PROCEEDINGS OF SCIENCE

CMS Level-1 Track Finder for the Phase-2 Upgrade

Christopher Brown[a,*] **on behalf of the CMS Collaboration**

[a] *Imperial College London,*
South Kensington, London, United Kingdom

E-mail: cebrown@cern.ch

The HL-LHC will increase the delivered luminosity to the CMS experiment allowing it to broaden its physics reach. The average number of pileup interactions will increase to 200 in the detector meaning the current trigger systems will be unable to maintain their selectivity for electroweak physics. To maintain manageable trigger rates, silicon-tracker based tracking at 40 MHz is to be included in the Level-1 trigger for the first time.

To achieve data rates low enough for track triggering the CMS outer tracker will use modules with closely-spaced silicon sensors which will only read out hits compatible with charged particles with a transverse momentum, p_T, above 2 GeV/c. These hits will be used in the backend Level-1 track finding system built on commercially available FPGA technology to reconstruct the charged particle tracks. The track finding algorithm is split into a track seeding stage forming tracklets from pairs of stubs and a Kalman Filter fitting algorithm to identify final track candidates and determine track parameters. The implementation of this algorithm in firmware is ongoing and a preliminary demonstration of the system in hardware is discussed.

The 32nd International Workshop on Vertex Detectors (VERTEX2023)
16-20 October 2023
Sestri Levante, Genova, Italy

*Speaker

PoS(VERTEX2023)022

1. Introduction

The Compact Muon Solenoid (CMS) experiment [1] is situated on the Large Hadron Collider (LHC) at CERN. It is one of the two general purpose detectors designed to reconstruct the proton-proton interactions created by the colliding LHC beams. Collisions of proton bunches occur at a maximum instantaneous rate of 40 MHz in the centre of the detector. Due to the large amount of data being created per event CMS cannot process and store all events, and hence uses a two-stage trigger system to filter events based on their physics potential. The first stage, the Level-1 (L1) trigger, is hardware-based running on Field Programmable Gate Arrays (FPGAs) with a fixed latency of $<$ 4 µs and an output rate of 100 kHz [2]. The second stage, the high-level trigger, runs on a CPU farm with an output rate of 1 kHz for final storage and offline analysis.

The High-Luminosity LHC (HL-LHC) is the next phase of the LHC upgrade programme where the instantaneous luminosity of the collider will be increased by seven times resulting in a overall increase of the integrated luminosity by a factor of ten by the end of the 2030s. The increase in delivered luminosity in the experiments means rare processes occur more frequently boosting the physics potential of the LHC experiments. However, a consequence of the increased luminosity is that the average number of proton-proton interactions occurring per bunch crossing (pileup) will increase from the current 60 to 200 in the ultimate scenario. Not only will the higher luminosity increase the radiation dose being received by the innermost parts of the detectors but poses a significant challenge to the data processing.

1.1 The CMS Phase-2 Upgrade

CMS will be upgraded to take advantage of this higher luminosity [3]. The Phase-2 upgrade programme covers upgrades and replacements of a wide range of different detector components and data processing systems. The innermost detector, the tracker, will be replaced with higher radiation tolerant materials with a higher η coverage, schematically shown in Fig 1, and a redesign of the outer tracker modules. These modules will consist of two closely-spaced silicon layers with a tuneable window that correlates simultaneous hits in both layers [5]. This tuneable window will be used to filter detector hits coming from charged tracks with a $p_T > 2$ GeV/c and create "stubs" in the detector. At the stub level, the data rate from the modules is reduced by a factor of ten, meaning that the data bandwidth from the outer tracker is low enough for tracks to be reconstructed at the highest expected instantaneous collision rate of 40 MHz in the L1 trigger. To incorporate this track finding the L1 trigger is also being fully replaced with an increased output rate of 750 KHz and a longer latency of $<$ 12.5 µs [6]. The system will be fully implemented on large resource-abundant Xilinx Ultrascale+ VU13P FPGAs giving increased flexibility to the trigger system. Algorithms such as primary vertex finding [7] and particle flow [8] will be possible using these tracks giving the L1 trigger far better discrimination between background pileup and potentially interesting physics signatures. This will not only allow the trigger system to maintain its physics selectivity in the increased pileup but potentially open new pathways for triggering on rare physics signatures otherwise unobtainable by the CMS trigger system.

