UNIVERSIDAD
POLITECNICA
DE VALENCIA

Study and Design of the Readout Unit Module for the LHCb Experiment

José Francisco Toledo Alarcón

Under the direction of Dr. Francisco José Mora Más

Doctoral Thesis

Electronics Engineering Department

Universidad Politécnica de Valencia, 2001

To Hans, with gratitude.

I couldn’t have got this far

without your experience, support and leading.

List of acronyms

CPLD	*Complex Programmable Logic Device*
DAQ	*Data Acquisition*
DB	*Data Block*
DPM	*Dual-Port Memory*
DYB	*Directory Block*
EBI	*Event-Building Interface*
ECS	*Experiment Control System*
FCFS	*First Come, First Served*
FEE	*Front-End Electronics*
FEL	*Front-End Link*
FEM	*Front-End Multiplexer*
FLIC	*Flexible Input/Output Card*
FPGA	*Field-Programmable Gate Array*
GAL	*Gate Array Logic, a kind of programmable device*
GPIO	*General Purpose Input/Output*
HEP	*High-Energy Physics*
HDL	*Hardware Description Language*
IC	*Integrated Circuit*
L1	*Level-1 Trigger*
MCU	*Monitoring and Control Unit*
NIC	*Network Interface Controller*
ODE	*Off-Detector Electronics*

PLD	*Programmable Logic Device*
PMC	*PCI Mezzanine Card*
RN	*Readout Network*
RU	*Readout Unit*
SEB	*Sub-Event Buffer*
SEM	*Sub-Event Merging*
SFC	*Sub-Farm Controller*
STF	*Sub-event Transport Format*
TTC	*Trigger and Timing Control*
VELO	*Vertex Locator*

Preface

"A beginning is the time for taking the most delicate care that the balances are correct"

From the novel "Dune" by Frank Herbert.

Introduction

This engineering doctoral thesis has been carried out at the Electronics Design group, Experimental Physics division, of the European Laboratory for Nuclear Research (CERN) in Geneva, Switzerland. The CERN Doctoral Student programme gave me the opportunity to obtain a CERN doctoral student grant (from July 1998 until September 2001) and thus have the privilege to work closely with members of the Electronics Design group (CERN EP/ED) and the LHCb Collaboration at CERN.

This international research organization is building its new generation of high-energy physics (HEP) experiments around its new TeV-range proton synchrotron: the LHC (Large Hadron Collider), an upgrade to the existing LEP (Large Electron-Positron) collider. One of the four approved experiments at the LHC is a single-arm spectrometer designed to study the CP[1] violation in the decay of the B meson: the LHCb experiment. Its purpose is to gain a better understanding of symmetry violations in Nature and help to answer questions such as why is there more matter than anti-matter in our universe[2]. Another way to search for new physics beyond the Standard Model in LHCb is the study of rare or forbidden B-meson decays.

Data Acquisition (DAQ) and Trigger systems in the four LEP experiments (OPAL, L3, ALEPH and DELPHI) were designed in the eighties to handle data rates in the order of megabytes per

1. CP symmetry, C (particle-antiparticle interchange) and P (space inversion). The product CP was a good symmetry until found to be violated in K^0 decays in 1964.
2. A less romantic description of the LHCb experiment's goal is to measure, with more accuracy than ever, the coefficients of a three-by-three matrix (namely the CKM matrix) that describes the B meson decay.

second, whilst LHC experiments (ATLAS, ALICE, CMS and LHCb) will have to deal with gigabyte-per-second data flows. What makes such an enormous difference?. Firstly, it must be taken into account that proton colliders like LHC provide “dirty” events (i.e., the interesting interactions are surrounded by a large background of non-interesting phenomena) when compared to electron-positron colliders. This translates into larger events and more complex event filtering, as more refined algorithms are required.

Secondly, the high luminosity[1] of LHC (one or two orders of magnitude higher than previous high-energy colliders, including LEP) will cause multiple interactions per bunch crossing (pileup), which will also increase event sizes and will make event filtering and reconstruction more difficult. Thirdly, high bunch crossing rates (40 MHz) and million-data-channel granularity will produce higher data throughputs.

As a consequence, the LHC experiments will require unprecedented challenging, cutting-edge electronics for detector readout, event filtering, event building and storage. The Fastbus- and VME-based tree architectures of the eighties run out of steam when applied to LHC’s DAQ requirements. New concepts and architectures change rack-mounting backplane buses for high-speed point-to-point links, abandon the centralized event building and use switched networks and parallel architectures instead, and replace mainframes and workstations with cheaper PC farms.

Following these trends, and in the context of the LHCb data acquisition (DAQ) and trigger electronics, this doctoral thesis aims at investigating the feasibility, studying the architectures and implementations, developing and testing a crucial electronics module for the LHCb experiment: the **Readout Unit** (RU).

Objectives

HEP experiments in general, and LHCb DAQ and trigger systems [LHCC98-4] in particular, share the need for readout modules that accept incoming event fragments from several inputs and assemble them into larger sub-events, outputting to a single higher-bandwidth link. Thus, data is processed on-board and routed via I/O links in the front- and back-panel, rather than the traditional approach of data processing and routing across a crate backplane. This basic functionality called sub-event building is recurrent in HEP experiments and usually requires custom design for high-throughput applications, due to the lack of commercial systems that comply with the specific requirements. Besides the sub-event building functionality, the readout modules must provide enough buffering to compensate for fluctuations provoked by a number of causes such as flow control and error handling.

A functional block diagram of such a readout module is shown in the following figure. Three main functional blocks can be identified:

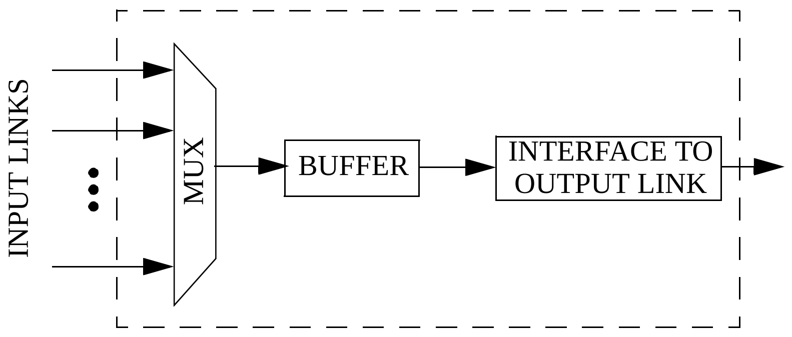
1. LHC will also produce, besides proton-proton collisions, nucleus-nucleus interactions for heavy ion experiments like ALICE. In this case, both luminosity and bunch crossing rate will be lower than in other LHC experiments. Large event sizes will compensate for these low figures, resulting also in challenging trigger and data acquisition systems. LHCb will exploit proton-proton interactions only.

1. An **input stage** which merges incoming event fragments sharing the same event number.
2. An intermediate **sub-event buffer** in which these event fragments are stored according to a defined format. Input and output stages are isolated by this buffer that operates logically as a dual-port memory.
3. An **output stage** that reformats fragments in the sub-event buffer into a single sub-event which is sent to the next stage according to defined protocols and data formats.

Three modules requiring sub-event building and buffering have been identified in LHCb, and a common approach to the three of them has been carried out under the name of the Readout Unit Project for the LHCb experiment. These modules are:

1. Front-end Multiplexer module (FEM) for the off-detector electronics [LHCb-98-069].
2. Readout Unit (RU) for the DAQ system [RUnote].
3. Readout Unit for the Level-1 Vertex Locator (VELO) trigger [Schulz01].

Although the basic functionality is the same in all three applications, there are important differences in terms of throughput (40 to 240 MByte/s), trigger rate (40 kHz to 1 MHz) and sub-event building protocols. Thus, the RU must be a flexible and reconfigurable general-purpose module to be used as an entry stage to a DAQ or trigger system. Other functionalities, like an interface to the Experiment Control System are required in the LHCb applications.



Basic functionality of a Readout Unit module.

The objectives of this thesis are:

1. **Investigate the feasibility of the Readout Unit for the LHCb experiment and define its architecture and algorithms in the context of the LHCb DAQ and trigger systems**.

   Modules with the above mentioned functionalities have been designed for HEP experiments (either as custom-made or commercial products) since the eighties, most of them based on processors. There is nowadays, though, a trend towards programmable-logic based designs justified by higher throughput requirements, as posed by LHC experiments, leading to new solutions and architectures like the ones presented in this thesis.

   The feasibility of an architectures is ultimately determined by the state of the art in the basic building blocks (like FIFO and dual-port memories, FPGA, network and link technologies, etc.), and so architectures become quickly obsolete as new technologies emerge to replace the “older” ones. This inherent obsolence, clearly a limitation for upgrading and maintenance during the life span of the module (around fifteen years), poses the twofold requirement of using state-of-the-art architectures, technologies and components that provide the required performance and, on the other hand, that will remain in the market for a long-enough period.

2. **Design and implement the Readout Unit**.

   Working with state-of-the-art technology usually means dealing with non-stable and incomplete documentation and tools, unknown bugs and errata in data sheets and integrated circuits. Specific issues related to new technologies must be usually sorted out and evaluated before solutions are brought to market. All these problems must be solved in order to come to a RU module implementation. As a result, design becomes an important issue in this work.

3. **Validate the design through laboratory tests and measurements**.

   In-system tests cannot be carried out, as the previous and next stages in the data flow chain have not been implemented so far. Thus, validation must be done, for the time being, through laboratory tests using pattern generators for Level-1 data emulation and logic and PCI analyzers to capture the output link data.

4. **Participate in the definition of the LHCb DAQ and trigger link technologies, data formats and protocols**.

   Most of the algorithms, protocols and data formats affecting the Readout Unit were undefined at the time the Readout Unit project started. Moreover, the upstream and downstream[1] components (Level-1 electronics, DAQ and Level-1 trigger readout networks) and link technologies were not designed either. Flexible solutions must be found to evaluate different link technologies until a final decision is taken and to allow upgrading during the lifetime of the experiment. The chosen data formats must be characterized by low framing overhead and easy error logging, tracing and detection.

The feasibility of the module has been investigated an several architectures have been studied, leading to the construction of two different prototypes. Work has also been carried out in the definition of the link technologies, data formats and protocols between the Front-end Electronics and the RUs and between the RUs and the destination (application dependent). The proposed interface technologies and data formats were accepted by the LHCb Collaboration. All these achievements are described in this thesis.

Thesis structure

This thesis is divided into seven chapters.

An historical review of the technologies and architectures used in DAQ and trigger systems in HEP experiments is presented in **chapter one**. In an evolution characterized by a continuous increase in data rates and event sizes, the DAQ electronics shifted from the 70´s CAMAC-based systems to the 80´s and 90´s Fastbus and VME systems. Some innovations were introduced in the 90's, like switched networks for event building, PC-based computing farms and high-speed point-to-point links for inter-module and inter-crate communications. In a recent trend, VME is

1. Upstream is towards the detector.

replaced by PCI. The Flexible I/O concept is exposed and our implementation, the PCI-FLIC, is presented.

Chapter two gives an insight into the LHCb DAQ and Trigger systems and allows to place the work in its context. High trigger rates (40 kHz) and event sizes in the order of 100 KByte produce about 4 GByte/s of data through the DAQ system. For comparison, ALEPH (a large LEP experiment) produces also 100 KByte events through its DAQ system, but at a 10 Hz rate. Not less challenging, the four-level Trigger system reduces the initial 40 MHz bunch crossing rate down to 200 Hz event rate to permanent storage.

One characteristic of LHCb is the indirect flow control. Instead of using duplex data links, as other LHC experiments do, data sources push data into the destinations via simplex links. The destinations must be always ready to accept data. This results in cheaper links and simpler protocols. On the other hand, buffer overflow must be avoided. Both requirements can be met by implementing a “throttle output” in all destination modules which is asserted when buffers get full beyond a certain threshold. These signals are centrally monitored by a supervisor unit that inhibits the Level-1 trigger whenever there is a risk that any buffer in the system overflows. The flow control logic is then a critical sub-system with the Readout Unit as one of its components.

The Readout Unit applications, requirements, architecture and performance studies are covered in **chapter three**. With applications ranging from 40 to 240 MByte/s, event rates from 40 kHz to 1 MHz and the need to support a wide range of link technologies, reconfigurability, flexibility, scalability and performance are the key parameters that determine the architecture. Different architectures are analyzed and the module feasibility (in terms of performance and buffer size) is studied.

The RU I/O technologies proposed in this work are presented (CERN S-Link for data input and slow output, and PCI as interface to commercial network interface cards for high bandwidth output), together with our proposed Sub-event Transport Format (STF) for data transport across the DAQ system.

The architectural studies presented in the previous chapter led to a first prototype implementation described in **chapter four**. This prototype is a 9U Fastbus board with FPGA-based input and output stages, dual-port-memory sub-event buffer and plug-in I/O and MCU mezzanine cards. FPGAs and dual-port memory provide high performance, mezzanine cards allow flexibility to test different technologies and the Fastbus form factor provides the advantages described in section 4.1. The backplane is only used for power, routing all the I/O via data links in the front- and back-panel. Intended as a demonstrator unit and an evaluation platform for critical components and technologies like the PCI bus, the MCU and the FPGAs, its design and manufacturing turned out to be a invaluable training and experience to tackle the design of its successor: the Readout Unit II.

The Readout Unit II, described in **chapter five**, is intended as a final module design. It relies on a two parallel 32-bit data-path concept rather than on the 64-bit architecture of its predecessor. It implements a number of simplifications and improvements like support for the Level-1 VELO application, larger sub-event size, and a new MCU in a standard PMC form factor, removing the dependence on a single-company product.

Only the most relevant implementation issues are presented in chapters four and five for the sake of clarity, hiding the effort implied in the design of such complex boards.

The RU-II design and FPGA code have been validated by laboratory tests described in **chapter six**. S-Link input to S-Link output tests have been carried out (see section 6.2) to demonstrate the input stage sub-event merging (valid for all three applications) and sub-event building (FEM application only) functionalities as well as the correct behavior of the hardware, thus validating most of the design. In order to complete this validation, the PCI subsystem has been tested (section 6.3) for both the MCU-based monitoring and control and the PCI-based sub-event building functionalities. The latter validates also the DAQ and VELO application sub-event building concepts.

This chapter is followed by the **conclusions**, which summarize the work carried out, the scientific publications derived from it, and points at future lines of work.

The **first appendix** (Application Development) gives an insight into the development methodology and tools for the Readout Unit's FPGAs and MCU.

This thesis describes a novel solution for a large HEP experiment. For the sake of completeness, a solution for a small experiment based on the PCI-FLIC card is presented in the **second appendix**.

Contributions by other engineers and physicists

The Readout Unit Project started in July, 1998, when I joined the EP/ED group at CERN. At that point, the RU was just a black box outlined in the LHCb Technical proposal document and sketched in a few LHCb technical notes. Three permanent members (Dr. Hans Müller, project leader and supervisor at CERN for this thesis work, François Bal, electronics design engineer working part-time in the project in data-link related topics, and me, doctoral student responsible for the RU implementation studies and design), with the reinforcement of other students whenever possible, formed the Readout Unit Team. This team worked closely with other members of the LHCb Collaboration in the framework of the large and multi-national project which is LHCb.

As a consequence, some of the work done in the project and mentioned in this document has been carried out by other physicists and engineers and this thesis should not be credited with it. In particular:

1. The Level-1 VELO readout network (2-D SCI torus architecture with TagNet scheduling bus) described in section 3.1.3 and the TagNet command format in section 5.5.1 are part of the work done by V. Lindendstruth, M. Schulz and A. Walsch from the Kirchhoff-Institute für Physik, Heidelberg in the context of the Level-1 trigger implementation project.
2. The sub-event building protocol for the DAQ application described in section 3.5.2 was originally proposed by B. Jost and N. Neufeld, from the CERN LHCb Group.
3. The MCU programming for the first Readout Unit was done by B. Bruder, a French cooperant who worked at CERN EP/ED. He should also share in a 50% the credit for the design of the MCU for the Readout Unit II (section 5.8).

4. The final version of the input stage FPGA code for the Readout Unit II was developed by D. Domínguez, technical student at CERN EP/ED. He also spent many hours at the laboratory testing the Readout Unit modules and co-wrote the *flasher* utility to program the FPGAs from PCI (section I.II).
5. The STF pattern generator card (section 6.2) and the S-Link rx./tx. card for the Readout Unit test station (section 6.3) were designed by F. Bal, design engineer at CERN EP/ED.
6. A. Guirao, from CERN EP/ED, tested and debugged the MCU for the Readout Unit II and developed the output stage FPGA code for the tests presented in section 6.2.
7. The *flasher* utility to program the FPGAs from PCI, the foundation for the rest of the PCI programming in section I.II and the PCI-FLIC driver for Linux are part of the work done by A. David, doctoral student at CERN for the NA-60 experiment.
8. The LHCb DAQ and trigger system architecture were defined by the LHCb collaboration [LHCC98-4].
9. The NA-60 experiment, briefly described in appendix II, is not part of this thesis. The sole relation with this thesis is the fact that I designed the PCI-FLIC card, backbone of the NA-60 DAQ system.

CHAPTER 1

Data acquisition in high-energy physics experiments

"All DAQ systems are by definition beyond the state of the art. No one designs a new one when an old system can be adapted or upgraded to do the job"

Peter S. Cooper, Fermi National Accelerator Laboratory.

1.1. Particle accelerators

High energy physics (HEP) study phenomena occurring at very high energies, creating conditions under which matter behaves in non usual ways and then new Nature laws manifest. HEP experiments consist normally in colliding accelerated particles with other accelerated particles or with fixed targets. Collision produces complicated reactions which are detected, stored and studied in order to find these new Nature laws.

The first HEP experiment [Fraser97] took place at the Rutherford's laboratory in 1932 and consisted on hitting litium nuclei with protons which had been previously accelerated in a high-vacuum tube. The voltage difference across the tube gave enough energy to the protons to break litium nuclei: a breakthrough in the study of the nuclear structure. But it was clear that new instruments were needed in order to reach higher energies in the proton beams. This need gave birth to the first particle accelerators.

The particle accelerator history started in the early thirties when Ernest Orlando Lawrence at Berkeley turned his attention into the acceleration of charged particles by magnetic fields. His idea was to make particles spiral by the effect of magnetic fields, gaining energy each time they passed through the voltage gap of an oscillator. The practical implementation of this idea was the **cyclotron**. His first demonstration model reached only 80 keV, but before the end of the decade Lawrence's cyclotrons progressed up to 19 MeV. The first applications of these machines were the production of radioactive isotopes and plutonium and the separation of uranium isotopes during the Second World War. As beam energies (i.e., particle speed) become higher and higher, the applied magnetic field had to be shaped to compensate for relativity. Magnetic field modulation allowed cyclotrons (now called **synchrocyclotrons**) to go up to 190 MeV in 1946.

Using quite a different approach, the advances in klystrons during the Second World War allowed the construction of relatively-small **linear accelerators**, with the Stanford Linear Accelerator Center (SLAC) in California leading the developments in this technology. Starting in 1947 with a 6 MeV machine, Stanford reached 900 MeV in 1957.

Still in wartimes, the idea of having a beam of charged particles circulating in a ring and behaving as the secondary of a transformer gave birth to the **betatron** in 1940. It was used as a portable x-ray machine in hospitals and for bomb detection applications. The concept was extended by Veksler in Russia and McMillan at Berkeley, introducing the idea of ‘phase stability’ in which particles were packed in bunches that are accelerated by synchronized pulses. This new machine was christened **synchroton**. In 1953, a synchroton in Brookhaven reached 3.3 GeV and was christened **cosmotron**, as it was the first machine to achieve cosmic-ray energies.

A further improvement to the synchroton was the ‘strong focusing’, in which magnets are alternately facing inwards and outwards and not only outwards as it had been done previously. This modification increased the focusing power and thus allowed the beam to be squeezed in a smaller beam pipe with the benefit of a reduced magnet cost (as magnets enclose the beam pipe). At the end of the fifties, the world’s highest-energy machine was CERN’s Proton Synchroton (PS) which reached 25 GeV. New terms were coined for the new synchrotons in the seventies, with energies in the order of several hundreds of GeV: **supersynchroton** and **tevatron**.

The next step in the evolution of particle accelerators towards higher energies was to collide particle beams together. This way, with no kinetic energy loses in a fixed target, more energy is available for particle-to-particle interactions. This could be achieved either by colliding two beams circulating in two different rings that met tangentially or, more efficiently in terms of costs, by accelerating in the same ring two beams with particles of opposite charge. In 1971, the ISR (Intersection Storage Ring) at CERN collided two 31 GeV proton beams, which was equivalent to a 2 TeV proton beam colliding a fixed target. LEP (Large Electron-Positron collider) at CERN, SLC (Stanford Linear Collider) at SLAC and PETRA at DESY (Hamburg, Germany) are three modern examples of electron-positron **beam colliders**. HERA at DESY is an example of electron-proton collider, SPS at CERN is a proton-antiproton one. These experiments took place in the eighties and nineties.

Last (so far) in the dynasty of large colliders, the LHC at CERN will be a proton[1] synchroton beam collider conceived for a TeV energy range. Superconducting magnets to produce the high magnetic fields needed to bend the beams and higher luminosity[2] are just two of the technical challenges that LHC designers and builders have to face.

1. Synchroton radiation losses at LHC energies in an electron beam would be too high, so protons (which have a much larger mass) are used in LHC.
2. Required to compensate for the resulting reduced cross section at high energies.

1.2. Particle detectors

During the fifties, synchrocyclotrons, synchrotons, linear accelerators and cosmic rays[1] were the main sources of high-energy particles. Particle beams smashed onto fixed targets producing a spread of particles that had to be detected. The instrument that had been used so far was the **cloud chamber**[2], also called **Wilson chamber**. A chamber was filled with a gas, in fact, a mixture of vapor in equilibrium with liquid and a non-condensing gas. This mixture was brought into a supersaturated state by expansion. Condensation started around the ions generated by passing charged particles, and the resulting drops were photographed. Already in the early fifties it was clear that this instrument was not able to work properly in a new environment characterized by:

- High energy particles that could not be easily stopped by gases or vapors.
- Interaction rates too high for the chamber's expansion time.

In order to alleviate these limitations, the **bubble chamber** was invented in the early fifties. The sudden reduction in pressure in a liquid (hydrogen, propane or freon) made it boil following the track left by a particle. Liquids could stop high energy tracks that gases could not, and so the bubble chamber was better suited for the new particle accelerators. A good example is CERN's GARGAMEL bubble chamber, which worked from 1971 until 1984. Thousands of photographs had to be taken in experiments until a rare (and thus interesting) event was captured. It was not unusual that HEP institutes claimed a new discovery with the sole evidence of one or two photographs.

But the future of particle detectors was clearly in electronics. The antiproton was the first particle discovered by electronic means in the late fifties. In late sixties, Charpak at CERN invented the **multiwire chamber** [Charpak76]. It was conceived as a detector for charged particles consisting of thin parallel and equally spaced anode wires sandwiched between two cathode planes. Cathodes have a negative voltage whilst anodes are grounded, creating an electric field. Particles passing through the detector ionize the gas in the chamber and the liberated electrons, which are collected by the anodes, are accelerated as the electric field increases close to the anodes, producing further gas ionization and resulting in an avalanche of electrons reaching the anode. The multiwire chamber provided tracking information with a spatial resolution of less than 1 mm as well as momentum information (indicated by the pulse amplitude in the anode wire). The multiwire chamber could be read out by a computer, with all the implied benefits (on-line data analysis, detector read out at kilohertz rates, possibility to trigger on specific events, measurement automatization, etc.).

The multiwire chamber was replaced by more sophisticated variants like the **drift chamber** [Charpak74], in which the spatial resolution is enhanced by combining the wire coordinates with temporal information (the drift time, i.e., the time it takes to the electrons to drift from the place where the high-energy particle ionized an atom to the nearest sense wire). All kinds of chambers share the drawback of having long drift times (up to 100 μs) that limit the frequency of operation to a few kilohertz.

1. From the fifties on, as the machines were able to produce energies beyond the GeV, cosmic-ray experiments lost protagonism. Nowadays, cosmic rays are still used in the absence of particle beams for test purposes.
2. By the end of the forties, another technique had been introduced: blocks of photographic emulsion in which traversing particles left tracks.

Other types of detectors used nowadays which require an electronic readout system are:

- **Scintillators**. There is a large number of materials that can be used to build scintillators, though plastic is the most common one. Scintillators utilize the ionization produced by charged particles to generate optical photons. PMTs (photo multiplier tubes) transform the photon's energy into an electrical signal. With only 10 ns dead time and 150 ps time resolution, these are one of the fastest detectors.
- **Silicon detectors**: Semiconductors have been used as particle detectors since the fifties, exploiting the low ionization energy of silicon (3.6 eV per e-hole pair). The basic structure for silicon strip detectors [Charles99] consists of a sandwich structure of a N-type bulk between P+ and a N+ implants. The structure is reverse biased (20 to 30 V) to produce complete depletion in order to allow charge drift, which is collected in metallic contacts at the implants. Long and thin parallel implants (strips) provide spatial resolution better then 50 µm and multimegahertz readout speed. Other kinds of silicon detector, the silicon drift detector and the pixel detector, are also used nowadays. Silicon drift detectors enhance spatial resolution at the expense of speed by taking into account the charge drift time, whilst pixel detectors provide high-granularity two-dimensional spatial information.

  Calorimeters: Gaseous, Cherenkov and silicon detectors work on charged particles. Calorimetry is the only practicable way to measure neutral particles. Calorimeter detectors use total absorption of particles to measure their energy and position. In the process of absorption, particle *showers* are generated by cascades of interactions. In the course of showering, most of the incident particle energy will be converted into heat, which explains the name calorimeter for this kind of detector. No temperature is measured but characteristic interactions with matter (e.g. atomic excitation, ionization) are used to generate a detectable effect via particle charges. Calorimeters can also provide signatures for particles that are not absorbed: muons and neutrinos. Muons do not shower in matter, but their charge leaves an ionization signal, which can be identified in a calorimeter. Neutrinos, on the other hand, leave no signal in a calorimeter, but their existence can sometimes be inferred from energy conservation.

  Typically, incident electromagnetic particles like electrons are fully absorbed in the **electromagnetic calorimeter**. Incident hadrons[1], on the other hand, may start their showering in the electromagnetic calorimeter, but will be fully absorbed in later layers, i.e. in the **hadronic calorimeter**, built precisely for their containment. Discrimination, often at the trigger level, between electromagnetic and hadronic showers is a major criterion for a calorimeter.

1.2.1. Particle detector readout

The typical analog readout front-end for most detectors is shown in Figure 1-1 [Groom00]. The detector is represented by a capacitance C_d. The detector is biased through a resistor R_b which has a certain capacitance to ground (C_b). The charge in the detector is coupled to the preamplifier through a blocking capacitor (C_c) and an equivalent series resistor R_s.

Typically, the detector output is a pulse with a fast rising edge but a long tail which has to be cancelled in order to allow fast detector operation. The preamplifier provides gain and feeds a

1. Particles are divided into two main families: hadrons (which take part in strong force interactions) and leptons (which do not interact with the strong force, like electrons, muons and neutrinos). Hadrons are divided into mesons (light mass particles, like B and K mesons) and barions (heavy mass particles, like neutrons and protons). Nevertheless, there are mesons heavier than barions.

pulse shaper that optimizes the signal-to-noise ratio while performs tail cancellation to limit the pulse length, though not all applications require a pulse shaper [Simoes01]. Noise is an important issue when working with the low signal levels involved in detector readout and it is normally expressed in charge units.

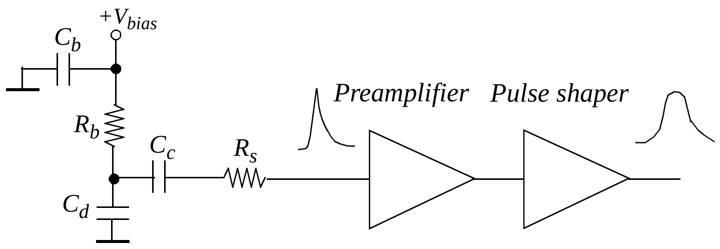


Fig. 1-1. Typical detector readout analog front-end circuit.

Today’s detectors can have up to some hundred thousand channels and thus the analog front-end is integrated in ASICs that accept more than one hundred input channels each. In a novel trend in detector readout, the ASIC integrates a number of A/D converters, digital filters for pulse shaping [Mota00], zero suppression, data formatting logic and an interface to transmit the data to the following downstream component (off-detector electronics), normally some meters away. In a different approach, data is transmitted in analog form to the off-detector electronics and is digitized in a later stage (as an example, see “Front-end electronics overview” on page 30 for a description of the LHCb front-end electronics system).

The data from all sub-detectors must be gathered into a single data unit called **event** which is written into permanent storage for later analysis. This is the mission of the DAQ system in a HEP experiment. Events have to be filtered, as not all events are relevant for the experiment’s purposes and also the permanent storage bandwidth is limited. Filtering is the task carried out by the trigger system. Next section describes the evolution of DAQ systems in HEP experiments during the last decades.

1.3. Evolution of DAQ systems

During the past four decades HEP experiments have become progressively more complex and larger. The search for new physics and rare phenomena with low cross section demanded progressively higher bunch crossing rates and luminosities in order to keep acceptable event rates[1] and thus collect the required number of events during the lifetime of the experiment. Higher beam energy was also required as experiments targeted particles and interactions that required higher energies. These needs stimulated advances in a number of engineering fields like electronics, microwaves, superconductivity, magnet design, vacuum and cryogenics, making possible the construction of modern accelerators.

