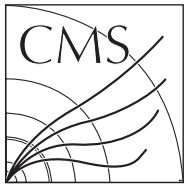
Available on CMS information server

CMS CR -2022/278



The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

12 December 2022 (v4, 16 December 2022)

The Level-1 Track Finder for the CMS High-Luminosity LHC Upgrade

Mei-Li Holmberg on behalf the CMS Collaboration

Abstract

The upgrade of the Large Hadron Collider (LHC) to the High-Luminosity LHC (HL-LHC) will provide an increased instantaneous luminosity, yielding more data for physics analysis, but imposing greater demands on the triggering systems of the detectors. In response, the CMS Collaboration is designing a novel Level-1 (L1) hardware track trigger using data from the Outer Tracker. This paper describes an overview of the L1 track finder which runs a pattern recognition algorithm on programmable chips (FPGAs). It aims to reconstruct trajectories of all charged particles with a transverse momentum (p_{T}) greater than 2 GeV within 4 μs. L1 track reconstruction using tracker data is a novel technique at CMS and has been made possible thanks to the innovative design of the new Outer Tracker. The Outer Tracker sensor modules transmit only the hits compatible with particles of $p_{\mathrm{T}} > 2$ GeV to the off-detector track-finder electronics, which in turn sends the reconstructed tracks to the L1 trigger. The track-finding algorithm has been implemented in firmware, and smaller tests of the firmware have successfully been performed in both simulations and hardware. The current state of the L1 track finder will be discussed.

Presented at *Vertex2022 The 31st International Workshop on Vertex Detectors*

The Level-1 Track Finder for the CMS High-Luminosity LHC Upgrade

Mei-Li Holmberg[1,2] on behalf of the CMS Collaboration

[1]*Rutherford Appleton Laboratory, Oxfordshire, United Kingdom*
[2]*University of Bristol, Bristol, United Kingdom*

E-mail: mei-li.holmberg@cern.ch

(Received December XX, 2022)

Abstract: The upgrade of the Large Hadron Collider (LHC) to the High-Luminosity LHC (HL-LHC) will provide an increased instantaneous luminosity, yielding more data for physics analysis, but imposing greater demands on the triggering systems of the detectors. In response, the CMS Collaboration is designing a novel Level-1 (L1) hardware track trigger using data from the Outer Tracker. This paper describes an overview of the L1 track finder which runs a pattern recognition algorithm on programmable chips (FPGAs). Its aim is to reconstruct trajectories of all charged particles with a transverse momentum (p_{T}) greater than 2 GeV within 4 µs. Track reconstruction at L1 is a novel technique at CMS and has been made possible thanks to the innovative design of the new Outer Tracker. The Outer Tracker sensor modules transmit only the hits compatible with particles of $p_{\mathrm{T}} > 2$ GeV to the off-detector track-finder electronics, which in turn sends the reconstructed tracks to the L1 trigger. The track-finding algorithm has been implemented in firmware, and smaller tests of the firmware have successfully been performed in both simulations and hardware. The current state of the L1 track finder will be discussed.

KEYWORDS: Pattern recognition, Low-latency track finding, Level-1 trigger.

