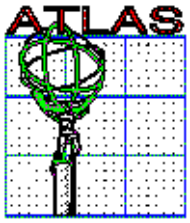


ATLAS
DAQ, EF, LVL2 and DCS

Technical Progress Report

Issue:	1
Revision:	0
Reference:	CERN / LHCC 98–16
Created:	30 June 1998
Last modified:	18 August 1998
Prepared By:	ATLAS Collaboration

ATLAS Collaboration

Armenia
Yerevan Physics Institute, Yerevan

Australia
Research Centre for High Energy Physics, Melbourne University, Melbourne
University of Sydney, Sydney

Austria
Institut für Experimentalphysik der Leopold-Franzens-Universität Innsbruck, Innsbruck

Azerbaijan Republic
Institute of Physics, Azerbaijan Academy of Science, Baku

Republic of Belarus
Institute of Physics of the Academy of Science of Belarus, Minsk
National Centre of Particle and High Energy Physics, Minsk

Brazil
Universidade Federal do Rio de Janeiro, COPPE / EE / IF, Rio de Janeiro

Canada
University of Alberta, Edmonton
Department of Physics, University of British Columbia, Vancouver
University of Carleton / C.R.P.P., Carleton
Group of Particle Physics, University of Montreal, Montreal
Department of Physics, University of Toronto, Toronto
TRIUMF, Vancouver
University of Victoria, Victoria

CERN
European Laboratory for Particle Physics (CERN), Geneva

Czech Republic
Academy of Sciences of the Czech Republic, Institute of Physics and Institute of Computer Science, Prague
Charles University, Faculty of Mathematics and Physics, Prague
Czech Technical University in Prague, Faculty of Nuclear Sciences and Physical Engineering, Faculty of Mechanical Engineering, Prague

Denmark
Niels Bohr Institute, University of Copenhagen, Copenhagen

Finland
Helsinki Institute of Physics, Helsinki

France
Laboratoire d'Annecy-le-Vieux de Physique des Particules (LAPP), IN2P3-CNRS, Annecy-le-Vieux
Université Blaise Pascal, IN2P3-CNRS, Clermont-Ferrand
Institut des Sciences Nucléaires de Grenoble, IN2P3-CNRS-Université Joseph Fourier, Grenoble
Centre de Physique des Particules de Marseille, IN2P3-CNRS, Marseille
Laboratoire de l'Accélérateur Linéaire, IN2P3-CNRS, Orsay
LPNHE, Universités de Paris VI et VII, IN2P3-CNRS, Paris
CEA, DSM / DAPNIA, Centre d'Etudes de Saclay, Gif-sur-Yvette

Republic of Georgia
Institute of Physics of the Georgian Academy of Sciences and Tbilisi State University, Tbilisi

Germany
Physikalisches Institut, Universität Bonn, Bonn
Institut für Physik, Universität Dortmund, Dortmund
Fakultät für Physik, Albert-Ludwigs-Universität, Freiburg
Institut für Hochenergiephysik der Universität Heidelberg, Heidelberg
Institut für Physik, Johannes-Gutenberg Universität Mainz, Mainz
Lehrstuhl für Informatik V, Universität Mannheim, Mannheim
Sektion Physik, Ludwig-Maximilian-Universität München, München
Max-Planck-Institut für Physik, München
Fachbereich Physik, Universität Siegen, Siegen
Fachbereich Physik, Bergische Universität, Wuppertal

Greece
Athens National Technical University, Athens
Athens University, Athens
High Energy Physics Department and Department of Mechanical Engineering, Aristotle University of Thessaloniki, Thessaloniki

Israel
Department of Physics, Technion, Haifa
Raymond and Beverly Sackler Faculty of Exact Sciences, School of Physics and Astronomy, Tel-Aviv University, Tel-Aviv
Department of Particle Physics, The Weizmann Institute of Science, Rehovot

Italy
Dipartimento di Fisica dell' Università della Calabria e I.N.F.N., Cosenza
Laboratori Nazionali di Frascati dell' I.N.F.N., Frascati
Dipartimento di Fisica dell' Università di Genova e I.N.F.N., Genova
Dipartimento di Fisica dell' Università di Lecce e I.N.F.N., Lecce
Dipartimento di Fisica dell' Università di Milano e I.N.F.N., Milano
Dipartimento di Scienze Fisiche, Università di Napoli 'Federico II' e I.N.F.N., Napoli
Dipartimento di Fisica Nucleare e Teorica dell' Università di Pavia e I.N.F.N., Pavia
Dipartimento di Fisica dell' Università di Pisa e I.N.F.N., Pisa
Dipartimento di Fisica dell' Università di Roma 'La Sapienza' e I.N.F.N., Roma
Dipartimento di Fisica dell' Università di Roma 'Tor Vergata' e I.N.F.N., Roma
Dipartimento di Fisica dell' Università di Roma 'Roma Tre' e I.N.F.N., Roma
Dipartimento di Fisica dell' Università di Udine, Gruppo collegato di Udine I.N.F.N. Trieste, Udine

Japan
Department of Information Science, Fukui University, Fukui
Hiroshima Institute of Technology, Hiroshima
Department of Physics, Hiroshima University, Higashi-Hiroshima
KEK, High Energy Accelerator Research Organisation, Tsukuba
Department of Physics, Faculty of Science, Kobe University, Kobe
Department of Physics, Kyoto University, Kyoto
Kyoto University of Education, Kyoto-shi
Department of Electrical Engineering, Nagasaki Institute of Applied Science, Nagasaki
Naruto University of Education, Naruto-shi
Department of Physics, Faculty of Science, Shinshu University, Matsumoto
International Center for Elementary Particle Physics, University of Tokyo, Tokyo
Physics Department, Tokyo Metropolitan University, Tokyo
Department of Applied Physics, Tokyo University of Agriculture and Technology, Tokyo

Morocco
Faculté des Sciences Aïn Chock, Université Hassan II, Casablanca, and Université Mohamed V, Rabat

Netherlands
FOM - Institute SAF NIKHEF and University of Amsterdam / NIKHEF, Amsterdam
University of Nijmegen / NIKHEF, Nijmegen

Norway
University of Bergen, Bergen
University of Oslo, Oslo

Poland
Henryk Niewodniczanski Institute of Nuclear Physics, Cracow
Faculty of Physics and Nuclear Techniques of the University of Mining and Metallurgy, Cracow

Portugal
Laboratorio de Instrumentação e Física Experimental de Partículas (University of Lisboa, University of Coimbra, University Católica-Figueira da Foz and University Nova de Lisboa), Lisbon

Romania
Institute of Atomic Physics, Bucharest

Russia
Institute for Theoretical and Experimental Physics (ITEP), Moscow
P.N. Lebedev Institute of Physics, Moscow
Moscow Engineering and Physics Institute (MEPhI), Moscow
Moscow State University, Institute of Nuclear Physics, Moscow
Budker Institute of Nuclear Physics (BINP), Novosibirsk
Institute for High Energy Physics (IHEP), Protvino
Petersburg Nuclear Physics Institute (PNPI), Gatchina, St. Petersburg

JINR
Joint Institute for Nuclear Research, Dubna

Slovak Republic
Bratislava University, Bratislava, and Institute of Experimental Physics of the Slovak Academy of Sciences, Kosice

Slovenia
Jozef Stefan Institute and Department of Physics, University of Ljubljana, Ljubljana

Spain
Institut de Física d'Altes Energies (IFAE), Universidad Autónoma de Barcelona, Bellaterra, Barcelona
Physics Department, Universidad Autónoma de Madrid, Madrid
Instituto de Física Corpuscular (IFIC), Centro Mixto Universidad de Valencia - CSIC, Valencia

Sweden
Fysiska institutionen, Lunds universitet, Lund
Royal Institute of Technology (KTH), Stockholm
University of Stockholm, Stockholm
Uppsala University, Department of Radiation Sciences, Uppsala

Switzerland
Laboratory for High Energy Physics, University of Bern, Bern
Section de Physique, Université de Genève, Geneva

Turkey
Department of Physics, Ankara University, Ankara
Department of Physics, Bogaziçi University, Istanbul

United Kingdom
School of Physics and Astronomy, The University of Birmingham, Birmingham

Cavendish Laboratory, Cambridge University, Cambridge
Department of Physics and Astronomy, University of Edinburgh, Edinburgh
Department of Physics and Astronomy, University of Glasgow, Glasgow
Department of Physics, Lancaster University, Lancaster
Department of Physics, Oliver Lodge Laboratory, University of Liverpool, Liverpool
Department of Physics, Queen Mary and Westfield College, University of London, London
Department of Physics, Royal Holloway and Bedford New College, University of London, Egham
Department of Physics and Astronomy, University College London, London
Department of Physics and Astronomy, University of Manchester, Manchester
Department of Physics, Oxford University, Oxford
Rutherford Appleton Laboratory, Chilton, Didcot
Department of Physics, University of Sheffield, Sheffield

United States of America
State University of New York at Albany, New York
Argonne National Laboratory, Argonne, Illinois
University of Arizona, Tucson, Arizona
Department of Physics, The University of Texas at Arlington, Arlington, Texas
Lawrence Berkeley Laboratory and University of California, Berkeley, California
Department of Physics, Boston University, Boston, Massachusetts
Brandeis University, Department of Physics, Waltham, Massachusetts
Brookhaven National Laboratory (BNL), Upton, New York
University of Chicago, Enrico Fermi Institute, Chicago, Illinois
Nevis Laboratory, Columbia University, Irvington, New York
Department of Physics, Duke University, Durham, North Carolina
Department of Physics, Hampton University, Virginia
Department of Physics, Harvard University, Cambridge, Massachusetts
Indiana University, Bloomington, Indiana
University of California, Irvine, California
Massachusetts Institute of Technology, Department of Physics, Cambridge, Massachusetts
University of Michigan, Department of Physics, Ann Arbor, Michigan
Michigan State University, Department of Physics and Astronomy, East Lansing, Michigan
University of New Mexico, New Mexico Center for Particle Physics, Albuquerque
Physics Department, Northern Illinois University, DeKalb, Illinois
Department of Physics and Astronomy, University of Oklahoma
Department of Physics, University of Pennsylvania, Philadelphia, Pennsylvania
University of Pittsburgh, Pittsburgh, Pennsylvania
Department of Physics and Astronomy, University of Rochester, Rochester, New York
Institute for Particle Physics, University of California, Santa Cruz, California
Department of Physics, Southern Methodist University, Dallas, Texas
State University of New York at Stony Brook, Stony Brook, New York
Tufts University, Medford, Massachusetts
High Energy Physics, University of Illinois, Urbana, Illinois
Department of Physics, Department of Mechanical Engineering, University of Washington, Seattle, Washington
Department of Physics, University of Wisconsin, Madison, Wisconsin

Acknowledgements

The Editors would like to thank Mario Ruggier for preparing the FrameMaker template upon which this document is based. The Editors also warmly thank Richard Cook and Susan Leech-O'Neale for copy-editing the document, Arlette Coudert for the front cover, and the CERN Print-shop staff for the helpful, friendly and efficient service. Our thanks to all those who have helped in the production of the material for this document, but who are not named in the institutes list. Finally, our thanks to Ian Brawn for his efforts in the last few days to ensure the readiness of the document.

Table Of Contents

ATLAS Collaboration . . . . . **iii**

Acknowledgements . . . . . **vii**

1 Summary **1**
1.1 Introduction 1
1.1.1 Physics requirements 1
1.1.2 Basic architecture 1
1.1.3 System requirements and technology considerations 3
1.2 Current status 4
1.3 Progress report 4
1.3.1 The LVL2 trigger system 5
1.3.1.1 Functional description of the LVL2 components 6
1.3.1.2 Issues and constraints 7
1.3.1.3 Development work 8
1.3.2 The DAQ/EF system 9
1.3.2.1 Detector interface 10
1.3.2.2 Dataflow 11
1.3.2.3 Event filter 12
1.3.2.4 Back-end DAQ 13
1.3.2.5 Test beam 14
1.3.3 Detector control system 14
1.4 Outstanding issues and workplans 14
1.4.1 The LVL2 trigger system 14
1.4.2 The DAQ/EF system 16
1.4.3 Detector control system 16
1.5 Integration plans within trigger/DAQ and with other systems. 16
1.6 References 17

2 Introduction **19**
2.1 Purpose and scope of the technical progress report 19
2.2 Back-up documents 20

3 General description **21**
3.1 Physics requirements. 21
3.2 System overview 24
3.2.1 Definition of sub-systems. 24
3.2.2 Interface to front-end systems 27
3.2.3 Interface to LVL1 28
3.2.4 Dataflow functional view. 29
3.3 Factorization of system studies (based on functions) 31
3.4 References 31

4 Level-2 trigger **33**
4.1 Introduction 33
4.1.1 Requirements for the level-2 trigger 33

4.1.1.1 Event selection . . . 34
4.1.1.2 Operational modes and partitioning . . . 34
4.1.2 Functional description of the level-2 components . . . 35
4.1.2.1 The ROB complex . . . 35
4.1.2.2 The supervisor and RoI builder . . . 35
4.1.2.3 The networks . . . 35
4.1.2.4 The processing farms . . . 35
4.1.3 Issues and constraints . . . 36
4.1.4 Project phases . . . 38
4.2 Input parameters . . . 39
4.2.1 Information from LVL1 . . . 39
4.2.1.1 Information provided by LVL1 to LVL2 . . . 39
4.2.2 Information from the ROBs . . . 42
4.3 Selection process and strategies . . . 42
4.3.1 Introduction . . . 42
4.3.2 Muon feature extraction . . . 43
4.3.3 Calorimeter feature extraction . . . 45
4.3.3.1 e / g feature extraction. . . . 46
4.3.3.2 Tau / hadron feature extraction. . . . 46
4.3.3.3 Jet feature extraction. . . . 47
4.3.3.4 Calculation of and . . . 47
4.3.4 Inner detector feature extraction (TRT, SCT and Pixel) . . . 47
4.3.4.1 B-tagging algorithm . . . 49
4.3.5 Benchmarking of algorithms . . . 50
4.3.6 From features to trigger objects, and decisions . . . 51
4.3.7 Trigger menus . . . 53
4.3.8 An example of sequential selection . . . 53
4.4 Options for architecture and technologies . . . 54
4.4.1 The architectures . . . 54
4.4.2 The components and their technologies. . . . 59
4.4.2.1 Supervisor. . . . 60
4.4.2.2 Readout buffer . . . 62
4.4.2.3 Networks . . . 63
4.4.2.4 Processors . . . 64
4.4.2.5 Software components . . . 64
4.4.3 Summary . . . 65
4.5 Results from the R&D programme . . . 65
4.5.1 Overview and goals . . . 65
4.5.2 Demonstrator prototypes . . . 66
4.5.2.1 Vertical slice and component work . . . 66
4.5.2.2 ROB issues . . . 76
4.5.2.3 Network issues and results . . . 83
4.5.2.4 Event supervision . . . 89
4.5.2.5 Event processing . . . 94
4.5.3 Modelling and emulation . . . 101
4.5.3.1 Paper models. . . . 101

4.5.3.2 Computer simulation 106
4.5.3.3 Emulation 108
4.5.4 Conclusions of the Demonstrator Programme 112
4.6 Options retained and plan up to the Technical Proposal. 114
4.6.1 Definition of the pilot project 114
4.6.2 LVL2 primary critical issues. 115
4.6.2.1 Data collection 115
4.6.2.2 Data flow optimization 116
4.6.2.3 Large system aspects 116
4.6.2.4 Conclusions and guidelines for the future 117
4.6.3 Activity matrix: general overview. 117
4.6.4 Functional components 118
4.6.4.1 ROB complex. 118
4.6.4.2 Supervisor. 120
4.6.4.3 Processors, interfaces and networks 120
4.6.5 Software test beds 121
4.6.5.1 Reference test bed 122
4.6.5.2 Application testbeds 123
4.6.6 System design 125
4.6.6.1 Modelling 125
4.6.6.2 Integration. 126
4.7 References 127

5 Data acquisition and event filter 131
5.1 Introduction. 131
5.2 The DAQ/EF -1 Project 133
5.2.1 Detector interface 134
5.2.1.1 Definition, purpose and goals of the detector interface . 134
5.2.1.2 Areas of work. 134
5.2.2 Dataflow 136
5.2.2.1 The dataflow system in prototype-1 136
5.2.2.2 Relevance to the final system 137
5.2.2.3 Boundaries with other parts of the system 138
5.2.2.4 Dataflow high-level design 139
5.2.2.5 Dataflow development 144
5.2.2.6 Dataflow integration 162
5.2.3 Event filter. 163
5.2.3.1 Definition of the EF event filter system 163
5.2.3.2 Functions and requirements. 164
5.2.3.3 Event filter high-level design 170
5.2.3.4 Prototypes, benchmarking and modelling 174
5.2.4 Back-end DAQ subsystem 179
5.2.4.1 Overview 179
5.2.4.2 Operational environment. 181
5.2.4.3 Back-end DAQ software components 181
5.2.4.4 Trigger / DAQ and detector integration components . . 183

5.2.4.5 Software technologies . . . 184
5.2.4.6 Software process . . . 188
5.2.4.7 Software development environment . . . 190
5.3 Use of the DAQ/EF-1 system . . . 191
5.3.1 Dataflow . . . 191
5.3.1.1 Introduction . . . 191
5.3.1.2 ROC performance . . . 192
5.3.1.3 Event builder studies . . . 195
5.3.2 Event filter . . . 202
5.3.2.1 Introduction . . . 202
5.3.2.2 Event filter software and physics strategy . . . 202
5.3.2.3 Prototype construction and development . . . 203
5.3.2.4 Additional functionality . . . 204
5.3.2.5 EF / dataflow integration . . . 204
5.3.2.6 EF / back-end DAQ integration . . . 205
5.3.3 Back-end DAQ . . . 205
5.3.3.1 Introduction . . . 205
5.3.3.2 Core component unit tests . . . 206
5.3.3.3 Configuration database . . . 208
5.3.3.4 Message reporting system and information service. . . . 211
5.3.3.5 Process manager . . . 213
5.3.3.6 Trigger/DAQ and detector integration components . . . 214
5.3.3.7 Integration tests . . . 215
5.3.3.8 Further development . . . 216
5.4 The test beam . . . 217
5.5 References . . . 219

6 Detector Control System . . . 225
6.1 Scope of the DCS . . . 225
6.2 Architecture . . . 225
6.3 Prototypes . . . 226
6.3.1 Configuration database . . . 227
6.3.2 Supervisor system . . . 227
6.3.3 Front-end I/O via a fieldbus . . . 227
6.3.4 Interim stand-alone controls system . . . 228
6.3.5 Embedded CAN systems . . . 228
6.4 Organizational points . . . 229
6.5 References . . . 230

7 Plans for work up to the technical proposal. . . . 231
7.1 Objectives for the technical proposal . . . 231
7.2 Analysis of requirements . . . 232
7.2.1 Trigger performance . . . 232
7.2.2 Detector interface . . . 233
7.3 Integration and validation strategies . . . 233
7.4 Integration and validation plans for LVL2 trigger . . . 234

7.4.1 Work plan overview 234
7.4.2 Functional components workplan 235
7.4.2.1 ROB complex 237
7.4.2.2 RoI builder and multi-processor supervisor 237
7.4.2.3 Technologies for processors, interfaces and networks . . 237
7.4.3 Testbed workplan 238
7.4.3.1 Reference software algorithms 238
7.4.3.2 Reference software framework 239
7.4.3.3 Application testbeds 239
7.4.4 System design workplan 240
7.4.4.1 Modelling and emulation. 240
7.4.4.2 Integration 241
7.5 Integration and validation plans for DAQ/event filter 241
7.5.1 Dataflow integration and validation 243
7.5.1.1 Readout crate studies 243
7.5.1.2 Event Builder studies 245
7.5.1.3 Sub-farm studies. 246
7.5.1.4 Dataflow global studies 246
7.5.2 Back-end DAQ integration and validation 247
7.5.2.1 Unit testing of core components 247
7.5.2.2 Trigger/DAQ and detector integration components . . 248
7.5.2.3 Global back-end DAQ integration. 248
7.5.2.4 Further developments 249
7.5.3 Event filter 250
7.5.3.1 Event filter software and physics strategy 250
7.5.3.2 Prototype construction and development 250
7.5.3.3 Additional functionality 251
7.5.4 Dataflow—back-end DAQ integration 251
7.5.5 Event filter Integration in the dataflow 251
7.5.6 Event filter Integration with the back-end DAQ 252
7.6 Detector Control System 252
7.7 External systems 252
7.7.1 Detectors 252
7.7.2 LVL1 trigger 252
7.7.3 Data recording 253
7.7.4 Offline reconstruction and physics analysis code 253
7.8 Overall integration plan 253
7.8.1 DAQ/EF – LVL1 trigger integration 253
7.8.2 LVL2 trigger – LVL1 trigger integration. 254
7.8.3 LVL2 trigger – DAQ/EF integration 254
7.8.4 DCS – DAQ/EF integration 254
7.8.5 DAQ/EF – first detector integration 255
7.8.6 DAQ/EF – data recording and offline integration 255
7.8.7 DAQ/EF – other detectors integration 255
7.9 References 256

8 Project organisation and management 257
8.1 Participating institutes 257
8.2 Responsibilities and work organisation 259
8.3 Management organisation 260
8.4 Schedule and milestones 261
8.5 Cost and resources 261
8.6 LVL2 pilot project organization 261
8.7 DAQ/EF-1 organization. 263
8.8 References 266

A Appendix: definitions, acronyms and abbreviations 267

1 Summary

1.1 Introduction

This document describes and discusses the work of the data acquisition and event filter (DAQ/EF), the level-2 (LVL2) trigger, and the detector control system (DCS) groups since the ATLAS Technical Proposal [1-1], and presents work plans for the future, concentrating on the period up to the high level triggers and DAQ Technical Proposal. A technical design report on the level-1 (LVL1) trigger [1-2] and a status report on trigger performance [1-3] are published at the same time.

The task of the trigger/DAQ system is to select interesting physics in an efficient and controlled way, and to move the data produced by the ATLAS detector for these events to permanent storage for later analysis. To meet the required event rate reduction, of about 10^7, between the primary interactions in the detector and the event rate that can be written to permanent storage, the event selection (trigger) function of the ATLAS trigger/DAQ system is organized in three levels, see Figure 1-1.

1.1.1 Physics requirements

Much of the ATLAS physics programme at high luminosity can be carried out using very inclusive triggers, at least at the earlier selection stages. These include inclusive selections of events containing high-p_T muons, photons, electrons, taus, hadrons and jets, as well as events with very large missing transverse energy or total scalar transverse energy.

In the later selection stages, inclusive selections can be made of leptonic W and Z decays, making use of invariant mass information in the case of Z decays and missing transverse energy in the case of W decays. An advantage of inclusive triggers is that one may hope that they cover new, unpredicted physics. This is very important given the fact that LHC will explore a new energy regime.

It is assumed that the initial luminosity at LHC will be about 10^{33} $cm^{-2}s^{-1}$, which we refer to as low luminosity. During this initial phase, the search for new physics will start in parallel with an extensive programme of B physics. The B-physics programme requires the use of complex signatures in the later selection stages.

1.1.2 Basic architecture

As stated above, the trigger is organized in three levels. At LVL1, special-purpose processors act on reduced-granularity data from a subset of the detectors. LVL2 uses full-granularity, full-precision data from the detectors, but, for most triggers, examines only regions of the detector identified by the LVL1 trigger as containing interesting information. At the third trigger level, the event filter (EF), the full event data are used together with the latest available calibration and alignment information to make the final selection of events to be recorded for offline analysis.

The LVL1 trigger accepts data at the full LHC bunch-crossing rate of 40 MHz and provides a decision for each bunch crossing. The latency of the LVL1 trigger system (that is the time taken to

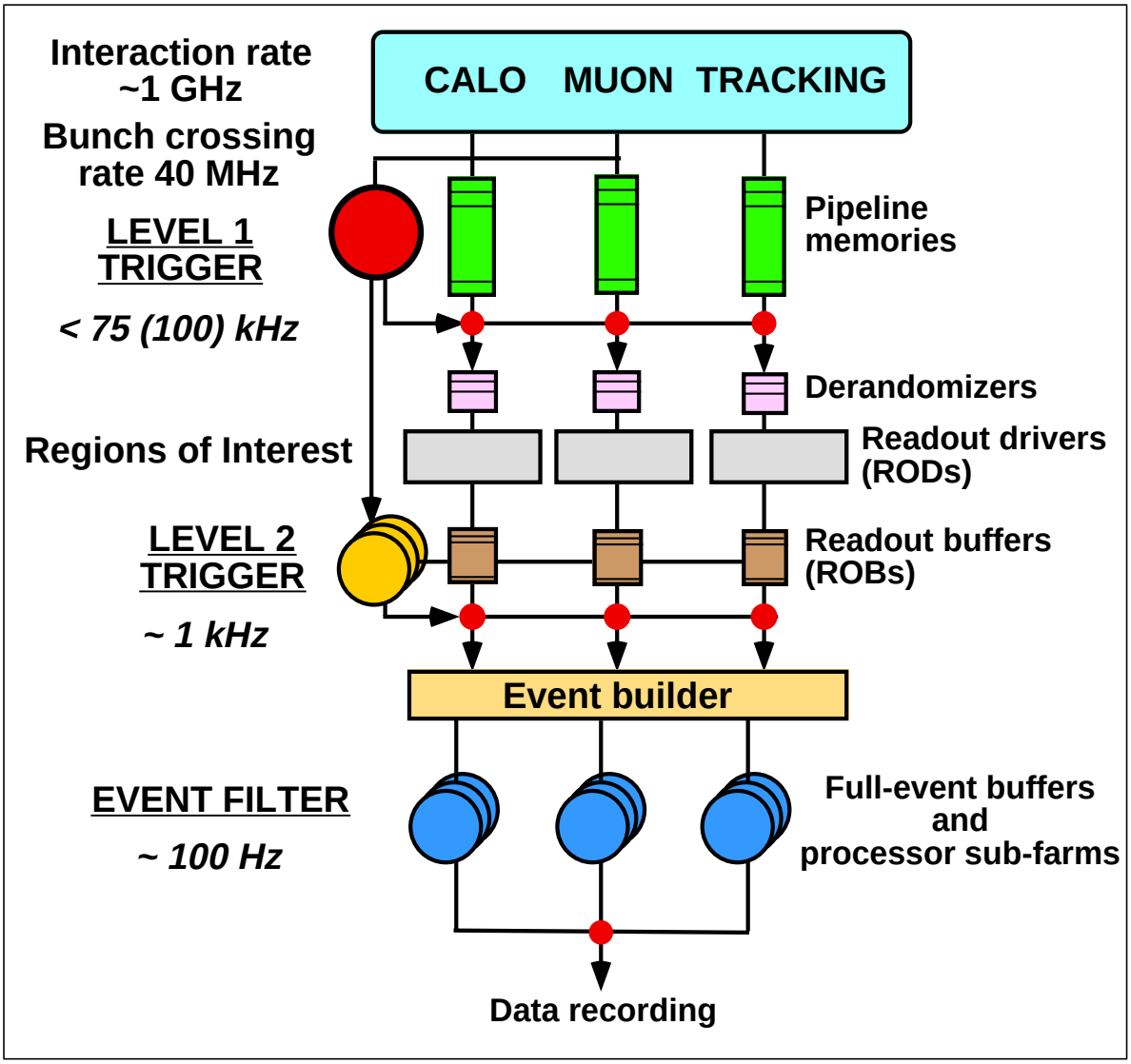


Figure 1-1 Three levels of the ATLAS trigger.

collect data, form the LVL1 trigger decision and distribute it) is ~2 μs, and all detector data are held in pipeline memories during this period. The maximum accept rate of the LVL1 trigger is set at 75 kHz and is determined by the capabilities of the subdetector readout systems. ATLAS requires that it must be possible to upgrade these systems to operate at 100 kHz with a somewhat higher dead time (a few per cent) and, therefore, the DAQ and LVL2 trigger systems must be designed to accept this higher input rate.

After a LVL1 trigger accept, data are moved off the detector and stored in readout buffers (ROBs) during the LVL2 trigger processing and the event-builder collection time.

The LVL2 trigger has to reduce the acceptance rate to ~ 1 kHz. Its architecture is based on the use of regions of interest (RoIs). The LVL1 trigger is used to identify, for each event, regions of the detector containing interesting features such as high-p_T electromagnetic clusters, jets and muons. The LVL2 trigger then has to access and process only a small fraction of the total detector data, with corresponding advantages in terms of the required processing power and data-movement capacity. LVL1 also provides LVL2 with information about global event properties

— the total scalar E_T and the missing-E_T vector — and it specifies which signatures led to the event being selected by the LVL1 trigger.

The LVL2 trigger uses full-precision information from the inner tracking detectors, as well as from the calorimeters and muon detectors. Data from the subdetectors are combined to provide better particle identification and higher measurement precision than are possible in the LVL1 trigger. This is necessary if the rate reduction required at LVL2 is to be achieved. The average decision time for the LVL2 trigger is estimated to be ~10 ms.

After an event is accepted by the LVL2 trigger, all the data for that event are sent to an EF processor via the event builder. Complete event reconstruction is possible at this third trigger level and decision times of around one second are expected. The EF system must achieve a data-storage rate of the order of 100 Mbyte/s by reducing the event rate and/or the event size. For some triggers the full event data of about 1 Mbyte will need to be recorded, implying a maximum event rate of 100 Hz, while for others a reduced readout is sufficient, allowing a higher event recording rate.

The boundary of operations between the LVL2 trigger and the EF is overlapped during the R&D phase so that the trade-off of moving event selection between the two can be examined. This boundary is likely to remain flexible during running to optimize the physics performance of ATLAS.

The DAQ system is responsible for dataflow through the system from the front-end to permanent storage but, to allow flexibility for the subdetectors, a clean interface has been established at the ROBs and it is planned to have only one ROB design for all ATLAS. From the front-end to this buffer, and within some general constraints and agreements between the detector and the trigger/DAQ communities, the detector communities are free to pursue their own solutions. The LVL2 trigger and the event builder only access detector data from the ROBs and thus a clean interface is obtained. Based on readout links running at around 1 Gbit/s, the current estimate of the number of links required is around 1700.

Because of the size and the complexity of the ATLAS detector a dedicated control system is needed for its operation. The main task of the DCS is the supervision and monitoring of the ATLAS detector in the widest sense and the acquisition, treatment and storage of certain non-physics-event data such as temperatures and pressures, which are needed to understand the behaviour of the detector and which impact on the physics analyses.

1.1.3 System requirements and technology considerations

The total amount of data to be acquired, formatted, moved, monitored and checked drives the trigger/DAQ architectural decisions, the technology choices on which they are based, the number of data links, and the amount of processing power needed. Not all of the detector requirements are well established or fully understood. It is clear, however, that they place an unprecedented challenge on the trigger/DAQ system.

From work to date we know that for both hardware and software:

- The system must be based on innovative architectural solutions; classical ones could not provide the required performance.
- At many levels the trigger/DAQ process seems to require technologies which are either not available or whose utilization we have not, as yet, fully mastered.

Therefore, we are not yet ready to approach the final design of the trigger / DAQ system, which would be premature given the time-scale of the experiment. Entering this phase too early would imply a number of risks:

- The system might be obsolete before it is built and / or inadequate and unmaintainable once it is built.
- Instead of being based on a sound architectural description, design decisions may be driven by specific technologies and / or components, as has often happened in the past.
- Over-designing of components or subsystems may occur, as we have seen when expensive brute-force solutions are used in systems where the design has been inadequate.
- Under-designing of components or subsystems may occur as well, which would limit the performance of the complete system.
- We wish to take full advantage of the continual improvement in performance and prices of electronics and computing systems by specifying the components, hardware and software as late as possible.

The above considerations are valid equally for the DAQ, the LVL2 trigger and the EF.

1.2 Current status

Before the complete trigger / DAQ system can be designed it is necessary to understand the requirements of each part of the system. Therefore, we have put in place programmes, for each part, to investigate the requirements and their implications for architectures and technologies.

For the LVL2 trigger, the programme has looked at a range of options to solve the problem of a decision frequency of up to 100 kHz combined with a rate reduction of a hundred, see Chapter 4. Further work is required on the technology options, and only now are the software and system management problems being studied in depth.

The EF is, functionally, an integral part of the main DAQ dataflow. Events being processed by the EF program are stored in the EF processor memory before being discarded or accepted for permanent storage. Therefore, the DAQ and EF are considered as one system (DAQ / EF). For the DAQ / EF system, a preprototype programme, the DAQ / EF-1 project, has been developed, see Chapter 5. This project aims at producing a system with full functionality but not necessarily the full performance required for the final system.

The DCS, see Chapter 6, must operate in close conjunction with the DAQ but is operationally independent. Prototype work is going on to address a number of pressing issues.

1.3 Progress report

The descriptions given below include the current state of understanding of the systems being worked on. It should be appreciated that reaching this level of knowledge represents a considerable amount of work and effort. More detail is given in the relevant chapters and in the back-up documents and other references.

1.3.1 The LVL2 trigger system

The normal mode of operation for the LVL2 trigger system for high-p_T triggers is based on the use of regions of interest (RoIs), defined by the LVL1 trigger, see Figure 1-2. A region of interest is signalled by LVL1 for candidate high-p_T muons, electrons / photons, taus / hadrons, and jets. The information on the RoIs, in particular their position in η and ϕ and the LVL1 selection criteria, are made available to LVL2. Based on this information, LVL2 decides which data from each subdetector are associated with each of the RoIs, and only these data are transferred to the LVL2 processors. This selection of data for the LVL2 trigger reduces the load on the LVL2 system and allows for substantial simplification of LVL2 algorithms.

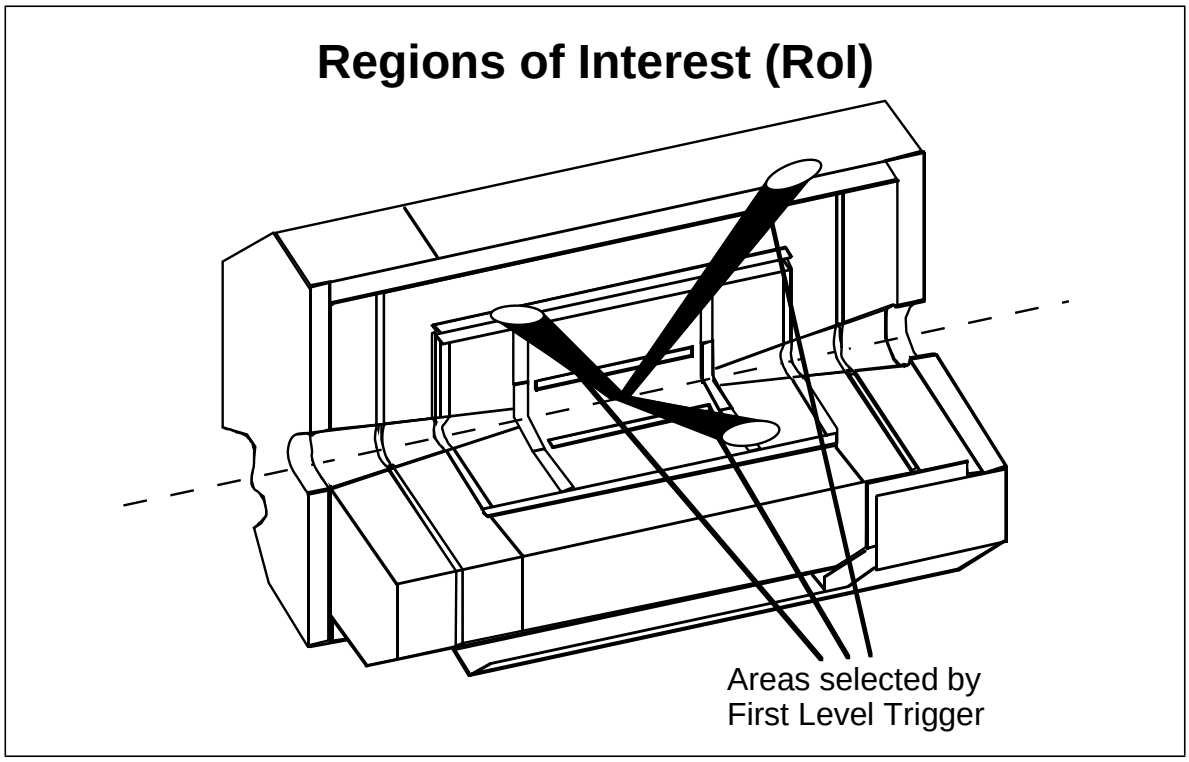


Figure 1-2 Region of interest (RoI).

At low luminosity, the LVL2 trigger system will process both high-p_T and B-physics event candidates simultaneously. B-physics events, e.g. for CP-violation studies, are selected at LVL1 by requiring a high-p_T muon (p_T>6 GeV). Thus, the only RoI passed to LVL2 for these events is likely to be that of the LVL1 trigger muon. The mode of operation of the LVL2 trigger for such events is a sequence of steps. The first step is to use data from the trigger muon RoI to confirm the muon (About half of the LVL1 muon triggers are expected to be rejected at LVL2, mostly from applying a sharper p_T cut.) Once the trigger muon is confirmed, data from other subdetectors are required to test the LVL2 B-physics hypotheses, without any RoI guidance from LVL1 being available. A full scan in one of the tracking systems (assumed to be the TRT) is used to identify tracks, and these then define the RoIs for selection of data from the other subdetectors for further analysis.

Event processing at LVL2 can be decomposed into a number of broad steps: feature extraction (FEX), object building and trigger-type selection. In FEX, the data for one RoI for one detector are gathered together and processed to give physics-like quantities (e.g. for the calorimeter, this process would take cell information and produce cluster parameters; for a tracker, the basic hit

information would be converted to track or track-segment parameters). Object building takes the features for one RoI from all relevant detectors and returns the particle parameters and possibly the particle type. If the data are consistent with more than one particle type, subsequent processing performs tests with each of the possible object types. In the trigger type selection phase, all the objects found in the event are combined and compared with the topologies (particle types, momenta, inclusive/missing mass, etc.) expected for a menu of physics selections. Flags are set for each menu item for which a match is found.

Relatively simple trigger menus are sufficient to demonstrate that manageable rates can be achieved while satisfying the goals of the ATLAS physics programme (see Chapter 3). It is anticipated that much more extensive menus will be used in practice and more detailed menus are needed for certain studies, e.g. when comparing different implementation strategies.

1.3.1.1 Functional description of the LVL2 components

The LVL2 system can be described in terms of four main functional blocks: the ROB complex, the supervisor and RoI builder, the networks, and the processing farms.

The ROB complex
For events accepted by LVL1, the data from all the front-end electronics channels are transferred to the ROBs. The ROB is a standard functional component common to both LVL2 and DAQ/EF dataflow. Its main function is to store raw data fragments during the LVL2 selection procedure.

For the communication with the LVL2 system there is a control interface and a data interface. The control interface receives requests for data and LVL2 decisions from the LVL2 system. The data interface sends the requested data from the ROBs to the LVL2 system. The data interface may contain options for preprocessing data within or local to the ROBs and options for combining data from several ROBs (via a bus or network) prior to transmission.

The supervisor and RoI builder
The supervisor and RoI builder are tightly coupled. The task of the RoI builder is to receive the RoI information fragments from LVL1 at the LVL1 accept rate and build the RoI record for each event. This record contains a list of RoIs with their types and η, ϕ positions. One of the principal functions of the supervisor is to assign LVL2 processors and to route the RoI record to the appropriate place or places in the LVL2 system; another is to receive and broadcast LVL2 decisions to the ROB complex.

The networks
There are two logical networks. The data collection network that transmits the RoI data to the various processing elements and the control network that carries LVL1 accept, LVL1 RoI request and LVL2 accept/reject messages. Depending on the architecture and implementation choices each of these networks could be implemented as multiple physical networks or they could be combined into a single physical network for all of the traffic.

The processing farms
Corresponding to the main types of algorithm used in the LVL2 selection steps, there are three main tasks to be done in the processor elements:

- feature extraction, which processes data from a single RoI and single detector to extract features such as calorimeter cluster energy and track parameters.
- object building, which combines the features for a single RoI from the different detectors and returns the particle properties and, possibly, type.

- global processing, which combines the objects and compares them with trigger menu items to produce the LVL2 accept or reject decision.

1.3.1.2 Issues and constraints

Several issues are driving the overall study of the LVL2 system.

The LVL2 system must be flexible, allowing different scenarios of event selection and changes of the boundary with the EF, as the physics interest evolves and the luminosity of the LHC machine increases.

On the technical side, the continual development of the 'high technology' components required for the system promises major benefits.

The constant evolution of RISC/CISC processors continues to follow Moore's law, with computing power approximately doubling every 18 months. The 300–500 MIPS of today's standard commercial processors would extrapolate to 2–4 GIPS at the start of LHC operation and even more for high-luminosity running, indicating that the pure computing power for FEX and global processing will not be a major issue. Similarly the size and power of FPGA devices continues to grow, and already they have demonstrated adequate power for some of our requirements.

The memory chips used in the ROBs are becoming larger and cheaper, with the consequence that a 10 ms LVL2 latency is no longer considered to be a major issue.

The aggregate bandwidth required for the total LVL2 network traffic is estimated to be of the order of several Gbyte/s. This bandwidth cannot be handled by traditional bus-based data-acquisition systems. However, high-end commercial network components promise performance levels of the order required and, if current trends continue, at an affordable price. Building such a large, high-performance network will still be a complex task, and to attain the bandwidth and latency requirements will place extreme demands on the components and the configuration. To gain the necessary knowledge and experience it is necessary to perform detailed evaluations and comparisons of these technologies.

The complete LVL2 system will be very complex and will require long-term maintenance. In order to reduce the overall development phase and facilitate the maintenance, it is strongly desirable to use commercially available components in a modular configuration, wherever possible. In addition compliance with widely adopted standards will assist the interoperability (software and hardware) of equipment from different vendors.

Data extraction from ROBs and data movement to the processors is a major issue. From the hardware side, the interfaces and input/output (I/O) devices between the different high-speed elements (links and networks) are in general the critical areas to be optimized. From the software side, the performance of device drivers, and various overheads such as context switches and interrupt handling will need to be optimized.

The extraction and collection of the RoI data from the ROBs imposes critical constraints on the ROB design and other components. The number of ROBs which contribute to each RoI and the rate at which each ROB is accessed are strongly coupled to the mapping of the detector elements into the ROBs. Thus details of the mapping of each subdetector into the ROBs could have a major impact on the LVL2 system performance. It is therefore important that they are chosen

to minimize the overhead of extracting and transferring the restricted data required by LVL2. This is a strong requirement from the LVL2 and a real issue if not satisfied.

The format of event fragment data may affect the data volume to be transferred within the LVL2 system and / or the performance of the FEX algorithms. In some cases reformatting of the data will be desirable before transfer across the network. Optimization of the format coming from the detectors could thus make important savings in bandwidth and minimize the data manipulation in the ROBs or LVL2 system. In some cases optimization could be achieved by preprocessing at the ROD level, but clearly this would influence the ROD design. The balance between extra complexity of the RODs and benefits from the ROB data extraction and formatting is only partially understood at this stage and requires further studies.

1.3.1.3 Development work

Work has concentrated in a number of areas:

- algorithm development and benchmarking,
- a demonstrator programme of architecture and technology studies,
- modelling and emulation[1].

The LVL2 trigger code must be optimized for fast execution using a minimum amount of information from the RoIs defined by the LVL1 trigger. The trigger selections will change with the LHC luminosity, more importance being given to the low-p_T channels in the initial, lower-luminosity running.

The RoI data will be processed by FEX algorithms to extract quantities which are used in the global decision. The selected methods will be simpler and less iterative than in offline code and the algorithms must not depend on the ultimate resolution of the detector since the full set of calibration data may not be available at the trigger level. The LVL2 trigger performance work is described elsewhere [1-3] and in this document we concentrate on the methods and implementation of the FEX algorithms and their performance on a range of machines.

Standalone versions of algorithms have been implemented and optimized on different hardware and software platforms. A library of the online code written in C or C++ (FEX algorithms for muons, electrons, gammas, taus / hadrons, jets, total and missing E_T, including combined reconstruction of trigger objects) is available for testing various trigger strategies on test beds. Benchmarking results for the library code have been obtained for each algorithm description.

The demonstrator programme, running from late 1996 to February 1998, studied components for a number of representative architectures and different technologies. These components could be combined to form various architectures, but were used primarily for the following:

a. FPGA-based FEX followed by object building and global processing in a general-purpose processor farm.

b. A number of parallel farms extract features for each detector, followed by a second step where features are combined in a global farm.

c. All of the algorithm processing for an event is performed sequentially in a single processor in one processor farm connected by a single network.

1. See Appendix A for a short description of some terms used in this document.

These options provide a range of solutions to be adopted to match the technology available at the time of construction.

In all cases, the demonstrations in the form of thin vertical slices of the architectures worked successfully and, where appropriate, simple models of the slices reproduced the results obtained.

Modelling and emulation

With the wide variety of choices facing the demonstrator programme in architectures, technologies and event-selection strategies, a fast method of assessing the impact of these options was needed.

A spreadsheet package has been used for simple calculations of the rates, loads, occupancies and latency of the trigger system: the so-called ‘paper models’. Average numbers are used to represent the input events and data and hardware are modelled as generically as possible by simple parametrizations. The results give a preliminary indication of potential bottlenecks and hardware requirements, and provide input to computer modelling and emulation, but cannot take into account the contention and queuing likely to occur in real systems. Calculations have been performed for single and multiple farm architectures and for several event-selection strategies. As well as providing input to computer models, results obtained highlighted the high loading of the hadron calorimeter ROBs and the importance of a low I/O overhead in processors.

Computer modelling is vital for extrapolating results from the small prototype systems (which we can afford to construct in the laboratory) to the large systems which will be required by ATLAS. A C++ simulation program has been developed and used where, at present, generic components are modelled. Results have been obtained which show agreement with the paper models when the same conditions are imposed and, therefore, provide considerable confidence in the veracity of the model. When realistic operating conditions were applied problems associated with contention for data from the hadron calorimeter ROBs were seen at event rates well below those predicted by the paper models. Similar results were obtained from the emulation of the networks, where ATLAS-like LVL2 trigger traffic was run through a large (1024-node) switch. The ability of this large network to handle LVL2 trigger traffic provides confidence in our basic architectural design and in the computer modelling results for a large system.

1.3.2 The DAQ/EF system

From the architectural viewpoint the DAQ/EF system can be seen as comprising four main subsystems: the interface to the other ATLAS systems, the dataflow, the back-end DAQ and the EF. The interface to other systems comprises the specification of the boundaries and the requirements that the other systems place on the DAQ/EF system. Dataflow is responsible for the transport, management, and monitoring of the ATLAS physics data, from the detector readout electronics to permanent storage. The back-end DAQ encompasses all the software for configuring, controlling and monitoring the DAQ system. The EF is responsible for the last stage of event selection before permanent storage.

Rather than develop a set of ad hoc components, we have chosen to build a fully-functional pre-prototype system, called DAQ/EF-1, covering the complete range from detector readout to data recording (see Figure 1-3). This prototype is the basis for predesign architectural studies, tech-

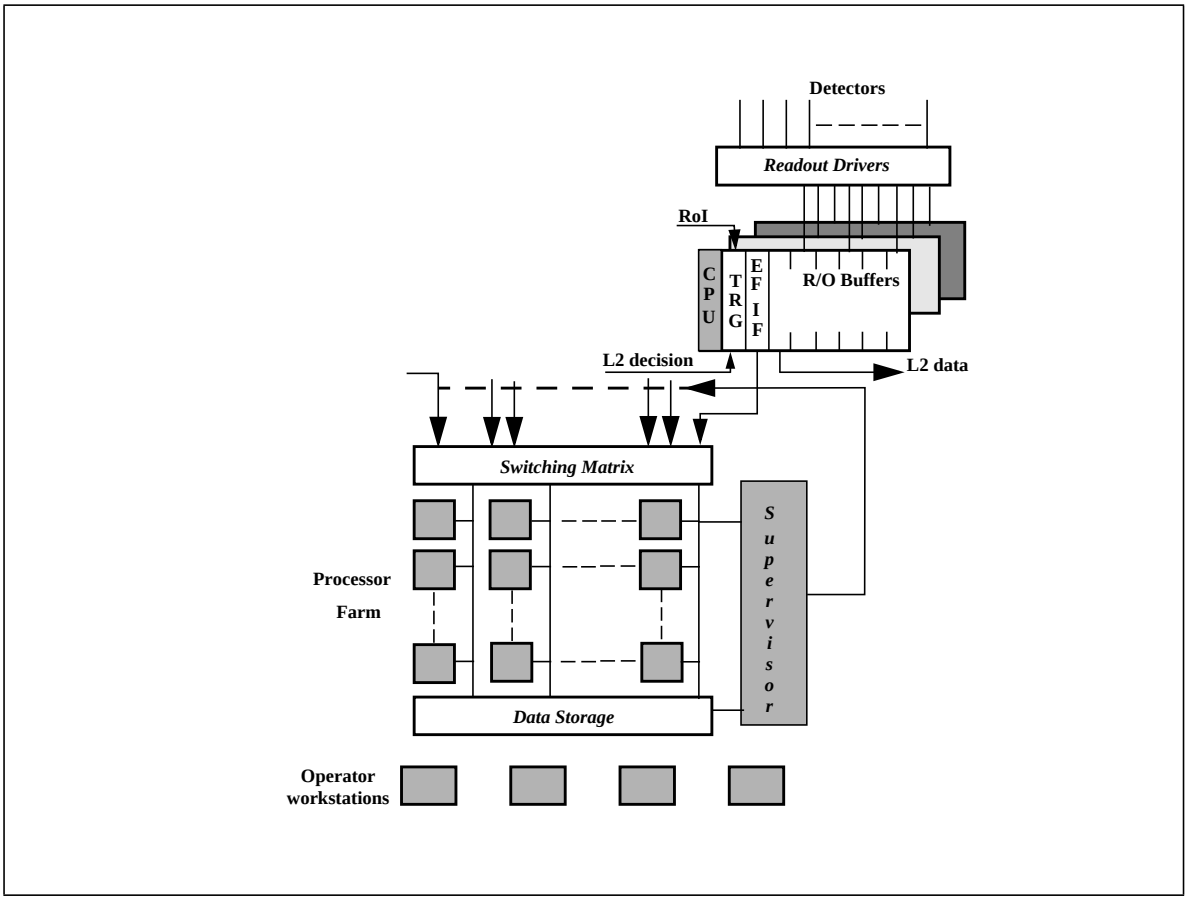


Figure 1-3 The DAQ/EF Prototype-1 system architecture.

nology (hardware and software) evaluations and comparisons, performance measurements, and assessments of solutions. It is also the means to study global system requirements.

Work on the DAQ/EF-1 system is in three stages, namely, predesign and high-level design studies, detailed design and implementation, and system integration and exploitation. Currently the project has reached the start of the integration phase.

1.3.2.1 Detector interface

The principal aim of the detector-interface working group is to collect together and understand the detector requirements on the DAQ system, and to attempt to map them as closely as possible to the final requirements of the ATLAS experiment. There are many areas where detectors expect certain services, standards, or constraints from the DAQ system, and vice versa. The definition and clarification of these areas is vital for the successful integration of the entire experiment.

The group, set up in September 1996 by the ATLAS DAQ community, includes a representative for each of the ATLAS systems. The first stage of the work was concluded with the release of a summary document.

The areas of work of the detector interface working group can be identified in several broad and overlapping sections: front-end interfaces and control; monitoring, calibration and databases;

system partitioning and non-physics triggers; event and data formats and data recording; run control debugging and testing.

1.3.2.2 Dataflow

Three main functions are provided by the dataflow: the collection and buffering of data from the detector (the front-end DAQ); the merging of fragments into full events (the event builder); and the interaction with the EF. In addition to the event-building function we have identified two building blocks, the readout crate (ROC) and the subfarm DAQ.

The ROC is the modular element responsible for handling the movement of data between the readout links and the event builder. It performs the following principal functions: detector readout, buffering and data distribution to other dataflow elements in the crate; control of the flow of data within the crate; assembly of crate fragments from event fragments; and, ancillary functions such as error handling.

The event builder merges the event fragments from the ROC into a complete event at a destination. While different technologies might be used to implement the actual transfer and merging of data into full events, performance considerations indicate that this will be implemented by means of a switching network, allowing concurrent merging of events. It is one of the objectives of DAQ/EF-1 to try out different commercial technologies for the event-builder switching network.

The subfarm DAQ receives full events from the event builder and sends them to the EF. It also records events sent from the EF for output to mass storage.

Partitioning and event format are issues common throughout the DAQ/EF-1 system and, in particular, across the dataflow system. The structure of the data at various stages is defined by the event format. It defines how data of a ROB, crate fragment, or subdetector may be directly accessed by processing tasks. Moreover, it defines additional data that are added to the detector data (such as event tags) by elements of the dataflow to allow processing tasks to identify, for example, the type and origin of the event.

Dataflow developments

A number of initial selections have been made in the area of the ROC. VMEbus is used as the crate integration bus and for intracrate links. PCI has been chosen as the I/O bus within an I/O module, with the PMC format preferred for I/O interfaces. I/O modules are implemented using VME-compatible PowerPC-based processor cards with two (or more) PMC sites.

Hardware libraries have been developed to support the generic I/O module. Studies of operating systems for real-time operations have been performed. Intracrate communications protocols have been defined and prototyped, and a local control and monitoring subsystem designed. Based on these hardware choices and software developments, a prototype ROC has been built and used to test design ideas and measure some relevant performance parameters.

A model of the ROC has been simulated so that rate-limiting factors and bottlenecks can be studied. Good agreement is obtained between the model and measured parameters.

For event-builder studies, a number of switching technologies have been assessed and conclusions drawn from the measurements obtained. The tests show that current technologies provide the necessary components but that interfaces and scalability of the systems are still issues and

need to be supported by modelling studies. Additional tests with larger systems and emerging technologies are planned.

Other aspects for study are event-building protocols, reliability, additional performance measurements and the role of the dataflow manager.

The local DAQ (LDAQ) subsystem provides all the common DAQ functions which are not related to moving event data. A prototype implementation of the ROC LDAQ has been made. The same LDAQ prototype could be used in the subfarm DAQ. Studies of operating systems for real-time applications including the possibility of using PCs and/or the WindowsNT operating system, at least in the area of the LDAQ, have also been performed.

1.3.2.3 Event filter

The EF system implements the LVL3 trigger processing strategy as outlined in the ATLAS Technical Proposal. Many processes will be selected through multiple trigger signatures, thus providing optimal efficiency and several means of controlling the crucial aspects of the trigger efficiency. Physics studies performed since the Technical Proposal have warranted more work to show that the strategy still covers most of the physics goals.

Access to geometry and calibration constants, data decoding and transformation, and full event reconstruction will all be required before it is possible to apply the selection algorithms to the event. As far as possible, offline code will be used in the EF in order both to economize effort and to allow direct comparisons between offline and online performance.

In addition to its primary task of event filtering and data reduction, the EF, being the first element in the DAQ chain having access to complete events, will also be used for monitoring and calibration/alignment studies.

The EF will be highly CPU-bound with data I/O representing a small fraction of the overall latency. Extrapolations indicate a minimum required EF processing power of 10^6 MIPS. A farm of commercial processors is considered to offer the best approach to implement a solution but any final choice of architecture for the EF should be delayed as long as possible. Candidate solutions include farms of PC-style processors, commercial high-end computing solutions based on high-speed processors connected together with a high-performance bus or switch, and MPP machines.

The EF is organized as a set of independent subfarms, each with a direct connection to the event-building switching fabric. One of the goals of the DAQ/EF-1 project is to reproduce a functional vertical slice of the final ATLAS DAQ system. In this context, the EF functionality will be represented by a small number of prototype subfarms, each capable of handling ~1% of the final EF input load.

Event filter developments

Two prototypes based on the contrasting architectures of network-interconnected PCs, and symmetric multiprocessors are currently being implemented. Each prototype will be implemented as a single subfarm with a view to integrating it into the DAQ/EF-1 architecture. This will enable us to compare and contrast the various prototypes in a homogeneous and realistic environment. Issues of physics algorithm performance, system management and control, reliability, error handling, and scalability will be investigated. The prototypes will also be the subject

of system modelling studies in order to try to extend the knowledge gained from the prototypes to larger scale implementations of the same architectures.

The detailed study of prototypes, physics benchmarks and complementary modelling will be the major area of activity in the EF group in the coming two years.

1.3.2.4 Back-end DAQ

The back-end software encompasses the software for configuring, controlling, and monitoring the DAQ but specifically excludes the management, processing, and transportation of physics data. It must coexist and co-operate with the other subsystems. In particular, interfaces are required to the many subsystems of the DAQ and external entities. These comprise the trigger, for configuration, processor farms, the LHC beam information, the event builder, the LDAQ system and the DCS. In many cases the data transfers are bi-directional.

It is expected that the operational environment for the back-end software will be a heterogeneous collection of UNIX workstations, PCs running Windows NT, and embedded systems (LDAQ processors) running various flavours of real-time UNIX operating systems connected via a local area network (LAN). It is assumed that network connections running the most popular protocols will be available to all the target platforms for exchanging control information and that use of this network will not interfere with the physics data transportation performance of the DAQ.

The core components of the system are the run control, which controls data-taking activities; the configuration database, which holds information on the system architecture; a message reporting system; a process manager; and an information service. When integrated with other systems additional components are required, namely, the partition and resource manager, a status display, the run bookkeeper, an event dump, a test manager and a diagnostic package.

Back-end DAQ developments

Candidate freeware and commercial software packages were evaluated to find the most suitable product for each technology and selections made for a general-purpose toolkit, persistent data storage, a lightweight in-memory object manager and a database system. Systems have also been evaluated for interprocess communication, dynamic object behaviour and graphical user interfaces.

A software process (a set of tools, methods and practices) has been used to develop the back-end DAQ. The work has been divided into phases to help pace and organize the activity. The phases are to collect the requirements, initial investigations of candidate technologies, high-level design, detailed design (including implementation and unit testing), integration and deployment.

Within the back-end DAQ, elements of the software development environment, including object-orieted methods, CASE tools, and a configuration management system, have been assembled to cover the analysis, design, implementation, and testing phases of the software life cycle. Such elements have been selected to be compatible with similar work in the offline software groups.

Performance measurements have been made on many aspects of the system but further investigations are needed, including evaluation of alternative implementations of components and systems.

1.3.2.5 Test beam

As part of the validation process of the DAQ/EF architecture, an implementation of the DAQ/EF prototype will be used at the ATLAS test beam. The integration of the system with detectors, data recording and offline analysis will be achieved in a realistic and operational environment which the ATLAS test beam represents. The robustness, reliability and maintainability of the system will be demonstrated with a real application and real users.

1.3.3 Detector control system

The requirements of the DCS have been identified and compiled. As far as the connection to the rest of the DAQ system is concerned, operational independence is required, but a good interconnection is needed for the exchange of information and database parameters.

The overall architecture of the DCS is organized in three hierarchical layers: the overall supervisor and servers, the local control stations, and the front-end I/O system.

The front-end I/O system, which interfaces to the subdetectors, is the one which is needed first. Fieldbuses and programmable logic controllers (PLCs) will be employed and prototype work in this area has started. An interim standalone control system has been selected which assumes the functions of the upper layers. It allows both R&D work of the I/O system and tests of subdetector prototypes to be carried out.

1.4 Outstanding issues and workplans

Detailed workplans are in place for the period up to the Technical Proposal (December 1999) and they are described briefly below and in more detail in later chapters. The LVL2 activity is organized as one project which is divided into seven subprojects covering three broad areas. The DAQ/EF activity remains focused on the DAQ/EF-1 project and is divided into four work areas. The DCS team is working towards setting up a common LHC project.

The work remains factorized for the present but the workplan addresses issues of integration both within the trigger/DAQ area and with other systems. Close co-operation between LVL2 and DAQ/EF in the detector interface area is agreed as it is vital that the interface to the detectors is understood by all parties. Co-operation and integration between the trigger performance group and the LVL2 and EF groups is seen as another area vital to the success of the selection process and joint work programmes are being set up.

1.4.1 The LVL2 trigger system

The main activity for the LVL2 community up to the Technical Proposal is focused on the pilot project where work has been organized into studies on functional components, test beds and integration issues.

To allow us to select candidate technologies we will study the ROB complex, the supervisor and RoI builder, and processors, interfaces and networks.

To date, ROBs have been considered as individual items with an input from the detector and an output to the network. The number of ROBs is large (1700) and most of the LVL2 network traffic is the transfer of data from the ROBs to the processors. Work is under way to look at both how to reduce the physical size and packing density of the buffers, and how to optimize grouping of the buffers to allow data reduction by local processors before sending information over the network.

For the RoI builder the principal issue to be considered is the generation of the RoI records in the form needed by LVL2 at the rate delivered by LVL1 (up to 100 kHz). For the supervisor, the work programme will check the scaling of using multiple supervisor processors. Processor allocation strategies and event management issues — how best to perform the event process control tasks — will also be studied

Because of their specifications, the ROBs, the supervisor and the RoI builder are almost certain to contain specially designed and built components. On the other hand, we would prefer to use, where possible, commercial hardware and software as we see these offering advantages in terms of cost and maintenance and for providing a convenient upgrade path. To this end, we are evaluating systems composed of commodity networking and processing components. However, present indications are that some aspects of these systems do not meet our requirements and we are investigating these areas, which include interfaces, protocols and network scaling. We are continuing some work on current non-commodity systems which have attractive features and are keeping a watching brief on technological developments. FPGA-based systems will be developed and studied to see where they can make a cost-effective contribution to the system, for example, in preprocessing and coprocessing.

Up to now, LVL2 activity has concentrated on the basic architecture. A new programme, referred to as reference software development, has been set in place to integrate the software elements required for the LVL2 trigger process. This includes message-passing protocols, the framework for embedding algorithms and the event processing strategy, and the provision of training samples.

Application test beds are being set up to test the reference software and evaluate selection-strategy issues. These test beds are complete systems containing a fairly large number of processors (typically 32). In addition to the software and process testing and development, optimized components of hardware and software will be integrated into the test beds for assessment. The long-term aim of these test systems is to show the complete LVL2 process operating.

Paper modelling, using revised system parameters as they become available, will continue to be used to provide a first indication of the lower bound of performance requirements and starting values for the computer simulation studies. Computer modelling will be an important and critical activity up to the final system construction. The test systems constructed for the pilot project will provide parameters for the models used in the simulations. After showing that the simulations do reproduce the characteristics of the test systems, the models will be extrapolated to full system size to provide predictions of performance.

The information obtained from the various elements of the pilot project programme will be used to obtain a view of the complete LVL2 system. Included in this are studies of the consequences of an implementation on the rest of the experiment, integration of the system into the experiment, and costing issues.

1.4.2 The DAQ/EF system

The DAQ/EF-1 programme continues with the integration and exploitation phases being covered between now and the Technical Proposal.

As explained above, the detector interface group will continue the work to understand the detector parameters and requirements in close conjunction with LVL2.

For the dataflow, we are currently in the phase where the three subsystems, namely the ROC, the event builder and the subfarm DAQ, are ready to be integrated into a complete, baseline dataflow system. The first step of the workplan is the production of a baseline dataflow system. This system will be used as a basis for improvement and for studies aimed at evaluating and assessing DAQ architectural and operability issues. Two complementary areas will be studied: those which address local aspects such as a single element or subsystem and those which address the global aspects of dataflow.

A broad area of work is planned for the EF up to the Technical Proposal covering prototype, integration and physics studies. The prototype studies will be used to compare and contrast architectural alternatives and to develop the techniques for handling large data-processing farms. The EF will be integrated with the other elements of the DAQ/EF-1 prototype for testing and validation. Physics studies will be pursued to map offline software to the EF requirements. In addition, the LCB is setting up a project to study components common to event-filter farms.

The back-end DAQ subsystem of the DAQ/EF-1 project will be used from now until the Technical Proposal to determine if the architecture and functionality of the back-end software is suitable for the final ATLAS system. It is necessary to determine if the architecture delivers the necessary qualities of scalability, availability, performance and maintainability. Test-plans are being established for each component which will lead to a review of their suitability. The core components are expected to be in place by the end of 1998.

The component and integration tests are intended to provide a first version of the back-end system for use within the DAQ/EF-1 project. It is expected that integration with other sub-systems will lead to the identification of possible improvements which, coupled to evolving underlying technologies, will form the basis of further developments of the back-end components.

1.4.3 Detector control system

We participate actively in the CERN-wide joint controls project (JCOP) together with the controls group, IT/CO, and the other LHC experiments. The project aims to provide a common controls kernel for all LHC experiments.

Agreement is needed on the scope of the project, its organization, timescale, and resources, as well as on the main technical and implementation issues it should address. The project plan has to be established well before the end of 1998 if we are to place confidence in this approach.

1.5 Integration plans within trigger/DAQ and with other systems.

The plans for integration and validation are structured in several steps, see Chapter 7. The first step is to validate each component of the system, then to integrate subsystems and to test and

validate at each stage of integration. When each project area has a fully functional system, work will be put in place to integrate systems pair-wise before a global trigger / DAQ integration or integration with other systems. During the development stages, steps will be taken to inform other areas of the trigger / DAQ project of architectural and design considerations inside each area so that we can take advantage of common development needs and avoid unnecessary divergence between the systems.

1.6 References

1-1 ATLAS Technical Proposal, CERN / LHCC 94–43, LHCC / P2 (1994).

1-2 ATLAS LVL1 trigger technical design report, CERN / LHCC 98–14, ATLAS TDR 12 (1998).

1-3 ATLAS trigger performance status report, CERN / LHCC 98–15 (1998).

2 Introduction

2.1 Purpose and scope of the technical progress report

This document describes and discusses the work of the data acquisition and event filter (DAQ/EF), the level-2 trigger (LVL2) and the detector control system (DCS) groups since the ATLAS technical proposal. In addition, it presents work plans for the future, concentrating on the period up to December 1999 when the technical proposal for the DAQ and high level triggers (HLT) is scheduled.

The work of the level-1 trigger and the trigger performance groups are referenced here but are not described or discussed except in relation to the work of the groups given above. A technical design report on the level-1 trigger and a document covering the work on trigger performance are published at the same time as this document.

Chapter 3 provides the physics motivation for the work and an introduction to the requirements on the systems from the physics and the detector. The rationale behind the current factorization of the work into manageable projects is given and includes a general description of the aims and objectives of these projects.

The level-2 trigger work is presented in Chapter 4 and covers the current level of understanding of the LVL2 requirements and operations and the work in hand investigating outstanding issues. The parameters of the inputs from the detectors and the first level trigger are described and the selection processes and strategies being studied for the LVL2 are discussed. Options for architecture and technology are reviewed and results from R&D work are presented. The work programme on outstanding issues is described.

The work of the DAQ/EF group is described in Chapter 5. The requirements for the system are factorized into detector requirements, dataflow, event filter and back-end data acquisition, and the work programmes organized in each area are described and discussed. The DAQ/EF-1 project is introduced and the work on this and the results obtained are described. The plans for validating and integrating the various aspects of the project are discussed.

Chapter 6 discusses the design considerations for the detector control system and presents the current activity and plans in this area.

The reports on work performed and that planned for the LVL2, the DAQ/EF and the DCS programmes, Chapters 4, 5 and 6, have been produced by representatives of the respective communities.

The overall plans for work up to the technical proposal for the LVL2, the DAQ/EF and the DCS groups are presented in Chapter 7. Chapter 8 contains information on the groups involved, the organization and management of the trigger/DAQ group and the management of the projects.

A glossary of some terms, abbreviations and acronyms used in the document is supplied as an appendix.

2.2 Back-up documents

Documents have been produced covering in detail all aspects of the work performed and plans for future activity. These have been issued as ATLAS internal notes and are referenced at the end of each chapter. Other documents are referenced where relevant.

3 General description

3.1 Physics requirements

The ATLAS physics programme was outlined in the ATLAS technical proposal [3-1], and there is much work going on to evaluate in detail the physics potential of the experiment. An expanded and updated description of the ATLAS physics capabilities is in preparation [3-2]. The physics programme includes discovery physics such as searches for Higgs bosons and for supersymmetry, as well as precision, standard model physics such as measurements of the masses of the W boson and the top quark. It also includes B-physics studies, for example measurements of CP violation and the study of rare decay modes.

Trigger performance studies, covering both the LVL1 and the LVL2 triggers, as well as outline plans for the event filter, are reported in a separate document [3-3]. The requirements placed on the LVL2 trigger in terms of physics selections are described in the LVL2 user requirements document (URD) [3-4]. Here we only give a brief summary of the physics requirements.

Much of the ATLAS physics programme can be carried out using very inclusive triggers, at least at LVL1 and LVL2. These include inclusive selections of events containing high-p_T muons, photons, electrons, taus, hadrons and jets, as well as events with very large missing transverse energy or total transverse energy. Isolation will be required in the case of muons, photons, electrons, taus and hadrons, except at very high p_T. Lower thresholds are possible for multi-object triggers than for single-object ones.

In the event filter (possibly also in LVL2) inclusive selections can be made of leptonic W and Z decays, making use of invariant mass information in the case of Z decays and missing transverse energy in the case of W decays.

An advantage of inclusive triggers is that one may hope that they cover new, unpredicted physics. This is very important given the fact that LHC will explore a new energy regime.

An example of a more complex trigger that is under investigation for implementation at LVL2 is b-jet tagging based on displaced secondary vertices [3-3], [3-5].Issues to be addressed include the feasibility (beam position stability, alignment, etc.) and a comparison of the merits of making the selection at LVL2 or in the event filter (where more complex algorithms and better calibration and alignment constants may be available).

Studies have been made of lists of criteria for selecting events, which we refer to as trigger menus [3-6]. While these menus will evolve and be adapted once the first physics data are seen by ATLAS, it is important to consider such menus now. Relatively simple trigger menus, such as those presented in the ATLAS technical proposal [3-1] and the trigger performance document [3-3], are sufficient to demonstrate that manageable rates can be achieved while satisfying the goals of the ATLAS physics programme. More detailed menus (see for example [3-7], [3-8]) are needed as input to system design and trigger strategy studies, including modelling [3-9].

Simple menus for use in the LVL1 and LVL2 triggers at high luminosity (approximately 10^{34} $cm^{-2}s^{-1}$) are shown in Tables 3-1 and 3-2. Note that, while the signatures shown in the menu items retain the events required for the ATLAS physics programme as described in [3-3], the list of items is not claimed to be exhaustive. Indeed, it is anticipated that much more extensive menus will be used in practice, including, for example, prescaled triggers with lower thresholds

than the ones shown here. For information on the values in the tables, consult [3-6]. Trigger selection strategies for the event filter are discussed in Section 5.2.3.2.

Table 3-1 Example of a high-luminosity LVL1 trigger menu.

LVL1 trigger requirement	Rate in kHz
MU20	4
MU6 × 2	1
EM30I	22
EM20I × 2	5
J290	0.2
J130 × 3	0.2
J90 × 4	0.2
J100 + XE100	0.5
T60 + XE60	1
MU10 + EM15I	0.4
Other triggers	5
Total	**40**

Table 3-2 Example of a high-luminosity LVL2 trigger menu.

Trigger	Rate in Hz
μ20i	200
μ6 × 2 + m_B	10
μ10 × 2	80
e30i	600
e20i × 2	20
γ60i	400
γ20i × 2	200
j290	120
j130 × 3	80
j90 × 4	80
j100 + xE100	~few 100
τ60 + xE60	~few 100
μ10i + e15i	20
Other triggers	100
Total	**2000**

Trigger menus have also been prepared for lower luminosity running. It is assumed that, following the start-up of LHC, the luminosity will initially be about 10^{33} $cm^{-2}s^{-1}$ which we refer to as 'low' luminosity. During this initial phase, the search for new physics will start, using menus similar to the ones shown in Tables 3-1 and 3-2, but with looser cuts, see Tables 3-3 and 3-4. In particular, it will be possible to operate the trigger with lower p_T thresholds than at high luminosity. In parallel, an extensive programme of B physics will be carried out as discussed below.

Table 3-3 Example of a low-luminosity LVL1 trigger menu.

LVL1 trigger requirement	Rate in kHz
MU6	23
EM20I	11
EM15I × 2	2
J180	0.2
J75 × 3	0.2
J55 × 4	0.2
J50 + XE50	0.4
T20 + XE30	2
Other triggers	5
Total	**44**

Table 3-4 Example of a low-luminosity LVL2 trigger menu.

Trigger	Rate in Hz
μ20	200
e20i	100
e15i × 2	~few Hz
γ40i	100
γ20i × 2	5
j180	100
j75 × 3	80
j55× 4	40
j50 + xE50	250
τ20 + xE30	400
μ6i + e15i	15
B physics	1130
Other triggers	100
Total	**2400**

The ATLAS B-physics programme requires the use of complicated signatures in the LVL2 trigger. Here, the LVL1 trigger will require a muon of $p_T > 6$ GeV, typically coming from the semi-leptonic decay of a B particle or from $B \rightarrow J/\psi X$ followed by $J/\psi \rightarrow \mu^+\mu^-$. The muon candidate can be validated at LVL2, eliminating fake tracks and making a sharper p_T cut; this reduces the rate to ~10 kHz, possibly too high for input to the event filter.

Further selection of B-physics events at LVL2 is based on finding and reconstructing tracks in the inner detector, without the benefit of LVL1 region-of interest (RoI) guidance (see Section 3.2.1), and by selecting, as inclusively as possible, the physics channels of interest. A possible B-physics menu is shown in Table 3-5, which is consistent with the B-physics analysis studies that are being performed in ATLAS, and reduces the rate to a level acceptable for input to the event filter.

Table 3-5 Example of B-physics trigger menu.

Trigger signature	Rate (Hz)	Example channel
$\mu 6$ + additional $\mu 5$	170	Inclusive $J/\Psi \rightarrow \mu^+\mu^-$
$\mu 6 + e0.5 \times 2 + m_{ee}$	310	$b \rightarrow \mu X$, $B \rightarrow J/\Psi\ X \rightarrow ee$
$\mu 6$ + e5	100	$b \rightarrow eX$, $B \rightarrow J/\Psi\ X \rightarrow \mu\mu$ (second μ not required in trigger)
$\mu 6 + h5 \times 2 + m_B$	60	$b \rightarrow \mu X$, $B_d \rightarrow \pi^+\pi^-$
$\mu 6 + h1.5 \times 3 + m_\phi + m_{Ds}$	190	$b \rightarrow \mu X$, $B_s \rightarrow D_s(\phi^0(K^+K^-)\pi)X$
$\mu 6$ +...	300	reserved for additional B channels
Total	**1130**	

The B-physics triggers used at low luminosity (10^{33} $cm^{-2}s^{-1}$) are relatively complicated. However, the low input rate to the complex algorithms (~10 kHz after validation of the LVL1 trigger muon) and the low occupancy in the tracking detectors eases the task. Studies are in progress to understand the ability of ATLAS to continue the B-physics programme at intermediate luminosities above 10^{33} $cm^{-2}s^{-1}$.

In the menus given here, the LVL2 trigger output rate is above 1 kHz at both high and low luminosity. However, these menus are primarily ‘inclusive trigger’ selections and the trigger acceptance rates can be lowered without damaging the physics programme by applying more sophisticated selections (e.g. mass cuts, opening angles, thresholds and scaling).

3.2 System overview

3.2.1 Definition of sub-systems

The event selection, or trigger, function of the ATLAS trigger / DAQ system is organised in three levels as shown in Figure 3-1. At the first level (LVL1 trigger), purpose-built processors act on reduced-granularity data from a subset of the detectors. The second trigger level (LVL2 trigger) uses full-granularity, full-precision data from most of the detectors, but usually examines only regions of the detector identified by the first level trigger as containing interesting information.

At the third trigger level, the event filter, the full event data are available together with the latest available calibration and alignment information to make the final selection of events to be recorded for off-line analysis.

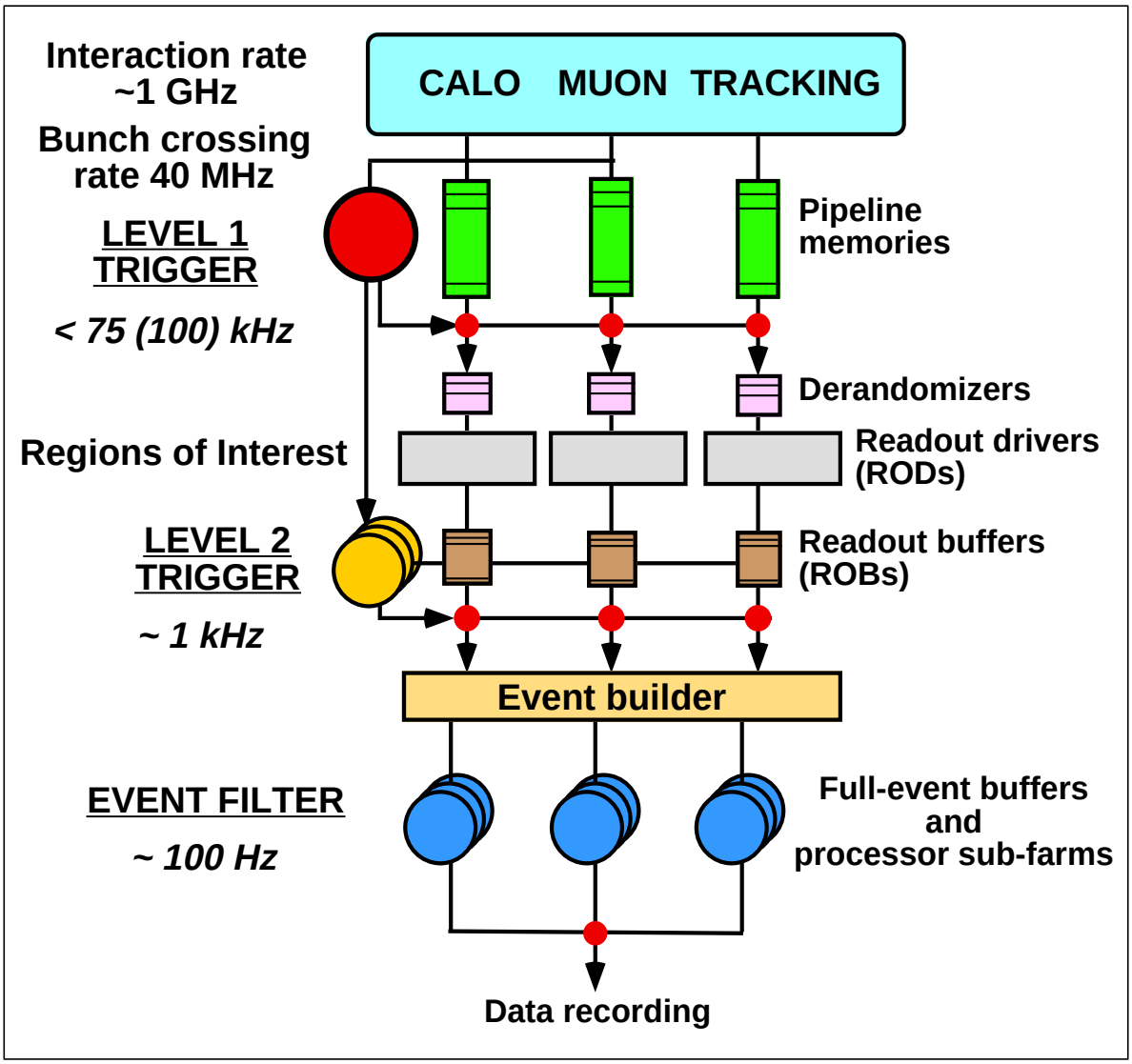


Figure 3-1 Three levels of the ATLAS trigger.

The LVL1 trigger accepts input data at the full LHC bunch-crossing rate of 40 MHz and provides a decision for each bunch crossing. The latency of the LVL1 trigger system (that is the time taken to collect data, form the first level trigger decision and distribute it) is ~2 μs, and all detector data are held in pipeline memories during this period. The maximum accept rate of the LVL1 trigger is set at 75 kHz and is determined by the capabilities of the subdetector readout systems. ATLAS requires that it must be possible to upgrade these systems to operate at 100 kHz with a somewhat higher deadtime (a few percent) and, therefore, the DAQ and LVL2 trigger systems must be designed to accept this higher input rate.

After a LVL1 trigger accept, data are moved off the detector and stored in readout buffers (ROBs) until cleared by a LVL2 reject signal or moved to the event builder.

The LVL2 trigger has to reduce the acceptance rate to ~ 1 kHz. The LVL2 trigger architecture is based on the use of RoIs. The LVL1 trigger is used to identify, for each event, regions of the detector containing interesting features such as high-p_T electromagnetic clusters, jets and muons. The LVL2 trigger then has to access and process only a small fraction of the total detector data, with corresponding advantages in terms of the required processing power and data-movement capacity.

The LVL2 trigger uses full-precision information from the inner tracking detectors, as well as from the calorimeters and muon detectors. Data from the subdetectors are combined to provide better particle identification and higher measurement precision than are possible in the LVL1 trigger. This is necessary if the rate reduction required at the second trigger level is to be achieved. The average decision time for the LVL2 trigger is estimated to be ~10 ms.

After an event is accepted by the LVL2 trigger, all the data for that event are sent to an event-filter processor via the event builder (EB). Complete event reconstruction is possible at this third trigger level and decision times of around one second are expected. The event-filter system must achieve a data-storage rate of the order of 100 Mbyte/s by reducing the event rate and/or the event size. For some triggers the full event data of about 1 Mbyte will need to be recorded, implying a maximum event rate of 100 Hz, while for others a reduced readout is sufficient, allowing a higher event recording rate.

The boundary of operations between the LVL2 trigger and the event filter is overlapped during the R&D phase so that the trade-off of moving event selection between the two can be examined. This boundary must remain flexible during running to optimize the physics performance of ATLAS.

The DAQ system is responsible for the flow of data through the system from the front-end to permanent storage but, to allow flexibility for the subdetectors, a clean interface has been established at the ROBs and it is foreseen to have only one ROB design for all ATLAS. From the front-end to this buffer, and within some general constraints and agreements between the detector and the trigger/DAQ communities [3-10], [3-11], the detector communities are free to pursue their own solutions.

Because of the size and the complexity of the ATLAS detector, a dedicated controls system is needed for its operation and, wherever possible, procedures should be automated. The main task of the detector control system (DCS) is the overall supervision and monitoring of the ATLAS detector and the acquisition, treatment and storage of certain non-physics-event data such as gas temperatures and pressures which are needed to understand the behaviour of the detector and which impact on the physics analyses. The DCS is not concerned with the physics-event dataflow and its quality monitoring.

The classification of data into ‘physics events’ and ‘non-physics-event data’ marks the natural separation between the DAQ/EF system and the DCS. As there are functions which involve both types of data (e.g. calibration), exchange of information and commands between both systems is mandatory. The choice of which system initiates and supervises the process depends on which type of data predominates.

As far as operation goes, the DCS has to present all subdetectors to the operator in a coherent way and to allow him to interact with the infrastructure services and with the LHC accelerator. It has to cover the full range of operation starting from shift operator commands down to expert interaction. In fact, it is mandatory that all operator commands be handled by the DCS and all

error messages be reported via the DCS. Therefore connections to all ATLAS subdetectors are needed. These should be done in a standard way with a well-defined interface point.

Concerning safety, the DCS has to guarantee the safe operation of the detector. The overall safety of the personnel and the material is not within its scope, although the DCS has also to present this dedicated safety system to the operator.

The DCS has to be strongly coupled to the trigger / DAQ system, but it must be operationally independent, as it is needed at times when that system is not running.

3.2.2 Interface to front-end systems

To allow the trigger / DAQ group and the Detector groups to develop their systems in a well-defined way, it was necessary to provide a clean interface between the systems. To this end, the 'Trigger and DAQ interfaces with front-end systems: requirements document' [3-10] was produced. The dataflow chain, see Figure 3-1, from the detector to the ROB is considered to consist of the LVL1 buffer, where data are stored during the LVL1 trigger decision time, a derandomizer, which is used to smooth the flow of data out of the front-end system, a readout driver (ROD), which collates the information from a number of front-end links, a readout link (ROL) connecting the ROD to the ROB and the ROB itself.

The interface is set as the input to the ROB so that only one design of buffer can be used across the whole experiment with one type of ROL and data-transfer protocol. Any detector-specific requirements are constrained to the RODs which must provide an interface to the standard link. The LVL2 trigger and the event builder only access detector data from the ROBs and thus the clean interface is obtained.

Based on ROLs running at around 1 Gbit / s, the total number of links is estimated at around 1700. Table 3-6 gives information on the data produced by each system and the number of links employed. More detailed information is available in the detector interface working group document [3-11].

Table 3-6 Detector channel count and estimated interface requirements.

Detector	Channels count	FE occupancy %	FE raw data bit	FE[a] bandwidth Gbit/s	No. RODs
Pixel	1.4×10^8	0.01	48	50	81
SCT	9 314 304	1.0	32	224	256
TRT	424 576	20	32	204	256
EM calorimeter	173 952	100	32	417	704
Hadron calorimeter	25 714	100	32	62	120
Muon trigger chambers	789 704	0.12	20	1	32
Muon precision chambers	431 392	10	32	103	256
Total	-	-	-	979	1705[b]

a. assuming 75 kHz LVL1 rate
b. additional (~16) RODs will be used for LVL1 output data

With a LVL2 trigger which operates primarily on RoIs, the ease of collection of the data relevant for each event is affected by the distribution of the data over the buffers. Matching the grouping of front-end connections onto ROLs to RoIs or trigger towers (in the case of multilayer systems) can make a significant difference to the number of Readout Buffers which have to be accessed by the LVL2 trigger system and the amount of data which is transferred through the network to the processors. Where possible the grouping should be arranged to minimize the number of buffers participating in an RoI. Table 3-6 indicates the number of RODs in each system and Table 3-7 shows the eta range covered and the channel grouping for each detector.

Table 3-7 Eta coverage and channel grouping

Detector	Eta range	channel grouping $\Delta\eta \times \Delta\phi$
SCT and Pixel	-2.5 to 2.5	0.2×0.2
TRT	-2.5 to 2.5	0.2×0.7 (barrel)
EM calorimeter	-3.2 to 3.2	0.1×0.1
Hadron calorimeter and forward calorimeter	-5.0 to 5.0	0.1×0.1
Muon trigger chambers	-2.2 to 2.2	0.1×0.1
Muon precision chambers	-2.7 to 2.7	0.1×1.0

The basic principle for optimized mapping at the ROB level would be to organize the readout channels of each subdetector in areas (of dimensions appropriate to each subdetector) which are loosely matched to LVL1 trigger towers [3-12]. The ROD is the element where such groupings can me made. For the tracker detectors, the SCT and pixel detector and the TRT, the readout of the cylindrical barrel and endcaps are organized into 32 ϕ-sectors. For the electromagnetic and hadronic calorimeters, an organization into 0.1×0.1 LVL1 trigger towers is foreseen. The muon trigger devices (RPC and TGC), define regions of 0.1×0.1. The muon precision-tracker read-out organization will probably follow the rest of the tracker, in 32 ϕ-sectors, Discussions are in progress with all detector groups to reach agreement on configurations which satisfy the requirements on the detector side (e.g. cost and convenience) and on the LVL2 side of optimum grouping into RoIs.

In addition to grouping front-end channels in a way which aids the operation of the LVL2 trigger system by reducing the number or volume of data transfers inside the LVL2 system, it may be more convenient for the RODs to compute certain types of information, for example energy sums, than have them computed later.

3.2.3 Interface to LVL1

The LVL2 trigger relies on information from LVL1 to guide the processing and the movement of selected data. In the RoI mechanism, the LVL1 trigger identifies for LVL2 geographical regions of the detector which contain candidate objects. These objects can be high-p_T muons, electrons / photons, taus / hadrons or jets. The LVL1 trigger provides information on the position of each of these objects and their transverse energy or momentum range. It also provides LVL2 with information about global event properties — the total scalar E_T and the missing-E_T vector — and it specifies which signatures led to the event being selected by the LVL1 trigger.

Figure 3-2 indicates which data and signals are passed from LVL1 to LVL2. More detailed information is given in Chapter 4 and the LVL1 TDR [3-13].

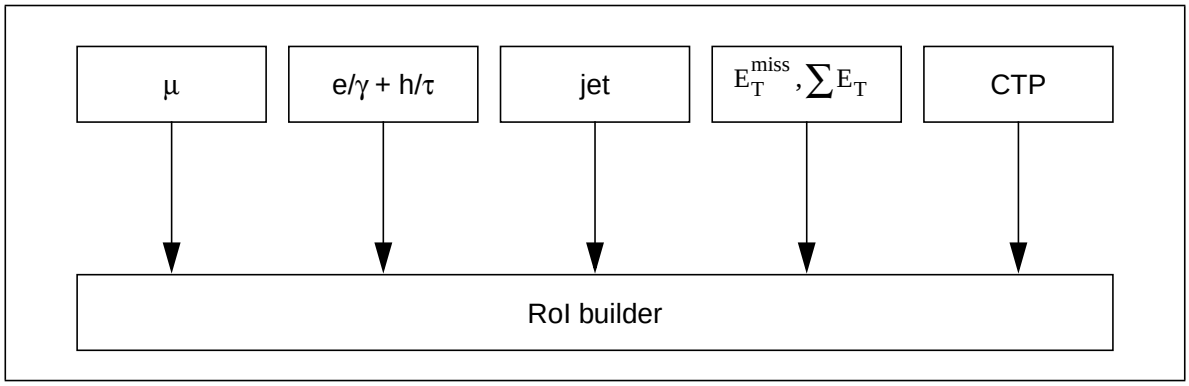


Figure 3-2 Illustration of data that are provided to LVL2 by the LVL1 trigger.

Information is provided from a number of sources in the LVL1 trigger: three in the calorimeter, and one each in the muon and the central trigger processors. In each case, data are accompanied by event and bunch-crossing identifiers. Details of the physical implementation are not finalized but it is anticipated that transfers will be over a number of high-speed links, possibly identical to those used for the ROLs.

The RoI builder of the LVL2 trigger system will use the information provided to distinguish between 'primary' RoIs, those objects which may have contributed directly to the selection of the event at LVL1, and 'secondary' RoIs, objects flagged by LVL1 but which played no direct role in the event selection. A low-p_T jet is a typical example of a secondary RoI.

3.2.4 Dataflow functional view

The dataflow is the primary function of a DAQ system. The way data are to be transferred through the system is an architectural decision and as such needs to be specified as part of the DAQ high-level design. The ATLAS DAQ architecture must support the following dataflow functions:

- At each bunch crossing, signals from each detector electronics channel are stored in detector-specific front-end pipelines for the duration of the LVL1 trigger processing (LVL1 latency).
- For each LVL1 accept, digitized data from all the front-end pipelines are transferred to ROBs, where they are stored for the duration of the LVL2 trigger processing (LVL2 latency).
- RoI data for all the detectors required by the LVL2 trigger algorithms are transferred from the ROBs to the LVL2 trigger system.
- For each LVL2 accept, all the data fragments corresponding to selected events are assembled into full events (event building) and transferred to the event-filter system.
- Each event (physics or calibration) accepted by the event-filter processing is recorded on permanent storage.

In ATLAS the DAQ architecture must satisfy two broad classes of requirements on the dataflow:

- detector-specific front-end electronics requirements, such as radiation tolerance, digitization and front-end readout schemes, geometrical constraints, power consumption and specific signal processing;
- trigger / DAQ system requirements, including the highest possible degree of system uniformity, coherent event format, adequate data organization for efficient trigger data selection and uniform data-transfer protocol.

In order to satisfy these requirements, the following functional elements have been introduced in the data-flow (see Figure 3-1).

- Detector-specific dataflow elements:
  - The front-end pipeline (see above).
  - The derandomizer decouples the instantaneous LVL1 rate of data from the front-end pipeline from the average LVL1 rate readout capability of the next stage electronics.
  - Multiplexing of data from front-end electronics channels may occur in several stages along the dataflow chain, either in dedicated units or as part of other elements. Part of the task of the ROD, see below, is to act as a muliplexer.
  - The ROD provides detector specific signal processing, organizes the data as required by the LVL2 trigger algorithms and prepares the data in the format required by the DAQ.

    The ROD has been introduced in the architecture as an interface between the detector specific dataflow elements and the standard DAQ ones [3-10]. Its role is to convert the detector-specific input into the defined trigger / DAQ standard output. It must also support all detector-specific processing so that the rest of the dataflow elements are uniform throughout the DAQ architecture.
- Trigger / DAQ system elements (standard for all detectors):
  - The ROL moves the event fragments formatted in the ROD to the ROB. The ROL protocol is specified by the DAQ and must be conformed to by all ATLAS systems.
  - The ROB is a multifunctional element of the dataflow: it receives LVL1-accepted event fragments from the ROL; it passes selected data to the LVL2 trigger system; it stores all event fragments during the latency of the LVL2 trigger processing; it transfers all event fragments of LVL2-accepted events to the event-filter system via the event builder.
  - The LVL2 trigger is not on the main dataflow path. The LVL2 system consists of a data-collection stage and a processing stage. The first supports the transfer of selected event fragments from the ROBs to the LVL2 processing stage, where algorithms running on partial event data provide LVL2 event selection or rejection. Both functions may proceed in multiple steps.
  - The event-builder system assembles all the event fragments of the LVL2-accepted events from the ROBs and transfers them as full events to the event filter. As for LVL2, this function may proceed in multiple steps.
  - The event-filter system consists of an event distribution stage and a processing stage. The first receives full events from the event builder and distributes them to the event-filter processing units, where fully reconstructed events are processed for final selection before recording.

A variety of architecture designs capable of providing the data-flow functionality described above can be envisaged. Depending on the architectural solutions being considered for the dataflow and the degree of integration foreseen, the functional elements described above may be mapped onto separate or compound system elements.

3.3 Factorization of system studies (based on functions)

During the early stages of development of the ATLAS trigger and DAQ system, it became apparent that a multistage trigger was needed to provide the necessary event-rate reduction between primary interactions and writing data to permanent storage. While it was clear that the LVL1 trigger would require special, dedicated processing and that the final stage would use general-purpose processor farms, the technology required for the intermediate trigger stage was debatable. In particular, how much could be accomplished by general-purpose processors and how much required dedicated or special hardware.

Therefore, we decided to put in place programmes to investigate the requirements for each part of the system and the implications of these requirements for architectures and technologies. We have developed such work programmes and these have been or are being actively pursued.

For the LVL2 trigger, the programme has looked at a range of options to solve the problem of a decision frequency of up to 100 kHz combined with a rate reduction of order one hundred, see Chapter 4. Further work is required on the technology options, and the software and system management problems are only now being studied in depth.

For the DAQ and event filter the DAQ/EF-1 programme has been developed to study the functional requirements of this combined system where data must be gathered from around 1700 buffers and assembled into complete events for reconstruction and analysis, see Chapter 5.

The DCS, see Chapter 6, must operate in close conjunction with the DAQ but is operationally independent. Prototype work is going on to address a number of pressing topics: the definition of the interface point to the subdetectors; the development of a common approach for the front-end controls readout; the definition and development of interfaces to external systems; and, the evaluation of commercial solutions for the supervisory controls system.

A standalone control system is being set up in order both to validate the components and to help the subdetector groups to carry out practical work in test beams and in the laboratory.

3.4 References

3-1 ATLAS Technical Proposal, CERN/LHCC/94–43, LHCC/P2 (1994).

3-2 ATLAS physics technical design report (in preparation).

3-3 ATLAS trigger performance status report, CERN/LHCC 98–15 (1998).

3-4 ATLAS level-2 trigger user requirements document, ATLAS Internal Note, DAQ–NO–079 (1997).

3-5 ATLAS pixel detector technical design report, CERN/LHCC 98-13, ATLAS TDR 11 (1998).

