Track Reconstruction at Level-1 in CMS for HL-LHC

Derek Cranshaw (Cornell University)
On behalf of the CMS Collaboration

Lake Louise Winter Institute
February 2022

Outline

- Motivation for L1 Track Finding
- Algorithm Description
- Strategy for Firmware Implementation
- Current Status
- Summary

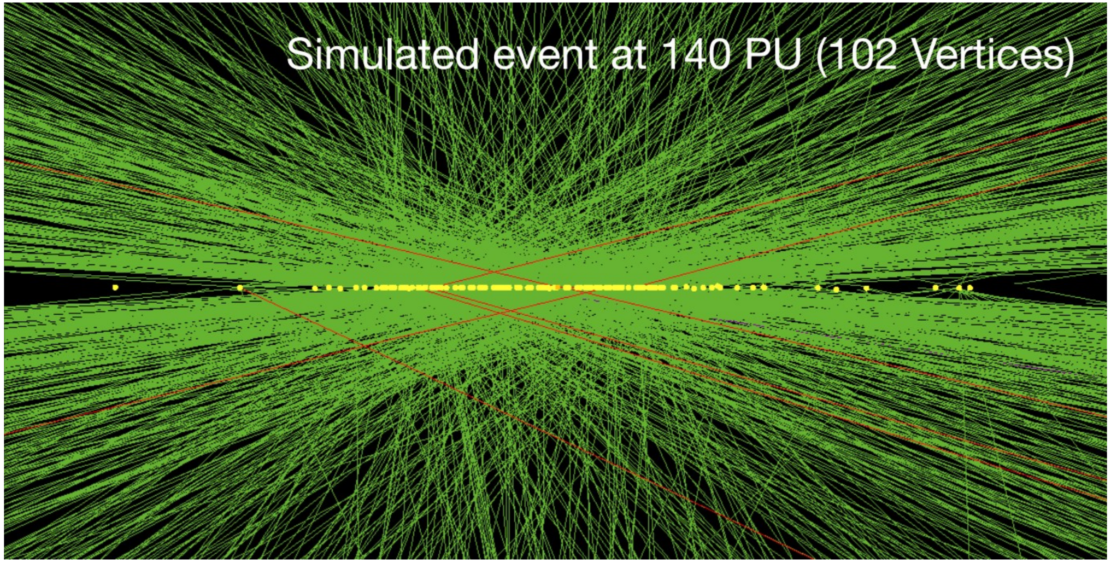
Opportunities & Challenges of the HL-LHC

- HL-LHC runs expected to deliver 3000 fb^{-1} of data
  - 10x more than LHC runs 1-3
  - Search for rare processes & constrain SM particle properties
  - Plan to start in 2029

*pileup = # proton-proton collisions per bunch crossing

- ~4x increase in pileup* – New handles needed to control trigger rates

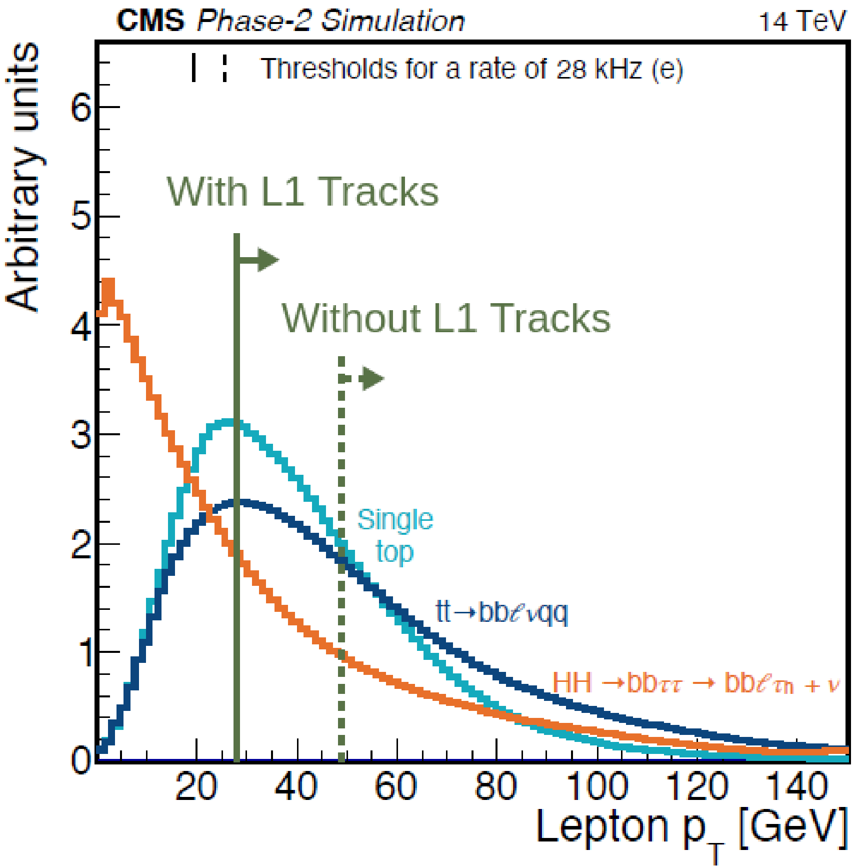
Will add tracking information to Level-1!