1. Introduction

The Compact Muon Solenoid (CMS) [1] is one of the two multipurpose detectors at CERN's Large Hadron Collider (LHC) [2]. The CMS detector is designed for precision measurements of particles and physics objects, such as leptons, photons, and jets, mainly from proton-proton collisions. In the upcoming years, the LHC will be upgraded to the High-Luminosity LHC (HL-LHC) [3]; an upgrade that will increase the number of simultaneous proton-proton collisions, thus increasing the data rate. These upgrades will allow for high-precision measurements of rare particles such as the Higgs boson, as well as searches for new physics. The protons will collide in bunches every 25 ns (40 MHz), and we anticipate 200 simultaneous collisions per bunch crossing. Only a small fraction of these collisions are of interest for further physics analysis, hence a trigger system is used to keep the interesting events. The first step of the trigger system is the Level-1 (L1) hardware trigger [4], which can make decisions within a few microseconds but does not include all detector systems nor the full precision of the data. The HL-LHC demands an upgrade of the trigger system to keep the trigger rate below the allowed limit of 750 kHz. CMS will, therefore, for the first time include data from the new silicon Outer Tracker [5] in the L1 trigger. The tracks produced from the Outer Tracker data will mainly be used for L1 algorithms that perform vertex finding and particle identification, such as particle flow [6]. The HL-LHC trigger rate limit has been increased compared to the current LHC Run-3 rate of 100 kHz. However, without tracking information, L1 rates will quickly rise to exceed the available bandwidth with an unacceptable loss in physics performance [4].

The L1 track finder will reconstruct all charged particle trajectories (tracks) with a transverse momentum (p_T) greater than 2 GeV using Outer Tracker hit-pairs called stubs. Approximately 10 000 stubs will be produced for every event, yielding around 200 tracks that need to be reconstructed within 4 µs before being sent to the L1 trigger for the event selection [7]. The track-finding algorithm consists of a road-searching algorithm that connects stubs to create track candidates [8], and a Kalman Filter to do the final track parameter fit [9]. Programmable integrated circuits, i.e. FPGAs, were chosen to run the track finding due to their speed and commercial availability. The L1 track finder has been made possible thanks to the new CMS Outer Tracker sensor module design, which can filter data depending on the p_T, as well as the recent improvements in FPGA technology.

2. Outer Tracker

The CMS Outer Tracker is a completely new tracker system planned for the HL-LHC upgrade. Like its predecessor, it has a barrel consisting of multiple cylindrical layers, closely surrounding the LHC beam and the interaction point where the collisions occur. At each end of the barrel, endcaps consisting of disk-shaped layers enclose the Outer Tracker volume. A layout of the HL-LHC tracker can be seen in Figure 1.

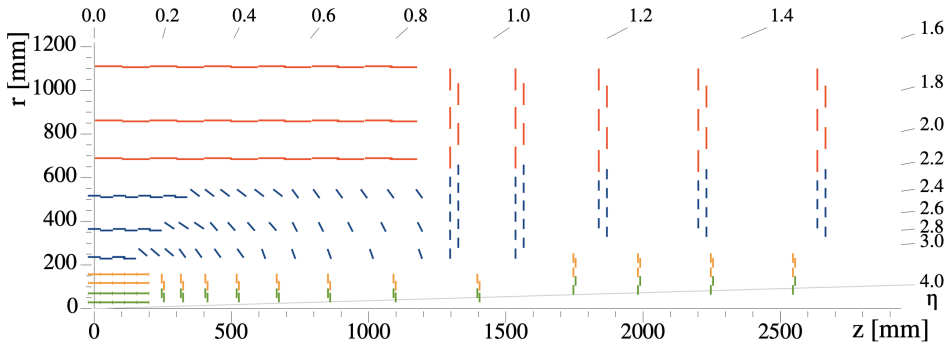


Fig. 1. An illustration of the tracker layout for the HL-LHC [5]. It shows one-quarter of the tracker in the $r-z$ plane, with the beam along the z-axis and the interaction point at the origin. The Outer Tracker PS and 2S sensor modules are depicted in blue and red, respectively. The Inner Tracker is depicted in green and orange.

