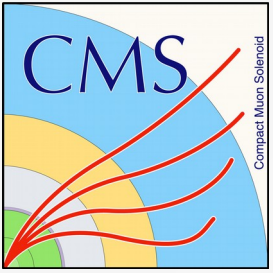
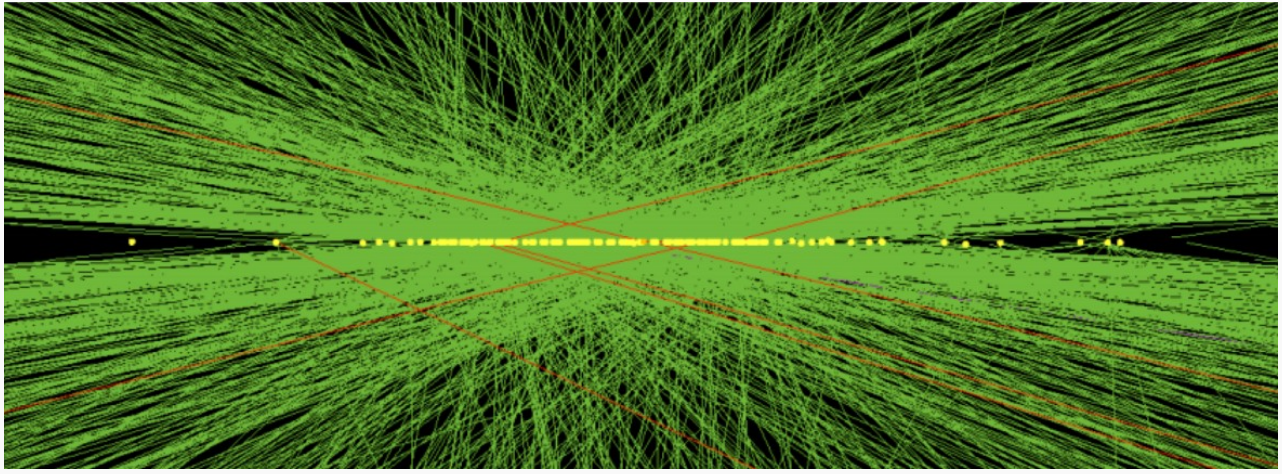


Level 1 Track Finder at CMS

Andrew Hart for the CMS Collaboration
Rutgers, The State University of New Jersey

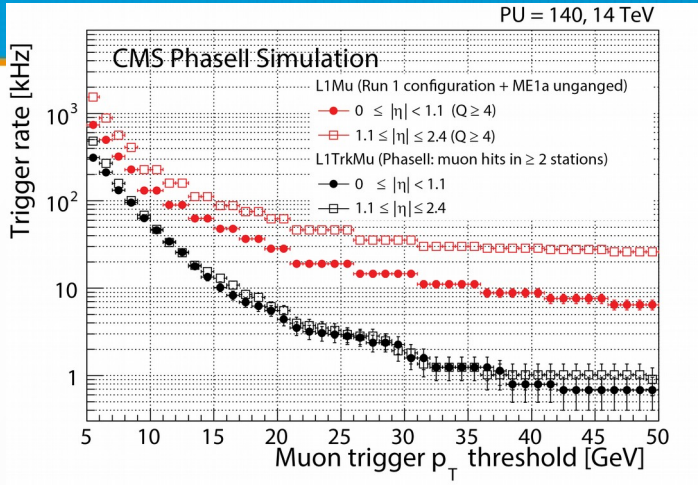
The High-Luminosity LHC

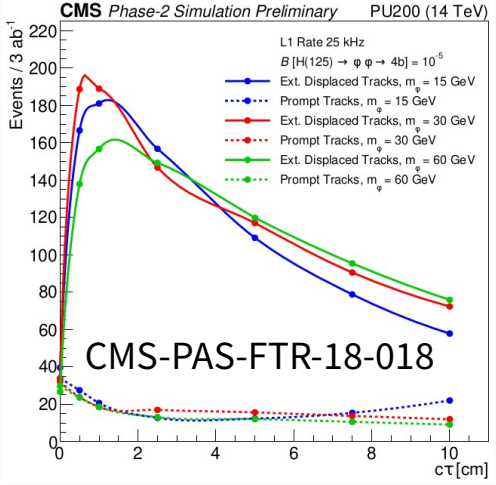
- The High-Luminosity LHC (HL-LHC) is expected to achieve luminosities up to 5×10^{34} cm^{-2} s^{-1} (7.5×10^{34} cm^{-2} s^{-1} in ultimate performance scenario):
  - Pileup at the level of 140-200 interactions per bunch crossing
- Great opportunities for physics, but very challenging for data analysis
- How do we optimize CMS to ensure we get the most out of this data?

