Design for a L1 Tracking Trigger for CMS

J Hoff,[a] M. Johnson,[a,*] R. Lipton,[a] G. Magazzu,[b] N. Pozzobon,[c] A. Ryd,[d] and E. Salvati[d]

[a] *Fermi National Accelerator Laboratory[†], Batavia, Il., USA 60510*
[b] *University of California, Santa Barbara, CA USA*
[c] *Università degli Studi di Padova and INFN - Sezione di Padova, Padova, Italy*
[d] *Cornell University, Ithaca, NY USA 14853*
E-mail: mjohnson@fnal.gov

ABSTRACT: We present an electronics design for a tracking trigger for the CERN SLHC. The on detector part uses asynchronous logic so the only clock required is the LHC crossing clock. High Pt tracks are identified with a hierarchical method that finds track stubs using closely spaced pairs of detectors. Track segments (called tracklets) are then formed from pairs of stubs that are separated by 40 mm. This separation is close enough so that matching stubs is relatively easy but far enough apart so that the tracklets can be projected to another layer with mm accuracy. Matching segments in two or more layers then define a track.

KEYWORDS: Triggers; mousetrap; pipeline; asynchronous.

[*] Corresponding author.
[†] Operated by Fermi Research Alliance, LLC under Contract No. De-AC02-07CH11359 with the United States Department of Energy.

Contents

1. Level 1 Trigger Design

Many previous designs for tracking triggers [1,2] have employed content addressable memories (CAMs) to identify hit patterns that correspond to tracks of interest. This approach is very flexible but for large silicon based trackers such as the one for CMS, the number of trigger patterns is very large and the amount of information that must be transferred from the detector to the pattern look-up unit is also large. These issues combined with the limited time available for a Level 1 decision make a design using this approach extremely challenging. Another approach [3,4] is to adopt a process that is similar to many track reconstruction programs where one starts with a seed track and then extrapolates the track to the next set of points and checks that the points all come from the same track. This process is repeated until all the tracking layers have been processed. This approach minimizes the amount of data that needs to be moved from one place to another and, if the detector is designed for this task, greatly simplifies the information processing.

One important difference in this design from a reconstruction program is that it uses the detector design to easily identify hits that come from tracks with Pt greater than about 2 GeV/c. This allows the Level 1 processor to ignore all the data from low Pt tracks and to concentrate the processing on tracks of interest.

Identifying points from high Pt tracks is done by using two silicon sensors spaced about 1 mm apart (called a stack, see Fig. 1). This spacing depends on the radius of the detector element, the pitch of the silicon strips and the detector magnetic field. The prototype chip design assumes that tracks of interest are displaced by at most by 4 strips. A readout chip (ROC) is located between the two layers and directly connected to the strips in both sensors. This allows hits from both sensors to be read out in parallel and it is then simple combinatorial logic to find pairs of hits (called stubs) that are displaced by less than 4 strips.

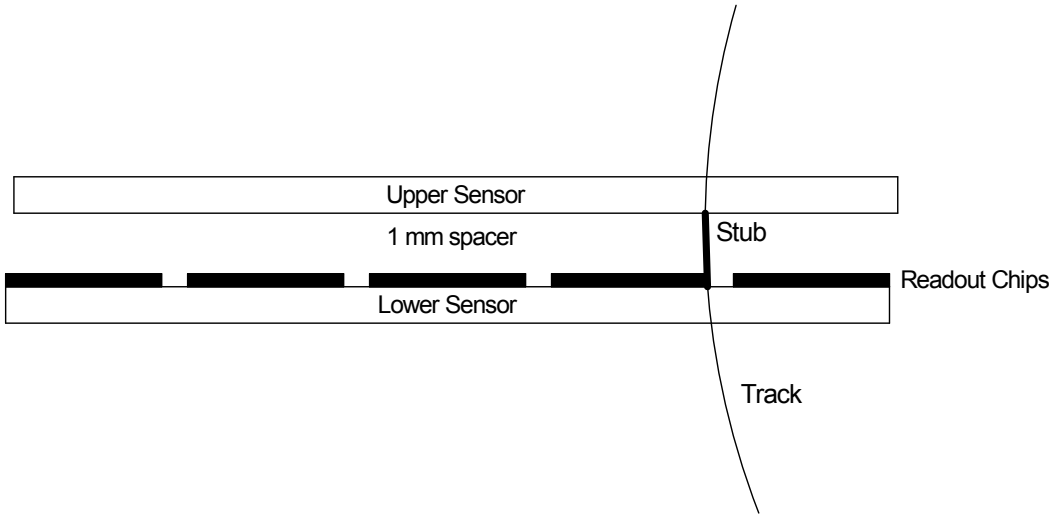

Figure 1: Data from the upper and lower sensors are read into the same chip. A track stub is found if the ϕ offset between the two sensors is less than 4 strips. The sensors are 100 mm by 100 mm. Each sensor has 1000 rows of strips spaced by 100 μm. Each row in the upper (slave) sensor has ten 10 mm long strips. Each row in the lower (master) sensor has eighty 1.25 mm long strips. The strips are arranged so that 1 upper sensor strip covers 8 lower sensor strips. There are 25 chips in a stack arranged in a 5 by 5 array so each chip services 200 rows of 2 long and 16 short strips.