3-6 ATLAS trigger menus, ATLAS Internal Note, PHYS–NO–124 (1998).

3-7 J. Bystricky et al., ATLAS trigger menus at luminosity 10^{33} $cm^{-2}s^{-1}$, ATLAS Internal Note, DAQ–NO–054 (1996).

3-8 A. Amadon et al., ATLAS trigger menus at luminosity 10^{33} $cm^{-2}s^{-1}$, ATLAS internal note, DAQ–NO–110 (1998).

3-9 S. George, J.R. Hubbard and J.C. Vermeulen, Input parameters for modelling the ATLAS second level trigger, ATLAS Internal Note, DAQ–NO–070 (1997).

3-10 Trigger and DAQ interfaces with front-end system: requirements document version 2, ATLAS Internal Note, DAQ–NO–103 (1998).

3-11 Detector interface working group, summary document (1997). http://atddoc.cern.ch/Atlas/DetfeIf/document/Detinfo.ps

3-12 R.K. Bock and P. Le Du, Detector readout specification for the LVL2 demonstrator programme, ATLAS Internal Note, DAQ–NO–53 (1996).

3-13 ATLAS LVL1 trigger technical design report, LHCC 98–14, ATLAS TDR 12 (1998).

4 Level-2 trigger

4.1 Introduction

4.1.1 Requirements for the level-2 trigger

This section introduces the requirements which the ATLAS second level trigger system has to fulfil, seen from the physics and from outside systems.

The level-2 (LVL2) trigger acts on events at the rate retained by the level-1 (LVL1) trigger, which is estimated at 75 kHz (safety factor included), and has to be scalable to 100 kHz. Its role is to reduce the trigger rate to a level which can be sustained by the event building system, in the range of a few kilohertz. In the current design, full-granularity, full-precision data would be available to LVL2 from selected regions in the event indicated by LVL1. Where possible, general-purpose commercial components will be used. The third level, called the Event Filter system, operates after event building and will reduce the data rate to match permanent storage capabilities. The division of tasks between LVL2 and the Event Filter system is to be determined as part of the overall trigger system optimization (e.g. cost, performance).

The normal mode of operation for the LVL2 trigger system for high-p_T triggers is based on the use of regions of interest (RoIs), defined by the LVL1 trigger. A region of interest is signalled by LVL1 for candidate high-p_T muons, electrons / photons, taus / hadrons, and jets. The information on the RoIs, in particular their position in η and ϕ and the LVL1 selection criteria, are made available to LVL2; based on this information, LVL2 decides which data from each subdetector are associated with each of the RoIs, and only these data are transferred to the LVL2 processors. This selection of data for the LVL2 trigger reduces the load on the LVL2 system and allows for substantial simplification of LVL2 algorithms.

At low luminosity, the LVL2 trigger system will be capable of processing both high-p_T and B-physics trigger candidates simultaneously. For B-physics triggers, only the LVL1 trigger muon is likely to have an RoI defined by the LVL1 trigger system. The proposed mode of operation of the LVL2 trigger is a sequence of steps. The first step is to use data from the trigger muon RoI to confirm the muon (about half of the LVL1 muon triggers are expected to be rejected at LVL2, mostly from applying a sharper p_T cut). Once the trigger muon is confirmed, data from other subdetectors are required to test the LVL2 B-physics hypotheses, without any RoI guidance being available from LVL1. A full scan in one of the tracking systems (assumed to be the TRT) is used to identify tracks, and these then define the RoIs for selection of data from the other subdetectors for further analysis.

Event processing at LVL2 can be decomposed into a number of broad steps: feature extraction, object building, and trigger type selection. In feature extraction, the data for one RoI for one detector are gathered together and processed to give physics-like quantities (e.g. for the calorimeter, this process would take cell information and produce cluster parameters; for a tracker, the basic hit information would be converted to track or track-segment parameters). Object building takes the features for one RoI from all relevant detectors and returns the particle parameters and possibly the particle type. If the data are consistent with more than one particle type, subsequent processing makes tests with each of the possible object types. In the trigger-type selection phase, all the objects found in the event are combined and compared with the topologies (parti-

cle types, transverse momenta, inclusive / missing mass, etc.) for a menu of physics selections. Flags are set for each menu item for which a match is found.

4.1.1.1 Event selection

The task of the LVL2 system is the selection of events according to a trigger menu. Items in the menu express physics selections in terms of seven basic trigger objects: muons, electrons, photons, charged hadrons, taus, jets, and missing-E_T (the discrimination between electrons and photons is possible due to the tracking information available in LVL2). Each item in the trigger menu defines one set of trigger conditions, including the required number of objects of each type and the requirements on properties of these objects (e.g. p_T thresholds, isolation criteria, etc.). Further trigger conditions applying to several trigger objects can also be defined, such as cuts on effective mass. For each trigger object, there is usually more than one transverse energy or momentum threshold. The lower thresholds usually correspond to the trigger thresholds for an object participating in multi-object triggers. The higher thresholds usually correspond to non-isolated single-object triggers. Confirmation of individual trigger objects can be an important step in the selection; however, nonconfirmation of a given candidate trigger object does not necessarily permit the rejection of an event.

The trigger selection criteria will be considerably looser than those used for the final data analysis and they must be kept sufficiently flexible to allow the efficient selection of new and interesting physics phenomena. The LVL2 trigger will allow for prescaling of trigger items, and the forced acceptance of some events. Prescaling allows the number of events from well-known high-cross-section processes to be reduced while still accepting sufficient to monitor performance.

The geometrical limits of acceptance of trigger objects are given by the detector: the η limits of the LVL2 muon triggers are constrained by the muon trigger chambers ($|\eta| < 2.4$), the tracking and electromagnetic calorimeter triggers are limited to $|\eta| < 2.5$ by the inner detector and calorimeter geometry, jets will include information up to $|\eta| < 3.2$.

4.1.1.2 Operational modes and partitioning

The LVL2 system will support several operational modes. In normal mode, the system is used for event selection and only accepted events are passed to the event filter. In tagging mode, the system processes all events, but does not reject any, and all events are passed to the event filter. For both these modes, the LVL2 system operates under the control of the back-end DAQ. The test mode is limited to test procedures using either test or event data. Tests may involve several parts of the trigger and the data acquisition chain or can be internal, standalone, tests of the LVL2 system. For tests not utilizing the back-end DAQ, local control and local DAQ systems will be used.

Subdetector systems must be able to operate independently during development, debugging, calibration and testing and, therefore, it must be possible to partition the LVL2 system. To operate independently, each of the partitions has its own local control and local DAQ systems.

4.1.2 Functional description of the level-2 components

The LVL2 system can be described in terms of four main functional blocks: the ROB complex, the supervisor, the networks and the processing farms. This section does not consider any particular architecture or implementation, but tries to identify and describe the different characteristics of the logical functions of the four blocks. A more detailed analysis and the way these functions are grouped and linked together is described later in Section 4.4.2.

4.1.2.1 The ROB complex

For events accepted by LVL1 the data from all the front-end electronics channels are transferred to the readout buffers (ROBs). The ROB is a standard functional component common to both LVL2 and DAQ/EF dataflow. Its main function is to store the data event-fragments during the LVL2 latency.

For the communication with the LVL2 system there is a control interface and a data interface. The control interface receives requests for data and LVL2 decisions from the LVL2 system. The data interface sends the requested data from the ROBs to the LVL2 system. The data interface may contain options for preprocessing data within or local to the ROBs and options for combining data from several ROBs (via a bus or network) prior to transmission.

4.1.2.2 The supervisor and RoI builder

The supervisor and RoI builder has three main functions. The first is to receive at the LVL1 accept rate (up to 100 kHz) the RoI information fragments from LVL1 and build the RoI record for each event. The RoI record contains a list of RoIs with their types and η, ϕ positions. The second function is to assign LVL2 processors as needed and to route the RoI record to the appropriate part of the LVL2 system for the processors to obtain the RoI event data; this function may be coupled to monitoring of processor usage and application of event time-outs. The third function is to receive LVL2 decisions and broadcast them to the ROB complex.

4.1.2.3 The networks

There are two logical networks: the data collection network that transmits the RoI data to the various processing elements; the control network that carries LVL1 accept, LVL1 RoI request and LVL2 accept/reject messages. Depending on the architecture and implementation choices, each of these networks could be implemented as multiple physical networks or they could be combined into a single physical network for all of the traffic.

4.1.2.4 The processing farms

There are three main tasks to be done using processor elements corresponding to the main types of algorithms used in the LVL2 selection steps.

- Feature extraction, which processes data from a single RoI and single detector to extract features such as calorimeter cluster transverse energy and track parameters.
- Object building, which combines the features for a single RoI from the different detectors and returns the particle properties and possibly type.

- Global processing, which combines the objects and compares them to trigger menu items to produce the LVL2 accept or reject decision.

Processors may be grouped in clusters or 'subfarms'. The 'subfarms' would connect to the network via a 'Switch to Farm Interface' (L2-SFI).

4.1.3 Issues and constraints

The fundamental technical problem for the implementation of the LVL2 is how to make the data available to the processors executing the trigger algorithms and how to control the use of these processors. The full size of each event is ~1 Mbyte, they are received at the LVL1 accept rate (up to 100 kHz), the data for each event is spread over the ~1700 ROBs of the system, the processing power required for the LVL2 algorithms is of the order of 10^6 MIPS, the decision for each event is required in an average time of ~10 ms.

Various strategies have been proposed which singly or in combination make this problem more tractable:

- To transmit to the LVL2 system only the data contained within RoIs found by the LVL1 system. This reduces the data to a few percent of the total for each event, but introduces the requirement to distribute data requests to the appropriate ROBs. Some further reduction of the data volumes to be transmitted can be obtained, for some subdetectors, by preprocessing data in or close to the ROBs, for example by reformatting the data to more compact forms.
- To use many processors working in parallel. This can be done at various levels: processing of different events; algorithm processing within an event, e.g. the feature extraction for different RoIs and/or different subdetectors; preprocessing of data from different ROBs or groups of ROBs; processing associated with data gathering. These various forms of parallel processing affect the rate and latency for Level-2 decisions, but they also impact the total number of processors and the problems of processor allocation and control.
- To use very high speed data driven processors to run feature extraction algorithms. This reduces the number of processors and eases the control problem, but introduces new problems for data collection.
- To construct the network connecting the ROBs to the processors from a number of smaller networks. This is because in general it is easier and cheaper to build several small networks than a single large network, particularly if the network is to remain non-blocking (i.e. that the use of any pair of ports at maximum bandwidth, does not reduce the bandwidth available between any other pair of ports).
- To reduce the number of events requiring processing of some of the algorithms by applying sequential selection. For example, in the first step of the processing to use only trigger RoIs in the calorimeter and muon systems, and then proceed to other processing of the event only if the LVL1 trigger is confirmed. As was already noted in the ATLAS Technical Proposal [4-1] this can lead to significant savings in data bandwidth and the total processing required. Extending this to several steps leads to a more flexible trigger, although it affects the degree to which parallel processing can be applied within an event. As the number of steps increases so does the number of data request phases (but not the total number of data requests). This has implications for the mechanism used to transmit the requests to the ROBs and, to avoid undue idle time waiting for data, requires overlapping

the processing of different events within each processor, leading to increases in context switching time.

The first two of these strategies are assumed for all of the ATLAS Level-2 trigger studies, the degree of parallel processing and the other strategies, however, continue to be matters for study. The importance of each strategy and the optimal way to combine them is evolving as various constraints change.

On the technical side, the continual improvement of the commercially available, high-technology components required for the system promises major benefits and simpler solutions.

- The constant evolution of RISC / CISC processors continues to follow Moore's Law with computing power approximately doubling every 18 months. The 300–500 MIPS of today's standard commercial processors would extrapolate to 2–4 GIPS at the start of LHC operation and perhaps to 10 GIPS for high-luminosity running, indicating that the pure computing power for feature extraction and global processing will not be a major issue (although it will require corresponding increases in input / output (I / O) performance for the extra power to be fully utilized). Similarly the size and power of FPGA devices continues to grow, and already they have demonstrated adequate power for some of our requirements.
- The memory chips used for the data storage in the ROBs are becoming larger and cheaper, with the consequence that a LVL2 latency of ~10 ms is no longer expected to be a major issue.
- The aggregate bandwidth required for the total LVL2 network traffic is estimated to be of the order of several Gbytes / s (see Table 4-3). This high bandwidth cannot be handled by the traditional bus-based data-acquisition systems. However, high-end commercial network components promise performance levels of the order required and, if current trends continue, at an affordable price. Building such a large high-performance network will still be a complex task and to attain the bandwidth and latency requirements will place extreme demands on the components and the configuration. To gain the necessary knowledge and experience it is now appropriate to perform detailed evaluations and comparisons of these technologies.

The complete LVL2 system will be very complex and will require long-term maintenance. In order to reduce the overall development phase and facilitate the maintenance, it is strongly desirable to use commercially available components in a modular configuration, wherever possible. In addition, compliance with widely adopted standards will assist the interoperability (software and hardware) of equipment from different vendors.

Data extraction from ROBs and data movement to the processors is a major issue. From the hardware side, the interfaces and I / O devices between the different high-speed elements (links and networks) are in general the critical areas to be optimized. From the software side, the performance of device drivers, and various overheads such as context switches and interrupt handling will need to be optimized.

The extraction and collection of the RoI data from the ROBs imposes critical constraints on the ROB design and other components. The number of ROBs which contribute to each RoI and the rate at which each ROB is accessed are strongly coupled to the mapping of the detector elements into the ROBs. Thus details of the mapping of each subdetector into the ROBs could have a major impact on the LVL2 system performance.

The format of event-fragment data may affect the data volume to be transferred within the LVL2 system and / or the performance of the feature extraction algorithms. In some cases reformatting of the data will be desirable before transfer across the network. Optimization of the format coming from the detectors could thus make important savings in bandwidth and minimize the data manipulation in the ROBs or LVL2 system. In some cases optimization could be achieved by preprocessing at the ROD level, but clearly, this would influence the ROD design. The balance between extra complexity of the RODs and benefits from the simplification of ROB data extraction and formatting is only partially understood at this stage and requires further studies. It is therefore important that the formats are chosen to minimize the overhead of extracting, transferring and any necessary reformatting of the restricted data required by LVL2. This is a strong requirement from the LVL2 and a real issue if not satisfied.

Finally the LVL2 system must be flexible, allowing different scenarios of event selection and changes of the boundary with the event filter as the physics interest evolves and the luminosity of the LHC machine increases.

4.1.4 Project phases

The first phase, the demonstrator programme, started in 1996 and finished in February 1998. This R&D programme studied components for a range of representative architectures and different technologies. The components included: data-driven processors based on FPGAs (ENABLE++); an RoI collector to assemble data fragments for the data driven processors; an RoI distributor for ‘push’ architectures; an RSI (ROB switch interface) to fetch data fragments for ‘pull’ architectures; emulated ROBs; a system supervisor; farms of processors; various network technologies (DS-link, SCI, ATM). These components could be combined to form various architectures, but may be understood most simply in terms of the following simple cases:

a. Architecture using a few data-driven FPGA processors (ENABLE++) to extract features in each detector using simple algorithms and minimum latency. A second step combined the features for each event in one of a farm of general-purpose processors.

b. Architecture with local and global farms using a number of parallel farms to extract the features in each detector and followed by a second step where features are combined in a global farm.

c. Architecture performing all of the algorithm processing for an event sequentially in a single processor in a processor farm, receiving the data via a single network.

In addition to these architecture / technology studies some related activities (modelling, emulation, communication benchmarks[1]) completed this R&D programme. The results of this programme were used to select the main options to be retained for the next step.

The second phase, the pilot project, as described in this chapter will study the main critical issues for LVL2. The aims are to refine architectural ideas and trigger selection strategies using vertical slice testbeds; to optimize critical components; and to study system integration issues. In this way to prepare and collect the information required in preparation for the technical proposal due in December 1999.

1. See Glossary for a brief description of terms

4.2 Input parameters

This section summarizes the present knowledge of the various input parameters for system design coming from the physics simulation and specification of the detector electronics. Some numbers are subject to further optimization, but this set of tables is the basic ingredient for the next step of the LVL2 system modelling.

4.2.1 Information from LVL1

Examples of LVL1 menus and input rates for the LVL2 selection process for high and low luminosity were given in Chapter 3 Tables 3-1 and 3-3. An average LVL1 trigger rate of ~40 kHz is foreseen, compatible with the 75–100 kHz design limit and allowing for some safety margin.

4.2.1.1 Information provided by LVL1 to LVL2

The information transmitted from LVL1 to LVL2 is summarised below:

High-p_T muon RoIs:

- p_T range (muons classified in six p_T ranges)
- RoI location in detector with a typical η–ϕ resolution of 0.1×0.1
- Flags (indicating possible chamber overlaps, overflows in resolving nearby muon candidates, first- or second-highest p_T muon within a sector).

High-p_T electron / photon RoIs:

- p_T range (isolated electromagnetic clusters classified in eight p_T ranges),
- RoI location in detector with a typical η–ϕ resolution of 0.2×0.2.

High-p_T tau / hadron RoIs:

- p_T range (isolated hadronic clusters classified in eight p_T ranges),
- RoI location in detector with a typical η–ϕ resolution of 0.2×0.2.

High-p_T jet RoIs:

- p_T range (jets classified in eight p_T ranges),
- RoI location in detector with a typical η–ϕ resolution of 0.4×0.4.

Scalar and missing E_T information:

- Scalar E_T value,
- E_x and E_y values (missing E_T vector).

Trigger type information:

- Multiplicities per p_T threshold for each kind of object,
- Results of threshold comparisons of scalar and missing E_T,
- Other trigger inputs (e.g. special calibration triggers),

- Information on LVL1 menu items for which selection criteria were met (before and after prescaling applied).

In each case some header information will be provided giving the event identifier and bunch-crossing identifier numbers, as well as a few bytes for error flags and for framing. For information on the implementation of the LVL1 trigger, including its interfaces to LVL2, the reader is referred to [4-2].

Details of the physical implementation of the LVL1-to-LVL2 interface are not yet fully defined, although some more information can be found in [4-2]. However, it is anticipated that data will be sent from LVL1 to the LVL2 trigger over a number of high-speed data links associated with different object and information types.

The RODs associated with the cluster and jet/energy-sum processors of the LVL1 calorimeter trigger, and the MUCTPI (see Chapter 13 of [4-2]) of the LVL1 muon trigger, as well as the LVL1 CTP would be equipped with links to the RoI builder. These links would be separate to the readout links that go to the ROBs, but would use the same (or a similar) link technology.

The bandwidth for the various links has been evaluated using estimated average values for the number of RoIs of each type per event. The bandwidth values are shown in Table 4-1 assuming 100 kHz event rate from LVL1. The header/trailer size is conservatively assumed to be 16 bytes. For the RoI information, it is conservatively assumed that there is one 32-bit word of information for the event, followed by two 32-bit words of information for each RoI. It is anticipated that the data volume in the final implementation will be less than this.

Table 4-1 Data rate per link for LVL1-to-LVL2 interface

Information type	Header/trailer size (bytes)	Fixed information (bytes)	No. of ROIs per event	Size of RoI (bytes)	Data Rate (Mbyte/s)
Muon RoI	16	4	1	8	2.8
Electron/photon RoI	16	4	1	4	2.8
Tau/hadron RoI	16	4	1	4	2.8
Jet RoI	16	4	3	4	4.4
$\sum E_T$, E_T^{miss}	16	6			2.2
LVL1 trigger type	16	40			5.6

Although there remain uncertainties both on the number of RoIs per event and on the amount of data to be transferred per RoI, it can be seen that the data rate per link is such that it can easily be handled by high-speed serial links. We expect to use the same kind of link as is used for the ROD-to-ROB connection (so-called readout link). Prototyping of the LVL1-to-LVL2 interface will be carried out in the next phase of the project using the ATLAS standard S-Link [4-3] system.

Each RoI identified by the LVL1 trigger is characterized by its type (muon, electron/photon, jet, etc.), its spatial coordinates, and information such as energy range and isolation threshold. Two categories of RoIs can be distinguished. The trigger, or primary, RoIs are those which probably contributed to the LVL1 decision. The secondary RoIs, generally with considerably lower thresholds, give additional information on the global topology of the event. Figure 4-1 shows an

example of the distributions of the total number of RoIs and the number of trigger RoIs for the calorimeter triggers at high luminosity. It should be noted that the distribution for the total number of RoIs is strongly dependent on the thresholds used for the secondary RoIs in the LVL1 menu.

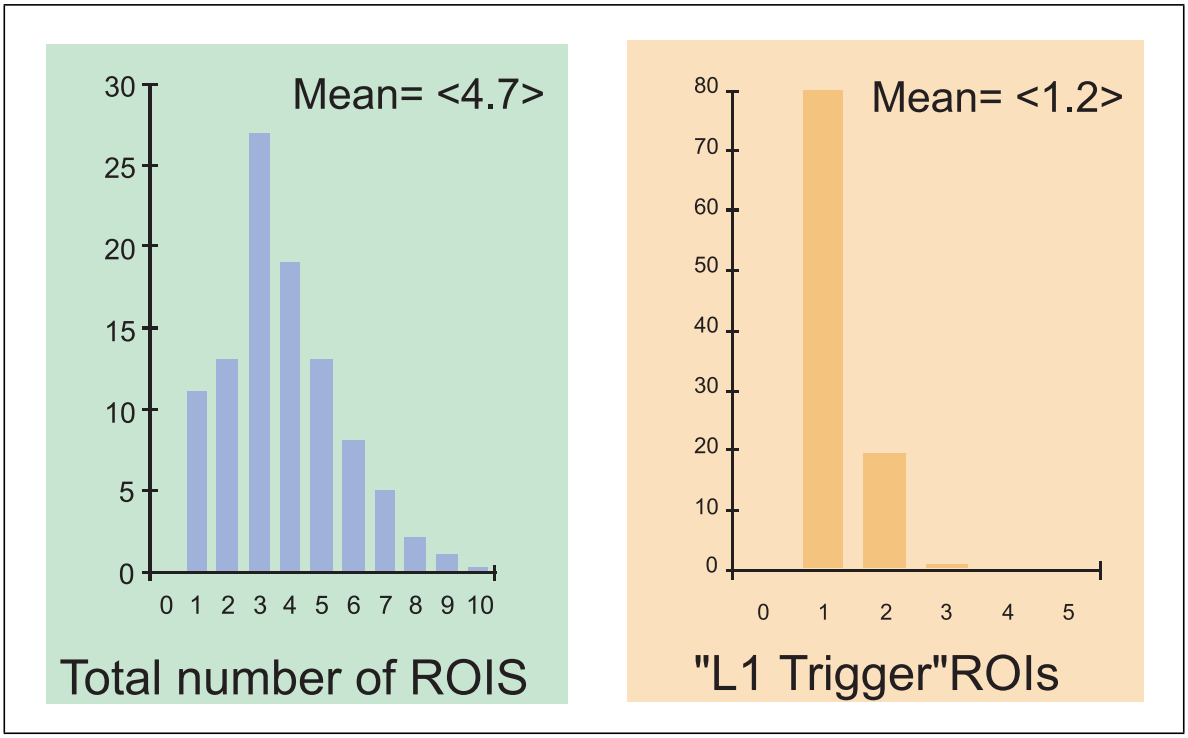


Figure 4-1 Number of RoIs per event for calorimeter triggers at high luminosity.

Whilst the total number of RoIs is around five, in 80% of events there is only one trigger RoI. A significant event rejection can be achieved by processing only LVL1 trigger RoIs as most of them are not confirmed with the higher granularity and improved precision of the LVL2 algorithms. The calorimeter and / or muon detector data alone can be used to sharpen the energy and momentum cuts and to refine the region of interest. In addition, confirmed muon and electromagnetic cluster RoIs can be matched to tracks in the inner detector to lower the event rate even further. As an example, Table 4-2 shows the expected background rejection factors due to confirmation of the electromagnetic RoI at low (10^{33} $cm^{-2}s^{-1}$) and high (10^{34} $cm^{-2}s^{-1}$) luminosity. Each rejection factor corresponds to the rate reduction obtained for 90% efficiency for isolated electrons at the nominal threshold [4-4].

Table 4-2 Background rejection factor at 90% efficiency for electromagnetic RoI trigger.

Luminosity	Low	High
Nominal threshold (GeV)	20	40
Calorimeter algorithm rejection factor	3	10
Calorimeter and Inner Detector algorithms rejection factor	25	60

4.2.2 Information from the ROBs

The number of ROBs for each subsystem and the type of mapping of the detector into the ROBs are shown in Table 4-3, together with the typical RoI data size and an estimated bandwidth for data to be transferred to LVL2 from each subsystem.

Table 4-3 ROD – ROB count, type of mapping, RoI size and LVL2 bandwidths by subsystem.

Subsystem	No. RODs	FE-ROD mapping	Average RoI size (kbytes)	LVL2[a] bandwidth (Gbit/s)
SCT-Pixel	337	Sector	1.5	0.12 – 4.2
TRT	256	Sector	0.6	0.05 – 1.8 (13)[b]
EM calorimeter	704	LVL1 tower	1.2	0.77 – 3.6
Hadron calorimeter	120	LVL1 tower	0.8	0.13 – 2.0
Muon trigger chambers	32	RoI tower	0.5	0.15 – 2.4
Muon precision chambers	256	RoI tower	1.9	
LVL1 Trigger	16			
Total	1721			1.3 – 14 (27)

a. assuming 75 kHz LVL1 rate
b.) including TRT Full Scan

It should be noted that the bandwidth ranges quoted correspond to one trigger selection process. Considerable differences in bandwidths (some outside the ranges given here) are to be expected with changes in luminosity, selection process and trigger menu.

4.3 Selection process and strategies

4.3.1 Introduction

The LVL2 trigger uses full granularity data from the muon, calorimeter and inner tracking detectors to create trigger objects (viz. particle or jet candidates certified by a sequence of algorithms). The LVL2 trigger code is optimized for fast execution using a minimum amount of information from the regions of interest (RoIs) defined by the LVL1 trigger. Several physics triggers will be implemented, such as inclusive triggers, triggers on global quantities and combination of signatures, B-physics triggers. Triggers for luminosity measurements and beam monitoring, calibration and alignment will also be implemented but are not discussed here.

We assume that LVL2 can reduce the trigger rate by a factor of about 100. This is achieved by sharpening the thresholds for inclusive triggers and performing more complex processing than is possible at LVL1. In particular, LVL2 is the first place where data from the tracking detectors are available. The trigger selections will change with the LHC luminosity, more importance being given to the low-p_T channels in the initial, lower-luminosity running.

The RoI data will be processed by feature extraction algorithms to extract quantities which are used in the global decision. The selected methods will be simpler and less iterative than in offline code. The algorithms will not depend on the ultimate resolution of the detector since the full set of calibration data may not be available at the trigger level.

The basis of the algorithms being studied for each type of object are introduced here and explained more fully in later sections.

Muons identified at LVL1 will be confirmed at LVL2 in the muon spectrometer, using both trigger and precision chamber data (thus rejecting background and sharpening the threshold). The inner detector could be used for additional confirmation, and isolation of the muon candidate would be checked with calorimeter data.

For electrons / photons identified at LVL1, more refined analysis at LVL2, using full-precision, full-granularity calorimeter data, will be used to reject background from jets and sharpen the threshold. Further refinement of candidate electrons can be achieved by matching the calorimeter data with the inner-detector track parameters and use of the transition radiation signature.

Taus (decaying to hadrons) and isolated hadrons identified at LVL1 will be subject to refined analysis at LVL2 using full-precision, full-granularity calorimeter data to reject background from jets and sharpen the threshold. We can also require a matching inner-detector track (at least for a one-prong topology).

Thresholds for jets identified at LVL1 will be sharpened using more refined analysis at LVL2 using full-precision calorimeter data. This, however, does not give a large background rejection factor as jets are the dominant high-p_T process. Inner detector information could be used to identify b-jets at LVL2 by using an impact parameter algorithm.

Full recalculation at LVL2 for missing-E_T signatures from LVL1 cannot be done at a high rate, but corrections for high-p_T muons, mismeasured jets, etc. could be applied. It would however be possible to do full recalculation for a small fraction of events.

Section 4.3.2 to Section 4.3.4 give brief details of LVL2 algorithms used for feature extraction in the different detectors. The trigger code has been optimized on different hardware and software platforms and is available, together with selected data samples, for testing various trigger strategies on testbeds. Benchmarking results (algorithm execution times) are reported in Section 4.3.5. In Section 4.3.6, we discuss how features can be combined between detectors to identify objects such as electrons or isolated muons. This section also addresses the important topic of B-physics triggers. Trigger menus, on which the global decision will be taken, are discussed in Section 4.3.7 and an example of ‘sequential’ selection is described in Section 4.3.8.

More details of the algorithms, together with performance in terms of rates and efficiencies, can be found in [4-5].

4.3.2 Muon feature extraction

The muon algorithm identifies muon tracks in the RoIs selected by the LVL1 muon trigger and performs an accurate calculation of their space position and transverse momentum. Full granularity and resolution of data is used to allow the sharpening of the p_T threshold which reduces the LVL1 muon rate by a factor of around two.

Within each RoI selected by the LVL1 trigger ($\Delta\eta \times \Delta\phi \sim 0.1 \times 0.1$), the algorithm performs two steps: pattern recognition of the muon tracks in the precision chambers and transverse momentum fit. The differences in layout, magnetic field, and background conditions of the barrel and end-caps of the muon spectrometer lead to slightly different implementations of the algorithms in the two regions.

Because of the high background rate, the pattern recognition in the MDTs is initiated by the information from the fast trigger chambers used in the LVL1 trigger RPCs in the barrel and TGCs in the end-caps. Input data to the algorithm are RPC strip coordinates (r-ϕ and r-z views), TGC strips and wires, LVL1 threshold and RoI information, MDT hits (tube number and drift times) for the chambers indicated by the RoI. Output data are track parameters, that is the sign, the muon p_T, η and ϕ in the standalone system. Extrapolation to the calorimeter and to the inner tracker can be provided as further results.

The method used for the pattern recognition is simple and fast as it allows the definition of the muon roads using a simple geometrical model, which ignores the magnetic field map. It takes into account only the centres of the hit tubes, eliminating the ambiguity of the drift time.

In the barrel, for sufficiently high p_T, the track will cross three RPC stations; for lower p_T only two. The averages of the strip coordinates for the different trigger planes give three (two) points which are used to define the muon trajectory.

For the high-p_T trigger, the trajectory is parameterized as a circle through three points in the bending plane (r, z) and for low p_T as a circle through two points, with the condition that its tangent at the *r* value corresponding to the limit of the no-field region goes through the nominal vertex position. Precalculated road widths allow the definition of a region in the precision chambers where hits will be selected.

From road coordinates, the selection of MDT tubes within the road is performed using a ‘contiguity algorithm’ to filter out noise hits: for the selected set of tubes, the information of the drift times is used to produce track points, in most cases one per tube plane. Presently, the method used to determine the track points [4-6] is an offline-like procedure where the time to space (r-t) relation is applied to the selected hits. However, faster techniques are being studied and compared to the optimal one in order to understand the limits in resolution and the gain in processing time.

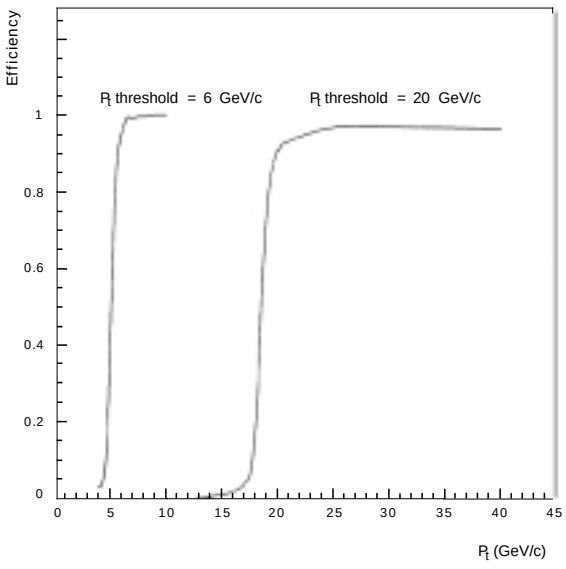


Figure 4-2 LVL2 trigger algorithm efficiency versus muon p_T (barrel region).

In the barrel, by averaging the hits per chamber plane obtained from the previous step, three ‘superhits’ are computed and used to determine the (η,ϕ) position of the muon track and its effective sagitta. The sagitta is related to p_T by a function which has been evaluated for each (η,ϕ) position using detailed simulation. Sagitta maps are stored in look-up tables (LUTs) and used for fast processing.

This method yields a p_T resolution of 2–3% for $p_T < 100$ GeV/c (~ 7% for 6 GeV/c muons). The LVL2 trigger efficiency versus the muon p_T is shown in Figure 4-2 for the barrel region.

In the endcap regions the pattern recognition is initiated by the TGC information and a track segment is reconstructed in the outer and middle MDT stations using precalculated roads. Then a straight line is fitted through them. The distance (d) between this line and the interaction point gives a measurement of the p_T using a look-up table (LUT).

If there is not enough resolution for the calculation of 'd', TGCs and MDTs in the inner MDT station are used to compute a third point to determine p_T (always using LUTs). If the muon doesn't reach the outer station, the track segment reconstructed in the middle station is used to 'drive' the hit search in the TGCs and MDTs of the transition and inner stations.

Details of the muon algorithms and their performances are reported in [4-5].

4.3.3 Calorimeter feature extraction

The calorimeter feature extraction algorithms search for clusters in the data from the electromagnetic and hadron calorimeters, restricting the searches to RoIs. For each cluster found, the center of gravity in η and ϕ, the energy and other quantities describing the transverse and longitudinal properties of the shower are calculated, including isolation parameters. Shape quantities in the first compartment of the electromagnetic calorimeter are also computed, e.g. to distinguish e/γ from π^0 showers.

In the following, we summarize the feature extraction algorithms for electromagnetic (e.m.) clusters, tau/hadron signatures, and jets, including the global triggers for missing transverse energy (E_T miss) and total sum of transverse energy, $\sum E_T$.

For its trigger objects, the LVL1 trigger can use up to eight thresholds, where the 'thresholds' for e.m. cluster and τ triggers consist of a triplet of thresholds: E_T of the cluster, E_T for e.m. isolation and E_T for hadronic isolation. Details of the LVL1 algorithms are described in [4-2]. For each of the accepted LVL1 triggers, the threshold and RoI information is transmitted to LVL2. The RoI position is provided with a resolution comparable to the size of the LVL1 trigger cells, e.g. for the e.m. cluster the precision is $d\eta \times d\phi = 0.1 \times 0.1$. The LVL1 algorithms work on matrices in $\eta \times \phi$, which store the energies per trigger tower and for both e.m. and hadronic calorimeter. These matrices are also available to LVL2 for cross checks and verification.

LVL2 refines the LVL1 information using full-granularity, full-precision information. The data are collected from the ROBs in $d\eta \times d\phi$ regions of a size determined by the precision of the RoI position, and the type of algorithm to be executed. Typically, the LVL1 decision is verified by recalculating the E_T and improving the cluster position in η and ϕ. All further analysis is then performed in windows centred at this position. The improved evaluation of E_T results in sharper thresholds and allows tighter E_T cuts. Due to the steeply falling E_T spectrum of the background, the E_T selection is one of the main handles to control the LVL2 output rate.

LVL2 can also take advantage of improved, though not final calibrations and thresholds that can be adapted to the trigger object and the luminosity conditions.

The following short descriptions of the algorithms emphasize only the important features. More details are presented in [4-5] and the backup documents quoted there.

4.3.3.1 e/γ feature extraction.

The events selected by the LVL1 e/γ trigger are dominated by $\pi^0 \rightarrow \gamma\gamma$, charge exchange interactions of isolated pions and narrow hadronic showers. They may be rejected by analysis of the shower shape in the e.m. and hadronic calorimeters and by the detailed analysis of the early development of the e.m. shower in the pre-shower, with its fine granularity in η. The algorithm for e/γ identification uses different window sizes from 3×3 to 7×9 standard cells, where a standard cell has the size of $d\eta \times d\phi \sim 0.025 \times 0.025$. The window used for collecting the energy is elongated and sufficiently wide to accept efficiently γs that have converted (electrons from the conversion open up in φ, due to the magnetic field). The algorithms were studied in [4-5]. About 30% of the γs convert before the calorimeter, 75% of these conversions occur in the Inner Detector. Elaborate algorithms to detect conversions in the ID exist [4-7], but are presently considered too complicated to be appropriate for the LVL2. The strategy is therefore to optimize the γ identification criteria such that the γ efficiency is the same for converted photons and unconverted photons.

The lateral and longitudinal shape of the energy deposition is required to be consistent with the cluster shape expected for an electron or a single photon. A hadronic energy veto is applied using the E_T measured in a hadronic tower of size $d\eta \times d\phi = 0.4 \times 0.4$ behind the cluster. The depositions in the fine-grained pre-shower are analysed to be consistent with the depositions expected from an electron or single γ. In a search for the highest and second highest maximum, showers due to e/γ exhibit a high first maximum, whereas π^0s show often two peaks, separated by few millimetres; background jets exhibit many smaller peaks. The selection criteria must be sufficiently loose to accept electrons that experienced bremsstrahlung or γs that converted in the material in front of the calorimeters. These interactions cause tails in all distributions, and limit the rejection power of the LVL2, especially when high efficiency is requested.

To identify electrons with more certainty, information from the tracking detectors can be used. The final rejection factor depends on η; a rate of 0.7 kHz can be achieved at high luminosity for electron efficiencies of 83% and $p_T > 30$ GeV [4-5].

4.3.3.2 Tau/hadron feature extraction.

Tau identification relies on the selection of narrow, isolated jets, associated in the tracker to few tracks. The shower shape and isolation are calculated for the e.m. and hadronic calorimeters separately.

Isolation of the cluster is required for both the e.m. and hadronic parts of the shower. It is determined from the fraction of energy deposited in the core of the cluster, compared to the energy in a larger window positioned at the cluster center. The shower shape is analysed in the e.m. and hadronic calorimeters by forming the energy-weighted radius. Details of the selection cuts used and the estimated efficiencies can be found in [4-5].

The additional rejection that can be achieved in the trigger by including tracker information has not yet been studied; offline studies indicate that the number of tracks should be limited to 1 to reach good sensitivity for the $A \rightarrow \tau\tau$ channel [4-8]. At LVL2, more generous limits will have to be used.

4.3.3.3 Jet feature extraction.

The LVL2 jet algorithm has the task of improving the energy and position measurement of jet candidates found by LVL1. The improvement is achieved by improved energy calibrations, sharper thresholds, and refined jet definition. Using e.g. a cone algorithm on the information inside the RoI window improves also the position reconstruction, the LVL1 resolution being 0.2×0.2 in $d\eta \times d\phi$ [4-9].

An important issue is the amount of data transfer from the ROBs to the LVL2 system associated to jet analysis: many ROBs, e.m. and hadronic, are needed to get all energies in a jet RoI (typically $d\eta \times d\phi = 0.8 \times 0.8$). The bandwidth needed can be reduced by a pre-summation of the cells energy onto a matrix of trigger towers ($d\eta \times d\phi = 0.1 \times 0.1$), but the number of data transfers is still high. We expect the improved selectivity at LVL2 to outweigh the additional load, for some channels. A careful optimization of thresholds will be necessary, particularly for secondary signatures. LVL2 could also select only certain classes of jets for detailed analysis.

4.3.3.4 Calculation of E_T^{miss} and $\sum E_T$

The LVL2 trigger receives from LVL1 the values of the total scalar E_T and the components of the E_T^{miss} vector, E_X and E_Y. These are obtained by summing over all the ATLAS calorimeters, covering $|\eta| < 4.9$.

It is not foreseen to recompute these quantities from scratch (except, possibly, for a small fraction of events at the end of the LVL2 selection procedure, assuming sequential selection). However, corrections can be applied for high-p_T muons (not included in the calculations performed by the LVL1 calorimeter trigger), for saturation in the LVL1 system (e.g. overflow in the 10-bit ADCs used at LVL1), and for mismeasured high-p_T jets for which RoI information will be available.

LVL1 also implements a trigger on $\sum E_T$ which, for many physics processes, can be used instead of the total scalar E_T. This quantity can be recalculated at LVL2 using improved jet information.

4.3.4 Inner detector feature extraction (TRT, SCT and Pixel)

A search for tracks in the TRT and precision layers of the inner detector is used to complement the information from the calorimeter and muon systems. This, together with a refinement of the information from the detectors used at LVL1, allows a large reduction in the rate at which events are passed on to the event filter of EF.

The trigger strategies for the Inner Detector (ID) fall into two categories:

- high-p_T trigger, that is a search for a high-p_T track (p_T > ~5 GeV) in a limited region (RoI) indicated by the LVL1;
- low-p_T trigger, that is an unguided search in the TRT for tracks with $p_T > 0.5$ GeV which are extrapolated into the precision layers when necessary.

While the first trigger type is designed to accept channels with high-p_T muons or electrons from known physics sources (W, Z, top) or new physics (Higgs, SUSY, etc.), the other is designed for B-physics channels based on the topologies of low-p_T tracks.

The LVL2 track searches are performed separately in the TRT and the precision layers using a histogramming method to select sets of points to be passed on to a fit. The advantage of such a method is that execution time scales more slowly with occupancy than a combinatorial method.

Briefly, the algorithm can be decomposed in three steps:

1. Data preparation.
2. Hough transform and histogram: space points are transformed into (slope, intercept) space. The hypothesis is a straight track constrained to the vertex in the r-ϕ or ϕ-z plane, for tracks predominantly in the barrel or the end-cap respectively.
3. fitting: a fit is performed to the points selected by the histogramming stage in order to obtain the parameters of the track.

The highest-p_T track is returned.

Precision tracker (SCT and Pixel)

Hits on adjacent strips (SCT) or pixels are associated into clusters. This is a fast 1-D association for the SCT and a slower 2-D process for the pixels. The SCT detectors are grouped in pairs with the strips of one detector rotated at a small stereo angle to those of the other. The clusters from each detector in the stereo pair are associated to give a 3-D position measurement. The 3-D space points measured by the SCT and pixel detectors form the input to the histogramming stage. The maximum absolute value of the slope for the histograms corresponds to a minimum p_T of 5 GeV. Typical histogram size is of 100×100 bins, while at least five points in a bin are required to make a fit. The least squares straight-line fit is made in both r-ϕ and ϕ-z planes independently, with all combinations of the four or more points, at least one per detector plane. The selection of the points used in the fit is determined only by the histogramming process. The optimum bin size depends on the technology selected for the trigger implementation and is a balance between the times spent on filling the histogram and performing the fit.

Performance of the algorithm, see Figure 4-3, which is described in more detail in [4-10] and [4-11], are reported in [4-5].

It is also shown that, including data from the pixel detectors in the LVL2 trigger significantly improves the track finding efficiency, but at the cost of additional processing time, particularly in the pixel clustering step. The algorithm is described in more detail in [4-12].

TRT

In the case of the TRT, the input data to the LVL2 FEX algorithm are the positional information for all straws in the RoI. The result is based on the number of straws along the trajectory of the track candidate with and without hits (signal passing a low threshold), the number of TR hits (straws with a signal passing a higher threshold) and drift-time hits (straws with drift-time measurements).

The initial track search is performed using a histogramming method based on a Hough transform. In alternative implementations the histogram bins are either calculated at run time, or pre-calculated and stored in a look-up-table. Hits selected by this initial track search are then used in a fit to find the track parameters. Optionally, if the highest resolution is required, the fit can take the drift-time information into account [4-13] [4-10].

The track search may give several candidates with p_T above threshold and the choice of the best candidate is carried on the basis of a quality parameter which is defined, for each track candidate, by the number of straws with and without hits, the number of drift-time hits and

high-threshold TR hits, each with an appropriate weight. The track with a value for the quality parameter above a pre-defined threshold and with the most hits is chosen and passed to the global LVL2 algorithm. The value of the cut is chosen such that the efficiency to find a given track is approximately constant as a function of η.

The large number of points measured in the TRT provides excellent separation between signal and background. Figure 4-4 shows the efficiency as a function of η for single electrons with pile-up when hit and drift-time information is used.

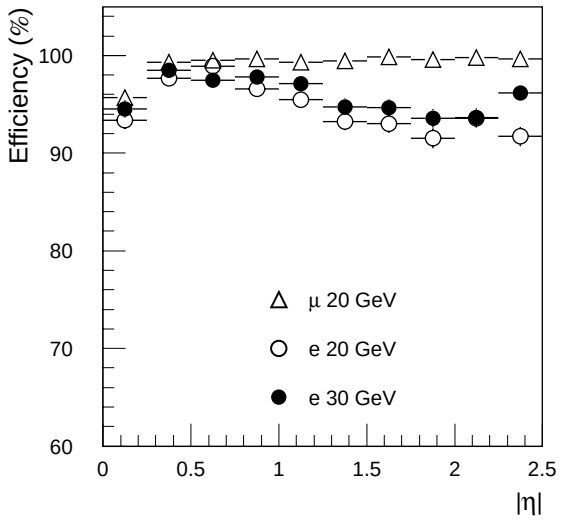


Figure 4-3 Track finding efficiency in the Precision Tracker for muons and electrons.

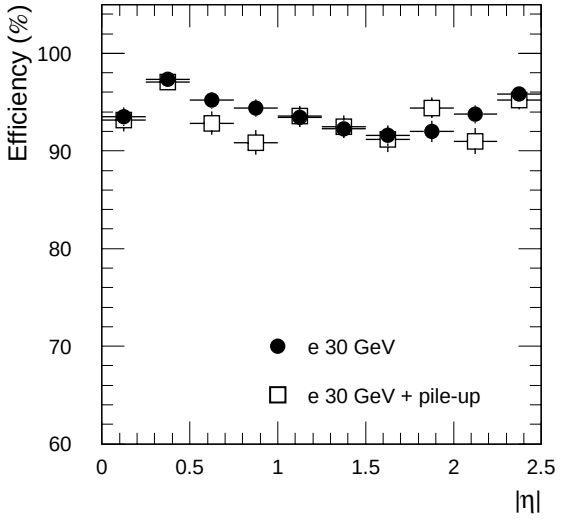


Figure 4-4 Track finding efficiency in the TRT as a function of η for single electrons with and without pile-up, when hit and drift-time information is used.

4.3.4.1 B-tagging algorithm

The reconstruction of the impact parameter in the transverse plane is crucial for a b-jet trigger. This could be computed through a track search and a track fit in the seven layers of the Inner Detector. This method might be used at the trigger level, but only for a very small fraction of the events because of the large computing power needed, especially at high luminosity. An alternative algorithm for b-jet tagging is being developed, which could be suitable for use perhaps at LVL2 or the EF, uses data only from the Pixel detector.

The Pixel LVL2 algorithm [4-11] uses the space points provided by the pixel detector (only in the barrel region for the moment) to reconstruct tracks. The barrel part of the ATLAS pixel detector is made of three cylindrical layers, so the algorithm has to pick, among all the possible combinations, three points belonging to the same track. The search can be performed on the complete detector or inside a RoI. The first operation to be performed is the determination of the space points by clustering together nearby hits. The clustering is done at the module level, so that this could be done as part of the data compression during readout.

An initial track candidate is constructed using a pair of space points from the first and second layers, with the additional requirement that it extrapolates back to the region of the primary vertex. The track is extrapolated to the outer layer and if there is a cluster within pre-defined

limits in this layer the three points are retained as a track candidate. The circle connecting the three points in the r-ϕ plane is then used to measure the transverse momentum and the r-ϕ projection of the impact parameter. The IP resolution obtained is of the order of 60 μm.

The performances of the algorithm are reported in [4-5]. The calculation performed on each combination of points is very simple, so the algorithm optimization consists mainly in reducing the number of combinations to test. Assuming there are on average one or two clusters per module at design luminosity, it is estimated that it will be possible to reduce the number of combinations to be tested to less than 100 per jet.

At present, without any specific optimization and on average 2000 combinations to test, the time needed to process a b-jet is of the order of 1 ms on a 200 MHz Alpha processor. Note that this time does not include the time to associate the pixel hits into clusters.

4.3.5 Benchmarking of algorithms

In preceding sections algorithms have been introduced under the headings of preprocessing, feature extraction, and global decision. The physics quality of algorithms is mainly concentrated in feature extraction and the subsequent use of the features in global decision making. To summarize, the existence of preprocessing reflects the fact that, unlike the case of the offline programs, in the LVL2 environment data are distributed in multiple buffers (the ROBs), and are formatted with constraints originating from limitations in the front-end electronics and readout drivers (RODs). Before feature extraction algorithms can run on such data, the data have to be collected and, possibly, reformatted. These actions compete for processor time with control of the communication processes and some task switching. It is not a priori clear, which processors will be executing which tasks, and with how much parallelism, but in the simplest implementations of the LVL2 trigger, only one general-purpose processor will look after all of these tasks for an event (and for many events simultaneously), resulting in a competition for resources.

Benchmarking consists of measuring how much computing can be done in a unit of time, given a program or algorithm and a combination of resources. In order to be able to model candidate LVL2 trigger implementations, it is of prime importance to estimate the resources required for all contributions, including data collection, communication and task switching overheads, so that the limit of performance can be estimated for a given implementation. In the context of this document, we call benchmarking only the measurement of the CPU time necessary for several algorithms of the preprocessing and feature extraction type.

More information on algorithms is given in the trigger performance document [4-5], in particular the performance of different algorithms in terms of output quality. The algorithms used in obtaining Table 4-4 (also copied from that document) are representative examples of these, but are far from exploring the full set of possibilities; see [4-14] [4-15]. In particular, the relation between necessary resources (like CPU time) and quality of output has been explored only locally, it being understood that ‘quality’ should primarily refer not to precision of computed variables but to the context of triggering, viz. the effect on the overall rejection rate at constant efficiency for the signal events.

A stable set of portable algorithms is not yet available, so that the measurements shown have been made under different conditions on different machines. The numbers should be taken as indicative and should not be used for comparison purposes. We also note that tuning of algorithms on specific systems can result in substantial savings of execution time; this has been done on some, but not all algorithms in Table 4-4.

Table 4-4 Examples of results from algorithm benchmarking.

Detector	Algorithm	System [MHz]	Time (μs)	Comments
Calorimeter	Preprocessing	Alpha [300]	110 (125 max.)	η -dependent
Calorimeter	Cluster finding	Alpha [300]	62 (80 max)	Simple cluster algorithm
TRT barrel	Preprocessing	Alpha [300]	168	At 30% occupancy
TRT endcap	Preprocessing	Alpha [300]	500	At 30% occupancy
TRT barrel	Track in RoI	Alpha [300]	308	At 30% occupancy
TRT endcap	Track in RoI	Alpha [300]	850	At 30% occupancy
TRT barrel	Tracks (full scan)	Alpha [400]	38 000	At 10^{33} $cm^{-2}s^{-1}$
TRT endcaps	Tracks (full scan)	Alpha [400]	125 000	Both. At 10^{33} $cm^{-2}s^{-1}$
Silicon det.	Preprocessing	Pentium [400]	482	Includes pixels
Silicon det.	Track in RoI	Pentium [400]	2000–9000	Many options being explored
Muon ch.	Track in RoI	MIPS [40]	280	early version of algorithm

The long execution time for the TRT full scan corresponds to the time to perform the task in one processor. This indicates that the operation requires substantial CPU resources and either it can only be performed at a low rate or it requires an implementation with substantial parallel processing. An FPGA-based implementation of the full-scan algorithm has been studied with the results shown in Table 4-10.

4.3.6 From features to trigger objects, and decisions

The LVL2 trigger decision procedure has to support different trigger menus. A menu is composed of trigger menu items, each item contains one or more trigger objects and requires the execution of a certain number of trigger algorithms. An example of a trigger item is the combination of an isolated electron with threshold of 15 GeV and a muon with threshold of 6 GeV. One example of a trigger menu, based on physics goals, has been presented in [4-16]. Trigger code is being prepared so that such menus can be implemented on the LVL2 testbeds [4-17]. The global processing steering routine combines the menu with ‘algorithm tables’ defining rigorously the objects and, possibly, the sequence of decisions. Details of the menus to be used for ATLAS will depend on the luminosity and physics priorities.

Identification of the trigger objects can start with the confirmation of the information from Level-1, followed by refinements by building up complete trigger objects by combining features from different detectors, and where necessary combining trigger objects.

An important example of confirming the Level-1 information is for single isolated em clusters, which will make up a significant part of the Level-1 trigger. It is envisaged to use an electromagnetic E_T threshold of ~20 GeV for low luminosity running and ~30 GeV at the design luminosity. At LVL2 the calorimeter algorithm alone may reduce the rate by a factor ~10, and (depending on implementation) it may be indicated (to save data transmission and computing resources) to use the rejection rate offered by such a simple step, thus decomposing the LVL2 decision into two or more ‘sequential’ steps.

The production of trigger objects requires algorithms which then combine the information obtained by feature extraction in different detectors.

For this task the following criteria are used:

- matching of calorimeter and inner detector information for electrons;
- matching of muon spectrometer and inner detector information for muons (especially relevant at lower p_T); track isolation in the calorimeter for muons;
- for B-physics: find tracks in the inner detector without guidance from RoIs, down to comparatively low p_T;
- possibly B-jet tagging by impact parameter.

For electrons, the criteria for matching to the inner detector have been studied using samples of single electrons and jets with and without pile-up. Relatively loose cuts on the agreement in ϕ between the trackers and calorimeter (within 30 and 50 mRad respectively for the precision detector and TRT) reduces the contribution from jets, whilst maintaining a good efficiency for electrons. The selection is further improved by requiring that the ratio p_T is >10 GeV for the precision tracker and >5 for the TRT. With these cuts, plus other refinements (see Section 9.3 of the Trigger Performance Status Report [4-5]), the trigger rate can be reduced to less than 1 kHz with an overall efficiency for 30 GeV electrons with pile-up of ~83% (i.e. track matching efficiency of ~92%). At low-luminosity the efficiency for 20 GeV electrons is lower (~82%) and the trigger rate is ~0.1 kHz.

Criteria for muon isolation in the calorimeter have been studied on a sample of $W \rightarrow mu + jets$ using a background event sample of $bb \rightarrow mu + X$. Only high p_T muons ($p_T > 20$ GeV) have been considered, using their energy deposit in both the electromagnetic and hadronic calorimeters (ECAL and HCAL). The algorithm defines a cone around the extrapolated track, large enough to contain all muon information. An outer annulus extending to a radius in $(\Delta\eta,\Delta\phi)$ of 0.5 is used for isolation. Once the radius of the inner cone has been defined, a combination of cuts have been adjusted to optimize the isolation criteria. An efficiency of ~88% has been obtained in the HCAL and ~95% in the ECAL, with a background efficiency of 10%.

For B-physics, having confirmed a LVL1 muon trigger, further selection at LVL2 is based on reconstructing tracks in the Inner Detector by making a full scan of the TRT. At the analysis level, current studies typically make use of reconstructed tracks with $p_T > 1$ GeV. However, at the trigger level, the initial track scan should find tracks with reconstructed p_T down to about 0.5 GeV, to maintain good efficiency for particles with true $p_T > 1$ GeV (in particular in the case of electrons that may have undergone bremsstrahlung).

Having found the tracks, additional information is required before LVL2 selections can be made. Electrons and muons must be identified, and the track parameters must be determined, including information on the particle direction in space that is required to calculate invariant masses. Since the TRT full scan gives precise information only in the transverse plane, additional information must be obtained from the precision tracker. Electrons with p_T as low as 1 GeV may be identified using transition radiation information from the TRT, and also cluster parameters from the calorimetry. Muons with $p_T > 5$ GeV can be identified by extrapolating inner detector tracks and looking for track segments in the external muon spectrometer. The possibility of identifying in the LVL2 trigger muons with p_T down to approximately 3 GeV, using information from the hadron calorimeter in the barrel region, is under study.

At this stage it will also be necessary to identify and eliminate overlapping RoIs (i.e. RoIs that are likely to have been generated by the same physics object) of different types (the LVL1 system will handle those of the same type).

Further details and examples of possible combined algorithms and strategies are described in the trigger performance status report [4-5].

4.3.7 Trigger menus

It is the task of all work on triggering, and in particular of LVL2 work, to understand, for each physics signature, the event selection sequences and their implications for the overall system. In the preceding section, trigger menus were introduced as a means to describe the physics signatures for the triggering process. In this role, a menu instructs the steering programme in detail which basis to use for deciding the fate of each event, including a concise definition of which algorithms to use for finding variables, and which decision thresholds or limits to apply. Such a trigger menu is a table containing in each line an 'item' defined by a number of trigger objects (like 'electron', 'jet', 'muon'), each associated to criteria to be applied (like isolation, or a minimum p_T value). For an event to be accepted, all objects must be present and satisfy the associated criteria, for at least one trigger item. The menus to study the feasibility of implementation in level-2 must contain all items that require special attention in writing and combining algorithms.

Menus are used in different contexts: sample trigger menus containing a view of the most prominent physics channels have been presented in the trigger performance status report ([4-5]; more detailed trigger menus covering a wider range of physics channels have been presented in [4-18] and [4-16]. The trigger menus can be expected to evolve as physics studies proceed and new results from other experiments are published (e.g. Tevatron physics). Menus will also need adapting to improved physics understanding as the future experiment evolves.

Different menus will have to be used at run time according to the accelerator’s luminosity; presently, a distinction is made between ‘low’ and ‘high’ luminosity, but a more refined distinction is thinkable, to optimize running at any given moment; changing menus used in triggers, at run time, will however be limited by constraints of stability, and by the inevitable loss of time associated to changing tables in the large number of processors using them.

Menus further play a major role in modelling: the choice of physics signatures determine which menu items are to be considered; the associated rates, along with the timing (benchmarking) of algorithms, indicate where computer resources, CP time and transmission capabilities, will be required most. For this application of menus, it is most important to consider the channels dominating the rates, whereas completeness is of lesser relevance; this point will be expanded below (see Section 4.5.3).

4.3.8 An example of sequential selection

An example using sequential selection has been studied for low luminosity operation [4-16], including B-physics. In this example processing and decision making are broken down into six processing steps for the high-p_T physics channels, plus an additional three processing steps for B-physics candidates. Each processing step corresponds to a limited number of trigger algorithms and requires data from a limited number of subdetectors. All data required for all RoIs

concerned by that processing step can be requested at the same time. In this case, the maximum number of data requests for any event is equal to the maximum number of processing steps, nine. Each processing step has an associated input menu to choose the events that need information from that processing step. While waiting for data for the next processing step, the processor can execute algorithms or request data for other events that have been assigned to it. The LVL2 processing steps and the associated data requests for a low luminosity ($L = 10^{33}$ $cm^{-2}s^{-1}$) trigger menu, are shown in Table 4-5.

Table 4-5 Example of sequential LVL2 selection steps for low luminosity.

Analysis	Event rates (Hz)	Data sources	Number of RoIs (avge.)
Trigger RoI	49000	Muon ch., calorimeter	1.6
Trigger track	17000	Inner detector	1.2
Non-trigger RoI	3000	Muon ch., calorimeter	2.1
Non-trigger track	700	Inner detector	1.4
Missing-E_T	2000	Calorimeter	(no RoIs)
b-jet tag	3000	Pixels and SCT	2.3
B-physics candidates:			
TRT full scan	6000	TRT	(no RoIs)
New track pz	6000	SCT	5.8
New track ID	3000	Muon ch., calorimeter	2.1

Note that the missing-E_T and TRT full scan algorithms are based on all detector elements, not on LVL1 RoIs.

The sequential selection reduces the number of events and the number of RoIs that must be processed at each step. Thus in this example, non-trigger RoIs are treated only after reducing by a factor of more than 14, obtained by confirmation of the trigger RoIs. Likewise, time-consuming tracker algorithms are executed only if the faster stand-alone muon or calorimeter algorithms have been confirmed. Such a scheme also allows the execution at LVL2 of more complex algorithms which cannot be executed for all LVL1 events because they require heavy use of the network (missing-E_T recalculation) or long execution times (b-jet tags) or both (TRT full scan).

This selection at low luminosity for this sample trigger menu is estimated to result in an output rate of 360 Hz for high-p_T physics triggers, and of 300 Hz for B-physics triggers.

4.4 Options for architecture and technologies

4.4.1 The architectures

Figure 4-5 shows the main features of a generic level-2 trigger, with interfaces to several external units: a) the level-1 trigger, from which event identifiers and region-of-interest (RoI) information are received; b) readout buffers (ROBs), which contain the raw detector data; c) the parts

of the overall data-acquisition system (DAQ) which either deletes events or passes them to the Event Builder for further processing after a level-2 decision has been reached (a final interface not shown is to the run control system (the Back end DAQ – see Chapter 5) which serves programs and tables to all units, and allows monitoring and error reporting).

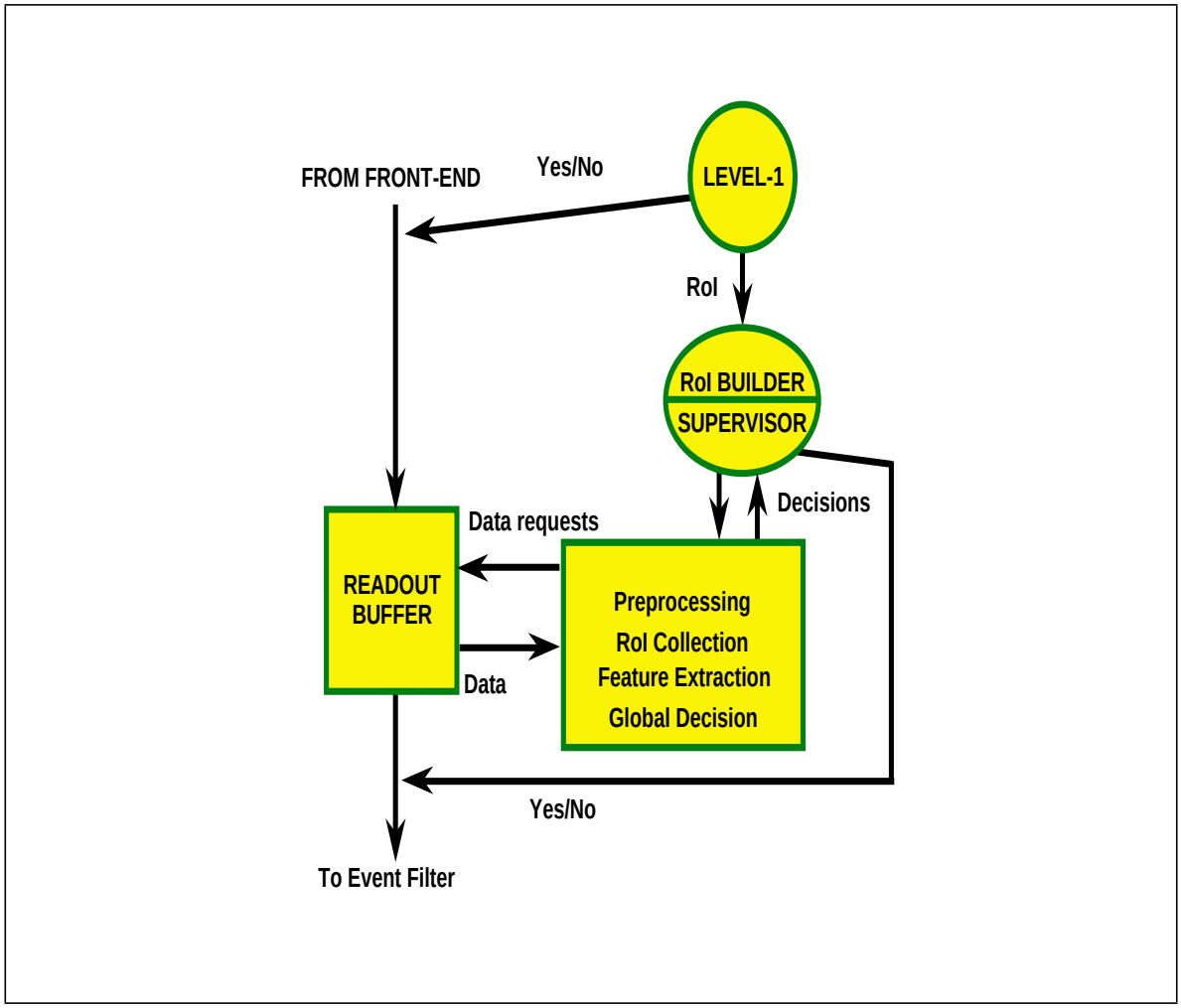


Figure 4-5 Main features of a generic LVL2 trigger.

Inside the level-2 architecture, several functional units can be distinguished: a) the RoI builder and supervisor, where the LVL1 information is transformed and used to steer the data-related operations through the architectural components; b) the data-related operations, which can be decomposed functionally into several distinct algorithmic steps: - preprocessing (which can be local to a ROB); RoI collection (which normally needs access to several ROBs in order to collect all data in an RoI); feature extraction (which operates on data belonging to an RoI in a single subdetector, transforming the raw data into features); global decision (which first combines features from different subdetectors within the same RoI, and then combines information from all RoIs to make a final decision, using a comparison of the level-2-derived variables with physics based selection criteria).

Three representative level-2 architectures were used as a basis for studying the architectural options and components, each potentially capable of implementing all functional steps above.

Note that these were a guideline for study only; they do not pre-empt the possibility of other or combined architectures being implemented for ATLAS. The models used are as follows:

- (A) a local / global partition of level-2 processing, with processing up to and including feature extraction being executed in fast pipelined ('data-driven') processors, and global processing in a farm of general-purpose microprocessors.

  In this model, shown in Figure 4-6, RoI information is communicated by the Supervisor to RoI collectors (RoIC). Data in RoIs are pulled from the ROBs through preprocessing and RoI-collecting units and pushed into a few very fast data-driven feature extractors (DD). Throughout this part of the operation, the processing units can cope with the maximum possible rate of events; the operations are entirely independent in the different subdetectors, and for each RoI a separate data stream is generated. Thus a maximum number of RoIs for each subdetector is defined by the number of separate streams available. After feature extraction, a general switching network mediates the transfer of all feature data of a given event into one processor (GP) of a farm. This latter phase is also controlled by the Supervisor: it allocates the global processor to an event, and receives the decision result, for communication to the DAQ system. Note that feature data from each feature processor can be produced at the full level-1 event rate, which is too high a packet rate for a single port of most general networks currently available. It was therefore assumed that the single high-rate channel from each feature processor is multiplexed to a number of low-rate channels more suited to the general network nodes.

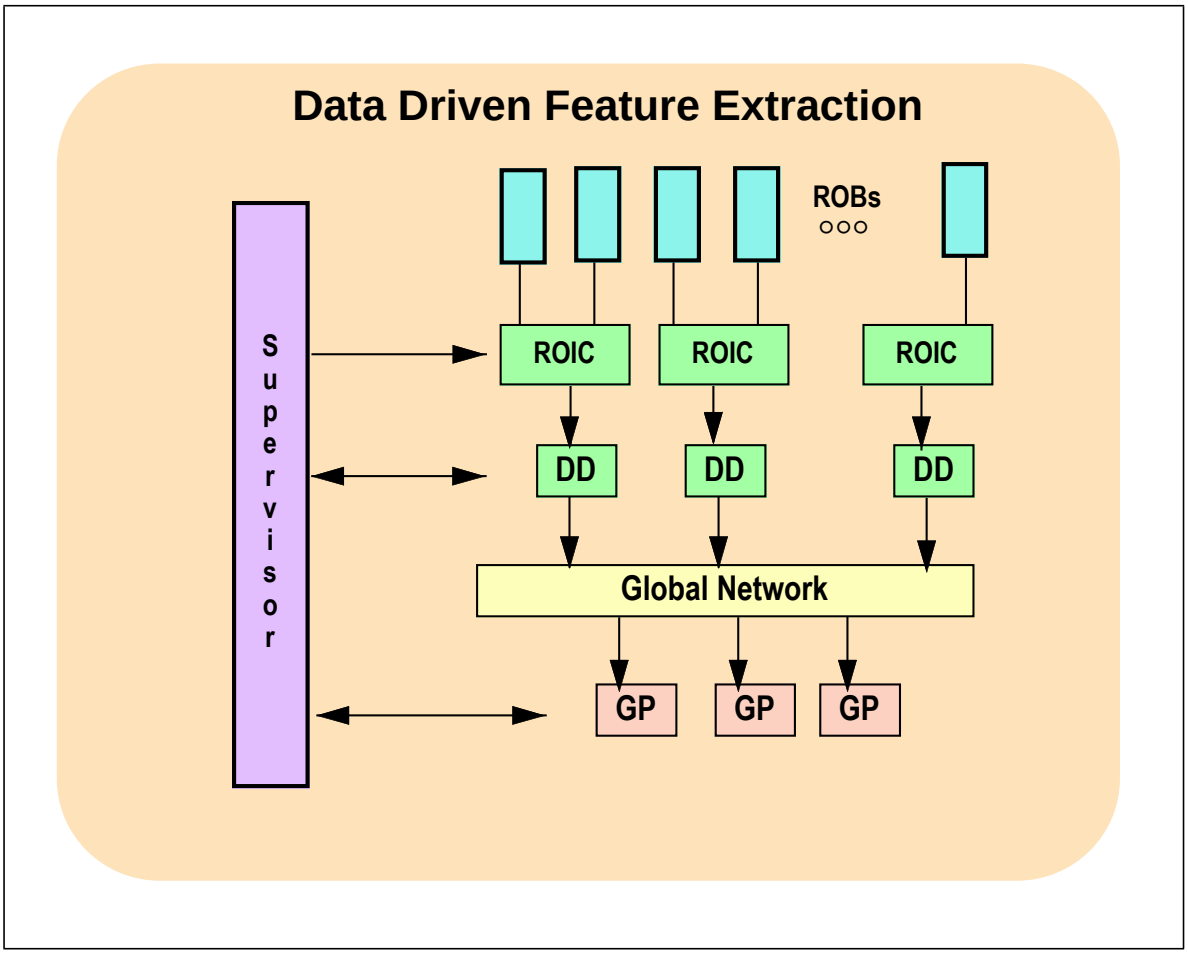


Figure 4-6 Architecture A.

- (B) a local / global partition of general-purpose microprocessors arranged into several farms, all made of identical or similar components, with parallel feature extraction in each subdetector for each RoI.

  In this model, shown in Figure 4-7, RoI information is communicated to the Supervisor, which controls the entire flow of data (the implementation of this model allows a wide variety of possibilities for distributing the supervisor task throughout the system - for the purpose of this discussion we still treat this as a single logical supervisor). The Supervisor control is mediated through the ROBs which push the data to the processor farms. RoI data flow from ROBs through preprocessing units into general switching networks and from there into feature extraction processors (LP). In this phase, subdetectors are entirely independent. The feature extraction processors, in turn, communicate feature data through a network (not necessarily physically separate) connected to global processing processors (GP). Different subdetectors have their own dedicated processor farms for feature extraction. Supervision of these local farms may be delegated to their own farm-supervisors. Again the decision of the global processor is passed back to the supervisor for communication to the DAQ system. Architecture B is optimized for parallel processing.

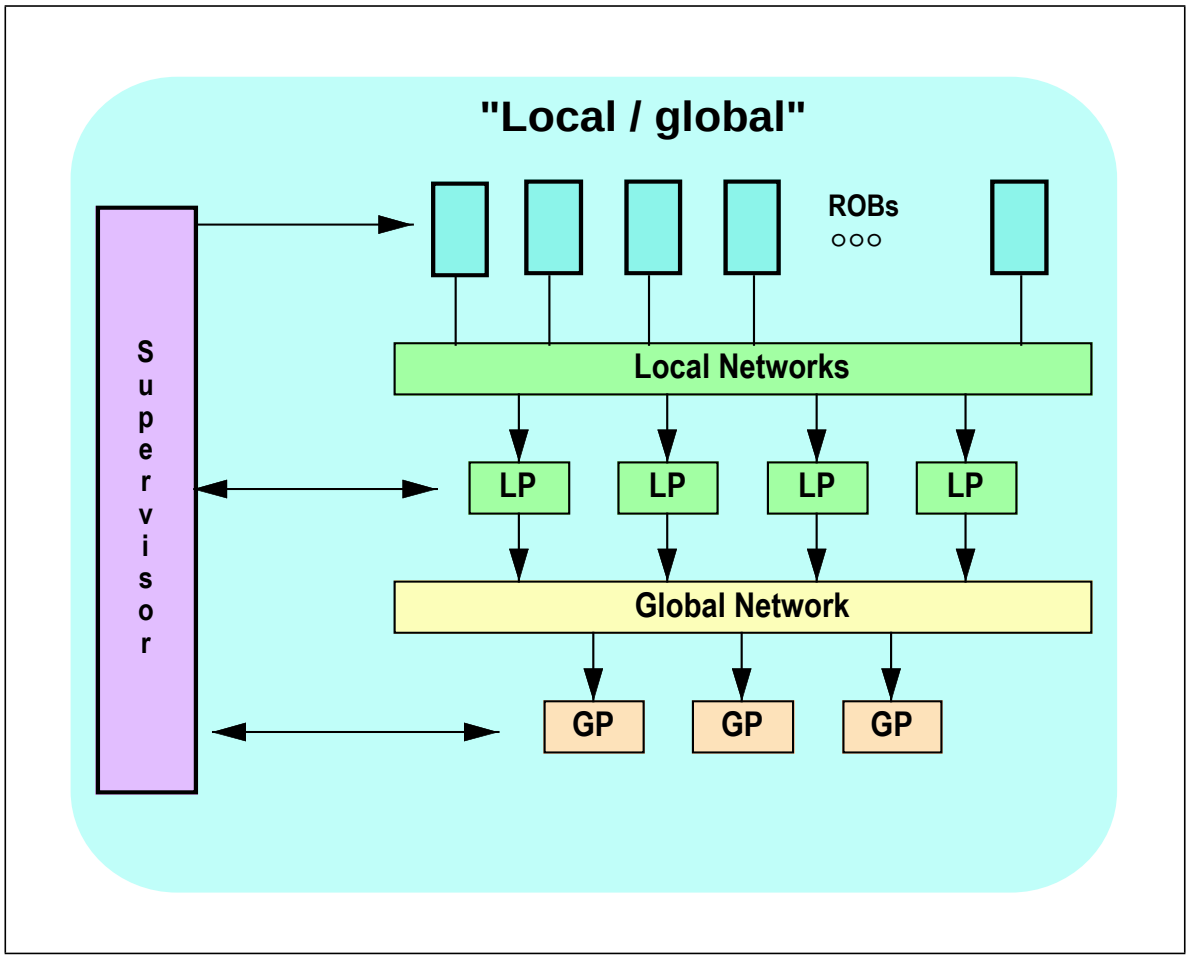


Figure 4-7 Architecture B.

- (C) a single farm of level-2 processors, one processor performing the full level-2 processing for an event.

  In this model, shown in Figure 4-8, RoI information again flows from a central Supervisor, which assigns a single processor node (P) to the event. The Supervisor communicates the relevant information to this processor (via a general network), which then 'pulls', by sending request messages, the RoI data from the ROBs as needed for the execution of the algorithms. The communication between the processors and ROBs, both for control and data, passes through a single, general switching network connected to all ROBs and processors, and to the supervisor. Finally the global decision is again passed back to the supervisor for communication to the DAQ system. Essential features of this model are that all communication passes via a single network and the control of processing within an event is passed from the Supervisor to a single processor node used for that event. Architecture C is optimized for sequential selection.

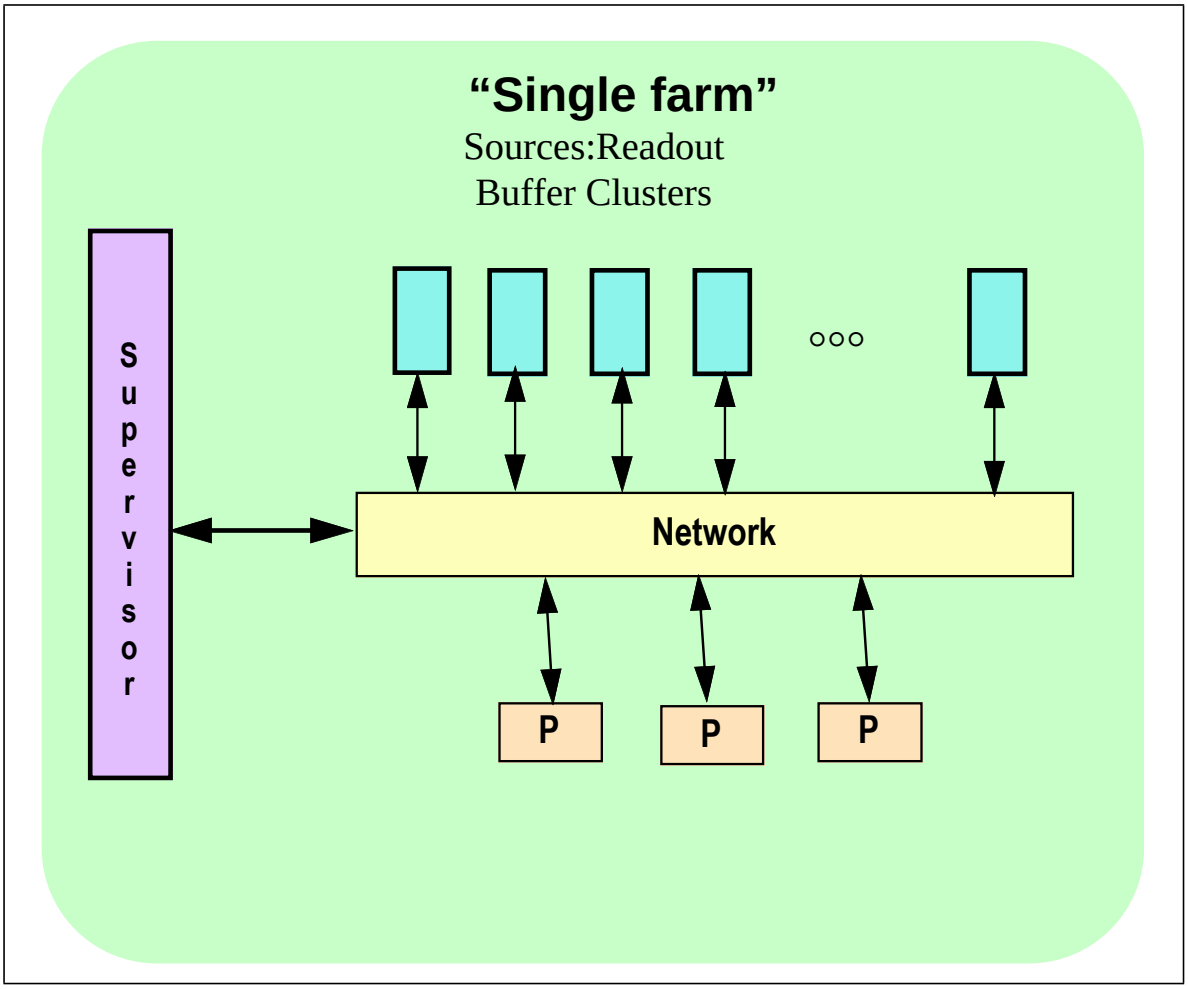


Figure 4-8 Architecture C.

In models A and B, the algorithms up to and including feature extraction refer only to data of individual subdetectors, and run in parallel. In models A and B, close control communication between Supervisor and ROBs is necessary to selectively transmit RoI-related data between ROB and feature extraction processors. In model C, the same control flows through the network. In model A, RoI collection (including local resynchronization) has to occur before transmitting data into the feature extraction units. In models A and B, deviations from the simple

parallel flow of data for different subdetectors, as required for ‘sequential processing’, will also have to be controlled by the Supervisor; in model C, this task is mediated by the assigned event processor.

Model A places the least demands on the network, since it uses custom-designed communication paths for both the transfer of control and raw data traffic at level-1 frequency, with only feature results and final decisions passing via the network. This model simplifies supervisory tasks (it assumes no sequential processing, and there are few processors to schedule), at the expense of a more subtle interfacing between ROBs and feature extraction. Model B passes the raw data traffic over a network, with separate (possibly custom-designed) control communication between Supervisor and ROBs. In this case individual networks may be small and have a natural partitioning. Model C assumes that all communications pass through the network, including control messages at level-1 frequency. For this model a large, single network is required, although partitioning (desirable for test purposes) could also be accomplished by using, for example, a two-stage switch.

In model A, the feature extraction processor must accept all data for an RoI in a subdetector, at level-1 trigger rate. In model B, the feature extraction processor accepts all data for an RoI in a subdetector, whenever assigned this task by the Supervisor. In model C, the processor accepts event and RoI information from the Supervisor; based on this information, it ‘pulls’ all data for an RoI in a subdetector and performs the feature extraction task. Having access to different subdetectors it may perform this task for the several subdetectors, possibly in sequence, and continue with additional algorithmic tasks.

In models A and B, the global processing step receives feature data related to all RoIs and all subdetectors from several feature extraction processors; the step consists of transforming them into a final decision on the event. In model C, the event processing can readily be split into several sequential processing steps, each of which requires only a limited set of features, and each of which is followed by a set of ‘global’ selection criteria. Finally the global decision is communicated to the Supervisor. (For model C it would be possible to continue the processing with Event Filter algorithms.) In all cases, level-2 feature data and decision variables are made available to the DAQ system.

Thus Model A offers a small system with the speed advantages of FPGA processors, and simplified supervision and control, but has the least algorithmic flexibility and would be particularly difficult to include sequential processing steps; model B offers the increased flexibility of general-purpose processors, with parallel feature extraction to reduce the latency and the use of locality to reduce network loads, but it places the greatest demands on the supervisor and sequential processing steps increase complexity; Model C has the greatest flexibility both for algorithms and sequential processing steps, but it requires the greatest connectivity using a single network.

4.4.2 The components and their technologies

The models described above contain a number of components, several of which are common to the different models:

- a Supervisor implementing the functions described below and operating at level-1 frequency;
- ROBs with different options for control communication with the Supervisor, for data and possibly control communication with a network, and the option of RoI collection for the data driven processors. In addition there are options for data-driven preprocessing within

or local to the ROBs and for combining data from several ROBs prior to transmission via the network in an ROB switch-Interface (RSI);

- for architecture A, RoI collectors to route data to the data-driven Feature Extraction processors;
- switching networks to route data to the processing farms;
- processor farms. FPGA processors for feature extraction in architecture A and one or more farms of commercial processors in other cases. One option for the farms is a simple Switch-Farm-Interface (L2-SFI) to decompose processor farms into subfarms each with a single network interface.

4.4.2.1 Supervisor

Critical issues for the Supervisor were the implementation possibilities for the supervisor task, and its communication with ROBs and processors. It was decided to study an implementation based on a small farm of VME-based processors, intercommunicating via a high-speed custom-bus. For this study S-link was chosen for communication from level-1 to the ROBs, plus a suitable network interface for communication to the processors.

The requirements for the Supervisor in the three models are far from identical, with model B placing the greatest demand on this unit. In order to reduce the load on this Supervisor, a separate unit was conceived for model B which handled the distribution of the RoI information within a crate of ROBs and also the allocation of the Local processor. Thus even for Model B the Supervisor had only to consider the allocation of a single processor, and this greatly simplified the distribution of the RoI information to the many ROBs.

The Supervisor is the highest unit of level-2 control in all models. Its general function can be broken down into parts: RoI data control; processor control; communication of the level-2 decision; error detection, reporting and recovery. In architectures A and B, the supervisor could also provide algorithm control for sequential processing.

- RoI data control consists of receiving information about RoIs from the level-1 system, and informing other parts of level-2 about the RoIs.
- Processor control assigns processors of all farms to the respective tasks, possibly on the basis of lists following the occupation of processors; a further task is to impose a maximum response time to processors (time-out); finally, information about event decisions is received from processors, and has to be communicated to the DAQ system for broadcasting to all ROBs.
- Algorithm sequence control implies that for individual events a sequence of operations is followed, which depends on the trigger type and on the event data. A sufficient example is that of B-physics: after confirmation of the trigger muon in the muon chambers alone, the Supervisor could in architectures A and B initiate RoI-independent processing of TRT data; in this mode, the TRT algorithm generates additional RoIs. These are communicated to the Supervisor causing a second iteration of level-2 processing in other detectors.
- Communication of the level-2 decision consists of ensuring a continuation of the DAQ operation after a level-2 decision has been reached; each event is either accepted and will be processed further (event building, Event Filter processing), or is rejected and has to be eliminated, freeing space in ROBs. The Supervisor acts as a central focus for this information, which most likely is transmitted to it by the process(or) that takes this decision.

- Error detection, reporting and recovery. As a central point in level-2 the Supervisor would be likely to play a key role in at least the detection and reporting of errors.

Figure 4-9 shows the functional units of the supervisor and the connections to other systems.

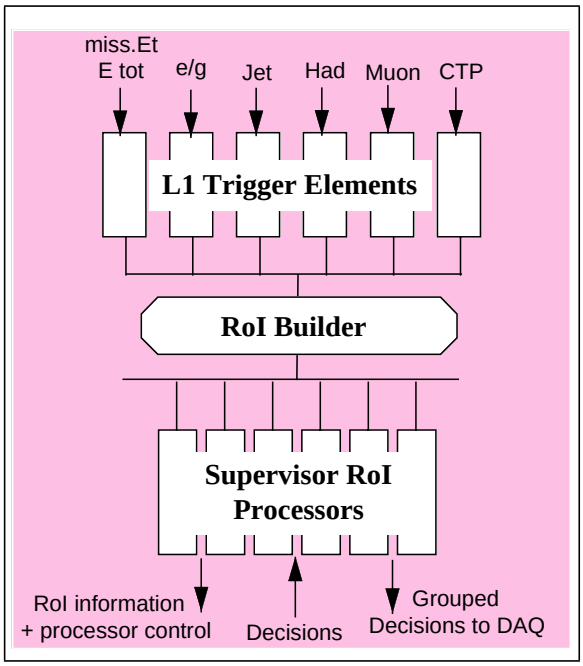


Figure 4-9 Functional units of the supervisor.

RoI data control

The Supervisor in all models must be able to accept level-1 signals at level-1 rate. This information must be communicated to some level-2 units and / or ROBs. In model A the Supervisor communicates event and RoI information to the RoI-collectors; the splitting up of the task of selecting RoI-concerned ROBs can only be between the RoI-collectors and the Supervisor. In model B, the Supervisor communicates event and RoI information to the ROBs; the splitting up of the task of selecting RoI-concerned ROBs can only be between the ROBs, the Supervisor (although with distributed supervision this could be a unit local to a group of ROBs). In model C, the Supervisor communicates RoI information to a processor in the farm; in this case the task of selecting RoI-concerned ROBs can be split between the LVL2 processors, the L2-SFIs, and the RSIs.

RoI builder

The final Supervisor would also contain units to receive and process the information from level-1 to build the RoI records, a list of the RoI types and positions, for each event. However, this RoI builder function is identical in the different models and was not studied during this phase of the work.

Processor control

The Supervisor in all models assigns farm processor(s), or processor nodes, to the algorithm tasks pertaining to a single event. The following supervisor functions occur at level-1 frequency. In model A, the Supervisor assigns a global decision processor (or node) to each event, and communicates this information, together with the RoI information, to the RoI-collectors (for onward transfer to the Feature processors). In model B, the Supervisor assigns to each RoI in each event, a processor in each of the subdetector feature extraction farms, and a processor in the global decision farm; this processor information is communicated to the ROBs, and a control message is sent to these processors. In model C, the Supervisor assigns a processor in the farm to each event, and transmits the event and RoI information to this processor. The Supervisor in all models also receives control signals from a farm processor about the final decision; the signals arrive randomly, at average level-1 frequency. The Supervisor communicates the level-2 decision to the DAQ system.

Algorithm control

The control of sequential processing in models A and B passes through the Supervisor as the only unit having access to all ROBs and (for model B) to all farms. It can be expected that sequential processing needs algorithm control at average frequencies lower than level-1; if we re-

strict ourselves to the B-physics example, this would be (after a first 'standard' level-2 pass with muon RoIs) around 10 kHz.

Communication of decision

The control of communicating the yes/no decision of the global processor to some part of the DAQ system passes through the Supervisor. The decisions arrive at the LVL1 frequency, but they will most likely be grouped in the Supervisor, before transmission to the DAQ system at a lower frequency (perhaps 1/100th of the LVL1 frequency).

If the control of sequential processing is made a task of the Supervisor, its complexity will be increased, perhaps substantially. Asynchronous tasks of quite different frequency will have to be taken care of, separation into different units is limited by the fact that they have to communicate via the same links with ROBs and processors. Fortunately B-physics needs algorithmic complexity at a much lower frequency than level-1.

4.4.2.2 Readout buffer

The ROB is simultaneously the interface between detector-dependent front-end electronics, data acquisition, and level-2. We assume here that the term ROB describes separate memories, corresponding to individual links from RODs; according to present readout descriptions, a ROB of this definition does not exceed on average a local event size of ~1 kbyte (following from the link bandwidth of 100 Mbytes/s and the maximum level-1 rate of 100 kHz). The interface between ROBs and level-2 has the three aspects of control interfacing with the Supervisor, data interfacing with the processor(s) that require ROB data, and additional operations that may be executed as part of these interfaces to alleviate bandwidth and/or control bottlenecks (e.g. preprocessing). An additional task (and possibly interface) arises from the necessity of error handling.

Control interface

In model B, information about events and their RoIs is transmitted to the ROBs from the supervisor. It was assumed that the most suitable transmission is by a broadcast to an aggregate of ROBs, although it was necessary to conceive a mechanism to address only those ROBs that are concerned with each RoI. In model A, this control is mediated by the RoI-collector, the implementation needs further study. In model C, this control by definition passes through the data network, and needs no separate implementation.

Data interface

For each event, some ROBs in subdetectors participating in level-2 algorithms will have to provide their data to a processing unit, for feature extraction. In general, several ROBs may be needed in a subdetector to constitute a full RoI. There is an issue of whether the ROB should transmit all of the data for that event or only the data actually in the RoI (the former placing more load on the data transmission and the later placing more load on the processing power of the ROB). Depending on the architecture model, and the capabilities and cost of the data transmission, there is a critical choice to make in concentrating/multiplexing the data traffic from ROBs into LVL2.

In the simplest form of model A, all ROBs of a subdetector must be available to a single unit (one per RoI) for collecting and sending RoIs to feature extraction processors. In practice it will probably be easier to exploit the locality of the data and split the RoI-collectors into a number of modules with neighbourhood connections. In either case the interface unit will need access to many ROBs and close integration will be required between the active functions of preprocess-

ing and RoI collection (see below). In model B, data blocks corresponding to ROBs are sent upon the Supervisor's request to a predetermined destination processor, data concentration is necessary only for reasons of control (see above) and/or economy (number of network ports). In model C, the ROB interfaces deal with individual requests for data blocks emanating from the processors. Aggregation of multiple ROBs in concentrators (RSIs) would allow matching the required bandwidth with the speed of the network links, and the grouping of event fragments from the ROBs so that larger blocks are transferred by the network and fewer fragments are handled on the processor side. In addition the number of interface cards and network ports is reduced. However, one disadvantage of this approach is that the frequency of requests to the RSIs increases as more ROBs are grouped.

Active functions in the data interface

Here two operations are considered, data preprocessing and RoI collection. A preprocessor is a unit attached, at least logically, to a ROB or a group of ROBs. Its function is to transform the data stored in the ROBs such that transmission bandwidth between ROB and feature extraction processors, and/or algorithm execution in the latter processors are alleviated. RoI collection is the operation of assembling the information of all ROBs (in a subdetector) that are needed to process that RoI, and possibly of re-arranging this information in a format and order to facilitate feature extraction. The preprocessor in all models must be able to accept ROB data at the average rate at which ROBs in the subdetector in question have to participate in trigger algorithms. The frequency to be handled will be higher as more ROBs pass through the same preprocessor. It is also conceivable that preprocessing is done on the (fine-grain) ROB level, independent of any concentration for control and/or data transmission.

RoI collection consists of selecting the subset of data related to the RoI, and of arranging it in a way facilitating the execution of feature extraction algorithms. 'Related to RoI' has been given above the interpretation at ROB level (coarse), a finer-grain selection, ultimately at channel level is conceivable. The RoI collection step is incorporated in or preceded by the preprocessing step in model A. In model B and C, the assembling of information from ROBs and resynchronizing is more likely to happen in the feature extraction processors. The task and its sharing with preprocessing is strongly dependent on details of data formats and data order in the different subdetectors.

Implementation

For implementations of the ROB, it was felt to be desirable for components or interfaces to be kept in common between the level-2 and DAQ/EF '-1' programmes. We had a major interest in using the same modules wherever possible, and in using modular commercial components, if available and affordable. The goals and time scales in the these programmes, however, were quite different and it was not considered appropriate to force a common design. For uniformity (both within level-2 and with DAQ/EF '-1') most tests used buffers based on the PowerPC based RIO2 VME card (the simplest - but an expensive option). For reasons of economy some of the studies used a set of C40-based buffers developed for earlier tests.

4.4.2.3 Networks

General networks are part of all models under discussion. In model A, a single, moderate-sized switching network mediates the transport of data between feature extraction processors and global processors. In model B, several parallel and independent switching networks mediate the transport of data between preprocessors and feature extraction processors; there is one network for each subdetector plus an additional switching network which mediates the transport of data between feature extraction processors and global processors. In model C, a single,

large-scale switching network mediates the transport of data between preprocessors or concentrators and processors, and simultaneously serves the transmission of control associated to data transfers.

It will be particularly difficult for a general switching network to cope with control at level-1 frequency. This suggests for model B separate control communication paths between the Supervisor and both the ROBs and the processors. For model C it is a critical concern since the general network must be used, although the task may be greatly eased by reducing the number of messages required, by for example the use of sequential selection. Technologies chosen for study were ATM and SCI, plus DS-links since, although it was not considered a long-term candidate, significant hardware already existed (e.g. Processors and switches from earlier tests and the large MACRAME 1024 node switch) and it gave an example of a general packet-switching network.

4.4.2.4 Processors

For model A, custom hardware and FPGAs are central to the implementation of the data-driven feature extractors and the associated RoI collectors required to gather at level-1 rate the data for each RoI from the ROBs of a given subdetector. In addition to the hardware there is considerable interest in developing software technology to allow the preparation of the algorithms to be used in the FPGAs using higher level languages.

A comparative evaluation of commercial processors was considered a noncritical topic, and was therefore given low priority. However, the interfacing of networks to the processors and the load the network places on the processor are of great significance. Thus for each network technology considerable effort was made in the low-level software used to drive the network. For example, developing custom-drivers for ATM, rather than using commercially available drivers. Most studies used simple processing nodes, however, some studies of subfarm implementations were also considered desirable.

Hybrid additions to standard farm processors can be seen either as accelerators or as alleviating the task of the farm in the form of a switch-farm interface. They are important in reducing the size of farms, as are multi-CPU nodes, but not as critical as other issues determining the architecture. These items were therefore largely studied separately, although issues such as how they could be interfaced to other components were considered.

4.4.2.5 Software components

Because of the very tight latency requirements it is important to keep operating system overheads under control, indeed some arguments were made against using any operating system in some of the more critical components. Practical considerations led to some diversity in the choice of operating systems (where used), e.g. LynxOS in the ROBs and VxWorks and WNT within the farms. However, the diversity was limited and account taken of the requirement to keep UNIX compatibility, to simplify later integration with other parts of TDAQ.

Although there was considerable separation of the effort on the different models the later merging of these efforts was foreseen and thus efforts were made to layer the software to form the basis for future standard libraries.

4.4.3 Summary

Many of the difficult technical problems are due to the attempt to combine operation at level-1 frequency with a degree of algorithmic freedom more characteristic of an event filter. (Driven in turn by the original definition of the frequency reduction from 100 to 1 kHz assigned to level-2. We maintain that the 1 kHz output rate for level-2 should not be seen as inviolate, it may well be more effective and cheaper to have the cross-over from level-2 to Event Filter at a higher or lower frequency.)