The Outer Tracker will consist of two different types of silicon sensor modules: the PS module featuring a macro-pixel and a strip sensor, and the 2S module featuring two strip sensors, forming 6 barrel layers and 5 double-disks per side. Each module consists of two closely separated sensors, along with front-end ASICs that correlate the signals from the sensors to separate the low-momentum particles from the interesting high-momentum particles, as shown in Figure 2. As a result of the magnetic field in CMS, particles follow a helical trajectory with a bend radius proportional to their momentum. Lower momentum particles will therefore tend to traverse the tracker modules at a higher angle to the perpendicular, producing hits in the two sensor layers that are further apart. Using the front-end electronics it is possible to choose a window in the upper sensor (further away from the beam) such that only pairs of hits corresponding to a charged particle with a transverse momentum greater than 2 GeV, referred to as a stub, get transmitted to the back-end for further processing. The stubs are a simplified version of the raw data due to bandwidth limitations when transmitted at 40 MHz. Nevertheless, this will make the processing faster by the track finder. While the stubs are being processed, the full events, including the low momentum hits, are stored on buffers waiting for an event selection.

The PS modules are located closer to the beam (Figure 1). The strips are positioned such that the resolution is high in the φ-direction (i.e. the direction that the particles bend), and the macro-pixels enable the module to also have a good spatial resolution in z (r) for the barrel (endcap). In the 2S modules, both sensor layers consist of strips as it is not necessary to keep the high z-granularity further out from the collisions where the occupancy is lower.

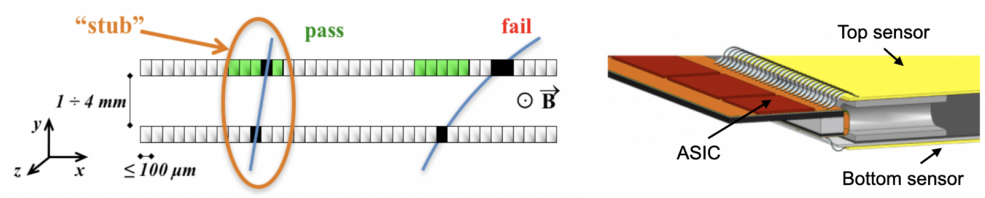


Fig. 2. A sketch of the stub-finding concept (left), and a sensor module with two closely separated sensors (right) [5]. A charged particle traverses both sensor planes. The particle will bend more if it has low momentum, thus a stub window in the top sensor can be chosen such that only particles compatible with $p_{\mathrm{T}} > 2$ GeV get transmitted by the on-detector electronics.

3. Track Finder Hardware

The track finder consists of FPGAs mounted on custom boards called Apollo [10]. Two Xilinx Virtex Ultrascale+ VU13P FPGAs are installed on each board and run a track-finding algorithm to form tracks from the incoming stubs. The track-finding algorithm firmware is described below. The boards also contain 25 Gb/s fibre optic links that receive stubs and transmit tracks to the L1 trigger boards. To make the track reconstruction fast, the track finder is split into 9 equally sized regions in φ that operate in parallel, shown in Figure 3. There is no communication between the sectors, so to avoid any missing tracks at the borders, the stubs in the overlap region are copied to the two adjacent track-finding sectors. The overlap region is chosen such that the edges correspond to tracks with $p_{\mathrm{T}} = 2$ GeV, to minimise the loss of tracks as well as the duplication of data.

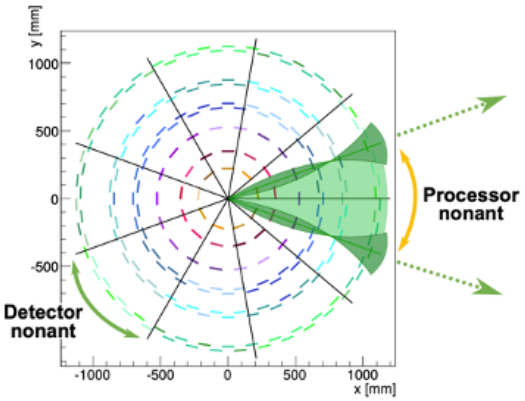


Fig. 3. A track-finding sector (processor nonant) illustrated in green, in an $x-y$ view of the Outer Tracker [4]. The dark green areas represent the regions where stubs are copied to both adjacent track-finding sectors. The detector sectors (nonants) are also illustrated. The track-finding sectors are shifted by 20° relative to the detector nonant edge, hence only two detector nonants need to be involved for each track-finding sector.

