Science & Technology Facilities Council
Rutherford Appleton Laboratory

University of
BRISTOL

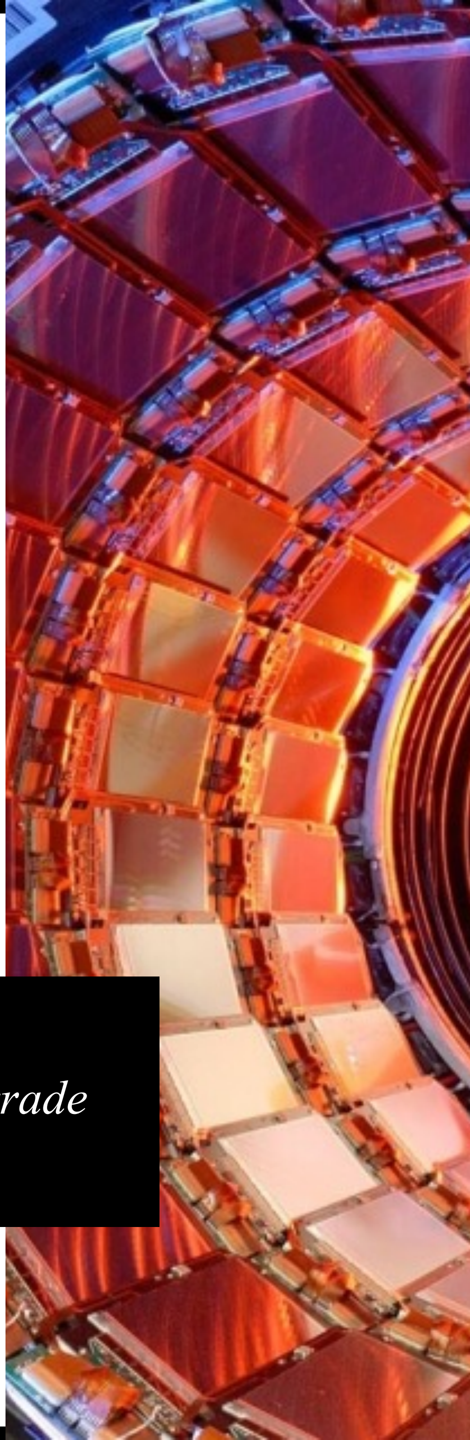
CMS
Compact Muon Solenoid

MEI-LI
HOLMBERG

LEVEL-1 TRACK FINDER

for the CMS HL-LHC upgrade

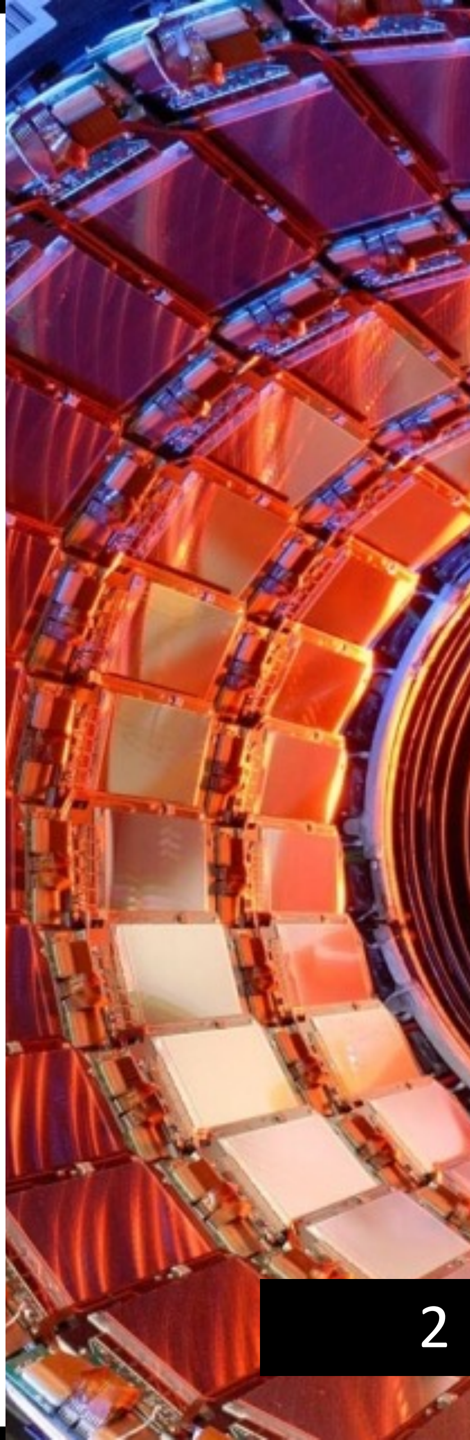
6 APRIL
2022



L1 TRACK FINDER

OVERVIEW

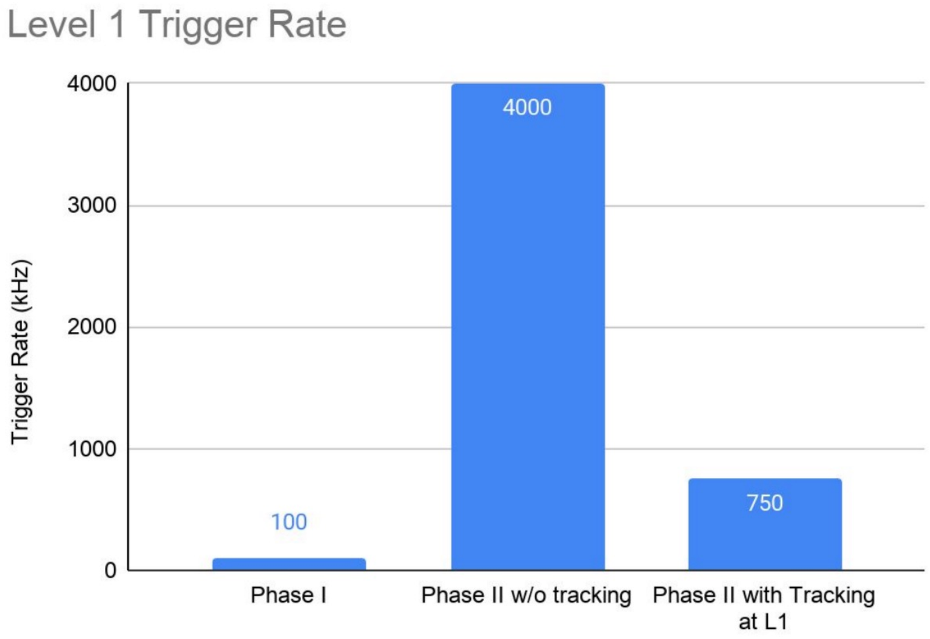
- Introduction
- Implementation
- Track Finding Algorithm
  - Overview
  - Virtual Modules
  - Project Design
- Algorithm Testing
  - Skinny Chain
  - Barrel-Only chain
- Summary

