
Options for the ROB Complex

Editors:

R.Cranfield (UCL) / J.Vermeulen (NIKHEF)

ATLAS LVL2 Pilot Project ROB COMPLEX
MASTER WORKING DOCUMENT

Last update: 3 April 2000

ABSTRACT:

This is the Master Working Document of the ROB Complex sub-project of the ATLAS LVL2 Pilot Project. It describes the options for the ROB Complex, a generalised ATLAS readout buffering element, from the LVL2 point of view. In its canonical form a ROB Complex handles the buffering of event data from a group of detector input links and provides selections of this data to the LVL2 Trigger and Event Builder as requested. This document comprises both a record of the detailed ROB Complex studies conducted during the Pilot Project and a checklist of the requirements and issues relevant to the future development of ATLAS readout buffering. The three main sections contain a comprehensive status report of design issues, scenarios for a full system, and performance measurements of prototype designs. Earlier sections describe the organisation of the sub-project and set the context of the ROB Complex.

CONTENTS

1.0	Document Introduction	5
2.0	Project Motivation	6
2.1	original Project Goals	6
2.2	Why is the ROB studied by LVL2?	6
3.0	Project Organisation.	7
3.1	Communication mechanisms	7
3.1.1	Meetings	7
3.1.2	Mailing list	7
3.1.3	Web pages:	7
3.1.4	Convenors	7
3.1.5	Main participants	7
4.0	Original Workplan Task List	8
4.0.1	Task Name: Provide input for ROB User Requirements Document	8
4.0.2	Task Name: Design studies, including performance measurements	8
4.0.3	Task Name: Production and support of buffer prototypes	9
4.0.4	Task Name: Production of APIs relevant to the ROB complex	9
4.0.5	Task Name: Paper studies of scenarios for a complete system	10
5.0	ROB Complex Description	11
5.1	Context Diagram	11
5.2	Functional Description & Terminology	11
5.3	Functional Components	12
6.0	Design Checklists	14
6.1	Technology	14
6.1.1	Technology prototyping activities:	14
6.1.2	Technology watch items:	15
7.0	Design Issues	16
7.1	Production planning	16
7.1.1	Timescale	16
7.1.2	Development after 1999	16
7.2	ATLAS Strategy	16
7.2.1	LVL1 ID	16
7.2.2	TTC interaction	17
7.2.3	LVL1 trigger system interaction	18
7.2.4	ROD-related timeouts	19
7.2.5	Standalone running	19
7.3	ROD Constraints	19
7.3.1	Event order	19
7.3.2	ROD latency	19
7.3.3	XOFF timing	19
7.4	Detector requirements	20
7.4.1	ROB-level monitoring	20
7.5	Data Profile	21
7.5.1	Event fragment sizes and ordering	21
7.5.2	Buffer sizes	22

7.5.3	Deadtime	23
7.5.4	Latencies	23
7.6	LVL2 Strategy	23
7.6.1	Pre-processing and data-selection	23
7.6.2	Grouping of ROBs	26
7.6.3	RoI Request distribution	27
7.6.4	LVL2 decision record	27
7.7	Data Format	27
7.7.1	Message types	27
7.7.2	Event fragment format	28
7.7.3	Format for other messages	31
7.7.4	Byte order	31
7.8	Operational framework	31
7.8.1	Run Control	31
7.8.2	Errors and error handling	33
7.8.3	Fault handling	36
7.8.4	Status monitoring and reporting	36
7.9	Implementation	37
7.9.1	Hardware architecture	37
7.9.2	Data volumes & request rates	39
7.9.3	Physical organisation	41
7.9.4	Front-end connections	42
7.9.5	Use of daughter boards	43
7.9.6	Level of integration	43
7.9.7	Power	44
7.9.8	Microprocessors & FPGAs	44
7.9.9	Memory	49
7.9.10	Buffer Management	51
7.9.11	Internal communication	51
7.9.12	Network connection	52
7.9.13	Commercial availability	52
7.9.14	Cost	53
7.9.15	Reliability	53
7.9.16	Maintainability	54
7.10	Software	55
7.10.1	Software design analysis	55
7.10.2	Application Program Interfaces (APIs)	55
7.10.3	Prototype software	56
7.11	Testing	59
7.11.1	Tests during and directly after production	59
7.11.2	Offline testing	60
7.11.3	Online testing	61
8.0	Scenarios for a full system	62
8.1	Modular ROB Complex Scenario (Saclay)	62
8.2	Point-to-point scenarios (NIKHEF)	64
8.2.1	Scenario I	64
8.2.2	Scenario II	67
8.3	FPGA-based scenario (Mannheim, Weizmann)	70
8.4	Active ROB Complex scenario (CERN, NIKHEF)	73
8.5	Simple-ROB scenario (UCL, RHUL, CERN)	75

9.0	Prototypes: description and performance	77
9.1	Mannheim & CERN Groups	77
9.2	NIKHEF Group	77
9.2.1	Paroli tests	78
9.3	Saclay Group	78
9.4	UCL/RHUL Group	79
9.5	Comparison of results	79
9.5.1	Conclusions from prototype studies	82
10.0	References	83

1.0 DOCUMENT INTRODUCTION

This document is the final version of the master working document for the ROB Complex subproject of the ATLAS Level2 Trigger Pilot Project.

It was created as an evolving repository of the current issues, results, decisions and conclusions concerning the design of the ROB Complex, accessible to all workers in the Project.

It is intended that this document will form a useful starting point for subsequent development of the ATLAS readout buffers, comprising both a record of the detailed studies conducted during the Pilot Project and a checklist of the requirements and design issues to be taken into consideration.

2.0 PROJECT MOTIVATION

2.1 ORIGINAL PROJECT GOALS

1. To produce documents describing the options for the ATLAS ROB.

These documents should provide input for the Trigger/DAQ Technical Proposal planned for the spring of 2000. They should draw conclusions as far as possible within the prevailing limits of available knowledge of ATLAS requirements and technological possibilities, as well as explaining the issues remaining to be resolved.

2. To produce input for the ROB User Requirements Document (URD).

It is assumed that there will be a revision of the ROB URD at the end of the period of the Pilot Project and it is a goal of the ROB Complex component of the Project to complete the LVL2 input for this URD. Although it may not be possible to detail all user requirements by this time, it is likely that the next revision will be the last one, as the development program passes into its system requirements phase. Hence every effort should be made to define user requirements at a meaningful level of detail.

3. To produce hardware for prototyping.

The development of ROB hardware serves two purposes: prototyping elements of the ROB Complex in their own right, and providing equipment for use in prototyping other LVL2 and DAQ components. In general the latter is likely to lag behind the former. It is assumed that, at the end of the Pilot Project when the main DAQ/LVL2 integration begins, there is still likely to be a requirement for ROB hardware for use in future prototyping test-beds.

The production of actual hardware prototypes helps, on the one hand, to highlight real-life design issues, but, if successful, the prototypes also serve as an existence proof for the feasibility of a ROB Complex.

2.2 WHY IS THE ROB STUDIED BY LVL2?

- Some of the trickiest ROB constraints and requirements are associated with RoIR handling in LVL2.
- ROBs must interface correctly to the LVL2 network.

but:

- The ROB is also a key DAQ component.
- The ROB complex overlaps with DAQ front-end components.

therefore:

- Integration between LVL2 and DAQ in matters of ROB design and development is crucial!

3.0 PROJECT ORGANISATION.

3.1 COMMUNICATION MECHANISMS

3.1.1 Meetings

1. Meetings of all interested parties, 2-4 per year
2. Informal discussion and brainstorming

3.1.2 Mailing list

3.1.3 Web pages:

1. Main evolving working document (maintained by editors)
2. Links to relevant documents (meetings minutes, background documents, detailed task reports, etc)
3. Comments and thoughts (probably an archive of emails to distribution list).

3.1.4 Convenors

Bob Cranfield (University College London)
Jos Vermeulen (NIKHEF, Amsterdam)

3.1.5 Main participants

Many people have contributed to the Pilot Project ROB Complex work, from both inside and outside the LVL2 collaboration. The following is a list of the main institutes who have participated in the production of the prototype designs and scenarios described in this document:

- CERN
- University of Mannheim
- NIKHEF
- Royal Holloway University of London
- DAPNIA CEA-Saclay
- University College London
- Weizmann Institute of Science

Contributions in the form of text input or detailed comment from the following individuals is acknowledged :

R.Bock, J.Bystricky, P.Clarke, R.Cranfield, N.Ellis, B.Green, M.Huet, P.Jansweijer, G.Kieft, A.Kugel, L.Levinson, E.Van der Bij, J.Vermeulen.

4.0 ORIGINAL WORKPLAN TASK LIST

4.0.1 Task Name: Provide input for ROB User Requirements Document

Task ID: ROBComplex-1-InputForURD

Organiser: R.Cranfield, J.Vermeulen

Need: Essential

Aims/Objectives: Further detailing of ROB Complex hardware requirements, of software requirements for LVL2 and for interfacing to the DAQ software.

Definition/Approach: Discussion with ATLAS colleagues, modelling to answer already prepared list of questions.

Duration: until end of Pilot Project (Feb 2000)

Deliverables: Input for revised ROB URD

Milestones:

internal: December 1998: Questions categorised (October 1998), 1st pass answers documented

external: End of Pilot Project (Feb 2000): Input ready for ROB URD revision

Dependencies: input from detector groups, DAQ groups & other LVL2 groups

Partners: CERN, MANCHESTER, MANNHEIM, NIKHEF, RHUL, SACLAY, UCL

Resources: manpower

Risks: low

Comment: This task is to provide updated and new material for what is assumed to be a last revision of the ROB URD at the end of the Pilot Project. The aim is to provide sufficient information to complete the URD so that it can form a proper basis for design work following the Pilot Project. As information is assembled it will also provide input for the task "ROBComplex-2-DesignStudies". Whilst some of the information sources are external to LVL2, some of the information will be derived from work in other areas of the LVL2 Pilot Project; this includes ROB Complex modelling, which will be carried out under the umbrella of the modelling sub-project.

4.0.2 Task Name: Design studies, including performance measurements

Task ID: ROBComplex-2-DesignStudies

Organiser: R.Cranfield, M.Huet, S.Kolya, A.Kugel, R.McLaren, J.Vermeulen

Need: Essential

Aims/Objectives: study critical issues through specific designs

Definition/Approach: prototype design of ROBIns, performance measurements, tests of possible use of PAROLI

Duration: until end of Pilot Project (Feb 2000)

Deliverables: Documents

Milestones:

internal: May - December 1998 : production ROBIN prototypes tests measurements PAROLI

June 1999 : results of performance measurements

October 1999 : software + first integration studies completed

external: Relevant sections of Master Document written. Feb 2000.

Dependencies: technical manpower in institutes
task ROBComplex-1-InputForURD

Partners: CERN, MANCHESTER, MANNHEIM, NIKHEF, RHUL, SACLAY,
UCL

Resources: internal to institutes

Risks: low

Comment: Because of the close interdependence between the functional components of a ROB Complex, it is planned to study design issues by pursuing different whole prototype designs at the participating institutes. Nevertheless, each design will have a main focus of study, and progress will be monitored against the key issues defined in the Master Working Document. The main deliverables will be sections of the Master Working Document, supported by additional documentation where necessary. Reporting points will occur naturally at meetings.

4.0.3 Task Name: Production and support of buffer prototypes

Task ID: ROBComplex-3-Prototypes

Organiser: R.Cranfield, A.Kugel, P.Le Du

Need: Driven by application testbed

Aims/Objectives: provision of prototypes for application testbed studies

Definition/Approach: complete existing development programmes

Duration: until December 1998

Deliverables: Prototype ROBIN hardware

Milestones:

internal: May - December 1998 : production ROBIN prototypes

external: December 1998 : prototypes and software ready

Dependencies: technical manpower in institutes

Partners: MANNHEIM, RHUL, SACLAY, UCL

Resources: internal to institutes

Risks: low

Comment: This task refers to the completion of existing programmes for the production of documented and supported prototype ROBIN boards to be used in both LVL2 and DAQ testbeds.

4.0.4 Task Name: Production of APIs relevant to the ROB complex

Task ID: ROBComplex-4-ROBComplexAPIs

Organiser: G.Crone, M.Huet

Need: design studies and application testbed
Aims/Objectives: provision of software for testbed studies
Definition/Approach: develop existing proposals
Duration: until February 1999
Deliverables: Specification
Milestones:
 internal: December 1998 : draft specification produced
 external: February 1999 : specification produced
Dependencies: Reference software
Partners: MANNHEIM, NIKHEF, SACLAY, UCL
Resources:
Risks: low

Comment : This task refers to the development of a (set of) common API(s) for use WITHIN the ROB Complex. It is clearly closely coupled with the development of an appropriate API within the Reference Software sub-project, but refers to a lower (more detailed) level. The same personnel are likely to be involved across both sub-projects. It is crucial that both developments take into account the analogous work for DAQ-1 and the need to move towards a common set of specifications for the eventual ROB design.

4.0.5 Task Name: Paper studies of scenarios for a complete system

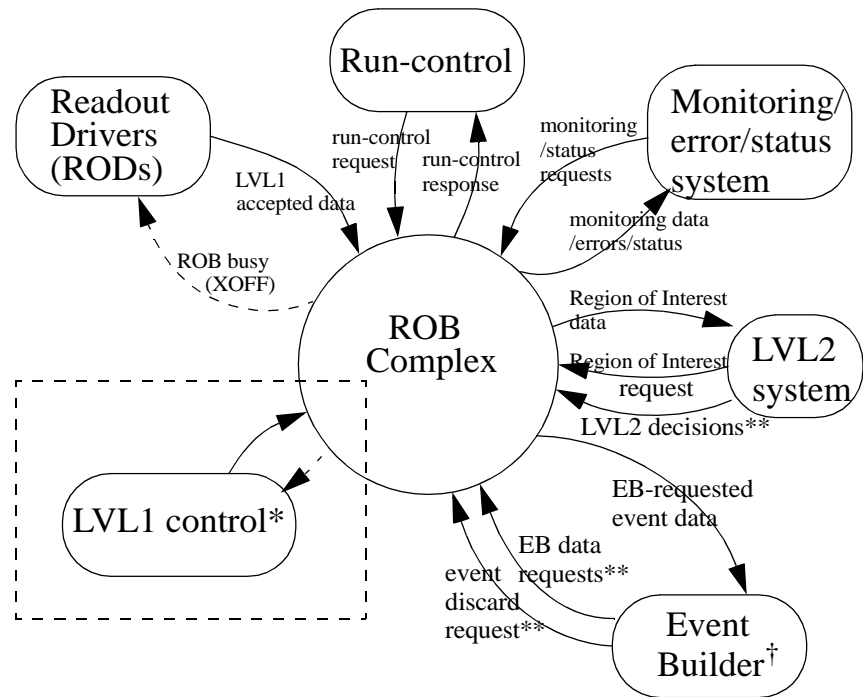
Task ID: ROBComplex-5-PaperStudies
Organiser: R.Cranfield, J.Vermeulen
Need: Essential
Aims/Objectives: to determine overall constraints and criteria
Definition/Approach:
Duration: until end of Pilot Project (Feb 2000)
Deliverables: Documents
Milestones:
 internal: October 1998: contributions available for discussion.
 external: February 2000: Section of Master Document written.
Dependencies: progress in design studies
Partners: MANCHESTER, MANNHEIM, NIKHEF, RHUL, SACLAY, UCL
Resources:
Risks: low

Comment: The aim of this task is to tackle some of the design issues by considering the realities of implementing a full system based on different design approaches. Relevant design issues include power, cost, space, and maintainability.

5.0 ROB COMPLEX DESCRIPTION

A description in terms of the Unified Modelling Language (UML) can be found in [1].

5.1 CONTEXT DIAGRAM



* The dashed box is meant to indicate that it is not clear that direct interaction with LVL1 trigger control is required. This issue is discussed later (Section 7.2.3, “LVL1 trigger system interaction,” on page 18).

** This exact manner in which a ROB Complex will be requested to discard event fragments or to send them on for further processing beyond LVL2 is not definitively decided. The dataflows shown are the ones most generally assumed.

[†] It is assumed that interaction with the next stage of the dataflow is via the Event Builder, which will pass selected events on for further processing downstream.

5.2 FUNCTIONAL DESCRIPTION & TERMINOLOGY

ROB Complex: A ROB Complex is some aggregation of readout buffers, together with appropriate interfacing and control components. Its basic purpose is that of a ROB (see definition below), but it potentially handles data from a number of readout links (ROLs) as a group. It thus allows for the possibility of concentrating data from these links through a smaller number of output connections (e.g. one), as well as performing data selection and compaction across a part of the detector larger than that associated with a single ROL.

For prototyping purposes a ROB Complex was originally conceived as being constructed from the following three types of component (though not all scenarios or developments preserve this demarcation):

- **ROBIn**: an input module, comprising an input connection and buffering for a single ROL,
- **ROB Controller**: the intelligent control unit for the ROB Complex,
- **ROBOut**: an interface to the LVL2 network and/or Event Builder.

ROB: The term “ROB” refers to a generic ATLAS ReadOut Buffer. Its basic role is to store detector data from events that have been selected at LVL1 whilst the events are inspected at LVL2 and/or are waiting to be requested by the Event Builder. The LVL2 processing methodology requires that only a subset of the data for each event be sent to the LVL2 processors, corresponding to “regions of interest” (RoI’s) indicated by LVL1. It may be desirable that some “pre-processing” be performed on data within the ROB before it is sent on to the LVL2 processors. It should also be possible to extract monitoring samples from the ROB data, and to control and inspect the ROB’s operating status.

Terminological note: the term ROB is often used in this document to refer in a general way to a readout buffer, without specifying precisely how buffers might be grouped.

ROS: This is an abbreviation (standing for ReadOut System) introduced for the purposes of the Technical Proposal to denote a generic component at a higher level of abstraction than generally assumed in the ROB Complex studies. In many cases in the current document it could be substituted for the terms ROB or ROB Complex.

Simple ROB: This term refers to a special case in which a ROB Complex is “collapsed” into a single module servicing just a single ROL. It contains an input buffer and management, a network interface and any necessary local intelligence. It is not excluded that there be an external module controlling a group of Simple ROBs, or that each Simple ROB has a second output (to the Event Builder), but the simplest form of Simple ROB is autonomous and has just two interfaces: a ROL input and a LVL2/DAQ network connection.

5.3 FUNCTIONAL COMPONENTS

The main functional components of a ROB Complex are:

1. ReadOut Link (ROL) interface

This component manages the data flow from the read-out driver (ROD) via the read-out link (ROL) protocol.

2. Buffer memory

Incoming event fragments are stored in a buffer memory.

3. Event fragment manager/selector

This provides access to event fragments in the buffer. The event identifier is used for this access.

4. Pre-processor

If required, data can be prepared before transmission to the level 2 processing. This preprocessing can include a part of the level 2 processing.

5. Interface to LVL2

This component receives requests from the level 2 processors and returns corresponding data. The level 2 decision is received by the same interface.

6. Interface to DAQ dataflow

This port transmits the complete event fragment to the Event Builder.

7. Local Data collector

A Rob Complex can include several event fragment buffers. Before delivery of data to the level 2 and the Event Builder, fragments from these buffers can be collected to build a common data block. The collection can be complete from all buffers, or selective according to the request.

8. Monitoring/error/status interface

This interface is connected to the on-line software for monitoring and error/status reporting.

9. Run-control interface

This interface receives commands from the on-line software for run-control.

10. Inter-component interfaces

These are the interfaces that handle transmission of messages between the internal components of a Rob Complex.

11. Test facility

6.0 DESIGN CHECKLISTS

As an initial aid to structuring the work for this project several checklists were drawn up. These comprised two technology-based lists and three lists of questions. The questions were used to generate a series of “design issues” which, grouped into several sections, now form the main content of this document, and are meant to themselves provide a checklist of all the points that need to be taken into account in the development and design of a ROB Complex. Points raised by the two technology checklists have also been incorporated into the design issues, but these two lists are still shown below as an aid to further cross-checking.

6.1 TECHNOLOGY

6.1.1 Technology prototyping activities:

1. Optical links

NIKHEF: possible use of Paroli; ROL multiplexing

CERN: optical version of S-Link

2. FPGAs

Mannheim: complete solution in FPGA

NIKHEF: logic for SHARC-based ROBIIn

Saclay: logic for i960-based ROBIIn

CERN: logic for SLIDAS

(DAQ-1: MFCC-based ROBIIn)

3. Processors

Saclay: i960JX in ROBIIn

RHUL/UCL: i960RD in ROBIIn

NIKHEF: SHARC in ROBIIn

(CERN DAQ-1 Group: PowerPC (MFCC ROBIIn, RIO2-based ROB))

CERN (Bock): SMP

4. PCI

Performance measurements (RHUL, NIKHEF, CERN?)

5. VME

CERN: DAQ-1 work on readout crate.

6. DSP links

RHUL: early C40-based buffer

NIKHEF: SHARC-based studies

7. ATM

Saclay: work on ATM-based ROBOOut.

8. Gigabit Ethernet

RHUL: work on Ethernet-based ROBOOut.

9. SCI & Blink

SCI work continues at RAL, Manchester & CERN. Early studies of Blink in the ROB Complex at Manchester -- currently in abeyance because SCI not favoured as a networking option.

6.1.2 Technology watch items:

1. Data input
2. Logic implementation
3. Memory
4. Memory management
5. Inter-module communication
6. Processing power
7. Network I/O

7.0 DESIGN ISSUES

This section attempts to identify and comment on all the points that should be taken into consideration when developing a design for an ATLAS ROB Complex, with the exception of some basic elements already specified in the ROB User Requirements Document. It refers to both requirements and implementation options, drawing on discussions and studies undertaken in the course of the project. One of the aims is to provide a written record of the state of current thinking to avoid repeating ourselves in the future.

7.1 PRODUCTION PLANNING

7.1.1 Timescale

Working back from a canonical date of 2005 for ATLAS startup, the following seems plausible:

2005: ATLAS startup

(Official milestone: Dec-2004: integration of detectors with trig/DAQ/DCS completed)

2004: integration/debugging of whole detector readout

(Official milestone: Dec-2003: DAQ/HLT construction completed)

2003: integration/debugging of trigger/DAQ + manufacture of production modules

2002: design/manufacture of production prototypes

Clearly the timetable differs depending on whether or not ROB components have to be custom-made; the above scheme implies that this decision should be taken early in 2002.

7.1.2 Development after 1999

Following the scheme above, the period 2000/2001 is available for refining the architectural options to the point where it is decided whether or not custom components are required.