Each of the sectors is operated by 18 boards, each running the track-finding algorithm in parallel but for different events. In other words, the system is time-multiplexed by a factor of 18 and there will be 162 live boards. With one spare per sector, there will be 180 boards in total. A board receives a new event every 18th bunch crossing, i.e. every time-multiplex period, and operates with a fixed latency of 4 µs. The FPGA clock determines how fast data can be processed, hence the clock is set to 240 MHz which is sufficient to process input stubs with negligible truncation. The modules start processing one stub or track candidate every clock cycle and the whole process is pipelined.

4. The Track-finding Algorithm

The track-finding algorithm is split into multiple processing steps that are connected with memories that temporarily store the output until the next processing step reads it. A flowchart of the processing modules and the intermediate memories can be seen in Figure 4. Each module is implemented independently and processes one event at a time, thus while the first step is processing the most recent event, the second step is processing the previous event and so forth. This means that multiple events are processed on the same board at the same time. To make the algorithm chain even more efficient there are multiple copies of each module that are processing different stubs in parallel but within the same event. The algorithm concept, also shown in Figure 5, and an explanation of what each processing module does is given below.

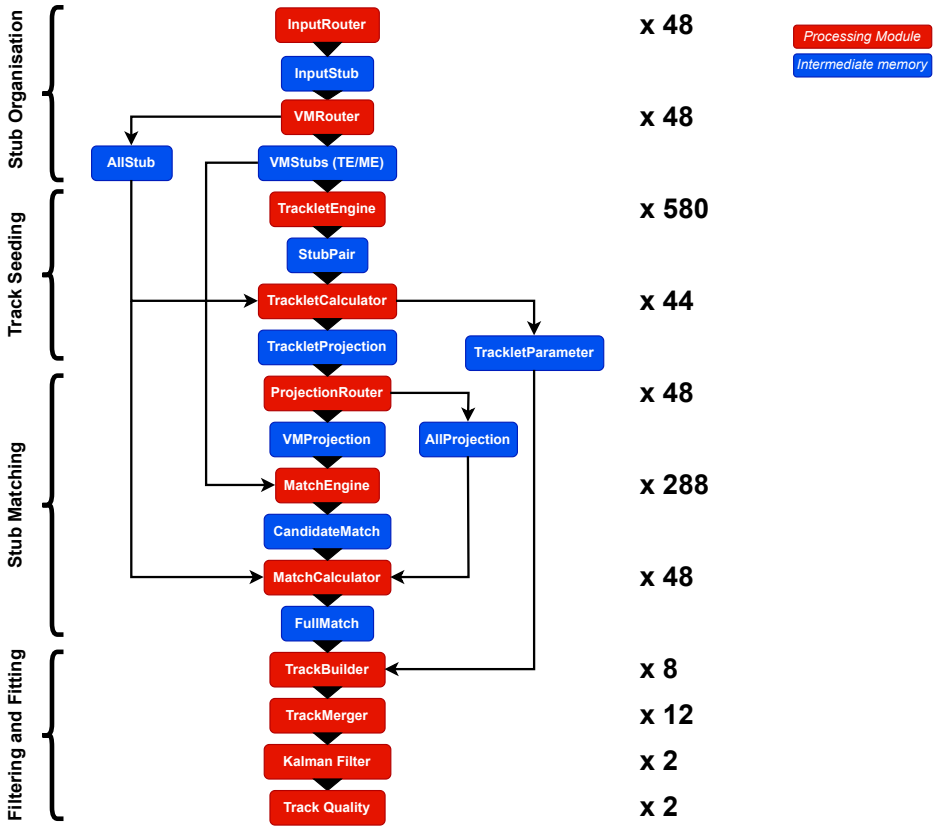


Fig. 4. A flowchart of the track-finding algorithm with processing modules in red and intermediate memories in blue. A brief overview of what each module does is given in Sections 4.1-4.4. The number of copies of each processing module is also given.