Three representative architectural models have been presented which cover a wide range of options. The three models should not be seen as entirely separate; indeed it is quite possible that the final system will use architectural ideas and technologies taken from all three.

4.5 Results from the R&D programme

4.5.1 Overview and goals

There were many issues to be resolved before it would be possible to decide on which of many architectures should be used for the level-2 trigger. In addition, although there are a number of emerging technologies which look promising for the implementation of these architectures, predicting which of these technologies will deliver the performance required at the best price and with a reasonable assurance of long-term availability is essentially impossible at this stage. Unfortunately while it would be attractive to think that it is possible to study the architectural and technology questions separately there is a very strong coupling between these two issues. However, we had neither the time nor the resources to test all possible architectures, nor to try all possible technologies in each case. It was agreed that the three representative level-2 architectures and the associated components described in Section 4.4 above would be used as the basis for the major studies. Thus components required to build each of these architectures were developed, studied as individual items and then used to construct small demonstrators of each of these architectures. It was also considered very important to ensure that all the best ideas were tested in at least one of these demonstrators and to retain the basic common components and interfaces so that other combinations can be tried in later phases of the programme.

Other hardware activities included studying the performance of network components in an architecturally independent way (the communication benchmarks); developing FPGA hardware for preprocessing; developing hardware for further ROB studies (some in conjunction with the DAQ/EF-1 project); and studying the use of sub-farms. The goals of this hardware work were to obtain better knowledge of the most promising technologies; to measure the component performances which could be achieved with them; where possible to demonstrate components operating at the full rate required in the final system; and to study implementation options for various components.

The scale of hardware produced is necessarily much smaller than the final system and a number of tools have been developed and used to obtain a better understanding of how the components would scale to full-size systems.

First are the ‘paper models’ (initially simple hand calculations, but later implemented as large spreadsheets) which assume likely average component performances and no problems from queuing or statistical variations in events (i.e. perfect scaling). These were used to identify po-

tential bottlenecks and to estimate the number of processors and total network bandwidths which would be needed in each architecture. The goals of the ‘paper models’ were to obtain a better understanding of the full implementation of each architecture studied and to assist in the initial optimization of various implementation options. These models were particularly important in the present phase of the work, however, as our understanding improves better models will be required.

Secondly, more complex models have been run using discrete event simulations which make proper allowance for queuing, variations in processing times, data volumes and event distributions. However, the tools for the discrete event simulation require much greater effort to develop and considerable resources to run and so have as yet been less extensively used. The discrete event simulations have a long-term goal of allowing models with more realistic scaling as the system size increases; however, in the present phase most of the effort here has gone into developing the tools to use as the basis for future studies, although even here some examples of problems due to scaling difficulties have been shown.

Thirdly, access to large network switches has allowed some cross-checks (‘emulations’) to be made on how the average component performance degrades with network size when they are used to transfer estimated traffic patterns likely for the final ATLAS system. In the present phase, emulation has been the main tool to demonstrate the limits of scaling and obtain a better understanding of the use of large switches and mechanisms to avoid the worst drops in performance.

4.5.2 Demonstrator prototypes

4.5.2.1 Vertical slice and component work

A vertical slice is defined as a small subset of the final LVL2 trigger, capable of performing all functions of the system. It has a scaled down supervisor, two to five ROBs, and three to six processors. A simple extrapolation from a system with a few buffers and a few processors to a system with thousands of components is not possible, but together with paper models and large-scale computer models it provides the necessary information to assess the functional and dynamic behaviour of the final system.

Vertical slices were constructed for architectures B and C. This section describes the vertical slices and the main results obtained from them; further details can be found in DAQ Notes 81, 111 and 104 [4-19] [4-20] [4-21]. It should be noted that in addition to the architectural differences, the vertical slices differed in the network technologies used; this was the major technology difference between them. Results concerning individual components are described in the relevant later sections, together with those from other component studies. The components include the supervisor, which receives information from level-1, manages the use of resources within level-2 and keeps status lists of events. The vertical slices used the same basic supervisor design, although there were variations of the hardware and software, especially for the communications with processors and ROBs. This is described in Section 4.5.2.4.

For the model A architecture only the central part of a vertical slice was constructed, consisting of the RoI collector and the FEX, as indicated in Figure 4-10. Details of these are described under the relevant component sections below and in the summary of the demonstrator A group work [4-22] [4-23] [4-24].

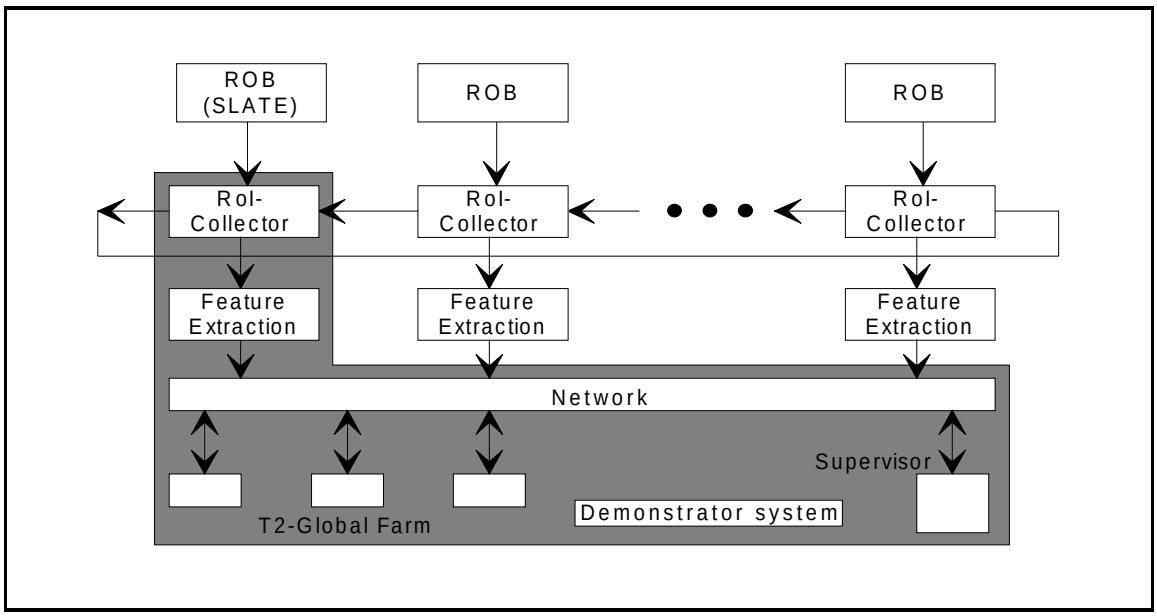


Figure 4-10 Demonstrator-A (shaded) part of architecture A.

The local-global (model B) vertical slice

In the local-global architecture feature extraction for each subdetector is performed in a farm local to that subdetector, the features are then combined in a farm of global processors. In this architecture the supervisor distributes information to the ROBs to push the data to the feature extraction processors and then on to the global processor.

Two thin slices of the trigger were used to demonstrate the message cycle in the local-global architecture and to measure the critical parameters which would determine the rate of triggers such a system would sustain. The first demonstrator [4-19] used DS-links for the communication networks, and although this technology was not considered a serious candidate for the final system it gave a useful cross-check with the large MACRAME test switch based on the same technology, and sufficient hardware already existed from other tests. The second demonstrator [4-20] used Scalable Coherent Interface (SCI) for the networking, with hardware supporting 200 Mbyte/s links. Both demonstrators used a common design of RoI distributor to transmit the RoI records from the supervisor to the appropriate ROBs as described in Section 4.5.2.4.

The DS-link implementation

The Data Strobe Link (DS-link) is a high-speed, point-to-point, bidirectional, serial-link connection, IEEE standard 1355, transmitting data bytes at speeds up to 100 Mbyte/s. The vertical slice is built around the STC104 packet routing device, with 32 data DS-links plus two DS-links for configuration and error monitoring. All 32 links can simultaneously communicate bidirectionally through the switch.

The slice consists of the supervisor, an RoI distributor, plus up to five ROBs and six TransAlpha processor modules. A TransAlpha can act as a FEX or a global processor. Each TransAlpha has a T9000 DS-link communication processor and an Alpha processor, both connected to PCI-bus. The T9000 takes care of the DS-link communication and the event fragment collection. When all data belonging to an RoI (FEX) or all RoI feature records are received (global), it is transferred to the Alpha where the feature-extraction algorithms or global event view algorithms are execut-

ed. The TransAlpha modules can be distributed between the FEX task and the global processing in any configuration limited only by the number of available modules.

The ROB function is emulated in C40 DSPs based on full ROBs designed and built for earlier tests [4-25]. Each DSP has its own special interface module connecting the C40 data link to a DS-link. It is clear that this set-up uses far too many different components for the final system, but it has been very useful for the development and tests of the protocols and functional relations between components.

In the first set of tests the supervisor, the RoI distributor, one ROB, one FEX and one global processor form a pipeline. The supervisor is able to limit the number of events in the system at any given time; this limit is referred to as queue size in the following. Two parameters characterize the system performance: the latency and the throughput. The latency is defined as the time between the supervisor receiving the LVL1 accept and the reception of the corresponding message from the global processor. The throughput is the number of events passing the system per second. Figure 4-11 shows the inverse throughput and latency as a function of the queue size. For queue size equal to one, the inverse throughput and the latency must be equal! As the number of events in the system increases the latency at first stays almost constant. With four events the latency starts to rise with a constant rate per extra event in the queue. An equivalent effect is observed in the inverse throughput. After three events the time between decisions is constant and equal to 127 μs. The effect can be easily explained as queuing in front of the slowest component in the system. As long as the latency over the slowest component times the queue size is smaller than the latency for queue size equal to one, the latency stays constant. When the product exceeds the single event latency, it becomes equal to queue size times slowest component latency.

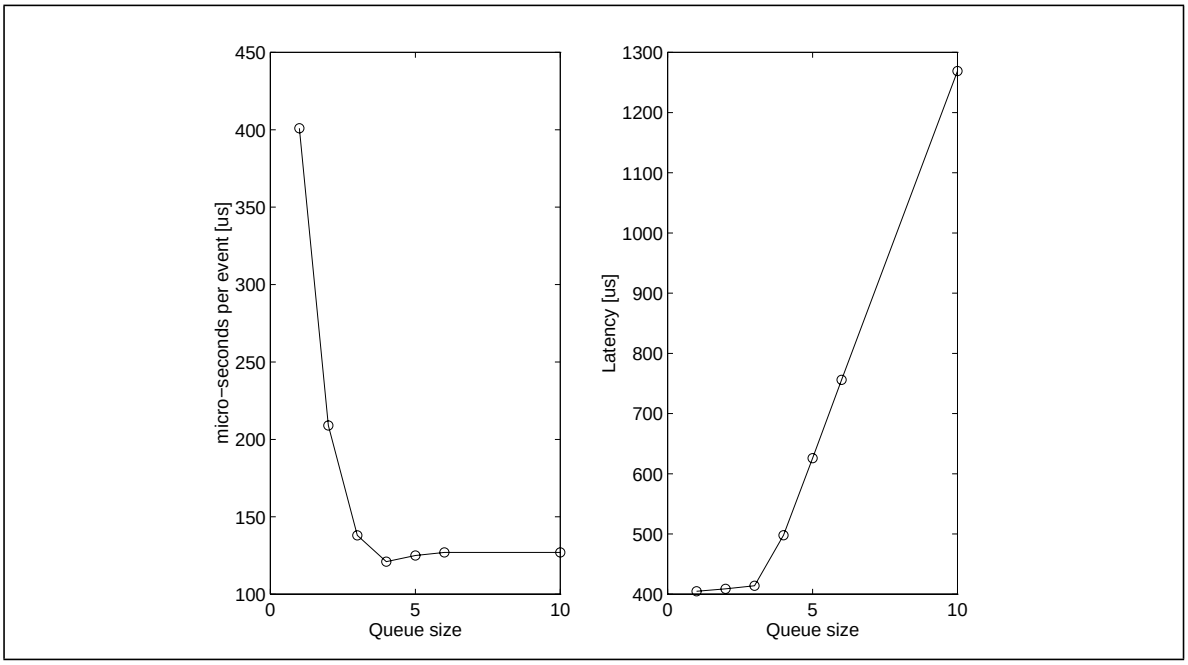


Figure 4-11 Inverse throughput and latency as a function of queue size.

A set of measurements with up to five RoBs, four FEXs and two global processors were made with the aim of finding the maximum throughput and the single event latency. Figure 4-12 summarizes the results. The measurements are reproduced by a simple linear model based on the idea that queuing in front of the slowest component is responsible for the latency when queue size is larger than a few events.

Configuration			Throughput			Latency		Limiting
GTPs	FeXs	RoBs	[μs]/event		[kHz]	[μs]		component
1	1	1	127	(125)	7.9	405	(405)	FeX
1	1	2	151	(125)	6.6	447	(450)	FeX
1	1	3	180	(170)	5.6	487	(495)	FeX
1	1	4	221	(215)	4.5	534	(540)	FeX
1	1	5	262	(260)	3.8	579	(585)	FeX
1	2	1	81	(80)	12.4	408	(405)	GTP
1	2	3	95	(85)	10.5	490	(495)	FeX
1	2	5	139	(130)	7.2	587	(585)	FeX
1	3	1	81	(80)	12.4	409	(405)	GTP
1	3	3	82	(80)	12.2	495	(495)	GTP
1	3	5	102	(94)	9.8	594	(585)	Dist
1	4	1	81	(80)	12.4	408	(405)	GTP
1	4	3	82	(80)	12.2	494	(495)	GTP
1	4	5	103	(94)	9.7	588	(585)	Dist
2	1	1	126	(125)	7.9	420	(405)	FeX
2	1	3	195	(170)	5.1	507	(495)	FeX
2	1	5	287	(260)	3.5	622	(585)	FeX
2	2	1	65	(63)	15.4	455	(405)	FeX
2	2	3	94	(85)	10.6	498	(495)	FeX
2	2	5	148	(130)	6.8	610	(585)	FeX
2	3	1	60	(50)	16.7	442	(405)	Dist
2	3	3	73	(72)	13.7	505	(495)	Dist
2	3	5	99	(94)	10.1	605	(585)	Dist
2	4	1	58	(50)	17.2	451	(405)	Dist
2	4	3	69	(72)	14.5	505	(495)	Dist
2	4	5	103	(94)	9.7	601	(585)	Dist

Figure 4-12 Summary of the performance measurements in the DS-link implementation of model B for a range of numbers of global processors (GTPs), feature extractors (FeXs) and readout buffers (RoBs). The numbers in parentheses are the predictions of the model. In the limiting component column, Dist refers to the RoI distributor.

Figure 4-12 compares measurements and model predictions. In most cases the predictions are within a few per cent of the measured values.

The DS vertical slice provided performance measurements and parameterizations of a real implementation of the LVL2 trigger. It demonstrated that the full protocol can be implemented and run on a system built out of existing hardware components. It showed that a small system has linear behaviour and that the network switch is far from saturated.

The SCI implementation

The Scalable Coherent Interface (SCI) is an IEEE standard for interconnecting multi processor systems. The simplest configuration of an SCI network is a ring, but more complex topologies are possible by interconnecting rings using switches.

These tests used commercial PCI-SCI interfaces [4-26] from Dolphin Interconnect Solutions based on their link controller LC-1 running at a link speed of 200 Mbyte/s. In these tests two modes of moving data over SCI have been evaluated: transparent mode and DMA mode; both described in Section 4.5.2.3.

The vertical slice consists of the supervisor, an RoI-Distributor, plus up to three ROBs and six Alpha processors. An Alpha can act either as a FEX or a global processor. The slice was implemented in two configurations, shown in Figure 4-13. In the first all the ROB, FEX, global, and supervisor nodes were connected with a single SCI ringlet. Packets from a message between any two nodes pass though all the SCI chip sets on the ringlet. A second configuration used an SCI switch to link four separate ringlets. This provided partitioning of the ROB–FEX traffic and gave separation from the Global-Supervisor traffic. It also allowed study of the effect of the transit time of the SCI switch.

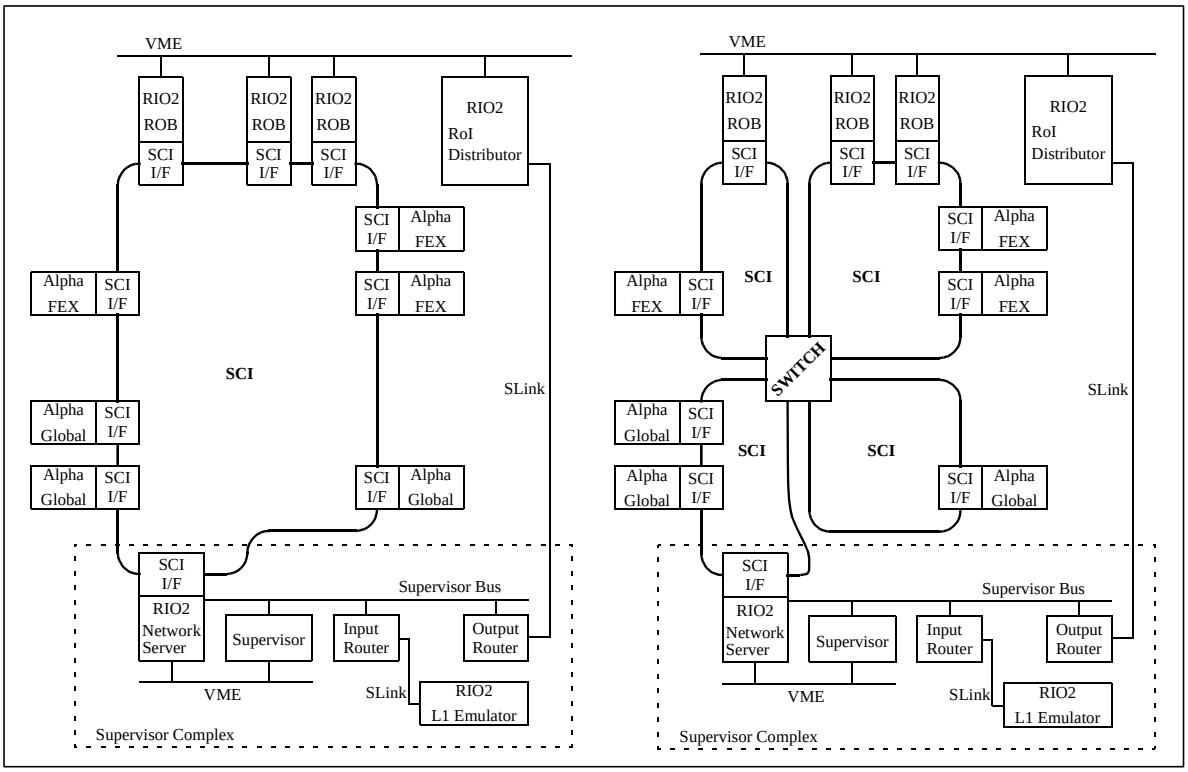


Figure 4-13 SCI implementation of Demonstrator B without and with an SCI switch.

The ROB function is emulated in CES RIO2 VME boards. Three of the Alpha processors were AXPpci33 AT, style single card computers, the other three were Multia units with a very similar design. Both of these designs use the DECchip 21066 Alpha processor with integrated PCIbus controller, all of the Multias operated at 166 MHz, the AXPpci33s at 233 or 166 MHz.

A series of topologies were used to study the effect of pipelining events; the effect of parallel pipelines; combining data from several ROBs into a FEX; and the effect of combining the results from several FEXs into a global processor. These topologies are shown in Figure 4-14. For each topology the latency and event throughput were studied as a function of the ROB–FEX data size and the number of events allowed to be queued (as in the DS-link implementation above).

For a single pipeline of one ROB-1, FEX-1, global and a ROB data size of 1 kbyte the loop latency is approximately 210 μs for transparent mode and 270 μs for DMA mode. If several events are

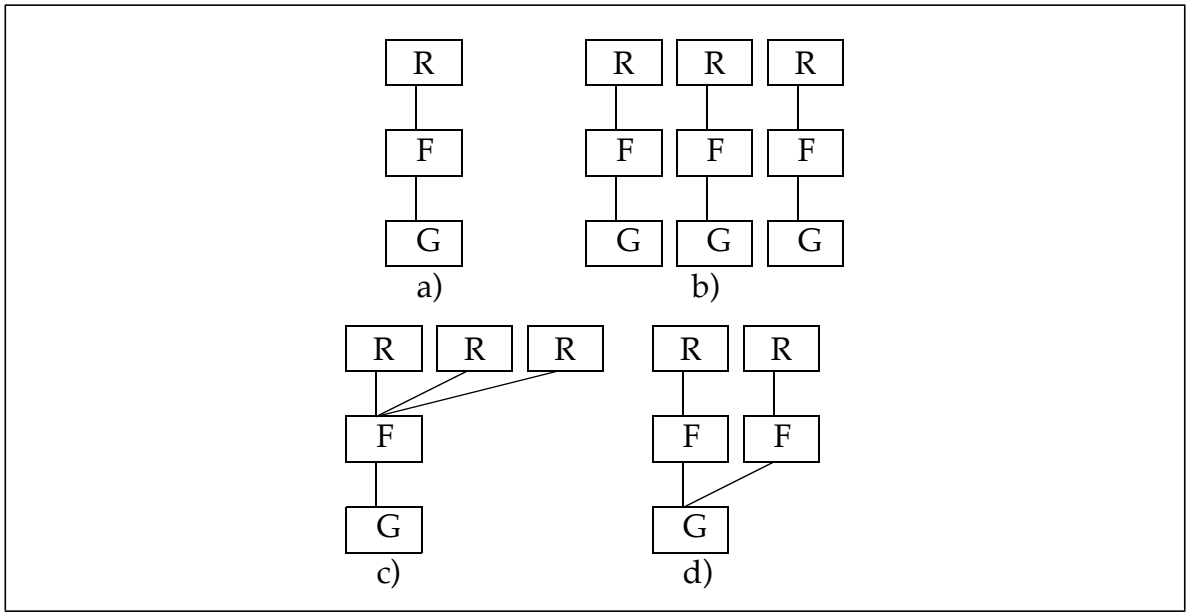


Figure 4-14 Different topologies that were studied in the SCI implementation of demonstrator B, to study scaling, between ROBs (R), FEX(F) and Global(G) processors.

allowed in the pipeline the time between events drops to 90 and 80 μs respectively (with four or more events in the pipeline).

Adding a second and third pipeline increased the throughput, but the heterogeneous nature of the system meant that the scaling was not by the factors of two and three to be expected for a homogeneous system. However, with three such pipelines an event rate of over 20 kHz was achieved, increasing to over 25 kHz if the ROB records were reduced to 64 bytes.

Combining data from several ROBs into a FEX was slightly slower than taking the same total amount of data from a single ROB – this effect was more marked for the DMA mode of SCI transfers, which requires more processing in the receiving node (see Section 4.5.2.3).

Adding the SCI switch to the configuration led to a very small improvement in the performance, indicating that the small additional latency of the switch (< 2 μs) for those packets which had to traverse it was compensated for by the separation of the different parts of the traffic.

These tests showed the very high performance which can be achieved with this technology, not just in simple benchmarks, but with the full message protocols required for the local-global trigger architecture. It should be noted that this performance due to the SCI technology is not only due to a high bandwidth, but also the small load placed on the CPUs of the nodes. The tests also showed that the design of the supervisor combined with the RoI distributor allowed this system to perform very well in spite of the greater supervision requirements of this architecture.

The single-farm (model C) vertical slice

In the single-farm architecture a farm of event-selection processors is connected to the detector ROBs by a communication network that transports event data and protocol messages. This architecture exploits a sequential event-selection strategy [4-27],[4-28] intended to meet the ATLAS physics goals while reducing the data transfer and processing power requirements to a minimum. Following this strategy, a single processor performs all the trigger algorithms for a given event and makes the final LVL2 decision. The event-selection algorithms are executed step-by-step; the data for the next step is requested only if the analysis is still consistent with at

least one set of trigger conditions. A decision can be issued at each step to reject background events as soon as possible.

This vertical slice [4-21] is designed to test the concepts of the single-farm architecture, measure the performance of a real system, understand certain technology issues, and provide feedback to the modelling activities.

Description of the single-farm demonstrator

Hardware configuration

A small-scale demonstrator has been built around an ATM switch (FORE ASX 1000) connecting four sources, four destination processors, a supervisor and a monitor (Figure 4-15). The switch is equipped with 16 ports at 155 Mbit/s. Data sources are emulated by PowerPC single-board 100 MHz computers running LynxOS. PentiumPro 200 MHz PCs running WindowsNT act as destination processors.

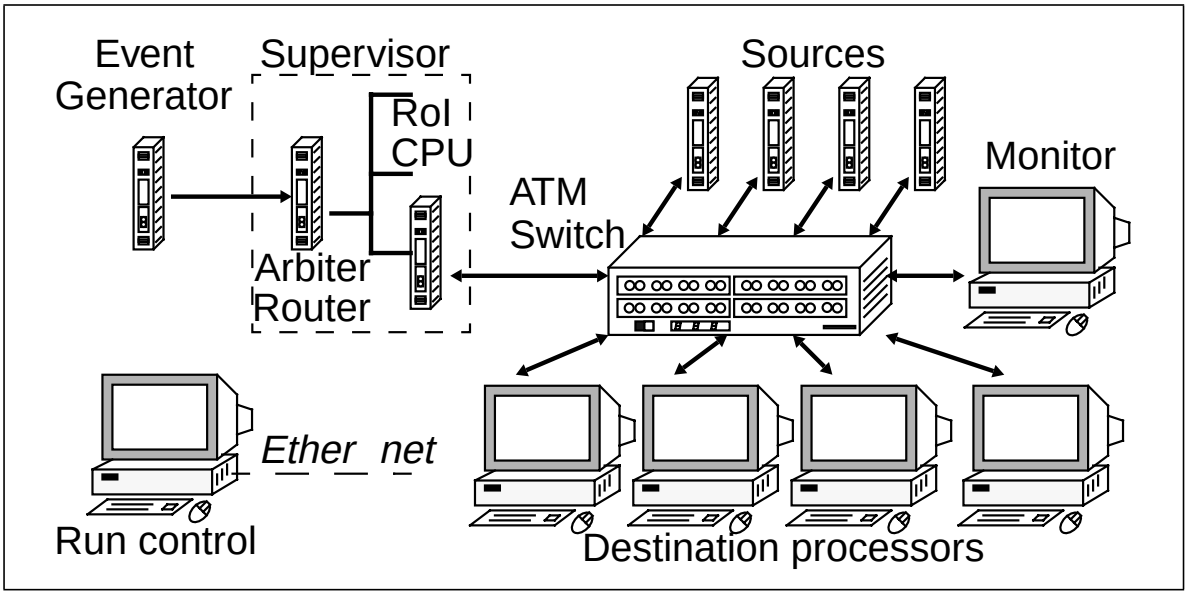


Figure 4-15 Configuration of an ATM-based demonstrator.

Portable code is used to allow flexibility in the configuration of the demonstrator. Any host platform can be a source, destination, monitor, or even a supervisor emulator. For performance reasons and because of mechanical constraints, LynxOS VME platforms are preferred for the source modules and the supervisor. PCs provide a cost-effective solution for the destination processors and the monitor.

The demonstrator set-up and operations are controlled via Ethernet by a workstation. The system configuration and a set of input parameters for the run are stored in a parameter file that is accessed by all nodes at start time. At the end of the run, each node dumps a log file on the screen or on disk for offline analysis.

Structure of the demonstrator software

The demonstrator software is a distributed application that runs on a heterogeneous environment. It has been organized in a way that ensures portability across host platforms and inde-

pendence with respect to the networking technology. The software has a layered and modular structure. It consists of more than 20,000 lines of C code split into three major blocks:

The core of the application contains those parts that are common to all types of node. This block does not depend on the particular operating systems. New types of node can be created quickly because they all use the same core part of the software.

The host-platform and operating-system specific modules implement thread creation and manipulation routines, a mechanism for synchronization and mutual exclusion, and timer routines. At present, this specific part is supported on Digital UNIX for Digital workstations, on LynxOS for PowerPC VME SBCs and on WindowsNT for PCs.

*The networking-technology specific module*s implement a certain number of functions to set up and close connections and to send and receive messages across the network. Currently Demo-C can use either ATM or UDP/IP. The same functions could be implemented for other networking technologies. The ATM layer is based on the optimized library described in [4-29]. By supporting UDP/IP, the demonstrator software can run on any cluster of machines interconnected via Ethernet, or even on a single workstation (to test soft real-time functionality). To further facilitate software development, the so-called 'File IO network emulation' has been implemented. It allows any demonstrator node to run in a stand-alone mode in which outgoing network messages are written to one file and incoming network messages are read from another file.

Network byte order can be chosen by the user. The Demo-C software runs with either big or little-endian byte ordering. Functions have been defined to convert internal representations of structures to or from their network representations.

System components

A *supervisor* with a minimum configuration (one RoI CPU) was used to demonstrate the supervisor operation with the rest of the system. Events accepted by the LVL1 trigger are routed to the RoI CPU by an arbiter/router. The RoI CPU assigns the events to destination processors and collects trigger decisions, packs them and multicasts them to sources. This is done via the ATM network. A credit-based flow-control mechanism avoids overloading the destination processors. A time-out mechanism is used to detect and isolate faulty processors. The performance of the supervisor with the ATM network is described in Section 4.5.2.4.

The *source modules* in this architecture consist of a variable number of ROBins [4-30] connected via a shared media (e.g. PCI bus) to a ROB-to-switch interface (RSI). The ROBins receive the event data from the detector front-ends at the LVL1 trigger rate and buffer them during the event selection process. The RSI is in charge of servicing requests transmitted via the ATM network. When a request for event data is processed by the RSI, it posts a request to each of the ROBins concerned. When all ROBins have replied, the RSI formats the data and sends them to the processor that requested them; a time-out mechanism ensures that a malfunctioning ROBin does not prevent the RSI from responding with data from the other ROBs. The RSI also distributes the trigger decisions received from the supervisor to the ROBins.

For this demonstrator the ROBin functionality is emulated in the RSI processor (single thread application). This approach is sufficient to test the demonstrator system but it does not give an accurate view of the ROBin/RSI interaction. A detailed description of this ROB complex, including the ROBin data buffer and the RSI is given in Section 4.5.2.2, along with results obtained in the demonstrator.

The *destination processor* is in charge of accepting or rejecting the events assigned to it by the supervisor. At each step of the selection process, the processor requests data from the relevant sources for the execution of the next selection algorithm. While waiting for the arrival of the requested data, it can continue processing selection algorithms and/or data requests for other events.

The processor task is implemented in a multi-thread application. On the network side, the Rx thread is in charge of handling incoming packets (interrupt driven) and the Tx thread is in charge of transmitting messages. A certain number of processing threads are in charge of running the selection algorithms (one event per thread). When a new event is received, it is passed to one of the processing threads. This thread posts a request to the Tx thread and becomes idle until data is delivered. When a small number of sources are concerned, an individual message is sent to each source. When a large group of sources is concerned (e.g. the whole calorimeter for missing-energy calculation, or all sources for full event building), the Tx thread sends a single message on a multicast channel to the group of sources. When the requested data has been completely gathered by the Rx thread, the block of data is posted to the processing thread that executes the next step of the selection algorithm. This sequence is repeated until the decision to accept or reject the event can be made. To make the system more robust, a time-out thread periodically scans the list of pending requests posted to the sources. If any of the sources fail to reply in the allotted time, the processing thread is informed that its data request has failed. At present, incomplete events are discarded. In the demonstrator, the selection algorithms are emulated by dummy processing of the desired duration.

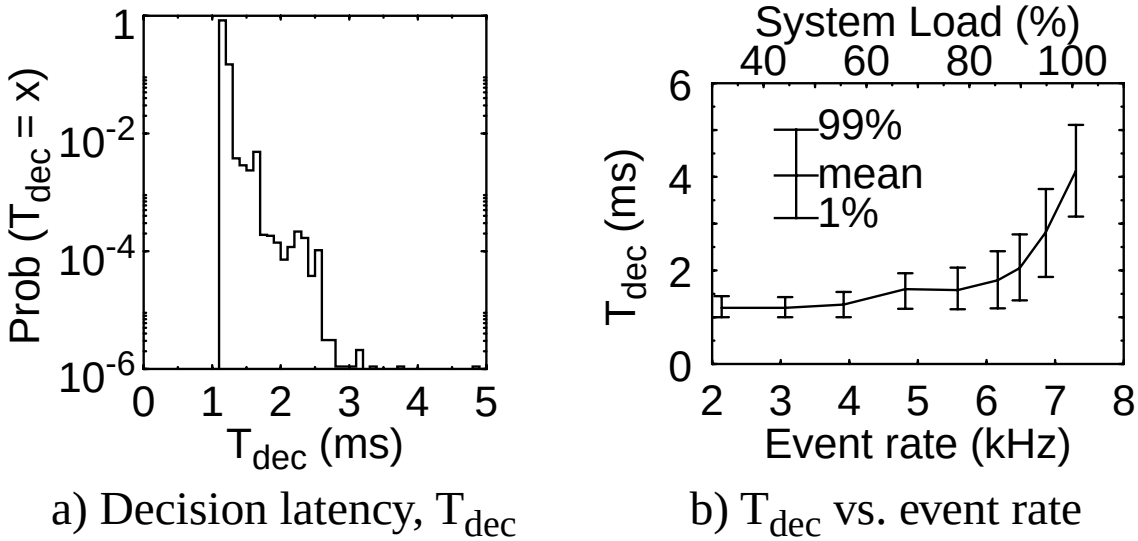
The destination software can execute on single processor platforms or on symmetric multiprocessor machines. It could easily be adapted to run on clusters of processors because the processing threads are independent of the Tx, Rx and time-out threads.

The *monitor* is in charge of regularly checking that all nodes are working properly and taking the appropriate action when a node does not respond. It is also in charge of periodically gathering relevant statistics and making these results available to a visualization program. The monitor itself does not include a graphical user interface and is therefore able to run on any platform. A separate visualization program communicating with the monitor via shared memory could be used to provide a more user-friendly interface. When used without graphical output, the monitor prints results on the screen at regular intervals and dumps the final statistics at the end of the run. Monitoring and gathering of statistics are done via ATM. The real-time constraints placed on the monitor are relatively loose (human reaction time) and the performance of this node is not a critical issue.

System performance analysis for the single-farm demonstrator

The performance of the individual components is given in Section 4.5.2.2 through Section 4.5.2.5. This section presents the demonstrator performance when all nodes are placed together as shown in Figure 4-15. The event decision latency T_{dec} is defined as the time elapsed between the event allocation by the supervisor and the return of a decision by the destination processor. The latency distribution shown in Figure 4-16a is based on the following scenario: For each event, a processor fetches 1 kbyte of data from each of the four sources. The selection is made in a single 100 µs step. The event rate is 3.6 kHz (0.9 kHz per destination), corresponding to ~50% utilization of the destination processors and ~20% utilization of the link bandwidth.

The average trigger latency in this scenario is 1.2 ms, and 99% of the decisions arrive within 1.4 ms. The latency of the communication between the supervisor and the destination processor is ~200 µs. The latency introduced by the destination is ~500 µs (four requests plus the 100 µs



a) Decision latency, T_{dec} b) T_{dec} vs. event rate

Figure 4-16 (a) latency distribution for the 3.6 kHz rate (0.9 kHz per destination) and (b) average trigger decision latency as a function of event rate.

algorithm). The transmission of the requests to the sources takes ~100 µs. The source adds ~60 µs and the transfer of 4 kbyte of data over a 155 Mbit/s link takes ~250 µs. The total amounts to 1.1 ms, which is the minimum observed. Additional delays are due to queuing in various places, operating system scheduling, etc.

The average trigger decision latency as a function of event rate is plotted in Figure 4-16b. The demonstrator operates safely at a rate of 1.5 kHz per destination (6 kHz for the four destinations). The saturation point is around 1.9 kHz per destination. The limitation for this set of input parameters comes from the destination processors. The rate limit is compatible with the model described in Section 4.5.2.5

$F_{dst} = 1 / (215 + 4 \times 45 + 100)\ \mu s = 2$ kHz.

Tests have been made using sequential selection algorithms with several steps having different execution times and requiring different numbers of sources, as shown in Figure 4-17a. The histogram of the latency due to the successive processing steps is plotted in Figure 4-17b. The event rate is 3.35 kHz for four destinations (50% load on the system). The first two steps execute at a priority higher than that of the last step so that new events are not delayed by the small fraction of events with longer processing times. This test demonstrates the capability of handling several events concurrently in destination processors. It also demonstrates the capability of using a single network to carry different types of traffic: requests, event data, monitoring and statistics. Unspecified bit rate (UBR) channels have been used to transfer data for the first and second step of the algorithm and constant bit rate (CBR) channels for the last step (full event building). The monitoring uses low-priority, low-bandwidth CBR channels. This bandwidth allocation scheme is sufficient to avoid network congestion and cell loss.

Summary of vertical slice results

Operation of a local-global architecture (Demo-B) has been demonstrated using two slices with different technologies. The slices provided performance measurements and parametrizations, using the full message protocols to support this architecture. Operation included

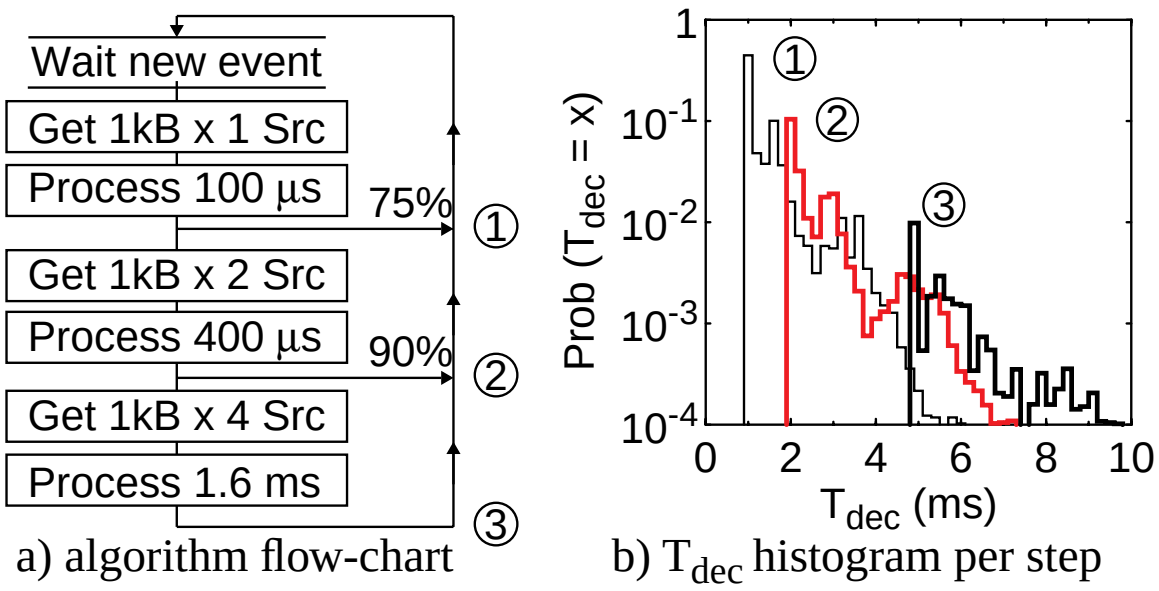


Figure 4-17 Sequential processing in Model C: (a) Algorithm flow chart and (b) decision latency

merging of data from several ROBs, and merging of features into a global processor, plus parallel operation of different processors and pipelining of events. The design of the RoI distributor which also performed the allocation of the Local farm processors handled the extra supervisory load required by this architecture.

The operation of a single-farm demonstrator (Demo-C) using the sequential event-selection option of the ATLAS LVL2 trigger has been presented. The request/response data-collection protocol was implemented. The possibility of making sequential data transfers and sequential selection using partial event data or full event data has been shown. The capability of handling several events simultaneously in a single processor has been demonstrated. The merging of different types of traffic on the same ATM network has been shown. Online statistics gathering and monitoring via ATM have been implemented. Several mechanisms for error detection and recovery were used to ensure the correct operation of the demonstrator. Parameters and simple formulas have been derived to characterize the system.

The results obtained from the Demo-C programme are an important step toward proving the advantages and the feasibility of the sequential event-selection scheme for ATLAS.

4.5.2.2 ROB issues

The LVL2 trigger uses only a small part of the data stored in the ROBs. (Similarly, only a small fraction of the events in the ROBs are transmitted to the event-filter system.) Therefore, the output bandwidth is lower than the input bandwidth. Thus it may be advantageous to group several buffers together to provide a better match for their output bandwidth and rate. In addition preprocessing which leads to large reductions in the data volume, can profitably be done close to the ROBs to yield a reduction in the bandwidth which must be carried by the main network. This has led to various ideas for the ROB complex. Components which could be included in the ROB complex are: the ROBin containing the event memory; a controller to manage the ROB; a coprocessor for local preprocessing; a network interface; and a backbone to link these components. In addition, for Demo-B an RoI distributor was conceived to assist with the distribution

of RoI requests within a ROB crate and the allocation of FEX processors. The collection of RoI data for the FPGA data-driven processors poses special problems since each FEX requires data from complete RoIs at the full LVL1 rate. To solve these problems Demo-A included studies of special hardware in the form of an RoI collector to perform this data gathering from the ROBs. The work and results from the current phase of R&D on these ROB related items are described below.

ROB emulators

In the DS-link implementation of model B the ROB function is emulated in C40 DSPs. Each DSP has its own special interface module connecting the C40 data link to a DS-link. Although this scheme uses far too many different components for the final system, the hardware and software already existed and thus provided a high-performance emulator at little cost.

In the SCI implementation of model B the ROB function was emulated in a CES RIO2 8061 PowerPC board running LynxOS. At the start of a run an area of memory allocated as the event buffer is loaded with event fragments. During the run successive events simply loop over these event fragments.

In both implementations the latency for the ROBs was ~50 μs for 128 byte data blocks. For longer events the latency increased by the extra transfer time which is determined by the effective bandwidth of the output, in both cases up to 20 Mbyte / s.

In the PowerPC-based implementation it was observed that the data output from the ROB had a small but measurable effect on the performance of the RoI distributor. Although this had not been anticipated it is readily explained as follows: The data output uses a significant fraction of the bandwidth of the PCI bus on the PowerPC module, and the RoI distributor accesses the memory of the PowerPC module via VME, which passes by the same PCIbus. Thus as the data rate out of the ROB increases there is increased contention on the PCIbus, which slows down the RoI distributor. Thus although this implementation uses a different path for the data out of and control traffic into the ROB, the two paths coincide in the PCIbus of the ROB.

Measurements relevant to local data collection were made in the SCI implementation of the model B architecture. Here data were gathered from a number of ROB emulators into a single processor over SCI. In each case the SCI transfers were made using the two different modes described in Section 4.5.2.1, transparent mode and DMA mode. In this case the transfer of data from each ROB was initiated by messages from the RoI distributor, because these messages are sent sequentially with an interval of approximately 12 μs the ROBs do not start simultaneously. The results for gathering a total of 2 kByte and 3 kByte are shown in Table 4-6.

Table 4-6 Gathering data from ROB emulators over SCI.

data size (kB)	No. of ROBs	Transparent mode (μs)	DMA mode (μs)
2	1	140	147
2	2	150	156
3	1	193	199
3	2	187	226
3	3	215	250

SCI technology allows the transfers from the different ROBs to overlap each other, so that transferring data from multiple sources might be expected to be faster than from a single source. However, the results do not show such an effect. There are a number of contributions to this. Firstly as noted above the overlap is reduced by the ROBs starting at different times. Secondly although the rate at which the Alpha processor node could receive SCI packets was greater than the rate at which the ROB emulators could send the SCI packets, it was by less than a factor of two. Thirdly there is an overhead for each transfer. All of these effects result in the time required generally increasing as the number of sources increases. It is clear, however, that this effect is less for the transparent mode, where most of the CPU cycles required for the transfer are in the ROBs rather than the destination.

RoI distributor

The transmission of the RoI details to the ROBs in Demo-B is mediated by RoI distributors. For each event the supervisor generates a single record with the details of all of the RoIs in the event, plus for each RoI which subdetectors should send any data in the RoI to the FEXs. This record is broadcast to all of the RoI distributors, each of which services a group of ROBs within a subdetector. Each RoI distributor then loops over the RoIs and checks the subdetector bits to see if its subdetector is required; if so it then checks to see if any of the ROBs it handles are required and sends an RoI request to any that are. Thus the RoI distributor gives a scalable mechanism for the distribution of the RoI information to the ROBs.

In these tests only a single RoI distributor was used, implemented on a CES RIO2 8061 PowerPC board running LynxOS. The processor used an S-link PMC card to receive RoI requests and the grouped LVL2 decision records from the Supervisor output router. It interpreted the message and transferred the requests to the appropriate ROBs using message passing via the VMEbus.

The most frequent message is the RoI request record, which is decomposed into separate RoIs. For each RoI a look-up table, based on η and ϕ, is used to determine which ROBs should be addressed; a second look-up table is used for the allocation of the local processor. Finally the RoI information is transmitted to each of the ROBs involved via VMEbus. When several ROBs have to be addressed this is done by a block move.

For the less frequent grouped decision records these are sent via a chained list to all of the ROBs, since the current VME system did not support a broadcast mode.

The performance of the RoI distributor was slightly different in the DS-link and SCI systems, owing to differences in the ROB emulators. In the SCI system, which was slightly faster, the execution times for the different steps are shown in Table 4-7. In principle the receipt of the supervisor message on the S-link and the output to the ROBs on the VME can be overlapped, but since they both pass via the same PCI-bus on the processor they are effectively sequential.

The ROBin

The function of the ROBin module is to read in data from an LHC-like data source and to buffer it for the duration of a LVL2 trigger decision, making selected fragments available to the trigger system as required. It must be sufficiently intelligent to handle the readout requirements of the different subdetectors. Two designs of ROBin module have been pursued, one following on from earlier work with C40-based buffers as part of the DAQ/EF-1 work (see Section 5.2.2.5) and the other, described below, with a greater emphasis on LVL2 issues. In both cases the primary purposes are to provide a realistic input component for prototyping studies in the devel-

Table 4-7 Performance of the RoI distributor.

Task	Execution time (μs)
Receive an S-link message	17
Processing and DMA set-up	5
Check the ROB	3
Start the DMA	5
VME transfer to each ROB	9.5 / ROB

oping ATLAS DAQ / trigger readout systems and to help form the basis of the eventual ATLAS ROB design.

The event decision latency at the ROBin level is a parameter that determines the depth of the event buffer in each ROBin. Assuming that 8 Mbytes of memory are placed on the ROBin and that this buffer is filled at 100 Mbyte / s, the storage capacity would be 80 ms.

ROBin design

The ROBin includes an input port, an event memory, a processor to manage memory pages and external requests, a PCI bridge for downstream connection and some glue logic.

The input port is a 32-bit parallel port. Data received are stored in a 64 kbyte FIFO to derandomize the dataflow between the input port and the event memory.

The memory must accept a data rate of 100 Mbyte / s, corresponding to 25 MHz transfer with 32-bit data. With a 66 MHz memory, about 50% of the time is dedicated to writing operations. The remaining time is sufficient for read access because the output dataflow is much lower than the input flow. Four Mbyte SDRAM has been chosen for speed and high capacity. This type of memory, which is very widely used in the PC world, is cost-effective. The memory is organized in pages. One event sits on one or more pages according to its size. A page cannot contain more than one event because of the event table management. The minimal size for a page is 1 kbyte owing to the organization of the SDRAM in rows and columns.

The input dataflow up to the event memory is completely hard-wired in programmable logic. The memory pages are managed by the local CPU. Two 16-bit FIFOs are dedicated to the free-page and the used-page queues. An event table located in the CPU memory associates the event number and the data location.

Host requests are received into a circular queue managed by the PCI bridge. The message content is deposited in the CPU memory. The event data are transferred out of the ROBin by DMAs located either in the local bridge or in the host bridge.

If errors are detected by the hard-wired logic, they are counted in registers located in this logic. If errors occur during event or request treatment, they are counted in counters located in the CPU memory. In either case, the event status is updated.

Statistics concerning the ROBin are located in the RAM of the ROBin CPU. The ROB controller can obtain these statistics either by request or by direct memory access.

No operating system is foreseen in the ROBin. Only a small monitor is used to facilitate downloading of the software and debugging. Application development is made on a remote station or on the host CPU.

The ROB-to-switch interface

Within Demo-C a source comprises a variable number of ROBins [4-30] connected via a shared medium (e.g. PCI bus) to a ROB-to-switch interface (RSI), as described in Section 4.5.2.1.

The ROBin functionality was emulated in the RSI processor (single thread application). This approach is sufficient to test the demonstrator system but it does not give an accurate view of the ROBin / RSI interaction.

Figure 4-18 presents the maximum rate of data requests, F_{src}, that can be serviced by a source as a function of the size of the response message.

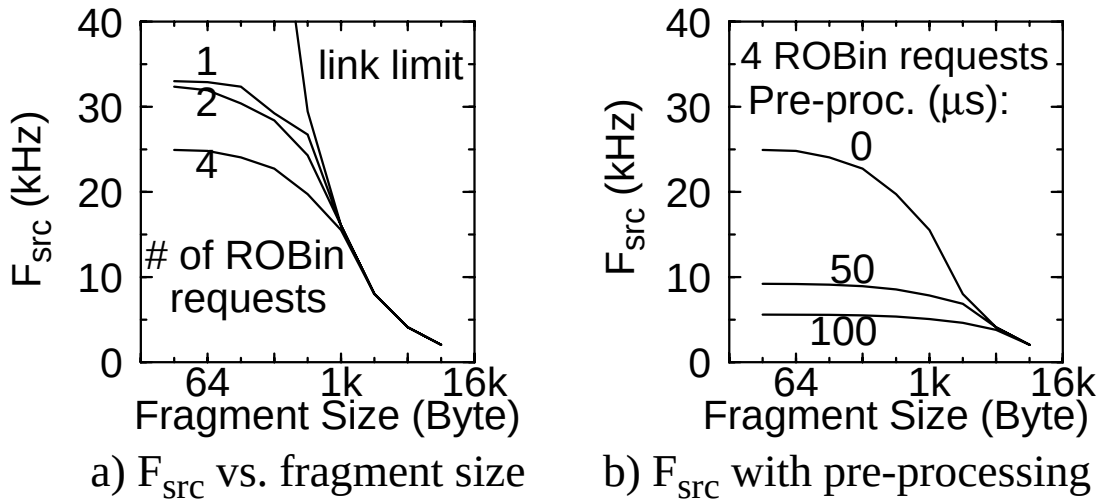


Figure 4-18 Performance of the source modules vs fragment size (a) without preprocessing and (b) with preprocessing.

Without preprocessing (Figure 4-18a), the limitation due to saturation of the ATM output link is reached for packets larger than 1 kbyte. On a 100 MHz PowerPC, the source code executes in:

$T_{src} = 30\ \mu s + 2.5\ \mu s \times number\_of\_ROBin\_requests$

When pre-processing is performed by the source, the maximum sustainable data request rate is reduced (Figure 4-18b).

Because there are no physical ROBins connected to the sources, more conservative numbers have to be considered for realistic sources. Investigations of the ROBin / RSI interaction are in progress.

For a source module, an important parameter is how frequently it participates in an event. This depends on the subdetector, the number of ROBins per source, their granularity, etc. It is estimated that some sources will participate in 20% of the events, so they will have to handle request rates of up to 20 kHz. Even though the results obtained with the ROBin emulation are compatible with this requirement, improvements will probably be needed for operation with real ROBins.

The ROB complex

A model of the ROB complex already started within the LVL2 programme is as follows. It consists of several components: one or several input ports (ROBin) containing event memory, an RSI, a controller to manage the ROB complex, an optional coprocessor for local preprocessing, and a backbone to link all of these different components (Figure 4-19).

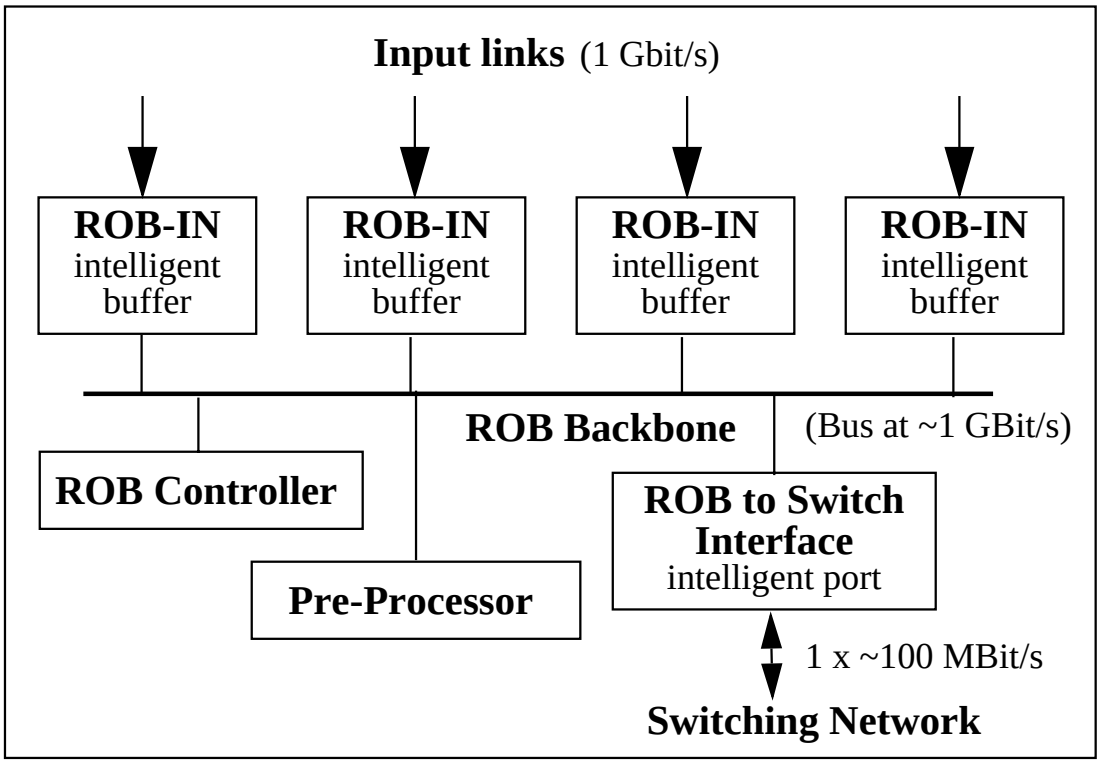


Figure 4-19 ROB complex.

A ROB prototype [4-30] is being designed to test the feasibility of the ROB complex. The design is based on a VME board with a PowerPC and a PCI bus. The PCI bus, with its 33 MHz frequency and 32- or 64-bit width, has the bandwidth necessary to be a good candidate for the backbone. Up to six PCI mezzanine cards (PMC) can be connected to the PCI bus using extender boards. One of the PMCs holds the network interface; the other PMCs contain ROBins or coprocessors. The VME host processor acts as the ROB controller.

A first version of the ROBin has been built in a PMC format, as described above. The integration of the various components of the ROB complex and the software development needed for its operation are in progress. As far as possible, this ROB complex complies with the ATLAS ROB URD [4-31].

The mechanical constraints and the power dissipation limit the number of PMCs that can be connected together on a VME board. In the next phase of this project, an industrial version of the PCI bus, the CompactPCI, will be used to increase the connectivity to eight or more components.

RoI collector for architecture A

Principles of the RoI collector

In architectures B and C all data fragments of an RoI are collected together via a network. In architecture A, however, this task is performed by a special subsystem called the RoI collector (RoIC). The RoI collection components (the RoIC) were built as submodules for the Enable++ FPGA processor system. The RoIC receives RoI fragments from a number of ROBs, assembles the fragments into full RoIs and transmits these RoIs to the associated FEX processor. The input and output connections are high-speed point-to-point links (S-links with GLM or SCSI modules in the demonstrator programme).

The RoIC system takes advantage of the fact that the ROBs can be grouped into crates using their geographical location within the detector. An optimized layout can guarantee that almost all possible RoIs can be assembled from ROBs in neighbouring crates.

Implementation details and measurements

Considering the performance of the available components (links, memory speed, etc.) a demonstrator system with four S-link inputs per RoIC was chosen, three of these inputs being fed by the concentrated output of three ROB crates and one of them by the preceding RoIC system thus closing the loop. The RoIC output is directly connected to two channels of an Enable++ I/O board feeding the FEX processor. Owing to resource limitations the four input channels to the RoIC had to be reduced to one. This had be compensated by a more complex structure of the ROB data sets used. Components only present by emulation were ROBs, Network, Global Farm and Supervisor.

The demonstrator RoIC included two modules – memory and control – each one covering two slots of a seven slot Enable++ I/O board. Up to eight memory modules were managed by a single control module. Configuration and monitoring was accomplished by the host computer of the Enable++ system. Data input was provided by a single S-link, first driven by a SLATE [4-32], later by a microEnable board. Arbitrary distributions of RoIs and ROB contributions were emulated simply by the corresponding definition of the data set.

During the demonstrator programme a number of parameters of this RoIC system have been measured. Results and important properties of the RoIC are presented in Table 4-8.

Overall the RoIC demonstrator system showed a very good performance. The assembling of the RoIs is done with a latency smaller than the time needed for the bare transmission to the FEX, thus leading to a deterministic behaviour. The performance starts to degrade at a fragment size of 64 byte, which is much smaller than with large networks. The event parameters (ROBs/RoI, etc.) are, with very few exceptions, sufficient or better than the requirements defined by [4-33] for all subdetectors.

The present RoIC fulfils the demonstrator requirements and is a good starting point for the approaching pilot project. However, owing to the similarity in their tasks, it seems useful to explore an alternative integrated ROB/RoIC system.

Table 4-8 Results and important properties of the RoIC demonstrator.

	Measured	Comments
Available memory / RoIC	2 Mbyte	Avg. RoI volume is 10–60 kbyte [4-33]
Size of RoI fragments	≤ 4 kbyte	Average is below 1 kbyte [4-33]
Max. no. of concurrent events in RoIC	16	
Max. number of RoIs / event	8	
Max. number of fragments / RoI	24	Usually up to 8, max. is 32 [4-33]
Max. size of RoI	32 kbyte	
Max. fragment rate	740 kHz	Data from 1 ROB only
Input bandwidth	104 Mbyte / s	Max. 200 Mbyte / s @ 50 MHz, large packets
Output bandwidth	240 Mbyte / s	Max. 400 Mbyte / s @ 50 MHz, large packets
Event latency	3–12 μs	Plus fragment delay from ROBs

4.5.2.3 Network issues and results

ATM networking technology studies

The Demo-C programme has largely profited from the experience of the RD31 project in ATM networking [4-34]. ATM technology seems to be adequate to handle the data traffic for large event-building applications. Simulation studies of the single-farm architecture confirm that specific features of ATM allow the construction of high performance networks capable of transporting the various types of traffic characteristic of this application simultaneously [4-27]. Congestion avoidance mechanisms implemented with industrial ATM components minimize the influence of network contention on system performance. Furthermore, ATM supports multicast services efficiently. These features make ATM an attractive candidate for a network that could transfer both LVL2 and event-filter data, as well as data-collection protocol and monitoring traffic.

Within the Demo-C programme an optimized ATM library has been developed for network interface cards that use the IDT NicStar segmentation and reassembly chip [4-29]. The library runs on LynxOS and WindowsNT. It avoids the context switches between user and kernel modes used in traditional network drivers, thus reducing the communication latency and increasing the throughput. It also implements a true zero-copy protocol, placing a minimal load on the host CPU for data movement. This library exploits the full bandwidth of the 155 Mbit / s link for messages larger than 100 byte (2–3 ATM cells) due to its low data transfer and receive overheads of ~10 μs and ~20 μs respectively. This can be compared with the performance of commercial ATM products from Digital with ~20 μs transmission and ~50 μs receive overheads, that allow full bandwidth utilization only for messages larger than 1 kbyte [4-35]. Nonetheless, the trend in high-speed networking technologies indicates that future host platforms equipped with commercial network adapters will be able to make efficient use of the bandwidth offered by these technologies.

ATM seems to be gaining acceptance in the telecommunications market, and 100 Gbit / s switches are now available. Several vendors provide a large choice of LAN backbone switches with 10 Gbit / s aggregate bandwidth at a moderate cost of ~US$ 1000 per 155 Mbit / s port. Although

the acceptance of ATM on the LAN market is relatively slow because of competing technologies such as Fast and Giga Ethernet, an ATM network in the range of 40–80 Gbit/s should be affordable, and adequate for ATLAS needs.

DS-link studies

The DS-link is a high-speed, point-to-point, bidirectional, serial-link connection, IEEE standard 1355. The DS-link transmits streams of data bytes at speeds up to 100 Mbyte/s. Flow control is maintained by adding extra tokens into the data stream. Switching is handled by the STC104 packet routing device. It has 32 data DS-links plus two DS-links for configuration and error monitoring. It performs dynamic packet-switching based on the content of the packet header. All 32 links can simultaneously communicate bidirectionally through the switch. The vertical slice for Demo-B offered few new insights into the functioning of DS-links. However, much experience has been gained by other studies of DS-links, especially those within the MACRAME project described in Section 4.5.3.3 below. The work on DS-links has evolved smoothly into studies of Fast and Gigabit Ethernet. Baseline measurements have been carried out and extensive work has been incorporated in the pilot project work plan.

SCI studies

Within ATLAS the studies of SCI [4-36] have greatly benefited from the earlier RD24 project [4-37] and the still active EU-funded SISCI project (ESPRIT Project 23174, Standard Software Infrastructures for SCI-based Parallel Systems). An SCI network can be seen as a logical bus but it is actually constructed from unidirectional point-to-point links between devices (typically processors) forming a ring structure. In addition to rings more complex topologies are possible by interconnecting rings using switches. Communication is based on the exchange of small packets with a split transaction protocol. An SCI packet has a 16-byte header, a 16 - or 64-byte payload and a two-byte CRC.

Although commercial software is becoming available for more and more platforms, all of the software used in these tests, including simple device drivers, has been developed internally. Alternative software is being developed under the EU-funded SISCI ESPRIT project with the aim to provide a standard low-level API.

Recent tests have concentrated on commercial PCI-SCI interfaces [4-26] from Dolphin Interconnect Solutions based on their link controller LC-1 running at a link speed of 200 Mbyte/s. With this interface the performance into or out of a node tends to be entirely limited by the performance of the PCI on the driving processor card. For example, transfers between the RIO2-based ROB emulators and the Alpha processors tends to be limited to less than 20 Mbyte/s (for 1 kbyte blocks), whereas between a pair of Alphas this can nearly be doubled.

The Dolphin interface offers a number of modes of use. Within these tests two ways of moving data over SCI have been evaluated:

- Transparent mode to transparent mode, or remote shared memory. The sending CPU wrote the data message into a buffer in its own virtual memory and the SCI hardware copied it directly into the memory of the remote node. A message-passing library was implemented over the shared memory, such that the receiving CPU only had to poll a set of control words (one for each sender) to determine when a message had arrived.
- DMA mode to ring buffer mode. Here the sending CPU had to set-up the DMA engine on the source SCI interface to copy data from the source to the SCI destination. The receiving SCI interface placed the data into a previously specified ring buffer. In this mode the re-

ceiving CPU has to poll a CSR register on its SCI interface to determine when a message had arrived, but then also had to copy the data from the ring buffer into the working space.

Thus transparent mode places a greater CPU load at the sender, whereas the DMA mode has a greater CPU load at the receiver.

Many tests have been performed with small configurations to understand the basic performance of the SCI interface with each processor board and to study the various modes and options for SCI operation [4-20]. For example with a pair of 166 MHz Digital Alpha processor boards on a ring transfer rates of over 35 Mbyte / s have been measured. If the nodes are placed on separate rings connected by a Dolphin four-port switch the performance drops by 10–20% due to the transit time across the switch of ~1.5 μs. Tests have also included measurements of retry traffic, which although normally negligible, can occur if an SCI packet arrives at its destination node and the node is unable to receive it because the input FIFO is occupied.

Related tests of SCI, allowing comparison with other technologies, are also described in the section on communication benchmarks below.

Communication benchmarks

A comparatively naive set of application-independent communication benchmarks has been designed and implemented over a number of technologies. This section gives a brief description of these benchmarks and the results obtained. A fuller description can be found in [4-38].

The goal of these tests was to derive from these detailed results several basic communication parameters. They include obvious ones like bandwidth, but also various overheads associated with switching technologies or arising from interfacing to operating systems, and measures of traffic interference. The tested systems include a number of different architectures from clusters of workstations to tightly coupled massively parallel systems. Also included were the technologies that were prominent in the ATLAS demonstrator programme, SCI, ATM and DS-links.

Description of the benchmarks

The basic communication benchmarks used were ping-pong, two-way, all-to-all, pairs, outfarming and broadcast, and funnel. To these were added two application-specific tests of push-farm and pull-farm. All abstract basic benchmarks are executed for a varying number of packet sizes (minimum, 64, 256, 1024) and a varying number of processors where applicable (2, 4, 8, 16, 32, 64). Packet sizes are restricted to those expected in our application, although some implementations have scanned a wider parameter space.

The basic ideas of each benchmark program are as follows (N is the total number of nodes):

- Ping-pong: One node sends data to another node, and waits for the data to be sent back. The benchmark measures the round-trip time.
- Two-way: Both nodes are sending and receiving data simultaneously.
- All-to-all: All nodes are sending to each other simultaneously. Increasing the number of nodes in the system increases the overall throughput.
- Pairs: $N / 2$ nodes send to $N / 2$ receivers unidirectionally.

- Outfarming and broadcast: For outfarming one node sends packets to $N-1$ receivers, while broadcast uses hardware broadcast, if present. Thus in outfarming the data could be different in each send, whereas broadcast sends always the same data.
- Funnel and push-farm: In the funnel benchmark $N-1$ senders send data to one receiver.

  The push-farm represents a type of communication, in which the data is sent from $N-1$ nodes to one receiver, in much the same way as in the funnel. The difference is that in the push-farm benchmark additional computing cycles can be included; the computing represents analysis of these measurements, and we execute dummy code lasting 0, 100, 400 or 1600 microseconds. Each time before the computing cycle is started, the request for the next data item has been issued, allowing overlap of computing and communication.
- Pull-farm: Pull-farm represents a type of communication, in which first a control message (64 byte) is sent from the receiver to $N-1$ senders, and subsequently an amount of data (1024 byte) is received back from each sender. Computing cycles can be included in the same way as in push-farm.

Of particular relevance to our application are the benchmarks ping-pong, pairs, push-farm and pull-farm. The latter two obviously have been specifically designed to correspond to communication patterns typical for our application. Ping-pong tests the request–acknowledge cycle, which is needed in several kinds of transmissions. The pairs benchmark tests one-way communication performance from point to point, which is characteristic for communication without need for acknowledgement.

Benchmark Implementation

The benchmarks have been implemented in different technologies by designing a separate intermediate layer for each technology. This layer contains the message-passing routines, such as sending and receiving the message, initialization, cleaning up and broadcasting. The routines include non-blocking send and receive operations.

The measurements have been done in a number of technologies:

- SCI on a RIO2 8061 embedded processor board using the LynxOS operating system, on PC under LINUX, and on Digital Alphas under Digital UNIX;
- Digital Memory Channel connecting Digital Alphas;
- ATM on RIO2 (and RTPC) embedded processor boards, with the LynxOS operating system, on PCs under Windows NT, and on Digital Alphas under Digital UNIX;
- Cray T3E shared memory application programming interface;
- Raceway bus, using a Matra Systems and Information PeakWare toolkit;
- T9000 using IEEE 1355 DS-links (GPMIMD);
- Meiko CS-2 communication library.

In addition, shared memory and Message Passing Interface (MPI) have been used in several systems.

Notes:

- The bandwidth results with SCI were limited by the performance of PCI buses.
- The basic benchmarks with ATM used undefined bit rate (UBR) connections. The data was sent at full speed; as ATM has no flow control, the receiver can in some cases lose

packets. This is especially apparent in pairs, funnel and push-farm benchmarks. In the funnel benchmarks, when four senders and large packets were used, no meaningful measurements were possible.

- The fast RIO2 results for the pairs test demonstrate the performance gain which can be achieved by investing in low-level design work with ATM drivers compared with directly available commercial implementations.
- The performance difference between the measurements with different generations of processors from Digital is surprisingly large, which should be attributed not so much to the faster processor technology (communication benchmarks are not very CPU intensive), but to other architectural changes, which have taken place in recent years.
- In the push-farm tests the receiver processor establishes a constant bit rate connection with each sender. The bandwidth of this channel is 1/(number of senders) to avoid congestion in the switching network and the receiver.

Benchmark Results

The large number of different benchmark results have been condensed into a few meaningful parameters, in Figure 4-20.

The parameters are the following.

- In ping-pong: the latency derived from dividing the round-trip time for the smallest packet size by two.
- For the two-way test latency has also been measured. However, in this measurement the time has not been divided by two, since both the nodes are sending and receiving data simultaneously. Note that the two-way latency is larger than in ping-pong, since set-up times of both sending and receiving are included.
- In the pairs benchmark: overhead and effective bandwidth. Overhead is the unidirectional communication speed. The effective bandwidth has been calculated from this benchmark using a packet of 1 kbyte.
- A ratio has been extracted from the outfarming and broadcasting benchmarks by dividing the outfarming ratio by the broadcasting ratio. The ratios have been calculated by dividing the time for 2 N nodes by the time for N receiving nodes, from two to four, four to eight and eight to sixteen processors when possible, from which the average is taken. The parameter shows how well the broadcasting has been implemented in each system: the larger the number, the more efficient the broadcasting compared to the outfarming performance of the same system.
- The funnelling (and push-farm) benchmark: cycle time is the time for four nodes to send a 1 kbyte packet each to one receiver. Where up-to-date funnel results have not been available, push-farm results were used instead.
- Pull-farm: cycle time is the time for the receiver to send a 64-byte control packet to four data senders and for each of these data senders to each return a 1 kbyte packet.
- Overlap: an estimate of the overlap between communication overhead and computation in the nodes. In this case taken from the push farm.

Architecture	**ping-pong** latency µs	**two-way** latency µs	**pairs** overhd µs	**pairs** bandw. MB/s	**broad cast** ratio	**funnel** cycle time µs	**pull-farm** cycle t µs	**over-lap** %
ATM, RIO	104.8	108.9	9.1	16.2	2.05	250 **	376 **	-
ATM, DEC	85.4	90.3	33.3	13.6	-	-	-	-
Cray T3E	4.8	5.5	5.0	57.4	1.55	35.7	57.2	12
Cray T3E (MPI)	21.7	34.4	19.2	19.1	1.67	243.7	349.6	0
DEC MC	6.4	7.8	3.7	31.2	1.92	83.3 *	90.0	71
DEC MC (MPI)	26.5	51.3	21.3	11.4	1.28	-	-	-
DEC 8400	4.0	7.8	4.5	32.2	1.15	128.3*	138.3	0
DEC 8400 (MPI)	13.3	22.9	11.0	13.1	1.60	-	-	-
GPMIMD	6.6	-	13.1	3.1	1.05	1132.4	-	-
IBM SP2 (MPI)	74.2	88.5	31.0	10.8	1.56	-	-	-
Matra, Raceway	8.8	12.5	10.0	59.0	-	47.5	-	-
Meiko (Channel)	20.3	34.4	22.4	11.4	1.88	124.0*	172.0	78
Meiko (MPI)	128.5	137.0	102.0	6.2	1.23	-	-	-
Parsytec	217.0	354.0	188.0	3.7	1.50	1103	-	-
SCI, DEC	9.9	14.9	8.3	38.5	-	-	-	-
SCI, RIO/DEC	12.5	18.1	9.9	21.2	-	76.6 *	98.3	61
SGI Origin	7.1	12.7	8.8	31.3	1.25	178.3	207.2	-
SGI Challenge	12.9	20.1	14.0	12.2	1.23	262.6	-	-
SGI Chall. (MPI)	66.1	81.4	34.8	5.8	-	546.4	-	-

Figure 4-20 Parameters extracted from benchmark results. Push-farm results were used instead of funnel, when either funnel results were not available or push-farm results were considerably newer. The push-farm results have been marked with an asterisk (*). Push-farm and pull-farm results of ATM, marked with a double asterisk (**), have been achieved by slightly different measurements, and are described in more detail in [4-38].

Discussion of the Benchmark results

From the parameters in Figure 4-20 a number of observations can be made. The latencies vary considerably, from a few to a few hundred microseconds. The same observation applies to overhead in pairs benchmark. The lowest overheads are not necessarily obtained by the tightly coupled shared memory systems, even though that might be expected, since for example Digital's Memory Channel reaches less than 4 µs and the Cray T3E 5 µs.

The bandwidths measured by sending a kilobyte packet vary between 3.7 and 59.0 MByte / s, although many of the systems can do better with a larger packet size. The latencies and overheads with MPI, at least with the current MPI versions, are large, typically 3–6 times larger than with lower level APIs. On the other hand, since MPI is available in multiple platforms, it provides portability; it can be debated whether the difference in performance justifies additional programming work.

The overlap results show the best overlap for the Meiko CS-2, as is explicable from its powerful per-node communication coprocessors. The AlphaServer 8400 occupies the other end of the spectrum as a typical SMP multiprocessor, with no overlap at all (and low absolute latencies at the same time). Some other observed overlaps must be attributed to artefacts from implementations of intermediate software layers, that stemmed from different authors, and are not so easily explained.

The large number of benchmark results has been obtained over a time span of more than a year. During that time a number of hardware and software upgrades took place, so that the results do not necessarily represent the most up-to-date situation. In addition, the maximum number of processors presented in some benchmark results depended on access to the systems.

Benchmark Summary

There is a large number of results for these benchmarks available. This makes it possible to widely compare the communication performance of different systems. Many of the latest-generation parallel technologies have been measured.

Several of the systems show communication overheads below 10 µs for small packets. Some of the systems additionally have proven a good scalability, which has been tested in some cases with up to 64 processors. Owing to the good scalability of some of the systems within the tested range it is predictable that scalability to hundreds of processors will either already now, or at least in the near future, also be quite efficient.

The communication parameter most typical for our application, pull-farm with four senders, is completed in some systems in around 60–90 µs. The best result for the push-farm, for the same number of processors and 1 kbyte data, is around 40 µs. We consider these numbers as promising for our trigger work.

4.5.2.4 Event supervision

The main Supervisor

A brief description of the supervisor used for the model-B and -C vertical slices is given below. A fuller description can be found in [4-39].

The supervisor for the model-B and -C demonstrators shared some common characteristics but there were important differences. In both programmes, the supervisor provided an interface to a LVL1 trigger emulator. The job of this interface was to distribute the LVL1 input, which consisted of RoI fragments, to supervisor processors in order that an RoI record could be constructed and sent to the LVL2 system for processing. The supervisor processor allocated a processor from a farm using a credit-based scheme in order to balance the processing load. The hardware implementation used commercial single-board processors (CES PowerPC-based RIO2 processors) and network interfaces, plus a number of custom items described below. In both cases only a minimum configuration (one supervisor processor) has been deployed to demonstrate the supervisor operation with the rest of the system.

The manner in which the RoI record was sent differed in the two architectures. In the demo B push architecture, the RoI records were sent via a custom S-link router to RoI distributor(s) that directed each record to selected ROB emulators. Processing proceeded through the LVL2 system to a global processor where a final trigger decision was made and a decision record forwarded to the supervisor through a network interface, either SCI [4-36] or DS-link [4-40] on a dedicated

network server. This server forwarded the decision back to the supervisor processor via VME shared memory. The demo B supervisor is shown schematically in Figure 4-21.

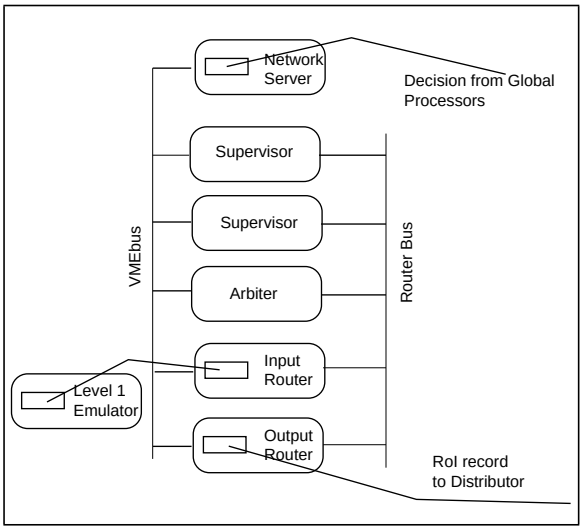


Figure 4-21 Schematic view of the supervisor for demonstrator B.

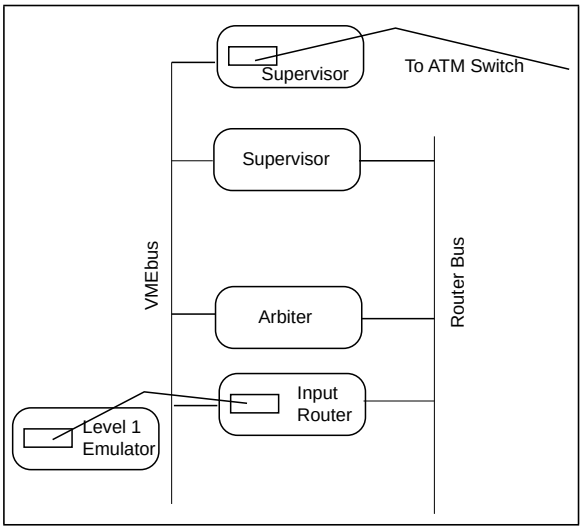


Figure 4-22 Schematic view of the supervisor for demonstrator C.

In the demo C pull architecture, the RoI record was sent to a destination node, which performed a function analogous to the global processor, except that it had to pull the data from the ROB emulators, called sources, make a trigger decision and return the decision record back to the supervisor. Both the RoI and decision record were handled in the supervisor with a single network interface card which was an ATM device. The demo C supervisor is shown in Figure 4-22.

The Supervisor components are built around a custom bus with two 32-bit channels. The custom components attached to this bus are the input router; an output router; the PCI mezzanine cards, to connect the commercial supervisor processor modules; and a central arbiter.

The input router buffers emulated LVL1 data from the S-link [4-3] interface in a 4 kword FIFO; chooses a supervisor processor based on the low-order 12 bits of the event ID; and sends the output to the router bus via another 4 kword FIFO for transmission to the supervisor processor. The output router receives RoI records from the supervisor processors via the output router bus; buffers these in a 4 kword FIFO and transmits them to the LVL2 RoI distributor via an S-link output port. The bandwidth of the input router was measured to be 20 Mbyte/s and that of the output router 14 Mbyte/s. A single PMC, with 4 kword FIFOs to buffer the input and output, connects both the input and output buses to each supervisor processor. A simple software API was developed, using an object-oriented approach, to operate this card from the supervisor processor. The arbiter controls the data allocation of the router buses, using a 10 MHz clock to cycle through a lookup table loaded with a list of grant addresses. In these tests a round-robin allocation scheme was used.

The operation of the input router and the distribution of RoI fragments to the supervisor processors was demonstrated using a LVL1 emulator. The emulator program, running in a CES PowerPC single-board computer with a simple PCI to S-link interface, read RoI fragment data from a file, constructed the record fragments and sent them over the S-link interface to the supervisor. This emulator ran at up to 22 kHz on a 100 MHz PowerPC board. For some tests use of the emulator was avoided by reading this file directly into the supervisor processor. Data files

consisting of 100–1000 events were generated with different numbers of RoIs and η-ϕ values to measure different topologies of the LVL2 network.

The Supervisor processor performed the following tasks:

- Read RoI fragments information from LVL1.
- When a complete event had been received, formed a RoIR record, selected a global or single-farm processor and recorded a time stamp for the event. The record was then sent to the output router in model B or the network interface in model C. (The time stamp could be used to implement a time-out mechanism to detect and isolate faulty processors.)
- Polled for the presence of a trigger decision and processed the decision if present. The time was recorded and a latency calculated and histogrammed. The destination processor was returned to the pool of available nodes. The trigger decision was stored in a structure used to build a record containing a group of trigger decisions.
- When a sufficient number of decision records had been received, the grouped decision record was sent to the ROB emulators, via the output router in model B and via a multicast from the network interface in model C.

An example of a set of measurements of the times required for each of these tasks is given in Table 4-9.

Table 4-9 Times measured on the supervisor.

Task	Execution time (μs)
Pack RoI fragments	8
Build RoI record	5
Send RoI record	9
Process global decision record	11
Send grouped decision record	12 (for a group of 16 events)

Tests with demonstrator B

Apart from the network server the operation of the supervisor for the Demo-B variants was identical. The supervisor processor was controlled through a series of environment variables. The measurements made by the supervisor were the latency, for which a running average as well as a histogram was kept, the throughput and the LVL2 pipeline occupancy, measured with a histogram. Typical distributions of the latency and pipeline occupancy are shown in Figure 4-23.

The supervisor processor could be configured to run either in fixed-frequency or free-running mode. A typical set of runs taken in free-running mode involved varying the number of events allowed to queue up per global processor. With this variable set at one, the average latency equals the time per event (the inverse of the rate) and gives a measure of the sum of the processing times of all elements in the LVL2 pipeline. As the number of events allowed to queue grew, the throughput increased to some maximum. The time to process an event at this point gave a measure of the time spent in the slowest element in the pipeline. Figure 4-24 shows data from a typical set of runs where the latency and time per event are plotted as a function of queue size.

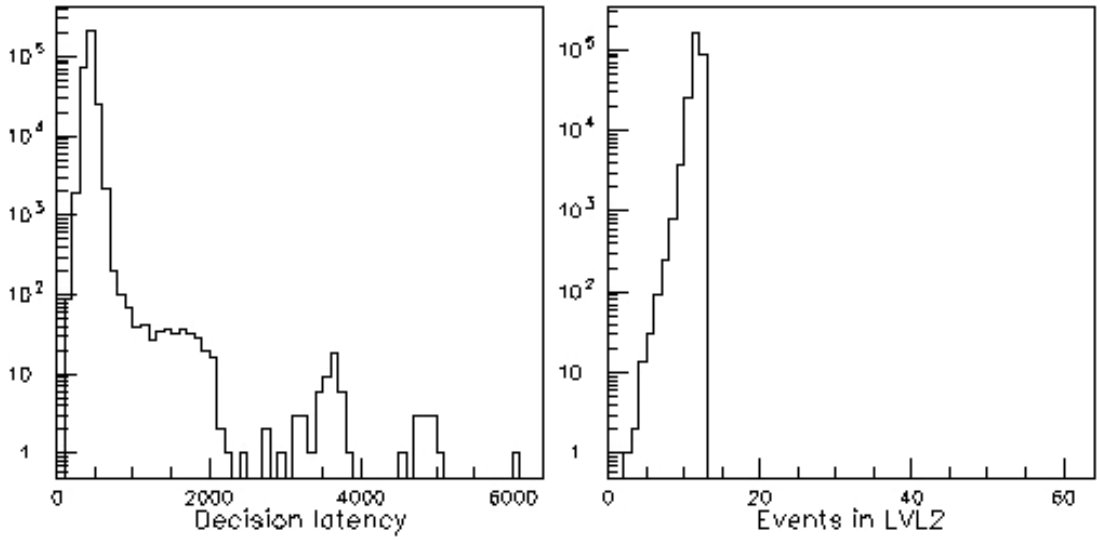


Figure 4-23 Supervisor tests with demonstrator B. a) Distribution of decision latency (μs), b) pipeline occupancy

The histogram of events queued in the LVL2 network was used to verify that the supervisor was not the slowest element in the system.

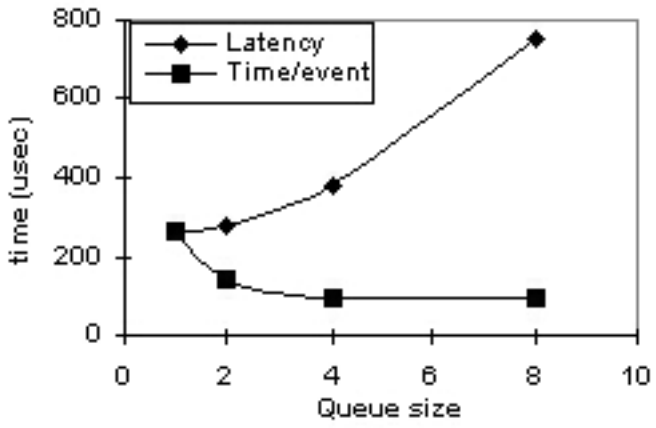


Figure 4-24 Latency and time per event vs. queue size.

Tests with demonstrator C

The supervisor software for Demo-C was adapted to operate within the framework used by all the nodes connected to the ATM switch. This framework handled the task of initializing the ATM device, managing a table of node addresses based on logical name and setting process priority. After initialization, it called the supervisor main task, implemented in a single thread, which then controlled the running.

Data for different points were taken in separate runs. The program was started under a unified run control system. The supervisor process ran as a single thread at elevated priority. The LVL1

emulator program was also started by the run control system. Additional configuration details were controlled via a configuration file.

During the runs, the decision latency was histogrammed. Runs were taken driving the supervisor from the LVL1 emulator at fixed frequencies so that the latency could be plotted as a function of event rate. A typical latency distribution is shown in Figure 4-25.

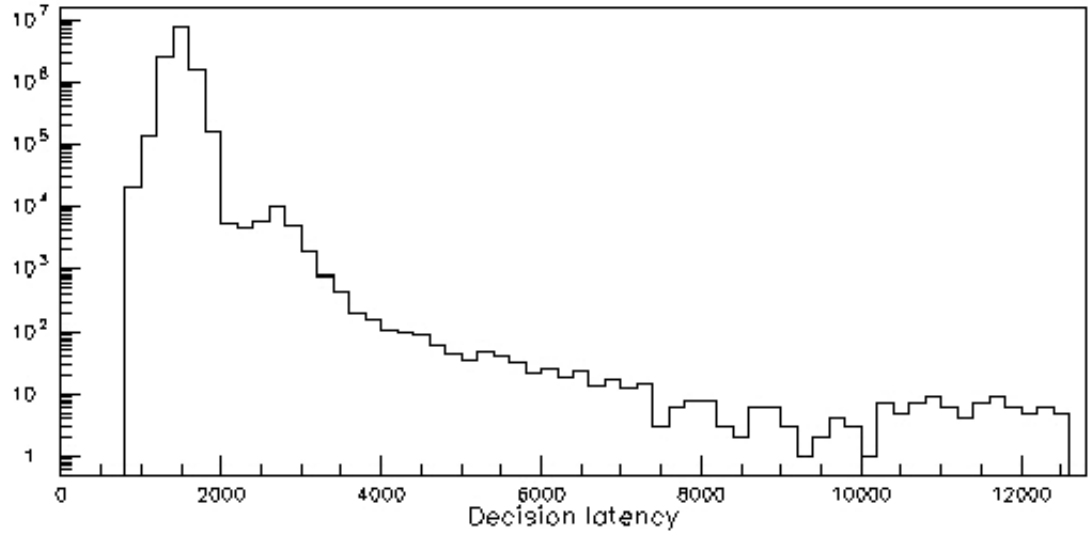


Figure 4-25 Supervisor tests with demonstrator C. Distribution of decision latency (μs)

Main supervisor conclusions

We have implemented a supervisor for the LVL2 demonstrator programme which supports vertical slices of architecture B and C. The supervisor provides an interface between a LVL1 emulator and a network of LVL2 trigger processors. Three different types of LVL2 network technologies were tested in the programme: DS-link, SCI and ATM networks. The supervisor balanced the processing load among the LVL2 processors and provided a measurement of the latency for reaching a LVL2 decision. Rates of up to 25 kHz have been obtained with events containing a single RoI. The model of the supervisor is scalable by adding additional commercial single-board processors to a custom bus that reads LVL1 input records and can also route RoI information to the LVL2 system.

Supervision of local-farm processors

Within the model-B architecture studies, the allocation of the local farm processors within each subdetector was handled by the RoI distributors (thus reducing the load on the main supervisor). This was done using a geographical allocation scheme based on the η value of the RoI with a look-up table to translate this value into a processor address. For this scheme to work with multiple crates of ROBs it is merely necessary for the RoI distributor within each of the ROB crates of a given subdetector to be loaded with the same look-up table. Thus even if an RoI spans more than two or more ROB crates they will still send the data for a given RoI to the same local processor, without requiring any direct communication between the ROB crates. This scheme avoids the need to maintain a list of free processors, but does require that the local proc-

essors can queue multiple events (and perhaps even RoIs). The average rate at which any given processor is used can be determined by the range of η (and ϕ) assigned to that processor.

Sequential steps

Independent of the discussions of the relative merits of parallel and sequential processing within an event, there are some tasks foreseen for the ATLAS LVL2 trigger which cannot be performed in a single step. For example, with B decays where LVL1 merely identifies a muon, so that a first step might be to confirm the muon and find further RoIs in the TRT, followed by a second step to use these additional RoIs require sequential steps. Architectures of the form of model C clearly allow such tasks to be included in a simple way. However, the situation is more complex for architectures, such as A and B, where the supervisor initiates the data transfer from the ROBs to the processors. In these cases options which might be considered are for the global processor to request the supervisor to initiate the second step; and for the global processor to control the second step itself. The first option places a significant extra burden on the supervisor; the second option avoids this problem, but requires complications to the control path back to the ROBs. No direct study was made of these issues within the demonstrator programme, but they were important considerations when drawing conclusions.

4.5.2.5 Event processing

Preprocessing versus feature extraction

Preprocessing in this context means a processing step executed at the ROB level in order to prepare data for the subsequent feature extraction. The feature extraction algorithm in contrast requires all of the data from an RoI to be collected together before it can proceed. The most common preprocessing tasks are zero-suppression, compression and format conversion. A convenient place to perform preprocessing would be either in the ROB itself or, better, at the output of a ROB crate. During the demonstrator programme the preprocessing for the TRT was investigated, an algorithm was developed and two implementations benchmarked. Preprocessing changes the TRT data format from pixel images with variable pixel length to coordinate lists with constant size. The data volume is significantly reduced up to an occupancy of 30%. Also, the new format is much easier to handle for standard CPUs, leading to shorter FEX processing times. Implementations were done on a 300 MHz Digital Alpha station and a 40 MHz microEnable FPGA processor. Preprocessing times for a single ROB in the barrel region are 60 to 165 μs with the Digital Alpha and 12 to 27 μs with the microEnable, for occupancies between 3% and 100%. These results show both the usefulness of a preprocessing step to reduce the data volume and the superiority of low-cost FPGA coprocessors over standard CPUs for such a task. Further investigations will explore the suitability of microEnable for a broader set of preprocessing algorithms.

Destination (or single-farm) processor

The tasks of the single-farm destination processors in architecture C are described in Section 4.5.2.1.

Figure 4-26a shows the time needed to handle one event on a single-processor platform, T_{dst}, versus the size of the event fragment sent by each of several sources, for a single-step algorithm with zero processing time.

On a 200 MHz PentiumPro, the code executes in:

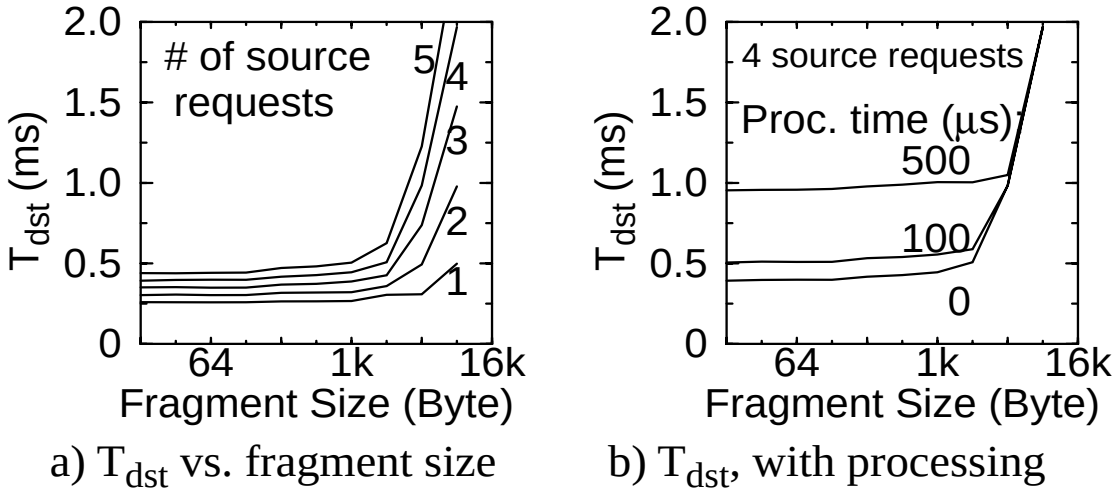


Figure 4-26 Performance of the destination processors versus event fragment size (a) without processing and (b) with processing.

$T_{dst} = 215\ \mu s + 45\ \mu s \times number\_of\_source\_requests$

For blocks larger than ~2 kbyte the data transfer time is dominant; for smaller data blocks, processor overheads are dominant. The maximum event rate that can be sustained by a single destination is $F_{dst} = 1/T_{dst}$. For example, a destination can accept short events with data spread across four sources at rates up to ~2.5 kHz.

When processing is introduced, the amount of time to handle an event is increased (Figure 4-26b). The destination processor reformats the received data, then executes the selection algorithm. Measurements show that the data received is copied at a speed of 30 Mbyte/s. When the processing time is dominant, the maximum event rate is given by $F_{dst} = 1/(T_{dst} + T_{algo})$. When the data transfer time is dominant, the speed of the link determines the maximum event rate: $F_{dst} = 1 / T_{transfer}$.

A correct match of the CPU power of the destination and the speed of the link should be made to avoid a waste of network bandwidth or processing resources. Assuming a typical feature-extraction algorithm requiring ~100 µs to process ~4 kbyte of data spread over four sources and an overhead derived of 400 µs from the equation given above, the corresponding data transfer rate is 80 Mbit/s, well adapted to a 155 Mbit/s link.

The number of destination processors can be adjusted to cope with the rate of incoming events. The first step of event selection in ATLAS is the confirmation of primary RoI (~ one such RoI per event) using a feature-extraction algorithm such as the one mentioned above. The model described here predicts that today's processors should be able to process such events at ~1.5 kHz. The computing power needed for this first step of event selection would be ~66 processors for a 100 kHz LVL1 trigger rate.

Local (FEX) and global processors

With the DS implementation of the local–global vertical slice, essentially the same code ran in the local and global processors. The latency for the local processors was measured to be

$T(FEX) = Max(125\ \mu s, 80\ \mu s + (N_{ROB} - 1) \times 45\ \mu s)$

The estimate for the global processor is slightly smaller at

$T(\text{Global}) = 80\ \mu s + (N_{FEX} - 1) \times 40\ \mu s$

It should be noted that both of these times are purely for protocol handling and fragment building, and that all of this code runs in the transputer processor. Processing of the algorithms would be done in the Alpha processor which was unused in these tests (except for some trials where it performed some wait code to simulate algorithm processing).

The SCI implementation of the local-global vertical slice also used similar code in the local and global processors, again with no algorithm processing. Here, however, the heterogeneous nature of the processors caused some difficulties in the interpretation of results. For example, although all the processors used the same Alpha chip (21066), there are two different designs of motherboard with small but significant differences. This was further complicated by two different clock speeds (166 and 233 MHz) in one of the board types, and in some measurements 10–20% differences which appear to be due to revision levels of otherwise identical boards. However, in spite of these differences, measurements were made in the processors of the CPU cycles used in both the local and global processors (for the transparent-mode SCI tests). Note that this execution time does not include (although it may overlap with) the time for the data to be received by the SCI, which in this mode is handled by the hardware in the receiving node. The global processing took 30–40 μs (with 233 MHz processors) and no difference is seen in this time for one or two FEXs sending to the global. The feature-extraction processing time was more variable, but was typically in the range 40–70 μs (with 166 MHz processors). Although the code to be executed is independent of the size of data records coming from the ROBs, it was observed that the execution time increased as the data rate being received increased. One possible explanation of this is the tight coupling between the I/O and memory systems on the processor board. In the case of the feature-extraction processing, the execution time does appear to increase by some 10 μs for each extra ROB.

