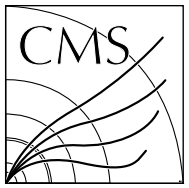
Available on CMS information server

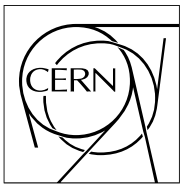
CMS NOTE -2017/009



The Compact Muon Solenoid Experiment

CMS Note

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



25 January 2017 (v9, 19 September 2017)

An FPGA-Based Track Finder for the L1 Trigger of the CMS Experiment at the High Luminosity LHC

R. Aggleton, L. Ardila-Perez, F. A. Ball, M. N. Balzer, G. Boudoul, J. Brooke, M. Caselle, L. Calligaris, D. Cieri, E. J. Clement, S. Dutta, G. Hall, K. Harder, P. R. Hobson, G. M. Iles, T. James, K. Manolopoulos, T. Matsushita, A. D. Morton, D. Newbold, S. Paramesvaran, M. Pesaresi, N. Pozzobon, I. D. Reid, A. W. Rose, O. Sander, C. Shepherd-Themistocleous, A. Shtipliyski, T. Schuh, L. Skinnari, S. P. Summers, A. Tapper, I. Tomalin, A. Thea, K. Uchida, P. Vichoudis, S. Viret, and M. Weber

Abstract

A new tracking detector is under development for use by the CMS experiment at the High-Luminosity LHC (HL-LHC). A crucial requirement of this upgrade is to provide the ability to reconstruct all charged particle tracks with transverse momentum above 2–3 GeV within 4μs so they can be used in the Level-1 trigger decision. A concept for an FPGA-based track finder using a fully time-multiplexed architecture is presented, where track candidates are reconstructed using a projective binning algorithm based on the Hough Transform, followed by a combinatorial Kalman Filter. A hardware demonstrator using MP7 processing boards has been assembled to prove the entire system functionality, from the output of the tracker readout boards to the reconstruction of tracks with fitted helix parameters. It successfully operates on one eighth of the tracker solid angle acceptance at a time, processing events taken at 40 MHz, each with up to 200 superimposed proton-proton interactions, whilst satisfying the latency requirement. The demonstrated track-reconstruction system, the chosen architecture, the achievements to date and future options for such a system will be discussed.

Contents

1 The High Luminosity Large Hadron Collider

To fully exploit the scientific potential of the Large Hadron Collider (LHC) [1], it is planned to operate the machine at a luminosity up to one order of magnitude higher than obtained with the nominal design. Installation of the High-Luminosity LHC (HL-LHC) upgrade [2] is expected to occur during a 30 month shut-down starting around 2024, leading to a peak luminosity of 5–7.5 $\times 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, corresponding to an average number of 140–200 proton-proton interactions, named pileup (PU), per 40 MHz bunch crossing. Targeting a total integrated luminosity of 3000 fb^{-1}, the HL-LHC will enable precision Higgs measurements, searches for rare processes that may deviate from standard model predictions, and increases in the discovery reach for new particles with low cross-sections and/or multi-TeV masses.

2 The CMS Tracker Upgrade

The Compact Muon Solenoid detector (CMS) is a large, general purpose particle detector at the LHC, designed to investigate a wide range of physics phenomena. The apparatus' central feature is a superconducting solenoid of 6 m internal diameter, providing a magnetic field of 3.8 T. Within the solenoid volume, a small silicon pixel Inner Tracker and larger silicon strip Outer Tracker are surrounded by an electromagnetic and hadronic calorimeter. Forward calorimeters extend the angular coverage. Gas-ionization detectors embedded in the magnet's return yoke are used to detect muons. A more detailed description of the CMS detector, together with a definition of the coordinate system used and the relevant kinematic variables, can be found in [3].

During the shut-down preceding the start of HL-LHC operation, the CMS tracker will need to be completely replaced. This is primarily due to the expected radiation damage of the silicon sensors following approximately 15 years of operation. The HL-LHC environment will provide a significant challenge for the new tracker [4]. The new tracking detector must maintain a high track reconstruction efficiency and a low misidentification rate under increased pileup conditions, requiring an increase in sensor channel granularity. The radiation hardness of the tracker must also be improved in order to withstand the increased fluence.

For the first time, the detector will be designed to allow the provision of limited tracking information to the Level-1 (L1) trigger system. The L1 trigger, based on custom electronics, is required to reject events that are deemed uninteresting for later analysis, and it is expected that data from the Outer Tracker could be used as an additional handle to keep the L1 acceptance rate below the 750 kHz maximum, while maintaining sensitivity to interesting physics processes.

Given the bandwidth implications in transferring every hit off-detector to the L1 trigger at the LHC bunch crossing rate of 40 MHz, a novel module design is being incorporated into the Outer Tracker upgrade. The proposed "p_T module" [5, 6] will comprise two sensors, separated by a few millimetres along the track direction, to discriminate on charged particle transverse momentum (p_T) based on the local bend of the track within the magnetic field (B), as shown in Fig. 1. Pairs of clusters consistent with a track of p_T greater than a configurable threshold (typically 2–3 GeV) are correlated on-detector, and the resulting *stubs* are forwarded to off-detector processing electronics, providing an effective data rate reduction of approximately a factor of ten [7, 8].

Two p_T-modules are in development for the Outer Tracker upgrade: 2S *strip-strip* modules and PS *pixel-strip* modules, both shown in Fig. 2. The 2S modules, each with an active area of

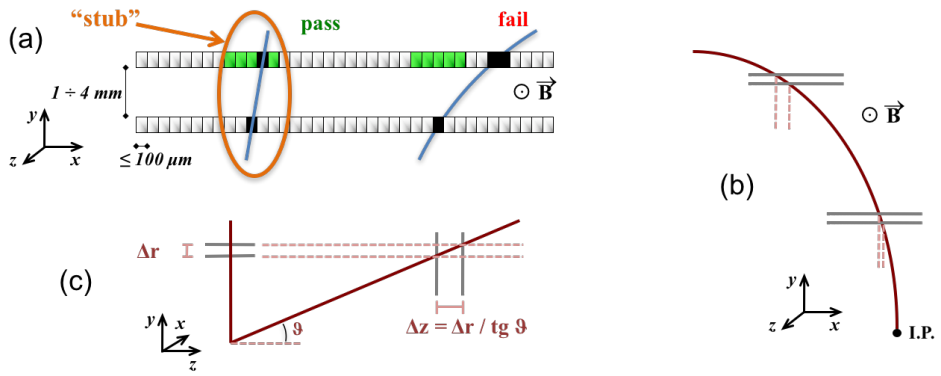


Figure 1: Cluster matching in p_T-modules to form stubs [9]. (a) Correlating pairs of closely-spaced clusters between two sensor layers, separated by a few mm, allows discrimination of transverse momentum. This is based on the particle bend in the CMS magnetic field and assumes that the particle originates at the beam line. (The z-axis of the coordinate system is parallel both to the field and to the beam-line). Only stubs compatible with tracks with $p_T > 2$–3 GeV are transferred off-detector. (b) The separation between the two clusters increases with the module's distance from the beam line, if the sensor spacing remains unchanged. (c) To achieve comparable discrimination in the endcap disks, which are orientated perpendicular to the beam line, a larger sensor spacing is needed, because of projective effects.

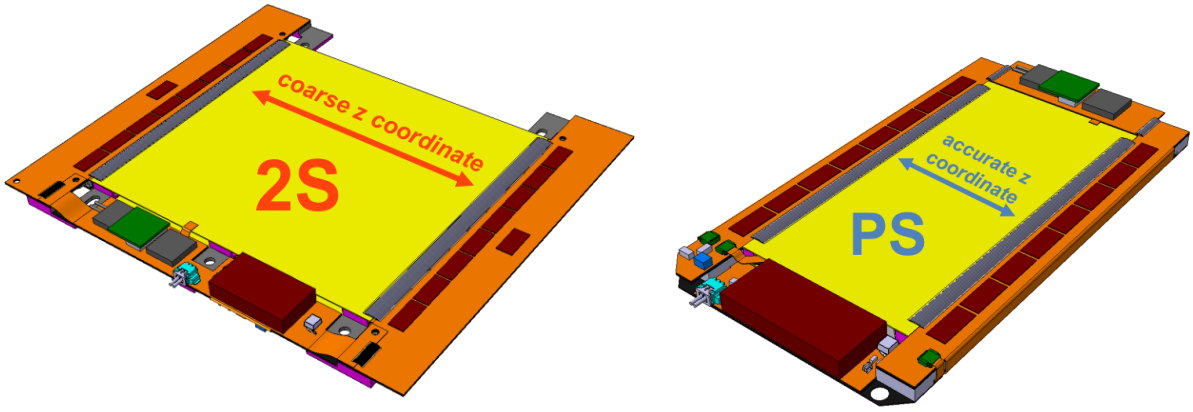


Figure 2: The 2S module (left) and PS module (right), described in the text [9].

10.05 cm $\times$ 9.14 cm, are designed to be used at radii $r > 60$ cm from the beam axis, where the hit occupancies are lower. Both upper and lower sensors in the 2S modules have a pitch of 90 μm in the transverse plane, r-φ, and a strip length of 5.03 cm along the direction of the beam axis, z. The PS modules, each with an active area of 4.69 cm $\times$ 9.60 cm, will be used at radii $20 < r < 60$ cm where the occupancies are highest. The PS modules consist of an upper silicon strip sensor and a lower silicon pixel sensor, both with a pitch of 100 μm in r-φ, and a strip length in z of 2.35 cm for the strips and 1.47 mm for the pixels. The finer granularity afforded by the pixel sensors provides more accurate pointing resolution along the z axis, which is crucial for identifying interaction vertices under high pileup conditions.

To perform stub correlation in the 2S modules, the signals of the top and bottom sensor are routed to the same CMS binary chip (CBC), which performs the correlation logic. This is possible by folding the readout hybrids around a stiffener. In the PS modules, strip signals are processed by the strip-sensor ASIC (SSA), and macro-pixel signals by the macro-pixel ASIC

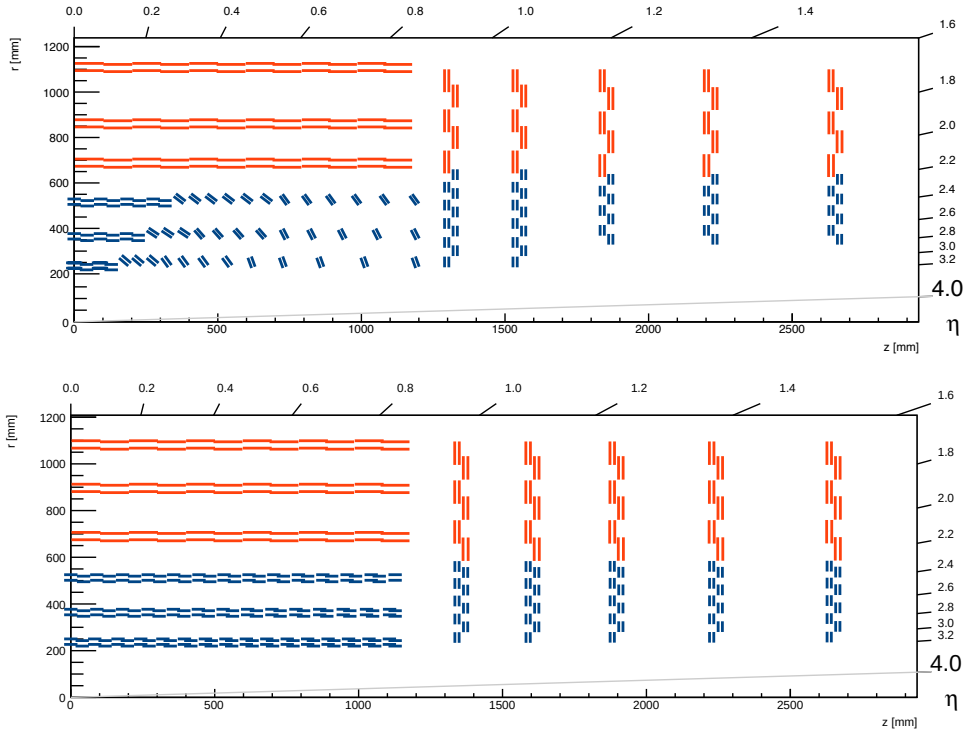


Figure 3: One quadrant of the upgraded Outer Tracker layout, showing the 2S (red) and PS (blue) module placement. The upper diagram shows the currently proposed layout, known as the *tilted barrel* geometry [9, 10]. The tilt of the modules in the three PS barrel layers improves overall performance and reduces construction costs. The lower diagram shows an older proposal for the layout, known as the *flat barrel* geometry [4], which was adopted for all the studies presented in this paper, except where stated otherwise.

(MPA). The strip data is routed from the SSA to the MPA via a folded hybrid, which then performs the cluster correlation. A detailed description of the front-end electronics can be found in [9].

The upper diagram in Fig. 3 depicts the currently proposed layout of the upgraded Outer Tracker, known as the *tilted barrel* geometry [9, 10], indicating the 2S and PS module positions. It includes six barrel layers, and five endcap disks on each side. Only modules located at $|\eta| < 2.4$ will be configured to send stub data off-detector. This geometry derives its name from the fact that some modules in the three innermost barrel layers are tilted, such that their normals point towards the interaction region. This improves stub-finding efficiency for tracks with large incident angles and reduces the overall cost of the system [9].

During the time in which the demonstrator described in this paper was constructed an older proposal for the upgraded Outer Tracker layout was in use within CMS. This design, known as the *flat barrel* geometry [4], is shown in the lower diagram in Fig. 3. It was adopted for all the studies presented in this paper, except where stated otherwise. As will be shown in Section 7, there is reason to believe that performance would improve when adapting to the tilted barrel geometry.

3 Track Finding at the Level-1 Trigger

The provision and use of tracking information at L1 in the form of fully reconstructed tracks with $p_{\mathrm{T}} > 3\,\mathrm{GeV}$ is a necessity if trigger performance is to be maintained or even improved upon relative to low luminosity running. It is estimated that under a high pileup scenario (200 PU), with trigger thresholds chosen to give good physics performance, the L1 rate could be reduced from 4 MHz to below 750 kHz, by using tracks to enhance the discriminating power of the trigger [4]. Flexibility to reconstruct tracks down to an even lower p_{T} threshold of 2 GeV may be desirable, if trigger requirements demand it. However, a 3 GeV threshold was used to obtain the results presented in this paper, except where stated otherwise.

The total L1 latency is limited to at most 12.5 μs, of which it is estimated that the L1 trigger electronics will require about 3.5 μs to correlate tracks with data primitives from the calorimeter and muon systems, and to take a decision as to whether the event is of interest. Propagation of the L1 decision to trigger the front-end buffers then takes another 1 μs while a further 3 μs is required as a safety margin [4]. This means that if tracks are to be utilised by the trigger successfully, stubs must be extracted from the tracker front-end electronics, organised, and finally processed to reconstruct tracks within approximately 5 μs of the collision. Since approximately 1 μs of this will be required for generation, packaging and transmission of stubs from the tracker front-end electronics to the first layer of off-detector readout electronics, known as the Data, Trigger and Control (DTC) system, the processing latency target to reconstruct the tracks starting from data arriving at the DTC is set at 4 μs [9], as shown in Fig. 4.

Each p_{T}-module will be served by a pair of optical fibres, one upstream and one downstream, which interface directly to the DTC system. Depending on the module radius, these links will be capable of transferring data off-detector at either 5.12 or 10.24 Gb/s, providing an effective bandwidth of between 3.84 and 8.96 Gb/s accounting for error correction and protocol overheads [11]. Approximately 75% of this bandwidth will be dedicated to readout of stub data from bunch crossings every 25 ns. The stub data format itself is dependent on the p_{T}-module type, but will contain an 11-bit address corresponding to the centroid of the seed cluster in the stub (in half-strip units) and a 3-bit (PS) or 4-bit (2S) number known as the "bend", which corresponds to the distance in strips between the two clusters in the stub and is related to the local bend of the particle trajectory. For PS modules only, a 4-bit address describing the z position of the stub along the sensor is additionally provided. The remaining approximately 25% of the module readout bandwidth will be dedicated to transmission of the full event data including

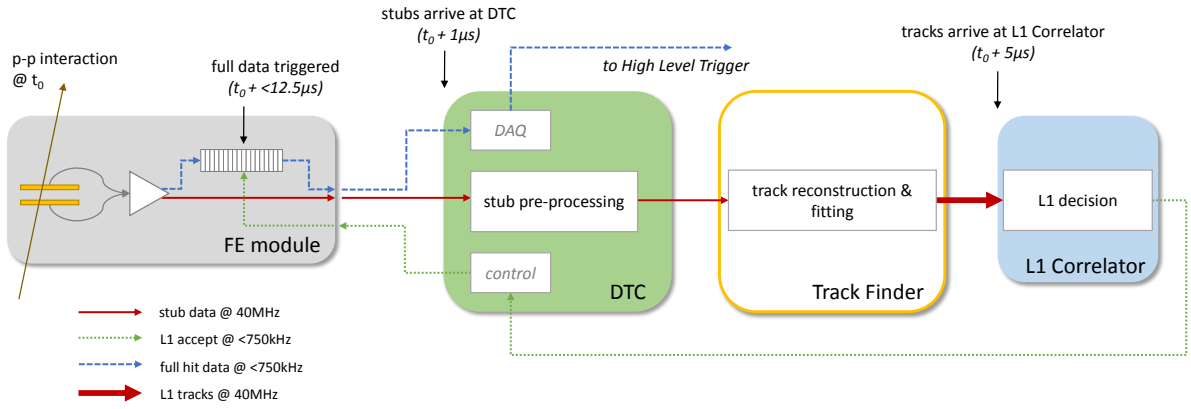


Figure 4: Illustration of data-flow and latency requirements from p_{T}-modules through to the off-detector electronics dedicated to forming the L1 trigger decision.

all hit strips/pixels, triggered by a L1-accept signal [9].