The combination of high luminosity and low-cross-section interesting events implies that these interesting events are masked by a large background of non-interesting but higher-cross-section interactions. From the data acquisition and trigger systems point of view this implies a number of problems:

- **Event reconstruction** becomes more complex, as interesting tracks and particles are but a small fraction of the observed ones.
- High luminosity can cause **multiple interactions** per bunch crossing, complicating trigger algorithms and difficulting event reconstruction.
- **Channel occupancy** in the sub-detectors (understood as the percentage of bunch crossings in which a channel carries data) increases and thus the DAQ system throughput.
- **Radiation** levels in the vicinity of the detectors increase, posing radiation-hard or radiation-tolerant requirements to the front-end electronics.

But luminosity and cross section are not the only basic parameter to take into consideration to study the evolution of HEP experiments from the DAQ system point of view: the ever-growing bunch crossing rate and detector channel count (the former to keep a sufficient event rate and the latter to increase spatial resolution by using finer-grained detectors) have to be taken into account to understand today's DAQ system complexity.

The initial scenario in the late sixties, in which a few hundred data channels had to be read at a low rate (a few hertzs), allowed a single mini computer to carry out the readout. Nowadays, large experiments have millions of channels that have to be readout at megahertz rates. Distributed processing, complex trigger systems, switch networks and PC-based computer farms characterize today's DAQ systems. The following sections describe how DAQ systems evolved during the last four decades.

1.3.1. Instrumentation buses for HEP in the 60's and in the 70's

In a typical HEP experiment in the seventies and late sixties, the front-end electronics were read out by a single minicomputer (the first ones arrived to market in the mid-sixties). This architecture (Figure 1-2) lacked of parallelism and allowed data rates in the order of kilobytes per second.

Signals coming from multiwire chamber anodes were connected to amplifiers. Scintillators were connected to counters. Amplifier and TAC (time-to-amplitude converter) outputs were digitized by ADCs. ADC and coincidence circuits sent pulses to counters which were read out by a minicomputer via a non-standard bus. The front-end electronics used non-standard interconnects and modules too, generating confusion and inefficiency.

This situation led to an effort of standardization in bused modular data acquisition systems. As most of the electronics were at the front end rather than at the readout, the standardization effort focused first on the front-end area, leading to the release of the **NIM** system standard [NIMstd]

1. The luminosity (L) defines the intensity of colliding beam machines and is proportional to the bunch interaction frequency, to the square of the number of particles in the bunch and is inversely proportional to the bunch section in the direction perpendicular to the beam. It is expressed in $cm^{-2}s^{-1}$. Together with the cross section of the interaction (σ) and the detector acceptance (A), the luminosity allows to calculate the number of events (N) during a certain amount of time: $N = A \cdot \sigma \cdot \int L dt$. The average rate of detected events is simply $A \cdot L \cdot \sigma$. The acceptance is the average detection efficiency, or the probability of detecting an event if it has taken place. The cross section (expressed in cm^2 or in barn, $10^{-24}cm^2$) is a measure of the probability of interaction of two particles. Thus, interesting events with a small cross section will require experiments with high luminosity and bunch crossing rate in order to provide the required number of events during the life of the experiment.

in 1964, by the AEC Committee on Nuclear Instrument Modules (NIM) in the USA. This standard defines the connector and module mechanics, signal levels and power supplies, but does not define a backplane bus which could be used for readout. The NIM standard found widespread application in all areas of nuclear research and is still alive today despite its age.

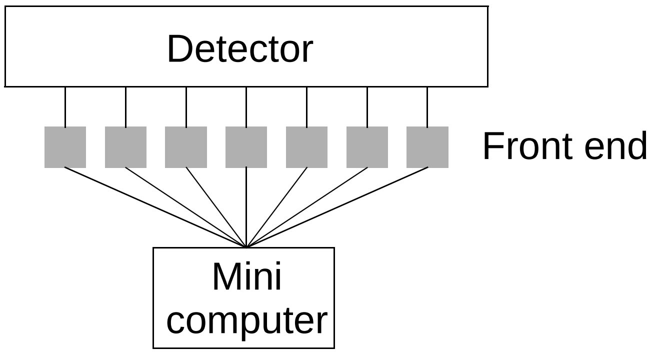


Fig. 1-2. DAQ architecture in the seventies: NIM-based front end readout via CAMAC by a minicomputer.

The standardization effort moved later into the computer bus arena leading to the publication in 1969 of the CAMAC (Computer Automated Measurement And Control) specification [CAMACstd] by ESONE (European Studies on Norms for Electronics). CAMAC defines a modular computer-controlled bus. In its simplest architecture, a crate controller controls the modules residing in its crate (crates have up to 25 slots), though multi-crate systems can also be built.

The modularity of CAMAC is exploited in applications such as the DAS CAMAC at TRIUMF [Tam89], dating from the late eighties (see Figure 1-3).

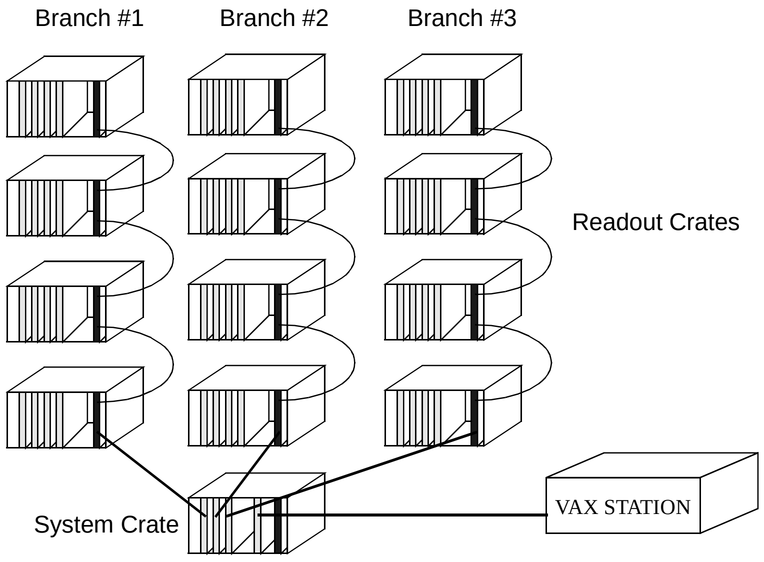


Fig. 1-3. TRIUMF’s DAS CAMAC architecture example.

In this application, a system crate containing an executive crate controller module controls up to seven branches interfaced via branch coupler modules. Each branch allows up to seven readout crates interconnected in a daisy-chain fashion. Each of these readout crates houses a crate controller and several data collecting modules. A VAX computer interfaced to the system crate controls the whole system via defined commands. The command´s address is specified by branch number, crate number, slot number and function number. The weak point of this architecture is the inherent lack of parallelism that limits the use of this system to low rate applications.

HEP experiments in the seventies used NIM-based front-end electronics modules readout by a minicomputer using CAMAC as readout bus. But CAMAC did not fulfill the requirements of a new scenario in which:

1. An increase in bandwidth requirements as the result of an increase in the number of readout channels (when multiwire chambers and drift chambers started being used in the seventies) highlighted CAMAC bandwidth limitations.
2. The advent of the microprocessor added parallelism to the new readout architectures by the possibility to use multiple processors, but CAMAC had been defined for a single host in the system and thus readout architectures were centralized.

Nevertheless, this centralized architecture is still used in small DAQ systems like the CAMAC systems described in [Morhac95] and [Watzlavik92] though sometimes in hybrid VME-Fastbus-CAMAC systems like in [Erven92].

1.3.2. DAQ and trigger systems in the 80's

The following trends can be observed during the eighties:

- The increase in detector resolution increased the number of channels. The study of rare events made it necessary to increase the event rate to achieve the required statistics. The conjunction of these two parameters provoked and increase in bandwidth requirements through the DAQ system to several MByte/s, requirement that could be fulfilled with the use of Fastbus and VME as the DAQ backbone and the **parallelism achieved with distributed computing**.
- Permanent storage bandwidth was in the order of 100 KByte/s [Cattaneo97], thus requiring a **data rate reduction by means of a trigger system** to filter out non-interesting events.
- The above mentioned increase in sub-detector complexity justified the **DAQ partitioning** into several DAQ systems which could work autonomously during development and commissioning phases and for calibration and test runs during operation. All the DAQ partitions (which grouped all channels from a sub-detector) could of course work together in a single DAQ system.

By 1976, it was judged that the time had come for a new bus standard that, retaining the strong points of CAMAC, was based on new electronics technology [Pointing91]. Both NIM and ESONE worked on the development of a new bus and joined forces, resulting in the release of the first Fastbus specification in 1983 [Fastbus83] which evolved into the definitive specification in 1986 [Fastbus86]. Strong points of Fastbus are its high bandwidth and support for multi-segment and multi-host architecture (exactly what CAMAC was lacking).

The first microprocessors came to market in the seventies. But they did not have an external bus that satisfied the needs of the HEP applications. It was the advent of 16-bit devices like the Motorola 68000 which triggered a massive use of microprocessors in the HEP arena. The

Motorola 68000 was quite well suited for bus-oriented applications and thus widely used, leading to the definition of the VME (VERSA Module Europe bus) standard [VMEstd]: an adaptation of the Motorola's proprietary VERSAbus from 1981 to the Eurocard mechanics. One weak point of VME was the lack of inter-crate communication in the original specification, solved at the end of the decade with the release of the VICbus (VME Inter-Crate bus) specification [Parkman91]. The VME specification dates from 1987, though VME has been used since 1982 [Parkman94].

Even with Fastbus and VME as high-bandwidth readout buses and NIM as front-end interconnection standard, CAMAC still found a niche. The already existing modules, developments and know-how in CAMAC could not be neglected, leading to a situation in which NIM, CAMAC, Fastbus and VME coexisted quite frequently, with a clear trend towards the end of the decade to use VME as the modular computing element, whilst NIM, CAMAC and Fastbus were relegated to the front end [Parkman94].

Typical DAQ architectures had a hierarchical tree-like structure (see Figure 1-4) with computing nodes at the branch intersections, profiting from the VME and Fastbus multi-master feature that allowed distributed computing. The tree-like structure eased the creation of DAQ partitions and allowed data rates in the order of megabytes per second.

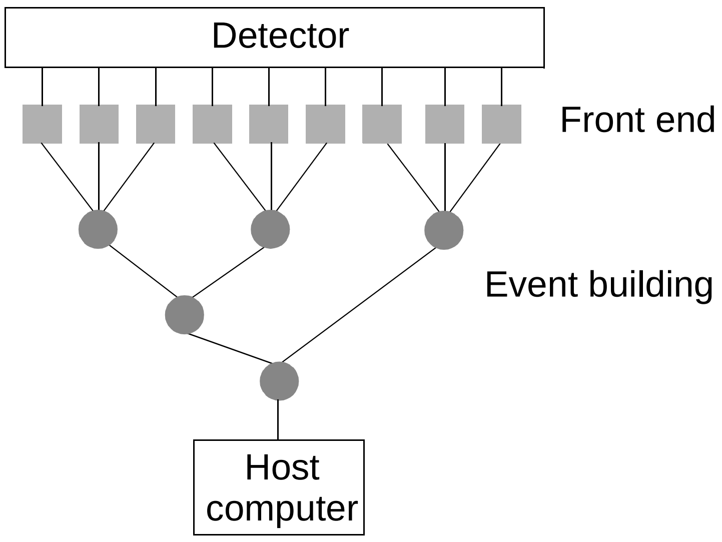


Fig. 1-4. DAQ in the eighties: tree-like distributed architectures, DAQ partitioning and trigger systems.

A good example can be found in the CERN's ALEPH DAQ System [Rüden89] [Cattaneo97] based on Fastbus (see Figure 1-5). The experiment was approved in 1982 and started taking data in 1989. Readout Controllers (ROC) and Event Builders (EB) are based on Motorola 68020 processor boards. The two modules are very similar with the only difference of additional fast synchronization circuitry in the EB boards.

ROCs and EBs are masters towards the detector and slaves towards the Main Readout Computer. The hierarchy is strengthened by the fact that ROCs and EBs do not communicate with other nodes in their same hierarchical level. ROCs read out the sub-detectors upon reception of a trigger accept signal, format and zero-suppress the data. EBs build a sub-event at the level of each sub-detector, whilst the Main EB builds a complete event (around 100 KByte in size) and sends it via an optical link to the Main Readout Computer. So, no event filtering is done at the ROC and EB level.

ALEPH was designed with a three-level trigger system. The first two levels were implemented in hardware and reduced the rate from 50 kHz to 500 Hz (Level-1) and down to 10 Hz (Level-2) which is the DAQ rate. Level-3 reduction takes place in the Main Readout Computer and lowers the rate to 1 Hz; thus matching the bandwidth to the tape storage capabilities (in the order of 100 KByte/s).

A similar architecture for the DELPHI experiment is described in [Adam92]. Smaller scale examples of tree architecture can be found in [Geesaman89] [Essel92].

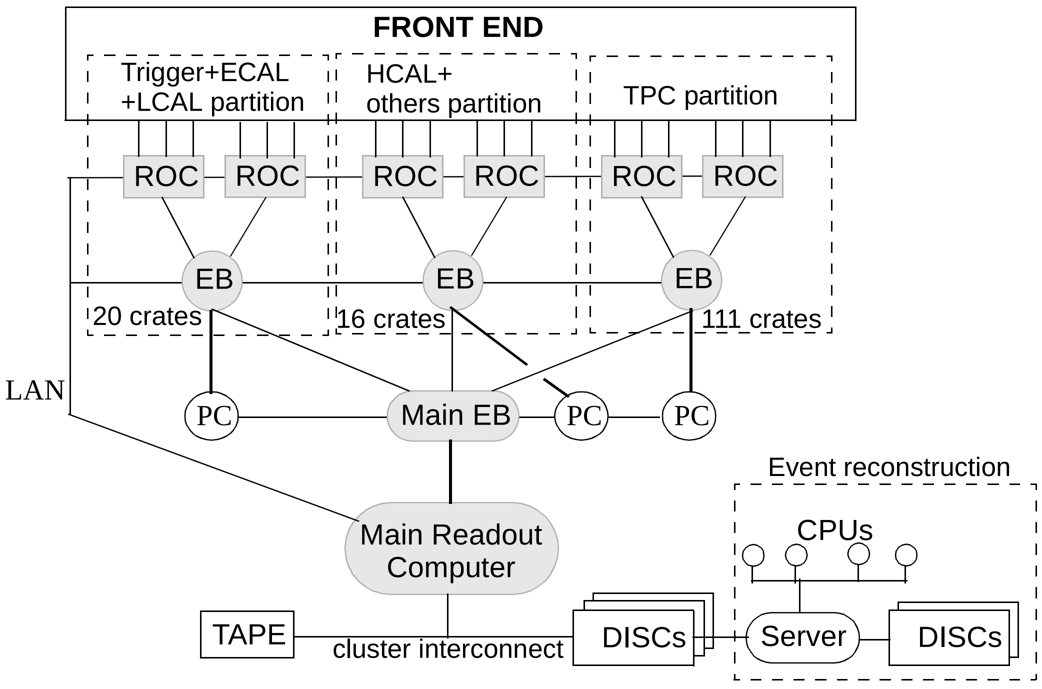


Fig. 1-5. ALEPH DAQ system architecture.

1.3.3. New trends in the 90's

Today's DAQ and trigger architectures are based on ideas and trends introduced during the nineties:

1. The decade began with CAMAC and Fastbus coexisting as the front-end I/O standards, though a growing tendency to use VME not only at the back end but also at the front end resulted in **all-VME experiments**. This was possible due to the extension of the VME board size to 9U to accommodate more electronics in a single board and the use of ASICs that reduced the board size requirements for both the front-end and the VME interface logic.
2. VME did well during the eighties and early nineties, but the crate-based backplane architecture cannot be used as the backbone in large DAQ systems with data rates in the order of gigabytes per second. **Switch-based event builders** and **point-to-point high speed links for communication between modules** replace VICbus for inter-crate connection and the VME backplane at the crate level. An early discussion on switch-based event building

can be found in [Barsotti90]. Table 1.1 shows the bandwidth limitations of the backplane and inter-crate buses [Parkman90].

Table 1.1. Bandwidth limitations of backplane and inter-crate buses

Bus	Raw bandwidth	Effective bandwidth
CAMAC	20 MByte/s	below 1 MByte/s
VME	40 MByte/s	10-20 MByte/s
VICbus	30 MByte/s	3 MByte/s
Fastbus[a]	200 MByte/s	40-60 MByte/s

a. Performance on Fastbus segment interconnects is around 40 MByte/s.

3. At the beginning of the nineties, the way of collecting data was discussed and **push architectures** were studied [Dorenbosch91]. Classical architectures relied on computing nodes pulling data from the upstream stage, building a sub-event and buffering it until pulled from the downstream stage (like in the ALEPH DAQ system). An increased performance[1] can be obtained with push architectures in which processing nodes accept data from the upstream stage, build a sub-event and push it to the downstream stage. Flow control becomes then an issue in order to avoid destination buffer overflow.
4. Expensive computing facilities are replaced by **PC-based computer farms** running Linux [Sphicas99], interconnected via network technologies like Ethernet or Myrinet.

These trends can be found in experiments designed in the nineties like the ALICE experiment at LHC. This experiment [LHCC95-71] will exploit proton-proton and heavy ion-ion (Pb-Pb and Ca-Ca) collisions. Thus, the trigger systems must be designed for reconfigurable trigger algorithms and the DAQ system must allow different combinations of event rates (40 Hz to 1 kHz) and sizes (5 to 40 MByte).

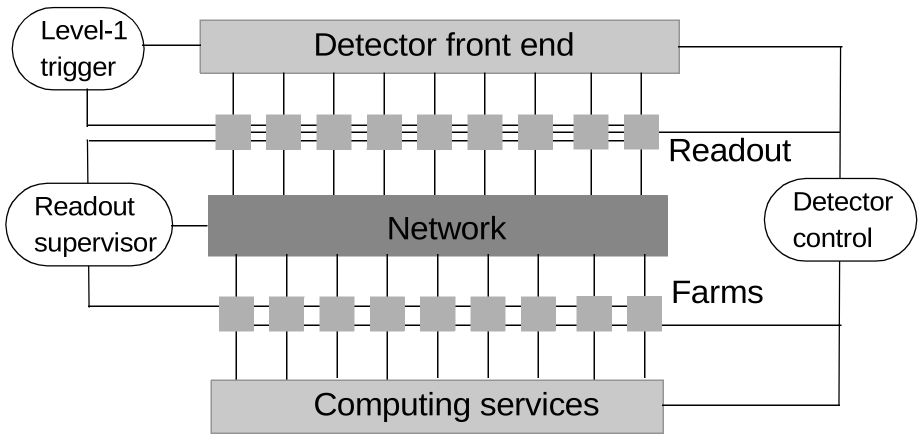


Fig. 1-6. DAQ in the nineties: event-building networks for gigabyte data rates.

Data is transferred from the front-end electronics to RORC (Read-Out Receiver Cards) modules, which are grouped in Front-end Digital Crates. Event fragments from a number of RORCs are merged into sub-events, buffered and transmitted to the GDCs (Global Data Concentrator) by

1. As a result of a simpler protocol.

LDC cards (Local Data Concentrator). Thus, the combination of RORCs and its associated LDC perform the same functionality as the LHCb Readout Unit. Throughput requirement for a LDC is in the order of 100 MByte/s. GDCs are the final destinations and build complete events.

The large event size in ALICE (the largest in all four LHC experiments) poses the double requirement of very high bandwidth and capacity per permanent Data Storage (DS) unit.

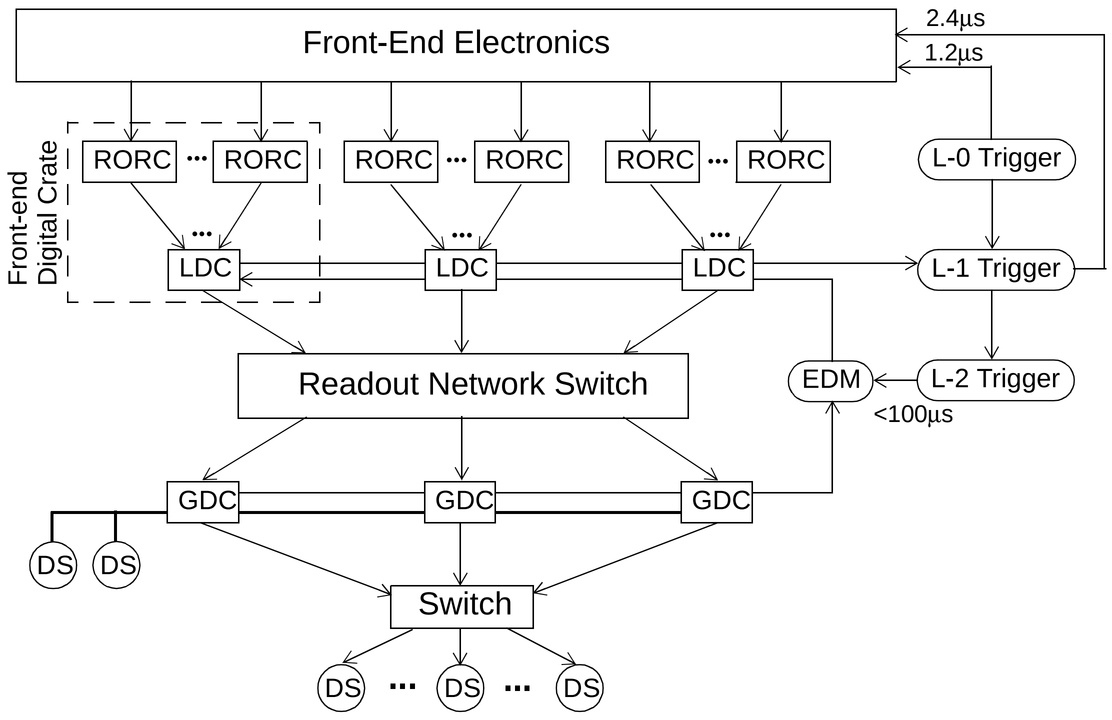


Fig. 1-7. **ALICE DAQ and Trigger architecture.**

1.3.4. Trends in DAQ systems for the 21st century

Today’s high-bandwidth applications require up to 200 MByte/s throughput per DAQ module and thus an alternative to the conventional approach of “data processing across the VME backplane” must be found. Nowadays, there is a clear trend towards using **PCI** as a high-speed readout and DAQ bus. Two key technologies boosted the widespread of PCI in the HEP arena: the PMC standard and the advent of high-gate-count FPGAs.

The PMC standard (PCI on a CMC form factor, i.e., 15-by-7.5 cm mezzanine card size), which dates from 1995, added a large degree of freedom, allowing drop-in integration any device compliant with the PMC standard, like for example network interfaces for network standards, embedded CPUs and high-speed I/O interfaces. This standard invaded the VME arena and the embedded processing migrated from the “bus-resident crate controller” towards exchangeable, local processing units. The FPGA technology backed this trend in two ways:

1. **Boosting local processing** with FPGA-based coprocessors. The implementation of specialized algorithms in an FPGA using only integer and boolean operations, in comparison to a CPU, can result in a large performance gain [Hinkelbein00]. Data compression, track

finding, trigger algorithms and sub-event building can be implemented either on FPGA-based co-processor mezzanines, assisting a CPU on the main board, or entirely in FPGAs.

2. **Increasing the logic density of integration** by packing in a single device control logic, glue logic and small memories, thus easing the integration of complex systems in a small mezzanine form factor. Recent examples can be found in PMCs for real-time data processing [Pascual01], in the S-Link specification for link technologies on CMC mezzanine cards [Bij97] and ADC cards with FPGA control and processing [Baird00].

The use of PCI in readout and DAQ applications can be classified in two paradigms:

1. **On board data processing**: Data is actually more efficiently processed locally on-board (migrating from “data processing across the backplane” to “data processing across the on-board bus”) and I/O is routed using today’s high-speed link technologies. This relegates VME and Fastbus to the role of convenient mechanical frameworks with a “power and control bus connector” rather than providing bus functionalities via backplanes. This paradigm (see Figure 1-8) relies on the use of PMC [PMCstd] and other non-standard mezzanine cards for exchangeable I/O interfaces. The motherboard provides an on-board PCI bus for mezzanine interconnection, on-board memory and additional FPGA- or microprocessor-based intelligence. In the conservative approach, the backplane is left in the crate in order to maintain backwards compatibility, but the PCI bus is integrated into the DAQ module logic for high-speed chip-to-chip and mezzanine-to-mezzanine communication.

   A recent implementation of this paradigm is described in this thesis: the Readout Unit for the LHCb experiment [Tol01-1]. It is a 9U-sized card with four CMC mezzanines in the front panel for data input, on-board PCI bus for FPGA-based event building and a PMC network interface card on the rear for data output. Hybrid solutions like the one described in [LeVine00] use front-panel links for data input, mezzanines for data processing, PCI for on-board mezzanine interconnection and VME64 for board readout.

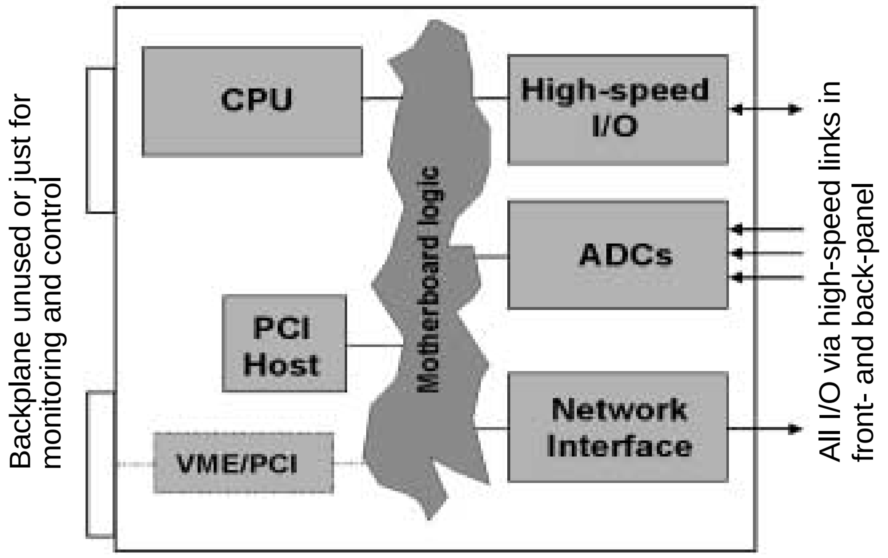

Backplane unused or just for monitoring and control
CPU
High-speed I/O
ADCs
PCI Host
Motherboard logic
Network Interface
VME/PCI
All I/O via high-speed links in front- and back-panel

Fig. 1-8. On-board data processing paradigm.

2. **Data processing across the PCI backplane**: This second paradigm aims at using a PCI backplane as DAQ backplane, with the benefit of superseding VME bandwidth limitations.

Some PCI derivatives, like the CompactPCI standard appeared, but the acceptance was mild. The PC is a much more successful (and cheaper) PCI readout platform, housing up to six PCI cards in an off-the-shelf PC, which provides power and cooling, mechanical support, a PCI backplane and control and monitoring via the CPU. Additionally, the PCI cards become part of the host's plug&play domain.

Following this second paradigm, a PCI card with FPGA, memory and mezzanine connectors can result in a generic readout platform (see Figure 1-9). For a given application, only the specific mezzanine card has to be designed (ADCs, I/O interfaces, etc.). The FPGA interfaces the mezzanine, the PCI bus and the memory, allowing on-board data processing and the flexibility to implement typical DAQ architectures [Muller01-2]. Scalability is limited by the number of PCI slots on the motherboard (typically six), the PCI bus bandwidth and the processing requirements in the CPU. References [Brosch98], [Drochner01], [Müller01-2] and [RACE-1] describe PCI cards approaching towards the described concept which we call Flexible I/O concept [Tol01-2].

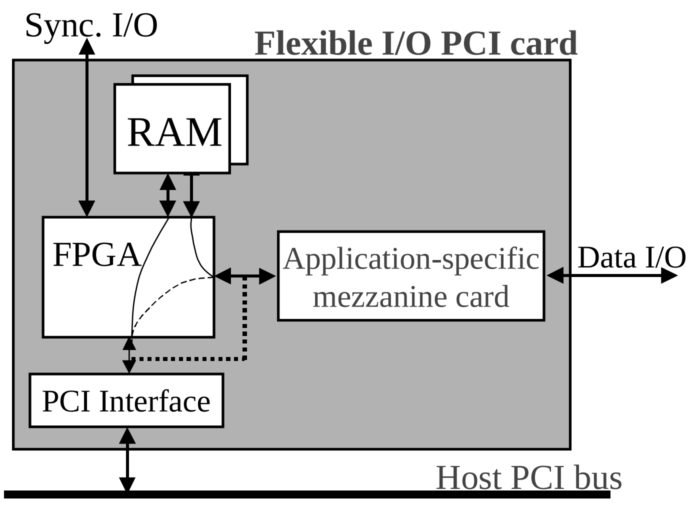


Fig. 1-9. The Flexible I/O concept.

1.4. The PCI Flexible I/O Card (PCI-FLIC)

The PCI-FLIC represents our implementation of the Flexible I/O concept. It was designed in a late stage of the thesis work, targeting the NA-60 experiment at CERN (see Appendix II) and the Readout Unit test station (see Chapter 6).