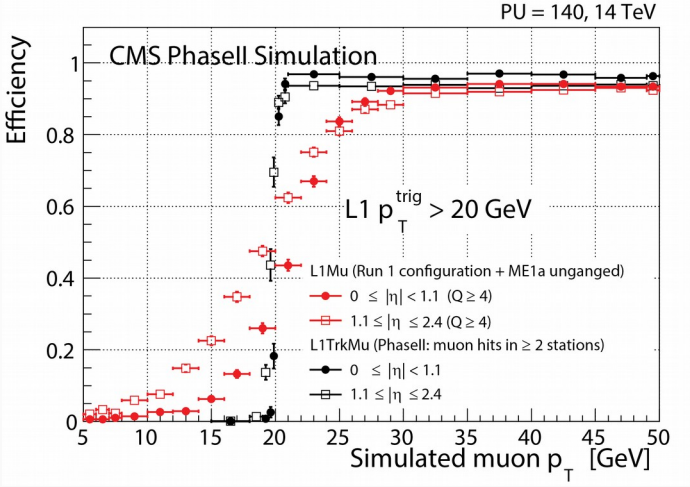


Level 1 Track Trigger

- Providing tracks to the Level 1 (L1) trigger is a key part of the strategy for CMS Phase 2:
  - Helps mitigate the effects of pileup at L1
  - Improves the measurement of objects with tracks (e.g., leptons)
  - Opens up the possibility for new kinds of triggers (e.g., displaced or disappearing tracks)

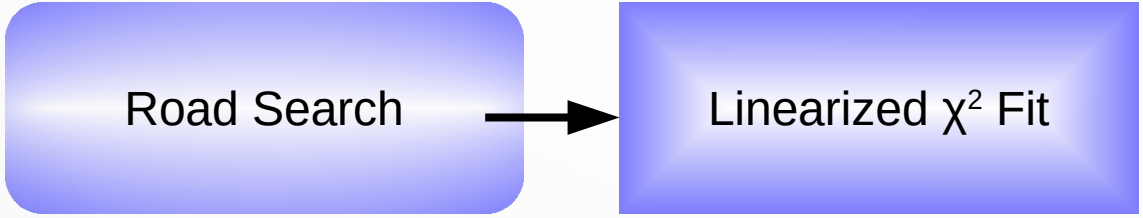




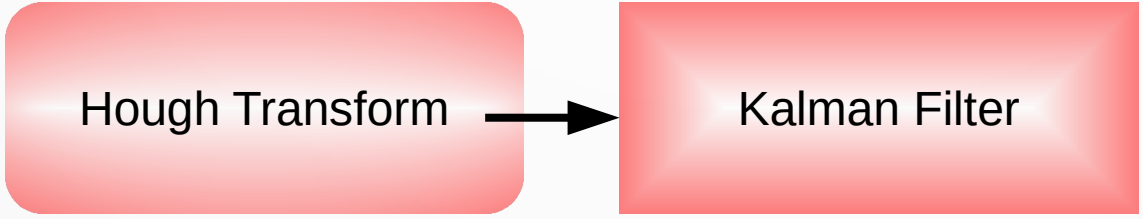


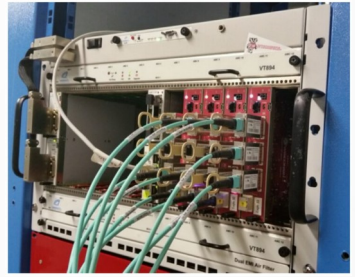
Track Trigger Algorithm

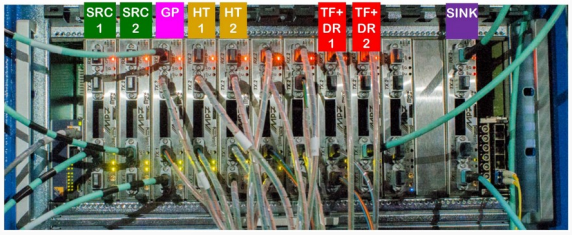
- Historically, two all-FPGA algorithms have been developed:
  - Tracklet