It remains to be decided how the above refinement of options should proceed during 2000 and 2001. This needs to be addressed in the Technical Proposal. A relevant milestone is the Technical Design Report, due in Jun-2001.

7.2 ATLAS STRATEGY

7.2.1 LVL1 ID

In the ROB an event ID is needed for “indexing”. Where all the event data has to be transmitted (e.g. in test and calibration measurements) the ROB may not need the ID information, but in that case ID information will most likely be required in the event building phase.

The ROD should transmit events with the full 24-bit event id and full 12-bit beam crossing id for all types of events, including calibration, etc. events. According to [2] RODs must in all cases provide event fragments following a standard data format. This includes an L1ID in the header. The L1ID must be 'unique' within TTC/DAQ partition.

In the case of normal data events, the event ID is the LVL1 ID. Should events be sent to the ROB that are triggered locally in a calibration or test run, an event ID must be generated by the ROD which is unique for that run within each partition. Any calibration or test procedure that does involve the ROBs implies event building (perhaps across TTC boundaries) for that partition. In this case the event ID from the ROD is used to match fragments and so must be generated to allow consistent event building.

LVL1 ID uniqueness:

The period of uniqueness of the L1ID from the TTC system is limited by:

- 24 bit range,
- periodic resets (requested by detector groups to clear f/e errors),
- random resets (in response to error conditions).

The L1ID uniqueness cannot be guaranteed for long periods like the length of time that events might reside in ROB, or even in RODs. This implies that the L1ID must be extended or replaced. A possibility that has been discussed (P.Farthouat) is to extend the L1ID to 32 bits by adding the 8 most significant bits from a counter triggered by the ECR (Event Counter Reset) signal in the RODs.

Note that the period of uniqueness has relevance to the event indexing scheme used in buffer management (see Section 7.10.3.9, "Event management:," on page 58).

7.2.2 TTC interaction

Current thinking does not see a need for a TTC connection to the ROB for normal event data runs. Readout of data from the ROB is triggered by the appropriate requests from the LVL2 and DAQ systems (e.g. RoI requests, Data-Flow-Manager requests). Meanwhile, identification information (L1ID, BCID, trigger type) is incorporated by the ROD into each event fragment. TTC information provided to the ROB may be of marginal use for checking that the ROD does indeed send all the events, but seems unlikely to be worth the complexity and expense of the extra connections needed to supply it.

For partitioned calibration or test runs, on the other hand, different parts of ATLAS must be able to acquire data concurrently and independently. This is achieved by partitioning TTC and DAQ systems so that each DAQ partition maps onto one or more TTC zones that are covered by a single trigger/deadtime source. In partitioned calibration running LVL1 and LVL2 triggers play no role (the CTP cannot control more than one partition). We need, therefore, to decide how to initiate data acquisition in each partition.

The TTC system is a natural choice for trigger signals in partitioned mode (all detectors must be able to use the TTC system to drive their front-end electronics

and RODs, however different their triggering and deadtime logic). But the TTC trigger source can be used in several ways:

1. To provide signals direct to a module in each ROB Complex (as in the DAQ-1 ReadOut Crate TRG module). Note that if a ROB (on the surface) is to be in the same TTC partition as its partner ROD in USA15, then a long link to the TTCvi, which presumably sits near the ROD crate, is needed, but this is not thought to be a major problem.
2. To provide information in event fragment headers that can be extracted on one of the input links to each ROB Complex. Note that this implies that one ROBIn per complex takes on a special role.
3. To provide TTC signals to some higher-level component in DAQ (supervisor, Data Flow Manager?). This would need some multiplexing of TTC signals from multiple TTC partitions to the supervisory components of multiple DAQ partitions.

Whatever method is used, calibration running must be possible with LVL1/2 systems offline. This might argue against a TTC multiplexer tightly coupled to the LVL2 RoI Builder. On the other hand the distribution of destination information appears to be more straightforward with the third method. Note that the first two solutions are very similar.

See below for a caveat in sending non-ROD per-event information to the ROB.

7.2.3 LVL1 trigger system interaction

There is no clear need for the ROB to assert a “buffer almost full” condition to the Central Trigger Processor in order to slow down the event rate. It is assumed that each ROB Complex buffer will inform its associated ROD when its memory is almost full. This could be done by signalling an XOFF condition on a return path in the ReadOut Link. The ROD would use this signal to provide ‘back-pressure’ to eventually hold off detector readout when all the intervening buffering was effectively occupied. Such an XOFF facility is provided by S-link. The ROD can respond to this within a few clocks. If this ROD’s buffers begin to fill up it will assert BUSY to the Central Trigger. Directly asserting busy from the ROB would more likely cause instabilities than meaningful control since 1) the ROB input is so strongly derandomized by the ROD and lower levels of buffers, 2) the ROD has its own processing fluctuations, and 3) each ROB’s busy messages are uncorrelated to those of other ROBs.

One thing to consider: what type of per-event information arrives at the ROB from sources other than the ROD, and how is it buffered? If there is limited buffering for per-event information which comes from sources other than the ROD, then all of the additional event data buffering capacity of the ROD actually works against XON/XOFF being used as a safeguard against that sort of information being lost. For example, let’s say the CTP sends the ROB a piece of information, Y, each time a trigger is generated. If the Y buffer is getting full, XON/XOFF is a poor mechanism to protect it - the CTP might generate many more Y’s before the ROD issues ROD-BUSY. The TTC information mentioned above is an example of this.

For ROD builders: During system debugging it must be possible to identify the cause of a ROD’s BUSY. The XON/XOFF status from the ROB should be available both from the ROB and via the ROD to determine the ultimate source of a ROD’s BUSY signal. This is an important requirement for debugging the system.

Despite the lack of a clear need for the ROB to be able to assert Busy to the CTP, it is easy and inexpensive to add this capability to the BUSY system. ROB's should probably be built with a BUSY output, even if it is never used.

7.2.4 ROD-related timeouts (see also: Section 7.8.2.6, "Timeouts," on page 35)

The ROB timeouts specific to the ROD are those due to the failure of fragments to arrive within the maximum time interval after the associated RoI requests. If there are other per-event inputs to the ROB (e.g. from the CTP), then for each there is also an associated timeout. If the ROD fails to respond to an XOFF within a set time, an error should be reported. See also Section 7.8.2.6, "Timeouts".

The ROB URD requirement, UR-GBL-HIST, requires that the ROB keep a history of all signals and messages received. In addition to being reported, all timeouts must be entered into this history so they can be correlated to external events.

7.2.5 Standalone running

The ROB Complex should be able to operate without any global trigger systems (LVL1 or LVL2). When the detector is setting up, for example, sub-detectors may need to run in their own partitions with some simple local triggering provided by the TTC system. This is also true for partitioned calibration running.

This requirement could be met in various ways. The ROB Complex could switch into a mode in which it passes on all event fragments to the downstream Event Builder. Alternatively Event Builder requests could be sent to request consecutive event ids. for each local partition. The requests might alternatively be provided by a special mode of the LVL2 Supervisor, which could still be used even if the LVL2 trigger as such was not in operation. Yet another possibility would be to use the TTC system to trigger event requests. For a further discussion see Section 7.2.2, "TTC interaction," on page 17.

7.3 ROD CONSTRAINTS

7.3.1 Event order

It is an agreed requirement that the ROD will provide the ROB with events in order. (See Trigger & DAQ Interfaces with Front-end Systems: Requirement Document, Version 2.0, DAQ-NO-103). RODs with variable event processing times must buffer all following events until a late event is finished.

7.3.2 ROD latency

The maximum ROD latency will be about 1ms. The number of 100µsec, previously assumed, is too short.

7.3.3 XOFF timing

The ROB and ROD XOFF/XON behaviour must be designed consistently. The storage remaining in the ROB when it triggers its buffer-almost-full XOFF must be suf-

ficient to hold data arriving at full link speed for the time required for the ROD to respond, plus any data in link FIFOs and in the fiber.

7.4 DETECTOR REQUIREMENTS

This section addresses the requirements which may be placed on the ROB by any one or all of the detectors. So far the only such requirement that has been considered in any detail is ROB-level monitoring (see Section 7.4.1, “ROB-level monitoring,” on page 20).

Since the detector designers have not yet fully defined their requirements, it is not possible to make definitive statements at this stage. Those detector groups responding to a query so far did not foresee the need for detector specific monitoring in the ROB. They expect to do most of their monitoring and calibration in the ROD crate. They do intend to do some monitoring and calibration chores on the Event Filter. However, not all detector groups responded. It is proposed that the following specific questions (or similar) be posed to detector groups at the appropriate time via the Detector Interface Group:

a) The ROB will monitor ROD data in a NON-detector specific way, i.e. data format integrity checks. Do you see the ROB as a location for monitoring some aspects of your detector? Will your ROD or ROD crate be sufficient for all your monitoring needs? Would you like the ROB to have the ability to send your events, or 1-in-N events, or events of a certain type, to some specified node for detector specific processing? Will your ROD or ROD crate do such a thing?

Do you foresee the requirement to send the same event to a specified node from more than one ROB, perhaps even from another detector’s ROB?

b) Calibration: How do you see the ROB as part of your calibration scenario? Will all your calibration data sent to a private node on the LAN by your ROD or ROD crate, or will you expect to have available an Event Filter node running your calibration program?

Meanwhile some concern has been expressed that the ROB should not be overloaded with functions, but kept as simple as possible.

7.4.1 ROB-level monitoring

It is not yet confirmed what specific monitoring will be required at the ROB level. Since there are likely to be detailed channel-level monitoring facilities in the RODs, and full-event monitoring (of requested events) will always be possible at the Event Builder or beyond, it may even be decided that ROB-level monitoring is redundant. However the understanding so far is that:

- The ROB will probably monitor inbound data in a non-detector specific way (i.e. for data integrity checks, missing fragments, etc.),
- The ROB will NOT monitor data in a detector specific way. For example it will not collect hit map histograms,
- The ROB might be asked to pass on certain types of data to some specified node where other (possibly detector-specific) monitoring will take place. This action is very similar to that of an RoI request, or an Event request.

It is important to note that it is likely to be a significant design constraint if the ROB is expected to maintain lists of copies of event fragments in anticipation of requests not yet received. The alternative in which a ROB only selects event fragments arriving subsequent to a monitoring request may well be sufficient and could be implemented in the manner described below. It is only likely to be problematic if a requested event-type is so rare that the system has to wait an unacceptably long time for an event sample.

Clearly, the issue of monitoring of event data at the level of the ROB complex requires broad discussion, but, meanwhile, the following description outlines a plausible option:

The ROB should accept a “DataTypeRequest” message. The exact content of this message needs to be formulated, but should at least contain:

- Destination address,
- Data type requested,
- Selection factor f to indicate that 1 in every f of these events are required,
- Number of events requested,
- First event ID to which the request applies. (For synchronisation across ROB (ROBs) in the case that this is required; assuming events arrive in the ROB in the order implied by their IDs, synchronisation could be achieved by requesting all ROB to start sending data for the same event ID and for the same selection factor f . Additionally, in the case that certain data is stored in anticipation of monitoring requests, events may be buffered for a long time and we may need to specify a starting ID to ensure we don't get ancient ones).

The meaning of the DataTypeRequest message is understood to be a standing order to apply to all data which arrives subsequent to the reception of the message. Upon reception, the ROB will look at a Type specifier which is part of the data block. It will do this at the point when it is indexing the event. It is not expected to understand the Type, merely to be able to compare it with the Type requested in the message.

If an event of the specified type arrives then the event ID and delivery address are put in a queue to be dispatched with whatever priority has been assigned to this function. Whilst awaiting dispatch the event is marked in such a way that it will not be deleted until all legitimate calls upon it have been satisfied.

7.5 DATA PROFILE

7.5.1 Event fragment sizes and ordering

Numbers for the fragment sizes used for modelling are shown in Table 1 and Table 2 (where the numbers in brackets are sizes after pre-processing):

TABLE 1.

Event fragment sizes (in bytes) at high luminosity:

	pixels	SCT	TRT	cals	muon prec	muon trig
muon RoIs	800	1600	1000	1800 (1024)	800	380
em/hadron RoIs	800	1600	1000	1800 (1024)		
jet RoIs				1800 (1024)		

TABLE 2.

Event fragment sizes (in bytes) at low luminosity:

	pixels	SCT	TRT	cals	muon prec	muon trig
muon RoIs	80	250	750 (300)	1800 (1024)	800	380
em/hadron RoIs	80	250	750 (300)	1800 (1024)		
jet RoIs				1800 (1024)		

Note 1: Tracker and muon numbers are averages.

Note 2: The full calorimeter data as received from the RODs has to be sent to the Event Builder, but it is possible to send zero suppressed data from the TRT.

These are the sizes of the raw data arriving via the Read-Out Link. For modelling it is assumed that a header of 32 bytes is added to each fragment output by a ROBIN. For a further discussion on headers see Section 7.7, “Data Format,” on page 27.

It is desirable that event fragments arrive in the order specified by their event numbers for easy detection of missing event fragments; this has been agreed as a requirement on the ROD design (see: Section 7.3, “ROD Constraints,” on page 19).

7.5.2 Buffer sizes

The distributions of fragment sizes and LVL2 latencies are expected to be the dominant factors determining the required buffer size.

The actual size of buffer memory required can be estimated from the product of the average amount of data per fragment, the average length of the time interval between event fragment arrival and finishing handling the fragment (discarding has always to be done also when the fragment is passed to the event filter) and of the average event rate. All averages are maxima for averages over a short interval (typically 1 s). Statistical fluctuations may cause temporary buffer overflows, so a safety factor has to be used. A simple rule is that for 1 ms latency and 100 kHz LVL1 accept rate 100 event fragments of 1 kByte each = 0.1 MByte have to be stored.

Computer modelling using current modelling assumptions (an important one is here the use of 15 MByte/s links) provides estimates for the required capacity of the buffer memory. For a good farm processor allocation assignment scheme and for a farm with a size that is not too small (average processor utilization not higher than about 70 - 80 %) the maximum decision times of the LVL2 system for low luminosity, at 40 kHz LVL1 rate, are found to be below 125 ms. For the high luminosity trigger at 40 kHz LVL1 rate a maximum of about 40 ms is found. This implies that the maximum difference in id's of events that have to be stored in the buffer memory of the ROB at the same time is about 5,000. With a maximum fragment size of 1.8 kByte, a circular buffer of 9 MByte without overflow management would therefore be possible. The average number of fragments to be stored is considerably smaller than the maximum difference in event id's : for the low luminosity trigger a typical number is about 1,000, for which 1.8 MByte would be needed. (The fraction of events to be moved in a circular buffer of a certain size could also be estimated by modifying the modelling program if required). Note that it in some designs it may be possible to balance memory requirements between ROBIns and ROBOut by moving long-lived event fragments to the ROBOut.

7.5.3 Deadtime

The only potential cause of deadtime at the ROB level is blocking of the readout chain due to full buffers or malfunction. There should be adequate buffering in the readout chain as a whole to ensure that this does not happen in normal running (see: Section 7.5.2, "Buffer sizes," on page 22).

7.5.4 Latencies

The time required for data requested to become available on the output link of the ROB or ROB Complex is not very critical, assuming that the internal operation of the ROB or ROB Complex is pipelined, so that this time does not determine the maximum rate.

7.6 LVL2 STRATEGY

7.6.1 Pre-processing and data-selection

The processing of data in the LVL2 trigger is usually split into the steps of feature extraction and decision making. In the (dominant) mode of region-of-interest guided analysis this is a particularly advantageous factorization of work: the first step runs on data of a single (sub-) detector and transforms raw data into a physics-related and shorter set of parameters (features), the second operates on multiple sets of features and looks for their compatibility with one or several of the physics hypotheses described in menu tables.

Raw data, however, are not presented to the trigger system in a grouping and format optimally adapted to feature extraction; they are constrained by front-end electronics and transmission up to the readout buffer, whereas the processing algorithms expect them arranged in handy arrays or objects. Summarily, the operations preparing data for feature extraction are named 'preprocessing'. Typically, preprocessing does not transform raw data in an irreversible way, although for some detectors parts of the data present in the readout buffer may be considered unnecessary for LVL2 processing, and therefore omitted, particularly if this alleviates algorithm and transmission requirements.

Most preprocessing also consists of operations that can be executed locally, e.g. inside a readout buffer - with the obvious exception of collecting all data for a region of interest from multiple readout buffers. While it is possible to consider preprocessing as first step of feature extraction, it is also reasonable to imagine architectures that have the required intelligent nodes upstream of feature extraction processors, e.g. implementations that ascribe preprocessing to (groups of) readout buffers ('ROB complex'). Note that the execution of the preprocessing steps in a general-purpose processor can take non-negligible computer time, and the gain by executing these steps in parallel and before a general transmission network is used, can result in critical savings in computer and transmission capacity.

Preprocessing is very much a detector-dependent operation. Below, we describe summarily the possible preprocessing operations for the major detectors (see also LHCC 98-15, chapter 7), the time spent in a general-purpose computer for executing these operations, and the effect they have on subsequent operations. Results have been taken from various measurements on 300 or 400 MHz processors, with no corrections.

(It has been asked whether it might be required to pre-process data merged across sub-detectors. The consensus is that such pre-processing would require an unjustifiable increase in complexity.)

7.6.1.1 Processing for the precision trackers (SCT and pixels)

The readout of the wafer-based detectors presents several specific problems. Wafers are constant-size detector elements that cannot be arranged to cover specific areas in η/ϕ , and the information is sparse compared to the number of channels; the two facts together result in comparatively large and irregular areas collected in each ROBIN. We assume the readout grouping in layers [3]. Furthermore, tracks leave multiple signals in each layer, so that a clustering procedure is necessary. The steps of preprocessing are then the following:

- selection of wafers, inside a readout buffer, that are part of the currently considered region of interest,
- clustering of adjacent strips or pixels into hits corresponding to a single track,
- conversion of local coordinates (wafer ID and strip or pixel number) to global coordinates ($\eta/\phi/\text{radius}$ or similar); this includes combining the normal and stereo wafers for the SCT (present algorithms all are based on space points only, but this may not remain true forever). Further selection of data would be possible using space points to check that hits fall within the ROI region.

Benchmarking of these algorithm parts for the SCT with a commodity 400 MHz Pentium II PC has given the following numbers per ROBIN [4] : selection of wafers: 0.020 msec; clustering: 0.025 / 0.170 msec (SCT / Pixels); coordinate transformations and space points: 0.100 / 0.140 msec (SCT / Pixels). The accumulated effect is a data volume reduction by a factor 3-5, most of it due to wafer selection.

7.6.1.2 Processing for the transition radiation tracker

(See [20] for details of the preprocessing implemented on the microEnable).

The readout of the TRT has been arranged such that the relation between RoIs and ROBINs is optimal, and comparatively little non-RoI information is transmitted

without selecting data inside ROBINs. A subdivision in eta of this detector in the barrel does not exist (except left/right halves), and would not help the data volumes much in the endcap; selection of a suitable phi-sector thus is sufficient to obtain access to all digitizings relevant for a region of interest. On the other hand, the probability of wires being hit by some track ('occupancy') can go to very high values, and the preliminary decisions taken for the TRT data format are giving high priority to the high-luminosity (= high-occupancy) case; this makes it likely that, at least for low luminosity bulk analysis of the TRT ('full scan') a more efficient zero suppression is possible, and can gain more than a factor of 2 on bandwidth requirements without information loss (see Table 2, "Event fragment sizes (in bytes) at low luminosity.;" on page 22).

The TRT data in the ROB contains some bits for every straw. For those with good hits more information is included than those without good hits. It may therefore be useful to pre-process this into a zero suppressed list with addresses, and at the same time reduce the good hit information to 1 low/high threshold bit. Its effect is a more suitable data format (algorithm time reduction by ~30%) and a luminosity-dependent reduction of data volume, which at high L is not more than 1.3. (However dropping drift-time information may compromise algorithm performance.)

In the TRT barrel and endcap the ROB contains segments $2\pi/32$ and $2\pi/96$ wide in ϕ respectively. It might therefore be thought that if enough could be grouped into one place then feature extraction could be done locally. However the FEX algorithm is a Hough transform histogram method. Complete tracks are contained within about 30 degrees and the present thoughts are that it is not possible to do partial peak finding before joining adjacent sectors. In this case it would be necessary to send all histogram bins, resulting still in more data than if just the raw hit information was sent. Alternately one could ignore the overlap problem at some loss of efficiency. Studies (M.Smizanska private communication) indicate that at least for the barrel this is feasible. The best recommendation at present is that a dedicated study of this should be performed.

7.6.1.3 Processing for the calorimeters

For the calorimeters data is most likely grouped inside layers. Preprocessing for the calorimeters consists of selecting the RoI-related subsets of data (first 1024 Bytes of event fragments of 1800 Bytes) in each ROBIN. The arranging of data for the algorithms typically takes more time than feature extraction, although it can only be done partially inside ROBINs. Potentially a large reduction in the amount of data to be passed to the LVL2 system is possible if the RoI related data is zero suppressed. For jets, preprocessing could consist of presenting the data in a much coarser and unified granularity, and would then substantially reduce the volume of data to be transmitted. Grouping towers of ROBs may be useful for operations which are not affected by the boundary, such as computing tower energy sums.

7.6.1.4 Processing for the muon Detectors

The current understanding is that no pre-processing of muon data will be required. Some muon RODs are being designed with this functionality built-in. This understanding should be confirmed at a later date via the Detector Interface Group.

7.6.2 Grouping of ROBs

(Section 8.0, “Scenarios for a full system,” on page 62 and the documents on which it is based contain a discussion of the impact of grouping. See also: [5].

Grouping buffers (ROBIns) together such that they are locally connected potentially offers several advantages:

1. concentration of output data through a high-bandwidth network port
2. reduction in the number of output ports
3. reduction in the number of control and data messages
4. building of RoI fragments
5. other pre-processing and data selection (see Section 7.6.1, “Pre-processing and data-selection,” on page 23)
6. possible (limited) feature extraction

Against these there is the danger that grouping many ROBs together in a ROB Complex may lead to high internal message frequencies and high internal and output bandwidth requirements.

