DØ note 3697

Digital Board Algorithm for the L1 FPS Trigger

Arnaud Lucotte

SUNY at Stony Brook

Abstract

This document describes the Level-1 (L1) Trigger algorithm developped for the FPS detector at the Digital Board level.
The first section presents the motivations and constraints that drive the development of the algorithm. Section 2 defines the FPS detector parameters relevant for our study (geometry, data readout). Section 3 recalls briefly the trigger architecture (and data path) that is used to generate a L1 decision. The following section is devoted to the description of the L1 trigger logic that is supposed to be implemented in the daughter board of the Digital Board (DB). This algorithm generates a cluster counts, that are used at the Trigger Manager level, and a cluster list, that is sent to the L2pre-processor upon a L1 decision. The algorithm is written using the *vhdl* language, which is a standard language developped for both PLD's and FPGA's. Simulations are perfomed in the Altera context, and results are reported in terms of resources (Logic Cells) needed and time estimates.

contacts:

M. Bhattacharjee, F. Borcherding, P. Grannis, M. Martin, J. Olsen
A. Lucotte at lucotte@fnal.gov

Contents

1 Motivations and constraints

1.1 Physics motivations

Motivations and performance for L1 electron/photon triggers using the FPS detector have been extensively studied elsewhere [1] [2]. Of relevance for this note, we recall that:

- High and low p_T electron/photon trigger are based on the combination of EC calorimeter and FPS cluster information. A high (5-10 mips) threshold is used to detect high p_T object in the FPS, like W/Z electrons. A low (2-3 mips) threshold is used to cover lower p_T range in physics, as it is needed for B physics, where electrons typically have p_T = 2-5 GeV/c –see Fig.1.1 and 1.2

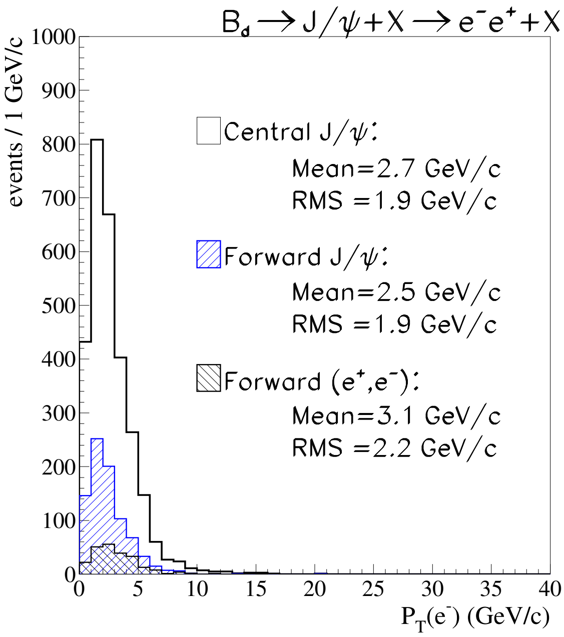

Figure 1.1: *Electron* p_T *in* $J/\psi \to e^- e^+$ *decays*

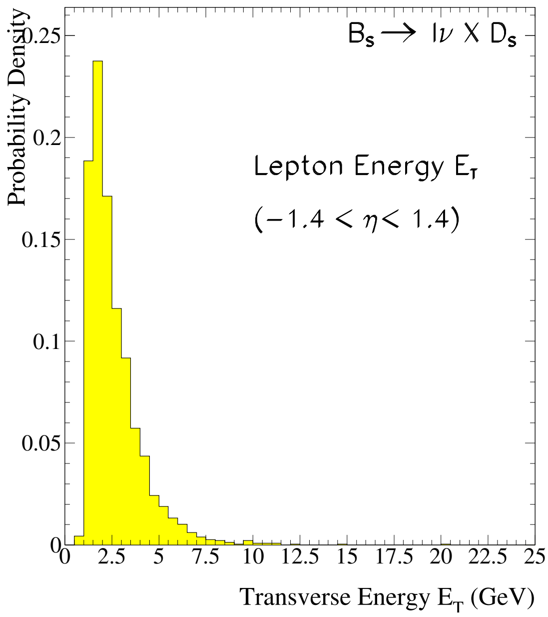

Figure 1.2: *Electron* p_T *in semi-leptonic b decays*

- L1 trigger is based on the coincidence of a FPS cluster with a EC tower above a certain threshold within the same quadrant. It is shown that the quadrant matching brings a factor 2-4 in rate rejection at L1, while keeping high trigger efficiency. besides, quadrant match is a mandatory condition to detect low p_T eletron trigger, as needed in B physics via $J/\psi \to e^- e^+$ and b semi-leptonic decay [2].

- L2 trigger is based on the reconstrucion of 3-D FPS clusters which are matched with 0.2×0.2 EC towers at the L2 Global. Again, the combination EC -FPS alone is expected to bring a factor 2 improvement in rate rejection [3].

1.2 Physics Constraints

1.2.1 Occupancy (Low p_T electron signature)

The use of a low threshold in EC (2.5 GeV) and in the FPS(3 mips) is required to detect low p_T di-electrons. Of course, not all clusters can be reported by the L1 to L2. For example, out of the 144 strips of a large module, there is no way to report the 72 possible 1-strip wide clusters to L2 without huge resources.
The maximum number of clusters per event seen in one module (22.5^0 sector) in a mip layer and in a shower layer are shown in Fig. 1.3 and Fig. 1.4 respectively. The L1 FPS decision is driven by the maximum number of clusters detected in the shower layer. A maximum of 9.4 clusters per module are expected with a 1 mip threshold for u or v sublayers.

