Masking Problematic Channels in the Liquid Argon Calorimeter for the High-Level Trigger of ATLAS

by

Ryan Paul Taylor

B.Sc., University of Regina, 2006

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in the Department of Physics and Astronomy

University of Victoria

Masking Problematic Channels in the Liquid Argon Calorimeter for the High-Level Trigger of ATLAS

by

Ryan Paul Taylor

B.Sc., University of Regina, 2006

Supervisory Committee

Dr. R. Kowalewski, Supervisor (University of Victoria)

Dr. M. Lefebvre, Member (University of Victoria)

Dr. D. Karlen, Member (University of Victoria)

Dr. D. Steuerman, Outside Member (University of Victoria)

Supervisory Committee

Dr. R. Kowalewski, Supervisor (University of Victoria)

Dr. M. Lefebvre, Member (University of Victoria)

Dr. D. Karlen, Member (University of Victoria)

Dr. D. Steuerman, Outside Member (University of Victoria)

Abstract

Read-out channels in the liquid argon (LAr) calorimeter of the ATLAS detector are susceptible to various kinds of faults, which can impair the selection of events made by the trigger system. General-purpose software tools have been developed for dealing with problematic calorimeter channels. In order to give High-Level Trigger (HLT) algorithms robustness against detector problems, these tools have been applied in the HLT calorimeter data preparation code to mask problematic channels in the LAr calorimeter. Timing measurements and optimizations have been conducted to assess and minimize the impact of these operations on the execution speed of HLT algorithms. The efficacy of the bad-channel masking has been demonstrated using cosmic-ray data.

Table of Contents

Supervisory Committee **ii**

Abstract **iii**

Table of Contents **iv**

List of Figures **vi**

Acknowledgements **viii**

1 Introduction **1**

1.1 Introduction 1

2 The ATLAS Detector **5**

2.1 Detector Hardware 5

3 Trigger and Data Acquisition Systems **15**

3.1 Data Preparation for HLT Calorimeter Algorithms 17

3.2 Missing E_T Trigger Slice 19

3.3 Monitoring 21

4 Trigger Robustness **24**

4.1 LAr Bad-Channel Software 24

4.2 Optimization and Timing Tests 32

5 Results and Conclusions **37**

5.1 Results of Bad-Channel Masking Tests 37

5.2 Summary 39

A Attribution of Work **45**

B List of Abbreviations **46**

List of Figures

1.1 Cross-sections of several benchmark physics interactions at the LHC . 3

2.1 Cut-away view of the ATLAS calorimetry 6
2.2 Layering and granularity of the LAr subcalorimeters 8
2.3 Depiction of a trigger tower in the LAr EMB subcalorimeter 9
2.4 Ideal pulse shape of a LAr channel 10
2.5 FEB schematic diagram 11
2.6 Table of channel problems 13
2.7 Summary of known problematic channels in the LAr calorimeter . . . 14

3.1 Schematic diagram of the Trigger and Data Acquisition systems . . . 17
3.2 LAr monitoring histogram showing noise burst 22
3.3 L1 trigger monitoring histogram showing noise burst 23

4.1 LArBadChannelMasker conceptual flowchart 30
4.2 Illustration of LArBadChannelMasker masking logic 31
4.3 Comparison of different LArBadChanTool look-up methods 34
4.4 Scaling of masking time with RoI size 35
4.5 Final timing evaluation of per-event cell masking 36

5.1 SumET distributions with and without masking 38
5.2 MET phi distributions with and without masking 40
5.3 Cell energy differences due to masking 41

5.4 Photons in L2 from cosmic ray data, with and without masking . . . 42

Acknowledgements

I wish to thank my supervisor Bob Kowalewski for his guidance, experience and patience. I would also like to thank Teddy Todorov and Denis Damazio for the valuable time they spent collaborating with me; I learned a lot from their expertise.

Chapter 1

Introduction

1.1 Introduction

Fundamental discoveries in particle physics are expected at the tera-electronvolt energy scale. The Large Hadron Collider (LHC), a synchrotron particle accelerator at the European Organization for Nuclear Research (CERN), has recently been constructed in order to explore this new regime of physics.

The primary motivation for the LHC is to understand the nature of electroweak symmetry breaking, the last part of the Standard Model of particle physics which remains unknown. Electroweak symmetry breaking is the mechanism which causes some particles to have mass, while others remain massless. The Higgs mechanism is postulated as a possible origin of electroweak symmetry breaking, and predicts the existence of a scalar boson called the Higgs boson [1]. Searching for the existence of this particle will make it possible to either confirm or rule out the Higgs mechanism, and is therefore one of the main goals of the LHC physics program. Another important goal is to search for the existence of supersymmetric particles, which are partners of Standard Model particles but have greater mass, and spin quantum numbers differing by $\frac{1}{2}$. Observation of such particles would confirm supersymmetry, a theoretical framework which solves the hierarchy problem, unifies the electroweak force with the strong force (forming a grand unified theory), and could lead to a

theory of quantum gravity [2]. Additional goals include searching for particles which constitute dark matter [3], probing for extra hidden dimensions of spacetime [4], and understanding the asymmetry between the particle and antiparticle content of the universe [5].

In order to achieve these goals, the LHC [6, 7, 8, 9] has been designed to surpass the energy and luminosity of previous hadron colliders by orders of magnitude. Luminosity is a measure of the particle flux, or intensity, of the beams. The rate at which a particle interaction occurs is equal to the cross-section for that interaction multiplied by the luminosity. Since most of the physics processes of interest at the LHC have a very small cross-section, a very high luminosity is required in order to produce enough of those interactions to make a significant observation in a reasonable amount of time. Figure 1.1 shows the cross-sections of several benchmark interactions at the LHC's center-of-momentum energy. The design luminosity of the LHC is 10^{34} cm^{-2}s^{-1}, or 10 nb^{-1}s^{-1}. The LHC collides two proton beams into each other with a total energy of 14 TeV in the center-of-momentum frame.[1] The protons in each beam are clustered in bunches, which collide every 25 ns. At the design luminosity, each of these bunch crossings results in about 23 proton-proton interactions, most of which are mundane quantum-chromodynamical (QCD) processes. This high background of QCD processes makes it challenging to detect the few rare interactions of interest.

The ATLAS detector [11], one of several particle detectors at the LHC, is a large general-purpose detector designed to tolerate the intense background conditions and take full advantage of the high-luminosity environment of the LHC. It is shaped like a cylinder, about 46 m long and 22 m in diameter, and weighs around 7000 metric tons. The detector has symmetry and full coverage in ϕ, the azimuthal angle, and can

[1] However, the energy of an interaction varies and is usually much less than 14 TeV, since the interacting quarks and gluons only have a fraction of the protons' energy.

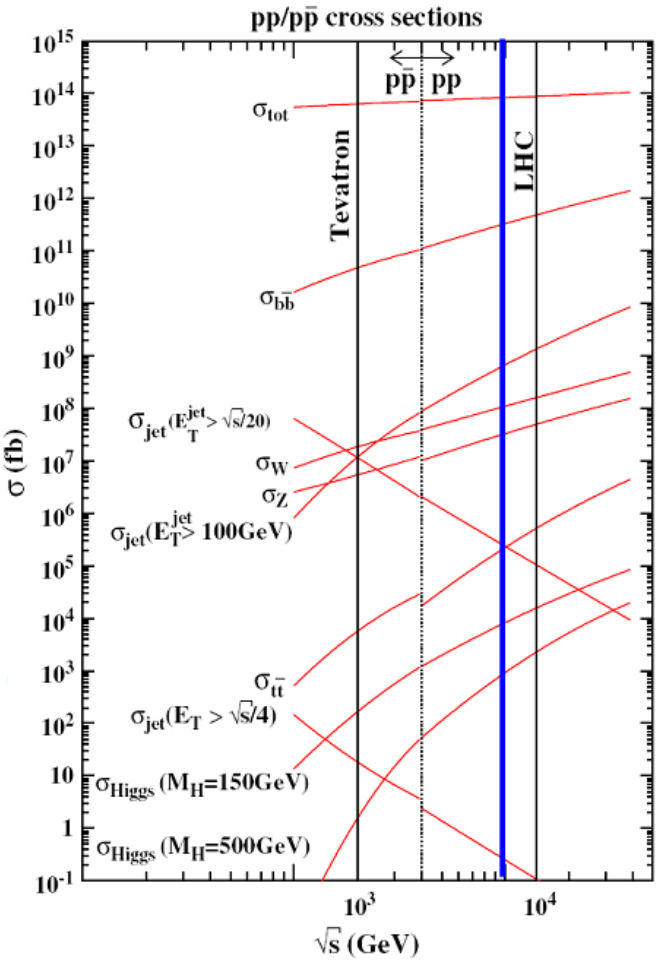

Figure 1.1: Cross-sections of several important physics interactions [10]. Vertical lines indicate the center-of-momentum energy ($\sqrt{s}$) of the LHC at 14 TeV (the design energy), and the Tevatron at 2 TeV for comparison. The thicker vertical line at 10 TeV marks the energy of the LHC's initial run, below the design energy. The cross-sections to the left of the dotted vertical line are for proton-antiproton interactions; the ones to the right are for proton-proton interactions.

make precision measurements in the pseudorapidity[2] range $|\eta| < 2.5$, and coverage for energy measurement up to $|\eta| = 4.8$.

A trigger system is required to reduce the data rate of bunch crossings (40 MHz) to a manageable level by discarding the uninteresting majority of events [12]. The rate at which bunch-crossings can be recorded for further analysis is limited to roughly 200 Hz by the available storage capacity and network infrastructure. The trigger

[2]Pseudorapidity is a convenient measure of the polar angle θ, defined as $\eta \equiv -\ln[\tan(\frac{\theta}{2})]$.

system must therefore achieve a rejection factor in real-time on the order of 2×10^5, while retaining as many of the rare and interesting events as possible for storage and further analysis. It is also important for the trigger system to operate robustly, since the trigger decision dictates the content of the dataset used for physics analysis. The work presented in this thesis enhances the robustness of the trigger system against defective detector elements by masking their signals.

The detector will be further described in Chapter 2, and the trigger and data acquisition systems will be further described in Chapter 3. Chapter 4 will discuss issues of trigger robustness and the bad-channel and masking software. Chapter 5 will demonstrate the efficacy of the masking software on simulated and cosmic ray data.

Chapter 2

The ATLAS Detector

2.1 Detector Hardware

The ATLAS detector has three categories of particle detection systems, located in concentric cylindrical formations around the beamline. Closest to the beamline, within a solenoidal magnet, the Inner Detector measures the trajectories of charged particles with sub-millimeter precision. Farther out, particles pass through the Electromagnetic Calorimeter, which measures the energy deposited by electrons and photons, and then the Hadronic Calorimeter, which measures the energy of jets.[1] The Muon Spectrometer is mounted on the outermost part of the detector amid a toroidal magnet, and measures the trajectory of muons as they escape the detector, in order to determine their momenta. For the purposes of this thesis, only the subcalorimeters which use liquid argon as the sampling medium are directly relevant,[2] and will be further described in this chapter.

2.1.1 Liquid Argon Calorimeter

The ATLAS calorimeters are sampling detectors with full symmetry in ϕ. Liquid argon (LAr) is used as the sampling medium for all electromagnetic calorimetry,

[1] Jets are showers of hadronic particles.