Stubs provide a useful first level filter but they do not provide the measurement accuracy to predict track locations on other layers. In order to overcome this limitation two stacks are placed about 40 mm apart. The inner layer of each stack has 1.25 mm long strips (the outer layer uses 10 mm strips) in order to increase the tracking accuracy in the beam (z) direction This is used to check that the z positions of a potential track point back to the beam-beam interaction region and to project the track position in another layer. These features allow a simple combinatorial search for hit matches. The combination of two matching stubs forms a tracklet. Tracklet information combined with the beam interaction region is sufficient to project the tracklet position to all other layers of the tracker. Fig. 2 shows a cross section of one sector of the detector and Fig. 3 shows the results of a Monte Carlo study that projects tracklets found in the inner layer to the middle layer. The accuracy is less than 0. 1 mm in ϕ and less than 5 mm in

z.

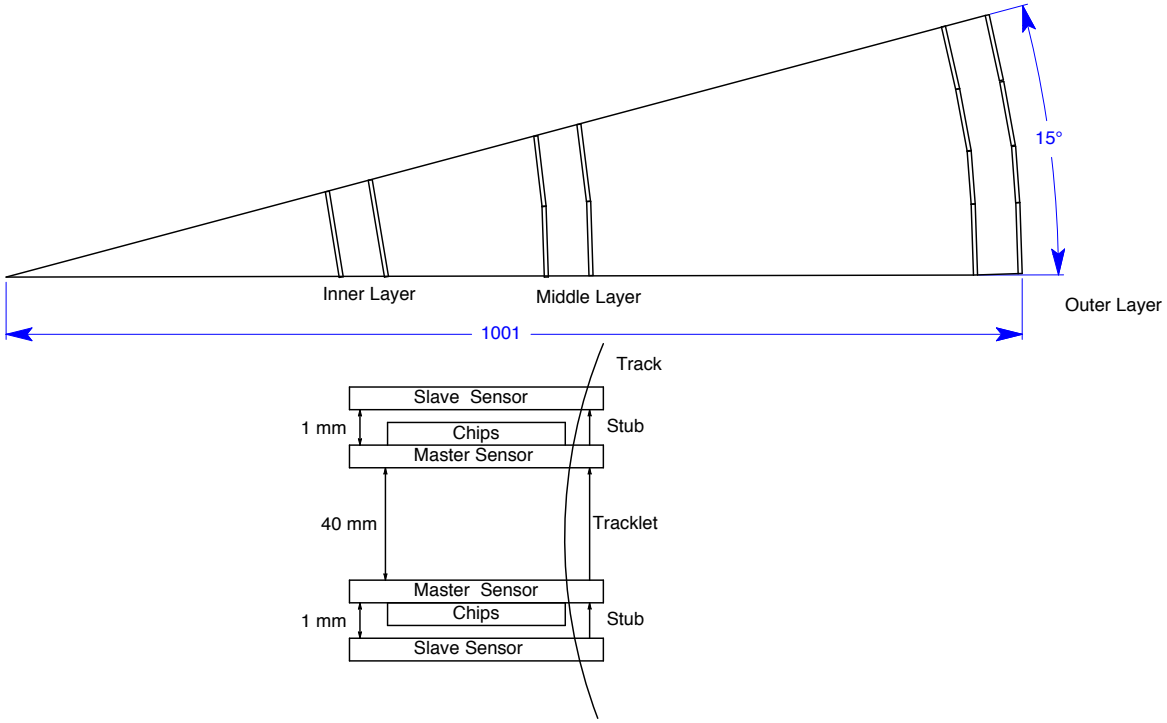

Figure 2: Cross section of a 15-degree sector of the tracker showing 3 layers. The lower part of the figure shows a detailed cross section view of one of the 3 layers. There are 2 stacks of 2 sensors each. The readout chip is located between the two sensors of a stack. A stub is the portion of a track found in a stack of sensors. A tracklet is formed from 2 stubs in the same layer. The z coordinate is normal to the page and ϕ is the polar coordinate in the plane of the page. Sensors are mounted on an approximately 40 mm square carbon fiber support structure called a rod. One rod extends over either the +z or –z half of the tracking detector.