The DTC will be implemented as a custom-developed ATCA (Advanced Telecom Computer Architecture) blade based on commercial FPGAs and multi-channel opto-electronic transceivers. Each board can be expected to interface to several modules, depending on overall occupancy or constraints due to cabling of the tracker fibres, up to a proposed maximum of 72. The DTC will be required to:

- extract and pre-process the stub data before transmission to the Track Finder layer,
- extract and package the full event data sent from the front-end buffers before transmission to the data acquisition system (DAQ),
- provide timing and control signals to the modules for correct operation during data-taking, including configuration and calibration commands.

An aggregated data rate of 600 Gb/s per DTC will be provided to transmit stubs to the Track-Finder layer, corresponding to $\sim$36 links at 16.3 Gb/s based on current commercial FPGA technology. This is expected to be more than sufficient to handle fluctuations even in the highest occupancy DTCs serving the innermost layers at 200 PU, though the average rate of stubs per DTC will be much lower. A total complement of approximately 250 DTCs will be required to service the full Outer Tracker (256 DTCs for the flat barrel geometry described in Fig. 3).

4 An FPGA Based Track Finding Architecture

There is some flexibility when it comes to defining the track finding architecture, including the choice of how track finding is parallelised across processors. Constraints arise from how the detector is cabled to the DTC system, and on the number of high speed optical links available on the DTC and Track Finder boards. In terms of the cabling of the detector to the back-end system, it is assumed that the DTCs will be arranged such that a set of 32 blades will together process all data from an approximate octant (i.e. 45 degree φ-sector) of the Outer Tracker. These wedges, referred to here as *detector octants*, do not have uniform boundaries as an exact eight-fold symmetry does not occur in the tracker layout.

In this paper we propose the concept of a scalable, configurable and redundant system architecture based on a fully time-multiplexed design [12, 13]. In systems where processing bandwidth is limited, data from a single event but from multiple sources can be buffered and transmitted over a longer time period to a single processor node, while processing of subsequent events is carried out on parallel nodes in a similar fashion. This approach, known as time-multiplexing, requires at least two processing layers with a switching network between them. The switching network could, for example, be implemented as a dynamic traffic scheduler, as in the case of the CMS High Level Trigger (HLT) [14], or alternatively, like the static fixed-packet router used in the Level-1 Calorimeter Trigger [15]. Provided data are suitably formatted and ordered in the first layer, the majority of the processing or analysis, such as track finding, can take place in the second layer. For a fixed time-multiplexing factor of n, one would require n nodes in the second layer, where each time node processes a new event every $n \times 25$ ns, where 25 ns is the time interval between events at the LHC.

One advantage of using time-multiplexing in this way is the flexibility it affords to overcome the regional segmentation of the detector, so that in the case of the track finder, all stub data consistent with a track can be brought to the same card for processing. Another feature is the fact that only a limited amount of hardware is needed to demonstrate an entire system, since each node is carrying out identical processing on different events. By treating the DTC as the

first layer in a time-multiplexed system, it should be feasible to stream the full set of stubs for a large fraction of the detector into a time node, or Track Finding Processor (TFP). While in principle the system could be configured so that a TFP processes data from the entire tracker, in practice this is prevented by limits on the number of input links and total bandwidth a single FPGA-based processor could handle. In this paper we consider the division of the time-multiplexed Track Finder layer into octants, to match the number of regions in the DTC layer.

In order to handle duplication of data across hardware boundaries a simplification can be applied at the DTC-TFP interface. Defining *processing octant* boundaries that divide the tracker into eight 45 degree φ-sectors, each rotated by approximately 22.5 degrees in φ with respect to the detector octant boundaries, implies that a DTC handles data belonging to no more than two neighbouring processing octants (Fig. 5). As such, the first step of the DTC can be to unpack and convert the stubs from the front-end links to a global coordinate system. A globally formatted stub can be described adequately with 48-bits. This can be followed by an assignment of every stub to one of the two regions, or if it is consistent with both, by duplicating the stub into both processing octants. This duplication would occur whenever a stub could be consistent with a charged particle in either processing octant, from the knowledge that a track with $p_{\mathrm{T}} = 3\,\mathrm{GeV}$ defines a maximum possible track curvature. However, the measurement of the stub local bend as described in Section 3 can also be deployed to minimise the fraction of stubs duplicated to both octants. The exact logic is identical to that which will be described in Section 5.1, to assign stubs to sub-sectors.

A baseline system design is illustrated in Fig. 5. As described in Section 3, each DTC will dedicate ∼600 Gb/s of bandwidth for transmission of stub data to the Track Finder layer, corresponding to 36 links at 16.3 Gb/s. By applying the duplication technique described above, the DTC is expected to send 50% of its data to one processing octant, and 50% to its neighbour, on average, which can be provided by 18 links per DTC per processing octant. This allows each DTC to be capable of time-multiplexing its data to up to n=18 time nodes, where one can assign a single optical link to each node or TFP. A DTC therefore sends its data to 36 independent TFPs (18 time nodes $\times$ 2 processing octants).

Conversely, to ensure that a single TFP receives all data for one processing octant for one event it should be capable of receiving 64 links (one link from each DTC in two neighbouring detector

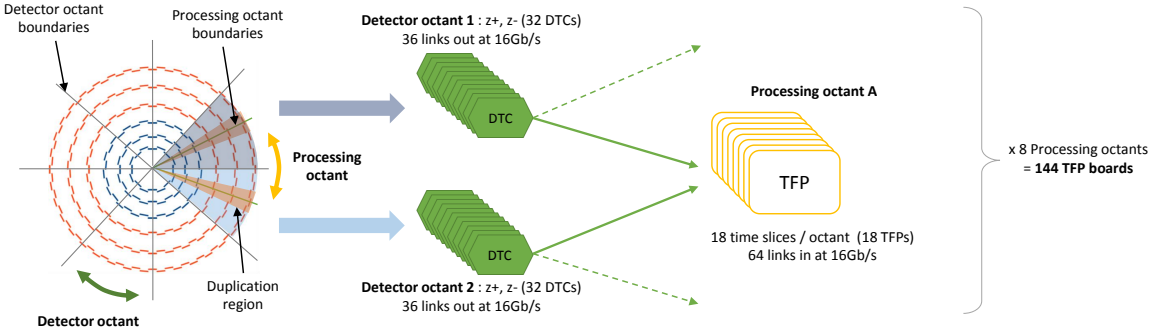


Figure 5: Baseline system architecture whereby DTCs in two neighbouring detector octants time-multiplex and duplicate stub data across processing octant boundaries before transmission to the Track Finding Processors (TFPs). With 18 time nodes and eight processing octants, the full track finding system would be composed of 144 TFPs.

octants), which is feasible using existing FPGA technology. With 18 time nodes and eight processing octants, the full track finding system would be composed of 144 TFPs, each processing a new event every $18 \times 25\,\text{ns} = 450\,\text{ns}$.

The proposed architecture is easily scalable, as one can adapt the system to different time-multiplexing factors (by adjusting the number of links from the output of the DTC), or different processing loads on the TFP (by adjusting the link speed and therefore total TFP input bandwidth). The regional segmentation of the track finding layer can be adapted to match the final regional segmentation of the DTC layer. Since each TFP operates independently and no data sharing between boards is necessary (as all relevant data is pre-duplicated at the DTC), this reduces requirements on synchronisation throughout the entire system. Since all TFPs perform identical processing on φ-symmetric regions of the detector, the same FPGA logic can be implemented on every board, simplifying operation of the running system.

One additional advantage of a time-multiplexed design is the possibility, providing a couple of spare output links on the DTC, to incorporate extra nodes into the system for redundancy or for parallel development of new algorithms. A redundant node can be quickly switched in by software if a hardware failure is discovered at one TFP, or in one of its input links, so that any downtime is minimised. Alternatively, data from $1/n$ events can be automatically duplicated into the spare node so that any changes to the algorithm can be verified on real data, without affecting performance or running of the system. In the baseline architecture only eight redundant nodes would be required, one per octant.

The Track Finding Processor logic is divided into four distinct components, described in Section 5 in further detail:

- **Geometric Processor (GP)** - responsible for processing of the stub data before entry into the subsequent stage, including subdividing of the octant into finer sub-sectors in η and φ to simplify the track finding task and to increase parallelisation;
- **Hough Transform (HT)** - a highly parallelised first stage track finder that identifies groups of stubs that are coarsely consistent with a track hypothesis in the r-φ plane, so reducing combinatorics in the downstream steps;
- **Kalman Filter (KF)** - a second stage candidate cleaning and precision fitting algorithm to remove fake tracks and improve helix parameter resolution;
- **Duplicate Removal (DR)** - final pass filter using the precise fit information to remove any duplicate tracks generated by the Hough Transform.

5 The Track Finding Processor

The Track Finding Processor (TFP) logic can be divided into a series of components whose operations will be described in the following section. An overview of the TFP logic is provided in Fig. 6 illustrating the main components and their interconnectivity.

5.1 Geometric Processor

Each GP pre-processes the 48-bit DTC stubs from one processing octant, both unpacking the data into a 64-bit extended format to reduce processing load on the HT, and assigning the stubs to geometric sub-sectors, which are angular divisions of the octant. The GP firmware consists of a pre-processing block, which calculates the correct sub-sector for each stub based on its global coordinate position, followed by a layered routing block. The stubs associated to each sub-sector are routed to dedicated outputs, such that data from each sub-sector can be processed

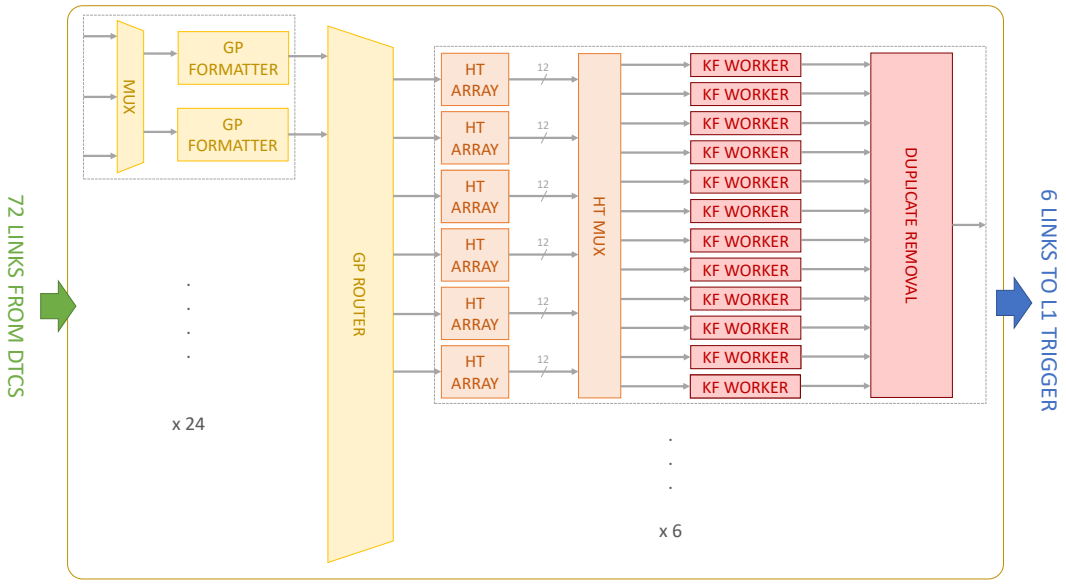


Figure 6: An overview of the TFP illustrating the main logic components and their interconnectivity, each described in detail in the following section. Yellow components are part of the Geometrical Processor, orange components are part of the Hough Transform, red components are part of the Kalman Filter or Duplicate Removal. The TFP is capable of processing data from up to 72 input links (one per DTC). This allows for some margin in the exact number of DTCs, which is yet to be determined.

by an independent HT array.

As depicted in Figure 7, the GP subdivides its processing octant into 36 sub-sectors, loosely referred to as (η, φ) sub-sectors, formed from two divisions in the r-φ plane and 18 divisions in the r-z plane. The division of the octant into sub-sectors simplifies the task of the downstream logic, so that track finding can be carried out independently and in parallel within each of the sub-sectors. The use of relatively narrow sub-sectors in η has the added advantage that it ensures that any track found by the HT stage must be approximately consistent with a straight line in the r-z plane, despite the fact that the HT itself only does track finding in the r-φ plane.

Each sub-sector is used by the TFP to find tracks in different ranges of ϕ_T and z_S, where ϕ_T (z_S) is defined as the φ (z) coordinate of a track trajectory relative to the point where it crosses a cylinder of radius T (S) centred on the beam line. The values of these two parameters are chosen to be $T = 58\,\text{cm}$ and $S = 50\,\text{cm}$, since this minimises the fraction of stubs that are consistent with more than one sub-sector. The ranges in ϕ_T or z_S covered by neighbouring sub-sectors are contiguous and do not overlap. In the r-φ plane, the sub-sectors are all equally sized, whereas in the r-z plane their size varies so as to keep the number of stubs approximately equal in each sub-sector. The GP must assign each stub to a sub-sector based on whether the stub could have been produced by a charged particle with a trajectory within the ϕ_T or z_S range of that sub-sector while originating from the beam line. If the stub is consistent with more than one sub-sector, then the GP duplicates it. This can occur because of the curvature of tracks within the magnetic field (constrained by the configurable track finding p_T threshold, chosen for the studies presented here to be $p_\text{T}^\text{min} = 3\,\text{GeV}$) or because of the length of the luminous region along the beam axis (where a configurable parameter w, chosen to be 15 cm, defines

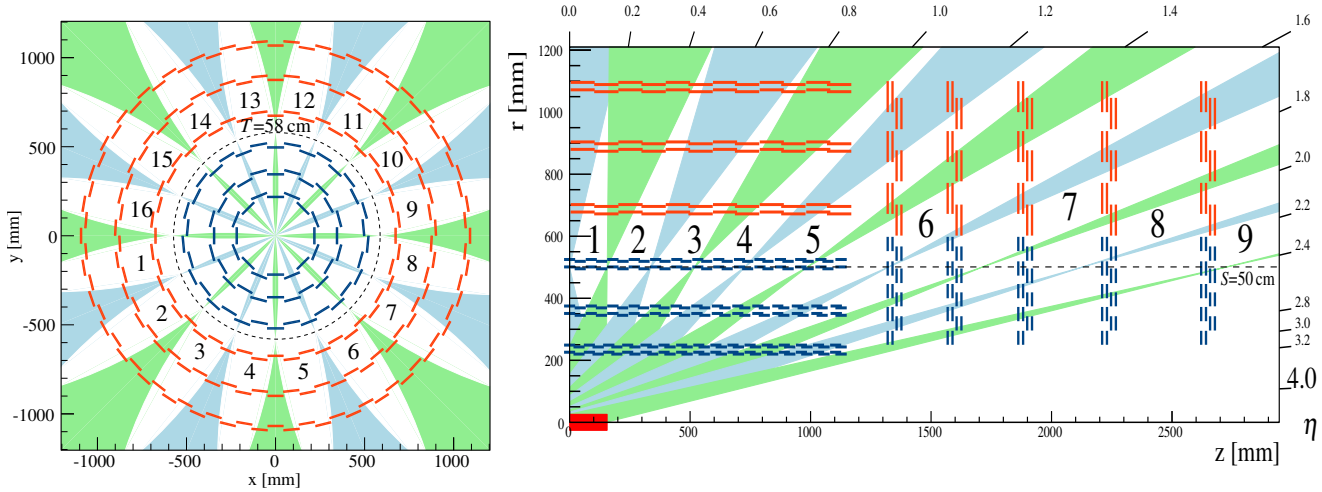


Figure 7: The segmentation of the tracker volume into φ sub-sectors (left) and η sub-sectors (right). The numbered areas in white represent the regions that are associated to only one sector, whereas the coloured areas (where there is no difference in meaning between green or blue) represent the overlap region between neighbouring sectors where stubs may need to be assigned to both sectors. The two cylinders mentioned in the text of radius $T = 58\,\text{cm}$ and $S = 50\,\text{cm}$, are indicated by dashes in the left and right-hand figures, respectively.

the half-width of the beam spot along z). Using the algorithm described below, each stub is assigned to an average of 1.8 sectors.