To get a feel for the relative effects of different sizes of grouping, and looking specifically at the benefits of RoI fragment building, numbers are given below for the relative RoI fragment rates for groups of 4 - 40 ROBIns (i.e. the output fragment rate from a set of grouped ROBIns divided by the total output fragment rate from the same number of individual ROBIns). This range of group sizes corresponds to possibilities presently under investigation: all ROBIns connected to a network, few (4) ROBIns grouped to a ROBOut, a smallest component of a HPCN (12 PCI slots), and a fully stuffed readout crate (DAQ-1 prototype, whose substructure we ignore). The numbers have been calculated with the ROBsPerROI program using the standard assumptions about RoI locations and RoI sizes. The ROBs are grouped in the same order as the order in which the eta-phi boundaries of the different ROBs are specified (in general : from low to high eta and for each eta range from $\phi = 0$ to $\phi = 2 \frac{1}{4}$ [5]). These numbers are averaged over all groups for a subdetector. They are not valid for the inner tracker for the B-physics trigger, as for that trigger all ROBs have to supply data.

TABLE 3.

Em ROIs : fractional reduction of the output fragment rate

Grouping factor	Em calo	Had Calo	TRT	SCT	Pixels
4	0.701	0.559	0.531	0.661	0.783
12	0.536	0.319	0.425	0.534	0.627
40	0.385	0.254	0.392	0.376	0.301

TABLE 4.

Jet ROIs : fractional reduction of the output fragment rate

Grouping factor	Em calo	Had Calo	TRT	SCT	Pixels
4	0.588	0.477	0.337	0.522	0.557
12	0.392	0.239	0.192	0.354	0.379
40	0.238	0.160	0.139	0.211	0.173

TABLE 5.

Mu RoIs : fractional reduction of the output fragment rate

Grouping factor	Mu trig	Mu prec	Em calo	Had Calo	TRT	SCT	Pixels
4	1.000	1.000	0.702	0.559	0.415	0.666	0.774
12	0.966	1.000	0.540	0.312	0.305	0.535	0.617
40	0.898	0.645	0.378	0.243	0.259	0.378	0.296

Note: The numbers in tables 3, 4 & 5 depend a lot on the way the ROBINs are grouped. For example, with a slightly different grouping the value for HadCalo at a grouping factor of 4 comes down from 0.559 to 0.433 for Em RoIs and from 0.477 to 0.356 for Jet RoIs. At a grouping factor of 40 the values are close to the limit of 1 ROB complex per RoI and are less dependent upon how the ROBINs are grouped.

The request rates can be much reduced for a number of ROBs by further sub-sequencing the algorithms: using only the middle layer (2nd sampling, at the maximum shower energy) of the calorimeters for lateral shape discrimination, a gain in rate by a factor ~ 2 is achieved. With data from the first layer an overall rate reduction of a factor of about 5 is achieved. A further small gain in reduction is possible by using the data from the presampler and back layer [6].

It is clear from the tables that grouping can offer significant reductions in average fragment rates. A more detailed analysis for a ROB complex has been documented in [6].

7.6.3 RoI Request distribution

The current LVL2 assumption is that RoI requests will be transmitted via the network connecting the ROBs to the LVL2 processors. For the TRT scan and for a LVL2 missing energy trigger a possibility for broadcasting may be necessary. The requests need to be buffered in the ROBs, computer model results [7] indicate that probably not more than 40 requests need to be buffered at any time.

7.6.4 LVL2 decision record

The current assumption is that LVL2 decision records will be transmitted via the network connecting the ROBs to the LVL2 processors. A possibility for broadcasting may be necessary. (The possibility of using the TTC technology for distributing decision records has been raised, but such a mechanism would have to be clearly separate from the actual TTC system, and is not currently being pursued). Decisions will probably be grouped for more efficient handling in the ROB Complexes. Accepts could be sent individually and rejects in groups, or accepts and rejects could be sent together. It is also possible that only rejects are passed directly to the ROBs, while accepts are sent to the DAQ system, which then requests data from the ROBs.

7.7 DATA FORMAT

7.7.1 Message types

The ROB Complex has to deal with the following input messages (see also Section 5.1, "Context Diagram," on page 11) :

- ROD data (event fragment data, format as described in Section 7.7.2, “Event fragment format,” on page 28) arriving via the Read Out Link (ROL),
- RoI request,
- LVL2 decisions,
- EB data requests,
- Event discard requests,
- Monitoring / status requests,
- Run-control requests.

The following output needs to be generated :

- RoI data (event fragment data, format as described in Section 7.7.2, “Event fragment format,” on page 28),
- EB data (event fragment data, format as described in Section 7.7.2, “Event fragment format,” on page 28),
- Monitoring/status data,
- Run-control responses,
- Error messages.

7.7.2 Event fragment format

A standard format for event fragments was presented at the November 1998 ROD workshop and is also documented in [24]. This latter format is intended as a first step in defining the final ATLAS event format. The following discussion is based on the information in this note. Several issues remain to be decided for the final format. These are listed at the end of this section.

An event fragment as sent by the RODs consists of three parts : (i) a header, (ii) detector specific data and status information and (iii) a trailer.

The header contains the following items (one 32-bit word per item) :

- word 0 : start of header marker, also indicating the type of fragment (ROD),
- word 1 : header size,
- word 2 : format version number,
- word 3 : identifier of the ROD,
- word 4 : LVL1 ID,
- word 5 : bunch-crossing ID,
- word 6 : LVL1 trigger type,
- word 7 : detector specific event type.

The trailer contains the following items (one 32-bit word per item) :

- number of status words,
- number of data words,
- flag indicating whether the status information is located directly after the header or just in front of the trailer.

It is also necessary for the buffer addressing hardware to be able to immediately recognise the start and end of an event fragment. In the S-link implementation this is achieved by control words which bracket the complete fragment: the above header is preceded by a control word signalling the start of the fragment, and the trailer is followed by a control word signalling the end of the fragment.

The data to be output by the ROB complex consists of a series of ROC fragments (ROC can be read as Read Out Crate or as ROB Complex). Each ROC fragment consists of a header with a generic part and a source dependent part followed by a block of data, assembled from ROB(In) fragments. A ROB(In) fragment has the same structure as the ROC fragment : a header followed by a block of data, consisting of the ROD data.

The header contains the following items (one 32-bit word per item) :

- word 0 : start of header marker, also indicating the type of fragment (ROB = ROBIN, Crate = ROB Complex, Sub-Detector, Full Event),
- word 1 : total fragment size,
- word 2 : header size,
- word 3 : format version number,
- word 4 : source identifier,
- word 5 : NS = number of status words ("status elements"),
- words 6 to 6 + NS : block of status words ("status elements"), contents to be defined,
- word 6 + NS : ND = number of data blocks ("offset elements") present in the fragment,
- words 6 + NS to 6 + NS + ND : offset in fragment of data block ND,
- word 6 + NS + ND : NF : number of data words with fragment specific information,
- words 6 + NS + ND to 6 + NS + ND + NF : data words with fragment specific information.

The status words could be used for storing information on the preprocessing algorithm applied, and the RoI request identifier to match RoI data with RoI requests, as the same ROB could be asked for data for several RoIs in the same event. In this format the event number is embedded in the ROD data, but could also be contained in addition in one of the status words.

With respect to the design of the ROB two different questions have to be answered :

1. Does the format of the input data as defined cause problems with respect to the design of the ROB?

Answer: The format of the header and the trailer is independent of the data contents of the fragment. This makes it possible to find all relevant information (in particular the LVL1 ID) at a fixed offset relative to the start or end of the total fragment (including head and trailer), as required for the ROB (in view of handling by dedicated hardware or minimal software).

2. The fragments to be output by the ROB Complex are derived from the input fragments. How do these output fragments have to be formed and what are the consequences for the design and/or the performance of the ROB Complex?

Answer (without pre-processing):

In the case that there is no preprocessing at the ROB level, then the ROBIn and the ROBOut have only to add header information to the ROD fragments, but do not have to change information inside them. The ROBIn has to build the header described earlier. Input for building this header is the ROD fragment size, the format version number, the identifier of the ROBIn and status and monitoring information (e.g. filling degree of buffer, number of fragments in buffer, etc.). The ROD fragment can be sent directly after this header to the ROBOut. The ROB Controller has to build the same type of header as the ROBIn has to build. Sending the ROC fragment consists of sending the header immediately followed by the ROBIn fragments that together form the ROC fragment, in combination with the appropriate network specific framing information (including the address of the destination). Building a complete fragment in memory is not necessary when the network interface can handle messages consisting of several parts stored in non-contiguous memory efficiently.

Answer (with pre-processing):

A pre-processing algorithm inside the FPGA, as done, for example, with microE-nable, changes the format of the fragment data. So the fragment data format (not the header) for input as well as for output impacts the FPGA implementation. As the input format seems to be untouchable, a good output format for pre-processing is necessary.

For building the headers the amount of computation required seems to be small. The impact of it seems to be largest for an all-FPGA based ROB Complex in which case it depends strongly on the kind of information that needs to be collected.

Issues that remain to be decided include:

- It should be checked by calculation that the proposed format does not imply an excessive overhead when sending sparse data.
- Should the same format be used for data to LVL2 and data to DAQ?
- Is it needed to indicate if a pre-processing algorithm has been applied (and which)?
- Is an RoI request identifier needed to match RoI data with RoI requests, since the same ROB could be asked for >1 RoI in the same event?
- Does it matter that the event ID is not available at the level of the ROC and ROB(In) fragment headers, and only appears embedded in the ROD data?

Note: The possibility exists to include monitoring or status information (e.g. the filling degree of the buffer memory in the ROBIn) in the header (probably to be stored in the fragment specific words). This could be used to obtain an overview of the behaviour of the whole ROB system at the level of the EF. Statistics could be accumulated per EF processor and be collected regularly or on request. This type of monitoring has the advantage that for collection of information no additional data paths are needed other than those used for acquiring the event data: the event builder takes care of bringing together the information for the whole system. As monitoring takes place for events accepted by the LVL2 trigger some bias may be present in the monitoring information collected. The amount of bias can be estimated with control triggers also used for monitoring the correct functioning of the LVL2 trigger. Including monitoring information in the header also allows for straightforward (as monitoring and event data come together) fact-finding in the off-line analysis about the behaviour and status of the DAQ system.

In general the nature of the format should have negligible influence on performance and cost of the ROB Complex, though care needs to be taken that data added to the raw data (in the form of headers and trailers) does not represent a large fraction of the total data. It cannot be excluded that the choice of format could become critical if the formatting needs to be done on-the-fly in an FPGA-based design.

7.7.3 Format for other messages

Several prototype formats have been used in the LVL2 demonstrator, pilot project and DAQ-1 projects for messages like RoI request and decision messages. These prototype formats should be revisited jointly by the DAQ and higher level trigger groups (including the Detector Interface Group). At the time scale of the TDR the format needs to be fixed.

7.7.4 Byte order

The data handled by the ROB is usually moved as 32-bit chunks inside the ROB Complex. Header and trailer information for the event fragment data following the format referred to in Section 7.7.2, “Event fragment format,” on page 28, is 32-bit data. However, the event data and the contents of control messages may contain bit fields. For correct isolation of these bit fields knowledge about the byte ordering is crucial. Furthermore transmission via network interfaces in general involves breaking 32-bit words into smaller chunks that are assembled again in 32-bit words after reception. Also here the byte ordering has to be taken into account. Transmission via S-link does not affect the byte ordering inside 32-bit words transmitted.

Byte ordering problems are not specific for the ROB Complex, but have to be dealt with ATLAS-wide. It is desirable that at the time scale of the TDR the byte ordering is defined.

7.8 OPERATIONAL FRAMEWORK

The software framework within which a ROB will operate is not yet finally decided. Relevant prototyping studies have taken place in the contexts of the LVL2 Reference Software[9] and DAQ-1[23]. Both these developments assume eventual convergence with facilities provided by the DAQ BackEnd software[10]. Meanwhile the development of software within the ROB Complex project has necessarily involved consideration of operational issues. Other studies involving specification of the operational framework are documented in the ROB URD [8], and the UML description of the ROB Complex [1].

The sections below describe the functionality that will be required from the ROB point of view.

7.8.1 Run Control

From the Run-Control point of view the ROB Complex is a state machine that will be in one of a number of states, and must be responsive to run control commands to change its state. Other external (to the ROB Complex) conditions may also cause a change of state. This implies also that the ROB Complex maintains knowledge of its own state. In general, it will probably also be necessary for the ROB Complex to provide the run control system with a notification that a particular state change has completed.

The major state changes that are likely to be relevant to the ROB Complex are:

Initialisation:

This includes downloading of software and parameters, establishment of hardware addresses, setting initial values for internal counters. This might be done automatically at power-on and/or system boot.

The ROB URD specifies that a “ROB must be able to be ‘rebooted’ without interrupting data-taking”. If this remains a requirement it implies a synchronisation mechanism, using event_ID and/or bunch_crossing information. This is automatic, based on event ID, in the event indexing schemes implemented so far, though they would obviously result in missing fragments at the feature extraction stage of LVL2, which would need to be handled appropriately.

Configuration:

This refers to the setting up of information that might differ between ROBAs or between runs. It includes establishing the correlations between ROBIN links and RODs, and between RODs and regions of interest. It might also involve the downloading of special software and associated parameters for particular sub-detectors (e.g. for pre-processing). Also some configuration parameters may need to be set differently for different sub-detectors (e.g. page-lengths in a page-managed buffer).

If a ROB Complex is ultimately composed of more than one physical module, consideration must be given to the identification of its components. It is assumed that the physical configuration of the system will be tracked by housekeeping software and reflected in an appropriate database, and this may well be used as the primary source of component identification. Nonetheless it seems prudent to implement a mechanism for checking the database against the physical reality. All modules should therefore have some form of intrinsic identification that is electronically readable. Moreover conventions should be established to confirm the connections with components external to the ROB Complex, such as RODs, LVL2 feature extractors, and event builders.

Reset:

To return the ROB Complex components to their starting state (i.e. after initialisation and configuration), e.g. in the event of a system problem. This should include zeroing counters and resetting tables and pointers to their initial values. It should also include presenting appropriate signals to external interfaces (e.g. XOFF on the Read Out Links). Note that, in practice, several types of reset may be required, but it is assumed at this stage that these can be regarded as combinations of the state changes listed here.

Mode change:

It is envisaged that a ROB Complex will be capable of running in different modes. These may be fundamentally different modes of operation, such as testing versus running, or they may just involve parameter changes, such as switching monitoring types.

Basic modes are likely to include:

- Internal testing,
- Full data sampling,
- Calibration,
- Normal data taking.

Parameters which could be changed include:

- Monitoring parameters.

Start:

This action is basically a matter of releasing the initial hold signals on the various inputs (e.g. XOFF on the ReadOut Links, or whatever hold-off mechanism is used on the RoI Request and Event Builder inputs). However, the run control system should also be notified that the ROB Complex has 'started', so that it can inform the components that communicate with it.

Stop:

This action is the inverse of 'Start'. It asserts the various hold-off mechanisms and informs the run control system. It is not the same as 'Reset'.

Fault:

This is a state that the ROB goes to automatically in case of errors that cannot be recovered locally.

7.8.2 Errors and error handling

The following sub-sections describe possible errors and suggestions for the associated error handling. These sections should be used as checklist in compiling a definitive error table.

Errors should be reported to different locations according to their type:

- in event status if the error can be defined as specific to an event: loss, corruption during transmission, etc.,
- in local rob statistics,
- to monitor in charge of this ROB,
- to run control if the working of the rob is seriously disturbed, going up to the reboot of the ROB.

The number of error reports per time interval may need to be limited.

7.8.2.1 Errors in and related to event data received:

- Transmission error on ROL :
 - link down,
 - BOB or EOB not recognized : leads to fragment with error,
 - missing fragment,
 - BOB and EOB recognized : fragment with error,
- Error flag in fragment set before transmission by ROD : fragment with error,

- Event id not equal to event id of last fragment + 1,
- Fragment too big,
- Fragment buffer memory overflow,
- BCID mismatch between fragments.

Error handling:

- Fragment with error : output if requested with error flag set,
- Incorrect event id : if id = id of last fragment + 2 : set missing flag for id-1 and continue, else STOP data input (better approach may be possible here),
- Fragment too big : truncate in buffer memory and hence also on output,
- Fragment buffer memory overflow : XOFF -> ROD if buffer memory almost full.

7.8.2.2 Errors in and related to messages from LVL2:

- Message not understood,
- RoI request for fragment with id larger than id of last fragment received + threshold¹,
- Invalid destination for data,
- Reject for missing fragment,
- If applicable : accept for missing fragment,
- RoI request buffer overflow,
- Decision buffer overflow.

Error handling:

- Message not understood : ignore,
- RoI request for fragment with id too large : ignore,
- Invalid destination : do not output data,
- Reject or accept of missing fragment : ignore,
- Buffer overflow : ignore input.

7.8.2.3 Errors in and related to messages from EB:

- Message not understood,
- Request for missing fragment,
- Invalid destination for data requested,
- Request buffer overflow,
- If applicable : delete command for missing fragment,

1. RoI requests could in principle arrive before event fragments. The difference between the id of the fragment requested and of the id of the event fragment last arrived can be required to be not larger than a certain threshold value. However, care needs to be taken here: fragments could arrive up to about 1 ms after LVL1; on the other hand ID values will wrap modulo some number.

- If applicable : delete buffer overflow.

Error handling :

- Message not understood : ignore,
- Request for missing fragment : ignore,
- Invalid destination : do not output data,
- Buffer overflow : ignore input.

7.8.2.4 Other errors in and related to messages received:

- Message not understood,
- Monitoring information requested can not be delivered,
- Status information requested can not be delivered,
- Invalid destination data for monitoring or status data requested,
- Requested command cannot be executed.

Error handling :

- Message not understood : ignore,
- Information requested cannot be delivered : send appropriate reply,
- Invalid destination : do not output data,
- Command cannot be executed : send appropriate reply.

7.8.2.5 Errors with output of data:

- Overflow of output buffer for RoI data,
- Overflow of output buffer for data to EB,
- Overflow of output buffer for monitoring data,
- Overflow of output buffer for status data,
- Output impossible (e.g. output link down).

Error handling :

- Overflow of output buffer (will also occur if output is impossible) : depends on the destination. For LVL2 and EB : wait until output is freed. Other information could be discarded; the flow of data to LVL2 and EB should not be hindered.

7.8.2.6 Timeouts

- Fragment data not arriving within maximum time interval after RoI request,
- If applicable : no EB request within certain maximum time interval after LVL2 accept,
- No delete request after certain maximum time interval after arrival of fragment.

Error handling :

- Fragment not in time for RoI request : discard request,
- EB request or delete too late : discard data (otherwise buffer will fill up).

7.8.3 Fault handling

This sub-section refers to various kinds of failures that require corrective action. A comprehensive survey of possible faults has not yet been conducted and is anyway premature whilst key architectural and implementation issues remain undecided. The sub-section is included to serve as a checklist item for future development.

The kind of action that might be needed is illustrated by the following suggestion for handling the failure of a replaceable ROB component:

Stop data taking, swap as quickly as possible, run ROB self test, when problem does not disappear resume data taking with ROB made inactive and try to find cause of problem while not disturbing data taking.

This sort of prescription affects the running of the whole experiment and obviously requires agreement at a very general level.

7.8.4 Status monitoring and reporting

It is assumed that the ROB complex will need to report information to the general Trigger/DAQ status-reporting system. Much of this information will be reported only on request, but it should be considered whether or not some information should be reported routinely. At least it might be desirable to switch into a mode where information is reported on a regular basis without specific requests (or will this always be handled by the status requesting system e.g. with periodic requests?).

There are at least three categories of information that might be requested:

- ROB performance statistics,
- Current state of components, counters, and flags,
- Identification and configuration information.

Perhaps the first category should be listed separately (it is a kind of monitoring, but it should not be confused with data monitoring at the ROB level).

As discussed in Section 7.7.2, “Event fragment format,” on page 28, it may make sense to include the information mentioned or a subset of it in the event fragments passed to the event builder.

7.8.4.1 ROB Performance Statistics:

There are a number of quantities that would be interesting to monitor, primarily to help to identify bottlenecks and/or possibilities for further optimization. The problem in all cases is the extent to which the measurement process itself might distort the results. There is also the related danger that the measurements might degrade the overall performance of the readout.

Note that the event rate itself, along with the information about the time that event data stays in the buffers, is common across all the buffers and can better be monitored elsewhere.

The main quantities that come to mind are:

- Overall CPU loading (for each of the CPUs involved in a ROB Complex),

- More detailed profiling of the software, to determine the relative time spent on different tasks (event fragment indexing, RoI request handling, RoI data transmission, event fragment release, etc.),
- Buffer occupancy (average and peak) for all buffers in the complex (main data fragment buffer, network output buffer if present, RoI request input buffer, Event Builder request input buffer, fragment release message buffer, etc.),
- Event fragment size histogram values. Alternatively the average and peak numbers of events in the buffer; these combined with the buffer occupancy would give an indication of the average event size. Note that event fragment sizes are also available at the RODs, and will almost inevitably be monitored there for local sub-detector purposes, so it is not clear that they should be monitored in the ROBs,
- Percentage of buffers (ReadOut Links) within a ROB Complex that are involved in an RoI,
- Statistics on errors,
- Statistics on XON/XOFF occurrences.

7.8.4.2 Current state of components, counters, and flags:

Although the values of error counters would get reported to the error reporting system, at least when they reach certain thresholds, the instantaneous values of these counters are likely to be requested also by the status reporting system. Additionally some of these ‘counters’ might just be flags – indicating the presence of an error condition (which may or may not be serious enough to be reported to the error reporting system). The difference is that errors get reported when they happen, whilst status reports show the current conditions.

Similarly, state information reported to the run control system might also be requested by the status reporting system.

With a multi-component ROB Complex the individual components might also be in various states of functioning and might consequently be reported as e.g. ‘good’, ‘bad’, or ‘under investigation’. This might specifically be the case with individual readout channels (e.g. ROBINs).

7.8.4.3 Identification and configuration information:

The ROB Complex should presumably be able to identify itself and its components, along with their current configurations.

7.9 IMPLEMENTATION

7.9.1 Hardware architecture

Various possibilities for constructing the buffer section of the final system are described in the chapter on “scenarios”. These extrapolate current development work into paper studies of full solutions. The range is wide and the final choice will depend on assessment of trade-offs discussed in other sections. The scenarios can be grouped into the following three broad categories according to the topology of the connections between buffers and processors, bearing in mind that this topology may be different for LVL2 processors and Event Filter processors.

1. “simple ROB” : one ROL per ROB, no separate ROB Controller and an individual network connection per ROB,
2. multi-buffer ROB complexes with network connection per complex,
3. multi-buffer ROB complexes with direct connections to farm processors taking care of network interfacing for the ROB as well as trigger processing.

