

DSP Online Algorithms for The ATLAS TileCal Read-Out Drivers

A. Valero, J. Abdallah, V. Castillo, C. Cuenca, *Member, IEEE*, A. Ferrer, E. Fullana, V. González, *Member, IEEE*, E. Higon, J. Poveda, A. Ruiz-Martínez, B. Salvachua, E. Sanchis, *Member, IEEE*, C. Solans, J. Torres, and J. A. Valls

Abstract—TileCal is the hadronic tile calorimeter of the ATLAS experiment at LHC/CERN. The central element of the back-end system of the TileCal detector is the Read-Out Driver (ROD). The main components of the TileCal ROD are the Digital Signal Processors (DSPs) placed on the Processing Unit (PU) daughterboards. This paper presents a detailed description of the code developed for the DSPs. The code is divided into two different parts: the first part contains the core functionalities and the second part the reconstruction algorithms. The core acts as an operating system and controls configuration, data reception and transmission and synchronization between front-end data and the Timing, Trigger and Control (TTC) information. The reconstruction algorithms implemented on the DSP are the Optimal Filtering (OF), Muon Tagging (MTag) and Missing E_T calculation. The OF algorithm reconstructs the deposited energy and the phase of the signal for every calorimeter channel within a front-end module. This reconstructed energy is used by the MTag algorithm to tag low transverse momentum muons that may escape the ATLAS muon spectrometer Level 1 trigger whereas the Missing E_T algorithm computes the total transverse energy and the projection on X and Y axis for the entire module that will be used by the Level 2 trigger system.

Index Terms—ATLAS, digital signal processors, optimal filtering, read-out driver.

I. INTRODUCTION

ATLAS is a general purpose pp detector of the four particle detector experiment for the LHC. TileCal is the hadronic tile calorimeter of the ATLAS experiment and it is made out of steel as the passive medium and scintillating tiles as the active material [1]. The Digital Signal Processor (DSP) is the main component in the TileCal Read-Out Driver (ROD) [2], which receives data validated by the first level trigger. The DSPs are responsible for data reconstruction in real time at the ATLAS first level trigger rate (100 KHz) [3]. The DSP has to compute energy, phase and Quality Factor (QF) for all the channels in less than 10 μ s at the ATLAS maximum rate and send the reconstructed data to the second trigger level. There are two Processing Unit (PU) per ROD and two DSPs per PU.

Manuscript received May 10, 2007; revised November 30, 2007. Paper no. TNS-00192-2007.

A. Valero, J. Abdallah, V. Castillo, C. Cuenca, A. Ferrer, E. Fullana, E. Higon, J. Poveda, A. Ruiz-Martínez, B. Salvachua, C. Solans, and J. A. Valls are with the Institut de Física Corpuscular (IFIC), Edifici Instituts d'Investigació, Paterna, 46071 Valencia, Spain (e-mail: jvalero@cern.ch).

V. González, E. Sanchis, and J. Torres are with the Department of Electronic Engineering, University of Valencia, Burjassot, 46100 Valencia, Spain

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TNS.2007.914019

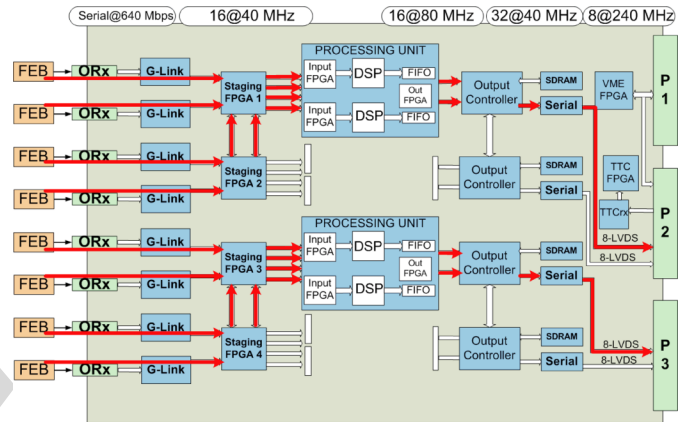


Fig. 1. TileCal read-out driver dataflow.

The PU daughterboards architecture allows future upgrades by increasing the number of PUs per ROD up to four or by redesigning the PU. Each DSP processes the data coming from two Front-End Boards (FEB). Besides, the DSP synchronizes the front-end data with the Timing, Trigger and Control (TTC) information [4] received in the ROD through the Trigger and Busy Module (TBM) [5]. Finally, some other algorithms, histogramming and commands are also processed by the DSP.