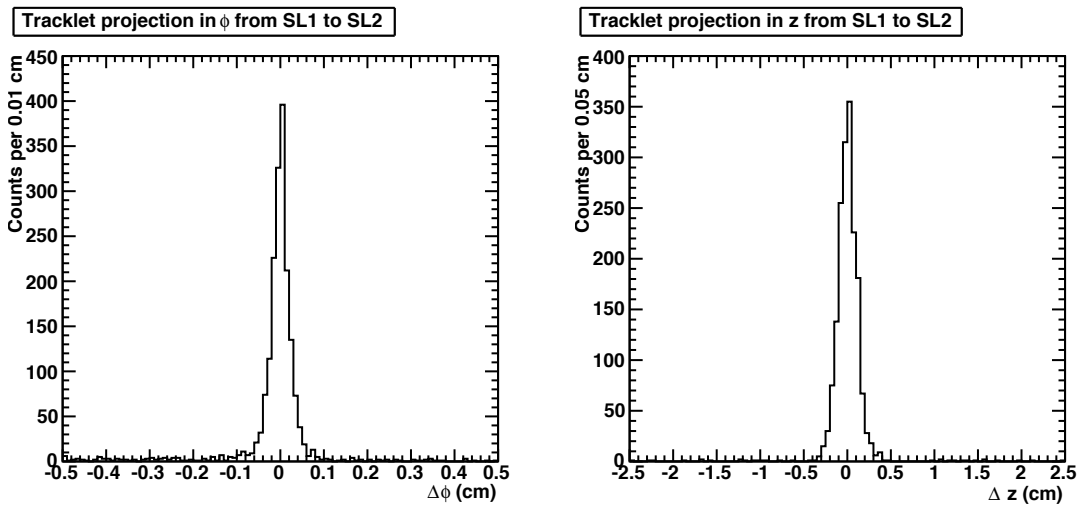

Fig. 3. Error resulting from the projection of a track from the inner layer to the middle layer of the detector shown in Fig. 2. A Gaussian fit gives a standard deviation in ϕ of 0.21 mm and 1.1 mm in z.

.

This design uses three layers and it projects tracklets from a given layer to the other two layers. This requires a final stage to remove duplicates but it allows a very simple algorithm for matching tracklets into tracks and it provides redundancy in case of hardware failures.

Tracks are found in a given layer by comparing the list of received tracklets to the list that the receiving layer sent to the other layers. For example, a tracklet found in layer 2 projects to a small region in layer 3. Layer 2 sends its tracklet to the FPGA that handles this region in layer 3. Upon receiving the tracklet from layer 2, the FPGA in layer 3 looks to see if it has a tracklet whose projection to layer 2 matches the position in layer 2 of the received tracklet. If the two tracklets match, it publishes a track. If the receiving layer does not have a tracklet but does have a stub it checks that the stub is within the allowed window for the tracklet and that the sign and Pt of the stub match the tracklet. The minimum requirements to identify a track are one tracklet and one stub in two of the three layers but more stringent requirements may be needed to reduce the accidental rate.

2. Readout Chip Design

Data processing is broken up into a series of pipeline steps. Table I lists the basic steps in acquiring stub data and transmitting it to the off-detector system. Some details such as averaging stub data from two crossings have been omitted. The following sections describe design features in more detail.

Table I Pipeline Steps for Readout Chip

25 ns Step	Operation
1	Discriminate input data
2	Load edge strip data into ϕ and z pipelines
3	Transmit data between chips in the ϕ and z directions.
4	Form clusters of 3 or fewer strips. Larger clusters are ignored
5	Find stubs in each short z section
6	Encode up to 4 stubs in each short z sections
7	Select up to 8 stubs from the 16 short z sections on the chip
8	Load output pipeline
9	Transmit data in ϕ direction to edge chip
10	Transmit data in z direction to optical transceiver
11	Send data over the optical transceiver

2.1 Data Flow

The sensor must be nearly covered with chips in order to read out the 1.25 mm long strips so finding clusters and stubs seamlessly across the sensor requires sending edge hits between chips in both the ϕ and z direction. This means that there is hit data transfer between neighboring chips as well as stub data transfer to the off-detector processor. This is schematically illustrated in Fig. 4 which shows a 2 by 2 chip array. Hit data is transferred in both the ϕ and z directions. Stub data is transferred off-detector by a two step process. It is first transferred to the end of a row in the ϕ direction and then along the right hand column in the z direction.