Note that some of these scenarios propose solutions in which not all ROB complexes are identical. There is general agreement that the goal of common hardware across sub-detectors at the ROB stage is desirable and most scenarios strictly achieve this at the module level, but exact interchangeability of complete ROB complexes may be worth trading off against savings in overall system size and cost.

In category 1 the components of a ROB complex are collapsed into a single module, with the ROBIn buffer-management processing element playing also the role of a ROB Controller, and the ROBOut comprising just the appropriate network interface. Such modules, being of similar size to current ROBIn implementations, could be physically grouped in the same sort of way, e.g. 6-12 modules on a 9U card. This solution represents a straightforward option if the cost of the associated networking (interfaces and switching) is low enough. Since there is no separate ROB Controller, paths must be provided for the relevant functions such as initialisation, run-control, status and error reporting, and monitoring. Physically it seems prudent to provide a separate secondary network path for these, possibly shared between the modules making up a card. The associated external functionality could still be provided by a component at crate-level, or by a more global component of the supervisory or back-end infrastructure.

Category 2 represents the “classic” ROB complex. It offers the potential to save on networking costs by tuning the size of the complex to the estimated access rates and output bandwidth. It also offers the potential to reduce the output bandwidth, by selection and/or pre-processing (though it should be noted that this is not excluded in Category 1 solutions), and to reduce message rates by local fan-out of requests inside a complex. The canonical structure of such a complex is a grouping of several ROBIns with a smaller number (possibly just one) of output interfaces (ROBOuts), all under the control of a local ROB Controller. Note that, with different terminology, such a structure could also describe the DAQ-1 style Readout Crate. In this category the connection to processors (LVL2 or Event Filter) is decoupled from the interconnection between ROB complex components. This latter can itself be bus-based or point-to-point (see: Section 7.9.3, “Physical organisation,” on page 41 below). The SMP-based scenarios could be considered to be in this category, but offering the potential added advantage of pre-processing across the buffers of the ROB Complex and the use of multiple PCI buses, forming what has been described as an “active ROB Complex”.

Category 3 refers to more radical proposals in which processors play a more integrated role in the overall system. The scenarios share the potential advantages of multi-buffer complexes in Category 2, but also seek to simplify the networking task by employing more direct connection between buffers and processors. In the second set of SHARC-based scenarios there are point-to-point links from ROB complexes to processors, and the interconnected processors themselves comprise the LVL2 (and possibly Event Filter) “network”. More extreme versions of the SMP approach, in which the SMP processors perform tasks beyond pre-processing, might also fall into this category.

7.9.2 Data volumes & request rates

The following tables summarise the bandwidths and rates relevant to ROB complex grouping that have been derived from modelling studies.

(NB : the nominal LVL1 rate is 40 kHz for low and high luminosity, the nominal accept rate is 2 kHz)

7.9.2.1 ROBIN to LVL2 bandwidths:

TABLE 6.

40.1 kHz LVL1, low luminosity, bandwidth required per ROBIN to LVL2, B-physics trigger, *italic* numbers indicate the use of pre-processing in the calorimeter and TRT ROBs

	muon MDT	muon trig	e.m. cal	had cal	TRT	SCT	Pixels
Lower limit data volume per ROBIN (MByte/s)	0.15	0.22	0.46 <i>0.26</i>	1.26 <i>0.73</i>	8.07 <i>3.43</i>	2.88	1.18
Average data volume per ROBIN (MByte/s)	0.45	0.47	1.43 <i>0.83</i>	2.93 <i>1.69</i>	8.08 <i>3.43</i>	3.00	1.24
Upper limit data volume per ROBIN (MByte/s)	0.98	0.73	3.80 <i>2.19</i>	4.91 <i>2.83</i>	8.12 <i>3.45</i>	3.11	1.34

TABLE 7.

39.4 kHz LVL1, high luminosity, bandwidth required per ROBIN to LVL2, *italic* numbers indicate the use of pre-processing in the calorimeter ROBs

	muon MDT	muon trig	e.m. cal	had cal	TRT	SCT	Pixels
Lower limit data volume per ROBIN (MByte/s)	0.04	0.06	0.18 <i>0.10</i>	0.56 <i>0.32</i>	0.28	0.37	0.41
Average data volume per ROBIN (MByte/s)	0.12	0.13	1.44 <i>0.83</i>	1.58 <i>0.91</i>	0.30	0.90	0.69
Upper limit data volume per ROBIN (MByte/s)	0.27	0.20	3.84 <i>2.21</i>	2.63 <i>1.52</i>	0.34	1.34	1.24

Conclusion : max. bandwidth required to LVL2 low lumi = 8.1, high lumi = 3.8 MByte/s

7.9.2.2 ROBIN total output bandwidths:

TABLE 8.

40.1 kHz LVL1, low luminosity, B-physics trigger, bandwidth required per ROBIN, *italic* numbers indicate the use of pre-processing in the calorimeter and TRT ROBs

	muon MDT	muon trig	e.m. cal	had cal	TRT	SCT	Pixels
Upper limit to LVL2 (MByte/s)	0.98	0.73	3.80 <i>2.19</i>	4.91 <i>2.83</i>	8.12 <i>3.45</i>	3.11	1.34
Output to EB per ROBIN (MByte/s)	1.67	0.83	3.68	3.68	1.57 <i>0.67</i>	0.57	0.22
Maximum output per ROBIN (MByte/s)	2.65	1.55	7.47 <i>5.87</i>	8.58 <i>6.51</i>	9.69 <i>4.11</i>	3.68	1.56

TABLE 9.

39.4 kHz LVL1, high luminosity, bandwidth required per ROBIN, *italic* numbers indicate the use of pre-processing in the calorimeter ROBs

	muon MDT	muon trig	e.m. cal	had cal	TRT	SCT	Pixels
Upper limit to LVL2 (MByte/s)	0.27	0.20	3.84 <i>2.21</i>	2.63 <i>1.52</i>	0.34	1.34	1.24
Output to EB per ROBIN (MByte/s)	1.64	0.81	3.61	3.61	2.03	3.21	1.64
Maximum output per ROBIN (MByte/s)	1.91	1.01	7.45 <i>5.82</i>	6.24 <i>5.13</i>	2.38	4.55	2.88

Conclusion : max. average bandwidth required low lumi = 9.7, high lumi = 7.5 MByte/s (without pre-processing)

Overall : 16 MByte/s allows scaling up to LVL1 rate of 100 kHz, if scan rate is kept the same for the low luminosity trigger.

7.9.2.3 RoI request rates:

TABLE 10.

40.1 kHz LVL1, low luminosity, B-physics trigger, RoI request rate per ROBIN

	muon MDT	muon trig	e.m. cal	had cal	TRT	SCT	Pixels
Minimum RoI request rate per ROBIN (kHz)	0.18	0.54	0.25	0.69	10.32	10.21	10.58
RoI request rate per ROBIN (kHz)	0.54	1.14	0.78	1.60	10.34	10.65	11.04
Maximum RoI req. rate per ROBIN (kHz)	1.18	1.76	2.07	2.68	10.39	11.05	11.93

TABLE 11.

39.4 kHz LVL1, high luminosity, RoI request rate per ROBIN

	muon MDT	muon trig	e.m. cal	had cal	TRT	SCT	Pixels
Minimum RoI request rate per ROBIN (kHz)	0.05	0.15	0.10	0.31	0.27	0.23	0.49
RoI request rate per ROBIN (kHz)	0.15	0.32	0.79	0.86	0.29	0.55	0.82
Maximum RoI req. rate per ROBIN (kHz)	0.33	0.49	2.10	1.44	0.33	0.82	1.49

Conclusion : max. request rate low lumi = 11.9 kHz, high lumi = 2.1 kHz

Overall : 15 kHz allows scaling up to LVL1 rate of 100 kHz, if scan rate is kept the same for the low luminosity trigger.

7.9.3 Physical organisation

As buffering will need to be provided for something over 1500 readout links in total it is worth paying some attention to minimizing the costs of racks, crates, power supplies, etc.

A system consisting of boards in crates is desirable for reasons of reliability and maintainability and would permit the building of a compact system. An alternative might be, however, to use the backplanes or housing systems of commercially available processors, as, for example, in the case of the SMP-based scenarios. None the less most of the scenarios assume a standard crate housing, 6U or 9U in height. Depending on the architecture such crates have standard backplanes, proprietary backplanes, or no specific backplanes. Thus the simple-ROB scenario requires only power and network connections, with modules mounted on standard size baseboards, in either embedded or daughter-board fashion. The scenario 1 SHARC configurations are assumed to sit in VME crates, but the connections to ROB Controllers are via proprietary PCI connectors which directly couple neighbouring boards, whilst the connections to the Event Builder are via SHARC links implemented on a dedicated backplane. In the DAQ-1 system modules sit on a VME backplane in a standard VME crate, but may use PVIC for distribution of high-frequency messages.

As far as standard backplanes are concerned current work assumes either VME or PCI. The latter already seems a plausible option for a ROB complex (see: Section 7.9.11, "Internal communication," on page 51), whilst the former may also be viable in conjunction with e.g. PVIC as mentioned above. Two scenarios explore the use of a Compact PCI crate, one of them incorporating an additional proprietary bus alongside PCI (Atlantis proposal). The other demonstrates the flexibility of bridging different segments in Compact PCI to build different size ROB complexes to cope with the different bandwidth and request frequencies associated with different sub-detectors, whilst still achieving efficient utilisation of crate space. It is calculated that the projected 1530 buffers could be organised into 300 ROB complexes with efficient use of grouping factors of 4, 5 and 12. With one ROBIN assumed to occupy one 3U card this would require 81 6U-size Compact PCI crates.

For 9U-size VME crates and assuming 6 ROBIns per card the total number of crates is projected to be 25-40 (SHARC scenario 1 configurations), or, if direct S-link outputs connected at the rear of the crate allow 8 ROBIns per card, possibly as low as 10 (SHARC scenario 2)!

7.9.4 Front-end connections

Two kinds of Read Out Link will probably need to be accommodated: cheap electrical ROLs for when RODs are near to ROBIs, and more expensive optical ROLs for when RODs are remote from the ROBIs. S-link implementations are taken as a benchmark, since S-link is a connection option proposed for ATLAS that is the basis for most current ROB and ROD prototype work.

Connecting more than one or two Read-Out Links to a single motherboard can arguably be done best via the front panel, particularly if we assume a parallel data path as in the current the S-link connector. A single 2U high daughter board may provide 2 S-link connections, though small size optical connectors will probably allow at least 8 bi-directional links to fit onto a 6U motherboard (approx. 20cm height). The same is true for electrical interfaces using RJ45-style connectors like fast/gigabit ethernet. On the other hand, the possibility of using rear connector modules on new-specification crates with extra depth should be also borne in mind¹.

The possible requirement of hot-swappability raises further questions:

1. Will a lot of links attached to the front-panel of the crate effectively prohibit hot-swapping, because all or a large number of boards have to be disconnected in order to replace one of them ?
2. If the links are attached from the rear, is there a standard supporting hot-swapping for this type of connections or will we have to develop an in-house solution?

It is possible to perform optical to electrical conversion for a number of optical fibers in a single device with a footprint comparable to that of a single integrated circuit. The Infineon (formerly Siemens) PAROLI device provides an example supporting conversion for 12 fibers each carrying up to 1.25 Gbit/s data streams. A connector for an optical ribbon cable with 12 fibres (fibre spacing in the cable is 0.25 mm) is an integral part of the PAROLI. The outside dimensions of the connector are 15.4 mm x 6.8 mm. This type of technology therefore allows to connect many ROLs with a small connector if a bandwidth of 1.25 Gbit/s per ROL is sufficient and if ROLs can be combined in a ribbon cable. A possibility to achieve the latter is to use implementations of S-link interfaces with serial electrical output in the RODs and to feed the outputs of 12 RODs into a single PAROLI transmitter chip. The signals received at the other end of the ribbon cable could be sent from the PAROLI receiver chip via short cables to different modules (with serial electrical input) if 12 optical fibres cannot be handled by a single module. Although in this

1. For rear connection the number of connector pins needs to be watched. Using just the 5-row P2 and P3, four S-link inputs can be accommodated in total on a 9U board. Anything more than four S-link inputs will require extra connectors, i.e. the P1 and both P0 and P4 (the smaller hard-metric connectors in between P1, P2 and P3).

case the advantage of the high density interconnection is less, if prices develop as anticipated, potentially the system cost may be reduced. At NIKHEF this latter option has been studied with PAROLI devices on loan from CERN, see section 9.2 for a short report.

7.9.5 Use of daughter boards

Since present electronic components are very thin, mother/daughter-board constructions are now quite feasible and are recommended by some of the present prototype designers, provided that there are no problems with cooling (though the thin components leave space for air flow). This type of construction offers more board space per slot and also gives flexibility with respect to the part of the design residing on the daughter boards. Using S-link daughter boards makes application of different technologies for the Read-Out Link possible and may be attractive.

Also if multiple links per board are used, daughterboards will allow single links to be replaced in the event of failure (no need to replace the whole board).

7.9.5.1 Crate size

The size of modules, and consequently the crate to house them in, depends upon a balance of various factors, such as the ROB Complex grouping size, the size of other modules sharing the same crate, connection constraints, power constraints, maintainability, standardisation, etc. As far as the ROB Complex design is concerned, the crucial factor is the desired grouping size.

For 19-inch crate mechanics the choice is presumably between 6U and 9U (and possibly 3U).

A 9U card mechanically could possibly accommodate 4 daughter boards (giving, for example, 8 S-link inputs at 2 per daughter board), whilst a 6U card might allow 3 daughter boards, giving, say, 6 front-end inputs. However care needs to be taken with such estimates. For instance three standard mezzanine cards (e.g. CMC format) on a 6U card leave no space for handles, whilst the use of 5 cm daughterboards (like those used for M-Link) instead of 7cm (like the present S-link) would allow 4 cards per board, with handle space.

If a TTC connection is required at the ROB level, it is worth considering whether any advantage can be taken of commonality with ROD implementations. The RODs are most likely to be housed in 9U crates, and will carry a TTC connection and distribution system. However it is not obvious that this is relevant to the ROB design since the 40 MHz TTC signals are not required at the ROB level, and the remaining TTC information could probably be distributed in a standard crate without special modifications.

7.9.6 Level of integration

A complete ROBin, excluding buffer memory, perhaps can be designed as an ASIC, using, for example, an existing DSP core. Intrinsic pros and cons of ASICs are:

- potential pros: low-cost in big enough quantities; small size; reliability; low power,

- potential cons: high-cost in small quantities; cannot be easily modified; design is work for specialists.

Since the ultimate benefit of ASICs would be the lower cost associated with a smaller system, this saving should be compared against the cost of manufacturing on a relatively small scale. It looks unlikely that this comparison would favour the use of ASICs, especially since the ROB design does not have to confront issues of rad hardness, simultaneous use of binary and analogue signals, >1M gates, very high clock rate, large number of chips, or other extreme requirements that force giving up the flexibility of FPGAs. Furthermore the evolution in FPGA technology is rapid and a combination of buffer memory and FPGA and a processor, or even only buffer memory and an FPGA could probably take care of the ROBIn functionality in just two or three chips.

It should be noted that there are various options for converting FPGA designs to chips which are cheaper than FPGAs for intermediate quantities. However the flexibility of reconfiguring is lost.

7.9.7 Power

A 6U crate for a 19 inch rack has typically 20 slots and a power supply of 800 W - 1.2 kW, i.e. 40 - 60 W per slot. In a 9U crate about 75 W per slot may be available. With 6 or 8 input links per board less than 10 W per read-out link will be available in a crate completely filled with boards. In principle it should be possible to handle the heat produced in a fully-loaded crate, but it may require something more than air-cooling.

The VME64xP standard gives the following maximum currents for a typical 9U Power supply (Appendix D): 200A@3.3V and 300A @5V.

The current generation FCS-link dissipates 5 Watt per side. This number is going to be less for newer generations, even for ones that go full speed (40 MHz, 32-bits data), due to the use of newer 3V3 components.

An FPGA board (like the new Atlantis AIB) is expected to consume less than 10A@3.3V plus 2A@5V (= 43W), without link modules. Adding 8 modules @5W the total power consumption would be 83W for 8 links. The power consumption will go down in the future when 2.5V and 1.8V technology will become available.

Overall it looks as though the anticipated front-end link densities could just about be accommodated using current technology, so there should not really be any problem with providing and dissipating power if lower voltage technologies are used in the production system. This will need to be watched, however, if link densities are increased much beyond 8 per board, for example in an embedded design.

Note: It has been suggested that high track currents might become a problem at low voltages (3.3V, 1.8V), but in fact the currents will even become less when going to lower voltages.

7.9.8 Microprocessors & FPGAs

In so far as the ROB Complex might well involve some degree of custom design, consideration has to be given to the choice of processing elements.

General purpose processors, like members of the Pentium or PowerPC families, are complex devices which are not easy for research groups to incorporate into in-house designs. They also tend to be power hungry, making them less suitable for use in high-density configurations, whilst their main selling points (e.g. memory management and large caches) are irrelevant, for example, to the operation of a ROBIN. More appropriate in this case are I/O processors like members of the i960 family or digital signal processors, which provide the data movement capability and fast handling of incoming messages and data relevant for the buffer management. Such I/O processors also have the advantages of compactness, on-chip memory, and low power dissipation.

The microprocessor market is evolving rapidly, however, due partly to the enormous demand in mobile applications, and new options will almost certainly emerge on the timescale of ATLAS final design decisions. The advantages of higher performance (to guarantee the integrity of the buffer management task, and to increase the amount of possible pre-processing) and lower power (to enable greater board densities) will have to be weighed against the effort required to gain the relevant experience with new families of devices.

Due to the high input bandwidth the ROBIN task cannot be implemented completely in software with the current generation of processors. However, it is not excluded that it will be possible with a future generation. At present dedicated hardware has to take care of this task or has to assist a processor. FPGA technology lends itself well for implementing the required functionality of the dedicated hardware. Per ROBIN, probably a single FPGA can be used for all logic and FIFOs between S-link and buffer memory.

7.9.8.1 FPGA Prospects

The first thing which comes in mind when thinking about FPGAs is "how much logic can be implemented on a single chip" (usable gate count). However this is not the only important factor when choosing a FPGA. The following aspects should be considered:

1. *Usable gate count*

Roughly spoken this defines how much logic can be implemented on chip. Gate counts of up to 250,000, are now common. At this moment 1 million gate FPGAs are available. In the near future this will be in the order of 1.5 million gates and more.

There is an exponential growth because manufactures tend to use smaller processes. A few years ago a 0.8 micron process was standard, now 0.35 micron is common (250,000 gates). At this moment 0.18 micron processes are used. Processes of 0.15 micron will become standard in the near future. This is more than a factor 2 smaller, so on the same silicon area a factor of 5 to 6 more logic can be implemented. The ever-decreasing gate size has its physical limits. This limit is not reached yet but it has to be seen if there will ever be FPGAs with more than several million gates.

2. *Technology*

There are three FPGA technologies: SRAM, anti-fuse and FLASH based devices. They do not only differ in the way they are configured (see "Configuration" paragraph below) but there are also differences in speed and chip area used. Anti-fuse based FPGAs are fast because they often use low impedance and low capacitance tungsten plugs as connections. SRAM and FLASH based FPGAs are slower

because they build their connections using "pass through transistors". Those have intrinsic higher impedance and capacitance than tungsten plugs. An SRAM cell consumes more silicon area than a simple tungsten plug. Therefore the die size (relative to the gate count) of an SRAM based FPGA is usually bigger than the die of an anti-fuse based FPGA. Note that die sizes are related to cost.

3. *Configuration*

FPGAs need to be configured. How this is done depends on the technology used.

- The SRAM based devices need to load their configuration data each time when supply voltage is applied to the FPGA. Often a serial EEPROM is used to store the configuration data. However a FPGA can be reconfigured in system (In System Programmable: ISP) by a processor in stead of an EEPROM. This makes it possible to load different configurations. During the prototyping phase of a design it is clear that reconfiguration is an advantage. Moreover, one could see an In System Programmable FPGA as a "hardware processor" which can be configured to perform a specific task,
- The anti-fuse based devices can only be programmed once (One Time Programmable: OTP) and need not to be loaded with configuration data,
- FLASH based devices do not need a configuration EEPROM but can still be reprogrammed.

4. *On chip memory*

Nowadays complete Systems are designed On a Chip (SoC). Often a complete system makes use of memory in some form: FIFO, RAM etc. Therefore most manufacturers of FPGAs implement memory on chip. The amount of memory is usually more than enough, although one shouldn't expect to buffer large amounts of data. Typically several 32 bit wide, 256 deep memories can be implemented. The amount of memory on a FPGA is growing just as fast as the amount of logic. Now, about 5 to 16 FIFOs of 32x256 can be implemented. Soon this will be about 30 or more of such FIFOs.

The possibility of simultaneous memory access via independent ports is of importance, as a true dual ported memory gives the possibility to create "real" FIFOs which can separate two different clock domains. In the past, some FPGAs had memory that could only run on a single system clock. Of course one could mimic the behaviour of a FIFO, but one could never separate two clock domains. Manufacturers of FPGAs also noticed this disadvantage, so today most FPGAs have true dual ported memory on chip.

Memory can be implemented in different ways : some manufacturers have dedicated memory blocks on chip while others build their memory out of LUTs (Look Up Tables that are the basic building blocks of a logic cell). The latter has the disadvantages that a lot of logic and routing resources are spend to build the memory, and interface it to the system (viz. read and write pointers in a FIFO application). Long connection paths also degrade the speed of the system.

5. *System speed*

Most manufacturers give excellent speed figures in their data sheets. Those speeds usually refer to counters that are implemented in an optimum way. Speeds of 200 MHz are well possible. A complete system however will not be built of optimally implemented counters alone but will contain address decoding, data-bus multiplex-

ing, state machines and so on. It is therefore more realistic to talk about 'system speed'. More properties play a role in the system speed, like design complexity, synthesis quality, user supplied constraints (timing, placement etc.), resource usage and of course device speed grade. Experience tells that 40 MHz system speed is possible nowadays quite easy, while with some effort 60 MHz can be achieved. A system speed of 80 MHz is possible at the price of considerable design effort : synthesis and fitter tools should be guided very carefully, as writing high level VHDL or Verilog is not sufficient anymore. With 80 MHz system speed the motto is "what you want, is what you probably don't get".

One other important thing to keep in mind is that as long as signals are on chip there are no problems, but somehow signals should go to the "outer world". This usually takes a very long time with respect to the propagation delays on chip. Most manufacturers have implemented flip-flops in their IO-cells in order to minimize setup and hold times for external signals. Typical setup and hold times are about 3 ns and 0 ns respectively. When it is not possible in the design to have a pipeline stage in the IO-cell then problems easily arise. Setup and hold times can grow to 30 ns and more! This occurs if one tries to multiplex a lot of registers on a data bus. Another aspect is the time it takes for an IO-pin to go tri-state.