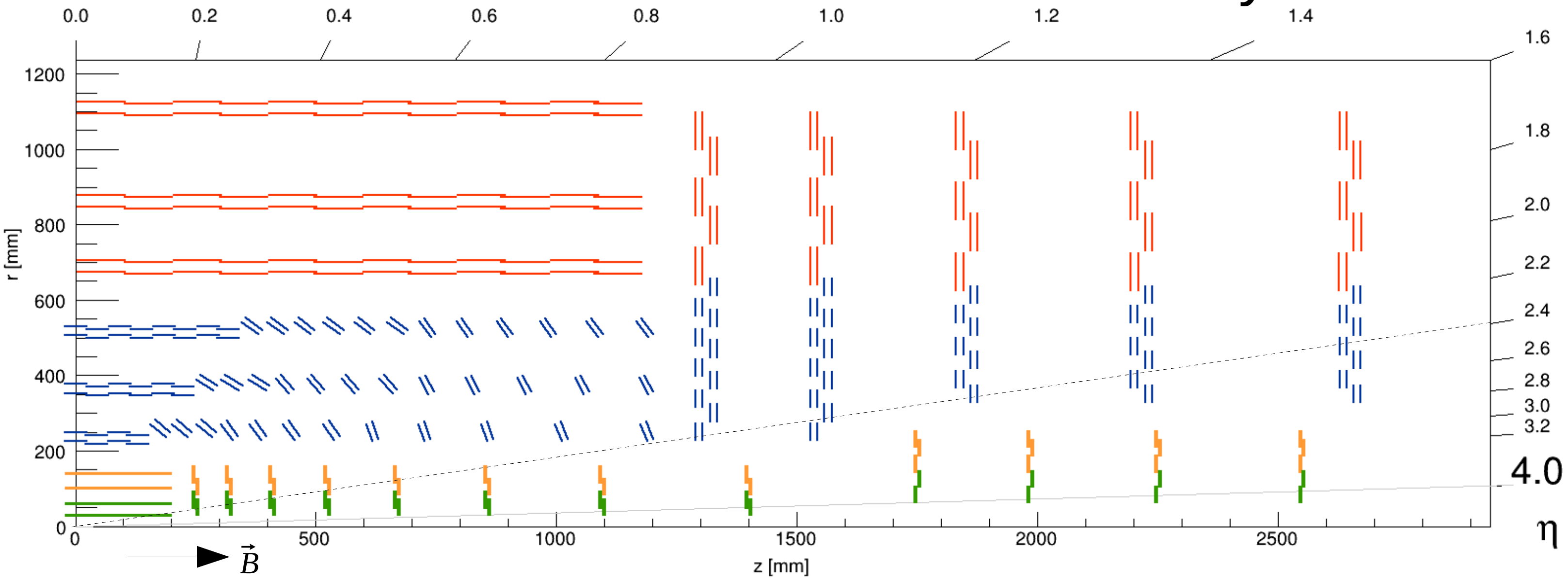
Tracking Information in L1 Trigger

- Motivation for L1 Track Finding
  - Improves p_T^{μ}, p_T^{e}, MET, and vertex reconstruction, keeping thresholds low without driving up trigger rate

- Very challenging!
  - Bunch Crossings (BX) every 25 ns
  - ~15k correlated p_T module hit pairs ('stubs') with p_T > 2 GeV
    - ~200 tracks to reconstruct per BX
  - 4 μs budgeted for track finding



Tracker Geometry

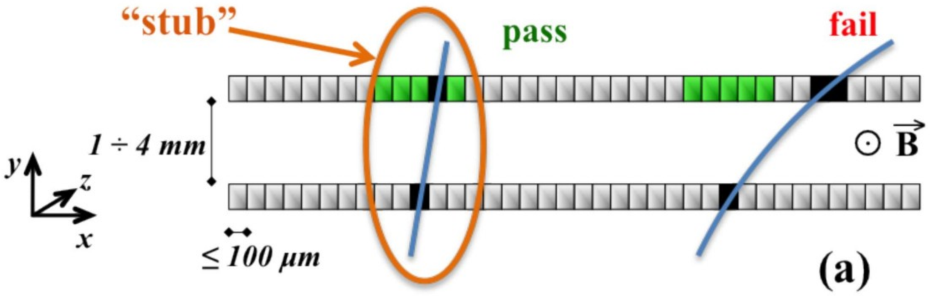


Red & Blue:
Outer Tracker
(Used in L1)

Green & Orange:
Inner Tracker
(Not used in L1)

- Cylindrical shape - 6 ‘barrel’ layers, 5 ‘endcap’ disks per side
- L1 Tracking out to |η|<2.4

- ‘p_T modules’ - Two closely spaced sensors, correlates hits on common front-end ASIC
  - Reject hits from low-p_T tracks
  - Reduced data by ~10x-20x
    Necessary for track finding at 40 MHz!



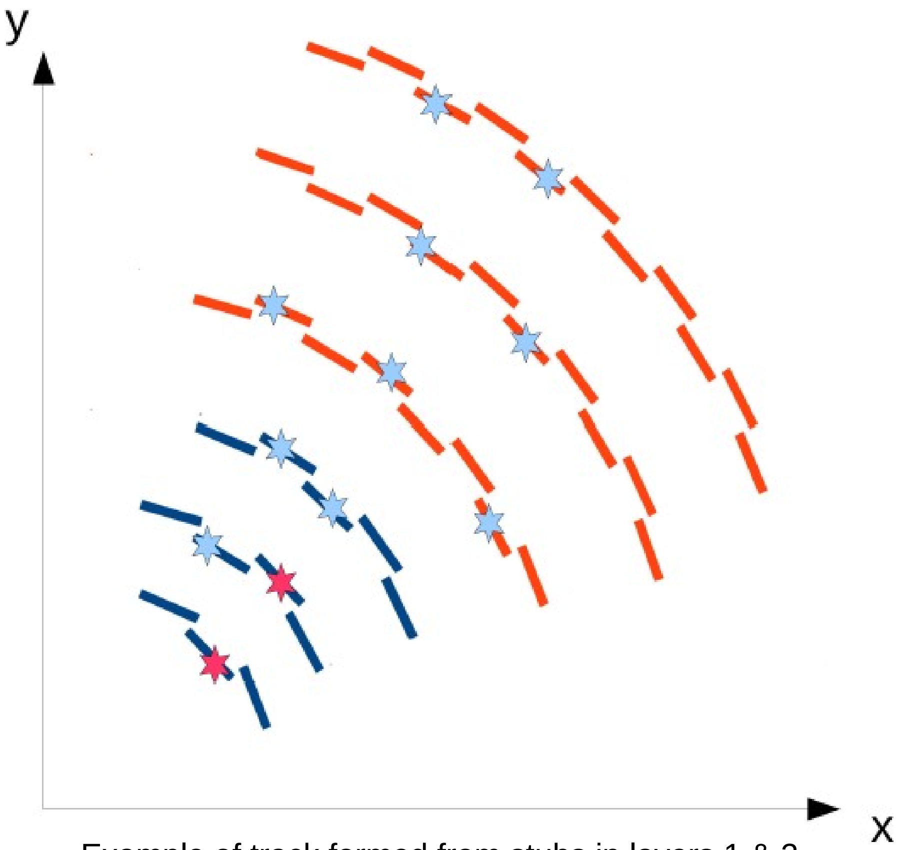
Track Finding Overview

- <u>Track Finding Strategy – Road Search*</u>
  - Naturally pipelined
  - Modest system size
  - Simple software emulation

- <u>Algorithm description:</u>
  1. Stub pairs form 'Tracklets'
  2. Tracklet projects to other layers
  3. Match stubs to projections
  4. Refine track using Kalman Filter

- Classic road search style algorithm
  - *<u>Challenge is to implement on FPGA!</u>*

(FPGA: Field Programmable Gate Array)



Example of track formed from stubs in layers 1 & 2 (L1L2 "seed") with projections in the barrel