Compared to other Flexible I/O cards, all data paths are 64-bit wide, including the PC-100 SDRAM and the PMC connector. Combined with 64-bit 66-MHz PCI bus capability, the PCI-FLIC is well suited for very high throughput applications. Different readout architectures (dual-port memory, FIFOs, circular buffers and swing-buffers of different widths up to 64 bits) are supported. Data processing may be implemented in the FPGA or in the mezzanine.

Physically (Figure 1-12), the PCI-FLIC card is a 32/64-bit 33/66-MHz universal PCI card with connectors for a variety of mezzanine adapters, centered on a 120-kilogate FPGA (center, top) with an embedded PCI master/target core. This PCI card contains two banks of 32-MByte SDRAM (right) and two general-purpose LVDS I/O plugs (top left corner). The FPGA logic serves as programmable interface between the connectors, the PCI bridge (center, bottom) and local SDRAM. There are five mezzanine connectors for PMC-32 (including user defined I/O connector), 64-bit PMC extensions and custom mezzanines like S-Link.

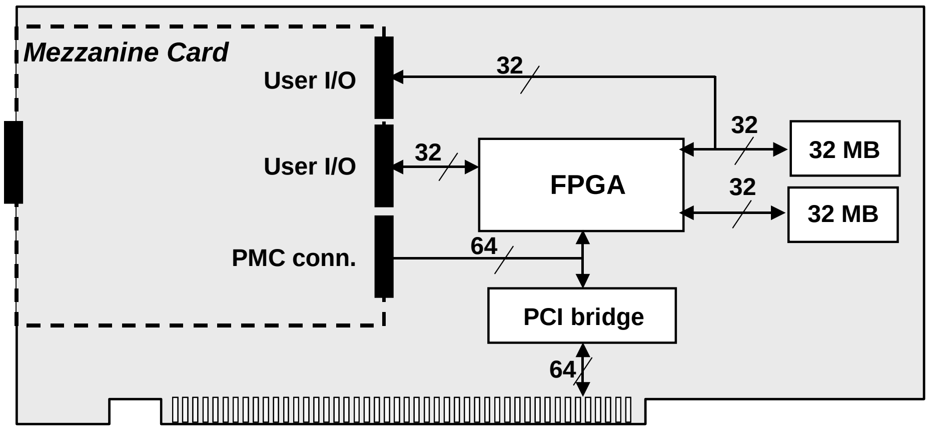


Fig. 1-10. PCI-FLIC card architecture.

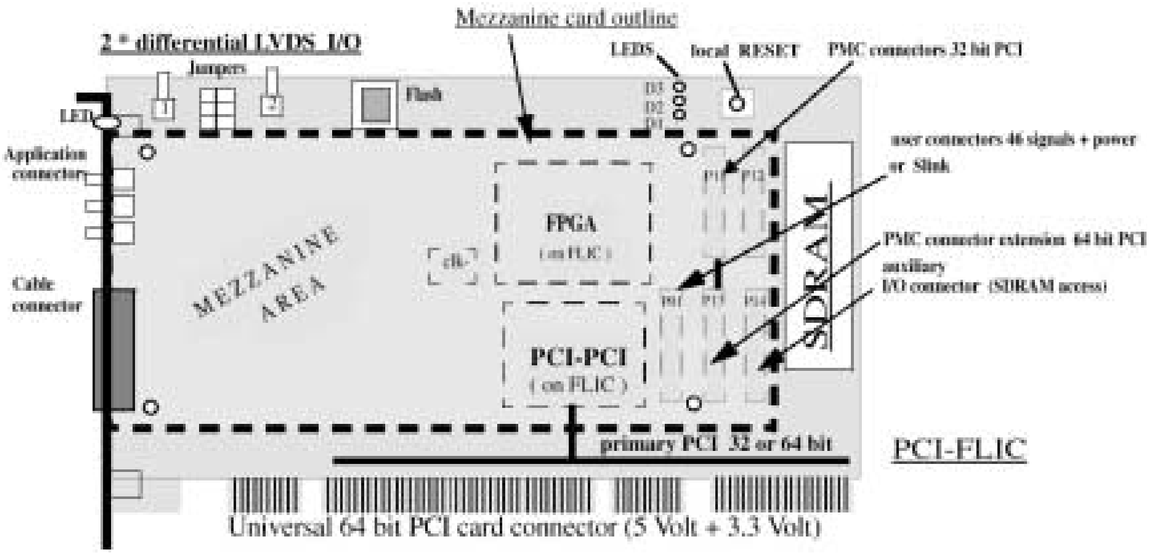

Mezzanine card outline
2 * differential LVDS I/O
Jumpers
Flash
LEDS
local RESET
PMC connectors 32 bit PCI
LED
Application connector
Cable connector
MEZZANINE AREA
clk
FPGA (on FLIC)
PCI-PCI (on FLIC)
user connectors 46 signals + power or Slink
PMC connector extension 64 bit PCI
auxiliary I/O connector (SDRAM access)
SDRAM
primary PCI 32 or 64 bit
PCI-FLIC
Universal 64 bit PCI card connector (5 Volt + 3.3 Volt)

Fig. 1-11. PCI-FLIC outline.

Using a commodity PC as host, two major environments can be targeted:

1. **Linux environment**: A Linux 2.2.x driver allowing up to five PCI-FLIC cards was developed by the NA-60 experiment. It supports CPU-initiated data transfers to and from the PCI-FLIC SDRAM at 37.8 MB/s [David01]. A utility was written to directly reconfigure the FPGA from the host CPU via the PCI bus, allowing (in a networked PC) remote reconfiguration [David01-2].
2. **Windows environment**: A Dynamic Link Library (DLL) for Windows 98/2000/NT has been written using Jungo's Windriver utility[1]. This library provides functions to access the

memory and configuration registers in the FCI-FLIC card, and can be used by higher-level software layers. An interface for LabView has been developed based on this DLL, allowing control and monitoring applications via graphical interfaces [Martínez&Toledo].

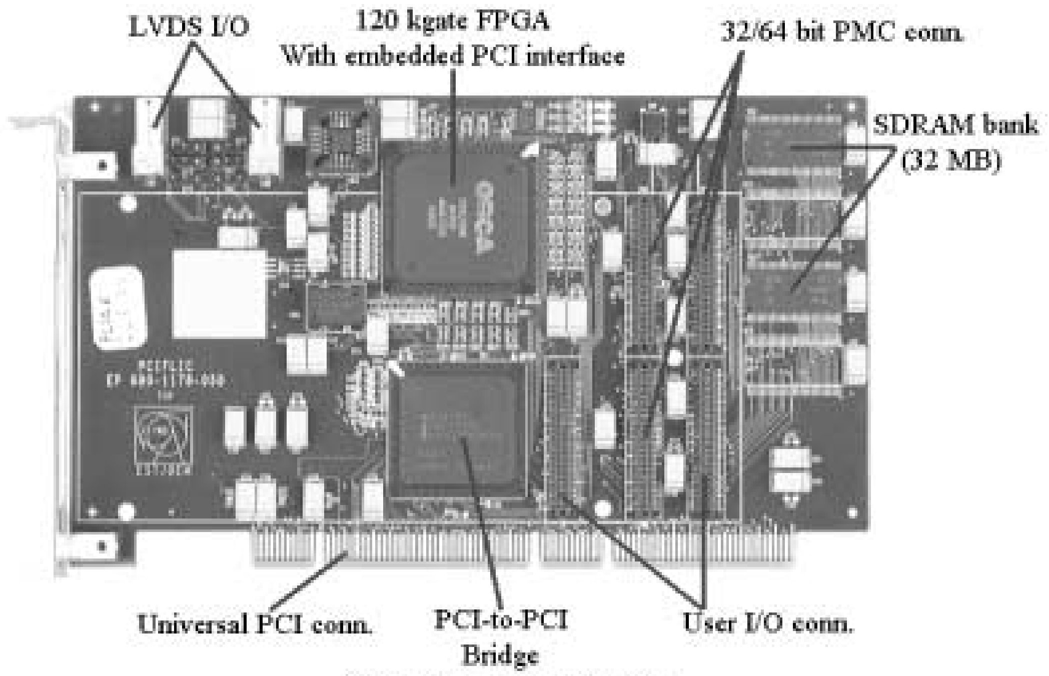


Fig. 1-12. Picture of the PCI-FLIC card.

1.5. *Conclusions*

The next generation of large HEP experiments, represented by the four LHC experiments, will work under unprecedented combinations of trigger rate and event sizes (see Figure 1-13, extracted from [Sphicas99]), requiring extensive application of the new trends and paradigms presented in the previous sections.

In one of its applications, the LHCb Readout Unit will have to assemble sub-events at a 1 MHz trigger rate, with a 200-to-240 MByte/s throughput per module. As it will be described in this thesis, this figure is not so far away from today's technology limits. Under these conditions, VME is not a valid solution any more.

As an industry standard for telecommunication, control and data acquisition applications, PCI is the right bet for today's DAQ system in both large and small experiments. Two approaches are followed nowadays: the "data processing across the on-board bus" used for the Readout Unit, and the "data processing across the PCI backplane". The latter approach finds an interesting platform in commodity PCs, leading to the "Flexible I/O concept" in which standard mezzanines and PCI cards co-exist inside a PC to create flexible and cost-effective DAQ solutions.

1. See http://www.jungo.com

One of the most powerful representatives of this new concept has been designed as a part of this thesis: the PCI-FLIC card.

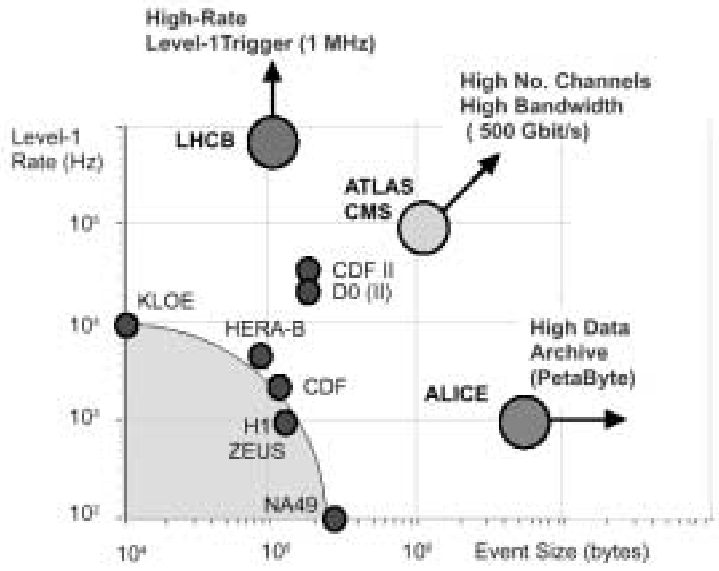


Fig. 1-13. LHC experiments compared other experiments in terms of event rate and size.

CHAPTER 2 DAQ and Trigger systems in LHCb

"Never worry about the theory as long as the machinery does what it's supposed to do"

Robert A. Heinlein, Science Fiction writer.

2.1. The LHCb experiment at CERN

2.1.1. LHC: A new accelerator for new physics

In the quest for new physics, extremely high collision energies are needed. The next research instrument at CERN will be an upgrade to the existing LEP (Large Electron-Positron collider) accelerator: the LHC (Large Hadron Collider). It is designed to share the 27-kilometre LEP tunnel and will be fed by existing particle sources and pre-accelerators. The LHC will be a versatile accelerator. It will be able to collide proton beams and also heavy ions beams such as lead with a total collision energy in excess of 1250 TeV, about thirty times higher than at the Relativistic Heavy Ion Collider (RHIC) under construction at the Brookhaven Laboratory in the USA. Joint LHC/LEP operation can supply proton-electron collisions with 1.5 TeV energy, some five times higher than presently available at HERA in the DESY laboratory, Germany.

Four main experiments have been designed for the LHC:

- **ATLAS** (A Thoroidal LHC Apparatus): It is a general-purpose experiment for recording proton-proton collisions at LHC. The detector has been optimized to cover the largest possible range of LHC physics like searches for the Higgs boson, supersymmetric particles, new gauge bosons, leptoquarks and quark and lepton compositeness indicating extensions to the Standard Model, CP violation in B-decays, measurement of quark properties, etc.
- **CMS** (Compact Muon Solenoid): It is also a general-purpose detector for Higgs, super symmetry and heavy ion physics at LHC.
- **ALICE** (A Large Ion Collider) is a general-purpose heavy-ion detector designed to study the physics of strongly interacting matter and quark-gluon plasma in nucleus-nucleus collisions.

- **LHCb** (A LHC Beauty experiment) is an experiment dedicated to the study of CP violation and other rare phenomena in B-meson decays.

2.1.2. A Large Hadron Collider Beauty Experiment (LHCb)

The main purpose of the LHCb experiment is the study of the CP violation in B-meson decays. CP violation was first observed in neutral kaon decays in 1964 and has been object of study since then. If it is true that the Standard Model explains the observed asymmetry in the neutral-kaon systems, it cannot account for the degree of CP violation that would explain the dominance of matter over antimatter in our universe, thus suggesting the existence of CP violation sources beyond the Standard Model.

The Standard Model makes precise predictions for CP violation in B-meson decays, so its study is an attractive place for searching for new physics and a number of HEP experiments have been designed to study the B-meson system. But none of them with the $B\overline{B}$ cross section and luminosity of the LHC that will make it the most copious source of B mesons.

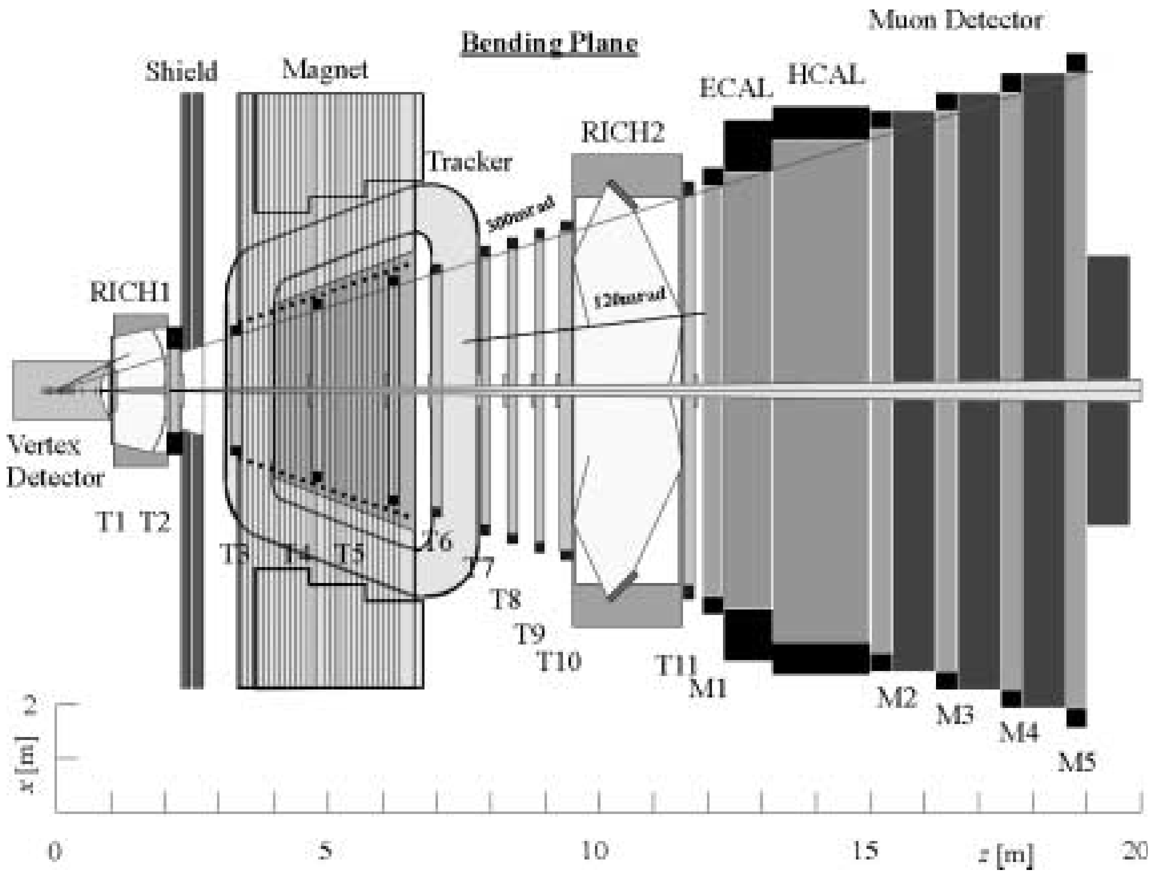


Fig. 2-1. LHCb detector seen from above (bending plane).

The LHCb is a single-arm spectrometer[1] which covers only particles produced between 15 and 300 mrad, comprising a silicon vertex detector, a tracking detector, two RICH[2] detectors (RICH1 and RICH2), an electromagnetic calorimeter, a hadron calorimeter and a muon detector

1. As proton-proton interactions at LHC energies will produce B and $\overline{B}$ mesons in the same forward cone.

(Figure 2-1). A superconducting magnet is placed behind RICH1, surrounding the tracking system, and an iron shield protects the vertex and RICH1 detectors from the magnetic field. The vertex detector provides precise information on the primary and decay vertices and is the base for the Level-1 trigger. It includes also a pile-up veto counter (two dedicated planes of silicon detectors used to count the number of primary vertices) to reject at the Level-0 trigger events with multiple proton-proton interactions. The detector consists of seventeen stations perpendicular to the beam, spaced in such a way that a typical track traverses between five and six stations. Each station consists of two silicon discs, one providing information on radius and the other on angle.

The tracking system provides information on track reconstruction and momentum measurement in charged tracks, as well as direction information for the RICH. The RICH detectors identify charged particles. The electromagnetic and hadron calorimeters identify position and energy of electrons and hadrons and are used for both off-line and trigger.

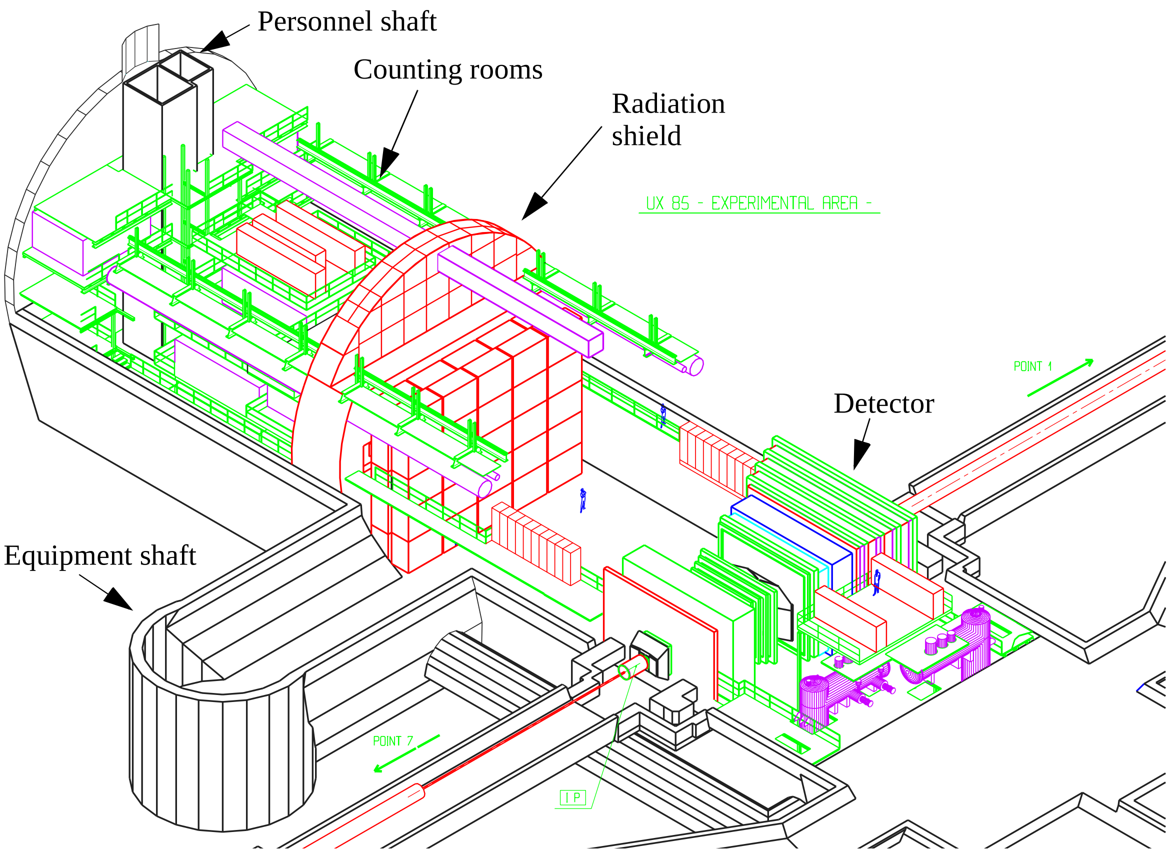


Fig. 2-2. Isometric view of the LHCb experimental area.

The muon detector identifies muons and its information is used in Level-0 trigger. Iron absorbers surrounding the detector stations block all particles but muons, thus providing efficient muon identification.

2. RICH stands for Ring Imaging Cherenkov, a charged particle detector. It is based on the Cherenkov effect, according to which a charged particle traversing a medium faster than the local speed of light produces the emission of light. Pavel Alekseyevich Cherenkov was awarded the 1958 Physics Nobel Prize for the discovery of this effect.

The experimental area (Figure 2-2) is divided into two zones by a radiation shield. Front-end electronics residing close to the detector must be designed to stand high doses of radiation whilst the counting rooms behind the concrete wall provide a radiation-free environment.

Front-end electronics for Muon, Calorimeters, Outer tracker and possibly also Inner tracker and Vertex detector will reside in the cavern close to the detector. Level-0 and Level-1 trigger electronics, RICH detectors and DAQ system will be placed in the counting rooms.

The integrated data rate among all detectors is 40 TByte/s, for a LHC bunch crossing frequency of 40 MHz and an average zero-suppressed event size of 100 KByte. The resulting amount of data is too high for storage and off-line analysis, so events are filtered and only a small fraction are stored. The filtering (trigger) and detector data readout after the Level-1 trigger (data acquisition) systems are presented in “LHCb trigger and DAQ architecture” on page 31.

2.2. Front-end electronics overview

The front-end system is defined as the processing and buffering of all detector signals until they are delivered to the DAQ system via high-speed optical links. The analog signals coming from the detectors are amplified, digitized and buffered during the latency of Level-0 and Level-1 trigger levels and finally zero suppressed and formatted for the DAQ system (see Figure 2-3).

The Level-0 electronics receive the amplified analog detector signals from the analog front-end, condition and store the data into the Level-0 pipeline buffer during the fixed 3.2 µs Level-0 latency. Level-0 buffer is 128-event deep, matching the Level-0 latency ($128 \cdot 25ns = 3,2\mu s$). So, after the Level-0 latency, data are either discarded or passed to the derandomizing buffer waiting to be transferred to the Level-1 electronics. At the output of the Level-0 derandomizing buffer the data rate has been reduced in a factor of 40 by the Level-0 trigger. Data from 32 channels can now be multiplexed to share a link to the Level-1 electronics and/or an ADC.

The Level-1 electronics receive the event data accepted by the Level-0 trigger, which is stored into the Level-1 buffer waiting to be accepted or rejected by the Level-1 trigger. Analog signals are digitized, as the long Level-1 latency (up to 256 µs) does not allow analog data to be stored with sufficient precision at an acceptable cost. After a Level-1 accept, event data are passed to the Level-1 derandomizing buffer waiting to be serviced by a zero-suppression unit (with the possible exception of special monitoring and calibration events). After the zero suppression, data must be properly encapsulated with header and trailer information to enable the DAQ system to handle each event fragment[1].

The DAQ system is specified to be capable of handling all event data generated by the front-end without any back pressure (Xon/Xoff) mechanism on the local DAQ links.

1. According to the transport format described in “Proposed Sub-event Transport Format (STF) for the DAQ” on page 73.

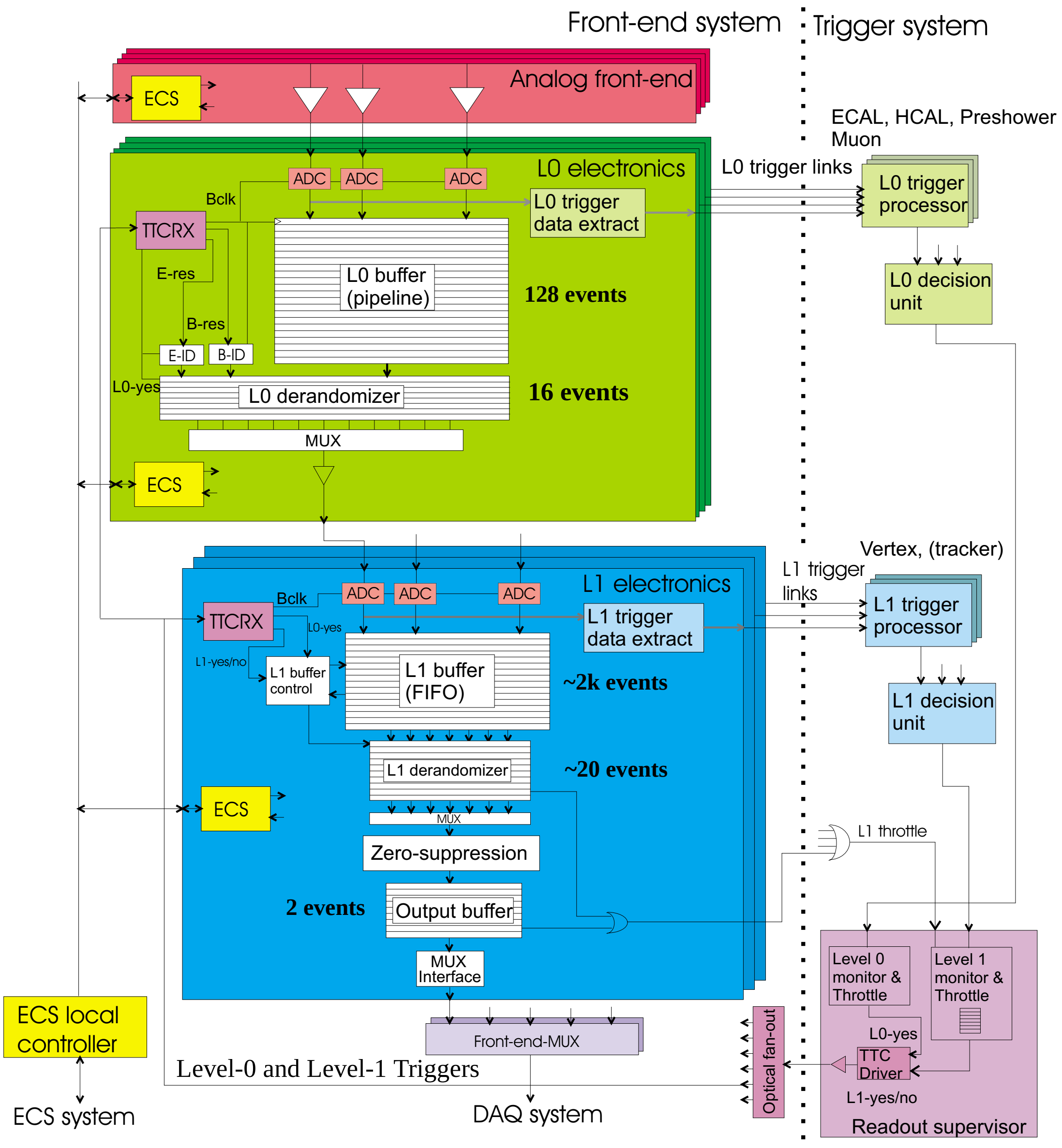


Fig. 2-3. Front-end electronics architecture in LHCb.

2.3. *LHCb trigger and DAQ architecture*

Figure 2-4 depicts the LHCb trigger and DAQ system architecture as defined in [LHCC98-4]. The challenge is not small: close to one million detector channels are read each 25 ns (determined by the 40-MHz LHC bunch crossing frequency) generating an integrated data rate of 40 TByte/s. The data from these million channels must be gathered into a single event. Identification and reconstruction algorithms are applied and only those events which contain interesting phenomena (like specific B-meson decay modes) are finally sent to permanent storage for off-line analysis. In order to cope with these complex and time-consuming tasks (event-building and filtering) two strategies are applied:

- The **event-building** functionality is distributed across several stages of multiplexers. Those stages comprised between the sub-detectors and the Level-1 buffer outputs are grouped under the name of “front-end electronics”, whilst those stages beyond the Level-1 buffers form what is called the data acquisition system (DAQ).
- The **event filtering** functionality is carried out in successive steps of increasing algorithm complexity, each step reducing the number of events that reach the next. Each step is called “trigger level” whilst the whole system is called “trigger system”.

