LHCb 2003-036
CALO
23 May 2003

Functional specification of the PGAs for the ECAL/HCAL Front-End card

Christophe Beigbeder, Dominique Breton, Olivier Callot, Daniel Charlet,

Olivier Duarte, Jacques Lefrançois, Frédéric Machefert

Laboratoire de l'Accélérateur Linéaire - Orsay

ABSTRACT

This note specifies the functionalities of the PGAs on the ECAL/HCAL Front-End board. The signal treatment for pedestal subtraction and gain correction is described. The scheme used to produce the trigger information, and to store the data during Level-0 latency is also detailed.

1 General overview

The Calorimeter Front-End card handles 32 channels. This note concentrates on the digital part of the card, from the output of the ADC to the readout. It also describes the way the trigger data are processed.

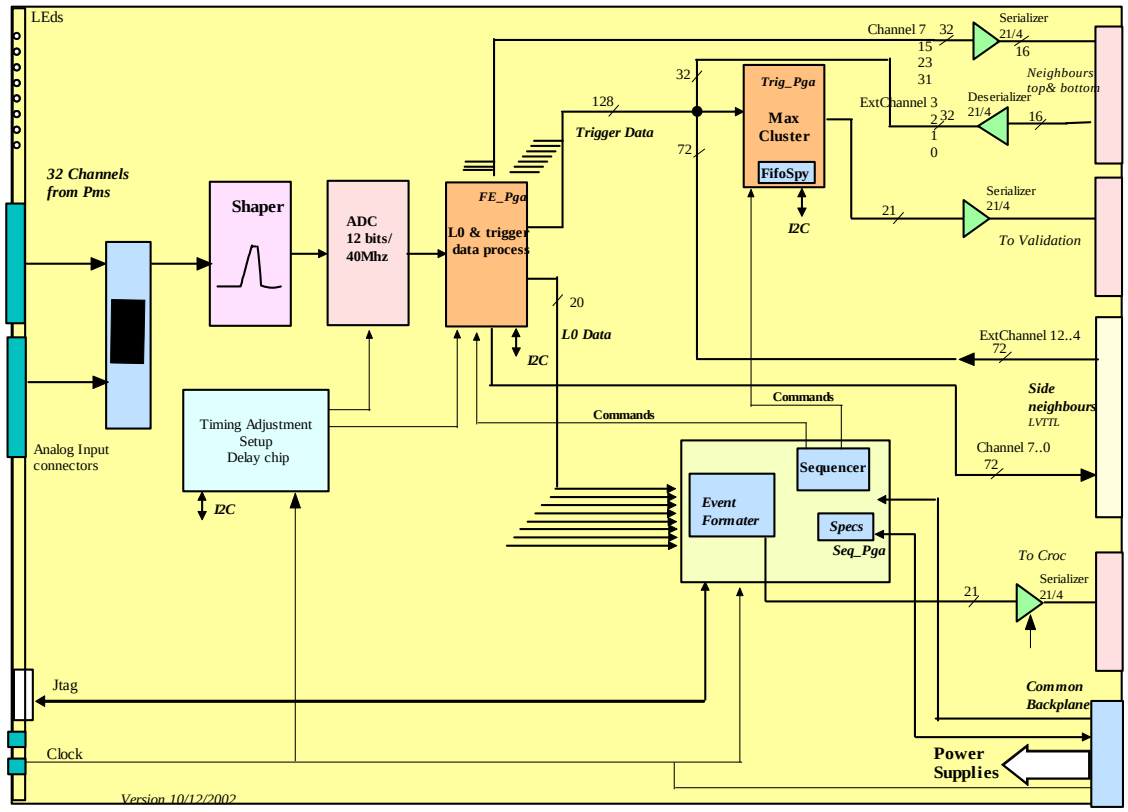

Figure 1 Block diagram of the FE card.

