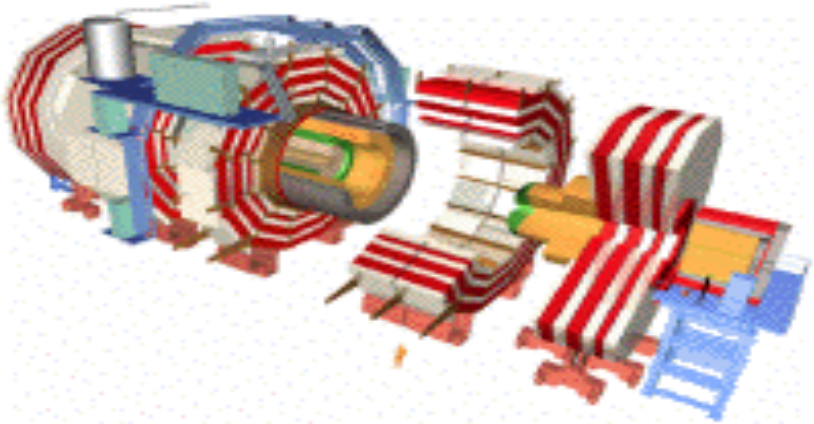
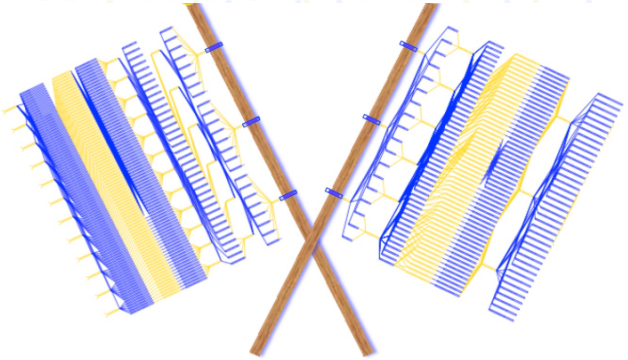
CMS FPGA Based Tracklet Approach for L1 Track Finding





The Tracklets

Anders Ryd
(Cornell University)

On behalf of the
CMS Tracklet Group

Presented at AWLC

June 29, 2017

Outline/Introduction

- Motivation for tracking in L1
  - Trigger Challenge and CMS HL-LHC tracker
- Tracklet algorithm
  - Simulation and challenges
- Tracklet demonstrator
  - Performance and results
- Next steps and outlook

This project has involved developing tracking algorithms suitable for execution on FPGAs

- Student contributions to this effort has been critical
  - Firmware written by students with engineering 'guidance'

The goal of this project was to demonstrate the feasibility of tracking at 40 MHz by the fall of 2016 for the CMS tracker TDR

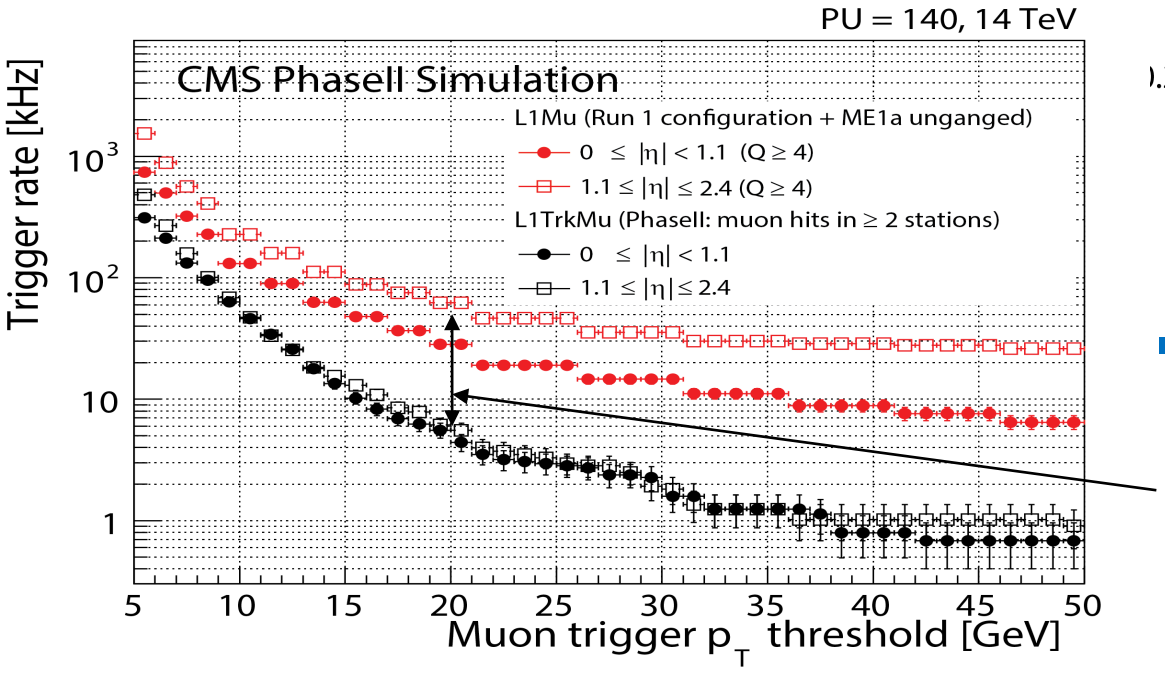
- We are now proceeding to defining the upgrade project based on the demonstration

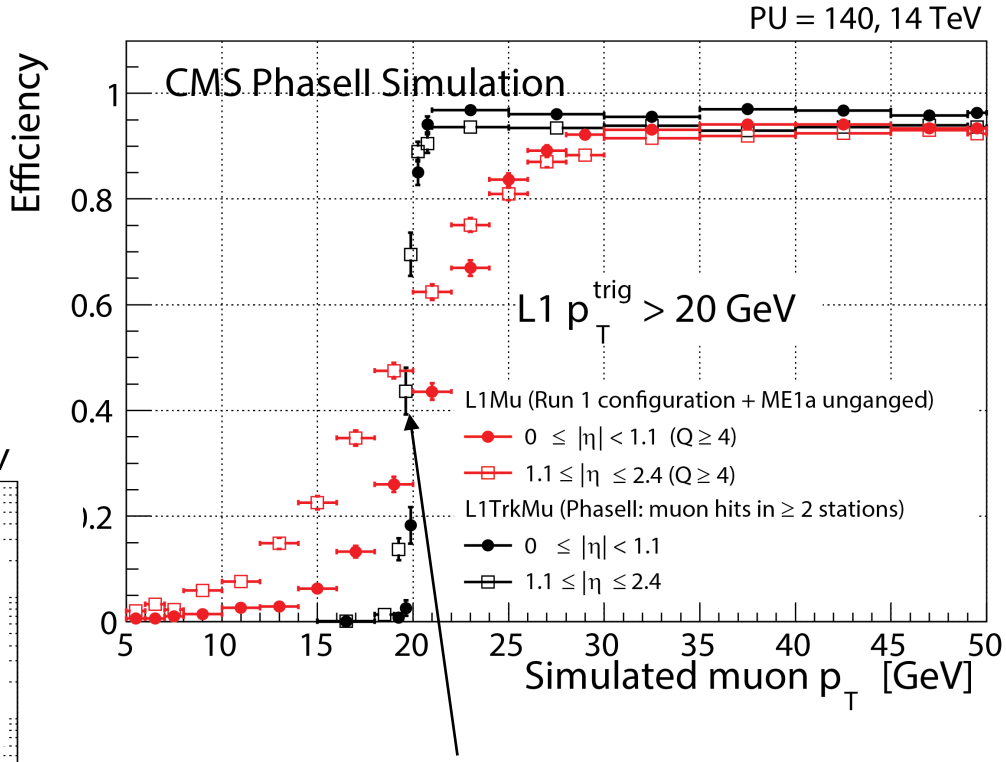
HL-LHC Trigger Challenge

- The LHC is planned to be upgraded during "Long Shutdown 3" (LS3) in 2024-2026
  - Leveled luminosity up to 7.5×10^{34} $cm^{-2}s^{-1}$
  - Pileup of up to 200
    - 200 proton-proton interactions on average in each proton-proton bunch collision every 25 ns (40 MHz)
    - About 8×10^{9} proton-proton interactions per second
- For the HL-LHC operations CMS plans to record up to 7.5 kHz of collisions
  - Need to reduce rate by factor of ~5,000. This is done in two stages:
    - L1 (hardware trigger) from 40 MHz → 750 kHz
    - HLT (High Level Trigger in software) from 750 kHz → 7.5 kHz.
- The current L1 trigger in CMS does not have access to tracking information.
  - To meet the HL-LHC requirements the CMS L1 trigger will have access to tracking information (p_T>2 GeV) for each bunch crossing

Example of tracking in L1: Muon Triggers

- Track matching to muon candidates has high efficiency
- Muons+L1Tracks provide much sharper threshold





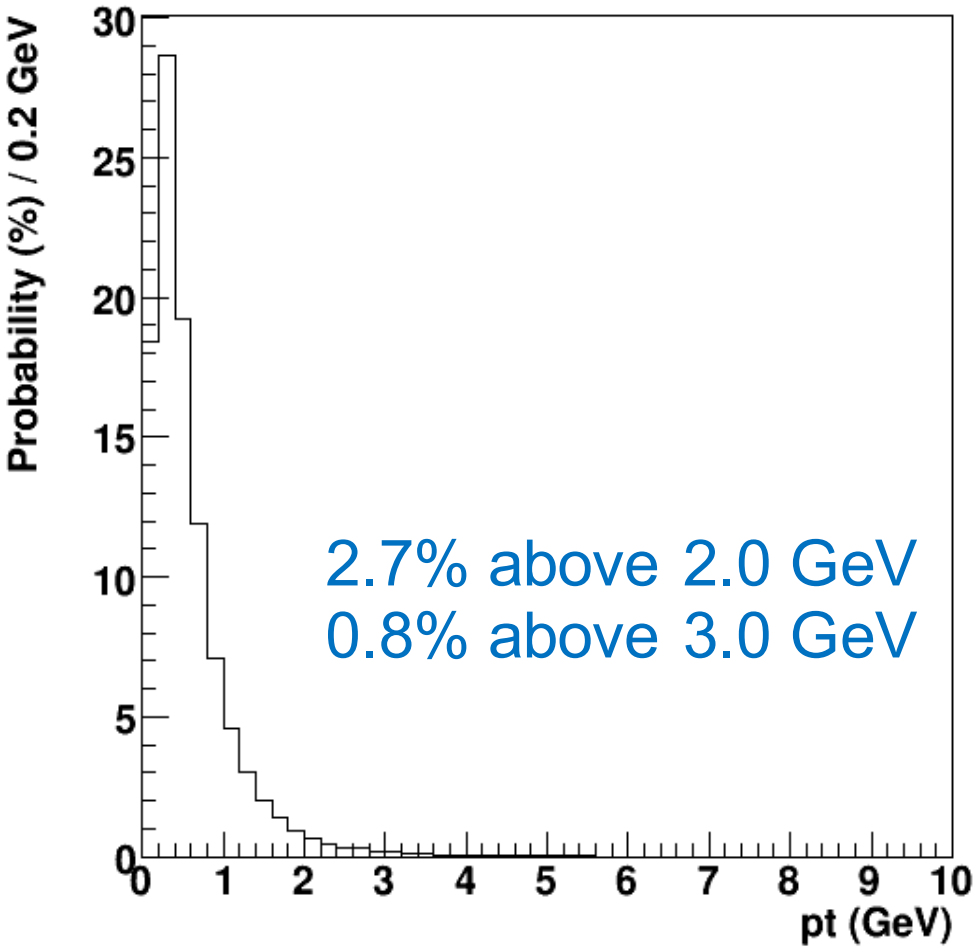
- Sharp threshold allows a significant rate reduction:
  - Factor ~10 reduction @20 GeV