Use of locality

Within the LVL2 system, moving data is in general a greater problem than the processing requirements themselves. Thus any processing which reduces the volume of data should wherever reasonable be done close to the original location of the data. This is a principal argument for preprocessing to be done within the ROB complex. Feature extraction, however, poses additional problems since it is necessary to gather all of the data within the RoI from a number of ROBs. This gathering of data requires connectivity to all of the ROBs of the subdetector. Within the model-A and model-B architectures this is handled by the RoIC and the local network, respectively. Within the model-C architecture the RSI gathers data from many ROBs, thus reducing the number of messages (but not the total bandwidth) required to traverse the switch. Although the use of locality was not studied directly within the demonstrator programme, it remains an important concept and will need to be studied further for whatever architecture is finally chosen.

Sub-farms

Sub-farms where a single processor with a network connection acts as switch farm interface (L2-SFI) for a number of interconnected processors have been investigated [4-41]. The investigations have mainly considered clusters of powerful commercial processors. The improvement in performance which can be achieved with a sub-farm compared to a single processor is dependent on the ratio (execution time)/(communication latency) .

Thus when the communication latency is negligible compared to the execution time, the L2-SFI can support many processors within the sub-farm. In contrast, if the communication latency is

larger than the execution time the sub-farm brings no advantage; indeed the additional internal latency within the farm may lead to a drop in overall performance.

For feature extraction the algorithm can be divided into three main phases: fragment collection, data preparation, and feature extraction itself. The fragment collection deals with the task of collecting ROB packets that are contributing to a given RoI (each packet contains a fragment of the RoI). The other two parts are concerned with data processing. The most natural idea for a FEX implementation using the sub-farm concept is to split the fragment-collection phase from the other phases. The L2-SFI collects the ROB packets and assigns one of the sub-farm's nodes to process the set of fragments for a given RoI.

Using the calorimeter FEX algorithm as an example, this problem was studied on a Digital Alpha MEMORY CHANNEL cluster consisting of one AlphaServer 4000, running at 300 MHz, and four AlphaStation 200s, running at 166 MHz. The five machines communicate with each other via a MEMORY CHANNEL hub. The server plays the role of the L2-SFI, and runs the fake fragment collector. However, in this configuration the communication latency within the cluster was greater than the execution times in the Alpha processors, thus making the distributed processing perform slower than a single processor. This was in spite of the high performance of MEMORY CHANNEL. Other parallel-processing strategies have been tried as well, namely the execution of different parts of the algorithm in different nodes, but the performance was still not better than one Alpha running in standalone mode with no packets exchanged.

If shared memory machines were used, where no explicit data transfer is required between the processors, the internal latency in the cluster would be avoided and the balance of communication to processing would be more favourable. Similarly, if the communication were driven by dedicated I/O processors rather than the processors themselves the situation would also improve.

However, the general conclusion must be that for the sub-farm concept to be of benefit requires problems where the ratio of the execution time to the communication latency is much greater than those of the cases investigated. One example might be the TRT full scan.

FPGAs as preprocessors, processors and/or coprocessors

Introduction to FPGA processors

In standard processors – or computers – an algorithm is implemented by having a single CPU executing a sequence of single programming threads, each one at a time. In contrast an FPGA processor is the implementation of an algorithm directly in hardware which leads to an enormous increase in speed. This type of implementation is a well-known approach for tasks near LVL11 where dedicated application-specific hardware is widely used. However the hardware of the FPGA processor itself – all the chips on the board – remains the same for all different applications and the *effective hardware* is determined by the configuration loaded into the FPGA chips. This is why we use the wording FPGA *processor* to indicate that it is a programmable object. And in fact an FPGA processor is a device which combines the speed of hardware with the flexibility of software.

Currently most of the programming is done using a more or less hardware-oriented programming language like VHDL, but efforts are being made worldwide to develop more abstract design and test methods. Although FPGA processors are, for the time being, not as easy to program as standard computers are, the superior level of performance justifies their use. This is especially true for many trigger related algorithms which employ high implicit parallelism, unaligned bit manipulations, heavy use of look-up tables and very high I/O rates. Speed-up rates

in the range of 100 - 1000 compared to RISC workstations have been demonstrated in the past [4-42].

Two different FPGA processors have been developed and successfully used in the ATLAS LVL2 demonstrator programme, a large one (Enable++) and a small one (microEnable).

FPGA processors in demonstrator A

FPGA processors have been used in the Demo-A programme to speed up several critical tasks of the ATLAS LVL2 trigger in the following areas:

- full subdetector feature extraction (FEX), the main task under investigation by Demo-A;
- partial feature extraction, namely the initial track finding for the full-scan TRT, where the FPGA processor operates as a coprocessor to the main event processor;
- preprocessing of raw event data, done at the ROB level previous to data transmission;
- RoI collection, as described in Section 4.5.2.1, subsection RoI collection.

Also the ROBin-work has been heavily influenced by the use of FPGAs and studies of the suitability of commercial FPGA processors for this task will continue to be studied.

The FPGA-processor systems used in the Demo-A programme are

- Enable++, used for full and partial FEX,
- the RoIC system (see above), and
- MicroEnable, used for preprocessing and S-link source and destination driver.

FPGA feature extraction

According to the architecture-A trigger concept extremely fast data-driven FPGA processors are used for feature extraction on a per-subdetector basis. These processors are able to handle events synchronously with the expected 100 kHz event rate, allowing a very compact FEX system of approximately 15 FEX processors in total followed by a small farm of 50 processors performing the global decision.

An alternative approach was proposed and elaborated as a paper model [4-43] in 1997 which integrates FPGA processors and standard CPUs in a hybrid system: the LVL2-prescaler. The *initial track finding* for the full-scan TRT as a partial feature-extraction task can be viewed as a subset of this proposed hybrid system.

During the Demo-A programme measurements of the feature-extraction performance of Enable++ have been done, both for the RoI-guided TRT (full FEX) and the full-scan TRT (partial FEX). A summary of the results is presented in Table 4-10 and Table 4-11.

Integration of FPGA processors into a farm of standard CPUs

During the last week of November 1997 a short but successful integration test of the two different demonstrator projects, A and C, took place. For this first step the following goals were defined:

- Attach the Enable++ system to the ATM network via a special node (PC with Windows NT).

Table 4-10 Results from demo-A TRT FEX measurements

Item	Measured	Comments
Parameters		
Input data bandwidth (S-link)	72 Mbyte / s	Max. S-link data rate ~ 92 Mbyte / s @ < 2 kB per packet
Number of event data-sets	250	No pile-up
Number of valid tracks per event	1	
Number of found tracks before	7.2	Initial track finding only; further reduction to 1.x
Average event size	450 entries	4 bytes / entry on S-link, 2 bytes internally
RoI-scan results		
Number of RoI search patterns	240	30 ϕ, 8 p_T
RoI occupancy	30%	
Event rate	38 kHz	Resource usage ~ 30% of single Enable++ board
Event latency	5.3 µs	Plus data transmission time
Possible parallelism at 100% board usage	3	Leads to > 110 kHz event rate per board
Full-scan results		
Total number of full-scan patterns	~ 80000	
Number of full-scan pseudo-RoIs	32	
Full-scan search patterns per pseudo-RoI	2400	30 ϕ, 80 p_T (down to 0.5 GeV)
Number of parallel patterns on Enable++	448	Max. is 864, only reduced capacity available in test
Occupancy	1%	
Number of passes for 2400 patterns	3	six passes needed in test with 448 patterns / pass
Event rate	6 kHz	

Table 4-11 A comparison of recent results for the high-luminosity TRT trigger

Processor type	Execution Time	Comment
100 MHz HP workstation, no I / O processing	2000 ms	
400 MHz Digital Alpha workstation, no I / O processing	400 ms	
50 MHz Enable++ FPGA processor, with I / O processing	3.5 ms	Optimized design, no I / O bandwidth limitation
20 MHz Enable++ FPGA processor, with I / O processing	26 ms	Unoptimized test design, with I / O bandwidth limitation

- Adapt the DATASOURCE software module to this special node for coprocessing operation (full- scan TRT). The new term for this node will be COP.
- Emulate the RoIC system on the COP feeding Enable++ via S-link.
- Receive results from Enable++ via S-link on COP.
- Run the complete Demo-C software framework including COP and Enable++.

The preliminary results are as follows:

- A successful integration of the two demonstrators at the protocol level was performed.
- The software framework provided by Demo-C could be ported to the Demo-A system with reasonable ease.
- A simple API with the functions *Open*, *Close*, *Request*, *Response* is sufficient to communicate with the microEnable boards serving the S-links.
- The complete demo-C system, including supervisor emulation and monitoring, was able to run stable up to 1 kHz message rate with an increased latency from 600 μs to 800 μs caused by the COP. At 10 kHz message rate many packets were lost.

FPGA processor summary

During the demonstrator programme the stable operation of all Demo-A system components in a integrated test set-up could be established and the main goals defined at the start of the programme could be achieved:

- FEX track reconstruction quality is comparable to offline code (xrecon) except for p_T resolution, which is worse. This is a property of the pattern-matching algorithm.
- A speed-up factor of 100 compared to high-end RISC workstations is still available, even though Enable++ is not up-to-date technology.
- Message rates of 100 kHz @ 1 kbyte size can easily be handled with the current FPGA systems.
- The event rates achievable with the FPGA processors fulfil the ATLAS requirements.
- The successful first integrations step of demonstrators A and C shows that hybrid systems are feasible and advantageous.
- The PCI-based FPGA-processor microEnable is perfectly suited for preprocessing tasks and buffer management.

FPGA processors are promising candidates for the most demanding trigger tasks in ATLAS LVL2.

Other processors (e.g. DSPs)

Most of the processors studied within the demonstrator programme were either RISC/CISC or FPGAs. However, other types of processors have been considered, particularly DSPs and I/O coprocessors. DSPs have previously been foreseen for use in preprocessing and feature extraction. However the relative developments in DSPs and general-purpose processors now make DSPs less attractive for these applications and their main interest remains as I/O processors.

Although I/O coprocessors were not an item explicitly studied within the demonstrator programme, a potential use was amply demonstrated in the DS-link implementation of the mod-

el-B architecture. Here the TransAlpha modules used for the FEX and global processors contain a transputer to handle the I/O and a Digital Alpha chip for the algorithm processing. In this application the transputer handled not only the I/O, but also the fragment building so that in the FEX processors for example data for a complete RoI was assembled by the transputer, and the Alpha processor was presented with this complete data structure. Thus the Alpha processor not only avoided the computation required to handle the I/O itself, but also the considerable overheads associated with the many data packets received.

4.5.3 Modelling and emulation

4.5.3.1 Paper models

With the wide variety of choices facing the demonstrator programme in terms of architectures, technologies and event-selection strategies, a fast method of assessing the impact of these options is needed.

A paper model is a simple calculation of the rates, loads, occupancies and latency of the trigger system. Average numbers are used to represent the input events and data. Hardware is modelled as generically as possible by simple parameterizations. The results give a preliminary indication of potential bottlenecks and hardware requirements, and provide input to modelling and emulation. Following an initial model for architecture C [4-44], the calculations have now been implemented in two independent spreadsheets [4-45]with a common set of inputs and assumptions covering architectures B and C [4-46].

Description of inputs and assumptions for modelling

The LVL2 trigger only accesses the data for RoIs which are built from the regions of the detector where LVL1 triggers occur. The LVL1 trigger menu is therefore used as a basis for the physics input to the model. A list of LVL1 trigger items has been made and the exclusive rate for each item was estimated [4-18]. The size of RoIs and the mapping of the detector to the ROBs was then used to get the data request rates for the ROBs. Tables 4-12 and 4-13 show these rates for various strategies.

Table 4-12 ROB request rates at low luminosity for various trigger strategies.

	Trig. RoIs only	All RoIs parallel	All RoIs sequential	All RoIs sequ. EtMiss + bjet
ECAL	0.99	3.67	1.87	4.17
HCAL	2.52	8.10	4.30	6.60
MUON	0.22	0.22	0.22	0.22
SCAN	4.10	4.13	4.13	4.13
TRT	0.43	0.50	0.23	0.23
SCT	1.69	1.78	1.46	1.59

The trigger technology is modelled as follows. Links are assumed to have a bandwidth of 10 Mbyte/s. The elapsed time for a data transfer is T^0 + link BandWidth / message size. T^0 is tak-

Table 4-13 ROB request rates at high luminosity for various trigger strategies.

	Trig. RoIs only	All RoIs parallel	All RoIs sequential	All RoIs sequ. EtMiss + bjet
ECAL	0.87	2.49	1.22	2.22
HCAL	2.50	5.93	3.18	4.18
MUON	0.24	0.24	0.24	0.24
TRT	0.64	0.71	0.21	0.21
SCT	0.64	0.72	0.16	0.22

en to be 100 μs. The time for which a processor is occupied during the sending or receipt of a message is $0.5 \times T^0$, i.e. 50 μs.

Feature-extraction algorithm times are normalized to 500 MIPS processors. Typically muon and calorimeter feature extraction (including the missing-E_T recalculation) are assumed to take ~ 100 μs, tracking ~ 500 μs, the TRT scan 50 ms and b-jet tag 250 ms. Process models were developed for the other processing nodes of the trigger systems.

There are 1462 ROBs, with 4 ROBs to an RSI for architecture C. The average event data sizes are 1.3 kbyte in the ECAL, 0.89 kbyte in the HCAL and 0.58 kbyte in the muon detectors. Data from tracking detectors is zero-suppressed so the event size depends on detector occupancy. It varies from ~ 0.3 kbyte at low luminosity to ~ 1.0 kbyte at high luminosity.

The following event-selection strategies were used for high luminosity:

1. only trigger RoIs, event decision taken with all features
2. all (trigger + non-trigger) RoIs, event decision taken with all features
3. all RoIs, with sequential selection
4. all RoIs, sequential, including complex algorithms (missing-E_T and b-jet tags).

The low-luminosity strategies are the same except that there is a TRT scan which is always done sequentially after muon confirmation, which is expected to reduce the input rate of the TRT scan by half. Unless otherwise stated the results below for architecture B assume the first of these strategies, whereas those for architecture C assume the third strategy.

Results of paper models

Results for trigger strategies 1–3 will be presented here. Results for trigger strategy 4, including the LVL2 b-jet vertex tag, will be shown in the following subsection. For strategies 1–3, the model shows that the overall system requires 500–700 processors at low luminosity and 100–250 at high luminosity. Over half of the processors at low luminosity are used for the TRT scan. The total bandwidth inside LVL2 is 1300 – 1800 Mbyte / s at low luminosity and 600 – 1300 Mbyte / s at high luminosity. This is dominated by data sent from the readout buffers. At low luminosity the bandwidth is higher because of the TRT scan (700 Mbyte / s). At both luminosities, calorimeter data accounts for 500–700 Mbyte / s of the total.

The I / O overhead for processor communication is found to be a critical parameter. At the default value of 50 μs, half of the processing power of LVL2 is used for I / O. Figure 4-27 shows that

the number of processors needed at LVL2 depends steeply on this parameter. This discovery also implies that the network interface is a critical technology for the trigger.

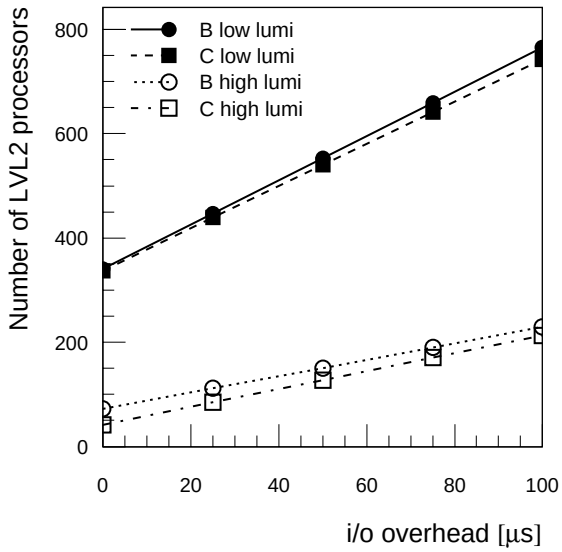


Figure 4-27 Number of LVL2 processors required for trigger strategies 1 and 3 as a function of I/O overhead, assuming 100% CPU utilization. The nominal overhead is 50 μs.

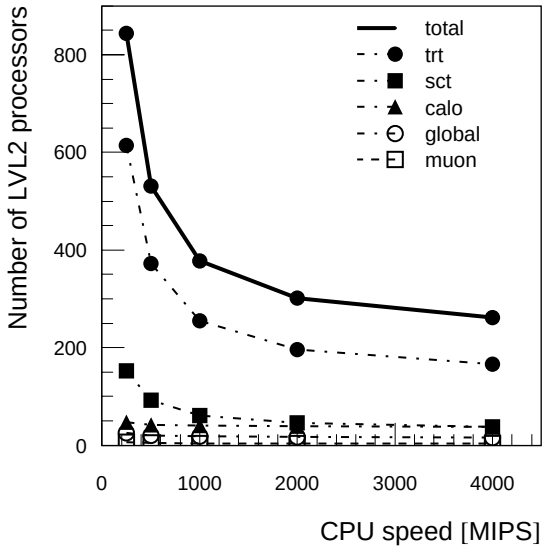


Figure 4-28 Number of LVL2 processors required as a function of CPU speed for sequential processing.

With the configurations of ROBs and processors used in these models there is generally no requirement for link speeds of more than 10 Mbyte/s. The one exception was the HCAL ROBs in architecture C where up to 20 Mbyte/s was required, but this can easily be avoided by reducing the number of these ROBs feeding into each network link (see below). Computational-intensive algorithms, in particular the TRT scan, mean that there is some dependency of the required number of processors on CPU speed (Figure 4-28). When algorithm times are decreased by increasing the processor speed from 500 MIPS to 1000 MIPS, 200 fewer processors are needed.

ROB occupancy is the product of the request frequency (Tables 4-12 and 4-13) and CPU time per request. The ROB has little processing to do in addition to the I/O overhead, so the total time per request is 121 μs. With these assumptions the loading on the HCAL ROBs, and the TRT ROBs at low luminosity, are particularly high. The paper models predict bottlenecks at these points, the nature of which has been investigated by modelling and emulation studies (see Section 4.5.3).

The concentrating function of the RSI and L2-SFI reduces the number of ports needed on the switch, which will affect the cost. But the down side is the three to four times higher rate and bandwidth in the RSIs compared with the ROBs. This could produce severe bottlenecks, especially in the HCAL system. Fortunately, the number of HCAL ROBs is very small, so this particular bottleneck can be resolved by coupling those ROBs directly to the switching network.

A sequential selection strategy reduces the request rates to the ROBs. It is especially beneficial to reduce that rate at which secondary RoIs are requested. When data requests for sequential selection are routed through the supervisor, the rate of messages it receives from the global processors is doubled. As the supervisor is already a bottleneck in the system it may be better to route requests through the network.

Parallel processing of RoIs (as is possible with architecture B) reduces the average decision latency.

Further Architecture-C paper modelling

Additional studies of architecture C have been made using a spreadsheet model [4-47]. These assume that:

- for each event, the feature extraction and global algorithms are processed in sequence in a single processor,
- these processors request RoI data and execute event selection algorithms sequentially,
- the ROB and farm network interfaces act as concentrators / dispatchers for data transfers.

The [4-18] low-luminosity trigger menu was used for this study. It includes b-jet tags and a missing-E_T trigger. The b-jet tagging algorithm is assumed to take 10 ms in 500 MIPS processors, thanks to a recent pixel version of the track finding part. Some parameters from DAQ–NO–70 have been updated to account for improvements in the process model. Inclusive RoI rates were used at each processing and selection step to find the RoI rates in each of the subdetectors. The TRT full scan was performed for every confirmed trigger muon. After the TRT full scan, an average of five track RoIs are selected for further study by the SCT. The number of ROBs per RSI is set to one for the hadron calorimeter to avoid a bottleneck. Overhead times are set to 10 µs for I / O processing and 40 µs for context switching. The context switching is applied to all destination nodes except the ROBs and RSIs. Data merging CPU time was considered only in the L2-SFIs, not in the RSIs, since DMA data transfer is foreseen between ROBs and RSI. The transfer through the same network of all event data to the EF processors was included as an option. The RoI rates shown in Table 4-14 were obtained for a 38.7 kHz LVL1 trigger rate.

Table 4-14 RoI rates in the architecture C paper model, full trigger menu, low luminosity, level-1 rate 38.7 kHz.

RoI rates (kHz)	Muon system	ECAL	HCAL	SCT	TRT	Full scan
Muon RoI	8.4	4.4	4.4	6.3	6.3	
Electromagnetic RoI		21.3	21.3	3.8	3.8	
Jet RoI		24.0	24.0	3.2[a]		
Special RoIs (missing-E_T, TRT full scan)		0.83	0.83	20.0		4.0

a. includes b-jet vertex tag algorithm

Occupation levels were calculated for each of the system components. Encouraging results were found with a model based on a 1024-node network and 15 Mbyte / s links. No difficulties were found for operation at the nominal 38.7 kHz LVL1 rate. Even when the trigger rate was scaled to the maximum possible LVL1 rate of 100 kHz, all the system components had occupancies below 100%, as shown in Table 4-15.

Very recently, new paper studies have been carried out to evaluate the effect of technological evolution on our system performance. With 10 µs context switching time, 2000 MIPS LVL2 processors, 60 Mbyte / s network links, 80 Mbyte / s RoI fragment merging speed and 2 ROBs per RSI for the ECAL, occupation levels fall below 60% even including event building traffic for all sys-

Table 4-15 Occupancy results from the architecture-C paper model, scaled to a 100 kHz LVL1 rate.

	Muon MDT	**Muon trigger**	**ECAL**	**HCAL**	**SCT**	**TRT**	**TRT scan**
RoI request rate per ROB (kHz)	0.37	2.41	5.44	10.11	1.56	0.53	10.34
RoI fragment volume out per ROB (Mbyte/s)	0.23	0.32	1.67	8.04	0.58	0.16	3.22
Average ROB occupancy (%)	13.7	22.3	70.0	82.0	18.7	14.3	43.6
ROB occupancy including EF data transfer	16.8	25.4	73.2	85.1	21.8	17.5	43.6
Number of ROBs per RSI	3	4	4	1	4	2	2
RoI fragment rate out per RSI (kHz)	0.48	4.59	7.99	10.11	5.81	0.57	10.34
RoI fragment volume per RSI (Mbyte/s)	0.70	1.27	6.68	8.04	2.33	0.33	6.45
Average RSI occupancy (%)	7.2	33.5	64.5	42.5	29.1	6.2	62.0
RSI occupancy including EF data transfer	11.2	38.5	69.5	44.5	34.1	9.2	62.0
Network occupancy by LVL2 data, per RSI port (%)	4.6	8.5	44.5	53.6	15.5	2.2	43.0
Network occupancy including EF data (%)	16.6	11.2	79.2	59.6	24.6	48.9	

EF traffic included	No	Yes
Total data rate (Mbyte/s)	3043	3995
Number of L2-SFIs available on a 1024-port switch	438	438
Data rate per L2-SFI (Mbyte/s)	6.95	9.12
Network occupancy per L2-SFI port (%)	46.3	60.8
Average L2-SFI occupancy (%)	84.2	91.8
Minimum number of LVL2 processors required	777	
Number of LVL2 processors per L2-SFI	2	
LVL2 processor occupancy (%)	88.7	

tem components but the TRT RSIs (full scan). The TRT scan would probably better be handled by dedicated FPGA co-processors.

4.5.3.2 Computer simulation

Introduction

Computer modelling, i.e. system simulation, takes into account and provides insight into contention for resources and the resulting queuing. The latency (the time required to produce an accept or reject decision) distribution for the system modelled can be determined, as well as distributions of the filling degree of queues (i.e. of the required buffer sizes) and of the utilization of resources such as communication-link bandwidth and processor capacity. It should be noted that the computer model is essential for building up confidence that the whole system will work before constructing it. The results of the paper model provide a cross-check for the results of the computer model, as the average resource utilization in both models should be the same, provided that the same parameters and models are used, that queues in the simulated system on average have sufficient storage capacity and that the average utilization of the available resources (link bandwidth, processing power) is less than 100%.

Goals

The following goals have been formulated:

- input to decision making,
- acquisition of knowledge about the factors controlling system behaviour and resource requirements,
- understanding relevant technologies with respect to behaviour and resource requirements if applied in the ATLAS environment,
- provision of help in understanding results obtained with test set-ups, with as a side-effect ‘confidence building’ with respect to modelling results.

It should be noted that simulation is necessary to acquire a good understanding of the factors controlling the behaviour of the LVL2 trigger system. This is due to the large number of processors in combination with the networks and switches providing the communication facilities required and the use of RoIs for control of the dataflow, possibly in combination with sequential processing.

Method

The type of simulation used is called discrete event simulation. Events (not to be confused with events due to particle interactions, observed in an experiment and in this section referred to as ‘physics events’) occur at certain simulation times. For each event the response of the simulated system is determined. The response can consist of the generation of new events at the same simulation time as the original event occurred or at future simulation times.

In order for the results of simulation to make sense many input parameters need to be set to realistic values. The operation of the system also depends on the type of events selected by the LVL1 trigger and the number and types of RoIs associated with them. Two approaches are possible here. In the first the relevant information is extracted from simulated events and used as input for the simulation on an event-by-event basis. Alternatively an estimate can be made of the number of each type of event selected by the LVL1 trigger, of the number of RoIs associated with each type and of the distribution of and correlation between the RoI positions. With this information the relevant properties of the events can be generated during running of the simulation program. This approach in principle is less accurate than the first, but provides a good first-order estimate without requiring access to large samples of Monte Carlo events.

For simulation of the ATLAS LVL2 trigger the SIMDAQ program has been developed. Its first implementation has been in MODSIM-II. This version [4-48], apart from the code for the simulation of the SCI and ATM technologies, has been translated and extended in C++ [4-49]. The emphasis until now has been on the organization of the simulation program, on simulating 'generic' models and on the correlation with paper models. Implementation of models of the various network technologies of interest has been partially done in C++.

The C++ program makes use of a platform-independent graphical user interface. UNIX, Windows95 / Windows NT and MacOS versions are available.

The model to be simulated is specified, at the level of processors and switches and their connections with a configuration file. The details at lower level can be controlled by parameters that can be specified in the configuration file.

Present status

The current version of the program supports efficient simulation of the full model-B architecture as used for the paper model and as outlined in Figure 4-29. It has been found that results of the simulation program for processor and communication link utilization are in excellent agreement with the paper-model results. Results with respect to the decision latency are discussed below.

Since last summer the workforce has expanded from a single person to a core group with participants from several institutes.

Recently some work has also started with Ptolemy [4-50], a general-purpose simulation tool that also supports discrete event simulation. This tool seems to be well suited for studying relatively small systems. This type of studies may provide valuable input for large-scale simulations.

Some results

A detailed study has been conducted on the model-B LVL2 trigger system (see Figure 4-29), using the models and parameters documented in [4-46]. However, for the switches and network technology no models are provided in this section. For the study mentioned the switches are crossbar switches with unlimited buffering on input and output links and with arbitration for access to the output buffers. This arbitration can be switched off to study the effect of it. Aggregate switches can be built from these switches. The links transport data with a fixed speed of 10 Mbyte / s.

For trigger strategy 2 — all RoIs, event decision taken with all features — (see Section 4.5.3.1), at high luminosity, a surprising result was found for the latency distribution, using the nominal LVL1 trigger accept rate of 40 kHz. With this trigger menu the utilization of the processing resources of the hadron calorimeter ROBs would be more than 100% when the parameter values of [4-46] are used. Therefore the task switching time was reduced from 50 to 35 μs in order to obtain utilization below 100%. All other parameter values were set as specified in [4-46], resulting in processor utilizations ranging from 28% to 42%. For a bandwidth of 10 Mbyte / s the link utilizations are all below 30% except for the hadron calorimeter ROB output link with an average utilisation of 59%. All processing times were fixed, as well as the length of the interval between consecutive LVL1 trigger accepts. The switches were modelled as single crossbar switches of the type described earlier in this section. The internal bandwidth was set to 50 Mbyte / s per connection .

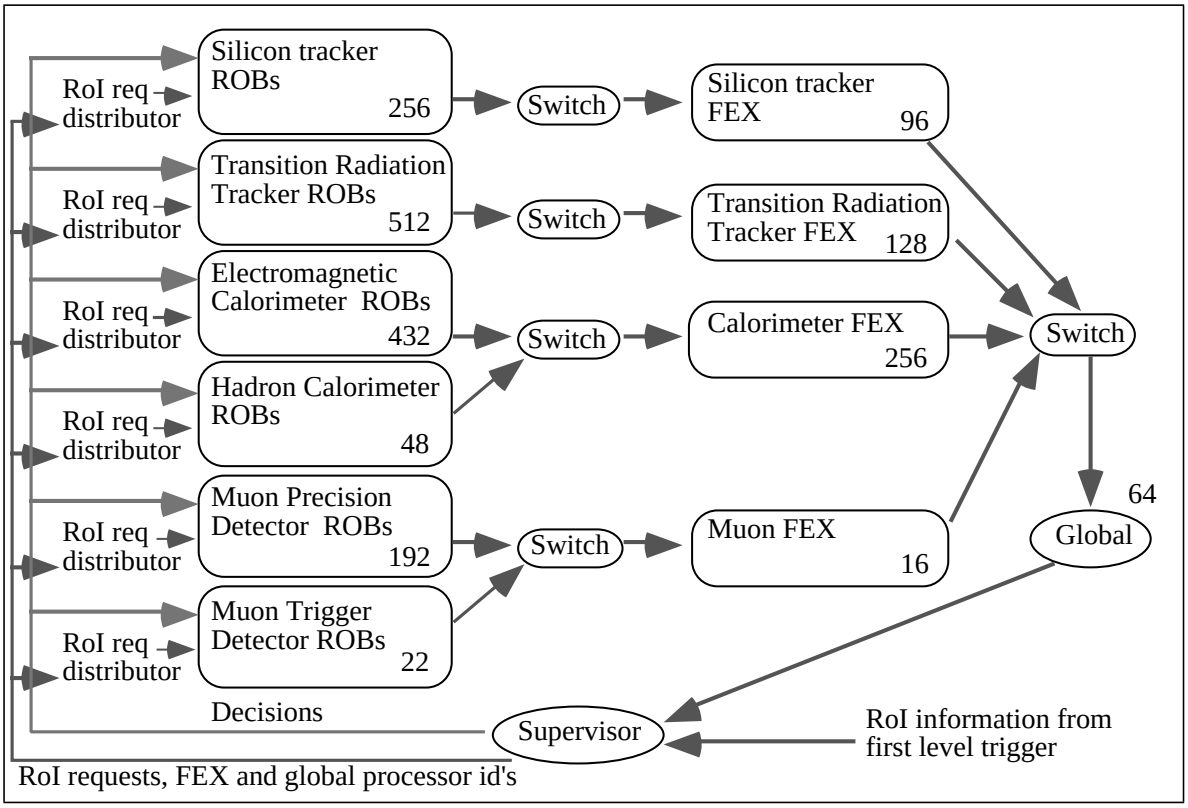


Figure 4-29 The full architecture-b system, as defined for the paper model. The numbers indicate the number of ROBs or processors, FEX stands for feature extraction. The switches are either single switches or aggregate switches built from two layers of switches. The 100 Mbyte/s data links (one per ROB) transporting the raw data to the ROBs are not shown.

Figure 4-30 shows the latency distribution obtained, with a number of peaks spaced at equal separation. These peaks are due to congestion in the switch varying periodically in time - the period being the time for the round robin allocation to loop over the feature extraction processors. The figure also shows how the latency distribution broadens with time as the system becomes increasingly unstable. The congestion in the switch is due to head-of-line blocking, where data in the input buffer of a switch port (in this case those receiving data from the hadron calorimeter) is blocked by earlier data in the same buffer waiting to be transferred to a busy output buffer. Further studies have shown ways in which this behaviour can be overcome, but the important point is that this example illustrates very clearly the additional power of the computer simulations, compared to the paper models. Here the paper model would have indicated that all parts of the system were working within their capacity, but the computer simulations which include statistical variations in the process show very clearly that the system is in fact unstable.

4.5.3.3 Emulation

The motivation behind the emulation programme was to learn about the performance of large networks when transporting ATLAS LVL2 traffic. The MACRAME 1024-node switching testbed [4-51], built with DS-links and switches [4-40] was used for this purpose.

The Macrame test bed is a modular set up of four components: traffic-generating nodes, timing nodes, C104 [4-52] 32x32 crossbar switches, and DS-links operating at 100 Mbits / s. The configu-

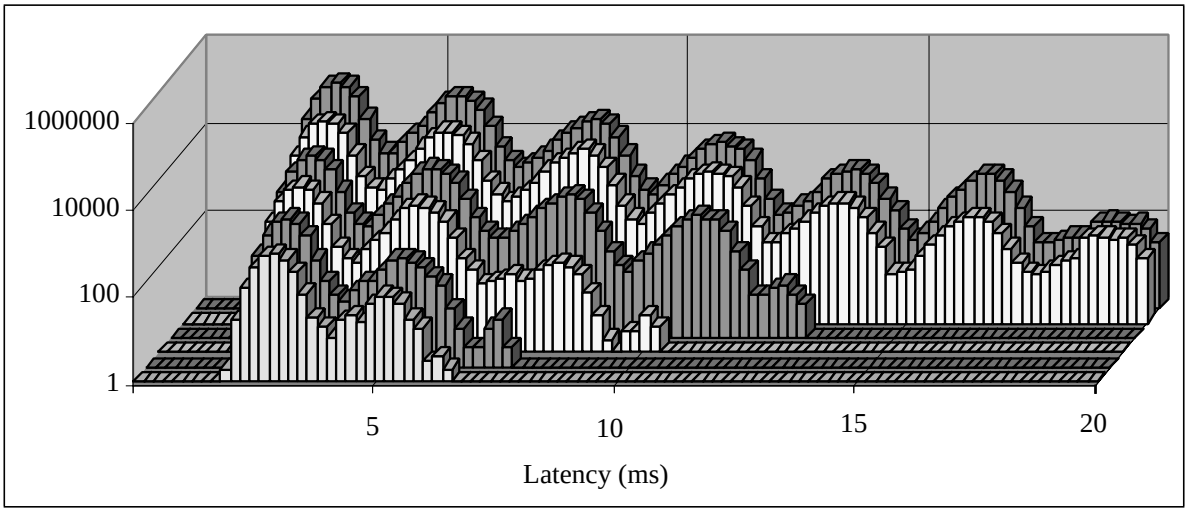


Figure 4-30 Distributions for the decision time (latency) of the architecture-B LVL2 trigger, as obtained after about 0.1, 0.3, 1.0, 3.0, 10.0 and 30.0 s of simulated time for the system of Figure 4-29, when single switches are used. The numbers along the vertical axis indicate the number of events. The relatively large increase of the tail at longer simulated times shows that the system is nearly or just unstable (i.e. longer and longer latencies may occur, until buffer overflow reduces the number of events to be processed).

ration of the testbed used for the measurements quoted here was 512 traffic nodes connected by a Clos topology network [4-54]. Two timing nodes placed on different final-stage switches are used to measure the single packet latency across the network. Before the emulation, each of the traffic nodes is loaded with traffic descriptors. These traffic descriptors inform the node when to send how much data to whom. The traffic nodes do not respond to incoming messages except to keep track of how much data they receive per second. Similarly they also keep information about the amount of data they have sent.

General results from the test bed in various configurations have been reported in [4-53] [4-55] [4-56]. Some results for architecture B and architecture C will be presented here.

Architecture B results

The generally assumed parameters within the demonstrator project as compiled in [4-46] have been the foundation for the emulation of architecture B.

The emulations were performed for one network at a time. The full global, SCT and muon local networks were emulated. Only parts of the TRT local network (80%) and the calorimeter local network (50%) could be emulated due to size constraints. See Table 4-16 for a more detailed description of the configurations used. For full details on the emulations of architecture B see [4-57].

Measurements and results

By using a mapping of the ROB and FEXs onto the traffic nodes where both the sinks and sources were distributed as much as possible, the emulations could sustain a LVL2 input event rate of 125 kHz for all but the local calorimeter network. The emulation of this network could only sustain a 25 kHz event rate. For the calorimeter we emulated a threefold increase in link speed on the HCAL links by having three traffic nodes send one third of the data at the same time. This scenario we refer to as ‘salvo.’ Another scenario is the ‘Round Robin’ where three nodes take turns to send the HCAL data, increasing the time between the packet dispatch time. The improvement in performance is shown in Figure 4-31. The increased link speed improved the

Table 4-16 Configuration of emulated networks.

Network	Sources	Sinks
SCT (100%)	256 ROBs	80 FEXs
TRT (80%)	414 ROBs	94 FEXs
Calorimeter (50%)	216 ECAL ROBs, 24 HCAL ROBs	143 FEXs
MUON (100%)	192 MDT ROBs, 22 RPC ROBs	10 FEXs
Global (100%)	495 FEXs	12 GTPs
Architecture C (100%)	64 CAL RSIs + 64 TRT RSIs + 32 MUON RSIs + 32 SCT / PIX RSIs	240 LVL2s
Event building(100%)	64 CAL RSIs + 64 TRT RSIs + 32 MUON RSIs + 32 SCT / PIX RSIs	112 EFs

throughput such that a 50 kHz event rate could be sustained, and decreasing the hit rate of the HCAL readout buffers allowed it to handle an event rate near 80 kHz.

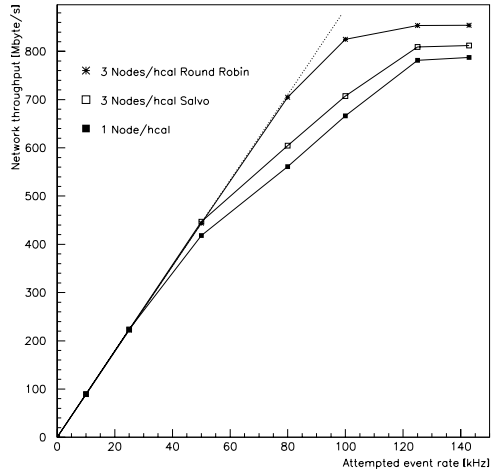


Figure 4-31 Throughput measurement on the HCAL readout buffer variations.

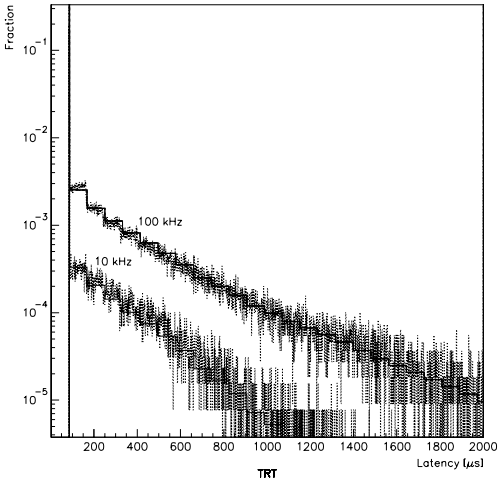


Figure 4-32 Single packet latency for the TRT local network.

We modelled the latency distribution as a competition for access in the switching network. We assume that the packets have to compete for access with the other packets in the network; a 'losing' packet must wait while the 'winning' packet is transferred before again competing for access. Owing to overlapping events there might be more packets competing in the second round. The packet latency is measured from the time when a packet first enters a competition until it wins, plus the nominal transfer time. Applying this model to the TRT local network, we obtained good agreement both at the 10 kHz event rate and at 100 kHz (Figure 4-32).

Architecture C results

Emulating architecture C on the Macrame test bed was done [4-58] using the same hardware configuration as that used for architecture B.

Input assumptions

The emulation of the architecture-C option for LVL2 has been performed using the parameters from [4-44]. The algorithm sequences are produced using a LVL2 trigger menu from [4-18], including missing-E_T and b-jet vertex tags. This menu is called 'LVL3' in [4-18].) The input rate is 34 330 Hz, and 75 different sequences are possible [4-18]. Sixteen nodes are dedicated to the supervisor, 240 to processors, and 192 to RSIs (32 muon, 64 calorimeter, 64 TRT, 32 SCT). The events are generated with a Poisson distribution in time, and the messages from different events are mixed. The messages are distributed on each source node and ordered by time stamp.

The emulation has been extended to full event building using the barrel-shifted transfer mode. For each event, each of the 192 sources sends 4 kbyte of data to the same processor; the transmission from successive sources is delayed by 500 μs. (The full event data is transmitted in about 100 ms). For this test, 112 nodes were dedicated to the event-building destinations. The events were generated with a Poisson distribution in time. The details of the hardware configuration for the architecture-C and event-building emulations are shown in Table 4-16.

Measurements and results

No congestion was observed for LVL2 or event-building traffic up to 60% port occupation. Congestion occurred when the occupation exceeded 60% on one of the input nodes (for the LVL2 traffic) or on one of the output nodes (for the event-building traffic). More studies will be necessary to explore any systematic limitations.

With the LVL2 model used in this study, the system was able to operate for input rates up to 75 kHz with a DS-link speed of 100 MBit/s. For the event-building traffic with the barrel-shifted transfer mode and Poisson-distributed event timing, the system was able to operate up to 850 Hz (LVL2 output rate).

Conclusion

Through the emulation studies on Macrame with architecture B we learned that distributing the sources and sinks evenly over the network rather than grouping them is important for enhancing the amount of available bandwidth. Emulating the calorimeter network taught us that decreasing the hit rate can be necessary to improve the throughput. We understood the latency distribution by a simple model of how head-of-line blocking can occur throughout the switching network.

The emulation studies with architecture B showed that the calorimeter local network could be run at an event rate of almost 80 kHz and the other networks in architecture B could be run at 125 kHz. The emulation studies with architecture C show that traffic patterns generated by a sequential LVL2 including complete algorithms such as b-jet tags, missing-E_T and the TRT full scan can be handled successfully.

Extrapolating from these results, we would expect that commercial networks could operate successfully for LVL2 input rates above 100 kHz and event building input rates above 1 kHz. On the other hand, a larger switching network might be required for stable operation at these high rates. (The design studied for architecture C was based on a switching network with 1024 nodes.) Further studies will be necessary to understand the influence of the particular Macrame hardware, such as the very limited size of the Macrame input buffers.

4.5.4 Conclusions of the Demonstrator Programme

At the end of the demonstrator programme the results of the various activities were reviewed and a number of conclusions drawn, these conclusions are presented here.

Network Technologies

Perhaps the most important conclusions from the demonstrator programme are in the area of network technologies. A better general understanding of switching technologies was obtained, especially from the measurements in the large scale MACRAME fabric. Measurements here have demonstrated that Clos networks are good at handling random traffic and that this applies also to the traffic patterns expected for the LVL2 system. However even in this configuration the links in a large network typically achieved only ~60% of their nominal performance with these traffic patterns, but will the final figure be better or worse than this? In addition if there are particularly active ports in the network head-of-line blocking can cause the system to become unstable even though most links are well below this loading.

The paper models indicated that the total network capacity required will be a few Gbyte/s, but that if the processors to be used have a speed of ~500 MIPS individual links only need to handle 10-20 Mbyte/s. With current trends in network technologies there is now increased confidence that commercial networks will offer these levels of performance in the timescale.

Within the programme important measurements were also obtained with two candidate technologies. Firstly it was demonstrated that ATM networking was able to support the performance required in a small system if hard sequential selection was used, however, this required the use of custom software drivers. Secondly many encouraging results were obtained with SCI, which exhibits very high performance, but although this technology is gaining an increased acceptance it is still not fully mature and it is not clear that all the components required to build a large system will be generally available.

Processor Technologies

General purpose RISC/CISC processors are the easiest to use and the most flexible for algorithms. Dual processor boards which are becoming increasingly common offer interesting possibilities, particularly to use one as an I/O processor. At the start of the demonstrator programme, there was a tendency to assume that many of the processors would use a VME form factor, however it soon became clear that for most of the processors a PC form factor was generally preferable (better price and performance) and the extra facilities offered by VME modules were not required. This conclusion remains true for the next phase and will need to be reviewed again for the final system. The continuing increase in processor speeds would now suggest that by 2003 processors with 2 GIPS performance will be appropriate - however this would also imply increasing network link bandwidths to ~50 Mbyte/s.

FPGAs have demonstrated a clear advantages for pre-processing and for algorithms like the TRT full scan. In addition the FPGA based ENABLE system demonstrated that it can support parallel feature extraction (for simple algorithms) at high rate. However, this requires RoI data collection which was partially demonstrated, but if it were to be used it would have a major effect on the ROB design. Given the substantial cost of this form of feature extraction (i.e. low-level custom programming, embedding the RoI Collector into the ROBs) it was agreed that this should be limited to as few (sub-)detectors as possible, and only used if other technologies are unable to solve the problem or lead into comparable complications.

The paper models had generally assumed an I/O processing overhead of 50 μs, measurements in the demonstrators indicated that with current processors the overhead was much lower, perhaps as little as 10 μs. Previously there had been interest in using Digital Signal Processors (DSPs) in various roles including for feature extraction. However, it was concluded that with the relative developments in DSPs and other processors, the niche for DSPs in our application is increasingly limited and the main interest remaining was for I/O processing.

Within the computing industry there are various developments of HPCN (High Performance Computing and Networking) systems, which provide distributed computing across large systems of processors. Initial measurements on these system seem to be encouraging and if current industry trends continue, HPCN systems could offer a very attractive solution, but they would have major influence on event building and event filter, and our concept of a ROB.

Architectural Issues

It is evident that sequential processing steps are required (at the very least for B-physics) so that the architecture must be able to handle this. It is also evident that parallel processing within an event is needed (e.g. Pre-processing, data-collection, TRT full-scan), although if sequential selection is used the case for parallel feature extraction is less clear. It was also noted that whilst the advantages of sequential selection have been well stated, there needs to be a Trigger Performance study as to whether the cuts assumed introduce any bias.

The RoI-Distributor demonstrated within Demo-B illustrated a scalable mechanism for the distribution of RoI information from the supervisor to the ROBs and the allocation of feature extraction processors within the local-global farm. The implementation, however, was limited by its use of VME to communicate to the ROB emulators, but even if an alternative path were used the performance would still be limited by the single PCI in the ROB emulators. It is clear that for the ROB emulators there is no fundamental difference whether the data requests are received via the general data network interface or via a separate path. It was concluded that there was no real advantage gained and significant extra complications of having a separate path for data and control messages and it was therefore agreed that in future we should use the general data network to pass control messages to the ROBs. (This might not apply for requests for data to be supply to ENABLE, although even here it would be acceptable if the network supports a broadcast facility).

Similarly for the ROB there is no distinction between the push and pull architecture, indeed even for a group of ROBs there is effectively no difference (in one case the request comes from the supervisor and in the other it comes from a processor). The fundamental difference between these modes is in the supervisor and the processors and on balance these were simpler to handle in the pull mode, it was therefore agreed to drop investigation of the general push architecture.

Concerns had previously been expressed about the extra load that an operating system might have on the processors within the LVL2 system. However, with the light-weight systems used the benefits far out-weighed the very slight loss in performance. It was therefore concluded that we should in general use an operating system - one possible exception being in the ROB. However, it was important that the next phase should include comparative tests of the performance of various micro-kernels and operating systems.

Error rates in the technologies used were low and in the small systems used very simple error handling was adequate. However, the full size systems will need better error handling.

Data Collection from ROBs was identified as a critical problem - especially for the calorimeter. Further study was required for all aspects of the ROB Complex.

Studies of using a sub-farm for the Level-2 tasks indicated that the CPU vs I/O balance did not match those of a probably subfarm. Essentially more CPU intensive tasks would be required to obtain a good match.

A supervisor with a design which allowed for multiple supervisor processors has been demonstrated. Very promising performance had been obtained with a single supervisor processor, but scaling had yet to be demonstrated. The supervisor had been used with both a push and pull architecture and in each case had been used to allocate a single processor from a farm (single farm or global) for each event. The allocation scheme used in both case was a round robin with free-list/tariff scheme to allow multiple events to be outstanding with a given processor. For the next phase it was agreed that the supervisor should pass full control for each event to a single processor, thus providing flexible distributed supervision within the event.

General conclusions

The demonstrators had shown that it was not possible in this case to factorise the architecture and the technology, the two being tightly coupled. It was suggested that for the next phase a few technologies should be chosen and then the architecture optimised for those technologies. If it was found that the technology does not match the problem then it may be possible to change the specification of the problem.

All of the demonstrator systems were smaller than originally foreseen (~1/4), however, an impressive set of results have been produced with many groups participating. Most of these results have now been written up as ATLAS Internal Notes.

4.6 Options retained and plan up to the Technical Proposal

4.6.1 Definition of the pilot project

This section presents the motivation for the pilot project [4-59] which should bring together the LVL2 community from several parallel R&D projects to an integrated common work focused essentially on understanding the fundamental task of the LVL2 trigger, i.e. to reduce the LVL1 rate by a factor of the order of 100 or more.

The pilot project should satisfy the following requirements:

- A duration of 18 months starting from March 1998 (T/DAQ Review) and completed by the end of 1999 (technical proposal).
- The details should be described by a clear and complete activity list, with realistic goals and work plans, and specified deliverables and milestones.
- It should be formed as a set of coherent and complementary activities, that map the resources and commitments all of the LVL2 groups.

It should allow a steady evolution from the present to the future work of the LVL2 community.

Since the ATLAS technical proposal, much work has been done which will help to define the new programme:

- A LVL2 user requirements document has been written [4-60].
- Studies of the selection process and trigger strategies reported in Section 4.3 have advanced our understanding of the trigger process.
- The demonstrator programme and other activities reported in Section 4.5 have produced many results on the technologies available.
- Over the last months a pragmatic analysis of 'where are we ?' has been made in the Trigger/DAQ community, with many discussions in the LVL2 community and with the DAQ/EF community.

The goals of the pilot project are to select a few technology candidates and to produce material for the technical proposal:

- Evaluate the best selection strategy for LVL2 to produce a reduction factor of ≥ 100.
- Contribute to the definition of the full global T/DAQ architecture, including integration with DAQ/EF.

During the demonstrator programme a variety of options to control the dataflow of Level-2 event fragments were considered. The main technical issue is that this task should be done at the speed of the Level-1 accepts (up to 75–100 kHz). At the end of the programme it was concluded that only certain options should be retained (i.e. that control traffic and data traffic should use the same network, that the supervisor passes control of data transfers within an event to a farm processor - see Section 4.5.4 above for details). During the course of the pilot project, when more experience has been gained, these conclusions will be reconsidered to confirm whether or not they are fully valid.

4.6.2 LVL2 primary critical issues

This section presents the main issues that should be studied in the short term since they may influence the design of the front-end electronics, including the readout drivers (ROD), and the overall integration of the Trigger/DAQ system.

4.6.2.1 Data collection

ROBs

The ROB is the functional component common to both Level-2 and the DAQ/EF dataflow. The requirements for the extraction of the RoI data for Level-2 are a critical constraint on the ROBs and may have some important consequences on their design parameters. Details of the mapping of each subdetector into the ROBs and the event data format could have a major impact on the Level-2 system. It is important that these are chosen to minimize the overheads of extracting, transferring and any necessary reformatting of the restricted data required by Level-2.

Raw data format and preprocessing

Another important parameter is the format of the raw data coming from the front-end detector electronics. Optimization of this format at the early stage of the chain can save a lot of band-

width and minimise the data manipulation in the ROBs or Level-2 system. Some problems could be reduced by preprocessing at the ROD level, but clearly this would influence the design of the RODs. The balance between extra complexity of the RODs and benefits for the ROB data extraction and formatting is only partially understood at this stage and requires further study.

4.6.2.2 Data flow optimization

The optimization of the dataflow is a key element of the architectural design. This aspect depends on both the trigger process strategies and optimized data/control paths.

Trigger process strategies

The organisation of the dataflow should be adapted to the various strategies from low luminosity, where B-physics is an important element, to high luminosity, where event pile up increases the complexity and decreases the efficiency of the on-line rejection. The way the various algorithms are organised and the strategy to accept or reject events is strongly connected to the physics requirements. Whilst many details of the trigger process can be justified using physics arguments and realistic rate estimates, it is important not to ignore other effects such as unforeseen or underestimated backgrounds, inefficiencies or failures in detector or trigger elements. Flexible organisation of the trigger process, plus redundancy in the system will be important aspects of the solution. Evaluation of the performance of various steering sequences and their consequence for the architecture will help to understand such fundamental issues.

Optimal data/control paths

The trigger strategy and the hardware architecture will determine the message passing rate and the resulting bandwidth, which in turn will influence the design of the dataflow. With the many components involved in each trigger decision it will be necessary to fully optimize the interfaces and various software overheads.

4.6.2.3 Large system aspects

The final system will connect some 1700 ROBs to a few hundred or more processors. The size of this system as well as the evolution of commercial technologies make it impractical to realize a ‘full-scale’ prototype. Thus the tests are to be performed on relatively small slices of a complete system and we are obliged to rely heavily on simulation models to scale the behaviour to a full-size system. In addition, these initial studies are being made separately from the other parts of the Trigger/DAQ system and the latter stage of the next phase should start to address this integration.

Division between LVL2 and event filter

The present logical division between Level-2 and the event filter (EF) is the following:

- The Level-2 task is to reduce the rate by a factor of ~ 100 using RoI data fragments collected from the ROBs. The global Level-2 processor issues a Level-2 accept or reject to the Supervisor. Because of the rate requirements of the Level-2 system it uses optimized and simplified code for the algorithms and possibly reduced calibration and alignment parameters.
- The DAQ/EF collects for each Level-2 accepted event the ROB raw data fragments and assembles the entire event prior to processing a filter algorithm based on offline code.

This division may change according to the Level-2 and event filter final strategies, and depending on progress in the technology available and our understanding it may be possible to integrate them into a single high-level trigger (HLT) sequence.

Can LVL2 and event building for EF use the same network technology?

Level-2 needs a fast network capable of transferring small distributed packets of ~ 1 kbyte at a high rate (10 KHz) and with low latency. The full event building for the EF needs a network capable of transferring uniformly from every ROB a ~ 1 kbyte packet at 1kHz rate with modest latency requirements. Thus the technical requirements for the data-collection networks are different, but taking into account the evolution of commercial networking technologies, it is possible that they may be able to use the same technology. Such a decision would have to be taken closer to the construction phase when a full analysis of the advantages and disadvantages, including total system complexity and cost could be properly assessed.

4.6.2.4 Conclusions and guidelines for the future

Taking into account the conclusions of the demonstrator programme and the analysis of the issues to be addressed, the following list of topics should be studied in this next phase of the work:

- Quantify the advantages of different Level-2 trigger strategies and architectural implementations.
- Evaluate alternative approaches to the ROB complex.
- Demonstrate an RoI builder and the scalability of the supervisor.
- Draw conclusions about where FPGA processing is relevant.
- Investigate and quantify the problems of building very large high performance networks and of interfacing efficiently to them.
- Design, test and optimize the Level-2 trigger ‘evaluation’ software.
- Model full (alternative) architectures, including trigger strategies and network technology choices.
- Make revised cost estimates for the Level-2 system
- Investigate integration issues with other parts of ATLAS, in particular the Level-1 trigger and the DAQ/EF (including networks and other components, and the division of event selection between Level-2 and the event filter).

4.6.3 Activity matrix: general overview

The main objective is to develop the themes progressed in the demonstrator programme into a more fully integrated scheme mapping the various activities of the Level-2 community and the technical issues involved. Figure 4-33 shows the organizational chart of the different activities.

It is divided logically into three main complementary areas:

- The ‘functional components’ study and optimize in a ‘standalone’ mode each functional element of the system: the level-2 aspects of the ROB complex; the Supervisor; processors, coprocessors and networks; and watches general technology developments.

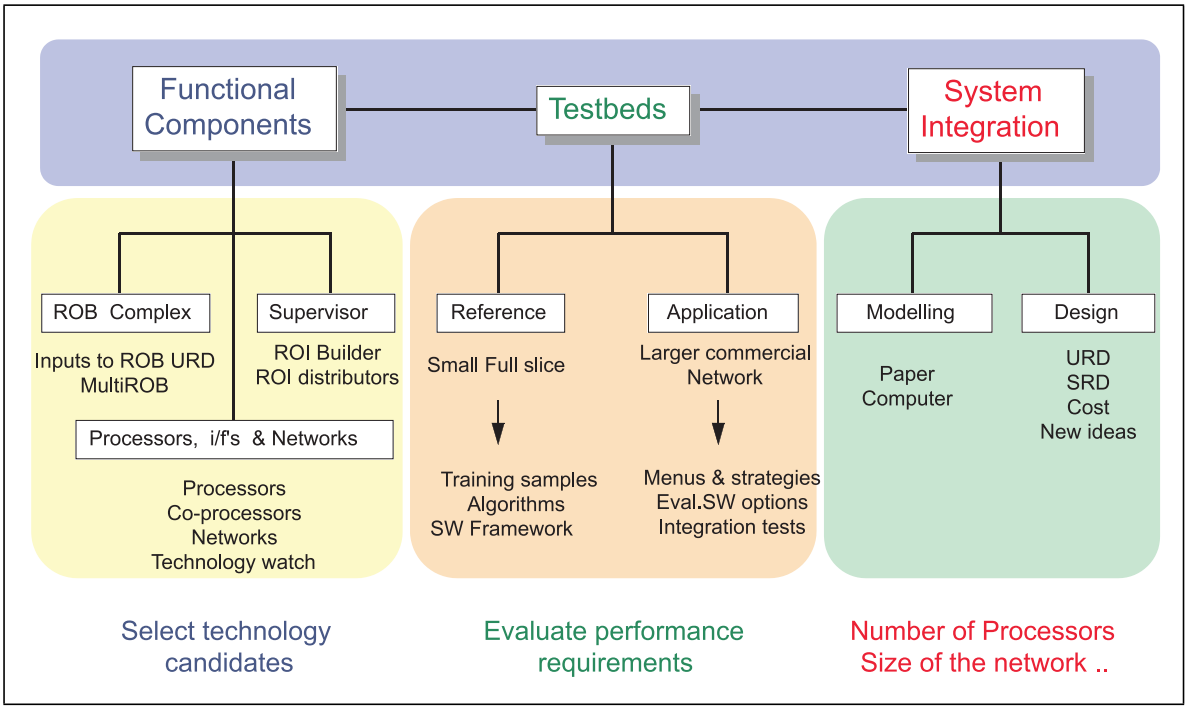


Figure 4-33 Activity matrix.

- The ‘software test beds” develop and evaluate all the software aspects using small functional ‘full-slice’ modular test systems. The ‘reference’ software is for the development of the software and uses a small system based entirely on commodity hardware. The ‘application test beds’ are used to evaluate various selection strategies and operational control modes as well as measuring the performance requirements of the full system. These application tests are run on somewhat larger systems also based largely on commodity hardware. It is also foreseen that some tests will be run integrating into the test beds items from the functional components.
- The ‘system integration’ uses a top-down approach to the system and combines the information from the other two areas to form a picture of the complete system. Modelling tools and techniques are used to simulate and scale the results from the small test beds to full systems. Overall system designs are checked against the URD and cost estimates made.

4.6.4 Functional components

This section describes the activities studying the hardware components and their optimization.

4.6.4.1 ROB complex

This activity studies the ROB complex from a LVL2 perspective. The work divides into four areas, studies of the MultiROB concept, the ROBin, preprocessing and the ROB URD.

MultiROB concept

The multiROB aims to match the input data rate from a standard ROD link to an optimum data request frequency and data size for the Level-2 system. It contains the following functional components:

- The input buffer (called a ROBin) which receives data from a single ROD. The local buffer manager task of the ROBin is performed by a microcontroller which is either on the ROBin or shared between several ROBins.
- A bus or local network, used to collect RoI data for Level-2 from several ROBins.
- A control processor.
- An output interface to the Level-2 (local or global) network.
- An optional coprocessor performing local processing or preprocessing.

ROBin developments

Because of the importance of the ROBins a number of implementation options are being studied within this co-ordinated activity. These option build on developments which have been proceeding for a considerable time in several institutes.

- The ROBin being developed originally for DAQ/EF-1 has an S-link data input feeding memories controlled by an i960 microcontroller. The data is read out via a PCI bus. The ROBin may exist in PCI or PMC form factors. This development is being done as part of the DAQ/EF-1 project and will be used by that project for studies of the readout crate.
- The other ROBin being developed also uses an i960 for the buffer management and a PCI bus for the readout, but here the use of memory based on SDRAM is being investigated. In this case the initial aim is to study the internal memory and the buffer management and there is no input link (at least in the first phase). The first implementation uses a PMC format, in later phases it is planned to use compact PCI, with a compact PCI crate to collect data from a number of such ROBins.
- The third development is a SHARC-based card. This design study aims to investigate: moving as many functions as possible from hardware to software; a low-power alternative technology; using a ring buffer for the main memory, rather than page-managed memory; using SHARC links to provide a MultiROB capability. The readout of the card would go via SHARC links to a SHARC which outputs the data via PCI.

In addition solutions using SCI technology components in combination with direct interfacing to SCI and the exclusive use of FPGAs in combination with commercial general-purpose systems are being investigated.

Preprocessing

An integral part of the ROB complex is the processor used to prepare data from one or more ROBins prior to the data being sent out on the general network. At the present time the main candidate for the preprocessing processor is the microEnable. In the case of a Multi-ROB it is vital that this be studied as an integral part of the ROB complex since several ROBins could supply data to a single preprocessor. However even in the case of a preprocessor only handling data from a single ROBin there is still a tight coupling within the ROB complex because of the shared use of the ROB PCI bus.

Input to the ROB URD

An initial user requirements document has been written for the ROBs, and the ROB complex group has the responsibility to ensure that the requirements of the Level-2 system are included in this document.

4.6.4.2 Supervisor

The Supervisor complex receives data from Level-1 with information about the RoIs found, reformats this data into a form usable by Level-2, assigns a main processor for each event and passes control for the event to that processor. After the Level-2 system has finished processing each event it receives the decision for the event and after taking into account issues such as pre-scaling and the current mode of operation (e.g. Level-2 selection or tagging) it makes the final Level-2 decision. These decisions are then grouped prior to broadcasting to all of the ROBs. The grouping of decisions reduces the frequency with which the ROBs have to handle decision records, since there is a significant I/O overhead associated with handling each record, independent of the record length. This mechanism reduces significantly the load on the ROBs).

The Supervisor complex consists of two main parts:

- The RoI builder, which receives the data from Level-1 and reformats it for Level-2
- The Supervisor itself, which contains a farm of processors to handle the allocation of events to the Level-2 processors and the processing of the decision records back from the Level-2 system.

The basic principles of the Supervisor were demonstrated as part of the demonstrator programme where a Supervisor with a single processor to handle the event allocation was run at event frequencies of up to 27 kHz. However, although scalability was built into the basic design it has still to be fully demonstrated.

Detailed work on the RoI builder has started only recently. With the submission of the LVL1 technical design report [4-2], the interface between LVL1 and LVL2 is now better defined and it is timely to start working on the detailed design of a prototype for this component and to demonstrate a solution compatible with the extreme demands of both the LVL1 system and the supervisor design.

4.6.4.3 Processors, interfaces and networks

This activity covers a wide area of technologies, but the inter-relationships make any split artificial. The main headings can be summarized as:

- Special-purpose coprocessor. Here we consider especially the FPGA work of Enable++ foreseen for such tasks as the TRT full-scan. More general FPGA solutions could be included in this study.
- Processors. High-performance computer networks are currently receiving much emphasis within the computing industry. If as some people predict these machines become more affordable they offer some exciting possibilities for the Level-2 problem. In particular they promise an appropriate mix of the CPU power and the interprocessor communication. In the first instance the activity would be limited to small-scale studies, in conjunction with the manufacturers of such systems, to see if the systems are as good as the promise and if so how the raw data can be fed into the memories of the system.
- Networks. Networking technologies are currently developing very fast. A number of technologies, listed below, are being investigated for the Level-2 networks. Others such as Myrinet and Raceway will not be actively pursued, but will be included in the technology watch. In addition some HS-link components already exist in CERN and these might be used internally as part of a switch for other technologies.

- Fast Ethernet is an established and very widely available commodity technology. Over the last year the status of the Gigabit Ethernet standard has evolved rapidly, with ever increasing numbers of products becoming available. The Ethernet set of standards is now well placed to offer the building blocks necessary to construct networks for the ATLAS second level trigger. Given the very large number of sites which are likely to upgrade campus networks largely based on Ethernet over the next few years, this technology promises to give very good price / performance ratios. A key area to be evaluated, however, is the performance of large switching fabrics. The performance of switches under heavy loads is already seen to be very dependent on internal details which vary greatly from manufacturer to manufacturer. It will be necessary to understand the critical factors and to evaluate whether the solutions chosen by manufacturers yield the performance in our particular problem.
- SCI has over the last year or so become more established, especially inside high-end server machines. However the availability of switches is currently very limited, with only very small switches available, as yet commercially unproven. However this technology gave the best communication results within the demonstrator programme and so this technology, and particularly switches, justifies further study.
- ATM is an established technology for wide area networks (WAN) and more generally in the telecommunications industry. However its promised take-up for LANs has been much slower to develop. Over the last year a new generation of modular ATM switches (with typically 16–64 ports) has led to an expansion in this area and an increasing number of sites are using ATM for backbones. However, with the rapidly changing availability of Giga-Ethernet equipment it is difficult to predict how ATM in the LAN market will continue to evolve.

- Technology watch. We have listed above many technology developments which could impact the ATLAS Level-2 problem. There are clearly far too many developments to study them all, however, it would seem prudent to have as a small activity a group keeping an eye on new developments across the technologies listed (and others) so that if something especially relevant emerges we are aware of it as early as possible.

4.6.5 Software test beds

The main objective of this integrated activity is to complete and complement items from the demonstrator programme, e.g. the full Level-2 trigger software chain from extracting the RoI data to the Level-2 Accept / Reject signal. This software includes four main topics

- production of realistic raw data samples from physics simulations,
- optimization of appropriate algorithms in online form,
- selection strategies and steering sequences,
- a common software framework including the data collection and request / response protocol.

The common framework, together with the steering sequences and online algorithms combine together to form the ‘reference software’.

To achieve the goal of this activity, it is foreseen to assemble a set of full-slice test beds to develop and run the reference software. The testbeds will take the form of a reference hardware configuration and larger application test beds.

4.6.5.1 Reference test bed

The reference test bed hardware configuration (see Figure 4-34) will allow the development of the reference software. This includes the message-passing protocols and the infrastructure to run the Level-2 applications. In addition it should provide an environment to optimize the code of the algorithms and for their benchmarking. Features of the test bed are that

- all ‘data sources’ connect to all processors,
- a single physical network is used for data collection and control,
- processor elements can be used for local or global processing.

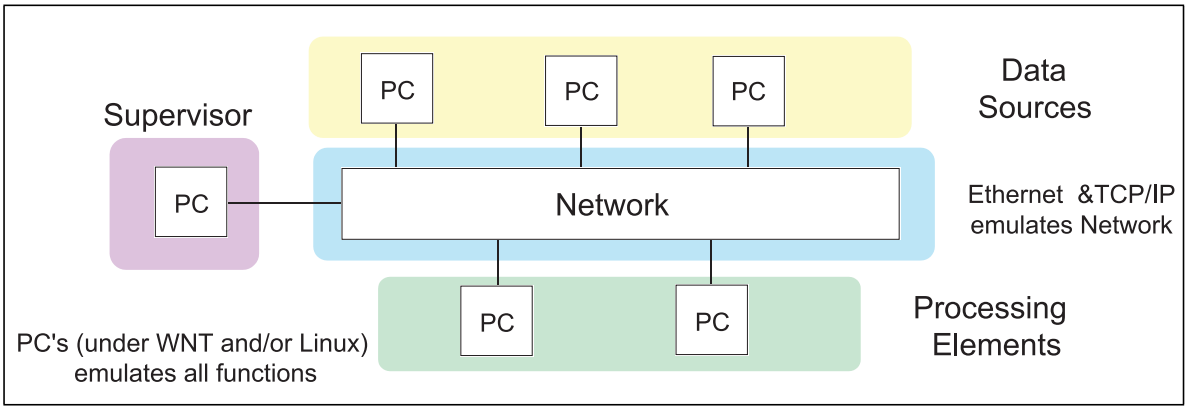


Figure 4-34 Reference testbed configuration.

Training samples

Data samples are needed for algorithm benchmarking, including data preprocessing, and to run the full trigger strategies on the application testbeds. The production of data samples will be accomplished offline using a special version of the trigger simulation program, ATRIG. It uses simulated detector data from DICE and writes them onto data files in a pseudo-raw data format, together with Level-1 and Level-2 trigger results. Starting from these files, which contain selected physics channels or background data, new files will be produced with a realistic raw data format as soon as this becomes available for each subdetector. These files will be particularly useful in the evaluation of data preprocessing.

The training samples will consist of a set of data files containing a collection of different event types (signal plus noise). These training samples will be used to study trigger strategies and processing sequences.

Algorithm optimization and benchmarking

The algorithm optimization and benchmarking will need the selection of at least one platform to be used as reference system. In this way a library of standalone versions of the trigger code running on the same type of processors (e.g. the testbed farms) will be available and also provide a coherent set of processing times to be used for modelling.

The online versions of the algorithms should be coded in C or C++. The library should contain all the feature-extraction code for muons, electron / photons, hadrons, jets and missing E_T, including combined reconstruction of trigger objects (e.g. muon isolation, matching of calorimeter and inner detector, etc.) and specialized algorithms (e.g. B-vertex tag). It should also include algorithms for the B-physics triggers that are foreseen for low-luminosity running. The steering sequence of the Level-2 trigger (menu driven) should call this library to identify trigger objects and take the global decision on the event.

Online framework

- Run control, monitoring and error handling.
- Configuration (from text file to database?).
- Request / response protocol (event flow control and message passing).
- Technology- and OS-specific libraries.

Software overview

The aim of the 'Reference Software' group is the development of the full Level-2 trigger process which will run on the application testbeds (Figure 4-35).

It is proposed to prepare a flexible, well-engineered, technology-independent software whose development will be based on the following assumptions:

- it will reuse and adapt previous work (i.e. from the demonstrator programme) where appropriate;
- it will have a modular structure to map easily across nodes and networks;
- it will not be the final software.

The design of this software should be as flexible as possible, supporting 'generic' and 'specific' implementations, and except for the lowest layers be independent of the operating system and hardware platform. In particular the code should be portable between Windows NT, LynxOS, Linux and applicable to ATM and Fast-Ethernet technologies in a transparent way. The use of suitably layered software with appropriate APIs should match this goal.

The effort in the development of software modules (such as the run control) which will be provided in future by the DAQ group will be kept to a minimum. These modules will only have the minimal functionality required to run the entire Level-2 software process. In contrast the development of core software will concentrate on specific Level-2 issues and optimization of message-passing protocols which match the Level-2 requirements.

4.6.5.2 Application testbeds

The full slice application testbeds will embed the Level-2 trigger process as developed in the reference software on relatively larger and better optimized systems (see Figure 4-36) to address the following issues:

- the study of full algorithms, steering sequences and global menus (menus based on trigger RoIs only, full menus with secondary RoIs, B-physics menus);
- the evaluation of protocols and measurements of the hardware performance required for a complete system (e.g. latency, link occupancy and software overheads);

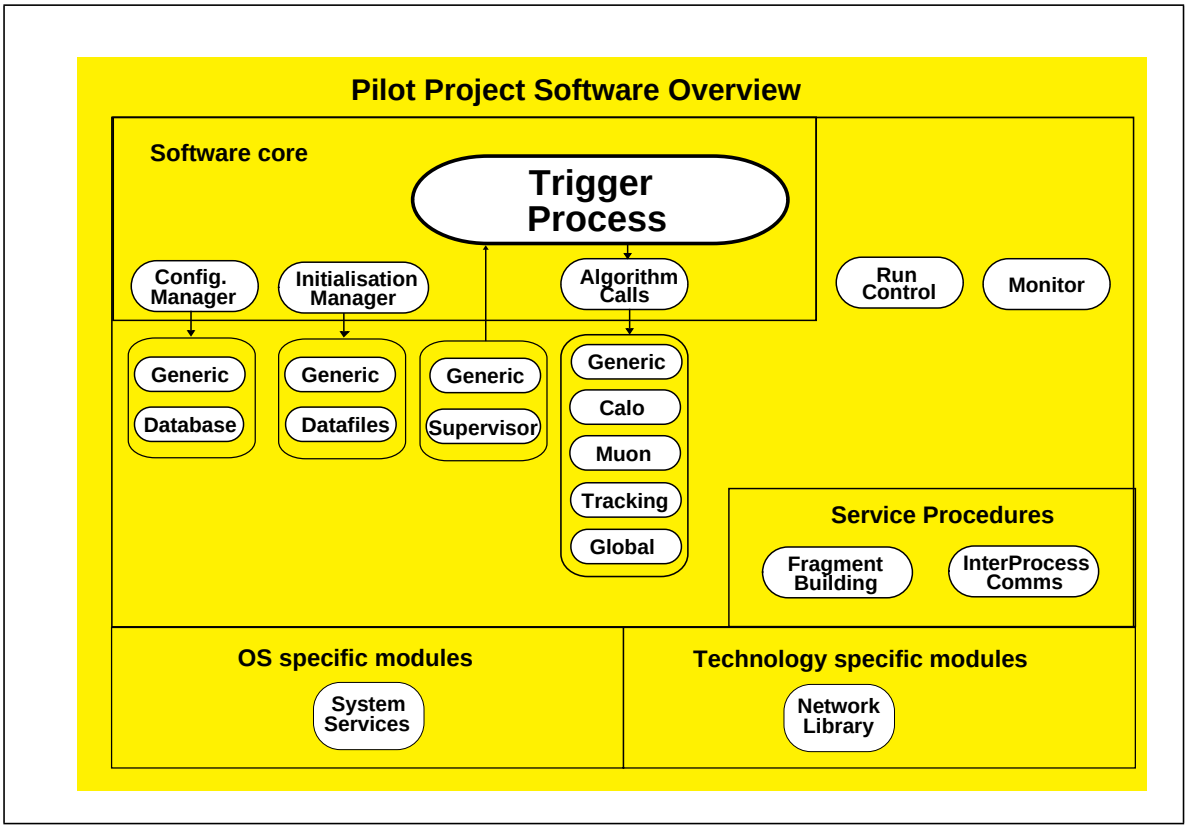


Figure 4-35 Software overview.

- the integrated tests of optimized components (e.g. ROB prototypes, data preprocessing units, etc.) and better commercial switching networks once they become available

For the first two issues the hardware platforms which have been selected use ATM and Fast Ethernet for the network technology; Pentium processors for the processing farms and for ROB and Supervisor emulation; optionally PowerPC-based cards for ROB emulation and Supervisor prototypes. The software should allow the use of Windows NT and Linux in PCs and LynxOS in the VME cards.

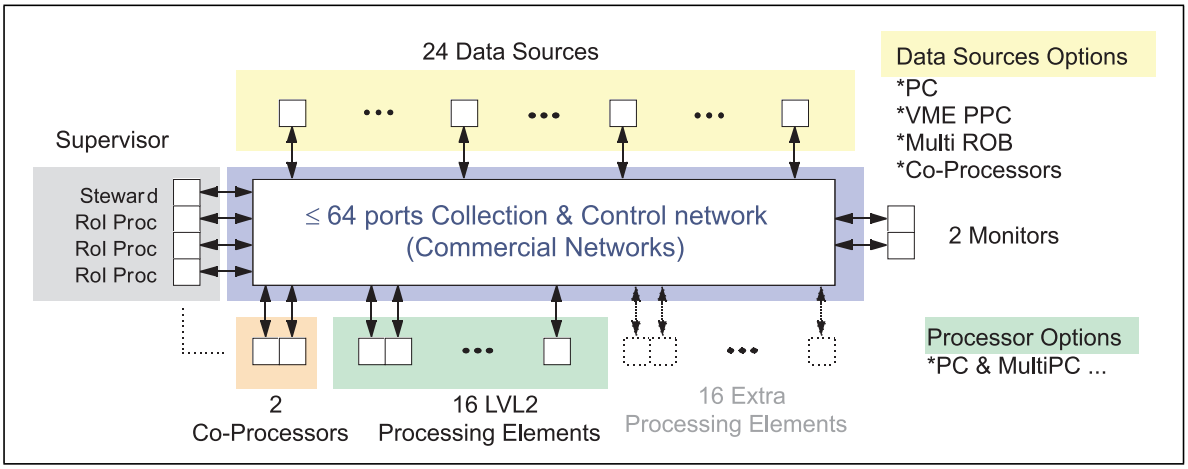


Figure 4-36 Application testbed configuration.

Asynchronous Transfer Mode

The proof of principle to use Asynchronous Transfer Mode (ATM) technology for the central network was demonstrated with a limited number of 155 Mbit/s nodes (10) and a simple message exchange scheme. The ATM testbed will be used to study the options for the trigger process as detailed above, but it will also allow further evaluation of the technical details and performance of this technology with a larger number of nodes. The idea is to assemble several (3 to 4) standard commercial 16-nodes subsystems in order then to reconfigure the components into a larger one (48 to 64 maximum). This will allow us to gain experience with multiswitch and multipath networks by building in a first step a two-stage 16-port interconnect and then a Clos multipath network. The final 48–64-port full-slice system would also be used to evaluate the performance of the trigger process itself on a moderately large system. An important element of the programme will be measurements of the various parameters (overheads, latencies and occupancies) throughout the system as input to the modelling (see below).

Ethernet

Ethernet is a popular and widely used networking technology. The emergence of faster 100 Mbit/s – 1 Gbit/s devices (interfaces and switches) now makes this technology very attractive and very cost-effective. However, a demonstration is needed in a large full-slice system that the performance and technical features are adequate to solve our specific problems, in particular the efficient transmission of small messages. Issues of scalability and performance for larger switching fabrics also need to be addressed. In addition to studying the basic technology (interface cards and custom drivers) within the 'technology activity', it is planned to use a large system of 32 to 64 nodes to study the options for the trigger process and to evaluate and measure the various critical parameters using the same criteria and trigger models as in the ATM testbed.

4.6.6 System design

This area groups together the information from the component studies, the software testbeds, and the requirements and constraints from other parts of ATLAS to obtain a view of the complete Level-2 trigger system. In particular it has to understand the consequences of a given option or implementation including integration aspects with the rest of the detector and T/DAQ.

4.6.6.1 Modelling

Whilst the application testbeds allow the study of moderately sized systems with large traffic loads, they do not give the complete picture of how the final full-size system will work under the full load. To obtain information on how the performance would continue to scale to the final system it is essential to use modelling techniques. The main aspects of this activity are to optimize various ROB layout and network options, to study congestion and contention within the system.

Paper model

The paper model is a very simple calculation used for initial optimization and to estimate the network size and the computing needs of a given option for the trigger process. It uses a simplified model of the process under study and a complete set of average parameters: trigger rate, data volumes, transfer rates, execution times and software overheads. No account is taken of congestion or queuing or variations in processing times or data volumes, and thus the results give a lower bound to the size and performance of the system required. (Note that in reality the

paper models use large spread-sheet calculations, but in essence they are simple calculations which could be done by hand.) However, as was shown during the demonstrator programme, this technique allows a relatively rapid study of many options and allows the initial optimization to be obtained, worthy of further study by more detailed models. In particular it is now necessary to update this work with revised parameters.

Computer model
This is a complementary activity to the paper models using more refined and complex 'discrete event simulation' programs (SIMDAQ written in C++, Ptolemy code). The average parameters are replaced by distributions or dynamic values and full account is taken of queuing and congestion. Detailed models of each individual component of the Level-2 system (ROB complex, Supervisor, Networks, Interfaces, etc.) should help to optimize and understand the critical features of these devices. A generic model of the various application testbeds will allow the model to be normalized and to check the validity of the model. The generic model would then be used to scale the performance to a full system. Finally, adding technology-dependent elements into the model will allow a complete cross-check with the testbeds and scaling to a full system with those technologies.

The current work plan is to complete additions to SIMDAQ in order that it can simulate all architectures currently being considered, and also include sequential trigger strategies. Then comparative simulations of these architectures under different conditions will be made to provide information for decisions to be taken later on. A next step is generic simulation (with relatively simple models) of laboratory set-ups and identification of main factors determining behaviour. This is to be followed by detailed simulation of technologies calibrated using results of laboratory measurements and reusing code / models developed earlier for ATM, SCI and DS-link technologies.

4.6.6.2 Integration

This activity uses the information from all the other Level-2 activities, plus information from other systems to produce a top-down view of the complete Level-2 trigger system, including models of the cost.

User requirements document
The user requirements document (URD) cannot yet be considered to be a frozen document: parameters of other parts of ATLAS continue to change and the understanding of the physics needs continues to evolve. The task is to determine, with the appropriate parties within ATLAS, the changes required in the URD and to incorporate these changes in new versions.

System requirements document
Over the duration of the pilot project the URD should become sufficiently stable and our understanding of the possible implementation of the Level-2 trigger system, including the interfaces to other systems, sufficiently advanced that we can proceed to the next phase of the design process with the production of a system requirements document (SRD).

Cost overview
Among the parameters to be taken into account in choosing the final Level-2 system will be the relative costs of different options. We aim for the technical proposal to have information on the likely cost of each component, and to combine these costs and the best estimates of the size of system required to obtain a revised costing model for the complete system.

Integration with other systems

The proper design, and later integration, of the Level-2 system requires a detailed knowledge of the requirements and constraints from other parts of ATLAS. In addition the Level-2 system will impose some requirements and constraints on other parts of ATLAS. The task is to collect and document these requirements and constraints, and where necessary to ensure that changes are made in an optimal way. It is assumed that much of this work will take place as part of the activity of the detector interface group. Particularly important examples are: the mapping of detector elements into ROBs; the format of the event data in ROBs; the interface between the Level-1 system and the Supervisor; the interface to DAQ/EF.

4.7 References

4-1 ATLAS Technical Proposal, CERN/LHCC 94–43, LHCC/P2 (1994).

4-2 Level-1 Trigger Technical Design Report, CERN/LHCC 98–14, ATLAS TDR 12 (1998).

4-3 O. Boyle et al., The S-Link Interface Specification, CERN ECP (1997). http://www.cern.ch/HSI/s-link/spec

4-4 ATLAS Inner Detector Technical Design Report (Volumes I and II), CERN/LHCC 97–16 and 97–17 (1997).

4-5 ATLAS Trigger Performance Status Report, CERN/LHCC 98–15 (1998).

4-6 P. Creti et al., Testbeam results from the Calypso MDT chamber, ATLAS Internal Note, MUON–NO–196 (1997).

4-7 U. Egede, LUNDF6/(NFFL-714X). http://atlasinfo.cern.ch/Atlas/GROUPS/INNER_DETECTOR/LAYOUT/DOC/xconver.ps

4-8 D. Cavalli and S. Resconi, Tau-jet separation in ATLAS detector, ATLAS Internal Note, PHYS–NO–118 (1998).

4-9 S. Tapprogge, Jet Reconstruction at the ATLAS Second Level Trigger, ATLAS Internal Note, DAQ–NO–115 (1998).

4-10 I. Gavrilenko, ATLAS Internal Note, INDET–NO–16.

4-11 P. Morettini et al., An impact parameter based Level-2 trigger using ATLAS pixel detector, in Proceeding of the Third Workshop on Electronics for LHC Experiments, London, September 1997.

4-12 S. Sivoklokov et al., Second Level Triggering in the Forward Region of the ATLAS Inner Tracker, ATLAS Internal Note, INDET–NO–111 (1995).

4-13 P. Eerola, Description of the TRT track trigger simulation in ATRIG, ATLAS Internal Note, DAQ–NO–050 (1996).

4-14 R.K. Bock et al., TRT Preprocessing — Algorithm, Implementation and Benchmarks, ATLAS Internal Note, DAQ–NO–66 (1997).

4-15 R.K Bock and B. Kastrup, Realistic Calorimeter Feature Extraction, Algorithm, Benchmarks, and Implementation Options, ATLAS Internal Note, DAQ–NO–65 (1997).

4-16 A. Amadon et al., ATLAS Trigger Menus at Luminosity 10^{33} $cm^{-2}s^{-1}$ —Version, ATLAS Internal Note, DAQ–NO–110 (1998).

4-17 S. Tapprogge et al., Implementation of Trigger Menus for the ATLAS Second Level Trigger, ATLAS Internal Note, Reference Software Note 15 (1998).

4-18 J. Bystricky et al., ATLAS Trigger Menus at Luminosity 10^{33} $cm^{-2}s^{-1}$, ATLAS Internal Note, DAQ–NO–54 (1996).

4-19 P. Maley et al., A Local-Global Implementation of a Vertical Slice of the ATLAS Second Level Trigger, ATLAS Internal Note, DAQ–NO–81 (1998).

4-20 R.E. Hughes-Jones et al., Using SCI to Implement the Local-Global Architecture for the ATLAS Level 2 Trigger, ATLAS Internal Note, DAQ–NO–111 (1998).

4-21 D. Calvet et al., An ATM demonstrator system for the Sequential Option of the ATLAS Trigger, ATLAS Internal Note, DAQ–NO–104 (1998).

4-22 A. Kugel et al., ATLAS Level-2 Trigger Demonstrator-A Activity Report — Part 1: Overview and Summary, ATLAS Internal Note, DAQ-NO-85 (1998).

4-23 A. Kugel et al., ATLAS Level-2 Trigger Demonstrator-A Activity Report — Part 2: Demonstrator Program, ATLAS Internal Note, DAQ-NO-84 (1998).

4-24 A. Kugel et al., ATLAS Level-2 Trigger Demonstrator-A Activity Report — Part 3: Paper Model, ATLAS Internal Note, DAQ-NO-101 (1998).

4-25 B.J. Green et al., A second level data buffer with LHC performance, NIM A360 359 – 362 (1995).

4-26 Dolphin Interconnect, PCI-SCI Bridge Functional Specification.

4-27 J. Bystricky et al., A Sequential Processing Strategy for the ATLAS Event Selection, IEEE Transactions on Nuclear Science, vol. 44 (1997).

4-28 P. Le Du et al., Overview of the Sequential Single farm option for the ATLAS High Level Trigger, ATLAS Internal Note, DAQ–NO–109 (1998).

4-29 D. Calvet et al., Performance Analysis of ATM Network Interfaces for Data Acquisition Applications, in Proceedings Second International Data Acquisition Workshop on Networked Data Acquisition Systems, Osaka, Japan, 13–15 November 1996, World Scientific Publishing 1997, pp. 73–80.

4-30 M. Huet et al., A Read-Out Buffer for ATLAS, ATLAS Internal Note, DAQ–NO–105 (1998).

4-31 ATLAS ROB User Requirements Document, ROB–URD–V1.2.0 (1996). http://www.cern.ch/HSI/rob/URD

4-32 B.J. Green et al., A test system for second level trigger and data acquisition architectures, Proc. CHEP'92 701-705, CERN 92–07 (1992).

4-33 R.K. Bock and P. Le Du, Detector and readout specifications, and buffer relations, for the level-2 trigger demonstrator program, ATLAS Internal Note, DAQ–NO–62 (1997).

4-34 M. Costa et al., Results from an ATM-based Event Builder Demonstrator, IEEE Trans. on Nuclear Science, vol. 43, no. 4, 1996, pp. 1814–1820.

4-35 I. Mandjavidze, Evaluation of the RCNP ATM Network and Computing Facility for Emulation Purposes of Network Based Trigger/DAQ Systems, ATLAS Internal Note, DAQ–NO–069 (1997).

4-36 IEEE Computer Society, IEEE Standard for Scalable Coherent Interface (SCI), IEEE Std 1596–1992.

4-37 LCB Status Report/RD24, CERN/LHCC LHCC 96-33, (1996).

4-38 R.K. Bock et al., Benchmarking Communication Systems. ATLAS Internal Note, DAQ–NO–77 (1997).

4-39 R. Blair et al., Supervisor for the Demonstrator Project. http://pc41.hep.anl.gov/Atlas/T2/Supervisor_demo.htm

4-40 Standard for Heterogeneous InterConnect (HIC): Low Cost Low Latency Scalable Serial Interconnect for Parallel System Construction, IEEE Std 1355–1995. http://stdsbbs.ieee.org/groups/1355

4-41 B. Kastrup et al., Study of the FEX Subfarm Concept with C40s and Digital Memory Channel Cluster, ATLAS Internal Note, DAQ–NO–68 (1997).

4-42 R. Hauser and I. Legrand, Algorithms in second-level triggers for ATLAS and benchmark results, ATLAS Internal Note, DAQ–NO–27 (1994).

4-43 A. Kugel et al., A Hybrid Approach for the ATLAS Level-2 Trigger, ATLAS Internal Note, DAQ–NO–78 (1997).

4-44 J. Bystricky et al., A Model for Sequential Processing in the ATLAS LVL2/LVL3 Trigger, ATLAS Internal Note, DAQ–NO–55 (1996).

4-45 M. Dobson et al., Paper Models of the ATLAS Second Level Trigger, ATLAS Internal Note, DAQ–NO–113 (1998).

4-46 S. George et al., Input parameters for modelling the ATLAS LVL2 trigger, ATLAS Internal Note, DAQ–NO–70 (1997).

4-47 A. Amadon et al., Architecture C performance from Paper Models, ATLAS Internal Note, DAQ–NO–106 (1998).

4-48 S. Hunt et al., SIMDAQ — A System for Modelling DAQ/Trigger Systems, IEEE Transactions on Nucl. Sci., vol. 43, no. 1, 1997 pp. 69–73.

4-49 J. C. Vermeulen et al., Discrete Event Simulation of the ATLAS Second Level Trigger, in Proceedings of the Xth IEEE Real Time Conference, Beaune, France, September 1997, pp. 463–468; also: ATLAS Internal Note, DAQ–NO–86 (1997).

4-50 The Almagest — Volume I: Ptolemy 0.7 User's Manual. http://ptolemy.eecs.berkeley.edu/papers/almagest/user.html

4-51 Realisation of a 1000-node high speed packet switching network. Presented at ICS-NET '95, St.Petersburg, Russia (1995). http://www.cern.ch/HSI/dshs/Welcome.html

4-52 SGS-Thomson Microelectronics, The STC104 Asynchronous Packet Switch, Preliminary Data Sheet, SGS-Thomson Microelectronics, June 1994. http://www.cern.ch/HSI/dshs/Welcome.html

4-53 S. Haas et al., Results from the Macrame 1024 Node Switching Network, Conference on Computing in High Energy Physics '97, Berlin, Germany.

4-54 C. Clos, A Study of Non-Blocking Switching Networks, Bell Syst. Tech. J. 32, page 406–424, 1953.

4-55 S. Haas et al., published in B.Hertzberger & P. Sloot (Eds.), High-Performance Computing and Networking, Lecture Notes in Computer Science, Springer 1997, presented at HPCN Europe 1997, Vienna, Austria, April 1997.

4-56 M. Zhu et al., Realisation and Performance of IEEE 1355 DS and HS Link Based, High Speed, Low Latency Packet Switching Networks, presented at RT 97, Beaune, 22–26th September 1997.

4-57 R.W. Dobinson et al., Evaluation of the Level-2 trigger, architecture B, on the Macrame Testbed, ATLAS Internal Note, DAQ–NO–102 (1998).

4-58 B. Thooris et al., Emulation of Architecture C on MACRAME, ATLAS Internal Note, DAQ–NO–107 (1998).

4-59 The Level-2 Pilot Project, ATLAS Internal Note, DAQ–NO–118 (1998).

4-60 ATLAS Level-2 Trigger User Requirements Document, ATLAS Internal Note, DAQ–NO–79, LVL2–URD–1.00 (1997).

5 Data acquisition and event filter

5.1 Introduction

The design of the ATLAS data acquisition system must satisfy the requirements for performance imposed by the ATLAS physics goals and the LHC experimental environment. They drive the design of the ATLAS detectors and their granularity and dictate the number of electronics channels, the trigger rate and event reduction. Ultimately, the total amount of data to be acquired, formatted, moved, monitored and checked drive the trigger and DAQ architectural decisions, the technology choices on which they are based, the number of data links and the amount of processing power needed. Not all of the detector requirements are well established or fully understood. It is clear, however, that they place an unprecedented challenge on the trigger / DAQ system.

After a few years of intensive R&D we know today that:

1. The system must be based on innovative architectural solutions, classical ones could not provide the required performance.
2. At many levels the DAQ process seems to require technologies which are either not available or whose utilization we do not fully master.

The above applies to both hardware and software.

We are not, therefore, ready yet to approach the final design of the DAQ system, which anyway, given the time-scale of the LHC experimentation, would be premature. Entering the design phase too early would imply a number of risks, such as:

- The system may be obsolete before it is built and / or inadequate and unmaintainable once it is built.
- Instead of being based on a sound architectural description, design decisions may be driven by specific technologies and / or components, as has has often happened in the past.
- Over-designing of components or subsystems may occur, as we have often seen as expensive brute-force solutions in systems not adequately designed.
- Under-designing of components or subsystems may occur as well, which would limit the performance of the overall system.

The above considerations are equally valid for the DAQ, the LVL2 and the event filter (see Section 3.2). As described in Section 3.3, irrespective of the final design, the LVL2 has very demanding rate and control requirements although it is not required, at least functionally, to be part of the main flow of data. Therefore, the specifics of the LVL2 subsystem are better studied and addressed, at least to start with, independently from the DAQ and the event filter. On the other hand, the event filter is functionally an integral part of the main DAQ data-flow. In fact, while being processed by the event filter program, events are stored in the event filter processor memory, in contrast to the LVL2 which processes copies of relevant event fragments, while the event data remain in the readout buffers.

This chapter addresses specifically the data acquisition and event filter which, for the reasons just stated, have also functionally a higher coupling than the previous trigger levels. The em-

phasis of the activities in this area has been, and will be for the rest of the pre-design phase (Section 7), on two broad complementary lines:

1. Develop an architectural description of the data acquisition and event filter system as the basis for making the architectural choices which would define the elements and their relationships in a coherent overall system, operational at the required performance. Reaching a mature architectural description is the necessary prerequisite to start the work on the final design in useful time for ATLAS and yet postpone to the latest possible moment decisions on technologies and components and to allow technological growth and system evolution.
2. Pursue selected technology investigations in order to gain a real understanding of the parameters of the components available. Features of commercial busses, links, switching networks, processors, interfaces, but also operating systems, modelling and CASE tools must be well understood in order to be used at the best in the architectural description at the moment of the final system design.

From the architectural viewpoint the DAQ/EF system can be seen as made of four main subsystems: the interface to the other ATLAS subsystems, the data-flow, the back-end DAQ and the event filter.

1. The interface to the other ATLAS subsystems consists of two main areas:
    - the set of specifications which define the boundaries between the central DAQ/EF system and all the other experiment subsystems
    - the set of requirements of each subsystem for the central DAQ/EF. These are requirements such as the subsystem readout, initialization, calibration, monitoring, control, etc.
2. The data-flow is the subsystem (hardware and software) responsible for the transport, management, and monitoring of the ATLAS physics data, from the detector readout electronics to permanent storage. The data-flow subsystem is not responsible for data selection (trigger, event filter) and must be operational with or without it.
3. The back-end DAQ subsystem encompasses all the software to do with configuring, controlling and monitoring the data acquisition system, excluding anything that has to do with the functionality of the data-flow subsystem, to which it interfaces at various levels.
4. The event filter is the subsystem responsible for the last stage of event selection before permanent storage. It acts on fully built events, any selection conveniently possible with partial event analysis being the role of the level-2 trigger. The current ATLAS strategy for event filtering is to use to the maximum possible extent the offline reconstruction and analysis programs.