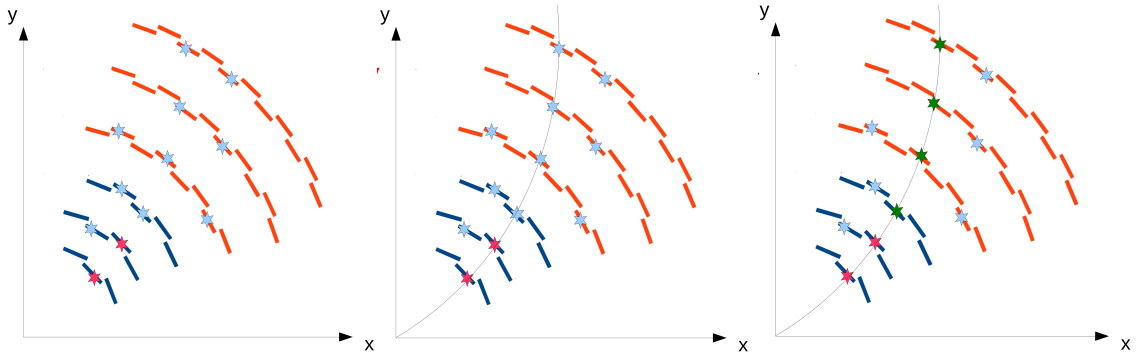


Fig. 5. An illustration of the track-finding algorithm concept [8]. First, stubs are paired up to create seeds (left). Using the seed and the origin, a rough helical track is calculated and projected to the other layers (middle). Then, stubs that are close enough to the track estimation in the other layers are matched to the track (right).

The majority of the processing modules are implemented in firmware using High-Level Synthesis (HLS) [11]. It takes code similar to C++ and compiles it into firmware. A few modules, such as the Kalman Filter and the Track Quality module, as well as the intermediate memories, were implemented using the hardware description language VHDL. This grants more control over resource usage. The connections of each module and memory are made by using a VHDL top-level script. However, due to the great number of modules and memories involved, a script instantiates and compiles all module copies. A C++ software emulation writes out a map of all module and memory connections, and a Python script uses that map to write the top-level VHDL. This Python script also allows for top-level functions that only contain a fraction of the full track-finding chain for small-scale testing.

4.1 Stub Organisation

The first two steps organise the stubs into bins to reduce the processing time and number of combinations in future steps. The first module, the Input Router, sorts the stubs depending on which layer they come from. The data is streamed directly into the Input Router and the output is then saved to intermediate memories that are being read one-by-one by the VMRouter. The VMRouter organises the stubs into so-called Virtual Modules (VMs) that in each layer represent small regions in φ. The track-finding algorithm can then minimise the number of combinations simply by trying to create tracks using VMs that are compatible with a track $p_{\mathrm{T}} > 2$ GeV. Each VM corresponds to one intermediate memory in the upcoming processing.

4.2 Track Seeding

In the following step, the Tracklet Engine, the stubs in each layer are paired with stubs in the neighbouring layer to form track seeds. Only a few VMs need to be checked to find compatible matches. All stub pairs are then saved in a memory that is read by the Tracklet Calculator. It is worth noting that the same stub can be used in multiple pairs and that the seeding is performed in multiple layers. This creates redundancy to increase robustness against inefficient sensor modules. In the Tracklet Calculator, the seeds along with the interaction point are used to form track estimations, called tracklets. Thanks to the uniform magnetic field in CMS, the particles will travel according to a helix, thus the track parameters are evaluated by simple calculations. The tracklets are then projected to the other layers.