Tracking at L1 is also powerful tool for electrons, τ's, jets, photons as detailed in the CMS HL-LHC TP: https://cds.cern.ch/record/2020886

HL-LHC Environment: Minbias

- 14 TeV minbias:
  - 6.5 charged particles, mostly pions, per unit of rapidity or 33 charged particles in the tracking volume $|\eta|<2.5$
  - With 200 PU → 6600 charged particles per bunch crossing.
  - Soft spectrum – peaks at p_T of about 200 MeV.
- The average min bias event has $33\times2.7\%=0.89$ tracks with $p_T>2.0$
  - For PU=200 we expect ~180 tracks with $p_T>2.0$.

p_T distribution in 14 TeV minbias



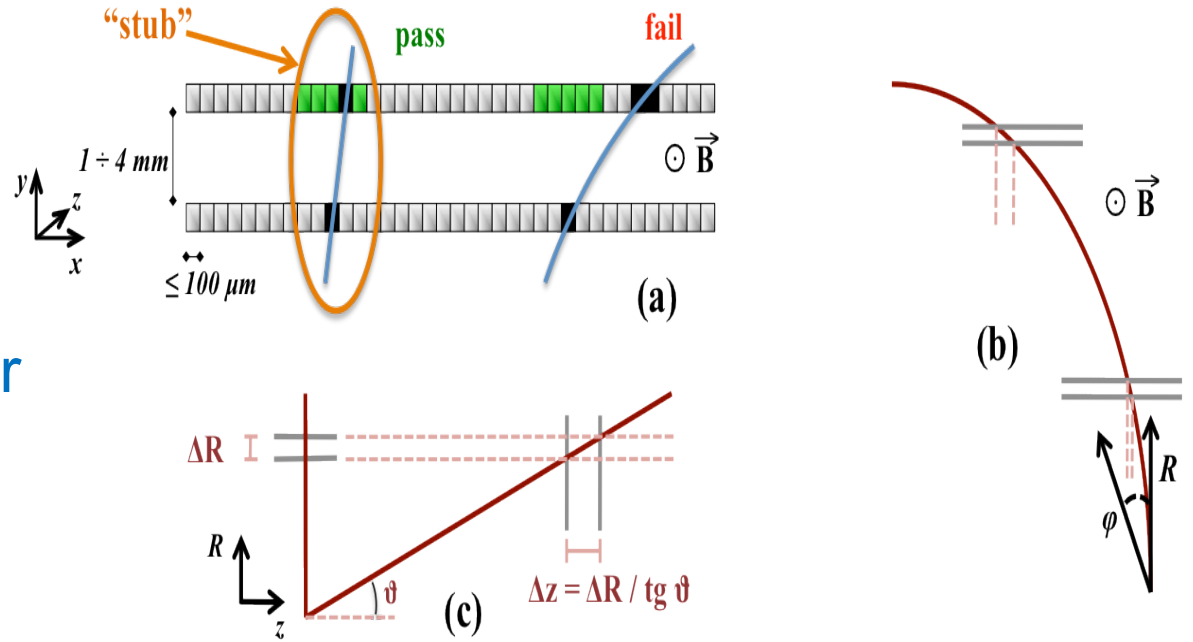
Data volume:

- 200 PU average of about 13,000 stubs/bunch crossing
- Each stub is ~36 bits
  - At 40 MHz BX rate we have 20 Tbits/s

Only ~5% of stubs are from tracks with $p_T>2$GeV

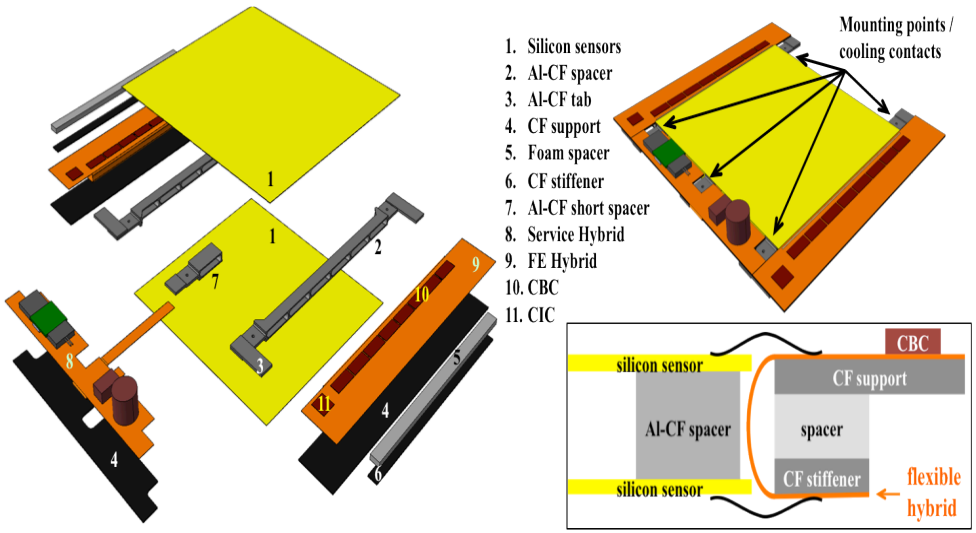
p_T Modules

- Correlating hits in closely spaced sensors give p_T discrimination in B=3.8 T
- Correlations formed on module – data reduction for trigger readout