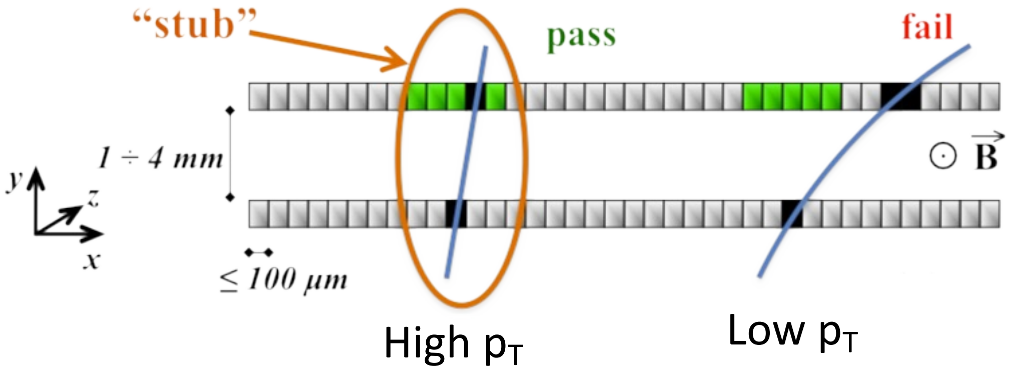


L1 TRACK FINDER

INTRODUCTION

- High-Luminosity LHC (HL-LHC) will increase the number of simultaneous proton-proton collisions
- CMS will include particle tracks at L1 trigger
  - Reduces L1 trigger rate from 4000 kHz to 750 kHz
- Outer tracker sensor filtering
  - Transmits hits from p_T > 2 GeV charged particles: Stubs
  - Reduces data rate by a factor ~20
- Track Finder reconstructs tracks of high energy particles for the L1 Trigger using stubs
  - ~200 tracks per event
  - p_T > 2 GeV
  - Done in 4 µs