*Detailed in Tracklet Paper

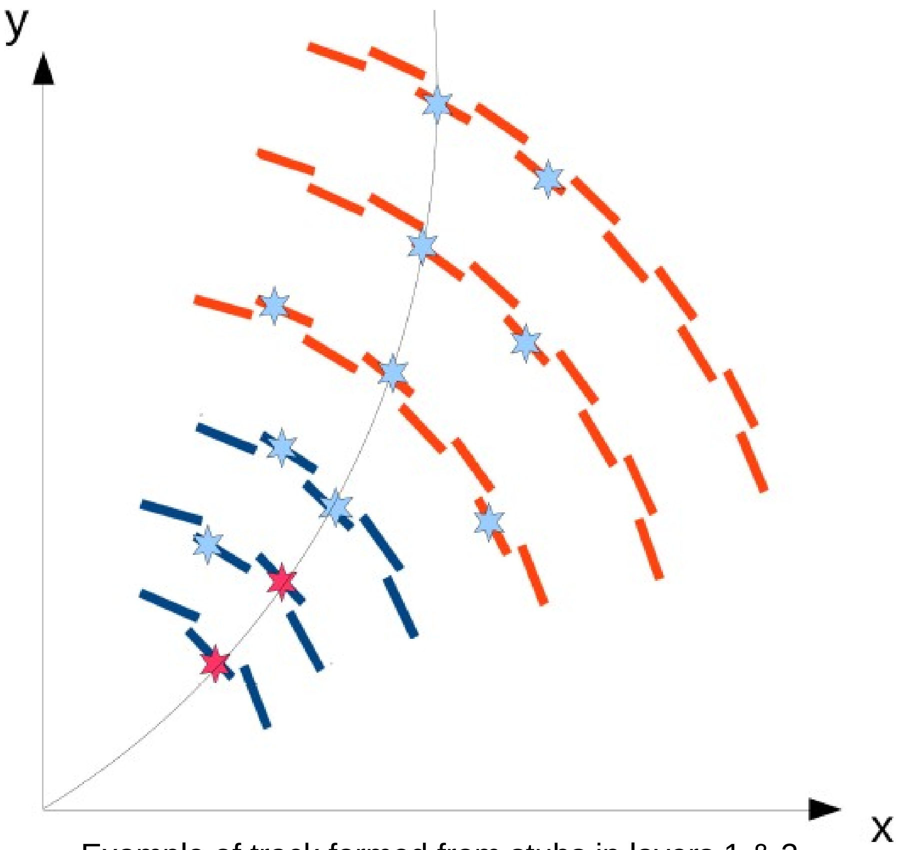
Track Finding Overview

- <u>Track Finding Strategy – Road Search*</u>
  - Naturally pipelined
  - Modest system size
  - Simple software emulation

- <u>Algorithm description:</u>
  1. Stub pairs form ‘Tracklets’
  2. Tracklet projects to other layers
  3. Match stubs to projections
  4. Refine track using Kalman Filter

- Classic road search style algorithm
  - *<u>Challenge is to implement on FPGA!</u>*

(FPGA: Field Programmable Gate Array)



Example of track formed from stubs in layers 1 & 2 (L1L2 “seed”) with projections in the barrel

*Detailed in Tracklet Paper

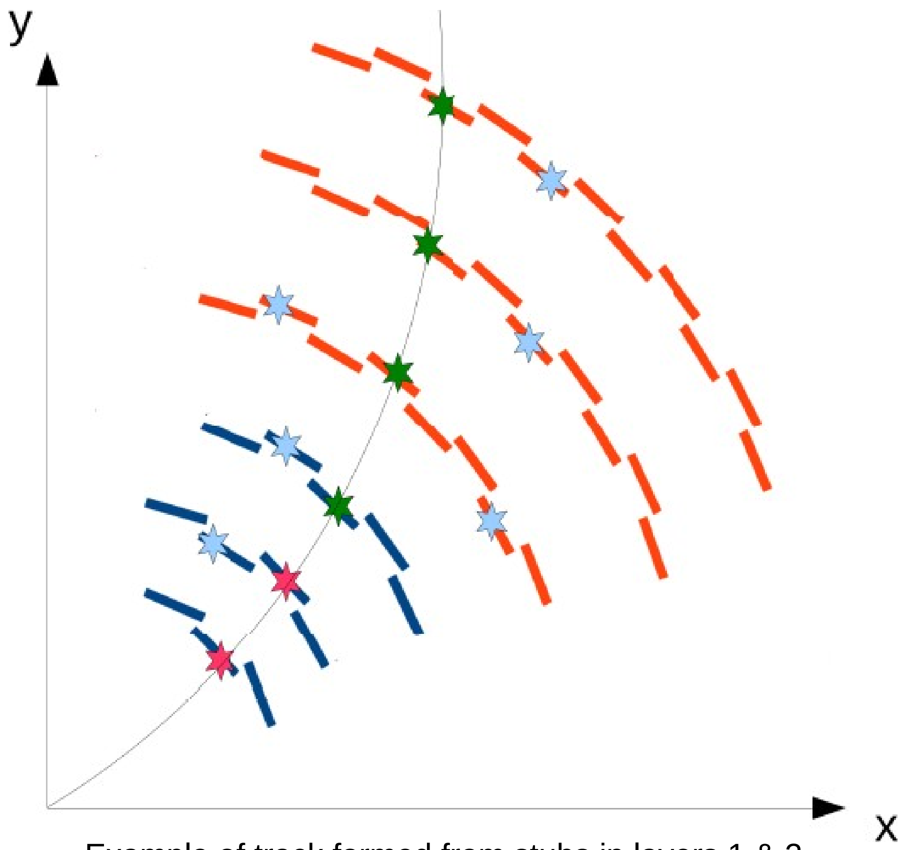
Track Finding Overview

- <u>Track Finding Strategy – Road Search*</u>
  - Naturally pipelined
  - Modest system size
  - Simple software emulation

- <u>Algorithm description:</u>
  1. Stub pairs form 'Tracklets'
  2. Tracklet projects to other layers
  3. Match stubs to projections
  4. Refine track using Kalman Filter

- Classic road search style algorithm
  - *<u>Challenge is to implement on FPGA!</u>*

(FPGA: Field Programmable Gate Array)



Example of track formed from stubs in layers 1 & 2 (L1L2 “seed”) with projections in the barrel

*Detailed in Tracklet Paper

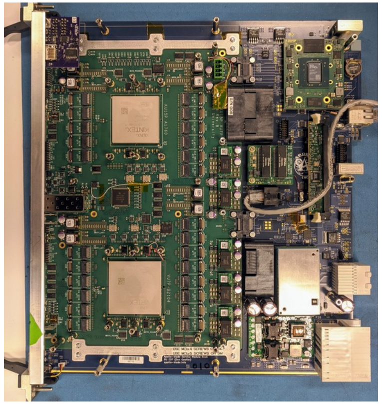
Track Finding Overview

- Track Finding Strategy – Road Search*
  - Naturally pipelined
  - Modest system size
  - Simple software emulation

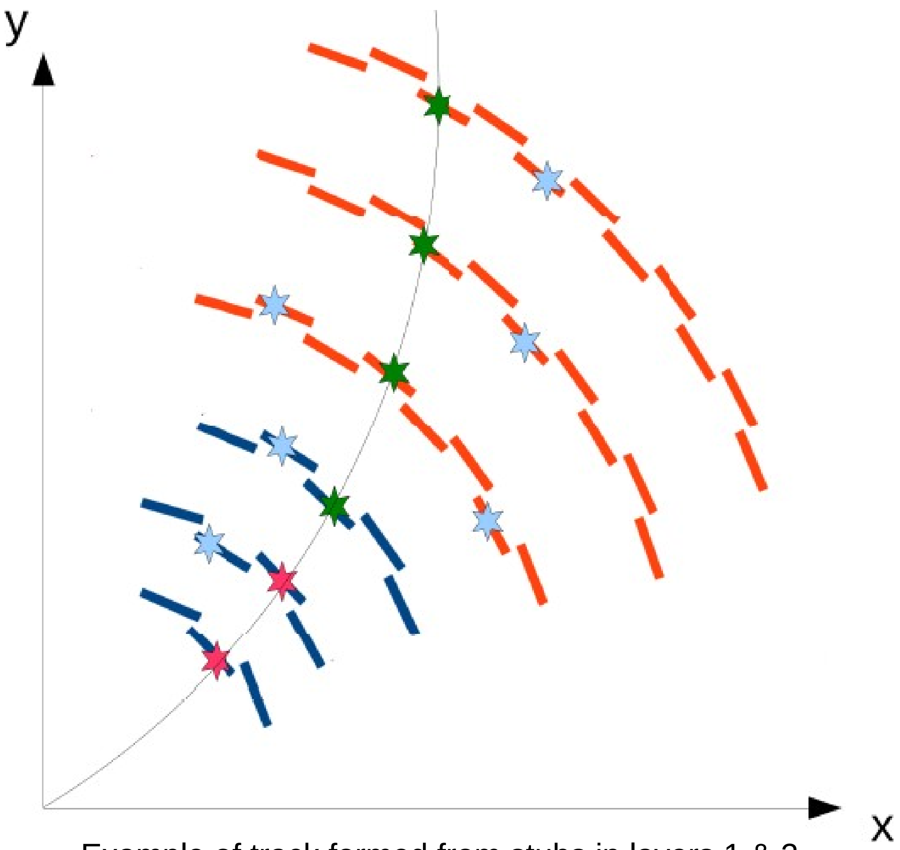
- Algorithm description:
  1. Stub pairs form 'Tracklets'
  2. Tracklet projects to other layers
  3. Match stubs to projections
  4. Refine track using Kalman Filter