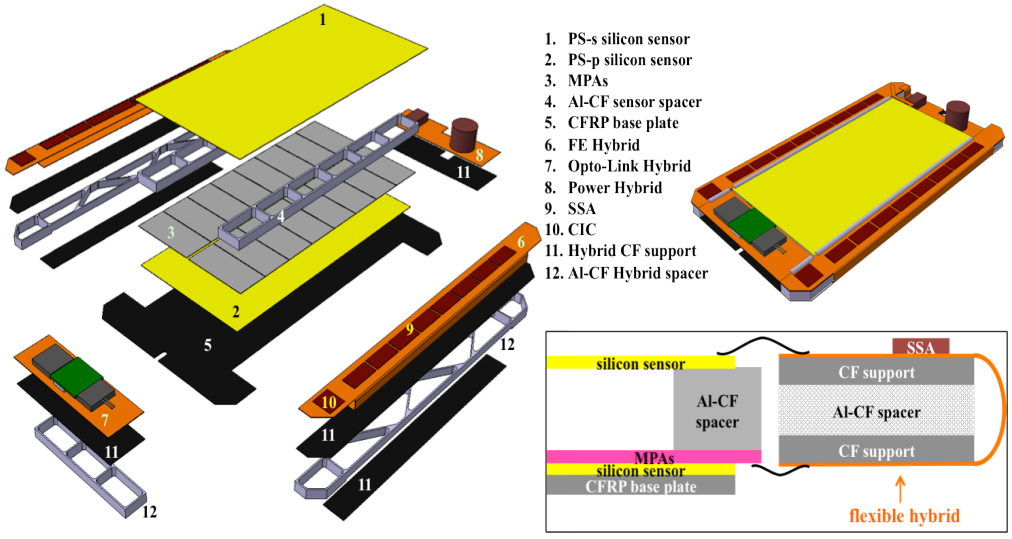
Strip-strip (2S) Modules

2×5 cm strips 90 um pitch

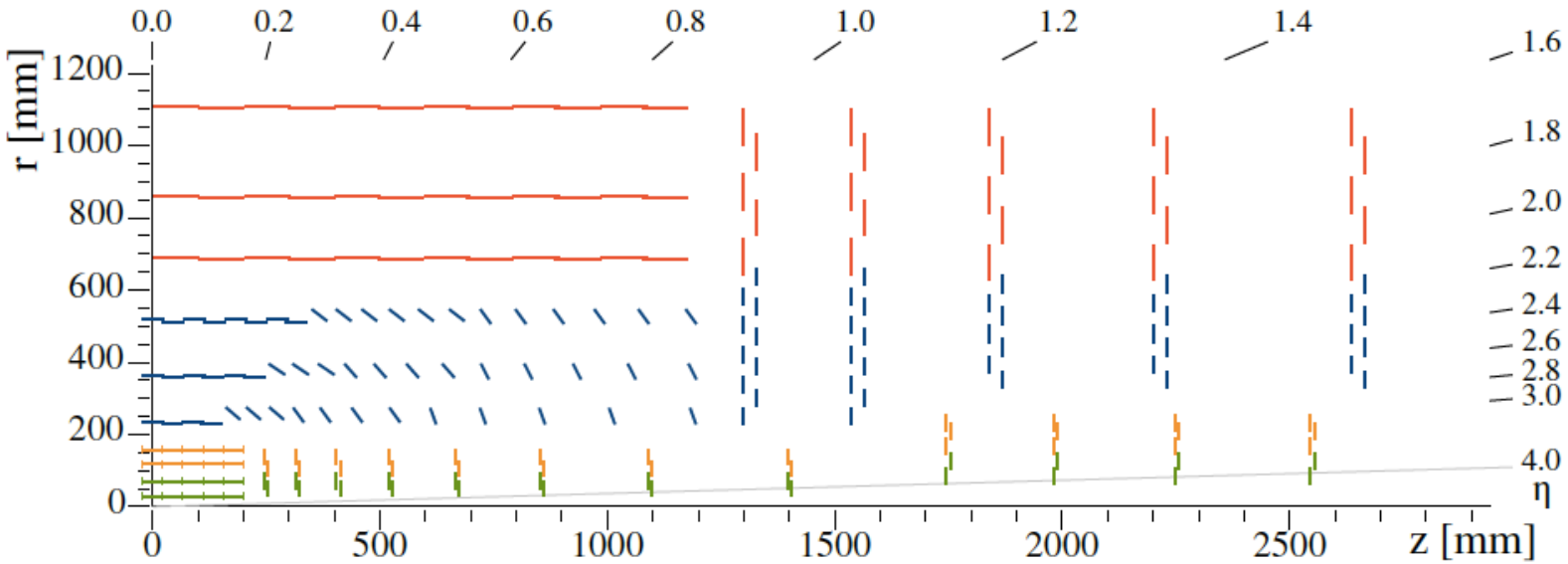


Pixel-strip (PS) Modules

2×2.5 cm strips 100 um pitch
1.5 mm macro pixels



CMS HL-LHC Tracker Layout

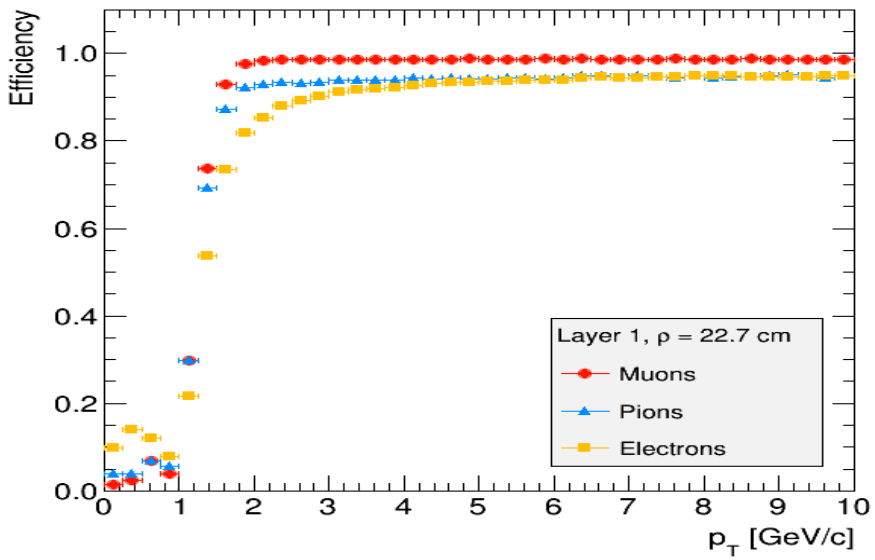


¼ of Outer Tracker

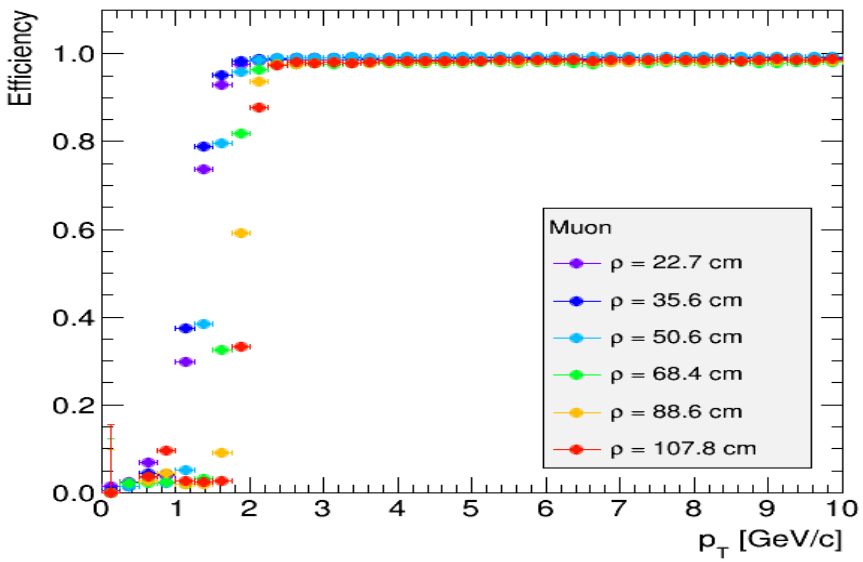
6 Layers

5 Disks

Layer 1 Stub Finding Efficiency



Stub Finding Efficiency per Layer



L1 Tracking Goals

- Highest possible efficiency for high p_T isolated tracks
  - Lepton triggers: e, μ, and τ
- Good z resolution, ~1 mm, for pileup suppression
  - Veto objects from different pp interactions
- Good efficiency for tracking in jets
  - Need to be able to identify jet vertex for pileup suppression
- Efficiency down to p_T=2 GeV
  - Low momentum tracks used for track based isolation
- Good track purity
  - Important e.g. for τ identification
- More powerfull, but complicated, L1 algorithms such as 'particle flow' are under study

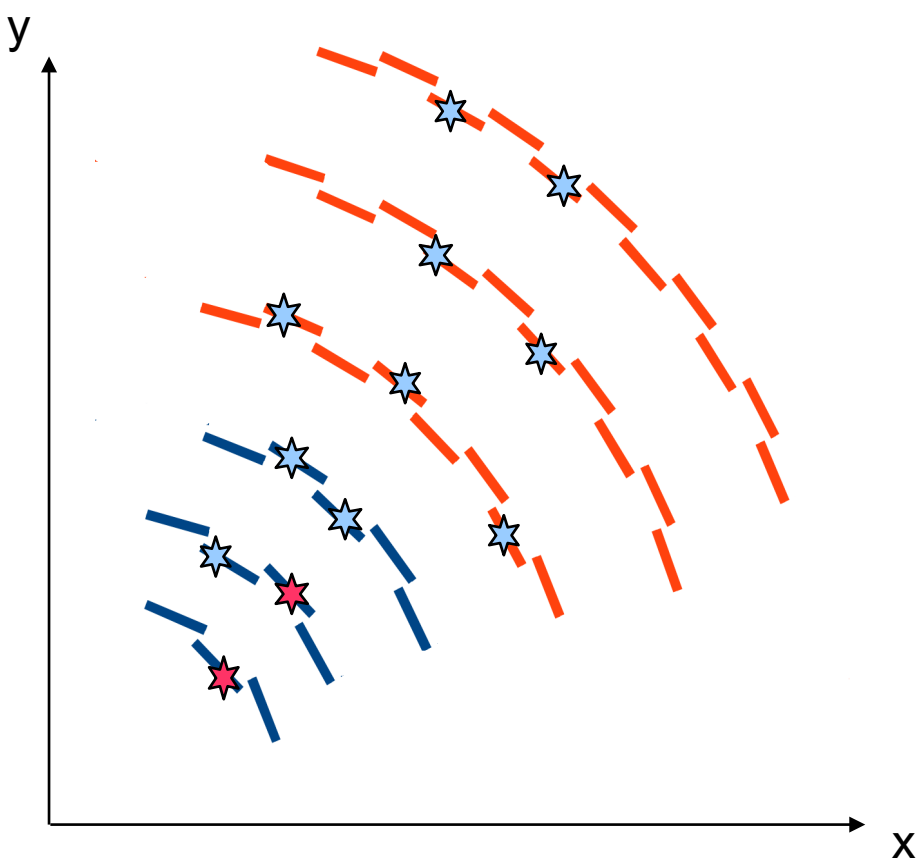
All this delivered in ~5 μs to meet overall L1 trigger latency requirements

Tracklet – an all FPGA Approach

- The Tracklet approach described in this talk is an all FPGA solution that takes advantage of the rapid development of commercial FPGAs
- Algorithm implements a conventional road search technique
  - Seeds are formed from pairs of stubs from neighboring layers (or disk)
  - Projected to other layers (disks) to add stubs
  - Combinatorics addressed by aggressive partitioning
    - Implementation massively parallel
- The implementation makes use of the high precision of the hits in the silicon detector to project the seeds to neighboring layers
  - Calculations are performed in the FPGAs using the powerful Digitial Signal Processing (DSP) units
  - Data routing capability important for the hardware implementation of the algorithm

Tracklet Based Track Finding

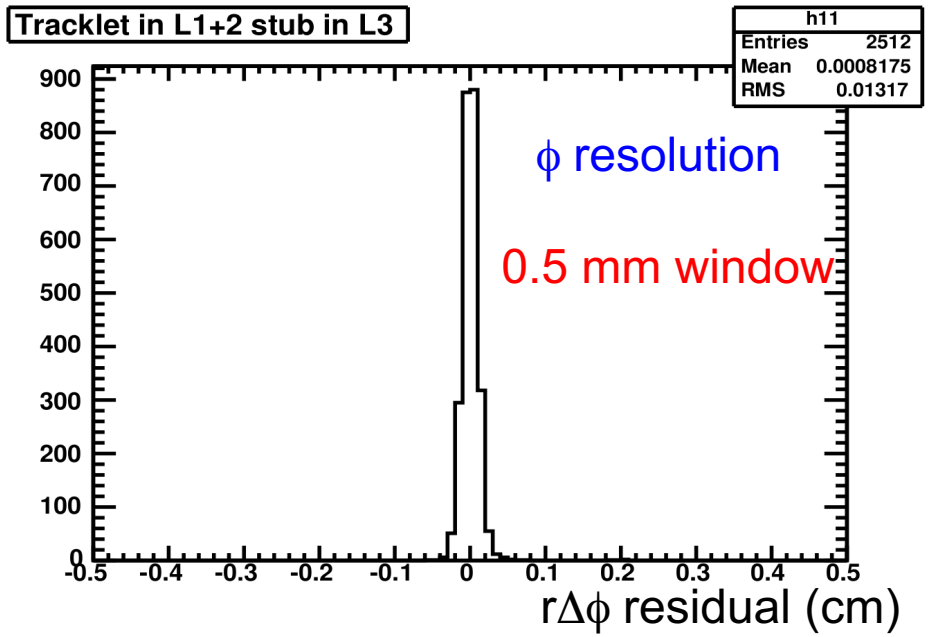
- Form track seeds, tracklets, from pairs of stubs in neighboring layers

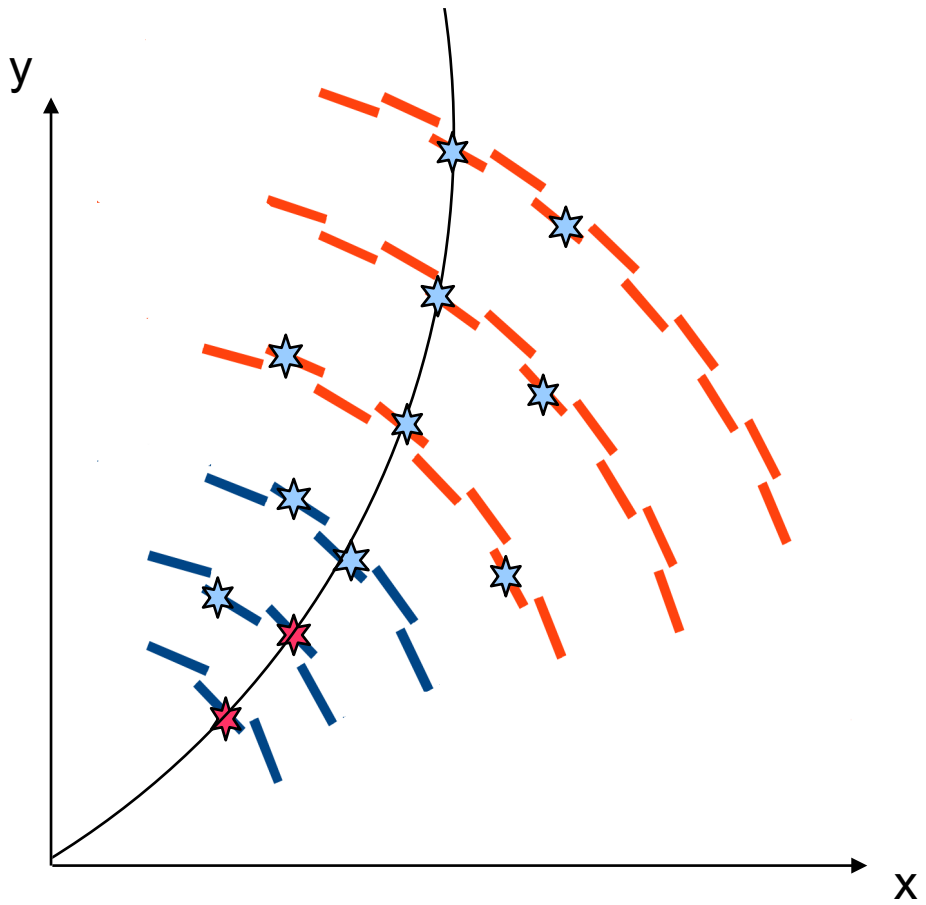


Tracklet Based Track Finding

- Form track seeds, tracklets, from pairs of stubs in neighboring layers
- Match stubs on road defined by tracklet and IP constraint