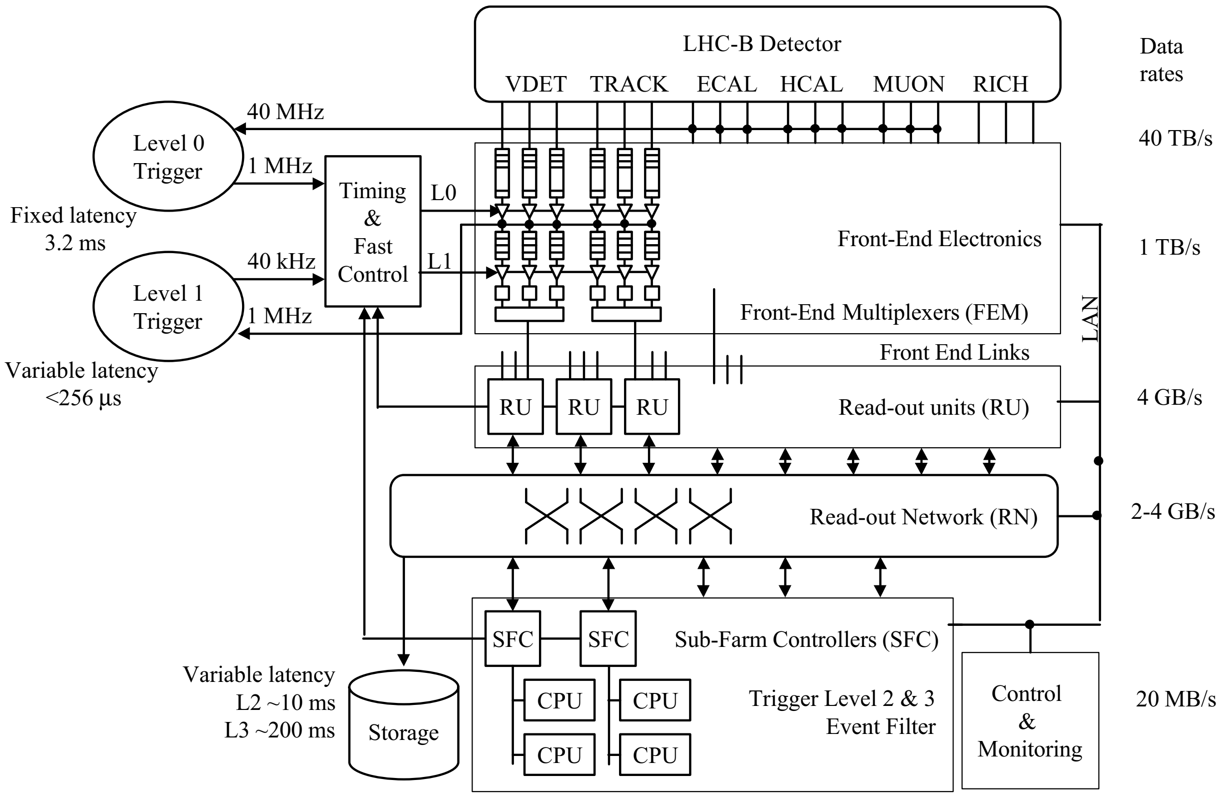


Fig. 2-4. LHCb Trigger and DAQ system architecture.

2.3.1. *Trigger system*

The trigger system is responsible for delivering trigger decisions that allow the front-end system to reduce the amount of data delivered to the DAQ system by three orders of magnitude (from 40 MHz down to 40 kHz) and allow the event filter farm to reduce this 40 kHz rate down to 200 Hz for permanent storage. The first level trigger (Level-0) is a hardware-implemented constant-latency trigger, whilst the others are software-implemented and variable-latency triggers. Four levels of trigger in total are defined to achieve a total suppression factor of $2 \cdot 10^5$ in 40-25-8-25 steps:

- **Level-0** uses information from the pile-up veto (to reject events with more than one proton-proton interaction[1]) and from the calorimeters and muon chambers (to reject events with low

1. LHCb has chosen to operate at a low luminosity ($2 \cdot 10^{32} cm^{-2} s^{-1}$) to reduce the number of events with multiple proton-proton interactions, which are more difficult to reconstruct. This results in 9.3 MHz single proton-proton interactions.

transverse momentum), operating at a frequency of 40 MHz and providing a suppression factor of 40. Event data is stored in the 128-event-deep Level-0 buffers and, after a fixed 3.2 μs Level-0 latency, events are either discarded or forwarded to the Level-1 buffers.

- **Level-1** uses information from the Vertex Locator (VELO). Level-1 trigger logic works at 1 MHz and provides a suppression factor of 25, resulting in a 40 kHz Level-1 accept rate.
- **Level-2** achieves a suppression factor of 8 by eliminating events with fake secondary vertices using information from the Vertex and Tracker detectors [LHCb98-017]. The Level-2 algorithms are executed in a processor farm with an estimated processing power requirement of $4 \cdot 10^5$ MIPS.
- **Level-3** trigger works at an average 5 kHz frequency. The level-3 algorithms are designed for filtering the decay modes of interest, resulting in an estimated suppression factor of 25 and thus a final accept rate of 200 Hz.

Being one of the three applications of the RU, the Vertex Locator and Level-1 trigger deserve a more detailed description. The vertex detector is depicted in Figure 2-5. It consists of 17 stations, each composed of two silicon-strip discs with different strip layout: one disc provides information about the distance to the beam axis (radius coordinate, *r*) whilst the other gives information about the angle (*phi* coordinate). The polar coordinates of a particle hit, together with the station number and sector number (discs are divided into six sectors each) identify a point in the space.

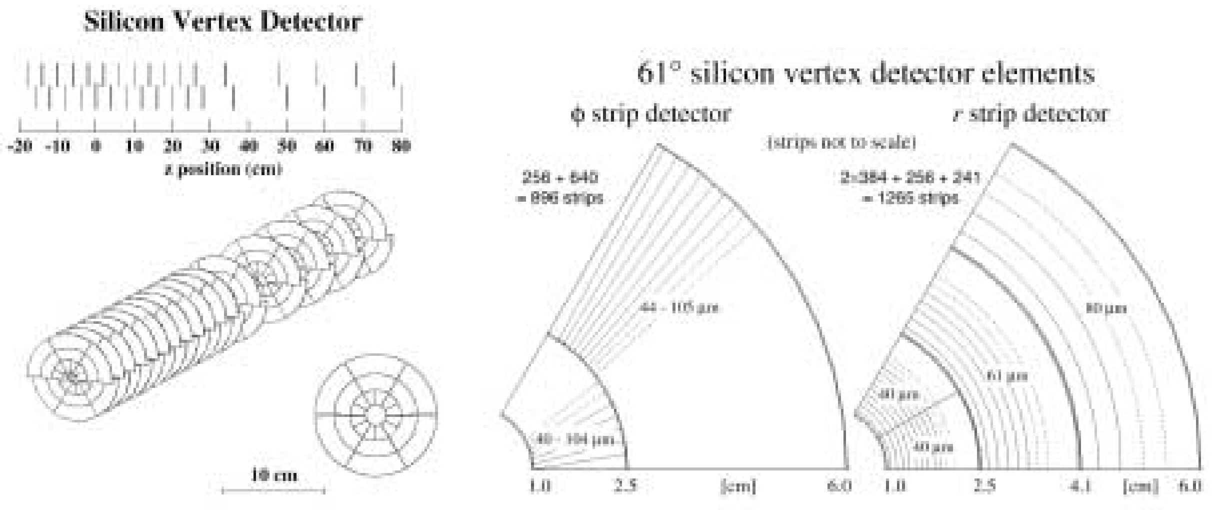


Fig. 2-5. The vertex detector topology showing the individual *r* and *phi* detectors.

The Level-1 algorithm consists of the following steps [LHCb00-001], [LHCb98-006]:

1. **Track finding** in the *r*-projection. A track is identified by three colinear points in three consecutive stations (triplet). Tracks must have positive slope and a hit on a radius smaller than 25 mm to be accepted.
2. **Calculate the primary vertex** position by histogramming the intersection point of the previously computed tracks with the beam axis.
3. **Calculate the impact parameter** for all tracks (a measure of the distance to the primary vertex). If the number of tracks with large impact parameter exceeds a maximum, pile-up is suspected and the event is rejected.

4. **3-D reconstruction** of large impact parameter tracks by adding the *phi* information.
5. **Search for secondary vertices**, i.e., potential B decays.

Among different possibilities [LHCb98-002], [LHCb98-033], [Walsch01-2] the detector readout and trigger algorithm will be implemented on a two-dimensional torus network based on SCI ringlets, as described in “The Readout Unit as readout module for the Level-1 Vertex Locator Trigger” on page 45.

The timing control of the complete front-end system and the delivery of the two first trigger level decisions are performed by a global **Readout Supervisor** unit (TFC in Figure 2-4) over a TTC[1] system using optical fibers. The Readout Supervisor has a vital role in collecting Level-0 and Level-1 trigger decisions from the respective trigger decision units and only passing trigger accepts that do not risk to overflow any part of the front-end and DAQ system. Each of the triggers must be closely monitored and trigger inhibition (throttling) must be applied based on a well established functional model of the front-end system.

2.3.2. *Data Acquisition (DAQ) system*

The purpose of the DAQ system is to accept and buffer incoming event fragments from the front-end electronics following a Level-1 trigger accept and assemble them into complete events. The main functional blocks of the DAQ system are:

- **Front-end multiplexer** (FEM): Data from the Level-1 electronics (in some cases called ODE, Off-Detector Electronics) typically needs to be concentrated by a factor between eight and sixteen in order to reduce the number of links going from the front-end electronics to the DAQ system (see Figure 2-3). This will in most cases require one or possibly two FEM modules per front-end electronics crate. This concentration could in some cases be performed using a standard backplane bus (e.g. VME). A backplane bus cannot be easily scaled (bandwidth at this level is highly dependent on detector occupancy as zero-suppression has been performed), so it is much more attractive to use point to point links between the modules and the concentrator, as such a scheme can be reconfigured quickly to handle higher bandwidths.

  As the FEM handles data from many detector channels it is vital that it can be serviced while LHC is actively running. It should therefore be located in the counting room and long distance (~100 m) optical link technologies must be used if the corresponding Level-1 electronics are placed close to the detector.
- **Readout Unit** (RU): The entry stage to the DAQ system consists of RUs that assemble incoming event fragments into larger sub-events and send them to the destinations (sub-farm Controllers, SFCs) via the Readout Network (RN). The two multiplexing stages following the front-end modules (i.e., FEMs and RUs) concentrate the data to match the number of links and bandwidth per link to the requirements of the RN. Additionally, FEMs and RUs provide the required buffering between the front-end and the RN. The multiplexing factor in the RUs is a number between one and four.
- **Readout Network** (RN) is a N-by-M switch that routes all sub-events belonging to the same event to one out of M destinations (sub-farm controllers). The throughput in the network is 100 Kbyte at 40 kHz, i.e., 4 GByte/s.

1. TTC: Trigger and Timing Control.

- **Sub-farm Controller** (SFC): The SFCs build complete events and allocate each event to one of the CPUs it controls. Level-2 and Level-3 trigger algorithms execute on this CPU.
- **Readout Supervisor**: The basic concept in the LHCb data flow is that destinations must always accept data sent by the sources. If a destination buffer fills beyond a certain point, the destination will a assert a throttle signal. All throttle signals from FEMs, Readout Units and Sub-Farm Controllers, reach the Readout Supervisor module (Timing and Fast Control in Figure 2-4), resulting in a Level-0 and Level-1 trigger inhibition when any of the throttle inputs is asserted. This implies that, in order not to lose already buffered events, destinations must still have room to accept all buffered events in the upstream sources when asserting the throttle signal.

  The Level-0 trigger must be passed to the TTC driver in a completely synchronous manner with a minimum latency. A global Level-1 derandomizer buffer will be required before the TTC driver as the TTC system only has a limited bandwidth available for the Level-1 trigger distribution. In case of the Level-1 trigger, the front-ends and the DAQ system must also be capable of throttling the accept rate in order to prevent buffer overflows.

  When the Level-1 throttle is asserted, the Readout supervisor will translate all Level-1 trigger accepts into Level-1 trigger rejects and thereby stop the flow of event data into the Level-1 derandomizer. The Level-1 throttle network will have certain delay before the readout supervisor will actually enforce Level-1 trigger rejects. The Level-1 throttle network is considered part of the TFC (Timing and Fast Control) system which includes features for partitioning via a set of programmable routing switches.

Data flow across the DAQ system according to a defined readout protocol. Two protocols are currently under investigation [LHCb98-028]:

1. **Full readout protocol**. The preferred full readout protocol is conceptually simple: a continuous, write-only data flow is maintained from source to destinations, i.e. from the RU's sub-event buffers to the Subfarm Controllers (SFCs) of the CPU farm. This implies that full events (4 GByte/s) are transferred across the Readout Network.
2. **Phased readout protocol**. Under this LHCb protocol, only part of each event data are transmitted for making Level-2 trigger decisions whilst the rest of the event is queued in the RU's sub-event buffer. Since the remainder of the data is only transferred after a positive Level-2 decision, the bandwidth across the Readout Network may be reduced by a factor of two.

The full readout protocol is preferred for its simplicity, despite the fact that it implies more bandwidth across the network.

2.3.3. Flow control in LHCb

When a buffer in any of the ~1000 Level-1 front-end boards, a FEM or a RU gets full beyond a programmable threshold, the corresponding module asserts a throttle signal and the Readout Supervisor inhibits further Level-1 and Level-0 triggers (see Figure 2-7).

The LHCb policy is to process the events that are already stored in the Level-1 buffers. Thus, it must be guaranteed that even under worst case conditions buffers will never overflow (which would require a system resynchronization). According to Figure 2-7, up to eighteen events (sixteen in the derandomizing buffer and two in the outgoing buffer) are stored in Level-1 boards after a Level-1 accept. A FEM module must still accept 18 maximum-size event fragments after asserting its throttle output. This implies 288 KByte, assuming 16 KByte as the maximum sub-event size at the FEM output (sixteen times the nominal value[1]).

The latencies involved in the trigger throttling determine the low and high watermarks for the throttle logic in FEM, RUs and SFCs (Figure 2-6). When the fill state reaches the high watermark the buffers continue to fill during the throttle latency and then drop at the speed of the readout until the low watermark level is reached and the throttle is released. The buffer will keep on emptying during the throttle-off latency. If the low watermark is set too low as in this example, it results in an empty state of the buffer (and thus in DAQ system inefficiency).

For a FEM module, the high watermark must be set to allow at least 18 more maximum-size sub-events and the low watermark must be set to a level greater than 12 maximum-size sub-events. The latter is estimated by considering a maximum 256 µs Level-1 trigger latency (i.e., 10.2 events at 40 kHz) and rounding up to 12 to include other latencies. Assuming a maximum sub-event size of 16 KByte, the distance between the high watermark and the buffer overflow level must be at least $18 \cdot 16KByte = 288KByte$. The low watermark must be set to at least $12 \cdot 16KByte = 192KByte$, thus determining a minimum FEM buffer size of half megabyte.

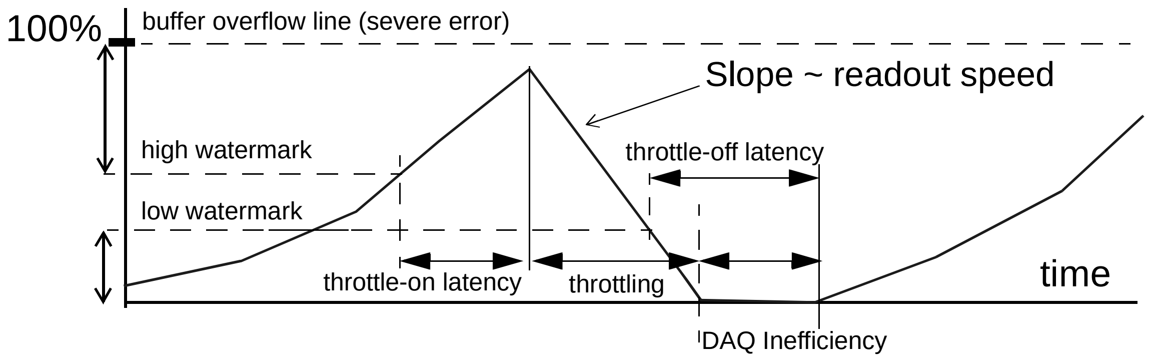


Fig. 2-6. Sub-event buffer fill state monitoring and throttle signal generation.

A DAQ RU module must still accept from each FEM, after asserting its throttle output, the already buffered data in the FEMs plus additional eighteen maximum-size sub-events due to the Level-1 buffers flushing. As it has been estimated, this does not exceed half megabyte per FEM input in any case. The distance between the high watermark and the buffer overflow level must be at least four times this value (i.e., two megabyte) for a four-input DAQ RU.

The estimated Level-1 output to DAQ-RU input latency is 200 microseconds maximum (see Figure 3-15), which is equivalent to eight Level-1 triggers. For a multiplexing factor of four, the low watermark level must be set to at least $8 \cdot 4 \cdot 16KByte = 512KByte$. Thus, the minimum buffer size in a DAQ RU is 2.5 MByte.

The small sub-event size in the VELO application (256 byte) and the fact that in this case the RUs are directly connected to the Level-0 trigger data extract, result in extremely low buffering requirements.

1. This is a reasonable assumption, though this parameter has not been fixed yet in the LHCb experiment.

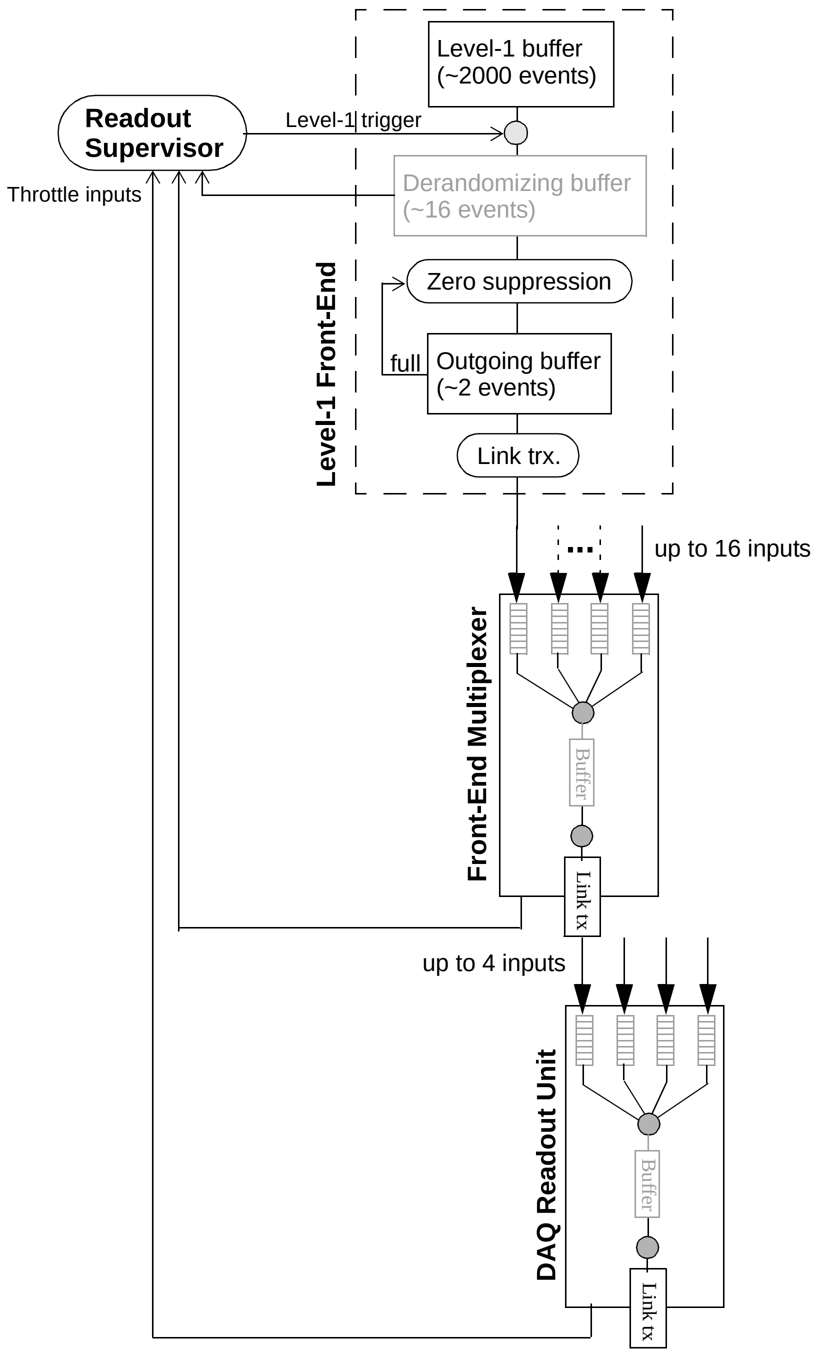


Fig. 2-7. Level-1 Trigger throttling.

To sum up, the buffer requirement is 2.5 MByte for the DAQ RU application, 0.5 MByte for the FEM and even less for the VELO application.

CHAPTER 3

The Readout Unit for the LHCb experiment

"Truth suffers from too much analysis"

From the novel "Dune Messiah" by Frank Herbert.

3.1. Target applications in the LHCb experiment

As it was mentioned on page 3, three modules belonging to the LHCb front-end electronics (FEE), trigger and DAQ systems share a common development under the name of the Readout Unit Project. These modules are:

1. Readout Unit for the Level-1 Vertex Locator (VELO) Trigger.
2. Readout Unit for the DAQ system as an entry stage into the Readout Network (RN).
3. Front-end Multiplexer (FEM) for the interface between the FEE and the DAQ system.

The benefits of having a single module for these three applications are evident: reduced costs and ease of maintenance and support. The LHCb technical proposal [LHCC98-4] considers these three modules as separate designs. An investigation of the required functionalities and performance of the FEM in 1998 [Tol98-1] led to the conclusion that the DAQ RU and the FEM could be the same design. The first RU prototype was designed to target these two applications [Tol99-1].

The definition of the architecture of the Level-1 VELO trigger electronics during 1999 allowed to identify the interface module between the Level-1 electronics and the VELO readout network as a third target application for the LHCb Readout Unit project. The Readout Unit II was designed in year 2000 to fulfil also the requirements of this third application [Müller00], [Müller01-3].

The three referred application areas (shaded blocks in Figure 3-1) are described in detail in the following sections.

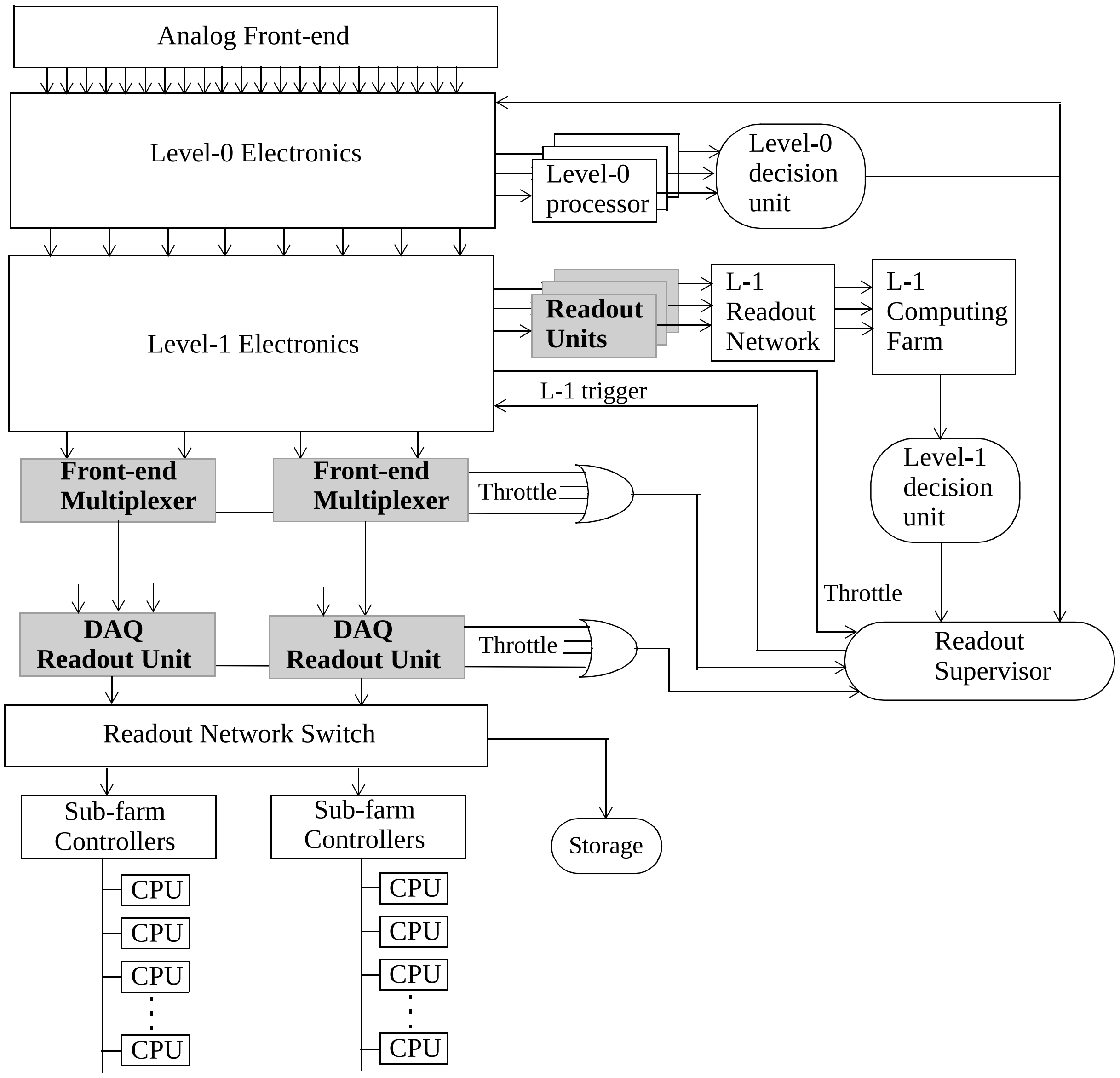


Fig. 3-1. The three areas of application of the Readout Unit in the LHCb DAQ and Trigger systems.

3.1.1. The Readout Unit as input stage to the LHCb DAQ system

The DAQ RUs belong to the DAQ system as depicted in Figure 2-4. In the standard DAQ application, as described in the technical proposal [LHCC98-4], RUs receive sub-event fragments from multiple front-end links sourced by Front-end Multiplexers (see Figure 3-2), assemble them into larger sub-events and transfer them to the next stage for further event building. The RU is then part of the event building system. Sub-events are built according to the LHCb sub-event transport frame format ("Proposed Sub-event Transport Format (STF) for the DAQ" on page 73). Sub-events are queued in a sufficiently large sub-event buffer (SEB) inside the RUs to compensate for fluctuations and throttling latencies.

Next stage in the event building system consists of a layer of sub-farm controllers (Figure 3-1). Both layers, the RU and the SFC, are interfaced via a Readout Network as described in “Data Acquisition (DAQ) system” on page 34. The multiplexing factor in the RUs is chosen to match the aggregated bandwidth on its inputs with the output bandwidth towards the Readout Network and is a number between one and four.

The LHCb readout system requires that an average throughput of 4 GByte/s (100 KByte events at a 40 kHz rate) flows through a stage of DAQ RUs. Each RU is connected at its output to an input port of a N-by-M switch, where N, M depend on the chosen network technology. The RU and FEM layers in the LHCb readout architecture are shown in detail in Figure 3-2.

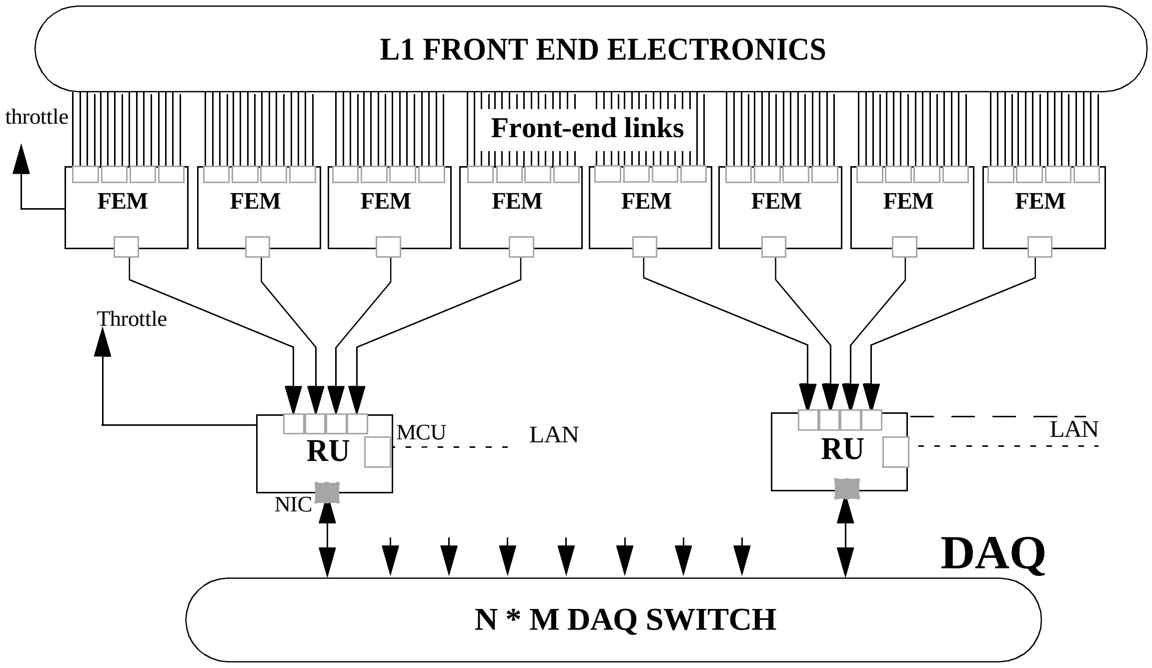


Fig. 3-2. The Readout Unit as entry stage to the DAQ system.

The transmission of sub-events from the SEB buffer to individual subfarm controllers (SFC) is subject to the event-building protocols which also depend on the Readout Network and switch technology chosen by LHCb.