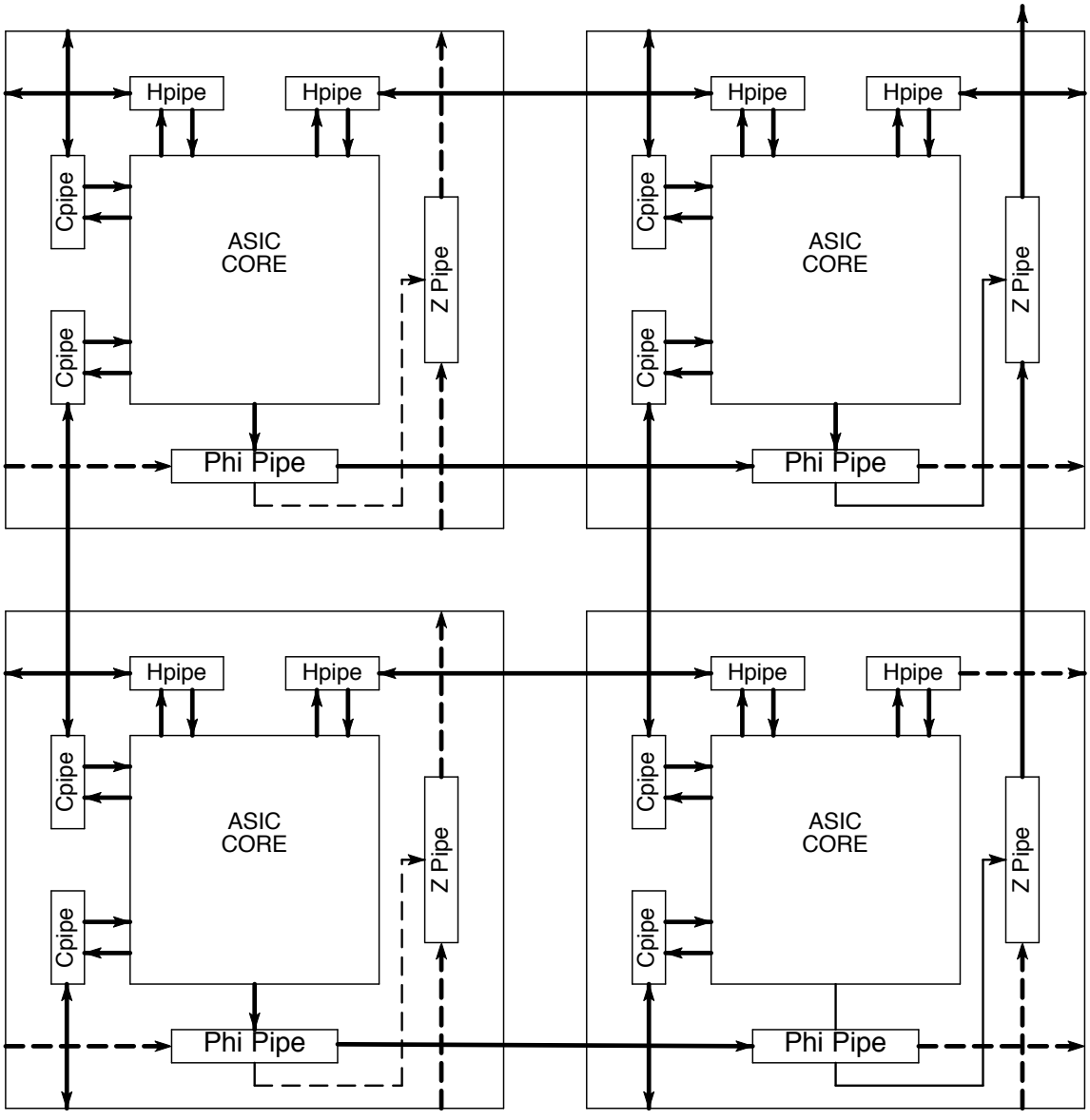

Figure 4: Two by two array of chips to illustrate hit data transfer and stub readout. CPipe is the hit data transfer in the z direction between adjacent chips. HPipe is the same as CPipe but for ϕ hits. PhiPipe is the row transfer of stubs in the ϕ direction and ZPipe is the column transfer of stubs. The light dashed line from CPipe to ZPipe indicates that row data is transferred for the row pipe to the column pipe and then sent off of the detector.

2.1.1 Inter-Chip Data Transfer

The ROC finds clusters up to 3 hits wide and it rejects wider clusters. This requires transferring hit data from the two edge strips between neighboring chips in the direction perpendicular to the strips (ϕ direction). Finding stubs with a maximum displacement of 3 strips requires transferring hit data from five edge strips but only for the outer (slave) sensor. The initial chip design has 16 sets of short z strips and 2 long z strips per chip. Thus, hit data from 42 strips must be sent between each pair of chips every 25 ns. The number of pads available is limited so the data is organized into a 6 deep pipeline that is 7 bits wide. Each chip has two sending pipelines (to the left and to the right) and two receiving ones. This is described in more detail in the pipeline section below.