  - Time-Multiplexed Track Finder (TMTT)



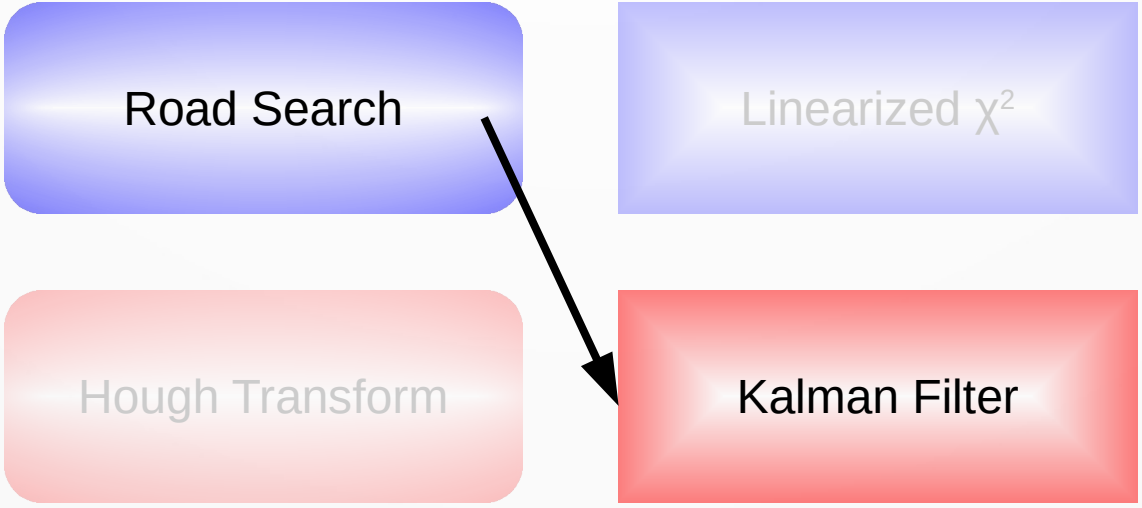




- Similar efficiencies and resolutions for both
- Technical demonstrations in 2016 proved the feasibility of both approaches

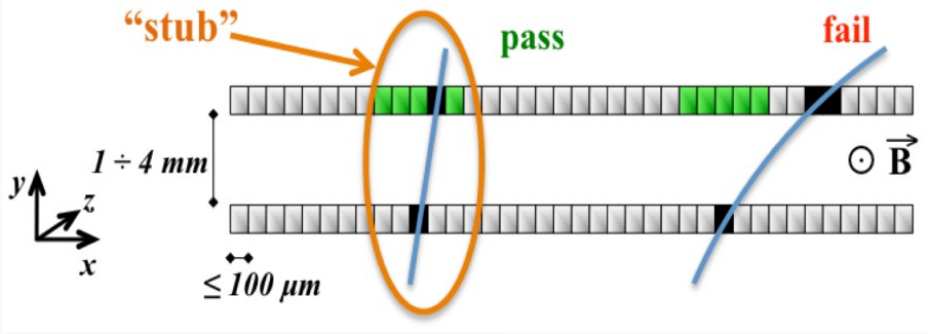
Track Trigger Algorithm

- Current focus is on a hybrid of the two:
  - Combines most sophisticated parts of both algorithms
- The following slides give an outline of this hybrid algorithm



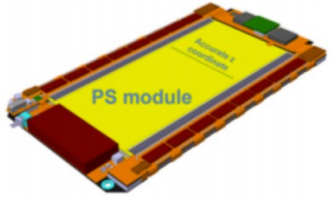
Track stubs

- Track finding starts with track stubs
- Stubs formed from two types of p_T modules
- Two-sided modules allows for front-end p_T discrimination:
  - Stubs with too low of p_T are rejected
  - Data reduction factor of 10-100