[2] The other type of calorimeter is the Tile Calorimeter, which is the hadronic calorimeter in the barrel and extended-barrel regions.

and for hadronic calorimetry in the forward and end-cap regions. Liquid argon was chosen for its intrinsic properties of radiation-hardness and linearity (i.e. deposited charge $\propto$ energy of incident particles), and response stability over time [11]. The LAr calorimetry system is housed in three cryostats: one for the barrel region, and one for each end-cap region. In the barrel region (i.e. the precision-physics region), the electromagnetic barrel (EMB) subcalorimeter makes precise energy measurements with fine granularity. The end-cap subcalorimeters are the electromagnetic end-caps (EMEC), the hadronic end-caps (HEC), and the forward calorimeters (FCal). Each subcalorimeter has two halves, called the A side ($\eta > 0$), and the C side ($\eta < 0$). The layout of these subcalorimeters within the detector is shown in Figure 2.1. The table in Figure 2.2 gives the geometric granularity in (η, ϕ) coordinates of the readout channels in each layer in depth (or sampling) of each LAr subcalorimeter. Figure 2.3 shows the geometry of channels in the EMB and how they are arranged in sections called trigger towers (TTs).

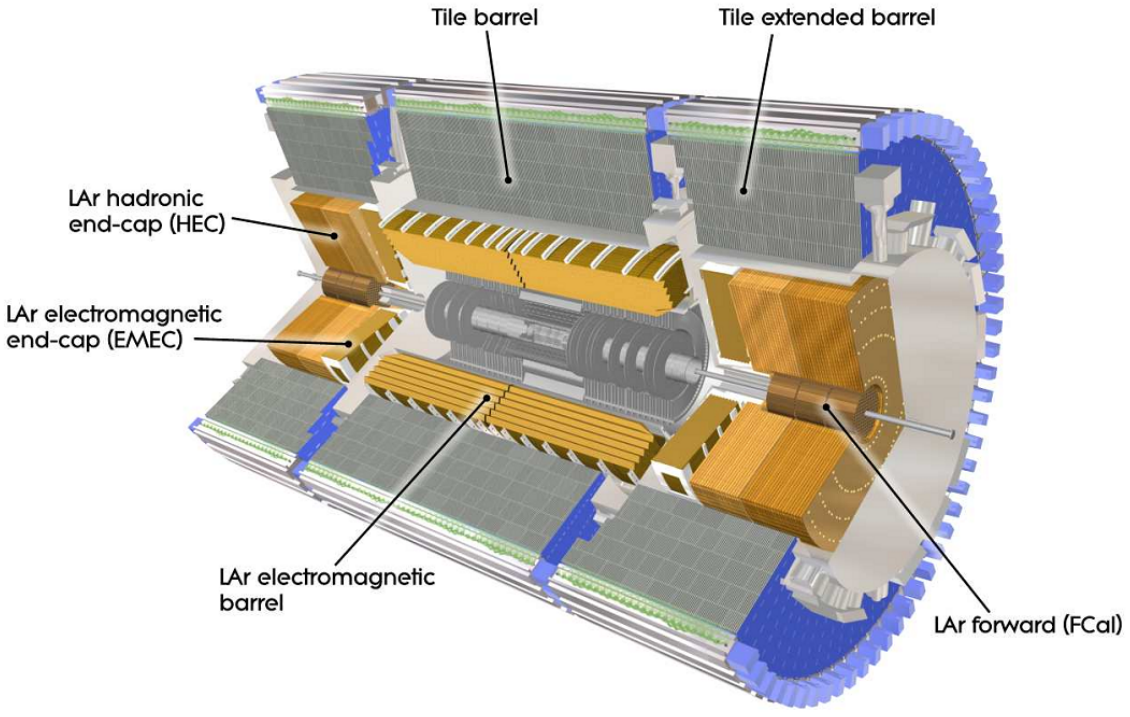

Figure 2.1: Cut-away view of the ATLAS calorimetry [11].

Front-End Electronics and Signal Readout

When a charged particle impinges on one of the LAr subcalorimeters, an electromagnetic shower is induced by the absorber material,[3] and the charged particles in the shower ionize the liquid argon. The voltage applied across the liquid argon medium by the electrodes causes the free electrons to drift towards the positive electrodes. This drift current constitutes the analog signal read out by a channel; the peak of the current is proportional to the energy deposited by the incident particle. The analog signals are carried out of the cryostats by feedthroughs (FTs), then amplified and sampled at 25 ns intervals and digitized in the Front-end Boards (FEBs). There are 1524 FEBs in total, and each one services 128 channels. An ideal signal is shown in Figure 2.4, and the FEB electronics are illustrated in Figure 2.5. Since the duration of the signal is long compared to the bunch-crossing time of 25 ns, it is possible for particles originating from collisions in different bunch crossings to have overlapping signals. This is referred to as out-of-time pile-up, and it is a significant source of noise in the calorimeter energy measurements, especially at high luminosities. The effect of out-of-time pile-up is reduced by the fact that the integral over time of the shaped signal is close to zero. Therefore, the small but frequent energy depositions of soft QCD background activity tend to cancel out on average.

After the pulse is sampled and digitized in the FEBs, the digital samples are transmitted to the Readout Drivers (RODs), where the Optimal Filtering (OF) algorithm is used to reconstruct the energy of the pulse. For all cells with energy exceeding a certain threshold, the time of the signal and a data quality factor are also calculated. The energy and time are calculated by linearly summing the amplitudes of all the samples and weighting them with the Optimal Filtering Coefficients (OFCs), which are parameters obtained by minimizing the variation in energy due to electronic and

[3]The absorber material is lead in the electromagnetic calorimeters, copper in the HEC, and a combination of copper and tungsten in the FCal.

	Barrel		End-cap	
EM calorimeter				
Number of layers and $\|\eta\|$ coverage				
Presampler	1	$\|\eta\| < 1.52$	1	$1.5 < \|\eta\| < 1.8$
Calorimeter	3	$\|\eta\| < 1.35$	2	$1.375 < \|\eta\| < 1.5$
	2	$1.35 < \|\eta\| < 1.475$	3	$1.5 < \|\eta\| < 2.5$
			2	$2.5 < \|\eta\| < 3.2$
Granularity $\Delta\eta \times \Delta\phi$ versus $\|\eta\|$				
Presampler	0.025×0.1	$\|\eta\| < 1.52$	0.025×0.1	$1.5 < \|\eta\| < 1.8$
Calorimeter 1st layer	$0.025/8 \times 0.1$	$\|\eta\| < 1.40$	0.050×0.1	$1.375 < \|\eta\| < 1.425$
	0.025×0.025	$1.40 < \|\eta\| < 1.475$	0.025×0.1	$1.425 < \|\eta\| < 1.5$
			$0.025/8 \times 0.1$	$1.5 < \|\eta\| < 1.8$
			$0.025/6 \times 0.1$	$1.8 < \|\eta\| < 2.0$
			$0.025/4 \times 0.1$	$2.0 < \|\eta\| < 2.4$
			0.025×0.1	$2.4 < \|\eta\| < 2.5$
			0.1×0.1	$2.5 < \|\eta\| < 3.2$
Calorimeter 2nd layer	0.025×0.025	$\|\eta\| < 1.40$	0.050×0.025	$1.375 < \|\eta\| < 1.425$
	0.075×0.025	$1.40 < \|\eta\| < 1.475$	0.025×0.025	$1.425 < \|\eta\| < 2.5$
			0.1×0.1	$2.5 < \|\eta\| < 3.2$
Calorimeter 3rd layer	0.050×0.025	$\|\eta\| < 1.35$	0.050×0.025	$1.5 < \|\eta\| < 2.5$
Number of readout channels				
Presampler	7808		1536 (both sides)	
Calorimeter	101760		62208 (both sides)	
LAr hadronic end-cap				
$\|\eta\|$ coverage			$1.5 < \|\eta\| < 3.2$	
Number of layers			4	
Granularity $\Delta\eta \times \Delta\phi$			0.1×0.1	$1.5 < \|\eta\| < 2.5$
			0.2×0.2	$2.5 < \|\eta\| < 3.2$
Readout channels			5632 (both sides)	
LAr forward calorimeter				
$\|\eta\|$ coverage			$3.1 < \|\eta\| < 4.9$	
Number of layers			3	
Granularity $\Delta x \times \Delta y$ (cm)			FCal1: 3.0×2.6	$3.15 < \|\eta\| < 4.30$
			FCal1: $\sim$ four times finer	$3.10 < \|\eta\| < 3.15$,
				$4.30 < \|\eta\| < 4.83$
			FCal2: 3.3×4.2	$3.24 < \|\eta\| < 4.50$
			FCal2: $\sim$ four times finer	$3.20 < \|\eta\| < 3.24$,
				$4.50 < \|\eta\| < 4.81$
			FCal3: 5.4×4.7	$3.32 < \|\eta\| < 4.60$
			FCal3: $\sim$ four times finer	$3.29 < \|\eta\| < 3.32$,
				$4.60 < \|\eta\| < 4.75$
Readout channels			3524 (both sides)	

Figure 2.2: This table details the layering, granularity, and number of readout channels of all regions of the LAr calorimeters [11]. In regions of higher η, where the background conditions are more intense, the granularity in η decreases. There are a total of 182468 LAr channels, most of which are in the EMB.

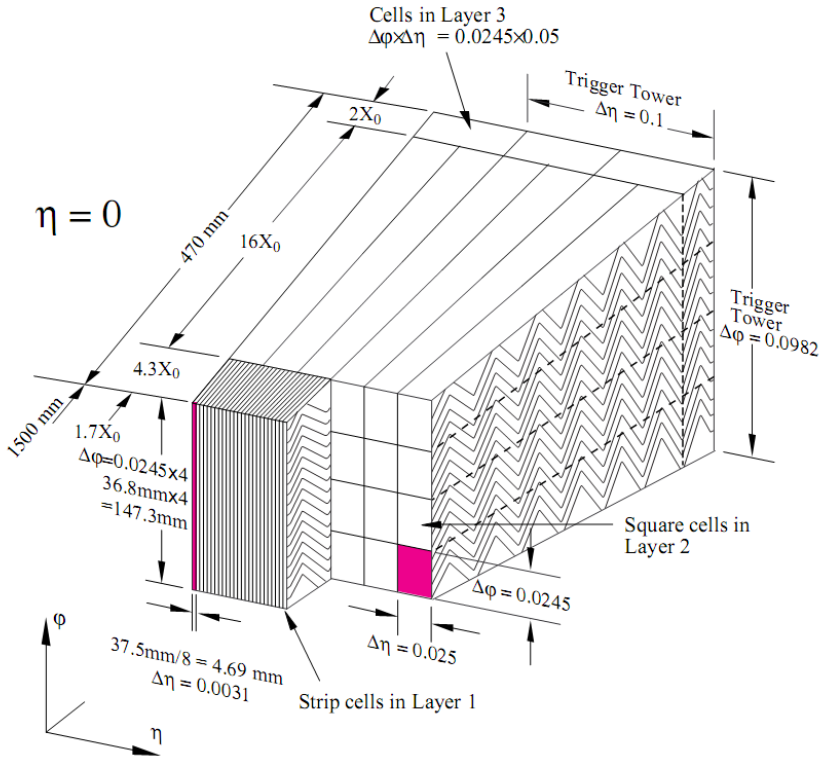

Figure 2.3: A depiction of several cells and a trigger tower in the EMB subcalorimeter [11]. In this region, a trigger tower is composed of a 4×4 grid of cells in layer 2, along with the cells in front and behind in layers 1 and 3. The depth of each layer is given in radiation lengths, X_0.