Rather than via a fragmentation of ad hoc investigations, architectural and technological studies in each of the above areas are better done in a system-wide context, where the validity of components and subsystems can be ascertained not only in themselves but also integrated in their system environment. We have chosen, therefore, to build a DAQ/EF prototype system supporting the full functionality from detector readout to data recording. The prototype is the basis for predesign architectural studies, technology (hardware and software) evaluations and comparisons, performance measurements, and assessments of solutions. It is also the means to study global system requirements other than the basic performance ones, such as those coming from practical and operational needs. Examples are: system initialization, partition and scalability, er-

ror detection and recovery, control, calibration and monitoring needs, software operability and maintainability.

Although it is not expected that any of the prototype physical implementations (hardware and software) will be used for the final ATLAS DAQ/EF system, work and results from the DAQ/EF prototype project are of primary relevance to the final system. Examples are:

- the detailed specification of the elements and subsystems requirements
- the study of architectural and technological solutions (hardware and software)
- the evaluation of software engineering (SE) tools for the design and development of the final system; in general, the specification of the software development environment
- the definition of the software process
- the understanding of which elements can be bought and which ones must be custom made
- in general, the acquisition of technical and organisational skills for the development of the final system.

The rest of this chapter is mostly dedicated to the description of the project, which is referred to as the DAQ/Event Filter Prototype '-1' Project, and the way it has been designed in order to address the above issues.

5.2 The DAQ/EF -1 Project

The Data Acquisition and Event Filter Prototype '-1' Project (DAQ/EF '-1') [5-1] takes its name from the definition of the project as being able to support the full functionality of the final system, but not necessarily to achieve the final performance. The prototype consists of a full 'vertical' slice of the functional data acquisition and event filter architecture outlined in the ATLAS Technical Proposal [5-2] and detailed in [5-3]. It includes all the hardware and software elements of the dataflow, its control and monitoring, as well as all the elements of a complete online system. This project follows, and in part is based upon, previous investigations of DAQ related issues for the LHC experiments [5-4]. Although aimed at providing the functionality described in the ATLAS Technical Proposal architecture, the prototype is designed to support the evaluation of various technological and architectural solutions. Issues related to the software development environment, from the operating systems to software engineering and CASE tools, are also addressed.

Started in December 1995, after approval by the ATLAS Collaboration, the project has been planned in three phases:

- pre-design studies and High Level Design (1996–97)
- detailed design and implementation (1997–98)
- system integration and exploitation (1998–99).

At the moment of writing, the project is reaching completion of the final implementation of individual components and has started the system integration process. Once the full system integration is concluded, the prototype will be the basis for a number of performance measurements and operability studies as described in the work-plans of Section 5.3 and Chapter 7.

The description and status of the prototype is given for the four subsystems introduced above, namely the detector interface (Section 5.2.1), the dataflow (Section 5.2.2), the back-end DAQ (Section 5.2.4) and the event filter (Section 5.2.3).

The work-plan for integration and exploitation of the system is given in Section 5.3 for the subsystem level and in Chapter 7 for the full system level.

5.2.1 Detector interface

5.2.1.1 Definition, purpose and goals of the detector interface

The principal aim of the Detector Interface Working Group is to collect together and understand the detector requirements on the DAQ system, trying to map them as closely as possible to the final requirements of the ATLAS experiment. There are many areas (see below) where detectors expect certain services, standards, or constraints from the DAQ system, and vice versa. The definition and clarification of these areas is vital for the successful integration of the entire experiment in the long term, and for the efficient use of the DAQ system in test-beam runs for detectors in the shorter term. Having a complete and coherent picture of the detector requirements will also highlight points which are either impossible to fulfil, or incompatible with the current global DAQ design. It will also allow us to identify areas where there are similar but conflicting requirements between different detectors.

The group, set up in September 1996 by the ATLAS DAQ community, includes a representative for each of the following ATLAS subdetectors: SCT, TRT, electromagnetic and hadronic calorimeters, muon chambers, LVL1 calo and muon and LVL2. The first stage of the work was concluded with the release of a summary document [5-5].

Recently, a new document has been circulated (in the form of a questionnaire) [5-6]to complement and complete the information already collected in [5-5]. The next step is to hold a series of meetings between members of the DAQ group and individual detector and physics groups in order to discuss the information already collected in [5-5] and the issues raised in [5-6], with a view to establishing a complete and clear picture of the interface between the detectors and the DAQ system.

5.2.1.2 Areas of work

The areas of work of the Detector Interface group can be identified in several broad and overlapping sections, the most important of which are briefly described here.

Front-end interfaces and control

Hardware and software at the ROD level are the responsibility of the individual subdetector groups. It is not currently foreseen in the DAQ Front End Interfaces document [5-7], to supply general DAQ functionality (such as error message and alarm handling, run control, etc.) in the ROD crates. However, it is important to identify the various functions which are planned to be performed at the ROD level (including local calibration and data acquisition) in order to proceed in a coherent way.

Monitoring, calibration and databases

Some of the current ideas of the DAQ community in terms of monitoring are summarized elsewhere [5-8]. A clear picture of the way each group sees its monitoring requirements at various stages (local hardware monitoring, interdetector 'efficiency' type monitoring, trigger performance and physics quality assurance) is necessary, with some emphasis on the support required in terms of DAQ. In particular, it is important to understand whether the functionality currently foreseen for the DAQ system coincides with these requirements.

Concerning the calibration, the various calibration processes for each detector need to be understood and appropriate facilities to realise them, foreseen. An initial list has been compiled in [5-5], and this now needs to be expanded and completed.

Many different databases are stated to be required in [5-5], at various levels of the detectors' systems. A considerable amount of work needs to be done to make these requirements coherent, in order to prevent an explosion in the number and types of databases used. It is particularly important to try to find as much interdetector commonality as possible for a given database functionality.

System partitioning, special non-physics trigger

Particular emphasis will be given to the software partitioning, i.e the need to calibrate and test different parts of a given detector in parallel. The ability to partition the ATLAS detector into several parallel functioning systems will enable parallel development, debugging, commissioning, etc. Some ideas from the DAQ group are available elsewhere [5-9]. Currently, the maximum number of independent partitions which may receive independent triggers in a detector system is determined by the number of available TTC stations (it is currently foreseen to have four per subdetector). However, the relative size of each of these partitions is variable (from one readout crate to the whole detector).

Special triggers will be required for specific calibration and monitoring purposes. A list of these triggers has to be established in collaboration with the trigger performance group. One also needs to understand how / where these triggers will be defined, and how they will fit into the overall ATLAS trigger scheme.

Event format, data recording, data formats

A complete and coherent list for each detector, of the number of channels to be read out, as well as the data size and format for both the next test beam scenarios and for the final ATLAS configuration is required. Much progress has been made in this area in [5-5], but more work needs to be done. A proposal for the ATLAS online raw data format from the ROBs is presented in [5-10], and it has to be compared with the ROD data format proposed by the various subdetectors, in order to eliminate any inconsistencies. Not all the data taking will be recorded centrally. It is therefore also important to quantify the amount of data which will be recorded and stored privately by the detector groups.

Run control, debugging and testing

Run control issues are relevant at several levels, in particular at the local or partition level, where a given detector will run or test independently from all other detectors or partitions, and at the global level, where the whole detector needs to be controlled coherently. The functionality of standalone DAQ systems as well as their relationship to the final system needs to be investigated.

System debugging and testing is being addressed by each system independently. However, there is a development underway in the DAQ group [5-11] which will be used to address some of the testing and debugging problems within the DAQ system, and which may be also useful at the detector level. Further investigations are necessary.

5.2.2 Dataflow

In this section we define the dataflow system [5-12] as part of the overall ATLAS DAQ/EF prototype-1 [5-1], we describe its high level design and we outline the status of the developments.

After a definition of the dataflow system and an outline of its boundaries with other parts of DAQ/EF prototype-1, an overview of the dataflow operations is given in terms of the behaviour of its three main building blocks: the readout crate (ROC), the event builder (EB) and the subfarm DAQ. Based on the above, a factorization into three subsystems, the Local DAQ (LDAQ), the DAQ-Unit (DU) and the Event Builder (EB), is defined. More information can be found at a dedicated website[1]. Definitions and abbreviations are defined in [5-13].

5.2.2.1 The dataflow system in prototype-1

The dataflow component of the ATLAS DAQ/EF prototype -1 [5-12] is responsible for moving the event data from the detector readout links to the final mass storage. It also provides event data for monitoring purposes and implements local control for the various dataflow elements. A global view of the dataflow component is shown in Figure 5-1.

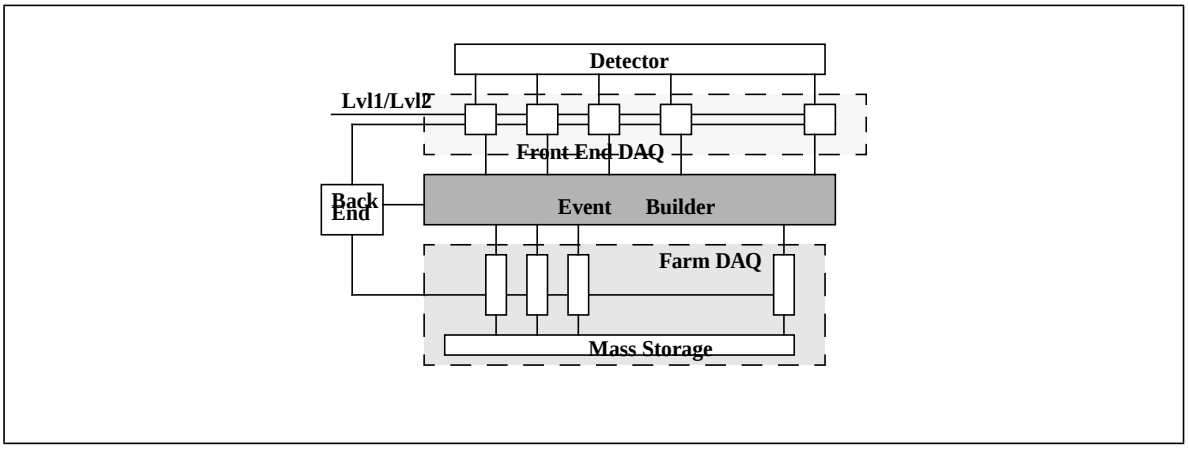


Figure 5-1 Dataflow system.

Factorization of the Dataflow system

Three main functions are provided by the dataflow. Specifically: the collection and buffering of data from the detector (the front-end DAQ), the merging of fragments into full events (the event builder) and the interaction with the event filter (EF) (the farm DAQ). In addition to the event building function (which we see as a dataflow building block), we are lead to the identification in the system of two modular functions (or building blocks):

1. http://atddoc.cern.ch/Atlas/FrontEnd/Welcome.html

The readout crate (ROC): the segmentation of the detector readout into readout driver (ROD) modules suggests how the front-end DAQ should be organized into a number of modular, independent elements (ROCs) each supporting the readout from one or more RODs and having one or more connections to the event builder.

The subfarm DAQ: the event filter farm may also be viewed as being (logically or physically) segmented into independent modular elements, each connected to one of the event builder outputs and treating one or more events. Such a modular element is called a subfarm, its data acquisition part is the subfarm DAQ.

The term modular here refers to the fact that a ROC (or a subfarm) works concurrently and independently with respect to any other ROC (subfarm) in the system. Each modular element connects independently to the event builder. It is indeed the role of the event builder to combine the various 'modules' into a coherent dataflow system.

5.2.2.2 Relevance to the final system

The design and development of the dataflow component is, as for the overall ATLAS DAQ Prototype -1, an essential step towards the design of the final ATLAS trigger / DAQ system. We outline the relevance to the final system of the work in the dataflow area in the following points:

- Requirements: a thorough understanding of the user and system requirements is a prerequisite for the successful design of the final system. Given the unprecedented level of size, complexity, and performance target of the final ATLAS dataflow [5-2], the prototyping work is essential to address, eventually in the real life conditions of a test beam, various aspects of the dataflow process. As an example we mention the functionality required to perform monitoring and calibration of the detector, the requirements related to a particular subsystem (such as the event builder) or components (such as the readout link), the requirements related to the integration with other DAQ or detector elements.
- Different architectures: an objective of the prototype is also that of addressing issues which are relevant to different Trigger / DAQ architectures. Examples are the ROC data collection studies and error handling.
- High-level designs: a fundamental step in the process of the development of the dataflow for the ATLAS DAQ prototype-1 project is that of defining high-level designs for both the overall dataflow and its main components. Contrary to the developed prototype, which will not be reused for the final system, we expect that the high-level designs will provide a starting point for the design of the final system.
- Performance studies: we intend to address performance issues from two points of view. On the one hand we wish to study the performance of the architectural and implementation options addressed by the prototype. On the other hand, although it is not the goal of the ATLAS DAQ prototype -1 project to reach the performance required by the final ATLAS T / DAQ system, we aim at being close to the final performance target locally, for example around the ROB module and the event builder control. Where the required performance is not reached, we want to understand where the limitations come from: architecture, implementation, technology, interplay of the latter.
- Global issues: prior to launching into the final design, it is necessary to address a number of topics relevant to the whole dataflow. The system development process, i.e. the set of procedures to be used to design and develop the dataflow subsystem, and the global event builder, in particular concerning issues related to its size (in terms of number of in-

put and outputs), are two example of topics to which the prototype work will provide a useful background.

5.2.2.3 Boundaries with other parts of the system

The dataflow has boundaries with other systems which are part of the DAQ/EF -1, such as the event filter, or not a part, such as the triggers. This is sketched in the context diagram of Figure 5-2 where we have also indicated the relevant issues for each external system.

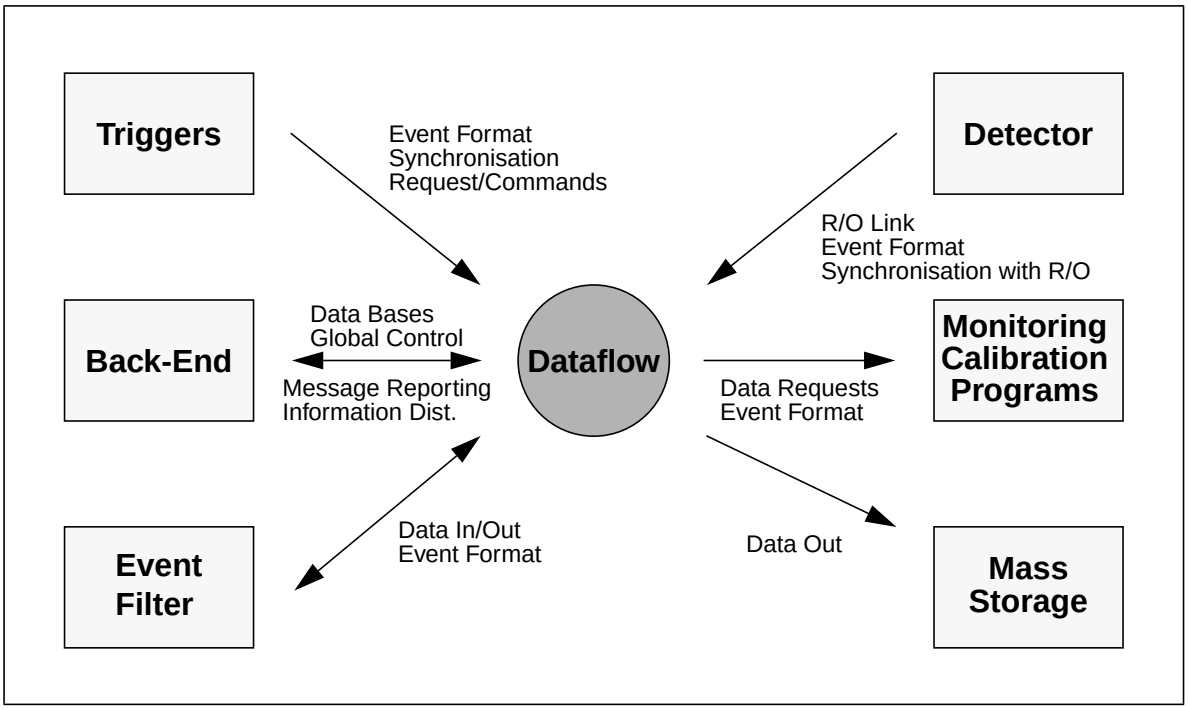


Figure 5-2 Dataflow context diagram.

The detectors: the boundary between the dataflow and detector systems is defined in [5-7] to be at the level of the readout link (ROL) connecting the ROD with the readout crate. The format of the data transferred over the ROL and the synchronization (including issues such as back pressure and flow control) between this latter and the DAQ are two issues relevant to this boundary.

The triggers: the dataflow, and in particular the ROC, exchange control commands (e.g. the level-1 accept for a certain event) and data (e.g. Region Of Interest (ROI) information) with the trigger systems. The issues are in this case related to which trigger(s) the dataflow communicates with and to the format of the data and control commands exchanged.

The back-end: the dataflow and the back-end functions (e.g. the overall run control) ought to integrate into a unique system. Integration between dataflow and back-end is performed at the level of an element, part of each building block (ROC, subfarm DAQ and event builder), which provides those dataflow functions which are not specifically concerned with the main flow of the event data.

The event filter: the event filter interacts with the subfarm DAQ by receiving full events, and outputs new events, such as reformatted accepted events, to the subfarm DAQ. How data (raw and

filtered events) are exchanged with the event filter and in what format are the issues related to the integration of the dataflow with the event filter.

The mass storage: the subfarm DAQ is also responsible for recording data from the event filter onto permanent storage . The main issues here are related to the format of the data, as produced by the event filter, and the characteristics of the mass storage system. A suitable software interface, to output full events, is the place where dataflow and mass storage integrate.

The monitoring programs: dataflow and monitoring programs interact in terms of transactions (requests / replies) for event data. The boundary is specified as a software interface to retrieve an event, with some predefined characteristics (e.g. its trigger type), from the dataflow. The format of the event data is an issue.

5.2.2.4 Dataflow high-level design

Starting from the top level view of the dataflow as defined in section 5.2.2.1, we refine the dataflow high-level design by defining the three main dataflow functional elements (ROC, event builder and subfarm DAQ) and discussing their integration into the dataflow system.

The readout crate (ROC)

The ROC (see Figure 5-3 for a functional sketch) is the modular element responsible for handling the movement of data, originating from a group of RODs, between the ROL and the event builder. It provides the following main functions,.

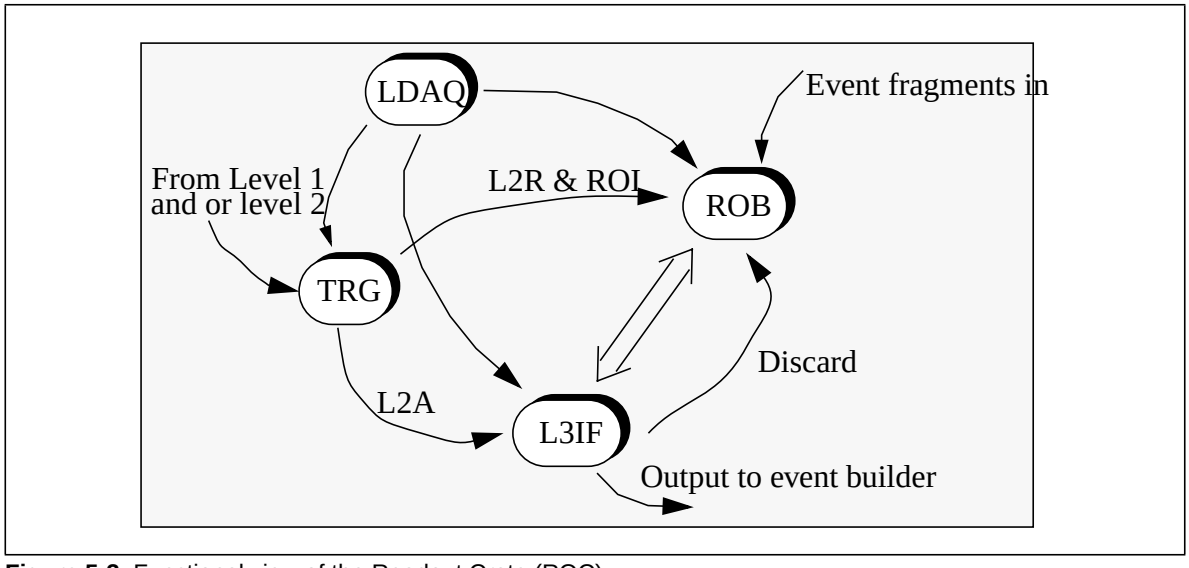


Figure 5-3 Functional view of the Readout Crate (ROC).

1. Detector readout, buffering and data distribution to other dataflow elements in the crate. This function is provided by the ROB (readout buffer) element in Figure 5-3. The ROB is the integration element between the dataflow and the detectors.
2. The control of the flow of data within the crate. For example, event fragments buffered in the ROBs have to be discarded or moved to the EB according to the response provided by a data reduction device such as the LVL2 trigger system. The TRG (trigger) element pro-

vides the integration element between triggers and dataflow and therefore controls the flow of data within the crate..

3. Fragments of accepted events are moved from the ROB memories and merged into a ‘crate fragment’ (consisting of all the elementary fragments from the individual ROBs). These are buffered and then sent to the event builder. This function is defined as ‘data collection’ and it is implemented by the EBIF (interface to the event builder) module.
4. Other ancillary functions must also be provided locally in the crate: the control of the crate, the handling of errors and support for event monitoring (the capability for external application programs to receive, upon request, a sample of the data flowing in the crate). An interface point to back-end DAQ functions is also needed. The component which is assigned to these ancillary tasks is defined as the LocalDAQ (LDAQ). The LDAQ also plays the role of integration element with both the back-end and the monitoring programs.

An important role is played by the *intracrate links* (e.g. the one connecting the ROBs and the EBIF to support the data collection function) and the related communication protocols (e.g. the data collection protocol).

The event builder

The event builder (EB) merges the event fragments from the ROC (or ROB, in the case of a direct connection between ROB and event builder) having the same event ID[1] into a complete event at a destination.

The event builder has boundaries (see Figure 5-4) withthe ROC, at the level of the EBIF, and with the subfarms, at the level of the switch to farm interface (defined below). In both cases a buffer for ROC fragments (EBIF) and full events (SFI) is the physical boundary between the event builder and the other parts of the dataflow. The event builder further connects to the back-end DAQ and the LVL2 trigger.

While different technologies might be used to implement the actual transfer and merging of data into full events, performance considerations indicate that this will be implemented by means of a switching network, allowing concurrent merging of events. It is one of the objectives of DAQ/EF prototype -1 to try out different commercial technologies for the event builder switching network. ATM, FibreChannel and Switched Ethernet are examples of candidate technologies.

To fulfil the above objectives, the event builder itself is partitioned into two layers:

A *technology-independent layer,* which implements the event building protocol (e.g. it determines which destination subfarm will receive a given event). This consists of the Dataflow Manager (DFM) and sources (Src) and destinations (Dst) depicted in Figure 5-5. The DFM implements the high-level event building protocol while the sources and destinations are responsible for sending/receiving the event fragments from/to the buffers delimiting the event builder via the technology-dependent layer.

A *technology-dependent layer* which interfaces, in a common way, the technology-independent components to the features of the switching hardware. It hides the technology-specific features

1. By event ID we mean the value (over 24 bits) uniquely identifying an event provided by the level-1 trigger [5-2].

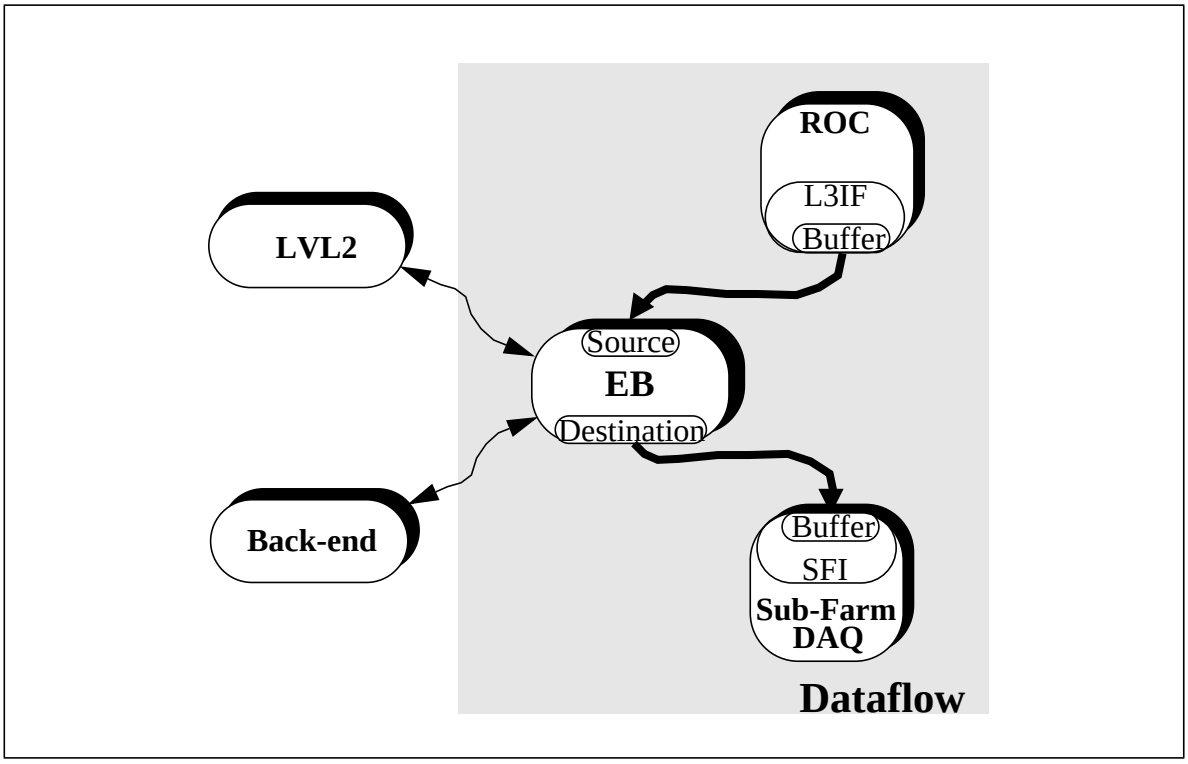


Figure 5-4 Event builder and its boundaries.

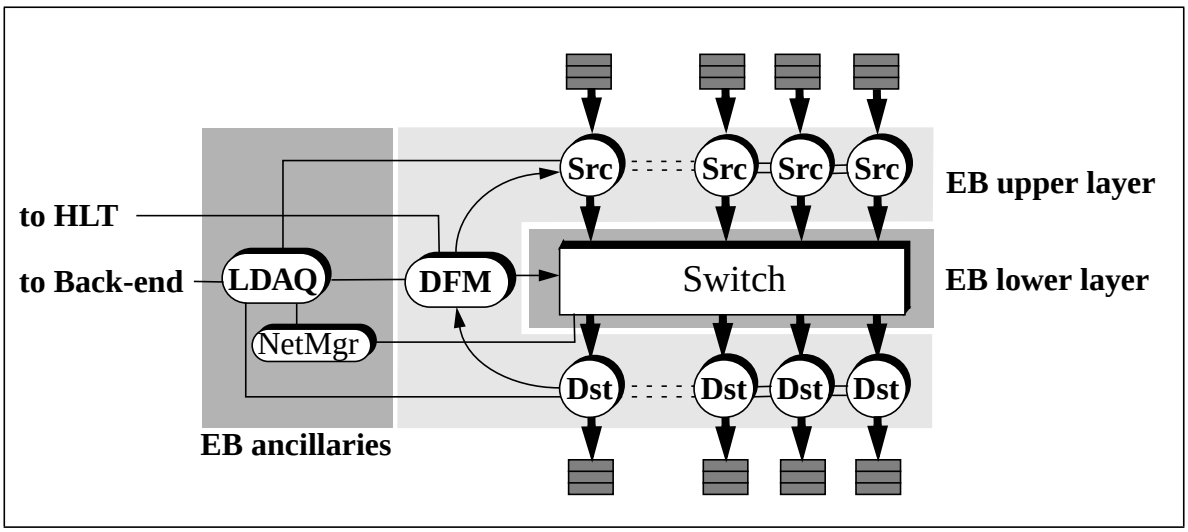


Figure 5-5 Event builder elements.

required for sending and receiving of data and control messages from the technology-independent layer.

Ancillary functions, such as local control and monitoring of the behaviour of the event builder as well as interfacing to the back-end DAQ system, are also relevant. To this end the event builder has to be complemented with an LDAQ subsystem. For the details of the LDAQ subsystem see section 5.2.2.5. The Network Manager (NetMgr) is responsible for managing the switch and is itself put under the control of the LDAQ subsystem.

The subfarm DAQ

The subfarm DAQ (Figure 5-6) receives full events from the EB, buffers and sends the events to the EF and records events, as produced by the EF. The element between the event builder and the event filter is called the Switch to Farm Interface (SFI). The role of the SFI is to run the event builder destination process, buffer full events and send full events to the event filter. The element sitting between the event filter and the mass storage is defined as the Sub Farm Output (SFO). The role of the SFO is to collect events as 'produced' by the event filter, buffer them and then record the resulting events onto the (external) mass storage system.

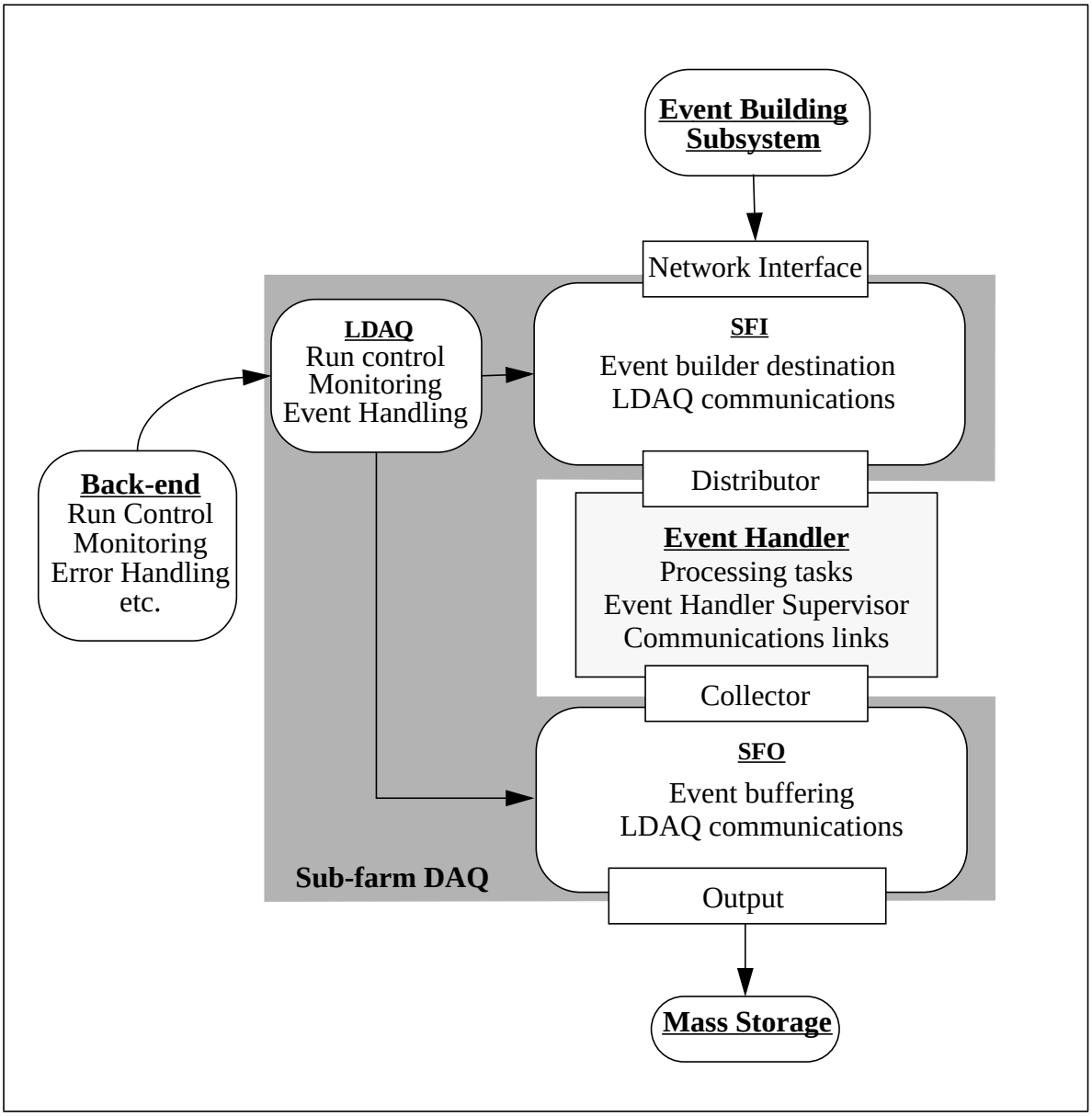


Figure 5-6 Subfarm DAQ.

Support for functions such as local control and event monitoring has to be provided by the subfarm DAQ as well. The subfarm DAQ also requires a LDAQ.

Although there are different performance requirements between the dataflow in the readout crate and in the subfarm, they are functionally very close. One common aspect is that, as far as the dataflow is concerned, individual crates as well as individual subfarms, are independent modular elements. Another aspect is that of the presence in the subfarm DAQ of elements providing a combination of event data input/output and buffering (the SFI and SFO elements). These considerations suggest a common approach, at least in the high-level design, between the DAQ support in the ROC and in the subfarm.

Common issues

Partitioning and event format are issues common throughout the DAQ/EF prototype -1 and, in particular, across the dataflow system. Here we define them and indicate where the dataflow provides support for their implementation.

Dataflow partitioning

Partitioning is defined as the capability of running multiple, independent, concurrent DAQ systems with full functionality on subsets of the detector. As such it is a requirement for the overall DAQ and, in particular, for the dataflow.

A dataflow partition is therefore a subset of the dataflow hardware and software which provides full DAQ functionality. This means that a partition has the capability of reading out, recording and monitoring/calibrating a subset of the detector.

We currently define the smallest detector partition (the granularity of the partition) as the readout of one single ROC. We may have therefore two basic types of partitions:

- A single (stand-alone) ROC: a subset of the detector (corresponding to those RODs ending up into the specific ROC) is readout and the data are output from the EBIF to some form of mass storage, calibration task, etc. In this case there is no intervention of any other part of the dataflow (event builder or farm).
- A vertical slice of the dataflow, including one or more ROCs, the event builder and one or more subfarms. This is expected to be the ‘standard’ way of partitioning the dataflow.

In both cases defined above we remark that:

- ROCs and subfarms are partition blind, in the sense that they do not have to perform anything special to run in a particular partition and nor do they actively participate in the process of partitioning the dataflow. They are passive elements as far as partitioning is concerned. The single, stand-alone ROC is just configured as using a different ROC output task.
- The event builder is the subsystem which implements the partitioning of the dataflow. At the level of the event builder, dataflow partitions are nonintersecting sets of sources and destinations (i.e. groups of ROCs and subfarms). The event builder is physically shared by all the running partitions while the dataflow manager implements the association between event builder sources and destinations.

Event format

The structure of the data [5-10], at various stages within the DAQ/EF prototype -1 in general (and the dataflow in particular) is defined by the event format. The event format allows elements of the dataflow, as well as processing tasks, to access part of the data without resorting to other resources (such as a database). It defines how data of a ROB, crate fragment, or subdetec-

tor may be directly accessed by processing tasks. Moreover it defines additional data that are added to the detector data (such as event tags) by elements of the dataflow to allow processing tasks to quickly identify, for example, the type and origin of the event.

Formatting of the event, in the sense of adding information (header, pointers, etc.), is supported in the dataflow by any element which handles the data (ROB, EBIF, SFI).

Integration

Internal dataflow integration

Integration between ROCs, event builder and subfarm DAQs is realised at the level of the boundary between subsystems.

The EBIF element realises the interface between ROC and event builder insofar as it provides data collection at the ROC level (the last dataflow step in the ROC) and performs the source task of the event builder. The EBIF buffer is both the boundary between ROC and event builder as well as the place were integration happens.

The SFI element realises the interface between the event builder and subfarm DAQ. It plays a role analogous to that of the EBIF for the ROC and the event builder. It performs the event builder destination task and interfaces to the distribution of events to the event filter. Again the SFI buffer acts as both the boundary and the integration point between event builder and subfarm DAQ (see Figure 5-6).

External dataflow integration

Integration with systems external to the dataflow happens at the following points, as depicted in Figure 5-2:

- The ROB module supports the integration of the ROC with the detectors (i.e. the readout link from the detector ROD).
- The TRG module integrates the ROC with the trigger systems.
- The LDAQ integrates all building blocks with both the back-end and the monitoring and calibration programs.
- The SFI and SFO support the integration of the subfarm DAQ with the event filter.
- The SFO integrates a subfarm with the mass storage system.

5.2.2.5 Dataflow development

Preliminary work

The first phase of the project, one of predesign analysis and early prototypes both in the area of the readout crate and the event builder, was completed in 1996. Extensive documentation in terms of summary documents, workplans and technical notes is available in [5-14]to which reference should be made for detailed results. Here we discuss some relevant topics.

Predesign analysis of the readout crate and the subfarm DAQ has resulted in the identification of the concept of an I/O module: the combination of a memory, a processor and a number of I/O channels. For example, a ROB is an instance of an I/O module with one high-speed input

channel (from the detector) and at least two output channels (one to the EBIF and the second to the LDAQ); the EBIF, TRG and SFI are other instances of an I/O module.

A number of initial selections of hardware elements have been made in the area of the readout crate.

- VMEbus is used as the crate integration bus as well as the initial implementation of the intracrate links; evaluations and studies to select other technologies for intracrate links, such as the Raceway Interlink and the PVIC systems, are under way.
- PCI has been selected as the I/O integration bus within an I/O module. The PMC format is the preferred one for I/O interfaces.
- I/O modules are currently implemented by VMEbus, PowerPC based processors with two (or more) PMC sites.

Hardware libraries (e.g. interfaces to the module hardware features such as the PCI bus) and generic (hardware and operating system independent) packages (such as buffer management and I/O scheduling) have been developed to support the generic I/O module element.

Some studies of operating systems for real-time applications including the possibility of using PCs and/or the WindowsNT operating system, in the area of the LDAQ, have also been performed [5-15].

Intracrate communication protocols have been defined and prototyped to support both the local DAQ (e.g. control transactions between LDAQ and a ROB) and the dataflow (e.g. the data control messages exchanged between the TRG module and the EBIF).

A local control and a monitoring subsystem have also been prototyped. The event monitoring subsystem has been designed to decouple the sampling of event data (from the dataflow) from the request of events (from analysis programs). This decoupling is achieved via a database which is periodically updated to contain a consistent snapshot of the data flowing in the crate. The same LDAQ functionality is applicable to the subfarm DAQ.

Based on the above hardware choices and software basis, a readout crate prototype has been built. It covers the full internal functionality of the crate and is being used to test design ideas and to measure performance in certain areas. For example, the TRG module has been shown to be capable of performing part of its task (input from PMCs, some calculation on the input and then re-sending the result to one I/O module in the crate) at a rate compatible with the final maximum ATLAS requirement of 100 kHz (LVL1 rate).

In the event builder area we have addressed some of the issues related to the general event builder process as well as performing technology evaluations [5-16].

Design and development

The definition of an appropriate System Development Process is one of the objectives of the DAQ -1 project.

We have adopted a flexible approach to the development of the dataflow system based on a number of phases. Prototypes produced during a phase may be carried forward to the next one. Tools, such as emulators for modular testing and code management schemes, are used when they exist or developed if necessary.

The following phases have been defined:

- Predesign phase. This has focused on problem understanding, based on the ATLAS technical proposal architecture, and on the selection of candidate technologies for evaluations. To evaluate technologies an initial high-level design for the ROC was done and an overall document summarizing the predesign phase was produced [5-14].
- Evaluation phase. The objective was that of becoming familiar with the selected technologies by developing, based on the high-level design of the previous phase, a prototype of the ROC. An integrated prototype ROC was produced [5-17].
- High-level design phase. This has capitalized on the experience of the prototypes to refine the existing high-level designs of the ROC and its components [5-18][5-19]. High-level designs, and corresponding prototypes, have been produced from scratch where not existing [5-20] [5-21]. An appropriate template for the high-level design documents was defined, so as to have uniformity across high-level design documents.
- Detailed design phase. A detailed design, for which the appropriate document template was defined, has been produced for some of the components, when this was considered useful.
- Implementation. Capitalizing on the above phases, the prototypes evolve into an implementation of the dataflow components. First at the level of elements, such as a ROC DAQ-Unit, and then at the level of a ROC, event builder and subfarm DAQ capable of running in stand-alone mode (i.e. with emulated input and output).
- System integration. The last phase integrates the basic dataflow components, ROC, event builder and subfarm DAQ, into a running, stand-alone dataflow subsystem.

Subsystems

The functional overview of the dataflow as described in Section 5.2.2.4 suggests a factorization of the dataflow system into three subsystems, each fulfilling a function: the *movement* of the data within a modular unit, the *merging* of event fragments, and all the functions *unrelated* to the movement of the data. We call them, respectively, the *DAQ-unit*, the *Event Builder* and the *localDAQ* subsystem.

The above factorization is related to the design of the dataflow, it aims at exploiting functional commonalities across the system. We also expect that part of this factorization will be carried forward to the implementation phase: for example localDAQ functions are candidates for common implementation across the dataflow system.

The DAQ unit

High-level design

The main flow of data into, within, and out of the DAQ unit is handled by instances of an I/O Module (IOM), more specifically the: TRG, ROB EBIF, SFI and SFO. The IOM performs standard I/O functions, e.g. data input, data output, data buffering. In addition, different instances of IOMs exchange data control messages to ensure the correct functionality within the DAQ unit. Subsequently, IOMs are driven by external or internal stimuli and associated to a stimulus is a well-defined task (a set of actions) e.*g.* move data from the buffer to an output link.

The functionality common to these different instances has been factorized out to form an element in its own right. This element is called the generic I/O Module (g-IOM), it defines the structure of the IOMs, the skeleton, and by definition provides the functionality common to

more than one instance of an IOM. The different instances of an IOM consist of a g-IOM and an instance-specific element.

Common functions to be provided by the g-IOM have been, initially, identified as task scheduling, event management, inter-IOM message passing, software support for specific hardware, local parameter buffer, event sampling (for monitoring purposes), and run control. An initial high-level design of the g-IOM is described elsewhere [5-18].

Tasks are self contained and associated to a stimulus. This allows them to be implemented within a single process, as a single process or as a thread of a single process. In addition it allows the tasks of an IOM to be distributed over the processors of a multiprocessor platform.

The readout crate DAQ unit

The main flow of data within the ROC is handled by three IOMs: the TRG, the EBIF, and the ROB. The TRG receives and buffers data control messages from the level 1, and/or the level-2 trigger systems. The EBIF builds crate fragments and makes them available to the event building subsystem. The ROB receives and buffers ROB fragments. To achieve the required DAQ-unit functionality these IOMs exchange data control messages and ROB fragments between themselves and with other (sub)systems. An initial high-level design of the readout crate DAQ unit is given in [5-19].

For completeness, the relationship between the IOMs and the data control messages exchanged between them are shown in Figure 5-7. The TRG sends dataflow control messages to the ROBs and EBIF. In the former case the messages are of two types: those that inform the ROBs to reject one or more ROB fragments and those that inform the ROBs to forward data to the LVL2 trigger system. The message sent to the EBIF, the LVL2 accept, leads to the task of data collection. On completion of the latter, a data control message is sent to the ROBs from the EBIF. This message causes the ROBs to discard the ROB fragment which was subject to the data collection process. How much collapsing depends on the level of modularity required, technology, and performance. These issues will be studied in DAQ-1.

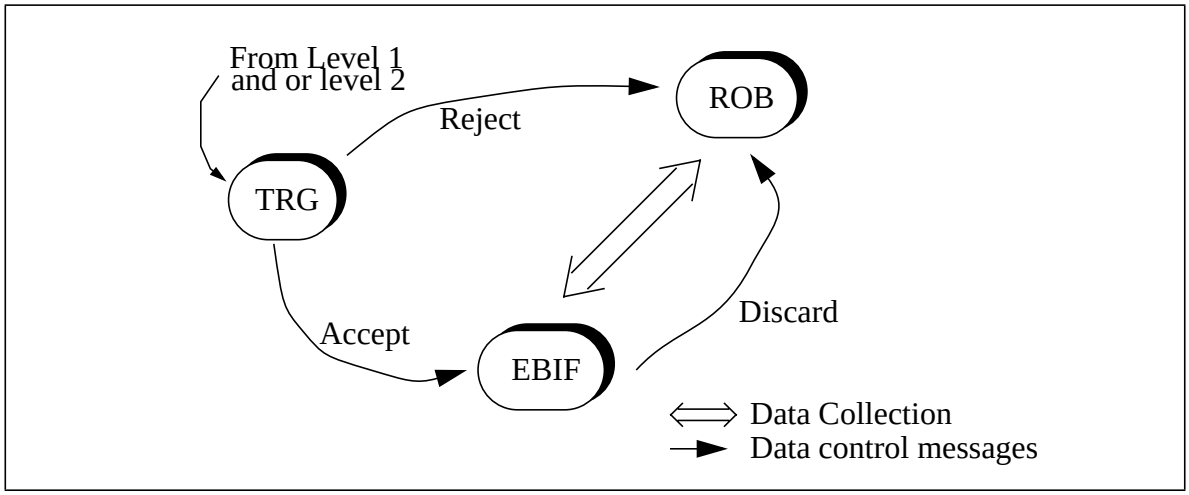


Figure 5-7 Relationship between the IOMs.

Logical modules and their functions have been identified. An implementation could use a physical TRG and a physical EBIF module. Alternatively some of the functions may be collapsed together. For example each ROB may connect directly to the event builder, in this case the logical

EBIF could be collapsed with the logical ROB to form a different physical implementation of the ROB, see Figure 5-8a. The TRG module may also be collapsed into the ROB enabling a ROB to be connected directly to the trigger, see Figure 5-8b.

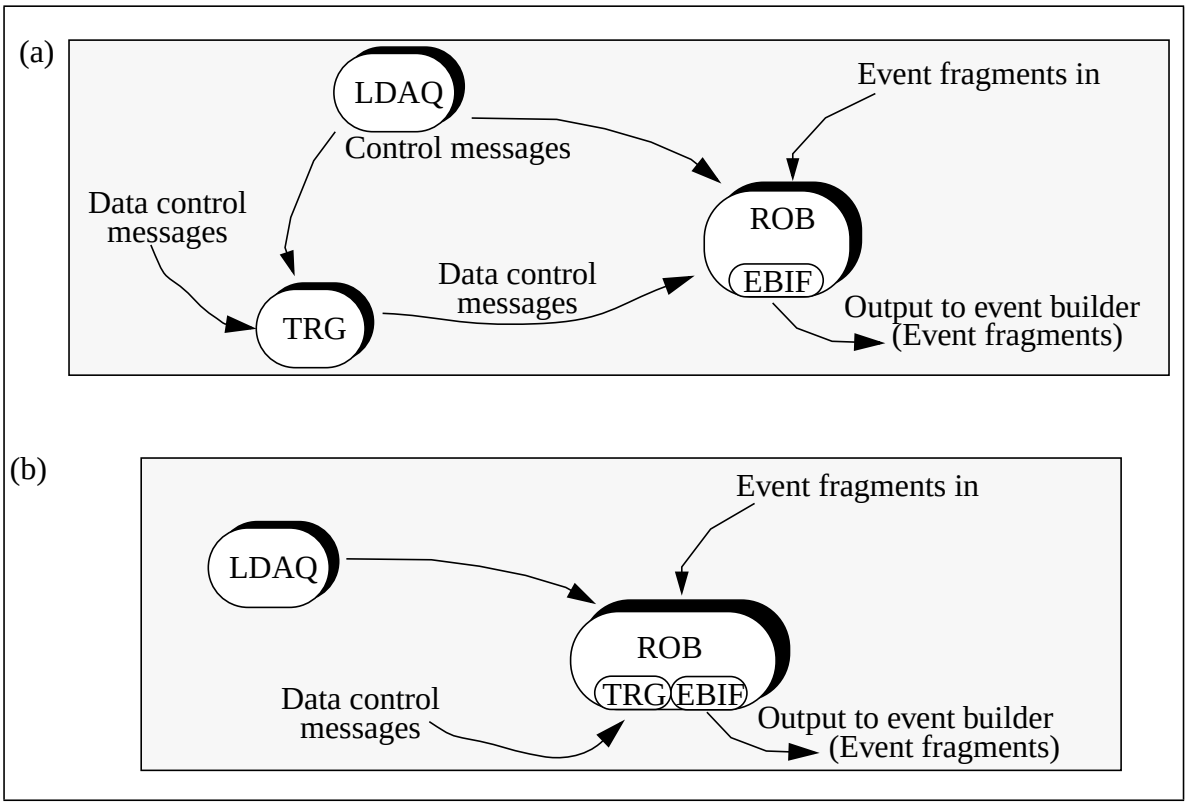


Figure 5-8 (a) EBIF function collapsed into the ROB element and (b) TRG and EBIF functions collapsed into the ROB element.

The trigger module

The TRG receives and buffers data control messages from the trigger systems. According to their type, the messages are sent to the ROBs or EBIF. The TRG also provides a mechanism whereby messages may be stacked according to their type. The TRG is structured around three tasks:

- *Input*: This task inputs and buffers the data control messages from the level-1 and / or level-2 trigger systems. The boundary with the trigger system is an interface which is managed by this task.
- *Processing*: This task processes the data control messages placed in a buffer by the input task. The data control messages are identified as those which must be sent to the ROBs or those which must be sent to the EBIF.
- *Communications with the LDAQ*: This is the task which integrates the TRG with the LDAQ and is discussed elsewhere [5-22].

The event builder interface

The EBIF, via the data collection component, builds the crate fragment into an internal buffer. These fragments are subsequently sent to the event building subsystem. The building of a crate

fragment is started via the reception of a data control message, e.g. level-2 accept, from the TRG. The EBIF is structured around four tasks:

- *Input*: This task receives data control messages from the TRG indicating that an event has been accepted by the level-2 trigger system (or the level-1 system in the absence of a level-2 system). The event identifier is decoded from the message and buffered.
- *Data collection*: This task performs the collection of ROB fragments from all ROBs in a ROC and performs any event formatting. The event identifier of the ROB fragments to be collected is accessed from a buffer filled by the Input task. Data collection may proceed based on a shared memory model between the EBIF and the ROBs or via message passing between the EBIF and the ROBs.
- *Source*: This task is responsible for the output from the ROC of crate fragments. Alternatively, in the absence of a EBIF this task becomes a ROB task and is responsible for the output of ROB fragments. This element is discussed elsewhere in the context of the event building subsystem.
- *Communications with the LDAQ*: This is the task which integrates the TRG with the LDAQ and is discussed elsewhere [5-22].

The readout buffer

The ROB inputs and buffers ROB fragments from the readout link. In addition, the ROB fragments must be copied to a high-level trigger system, accessed via the data collection subcomponent, removed from the buffering system, and sampled by to the LDAQ for monitoring purposes. The buffer management scheme is provided by the generic IOM. The ROB is structured around five tasks:

- *Input*: This task receives and buffers the ROB fragments coming from the ROL.
- *Communications with the TRG*: This task receives and processes data control messages from the TRG using the message passing functionality of the g-IOM. Two types of messages are received from the TRG: a type indicating that the ROB fragments must be forwarded to the level-2 trigger system, and a type indicating that ROB fragments must be removed from the buffer.
- *Communications with the EBIF*: This task receives a data control message, from the EBIF, whose type indicates that a ROB fragment must be discarded from the buffer. The data control message is received using the message passing functionality of the g-IOM.
- *Output*: This task is responsible for forwarding of ROB fragments to the level-2 trigger system.
- *Communications with the LDAQ*: This is the task which integrates the ROB with the LDAQ and is discussed elsewhere [5-22].

The input task must deal with a high input frequency and a large input data size. As such, it has been recognised that to meet performance requirements this task may need to be implemented on a separate processor and / or in custom hardware [5-23]. These activities, design and integration, are also being addressed within the design of the ROB.

ROBin Design

The ROBin module is designed to be self-contained with a fully-featured buffer manager integrated on-board. The buffer manager can be disabled and the buffer hardware accessed from an external processor if necessary to test alternative buffer manager algorithms. The current ver-

sion incorporates two standard protocols to fit in with the other trigger/DAQ prototyping work: S-Link for front-end input and PCI for all other communication, including data output and message transfer. Figure 5-9 shows a block diagram of this ROBin design. The board will be available in two formats: a PCI card that can plug directly into a PCI backplane slot, and a PMC format card for use with a baseboard, e.g. a RIO2 acting as ROB controller.

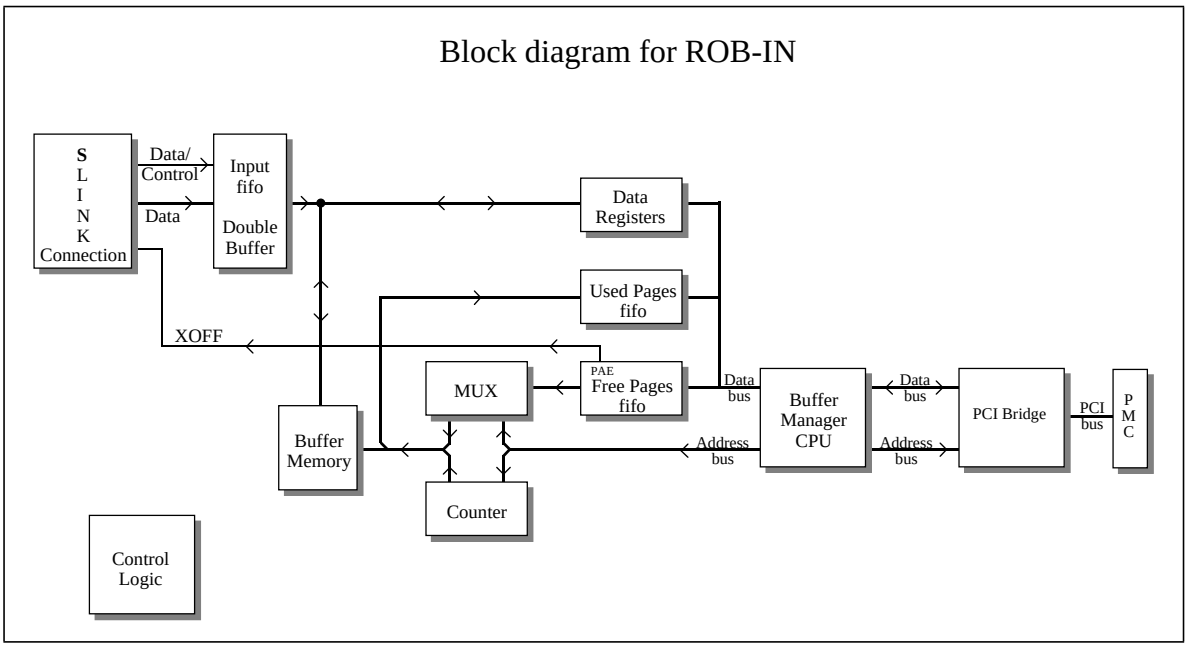


Figure 5-9 Block-diagram of the ROBin.

This board is a natural development of an earlier prototype ROB used for various LVL2 tests and demonstrations, taking note of the various ‘standards’ that are now emerging and conformance with the ROB URD.

Event-fragment data is received on an S-LINK LDC interface connection from a ROD source at an average rate of up to 100 MBytes/sec and written directly into pages of buffer memory using addresses taken from a free-page FIFO. The length of pages is adjustable in the hardware logic. At the end of each page, or at the end of each event fragment, the last address used is written to a used-page FIFO along with status bits; one of these is used to indicate the last page of the event fragment. The address logic then reads the next value in the free-page FIFO as a base for addressing the subsequent page. Under certain circumstances an XOFF signal back to the ROD is generated on the S-LINK interface in order to stem the flow of data. This occurs automatically whenever the number of entries in the free-page FIFO drops below a programmable threshold, or when the ROB-in is powered up or reset. XOFF is also asserted when the input FIFO reaches the half-full state.

The buffer manager: services the free-page and used-page FIFOs; indexes event-fragments; receives event-fragment-data requests (for RoI data, monitoring data and event filter data) and provides the requested data if possible; receives event release messages and make the relevant data pages available for re-use (by putting them in the free-page FIFO); monitors, records and reports error conditions; resets the buffer hardware and carries out appropriate test procedures on request.

Apart from the front-end data input, all communication takes place in PCI address space. Communication with the on-board processor, including the transfer of data destined for LVL2 and

event filter, is achieved through the use of circular buffers (software FIFOs) at user-definable locations.

The hardware of the ROBin-RP consists of an i960 RP processor, 1 Mbyte fragment buffer memory, the free-page and used-page FIFOs, MACH-5 PLD control logic, plus program and boot memory as shown in Figure 5-10.

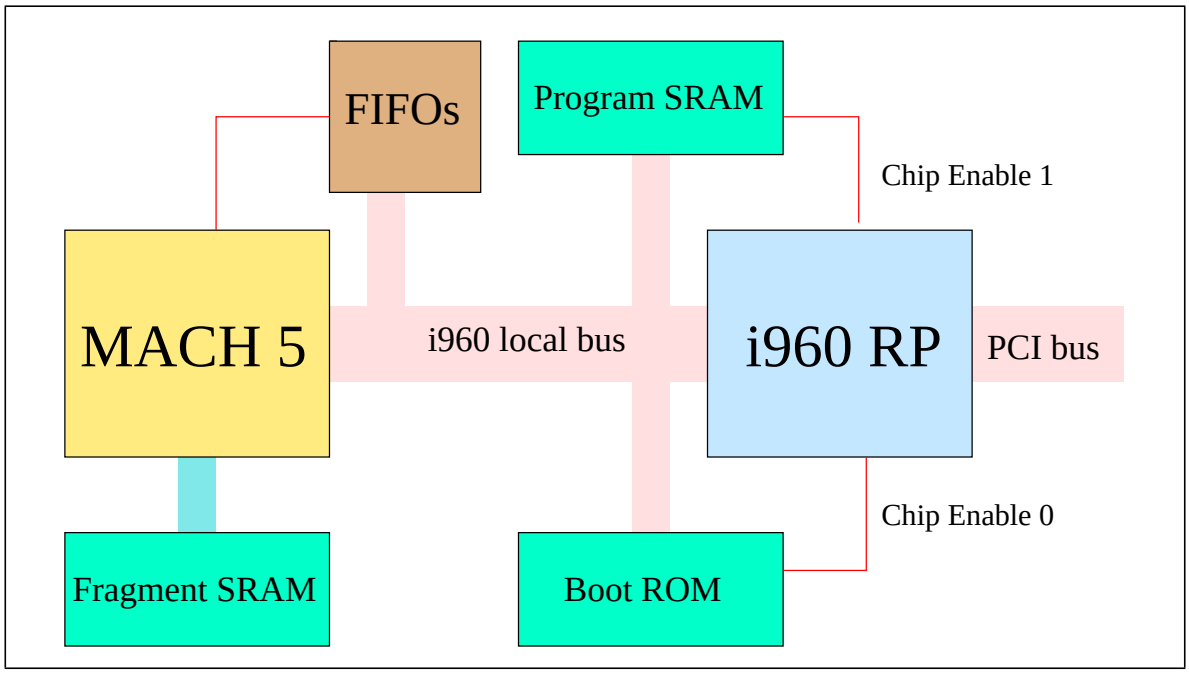


Figure 5-10 Hardware of the ROB-in RP.

Data arrive at the front-end input and are routed to the fragment buffer under control of the MACH. The fragment memory is organised as 1024 pages of 256 32-bit words (1024 bytes). The MACH determines which page of the buffer to write to by loading the upper bits of its address generator from the free-page FIFO. After writing the last word of an event fragment, or the last word of a page (whichever comes first), the address bits are written into the used-page FIFO along with some status information. It is the responsibility of the buffer management software, whether running on the on-board i960 RP or an external processor, to keep the free-page FIFO supplied with the addresses of free pages.

The buffer management software can build an index of which data are stored in the used pages based on the information returned in the used-page FIFO and in the fragment memory. For the input logic to allocate correctly a new page for each event fragment, data must be framed with the beginning-of-fragment and end-of-fragment control words as described in [5-24]. No other words are treated as special by the hardware. The buffer management software must look in the fragment memory to find event-identification information.

The subfarm DAQ crate

The main flow of data within the subfarm DAQ [5-25] is handled by two IOMs: the SFO and SFI. The SFI receives and buffers crate or ROB fragments from the event building subsystem. In addition, it performs the final tasks of event building (fragment ordering) and performs any event formatting. Subsequently, complete events are output to the event filter. In many respects the SFI is very similar in functionality to the EBIF and this similarity is exploited in its design

phase. The SFO receives event data from the output of the event filter and forwards them to mass storage.

An implementation could collapse the SFO and SFI functionality onto a single physical module. In addition, the design also allows the SFI to send events directly to the SFO in the absence of an event filter.

The SFI

The SFI is structured around three tasks:

- *Destination*: This task receives ROB or crate fragments from the event building network. It also performs the final tasks associated with event building: fragment reordering and event formatting.
- *Output*: This task performs the output of complete events to the event filter.
- *Communications with the LDAQ*: This is the task which integrates the SFI with the LDAQ and is discussed elsewhere [5-22].

The SFO

The SFO is structured around four tasks:

- *Input*: receiving and buffering complete events from the event filter.
- *Output*: output of complete events to mass storage.
- *Communications with the LDAQ*: integrating the TRG with the LDAQ and is discussed elsewhere [5-22].

DAQ unit status

An initial implementation of the ROC DAQ unit has been made based on the prototype done during the high-level design described in the previous sections, specifically the g-IOM and instance specific elements. The implementing hardware is VMEbus-based single-board computers with the following features: a single CPU, a single PCI bus for system I/O and the capability to implement two or more PMC slots.

An implementation of the IOMs on today's commercially available hardware limits the achievable performance. Although performance is not the primary concern within DAQ prototype -1, it is still considered an issue. With this in mind the current implementation of the IOMs are not multiprocessing or multithreaded and are not interrupt-driven. This has the bonus of reducing the dependence on operating system functionality and the overheads in context switching. Subsequently, the IOMs are based on the sequential execution of their tasks.

The g-IOM provides: the basic framework of the IOM, the task scheduler, the inter-IOM message passing, an event manager, local parameter buffer, specific hardware-related software packages, and the task to communicate with the LDAQ. The performance, in terms of the achieved functionality and throughput of this initial implementation is described elsewhere [5-17].

Modelling

To study rate-limiting factors and bottlenecks, a model ROC has been simulated [5-26] within the discrete event domain of the Ptolemy framework [5-27]. The model is sketched in

Figure 5-11. The external blocks (*Level1,2,3*) are simplified. *Level1* distributes ROB-fragment in parallel and notifies TRG and *Level2* at each new event. For each tenth event, *Level1* sends an RoI message to TRG for redistribution to all ROBs. Each ROB has to send the same message to *Level2*. Once all fragments have been received and the processing time has elapsed, *Level2* sends a L2decision to TRG. On a L2accept (every hundredth trigger), TRG will inform EBIF, which will then collect the data from the ROBs into a ROC fragment. As soon as EBIF has passed this fragment to *Level3*, it will notify all ROBs to clear the corresponding buffers. On a L2reject, TRG will collect the identifiers of the rejected events into a list and distribute this list to all ROBs on the ROC (to release the buffer space) whenever 30 entries have been accumulated. The maximum L1rate is shown in Figure 5-12 for two cases:

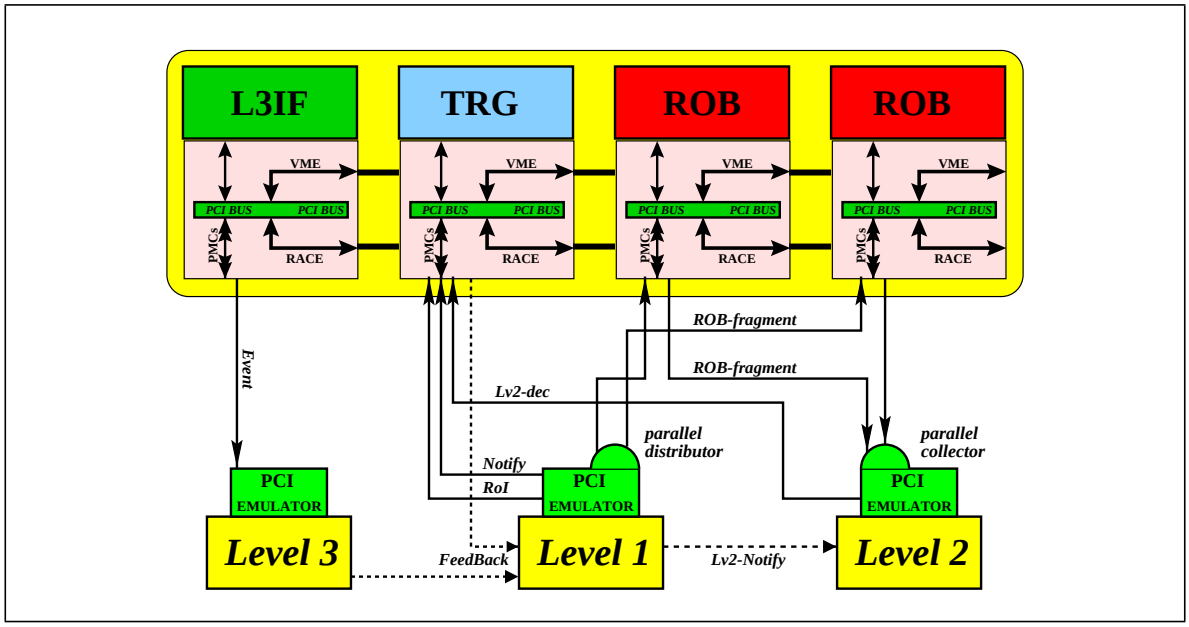


Figure 5-11 ROC model.

- *Use a single intracrate bus (VME).* All intracrate traffic is routed via the *non-broadcast capable* VMEbus, all external links have infinite bandwidth, each ROB synchronizes to the TRG whenever it receives a new fragment (one single cycle transfer) and each exchange of a *control message* is accompanied by a synchronization overhead, amounting to three single cycle transfers. The results from the simulation agree quite well with existing laboratory measurements for this case (less than or equal to 5 ROBs, the external blocks have been collapsed onto the ROC blocks).
- *Use two intracrate busses (VME and RACEway).* All external links run at the nominal PCI speed, *except for the Level1-to-ROB links, which still have infinite bandwidth (otherwise the maximum rate would be limited to 70 kHz), the synchronizations (ROB-to-TRG and control messages)* have been abandoned, all transfers are done in DMA mode. VME is only used for *data collection* (ROBs to EBIF), all other intracrate traffic runs over the *broadcast capable* RACEway. For the number of ROBs less than five, the maximum L1rate is independent of the number of ROBs in the crate (the slight rise is caused by retries during the data collection: since there is no *internal synchronization*, it may happen that EBIF tries to read a fragment from a ROB before that fragment has actually been received) because TRG *broadcasts* to the ROBs, and the PCI bus on the TRG is the bottleneck. Increasing the number rof ROBs will move the bottleneck to the PCI bus on the EBIF (the L1rate is then inversely propor-

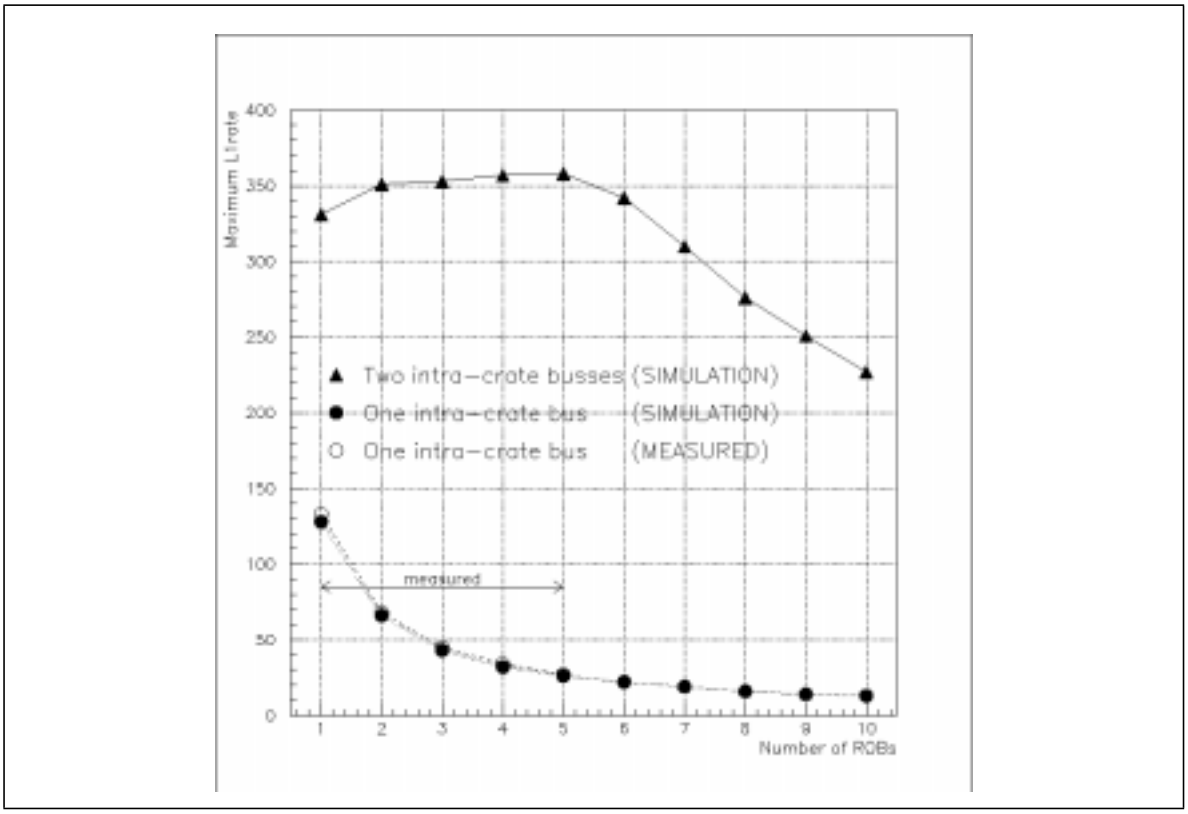


Figure 5-12 Maximum LVL1 rate.

tional to the number of ROBs due to the *sequential data collection*), which could be avoided by the EBIF sending the ROC fragments to *Level3* via a second, independent PCI bus.

The event builder

The event builder protocol

A high-level design of the event builder is defined in terms of logical elements and their behaviour. The event builder protocol defines the behaviour of the event builder elements while a functional decomposition of each element reveals the different activities within the element. This high-level design [5-20] is a functional description of the event builder and does not impose a particular implementation. In particular, the high-level design does not define how the different components of each element interact and how the 'message' between the elements of the event builder are exchanged.

The event building protocol [5-20] is shown in Figure 5-13. When a source receives an event fragment, it uses the 'GetID' mechanism with the DFM to obtain the identifier of the destination to which it has to send the fragment. The actual sending of the fragment is performed by the 'Transfer' mechanism. The destination notifies the DFM, via the 'End-of-Transfer' (EoT) mechanism, when an event fragment has been received. The 'End-of-Event' (EoE) mechanism is used by the DFM to inform the destination that a complete event has been sent from all the sources to this destination. In case a destination runs out of resources (e.g. buffer space), the 'Busy / $\overline{\text{Busy}}$' (B / $\overline{\text{B}}$) mechanism is used to cause the DFM to suspend the flow of event fragments to this destination. The same mechanism is used to resume the flow of event fragments.

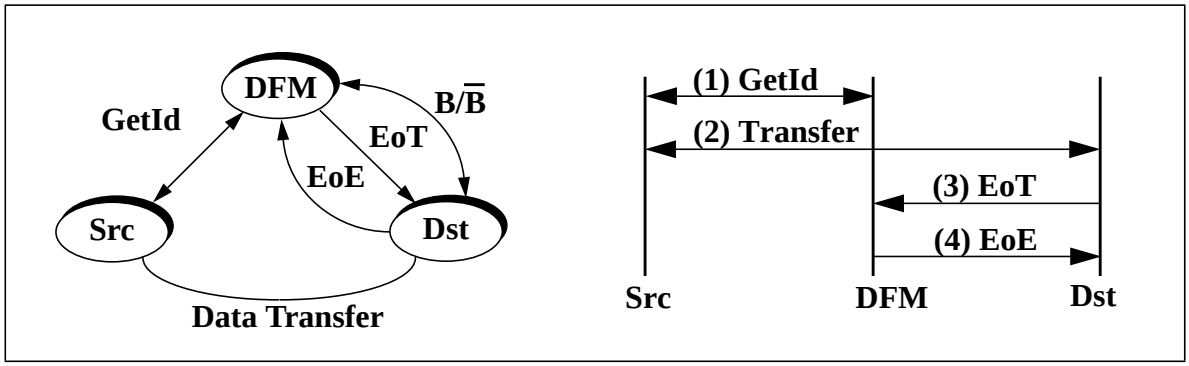


Figure 5-13 Event building protocol.

The DFM [5-28] is a logical component of the event builder which assigns destinations to global event IDs, records the arrival of event fragments at destinations, notifies the destinations when a complete event has arrived and can suspend and resume the dataflow to a given destination when the latter is busy, or $\overline{\text{busy}}$, respectively. The functional decomposition of the DFM reveals four active elements as shown in Figure 5-14. The 'Partition Control' controls the other components of the DFM. The 'Destination Assignment' receives event IDs from the HLT and assigns destinations to them. The pairs of event ID and destination ID are then broadcast to all sources in the partition, the GetId mechanism. The 'Event Completion' checks the completion of full events using the EoE and EoT mechanism with all destinations. The 'Busy / $\overline{\text{Busy}}$' marks destinations as busy, so that they do not take part in the destination assignment, or marks them as not busy, so that they can take part in the destination assignment. The components of the DFM share information about destinations and event IDs in a common data structure.

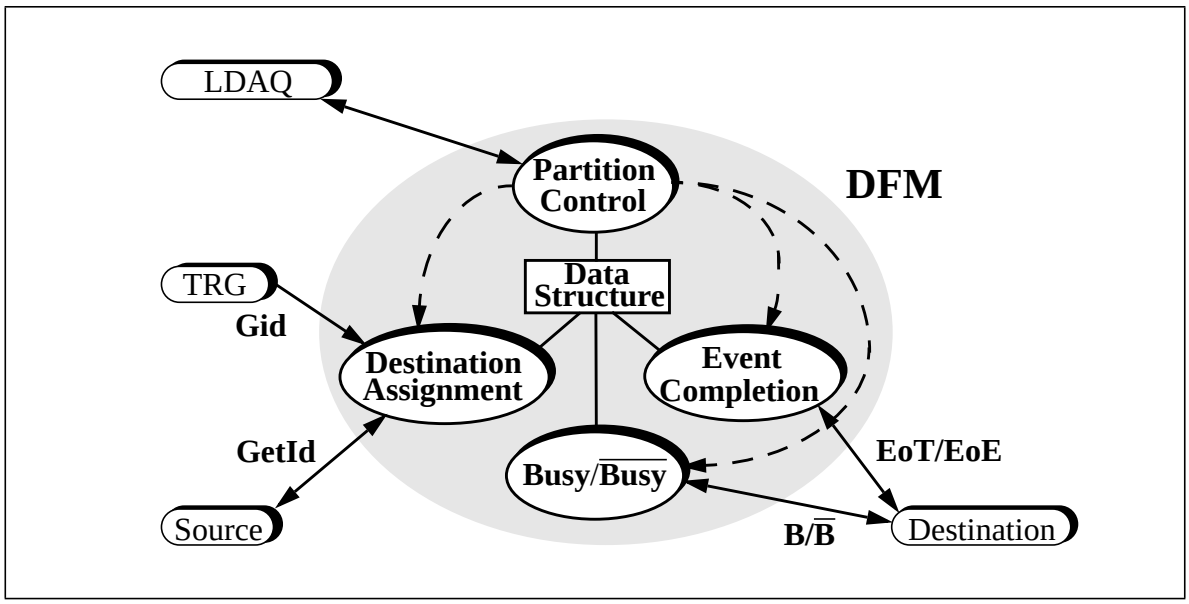


Figure 5-14 The DFM high-level design.

The sources and destinations [5-21] are responsible for sending and receiving the fragments and communicating with the DFM following the event builder protocol. The functional decomposition of the source reveals three components. One component receives destination information, in the form of pairs of event ID and destination ID from the DFM. Another component receives event fragments from the input buffer. And the last component sends event fragments to the

switch once the event fragment and the destination information are available. The functional decomposition of the destination reveals four components:

- One component receives event fragments from the switch.
- Another component exchanges event information with the DFM: for each event fragment received an EoT message is sent to the DFM. If an EoE message is received from the DFM the corresponding event is marked as complete.
- Another component of the destination accepts complete events, formats them and sends them into the output buffer.
- The last component checks the buffer space of the destination and sends Busy / $\overline{\text{Busy}}$ to the DFM when necessary.

Source and destination deal with event descriptors and do not move any event data (see Figure 5-15). The event descriptors and other information are shared between the different components of the source or the destination in a data structure.

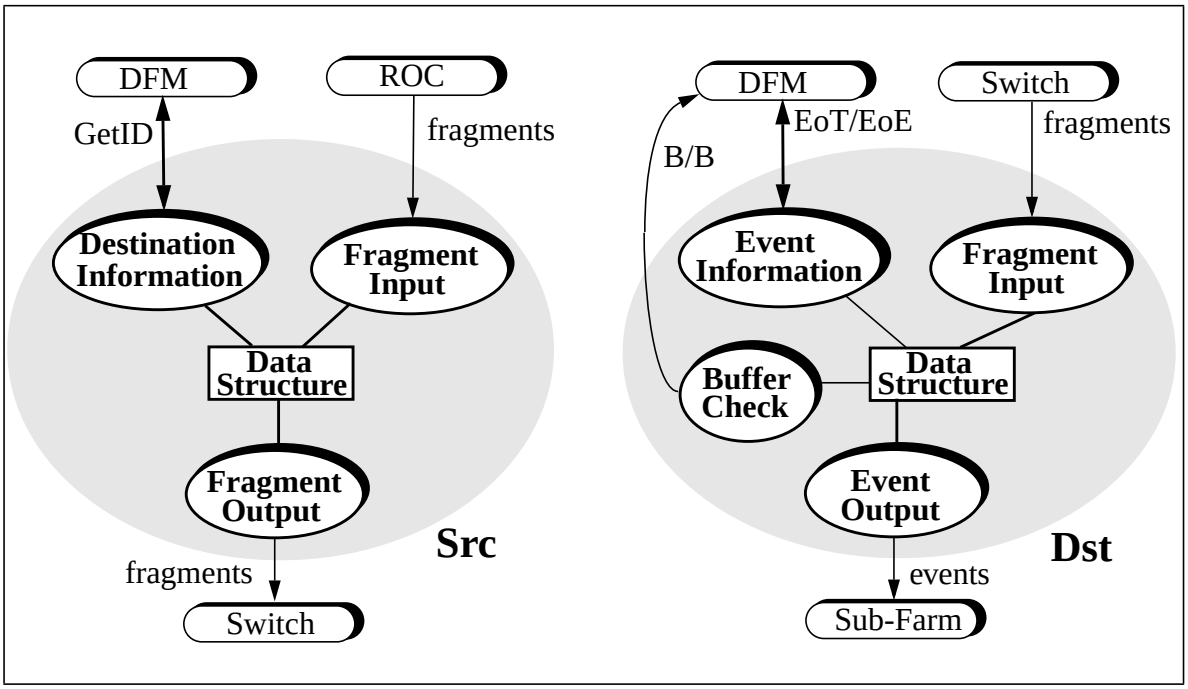


Figure 5-15 Src/Dst high-level design.

Event builder technologies

The data transfer through the switch used by sources and destinations as well as the transfer of control messages between sources and destination on one side and the DFM on the other side are accomplished by the means of a general I/O library [5-29]. This library hides all the technology specific to the upper layer and provides generic functions to initialize the interfaces with the switch, to establish logical connections between senders and receivers, and to transfer actual data or control messages. The library allows the development of the EB upper layer without dependence on a particular technology and reveals a generic set of functions for sending and receiving data and control messages in the event builder. The actual technologies can be developed independently and can be exchanged without changing the upper layer design.

Current technologies being investigated for the DAQ/EF prototype-1 are: ATM, FibreChannel and Switched Ethernet. These technologies allow a high degree of connectivity between many sources and destinations, high aggregate bandwidth and concurrent merging of event data. The event builder I/O library will be implemented for each of these technologies. In case several transfer protocols for a given technology are available (like TCP/IP or a native transfer protocol), both will be implemented. For details on the measurements and results obtained with the different technologies see Section 5.3.1.

Event builder integration

The integration of the EB system with the dataflow system is accomplished by implementing the source and destinations as part of the EBIF or the SFI, respectively. Both will be tasks executing on the respective I/O modules and using the different resources, like the I/O module scheduler and the event manager. For the simplification of the integration the event builder will also have emulated input and output, as well as an emulation of the DFM part, in order to test sources and destinations independently.

The integration with the back-end subsystems is achieved through the LDAQ subsystem described in the following section. The integration of the event builder with the high-level triggers will have to be achieved through the definition of an interface between the high-level trigger subsystems and the DFM. For the DAQ/EF prototype -1 a working definition is assumed and a mechanism defined to put event IDs into the DFM corresponding to the event IDs of the fragments in the sources.

The local DAQ

The local DAQ functions

The Local DAQ (LDAQ) subsystem provides all the common DAQ functions which are not related to moving event data. These functions, although with some degree of specialization, are common throughout the dataflow system: in the Readout Crate (ROC), event builder and subfarm DAQ.

A minimal set of functions provided by the LDAQ includes:

- local control within the dataflow: both to control a data-taking session and to deal with errors produced during the data-taking session;
- support to event monitoring programs: a statistical sample of the events flowing in the dataflow (ROC or subfarm) has to be provided to user programs for analysis/monitoring purposes;
- access to configuration data bases: the dataflow system configuration (for example how the dataflow is partitioned) and the dataflow parameters (such as buffer sizes) will be retrieved from a database. The contents of the database will be made available by the LDAQ to the other dataflow subsystems.
- interface to back-end: the LDAQ is the point in the dataflow system which connects to the back-end system, for example, as far as run control and message reporting are concerned. The LDAQ is also the interface point to the back-end DAQ system for the run control, the configuration database, the Message Reporting System and the Information Service.
- stand-alone operation: the LDAQ will also provide the facility to run a dataflow building block, such as a single ROC, in stand-alone mode (e.g. for debugging purposes). This means that suitable emulation of back-end functions will be provided by the LDAQ.

The LDAQ elements

Based on the functions listed in the previous section, the following logical elements have been initially identified in the high-level design of the LDAQ subsystem: the local control, the event monitoring, the configuration and parameters database, the LDAQ communications and the LDAQ link. An initial high-level design of the LDAQ subsystem is presented in Figure 5-16 and a detailed description is available in [5-22].

Local control

The local control element [5-30] provides run control and error handling within a dataflow building block (ROC, subfarm, Event Builder). The local control is structured around a main task, a finite state machine (FSM), which ensures that the controlled dataflow elements are always in a coherent state. An input task accepts a defined set of user commands for initializing, configuring, starting, stopping and pausing data-taking sessions. Additional commands are available for the handling of a partitioned event builder. The execution of a LDAQ control command is associated with an action which is executed remotely on one or more dataflow elements. The successful execution of a control command implies a state transition in the FSM. State transitions may also the result of asynchronous error conditions and/or notifications of state changes raised by the dataflow elements during data taking sessions. Detailed descriptions of the LDAQ control state transition diagram as well as run control command and parameters are available in [5-30]. The control element communicates with a local user interface program in stand-alone mode or is interfaced to the back-end run control in an integrated environment.

Event monitoring

The monitoring subsystem [5-31] provides the sampling of event data flowing through the DataFlow system. The monitoring subsystem does not provide the possibility to sample all events (possibly of a defined type) flowing through the DAQ; this function is available elsewhere in the system (i.e. by the event handler component of the subfarm). There is no provision for sampling of correlated fragments from different sources (e.g. from two ROBs in the same crate or from several EBIF interfaces from different crates). Also there is no provision for the sampling of a guaranteed percentage of events.

The LDAQ monitoring element is structured around two main tasks to decouple the sampling of event data (from the dataflow) from the request of events (from analysis programs). This decoupling is achieved via a database which is periodically updated to contain a consistent snapshot of the data flowing in the crate. The sampling task collects event fragments from the dataflow elements and stores them in a database. The distribution task handles user requests and the distribution of data from the database.

User programs requests ROB or crate event fragments or full events with particular characteristics (e.g. level1 trigger type, any event currently available for monitoring) from the monitoring element through a software interface which implements:

- the translation of the user request into a command to the relevant monitoring subsystem;
- the communications (and transport of the sampled data) between the monitoring subsystem and the user program.

The configuration and parameters database

The configuration and parameters database is actually part of the back-end system, nevertheless the contents of and the access to the database is relevant to the LDAQ. At DAQ start-up, the dataflow building block elements must be initialized and properly configured. The information required to accomplish this task is retrieved from the database and consists of the description of the dataflow elements in terms of hardware and software components and how these components are interrelated. The configuration parameters needed by the dataflow elements to start a new data-taking run (e.g. buffer parameters) are also retrieved from the database. A simple software interface in the form of a data access library enables LDAQ applications (control, monitoring) to traverse the database according to various criteria and to hide details of the database schema.

The LDAQ communication

This is the element (Figure 5-16) [5-32] which integrates the LDAQ with the DAQ unit / event builder. The LDAQ communication allows the LDAQ to exchange information with the DAQ unit / event builder elements on a transaction basis (send / reply) always initiated by the LDAQ.

Two different types of transactions are supported:

- Control transactions consisting of sending control commands to dataflow elements and receiving back acknowlegments and asynchronous notifications. These messages are typically short (up to a few hundred bytes) and are exchanged at a low rate (a few per second maximum).
- Transfer of event monitoring data on request from the LDAQ. These messages contain event or crate fragments and are up to one megabyte in size and are exchanged at medium rate (up to 100 Hz).

The LDAQ link

This element provides the physical path to integrate the LDAQ with the dataflow elements in the ROC, subfarm and event builder.

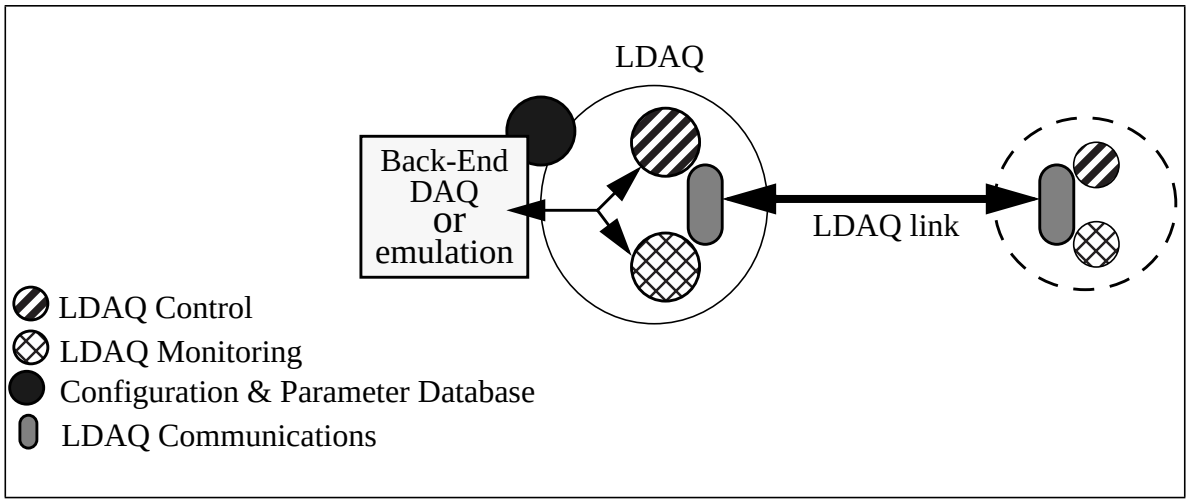


Figure 5-16 The LDAQ elements.

Integration with Dataflow building blocks

The interface between the LDAQ and the DAQ unit / event builder subsystems [5-33] has been designed to allow for a maximum of flexibility with respect to the LDAQ elements. The model of LDAQ integration is based on a message passing system between LDAQ elements: the elements located in the LDAQ and the corresponding ones located on the DAQ unit / event builder as depicted in Figure 5-16. A logical type is associated to each message and identifies the function to be executed on the DAQ unit / event builder side. The reception and the decoding of messages on the IOM side is under the responsibility of the LDAQ. In other words, the functionality of the LDAQ is extended to the DAQ unit / event builder. The interface between the LDAQ and the IOM is a functional Application Programming Interface (API) on the IOM side. Each function of the LDAQ elements is accessible on the DAQ unit / event builder side via a specific function call and a list of parameters.

This model of integration allows for a greater flexibility on the LDAQ side. The LDAQ functionality can easily be distributed in the DAQ unit without affecting the DAQ unit design. This model requires an API which is complex. One API must be defined per LDAQ element (control, monitoring). More details of the integration of the LDAQ with the dataflow subsystems follow.

The ROC LDAQ

The ROC LDAQ is realised by integrating a ‘generic LDAQ’ as described in the previous sections with the DAQ unit elements of the ROC, namely the TRG, the EBIF and the ROBs. The ROC LDAQ is partition blind.

The subfarm LDAQ

The subfarm LDAQ is realised by integrating a ‘generic LDAQ’ described in the previous sections with the DAQ unit elements of a subfarm DAQ, namely the SFI and the SFOs. The subfarm LDAQ is partition blind. In addition the LDAQ may have to support some of the subfarm supervisor function.

The event builder LDAQ

The integration of an LDAQ with the event builder system is slightly more complex than the one used in the front-end and farm DAQs. Because there is only one interconnecting network (switch) and the possibility to have multiple concurrent physical partitions at one time, there is a need for a double level of control for the event builder:

- A global control (a kind of ‘super’ LDAQ’) which is responsible for the initialization of the interconnecting network and the handling of physical partitions.
- Within a physical partition a ‘generic LDAQ’ controls an instance of the DFM. All the DFM LDAQs are under the supervision of the event builder LDAQ.

The relationship between the LDAQ and the event builder is shown in Figure 5-17.

Source (*Srcs*) elements of the event builder are controlled by their associated *ROC LDAQs* which are partition blind. In a partitioned event builder, the *ROC LDAQ* only needs to know its crate ID.

Destination (*Dsts)* elements of the event builder are controlled by their associated *subfarm LDAQs* which are partition blind. In a partitioned event builder, the *subfarm LDAQ* only need to know its crate ID.

Dataflow Manager (DFM) elements are associated to one physical partition of the event builder and are controlled by a ‘generic LDAQ’.

The special *Event Builder LDAQ* is partition sensitive, needs to access partition information (list of sources and destinations) and pass this information to its associated DFM LDAQ. In a partitioned event builder, there is one instance of event builder LDAQ.

The *NetMgr* is under the control of a ‘generic LDAQ’.

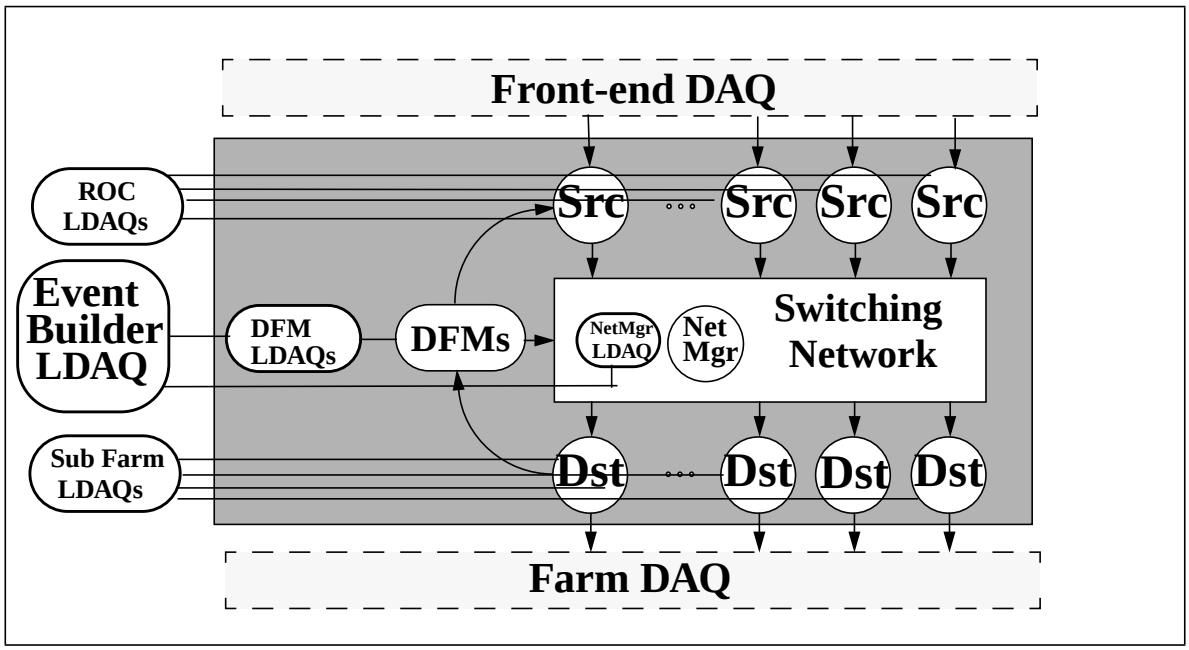


Figure 5-17 Relationship between the LDAQ and the event builder.

LDAQ status

A prototype implementation of the ROC LDAQ based on the high-level design presented in the previous sections has been made. The model of integration between the LDAQ and the IOMs is defined as an API for control and monitoring functions on the IOM side [5-33]. The same LDAQ prototype could be used in the subfarm DAQ.

The hardware implementation of the LDAQ is done on a VMEbus based CES RIO2 processor running LynxOS. The LDAQ is partially running on a Pentium-based VMEbus module running WindowsNT.

The database describing the hardware and software configuration of the ROC elements is based on the OKS [5-34] in memory database and a simple data access library.

The LDAQ communication is implemented as message passing over VMEbus shared memory and a protocol has been defined to support control and monitoring transactions between the LDAQ and the IOMs [5-32].

The LDAQ control is implemented as an event-driven multithreaded application which receives commands from a user local interface task via a TCP/IP socket connection. The user interface application inputs control commands either in command line mode or graphically via a simple Tcl/Tk interface running on a Unix workstation or on a PC. An error classification

scheme based on the source of dataflow errors and severity has been defined and a minimum of error handling has been implemented [5-35].

The LDAQ monitoring [5-31] is a multithreaded application which implements the sampling and the distribution tasks. The database used to store samples of events is implemented on top of the Rogue Wave tools library. The distribution task communicates with user monitoring programs via the ILU system.

Studies of operating systems for real-time applications including the possibility of using PCs and / or the WindowsNT operating system, at least in the area of the LDAQ, have also been performed.