A stub with coordinates (r, φ, z) is compatible in the r-z plane with a sub-sector covering range $z_S^{\text{min}} < z_S < z_S^{\text{max}}$ if

$$\frac{r \cdot z_S^{\text{min}}}{S} - w \cdot \left|\frac{r}{S} - 1\right| \; < \; z \; < \; \frac{r \cdot z_S^{\text{max}}}{S} + w \cdot \left|\frac{r}{S} - 1\right| \; . \tag{1}$$

To further improve the sector granularity in the TFP without using significant extra FPGA resources, each of these η sub-sectors can be further divided by an additional factor of two in the r-z plane, with this division positioned at the mid-point between the sub-sector's boundaries $(z_S^{\text{min}}, z_S^{\text{max}})$. Whenever a stub is assigned to a sub-sector, the GP checks the consistency of the stub with each of these sub-sector halves, allowing for some overlap, and stores this information as two bits within the stub data, for subsequent use by the HT.

The corresponding equation for the compatibility in the r-φ plane of the stub with a sub-sector is

$$|\Delta\varphi| < 0.5 \cdot \frac{2\pi}{N_\varphi} + \varphi_{\text{res}}, \tag{2}$$

where $\Delta\varphi$ is the difference in azimuthal angle between the stub and the centre of the sub-sector and N_φ is the number of φ sub-sectors and is always a multiple of eight (due to the track-finder division into octants), currently set to 16. The azimuthal angle of the centre of sub-sector i is $\varphi_i = \frac{2\pi i}{N_\varphi}$, where $1 \leq i \leq N_\varphi$. The parameter φ_{res} accounts for the range of track curvature in φ allowed by the threshold $p_{\text{T}}^{\text{min}}$, and is equal to

$$\varphi_{\text{res}} = \frac{0.0015\, qB}{p_{\text{T}}^{\text{min}}} \cdot |r_T|, \tag{3}$$

where $r_T = r - T$, q is the particle charge in units of e, and the variables p_{T}, B, and r_T are measured in units of GeV, Tesla and cm respectively. With the chosen value of $N_\varphi = 16$, no individual stub can be compatible with more than two neighbouring φ sub-sectors, providing that $p_{\text{T}}^{\text{min}}$ is not reduced below 2 GeV.

However, the stub can also be tested against a second condition in the r-φ plane, to reduce the number of stubs that need to be duplicated. This test exploits the stub bend measurement b, measured in units of the strip pitch, which is provided by the p_{T}-modules. The bend further constrains the allowed q/p_{T} range of the track to lie in the range $(q/p_{\text{T}})_{\text{min}} < (q/p_{\text{T}}) < (q/p_{\text{T}})_{\text{max}}$ where:

$$(q/p_{\text{T}})_{\text{max/min}} = \frac{(b \pm k_b)\, \rho}{0.0015\, rB}\,, \tag{4}$$

$\rho = (p/s)$ for barrel stubs and $\rho = (p/s) \cdot (z/r)$ for endcap stubs, and p and s are the pitch and separation of the two sensors in a module, respectively. As there are only eight possible values of (p/s), this quantity is retrieved from a look-up table in firmware. This equation assumes that the resolution in the bend, when measured in units of the sensor pitch, is the same everywhere in the tracker. Simulations confirm this assumption to be valid and indicate an approximate value of $\sqrt{2/12}$ for the resolution. The true bend is assumed to lie within k_b of the measured value, where k_b is a configurable cut parameter whose value is chosen to be 1.25 (approximately three standard deviations).

This constraint on q/p_{T} leads to the condition:

$$|\Delta\varphi\prime| < 0.5 \cdot \frac{2\pi}{N_\varphi} + \varphi\prime_{\text{res}} \tag{5}$$

where

$$\Delta\varphi\prime = \Delta\varphi + b\rho\frac{r_T}{r} \tag{6}$$

and $\varphi\prime_{\text{res}}$, which allows for the resolution in the stub bend, is given by

$$\varphi\prime_{\text{res}} = k_b\rho\left|\frac{r_T}{r}\right|. \tag{7}$$

The GP routing block is implemented as a three-stage, highly pipelined mesh. It can route stubs from up to 72 inputs, one per DTC (with up to 36 DTCs assumed in each of the two detector octants from which the GP receives data), to any of 36 outputs, where each output corresponds to a sub-sector. The first layer organises stubs into six groups of three sub-sectors in η, which in turn are each arranged according to their final η sub-sector in the second layer. The third layer routes the stubs by φ sub-sector. Each arbitration block in this router is highly configurable, and can easily be adapted for alternative sub-sector boundaries.

The GP for a processing octant can be implemented within a single Xilinx Virtex-7 XC7VX690T FPGA. The FPGA resource usage is shown in Table 1. Running at 240 MHz, the latency (de-

Table 1: Resource utilisation of each pre-processing block (with 48 needed per TFP) and the entire routing block of the GP as implemented in the Xilinx Virtex-7 XC7VX690T FPGA. The usage as a percentage of the device's available resources are shown in parenthesis. Four types of FPGA resources are given: look-up tables (LUTs); digital signal processors (DSPs); flip-flops (FFs); and block RAM (BRAM), a dedicated two-port memory module containing 36 Kb of RAM each. A description of the FPGA and each type of logic resource can be found in [16].

	LUTs	**DSPs**	**FFs**	**BRAM (36 Kb)**
Pre-processing block	1942 (0.4%)	22 (0.6%)	2416 (0.3%)	1 (0.0%)
Routing block	27700 (6.4%)	0 (0.0%)	89531 (10.3%)	174 (11.8%)

fined as the time difference between first stub received and first stub transmitted) of the pre-processing and routing blocks is 58 and 193 ns, respectively. A version of the GP router has been developed to run at 480 MHz. In this version, additional registers were required to meet timing constraints, leading to an overall latency reduction of 60 ns.

5.2 Hough Transform

5.2.1 Algorithm Description

The Hough Transform is a widely used method of detecting geometric features in digital images [17]. As such, it is well suited to the task of recognising tracks from a set of stubs. Here, it is used to reconstruct primary charged particles with $p_{\mathrm{T}} > p_{\mathrm{T}}^{\mathrm{min}}$, using data from the Outer Tracker in the r-φ plane. An independent Hough Transform is used for track finding in each of the 36 sub-sectors defined by the GP within each processing octant.

Charged particles are bent in the transverse plane by the homogeneous magnetic field (B), and their radius of curvature R (in cm) expressed as a function of the particle's p_{T} and charge q is

$$R = \frac{p_{\mathrm{T}}}{0.003\, qB} \,. \tag{8}$$

Particles originating at or close to the interaction point are of most relevance to the L1 trigger. The trajectory of such particles in the transverse plane is described by the following equation.

$$\frac{r}{2\,R} = \sin\left(\varphi - \phi\right) \approx \varphi - \phi \,. \tag{9}$$

Here ϕ is the angle of the track in the transverse plane at the origin [6], and the small angle approximation used is valid for tracks with transverse momentum above about 2 GeV (large R). Furthermore, the (r,φ) coordinates of any stubs produced by the particle will be compatible with this trajectory, if one neglects effects such as multiple scattering and bremsstrahlung.

Combining the two previous equations, one obtains

$$\phi = \varphi - \frac{0.0015\, qB}{p_{\mathrm{T}}} \cdot r \,. \tag{10}$$

This equation shows that a single stub with coordinates (r,φ) maps onto a straight line in the track parameter space (q/p_{T}, ϕ), also known as Hough-space. If several stubs are produced by the same particle, then the lines corresponding to these stubs in Hough-space will all intersect at a single point, neglecting effects such as detector resolution and multiple scattering for the

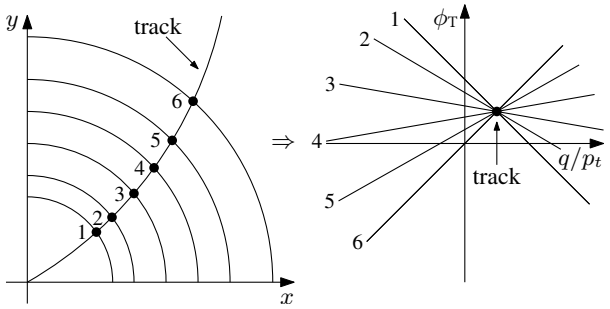


Figure 8: Illustration of the Hough Transform. On the left-hand side is a sketch of one quarter of the tracker barrel in the x-y plane, showing the trajectory of a single particle together with the stubs it produces, shown as dots, in the six barrel layers. On the right-hand side, the same six stubs are now shown in Hough-space, where the axes correspond to track parameters (q/p_{T}, ϕ_T). Each stub is represented by a straight line, and the point where several such lines intersect both identifies a track and determines its parameters (q/p_{T}, ϕ_T).

time being. This intersection of stub-lines can be used to identify track candidates. Furthermore, the coordinates of the intersection point provide a measurement of the track parameters (q/p_{T}, ϕ). This is illustrated in Fig. 8.

In this Hough-space, the gradient of each stub-line is proportional to the radius r of the stub, so is always positive. It is preferable to instead measure the radius of the stub using the variable r_T, defined in Section 5.1. This transforms the previous equation into

$$\phi_T = \varphi - \frac{0.0015\, qB}{p_{\mathrm{T}}} \cdot r_T \,, \tag{11}$$

where the track parameters are now (q/p_{T}, ϕ_T), with ϕ_T defined in Section 5.1. In this new Hough-space, the stub-line gradient is proportional to r_T, so can be either positive or negative, as was assumed when drawing Fig. 8. The larger range of stub-line gradients improves the precision with which the intersection point can be measured, resulting in fewer misreconstructed or duplicate tracks.

To implement the HT algorithm, the Hough-space can be subdivided into an array of cells, bounded along the horizontal axis by $|q/p_{\mathrm{T}}| < q/p_{\mathrm{T}}^{\mathrm{min}}$, where $p_{\mathrm{T}}^{\mathrm{min}} = 3\,\mathrm{GeV}$ is used for the results presented in this paper, and along the vertical axis by the range in ϕ_T covered by the individual sub-sector. An array granularity of 32×64 cells in $q/p_{\mathrm{T}} \times \phi_T$ is chosen as a compromise between tracking performance and FPGA resource use, although the cell size could not be reduced significantly further without making the HT sensitive to deviations from Eq. 11 caused by multiple scattering or detector effects. Stubs are added to any cell that their stub-line passes through.

Each stub also contains the bend information, which can be used to estimate an allowed range in q/p_{T} of the particle that produced the stub, as given by Eq. 4. Each stub need therefore only be added to those cells in the HT array, whose q/p_{T} column is compatible with this allowed range. This substantially reduces the probability of producing combinatorial fake candidates. The compatible q/p_{T} column range is precalculated by the GP.

A track candidate is identified if stubs from a minimum number of tracker barrel layers or endcap disks accumulate in an HT cell. Primary charged particles with $p_{\mathrm{T}} > 2\,\mathrm{GeV}$ and $|\eta| < 2.4$ are usually expected to traverse at least six of these stations. However, to allow for detector

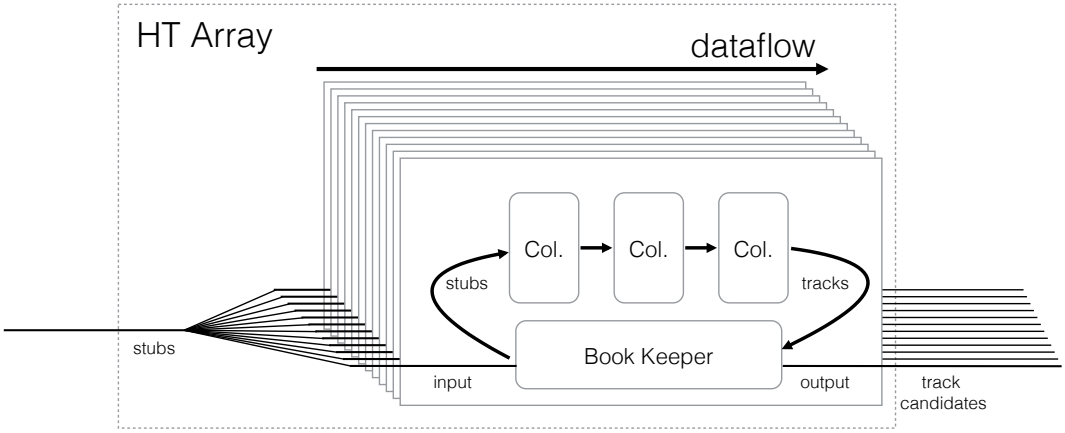


Figure 9: Firmware implementation of one HT array, as used within an individual sub-sector. In each of the twelve pages visible in the figure, a 'Book Keeper' is connected to a daisy chain of two to three 'Columns (Col.)' (8 Book Keepers $\times$ 3 Columns and 4 Book Keepers $\times$ 2 Columns $=$ 32 Columns). Internal components are shown as boxes and data paths as lines, where arrows indicate the direction of data flow.

or readout inefficiencies, and for geometric coverage, the threshold criteria used to identify a track candidate only demands stubs in at least five different tracker barrel layers or endcap disks, and this requirement is reduced to four in the region $0.89 < |\eta| < 1.16$ to accommodate a small gap in acceptance between the barrel and the endcaps.

5.2.2 Implementation

The HT track-finder has been implemented in FPGA firmware where each TFP employs 36 HT arrays running in parallel. Each individual HT array processes data from one input channel, corresponding to the stubs consistent with a single geometric sub-sector, as defined by the GP.

The design of each HT array can be split into two fully pipelined stages: the filling of the array with stubs; and the readout of the track candidates it finds. Each stage processes one stub at 240 MHz.

The firmware design of each independent HT array is shown in Fig. 9. It consists of 32 firmware blocks named 'Columns', each corresponding to one of the q/p_{T} columns in the HT array, and a number of firmware blocks named 'Book Keepers', each responsible for managing a subset of Columns. In the current design, there are twelve Book Keepers, each of which communicate with between two and three daisy-chained Columns.

The Book Keeper receives one stub per clock cycle from the input channel, which it stores within a 36 Kb block memory. The Book Keeper then sends the stub data to the first Column that it is responsible for in the HT array. However, as the stub's z coordinate is not needed for the HT, only a subset of the stub information is sent to the Column, consisting of the stub coordinates in the transverse plane (r_T,ϕ_T) with reduced resolution, an identifier to indicate which tracker layer the stub is in, the range of q/p_{T} columns that are compatible with the stub bend, and a pointer to the full stub data stored in the Book Keeper memory. On each clock cycle a stub propagates from one Column to the next Column along the daisy chain managed by the Book Keeper. The components of a Column are shown in Fig. 10.

The stub propagation from Column to Column is based on Eq. 11, where the value of ϕ_T at the right-hand boundary of the n^{th} Column is given by the following calculation, which is carried out in the component labelled 'Hough Transform' in Fig. 10,

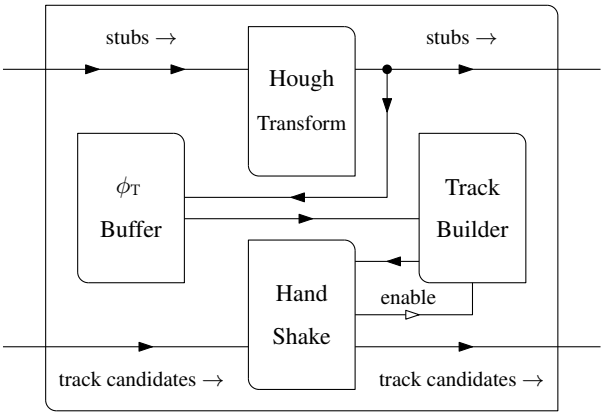


Figure 10: Firmware implementation of one Column, which corresponds to a single q/p_T column in the HT array. A number of Columns are daisy-chained together, starting and ending with the Book Keeper.

$$\phi_T(n) = \phi_T(0) + n \cdot \Delta_{q/p_T} \cdot r_T \ . \tag{12}$$

Here, Δ_{q/p_T} is the fixed width of a q/p_T column, which must be multiplied by an integer n, defining the q/p_T column index. The value $\phi_T\,(0)$ is given by the φ coordinate of the stub. In the firmware algorithm, both $\phi_T\,(n)$ and φ are measured relative to the azimuthal angle of the centre of the sub-sector. Furthermore, the constants appearing in Eq. 11, such as the magnetic field, are absorbed into the definition of q/p_T.