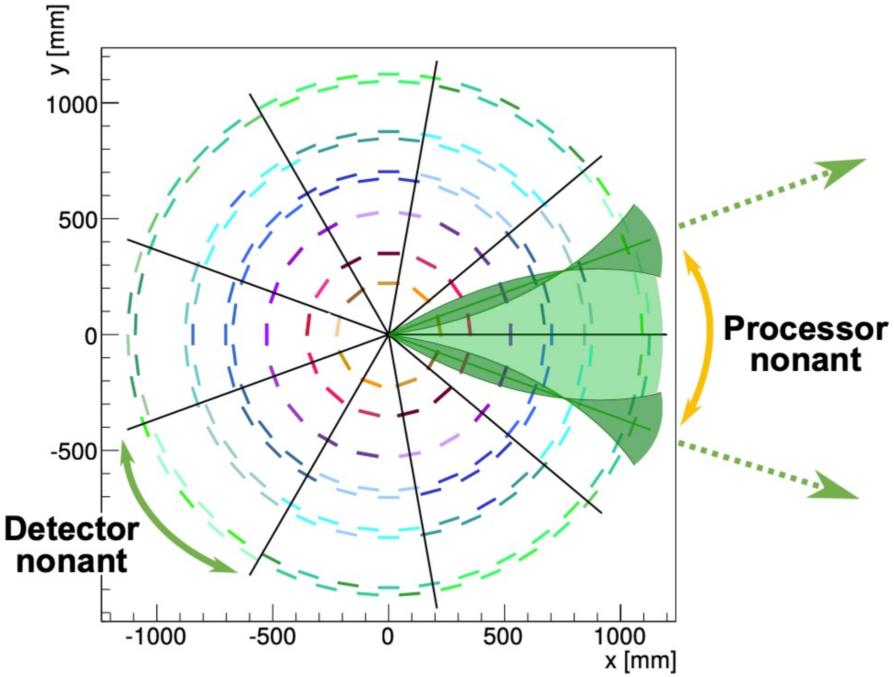




L1 TRACK FINDER

IMPLEMENTATION

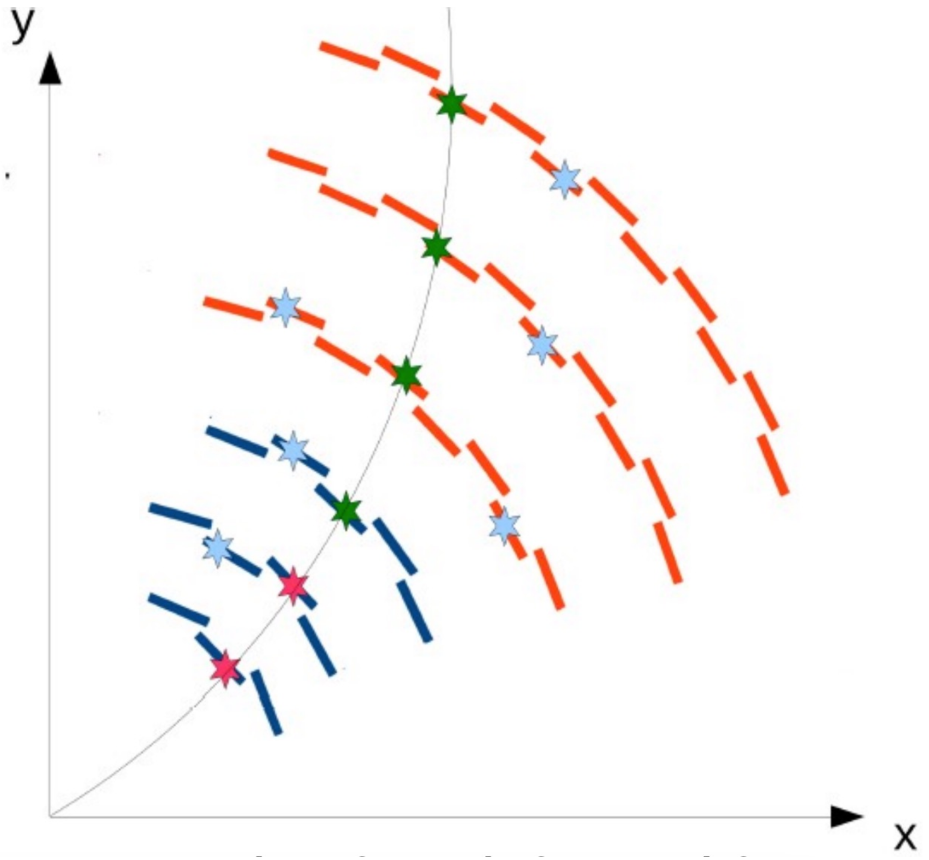
- Track finding algorithm will be implemented on FPGAs
  - Field-Programmable Gate Array (FPGA)
  - Very fast and programmable integrated circuits
  - Programmed using a hardware description language (HDL)
  - Mounted on Track Finder (TF) boards
- The Track Finder is split into 9 equally sized sectors in φ
  - 18 TF boards per nonant/sector, each processing different events
  - Data at the borders is copied to both neighbouring sectors
- No communication between sectors is necessary
  - Processes can run in parallel