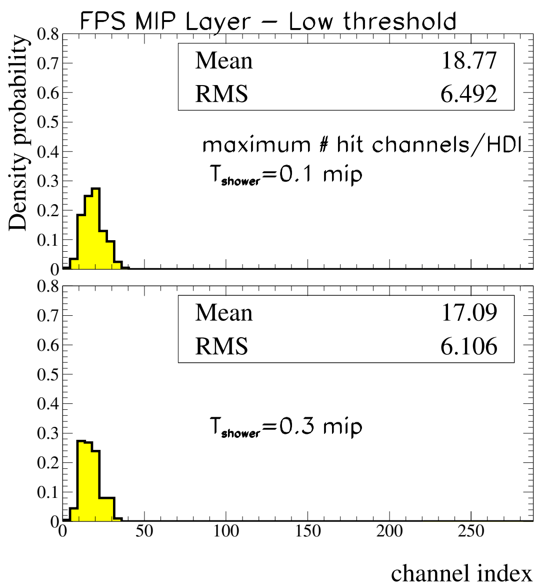

Figure 1.3: *Maximum number of clusters in an upstream layer. Dijet + 2 mbias events generated in 5 bins from 2 to 500 GeV have been used.*

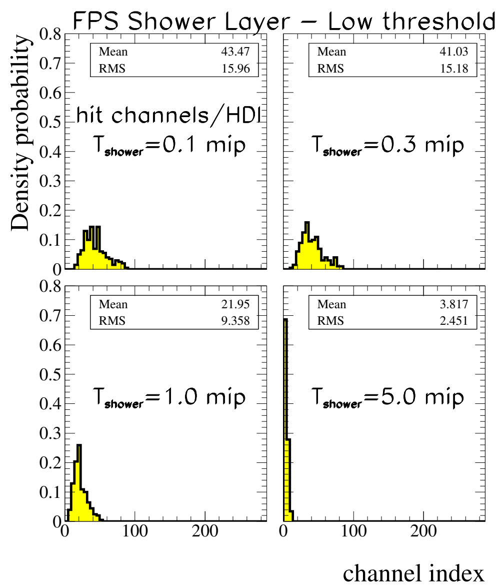

Figure 1.4: *Maximum number of clusters in a shower layer. Dijet + 2 mbias events generated in 5 bins from 2 to 500 GeV have been used.*

A constraint on the L1 algorithm is given by the separation between electrons issued from a $J/\psi \to e^-e^+$. Fig. 2.7 displays the separation of the two electrons using the generated

MC information in terms of $\Delta\phi$, $\Delta\eta$, *etc*.... Electron pairs appear quite collimated in the forward region. Fig. 2.8 shows the strip separation in terms of number of strips separating two clusters in a shower layer. It shows that we must be able to report two distinct clusters in a 10 strip-wide region.

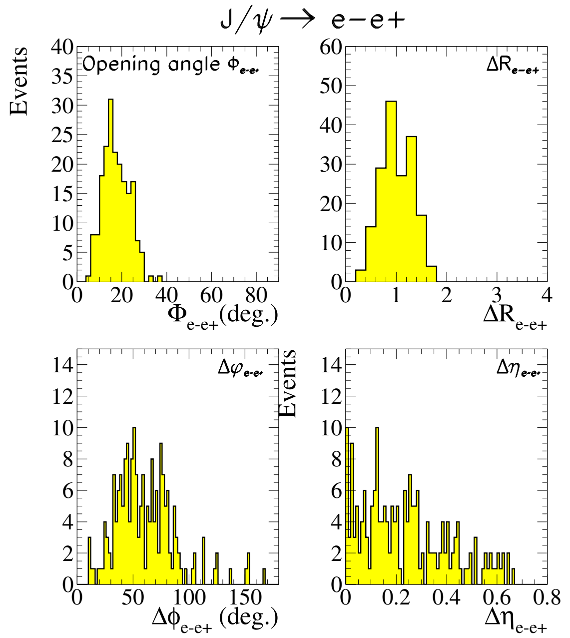

Figure 1.5: *Opening angle* $\Delta\Phi_e e$*,* ΔR*,* $\Delta\phi$ *and* $\Delta\eta$ *of the two electrons originated from* $\mathrm{J}/\psi \rightarrow \mathrm{e}^-\mathrm{e}^+$ *decay*

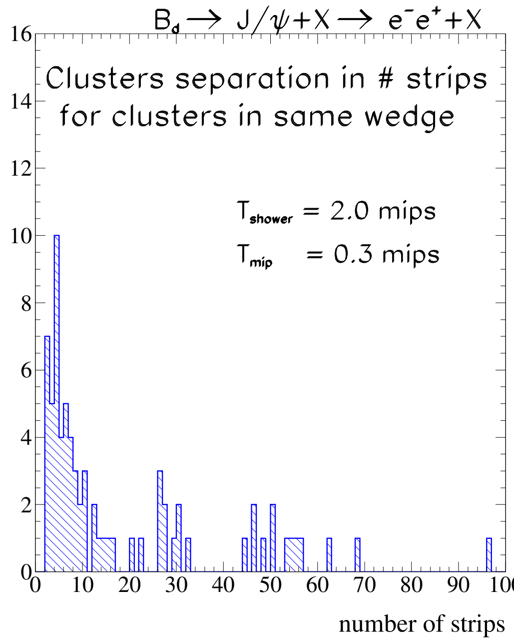

Figure 1.6: *Separation in number of strip for those di-electron targetted in the same* ϕ*-sector for electrons originated from* $\mathrm{J}/\psi \rightarrow \mathrm{e}^-\mathrm{e}^+$ *decay*

1.2.2 Constraints due to EC vs FPS matching

In order to keep an acceptable level of di-jet rates, calorimeter match with FPS clusters is mandatory, as we mentionned earlier. The fine mathing between 0.2×0.2 EC towers and FPS 3D clusters takes place at Level-2 (L2), after (u, v) pairs are transformed into (η, ϕ) coordinates. The important point is that L2 trigger uses (only) as input the L1 cluster list. There is no access to SVX-II chips (digitized information –section 2) from L2. It is therefore of vital importance that the cluster list established and reported at L1 reflects the proper location of the clusters in the module *per* η *bin*.

The constraints on the L1 algorithm are thus:

1. to be able to report in priority at least one cluster per 0.2 η bin, so that the matching efficiency is not affected at L2

2. to report as many L1 clusters as possible, given the limitation in time and resources of the FPS Trigger design

2 FPS Detector Parameters

2.1 FPS geometry

The FPS geometry is defined in detail in [4] [5]. Table 1 regroups the relevant parameters for our purposes.