Since the range of q/p_T columns compatible with the stub bend is pre-calculated in the GP, only a comparison is needed to check column compatibility with the bend. Two DSPs are required to carry out the Hough Transform calculation described in Eq. 12, since the $\phi_T(n)$ values of both the left- and right-hand boundaries of the Column are needed for the next step.

In each q/p_T column, the array has 64 ϕ_T cells. Stubs with a steep stub-line gradient can cross more than one (but by construction, never more than two) of these cells within a single column. Such cases are identified by comparison of the values of ϕ_T, from the Hough Transform calculation, at the left- and right-hand boundaries of the column. If a stub is consistent with two cells in the column, then it must be duplicated and buffered within the 'ϕ_T Buffer', from where the second entry will be processed at the next available gap in the data stream. The 'Track Builder' places each stub it receives into the appropriate ϕ_T cell, where it implements the 64 ϕ_T cells using a segmented memory. This uses one 18 Kb block memory, organised as two sets of 64 pages of memory, where the two sets take it in turn to process data from alternate LHC collision events. Each page corresponds to a single ϕ_T cell and has the capacity to store up to 16 stub pointers, so this is the maximum number of stubs that can be declared consistent with an individual cell.

In each ϕ_T cell in the Column, the Track Builder maintains two records of which barrel layers or endcap disks were hit by the stubs stored in the cell. The reason that two records are used rather than one is to profit from the fact that, as described in Section 5.1, the GP sub-divides each sub-sector into two halves in rapidity, and records the consistency of each stub with each of these halves. If the threshold criterion on the number of hit layers/disks is met in either of these two records, then a track candidate has been found, so the cell will be marked for readout. The use of half sub-sector information provides the equivalent to an additional factor of two in η segmentation in terms of the number of track candidates per event, obtained without the cost of doubling the parallelisation, and therefore logic. On the other hand, the fraction of correct

Table 2: Resource utilisation of one Column (in the HT array) and of one entire HT array, as implemented in the Xilinx Virtex-7 XC7VX690T FPGA [16]. The usage as a percentage of the device's available resources are shown in parenthesis. The entire TFP needs 36 HT arrays. The resources needed to implement the multiplexer are not included here, but are relatively small in comparison.

	LUTs	**DSPs**	**FFs**	**BRAM (36 Kb)**
One Column	188 (0.0%)	2 (0.1%)	204 (0.0%)	1 (0.1%)
One HT array	6014 (1.4%)	64 (1.8%)	6718 (0.8%)	33 (2.2%)

stubs on track candidates is unchanged as all the stubs stored in cells meeting the threshold criteria are read out, rather than only those compatible with just one of the sub-sector halves.

The 'Hand Shake' component is responsible for shifting the track candidate stubs from Column to Column, until there are no more stubs in the pipeline. It then enables read out of the Track Builder, such that a contiguous block of stubs from matched track candidates will be created. A track candidate stub now contains a record of the track parameters, ϕ_T and q/p_{T} (as Hough array indices), and a stub pointer, which is used to extract the full stub information from the Book Keeper memory.

To minimise the number of Book Keeper outputs, a multiplexer groups the candidates from six Book Keepers onto a single output, resulting in a total of 72 outputs from the HT per TFP. At this stage load balancing is applied across sub-sectors so that if an excessive number of tracks is found in a single HT array, typically within dense jets, candidates are assigned to different outputs to ensure all data is passed on to the next stage efficiently.

Table 2 shows the resource utilisation of one Column (in the HT array), and of one HT array. Through use of common memory structures it is possible to map the complex Hough Transform array into the FPGA in an extremely compact way. Division of the array into daisy-chained Columns is particularly advantageous, as it enables highly flexible placement and routing possibilities.

5.3 Kalman Filter

5.3.1 Algorithm Description

A Kalman filter was chosen to fit and filter the track candidates produced by the Hough Transform. The filter begins with an estimate of the track parameters and their uncertainties, also referred to as the *state*. Stubs are used, iteratively, to update the state following the Kalman formalism, decreasing the uncertainty in the state with each measurement. A weighting derived from the relative uncertainties in the state and measurement, and termed the Kalman gain, controls the adjustment of the track parameters. The choice of a Kalman filter for the track fitting was guided by the features of the track candidates presented by the HT.

In simulation, over half of the track candidates identified by the HT that match a genuine track contain at least one stub from another particle. Any fit to a stub collection containing incorrect measurements will adversely affect the fitted parameters, so removal of such stubs is desirable. Furthermore, simulations indicate that approximately half of the tracks found by the HT do not correspond to genuine tracks. Discarding these fake tracks, without significant loss of efficiency, is also desirable. The KF is capable of rejecting these incompatible stubs (in addition to fake tracks) 'on the fly', to get the best possible estimate of the track parameters. In addition to the advantages of the Kalman filter for track reconstruction discussed by Frühwirth in [18], the algorithm has several aspects making it suitable for implementation on FPGAs compared

to global track fitting methods:

- The matrices are small, and their size is independent of the number of measurements, meaning logic usage is minimised.
- The only matrix inversion is of a small matrix.

The iterative procedure required by the filter adds some complication, but iteration is not unique to the KF method of track fitting.

The track parameters used in this implementation of the KF are shown in Eq. 13

$$s = (1/2R, \phi, \cot\theta, z_0), \tag{13}$$

where R is the track radius of curvature and related to q/p_{T} according to Eq. 8, ϕ is the azimuthal angle of the track in the transverse plane at the beam line, θ is the polar angle and z_0 is the longitudinal impact parameter.

Assuming that tracks originate at $r = 0$, the track equations expressed in terms of stub radius r are as follows:

$$\varphi = \frac{r}{2R} + \phi, \tag{14}$$

$$z = \cot\theta \cdot r + z_0, \tag{15}$$

where it is evident that these equations are linear in r.

The track equations naturally suggest using the radius r as the stepping parameter in the KF. This is an appropriate choice for the tracker barrel, where modules are arranged in layers of approximately constant radius. However in the detector endcaps the modules have an orthogonal orientation to those in the barrel and this naturally leads to using the z coordinate as the stepping parameter. Since most tracks will pass through modules in the barrel before reaching the endcap, they would preferably be described by two different parametrisations along their trajectory. However, transforming the state across this boundary would require operations on the state vector s and its covariance matrix, and distinct processing blocks for the update of barrel and endcap states would also be needed. For a fast and lightweight FPGA implementation of the KF, this would not be desirable, so instead r is used as the stepping parameter throughout, and the uncertainty in r due to the strip length in endcap modules is folded into the z uncertainty using $\sigma_z^2 = \sigma_r^2(\cot\theta)^2$.

Figure 11 shows the fitting procedure for an example candidate, which is now described. A seeded estimate for the state is obtained from the HT array index (q/p_{T},ϕ_T) and sub-sector in which the track candidate was found. Starting with the seed state and its covariance matrix, stubs are used to update the state, ordered by radius from inside-out. To allow for detector inefficiencies or for the possibility that no compatible stub is found on a given layer, up to two non-consecutive layers may be skipped. In the case that a track candidate contains more than one stub on a given detector layer (when only one is realistic, or occasionally two when detector elements overlap), each combination of stub and incoming state is propagated separately. This eliminates any possibility of the incorrect stub affecting the fit of the genuine combination. The resulting states are ordered, giving preference first to states with fewer missing layers, and then with the smallest χ^2. Only the best state according to this measure is obtained from the filter, so no extra duplicates are introduced.

5.3.2 Implementation

The algorithm itself can be separated into two parts,

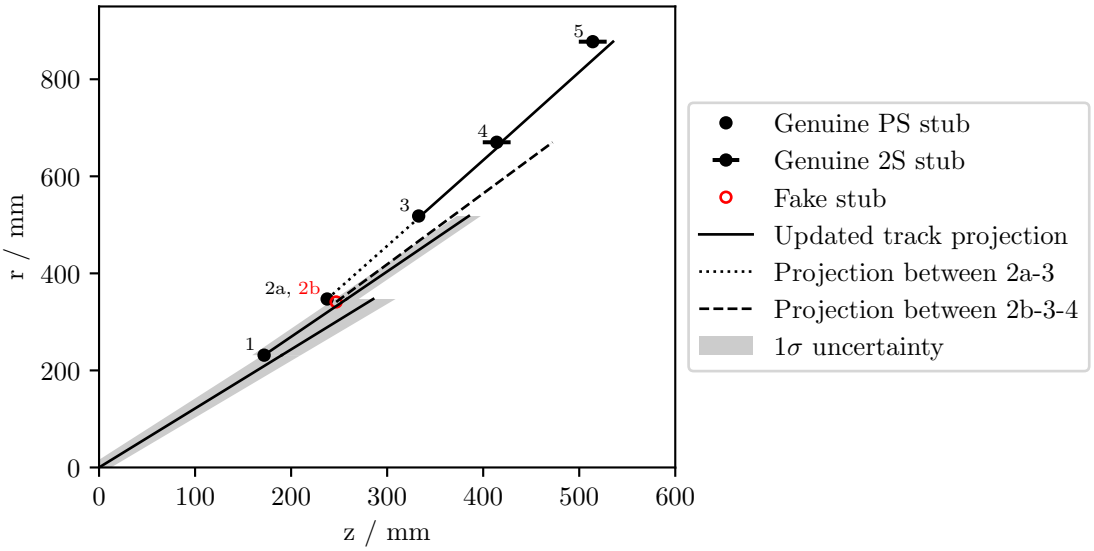


Figure 11: An example of the Kalman Filter fitting procedure for an HT candidate in the barrel, shown in the r-z plane. Genuine stubs are those associated with the same simulated charged particle, and fake stubs are those which are not. Line segments represent the fitted track trajectory at that point of the fit, updating with increasing radius, with the shaded area around the line showing one standard deviation of the track parameter estimate. Dashed track segments highlight the different result after fitting with stub 2a or 2b. The state that includes stub 2b is rejected after propagation to stub 4, due to failing a χ^2 cut in two consecutive layers.

- a 'data-flow' part consisting of the State Update block to carry out the matrix operations described by the Kalman formalism. This logic updates the state, including the track parameters, the covariance matrix, and the χ^2, along with additional parameters to be used for selection or filtering;
- a 'control-flow' part which must gather stub and state information to present it to the State Update block, store and select on updated states, and handle the iterative nature of the algorithm.

The State Update block is implemented in fixed-point arithmetic, which uses fewer resources and clock cycles than floating-point operations. Profiling of the parameters in a C++ simulation was used to tune bit sizes and precision in the design of the firmware. The high level synthesis language MaxJ [19] was used for the implementation of the calculations, and the design benefits from the built-in fixed-point support and pipeline scheduling provided by the tool.

With the Xilinx Virtex 7 series, an 18-bit and 25-bit quantity can be multiplied in a single DSP unit, while two units can be used to multiply one 35-bit and one 25-bit quantity for higher precision. Matrix multiplications are implemented, performing all required multiplications in parallel in separate DSP instances, with a balanced adder tree used for the sums. The higher precision multiplier variant is predominantly used in the covariance matrix update path, while the track parameter update is implemented with single DSPs.

A custom division algorithm was devised for the matrix inversion, which is fast and lightweight, requiring one lookup and one multiplication. Consider the inversion of the 2×2 diagonal matrix $\mathbf{X}$. This matrix is simple enough to invert using the analytic solution:

$$X^{-1} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}^{-1} = \frac{1}{ab} \begin{bmatrix} b & 0 \\ 0 & a \end{bmatrix} = \begin{bmatrix} 1/a & 0 \\ 0 & 1/b \end{bmatrix}. \tag{16}$$

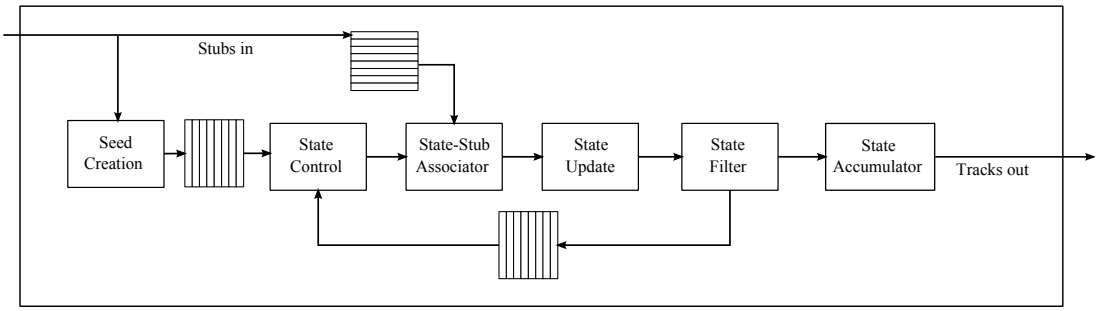


Figure 12: Connection of logical elements within a Kalman Filter worker.

The final expression requires fewer processing steps than the intermediate solution, and allows for finer control over the precision of the two non-zero elements. An implementation of the function $1/x$ is therefore required, which is usually an expensive operation in an FPGA. The algorithm must also be fast, in order to meet the latency requirement. A lookup would be the fastest possible algorithm, but since the divisor is a 25-bit quantity, the cost in memory is too large. As a result, an algorithm using a single 36 Kb memory for a lookup has been developed.

The divisor x can be expressed as the sum of individual powers of two as: $x = \Sigma_n x_n 2^n$ where x_n can be 0 or 1. This sum can in turn be expressed as the sum of two smaller sums:

$$x = \sum_{n=m}^{\infty} x_n 2^n + \sum_{n=0}^{m-1} x_n 2^n = x_H + x_L \tag{17}$$

where m bits are used to encode x_L. Then:

$$\frac{1}{x} = \frac{1}{x_H + x_L} = \frac{1}{x_H\left(1 + \frac{x_L}{x_H}\right)} \approx \frac{x_H - x_L}{x_H^2} \tag{18}$$

where a binomial series, truncated after the second term, was used for the last step. The value of m, that is the number of bits used for x_L, is chosen such that x_H uses 11-bits, and therefore one 36 Kb memory is used to lookup $1/x_H^2$. In the implementation a shift is performed such that the most significant bit of x has value 1, thereby giving the best precision of x_H. A corresponding shift is performed on the result. After the quantity $x_H - x_L$ is calculated, the result is multiplied by $\left(1/x_H^2\right)$ using DSPs.

The control-flow part of the design manages the stub and state data to produce filtered tracks from the KF. Figure 12 shows the connection of the logical elements within a KF worker, and their operation is described below:

- Stubs for a set of track candidates arrive in packets from the HT. Since the algorithm is iterative and an iteration takes many clock cycles, the stubs are immediately stored in memory for later retrieval.
- The Seed Creator outputs the state of Eq. 13 in the required format. As any given HT array index for a given sub-sector can only produce one track candidate, the array index and sub-sector is used as a unique ID for the candidate, providing a reference to the stubs stored in memory at the first step.
- Only one state can enter the State Update block on each clock cycle, and there may be competition between partially worked states and a new candidate arriving into the worker. The State Control block multiplexes the incoming states, giving preference to new candidates.

- The State-Stub Associator block uses the IDs stored with the state to retrieve associated stubs, in order to update the current state. The block determines which iteration the current state is on and passes any stubs within the candidate assigned to the next layer, or even the next-to-next layer in the case of a skipped layer, one per clock cycle. Stubs from the next-to-next layer can only be forwarded to the State Update block if the current state indicates that it has not skipped two layers already.
- The Kalman filter is run in the State Update block using the current state in association with the stub. The track parameters, covariance matrix, χ^2 value, and other status information are all updated for the next iteration.
- At the output of the State Update block, any states that fail a set of configurable cuts are immediately discarded. The State Filter is able to select against states based on p_T, χ^2, z_0, sub-sector compatibility and a minimum requirement on number of stubs from PS modules. Additionally the State Filter is capable of preserving the best N output states, by χ^2, for a given state from the previous iteration. On the first iteration the best four states are kept, on subsequent iterations this is reduced to one. This helps minimise the total number of states circulating in the worker at any point in time.
- The surviving states are written into a FIFO, to complete further iterations of the Kalman filter. A completed track is one where a state has finished four iterations of the KF, after which the state is no longer re-inserted into the FIFO.
- The surviving states are also presented to the State Accumulator where the best state for each candidate is stored until an accumulator time-out signal is propagated, and the fitted tracks are read out. In the accumulator, preference is given to states with fewer missing layers, and then with the smallest χ^2. This block allows readout of partially filtered states on receipt of the time-out, which may occur in particularly dense jets with many candidates and many stubs per candidate.

The resource usage of a single KF worker is summarised in Table 3. As the resource usage is small compared to the total available, multiple filter workers can be used in parallel.