As shown on Figure 1, one can identify 3 major components:

1. The FE-PGA in charge of processing 4 channels. It consists of four functional blocks: Pedestal processing, trigger processing, storage and retrieval after L0 and test handling. 8 identical copies will exist on the card. Two blocks of memory per channel are used in each FPGA for the Level-0 pipeline and derandomiser

2. The Seq-PGA in charge of the global control of the card, mainly building of the data block after L0-Yes, and sending them to the CROC. It also issues control and synchronisation signals for the other 8 FPGA. .

3. The Trig-PGA, which handles the processing for the production of the L0 information.

This note will describe the details of the functions implemented in these components. The target is to use ACTEL AX family PGAs that are SEU immune. All registers holding permanent information should also be protected by triple voting techniques.

The FE and Seq PGA are working in close collaboration. Their main functions, and the relevant connections, are shown on Figure 2 and Figure 3

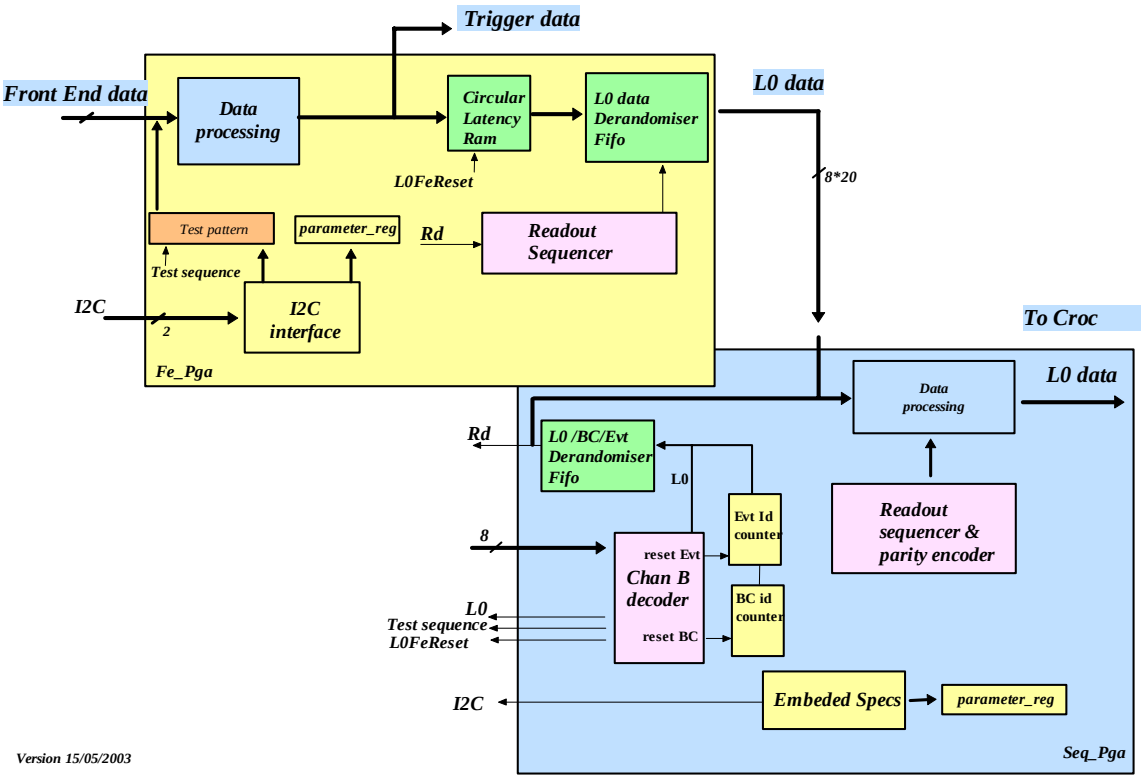

Figure 2 Global view, FE–PGA and Seq-PGA

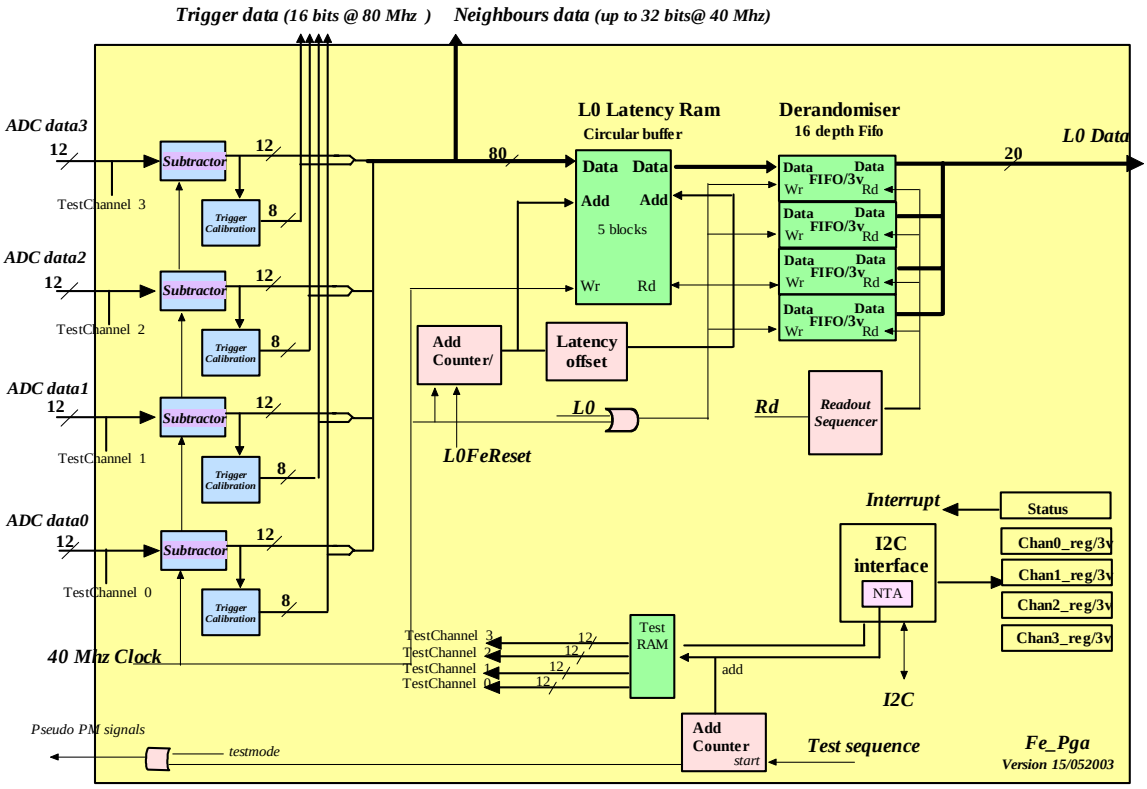

Figure 3 Front end FPGA

2 The Front-End PGA