The nominal requirements for a DAQ Readout Unit are given by the LHCb Level-1 trigger rate and sub-event size per link. As shown in Table 3.1, the detector with the highest occupancy (RICH 1) produces 690-byte event fragments into the DAQ RU[1]. Including some formatting overhead this can be extrapolated to roughly 1 KByte maximum event fragment size per FEM output. With four detector links at a 40 kHz Level-1 rate, this corresponds to a 160 MByte/s nominal throughput.

The described functionalities and requirements allow to identify the following functional blocks inside the DAQ RU (see Figure 3-3):

1. According to the LHCb Technical Proposal.

- **Input FIFOs**: Also called ‘derandomizing buffers’ in the HEP jargon, store incoming sub events until they are read by the data merger block.
- **Data merger**: Identifies sub-events with the same event number and merges them in the sub-event buffer.
- **Sub-event Buffer**: Decouples the input stage (FIFOs and data merger) from the output stage (sub-event builder and NIC), compensating for the fluctuations in data throughput provoked by congestion in the Readout Network and flow control, NIC latencies and throttle latencies.
- **Sub-event builder**: Builds larger sub-events from the sub-events stored in the buffer and sends them to the NIC according to the NIC’s technology-specific requirements.
- **NIC (Network Interface Card)**: Intended as a commercial plug-in card[1] to allow technology changes and upgrades, interfaces the RU to the Readout Network.
- **MCU (Monitoring and Control Unit)**: Allows to monitor, configure and control the RU from the slow control system (Experiment Control System, ECS).

Table 3.1. Simulated event fragment size into the RUs for the different sub-detectors

Sub detector	Event size (bytes)
Vertex	240
Inner Tracker	230
Outer Tracker	600
RICH 1	690
RICH 2	250
Preshower	330
E. Calorimeter	500
H. Calorimeter	500
Muon calorimeter	125
Trigger	500

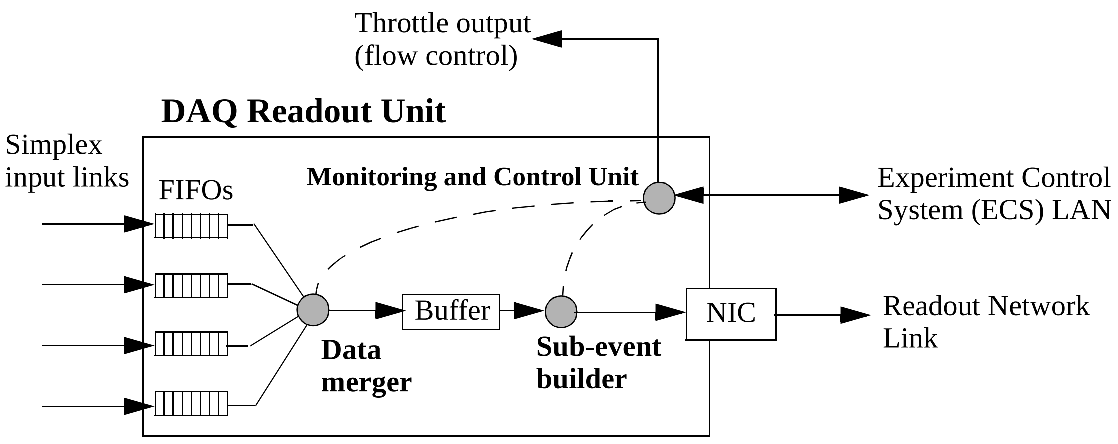


Fig. 3-3. DAQ Readout Unit functional blocks.

1. In a PCI or PMC form factor, as PCI is today’s *de facto* industry standard for Network Interface Cards.

A throttle output to the Readout Supervisor is also needed to avoid buffer overflow, as described in the previous chapter. The design parameters are summarized in Table 3.2.

Table 3.2. Requirements for DAQ application

Parameter	RU
Front-End Link inputs	Up to four Gbit/s links with nominal 40 MByte/s occupancy, maximum 80 MByte/s. Flexible link technology (parallel, serial, copper, optical)
Output link	NIC interfaced via PCI bus
Throughput (nominal)[a]	160 MByte/s at a 40 kHz operation
SEB buffer[b]	Minimum is 2 MByte
Sub-event building	According to LHCb transport format convention
MCU and LAN	Mandatory for initialization, PCI, Monitoring and Control. 10/100 Mbit/s port via RJ45
Event building protocols	Full readout (Phased readout only on demand)

a. multiplexing factor of four.

b. In the previous chapter, a 2.5 MByte requirement was set. This includes input FIFOs and sub-event buffer. FIFO size is determined by the 64-KByte non-zero-suppressed event fragment size in calibration runs. Thus, only two megabyte are needed in the sub-event buffer if 128-KByte FIFOs are used.

3.1.2. *The Readout Unit as Front-end Multiplexer for the Level-1 Electronics*

The number of Level-1 buffer output links (in the order of 1000) are reduced via Front End Multiplexers (FEMs) before transferring the data to the RU modules (see Figure 3-2). For this purpose, the FEM modules must provide a multiplexing factor that matches the aggregated input bandwidth on its inputs with the output bandwidth into the RU layer. This factor is estimated to be a variable number between 1 and 16. There are in principle two approaches to multiplexing: simple concatenation (which does not require buffering) and sub-event building. FEM modules in LHCb use the latter approach and thus the requirements and functionalities are similar to the DAQ RU application, however event-building protocols across the Readout Network are normally unused and the sub-event buffer can be smaller. The nominal throughput is in the order of 40 MByte/s, since four FEM outputs connect to one Readout Unit with 160 Mbyte/s nominal throughput.

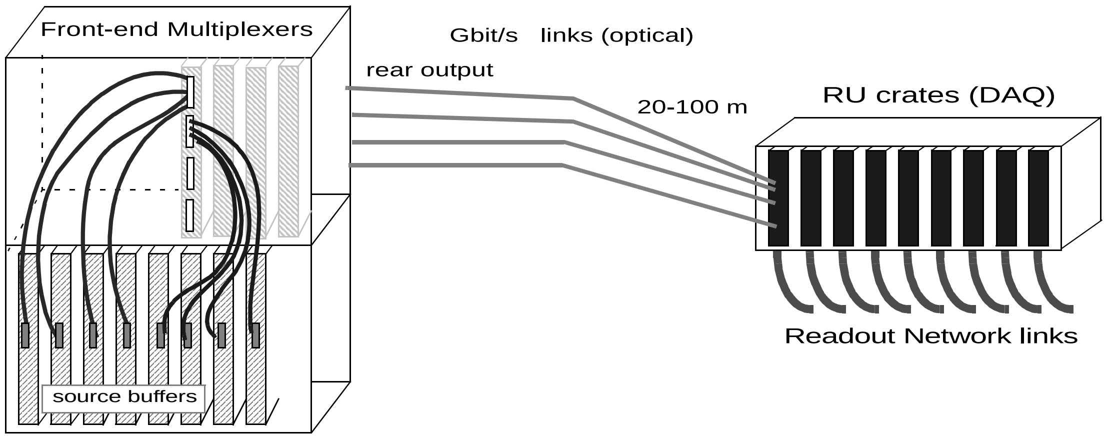
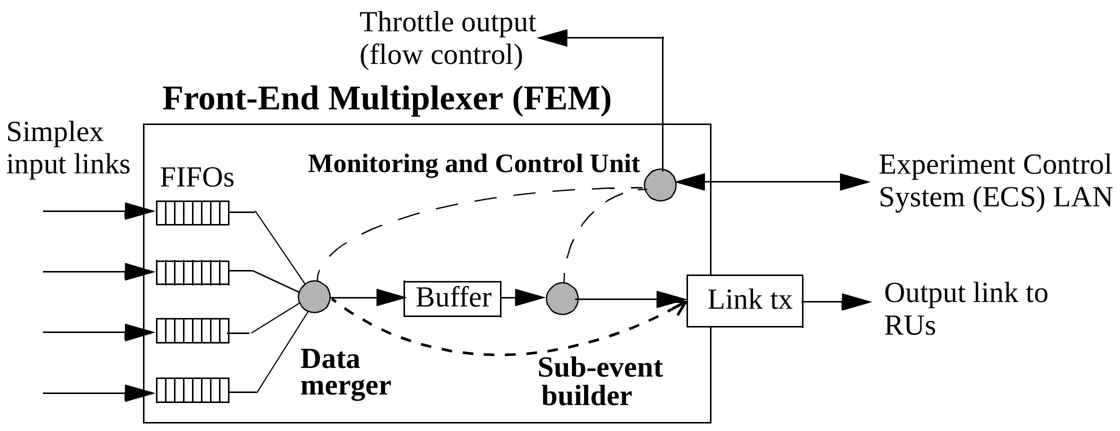


Fig. 3-4. Combined use of RUs as Front-end Multiplexer and DAQ entry stage.

If the FEM and the DAQ RU are the same design, it would allow a FEM unit with enough output bandwidth to output directly to the DAQ Readout Network via a NIC. Figure 3-4 shows a front-end crate housing FEM modules which receive data from the front-end sources and send sub-events to the DAQ Readout Units. The described functionalities and requirements allow to identify the following functional blocks inside the FEM (see Figure 3-5):

- **Input FIFOs**, **MCU (Monitoring and Control Unit)**: as described in 3.1.1.
- **Data merger**: Identifies event fragments belonging to the same event and either merges them in the sub-event buffer or directly outputs the concatenated event fragments to the FEM output.
- **Buffer**: Only present in buffered applications. Decouples the input stage (FIFOs and data merger) from the output stage (sub-event builder). Small buffering space required, as the LHCb data flow defines that FEMs push data to the next stage without any flow control mechanism.
- **Sub-event builder**: Only present in buffered applications. Builds sub-events from the event fragments stored in the buffer and sends them to the RU stage for further event building.



Throttle output (flow control)
Front-End Multiplexer (FEM)
Simplex input links
FIFOs
Monitoring and Control Unit
Experiment Control System (ECS) LAN
Buffer
Link tx
Output link to RUs
Data merger
Sub-event builder

Fig. 3-5. FEM functional blocks.

The design parameters for the FEM application are summarized in Table 3.3.

Table 3.3. FEM Requirements

Parameter	RU
Front-End link inputs	Up to 16 Gbit/s links with nominal 2.5 MByte/s. Flexible link technology
Output link	Default is flexible Gbit link technology. Optionally, also NIC card
Throughput (nominal)[a]	40 MByte/s at a 40 kHz rate
SEB buffer	Minimum is 0.5 MByte
Sub-event building	According to LHCb transport layer convention
MCU and LAN	Mandatory for initialization PCI, Monitoring and Control. 10/100 Mbit/s port via RJ45
Event building protocols	Only applicable if FEM outputs directly to RN. Full readout (phased readout only on demand)

a.Multiplexing factor of sixteen.

3.1.3. The Readout Unit as readout module for the Level-1 Vertex Locator Trigger

In the L1-VELO application, RUs receive event fragments from the VELO detector at a nominal rate of 1 MHz (maximum 1.17 MHz according to [LHCb99-031]). With a 1-MHz operation from all the detector's *r* and *phi* stations and 1100 hits per event[1] coded in two bytes for each *r* and *phi* coordinates, this corresponds to a 4.4 GByte/s data flow. With an input multiplexing factor between three and of four, there are around 20 RUs which have to stand 180-240 MByte/s throughput each. Event fragments are received in a Readout Unit, merged into larger sub events with a multiplexing factor of three or four, and transmitted to a shared-memory event-building network which is arranged like a 2D-torus with Level-1 trigger farm CPUs at each crossing point [Schulz01]. The torus is currently implemented as horizontal and vertical SCI [SCIstd] ringlets of 800 MByte/s bandwidth each[2], thus requiring three or four RUs per ringlet. The application for the L1-VELO network is shown in Figure Figure 3-6.

A daisy-chained token bus (namely TagNet) connects the RUs to a Scheduler Unit which takes care of destination allocation and traffic scheduling. It is important to make sure that each column is used only by one sender at a time, otherwise network congestion may occur. The Scheduler Unit keeps track of available CPUs, pending events (i.e., those for which a token has been generated but has not reached back the Scheduler Unit) and columns in use. So, it is possible for the Scheduler Unit to issue tokens containing clever x-y routing information to ensure proper operation of the network. According to [Müller00] the Scheduler Unit can be implemented on one additional Readout Unit for uniformity.

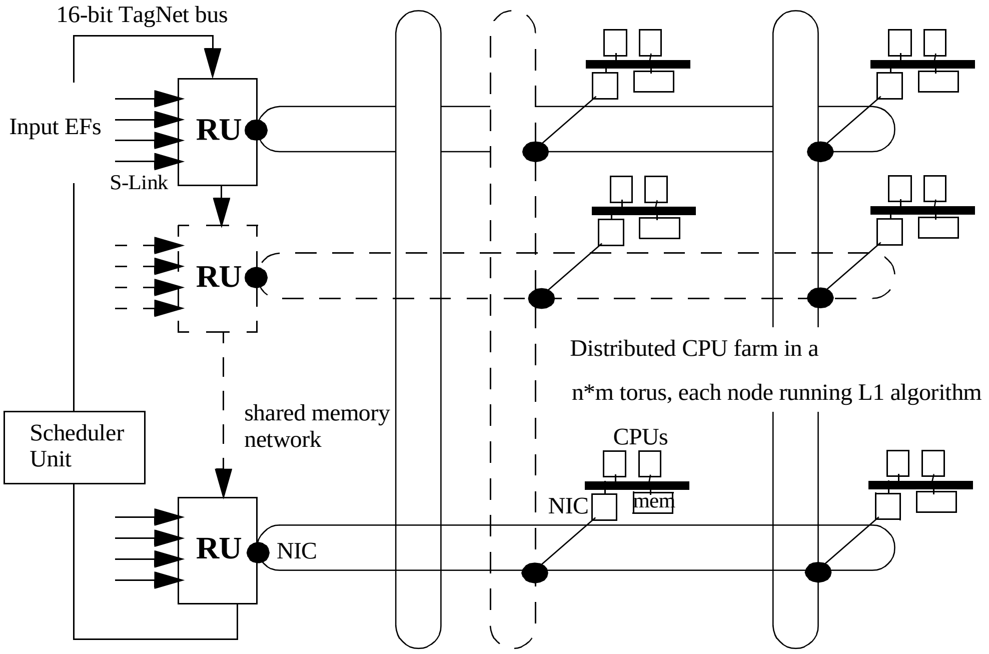


Fig. 3-6. 2-D torus network with RUs as data sources for the Level-1 VELO trigger data acquisition.

1. Noise accounts for 400 hits, so only 700 hits in average are due to real data [LHCb99-031].
2. A six-node torus has been built and its performance has been studied using 64-bit 66-MHz PCI-to-SCI NIC adapters [Walsch01-2], demonstrating that SCI NICs can handle more than 300 MByte/s each and that the integrated bandwidth in a torus can reach 690 MByte/s.

The described functionalities and requirements allow to identify the following functional blocks inside the VELO RU (see Figure 3-7):

- **Input FIFOs, Data merger, MCU (Monitoring and Control Unit), Buffer**: as described in 3.1.1.
- **TagNet interface**: Receives and transmits scheduling and destination information from/to the TagNet daisy-chained bus.
- **Sub-event builder**: Builds complete sub-events from the event fragments stored in the sub-event buffer and sends them to the SCI NIC using a push protocol.
- **NIC**: An SCI interface card which interfaces the RU to the SCI ringlet in which it is inserted.

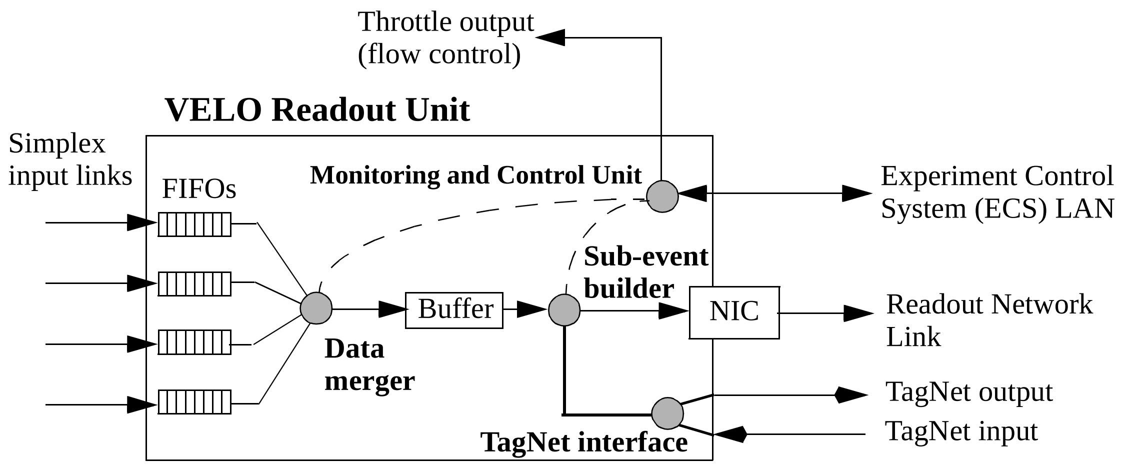


Fig. 3-7. VELO Readout Unit functional blocks.

The RU requirements for this application are shown in Table 3.4. Apart from the TagNet interface, the functionalities are compatible with the DAQ and FEM applications. The low latency implied in the TagNet operation can be achieved if implemented on FPGAs.

Table 3.4. RU requirements for the Level-1 Vertex application

Parameter	RU
Front-end link inputs	Up to four Gbit/s links with 60 MByte/s occupancy each. Flexible technology (parallel, serial, copper)
Output link	SCI
Throughput (nominal)	240 MByte/s at a 1-MHz operation
SEB buffer[a]	0.5 MByte[b]
Sub-event building	According to Level-1 convention
MCU and LAN	optional for Monitoring and Control. 10/100 Mbit/s port via RJ45
Event-building protocols	Shared memory, TagNet scheduled transmission

a. Limited by the dual-port memory integration level.

b. Enough for 2000 sub-events or 2 ms of operation.

3.2. Design criteria and parameters

Design criteria for the RU module are:

- **Simplicity, reliability and ease of maintenance**. A large number of RUs (around two hundred modules) installed in the LHCb cavern will channel all the detector’s data for the lifetime of the experiment (10-15 years); thus a simple, reliable and easy to maintain design is required. The more components on a board, the higher the probability of failure, and the more different parts the more difficult it is to keep a reasonable stock or find spare parts. So, the RU must be designed for a low component count and a small number of different components. The use of mezzanine cards in the design will further simplify test and repairing operations. The development of a test station with diagnostic tools to be used during all phases of the project’s life (commissioning, installation and data taking) is mandatory.
- **Scalability** in terms of throughput and number of input links at the system level, and throughput at the board level. Level-1 accept rate is defined as 40 kHz in the LHCb Technical Proposal, though trigger efficiency considerations may force an increase that should be also foreseen in the design. At the system level, the scalability is achieved by adding more RUs (and hence reduce the multiplexing factor and the throughput per module).
- **Capability to follow developments in I/O technologies**. As the I/O scenario evolves in industry, it must be possible to change the I/O technologies in the RU. This implies that the I/O data formats and protocols should be programmable and technology independent. The use of mezzanine cards will allow this feature.

The original RU design parameters for the three applications in the LHCb experiment are summarized in Table 3.5.

Table 3.5. Design parameters for RU applications in LHCb

PARAMETER	FEM	DAQ	VELO
Number of inputs	1 to 16	1 to 4	1 to 4
Nominal fragment size per input	64 byte	1 KByte	60 byte
Nominal sub-event size	1 KByte	4 KByte	240 byte
Nominal rate	40 kHz	40 kHz	1 MHz
Throughput	40 MByte/s	160 MByte/s	240 MByte/s
Max. sub-event buffer capacity[a] (sub-events)	2000	500	8000

a. For a 2 MByte buffer.

3.2.1. *DAQ Readout Unit functional requirements*

The following points summarize the RU functionalities and requirements in the context of the DAQ RU application described in [Harris98] and [LHCC98-4], reference documents released before the Readout Unit project started. Clarifications and actualizations are shown in the footnotes:

- The RU mission is to receive sub-events from several front-end links and assemble them into larger sub events. Once a new sub event is assembled, the RU transfers it to the next stage for further event building.
- The multiplexing factor in the RU is chosen to match the bandwidth on its inputs with the output bandwidth towards the Readout Network.
- A destination must be assigned to each sub event. This can be done locally, by means of a look-up table (dynamically updated), or can be done by an external dedicated controller.
- Must support both the Full Readout and the Phased Readout schemes. The latter implies in average 10 ms buffering. This sets the required buffer size[1].
- The RU stage must support an average event size of 100 KByte at an average Level-1 rate of 40 kHz. In average, at every Level-1 trigger, between 0.2 and 0.7 KByte of data per front-end link have to be handled in the RUs.
- Must support remote control, monitoring and configuration via an external LAN.
- All RUs must be equal. Simplicity, scalability, ease of maintenance and updating to technology improvements are design criteria.
- Error detection must be performed on the incoming front-end links. Error recovery procedures should be local[2].
- A fast reset must be available for all internal buffers in the RU to enable easy error recovery.
- Must support partitioning in DAQ system in order to run test on different partitions concurrently and asynchronously. A partition is defined as a subset of DAQ system that has been configured to function independently of the rest of the system.
- Must support several modes of running: physics data taking, cosmic triggers, calibration running (not zero-suppressed data, large events but at a lower rate) and test pattern running (artificially generated at the front end).
- Must be able to generate data patterns for testing other parts of DAQ system (test triggers).
- Must generate a throttle signal to avoid the overflow of its internal buffers. This signal is fed into the trigger supervisor[3].
- Event number: Assuming a lifetime of an event in DAQ of 1s, an Event Number ID of 20 bits is required[4].
- The trigger number and bunch crossing number must be added into each event fragment[5].
- In order to allow proper synchronization of event building, null fragments should be generated when the sub detector element has no data.
- All detector data, on a Level-1 accept, must be pre-processed and multiplexed in order to put zero-suppressed data, in an agreeded protocol[6], onto high bandwidth links leading to the DAQ.

1. Obsolete. The buffer size is determined by the throttle latency.
2. After careful evaluation, it was decided that transmission error recovery mechanisms will not be implemented.
3. Readout Supervisor according to the current nomenclature.
4. This corresponds to a Level-0 event number (1 MHz rate). DAQ event numbers are incremented after a Level-1 accept, so 20 bits are enough for more than 26 seconds at a 40 kHz rate.
5. Obsolete. The DAQ event number is the only reference used by the DAQ system.
6. A protocol is suggested in this thesis, which was accepted by the LHCb Collaboration. See “Proposed input and output link technologies” on page 79.

3.2.2. *Global decisions on the Readout Unit module design*

The feasibility and the architecture of the Readout Unit have been studied taking into account these requirements and the functionalities described in previous sections. The RU study and design project was started in 1998 [Müller98] with regular meetings [RUhist] in course of which a series of decisions were taken and presented to the collaboration in the LHCb october DAQ workshop 1999 [Müller99]. The global design decisions for building a common module targeting both the DAQ RU and FEM applications were:

- The sub-event building logic is to be implemented on FPGAs, as a microprocessor- or DSP-based solution would not yield the required performance.
- Physical format: The electro-mechanical RU crate standard is the existing LEP crate standard (9U Fastbus, IEEE 960) but without using any Fastbus signals (only power and cooling). The backplane remains unused as high-speed point-to-point links are used for inter-module communication (following the trend presented in “New trends in the 90’s” on page 18) and there is no need of communication between RUs that are at the same hierarchical level. The reasons to use Fastbus and not VME mechanics are: (1) the availability of a large number of Fastbus crates which could be acquired for free as a result of the dismantling of LEP experiments, and (2) the upper half of the Fastbus backplane can be removed and thus Fastbus boards can accommodate two PMC slots on the back-panel. VME, on the other hand, inconveniently requires transition modules to use the back-panel.
- Extensive use of mezzanine card technology to allow for evaluation and upgrade of all interfaced technologies.
- S-Link with derandomizing FIFOs at the link receiver inputs.
- PCI and S-Link at the output.
- Embedded processor system for initializing the PCI bus and PCI network cards.

The decisions taken during all following LHCb workshops and meetings are listed below:

- Sub-event building in the Readout Unit is carried out in the same way for the FEM and the RU DAQ, using the proposed sub-event Transport Format (STF).
- The event data (payload) transported within the STF transport frame are ignored by the RUs and FEMs, hence they need to be independently formatted.
- Error data are optionally appended in the trailer of the STF (Error Blocks).
- For error tracing, the identification number of the module from which sub-events are transmitted is used as a geographical identifier of the link.
- Fast error flags are contained in the STF trailer to allow any stage (Level-1 electronics, FEM, RU) to mark error conditions, or to quickly identify incoming erroneous data.
- The DAQ event number in the STF header is assigned by the Level-1 data sources and must be monotonically incremented. This adds one degree of consistency in the information.
- There is exactly one event fragment sent out by any data source per trigger. This implies that also empty data frames must always be sent for consistency, otherwise the sub-event building algorithm should include a time-out mechanism, adding unnecessary large latencies.
- Readout Units can be fully configured and controlled remotely via a networked server (RUs have no connection to a backplane bus). RUs can be individually reset via an independent reset line.
- The full readout[1] DAQ protocol is deemed to be easier to implement than the phased readout[2] of the Technical Proposal document. The latter has been discarded in LHCb.

- The trigger throttling via the Experiment Control System (ECS) was discarded since the implied latencies (order of 20 ms and more) would have required to implement very large event buffers (expensive dual-port memory).

Pending are decisions on the conventions for error handling, error structure, error types etc. however all of these are believed to be independent on the RU hardware, i.e. part of the VHDL programming domain.

3.2.3. Architecture and operation

Any architecture for the RU that complies with the requirements of the three target applications must include the functional blocks shown in Figure 3-8:

- An **input stage** that merges incoming event fragments from up to sixteen input links, checks integrity and stores them into the sub-event buffer. For unbuffered FEM applications, a direct path to the S-Link output may be provided.
- A **dual-ported sub-event buffer** (SEB) that allows simultaneous operation from both ports (input and output stages) to achieve a high system throughput. This buffer must compensate for fluctuations in the RU output availability and throttle latencies.
- An **output stage** consisting of the **sub-event builder** (SEB) that reads event fragments from the SEB and builds a sub-event that is transmitted either to the **S-Link output** (FEM application) or to the **Network Interface Card** (NIC) output (DAQ RU and VELO application).
- A **TagNet interface**, needed for traffic scheduling and destination allocation in the VELO application.
- An **embedded CPU** to perform monitoring and configuration tasks and to interface the RU to the Experiment Control System.

The maximum throughput in both input and output stages must be equal, as the lower one will limit the RU performance. For a target throughput of 160 MByte/s (DAQ RU) and a conservative estimation of the FPGA operation frequency of 40 MHz[1], 64-bit paths are needed. For the VELO application, the nominal 240 MByte/s requirement also implies 64-bit data paths. Thus, the RU must be designed with 64-bit data paths.

As it is justified in “Proposed input and output link technologies” on page 79, S-Link receiver cards will be used at the RU input stage. These cards follow the CMC standard, and thus a maximum of four cards fit on a 9U board front panel. This is convenient for the DAQ RU and VELO applications (maximum four input links) but not sufficient for the FEM application.

In order to accomplish a multiplexing factor of 16 for the FEM application, a solution is proposed in which the multiplexation is carried out in two stages: first in four-input S-Link mezzanine cards (4 to 1) and then in the RU motherboard (4 to 1). A possible architecture for the

1. A push architecture with round-robin destination assignment.
2. A combined push and pull protocol where the full event is only read after a positive Level-2 decision.
1. In 1998, when the project started, a realistic estimation of the system clock frequency for a 64-bit-data-path logic in the chosen FPGAs was 40 MHz, reaching 60 MHz in 2001. A conservative 40-MHz value is assumed though in this discussion.

4-to-1 multiplexer mezzanine cards is shown in Figure 3-9. Such a card has been designed in the framework of the Readout Unit Project [Bal01].

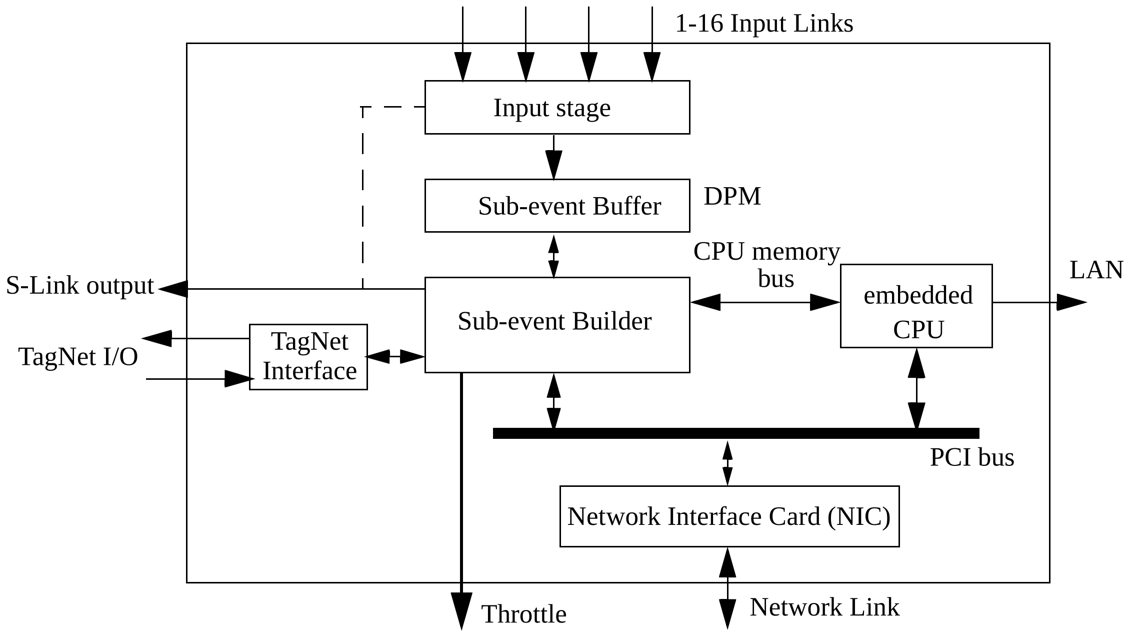


Fig. 3-8. Readout Unit functional blocks.