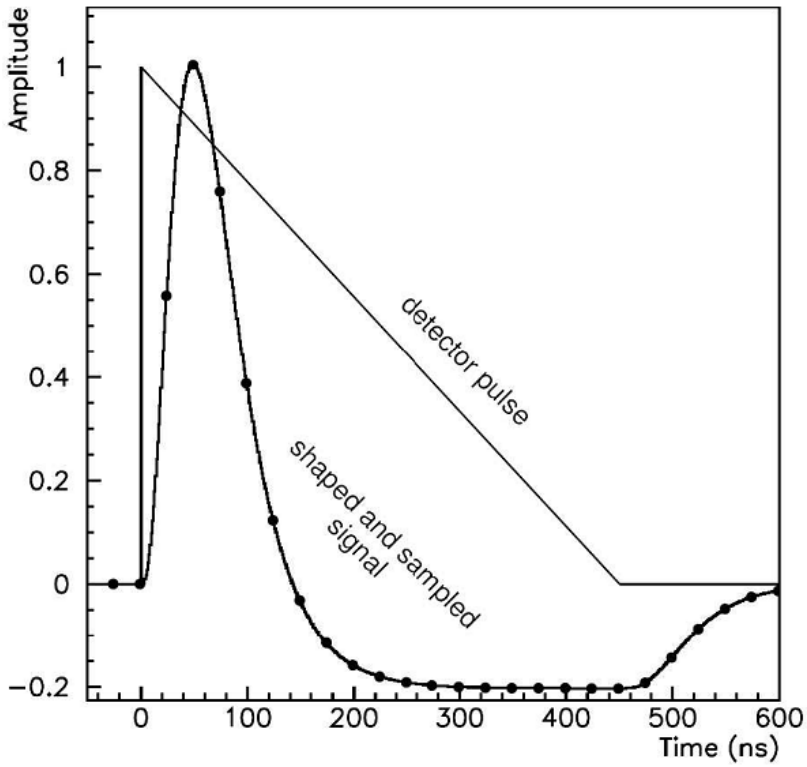

Figure 2.4: The triangular current pulse from a LAr cell, and the FEB output signal after shaping [11]. Dots indicate the sampled points at 25 ns intervals. Up to 32 samples can be taken, but only 5 samples are read out during normal data taking, in order to reduce the volume of data transferred to and processed in the RODs. The integral of the shaped signal is close to zero.

pile-up noise. The data quality factor is a measure of how closely the measured pulse shape conforms to the ideal pulse shape, similar to a χ^2 statistic. It can be used to flag cells with large pile-up noise or other problems which distort the pulse shape.

The front-end electronics include calibration boards which inject pulses of a precisely known magnitude at the electrode readout. Calibration runs are conducted with charge injection, to measure the response per unit of input charge over a wide dynamic range, and without charge injection, to measure the mean and standard deviation (σ) of cell energies in the absence of any signal. This calibration data is regularly collected and processed in order to produce new OFCs, maintaining the calibration of the energy reconstruction. Calibration runs are also important for testing

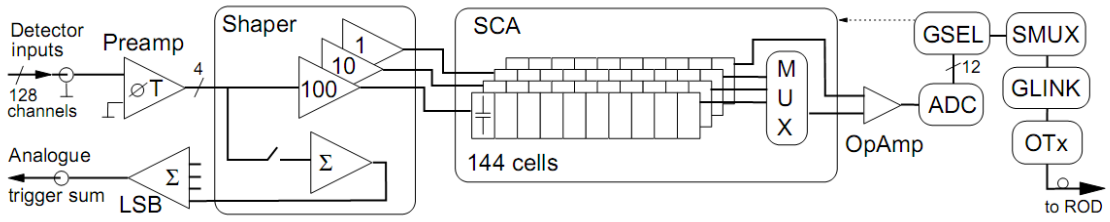

Figure 2.5: Schematic diagram of the signal processing in a FEB [11]. The SCA (Switched-Capacitor Array) is the pipeline memory which stores the analog signals during the L1 latency. Some parts of the readout chain are split into three parallel signal paths, one for each gain (labelled 100, 10 and 1 for high, medium and low gain, respectively). A FEB has 32 pre-amplifiers, shapers and SCAs, each of which processes 4 channels.

the electronic read-out chain and detecting many types of anomalous behaviour in the readout electronics.

Problematic Channels

A LAr channel is susceptible to various kinds of faults, whether in the front-end electronics, readout signal line, or the detector element itself. A common source of most types of problems is defective or damaged electronic connections or circuit components. Most types of problems can be broadly categorized in terms of the symptoms: increased standard deviation or mean of the channel's energy readings (referred to as "noisy" or "hot", respectively), weakening or loss of the signal, distortion of the pulse shape, or complications in the calibration system. Specific classifications of problems have been defined for use in the LAr bad-channel software. The `deadPhys` problem means that the signal is cut off within the detector, and does not arrive at the front-end electronics. Since these types of faults are inaccessible inside the detector, they are irreparable. In contrast, channels with the `deadReadout` problem have their signal cut off within a FEB. This is often due to a poorly soldered connection or a bent pin on a pre-amplifier or other circuit component, which can be repaired during long-term accelerator shutdowns. The `deadCalib` problem signifies that the calibration signal of a channel is distorted, due to defects or damage in its calibration

board. As a result, good OFCs can not be obtained for channels with this problem; the average of OFCs of channels neighbouring in ϕ is used instead. This problem often affects channels in large groups, since up to 32 channels can share the same calibration line. Channels classified as `distorted` have either a distorted pulse shape, or an atypical amplitude. In either case, the difference from a nominal pulse is not severe, and the impact on physics performance is expected to be minor. There are several classifications for increased noise levels, according to the severity and intermittency of the problem. Since noise problems can manifest independently for each gain, each category of noise is broken down into three classifications for high gain (HG), medium gain (MG) and low gain (LG). There are many possible causes for increased noise, including problems in pre-amplifiers, high-voltage supplies, cabling or shielding. The `sporadicBurstNoise` problem is caused by faulty pre-amplifiers, so it occurs in groups of four consecutive channels connected to the same pre-amplifier. All the bad-channel problem classifications are summarized in the table in Figure 2.6, and all known bad channels are shown in Figure 2.7.

Status	**Description**
deadReadout	Signal is cut off at front-end electronics (both physics and calibration signals)
deadCalib	Distortion of calibration pulse
deadPhys	Signal is cut off at the detector (both physics and calibration signals)
almostDead	Pulse shape is normal but the amplitude is very small
short	Electrical short merges two cells
unstable	The signal changes over time
distorted	Distortion of calibration and physics pulses
lowNoiseHG	Noise of 5 to 10 σ at high gain
highNoiseHG	Noise of at least 10 σ at high gain
unstableNoiseHG	Erratic noise at high gain
lowNoiseMG	Noise of 5 to 10 σ at medium gain
highNoiseMG	Noise of at least 10 σ at medium gain
unstableNoiseMG	Erratic noise at medium gain
lowNoiseLG	Noise of 5 to 10 σ at low gain
highNoiseLG	Noise of at least 10 σ at low gain
unstableNoiseLG	Erratic noise at low gain
missingFEB	The channel is part of a FEB which is absent
peculiarCalibrationLine	For example, a calibration line that leaks signal into nearby channels
problematicForUnknownReason	An undiagnosed problem
sporadicBurstNoise	Erratic bursts of noise occur due to a preamplifier problem

Figure 2.6: A list of problems defined for use in the LAr bad-channel software.

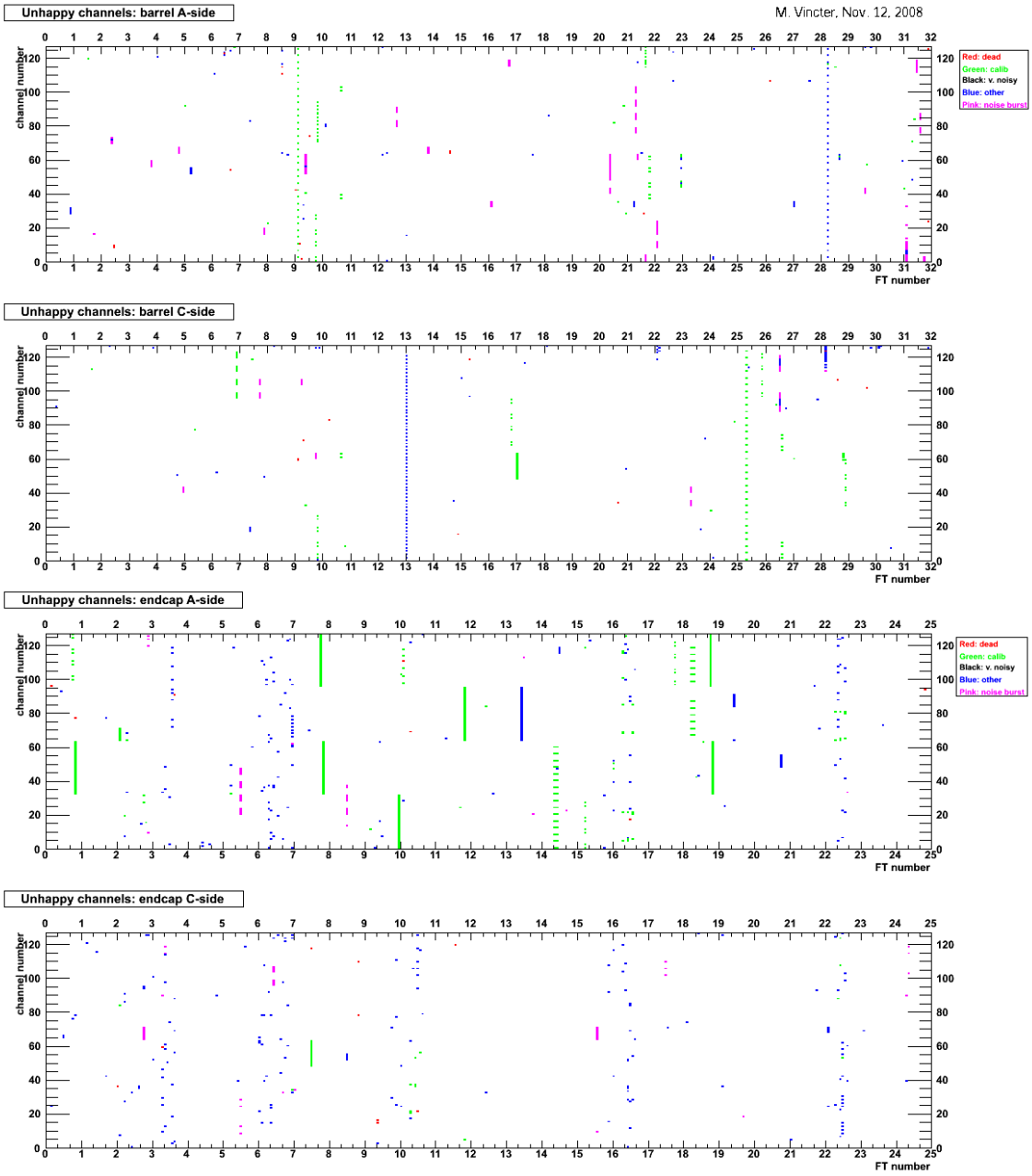

Figure 2.7: A map showing known problematic channels in the LAr calorimeters, by channel number and feedthrough (courtesy of M. Vincter). From top: EMBA, EMBC, endcap A and endcap C. The majority of problems are either related to the calibration signal (`deadCalib` and `peculiarCalibrationLine`), or minor distortions (`distorted`, categorized as "other" in the legend). In total, there are approximately 1300 problematic LAr channels, which is about 0.7% of all LAr channels.

Chapter 3

Trigger and Data Acquisition Systems

The ATLAS trigger system [12] is divided into three levels: level 1 (L1), level 2 (L2), and the Event Filter (EF). L1 is a hardware-based system, while L2 and EF, which are collectively referred to as the High-Level Trigger (HLT) [13], are software-based and massively parallel. L1 reduces the input rate to about 75 kHz, with an average time budget per event of around 2.2 μs, and makes selections based on the presence of objects with high momentum transverse to the beamline. L2 reduces the rate from 75 kHz to about 2 kHz, and each processing node must make a decision within about 40 ms, so fast reconstruction algorithms must be used. Finally, the EF reduces the rate to around 200 Hz, and is allotted roughly 4 seconds to process an event, which allows sophisticated (offline-like) reconstruction algorithms to be run, including basic calibration and alignment corrections. The HLT processor farm contains about 2300 nodes (500 for L2 and 1800 for the EF), each of which contains two quad-core 2.5 GHz processors.