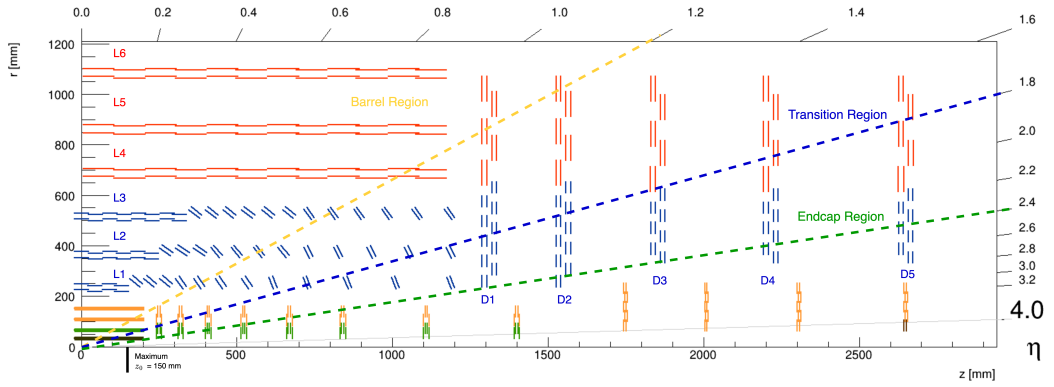


Figure 1: A slice in r—z of the Phase-2 CMS tracker, adapted from [4]. In yellow and green are the inner tracker modules. In blue, the pixel-strip modules, and in red, the two-strip modules both using the p_T-module concept. The barrel layers are labelled L1 through L6 in increasing r and the disk layers are labelled D1 through D5 in increasing z. The barrel ($0 \leq |\eta| < 1.2$), transition ($1.2 \leq |\eta| < 1.8$), and endcap ($1.8 \leq |\eta| < 2.4$) regions of the outer tracker are also shown.

2. System Design

The L1 track finder will need to be able to reconstruct $O(100)$ charged particles tracks from $O(10{,}000)$ outer tracker stubs at the full 40 MHz bunch crossing rate. It needs to have high efficiency (> 95%) for the full outer tracker coverage between $-2.4 \leq \eta \leq 2.4$ and a $p_T > 2$ GeV/c. It needs flat efficiency in the azimuthal angle ϕ and in a beam spot region of $-15 \leq z_0 \leq 15$ cm. Finally it needs to have a low fake rate, which is defined as the rate of tracks not originating from a true charged particles but instead from unrelated combinations of stubs in the detector.

In order to provide the L1 trigger with tracks the algorithm must run in 4 µs and be implemented in firmware to run on a VU13P FPGA. As the number of stubs coming from the full outer tracker is too high for a single FPGA to deal with the algorithm is parallelised both regionally and in a time-multiplexed manner. Firstly, the detector is divided into nine ϕ regions or "nonants" with hourglass-shaped overlap regions with duplicated stubs to avoid any need for cross communication between nonant processing. These nonants are read out by a detector trigger and control board which performs stub pre-processing and routing to the track finder boards. Secondly, the processing is time-multiplexed by 18 meaning a new event is processed by a single board every 450 ns with 18 track finder processor boards per detector nonant.

3. Track Reconstruction

The track reconstruction chain is divided into two main sections. The first is a "tracklet" pattern matching step that finds track candidates from pairs of stubs [9] followed by a Kalman Filter (KF) track fit [10]. After the KF a track quality boosted decision tree (BDT) (described in section 3.3) is used to evaluate the quality of tracks before being sent downstream to the rest of the L1 trigger.