In the future the on chip speed will probably increase further. This will make it easier to achieve the desired system speed. Propagation delays for signals on and off chip will not keep pace with the increasing on chip speed. Therefore overall system speed will increase less, although the higher on chip speed makes it easier to implement the design.

Another item that is of importance for the system speed is the basic architecture of a logic cell as is explained below.

6. Basic architecture of the logic cells

The basic architecture of a logic cell plays an important role in the System speed as well. Most FPGAs use Look Up Tables (LUTs) which can implement any logic function of four inputs. A complex function has to be broken into a chain of LUTs, which can add up propagation delay and degrade system speed. Note that it is important how many inputs a LUT has. With 4 inputs on a LUT, to implement a function with 10 inputs at least 3 LUTs and 2 delays are necessary. With 3 input LUTs this becomes 5 LUTs and 3 delays!

Some manufacturers have added special inputs and outputs to their logic cells (Carry and Cascade) in order to speed up counters, adders, etc. However, a big state-machine still has to be decoded over several stages of LUTs. A solution for this problem is to add product term logic to the FPGA. A product term makes it possible to build a logic function with many inputs (as with the older Simple Programmable Logic Devices).

7. Number of IO pins

Most packages used today, are either Quad Flat Pack (QFP) or Ball Grid Arrays (BGA). The QFP packages can support up to 240 pins. Such a package leaves some 180 IO pins (75%) for the user. When Fine Line Ball Grid Arrays (BGA) are used, 672 pins are possible which leaves about 470 IO pins (70%). When using BGA packages, board layout can be more complicated because of the high density of the routing to the inner rows of pins on the package. Using buried vias could be helpful. When using a BGA, one has to be aware that signals can not be probed anymore, so a good strategy for testing is a must. Connectivity tests could be done using the

JTAG interface, present on most FPGAs today. The JTAG interface could be used for system test purposes as well, although this proves to be a tedious task. Modifying a board with BGA packages during prototyping is almost impossible. On the other hand, BGA packages are small and thin.

In the near future it is not foreseen that packages will become denser. If there is a need for even more IO pins then the trend will probably be to bond naked chips on carrier modules (Multiple Chip Modules: MCM). On the other hand if designs become System On a Chip then more IO pins in the future may not be necessary.

8. *Supply voltage, chip process, power, number of metal layers,*

As described above in the "usable gate count" paragraph the process in which the chip is fabricated determines the amount of logic on the chip. Also the supply voltage depends on the process. Most FPGAs now use 3V3 supply voltage (0.35 micron). At this moment there are FPGAs available that use 1.8 V supply voltage (0.18 micron). Supply voltage may drop even further when going to 0.15 micron processes and smaller. Power consumption of the FPGAs will go down as well although the decrease in power consumption will not be linear with the decrease in supply voltage.

It is a trend to split the core supply voltage and the supply voltage to the IO pins. Many devices are tolerant for higher IO voltages and support a range of electrical standards on the IO pins like LVTTTL, LVCMOS, LVDS, 3V3 PCI, etc.

Manufacturers tend to use more metal layers. This influences the connectivity between the logic cells. Currently 5 metal layers are common. In the near future 6 metal layers will be used to increase connectivity. Experience learns that present devices can be filled up to about 60% without problems. When more logic is put into a single device then routing is becoming more difficult and the time to "fit" the design is growing rapidly. Speed is usually degraded as well because the fitter has to create longer paths in order to provide the requested connectivity with the limited routing resources.

9. *Place & route software*

With the increase in gate count and design complexity, the job for FPGA place and route software will become harder. For a 50,000 gate design, processing for place and route can already take a few hours. Therefore the trend for place and route software is to partially place and route the design. This can shorten place and route time significantly, so that e.g. rerouting of a whole design because a single gate was changed is a smaller burden.

It will become a necessity to carefully floor-plan the design. Place and route software should be capable to do so. Floor-planning and partially place and route make it possible to work with multiple designers on one design, since it is becoming a difficult task for a single designer to fill a multi-million gate FPGA.

10. *Radiation hardness*

Radiation hardness is a difficult issue because there are several parameters that influence the performance such as the kind of radiation, the energy of the radiation, the chip structure (process), etc. Studies are being done to learn more about the effects of radiation on FPGAs. In general, FPGA devices that use anti-fuse technology are to be preferred since radiation can not alter the configuration of the FPGA while an SRAM bit can flip over easily. Furthermore the effect of radiation is smaller when the size of the transistors on the chip is decreased.

11. Pricing

Since FPGAs are becoming Systems On a Chip one shouldn't expect prices to be low. A high-end 1 million gate FPGA today can have a price of a few thousand USD. Low end FPGAs (10,000 gates) will cost only a few USD. Prices highly depend on gate count, since this is related to die size.

12. Conclusion

In view of a ROB design it is well possible today to implement at least a part of a ROBIn in an FPGA. Especially the combination of an FPGA and a processor seem to be very powerful and makes the design more flexible. Such a combination makes it possible to split the relatively simple high speed jobs from the slower intelligent ones (like e.g. bookkeeping tasks). It may be an option to include a processor core as a piece of Intellectual Property (IP) in the FPGA so a separate processor is not necessary. Although IP is a fast growing market there is doubt that processor cores with properties such as necessary (routing large amounts of data high speed) will be available in the future. A separate processor for the ROBIn may therefore be attractive in the future as well.

FPGA devices today already have enough usable gates and on chip memory to implement the high-speed input part of a ROBIn. However now and in the future it will not be possible to implement the rather large ROB buffer memory on chip. Separate memory has to be used. With some effort the high-speed input part of the ROBIn design can be implemented considering a system speed of 40 MHz. In the future it will become easier because of the increase in on chip speed. The speed of interfacing to the outer world (IO-pins) will not increase as much as on chip speed. Therefore system speed for future ROBIns will probably increase less than on-chip speed would suggest. Because gate count is growing exponential it may be an option to put more high-speed input parts of different ROBIns together on one chip. In this case the number of IO pins may become the limiting factor.

A 3V3 supply voltage is commonly used nowadays. As the processes get smaller, the supply voltage and power consumption will decrease. It is not necessary for the ROB design to be radiation tolerant so SRAM based FPGAs may be used. Anti-fuse or FLASH devices could be used when board space is critical.

7.9.9 Memory

When there is a need for fast memories then Synchronous Dynamic RAM (SDRAM) and Synchronous Static RAM (SyncSRAM) are most commonly used today. Other types of memory may become relevant in the future (e.g. that used in level 2 cache memories). The type of technology used for the buffer memory determines the maximum size.

7.9.9.1 Synchronous Dynamic RAM

SDRAM devices have sizes from 16 Mbit to 128 Mbit and are cheap, but they demand bursting to reach high bandwidths. They need several clockcycles between the start of a burst and the first data transfer. Following data transfers need only 1 clockcycle. The maximum clock frequency is about 160 MHz, which will go up to beyond 200 MHz in the future. New types of SDRAM will support even higher bandwidths, for instance by transferring data on both edges of a clockcycle (Double Data Rate devices). With SDRAMs a large buffer in a ROBIn is easily implemented and cheap. But because bursts are necessary for high bandwidths, extra FIFOs

might be needed to temporarily store data when the memory has to be read or written simultaneously. To manage such memories is then more complex and needs more logic.

7.9.9.2 Synchronous Static RAM

SyncSRAMs are 1 Mbit to 8 Mbit in size and available for frequencies to 200 MHz. The delays are shorter than with SDRAMs: 1 or 2 clocks to start a burst. Newer types of synchronous SRAM, like Zero Bus Turnaround (ZBT) SRAM and Quad Data Rate (QDR) SRAM, allow for higher bandwidths, 100% bus utilization and intermixed reads and writes. This will simplify the control of the data. A disadvantage is that large buffers need more devices and thus boardspace.

The SyncBurst SRAMs are optimised for long bursts of reads or writes and can operate till clockfrequencies of 150 MHz. The disadvantage of these memories is that they require an idle cycle to turn the databus around in applications with intermixed reads and writes. In this case the memory data-bandwidth can drop with 50%. Zero Bus Turnaround (ZBT) SRAM was developed to eliminate the idle cycles in applications with intermixed reads and writes (like in telecommunications systems). These memories provide 100% bus utilization and have a common databus which is turned around in 1 buscycle. Currently ZBT-SRAM is available in sizes of 4 Mbit to 8 Mbit with a defined roadmap till 64 Mbit in the future. ZBT-SRAM is available in either a Flow-Trough or a Pipelined version. The maximum frequency is around 150 MHz for the Flow-Trough and 200 MHz for the Pipelined ZBT-SRAM. A problem when using ZBT-SRAMs is that they have very short data-enable times. This requires connected devices like a FPGA to disable the I/O pins very fast to prevent bus contention. At the moment most FPGAs are not capable to disable the I/O pins fast enough to completely prevent bus contention.

To reduce the bus contention problem Smart ZBT-SRAMs are developed. Smart ZBT-SRAMs have a data-enable time which depends on the frequency of the clock, but the data-valid time is fixed. In systems with lower clockfrequencies the Smart ZBT-SRAM will have a longer data-enable time to give a connected FPGA more time to disable its I/O pins.

7.9.9.3 Further discussion

ZBT-SRAM is well suited for use as a buffer memory in the ROBIN. Data from the ReadOut Link can be continuously written in the buffer, while intermixed data can be read from the buffer to be send to either the LVL2 or the EF. High clockfrequencies and 100% bus utilization are feasible.

Of interest for the buffer memory of the ROBIN are Quad Data Rate (QDR) SRAMs. These devices will become available in the near future. QDR-SRAMs have a separate input and output databus to eliminate bus contention between the memory and a FPGA. Data is transferred on both edges of the clock and reads and writes can occur simultaneously on the input and output databus. The clockfrequency will go to 250 MHz and the size to 64 Mbit.

In general the speed of currently available SRAM or SRAM-like components seems already to be adequate for a ROBIN (writing 32-bit words at up to 40 MHz, plus reading at some fraction of this rate), but the amount of memory per chip is on the low side.

As an example, the Atlantis system uses Synchronous SRAM, which is available in densities up to 9Mbit (512k*18) at a price of ~80 CHF. Its maximum operating frequency is 100-150 MHz, depending on the grade. For the same price one could buy ~64Mbyte of SDRAM, but it would need more effort to control. A good choice might be to use a true dual-port memory right at the link input (for non-blocking, high-speed input) and to store fragments in a larger SDRAM buffer.

7.9.10 Buffer Management

Four different schemes for buffer management have been studied:

1. Circular buffer with overwriting of fragments when new data needs to be stored at the place of a fragment arrived earlier that has not yet been discarded,
2. Circular buffer in combination with overflow buffer, managed by a programmable processor and with dedicated hardware supplying the start address, length and event id of each fragment in the buffer memory to the processor.
3. Paged buffer with numbers of pages used communicated to processor by dedicated hardware and numbers of pages released communicated by the processor to the dedicated hardware,
4. Paged buffer with numbers of pages used communicated to processor, in combination with maintaining linked lists in the buffer memory of free and of used pages.

Variant 1) does require larger buffer memories than the other possibilities and requires a guaranteed maximum LVL2 decision time to prevent overwriting of event data. This variant therefore is not acceptable. Possibility 2) & 3) would both work as is shown by the performance measurement results available; 4) is a variant of 3) that allows more efficient use of memory. The final choice from the last three possibilities can be left for the detailed design.

An FPGA implementation of the ROBIN without programmable processor could make use of the strategy of variant 2 (circular buffer with overflow management).

7.9.11 Internal communication

In most of the scenarios in Section 8.0, "Scenarios for a full system," on page 62 communication between ROBIN and ROB Controller is over PCI bus. The most obvious exception is the use of DSP links in the SHARC scenarios. Some scenarios also include the possibility of "sideways" communication between neighbouring buffers e.g. use of the Atlantis private bus or the SHARC inter-processor bus, or even the pairs of buffers with a common FPGA in the Atlantis FPGA-based ROBIN.

Potential advantages of separate local communication between buffers are that collection of fragments can be done locally thus reducing the number of requests to be issued and leading to longer blocks transferred per request (improving utilization of the PCI bus) and that a larger fraction of the RoI can be preprocessed internally (reducing the amount of data to be transferred per event).

Clearly the use of proprietary point-to-point links or buses may be an available design option if a particular processing technology is chosen for other reasons. Judged on their own the viability of point-to-point links is relatively easy to predict based on their bandwidth specifications. For PCI, as a representative bus, it remains to be established whether it can carry the simultaneous traffic for a ROB complex of the required size. Studies so far indicate that PCI performance is very dependent on

the particular bridges used and their parameter settings. In the proper circumstances, however, the theoretical bandwidth can be closely approached. This bandwidth itself is likely to increase as long as PCI remains of mainstream commercial interest, as is seen in the coming new 66 MHz x 64 bit standard.

Of course PCI is used because it is a convenient standard prototyping element, which has sufficiently high performance to be taken seriously as an actual candidate for the final system. However, the DAQ-1 team in particular have studied other bus options and further technology watching of buses is advisable over the next development phase. It should be noted that other buses already come into play in the existing scenarios in the form of the local buses of the various processors. These buses generally only affect traffic local to a particular ROB complex component and their performance has to be assessed in association with the connected cpu and memory.

7.9.12 Network connection

The physical network connection of a ROB Complex obviously depends on the kind of networking which is being used and on the internal architecture. The model currently adopted for prototyping work in LVL2 assumes that the connection to the LVL2 is via a separate module, the ROBOut, and that the LVL2 network protocol itself is either ATM or Ethernet. DAQ prototyping assumes the connection to the Event Filter is via an Event Builder module across an appropriate backplane bus.

Issues which arise include:

- Is data buffered in the network interface?
- Is the network addressing information known only to the interface (e.g. ROBOut)?
- Does the interface need to distinguish between different priorities for different categories of network data?
- In the case that a ROB Complex might include more than one network interface, is any one external component (e.g. LVL2 feature extractor) always associated with the same interface?
- What is the optimum number of network interfaces for a ROB Complex?
- What are the estimated data rates through the interface?

These questions will need to be addressed in the next phase of development, partly through further work in the areas of networking, reference software and application testbeds.

If data has to be passed from the ROBIns to the ROB Controller via the same PCI bus as is used for passing the data to the network interface overall performance will drop. Transferring data directly from one PCI device to another PCI device (the network interface) under control of the ROB Controller is possible as was shown by the Saclay group.

7.9.13 Commercial availability

It is too early to draw conclusions about the commercial availability of ROB Complex implementations, but the following is useful as a checklist of factors which should eventually be weighed when choosing a supply source:

1. Availability of documentation
2. Legal responsibility
3. ISO-9000 assurance
4. Continuity of support
5. Conformance with COTS standards
6. Maintenance
7. Availability of second sources
8. Upgrade path
9. Cost
10. Compactness
11. Design compromises

It is highly likely that the balance of these factors would favour a commercial source where one is available. However, it is important, especially when considering long-term maintainability (Section 7.9.16, “Maintainability,” on page 54), to distinguish between items produced in small quantities by specialist suppliers and true COTS (Commercial Off-The-Shelf) products manufactured industry-wide on a large scale with alternative sources. Looking at the current ROBIN choices, for example, there are arguably no true COTS options, whilst ROBOuts and ROB Controllers look as though they might well be based on COTS products.

7.9.14 Cost

Low cost is important, but should be weighed against performance, reliability and maintainability. FPGA and memory prices have dropped dramatically during the past years and this process will probably continue, at least as far as FPGAs and SDRAM are concerned. For high speed synchronous SRAM (or ZBT) prices did not change so much. It is therefore too early to estimate costs reliably. A costing exercise should be conducted as part of the next development phase.

7.9.15 Reliability

Reliability should be excellent, therefore all components in a design should operate well within the specifications, electrically, thermally as well as mechanically; a key component of reliability is clean electrical power (220V) and strong cooling.

Precautions need to be taken against the possibility to make wrong connections after exchanging a board.

For programmable devices (FPGAs, processors or whatsoever) a downloadable self-test should be mandatory, JTAG-Boundary Scan capability should be highly recommended (if not mandatory) (see Section 7.11, “Testing,” on page 59).

If programmable processors are used in a ROB Complex the code to be executed has to be downloaded in the processors. Downloading from on-board ROMs or flash memories, once a stable version of the software is available, may be of advantage in view of stand-alone use for testing purposes and in view of easy repairs (e.g. by hot swapping of boards with all necessary software downloading and initialization done after power-up). With all software in an on-board ROM in the case of hot-swapping the rest of the system of course still has to be notified that the faulty part has been replaced, after which the system has to reconfigure and to supply any set-up data required to the card replaced (see also the discussion on initialisation in Section 7.8.1, “Run Control,” on page 31).

Hot swapping

On the face of it hot-swappability seems highly desirable for a large, complex system like ATLAS. However, the implementation and use of such a feature looks very uncertain, and cannot currently be counted on.

For implementing hot swapping a new set of knowledge is needed that we in the physics world currently don't have. The question is whether there is enough time to let all engineers learn it; and the risk would be that some parts are built using hot-swap capabilities (which adds cost), while other parts don't have it. This could lead to a situation in which hot-swapping is not used as it doesn't always work.

Even the commercial standards are not complete or have different implementations. cPCI is one of the few standards that are implementing it, but even there it is not obvious which way to implement it. It is especially in the software area that even in standards such as cPCI not all problems have been solved. If well-organised commercial groups can't get the standard working after four years effort, it is likely that hardware and software engineers in the high-energy physics community also will need quite some time. So even if the hardware is made hot-swap capable, it still may be that nobody uses it. After all, what is the use of being able to hot-swap the cards while it is still needed to type in commands at a terminal? Also, what does it help that one does not have to turn power off to replace a board if one does have to reboot anyway because the operating system crashed due to the board failure? Without an OS able to safely recover from board failures (e.g. PCI bus timeout) hot-swapping is useless.

Before deciding to design hot-swappable boards, calculations should be made of downtime caused by having non-swappable boards, and the costs (training, extra or special components etc.) and risks assessed.

7.9.16 Maintainability

Three issues have to be distinguished with respect to maintainability :

1. Accessibility of boards, ease with which a board can be replaced, possibility of hot swapping,
2. Hardware repairs, for which spare parts may be needed that may be obsolete at the time of repair and for which a specific expertise is needed,
3. Required changes in functionality, for which reprogramming FPGAs or modification of software is necessary. Again specific expertise is needed, as well as tools that may depend on technology that has become obsolete by the time the tools need to be used (e.g. an FPGA programmer that only runs under a particular operating system).

Producing enough spare boards can help prevent problems with respect to the first point. The second point is a matter of a good design (mechanically, and also electrically if hot swapping is required). For preventing problems with respect to the third point computers have to be reserved exclusively for software development and /or FPGA program development and maintained ready for use at the required time (i.e. complete with their own spares!).

In general, the fewer special-purpose pieces of hardware are used, the better. Replacing a custom-made ROBIN if parts are no longer available is likely to be

more difficult than replacing a commercial multi-purpose board (with a broader range of applications) with a compatible or equivalent one.

7.10 SOFTWARE

This section describes studies mainly undertaken within the LVL2 Pilot Project, though some reference is made to the DAQ-1 Project and associated publications.

Overall it is thought to be premature to try to produce a definitive design at this stage i.e. before the architectural and hardware design choices have been narrowed down. Some requirements-oriented design analysis has been started, however, and example APIs have been defined which give a feel for the kind of software that will ultimately be needed. Meanwhile software has been produced for the specific ROB prototypes under study. In the case of the DAQ-1 prototype such software comes complete with high-level design descriptions and API definitions. For the LVL2 Pilot Project studies APIs have been produced both for the operation of a ROB complex within the Reference Software framework (“External API” below) and for the internal operation of the ROB complex components (“Internal API”). The External API has been directly implemented in the Reference Software, whilst the Internal API evolved from existing software for specific LVL2 prototypes and has been used as a basis for the software for newer prototype designs.

7.10.1 Software design analysis

A UML-based design analysis is presented in [1]. This analysis starts from the functional requirements of a ROB complex and incorporates a generalisation of the design concepts followed in different prototyping studies.

7.10.2 Application Program Interfaces (APIs)

7.10.2.1 External API

An API has been defined for the external use of a ROB in the context of the LVL2 Pilot Project Reference Software, based on treating the ROB as an object [11]. It is described briefly here:

The ROB’s main task is to answer external data requests:

- Request Data is used to collect RoIs or event fragments according to the content of the request message. The answer to Request Data is asynchronous. The ROB sends later an AcceptRobData function to the requester with the data included.

Another function handles the purging of data:

- Delete Fragment is used to release events from the ROB memory. Usually a list of event identifiers is used to decrease the number of messages.

Functions are provided to manage the ROB:

- creator and destructor,
- parameter initialisation and retrieval,
- request for internal statistics.

The main attributes of the ROB are:

- an identifier,
- an operational mode,
- its eta/phi coverage.

The request data message includes the event identifier and the eta/phi range. If only the event identifier is provided the whole set of data is returned. The statistics include the content of internal counters like the count of fragments received and deleted, and the number of messages received and handled.

For test purposes or to run in a context with no external data generator a ROB emulation is used, with data being preloaded into the ROB memory. To deal with specific implementations, configuration items are added, such as the buffer arrangement.

7.10.2.2 Internal APIs

A draft document by G. Crone and M. Huet is available [12] describing a proposal for internal APIs. This was originally intended as a possible common basis for prototype development; in the event it hasn't been strictly adhered to in the prototypes, but it nevertheless represents the common spirit of the various implementations.

Note: the internal APIs do not specifically address issues such as pre-processing, or whether there are separate connections from the ROBOut to LVL2 and DAQ.

The internal APIs are based on the assumption that the ROB Complex is made of three main components:

- a ROBIN that receives event fragments from the ROD and stores them,
- a ROBOut that connects the ROB to the L2 and DAQ systems,
- a ROB Controller that manages the ROB behaviour.

The ROBIN API is closely analogous to the ROB API (see Section 7.10.2.1, "External API," on page 55 above).

The ROB-ROBOut API deals with the reception and the transmission of messages. The receiving part keep the requests in two lists, one for the data request and one for the delete fragment request. On internal request it returns next in one of both lists. A send function transmits messages on the connected network.

7.10.2.3 DAQ-1 APIs

See references [21] and [22].