II. HARDWARE SYSTEM OVERVIEW

A. Tilecal Read-Out Driver

The back-end hardware for the first level trigger and Data Acquisition (DAQ) of TileCal consists of four ROD crates [2]. Each ROD crate contains eight RODs and reads out one out of four partitions in the calorimeter. The RODs are custom 9U VME64x boards and are equipped with two PU pluggable daughterboards (Fig. 1).

The ROD has 8 input links which provide an input data bandwidth of 5.12 Gbps while the output data bandwidth is 2.56 Gbps... Hence, the data have to be reduced at ROD level by compressing the information.

The data coming from eight FEBs are received at ROD level through the Optical Receivers (ORx) (Fig. 1). Then, the Staging FPGAs route the data to the corresponding PU. In the PU, the data are received through two Input FPGAs where the data are stored and transferred to the DSPs. Therefore, each DSP processes the data coming from two FEBs.

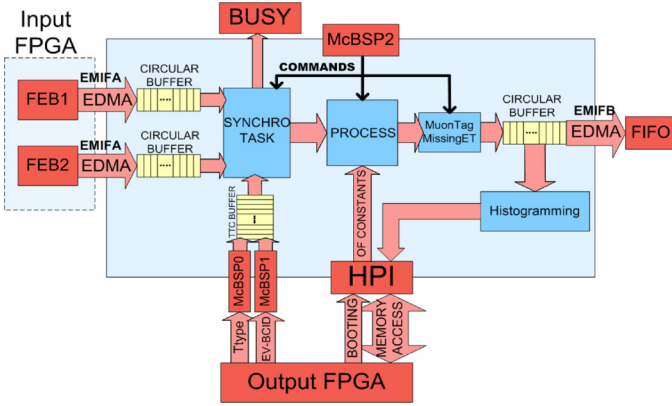


Fig. 2. DSP hardware structure.

B. Processing Units

The PUs of the TileCal ROD have two Texas Instruments TMS320C6414 DSPs [6], which provide an instruction cycle frequency of 720 MHz, 1024 KB of user memory and an interrupt latency of 900 ns. Besides, the PUs are also equipped with two Input FPGAs to check and transfer the input data, two FIFOs to store the output processed data and an Output FPGA to receive the TTC information and to provide the interface with the VME bus.

When an Input FPGA receives a complete event it sends an interrupt to the corresponding DSP and the whole event is transferred to the DSP input buffer. The data transfer between the Input FPGA and the DSP is performed through the External Memory Interface A (EMIFA) [6]. This is configured as synchronous memory interface and the width of the bus is 64 bits. This transfer is clocked at 100 MHz. There are two different interrupts from each Input FPGA in order to indicate which FEB is the data source (Fig. 2). The DSP Enhanced Direct Memory Access (EDMA) stores the received events in two circular buffers, one per FEB. These input buffers store up to 16 events. When the buffer is full it stores the next event received in the first position. Once the event is reconstructed, it is copied to the DSP output circular buffer and transferred to a FIFO placed in the PU. The transfer between the DSP and the FIFO is handled by the EMIFB. In this case, the EMIFB has a 16-bit bus width, and it is clocked at 100 MHz.

The TTC information is received at PU level through the Output FPGA, which provides the communication with the TTC FPGA. The TTC information is then transferred to the DSP through two Multichannel Buffered Serial Ports (McBSP) (Fig. 2). The McBSP0 is used to receive the Bunch Crossing Identification (BCID) and the Event Identification (EVID), whereas the McBSP1 is used for the Trigger Type (Ttype). The McBSP2 is used to receive commands and to read-out internal registers.

Finally, the Host Port Interface (HPI) of the DSP is used to boot the DSP code and to read the internal memory and histograms while the system is running (Fig. 2). This information is accessible through the VME bus.

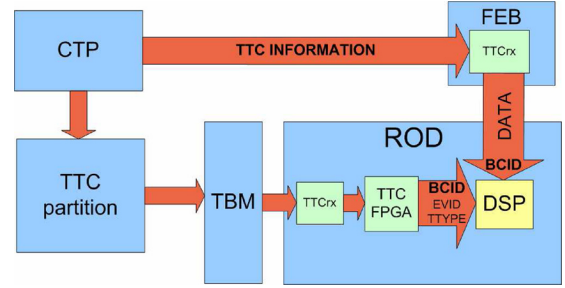


Fig. 3. Distribution of the TTC information.