Each logical element in Fig. 12 is implemented with a fixed latency. The latency of a single KF iteration is dominated by the matrix operations involved in the State Update block, which takes 55 clock cycles. With a 240 MHz clock frequency this is 230 ns. At each iteration, multiple stubs go into, and (after a 55 clock cycle delay) come out of, the State Update block on subsequent clock cycles. Allowing independent propagation of multiple stubs on a layer slightly increases the total latency compared to just four passes of the single iteration latency. An accumulation period of 1550 ns before time-out is set, after which point all tracks, completed or uncompleted, for one event are output. Measurements (as described in Section 7) show that fewer than 0.1% of tracks in $t\bar{t}$ events with 200 PU fail to be fully reconstructed within this accumulation period. Since the state keeps track of the current iteration (identical to the number of stubs on the state), quality cuts can be placed on the final tracks, if, for example, only completed KF tracks

Table 3: Resource utilisation of the Kalman Filter state update block, and one full Kalman Filter worker, as in the Xilinx Virtex-7 XC7VX690T FPGA [16]. The usage as a percentage of the device's available resources are shown in parenthesis. For a single TFP, a total of 72 workers processing data from 36 HT arrays are used.

	LUTs	**DSPs**	**FFs**	**BRAM (36 Kb)**
State Update block	4014 (0.9%)	70 (1.9%)	3094 (0.4%)	6 (0.4%)
One Kalman worker	5520 (1.3%)	71 (2.0%)	4370 (0.5%)	24.5 (1.7%)

are required.

5.4 Duplicate Removal

The Duplicate Removal algorithm is the last element in the Track Finding Processor chain. At the input to the DR, over half the track candidates are unwanted duplicate tracks created by the HT, and the purpose of the DR is to eliminate these.

The DR algorithm is based on an understanding of how duplicate tracks form within the HT. This is illustrated in Fig. 13, where in the example shown, five stubs (blue lines) from a single particle produce three track candidates in the green and yellow HT cells. Since these three tracks contain the same stubs, when they are fitted they will all yield identical fitted track parameters. These fitted parameters should correspond to the yellow cell, where the lines intersect.

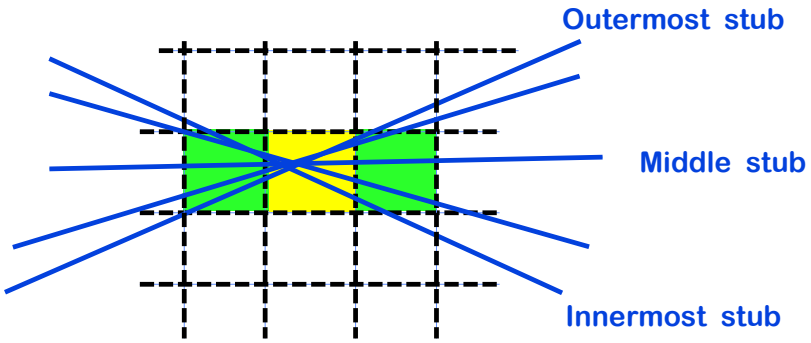


Figure 13: r-φ Hough Transform showing formation of duplicates. The yellow cell represents the genuine track-candidate, whereas the green cells depict duplicate track candidates generated within the HT by the same set of stubs.

Based on the above, the DR algorithm can be described as follows: after the track fitter, any track whose fitted parameters do not correspond to the same HT cell as the Hough Transform originally found the track in, is eliminated. Hence, in the example of Fig. 13 the green cells will be eliminated and the yellow cell will be kept. The advantage of this algorithm is that it identifies duplicates by looking at individual tracks. As a result, there is no need to compare pairs of tracks in order to find out if they are similar. There is however a small subtlety: the described algorithm loses a few percent of efficiency due to resolution effects. The efficiency can be recovered by performing a second pass through the rejected tracks. During that pass, tracks whose fitted parameters do not correspond to the HT cell of a track from the first pass are probably not duplicates, so they are rescued.

5.4.1 Implementation

The implementation of the duplicate removal algorithm is shown in Fig. 14. The DR block shown in that figure processes the tracks found by the KF in six sub-sectors, so six such DR blocks must be instantiated to process tracks from all 36 sub-sectors in the processing octant. Designing the DR block to process six sub-sectors instead of one minimises the resource usage.

Within the DR block, a 'Matrix' representing the HT arrays of the six sub-sectors is implemented in a 18 Kb memory, and is addressed using the sub-sector number and (q/p_{T}, ϕ_T) cell location within the HT array. Any KF track that is flagged as 'consistent' (i.e., its fitted helix parameters correspond to the same HT cell as the HT originally found the track in) is forwarded

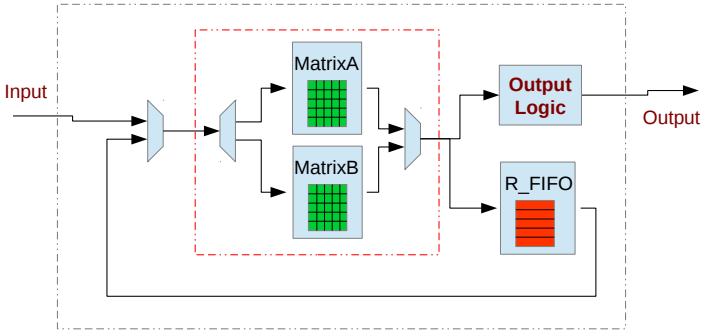


Figure 14: Architecture of the Duplicate Removal algorithm implementation. A single DR logic block is shown, which processes the KF tracks from six sub-sectors (which arrive via 'input'). Therefore, six such blocks are needed to process all 36 sub-sectors in the processing octant.

to the output channel, and in addition, the corresponding Matrix address is marked. In contrast, tracks which are 'inconsistent' are added to a FIFO (named 'R_FIFO'). After all tracks have arrived from the KF, the inconsistent tracks are read out from R_FIFO, and if one has fitted track parameters corresponding to an HT cell location not yet marked in the Matrix, the track is rescued by forwarding it to the output channel, and marking the corresponding address in the Matrix.

A complete reset of the Matrix is required before processing tracks from another LHC bunch crossing, so two Matrices (labelled 'Matrix A' and 'Matrix B' in Fig. 14) are instantiated, which take it in turn to process alternate LHC events. There are thus always one active Matrix and one resetting Matrix. Along with them, two 'clear' FIFOs are used, one for each matrix, to store the addresses that were marked and hence need to be cleared in readiness for a new event. Each Matrix plus its corresponding clear FIFO occupy one 36 Kb memory block.

The FIFO in which the 'inconsistent' tracks are temporarily stored uses two 36 Kb block RAMs. Therefore, a total of four 36 Kb block RAMs are used for the entire DR block design, which handles six sub-sectors. As well as being a lightweight design, it also has a low latency of only four clock cycles. The total resource utilisation, including other types of resources, is reported in Table 4.

Table 4: Resource usage of a single Duplicate Removal block for six sub-sectors, as implemented in the Xilinx Virtex-7 XC7VX690T FPGA [16]. The usage as a percentage of the devices' available resources are shown in parenthesis. The entire TFP needs six of these DR blocks.

	LUTs	DSPs	FFs	BRAM (36 Kb)
One Duplicate Removal block	291 (0.1%)	0 (0.0%)	496 (0.1%)	4 (0.3%)

6 The Hardware Demonstrator Slice

A demonstrator system has been constructed in order to implement a slice of the proposed L1 track finder on real hardware, and to measure and validate its performance within the latency constraints. Input data to the demonstrator, in the form of stubs, is generated using CMS simulation software (CMSSW) for the tracker geometry illustrated in the lower diagram of Fig. 3, using Monte Carlo physics events generated under HL-LHC conditions.

The track finder slice corresponds to one Track Finding Processor as described in Section 5 and is designed to allow the demonstration of the concept using currently available technology. While the slice processes data from 1/8 of the tracker in φ, and all of the tracker in η,

since each TFP operates independently from the other, one can run data for all eight φ-octants sequentially, allowing the entire event to be reconstructed in hardware.

Located at the CERN Tracker Integration Facility (TIF), the demonstrator consists of one custom dual-star MicroTCA [20] crate, equipped with a commercial NAT MicroTCA Carrier Hub (MCH) for Gigabit Ethernet communication via the backplane, and a CMS specific auxiliary card known as the AMC13 [21] for synchronisation, timing and control. The TFP algorithms are implemented on a set of five Imperial Master Processor, Virtex-7, Extended Edition (MP7-XE) double width AMC cards [22].

Designed for the CMS L1 time-multiplexed calorimeter trigger, each MP7 is equipped with a Xilinx Virtex-7 XC7VX690T FPGA, and twelve Avago Technologies MiniPOD optical transmitters/receivers, providing 72 optical link pairs each running at up to 12.5 Gb/s, for a total optical bandwidth of 0.9 Tb/s in each direction. For the demonstrator, the links are configured to run at 10.0 Gb/s with 8b/10b encoding for an effective 8 Gb/s transfer rate. As a result the system bandwidth is a factor of two smaller than that defined in Section 4 where a 16.3 Gb/s system is required (assuming 64b/66b encoding). This is accommodated in the demonstrator by using a time-multiplexed factor of 36 instead of 18. A discussion of how the hardware and algorithms would scale to the full system is provided in Section 8.

Infrastructure tools are provided with the MP7, including core firmware to manage transceiver serialisation/deserialisation, data buffering, I/O formatting, board and clock configuration as well as external communication via the Gigabit Ethernet interface. Any algorithms deployed are segregated from the firmware responsible for these tasks, allowing a system such as the demonstrator to be built up of processing blocks, each running on a single MP7-XE, daisy-chained together with high-speed optical fibres. Division of the demonstrator in this way allows firmware responsibilities to be easily divided between personnel, provided I/O formats between the processing blocks are defined. By parallelising or daisy-chaining of algorithms across multiple boards the ability to estimate final system performance, without limitations in the resources available in the technology used, is achievable. As such an upper limit on the total FPGA logic requirements for a future processing card can be extracted from the demonstrator.

The firmware components and the connections between them are shown in Fig. 15 and relate to the components described in Section 5. Eight MP7-XE boards are currently used for the demonstrator chain. Two boards, named sources, each represent data from a set of up to 36 DTCs. Each source board is implemented as a large buffer for the storage of stub data from a detector octant, where the data is loaded directly from simulation via IPBus [23]. Each output stream from the source boards represents a separate DTC injecting pre-formatted 48-bit stubs into the Geometric Processor and is capable of playing up to 30 consecutive events through the demonstrator. Two sources are required to emulate how data from two adjacent detector octants can feed a single TFP for tracks that cross the detector boundary. The TFP itself is implemented on five boards: one being used for the GP, two for the HT, and two more for the KF and DR. One additional board, the sink, is used to capture the track-finder output from up to 30 simulated physics events before being read out, again with IPBus. For standalone testing of firmware blocks, or parallel data taking alongside the full chain, an additional three boards are also installed in the demonstrator crate. The demonstrator crate is shown in Fig. 16.

7 Demonstrator Results

Simulated physics events (typically top quark pair-production, $t\bar{t}$) with a PU of up to 200 proton-proton interactions per bunch crossing were produced with the CMS simulation soft-

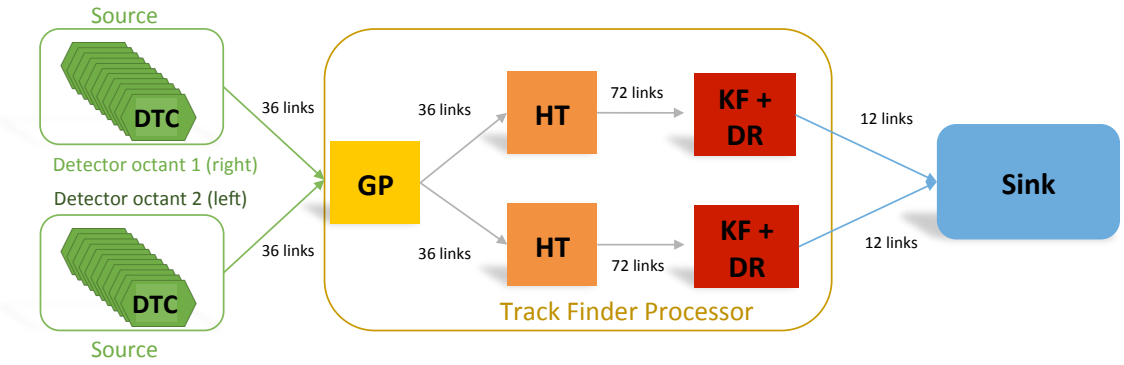


Figure 15: The demonstrator system consists of five layers of MP7s; source, Geometric Processor (GP), Hough Transform (HT), Kalman Filter + Duplicate Removal (KF+DR), and sink. A total of eight MP7-XE boards are used, each indicated by a separate coloured block in the diagram.

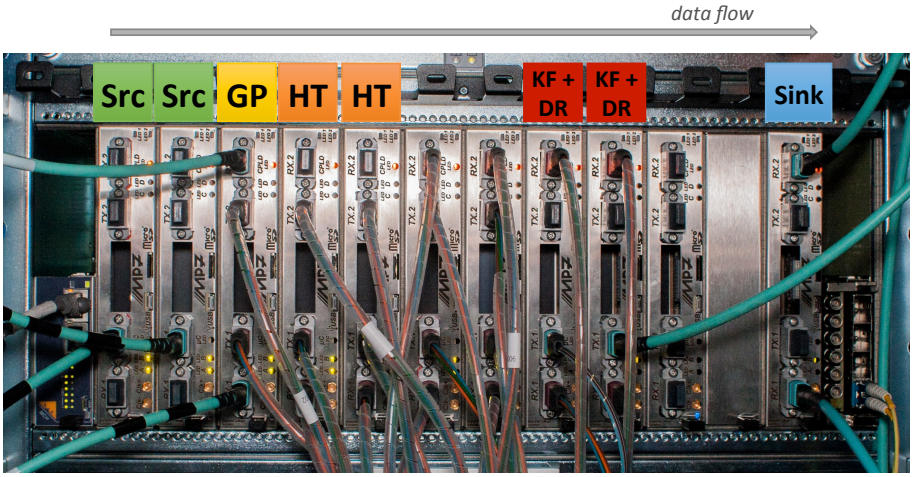


Figure 16: The demonstrator crate is equipped with 11 MP7-XE boards, an AMC13, MCH and the required optics.

ware (CMSSW), including modelling of particle interactions with the detector and the generation of stubs.

Software developed to study the performance of the hardware slice is used to inject stubs from these samples into the demonstrator chain, converting them to a text file format before transmission over IPBus. Tracks reconstructed by the demonstrator using these stubs are retrieved via IPBus at the end of the chain and are stored for later analysis.

An *emulation* of the hardware chain has also been developed in software, which is able to process the same integer formatted stub data used as input to the demonstrator to produce tracks, for offline validation with hardware output. The emulator uses fixed-point mathematical operations where appropriate and simulates the logic implemented within the FPGAs as closely as possible in order to model time-dependent effects. However, as it is not a clock-cycle accurate emulation, small differences between hardware and emulation are occasionally expected. The emulator software can be optionally configured to use full floating-point precision for comparison. Both emulated and hardware tracks are analysed by a comparison software package, which checks for consistency on an event-by-event basis.

7.1 Track Reconstruction Efficiency

The track reconstruction efficiency is measured relative to all generated charged particles from the primary interaction that produce stubs in at least four layers of the tracker, and lie within the kinematic acceptance ($p_T > 3\,\text{GeV}$, $|\eta| < 2.4$, $|z_0| < 30\,\text{cm}$ and $L_{xy} < 1\,\text{cm}$, where L_{xy} is the transverse distance from the beam line to the particle vertex). A charged particle is defined to be successfully reconstructed and contributes to the efficiency if:

- a reconstructed track has stubs associated to the particle in at least four different tracker layers;
- this reconstructed track has no incorrect stubs (i.e. all its stubs were produced by the same particle).

However, the second of these two matching requirements is only imposed when quoting results from the entire demonstrator (i.e, the 'full chain'), since it is natural that some incorrect stubs are present if only part of the chain is used. Successfully reconstructed tracks are also known as *matched* tracks. Unmatched reconstructed tracks are also known as *fake tracks*. If a charged particle matches more than one reconstructed track, subsequent tracks are labelled *duplicates*.

Table 5 shows how tracking performance evolves as data progresses through the reconstruction chain, with these results obtained without imposing the second of the two matching requirements mentioned above. It can be seen that the HT finds tracks with high efficiency, but many of the tracks are either fake or duplicate tracks. The KF eliminates the majority of the fake tracks whilst the duplicate removal algorithm removes almost all the duplicates. For the full chain, 100.0% of the tracks satisfying the first of the two matching criteria above also satisfy the second one, mainly due to the ability of the KF to reject incorrect stubs.

Table 5: Track finding performance on simulated $t\bar{t}$ events at a PU of 200, after each stage of the demonstrator chain. The track finding efficiencies at each stage are listed using the efficiency definitions given in the text. Also quoted are the mean numbers of reconstructed tracks per event in the entire tracker, and the subset of these tracks that are unwanted as they are either fake or duplicate tracks.