TRACK FINDING ALGORITHM

OVERVIEW

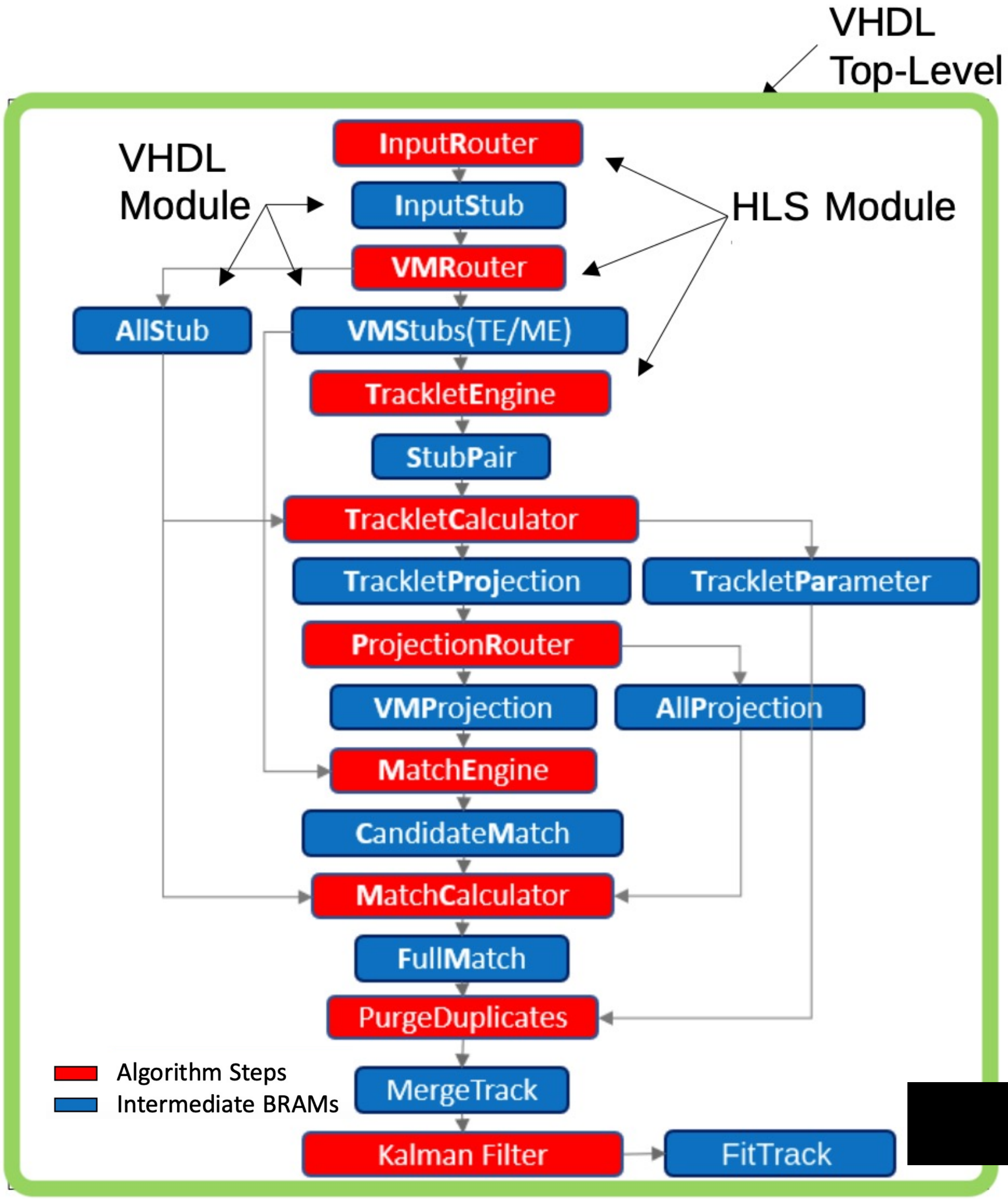
1. Take two stubs in adjacent layers and estimate track parameters
2. Project potential track to other layers
3. Look for stubs close to the track in the other layers
   - Reject tracks if not enough stubs were found in the other layers
4. Remove duplicate tracks
5. Use stubs and track candidate to calculate final track parameters
   - Using a Kalman Filter