This FE-PGA has four functional blocks. The first one is to process the input ADC data, which needs to be re-synchronized (each ADC has its own clock), and processed, to remove the low frequency noise and to subtract the pedestal. The second block produces the trigger data, converting the 12 bits ADC to 8 bits. The third block is in charge of storing the data (40 bits per FE-PGA) during the L0 latency, and to send them on L0-Yes to the Seq-PGA. The last block allows injecting test values at the input, in place of the ADC, to check the proper behaviour of the card.

2.1 ADC processing

The first step is to re-synchronize the ADC input. Each ADC (12 bits) arrives with its own time delay; it should be phased so that all inputs from the same event become fully synchronous. The range of time is in principle ± 6 ns, but the maximum possible range is ± 12 ns, close to a full cycle. Care should be taken to sample at a time where the signal is stable. This can be done using either the normal clock or an inverted clock, selecting the proper one on a channel per channel basis.

Then one wants to suppress the low frequency noise. We will implement two options for this suppression

1. Standard method: One subtracts the smallest of the previous two measurements. The idea is that a channel has a low occupancy, and then the probability that there was a signal in both previous crossing when there is some signal in the current one is negligible. It has been shown that this underestimates the pedestal (one selects the low fluctuations) by 0.42 times the pedestal width, which means about half a count.