7.10.3 Prototype software

(See also buffer management: Section 7.9.10, "Buffer Management," on page 51).

The various software implementations actually used in current Pilot Project prototypes broadly follow the approach of the Internal API described above, but have been tailored for the system performance studies that are of main interest. Software aspects that need to be considered are listed below, together with comments on current prototype implementations and implications for future development.

7.10.3.1 Operating systems:

The general decision has been to use an operating system on the ROB Controller, but not on the ROBIns or ROBOuts. This is a natural choice arising from the specialised nature of the current ROBIn and ROBOut processors on the one hand, and the relative complexity of the controller task on the other. The situation could change in future development if processors that are more general purpose are used on the ROBIns or ROBOuts, or if preprocessing introduces greater complexity into the ROBIn task.

7.10.3.2 Host (ROB Controller) software:

Although there is, in general, an operating system present on the ROB Controller, this does not necessarily impact greatly on the overall efficiency of the system. More important is how the operating system facilities are used, and what overhead is incurred by the use of drivers and interrupts. Some designers have preferred to use a simple high-priority polling loop for the main controller task, whilst others have organised their software around the use of threads. Yet another prototype assumes a fully-fledged operating system on the host and embeds some of the inter-component functionality into a custom-written driver. Use of operating system features such as drivers and threads might be advantageous if the ROB Complex incorporates a multi-CPU controller and/or multiple communication busses, as proposed in the Active ROB Complex[6] scenarios. Otherwise it may be more efficient to avoid operating system overheads in the main controlling task.

7.10.3.3 Layered software structure:

Operation of a ROB Complex requires communication between its internal components, and all the prototype implementations are accordingly structured into at least two layers: one at application level and a lower one at communication level. One prototype also defines hardware and platform “abstraction” layers which aid the portability of the higher level code. Whether or not it is worthwhile to incorporate such abstraction layers into the final software will depend upon future design and implementation decisions. The overhead of layering will have to be considered against the likelihood of platform modifications and in light of the performance levels of the chosen components.

7.10.3.4 Application APIs:

The application-level APIs that define the interaction between the ROB Controller and ROBIns or ROBOuts provide a simple set of basic functions, just as in the generic proposal in (Section 7.10.2.1, “External API,” on page 55). For the present prototypes the APIs have been defined and implemented procedurally, with high level code in C rather than C++.

7.10.3.5 Message-passing protocol:

Each prototype uses some form of simple message-passing protocol for communication between ROB Complex modules. These involve, in general, a command list, one or more command buffers (queues), and a signalling mechanism. They are based either on the use of shared memory accessible between components, or, in the case of point-to-point links, on communication FIFOs on the links. A couple of the PCI-based prototypes take care to minimise bus traffic by arranging for polling

flags to be written across the bus, but polled locally. For the bus-based shared-memory designs the command queues are generally organised in circular buffers.

7.10.3.6 Polling and interrupts:

One important choice that needs to be made is between polling and interrupt fielding. Interrupts are advantageous when different inputs need to be handled at different priorities. For the most part this is not the case, at least for the main dataflow messages, and designers have generally opted for polling in order to avoid context switching overheads.

7.10.3.7 Request blocking:

A related issue is whether requests issued by the ROB Controller are blocking or non-blocking. Blocking requests can become an important overhead when several components, ROBIns for example, are being accessed essentially independently. Most of the prototypes have consequently adopted a non-blocking request mechanism.

7.10.3.8 Use of DMA:

DMA is used for moving the fragment data itself and, in some cases, for other transfers as well. Features that have been used to increase efficiency are “chained DMAs”, where one DMA automatically triggers the next on finishing, and “wrapped DMA” where a circular buffer is doubly mapped to permit a single DMA to run across the wrapping point without interruption. Various techniques have been used, too, to handle the end of a DMA transfer. Essentially a choice has to be made again between polling and interrupting.

7.10.3.9 Event management:

In any implementation event fragments must be identified and tracked. All the prototypes maintain some form of hash table referring to event descriptors which contain identification and location information for the fragments. In general the hashing schemes are simple, based, for example, on the lower significant bits of the LVL1 ID. Since the latency distribution of events in the system is likely to have a long tail, allowance must be made in such simple hashing schemes for the possibility of duplicates. Current approaches do this by the employment of linked lists.

7.10.3.10 Object-oriented design and coding:

So far all internal software has been designed and written procedurally, largely in C. The current (external) reference software, on the other hand, follows an object-oriented design and is coded in C++. The two domains clearly have to interact smoothly. Whether or not it will make sense to change the approach in one of these two domains will depend on future design decisions together with compiler availability and associated hardware and software performance. The motivation for an OO approach to the internal ROB Complex software will presumably be consistency with other system software, assuming that adequate performance can be attained, but it is not clear that this will warrant abandoning the simple efficiency of the current prototype implementations.

7.11 TESTING

Testing will need to be carried out both during and immediately after production, and to continue through the lifetime of the system. Clearly it will be necessary to provide detailed testing specifications before the production phase, but it is important that the requirements for testing also be taken into account during the design phase.

Of course for in-house designs it is highly likely that a professional board manufacturer will be used for the final production (and also possibly for larger scale prototyping exercises). In this case the manufacturer will produce the boards from the PCB datafiles, buy the parts, assemble the boards and do the basic test using the test procedure supplied, guaranteeing to supply boards which pass the test. In that case it would only be necessary to define the test procedure (for all the required functions) and supply the necessary test equipment (e.g. a test data source).

The sub-sections below are provided as a starting-point for full testing specifications and as checklists for future design.

Note on JTAG

JTAG is favoured for a number of reasons and is already used in most of the current prototyping exercises. The JTAG standard defines a 4 pin Test-Access-Port (TAP) which can be connected to a variety of JTAG controllers, including commercial ones. JTAG is well documented, software can be written relatively easily and packages are available in the public-domain.

JTAG can be conveniently used as a standardised facility for programming In-System-Programmable devices. Via a simple standard cable-connection with a PC, modification of programmable logic is quick and easy and can also easily be done in the 'field'. It may well be prudent to forbid the use of programmable devices (ROMs, CPLDS, etc.) which are not in-system-programmable, whether by a standard JTAG interface or some other proprietary method.

JTAG is not suited for functional testing, see also Section 7.11.2, "Offline testing," on page 60. The reason is the long time required to set up all external signals, as every signal has to be set via a long shift register. This also causes that errors caused by timing problems or related to high system speed can not be detected. JTAG can also be used as a facility suitable for e.g. downloading software or calibration constants. It is unlikely that this will be used for the ROB.

It is not clear that there is any need to standardise on a JTAG connector, programmer or controller as long as the basic JTAG specification is followed and access is provided to the 4 magic pins. The IEEE standard itself does not specify a connector, though a standard exists for the connector which might be useful.

7.11.1 Tests during and directly after production

7.11.1.1 PCB

- **Mandatory:** electrical test against design data: before assembling a PCB, it must be electrically tested against the design data used to produce the PCB. This will assure that all connections are present without short circuits between different connections.

7.11.1.2 Assembling

- Mandatory: visual inspection after assembling of a module: check for correct device types, correct value's and correct orientation of devices,
- Probably needed: X-ray inspection of soldering process: for devices packaged in a BGA,
- Desirable: JTAG - connectivity test: testing of PCB-traces and connections to device-pins and connector-pins. Problem is that probably not all devices will support a JTAG test, making it difficult and laborious to perform a 100% connectivity test. Additional JTAG test equipment will be needed for a connectivity test of connectors. Special software is needed to generate testpatterns for the JTAG connectivity test. For testing of this type a JTAG - Standard connector for access to JTAG chain is mandatory.

7.11.2 Offline testing

7.11.2.1 Acceptance testing

- Mandatory: duration test program: extensive test of all parts of the ROB. The duration test must be able to run unattended for a relatively long time and write errors in a logging file. If there are no errors during this test, it is assumed that the ROB functions correctly.

7.11.2.2 Interactive testing

- Mandatory: interactive test program: all parts of the ROB can be tested separately with interactive control of the test parameters and test output. Optionally the test program can be put in high speed testloops for easier testing with logic analyser or oscilloscope.
- Not needed: JTAG - functional test: only a limited and laborious functional test will be possible through the use of JTAG. Therefore using JTAG for a functional test of the ROB can not be recommended. Maybe a JTAG functional test can be used to trigger the self-test function of devices, which support this feature.
- Probably not needed: JTAG - processor emulation: depending on the type of processor, JTAG could be used to debug processor operation. Special emulation software is needed for this.

7.11.2.3 Test equipment

- Logic analyser: special test connectors should be available on the PCB, which allow for quick connection of logic analyser probes to check some important signals. A problem at the moment is that these connectors are relatively large and can take up to much boardspace. A smaller high-density connector would be better.
- SLIDAS – S-link Data Source: with a SLIDAS module special test data can be sent to the ROBIn through the S-link connector, without the need for a real S-link connection. The SLIDAS can sent adjustable test patterns at full speed of the S-link.
- Bus analyser: a standard bus-system like PCI may be used. For checking the data transport via this bussystem a bus analyser will be very handy.

7.11.3 Online testing

7.11.3.1 Power-On Self-Test (POST)

- Mandatory (where possible): Power-On Self-Test: after power-on or a global reset, if possible a POST must start autonomously and perform a test of specific parts of the ROB, such as: registers, memory, bus-interface, etc. The POST should not take more than several seconds to complete, so only a limited functional test will be possible during the POST.

Of course an FPGA, or indeed any embedded processor without local ROM (i.e. downloadable code only), generally cannot perform any action unless it is configured by the host cpu. In this case the Self-Test, which should be mandatory, would have to be performed upon request (i.e. after downloading the Self-Test code).

7.11.3.2 Interactive in-system testing

- Mandatory: interactive test in ROB processor: interactive test program which can test separately the different parts of the ROB (e.g. registers, memory, bus-interface, etc.) and which gives detailed information in case of errors. This could be partly the same software as used during the POST, but the interactive test will probably do a much more extensive functional test than the POST. If the processor in the ROB can inject test data close to the S-link connector, then the flow of data through the ROB is also testable. By injecting test data containing errors the reaction of the ROB on errors in the data can be tested.

8.0 SCENARIOS FOR A FULL SYSTEM

This chapter summarises various scenarios that can be envisaged for implementing a full readout buffering system for the whole of ATLAS, based on extensions of the current development prototypes.

The summaries give a basic description and highlight the key points. The complete descriptions as submitted by their proponents are accessible via the ROB Complex Web page [19]. Implications for the implementation and architecture of a ROB Complex are to be found in Section 7.9, “Implementation,” on page 37.

The proposals intentionally span a wide range of architectures. The aim was to “brainstorm” the options in order to explore the solution space as fully as possible.

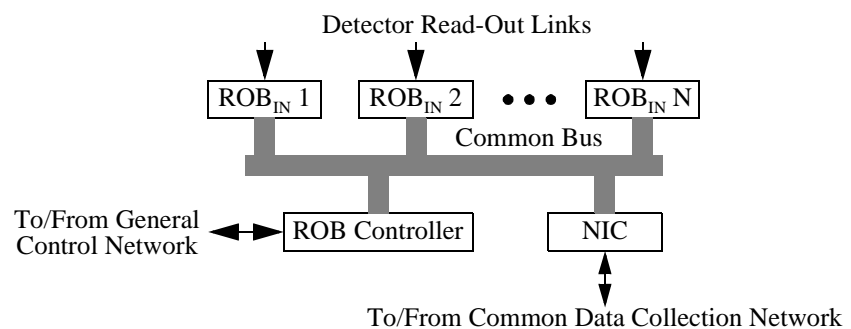
8.1 MODULAR ROB COMPLEX SCENARIO (SACLAY) [15]

This scenario is based on a multi-buffer ROB Complex, i.e. several input buffers sharing a network interface. A key aim of the scenario is flexible matching of the number of buffers in a complex (grouping factor) to the requirements of each sub-detector.

A complex is assumed to be composed of several buffers (ROB_{IN}s), a control processor (ROB Controller) and a network interface (ROB_{OUT}), all communicating across a PCI bus. Initialisation, servicing network messages, etc. are all done by the ROB Controller. Data is either copied from buffer to network interface via the ROB Controller, or moved straight directly to the network interface using direct address mapping. In either case DMA (or chained DMA) can be used to help.

FIGURE 1.

ROB Complex organisation.



It is noted that the desirable degree of grouping (number of buffers per complex) depends on the output bandwidth and the request rate. The ROB Controller must deliver messages at a frequency up to the number of ROB_{IN}s times the request rate plus the number of ROB_{IN}s times the event clear rate. The former depends on whether or not all ROB_{IN}s are involved, whilst the latter assumes no broadcast.

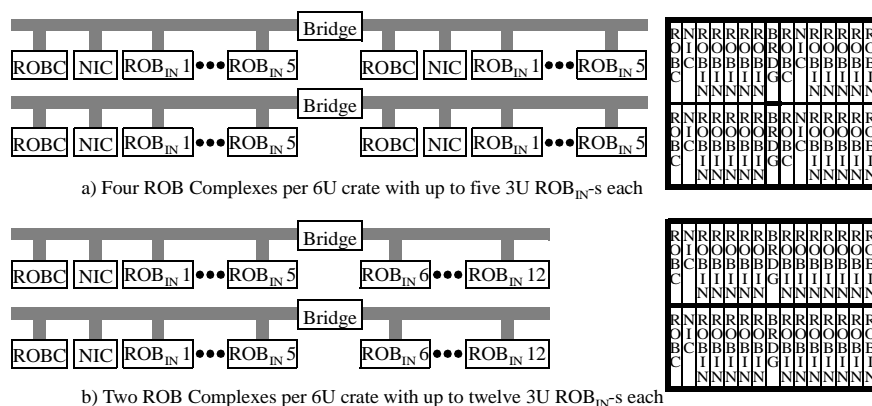
Based on results obtained from modelling (see Table 6 on page 39 - Table 11 on page 41) it is concluded that the full-scan subsystems suggest an optimal grouping factor of about four; for the calorimeters the suggested grouping factor is determined by the bandwidth of the output link and is between about four and six; whilst

the grouping factor for the muon detectors is not critical - it might depend on partitioning and full data collection bandwidth and is probably in the range twelve to sixteen.

This scenario stresses the importance of modularity in the composition of a ROB Complex, arguably a good design principle anyway for a long running experiment. Specifically CompactPCI is proposed as a choice for housing and backplane to support this modularity; its features include high bandwidth and performance, robustness, industry support, and reasonable cost. Presently there are eight slots available per cPCI bus, but these can be transparently bridged. Current options include 3U crates with 14 slots or 6U crates with 27 slots, each with a variety of mechanical support configurations.

FIGURE 2.

Some possible configurations of the ATLAS read-out units in a CompactPCI chassis.



Possibilities for a ROB Complex based on a 6U chassis are: 5 or 12 ROBINs. 4 * 6 independent ROBINs up to 26 ROBINs with transparent bridging of 4 segments

These groupings would give a possible overall ATLAS configuration as follows:

TABLE 12.

Estimate for the ATLAS read-out organization based on ROB Complex units (assumes 80 MB/s b/w on data collection links, & 3U modules in 6U chassis).

Detector Subsystem	Number of ROB _{IN} -s	Grouping Factor	Number of ROB Complexes	ROB Complexes per crate	Number of crates
Muon Precision	192	12	16	2	8
Muon Trigger	48	12	4	2	2
E.m. cal.	760	6	127	4	32
Had cal.	98	6	17	4	5
TRT	256	4	64	4	16
SCT	92	4	23	4	6
Pixels	84	4	21	4	6
Total	1530		272		75

The grand total would be 1530 buffers in 272 ROB Complexes in 75 cPCI chassis.

Note : the numbers in this section are from [15] and are slightly different from the numbers in the scenarios document accessible from the ROB Complex web page [19]: the grouping factor for the calorimeters is here 6 in stead of 5, resulting in 272 in stead of 300 ROB Complexes and 75 in stead of 81 cPCI chassis.

8.2 POINT-TO-POINT SCENARIOS (NIKHEF)

These two scenarios explore the possibilities of point-to-point linking between ROB Complex components. The first scenario is for a more “conventional” ROB Complex, with its own ROB Controller, connected to the LVL2 and DAQ systems in the generally assumed manner. The second scenario aims to avoid the overhead of dedicated ROB Controllers and network interfaces by considering direct connections to the processors of LVL2 and EF. Both scenarios have configurational alternatives.

The scenarios are developed around a SHARC DSP based ROBIN (“CRUSH”), which has been prototyped at NIKHEF for ROB Complex studies. Input is assumed to be over S-link with two link connections per S-link daughter board (via the front-panel). The particular point-to-point links considered are the 40 MB/s SHARC links, but they could be implemented differently, in an FPGA for example. The SHARC links have the advantage that communication across them is automatically handshaked. Additionally the latest, faster, SHARCs have 100 MB/s links. Use is also made, in the scenarios, of the special SHARC bus, which can transparently interconnect up to six SHARCs. Two other SHARC features that could prove useful in the ROB context are bus broadcast and a pair of 40 MB/s serial links.

It is assumed that low access ROBINs are mixed with high-access ROBINs in same ROB Complex group. This means that average request rates can be used in the modelling projections. Modelling has been done for 40 and 75 kHz LVL1 rates, assuming an LVL2 accept rate of 3.75 kHz for 75 kHz LVL1 rate. In more recent modelling studies the LVL2 accept rate is limited to 2 kHz.

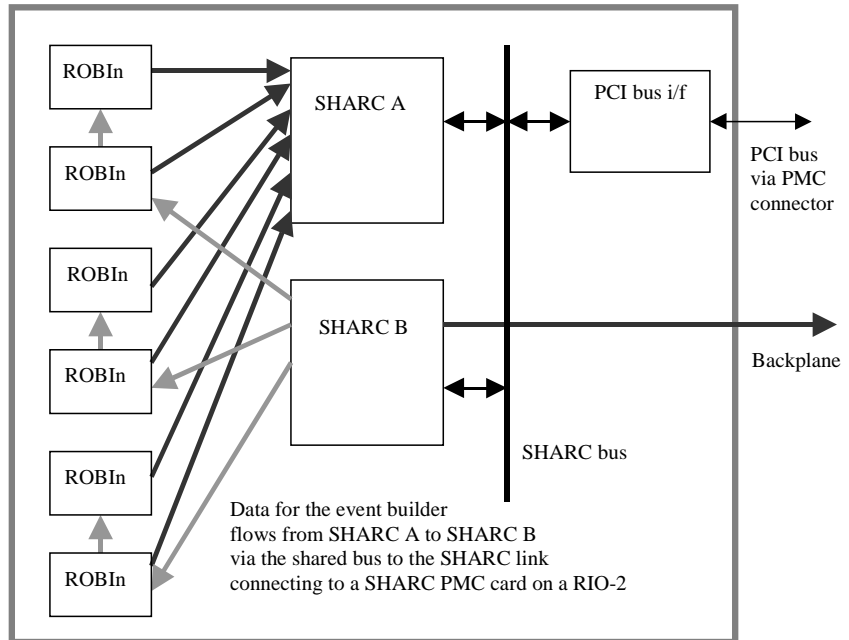
8.2.1 Scenario I

6U VME cards are used with 4-6 ROBINs per card. The ROBINs are connected via PCI to VME CPUs carrying the LVL2 and EB interfaces. Only 1 PCI interface is used per PCI segment.

Several configurations are studied, with either 80 MB/s or 15 MB/s output connections.

8.2.1.1 Configuration I

FIGURE 3. Structure of a ROB card for Scenario I, Configuration 1.



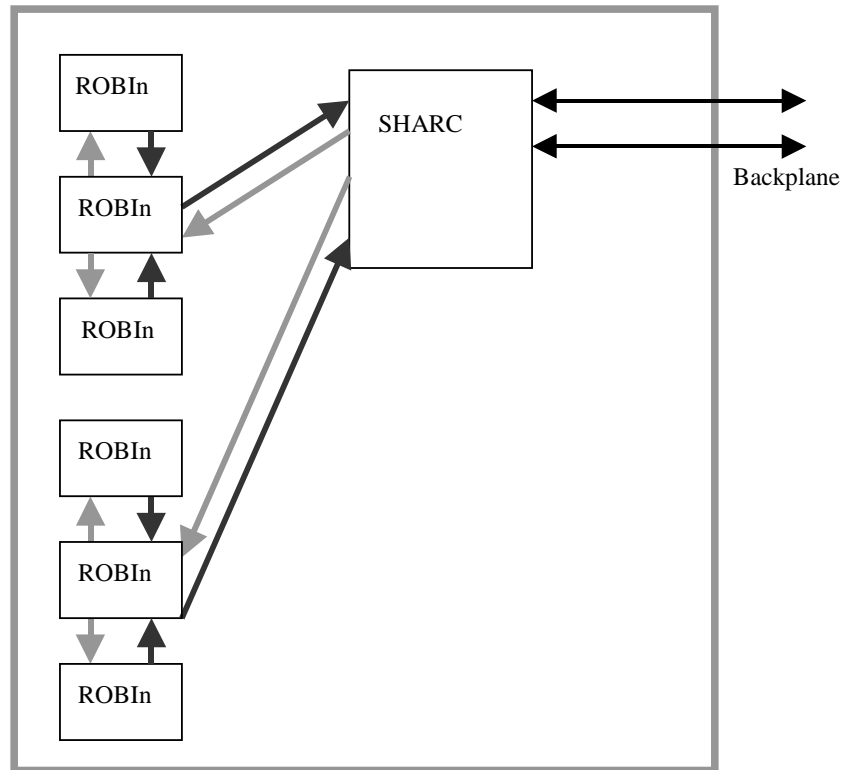
80 MB/s output, e.g. via Gigabit Ethernet, is assumed. The ROB cards each contain six SHARC-based ROBIns plus two extra SHARCs: one connecting via PCI to a ROB Controller (e.g. RIO2) which also carries a LVL2 interface, the other to a SHARC PMC on a PCI processor card via a dedicated SHARC-link backplane. This card carries a Gigabit Ethernet interface to the EB. Eight ROB cards can be connected to one such interface. The configuration can be varied -- more SHARCs would avoid communications to ROBIns through other ROBIns.

The grand total for this configuration is 43 crates max, 257 LVL2 links, 64-180 EB links. EB link utilisation can be high. PCI would bottleneck the 80 MB/s Gigabit Ethernet bandwidth down to 66 MB/s (it has been assumed that data is passed from SHARCs to the ROB Controller via PCI and then to the network interface via the same PCI bus).

8.2.1.2 Configuration 2:

FIGURE 4.

Structure of ROB card for Scenario I, Configuration 2.