TRACK FINDING ALGORITHM

PROJECT DESIGN

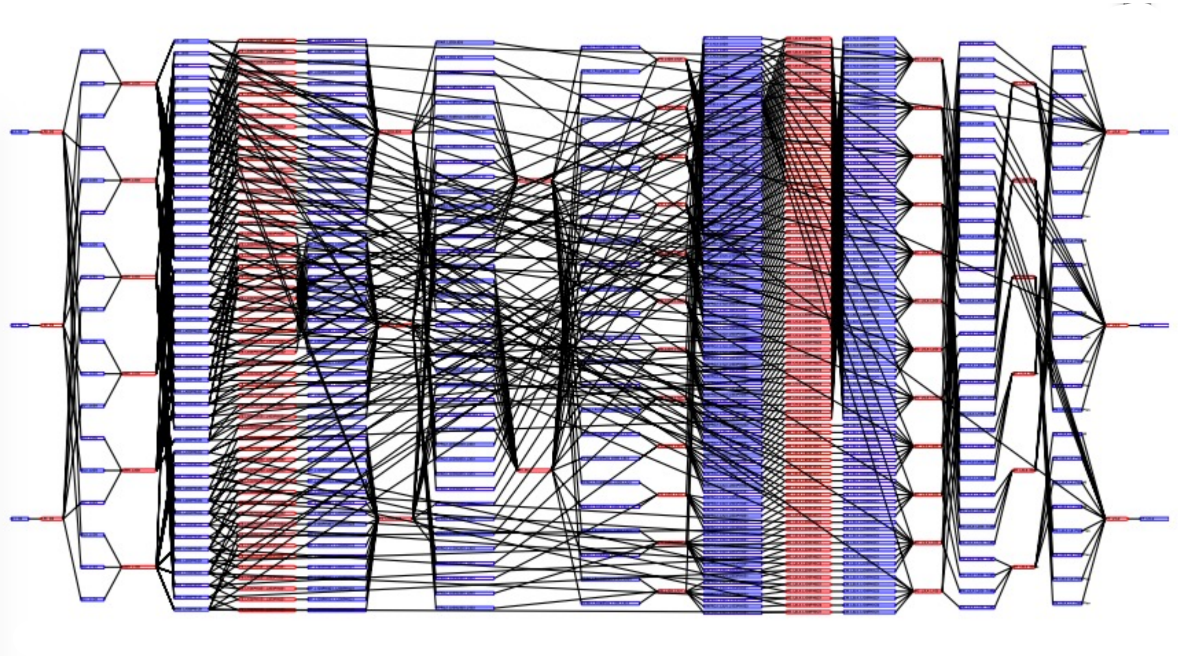
- Algorithm is split up into 9 steps/modules
  - Each processing module is implemented separately using HLS (except Kalman Filter)
  - HLS syntax is like C++
  - Process one event at a time
- Memories temporarily store the output of each module
  - Implemented in VHDL
- Top-level function connects the whole chain
  - Implemented in VHDL



TRACK FINDING ALGORITHM

PROJECT DESIGN

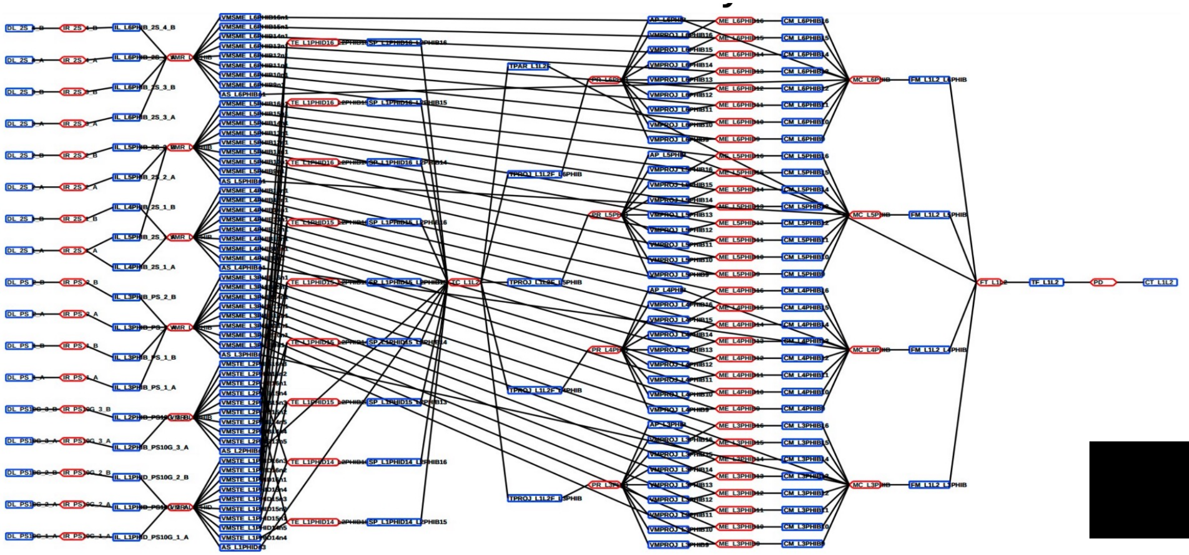
- Multiple versions of each processing module work in parallel
- Scripts create and synthesise the processing modules
- VHDL top-level that connects all the modules is written by a python script