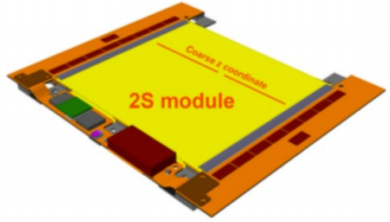
Pixel-strip (PS) modules

- Top sensor: 2×960 strips, 2.4 cm long, 100 μm pitch
- Bottom sensor: 32×960 pixels, 1.5 mm × 100 μm



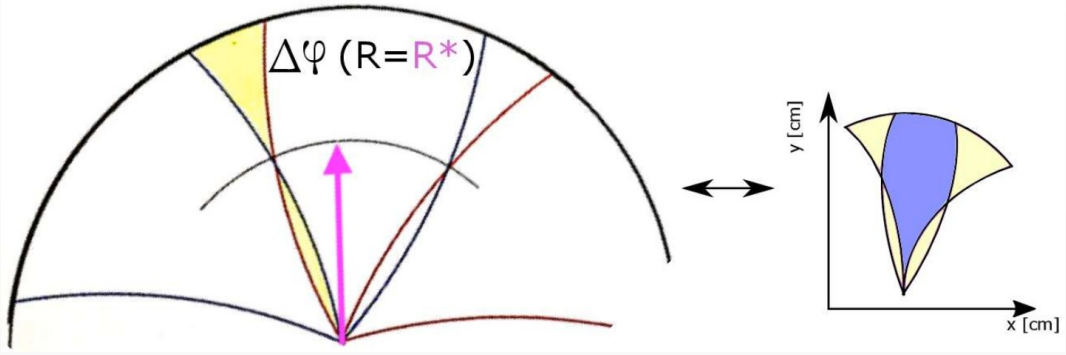
Strip-strip (2S) modules

- Both sensors are strips
- 2×1016 strips, 5 cm long, 90 μm pitch



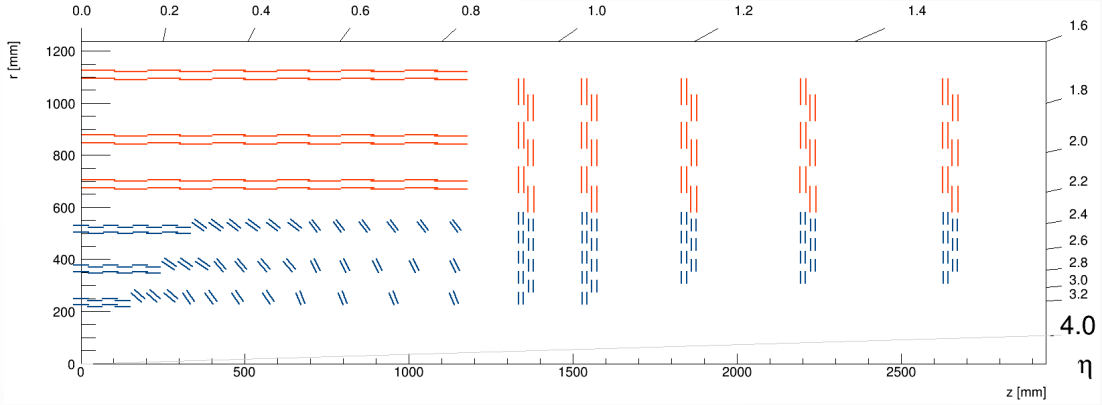
Parallelization

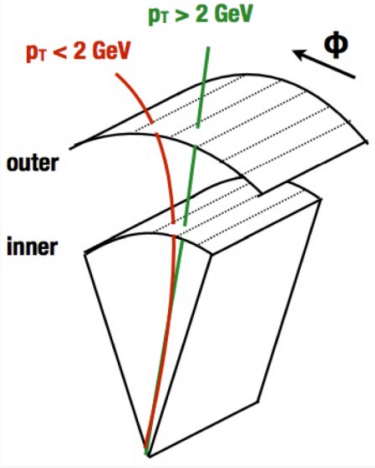
- Track-finding will be parallelized, both in time and space
- Time multiplexed with a factor of 18 in the current design
- Detector divided into nine “hourglass” sectors:
  - Hourglass shape prevents tracks above given p_T threshold from entering more than one sector
    ⇒ no cross-sector communication of tracks needed
  - Critical radius tuned to minimize overlap of stubs