In this configuration the connection to LVL2 is now similar to that for the EB, namely via the SHARC-link backplane. There is now only one SHARC per ROB card, with two SHARC links to the SHARC backplane.

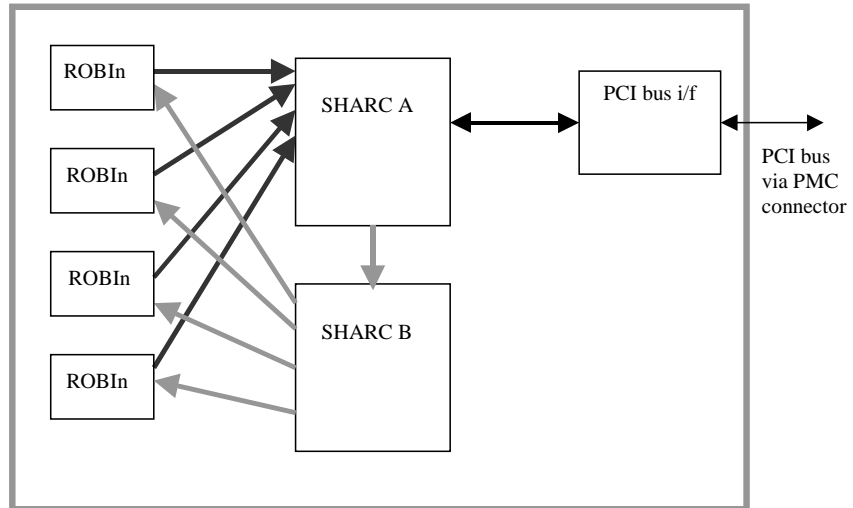
The number of crates remains essentially the same as for Configuration 2, but the number of LVL2 links is reduced to 75.

Possible drawbacks are: - high output bandwidth for TRT limits the number of ROBIns per TRT ROB card to four. - many links run at high utilization, requiring direct transfer from SHARC memory (SHARCs on the PMCs) to network interface. - the use of bidirectional links might result in inefficiencies.

8.2.1.3 Configuration 3:

FIGURE 5.

Structure of ROB card for Scenario I, Configuration 3.



In this configuration the ROB cards are interfaced to the EB via PVIC. Each ROB card has four ROBIns and a PCI interface which connects to the PCI extension connector on a RIO2. The RIO2 carries both LVL2 & PVIC interfaces. Up to eight ROB cards are interfaced to the EB via PVIC. Alternatively the LVL2 & EB interfacing methods can be interchanged. Links assumed to be ATM at 15 MB/s.

This configuration is similar to the DAQ-1 Read-Out Crate. It uses a large number of RIO2 cards and PVIC interfaces. The modelling results imply that it cannot achieve event building for a 75 kHz LVL1 rate, but the assumed event building rate of 3.75 kHz may be too pessimistic. The grand total for this configuration assuming 40 kHz LVL1 rate is 96 crates max, 281 LVL2 links and 575 EB links or 575 LVL2 links and 414 EB links.

8.2.2 Scenario II

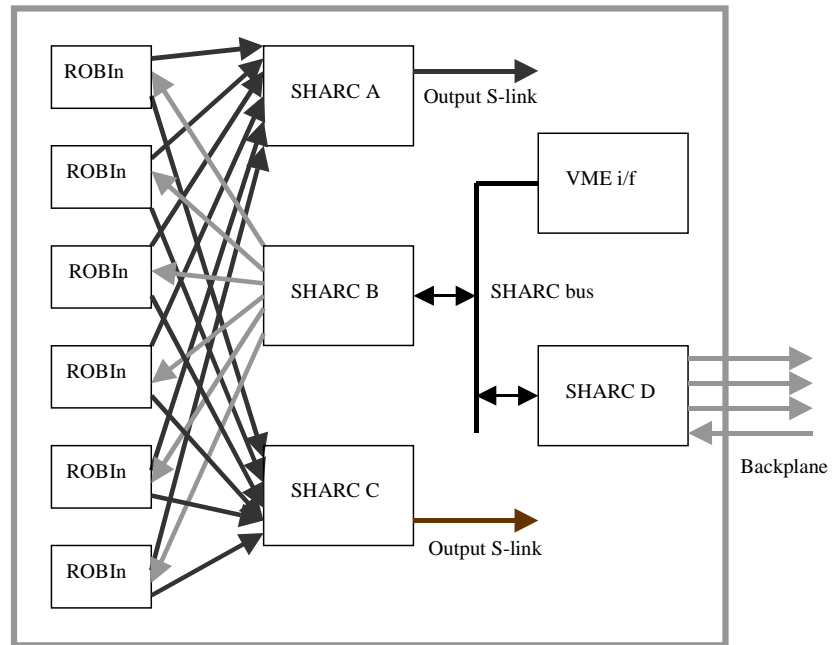
The motivation for this scenario is to avoid large numbers of RIO2s and network interfaces. SHARC links are connected to S-link outputs, which are connected to PCs on the LVL2/EB networks. Data that needs to be collected together from ROB-Ins attached to different PCs is transferred across the network that joins the PCs.

9U VME cards are assumed, with six or eight ROBIns per card. Two configurations are studied; the second provides support for full RoI fragment & full event building, but is more complex. Both configurations need less hardware than Scenario I.

8.2.2.1 Configuration 1:

FIGURE 6.

Structure of ROB card for Scenario II, Configuration 1: ROB card with 6 ROBIns and 2 S-link outputs, one for LVL2 data and the other for EB data.



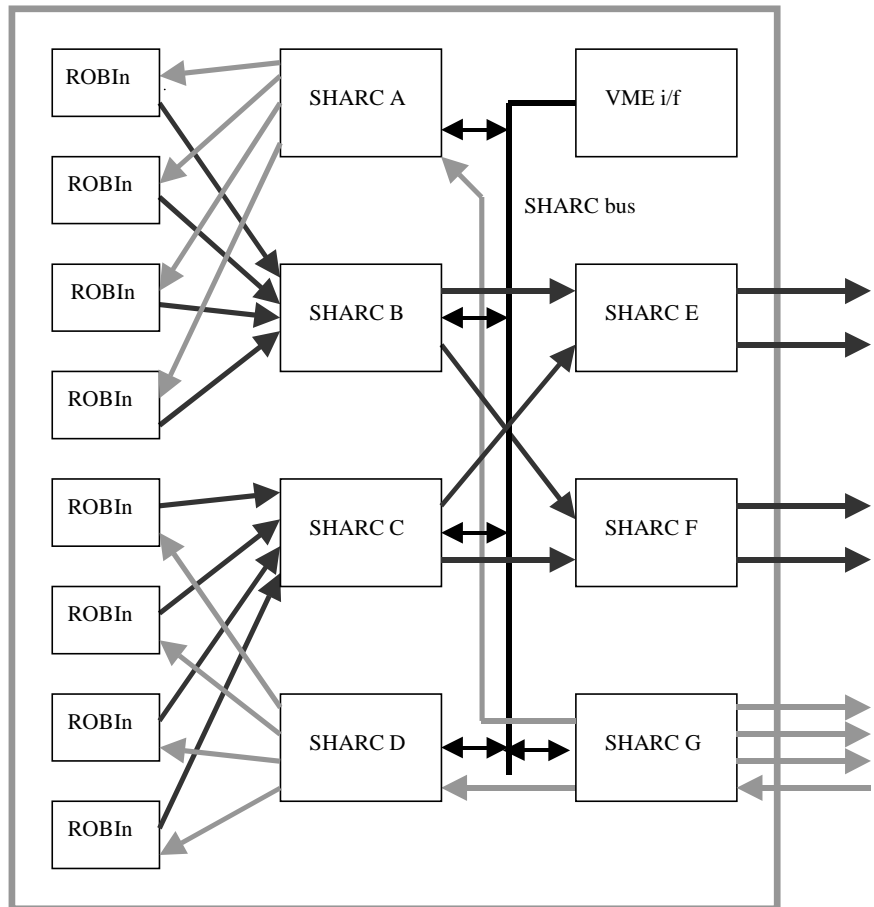
Each card is assumed to have six ROBIns and two S-link outputs (it might therefore be possible to fit this configuration on 6U cards rather than 9U). Each card also has a VME interface for auxiliary functions. Five cards are connected together via a SHARC on one of them that is in turn connected to a SHARC-link dedicated backplane. One of the five connected cards is designated the “root” and is used to distribute requests to itself and the other four cards. There are three other SHARCs on each card (apart from the ROBIns): one is used to fan out requests to ROBIns, the other 2 are use to fan in data to each of the two S-link outputs.

This configuration leads to high message rates, but this is probably ok for SHARCs. 80 MB/s S-links are assumed, leading to 514 S-links, 257 ROB cards, and 13-15 crates (depending on the quantisation of crates per sub-detector). If LVL2 AND EB data were sent through the same S-links only 257 would be needed. If eight ROBIns can be accommodated on a card, only ten crates would be needed. Mapping towers across sub-detectors, or grouping requests for several events would help reduce request rates.

8.2.2.2 Configuration 2:

FIGURE 7.

Structure of ROB crate input board for Scenario II, Configuration 2. The VME interface is a simple 32-bits slave interface. SHARC B, C, E and F build event fragments from the data received. Each SHARC link used for outputting event data to the backplane can receive data from all ROBIns.



In this configuration eight ROBIns are accommodated on a 9U card and the S-link outputs are located on separate cards; for example a crate could house 128 ROBIns together with 32 S-link outputs. Interconnection between ROBIns and S-links is assumed to be via a SHARC-link backplane (e.g. 2.56 GBytes/s on 64 links).

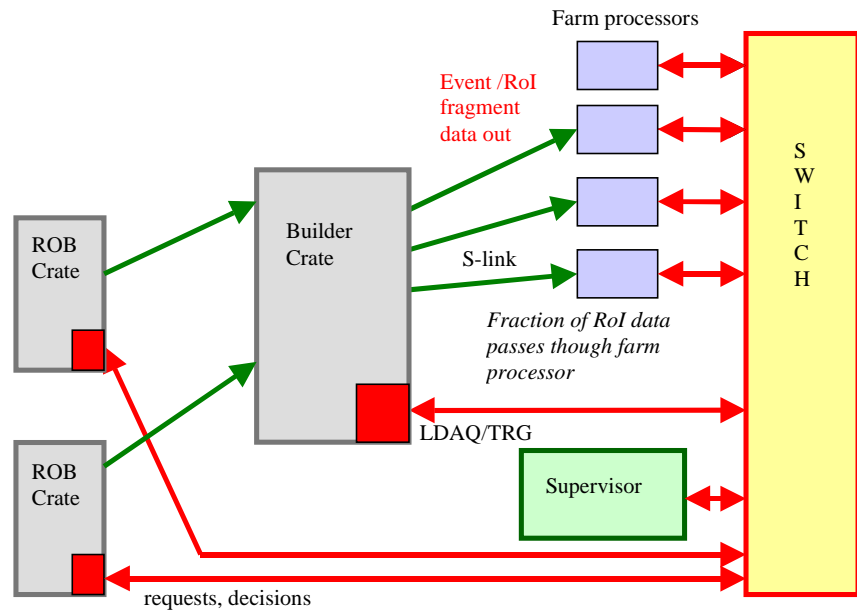
It can be seen that fragment rates are reduced due to optimisation of the number of output links used by each ROB card.

8.2.2.3 Configuration 2 extension:

An extension to Configuration 2 is to use some of the ROB crates as “builder crates” to build complete RoIs. In the full description these are called Intermediate ROBs or IROBs. Efficiency would be improved if links between ROBs and IROBs were bidirectional -- e.g. 100 MB/s SHARC links, or future high bandwidth FireWire.

FIGURE 8.

Event / RoI fragment building with "builder" crates equipped with the same hardware as the ROB crates.



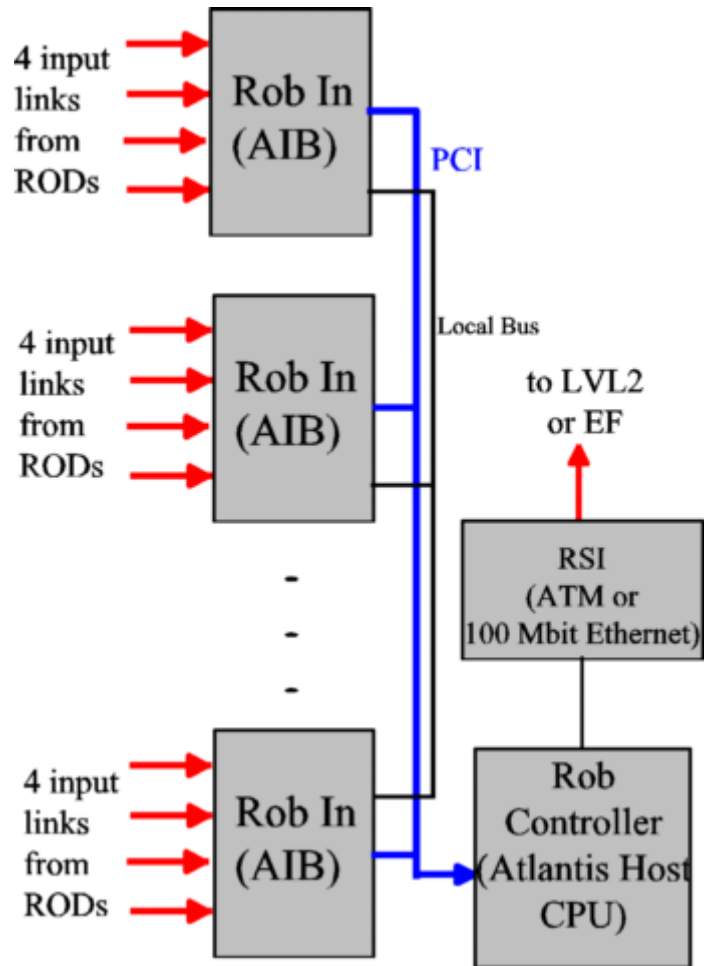
Extrapolation to newer SHARCs may reduce the total number of SHARCs, but the board architecture is determined by numbers of links, so remains unchanged.

Note that numbers can be reduced in Scenario I if the interfacing to LVL2 and EB is combined. In Scenario II this would halve the number of links, and LVL2 and EB traffic could still be split at the farm processors.

8.3 FPGA-BASED SCENARIO (MANNHEIM, WEIZMANN)

The scenario investigated by Mannheim is based on the use of FPGAs for the high-bandwidth input task. Since the FPGAs can be programmed to process data at hardware-like speeds, various selection and pre-processing tasks are assumed to be possible "on-the-fly". The scenario is constructed around a system of FPGA-based boards in a Compact PCI crate which is equipped with a special secondary private bus. Such a system is being developed for a variety of possible applications in ATLAS and has been dubbed "ATLANTIS".

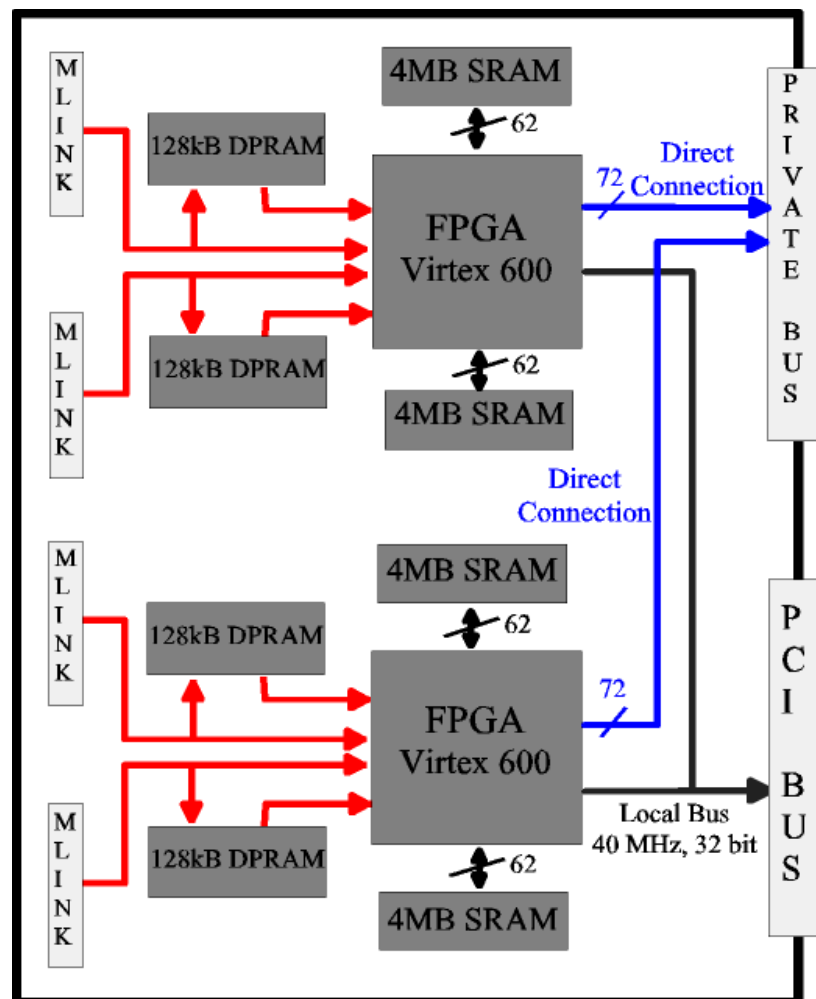
FIGURE 9. An ATLANTIS based ROB Complex



...

FIGURE 10.

A 4-ROBIn Atlantis I/O board.



The ATLANTIS system is housed in a 6U cPCI crate with up to 14 slots, of which eight are currently used. The crate is populated with 6U Atlantis I/O boards used as 4-port ROBIns, a PC processor, and an Ethernet NIC (currently fast Ethernet, but could be Gigabit Ethernet) or an ATM NIC. All I/O boards are connected by a private bus (0.5 GB/s) in addition to the PCI. The ROBIns each have four “M-link” inputs which act like LVDS electrical S-link cards, but are smaller since some of the S-link functionality is embedded in the ROBIn FPGAs. Each 4-link ROBIn contains two FPGAs and four memories (4-8 MB each).

Data from several fragments can be assembled into larger fragments by the FPGAs via a private backplane communication prior to transmission to the host (ROB Controller) across PCI. This local collection of fragments and the optional preprocessing capability of the FPGAs can significantly reduce the required bandwidth and message rate to the host. Thus relatively large ROB Complexes can be built even with today's PCI buses.

8.4 ACTIVE ROB COMPLEX SCENARIO (CERN, NIKHEF)

This scenario addresses the problem that COTS components may not be able to handle the full ATLAS event rate with all-to-all connection, primarily because of limits on the rates at which the software can handle requests and event processing. Possible solutions are:

- to optimise critical communications software,
- to scatter RoI data across buffers to reduce access frequencies,
- to add special pre-processing hardware, e.g. co-processors,
- to use an “Active ROB Complex”, i.e. do some of the farm processing job in the ROB Complex itself.

This scenario considers the last option. It is based on a local network of processors, which could be considered to be ROB Controllers, each with access to several ROBINs. An example today would be a COTS system of eight SMP processors and 20 PCI slots. A similar effect could be achieved by adding computing capacity in a ROB crate, e.g. a DAQ-1 ROC or a bus-based ROB Complex.

Potential benefits include:

- reduction of request frequencies from steering/general processors by grouping requests over several ROBINs,
- reduction of output data bandwidth by selecting RoI-only data from ROBINs,
- reduction of computing load on farm processors by pre-processing,
- possible elimination of feature extraction in the farm processors by performing feature extraction in parallel in the local processors instead.

The following tables, derived from modelling, show how the access frequencies and output bandwidths vary with varying numbers of ROBINs in an Active ROB Complex.

TABLE 13.

Numbers of ROBINs & AROBCs, access frequencies, and output bandwidths for each sub-detector (high luminosity).

Detector	#ROBIN / #AROBC	Access frequencies [kHz]				Required bandwidth [MB/s]			
		(per ROBIN)	average	upper limit	lower limit	(per ROBIN)	all data	with pre-proc.	+ Ev. Buil.
Pixels	84/12	0.7	2.8	4.9	1.1	0.57	4.1	1.0	11.5
SCT	92/12	0.4	1.8	3.1	0.7	0.75	5.8	1.4	24.7
TRT	256/24	0.3	0.9	1.1	0.8	0.26	2.8	2.2	21.7
Ecal	760/56	1.0	2.2	2.9	0.7	1.44	19.6	9.8	49.0
Hcal	98/10	0.9	2.2	3.3	0.4	1.53	15.1	7.6	35.4
MuPrec	92/16	0.2	0.9	1.0	0.8	0.12	1.5	-	19.7
MuTrig	48/ 4	0.3	3.3	3.3	3.3	0.13	1.7	-	9.7

TABLE 14.

Numbers of ROBINs & AROBCs, access frequencies, and output bandwidths for each sub-detector (low luminosity).

Detector	#ROBIN / #AROBC	Access frequencies [kHz]				Required bandwidth [MB/s]			
		(ROB In)	av.	upper limit	lower limit	(ROB In)	all data	with pre-proc.	+ Ev. Buil.
Pixels	84/12	11.0	14.4	17.9	11.8	1.2	9.1	2.3	1.6
SCT	92/12	10.6	12.8	14.9	11.0	3.0	23.4	5.8	4.3
TRT	256/24	10.3	11.4	11.7	11.2	8.1	86.5	17.3	16.7
Ecal	760/56	0.7	1.7	2.1	0.6	1.1	15.0	7.5	49.9
Hcal	98/10	1.2	2.9	4.2	0.7	2.0	19.7	9.8	36.0
MuPrec	92/16	0.3	1.8	2.0	1.6	0.2	3.0	-	20.0
MuTrig	48/ 4	0.6	6.5	6.5	6.5	0.3	3.4	-	9.9

The following table gives the projections relevant to local feature extraction:

TABLE 15.

TABLE of access frequencies, computing load estimates, and probabilities of contained ROI, for each sub-detector.