3.1 Tracklet Pattern Matching

The tracklet algorithm is split into four stages shown in Fig. 2. The first is an initial seeding using pairs of seeding stubs in different layers of the outer tracker to form a two-stub tracklet. The

layer combinations that are used are L1+L2, L3+L4, L5+L6, L1+D1, L2+D1, D1+D2, D3+D4 and L2+L3 (with the layer numbering corresponding to that in Fig. 1). This set of seeds provides a wide number of options across the outer tracker in order to maximise efficiency.

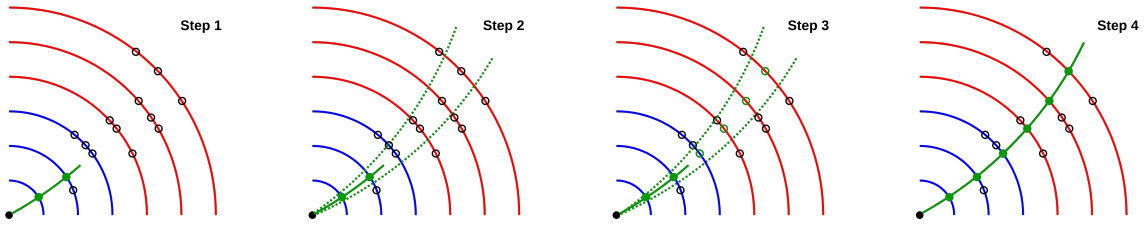


Figure 2: The tracklet stage of track reconstruction. Step 1 shows the initial seeding stubs and their tracklet. Step 2 shows the projection of this tracklet's parameters out to the other layers. Step 3 shows the selection of stubs (now highlighted green) based on their position in the window. Step 4 shows the final track for sending to the KF.

The second step of the algorithm (step 2 in Fig. 2) is to project track candidates both outwards and inwards from the tracklet with a beam spot constraint ($x, y = 0$) to form a search window in each layer.

Step 3 adds stubs to the track candidate and in cases with multiple matches takes the stub with the smallest residual. This process adds a minimum of four stubs but a maximum of six stubs. At this stage any track candidates with shared stubs are merged to avoid additional processing for the KF and to reduce the duplication rate of the algorithm.

Step 4 sees the final track candidate with its associated stubs being passed downstream to the KF after some reparameterisation.

3.1.1 Firmware Implementation

The tracklet algorithm is implemented in Vivado HLS due to the flexibility of using the C++ interface to develop the firmware. The algorithm is split into 16 alternating processing and memory modules which allows tracks from multiple events to be processed simultaneously with different events being processed at different stages of the chain at any given moment. Individual processing modules are duplicated so that track candidates can be processed in parallel, for example, all eight stub seeding pairs are processed simultaneously and while this does create duplicates due to the overlap between certain stub seeding combinations it ensures no loss of tracks due to truncation.

In order to reduce the overall latency and resource usage of the tracklet firmware some modules are combined allowing 450 ns to be saved for every merged module as one fewer intermediate memory is implemented. This also reduces the number of Block RAMs needed by the algorithm.

3.2 Kalman Filter Track Fit

The Kalman Filter receives the track candidates to form a KF state and covariance matrix. Associated stubs are also sent and used to iteratively update the KF state. There are two steps to the KF; the prediction step and the filter step. The prediction step multiplies the state and covariance matrix by a state transition matrix to produce an estimate of the track parameters in the next layer. The filter step then uses the stub information to update the state and reduces the errors of the state in the covariance matrix so that the next step update will be a better estimate of the track parameters

in the next layer. From the covariance matrix goodness-of-fit χ^2 values in the r—ϕ and r—z planes can be derived. An advantage of the KF is that if these χ^2 values become too large a track can be killed before it is fully fit preventing fake tracks from being created. This process occurs for each of the stubs up to the maximum of six to form well fit track parameters.