2. Improved method, to avoid brutal fluctuations. The estimated pedestal is the lowest of the previous pedestal plus a constant, and of the previous measurement. The constant is small, between 1 and 3 counts. Using 2 counts as pedestal step, the shift of the pedestal is less than .1 counts for a nominal sigma of 1.5 counts.

Simulation shows that the second method is better. The measured sigma is lower than with the other method by .2 counts. When adding an exponential signal with a width of 100 counts and a frequency between 1 and 10 %, the degradation of the sigma is small, less than 15 %, while it starts to be huge for the standard method, as the case of 2 consecutive signals starts to become relevant. However, as the second method may not follow fast enough upwards variation of the pedestal, and more generally depends on a longer history of the signal, we propose to implement both methods, with a control switch, and a parameter on 2 bits for the second method.

For testing purpose, it may be useful to not subtract any data at all, to read the raw ADC data. This is mainly for debugging, and is obtained by forcing a 'reset' on the register storing the pedestal.

After correction, the signal can be negative. We will store the value+256, saturated at 0 and 4095. This means that the effective range of pedestal-corrected ADC value is –256 to 3839.

The schema for this processing is show in Figure 4 where several control register are clearly shown: "Clkinvsel" to select if the inverted clock is used, "Oldsubmod" to select the two modes of pedestal subtraction, "Thresh" which contains the increment for the second method, "Sub

zero" to avoid subtracting any pedestal. The top right part of the figure contains a subtractor by 256 with negative value saturated to zero to restore a range 0-3839 required by the trigger part.

All these constants are loaded by the ECS and protected by Triple Voting Register (TVR)

