The ATLAS ROBIN – A High-Performance Data-Acquisition Module

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

Vorgelegt von

Dipl. Ing. (FH) Andreas Kugel
aus Lindau

Mannheim, Juni 2009

Dekan:	Prof. Dr. Felix Freiling, Universität Mannheim
Referent:	Prof. Dr. Reinhard Männer, Universität Mannheim
Korreferent:	Prof. Dr. Peter Fischer, Universität Mannheim

Tag der mündlichen Prüfung: 19. August 2009

The ATLAS ROBIN – A High-Performance Data-Acquisition Module

Abstract

This work presents the re-configurable processor ROBIN, which is a key element of the data-acquisition-system of the ATLAS experiment, located at the new LHC at CERN. The ATLAS detector provides data over 1600 channels simultaneously towards the DAQ system. The ATLAS dataflow model follows the "PULL" strategy in contrast to the commonly used "PUSH" strategy. The data volume transported is reduced by a factor of 10, however the data must be temporarily stored at the entry to the DAQ system. The input layer consists of approx. 160 ROS read-out units comprising 1 PC and 4 ROBIN modules. Each ROBIN device acquires detector data via 3 input channels and performs local buffering. Board control is done via a 64-bit PCI interface. Event selection and data transmission runs via PCI in the baseline bus-based ROS. Alternatively, a local GE interface can take over part or all of the data traffic in the switch-based ROS, in order to reduce the load on the host PC. The performance of the ROBIN module stems from the close cooperation of a fast embedded processor with a complex FPGA. The efficient task-distribution lets the processor handle all complex management functionality, programmed in "C" while all movement of data is performed by the FPGA via multiple, concurrently operating DMA engines.

The ROBIN-project was carried-out by and international team and comprises the design specification, the development of the ROBIN hardware, firmware (VHDL and C-Code), host-code (C++), prototyping, volume production and installation of 700 boards. The project was led by the author of this thesis. The hardware platform is an evolution of a FPGA processor previously designed by the author. He has contributed elementary concepts of the communication mechanisms and the "C"-coded embedded application software. He also organised and supervised the prototype and series productions including the various design reports and presentations.

The results show that the ROBIN-module is able to meet its ambitious requirements of 100kHz incoming fragment rate per channel with a concurrent outgoing fragment rate of 21kHz per channel. At the system level, each ROS unit (12 channels) operates at the same rates, however for a subset of the channels only. The ATLAS DAQ system – with 640 ROBIN modules installed – has performed a successful data-taking phase at the start-up of the LHC in September.

Der ATLAS ROBIN – Eine Hochleistungs-Datenerfassungsbaugruppe

Zusammenfassung

Diese Arbeit beschreibt den re-konfigurierbaren Prozessor ROBIN, der ein Schlüsselelement im Datenerfassungssystem des ATLAS-Experiments des LHC am CERN ist. Der ATLAS Detektor liefert Daten über 1600 Kanäle gleichzeitig an das DAQ System. Im Gegensatz zur üblichen „PUSH“ Strategie für den Datentransport kommt bei ATLAS eine „PULL“ Strategie zur Anwendung, wodurch das zu transportierende Datenvolumen um den Faktor 10 reduziert wird. Dazu müssen die Daten am Eingang des DAQ System zwischengespeichert werden. Die Eingangsstufe nimmt die Daten in 160 ROS Ausleseeinheiten entgegen, die jeweils aus 1 PC mit 4 ROBIN Karten bestehen, jede ROBIN Karte ist wiederum mit 3 Detektorkanälen verbunden. Die Daten werden auf den ROBINs gespeichert. Die Überwachung der Baugruppe geschieht vom PC aus über ein 64-bit PCI Interface. In der Bus-basierten Basisimplementierung des ROS erfolgt die Auswahl und Übertragung der Daten ebenfalls über das PCI Interface. Eine lokale Gigabit-Ethernet Schnittstelle übernimmt in der alternativen Netzwerk-basierten Implementierung einen Teil oder den gesamten Datenverkehr, um den PC zu entlasten. Die hohe Leistungsfähigkeit und Flexibilität des ROBIN ergibt sich aus der Kombination eines schnellen eingebetteten Prozessors mit einem hoch integrierten FPGA. In einer effektiven Aufgabenverteilung bearbeitet der Prozessor alle komplexen Verwaltungsaufgaben, die in „C“ programmiert sind, während das FPGA sämtliche Datenbewegungen mit mehreren, gleichzeitig arbeitenden DMA Einheiten durchführt.

Das ROBIN Projekt wurde von einem internationalen Team durchgeführt. Es umfasst die Spezifikation des Designs, die Entwicklung der Hardware und der Firmware (VHDL und C) der ROBIN Baugruppe, PC-Software (in C++) für Ansteuerung, Emulation und Test, Produktion und Test von Prototypen sowie die Serienfertigung und Inbetriebnahme von 700 Baugruppen.

Das Projekt wurde unter Leitung des Autors dieser Arbeit durchgeführt. Die Hardwareplattform ist eine Weiterentwicklung eines vom Autor entworfenen FPGA Prozessors. Grundlegende Konzepte der Kommunikationsmechanismen stammen vom Autor, ebenso die „C“-Anwendungssoftware. Ebenfalls wurde die Herstellung der Prototypen und die Serienfertigung inklusive der notwendigen Statusberichte vom Autor vorbereitet und überwacht bzw. durchgeführt.

Die Ergebnisse zeigen, das die ROBIN Baugruppe die hohen Leistungsanforderungen von 100kHz Ereignisrate am Eingang bei gleichzeitig 21kHz Abfragerate – jeweils auf allen 3 Kanälen gleichzeitig – erfüllt. Auf Systemebene liegt der Arbeitspunkte für jede normale ROS-Einheit mit 12 Kanälen bei 100kHz Eingangsrate und 21kHz Abfragerate, jedoch nur für einen Teil der Kanäle. Das ATLAS DAQ System mit 640 installierten ROBIN Baugruppen hat den Datenerfassungsbetrieb zum Start des LHC erfolgreich aufgenommen.

Contents

1 Introduction....7
2 Data Acquisition Systems....11
2.1 LHC Experiments....11
2.1.1 CMS....13
2.1.2 ALICE....17
2.2 LHCb....18
2.3 Summary....19
3 ATLAS DAQ....21
3.1 System View....21
3.2 Dataflow....25
3.3 ROS....30
3.3.1 Event handling....31
3.3.2 Configuration and monitoring....31
3.3.3 Baseline bus-based ROS....32
3.3.4 Switch-based ROS....37
3.4 Summary....38
4 FPGA Technology....41
4.1 Device types....41
4.2 Basic elements....41
4.3 Tools....44
4.3.1 Specification....44
4.3.2 Simulation and test....45
4.3.3 Synthesis....47
4.3.4 Vendor specific tools....48
4.4 IP-Cores....48
4.4.1 Embedded processors....49
4.4.2 Ethernet MAC....49
4.5 Summary....50
5 ROBIN....53
5.1 Requirements....53
5.2 Project Management....54
5.3 Implementation....55
5.3.1 Hardware....56
5.3.2 VHDL Firmware....72
5.3.3 Software....79
5.4 Installation and commissioning....89
5.5 Summary....90
6 Results....93
6.1 Stand-alone....93
6.1.1 PCI access....94
6.1.2 Network access....96
6.1.3 Event bookkeeping....97
6.2 System performance....99
6.2.1 Bus-based ROS....99
6.2.2 ATLAS Dataflow....100
6.3 Issues....101
6.4 Reliability....103
6.5 Summary....105
7 Discussion and Conclusions....107
7.1 Performance Assessment....107

7.2 Prospects....108
7.2.1 Short term developments....108
7.2.2 Upgrade Phase-1....109
7.2.3 Upgrade Phase-2....110
7.3 Summary....110
8 Appendix....111
8.1 Pre-ROBIN hardware....111
8.1.1 MPRACE-1....111
8.1.2 µEnable....112
8.2 ROBIN configuration parameter set....113
8.3 ROBIN statistic....114
8.4 ATLAS detector parameters....116
8.5 Glossary....119
List of figures....121
List of tables....122
Bibliography....123
Acknowledgements....129

1 Introduction

The research objective of Particle Physics is the structure of matter. According to the Standard Model [SMS1][SMS2][DISSFLICK] there are 12 particles which form matter and 4 force carrier particles which describe the interaction between the matter particles. Matter particles are distinguished in quarks and leptons and further into 3 generations. Each generation is composed from a pair of each type of particles. Only the 4 particles of the first generation – the electron with its neutrino and the up and down quarks – build the matter we are aware of. The interactions between particles are called forces and are bound to the the exchange of force coupling particles, the bosons. Two of the forces – gravity and electromagnetism – are quite commonly known. The two others – the weak and the strong force – are related to interactions between and inside nuclear particles, like protons, neutrons and quarks. The strong, weak and electromagnetic force cover a relative strength range of about 1:100, but the strength of the gravity is 43 orders of magnitude smaller than the strength of the strong force. This is the reason why gravity is ignored[1] by many of the Standard Model calculations. We have experimental evidence for the existence of the bosons carrying the first three forces. Differently however for gravity, the boson carrying the mass – the Higgs-boson – is still undiscovered.

To find the Higgs and to explore other important issues of particle physics – like SUSY [SUSY], CP violations [CPV] or QCD [QCD] – a huge machinery has been developed: the LHC [LHCJINST] at CERN. At the LHC two proton beams are accelerated and are travelling in opposite direction in the LHC ring. At 4 experimental stations, ATLAS [ATLJINST], ALICE [ALICEJINST], CMS [CMSJINST] and LHCb [LHCBJINST], the beams can be focused such that the protons collide with a maximum energy of 2 * 7TeV, an energy level close to that at the origin of the universe.

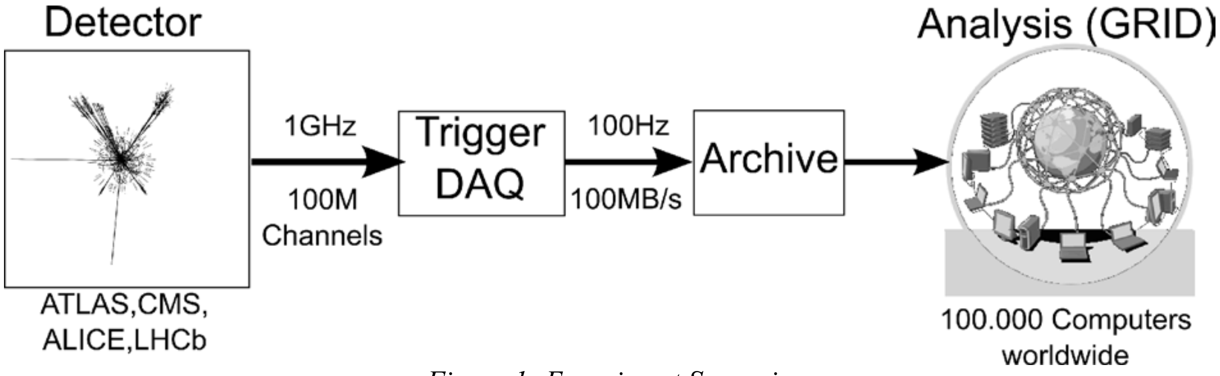


Figure 1: Experiment Scenario

Each of the experimental stations, where ATLAS is the largest of, employ very complex equipment – called a detector[2] – to find all the particles which are produced during and immediately after the collisions. The ATLAS detector is a 44m long, 25m high, 7000t heavy precision measurement instrument consisting of 7 different sub-detectors[3], generating data on roughly 100 million channels at

1 This is clearly opposite to the human common sense, which easily accepts gravity as the strongest and most obvious force, while it is not aware of the strong and weak forces.

2 The "detector" is in fact a collection of different detectors, sensitive for different particles like electrons and muons.

3 The sub-detectors are: Muon chambers, semiconductor tracker, transition radiation tracker, pixel detector, LAr electromagnetic calorimeters, LAr hadronic calorimeters and tile calorimeters.

a interaction rate of the protons of 1GHz[4]. The event recording rate however is limited to 200Hz at an event size in the order of 1MB. To achieve the required data reduction of $5x10^6$ a complex and efficient trigger and data-acquisition system (TDAQ) is needed. A simplified view of this general scenario is shown in Figure 1.

Traditionally, trigger and data-acquisition systems in high-energy physics (HEP) utilise custom electronics in many areas, for example to interface to the detectors, to satisfy extreme performance or density requirements or because the electronics is exposed to radiation. There are also several disadvantages to custom electronics, in particular cost, long development times and a long-term dependency on particular know-how and components. Considering the development time of 15 years, the operation time of another 10 years or more and the cost, there was a strong desire to use COTS wherever possible to build the ATLAS TDAQ system. The ATLAS solution to this dilemma in the TDAQ area is to equip standard PCs (the COTS) with custom modules, which provide the missing performance and functionality.

ATLAS TDAQ uses a layered trigger architecture to reduce the amount of data to be transferred. The levels are called level-1 trigger (L1), level-2 trigger (L2) and event-filter[5] (EF), the latter two form the higher-level triggers (HLT). The L1 is closely coupled to the front-end electronics of the sub-detectors and uses custom electronics to get the event rate down to 100kHz. L2 and EF are based on farms of commodity PCs. Unlike many HEP experiments the L2 of ATLAS does not operate on the full detector data volume but reduces the amount of data with two mechanisms – the region-of-interest (RoI) principle and the sequential selection strategy. Figure 2 shows the traditional approach on the left, where data rate and volume are reduced layer by layer while data is pushed through the TDAQ system and the ATLAS approach on the right, where the the data is stored after the first step (L1) in the ROS and L2 and EF request only a fraction of the data.

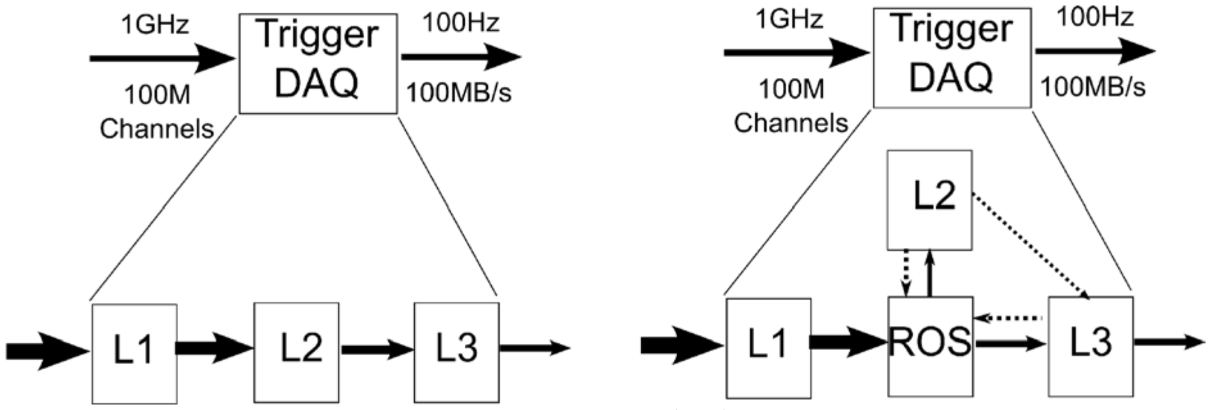


Figure 2: TDAQ levels

The ROS, again a farm of PCs, implements the buffering capabilities of the DAQ and interfaces to the detector/L1 system on one side and to the HLT system on the other side. The detector/L1 side generates fragments of the event data on 1600 channels and transmits them at the L1-rate of up to

4 The proton beams are structured in bunches and the bunch crossing frequency is 40MHz. For individual protons the collision rate becomes 1 GHz.

5 The third trigger level (L3) is called Event Filter (EF) in ATLAS terminology.

100kHz through unidirectional[6] optical links (ROL) with a nominal bandwidth of 160MB/s. L2 and EF subsequently request portions of the data from the ROS. Typically, L2 requests occur at a higher rate but address only a fraction of the fragments while EF requests occur at a lower rate but address full events. Rate and bandwidth exceed the capabilities of commodity PCs by far.

To achieve the required performance, a custom FPGA-based I/O-coprocessor – the ROBIN – was developed. Early studies [DJFROB] showed, that the handling of a single ROL using a combination of re-configurable logic and processor is possible[7]. Subsequent technology studies led to the development of different single channel modules, followed by a dual-channel ROBIN-prototype. The final implementation of the ROBIN concentrates three input channels on a 64-bit PCI card. The architecture and hardware design of the ROBIN is based to a large extent on systems developed previously by the author (MPRACE-1, see 8.1.1) or to which he has contributed significantly (µEnable, see 8.1.2).

In addition to the baseline readout scenario via the PCI bus it also supports the switch-based readout scheme via an integrated 1G-Ethernet port. Due to the high performance requirements a custom data-path controller is needed, which is implemented in a 2M-gates FPGA. FPGA technology enables to create efficient and optimised I/O interfaces, memory controllers and logic functions in a single device, while keeping flexibility and maintainability. An attached processor implements the management, messaging and monitoring functionality.

This work describes the development, implementation, production and installation of the ATLAS ROBIN. The results show that the ROBIN as a component satisfies the requirements with respect to performance and cost. In addition, the ATLAS ROS, which is the framework for the ROBIN, has demonstrated to reach its performance goals. The low failure rate observed during approximately one year of operation prior to the LHC start-up in September 2008 demonstrates the reliability of the ROBIN.

This thesis is structured as follows:

- Chapter 2 provides an overview of typical HEP data-acquisition systems, with a focus on the CMS experiment.
- The ATLAS Trigger and DAQ system is described in chapter 3 .
- Important technological aspects – in particular the FPGA technology – used for the implementation of the ROBIN are introduced in chapter 4 .
- The details of the ROBIN: development, implementation, production and commissioning are described in chapter 5 .
- The results from stand-alone and system tests are presented in chapter 6 .
- Chapter 7 concludes with an assessment of the achieved goals and an outlook for the upcoming ATLAS operation and upgrade phases.
- The appendix provides information on previous systems and some additional tables relevant for the operation of the ROBIN.

6 The ROLs are physically bi-directional, but the sole use-case for the return path is flow-control.

7 The early prototypes didn't have any of the operational monitoring features present on the current ROBINs.

2 Data Acquisition Systems

This chapter presents the main issues typical data acquisition system have to cope with. The “PUSH” data transfer model followed by ALICE, CMS and LHCb is introduced and its distinction from the “PULL” model employed by ATLAS. The implementation of CMS – the largest partner experiment to ATLAS – is shown in detail and the characteristic parameters of ALICE and LHCb are presented.

In short, the task of a data-acquisition system is to collect data from an arrangement of sources and to deliver it to a unit which performs storage and typically analysis. In the case of ATLAS and the other LHC experiments the individual DAQ systems have to deal with some or all of the following challenges:

- Large number of input channels
- Large data volume
- High rate of events
- High connectivity and throughput requirements on networks
- High performance requirements for event processing
- Buffering required to compensate latencies
- Low cost, longevity, scalability, etc.

2.1 LHC Experiments

The LHC proton-proton collider is situated at CERN, Switzerland in a 27km long circular, underground tunnel (see Figure 3). There are four main experimental sites: ATLAS, ALICE, CMS and LHCb. ATLAS and CMS are general purpose experiments for the study of proton-proton collisions. ALICE is specialised for the study of heavy-ion collision, LHCb is specialised for the study CP violations in B-meson decays. ATLAS and CMS are the two largest experiments, which pose similarly high requirements on the Trigger/DAQ system: L1-rate of 100kHz, event size of 1MB and a storage rate of 100Hz, in other words an input bandwidth of 100GB/s and an output bandwidth of 100MB/s. ALICE and LHCb have different demands – higher L1-rate at smaller event size or vice versa. The DAQ of LHCb runs at an event rate of 1MHz and an output rate of 2kHz, the average event size is 35kB [LHCBDAQ]. In contrast, ALICE DAQ has a low input rate of around 8kHz but event sizes can be as large as 85MB. ALICE operates in different modes, with and without trigger, the maximum sustained output bandwidth is assumed to be 1.25GB/s.

The experiments ALICE, CMS and LHCb follow the traditional “PUSH” model for data transportation in the DAQ system, although different options have been looked at during the development phases (e.g. so called “phased” readout in LHCb [LHCBTB]). The main characteristic here is the fact, that for every event the entire data volume is unconditionally transported – typically over several layers of different media – to a node which assembles the full event prior to the recording on permanent storage, which is based on a trigger decision. This leads to very high bandwidth requirements on the interconnect infrastructure. Also, low latency transmission is needed to avoid substantial intermediate buffering. In ATLAS and CMS for example the stages feeding the DAQ dataflow system provide virtually no buffering beyond the 3 to 5µs latency of the L1 trigger. Many networking technologies have been investigated during the development phases of the LHC experiments with an emphasis on Ethernet due to its dominance in the mass market. To overcome the limitations of its technical properties – in particular the missing reliability, which is frequently compensated by reliable protocols like TCP/IP at the expense of long latencies – optimisations had to be implemented by the two experiments. CMS uses a reliable initial network layer to form sub-events which are then transported via a reliable protocol over GE. ATLAS reduces the required network load with the help of the additional L2 and ROS stages, which allows to tolerate an unreliable network. For ALICE and LHCb the total network bandwidth is lower hence better adapted to GE technology. The total connectivity of the dataflow networks is in the order of 1500 ports for ATLAS, CMS and LHCb and 350 ports for ALICE.

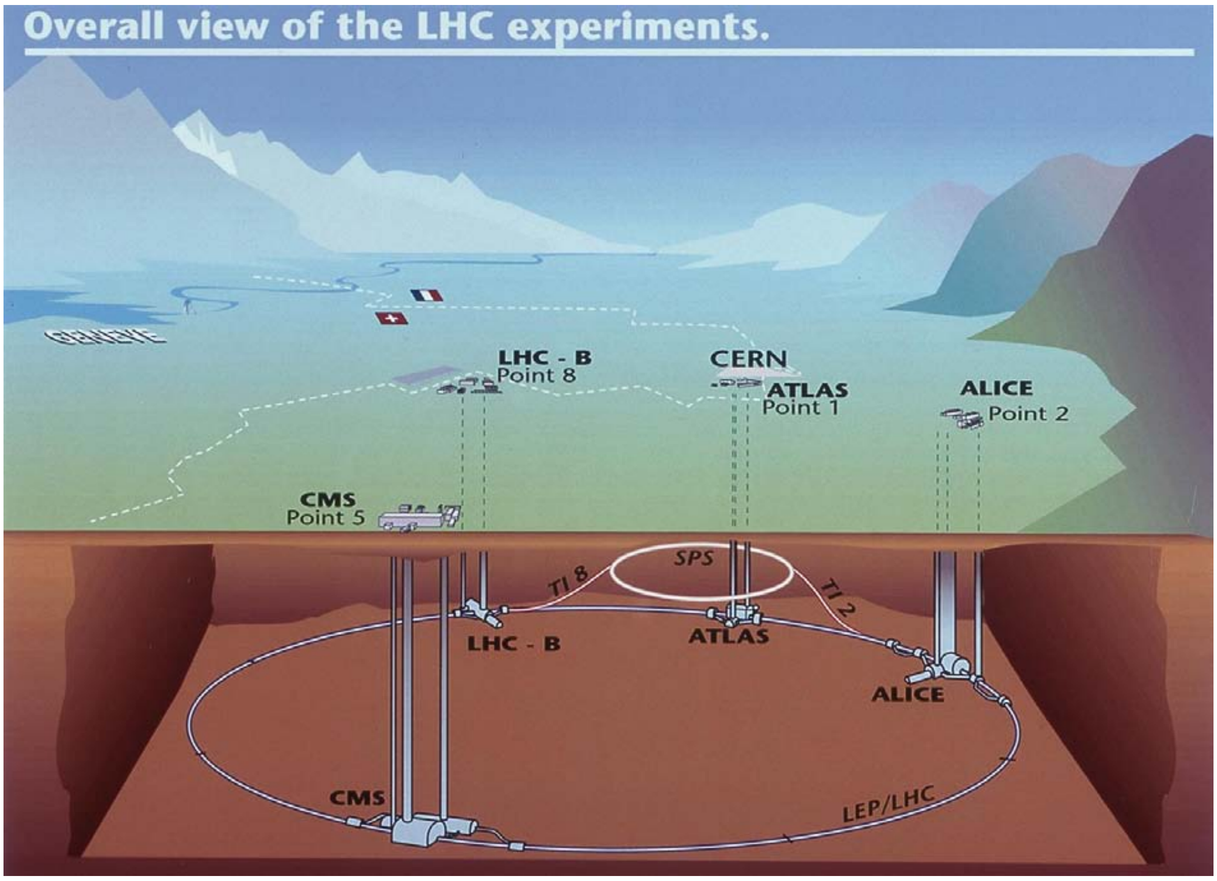


*Figure 3: LHC at CERN**

The "PUSH" model was considered by ATLAS initially as well, but then the experiment has made a different choice by implementing a true "PULL" model, in order to take full advantage from the sequential selection scheme and the RoI principle (see section 3.1). Here, the detector data are captured directly after L1 in an intermediate buffer in the ROS which forwards them only upon request to the HLT and DAQ system. This approach reduces the amount of data to be transferred to about 6% of the initial data volume after L1 for the typical case. This reduction in the total throughput of the dataflow enables the use of an all-commodity network infrastructure. The drawback is the increased system complexity due to the additional flow of control messages (data request and delete messages), the additional intermediate buffering stage and L2 trigger.

All LHC experiments employ a custom component to receive data from the sub-detectors over dedicated links and to feed them – directly or via a PC – into a network infrastructure. Due to the transport model used in ATLAS this component – the ROBIN – has a complexity far higher than that of the other experiments. Technically, the ATLAS ROS could be tuned from its regular operation mode (reducing the data volume by a factor of 10 roughly) up to a full readout mode (forwarding the entire data volume), of course at the expense of additional components. Also, conversion to “PUSH” mode would be viable (however is not foreseen) by re-programming of the existing components.

The “PULL” mode is not viable for the DAQ systems of the other LHC elements, as the network infrastructure is not prepared to handle the required rate of request messages and there is no intermediate buffering capacity available to cover the latencies of the HLT systems. Investigations during the upcoming SLHC upgrade programme will certainly revisit both models for all experiments.

2.1.1 CMS

The CMS experiment is a large multi-sub-detector precision measuring device for charged particles and photons. The momentum of charged particles is measured via the curvature of particle tracks, caused by the bending force of a strong, superconducting magnet. The inner tracking detectors (pixel, silicon strip) and the hadronic and electromagnetic calorimeters are placed inside the 13m long magnet. Four muon-detecting stations are located externally to the magnet. The arrangement is shown in Figure 4. When operating at design luminosity approximately 1.000 particle tracks per bunch-crossing (25ns) will pass the inner tracking detectors. This requires a high resolution of the pixel and silicon strip sensors, leading to high power and hence cooling requirements. However, the resulting installations (cables, cooling pipes) introduces unwanted effects[8], so the design is a compromise.

2.1.1.1 Trigger

CMS uses a 2-level trigger system. The L1 is implemented in custom hardware and reduces the 40MHz bunch-crossing rate to an L1-event rate of 100kHz. The generated data volume is in the order of 100GB/s, distributed over 626 data sources from the various sub-detectors. This entire data volume has to be transferred to a large computing farm running the HLT algorithms [CMSTRIG]. Subsequently, filtered events are sent at 100Hz to permanent storage.

8 All material in the detector can cause multiple scattering, bremsstrahlung, photon conversion and nuclear interactions.

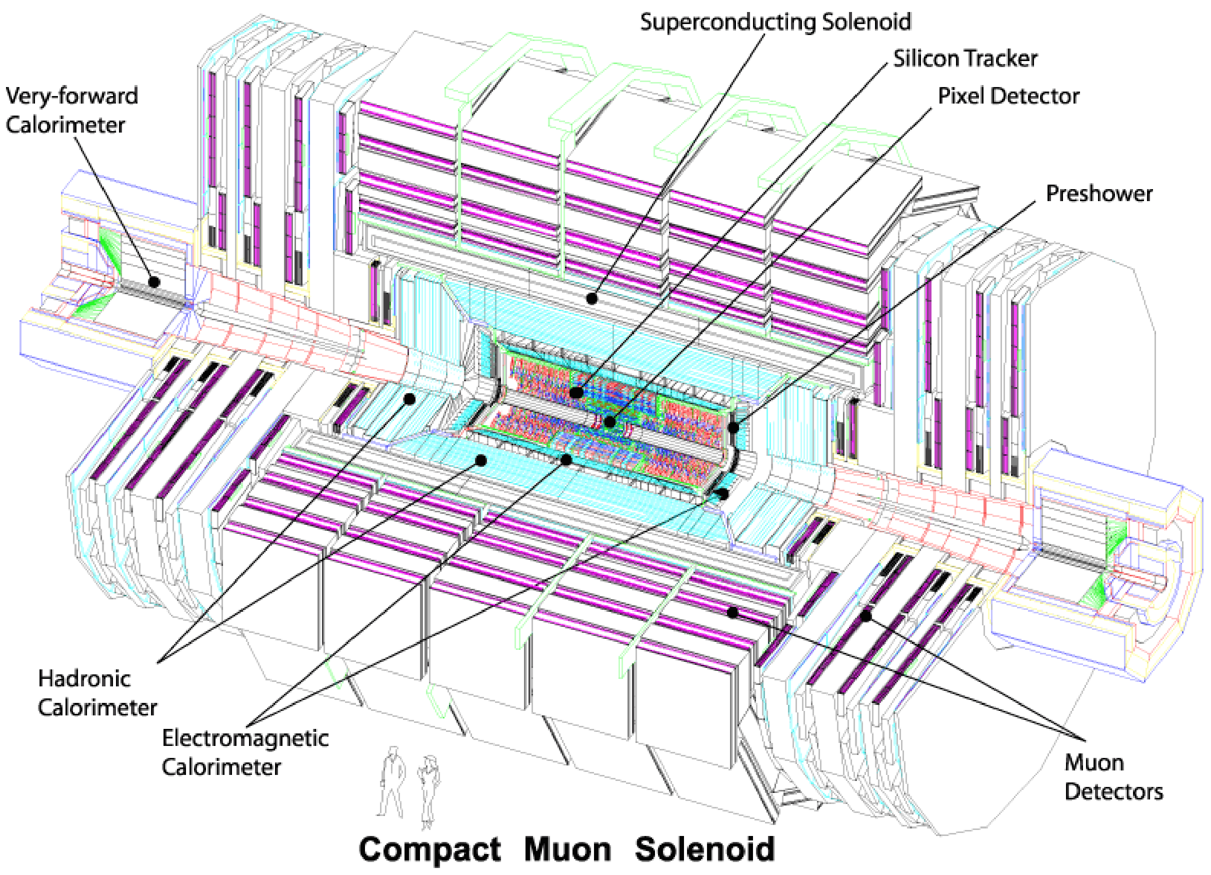


Figure 4: CMS Detector [CMSJINST]

2.1.1.2 DAQ

The DAQ architecture responsible for the transport of data is shown in Figure 5. All detector channels – approximately 55 million – are assembled into 626 event fragments of 2kB size in the detector front-end electronics and buffered during the L1 latency of 3.2µs in the front-end-drivers (FEDs). The event fragments are pushed at L1-rate from the FEDs through LVDS cables[9] into 458 front-end read-out links[10] (FRLs). A Myrinet[11] based FED-builder (FB) network creates 72 super-fragment [CMSSFB] streams – each composed of 8 fragments from different FRLs – which are directed to 72 readout units (RUs). Full events are then built by combing the data from all 72 RUs via the readout-builder-network in the builder-units (BUs), which also run the HLT algorithms via the filter-units (FUs) applications. As a pure "push" architecture CMS DAQ does not require any significant buffering of the event fragments but a very large network bandwidth.

This architecture has an inherent scalability provided by the 8x8 switches of the FB layer. Every FB switch can route the fragments from the 8 input links to up to 8 output links, for example based on the L1ID. CMS uses this feature to build a "sliced" DAQ system, where the total performance can be tuned via the number of slices attached to the FB layer (see Figure 5:side view). A single slice consists of 72 RUs, a builder-network switch and 288 BUs/FUs and is able to process 12.5kHz of L1-rate. The

9 The LVDS links follow the S-Link64 specification [SLINK64]
10 The canonical number for FRLs is 576. Only FRLs are equipped with 2 input links which connect to FEDs with lower output bandwidth, such that the nominal input bandwidth of 200MB/s per FRL is not exceeded.
11 http://www.myri.com/open-specs/index.html

full performance of 100kHz L1-rate is obtained with 8 slices, which corresponds to 576 RUs and 2304 BUs.

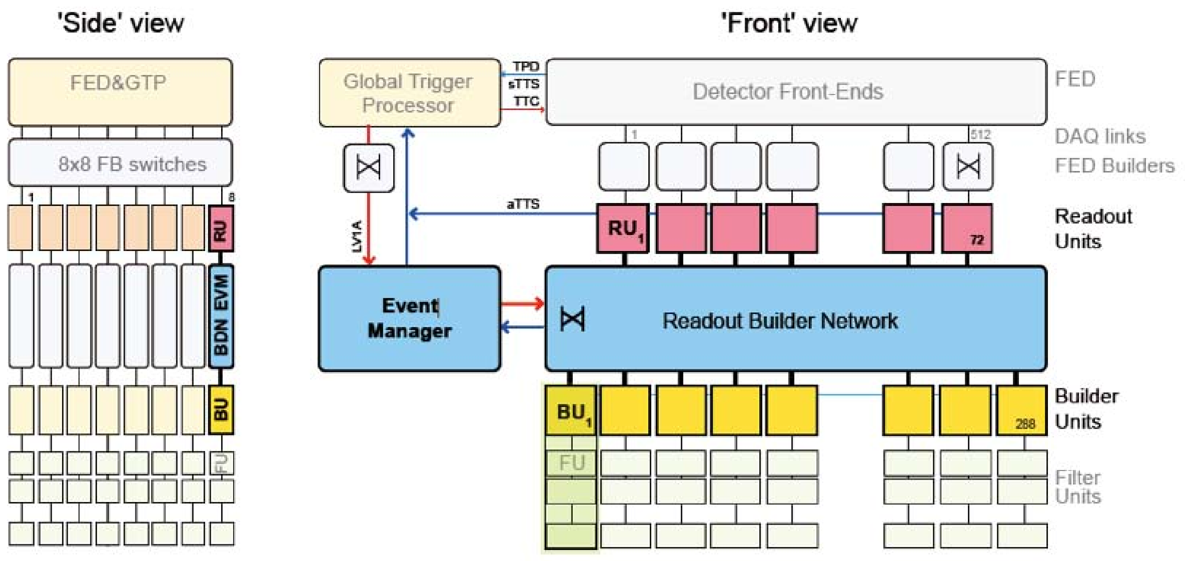


Figure 5: CMS DAQ [CMSJINST]

2.1.1.3 Readout Builder

The second stage of the CMS event-builder – RUs, builder network and BUs – is made from COTS components: PCs and a Gigabit-Ethernet (GE) network. Due to the layered and sliced architecture the required connectivity is only 72 x 288 per readout-builder network (instead of 512 x 2304 for a full single layer event-builder). However, GE does not match the input bandwidth (200MB/s) on the RUs from the Myrinet FED builder, hence a multi-rail implementation [CMSGBE] with up to 4 GE-NICs per node was chosen, leading to an aggregate bandwidth of 500MB/s. To achieve a high network performance the Ethernet packet size had to be increased to 7kB from the standard MTU size of 1500 byte. Results from a 16RUs x 60BUs test-bed[12] using 2 GE rails show a throughput for an RU well above the required 200MB/s for the standard super-fragment packet size of 16kB. The steering of the super-fragments into the RUs and from the RUs to the BUs is controlled by an event-manager (EVM) which also checks the memory allocation at the RUs and eventually requests to reduce the L1-rate via an interface to the global trigger processor (GTP). The BUs send readout-request to the EVM which in turn provides event-tag information to the RUs, which then send the super-fragments to the indicated BUs.

2.1.1.4 FED Builder

The first stage of the CMS event-builder is composed of the FRLs and the Myrinet FB. Myrinet was selected due to the inherently reliable transmission which is achieved by using a physical media[13] with

12 The test-bed used dual single-core XEONs with 2.6GHz.

13 Initially, Myrinet used parallel LVDS transmission. The current implementation is based on optical transmission with 2Gbit/s.

a low error rate, built-in CRC error detection and an efficient flow-control mechanism using a “slac”[14]-buffer. Myrinet packets can be of any length. In addition, all Myrinet network adapters and switches are equipped with a user programmable RISC processor which is used by CMS to add custom functionality. The FB connects all FRL sources (up to 512) with all 572 RU destinations via 8x8 port switches[15]. The CMS routing strategy implemented on each of the 8x8 switches combines one fragment per input into a super-fragment. Super fragments are sent to the RUs attached to the output ports depending on the event-number and a pre-loaded routing table, which in turn depends on the number of DAQ slices (every output represents a slices). As the FRLs on the input are equipped with dual-link Myrinet NICs the FB is composed from two independent “rails”, each connected to one of the FRL outputs. The two-rail structure is shown in Figure 6.

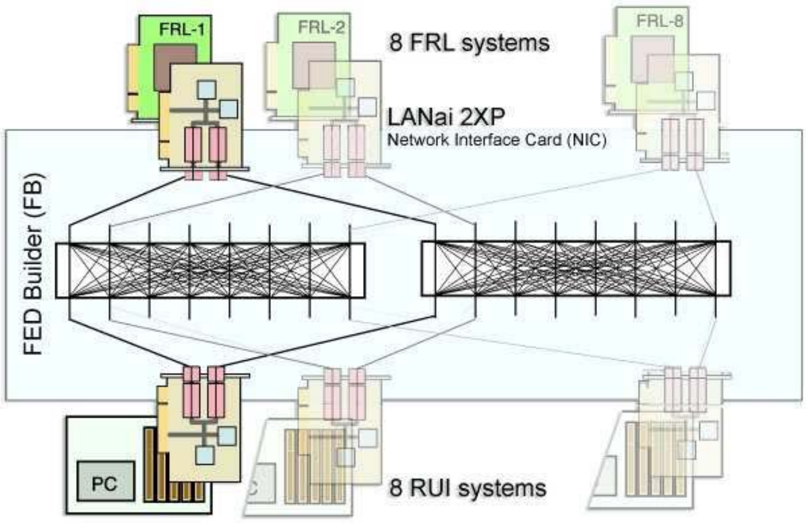


Figure 6: CMS FED-Builder Rails [CMSJINST]

Considering its location at the interface to the detector the FRL is the CMS component which is most equivalent to the ATLAS ROBIN – although it does not provide significant buffering. Furthermore, it has certain technical similarities, namely the combination of FPGA, embedded processor and host CPU. The FRL [CMSFRL] is a single width cPCI[16] card which consists of a custom base module, attached to the cPCI bus plus a commercial dual-port Myrinet network interface adapter (NIC[17]), plugged on to the base module via a standard PCI connector (Figure 7). The main FRL FPGA[18] merges the fragments from the two S-Link64 inputs (with the help of a small amount of external memory), checks the CRC and writes the combined data in blocks of fixed length to the Myrinet NIC. Additionally, the FRL provides monitoring features like fragment size histograms and can take data samples. These auxiliary functions are available via the cPCI bus to the crate controller CPU which

14 The Myrinet slac buffer is basically a FIFO with low and high watermark indicators, which are used to stop and start incoming transfers.
15 In practice a larger switch is used to implement multiple independent 8x8 groups.
16 Compact PCI (cPCI) is the industrial version of PCI, see http://www.picmg.org/test/compci.htm
17 http://www.myri.com/vlsi/LanaiX.Rev1.1.pdf
18 The main FPGA is an ALTERA EP1S10 device with 10k logic elements (LEs), 60 block memories and 6 DSP blocks.

controls up to 16 FRLs per crate. The nominal bandwidth on the input and output is 200MB/s, which is easily met by the S-Link64 (400MB/s). The Myrinet NIC is a dual-link version and both links together provide 4Gbit/s.

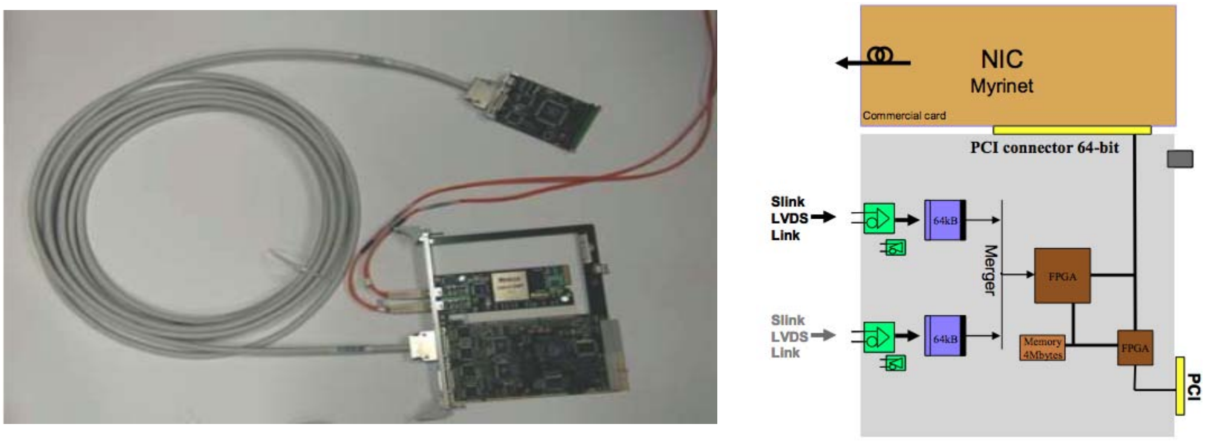


Figure 7: FRL image and block diagram[CMSFRL]

Although the NIC is a commercial device, it is used together with a custom firmware which allows it to communicate with the main FPGA on the base module via a private PCI bus. The FPGA deposits the merged event fragments into the local buffer of the NIC and instructs it to transmit the packets by writing a DMA descriptor. At the system level, a throughput of 300MB/s per FRL for varying size fragments with 2kB average has been measured [CMSSFB], well above the requirement of 200MB/s.

2.1.2 ALICE

The ALICE dataflow architecture is displayed in Figure 8. Raw data are collected close to the detector in front-end readout cards (FERO) and transmitted via optical links (DDL) to data receiver cards (D-RORC), two of which are hosted each by one local data concentrator (LDC). The LDCs build sub-

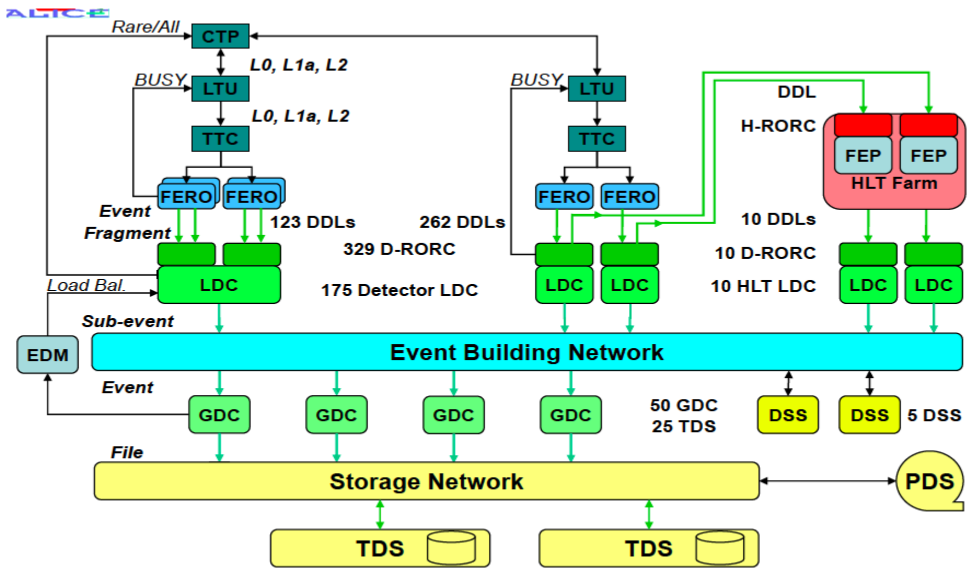


Figure 8: ALICE dataflow [ALICEJINST]

events (similar to CMS super-fragments) and output them to a GE network. Full events are assembled by global data concentrators (GDC) and ultimately shipped to mass storage, eventually based upon decisions from the HLT farm. The HLT trigger is fed from a subset of the detector only, via bypass links each established by one of the D-RORC DDL channels.

