# A novel FPGA-based track reconstruction approach for the Level-1 trigger of the CMS experiment at CERN

R. Aggleton[†], L. Ardila-Perez[||], F. A. Ball[†], M. N. Balzer[||], J. Brooke[†], L. Calligaris[**], M. Caselle[||], D. Cieri[**], E. J. Clement[†], 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[¶], I. D. Reid[‡], A. W. Rose[¶], O. Sander[||], T. Schuh[||], C. Shepherd-Themistocleous[**], A. Shtipliyski[¶], S. P. Summers[¶], A. Tapper[¶], I. Tomalin[**], K. Uchida[¶], P. Vichoudis[§] and M. Weber[||]

[*]Austrian Academy of Science, [†]University of Bristol, [‡]Brunel University London, [§]CERN
[¶]Imperial College, [||]KIT- Karlsruhe Institute of Technology, [**]STFC - Rutherford Appleton Laboratory
Email: konstantinos.manolopoulos@stfc.ac.uk

***Abstract*—The Compact Muon Solenoid (CMS) experiment at CERN is scheduled for a major upgrade in the next decade in order to meet the demands of the new High Luminosity Large Hadron Collider. Amongst others, a new tracking system is under development including an outer tracker capable of rejecting low transverse momentum particles by looking at the coincidences of hits (stubs) in two closely spaced sensor layers in the same tracker module. Accepted stubs are transmitted off-detector for further processing at 40 MHz. In order to maintain under the increased luminosity the Level-1 trigger rate at 750 kHz, tracker data need to be included in the decision making process. For this purpose, a system architecture has to be developed that will be able to identify particles with transverse momentum above 3 GeV/c by building tracks out of stubs, while achieving an overall processing latency of maximum 4us. Targeting these requirements the current paper presents an FPGA-based track finding architecture that identifies track candidates in real-time and bases its functionality on a fully time-multiplexed approach. As a proof of concept, a hardware system has been assembled targeting the MP7 MicroTCA processing card that features a Xilinx Virtex-7 FPGA, demonstrating a realistic slice of the track finder. The paper discusses the algorithms' implementation and the efficient utilisation of the available FPGA resources, it outlines the system architecture, and presents some of the hardware demonstrator results.**

## I. Introduction

The Compact Muon Solenoid (CMS) [1] is a large-scale, general purpose particle detector at the Large Hadron Collider (LHC) [2] at CERN, designed to investigate a wide range of physics phenomena and to improve our understanding of the Standard Model. Currently in the LHC, proton bunches collide together with a rate of 40 million times per second, at centre of mass energy of 13 TeV. The High Luminosity LHC (HL-LHC) [3] is an upgrade of the current collider, scheduled to be completed by the year 2026. After this upgrade the instantaneous luminosity will be increased from $1\times10^{34}$ cm$^{-2}$ s$^{-1}$ to $5-7\times10^{34}$ cm$^{-2}$s$^{-1}$, increasing the average number of proton-proton collisions per bunch crossing (pile-up or PU) up to 140-200. Due to the increasing integrated and instantaneous luminosity an upgrade of the CMS detector is also necessary, in order to meet the demands of the new collider and to further improve the current level of performance. Part of this upgrade is the complete replacement of the CMS silicon tracker [4], [5]. The tracker is the innermost part of the detector and after 15 years of operation and exposure to radiation it is imperative to be replaced with a new model that can withstand the higher radiation levels expected by the HL-LHC.

The higher interaction rates of the HL-LHC will pose a significant strain to the CMS Level-1 trigger. The L1 trigger is an event selection system implemented in hardware that currently uses coarse grain information in order to accept events that are deemed as interesting for subsequent analysis and reject the remaining ones. Under the existing LHC conditions the L1 trigger reduces the data rate from 40 MHz down to 100 KHz, but with the higher interactions rate of the HL-LHC the L1 trigger is expected to reduce the rate down to 750 KHz, while processing a significantly larger volume of data and without suffering from any losses of potentially interesting events. The proposed solution is to include at L1 charged particle track information (L1 Track Trigger).

The proposed design for the outer tracker upgrade [6] is based on utilising two types of double sensor $p_T$ modules (called 2S and PS), capable of rejecting on detector hits generated by particles with low transverse momentum ($p_T$). This is achieved by looking at the correlation of hits in the two sensors of a single module. Correlations compatible with a high-$p_T$ (> 2-3 GeV) track are called stubs (Fig. 1).