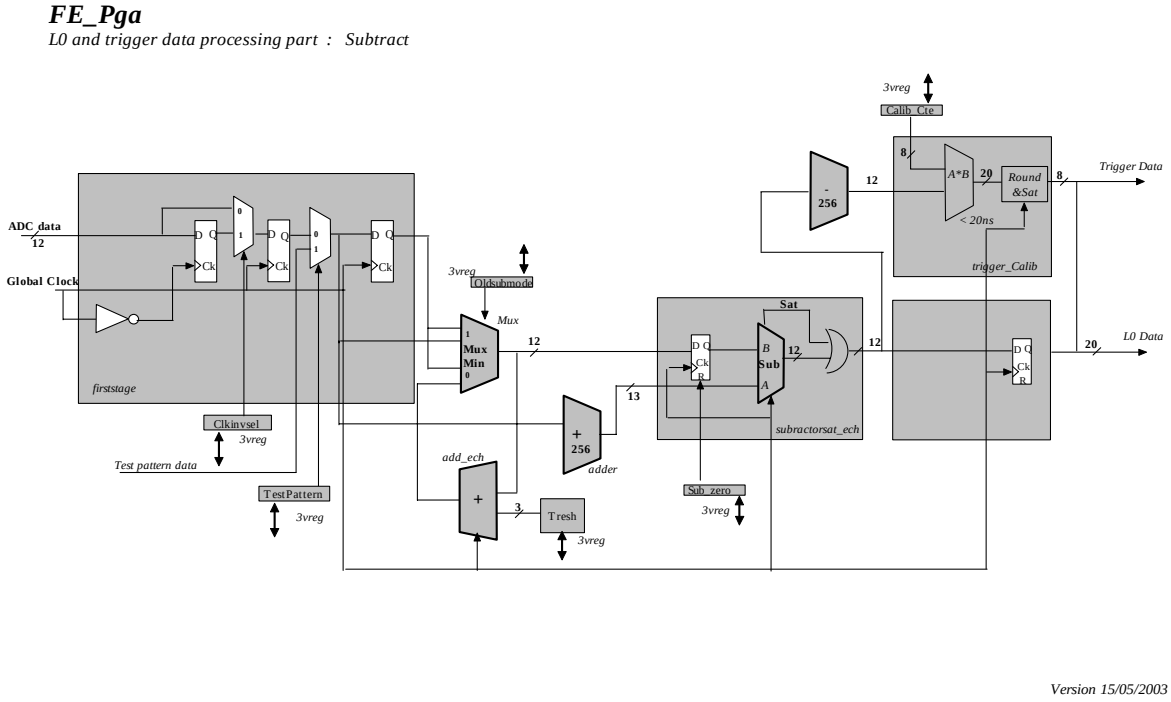

Figure 4 ADC data processing

2.2 Trigger processing

The purpose of this processing is to convert the ADC data to 8-bit E_T with saturation. The ADC will have a full scale around 10 GeV E_T with variations from channel to channel due to the gain of the PM, but one wants calibrated data with saturation at 5 GeV E_T for the trigger. We had foreseen a Look-Up Table for that in the initial draft of the card, but this is no longer acceptable as it is not SEU immune. However, the linearity is good enough so that we need only to multiply the (pedestal corrected) ADC value by a number. And we don't need a very accurate calibration.

The proposed solution is to multiply the 12 bits by an 8 bits number, and select the appropriate bits in the result such that a well-calibrated channel (4095 = 10 GeV E_T) will produce a calibrated trigger signal (5 GeV E_T = 255) for a calibration constant of 128. We can then recover a factor between 0.5 and 2 in gain, without loosing too much in precision. The exact method is the following:

1. The 12 bits ADC data should be put back with a true zero: This is done by subtracting 256 from the final result, and applying saturation. The input is then in the range 0-3839.

2. Lets write the input as ABCDEFGHIJKL where each letter is one bit. Lets write the coefficient ZYXWVUTS with the same convention. One can write the multiplication as a big addition of shifted number, multiplied by one bit:

```
          ABCDEFGHIJKL  *S
         ABCDEFGHIJKL   *T
        ABCDEFGHIJKL    *U
       ABCDEFGHIJKL     *V
      ABCDEFGHIJKL      *W
     ABCDEFGHIJKL       *X
    ABCDEFGHIJKL        *Y
   ABCDEFGHIJKL         *Z
-------------------------
abcdefghijklmnopqrst
```

3. One could perform the multiplication directly, however it might be more efficient to do it as additions since one does not need the full accuracy.

4. The first step is to perform the 4 additions of the lines two by two (1+2, 3+4, 5+6, 7+8) (The result of these 4 additions is the same when S&T or U&V or W&X or Y&Z are =1 only one adder "FirstAdder" is needed), then the two additions of the four results two by two (1+2+3+4, 5+6+7+8), and then the final addition. The result of the first 4 additions is chosen by 12 4-entries multiplexers as zeros, the original, the shifted number or the result of "FirstAdder" depending on whether S,T,U,V etc... are 0 or 1, thus decreasing the needed number of adders. One doesn’t need to keep all the bits, as the final result is the part “cdefghij” of the 20 bits mathematical result. In fact keeping 12 bits for all intermediate results is enough. At the end, one should round the number (if the first dropped bit “k” is one, increment the result) and then saturate at 255 if “ab” is non-zero.

The resulting 8 bits are sent towards the Trig-PGA, multiplexed at 80 MHz to decrease the number of pins on this PGA, and also at 40 MHz for those channels that have to be sent to neighbouring cards. These 8 bits are also joined with the (delayed) 12 bits data to form the 20 bits information we want to keep for this channel. The delay is short, (one or two clock cycles, simulation ongoing).

2.2.1 Masking channels

It may happen that a channel produces bad data, permanently or intermittently, which may affect the trigger by causing spurious trigger at high rate. It is very simple to mask such a channel, just by setting the gain parameter to zero. Then the signal sent to the trigger PGA is always zero, and the channel is ignored in the trigger.

2.3 Data storage

THIS FUNCTION CAN BE IMPLEMENTED, IN PRINCIPLE, IN AN IDENTICAL WAY IN THE PS/SPD CARDS

Each FE-PGA produces 80 bits of data at 40 MHz. The format of the stored data is described in reference [1].