Matching resolutions for tracklets in L1+L2 projected to L3

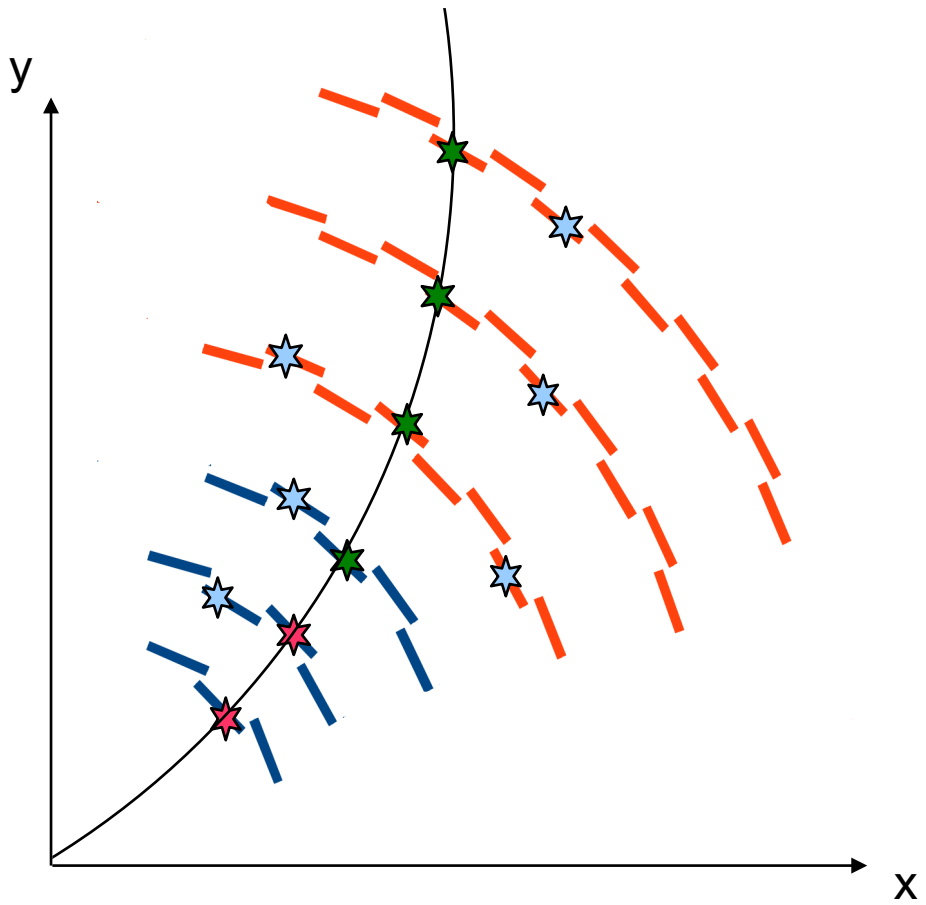




Residual from matching used in final trackfit

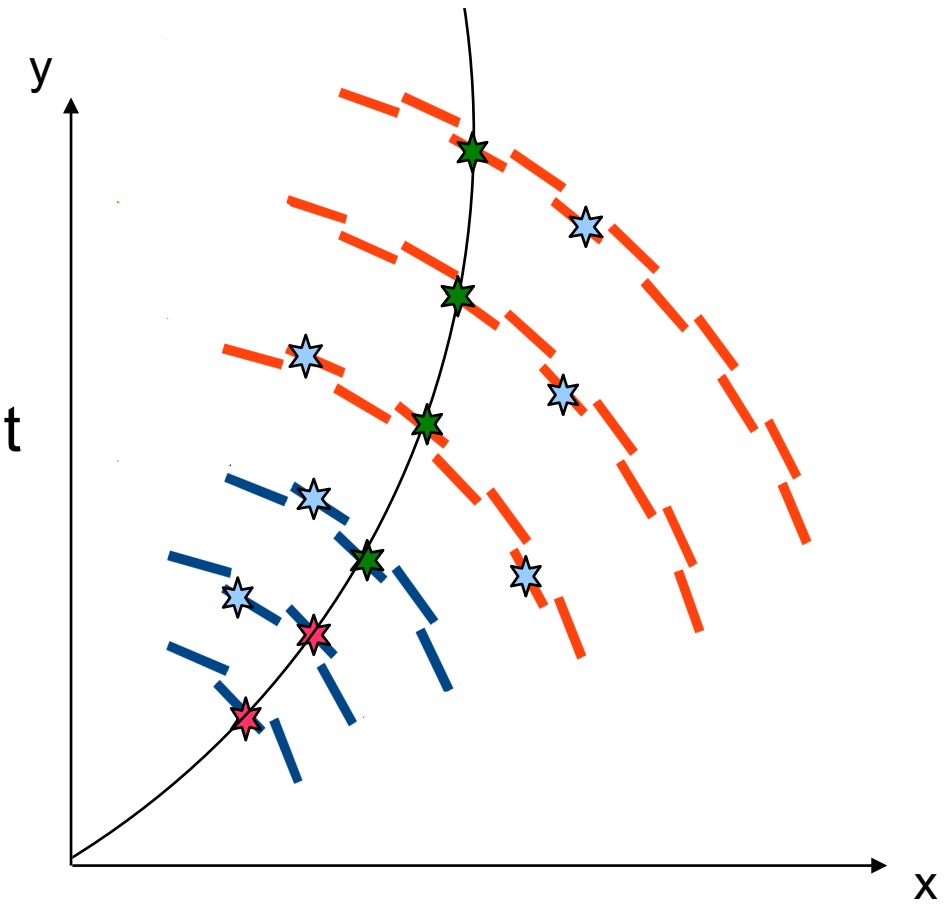
Tracklet Based Track Finding

- Form track seeds, tracklets, from pairs of stubs in neighboring layers
- Match stubs on road defined by tracklet
- Fit the hits matched to the tracklet using a linearized χ^2 fit
  - Tracklet parameters good – linear fit works very will



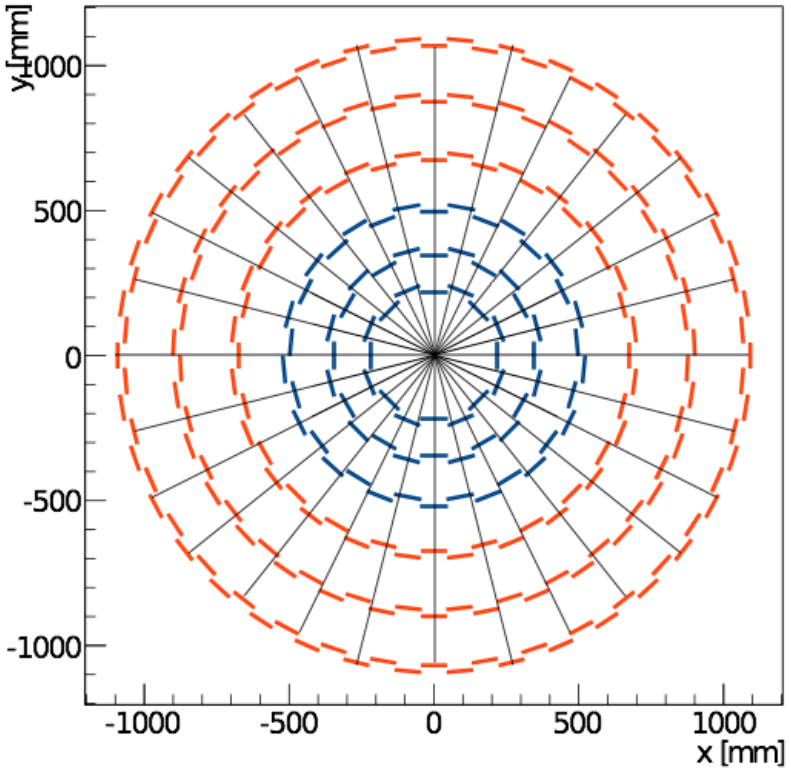
Tracklet Based Track Finding

- Form track seeds, tracklets, from pairs of stubs in neighboring layers
- Match stubs on road defined by tracklet
- Fit the hits matched to the tracklet using a linearized χ^2 fit
- Seeding is done in parallel in different layers
- Duplicate tracks are removed if they share 2 or more stubs



Implementation Approach

- Make use of multiplexing:
  - Spatial: Divide detector into 28 ϕ sectors. Seeding (tracklet finding) local within sector - ~10% stub duplication
  - Time: Factor of 6 time multiplexing. Each sector processor receives a new event every 150 ns.
- From simulations (at 200 PU) we have per sector:
  - ~500 stubs
  - Form ~70 tracklets
  - Find ~15 track (including duplicates)



What computational resources are need to implement this tracking in an FPGA?

Resource Estimates

- Approximate number of DSP operations required for the different steps in the algorithm

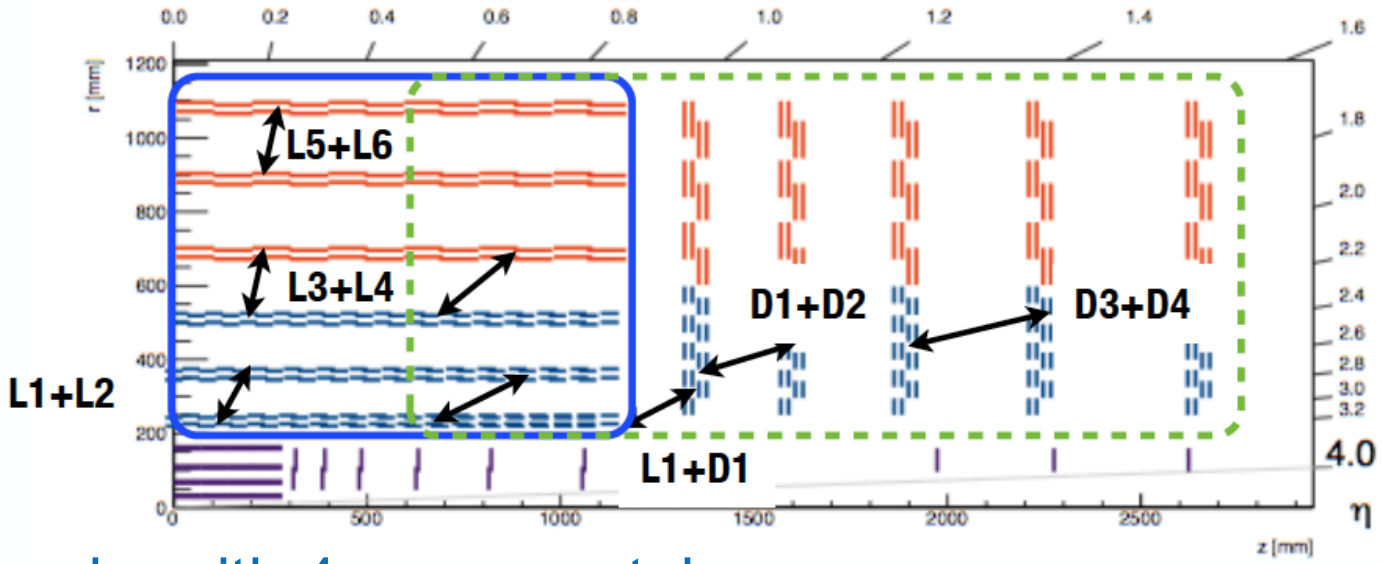
Task	# objects	DSP operations per object	Total DSP operations
Tracklet Parameters	70	20	1400
Tracklet Projections (4 per tracklet)	70	4×10	2800
Matching	200	4	800
Track fit	15	20	300
Total			5300

- Assuming a factor of 6 time multiplexing (150 ns between events) and a 240 MHz project clock each DSP in the FPGA can perform 36 operations. A vertex 7 (690T) has about 3600 DSPs and can perform ~130,000 operations per event
  - We need about 4% of the resources
- The Ultrascale+ FPGAs will have up to 12,000 DSPs and O(1%) would be needed for the L1 tracking
- However: Challenge is to handle combinatorics and tails