- Classic road search style algorithm
  - *Challenge is to implement on FPGA!*

(FPGA: Field Programmable Gate Array)



'Apollo' platform being built, details in this paper and talk

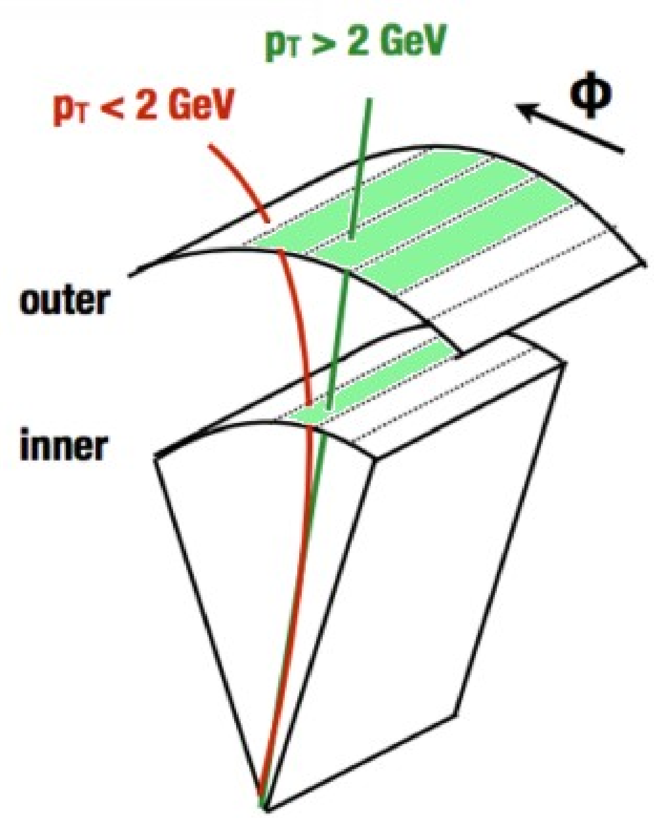


Example of track formed from stubs in layers 1 & 2 (L1L2 "seed") with projections in the barrel

*Detailed in Tracklet Paper

Reducing Combinatorics – Virtual Modules

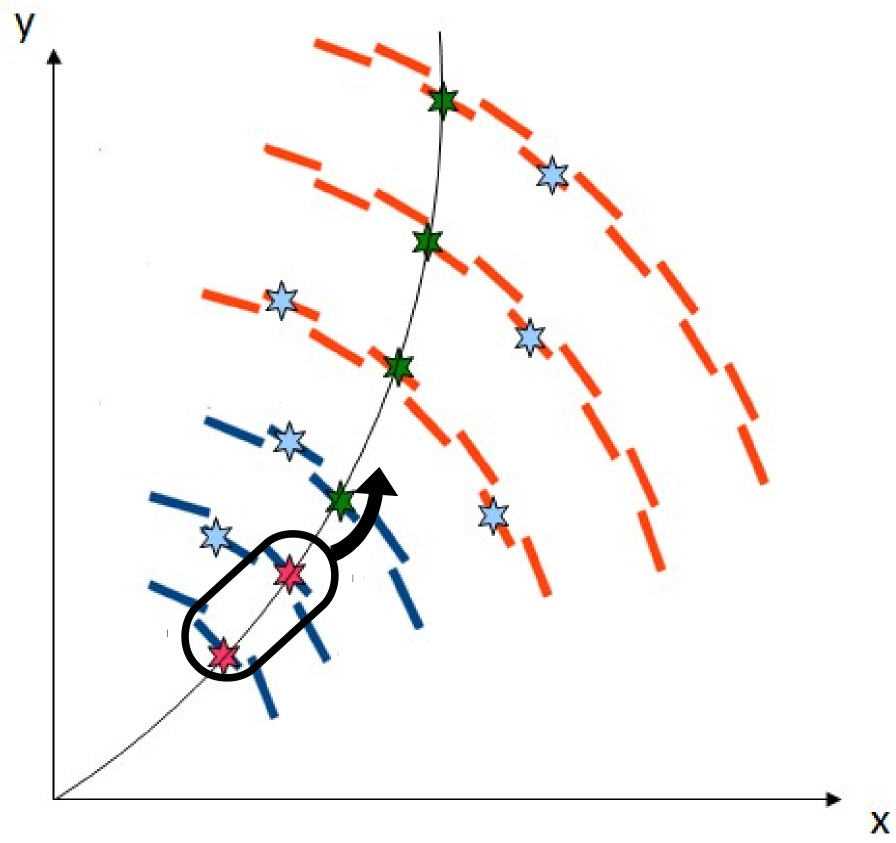
- Many stubs/BX – cannot consider all stub pairs
- Tracker divided into ϕ slices. Only consider slice pairs that produce tracks ≥ 2 GeV
  - Exploit FPGA resources by processing pairs in parallel
  - Greatly reduces total stub pairs considered



Removing Duplicate Tracks

- Tracklets formed in several layers/disks combinations (‘seeds’)
  - Ensures good efficiency over η
  - Different seeds can find same track
  - Two nearby stubs can make similar tracks