The FPS is divided into two detectors: South and North detectors. Each side is composed with sixteen 22.5^0 $\phi-$sectors –see Fig. 2.7 and 2.8. Each sector is a set of four layers, two "mip" layers (u,v) in front of a lead sheet, and two "shower" layers (u,v), behind the lead. Among these sixteen sectors, fourteen "standard", covering the range $1.5 < |\eta| < 2.5$ and have each a total number of strips:

$$103 \times 2(\mathrm{u}+\mathrm{v}) \ + \ 144 \times 2(\mathrm{u}+\mathrm{v}) = 494 \text{ strips}$$

The two remaining sectors ("special") are troncated due to solenoid magnet services. They cover the range $1.8 < |\eta| < 2.5$ and have a total of:

$$72 \times 2(\mathrm{u}+\mathrm{v}) \ + \ 72 \times 2(\mathrm{u}+\mathrm{v}) = 288 \text{ strips}$$

Module type	# modules N+S	# channels / module u+v	η coverage
Large Module	28	288	1.5 - 2.5
Small Module	28	206	1.65-2.5
Special Module	4	144	1.8 -2.5
Total FPS	64	14,984	1.5-2.5

Table 1: *FPS Detector Parameters*

2.2 FPS readout

Clear optical waveguides route the signal from the FPS to the Visible Light Photon Counter (VLPC) cassette. One scintillator strip from the detector couples to one channel in the VLPC. Here, light signal is converted into electrical signal with high efficiency and high gain (40K). A schematic of the VLPC is shown on Fig. 2.9. A VLPC cassette has two boards of

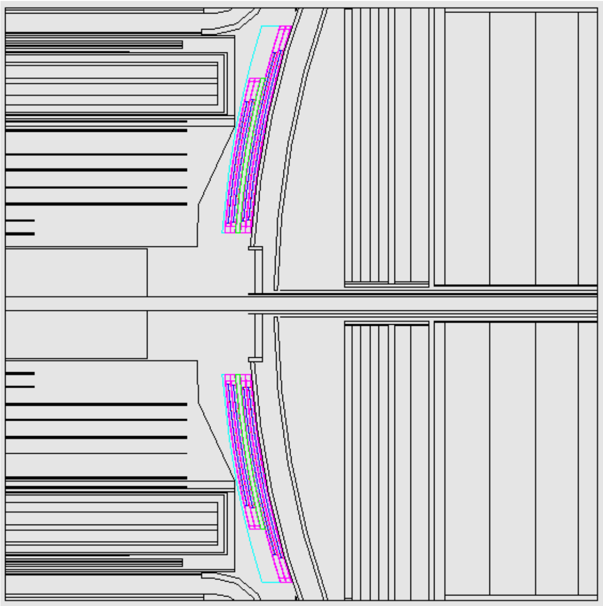

Figure 2.7: *Z-view of the FPS detector*

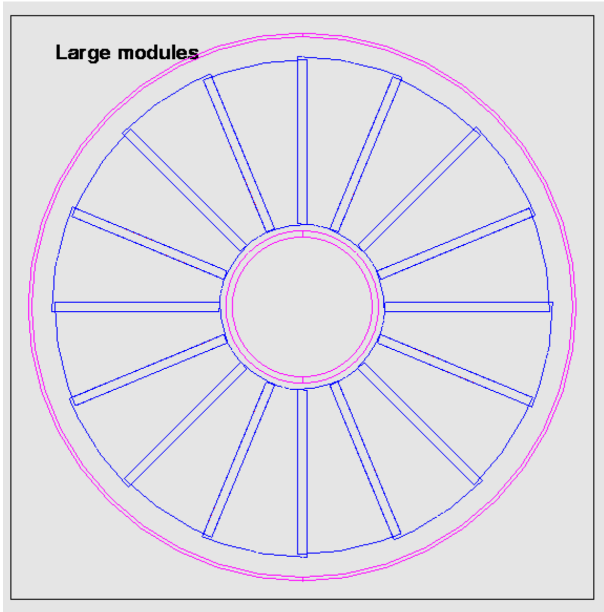

Figure 2.8: *XY-view of the 16 sectors of the FPS*

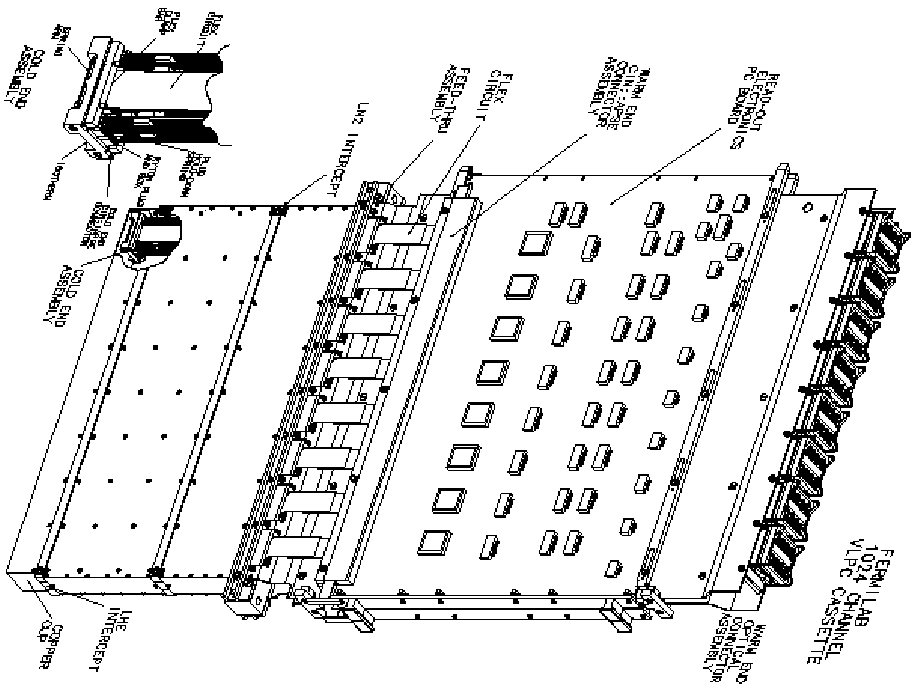

Figure 2.9: *VLPC cassette with two Boards of 512 channels*

512 channels, and typically receives the 288+206=494 channels of a 22.5^0 readout $\phi-$sector.