These data will be stored in the AX memory blocks. The blocks can be configured to be 256 deep and 18 bits wide, 256 deep is more than sufficient for the Level-0 latency which is 160, 5 blocks offering 90 bits of data memory will be sufficient. The derandomiser is a FIFO 16 deep, the memory blocks will therefore be configured for this function as 128 deep time 36 wide, 4 blocks will used one per each channel, since the channels are readout one after the other. A total of 9 blocks will therefore be needed out of the 12 blocks of an AX250.

As can be seen in the picture the write address of the latency buffer can be reset by a signal provided by the sequencer FPGA. The sequencer FPGA also distributes the L0 trigger and the read command that commands the readout of the derandomiser.

2.4 Test control

The idea is to inject a known pattern of 12 bits in place of the ADC input, very early in the processing. The choice of ADC or test pattern is done by a multiplexer controlled by one bit of the channel registers protected by triple voting, each channel can therefore be selected individually to be in the pattern mode or in the normal ADC mode. Three blocks of memory of the AX250 remain, which can be configured as 256 addresses x 3x18 bits of data this is enough to generate 256 successive data with 4X12bits simulating the output of the 4 ADC. The required pattern is loaded in the block memories by ECS through theI2C interface There is no need to protect the pattern from SEU by triple voting since either the test are done without beam or the loading of memory just precede the test. The start of the 256 value test sequence is controlled by a test sequence command issued by the sequencer; in this way the pattern are synchronized between all the FPGA that are in the test pattern mode.

Finally another test can be done by injecting through a 4 channel 74F125 (Quadbuffers used as switches) and small capacitors a test charge at the input of the amplifiers. The command pulse is derived from the test sequence pulse from the sequencer FPGA; a register loaded by ECS in each front end PGA choose whether the test sequence pulse drives this amplifier test or the test pattern described above. In this test all 32 inputs of the card are tested simultaneously.

3 The Sequencer PGA

AGAIN THIS PART IN PRINCIPLE CAN BE IDENTICAL IN THE PS/SPD CARD

This single PGA is in fact mainly handling the L0-Yes and the channel-B broadcast of the TTC. Its functionality is to collect the 32 data words coming from the FE-FPGA, prefix them with the header, and compute longitudinal and vertical parity (trailer). The 21-bits-word data are then sent one after the other to the CROC. In order to do so the 21 bits are serialized further at 280 MHz on 3 lines plus one clock using a DS90CR216

3.1 Handling the L0-Yes

The process upon reception of an L0-Yes is similar to the one taking place in the FE-PGA.

- Upon reception of a level0-yes the current BC-ID and the level0 ID are put in a FIFO used as a LEVEL 0 derandomiser. This will label the event and will be placed in the first header word. This BC-ID is a 12 bits counter with triple voting. The 12 bits level0-ID is incremented just after that; it is also a triple voting counter.

- When the FIFO is not empty, the first word is extracted, the anti-parity is computed (not the parity to avoid words with 21 zeros) and the complete 21-bits word is sent to the CROC.

- The 32 data words arrive one after the other, on the 8 input busses. Each 20-bits word is parity encoded and sent to the CROC

- After the last word, the vertical anti-parity (sum of the bits of the same rank in the 33 previous words) is computed, anti-parity encoded and sent to the CROC

- The CROC should be warned that useful data is coming through its input deserialiser. In order to do this, a word with the 21 bits at Zero is sent after the 34 word of an event. In this way the CROC PGAs will be able to recognize the header as the first non-zero word after one Zero word (during the 33 words after a header the header recognition system is blocked)

The Seq-PGA is then ready for the next L0-Yes, which may be already in the FIFO.

3.2 Handling TTC broadcast information

The Seq-PGA handles the full channel-B broadcast, and then decodes what is needed. This includes

- The beam crossing number reset, which resets (in fact presets to a fixed value) the BC-ID counter.
- The Level-0 Front-End reset
- The Level-1 Front-End reset
- The start of test sequence. This can be used to control the cycling of the test memory described in section 2.4 .