Stage	Efficiency [%]	Total # of tracks	# of fakes	# of duplicates
HT	97.1	331	139	126
KF	95.1	190	27	103
DR	94.4	79	16	3
Full chain	94.4	79	16	3

The mean tracking efficiency (over all applicable $|\eta|$), in $t\bar{t}$ events with 200 PU, as measured in hardware, is 94.5%. This number agrees to within 99.5% with the results generated by emulation. The agreement when studied as a function of the particle kinematic properties is shown in Fig. 17. The tracking efficiency from emulation is identical regardless of whether floating-point or integer precision stub data are used.

The efficiency to reconstruct leptons in the $t\bar{t}$ events exceeds 97% for muons over the entire acceptance, but is somewhat lower for electrons, as shown in Fig. 18. The loss of electron efficiency is expected and is mainly due to bremsstrahlung effects, which cause the particle trajectory to deviate from the helix assumed by the tracking algorithm. Some recovery of this efficiency loss should be possible. For example, the KF algorithm can be modified to allow for multiple scattering. The agreement between hardware and emulation is equally good for leptons.

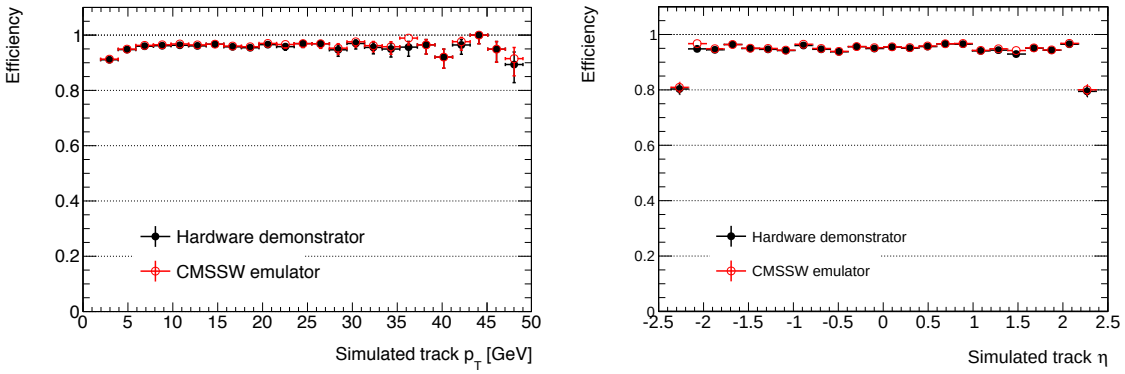


Figure 17: Track reconstruction efficiency, measured in both hardware and emulation, for tracks originating from the primary interaction in $t\bar{t}$ events with 200 PU, as a function of p_T (left) and η (right).

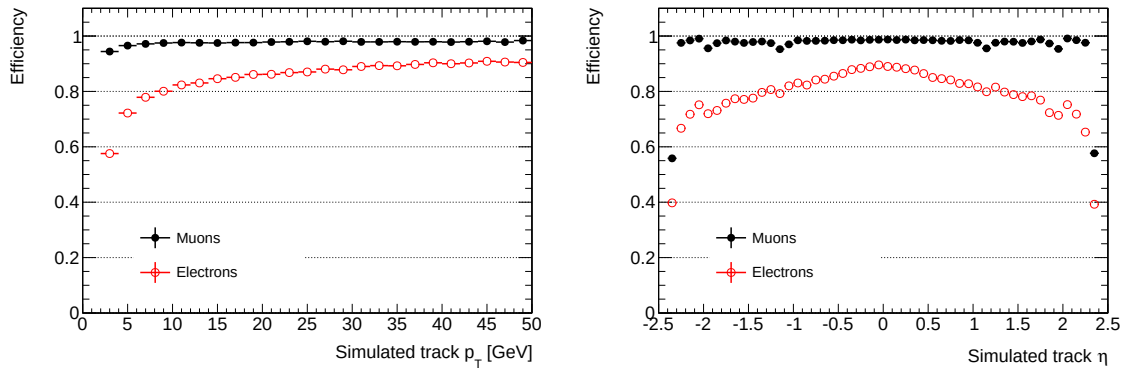


Figure 18: Track reconstruction efficiency, for electrons and muons from $t\bar{t}$ events with 200 PU, as a function of p_T (left) and η (right). These results are obtained from emulation.

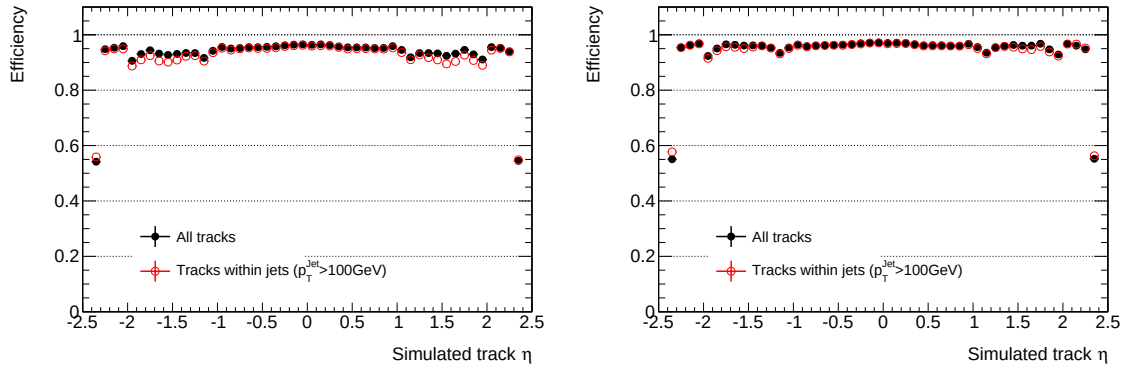


Figure 19: Track reconstruction efficiency as a function of η, for all tracks originating from the primary interaction (black dots), or for only the tracks contained within a primary jet that has a total transverse momentum exceeding 100 GeV (red open circles), for $t\bar{t}$ events at 200 PU. Either no incorrect stubs are allowed on the track (left), or one incorrect stub is allowed on the track at most (right). These results are obtained from emulation.

The quality of tracks in the core of dense jets is slightly degraded due to the increased likelihood of incorrect stubs being included in the track candidate. This can be seen even in our default sample of $\mathrm{t\bar{t}}$ at 200 PU. Figure 19 shows that when selecting on charged particles in jets which have total transverse momentum exceeding 100 GeV, there is a small efficiency loss, particularly in the region $|\eta| > 1$. As shown in Fig. 19, when adjusting the second matching requirement to allow reconstructed tracks with at most one incorrect stub to contribute to the efficiency this effect is reduced. This indicates that much of the loss in the region $|\eta| > 1$ comes from the reduced track purity in these high-energy jets. Better rejection of these incorrect stubs by the KF should help improve the overall performance of the track-finder.

7.2 Track Parameter Resolution

Figure 20 shows the resolutions of the four track parameters $(p_\mathrm{T}, \phi, \cot\theta, z_0)$ for reconstructed primary tracks from both hardware and emulation in $\mathrm{t\bar{t}}$ events at PU of 200. The fifth track parameter, d_0, is not currently reconstructed by the KF, but may be in the future (see Section 8 for further discussion). The resolutions compare well to those obtained with offline track reconstruction [9], and good enough to ensure that the tracks will be useful to the L1 trigger. The level of agreement between hardware and emulation is reasonable, with remaining differences due to the use of floating-point arithmetic in parts of the emulator code. The degradation in resolution with increasing pseudorapidity is expected, due to a combination of the shorter effective lever arm available, the reduced effective precision for hits in the endcap and the impact of increasing material traversed by particles.

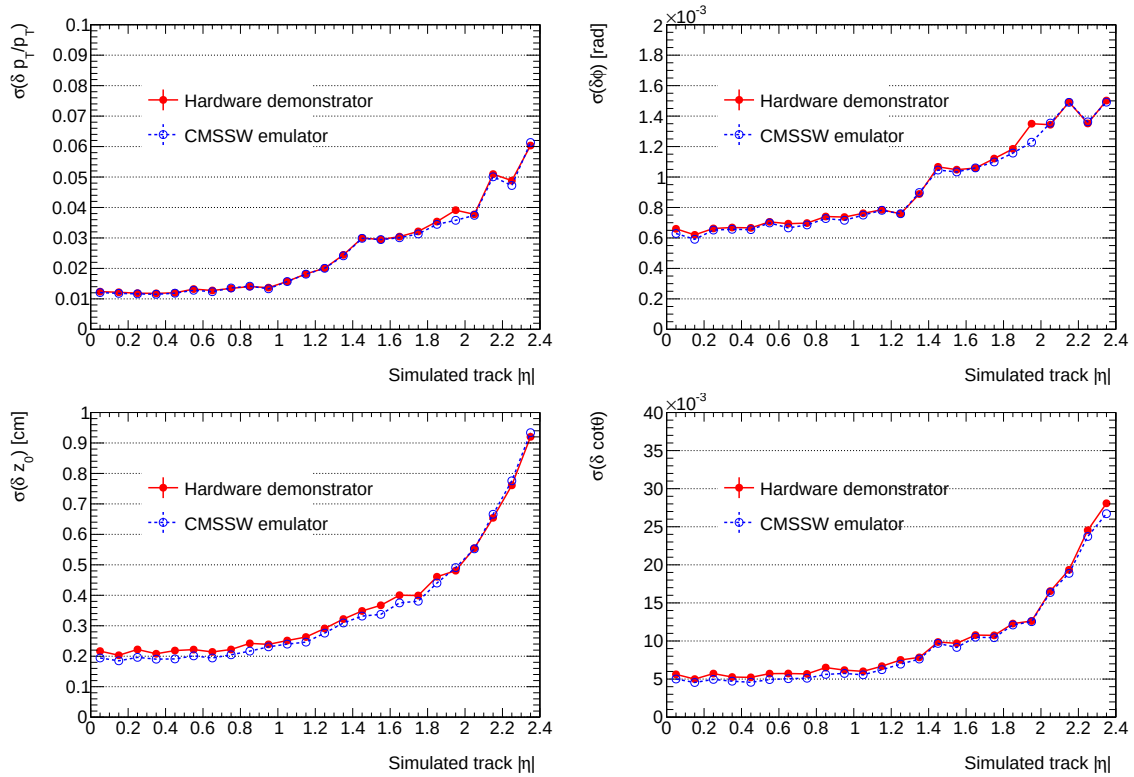


Figure 20: Relative p_T resolution, ϕ resolution, z_0 resolution and $\cot\theta$ resolution measured for tracks originating from the primary interaction in $\mathrm{t\bar{t}}$ events at PU of 200 as determined from both hardware and emulation.

It is also instructive to compute the parameter resolutions for particles with different transverse momenta. The resolution of the track parameters for single isolated muons, as measured using

the emulator only, is provided in Fig. 21. Multiple scattering effects dominate at low transverse momenta and this is particularly evident in the estimate of the ϕ parameter, where the resolution is better than 0.4 mrad for muons with $15 < p_{\mathrm{T}} < 100$ GeV and between 0.7 mrad and 1.5 mrad for muons with $3 < p_{\mathrm{T}} < 5$ GeV. Similar effects are observed for the $\cot\theta$ resolution at large pseudorapidities. The relative precision of the transverse momentum for muons with $p_{\mathrm{T}} < 15$ GeV is limited by multiple scattering, but conversely degrades with increasing p_{T} due to the decreasing radius of track curvature.

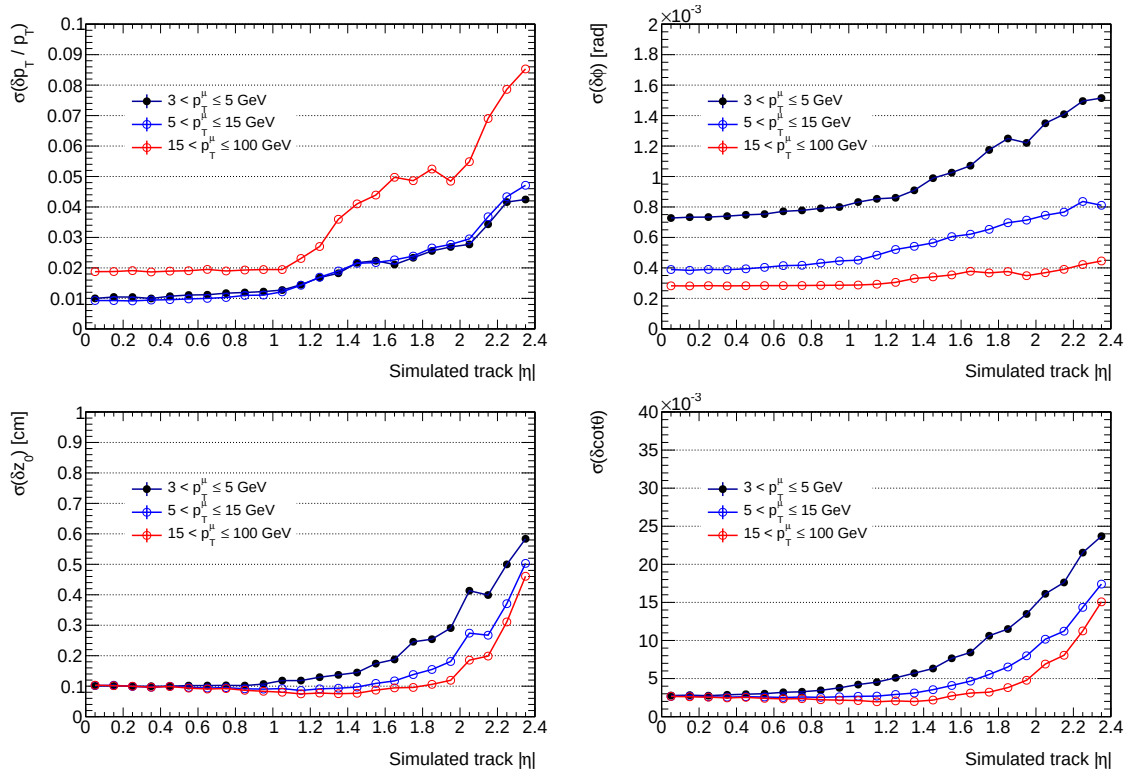


Figure 21: Relative p_{T} resolution, ϕ resolution, z_0 resolution and $\cot\theta$ resolution measured for single isolated muons with $3 < p_{\mathrm{T}}^{\mu} < 5$ GeV, $5 < p_{\mathrm{T}}^{\mu} < 15$ GeV, and $15 < p_{\mathrm{T}}^{\mu} < 100$ GeV. These results are obtained from emulation.

The resolution of the four parameters in Fig. 21 also compares well with offline simulation [9], which is able to utilise all available information from the tracking system and more sophisticated reconstruction algorithms. In offline simulations, for 10 GeV muons passing through the centre of the tracker barrel, the ϕ resolution is approximately 0.2 mrad while the p_{T} resolution is ∼0.5%. On the other hand, mostly due to the inclusion of hit information from the pixel detector, the precision of the remaining two parameters is more than an order of magnitude better in offline simulations compared to the demonstrator.

In comparing the z_0 resolution of single isolated muons in Fig. 21 with earlier simulation studies of the track finder [4], the demonstrator appears to show approximately half the expected precision in the barrel. This degradation of resolution in the demonstrator system comes as a result of choosing to encode the r and z stub coordinates too coarsely (using 10 and 12 bits, respectively) and therefore reducing the ultimate precision of the track parameters. Figure 22 shows that improving the encoding of the stub coordinates by assigning an additional two extra bits to both r and z recovers the lost precision, meaning that the z_0 resolution reaches ∼1 mm for muons with $5 < p_{\mathrm{T}} < 15$ GeV and $|\eta| < 2$. A small improvement can also be observed for the $\cot\theta$ resolution as well. This change can be implemented without degrading

the performance elsewhere in the demonstrator system. Figure 22 also shows that with this improved encoding scheme, the precision of all four parameters approaches that obtained by the floating-point simulation of the demonstrator.