III. INPUT DATA

A. Front-End Data

Each FEB has two inputs to receive the TTC information and the Tile Data Management Unit (DMU) data. The FEB mounts the event and transmits it through an optical link to the RODs [7]. The DSP receives the data from two FEB links.

Each event contains the data of 16 DMUs, 3 channels each, and additional control and consistency check words.

Each DMU block data includes a header, the samples of three channels and a trailer. The main information in the header is the BCID of the event and the gain of each channel. At DSP level this BCID is used to synchronize the data coming from front-end and the TTC information generated in the Central Trigger Processor (CTP) (Fig. 3).

B. TTC Information at DSP Level

The TTC information generated in the CTP is distributed to the front-end and to the TTC partitions (Fig. 3). Each TTC partition receives the TTC information in a Local Trigger Processor [4]. The TTC partition transmits this information through an optical fiber to the TBM, which distributes the information via the backplane to all the RODs in the crate. The RODs receive the TTC data in the TTC receiver (TTCrx) [8] chip where it is processed and then distributed to all the DSPs through the TTC_FPGA. TTC events are composed by a BCID, an EVID and the Ttype.

With this information, the DSP has to synchronize two BCIDs received within the data, and one BCID received through the TTCrx. If these three BCIDs match, the BCID, extended EVID and Ttype are included in the header of the event.

IV. CODE STRUCTURE

The DSP code is programmed in C using Code Composer Studio provided by Texas Instruments. Immediately after booting the DSP code the system is initialized. The EDMA ports, interrupt services and serial ports are disabled. Then, the DSP is configured with default values. During this configuration the busy signal is set to avoid reception of data. Then, the input and output buffers are mapped with the default event size. Once the default configuration has finished, it remains in an idle state until a command is received. At this point, new configuration commands can be asserted in order to reconfigure the DSP with non default values. The *start* command clears the busy signal and allows the reception of events. Then, if events

are received, the synchronization task is executed. The synchronization task sends the synchronized events to the process function. This function can either compute physics magnitudes or simply copy the raw data in the output data buffer. The DSP can be configured to compute the Muon Tagging and Missing E_T algorithms with the reconstructed data as well as to build histograms. Finally, the *send* task manages the transfer of each single processed event in the corresponding format. The DSP processes events until a *stop* command is received. Whenever an interrupt is received from the Input FPGA, the DSP holds the current process and services the interrupt. The received event is copied to its assigned position in the input buffer and the *busy* function is processed. Every time the synchronization task is processed, the busy status is also checked and updated. In this state the timer interrupts are enabled updating time information used by the TTC synchronization task. All these functionalities are described in more detail in the next sections.

A. Input and Output Buffers

Two input circular buffers (one per FEB) and one common output circular buffer are programmed in the DSPs. The maximum number of events inside these buffers is calculated in the DSP during configuration time and depends on the size of the events. The default value for typical TileCal events is set to 16.

In addition to the maximum number of events, when the *configuration* command is received or during the startup, the DSP computes the pointer address for the first event and the last memory address of the buffers. Each time a new event is received, the DSP also computes the pointer where the event has to be stored by adding the event size to the previous event pointer. If the previous pointer address plus the event size exceeds the buffer limit, the next event is stored in the first position.

Moreover, there are counters for the number of events received and sent, which are used for the busy management.

B. Commands and Internal Registers

With the DSP booted, it is possible to send commands through the VME bus to the DSP in order to change the configuration or to change the current status. The commands are received in the DSP through the McBSP2 port. The lower 8 bits correspond to the command field.

The *configuration* command sets the DSP processing variables such as event size, processing function, operation mode (1 or 2 FEBs per DSP), synchronization task as well as the activation of the Missing E_T and Muon Tagging algorithms and building of histograms.

The *prepare_for_run* command configures the variables of the run. This information, which is included in the header of each processed event by the DSP, is composed by the run number, data format version, source identifier, detector event type, sub-fragment header and the FEB identification for both links. However, before doing this, the previous values of these variables are copied by the Trigger and Data Acquisition (TDAQ) [9] software before starting the run to dedicated internal memory addresses.