Strip data must also be sent in the direction parallel to the strips (z direction). The short z length is 1.25 mm and the sensor separation is 1 mm so the aspect ratio is nearly 1 to 1. Thus, it is fairly common for a hit in an edge short z strip to be associated with a hit in the neighboring chip long z strip. The amount of data that needs to be transferred depends on the sensor location. For sensors near z=0, a single short z strip needs to be sent in both directions. For larger z, this changes to one strip in one direction. For still larger z, 2 short z strips must be sent in one direction. This increases to 3 strips near the end of the detector. Only cluster data needs to be sent in the z direction but encoding the data and sending addresses might result in more data than simply sending the hits. The design choice depends on the performance of the pipelines so this feature depends on the results from test chips.

2.1.2 Stub Readout

There are 80,000 short z strips on a sensor so it takes 17 bits to uniquely identify a short z strip. Three bits are needed for the Pt and 3 bits are used to identify the crossing. Adding 1 bit for future expansion gives 24 bits for every high Pt track. All 24 bits are read out in parallel. Monte Carlo simulations for very high energy events show that an individual chip may have as many as 9 stubs and a sensor may have many more. Reading out all this data in a single 25 ns crossing interval requires an extremely high readout rate. However, it is extremely unlikely to have two high energy events in successive crossings. Therefore, this design combines the data from two crossings and reads out the data in 50 ns periods. Most of the time, a sensor has no stubs or almost no stubs so averaging the data over two crossings nearly halves the maximum data rate.

Each chip has a maximum capacity of 8 stubs arranged in a pipeline that is 8 cells long. The pipeline is zero suppressed so its length ranges from 0 to 8 cells. The pipelines from all 5 chips in a row or column are strung together to form one long pipeline with a length between 0 and 40 cells. The pipeline transfer is asynchronous (see the next section) so timing estimates are only approximate. This will be measured with the test chip.

There are actually two pipelines in each chip. One pipeline is being loaded while the other is being transmitted ("ping pong" operation) so that the transmission has the full 50 ns to finish. The ϕ and z transfer pipelines also employ "ping pong" loading and transmission.

2.2 Asynchronous Readout

Power and speed are major limitations in the design of the readout chip. The readout chip is embedded between two sensors so noise from the readout coupling into the sensors is also a concern. These requirements suggest asynchronous pipelines for the data transfer. The designers of integrated circuits often fear asynchronous architectures for data transfer because

they can lead to unstable operation due to glitches and bus skew. However, they provide clear advantages in speed, low power and low noise: if no data must be transmitted, the circuitry remains inactive with a significant power saving and there is no need of high speed serial links that could be a potential source of noise.

The best match to this design is the MOUSETRAP micro-pipelines developed at Columbia University [5]. This pipeline is used both for transferring hit data between chips in the ϕ direction and also for sending stub data and event readout data off of the detector. Fig. 5 shows a block diagram of a 3 latch MOUSETRAP pipeline. The control logic is quite simple. It consists of only two lines: request and acknowledge. Moreover, once the request line has passed through the latch circuit, it is returned to the previous stage as the acknowledge signal. Latch control is by a single exclusive NOR (XNOR) gate.

Pipeline operations proceeds as follows: Initially all the latched data is 0 so both the acknowledge and done lines are 0, the output of the XNOR is 1 and the latch is transparent. Data is then loaded into latch 0 with the done bit set to 1. After it passes through latch 1, the XNOR for latch 1 goes to 0 and the data is latched into latch 1. The data continues to propagate through latch 2 which in turn sets the XNOR of latch 2 to 0. This latches the data into latch 2. At the same time, the done from latch 2 propagates back to latch 1 on the acknowledge line which sets the XNOR for latch 1 back to 1 so latch 1 becomes transparent and ready for the next set of data. To load latch 1 with new data, data is loaded into latch 0 with the done bit set to 0. When this propagates through latch 1, the XNOR again goes to 1 and the data is latched. The data cannot propagate further until latch 2 goes transparent again.

This is an elegantly simple protocol but it is very sensitive to glitches on the control lines. In the above example, when latch 1 is in the latched mode with both done and acknowledge set to logic 1, a momentary glitch to 0 on the acknowledge line will make the latch momentarily transparent and latch new data. A VERILOG model for the front end chip shows this problem. A simple solution is to adopt the dual rail protocol for the control lines. Dual rail uses 2 lines for every bit with 01 representing logic 0 and 10 representing logic one. The 00 and 11 states are ignored so any glitches on the two control lines are ignored. This feature was inserted into the VERILOG models and all the glitches were eliminated. Of course, there may be other subtle effects that are not properly included in the VERILOG model so a prototype chip is being designed to test out these features.