The CPLDs recover the STF format from the physical layer data and store event fragments in a FIFO memory. An FPGA scans the FIFOs, **concatenates** event fragments with the same event number (one per active input link) and writes them into the on-board RU input FIFOs via the S-Link connectors. The FPGA also emulates the S-Link protocol, thus being fully compatible with the RU hardware. The RU input stage logic will receive then up to four event fragments per S-Link connector and event, compared to the single event fragment received in the DAQ and VELO applications.

At the RU input stage, all FIFOs are read out by the input-stage logic which stores the fragments sequentially in the sub-event buffer (SEB). Pointers to their consecutive storage locations in the SEB are kept in a small directory in the SEB (see “Memory management scheme” on page 61). At the output side of the SEB, the sub-event builder logic scans the descriptor section for entries with the expected DAQ event number. The efficiency and safety of this scheme is enhanced by the fact that incoming DAQ event numbers are monotonically incremented[1]. A set of (between one and four) linked pointers defines a sub-event descriptor which can be transferred by a DMA engine to the readout network.

The SEB buffer is logically operated like a FIFO which is filled from the input stage, and read out concurrently by the output stage. Whilst event fragments are individually coming in, and being queued in the SEB, the oldest events are read out. For outputting sub-events from the SEB, a DMA engine is permanently transferring to the PCI-resident Network Interface (or to S-Link). If the downstream system is free of congestion, the RU logic can autonomously maintain

1. In other LHC experiments like CMS, this basic feature is not implemented, thus requiring more complex sub-event building algorithms.

a N-to-1 data flow between 1 to 16 inputs and one PCI (or S-Link) output. Congestion can be avoided by careful design of the downstream system which is receiving data from the RU. In case of congestion, a feedback signal is returned to the Readout Supervisor to protect the system from a potential buffer overflow (see “Flow control in LHCb” on page 35). During sub-event-building, consistency in the received data fragments can be checked, and any errors can be marked in the error bit fields of the STF.

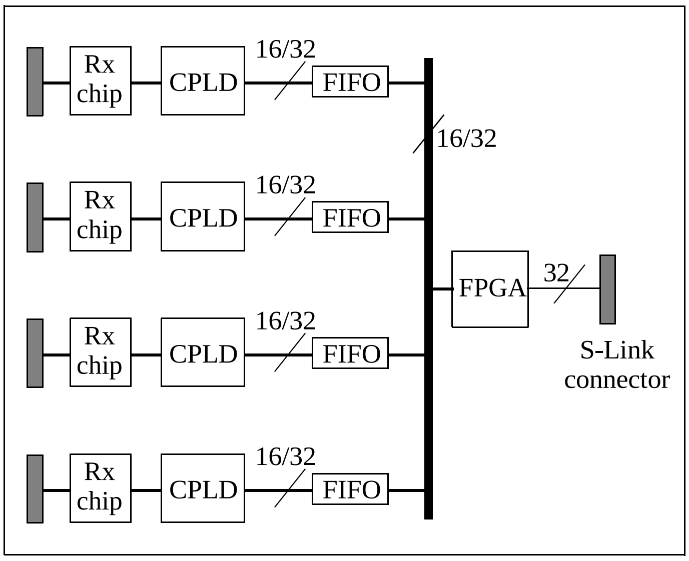


Fig. 3-9. 4-to-1 multiplexing card for FEM application.

There are programmable options for error handling: erroneous data may be passed on, filtered or only counted for error statistics. Any severity level of errors can be reported to the ECS. Finally, a fast reset for SEB buffers is an option which allows to extend the LHCb buffer reset scheme beyond Level-0 and Level-1 to SEB-RESET.

3.3. Input stage

3.3.1. Input stage architectures

Any efficient implementation based on today's microprocessors and FPGAs can be find its equivalent in one of the following three architectures[1], depending on the number of concurrent processing elements in the input stage: one, two or four. The processing elements can be implemented in the same or in separate physical devices and these can be either FPGAs or microprocessors. The implementation is not discussed in this section but the architecture implications on system performance.

Figure 3-10 shows a possible architecture with a single processing element. The design parameters (section 3.2 on page 47) require a 64-bit architecture. Input FIFOs are needed as data arrives asynchronously from the several input links. Incoming data is stored in FIFOs, thus allowing two data merging schemes: FCFS (Fist-Come First-Served) and Polling.

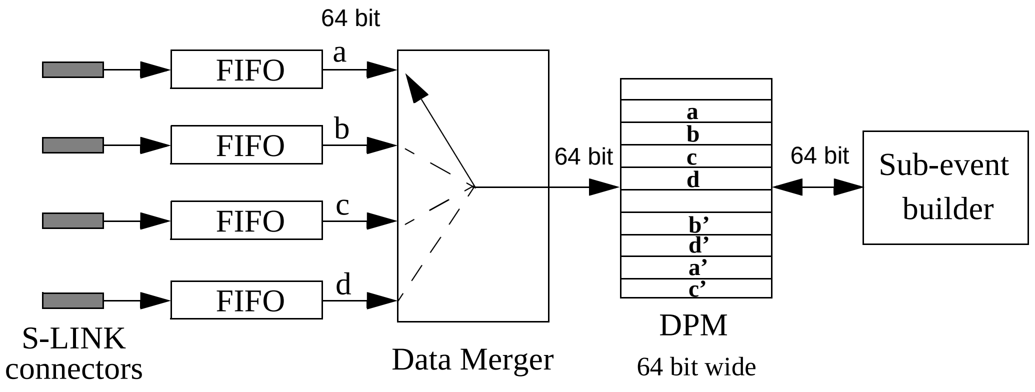


Fig. 3-10. Single processing element in the input stage.

- In the **Polling scheme**, the Data Merger polls input "a" until a frame with the expected event number arrives and then stores the frame in dual-port memory (DPM). Data merger now waits on input "b" until a frame arrives, stores it in DPM and polls input "c". Input "d" is the last one to be polled. As a result, we get the sequence a-b-c-d in memory.
- In the **FCFS scheme**, the first frame that arrives is stored in memory, so we can get different sequences in memory (like b'-d'-a'-c', as shown in Figure 3-10). If two frames arrive at the same time, an arbiter decides which one is read out first. The FCFS scheme has the advantage of requiring smaller FIFO lengths, as in average a frame must wait a shorter time to be read out.

In both cases, a time-out mechanism must be implemented in the input stage to handle missing frame and broken link conditions.

In the scenario depicted in Figure 3-11, each of the two processing elements handles one or two input links. The difference between the polling and FCFS merging schemes in terms of FIFO

1. This statement will become obsolete when an array of processors on a chip be available. At the time of writing this thesis, one of such devices [NP4GS3] has been announced by IBM. Its potential use for a RU implementation is being evaluated at CERN.

length is smaller that in the previous case. This architecture results in two independent input stages working in parallel and thus 32-bit data paths provide the required performance. The sub-event buffer must then be divided into two separate memory banks. The advantage of this architecture resides in the reduction in a factor of two of the dead time implied in the FIFO switching (both at the physical level and in the state machines in the processing elements). The performance increase is marginal in applications with large event sizes, but may be significant in small-event-size ones.

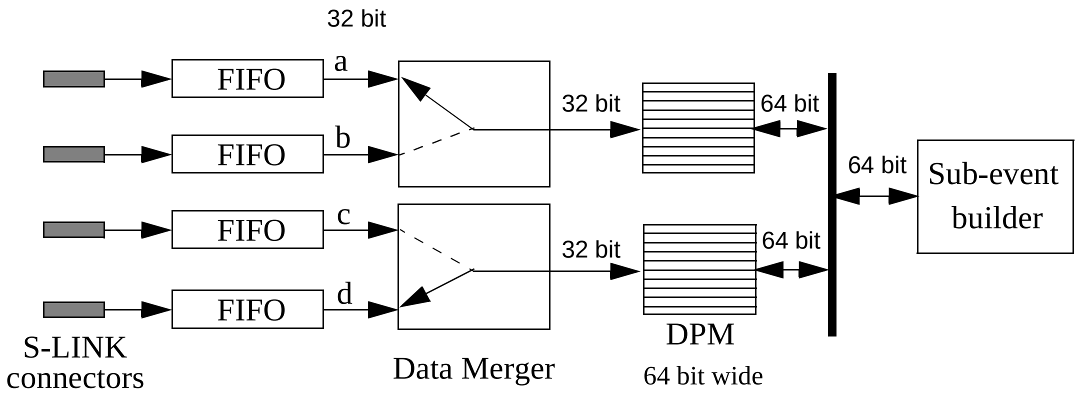


Fig. 3-11. Two processing elements in the input stage.

The four-processing-element architecture shown in Figure 3-12 has the advantages of not requiring input FIFOs as there is a dedicated processing element per input link and thus incoming frames can be analyzed and stored in DPM on-the-fly. The absence of dead cycles implies a higher throughput.

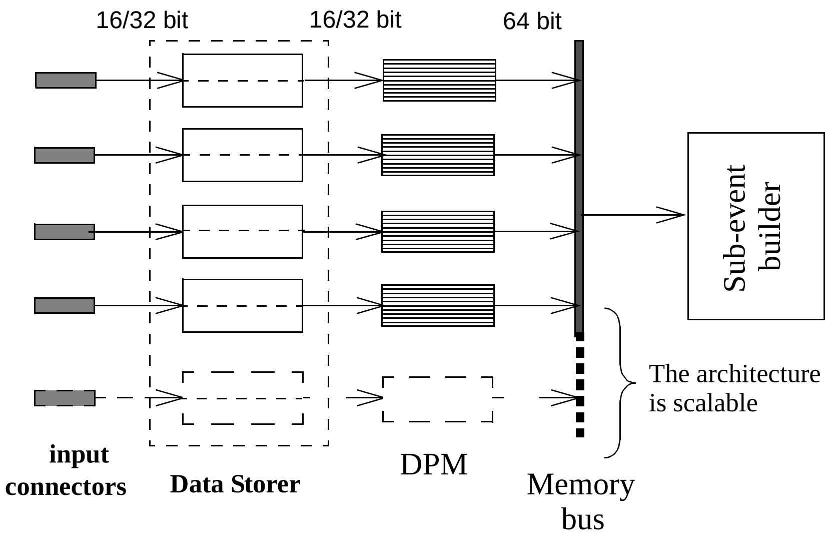


Fig. 3-12. Four processing elements in the input stage.

Which architecture to use depends on the following criteria:

1. **Required input FIFO size**: Worst case corresponds to the polling scheme in the single-processing-element scenario for the DAQ RU application (the one with the largest event fragments). This worst case situation was simulated (see “Modelling and simulation” on page 57), showing that the maximum required size was less than 9 KByte per input link, with an average occupancy of 1.4 KByte. This is a light requirement for commercial devices and thus the selected input stage architecture will not depend on the FIFO size.
2. **Impact on the output stage performance**: Processing elements must store event fragments on separate memories to operate with independence and achieve a high throughput. Two or more processing units sharing the same memory would result in turnaround cycles that would spoil performance. In order to build sub-events, the output stage will have to access as many memory banks as processing elements in the input stage, with the corresponding latencies. The impact on the output stage performance will be discussed in “Output stage requirements” on page 64, but is marginal for FEM and DAQ applications.
3. **Cost**: Using 32-bit wide DPMs and FIFOs, the three options could be implemented (according to bus widths defined in figures 3-10, 3-11 and 3-12) with the number of ICs and relative cost shown in Table 3.6. For the relative cost calculation, DPMs and FIFOs compute as 1 unit and processing elements as 2 units (3 units for the single-processing-element architecture, as it requires at least 64 I/Os more than the other options, and thus a larger physical package).

Table 3.6. Comparison in terms of relative cost and number of ICs

Scenario	FPGAs	FIFOs	DPMs	Nr. ICs	Relative cost
Single processing unit	1	8	4	13	15
Two processing units	2	4	4	10	12
Four processing units	4	0	8	12	16

The cost and number of chips is similar for all three options and thus it will not be taken into account to decide on an architecture.

4. **Input stage performance**: An analytical expression for the maximum input stage throughput as a function of the number of processing elements (*P*), fragment size in bytes (*N*), the FIFO bus width in bytes (*B*), the input stage clock frequency in MHz (f_{clk}) and the number of overhead clock cycles (*d*) imposed by the FIFO read first-word latency, turnaround cycle and dead time due to the algorithm implementation, is given by expression 3.1.

$$Throughput(MByte/s) = \frac{P \cdot N \cdot f_{clk}}{\frac{N}{B} + d} \qquad (3.1)$$

The parameters and resulting throughput for each of the three options is shown in Table 3.7. All three architectures meet or exceed the nominal throughput for VELO, DAQ and FEM applications.

Table 3.7. Input stage throughput[a]

P	*B (byte)*	*Throughput DAQ (MByte/s)*	*Throughput FEM (MByte/s)*	*Throughput VELO (MByte/s)*
1	8	315	301	256
2	4	317	310	284

Table 3.7. Input stage throughput[a]

P	*B (byte)*	*Throughput DAQ (MByte/s)*	*Throughput FEM (MByte/s)*	*Throughput VELO (MByte/s)*
4	4	635	620	568
4	2	318	315	301

a. Assuming $d=2$, $N_{DAQ}=1024$ byte, $N_{FEM}=256$ byte, $N_{VELO}=64$ byte, $f_{clk}=40$ MHz.

In terms of throughput, cost and required input FIFO size, all three options are valid. The single-processing scheme minimizes memory access latencies at the output stage and the two-processing-element scheme also has some advantages (see section 3.6.2). The four-processing-elements solution is discarded due to negative impact on the output stage performance provoked by memory access latencies.

From expression 3.1, assuming $d=2$ and expressing f_{clk} in kilohertzs, the maximum trigger rate as a function of the event fragment size (N) can be obtained from expression 3.2 for the single-processing-element architecture and from expression 3.3 for the two-processing-element scheme.

$$Rate(kHz) = \frac{f_{clk}}{\left(\frac{N}{2}+8\right)} \tag{3.2}$$

$$Rate(kHz) = \frac{f_{clk}}{\left(\frac{N}{2}+4\right)} \tag{3.3}$$

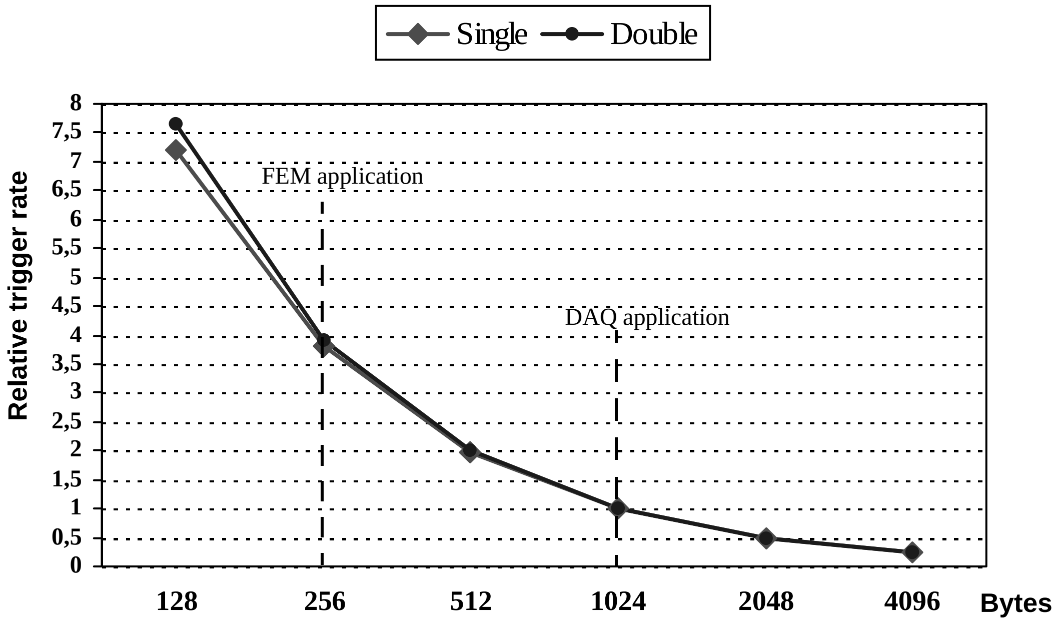


Fig. 3-13. Maximum trigger rate as a function of the event fragment size, normalized to 1 KByte.

As it can be observed in figures 3-13 and 3-14, the difference in performance is negligible for the FEM and DAQ applications (256 byte[1] and 1024 byte event data, respectively). A 10% improvement can be achieved with the two-processing-element architecture for the VELO application (64 byte event fragments). For a 66 MHz operation on the two-processing-element architecture, the VELO application can run up to 1.8 MHz, the DAQ application up to 127 kHz and FEM up to 500 kHz. The strong dependence with the fragment size in the VELO application is in evidence.

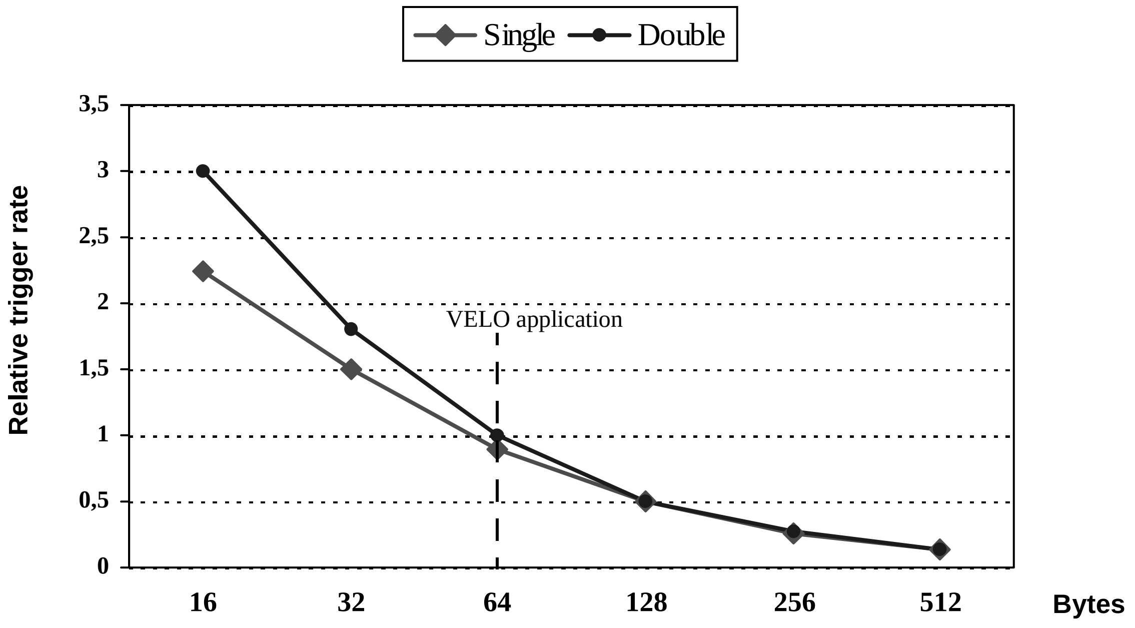


Fig. 3-14. Relative trigger rate for small event fragments, normalized to 64 byte.

3.3.2. Modelling and simulation

A complete slice of the readout system has been modelled and simulated (64 Level-1 front-end sources, 4 FEMs with 4-to-1 multiplexing mezzanine cards and one RU) using Innoveda's VisualHDL for the model description[2] and verification. Cadence's Leapfrog was used for the complete slice simulation as it is much faster than VisualHDL's simulator. The model is based on an existing VHDL library for queue modeling [Mohanty94].

Both the Level-1 trigger and event fragment size are modelled as random generators. The parameters used for these simulations were (see Table 3.8):

- **Level-1 trigger generator**: Log-normal distribution[3].
- **Event fragment size at the Level-1 source**: Results from multiplying two random variables: *base_length* and *modifier_lengh*. The former is common for all Level-1 source instances in

1. Considering the four event fragments received from the 4-to-1 multiplexer card as a unit.
2. Single-processing-element input and output stages.
3. A poisson distribution is preferable, but the available VHDL model for poisson distributions showed frequent convergence problems.

the model, while the latter is different for each instance, thus resulting in a more realistic spread of data block sizes for each FEM instance.

- **Link and NIC latencies**: have not been considered as they are implementation dependent.

Table 3.8. Level-1 trigger rate and block size

	Level-1 trigger generator	**Fragment size at the Level-1 source**[a]	
	Level-1 trigger (µs)	*base_length* (bytes)	*modifier_length* (bytes)
Type of distribution	log normal	log normal	gaussian
Mean value	25	64	1
Std. deviation	12	32	0.5

a.Fragment size is *base_length*modifier_length*

Processing times for each of the three levels of multiplexing (4-to-1 multiplexer, FEM motherboard and RU motherboard) depend on FIFO bus width, logic clock speed and fragment size, according to the expression 3.4, where size is expressed in bytes and clock speed in MHz.

$$ProcessingTime(\mu s) = \frac{FragmentSize}{FIFObusWidth \cdot f_{clk}} \tag{3.4}$$

The parameters for each level in the hierarchy are shown in Table 3.9.

Table 3.9. Multiplexing stage parameters

Mux level	**FIFO read bus size (bytes)**	**clock speed (MHz)**	**Available bandwidth (MByte/s)**
4-to-1 mux	2	10	20
FEM motherboard	4	20	80
RU motherboard	8	40	320

Figure 3-15 shows that the average latency between data arriving at FEM inputs and data coming out from the RU output is close to two triggers (63 µs), with a maximum of about 200 µs (eight triggers). Link latencies need to be added for more realistic results.

Figure 3-16 is an histogram of the DAQ RU motherboard’s FIFO occupancy, showing that the average value is approximately1400 bytes and that the maximum occupancy is below 9 KByte. This result poses a light requirement to the input FIFO size, which will be ultimately determined by the maximum fragment size[1] rather than by normal operation parameters.

It can be concluded that the input stage is technologically feasible and several architectures are possible.

1. Non-zero-suppressed data will be sent across the DAQ system during calibration runs, resulting in large event fragments at a reduced trigger rate.

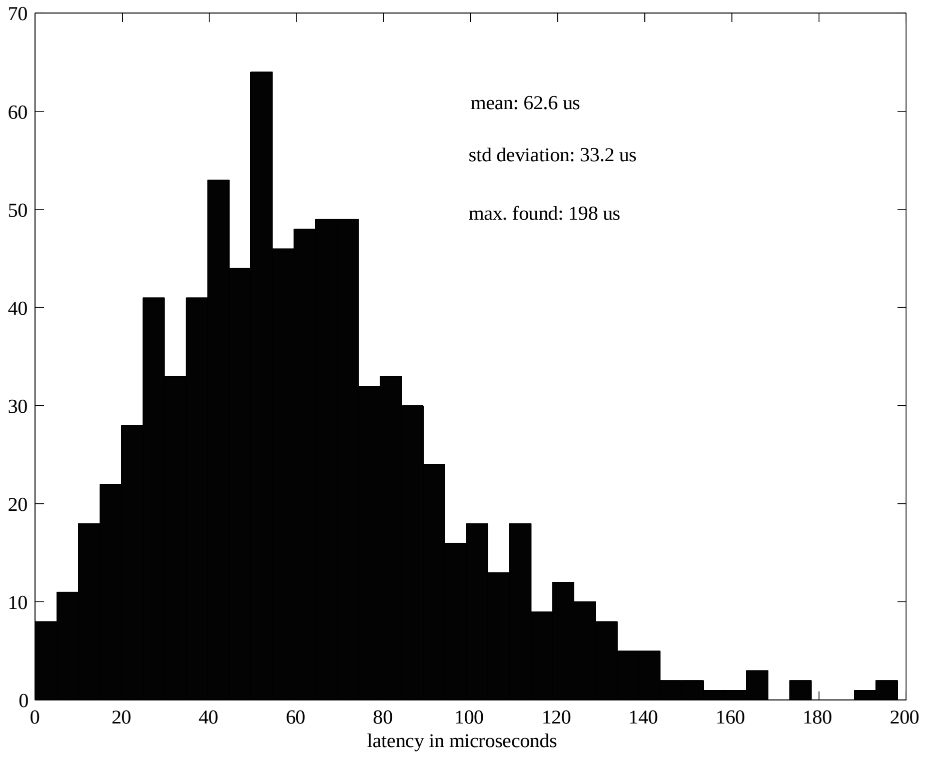


Fig. 3-15. Latency from Level-1 output to DAQ-RU input.

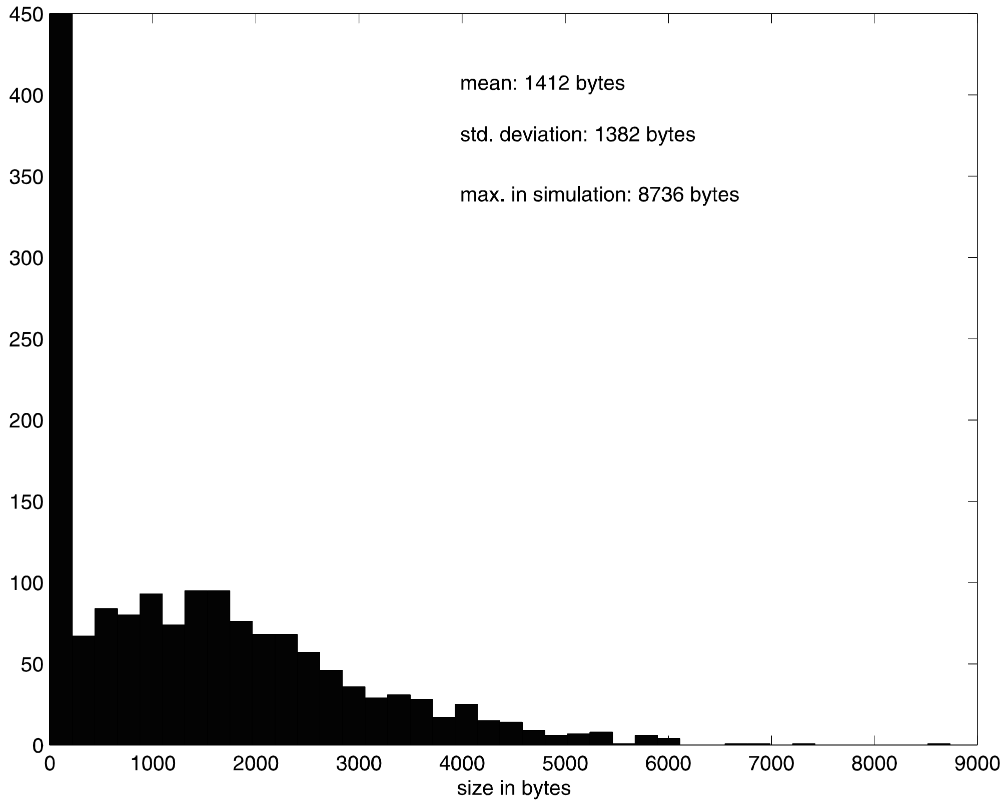


Fig. 3-16. DAQ-RU FIFO size histogram (worst case).

3.4. Sub-event buffer

The sub-event buffer feasibility is determined by the required throughput and memory size and the need to access the buffer simultaneously from both ports (input and output stages). Two different implementations are possible: (1) emulate a dual-port memory with SDRAM and control logic, and (2) use true dual-port memories.

True dual-port memories are expensive and their density of integration is much lower than SDRAMs. As a result, DPM emulation with SDRAM is not a rare option. Some attempts have been carried out in this direction. In [Antchev97] a PMC card is described in which one FPGA is used as control logic to arbitrate the access to the SDRAM from two ports, whilst two bi-directional FIFOs act as intermediate buffers. This implementation allows up to 33-MHz operation on 32-bit buses, resulting in a raw bandwidth per port of 133 MByte/s. A more sophisticated approach, described in [Gigi99], requires a long size PCI board with four 64-bit 66-MHz PCI bridges, one SDRAM controller, one FPGA and other interface logic in order to allow 200 MByte/s bandwidth per port.

In the case of the Readout Unit, as the buffering requirements are not too high (in the order of 2 MByte) and a sustainable integrated bandwidth in excess of 500 MByte/s[1] is beyond what has been achieved so far in DPM emulation with SDRAM, a true dual-port buffer is preferred[2].

The buffering requirements were determined by the throttle latency across the ECS system[3] before a fast throttle implementation was decided. As justified in “Flow control in LHCb” on page 35, 2.5 MBytes are needed to compensate for the fast throttle latency. This buffering is partly provided by the input FIFOs (up to half megabyte for 128-KByte per input link[4]) and the rest by the sub-event buffer.

At the time of writing this thesis, the largest DPM integrated circuit in market is a half megabyte chip [IDT70V3599]. This allows up to 2 MByte of buffering space in a compact four-chip configuration (see Figure 5-12 on page 114). If the currently 2 MByte buffer implemented on the RU-II module is found to be insufficient, two complementary actions can be taken:

1. **Increase the sub-event buffer size**. Larger buffers can be achieved at a higher cost and board area by using six or eight memory chips on BGA packages, which would allow up to 4 MByte on-board buffer space. This option implies a module redesign.
2. **Reduce the multiplexing factor in the FEMs**. More FEMs and RUs are needed then, but the buffering requirements decrease. This would affect the DAQ switch implementation.