During data taking it is possible to read out some status registers in order to extract some information regarding the DSP performance. These registers are read through the HPI and the

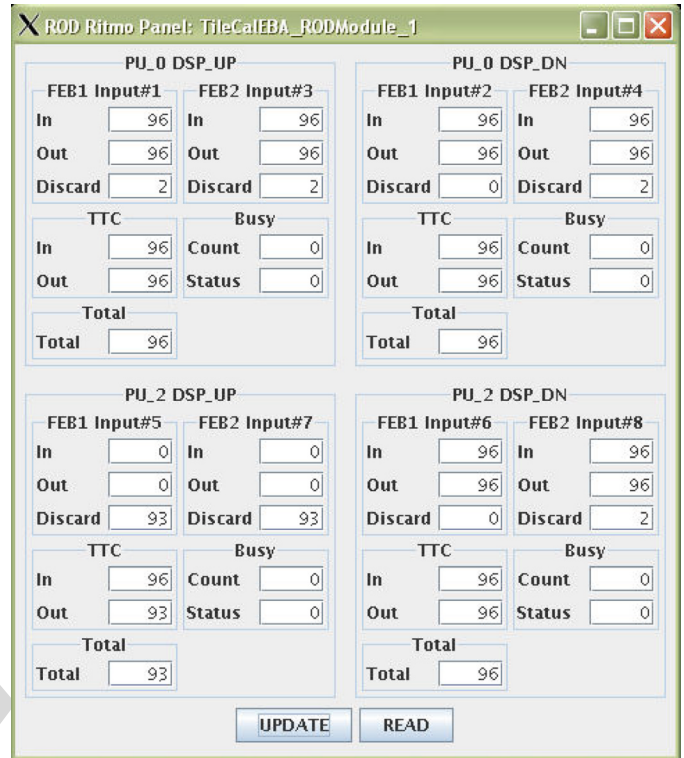


Fig. 4. Picture of the RITMO panel.

addresses of these registers are consecutive addresses starting from a base address. The *probe* command updates the content of these registers.

All this information is very useful to online monitor the detector performance and it is available in the TDAQ software RITMO panel (Fig. 4). The RITMO panel shows for each DSP the number of input and output events per link, the number of TTC events received and sent, the number of discarded events, the number of busy signals and the current busy status. For instance, it is possible find out if a module is not sending data or if it is sending wrong BCID with the data.

C. Busy Generation

In order to avoid losing events, the level 1 accept generation must stop when the circular input buffer is almost full with non processed events. If so, the ROD busy is set and it is cleared with hysteresis to avoid very short glitches in the busy signal.

D. Synchronization Task

The synchronization task is selected during the configuration time and it can only be changed by reconfiguring the DSP. There are basically two types of synchronization, depending on the TTC information availability at DSP level.

1) *Synchronization Without TTC Information*: In this case the DSP must synchronize the data of two input links. Each input event has the BCID in the DMU header. The DSP checks if the BCID on both input events are the same. If so, both events are processed. If there is a BCID mismatch, the DSP looks for coincidences in both input buffers. If it finds two events with the same BCID, these events are processed and the previous events are discarded. If no coincidence is found, the DSP processes

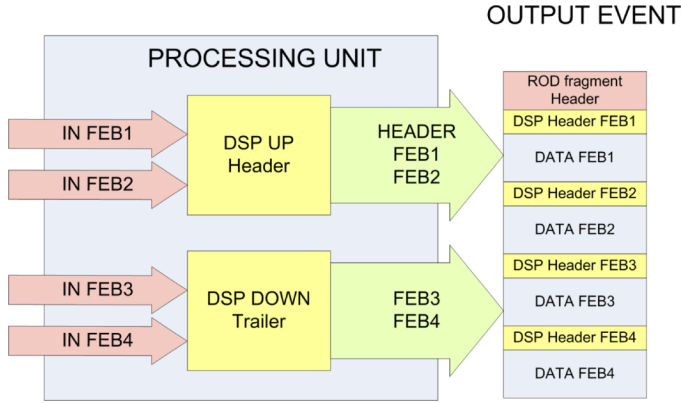


Fig. 5. ROD sub-fragment output data.

the first link and sends a null event instead of the second link. Furthermore, if one link stops sending data, the DSP sends null events to avoid blocking the acquisition.

In both cases, the TTC information included in the header of the output event is: the BCID received with the data, a default Ttype and the EVID is the number of events sent.

2) *TTC Synchronization*: If the TTC synchronization is selected, the DSP takes the TTC events as reference as they are assumed to be uncorrupted. Hence, the TTC events are always processed in the same order as received.

For every input event, the DSP checks the coincidence between the TTC event and both input links. If there is a coincidence, the events are processed. If there is no coincidence, the DSP tries to resynchronize both links searching in the events inside the input buffer. If no coincidence is found, the DSP tries to resynchronize only one of the input data links with the TTC data. In any case, if it finds a coincidence, the event is processed and the previous events are discarded. If one or both links stop sending data, the DSP will process the TTC event with null data events after a defined time. The time is given by the preceding timer interrupt, where the timer period is hard coded to 60 μ s.