Trigger algorithms are categorized according to the different types of physics signatures that exist. These categories are called slices; some examples of trigger slices are electron/photon, muon, jet, tau, and missing transverse energy (which is discussed in Section 3.2). Each slice contains a number of chains, which are sequences of algorithms executed in a step-wise manner, from L1 to L2 to the EF, with each

result seeding the next step. If the event features do not satisfy the thresholds (or other criteria) at any point of a chain, that chain is terminated, so usually only a subset of trigger chains are satisfied on any given event. It is important to use this principle of early rejection to discard unwanted events as quickly as possible. Since the majority of events are ultimately rejected, the amount of time spent on rejected events is the dominant contribution to the total processing time.

The trigger system is integrated with the Data Acquisition (DAQ) system, which buffers data and transfers it from one trigger level to the next, and finally to storage. Figure 3.1 schematically illustrates the combined trigger and DAQ systems. The readout from the calorimeters and muon spectrometer branches into two independent and parallel paths: coarse information[1] is sent to the central L1 trigger processing unit in the form of analog signals, while the fully-detailed information remains buffered in pipeline memories, for potential delivery to the HLT. Since L1 and the HLT have parallel readout chains, it is possible for problems to affect L1 data but not HLT data, and vice versa. L1 identifies Regions of Interest (RoIs) in the event wherever significant energy deposits or muon tracks are found, and determines which RoIs pass which thresholds. If any RoIs pass, the full event information is sent from the pipeline memories to the Read-Out Drivers (RODs) and Buffers (ROBs), and those RoIs are assigned to a node in the L2 trigger, in continuation of the trigger chain. The L2 algorithms seeded by those L1 RoIs retrieve event information (in full granularity) from the ROBs as needed, based on the RoI locations, using a software interface that will be described in further detail in Section 3.1. Since L2 reconstruction is seeded by the RoIs received from L1, only a small portion (about 2% [12]) of the event information is retrieved at this stage, in order to minimize network data transfer time. If L2 accepts an event, the Sub-Farm Input (SFI) nodes collect all data for that event from the ROBs, and assemble it for a node in the EF farm, where the EF

[1]The information used by L1 must be low-resolution in order to expedite the data transfer and processing. The L1 calorimeter information has the geometric granularity of Trigger Towers.

algorithms seeded by successful L2 chains are executed. If the EF accepts the event, the data are sent to the Sub-Farm Ouput (SFO) nodes for storage, in the form of bytestream files.

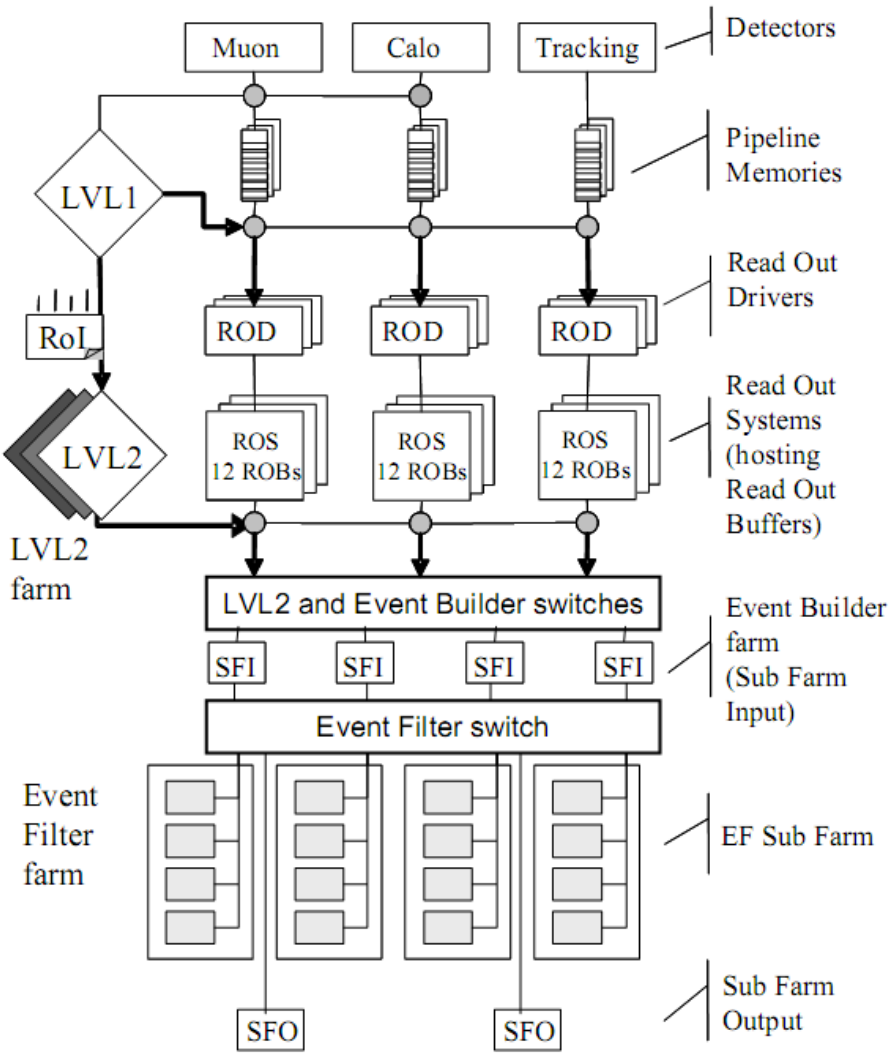

Figure 3.1: Schematic diagram of the Trigger and DAQ systems [12].

3.1 Data Preparation for HLT Calorimeter Algorithms

An understanding of the HLT data preparation software, described in this section, will be important for Chapter 4. This software is part of the interface between the raw bytes of data transmitted from the calorimeter read-out electronics, called

a bytestream, and the object-oriented cell information used in HLT algorithms. Broadly speaking, this interface is made up of two layers: the bytestream decoding (LArRodDecoder), which converts bytestream data into cell information and assigns it to the appropriate cell objects, and the data access mechanism (TrigDataAccess), which serves the prepared cell data to HLT algorithms corresponding to the geometric region requested. These interfaces are the only way to access calorimeter data in the HLT, so they are used by many HLT algorithms.

Calorimeter cells are represented in software by CaloCell objects, which contain basic information such as a cell's energy, the gain at which it was read out, and a unique identifier.[2] CaloCells are organized in vectors called collections (LArCellCollection), each of which holds the cells corresponding to one ROD (two FEBs, at most 256 cells). All of the collections are further organized in a container called LArCellCont, which is a helper class for TrigDataAccess. At the beginning of a run, LArCellCont prepares a map which associates each trigger tower (TT) with the cells contained in it. Also, in order to avoid time-consuming dynamic memory allocation, all cells and collections are allocated at the beginning of a run, so during event processing, the cell information for an event (energy, time, quality, etc.) is written into the pre-allocated cell objects. The TrigDataAccess interface keeps track of which cells have been accessed in an event, so that cell information is only unpacked into a CaloCell object the first time in an event that that cell is accessed. Subsequent access to a calorimeter region during the same event uses cached data, and is therefore faster than the initial access.

The data preparation process is initiated when an algorithm requests the calorimeter information in a specified geometric region. First, a service called RegionSelector is used to determine which ROBs must be accessed in order to fulfill the request. It is essentially a large matrix containing the minimum and maximum (η, ϕ) coor-

[2]Identifiers will be discussed further in Section 4.1.

dinates of each trigger tower, and the ROD identifier(s) associated with each TT. Then, a data request is sent on the network to the relevant ROBs, and bytestream conversion is initiated on the ROD fragments when they are received. The collection which should be filled by the decoded data is determined based on the ROD fragment identifier, and the collections are filled with the decoded data. Finally, the maps in LArCellCont are used to return all cells in the trigger towers encompassed by the (η, ϕ) region requested by the HLT algorithm.

3.2 Missing E_T Trigger Slice

The Missing E_T (MET) trigger slice consists of two types of signatures: the vector sum of missing transverse energy,

$$\not\boldsymbol{E_T} \equiv - \sum \boldsymbol{E_T} = - \sum_i (E_{xi}\, \hat{\boldsymbol{x}} + E_{y_i}\, \hat{\boldsymbol{y}})$$

and the scalar sum of transverse energy (denoted SumET),

$$\sum |\boldsymbol{E_T}| = \sum_i \sqrt{{E_{xi}}^2 + {E_{y_i}}^2}.$$

Here energy is considered a vector; it is essentially equivalent to momentum. The summations comprise all energy measured by the detector: the energy deposited in the hadronic and electromagnetic calorimeters, and the energy of muons traversing the muon spectrometer. There are different types of elements i over which the summation can be conducted: individual calorimeter cells or coarser detector elements such as FEBs (as discussed later in this section), or reconstructed physics objects, as in the case of offline MET algorithms. In any case, the vectorial direction of each element in the sum is simply its geometric position (projected in the transverse plane) with respect to the center of the detector. Missing transverse energy is the deficit by which the net transverse energy in an event is ostensibly not conserved, which

should ideally equal the amount of undetected transverse energy carried away by weakly-interacting particles. However, instrumental effects, such as spurious energy signals due to problematic calorimeter channels, are a source of fake MET that can degrade the measurement of such particles or mimic their presence. The impact of instrumental effects on the trigger will be discussed further in Chapter 4.

The L2 MET algorithm only takes the MET result from L1 and adds the transverse energy contribution of muons detected by the Muon Spectrometer system, which is generally small compared to the calorimeter contribution. For this reason, the L2 decision is rarely different from the L1 decision, so the L2 MET trigger has little potential to refine the L1 selection. This limitation comes about because only calorimeter data within RoIs can be accessed in L2, and MET is a global quantity that requires information from the entire calorimeter. Therefore, the EF is the only stage at which MET can be calculated using cell-level granularity.

The EF MET algorithm loops over all cells in the Tile and LAr calorimeters, and over the muons obtained from the EF muon algorithm, summing up their energies to calculate the global quantities ΣE_x, ΣE_y and ΣE_T, among others. This task is modularized by the use of TrigMissingEtComponent objects, which contain partial sums of E_x, E_y, and E_T. Each component corresponds to one subdetector sampling, with an additional component for the contribution of muons.[3] The global results are obtained by combining the partial sums in all the components, applying to each a calibration to adjust the slopes and offsets of the energy distributions to provide a better estimate of the true deposited energy.

There are three variations of the EF MET algorithm: the cell-based one described here, one that suppresses noise by skipping all cells with energy less than two standard deviations above the mean, and one that is FEB-based instead of cell-based.

[3]This is the default configuration. It amounts to 25 components, corresponding to: 4 EMB samplings, 4 EMEC samplings, 4 HEC samplings, 3 FCAL samplings, 3 samplings each in the Tile barrel, extended barrel, and gap, and lastly the muon collection.

Instead of retrieving energy information from every cell, the FEB-based tool constructs global sums using sums of each FEB (calculated in the RODs), which greatly reduces the data-retrieval time. However, the disadvantage of FEB-level granularity is slightly decreased resolution, and more importantly, less flexibility in dealing with problematic channels, since individual channels can not be masked in this approach.

3.3 Monitoring

All systems of the ATLAS detector make use of an extensive monitoring framework. The monitoring assembles histograms in real-time for display in the ATLAS control room and elsewhere, and also produces more detailed histograms for further analysis offline. These histograms are useful for assessing data quality and diagnosing instrumental problems. For example, the trigger monitoring histograms can show sudden increases in trigger rates, and the LAr monitoring histograms can reveal problems such as noisy cells or dead regions, and help characterize problematic channels with the statuses listed in Table 2.6. The monitoring histograms in Figures 3.2 and 3.3 show noise bursts (the `sporadicBurstNoise` problem classification) occurring in cosmic ray runs, due to problems in the preamplifiers of certain FEBs.