The ALICE component which is most equivalent to the ROBIN is the D-RORC [DRORC], which receives two optical inputs at 200MB/s each and stores the data into the memory of the host PC. Alternatively, one of the links can be used to send a copy of the input data to the HLT farm. As can be seen from the block diagram [ALICETDR] in Figure 9 the D-RORC does not contain any local memory or processor, but is just an I/O extension to the PC. Measured throughput into host-memory reached 484MB/s, well above the combined bandwidth of two optical links.

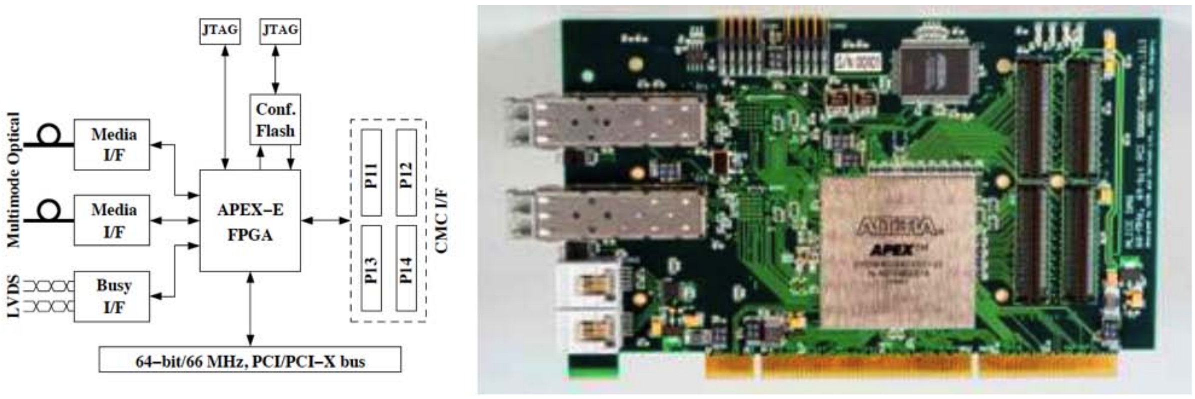


Figure 9: ALICE D-RORC [DRORC]

2.2 LHCb

The dataflow architecture [LHCBADD] of LHCb as shown in Figure 10 is relatively simple, as it does not use a separate L1[19] concentration layer. Instead, each of the 310 front-end (FE) units [LHCBTELL]

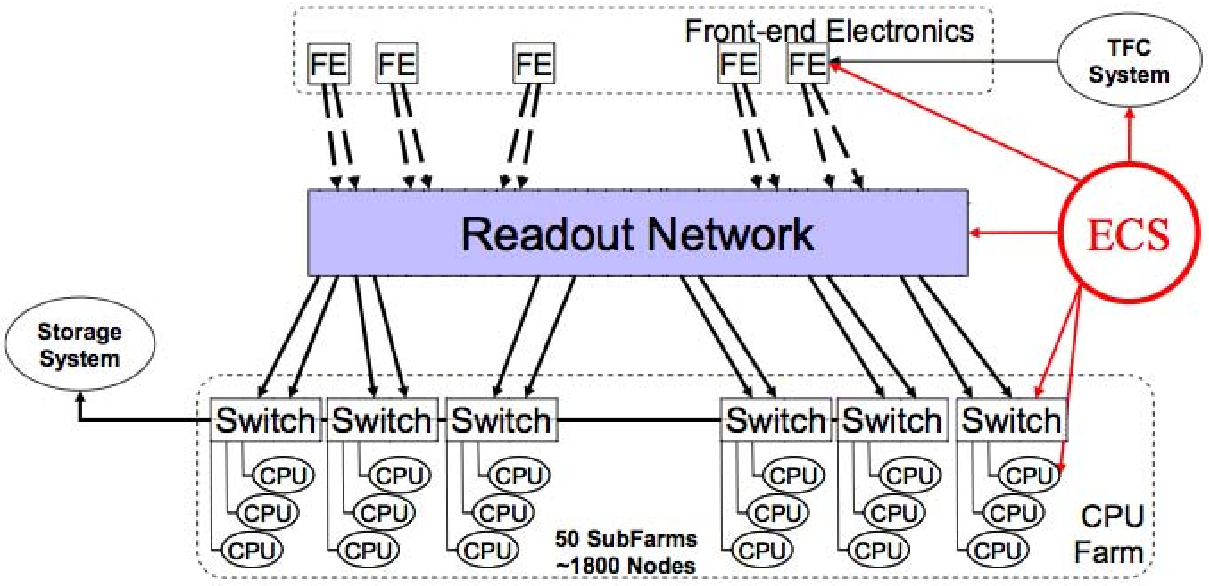


Figure 10: LHCb dataflow[LHCBADD]

19 In LHCb the first trigger level is called L0. However to avoid confusion the term L1 is used in this document.

located close to the detector collects data from a combination of up to 64 analog or 24 digital[20] channels respectively and transmits L1 accepted events via 4 built-in GE output ports to the readout network, which provides a total connectivity in the order of 2500 ports maximum. LHCb aims to keep the load on the GE network below approximately 70%, as experimental studies have shown that the risk for message loss is close to zero for loads below this value.

The FE card – called TELL1 – is a 9U VME card (Figure 11, right) which features several different I/O interface in addition to the data input and output ports. The ECS ports connects to the slow-control system of the experiment, the TTC port receives the central timing and control information. A throttling mechanisms slows down the L1 accept rate once the local buffers, which need to cover the 4µs L1 latency, become saturated. The PP-FPGAs (Figure 11, left) perform sub-detector specific processing of the raw detector data, while the SyncLink-FPGA is responsible to assemble and transmit event fragments.

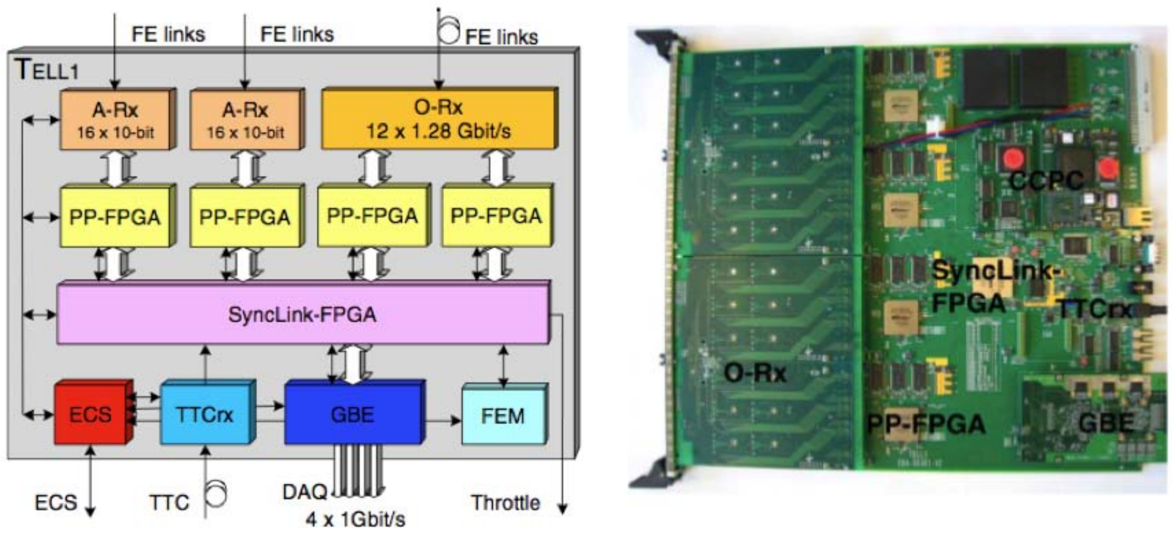


Figure 11: LHCb TELL1 [LHCBTELL]

Compared to the other experiments the TELL1 cards looks rather like the combination of front-end readout and DAQ interface, an approach which was also investigated by ATLAS in the course of the ROB-on-ROD project (see chapter 3.3.4).

2.3 *Summary*

The large LHC experiments CMS and ATLAS generate a few 100 particles every 25ns when operating at design luminosity. Particle tracks are recorded by different detector subsystems via 50 to 100 million sensor channels. A custom first level trigger located close to the detectors is used to select interesting events and provide them to the DAQ/HLT subsystem, which in turn is characterised by a an input rate of 100kHz and an input bandwidth in the order of 100GB/s. The event building processes require to connect roughly 1.000 sources to 1.000 destinations. The latter filter the events according to trigger algorithms, leading to an overall reduction in bandwidth and rate towards mass storage by a

20 The nominal speed of the digital inputs is 1.25Mb/s.

factor of 1000. All LHC DAQ systems are in general based on commodity technology - GE networks and standard PCs - but need some custom components at the input stage. In particular for CMS and ATLAS a scalable system was required due to financial issues, which allows to operate the system with reduced resources at reduced performance.

The CMS approach as described in this chapter transports the entire data volume – nominally events of 1MB size at a rate of 100kHz – via a layered architecture, following the traditional "PUSH" model. The initial layer employs 458 custom FPGA-based data converters (FRLs) and a Myrinet builder network (FB) made from 72 8x8 switches. The FB provides data from all sources on 8 parallel outputs which feed up to 8 slices of the subsequent stage. Each of the slices consists of 72 readout-units (RU) receiving Myrinet packets at 200MB/s, a GE network and 288 builder units (BU), which assemble the full events and run the event filter algorithms. The performance is scalable at 12.5kHz per slice.

ALICE and LHCb have somewhat different characteristics in terms of rates and event sizes and lower requirements with respect to total bandwidth. However, both share the same “PUSH” model for the dataflow as CMS does and employ similar architectures and technologies for the custom DAQ components.

In contrast, ATLAS introduced the intermediate L2 layer operating on RoIs – typically only a few kB at a rate of 100kHz – and transporting full events – about 1MB each – at a low rate of a few kHz. This architectural decision was taken with the aim for a full commodity dataflow network and under the assumption that processing power at the L2 will be relatively inexpensive at the time of operation.

All four experiments use custom FPGA components at the boundary between the sub-detectors and the dataflow system, which translate from proprietary signals into some networking standard. In case of ALICE and CMS the functionality is basically limited to this translation step. The LHCb component integrates additional functionality related to data processing and first level event selection. The ATLAS dataflow architecture is unique in its demands for buffering at the input and communication capabilities to serve the “PULL” model and requires high-performance FPGA technology combined with high-performance embedded processing – which are realised by the ROBIN component. The ATLAS approach is explained in the following chapter.

3 ATLAS DAQ

ATLAS is the other "large" experiment at the LHC, with requirements very similar to CMS (see section 2.1.1). This chapter introduces the ATLAS detector with the focus on the architecture of the ATLAS TDAQ system. The unique ATLAS approach to reduce the throughput requirements by a significant amount using the RoI-principle and sequential-selection in the L2 trigger stage is explained, due to the impact on the architecture of the dataflow. The baseline dataflow system of ATLAS and the ROS are described in detail, which define the requirements on and the environment of the ROBIN.

3.1 System View

The ATLAS detector is – like CMS – composed from different sub-detectors to identify charged particles (muons, electrons, …) and to measure other observables (vertex displacement, missing transverse energy, …). The layout of the magnets is different from CMS, which uses a single large solenoid magnet while ATLAS has a an inner solenoid and a group of toroid magnets further away from the centre. A sketch of the 7000t heavy detector is shown in Figure 12.

The seven sub-detectors of ATLAS together generate data on roughly 100.000.000 channels, the vast majority of belonging to the pixel detector. These channels are collected by electronics on or close to the detector and combined into 1600 ROD modules. The RODs are also connected to the L1 and the timing system of ATLAS and format data plus event identification into ROD fragments. A simple

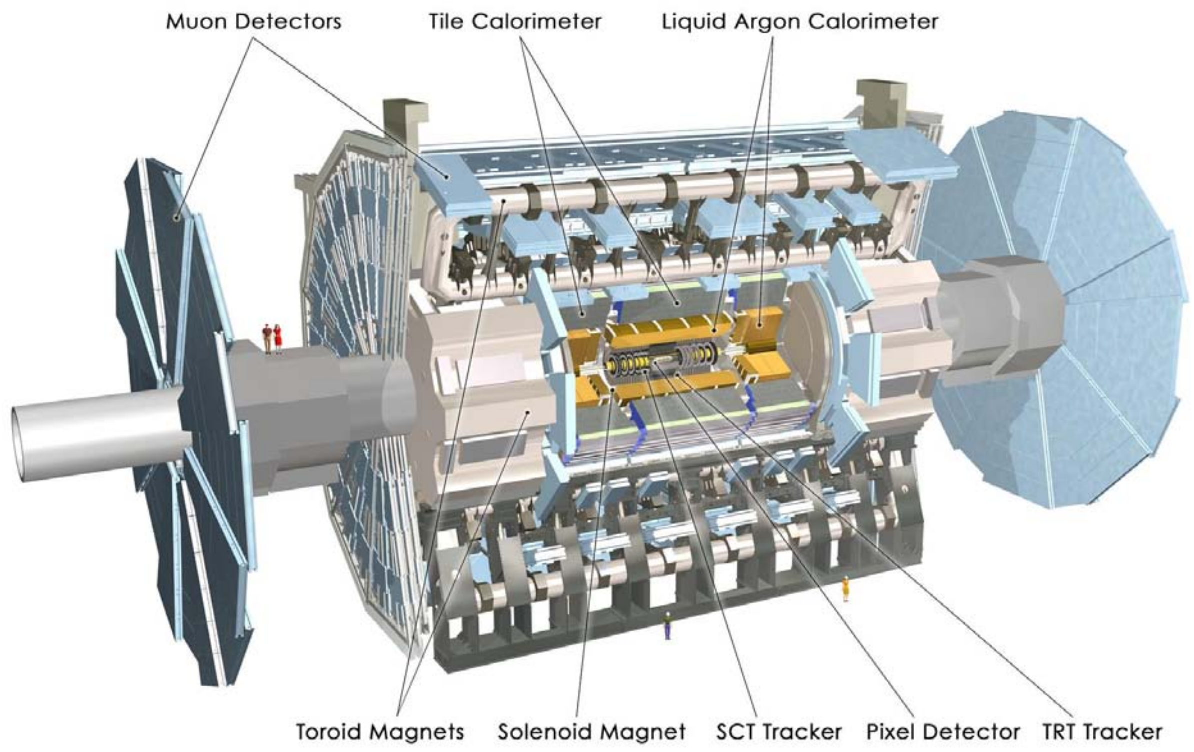


*Figure 12: ATLAS detector**

unidirectional link, the S-Link [SLINK], is used to connect the RODs with the TDAQ system. From the TDAQ point of view the ATLAS detector is a data source with 1600 channels, 100GB/s bandwidth[21] and 100kHz event rate. The task of ATLAS TDAQ is to select and store only 0.1% of the generated data volume.

The initial concept to implement the ATLAS TDAQ system as documented in the "ATLAS Technical Proposal" [ATLASTP] is shown in Figure 13. After L1 a digital buffer memory is used to store L1

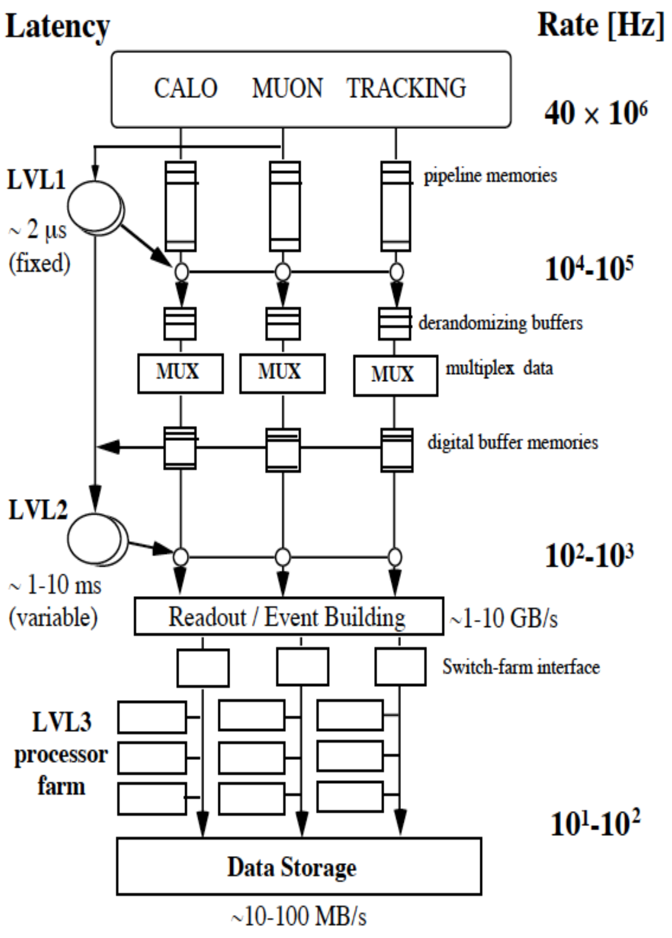


Figure 13: ATLAS TDAQ TP-version [ATLASTP]

data. An RoI-collection (RoIC) subsystem[22] copies the RoI-portions of some sub-detectors to the L2 system. An RoI is derived from the geographical information[23] attached to the L1 event identifier (L1ID), and defines a subset of a sub-detector, typically around 2%. The L2 is a modular system and operates in two steps. In an initial feature-extraction step all sub-detectors are individually and in

21 The maximum bandwidth is 1600 channels * 160MB/s = 256GB/s. However, the nominal fragment size is 1kB and even less for quite some of the sub-detectors.
22 The digital buffer memories in Figure 13 have two output paths, the main towards "Readout/Event Building" and a second one, which builds the RoIC, towards LVL2.
23 The particles generated in the collisions produce electrical signals in the detectors while they escape from the the interaction point. Any signal generated on one of the detect channels is called a hit.

parallel analysed for interesting data. The features are subsequently combined by a global decision step into the L2 decision, which is then distributed to the event building stage. At that time different approaches for the L2 implementation were discussed and investigated in a demonstrator programme [ATLDEMPROG] – a global L2 farm, the use of a local sub-farm[24] per sub-detector and the use of custom data-driven L2 processors based on FPGAs either replacing the local sub-farms or in a hybrid fashion together with sub-farms. The different L2 options also affected the design of the readout subsystem, e.g. the latency of the data-driven L2 was much lower than that of the global and local farm approaches. However, it required the fast distribution of the RoI-information to the readout buffers and the RoIC subsystem. The network technologies proposed for the L2 farm interconnects were ATM, SCI and Fibre-Channel, none of which has a significant market share today. Also, three different networks were to be used to transport event data from the buffers to the L2, from the buffers to the L3 and to transport L2 decisions.

A significant simplification for the TDAQ architecture was achieved by the introduction of the sequential selection strategy [SEQSEL] for L2. Sequential selection takes advantage from the fact that all important physics events require the presence of signals in the Muon detector and the calorimeters. Thus looking for Muons first allows to reject ¾ of the events at the L2, as shown in Figure 14. A subsequent check for electrons, photons and jets enables rejection of another 60% of events. The sequential selection strategy reduced the requirements on the system throughput and allowed to merge the separate, sub-detector specific L2-subfarms of the initial architecture into a single, uniform L2 farm. The required processor farm size for the sequential execution of the L2 was estimated by modelling [PAPMOD] to be in the order of 200 machines. However, sequential selection also has a drawback, which is the increased complexity of the dataflow architecture. The traditional “push”-mode has to be replaced with a “pull”-mode dataflow, which needs relatively advanced interactions between the subsystems (see chapter 3.2).

A test-bed [PILPRO] was setup to verify the performance of the associated dataflow system with different network and readout buffer implementations[25] [ROBCPLX]. The range of technologies investigated is summarised in Table 4.

Technology	**Options**			
FPGA	High-end, provides core functionality	Medium, I/O plus auxiliary functions	Low-end, I/O only	
Processor	Host only	Local DSP	Local 32-bit Microcontroller	High-end SMP host
Bus	PCI (PMC mezzanine)	PCI (standard format)	VME	
Network	ATM	SCI	Gigabit-Ethernet	
Optical links	2.5Gbit/s single fibre	Multiple fibre 15Gbit/s (Paroli)		

Table 1: Pilot Project Technologies

24 At the time of the TP approximately 300 processors would have been needed for a local sub-farm.

25 During the pilot project phase the terminology “ROB Complex” was used which corresponds to the current ROS.

The pilot project defined the ROS to be the aggregation of a number of input units receiving data from the RODs, associated buffers, some intelligent controller(s) to manage the event fragments and handle requests and a number of output units interfacing to the L2 and EB network. The range of configurations investigated covered the most simple one (single input unit, single controller, single output) up to the "Active ROB Complex" [AROBC] where many input units were handled by a powerful SMP host. The latter concept also included data-processing at the level of the readout subsystem.

The results[26] obtained during the pilot project phase showed that the requirements of the ATLAS dataflow could be satisfied in principle with the proposed ROS architecture, however a solution was needed in order to achieve the goals for density and cost. From the prototype implementations the ones with a large fraction of functionality implemented in FPGAs provided the best performance, while the processor-oriented suffered from the high rate. Two other areas were identified as potential bottlenecks: the memory subsystem – in particular if shared by processor and buffer – and the parallel bus, specifically the drop in available bandwidth for high message rates. All issues could be addressed by the design of the subsequent ROBIN.

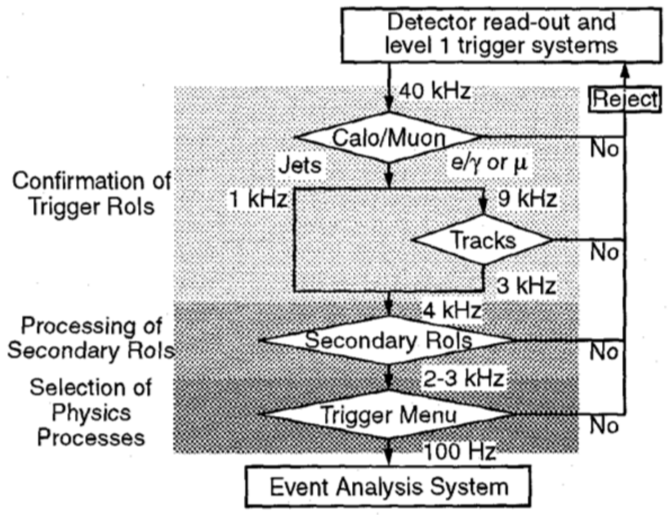

Detector read-out and level 1 trigger systems
40 kHz
Reject
Calo/Muon
No
Jets
e/γ or μ
Confirmation of Trigger RoIs
1 kHz
9 kHz
Tracks
No
3 kHz
4 kHz
Processing of Secondary RoIs
Secondary RoIs
No
2-3 kHz
Selection of Physics Processes
Trigger Menu
No
100 Hz
Event Analysis System

Figure 14: Sequential Selection [SEQSEL]

As a consequence, the ROS consisting of a standard COTS PC equipped with a number of FPGA-based ROBINs became the baseline implementation (see chapter 3.3.3) as documented in the ATLAS Trigger and DAQ Technical Design Report (TDR) [ATLASTDR], which addressed the issues mentioned above in the following way:

- The use of COTS PCs reduces cost
- Concentration of multiple input links per PCI card allows to build a compact system

26 The pilot project considered two operating conditions – low and high luminosity – of the LHC, with corresponding L1-rates of 40kHz and 75kHz respectively. This is in derivation of the standard 100kHz/high luminosity case assumed elsewhere in this work.

- A central FPGA per ROBIN enables high-rate and high-bandwidth I/O handling
- An auxiliary local processor provides flexibility
- An auxiliary local network interface provides an upgrade path for bandwidth- or rate-intensive request schemes

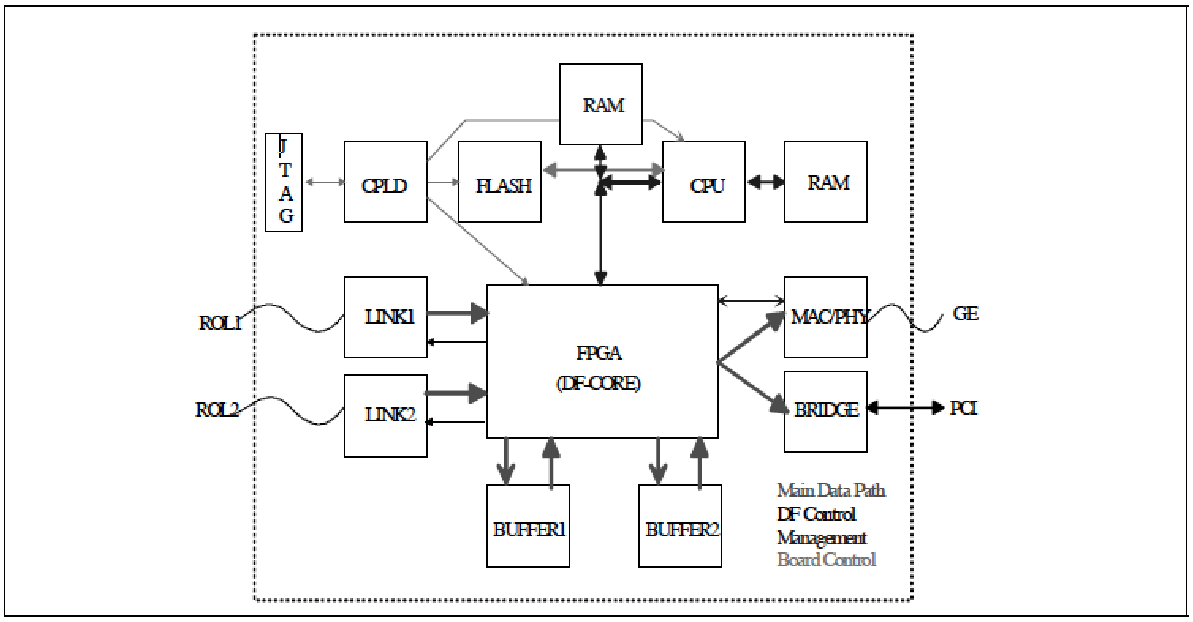


Figure 15: TDR ROBIN Blockdiagram

Figure 15 illustrates the TDR ROBIN design, with a central FPGA comprising the main data path from the two input links to the two buffer memories and onwards to the alternative output interfaces PCI and GE. FPGA technology – which is described in chapter 4 – enables to implement I/O interfaces, memory controllers, direct-memory-access (DMA) engines and various other functions on a single device in an optimised and highly efficient way while maintaining flexibility and upgradability. The local processor is attached to the FPGA as well, but separately from the main data path. The final design of the ROBIN is described in chapter 5.3 .

3.2 Dataflow

The ATLAS Dataflow system is responsible to move the detector data from the interface to the detectors – implemented by the ROLs – up to permanent storage, attached via a standard network. A set of subsystems constitute the main data path (Figure 16, center) from the RODs to the mass storage. Additional subsystems provide control and configuration functionality (Figure 16, left and right). Due to the specific properties of ATLAS TDAQ the dataflow uses a mixture of “push” and “pull” mechanisms in order to transport data from one stage to the next one.

Starting at the RODs, event fragments are pushed to the ROBs. Concurrently, RoI information is pushed from L1 to the RoIB and onwards via the L2SVs to the L2PUs. The L2PUs use the L2 network to pull event fragments according to the RoI information via the ROS from the ROBs. The results from L2 are pushed via the L2SVs to the DFM, which pushes “accept” decision to the SFIs and “reject” decisions to the ROS. The SFI again pull all event fragments from the ROS via the EB network, build the full events and push them to the EFPs. Finally, the events accepted by EF are pushed via the SFOs to mass storage.

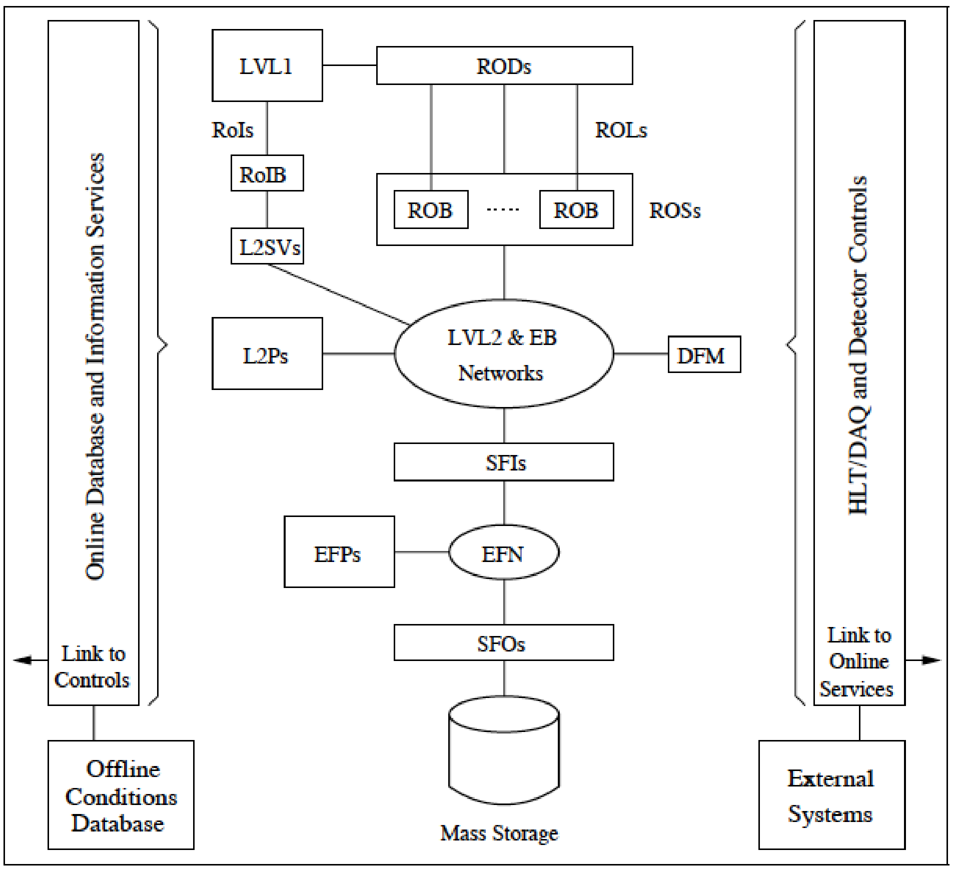


Figure 16: Schematic layout of ATLAS TDAQ system [ATLASTDR]

In the baseline dataflow model the ROBs are installed in ROS-PCs, each of which typically houses 4 ROBINs each representing 3 ROBs, so one ROS-PC serves 12 channels. The ROS-PC forwards all relevant requests to the ROBs via the PCI-bus and combines the responses to SFI-requests into larger fragments. In addition, the ROS-PC is responsible for configuration and monitoring of the installed ROBs.

Alternative scenarios bypass the ROS-PC for L2 and/or SFI requests and pull fragments directly from the ROBs, via their private Ethernet ports.

Apart from the bandwidth requirements ATLAS dataflow requires a network with good real-time performance, high reliability and high rate capabilities in particular on the L2 network. The requirements are largely comparable to the requirements of enterprise-size commercial networks where Ethernet is the de-facto networking standard. However, there is one difference: ATLAS DAQ requires low latency and reliability at the same time, while most of the typical Ethernet areas require only one of them. Ethernet networks follow the best-effort principle and are not reliable per-se, like

Myrinet is. Client-server applications for example introduce reliability by using a reliable protocol like TCP/IP over Ethernet, which can add significant delays to the transmissions. On the other hand, multimedia applications like IPTV or video conferencing are sensitive to latency but quite tolerant to packet loss and the unreliable UDP protocol is used frequently here.

The L2 system has an intrinsic latency of a few ms, caused by the execution time of the L2 algorithms, which defines the buffering requirements at the ROS. Any additional latencies introduced by the network increase the required amount of buffering capacity and make the system less stable. The use of a reliable protocol like TCP is not a general solution, due to the relatively long retransmission delays[27] in case of packet loss and its unavailability[28] on the ROBINs. An additional requirement on the reliability of the network at the low level is the use of multicast messages to distribute the DFM results to the ROS. Although Ethernet does not appear to be the ideal candidate for the ATLAS dataflow network from a technical point of view it was selected for reasons of cost, general availability, ease of use and expected longevity, which is important for an operation time of 10 years or more.

The total size of the ATLAS TDAQ system from the dataflow perspective is defined by the number of external data sources and by the number of networking nodes. The number of ROBINs and ROSes corresponds to the number of detector links and is fixed. The number of L2PUs and EFPs define the trigger capabilities and thus influence[29] the maximum L1-rate the system can handle. The expected numbers for the individual components is given in Table 2.

Component	Instances in final system	Comment
ROL	1600	
RoIB	1	RoI: 2% of full event
ROBIN	600	100kHz L1-rate
ROS	150	Separate network ports for L2 and EB
L2SV	10	
L2PU	500	Dual-CPU systems, ~100 events/s per CPU
DFM	35	
SFI	100	Full event size ~1.5MB
EFP	1600	~1 event/s per CPU
SFO	30	
L2 network nodes	700	100 kHz, RoIs
EB network nodes	300	3 kHz, full events

Table 2: Data-flow components (TDR)

The large number of network nodes cannot be attached to a monolithic magic box which provides the full connectivity for more than 1000 ports. Instead, the Ethernet approach for interconnecting nodes is

27 The TCP retransmission timeout is typically set to 3s, which results in delays in the order of seconds.

28 TCP requires to monitor every logical connection, which is too resource consuming on the ROBIN, where several hundred simultaneous connections may exist.

29 Good linear scaling of L2, EB and EF documented in [ATLASTDR]

to cascade a number of units called switches, each serving up to a few hundred ports in a cross-bar fashion. Significant effort has been put by ATLAS into the analysis of Ethernet behaviour and equipment, well documented in [STANCU]. As mentioned above, the main issues with Ethernet – apart from the connectivity – are latency and message loss. Both factors strongly depend on the technology of the switches. In principle, a switch has a unit handling the input (ingress), a unit handling the output (egress) and the cross-bar[30] connecting input and output. The routing path over the cross-bar is determined by the ingress unit from the header of every individual Ethernet packet, which starts with a source and destination address identification. The switch constantly monitors all source addresses and builds a map of addresses and ports. If a destination address is already in the table, the packet is routed to the corresponding port, otherwise it is replicated and sent to all ports. The latency introduced by this routing process is relatively small, as the evaluation of the routing path starts while the packet is still being received from the source. An obvious problem is the case where multiple source are sending to the same destination (so called funnel traffic) and such exceeding the egress bandwidth limit. While Ethernet allows the switch to simply drop packets in this case the most common solution is to queue packets at the input. If an input queue become full an Ethernet flow-control message is sent to the source, asking to pause the transmission. A common complication in Ethernet switches is head-of-line blocking, which occurs when an congested egress is blocking an ingress queue which contains also packets for another egress port. That egress can be idle in the worst case despite the fact that packets are available for it in an ingress queue. Some switches improve the situation by providing separate ingress queues[31] for some or all egress ports but at high load both loss and latency of Ethernet switch are inherently indeterministic.

Various test-beds have been set up to study the behaviour of the individual components and the performance using different networking protocols. To achieve reasonable sizes for the test-beds typically several components had to be emulated. For example, every ROBIN has built-in data-generators able to provide event fragments of arbitrary size at full speed. FPGA-based data-generators [GETB] [ETHERT] providing up to 128 ports and programmable NICs providing up to 16 ports were used to create network traffic with relatively simple patterns at high rate. More complex traffic pattern were created using PCs with special test programs. Using such systems, throughput in the order of 10% of the final system has been demonstrated for the baseline bus-based [BASEDF] readout architecture. Large-scale tests [LARSC] were performed on computing clusters with several hundred of machines, emulating different portions of the dataflow system up to the full size in most areas. The analysis of the congestion management of various switches indicates that at loads up to 60% of the nominal switch capacity the rate of lost message is virtually zero[32] for random traffic patterns, if the components are properly selected [ETHER]. Hence the capacity of the ATLAS dataflow network is tailored such that the load on the switches stays below this margin. However, care must be taken that the actual traffic patterns do not derive too much from a random distribution, such overloading certain switch ports. A potential problem is the event building step where an SFI needs to get data from a large number of sources. If the SFI would issue all data requests simultaneously the response packets would

30 There are also switches which use a shared-memory instead of the cross-bar, however the required memory bandwidth poses a limit on the throughput and number of ports of such implementations.

31 This is also called virtual output queuing.

32 This is in line with the results obtained by LHCb (section 2.2).

certainly introduce head-of-line blocking in the switch with the associated effects, increased latency and packet loss and ultimately reduced system performance. For example, a message loss rate of 0.01% results in a performance reduction of 20% of the SFI [STANCU]. The implemented strategy at the SFI therefore uses a random delay after every data request.

Another consideration for the design of the ATLAS dataflow network was the communication pattern – certain nodes communicate with each other and others do not. For example, the ROS nodes need to communicate will all L2PUs and SFIs but the SFIs never communicate with the L2PUs. Also, nodes of the same kind do not communicate with each other. The analysis of the corresponding bandwidth requirements shows that concentrating switches can be used in to aggregate ROSes or L2PUs. The small concentrating switches are then connected to the central switches, which also connect to the SFIs. The uplinks from the concentrators to the central switches use either several GE or a single 10GbE link. Figure 17 Shows the layout of the dataflow network using concentrator switches for the L2PUs, for the control nodes (DFM, L2SV and pROS) and for some of the ROSes. Two large central switches with a nominal capacity in the order of 250Gbit/s each build the core of the network. L2 and EB traffic is mixed in this scenario which provides also a certain degree of fault tolerance, as the system can continue to run even after the failure of a central switch, although at a lower rate. The alternative scenario where one central switch is used for EB and the other one for L2 was preferred earlier as it keeps the subsystems separate, however at the expense of missing flexibility and fault tolerance.

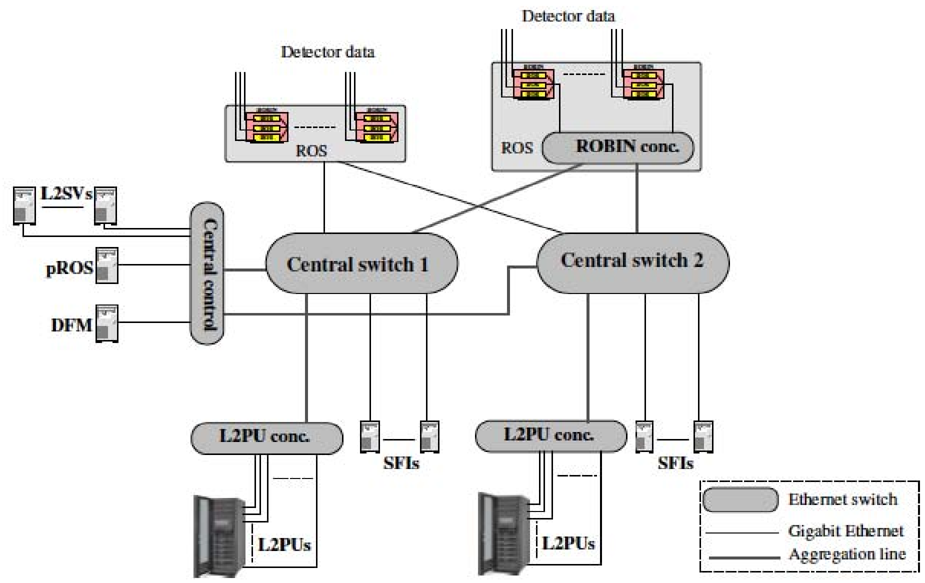


Figure 17: Dataflow network [STANCU]

The networks have been characterised using UDP and TCP and both protocols can be used concurrently in the system. In general, network latencies and message loss can be kept at an acceptable low level using standard – sometimes selected – network components [DFNET][DFROS]. Message loss on UDP based L2 data-requests is handled at the L2 application level, by re-requesting the data or by generating a “forced-accept” decision. Message loss on multicast delete messages is handled at the ROS by a garbage-collection mechanism (see chapter 3.3.2). The buffering requirements introduced by the L2 trigger latency depends on the actual event (due to the sequential selection) and was estimated most recently to below 100ms in a relatively large test setup consisting of 134 ROS, 32 SFI and 130 HLT nodes, using 4.000 simulated events [L2PROC]. The distribution of the processing time is shown in Figure 18.

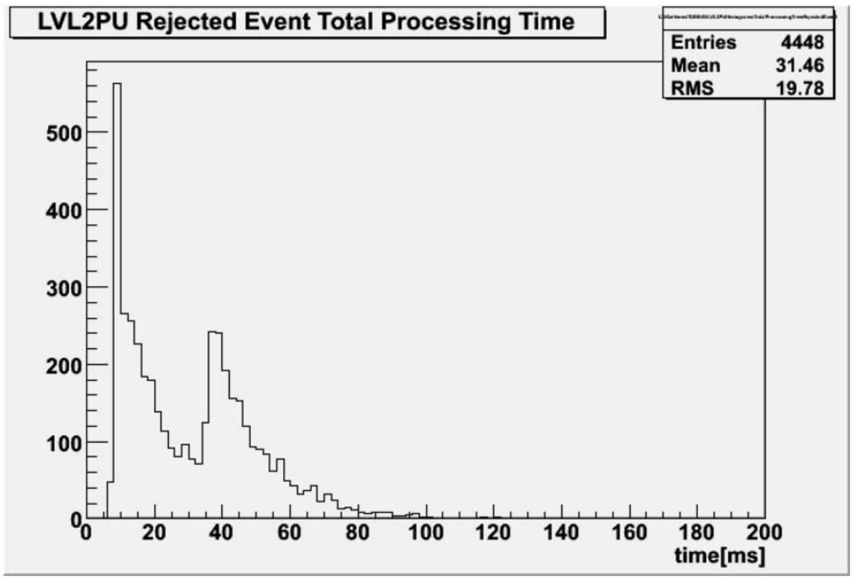


Figure 18: L2 processing times [L2PROC]

3.3 ROS

The ROS implements the buffering capabilities of the DAQ and the interfaces to the detector/L1 system on one side and to the HLT system on the other side. Event fragments are arriving from the detector/L1 system through the ROLs at the L1-rate of up to 100kHz and with a nominal bandwidth of 160MB/s per link. The actual fragment sizes depend on the type of sub-detector and vary typically between 400 and 1200 byte ([ATLASTDR], see chapter 8.4). Fragments generated for detector calibration purposes may be much larger (approx. 10 - 100kB), however occur at a very low rate. A full event is composed from all fragments corresponding to the same L1ID and has a typical size between 1MB and 1.5MB.

A baseline ROS architecture [Ibid.] has been developed, which concentrates a significant number of ROL channels into a single PC via the PCI-bus, satisfying the typical requirements. For a 12-channel ROS-PC the total L2 request rate varies between 300Hz for the TRT sub-detector and 18.6kHz for the electromagnetic calorimeter [Ibid.]. While under typical conditions the L2 and EF requests average to 6kHz per channel at a L1-rate of 100kHz the theoretical worst-case rate[33] on an individual channel is around 21kHz [MADROS]. The bandwidths related to the typical and worst-case request rates are in the order of 10MB/s and 33MB/s per ROL respectively. Enhanced ROS performance is achieved either by reducing the number of channels per PC or by employing additional networking connectivity

33 The 21kHz are composed of 18kHz L2-rate plus 3kHz EB-rate. Under normal conditions it is very unlikely that the full L2-rate goes to a single channel.

bypassing the PCI-bus. The ROBIN – the only custom component of the ATLS dataflow apart from the RoIB – was designed to handle and buffer the input data with the flexibility to interact with the ROS-PC as well as directly with the dataflow system via a private network interface. In addition to the tasks above, which are related to the transport of detector data, the ROS interfaces to the configuration database and run control, the monitoring system and the detector control system (DCS) [ATLDCS]. The requirements on the ROS are summarised in a ROS user requirements document[ROSURD].

3.3.1 Event handling

In ATLAS, events are identified by a 32 bit event number composed from a 24 bit identifier generated by the L1 trigger – the primary L1ID – plus an 8 bit event-counter-reset (ECR) value, incremented by the RODs upon the wrapping of the event identifier to 0. For simplicity the event number is normally and in this thesis referred to as L1ID. A design constraint limits the frequency of the ECR to the range from 0.1Hz to 1.0Hz. As a result, the maximum time between two zero-crossings of the event number is 256s, equivalent to 25.6 million events at a L1-rate of 100kHz. The minimum time is 25.6s. As a typical ATLAS run (a period of continuous operation) can extend to several hours the L1ID is not necessarily unique for all events and an additional mechanism needs to be put in place for event identification. As the minimum time covers the range of L2 and EF latencies the additional information is inserted at the event building stage by adding appropriate time-stamp information. At the ROS level the limited amount of buffering space requires to delete events as soon as possible. This is done by explicit delete messages distributed by the DFM, after the event has been rejected by the L2 or processed by the EF. To reduce the rate for delete messages they are sent out via a multicast mechanism typically in groups of 100.