E. Processing Task

The processing function is called from the synchronization task. If data and TTC are synchronized the *process task* is processed with the input events, whereas if there is no BCID coincidence the *process task* is executed with a null event. The *process task* executes the reconstruction algorithm which is selected at configuration time through the *configuration* command. If the copy mode is selected, the output data is exactly the same data received from the front-end. If the reconstruction algorithm is selected the output data (Fig. 5) is the reconstructed information for the energy, phase and QF.

1) *Optimal Filtering Algorithm*: The Optimal Filtering (OF) algorithm reconstructs the amplitude and phase of a digitized signal by a linear combination of its digitized samples, pedestal subtracted.

$$A = \sum_{i=1}^n a_i (S_i - p) \quad (1)$$

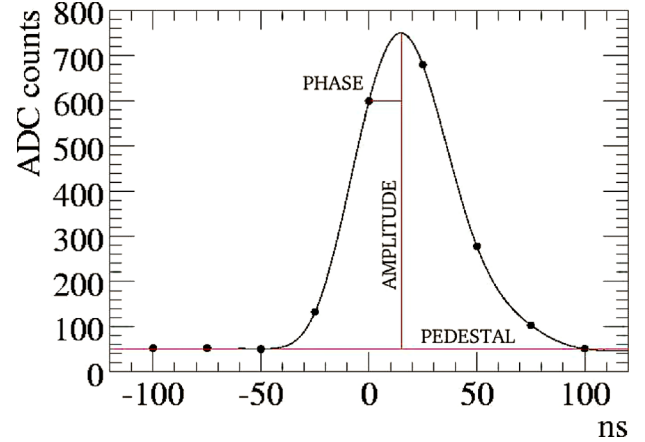


Fig. 6. Definition of amplitude, phase and pedestal.

$$\tau = \frac{1}{A} \sum_{i=1}^n b_i (S_i - p) \quad (2)$$

$$\chi = \frac{1}{A} \sum_{i=1}^n |(S_i - p) - A g_i| \quad (3)$$

where S_i represents the digital sample i and n is the total number of samples. The number of samples is 7 for physics and 9 for calibration runs [7]. We define the pedestal, p , as the baseline of the signal. The amplitude, A , is the height of the signal measured from the pedestal, τ , is the phase and it is defined as the time between the central sample and the peak of the pulse (Fig. 6).

The weights, a_i and b_i , are obtained from the pulse shape of the photomultipliers and the noise autocorrelation matrix, g_i . Correctly determined weights minimizes the effect of the noise in the amplitude and phase reconstruction [10].

Fig. 7 shows the algorithm to compute the three physics parameters in the DSP. Each raw data event is made of 16 DMU blocks. Each DMU block has a header, 7 samples from three channels and a trailer. The header includes information about the gain for each channel. The first step in the algorithm is to obtain the gain for each channel of the DMU. Then, the pedestal is assigned as the first sample. The energy of each channel is calculated from the gain, the pedestal and the samples. After that, the energy is rounded, re-scaled and packed to adapt it to the output data format. The second magnitude computed is the phase that is also adapted to the output data format. The final phase value has to be divided by the energy amplitude. This division is done by using a Look Up Table with the value of the inverse of the Energy already defined and stored in the DSP internal memory. Finally, the quality factor is computed and packed as needed by the data format.

Both, the energy and the phase reconstruction need a suitable set of weights for their calculation. These weights have to be downloaded into the DSP from a database at configuration time by the TDAQ software.

The weights are calculated assuming fixed phases. However, if the data is not synchronized with the trigger clock, as it occurs during TileCal commissioning it is still possible to estimate the arrival phase of the data with an iteration procedure [10]. The