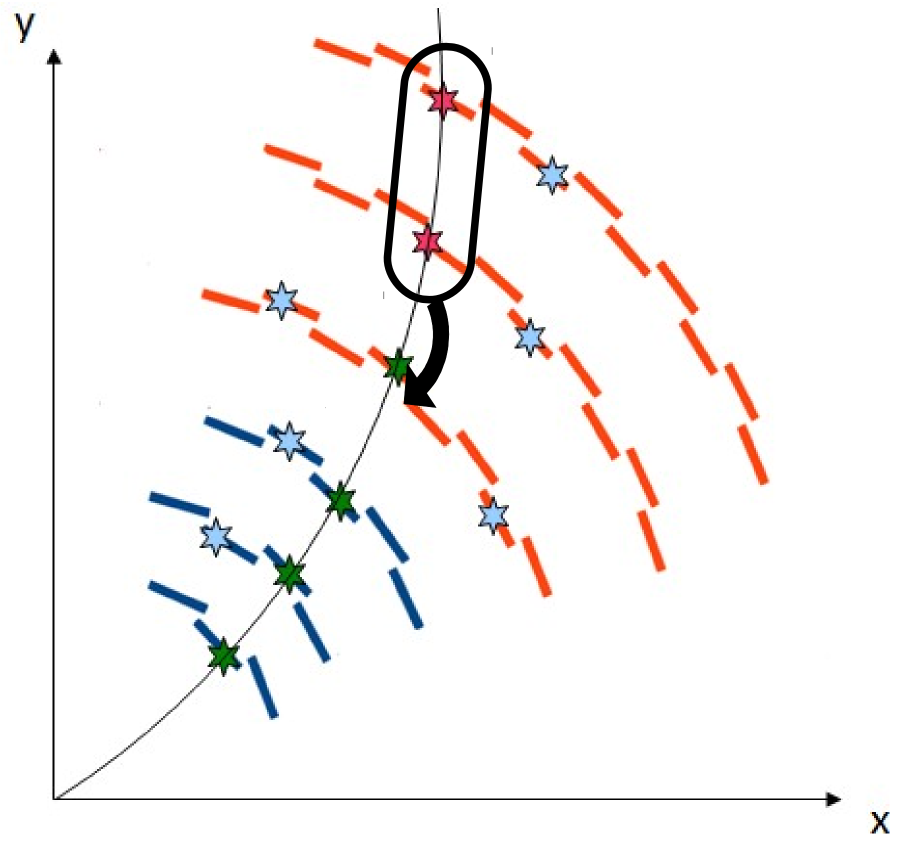
- Want to use Kalman Filter to obtain best possible tracks
  - Merge stub lists of duplicate candidates
  - More thorough exploration of track possibilities by KF than removing a track



Removing Duplicate Tracks

- Tracklets formed in several layers/disks combinations (‘seeds’)
  - Ensures good efficiency over η
  - Different seeds can find same track
  - Two nearby stubs can make similar tracks

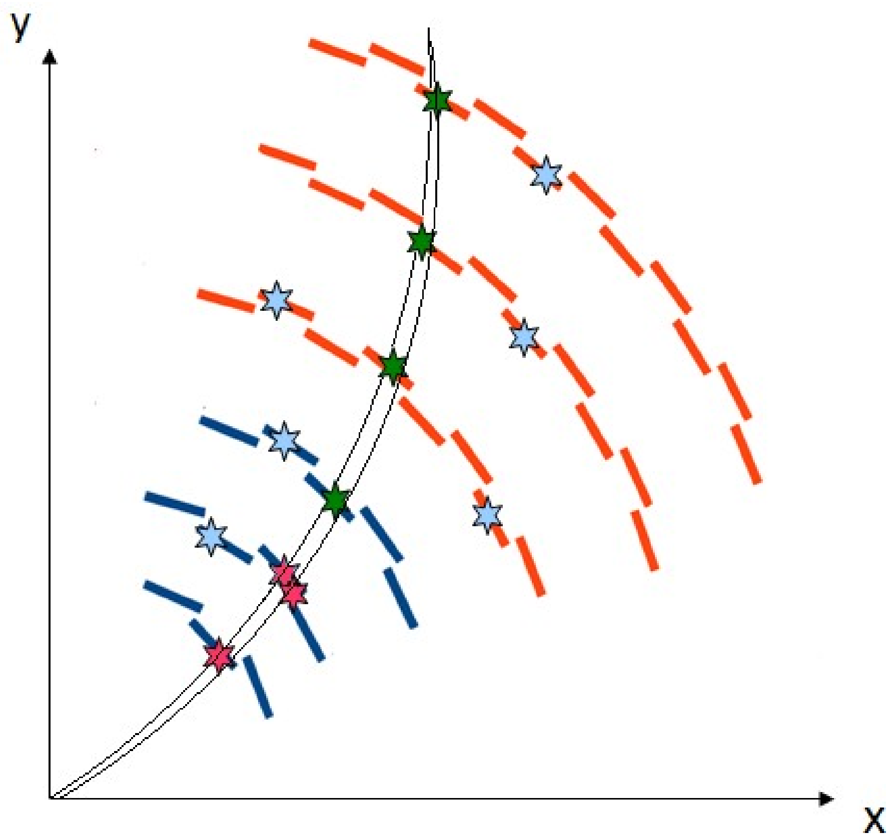
- Want to use Kalman Filter to obtain best possible tracks
  - Merge stub lists of duplicate candidates
  - More thorough exploration of track possibilities by KF than removing a track



Removing Duplicate Tracks

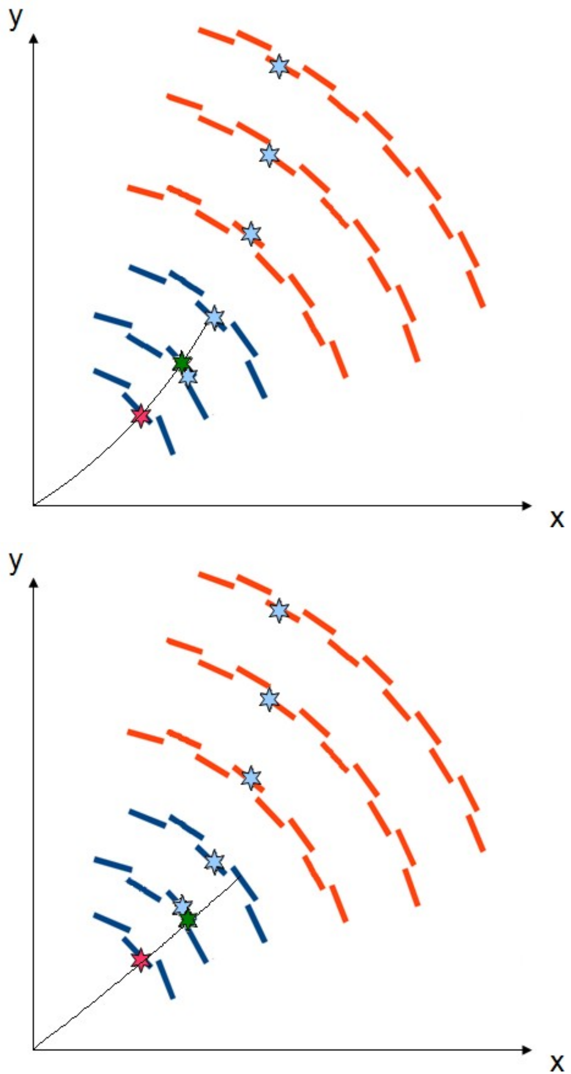
- Tracklets formed in several layers/disks combinations (‘seeds’)
  - Ensures good efficiency over η
  - Different seeds can find same track
  - Two nearby stubs can make similar tracks

- Want to use Kalman Filter to obtain best possible tracks
  - Merge stub lists of duplicate candidates
  - More thorough exploration of track possibilities by KF than removing a track