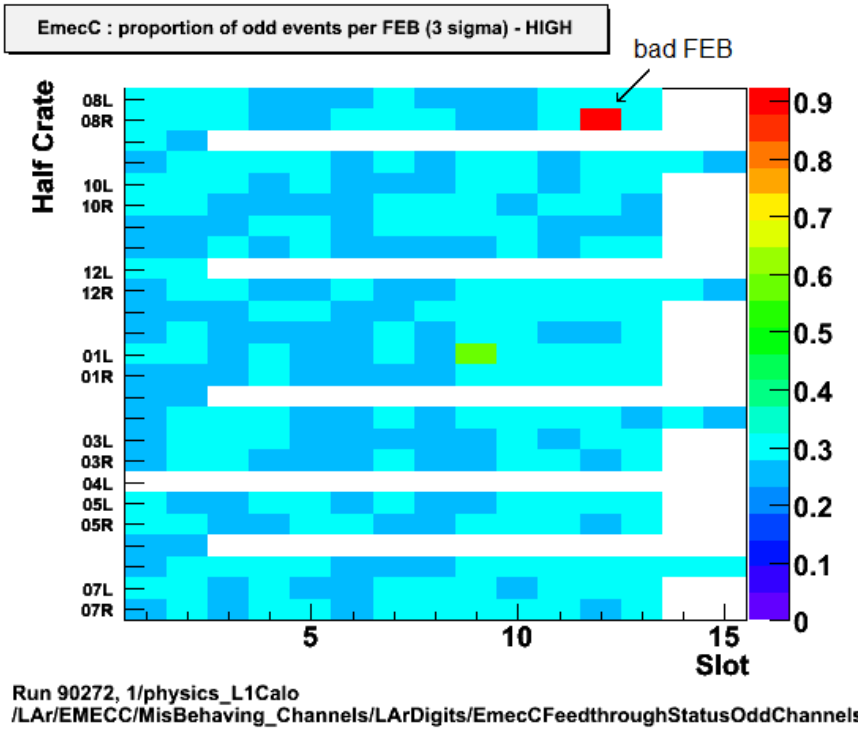

Figure 3.2: This monitoring histogram shows the frequency of cells registering energies more than 3σ away from the mean in EMECC. Cells with Gaussian energy distributions are expected to deviate from the mean by at least 3σ in 0.27% of events. The proportion is much higher in one of the FEBs, indicating a problem in it. The coordinates "Half Crate" and "Slot" refer to pieces of hardware, parts of which are not associated with this subcalorimeter; this explains the blank spaces in the histogram.

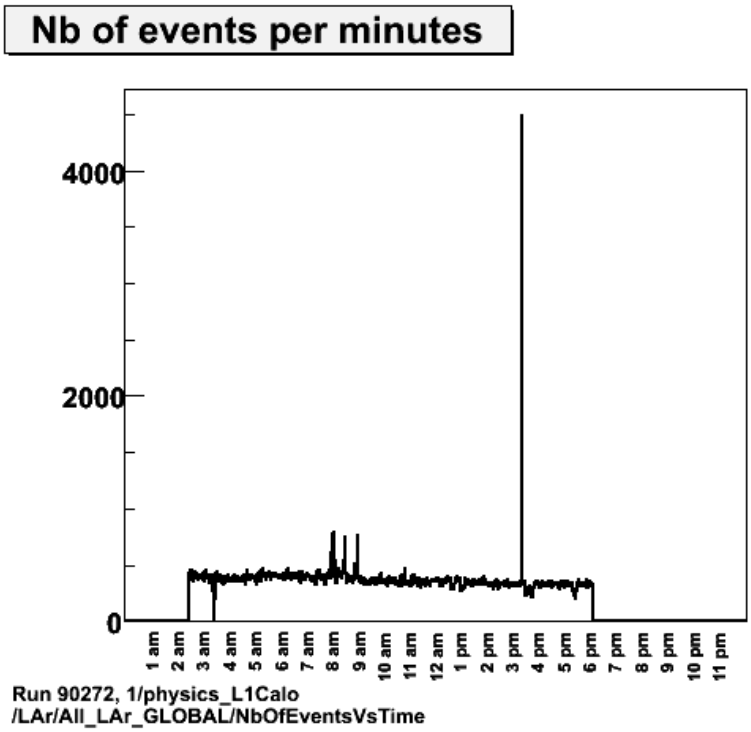

Figure 3.3: This sudden spike in the L1 trigger rate is symptomatic of a noise burst.

Chapter 4

Trigger Robustness

Instrumental issues such as problematic calorimeter channels can have an adverse effect on the trigger. The spurious energy signals of hot or noisy channels can be mistaken for the energy genuinely deposited by particles. Also, dead channels (especially in contiguous groups) can degrade and bias the measurement of MET. Transient problems such as sporadic bursts of noise can also occur. These effects, among others, can inflate the trigger rate with unwanted events, which increases operational deadtime and pollutes the dataset. Even a single hot or noisy channel can be detrimental if it fires often and at a high energy. The trigger system must be robust against these problems, and in particular, it is important for the HLT to have the capability to mask channels which are known to be consistently problematic. The following subsections describe the LAr bad-channel software (Section 4.1) and the timing tests and optimizations done to assess and minimize the impact of masking bad channels on the HLT processing time (Section 4.2).

4.1 LAr Bad-Channel Software

The LAr bad-channel software consists of two main tools, the LArBadChanTool and the LArBadChannelMasker, some helper tools used by the LArBadChanTool,[1]

[1]For example, the LArBadChannelParser, LArBadChannelDecoder, and some tools for transactions with the bad-channel database. These will not be described in full detail.

and several simple but essential classes which form the groundwork for the LAr bad-channel software environment. The main tools will be described in Sections 4.1.1 and 4.1.2, and their use for masking bad channels in Sections 4.1.3 and 4.1.4. First, some of the aforementioned simple foundational classes and other introductory material will be presented here.

The status of a channel is encoded in a LArBadChannel object, as a 32-bit word, allowing up to 32 problems to be defined.[2] The problems are defined by an enumeration in the LArBadChannelEnum class, and are listed in the table in Figure 2.6. To obviate widespread coding of bitwise operations (which can be error-prone and not very human-readable), a bit-packing class, LArBadChanBitPacking, is used to manipulate the bits of LArBadChannel objects: the user has only to specify the enumerator or string corresponding to a given problem in order to set the appropriate bit. An additional benefit of this data abstraction layer is the capability to transparently change the assignment between problems and bits, so that bits can be rearranged or redefined in a new format if needed, with backwards-compatibility such that all formats remain readable.

All bad-channel problems are currently represented by associating a status object (LArBadChannel) with a channel identifier.[3] Two kinds of identifiers are commonly used in the ATLAS calorimeter software: the Identifier class, and the HWIdentifier class. A HWIdentifier, or hardware identifier, can represent a channel or any other piece of hardware, such as a LAr FEB. This type of identifier is commonly used in offline reconstruction, and in low-level calorimeter data preparation code. An Identifier (sometimes called "offline identifier") can represent any LAr or Tile channel, so this type of identifier is required when handling LAr and Tile channels together in the unified Calorimeter context, and is used in higher-level parts of the HLT code.

[2]This number can be expanded if needed.

[3]This may not be the case in a future implementation; more complex forms of bad-channel information, such as crosstalk and shorts, could be represented by associating a problem with two or more channels.

The bad-channel software accommodates both types of identifiers.

4.1.1 LArBadChanTool

The LArBadChanTool is the central piece of the LAr bad-channel software infrastructure. It provides access to all known bad-channel information about the LAr calorimeter. As such, it is a widely-employed tool in ATLAS software, used in online and offline monitoring, some calibration algorithms, and in the treatment of problematic channels in offline reconstruction and the HLT. Its public interface is:

```
LArBadChannel status(HWIdentifier id) const;                    (4.1)

LArBadChannel status(const HWIdentifier& FEBid,
                     int chan) const;                           (4.2)

LArBadChannel offlineStatus(Identifier id) const;               (4.3)

bool febMissing(HWIdentifier febId) const;                      (4.4)

bool febAllGood(HWIdentifier febId) const;                      (4.5)
```

Functions 4.1 and 4.3 return the status of a channel, given a hardware or offline channel identifier, respectively. Function 4.2 is a more optimized form of Function 4.1, for use with the FEB-based hash described later in this section. Function 4.4 is used to query whether a FEB is absent from the detector read-out (due to failure of a power supply, for example), and Function 4.5 is used to query whether a FEB contains any bad channels.

Bad-channel information can be input to the LArBadChanTool in two ways: from the conditions database, and from ASCII files. In order to read ASCII files, the LArBadChanTool uses the LArBadChannelParser to parse the text and check that

it is correctly formatted. The textual information is then passed to the LArBadChannelDecoder, where it is converted into status and identifier objects, and further checked for validity before finally being passed to the LArBadChanTool. This mode of operation is needed for the LArBadChanTool to initially populate the database with information and update it thereafter,[4] and it is also useful for testing purposes. One can write a set of ASCII files containing any desired state of bad channels, and use that in replacement of the conditions database, or to augment the information from the database. However, during normal live running, the conditions database is the sole source of input. The LArBadChanTool retrieves bad-channel information from the database at the beginning of every run. The HLT farm can only access the online replica of the database, which can only be updated between runs. As a result, if a new problematic channel arises during a run, it can not be masked in the HLT without stopping the run, updating the database, and starting a new run.

Once the input data is read from ASCII files and/or the conditions database, the LArBadChanTool stores it in main memory[5] for the duration of the run, in order to have low-latency access to it. Throughout its evolution, the LArBadChanTool has used several different container classes to store its information. In the first implementation, pairs of LArBadChannels and identifiers were stored in a STL vector, sorted by the value of the identifier. A binary search was used to look up a given identifier, giving speed performance that scaled logarithmically with the number of bad channels stored. This data structure was later replaced in favour of a hash table, which occupies more memory but allows any channel to be retrieved in constant time. The data structure has since been further revised such that the hash table is segmented by FEB: 1524 LArBadChanFebState objects are stored in a vector, each

[4]The ASCII files used for database uploads are compiled and maintained by members of the LAr Offline Commissioning community, and contain all bad-channel information which is known to and vetted by them.

[5]It is important for the LArBadChanTool to use memory sparingly, since the ATLAS software is running close to the 2 GB limit, and must not exceed it in order to run on most current computers.

of which represents a FEB and therefore contains a miniature hash table of 128 (or 0 if empty) LArBadChannel objects. This structure retains constant-time access, but has the benefit of smaller memory occupation: if a FEB has no bad channels, the corresponding LArBadChanFebState object will remain at size 0, instead of being expanded to contain 128 empty elements (i.e. good channels). When using Function 4.2, the correct channel can be quickly retrieved by accessing the `chan`th element of the LArBadChanFebState object corresponding to `FEBid`, where `FEBid` and `chan` are the input parameters of the function. The LArBadChanTool creates an auxiliary container indexed by Identifier only when required by a call to Function 4.3, so as to conserve memory. The structure of this auxiliary container is the type which requires binary searching, so querying a channel's status via Identifier is slower than querying by HWIdentifier, which is in turn slower than querying by channel number and HWIdentifier of a FEB.