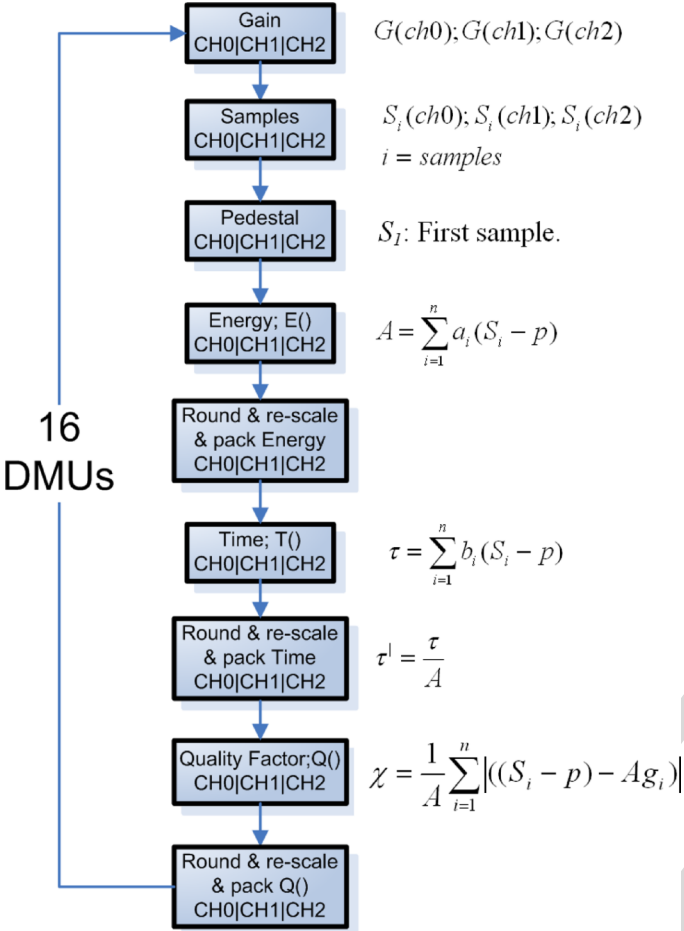


Fig. 7. Optimal filtering DSP algorithm.

initial phase (τ_0) is estimated as the phase between the maximum and the central sample in units of 25 ns, then the amplitude and phase are calculated in each iteration as

$$A_k = \sum_{i=1}^n a_i |_{\tau_{k-1}} (S_i - p) \quad (4)$$

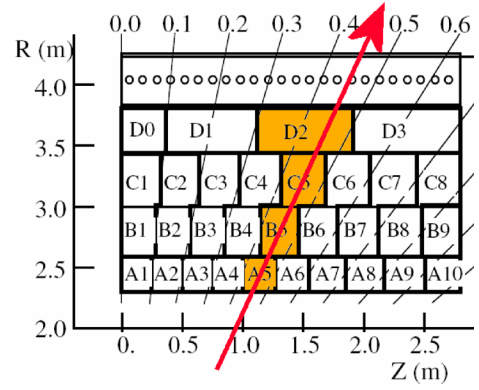
$$\tau_k = \frac{1}{A_k} \sum_{i=1}^n b_i |_{\tau_{k-1}} (S_i - p) \quad (5)$$

where k is the iteration index and runs from 1 to 3. It is being observed that the amplitude converges with three iterations.

2) *The Muon Tagging Algorithm:* The primary goal of the MTag algorithm [11] is to search for low P_T muons taking into account the energy deposited in each layer of TileCal, so that, this information can be used at the second level trigger. Fig. 8 shows the TileCal cell structure with 3 depth layers (A, BC and D cells) and with a projective geometry in η . In order to be identified as a muon its energy deposition must be within the limited of an upper and a lower thresholds:

$$thr_i^{low} \leq E_i \leq thr_i^{high} \quad i = 1, 2, 3. \quad (6)$$

If this condition is fulfilled in each of the 3 layers with a projective pattern in pseudo rapidity (η), the muon is tagged.

Fig. 8. Muon path going through a tower in η .

In order to gain efficiency for events where the muon loses a considerable fraction of its energy in one of the layers, muons are also tagged if (6) is fulfilled in two layers while in the third layer the energy deposition is above the upper threshold.

The lower energy threshold is meant to cut electronic noise and minimum bias contributions. In this algorithm, the lower energy threshold has been set to the same value for all cells. The upper energy thresholds are meant to separate muons and hadronic showers. These thresholds are determined for each cell individually, taking into account the η of the muon trajectory. This algorithm outputs the number of muons identified inside a TileCal module and the pseudo rapidities of these muons. Fig. 9 shows the algorithm implemented in the DSP to identify the muons.

First, the energy deposited on each cell is calculated from the channel energies coming from the OF algorithm, taking into account the channel gain. A search is performed on all the D layer cells, and in case the energy for any cell is comprised between its corresponding thresholds, a candidate is created. The next steps in the algorithm search the inner layers for the η values where first candidates were found. If the conditions explained above are fulfilled the candidate is tagged as a muon.