ALGORITHM TESTING

SKINNY CHAIN

- Most firmware processing modules pass HLS simulations when run separately
  - Does not mean the full chain will work out of the box
- Implemented and tested a small slice of the algorithm
  - ~4% of the full project
  - Did not include the Duplicate Removal modules
- Ran hardware simulations using 1000 events*
  - 98% of events match emulation
  - Debugging in process
- Ran in hardware



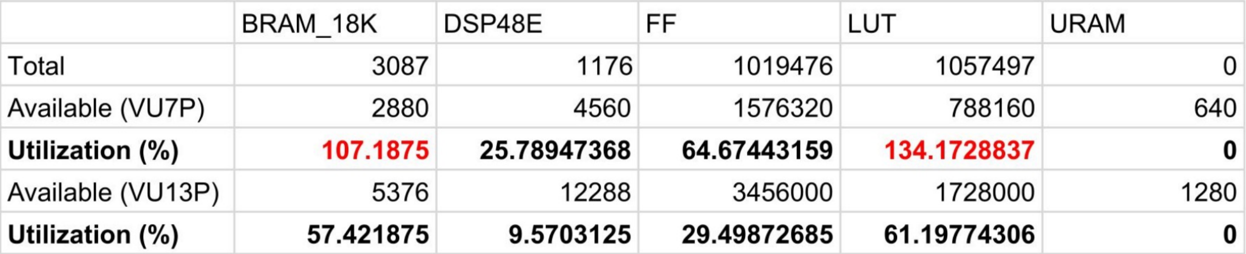
*ttbar 200 pile-up

ALGORITHM TESTING

BARREL-ONLY CHAIN

- Implement a chain with all the processing modules for the barrel
  - 2/3 of the full project
  - Does not include the Duplicate Removal modules and the Kalman Filter
- First good resource usage estimation
  - Uses too many LUTs and BRAMs
  - No URAMs currently used
  - Needs to be optimized
- No simulations yet…

Resource utilization (post-synthesis)

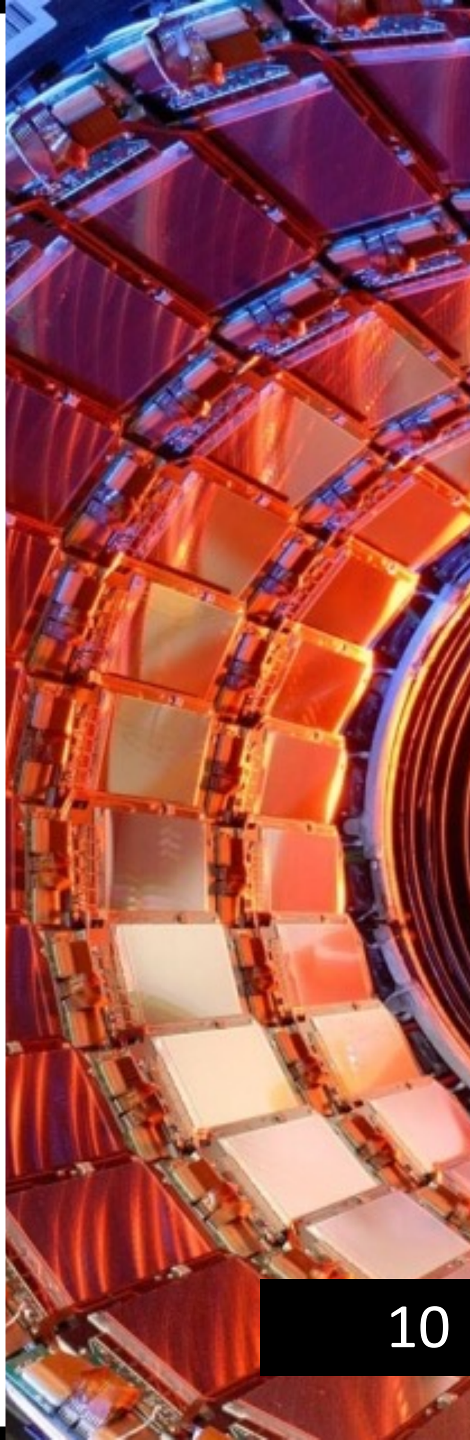


	BRAM_18K	DSP48E	FF	LUT	URAM
Total	3087	1176	1019476	1057497	0
Available (VU7P)	2880	4560	1576320	788160	640
Utilization (%)	**107.1875**	**25.78947368**	**64.67443159**	**134.1728837**	**0**
Available (VU13P)	5376	12288	3456000	1728000	1280
Utilization (%)	**57.421875**	**9.5703125**	**29.49872685**	**61.19774306**	**0**