4.1.2 LArBadChannelMasker

The LArBadChannelMasker is a simple interface to the LArBadChanTool. Its utility is due to two main characteristics: it allows user-level configurability in the usage of bad-channel information retrieved from the LArBadChanTool, and it distills the complete status information of a LArBadChannel object into a boolean value which indicates whether the channel satisfies the user-specified criteria. The LArBadChannelMasker's public interface consists of Functions 4.6 - 4.8. Functions 4.6 and 4.7 tell whether a given channel meets the user-specified status criteria, based on the corresponding hardware or offline identifier, respectively. Function 4.8 is an optimized form of Function 4.6, used in conjunction with Function 4.2.

```
bool cellShouldBeMasked(
   const HWIdentifier& hardwareId,
   const int gain=CaloGain::UNKNOWNGAIN) const;
```
(4.6)

```
bool cellShouldBeMasked(
   const Identifier& offlineId,
   const int gain=CaloGain::UNKNOWNGAIN) const;
```
(4.7)

```
bool cellShouldBeMaskedFEB(
   const HWIdentifier& FEBid,
   const int channelNumber,
   const int gain=CaloGain::UNKNOWNGAIN) const;
```
(4.8)

A configurable user-level interface to the LArBadChanTool is not just convenient, but necessary: there may be many clients of the LArBadChanTool, each of which may wish to use its information in a different way, but there should be only one instance of the LArBadChanTool, since it is the definitive source of bad-channel information and it occupies a significant amount of memory. Each software client therefore configures its own instance of the LArBadChannelMasker with which to interface with the common LArBadChanTool. The most important configurable property of the LArBadChannelMasker is a set of problems called ProblemsToMask, which constitutes the user-specified criteria for selecting channels. The flowchart in Figure 4.1 gives a conceptual illustration of the use and configuration of the LArBadChannelMasker.

The logic used to distill the 32-bit status information of a LArBadChannel object to a boolean value is illustrated by an example in Figure 4.2. A logical `AND` operation is carried out between two or three bitwords. The first bitword is the channel status, which the LArBadChannelMasker retrieves from the LArBadChanTool. The second bitword comprises the problems specified by the user in the ProblemsToMask property. If the gain at which the channel was read out is known, it can be provided as an optional parameter in the function call. Then, a third bitword is also used, which filters out any problems which are not relevant at the given gain. If the result of the `AND` operations is non-zero, the return value is `true`, indicating that the channel has at least one of the problems specified by the user, and if applicable, that the problem is manifest at the given gain.

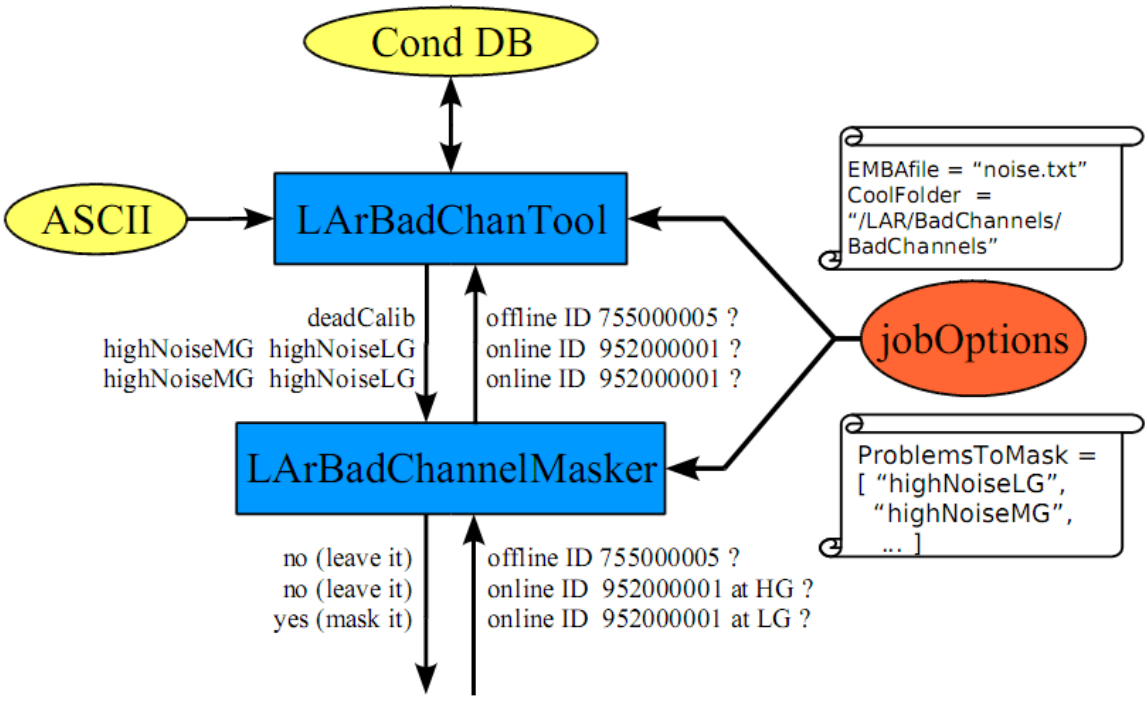

Figure 4.1: A conceptual flowchart of the use and configuration of the LArBadChannelMasker. Queries from the end-user are forwarded to the LArBadChanTool. The status information obtained from the LArBadChanTool is filtered according to the problems given in the ProblemsToMask property, and the gain at which the channel is read out, if applicable. The resulting boolean value is returned to the end-user.

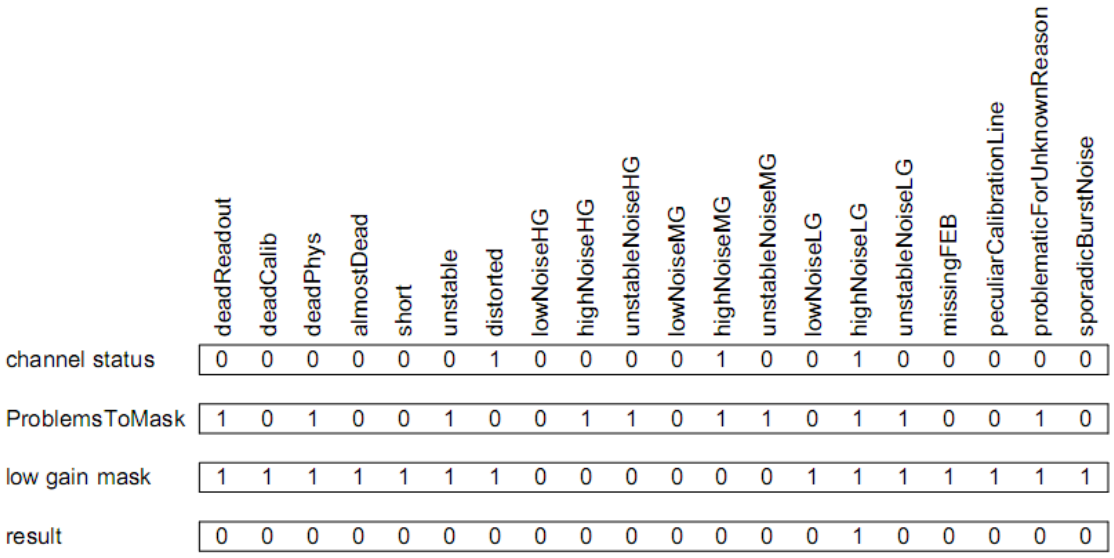

Figure 4.2: In this example, a typical noisy cell is read out at low gain. The value of the ProblemsToMask property corresponds to the default criteria used for HLT masking. The low-gain mask excludes any problems which only manifest at medium or high gain. This channel would be masked due to the `highNoiseLG` problem.

4.1.3 Per-run HLT Masking

Here the first, most basic implementation of masking bad channels in the HLT is described. This implementation of masking is installed in the LArCellCont container described in Section 3.1. Cells are masked by omitting them from the map which associates each trigger tower with the cells contained in it. This does not alter the masked cells themselves, but causes them to be left out of the collections that are returned by TrigDataAccess. Since the TT-cell map is constructed at the beginning of the run and persists throughout the duration of the run, the result is that the masked cells are inaccessible to all HLT algorithms and appear to be missing, for the duration of the run.

This method has the advantage of taking place only at the beginning of the run, so that no additional CPU time is required during event processing. However, the corresponding disadvantage of the permanency of this masking method is the lack of flexibility to deal with channels on an event-by-event basis, based on the gains at

which they are used. Channels can only be masked for the entire run, or not at all. This "all or nothing" approach can cause channels with gain-specific problems to be masked needlessly in events where they are functioning correctly, which decreases the acceptance of the detector.

4.1.4 Per-event HLT Masking

This method of masking channels avoids the drawback of the previous method. It is installed in the LArRodDecoder tool, which is part of the bytestream conversion code discussed in Section 3.1. This tool fills cell collections with energy and gain information extracted from the ROD data fragments. Before the energy is written into a cell, an instance of the LArBadChannelMasker checks whether the cell should be masked, taking its gain into account. If the cell should be masked, then its energy is set to zero instead of the value extracted from the ROD data.[6] Since this process occurs on every event, it costs extra CPU time and slows down the HLT data access somewhat. However, the capability to make the decision to mask on an event-by-event basis, taking the gain into account, can potentially save channels from needlessly being masked if they have gain-dependent problems.

4.2 Optimization and Timing Tests

The operations involved in the per-event method of masking must be stringently optimized, since they are part of the bytestream conversion which is performed on every cell that is accessed by the HLT. This is a concern at both L2 and the EF, since the L2 trigger has a very tight time budget, and when the EF MET algorithm is run, every calorimeter cell is accessed. A series of timing tests were conducted in order to assess the impact of the masking operations on the execution time of the calorimeter data preparation code, and several simple optimizations were implemented in the LArBadChannelMasker to ensure that it would run quickly enough for this purpose.

[6]This method of cell masking does not alter the bytestream files recorded for offline processing; only the HLT is affected.

The most basic optimization was to allow the client of the LArBadChannelMasker to query whether masking was activated. This way, when masking is deactivated, the client code can simply check a local boolean variable instead of making a virtual function call, which reduces the overhead of the masking code to a negligible level when masking is deactivated. Before this optimization (among others), the presence of the masking code contributed to a significant slowing of the bytestream decoding even when masking was switched off.[7]

More speed gains were made by simply checking whether a channel is good before conducting the bitwise `AND` operations explained in Figure 4.2. Since the majority of channels have no problems, this almost always obviates the bitwise operations, allowing a result of 0 to be returned immediately. Using the Valgrind code-profiling suite revealed a less obvious opportunity for optimization: the ToolHandle class (the standard "smart pointer" used to configure, retrieve and access tools in ATLAS software) was prohibitively slow to dereference in this specialized application. By using a ToolHandle only for the configuration, and caching and dereferencing a bare C++ pointer during algorithm execution, the time to query one cell's status was decreased by about 10%, and the total execution time of the EF MET feature extraction algorithm decreased by about 4%. The most significant speed improvement was changing the search method of the LArBadChanTool from a binary search to a FEB-based hash method, as described in Section 4.1.1. A comparison of different look-up methods before optimization is shown in Figure 4.3, and Figure 4.4 shows how the time required for masking scales with RoI size. The final impact on the EF MET feature extraction algorithm of per-event cell masking after all optimizations is shown in Figure 4.5.

The timing tests shown in Figures 4.3 and 4.4 were conducted with the Valgrind profiling tool, so they are completely reproducible; the same test run twice will produce identical results. The test shown in Fig. 4.5 measures the real time spent by the

[7]This can be noticed in Figure 4.4.

CPU, so care was taken to ensure that the standard deviations of the measurements were small, and they were reproduced several times to ensure consistency.