3 Trigger Architecture and timing

3.1 Architecture and data path

The Trigger Architecture is described in detail in [7] and presented in Fig.3.11. As a reminder:

- **AFE Board**: The Analog Front End (AFE) Boards are located on the cassette itself – see Fig.2.9. A fraction of the ouput charge is drained away before input to SIFT chips. The signal from each VLPC channel is then charge divided into two analog outputs: a low gain arm (27% charge fraction) is used to set a High threshold, and a high gain arm (9%) is used to set the Low threshold. These settings (charge division, dynamical range ..) are reported in [6]. These outputs feed two disctinct **Multi Chip Modules (MCM's)**, one for each arm. Each of these arms has its own **SIFT** (analog signal) and **SVX-II** (digitized signal) readout chip. One MCM of 4 SVX-II chips is devoted to the readout of mip signals, while a bigger MCM of 8 SVX-II chips is used to readout and digitize shower layer signals (2 thresholds).

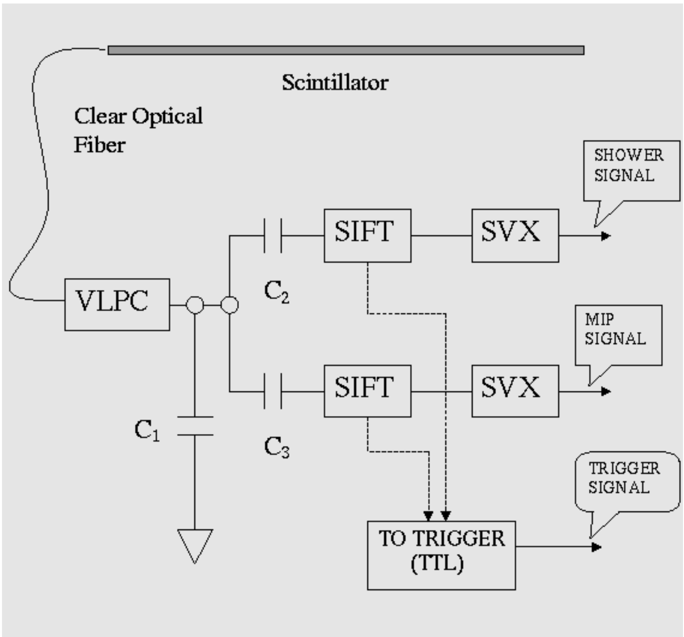

Figure 3.10: *Readout scheme in the AFE Board.*

A schematic of the readout is presented in Fig. 3.10.
SIFT chips provide a fast trigger signal and also drives the analog signal to the SVX-II chip. Trigger pickoff signals are then fed into a series of large Field Programmable Gate

Arrays (FPGA's –Xylinks product) or Programmable Logic Devices (PLD's –Altera product). The PLD's are pre-loaded with the logic to form the clusters.

- **Digital Board (DB)**: There are 8 DB per side of the detector (16 total). Each DB collects the information from two AFE Boards, ie from a di-sector ($2\times$ 22.5^0 sectors in ϕ) Thus the total number of input channels per DB is:

$$(144 \times 2(\mathrm{High/Low}) + 103) \times 2(\mathrm{u}+\mathrm{v}) \times 2(2 \text{ sectors}) = 1,564$$

u and v layers are treated independantely, and for each strip, both High and Low channel are received by the DB for the shower layers. The role of the DB is to form and count u clusters as well as v clusters independentely, generating outputs to L2pre-processor (list of clusters) and to L1 Trigger Manager (cluster count). Again, u and v are treated completely independetely in two distinct daughter boards.
The DB forms and counts the clusters, before sending them to the Collector Boards. A total of 16 H+L u clusters and 16 (H+L) v clusters can be sent to the Collector Board [9]. This number has been updated lately.

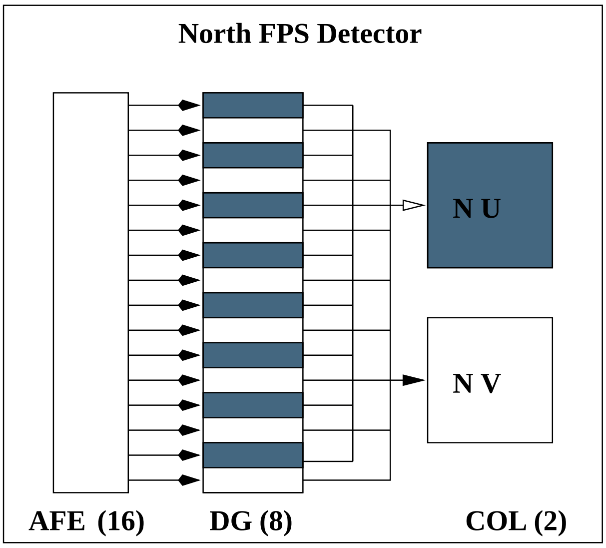

Figure 3.11: *Block diagram of the Trigger Architecture for the FPS detector –North side.*

- **Collector (COL) and Broadcaster Board (BR)**: There are four collector boards (COL) for the entire detector (2 sides). Each board collects the outputs of eight octants (1 octant = 2 22.5^0 sectors) for u North, u South, v North and v South independentely.

The maximum number of cluster one collector can receive is 16 H+L clusters. The Collector/Broadcaster then regroups these clusters into a list of 48 clusters total per side, ie 48 u South, 48 u North, 48 v South and 48 v North. The details in the troncation scheme are still under discussion.
The cluster count is then transferred from the COLs to the four L1 Brodcasters (BR) and then to the L1 Trigger Manager. Upon a L1 decision, the cluster list is sent to the L2 pre-processor crate.

- **Level 1 Trigger Manager (L1 TM)**: The TM receives the L1 COL data. Its role is to define 6 sets of trigger terms formed using the COL cluster counts, before sending them to the L1 Framework, where a combination of several detector specific terms are used to reach a L1 decision.