The L1 track trigger, using these stubs as an input, will need to identify and fully reconstruct all possible tracks, that will be used in a following stage to discriminate the signal from background events. The track identification process consists of mainly three steps: data formatting, track reconstruction and track fitting, with the prerequisite that all three steps are

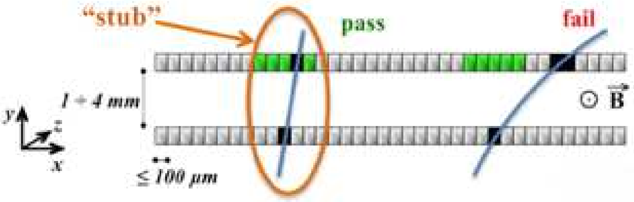

Fig. 1. Cluster matching in $p_T$-modules. Correlating closely spaced clusters between two  mm separated sensor layers allows discrimination of transverse momentum based on the particle bend in the CMS magnetic field. Only tracks with $p_T > 2$ - 3 GeV/c are transferred to the L1 trigger.

completed within an overall latency of 4 us. Aiming at meeting the above requirements and providing a feasible solution for the L1 track trigger, using current FPGA technology, this paper proposes an FPGA-based track finding solution that bases its functionality on a fully time-multiplexed architecture, while using a projective binning algorithm based on the Hough Transform to identify the track candidates. The fitting stage is performed by implementing a filter that identifies tracks consistent with a straight line in the *r*-*z* plane, followed by a linear regression technique that fits the track parameters using independent straight lines in both *r*-*z* and *r*-$\phi$ planes. Each step is implemented on a different MP7 processing card (or cards) [7] that features a Virtex-7 FPGA. These cards are interconnected in a daisy-chain fashion, and this chain is further referred to as the Track Finding Processor (TFP).

## II. Time-Multiplexed Track Finding Processor

The Track Finding Processor (TFP) described in this paper bases its functionality on a time-multiplexing approach. The entire tracker is divided into $\phi$ octants, where $\phi$ is the azimuth angle of the track. Data from each octant is read out by a separate group of Data, Trigger and Control (DTC) boards. The role of these boards is to calculate the global coordinates of each stub, duplicate stubs in overlap regions between the octants and send these stubs to the neighboring processing octants. Each octant is processed by *N* identical TFPs, that will reconstruct all tracks in that specific octant. Each TFP processes only one event in N. For our demonstrator system we choose a value of *N* = 36, based on the current electronics and on the available I/O links. Hence, all data of the 1st event will be sent to the first TFP, the 2nd event will de directed to the second TFP etc, the *N*th event will be processed by the last TFP and the Nth+1 event will again be directed to the 1st TFP and so on. In order to also parallelise the track finding process within each octant we apply a divide-and-conquer approach: each octant in $\phi$ is subdivided in two sub-sectors in $\phi$ and 18 in $\eta$, where $\eta$ is the pseudo-rapidity, and track-finding is performed in parallel in each sector.

## III. Track Finding Architecture

Fig. 2 shows the block diagram of our Track Finding Processor. The TFP consists of three steps: first, the Geometric Processor handles the data formatting and distribution, next, based on a Hough Transform implementation the track building takes place, and finally, the fine Track Fitting is performed. Each block in the fig.2 is implemented on a single MP7 board.

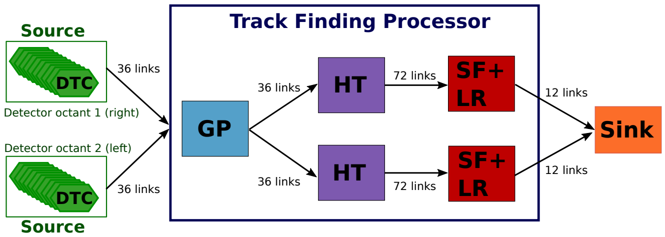

Fig. 2. The Track Finding Processor consists of the Source, Geometric Processor (GP), Hough Transform (HT), Seed Filter & Linear Regression (SF+LR) and the Sink.