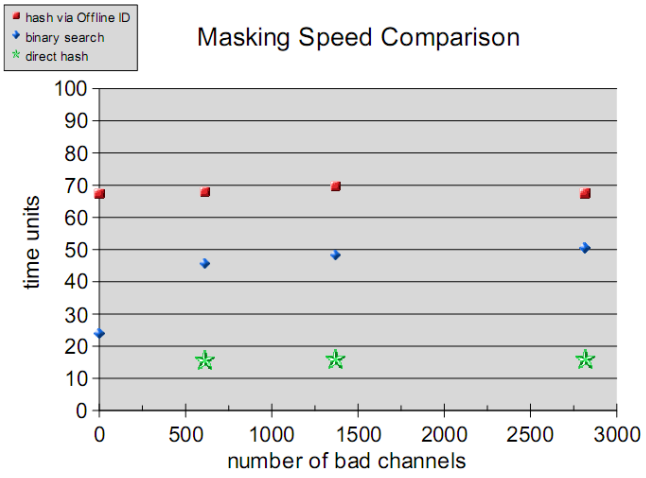

Figure 4.3: Comparison of different look-up methods tested in the LArBadChanTool. The speed of the binary search scales logarithmically with the number of bad channels in the database, and the hash methods have constant speed. The "hash via Offline ID" method has a high overhead due to a cumbersome method of calculating hash values from offline IDs. The "direct hash" method circumvents the computing of the hash values by looking up pre-calculated hashes instead. The final method chosen uses a simplified hash calculation based on FEB and channel numbers, and has performance similar to but slightly better than the "direct hash" method shown here.

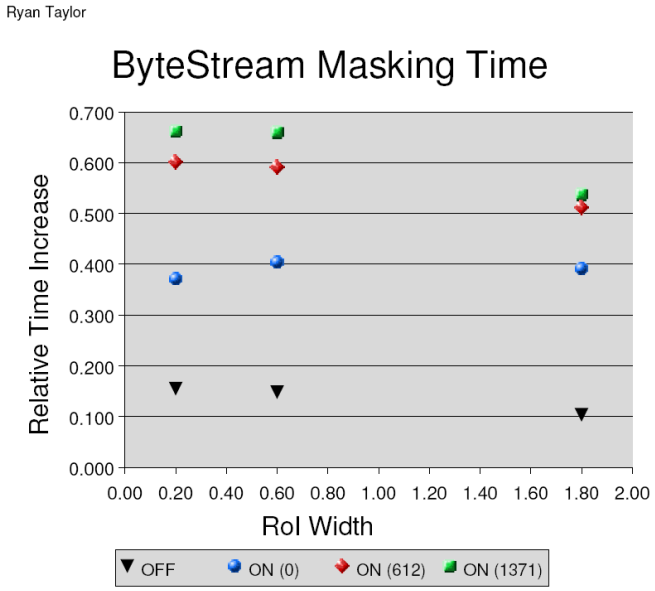

Figure 4.4: Scaling of masking time with RoI size, before optimizations. The slower binary search method was used to look up cells. The time increase due to the masking code is shown relative to the case where no masking code is installed in the bytestream converter. The legend indicates data points where masking was installed but switched off, and switched on with varying numbers of bad channels stored in the database (0, 612 and 1371). The point at an RoI width of 0.60 with masking turned on and zero bad channels is slightly slower than the other points in that series, because it was measured before a minor optimization.

Ryan Taylor	TrigTimerSvc EFMissingET_Fex		Release 14.0.0
	Masking OFF (ms)	Masking ON (ms)	Relative increase
Trial 1			
EMLoadCollections	19.1	23.0	0.20
HECLoadCollections	2.75	2.82	0.03
FCalLoadCollections	0.446	0.528	0.18
TotalTime	63.9	68.4	0.07
Trial 2			
EMLoadCollections	18.3	23.6	0.29
HECLoadCollections	2.69	2.83	0.05
FCalLoadCollections	0.436	0.544	0.25
TotalTime	62.6	68.4	0.09

Figure 4.5: Summary of the increase in execution time due to masking bad channels on each event, after all optimizations. The feature extraction algorithm for EF MET is used as a gauge. The total time increase due to turning on masking is about 8%.

Chapter 5

Results and Conclusions

5.1 Results of Bad-Channel Masking Tests

5.1.1 Software Tests Using Simulated Data

The first demonstration of the bad-channel masking was a simple proof-of-concept test. In this test, the method of masking cells was the same as described in Section 4.1.3 - cells were masked by omitting them from the map which associates cells to trigger towers. However, the LArBadChannelMasker and LArBadChanTool tools were not yet fully developed; instead, a decision was hard-coded in the LArCellCont container to arbitrarily mask cells based on their identifiers. About 14% of the cells in the top half ($\phi > 0$) of the LAr calorimeter were masked in this way, distributed sparsely and fairly uniformly. Two types of simulated QCD di-jet data were used for this test: di-jets with energy ranging from 35 to 70 GeV (denoted J2), and from 280 to 560 GeV (denoted J5). The ATLAS data simulation does not emulate the effects of problematic channels, so the cells masked in this exercise are nominally good; however, the purpose of this software test was only to demonstrate that the mechanism of masking cells worked as intended. Figure 5.1 shows that masking 7% of the cells in the calorimeter causes a commensurate decrease in the scalar sum of deposited transverse energy of about 7%, as expected. Since all the masked cells are

in the region of positive ϕ, the direction of the MET vector is accordingly biased in that direction, as shown in Figure 5.2.

In a later test, the cell-level energy differences due to masking were examined. This test also exercised the ASCII-based method of input to the LArBadChanTool. A text file containing a preliminary list of bad channels (identified by the LAr Commissioning community using calibration runs and cosmic ray data) was parsed and loaded into the LArBadChanTool. Most of the bad channels identified at this time were in end-cap A. Figure 5.3 shows that most of the masked energy is accordingly in the region $\eta > 1.50$. This confirms that the correct channels are being masked.

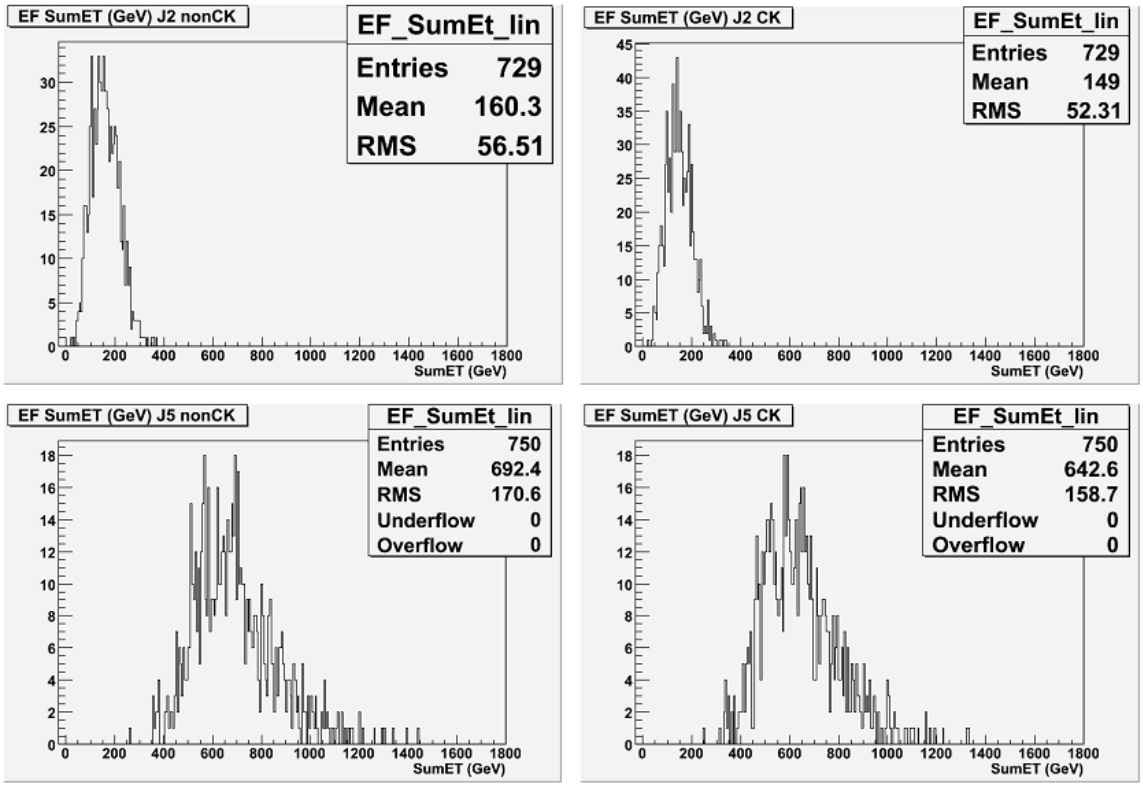

Figure 5.1: SumET distributions without (left) and with (right) cell masking. Two types of simulated QCD di-jet data were used; J2 (top) and J5 (bottom). The means of the SumET distributions decrease by about 7% when 7% of cells are masked arbitrarily.

5.1.2 Validation Using Cosmic Ray Data

The bad-channel masking underwent a more realistic test using cosmic-ray data from the M7 (Milestone 7) commissioning runs. In run 69373, there were several miscalibrated FEBs in addition to noisy channels, which caused large spikes in the (η, ϕ) distributions of L2 photons. By masking the miscalibrated FEBs and noisy channels, the spikes are removed from the distribution of photons, significantly improving the data quality, as seen in Figure 5.4.

5.2 Summary

A software framework for cataloguing and querying the status of LAr channels has been developed, and put to use in the HLT data preparation code to provide two mechanisms for masking problematic LAr channels. These tools have been stringently optimized in order to minimize the impact on the execution speed of HLT algorithms. Following the testing and validation described in Section 5.1, the per-run method of cell masking was enabled in ATLAS production software. The LAr cell masking is now ready to be used when the LHC starts running, to safeguard the HLT from instrumental effects and improve the data quality of events recorded for physics analysis.

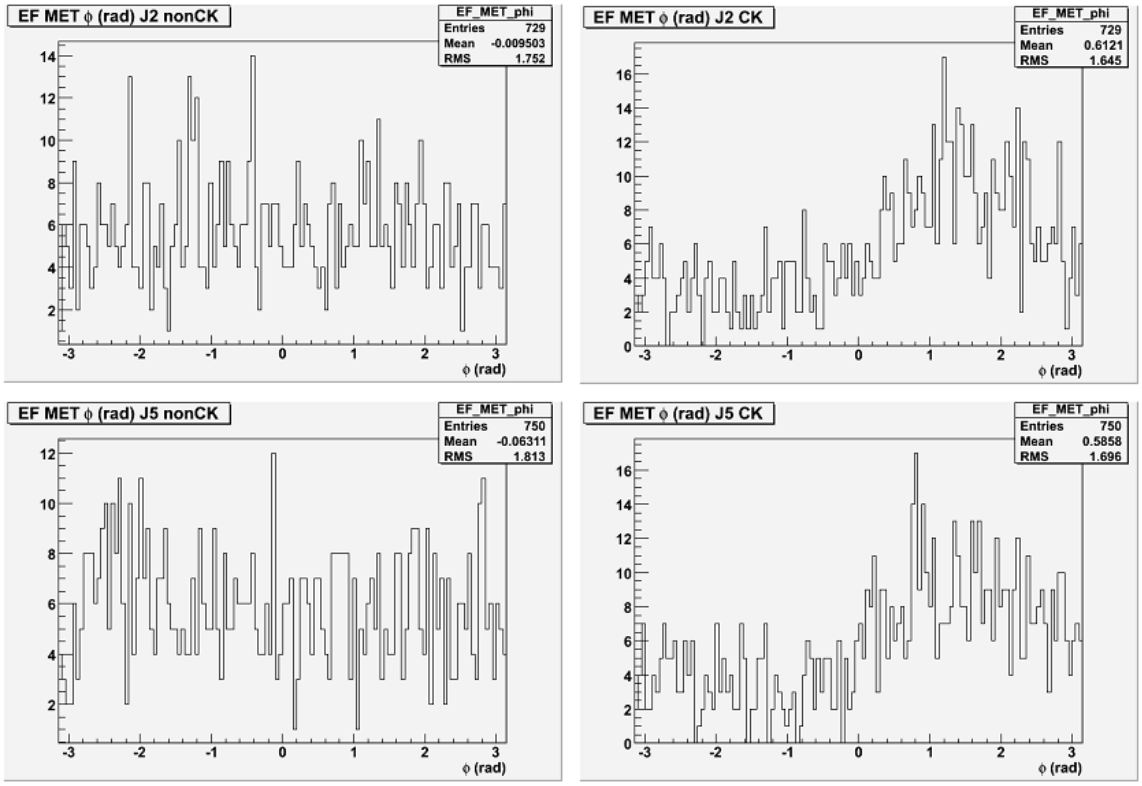

Figure 5.2: Distributions in ϕ of MET without (left) and with (right) cell masking. Two types of simulated QCD di-jet data were used; J2 (top) and J5 (bottom). This software test confirms that arbitrarily masking 14% of the cells in the top half ($\phi > 0$) of the LAr calorimeter results in a bias of the direction of MET towards positive ϕ.