As stated in chapter 3.2 the dataflow system is designed to minimise packet loss on the network switches, however losses are not fully prevented. While data requests are point-to-point interactions and can be protected by a reliable protocol this is not the case for delete messages. Lost delete message lead to orphaned fragments in the ROS and reduce the amount of buffer space available. While the event numbers at the ROS level restart at 0 after a maximum of 256s it cannot be guaranteed that all orphaned events will be replaced, as the L1ID does not have to be strictly sequential. Therefore, a "garbage collection" mechanism is required to clean up the buffer. The implementation of this mechanism requires to distribute the "oldest" valid L1ID in the dataflow system, which is piggy-backed to the delete messages. The loss of a message is detected by a jump in the message sequence numbers. Once a lost delete message is detected, the ROS compares the oldest valid L1ID to the most recent L1ID received from the ROLs, creates a range of valid L1IDs and deletes all fragments with event numbers outside of this range. The actual garbage collection procedure is executed on the ROBINs, which need to do a time-consuming scan of their entire buffer in order to build the list of stored fragments. To avoid excessive load the garbage collection is executed only when the buffers on the ROBIN have reached a certain filling level.

3.3.2 Configuration and monitoring

Every ATLAS run is associated with a set of configuration parameters, stored in the global ATLAS configuration database. The range of parameters is very broad and includes calibration values for detector front-end electronics, RoI-mappings and IP-addresses. A number of these parameters controls the behaviour of the ROBINs, distinguished into regular parameters and expert parameters. The regular parameters include buffer memory page sizes[34], truncation limit, IP-address, channel identifier etc. The expert parameters include values which need to be modified in order to enable or disable particular functionality required to perform specific tests. During a regular run all expert parameters are set to their default values. The configuration of the ROBIN is controlled by the ROS-PC via a the normal requests/response mechanism.

The ATLAS TDAQ system requires a detailed online view of all activities in the system, in order to properly react to any malfunction. Thus, every subsystem has to provide functions related to operational monitoring. At the ROS level, operational monitoring gathers statistics information of received and lost messages, of buffer pages, processed fragments and of errors and histograms of buffer occupancies and fragment sizes. Most of this information is prepared by the ROBIN and transported to the ROS-PC via the regular requests/response mechanism.

3.3.3 Baseline bus-based ROS

According to the baseline bus-based[35] architecture, the ROS is built from 150 PCs installed into racks with up to 12 PCs each (see Figure 19, front and rear view of rack). Each ROS-PC[36] attaches to 12 ROLs with a total input bandwidth of almost 2GB/s at a fragment rate of 1.2MHz. On the DAQ/HLT side the ROS has to handle the requests from L2 and EF for event data[37] and event rejection[38]. Connectivity to the L2 and EB networks is implemented with a 4-port NIC, which uses 1 port for each of the networks in the default configuration.

An additional NIC port is used for the operating system's network interconnection and for control and configuration. The bandwidths corresponding to the typical conditions are 60MB/s per network. The performance requirements as documented in the ATLAS TDR relate to fractions of the L1 input rate and translate for the standard ROS to 4kHz EB plus 12kHz L2 (RoI size of 1) at 100kHz L1 rate and 1kB fragments. A “hot-spot” condition was defined with 17% of L2 (RoI size of 2) and a fragment size of 1.4kB. Early measurements for a typical 12 channel ROS, equipped with data emulators, showed that the standard requirements were within reach (94kHz L1 rate achieved). For the “hot-spot” ROS either the EB rate had to be lowered to 2% or the L1 rate to 75kHz. As the “hot-spot” condition applies only to a few ROS-PCs attached to the electromagnetic calorimeter, the proposed solution at the time of the TDR was to reduce the number of ROBINs to 3 or even 2 in the few ROS-PCs affected.

34 The buffer memory page size on the ROBIN has a default values of 2kB. Calibration runs for example can use larger pages.

35 The baseline bus-based architecture uses the ROS-PC to interact with the DAQ/HLT system via the network and to select and collect event fragments from the individual channels of the ROBINs. An alternative switch-based architecture allows the ROBINs to directly communicate with some of DAQ/HLT components via a private network interface. This scenario is intended for special cases with more demanding performance requirements.

36 The terminology of ATLAS TDAQ sometimes uses ROS synonymous for ROS-PC.

37 An L2 request addresses a single or a few channels, while an EB request addresses all channels.

38 Event rejection (delete) messages are issued by a separate DAQ component, the dataflow-manager (DFM).

Figure 19: Racks with ROS-PCs*

A further, even more demanding use case was defined after the TDR with a L2 request rate of 18 kHz on all channels plus 3kHz of EB, at 100kHz L1 rate. Going even beyond that, a ROS could theoretically be configured for 100% readout ratio, which would make it look somewhat like the CMS FRL unit (see chapter 2.1.1.4).

3.3.3.1 ROS-PC

The typical ROS-PC comprises a single CPU, 1GB of main memory, a 4-port PCIe NIC and 4 ROBINs, as shown in Figure 20. The mainboard[39] provides multiple PCI-X buses such that a

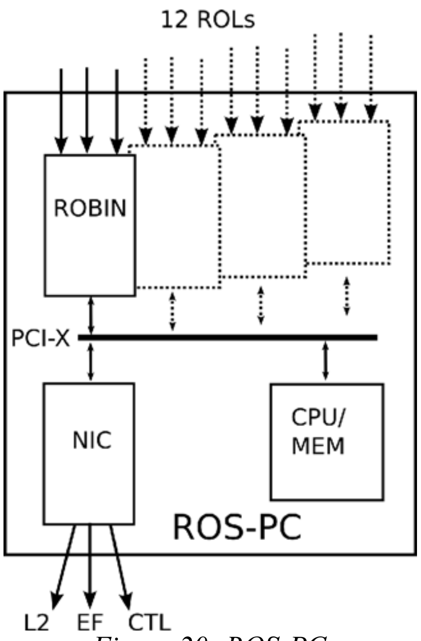


Figure 20: ROS-PC

39 Supermicro X6DHE-XB, http://supermicro.com/products/motherboard/Xeon800/E7520/X6DHE-XB.cfm

maximum of 2 ROBINs are connected to the same bus. This way, the maximum output bandwidth[40] of a ROBIN can be fully utilised. The NIC is placed on a separate PCIe bus. Both the motherboard and the chassis were selected after testing a number of different machines for performance and stable operation. Special attention has been paid to the quality of the power supply for PCI cards and to the cooling facilities. Concerning the power supply it was observed that several motherboards did not provide proper 3.3V if 4 ROBIN cards were installed. In some cases the voltage dropped below 3.0V, which triggered the under-voltage reset circuitry of the ROBIN and prevented the boards from starting. The thermal behaviour of the PC was tested by installing up to 5 custom "load-cards" (Figure 21) with configurable dissipation between 1W and 28W. The air-flow generated by the front fan of the case passes across all installed cards and exits through special openings at the rear of the case. The card temperature has been measured for different cooling conditions (case open/fan on, case closed/fan on, case closed/fan off) at a room temperature of 35°C. The temperature difference between the edges and the centre of the cards on one hand and between cards at different positions is in the order of 10°C for the situations where the fan is active, and the maximum temperature is around 65°C, which is acceptable considering the high room temperature. If the fan is turned off, the maximum temperature comes close to 80°C which is beyond the spec for many components. A regular monitoring of the temperature of the ROBINs can be done with the on-board temperature sensor. Also, there are sensors on the motherboard which can be used to detect failure of the cooling system in the PC.

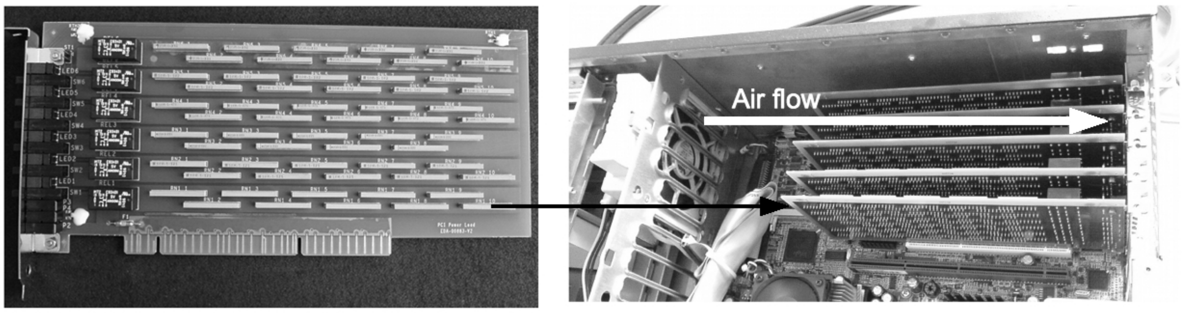


Figure 21: PCI load board

3.3.3.2 Message passing

The ROS-PC runs a standard 32-bit Linux kernel, however with a patch[41] that enables applications to acquire a large amount of physically contiguous memory. The ROS application uses this memory to build a pool of fixed-sized memory pages as destination buffers for ROBIN responses. The communication between the main CPU and the ROBINs works in a messaging passing fashion: the application sends a request to the ROBIN and the ROBIN returns a response. The requests follow a standard format, which comprises the request code, a channel identifier, a sequence number, a destination address, a length field and eventually any data related to the request. Requests are written into a dual-ported memory area on the ROBIN, which is mapped into PCI memory space. A request descriptor identifying the length and the location of the request is written to a separate memory area, which is implemented by a FIFO on the ROBIN. The memory sizes of FIFO and dual-ported memory

40 The output bandwidth of a ROBIN is 256MB/s, the PCI-X bus supports 512MB/s at 66MHz.
41 The patch is called "bigphysarea", see e.g. http://lkml.indiana.edu/hypermail/linux/kernel/0411.1/2076.html

form a 32 entry deep hardware queue for requests from the ROS application. The number of available entries is maintained by the ROS application. To provide the destination addresses, the ROS application selects a buffer from the memory pool. A ROBIN configuration parameter assures that the actual fragment size cannot go beyond the size of the buffer area. This is done by setting an upper limit to the maximum number of fixed-size memory pages the ROBIN may use for any event fragment. Any fragment exceeding that size is truncated by the ROBIN during reception from the link.

3.3.3.3 ROS Software

The ATLAS TDAQ online software is made up from a very large number of individual packets, which control trigger, DAQ, database access, monitoring etc. Alone the dataflow section, which is the main framework for the ROS and the ROBIN consists of 200 software packages. Among these, 27 deal with the ROS, 2 contain the application and boot code of the ROBIN and 1 covers the ROBIN FPGA code. Three of the ROS packages are relevant for the ROBIN and contain device drivers, a library and a number of applications. These packages are currently maintained by the CERN ROS software team and the group at RHUL. The Mannheim team has been and will be active in this area as well, however there is no manpower available at this time.

The main device driver performs the initialisation of the kernel structures related to PCI devices and makes the resources available to user applications. The driver also accepts interrupts from the ROBIN and provides the hooks for a corresponding user level interrupt handler. Additionally, it provides some debugging information which is available through the Linux "/proc" filesystem and which reports for every ROBIN card the serial number, the version number of FPGA and application code, the real-time status of the S-Link inputs, occupancies of the request queues and the values of the mailbox communication registers. Apart from the debugging interface the device driver is very generic and leaves most of the device specific functionality to a user level library and the associated applications. This approach makes the device driver less sensitive to modifications of the ROBIN functionality, which is advantageous as installing a new device driver has to be done by the system administrators while the applications can be updated by regular users.

A second device driver is available which provides a standard serial port to the host. This serial port is implemented at the hardware level by the the ROBIN FPGA, which in turn attaches to a serial port of the ROBIN processor. The purpose of this driver is to gain access to a ROBIN terminal port without attaching a cable. The latter is not practical, as the ROS-PC has only one internal serial port. Changing cables or adding an external USB-to-serial expander are not viable options under normal operating conditions. The serial interface is then used for testing and debugging purposes, for example the test suite (see below) uses this feature to set configuration values and to retrieve status and debug messages. There are two reasons to keep this driver separate from the main driver. Firstly, the functionalities are completely different and the serial port is only used for debugging and maintenance. Secondly, the serial driver interferes with the firmware upgrade procedure and must be unloaded beforehand. To minimise the chance for a system crash due to this interference, the serial driver is by default not loaded.

In addition to the two ROBIN device drivers there are other drivers used by the ROS software, for example to allocate physically contiguous memory for the communication with the ROBIN.

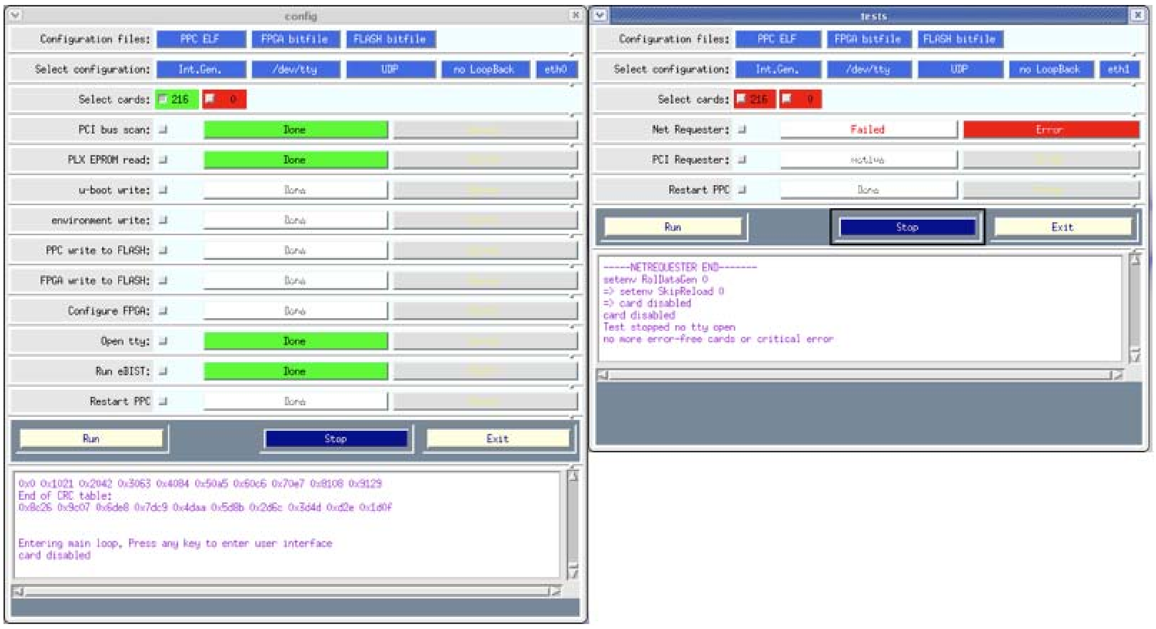
The library used by the drivers and the applications contains functions related to the following areas:

- Allocation of boards and resources, memory mapping
- Access to the PLX PCI-bridge device
- JTAG access to FPGA and CPLD
- Exception and interrupt handling
- Handling of FPGA bitstreams, incl. FPGA configuration
- Message passing

The application “robinconfigure” is responsible to create the link between the ROBIN cards and the memory pool used for the response messages. This application is normally called by the device driver during initialisation, but can be used later on to modify the memory settings. Every ROBIN consumes approximately 80kB per channel from a pool of 1MB of physically contiguous memory. This large contiguous memory is obtained from the Linux operating system at boot time via the “bigPhysArea” patch.

The utility “robinscope” is the main test and debug tool for the regular maintenance and makes the full functionality of the PCI message passing interface available to an expert user. The configuration parameters and monitoring values can be retrieved, the configuration can be modified, fragments can be uploaded to the ROBIN in different emulation modes and subsequently requested. This includes the generation of incorrectly formatted fragments and the check for proper error handling plus a simple performance tests using the internal fragment generator.

Further test utilities are “robinTestSuite” and “robinstress”. “RobinTestSuite” is basically a tool for testing at the factory level.



config

Configuration files:	PPC ELF	FPGA bitfile	FLASH bitfile		
Select configuration:	Int.Gen.	/dev/tty	UDP	no LoopBack	eth0
Select cards:	216	0			
PCI bus scan:	Done				
PLX EPROM read:	Done				
u-boot write:	Done				
environment write:	Done				
PPC write to FLASH:	Done				
FPGA write to FLASH:	Done				
Configure FPGA:	Done				
Open tty:	Done				
Run eBIST:	Done				
Restart PPC	Done				

```
0x0 0x1021 0x2042 0x3063 0x4084 0x50a5 0x60c6 0x70e7 0x8108 0x9129
End of CRC table:
0x8c26 0x9c07 0x6de8 0x7dc9 0x4daa 0x5d8b 0x2d6c 0x3d4d 0xd2e 0x1d0f


Entering main loop. Press any key to enter user interface
card disabled
```

tests

Configuration files:	PPC ELF	FPGA bitfile	FLASH bitfile		
Select configuration:	Int.Gen.	/dev/tty	UDP	no LoopBack	eth1
Select cards:	216	0			
Net Requester:	Failed	Error			
PCI Requester:	Active				
Restart PPC	Done				

```
-----NETREQUESTER END-------
setenv RolDataGen 0
=> setenv SkipReload 0
=> card disabled
card disabled
Test stopped no tty open
no more error-free cards or critical error
```

Figure 22: ROBIN TestSuite

It steers and monitors the ROBIN BIST procedure via the serial interface (cable or driver) and tests the network interface with the help of external network requester program. It also interfaces to the low-level configuration tool to enable the factory programming. Multiple ROBINs can be processed in parallel while the results are displayed on a simple GUI (Figure 22). The purpose of “robinstress” is to request fragments over PCI at the maximum rate while checking the fragment CRC code in order to verify data integrity. Normally, there shouldn't be a problem with this on PCI but there were a number of incidents as described in section 6.3 .

The resident firmware of the ROBIN has different sections, which are all contained in a single FLASH memory device. The tool “robin_firmware_update” allows to access the FLASH. The FPGA firmware and the application code are updated most frequently. As the versions of the two must match, the tool updates both of them in simultaneously. The boot code and the low-level environment settings are not very likely to change and normally need to be written only once after the production. Product data – serial number, hardware version, production date and site – are written to a special region in the FLASH device which is one-time-programmable and must be initialised after the factory test.

The main ROS application “ROSApplication” uses the library to set the configuration parameters according to the values in the global configuration database, but limited to non-expert parameters. The channels of the ROBIN are enabled or disabled according to the global run-control state of the TDAQ system. The online monitoring systems requests the operational monitoring data on regular intervals. Fragments are requested by a multi-threaded request handler, which queues a number of requests per

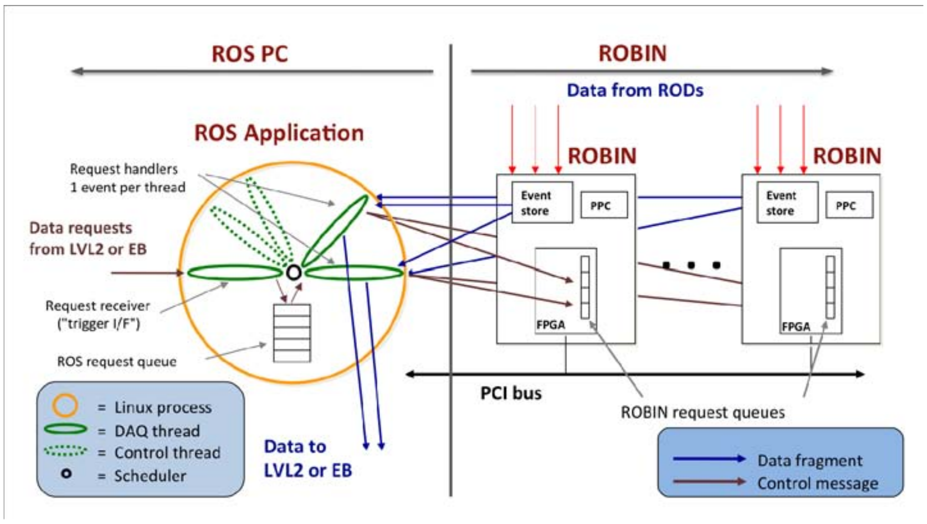


Figure 23: ROS-ROBIN interaction

channel to the ROBIN. Delete requests can be interleaved into the queue. The interaction between the ROS application and the ROBIN is depicted in Figure 23.

3.3.4 Switch-based ROS

As explained in chapter 3.3.3 the performance of a standard 12-channel ROS-PC is just above the standard requirements. If the search for new physics requires to run the TDAQ system at higher rates more ROS-PC with fewer ROBINs are required, which consumes eventually much more rack space. An alternative approach to achieve higher performance with almost the same system density is the switch-based ROS. Here, the ROBINs are connected to the dataflow network via their private GE interfaces. Due to the reduced load on the ROS-PC a fifth ROBIN can be installed. The 5 additional network ports per ROS-PC can be connected to an additional switch per rack, which takes the place of one of the ROS-PCs. As the total number of channels per rack does not necessarily change, this implementation has relatively little impact on the overall installation.

A more extreme variant of the switch-based ROS has been investigated based upon the idea of a ROB-on-ROD: the ROB replaces the S-Link source card on every ROD and connects directly to the dataflow network [ROBROD]. This implementation has a very high flexibility and performance but requires a large number of network ports and significantly complicates commissioning. Although a procedure to solve the commissioning issues was proposed [ROBRODC] this solution was mainly dropped due the problems expected in that area, due to the ROBs belonging physically to the ROD system but logically to the TDAQ system. Nevertheless, the requirement remained on the ROBIN to be able to prototype such an architecture.

The current view of the switch-based ROS is just a variation of the standard bus-based architecture and is applied only to sub-detectors which require performance not achievable otherwise. The most likely scenario is that the ROS-PC will remain responsible for configuration, monitoring and distribution of delete messages to the ROBINs. Possibly, it will also collect fragments from the ROBINs for the EB requests. The ROBINs will individually respond to L2 requests via their network interfaces in a way, that every ROBIN effectively implements a ROS subsystem containing 3 channels. With 5 ROBINs per ROS-PC the available network bandwidth on the L2 network is about 5 times higher than of a standard ROS-PC, which roughly matches the rate-performance ratio of a ROBIN and a standard ROS-PC.

3.4 Summary

The main parameters of the ATLAS detector are very similar to CMS. A custom L1 stage reduces the initial event rate (GHz range) to 100kHz. The data corresponding to a full event are generated by 7 sub-detectors, distributed over 1600 sources and pushed into the DAQ/HLT subsystem via optical S-Links. The nominal output to mass storage operates at 100Hz with an event size in the order of 1MB. The entire ATLAS DAQ/HLT system is build from commodity technology – standard PCs and GE networks – with the exception of the ROBIN components which establish the interface to the detectors plus a unit which controls the event assignment – the region-of-interest-builder (RoIB).

The mechanism of the dataflow internal to DAQ/HLT is very different from the traditional “PUSH” model. Initially, all event fragments are received by the read-out-system (ROS) which provides buffering for a few hundred ms. The ROS is composed of standard PCs each housing typically 4 ROBINs. About 2% of the stored fragments are pulled from the ROBINs by the L2 trigger subsystem, which performs a quick analysis based upon regions-of-interest and sequential selection. For events passing L2 all fragments are pulled by the event builder (EB) subsystem, for full online analysis. Due to the “PULL” model, all events have to be explicitly deleted from the ROBINs, which is done via broadcasts generated by a central dataflow-manager (DFM).

For the ROS two architecture variants are considered. In the baseline bus-based ROS only the PC interacts with the rest of the system, forwards the requests to the ROBINs and collects the returning data. The performance of the PC's processor and memory limits the maximum request rates in this case to around 10kHz. The enhanced switch-based ROS allows the ROBINs to interact directly to the rest of the system via its private GE port. In this case, the ROBIN becomes the performance limiting component, at a request rate around 20kHz.

The size of the system with respect to network connectivity is in the order of 300 (baseline) to 1.000 (enhanced) ports for the ROS, 700 ports for L2 and 300 ports for EB. The final implementation uses two large central switches plus a number of concentrator switches which group some ROSes and L2 processors respectively. The network components are arranged such that the load under nominal conditions stays below 60% capacity of the switches, in order to avoid message loss. Simulations and measurements on large-scale test setups have shown that the required performance can be obtained with GE.

The ROBIN component is exposed to the high input rate of 100kHz on each of the 3 input channels and has to perform bookkeeping of up to 64k stored events per channel. On the output interface, it has to deal with requests from PCI and GE, which can be active concurrently with a combined nominal rate of 6kHz and a maximum rate of 21kHz per channel. In addition, it has to run complex operational monitoring tasks. The final ROBIN implementation uses the combination of microprocessor and reconfigurable logic (FPGA) technologies, providing a cost efficient design tailored to the specific requirements of ATLAS.

The ATLAS dataflow system does not provide scalability via an inherent granularity as CMS does. The entire ROS and the central switches must be present in any case, which is a significant constant offset in terms of resources. On the other hand, HLT performance can be increased virtually in terms of single machines.

4 FPGA Technology

FPGA technology addresses application areas which require more performance than software solutions can provide on one hand but which cannot use custom hardware design with digital logic components and ASICs on the other hand due to flexibility and cost issues. In principle, an FPGA is a silicon chip with a large number of simple, uniform logic elements (LE) and a large number of I/O pins. These elements are used to hard-wire the required functionality, while the re-programmability of the device allows to update or entirely change the behaviour of the circuitry under user control.

FPGAs are ideally suited to implement any simple logic functions, dataflow and memory controllers and certain processing algorithms, for example which use primarily simple parallel processing. These features match very well the requirements of this project. The following paragraphs introduce the basic elements of FPGA technology, the development tools and some prominent examples of library elements used in or at least considered for the project, with a focus on the XILINX Virtex-2 device family used on the ROBIN.

4.1 Device types

FPGA technology was introduced in 1984 by XILINX. Since then, a number of different FPGA types have been produced by XILINX and other vendors like ALTERA, ATMEL and LATTICE. To date, there are two main branches in FPGA technology: one-time-programmable (OTP) and volatile. The OTP branch directly addresses high-volume applications, where true ASICs cannot be used for whatever reason. Also, radiation hardness is quite good with OTP technology, which makes it the first choice for space-bound and similar application areas. The volatile branch uses on-chip static RAM to store configuration data. This requires chip initialisation after every power-on transition but provides an infinite number of reconfigurations. Due to the flexibility required for the ATLAS ROBIN, only the volatile technology is viable and OTP technology has never been considered.

4.2 Basic elements

The LEs that build the dominant portion of an FPGA are based upon a small memory and a storage cell. Typically, the memory has 4 address inputs and a single data bit with separate input and output. Each of the addresses is an input pin to the LE. The data output can be routed directly or via a storage cell to the output of the LE. In the VIRTEX-2 [XLNXDS31] FPGA family two LEs are grouped together to a slice. Additional functionality per slice enables to configure polarities, clock edges, carry-chains, dual-port memory usage etc. The arrangement of a single LE is shown inFigure 24.

The logical function of the LE is implemented by an appropriate initialisation of the memory, which is used as a look-up-table (LUT). Due to the low number of inputs, complex logic requires the cascading of LEs in order to generate functions depending of many inputs. Alternative to logic functionality the LE memory can be used as dedicated memory, for which depth (address lines) and width (data bits) expansion is possible as well. FPGAs can also provide a large number of I/O pins, called IOBs. To accommodate the use in many different environments I/O voltages and I/O standards are configurable with voltages in the range from 1V up to 3.3V.

The Virtex-2 family for example supports 25 different I/O standards including differential ones, plus programmable drive strength and on-chip termination. Standard I/O cells are capable to run double-data-rate (DDR) up to 1.25Gbit/s in Virtex-5, and specialized I/O cells up to 6.5Gbit/s. A Virtex-2 or Virtex-5 chip is divided into several I/O banks. All IOBs in a bank share the same I/O voltage and access to global resources like clocks and reset signals. I/O standards within a bank can be different however, as long as the voltage requirements are compatible. For example, LVTTL, LVCMOS, LVDS

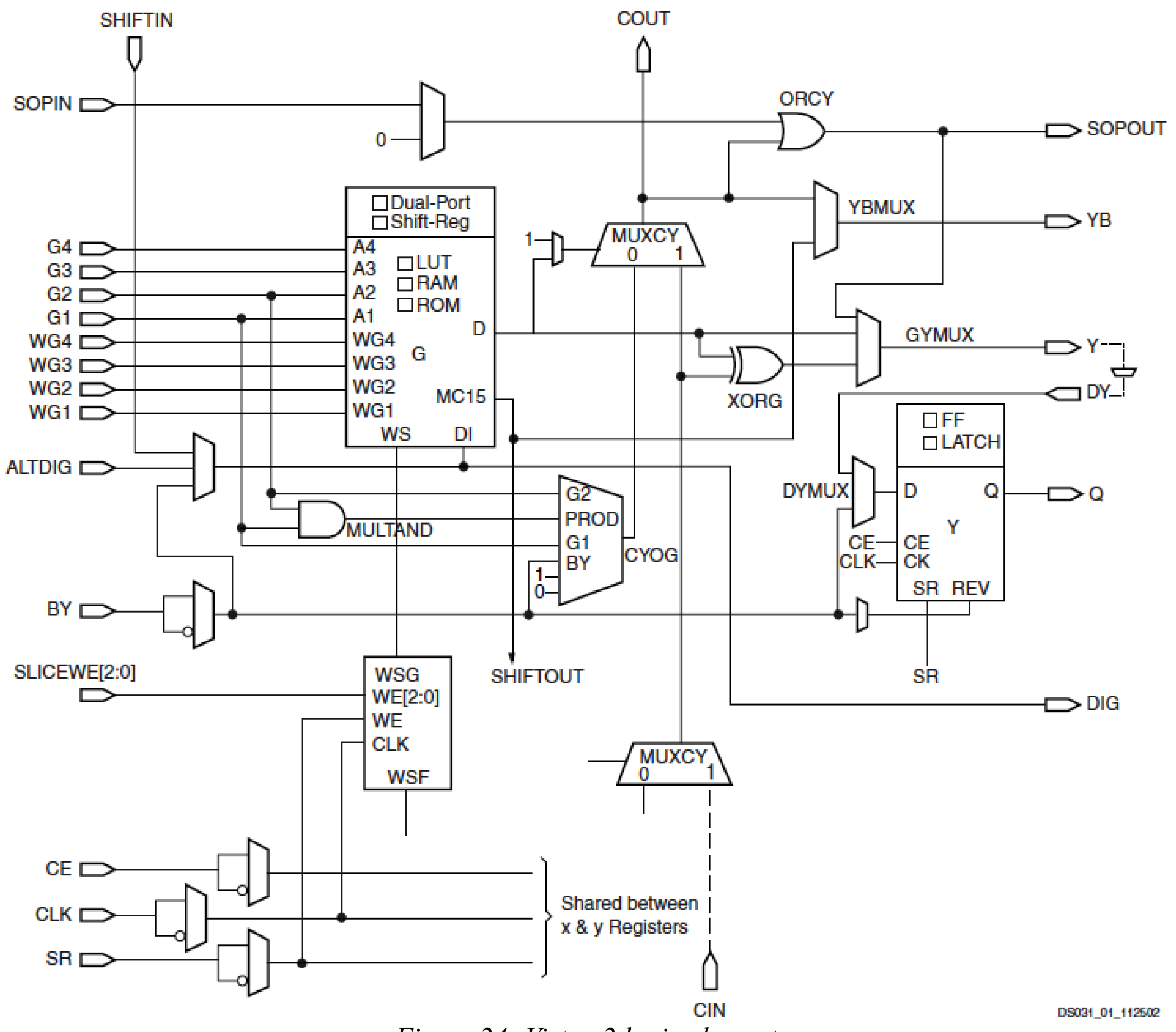


Figure 24: Virtex-2 logic element

and SSTL can be used in a single bank running from 2.5V. The package layout and I/O banking scheme of the FPGA on the ROBIN is shown in Figure 25.To connect the LEs and the IOBs a flexible routing fabric is required, which is realised via traces implemented in several layers[42] of metallisation and programmable transistors for the connections. All resources – slices, IOBs and special functions – are arranged in a matrix with row and column organisation. Every row and column has access to routing resources of different layers:

42 Virtex-2 has 10 metal layers, Virtex-5 has 12 layers.

- Long lines
- Hex lines
- Double lines
- Direct connect lines

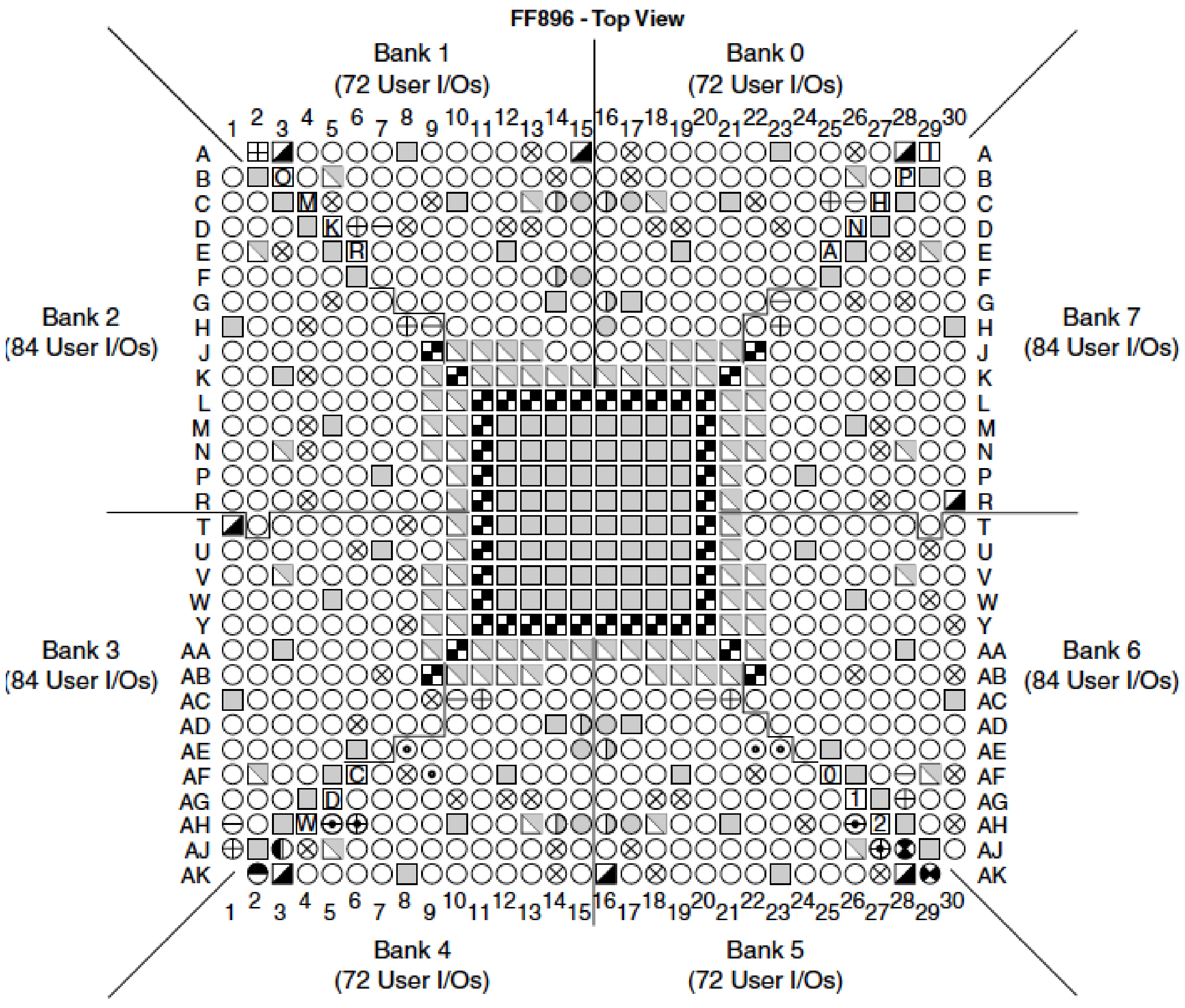


Figure 25: Virtex-2 FF896 package

For example, direct connect lines connect adjacent LEs and IOBs only while long lines provide connectivity across the entire device.

The evolution of FPGA technology over the last years is shown in Table 3. In the beginning, FPGAs were mainly used to implement simple logic functions (glue-logic). With increasing size full applications – for example image processing algorithms – could be implemented. The addition of special functions enabled to build complete systems on a chip (SoC).

Year	Family	Structure	LEs	System speed	Pins	Special functions
1995	XC4000	250nm	5000	80MHz	500	No
2003	Virtex-2	150nm	20000	150MHz	1500	Some (internal)
2008	Virtex-5	65nm	100000	300MHz	1700	Many (internal, I/O)

Table 3: XILINX FPGA families

For the Virtex-2 family, extra functional blocks were added to the silicon for memory, clock management and math functions. Moreover in Virtex-5, there are embedded processors, triple-speed Ethernet controllers and high-speed serial I/O blocks.

4.3 Tools

There are four categories of tools related to FPGA designs which cover the following areas:

- Specification of the functionality
- Simulation and test
- Synthesis
- Vendor specific tools

and which are all together used in a typical design process.

4.3.1 Specification

There are different methods to specify the functionality of an FPGA. In the early days and with simple functionality the logic functions of the LE were edited and the connections created manually or via scripts. Schematic editing tools were used frequently later, which enable to use libraries of standard TTL functions like multiplexers, counters, flip-flops et cetera to create a design description for an FPGA just like for a electronic board. However, this method also requires a thorough hardware expertise from the designer and for large devices the schematic drawings become unmanageably large. A further step was the application of a standard hardware description language (HDL) like VHDL or Verilog to FPGA design. The HDL approach enables to create a text based structural description using the same elements as used for the schematic description. But more important it allows to define functionality in a procedural way, which resembles the standard software development process, at least to a certain extent. In VHDL for example, the designer can define data types, variables, constants, functions and procedures and make use of control structures like if-else, case and while statements. A typical VHDL code snipped is shown below:

```
--------------------------------------------------------
-- Multiplexer for transmitting data to TLK
--------------------------------------------------------
tlk_tx_multiplexer: process (tlk_tx_clk, bypass_cntl_data(0))
begin
   If bypass_cntl_data(0) /= '1' Then
      tlk_txd       <= hola_txd;
      tlk_tx_en     <= hola_tx_en;
      tlk_tx_er     <= hola_tx_er;
   Elsif Rising_Edge(tlk_tx_clk) Then
      If txFifo_data_at_output = '1' Then
         tlk_txd    <= txFifo_dout(15 downto 0) after 1 ns;
         tlk_tx_en  <= txFifo_dout(16) after 1 ns;
         tlk_tx_er  <= txFifo_dout(17) after 1 ns;
      Else
         tlk_tx_en  <= '0' after 1 ns;
         tlk_tx_er  <= '0' after 1 ns;
      End If;
   End If;
end process tlk_tx_multiplexer;
```

The code starts with comments lines, then a control structure with the name *tlk_tx_multiplexer* of type *process* is defined. The parameter list to the process defines input signals to which the process is sensitive, here *tx_tlk_clk* and *bypass_cntl_data(0)*. There are other input signals as well, which are not on the sensitivity list. The difference is, that the transition of any output of the process occurs only when a signal from the sensitivity list changes. A transition of a normal input has no immediate effect on the outputs. The variables used in this code snipped are signals which can normally have one of two values – '0' or '1' – if seen as electrical signals. However in a typical VHDL description the signals are logical signals which can have one of multiple values: apart from '0' and '1' there are 'Z' to indicate a high-impedance state, 'X' to indicate unassigned, 'U' to indicate undefined and 'H' and 'L' to indicate weak pull-up and pull-down states respectively. The main purpose of this multi-value-logic (MVL) is to improve simulation of the circuitry. Signals can be grouped to vectors and vectors can be references to as a whole or by single or multiple elements.

Nevertheless, creating a design specification with VHDL still requires the skills of hardware expert, as there are many constructs which are not common to software programmers. Other approaches have been developed to generate a hardware description from “C”-style code[43], enhanced with some features to specify parallelism, or to integrate FPGA libraries into graphical programming frameworks[44]. Such tools make it easier for software experts to create FPGA design specifications but still lack flexibility and performance compared to the HDL methodology.

4.3.2 Simulation and test

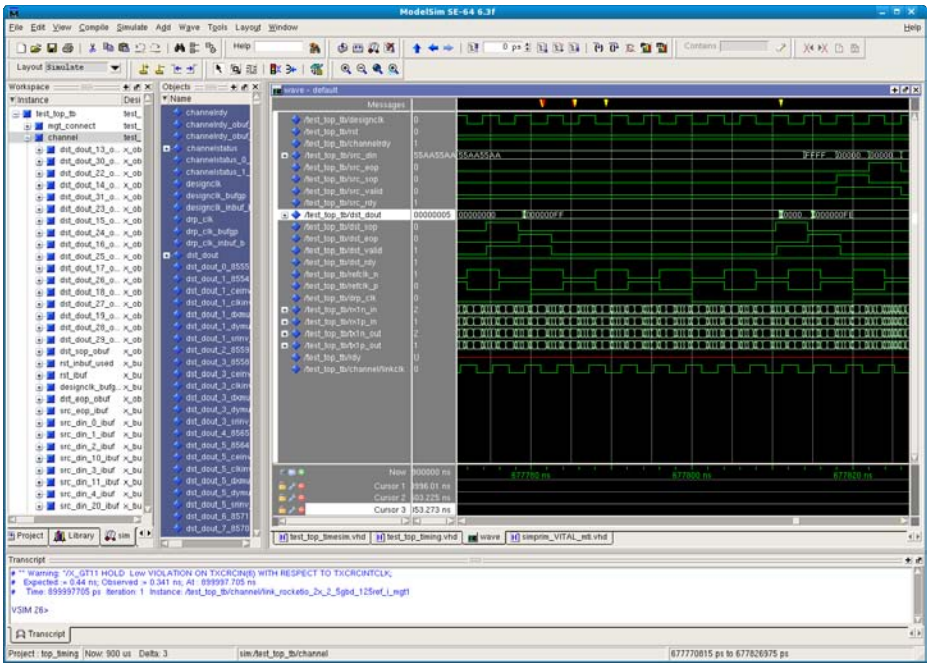
Once a designs specification has been created, the functionality has to be verified using a functional simulation. For a VHDL based specification, this is done using a VHDL simulator[45]. The simulator creates an executable representation of the design which enables to probe and stimulate any signal in the design. Typically, the external signals of the design are stimulated by a series of so-called test-

43 A tool to create FPGA code from a C-style description is “Impulse-C”: www.impulsec.com
44 A common graphical programming framework is “Matlab” (www.matlab.com) which is specialised for DSP style FPGA applications.
45 A common VHDL/Verilog simulator is “Modelsim” from Mentor (www.mentor.com).

vectors, which are applied one after the other. For every test-vector the executable is run until a steady state is reached, then the outputs are updated. Simple test-vector sets can be created manually. Complex test-vector sets, for example the access sequence of a microprocessor over a bus, can be generated via scripts or via external programs. The results of a simulation run (the application of a set of test-vectors to the simulation executable) are normally viewed as a waveform, as shown in Figure 26.

Simulation has a number of advantages. First, the simulation can monitor internal signals which are not visible on the physical boundary of the device. Next, creating the functional simulation executable can be done much quicker than creating the configuration data-set. Finally, simulation can be done without having the target hardware available. However, there are disadvantage as well:

- The precise timing behaviour of the design is not properly considered, which can lead to a mismatch between the results of the functional simulation and the real design.
- Simulation is normally much slower than real operation, for example simulating 1ms of real time can require 1min of simulation time or more. To achieve acceptable simulation times the source designs have frequently to be modified[46] for simulation purposes.



```
# ** Warning: */X_GT11 HOLD  Low VIOLATION ON TXCRCIN(6) WITH RESPECT TO TXCRCINTCLK;
#    Expected := 0.44 ns; Observed := 0.341 ns; At : 899997.705 ns
#    Time: 899997705 ps  Iteration: 1  Instance: /test_top_tb/channel/link_rocketio_2x_2_5gbd_125ref_i_mgt1

VSIM 26>
```

Project : top_timing | Now: 900 us Delta: 3 | sim:/test_top_tb/channel | 677770815 ps to 677826975 ps

Figure 26: Simulation waveform

To compensate the potential timing mismatch modern VHDL simulators are able to use vendor supplied simulation models, which provide information of inherent signals delays related to the basic FPGA elements. Signal delays introduced by signal routing however are basically ignored. A precise timing model – including all routing delays – of a design can be obtained after the entire design compilation has been completed. However, the execution time of a simulation using a timing accurate

46 For example, an I/O controller requires stable input signals for initialisation during 1ms of real time. For simulation purposes this timing requirement could be shortened to 10µs.

model will normally be higher by a one or two orders of magnitude, which limits the usefulness to very specific situations.