4 The Trigger PGA

The function of this PGA is to compute the 32 sums of 2x2 cells, using the 32 channels of the card and 13 channels from neighbouring cards, namely 9 'horizontal' neighbours connected via the backplane and 4 'vertical' neighbours connected by cables, coming from a nearby crate. The card sends also 9 'horizontal' neighbours on the backplane to the next card, and 4 'vertical' neighbours on cable to the next crate. These connections are described for example in [2]. As indicated in Figure 3, the 32 inputs from the card are received on 80 MHz multiplexed lines, that have to be de-multiplexed, while the data from neighbours are received as 40 MHz signals.

A first step of the processing is to wait for all the information of the same event to be ready. As some data arrive by cable or by the backplane, the appropriate wait (pipe-line mode) has to be added on the local channels, and a different one on the backplane channels, faster than the (serialized) cable connection. We will use the memory banks of the AX to implement these pipelines, with a read pointer obtained by adding a constant offset to the write pointer, the constant being protected by triple voting. This insures that the data is correct, or wrong only during a few cycles if the pointer is modified by an SEU. This is schematised in Figure 5

Trig_PGA : Functional blocs in ACTEL AX500

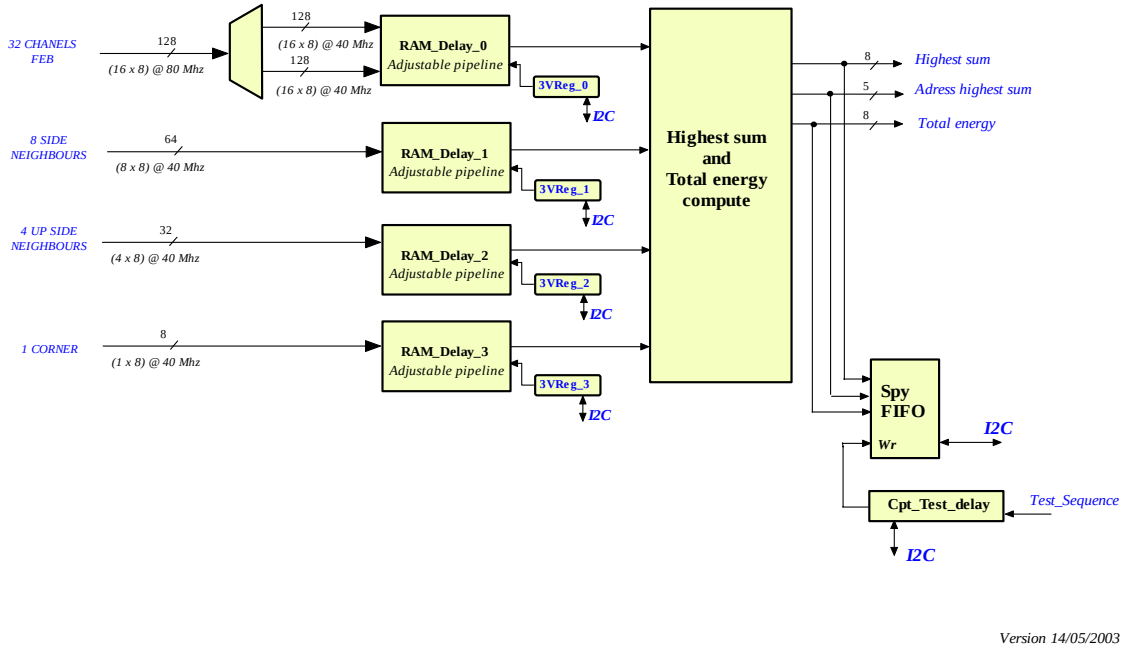

Figure 5 Functional block diagram of the Trigger PGA

The second step is to compute the 32 sums of 2x2 cells. The sum is saturated, i.e. if the result would overflow it is forced to stay at 255. This takes part of one clock cycle. The next step is to select the highest of the 32 sums. A binary search is performed; first 16 comparators select the highest in their two inputs, keeping track of which one was selected. Then the 16 results are compared two by two by 8 similar comparators; the results are compared by 4, then 2, and finally by a last one. 5 steps are needed, and a 5-bit address is produced, to keep track of which sum was the highest. This is shown on Figure 6.