Baseline Implementation

We have established a 'baseline project' for demonstration:

- Factor of 6 time MUX: Each board receives a new event every 150 ns (LHC bunch crossing frequency is 40 MHz)
- 28 sectors - Tracking performed for p_T>2 GeV
  - Tracklets formed locally in sector
  - Projections sent to neighboring sectors
  - A sector covers the full η range
- Seeding (tracklet finding) done in multiple layer (disk) combinations:



- Fit tracks with 4 or more stubs
  - I.e. tracklet + at least two matched stubs
  - With tracklet seed a linearized χ^2 fit works very well

Combinatorics

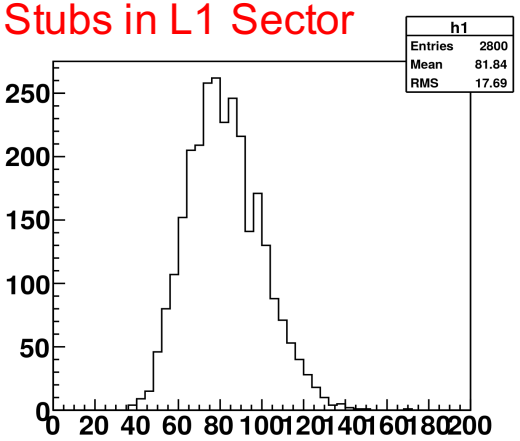
- The obvious challenge for implementing a tracking algorithm with a fixed latency is how to handle the combinatorics.
- The tracklet approach described here has two key steps were combinatorics is an issue:
  - Forming tracklets from pairs of stubs
  - Matching tracklets projected to other layers (or disks) to stubs
- Both cases are addressed in a similar manner:
  - The detector is divided into smaller regions and the combinatorics problem is solved by massive parallelism

- To illustrate this we consider the tracklet finding in some detail
  - Remember that we work here in one sector, so 1/28 of the full detector
  - The plots of the occupancy is taken from a sampler of $t\bar{t}$ with 200 minbias events superimposed

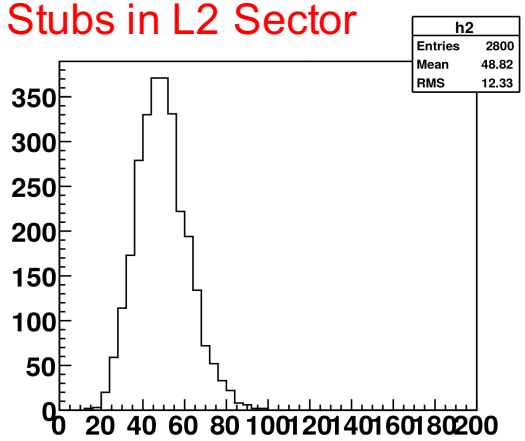
Tracklet Formation

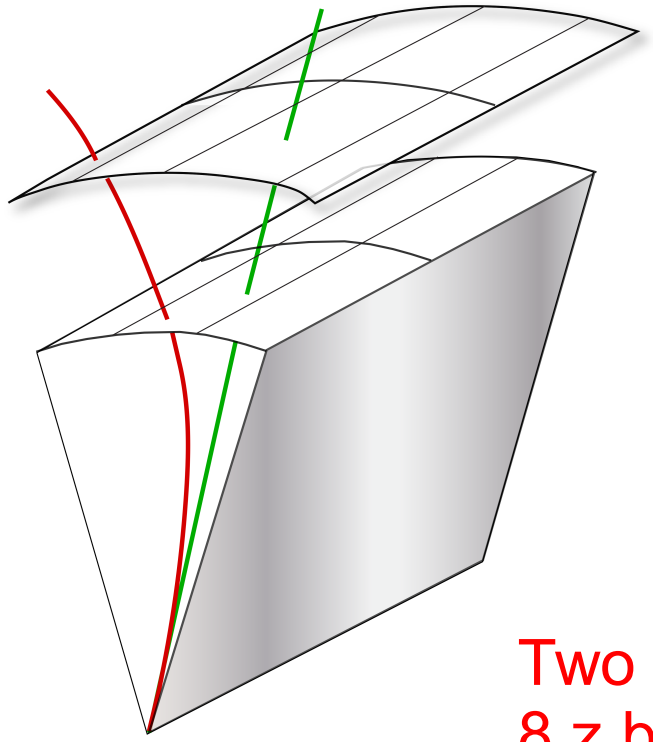
- Average number of stubs in a sector:
  - Layer 1: ~80 stubs
  - Layer 2: ~50 stubs
- Total of ~4000 possible pairs
  - Most of these don't satisfy:
    - $|z_0|<15$ cm
    - $p_T>2$ GeV
- Divide each layer into sub regions ('Virtual Modules' VMs)
  - Layer 1: 8 z × 3 ϕ = 24 VMs
  - Layer 2: 8 z × 4 ϕ = 32 VMs
- Total of 24×32=768 pairs of VMs
  - 96 of these can form tracklets
  - Search for tracklets in parallel in these 96 pairs of VMs.