5.2.2.6 Dataflow integration

The objective of the integration of the dataflow subsystem of the ATLAS DAQ prototype -1 is to produce a running, stand-alone dataflow subsystem. The purpose of a stand-alone dataflow is twofold: to make debugging of internal, dataflow related problems easier in view of its integration into the overall DAQ -1 and to provide a means for studying dataflow functional and performance issues.

Two classes of configurations are targeted by the dataflow integration:

1. A configuration including two ROCs, using data collection (i.e. grouping a number of ROB outputs into a single event builder input via a fully functional EBIF), one event builder implemented on more than one technology, and two subfarms (with a dummy event handler).
2. A similar configuration but with each ROB connected directly to an event builder input (collapsed ROB and EBIF configuration).

Configuration 1), based on the current hardware implementation for the I / O modules (VME single board computers) and the use of VME bus for all the intra-crate links will be the baseline configuration for the integration. Configuration 2) as well as variations, e.g. using different platforms for the I / O module implementation or secondary busses for intra-crate communications, will be considered after the successful integration of the baseline dataflow.

The building blocks participating in the dataflow integration are the readout crate, the event builder and the subfarm DAQ.

The steps to be undertaken towards the dataflow integration are:

- The development of the three building blocks running in stand-alone mode, i.e. with emulated input and outputs. The ROC will be provided with an emulation of both the trigger subsystems, to produce input commands to the TRG, and of the detector system, to provide input into the ROBs. The event builder will be provided with emulated EBIF and data sinks in the SFI module. The subfarm will be provided with emulated event builder destination tasks and a data sink for the SFO. This step also implies full integration between the LDAQ and the DAQ unit or event builder subsystems.
- The integration of the ROC with the event builder, by merging the event builder source task into the ROC EBIF module.
- The integration of the subfarm DAQ with the event builder, by merging the event builder destination task into the SFI element.

- The integration of the full dataflow, by adding the output half of the event builder, i.e.integration of this latter with the subfarm, to the integrated ROC / event builder complex.

5.2.3 Event filter

5.2.3.1 Definition of the EF event filter system

Introduction

The event filter (EF) system in the ATLAS Trigger / DAQ architecture is required to implement the LVL3 trigger processing as outlined in the ATLAS Technical Proposal [5-2] [5-36]. A schematic design of this architecture is shown in Figure 3-1. The EF is situated downstream of the event builder and will operate on complete event data. Starting with possible input rates from LVL2 of 1–10 GB / s, it must achieve a final data storage rate of ~ 100 MB / s, which, assuming a full event size of ~ 1 MB, implies a maximum output event rate of ~ 100 Hz. This reduction will be achieved both by minimizing the amount of data actually output to storage for selected events (it may not be necessary in all event classes to output the entire event raw data), and by rejecting events by the application of sophisticated physics algorithms designed to select desired channels. It should be emphasized that actual offline code will be used as much as possible in the EF in order both to economize effort and to allow direct comparisons between offline and online performance.

It may be desirable to ‘seed’ the EF algorithms with the results of LVL1 and LVL2 calculations in order to minimize event processing time, but even in this case, the EF latency will be measured in seconds. Access to geometry and calibration constants, data decoding and transformation, and full event reconstruction will all be required before it is possible to apply the selection algorithms to the event.

In addition to its primary task of event filtering and data reduction, the EF, being the first element in the DAQ chain having access to complete events, will also be used for monitoring and calibration / alignment studies. Many types of monitoring, including inter-detector efficiencies, physics quality assurance and trigger performance can be envisaged, as well as specialized calibration / alignment tasks.

Given the above, the EF will be highly CPU-bound with data I / O representing a small fraction of the overall latency. Extrapolations based on CDF / D0 data and ATLAS Monte Carlo data suggest that a lower limit of 25 SPECint95 sec per event will be required for the event filtering alone. This would indicate a minimum required EF processing power of 10^6 MIPS. A farm of commercial processors is considered to offer the best approach to implement a solution. Given the rapidity with which computing technology is currently evolving, and the timescale of the experiment, it is clear that any final choice of architecture for the EF should be delayed as long as possible. Possible candidate solutions include farms of PC-style processors (requiring some 1000 processors rated at 1000 MIPS each), commercial high-end computing solutions based on high-speed processors connected together with a high-performance bus or switch, and MPP machines.

Context within DAQ/EF -1 and ATLAS T/DAQ

The EF computing engine is organized functionally as a set of independent ‘subfarms’, each with a direct connection to the event-building switching fabric Figure 5-1. One of the goals of

the DAQ/EF-1 project is to reproduce a 'functional vertical slice' of the final ATLAS DAQ system. In this context, the EF functionality will be represented by a small number of prototype subfarms, each capable of handling ~ 1% of the final EF input load.

In implementing these prototypes, various competing candidate architectures will be studied with a view to learning as much as possible about their relative benefits and drawbacks in order to facilitate the final design choice. Issues of system management, control and supervision, data distribution and collection, and operating systems will be studied.

Benchmark programs based on real ATLAS decoding, reconstruction and physics selection software will be prepared. Running these on the various prototypes will enable comparative studies of the prototypes' performance to be made as well as improvements to the precision of the estimated overall required computing capacity (current figure quoted above) of the EF.

The EF is one of several subsystems which make up the overall DAQ/EF-1 prototype project. Its boundaries with these other systems are shown in Figure 5-18.

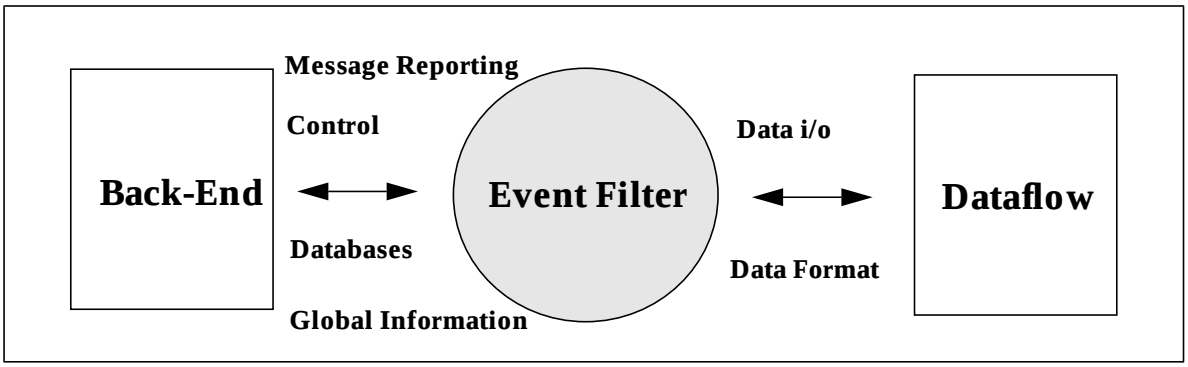


Figure 5-18 EF boundaries.

5.2.3.2 Functions and requirements

Event filter functions

As stated above, the main task of the EF is to reduce the dataflow from LVL2 by an order of magnitude, giving a maximum rate of ~100 Hz, if the full event data are to be recorded. Event-summary information could obviously be recorded at much higher rates, possibly for certain specific triggers (e.g. single-jet or multi-jet triggers for high-statistics QCD studies involving only the calorimeter information), but certainly not for the most challenging trigger items which make up the bulk of the output rate from LVL2. To achieve this, a full knowledge of the event is required. In view of the availability of powerful processors, it seems natural to use as much as possible algorithms, and possibly software, coming directly from the offline suite.

Given its position in the overall data-acquisition stream (i.e. after the event builder), the EF is also a good candidate for performing other operations not directly related to data reduction, such as performing global monitoring operations, and calibration and alignment checks. One has also to consider the eventuality of dealing with special events for calibration purposes, possibly intermixed with normal physics events.

The final performance of the system will be determined by the maximum manageable complexity and the available financial resources.

Physics requirements

In contrast to LVL1 and LVL2, some of the EF output rates expected for inclusive signal processes of interest are of the same order of magnitude as the signal itself, as shown below in a few cases for low- and high-luminosity operation. After event building, the EF will be able to perform detailed reconstruction using the complex algorithms of the offline code itself, albeit without the final calibration and alignment constants. Thanks to this potential performance, cut adjustments which are not possible at LVL1/LVL2 will be feasible; the exact selection criteria to be used in the EF have not yet been chosen, since their final optimization will most likely be performed using data taken during initial operation rather than simulated Monte Carlo samples.

In terms of physics benchmarks, most of the physics of interest, which is challenging for the EF performance, is limited to a reasonably short list of processes. This is because most of the exclusive final states of interest (these have been studied over the full spectrum of physics channels considered within ATLAS) are expected to trigger the detector at rates well below 1 Hz.

Many processes will be selected through multiple trigger signatures, thus providing optimal efficiency and several means of controlling the crucial aspects of the trigger efficiency. This trigger strategy remains essentially the same as at the time of the ATLAS Technical Proposal [5-2], but the physics studies performed since (most prominently in the areas of B-physics [5-37], MSSM Higgs physics [5-38], and supersymmetry [5-39]) warrant a justification of the statement that the ATLAS trigger strategy still covers most of the physics goals.

Inclusive lepton and di-lepton triggers provide $W \rightarrow l\nu$ and $Z \rightarrow l^+l^-$ selections, where l designates an electron or a muon. They therefore give an unbiased trigger for many Standard Model physics processes and also for many searches for physics beyond the Standard Model:

- Gauge-boson pair production for the study of anomalous couplings and to investigate the behaviour of the production cross-section at high mass.
- Top-quark production (single top or $t\bar{t}$ pairs), for all cases except production with fully-hadronic top decays.
- Direct production of SM or MSSM Higgs bosons with $H \rightarrow ZZ^*$, WW^* decays, over the full Higgs mass range of interest. Associated production of SM Higgs bosons through $WH/ZH/t\bar{t}H$ processes, with $H \rightarrow b\bar{b}$ or $H \rightarrow \gamma\gamma$, and $W \rightarrow l\nu$ or $Z \rightarrow l^+l^-$.
- Decays of MSSM Higgs bosons, such as $A \rightarrow Zh$, $H/A \rightarrow t\bar{t}$, $H/A \rightarrow \mu\mu$ and also $H/A \rightarrow \tau\tau$ with one leptonic τ decay. Production of $t\bar{t}$ pairs with one top decay to $bH^{\pm}$, where the other top-quark decay provides the inclusive W trigger.
- Production of new vector gauge bosons (W'/Z'), with W'/Z' decays to leptons. Also, resonance production at the TeV scale (strongly interacting Higgs sector), with resonance decays into gauge-boson pairs.
- Production of supersymmetric particles with final states containing:
  a. at least one high-p_T lepton and large E_T^{miss} in the case of R-parity conservation;
  b. at least one high-p_T lepton (e.g. from $\chi_2^0 \rightarrow ll\chi_1^0\chi_1^0$ decay) with or without large E_T^{miss} in the case of R-parity violation with $\chi_1^0 \rightarrow 3$ jets, $\chi_1^0 \rightarrow l\nu\nu$, or $\chi_1^0 \rightarrow ll'\nu$.
- Searches for leptoquarks decaying into electrons or muons; searches for compositeness in the lepton sector through Drell-Yan production.

The most challenging of these inclusive $W \rightarrow l\nu$ or $Z \rightarrow l^+l^-$ triggers is certainly the $W \rightarrow e\nu$ trigger, which has an expected output rate from LVL2 of 700 Hz at design luminosity [5-40] for $p_T^e > 30$ GeV. Most of this output rate is still due to background (from charged hadrons and from photon conversions [5-41]). On the other hand, the expected rate for inclusive $W \rightarrow e\nu$ signal events with $p_T^e > 30$ GeV and an additional cut requiring $E_T^{miss} > 25$ GeV is of order 50 Hz.

The above numbers clearly show that one of the main tasks of the EF will be to bring the rate of inclusive $W \rightarrow e\nu$ trigger candidates as close as possible to the real physics rate through a combination of tighter electron-identification cuts and of loose E_T^{miss} cuts. Whether the total expected rate of ~ 50 Hz would be acceptable or not is a matter for further study. More exclusive processes containing $W \rightarrow e\nu$ decays are, however, expected to have much lower trigger rates after the event filter:

1. The rate for signal events containing a $W \rightarrow e\nu$ decay and two jets with $E_T > 30$ GeV and within $|\eta| < 2.5$ is expected to be below a few hertz. This shows that the EF can provide a moderate output rate for all physics searches of the type $WH/ZH/t\bar{t}H$ production with $H \rightarrow b\bar{b}$ and $W \rightarrow l\nu / Z \rightarrow l^+l^-$ without processing the event further in the inner detector than necessary for better electron identification than that provided by LVL2;
2. The rate for signal events from top-quark production containing at least one high-p_T electron (or muon) is expected to be of the order of 1 Hz at design luminosity. Again, these events can be selected by the EF in a very inclusive way without any further processing of the inner detector information.

Obviously, the task of the EF in terms of selecting inclusive $W \rightarrow \mu\nu$ decays or $Z \rightarrow l^+l^-$ decays is easier than that of selecting inclusive $W \rightarrow e\nu$ decays. This is because the expected LVL2 output rates for the high-p_T single muon trigger and for the isolated high-p_T di-lepton triggers are much lower than for the high-p_T single isolated electron trigger. The expected rate for inclusive $Z \rightarrow l^+l^-$ signal events is 10 Hz at design luminosity, and the EF will clearly be able to approach this output rate by using, for example, a mass cut on the lepton pair.

The physics channels not covered by the inclusive lepton / di-lepton (and electron + E_T^{miss}) triggers discussed above are:

- B-physics, which is a particular case, since each class of exclusive channel of interest needs its own dedicated trigger study for LVL2 and for the event filter, given the very high rate of inclusive B-hadron events containing a high-p_T muon used to trigger at LVL1.
- $H \rightarrow \gamma\gamma$ decays from direct Higgs production. These triggers also cover possible MSSM Higgs boson decays such as $H \rightarrow hh \rightarrow bb\gamma\gamma$. Significantly looser kinematic cuts than those used in the analysis, e.g. $P_T^\gamma > 20$ GeV and $m_{\gamma\gamma} > 60$ GeV, will reduce the EF output rate for such events to a few hertz.
- Searches for supersymmetry involving high-p_T jets with or without E_T^{miss}. At low luminosity, the combination of jet + E_T^{miss}, single-jet and multi-jet triggers at LVL1 and LVL2 (see Chapter 11 in [5-42]) provides excellent coverage for all exclusive final states of interest not containing leptons. In the case of R-parity conservation, final states containing at least two high-p_T jets and large E_T^{miss} (typically two jets with $E_T > 150$ GeV and $E_T^{miss} > 200$ GeV) provide a broad inclusive sample for more exclusive searches. This also applies largely to SUSY searches in the framework of gauge-mediated SUSY-breaking models. In the case of R-parity violation, with χ_1^0 decaying predominantly to three jets, the final-state jet multiplicities range from 8 to 16, which completely suppresses the characteristic E_T^{miss} signature of SUSY events. Here the multijet triggers will surely cover most of the exclusive final states of interest. It is important to note, however, that to date the only

exclusive final states which have been proven to be observable above the huge potential QCD background are final states containing one or preferably two isolated high-p_T leptons. At high luminosity, the higher thresholds applied to the various jet triggers and to the jet + E_T^{miss} trigger will be well suited to searches for higher-mass SUSY particles. Further study would be needed to optimize the triggers for systematic studies of lower-mass SUSY particles, should they be discovered in the initial years of operation.

- Searches for leptoquarks decaying into jet and neutrino that rely on the jet + E_T^{miss} trigger.
- Searches for resonances decaying into jets and for compositeness in the quark structure. These rely largely on the inclusive single-jet trigger. The thresholds quoted for this trigger at low and high luminosity will provide good coverage of these physics channels, given the expected reach of the TeVatron experiments in this field before the LHC starts operation.

As stated above, the area of B-physics is another challenging task for the EF :

- since the expected output rate from LVL2 is of order 1000 Hz for B-physics alone at low luminosity,
- and since most of the candidate events are genuine B-events, for which a complete and accurate reconstruction of the inner detector information is necessary in order to further reduce the rate, in particular using vertexing cuts.

The largest rate from physics events in this field arises from inclusive $J/\psi \rightarrow \mu^+\mu^-$ decays with $P_T^{\mu_1} > 6$ GeV and $P_T^{\mu_2} > 5$ GeV. The total expected rate for signal events is about 5 Hz from direct J/ψ production and about 3 Hz from inclusive $B \rightarrow J/\psi$ production. These events are expected to be heavily used in CP-violation studies with jet charge and B-π tagging methods applied in addition to the more traditional lepton tagging. They are also expected to be used as calibration events to study in particular the systematic uncertainties in the knowledge of the absolute mass-scale in the experiment. To reduce the LVL2 output rates to values close to the physics rates quoted above, a combination of vertexing cuts on the muon pair and of tighter kinematic cuts, including mass cuts, will have to be performed by the EF. The more exclusive B-decays are expected to contribute at much lower levels of typically 0.1 Hz per channel or less.

It is also hoped that the large variety of fairly inclusive triggers presented here would be sensitive to unexpected new physics. Finally, it is important to emphasize that much of the early large cross-section physics (e.g. QCD jets, direct photons, etc.) will be studied using more inclusive triggers than the ones quoted explicitly in [5-40] as well as dedicated algorithms in the EF.

The menus and rates presented in [5-40] will be used as a basis for menus for more detailed studies of the EF in terms of performance and of implementatio. Those trigger items that are considered particularly challenging or critical will be subject to detailed trigger performance studies using fully simulated data as input and offline reconstruction code as a reference. Wherever possible, the trigger performance results will be parametrized for use in fast simulations [5-43] with high-statistics background samples.

It is clear that a complete set of output rates for the EF can only be obtained through a combination of detailed full-simulation studies and of parametrized fast-simulation studies. In conclusion, the role of the EF will be very important in determining the scope and breadth of trigger channels available to ATLAS to study in detail the physics channels of interest and to constrain as well as possible the background estimates to possible new physics. It is hoped that most of

the physics goals, with the notable exception of B-physics, can be achieved without any complete processing of the inner detector information.

Trigger and detector monitoring requirements

Global trigger and detector monitoring is essential to ensure the quality of recorded data and should be performed in a regular manner. The EF is ideally situated in the data-acquisition stream to perform such checks online and at a minimal cost in terms of required CPU since events will have already been, at least partially, reconstructed for filtering purposes. Because the full event information is available, many cross-checks can be performed, which are impossible earlier in the dataflow. It is also important to monitor the trigger performance at both a global level by looking at the rates of the different channels and at a more detailed level by comparing the decisions of all the processors in the trigger chain to the results of the trigger algorithms running offline on real minimum-bias data. This comparison technique should provide a powerful tool for the debugging and the quality control of the trigger system.

The following tasks could be performed at the EF level:

- LVL1 and LVL2 trigger monitoring,
- matching and correlations between different detectors,
- monitoring of individual physics channels from the rates of accepted and rejected events.

In order to reap the maximum benefit from the reconstruction work already performed in the filtering stage, two possibilities can be considered for combining the two functions:

- the monitoring activity is performed in the same computing context as the event filtering,
- the results of the reconstruction are appended to the event at the output stage and monitoring is performed in another farm, possibly a separate partition of the EF farm.

The trigger, detector and physics monitoring activities described above will be installed in the EF context only if they would not reduce the overall filtering capabilities allowed by the available computing power. Monitoring algorithms are the responsibility of the detector groups. The EF should provide convenient hooks to execute these algorithms and a flexible event-sampling policy.

Calibration and alignment requirements

Some calibration and alignment tasks can be performed in the EF by using specially selected events. This function is important since the quality of the EF decision relies strongly on the quality of the detector calibration and alignment parameters. In general, these parameters need to be calculated more or less continuously, using complete event data from the detector. Often, specific calibration triggers are required (pulser, source, cosmic rays, minimum-bias events, etc.). The only logical level in the Trigger and DAQ architecture where one can perform these calculations is that of the EF. Otherwise, they must be done either entirely offline (processing delay may well not be acceptable) or on a separate platform (creating duplication of effort). If statistics can be gathered on a long-term basis, one could envisage the full execution of the calibration and alignment tasks directly in the EF farm. It would also be possible to analyse directly the special events generated by the detectors for calibration and alignment purposes, if these events are transferred to the EF.

A few examples of physics events which will be of use for calibration and alignment tasks are given below:

- single electrons with $p_T > 20$ GeV, $Z^0 \rightarrow e^+e^-$ and $W^\pm \rightarrow e\nu$ decays for the inner detector and the electromagnetic calorimeters;
- single muons with $p_T > 6$ GeV, $Z^0 \rightarrow \mu\mu$ and $W^\pm \rightarrow \mu\nu$ decays for the inner detector and the muon chambers;
- γ + jet, Z^0 + jet events, isolated charged hadrons from $W^\pm \rightarrow \tau\nu$ decays for the hadron calorimeters;
- $W^\pm \rightarrow e\nu$ and $W^\pm \rightarrow \mu^\pm \nu$ decays to study the E_T^{miss} scale and resolution;
- minimum-bias events to measure the beam position (transverse and longitudinal).

The Detector Interface Group will collect the needs of the various detectors and organize a plan of work aiming to fulfil these needs in an optimal way. The procedures used to determine the calibration and alignment parameters will be very similar to the monitoring tasks mentioned above, and similar constraints will apply.

Data security requirements

The need to minimize the loss of data and guarantee data integrity at every step of the processing is obvious. Opportunities for failure are numerous, considering the size of the processing farm and the complexity of the software. Failure may occur at the input or output of the EF or while moving or processing the event within the farm. It is not easy to be exhaustive on the possible causes and it will be an issue of the DAQ/EF-1 to help to identify, understand and propose correction procedures for possible defects. One may already identify the following global categories of failure:

- hardware failure at all levels (data I/O, processing),
- run-time error while processing the event,
- data-integrity loss while moving the data

A real-time backup of all events on input to and output from the EF will be performed in order to be able to recover events in the case of failure during processing or output. Efficient strategies will be provided to recover from crashes in a safe and transparent manner. They will in particular provide the means to perform some consistency checks so that both data and run integrity are ensured.

Strong emphasis on reliability and robustness will be made in the design and implementation of the different stages of the software. Integrity checks (e.g. through the use of CRC tests), will be performed as often as possible without degrading the overall performance.

Scalability requirements

In order to maintain the R&D studies within the boundaries of the currently available funding, only rather small-scale prototypes will be realised in the short-term future. It is therefore of major importance to include scalability issues in the design strategy for the EF.

The global organization of the EF into several subfarms, the number of which could reach the order of one hundred in the final implementation, each connected to an output port of the event builder switch, makes the design scalable in terms of reaching the required computing power.

The design of the EF subfarms will be as modular as possible, so that any change in this design can be implemented in a smooth and transparent way.

Modelling is an important tool for the validation of prototype scalability. Models will be developed in parallel with the prototype designs in order to reflect the details of this design as accurately as possible.

5.2.3.3 Event filter high-level design

Introduction

The high-level design of the EF addresses questions of overall supervision, control, and the internal flow of data. It also addresses the issues of interfacing the flow of data within and between the processors of the EF with the overall dataflow as described above. The work which has been done to date is more particularly concerned with DAQ/EF-1 issues at the subfarm level. The aim is to produce a generic design for a subfarm unit which is totally independent of implementation and architectural details, with a view to being able to use the same design for the different physical prototypes which are constructed within the context of DAQ/EF-1.

Interface with dataflow

The EF subfarm has been factorized into several components in order to aid parallel development and to identify the functional composition of the subfarm (Figure 5-6). The subfarm Input (SFI) receives fully assembled events from an output channel of the event builder, performs final event formatting and makes the events available to the Event Handler (EH) via the distributor interface. The EH comprises the processing power of the EF on which run the processing tasks responsible for the event filtering and selection. Accepted events are passed via the collector interface to the subfarm Output (SFO) from where they are sent to data storage (Figure 5-19). The SFI and SFO are elements of the dataflow whose functions are described elsewhere.

The particular function of the distributor and collector is to isolate the internal functions and architecture of the dataflow in the SFI and SFO from those of the event handler. The latter will therefore appear as a ‘black box’ to the former. The distributor will provide the functionality for the SFI to inject events into the EH, and the collector will provide access to selected events to the SFO for further disposal. This approach permits parallel development of components of the dataflow and EF, enables removal and insertion of different subfarm prototypes into the overall DAQ/EF-1 architecture and also establishes a clear boundary of responsibility.

Interface with back-end

The major services of the back-end DAQ required by an EF subfarm are those of Configuration Databases, Information Service and the Message Reporting System. As with the Dataflow, the Back-End should see the EH as a ‘black box’, communicating via a single interface, the EH Supervisor (see below).

Event distribution

Load balancing across the subfarms, system partitioning and the detailed strategy of event distribution to the SFIs, and consequently to the subfarms, will be studied within the context of the Dataflow group. It may well be desirable, for example, to direct certain specific event types (such as dedicated detector monitoring events) to a particular SFI so that they would subsequently be processed by a dedicated subfarm unit of the EF.

Once a data stream arrives at its SFI, it is input to the distributor. There will be a wide spectrum of event types of which we can identify the following global categories:

- Physics events accepted by the L1 and L2 triggers (physical events)
- Random and other experiment wide test and calibration triggers (nonphysical events)
- Test and calibration triggers specific to and generated by single detectors (nonphysical events)

Clearly, the event processing will be different depending on the global event type. The event distribution scheme so far developed [5-44] [5-45] provides for the distribution of events by the Distributor to a given processing task type within an EH, as a function of the event's global event type.

This scheme allows the architecture of the processing tasks themselves to be somewhat simpler and therefore easier to maintain.

On input to the distributor, a copy of the event is stored for the entire processing time. It is deleted only when either the event has been rejected by the filtering algorithms or it has been passed to the collector for output. The collector also retains a copy of the output event until it has been informed that it has been successfully disposed of by the SFO. This buffering mechanism aims to ensure the maximum security of the data during its passage through the EF by making it possible to reprocess an event which would otherwise be lost due to hardware or software failure.

The dataflow within the EF is data driven and is controlled entirely by the relative occupancy of the internal buffers. The overall run control and supervision of the subfarm does not intervene at all at the event level.

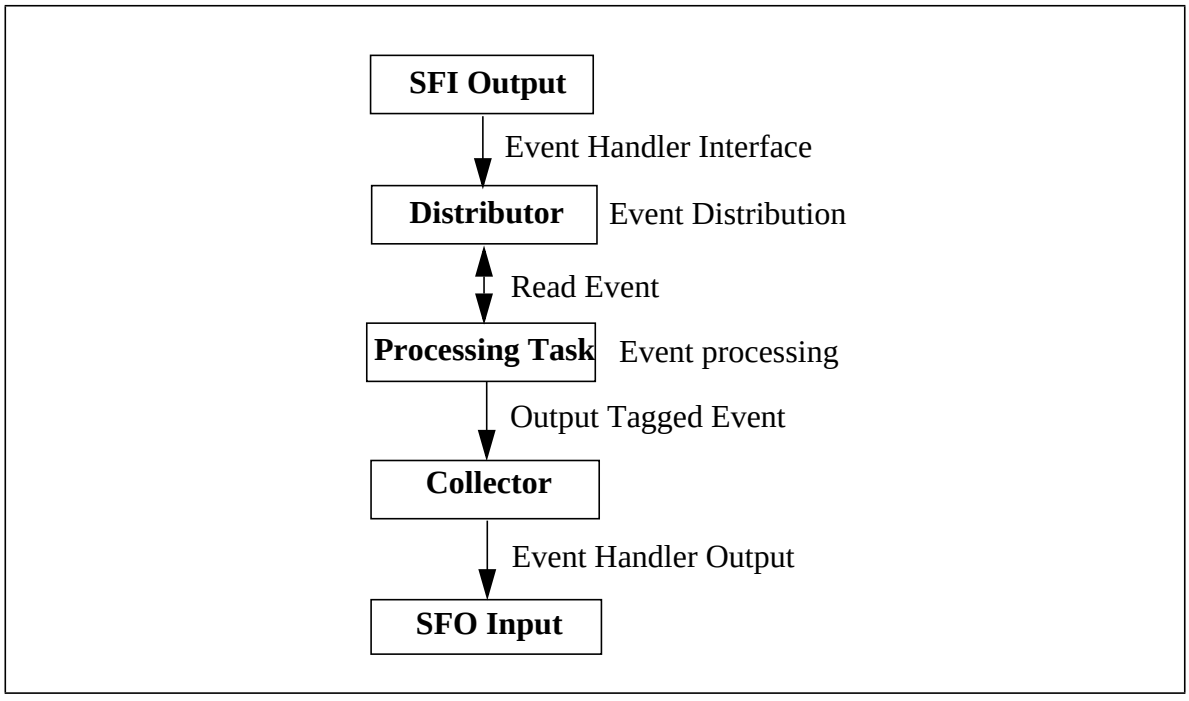


Figure 5-19 Event handler global dataflow.

Processing tasks

Processing task is the generic name for the task which processes an event in the event filter. In our present view, *events* are not limited to the strict definition of the assembly of fragments coming from the front-end crates (*physical* events). We foresee the possibility of treating the result of

data not coming from a beam-crossing interaction. Such *nonphysical* events, could be the result of laser shots or any action in view of calibration or detector integrity check. The more precise definition of such events is the subject of ongoing studies within the Detector Interface Group.

Physical and nonphysical events will be distinguished by the event type, located in the event header. The interleaving of physical and nonphysical events in the dataflow is an open question which must be better understood. Nevertheless, we do not want to reject *a priori* the possibility of online analysis of nonphysical events and therefore the concept of event distribution according to event type has been retained in the High Level Design of the Event Handler. In the following, we shall consider only *physical* events.

The primary instance of the Processing Task is event filtering. At the end of this procedure, the event is tagged for passing to the SFO and mass storage if accepted, or it is rejected. As already stated in Section 5.2.3.2, we may think of using reconstructed events for other purposes: 'express line' for gold-plated or strange events, monitoring, calibration and alignment. Reconstruction and filtering are to be done in every case. We therefore identify them as *primary* tasks. Tasks following this phase are labelled *secondary* tasks and should be executed on both accepted and rejected events.

The design of the processing task will be strongly influenced by the decision, not yet taken, about the platform on which secondary tasks are to be executed. Secondary tasks will need the whole event reconstruction information. Some monitoring tasks may only fill some histograms with parameters of the reconstruction and filtering tasks while others may involve complex analysis. It is therefore obvious that the execution of the secondary tasks in the same context as the primary task will allow a lot of savings as far as computing and communications resources are concerned. The drawback is an increased complexity in the software organization and an increased fragility of the processing task.

The processing task will be a stateless object which does not receive commands from the run control. As soon as it has been launched and initialized, it waits for events to process. Two internal modes can be then defined: *processing* and *waiting*. It is completely data driven. Several possible mechanisms for overall task control are under study. Care must be taken not to mix events not belonging to the same run.

Supervision and control

Each Event Handler is a separate control domain which can be independently supervised and controlled from the DAQ Run Control. The interface between the EH and the outside control world is the EH Supervisor (one per EH). The following section describes the EH supervision and control system [5-46] and the manner in which the subfarms are integrated and coordinated into the global DAQ run control system via the supervisor. A schematic diagram is shown in Figure 5-20.

The EH supervision and control system provides the following functions:

- start EH processes from a predefined configuration

  The EH supervisor should start in the suitable order all the processes that are needed to have the EH running. The configuration is described in a file or database. It contains the mapping of processes onto the EH machines and all the data required to start the processes.

- orderly stop the EH

  The EH supervisor is in charge of properly stopping the EH without event loss.

- provide EH local monitoring

  The EH supervisor collects the monitoring data that are produced by the various software components and provides the interfaces for users to browse and analyse them.

- capability of scanning the EH component status and publishing the information

  The EH supervisor may extract from the EH, a subset of local data which has been requested by users (programs or shift crew) and publish them.

- ensure processing tasks are ‘alive’

  The EH supervisor manages the processing tasks. Should they crash, it detects this and restarts them.

The EH supervisor architecture and the technology used to implement it have been designed in order to make:

- the supervisor independent of the event filtering tasks

  A failure of a supervisor task does not affect the filtering task. The latter continues processing while the supervisor is restarted. A crash of a processing task is however a ‘normal’ event for the supervisor, which is designed to handle it. Processing tasks are not aware of the supervisor’s existence.

- the supervisor functionally independent of EH architecture.

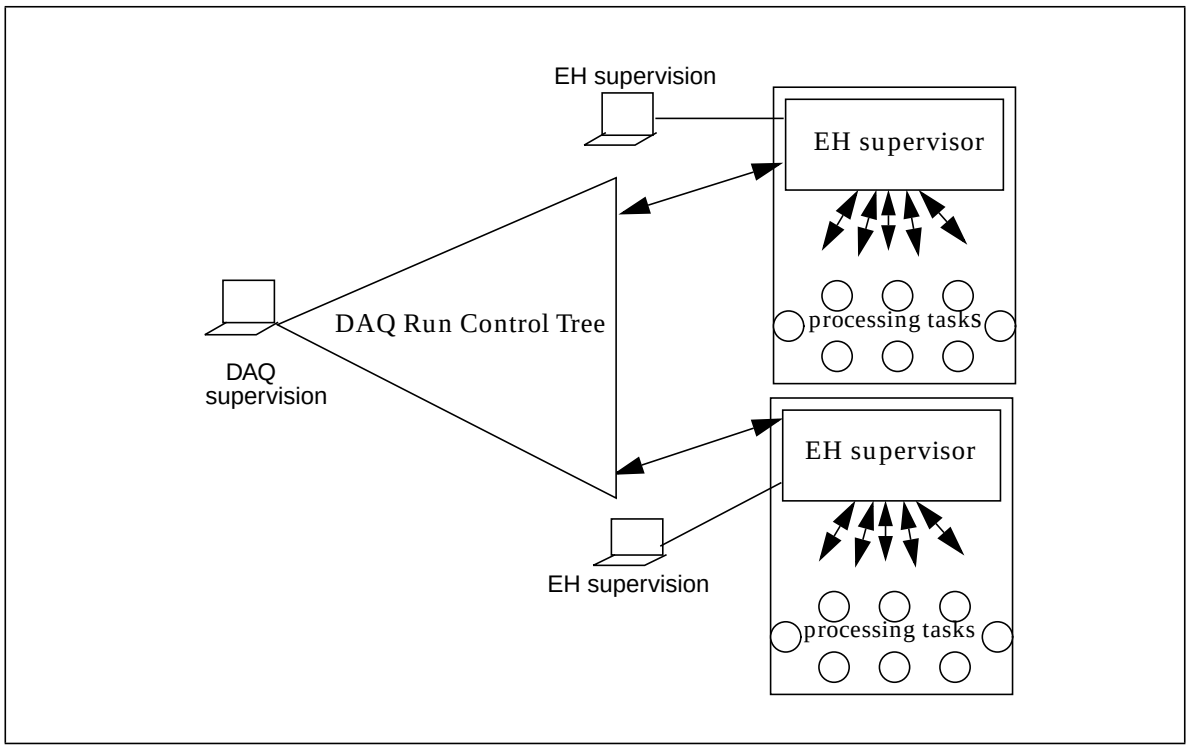


Figure 5-20 Integration of the EF event handlers in the DAQ run control system.

When it is under the control of the DAQ Run Control system, the EH supervisor receives commands from a dedicated Run Control Controller (RC_Controller) which implements standard

control procedures. This local RC_Controller broadcasts the DAQ-wide coordinated commands and looks after its EH state changes and associated actions.

The supervisor, in addition to its run control interface function, will also be responsible for implementing the configuration database for the EH and for sending/receiving status and error traffic via the Information Service and the Message Reporting System.

Error handling and recovery

Error handling and recovery is a topic which is still difficult to address in detail since it will strongly depend on the implementation of both hardware and software. For the time being, one can try to list possible origins of errors and propose a strategy for handling them.

Most run time errors will be caused by unforeseen event topologies. The chance that simple reprocessing resolves the error is very small. However, such events must clearly not be rejected. It is therefore proposed to direct them to a dedicated storage channel for further detailed analysis. Error messages and logging will of course complement the debugging information available on such failures. A detailed study of debugging techniques and procedures will be undertaken within the bounds of the EF prototypes (see below).

System errors are more likely to happen during the data transfers or in the control procedures. Reprocessing of the event will be the action undertaken when they occur. Events will be buffered during their progress through the event filter pipeline and kept until they have been safely transferred to the output (SFO) stage. Here again, error logging and reporting will be provided so that the appropriate corrective action may be taken where appropriate.

Considering the mean time between failures (MTBF – ~17000 hours for currently available PC processors) of the hardware, and the huge number of processors and devices in the farm, hardware errors are bound to happen several times a day. Safe storage areas such as disks must be used for temporary storage until the event has been safely disposed of. In the event of a failure, recovery procedures will be used to reprocess the event on another processing task. Replacement of faulty material should affect as little as possible the overall activity of the subfarm, making the capability of live insertion of components very desirable.

It is of major importance to evaluate as precisely as possible the ‘normal’ fault rate so that the event filter can be oversized and provide the required processing power at a minimal cost.

5.2.3.4 Prototypes, benchmarking and modelling

Introduction

In order to progress in the understanding of the problems inherent in implementing the EF in the ATLAS DAQ architecture, we have decided to construct several prototypes. The first two, based on the contrasting architectures of network interconnected PC processors, and symmetric multiprocessors are currently being implemented and are described briefly below. Each prototype will be implemented as a single subfarm with a view to integrating it into the DAQ/EF-1 architecture. This will enable us to compare and contrast the various prototypes in a homogeneous and realistic environment. Issues of physics algorithm performance, system management and control, reliability, error handling, and scalability are among the many issues which will be investigated. Of particular interest to the SMP-PC farm comparison will be an estimate of the primary investment and installation cost of an SMP farm vs. the cost of the more complex system management and maintenance of a PC farm. The various prototypes will also be the subject

of system modelling studies in order to try to extend the knowledge gained from the prototypes to larger scale implementations of the same architectures.

The detailed study of prototypes, physics benchmarks and complementary modelling will be the major domain of activity in the EF group in the coming two years.

Realistic benchmark algorithms

The presently available offline software accepts data produced by the ATLAS simulation code DICE, based on the GEANT3 package. The simulated events are stored in structures managed by the ZEBRA package (written in FORTRAN) which emulates dynamic structure management. Memory management by ZEBRA is not very efficient and slows down the execution of the code considerably. Moreover, the code has been written to evaluate as quickly as possible the detector performance and little effort has been made to optimize the execution performance. Benchmarks based on this code are therefore not very realistic in terms of execution time on the EF.

The ATLAS policy is to migrate progressively towards object-oriented architectures and software. Many problems have yet to be solved, in particular the question of the structure in which to store the events, which is in fact the input format of the reconstruction algorithms.

Meanwhile, realistic implementations of the algorithms are necessary in order to test the subfarm architectures, establish realistic recovery procedures, and also improve the quality of the evaluation of the overall required EF processing power. A C++ version of the reconstruction code of the electromagnetic calorimeter is now available. Events in ZEBRA format are converted to C++ classes by a dedicated translation program. This code will be used in the first prototypes of the event filter to validate the architectures.

In the near future, more implementations of the reconstruction code and of some filtering algorithms, involving other detectors, will be necessary. Profiling techniques will be used to detect software bottlenecks and optimize the quality of the implementation.

Modelling

Modelling and simulation are used to approximate the behaviour of hardware and software architectures of the event filter prototypes already at the stage of system design and testing.

Modelling is the process of making abstractions from real systems. It is essential that modelling be done in close collaboration with system designers and on the basis of a clear and unambiguous proposal for design or implementation. The design, i.e. the structure and metrics of a model, should be largely independent from the evaluation technique used for simulation.

Two different approaches have been identified for the evaluation of models for distributed computing architectures:

- analytic evaluation based on a spreadsheet (Paper Model),
- computational, discrete-event evaluation (Event-Action Model).

So far, an analytic model has been prepared to study the proposal for an event filter prototype based on PC computers in the DAQ-1 context [5-44] [5-47]. The modelling technique, the calculation scheme and the implementation of the spreadsheet are described in [5-48]. The model has been used to evaluate different hardware configurations, varying data transmission and computing resources in quantity and distribution. The results of this paper modelling effort have

been in overall agreement with performance observations on today's PC clusters. It was demonstrated that the overall task of event filtering within a processing element of the event handler is CPU-bound relative to the currently available resources of PC / workstations and commodity networking technology.

The aim of the analytic model was to estimate the impact of coarse-grain design parameters to the behaviour of the overall system. Parameters analysed, were among others:

- rate and size of events, processing requirements and efficiency of a filtering task,
- the number of processing elements per subfarm and their capabilities relative to data I / O and CPU power,
- the frequency and processing requirements of control messages.

The analytic model has demonstrated the feasibility of an implementation of the EF by means of distributed computing technology. Feasibility studies concentrated on performance issues only. The operational conditions and the behaviour of individual system components have not been studied in detail and are beyond the scope of the chosen analytic evaluation approach.

One fundamental shortcoming of the spreadsheet model is its failure to take into account the time behaviour of processes in the system (a process in this context refers to a certain well-defined task that has to be accomplished by one or several involved systems in a finite amount of time). Temporal and causal relationships between such processes are not reflected in the calculations. Effects of temporal ordering and interleaving of resource demands imposed by processes may be neglected on systems with generally lightly loaded resources, they can, however, significantly influence the behaviour of a system which is heavily loaded.

This shortcoming can be overcome by an event-driven modelling and simulation paradigm. Guidelines and fundamental principles on how to develop such models for distributed computing systems have been studied and evaluated in [5-49]. As the effort for the development of an event-driven model can be considerably higher than the development of a spreadsheet model, the field of investigation and the aims of modelling must be carefully defined in advance. In particular, subsystems of the event filter will be the subject of computational modelling. Issues of investigation will be:

- the vertical scalability of a subfarm, i.e. the number of processing elements and their individual performance characteristics relative to a common I / O path;
- the data and control flow within the event handler, featuring the software / process architecture of the event handler.

Although not crucial, the choice of modelling tool can contribute to the efficiency of a modelling and simulation process. A modelling tool should be understood as a facility to implement abstractions of a model for the purpose of evaluation, i.e. simulation.

The generality of a model and its corresponding simulation can be assumed to decrease with increasing detail and implementation reference of a model. The modelling efforts should thus be parallel to and closely collaborate with each stream of development for the different architectural proposals envisaged. In addition, performance analysis and benchmarks of prototype systems should provide valuable parameters to modelling.

PC prototype

An EF subfarm prototype based on a small farm of PCs is currently being implemented [5-47]. The outline design of the prototype is shown in Figure 5-21. A Sun multiprocessor server with a large disk containing several thousand ATLAS Monte Carlo simulation events acts as a data source for the farm. The server also acts as the SFI/SFO interfaces with the dataflow and will host the distributor and collector processes. Events are sent as fast as possible to the processing tasks running on three PCs. One is a single-processor Pentium Pro 300 MHz. The two other ones are Dual-processor Pentium Pro 300 MHz. Each runs a variable number of processing tasks, receiving events from the Distributor. The machines are currently connected together by a 100 Mbit Ethernet hub. Various different communication layer software options are under study (TCP sockets, ILU [5-50], TAO [5-51]) for the implementation of the distributor and collector.

The processing tasks running on the prototype are based on actual offline reconstruction software, translated from FORTRAN into C++ for benchmarking purposes (see above). Events used for processing are the direct output from the ATLAS Monte Carlo data, but have been converted from ZEBRA to C++ classes for ease of access by the processing software.

The farm will be used to study a wide variety of issues related to the final choice of EF architecture, namely:

- communication bottlenecks in an heterogeneous computing environment,
- required computing power evaluation based on realistic ATLAS Monte Carlo data and reconstruction and physics algorithm software,
- supervision control and monitoring benchmarks,
- database access via a data server for geometry and calibration constants required in event reconstruction.

It is also planned in the near future to replace the 100 MB/s hub by a 1GB/s Ethernet switch. We also intend to replace the inter-PC Ethernet connectivity with a shared memory interface implemented by PVIC vertical interconnection [5-52] in order to study the various advantages of direct memory connections.

SMP prototype

A natural alternative subfarm configuration to the network-connected multiprocessor implementation (such as the PC-based design, illustrated above) is the Symmetric Multi Processor (SMP). The SMP architecture offers advantages in data sharing and transfer between the different software and hardware components of the subfarm, thanks to the presence of a large shared-memory buffer, coupled symmetrically to all the processors via a high-speed internal bus.

The technical choice for this prototype [5-53] is a SMP machine from Hewlett-Packard (the choice of the platform is largely due to the convenience in porting existing analysis and reconstruction software) whose architectural scheme can be seen in Figure 5-22. This machine is a 4-way SMP, each processor accessing a private 1 MB + 1 MB (data + instruction) 2nd-level cache, while the shared memory and the external resources are concurrently available via the (64 bits @ 120 MHz) RunWay bus, with a sustainable rate of 750 MB/sec. We foresee two possible configurations for implementing a prototype EF subfarm:

- SFI/SFO on external VME cards, with fast Ethernet connections arbitrated by message passing protocols

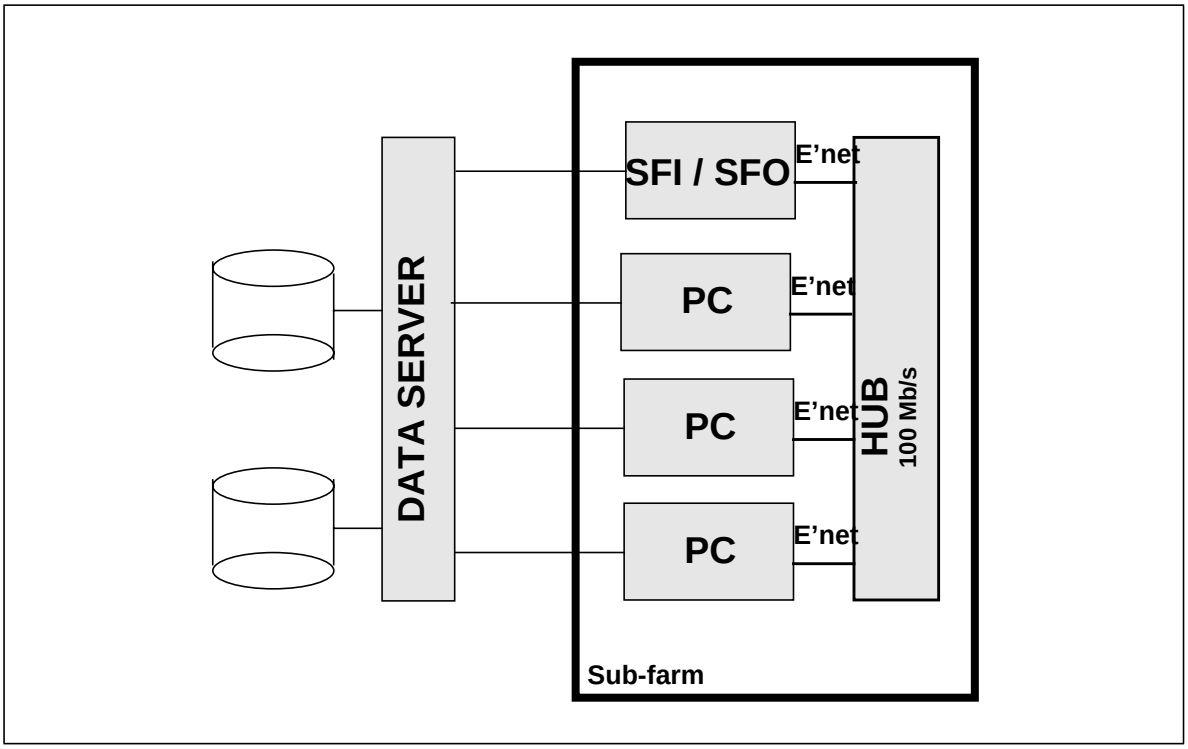


Figure 5-21 Schematic diagram of the PC prototype subfarm.

- SFI/SFO implemented internally in the SMP, to further minimize the global design complexity

In both cases, data distribution and collection inside the SMP will only require the movement of event pointers. The overall functional architecture of the high level design (see above) will be followed, while taking maximum benefit of the SMP architectural advantages for the implementation itself. We shall give highest priority to the investigation of native solutions.

In addition to confirming the measurements and evaluation listed in the previous paragraph, the SMP approach deserves a careful investigation of cache performance, asynchronous process management, and dataflow derandomization. Given the highly interlaced operation of this kind of structure, the possibility of destructive interference is increased with respect to the networked solution: hence, special care will be given to the issue of data integrity and recovery.

Other possible prototypes

In addition to the proposed prototypes, several other implementations of the event filter subfarm could be envisaged. Some of them are simply different incarnations of one of the two prototypes described above (e.g. VME cards connected by high speed bus in a crate, or a network of workstations). Performance measurements on them could be extrapolated from our experience with the PC and SMP prototypes, together with appropriate modelling studies.

We expect to have access to a prototype farm that will eventually consist of 24 (to 32) nodes, with each node containing four 500 MHz Pentium II processors. The operating systems envisaged are Windows NT and/or Linux. This prototype is ideally suited for our studies since it combines the advantages of a multi-processor sub-farm with those of a commodity-component machine. It is planned to base this prototype at CERN.

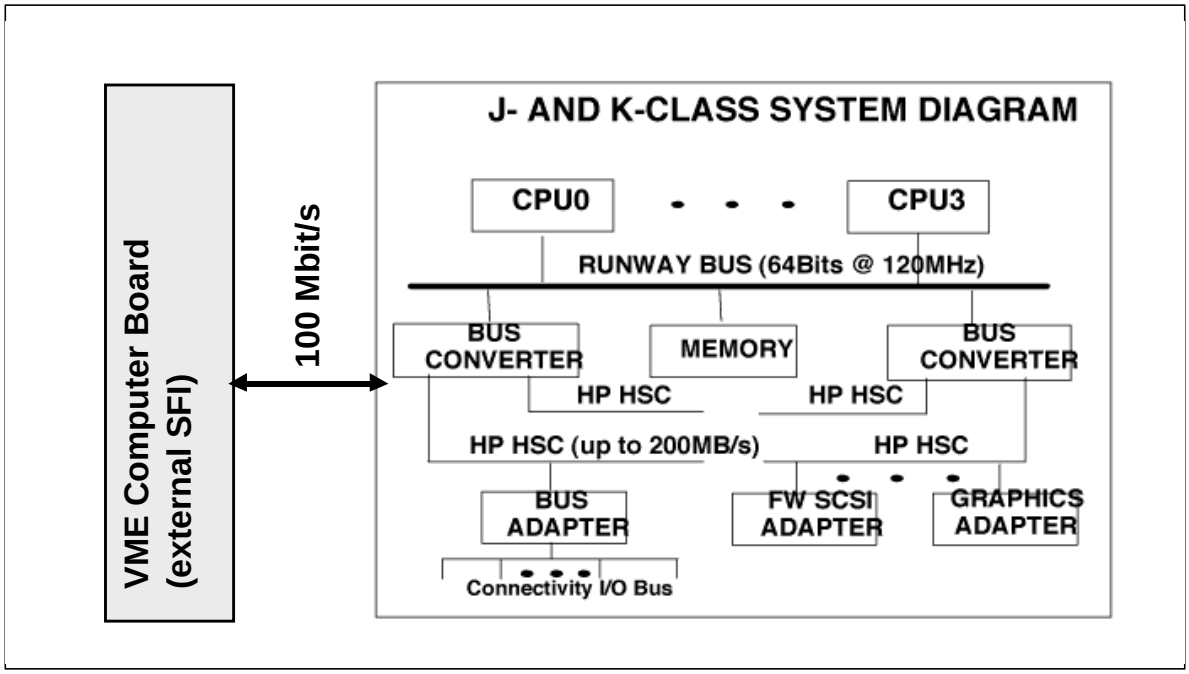


Figure 5-22 Schematic diagram of the SMP prototype subfarm.

There also exist other possibilities which are worth mentioning because they address some very specific architectural points:

- Scalable Parallel Processors (SPP) where independent processors are connected by a high performance switch. The very recently announced architecture S2MP (Scalable Shared-memory Multi Processors) by Silicon Graphics[1] with the *Origin 2000* series should be carefully investigated
- Massively Parallel Processing (MPP) machines which are certainly out of the scope of the DAQ/EF-1 but which should be evaluated in the more general framework of the ATLAS computing needs. Architectures such as the Sandia/Intel Parallel Processor [5-54] which claims 1 Teraflop performance using a Pentium Pro based machine should be investigated, even given their high cost and the fact that they seem more suitable for highly parallel computation applications

The progression of architectures available on the market will be closely followed throughout the lifetime of the project, keeping in mind our strong requirements of scalability and robustness.

5.2.4 Back-end DAQ subsystem

5.2.4.1 Overview

The back-end software encompasses the software for configuring, controlling, and monitoring the DAQ but specifically excludes the management, processing, or transportation of physics data. The back-end software is essentially the ‘glue’ that holds the subsystems together. It does not contain any elements that are detector specific as it is used by many configurations of the DAQ

1. http://www.sgi.com

and detector instrumentation. Figure 5-23 is a basic context diagram for the back-end software showing the exchanges of information with the other subsystems. This context diagram is very general and some of the connections to other subsystems may not be implemented in the DAQ/event filter prototype-1 project.

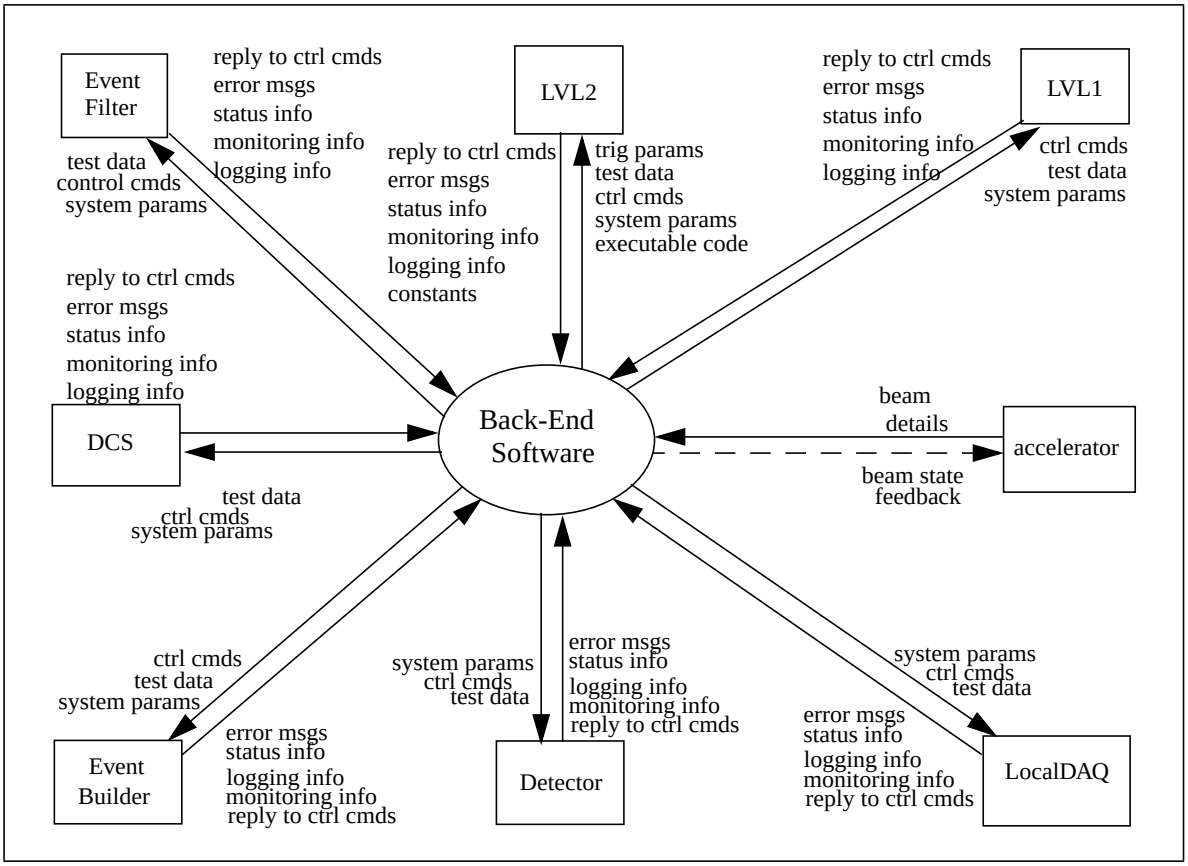


Figure 5-23 DAQ back-end software context diagram showing exchanges with other subsystems.

The back-end software is but one subsystem of the whole DAQ system. It must co-exist and co-operate with the other subsystems. In particular, interfaces are required to the following subsystems of the DAQ and external entities:

trigger

the back-end receives trigger configuration details and can modify the configuration

processor farm

back-end software synchronizes its activities with the processor farm when handling physics data

accelerator

beam details are retrieved from the accelerator and feedback is provided to the machine operators on the state of the beam at the detector site

event builder

back-end software synchronizes its activities with the event builder for the handling of physics data

Local DAQ
to send and receive control, configuration and synchronization information

DCS
receives status information from detector control system and sends control commands

5.2.4.2 Operational environment

It is expected that the operational environment for the back-end software will be a heterogeneous collection of UNIX workstations, PCs running Windows NT, and embedded systems (known as LDAQ processors) running various flavours of real-time UNIX operating systems (e.g. LynxOS) connected via a Local Area Network (LAN). It is assumed that network connections (e.g. Ethernet or replacements) running the most popular protocols (e.g. TCP/IP) will be available to all the target platforms for exchanging control information and that its use will not interfere with the physics data transportation performance of the DAQ.

The ATLAS prototype DAQ system will need to run using all or only a part of its subsystems. It will be assembled in a step-by-step manner, according to financial and technical dependencies and constraints. Many groups of people will interact at the various hardware and software levels, and so a high degree of modularity is needed to connect, disconnect, or add new components at will.

Figure 5-24 shows on which processors the back-end software is expected to run: that is the event filter processors, supervisor and the LDAQ processors in the detector readout crates. Note that the back-end software is not the only software that will be run on such processors. A set of operator workstations, situated after the event filter processor farm, dedicated to providing the man-machine interface and hosting many of the control functions for the DAQ system will also run back-end software.

5.2.4.3 Back-end DAQ software components

The software component model

The user requirements gathered for the back-end subsystem [5-55] have been divided into groups related to activities providing similar functionality. The groups have been further developed into components of the back-end with a well-defined purpose and boundaries. The components have interfaces with other components and external systems. Each component offers some unique functionality and has its own architecture.

From analysis of the components, it was shown that several domains recur across all the components including data storage, inter-object communication, and graphical user interfaces. For more details of the design of each of the back-end components see [5-56].

Core components

The following five components are considered to be the core of the back-end subsystem. The core components constitute the essential functionality of the back-end subsystem and have been given priority in terms of time-scale and resources for development.

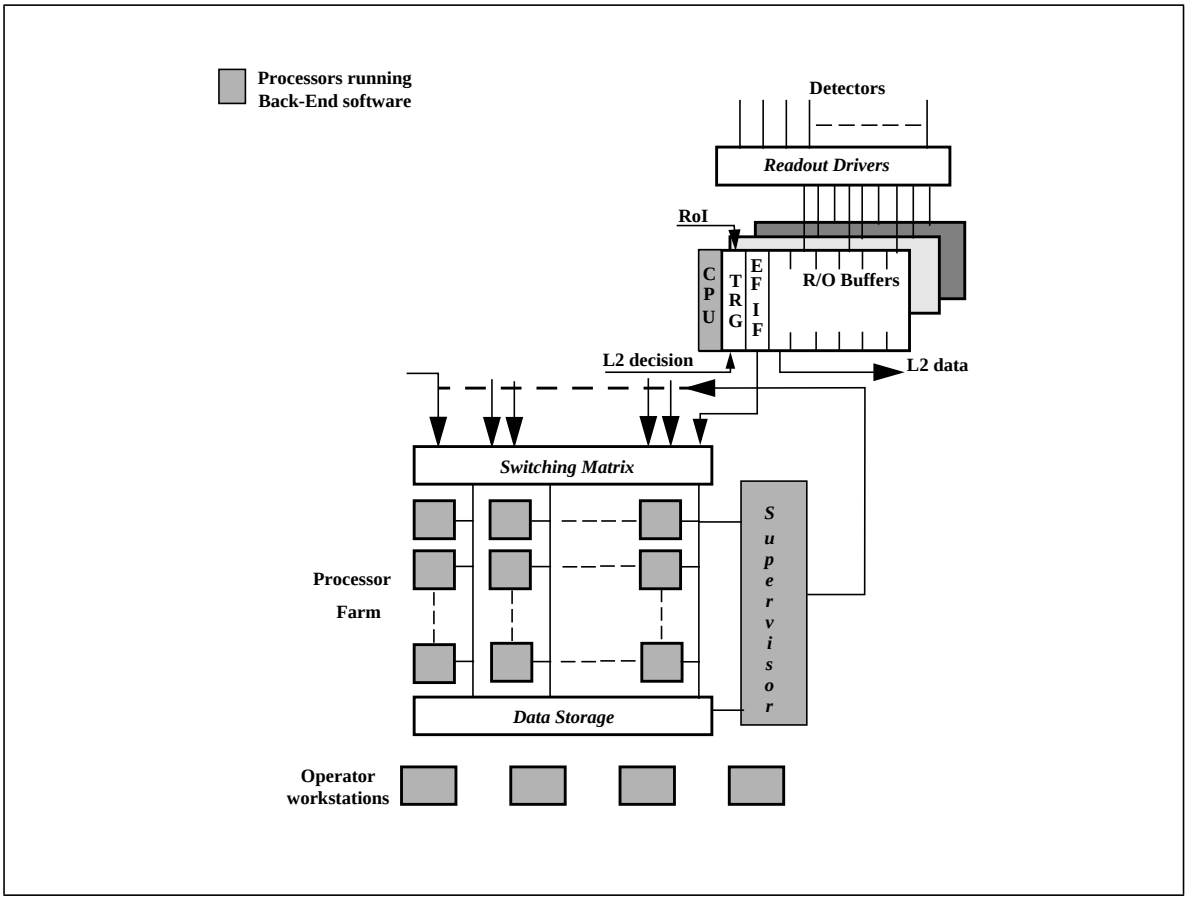


Figure 5-24 Back-End DAQ operational environment.

Run control

The run control system controls the data-taking activities by coordinating the operations of the DAQ subsystems, back-end software components, and external systems. It has user interfaces for the shift operators to control and supervise the data-taking session and software interfaces with the DAQ subsystems and other back-end software components. Through these interfaces the run control can exchange commands, status, and information used to control the DAQ activities.

Configuration database

A data acquisition system needs a large number of parameters to describe its system architecture, hardware and software components, running modes, and status. One of the major design issues of ATLAS DAQ is to be as flexible as possible and so the software is parametrized by the contents of the configuration database.

Message reporting system

The aim of the Message Reporting System (MRS) is to provide a facility which allows all software components in the ATLAS DAQ system and related subsystems to report error messages to other components of the distributed DAQ system. The MRS performs the transport, filtering, and routing of messages. It provides a facility for users to define unique error messages which will be used in the application programs.

Process manager

The purpose of the process manager is to perform basic job control of the DAQ software components. It is capable of starting, stopping, and monitoring the basic status (e.g. running or exited) of software components on the DAQ workstations and LDAQ processors independent of the underlying operating system.

Information service

The Information Service (IS) provides an information exchange facility for software components of the DAQ. Information (defined by the supplier) from many sources can be categorized and made available to requesting applications asynchronously or on demand.

5.2.4.4 Trigger / DAQ and detector integration components

Given that the core components described above exist, the following components are required to complete the back-end functionality when integrated with other online subsystems and detectors.

Partition and resource manager

The DAQ contains many resources (both hardware and software) which cannot be shared and so their use must be controlled to avoid conflicts. The purpose of the Partition Manager is to formalize the allocation of DAQ resources and allow groups to work in parallel without interference.

Status display

The status display presents the status of the current data-taking run to the user in terms of its main run parameters, detector configuration, trigger rate, buffer occupancy, and state of the subsystems.

Run bookkeeper

The purpose of the run bookkeeper is to archive information about the data recorded to permanent storage by the DAQ system. It records information on a per-run basis and provides a number of interfaces for retrieving and updating the information.

Event dump

The event dump is a monitoring program with a graphical user interface that samples events from the dataflow and presents them to the user in order to verify event integrity and structure.

Test manager

The test manager organizes individual tests for hardware and software components. The individual tests themselves are not the responsibility of the test manager which simply assures their execution and verifies their output. The individual tests are intended to verify the functionality of a given component. They will not be used to modify the state of a component or to retrieve status information. Tests are not optimized for speed or use of resources and are not a suitable basis for other components such as monitoring or status display.

Diagnostics package

The diagnostics package uses the tests held in the test manager to diagnose problems with the DAQ and verify its functioning status. By grouping tests into logical sequences, the diagnostic framework can examine any single component of the system (hardware or software) at different levels of detail in order to determine as accurately as possible the functional state of components or the entire system. The diagnostic framework reports the state of the system at a level of abstraction appropriate to any required intervention.

5.2.4.5 Software technologies

Introduction

The various components described above all require a mixture of facilities for data storage, interprocess communication in a LAN network of processors, graphical user interfaces, complex logic-handling, and general operating system services. To avoid unnecessary duplication, the same facilities are used across all components. Such facilities must be portable (the prototype DAQ will include processors running Solaris, HPUX and WNT). In particular they must be available on the LynxOS real-time UNIX operating system selected for use on the LDAQ processors. Candidate freeware and commercial software packages were evaluated to find the most suitable product for each technology. The back-end components use the software technologies described below as a common base. They also have interdependencies between them as represented (in a simplified manner) in Figure 5-34.

General-purpose programming toolkit

Rogue Wave Tools.h++ [5-57] is a C++ foundation class library available on many operating systems (including Unix and WNT) that has become an industrial standard and is distributed by a wide variety of compiler vendors. It has proven to be a robust and portable library that can be used for DAQ programming since it supports many useful classes and can be used in a multi-threaded environment. We have acquired the sources for the library and ported it to LynxOS.

Since this decision, the C++ language has been accepted as an international standard and the Standard Template Library (STL) has become widely available. However, for the length of the current prototype project we shall continue to use Tools.h++ since migration would involve modifying source code.

Persistent data storage

Within the context of the ATLAS DAQ/EF prototype-1 project, the need for a persistent data manager to hold configuration information was identified. The ATLAS DAQ group has evaluated various commercial and shareware data persistence systems (relational databases, object databases and object managers) but no single system satisfied all the documented user requirements [5-55].

As a consequence, it was decided to adopt a two-tier architecture, using a lightweight in-memory persistent object manager to support the real-time requirements and a full ODBMS as a back-up and for long-term data management [5-34].

Lightweight in-memory object manager

For the object manager, a package called OKS (Object Kernel Support) [5-58] has been developed on top of Rogue Wave's Tools.h++ C++ class library. The OKS system is based on an object model that supports objects, classes, associations, methods, data abstraction, inheritance, polymorphism, object identifiers, composite objects, integrity constraints, schema evolution, data migration, active notification, and queries. The OKS system stores database schema and data in portable ASCII files and allows different schema and data files to be merged into a single database. It includes Motif-based GUI applications to design database schema and to manipulate OKS objects (Figure 5-25). A translator has been developed between the OMT (Object Modeling Technique) object model and OKS object model implemented with StP[1] (Software Thru Pictures) CASE tool.

Objectivity/DB database system

Objectivity/DB is a commercial object-oriented database management system introduced to CERN by the RD45 project [5-59]. We evaluated the basic DBMS features (schema evolution, access control, versioning, back-up/restore facilities etc.) and the C++ programming interface. A prototype translator has been developed between the OMT object model and Objectivity object model implemented with the StP CASE tool [5-60].

Interprocess communication

Message passing in a distributed environment is a topic of major importance since it is the communication backbone between the many processes running on the different machines of the DAQ system. Reliability and error recovery are very strong requisites as well as the ability to work in an event-driven environment (such as X11). Many components of the back-end software require a transparent means of communicating between objects independent of their location (i.e. between objects inside the same process, in different processes or on different machines).

We chose to evaluate the Object Management Group's[2] (OMG) Common Object Request Broker Architecture (CORBA) standard. ILU [5-61] is a freeware implementation of CORBA by Parc Xerox. The object interfaces provided by ILU hide implementation distinctions between different languages, between different address spaces, and between operating system types. ILU can be used to build multilingual object-oriented libraries with well-specified language-independent interfaces. It can also be used to implement distributed systems and to define and document interfaces between the modules of nondistributed programs. ILU interfaces can be specified either in the OMG's CORBA Interface Definition Language (OMG IDL) or ILU's Interface Specification Language (ISL). We have ported ILU to LynxOS.

We have developed a package, called IPC [5-62], on top of ILU that hides as much as possible the ILU-specific details with the intention of simplifying the porting of our applications to other CORBA implementations.

Dynamic object behaviour

Many applications within the ATLAS DAQ prototype have complicated dynamic behaviour which can be successfully modelled in terms of states and transitions between them. Previously,

1. http://www.ide.com
2. http://www.omg.org

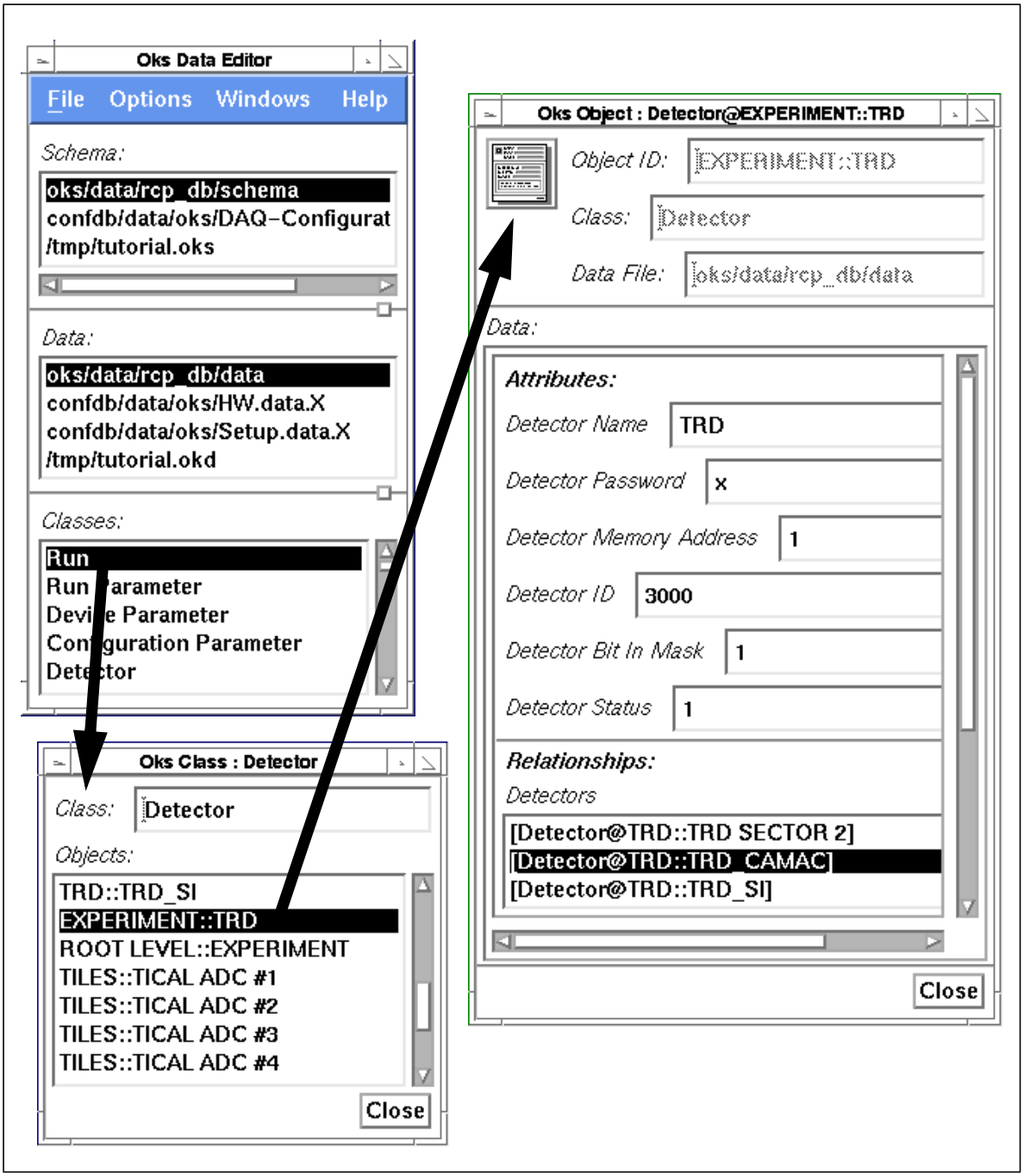


Figure 5-25 The OKS data editor.

state diagrams, implemented as finite-state machines, have been used which, although effective, become ungainly as system size increases. Harel statecharts address this problem by implementing additional features such as hierarchical and concurrent states.

CHSM [5-63] is an object-oriented language system which implements Harel statecharts as Concurrent, Hierarchical, finite State Machines supporting many statechart concepts as illustrated in the abstract example shown in Figure 5-26.

CHSMs are described by means of a CHSM description text file and Figure 5-26 shows the CHSM description corresponding to a statechart. The CHSM compiler converts the description to C++, which may be integrated with user defined code.

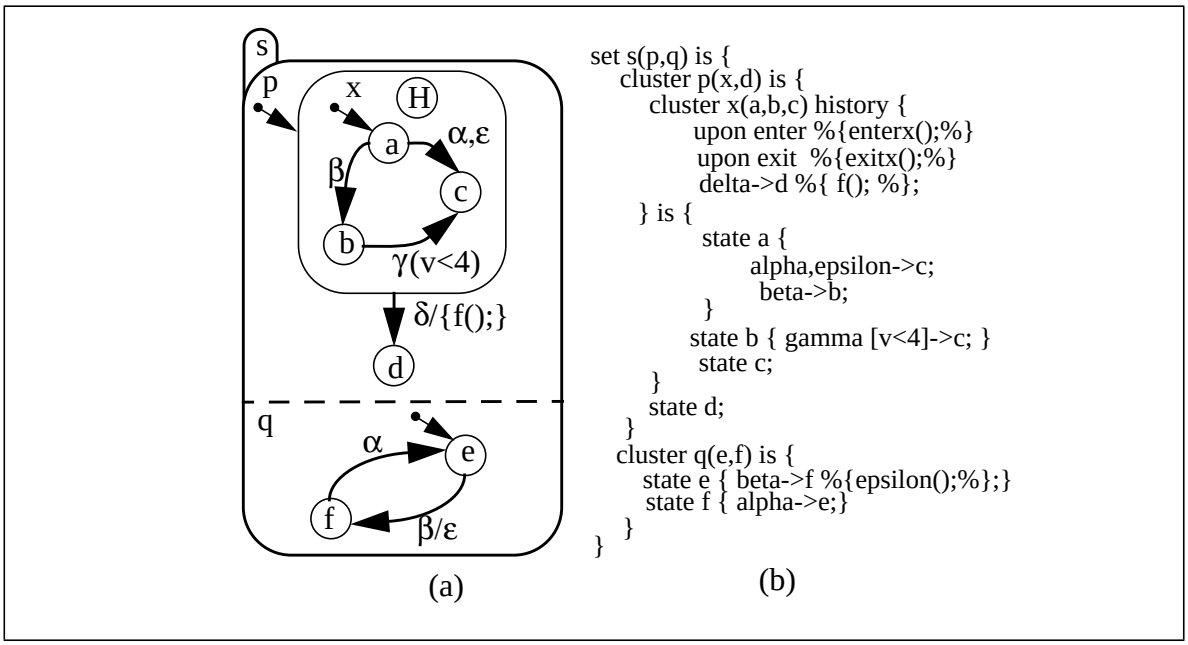


Figure 5-26 Harel statechart example.

We have evaluated the CHSM language system and have shown it to be suitable for describing the dynamic behaviour of typical DAQ applications [5-64]. The run control has been implemented using CHSM for the DAQ supervisor and controllers.

Graphical user interfaces

Modern data-acquisition systems are large and complex distributed systems that require sophisticated user interfaces to monitor and control them. X11 and Motif are the dominant technologies on UNIX workstations but the advent of WNT has forced us to reconsider this choice.

Java

Java is a simple object-oriented, platform-independent, multi-threaded, general-purpose programming environment. The aim of our evaluation was to understand if Java could be used to implement the status display component of the DAQ. A demonstration application was developed to investigate such topics as creating standard GUI components, client-server structure, use of native methods, specific widgets for status displays and remote objects. The performance was compared to X11 based alternatives. The demo contains three essential parts: servers, simulators and applets. Servers realise the binding with remote objects, simulators create simulation data and update remote objects (to mimic the DAQ), applets provide simulation data to remote objects or retrieve data from them for display purposes. The entire application is written in Java (JDK1.0), communication is realised by Java IDL (alpha2). The servers and simulators run on a SUN (Solaris2.5) and the applets can be loaded from any machine using a Java-compatible browser. Work is continuing on the integration of the demo status display with the ILU CORBA system described above.

MVC and X-Designer

X-Designer [5-65] is an interactive tool for building graphical user interfaces (GUIs) using widgets of the standard OSF/Motif toolkit as building blocks that has been used extensively in the RD13 project [5-4]. It is capable of generating C, C++ or Java code to implement the GUI. We investigated the construction of GUIs with a Model-View-Controller (MVC) architecture using the C++ code generation capabilities of the tool.

A number of widget hierarchies were created which correspond to commonly used patterns in GUIs (e.g. data entry field with a label). These also correspond to views or controller-view pairs in the MVC architecture. By making each hierarchy a C++ class it could be added to the X-Designer palette and used in subsequent designs. A GUI for the existing RD13 Run Control Parameter database was successfully built using these definitions.

Work will continue to investigate if the MVC approach and Java code generation facilities of X-Designer can be used to develop the DAQ graphical user interfaces. X-Designer is currently being used to develop a GUI for the run control component.

More details of the software technologies described above and their application to the back-end DAQ are given in [5-66].

5.2.4.6 Software process

This section describes the software process that is being used to manage the back-end DAQ subsystem project.

The software process is the set of tools, methods, and practices used to produce a software product [5-67]. The objectives of software process management are to produce products according to plan while simultaneously improving the group's ability to produce better products [5-68]. A software process [5-69] has been proposed for the offline software and has many points in common with the process described below.

Phases

The development of the back-end DAQ has been divided into a number of sequential phases intended to help pace and organize the work. Each phase has been defined to produce an obvious deliverable (i.e. document or code). The requirements phase helped enormously in defining the scope and boundaries of the project and showed differences of points of view of the people involved. The division into phases of the project has also helped everyone to understand better what is expected of them in their current work.

Organization

The people involved in the back-end DAQ come from a number ATLAS institutes and have not, in general, been able to work full-time on the project. Faced with this situation, we have tried to organize the work along the component structure. Typically a single institute has taken responsibility for developing a component during a particular phase. Such component groups are small (up to a maximum of five individuals) and since they tend to be in the same institute the localized development has simplified communication and reduced travel. The same individuals have tended to follow a single component through the various phases and hence ensured the continuity of the work.

Tools and methods

The use of the software development environment is described below. To summarise we have found that it has been more important to agree on a design method and notation than a common CASE tool. We have found that the design method gives a common language between groups and individuals which helps dispel misunderstandings and identify differences of opinion earlier in the life cycle.

Future

Because of the time-scale of the project, we have only gained experience with the earlier phases of the software life-cycle. In the short-term, we shall address the later phases of the cycle including further unit testing, integration, deployment and upgrades.

Project phases

The phases followed during the software life cycle for the back-end DAQ are now described.

- Collect requirements

  The goal of this phase was to collect the requirements for the back-end DAQ by recording them in a document and to define a work-plan for the next phase. It was organized via a working group of 19 individuals including DAQ members and detector representatives which met 9 times during 4 months (Jan. to Apr. 1996). The ESA-PSS05 Framemaker template from ECP/IPT group was used for the document.

  The document was announced for review at the Trigger/DAQ meeting of March 1996 ATLAS week and comments (very few) were received and incorporated. We also produced a summary document which did not contain detailed requirements but was shorter and easier to read. Representatives of the group visited LEP experiment sites to discuss back-end issues and compare the requirements specified in the document against working systems. The document divided the software into components and the work-plan ordered components according to priority.

  Based on the requirements, a number of common issues (e.g. access to persistent data, graphical user interfaces, interprocess communication etc.) were identified across the components and used to organize the following phase.

- Predesign investigations of candidate technologies

  The goal of this phase was to identify and evaluate candidate technologies and techniques capable of addressing the common issues identified from the requirements document. An evaluation note, established against an agreed set of criteria was produced for each evaluation conducted. The evaluations were made by small groups (one for each technology). This phase took five months (Jun. to Oct. 1996) and each evaluation note was reviewed in the back-end DAQ meetings. Based on the results a single technology was selected for each area (except GUIs: Motif & Java).

- High-level design

  The goal of this phase was to produce a design for each component covering the most important aspects. The component designs were documented with a short textual overview, descriptions of external interfaces and diagrams taken from the OMT/Booch methods, produced with StP. A small design group was organized for each component. OMT/StP training was organized on the CERN site and a central software repository created. This phase lasted five months (Oct. 1996 to Apr. 1997) and each high-level design document

was reviewed in back-end DAQ meetings leading to several revisions of the documents as the designs evolved.

At this phase the Partition Manager component was frozen and work continued on the more clearly defined resource manager. The need for the Information Service (i.e. general online information exchange) was also discovered during this phase.

- Detailed design, implementation and unit testing

  The goal of this phase was to produce unit-tested implementations of the ‘core’ components according to the high-level design. The deliverables include the code and test programs as well user and programmer documentation. Small groups were formed to work on each component with usually the same individuals following a component through definition and design. Initial work on the core components started in Apr. 1997 and became available for use at Christmas 1997. Inspections are being organized to review the code and documentation, starting with the basic components and working upwards according to the package dependencies (see Figure 5-34).

  The unit tests are based on use-cases identified in the high-level design documents. We are now producing test-plans which we would have preferred to do at the high-level design phase but met with limited success owing to a lack of information concerning their content. The unit tests are kept with the implementation in the SRT code repository.

- Integration

  The integration of the components is being made according to the package dependencies (see Figure 5-34). The integration with the data flow software is being defined at the project level on a package-per-package basis.

- Deployment

  Access to the software is made through AFS during development. Because of the limited reliability of Andrew File System (AFS), Network File System (NFS) based access is preferred during use of the software in test-beam situations. Work is going on within the configuration management tools to ease the migration and deployment of the software to the installation site.

5.2.4.7 Software development environment

The Software Development Environment is everything needed on the developer's desktop in order to participate in the orderly development or modification of a software product. It should be a fully integrated, operational software environment, and not a collection of individual task-oriented tools if it is to be of most use to the software developers. Elements may include but are not limited to, CASE tools, documentation and testing tools, compilers, linkers, debuggers, programming languages, class libraries and project and configuration management tools. ATLAS has defined requirements for a software development environment [5-68].

Within the back-end DAQ, elements of the software development environment have been assembled to cover the analysis, design, implementation, and testing phases of the software life cycle. Such elements have been selected to be compatible with similar work in the offline software groups and include:

object-oriented method

A mixture of OMT and Booch has been used to cover as many aspects as possible of the analysis and design phases.

design CASE tool

The Software Thru Pictures CASE tool has been used during the high-level design phase of all components. The facilities of the CASE tool have been extended to import requirements from FrameMaker documents and generate code for OKS (see page 184) and CHSM (see page 185) [5-60].

configuration management system

The Software Release Tools[1] is a product developed by ATLAS for the configuration and archiving of software packages. All software developed in the back-end DAQ is maintained using SRT.

programming tools

Debuggers, interactive development environments (e.g. Microsoft Visual C++) and various other programming tools have been installed on the main development machines.

testing tools

Tools to produce static software metrics (e.g. Logiscope[2]) and code coverage measurements (e.g. Insure++[3]) are being used for the testing of the components and have been integrated with the configuration management system with the aid of IT/IPT group.

training

Informal tutorial and official CERN training sessions have been organized for all of the above elements of the software development environment.

5.3 Use of the DAQ/EF -1 system

5.3.1 Dataflow

5.3.1.1 Introduction

One of the main objectives for an integrated dataflow subsystem is that of using it for studies aimed at evaluating and assessing DAQ architectural and operability issues.

We plan to address the studies along two complementary directions:

- Those which address local aspects of the system, where local means both a single element, such as the ROB, and an entire subsystem, such as a full ROC. Studies in this group broadly address the two following issues:
    a. Can performance be achieved, locally (e.g. in the case of the dataflow control within a ROC), which is close to the final performance required by ATLAS? If this is not

1. http://wwwcn.cern.ch/~lat/exports/srt/manual.ps.gz
2. http://www.cern.ch/PTTOOL/Logiscope/
3. http://www.cern.ch/PTTOOL/Insure

the case the reasons should be understood: inadequate technology, architectural bottleneck, software, interaction between any of the above, etc.

b. Functionality and performance at the component level, with emphasis on issues common to different architectural variations, such as the event builder DFM.

- Those related to the global dataflow subsystem aspects, i.e. relative to the overall dataflow subsystem, and as a function of architectural options.

In the following sections we discuss in more detail the dataflow performance studies motivated above.

5.3.1.2 ROC performance

Overview

The ROC consists of subsystems which in turn can be decomposed into a number of software and hardware elements. In order to understand the global performance of the ROC, the performance of these elements and subsystems has to be measured. We list these measurements and mention in each case their present status which in many cases depends upon the availability of the hardware. The current implementation of the ROC is based on VMEbus with the TRG, EBIF and ROB modules being PowerPC-based CPU modules from CES[1].

Given the architectures described in Sections 5.2.2.4 and 5.2.2.5, an implementation must start to address the following points:

- How far can one go by using Commercial Off The Shelf (COTS) components e.g. VMEbus and Real Time Operating Systems. What performance cost ratio is obtainable?
- What is the role for desktop PCs and commodity operating systems e.g. Windows NT?
- What are the pros and cons, performance and cost, of a physical implementation of a EBIF? In particular, what are the requirements on a link technology to perform data collection efficiently?
- What are the pros and cons, performance and cost, of a physical implementation of a TRG? In particular, could an implementation receive data control messages from a level-1 and / or level-2 trigger system and distribute these messages at 100 kHz? What are the requirements on an appropriate link technology? Efficiency would require low message passing overheads, on account of the small size of the messages and broadcast functionality.
- The ROB is the most demanding object within a readout crate. Indications are that COTS components alone may not meet our requirements, because of the large input frequency and message size. With this in mind, where is it best to apply custom components? How much functionality needs to be customized? Do we need a communications processor or a ROB-IN?

Element measurements

ROB input. The ROB input task, see Section 5.2.2.5, is the most demanding one in the DAQ system, as far as performance is concerned. It has to sustain the reading of the order of 1 kbytes of

1. http: / / www.ces.ch

data at a frequency of 100 kHz. Basic measurements of the input for the VMEbus single board computer based ROB have been performed using an S-Link interface [5-70] and different input emulations [5-71]. As an example we show in Table 5-1 the summary measurements in the case where the ROB input is emulated by a SLIDAS; the case of a simple S-link interface handler (simple destination) as well as the one for simple ROB application (including also full buffer management and dummy output task) are shown for two message sizes.

Table 5-1 SLIDAS performance (RIO2 8062).

Message size	Application	Transfer time (us)	Frequency (kHz)	Bandwidth (MB/s)
64	simple destination	8.3	120.4	7.3
64	ROB-like	15.0	66.6	4.1
1084	simple destination	21.0	47.6	49.2
1084	ROB-like	27.5	36.4	37.6

Figure 5-27 summarizes the SLIDAS based measurements.

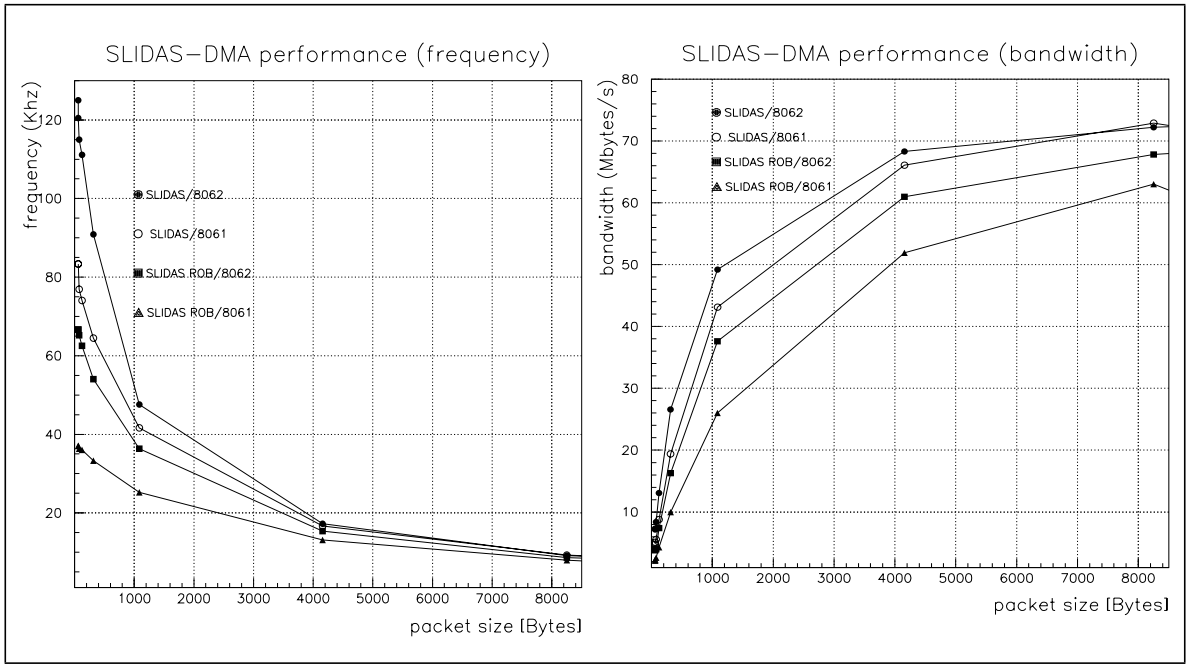


Figure 5-27 SLIDAS performance.

The VME SBC ROB implementation studies have to be completed by also taking into account the full ROB application and the interaction of the ROB with the rest of the ROC. We also plan to follow up the evolution of VME SBC modules [5-72], in particular new modules with both more powerful processors and multiple PCI buses which potentially provide a commercial solution to the bottlenecks of the VME SBC solution.The following step will be that of integrating a ROB-IN like [5-23] device, capable of autonomously serving both the input channel and the buffer management, and measuring the performance of the combined ROB. Comparisons with the SBC-based ROB will be performed, in particular with the view of assessing the potential of fully commercial solutions to provide a continuous path towards the final ATLAS performance.

TRG input. The TRG is another element receiving input, in the form of short control messages, at a rate of 100 kHz; as such it is also critical to the performance of the ROC. Basic performance measurements [5-17] have been done on the TRG input channel using a simple PMC to emulate the input. More realistic measurements will have to be done once the technology connection to high-level triggers is better understood.

Intra-crate links. Initial measurements on the prototype ROC [5-17] have shown the current inadequacy of a single VME bus to support both the dataflow control traffic (*TRG module messages*) and dataflow data traffic (*data collection*) in the ROC. Figure 5-28a shows the rate of TRG messages in the case where ROBs and EBIF modules are emulated (i.e. it depicts the performance of the TRG application only). The need for a secondary bus and/or a VME bus with both higher bandwidth and especially broadcast support has emerged [5-17]. Work on the VMEbus side will have to include both more detailed measurements of its performance and tracking its evolution. Since the need for secondary buses has emerged, evaluation and measurements of possible candidates, such as the PVIC[1] and Raceway[2] buses, will be performed.

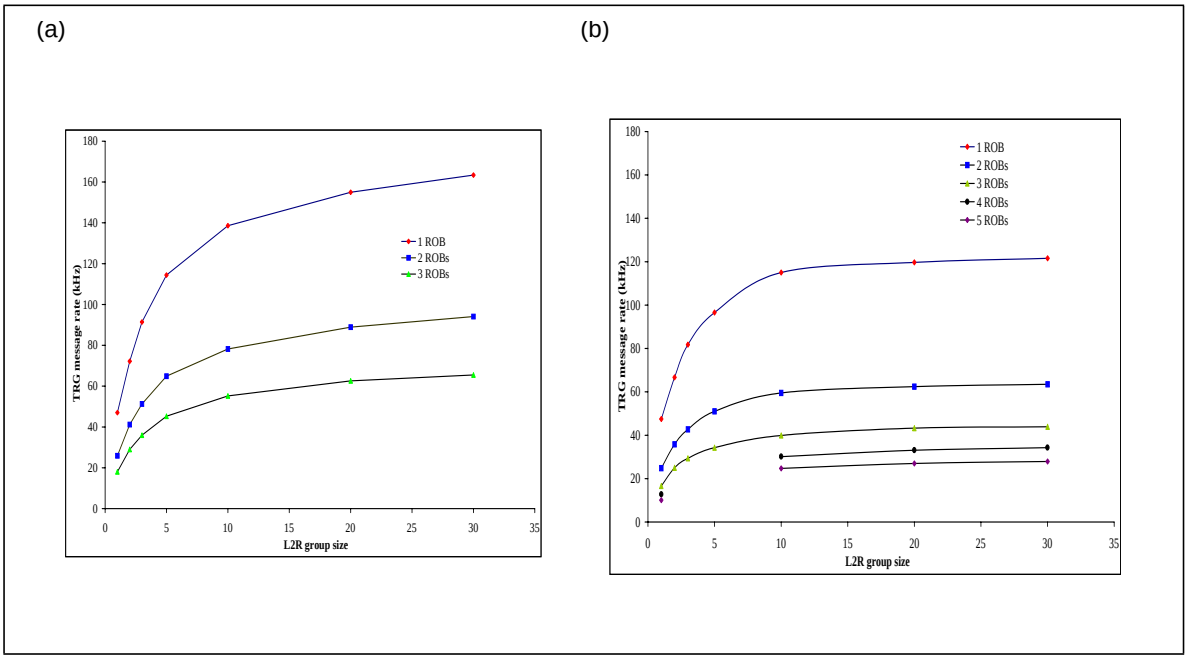


Figure 5-28 (a) TRG performance with emulated EBIF and ROB. (b) performance with full TRG, EBIF and ROB application.

Software performance. The performance of the software running in the critical elements, typically the TRG and ROB, will have a serious impact on the overall ROC performance. We plan to undertake a detailed performance study of software components such as the event manager, the I/O scheduler, and the libraries serving hardware elements.

Subsystem measurements

ROC DAQ-Unit. The overall performance of the ROC DAQ-Unit subsystem, including the TRG, the EBIF and several ROBs, has to be studied in detail. Measurements based on the current prototype implementation have been performed [5-17]. Figure 5-28b depicts the behaviour of the

1. http://www.ces.ch
2. http://www.mc.com/raceway

TRG module, expressed as number of messages per second versus the grouping of Level-2 rejects, in a functional ROC (i.e. including also data collection). These detailed measurements will be continued on alternate implementations, e.g. based on different hardware platforms or multiple buses, as well as on different architectural options, such as combined ROB and EBIF.

Multi-function I/O module. In addition to the baseline implementation, which assigns a DAQ-Unit function to separate modules, we shall study also alternative implementations which will combine two or more functions: ROB and EBFI, ROB and TRG, ROB and TRG and EBIF. For each of those we shall perform detailed performance measurements so as to be able to validate the various architectural options.