Due to the high cost and low level of integration of the dual-port memory compared to single-port SDRAM memory, an efficient memory usage is required. This discards fixed-length, fixed-position buffers for event fragment storage. An efficient memory management scheme is proposed in the following section.

1. In the VELO application, the average throughput per port is 240 MByte/s.
2. In fact, four chips are enough to implement a 2-MByte true DPM in the Readout Unit II module.
3. Up to 20 ms latency, which implies 3.2 MByte of buffering space for the DAQ RU application.
4. Required for non-zero-suppressed data in calibration runs.

3.4.1. Memory management scheme

In the proposed scheme, most of the buffer capacity is used for storing sub-event data; only a small part is used as a pointer directory. Each DPM bank is logically divided into two circular buffers: a Directory Block (DYB) and a Data Block (DB). There is also a special memory position used for buffer occupancy monitoring called Mailbox (Figure 3-17). By means of entries in the Directory Block pointing to event fragments in the Data Block, variable-length variable-position buffers can be implemented with economic memory usage.

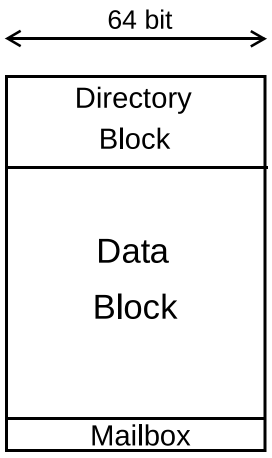


Fig. 3-17. Logic structure of the sub-event buffer.

A possible format for the directory entry in the DAQ and FEM applications is shown in Figure 3-18. It is a 64-bit word containing the following fields:

- **Event Number**: In the first code implementation on the RU-II, a fragment is assigned the next free entry in the Directory Block, using a 24-bit DAQ event number in order to accommodate a descriptor entry in 64 bits. If a 28-bit number is required[1], a 128-bit descriptor entry can be used. Nevertheless, the requirement of 28-bit DAQ event numbers is not justified.
- **Pointer**: Event fragment start address in the Data Block. Eighteen bits are needed to address 64-bit words in a 2-MByte SEB.
- **Length**: Size of the fragment in 32-bit words. In this example (14-bit field), a maximum 128 KByte sub-event is allowed (32 times larger than the nominal sub-event in the DAQ RU application).
- **Flags**: Includes errors reported by the data source, transmission errors and other kind of RU-specific errors. It also indicates if an Error Block from the STF trailer is stored in the Data Block following the fragment.

1. Some members of the LHCb Collaboration suggested to use a 28-bit number. At a 40 kHz trigger rate, a 20-bit number would already allow more than 26 s without repeating a number. This is clearly much more than the life time of an event in the DAQ system.

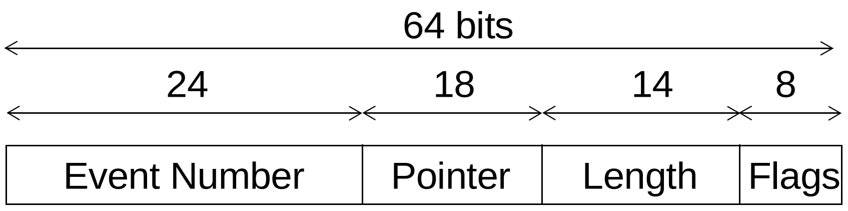


Fig. 3-18. Entry in the Directory Block.

Using this directory entry structure as an example, the memory looks like shown in Figure 3-19. The final implementation is obviously dependent on the chosen input stage architecture. Error blocks, as defined in the STF, can also be supported. In Figure 3-19, the entry in the Directory Block points to an error block descriptor, basically containing the length of the error block stored after the event fragment. A value of zero indicates absence of error block.

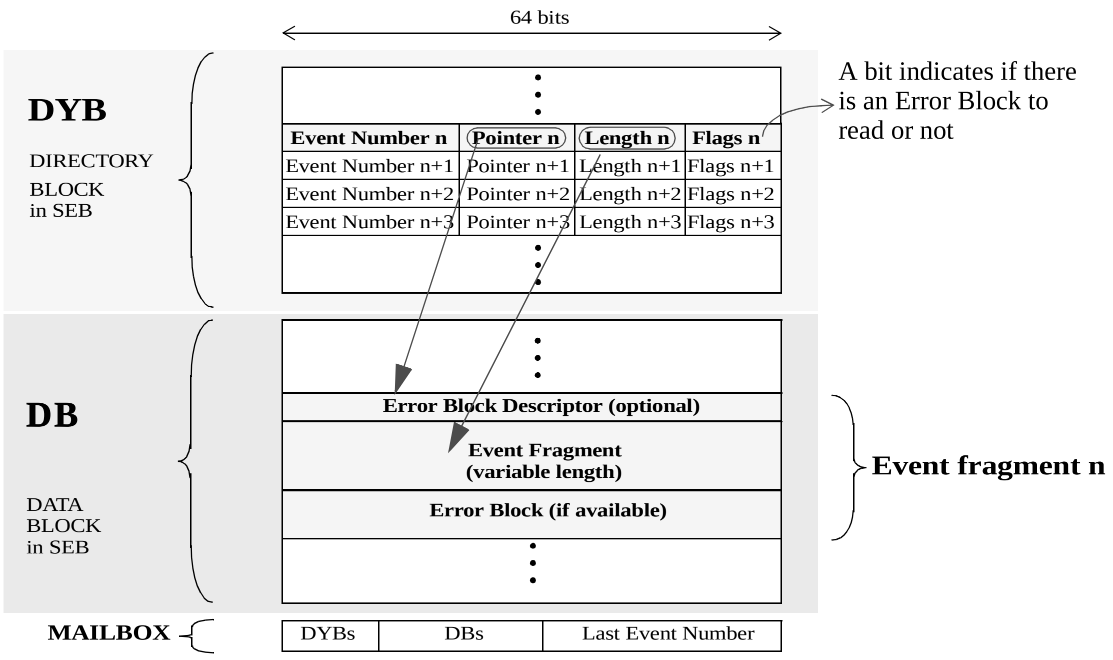


Fig. 3-19. Event fragment storage in SEB memory.

3.4.2. *Buffer occupancy monitoring*

The SEM logic keeps two pointers for the next free position in the Data Block (DB) and in the Directory Block (DYB): pointers **dbs** and **debs** respectively, where “s” stands for SEM. Both are pointers to circular buffers.

1. When a new event fragment is to be written into the DB, it is stored starting in the current value of **dbs**. After storage, **dbs** points to the next free position in the DB.
2. The corresponding directory entry in the DYB is written to the address pointed by **debs**, which is incremented afterwards.

3. The SEM logic writes **dbs**, **debs** and the event number in the **MAILBOX** (a dedicated memory position in the SEB) and toggles a I/O line (also called MAILBOX) to warn the output stage logic that a new sub-event has been written into memory.

The output stage logic also keeps two pointers: **dbe** and **debe** for the next locations to read from DB and DYB, which are treated as circular buffers by the output stage logic. Each time the MAILBOX signal toggles, the output stage reads the **MAILBOX** and recalculates the SEB buffer occupancy. Above a certain programmable threshold, the signal DPMFULL# is asserted to warn the SEM of this condition. If DPMFULL# or FIFO almost full conditions are detected by the SEM logic, the fast throttle output signal is asserted at the RU's front-panel connector.

In more detail, the output stage logic has to monitor the SEB occupancy in order to detect the following conditions: empty, full beyond the "low watermark" and "high watermark" thresholds and completely full. As both Directory Block and Data Block are logical circular buffers, the input stage pointers can be greater, equal o lower than the output stage pointers. An example is depicted in Figure 3-20, for a SEB total size of 256 positions. The DYB size is 32 and DB size is 256-32-1=223. The pointers point to the next position to be read (OP) and the next position to be written (IP).

In case *A*, $OP > IP$. The free space in the DB corresponds to the shaded area and can be computed as: $OP - IP = 18$.

In case $B, IP > OP$ thus the free space in DB (shaded area) can be computed as:

$$256 - 33 - (IP - OP) = (256 + OP - IP) - 33 = 201 .$$

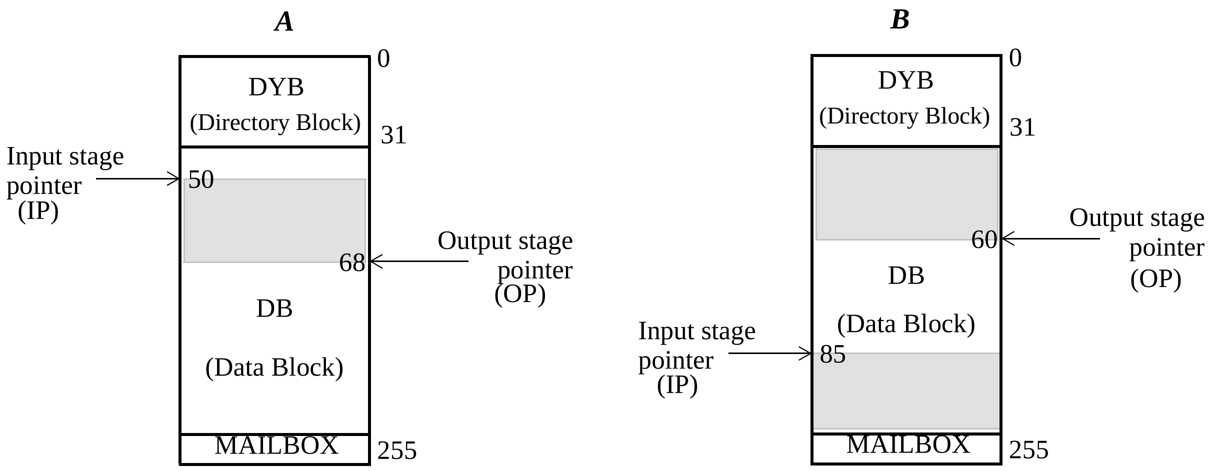


Fig. 3-20. Buffer occupancy monitoring.

Additions and substractions are performed in two's complement. For 8-bit wide addresses (as memory size is 256 positions), the expression $(256 + OP - IP)$, when $IP > OP$, equals to $OP - IP$. Then, $OP - IP$ can be computed for both cases and, if $IP > OP$ (which is detected by the carry bit resulting of $OP - IP$), 33 in this example, is substracted from the result. In the case of the DYB, as its size is a power of two, there is no need to perform any subtraction and the free space in DYB is always $OP - IP$.

The described mechanism allows to easily calculate the free memory in the circular buffers. An example of implementation in VHDL is shown in Figure 3-21. As the carry bit is not accessible, two’s complement operations are performed with N+1 bits and the most significant bit is used as the sign bit.

```
architecture simple of count_blk is

 signal dbp_int,free_dbp_int:std_logic_vector(14
downto 0);
 signal mailbox_p:std_logic_vector(8 downto 0);
 signal mailbox_d:std_logic_vector(14 downto 0);
 signal pbp_int,free_pbp_int: std_logic_vector(8
downto 0);
...
begin
...
-------------------------------------------
substractors:process(reset_counters,clk)
 variable free_dbp1_int:std_logic_vector(15 downto
0);
 variable free_dbp2_int:std_logic_vector(14 downto
0);
begin
...
  free_pbp_int<=pbp_int-mailbox_p;
  if(free_dbp1_int(15)='1') then
 free_dbp_int<=free_dbp2_int;
  else
 free_dbp_int<=free_dbp1_int(14 downto 0);
  end if;
  free_dbp2_int:=free_dbp1_int(14 downto 0)-
"1000000001"; --513
  free_dbp1_int:=('0'&dbp_int)-('0'&mailbox_d);
...
end process substractors;
-------------------------------------------
...
end count_blk;
```

Fig. 3-21. VHDL code for buffer fill state monitoring.

3.5. Output stage requirements

In the VELO and DAQ RU applications, the sub-event builder element is interfaced between the sub-event buffer (SEB) and the Network Interface Card (NIC), as depicted in Figure 3-8. As it is justified in “Proposed input and output link technologies” on page 79, the use of commercially available NICs in a PMC form factor will allow to follow trends in technology during the coming years, will make it unnecessary to carry out complex custom developments and will ease maintenance. On the other hand, a configuration agent is needed on the PCI bus to initialize and configure the PCI devices (including the NIC). This implies that the Monitoring and Control Unit (MCU) has a PCI interface. The MCU could also embed other PCI functions such as the arbiter, simplifying the output stage design.

In the FEM application, the sub-event builder element outputs to an S-Link transmitter card via a 16-bit bus (a 32- or 64-bit bus would be an overdesign as only a 40-MByte/s throughput is required). A block diagram of the RU output stage, including also the MCU, is sketched in Figure 3-22.

The sub-event builder can be implemented either in FPGA logic or in a microprocessor, in both cases with or without the assistance of a DMA controller. The FPGA solution has the advantages of yielding higher performance (if the algorithms are highly parallelizable and only integer operations are involved), the high I/O pin count, the direct connection (i.e., without any glue logic) to the SEB and S-Link output and the possibility to embed the PCI interface. Several PCI interfaces for FPGAs are commercially available. For these reasons, the output stage logic will be implemented on FPGAs.

The nominal bandwidth for the DAQ RU application (160 MByte/s) requires at least 64-bit 33-MHz PCI implementations (264 MByte/s raw bandwidth), whilst the VELO application needs a 64-bit 66-MHz PCI bus (528 MByte/s raw bandwidth).

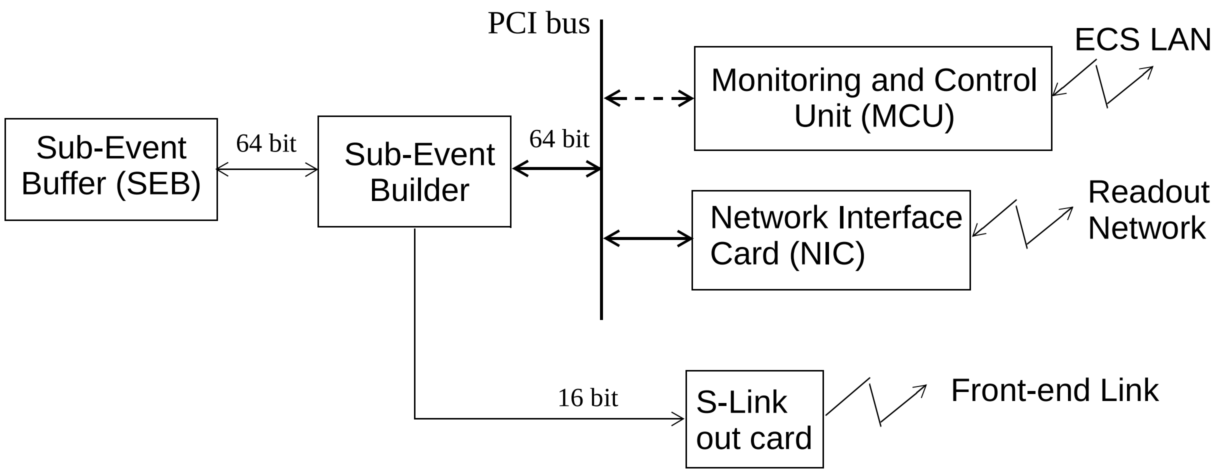


Fig. 3-22. Output stage block diagram.

3.5.1. FEM application requirements

The performance is limited by the FPGA clock frequency and the 16-bit output bus width. A first non-optimized implementation of the sub-event building algorithms in the RU-II module reached 45 MHz clock frequency, thus allowing for 90 MByte/s raw bandwidth over a 16-bit bus to the S-Link output.

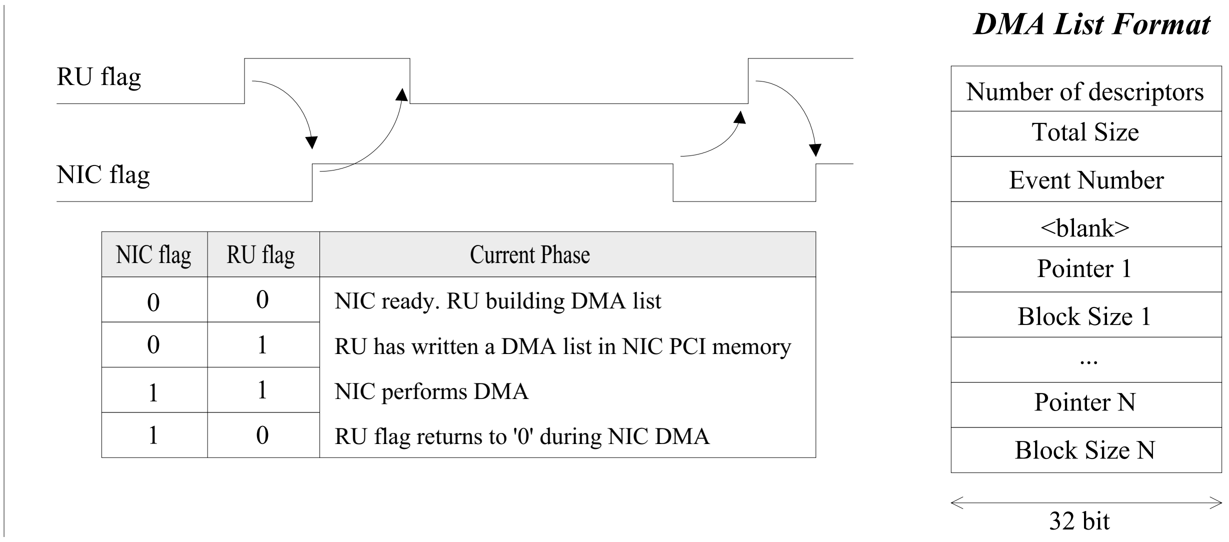
The first implemented algorithm on the RU-II module consumed 35 cycles for FPGAs synchronization and header/trailer generation, 8 cycles to send header and trailer to S-Link and 512 cycles to send the payload to S-Link (1 KByte). This makes a total of 555 cycles per event or 81 kHz trigger rate for a 45-MHz FPGA operation. Better results (100 kHz trigger rate) are expected with an optimized implementation. The maximum rate for an arbitrary sub-event size (S) in bytes is given by expression 3.5. This result implies that a 16-bit bus is enough and the basic architecture depicted in Figure 3-22 exceeds the nominal requirements by a factor of two.

$$f_{trigger} = \frac{1}{t_{event}} = \frac{f_{clk}}{43 + Size/2} \tag{3.5}$$

3.5.2. DAQ RU application requirements

The output stage must support both pull and push protocols between the sub-event builder and the NIC. In a **pull scheme**, an intelligent PCI-master-capable NIC pulls data from the RU via

DMA read operations on the PCI bus (this is the case of commercial Gigabit Ethernet NIC implementations preferred for the DAQ RU application). As a more efficient approach, in the **push scheme** event data are written (pushed) by the sub-event builder directly into the NIC buffers (this is the case of SCI, preferred for the Level-1 VELO application).



NIC flag	RU flag	Current Phase
0	0	NIC ready. RU building DMA list
0	1	RU has written a DMA list in NIC PCI memory
1	1	NIC performs DMA
1	0	RU flag returns to '0' during NIC DMA

Fig. 3-23. Protocol between the sub-event builder and the NIC for the DAQ RU application.

The DAQ RU application will implement a pull scheme, as the extra overhead implied in the RU-NIC protocol does not affect performance due to the large sub-event size, benefiting from traffic-shaping and other complex network protocols which can be implemented in an intelligent NIC. The proposed readout protocol (depicted in Figure 3-23) was agreeded in November 2000 with other members from the LHCb DAQ group and consists of the following steps:

1. The NIC indicates its availability to process a new sub-event by setting to zero a flag in a PCI-accessible status register in the NIC (NIC flag in Figure 3-22).
2. The sub-event builder polls this NIC flag until the NIC is ready.
3. The sub-event builder constructs a DMA list according to the format[1] shown in Figure 3-23, sends it to the NIC via a PCI write transaction and signals the action by setting high a flag in the NIC's status register (RU flag).
4. The NIC acknowledges by toggling the NIC flag and starts polling the sub-event via PCI read transactions.
5. The sub-event builder toggles the RU flag in response, and waits on the NIC flag until the DMA operation is completed and the NIC ready for a new sub-event.
6. The sub-event builder frees the memory and jumps to step three in this list.

This algorithm can be parallelized in order to reduce the overhead, as suggested in Figure 3-24. Nevertheless, a gigabit technology (i.e., 1 ns per bit, or 125 MByte/s) cannot transmit a 4 KByte sub event in less than $4096 \cdot 8 \cdot 1ns \approx 32, 8\mu s$, which limits the maximum trigger rate to 30.5 kHz. Thus, a multiplexing factor of three or less is required to achieve the nominal 40 kHz rate. Under these circumstances, the few overhead clock cycles implied in the protocol are negligible

1. Note that this format assumed that the NIC supports scatter-gather DMA.

and a sequential algorithm would simplify the sub-event builder implementation. The apparent mismatch between the achievable performance and the requirements can be avoided either by using a faster link technology or by scaling up the number of RUs in the system. The latter implies that a larger number of RUs will be connected to a larger switch, increasing the system cost, in order to keep the currently preferred Gigabit Ethernet technology choice[1]. The use of higher-bandwidth technologies in the context of the LHCb DAQ system is discussed in [LHCb98-030].

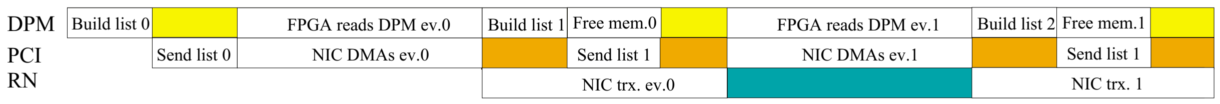


Fig. 3-24. Parallelized sub-event building protocol for the DAQ RU application.

Performance degradation occurs if the NIC is not able to pipeline sub-event operations (i.e., read one sub-event from the sub-event buffer at the same time a second event is being sent to the readout network) and then the duration of the DMA operation in the PCI bus has to be added to the 32.8 µs. This time, t_{DMA}, for a pull protocol, a sub-event size of E bytes and a 64-bit PCI bus clock period T_{clk}, is given by expression 3.6. One PCI turnaround cycle, one address phase, 15 wait states introduced by the FPGA when responding to a read operation[2] and four latency cycles for accessing the DPM (making a total of 21 cycles) have been considered for the PCI read transactions. This time[3] is 16 µs and would limit (together with the 32.8 µs transmission latency) the trigger rate to roughly 20 kHz.

From expression 3.6, the maximum trigger rate for a 64-bit 33-MHz PCI bus and a 4-KByte sub-event is 62.5 kHz. This required a 2-Gbit/s Gigabit Ethernet implementation.

$$t_{DMA} = (21 + E/8) \cdot T_{clk} \tag{3.6}$$

3.5.3. *Level-1 Trigger VELO application requirements*

In the VELO application, a new sub-event is built upon reception of a token from the TagNet bus. Each RU has a token input and a token output, and the RUs are interconnected forming a closed daisy chain including a readout supervisor that generates the tokens. When all RUs in the chain have built and sent its sub-event, the token finally reaches the readout supervisor.

A “push” protocol is assumed as it is the working mode of the commercially available SCI NICs. It was observed with a PCI bus analyzer that no wait states are introduced by the FPGA when performing PCI master write operation and that tested NICs add between three and seven wait states when responding to a target write. Additionally, the embedded PCI interface in the FPGAs inserts ten empty cycles between transactions (see Figure 6-14 on page 141). For the

1. The widespread of Gigabit Ethernet in industry implies lower costs per Gbit/s than other solutions. LHCb has decided to adopt this technology for the time being and wait for higher-throughput derivatives.
2. This latency has been measured on Lucent Technologies’ Orca3TP12 and Orca3LP26B devices. Similar latencies can be expected for other FPGA vendors. See “PCI subsystem test” on page 134.
3. For a 64-bit 33-MHz PCI operation and a sub-event size of 4 KByte.

basic architecture in Figure 3-22, an average event fragment size of 64 bytes and a multiplexing factor of four, a total of thirty-two 64-bit PCI data phases are needed.

The absolute minimum number of cycles required to transfer a complete sub-event to the NIC is then 50 cycles (taking into account the 32 data phases, 7 cycles for NIC latency, one address phase and 10 empty cycles between transactions inserted by the PCI interface). This implies that FPGA and PCI clock frequencies beyond 50 MHz are required to achieve the nominal 1-MHz rate. As a result, a 64-bit 66-MHz PCI bus must be implemented[1]. The maximum trigger rate for an event fragment payload size of *S* byte, a PCI clock frequency f_{clk} and a NIC target latency of *lat* cycles is given by expression 3.7.

$$f_{trigger} = \frac{f_{clk}}{11 + lat + S/8} \tag{3.7}$$

For 64-byte event fragments, this results in a theoretical maximum 64% PCI bus efficiency (Figure 3-25). This sets a performance limit in the PCI bus that can only be superseded by using PCI-X at frequencies between 66 MHz and 133 MHz. This option is not feasible for the time being due to the lack of commercially available PCI-X network interface cards.

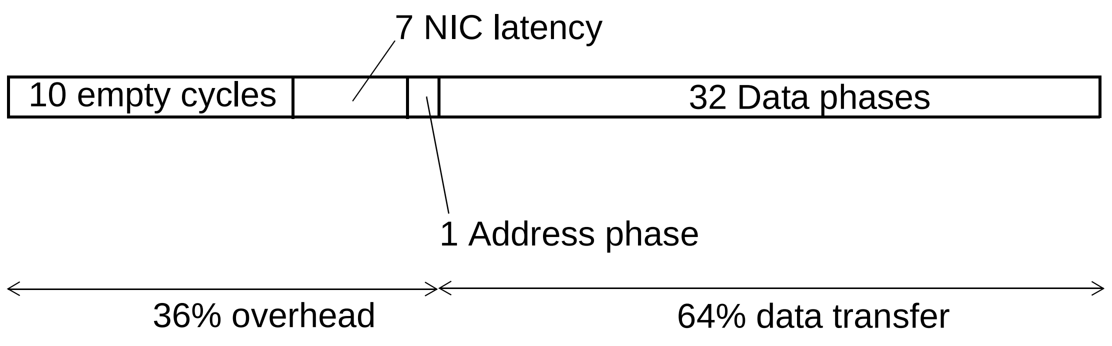


Fig. 3-25. PCI bus efficiency for VELO application.

3.6. Output stage architectures

Besides a single-processing-element output stage architecture (implemented in the first RU prototype) depicted in figures 3-22 and 3-26, a two-processing-element scheme (as implemented in the RU-II module) will be also discussed. In the latter scheme, a parallel operation of two FPGAs on the PCI bus can yield a higher performance, as suggested in [Walsch01].

3.6.1. Single-processing-element output stage

In Figure 3-26, each processing element in the input stage (depending on the chosen architecture) combines the incoming frame payloads into a single block and writes an entry in the Descriptor Block, as described in “Memory management scheme” on page 61. To build a

1. Any frequency in the range 33-66 MHz is considered in the PCI specification as 66-MHz mode PCI.

sub-event, the combined payloads have to be encapsulated in a frame according to the STF convention. The maximum performance for this architecture is limited by expression 3.7. In a two-processing-element input stage scheme, performance would drop for small-fragment applications, as a result of the additional overhead implied in switching between memories during the sub-event building process.

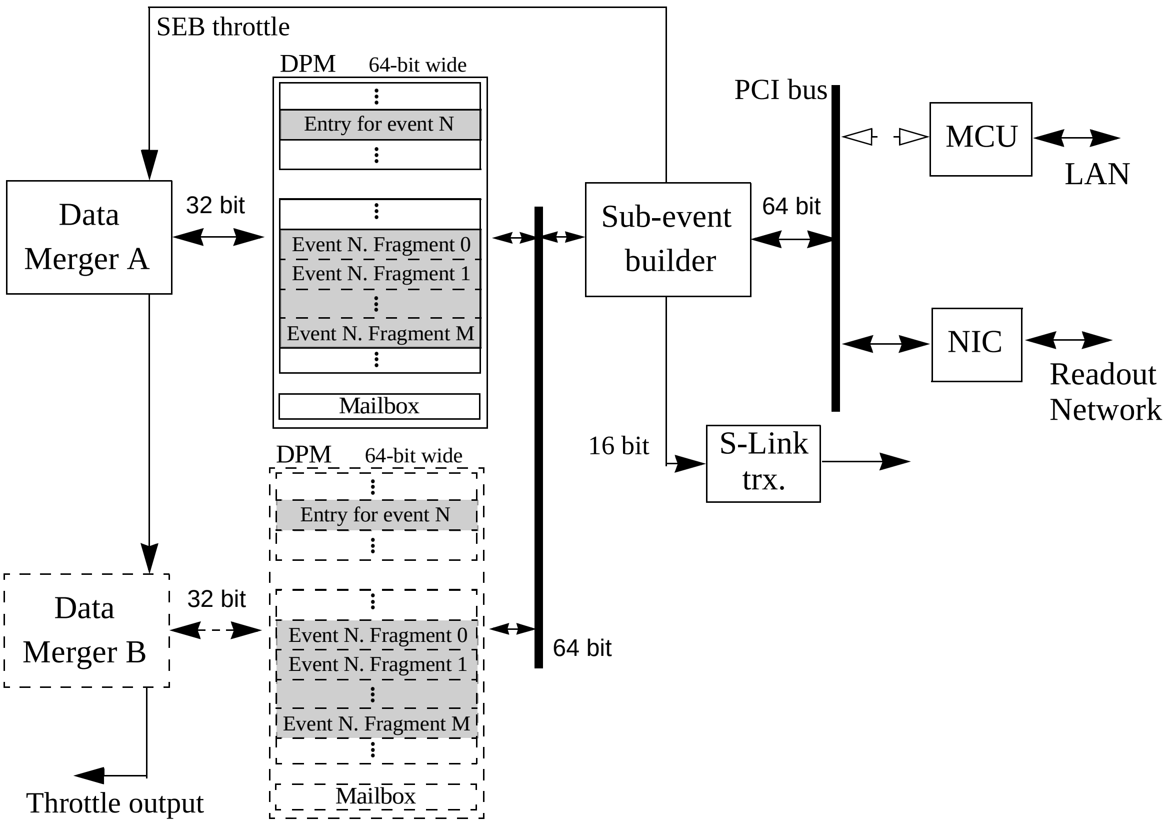


Fig. 3-26. Output stage architecture for a single- or double-processing-element input stage.