Seeds

- Tracks are seeded with pairs of stubs in adjacent layers:
  - Barrel only: L1L2, L3L4, L5L6
  - Disk only: D1D2, D3D4
  - Overlap: L1D1, L2D1

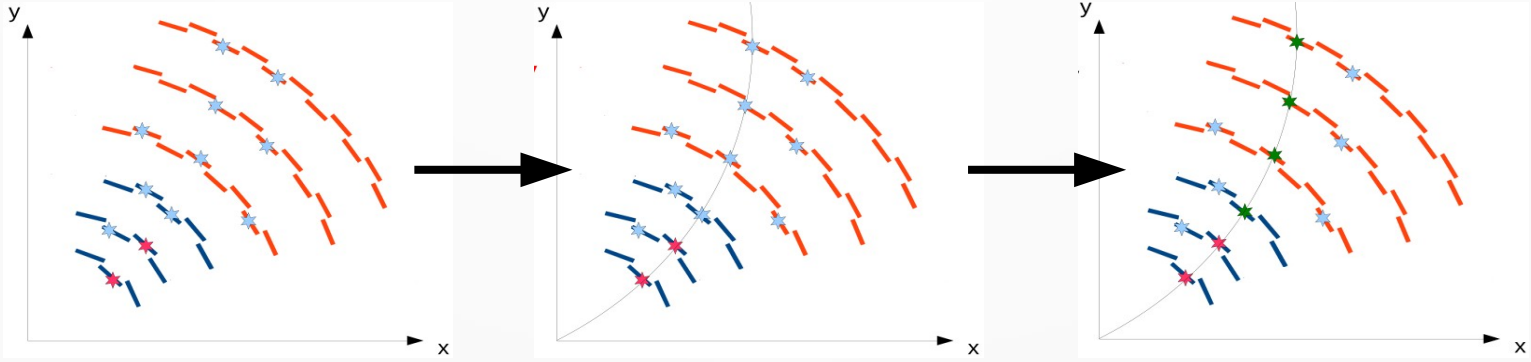




- Only stub pairs consistent with $p_T > 2$ GeV are kept:
  - Tracker layers coursely segmented into virtual modules (VM) (4 or 8 per layer per sector)
  - Only VM pairs consistent with p_T threshold are even connected in the firmware

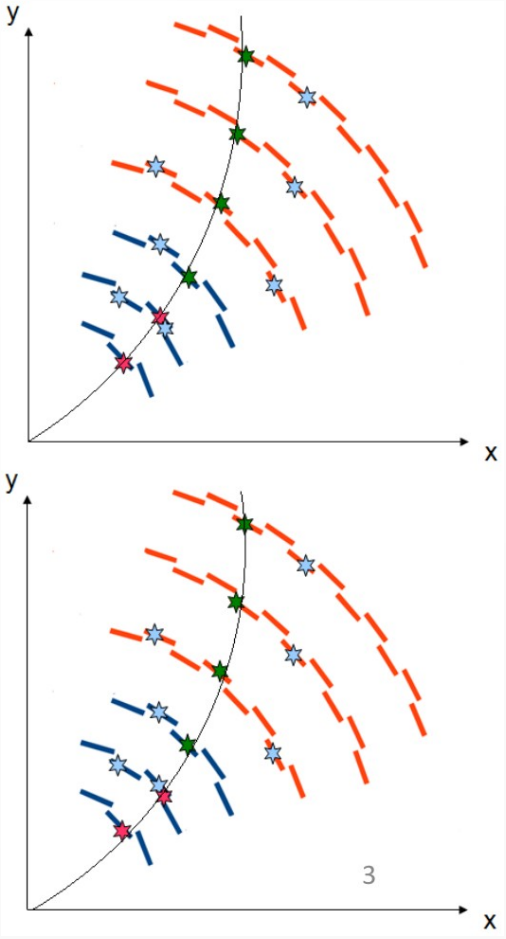
Matches in other layers/disks

- From these seeds, track parameters and projections to other layers/disks are calculated:
  - Assume tracks originate from beamline
- The projections are used to calculate residuals and match stubs in additional layers/disks:
  - This yields full tracks that are the inputs to the final track fit