3) *The Missing E_T Algorithm:* Since the DSP computes the energy per channel with the OF algorithm, it is possible to use this result in order to compute the total transverse energy in the module. Furthermore, since the DSP knows the position of the module which is read out, it is also possible to compute the X and Y projections of the total transverse energy. This information is used at the second level trigger to search for missing transverse energy over the whole detector.

Fig. 10 shows a diagram with the algorithm implemented in the DSP to compute the total transverse energy and its X and Y projections. The energy per tower is computed with the energy per channel obtained with the OF algorithm and the corresponding gain per channel. If both the muon tagging and the missing E_T algorithms are selected, the energy per cell computed in the MTag function is used by the missing E_T algorithm to obtain the energy per tower. The transverse energy per tower is then obtained by applying trigonometric factors.

The total transverse energy of the module is the sum of all the towers. Finally, X and Y projections are calculated. In order to make the algorithm run faster, all the relevant trigonometric factors are stored in a LUT.

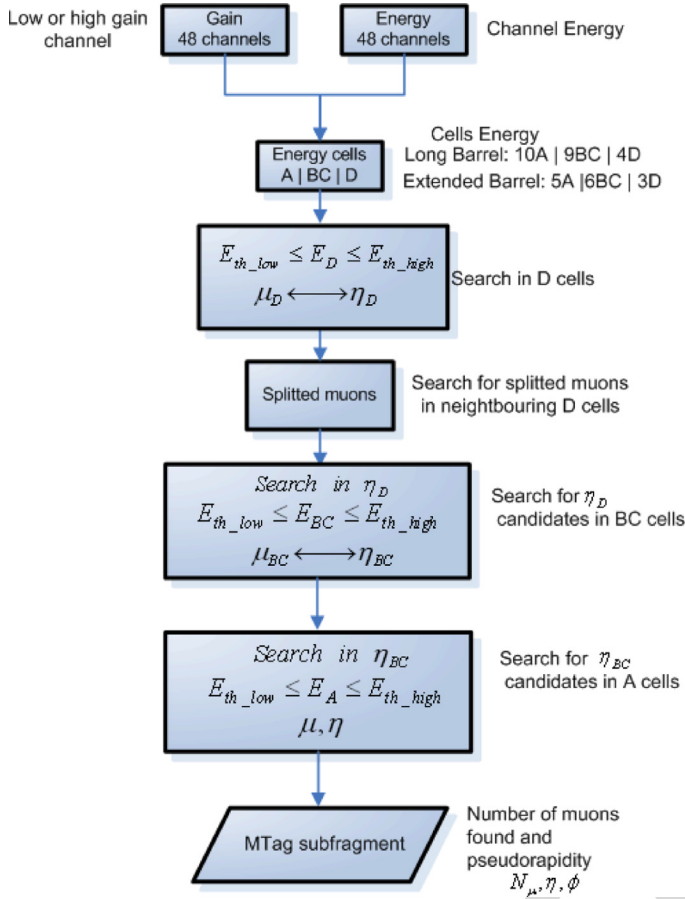


Fig. 9. Muon tagging DSP algorithm.

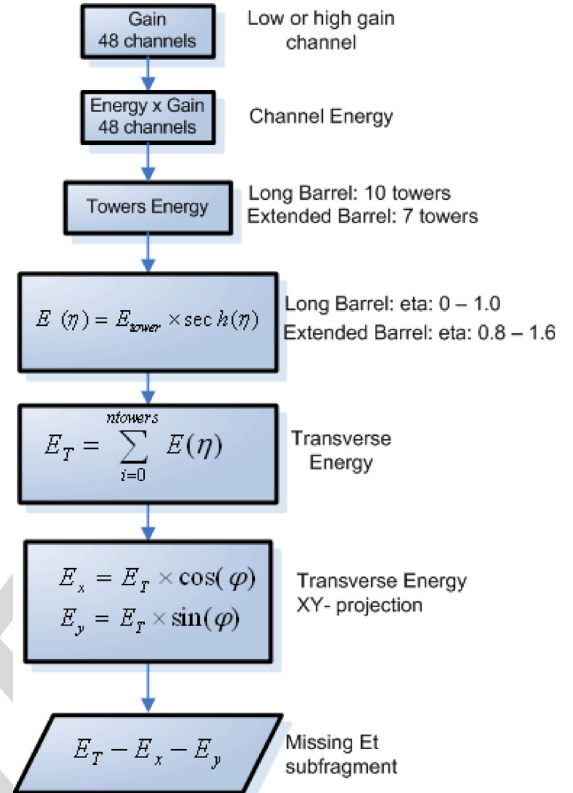
The missing E_T sub-fragment, shared with the muon tagging algorithm, has two words per module. The first 32-bit word encodes the X and Y projection whereas the second word includes the total transverse energy and the sign for the X and Y projections.