4.3 Stub Matching

The projections to the other layers from the previous step are again organised into VMs in the Projection Router. This simplifies the operation of the next module, the Match Engine, when it tries to find stubs in the other layers that are close to the projection. Both the projection and all the matched stubs are saved into the intermediate memory. The distance between the matched stubs and the tracklet projection is calculated in the Match Calculator. The calculated residuals are then used to determine if the matched stub should be kept or not.

4.4 Filtering and Fitting

Once the matching is done, the Track Builder reads from the final intermediate memory and assembles all matched stubs and tracklet parameters into unified track objects. The Track Builder also removes any tracks with stubs in fewer than 4 layers. The selected tracks are then streamed directly to the next module. As previously mentioned, there is some redundancy due to the seeding in multiple layers, thus the same track can be found more than once. Therefore, there is also a duplicate remover called Track Merger that merges the inputs that correspond to the same track. The remaining merged tracks are streamed to the Kalman Filter for a final and more precise calculation of the track parameters using all the matched stubs found in the previous steps. Lastly, the Track Quality module determines and adds a parameter to the track that describes the quality of the Kalman Filter output. This extra parameter can be used in the L1 trigger to cut on tracks that generally have a better track reconstruction quality, e.g. tracks from muons.

5. Testing

To confirm the correct operation of each module, input and output data created by the software emulation of the L1 tracking algorithm are used. The emulation contains each of the processing modules along with the intermediate memories, and the content of the memories is saved. Using the emulation input to the firmware simulations, comparisons between the simulation output with the emulation output can be performed. A complete agreement is expected as the emulation was written such that it uses the same number of bits as the firmware for each variable. The test data are made using up to 1000 top-quark pair-production events with 200 simultaneous collisions. When testing each processing module individually, the input to the modules is read from the emulation and saved into the intermediate memories upstream relative to the module that is being tested. The tested module then reads from those memories, processes the data, and saves it to the memories downstream. The data saved in the output memories are later read and compared to the emulation output. If a chain of modules is being tested, the firmware also saves the content of each intermediate memory for easier debugging.

All processing modules have been implemented in firmware and pass firmware simulations when tested individually. The modules also meet the timing requirements and can start processing new stub data every clock cycle using a 240 MHz FPGA clock.

5.1 Skinny Chain

The first attempt at synthesising a long chain, containing modules from the Input Router to the Track Builder, was the so-called skinny chain. This is a reduced version of the full track-finding chain. The skinny chain is centred around a single Tracklet Calculator, and only the modules and memories that were connected to it upstream and downstream were included. Approximately 4% of the modules of the full project are involved. The chain was first tested in firmware simulations. The input stubs were again taken from the software emulation and streamed into the Input Routers. The output from the Track Builder was saved and later compared to the emulation. After passing firmware simulations and meeting the timing constraints the chain was later tested on an FPGA (Xilinx Virtex Ultrascale+

VU7P).

5.2 Barrel-Only Chain

A chain containing the majority of modules in the Outer Tracker barrel layers has also been tested. The Track Merger, Kalman Filter, and Track Quality modules were excluded. The barrel-only project consists of approximately two-thirds of the full project. When synthesising the chain, it was set to be run on a single Xilinx Virtex Ultrascale+ VU13P, but due to congestion on the FPGA, the project does not yet meet timing at 240 MHz. When a large fraction of the resources on an FPGA is being used, the routing between the components can become so long that it can not keep up with the clock. The final track-finding chain is planned to run on two VU13P FPGAs and thus we have a lot of headroom. With better FPGA floor-planning of the resources, good progress has been made towards meeting the timing requirements. The resource usage is also being optimised and the preliminary resource usage numbers can be seen in Table I.

Table I. Preliminary resource usage for the barrel-only chain along with the available resources on a Xilinx Virtex Ultrascale+ VU13P. The fraction of the resource utilisation of the barrel-only chain on the VU13P is also shown. The FPGA consists of five different types of resources: lookup tables (LUTs), i.e. truth tables that can perform combinatorial logic; flip-flops (FFs) which store data between clock cycles; Digital Signal Processor (DSP) blocks that, for example, can efficiently perform multiplication; Block RAMs (BRAMs) and UltraRAMs (URAMs) that are used for storing large amounts of data on the FPGA.