Testing of the FPGA functionality in real life is on the first view limited to the observation of the I/O signals via an oscilloscope or a logic analyser. For complex FPGA designs this is totally insufficient. Tools to access internal signals in a way similar to accessing external signals are available from FPGA and synthesis tool vendors and build upon the feature of the FPGAs that the internal resources can be monitored and controlled via a JTAG interface. The typical approach followed by a tool like ChipScope[47] is to use some internal FPGA memory to store samples from a selection of interesting signals. The insertion of memory, controllers and of additional signal connections is done during the synthesis step. Trigger conditions can be defined and uploaded via JTAG at run time. The results are read via JTAG and displayed in a waveform view (see Figure 27), similar to the simulation view.

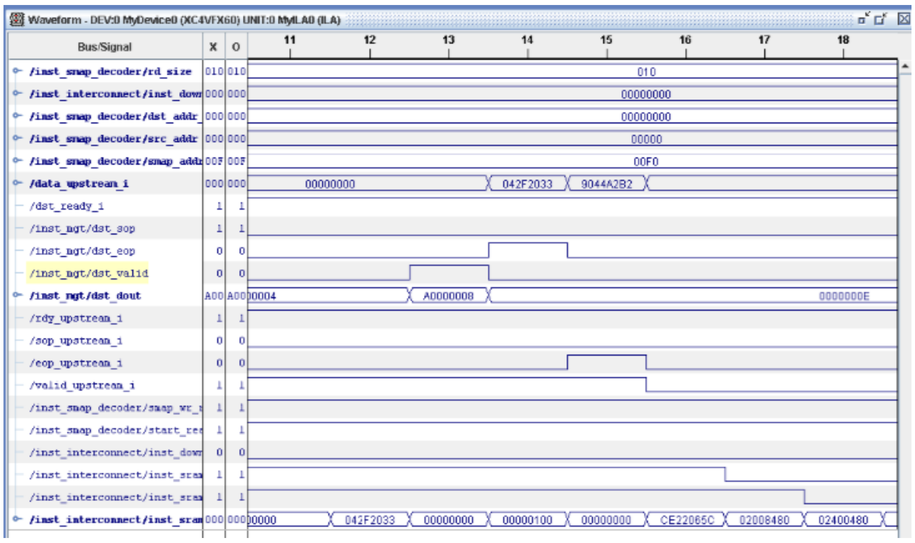

Waveform - DEV:0 MyDevice0 (XC4VFX60) UNIT:0 MyILA0 (ILA)
Bus/Signal
X
O
11
12
13
14
15
16
17
18
/inst_snap_decoder/rd_size
/inst_interconnect/inst_down
/inst_snap_decoder/dst_addr
/inst_snap_decoder/src_addr
/inst_snap_decoder/snap_addr
/data_upstream_i
/dst_ready_i
/inst_ngt/dst_sop
/inst_ngt/dst_eop
/inst_ngt/dst_valid
/inst_ngt/dst_dout
/rdy_upstream_i
/sop_upstream_i
/eop_upstream_i
/valid_upstream_i
/inst_snap_decoder/snap_wr
/inst_snap_decoder/start_req
/inst_interconnect/inst_down
/inst_interconnect/inst_sram
/inst_interconnect/inst_sram
/inst_interconnect/inst_sram
010
00000000
00000000
00000
00F0
00000000
042F2033
9044A2B2
00004
A0000008
0000000E
042F2033
00000000
00000100
00000000
CE22065C
02008480
02400480

Figure 27: ChipScope waveform view

This test mechanism enables a deep insight into the internal functionality during run-time. Compared to the simulation the number of samples is very limited (a few thousands) and every modification of the signal set for sampling or triggering requires to re-run the time-consuming physical compilation.

4.3.3 Synthesis

The synthesis step starts at the source of the specification, just like the simulation. For a structural description based upon libraries made from basic elements this is a straightforward process and done simply by expansion of the libraries to the network of basic elements. However an abstract HDL description requires a complex tool to analyse the description and to create the proper low-level description suitable for the FPGA. The statements of the HDL source must be translated into a network of basic FPGA elements, for example into flip-flop registers with asynchronous reset like in the code snippet above, or into embedded memory structures. This process is very specific for

47 ChipScope is the test tool from XILINX.

different FPGA families and vendors. Advanced synthesis tools[48] utilise timing information already at this step and optimise the generated logic according to various user constraints, e.g. they minimise delay or minimise space. The result is then a fully mapped structural description, where the basic elements are already grouped together according to the physical resource constraints[49] of the target FPGA. The performance of state-machines, arithmetic statements, counters and complex control structures depends strongly on the quality of this translation and sometimes a performance goal can only be met when the design is synthesised with a special tool.

4.3.4 Vendor specific tools

The network of basic elements created by the synthesis must be distributed across the resources available on the chip and the proper connectivity must be established. This process is called place-and-route (P&R). If the mapping has not already been done the synthesis tool, the logic functions and registers must be grouped to match the multiplicity of the physical FPGA structure. Once the P&R is complete, a timing analysis has to be performed to verify that the timing constraints are met. Timing constraints are defined for example by specifying the clock period for every clock network in the design and by “input setup to clock” and “clock to output” parameters. If the timing goals are met, the implementation behaviour should be consistent with the simulated behaviour. If the goals are not met, different settings of the P&R tool may improve the results, or the source design has to be modified. In many cases, the timing analysis provides a good indication which signal paths have to be improved. A typical modification is to add one or more register stages (pipeline registers) to a construct which requires many logic levels[50]. The final step after P&R is the generation of the configuration data-set – the bit-stream – which has to be downloaded into the device. During the bit-stream generation also a final design-rule-check is performed. A major task of the bit-stream generation program is to perform the mapping of configurable elements to bits in a way that prevents reverse engineering attempts. In newer devices there is also the option to encrypt the bit-stream. The decryption is done on-chip with a key stored in a battery-buffered memory.

4.4 IP-Cores

All FPGA vendors provide libraries of special functions which help implementing complex designs. Some of these library elements are available as HDL source code, but most of them are available in the form of encrypted netlists, which can be added via a wrapper-construct to the HDL or schematic specification. The netlist is then included by the vendor tool-chain after the synthesis step. There are also third-party companies which develop and sell such modules as intellectual property (IP) cores. Examples for such IP cores are processors, communication controllers like PCI or Ethernet, encryption/decryption modules, etc. Recent FPGAs also provide a number of embedded[51] cores like block-memories, multipliers, processors, Ethernet MACs and serial Gigabit transceivers. The

48 Advanced synthesis tools are for example “Synplify Pro” from Synplicity/Cadence (www.synplicity.com) or “Precision” from Mentor (www.mentor.com).

49 This grouping takes into account how functions and registers can be allocated to a slice of LEs

50 A logic level corresponds to a single LE. Multiple logic levels require cascading of LEs, which increases the delay.

51 Embedded cores are hardware modules on-chip, which do not consume FPGA resources.

synthesisable cores are commonly called soft-IP, the embedded cores hard-IP. The ROBIN makes use of both soft and hard IP. Hard IP is used in the form of memories blocks which implement dual-port memory and FIFOs. The Gigabit-Ethernet MAC is an example for a soft-IP core. The external processor of the ROBIN was selected to be of the same kind as the embedded processor of more recent XILINX FPGAs, in order to facilitate a potential migration to a denser implementation.

4.4.1 Embedded processors

Common embedded processors (hard-IP) are based on either ARM (ALTERA) or PowerPC (XILINX) technology. XILINX Virtex-2Pro and Virtex-4FX FPGAs include one or two PowerPC-405 cores which can operate at up to 400 or 450MHz respectively. XILINX Virtex-5FX FPGAs include one or two PowerPC-440 cores which can operate at up to 550MHz. Using the embedded processors one can implement a complete system-on-chip (SoC), equivalent to a micro-controller with customised peripherals. The interface between PowerPC core and FPGA fabric runs via the standard PowerPC processor local bus (PLB), which translates between the PowerPC core frequency and the clock domain of the FPGA logic. The embedded processor cores can be interfaced to internal block-memories quite easily but adding external memory consumes a significant amount of FPGA resources, at least for the Virtex-2Pro and Virtex-4FX FPGAs. Communication between the processor core and FPGA logic can be done via different interfaces, of particular interest are the “fast simplex link” (FSL) ports and the “auxiliary processor unit” (APU) interface, which are both supported by special processor commands. Both interfaces should provide very efficient communication mechanisms.

The XILINX tool suite “embedded development kit” (EDK) allows to develop processor based FPGA designs, where the processor(s) and a number of IP-cores can be combined to form an embedded system. Custom user IP-cores can be developed following an IP-reference design and added to the system. Programming is done using the open-source GNU[52] tool chain. The same tool-suite can be used for systems based on the XILINX soft-IP processor “MicroBlaze”, which has a 32bit RISC architecture similar to the PowerPC operating at 60 to 125 MHz, depending on the FPGA platform.

4.4.2 Ethernet MAC

Ethernet is a very common communication standard and is also widely used in embedded systems. Typically, an Ethernet interface is built from two devices, a physical interface adapter (PHY) and a media access controller (MAC). The PHY implements layer 1 of the OSI layer model [OSI] and attaches to the physical media, which can be optical or electrical. Common speed-grades are 10, 100 and 1000Mbit/s, 10Gbit/s is about to move into the commodity market, higher speeds are under development. The electrical media normally uses pulse-transformers to provide electrical isolation. The electrical media uses 1, 2 or 4 signal pairs in an RJ45 connector[53]. The interface between PHY and MAC follows one of the “media independent interface” standards – MII, RMII, GMII, RGMII or SGMII – depending on the speed and the device type[54]. There is an additional control interface to

52 GNU software is available from www.gnu.org.

53 The RJ45 connector allows full-duplex operation, if the link partners are peer-to-peer or connected via switches. There are other (older) technologies as well, which are not relevant for this work.

54 MII is for 10 and 100Mbit/s, the “G” types are for 1Gbit/s, “R” indicates dual-edge signalling, the “S” indicates serial interface.

access the registers in the PHY, implemented as a bi-directional serial bus.

The purpose of the PHY device is to translate between media signal levels and standard logic levels and to perform the negotiation of the capabilities[55] between the link partners. The MAC device implements layer 2 of the OSI model and runs the Ethernet protocol [ETH802.3]. Ethernet uses packets with a fixed structure:

- A 22 byte header[56]
- Payload, maximum[57] 1492 bytes
- Padding up to the minimum packet size of 64 bytes, if required
- A 4 byte frame-check-sequence (FCS)

On the ROBIN a 1Gbit/s MAC soft-IP (GMAC) is used to implement the MAC functionality and connects to the external PHY via the GMII interface, which has 8 data bits plus 2 control bits per direction, running at 125MHz. On the user side the GMAC implements two independent FIFO-buffered ports, one for transmission and one for reception. The ports are 32 bit wide, plus control bits to indicate start and end of packet. In addition the GMAC collects the standard network monitoring values like number of transmitted/received packets and TX/RX errors, which can be retrieved from a set of registers to facilitate debugging in case of communication errors. A drawback of this soft-IP is that is does not support multiple speeds, hence the ROBIN cannot communicate with link partners of lower speeds. A possible work-around would be to additionally implement a MAC which supports 10/100Mbit/s communication and to switch between the MAC cores according to the negotiation result of the PHY. However, this would require an prohibitively large amount of FPGA resources and is not necessary in the normal working environment of the ROBIN.

New FPGAs like Virtex-4 and Virtex-5 include multiple triple-speed (10/100/1000Mbit/s) hard-IP Ethernet MACs, which consume very little additional FPGA resources.

4.5 Summary

XILINX Virtex-2 FPGA technology as used on the ROBIN allows to run at system speeds of up to 150MHz, several hundred pins are available for user I/O plus several complex functional block like configurable memory arrays and DSP blocks. Functionality is realised via a bit pattern (so-called bitstream) which controls the operation of a large number of simple processing elements and I/O cells. The bitstream is stored in volatile on-chip memory, which allows an infinite number of configuration cycles. The configuration itself is created by a set of tools in a multi-step process which starts with the synthesis of a higher-level description – frequently called a design – into logic expressions – this is the RTL description. Pre-compiled library elements (so called IP-cores) can be used at this stage as well, which implement complex functions, for example a GE-MAC, a processor or a filter operator for image processing. Next, the equations have to mapped onto the basic elements of the device. Then the basics elements must be placed on the chip and the interconnects must be established. Finally, the

55 Link partners with different speed negotiate for the fastest common speed.
56 According to IEEE 802.3
57 The standard allows to go beyond the maximum packets size using “jumbo” frames of up to 16kB.

bitstream is generated which contains the configuration value for every single programmable element of the FPGA. The bitstream must be loaded into the device after power-up, typically from an external non-volatile memory, via a JTAG interface or from a microprocessor. A typical functional module[58] of an FPGA designs consists of many LE and requires to run at a specific operation frequency. The most critical task of the tools is to arrange the LEs on the chip in a way, that all interconnects are fully routable and the signal delays introduced by the routing are within the timing specification. This requirement is called timing-closure.

58 For example a counter or a basic computation of an image processing task.

5 ROBIN

This chapter presents the implementation of the ROBIN. It starts with a summary of the requirements, both in terms of performance and functionality, followed by a description of how the ROBIN project was managed from the early development stage through prototyping and volume production up to full operation. The basic hardware, the FPGA code and the processor application code are presented in detail. The chapter concludes with a description of the installation and commissioning procedures.

5.1 Requirements

The functionality of the ROBIN comprises different task areas. The most important group covers the very basic aspects of data handling – input and output – and the related bookkeeping. The key parameters (see section 3.3) are 100kHz event rate, 160MB/s input bandwidth, request rate of 21kHz, buffer latency in the order of 100ms and a multiplicity of 3 channels. The total nominal output bandwidth is relatively low, in the order of 100MB/s. To support both bus-based and switch-based ROS architectures a PCI interface and a private GE interface are required. The bookkeeping mechanism keeps track of the position of individual event fragments in the buffer memory. This information is needed to retrieve the data and to de-allocate the memory space once the event is processed.

Next, message handling functionality of various complexity is required in order to make use of the data handling. For the bus-based ROS, sending requests and receiving responses is straightforward, as the ROBIN is a standard PCI peripheral to the ROS-PC, which means peer-to-peer communication over a reliable media. In switch-based mode the ROBIN has to maintain network connections to several hundreds of nodes in parallel, running an unreliable protocol (UDP) over GE. Initially, the interfaces PCI and GE were meant to be operational mutually exclusive. However the planned approach to maximise the request rates is to use both concurrently.

Furthermore, ATLAS requires operational monitoring capabilities, like statistics of fragments, messages, buffer occupancies and errors. Initially, the main issues here were covered by simple counting of the associated occurrence (incoming fragments, for example). During the installation and commissioning phase of the ATLAS DAQ system however the monitoring was expanded significantly, for example by classification of transmission errors, a recording facility for corrupted fragments which do not follow the standard event format and an interrupt capability towards the host PC.

Finally, a set of functions must be available related to setup and configuration. The run-time behaviour is controlled via a number of configuration parameters. A small set of parameters controls the basic operation by setting the input mode and providing a tag which globally identifies every channel. Some parameters are related to the buffer management, like size and number of memory pages and input truncation threshold. A couple of parameters influence very particular behaviour and are used only during debugging. The static setup – FPGA bitstream, boot loader, application binary and default configuration parameters – is contained in a FLASH memory. The FLASH memory can be modified from the host PC via the PCI bus and by the ROBIN itself.

Besides the functionality listed a above, an important requirement on the ROBIN is flexibility. From a system perspective, the majority of the tasks are shared between the ROS-PC and the ROBIN in the bus-based architecture. Even for the basic dataflow the PC participates, by interfacing to the network and by combining the fragments from the individual channels into larger packets. In the switch-based architecture the ROBIN takes over the entire dataflow and even assembles the fragments from its channels prior to sending them off to the network, but configuration and monitoring are still handled in cooperation with the ROS-PC. In the most extreme scenario however – and the ROBIN was designed to prototype even that – the ROBIN operates fully in stand-alone[59] mode and still needs to provide the complete functionality. All configuration and setup – including firmware upgrade – has then to be handled by the ROBIN itself, controlled by a remote note over the GE interface.

The following sections describe the development and production process as well as the implementation of hardware and software of the ROBIN.

5.2 Project Management

As mentioned earlier there were already several prototype implementations existing prior to the current ROBIN design and the prototype results provided guidelines [ROBSUM] for the final development. These early prototypes were built by different groups[60] of the ATLAS TDAQ collaboration which used different form factors like PCI or PMC, different FPGAs, local processors – if any – and different buffer memory technologies like SRAM, ZBT and DRAM. The different approaches are explained in more detail in section 5.3 . After an initial analysis phase a small design team was formed, consisting of members from three institutes[61], with the mandate to propose a multi-channel ROBIN solution. The initial requirements were updated with respect to performance, functionality, space and cost. Concerning the hardware development it was decided to first build a new prototype[62] supporting two input channels and able to demonstrate all possible options with respect to the system architecture, even two different GE media interfaces – an optical and an electrical one. The final ROBIN should then be derived from the prototype-ROBIN, by leaving out unused functionality and eventually adjusting channel multiplicity and mechanical format. The design team – led by the author of this thesis – produced the prototype design proposal in the form of three design documents [HLDDPROT][DLDDPROT][SWIDPROT] which were reviewed and approved by a CERN expert group [FDRPROT]. Subsequently the prototype ROBIN schematics were created at Mannheim, followed by the specification of design rules for the PCB layout, which was done by an external company. Production and assembly of the initial cards was organised by the Mannheim group again, meanwhile the initial FPGA firmware and test software was produced at the three institutes and integrated afterwards. The location of the main developers at different sites and even different countries was a serious complication during debugging phases, in particular after the production of new initial cards. Video-conferencing tools were not satisfactory for detailed technical discussions,

59 A potential implementation would be to install a (eventually larger) number of ROBINs in a housing with a passive PCI backplane, which just provides the power to the boards. No host CPU would be required. Alternatively, one of the ROBINs could be configured as a PCI master and act as the host for configuration and monitoring.

60 Prototypes were build by CERN, Saclay/F, NIKHEF/NL, RHUL/UK and Mannheim/D.

61 Apart from the author there was one person each from NIKHEF/NL and RHUL/UK.

62 This was the so called Prototype-ROBIN.

hence personal meetings approximately every 6 weeks were held during hardware development phases in addition to the regular weekly telephone meetings.

In total[63], there were up to three persons working on the hardware design, five on the FPGA code and five on software in the different areas – boot and application code on the ROBIN, driver, library and test software on the ROS-PC, not counting the application software on the ROS-PC which was developed by the CERN ROS team. The prototype ROBIN modules were successfully tested and performance measurements provided good results. As a consequence, the design of the final ROBIN was – as expected – based on the prototype ROBIN and presented to the ROS community in another design document [ROBRDD], followed by another review at CERN [ROBFDR]. The preparation of schematics, layout and initial cards was executed in the same way – which basically means by the author – as for the prototype ROBIN. After the successful testing of the initial cards a mini-series of 10 cards was produced by the German production company who had done the initial cards as well. This series was tested not only for functionality and performance but also for reliability, using a climate chamber at NIKHEF and a programmable power-supply at Mannheim. After these tests the final stage of the CERN approval mechanism – the production readiness review [ROBPRR] – was prepared with documents on the final design [PRRDD] and performance [PRRPM], the test strategy [PRRTP] during production and for the regular self-test and a schedule for the volume production [PRRPS]. The volume production of 650 ROBIN cards, including some spare cards as specified in the policy for spares [ROBSPARE], was shared between two production sites[64]: the German company and a company in the UK, who used the German production documents to manufacture the printed circuit boards, to order the components and to assemble the cards. The supervision of the companies was done locally, in Germany by the author and in the UK by members of the UK group. All cards from the volume production were tested at the production sites by the local groups using a customized test-suite on top of the built-in self-test functions, which included the external interfaces PCI, GE and optical input, the latter via a loopback fibre. Sample cards were exposed to the same procedure for an extended period (typically over night) at the supervising institutes. Installation and commissioning of the ROBINs was performed primarily by the CERN team, with help from members of the design team. Finally, maintenance is organised in a way that non-experts at CERN are able to remove defective or suspective ROBINs from the system, after collecting as much information as possible using an analysis tool running on the ROS-PC. Those cards are then subsequently checked and repaired (if possible) by the hardware designers.

5.3 Implementation

The ATLAS ROS system (chapter 3.3) allows for different implementation variants of its elements which today are the ROC-PC and the ROBIN. The two existing architectures - bus-based ROS, which is the base line and comes historically first, and switch based ROS – differ in the way to concentrate the data. In the bus-based ROS this is done by a host computer, in the switch-based ROS by a network switch. Concerning the I/O sub-unit several options have been considered. For a bus-based

63 Most of the developers had particular skills – hardware, FPGA or software – bus a few worked in both or all areas.

64 Apart from the benefit of having a second source for the cards, the second reason for the sharing was related to financial arguments.

architecture the simplest solution would be to push all incoming data into the main memory of the host PC (the ALICE way, section 2.1.2) and to leave it to the main CPU to handle the events. The CERN prototype devices SSPCI[65] and S32P64[66] were used in such a setup as so-called software-ROBINs [SWROB]. The disadvantage of this approach is the very high load on the memory bus of the host PC and the high rate of events to be handled by the main processor. As a result, the link density per PC would be low. Also, this is not an option to implement the switch-based architecture. Solutions to store the data directly into the host memory while doing the bookkeeping locally on the I/O card were looked at, but not actually tested, because they solved only the memory bandwidth problem but did not support the switch-based approach either. In [MMROB] the first multi-channel prototype with local buffering and bookkeeping was described, using the general purpose PCI-based FPGA co-processor MPRACE-1.

A step towards the switch-based ROS was made with an S-Link module [GBELSC] using GE as the output media, which stimulated the investigation of a ROB-on-ROD mezzanine [ROBROD] and the associated architecture. This approach initially looks simple and cost-effective but comes with serious complications during commissioning and maintenance. The problem here is that the ROD belongs to the detector subsystem while the ROB belongs to the DAQ and mixing the two on a single module makes is very difficult to independently verify the functionality. This is a problem in particular for the DAQ, as every intervention on a ROB-on-ROD module would require to power off or on the hosting ROD unit, unless a remote power supply via power over Ethernet (PoE [POE]) is established – which would in turn require a more complex implementation due to multiple power supplies. Another switch-based option proposed to implement the ROBIN as a mezzanine for a VME [VME] card, with a single S-Link input and a FE output per mezzanine and approximately 6 mezzanines per VME card. The VME host would take over the configuration functionality.

An extensive evaluation of all the options led to the implementation of the present ROBIN, which has a very universal design and basically a superset of the functionality of all investigated choices.

5.3.1 Hardware

As the hardware of the final ROBIN and the prototype-ROBIN are very similar, this section will focus on the final ROBIN and make references to the prototype-ROBIN where appropriate.

The following basic elements are essential for the hardware implementation of the ROBIN:

- Multiple ROL/S-Link channels
- Buffer memory
- FPGA
- CPU
- PCI Interface
- Network Interface (GE)

65 http://hsi.web.cern.ch/HSI/S-LINK/devices/slink-pci/
66 http://hsi.web.cern.ch/HSI/S-LINK/devices/s32pci64/

The FPGA is the central unit and all other elements are grouped around it.

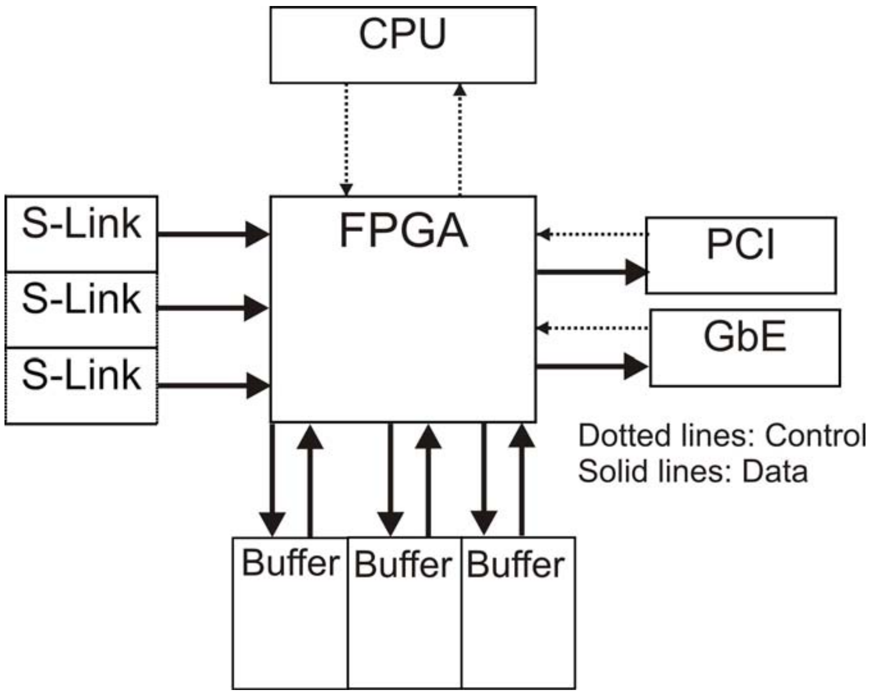


Figure 28: ROBIN basic hardware elements

The arrangement in Figure 28 satisfies the conclusions from the guidelines. The CPU is taken off the main data path, which is handled by the FPGA alone. The buffer management scheme (see 5.3.3.2) makes it straightforward to place the management memory at the processor, and in fact to make it part of the processor's main memory. The existence of both PCI and GE interfaces was initially required to investigate the two different ROS architectures – bus-based (see 3.3.3) and switch-based (see 3.3.4) – and they were assumed to be mutually exclusive. Later on, the GE interface was considered to be an additional path, operating concurrently with the PCI interface in certain conditions. The number of input channels was set to 3 for the final ROBIN and to 2 for the prototype-ROBIN. The MPRACE-1 board (see 8.1.1) was used as a template with respect to power-supply[67], PCI-interface[68] and FPGA family[69]. In addition MPRACE-1 was used to rapid-prototype some important functions prior to the implementation on the ROBIN, by building mezzanine modules for CPU and GE interfaces of both prototype ROBIN and final ROBIN.

Apart from the obvious requirements in terms of bandwidth and rate for the various interconnects of the data-path the communication required for the control-paths had to be analysed. There are 2 major interaction types to be mentioned:

- Buffer management related messages between FPGA and CPU. An information record of 16

67 MPRACE-1 uses a single 3.3V power source, which is supplied via the PCI connector or optionally via a ATX-style connector. All local voltages are generated from this main supply, mainly using switching regulators.

68 A PLX PCI9656 device is used. It bridges a 64bit/66MHz PCI2.2 bus to a 32bit/66MHz local bus interface.

69 XILINX Virtex-2

byte size is generated per page and deposited into a FIFO inside the FPGA, from where it has to be retrieved by the CPU. The nominal bandwidth required is 1.6 MB/s per channel.

- TDAQ request related messages. TDAQ requests typically involve multiple messages. First, a request message is being sent via the TDAQ-interface PCI or GE to the FPGA. Next, that message has to be retrieved by the CPU from the FPGA. Finally, a response message is sent from the CPU to the FPGA, which triggers the data transfer over the TDAQ-interface. A PCI data request requires 20 byte and a PCI delete-request 4 byte per event on average. On GE, data requests are approximately 4 times larger than on PCI, while the delete-requests have a similar size. At the nominal 20% request rate the bandwidth is approximately 0.8 MB/s per channel on PCI or 2 MB/s on GE respectively. The response message sizes – sent from CPU to FPGA – are in the order of 60 byte for PCI and 160 byte for GE, corresponding to bandwidths of 1.2 MB/s and 3 MB/s per channel.

Both CPU and PCI-interface provide 32bit/66MHz local buses which are used to connect to the FPGA. The GE-interface is connected via a standard GMII-interface[70]. Therefore, control messages consume only a small fraction of the maximum bandwidth of the various interfaces involved.

5.3.1.1 FPGA

The selection of the FPGA device family as XILINX Virtex-2 was an initial condition for the ROBIN. The main remaining parameters to select the particular device are logic resources, memory resources, clock resources and connectivity. Logic resources are needed to implement control functionality, for example to handle the HOLA-protocol or to steer data transfers. From the previous prototypes it was known that the requirements in this area are moderate. Memory resources are used in two different flavours: as DPR and as FIFO. FIFOs are typically used to transport short messages between the FPGA and the CPU, like fragment information or DMA descriptors. In case of longer messages, the FIFO contains message descriptors and the related data reside in a corresponding DPR. This approach is used for incoming PCI and GE messages.

70 GMII is 8 bit, 125 MHz. In contrast, the protoype-ROBIN had an external MAC connected to the FPGA via another 32 bit / 66 MHz local bus.

Memory item	Purpose	Size [Bit]	BRAMs[71]
HOLA Core FIFO	Buffer and clock decoupling	3*512*33	3
S-Link Handler FIFO	Flow control buffer	3*256*34	3
Test Input FIFO	Buffer test event fragments	3*512*34	3
LDC Bypass TX FIFO	Buffer TLK transmit data	3*2k*18	3
LDC Bypass RX FIFO	Buffer TLK receive data	3*2k*18	3
Buffer Input FIFO	Burst buffer dual-port emulation	3*512*33	3
Buffer Output FIFO	Burst buffer for dual-port emulation	3*1k*32	6
Free Page FIFO	Free page list for Buffer Manager	3* 1K*16	3
Used Page FIFO	Used page descriptors from Buffer Manager	2 * 512*128	12
Lbus Header FIFO	Header and DMA descriptor for outgoing message	512*32	1
Lbus DPR	Buffer request messages from Lbus	2k*32	4
Lbus Async FIFO	Buffer data to Lbus interface	32*32	0
Lbus Request FIFO	Request descriptor FIFO	32*32	0
MAC Header FIFO	Header and DMA descriptor for outgoing message	512*32	1
MAC Descriptor FIFO	Buffer descriptors of received messages	512*32	1
MAC Dual Port Memory	Buffer received messages (external memory)	2M*32	0
MAC Transmitter DPR	Decouples clock domains in MAC interface	2*2K*8	2
MAC Receive FIFO	Decouples clock domains in MAC interface	15*34	0
MAC status DPR	Ethernet statistic	2k*32	2
PPC Uart	Buffer data to host PC	2k*8	1
Sum:			51

Table 4: ROBIN FPGA memory utilisation

The various memory elements used in the ROBIN are shown in Table 4.

Peripheral components frequently send and/or receive data at their own frequency, which requires mechanisms to synchronise or decouple various clock domains in the FPGA. The Virtex-2 family provides DCMs (digitally controlled clock manager) to phase-synchronise internal clock signals with external clock signals and FIFO-elements with independent clocks to cross clock domains. A total of 8 different clock signals can be used globally[72] in the FPGA. The following clock domains are used in the ROBIN:

- ROL transmit clock (1): 100 MHz, used to send data to the TLK2501 devices.
- ROL receive clocks (3): 100 MHz, used to receive data from the TLK2501 devices.
- CPU clock (1): 66 MHz, used as the common “system” clock, e.g. as the main internal clock and on the local interconnects FPGA – CPU and FPGA – PCI bridge.
- Buffer and network transmit clock (1): 125 MHz, used to transmit data to the GE interface and

71 A value of 0 indicates implementation externally or in distributed memory.

72 There are 16 global clock available on chip, but not all of them can be used in all regions concurrently

for all buffer transactions.

- Network receive clock (1): 125 MHz, used to receive data from the GE interface.

The last global clock resource is used as a global reset signal. The use of DCMs to phase-align the external buffer clock or CPU clock to the corresponding internal clocks is possible but not actually required.

As already seen from Figure 28 there are numerous external components connected to the FPGA. On the ROBIN, all these components use a common electrical standard – 3.3V CMOS – which simplifies the signal distribution on the FPGA pins. The output signals towards the TLK2501 and the GE PHY use the XILINX DCI[73] feature to provide 50Ohm on-chip serial termination.

Interconnect	Pins per unit	Total number of Pins
CPU	98	98
ROL (embedded HOLA S-Link)	44	132
Local Bus to PCI bridge (multiplexed A/D)	52	52
GE PHY	27	27
Buffer Memory (32 bit wide)	57	171
MAC External DPR	55	55
Test Connector and S-Link Leds	50	50
Miscellaneous	12	12
DCI control pins	16	16
Sum:		613

Table 5: FPGA connectivity

The selected choice[74] for the FPGA is a XILINX Virtex-2 XC2V2000-5FF896. With 56 memory blocks and 624 I/O pins the requirements are met. A large fraction of the available pins are used by the ROBIN design, the remaining ones being routed to a dedicated test-connector, to facilitate hardware testing.

5.3.1.2 ROL/S-Link

The S-Link standard primarily specifies a protocol, plus electrical and mechanical properties, but not the physical transmission layer. In order to implement multiple S-Link channels on a standard PCI card a particular transmission layer had to be selected for an embedded implementation. Fortunately, a de-facto standard for the transmission layer – the HOLA[75] – was already established when the ROBIN development started. HOLA is based on optical transmission with bi-directional optical transceivers[76], at a rate of up to 2.5GBit/s. The SerDes (serialise/de-serialise) function is performed by a Texas-Instruments TLK2501[77] device, with separate 16 bit data paths for sending and receiving. IP-cores for FPGAs are available from CERN to handle the HOLA protocol for senders (LSC) and receivers

73 DCI means digitally controlled impedance, a features which allows to match the output impedance of selected IOBs to an external reference resistor, typically 50Ohm.
74 The prototype-ROBIN uses a XC2V1500-4FF896
75 http://hsi.web.cern.ch/HSI/s-link/devices/hola/
76 http://www.finisar.com/download_6ZZvZqFTLF8519P2xNLSpecRevJ.pdf
77 http://focus.ti.com/docs/prod/folders/print/tlk2501.html

(LDC), which attach to the TLK2501 via the I/O-pins of the FPGA and provide an S-Link protocol interface to the user application. One LDC-core consumes 1 BRAM and 500 slices in an XILINX Virtex-2 FPGA. 43 I/O pins are required, including control signals for the TLK2501 and the optical transceiver. The maximum bandwidth of 160 MB/s as specified by the S-Link standard does not require the maximum line rate of the optical link, therefore the rate was lowered to 2Gbit/s in order to improved the robustness. On the user side the nominal clock rate is 40MHz (160 MB/s divided by 4, for a 32 bit interface) but the HOLA cores can be operated at higher clock speeds, provided the FPGA timing constraints are met. On the ROBIN, the HOLA user clock is 66MHz, which means that the maximum user bandwidth is higher than the link bandwidth. Figure 29 shows the connections between the FPGA and three[78] S-Link channels. A 100 MHz clock oscillator provides the common transmit clock for all channels. On the receive path, every TLK2501 generates its private receive clock, therefore four clock domains have to be handled inside the FPGA. Control functionality comprises enable, loop-back and test mode for the TLK2501 plus enable and signal-detect of the optical transceiver. The serial signals between TLK2501 and SFPs are implemented as 100 Ohm differential pair transmission lines. The parallel signals between TLK2501 and FPGA are length-matched 50 Ohm traces. On the receiving side series termination resistors located at the TLK2501 are used. On the transmitting side the XILINX DCI-feature is used to adjust driver impedance.

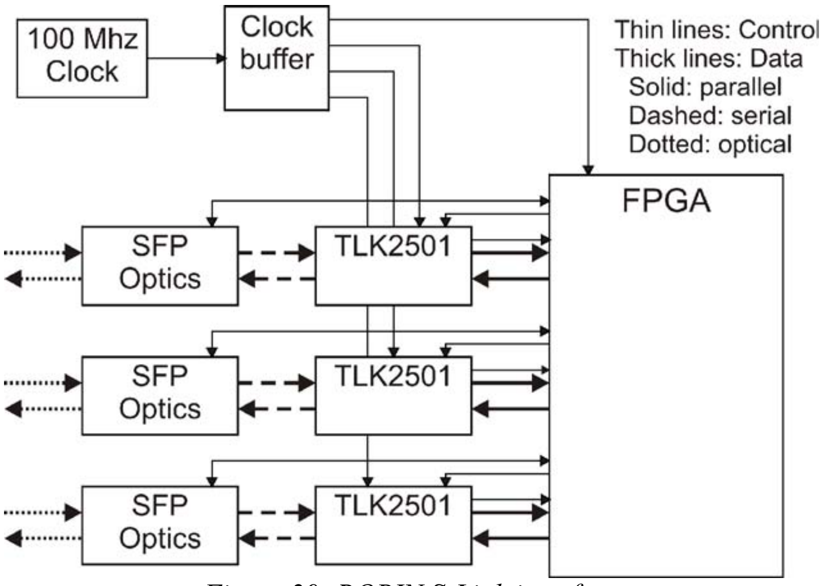


Figure 29: ROBIN S-Link interface

5.3.1.3 Buffer memory

The main criteria for the selection of the buffer memory are bandwidth and system latency. With respect to bandwidth the buffer must sustain 160 MB/s incoming data (the maximum S-Link bandwidth) plus 20% for outgoing data or roughly 200 MB/s. The worst case is defined by 100% request ratio, corresponding to 320MB/s. The relevant latency is determined by the processing time of the L2, which was initially assumed to be in the order of several 10ms (newer estimate is approximately 100ms, see Figure 18). From these figures the initial buffer parameters are defined: 200

78 The prototype-ROBIN has only 2 S-Link channels.

MB/s bandwidth, 1.6 MB size (for 10 ms latency). These values correspond very well to the values used on the older prototypes and numerous solutions[79] are viable. However in a switch-based system latency can be much larger, due to delays introduced by the network communication or by handling of eventual message loss. Hence a safety factor in the order of 10 was proposed at least for the prototype-ROBIN. A buffer size of 16 MB can sensibly only be implemented with dynamic memory, as the density of asynchronous SRAM devices is too low and the cost of synchronous SRAM is too high. A cheap and simple 2 chip[80] DRAM running at 125 MHz is used on the final ROBIN[81] to provide 64 MB of buffer space and 450 MB/s nominal bandwidth.

Figure 30 shows the effective bandwidths for 2 kinds of memory at two different operating frequencies. The SRAM has a fixed latency per burst of 2 cycles while the dynamic memory has a frequency dependent latency (due to the fixed timing parameters) and a penalty introduced by the periodic refresh. It is obvious that bandwidth drops rapidly when the burst-length is shorter than 64 words, which is equivalent to 256 bytes on the ROBIN. Due to the format of data coming from the ATLAS detectors there is no problem to keep burst-sizes sufficiently high, as the nominal conditions require fragments of 1.6kB for maximum bandwidth.

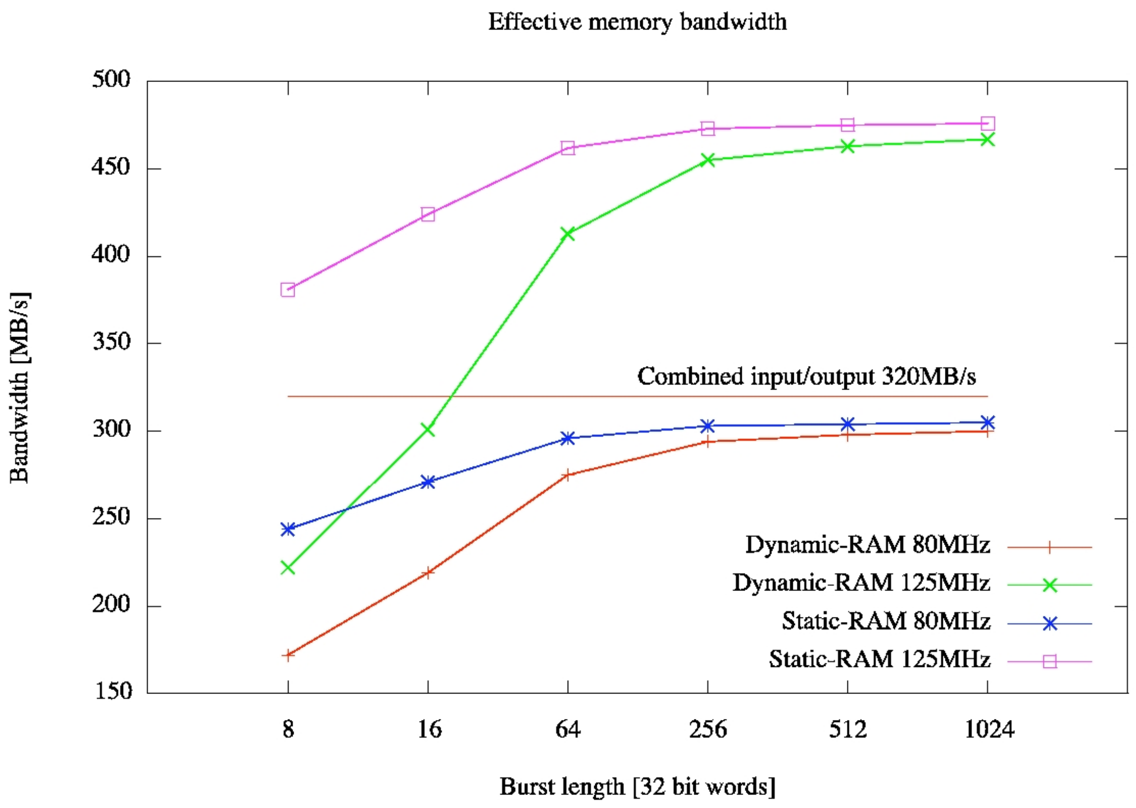


Figure 30: Burst-size dependant memory bandwidth

79 Virtually all available memory devices could be used to implement such a system, from simple asynchronous SRAM to high density DRAM

80 Devices are MICRON MT48LC16M16A2TG, 256MBit, 16M*16

81 The buffer memory on the prototype-ROBIN runs at 100 MHz only, due to the slower speed-grade of the FPGA.

The 80MHz used by some older prototypes are not sufficient to sustain concurrent input and output at nominal speed of the input link, which sums up to 320MB/s. In reality, the instantaneous bandwidth requirements can be higher, as the fragments are transferred at up to 200MB/s into the memory while a read access can take place at up to 264MB/s. Burst sizes of 64 words or larger can be easily handled by the FPGA, hence the 125MHz provide sufficient sustained bandwidth even for 100% readout at full input rate. The advantage in density of the dynamic memory more than compensates the higher bandwidth of static memory. All DRAM chips of the three channels are driven by a common 125MHz clock source, which is also used to drive the transmit section of the GE interface, hence they share the same clock domain in the FPGA. Per channel 57 I/O pins are required on the FPGA.

5.3.1.4 CPU

From the early ROB prototyping studies the minimum performance of the ROBIN processor for the buffer management of a single ROL, which is the major task for the processor, was estimated[82] to be around 60MIPS. A brief survey of micro-controllers has identified two appropriate families, IBM PowerPC4xx[83] and Intel XScale. Apart from a bus-interface to attach to the FPGA and a separate memory controller no peripherals are essentially required with the CPU core. Both IBM and Intel provide such simple devices[84], in addition to the fully-featured members[85] of the respective families. For the prototype-ROBIN the PPC405CR was selected for the following reasons:

- Sufficient safety margin with quoted performance of 370 MIPS @ 266 MHz
- Good match of features for application
- Compatibility with integrated processor of successor[86] FPGA families.

As seen from the measurement results, this processor was just powerful enough to fulfil the latest[87] requirements with 2 ROLs in the GE environment. As the required processing power scales pretty linearly with the number of channels, an adequate processor would need approx 50% more performance. For the final ROBIN a similar processor was selected: the PPC440GP, operating at 466 MHz. The 440 core is an out-of-order dual-issue machine with two execution pipelines combined with two integer units and one load/store unit. The 405 core contains a single-issue execution engine, and although it is a scalar processor, the core can perform loads and stores in parallel with arithmetic logic unit (ALU) operations. Reasons to select this type of processor for the final ROBIN were:

- Expected performance improvement by factor of 2 – 2.5, quoted value 932 MIPS @ 466 MHz
- Memory size and bandwidth improvements by factor of 2
- Cache sizes increased to 32 kB each for instruction and data (factor 2 and 4 respectively)
- Identical bus interface to prototype-ROBIN

82 See also UK-ROB project documentation at http://www.hep.ucl.ac.uk/atlas/rob-in/processor.html
83 The PPC4xx family has moved to AMCC in the meantime.
84 IBM PPC405CR, Intel 80200
85 Like PPC405GPr or IOP321
86 XILINX Virtex-2Pro and Virtex-4 are available with integrated PPC405 CPU cores.
87 When the prototype-ROBIN was developed a maximum readout-ratio of 10% was assumed. For the final ROBIN this number was increased to 20%. The prototype-ROBIN conforms to this higher figure as well.