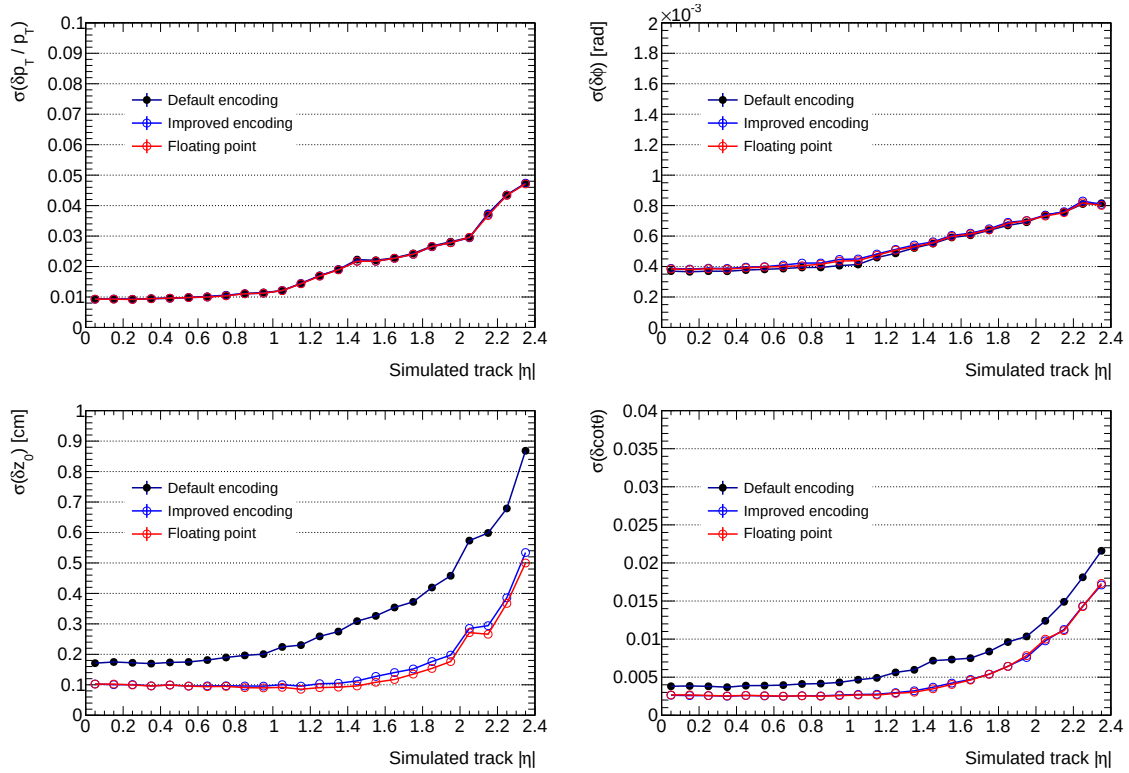


Figure 22: Relative p_{T} resolution, ϕ resolution, z_0 resolution and $\cot\theta$ resolution measured for single isolated muons with $5 < p_{\mathrm{T}}^{\mu} < 15\,\mathrm{GeV}$ obtained from emulation, using different levels of precision in simulation: default encoding (10-bit r, 12-bit z, 15-bit ϕ stub coordinates); improved encoding (12-bit r, 14-bit z, 15-bit ϕ stub coordinates); and full floating-point simulation.

7.3 Data Rates

As shown in Fig. 23, the number of tracks reconstructed per event increases with increasing pile-up. On average, 79 tracks/event are reconstructed in $\mathrm{t\bar{t}}$ events with 200 PU.

The demonstrator system can cope comfortably with the high data rates present in these events. Figure 24 (left) shows the distribution of the number of stubs per event transmitted from the GP to the HT in each sub-sector. Truncation of data would occur if these stubs could not all be sent within the 900 ns period that is defined by the time-multiplexed factor in the demonstrator. As stubs from each sector are transmitted at 240 MHz, this corresponds to a theoretical limit of 216 stubs per sub-sector, although in the current system the limit is actually about 175, due to gaps between the output stubs. This limit exceeds the average data rate by almost a factor two, so truncation effects in this part of the system are small: 0.3% of stubs are lost in $\mathrm{t\bar{t}}$ events at 200 PU, which in turn leads to a 0.5% loss of tracking efficiency.

Figure 24 (right) shows the number of reconstructed tracks in each sector output by the HT per event. It is striking that 70% of sectors contain no reconstructed tracks, which occurs because the 3 GeV p_{T} threshold used for track reconstruction is very effective at suppressing tracks from PU interactions, while 97.5% of sectors contain fewer than ten reconstructed tracks. Since on average each track is assigned about seven stubs by the HT, there is usually no difficulty in

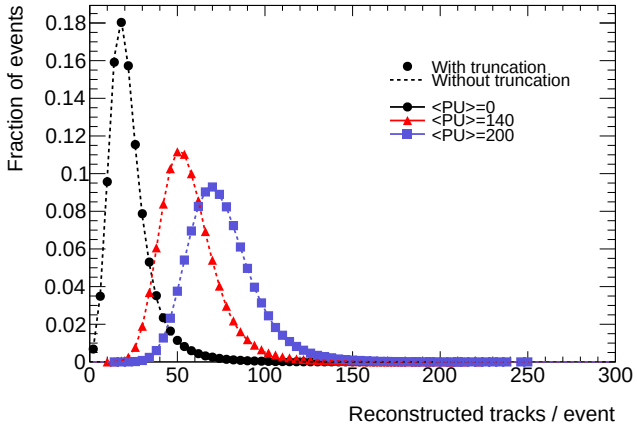


Figure 23: Total number of reconstructed tracks per event reconstructed in the tracker when processing $t\bar{t}$ events superimposed with 0, 140, and 200 PU events. These results are obtained from emulation, and are shown for when effects of truncation, caused by excess data flow through the system, are both included and excluded.

outputting tracks from the HT within the time-multiplexed limit. The only challenging case is due to collimated, high-energy jets from the $t\bar{t}$ system itself, which produce many particles and stubs in a narrow angular region, accounting for the tails seen in Figure 24. The load-balancing introduced at the back-end of the HT, as discussed in Section 5.2.2, addresses this challenge. The loss in tracking efficiency due to truncation at the output of the HT is below 0.1% when processing $t\bar{t}$ events at 200 PU.

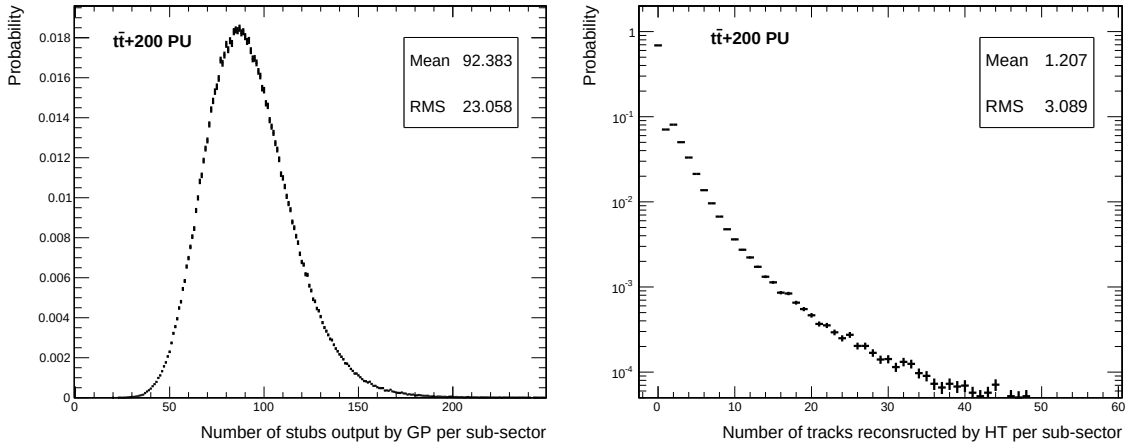


Figure 24: Illustration of data rates in key parts of the system when processing $t\bar{t}$ events with 200 PU. The left-hand plot shows the number of stubs transmitted from the GP to the HT per sub-sector per event. Truncation effects occur when this number exceeds about 175. The right-hand plot shows the number of reconstructed tracks from the HT per sub-sector per event.

As mentioned in Section 5.3.2, the latency of the KF track fitter is set at a value such that it has time to assign stubs from four tracker layers to almost all tracks. When processing $t\bar{t}$ events at 200 PU less than 0.1% of efficiency is lost in the KF when selecting on these four stub tracks. Again, this loss occurs mainly within particularly high-energy jets.

The loss in tracking efficiency from truncation effects of the tracking chain is determined to be less than 0.6% when processing $t\bar{t}$ events at 200 PU, as shown in Fig. 25.

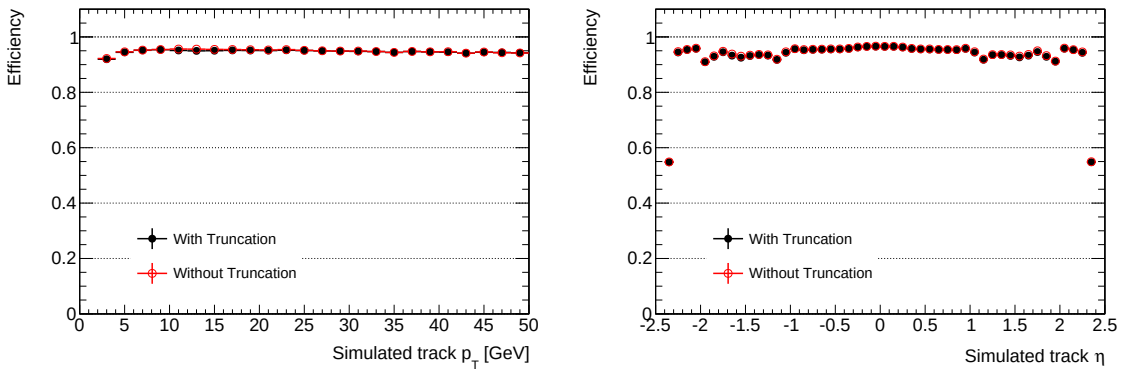


Figure 25: Track reconstruction efficiency for tracks originating from the primary interaction in $t\bar{t}$ events with 200 PU, as a function of p_T (left) and η (right). These results are obtained from emulation both including and excluding truncation effects.

Table 6: Total resource usage for the demonstrator TFP (with time-multiplexed factor of 36), as implemented in the Xilinx Virtex-7 XC7VX690T FPGA [16]. The resources needed to construct a complete TFP correspond to the sum of the numbers from the three rows labelled GP, HT, and KF and DR.

	LUTs [10^3]	DSPs	FFs [10^3]	BRAM (36 Kb)
GP	121	1056	205	222
HT	244	2304	299	1188
KF and DR	398	5112	316	1776
Infrastructure per MP7	90	0	91	291
TFP Total (excl. infrastructure)	763	8472	820	3186
Virtex 7 690	433	3600	866	1470

7.4 FPGA Resource Usage

Constructing the hardware demonstrator out of many MP7 boards avoids the logic constraints of a single, currently available, FPGA processing board. On the other hand, it is important to keep the total resource usage realistic such that a final system could be built at a reasonable cost, using FPGAs expected to be available on the timescale for production. Table 6 shows the total FPGA resource usage for each demonstrator component (where the numbers given for the HT and KF implementations are summed across the two boards used for each component). The combined total for the three components gives the resources used to demonstrate the functionality of one entire TFP with a time-multiplexed factor of 36. Each FPGA in the demonstrator also runs the MP7 core infrastructure firmware, which is required for board configuration, link buffering and error checking. This firmware was developed for the CMS calorimeter trigger, and while it does not constitute a significant fraction of logic in the TFP (as shown in Table 6), it is expected that with some optimisation it could be reduced in size and still deliver the functionality needed for the track-finder. In order to meet timing and routing constraints in the Virtex-7, the designs often prioritised the use of block RAM over LUT based distributed memory. This balance could be readdressed in the future, as the design is adapted to newer FPGAs.

7.5 Latency

Latency measurements of the full demonstrator chain have been made for each block independently and also for the total chain. These are shown in Table 7. Measurements include optical transmission delays and serialisation/de-serialisation (SERDES) latency on the links. The en-

tire demonstrator chain of source to sink and the sum of each individually measured layer give identical results. The latency of the complete system is fixed, regardless of PU or the number of events occupying the sector. In addition to the time difference between the first stub entering the system and the first track leaving it, the table also shows the time difference between the first stub entering and the last track leaving. Both these latency definitions are of interest to the L1 trigger, which will sit downstream of the track-finder system.

Table 7: Measured latency of the demonstrated components of the track reconstruction chain, including the serialisation/de-serialisation (SERDES) and optical transmission delays between each board.

System latency	Latency [ns]
SERDES + optical length 1	143
Geometric Processor	251
SERDES + optical length 2	144
Hough Transform	1025
SERDES + optical length 3	129
Kalman Filter + Duplicate Removal	1658
SERDES + optical length 4	129
Total: First out - First in	3479
Last out - First out	225
Total: Last out - First in	3704

7.6 Flexibility and Robustness of the System

The exceptionally low truncation rates at each stage of the demonstrator chain under challenging conditions, as reported in Section 7.3, are an indication of the margin in the system.

The latitude in the HT and KF stages to increased data flow in the system can be illustrated by reducing the threshold criteria in the HT from five hit layers (four in specific regions) to four hit layers over the entire tracking acceptance. Reducing this requirement would provide a significant level of robustness to detector inefficiencies or failures but at the cost of increasing the number of track candidates output by the HT by a factor 3.6, to 1190 candidates for $t\bar{t}$ at 200 PU. The demonstrator chain is able to process this much larger data rate, whilst keeping the tracking efficiency loss due to truncation to about 1.7%, as shown in Fig. 26 (left). This small additional loss of $\sim$1% with respect to the nominal configuration does not occur in the HT, but is caused by the lack of time for the KF to add stubs from four layers to all tracks. Increasing the accumulation period in the KF would recover this loss if required, at the expense of latency, although other optimisations to the design should mean that this is not necessary (Sec. 8). Alternatively, using the nominal threshold criteria, this significant margin could instead be sacrificed to extract latency or FPGA resource savings, or a balance between the two.

In the nominal configuration, to understand the level of performance in a situation where a fraction of modules in the tracker does not produce stubs, the demonstrator was tested on samples that emulated this scenario. Figure 26 (right) illustrates the localised loss in efficiency expected when all modules on barrel layer four, in the region $-1 < \eta < 0$ and $0 < \varphi < \pi$, are prevented from generating stubs in simulation. As shown in Fig. 26 (right), this efficiency loss can be recovered by relaxing, in the affected (η, φ) sub-sectors only, the threshold criterion on the number of hit layers in the HT from five to four. This threshold change leads to only a small increase in data rate, as shown in Table 8. The increase is caused by extra fake tracks, which occur as a result of the looser threshold.

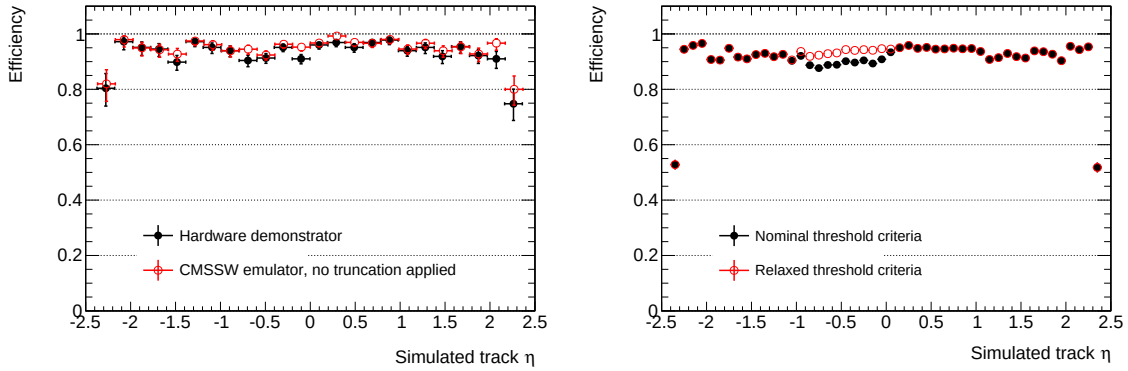


Figure 26: Left: track reconstruction efficiency as a function of η, measured in both hardware and emulation, when processing $t\bar{t}$ events with 200 PU, where a global threshold criterion of four hit layers in the HT has been applied. To demonstrate the total efficiency loss due to truncation in hardware, the emulation result does not include truncation effects. Right: track reconstruction efficiency as a function of η, measured in emulation, when processing $t\bar{t}$ events with 200 PU, where the tracker is affected by a failure of all modules in the region $-1 < \eta < 0$ and $0 < \varphi < \pi$ of barrel layer 4. Results are compared before (black dots) and after (red open circles) relaxing the threshold criterion on number of hit layers in the affected region, as described in the text.

Table 8: Study of a module loss scenario. The mean number of tracks from the HT per event is shown, when processing $t\bar{t}$ events with 200 PU.

No module loss	**With module losses**	
	Before recovery	**After recovery**
330	304	347

The CMS collaboration has recently developed a new proposal for the layout of the upgraded Outer Tracker, known as the tilted barrel geometry, which is described in Section 2. Preliminary Monte Carlo samples with the new geometry were made available to evaluate the impact of the design change on the track finding demonstrator. Results indicate that the use of the tilted barrel geometry substantially reduces the data rate out of the HT, as shown in Table 9. This is mainly due to the fact that the average number of stubs in the innermost three barrel layers, per event, is lower in the tilted case. The table also demonstrates that performance at the end of the full chain is similar for the two geometries. This suggests that the margin in the system when applied to the new geometry will become considerable, meaning that significant savings in FPGA resources, latency, and overall system scale should be achievable. Thus far, no optimisation of the tracking algorithm has been made for the tilted barrel geometry, so these

Table 9: Performance comparison between flat and tilted barrel tracker geometries using events containing a single muon ($p_T(\mu) = 10\,\text{GeV}$) superimposed on a PU of 200, reconstructed using the tracking algorithms optimised for a flat barrel geometry. The mean number of tracks found and the tracking efficiency are provided.