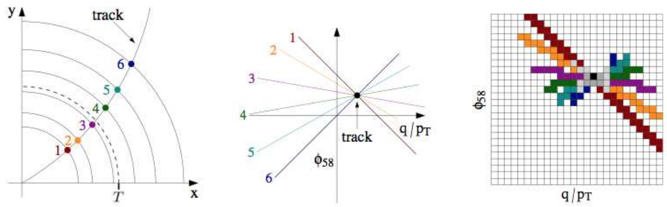

Fig. 3. Illustration of the Hough Transform. On the left-hand side is a sketch of one quarter of the tracker barrel in $r - \phi$. The track of a single particle is drawn and the stubs from six detector layers are shown as dots. In the middle is the track parameter plane where the six corresponding Hough-transformed stubs are drawn as lines and their intersection identifies the track and its parameters. On the right-hand side is the track finding histogram in the Hough transformed space.

### A. Geometric Processor

The Geometric Processor, converts the DTC stub coordinates into the extended format that will be used by the TFP and assigns each input stub to sub-sectors. Stubs which are compatible with two or more sub-sectors (usually due to the curvature of tracks in $\phi$) are then duplicated. A routing block realised as a three-stage, highly pipelined mesh is responsible for assigning the stubs to the proper output. The first routing stage implements a coarse sorting of the stubs based on their $\eta$ values, followed by a fine $\eta$ sorting in the second stage, and the final sorting in $\phi$ in the last stage. The router has been designed to be highly reconfigurable and can be easily adapted to any alternative sector definitions.

### B. Hough Transform

The Hough Transform (HT) algorithm is a method widely used to detect lines, circles or other parametric curves in image processing [8]. It can easily be applied to our case in order to identify tracks from stubs. We use the HT to identify charged particles with transverse momentum ($p_T$) greater than 3 GeV/c in the *r*-$\phi$ plane. The trajectory of these particles can be described as a linear equation between the initial azimuthal angle of the track $\phi_0$ and its $p_T$. Therefore, every stub position can be described with a straight line in the ($q/p_T$, $\phi_0$) parameter space. According to the Hough transform algorithm the point where several of these straight lines intersect describes a circle in the r-$\phi$ plane. Fig. 3 depicts the procedure of identifying a track produced by six stubs. A circle is considered as identified, only if it is comprised by at least one stub per layer and with hits in at least five layers out of the six available.

In order to implement the HT algorithm we bin the track parameter space into 1024 rows in $\phi$ and 32 columns in $q/p_T$. Based on simulation results this is the most efficient way to ensure a sufficiently good granularity and precise track candidate identification, while taking into account possible misalignments due to the hit resolution of the tracker modules. The method for finding tracks with the HT is the following: i) For each stub position in the ($r$, $\phi$) we calculate the corresponding straight line using the equation described in [9]. ii) A bend measurement embedded in stub data that corresponds to the distance between hits of the upper and lower sensor modules, is used as a rough estimate of the particle's $p_T$. Based on this, each stub is binned in a specific subset of columns in the track parameter array, that are consistent only with this bend value. iii) Create a histogram that counts the number of stubs in different tracker layers that cross its cells in the track parameter space. Identify valid track candidates that correspond to cells of the track parameter array containing stubs from at least five different detector layers. A detailed description of the Hough Transform architecture can be found in [9].

### C. Track Fitting

In the last stage of the TFP we perform the fine track fitting that can be split into two steps: during the first step we ensure that the stubs in a HT cell are compatible with a straight line in the $r$-$z$ plane (Seed Filter or SF) and then we fit helix parameters using independent straight line fits in the two planes, $r$-$z$/$r$-$\phi$, with a linear regression technique (Linear Regression or LR).

*1) Seed Filter:* The Seed Filter is implemented right after the r-$\phi$ Hough Transform. The task of the SF is to reduce the number of fake tracks that are produced by the HT, by exploiting the third coordinate ($z$) in order to filter out the fake tracks. Moreover, the algorithm removes spurious stubs from the tracks which are not consistent in the $r$-$z$ plane. The fundamental idea of the Seed Filter is to check the compatibility of the stubs in a cell with a line drawn from two seeding stubs, where a seeding stub is a stub that originates from a PS module in one of the innermost layers of the tracker detector. The algorithm collects all the possible pairs of seeding stubs, drawing a line passing through these stubs (seed). Only seeds compatible with a track originated from the beam spot and lying in the current pseudo-rapidity sector under investigation are kept. Surviving seeds are extrapolated to all other tracker layers, rejecting all stubs that are not compatible with the computed line. Only the closest stub to the line is kept for each layer. If there are less than 4 stubs on the seeding line, the track will be discarded.