Trig_PGA : Highest sum and Total energy compute

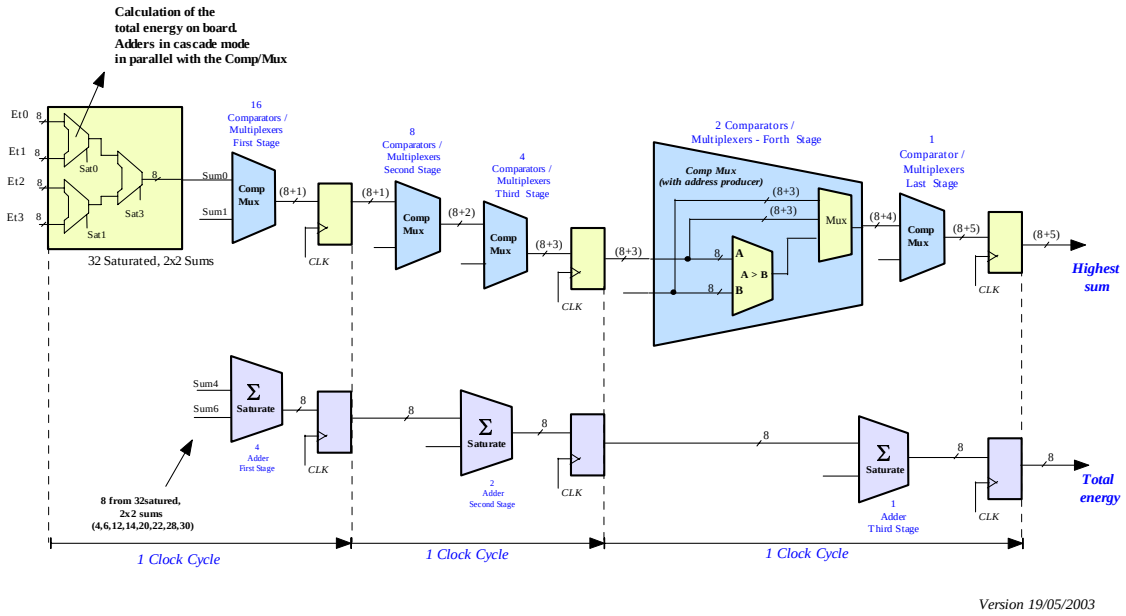

Figure 6 Computation of the highest cluster and total energy

In parallel, the total sum of the 32 channels is computed. As we have already made the 2x2 sums, it is enough to add 8 well selected partial sums, using four saturating adders with 2 inputs, followed by two adders and a last stage to get the total sum on 8 bits. This sum is computed at the same rhythm as the highest cluster, so that it is ready at the same time, and can easily be put in the same output word.

We have then the final result, 5 bit address, 8 bits highest, 8 bits sum. This is then sent to the Validation Card via the backplane as 21 bits on a serialised link. The address and highest candidate, together with 8 bits of BC-ID, are sent on two serialized cable connectors. One connection is used to send the address of the highest candidate to the Preshower card (ECAL cards), both connections are used to send the HCAL candidate to two ECAL Validation Cards (HCAL cards).

A FIFO is also implemented in this PGA, to log the output. This will permit to check the correct behaviour of this part of the card, using pre-loaded inputs as described in section 2.4 .

5 References

[1] O. Callot, “*Proposed Online Data Formats for the Calorimeters*”, Note LHCb 2003-031

[2] Ch. Beigbeder et al, “*The trigger part of the Calorimeter Front-End Card*”, Note LHCb 2003-037

.