	LUT	**FF**	**DSP48E**	**BRAM_18K**	**URAM**
Total	711720	1278914	1176	2732	224
Available (VU13P)	1728000	3456000	12288	5376	1280
Utilisation (%)	**41.2**	**37.0**	**9.6**	**50.8**	**17.5**

6. Combined Modules

There are many ways to optimise the track-finding algorithm, and the Combined Modules version of the algorithm is one of them. The track-finding chain is set up such that each module takes one time-multiplex period to process one event, thus reducing the number of processing steps will reduce the total latency. One way of decreasing the number of algorithm steps is to combine some of the processing modules into bigger but more advanced combined modules. The same algorithm concept is being implemented, but without the need for intermediate memories, thus the total resource usage will be reduced.

The Tracklet Engine and the Tracklet Calculator are combined into the Tracklet Processor, and the Projection Router, the Match Engine, and the Match Calculator are combined into the Match Processor. This turns five algorithm steps into two, in addition to having three fewer intermediate memory types. The Tracklet Processor and the Match Processor have been implemented in firmware, and the track-finding chain is being tested using these instead of the original modules.

7. Summary

At the HL-LHC, L1 track finding will be used by CMS to reduce the L1 trigger rate to a manageable level. The new tracking system will enable physics sensitivity at the same level as for the current detector, or even better, at higher luminosities. This will be the first time at the LHC that tracker data will be reconstructed for use in the L1 trigger, and this is possible thanks to the on-module data filtering based on p_{T} in the new silicon Outer Tracker. All charged particles with a

$p_T > 2$ GeV will be reconstructed within 4 µs. The L1 tracks are reconstructed by an algorithm run on FPGAs mounted on custom track-finder boards. The track-finder is split into nine equally sized regions, and each region has their boards and no communication between the regions is needed. The boards within the same region run the track-finding in parallel but for different events, and will receive new events every time-multiplex period of 18 bunch crossings. The track-finder algorithm is split into multiple pipelined steps, called processing modules. All processing modules have been implemented in firmware and successfully tested individually; they meet the timing requirements and pass simulations using known input and output data of events containing 200 simultaneous proton-proton collisions. Scripts have been created to instantiate and compile all processing modules, along with scripts that make the top-level function that connects all the modules to construct track-finding chains. The skinny chain, consisting of 4% of the full project including a majority of the different algorithm steps, has been successfully run on hardware. A barrel-only chain has also been created. The chain contains the majority of modules in the tracker barrel layers, corresponding to around two-thirds of the final project. Once the barrel-only chain has been tested on hardware, the tests will scale up to the full project.

References

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

[2] O. Brünig, et al., LHC Design Report, CERN-2004-003, (2004)

[3] G. Apollinari, et al., High Luminosity Large Hadron Collider (HL-LHC), CERN Yellow Rep., **4** (2017)

[4] CMS Collaboration, The Phase-2 Upgrade of the CMS Level-1 Trigger, CERN-LHCC-2020-004, CMS-TDR-021 (2020)

[5] CMS Collaboration, The Phase-2 Upgrade of the CMS Tracker, CERN-LHCC-2017-099, CMS-TDR-014 (2017)

[6] CMS Collaboration, Particle-flow reconstruction and global event description with the CMS detector, JINST, **12**, P10003 (2017)

[7] E. Bartz, et al., FPGA-based Real-time Charged Particle Trajectory Reconstruction at the Large Hadron Collider, CMS CR-2017/100 (2017)

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

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

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

[11] Xilinx, Vivado Design Suite User Guide: High-Level Synthesis, https://docs.xilinx.com/v/u/en-US/ug902-vivado-high-level-synthesis, UG902 v2020.1 (2021)