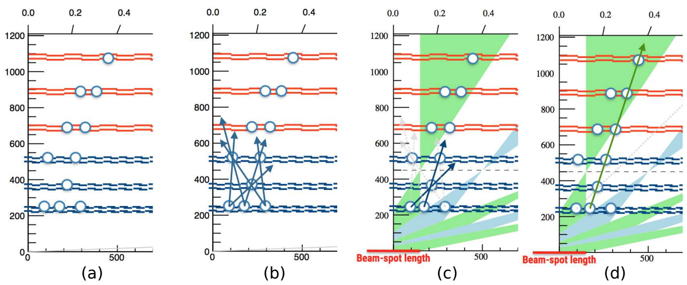

Fig. 4. Illustration of the 4 steps of the Seed Filter Algorithm. PS modules are depicted in blue colour and the 2S modules in red.

Fig. 4 shows the steps of the Seed Filter algorithm, which can be described as follows: i) First, the SF forms pairs of stubs which belong to different PS layers (Fig. 4b) and then it computes the lines that pass through these stubs (seeds). ii) Next, it discards the seeds that correspond to tracks that lie outside of the beam-spot and sector definition (Fig. 4c). iii) Following that, it extrapolates the surviving seeds to other tracker layers, rejecting stubs that are not compatible with the line (Fig. 4d). If there are multiple compatible stubs in a layer, then the algorithm keeps only the stubs that are closest to the seeding line. iv) Finally, only those tracks that still contain stubs in at least four layers are kept. If however, more than one seed satisfies the aforementioned condition, then the seed with the most layers is kept.

*2) Linear Regression:* The output of the Seed Filter is processed by the Linear Regression algorithm. Taking into consideration the fact that tracks with sufficient $p_T$ should form a straight line on both $r$-$z$ and $r$-$\phi$ planes, the LR algorithm performs two independent fits in the two planes in order to calculate the helix parameters that describe the track.

The algorithm is divided in four steps: i) Helix parameters are computed only in the $r$-$\phi$ plane using the full set of stubs. ii) For each stub the residual is calculated. Stubs with the largest residual are removed from the tracks, keeping only four stubs in total, with at least two belonging to PS modules. iii) Helix parameters are computed again using the full set on the $r$-$\phi$ plane, and only the PS stubs on $r$-$z$. iv) Finally, the $\chi^2$ is calculation takes place to reject the bad tracks.

*3) Track Fitting Architecture:* In each Track Fitting board 24 SF+LR modules are implemented. Stubs from each HT output channel enter the TF board and are stored in separate FIFOs. Six identical Control Units, operating in parallel, distribute HT track candidates from a set of six FIFOs to a set of four SF+LR modules in a Round-Robin fashion.

Each SF+LR module receives in input one stub per clock cycle. The filter processing has been implemented by means of two state machines. The first one (SM0) is responsible for the communication with the control unit, declaring if the SF+LR block is ready to accept a new track or not. In this first stage, stubs are sent to 15 Seed Finders blocks, which perform the identification of the pair of seeding stubs (seeds) in parallel. Seeds are stored in a Seed FIFO and read-out in the second state machine. Here, utilising a DSP chain, the consistency of the seeds with the beam-spot length and the $\eta$ sector definition is verified. Once all the seeds have been read-out from the

FIFO, SM0 marks the SF+LR module as free to receive new data. Meanwhile seeds that have passed the DSP check are sent to a different Seed Checker module, for a maximum of 10 good seeds. Each Seed Checker contains a copy of the input stubs and verifies the compatibility of each stub with the analysed seed. Checkers with stubs in enough layers send the stub addresses to a comparator module, in order to keep the stubs relative to the seed with the most layers. Finally, these stubs are used by the Linear Regression Module to compute the helix parameters.

## IV. Implementation Analysis and Results

The presented track finding system has been designed and implemented in VHDL. A set of five MP7 boards [7] has been utilised to accommodate the entire Track Finding Processor, operating at 240 MHz. Each board features a Xilinx Virtex-7 XC7VX690T FPGA, and 12 Avago Technologies MiniPOD optical transmitters/receivers, each providing 12 optical links able to operate up to 10.3 Gbps, thus providing a total optical bandwidth of 0.74 Tbps in each direction.