4) *Histogramming*: The online histograms built by the DSP are stored in a continuous memory region accessible through the HPI. Each histogram consists of:

- **Header**: includes the drawer number (input link 1 or 2), the physic magnitudes (energy low gain, high gain, phase and quality factor), the channel number (0–45) and the number of bins in the histogram.
- **Data**: The number of bins is defined by the user. By default there are 64 bins.
- **Underflow and Overflow**: Finally, there are two more words for the underflow and overflow.

The binning used by the DSP to build the histograms can be configured by the user. The binning is stored into the DSP memory and it can be configured through the HPI.

The information used to build the histograms is taken from the output data buffer, because the physics magnitudes are only available after processing the reconstruction algorithms. Nevertheless, it is still possible to build histograms with the input raw data. It is useful to build histograms of input raw data when only the reconstructed data is sent out. For instance, the value of the first sample, CRC errors or parity errors can be used to

Fig. 10. Missing E_T DSP algorithm.

build histograms at DSP level, since these variables are lost if only the reconstructed data is sent.

E. Conclusions

The DSP performance has been tested during the TileCal RODs production [12]. The DSP was tested in copy mode obtaining a Bit Error Rate better than 3×10^{-3} for a confidence level of 95%.

The performance of the reconstruction algorithms is being validated during the TileCal commissioning. The OF algorithm presents a difference in the amplitude between the offline and online reconstruction of around 0.03 ADC counts (0.02 GeV in low gain and 0.4×10^{-3} GeV in high gain) and the MTag algorithm shows an efficiency of around 80% for muons with P_T above 2 GeV and a fraction of fakes lower than 6% [12].

The DSP performance was also tested at high event rate during the production tests [13]. The maximum Level 1 event rate can not be achieved in copy mode due to the ROD output data bandwidth limitations and in OF mode due to the OF algorithm processing time, which is 23 μ s is the current DSP version. We expect to reduce the processing time to under 10 μ s by using OF without iterations and an optimized programming of the reconstruction algorithms in assembler.

REFERENCES

- [1] Tile Calorimeter Technical Design Report, ATLAS Tile Calorimeter Collaboration, CERN/LHCC 98-15.
- [2] J. Castelo *et al.*, TileCal ROD Hardware and Software Requirements, 2005, ATL-COM-TILECAL-2005-002.

- [3] ATLAS Technical Proposal, ATLAS Collaboration, 1994, CERN/LHCC/94-43.
- [4] B. Taylor, "TTC distribution for LHC detectors," *IEEE Trans. Nucl. Sci.*, vol. 45, no. 4, pp. 821–828, Aug. 1998.
- [5] P. Matricon *et al.*, The Trigger and Busy Module of The ATLAS LARG ROD System, ATL-AL-EN-0054.
- [6] TMS320C6414, Fixed Point Processor, Texas Instrument, 2003. [Online]. Available: <http://focus.ti.com/docs/prod/folders/print/tms320c6414.html>.
- [7] K. Anderson *et al.*, "ATLAS tile calorimeter interface card," in *Proc 8th Workshop Electronics for LHC Experiments*, ISBN 92-9083-202-9.
- [8] J. Christiansen *et al.*, Timing Receiver ASIC (TTCrx) Reference Manual. [Online]. Available: http://tc.web.cern.ch/TTC/TTCrx_manual3.5.pdf.
- [9] ATLAS High-Level Trigger, Data Acquisition and Controls TDR, ATLAS HLT/DAQ/DCS Group, 2003, ATLAS TDR-016.
- [10] B. Salvachua *et al.*, "Algorithms in the ROD DSP of the ATLAS hadronic tile calorimeter," *JINST*, vol. 2, no. 2, p. T02001, 2007.
- [11] A. Ruiz-Martínez, "Development of a Low p_T Muon LVL2 Trigger Algorithm with The ATLAS TileCal Detector," Master's thesis, Univ. Valencia, Valencia, Spain, 2006.
- [12] J. Poveda *et al.*, "Production performance tests of the TileCal read-out driver boards," *IEEE Nucl. Sci.*, to be published.
- [13] A. Valero *et al.*, "ATLAS TileCal read out driver production," *JINST*, vol. 2, p. P05003, 2007.

IEEE
PROOF