Stubs in L1 Sector

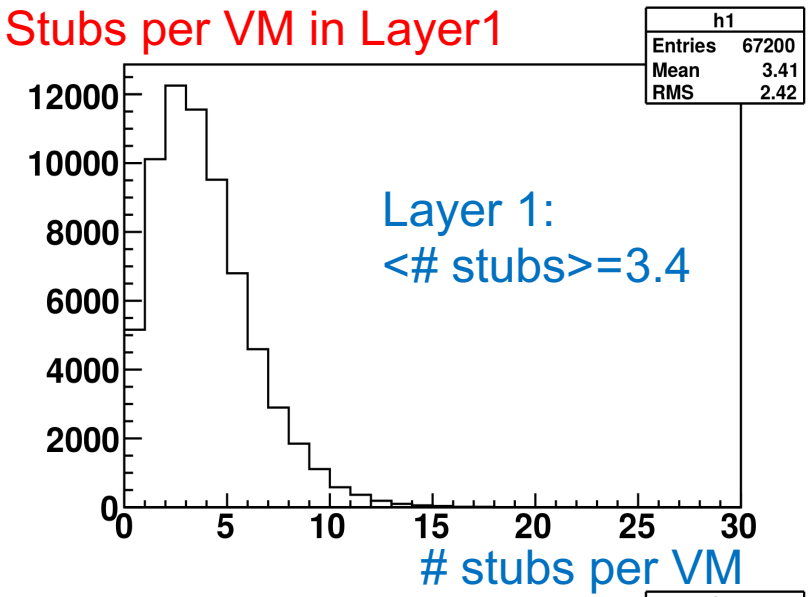


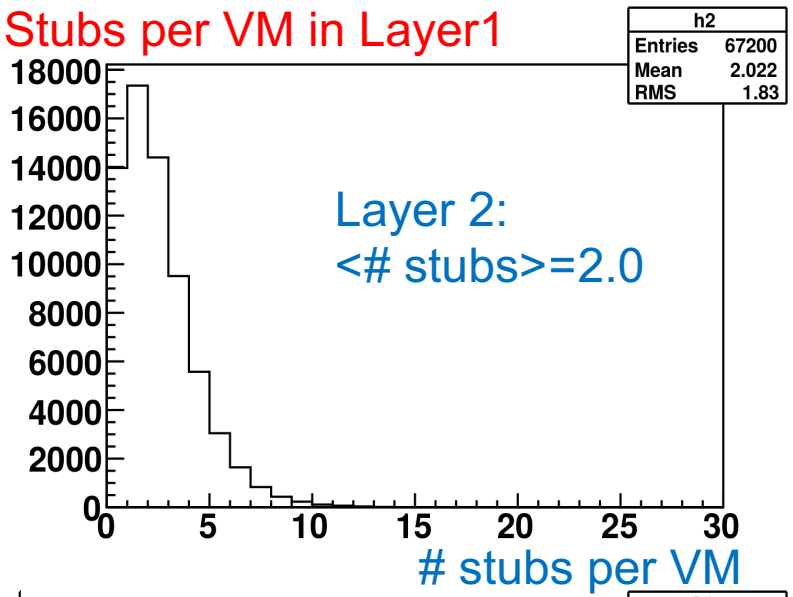
Stubs in L2 Sector



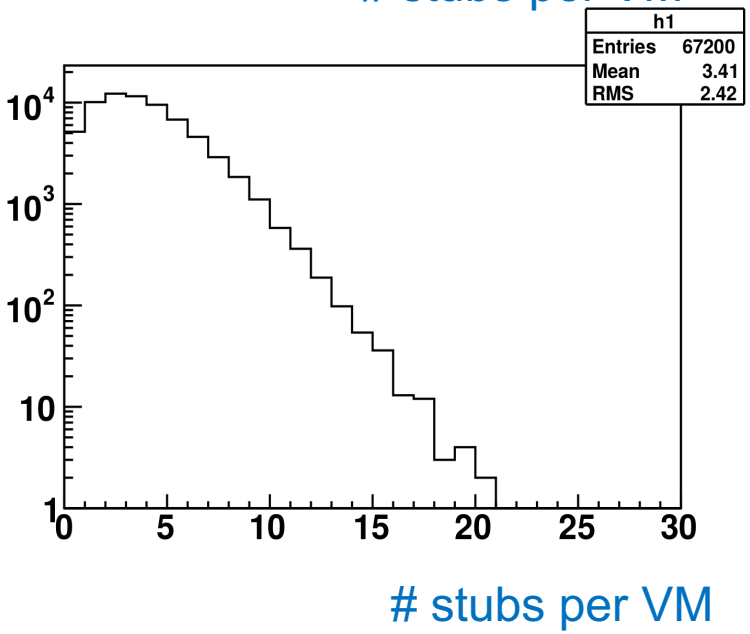


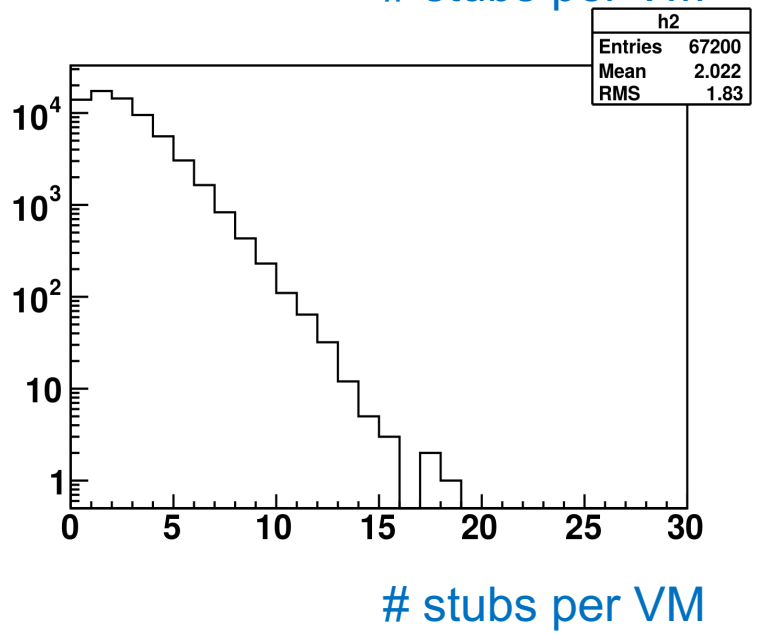
Virtual Module Occupancy





Linear scale

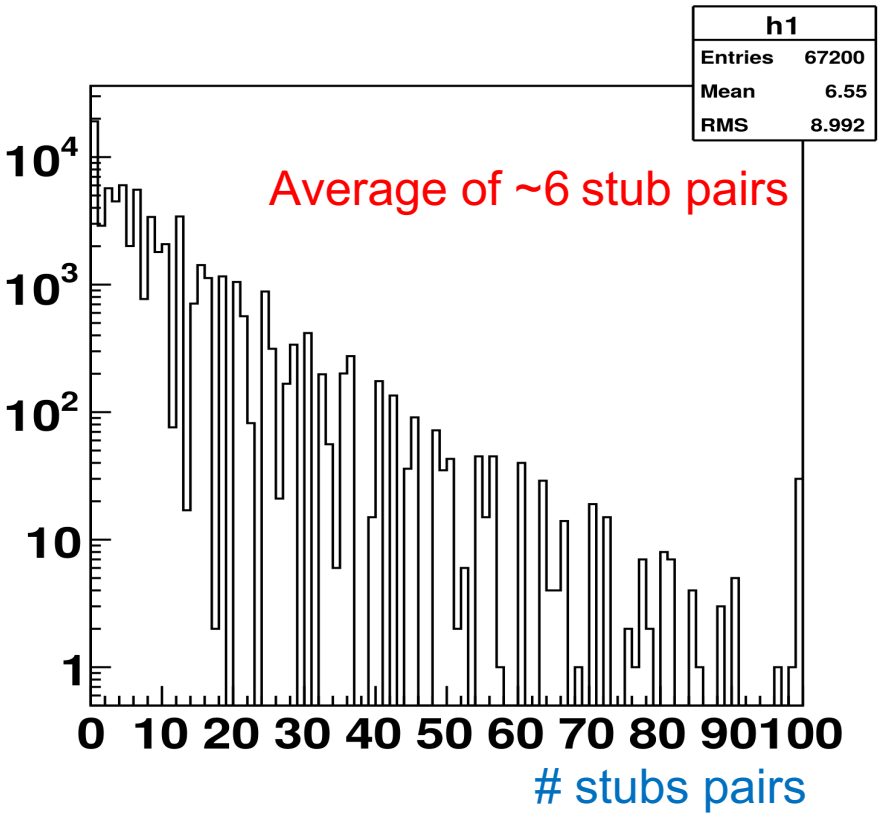




Log scale

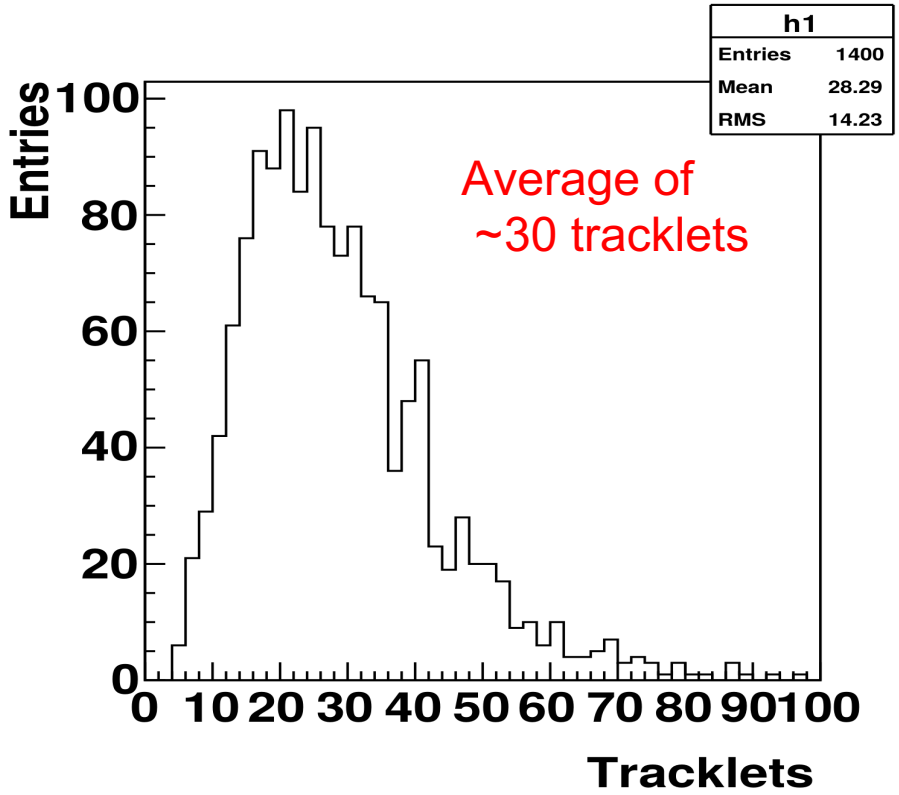
Tracklet Formation Combinatorics

Stub pairs per virtual module combination



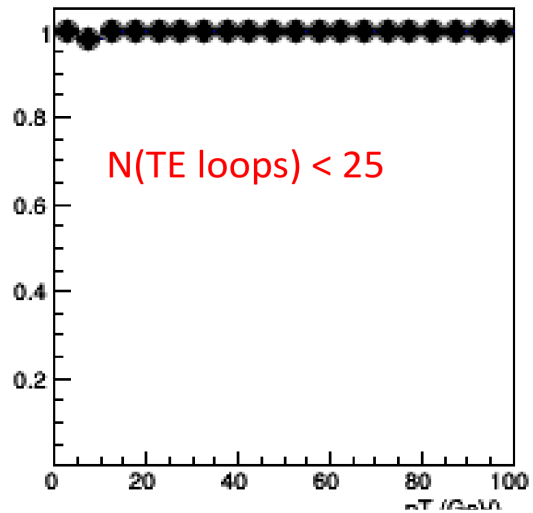
Tracklets found in L1+L2:

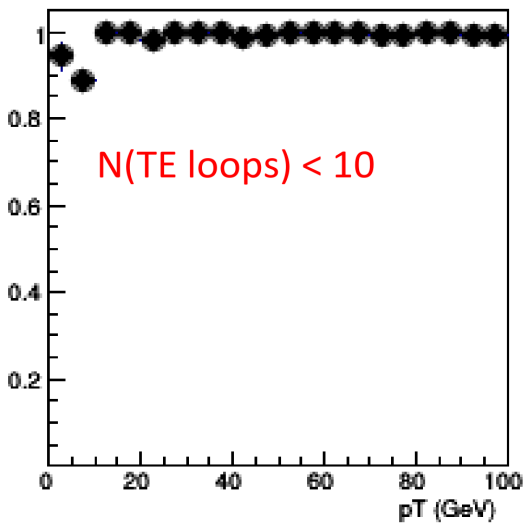
- $|z_0|<15$ cm
- $p_T>2$ GeV

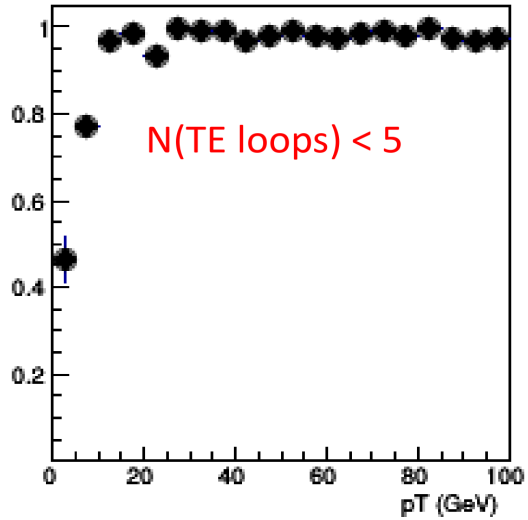


Truncating Tracklet Finding

- There will be a limit to how many stub pairs we can try in the tracklet engine (TE)
- The effect of the truncation for finding isolated muons in events with pileup is illustrated below





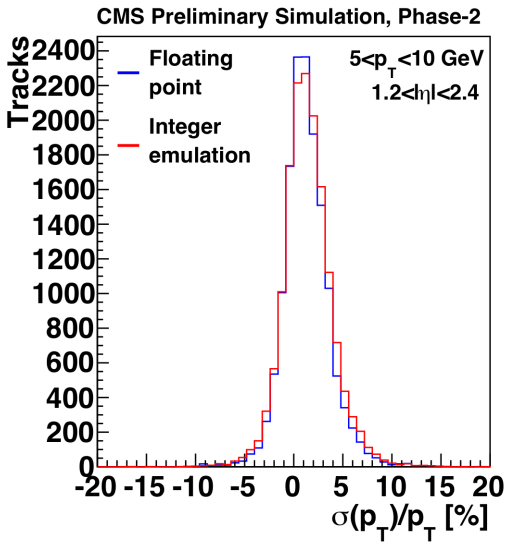


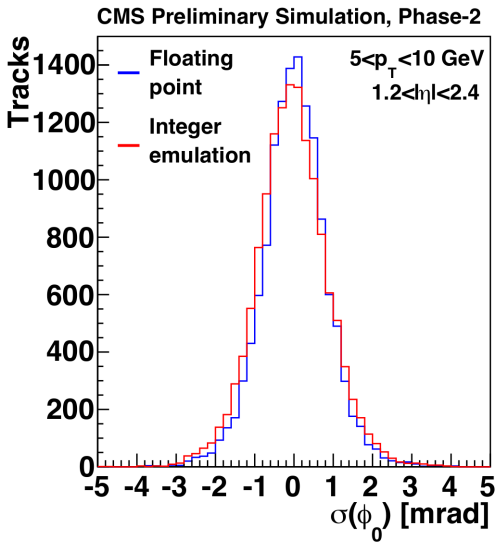
- Currently our implementation can comfortably process 36 stub pairs per 150 ns (factor of 6 time multiplexing)

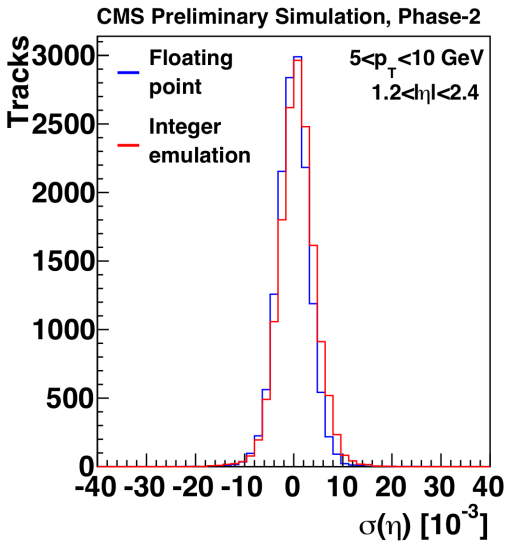
Redundancy from multiple seeding layers gives good tracking efficiency

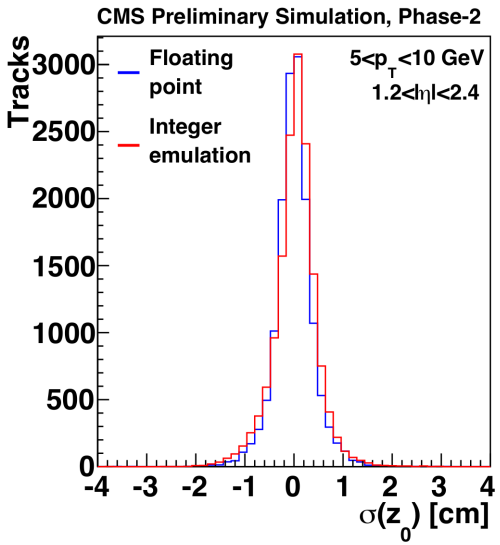
- Algorithm improvements will essentially eliminate truncation