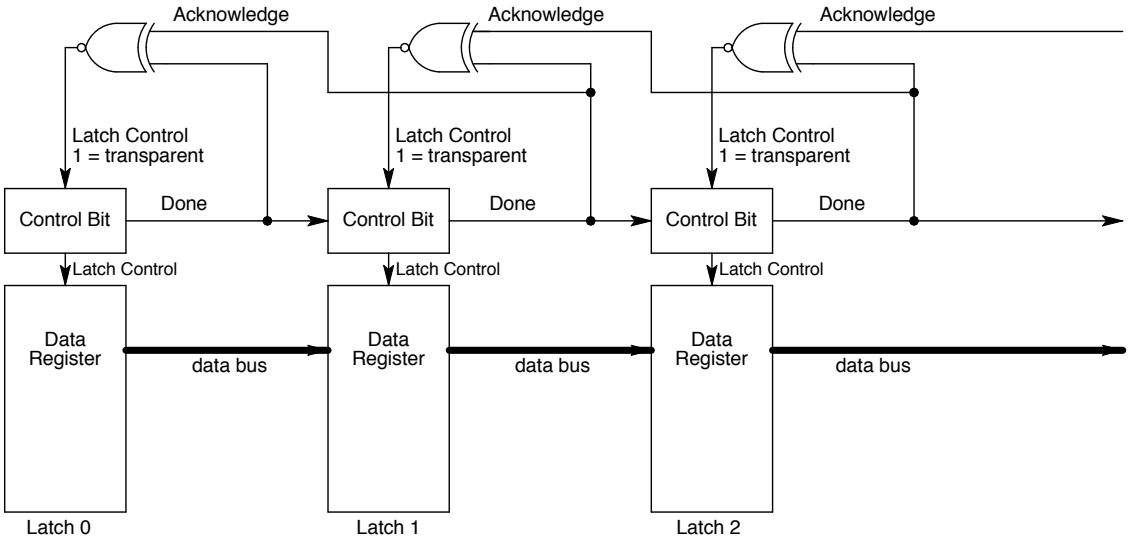

Figure 5. A three element MOUSETRAP pipeline

Fig.6. shows a simplified section of the MOUSETRAP pipeline for sending hit data in the ϕ direction between chips. There are two copies of the pipeline in each chip. Copy A is loaded in the left chip for the current cycle and unloaded in the right chip. The right chip data are delayed by 2 crossings. Data in copy B are being transmitted. At the next crossing clock, the pipelines are switched and data in copy B is transmitted to the other chip.