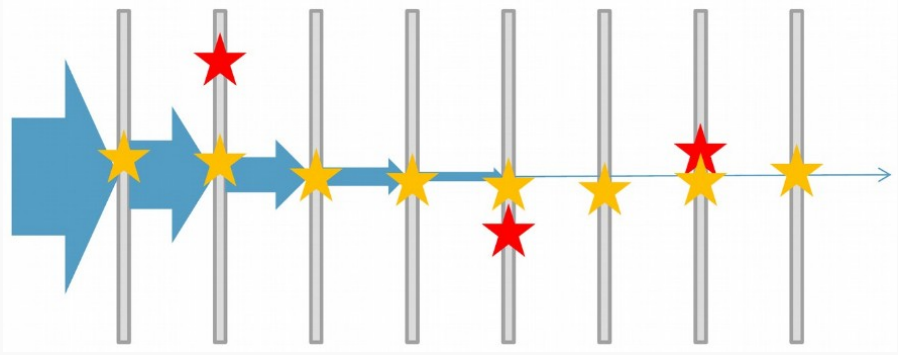
Duplicate removal

- The pattern recognition naturally produces duplicate tracks for a given charged particle:
  - Most come from redundancies in the seeds:
    - e.g., a central charged particle will usually be seeded three times: L1L2, L3L4, L5L6
  - Some come from nearby stubs in a given layer yielding very similar tracks
- These have to be removed before track fitting:
  - Currently merge any tracks that share ≥4 stubs, but this is a very active area of development



Kalman filter

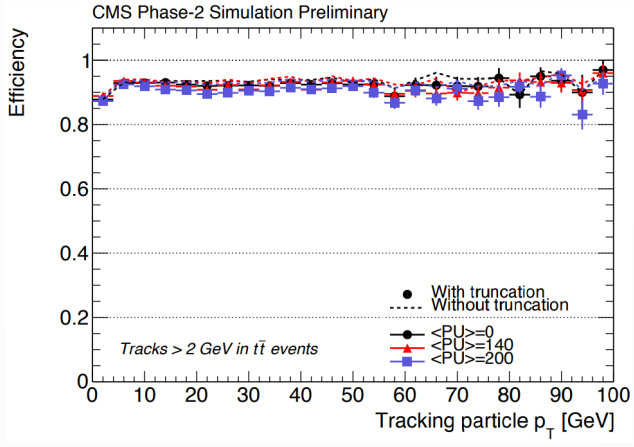
- The final fit of the tracks is done with a Kalman filter:
  - Equivalent to what is currently done in the offline tracking of CMS
  - Starts with coarse tracklet parameters from the seed
  - Adds stubs one by one, updating the helix parameters with greater and greater precision
- By default, there is a beamline constraint and four track parameters are fit:
  - Can easily remove this constraint and also fit for transverse impact parameter (d_0)



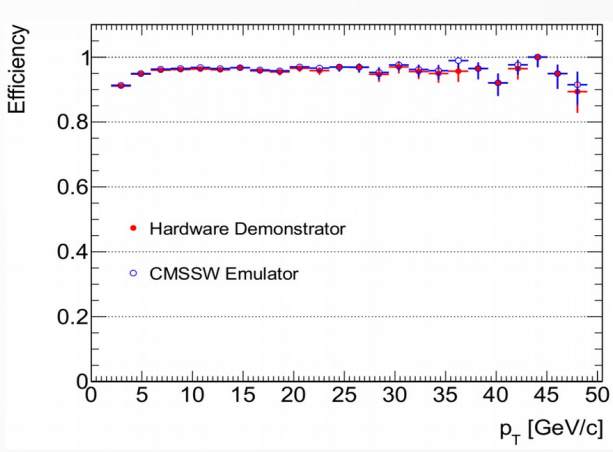
Performance

- Recently merged emulation infrastructures of tracklet and TMTT approaches
- Efficiencies and resolutions of the two approaches are very comparable:
  - Hybrid expected to be at least as good