In this implementation the KF state, x, is a four parameter vector:

$$x = (\frac{1}{2R}, \phi_0, \cot\theta, z_0), \tag{1}$$

where R is the radius of curvature of the track ($R = p_{\mathrm{T}}/(0.003qB)$, with q being the charge of the particle in units of e, and B being the CMS magnetic field strength; p_T, B, and R are measured in units of GeV/c, Tesla and cm respectively). θ is the polar angle of the track related to the more common pseudorapidity η as $\eta = -\ln(\tan(\frac{\theta}{2}))$. ϕ_0 and z_0 are the track azimuthal angle, in radians, and longitudinal impact parameter, in cm, respectively. The state is projected through layers with a stepping parameter r which is the radius of the layer in which the next stub is found. This forms the following linear approximation state project equations:

$$\phi = \frac{r}{2R} + \phi_0 \qquad z = (\cot(\theta))r + z_0, \tag{2}$$

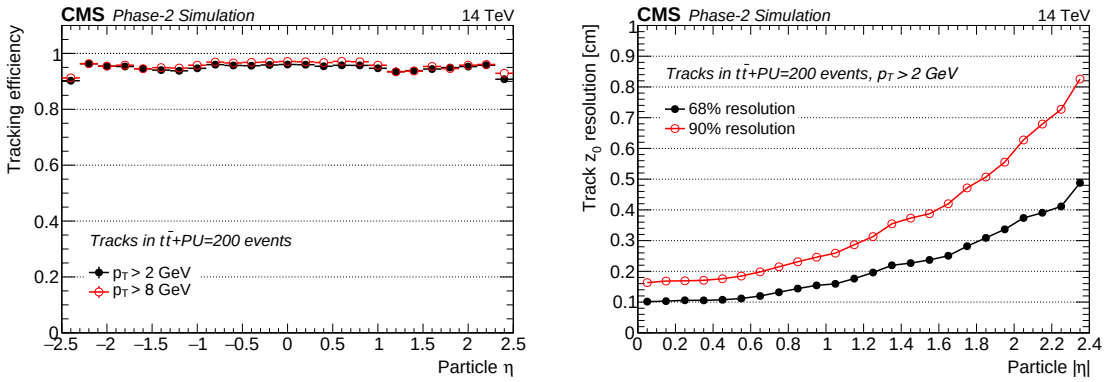
with ϕ and z being the track parameters at the radius r. This set of equations is linear in r motivating the use of R and θ instead of the more physical $\mathrm{p_T}$ and η. One of the advantages of a KF is that the track fit only has four parameters forming small 4×4 matrices whose size are independent of the number of measurements and simpler to implement in firmware.

The overall performance of the hybrid algorithm is shown in Fig.3a with flat high (> 95%) efficiency across η with the only small dips occurring at $\eta = 1.0$ because of the transition region between the barrel and endcap where on average fewer layers will be seen by a particle passing through. The algorithm has a cut on tracks of $|\eta| > 2.4$ causing the apparent dips in efficiency in the high η bins. The aim of the downstream vertex finding is to distinguish vertices across a 30 cm z_0 range in 200 pileup (PU) meaning a track z_0 resolution of at most 1.5 mm is needed. Fig.3b shows that this requirement is met by the track finder in the barrel region but the resolution worsens as η increases. This worsening of resolution is unavoidable with the tracker geometry where the fine grained measurement of η in the endcaps cannot provide high resolution in z_0. This is accounted for by downstream track to vertex association algorithms.

3.2.1 Firmware Implementation

The KF is written directly in VHDL due to the relative simplicity of the algorithm in comparison to the tracklet seeding stage and the advantage of having fine control over clock-level processing. The KF is generally split into two main parts; a state updating block and control logic. The control logic deals with processing stub information and passing it into the state updater as well as storing and selecting the best states. The state updater performs the matrix calculations required to propagate the state so is the highest latency and most resource intensive part of the algorithm.

The KF is parallellised into KF workers which each deal with a different seeding input, these workers run in parallel before sending their fitted tracks to a router for distributing to the output to the L1 trigger. This internal parallelism prevents track loss due to truncation.