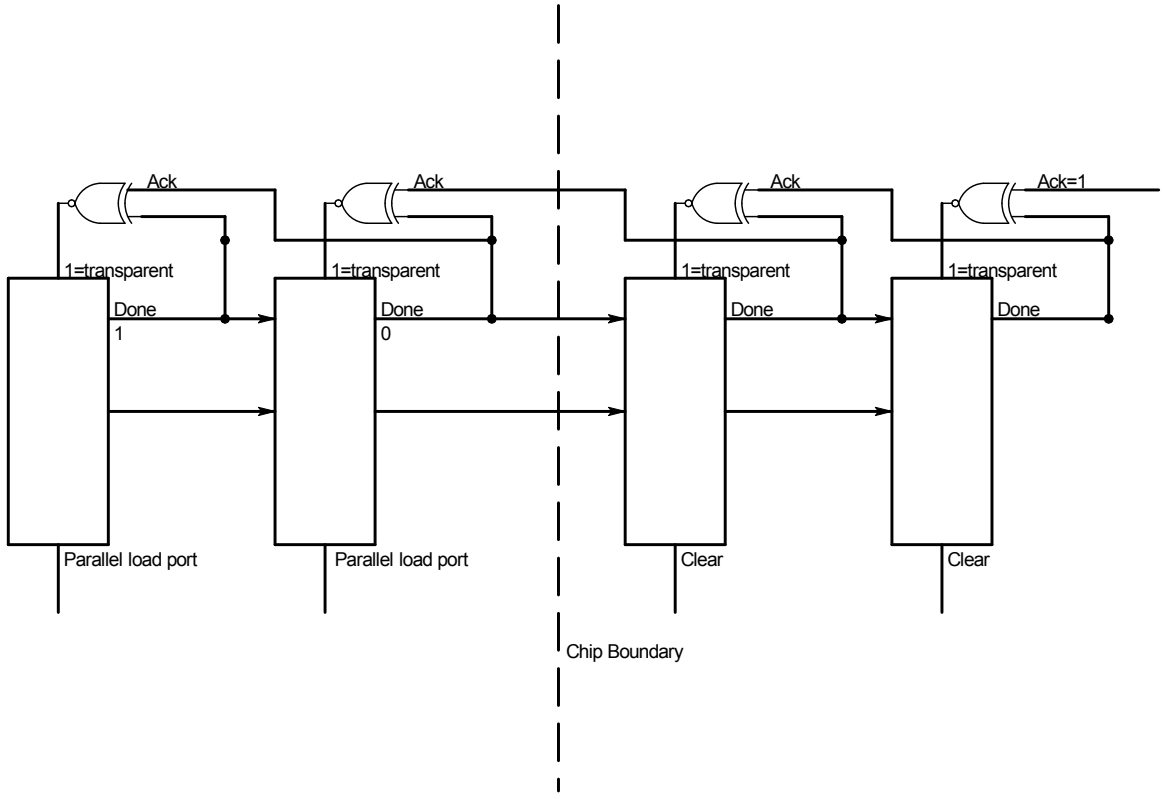

Figure 6. MOUSETRAP pipeline for sending edge strip hit information between two adjacent chips.

2.3 Priority Encoders

Each short z has 200 possible stub addresses and each chip has 16 short z sections so there are 3200 possible stub addresses. A busy jet event will have perhaps 10 stubs but there is a maximum of 8 stubs (ordered by Pt) per chip. This presents an ideal application of priority encoders. The encoding is done in two steps. Each short z section encodes up to 4 hits in parallel. Then, up to 8 hits from the 16 short z sets are encoded in approximate Pt order.

This design uses the Mephisto encoder [6] because it sends an enable signal back to the selected entry (Fig. 7). This feature is used in two ways. First, it gates off the found stub and sends all the rest of the stub addresses to a second encoder so that the two highest and two lowest stubs can be found in one crossing without any additional clocks. Second, it gates the stub Pt for the selected stub onto a bus so that the Pt can be stored with the stub address.

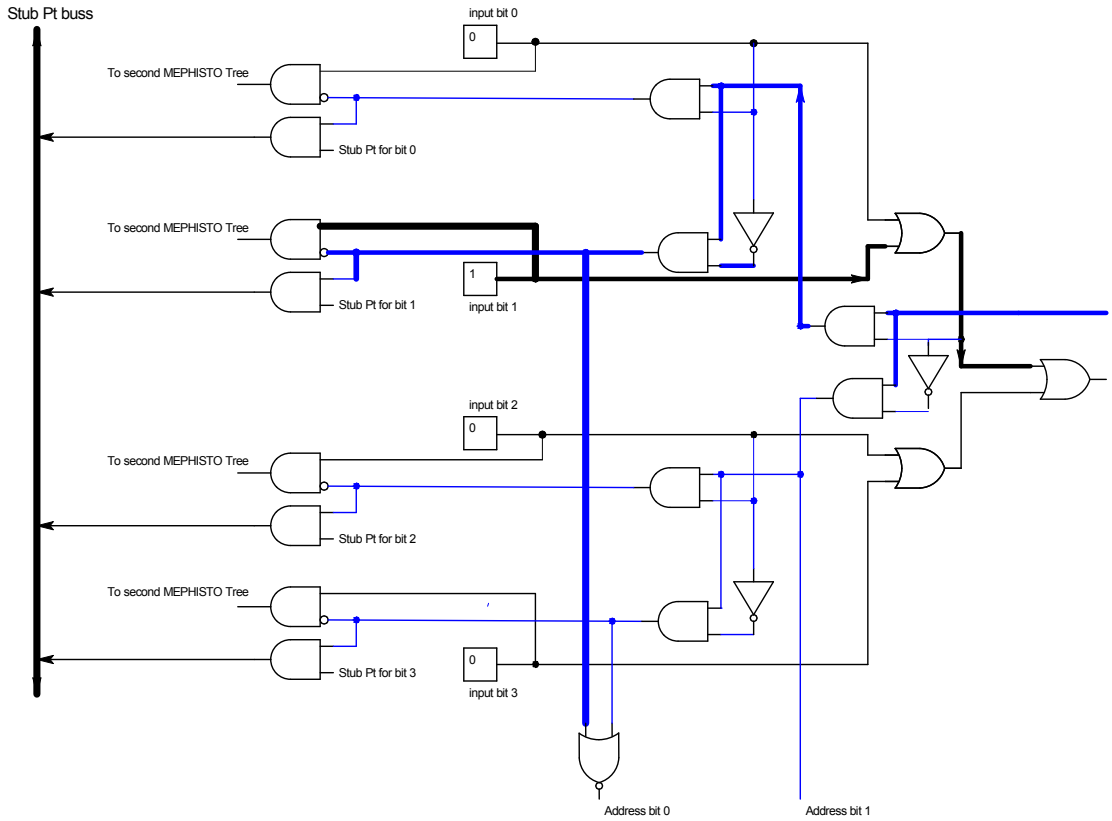

Figure 7. Logic diagram for a 4 bit MEPHISTO priority encoder. The selected bit is gated off and the data is sent to a second encoder so that the 2 highest entries can be found nearly simultaneously. The feedback is also used to gate the associated stub Pt onto a readout buss so that it can be recorded along with the address. For clarity only 1 bit of the stub Pt is shown in the figure.