tracklet



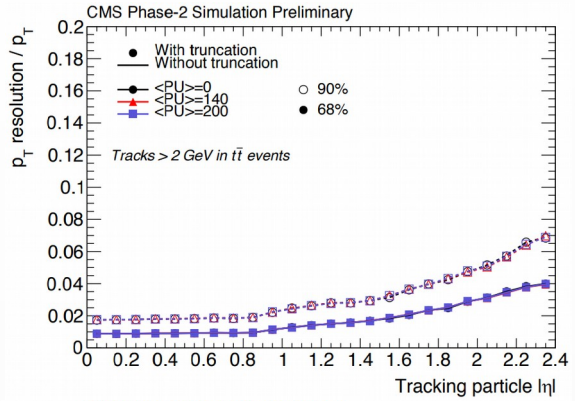
TMTT

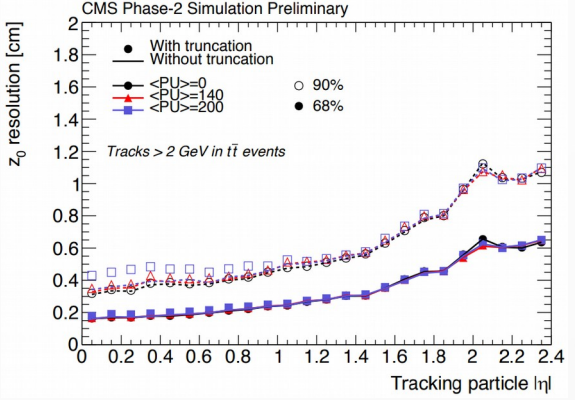


Performance

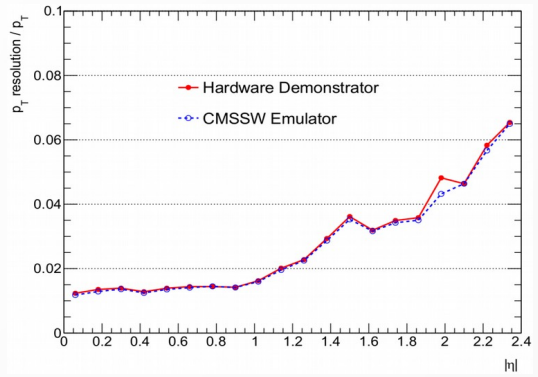
- Recently merged emulation infrastructures of tracklet and TMTT approaches
- Efficiencies and resolutions of the two approaches are very comparable:
  - Hybrid expected to be at least as good