- Software compatible to prototype-ROBIN

This performance gain expected from the quoted MIPS values was confirmed by the results available from “the Embedded Microprocessor Benchmark Consortium”[88], which showed a performance ratio between 2 and 2.5 for different application types (Figure 31).

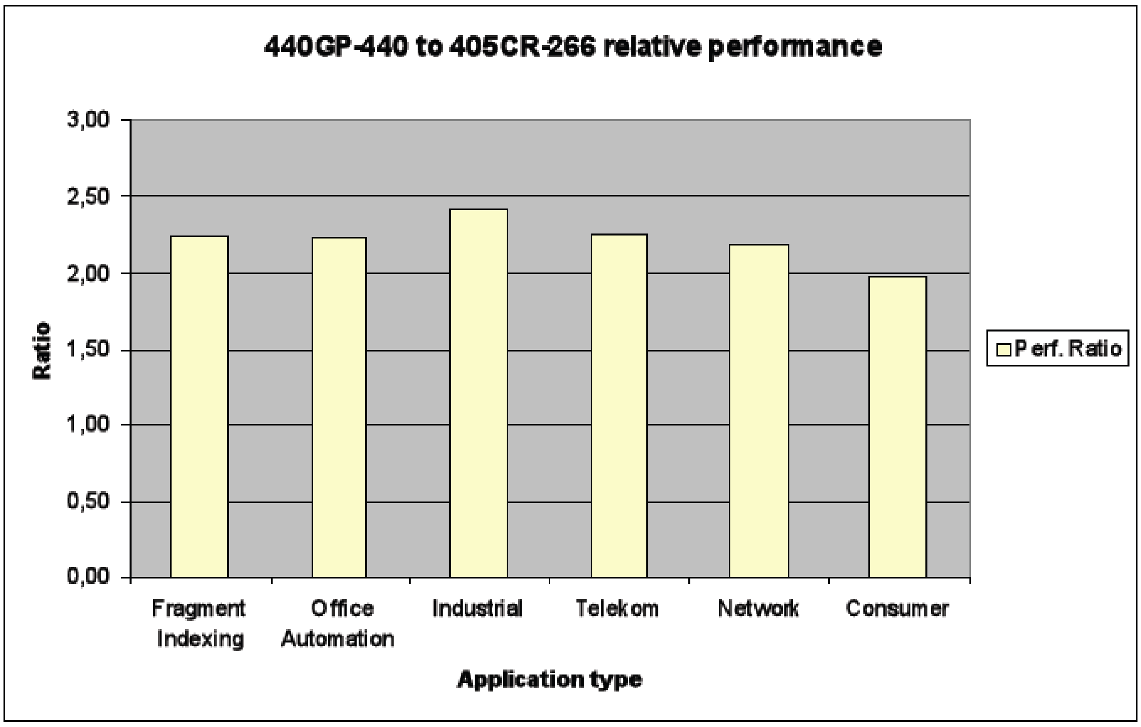


Figure 31: PowerPC performance comparison

A comparison published by IBM[89] led to similar results. A simple test application derived from the ROBIN buffer management code run on a sample set of 20k fragment pages produced 472 kHz processing rate on the PPC405CR compared to 1054 kHz on the PPC440 mezzanine (Figure 32), hence a factor of 2, as expected.

Memory subsystem

The memory subsystem of the CPU is made up from two 512MBit DDR-1 memory chips, providing 128MB total size and 1GB/s maximum bandwidth. The memory is used to hold the application program and a copy of the operating system[90].

Despite the lower frequency compared to the high-speed serial signals of the ROL/S-Link interfaces, the DDR memory is the most challenging part of the PCB layout. To avoid complications, the ROBIN

88 Source no longer available on the internet.
89 http://www-03.ibm.com/chips/power/powerpc/newsletter/jun2003/lead.html compares a PPC440GP with a PPC405GPr (already 50% higher clock speed than the PPC405CR)
90 The ROBIN uses only a simple boot-loader and monitor program as operating system. Refer to software section.

follows the implementation guidelines[91] for DDR-modules of the memory vendor.

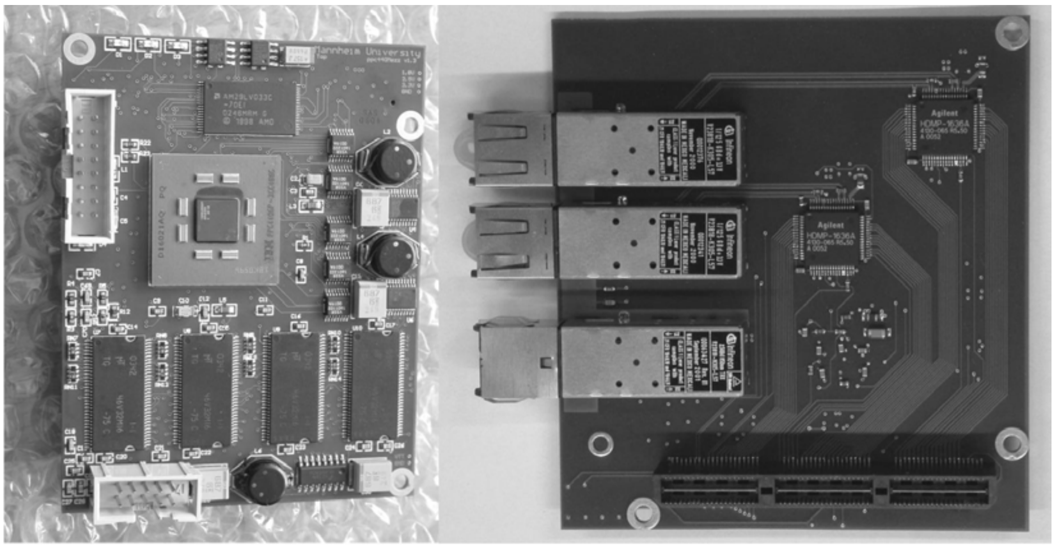


Figure 32: PPC440GP mezzanine

DDR-memories transfer data with both edges of the clock signals, but address and control signal change on the rising edge only. Signals can be logically divided into the three groups CLOCK (C), ADDRESS/CONTORL (A) and DATA (D), as shown in Figure 33.

The clock – which is a 120 Ω differential signal – from the memory controller inside the CPU is duplicated in a PLL clock buffer and distributed to the 2 memory chips. The address and control signals are referenced to the clock only, while data signals are masked and captured by separate DATA-MASK and DATA-STROBE signals, on a per-byte basis. The "A" group and the "D" group – all are 50 Ω single-ended signals – have particular timing requirements, which are translated into length-matching requirements for the traces on the three segments "CPU to series resistor"(38 – 72

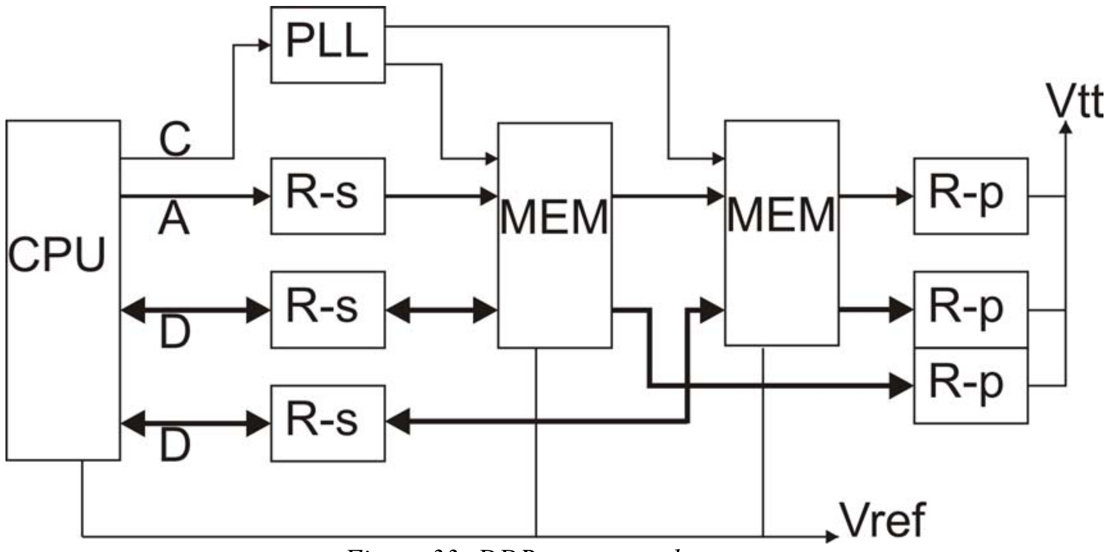


Figure 33: DDR memory subsystem

91 http://download.micron.com/pdf/technotes/TN4607.pdf

mm), “resistor to memory” (10 – 15 mm) and “memory to terminating resistor” (5 – 15 mm). The total length must be between 60 and 80 mm. Length matching of group “A” has to be adjusted to +/- 2.5mm with respect to group “C”. Length matching of group “D” has to be adjusted to +/- 2.5mm with respect to the data-strobe signal of the corresponding byte group and to +/- 12mm with respect to group “C”. Values of R-s and R-p are 22 Ω and 33 Ω respectively.

Board-level simulation was not performed prior to PCB production, as the ROBIN implementation with only 2 discrete memory devices has a significantly lower capacitive load than a memory-module design[92], which operates with 8 to 20 devices in parallel. A brief cross-check of some signals after the production run shows acceptable signal behaviour, see Figure 34.

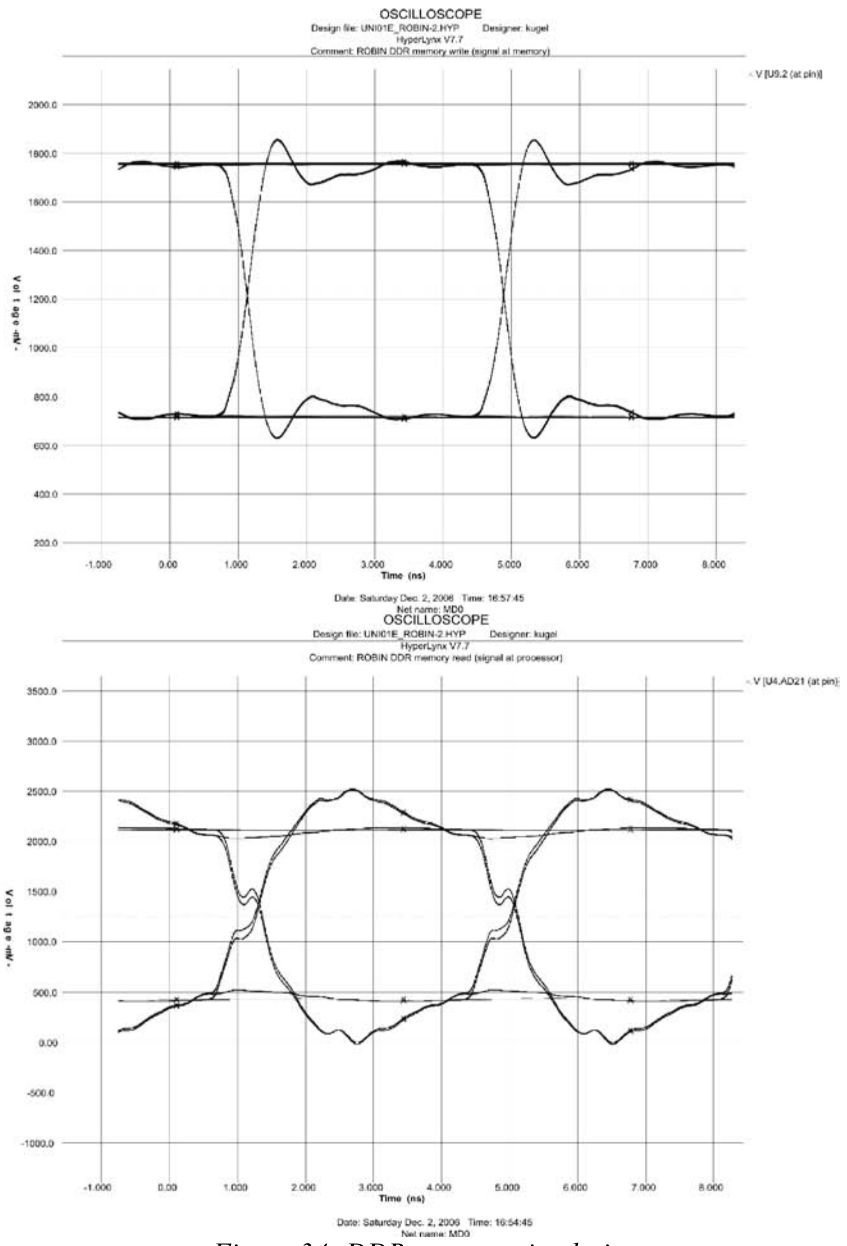


Figure 34: DDR memory simulation

92 http://download.micron.com/pdf/misc/ddrregrev1.2.pdf

FLASH memory

An 8 MB FLASH-memory is connected to the external bus of the CPU, which loads its boot-code and ROM-monitor after power-up from that FLASH. As the external bus is also connected to the FPGA, factory programming of the FLASH can be performed from the PCI interface, via a special FPGA design, while the CPU is held in reset. A special one-time-programmable sector[93] is available in the FLASH which is used to store a serial-number and production data.

External bus

The PPC440GP has two bus interfaces, a PCI-X bus and a so-called "external" bus (Xbus). As the PCI-X protocol is very complex the simpler Xbus is used to connect the CPU and the FPGA. The Xbus operates at 66 MHz with 32 bit non-multiplexed address and data and includes support for external masters and DMA.

The example in Figure 35 shows a series of write cycles. A burst is started by asserting a chip-select signal (PerCS), together with address (PerAddr), data (PerData) and read/write control (PerWE, PerOE). A ready-signal (PerReady) from the peripheral controls the insertion of wait-states after each data word. The burst is terminated with PerBlast.

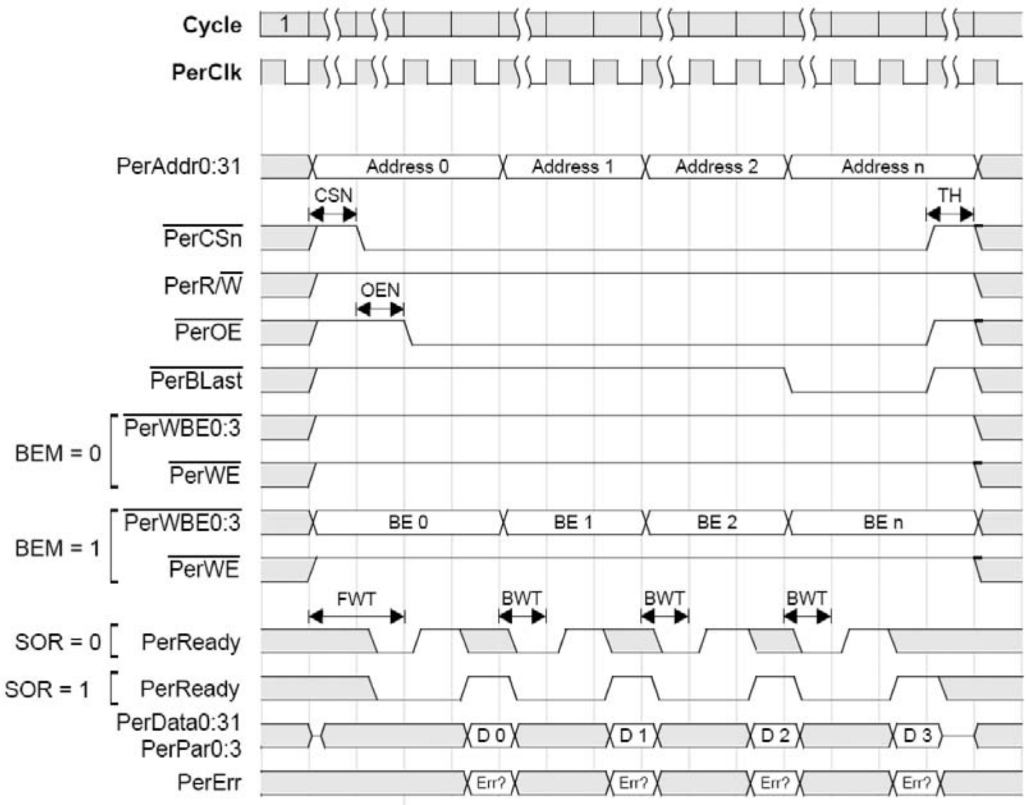


Figure 35: PPC Xbus device-paced burst write timing

93 This feature is called „secure silicon" (SecSi) sector. It provides 256 bytes of user data. This sector can be mapped to the area of sector 0 with a special command.

5.3.1.5 PCI Interface

The maximum data volume to be transferred from a ROBIN to the host PC is around 100MB/s for three channels, at 20% request-rate. Compared to this, the bandwidth required for messages passing is very low: 20 byte per fragment request, 420 byte per delete request of a group of 100 events, which sums up to less than 1MB/s per channel.

The PCI interface is copied from MPRACE-1 and uses a commercial PCI-to-local-bus device from PLX, the PCI9656[94]. On the PCI side it provides a 64bit/66MHz bus, compliant to PCI specification 2.2. On the local side – towards the FPGA of the ROBIN – there is a 32bit/66MHz multiplexed bus (Lbus). The throughput of the device is clearly limited by the local bus to 264MB/s, but the safety margin[95] is sufficiently high: a factor of 2.5. In addition to simple direct-slave PCI mode, the chip provides 2 scatter-gather DMA channels and local-bus-to-PCI direct-master[96] access. The PCI9656 supports PCI-interrupts via mailbox-registers and via a user-pin from the FPGA.

The Lbus interface of the PLX device operates in "J"-mode and is very similar to the PCI bus. A typical direct-slave write operation is displayed in Figure 36. The PXL device accepts the command from PCI, arbitrates for the Lbus using LHOLD and LHOLDA, places the address on the multiplexed address/data bus LAD together with the address strobe ADS. In the next cycle the first 32-bit word is placed on LAD. When READY is signalled from the FPGA the second 32-bit word is sent, together

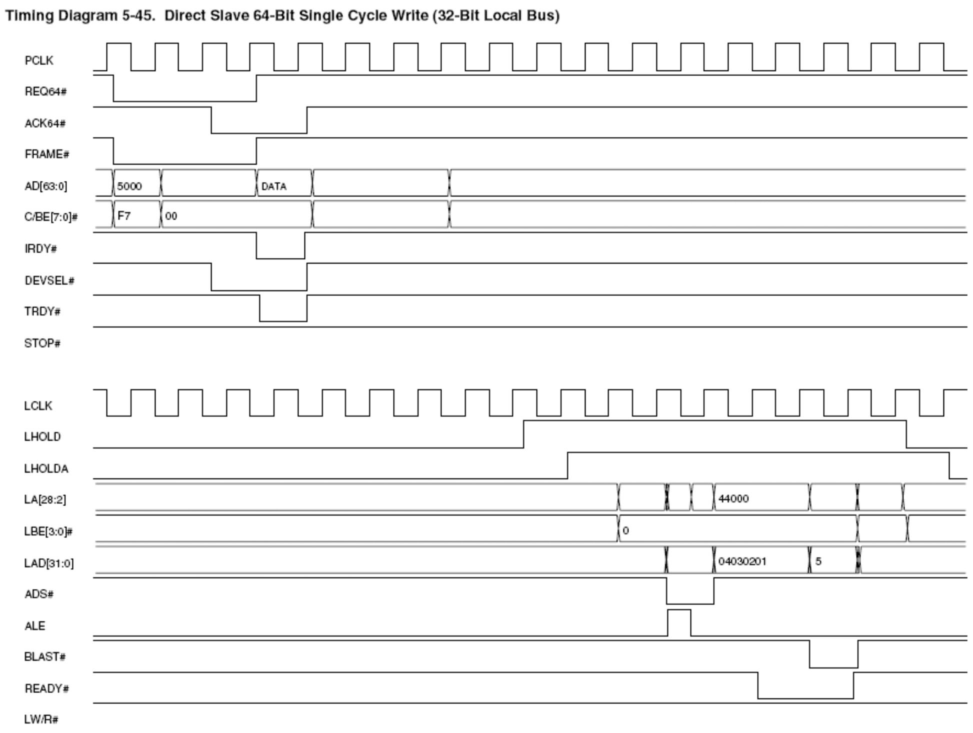


Figure 36: PLX local bus timing

94 http://plxtech.com/download/PCI9000/9656/databook/PCI_9656BA_Data_Book_v1.3_13Jan09.pdf

95 264 MB/s available vs. 100 MB/s required.

96 Both DMA and local-master modes are "initiator" modes on PCI. In DMA mode, the PLX device generates the addresses and access sequence. In local-master mode the device on the local bus generates addresses and access sequence.

with the terminating BLAST signal. Master transfers from the FPGA are done in the same fashion, but the FPGA arbitrates then for the LBUS, and the direction of the signals is reversed. A 256 byte deep FIFO in the PLX decouples the two buses on direct-master writes and provides for maximum throughput.

During normal operation the only communication mechanism is message-passing: direct-slave write-cycles to send messages to the ROBIN and local-master write-cycles to send corresponding responses from the ROBIN to the host. This scheme allows a very efficient utilisation of the PCI-bus, especially when multiple PCI devices are operating concurrently.

5.3.1.6 Network Interface

A GE interface employs functional blocks at the physical and the link layer, which are frequently implemented using two discrete devices, a PHY and a MAC. The selection of the PHY was driven by the aim to support both electrical and optical media. The MARVELL 88E1011S can be used for both GE media and the major MAC interface protocols – GMII and TBI – are supported as well. The prototype-ROBIN implemented both optical and electrical interfaces with automatic media detection. On the final ROBIN only the electrical media was retained.

For the MAC functionality three candidates have been analysed: a XILINX core[97], the LSI8104[98] and the Intel IXF1002[99]. The LSI 8104 doesn't support the GMII interface which makes it less flexible when attaching a PHY device. Also there is no support for VLAN tagging. The XILINX core was considered to be an interesting alternative in principle, but not mature for the prototype-ROBIN. The IXF1002 had all required features and was selected for the prototype-ROBIN. On the final ROBIN, the XILINX core was used, which saved some PCB area but more important allowed easy enhancement with optimisations like MAC address filtering, IP alignment (see 5.3.2.3) and input queue.

Due to the characteristics[100] of the network traffic in ATLAS TDAQ a buffer for a large number of incoming messages had to be implemented. For this purpose a single ZBT device[101] of 2MB size is attached to the FPGA and operated in dual-port emulation mode. Packets received by the MAC core are stored into this external device and a corresponding packet descriptor is pushed into a FIFO. The descriptor is read by the CPU, which subsequently receives the packet from the ZBT. The bandwidth of the ZBT is 264 MB/s, which allows for concurrent READ and WRITE at full GE line-speed.

5.3.1.7 Auxiliary components

Besides the main components described in the previous sections, some auxiliary functions had to be implemented. The most prominent ones are being briefly described here:

97 http://www.xilinx.com/products/ipcenter/GMAC.htm
98 http://www.lsi.com/DistributionSystem/AssetDocument/files/docs/techdocs/networking/8101_8104_DS.pdf. This device is obsolete.
99 Device is obsolete, no online document available.
100 On rare occasions burst containing up to 1000 message may be broadcast, using a non-reliable network protocol (UDP). A large packet buffer helps to minimise packet loss even in such cases.
101 Cypress CY7C1371D-100AXC, http://download.cypress.com.edgesuite.net/design_resources/datasheets/contents/cy7c1371d_8.pdf

- Power supply

  The main supply voltage is 3.3V, taken from the PCI connector. This voltage drives the buffers, GE PHY, PCI interface, the ROL/S-Link interfaces and auxiliary logic. Also, all other voltages are generated from this source. Switching regulators are used to generate 1.5V for the FPGA core supply, 2.5V for the GE PHY, PCI9656, CPU and DDR memory subsystem, 1.8V for the CPU core supply and 1.25V for the termination of the DDR memories. For testing purposes and factory programming, the power can be supplied through an ATX-style 8-pin connector. The power consumption of a ROBIN is around 15W under normal operating conditions.

- Control functionality

  Board level control functionality is realised in a XILINX XC2C256 Coolrunner-2 PLD "CPLD". A board-level RESET signal is derived from signals from a voltage-monitor device, a manual switch and the PCI reset. An 8-position DIP-switch is used to control configuration parameters, e.g. related to JTAG and RESET. The system-clock buffer is supplied with a frequency derived from a 66MHz oscillator, which is simply fed through for normal operation or divided for testing purposes. During power-up the CPU reads configuration values[102] from an I2C-controller implemented in the CPLD. LEDs are driven to indicate RESET and FPGA-configuration status. The topology[103] of the board-level JTAG chain is also defined by the CPLD.

- JTAG interface

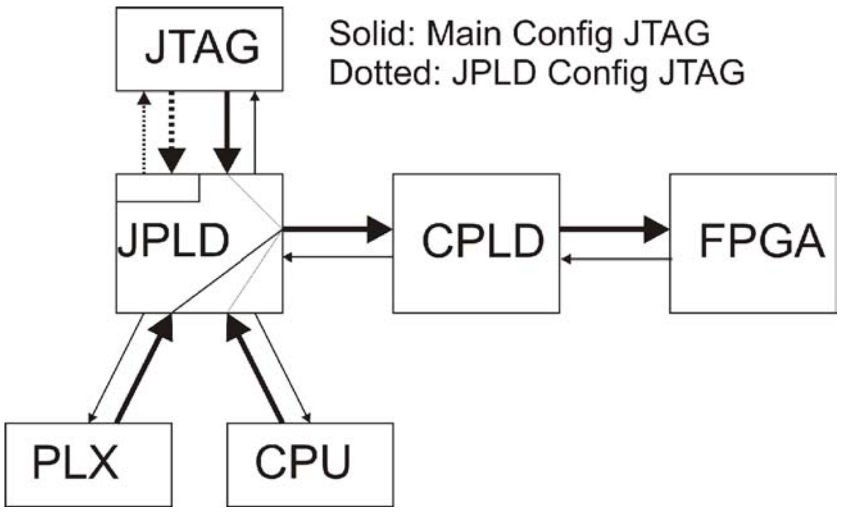


Figure 37: JTAG routing

Not all components[104] of the ROBIN support JTAG, hence using JTAG for board-level testing is limited. Instead, JTAG is mainly used for configuration and debugging. The CPU has a

102 So-called „strapping" values.
103 For testing purposes FPGA, CPU, PLX and PHY devices can be arbitrarily connected to a JTAG chain.
104 E.g. all memory and TLK2501 devices and don't support JTAG

private JTAG connector which allows connecting a JTAG-debugger[105] for software download and testing. Also, the FLASH memory can be initialised that way. JTAG access to the main JTAG chain – comprising Control-CPLD and FPGA – is possible from 3 different JTAG control ports: JTAG connector, PLX PCI-bridge and CPU (see Figure 37). Multiplexing between the three sources is done in a small XILINX XC9536XL PLD "JPLD", the content of this device is assumed to be stable[106]. The JPLD is factory-programmed via a separate JTAG port on the same JTAG connector. The content of the CPLD is stable but can be upgraded from the JTAG connector[107] or PLX JTAG port. FPGA configuration is normally done at power-up time from the CPU JTAG port, but can be done as well from the JTAG connector[108] or the PLX JTAG port[109].

- Testing is supported by a number of on-board test-points for voltages and clock-signals, a 50 signal mezzanine connector attached to unused FPGA pins and CPU based serial and fast-ethernet ports.

5.3.1.8 Realisation

The assembled PCB of the final ROBIN is shown in Figure 38. On the lower left side there are 3 optical transceiver sockets[110] with the TLK2501 SerDes devices next to them to the right. Above the

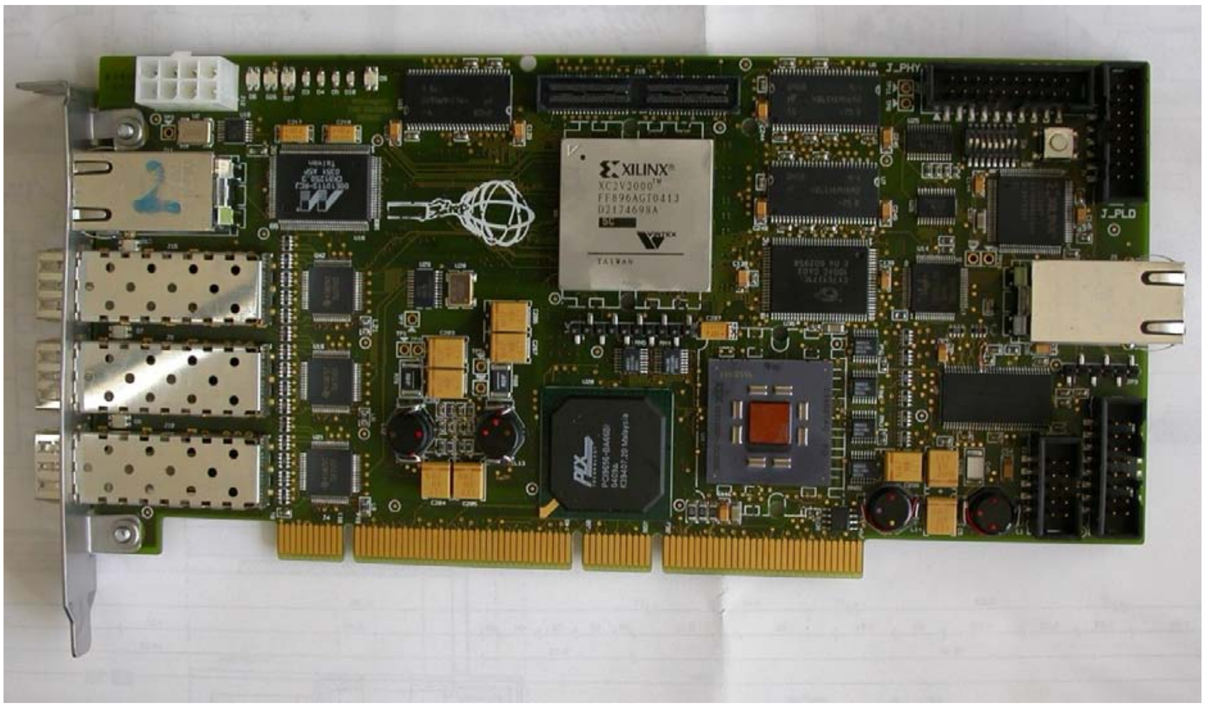


Figure 38: Final ROBIN

105 Abatron BDI2000 or BDI1000 debuggers are used.

106 The prototype-ROBIN does not have the separate JPLD, instead all functionality is in the single control CPLD. This makes in-situ firmware upgrades of the CPLD impossible.

107 The JTAG connector is used for factory-programming of the CPLD content.

108 The JTAG connector is normally used to verify proper JTAG operation of the FPGA during factory testing

109 The PLX JTAG port is normally used during factory-programming and to test new FPGA bit-streams before making them resident.

110The optical transceivers are hot-pluggable and not installed on this picture.

socket there are the GE connector (number “2” written on) and the GE PHY device. The ATLAS logos covers the free space towards the FPGA, which was occupied on the prototype ROBIN by the GE MAC device, which is now internal to the FPGA. Above the logo there is one of the buffer memory chips (there are 2 chips per channel, one each on opposite sides of the board) and below there is the switching power regulator which generates 2.5V and 1.5V from the 3.3V source on the PCI connector. Right to the logo one can see the FPGA and right to the power regulator the PCI bridge. Next to the bridge there are the PowerPC processor, the power regulators for 1.8V and 1.25V and one of the two chips of the processor memory. At the edge of the PCB are the (smaller) connector for the serial port of the processor and the (larger) connector for the processor's JTAG interface. Right to the FPGA we have two more buffer memories (top) and the network packet buffer. At the top right corner of the PCB there are two more JTAG connectors, the reset-button, DIP-switch and the CPLD. The RJ45 connector of the private Ethernet interface of the processor is located in the middle on the right side of the board. On the bottom side of the PCB there are mainly passive components like resistors and capacitors and the remaining memory devices. The connector in the top most left corner is used to power the card in stand-alone operation, for example during factory testing.

5.3.2 VHDL Firmware

The FPGA design is composed from a number of VHDL modules, jointly developed at the three institutes Mannheim, NIKHEF and RHUL and additionally including IP-cores from CERN and XILINX. Figure 39 displays all major VHDL modules.

The main elements used to implement the receiving and storing of event data comprise a ROL-Slice, which is replicated 3 times. The DMA engines on the path towards Lbus[111] and Ethernet incorporate multiplexers, to select one channel at a time. The CPU reads data from various sources, including the UPFs, via a multiplexer. The Control/Status block represents all functionality not related to the main data flow, e.g. CPU access to control and status registers which are available in each of the other modules, reset signals, writing test patterns etc.

In total, the FPGA handles 7 different clock domains. One of them can be viewed as the “system”-clock running at 66MHz. Modules crossing domains from system clock to one of the I/O clocks are indicated with bold borders. Each HOLA-module receives an individual input clock from the TLK2501 transceiver. Together the HOLA-modules share one transmit clock. Both HOLA input and output clocks operate at 100MHz. All buffer memories and the MAC transmit path share a common 125MHz clock. Finally there is a 125MHz receive clock at the MAC. CPU, Lbus-interface and MAC dual-port-memory operate at the system clock. A reasonable grouping of the modules leads to the following functional blocks:

- Input channels: the FPGA internal part of the ROL Slices
- Request channels: the Lbus and MAC dual-port-memory related blocks and the Ethernet MAC (shared with output)
- Output channels: the two DMA engines with the associated header and buffer FIFOs, plus the

111 The FPGA doesn’t connect directly to the PCI bus, but through a commercial PCI bridge device. The local bus (Lbus) of that device is a 32 bit multiplexed bus.

Ethernet MAC (shared with message)

- CPU access: although almost invisible in the diagram there is complex code related to address decoding and accessing control and status registers.

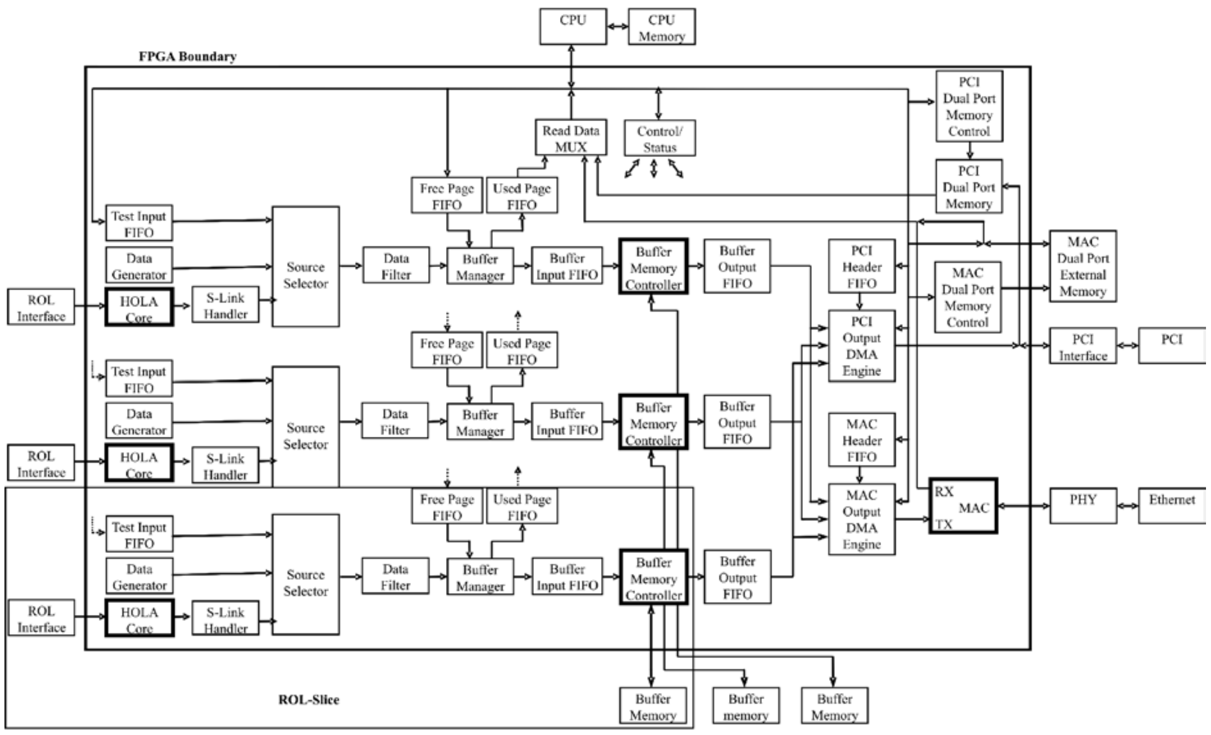


Figure 39: ROBIN VHDL modules

5.3.2.1 Input channels

Each ROL-Slice connects to an external ROL-Interface, which consists of a TLK2501 SerDes device, which in turn connects to an optical transceiver. The serial data received over the ROL is converted to 16-bit parallel data by the SerDes and passed on to the HOLA Core (see 5.3.1.2). Data to be sent back along the ROL (via S-Link Return Lines) is serialised by the SerDes before being transmitted over the ROL. S-Link is a unidirectional data path with flow-control and low-speed return lines, which are not used in the ROBIN application. The data-path is 33 bit wide – 32 bit of data plus 1 control bit, valid data is indicated via a single flag. If flow-control is turned on at the receiving end, the sending side will stop writing data after some delay, typically a few cycles plus the delay of the media (e.g. the optical fibre). The HOLA core already provides some amount of buffering to compensate the media and source delay. Additional buffering and handling of the upper layers of the transmission protocol is done by the S-Link handler. This module performs a number of consistency checks on the incoming packets and indicates errors to the subsequent stage:

- Framing: data section must be encapsulated by one begin-of-frame (BOF) and one end-of-frame (EOF) control word. If required, the S-Link handler terminates a erroneously framed packet by inserting an EOF control-word.

- Packet length: the minimal length of the data section must comprise the start of the header up to the L1ID. In addition, the actual number of words (header + payload + trailer) must match the packet length indicated in the corresponding trailer field.
- Transmission errors: flagged by the HOLA-core upon CRC mismatch
- Format: the format field in the header must match the expected format.

Apart from the HOLA-core data can be fed into the S-Link handler by a data-generator or a test-input FIFO. The data generator generates S-Link packets of programmable size at maximum speed and is used for stand-alone performance measurements. The L1ID-field of subsequent fragments is automatically being incremented and the pay-load section is filled with a test-pattern. In contrast the test-input FIFO is controlled directly by the CPU and is therefore slow, but more flexible than the data-generator and is used to test all possible S-Link error conditions.

All words going into the S-Link handler are forwarded to the buffer memory module, certain words are accompanied by special flags (e.g. error flags, control-word-flag, L1ID-flag, run-number, etc). The header-positions which trigger the L1ID and run-number flags are programmable.

Buffer-management controls the storing of incoming fragments into the buffer memory and the look-up of the fragments on requests from the TDAQ system. It is the central function of the ROBIN. The approach is based on the paged buffer management scheme developed by the UK group [UKROB], with some optimisations applied. All available buffer memory is logically arranged into pages of fixed but programmable size (typically 2kB). Every fragment occupies at least one of these pages. A free-page-FIFO (FPF) and a used-page-FIFO (UPF) decouple the buffer-manager module from the corresponding buffer-management code on the CPU.

The buffer-manager module (Figure 40) retrieves free pages from a free-page-FIFO one by one. It uses flow-control to stop data if there are no more free pages available. The free-page information generates the starting address for the fragment and this address, followed by the data, is sent to the buffer-input FIFO[112]. A bit in the buffer-manager control register selects if the BOF/EOF control words are retained or stripped off the data stream. For every processed memory page one information record (4 words) is written to the UPF, which contains the following information:

- Page status: error code, fragment termination
- Page address, length of data, trigger-type field
- L1ID field
- Run number field

The information is collected during the transmission of the page and transferred to the UPF in a single cycle of 128 bit. On the CPU side, the UPF represents 4 distinct 32 bit registers and is updated (read) when reading the top-most one (run-number).

The size of the decoupling FIFOs is 1k entries for the FPF and 512 entries for UPF.

112 The buffer-input FIFO provides a FULL flag which could also be used to trigger flow-control. However, the memory bandwidth is guarantied by design to always exceed the maximum bandwidth of the link.

The buffer-input FIFO queues the data to be sent to the Buffer Memory and the addresses to which they are to be written.

The final stage of each input channel in the FPGA is the buffer memory controller which arbitrates between write request (from buffer manager) and read requests (from DMA). The relative priority of input and output is compile-time programmable to accommodate different requirements. However, with the buffer memories operating at 125MHz, the available bandwidth is in the order of 400MB/s thus allowing a simple 1:1 ratio without imposing any limitation. On the reading side there is a triple-port interface, which accepts requests from the two DMA engines and from the CPU. The latter is used only for debugging and must not be activated during normal operation, as the CPU timing is incompatible with the input timing requirements. A read-request is just a command with a starting address, which flushes the buffer-output FIFO and initiates data transfer from the memory. Transfer pauses when the FIFO is full. A “stop” command ends the transfer, flushes the FIFO and prepares the controller for a new command.

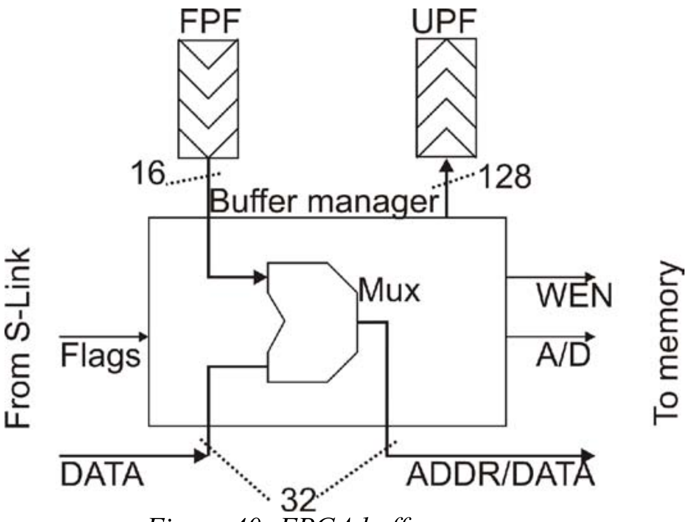


Figure 40: FPGA buffer manager

5.3.2.2 Request channels

The TDAQ system can send requests to the ROBIN through the host PC or directly via the network interface and both paths can be active simultaneously. At the VHDL level, the two interfaces are handled similarly: a request is stored into a dual-ported memory and a request descriptor is written to a FIFO. For the network case, this functionality is explained in [section output channels]. For PCI, the implementation is even simpler: requests from the PCI host arrive via the Lbus at two different address areas, one corresponding to an internal DPR, the other to the descriptor FIFO. The host software calculates addresses, sizes, and keeps track of the filling state of the request buffers; hence, the FIFO and DPR have smaller sizes than the equivalent units do on the network interface, and there are no special controlling units needed.

5.3.2.3 Output channels

The output channels are used to send fragments from the buffer memories as well as any other message data towards the downstream TDAQ system. In all regular cases, a DMA engine controlled by the CPU accomplishes the data transfers. There are a few exceptions to this rule, which allow the Lbus host to bypass the messaging mechanism and directly read particular information from the ROBIN:

- FPGA design id: a 32 bit value identifying version, revision and author of the code
- S-Link real-time status: link-up and x-off
- Embedded serial port: utility module, which connects a host terminal program to the ROBIN processor.

The two DMA engines for Lbus and network access are almost identical, but certain discrepancies in the control information prevented to use a generic DMA engine for both channels.

Every DMA engine monitors its header-FIFO and first retrieves a DMA descriptor block, which defines the number of header words to follow the DMA descriptor, the number of data words from the buffer memory, the buffer memory channel and the starting address in the memory. The DMA engines subsequently transfers all words from the header FIFO. Next, it issues a read-command to the buffer-memory-controller and transfers the requested amount of data from the buffer output FIFO. Zero-length header or buffer fields are possible. For example, a response to a non-data request will have a zero-length buffer field. Similarly, a multi-page fragment going to Lbus will not need header data for subsequent pages.

For Lbus, two additional words – the PCI destination address – are required in the DMA descriptor. When the DMA engine has data available, the Lbus interface stores some words in a small (32 words) internal FIFO, arbitrates for the local bus and subsequently sends the data using the direct-master mechanism of the PLX PCI bridge (see section 5.3.1.5). The same mechanism – but a different address range – is used to write into internal registers of the PLX device.