Track Fit Performance





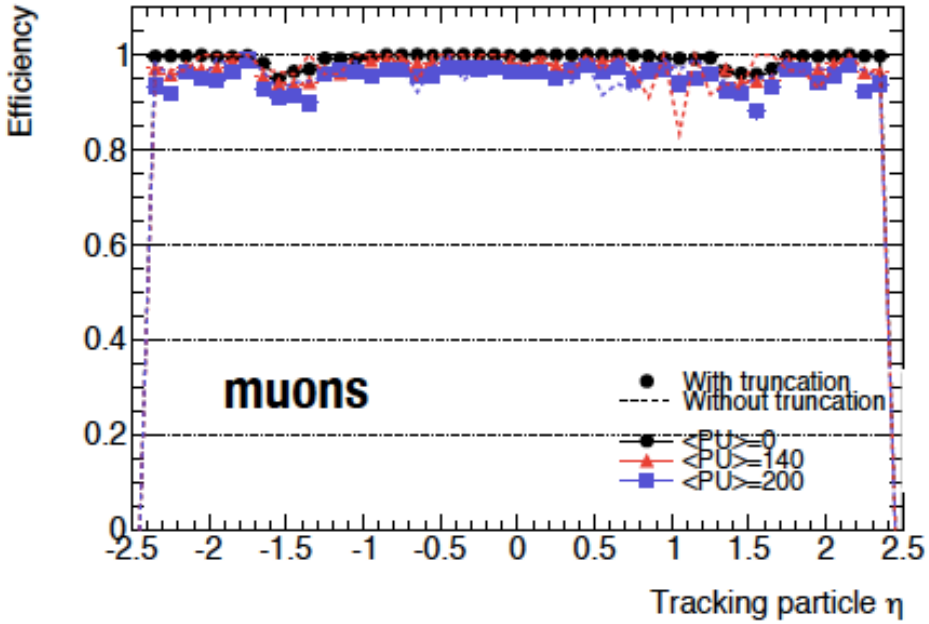


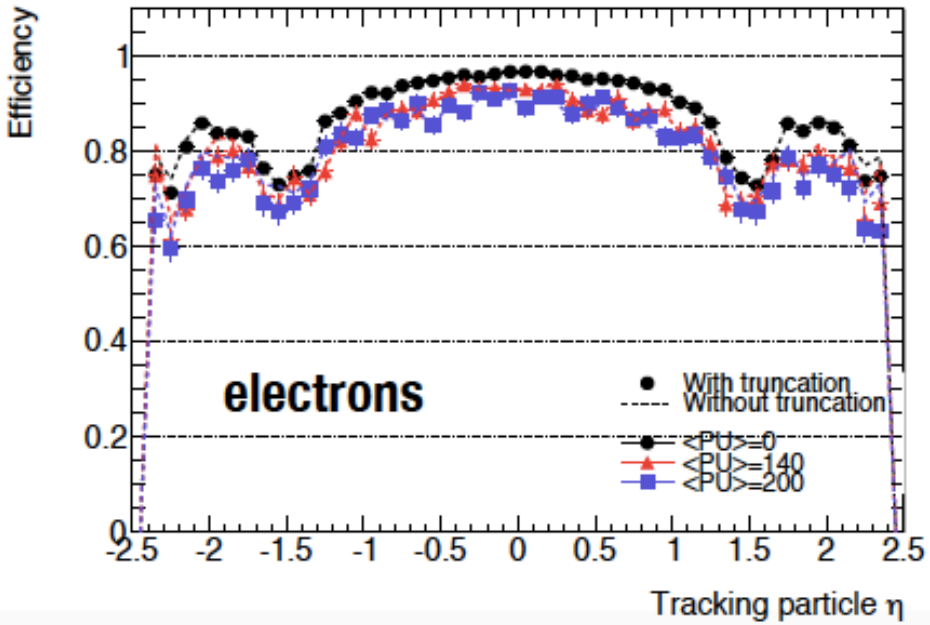


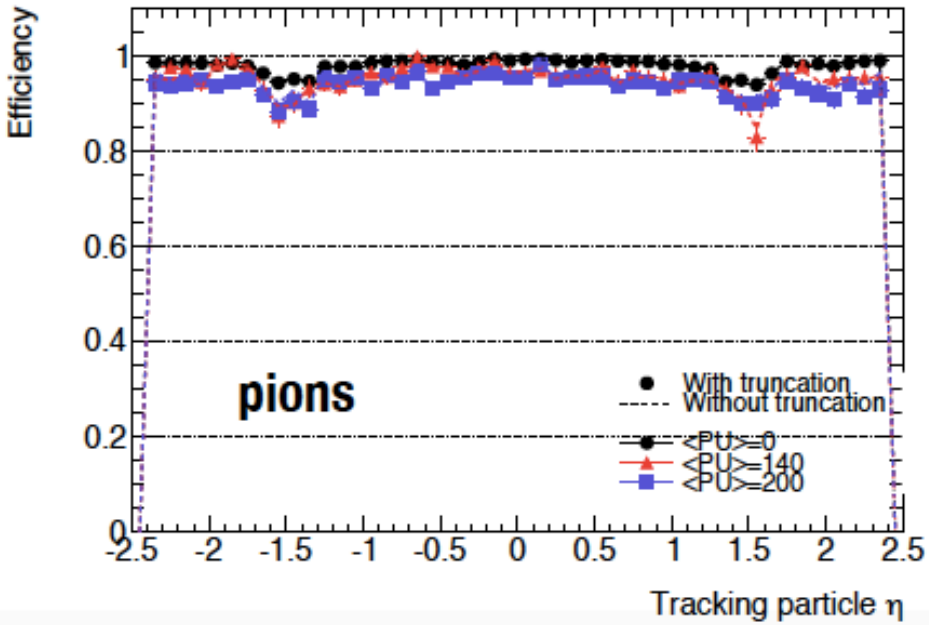
The integer performance and full floating point calculation gives the same track parameter resolution

Track Finding Efficiency

- Efficiency as function of η for single e/μ/π (p_T = 8-100 GeV)
  - Low statistics for PU=140 samples
  - Uses "default" (very tight!) MC truth matching
- High efficiencies obtained
- No impact of truncation