- **Level 1 Trigger Framework (L1 FWK)**: The role of the L1FWK Its role is to combine detector specific trigger terms from four distinct systems (CFT, Preshowers, Calorimeter, Muon). The L1FWK supports a total of 128 pre-programmed combinations of detector specific triggers through FPGA's. The L1FWK determines if at least one of these triggers is satisfied. Upon an accept decision, the event data is digitized (SVX-II chip for the PS) and routed from the pipeline to the L2 input buffers.

3.2 Timing

$4.2\mu s$ are needed to reach a L1 (global) decision without deadtime at 7 MHz. This time includes the time needed to transfer the signal from the VLPC to the SIFT (1 crossing or 132 ns), to form & count the clusters (DB algorithm, 1-2 cross.), to transfer the L1 DB data to the COL (1 cross.) and from there to the TM (1 cross.). Each step takes into account the time for the transfer and signal processing.
As event come out every 132 ns, the readout electronics system must be able to handle data while a L1 decision is formed, and be ready to receive the data from the next crossing. Each Front End digitizing crate includes pipelines to retain data from 32 crossings without a deadtime at 7 MHz. For the FPS trigger, these pipelines are used to store the list of clusters generated at L1 in the DB's, and to be transferred to the L2pre-processors upon a L1 decision.

We therefore have two timings for two data types:

1. L1 timing: it concerns the DB algorithm that computes and sends the cluster count to the L1 COL. Here the timing must be as short as possible, typically $\tilde{1}$ crossing.

2. L2 timing: it regards the algorithm that forms and sort a list of clusters to be used in the L2 pre-processor. A few μs are here available to transfer the data to the L2pp upon a L1 decicion. Event data must therefore be stored in some internal memory inside the DB daughter card or fed into FIFO's, so that the event from the next crossing can be readout.

We thus need to consider two kinds of data, defined after their destination and timing constraints.

4 L1 Digital Board Trigger

4.1 L1 Digital Board Data

4.1.1 L1 DB Data

The DB provide the information that is used at the L1 Framework, where the decision takes place. The L1 decision is based on cluster counts combined with other detector information. In the FPS, four cluster types per orientation (u or v) are reported and listed in table 2.

u layers	v layers	Maximum number reported by the DB
# Hu-electron	# Hv-electron	16
# Hu-shower	# Hv-shower	16
# Lu-electron	# Lv-electron	16
# Lu-shower	# Lv-shower	16

Table 2: *Cluster counts output formed at the DB level. u and v objects are formed and counted independently. H=High, L=Low thresholds. Cluster types: Electron are defined as matched clusters, while showers are unmatched clusters. The Matching is defined by the coincidence of a MIP signal in the 7-bit window in front of the shower.*

At the DB level, there is a limitations on the cluster counts: the maximum number of clusters that can be sent to the COL is 16 per type (16 H+L-electrons, 16 H+L-showers, for both u and v clusters).

4.1.2 L2 DB Data

At the DB, clusters are formed for each type and two different thresholds. The list of these clusters, together with all their characteristics, is sent to the L2 pre-processor upon a L1 decision. A cluster is completely defined by a set of 24 bits: where:

	Address								Width			type	A	G	H	L	MIP pattern						
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- **Address** : is the strip index of the cluster within the layer, given by the location of the 1st strip plus half the width of the cluster. As strip indices goe from 1 to 144, 9 bits are needed to code the info.
- **Width** : is the number of adjacent strips above the same threshold. a Maximum width of 8 is set. 3 bits are needed.
- **MIP bit pattern**: this is the window that is in front of the cluster found in the shower layer. Window boundaries are given by:

$$\text{Address} \;-\; 3 \;\pm 3 \text{ strips}$$

- **Type** : cluster type is determined by the coincidence of a MIP signal in front of the shower, located in the MIP window. 1 bit is needed. the cluster may be either a Matched cluster (=1) or a shower cluster (=0) when there is no mip signal in the MIP widnow.
- **Side G**: North / South information, 1 bit
- **Strip orientation A**: strips are u or v orientated, 1 bit.
- **Thredholds**: strip can be above a High (H) or Low (L) threshold. 2 bits are used here.

4.1.3 Electron and Shower Definition

There is NO "electron" nor "EM shower" at the DB level. There are only a set of clusters *of particular orientation* u or v that are matched or not with a MIP signal in the corresponding u or v mip layer. Therefore we have at the L1DB:

u electron, H/L threshold
u shower, H/L threshold
v electron, H/L threshold
v shower, H/L threshold

It is only at the TM level that "electron" and "EM shower" are defined: There, both u and v information (counts) are combined together to form:

$$\begin{aligned} \text{electron (H/L)} \equiv\ & u \ \text{electron(H/L) AND } v \ \text{electron(H/L)} \\ \text{shower (H/L)} \equiv\ & u \ \text{electron(H/L) AND } v \ \text{shower(H/L) OR} \\ & u \ \text{shower(H/L) AND } v \ \text{electron(H/L) OR} \\ & u \ \text{shower(H/L) AND } v \ \text{shower(H/L)} \end{aligned}$$

No "mixing" between H/L threshold have been tested so far in the TMalgorithm. This may be useful to keep high efficiency for high p_T leptons as suggested by the Central region algorithm [8].

4.2 Digital Board Algorithm

The purpose of the L1 DGB algorithm is to form and count clusters present in u and v layers. Note that within a DB, u and v clusters are formed in two different daughter boards, being therefore independent. We present here a schematic of the algorithm design for one daughter board, which is chosen to apply on u layers for illustration purposes.

4.2.1 DB Daughter Board block diagramm

We present here a schematic of the algorithm for one daughter board of the DB.