The MAC DMA-engine provides a feature which simplifies generation of IP-packets. The IP protocol uses 16-bit alignment while the FPGA uses 32-bit alignment. One of the descriptor bits enable the CPU to supply modified IP packets with a 16 bit padding field right after the IP header, which makes the payload section 32 bit aligned. The DMA engine removes the padding element before sending the packet to the MAC unit and sets a flag for the MAC transmitter. Two other flags indicate the start and end of packet words.

Figure 41 displays the MAC unit, which is composed of three major sub-units. The MAC transmitter accepts data and flags from the DMA-engine and stores packets in a double-buffered memory – one packet per buffer – which also converts the 32 bit wide data path to the 8 bit data path of the MAC core running at 125 MHz. The MAC core is a commercial[113] IP-core from XILINX, which supports – together with the external PHY device – a 1000Base-T Gigabit-Ethernet link, including standard Ethernet statistics. Configuration of the MAC core is done via a set of control and status registers, accessible from the CPU. The MAC address is shadowed in a separate register and used by the MAC receiver to validate uni-cast packets. The receiver converts the 8 bit data stream from the MAC core into 32 bit wide packet data, performs IP-realignment by inserting a 16 bit padding element. Packet

113 XILINX supported the research activity by donating this IP-core.

data are stored in the external MAC input buffer[114] and the associated packet-descriptors into a descriptor FIFO. Each packet descriptor indicates the length (in words) and the starting location of its associated packet in the memory. The depth of the FIFO is configurable to 511, 1023, 2047 or 4095 entries. A value of 1023 allows buffering of 1k maximum sized Ethernet packets with the present 2MB external memory. Flow-control is turned-on when either the buffer or the FIFO is almost full.

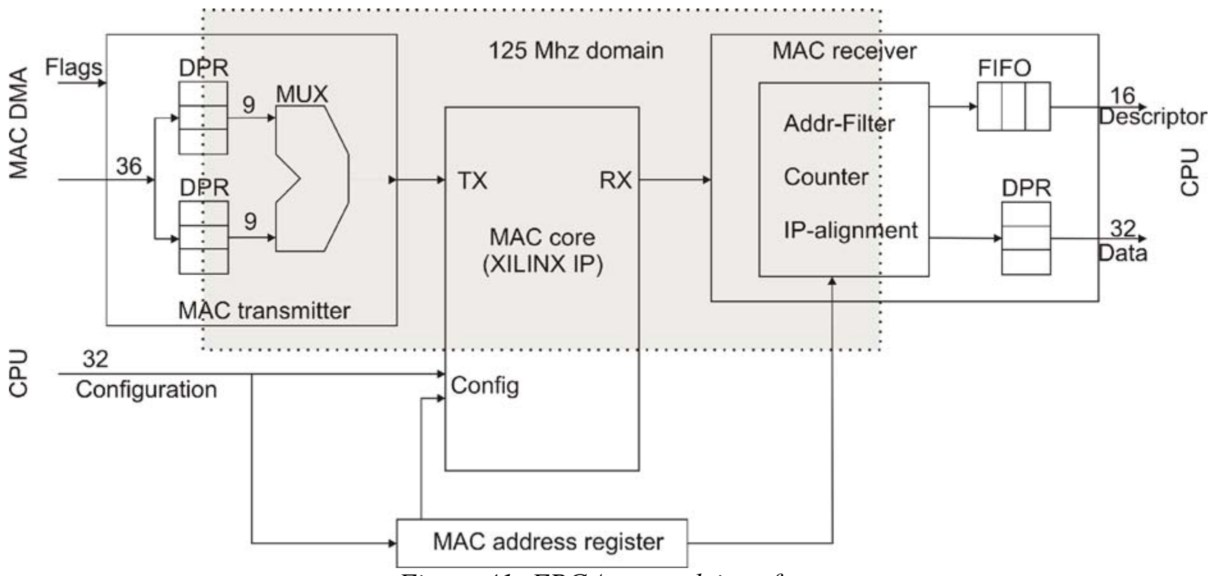


Figure 41: FPGA network interface

5.3.2.4 CPU access

The CPU communicates with the FPGA via a 32-bit non-multiplexed big-endian bus. The Lbus in contrast is a little-endian bus and to avoid confusion the CPU bus is converted to little-endian format right at the edge of the FPGA.

The CPU internally generates a number of chip-select signals, with programmable timing. The FPGA uses two of them to distinguish different areas. “Write”-timing is single-wait-state fixed for both areas.

“Read” timing is as follows:

- Internal registers, FIFO and the buffer memories (for debugging): variable timing, minimum two wait-states, non-cacheable.
- Dual-ported memories: one wait-state fixed timing, cacheable.

The dual-port area must be placed in a separate area to take advantage from burst-mode access of the CPU bus, which is only available when caching is enabled. Clearly, caching cannot be enabled for any kind of FIFO access. The total number of registers and FIFOs feeding the read-data-multiplexer is 43 (plus the 2 DPRs), which requires a prioritised multiplexing scheme, which puts the DPR areas at the higher level and all registers at the lower level with a slower access.

114 The external memory is not a true dual-ported memory device, but a ZBT memory. The dual-port memory controller in the FPGA emulates dual-port functionality by alternating read and writes access on a cycle-by-cycle basis. The guaranteed input bandwidth is still above Gigabit-Ethernet line speed (available: 4 bytes @ 33MHz, required: 1 byte @ 125 MHz = 4 bytes @ 31.25 MHz).

The CPU interface has to generate a number of read-enable and write-enable signals to load or unload the various FIFOs.

There are three DMA channels available, which can be used instead of the normal read-mode access to the three UPFs. CPU software can select between DMA mode and regular mode.

5.3.2.5 Resource utilisation

The design is implemented into a XILINX XC2V2000-5FF896C FPGA and consumes more than 2/3 of the available resources[115], as shown in Table 6. About 70 timing-constraints[116] guide the timing-driven mapping process, which distributes the synthesised logic across the FPGA fabric. Floor-planning is not used, as most of the modules are self-contained and automatically placed close to the corresponding I/O pads and/or according to the constraints.

Utilisation Summay			
Logic Utilization	**Used**	**Available**	**Utilization**
Total Number Slice Registers	14,279	21,504	66%
Number used as Flip Flops	14,265		
Number used as Latches	14		
Number of 4 input LUTs	13,357	21,504	62%
Logic Distribution			
Number of occupied Slices	10,555	10,752	98%
Total Number 4 input LUTs	15,569	21,504	72%
Number used as logic	13,357		
Number used as a route-through	1,227		
Number used for Dual Port RAMs	634		
Number used as Shift registers	351		
Number of bonded IOBs	589	624	94%
IOB Flip Flops	706		
IOB Dual-Data Rate Flops	1		
Number of Block RAMs[117]	50	56	89%
Number of GCLKs	8	16	50%
Number of RPM macros	24		
Total equivalent gate count for design	3,602,489		
Additional JTAG gate count for IOBs	28,272		

Table 6: FPGA resource utilisation

The source-code is implemented in 120 VHDL files with 43000 lines (including comments), plus 18 IP-cores. According to the different tools used at the various institutes the VHDL is quite portable and can be synthesised with XILINX XST, Mentor Graphics Precision and Synplicity Synplify. A VHDL design needs five compilation steps to generate the executable, which is frequently called a “bit-stream”. The total run-time for this project is about one hour. Synthesis and mapping[118] take in the

115 The figure „98% of slices occupied“ doesn’t mean that the device is actually full, as there are still LUTs and Flip-Flops available. The logic is just distributed across the entire device and a denser packing will allow to put more functionality into the device. However, placement and routing will become more difficult.

116 Definitions of clock frequencies, setup-to-clock, clock-to-output and invalid-paths.

117 The present implementation uses only a single buffer on the MAC transmit path, which saves one BRAM compared to the estimation.

118 The mapping process is timing-driven and generates a placement. Place-and-route doesn’t do another placement, just the routing, which makes is quick.

order of 25 minutes each, translate, place-and-route and bitgen use 5 minutes or less each.

5.3.3 Software

The software running on the ROBIN CPU consists of two parts: an open-source boot-loader/rom-monitor called “u-boot” and the “robin” application software. Both packages reside in the on-board flash-memory. After power-up, the CPU loads its initial configuration from an I2C-attached memory and subsequently loads the boot-code from the flash memory. U-boot then initialises the internal peripherals like DDR-memory controller, serial port, external bus and MMU and copies itself into the main memory (DDR). MMU initialisation is done simply by mapping each memory region[119] to a separate MMU entry. The MMU has space for 64 TLB entries, so there is still space for additional mappings. After being initialised, the two serial ports provide terminal access to the monitor functionality. U-boot listens on both serial ports for incoming data and switches the active port to the one most recently active. This is useful when no serial cable is attached and serial communication has to run via the FPGA.

After initialisation, u-boot loads the application binary from a pre-defined address, or enters terminal mode if no application is present. Terminal mode provides a number of useful commands to inspect or modify environment variables, registers and memory, to load programs, and for rudimentary debugging. More information on the commands is given in [ROBMAN].

U-boot on the ROBIN does not support a graphical debugger like “gdb”, so the only debugging aid are “printf” statements, which can be turned on or off under user control if the application was compiled with the debug-option. Furthermore the download of a new binary over the serial interface takes a couples of minutes. Both properties of the system make the software development process a bit inconvenient. To enable shorter turn-around times and improved debugging the main ROBIN application can be compiled in “host”-mode and as such run on the development machine (a Windows PC). In this case a large part of the FPGA functionality is replaced by a simple software model and the PCI communication by a shared-memory structure. A corresponding test program is able to attach to the shared-memory structure, alternative to the PCI communication. This way, both host and embedded application can be executed under the control of a graphical debugger on the development machine, which is much more convenient. Apart from the entire PCI-based message handling the emulation supports the functional test of the fragment processing via upload of event fragments and interrupts, basically everything except network operation. For the test program emulation and physical ROBIN are equivalent, apart from the different initialisation sequence.

The ROBIN application program is a monolithic, single-threaded executable, composed from 10 “.c” source-files and 20 “.h” header-files, in total[120] 11500 lines of “C”-code and 6500 comment lines. The 10 modules implement different functional groups as follows:

- robin.c: the “main” module, which steers the execution of the program with the taks-loop
- robin_bufman.c: handles UPF and FPF including error handling, manages fragment look-up

119 Pre-defined memory regions are: DDR-memory, internal peripherals, every chip-select area on the external bus

120 Lines counted with the „cccc“ utility, available at the open-source repository http://www.sourceforge.net

and deletion

- robin_pci.c: receives and converts requests from the PCI host, initialises Lbus DMA-engine
- robin_net.c: receives and converts requests from the network, initialises network DMA-engine
- robin_msg.c: final decoding and dispatching of requests
- robin_init.c: start-up initialisation and run-time re-initialisation
- robin_util.c: utilitiy functions like terminal output, profiling, etc.
- robin_serctl.c: JTAG configuration of FPGA
- robin_bist.c: self-test functions
- uboot_stubs.c: interface to u-boot system calls

The first five modules comprise the core functionality and are performance critical. The remaining modules contain code which is used infrequently.

The following section explains the most important functional elements of the modules, which are:

- Main task loop
- Buffer management and garbage collection
- Request - response messaging scheme
- Instrumentation
- Configuration and operational monitoring
- Initialisation

5.3.3.1 Main task loop

The tasks comprising the main task loop are buffer management and request handing. Figure 42 shows a simplified view of the loop. The GetFragment function reads the status of the UPF and processes the available pages, up to a programmable maximum number. This is done for all channels in sequence.

Accepted pages are stored in the page database. The next task in the loop is to check for an incoming request from the PCI host. If this fails[121], the loop checks for a request from the network. A request from any of the interfaces is converted into a generic format – however with interface specific information attached – and sent to the request dispatcher. The dispatcher calls the appropriate command – in the example either a data request or a delete request – which finally acknowledges the request via the appropriate DMA-engine.

A few parameters steer the priority of the tasks in the loop: the maximum number of pages processed by GetFragment and a pre-scaling value for each of the tasks. The default values are 10 for the maximum number of pages and 1 for pre-scaling, which provide good performance in the standard (= low request-rate) situation. When a high read-out ratio is required (e.g. ALL fragments have to be

121 In the environment of the ROBIN, only one of the two interfaces will issue requests at a high rate. Therefore, no prioritisation is done here.

retrieved), the GetFragment pre-scaler must be increased to allow more time for request handling.

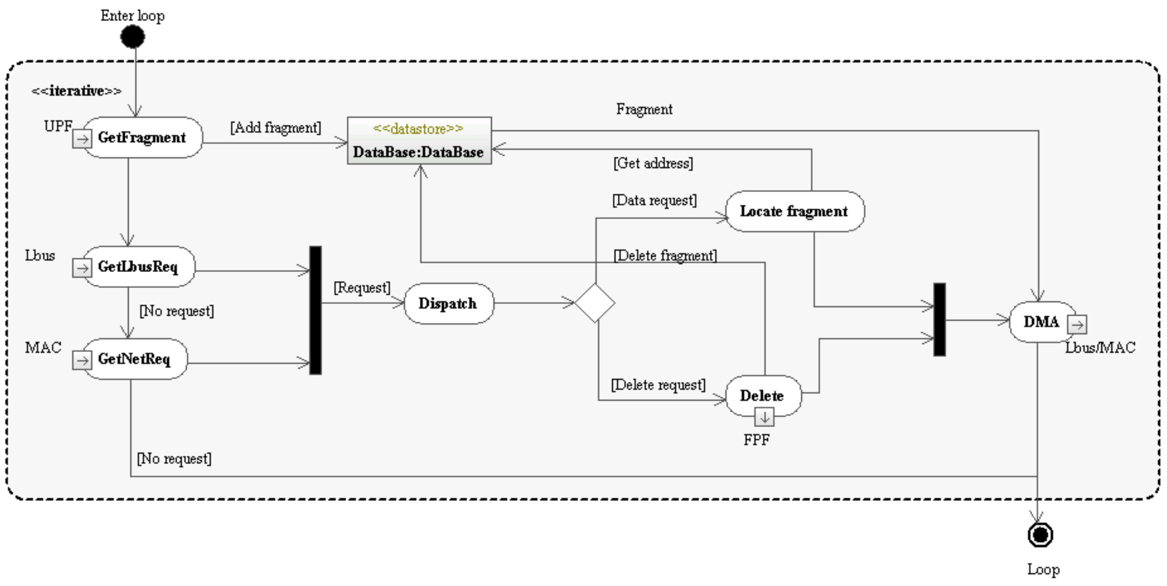


Figure 42: Software task loop

The actual implementation of the task loop is slightly more complex, e.g. the update of the FPF is done in a separate low-frequency task and not directly attached to the Delete function. There is an “idle”-task performing some internal bookkeeping. Input from the serial port branches to a simple debug interface, inside the application.

5.3.3.2 Buffer management

Every page has to be analysed for a number of conditions: new fragment, fragment termination, transmission errors, format errors and sequence errors. The acceptance policy of the ROBIN is to reject as few fragments as possible. Only if the fragment does not contain a L1ID field, it is rejected[122]. For all other error conditions, the fragment is kept and flagged as suspicious. Status and length information from all pages of a fragment are recorded in the corresponding fields of the first page.

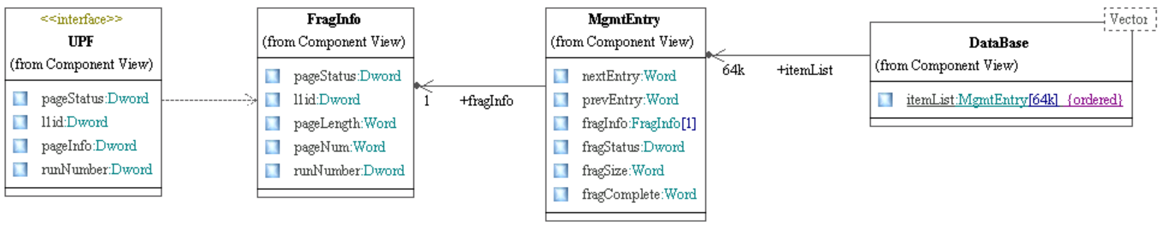


Figure 43: Software buffer manager

Figure 43 displays the data structures used in the buffer management scheme. The UPF provides the

122 A limited number of rejected pages is stored in a separate buffer, which can be retrieved by a special command.

basic information on a per-page basis, which is converted into a FragInfo structure. This structure is then copied to the associated field in the MgmtEntry element, which is indexed by the page number. Next, a hash-key is created from a configurable number of low-order bits of the L1ID field, typically[123] 16. The corresponding entry in the hash-list points to the first element (head) with this key in the item list.

The MgmtEntry pointers are used to create a double-linked list of pages with the same hash-key, terminated with a link to page 0, which is reserved. New entries are added at the head of the hash-lists. Figure 44 shows the mechanism for a few single-page and multi-page fragments.

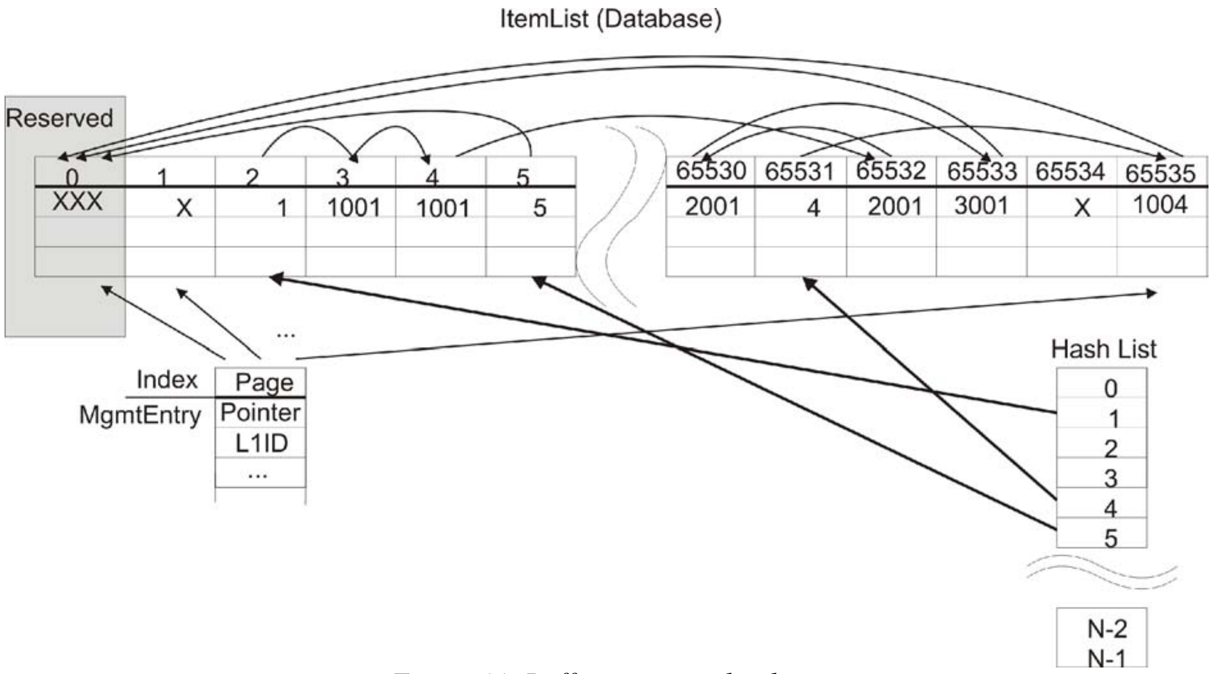


Figure 44: Buffer manager database

Deleted fragments are removed page by page from the linked list and the pages are pushed onto a stack, which keeps all available free pages. When appropriate, the FPF is re-filled from this stack.

ItemList, hashTable and freePageStack are static vectors and consume in the order of 10MB for all channels, which can easily be accommodated by the system.

The interaction between the FPGA part and the software part of the buffer management is shown in Figure 45.

123 Possible values fort he number of bits are 10 to 20. 16 bits will lead to exactly one entry per hash-list with 64k pages and linear ordering of the fragment. More bits will produce sparse occupancy, less bits will increase search time.

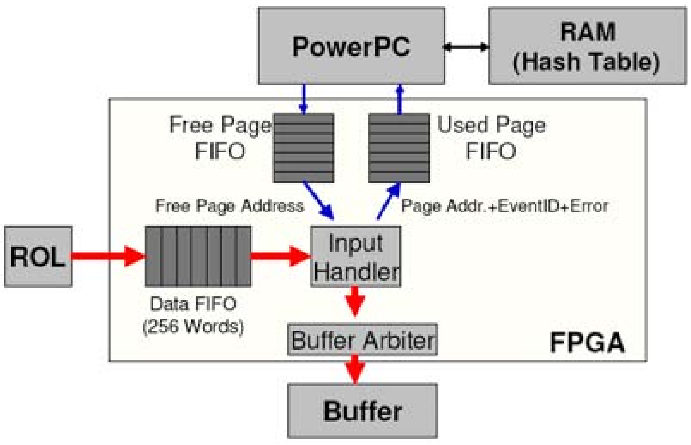


Figure 45: Fragment handling

5.3.3.3 Request – Response messaging

All communication between the TDAQ-system and the ROBIN uses a message-based[124] request – response scheme. The generic format of a ROBIN message is very compact and this format is used directly for bus-based transactions. In general, a request message specifies a command, selects a channel, indicates the address to which a reply shall be sent and contains information specific to the command. After completion, the ROBIN sends a response message to the source[125] indicated by the request. A transaction-identifier helps to verify proper handshaking. Receiving requests from a PCI host is simple, as it stores them into the Lbus DPR already in the right format.

For transactions over the network, the ROBIN has to accept requests defined by TDAQ message-passing protocol [DC022] – which are transported via UDP/IP – and to convert them into the generic format. Due to limitations of the early TDAQ software, a single ROBIN initially had to present itself as multiple ROSes, each with a single data channel[126]. The current implementation assembles the data from multiple channels, just like a ROS-PC does. The UPD implementation of the ROBIN supports response messages of arbitrary size, but only single packet requests. This does not impose a limitation, as data requests and delete requests smoothly fit into a single Ethernet packet and the TDAQ-system does not send other messages to a ROBIN over the network. There is one exception to the request – response rule: network delete messages are typically sent to a multi-cast address; hence, the return address is unknown and no response is generated.

In both cases, the media specific software module (robin_pci or robin_net) attaches a media specific control block to the (converted) request. For PCI communication, the control block keeps only the return address. For network communication, the control block must be initialised with proper Ethernet-, IP- and UDP header information. Subsequently the message with the control block is delivered to the request dispatcher (see also Figure 42).

124 A message in the context of this document is a logical unit of information, transporting control and/or data.
125 Normally the response goes back to the originator of the request, but this is not required by the protocol. However, the present implementation uses always the same media for request and response.
126 Channel selection is done via different IP-addresses on the same MAC-address.

Once the ROBIN has completed the command, it generates a response message. Data responses require a TDAQ-wide defined "ROB"-header [RAWFMT] preceding the fragment data from the buffer. All other responses use a generic ROBIN header, eventually followed by command dependent data[127]. All responses except data responses are handled with two unique DMA transfers:

- Main transfer: DMA-descriptor, header and response data are assembled into a contiguous memory block and pushed[128] into the Lbus header-FIFO. The DMA-engine sends the block to the host PC.
- Completion: A special block with only a single magic word is send via DMA to the starting address for the response, indicating end of transmission.

Data responses require special handling, as they may transfer data from different, non-contiguous, memory pages in a buffer. For PCI, the first transfer takes the header and the first page of the fragment. Subsequent pages are sent without header data. A completion transfer terminates the response. The mechanism for request and response is shown in Figure 46.

For the network, there are two additional complications:

- The maximum Ethernet packet size requires larger pages to be split
- Ethernet- and IP-header are needed on all packets and must be constantly updated

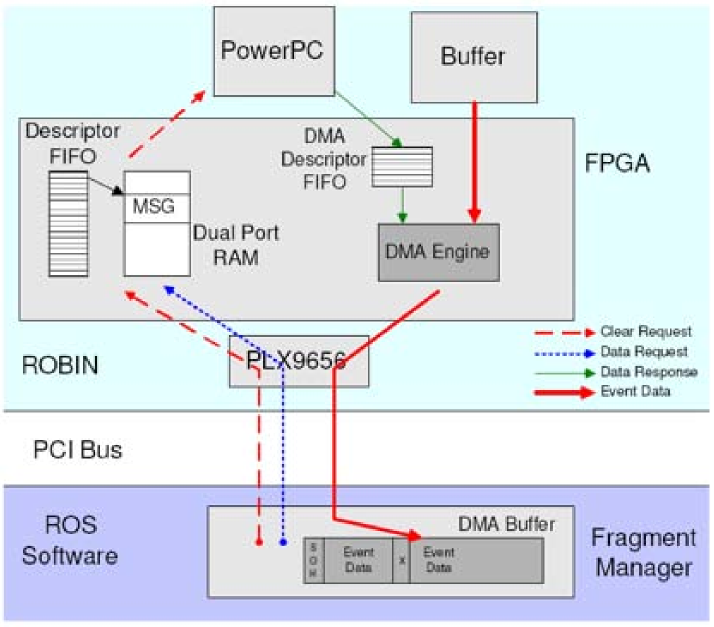


Figure 46: PCI fragment request and response

The software overhead (compared to PCI) to process network responses is significant, even if a completion transfer is not required.

127 For example, a delete request generates a list of L1IDs, which did not match any available fragment.
128 Memory-to-memory DMA might be used in this case, but is not yet implemented.

5.3.3.4 Instrumentation

The use of the GNU profiler utility “gprof” requires a file-system to deposit the profiling data, which is not available on the ROBIN. In order to get parameters though for the TDAQ modelling approach, all critical functions are equipped with custom timing measurement macros. Up to 16k execution-time values per interesting code section are recorded. The average values are printed on the terminal at the end of the application. The performance penalty compared to the non-instrumented code is in the order of 10%. As shown in Figure 47 CPU performance is mainly consumed by three functional blocks: handling of fragments, handling of messages and updating of the free-page-FIFO. The execution times obtained from running the instrumented code were use to predict the performance of the final code. A comparison of the predicted performance and the real performance will be presented in section6.1 .

An important debugging tool is printout to the serial terminal. Clearly, printing information permanently has a disastrous impact on the performance and makes the application virtually unusable. To overcome this, a debug print macro allows switching the printout dynamically on or off. For example during testing of the error handling, the applications starts-up with debug output disabled, then printout is enabled just before erroneous fragments are inserted, to display all details of the fragment processing. The application – with debug output disabled – still reaches about 50% of the performance of the non-instrumented version.

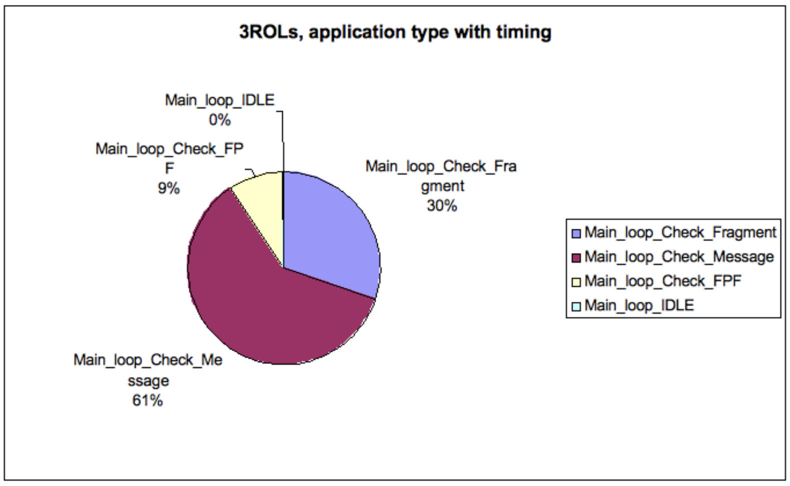


Figure 47: Software execution times from profiling

The timing and debugging macros do not generate any code in the regular (non-instrumented) application version.

5.3.3.5 Configuration and initialisation

A set of about 40 configuration parameters (section 8.2) controls certain aspects of the the operational behaviour of the ROBIN. The initialisation sequence scans the FLASH memory for defined parameters during the start-up of the ROBIN application and loads either the defined values or the hard-coded default values.

Approximately one third of these parameters relate to debugging or to special testing conditions. For example, the input handler module of the ROBIN can be forced to start without waiting for a valid input fragment, which is useful when new hardware is attached to the input links. Test functions enable running the ROBIN without an external data-source but with an internal data-generator instead. Fragment size and various data patterns can be specified in this mode of operation. A verbose mode can be enabled when the the debug version of the application software is loaded, which prints detailed information of every activity to the serial port.

Configuration parameters relevant for regular operation are for example IP address, detector source ID, maximum size for input fragments – above which truncation occurs – and start/stop signalisation for the input handler module.

Most of the configuration parameters can be arbitrarily modified at run-time by the host software. Some of the parameters – like the size of the memory pages – need a re-initialisation of the channel-specific management structures. This can be done while the other channels remain operational. A few parameters however, for example the IP address, require a full re-start of the ROBIN application. This is accomplished by writing the new value of the parameter to the FLASH memory, followed by a reset of the ROBIN. The new value is then loaded from FLASH during the initialisation sequence.

A consistent view at the configuration parameters for both the local ROBIN application and the host software is achieved by generating a special header file "robin_cfg_map.h" from a version of the ROBIN application, which is compiled and executed on the host. This file contains the textual identifiers, default values and classification flags which are needed for proper handling of the configuration parameters by the host software.

5.3.3.6 Operational monitoring

The ROBIN performs operational monitoring by gathering statistics of all important performance figures at the input and the message interfaces, in total about 60 individual numbers (section 8.3) including:

- Numbers of consumed, deleted and available pages
- Numbers of received, requested and deleted fragments
- Numbers of data-request, delete and other messages
- Error counters

In addition, the ROBIN monitors the input link status, the temperature alarm signal and the most recent event ID and builds histograms of the fragment sizes (16 bins in units of ¼ of the page size) and buffer filling state (16 bins).

The accumulated values are reset upon a special command from the host, which also has to compute performance figures, if desired, as the ROBIN does not use reference timing information.

5.3.3.7 Self-test functions

The Built In Self Test (BIST) is defined as a short and simple standalone facility to test the functionality and connectivity of the main components of the ROBIN and is part of the ROBIN application. The BIST is executed directly after the low-level initialisation of the CPU and configuration of the FPGA every time the application starts including after each reset. An extended version, EBIST, can be run if a more extensive buffer test is required.

No external devices or cables are required to initiate the test; the ROBIN need only be connected to power via PCI or a power cable. The results can be seen on a serial terminal and are provided within the status response of the ROBIN. If the application is able to start and run this self-test, the PPC processor and its RAM can be assumed to be functioning.

The tests of the functional components are divided into functions that are explained below. The FPGA test is the first test. Then the network test is performed once. The order of the other tests, which are executed for each channel, is not critical because there are independent. However, it is probably most efficient to leave the EBIST extended buffer test to last, as it takes the most time to execute and the results of the other tests functions can be read during this time.

FPGA Test

The design ID of the FPGA firmware is located in a register of the FPGA and shows the version and the revision of the firmware loaded. The expected version and revision is defined in the robin.h file. The version must be the same although the revision is permitted to be higher than expected. In the latter case normal operation should be possible but some new or corrected functionality in the firmware might be unavailable.

An incorrect version or a revision lower than expected generates a critical DesIdMissing or DesIdWrong error message respectively. Because the BIST cannot reasonably run under this condition, the application stops with an error. A higher revision results in a warning only.

If the firmware version is acceptable a reset of the FPGA registers is performed and the buffer status registers are read. If the buffer initialisation failed then a ResetError flag is set.

Buffer Test

The buffers store the fragment data for each ROL and each buffer has to be tested separately. Their capacity is 64 MByte each. Each buffer has a buffer controller implemented in the FPGA and, in addition, is connected to the external bus of the PPC.

There are three types of buffer tests:

- Data test – tests data lines (32)
- Address test – tests address lines (24)
- Extended buffer test – write marching “1” and “0” respectively and verify read data

The address and data tests are quick and integrated in the BIST; the extended buffer test takes ~10 sec per buffer and runs in the EBIST only. This is the only difference between the BIST and the EBIST.

At the beginning of each buffer test, the chosen buffer is enabled with the enableBuffer(rolId) function. If this fails, a BufEnableError is set. In the data test one bit is shifted left (starting with

0x00000001) and written to the first address, one junk word to another address and then the value of the first address is read out. If the read out value is not equal to the data the test causes a BufDataError. For data bits lower than 24 the corresponding bit in the address test should also return an error, because these data bits are used there, too. The address test writes the address as data to the address in the buffer for every 24-bit address. So e.g. the address 0x200 contains the value 0x200. If the read out value is not equal to the address the test causes a BufAddrError. One address line is tested at a time. If there is no BufDataError, a connection failure of this particular address line can be assumed. If many address and data errors occur, it is probably a hardware error on the clock signals to the buffer.

The extensive buffer test in the EBIST writes a marching “1” to every address in the buffer. This is like the data test, but covering the whole buffer address space at a randomly selected starting value. After filling the buffer (~2 sec), the data is read out and compared (~8 sec).

The steps are repeated for a marching “0”. If the values read out are different to the marching “1” and “0” respectively, a BufExtError is caused. This is not a critical error, but a warning is issued.

Network Test

The network interface is realized with a MAC (implemented in the FPGA on the ROBIN) and a PHY component. The PHY, which can be controlled via the MAC, has a loop-back function that is used in this test.

If the function used to initialize the MAC function in the FPGA firmware returns an error the NetMacCfgError is set.

The interface between the MAC and the PHY is GMII. If the GMII does not receive a ready signal after sending a command, a timeout occurs and the test causes a NetGmiiTimeout.

The PHY is set into loopback mode (if this fails NetPhyCfgError is set), and a packet of data is written to the transmit buffer of the MAC. The receiver buffer is read and the values are compared with the original data. If they are not equal, this results in a NetRxError. The receiver buffer is read again and if these values are different to the written data, an NetTxError is caused.

A RxError with no TxError indicates that the data has been routed through the PHY back to the FPGA correctly, but could be read out only at a second attempt.

A NetRxError and NetTxError together show that the data are not written correctly or the read values are inconsistent with the written data. This could indicate defective data lines between the FPGA and the PHY.

The loopback functionality of the PHY is turned off at the end of this test.

ROL Test

The connection to the Read-Out-Links is routed via the FPGA to a TLK 2501 SerDes for each ROL. In the FPGA a bypass is implemented which allows the direct control of the TLK control lines. The built in Pseudo Random Bit Stream (PRBS) test and the loop-back function are used for testing the connectivity of all data lines to the TLK and its functionality.

First the FPGA internal resets are asserted to initialize the FPGA (clearing FIFOs, etc.). The bypass and the PRBS are then enabled. If the Status register does not show the bypass enabled, a RolEnableError is set.

Next the RxFifo is read to remove any leftover data and to free the FIFO for the PRBS results. The PRBS is stopped and the results are read from the RxFifo. The TLK indicates a successful PRBS test with the RXERR bit set. If not, the test returns a RolTlkPrbs error. The RxFifo is again read to provide a pre-defined status.

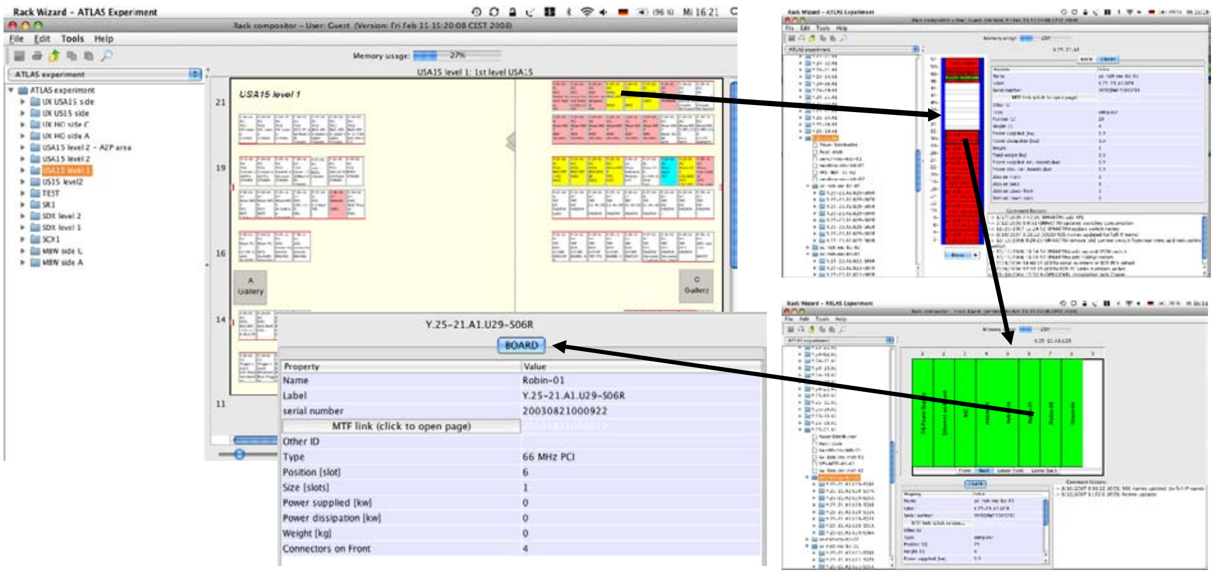
The internal loop-back function of the TLK is used to compare sent and received data. For this random data values are created from the lower time register. Inconsistent data causes a RolTlkLoop error.

If the RolExtLoop environment variable is set to 1, an optional external loop-back test follows. For this test, a fibre must be connected from the output of a ROL to the input of the same ROL. If this test is not successful, a RolTlkExtLoop warning is issued.

The bypass functionality of the TLK is turned off at the end of this test.

5.4 Installation and commissioning

The installation procedure is defined by assembling the ROS-PCs with the ROBIN cards and the PCIe NIC, mounting of the PC in the ROS rack and attaching power and network cables and the S-Link fibres. Also, the operating system is installed. The subsequent commissioning involves the integration of the ROS-PC with the detectors and the TDAQ software framework. In the course of this procedure the functionality of the system and the correctness of the interconnects between all subsystems are verified. To manage the large number of nodes a graphical database utility was created, the ATLAS *RackWizard*, see *Figure 48*. The main pane (top left) shows a single level of ATLAS building USA15, where every box represents a rack. The view of one particular rack is displayed on the top right pane,

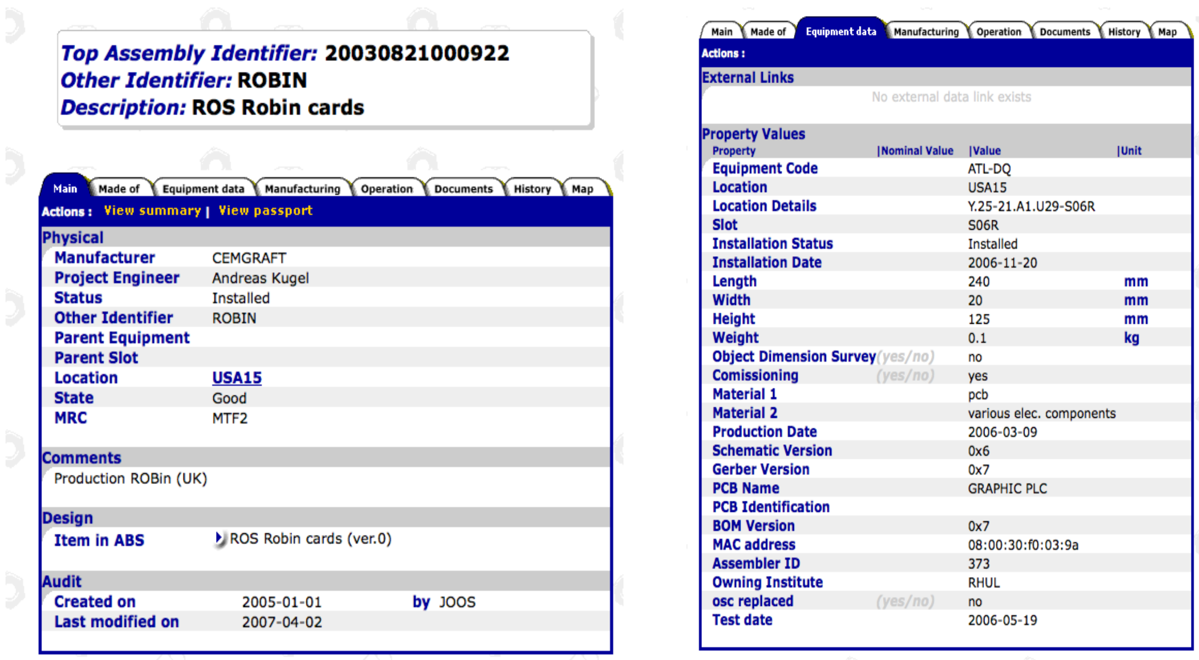


Y.25-21.A1.U29-S06R

Property	Value
Name	Robin-01
Label	Y.25-21.A1.U29-S06R
serial number	20030821000922
MTF link (click to open page)	
Other ID	
Type	66 MHz PCI
Position [slot]	6
Size [slots]	1
Power supplied [kw]	0
Power dissipation [kw]	0
Weight [kg]	0
Connectors on Front	4

Figure 48: Rack wizard

with 8 ROS-PCs in the lower area and the switches at the top. The lower right pane shows the rear view of one of the PCs, with the installed components. Finally the lower left pane displays the entry for one of the ROBINs with the serial number and “geographical” position within the rackWizard's address space. The full database information can be accessed from there via the link to the MTF[129] database (Figure 49).



Top Assembly Identifier: **20030821000922**
Other Identifier: **ROBIN**
Description: **ROS Robin cards**

Physical

Manufacturer	CEMGRAFT
Project Engineer	Andreas Kugel
Status	Installed
Other Identifier	ROBIN
Parent Equipment	
Parent Slot	
Location	USA15
State	Good
MRC	MTF2

Comments

Production ROBin (UK)

Design

Item in ABS	ROS Robin cards (ver.0)

Audit

Created on	2005-01-01	**by**	JOOS
Last modified on	2007-04-02		

External Links

No external data link exists

Property Values

Property	Nominal Value	Value	Unit
Equipment Code		ATL-DQ	
Location		USA15	
Location Details		Y.25-21.A1.U29-S06R	
Slot		S06R	
Installation Status		Installed	
Installation Date		2006-11-20	
Length		240	mm
Width		20	mm
Height		125	mm
Weight		0.1	kg
Object Dimension Survey	(yes/no)	no	
Comissioning	(yes/no)	yes	
Material 1		pcb	
Material 2		various elec. components	
Production Date		2006-03-09	
Schematic Version		0x6	
Gerber Version		0x7	
PCB Name		GRAPHIC PLC	
PCB Identification			
BOM Version		0x7	
MAC address		08:00:30:f0:03:9a	
Assembler ID		373	
Owning Institute		RHUL	
osc replaced	(yes/no)	no	
Test date		2006-05-19	

Figure 49: MTF Entry

At the start of the LHC on September 10th 2008 the ATLAS TDAQ system was commissioned with the full dataflow system and an HLT farm with 6800 processing cores[130] for L2 and EF corresponding to approx. 35% of the final size [ATLCOM].

5.5 Summary

The requirements on the ROBIN as a component of the ATLAS ROS have been presented. A single PCI card must accept 3 optical input channels operating at 160MB/s. Event fragments arrive at up to 100kHz on each link and are stored in 64MB page-organised buffers. Requests to transmit or delete event fragments arrive over a PCI or GE interface at up to 21kHz, depending on the ROS architecture – bus based or switch based. In addition to these basic functions complex operations related to configuration and monitoring are required. Continuous updates and adaptation to various detectors require a flexible design of the component. The hardware platform of the ROBIN is composed from a high-density FPGA device, which handles all high-speed and high-rate real-time operations and data movements. Data transfer towards PCI or GE are performed by DMA engines inside the FPGA, which provide direct paths between buffer memories and target interfaces. The FPGA application consists mostly of custom VHDL code plus external library elements for the S-Link and GE interfaces.

A high-performance embedded processor performs all complex task like message processing, event

129 MTF is the “manufacture and test folder “of the main ATLAS ORACLE database.
130 Many installed machines are equipped with multi-core processors, so the number of machines is lower.

bookkeeping and monitoring. The processor interacts with the FPGA through a 32 bit local bus with DMA capabilities. The embedded software is a monolithic "C"-application which runs on top of a simple boot-loader and monitor program. All binaries are resident in a local FLASH memory and booted after power-up.

Sophisticated tools for self-testing and to aid software development are available. A CERN-based component inventory database keeps track of all ROBIN cards and provides status and location information.