EBIF performance. The EBIF functionality in the ROC consists of performing the data collection function in response to Level-2 accept commands received from the TRG module. Some preliminary study, where all the internal ROC activity was performed on VMEbus, have been made [5-17]. We plan to study the performance and the functionality of the EBIF, with emulated DFM and event builder link, in particular with respect to the use of secondary buses, such as PVIC or Raceway, for data collection.

5.3.1.3 Event builder studies

The performance of the event builder subsystem is addressed according to two lines of analysis: one is related to candidate switching technologies, the second takes into account the specific event builder application.

- A number of switching technologies, ATM, Fibre Channel and switched Ethernet, are potential candidates for the physical implementation of the event builder. The technology functionality and performance, including switches, network interfaces, related system software and their interaction, have to be studied. The bulk of these studies has been completed, and a number of conclusions has been drawn [5-73]. Some more studies with cascaded switches in general as well as with the switched Ethernet technology are still under way.
- The event builder is a very special application characterized by two traffic patterns. The unidirectional data traffic where large messages (1 to several kBytes) from all the Event Builder inputs are directed towards the same output port. The control traffic characterised by a potentially large number of small messages (few tens of bytes) with bidirectional and broadcast traffic patterns. Performance studies have therefore to be made on the basis of the typical use and traffic patterns outlined above. The capability of the technology to deal efficiently with the typical event builder traffic patterns, the efficiency and effectiveness of the event builder high level protocol and its physical implementation have to be studied. This will be the main direction for the event builder studies in the next months.

Switching technologies

Current technologies investigated for the DAQ/EF prototype -1 are ATM, FibreChannel (FC) and Switched Ethernet. These technologies have been evaluated in terms of latency and maximum bandwidth in an environment close to the expected final environment of the DAQ/EF prototype -1. For each technology PCI interfaces have been used, either in the PCI or the PMC form factor. Single-board computers (RIO2) running LynxOS or personal computers (PC) running Windows NT have been used as host systems. Test programs to measure the performance

were developed on top of existing system level software, TCP/IP protocol or a native transfer protocol. For FibreChannel on LynxOS our own system software had to be developed.

Evaluation of ATM technology

The ATM technology as one candidate for constructing EB systems has already been investigated thoroughly in the CERN DRDC RD31 project [5-74]. Some studies have been repeated using the 155 Mbit/s standard (140 Mbit/s payload bandwidth) to integrate this technology in the ATLAS DAQ prototype [5-73].

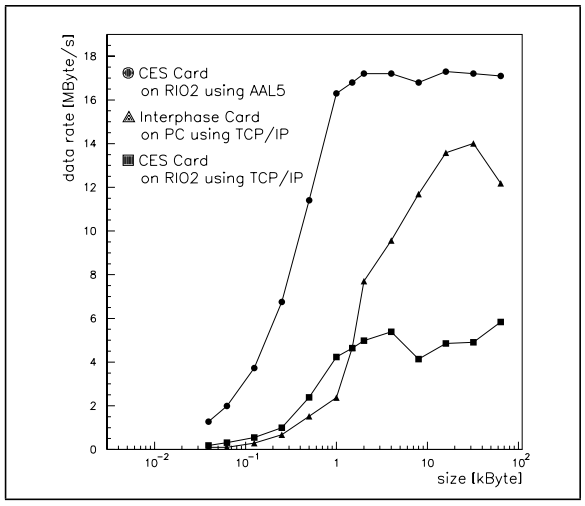


Figure 5-29 Data Rates for ATM Interfaces.

Two different sets of measurements have been carried out in order to estimate the overhead for connection-oriented (TCP/IP) and connectionless (AAL5) communication protocols. The set-up for these tests was based on two RIO2 connected to two ATM/PMC interfaces from CES[1] and on two Pentium PCs with ATM/PCI interfaces from Interphase[2]. The CES interfaces have been used both with the native AAL5 communication protocol and with TCP/IP, while the Interphase interfaces have been tested using TCP/IP only since the driver to access directly the ATM adaptation layer is not yet available

Figure 5-29 shows the different data rates obtained: the TCP/IP protocol (using option *TCP_NODELAY*) induces a substantially higher *overhead* (~150 µs compared to ~20 µs with the AAL5 protocol). The *speed* as measured with the CES interfaces using AAL5 is 17.5 MByte/s corresponding to 100% of the bandwidth available to the user. Because of the limited size of their receive buffer queue the CES interfaces reach data rate of only 6 MByte/s when the TCP/IP protocol is used. The same limitation occurs at much higher packet sizes in the interfaces from Interphase allowing for a throughput of about 15 MByte/s corresponding to ~85% of the bandwidth available to the user.

The high overhead of the TCP/IP protocol which dramatically degrades the transfer performance of packets with sizes interesting for EB purposes (1 to 10 kbyte) leads to the necessity of applying connection-less communication protocols as for example AAL5.

An ATM switch from the Fore Systems, Inc.[3] has been used to repeat the tests previously performed by the RD31 project. More details can be found in [5-74]. The results were confirmed and the switch can be used for the event builder of the DAQ/EF prototype -1. An implementation of the event builder I/O library using ATM through TCP/IP will be developed.

Evaluation of FibreChannel technology

Two types of FC/PCI interfaces have been tested: the FibreXpress FX/PMC interfaces from Systran Corp.[4] have an internal memory into which data have to be moved before they are picked

1. http://www.ces.ch
2. http://www.iphase.com
3. http://www.fore.com

up by the FC protocol chip. The 5526 PCI FC adapters from Interphase[1] have a PCI bus controller bridging the PCI bus directly to the FC protocol chip. In order to use them on the RIO2 they have to be connected using PCI/PMC adapters from Technobox Comp[2]. Both FC interfaces use the Tachyon chip from Hewlett-Packard as the FC protocol chip and use multimode optical link modules running at 1.0625 Gbit/s for the physical medium. For running on PCs a commercial driver was used, while for running on RIO2s our own system software was developed.

The performance of the FibreChannel (FC) has been measured in point-to-point, arbitrated loop and fabric topology, the latter one together with a switch. Measurements were based on test programs and on a FC analyser. The following will summarize the most important results, further details can be found in [5-75]. The data rates measured in point-to-point connections for both types of FC interfaces are shown in Figure 5-30. The *overhead* is about 10 µs for the Interphase interfaces and about 20 µs for the Systran interfaces. The *speed* values are about 64 MByte/s for the Interphase interfaces and about 55 MByte/s for the Systran interfaces. While the *overhead* probably is dominated by software, the *speed* is limited by the PCI bus on the pairs of host and FC interface. The FC physical bandwidth is used to 55% (Systran) or 64% (Interphase).

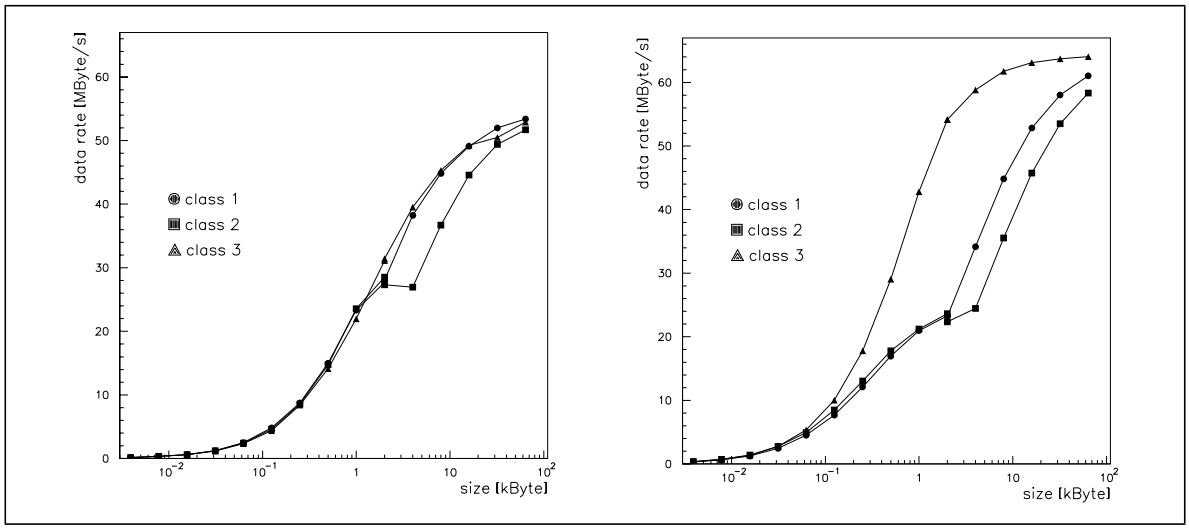


Figure 5-30 Data rate for the Systran and the Interphase FC interfaces.

The angle in the curve for class 1 and class 2 transfers can be explained by the parallelism when sending multiframe sequences, while single-frame sequences have to wait for acknowledgements: they are sent in a sequential way. For the Interphase interfaces, class 1 transfers have a slightly better performance than class 2 transfers.

The tests have also been repeated in different combinations of the FC interfaces. It turns out that the interfaces from the two different manufacturers interoperate without any problems. Tests with two Systran interfaces on Pentium 166 MHz PCs have been performed. The cards use a lightweight transport protocol (FXLP) provided by Systran. DMA measurements of data transfers between the PC's memory and the memory of the FC/PMC card have been carried out: the *overhead* is about 50 µs, the *speed* value from the FC/PMC interface to the PC's memory is about

4. http://www.systran.com
1. http://www.iphase.com
2. http://www.technobox.com

85 MByte/s, and about 100 MByte/s in the other direction. The data rates for transfers between two PCs in point-to-point connection show that the *overhead* is about 75 µs and the *speed* value is about 40 MByte/s. The same lightweight protocol can be used between PCs and RIO2s and similar rates have been observed.

In an FC-arbitrated loop several nodes share the same medium. The Systran and Interphase interfaces have been connected on an arbitrated loop and transfers in class 2 and class 3 have been tested successfully between any pair of them. The limitation of the classes comes from the Tachyon chip which does not support class 1 on an arbitrated loop.

The degradation of the data rate for single transfers due to the additional overhead from the arbitration phase compared to point-to-point transfers is of the order of a few per cent for most packet sizes, except for packet sizes of a few per cent of the frame size where the degradation is of the order of 10%. This has been tested with up to two concurrent but mutually exclusive transfers.

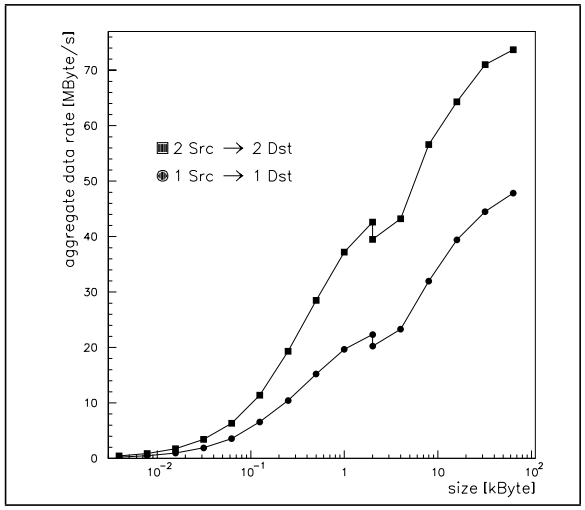


Figure 5-31 Event building using arbitrated loop (class 2).

On an arbitrated loop several transfers can overlap. In class 3, however, frames can be lost when several transfers go to the same receiving node. Tachyon can only handle one incoming multiframe sequence and will drop all incoming class 3 frames not belonging to the actual sequence. Using class 2, Tachyon has the same limitation and will also drop incoming class 2 frames which do not belong to the actual multiframe sequence, but in addition it will send back an "*N_Port Busy*" frame. The sending node can then retry to send this sequence. This can be handled by Tachyon automatically up to 16 times for the first frame of a sequence. If this is not enough, then the user level of the software has to take an action

Overlapping of class 2 transfers has been tested in an EB-like application where two nodes on the arbitrated loop act as EB sources and two as EB destinations. The data rate vs. packet size is shown in Figure 5-31. The aggregate data rate does not scale with the number of destinations since the maximum bandwidth of this configuration is limited to 100 MByte/s and the nodes spend some time for the arbitration. However, an increase of about 50% between the 1×1 EB-like system to the 2×2 EB-like system can be observed and a total *speed* value of 80 MByte/s has been fitted in the latter case, corresponding to 80% of the maximum bandwidth. This value is nevertheless not reached for packet sizes of 64 kByte because of an increased *overhead* value of about 40 µs and will only be reached at packet sizes of about a few hundred kByte.

The FC interfaces have been tested successfully with the *SilkWorm* FC switch from Brocade Communication Systems, Inc.[1]. This switch supports class 2 and class 3 transfers and allows full connectivity between any of its ports. Comparisons of point-to-point transfers and single transfers going through the switch have shown a delay of the sequences through the switch to be of about ~2 µs, confirming the manufacturer's specification. This was confirmed by using the FC

1. http://www.brocadecom.com

analyser. No limit on the bandwidth was observed. In fact, it turns out that the pairs of host and FC / PCI interface are the bottleneck.

Comparisons of single transfers using the switch and concurrent but mutually exclusive transfers using the switch have shown no observable degradation in the transfer parameters. Up to three concurrent transfers have been shown to have scaling behaviour: the data rates of the transfers add up. The switch has been loaded with about 200000 frames / s (for packets of 4 Byte) and with about 180 MByte / s (for packets of 64 kByte). This is only a small portion of the load the switch can accept according to the manufacturer which states that the switch is capable of accepting 8 million frames / s and 3.2 Gbyte / s aggregate bandwidth.

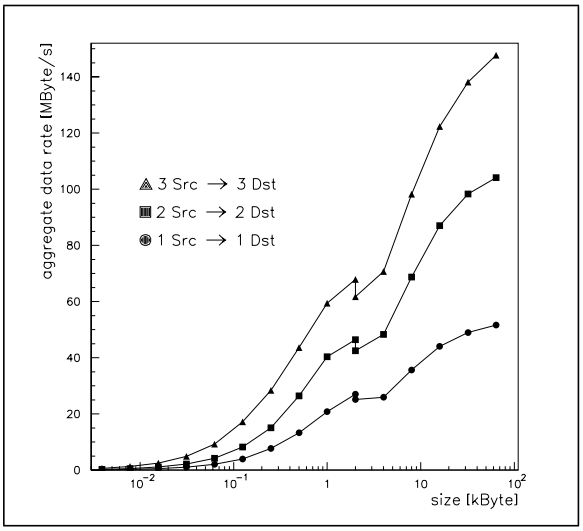


Figure 5-32 Event Building using an FC Switch.

EB-like transfers have been tested with up to six FC interfaces. The data rates are shown in Figure 5-32, for 1×1, 2×2 and 3×3 EB-like systems, all using class 2 transfers.

EB using the switch shows very clearly scaling behaviour of the data rate with the number of destinations. In particular, the data rate for packet sizes of about 10 kByte is 35 MByte / s for each destination. This corresponds to about two to three times the performance required by the ATLAS experiment.

The *SilkWorm* switch further allows multicast and broadcast. Multicast is defined for class 3 transfers and requires registration of the nodes in multicast groups. The registration is handled by an alias server which is an integral part of the *SilkWorm* switch. Broadcast is also defined for class 3 but does not require registration and uses an implicit address identifier instead. The protocol to register multicast groups with the alias server of the switch has been implemented and multicast and broadcast of transfers have been tested successfully.

Soon the SilkWorm switch will also allow arbitrated loops to be attached directly to the switch. This will allow EB systems to be built in which arbitrated loops are used to group several EB sources sharing the bandwidth of the arbitrated loop before they are connected to a network of switches. The number of switch ports can thus be reduced and each switch port will be used much more efficiently. This will reduce the cost of the system considerably.

Evaluation of Switched Ethernet technology

Gigabit Ethernet technology promises to be a cheap alternative to other technologies. It will offer high bandwidth and a high degree of connectivity at a lower price. This technology is in the tradition of Ethernet technology and it is to be hoped that most of the higher-level protocol and the user applications can be ‘inherited’ without much modification.

No equipment for Gigabit Ethernet technology has been available so far. However, in order to learn about Ethernet technology for event builder systems, Ethernet equipment running at 100 Mbit / s has been tested. It it to be hoped that this technology can be regarded as one step in the migration to Gigabit Ethernet.

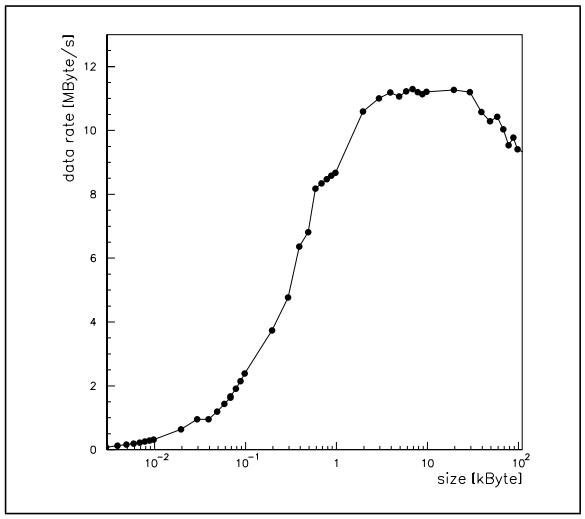


Figure 5-33 Data Rate of 100 Mbit/s Ethernet Interfaces.

100 Mbit/s Ethernet/PCI interfaces from Intel Corp.[1] have been tested between PCs. Using a TCP/IP driver (without the option *TCP_NODELAY*) the data rates shown in Figure 5-33 could be measured. The *overhead* value is about 30 µs and the *speed* value is about 11 MByte/s which corresponds to about 93% of the physical bandwidth available to the user. For packets of a size bigger than 30 kByte, however, the performance drops to about 9 MByte/s, probably due to an internal memory limitation of the PC.

Tests with a switch from 3COM[2] have been performed. The switch, which uses a store-and-forward technique, introduces an additional delay of 10 µs for packets of minimum size. This delay increases with the size of the packets by about 73 µs per kByte. The possible use of other switches allowing a cut-through technique is envisaged. An implementation of the event builder I/O library using TCP/IP over Ethernet has been implemented and tested successfully. This implementation can use 100 MBit/s Ethernet interfaces, as well as the 10 MBit/s interfaces without any modification in the software.

Conclusion

Several technologies for the high-speed interconnect in an event builder have been tested. It has been found that ATM and FC do provide the necessary components. Both technologies have mature products which can be used for EB systems or other components in the DAQ prototype. Differences in performance and cost will have to be understood in the background of the requirements of the ATLAS DAQ prototype. It is still too early to conclude on 100 MBit/s Ethernet technology, and Gigabit Ethernet technology will be investigated when equipment becomes available.

The tests have shown that the role interfaces play in the construction of an EB system are very important. The scaling of cascaded switches and the construction of large networks needs to be tested and to be supported by simulation. One of the goals to be achieved in the ATLAS DAQ/EF -1 prototype is certainly to obtain a better understanding of the scalability issue.

Another important issue, going beyond the rather technology-oriented studies presented in this paper, is the integration aspect of an EB system and of the ATLAS DAQ prototype. The interfaces, drivers, libraries, and switches have to be assembled in such a way that they can be used with the event builder I/O library. Interfacing between the high-speed interconnect technology and the dataflow systems in terms of hardware and software is an important issue and has a large influence on the performance.

1. http://www.intel.com
2. http://www.3com.com

Event builder functional and performance studies

In addition to the evaluation of candidate switching networks on a pure technological basis, the most important studies in the area of the event builder are related to the use, function, and traffic pattern typical of the application.

Event builder functionality. Studies and related measurements will have to be undertaken to assess the suitability of the event builder high-level design, as outlined in Ssection 5.2.2.4 in particular as regards the protocol defined to support the event building application.

Reliability. The baseline event builder architecture will be studied with respect to errors, malfunctions, destination occupation.

Event Builder performance. A number of detailed performance measurements will be performed on the baseline event builder architecture and using different technologies. For example: the sustained rate (frequency of events) and throughput, the influence of the event size variations, the influence of the typical event builder traffic pattern on the technology performance, etc.

Event Builder scalability. At the level of DAQ/EF-1 the event builder can only be studied on a small-scale, e.g. on an 8×8 configuration. A major question is therefore how laboratory measurements obtained on a small scale system can be extrapolated to one of the size of the final ATLAS event builder. The size of the final ATLAS event builder will be determined by the amount of grouping which can be done both at the source level, i.e. how many ROD links can be grouped into an event builder input, and at the destination level, how many processors can be attached onto an event builder output. This is a question which in addition to the pure event builder issues also spans the implementation of the ROC and subfarm: the amount of grouping determines the size of the switch. The behaviour of the event builder as a function of its size will be studied by simulation.

Dataflow Manager. The DFM is a key element for the event builder functionality and performance. We shall study its performance as a function of different architectural and implementation options: a centralized versus a distributed DFM, use of message passing versus shared memory, the use of the same (as for the event fragments) network versus a secondary one dedicated to the event builder control messages.

Global dataflow system performance

Global dataflow system studies will be performed on the integrated dataflow set-up consisting of ROCs, an Event Builder and subfarm DAQs (using a dummy event handler). The laboratory set-up will be configured according to different architectural options:

- one ROC per Event Builder input: this is the case where the ROC includes a separate EBIF module and the data collection function;
- one ROB per Event Builder input: this is the case where ROB and EBIF functionality is collapsed into a single module;
- one or more event builder output(s) per subfarm;
- different DFM implementations.

At the level of the global dataflow system we shall address the following issues:

- The evaluation of the overall performance and operability of the implementation. Here we are not addressing the final ATLAS performance, in particular because the size of the

system is too small. Nevertheless there will be a number of conclusions which we shall be able to draw on the global performance.

- Assessment of the functional architecture under the different alternative options (such as the possibility of having event builder input on a ROC (i.e. with ROC data collection), respectively on a ROB-basis) with respect to:
    a. Complexity (of the implementation, of its operation, etc.)
    b. Performance
    c. Cost
- Identification of key parameters to be used as input to modelling studies to address in particular scalability.

5.3.2 Event filter

5.3.2.1 Introduction

The planned work for the event filter subsystem in the 18 months up to the Technical Proposal will be split into three main areas, namely: event filter software and physics strategy, prototype construction and development (including system modelling), and additional EF functionality. Two prototype architectures are currently under investigation, and others will be considered during the coming year. The integration, testing and validation of these prototypes into the DAQ-1 system will also form an important part of the work plan.

5.3.2.2 Event filter software and physics strategy

The EF software comprises all the reconstruction and physics analysis software which will be run in the EF context. Unlike the LVL1 and LVL2 triggers, specific dedicated algorithms will NOT be written for the EF. The goal is to use exactly the same algorithms as used in the offline software. The EF may, however, use a reduced set of algorithms, and employ somewhat wider cuts and coarser grained parameters than those used in the offline. This approach has the advantage of facilitating studies of possible physics biases, detailed EF-offline comparisons, and minimizing duplication of effort.

A very extensive development is currently under way in the ATLAS offline group to port all reconstruction and analysis software to an OO platform. However, this is not expected to be stable and available to users in the immediate future. In order to progress in the understanding of software to be run in the EF, we have therefore decided to use the currently available FORTRAN software for most of our forthcoming studies. In addition to work directly for the EF, it is hoped that the results of these studies will serve as feedback to the offline group for their OO development. We shall begin to work with the new offline OO suite as soon as is practically possible. An outline of our plans is given below.

A large fraction of the offline reconstruction and analysis software will be used to address the following issues:

- The software architecture of event processing. In order to have the possibility of flexible processing in the EF, the algorithms used must be highly modular.

- Development of representative physics benchmarks to study relative processing times. This should also allow us to gain some understanding of the number of subfarms and total CPU power which will be required in the final EF system.
- The variation of input parameters such as tracking step size in order to investigate issues of reconstruction quality versus execution time.
- Running the offline algorithms on specific physics channels will also allow us to parameterize their performance in terms of execution time, database access and I/O time as a function of these physics channels. We shall then use these parameterizations, to develop an event filter 'Emulator' program (EFE). The EFE will be run on the various EF prototypes to emulate the behaviour of the processing task for given physics channels. This approach has the advantage, in particular in the short term, of avoiding the porting of the current FORTRAN software and its associated ZEBRA data structures to the online EF prototype environment, whilst providing a reasonable approximation of the processing resources required for the event filtering process.

The overall event processing and physics strategy in the EF is also a rather complex issue. Although we must be able to perform complete event reconstruction, it may not always be necessary, depending on the particular physics channel in question. Physics cuts used in the algorithms must be developed specifically for EF use. The ability to 'seed' processing, based on the LVL2 decision is also crucial, and must be a requirement on the future offline code if it is to be usable in the EF. One of the first tasks of the EF will be the confirmation of the LVL2 decision. Complementary studies of the LVL2 and equivalent EF algorithms which perform this confirmation will be performed. The design and development of the EF processing and physics selection strategy will form a large part of the work in this branch of the project.

This work was begun very recently, in close cooperation with the physics and offline groups. The overall planning will clearly be modified to take into account the progress in the offline OO development.

5.3.2.3 Prototype construction and development

Two prototype EF subfarms are currently being developed. The primary aim is to study the various system aspects of different candidate architectures and to study their performance, so as to provide input into the final design decisions. They will also be used as software test beds to develop event processing strategies in the online environment. The following global studies are well under way:

- Develop a generic event distribution scheme within the EF to be used on the various prototypes.
- Develop a control and monitoring scheme for the prototypes.

In addition, technology watches are being set up on the developments of all other major manufacturers. It is planned to construct further prototypes based on joint projects with one or more of these manufacturers in the forthcoming 18 months.

The various stages in a prototype project are listed below:

- Assemble a stand-alone prototype, and run physics emulator software (as explained above) in order to measure performance, and study processing and farm organization. Study system management, error handling and control issues.

- Enlarge the prototype to study scalability aspects of the design. Use measurements made on the prototype as input to a computer modelling process, whose results will complement those of the prototype.
- Integrate the prototype into the DAQ-1 Dataflow and Back-End systems.

The first prototype, based on PCs, is currently running on a farm of three machines, and will be enlarged to 25 machines in the second quarter of 1998 in collaboration with IT/PDP.

Recently, a Project Assignment Plan to study event filter Farms was accepted by the LCB. ATLAS will actively participate in this project. The main objectives, summarized here for completeness, are clearly very coherent with many of the points of the work plan outlined above:

- Identify detailed issues created by EF stages of the LHC experiments, and identify relevant activities already taking place in IT Division
- Address issues identified, and study possible solutions
- Share with other HEP experiments, experience and plans to use processing farms for data filtering
- Demonstrate prototype processor farms in LHC test beam operations, including access to mass storage. Demonstrate scalability

5.3.2.4 Additional functionality

In addition to its primary functionality of event filtering, the EF will also be required to provide monitoring, calibration, and possible alignment facilities for the experiment. Various types of monitoring requirements are envisaged, namely: physics quality, trigger efficiency studies, detector performance control. The identification of the requirements of monitoring, calibration and alignment, and the strategies to implement them in the EF will be the subject of detailed and continuing discussions within the Detector Interface Group in the forthcoming 18 months.

5.3.2.5 EF / dataflow integration

When the integration of an Event handler (EH) prototype has been completed, we shall have a component which is capable of:

- Receiving physics events from a static source (file) and distributing them to processing tasks, according to a pre-defined event type, via a Distributor
- Processing events (reconstruction and physics analysis) with a Processing Task
- Collating and outputting results to a static sink (file) via a Collector

This EH will be locally controlled by its Supervisor.

The subsequent step in the integration plan is to replace the data I/O from/to static files with I/O from/to the subfarm Input / Sub_farm Output (SFI/SFO) elements of the DataFlow, by implementing an Application Programmers' Interface (API). The Distributor will implement the interface with the SFI, and the Collector with the SFO. Via this API, the SFI will send complete events to the Distributor, and the SFO will receive selected events from the Collector. In this way, the EH is viewed as an event processing 'black box' by the dataflow elements.

In the current prototype developments, there will be one SFI-Distributor and one Collector-SFO pair per prototype.The functionality of the interface will be identical although the physical implementation will clearly differ.

5.3.2.6 EF / back-end DAQ integration

Most of the Back-End DAQ functionality required by an EH will be implemented by the EH Supervisor. Integration will therefore proceed in the following manner:

- Supervisor <-> Configuration Databases

The Supervisor is responsible for the initialization and configuration of the elements of the EH at start-up, therefore requiring the services of configuration databases

- Supervisor <-> IS and MRS

The Supervisor provides the global information and error interface between the EH and the outside world. It therefore needs the functionality of the IS and MRS

- Supervisor <-> PM and RC

The Supervisor is responsible for the process control of all the EH elements and implements the interface with the experiment run control system

In addition to the above, individual EH elements will require the ability to access general status information and report individual error states. They will therefore all require direct IS and MRS functionality.

5.3.3 Back-end DAQ

5.3.3.1 Introduction

This section gives an overview of how the Back-end DAQ subsystem will be used from 1998 until the Technical Proposal document at the end of 1999. The aim of this work is to validate that the architecture and functionality of the Back-end software are suitable for the final ATLAS system while using the DAQ/EF Prototype -1 project as a vehicle for such studies. It is necessary to determine if the architecture delivers the necessary qualities of scalability, availability (i.e. minimal down-time), performance and maintainability. To measure scalability and performance, test-plans are being established for each component. The tests will be made in the DAQ-laboratory rather than at the test beam. Initial estimates of availability and maintainability can be made in the laboratory but these must be confirmed by use in test-beam conditions with prototype detectors.

Certainly problems will be encountered during integration with the other subsystems but it is hoped that the effort put into gathering requirements and making high-level designs will keep these to a minimum. So far, a basic plan for integration with the dataflow and then the event-filter has been outlined. Later, integration is foreseen with other online systems (DCS and triggers) as well as detectors.

This experience (laboratory and test beam) will act as input to a review of the requirements and design with the intention of making updated versions of the documents that can be reused for

the production of the final system. It will also aid in judging the importance of each component and may identify new ones which were not originally included.

In parallel, it is important to continue the work on finding the most effective means of producing software. To date we have applied various software engineering techniques (Section 5.2.4.6) and developed a basic software development environment (Section 5.2.4.7). Continued improvements in the software process and development environment should result in measurable improvements in software quality.

5.3.3.2 Core component unit tests

This section gives an overview of the test plans that have been developed for unit testing of each back-end DAQ core component and details of how the integration tests are to be organized. The unit tests described below include only scalability, performance, and basic reliability tests. Functionality and resilience tests will be described in the individual component test-plan documents.

Run control

As can be seen from the component dependencies, see Figure 5-34, the run control requires most of the other core components to operate. In a sense the run control component represents a simple integration test in itself.

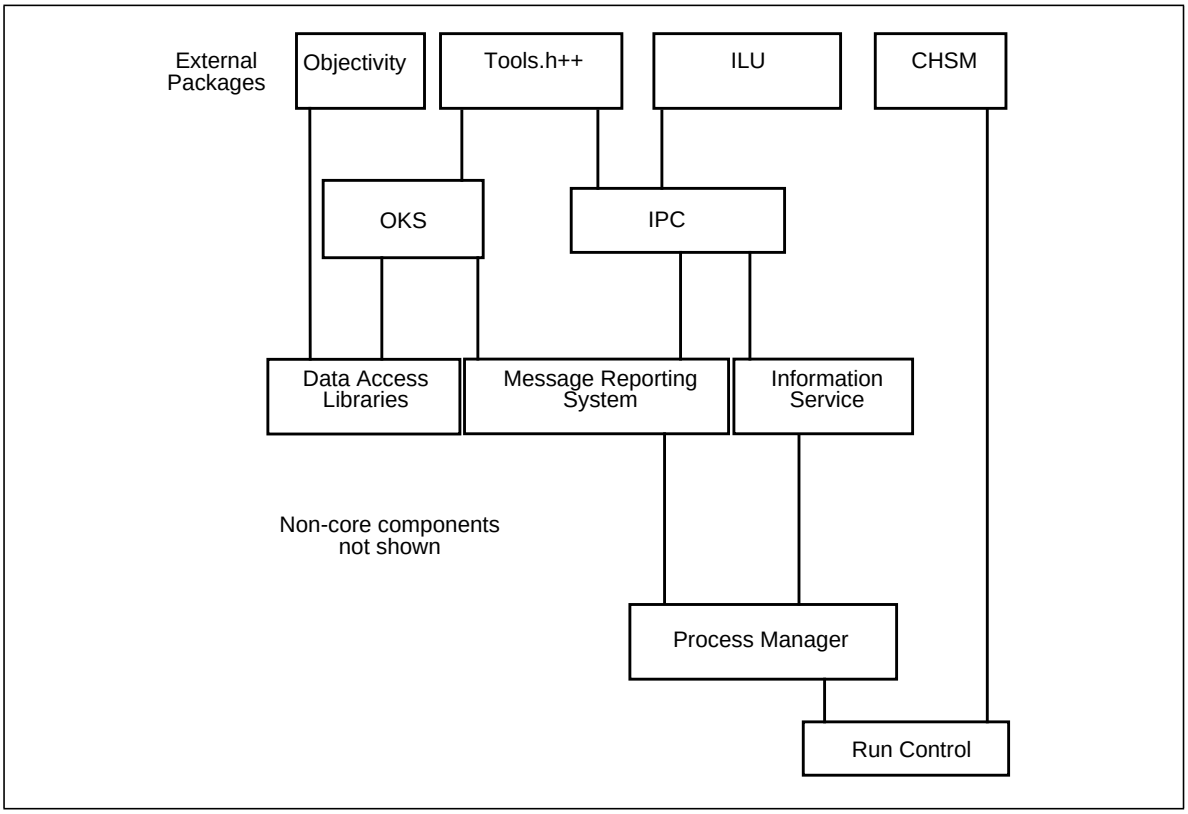


Figure 5-34 Dependencies between back-end core components and external packages.

In order to perform unit tests, a standard controller [5-56] that has empty actions and is compiled for all platforms has been implemented. This allows the run control to be tested without

the equipment to be controlled and software from other subsystems. Sample DAQ configurations (containing a hierarchy of controllers) have been defined in the configuration database.

Scalability and performance tests

The configurations described above are used to perform the following tests with varying numbers of controllers in the control hierarchy:

Cold start

Assume all processors are working and a Process Manager agent is running on each one. Measure the time it takes to start the DAQ Supervisor, all controllers, associated servers (MRS, Information Service etc.) and do all necessary transitions to get to the *Running* state. This test should correspond to the actions needed at the start of a data-taking session.

Cold stop

Measure the time taken to do the reverse of the cold start test (i.e from *Running* back to *Initial* and then stopping all processes so just the Process Manager agents are alive). This test should correspond to the actions needed at the end of a data-taking session.

Lukewarm start

Assume all necessary processes (controllers and servers) are started and in their *Initial* state. Measure time taken to get to the *Running* state. This test should correspond to the actions needed when starting with a new configuration. However, it does not take into account that it may be necessary to stop some processes which are not used in the new configuration and start others which were not present in the old configuration.

Lukewarm stop

Measure time taken to do the reverse of the lukewarm start test. This test should correspond to the actions needed when stopping a run before changing configuration.

Warm start

Assume all processes are alive and in the *Configured* state. Measure the time taken to get to the *Running* state. This test should correspond to the actions needed when starting a new run with the same configuration.

Warm stop

Measure time taken to do the reverse of the warm start test. This test should correspond to the actions needed to stop a run.

The preliminary results for those tests already completed are shown in Table 5-2. All the test described above will be made when integration with the Process Manager (for the cold start / stop tests) is finished and when sufficient hardware is available to allow individual controllers to reside on separate machines. The tests will also be repeated on a mixture of operating systems and with larger hierarchies containing 50 and 250 controllers to represent the final ATLAS system.

Recovery tests

The goal of the recovery tests is to evaluate the error handling capabilities of the run control component. From a configuration in which all processes are *alive* and all controllers in the *Running* state, the tests listed below are performed. In each case the tests are run with first a single

Table 5-2 Run control scalability results (time in milliseconds averaged over 10 cycles)

	Number of controllers in hierarchy		
	1	**5**	**10**
Operation[a]	**1 root**	**1 root, 2 detectors + 2 subdetectors**	**1 root, 9 detectors**
lukewarm start	250	500	810
lukewarm stop	220	500	820
warm start	60	180	290
warm stop	50	180	280

a. The timing values represent the total time (system and user) for sending a command to the Root controller of the hierarchy from the run control interface and waiting until it gets a confirmation of completion. All the tests were performed on the Solaris operating system with all the controllers residing on a single machine. No user-defined actions (i.e. code) were defined during the state transitions.

controller affected and then with several controllers affected. The tests should also be run when the affected controller(s) is a leaf and an intermediate node in the hierarchy:

- a controller makes an unsolicited state change,
- a controller dies,
- a controller exceeds the time limit for a transition,
- a controller leaves / joins the configuration,
- stop various elements: Information Service server; DAQ supervisor; root controller.

5.3.3.3 Configuration database

The unit tests of the configuration database presented below address two areas: the underlying technologies employed for data persistence, namely the OKS persistent object manager and Objectivity OODBMS; and a typical DAQ configuration database implementation based on OKS.

OO7 benchmark

Recently, a number of OODBMS systems have become publicly available. However, perhaps since the technology is so new, it is not yet clear how these systems differ in their performance characteristics. The OO7 Benchmark [5-76], has been designed as a first step toward providing a comprehensive OODBMS performance profile. Among the performance characteristics tested by OO7 are:

- the speed of various traversals,
- the efficiency of various updates, including updates to indexed object fields, repeated updates, sparse updates, and the creation and deletion of objects,
- the performance of the query processor on various queries.

There are three standard basic sizes of the OO7 Benchmark database: small, medium, and large. The tiny configuration has been identified by the ATLAS DAQ group to satisfy the initial re-

quirements of the Prototype -1 project. The number of interconnected objects and the size of the data file in each configuration are shown in Table 5-3.

Table 5-3 OO7 benchmark database configurations (extractions).

Configuration	Number of objects	OKS database file size (Mbytes)	Objectivity database file size (Mbytes)
tiny-3 / 6 / 9	5,195 / 8,195 / 11,195	0.6 / 0.9/ 1.1	1.2 / 1.4 / 1.6
small-3 / 6 / 9	42,095 / 72,095 / 102,095	5.6 / 8.5 / 11.5	6.4 / 8.1 / 10.5
med-3 / 6 /9	402,095 / 702,095 / 1,002,095	55.4 / 85.6 / 115.7	57.5 / 75.2 / 99.2
large-x	not applicable[a]	not applicable	not applicable

a. Large-x database is not relevant for DAQ configuration database usage. It is perhaps more useful for benchmarking the storage of physical data with offline software. It is expected that the ‘small-9’ OO7 configuration corresponds to the biggest ATLAS DAQ configuration database

The OO7 benchmark was chosen to evaluate the performance of OKS and Objectivity and determine if the OKS object model is comparable with the object models supported by commercial object databases. Successful completion of the benchmark would also indicate that OKS is a reliable implementation of a lightweight object manager.

Results of the OO7 benchmark

Figure 5-35 show extracts from the results of the OO7 benchmark.

Raw Traversal Speed traverses the assembly hierarchy. As each composite part is visited, performs a depth first search on its graph of atomic parts. Returns a count of the number of atomic parts visited when done.

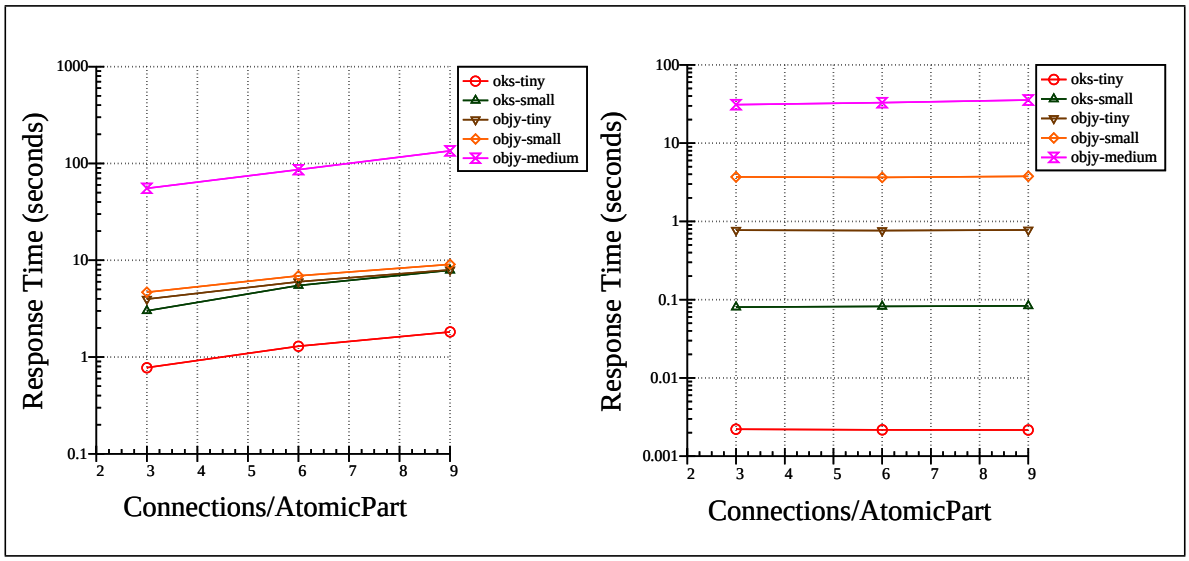


Figure 5-35 Results of raw (left) traversals and ad-hoc join queries (right)

Queries

Ad-Hoc Join Query finds all pairs of documents and atomic parts where the document ID in the atomic part matches the ID of the document. Also, returns a count of the number of such pairs encountered.

Performance benchmark of DAQ configuration databases using OKS

The OO7 benchmark described above is supplemented by further tests designed specifically for the DAQ Configuration databases. The intention of theses tests are to evaluate the performance for expected DAQ configurations:

- Single readout crate: ~200-400 database objects
- Prototype-1 DAQ configuration (10 crates): ~3000–6000 database objects
- Final ATLAS DAQ Configuration (200 crates): ~50000–100000 database objects

These tests have been performed with OKS only (and not Objectivity) because it is the run-time persistent object manager and hence directly affects the DAQ performance during data taking activities.

The DAQ configuration database schema was generated from the StP CASE tool and contains 49 classes. The tests were made on various computers including sunatdaq01 (Sun Ultra-Enterprise 3000, 2 CPU, 167 MHz, 128 MB) and lhctb19 (SPARCstation 20, 75 MHZ, 64 MB) running Solaris 2.5, rd13fedev (PowerPC on VME 8062, 200 MHz, 64 MB) and rd13fe11 (PowerPC on VME 8061, 100 MHz, 16 MB) running LynxOS 2.4, pcatd01 (IPM PC, 2 CPU, 133 MHz, 64 MB) running Windows NT 4.0 and atlas08 (HP workstation, 152 MB) running HP-UX 10.20. It was not possible to test large configurations (50000–100000 objects) on all the computers due to a lack of memory.

Database initialization and shutdown

To load the database configuration in memory an application must load schema and data files. If an application uses the database during initialization only and wants to release resources allocated by database, it must close the database. The time to load and close schema files should not depend on number of objects and requires between 50 and 300 ms (Figure 5-36).

Access database information

Figure 5-37 shows the time to access information inside the database. The diagram on the left presents the time to find an object by it’s identity and the right picture shows the time to access the value of an object’s attribute by its name.

From the above figures it is shown that the time to load a configuration even for slowest workstations is a few seconds and less than one second for the fastest. The memory requirements are quite moderate and an application with an average number of objects loaded uses 3–5 MBytes. The size of such an application without the OKS database is about 2 MBytes including application code, C++ run-time and system libraries.

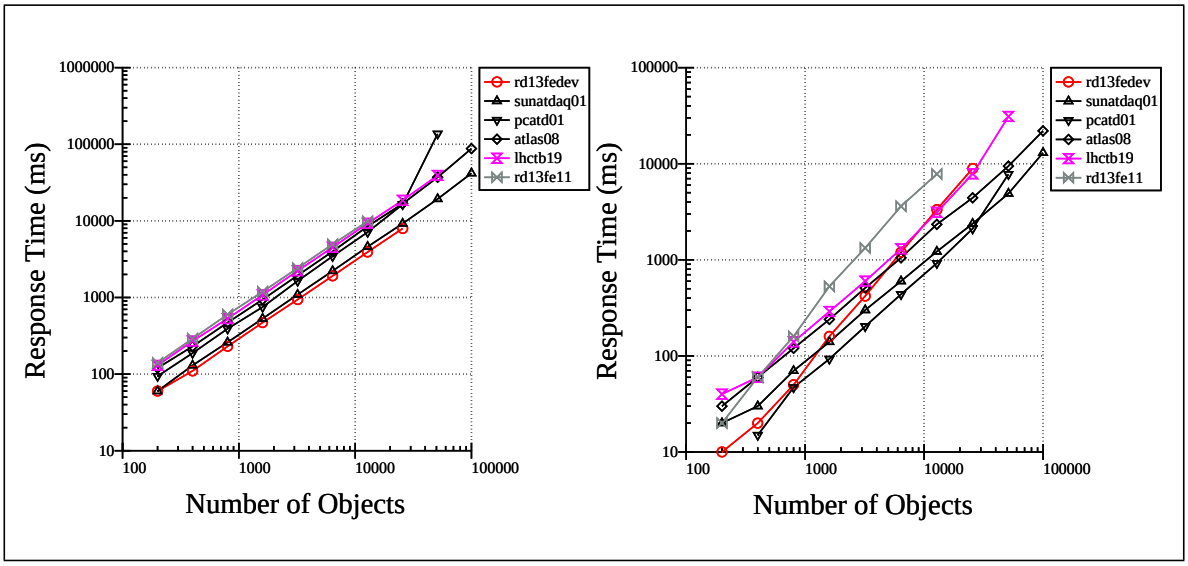


Figure 5-36 Time to load database data file (left) and close (right).

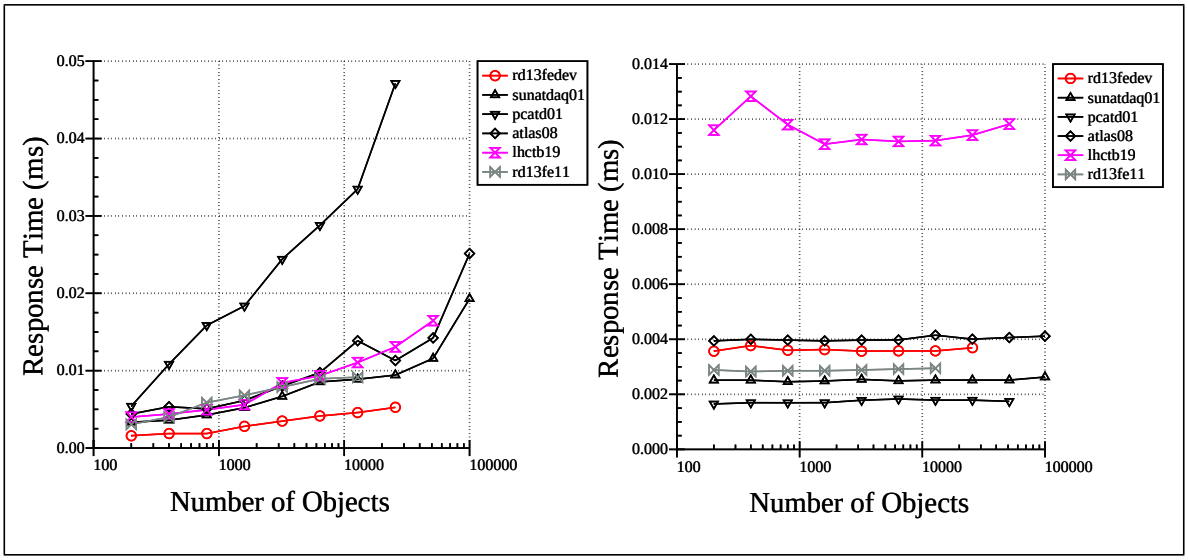


Figure 5-37 Time to access an object by identity (left) and value of attribute by name (right).

5.3.3.4 Message reporting system and information service

IS and MRS have related objectives and a similar internal structure and hence the tests for these two back-end components are described together.

Test goals

Scalability and performance tests for MRS provide information about the transport time of an MRS message from a sender to a receiver under a chosen set of test conditions. For IS, the tests provide information about the elapsed time from the initial publication, or a later update, of information to its notification to the user. A selection of boundary tests and studies of situations which can lead to saturation is included.

The tests show dependencies on the type of operating system, the configuration and the database size.

Test configuration

In defining the test configurations, both the ATLAS DAQ/EF prototype-1 and the final ATLAS DAQ systems are considered. The main users of the MRS and IS system are the LDAQ units, the run controller units and the other back-end components. The maximum numbers of sources for IS, senders for MRS and receivers are presented in Table 5-4.

Table 5-4 Maximum number of components for ATLAS DAQ/EF Prototype -1 and final ATLAS DAQ.

	Prototype-1		Final ATLAS	
Configuration	senders/ sources	receivers	senders/ sources	receivers
MRS	50	5	350	10
IS	50	10	350	25

Description of tests and results

- The operating systems involved in the tests are Solaris, HP-UX, LynxOS and Windows NT.
- The tests are performed with one or more active senders/sources and one or more active receivers.
- The MRS message is registered in the MRS message database.
- All tests use the same partition and therefore a single server.
- The transported message is typical for a MRS message. It is composed of the message name, severity and message text as defined in the database, one parameter, one optional parameter, the automatically appended qualifiers and one optional qualifier.
- As a typical type of information for IS, the general run control state information is used.
- The test results are registered when the host processors are at an average load.

A test plan has been established for both MRS and IS. Each test-plan includes a description of a test suite for functionality and error recovery tests as well as performance and scalability tests and the documentation of the tests programs developed.

A significant number of performance measurements on single platform configurations and on various combinations of distributed configurations have been made. Some preliminary results are presented in Tables 5-5 and 5-6.

The results shown in Table 5-5 have been obtained from MRS tests performed on a distributed configuration. The MRS server was running on a Solaris machine, and 1, 5 or 10 MRS receivers were running on another Solaris machine. The senders were equally distributed on 5 LynxOS processors. The results indicate the mean transfer time of one MRS message from one sender to one receiver when *n* messages were sent and reached *m* receivers.

The results shown in Table 5-6 have been obtained from IS tests performed on a distributed configuration. The IS server was running on a Solaris machine, and 1, 5 or 10 IS receivers were running on another Solaris machine. The information sources were distributed on two Solaris and three HPUX machines. The results indicate the mean information update cycle time (the time

Table 5-5 Results for MRS tests (in milliseconds)

	Number of senders		
Number of receivers	**1 sender**	**10 senders**	**50 senders**
1 receiver	4	15	80
5 receivers	8	65	400
10 receivers	15	100	450

Table 5-6 Results for IS tests (in milliseconds)

	Number of senders		
Number of receivers	**1 sender**	**10 senders**	**50 senders**
1 receiver	8	38	250
5 receivers	10	70	420
10 receivers	13	170	570

between the update of the information by a source and notification of all the subscribing receivers). The total number of information objects managed by the server during the test was 2000.

Work will continue to cover a wider spectrum of configurations to test scalability for the final Atlas DAQ system and to provide information about eventual limits and boundaries.

5.3.3.5 Process manager

Scalability and performance tests

To test the scalability and performance of the process manager, the time to create and to destroy a process in various circumstances will be measured. The process creation and destruction time will have an effect on the DAQ system availability when changing configuration or starting a new data-taking session. The time measured will represent the delay between a process manager client making the series of requests and receiving confirmation that the operations have been completed. The measurements will be made for increasing numbers of processes to represent various configuration sizes and on all the relevant platforms in both synchronous and asynchronous modes. Initial measurements made on Solaris platforms show that the delay in creating a process using the process manager agent increases with the number of processes managed. The delay ranges from 70 ms for the first processes and increases by 2.3 ms per managed process (see Table 5-7). This delay is explained by the way the current PMG DynamicDB is managed and could be improved.

These figures represent the most pessimistic situation: tests were made with all PMG processes and the test client running on the same Solaris machine making requests in synchronous mode. Using asynchronous creation the delay measured is approximately 20% less. The distribution of the client and processes on two machines does not significantly modify the performance thus implying the time is not dominated by communication delays.

PMG agents are not foreseen to manage more than about 50 processes per machine and most clients will manage less than 5 processes with the PMG. An important exception is the

DAQ_Supervisor which will boot the system and so start about 300 processes via many agents in a full ATLAS configuration.

Table 5-7 Preliminary results for process manager performance tests (in milliseconds on Solaris)

	Number of managed processes			
Operation	**1**	**10**	**100**	**300**
Process creation	70	90	300	750
Process destruction	56	56	56	56

Reliability tests

Process manager unit tests verify that each of the three elements (i.e. client, agent, and dynamic database) satisfy the requirements. A dedicated test program, implemented as a client, provides a way to send commands interactively, to activate all the remote methods in the agent's application programming interface. The tests involve creating a single process, in both synchronous and asynchronous modes, then executing all possible commands while verifying the following points:

- the created process is running,
- the process is registered in the dynamic database,
- the process can be killed by its creator,
- the agent is able to detect the death of the process and notify the client,
- the client is able to activate its call-back method when a created process is dead,
- the dynamic database is updated after the process death.

These tests are made twice: first by specifying all the process starting parameters and then by using the software configuration database to retrieve the process starting parameters.

A second series of tests is made with several autonomous test client programs running concurrently and making a collection of sequential process creations and destructions. After each creation and destruction, the test programs verify the dynamic database is coherent. These tests are made initially on a single platform and then with the clients and agents running on different platforms.

5.3.3.6 Trigger/DAQ and detector integration components

The core components of the back-end DAQ have been given priority in order to have a baseline subsystem that can be used for initial tests. Work has started on the development of some of the trigger / DAQ and detector integration components and is being organized according to the same software process (Section 5.2.4.6) used for the core components.

Partition and resource manager

A detailed set of requirements and high-level design have been prepared and reviewed for resource management. Implementation has started and it is expected that the unit testing will be completed by the end of 1998. Issues related to partition management (of which resource management is seen as a subset) are being defined in a separate document [5-9].

Status display

Predesign investigations are continuing into the manner of developing graphical user interfaces with Java and accessing information from back-end components (written in C++) from Java clients using ILU communication. The final design and implementation of the status display depends on the results of these investigations and data from other subsystems being available in the Configuration database, IS and MRS. It is expected that a first version of the status display will be tested during the integration with the dataflow subsystem.

Run bookkeeper

A detailed set of requirements and high-level design have been prepared and reviewed for gathering and storing DAQ information in the run bookkeeper. Implementation has started and it is expected that the unit testing will be completed by the end of 1998.

Event dump

The user requirements for the event dump have been defined [5-55]. Its development depends on the integration of the back-end core components, the completion of the monitoring task skeleton (page 158) and, as yet undefined, manpower resources for its implementation.

Test manager

A detailed set of requirements and high-level design have been prepared and reviewed for test management. Implementation will start during the summer of 1998 and it is expected that the unit testing will be completed by the end of 1998.

Diagnostics package

The user requirements for the diagnostics package have been defined [5-55]. Its development depends on the integration of the back-end core components, the completion of the test manager, and as yet undefined manpower resources for its implementation.

5.3.3.7 Integration tests

The integration tests described in this section address the issue of integrating the back-end components to form a full subsystem. It does not address the integration of the full DAQ (i.e. all subsystems including the back-end). For the purpose of the back-end integration tests, software from other subsystems will be emulated. For example, a simple LDAQ emulator is foreseen that is capable of communicating with the run control and producing IS and MRS messages but does not include any dataflow software.

The back-end component integration tests are based on an estimate of how many clients for each component will exist in the full online system. Below is an estimation of how the back-end components will be used in the other DAQ prototype -1 subsystems.

- Data-flow
    - for each Readout / Event Builder Partition / subfarm:
        - one LDAQ: DB reader, IS source, MRS sender, RC client
        - one RC controller: DB reader, IS source, MRS sender, RC server
        - one PMG Agent: DB reader, IS source, MRS sender, PMG server
    - one detector controller per detector: DB reader, IS source, IS receiver, MRS sender, RC server (estimate 1 detector controller per 10 creates)
    - one monitoring task: DB reader, IS source, MRS sender

- Event filter
    - for each subfarm
        - one subfarm LDAQ: DB reader, IS source, MRS sender
        - one PMG Agent: DB reader, IS source, MRS sender, PMG server
        - one Event Handler Supervisor: DB reader, IS source, MRS sender. RC server
- Back-end
    - IS: one IS server
    - MRS: one MRS server
    - one Resource Manager: DB reader, IS source, MRS sender
    - one Run Bookkeeper: DB reader, IS receiver, MRS sender / receiver
    - one Log file: IS receiver, MRS receiver
    - one Operator GUI: DB reader, IS receiver, MRS receiver, RC sender
    - one PMG Agent: DB reader, IS source, MRS sender, PMG server
    - one DAQ Supervisor: DB reader, PMG client, MRS sender

The estimates above are used to construct a sample configuration stored in the configuration database. The database is then used to drive tests involving all the core components. Starting with booted but idle machines, the tests simulate the start of data-taking activities by creating all the necessary processes in the defined order and then cycling the system through the states prescribed by the run control several times to simulate a series of data-taking runs. At the end of these cycles, the system is shutdown in an orderly manner and all the DAQ-related processes destroyed. The tests will initially be performed on a hardware setup including a single workstation and LDAQ processor. The hardware set-up can later be expanded to a collection of workstations running a variety of operating systems and LDAQ processors.

5.3.3.8 Further development

The unit and integration tests described above are intended to provide a first version of the back-end system for use within the DAQ / EF Prototype -1 project. It is expected that integration with other subsystems will lead to the identification of possible improvements which, coupled to evolving underlying technologies, will form the basis of further developments of the back-end components. To date, such further developments to be made after integration with other subsystems include:

Evaluate alternative CORBA implementations
CORBA is used in all the subsystems of the DAQ / EF prototype -1 project: back-end (communication for all components), event-filter (event movement and control) and dataflow (event monitoring). To date we have used ILU [5-61] for all the above applications. This choice was based on an initial set of requirements [5-55] and as the result of various evaluations [5-50]. The CORBA standard has progressed to become more popular and as a result there are more implementations available. We intend to investigate several other implementations and compare them to ILU according to a set of predefined criteria.

Investigate the use of OMG version 2 services

The OMG group, in a second version of its CORBA standard, has defined layered services, such as the Event Service, which offer facilities similar to those provided by the IS and MRS components. We intend to investigate such services and determine if they can be of use within the back-end DAQ.

Investigate Java/CORBA interfaces

To date we have made evaluations of the Java language for the development of graphical user interfaces. By using CORBA, we can evaluate implementing the client-side of components entirely in Java. This would allow us to provide Web / Java based interfaces to the DAQ which is itself implemented in C / C++.

Further integration of the Objectivity database and OKS

Currently we have a means of moving data and schemes between OKS and Objectivity. The implementation of this facility could be extended by using data access libraries independent of the underlying data manager (i.e. OKS or Objectivity). We intend to investigate the development of such data access libraries and their possible generation from the StP CASE tool.

Decision making inside the run control

The first version of the run control component that is currently undergoing unit testing, has a supervisor which uses a simple finite-state machine for starting and stopping the DAQ. Later, we would like to extend the DAQ supervisor's capacity for decision making and believe expert systems to be a good candidate technology for implementing the logic of such an intelligent supervisor. Similar requirements exist for the diagnostics package and it is hoped a single solution can be found.

5.4 The test beam

As part of the validation process of the DAQ / EF architecture, an implementation of the DAQ / EF prototype will be used at the ATLAS test beam. The integration of the system with detectors, data recording and offline analysis will be achieved in a realistic and operational environment which the ATLAS test beam represents. The robustness, reliability and maintainability of the system will be demonstrated with a real application and real users. A similar strategy had already been followed for the validation and integration of the DAQ system developed by the RD13 project [5-4], which in many aspects can be considered the predecessor of the DAQ / EF prototype '-1'. The RD13 DAQ system was first used in the ATLAS test-beam in 1992 with one subdetector as a proof of principle. After that, the system was used for the other ATLAS subdetectors one after the other and has now become the standard ATLAS test beam DAQ system. It has been in stable production phase for the last three years. The experience obtained in the development and support of the test-beam DAQ system has been a starting point for the detector interface and for the requirements and design of the dataflow and back-end subsystems.

The present RD13-based test-beam DAQ system [5-77] which is schematically shown in Figure 5-38 emphasizes a highly modular design, such that new hardware and software developments can be conveniently introduced. The system uses a VME-based readout and supports also CAMAC and HIPPI. It uses a real-time version of a UNIX operating system and runs with RISC-based front-end processors. Several sub-detectors can be taking data concurrently and completely independently one from the other. For combined data-taking of several subdetectors, an event building system is provided based on the VIC cable bus. The configuration of the hardware, the software, run and detector parameters is kept in databases. The run control is

based on finite-state machines and a commercial communication package. Graphical user interfaces were designed for the run control and the database editing. The DAQ software was developed making use of software engineering techniques and a number of commercial software products. Central data recording is being used by all subdetectors and data are recorded directly at the CERN computer centre.

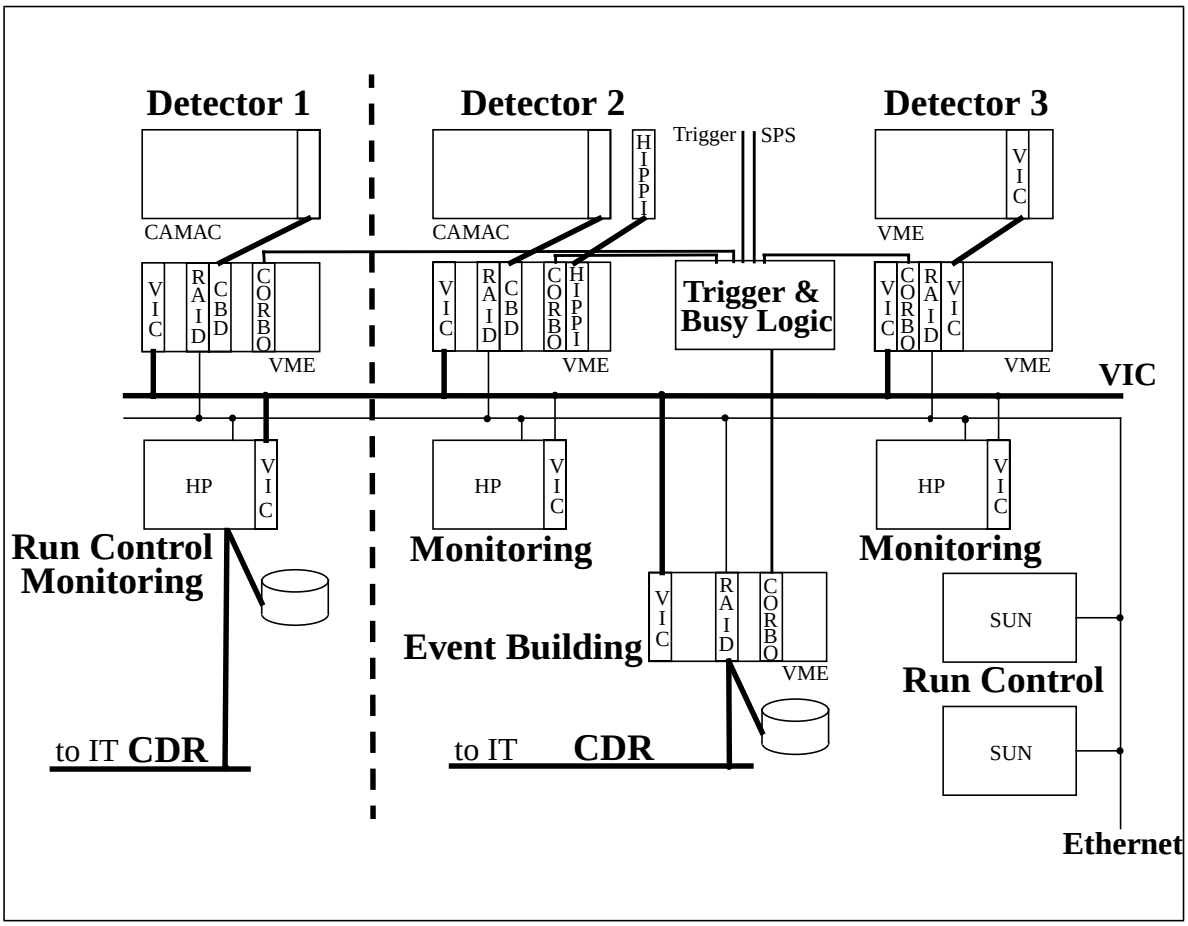


Figure 5-38 ATLAS test-beam DAQ system.

The test-beam DAQ system is being used for the simultaneous testing of ATLAS subdetectors, trigger and readout electronics. On account of the calibration programme of the subdetectors the data-taking period has extended beyond the SPS running time to essentially the whole year. The DAQ system is running stable and has not encountered major dead times. A total of 300 GByte worth of particle and calibration data have been collected.

The validation and integration of the ATLAS DAQ/EF architecture in the test beam will follow a similar strategy. The extent and modes of the test-beam use have not been defined yet and require careful planning with the subdetectors. An outline of the subdetector integration is given in Chapter 7.

5.5 References

5-1 G. Ambrosini et al., The ATLAS DAQ and Event Filter Prototype “-1” project, Presented at CHEP97, Berlin.
http://atddoc.cern.ch/Atlas/Conferences/CHEP/ID388/ID388.ps.

5-2 The ATLAS Collaboration, Technical Proposal for a general purpose experiment at the Large Hadron Collider at CERN. CERN/LHCC/94-43.

5-3 L. Mapelli, Global Architecture for the ATLAS DAQ and Trigger: Part I – Functional Model, ATLAS Internal Note, DAQ-NO-022 (1995).

5-4 L. Mapelli et al., A Scalable Data Taking System at a Test Beam for LHC, CERN/DRDC/90-64/P16, CERN/DRDC/90-64/P16 Add.1, CERN/DRDC/90-64/P16 Add.2, CERN/DRDC91-23, CERN/DRDC 92-1, CERN/DRDC 93-25, CERN/LHCC 95-47.
http://rd13doc.cern.ch/

5-5 Detector Interface Working Group – Summary Document.
http://atddoc.cern.ch/Atlas/DetfeIf/Detinfo.ps

5-6 DAQ/EF Questionnaire for Detector and Physics Groups.
http://atddoc.cern.ch/Atlas/Notes/076/Note076-1.html

5-7 Trigger & DAQ Interfaces with Front-End Systems: Requirement Document.
http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/FRONTEND/fereq.ps

5-8 Event Monitoring in the ATLAS Prototype “DAQ -1 “.
http://atddoc.cern.ch/Atlas/Notes/058/Note058-1.html

5-9 Partitioning issues in DAQ/EF prototype -1.
http://atddoc.cern.ch/Atlas/Notes/060/Note060-1.html

5-10 The event format in the ATLAS DAQ/EF prototype -1.
http://atddoc.cern.ch/Atlas/Notes/050/Note050-1.html

5-11 Test Manager design for the ATLAS DAQ Prototype -1.
http://atddoc.cern.ch/Atlas/Notes/066/Note066-1.html

5-12 G. Ambrosini et al., The Dataflow for the ATLAS DAQ/EF Prototype -1, ATLAS Internal Note, DAQ-NO-091.
http://atddoc.cern.ch/Atlas/Notes/069/Note069-1.html.

5-13 D. Francis, Definitions, acronyms and abbreviations in DAQ prototype -1. DAQ/EF Prototype -1 Project Technical Note 46.
http://atddoc.cern.ch/Atlas/Notes/046/Note046-1.html.

5-14 G. Ambrosini et al., Front-End DAQ discussion group, Summary document and work plan. ATLAS Internal Note, DAQ-NO-100.
http://atddoc.cern.ch/Atlas/FrontEnd/document/draft.ps

5-15 G. Ambrosini et al., Operating system studies for DAQ applications, ATLAS Internal Note, DAQ-NO-096, presented at CHEP97, Berlin.

5-16 A. Bogaerts et al., Event Builder Working Group. Summary document and workplan, August 1996.
http://atddoc.cern.ch/Atlas/EventBuild/document/summary.ps.

5-17 J. Petersen et al., The Read-Out Crate in the ATLAS DAQ/EF prototype -1. ATLAS Internal Note, DAQ-NO-097, presented at Syscomms’98.

5-18 G. Ambrosini et al., The generic I/O module in ATLAS DAQ prototype -1, ATLAS Internal Note, DAQ-NO-093, 22 May 1998.
http://atddoc.cern.ch/Atlas/Notes/041/Note041-1.html.

5-19 G. Ambrosini et al., The Main dataflow in the read-out crate of ATLAS DAQ prototype-1, ATLAS Internal Note, DAQ-NO-091, 22 May 1998.
http://atddoc.cern.ch/Atlas/Notes/047/Note047-1.html.

5-20 G. Ambrosini et al., A logical model for Event Building in DAQ -1, ATLAS Internal Note, DAQ-NO-112.
http://atddoc.cern.ch/Atlas/Notes/042/Note042-1.html.

5-21 G. Lehmann et al., A high level design for the Event Building Source and Destination in the ATLAS DAQ Prototype-1, ATLAS DAQ/EF Prototype -1 Project Technical Note 72.
http://atddoc.cern.ch/Atlas/Notes/072/Note072-1.html.

5-22 G. Ambrosini et al., The LDAQ in the ATLAS DAQ Prototype -1, ATLAS Internal Note, DAQ-NO-094.
http://atddoc.cern.ch/Atlas/Notes/040/Note040-1.html.

5-23 G. Crone et al., Intelligent I/O processors in the DAQ unit of the ATLAS DAQ/EF prototype -1. ATLAS DAQ/EF Prototype -1 Project Technical Note 65.
http://atddoc.cern.ch/Atlas/Notes/065/Note065-1.html

5-24 R. McLaren and O. Boyle, ATLAS Read-Out Link Data Format, Version 1.1.
http://www.cern.ch/HSI/atlas/format/

5-25 G. Ambrosini et al., The sub-farm DAQ of the ATLAS DAQ/EF prototype -1, ATLAS Internal Note, DAQ-NO-092.
http://atddoc.cern.ch/Atlas/Notes/044/Note044-1.html

5-26 B. Rensch, ROC Simulation with Ptolemy, DAQ/EF Prototype -1 Technical Note 90.
http://www.nbi.dk/~rensch.

5-27 The Almagest – Volume I: Ptolemy 0.7 User's Manual.
http://ptolemy.eecs.berkeley.edu/papers/almagest/user.html

5-28 G. Ambrosini et al., The Data Flow Manager, ATLAS DAQ/EF Technical Note 73.
http://atddoc.cern.ch/Atlas/Notes/073/Note073-1.html

5-29 F. Hogbe-Nlend et al., The Event Builder Network I/O Library, ATLAS DAQ/EF Prototype -1 Project Technical Note 67.
http://atddoc.cern.ch/Atlas/Notes/067/Note067-1.html

5-30 G. Ambrosini et al., LocalDAQ control prototype, ATLAS DAQ/EF Prototype -1 Project Technical Note 15.
http://atddoc.cern.ch/Atlas/Notes/015/Note015-1.html

5-31 C. Ottavi, The event monitoring in the LocalDAQ, ATLAS DAQ/EF Prototype -1 Project Technical Note 43.
http://atddoc.cern.ch/Atlas/Notes/043/Note043-1.html

5-32 G. Ambrosini et al., LDAQ communications system, ATLAS DAQ/EF Prototype -1 Project Technical Note 23.
http://atddoc.cern.ch/Atlas/Notes/023/Note023-1.html

5-33 G. Ambrosini et al., The LDAQ - DAQ unit/event builder interface API in the ATLAS DAQ/EF prototype -1, ATLAS DAQ/EF Prototype -1 Project Technical Note 88.
http://atddoc.cern.ch/Atlas/Notes/088/Note088-1.html

5-34 I. Soloviev et al., The OKS Persistent In-memory Object Manager, Xth IEEE Real Time 1997 Conference, Beaune, France, September 22–26.
http://atddoc.cern.ch/Atlas/Conferences/RTI97/Bob/paper_146/paper-146.html

5-35 D. Francis et al., Error classification and handling for the ATLAS DAQ/EF prototype -1. ATLAS DAQ/EF Prototype -1 Project Technical Note.
http://atddoc.cern.ch/Atlas/Notes/074/Note074-1.html

5-36 ATLAS Event Filter Project Definition Document.
http://atddoc.cern.ch/Atlas/EventFilter/URD/PDD.ps.gz

5-37 presentation to LHCC Committee (March 3rd, 1998).
http://wwwinfo.cern.ch/~eerola/lhcc.ps

5-38 E. Richter-Was et al., ATLAS Internal Note, PHYS-NO-074.

5-39 See presentations to LHCC Committee (October 30th, 1996) and published notes
http://atlasinfo.cern.ch/Atlas/GROUPS/PHYSICS/SUSY/susy.html

5-40 Trigger Performance Technical Design Report, ATLAS document to be published.

5-41 Inner Detector Technical Design Report, CERN/LHCC/97-16.

5-42 ATLAS Collaboration, Trigger Performance Status Report, CERN/LHCC 98-15, June 1998.

5-43 E. Richter-Was, ATLFAST 1.0: A package for particle-level analysis, ATLAS Internal Note, PHYS-NO-79.

5-44 D. Francis, et al., The Sub-farm DAQ for the ATLAS DAQ Prototype -1, ATLAS DAQ Prototype –1 Technical Note 44.

5-45 C.P. Bee et.al., A High Level Design of the Sub-Farm Event Handler, ATLAS DAQ Prototype –1 Technical Note 61.

5-46 C.P. Bee et.al., Event Handler Supervisor High Level Design, ATLAS DAQ Prototype –1 Technical Note 92.
http://atddoc.cern.ch/Atlas/Notes/092/Note092-1.html

5-47 C.P. Bee, et al., Proposal for an Event Filter Prototype based on PC Computers in the DAQ-1 context, ATLAS DAQ Prototype -1 Technical Note 45.

5-48 M. Michelotto, C. von Praun, A Paper Model for the ATLAS Event Filter Sub-Farm, ATLAS DAQ Prototype -1 Technical Note 55.

5-49 C. von Praun, A Framework for Modelling and Simulating Data Flows in Distributed Computing Systems, IEEE Real-Time Conference, Beaune, 1997

5-50 S. Kolos, Inter-component communication in the ATLAS DAQ Back-end software (evaluation of the ILU multi-language object interface system), ATLAS DAQ Prototype -1 Technical Note 3.

5-51 D.C. Schmidt et.al., Real-time CORBA with TAO (The ACE ORB).
http://www.cs.wustl.edu/~schmidt/TAO.html

5-52 PCI Vertical Interconnection, Creative Electronic Systems, Geneva, Switzerland.

5-53 P. Anelli et.al., Proposal for an Event Filter Prototype based on a Symmetric Multi Processor architecture, ATLAS DAQ Prototype -1 Technical Note 82.

5-54 Sandia/Intel announcement.
http://www.intel.com/pressroom/archive/releases/cn121796.htm

5-55 ATLAS DAQ Back-end Software User Requirements Document, PSS05-ATLAS-DAQ-SW-URD, ATLAS Internal Note, DAQ-NO-90. http://atddoc.cern.ch/Atlas/DaqSoft/document/draft_1.html

5-56 High-Level Design of the ATLAS DAQ Back-end software. ATLAS Internal Note, DAQ-NO-87.

5-57 Tools.h++ Introduction and Reference Manual Version 6. Thomas Keffer, Rogue Wave Software Inc, 1994.

5-58 I. Soloviev, OKS Documentation Set, ATLAS DAQ/EF Prototype -1 Project Technical Note 30. http://atddoc.cern.ch/Atlas/Notes/030/Note030-1.html

5-59 LCRB Status Report / RD45. CERN, LCRB 96-15. http://wwwinfo.cern.ch/asd/cernlib/rd45/reports/lcrb_mar96.ps

5-60 A. Patel et al., Use of a CASE tool for developing ATLAS DAQ software, presented at CHEP 97, Berlin. http://atddoc.cern.ch/Atlas/Conferences/CHEP/ID248/ID248-1.html

5-61 ILU documentation. ftp://ftp.parc.xerox.com/pub/ilu/ilu.html.

5-62 S. Kolos, Inter-Process Communication Package, ATLAS DAQ/Event Filter Prototype -1 Project Technical Note 75. http://atddoc.cern.ch/Atlas/Notes/075/Note075-1.html

5-63 P.J. Lucas, F. Riccardi, CHSM: A Language System Extending C++ for Implementing Reactive Systems. http://www.best.com/~pjl/software.html

5-64 P. Croll et al., Use of Statecharts in the Modelling of Dynamic Behaviour in the ATLAS DAQ Prototype -1, IEEE Trans. on Nucl. Science, vol. 45, No. 4, August 1988, also presented at RT97, available on CD.

5-65 X-Designer Release 4.7. User's Guide, Imperial Software Technology Limited, 1997. http://www.ist.co.uk/

5-66 D. Burckhart et al., Software Technologies for a Prototype ATLAS DAQ. Computer Physics Comms. 110 (1998) 1-7.

5-67 Watts S. Humphrey, Managing the Software Process, Addison-Wesley, ISBN 0-201-18095-2, 1990.

5-68 ATLAS Software Development Environment – User Requirements Document, 26 April 1996. ATLAS Internal Note, SOFT-NO-034. http://atlasinfo.cern.ch/Atlas/documentation/notes/SOFTWARE/note34/asde.html

5-69 S.M. Fisher, K. Bos and R. Candlin, The ATLAS Software Process, ATLAS Internal Note, SOFT-NO-02. http://atlasinfo.cern.ch/Atlas/GROUPS/SOFTWARE/NOTES/note27/asp.html

5-70 M. Niculescu, S-Link performance measurements in the environment of the ATLAS DAQ/EF prototype -1. ATLAS DAQ/EF Prototype -1 Project Technical Note 70. http://atddoc.cern.ch/Atlas/Notes/070/Note070-1.html

5-71 M. Niculescu, Event simulation with SLATE in the ATLAS DAQ/EF prototype -1. ATLAS DAQ/EF Prototype -1 Project Technical Note 79. http://atddoc.cern.ch/Atlas/Notes/079/Note079-1.html.

5-72 D. Francis et al., Hardware evaluation tasks. ATLAS DAQ/EF Prototype -1 Project Technical Note 89.
http://atddoc.cern.ch/Atlas/Notes/089/Note089-1.html

5-73 R. Spiwoks et al., The Event Builder for the ATLAS DAQ prototype -1, ATLAS Internal Note, DAQ-NO-098, presented at RT97 Conference, Beaune.
http://atddoc.cern.ch/Atlas/Conferences/RTI97/Ralph/ebtech.ps.

5-74 C. Bizeau et al., RD31 Status Report 1997, NEBULAS: High-performance Data-driven Event Building Architectures based on Asynchronous Self-routing Packet-switched Networks, CERN/LHCC 97-05; and the references therein.

5-75 G. Ambrosini et al., Experience with Fibre Channel in the environment of the ATLAS DAQ Prototype -1, ATLAS Internal Note, DAQ-NO-88, submitted to Nuclear Instrumentations and Methods.

5-76 M.J. Carey, D. J. DeWitt and J.F. Naughton, The OO7 Benchmark, Computer Sciences Department University of Wisconsin, Madison, 1994.
ftp://ftp.cs.wisc.edu/007

5-77 M. Caprini et al. et al., The Data Acquisition System of the ATLAS Combined Test-beam April 1996, ATLAS Internal Note DAQ-NO-058, November 1996.

6 Detector Control System

The task of the Detector Control System (DCS) is to enable the coherent and safe operation of the ATLAS detector. Its main requirements have been laid down [6-1] and prototype work is in progress. Early in 1998 the four LHC experiments and the CERN controls group, IT/CO, started a joint controls project with the aim of arriving at a common solution for all experiments. ATLAS participates actively in this initiative.

6.1 Scope of the DCS

The DCS interacts with three entities: the ATLAS subdetectors, the infrastructure services of the experiment, and the LHC machine.

The comprehensive supervision of all subdetectors is the most important task. It comprises the monitoring of all parameters which are relevant for the correct operation of the detectors and for the subsequent physics analysis. In particular, parameters relevant for the safety of the hardware will be monitored by the DCS. However, the ultimate safety of the detector has to be guaranteed by hardwired logic and interlocks and this responsibility stays with the subdetector groups. All shift operator interactions with the subdetectors have to be done via the DCS. Detailed interactions by experts should also use DCS tools. A mechanism for information exchange between subdetectors will be provided by the DCS.

The status of services such as cooling, ventilation, electricity distribution etc. in the cavern is also of importance to the experiment and methods to exchange information and commands between the DCS and these services have to be provided.

Connections with the LHC machine will have to be provided. Typical examples of information to be exchanged between the ATLAS and the LHC control rooms are luminosity, background and radiation data, and signals to dump the beam or inhibit injection.

6.2 Architecture

The DCS will be structured in several hierarchical levels (Figure 6-1). This is suggested by the organization of the ATLAS detector into subdetectors, detector elements, sub-elements, etc. and by topological and functional requirements.

The highest level consists of workstations and/or PCs interconnected by a LAN. These provide operator interfaces, general services like databases, data storage, etc. Functions needed ATLAS wide like alarm handlers, expert systems, and gateways to external systems also run on these servers.

The middle layer comprises local control stations (LCSs). These could be implemented either in VME modules or in PCs. They supervise a well-defined part of the experiment such as a set of chambers, a gas system, or even a whole subdetector, depending on the complexity of the task. They have to work almost autonomously and some must not be dependent on external services such as computer and electrical power distribution networks. Typical tasks in the LCS include local process supervision, using finite-state machines and closed-loop control, and handling

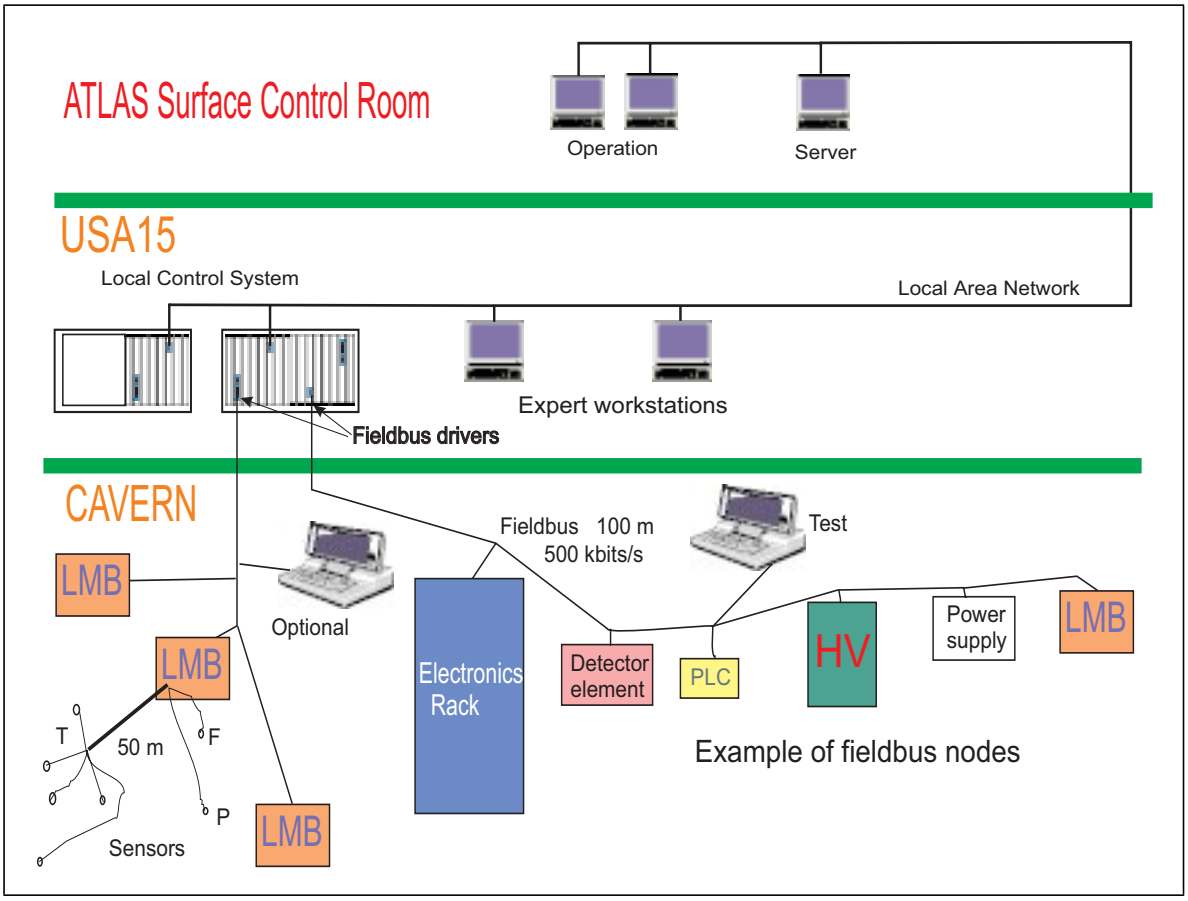


Figure 6-1 Architecture of the ATLAS DCS.

faults and taking corrective actions. The LCS also serves as a standardized interface to the subdetector equipment.

The lowest level is the front-end I/O system. It connects to the LCS via either hardware-specific or general-purpose interfaces. Where practical a commercial fieldbus will be used as this is the most appropriate method for a distributed environment. Following the CERN recommendations, ATLAS has chosen the CAN industrial fieldbus standard. It is specially suited for remote sensor readout and it also offers the possibility of local treatment of the data. We are currently prototyping this approach with the local monitor box (LMB) as discussed below. For more complex local data acquisition and control, programmable logic controllers (PLCs) will be used. There will also be a provision for connecting test and diagnostic tools for debugging and maintenance.

The hardware and software of the two higher levels will be handled ATLAS wide and thus guarantee homogeneity. The lowest level is subdetector specific, but care will be taken to use the same basic elements wherever practical.

6.3 Prototypes

In this section we describe current prototype work carried out both within ATLAS and together with the CERN controls group, IT/CO.

6.3.1 Configuration database

A general object model of the DCS representation in the configuration database has been developed where main object categories and basic abstract object classes for the system functionality and for hardware and software components have been defined. To prove the basic model and to study the description of a subdetector control in the configuration database, the control requirements of the TRT subdetector has been decomposed hierarchically and functionally. The top slice of some control subsystems (high voltage, detector ventilation) have been described in software classes and subclasses etc. [6-2]. This work has been done in conjunction with the DAQ group using the OKS data manager.

6.3.2 Supervisor system

It is our intention to use a commercial controls supervisor system if possible. To investigate the feasibility of this, we have participated in work done by IT/CO namely, the evaluation of two control systems and a general investigation of the products on the commercial market.

The first controls system evaluated was EPICS (Experimental Physics and Industrial Controls System). It is mainly used for accelerator control in the US. As a practical application, we studied the control of electronics crates via CAN. Details of the set-up can be found in Ref. [6-3]. We found that the architecture is well fitted to our needs and the client-server model suits our case. However, not all the functionality needed is available and much of the hardware foreseen in ATLAS is not supported. In addition, the system is not very easy to use by inexperienced operators as the different tools are only loosely coupled. As we have access to the source code it is possible to improve on these points, but the effort needed is substantial, both in upgrading and in maintenance, and is beyond the resources of ATLAS.

Next, the industrially supported version of EPICS, TIS-4000, was evaluated [6-4]. The set-up was very similar to that used for the EPICS tests. As the architecture and the data model are the same as EPICS, this product also fulfils the architectural requirements. Some of the features missing in EPICS are available, such as integration of the individual tools, better development facilities, and an enhanced user-interface. However, the system is much less open and, as it would be difficult to add devices, we would have to depend on the manufacturer providing them.

A general investigation of the commercial controls market is being carried out by IT/CO. Requests for information were sent to all major companies in the field. After eliminating technically unacceptable proposals, a short list of some ten products has been established and these are being inspected in more detail. An evaluation of the most promising product(s) will follow.

6.3.3 Front-end I/O via a fieldbus

A big fraction of controls applications require analogue and digital input and output channels. To deal with this and keeping the geographically distributed environment in mind, we are studying the concept of a local monitor box (LMB). This is a modular I/O system, with some local intelligence (Figure 6-2), which can be read out via CANbus. Commercial solutions exist (e.g. from WAGO) and have been evaluated, but as operation in a radiation environment and in a magnetic field is required and also for reasons of cost-effectiveness and packaging, we have built a prototype. The blocks shown in the middle line of Figure 6-2 have been implemented us-

ing a commercial CAN micro-controller and a high precision 16-bit ADC. The performance of this device will be tested on the liquid-argon calorimeter which needs a resolution of 10 mK and provides the most demanding temperature measurement in ATLAS. Radiation tests of this device are in progress. Once this concept has proven its validity it will be used by all ATLAS subdetectors and will be made available to the other LHC experiments, if they so wish.

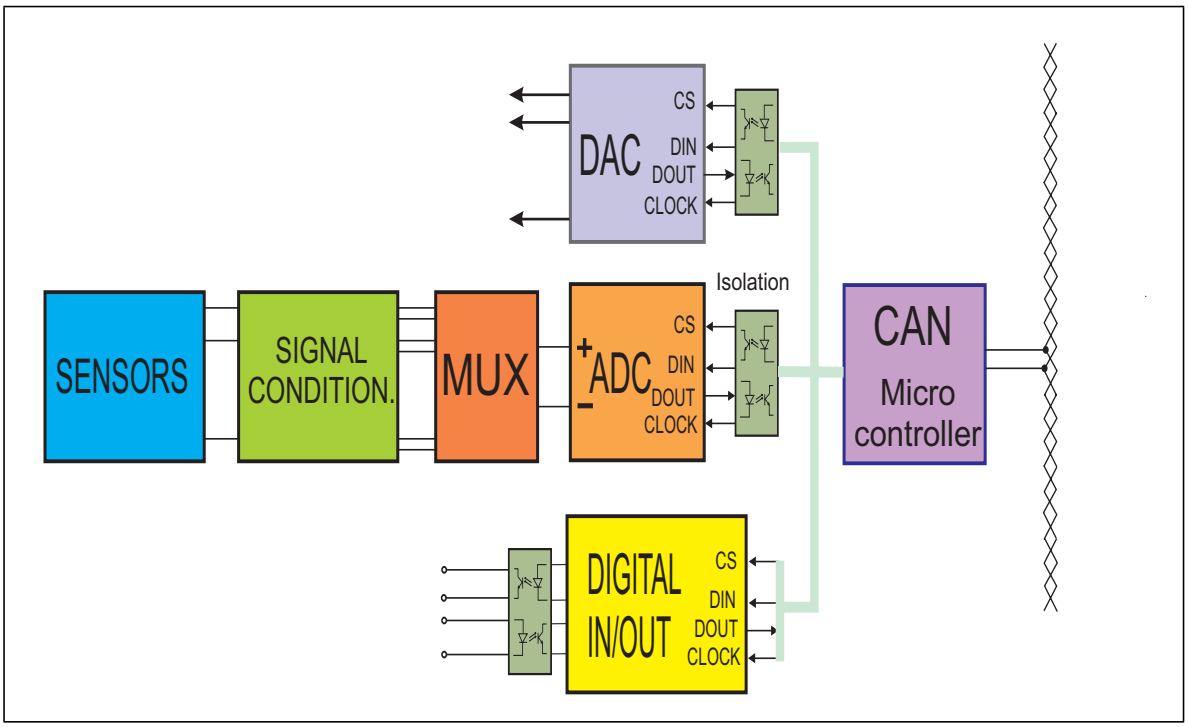


Figure 6-2 Block diagram of local monitor box.

6.3.4 Interim stand-alone controls system

ATLAS subdetector groups are carrying out tests on their prototypes which, in some cases, already contain the final controls hardware and electronics. These tests require small stand-alone control systems. They need to respect the defined interface point between their subdetector equipment and the common DCS in order to develop the final front-end now and to be able, later on, to connect it easily to the final supervisor system. The commercial package BridgeVIEW from National Instruments has been selected. As one main interface to the front-end system will be CANbus, we use the set up shown in Figure 6-3 to do the work mentioned in Section 6.3.3 and also to study the CANopen software protocol.

6.3.5 Embedded CAN systems

The muon MDT subdetector group has developed dedicated CAN nodes for chamber control [6-5]. The node comprises functions such as: generate test signals, adjust thresholds, disable noisy channels, control high voltage, monitor temperature, etc. Each chamber will be equipped with such a node and can be controlled as an independent device via the CAN fieldbus. This concept provides the flexibility to download new control algorithms.

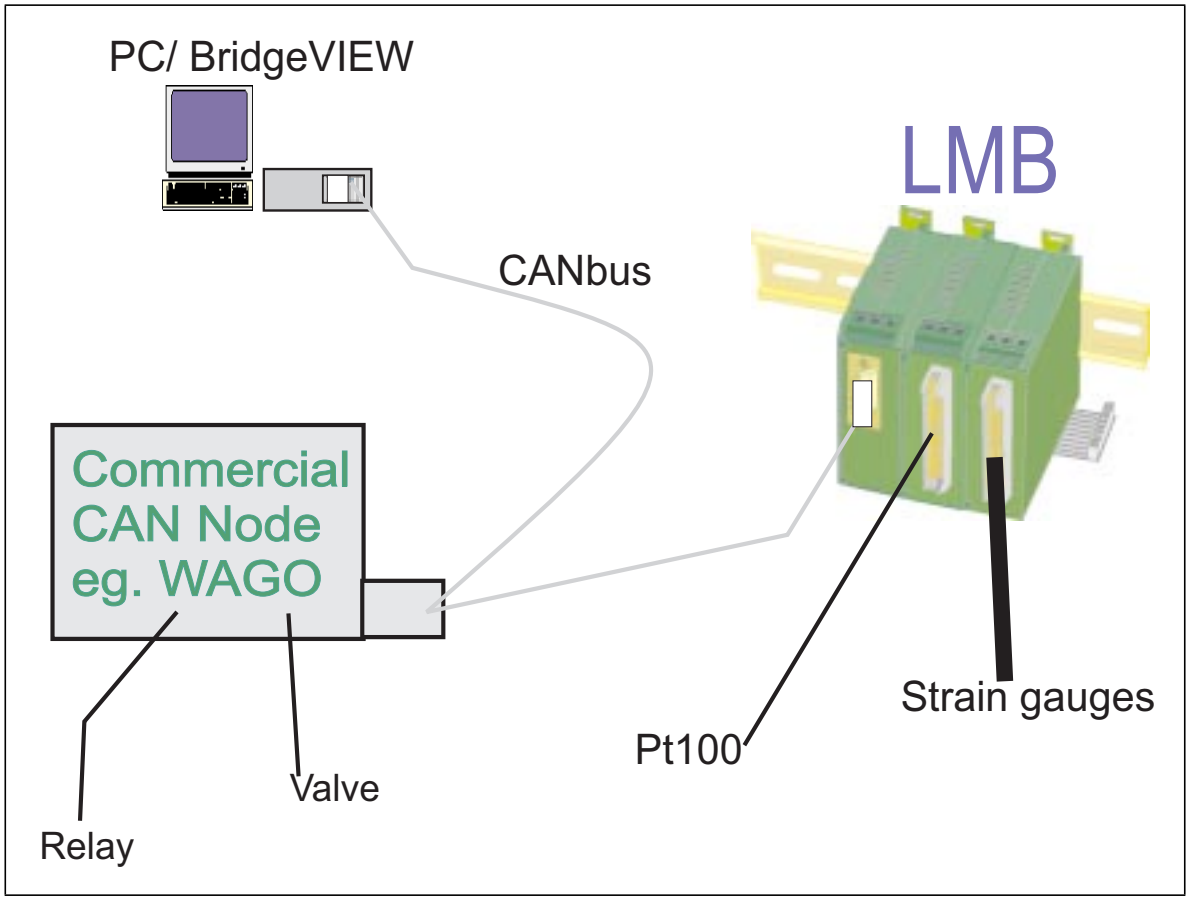


Figure 6-3 Stand-alone controls system reading CAN.