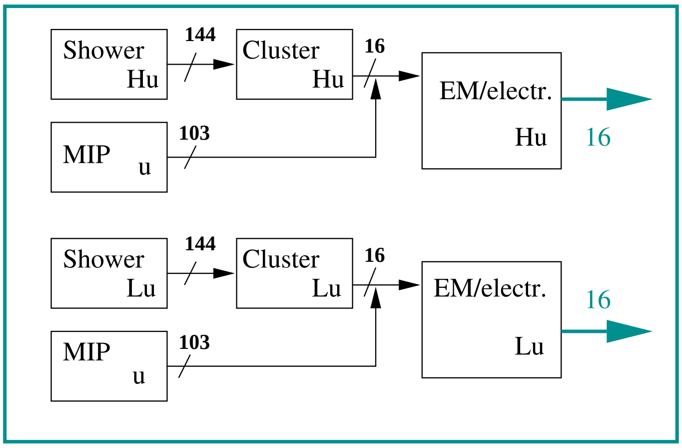

Figure 4.12: *Schematic of the algorithm used in one daughter board card for u clusters. 144 shower channels and 103 mip channels are used as input, for both high and low thresholds. A total of 16 H+L clusters are generated. The same design apply for v clusters in the other daughter board.*

Two algorithms are run in parallel to form both low and high threshold clusters. Fig. 4.12 displays the block diagram of the algorithm run in the board.

4.2.2 Cluster Finder

Segmentation within a FPS module

As mentionned in the first section, the constraint that drives the DB algorithm is the necessity to find and report at least one cluster per η bin. The correspondance η and *strip index* is not bi-univocal because of the 11.25^0 angle that stereo strips make with the horizontal plane. Fig. 4.13 displays the correspondance η vs *strip index*, showing both lower and upper boundaries (in strip index) for each η range. A front view of this projection onto the FPS module is illustrated in Fig. 4.14.

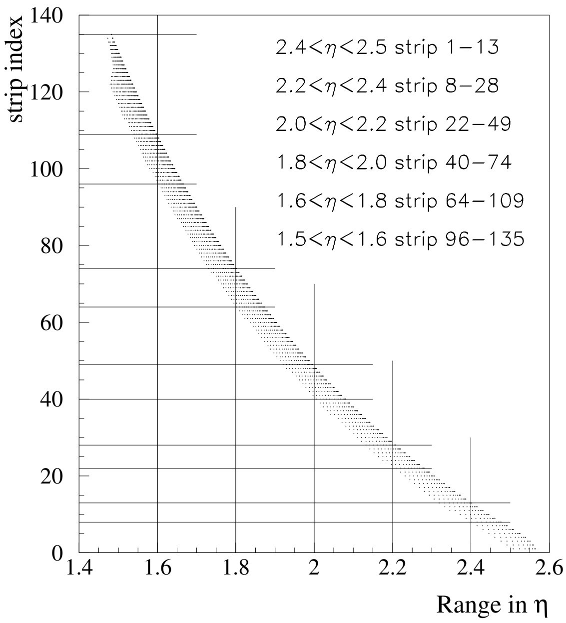

Figure 4.13: *EC calorimeter η segmentation versus strip index in a FPS module. A total of 6 swaths are defined from* $1.5 < |\eta| < 2.5$.

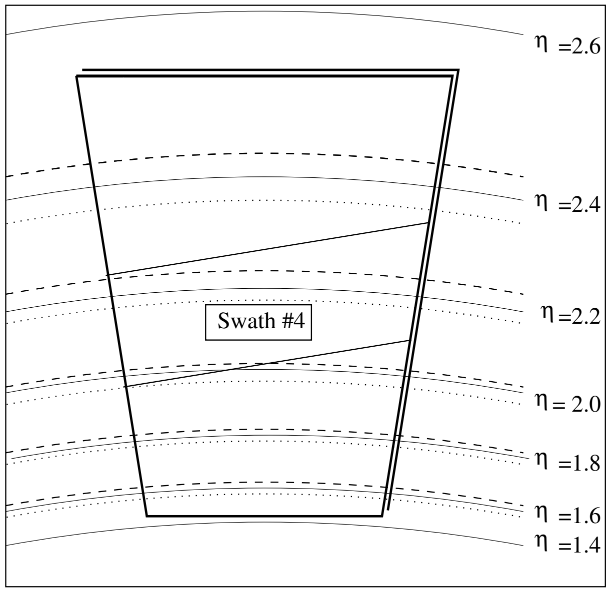

Figure 4.14: *EC calorimeter η segmentation projected onto a FPS module - front view. A total of 6 swaths are defined from* $1.5 < |\eta| < 2.5$.

Correponding values for these boundaries in strips are reported on Fig. 4.13. Table 3 report the values actually used. The differences are due to the necessity to include the overlap between swaths, and convenience reasons explained in the code itself.

η-range in EC	Boundaries / shower (in strip index)	Boundaries / MIP (in strip index)
2.4 < $\lvert\eta\rvert$ <2.5	1 - 14	1 - 14
2.2< $\lvert\eta\rvert$ <2.4	8 - 28	1 - 28
2.0< $\lvert\eta\rvert$ <2.2	22 - 49	15 - 48
1.8< $\lvert\eta\rvert$ <2.0	40 - 74	34 - 74
1.6< $\lvert\eta\rvert$ <1.8	64 - 109	58 - 109
1.5< $\lvert\eta\rvert$ <1.6	96 - 144	88 - 103

Table 3: *Boundarie of swaths defined in a u module of the FPS detector. The second column indicates boudaries for the u shower layer, the third for the u mip layer. At least 2 clusters per bin can be reported.*

Description of the Algorithm

The *vhd* files needed for the project are reported in appendix. The algorithm is here explained for ONE threshold (H) and ONE orientation (u). It is organized as follows: a main routine *fps_sector_uh* calls 6 subroutines that form and count and report at least two clusters. These 6 subroutines correspond to the 6 regions or "swaths" (corresponding subroutines are nammed "subsector") defined by the EC segmentation. Fig. 4.15 displays the structure of the algorithm.