(a) **(b)**

Figure 3: Figure (a) shows the track finding efficiency versus η for tracks in $t\bar{t}$ events with 200 PU. High (> 95%) track finding efficiency is seen with slightly improved efficiency for tracks with a p_T > 8 GeV/c. Figure (b) shows the track z_0 resolution as a function of η demonstrating a worsening resolution in the endcap region due to the orientation of the tracker modules. The resolution is calculated by binning tracks in η and fitting the residual of reconstructed track z_0 - true z_0 to a Gaussian. The 68% and 90% quartiles are shown for each η bin.

3.3 Track Quality

An inevitable part of any track reconstruction in high multiplicity environments is the creation of tracks from unrelated combinations of hits in the detector. Typically, the fit of these "fake" tracks will be worse (with larger values of the χ^2 goodness-of-fit variables) as the algorithm has had to fit a line to a set of points that do not sit on the expected helical path. An output of the KF fitting stage are χ^2 fit estimators in the r—ϕ and r—z planes as well as an additional χ^2_{bend} that is formed from how well the fitted track curvature matches the track curvature estimate in the individual stubs. The simplest method of removing poor quality tracks is to cut on their χ^2 parameters, however this can result in inefficiencies in geometrical regions due to the track fitting procedure slightly differing between barrel, endcap and transition regions.

Instead a boosted decision tree (BDT) is used for evaluating track quality due to its simplicity in firmware and better performance in comparison to using simple χ^2 cuts. A small ensemble of 60 trees with a depth of three was trained using $t\bar{t}$ Monte Carlo events with an additional 200 soft PU interactions. The BDT uses the track z_0, $\cot\theta$, χ^2 variables, number of stubs and the number of missing internal stubs (the number of layers skipped in the fitted track) as input features. The BDT outperforms basic cuts using just χ^2 as demonstrated by Fig. 4 where the BDT has higher track finding efficiency at the same fake track rate. An advantage of the BDT approach is the provision of a quality estimator, which can be used optimally for track selection by downstream algorithms.

3.3.1 Firmware Implemenentaion

The BDT is implemented in VHDL using the conifer package [11] which fully unrolls the BDT so that all 60 trees are evaluated in parallel with a weighting based on their weighting within the ensemble. The BDT is implemented after the KF meaning full track parameters can be provided

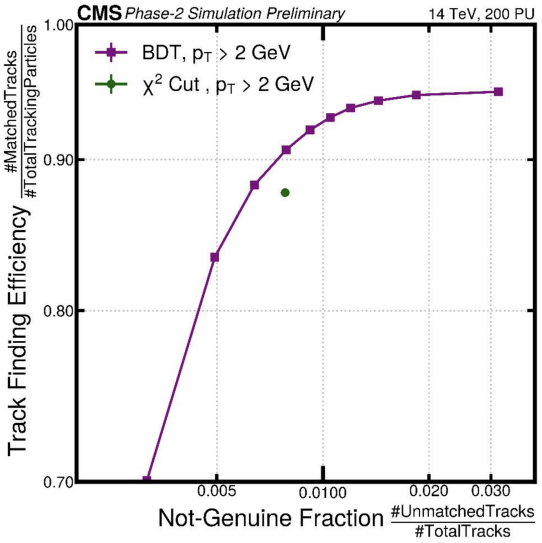


Figure 4: Track finding efficiency versus the non-genuine fraction (fake rate) for tracks after quality selection. In purple the BDT approach outperforms a simple χ^2 cut in green (this specific cut is used for downstream missing E_T calculations) with higher tracker finding efficiency for the same non-genuine fraction.

by the KF and the BDT only needs to be replicated twice lowering its resource requirements. The BDT uses fixed-point numbers at a lower granularity than is provided by the KF in order to reduce resource usage.