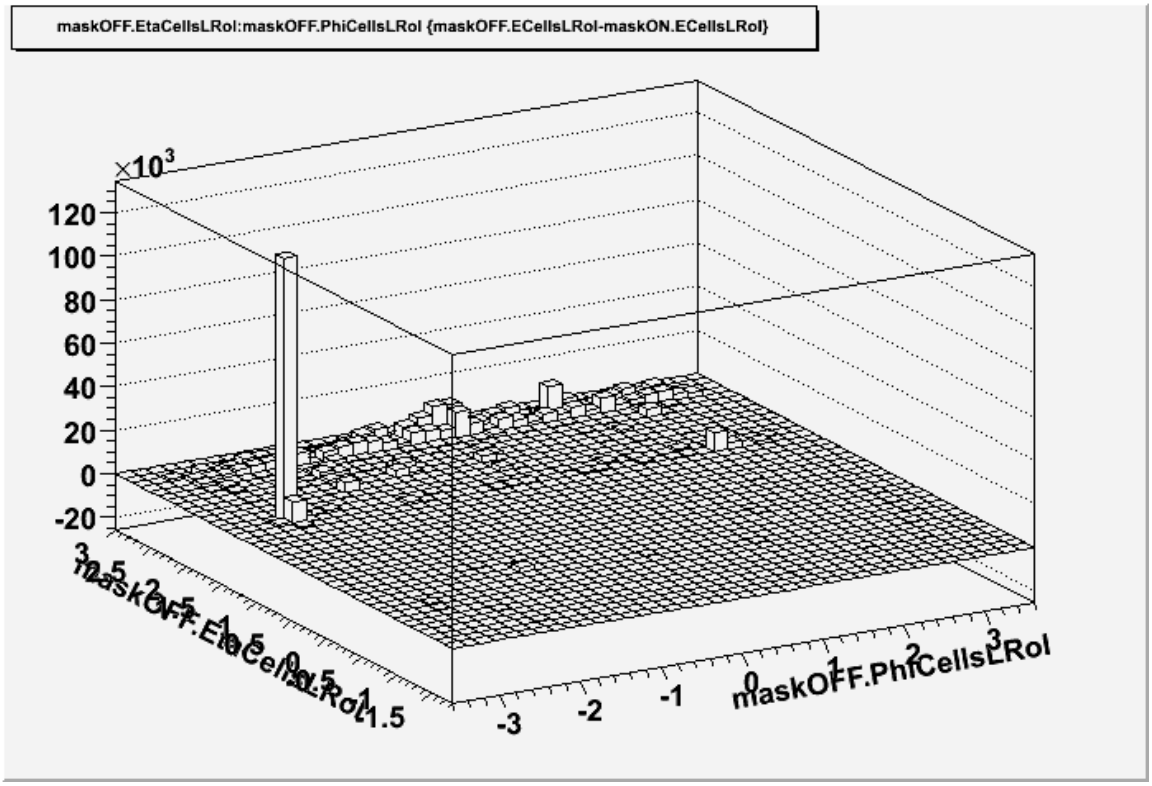

Figure 5.3: Lego plot showing the change in energy due to masking (cell energies without masking minus cell energies with masking). η is on the left axis and ϕ is on the right axis. The unit of energy is MeV. Twenty J5 simulated di-jet events were used.

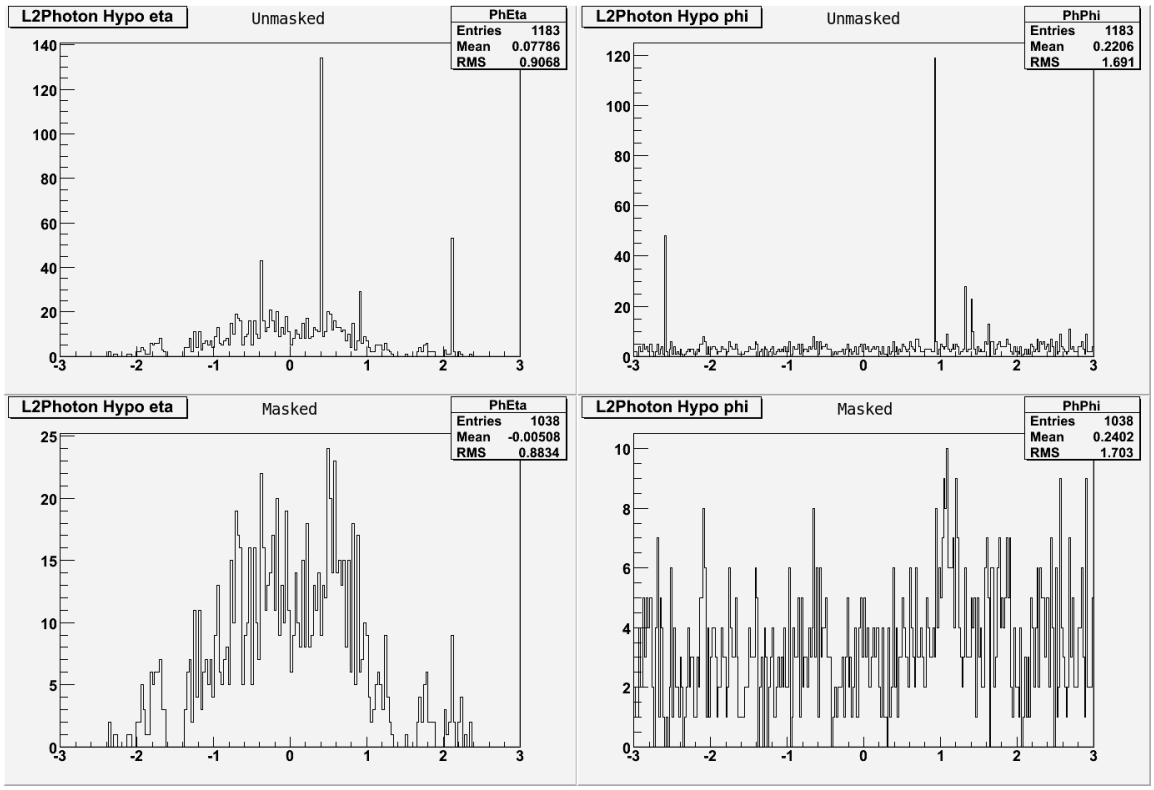

Figure 5.4: Distributions in η (left) and ϕ (right) of photons in L2 from cosmic ray data, without (top) and with (bottom) masking.

Bibliography

[1] W.-M. Yao *et al.* "Higgs Bosons: Theory and Searches." J. Phys. G **33** (2006) and 2007 partial update for the 2008 edition available on the PDG webite:

http://pdg.lbl.gov/2007/reviews/higgs_s055.pdf

[2] S. Dimopoulos and H. Georgi. "Softly Broken Supersymmetry and SU(5)." Nucl. Phys. B **193** (1981): 150-162.

[3] G. Polesello and D. R. Tovey. "Constraining SUSY Dark Matter with the ATLAS detector at the LHC." JHEP Issue 05 (2004): 12pp.

[4] A. Gupta (for the ATLAS Collaboration). "Search of extra space dimensions with ATLAS." Pramana **62** Issue 3 (2004): 607-610.

[5] D. Rousseau. "CP Violation with the ATLAS detector." Nucl. Phys. B, Proc. Suppl. **75** Issue 3 (1999): 351-355.

[6] L.R. Evans. "The Large Hadron Collider." Proceedings of the 1995 Particle Accelerator Conference, Vol. 1 (1995): 40-44.

http://epaper.kek.jp/p95/ARTICLES/FPD/FPD04.PDF

[7] O. Bruning, P. Collier, P. Lebrun, S. Myers, R. Ostojic, J. Poole, P. Proudlock. "LHC Design Report, Volume 1: The LHC Main Ring." CERN-2004-003-V-1, CERN-2004-003 (2004): 556pp.

[8] O. Bruning, P. Collier, P. Lebrun, S. Myers, R. Ostojic, J. Poole, P. Proudlock. "LHC Design Report, Volume 2: The LHC Infrastructure and General Services." CERN-2004-003-V-2, CERN-2004-003 (2004): 230pp.

[9] M. Benedikt, P. Collier, V. Mertens, J. Poole, K. Schindl. "LHC Design Report, Volume 3: The LHC Injector Chain." CERN-2004-003-V-3, CERN-2004-003 (2004): 356pp.

[10] J. M. Campbell, J. W. Huston and W. J. Stirling. "Hard interactions of quarks and gluons: a primer for LHC physics." Rep. Prog. Phys. **70** (2007): 89-193.

[11] The ATLAS Collaboration. "The ATLAS Experiment at the CERN Large Hadron Collider." JINST **3** S08003 (2008): 407pp.

[12] K. Kordas *et al.* "The ATLAS Data Acquisition and Trigger: concept, design and status." Nucl. Phys. B, Proc. Suppl. **(**172) (2007): 178-182.

[13] ATLAS HLT/TDAQ/DCS Group. "ATLAS High-Level Trigger, Data Acquisition and Controls Technical Design Report." CERN/LHCC/2003-22 (2003): 364pp.

http://atlas-proj-hltdaqdcs-tdr.web.cern.ch/atlas-proj-hltdaqdcs-tdr/

Appendix A

Attribution of Work

The LAr bad-channel software framework was designed and implemented by T. Todorov. I contributed by coding and testing the LArBadChannelParser, a class which parses channel information from ASCII files and provides it to the LArBadChanTool as input. I then expanded the functionality of the LAr bad-channel software framework by proposing, designing, implementing and testing the LArBadChannelMasker, a useful and efficient interface to the LArBadChanTool. In addition, I assisted T. Todorov with ongoing maintenance and debugging of the LAr bad-channel software, and a few minor improvements.

In order to apply the bad-channel tools to mask channels for the HLT, I worked with D. Damazio in the area of the HLT calorimeter data preparation code. This entailed testing the methods of masking cells, and conducting extensive timing tests and profiling to assess the efficiency of the per-event masking operations. I identified several opportunities for optimization in the LAr bad-channel code (including the LArBadChannelMasker), and continued to collaborate with T. Todorov to quicken it.[1]

[1] I also brought about significant optimizations in the MET feature extraction code and the calorimeter cell code, but those optimizations are not directly relevant to, nor described in, this thesis.

Appendix B

List of Abbreviations

CERN	European Organization for Nuclear Research
DAQ	Data Acquisition
EF	Event Filter
EMB	Electromagnetic Barrel calorimeter
EMEC	Electromagnetic End-cap calorimeter
FCal	Forward Calorimeter
FEB	Front-end Board
FT	Feedthrough
HEC	Hadronic End-cap calorimeter
HLT	High-Level Trigger
LAr	Liquid Argon
LHC	Large Hadron Collider
L1	Level-1 Trigger
L2	Level-2 Trigger
MET	Missing E_T; Missing Transverse Energy
OF	Optimal Filtering
OFC	Optimal Filtering Coefficient
QCD	Quantum Chromodynamics, Quantum Chromodynamical
ROB	Readout Buffer
ROD	Readout Driver
RoI	Region of Interest
SCA	Switched-capacitor Array
SFI	Sub-Farm Input
SFO	Sub-Farm Output
TT	Trigger Tower