The next step is to select the 8 highest Pt hits from the 64 possible candidates. Simulations indicate that it is very rare to have more than 8 stubs per chip so a somewhat crude Pt ordering is adequate. Only the absolute value of the Pt is needed for ordering and 3 strips have at most 4 different values of Pt. When a stub is found in a short z, it is stored either in a high Pt list (Pt values of 0 and 1) or a low Pt list (Pt values of 2 and 3). These values are then strung together into a single list that is 128 bits long. The first and last 32 element groups are the high Pt registers and the middle 64 are the low Pt registers. Selecting the 4 highest and 4 lowest then selects the highest Pt values first. Preliminary work indicates that this may be done in one LHC crossing.

3. Off-Detector Data Processing

Simulations show that the projection accuracy of tracklets from the inner to the middle layer is of order 100 μm in ϕ and less than 5 mm in z (Fig. 3). This is accurate enough so that additional fitting is not needed. All that needs to be done is to match the incoming tracklet with a local tracklet. The main issues are the search algorithms to match stubs into tracklets, the algorithms to match tracklets between layers and routing the data from one layer to another.

Minimizing the amount of data transfer between detector elements is very important. The detector is constructed so that the outer stack on a rod (Fig. 2) covers the inner stack in ϕ so no communication between rods in a layer is required. Communication along the rod in z is accomplished by reading a rod into a small number of FPGAs. Data from sensors at the boundaries between FPGA's are sent in parallel to both FPGAs

Efficient stub matching is accomplished by dividing the inner stack into regions that are roughly the size of a readout chip. This size is chosen to limit the number of stubs that are likely to be in the region so it is dependent on the stack position. The upper stack region is large enough to include all possible high Pt tracklets. That is, any tracklet with a stub in a lower stack region will be contained in the upper stack region. Incoming rod data is sorted into these regions as it arrives from the detector. All possible combinations of stubs are then checked to find valid tracklets. The number of stubs is limited to 8 per region so there are at most 64 comparisons. The beam axis and length of the interaction region are constraints in finding valid tracklets.

Projections to other layers are done via table lookups. For a perfect cylindrical detector, the projection in the phi direction would depend only on the difference in phi between the two stubs (assuming that the track comes from the beam axis). Real sensors are flat rather than cylindrical but one can compensate for the sensor flatness by simply adding a correction that is based on the distance of the distance of the lower stub from the center of the sensor. Thus, the projection operation requires two parallel table look ups and an add.

Tracklets are projected to regions in the other layers. Tracklets that project to the boundary area of a region are sent to the two (or more) adjacent regions. Since the regions are selected to have only a few high Pt tracks, it is easy to compare arriving tracklets with the local tracklets. Simulations studies are on going but it appears that matched tracklets are accurate enough for a first level trigger.

4. Summary

A new approach to level 1 tracking triggers that relies heavily on the design of the detector to filter out tracks with Pt < 2 GeV/c is being developed for the SLHC. Tracks are constructed incrementally from this filtered data by first finding track stubs on the 1 mm scale, then finding short tracks (tracklets) on the 40 mm scale and finally assembling tracklets from 3 different layers into high Pt tracks.

Asynchronous logic holds great promise for reading out the large amount of trigger data within a limited power budget and without generating excessive noise in the silicon sensors. The simple control logic of the MOUSETRAP pipeline adds very little overhead either to chip logic or chip pads so it easy to include in the design. We feel that the combination of detector design and asynchronous logic will enable building a viable L1 tracking trigger system.

Work is proceeding on the design of both the off-detector processing system and a prototype chip to test the readout system.

References

[1] W. Ashmanskas et. al., *Silicon Vertex tracker: a fast precise tracking trigger for CDF*, Nucl. Inst. and Methods, A**447** (2000) 218

[2] A. Andrezza et. al., *The Fast Tracker Real Time Processor and Its Impact on Muon Isolation, Tau and b-Jet Online Selections at ATLAS*, Real Time Conference (RT), 2010 17Th IEEE-NPSS

[3] M. Johnson et al., *Readout chip for an L1 tracking trigger using asynchronous logic,*JINST 7,(2012) CO8004

[4] E. Hazen et al., *Architecture of a level 1 track trigger for the CMS experiment*, JINST 5 (2010) CO8004

[5] A. Pietropaolo et al., *MOUSETRAP: Ultra High Speed Transition Signalling Asynchronous Pipeline*, Proc. International Conf. Computer Design (ICCD), Pages 9-17, Nov. 2001

[6] P. Fischer et al.,NIM. A431(1999) 134.