4. Hardware Demonstration

In order to test the firmware implementation of the algorithm various hardware tests are performed in gradually increasing levels of complexity. It also provides opportunity to trial the processing board hardware, in the case of the tracker this is the Apollo ATCA platform [12]. The first full chain test was using a reduced set of inputs to simplify the processing. Only a single narrow slice of the detector was trialled with a single tracklet seed and no combined modules. This test proved successful with the algorithm fitting easily within the target FGPA (a Virtex UltraScale+ VU7P), a latency of 3.4 µs, and matching to the emulation to a greater than 99% success rate. This firmware was also used to test the link between the track finder and the L1 trigger with testing performed between two separate boards running different firmware, one reconstructing the tracks from stub information and the second running a vertex finding algorithm using the tracks. Again, this test was successful and demonstrates the validity of the firmware both in the tracks being created but also meeting the firmware and latency requirements.

Work is underway scaling up these tests. The first step is to scale up the track finder firmware to include the entire barrel project which creates certain difficulties due to the size and complexity of the firmware. The target FPGA, the Xilinx Ultrascale+ VU13P requires careful floorplanning to ensure the firmware meets timing requirements. There is also additional optimisation of the HLS modules to reduce their complexity with some modules being rewritten in VHDL as well as the use of combined modules to simplify the firmware.

5. Conclusion

The development of a track trigger for the HL-LHC is vital for the CMS Level-1 trigger to maintain its physics selectivity. The use of an on-detector p_{T} cut combined with both regional division into nonants and time-multiplexing allow track finding to be performed on commercially available FPGAs. The track finding algorithm is split into two main parts. A tracklet seeding step uses pair of detector hits to form tracklets which are projected into the other layers to find additional matching hits. This part of the algorithm is implemented in HLS with an alternating processing-memory design.

The second step, a Kalman Filter, follows the tracklet seeding step to determine track parameters. This step is implemented in VHDL and currently scales well in different hardware tests. The quality of the resultant tracks is evaluated by a boosted decision tree due to its classification power over simple goodness-of-fit parameter cuts. The BDT has also been implemented in VHDL using a fully parallel implementation to reduce latency.

The development and implementation of the full algorithm is entering its final stages with the main target over the coming year to implement the full algorithm on the target hardware and for testing of both the standalone algorithm and wider trigger system to be ramped up.

References

[1] The CMS Collaboration, The CMS Experiment at the CERN LHC. The Compact Muon Solenoid Experiment, JINST **3** (2008) S08004

[2] The CMS Collaboration, Performance of the CMS Level-1 trigger in proton-proton collisions at $\sqrt{s}$ = 13 TeV, JINST **15** (2020) P10017

[3] D. Contardo et al., Technical Proposal for the Phase-II Upgrade of the CMS Detector, tech. rep., 2015, CERN-LHCC-2015-010, https://cds.cern.ch/record/2020886

[4] The CMS Collaboration, The Phase-2 Upgrade of the CMS Tracker Technical Design Report, tech. rep., 2017, CERN-LHCC-2017-009, https://cds.cern.ch/record/2272264

[5] G. Hall, M. Raymond, and A. Rose, 2-D PT module concept for the SLHC CMS tracker, JINST **5** (2010) C07012

[6] The CMS Collaboration, The Phase-2 Upgrade of the CMS Level-1 Trigger, tech. rep., 2020, CERN-LHCC-2020-004, https://cds.cern.ch/record/2714892

[7] C. Brown et al, Neural Network-Based Primary Vertex Reconstruction with FPGAs for the Upgrade of the CMS Level-1 Trigger System, J. Phys.: Conf. Ser. **2438** (2023) 012106

[8] C. Herwig on behalf of the CMS collaboration, Particle flow reconstruction for the CMS Phase-II Level-1 Trigger, JINST **18** (2023) C01037

[9] E. Bartz, et al., FPGA-based Tracking for the CMS Level-1 Trigger Using the Tracklet Algorithm, JINST **15** (2020) P06024

[10] R. Aggleton, et al., An FPGA Based Track Finder for the L1 Trigger of the CMS experiment at the High Luminosity LHC, JINST **12** (2017) P12019

[11] S. Summers et al., Fast Inference of Boosted Decision Trees in FPGAs for Particle Physics, JINST **15** (2020) P05026

[12] E. S. Hazen et al., The APOLLO ATCA Platform PoS(TWEPP2019) **370** (2020) 120