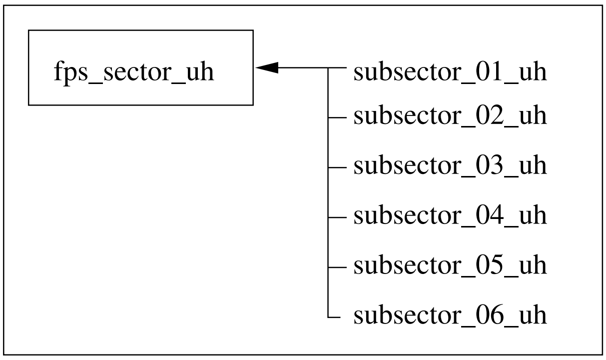

Figure 4.15: *Algorithm used for the entire FPS sector. Only u channels are considered, with a high threshold. All subroutines are run in parallel processing*

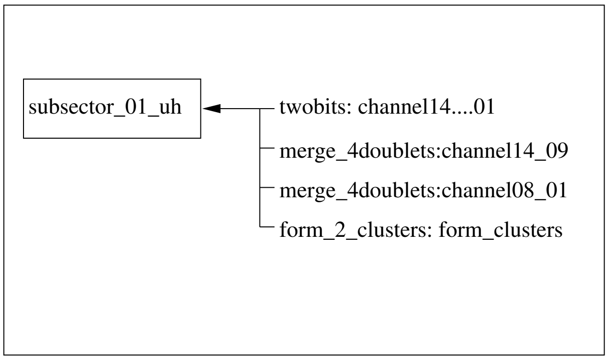

Figure 4.16: *Algorithm used for one swath (here the 1st one). All subroutines used are displayed on the figure. All subroutines are run in parallel processing*

In each "swath", the algorithm does the following tasks –see Fig.4.16:

- **twobits**: fills 8 vecteurs with the location of the first strip of all 1, 2, 3 ...,8-strip wide clusters. Each of these clusters are tested for their lower and upper bound so that a

4-strip wide cluster does not appear as four 1-strip wide clusters, nor as two 2-strips wide clusters etc....
An example of application for this subroutine is shown in table 4.

subroutine: twobits	
Input: shower strips (16)	Output: oiw(..) vectors
sho()="0001110000110000"	o1w()="0000000000000000" o2w()="000000000100000" o3w()="00010000000000" o4w()="0000000000000" o5w()="000000000000" o6w()="00000000000" o7w()="0000000000" o8w()="000000000"

Table 4: *Input and output to the subroutine* **twobits**. *Inputs are the shower strips from swath xx, outputs are vectors of cluster address for 1, 2, ...8 strip wide clusters. A component is "1" for the 1st strip in the cluster, "0" otherwise*

This procedure does not take more than 10 ns.

- **merge_4doublets**: Merge clusters found in four adjacent doublets. One cluster maximum can be found per strip doublet. A maximum of 4 clusters per octuplets thus can be formed. the "merging" operation keep only one of these (maximum) four. The "merging" of clusters is such that the first cluster found is taken, which has the highest strip index. This implies that low η clusters are favoured within an octuplet.
  Cluster Address and width are computed and formatted into an 12-bit word (3 bits for the width, 9 for the address) and then returned to the main routine.
  This subroutine is called $int(N_{doublet}/4)$ times if $N_{doublet} > 3$ All calls to this subroutine are processed in parallel, each call taking less than 10 ns. Example is given in table 5.

- **merge_2clusters**: if needed, one merges clusters found in adjacent 8-strip wide areas, following the same procedure as the one described before (keep lowest η cluster). Example of input/ouput to this subroutine is provided in table 6. Each call is, again, processed in parallel for cluster pairs. The subroutine takes less than 5-8 ns per call.

- **form_cluster**: Form the two output clusters found in the swath. For each cluster, the center of the window inthe MIP layer is computed (integer operation) and the MIP

subroutine: merge_4doublets	
Input:	Output:
logic_01 = '0'	logic = '1'
logic_02 = '1'	cluster = "010 00001011"
logic_03 = '1'	
logic_04 = '0'	
rel_add_01 = '1'	
rel_add_02 = '1'	
rel_add_03 = '1'	
rel_add_04 = '0'	
doublet_index_01 = 8	
doublet_index_02 = 9	
doublet_index_01 = 10	
doublet_index_01 = 11	
width_01 = "000"	
width_02 = "010"	
width_03 = "011"	
width_04 = "000"	

Table 5: *Inputs/Ouputs to merge_4doublets. Logic_01 and Logic_02 are '1' if a cluster has been found in the doublet, '0' otherwise.If Logic_01, _02, _03 or _04 are '1', "merge" keeps the 1st cluster, ie: the one with the lowest* η. *rel_add_0i is the relative address of the cluster found in the doublet (if logic_0i = '1'), width_0i its width and doublet_0i is the absolute location of the doublet in the module (from 1 to 72). Note that _01 index ALWAYS corresponds to the lowest* η *cluster by convention. Note also that the cluster is formatted at this stage as: wwwaaaaaaaa, where w=width, a=address.*

subroutine: merge_2clusters	
Input:	Output:
logic_01 = '1'	logic = '1'
logic_02 = '1'	cluster = "01100010001"
cluster_01 = "01100010001"	
cluster_02 = "00100011100"	

Table 6: *Inputs/Ouputs to merge_2clusters. Logic_01 and Logic_02 are '1' if a cluster has been found in the octuplet, '0' otherwise. If both Logic_01 and _02 are '1', "merge" keeps the 1st cluster, ie: the one with the lowest* η. *Note that _01 index ALWAYS corresponds to the lowest* η *cluster by convention.*

bit pattern (7 bits) added to the 12-bit word defined in the previous routine. Type, Orientation, Threshold bits are set at this point. The format used is defined in section 4.2. This is the subroutine that takes most time, about 30-40 ns for two clusters.

4.3 Routine to be implemented. Updates and adaptations

This code must be updated to treat the overlap between two adjacent swaths. By construction, one cluster can be seen in two adjacent swaths, given the size of the overlap. The existing code uses the subroutine **"compare"** which is designed to compare the location of the cluster candidates (after unpacking their address and their width) and check that they are not identical. So far, the routine returns a flag indicating wether the cluster is dupplicated or not. At the present stage, clusters are kept in each individual swath. Procedures have been envisaged to prevent the same cluster to be selected: one could calculate the distance from the cluster center to the two boundaries of the 2 adjacent swaths, and locate the cluster in the closest swath.