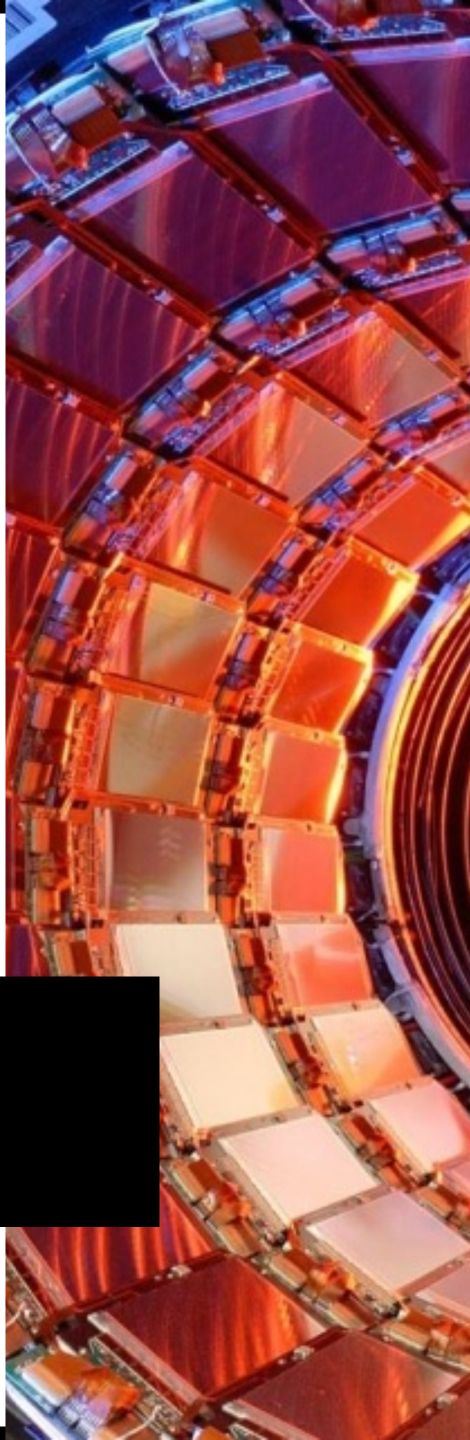
L1 TRACK FINDER

SUMMARY

- L1 Track Finding at CMS is necessary to reduce the L1 trigger rate to an acceptable level
- Algorithm modules have been individually tested
  - Pass simulations
  - Meet timing requirements
- Skinny chain has been successfully run on hardware
  - Debug the inconsistent outputs
- Barrel-Only chain has been synthesised
  - Optimise resource usage
- Scale the chain up to the full project in the future