Track Fitting - Kalman Filter*

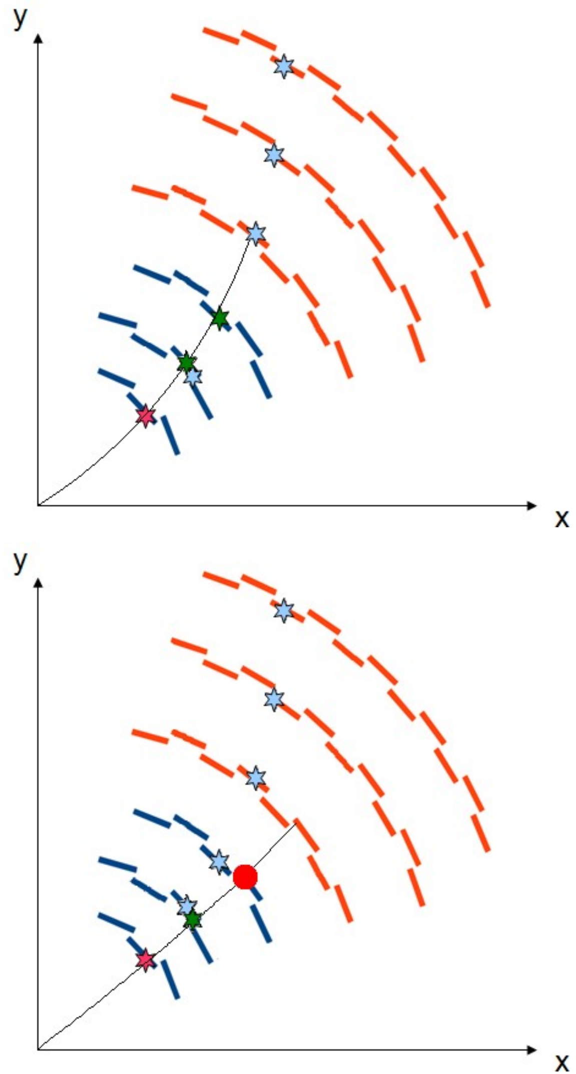
1. Take track from previous steps
2. Iteratively add stubs, updating track each step
3. If ≥ 2 stub per layer on track → calculate multiple projections
4. Too many layers missed → track discarded
   - KF selects best stubs & refines track parameters



*Detailed in HT-KF Paper

Track Fitting - Kalman Filter*

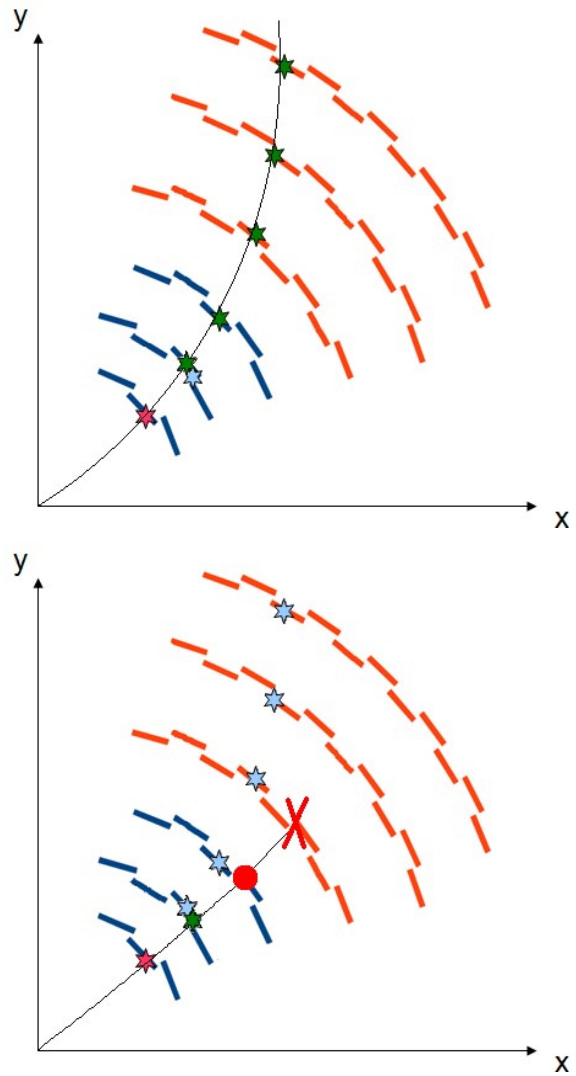
1. Take track from previous steps
2. Iteratively add stubs, updating track each step
3. If ≥ 2 stub per layer on track → calculate multiple projections
4. Too many layers missed → track discarded
   - KF selects best stubs & refines track parameters



*Detailed in HT-KF Paper

Track Fitting - Kalman Filter*

1. Take track from previous steps
2. Iteratively add stubs, updating track each step
3. If ≥ 2 stub per layer on track → calculate multiple projections
4. Too many layers missed → track discarded
   - KF selects best stubs & refines track parameters