Apart from the five MP7-XE boards that form the Track Finding Processor three more boards are used to complete the testing chain. The first two, named *Source*, contain large buffers that can store up to 30 events of stub data for a single detector octant. The Source boards represent data from a set of up to 72 DTCs. The stub data from the DTCs are injected into the large buffers of the Source boards via IPBus [10]. Each source provides a stream of data to the downstream board on 36 links, equivalent to the DTCs that make up adjacent detector octants. Input data from two adjacent octants are required, to be able to handle tracks that traverse the regional boundary. The last board is the *Sink* and it is used to buffer the final output of the TFP, before being read-out via IPBus.

Table I depicts the resources utilization for the different firmware blocks. The latency for each block is the following: GP=310 ns, HT=1025 ns, SF+LR=1400 ns and Infrastructure Latency=545 ns. Hence, the total latency of the system is 3280 ns. These measurements have been made for each block independently, but also for the full chain, and they include optical link traversal time and serialisation/de-serialisation (SERDES). The entire system maintains a fixed latency, independently of the type or the number of event it has to process.

The design functionality has been verified through simulation analysis and then extensive data taking tests. The latter involved running through hardware simulated physics events up to a pileup of 200 interactions per bunch crossing. The software framework that was used allows for simulated physics samples that were generated in the official CMS Software (CMSSW) to be converted into text files which are then injected into the hardware chain via IPBus. The output of the hardware is then converted back into a CMSSW format, and is compared with the results of emulation software running on the same simulated physics event. Hence, it is possible to compare the results of the hardware and the emulation software, validating any simulation results of track finder performance. Excellent matching between hardware and software of 99.32% has been measured, at a pileup of 0, 140 and 200.

TABLE I
Resource Utilization of the TFP

| | LUTS (K) | FF (K) | BRAM (36Kb) | DSP |
|---|---|---|---|---|
| GP | 121 | 205 | 222 | 1056 |
| HT | 244 | 299 | 1776 | 2304 |
| SF+LR | 7800 | 277 | 17 | 110 |

## V. Conclusion

The current paper presented a track finding architecture for Phase-2 L1 trigger upgrade of the CMS detector at CERN. The design follows a time-multiplexing approach, where track candidates are identified and reconstructed using the Hough Transform algorithm and the track fitting is performed by combining a Seed Filter with a linear regression technique. The presented system is implemented on a chain of MP7-XE FPGA boards and has been validated in both simulation and hardware. The efficient algorithms implementation provides to the system an increased flexibility, allowing us to easily adapt to any changes of the detector's specifications.

## Acknowledgment

This work was supported in part by the UK Science and Technology Facilities Council. We gratefully acknowledge their support. The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Unions Seventh Frame- work Programme FP7/2007-2013/ under REA grant agreement nr. 317446 INFIERI INtelligent Fast Interconnected and Efficient Devices for Frontier Exploitation in Research and Industry.

## References

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

[2] THE CERN LARGE HADRON COLLIDER: ACCELERATOR AND EXPERIMENTS, *LHC Machine*, JINST 3 (2008) S08001, doi: 10.1088/1748-0221/3/08/S08001.

[3] THE CERN LARGE HADRON COLLIDER: ACCELERATOR AND EXPERIMENTS CollaborationG. Apollinari, et al., *High-Luminosity Large Hadron Collider (HL-LHC): Preliminary Design Report*, CERN, Geneva, 2015, doi: 10.5170/CERN-2015-005.

[4] CMS Collaboration, *Technical Proposal for the Phase-II Upgrade of the CMS Detector*, Technical Report CERN-LHCC-2015-010. LHCC-P-008. CMS-TDR-15-02, Geneva, Jun, 2015.

[5] M. Pesaresi, *Development of a new Silicon Tracker at CMS for Super-LHC*, PhD thesis, Imperial College London, 2010.

[6] S. Mersi, CMS Collaboration, *Phase-2 Upgrade of the CMS Tracker*, Nuclear and Particle Physics Proc., Vol. 273275, pp. 1034-1041, 2016.

[7] 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.

[8] P. V. C. Hough, *Method and means for recognizing complex patterns*, December 18th 1962 US Patent 3,069,654.

[9] C. Amstutz et al., *An FPGA-based track finder for the L1 trigger of the CMS experiment at the high luminosity LHC*, Real Time Conference (RT), IEEE-NPSS, June, 2016.

[10] 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.