6.4 Organizational points

It is foreseen that DCS development is done as part of a common project with the other LHC experiments and IT/CO. As this common effort was started only at the beginning of 1998, a detailed project plan has not yet been established and it is not sensible to do this in isolation for ATLAS. However, dates for ATLAS are known and firm requirements are established which already impinge on the implementation of the DCS. From our side the target date for the controls system to be operational is the start of ATLAS installation at the beginning of 2003. Well before that date, however, some final control functionality is needed in the following areas:

- the acceptance test of elements at the production line (e.g. the muon chamber TGC production starts mid 1998),
- the calibration of final detector elements with beam (e.g. the TileCal group starts calibrating elements mid 2000 at the SPS),
- the pre-assembly of subdetectors at the surface (e.g. the LAr endcap calorimeter will be assembled and tested in the West Hall starting the beginning of 2001).

In June 1998 a joint controls workshop will be held in order to collect information on both the state-of-the-art of experiment controls and on the requirements of the LHC experiments. In addition, results of the investigation of the commercial controls market will be available. With this input all partners (the four LHC experiments and IT/CO) have to agree a project plan by the

end of 1998. This plan has to specify in detail system requirements, selection and decision mechanisms, time plan and milestones, and the resources needed and available. Only when this is achieved can we have confidence in the success of the joint controls project.

6.5 References

6-1 H. Burckhart and D.R. Myers, ATLAS detector control system user requirements document,
http://atlasinfo.cern.ch/ATLAS/GROUPS/DAQTRIG/DCS/urd951123.ps

6-2 V.Khomoutnikov and Yu. Ryabov, ATLAS DCS configuration database: Current state and general problems for discussion, ATLAS DCS-IWN 4. December 1997,
http://nicewww.cern.ch/~hallgren/dcs/note1297.PS

6-3 H.Burckhart, D.Fernandez-Carreiras, M.Gonzales-Berges, H.Milcent, D.R.Myers, A prototype application for evaluating EPICS and CANbus, ATLAS DAQ—No—80,
http://nicewww.cern.ch/~hallgren/dcs/evaluation.htm

6-4 M.Gonzales-Berges, H.Milcent, D.R.Myers, TIS-4000 evaluation, IT-CO Internal Note, February 1998, http://itcowww.cern.ch/jcop/subprojects/TIS4000/

6-5 W.P.J.Heubers, H.Boterenbrood, J.T.van Es, R.G.K.Hart, CAN field bus for industrial controls applications in high energy physics experiments, presented at the SYSCOMMS conference, CERN, March 1998.

7 Plans for work up to the technical proposal.

The overall development plan of the ATLAS high-level trigger, data acquisition and detector control system is proceeding according to the following phases and milestones:

- Work on the functional scheme. This includes the analysis and specification of the requirements, studies of architectural solutions, and investigations of technology. This phase is ongoing and will culminate with the choice of a baseline architecture which will be documented in the technical proposal at the end of 1999.
- Predesign studies. This includes the rationalization and optimization of the baseline architecture and predesign implementations of architecture components for validation. This phase will culminate with the development of the high-level design of the systems which will be documented in the technical design report in June 2001.
- Detailed design and implementation. Based on the overall high-level design, the detailed design of system components and their implementation will be done following a plan which will take into account the installation needs of the detector.

Based on the status of developments presented in the previous chapters, the detailed plan of the work necessary for the technical proposal is presented here. After the specification of the objectives for the technical proposal (Section 7.1), the work required to complete the analysis of the requirements, including those of the external systems, is presented (Section 7.2). The general approach of the validation and integration strategies is discussed (Section 7.3). The specification of the integration and validation plans for each of the systems are detailed for the LVL2 trigger (Section 7.4), the DAQ and event filter (Section 7.5), and the DCS (Section 7.6). Finally, the global integration and validation plan of the three systems as required by the technical proposal milestone is given (Section 7.7 and Section 7.8).

7.1 Objectives for the technical proposal

The goal of the ATLAS high-level trigger, DAQ and DCS technical proposal (TP) is the presentation of a description of the *global* ATLAS high-level trigger, DAQ and DCS *architecture* (hardware and software) that has been *proven to be valid*.

Architecture is the structure supporting the functional elements and their relationship in order to provide the required functionality.

Global high-level trigger, DAQ and DCS architecture does not necessarily imply, although it does not exclude, that all the functions are provided by a unique system. In the case of several coupled systems, interfaces and relational aspects must be defined and proven to fulfil the global requirements (see Section 7.2).

By '*proven to be valid*' we intend that at least one implementation (hardware and software) can be demonstrated to fulfil the requirements of performance, operability, and scalability at an affordable cost.

- Performance: at the component and subsystem levels should be close to final. If it is not, the reason for inadequate performance must be understood and corrective action must be taken by defining an evolution path (perhaps using different technologies) within the ar-

chitectural description. If the performance still remains inadequate then the architecture must be modified.

- Operability: all features that guarantee effective and efficient operation (e.g. maintainability, robustness, proper documentation, etc.).
- Scalability: via direct measurements and / or modelling.
- Affordable cost: total projected cost must not exceed the CORE 7 envelope, see MoU [7-1].

The architecture must be:

- open, comprising modular elements with well-defined interfaces, to allow implementations based on international standards; this feature is necessary in order to cope with evolution of requirements and technological advances;
- partitionable, to permit concurrent operations of the various detector subsystems.

7.2 Analysis of requirements

Although much effort has gone into the analysis of the ATLAS requirements on the trigger, DAQ and DCS, it has not yet been possible to complete their specification. In order to ascertain the validity of the architecture that the present prototype work aims to attain, it will be necessary to specify all of the relevant requirements, from physics and from all the ATLAS systems. This work is organised via two inter-system working groups: the trigger performance group and the detector interface group.

7.2.1 Trigger performance

Much of the physics at the LHC will be obtained by using inclusive triggers at all levels of selection. Some channels will require more exclusive triggers most of which will be performed at the event filter level where the event is fully assembled and more detailed analysis can be performed. Some semi-exclusive selections will be needed at LVL2, e.g. for B-physics. It is expected that in most cases, with the notable exception of B physics, the full information of the inner detector will not be required.

The tools developed up to now by the trigger performance group have been targeted to the specific design of the LVL1 and LVL2 triggers using dedicated algorithms, while the EF aims at reusing as much as possible the offline algorithms. An evaluation of the global trigger performance will be undertaken through a combination of detailed full-simulation studies and parameterized fast-simulation studies with high-statistics background samples. Data samples common to all studies will be produced. The menus and rates presented in [7-2] will be used as a basis for menus for more detailed studies of both the LVL2 trigger and the event filter, in terms of performance and of implementation. Those trigger items that are considered particularly challenging or critical will be subject to detailed trigger performance studies using fully simulated data as input and offline reconstruction code as a reference.

Considering the ever-increasing available computing power, the studies of the EF performance will show how close one can approach the offline full-event selection. In a similar way, the goal of the LVL2 studies will be to evaluate how complex the selection strategy can be within the implementation possibilities. An overlap between the LVL2 algorithms and the EF ones will exist

where a given selection can be made at either level. The boundary between the LVL2 and EF selection must remain flexible in order to match the evolving requirements of physics interest to changes in running conditions and the capabilities of the two systems. More precise milestones will be established at the time of the Technical Proposal.

7.2.2 Detector interface

The study of the problems associated with the integration of the detectors with the LVL2-DAQ/EF-DCS architecture will be the main area of work for the detector interface group in the coming 18 months. The principal requirements and limitations of all detector systems need to be listed and understood in the context of the overall trigger/DAQ architecture. Equally the requirements of the trigger/DAQ system on the detector systems need to be addressed. Coherent and exhaustive sets of figures describing channel counts, data sizes and rates etc. need to be compiled and are essential in order to gain a global view of the possible problems. A start has been made, as outlined in Section 5.2.1 with particular regard to DAQ/EF – detector integration issues. Examples of issues already identified are: monitoring and calibration capabilities and associated dedicated triggers, database requirements, ROD and ROB data format, mapping of detectors into ROBs, and local and global system control. Specific issues of integration between LVL2, DAQ/EF and DCS systems will not be treated directly by the detector interfaces group.

Joint discussions between representatives from each of the detector systems and from the LVL2-DAQ/EF-DCS community will begin in Q3/98 and continue into 1999. These discussions will be required to:

- Produce a detailed list of detector issues which need to be addressed
- Document in a coherent fashion the information which exists around each identified issue
- Establish a procedure to tackle individual points and identify a small group of people who will study and propose solutions to each of the problem areas.

7.3 Integration and validation strategies

The integration programme is the one dictated by the technical proposal milestone, i.e. the architectural description. The prototype implementations should be used to prove the validity of the architecture. The validation strategy foresees measurements, evaluations, and assessment of the architectural choices at three levels:

- the component level (e.g. back-end DAQ process manager or LVL2 supervisor),
- the subsystem level (e.g. event filter),
- the system level (i.e. trigger, DAQ or DCS).

The integration of the components into subsystems, and then subsystems into the systems follows the step-by-step plan described below. Each step generates a subsystem for which the relevant requirements, as specified later in this section, must be satisfied before proceeding to the next integration step.

The final integration step involves all the system prototypes in order to validate the architectural description of the ATLAS high-level trigger, DAQ and DCS. The global architecture will be based on three main subsystems:

- the LVL2 trigger,
- the DAQ / event filter,
- the detector control system.

In addition, it is necessary to integrate, at least partially, other systems to complete the architecture validation:

- a minimum number of detectors with their final prototype readout electronics (RODs);
- the LVL1 trigger and the TTC system prototypes or emulators.

Furthermore, one will have to ascertain the possible implications on the architectural choices, in particular for the last phase of the dataflow and for the event filter, of functions such as:

- the data recording,
- monitoring and calibration,
- the offline reconstruction and physics analysis.

Initial integration will involve minimal coupling of the systems, but sufficient to satisfy the technical proposal requirements. Although maybe not the cheapest, a minimal coupling integration is operationally the simplest. This will form the baseline solution for the ATLAS high-level trigger, DAQ and DCS architecture. Subsequently, the time available before starting the final design can be used to study and evaluate different architectural descriptions, with different levels of system integration, a different mapping of the functionality onto system elements, and alternative technologies. The final design will be based on the architecture and the technologies that optimize a number of criteria, including cost effectiveness, complexity, performance, and operability.

The following sections describe the specification of the validation and the integration plan of each system, followed by the overall integration plan. The time sequence of the plan is indicated via milestones with completion dates for the major steps.

7.4 Integration and validation plans for LVL2 trigger

7.4.1 Work plan overview

Detailed work plans for each of the activities are described in a separate document [7-3]. Figure 7-1 summarises the major phases and milestones of the project. There are five main phases, each one finishing with a series of milestones to be reviewed in a specific meeting or workshop.

Phase 1 from March 1998 to June 1998
This corresponds to the definition of the work plans for each activity. This phase ends at the June 1998 ATLAS LVL2 plenary meeting, with the approval of the work plans and milestones. A review of the standard parameter values commenced in this period.

Phase 2 from June 1998 to October 1998
This corresponds to the development phase of the components for the LVL2 testbeds. A review of the work will be done during the October trigger / DAQ workshop in Chamonix. In addition

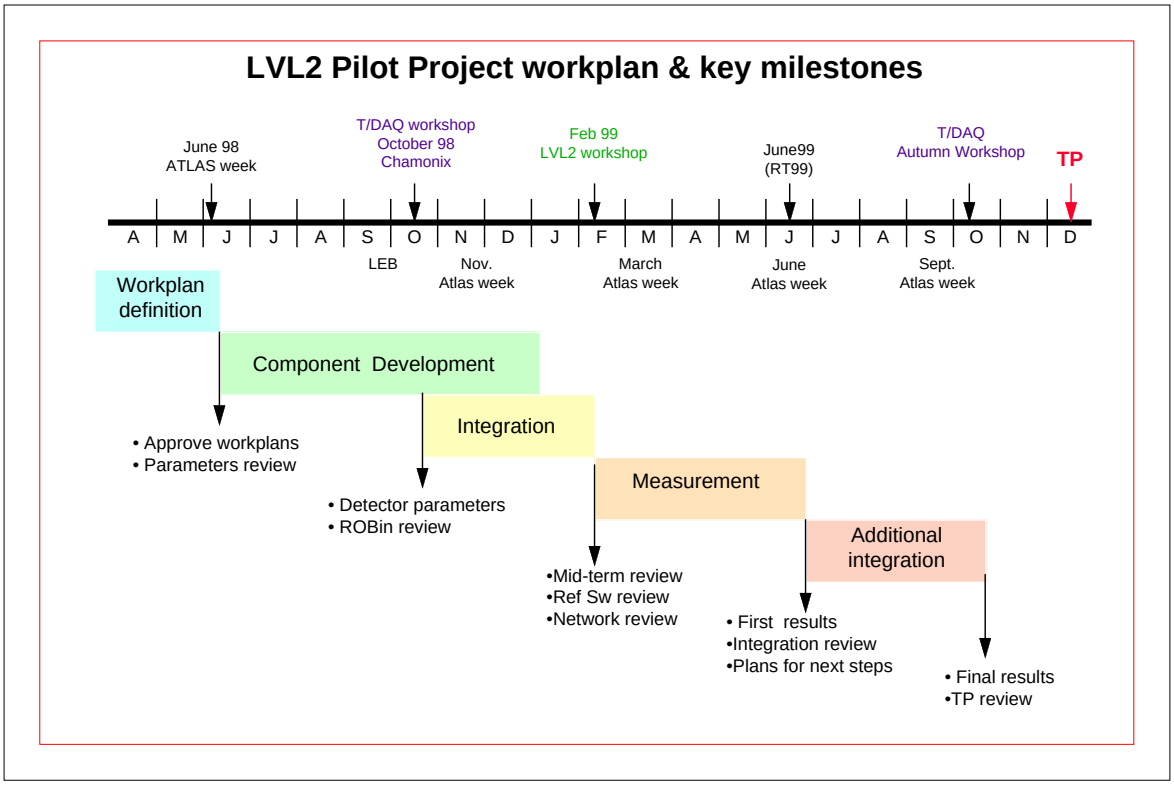


Figure 7-1 LVL2 integrated work plan and milestones.

at this workshop we should confirm the conclusions taken at the end of the demonstrator programme and consider if they are still valid and review the ROB Complex studies.

Phase 3 from October 1998 to February 1999
Start the local integration of testbed components (Reference Software, RoI-builder). At the end of this period, corresponding to the 'mid-term' of the project, the detailed plans for measurement and validation of the ATM and Ethernet testbeds will be finalised. In addition, it is planned to hold an internal review of the various networking technology developments

Phase 4 from March 1999 to June 1999
This corresponds to the global integration of all elements in a large application testbed at CERN. It is during this phase that the main performance measurements of the full slice system commence. A first set of results should be reviewed during the June 1999 ATLAS week.

Phase 5 from June 1999 to October 1999
This is the continuation of the measurements on the full-scale application testbed(s). It is also during this time that we can discuss the next phase of integration towards the ATLAS high-level trigger, DAQ and DCS architecture. Towards the end of this phase it is foreseen to hold a trigger / DAQ workshop, to review all the material necessary for the technical proposal.

7.4.2 Functional components workplan

The functional component activity (Table 7-1) is structured into three main areas, ROB complex, supervisor and technology studies (processors, interfaces and networks).

Table 7-1 Functional components workplan and milestones.

Activity				Milestones	
Description of the task	**Type**	**Objective**	**Use**	**First**	**Last**
ROB complex					
Requirements	paper, model	report	ROB URD	Dec 98	Oct 99
Functional prototypes	h / w, s / w	ROBin	test beds	Oct 98	Dec 98
Implementation scenario studies	paper study	notes	TP		Oct 99
Supervisor					
RoI builder	h / w, s / w	ROIB–SUP	test bed	Oct 98	Dec 98
Technologies					
FPGA					
Coprocessor (TRT full scan, etc.)	s / w	Enable++ integration	test bed	Oct 98	Jun 99
Study of general use	simulation, doc.	notes	TP	Dec 98	Oct 99
HPCN					
Traffic pattern	model	report	generic	Sep 98	Dec 98
System implementation	s / w	report	test bed	Sep 98	Dec 98
Fast/Gigabit Ethernet					
Baseline performance	measurements	report	test bed	Jun 98	Dec 98
Linux driver	s / w	integration	test bed	Dec 98	Feb 99
Scalable switching networks	h / w, s / w	64 ports HS link	ARCHES	Jul 98	Feb 99
Switch tester	h / w	integration		Jul 98	May 99
High performance network interface	h / w, s / w	intelligent NIC	test bed	Jul 98	Jul 99
SCI					
Interconnect and switch studies	s / w	report		Oct 98	Dec 98
ATM					
622 Mbit / s interface test	s / w	integration	test bed		Dec 98
Multiswitch-multipath networks	s / w, model	report	TP	Dec 98	Feb 99
Technology watch					
FPGA, processors, networks	doc., infos	notes	generic	Dec 98	Oct 99

7.4.2.1 ROB complex

The tasks and milestones in this area are:

- Complete development of functional prototypes of buffers (ROBin) to be used as elements of the ROB complex and integrated into the application testbeds (Q4/98).
  (For these studies one of the options is the intelligent PMC to be used in the studies of the ROB input channel in Section 7.5.1.1).
- Establish the LVL2 hardware and software requirements for the ROB complex. (Q3/99)
- Study different design options and implementation scenarios with respect to possible architectures. (Q3/99)

7.4.2.2 RoI builder and multi-processor supervisor

The milestone established in this area is:

- Demonstrate CPU scaling of supervisor. (Q4/98)

7.4.2.3 Technologies for processors, interfaces and networks

The programme in this area is to evaluate a small number of promising technologies, track industry trends, and develop systems which match ATLAS requirements. The tasks and milestones are:

- Model HPCN traffic patterns. (Q4/98)
- Implementation and test of HPCN system. (Q4/98)
- Establish baseline performance measurement of commodity, off-the-shelf, Ethernet hardware and software, and evaluate performance with respect to LVL2 needs. (Q4/98)
- Measure and understand I/O performance of SCI switches. (Q4/98)
- Evaluate 622 Mbps ATM interface. (Q1/99)
- Gain experience with multiswitch and multipath ATM networks. (Q1/99)
- Design and implement dedicated optimized drivers under Linux. (Q1/99)
- Gain early experience with scalable Gigabit Ethernet switch. (Q1/99)
- Build a prototype of Ethernet switch tester. (Q2/99)
- Evaluate performance and limit of FPGA-based system used as co-processor (TRT Full scan). (Q2/99)
- Study general applications of FPGA in LVL2. (Q3/99)
- Produce a design study for Fast and Gigabit Ethernet intelligent network interfaces. (Q3/99)
- Report on industry trends and developments in processor and networking technologies. (Q3/99)

7.4.3 Testbed workplan

The testbed activity (Table 7-2) is structured into three main area, the two parts of the reference software and the application testbeds.

Table 7-2 Testbed workplan and milestones.

Activity			Milestones	
Description of task	**Type**	**Deliverable**	**First**	**Last**
Algorithms				
Define representative menus	doc.	note	Jun 98	Oct 98
Establish training samples	s / w	data files		Nov 98
FEX development and benchmarks	s / w	files	Jul 98	Feb 98
Steering sequences	0	files	Dec 98	May 99
Reference software framework				
Design	doc	note		Jun 98
Implementation	s / w			Dec 98
Operation	s / w	report	Jan 99	Jul 99
ATM test bed				
Algorithms and reference software	s / w integration		Jun 98	Feb 99
RoIB–multi-processor supervisor integration	h / w, s / w			Dec 98
Enable++ full scan TRT integration	h / w, s / w			Nov 98
ROBin and ROB complex integration	h / w, s / w		Oct 98	Dec 98
Performance 32 / 64 port network	measurement	report	Jan 98	Jun 99
Commodity Ethernet				
Fast Ethernet 8 / 32 port test bed	s / w		Oct 98	Jan 99
Process model implementation	s / w		Jan 99	Jun 99
Performance with optimized components	measurement	report	Oct 98	Oct 99

7.4.3.1 Reference software algorithms

The reference software algorithms work plan is divided into the following tasks:

- Definition of representative menus and a training sample: select a set of physics reactions (signal and background) which will be sufficient, over a time span of the pilot project, to optimize trigger strategies and algorithms. (Q3 / 98)
- Establishment of training samples: provide a choice of input data for studies of trigger strategies and for testbed algorithms.(Q4 / 98)

- Preparation of tools for analysing training sample, in C-language: provide algorithms, later to become part of the testbed reference software. These algorithms take training samples as input, and serve as tools for trigger studies and algorithm optimization. (Q1/99)
- Provision of general algorithm steering in an object-oriented environment: develop a more general steering program for decision making, using available feature extraction algorithms; calls to the feature extraction algorithms are according to rules written down in tables (menus, algorithms, trigger objects). Document the available routines for other users. (Q2/99)

7.4.3.2 Reference software framework

The reference software framework work plan covers the following areas:

- Framework software design. The framework allows implementation as a distributed system and defines a protocol and messages for control and exchange of data between the system components. It includes an interface to algorithms (global, feature extraction, preprocessing) which may be linked in and executed on-line. An API for message passing will be provided but only tested with TCP/IP; optimizations for particular technologies are outside the scope. Likewise, ROBs and supervisor will be emulated but integration of custom hardware is outside the scope. The result will be a design based on analysis of the requirements and pre-implementation studies. OO design following good software engineering standards has been adopted. (Q2/98)
- Framework software implementation. Implementation of the software as described under design for PCs running Windows NT or Linux and TCP/IP for communication. Supervisor and ROBs will be emulated on PCs. Algorithms will be integrated and run on simulated data that may be preloaded in emulated ROBs. (Q4/98)
- Framework software operation. Continuous monitoring of the quality and performance of the common framework software; new releases when appropriate. (Q3/99)

7.4.3.3 Application testbeds

The testbeds will be used to integrate and test the reference software and, also, to integrate and test special components. The tasks and milestones are given below.

- Integrate ROBin modules in ATM testbed. Study data transfer, partial readout, and preprocessing. Demonstrate feasibility of grouping several data sources on a single network port. (Q4/98)
- Integrate RoI builder and multi-processor supervisor. (Q4/98)
- Evaluate performance, limits, savings and cost of hybrid full-scan TRT. Integrate and optimize track-finding on Enable++ with global farm. Compare to reference algorithm. (Q4/98)
- Implement trigger algorithms. Read data from training samples into testbed ROBs. Perform sequential LVL2 algorithms with multiple data requests. Install full trigger menus and global selection algorithms. (Q1/99)
- Set up a system of PCs running Linux connected together by a Fast Ethernet switch. Port the Reference software to the system. Implement a range of process models on the commodity testbed. (Q1/99)

- Establish a 32 to 64-port testbed for system performance studies with commercial components. Measure full system performance on a large testbed, including preparation and transfer of raw data and execution of trigger algorithms based on flexible trigger menus. (Q3/99)
- Measure performance parameters of testbed(s) running agreed menus and strategies. (Q3/99)
- Provide a platform for the integration and test of optimized components. (Q3/99)

7.4.4 System design workplan

The system design activity (Table 7-3) is structured into two main areas: modelling and global integration, most of the latter is described in the relevant parts of Section 7.8.

Table 7-3 System design workplan and milestones.

			Milestones	
Description of task	**Type**	**Deliverable**	**First**	**Last**
New parameters – paper model update	paper model	report	Jun 98	Oct 98
Fill generic 'architectural' model	SIMDAQ C++	report	Oct 98	Oct 99
Full generic 'test bed' models	SIMDAQ/Ptolemy	report		Oct 99
Full generic 'technological' models	SIMDAQ/Ptolemy	report		Oct 99
Large scale ATM emulator	s/w–emulation	report		Sep 98
Macrame-ARCHES emulators	s/w–emulation	report	Jun 98	Mar 99
Physics: baseline strategy for the TP	doc	note	Nov 98	May 99
LVL1	doc	note	Nov 98	Mar 99
Detector	doc	note	Jan 99	Jun 99
User Requirement Document	doc	TP	Sep 98	Jun 99
System Requirement Document	doc	TP	Oct 98	Oct 99
System Design and Costing Document (SD)	doc	TP	Oct 98	Jul 99

7.4.4.1 Modelling and emulation

The work programmes in computer modelling and system emulation are to:

- Measure performance for simulated LVL2 traffic on a large system using state-of-the-art multiprocessor servers and workstations interconnected by a commercial ATM backbone network. (Q3/98)
- Review parameters and update the paper model. (Q3/98)
- Measure the response of the Macramé and ARCHES switching testbeds to ATLAS trigger traffic. (Q1/99)

- Implement and study the behaviour of the full generic system model. Study different options with respect to possible designs of the LVL2 system. (Q3/99)
- Implement and study the behaviour of generic models (i.e. without detailed models of different technologies used) of testbeds systems. (Q3/99)
- Implement and study the behaviour of full models (i.e. with detailed models of different technologies used) of testbeds and options for complete systems. (Q3/99)

7.4.4.2 Integration

The following documents are needed as input for the technical proposal. The milestone for their completion has been set as Q3/99.

- The user requirements document (URD) is our principal benchmark for the specification of the function of the LVL2 trigger. The present version will need to be updated if this valuable tool is to remain of use for the next phase.
- A system requirements document (SRD).
- A system design document (SDD) with an outline LVL2 design and costing.

7.5 Integration and validation plans for DAQ/event filter

The architectural and technological studies necessary for the choice of the DAQ and event filter architecture have been structured in a system wide context, where the validity of components and sub-systems can be ascertained not only in themselves but also integrated in their system environment. We have chosen, therefore, to build a DAQ/EF prototype system (DAQ/EF-1) supporting the full functionality from detector readout to data recording.

Starting in December 1995, the project has been planned in three phases:

- Pre-design studies and high level design (1996–97)
- Detailed design and implementation (1997–98)
- System integration and exploitation (1998–99).

At the moment of writing the project is reaching completion of the final implementation of individual components and has started the system integration process, which is proceeding according to the following plan:

1. Integration and validation of all the dataflow units (Section 7.5.1)
2. Integration and validation of all the back-end DAQ core components (Section 7.5.2)
3. Integration and validation of all the event filter components (Section 7.5.3).
4. Integration of the dataflow units with the back-end DAQ core components (Section 7.5.4)
5. Integration of the event filter in the dataflow (Section 7.5.5)
6. Integration of the event filter with the back-end DAQ (Section 7.5.6).

Figure 7-2 is the top level chart of the DAQ/EF project planning. It summarises the main steps of the integration.

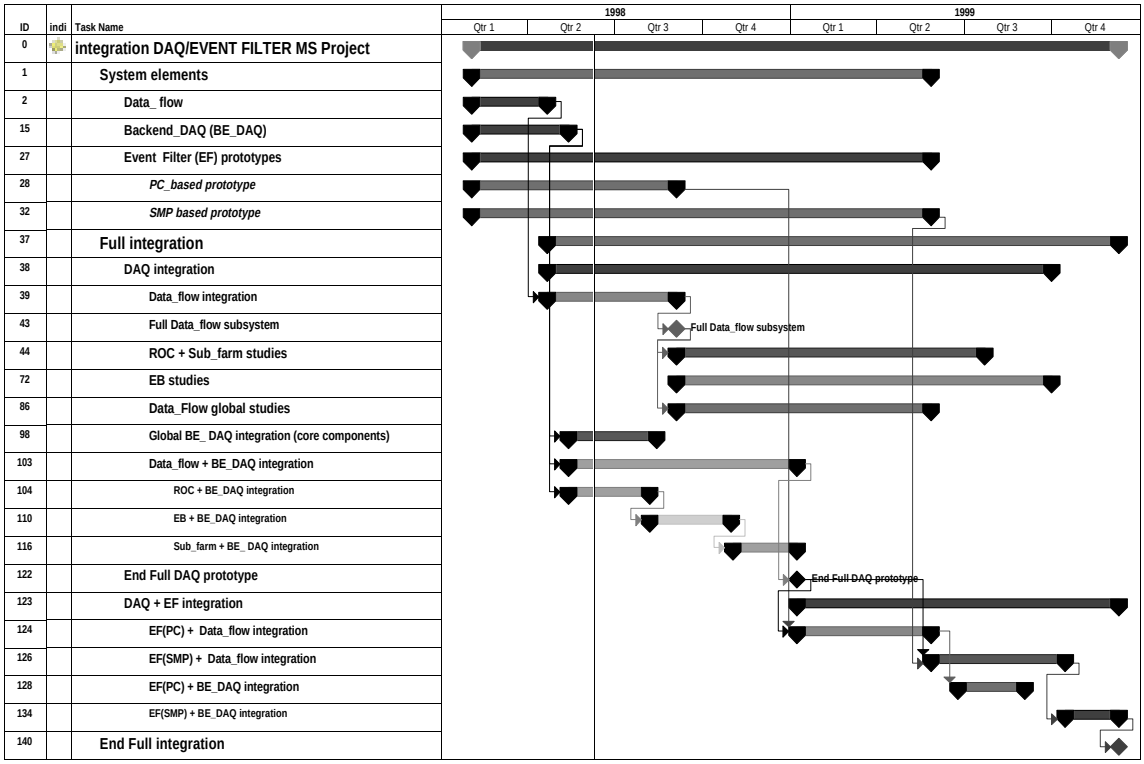


Figure 7-2 DAQ/EF main integration steps.

As shown in the chart, there are four major milestones:

- The completion of the full dataflow subsystem. (Q3/98)
- The completion of the back-end DAQ core components integration. (Q3/98)
- The completion of DAQ prototype system. The full integration of the dataflow and back-end DAQ sub-systems gives the complete and operational DAQ system core, providing all the functionality required for the transport, the local and central control, the local and central monitoring of unselected physics data from the input to the Read-Out Buffer to data recording, as well as those required for the configuration and operation of the data-taking process. (Q4/98)
- the completion of the full DAQ/EF integration. The integration of the event filter subsystem and the DAQ completes the integration of all the functional elements sitting on the main dataflow path and adds the full event processing capability to support any combination of functions such as: event selection, detector calibration, detector and physics monitoring, output stream organisation. (Q4/99)

Each of the above integration steps produces a subsystem which will be the basis for predesign architectural studies, technology (hardware and software) evaluations and comparisons, performance measurements, and assessment of solutions. The fully integrated system will also be the basis for the study of global system requirements other than the basic performance ones, such as those coming from practical and operational needs. Examples are: system initialisation, partition and scalability, error detection and recovery, control, calibration and monitoring needs, software operability and maintainability.

The timescale and milestones of the integration steps and successive studies are given in the remainder of this section. More details of the studies are described in Section 5.3,

7.5.1 Dataflow integration and validation

The fully integrated dataflow subsystem provides the complete functionality required for the transport, control and monitoring of physics data from the Read-Out Buffer to data recording. It can be used to assess the validity (or study alternatives) of the ATLAS dataflow. The full dataflow subsystem will be available, according to the steps outlined in 5.2.2.6 in Q3/98.

Once the baseline dataflow is integrated, a number of studies and evaluations of alternative options can be performed both at the level of its three units (the readout crate (ROC), the event builder (EB) and the sub-farm, Section 5.2) and at the global dataflow level. Such studies are grouped into two categories:

- Those which address local aspects of the systems, where local means both a single element, such as the ROB, or an entire unit, such as a full ROC. Studies in this group broadly address the following two issues:
    - Can performance be achieved, locally (e.g. dataflow control within a ROC), which is close to the final performance required by ATLAS? If this is not the case, the reason should be identified (e.g. inadequate technology, architectural bottleneck, software, interaction between any of the above, etc.).
    - Functionality at the component level, with emphasis on those components common to different architectures.
- Those related to the global dataflow subsystem aspects as a function of architectural options.

In the following sections we outline the work-plan for the dataflow studies both at the local level of the three dataflow units and at the global level of the integrated dataflow. More details of the studies are provided in Section 5.3.1.

7.5.1.1 Readout crate studies

The baseline implementation of the Read-Out crate is depicted in Figure 7-3 where we highlight:

1. The elements (see 5.2.x): ROB, TRG, EBIF and LDAQ
2. The connections to other systems: the detector (via the ROB element), the triggers (via the TRG element), the event builder (via the EBIF element) and the back-end DAQ (via the LDAQ element)
3. The intra-crate links:

- The data collection link, for event data flowing from the ROBs to the EBIF
- The TRG link, for dataflow control commands (such as L2 accept/reject) to be distributed to ROB's and/or EBIF

The LDAQ link, for communications (run control, errors, monitoring) between the LDAQ and the other main dataflow elements

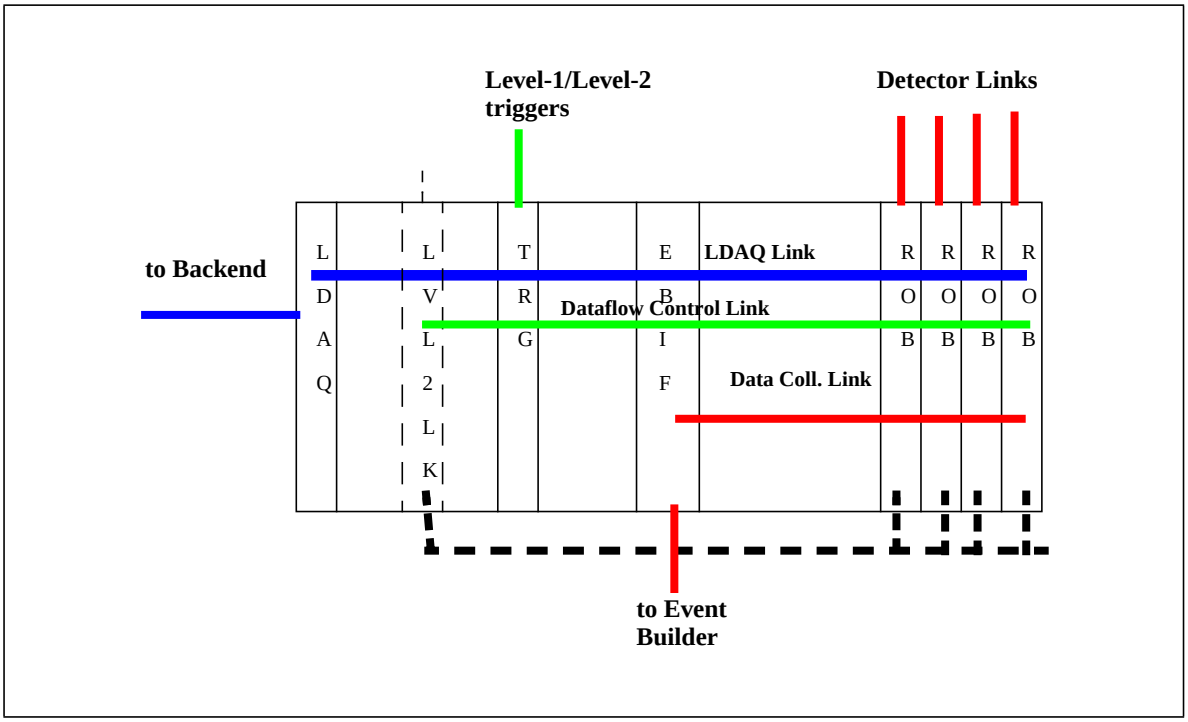


Figure 7-3 Baseline ROC implementation.

- Intra-crate communication, between the various ROC elements, which include:
  - Dataflow control: communication between individual ROC elements, i.e. the trigger interface (TRG), the event builder interface (EBIF) and the readout buffer (ROB) modules. We plan to address:
    - more detailed studies of the use of the VMEbus (Q3/98)
    - evaluation of PVIC and Raceway as secondary buses for dataflow control (Q4/98)
    - deliverable: summary document of results. (Q1/99)
  - Local data collection, the merging of event fragments from a number of ROB modules into a single 'crate' fragment, which includes:
    - more detailed studies of the use of the VMEbus (Q3/98)
    - evaluation of PVIC and Raceway as secondary buses for local data collection (Q4/98)
    - deliverable: summary document of results. (Q1/99)

- Read-Out Buffer, one of the critical elements in the dataflow. We plan to address:
  - ROB input channel:
    - completion of the baseline (RIO II) SBC-based ROB (Q4/98)
    - integration of an intelligent PMC providing ROB input and buffer management (Q4/98)
    - performance studies of a SBC-based ROB on more advanced platforms, including the use of multiple PCI buses (Q1/99)

- assessment on how to approach the ROB implementation: fully based on commercial components vs. customization of most critical parts (Q1/99)
- deliverable: summary document of results and assessment. (Q2/99)

- Single vs. multi-ROB input. One or more input channels could be grouped together under the control of a central ROB core, the latter providing the communication with the rest of the ROC. Studies in this area should be completed, and a document produced by the end of Q1/99.
- Software performance, including: the event manager performance, the I/O scheduler performance; deliverable: summary document of results. (Q1/99)

- Multiple I/O module, a stand-alone ROC, in different implementations of the functional architecture:
  - collapse TRG and ROB functionality into single implementation (Q2/99)
  - collapse EBIF and ROB functionality into single implementation (Q2/99)
  - collapse TRG, EBIF and ROB functionality into single implementation (Q2/99)
  - deliverable: document summarising the results from the different architectural options and assessing their relative advantages and disadvantages. (Q3/99)

See also Section 7.4.2.1.

7.5.1.2 Event Builder studies

The dataflow event builder is depicted in Figure 7-4. It highlights the architectural view of the event builder, consisting of:

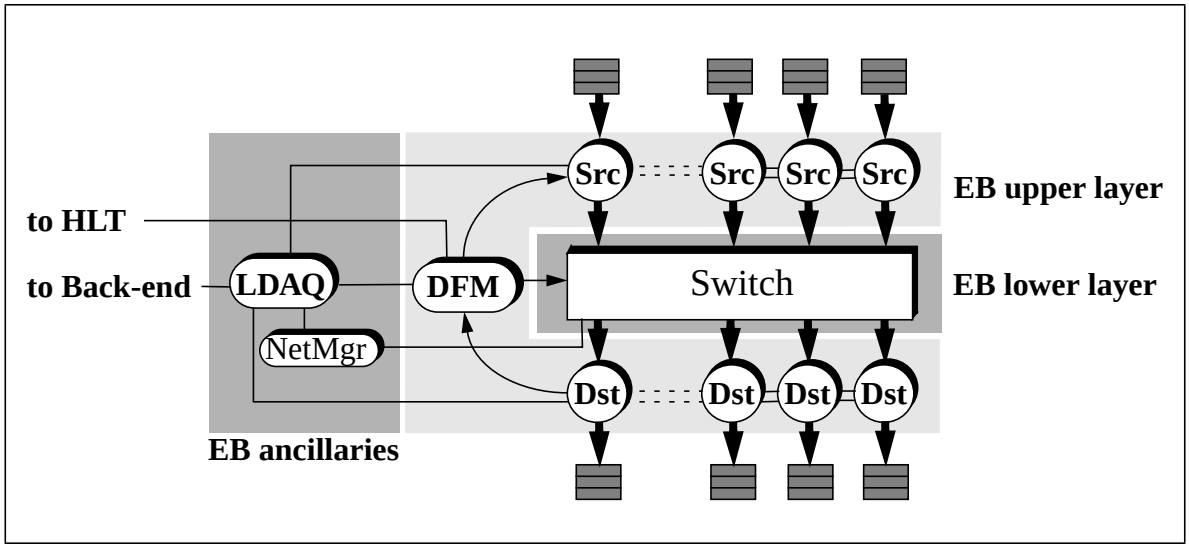


Figure 7-4 Event builder architectural view.

1. An upper (technology independent) layer including:
   - Source (Src) processes, feeding ROC fragments into the event builder

- Destination (Dst) processes, receiving fragments and assembly into a full event those belonging to the same event identifier.
- The Data Flow Manager (DFM) implementing the event builder control protocol (destination assignment, event completion, error handling)

2. A lower (technology dependent) layer consisting of the physical switching network which provides the transport of the data between sources and destinations. The switched Ethernet, Fiber Channel and ATM technologies will be used for the studies.
3. An ancillary layer consisting of control elements: the local DAQ (LDAQ) for run control and interfacing to the back-end DAQ, the Network Manager (NetMgr) to control the switching network

The studies in the area of event building will evolve around the following topics:

- The study of an NxM event builder using three technologies: 1) Switched Ethernet, 2) FCS, 3) ATM.
  - (N = no. of sources = 5–6, M = no. of destinations = 3–4 for initial local measurements) (Q1 / 99)
  - Deliverable: summary document of performance measurements. (Q2 / 99)
- Functionality and reliability studies:
  - suitability of the event builder protocol (Q2 / 99)
  - behaviour under error conditions. (Q2 / 99)
- Data Flow Manager performance according to:
  - implementation (distributed vs. centralised, mixture of the two) (Q4 / 98)
  - communication technology (same network as data, PVIC or other shared memory system, reflective memory network) (Q1 / 99)
  - deliverable: summary document of results and choice of optimal implementation. (Q2 / 99)
- Scalability: behaviour of the event builder in a large, final ATLAS-like configuration.
  - Modelling and emulation. Summary document. (Q3 / 99)

7.5.1.3 Sub-farm studies

The dataflow aspects of the sub-farm are studied, using a dummy event handler, with measurements and assessments analogous to those described for the ROC.

The studies more specifically related to the real event handler and selection are part of the event filter subsystem work plan.

7.5.1.4 Dataflow global studies

- Alternative configurations of the dataflow units:
  - one ROC / EB input
  - one or more ROBs / EB input

    - no. of EB outputs / Sub-Farms
- Overall performance
- Overall operability, including high availability and error handling/recovery.
- Assessment of overall dataflow functional architecture for different options as a function of:
    - complexity (of implementation, operation, etc.)
    - performance
    - cost.

The dataflow global studies will be performed, and a summary document produced during Q1+Q2/99.

7.5.2 Back-end DAQ integration and validation

The integration of the five core components of the back-end DAQ (see Section 5.2.4.3) is essential for the assessment of the validity of the DAQ/EF architecture. For each of them, test-plans have been developed for unit testing and validation prior to system integration. The test-plans are described in detail in Section 5.3.2. Their global features are summarised here for completeness of the overall work-plan description.

7.5.2.1 Unit testing of core components

Each of the five core components is subjected to unit tests to assess its functionality, performance, scalability, and reliability. The test are based on the two most relevant scenarios for an integrated DAQ system representing the likely configuration for the DAQ/EF-1 project and the final ATLAS DAQ/EF system. All tests are repeated on the four operating systems currently supported in the DAQ/EF-1 project (Solaris, HPUX, LynxOS and WNT) with a variety of compilers (GNU C++ and Visual C++) and combinations as required for a running DAQ system. The unit tests can proceed in parallel and are expected to be completed by the end of Q2/98.

The most relevant measurements of each component include:

- Configuration databases: Speed of various traversals, object updates and queries as defined by the 007 Benchmark with databases holding up to 100 000 objects. Further tests designed specifically for the DAQ configuration databases and usage of OKS.
- Message reporting system and information service: The unit tests for these two components are similar and include transport times from message/information sender to receiver and scalability tests to check the maximum number of senders and receivers that can be connected concurrently.
- Process manager: Process creation and destruction times are important because they contribute to the down time of the DAQ system when changing data-taking configuration or starting and stopping data-taking sessions. For scalability tests each agent (one per processor) creates and destroys up to 100 processes.
- Run control: The time taken to start and stop a data-taking run is crucial because it contributes to the down time of the DAQ system. For performance tests, the time taken to start and stop a data-taking run at the start of a data-taking session (cold start/stop),

when changing configuration (lukewarm start / stop) and between successive runs (warm start / stop) are measured. For scalability tests, the number of controllers in the control hierarchy is varied from 1 to 250.

7.5.2.2 Trigger/DAQ and detector integration components

The core components of the back-end DAQ have been given priority in order to have a baseline subsystem that can be used for initial tests. Work has started on the development of some of the trigger / DAQ and detector integration components and is being organised according to the same software process (Section 5.2.4.7) used for the core components:

Partition and resource manager
Implementation of the resource manager has started and it is expected that the unit testing will be completed by the end of Q4 / 98.

Status display
It is expected that a first version of the status display will be tested during the integration with the dataflow subsystem and be ready for the end of Q4 / 98.

Run bookkeeper
Implementation has started and it is expected that the unit testing will be completed by the end of Q4 / 98.

Event dump
The development of the event dump depends on the integration of the back-end core components, the completion of the monitoring task skeleton and, as yet undefined, manpower resources for its implementation. It is hoped that an implementation will be available for Q2 / 99.

Test manager
Implementation has started and it is expected that the unit testing will be completed by the end of Q4 / 98.

Diagnostics package
The development of the diagnostics package depends on the integration of the back-end core components, the completion of the test manager and, as yet undefined, manpower resources for its implementation. It is hoped that an implementation will be available for Q3 / 99.

7.5.2.3 Global back-end DAQ integration

The core components of the back-end DAQ will be combined and used to simulate the control and configuration of data-taking sessions. This requires completing the content of the configuration database to define all the necessary software elements. Processes will be launched by the DAQ supervisor (via the process manager) and the run control will marshal the hierarchy of controllers from the Initial to Running state then back again to simulate a single data-taking run. The cycle is then repeated to represent a full data-taking session. The integration of the core components is made in a step-wise, bottom-up manner according to the dependencies between components and the underlying external packages (both commercial and shareware) as shown in Figure 7-5. It is expected that the back-end integration tests will be completed during Q3 / 98.

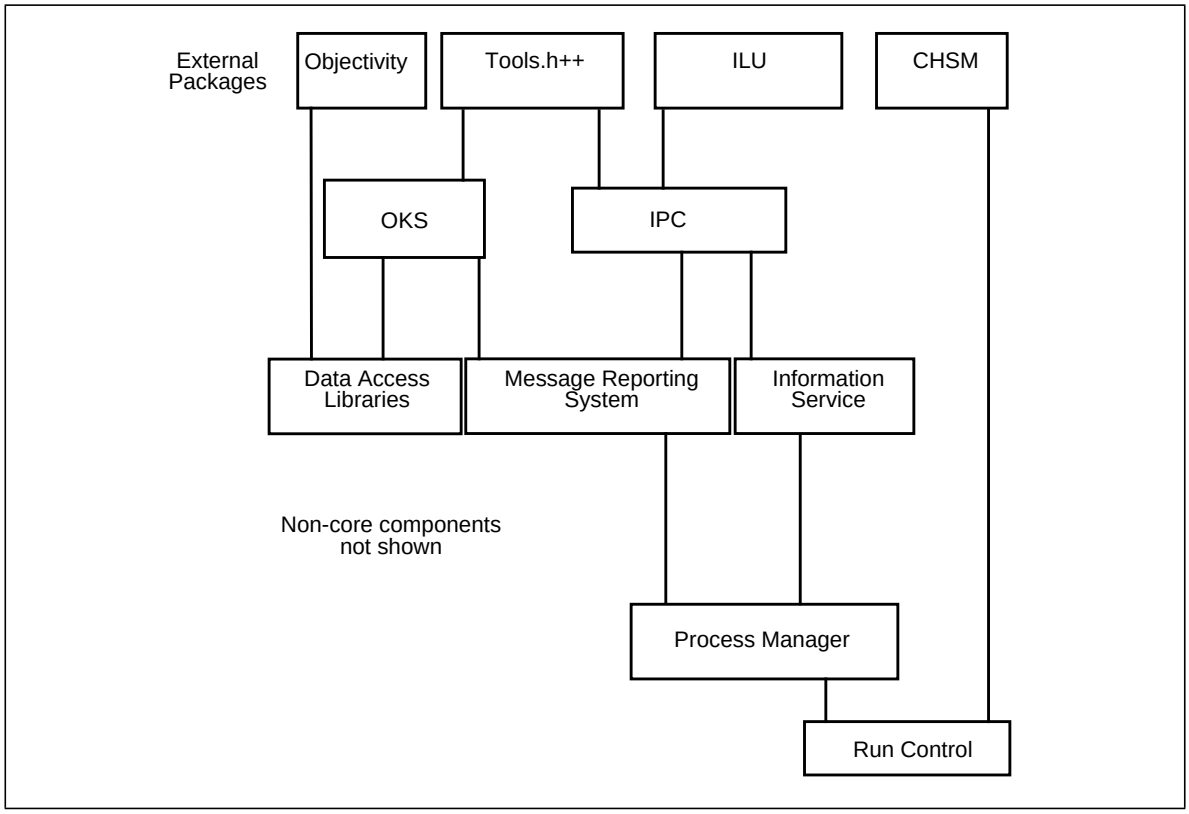


Figure 7-5 Dependencies between back-end core components and external packages.

7.5.2.4 Further developments

The unit and integration tests described above are intended to provide a first version of the back-end system for use within the DAQ/EF-1 project. It is expected that integration with other sub-systems will lead to the identification of possible improvements which, coupled to evolving underlying technologies, will form the basis of further developments of the back-end components. To date, such further developments to be made after integration with other sub-systems (i.e. starting after Q2/99) include:

Investigate alternative CORBA implementations and use of OMG version 2 services

Currently the basic ILU CORBA based communication package is used in the DAQ/EF-1 project but alternative implementations have recently become available and layered services have also been defined. We will investigate such packages and the possibility of interfacing Java clients to C++ servers.

Further integration of the Objectivity database and OKS

We intend to investigate the development of database independent (i.e. OKS or Objectivity) data access libraries and their possible generation from the StP CASE tool.

Decision making inside the run control

We would like to extend the DAQ supervisor's capacity for decision making and believe expert systems to be a good candidate technology for implementing the logic of such an intelligent supervisor.

7.5.3 Event filter

The event filter work is structured into three main areas: event filter software and physics strategy, prototype construction and development, and additional functionality.

7.5.3.1 Event filter software and physics strategy

- Study the options for physics strategies to be adopted in the EF.
- Study current reconstruction and analysis code to develop analysis strategies for EF event processing. Study input parameter precision as a function of processing time. This should allow the cost of a given precision level in analysis time to be estimated. Study the offline processing context, with particular emphasis on geometry and calibration databases and field map requirements.
- Develop realistic benchmark software based on the above to be used in studying the prototype performance.
- Develop an emulator program to simulate the performance and resource use of the analysis, to be used on the prototypes.

This phase of the project has started and will continue up to and beyond the Technical Proposal. Parameterization of the most relevant inclusive triggers and a better understanding of the EF processing should be achieved by Q1/99.

7.5.3.2 Prototype construction and development

Several prototypes and associated studies are envisaged based on different hardware architectures and technologies:

- A PC-based prototype is currently under test, running on 3 PCs today. It will be enlarged to 25 PCs by Q2/98.
- A HP-SMP prototype is planned for Q4/98 (other prototypes based on machines from other vendors are under active discussion).
- Finalize contacts with the University of Alberta and establish a tight collaboration for a sub-farm prototype based on quad Pentium II nodes. (Q3/98)
- Develop and test generic event distribution and monitoring schemes. A first design is completed and being used in the PC prototype. (Q2/98)
- Porting of event distribution scheme to HP-SMP. (Q4/98)

Other studies to be performed on the prototypes in Q4/98 and Q1/99

- issues related to system management, error handling and control, dataflow control, etc.
- physics performance as a function of the architecture parameters, making use of EF emulators
- scalability
- system modelling.

7.5.3.3 Additional functionality

Monitoring, calibration and possibly alignment tasks must be performed on the EF farm together with the primary event filtering.

- Design a prototype monitoring and calibration framework based on requirements identified in connection with detector, physics and offline groups. (Q4 / 99)

7.5.4 Dataflow—back-end DAQ integration

The three dataflow units are integrated in sequence with the five core components of the back-end DAQ. All the functions required from the dataflow side for the integration are provided by the local DAQ (LDAQ) component of the respective subsystems. The back-end DAQ core components are integrated with the dataflow units in the same sequence.

Integration plan:

- Read Out Crate (ROC)—BEDAQ. (Q3 / 98)
    - ROC LDAQ - Configuration databases
    - ROC LDAQ - Information service and message reporting system
    - ROC LDAQ - Process manager and run control.
- • Event Builder (EB)—BEDAQ (Q4 / 98)
    - EB LDAQ - Configuration databases
    - EB LDAQ - Information service and message reporting system
    - EB LDAQ - Process manager and run control.
- • Sub-Farm (SF) (DAQ aspects)—BEDAQ (Q4 / 98)
    - SF LDAQ - Configuration databases
    - SF LDAQ - Information service and message reporting system
    - SF LDAQ - Process manager and run control.

7.5.5 Event filter Integration in the dataflow

The integration of the EF with the dataflow essentially involves attaching the input / output elements of the dataflow Sub-Farm with the corresponding elements of an EF prototype. These interfaces will be implemented with a generic API, currently under definition.

- Implementation of API in Q3 / 98
- Integration of PC prototype in Q1 / 99
- Integration of SMP prototype in Q3 / 99.

7.5.6 Event filter Integration with the back-end DAQ

The event filter is integrated with the five core components of the back-end DAQ following the same sequence as described for the dataflow – BEDAQ integration. All the functions required from the event filter side for the integration are provided by the EF supervisor.

- EF supervisor – BEDAQ (Q2/99)
    - EF supervisor - Configuration databases
    - EF supervisor - Information service and message reporting system
    - EF supervisor - Process manager and run control.

7.6 Detector Control System

According to the current ATLAS strategy, the Detector Control System should be operationally independent from the Data Acquisition. However, that does not exclude the two systems from being based on common software tools and packages, where appropriate.

- One such common item is the configuration database. Work is in progress to assess the validity for DCS of the Objectivity/OKS scheme under use in the DAQ/EF prototype project. Assessment is expected by Q1/99.
- For other aspects, no time-scale can be specified before the possibility of a common DCS project amongst the four LHC experiment is established. The proposal for a DCS common project is expected by Q4/98.

7.7 External systems

As explained in Section 7.3, the integration (at least partial) of a number of other systems is necessary for the validation of the L2-DAQ/EF-DCS architecture.

7.7.1 Detectors

From the point of view of integration into the T/DAQ system, a detector is ready when:

- the ROD functions according to the standards of communication protocol and event fragment format specified in the FE/DAQ interface, the event format, and the detector interface working group documents [7-4][7-5][7-6];
- the requirements on configuration databases, run control, monitoring and calibration are specified and documented in the detector interface working group document.

7.7.2 LVL1 trigger

The control signals and trigger messages, such as the RoI related information, must be provided by emulators of the LVL1 trigger and TTC systems.

7.7.3 Data recording

Possible implications of the data recording system on the DAQ and event filter architecture must be analysed and understood before the architecture itself can be finalised. Similarly, the requirements of the DAQ and event filter on the data recording system must be specified.

7.7.4 Offline reconstruction and physics analysis code

Possible implications of the offline reconstruction and physics analysis code on the event filter architecture and of the offline event organisation on the DAQ architecture must be analysed and understood before the architecture itself can be finalised. Similarly, the requirements of the DAQ and event filter on offline code must be specified. Online and offline data formats are particularly important issues to be studied in collaboration with the offline community.

7.8 Overall integration plan

Once all the individual trigger/DAQ systems are internally integrated and validated according to the plans and criteria specified above, the global integration can commence. As explained in Section 7.3, the integration of the prototypes will be limited to elements and subsystems required by the validation of the global architecture for the Technical Proposal. The level of integration of the prototypes should be one of the minimal coupling to satisfy the requirements. There is no point investing in the integration of 'throw-away' prototypes into a detailed final-like system, beyond that necessary to ascertain the validity of architectural solutions.

The overall integration of all the DAQ, LVL2 trigger and DCS systems will proceed via the following global steps:

1. Integration and validation of DAQ/EF with LVL1 trigger function emulators (Section 7.8.1);
2. Integration and validation of the LVL2 trigger with LVL1 trigger function emulators (Section 7.8.2);
3. Integration and validation of the LVL2 trigger and DAQ/EF (Section 7.8.3);
4. Integration and validation of DCS and DAQ/EF (Section 7.8.4);
5. Integration and validation of T/DAQ one detector prototype (Section 7.8.5);
6. Integration and validation of T/DAQ data recording and offline prototypes (Section 7.8.6);
7. Integration and validation of T/DAQ with (an)other detector prototype(s) (Section 7.8.7).

Note – These steps might be done in a different order or in parallel.

7.8.1 DAQ/EF – LVL1 trigger integration

A minimum number of LVL1 functions are required in order to control the data acquisition and the successive trigger levels. Examples are: the LVL1 accept which initiates the read-out of the front-end electronics (RODs) into the ROBs, and the busy mechanism. A PMC-based prototype

of the TTC-receiver, implementing a subset of the final functionality, the TTC simple receiver (TTCsr) has been developed and is presently under test. Its functionality can be simulated during the DAQ/EF internal integration and so the integration of the TTCsr module is not required before Q2/99.

7.8.2 LVL2 trigger – LVL1 trigger integration

The LVL1 trigger sends RoI information to LVL2 via the RoI-builder in the LVL2 supervisor. In addition the LVL2 supervisor must be able to throttle the LVL1 trigger, in case of congestion in the supervisor or other parts of LVL2.

- Demonstrate event routing of LVL1 information to the LVL2 RoI-builder. (Q4/98)
- Propose a scheme for informing LVL1 of a potential overload in LVL2. (Q4/99)

7.8.3 LVL2 trigger – DAQ/EF integration

The integration of elements of the LVL2 prototype with corresponding elements of the DAQ/EF prototype, as required by the architecture validation for the technical proposal, proceeds on the following global steps:

- LVL2 – dataflow integration (Q4/99)
- LVL2 – back-end DAQ integration. Back-end DAQ is the master of the second level trigger during data taking and calibration.
    - definition of all functions which the LVL2 and back-end DAQ sub-systems will request of each other (Q3/99)
    - configuration databases (Q4/99)
        - LVL2 configuration databases
        - use of configuration databases by LVL2

7.8.4 DCS – DAQ/EF integration

In the final ATLAS online system, the DCS and DAQ will need to exchange information, data and signals for control and monitoring purposes while remaining independent sub-systems. Such exchanges concern several areas including databases, message reporting and logging systems, and run control.

The level of integration between the DCS and the DAQ, in particular the back-end, depends on the outcome of the initiative to have a common DCS project across all the LHC experiments. If the DCS project were to be organised in this way (a decision is expected and needed by the end of 1998) gateways for exchanges of information between the DCS and back-end DAQ would be required. Alternatively, if each LHC experiment were to define and implement its own DCS, then the overlap for ATLAS DCS and back-end DAQ would be larger and more software tools would be used in common.

The DCS group is also working on an interim stand-alone solution in order to allow the subdetector groups the development of their final front-end controls electronics. This system will be

used at test beams and at production line for acceptance tests. In situations where the stand-alone DCS and DAQ/EF-1 are working with the same detector, it will be necessary to implement a simple gateway between the DCS and back-end DAQ providing a minimal level of integration.

Independent of the outcome of the common DCS project, it is necessary to define in more detail the information exchanges that will take place between the DCS and back-end DAQ. It is planned to address this point in a series of meetings starting in the 3rd quarter of 1998.

7.8.5 DAQ/EF – first detector integration

The detector interface working group will determine the state of readiness of detector prototypes for integration with the data-acquisition prototype, based on the criteria outlined in 7.6.1. The choice of the first detector (Det_1) to be integrated will depend on the above and on other considerations such as availability of the detector experts and compatibility with the detectors' work plans. Although the detailed plan will have to be agreed and finalised with the relevant teams, we summarise here the expected global milestones.

- Joint discussions with detector experts (Q3 and Q4/98)
- Selection of Det_1 and joint development of detailed work-plan (Q1/99)
- Integration steps:
    - Det_1 – ROC integration (Q2/99)
        - ROD – ROB
        - Implementation of detector specific LDAQ functions
    - Det_1 – BEDAQ integration (Q2/99)
        - Configuration databases of Det_1
        - Information service and message reporting system integration of Det_1
        - Process manager and run control integration of Det_1.

7.8.6 DAQ/EF – data recording and offline integration

A work programme is presently being setup jointly with the ATLAS offline and physics working groups. Issues such as the use of Objectivity as the event database for test-beam data and the use of offline reconstruction and physics analysis code as the event filter program will be addressed. The time-scale will be specified by the joint work programme of the three groups involved. Priority will be given to those aspects likely to have implications on the DAQ/EF architecture.

7.8.7 DAQ/EF – other detectors integration

The necessity of integration of more than one detector for architecture validation will appear only after the integration of a single detector (Section 7.8.5).

If needed, the integration procedure would be the one of Section 7.8.5, although it is not excluded that a more limited integration is sufficient, provided it studies properly partitioning issues.

It must be stressed that the integration of detector prototypes with the LVL2, DAQ/EF and DCS prototypes planned and described here is for the purpose of architecture validation. The possible use of the LVL2, DAQ/EF and DCS prototypes for standard, production test-beam operations require the development of a work-plan to be addressed by the detector interface working group, and is outside the scope of this document.

7.9 References

7-1 Memorandum of understanding for collaboration in the construction of the ATLAS detector, RRB–D 98–44rev. (1998).

7-2 ATLAS trigger performance status report, CERN/LHCC 98–15 (1998).

7-3 The LVL2 pilot project, ATLAS internal note, DAQ–NO–118 (1998)

7-4 Trigger and DAQ interfaces with front-end systems: requirements document. http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/FRONTEND/fereq.ps

7-5 R. McLaren and O. Boyle, ATLAS read-out link data format, version 1.1. http://www.cern.ch/HSI/atlas/format/

7-6 Detector interface working group – summary document. http://atddoc.cern.ch/Atlas/DetfeIf/Detinfo.ps

8 Project organisation and management

8.1 Participating institutes

The institutes participating in the Data Acquisition and Event Filter, the Level-2 Trigger and the Detector Control Systems projects are listed below, together with the names of the physicists and senior engineers from each institute committed to the projects. Short names are given for each institute; for the official name and location see the opening pages of this document.

Alberta
R. Davis, P. Green, J. Pinfold

Ankara
Contact person: O. Cakir

Argonne
R. Blair, L. Price, E. May, J. Schlereth, G. Drake, J. Dawson

Bern
H.P. Beck, G. Lehmann, R. Mommsen, K. Pretzl

Bucharest
E. Badescu, A. Radu

CERN
G. Ambrosini, E. Barberio, R. Bock, J.A.C. Bogaerts, D. Burckhart, H. Burckhart, M. Caprini[1], M. Cobal, R. Dobinson, N. Ellis, A. Fernandes, D. Francis, F. Giacomini, B.I. Hallgren, R. Hauser, R. Heeley, F. Hogbe Nlend, C. Hörtnagl, M. Joos, R. Jones, L. Mapelli, B. Martin, R. McLaren, L. Monteiro, G. Mornacchi, M. Niculescu[1], J.O. Petersen, D. Prigent, A. Rios Alonso, J. Rochez, T. Shears, I. Soloviev[2], R. Spiwoks, S. Tapprogge, L. Tremblet, H.C. Van Der Bij, P. Werner, A. Wildish

Copenhagen
H. Bertelsen, M. Dam, J.R. Hansen, B. Rensch

Cracow
Z. Hajduk, W. Iwanski, K. Korcyl, J. Olszowska

Edinburgh
O. Boyle

Geneva
A. Clark, P. Demierre, I. Efthymiopoulos, L. Moneta

Genova
Contact person: P. Morettini

1. on leave of absence from Institute of Atomic Physics, Bucharest
2. on leave of absence from PNPI, St. Petersburg

Innsbruck
E. Kneringer, D. Kuhn, A. Nairz, D. Schweiger

Istanbul
Contact person: A. Mailov

JINR
I. Alexandrov, S. Baranov, A. Chasanov, V. Kotov, Z. Krumshtein, M. Micheev, V. Rumyantsev, K. Rybalchenko, V. Yamburenko

KEK
H. Fujii, A. Manabe, M. Nomachi, Y. Watase, Y. Yasu

Lecce
O. Palamara, S. Petrera

Liverpool
C. Anderson, P. Maley, J. Lokier, S. Haas

London RHBNC
M. Dobson, S. George, B.J. Green, N.A.H. Madsen, J.A. Strong

London UCL
P. Clarke, R. Cranfield, G. Crone, P. Sherwood

Mainz
L. Köpke

Manchester
R. Hughes-Jones, S. Kolya, R. Marshall, D. Mercer

Mannheim
O. Brosch, P. Dillinger, H. Högl, K. Kornmesser, A. Kugel, R. Lay, J. Ludvig, R. Männer, K.-H. Noffz, S. Rühl, M. Sessler, H. Simmler, H. Singpiel

Marseille
C.Bee, P.Y.Duval, F.Etienne, D.Laugier, C.Meessen, R.Nacash, Z.Qian, C.Rondot, D.Rousseau, F.Touchard

Michigan SU
M. Abolins, R. Brock, B. Pope, B. Gonzalez Pineiro, Y. Ermoline

Moscow SU
N.V. Nikitin, F.K. Rizatdinova, S.Yu. Sivoklokov, L.N. Smirnova

Nagasaki
Y. Nagasaka, Y.Tanaka

NIKHEF
H. Boterenbrood, R.J. Dankers, J.T. van Es, R.G.K. Hart, W.P.J. Heubers, P.P.M. Jansweijer, G.N.M. Kieft, N.M. Kruszynska, J.C. Vermeulen

Novosibirsk
Contact person: I. Tikhonov

Pavia
R. Ferrari, V. Vercesi

Petersburg NPI
V. Filimonov, V. Khomoutnikov, S. Kolos, I. Riabov

Portugal
A. Amorim, H. Wolters

Prague AS & CU
F. Hakl, M. Jirina, M. Smizanska

Protvino
Contact person: S. Kopikov

RAL
J.T. Baines, A. Belias, D.R. Botterill, R.W. Hatley, R.B. Middleton, F.J.Wickens

Rome I
A. Di Mattia, S. Falciano, C. Luci, L. Luminari, F. Marzano, G. Mirabelli, A. Nisati, E. Petrolo, S. Veneziano, L. Zanello

Saclay
A. Amadon, J. Bystricky, D. Calvet, J.M. Conte, J. Ernwein, O. Gachelin, T. Hansl-Kozanecka, R. Hubbard, M. Huet, P. Le Du, I. Mandjavidze, M. Mur, B. Thooris

Sheffield
C.N. Booth, S.J. Wheeler-Ellis

Tel Aviv
H. Abramowicz, E. Etzion

UC Irvine
A.J. Lankford, M. Schernau

Udine
F. Scuri

Weizmann
E. Duchovni, E. Gross, D. Lellouch, L. Levinson, G. Mikenberg, K. Nagai

Wisconsin
S. Gonzalez, D. Jared, Y. Pan, S. Qian, S.L. Wu, H. Zobernig

8.2 Responsibilities and work organisation

For planning purposes, the projects are divided into a number of subsystems. The institutes and funding agencies responsible for each of these are listed in the ATLAS Memorandum of Understanding [8-1]. The list of responsibilities is given in Table 8-1.

Table 8-1 Sharing of responsibilities for DAQ-EF, LVL2 and DCS project.

Item	Funding agencies	Institutes
DAQ/EF		
DAQ readout	CERN, Denmark, France, Italy, Netherlands, Switzerland, Turkey, UK, US	Ankara, Argonne, Bern, CERN, Copenhagen, Edinburgh, Geneva, Istanbul, Liverpool, London RHBNC, London UCL, Manchester, Michigan SU, NIKHEF, Pavia, RAL, Saclay, Sheffield, UC Irvine, Udine, Wisconsin
Event builder	CERN, France, Japan, Switzerland	Bern, CERN, Geneva, KEK, Nagasaki, Saclay
Event filter and back-end DAQ	Austria, CERN, Germany, Italy, Japan, JINR, Netherlands, Portugal, Switzerland, Turkey	Alberta, Ankara, Bern, CERN, Geneva, Innsbruck, Istanbul, JINR, KEK, Lisbon, Mainz, Marseille, Nagasaki, NIKHEF, Pavia, Udine
LVL2		
Calorimeter trigger	France, Germany, US	Argonne, Mannheim, Michigan SU, Saclay
Muon trigger	CERN, Germany, Israel, Italy	CERN, Genova, Lecce, Mannheim, Rome 1, Tel Aviv, Weizmann
tracking trigger	CERN, Czech Republic, Denmark, Germany, Netherlands, Poland, UK, US	CERN, Copenhagen, Cracow, London RHBNC, London UCL, Manchester, Mannheim, NIKHEF, Prague AS & CU, RAL, UC Irvine, Wisconsin
global trigger	CERN, France, Italy, UK, US	Argonne, CERN, Genova, Lecce, Liverpool, Manchester, Michigan SU, RAL, Rome I, Saclay
Supervisor and RoI builder	US	Argonne, Michigan SU
DCS		
Detector control system	CERN, Netherlands, Russia	CERN, NIKHEF, Petersburg NPI

8.3 Management organisation

The overall trigger / DAQ project of ATLAS, covering the LVL1 and LVL2 triggers, the data acquisition and event filter, and the detector control system, is organised following the normal ATLAS rules [8-2] [8-3]. A steering group (see Table 8-2), with members representing different areas of the project, is the executive body responsible for managing the project as a whole. An institutes board, with one voting representative per institute, is responsible for deciding policy and for matters of resources. Technical and scientific matters are discussed in working-group meetings for different areas of the project and in more specialised *ad hoc* working meetings. Responsibility for leading the project is currently shared between two coordinators, N. Ellis responsible for the trigger, and L. Mapelli responsible for the data acquisition and event filter. Steering group meetings are chaired by one or other of the coordinators. The institutes board is

presently organised by co-chair-people, M. Abolins and J.R. Hansen, who are *ex-officio* members of the steering group.

Table 8-2 Present composition (June 1998) of the trigger/DAQ steering group.

Activity	Members(s)
DAQ coordinator	L. Mapelli
Trigger coordinator	N. Ellis
LVL1 calorimeter trigger	E. Eisenhandler
LVL1 muon trigger	E. Petrolo
LVL2	S. Falciano, P. Le Du, F. Wickens
DAQ - dataflow	G. Mornacchi
DAQ - back-end	R. Jones
Event filter	F. Etienne
Detector control system	H. Burckhart
Trigger performance	T. Hansl-Kozanecka
FE electronics	Ph. Farthouat

8.4 Schedule and milestones

The proposed schedule for the projects is given in Table 8-3 and the overall milestones in Table 8-4.

The shorter term programme of activities and associated milestones are given in Chapter 7.

8.5 Cost and resources

The resource provision by funding agency is given in Table 8-5. The current cost estimate of each sub-project and the fractional funding by source are set out in Table 8-6. The delivery date for all subsystems is December 2003 except for the detector control system which is due in January 2003.

8.6 LVL2 pilot project organization

The main objectives of the pilot project are to integrate all of the activities connected to the LVL2 into a single programme and to produce results required for the technical proposal.

The LVL2 community is widely distributed around the world and it is sometimes difficult, especially for software tasks, to get a coherent, strong team. In order to solve this issue, a core of seven structured activities has been selected, covering the overall system. In order to avoid a

Table 8-3 Overall schedule of activities.

Item	Completion date
DAQ/EF	
Construction of the DAQ/EF-1 prototype showing a fully functional system from readout link to data recording	December 1998
Assessment of the DAQ/EF-1 prototype	December 1999
Integration studies of DAQ/EF and LVL2 trigger	December 2000
Detailed specification of the final system	December 2001
LVL2	
Pilot Project to study trigger architecture and technology options	December 1999
Integration studies of LVL2 with DAQ/EF	December 2000
Detailed specification of the final system	December 2001
DCS	
Detailed specification of the system and selection of hardware and software components	January 2000

Table 8-4 Overall future milestones.

Item	Date due
Technical proposal on DAQ and high level triggers	December 1999
Technical design report on DAQ and high level triggers	June 2001
Standalone DCS system completed	January 2003
DAQ and high level trigger construction completed	December 2003
Integration of detectors with the trigger, DAQ and DCS systems completed	December 2004

too-heavy bureaucracy and keep some flexibility at this stage of the project, the local management of each activity is delegated to one or two persons (convenors) active in the field.

The main tasks of each convenor are to call meetings to organize the work, organize web pages, collect and make available documentation and information, and report to the community and the LVL2 coordinators.

The role of the coordinators is to check that the overall programme is coherent and that each activity achieves the milestones associated with the individual work plan. Figure 8-1 gives an overview of these activities.

Given the wide-spread community, it is proposed to be very flexible in the organisation of meetings. Four kinds of meeting are foreseen: local meetings between individuals, groups or laboratories; dedicated activity meetings called by the convenors of one or more activity; ATLAS LVL2 plenary meetings, organized by the coordinators; and workshops or special meetings of the trigger/DAQ community. For the smaller, more frequent meetings, the use of telephone or video conferences is encouraged.

Table 8-5 Current value of deliverables in 1995 kCHF.

Agency	LVL2 Trigger	DAQ/EF	DCS
Austria		300	
Canada		a	
Czech Republic	40		
Denmark	500	500	
France (CEA & IN2P3)	1350	2550	
Germany BMBF	1025	770	
Israel	60		
Italy	900	2400	
Japan		1500	
Netherlands	325	325	200
Poland	150		
Portugal		300	
Romania			
Russia and JINR		100	
Switzerland		4000	
Turkey		150	
UK	1150	1070	
US (DoE and NSF)	3290	675	
CERN	1500	4400	1600

a. under discussion

8.7 DAQ/EF-1 organization.

The organisation and management structure of the DAQ/EF-1 project is very tightly mapped onto the structure of the project itself. As shown in Figure 8-2, the DAQ/EF-1 project, led by the project leader, is structured into four main areas. The work in each area, or subsystem, is organized by a coordinator who leads the corresponding working group. The four working groups are:

1. The detector interface working group, with one representative from each ATLAS system. This working group has no underlying substructure.
2. The dataflow working group, which is further structured into:
    - The DAQ unit subgroup.
    - The local DAQ subgroup.

Table 8-6 Cost estimate by sub-project in 1995 kCHF and fractional funding by source.

Item	Costing	Funding sources
DAQ/EF		
DAQ readout	9275	CERN (21.5%), Denmark (5.4%), Franc (20.5%), Italy (11.9%), Netherlands (2.7%), Switzerland (18.3%), Turkey (1.3%), UK (11.5%), U (7.3%)
Event builder	3710	CERN (27%), France (17.5%), Japan (27%), Switzerland (32.3%)
Event filter and back-end DAQ	5675	Austria (5.3%), CERN (24.6%), France, Germany (13.6%), Italy (22.9%), Japan (8.8%), JINR (1.8%), Netherlands (1.3%), Portugal (5.3%), Switzerland (19.4%), Turkey (0.5%)
LVL2		
Calorimeter trigger	1870	France (45.5%), Germany (10.7%), US (45.5%)
Muon trigger	1290	CERN (15.5%), Germany (27.9%), Israel (4.6%), Italy (52.7%)
Tracking trigger	4595	CERN (21.7%), Czech Republic (0.9%), Denmark (10.9%), Germany (10.1%), Netherlands (7.1%), Poland (3.2%), UK (18.5%), US (28.4%)
Global trigger	1590	CERN (18.9%), France (31.4%), Italy 13.8%), UK 18.9%), U (18.2%)
Supervisor and RoI builder	845	US (100%)
DCS		
Detector control system	1800	CERN (89%), Netherlands (11%)

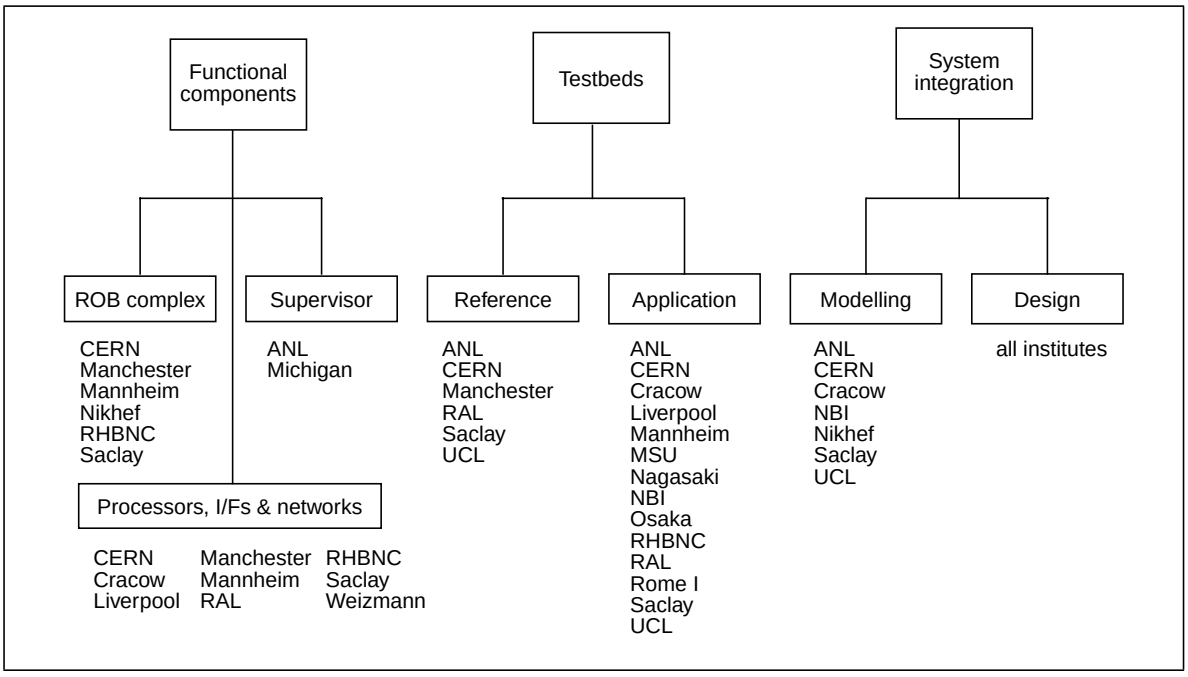


Figure 8-1 Overview of activities for the LVL2 pilot project.

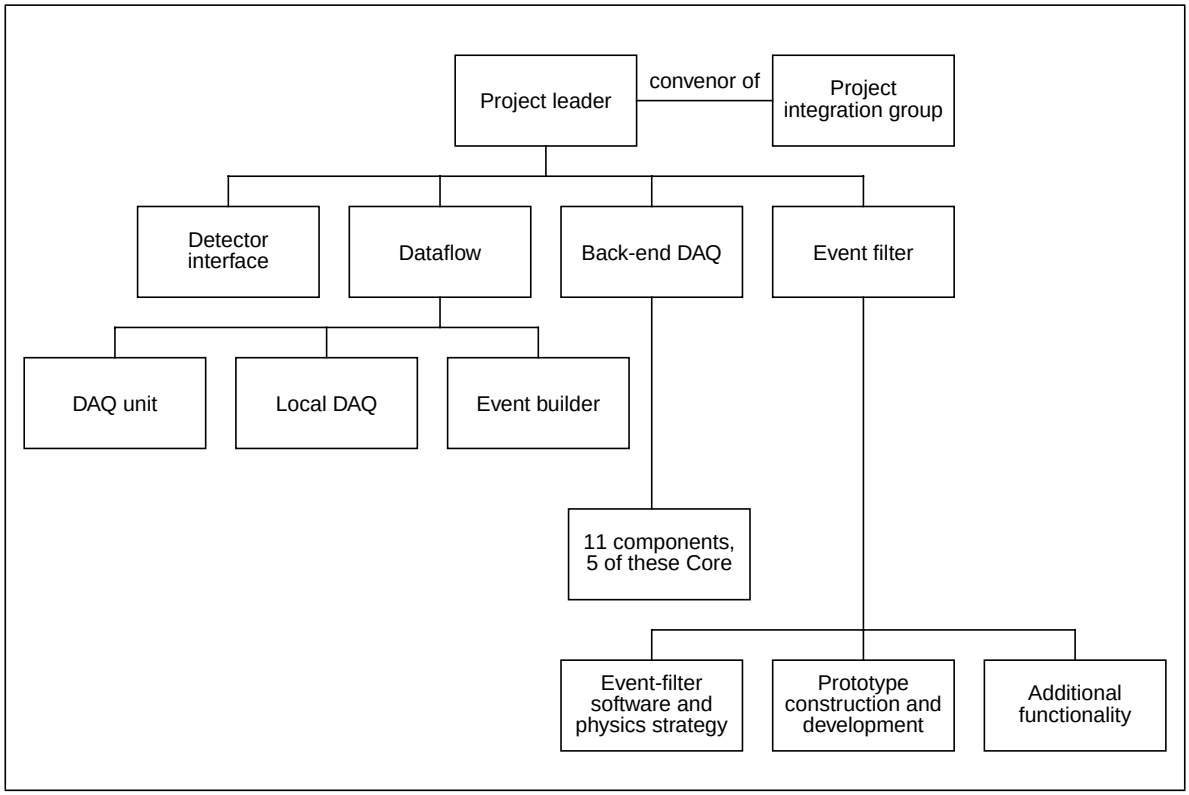


Figure 8-2 Organization and management of the DAQ/EF-1 project.

- The event builder subgroup.

Each of the dataflow subgroups has a coordinator who reports to the overall dataflow co-ordinator.

3. The back-end DAQ working group. This is further structured in a number of subgroups corresponding to the back-end DAQ components. Due to a lack of resources, not all the components can be developed in parallel. The present number of sub-groups is eight, the five core components and three of the detector integration components. The development of the other components will be organised as soon as the core ones are established.

   Each of the back-end DAQ sub-groups has a coordinator who reports to the overall back-end DAQ coordinator.

4. The event filter working group. This is further structured into:

   - The event filter software and physics strategy subgroup, which also maintains a tight link to the ATLAS physics and trigger performance working groups.
   - The event filter prototypes subgroup.
   - The subgroup addressing additional functionality, which has a big overlap with the preceding subgroups, but extends, somewhat, their domains.

A fifth group, the project integration group, is composed by all the coordinators and a number of other individuals in the project. This group is coordinated directly by the DAQ/EF-1 project leader and assures the smooth integration of the four subsystems into one coherent system.

8.8 References

8-1 Memorandum of understanding for collaboration in the construction of the ATLAS detector, RRB-D98-44, 28 April 1998.

8-2 ATLAS Organization, ATLAS internal note, GEN–No–009, 16 September 1994.

8-3 ATLAS System Organization, ATLAS internal note, GEN–No–016, 29 November 1996.

A Appendix: definitions, acronyms and abbreviations

Definitions given here are not formal but attempt to give a brief insight to help comprehension.

AAL
ATM adaption layer.

ADC
Analogue to digital convertor.

ANSI
American national standards institute.

API
Application programming interface.

ASIC
Application-specific integrated circuit. A custom-made integrated circuit.

ATLAS
A toroidal LHC Apparatus.

ATLFAST
ATLAS software package for fast particle-level simulation.

ATM
Asynchronous Transfer Mode

ATRIG
ATLAS software package for trigger simulation.

Barrel
The central-rapidity region of either the muon spectrometer, the electromagnetic calorimeter or the hadronic calorimeter.

BEDAQ
Back-end data acquisition.

Calorimeter cell
The smallest unit of calorimeter information to be read out and digitized.

CAD
Computer-aided design

CAN
Control area network. A field bus for controlling and monitoring, used in the Detector Control System.

CASE
Computer-aided software engineering.

CBR
constant bit rate.

Central Trigger Processor (CTP)
The part of the Level-1 Trigger System which combines results from the Level-1 Calorimeter and Muon Triggers to make the global (yes / no) Level-1 Trigger decision for each bunch crossing.

CERN
European laboratory for particle physics (Laboratorie européen de physique des particules).

CHSM
Concurrent hierarchical state machine.

CISC
Complex instruction set computer.

CMS
Compact muon solenoid.

Communication benchmarks
A standard set of measurements used to quantify the performance of communication links.

CPU
Central processing unit.

CRC
Cyclic redundancy check.

CSR
Control and status register.

CTP
See Central Trigger Processor.

DAQ
See Data Acquisition System.

DAQ/EF-1
A functionally complete pre-prototype DAQ and event filter system

Data Acquisition System (DAQ)
System responsible for the assembly and permanent storage of events accepted by all three levels of the trigger system (Level-1, Level-2, Event Filter) and data generated within the trigger systems.

DCS
See Detector Control System.

Detector
The complete ATLAS detector or one of its constituent detector systems.

Detector Control System (DCS)
The system which monitors and controls physical parameters of the sub-systems of the experiment, such as gas pressure, flow-rate, high voltage settings, low-voltage power supplies, temperatures, leakage currents, etc.

DICE
ATLAS software simulation

DMA
Direct memory access: a mode of fast data transfer.

DRAM
Dynamic random access memory.

DSP
Digital signal processor.

DVSM
Disitributed virtual shared memory.

ECP
Elan communication processor.

EF
See Event Filter.

Electron/photon cluster (e.m. cluster)
The areas in η–ϕ which are summed in the e.m. calorimeters in order to compare their transverse energy with electron / photon trigger thresholds. The elements used in the summing are e.m. Trigger Towers.

Emulation
Simulation of one system on another. In this document it refers to simulation of ATLAS trigger traffic on a large switching-network testbed.

EMI
Electromagnetic interference.

ENABLE++
An FPGA-based processor system.

Endcap
The high-η part of the muon spectrometer and calorimeter systems. The endcap covers the two ends of the cylindrical 'barrel' central region of ATLAS.

E.M.
Electromagnetic. (Can refer to either the calorimeters — e.m. vs. hadronic — or to the e.m. trigger, aimed at detecting electrons and photons.)

EPICS
Experimental Physics and Industrial Control System.

Event
The data resulting from a particular bunch crossing. At high luminosity, this could contain data from several physics processes.

Event Filter (formerly level-3 trigger, or LVL3)
The third level of event selection, responsible for reducing the trigger rate and hence the data rate to a value acceptable for permanent storage, roughly 100 Mbyte / s. A processor system which receives events from the Event Builder, these events having been selected by L2_accept or L2_request signals. The Event Filter carries out further processing and analysis and, if accepted, sends the event to data storage for offline analysis.

FC
Fiberchannel.

FCAL
Liquid-argon forward calorimeter.

FCT
Flow-control token.

FDDI
Fiber distributed data interface.

FE
See Front End.

Feature extraction
The process for turning basic detector data into physics-like parameters (e.g. hits in a tracking detector into track parameters).

FEX
Feature extraction or feature extractor (the processor used for feature extraction).

FIFO
First-in first-out. A type of buffer memory.

FLOPS
floating-point operations per second.

Forward
The forward trigger refers to the inner (low-radius) part of the TGC trigger system.

FPGA
Field-programmable gate array.

Front End
Shorthand for Front-End Electronics.

Front-end electronics
The detector sub-systems which generate and send trigger data to the Level-1 Trigger System and event data to their RODs for transmission to the data acquisition system.

GIPS
Giga instructions per second.

GEANT
A general Monte Carlo simulation package for describing detector geometry and tracking particles through detector material. Used to simulate the response of the ATLAS detector.

GNU
GNU is not UNIX.

GPMIMD
General purpose, multiple instruction, multiple data.

GUI
Graphical user interface.

Hadron/tau cluster
The areas in η–ϕ which are summed in order to compare their transverse energy with hadron / tau trigger thresholds. The elements used in the summing are Trigger Towers.

HEP
High energy physics.

High level trigger (HLT)
All selection levels save LVL1.

HIPPI
High-performance parallel interface.

HLT
See high level trigger.

HPCN
High performance computing node

h/w
Hardware.

ID
Inner Detector.

I/F
Interface.

IEEE
Institute of Electrical and Electronics Engineers (USA).

IP
Internet protocol.

IPC
Internet-process communication.

ISO
International standards organization.

JCOP
Joint project controls.

JTAG
Joint test action group

LAN
Local Area Network.

LAr
Liquid argon — refers to the ATLAS calorimeters, all of which except the Tile Calorimeter use liquid argon as a sampling medium.

LCS
Local Control Station.

LED
Light-emitting diode.

LEP
Large electron positron collider.

Level-1 Accept
A signal generated by CTP when an event has met the level-1 trigger criteria, i.e. is a level-1 trigger. It is distributed by the TTC system.

Level-1 calorimeter trigger
The part of the Level-1 Trigger System whose calculations are based on information from the ATLAS calorimeters. Trigger objects are e.m. showers, single hadrons (τ), jets, missing E_T, and total E_T.

Level-1 muon trigger
The part of the Level-1 Trigger System whose calculations are based on information from the ATLAS muon detectors. Trigger objects are high-p_T muons.

Level-1 trigger system (LVL1)
The first level of event selection, responsible for reducing the event rate from the bunch-crossing rate of 40 MHz to no more than 75 kHz averaged over short time periods (e.g. 10 ms), using a fast hardware processor. For accepted events, it issues Level-1 Accept to the front-end electronics and RoI_message to the Level-2 Trigger. The system consists of the Level-1 Calorimeter Trigger, the Level-1 Muon Trigger, and the Central Trigger Processor.

Level-2 trigger system (LVL2)
The second level of event selection, responsible for reducing the trigger rate from about 75 kHz(upgradable to 100 kHz)to a rate acceptable to the Event Filter, 1–5 kHz.It requests and receives RoI data from the ROBs and, after analysing it, sends an L2_accept or L2_reject signal for the event to the ROBs.

Level-3 trigger system
See Event Filter.

LHC
Large hadron collider.

LMB
Local Monitor Box.

LSB
Least-significant bit.

LUT
Lookup table.

LVI
Link VME interface.

LVL1
See Level-1 Trigger System.

LVL2
See Level-2 Trigger System.

LVL3
See Event Filter.

LynxOS
A real-time UNIX-like operating system from Lynx Real-Time Systems, Inc.

MACRAME
The large switching network testbed used for emulation studies.

MDT
Monitored drift tube.

Micro-ENABLE
An FPGA-based co-processor.

MIMD
Multiple instruction streams, multiple data streams.

MIPS
Multiple instructions per second.

MMU
Memory management unit.

Model
1. Computer based model of electronic system usually using discrete event simulation techniques.
2. Model A, B and C – demonstrator architecture models.

Modelling
Development of computer models.

MPI
Message passing interface.

MPP
Massively parallel processor.

MSB
Most-significant bit.

MSGC
Micro-strip gas counters.

MUCTPI
Interface between LVL1 muon-trigger processor and LVL1 central trigger processor.

MUX
Multiplexer.

NDL
Network description language.

NFS
Network file system.

NORMA
NO remote memory access.

NOW
Network of workstations.

NUMA
Non-uniform memory-access.

Object building
Assembly of features (from feature extraction) from different detectors, identification of particle type (or jet), and computation of attributes (energy, direction, position etc.).

ODBMS
Object-oriented database management system.

OKS
Object Kernel Support.

OO
Object oriented. A methodology for writing software.

OSI
Open systems interconnect.

PC
Personal Computer.

PCI
Peripheral component interconnect. An industry-standard bus system used mainly in personal computers.

PLC
Programmable Logic Controller.

POSIX
Portable operating system interface.

PS
Proton synchrotron.

PS-Pack
Patch-panel and slave-board package of the endcap muon trigger.

PVC
Permanent virtual machines.

PYTHIA
Monte Carlo program used to generate simulated proton–proton interactions for various physics processes.

QoS
Quality of service.

R&D
Research and development.

RAM
Random access memory.

Readout Buffer (ROB)
A standard module which receives data from the RODs via standard Readout Links, passes on request a subset of the data to the level-2 trigger, and buffers the data until a Level-2 Trigger decision has been reached whereupon, for accepted events, it transmits the data to the Event Filter.

Readout Driver (ROD)
The last element in the readout chain that is still considered part of the front-end electronics. This module collects one or more data streams from detector elements and merges them into a single stream which is fed via a standard Readout Link into a ROB.

Readout link (ROL)
The ATLAS-standard data-transmission link between a ROD and a ROB.

Region of Interest (RoI)
A geographical region of the experiment, limited in η and ϕ, identified by the Level-1 Trigger System as containing candidates for Level-2 Trigger objects requiring further computation. Their data will be further analysed by the Level-2 Trigger System to decide if the event is to be processed further. In the case of B-physics triggers at low luminosity, some RoIs may be defined internally within the Level-2 Trigger.

Resistive Plate Chamber (RPC)
Muon detector used for triggering in the barrel region of ATLAS.

RISC
Reduced instruction set computer.

ROB
See Readout Buffer.

ROC
Readout crate.

ROD
See Readout Driver.

RoI
See Region of Interest.

RoI builder
A unit, inside the Level-1 Trigger System, that collects and formats level-1 RoI information for use by the Level-2 Trigger.

ROL
Readout link.

ROM
Read-only memory.

RPC
Remote procedure call, or...

RPC
See Resistive Plate Chamber.

SAN
system area network.

SCI
Scalable Coherent Interface.

SCT
Semi-conductor tracking detector.

SE
Software Engineering.

Sequential processing
Processing information connected with an event, one feature at a time.

Sequential selection
Selecting to continue analysing or rejecting an event at one of a number of stages.

SFI
Switch-to-farm interface.

SHRIMP
Scalable high-performance really-inexpensive multi-processor.

SPARC
Scalable processor architecture.

SPEC
standard performance evaluation cooperative.

SPS
Super proton synchrotron.

SRAM
Static random access memory.

Sub-detector
A part of a larger detector system.

Sub-system
A part of a larger system.

SUSY
Supersymmetry.

SVC
Switched virtual circuit.

SVR4
UNIX system V release 4.

s/w
Software.

System
A complete system (may be part of something larger).

TCP
Transition control protocol.

TDM
Time division multiplexer.

TDR
Technical design report.

Testbed (Pilot project)
A set of processors interconnected by a network for the assessment of components of a full system.

Thin Gap Chamber (TGC)
Muon detector system used for triggering in the endcap region of ATLAS.

Tile calorimeter (TileCal)
Hadronic barrel calorimeter, using scintillating tiles as active medium.

Timing, Trigger and Control (TTC)
The standard system which provides and distributes trigger signals (e.g. Level-1 Accept), timing signals (e.g. system clock time stamps), and control signals to the various sub-systems of the experiment.

TP
Technical proposal.

TPR
Technical progress report.

TRD
Transition radiation detector.

Trigger
A decision made by a trigger system (LVL1 or LVL2) that a particular event is of potential interest and should be retained at least until the next stage of selection. At LVL1 and LVL2, this decision is based on a subset of data from the event.

Trigger chamber
A generic term for RPCs and TGCs.

Trigger menus
The set of trigger conditions in use at any particular time. They specify a list of items, each with threshold(s) and multiplicity, and the logic to be applied to them. The level-1 trigger menu is implemented in the CTP, and is a set of logical combinations of results from the Level-1 Calorimeter and Muon Triggers.

TRT
Transition-radiation tracking detector.

UBR
Unspecified bit rate.

UMA
Uniform memory access.

UMP
Universal message passing.

UNIX
Unixplexed information and computing system.

URD
User Requirements Document. Based (at least loosely) on software standard PSS-05 of the European Space Agency.

URL
Universal Resource Locator (address used by WWW).

VCI
Virtual channel identifier.

VHDL
VHSIC hardware description language.

VHSIC
Very-high-speed integrated circuit.

VI
Virtual interface.

VME; VMEbus
Versa-Module Euro. A crate backplane bus system.

VPI
Virtual path identifier.

WAN
Wide area network.

ZEBRA
A dynamic memory-management package for FORTRAN programs. Used in the ATLAS offline software.

Zero suppression
Compression of data by removal of values equal to zero.

This document has been prepared with Release 5.5 of the Adobe FrameMaker® Technical Publishing System using the Technical Design Report template prepared by Mario Ruggier of the Information and Programming Techniques Group, ECP Division, CERN, according to requirements from the ATLAS collaboration.

To facilitate multiple author editing and electronic distribution of documents, only widely available fonts have been used. The principal ones are:

Running text:	Palatino 10.5 point on 13 point line spacing
Chapter headings:	Helvetica Bold 18 point
2nd, 3rd and 4th level headings:	Helvetica Bold 14, 12 and 10 point respectively
Figure and table captions:	Helvetica 9 point