Detector	#active ROBINs/ AROBC	average frequency (hi/lo L)	computing load, high L (prepr.)	computing load, low L (prepr.)	computing load (appr) (feat.ex.)	probab.of ROI contained
Pixels	84/12	2.8 / 14.4	2.4	12.3	28 / 144	0
SCT	92/12	1.8 / 12.8	0.4	3.2		0.4
TRT	256/24	0.9 / 11.4	0.3	3.4	1 / 11	0.5
Ecal	760/56	2.2 / 1.7	0.2	0.1	0.2 / 0.2	0.1
Hcal	98/10	2.2 / 2.9	0.1	0.1	0.2 / 0.3	0.7

The main conclusions from modelling this scenario are as follows:

- the number of ports can be reduced by a factor of 11.4,
- the number of messages can be reduced correspondingly,
- the numbers shown in the tables are achievable within known technology limits (assuming 15 kHz & 20 MB/s max for LVL2),
- the ultimate bandwidth limit is set by the volume of traffic to the EB (assuming no data compaction is used for this traffic),
- for pre-processing computing capacity is not a problem,
- feature extraction in AROBCs is problematic because of the low probability that ROIs are completely contained within a likely grouping of buffers; local feature extraction cannot be ruled out at this stage, however, and may yet prove worthwhile,
- for feature extraction the computing load may be problematic, in particular for the precision trackers.

In addition to the modelling results, early tests with prototype ROBINs in a simple PC-based system have shown that:

- Interfacing COTS multi-processor systems to the LVL2 network is trivial,

- significant advantage can be taken of added I/O capacity of COTS systems e.g. multiple PCI buses.

8.5 SIMPLE-ROB SCENARIO (UCL, RHUL, CERN)

The final scenario summarised here returns to basics and considers the result of collapsing a ROB Complex into a single, simple ROB associated with a single Read-Out Link. This is the kind of configuration originally conceived for RoI processing, but attention shifted to multi-input ROB Complexes when it looked like network performance and cost might prove problematic. However recent evolution of the market in network switches and interfaces suggests that it may become cost-effective on the ATLAS timescale to put the entire communications burden onto a network and thus avoid the possibly greater complexity and reduced flexibility associated with multi-buffer ROB Complexes.

The basic assumption of the simple ROB scenario is that all normal communication and data transfers occur across the LVL2 network via a switching fabric. A secondary communications path seems desirable for auxiliary functions such as downloading, debugging, and possibly monitoring. An important aspect that needs further consideration is how simple ROB would fit in with the rest of the TDAQ system.

The network technology assumed is Ethernet (Fast or Gigabit) since this is where the market seems to be evolving most dramatically. Modelling exercises predict that up to two Fast Ethernet or a single Gigabit Ethernet connection will be needed per ReadOut Link, depending on sub-detector. The system could use a mix of Fast and Gigabit as required, or it may be better to settle on a single, standard, network interface for all connections; the choice would be between Gigabit and dual-Fast links, according to cost and availability.

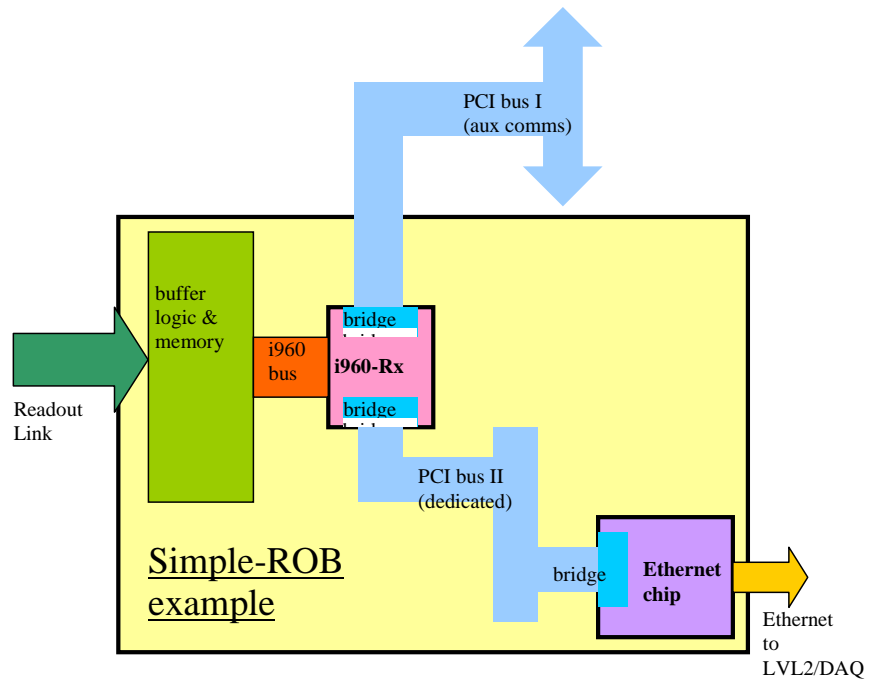
Memory and processing requirements for a simple ROB would be not much greater than for a ROBIN. A simple-ROB processor will have to incorporate some ROB-controller and ROBOut functionality i.e. decoding full RoI requests, management and network buffering (if required), but the natural evolution of ROB processor options over the relevant timescale is likely to make this unproblematic.

Simple ROBs need be only slightly bigger than ROBINs since they are basically a ROBIN plus a NIC chip. It is thus reasonable to assume that between six and twelve could be packed onto a 9U board; the limitation is likely to be the space required for S-link and Ethernet connectors.

A specific configuration example is to use an i960 family processor with two built-in PCI bridges, plus a standard Ethernet chip with one built-in PCI bridge. The processor and NIC chips would be interconnected via PCI in a completely dedicated and transparent manner, and the card itself could be connected to a PCI backplane acting as the auxiliary channel. This backplane could be Compact PCI or an alternative COTS option.

FIGURE 11.

Block diagram of i960-Rx based Simple-ROB.



Variations are possible, e.g.:

- An external ROB-controller could still be used, with access across Ethernet, but it would not be restricted to handling only ROB's within the same crate,
- The ROB could function as a ROBIN/ROBOut combo in a more conventional ROB Complex, with ROB Controller access across PCI.

9.0 PROTOTYPES: DESCRIPTION AND PERFORMANCE

Different possibilities for the ROB design were explored in four different prototype studies. This section contains a summary of the key features of the different designs and compares results of the performance measurements to the requirements as obtained from the paper model. Detailed descriptions and measurement results can be found in [13], [14], [15] and [16].

Different institutes have undertaken the different studies. In the alphabetical order implied by the institute names :

9.1 MANNHEIM & CERN GROUPS [13]

The Mannheim group has studied an exclusively FPGA based ROBIn which is implemented with the commercially available μ Enable board. This is a PCI board for use in standard PC environment. Data input is via S-link with a maximum frequency of 20 MHz. The buffer memory is built from 512 kByte SRAM that can be read out at 20 MHz while at the same time S-link data is input at 20 MHz. The memory functions as a cyclic memory. Consecutive events are stored in consecutive memory pages of 4 kByte. Measurements have been made with a PC as ROB Controller. It has also been demonstrated that pre-processing by the FPGA of data from the TRT is feasible. The CERN group studied together with the Mannheim group a variant, named "Active ROB Complex" (AROBC), see also Section 8.4 on page 73. In this system a four-processor PC server, equipped with 4 x 550 Mhz Pentium Xeon III processors on a Intel SC450NX board, 2 PCI buses of 32 bits/32 MHz with together 7 free slots and 512 MB shared memory was used as powerful ROB Controller capable of substantial preprocessing of the data. Measurements were made with 1 - 4 μ Enable boards connected to one or (for 3 or 4 boards) two busses. The Mannheim group is continuing their studies with the CompactPCI based ATLANTIS system. This system allows improved functionality, in particular with respect to the handling of memory fragmentation, and improved performance of the ROBIns.

9.2 NIKHEF GROUP [14]

The NIKHEF group has studied the design of a ROBIn system built from digital signal processors (the ADSP-21060 SHARC from Analog Devices). Each SHARC DSP has 6 communication links, which are used for interconnecting the different parts of the ROB complex. Two ROBIns were constructed, each with a SHARC DSP, a 10K100A Altera FPGA and 1 MByte of cyclic buffer memory. Buffer fragmentation is handled by software running on the SHARC associated with it. The buffer memory has been implemented with 1 MByte of ZBT RAM. Data input is via S-link with a maximum frequency of 40 MHz. Each ROBIn occupies a single PCI board without PCI interface. The output part, to which the ROBIns connect via SHARC links, is a PCI board with a SHARC DSP and a PCI interface constructed with a PLX chip. Part of the functionality of the ROB controller is residing in this output part, the remaining part is taken care of by the processor of the PC. In most measurements data which is normally passed via the memory bus interface by the FPGA to the SHARC were sent to the SHARC of that ROBIn via one of its links. The data mentioned consists of data extracted from the input data by the FPGA of the ROBIn and addresses generated in the FPGA. This made it possible to measure without real S-link input. It was shown that the performance results are almost the

same as would be measured with real input via a S-link. Measurements have been done with the two ROBIn connected via SHARC links to the output card. For studying the performance of the output part each ROBIn was programmed to send its data twice via two links, so that the output part in effect received data from 4 ROBIn sources.

9.2.1 Paroli tests

The NIKHEF Group also studied the possible use of the PAROLI devices supporting electrical to optical conversion and optical to electrical conversion for 12 fibers, each with an up to 1.25 Gbit/s data stream (see Section 7.9.4, “Front-end connections,” on page 42) has been studied. The purpose was to test whether it would be possible to use in the RODs electrical output and in the ROBs electrical input in combination with a high-density optical connection via PAROLI devices. The motivation for this study was the possible price advantage of connections for the ROLs based on these devices.

A board was constructed carrying PAROLI devices of the type V23814-K1306-M130 (transmitter) and V23815-K1306-M130 (receiver) on loan from CERN, with circuitry for conversion from LVDS to ECL and ECL to LVDS signals and with Gigabit Ethernet converters. The PAROLI devices use LVDS signalling, while for the cable connections standard Gigabit Ethernet signalling was used. With the help of a FibreChannel based S-link source and S-link destination with electrical interfacing a test set-up was constructed. In the setup data was passed from the S-link source (driven by a microEnable card [17]) via a 1.5 m Gigabit Ethernet cable, connector and signal convertor to the PAROLI transmitter, next via one fibre in an optical ribbon cable to the PAROLI receiver and then again via signal converter, connector and 1.5 m cable to the S-link destination card read out by a CRUSH module [14]. The S-link boards require a bidirectional connection : in the test setup the return channel could be provided with one of the remaining channels of the PAROLI devices. In reality a sender and a transmitter device will need to be used at each end of an optical connection (the Gigabit Ethernet cables and connectors provide a return channel). In the setup data was passed from sender to receiver at a speed of 53 MByte/s (with a bit rate of 1 Gbit/s on the fibre, 53 MByte/s was the highest speed that could be achieved for the combination of microEnable and S-link interface used). “Eye patterns” were inspected and found to be acceptable. During one test run apparently one of the two lasers used stopped operating¹. With two other pairs of lasers measurements have been made during up to 316 hours (passing $5 * 10^{14}$ bits) without observing errors (error checking done by the S-link hardware). Hence it was shown that a reliable connection is possible with the type of connection used in the test setup and for the 1 GBit/s signalling provided by the S-link interfaces.

9.3 SACLAY GROUP [15]

The Saclay group has studied a ROBIn implementation without input part. It makes use of a 33 MHz Intel i960Jx processor, interfaced to PCI via a PLX9080 chip and is in the form of a PMC mezzanine card. Three cards have been produced. Measure-

1. After the tests the board with the devices was sent to Infineon, where it was found that the safety circuitry for shutting down the laser was broken. The faulty device (from an early production series) was exchanged for a new completely functional device.

ments have been made with respect to data retrieval for up to three cards with either a CompactPCI board, a VME board or a PC as ROB controller and in conjunction with interfacing to an 155 Mbit/s ATM network. Further work on a complete ROBIN implementation with a 100 MHz Intel i960, 8 MByte of shared and paged event buffer SDRAM memory and CPLDs dedicated to handling the input data is in progress. As SDRAM memory is used for standard PC main memory a cheap and large buffer memory is possible. Buffer fragmentation will be handled partially by the CPLDs. With the current partial implementation of the ROBIN the feasibility of selection of the part of the calorimeter data needed by the LVL2 system has been demonstrated.

9.4 UCL/RHUL GROUP [16]

The UCL and RHUL groups have studied a complete ROBIN implementation based on the Intel i960Rx, a MACH5 CPLD and SRAM with paged memory management. Buffer memory fragmentation is handled partially by the CPLD. Data input is via S-link with a maximum frequency of 33 MHz. The bandwidth of the SRAM memory is 132 MByte/s and is shared between the input link and the i960Rx. The processor has a built-in PCI bridge. Direct access from the PCI bus to the buffer memory and control registers is possible. Electrically equivalent PCI and PMC implementations have been produced. Measurements with up to three cards connected to a single PCI bus segment have been done with as ROB Controller either a PC or a VME board (DAQ-1 setup) and with a modified version of the SLIDAS [18] as data generator.

9.5 COMPARISON OF RESULTS

For comparing the measurement results to the requirements obtained from the paper model the available measurement results had to be extrapolated. This was done with the help of simple equations fitted to the available data. The results for the devices with S-link input are presented in Table 16, Table 17 and Table 18. The Mannheim results are based on the “Active ROB Complex” measurements.

 TABLE 16.

Achieved input bandwidth and maximum required input bandwidth for 40 / 75 kHz LVL1 rate

	UK	Mannheim	NIKHEF	Required
Maximum input bandwidth (MB/s)	132 - output bandwidth	80	160	72 / 135

TABLE 17.

ROBIn performance for 40 kHz LVL1 rate, accept rate = 2 kHz

Maximum RoI request rate (kHz), 40 kHz LVL1 rate				
Subdetector (Bytes)	UK	Mannheim	NIKHEF ^a	Paper model
Pixels (112)	23	175	34 (70)	12
SCT (282)	20	137	34 (70)	11
TRT (332 / 782)	19 / 14	129 / 83	34 / 28 (68 / 37)	10
Calorimeters (1056 / 1832)	~14 / ~8	68 / 45	23 / 15 (28 / 17)	2.7

a. Numbers between brackets indicate the maximum rate when RoI data is requested and output via SHARC links only, i.e. without using the PCI bus

TABLE 18.

ROBIn performance for 75 kHz LVL1 rate, accept rate = 2 kHz

Maximum RoI request rate (kHz), 75 kHz LVL1 rate				
Subdetector (Bytes)	UK	Mannheim CERN	NIKHEF ^a	Paper model
Pixels (112)	14	175	33 (56)	14
SCT (282)	12	137	33 (56)	12
TRT (332 / 782)	12 / 9	129 / 83	33 / 27 (56 / 35)	11
Calorimeters (1056 / 1832)	input band- width insuffi- cient	input bandwidth insuffi- cient	22 / 14 (27 / 16)	4.0

a. Numbers between brackets indicate the maximum rate when RoI data is requested and output via SHARC links only, i.e. without using the PCI bus

TABLE 19.

ROB complex performance for 40 kHz LVL1 rate, accept rate = 2 kHz. Saclay results are for 1 / 2 / 3 ROBIns, other results are for 1 / 4 ROBIns. The Saclay results are determined exclusively by the ATM link bandwidth of 15 MByte/s for the TRT and the calorimeters.

Maximum RoI request rate per ROBIn (kHz), 40 kHz LVL1 rate				
Subdetector (Bytes)	Saclay	Mannheim CERN	NIKHEF	Paper model
Pixels (112)	26 / 23 / 17	175 / 71	34 / 15	12
SCT (282)	36 / 19 / 12	137 / 59	34 / 15	11
TRT (782)	20 / 10 / 6	83 / 39	28 / 13	10
Calorimeters (1056)	14 / 7 / 4.7	68 / 33	23 / 10	2.7
Calorimeters (1832)	8 / 4.1 / 2.7	44 / 22	15 / 6.2	2.7

TABLE 20.

ROB complex performance for 75 kHz LVL1 rate, accept rate = 2 kHz. Saclay results are for 1 / 2 / 3 ROBIns, other results are for 1 / 4 ROBIns. The Saclay results are determined exclusively by the ATM link bandwidth of 15 MByte/s for the TRT and the calorimeters.

Maximum RoI request rate per ROBIn (kHz), 75 kHz LVL1 rate				
Subdetector (Bytes)	Saclay	Mannheim	NIKHEF	Paper model
Pixels (112)	26 / 23 / 17	175 / 71	33 / 15	14
SCT (282)	36 / 19 / 12	137 / 59	33 / 15	12
TRT (782)	20 / 10 / 6	83 / 39	27 / 12	11

Maximum RoI request rate per ROBIN (kHz), 75 kHz LVL1 rate				
Subdetector (Bytes)	Saclay	Mannheim	NIKHEF	Paper model
Cal.meters (1056 Bytes)	14 / 7 / 4.7	68 / 33, input band- width not sufficient	22 / 9.6	4.0
Cal.meters (1832 Bytes)	8 / 4.1 / 2.7	44 / 22 input band- width not sufficient	14 / 5.8	4.0

In Table 19 and Table 20 results for a ROB Complex calculated from measurements available are presented. In the case of the Saclay design RoI requests were passed across an ATM link with 15 MByte/s bandwidth and the data requested was returned via this link. For the other measurements the RoI requests were generated internally in the processor of the CPU of the PC acting as ROB controller and the data requested was sent to the CPU across the PCI bus or busses connecting with the ROBINs.

9.5.1 Conclusions from prototype studies

It has been demonstrated that the requirements can be satisfied with current technology: projected input rates can be handled, output to LVL2 and to the Event Builder is achievable at the necessary rates and bandwidth, some on-the-fly pre-processing is possible with the use of spare processor capacity or of FPGAs in the ROBINs or in an Active ROB Complex, both page-managed and circular-buffer-based buffer management work, both point-to-point and bus systems are viable for internal communication in the ROB complex, software control provides flexibility.

The implementation studies show that for the ROBIN a compact design with acceptable power dissipation is possible. This allows the construction of a ROB Complex with several (3-6) ROBINs on a single Eurocard size board, or from a single board computer with ROBINs implemented on mezzanine boards. COTS hardware seems to be able to achieve the requirements with respect to output to the LVL2 system and to the EB. It has been shown that advantage can be taken of multi-processor and multi-bus COTS systems with minimal effort. However, true COTS hardware is not necessarily able to handle the input rates.

10.0 REFERENCES

1. ATL-DAQ-2000-009 . (ATL-COM-DAQ-2000-015) . An UML description of the ATLAS ROB viewed from the Level-2 Trigger. Huet, M; 09 Mar 2000.
2. ATL-DAQ-98-103 . (ATL-D-PN-103) . Trigger & DAQ Interfaces with Front-End Systems: Requirement Document Version 2.0. Trigger DAQ Steering Gp ; 09 Jun 1998.
3. ATL-DAQ-97-062 . (ATL-D-PN-62) . Detector and Readout specifications, and buffer RoI relations, for the Level-2 trigger demonstrator program. Bock, R ; Le Du, P; 27 Jan 1997.
4. ATL-DAQ-99-001 . (ATL-COM-DAQ-99-001) . A Data Preparation Algorithm for the Precision Tracker LVL2 FEX . Dankers, R. ; Baines, J.T.M.; 12 Jan 1999.
5. ATL-COM-DAQ-2000-022 . Paper modelling of the ATLAS LVL2 trigger system. Bystricky, J. ; Vermeulen, J.C .; 16 Mar 2000.
6. ATL-DAQ-2000-022 . (ATL-COM-DAQ-99-020) . The active ROB complex . Bock, R K ; Francis, D ; Vermeulen, J ; Wheeler, S ; 11 Nov 1999
7. ATL-COM-DAQ-2000-031. Computer modelling of the ATLAS LVL2 trigger system. Vermeulen, J.C. ; 17 Mar 2000.
8. <http://hsi.web.cern.ch/HSI/rob/>
9. ATL-DAQ-2000-019 . (ATL-COM-DAQ-2000-032) . The Atlas Level 2 Reference Software . Hauser, R. ; 17 Mar 2000.
10. ATL-DAQ-2000-001 . (ATL-COM-DAQ-2000-002) . Back-end Summary Document . Burckhart, D ; et al. 16 Dec 1999.
11. See: "The ROB-API" Design Note under "Reference Software and Testbed Notes" at: <http://atlas.web.cern.ch/Atlas/project/LVL2testbed/www/>
12. See: "Internal APIs for ROB Complex" (draft document by G. Crone, M. Huet), available via the ROB Complex page[19].
13. Unpublished, Measurements on a commercial Active Rob Complex. Bock, R.; Y.Ermolin Y.; A.Kugel A.; Werner P.
14. ATL-DAQ-2000-021 . A SHARC based ROB Complex : design and measurement results. Boterenbrood, H. ; Jansweijer, P. ; Kieft, G. ; Scholte, R. ; Slopsema, R. ; Vermeulen, J. ; 17 Mar 2000.
15. ATL-DAQ-2000-014 . (ATL-COM-DAQ-2000-007) . A Scheme of Read-Out Organization for the ATLAS High-Level Triggers and DAQ based on ROB Complexes. Calvet, D ; Gachelin, O ; Huet, M ; Mandjavidze, I ; 30 Nov 1999.
16. ATL-DAQ-2000-013 . (ATL-COM-DAQ-2000-028) . The UK ROB-in,A prototype ATLAS readout buffer input module. Boorman, G ; Clarke, P ; Cranfield, R ; Crone, G ; Green, B ; Strong, J ; 16 Mar 2000.
17. <http://www.silicon-software.de> (microEnable is manufactured by Silicon Software).
18. <http://hsi.web.cern.ch/HSI/s-link/devices/slidas/>; SLIDAS, "S-Link Infinite Data Source". Available from INCAA computers (<http://www.incaacomputers.com>).
19. <http://www.nikhef.nl/pub/experiments/atlas/daq/ROB.html>.
20. ATL-DAQ-97-066 . (ATL-D-PN-66) . TRT Preprocessing-Algorithm, Implementations and Benchmarks. Bock, R ; Noffz, K-H ; Sessler, M ; 21 Mar 1997.

21. DAQ-1 note 63. Inter and Intra-IOM Message Passing in the DAQ-unit. G.J. Crone, D. Francis, M. Joos and J. Petersen ; November 1997,
<http://atddoc.cern.ch/Atlas/Notes/>
22. DAQ-1 note 65. Intelligent I/O processors in the DAQ-unit of ATLAS DAQ/EF prototype -1. G. Crone, D. Francis and G. Mornacchi; November 1997,
<http://atddoc.cern.ch/Atlas/Notes/>
23. DAQ-1 notes, see : <http://atddoc.cern.ch/Atlas/Notes/>
24. ATL-DAQ-98-129 . (ATL-COM-DAQ-98-018) . The event format in the ATLAS DAQ/EF prototype -1. Bee, C ; Boyle, O ; Francis, D ; Mapelli, L ; McLaren, R ; Mornacchi, G ; Petersen, J ; 27 Oct 1998.