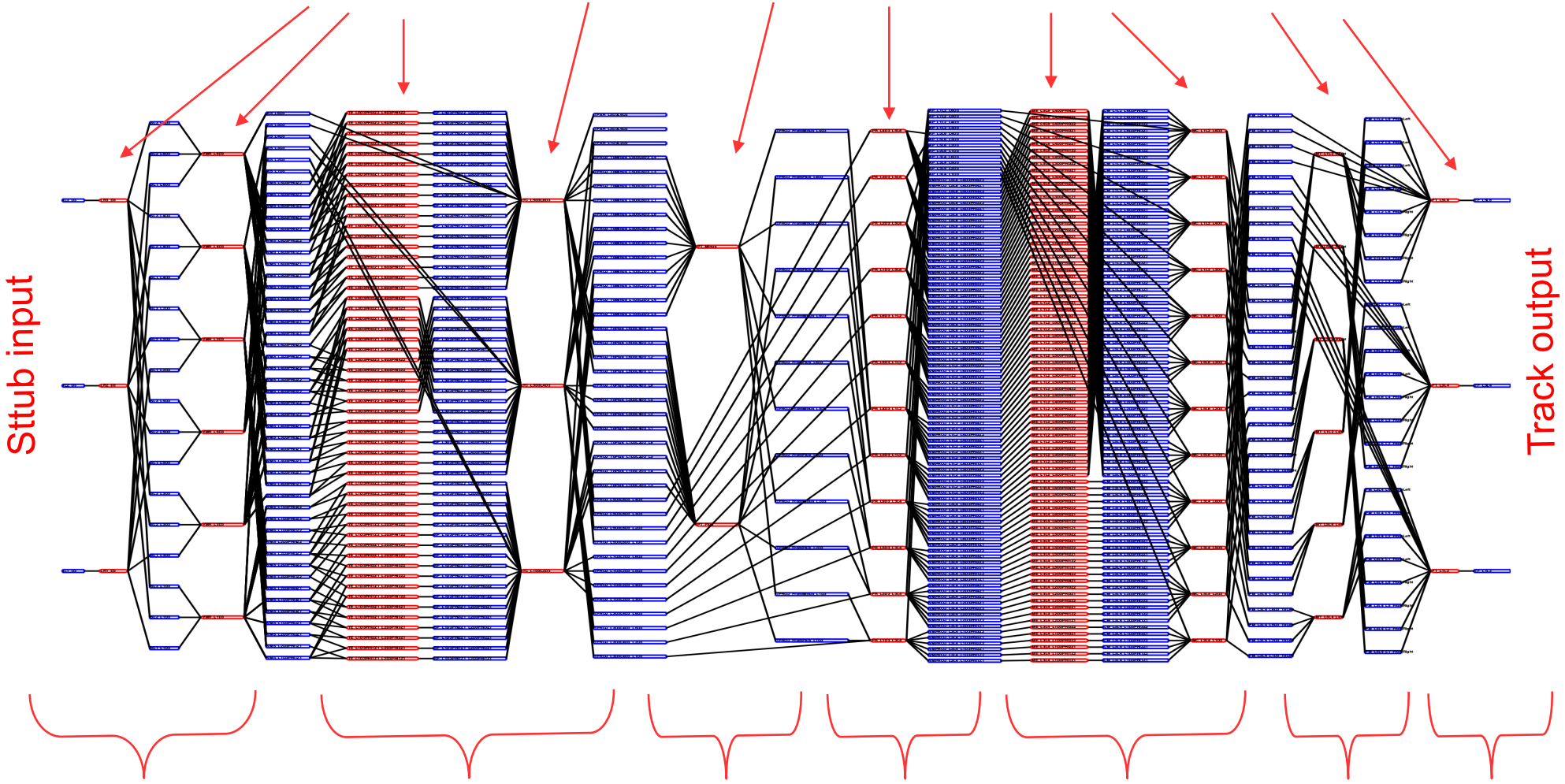






Overview of Firmware Project

Eight processing steps + two transmission (red) implements the algorithm



Stub organization

Forming tracklets

Projection transmission to neighbors

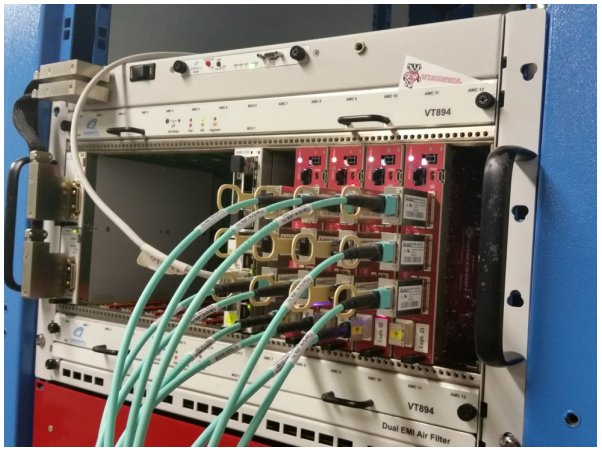
Organize tracklet projections

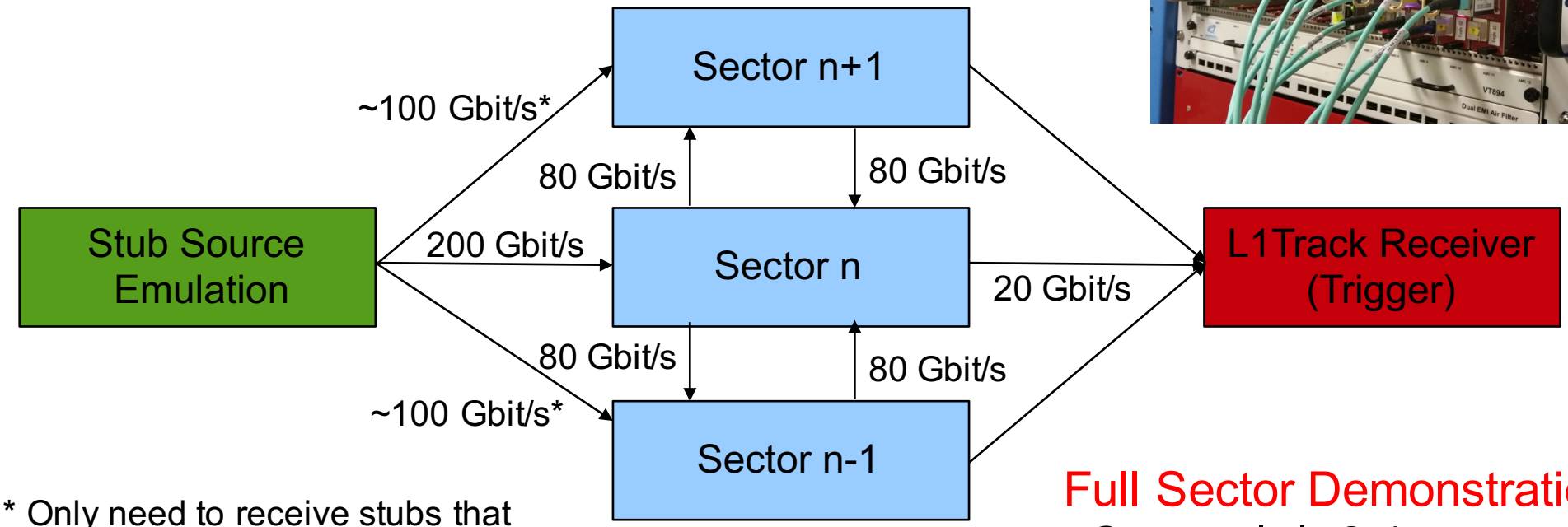
Match tracklet projections to stubs

Match trans-mission

Track fit

Tracklet Demonstrator





* Only need to receive stubs that needed to demonstrate functionality of the central sector

Same board used for stub source emulation and the L1Track receiver

Sector board
Input: 360 Gbit/s
Output: 180 Gbit/s

Full Sector Demonstration:
- Covers $|\eta|<2.4$
- Uses factor 4 time MUX
- $p_T>2$ GeV

Allow measurement of latency from stub input to tracks available

Latency Model

Step	Processing (ns)	Latency (clk)	Latency (ns)	Link latency (ns)	Total (ns)
Input link	0.0	1	4.2	316.7	320.8
Layer Router	150.0	1	4.2	0.0	154.2
VM Router	150.0	4	16.7	0.0	166.7
Tracklet Engine	150.0	5	20.8	0.0	170.8
Tracklet Calculator	150.0	43	179.2	0.0	329.2
Projection Transceiver	150.0	13	54.2	316.7	520.8
Projection Routing	150.0	5	20.8	0.0	170.8
Match Engine	150.0	6	25.0	0.0	175.0
Match Calculator	150.0	16	66.7	0.0	216.7
Match Transceiver	150.0	12	50.0	316.7	516.7
Track Fit	150.0	26	108.3	0.0	258.3
Dup Removal	0.0	6	25.0	0.0	25.0
Track Link	0.0	1	4.2	316.7	320.8
Total	1500.0	139	579.2	1266.7	**3345.8**

Latency Model

Step	Processing (ns)	Latency (clk)	Latency (ns)	Link latency (ns)	Total (ns)
Input link	0.0	1	4.2	316.7	320.8
Layer Router	150.0	1	4.2	0.0	154.2
VM Router	150.0	4	16.7	0.0	166.7
Tracklet Engine	150.0	5	20.8	0.0	170.8
Tracklet Calculator	150.0	43	179.2	0.0	329.2
Projection Transceiver	150.0	13	54.2	316.7	520.8
Projection Routing	150.0	5	20.8	0.0	170.8
Match Engine	150.0	6			
Match Calculator	150.0	16			
Match Transceiver	150.0	12			
Track Fit	150.0	26			
Dup Removal	0.0	6			
Track Link	0.0	1	4.2	316.7	320.8
Total	1500.0	139	579.2	1266.7	**3345.8**

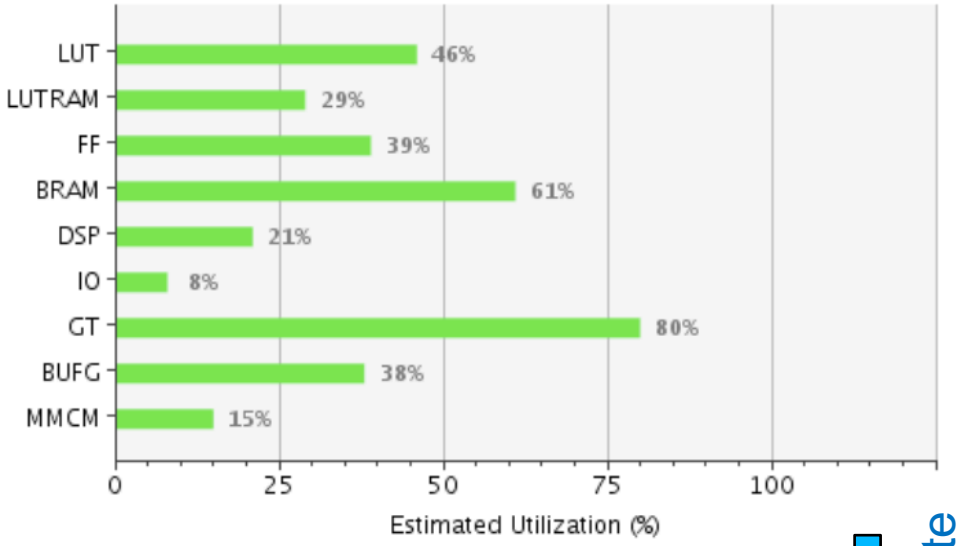
Measured Latency:
3333 ns

Agrees with model to 0.38% or three 240 MHz clocks

FPGA Resources

- From the resource usage in the half-sector project we can extrapolate to the full sector
  - The current Virtex-7 FPGAs dor have sufficient resources
  - Next generation Ultrascale+ FPGAs provide the needed resources
  - The cheaper Kintex FPGAs are also an interesting possibility

Half-Sector Resource Usage



	LUT Logic	LUT Memory	BRAM	DSP
Full sector	279733	151191	2721.5	1818
Virtex-7	65%	87%	185%	51%
VU3P	32%	81%	85%	80%
VU5P	21%	53%	58%	52%
VU7P	16%	40%	42%	40%
VU9P	11%	27%	28%	27%
VU11P	10%	27%	29%	20%
VU13P	7%	20%	22%	15%

Extrapolate

Future Improvements

- We are exploring a better detector segmentation (Virtual Module) configuration
  - Using narrow VMs in f that covers the full length of the detector allows a reduction in the combinatorics since jets are not contained in VM.
- Dynamic load balancing
  - Average resource utilization is low - resources assigned to handle tails. Dynamic load balance at the most parallel stages (Tracklet Engines and Match Engines) allow better resource utilization.
- Improved duplicate removal using χ^2 information
  - Select best track and optimize resource usage in duplicate removal
- Use of HLS (High-Level Synthesis) instead of Verilog
  - Simplify the development

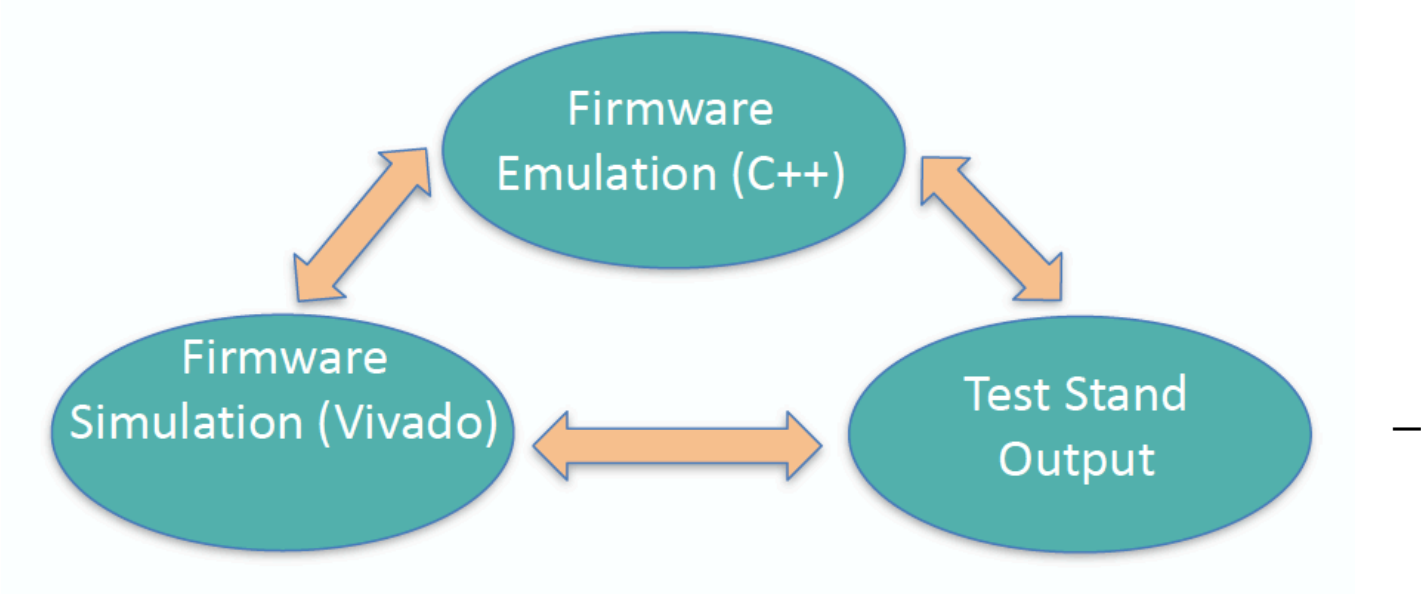
Summary/Outlook

- The tracklet approach – a road-search technique – implemented on FPGAs have been demonstrated
  - The approach meets the CMS requirements for latency and track performance
- The implementation provides robustness by seeding in parallel in different seeding layers
  - Further improvements are under study to implement load balancing to reduce truncation in tails and reduce FPGA resources needed
- New hardware is being developed for the next generation boards for the track trigger implementation
  - Will use Xilinx Ultrascale+ FPGAs and optical links at speeds up to 25 Gbits/s
- CMS is forming a common L1 tracking effort to define the reference track trigger project based on an all FPGA solution

Backup

Development

- We have a C++ simulation that reproduces bitwise the firmware.
  - Useful for algorithm development and physics performance studies.
- Implementation in Verilog (compared with C++)
  - FPGA firmware simulation
  - Execution on test stand



Seeding Redundancy