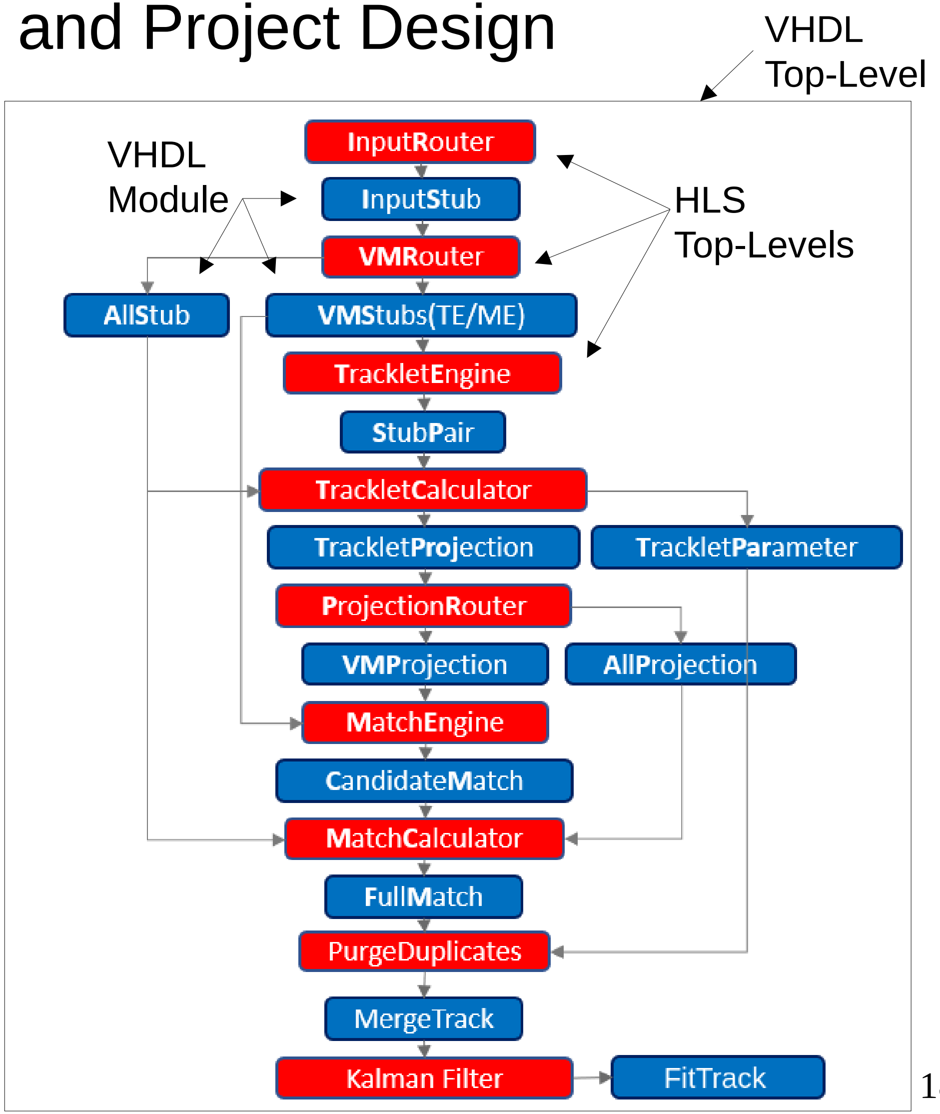
*Detailed in HT-KF Paper

High Level Synthesis (HLS)

- Previous iterations written in Verilog – Steep learning curve
- Switch to HLS – Allows programmer to specify firmware logic in a high-level language (C++ for us).
  - Faster & easier development of FW logic
- HLS is a useful tool, but has certain drawbacks
  - HLS-specific syntax constraints
  - More difficult to debug
    - Switching to HLS greatly simplified firmware development & maintenance!

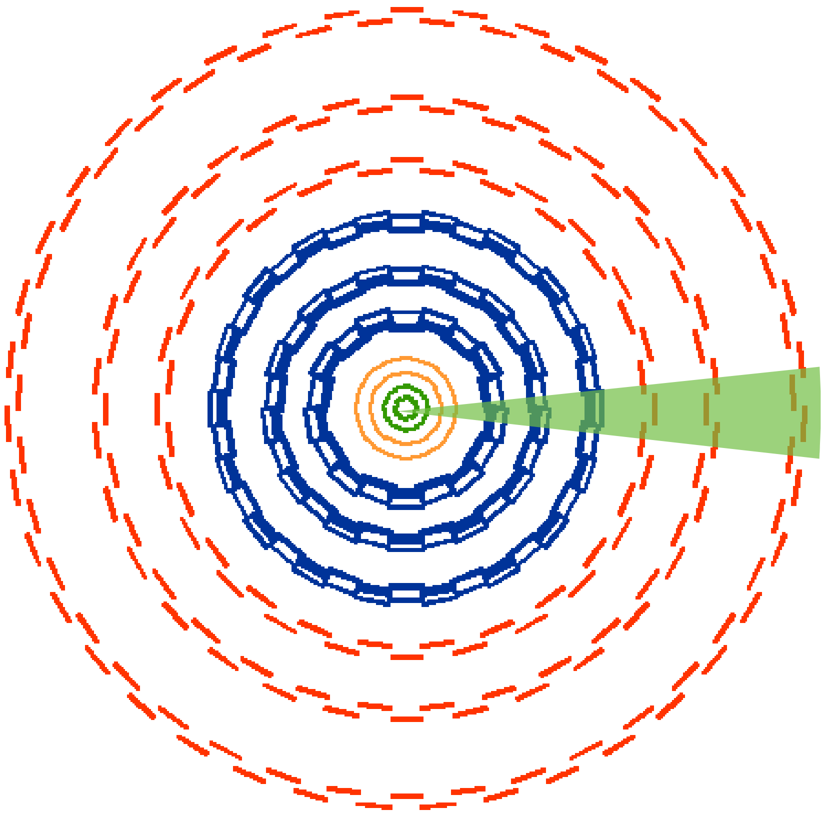
Algorithm Structure and Project Design

- 9 processing steps (red), 14 block RAMs (blue)
  - Each step is its own HLS function
  - Independently developed
- All steps successfully implemented & tested
  - CI ensures continuous validation of modules during development
- Many instantiations of HLS blocks wired up in top-level VHDL file
  - Current goal is to realize full end-to-end chain for narrow slice in ϕ



Near-term Goal - “Skinny” Chain

- Full forward & backward expansion around a single module
  - ~4% of the full project
  - Allows full demonstration of track finding chain
- Currently being tested in Modelsim over 1k $t\bar{t}$+200 pileup events

