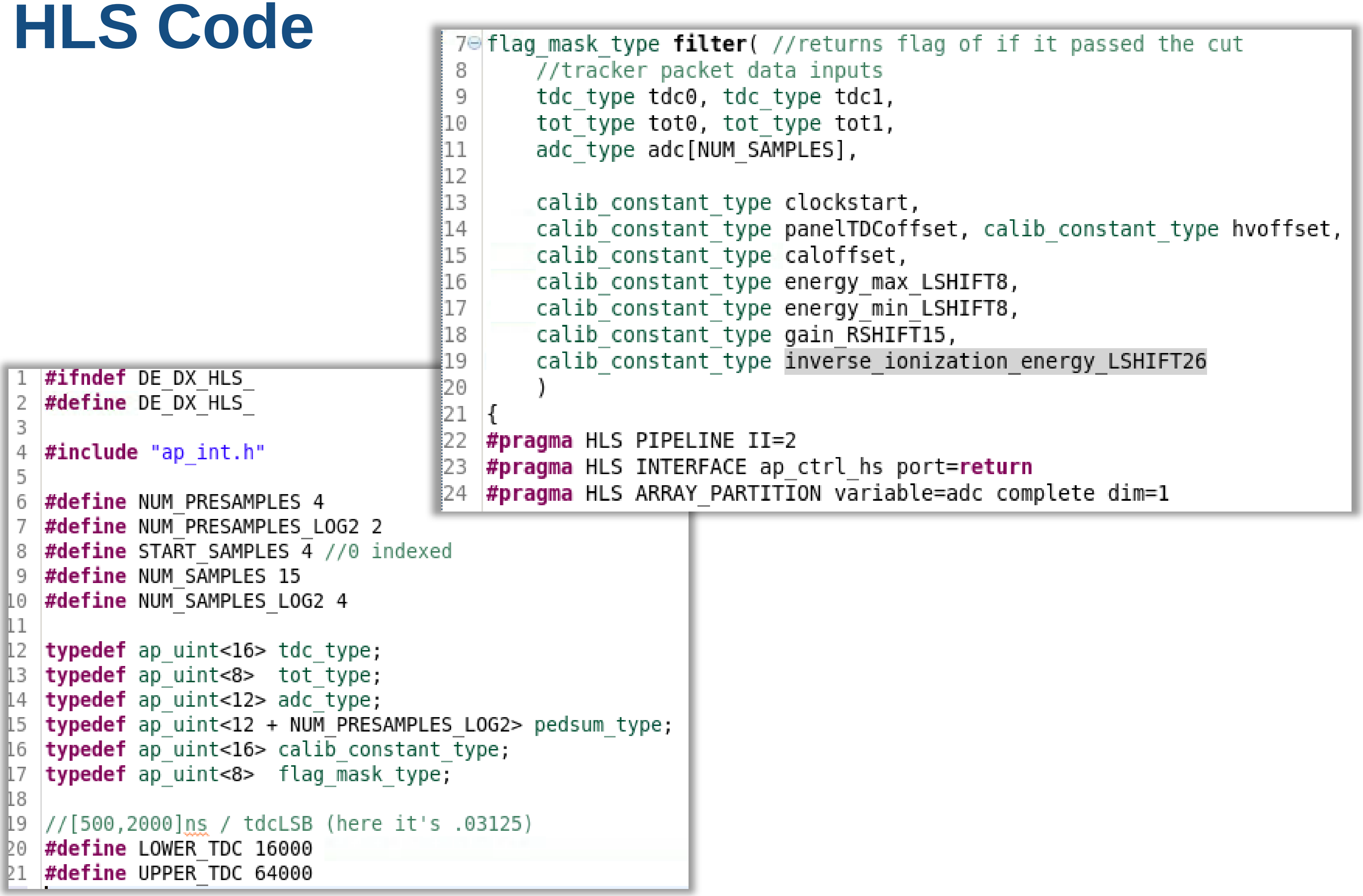
1. Which subsystems are streaming?
    a) What are the constraints imposed by rad-hard links?
2. Is it possible to have a low-latency Level-1 trigger with rejection power?
    - Lock an HLS developer and a firmware-system developer in a room for six months and tell them to understand the specs of a hardware 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?

Example Solution for 10x

- Keep same topology
- Assume gain of 2x in technology
- Buy 5x more hardware and software resources
  - Multi-stage event building switch

Backup Slides

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
```