6 Results

This chapter presents the results obtained with the ROBIN, presented in different views relating to the different requirements areas. The first section describes the test setups and measurements used to characterise the performance of the ROBIN as a component. The system performance of the base-line ROS is presented thereafter. A reasonable test environment for the switch-based ROS is still under preparation, therefore the system performance of the switch-based ROS will be presented elsewhere. Issues which occurred during the development process are presented in the last section, which also covers reliability aspects.

6.1 Stand-alone

The basic setup used to assess the performance of a ROBIN requires:

- a ROS-PC with the ROBIN installed in one of the PCI slots
- a PC emulating the detector via a multi-channel S-Link source card (FILAR[131])
- a PC emulating the TDAQ system
- a GE switch, connecting the ROS-PC, the ROBIN and the TDAQ emulator

In reality, the detector emulator is frequently substituted by the internal data-generator of the ROBIN, which has almost identical properties. The TDAQ emulator needs to represent different elements of the TDAQ system: a DFM, which is responsible to delete events, plus L2PUs and SFIs to request events. Again, there are simplifications: to measure the performance in the PCI environment no external TDAQ emulator is used. Instead, all requests and deletes are generated by the ROS-PC as otherwise the increased load on the ROS-PC would limit the performance. In the network environment the performance of a single TDAQ emulator machine may not be able to saturate a ROBIN. Also, the requests generated by L2PU and SFI are equivalent to the ROBIN. Therefore, the typical setup in this case will contain one DFM emulator and two SFI emulators. The same configuration can also be used to verify that ROBIN and ROS-PC are functionally equivalent[132] from the network point of view. The GE switch must properly handle flow-control messages and should provide virtual output queues (see chapter 3.2) to minimise packet loss.

As described in section 5.3.2 , the movement of data in the ROBIN is handled by hardware. Concerning the incoming bandwidth over the S-Links and the processing rate of memory pages there are basically no limits. The nominal 160MB/s of the S-Link are exceeded by far by the internal data generator (up to 250MB/s) and by the external data source (up to 200MB/s). The maximum fragment rate of the data generator had to be limited to 200kHz in order to stay approximately within the specifications even for small fragments. On the outbound interface, data is sent by the DMA engine from the buffer memory to the PCI bridge or to the network port. The buffer memory has a total bandwidth in the order of 400MB/s, hence the DMA can provide more than 200MB/s to the output port while the input is running at nominal speed concurrently. The throughput of the output ports is limited by the internal design of the PLX PCI bridge to 264MB/s or to the line-speed of 125MB/s in

131 FILAR documentation is avialable at http://hsi.web.cern.ch/hsi/s-link/devices/filar/
132 The ROBIN must respond in the same way to the same messages.

the case of the GE interface respectively.

The expectation for the stand-alone performance is therefore, that it is under regular operating conditions completely dominated by the performance of the on-board processor and that the impact of the hardware design is only visible – in the form of bandwidth limitation – for large fragments or very high rates.

A common complication in assessing the ROBIN performance with a stand-alone setup is the missing synchronisation between event generators and requesters. In the real system the L1 issues event identification to the HLT system and thus assures that requests address only existing events. The stand-alone setups however have to use free-running event generators. Tracking the event number in this scenario can become complicate, in particular if multiple nodes – e.g. L2, SFI and DFM - need to be synchronised to the locally generated events. The synchronisation mechanism has to monitor the value of the most recent event number processed by the ROBIN, which is transmitted via the status of every response message. Another field of the status indicates two cases of missing events: "lost" or "pending". A fragment is "lost" when it is not available in the buffer and the event number of the incoming fragments is (logically) larger than the number of the requested fragment. This is typically caused by a delete message which was issued asynchronously. In contrast, a fragment is "pending" when the request addresses an event with a number larger than the most recent one received by the ROBIN. In this case the request should be repeated.

6.1.1 PCI access

Performance measurements were done by exercising the ROBIN via the PCI message-passing interface with the program *testRobin*. Initially, it generates a number of requests for each of the ROBIN channels and submits them into the hardware queue of the ROBIN, which allows for a maximum of 10 requests[133]. For every response coming back a new request is submitted to keep the pipeline busy. After the steady-state of the operation is reached the program measures the execution time for a certain number of events, typically in the order of a million, to exclude caching effects and the influence of the various hardware FIFOs. The results of measurements [ROBJINST] with an external detector emulator are shown in Figure 50. Performance is expressed as sustained L1 rate as a function of the request rate. For small fragment sizes (100 and 200 words) the L1 rate drops linear with the request rate, which conforms exactly to the expectations. The characteristic parameters in this region are the processing time per fragment t_F and the processing time per request t_R. T_F includes the times to register a new fragment in the bookkeeping system and to delete it later on. T_R includes the times to process the request message and to setup the corresponding data transfer. From the diagram we can derive the parameters[134] to be t_F = 2.04ns and t_R = 4.6ns. The actual data transfer is an overlapping operation, executed by the DMA controller in the FPGA, hence the impact of the fragment size becomes visible only when the transfer[135] time is equal or greater than the processing time, which occurs around 1kB. For larger fragments t_R therefore is a function of the fragment size

133 Recent tests with an increased hardware queue size of 64 entries (~20 per channel) didn't show a significant improvement.

134 Note that the rates are shown per channel, the total rates are 3 times larger.

135 The maximum output bandwidth towards PCI is around 220MB/s

with t_R = size[kB]*4.6ns.

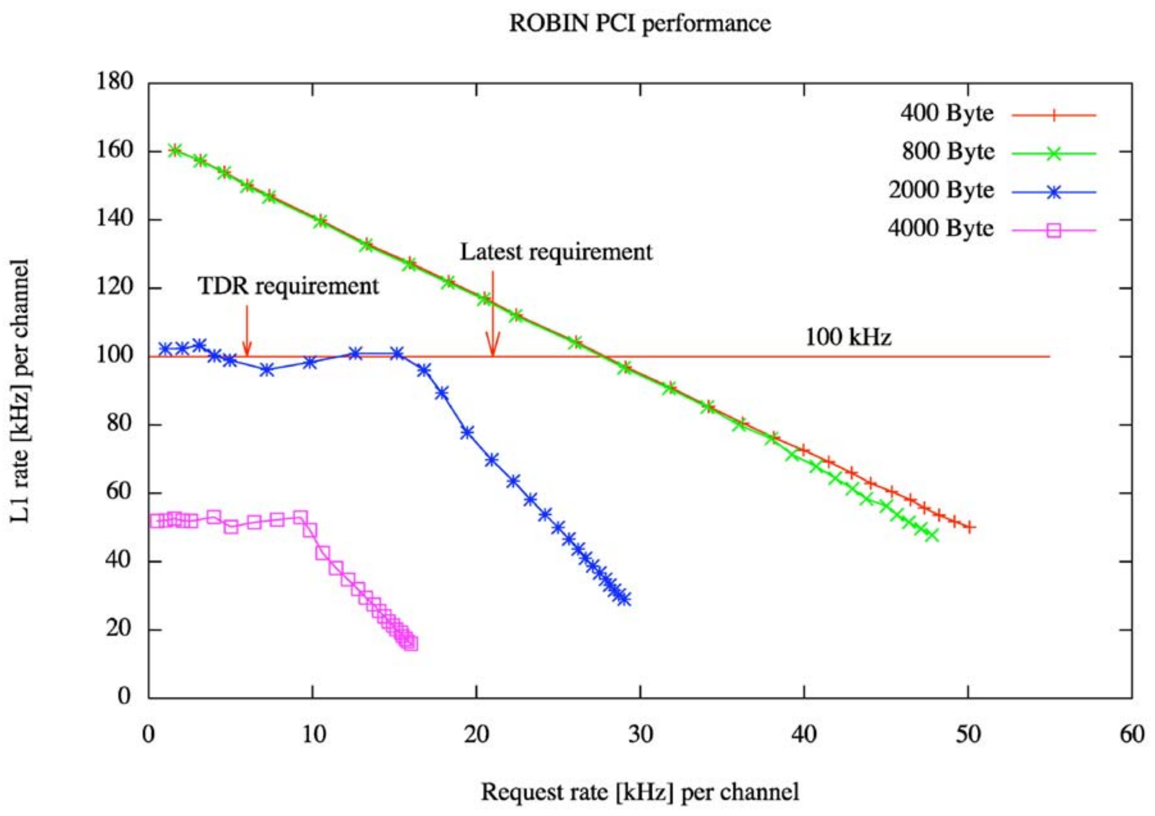


Figure 50: Actual PCI stand-alone performance

The requirements on the ROBIN are indicated by the two arrows, where the left one corresponds to the TDR assumptions for a bus-based ROS under full load and the right one represents the more recent understanding, in both cases for a nominal fragment size of 1kB. There is a significant safety margin available even in the extreme case. In addition to the performance calculation based on the parameters obtained from the measurements the performance can be estimated using the timing values obtained from the instrumented code. The fragment processing time is composed of the times to read the fragment from the FPGA and to enter the fragment into the bookkeeping structure plus the times to receive and process the delete message and finally the time to push the freed page back to the FPF. In total, this sums up to 2.02µs which is very close to the 2.04µs of the initial calculation. The request processing time is composed from message reception and processing time, summing to 3.7µs. The latter value is smaller than the one from the calculation above, which indicates sub-optimal conditions during the measurements. The calculated values are shown in Figure 51, where the "INSTR" prefix indicates values from the instrumented code.

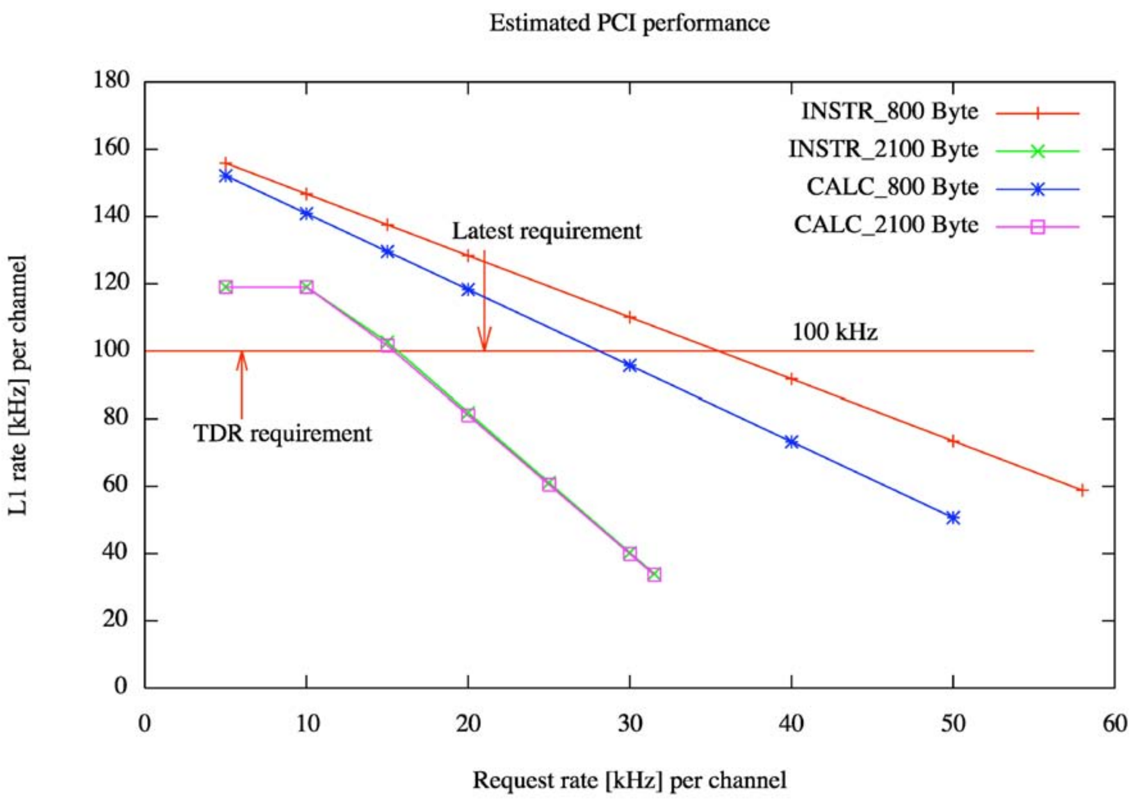


Figure 51: Calculated PCI performance

6.1.2 Network access

In case of the network setup only preliminary results are available, obtained with the simplified program *dcRequester* instead of the combination of DFM and SFI/L2PU nodes and an initial version of the network communication protocol. The same set of measurements was executed and the ROBIN parameters obtained are t_F = 1.85ns and t_R = 12.5ns. The improvement in t_F is a result of the more efficient fragment deletion procedure in this arrangement, where a single message deletes the fragments from all three channels and no acknowledge needs to be returned. In contrast, PCI delete messages are sent individually to the channels and expect an explicit acknowledge. The increase in t_R is due to the much more complex mechanism required to handle the UPD data request messages as compared to the PCI message interface. Figure 52 Shows the performance of the regular application (“normal”) for fragment sizes of 900 and 2000 bytes. In addition, the performance of the instrumented code (“timing”, lower performance) and the code without monitoring function (“xfast”, higher performance) is displayed. For comparison, the estimations based on the parameters from the instrumented code (t_F = 2.13ns, t_R = 12.4ns, label “INSTR”) and from the function fit (t_F = 1.85ns, t_R = 12.5ns, label “CALC”) are shown. It can be seen that already at a fragment size of 900 bytes the performance drops below the prediction for high request rates, which is probably caused by the increasing load on the machine running the requester program. For larger fragments the bandwidth limit[136] of approx. 75MB/s is approached for all 3 application versions (“timing”, “normal” and

136 The FPGA design used for these measurements required a non-overlapping copy of the network packets first from the buffer into a FIFO, then from the FIFO to the network which limits the max. bandwidth to

“xfast”), which corresponds to 12.5kHz request rate for 2kB fragments.

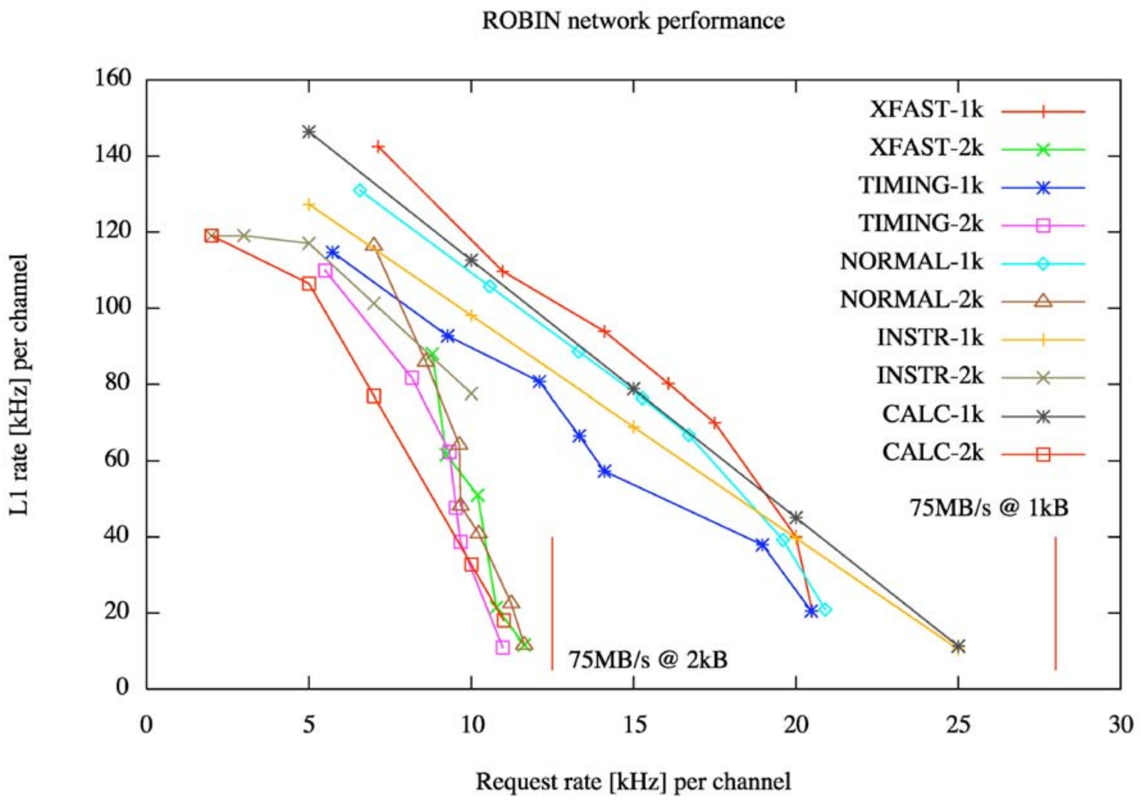


Figure 52: ROBIN network performance

The influence of the operational monitoring can be estimated by comparing the XFAST and NORMAL curves. XFAST yields in the order of 10kHz higher a L1 rate at the same request rate.

6.1.3 Event bookkeeping

As explained in section 5.3.3.2 the bookkeeping procedure uses some of the lower order bits of the L1ID as hash-key to index the events. The number of bits used has an influence on the processing time of the regular event lookup (hashing) during adding or removing of a fragment as well as on the special functions required to remove orphaned fragments from the buffer (garbage collection). The maximum number of bits which can be used is limited to 22 by the memory required to store the hash-tables (corresponding to 24MB for three channels).

All test setups used so far generate and delete events sequentially, such that the distribution of fragments in the ROBIN buffers is very regular and compact. Under realistic operating conditions however this will be different, for example there will be events which require processing times in the EF much longer than the nominal buffering latency of the ROBIN. Also, the rate of lost network packets might be non-zero, leading to residual fragments. The optimisation of the number of hash-bits and also the bit positions composing the hash-key can later on be done on the running system, based

(1/(1/200MB/s + 1/125MB/s)) = 77MB/s.

on the following results.

6.1.3.1 Hashing

Without experience from the experiment a regular distribution of L1IDs in the buffer is assumed. For this approach, the best performance of the hashing is achieved by using as many bits as possible, with the upper boundary being the maximum number of events in the buffer, which is defined by the number of buffer pages. With the typical value of 32k pages the target number of bits for the hash-key

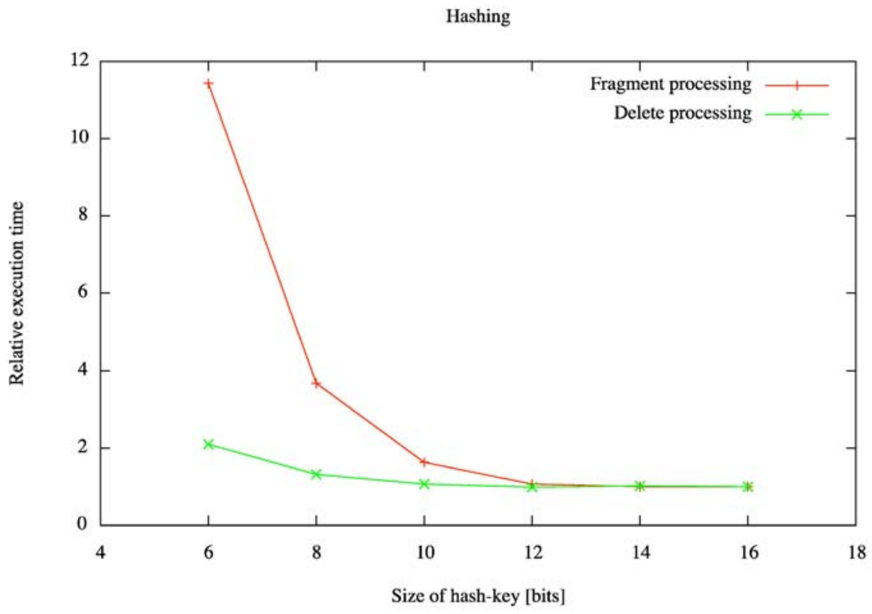


Figure 53: Effect of hash-key size

is 15. Figure 53 displays the two software functions which are affected by the hashing mechanism - fragment processing and event deletion – and the impact of the size of the hash key for a buffer size of 32k pages.

6.1.3.2 Garbage collection

Garbage collection consists of 2 steps, a linear scan of all buffer scan for fragments, followed by the selective deletion of fragments which fall outside the validity range. As shown in Figure 54, the deletion time is almost constant for a small number of buffer pages (“DEL-4k”) while for a larger number of pages the dependency on the size of the hash-key is obvious (“DEL-32k”). The building of the fragment list is equal in both cases (“BF-32k”, “BF-4k”) for hash-key sizes up to 12 bit, which corresponds to the number of pages of the small buffer. If the hash-key size is increased further in this case, a negative effect is introduced by the sparse occupancy of the hash-table, which is caused by the regular event processing in the test setup.

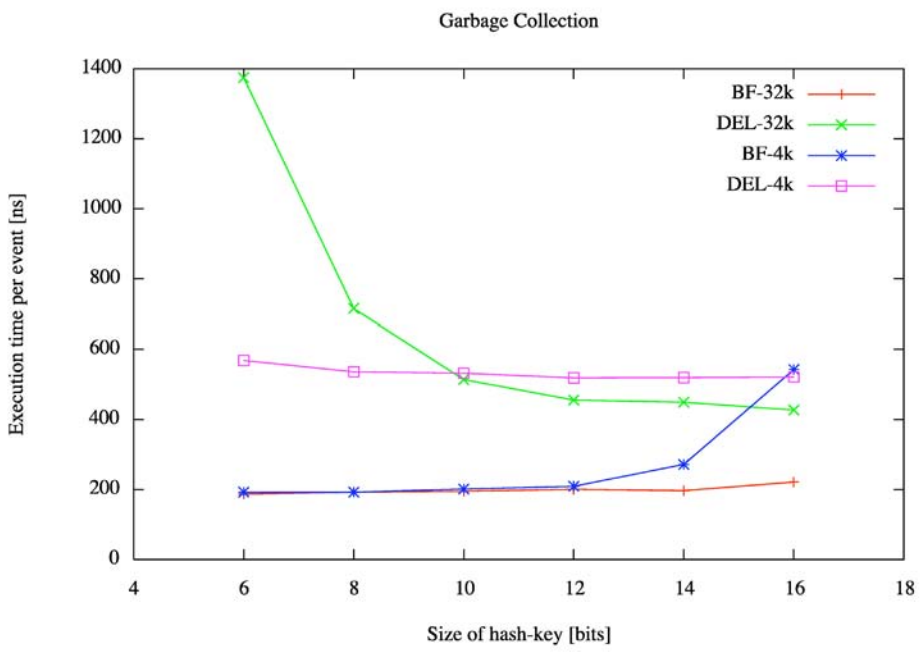


Figure 54: Garbage collection performance

6.2 *System performance*

System performance has been evaluated first at the level of a standard ROS-PC. This can be viewed as an extended test setup. In addition, initial measurements of the performance of the entire dataflow system are available from the first beam period.

6.2.1 Bus-based ROS

The baseline bus-based ROS is evaluated using an arrangement of a ROS-PC – equipped with 4 ROBIN cards – attached to 2 PCs emulating the dataflow system via 2 GE links (Figure 55). Data input is taken from the hardware data generators inside the ROBINs.

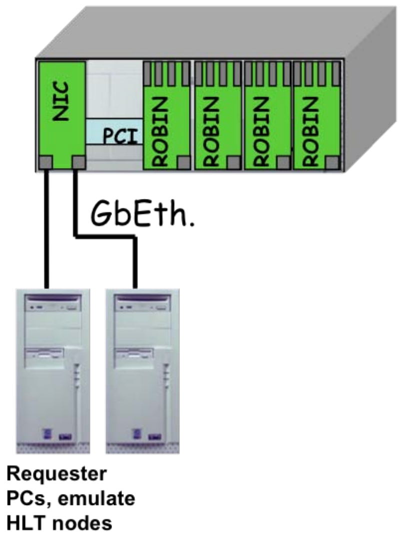


Figure 55: ROS test setup

The requirements have evolved since the TDR, which requests 3kHz of EB plus 17kHz of L2 for 2 ROLs at 100kHz1 L1 rate in case of the "hot-spot" ROS. The current assumptions used for the latest measurements [TIPP09] are 3kHz of EB plus 18kHz of L2 for 3 ROLs at 100kHz L1 rate. In the even more demanding case of a scan of the full sub-detector by L2 the RoI size grows to 12 ROLs which is equivalent to an EB rate of 21 kHz. Figure 56 shows the dependency of the total request rate (composed of a fixed EB portion and a variable L2 portion) to a ROS-PC on the fragment size, at a fixed 100kHz L1 rate, for the standard configuration and the recently optimised configuration. It can be seen that the original performance requirement is met by a standard ROS-PC, however only for small fragments at the RoI size of 3. After optimisations of the software, in particular by using a uni-processor kernel, tuning off hyper-threading and the security features of Linux, and by tuning the interrupt-coalescence of the network interface card, the situation improved and the performance for RoI sizes of 3 is now well above the requirements. It is to mention that the average load on the individual ROBINs in the system is relatively low, per channel only 6kHz for an RoI size of 2 and 8kHz for an RoI size of 3 respectively at the target figure of 21kHz total request rate on the ROS-PC.

High EB rates are problematic still, because the output bandwidth limit of the 2 GE links is reached already at 20kHz for the nominal 1kB fragments.

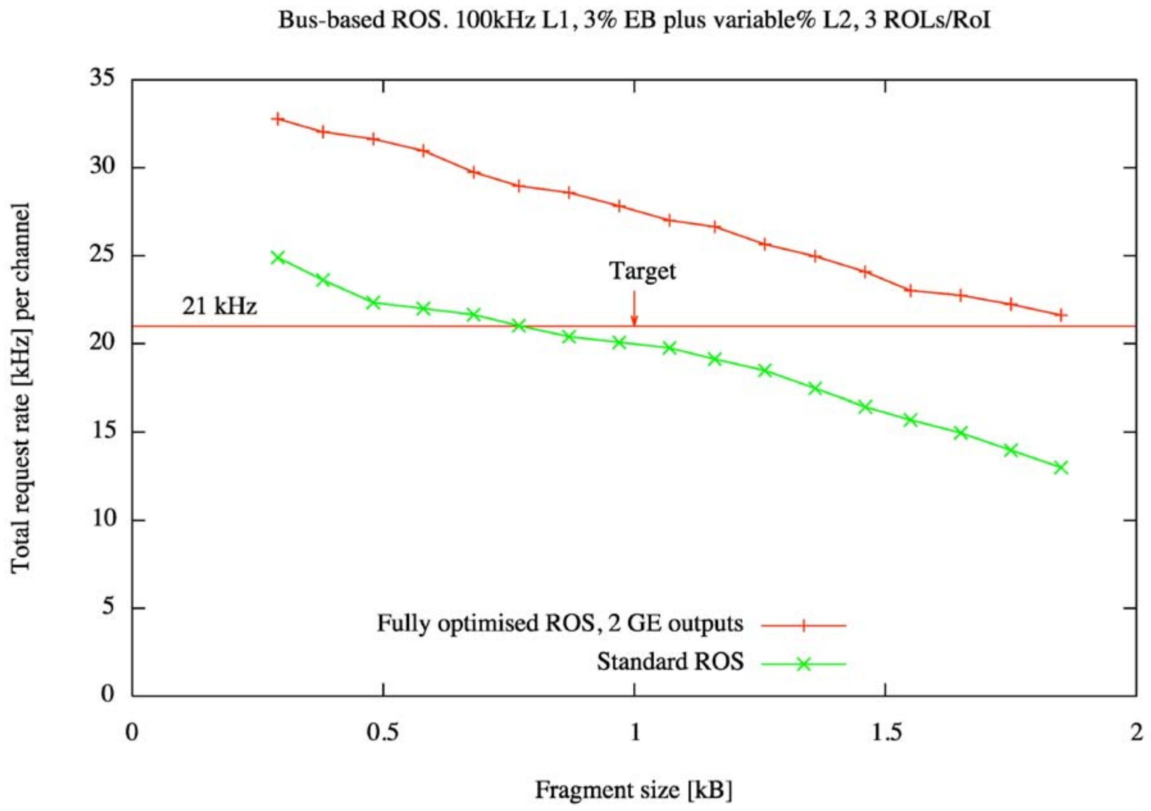


Figure 56: Performance of the bus-based ROS

6.2.2 ATLAS Dataflow

According to [ATLRDY] the full installed ATLAS dataflow system consists of 1500 computing nodes, which is a large fraction of the final size of about 2300 nodes. From the computing nodes 850 were quad-core machines with dual network interfaces, allowing to use them as both L2 and EF processing. On average, each node runs between 4 and 5 applications concurrently. For the final event building a target bandwidth of 5GB/s – generated by 59 EB applications – is envisaged. The full system was used in two configurations, one with data preloaded into the ROS-PC and running with emulated L1 triggers. Here, different L2 trigger menus are tested. The other configuration was the full operation during the first beam period between August and November 2008.

In the test with emulated input a L1-rate of 60kHz and an EB bandwidth of 3GB/s could be achieved for realistic trigger menus. The load on the ROS-PC varied between sub-detector – as expected – an was in the range of 5 to 30kHz, as shown in Figure 57.

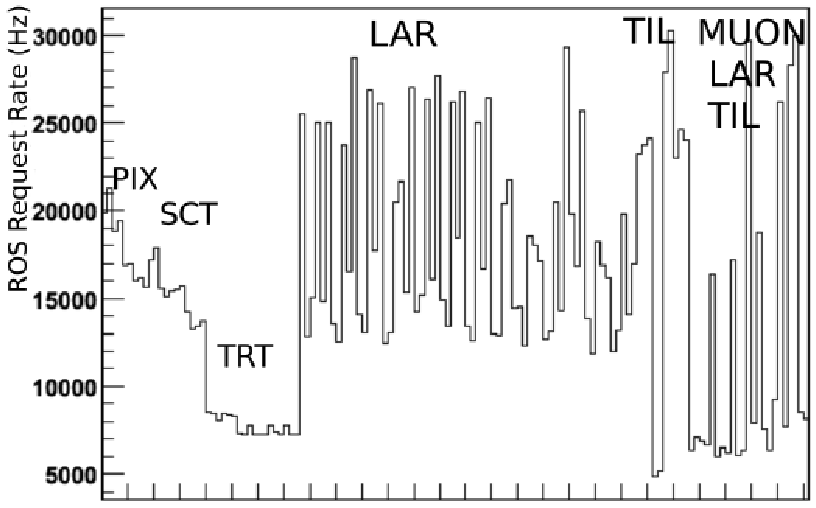


Figure 57: ROS request rate, emulated data

The analysis of the first beam period is not complete yet, however an initial result is constituted by the successful operation of the entire system during the full period, with the recording of 1PB of data.

6.3 Issues

During the design of the ROBIN hardware a few issues required re-layout or re-work. As shown in section 5.3.1.4 the layout of the DDR memory signal is critical. However, careful layout is not only required for the primary signals (address, control and data) but also for the power supply of the termination resistors. Improper decoupling and too long trances lead to spurious memory errors on the first version of the PCB.

After completion of the volume production frequent errors on the optical links were observed, after connection to the external data sources. These errors however did not occur during the optical loop-back tests, where data is transmitted locally on a ROBIN. The reason was an error during the manufacturing process of the 100MHz crystal oscillator providing the reference frequency for the TLK2501 SerDes devices. The result of this error was the locking of the oscillator frequency to the wrong frequency of the crystal after power up. This phenomenon is known as spurious mode oscillation[137] (Figure 58).

The measured frequency derivations were in the range of 500 to 1500ppm, with the specification being +/- 50ppm. The oscillator manufacturer had to supply new components, which were mounted to the boards after delivery to CERN.

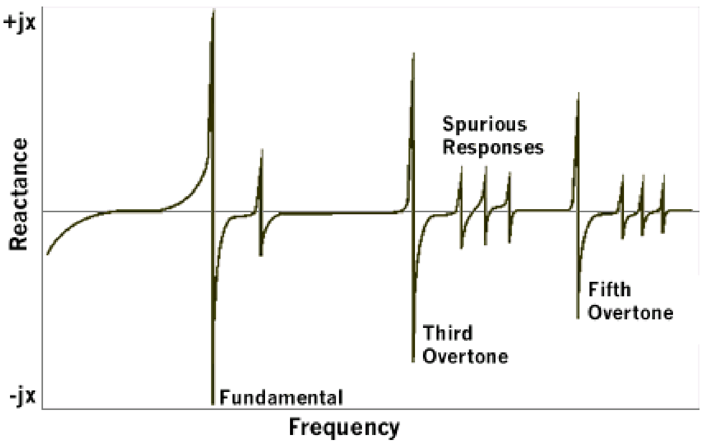


Figure 58: Spurious mode oscillation

A third issue came up with memory errors observed on the network packet buffer, occurring however only on the UK batch of the cards. The debugging process took several week and the reason was found only by accident – an incorrect assignment in the VHDL code generated a bus-contention between FPGA and memory during the addressing cycle and in turn a ground bounce effect on the address lines. A minor difference in the UK and German PCBs – power layer thickness of 18µm vs. 35µm – made the German cards tolerating the ground bounce while the UK cards didn't. The last PCB-related problem which concerns the readout of the FPGA die-temperature via the remote temperature sensor MAX1618. This device is connected to the base-emitter diode of an on-chip NPN transistor in the FPGA. The P-N junction resistance is temperature and current dependent and the resistance ratio for two different currents is a measure for the temperature. Unfortunately, a few high-speed traces of the memory system cross the two analogue signals between the diode and the sensor and the introduced noise prevents reliable measurements. The error was only detected after the production of the pre-series ROBINs, after the layout was in principle completed. The only way to avoid another re-layout was to remove the traces to the FPGA in the Gerber data-set and to replace the buffering capacitor with a diode. Although the diode is not on-chip but only in the proximity of the FPGA and its characteristics is different from the base-emitter diode of a transistor, this work-around at least allows to estimate the chip temperature. As visible from Figure 59 the transistor sensor follows the actual temperature – measured via a PT100 sensor – quite well, while the diode sensor has a significant, variable offset. With software-calibration the accuracy can be tuned to approximately 10°C, which is sufficient for the application.

An unexpected property of the external bus of the processor – which connects to the FPGA – poses a general performance limitation on the design. According to the datasheet, the external bus can provide a bandwidth of up to 266MB/s at 66MHz bus frequency. As explained above, the main use case for the communication between processor and FPGA is read and write access to FIFOs inside the FPGA,

137 Spurious mode information: “AT Crystal Spurious Modes”, http://www.ecliptek.com/tech/spurmodes.html

typically performed as a series of accesses to a fixed address. In this configuration the effective transfer speed from a FIFO to the main processor memory is only in the order of 20MB/s, because the external bus controller does not activate its burst mode. Improvements could be made by modifying the FPGA to implement dual-ported memory areas for the message data from the PCI and network interface. These memory areas are mapped into the processor's memory as cacheable regions, which allows the external bus controller to use bust-mode read access. The transfer bandwidth was improved to 55MB/s.

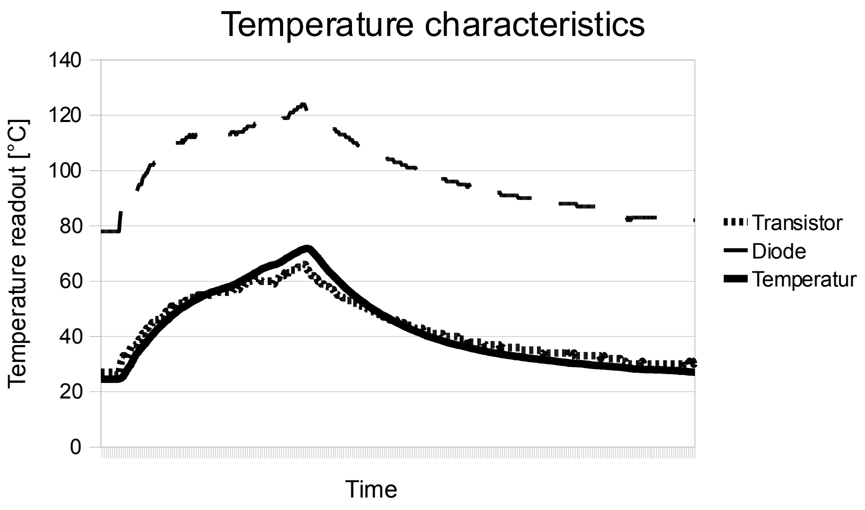


Figure 59: Temperature sensors

With a measured write bandwidth of approximately 300MB/s into main memory these numbers correspond to net read bandwidths of 22MB/s and 67MB/s respectively. For the reverse direction[138] the integrated DMA controller can be used to perform a memory-to-memory transfer into the command FIFOs. While this allows to overlap the transfer time with other processing, the total improvement is only marginal due to the additional setup required for the DMA. Finally, an attempt was made to use the DMA controller to copy the UPF information in larger chunks from the FIFOs into the main memory. Unfortunately there were spurious errors in the received UPF records which could never be completely resolved. Therefore the DMA feature is not in use for this purpose.

6.4 Reliability

The ROBIN cards are fully commissioned and in operation since more than a year and some experience concerning the reliability is available. In the first year of operation a relatively high number (approximately 50 occurrences) of hardware failures were observed. The majority of these failures could be repaired by rewriting the content of the on-board flash. There is no clear explanation for this behaviour, however after a modification of the firmware upgrade procedure the number of incidents went down significantly. So this issue is believed to be a minor software problem. From the remaining failures about 10 exhibited similar errors related to incorrect operation of the PCI interface and all of these card were from the UK production batch. During the error analysis the PLX PCI interface was

138 The single-cycle write bandwidth is only marginally higher: 28MB/s

removed on a few of these board for further investigation. The inspection of the solder pads indicated that a problem with the PCB is very likely, which is known as “black pad” phenomenon[139]. As a result certain solder pads are not properly or not all wetted by the solder. As the device is mechanically fixed onto the PCB by the correctly soldered pads an electrical connection can nevertheless exist but is not reliable. Frequently the devices pass the factory test but fail later on in the field, as it was the case with the ROBIN cards.

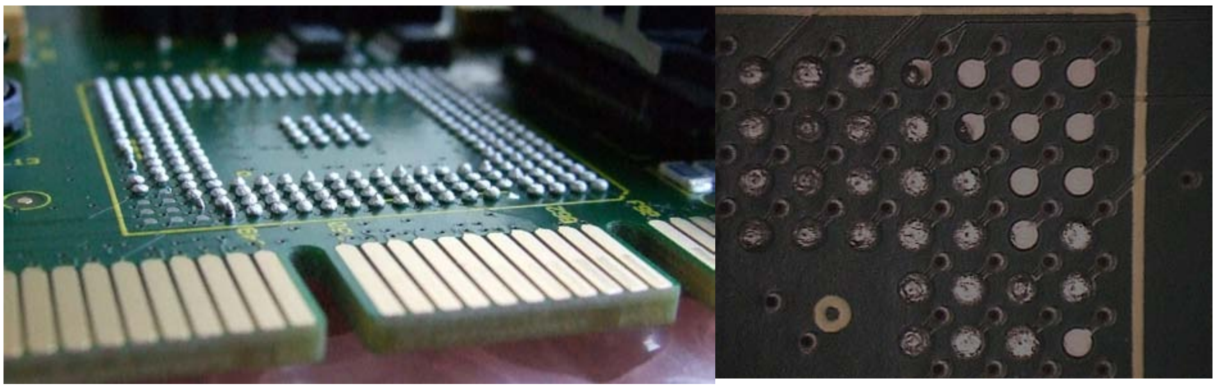


Figure 60: PLX unwetted pads

In Figure 60 it can be seen, that a few pins at the lower left corner look very different after the removal of the PLX device. A precise analysis of the situation would require a destructive analysis of the affected boards, which was not performed. About 20 cards exhibited errors which might be related to this “PLX problem”. Fortunately the error rate went down to a single defective board in 2008. Overall, the initial failure rate of the ROBINs was relatively high, but has now reached a plateau at about the same level as the ROS-PC and its other components which is below 1% per year. The high initial failure rate and the drop to a plateau is commonly described as “bathtub curve”[140].

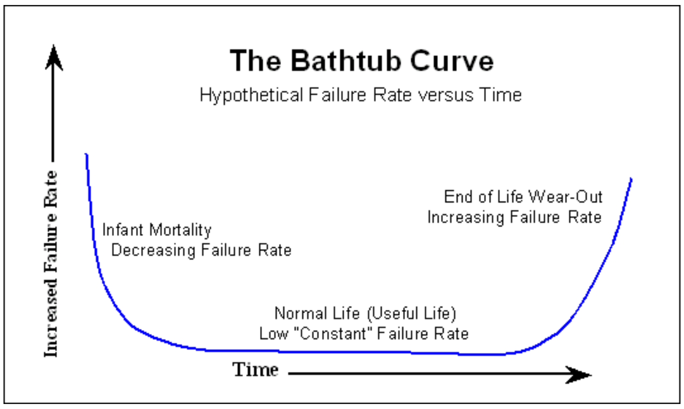


Figure 61: Bathtub curve

As the name indicates, one has to expect a rising error rate after some time of operation. To date, the system is still working at low rates of failures.

139 See e.g. http://circuitsassembly.com/cms/images/stories/pdf/0301/0301raytheon.pdf
140 See e.g. http://www.weibull.com/hotwire/issue21/hottopics21.htm

6.5 Summary

The results for the ROBIN components and for the ROS system have been presented. In stand-alone PCI operation the ROBIN can handle up to 27kHz request rate at 100kHz event rate, which is well above the requirements. The corresponding operation over the network yields only 15kHz, however there is still room for optimisations. A few issues were encountered after the volume production was finished, related to quality of layout, components and PCB. However, none of the issues was serious and the ROBINs are working very reliable since installation.

The baseline bus-based ROS meets the performance requirements for small event and RoI-sizes, which is acceptable for most of the ROS-PCs. A few ones only will be exposed to higher rates and can be tuned by reducing the number of active channels. The entire installed ATLAS dataflow system consisting of more than 1500 machines has shown stable operation and good performance during the first beam period in autumn 2008.

7 Discussion and Conclusions

The final chapter shows that the initial goals with respect to performance and functionality have been met and presents prospects for future developments based on the present implementation.

7.1 Performance Assessment

The results presented in chapter 6 verify that the performance goals of the ATLAS TDAQ system can be met. At the component level, the ROBIN card performs superior to the requirements in the baseline bus-based architecture. The network performance has already been improved since the time of the measurements presented here, by adding an alternating buffer mechanism to the network output path and by software optimisations. Latest results[141] indicate that the 21kHz request rate requirement can be met. The design paradigm to use an FPGA for high-speed and high-bandwidth functions and a standard microprocessor for the more complex but slower functions was a success. The FPGA firmware is mature and requires modifications only on relatively long time-scales (order of 6 months or more), for example after modification of the input data format. Most of the modifications during the regular maintenance and release procedures can be implemented by software updates, which is very advantageous due to the much more convenient development process compared to FPGA firmware (see also section 4.3). In the end, much more operational monitoring and error checking could be implemented on the ROBIN as initially planned. The firmware (FPGA) and software package are well integrated into the TDAQ software release framework, such that the maintenance of this custom component is well organised.

At the level of the ROS-PC two bottlenecks were identified: processor performance and memory bandwidth. Recent tests with a new type of PC motherboard using a dual-core processor and a faster memory subsystem show a significant performance boost, such that a ROS-PC with 12 active channels is only limited with respect to request rate by the performance of the ROBIN cards and with respect to output bandwidth by the number of GE links. The TDAQ system as a whole has been operated in test runs and during the first beam period with very good results and the required performance and reliability for the operation during the first few years seems to be available. Final results however can only be obtained when the LHC machine provides particle collisions at design luminosity, which is expected to start in autumn 2009. The quality of the individual sub-detectors and the physics algorithms will then define the actual rates the TDAQ system has to sustain.

To address performance requirements which cannot be fulfilled with the standard bus-based ROS, the following scenarios are envisaged:

- With the lower beam luminosity of the initial period, higher performance will be required on a few ROS-PCs of the LAr detector only. It is foreseen to lower the number of active channels on these PCs until acceptable performance is achieved. Most probably they will be equipped with two ROBIN cards.
- With increasing beam luminosity the number of affected ROS-PC might rise. At this point, the ROBINs of the relevant sub-detector will be operated in switch-based or hybrid mode. The

141 The results from the improved networking functionality will be presented elsewhere.

current view of the hybrid mode is to direct L2 requests directly to the ROBINs via the network interface, while EB requests and the delete commands will still be passed via the ROS-PC.

- Optimisations of the TDAQ software will eliminate the current duplication of requests for event building.

Overall, the ATLAS TDAQ system is ready to operate at design performance for the first LHC period up to 2012.