This code can be easily adapted to new changes:

- update the code as function of the needs and new requirements [9]. Basically, the number of output clusters generated per sector and per threshold can be easily reduced if needed (for eg. to 6) by modifying the output part of all **subsector_0i_uh.vhd** surboutines. On the other hand, if one wants to increase the number of clusters per sector, we can just skip calls to the **Merge_2cluster** subroutine and increase the number of returned clusters to

- pack the DB L1 and L2 data using the format defined with M. Martin [7]. This has to be done in the **fps_sector_uh**

- Merge High and Low threshold clusters (for both u - and v separately) so that only 16 u and 16 v clusters (H+L) total are generated per octant (= two sectors), or that 8 clusters (H+L) are generated per sector. A new "layer" has to be added to the code, that will use the 12 clusters generated by **fps_sector_uh** and **fps_sector_ul** as input and merge them into a set of 16 total clusters. Final design is still in discussion [9] at that time

4.4 Simulation of the Digital Board algorithm

Simulations are performed using ALtera MAX-PLUS-II.

4.4.1 Resources needed

Inputs to the algorithm are the High threshold shower u channels (144) together with mip u channels (103). The generated output is 12 clusters maximum (2 per η-range) and must be extended to 16 for the entire sector.
Resources needed to form a maximum of 2 clusters per η-range are reported in table 7. The number of Logic Cells used is function to the complexity & number of logical operations that are used in the algorithm. The device suggested using Altera products are indicated in the first column.

Device (chips)	Input Pins	Output pins	Memory utilized	LC's #	LC's %
EPF10K100ABC600-1	202	191	0%	4448	89%
EPF10K100AFC484-1	155	171	0%	3664	73%
TOTAL	357	362	0%	8112	81%
User Pins:	247	288			

Table 7: *Resources needed using the fps_sector_uh routine. Inputs are the u channels, High threshold (144+103=247). Output are 12 clusters of 24 bits in this design (12×24=288)*

The total resources needed for a complete module (u + v, H+L threshold) are therefore estimated to:

$$8,112 \times 2(u+v) \times 2(H+L) = 32,448\ LC's$$

4.4.2 Timing estimates

Timing for this algorithm is estimated using Altera simulation. As swaths routines run in parallel, the overall timing is dominated by the routine applied to the widest swath. Actually, the time needed to form, count and merge clusters in each swath takes less than 100 ns, independetely of the swath. Fig.4.17 displays an example of the timing simulation for a current design that provide 12 clusters as output per sector, threshold, and orientation (u,v). The time estimate is below 105 ns in the worse case.

5 Conclusion

An algorithm has been developped for the L1 trigger algorithm that should be implemented in the Digital Board. The current design forms and counts up to 12 the cluster per sector, with at least one cluster and at maximum 2 clusters per η-bin. This current design has been simulated using Altera. Resources have been established in terms of Logic Cells, and the

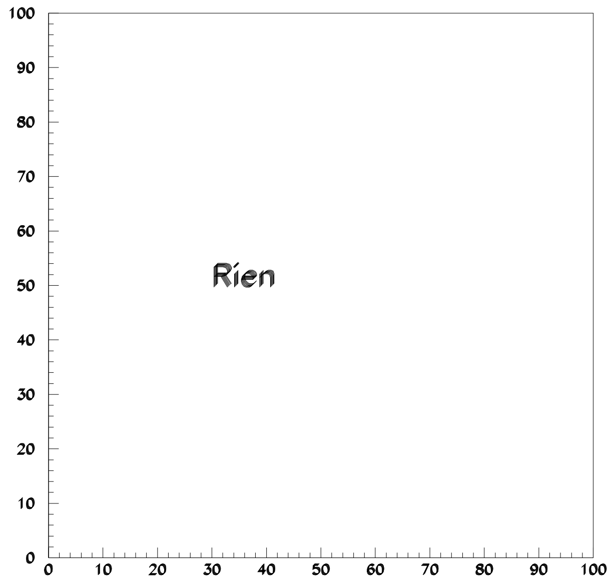

Figure 4.17: *Example of time simulation for the design described in the text.*

algorithm has been timed, and found to take no more than 105 ns to generate these outputs. What remains to be done is:

- update the code as function of the needs and new requirements [9].
- pack the DB L1 and L2 data using the format defined with M. Martin [7].
- Merge High and Low threshold clusters (for both u - and v separately) so that only 16 u and 16 v clusters (H+L) total are generated per octant (= two sectors), or that 8 clusters (H+L) are generated per sector. Final design is still in discussion [9] at the present time

A VHDL files location

The algorithm may be find on dpsb04 at the following location:

$$User\ Disk\ D : lucotte/2.FPGA/Version\_200/Version\_202$$

B VHDL files listing

subsector_01_uh	twobits / searbit merge_4doublets merge_2clusters form_01_clusters
subsector_02_uh	twobits / searbit merge_4doublets merge_2clusters form_02_clusters
subsector_03_uh	twobits / searbit merge_4doublets merge_2clusters form_03_clusters
subsector_04_uh	twobits / searbit merge_4doublets merge_2clusters form_04_clusters
subsector_05_uh	twobits / searbit merge_4doublets merge_2clusters form_05_clusters
subsector_06_uh	twobits / searbit merge_4doublets merge_2clusters form_06_clusters

Table 8: *Files used for the project fps_sector_uh. the extension .vhd is implied.*

References

[1] "Definition of a L1 e and photon trigger for the FPS", DØ note 3493

[2] "J/Psi Trigger in the forward region", DØ note 3566.

[3] "Definition of L2 trigger for the central and forward preshower", DØ note xxxx, M.Bhattacharjee and P. Grannis

[4] "FPS Detector Overview and Parameters", http://www-d0.fnal.gov/ abid/fps_overview.html

[5] "TDR for the Forward PreShower", DØ note 3445.

[6] Paul Grannis web page: $http://sbhep1.physics.sunysb.edu/\ grannis/sift9-99.prn$

[7] "The CTT System, Communication protocols and Data encoding" M. I. Martin $http//d0server1.fnal.gov/users/manuel/Protocols/Final\_version/final_FPS.pdf$

[8] Private communication with M. Bhattacharjee. My test files are on: /prj_root/786/mub_3/RunIIB/lucotte/3.sng_ele_trigger/

[9] M. Martin, M. Bhattacharjee, private communication