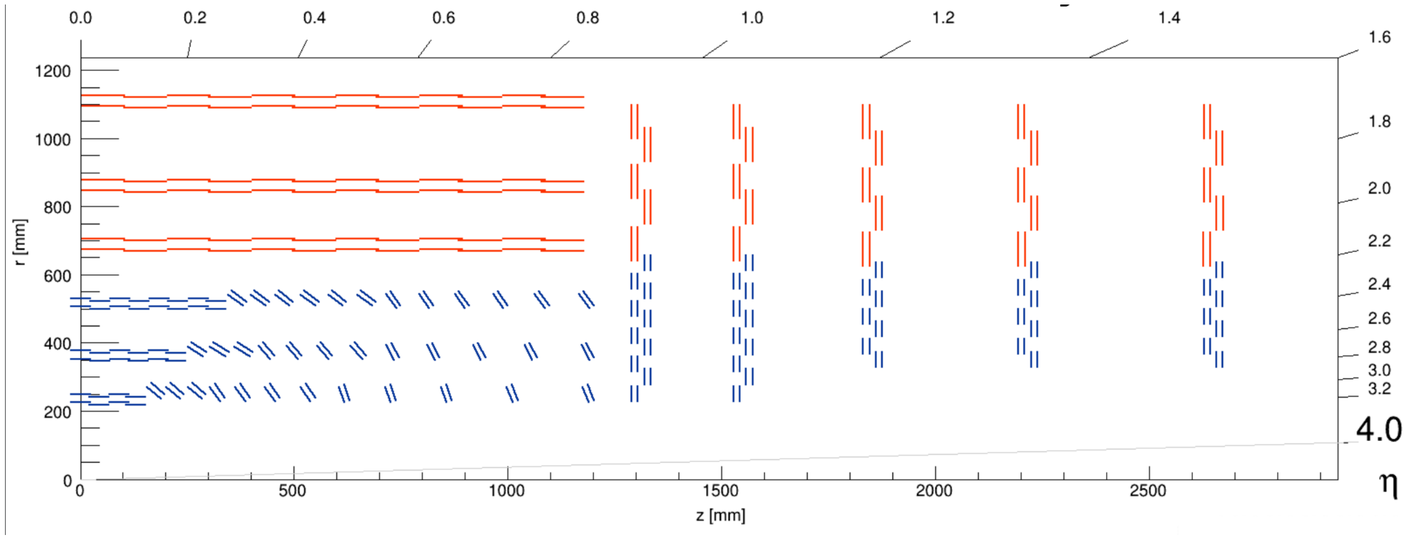
BACKUP SLIDES

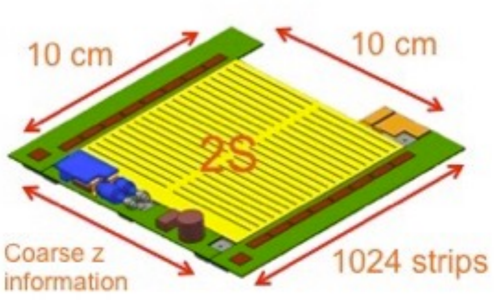


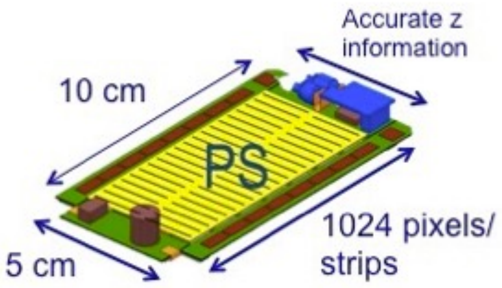
OUTER TRACKER

TWO-STRIP AND PIXEL-STRIP MODULES

- Two types of sensor pair modules are used for the Outer Tracker
- Two-Strip (2S) modules
  - Accurate information in φ but coarse in z
- Pixel-Strip (PS) modules
  - Accurate information in both φ and z



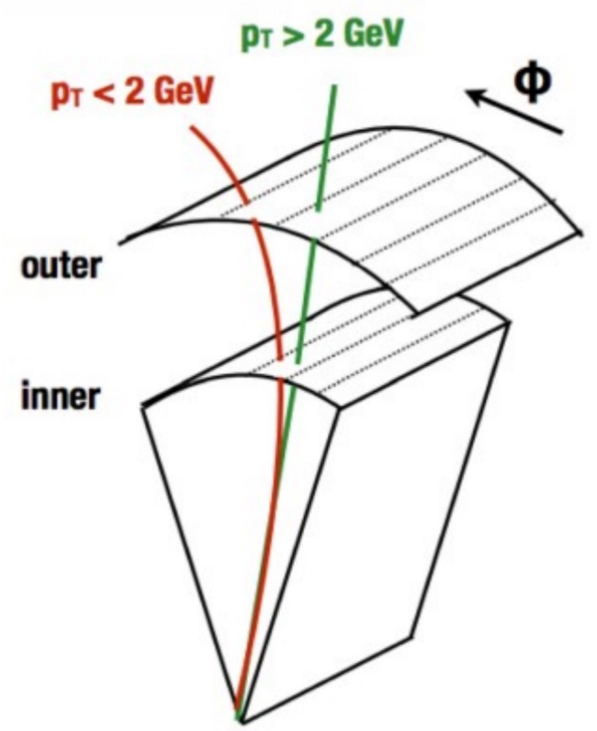




TRACK FINDING ALGORITHM

VIRTUAL MODULES

- Taking any two adjacent stubs in the first algorithm step results in large number of combinatorics
  - Inefficient as we are only interested in $p_T > 2$ GeV tracks
- Split each sector layers into slices in φ: Virtual Modules (VMs)
- Stubs in an inner layer VM are only compatible with some outer layer VMs
- In firmware the VMs are implemented as separate memories
  - Avoid having to go through lots of irrelevant stubs
- The VMRouter routes the stubs to the correct VM
  - Firmware written by me ☺



pT > 2 GeV
pT < 2 GeV
φ
outer
inner