7.2 Prospects

Despite all these good results it is clear from sections 5.3.2 and 6.1 that the safety margins at the ROBIN are quite small - FPGA resource utilisation is very high and the processor performance is already at the limit, at least for network based operation. Adding new functionality or supporting even higher rates will be close to impossible. Also, PCI as an ubiquitous host interface is fading out and being replaced PCIe. Finally, the upgrade scenarios for LHC – phase 1 and phase 2 – will require some modifications to the TDAQ architecture.

7.2.1 Short term developments

To address the PCI/PCIe issue and to increase the component performance a variation of the ROBIN is currently under development, which supports a PCIe interface, a faster processor and a larger buffer. The general design and all[142] remaining components are copied from the existing ROBIN. The modifications to FPGA firmware and software will be minimal, such that a common set of source files can be used. The new PCIe interface is of single-lane type and supports the same bandwidth as the present PCI interface. The new processor is the pin-compatible device PPC440GX, which operates at 667MHz instead of 466MHz and provides an on-chip 2nd level cache, which the present processor does not have. The buffer memory is increased by a factor of 2, which enables to provide 64k buffer pages of 2k each guaranteeing single page fragments for all possible sizes at 100kHz L1 rate. The expectation for the performance of the PCIe ROBIN is that a request rate of 25 to 30kHz in network mode can be sustained, corresponding to 75 to 90MB/s bandwidth for 1kB fragments.

The PCIe ROBIN will be produced in a quantity of approximately 100 cards. During the regular maintenance process failing ROS-PCs or the ones at the end of their life will be replaced by PCIe capable machines, equipped with PCIe ROBINs. This will be done primarily for sub-detectors with high performance requirements. If PCI-based PCs will still be available, the current ROBINs can be re-used for the other sub-detectors.

In the area of the switch-based ROS certain improvements have already been made and will be documented elsewhere. In particular the single outbound packet buffer which posed a bandwidth limit on the tests presented in 6.1.2 was replaced by a dual-buffer mechanism, which allows to reach GE line-speed for large fragments. The network protocol was simplified and the ROBIN now aggregates data from all 3 channels into a single UDP message. Further optimisations are under investigation, for

142 Apart from the substitution of obsolete parts with recent versions.

example changing the Ethernet MTU to 4kB, which should bring the request rate for standard 1kB fragments to above 20kHz and the bandwidths close to GE line-speed already at 2kB fragments.

Finally, tests with recent motherboards supporting dual-core processors and providing better memory bandwidth have been made, using the present ROBIN in bus-based mode. The results [ROSRT09] indicate that the performance of a new ROS-PC is about twice that of the present standard ROS-PC, using the existing ROBINs.

The switch-based ROS and the gradual replacement of ROS-PC with more recent machines – with present or new ROBINs – in critical areas allow for significant performance improvements at the system level during the next few years of ATLAS operation.

7.2.2 Upgrade Phase-1

The LHC phase-1 upgrade is scheduled for the year 2012 and will include modification of the pixel sub-detector via an insertable B-layer (IBL) and in general higher detector occupancies due to higher beam luminosity. No significant changes will be applied to the detectors in general and to the readout electronics. The HLT system will be improved by a fast track trigger (FTK) which will operate on the inner detector data and provide tracking information to the L2PUs with a latency of 1ms. The dataflow system will be exposed to the same rates as today, but to event sizes larger by 50%. For the ROBINs and the ROS, this will not pose significant problems, as they are limited rather by rate than by bandwidth.

However, an interesting project is foreseen concerning the IBL development, which allows to prototype a ROB-on-ROD module, installed in parallel to the standard S-Link output from the pixel ROD. The goal hereby is to implement the ROBIN functionality for a single channel on a small mezzanine card, minimising real-estate and power consumption. The interface towards the ROD will be the S-Link connector. On the TDAQ side there will be a GE interface – supporting both electrical and optical media – for the dataflow and a second electrical GE interface for control functionality. To address the issue of the decoupling of the two subsystems TDAQ and detector/ROD the mezzanine will support power-over-Ethernet (PoE[143]), driven via the control interface. To minimise the load on the driving switches a class-1 PoE implementation with a maximum power consumption of 3.8W will be aimed for. This requires careful optimisation of the design. A potential option is to use a low power FPGA (e.g. XILINX Spartan-6) and to distribute the processor functionality onto two embedded MicroBlaze[144] soft processors, one running the fragment management and the other the message processing. The performance of the two cores will be in the order of 300 MIPS which is roughly one third of the performance of the current PPC440GP processor. Some additional advantage can be expected from the improved integration of the MicroBlaze cores with the FPGA fabric. For example, the access to the FIFO and DPR structures, which is currently a bottleneck, could be done with customised processor instructions which are available via the MicroBlaze co-processor interface. Due to the similarity of the MicroBlaze and PowerPC architectures a large fraction of the existing software sources can be re-used.

143 http://standards.ieee.org/getieee802/download/802.3af-2003.pdf
144 http://www.xilinx.com/publications/prod_mktg/MicroBlaze_Sell_Sheet.pdf

7.2.3 Upgrade Phase-2

For the phase-2 of the LHC upgrade significant modifications of the entire TDAQ architecture might be necessary in order to deal with longer L1 latencies and the increase in event sizes by a factor of 10. Currently the expectation is that the basic architecture with an L2-trigger operating on RoIs will still be valid. As a result, a new readout system will be needed, probably integrated with the detector readout (iROS). The development of the iROS will build on the experiences gained during phase-1. Two likely options are the move from S-Link to 10GbE as the output interface of the iROS and a shared memory architecture of the L1 and the ROS fragments.

However, if the RoI principle cannot be pursued for phase-2 it is likely that an architecture similar to CMS will be used, pushing all L1 accepted data via a fast network (probably 10GbE) into a HLT farm.

7.3 Summary

The ATLAS readout architecture and the ROBIN component have been presented in this thesis. The design and implementation of the ROBIN component was a success. At the component level, all performance goals are already meet or at least can be met by tuning of the architecture. System tests have demonstrated proper operation and good performance, even though the crucial test – operation on the beam with particle collisions – could not be performed yet due to the delays on the LHC machine. The expected life time of the present dataflow system is in the order of 10 years, up to the LHC upgrade phase-2. From the present design of the ROBIN a number of topics have been identified which need to be modified to achieve higher performance or to support future features of the system. Prominent examples are the PCIe version of the ROBIN which is already in the development stage, the design of a ROB-on-ROD architecture for the pending phase-1 upgrade of the pixel sub-detector and the potential integration of ROD and ROS in the course of upgrade phase-2. It is expected that many features of the present ROBIN will be migrated via technology upgrades to the new designs.

8 Appendix

8.1 Pre-ROBIN hardware

8.1.1 MPRACE-1

MPRACE-1 is a PCI based FPGA co-processor developed by the author of this work. It consists basically of a large XILINX Virtex-2 FPGA, two banks of fast SRAM, one DRAM memory slot, a commercial PCI bridge and expansion connectors (Figure 62). MPRACE-1 was extensively used in various research and educational projects, for example as frame-grabber with on-board image processing [HEZEL1], for prototyping of GE and processor modules for the ROBIN (see chapter 5.3.1.4), prototyping of a fast L2 trigger algorithm for the TRT sub-detector of ATLAS [ATLTRT] and for acceleration of astrophysical simulations [NBODY1].

MPRACE-1 was also used to prototype the PCI communication mechanism used in the bus-based ROS. Due to the very encouraging results, the implementation was re-used with little modifications on the final ROBIN. In addition, the software part of the buffer management was implemented directly on

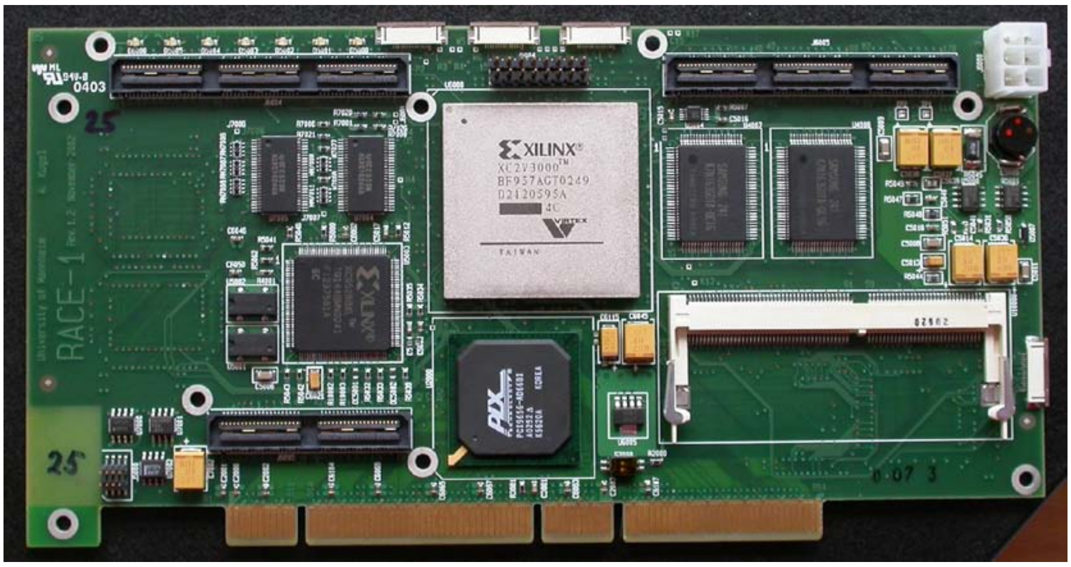


Figure 62: MPRACE-1

the FPGA in order to evaluate a processor-less ROBIN. While the raw functionality could be verified, complex monitoring and network oriented message passing mechanisms cannot be implemented with reasonable effort without processor.

8.1.2 µEnable

The early experiences at the university of Mannheim with FPGA technology led to the development of the FPGA co-processor µEnable, a PCI card with FPGA, local memory and expansion connectors. The card (Figure 63) was used in various research projects [MENABLE] and stimulated, together with the FPGA development tool CHDL [CHDL1][CHDL2], the foundation of the company SiliconSoftware[145], where the author is co-founder.

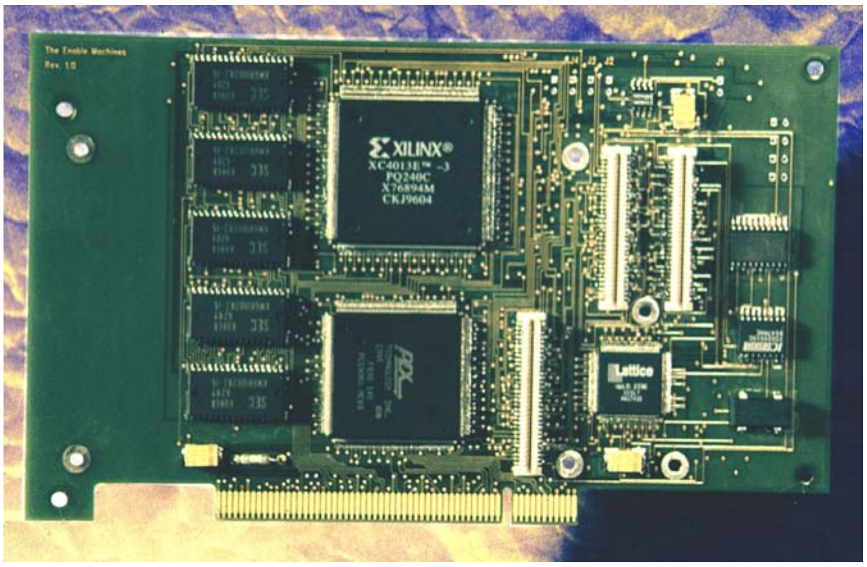


Figure 63: µEnable

The µEnable card uses a low-density FPGA of the XILINX XC4000 series and a 32 bit PCI interface and targets rather low-cost applications as compared to MPRACE-1, which provides more features at higher cost. The card was used as one of the first ROB prototypes, with a ring-buffer capable to store up to 1.000 fragments. The on-board connectors could be used with standard S-Link or PMC[146] mezzanines.

145 http://siliconsoftware.de/

146 PMC is a standard for PCI mezzanine cards according to IEEE P1386.1

8.2 ROBIN configuration parameter set

Parameter	Description	Expert
sernum	Board serial number, resides in the one-time-programmable area of the on-baord FLASH memory	Yes
BaseIPAddress	Network address, common to all channels.	No
SubDetectorId	ID of the sub-detector which is connected to the ROBIN. Value is inserted into the ROB fragment. Common to all channels.	No
Pagesize	Buffer memory granularity (page size), defaults to 2kB	No
Numpages	Number of memory pages, normally auto-calculated from buffer size and page size.	Yes
Hashbits	Number of hash-bits, defaults to 16	Yes
NetworkEnabled	Controls processing of network interface	No
Keepfrags	Deleted fragments are not actually removed from database. Used together with certain emulation modes.	No
Rolemu	Enables data emulation mode	No
TestSize	Size of emulated fragments	No
DebugEnabled	Enables debugging output via “printf” on serial terminal	Yes
DumpRolEnabled	Enable debug output of content of incoming fragments	Yes
Interactive	Enables interactive debugging via serial terminal	Yes
Divclearsize	Patch for incorrect request format in dataflow software	Yes
Keepctlwords	Enables capturing of S-Link framing information	Yes
Dpmcache	Enable caching of message memory area	Yes
RolDataGen	Emulation mode with hardware data generator	No
Macflowctl	Activate flow-control handling on network port	Yes
RolEnabled	Activate fragment processing	No
EbistEnabled	Enable extended build-in self test (BIST)	Yes
Emupattern	Data pattern in emulation mode	Yes
Continbist	Run BIST continuously	Yes
Ignorebist	Continue application even after hard BIST errors	Yes
Rolextloop	Enable link loopback test via external fibre. Defaults to chip internal loopback.	Yes
Mactxdrop	Enable dropping of network response packets if output queue full. Defaults to no dropping	Yes
Mgmtcache	Enable caching of bookkeeping memory	Yes
Dmafifocheck	Enable checking of free space in output queue. Defaults to ON	Yes
Upfdma	Use memory-to-memory DMA to read items from UPF	Yes
Hdrdma	Use memory-to-memory DMA to write DMA headers	Yes

Parameter	Description	Expert
Prescalefrag	Relative inverse priority of fragment processing in main task loop	Yes
Prescalemsg	Relative inverse priority of message processing in main task loop	Yes
Prescalefpf	Relative inverse priority of FPF processing in main task loop	Yes
UDPBasePort	UDP port number for network responses. Obsolete	No
Max1618check	Enables checking of temperature threshold	Yes
ChannelId	ID of ROBIN channel. Value is inserted into the ROB fragment	No
DcNodeId	ROBIN node id for switch-based ROS mode	No
Secsiemu	Emulate OTP sector for factory testing	Yes
GcLost	Minimum number of lost delete messages to enable garbage collection	No
GcPages	Maximum number of free buffer pages to enable garbage collection	No
MaxRxPages	Threshold for input fragment truncation	No
TempAlarm	Temperature threshold value	Yes
BofNoWait	Patch for incorrect fragment format	Yes
IrqEnable	Enable interrupts to host on error conditions	Yes
DiscardMode	Accept but do not store fragments in stop-less recovery mode	Yes
NetDeleteEnable	Enable processing of network delete messages	No

Table 7: ROBIN configuration parameters

8.3 ROBIN statistic

Statistics item	Description
ERRORS	
Hardware errors	Detection of an internal hardware error
Software errors	Detection of a software error condition (coding error)
Software warning	Detection of a condition which should not occur (e.g. unexpected code location)
Buffer full occurrences	Transitions into buffer-full state
ROL error	Errors signalled from HW page management
ROL DOWN occurencies	Transitions into link down status
ROL XOFF occurencies	Transitions into link up status
Buffer manager receive errors	Inconsistent event ids on subsequent pages of the same event
Buffer manager request errors	Inconsistent event database
Temperature warning	Temperature above threshold

Statistics item	**Description**
PCI DMA reset	PCI DMA reset due to excessive transaction delay
PCI DMA abort	PCI DMA abort after multiple resets
Interrupts	Interrupts sent to host
FRAGMENTS	
Frags received	Fragments received from link
Frags available	Fragments sent to PCI/Network
Frags not available	Fragments requested but not in database
Frags pending	Fragments unavailable but due to arrive
Frags added	Fragments added to database
Frags deleted	Fragments removed from database
Frags truncated	Fragments truncated due to oversized
Frags corrupted	Fragments with soft format or data errors
Frags rejected	Fragments with unrecoverable format errors
Frags replaced	Fragments overwritten due to duplicate event id
Frags out of sync	Mismatch of event id on subsequent fragments
Frags missing on delete	Fragments not in database during delete request
Frags TT sync error	Fragment with “sync” trigger type does not match on event id and trigger type mask
PAGES	
Pages received	Pages received from link
Pages added	Pages added to database
Pages deleted	Pages deleted from database
Pages provided	Pages sent to PCI/Network
Pages supressed	Pages discarded due to fragment truncation
Pages invalid	Pages with format error
Pages dma'ed	Pages received by fragment DMA
MESSAGES	
Messages received	Raw messages from PCI/Network
Messages accepted	Valid messages, after decoding
Messages rejected	Invalid format or request code
Messages lost	Lost messages, detected via message sequence number
Messages invalid	Invalid network format
Messages data request	Requests for data
Messages data response	Responses to data requests

Statistics item	Description
Messages clear request	Delete request messages
Messages broadcast	Ethernet broadcasts
Messages PCI TX queue full	PCI response submitted while response queue occupied. Introduces delay.
Messages NET TX queue full	Network response submitted while response queue occupied. Introduces delay.
Messages NET TX dropped	Network response dropped due to excessive delay on output queue
Messages NET RX frames OK	GE MAC statistics: received error-free ethernet frames
Messages NET RX frames FCS error	GE MAC statistics: received ethernet frames with CRC error
Messages NET RX frames length error	GE MAC statistics: received ethernet frames with incorrect length error
Messages NET TX frames OK	GE MAC statistics: error-free ethernet frames transmitted
Messages NET TX frames underrun	GE MAC statistics: outbound ethernet frames dropped due to underrun
Messages PCI Tested Empty	Check for new message did not yield request from PCI
Messages PCI Tested OK	Check for new message provided new request from PCI
Messages NET Protocol ARP	Network message for ARP
Messages NET Protocol IP	Network message using IP protocol
Messages NET Protocol RS	Network message using raw Ethernet protocol
Messages NET Protocol UDP	Network message using UDP/IP protocol
Messages NET Protocol unknown	Network message using unknown protocol
Messages NET received PAUSE frames	Incoming flow-control message on network

Table 8: ROBIN statistic record

8.4 ATLAS detector parameters

Sub-detector	Channels	ROLs	Event size [kB]
Inner detector			
Pixel	$1.4 * 10^8$	132	60
Silicon strip (SCT)	$6.2 * 10^6$	92	110
Transition radiation (TRT)	$3.7 * 10^5$	192	307
Calorimeter			
LAr calorimeter	$1.8 * 10^5$	762	576

Sub-detector	**Channels**	**ROLs**	**Event size [kB]**
Tile calorimeter	$1.0 * 10^4$	64	48
Muon system			
Monitored drift tube (MDT)	$3.7 * 10^5$	204	154
Cathode strip chamber (CSC)	$6.7 * 10^4$	16	256
L1 Trigger			
Resistive plate chamber (RPC)	$3.5 * 10^5$	32	12
Thin gap chamber (TGC)	$4.4 * 10^5$	24	6
L1 Calo	NA	48	28
Other	NA	2	0.3

Table 9: ATLAS detector parameters

8.5 Glossary

BERT	Bit error rate test
CERN	European Organisation for Nuclear Research, Geneva, Switzerland
COTS	Component of the shelf. In this context used for computers and peripherals designed for the mass market
CP	Charge/Parity
CPU	Central processing unit
CRC	Cyclic redundancy check
DAQ	Data acquisition
DPM	Same as → DPR
DPR	Dual-ported → RAM
DRAM	Dynamic → RAM
DSP	Digital signal processing
EB	Event building
ECR	Event counter reset
EF	Event filter
FE	Fast Ethernet (100Mbit/s)
FIFO	First-in first-out
FPGA	Field-programmable gate array
GE	Gigabit-Ethernet
GMAC	Gigabit-Ethernet media access controller
GUI	Graphical user interface
HDL	Hardware description language
HLT	High-level triggers
iROS	Integrated read out system
L1	First level trigger
L1ID	L1 event identifier, synonymous to event number
L2	Second level trigger
L2SUP	Second level trigger supervisor
L3	Third level trigger
LDC	Link destination card (S-Link receiver)
LHC	The **L**arge **H**adron **C**ollider, a proton-proton particle accelerator built underground at CERN with a circumference of 27km
LSC	Link source card (S-Link transmitter)
LSI	Large scale integration
LVDS	Low voltage differential signalling
MAC	Media access controller
MSI	Medium scale integration
MIPS	Million instructions per second
MTU	Maximum Transfer Unit (equivalent to Ethernet packet size)
PHY	Physical layer adapter
QCD	Quantum Chromo Dynamic, a sector of the → SM
RAM	Random access memory
RoI	Region of interest
RoIB	Region of interest builder
RoIC	Region of interest collection
ROS	Readout system
RTL	Register transfer level – a precise, low-level description of functionality
SFI	Switch to farm interface
SM	Standard model of particle physics

SRAM	Static → RAM
SUSY	Super symmetry, an extension to the → SM
TDAQ	Trigger and data-acquisition system
TRT	Transition radiation tracker
TTL	Transistor transistor logic
URD	User requirements document
VHDL	Very high speed hardware description language
ZBT	Zero bus turn-around → RAM, a variation of synchronous → SRAM

List of figures

Figure 1: Experiment Scenario....7
Figure 2: TDAQ levels....8
Figure 3: LHC at CERN*....12
Figure 4: CMS Detector [CMSJINST]....14
Figure 5: CMS DAQ [CMSJINST]....15
Figure 6: CMS FED-Builder Rails [CMSJINST]....16
Figure 7: FRL image and block diagram[CMSFRL]....17
Figure 8: ALICE dataflow [ALICEJINST]....17
Figure 9: ALICE D-RORC [DRORC]....18
Figure 10: LHCb dataflow[LHCBADD]....18
Figure 11: LHCb TELL1 [LHCBTELL]....19
Figure 12: ATLAS detector*....21
Figure 13: ATLAS TDAQ TP-version [ATLASTP]....22
Figure 14: Sequential Selection [SEQSEL]....24
Figure 15: TDR ROBIN Blockdiagram....25
Figure 16: Schematic layout of ATLAS TDAQ system [ATLASTDR]....26
Figure 17: Dataflow network [STANCU]....29
Figure 18: L2 processing times [L2PROC]....30
Figure 19: Racks with ROS-PCs*....33
Figure 20: ROS-PC....33
Figure 21: PCI load board....34
Figure 22: ROBIN TestSuite....36
Figure 23: ROS-ROBIN interaction....37
Figure 24: Virtex-2 logic element....42
Figure 25: Virtex-2 FF896 package....43
Figure 26: Simulation waveform....46
Figure 27: ChipScope waveform view....47
Figure 28: ROBIN basic hardware elements....57
Figure 29: ROBIN S-Link interface....61
Figure 30: Burst-size dependant memory bandwidth....62
Figure 31: PowerPC performance comparison....64
Figure 32: PPC440GP mezzanine....65
Figure 33: DDR memory subsystem....65
Figure 34: DDR memory simulation....66
Figure 35: PPC Xbus device-paced burst write timing....67
Figure 36: PLX local bus timing....68
Figure 37: JTAG routing....70
Figure 38: Final ROBIN....71
Figure 39: ROBIN VHDL modules....73
Figure 40: FPGA buffer manager....75
Figure 41: FPGA network interface....77
Figure 42: Software task loop....81
Figure 43: Software buffer manager....81
Figure 44: Buffer manager database....82
Figure 45: Fragment handling....83
Figure 46: PCI fragment request and response....84
Figure 47: Software execution times from profiling....85
Figure 48: Rack wizard....89
Figure 49: MTF Entry....90

Figure 50: Actual PCI stand-alone performance....95
Figure 51: Calculated PCI performance....96
Figure 52: ROBIN network performance....97
Figure 53: Effect of hash-key size....98
Figure 54: Garbage collection performance....99
Figure 55: ROS test setup....99
Figure 56: Performance of the bus-based ROS....100
Figure 57: ROS request rate, emulated data....101
Figure 58: Spurious mode oscillation....102
Figure 59: Temperature sensors....103
Figure 60: PLX unwetted pads....104
Figure 61: Bathtub curve....104
Figure 62: MPRACE-1....111
Figure 63: µEnable....112

Figures marked with *: Copyright CERN

List of tables

Table 1: Pilot Project Technologies....23
Table 2: Data-flow components (TDR)....27
Table 3: XILINX FPGA families....44
Table 4: ROBIN FPGA memory utilisation....59
Table 5: FPGA connectivity....60
Table 6: FPGA resource utilisation....78
Table 7: ROBIN configuration parameters....114
Table 8: ROBIN statistic record....116
Table 9: ATLAS detector parameters....117

Bibliography

[SMS1] CERN, Introduction to the Standard Model, CERN, 2008, http://public.web.cern.ch/Public/en/Science/StandardModel-en.html

[SMS2] Herrero, M.J., The Standard Model, Presentation at 10th NATO ASI on Techniques and Concepts of High-Energy Physics, St. Croix, USA,Jun 1998, arXiv:hep-ph/9812242v1

[DISSFLICK] Flick, T., Studies on the Optical Readout for the ATLAS Pixel Detector , Bergische Universität Wuppertal, Jul 2006, urn:nbn:de:hbz:468-20060600

[SUSY] Peskin, M., Supersymmetry in Elementary Particle Physics, SLAC,SLAC-PUB-13079, Jan 2008, http://arxiv.org/abs/0801.1928v1

[CPV] Peskin, M., The Matter with antimatter, SLAC,Nature 419,24 - 27, Sep 2002, http://dx.doi.org/doi:10.1038/419024a

[QCD] Hands, S., The Phase Diagram of QCD, Contemp.Phys.42:209-225,2001. May 2001, http://arxiv.org/abs/physics/0105022v1

[LHCJINST] Evans, L. et al., LHC Machine, Journal of Instrumentation,JINST 3 S08001, Aug 2008, http://dx.doi.org/10.1088/1748-0221/3/08/S08001

[ATLJINST] The ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider, Journal of Instrumentation,JINST 3 S08003, Aug 2008, http://dx.doi.org/10.1088/1748-0221/3/08/S08003

[ALICEJINST] The ALICE Collaboration, The ALICE experiment at the CERN LHC, Journal of Instrumentation,JINST 3 S08002, Aug 2008, http://dx.doi.org/10.1088/1748-0221/3/08/S08002

[CMSJINST] CMS Collaboration, The CMS experiment at the CERN LHC, Journal of Instrumentation,JINST 3 S08004, Aug 2008, http://dx.doi.org/10.1088/1748-0221/3/08/S08004

[LHCBJINST] LHCb Collaboration, The LHCb Detector at the LHC, Journal of Instrumentation,JINST 3 S08005, Aug 2008, http://dx.doi.org/10.1088/1748-0221/3/08/S08005

[DJFROB] D. Francis et al., The Read-Out Buffer in DAQ/EF Prototype -1, CERN,ATL-DAQ-2000-053, Aug 2000, http://doc.cern.ch//archive/electronic/cern/others/atlnot/Note/daq/daq-2000-053.pdf.

[LHCBDAQ] Alessio, F. et al., LHCb Online event processing and filtering, Journal of Physics,Conf. Ser. 119 022003, 2008, http://dx.doi.org/10.1088/1742-6596/119/2/022003

[LHCBTB] The LHCb Collaboration, LHCb Technical Proposal, CERN, Feb 1998, http://lhcb-tp.web.cern.ch/lhcb%2Dtp/postscript/tp.ps

[CMSTRIG] Afaq, A. et al., The CMS High Level Trigger System, IEEE transactions on nuclear science,Vol 55 Issue 1, Feb 2008, http://dx.doi.org/10.1109/TNS.2007.910980

[SLINK64] Racz, A. et al, The S-LINK 64 bit extension specification: S-LINK64, CERN, Aug 2003, http://cmsdoc.cern.ch/cms/TRIDAS/horizontal/docs/slink64.pdf

[CMSSFB] Bauer, G. et al., The Terabit/s Super-Fragment Builder and Trigger Throttling System for the Compact Muon Solenoid Experiment at CERN, CERN,CERN-CMS-CR-2007-020, May 2007, http://cdsweb.cern.ch/record/1046342/files/CR2007_020.pdf

[CMSGBE] Bauer, G. et al., CMS DAQ Event Builder Based on Gigabit Ethernet, IEEE Transactions on Nuclear Science,Vol 55, Issue 1, Feb 2008, http://dx.doi.org/10.1109/TNS.2007.914036

[CMSFRL] Arcidiacono, R., Flexible custom designs for CMS DAQ, Proceedings of the 10th Topical

Seminar on Innovative Particle and Radiation Detectors,Vol 172,174 - 177, Oct 2007, http://dx.doi.org/10.1016/j.nuclphysbps.2007.08.106

[DRORC] Carena, F. et al., The ALICE Data-Acquisition Read-out Receiver card, CERN,Proc. 10th Workshop on Electronics for LHC and Future Experiments,273ff, Boston, USA,Sep 2004, http://doc.cern.ch//archive/cernrep/2004/2004-010/p273.pdf

[ALICETDR] ALICE collaboration, Trigger, Data Acquisition, High Level Trigger, Control System Technical Design Report, CERN, 2004, https://edms.cern.ch/file/456354/2/DAQ_Chapters7-10.pdf

[LHCBADD] Tatsuja, N. et al., Addendum to the LHCb Online System Technical Design Report, CERN,CERN-LHCC-2005-039, 2005, http://cdsweb.cern.ch/record/903611/files/lhcc-2005-039.pdf?version=2

[LHCBTELL] Bay, A. et al., The LHCb DAQ interface board TELL1, Nuclear Instruments and Methods in Physics Research Section A, Vol 560/2,494ff, May 2006, http://dx.doi.org/10.1016/j.nima.2005.12.212

[SLINK] Boyle, O., The S-LINK Interface Specification, CERN, Mar 1997, http://hsi.web.cern.ch/HSI/s-link/spec/spec/

[ATLASTP] The ATLAS collaboration, ATLAS High-Level Triggers, DAQ and DCS: Technical Proposal, CERN, Mar 2000, http://cdsweb.cern.ch/record/434668/files/cer-2184259.pdf

[ATLDEMPROG] Blair, R. et al., OPTIONS FOR THE ATLAS LEVEL-2 TRIGGER, CERN,OPEN-99-149, Feb 1997, http://cdsweb.cern.ch/record/398762/files/open-99-149.ps.gz

[SEQSEL] Bystricky, J., A sequential processing strategy for the ATLAS event selection, IEEE Transactions on Nuclear Science,Vol 44 Issue 3,342 - 347, Jun 1997, http://dx.doi.org/10.1109/23.603668

[PAPMOD] Dobson, M. et al., Paper Models of the ATLAS Second Level Trigger, CERN,ATL-DAQ-98-113, Jun 1998, http://cdsweb.cern.ch/record/683664/files/daq-98-113.pdf

[PILPRO] Blair, R. et al., The ATLAS Level-2 Trigger Pilot Project, IEEE transactions on nuclear science,Vol 49 Issue 3,851 - 857, Jun 2002, http://dx.doi.org/10.1109/TNS.2002.1039577

[ROBCPLX] Cranfield, R., Vemeulen, J., Options for the ROB Complex, CERN,ATL-DAQ-2000-027, Apr 2000, http://cdsweb.cern.ch/record/684047/files/daq-2000-027.pdf

[AROBC] Boeck, R. et al., The active ROB complex, CERN,ATL-DAQ-2000-022, Mar 2000, http://cdsweb.cern.ch/record/683960/files/daq-2000-022.pdf

[ATLASTDR] Jenni, P. et al., ATLAS high-level trigger, data-acquisition and controls : Technical Design Report, CERN, Jul 2003, http://cdsweb.cern.ch/record/616089/files/cer-002375189.pdf

[STANCU] Stancu, S. N., Networks for the ATLAS LHC Detector : Requirements, Design and Validation, CERN,CERN-THESIS-2005-054, Jul 2005, http://cdsweb.cern.ch/record/913894/files/thesis-2005-054.pdf

[GETB] Matei Ciobotaru, Stefan Stancu, Micheal LeVine, and Brian Mart, GETB, a GigabitEthernet Application Platform: its Use in the ATLAS TDAQ Network, IEEE transaction on nuclear science,Vol 53 Issue 3,817 - 825, Jun 2006, http://dx.doi.org/10.1109/TNS.2006.873303

[ETHERT] Barnes, F.R.M. et al., Testing ethernet networks for the ATLAS data collection system, IEEE transactions on nuclear science,Vol 49 Issue 2,516 - 520, Apr 2002, http://dx.doi.org/10.1109/TNS.2002.1003791

[BASEDF] Beck, H.-P. et al., The Base-Line DataFlow System of the ATLAS Trigger and DAQ, IEEE transactions on nuclear science,Vol 51 Issue 3,470 - 475, Jun 2004, http://dx.doi.org/10.1109/TNS.2004.828707

[LARSC] Burckhart-Chromek, D. et al., Testing on a Large Scale : running the ATLAS Data Acquisition and High Level Trigger Software on 700 PC Nodes, Proceedings of 15th International Conference on Computing In High Energy and Nuclear Physics,60 - 65, Mumbai, India,Mar2006, http://cdsweb.cern.ch/record/941077

[ETHER] S. Stancu, R.W. Dobinson, M. Ciobotaru, K. Korcyl, and E. Knezo, The use of Ethernet in the Dataflow of the ATLAS Trigger and DAQ, CERN,Proceedings of Computing in High Energy and Nuclear Physics, La Jolla, CA, USA,Mar 2003, http://arxiv.org/pdf/cs.ni/0305064

[DFNET] Stancu, S. et al., ATLAS TDAQ DataFlow Network Architecture Analysis and Upgrade Proposal, Proceedings of NPSS Real Time Conference,Vol 53 Issue 3,826 - 833, Jun 2005, http://dx.doi.org/10.1109/TNS.2006.873302

[DFROS] Vermeulen, J.C. et al., ATLAS DataFlow : the Read-Out Subsystem, Results from Trigger and Data-Acquisition System Testbed Studies and from Modeling, Proceedings of IEEE NPSS Real Time Conference,Vol 10 Issue 10, Jun 2005, http://dx.doi.org/10.1109/RTC.2005.1547446

[L2PROC] Abolins, M. et al., Integration of the Trigger and Data Acquisition Systems in ATLAS, Journal of Physics,Conf. Ser. 119 (2008) 022001, 2008, http://dx.doi.org/10.1088/1742-6596/119/2/022001

[MADROS] Müller, M., ROS Architecture and Requirements, CERN, May 2004, http://agenda.cern.ch/fullAgenda.php?ida=a041681

[ATLDCS] Bariusso Poy, A. et al., The detector control system of the ATLAS experiment, Journal of Instrumentation,JINST 3 P05006, May 2008, http://dx.doi.org/10.1088/1748-0221/3/05/P05006

[ROSURD] Cranfield, R., LeVine, M., McLaren, R., ROS User Requirements Document, CERN,ATL-DQ-ES-0007, May 2004, https://edms.cern.ch/file/356336/1.0.2/ros_urd_v102.pdf

[ROBROD] Beck, H.P., Hauser, R., LeVine, M., Impact of a ROB on ROD Design on Atlas DataCollection, CERN,ATL-DQ-ER-0002 , Nov 2001, https://edms.cern.ch/file/391562/0.5/DC-027.pdf

[ROBRODC] Beck, H.P. et al., Commissioning with a ROB-on-ROD Based Readout, CERN,ATL-DQ-ON-0001 , Mar 2003, https://edms.cern.ch/file/374790/0.1/RoRcommissioning_v01.pdf

[XLNXDS31] XILINX, XILINX Virtex-2 Datasheet, XILINX,Vendor Datasheet, May 2005, http://www.xilinx.com/support/documentation/data_sheets/ds031.pdf

[OSI] ISO, Open systems interconnect reference model, ISO,ISO/IEC 7498-1, Jun 1996, http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994%28E %29.zip

[ETH802.3] IEEE, IEEE 802.3, IEEE, Dec 2005, http://standards.ieee.org/getieee802/802.3.html

[ROBSUM] ATLAS ROS Group, ROBIN Summary, CERN, Feb 2002, http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/ROS/documents/ROBINsummary.pdf

[HLDDPROT] Green, B., Kugel, A., Kieft, G., Prototype-RobIn HLDD, CERN, Sep 2002, https://edms.cern.ch/file/356324/2.4/hldd.pdf

[DLDDPROT] Green, B., Kugel, A., Kieft, G., Protoype-RobIn DLDD, CERN, Sep 2002, https://edms.cern.ch/file/356328/2.3/dldd.pdf

[SWIDPROT] Green, B., Kugel, A., Kieft, G., Protoype-RobIn Software Interface, CERN, Sep 2002, https://edms.cern.ch/file/356332/2.2/swid.pdf

[FDRPROT] Farthouat, P., Report of the Final Design Review ROBIN Prototype, CERN, Oct 2002, https://edms.cern.ch/file/359014/1/robin-pdr.pdf

[ROBRDD] Green, B. et al., ROBIN Design Document, CERN, May 2004, https://edms.cern.ch/file/473396/1.0/rdd.pdf

[ROBFDR] Farthouat, P., Report of the Final Design Review ROBIN, CERN, Aug 2004, https://edms.cern.ch/file/478897/2/Robin-fdr-may2004.pdf

[ROBPRR] Farthouat, P., PRR of the ROBIN, CERN, Mar 2005, https://edms.cern.ch/file/571432/1/Robin-prr-mar2005.pdf

[PRRDD] Green, B. et al., ROBIN Production Readiness Review: Design Document, CERN, Jan 2005, https://edms.cern.ch/file/555100/1/prr_design_050221.pdf

[PRRPM] Green, B. et al, , CERN, Jan 2005, https://edms.cern.ch/file/555103/1/prr_performance_050221.pdf

[PRRTP] Green, B. et al., ROBIN Production Readiness Review: Test Procedures, CERN, Jan 2005, https://edms.cern.ch/file/555102/1/prr_testing_050221.pdf

[PRRPS] Green, B. et al, ROBIN Production Readiness Review: Production Schedule, CERN, Jan 2005, https://edms.cern.ch/file/555104/1/prr_procurement_050222.pdf

[ROBSPARE] Kieft, G. et al., ROBin Spares Policy, CERN, Mar 2006, https://edms.cern.ch/file/714623/1.3/Robin_Spares_v1-3.pdf

[SWROB] Iwanski, W. et al., The software ROBIN, CERN, Feb 2002, http://indico.cern.ch/getFile.py/access?resId=1&materialId=0&contribId=s1t14&sessionId=s1&subContId=0&confId=a0281

[MMROB] Müller, M., Evaluation of an FPGA and PCI Bus based Readout Buffer for the Atlas Experiment, Mannheim University, May 2005, http://madoc.bib.uni-mannheim.de/madoc/volltexte/2005/1070/

[GBELSC] Blair, R. et al., A Gigabit Ethernet Link Source Card, Argonne national laboratory, 2002, http://lhc-electronics-workshop.web.cern.ch/LHC-electronics-workshop/2002/DAQ/B33.pdf

[ROBROD] Beck, H.P., Hauser, R., LeVine, M., Impact of a ROB on ROD Design on Atlas DataCollection, CERN,ATL-DQ-ER-0002 Nov 2001, https://edms.cern.ch/file/391562/0.5/DC-027.pdf

[POE] IEEE, IEEE Std. 802.3af - IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements, IEEE,Standard for Information technology 802.3af, 2003, http://ieeexplore.ieee.org/servlet/opac?punumber=8612

[VME] IEEE Computer Society, IEEE Standard for A Versatile Backplane Bus: VMEbus, IEEE,ANSI/IEEE Std 1014-1987, 1987, http://dx.doi.org/10.1109/IEEESTD.1987.101857

[UKROB] Boorman, G. et al., The UK ROB-in, a prototype ATLAS readout buffer input module, CERN,ATL-DAQ-2000-013, Mar 2000, http://cdsweb.cern.ch/record/684041/files/daq-2000-013.pdf

[ROBMAN] Kugel, A. et al., ATLAS ROBIN User Manual, CERN,ATL-DQ-ON-0018, Apr 2006, https://edms.cern.ch/file/719553/1/robinUserManual.pdf

[DC022] Beck, H.-P. et al., The Message Format for the ATLAS TDAQ DataCollection, CERN,ATL-DQ-ES-0035 , Oct 2008, https://edms.cern.ch/document/391557/2.5

[RAWFMT] dos Anjos, A. et al., The raw event format in the ATLAS Trigger & DAQ, CERN,ATL-D-ES-0019 , Feb 2009, https://edms.cern.ch/document/445840/4.0c

[ATLCOM] Morettini, P. et al., ATLAS Detector Commissioning, CERN,CERN-ATL-SLIDE-2008-178, Nov 2008, http://cdsweb.cern.ch/record/1140714/files/ATL-SLIDE-2008-178.pdf

[ROBJINST] Cranfield, R. et al., The ATLAS ROBIN, Journal of Instrumentation,JINST 3 T01002, Jan 2008, http://dx.doi.org/10.1088/1748-0221/3/01/T01002

[TIPP09] Crone, G. et al., The ATLAS ReadOut System - performance with first data and perspective for the future, Acc. for publication in proceedings of The 1st international conference on Technology and Instrumentation in Particle Physics, Tsukuba, Japan,Mar 12-172009,

[ATLRDY] Vandelli, W. et al., Readiness of the ATLAS Trigger and Data Acquisition system for thefirst LHC beams, 11th Topical Seminar on Innovative Particle and Radiation Detectors,ATL-COM-DAQ-2009-005 Siena, Italy,Oct 01-04 2008, http://cdsweb.cern.ch/record/1155455/files/ATL-COM-DAQ-2009-005.pdf

[ROSRT09] Della Volpe, D. et al., The ATLAS DataFlow System: Present Implementation, Performance and Future Evolution, Subm. for presentation at RealTime Conference 2009, Bejing, China,May 2009,

[HEZEL1] Hezel, S., FPGA-basiertes Template-Matching mit Distanztransformierten Bildern, Mannheim University, Jul 2004, http://madoc.bib.uni-mannheim.de/madoc/volltexte/2004/338/

[ATLTRT] Hinkelbein, C. et al, Using of FPGA Coprocessor for Improving the Execution Speed of the Pattern Recognition Algorithm for ATLAS – High Energy Physics Experiment, Lecture Notes in Computer Science, Vol 3203/2004,791-800, Aug 2004, http://www.springerlink.com/content/kj9hg110eadf2vjd/

[NBODY1] Spurzem, R et al., From Newton to Einstein – N -Body Dynamics in Galactic Nuclei and SPH using new special hardware and Astrogrid-D, Journal of Physics,Conf. Ser. 78 012071 2007, http://dx.doi.org/10.1088/1742-6596/78/1/012071

[MENABLE] Brosch, O. et al, MICROENABLE - A RECONFIGURABLE FPGA COPROCESSOR, CERN,Proc. 4th Worksh. on Electronics for LHC Experiments,402ff Rome, Italy,1998,

[CHDL1] Kornmesser, K. et al., The FPGA Development System CHDL, IEEE,Proc. of the 9th IEEE FCCM conference,271 - 272, Napa, CA,Apr 2001, ISBN: 0-7695-2667-5

[CHDL2] Kornmesser, K., The FPGA Development System CHDL, Mannheim University, Dec 2004, urn:nbn:de:bsz:180-madoc-8575

Acknowledgements

I wish to express my gratitude to all the people who helped me during this work, first of all to my supervisor Prof. Männer, who supported me with patience and advice in many respects and stayed confident in my skills to complete this thesis.

I thank my spiritual mentor Ursa Paul, who encouraged me and helped me to stay focused on this task.

My wife and my children helped me to keep my spirit up, in particular in the last period when putting all things together. Thanks to you!

This work would not have been possible without the collaboration with and contributions from former and present colleagues from my institute and from the other ROS groups - my thank goes to David Francis for getting this project on the way and to Benedetto Gorini for the good guidance of the overall ROS activity; to Jos Vermeulen for looking deeply into models and measurement results; to Markus Joos and Louis Tremblet for their help during all CERN on-site activities; to Barry Green and Gerard Kieft, Matthias Müller and Erich Krause for their friendly and efficient cooperation during the hardware and firmware developments; to Andrzej Misiejuk, Stefan Stancu and Hans-Peter Beck for their help setting the networking stuff up and to Nicolai Schroer, Per Werner and Kostas Kordas for running so many tests with it.

I also express my thanks to all the others who I didn't mention by name but who helped with ideas, hints and discussions or just with caring about the progress of this work.