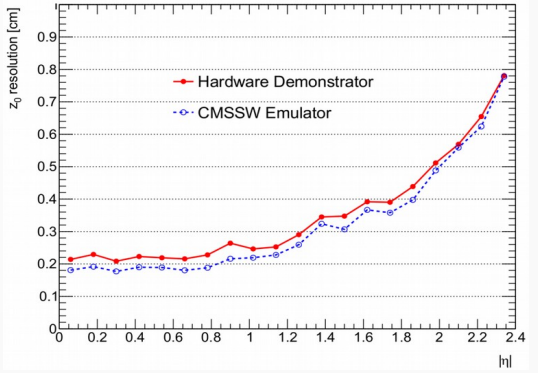
tracklet





TMTT





Firmware Status

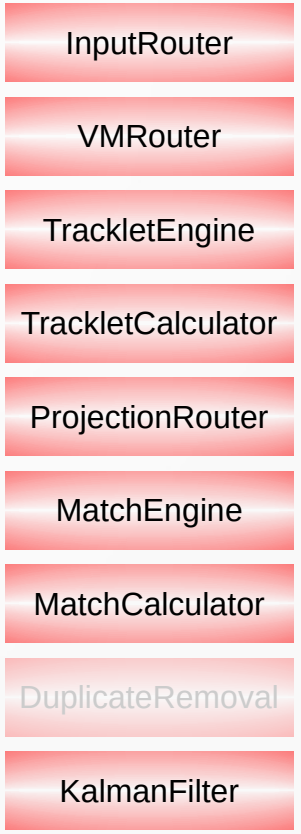
- For firmware development, we have chosen Vivado HLS:
  - Allows FPGA designs to be specified in C++ instead of an HDL like Verilog or VHDL
  - Enables more rapid development, the result is more maintainable, and new ideas can be prototyped more easily
- Not without its hiccups though:
  - HLS can generate incorrect RTL; imperative to verify (cosimulation)
  - Dependencies between read/write operations or between loop iterations can make pipelining tricky
  - Resource utilization and timing estimates from HLS can sometimes be quite inaccurate

Firmware Status

- There are nine processing steps in the current design, each of which will have multiple instances on the FPGA:
  - Memories used to communicate between steps

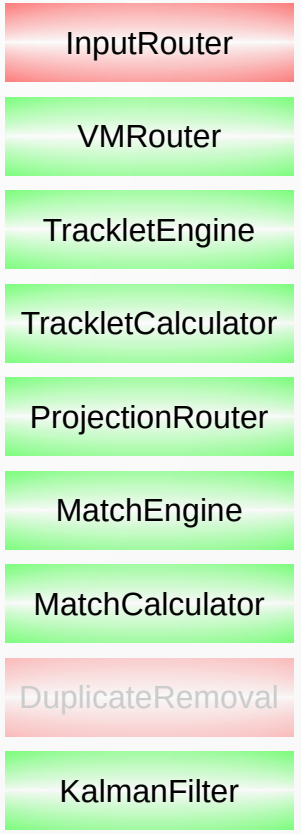
Firmware Status

- There are nine processing steps in the current design, each of which will have multiple instances on the FPGA:
  - Memories used to communicate between steps
- Nearly all have one instance written and tested to be functionally correct:
  - Different instances generated using C++ template programming



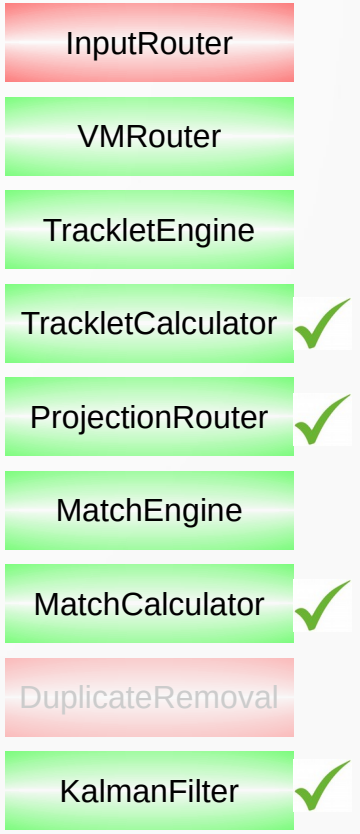
Firmware Status

- There are nine processing steps in the current design, each of which will have multiple instances on the FPGA:
  - Memories used to communicate between steps
- Nearly all have one instance written and tested to be functionally correct:
  - Different instances generated using C++ template programming
- Nearly all of these have achieved desired pipelining



Firmware Status

- There are nine processing steps in the current design, each of which will have multiple instances on the FPGA:
  - Memories used to communicate between steps
- Nearly all have one instance written and tested to be functionally correct:
  - Different instances generated using C++ template programming
- Nearly all of these have achieved desired pipelining
- About half have been fully verified with C/RTL cosimulation
- Goal is to have a full chain of modules ready for integration tests at CERN starting in September/October



Conclusion

- A common L1 tracking algorithm for CMS Phase 2 is finally emerging:
  - Based the most sophisticated aspects of two proven all-FPGA approaches: tracklet and TMTT
- Development of the firmware, written in Vivado HLS, is well underway:
  - About half of the processing steps have fully functioning modules written
- R&D on the algorithm itself is also ongoing, e.g., extending the algorithm to include displaced tracks:
  - Challenging even at the front-end, as stubs from displaced tracks have low efficiency in L1
  - Has the potential to benefit electron tracking, as well as novel triggers targeting BSM physics