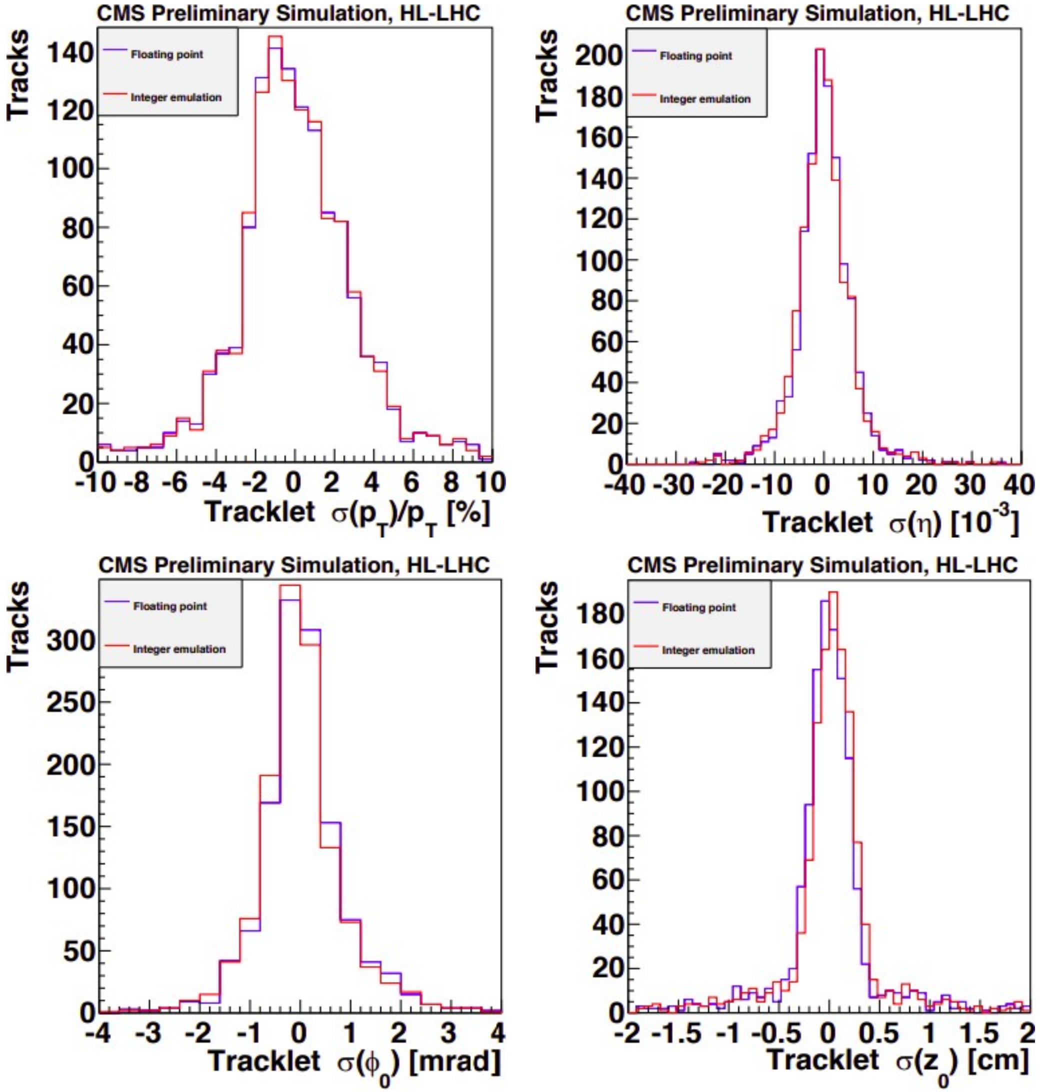


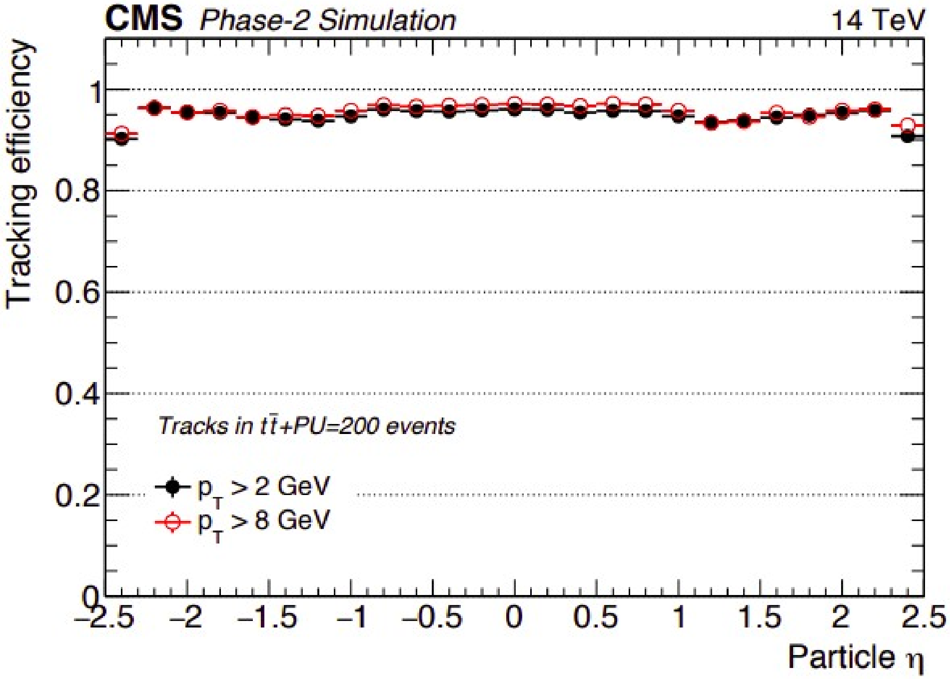
Summary

- Tracking information at L1 helps to maintain physics performance under high pileup
- Algorithm combines road search style track finding with Kalman Filter fit
  - Firmware developed in combination of VHDL and HLS
- On track to deliver tracking for CMS L1 for HL-LHC
  - All HLS module successfully synthesized & tested
  - Full end-to-end chain written & being tested
- Next step – scale up to full tracker

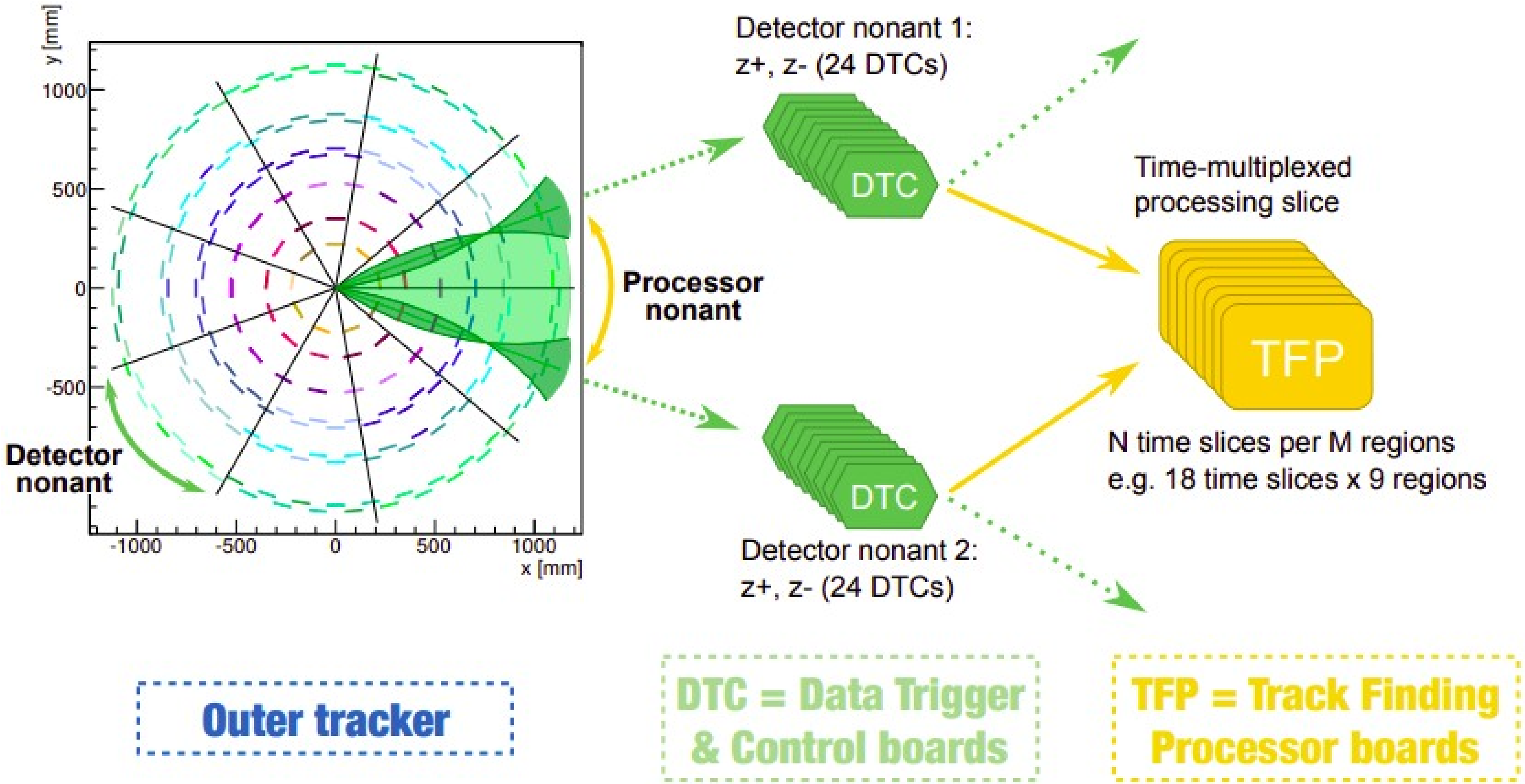
BACKUP

Efficiency & Resolution





Full System Architecture



*Slide by L. Skinnari