	Flat geometry	**Tilted geometry**
# of tracks after HT	229	161
# of fakes after HT	92	35
# of tracks after full chain	55	48
# of fakes after full chain	9	4
Efficiency after full chain	97.3%	97.3%

results could improve further.

The availability of tracking information down to 2 GeV may be of use to the L1 trigger, and the impact of this potential requirement on the proposed track-finder system has been studied. Lowering the minimum p_{T} threshold from 3 GeV to 2 GeV requires modifying the GP parameters to ensure adequate duplication in φ, and modifying the HT configuration by increasing the number of columns along the q/p_{T} by 50% to take into account the increased p_{T} range, while preserving the precision of the estimate. This increase in the q/p_{T} range consequently increases the required FPGA resources by 50% and results in a larger output data rate from the HT by a factor of 2.2. In comparison to the track reconstruction efficiency for $p_{\mathrm{T}} > 3$ GeV, a loss of tracking efficiency is observed in the range $2 < p_{\mathrm{T}} < 2.7$ GeV, mainly due to multiple scattering where stubs do not always intersect within a single HT cell and therefore fail to exceed the threshold criteria and generate track candidates. To mitigate this problem, it is possible to reduce the precision of the HT along q/p_{T} and ϕ_T, for the range $2 < p_{\mathrm{T}} < 3$ GeV only, by a factor of two. This variable precision HT, which has been implemented in firmware separately, is able to recover some of the loss (increasing efficiency from 65% to 75% in the range of $2 < p_{\mathrm{T}} < 2.15$ GeV) in simulation. Further improvements to the HT, such as p_{T} dependent threshold criteria, should be able to regain any remaining losses if necessary. Optimisation of the KF to the new minimum p_{T} threshold could also yield improvements.

The performance and use of the bend filter cut applied within the HT has also been studied. The assumed bend resolution can be easily adjusted in firmware, via a single parameter. Alternatively the filter can be turned off entirely in the unlikely scenario that the stub bend information became unreliable. Disabling the bend filter would increase the rate of misreconstructed and duplicate track candidates produced by the HT, but does not lose efficiency, or cause truncation in the HT processing as shown in Table 10. The performance of the full system is similar to that described by Fig. 26.

Table 10: The mean number of tracks found and the tracking efficiency after the HT, with and without application of the bend filter, for tracks originating from the primary interaction in $\mathrm{t\bar{t}}$ events with 200 PU

	Bend filter	**No bend filter**
Tracking efficiency after Hough Transform	97.1	97.9
Track candidates after Hough Transform	331	1285

8 Future Developments and Improvements

One advantage of a fully time-multiplexed, all-FPGA approach is the inherent flexibility to adapt and evolve the algorithm choices and implementation. These improvements can come in two forms: changes that fit within the current technological boundaries, and changes that utilise and exploit newly emerging, or available, technologies. So far, the project has benefited greatly from the former, as the exact algorithm and implementation has evolved greatly over the past two years. We expect to soon benefit from the latter.

8.1 Improvements to the Hough Transform Algorithm

The radial offset parameter, T, was fixed at 58 cm in the demonstrator for the entire tracker solid angle. However, this choice is not optimal at high pseudorapidities, where particles will not traverse the full radial extent of the tracker. At these high $|\eta|$ values, a smaller value of T would be preferred, as it would spread the gradients of the stub lines in Hough space equally

between positive and negative values. Reducing the value of T down to 47 cm in the highest $|\eta|$ sectors reduces the rate of track-candidates produced by the HT in these sectors by a factor of two. This change would be trivial to implement in firmware, without additional latency or significant resource usage.

The effect of using a grid of hexagonal or diamond cells in the HT instead of a conventional, rectilinear grid has also been studied, and indicates that reduction in the rate of track candidates by $\sim$20% may be possible, whilst maintaining the track finding efficiency. It is believed that such designs would be straight-forward to implement in firmware.

8.2 Improvements to the Kalman Filter Algorithm

The Kalman Filter used in the demonstrator is in some respects a minimal version of the full Kalman Filter algorithm that could be applied to the task of track reconstruction. This served to simplify the implementation of the KF in the FPGA at the expense of some tracking performance, although it is expected that some of these extensions could be added without significant redesign of the existing firmware. The first of these is the addition of the transverse impact parameter to the fit which will help identify and improve fitting performance for particles which don't originate from the beam line, such as those from B hadron decay. The change requires a small adjustment to the parameterisation. A fifth parameter would require some additional resources, since the matrix dimensions increase with the number of parameters.

Accounting for the effect of multiple scattering has also been investigated. An additive contribution to the forward prediction of the state, and its covariance matrix, is required and could easily be accommodated with the existing design. Since the magnitude of multiple scattering depends on the amount of material traversed, and the particle momentum, some additional computation is required. The resource usage for this is small compared to the other matrix calculations (one additional BRAM per KF worker), and can be executed in parallel. In this scenario, a scattered track, as found by the KF (stepping in increasing r), would point towards its multiple scattering point, rather than its origin. The net result could be a more efficient track finder, but with a potentially reduced resolution. Solutions to this issue could be to modify the algorithm to step in decreasing r, or to include a smoothing stage, which would re-fit the track in decreasing r. A smoothing stage would take four additional Kalman iterations of latency, but no additional FPGA resources.

8.3 Move to the Ultrascale platform: from demonstrator to baseline system

The next step in the development of the demonstrator project is to transition from the Xilinx Virtex-7 FPGA to a modern Xilinx Ultrascale device, with transceivers capable of data transmission speeds of at least 16.3 Gb/s. A system architecture that makes use of this increased link bandwidth would allow the demonstrator to scale to a time-multiplexed factor of 18, our baseline design, as described in Section 4. In this design, each TFP board covers the same region of the detector as the demonstrator slice, for a total of 8 x 18 = 144 boards. The number of FPGAs per TFP board is flexible (typically 1-3) and will depend on the extent to which the firmware can be optimised, alongside a compromise between latency and cost, and the only requirement is that the FPGA supports the necessary input bandwidth of about 1 Tb/s. The additional features described in the previous section are expected to have a small impact on system resources, in comparison with the firmware optimisations described below.

8.3.1 FPGA Resources

With the baseline design requiring a time-multiplexed factor of 18, each TFP would have to process the same volume of data at twice the rate of the demonstrator slice described in this paper. Naively, therefore, one may set an upper bound on the amount of logic needed at twice the requirement of the demonstrator, in order to handle this doubling in processing bandwidth. As shown in Table 11, a doubling of the FPGA resources would require a minimum of three Kintex Ultrascale 115 (KU115) FPGAs per TFP.

On the other hand, the reduction in the number of candidates out of the HT when using the tilted barrel geometry should mean that a 30% reduction in the number of KF workers is feasible without incurring losses due to truncation. It was also noted in Section 7.6 that the demonstrator suffers from significant under-utilisation of processing resources, even in the high occupancy conditions of $t\bar{t}$ events with a PU of 200. As was shown in Section 7.3, the HT input is idling around 50% of the time; and the HT output, which was designed to handle the challenging, but infrequent case of a sub-sector containing a jet without efficiency losses due to truncation, is usually idling because no jet is present. In addition, even in the centre of very high p_T jets, the State Update block of the KF containing the majority of the processing logic of the algorithm is idling more than 75% of the time. It is expected that by optimising the dataflow throughout the design, taking into account the gains from adapting to the tilted geometry, it should be possible to halve the logic resources required while maintaining performance.

Similarly, while all fabric on the current demonstrator is clocked at 240 MHz, parts of the system have been tested at increased speeds. The GP router runs successfully at 480 MHz on the MP7, while the HT is capable of running at 300 MHz in the KU115 and results indicate that this will improve with continued optimisation. The architecture of the Kalman Filter is such that there are very few fan-outs and the design is heavily pipelined, lending itself towards operation at higher clock frequencies. Preliminary studies indicate that the Ultrascale and particularly Ultrascale+ FPGAs are much better adapted to running large-scale single clock domain designs across the device, and this can be taken advantage of. Therefore, by targeting a clock speed of 480 MHz for most parts of the system, it should be possible to double the processing bandwidth of the design and halve the resources required per TFP, with respect to Table 11.

Considering the combined savings from maximising the processing bandwidth, through both optimising the design to minimise under-utilisation and running the algorithms at higher frequencies, it is highly feasible that a final TFP could be constructed with no more than two Kintex Ultrascale 115 FPGAs, or one Virtex Ultrascale+ 11P FPGA.

8.3.2 Latency

In comparison to the current demonstrator, the latency of the track finding system would be reduced when scaled to the baseline design with 16.3 Gb/s links. The data accumulation periods in the KF and HT, where the FPGA must wait for all the data to arrive before it can continue

Table 11: The upper limit of FPGA resources required per TFP for the baseline system (with time-multiplexed factor n=18), inclusive of infrastructure logic. The available resources for a number of compatible Xilinx Ultrascale FPGAs are shown for comparison [24].

	LUTs [10^3]	**DSPs**	**FFs** [10^3]	**BRAM (36 Kb)**
Upper limit per TFP	1629	16944	1742	6570
Kintex Ultrascale 115	633	5520	1266	2160
Virtex Ultrascale+ 9P	1182	6840	2364	2160
Virtex Ultrascale+ 11P	1296	9216	2592	2016

processing, would be reduced by 450 ns each. A conservative estimate of the latency of the baseline system, from input of stub data at the DTC, to input of tracks at the Level-1 correlator, is given in Table 12. The overall latency of the unpacking, formatting and regional assignment steps in the DTC is conservatively estimated to be 250 ns. An additional latency of $<$150 ns is incurred for the transfer of data from DTC to TFP.

If the entire design could be run faster than the currently assumed 240 MHz, further latency savings could be targeted. If running at 480 MHz, a conservative estimate of the maximal latency (first stub in to first track in) has been placed at 2.2 μs.

Table 12: Latency table for the baseline system, extrapolated from the existing demonstrator. Note that extrapolation to 480 MHz assumes that additional registers are necessary, meaning that rather than a 50% reduction in latency, a 30% reduction is assumed throughout. This number is based on what was achieved with the GP router at 480 MHz. Two SERDES stages internal to the TFP are also assumed, to cover the worst case scenario (for latency) of three daisy-chained FPGAs.

Latency [ns]	**240 MHz**	**480 MHz**
DTC	250	250
DTC $\rightarrow$ TFP (SERDES & fibre)	150	150
GP	251	176
HT	575	403
KF	1220	854
DR	38	27
Two internal SERDES stages	240	240
TFP $\rightarrow$ L1 (SERDES & fibre)	150	150
TFP First out $\rightarrow$ Last Out	225	225
Total: First in DTC $\rightarrow$ First in L1	2784	2243
Total: First in DTC $\rightarrow$ Last in L1	3025	2468

9 Conclusions

A hardware demonstrator has been assembled in order to prove the feasibility of a track-finder at Level-1 for CMS at the High Luminosity LHC. The demonstrator implements a Hough Transform algorithm for coarsely identifying track candidates and a Kalman Filter to clean and fit them, on FPGA-based hardware, along with corresponding emulation software. The hardware slice has successfully shown that track finding and fitting for charged particles with transverse momentum exceeding 3 GeV is possible at 40 MHz, and within 4 μs, in the challenging high occupancy conditions of the HL-LHC. This has been accomplished using currently in-hand technology (MP7 processing boards), and one can expect the latency and projected scale of the system to be reduced as algorithms are optimised and refined, and new technology becomes available.

Acknowledgments

We are grateful to the CMS collaboration for use of their detector simulation software.

This research work was supported by the Science & Technology Facilities Council (STFC), and the EU FP7-PEOPLE-2012-ITN project nr. 317446, INFIERI, "Intelligent Fast Interconnected and Efficient Devices for Frontier Exploitation in Research and Industry". One postgraduate

student working on this project was supported by Maxeler Technologies, and the Worshipful Company of Scientific Instrument Makers. We gratefully thank all sponsors for their support.

References

[1] L. Evans and P. Bryant, "LHC Machine", *JINST* **3** (2008) S08001, doi:10.1088/1748-0221/3/08/S08001.

[2] G. Apollinari et al., "High-Luminosity Large Hadron Collider (HL-LHC): Preliminary Design Report", CERN-2015-005 (2015,) doi:10.5170/CERN-2015-005.

[3] CMS Collaboration, "The CMS experiment at the CERN LHC", *JINST* **3** (2008) S08004, doi:10.1088/1748-0221/3/08/S08004.

[4] CMS Collaboration, "Technical Proposal for the Phase-II Upgrade of the CMS Detector", Technical Report CERN-LHCC-2015-010, CMS-TDR-15-02, 2015.

[5] J. Jones, G. Hall, C. Foudas, and A. Rose, "A Pixel Detector for Level-1 Triggering at SLHC". CERN-2005-011.

[6] M. Pesaresi, "Development of a new Silicon Tracker for CMS at Super-LHC". PhD thesis, Imperial College, London, 2010,. CERN-THESIS-2010-083.

[7] M. Pesaresi and G. Hall, "Simulating the performance of a p_T tracking trigger for CMS", *JINST* **5** (2010) C08003, doi:10.1088/1748-0221/5/08/C08003.

[8] G. Hall, M. Raymond, and A. Rose, "2-D PT module concept for the SLHC CMS tracker", *JINST* **5** (2010) C07012, doi:10.1088/1748-0221/5/07/C07012.

[9] CMS Collaboration, "The Phase-2 Upgrade of the CMS Tracker Technical Design Report", Technical Report CERN-LHCC-2017-009, CMS-TDR-17-001, to be published, 2017.

[10] D. Abbaneo, "Performance Requirements for the Phase-2 Tracker Upgrades for ATLAS and CMS", *EPJ Web Conf.* **127** (2016) 00002, doi:10.1051/epjconf/201612700002.

[11] CERN, "LpGBT specification document". https://espace.cern.ch/GBT-Project/LpGBT/Specifications/LpGbtxSpecifications.pdf.

[12] G. Hall, "A time-multiplexed track-trigger for the CMS HL-LHC upgrade", *Nucl. Instrum. Meth. A* **824** (2016) 292–295, doi:10.1016/j.nima.2015.09.075.

[13] C. Amstutz et al., "An FPGA-Based Track Finder for the L1 Trigger of the CMS Experiment at the High Luminosity LHC", in *2016 IEEE-NPSS Real Time Conference (RT)*, pp. 1–9. 2016. doi:10.1109/RTC.2016.7543102.

[14] CMS Collaboration, "CMS Physics: Technical Design Report Volume 1: Detector Performance and Software", Technical Report CERN-LHCC-2006-001, CMS-TDR-8-1, 2006.

[15] CMS Collaboration, "CMS Technical Design Report for the Level-1 Trigger Upgrade", Technical Report CERN-LHCC-2013-011, CMS-TDR-12, 2013.

[16] Xilinx, "7 Series FPGAs Overview, Product Specification", May, 2015. DS180 (v1.17), http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf.

[17] P. V. C. Hough, "Method and means for recognizing complex patterns", December, 1962. US Patent 3,069,654.

[18] R. Frühwirth, "Application of Kalman filtering to track and vertex fitting", *Nucl. Instrum. Meth.* **A262** (1987) 444–450, doi:10.1016/0168-9002(87)90887-4.

[19] M. Technologies, "MaxCompiler". https://www.maxeler.com/media/documents/MaxelerWhitePaperMaxCompiler.pdf.

[20] PICMG, "Micro Telecommunications Computing Architecture Short Form Specification", 2006. https://www.picmg.org/wp-content/uploads/MicroTCA_Short_Form_Sept_2006.pdf.

[21] E. Hazen et al., "The AMC13XG: a new generation clock/timing/DAQ module for CMS MicroTCA", *JINST* **8** (2013) C12036, doi:10.1088/1748-0221/8/12/C12036.

[22] K. Compton et al., "The MP7 and CTP-6: multi-hundred Gbps processing boards for calorimeter trigger upgrades at CMS", *JINST* **7** (2012) C12024, doi:10.1088/1748-0221/7/12/C12024.

[23] C. G. Larrea et al., "IPbus: a flexible Ethernet-based control system for xTCA hardware", *JINST* **10** (2015) C02019, doi:10.1088/1748-0221/10/02/C02019.

[24] Xilinx, "UltraScale Architecture and Product Data Sheet: Overview", February, 2017. DS890 (v2.11), https://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf.