L1 Track Finding in Barrel+Endcap Geometry with Tracklets

Anders Ryd (*Anders.Ryd@cornell.edu*) (contact for Cornell)
Peter Wittich (*pw94@cornell.edu*)
Charles Strohman (*crs5@cornell.edu*)
Jorge Chaves (*jec429@cornell.edu*)
Louise Skinnari (*louise.skinnari@cern.ch*)
December 23, 2013

Abstract

This proposal describes a proposed implementation of the L1 tracking using FPGAs. The pattern recognition is done using seeding based on two stubs in neighboring layers. The seeds, tracklets, at then projected to other layers where hits are added in a simple road search. A linearized χ^2 fit is performed to improved the track parameters from the seed.

1 Advantages with the FPGA based L1 tracking

FPGAs has a long track record for use in L1 trigger applications. The FPGA based road search algorithm described here provides several advantages. It makes use of commercially available components, FPGAs. This means that we do not need to develop specialized ASICs and we can take advantage of the rapidly improving performance of commercial FPGAs [1]. The implementation of the tracking algorithm in FPGAs naturally allows for processing the data in a pipeline. This means that we do not need to heavily time multiplex the processing which leads a much more complicated and expensive system to build. Of course, if a fully pipelined implementation is not possible due to the resources available in the FPGAs, time multiplexing can be used to provide more processing resources for an event. The proposed implementation with tracklets minimizes the data duplication. Tracklets that project to other sectors in the detector will be sent there instead of duplicating the hit data. Based on resource estimates the top-of-the-line FPGAs available today has sufficient DSPs and IO resources to implement the L1 tracking. The main challenge is to implement the tracking algorithm in firmware.

2 FPGA based L1 tracking

This proposal describes the use of FPGAs for the L1 tracking. The proposed implementation is based on a road search algorithm where track seeds are formed from pairs of stubs in neighboring layers. The performance of FPGAs has improved drastically over the last decade and continue to improve rapidly [1]. All aspects of FPGA performance has improved, the

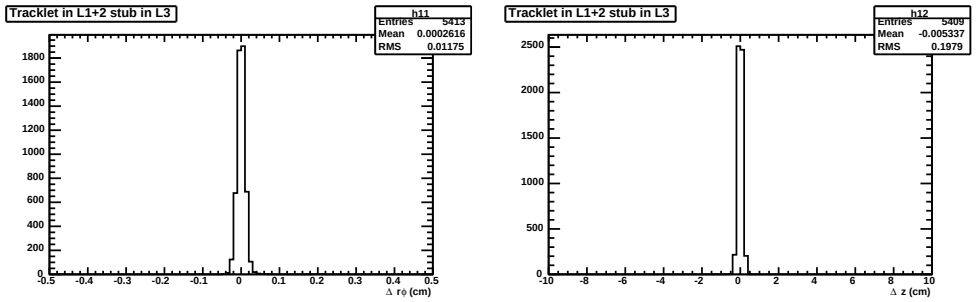


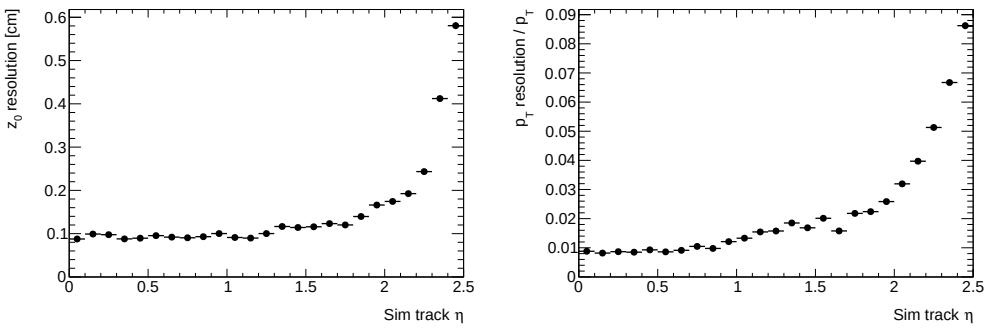
Figure 1: Seeds found in barrel layers 1 and 2 are propagated to layer 3 and the residuals of the projected trajectory with respect to the hits are shown. The sample used is single muons with a $p_{\mathrm{T}} = 10$ GeV.

number of logic gates, the number of DSPs, the number of memory blocks, and the I/O performance. Today the most powerful Xilinx FPGA, the Virtex-7 XC7VX1140T, has 3,360 DSP slices, 3,760 18-kbit memory blocks, 1,139,200 logic blocks, and 96 13.6 Gbits/s serial transceivers. As will be argued in this proposal these resources are sufficient for us to build a system that allows us to find L1 tracks within a 2 μs latency for each 25 ns bunch crossing. By using FPGAs we can take advantage of the rapid evolution of the commercially available hardware.

The tracklet-based approach is a road-based pattern recognition algorithm where pairs of stubs in neighboring layers, combined with a vertex constraint at the IP, are used to form seeds. The seeds are then projected to the other layers and matching stubs are added to form L1 tracks. The algorithm for adding stubs to the track is simple; if the stub is within a given distance from the projected tracklet the stub is accepted. The size of the matching window is a function of the layer, or disk, that the stub is on and which layers were used to form the stubs. (It should be investigated whether the window size should also depend on the track p_{T}.) If there are more than one stub in the matching window, the stub closest to the track projection is accepted. The tracklet and the matched stubs are then passed to a track fit. This track fit is a linearized χ^2 fit around the seed track. This fit works very well since the track parameters of the seed are already a very good approximation of the track. The input to the track fit is the residuals of the stubs with respect to the seed track. These residuals are calculated during the track projection step such that the track fit is simply a few multiplications and additions for each track parameter.

The algorithm described above has been implemented as a simulation. The code for this runs both in CMSSW (and has been used as the basis for most track trigger studies by the CMS Track Trigger Integration group) and as a standalone tool. These simulation studies demonstrate how well this rather simple algorithm works. To illustrate this we have included a few example performance plots. In Fig. 1, examples of the residuals when tracklets are projected to other layers are shown. The track parameter resolutions are shown in Fig. 2.

Simulations are also used to study data volumes and combinatorics. The number of stubs



(a) (b)

Figure 2: Track z_0 (left) and P_{T} (right) resolution as function of sim track η.

and tracklets found by the algorithm strongly affects how difficult it will be to implement the tracklet-based track finding.

The existing simulation of the track finding algorithm in the Barrel+Endcap geometry is not a bitwise emulation, as it would be implemented in hardware, but makes use of floating-point calculations. However, as was previously done for the Long Barrel geometry [2], an integer-based calculation has been determined and we will implement this in C++ during early 2014 to verify that the approximations used are sufficiently precise.

When implementing the track-finding algorithm in hardware, we need to break down the problem by dividing the detector into some form of sectors that can be handled in a processing unit. For the tracklet-based approach, we have focused on a division of the detector into ϕ sectors. The optimal division will be clear when we understand the resources needed in the FPGAs to perform the track finding. However, if we want to find tracks with $p_{\mathrm{T}} > 2$ GeV and only require communication between neighboring ϕ sectors, the number of sectors is constrained to about 28. For illustration we are here assuming 16 sectors. Furthermore, we divide the sectors into positive and negative η, resulting in a total of 32 half-sectors. We will explore the resources required to do the track finding in one of these half-sectors. The data required in a given sector will be sorted by the FEDs that receive the data from the front end and subsequently pass the stubs to the track finding.

For this division scheme we use the simulation to estimate the data volume into each sector, the number of stubs per layer, the number of tracklets found, and the number of tracklets projecting to a neighboring sector . This is illustrated in Fig. 3.

As discussed in the section on system architecture the I/O requirements for the tracklet based approach handled with the FPGAs available today. We have also estimated the required DSP resources. Based on the number of formed tracklets and produced tracks, we estimate that we have a factor of 6 more DSP resources in the virtex-7 FPGA than what is required to carry out the computations. Other resources such as memory blocks for implementing FIFOs and logic blocks for building state machines to process the data are much harder to estimate and the R&D challenge in this proposal is to develop the firmware

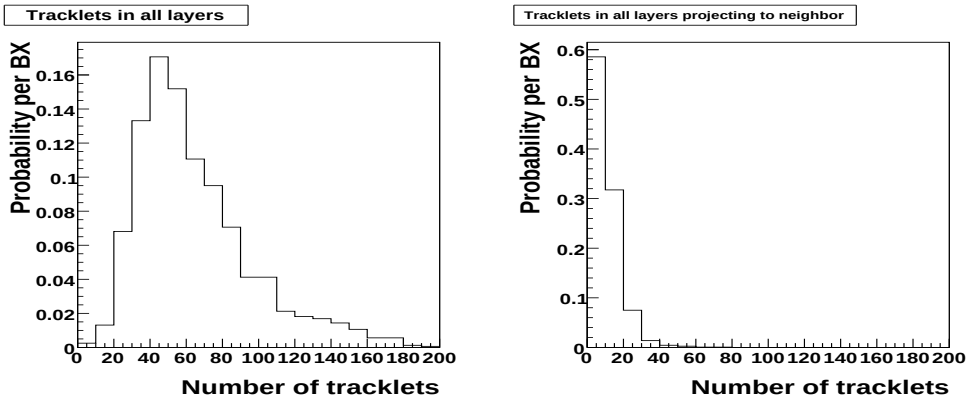


Figure 3: (left) The number of tracklets found in one half sector. (right) the number of tracklets in a sector that propagates to a neighboring sector. The sample is top-pair production with $\langle PU \rangle = 140$.

to understand if these resources are sufficient to implement the L1 tracking.

During summer 2012 we implemented an integer-based simulation of the tracklet-based track finding for the Long Barrel geometry. We showed that the calculations allowed finding tracks with similar resolution as the floating-point simulation we had studied earlier. In early 2013 we started implementing this algorithm in FPGA firmware. This work has been done by an engineer (Charles Strohman) and a graduate student (Jorge Chaves) at Cornell. What we have accomplished is to implement several of the steps in the track finding, in particular we have developed a state machine to perform the combinatorics in the tracklet finding step, the calculation of the track parameters for the tracklet, and projection of the track to different layers. Work is currently underway to understand how to best do the matching of the projected tracklet to stubs in a given layer. This a question of how to most effectively organize the data. So far we have not encountered any major surprises. Our estimates of the number of processing steps in the FPGA has been accurate; however it is to early to draw any conclusions about the overall resource usage.

2.1 System requirements

The system requirements for the implementation of the L1 track finding in FPGAs is a bit different than for the asociative memories. The overall data volume is smaller as only about 10% of the stubs data is duplicated. However, tracklets needs to be sent to neighboring sectors in order to do the hit matching. Alos tracks needs to be sent to the neighboring sector for the duplicate removal.

With today's available technology we can handle the data volumes in and out of a sector by one FPGA, the Virtex-7 XC7VX1140T, which has 96 13.6 Gbit/s links. We can allocate these as shown in Table 1.

Table 1: Allocation of data links

Task	Number of links	Data rate (Gbits/s)
Receive stubs	48	652.8
Send and receive tracklets to left neighbor	8	108.8
Send and receive tracklets to right neighbor	8	108.8
Send and receive stub matches to left neighbor	4	54.4
Send and receive stub matches to right neighbor	4	54.4
Send tracks to left neighbor for duplicate removal	4	54.4
Receive tracks from right neighbor for duplicate removal	4	54.4
Send final list of tracks	8	108.8
Total	88	

2.2 Goals and milestones

We propose the following milestones in 2014 for the development of the tracklet-based L1 tracking:

1. Complete the steps up to and including the track fit for the Long Barrel geometry. What remains here is to complete the stub matching and to perform the track fit. We estimate that this should be completed by March 1, 2014.

2. In parallel with completing the previous task, we will develop a detailed integer-based emulation of the algorithm in the Barrel+Endcap geometry. Same time line as the previous task.

3. Once the first two tasks are completed we will start the FPGA implementation of track finding in the Barrel+Endcap geometry. We expect to have the different building blocks ported to the new geometry by July 1. Most of the 'structure' from the Long Barrel implementation should still apply, but there are differences in the details of how calculations are done, and how the data is divided between different processing units.

4. In the second half of the year we will optimize the resource usage and the performance of the algorithm. This will be done on the test stand.

2.3 Facilities, equipment, and other resources

At the CLASSE, Cornell Laboratory for Accelerator based ScienceS and Eduction, we have access to electronics engineering and facilities for testing and designing electronics. In particular we have licences for xilinx development tools.

2.4 Request for 2014

To carry out this work we are asking for:

1. Three months of engineering time to develop firmware ($69,369).
2. Funds to establish a test stand at Cornell ($20,000).
3. Travel for engineer to Fermilab ($2,000).

For 2015 we assume this work will continue, and request funding at the same level.

References

[1] W. Smith, "Upgrade of the LHC Experiment Online Systems", presented at CHEP 2012, New York City, http://indico.cern.ch/getFile.py/access?contribId=585&sessionId=0&resId=0&materialId=slides&confI

[2] A. Ryd, "L1 data processing and track finding in Long-Barrel geometry",presented at CMS Tracker Review of backend electronics review, https://indico.cern.ch/materialDisplay.py?contribId=6&sessionId=1&materialId=slides&confId=24506