3.6.2. Tandem-FPGA output stage

The output stage that corresponds to a two-processing-element input stage architecture (as implemented in the Readout Unit II module) is shown in Figure 3-27. Data mergers store event-fragments in separate memories. The two sub-event builder elements can synchronize their operation via a 32-bit bus as suggested in “High bandwidth, tandem PCI master operation for the Level-1 VELO application” on page 119.

The single-and double-processing-element architectures can be compared in terms of maximum trigger rate for the VELO application (multiplexing factor of four, 64-byte event fragments and 66-MHz PCI operation) as a function of the target (NIC) latency. The maximum trigger frequency for both architectures for a given event fragment size *N* in bytes, trigger latency *lat* and PCI bus frequency f_{clk}, is given by expressions 3.8 and 3.9. The two curves cross at $lat = 7$ (see Figure 3-28), which is exactly the case of the new PCI-to-SCI adapter cards from Dolphin (LC3 link controller and PSB66 bridge chip) tested in [Walsch01].

$$f_{single} = \frac{f_{clk}}{\frac{N}{2} + 11 + lat} \tag{3.8}$$

$$f_{double} = \frac{f_{clk}}{\frac{N}{2} + 4 + 2 \cdot lat} \tag{3.9}$$

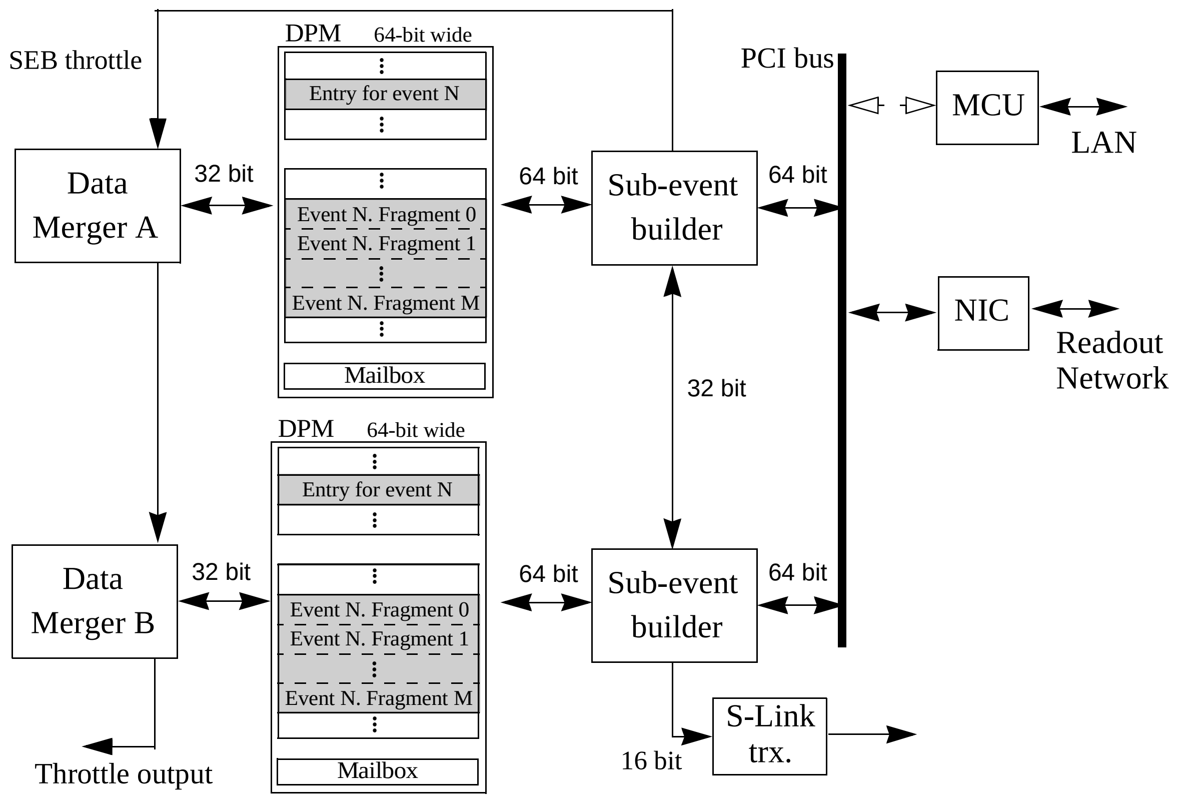


Fig. 3-27. Two-processing-element output stage architecture.

This result implies that, considering the PCI bus efficiency only, the two architectures yield the same performance for the VELO application.

The advantage of the two-processing-element scheme resides in the possibility to neglect the overhead provoked by directory reading and housekeeping tasks. While one FPGA is flushing its internal PCI FIFOs (i.e., performing a long burst PCI transaction) it can simultaneously perform housekeeping tasks (like checking free space in memory) and read descriptors form memory. At the same time, the other FPGA can fill its internal PCI FIFO with the next event fragment (or a fraction of it, depending on the fragment size) [Tol01-1].

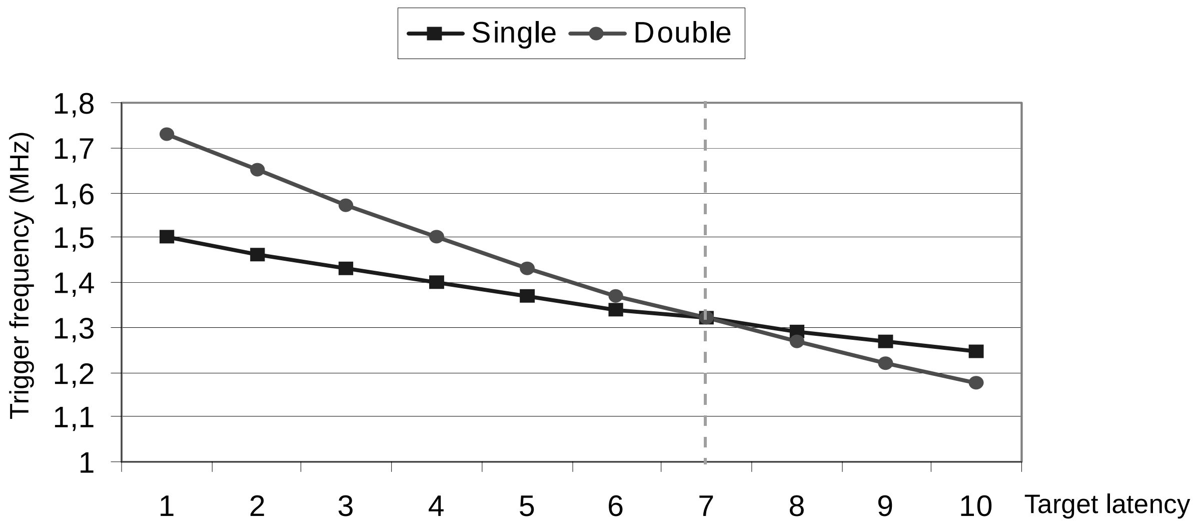


Fig. 3-28. Maximum trigger rate as a function of the NIC latency for the two output-stage architectures.

The following figure illustrates this concept applied to the VELO application, where two nominal-size event fragments fit in the internal FIFO of the RU-II's output stage FPGAs[1] and thus a single block transaction form memory is normally needed.

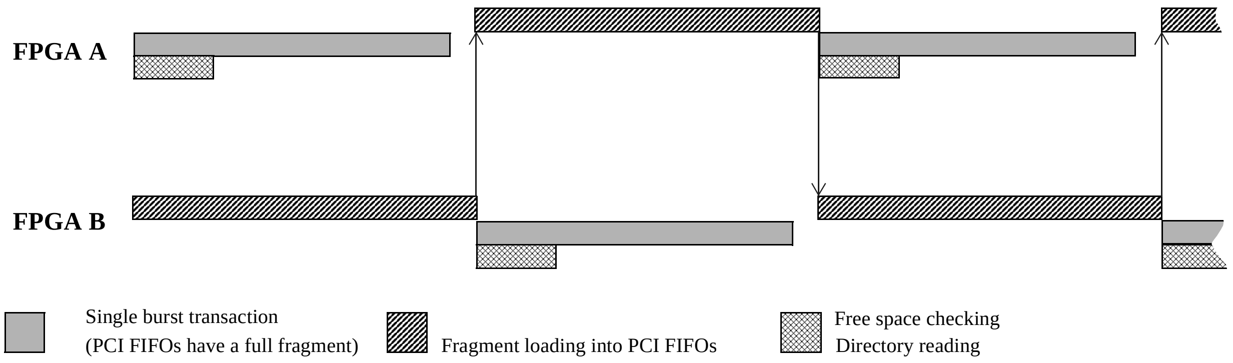


Fig. 3-29. Tandem operation of the two FPGAs.

3.7. Overall module performance

The 64-bit RU hardware architecture allows a 528 Mbyte/s raw bandwidth at 66-MHz. Table 3.10 compares the nominal requirements with the RU estimated performance for the three target applications. In the least demanding application (FEM) the output stage limits the trigger rate to 100 kHz.

1. 256-byte internal PCI FIFOs are provided in Lucent Technologies' OR3LP26 FPGA.

The DAQ application performance is limited by the Gigabit Ethernet network technology and the use of “pull” protocols. Network technologies with higher bandwidth and 66-MHz PCI bus operation at the RU’s output stage would improve the maximum trigger rate for this application.

Table 3.10. RU-II overall performance

Scenario	Sub-event size	Nominal rate	Maximum input stage rate	Maximum output stage rate	Max. Achievable throughput	Required throughput
FEM[a]	1 KByte	40 kHz	460 kHz	100 kHz	100 MByte / s	40 MByte / s
DAQ[b]	4 KByte	40 kHz	117 kHz	62.5 kHz	250 MByte / s	160 MByte / s
VELO[c]	0.25 KByte	1 MHz	2.64 MHz	1.3 MHz	333 MByte / s	240 MByte / s

a. For a 50-MHz FPGA operation.

b. Assuming “pull” protocol and an intelligent 2-Gbit/s NIC, 33-MHz PCI and 50-MHz FPGA operation.

c. For 66 MHz PCI and FPGA clock frequencies.

The maximum rate for the VELO application is limited to 1.3 MHz due to the 7-cycle NIC target latency of current PCI-to-SCI network adapters (Figure 3-28). Input and output stage performance for all architectures, as a function of the fragment size, can be approximated to expressions 3.3 and 3.9 (see Figure 3-30).

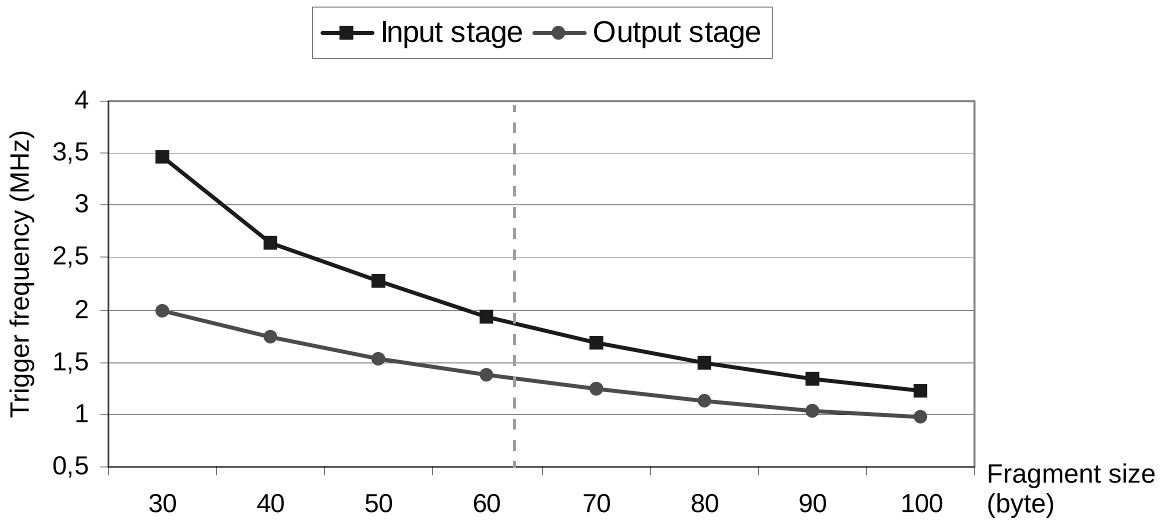


Fig. 3-30. Overall performance for input and output stages.

As it has been demonstrated along this chapter, a different module architecture would not make a significant difference and that the Readout Unit meets the requirements of the three target applications.

3.7.1. Scalability and performance upgrade

Scalability in terms of higher multiplexing factor is not an issue in the LHCb context, but in terms of higher trigger rate (as a Level-1 rate increase to 80 kHz is currently under study) and higher event fragment size (greatly determined by threshold cuts and zero-suppression algorithms). These two factors directly translate into a higher throughput requirement. Two complementary approaches are possible:

1. **At the system level only**, more RUs with a reduced number of incoming links per module would allow to handle higher throughputs. The drawbacks are evident: higher system cost and the requirement of a larger DAQ network switch, further increasing the system cost. This is the preferred approach until a higher-bandwidth technology is adopted.
2. **At the module level only**, higher FPGA clock frequencies as a result of optimized VHDL code and faster devices can enhance performance in the FEM application. In the DAQ application, 66-MHz PCI and higher-bandwidth network technologies would allow for a higher trigger rate.

3.8. Proposed Sub-event Transport Format (STF) for the DAQ

Raw data produced in the one million sub-detector channels are either transformed into a binary value (hit or no hit) or a digital value (via A/D converters). Channels carrying a no-hit or a digital value below a certain threshold level are discarded (Level-1 zero suppression). The remaining data are multiplexed and packed into a **data block** according to a sub-detector specific formatting convention, which is of course unknown by FEM, RU and SFC modules. Thus, an interface layer is needed between the Level-1 electronics and the DAQ ("MUX interface" in Figure 2-3) to encapsulate the detector-specific data blocks (payload) with header and trailer information to construct higher hierarchy frames whose format is understood by the DAQ system.The resulting frame (an **event fragment**) is sent to the DAQ via a Front-end Link.

The first stage in the DAQ system (FEMs) merges incoming event fragments into a **sub-event**, which has the same formatting convention (called **sub-event transport format**, shortened as **STF**) as the event fragments. DAQ RUs merge sub-events into larger sub-events and send them to a certain SFC, which receives all the sub-events and builds an **event**. The main purpose of the DAQ system is to carry out this task, called **event building**.

It can be concluded that a STF must be defined taking into account the following considerations:

1. The Data Block must be self-sufficient, i.e., all the information needed for Level-2 and Level-3 processing must be inside the block and do not concern to the readout protocols.
2. A low overhead format is required to keep the data flow in the DAQ as low as possible.
3. It must be simple to reduce the processing time in FEMs, DAQ RUs and SFCs (one more word to read means one more clock cycle).
4. It must take into account the capabilities of the link technology which will be used across the different stages in the DAQ system. Does the link technology already implement any sort of error checking/correction?. Does the link technology allow to mark some words as special (command or control) or allows any frame-delimiting mechanism?.
5. FEMs and DAQ RUs will merge incoming frames into a single outgoing frame with the same STF convention. How does this scale in terms of bandwidth and frame complexity?.

3.8.1. First proposals

Before proposing a frame format, the solutions adopted by other LHC experiments were studied and it was found that they introduced too much overhead. A major simplification of the ATLAS frame format [McLaren98] led to a first proposal shown in Figure 3-31.

According to this first proposal, two 32-bit header words (containing the event number, the data block size and a source identification number) and a single trailer word (containing status information and a 16-bit CRC code) encapsulate the payload (data block). Frame delimiting is achieved by means of a high-to-low transition in the flag bit, an additional signal available in a number of link technologies such as [Hewlett1]. As this is a slow-changing signal, transmission errors are rare and glitches can be detected, thus allowing for a safe frame delimiting. This logical frame format has its corresponding technology-dependent physical frame format. As most link technologies accept 16-bit input data, this is the assumed symbol size.

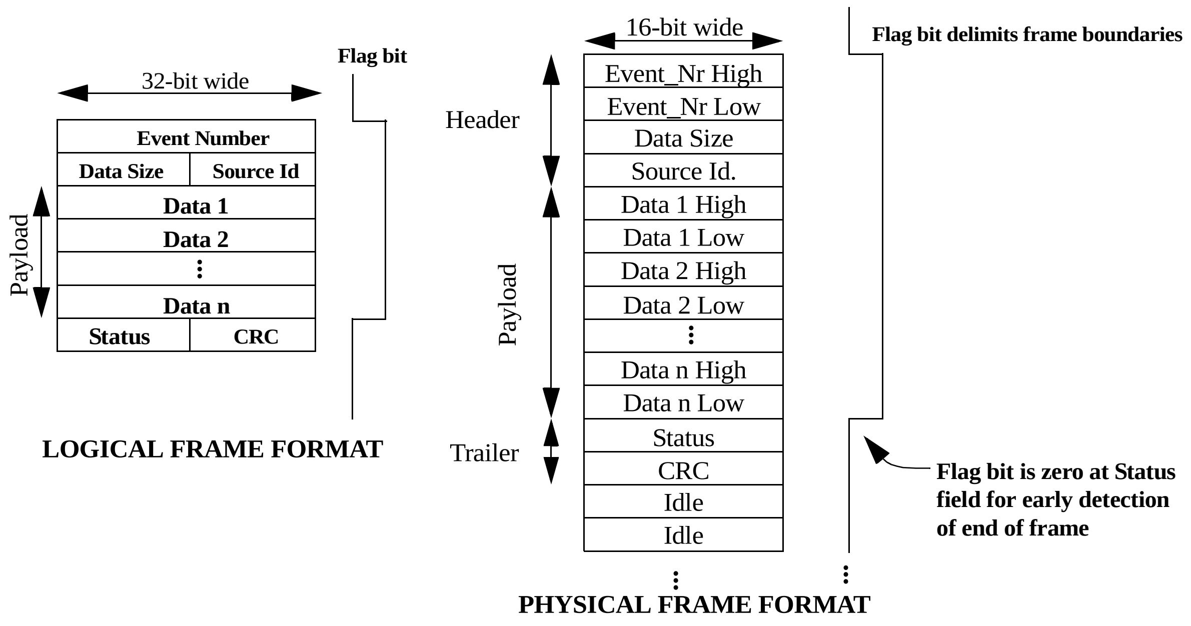


Fig. 3-31. First proposed STF.

FEMs and RUs are to be implemented using FPGAs and thus a 16-bit CRC code checking/ generation accepting 32-bit data words was implemented in a two-stage pipeline [Tol98-3]. The implementation consumed only 74 four-input LUTs (look-up tables), thus validating the feasibility of the frame format. Error correction schemes like Reed Solomon and Hamming codes were studied, leading to the conclusion that error correction techniques required too many resources for a practical on-the-fly implementation.

We shall consider how we should encapsulate these frames when crossing multiplexing stages. FEMs and DAQ RUs gather event data from its input links, merge them into a data block and construct a frame. Figure 3-33 illustrates this mechanism for a multiplexing factor of three. The same concept can be extended for an arbitrary multiplexing factor. The new frame has the same event number as the preceding ones, the data size is the addition of the lengths, the Source Id. field identifies the sender (the current FEM or DAQ RU) and the status word carries some flags (errors in transmission, in CRC checking, broken link condition, DPM full, etc.).

This scheme has the advantage of adding a small overhead and it is also a simple protocol itself. Once S-Link was adopted as the baseline link technology (see “CERN S-Link: a technology-independent interface to the Level-1 electronics” on page 80), the STF proposal was adapted to the characteristics of S-Link. Figure 3-32 shows the proposed frame format, in which a data block is encapsulated with only three framing words containing the following fields: event number, source identifier, type of frame, data size (in 32-bit words) and status information.

Start and end of frame are delimited by the **flag bit** signal, which is emulated in S-Link using a control word. As S-Link includes error detection, the CRC field from the first proposal was removed.

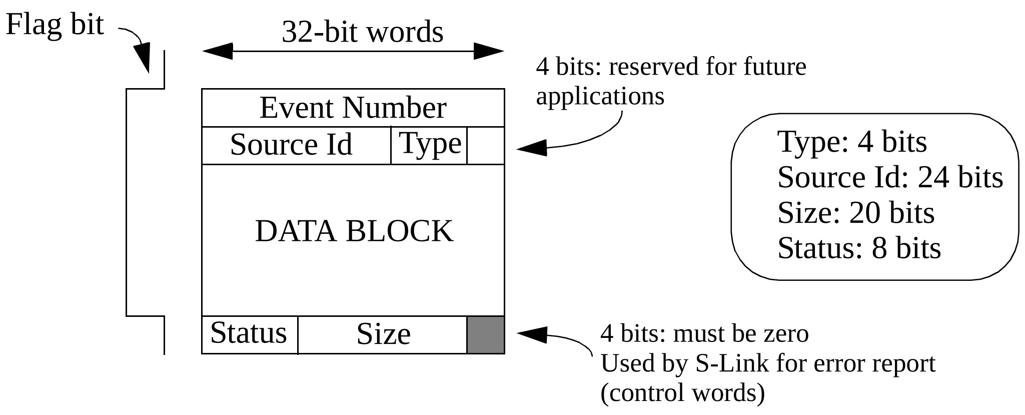


Fig. 3-32. S-Link based STF.

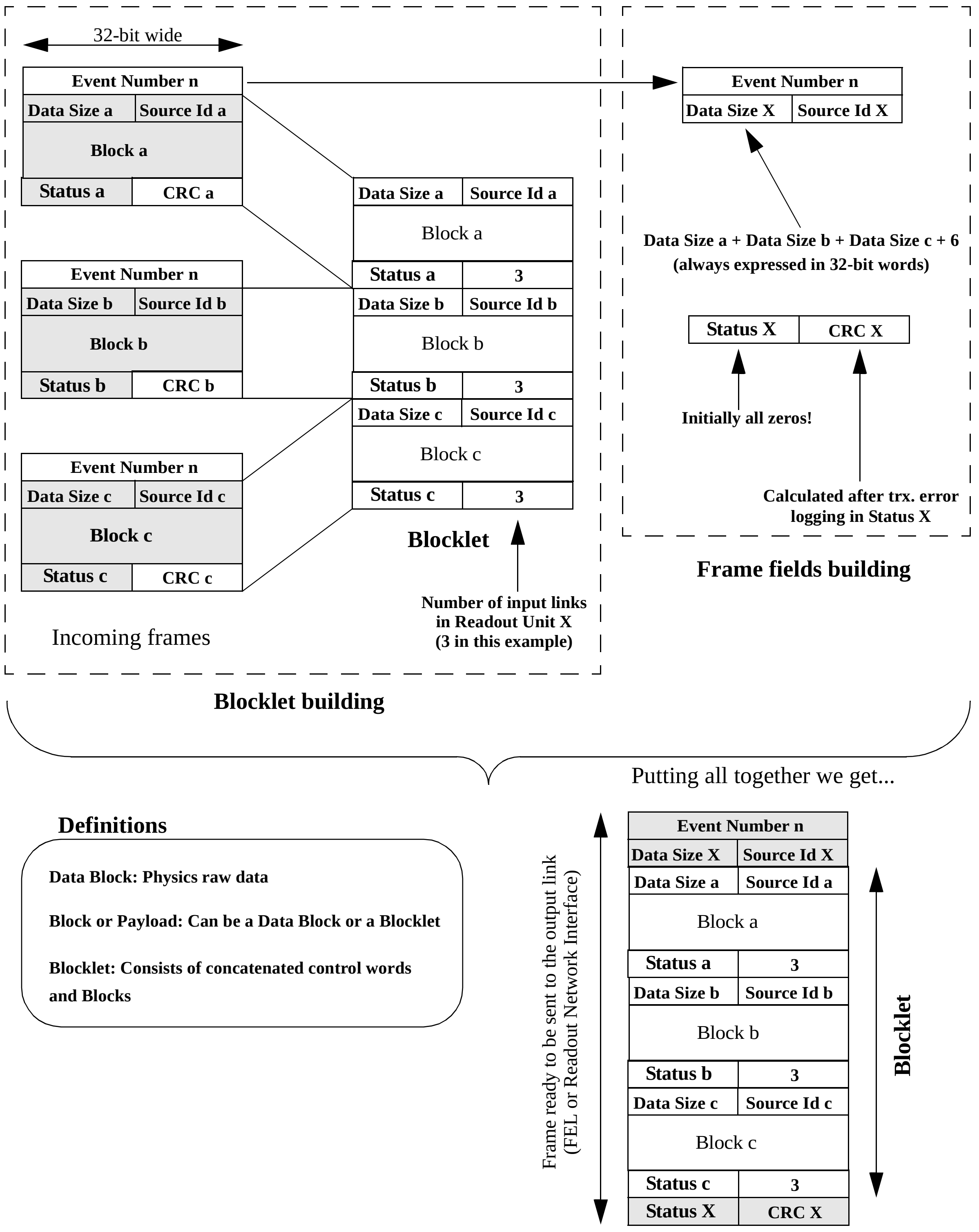


Fig. 3-33. Sub-event building according to the first proposed STF.

3.8.2. Final sub-event transport format

This work proposes the adoption of S-Link as link technology for the interconnections between Level-1 sources, FEMs and DAQ RUs. S-Link allows two kind of words: data and control. The proposed transport format (Figure 3-34) uses this feature to mark the first and last words in the frame as control words, thus providing easy frame boundary delimitation. According to the S-Link specification [S-Link] the four least significant bits in a control word are reserved for error logging, and thus the fields SH, ST (S-Link Header, S-Link Trailer) in the proposed transport format are imposed by S-Link.

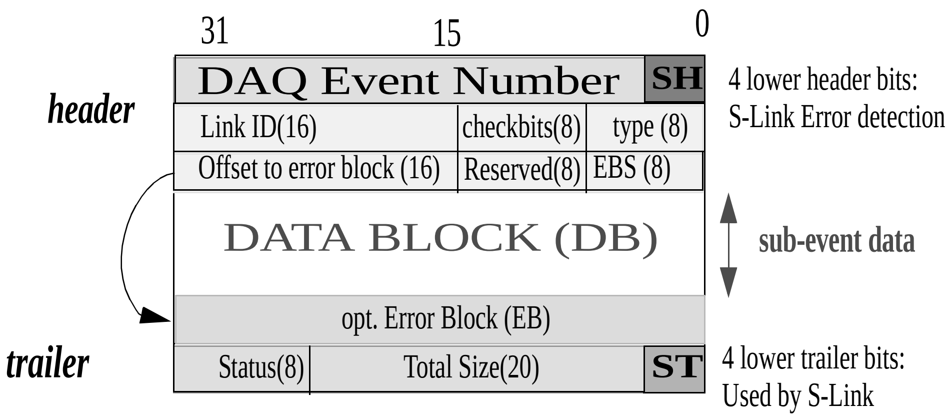


Fig. 3-34. Sub-event transport format for LHCb FEM and DAQ RU.

Frames consists of three header words, the payload data block, an optional error block and one trailer word. The resulting overhead is normally four 32-bit words[1]. In detail, the transport format is composed of the following fields:

- **DAQ event number**: It is a monotonically increasing number generated in the front-end electronics. It is incremented after a Level-1 accept, thus providing a “+1” search pattern for FEM and RU sub-event building engines. Though 28 bits are available, 20-bit numbers are already enough to provide unique event identification for more than 26 seconds and so there is no need to use all the 28 bits.
- **SH, ST (S-Link Header, S-Link Trailer)**: S-Link defined 4-bit error field.
- **LID (Link ID)**: It is a 16-bit field used to identify the sender of the frame. Each Level-1 source, FEM and RU have an unique identification number that is written into the LID field in every frame they produce.
- **CB (Check Bits)**: This is an optional 8-bit checkbit field for the header word, default is all zeroes.
- **Type**: Optional 8-bit field for type of data, default is all zeroes. It allows to distinguish between normal data taking, calibration runs and other modes of operation.
- **OEB** (Offset to Error Block): Offset (expressed in number of 32-bit words) to the optional error block in the trailer. This field is ignored (i.e., there is no Error Block) if EBS field is zero.

1. As errors do not happen frequently and error blocks are only built when errors occur.

- **Reserved**: 8-bit field reserved for future applications.
- **EBS** (Error Block Size): Error Block size expressed in 32-bit words.
- **DB (Data Block):** Payload data. Its structure is transparent to FEMs and RUs.
- **EB (Error Block)**: Optional error log consisting on one 32-bit word per error found. Its format is still to be defined.
- **Status**: 8 bits of fast status information for 8 types of error, identified by each bit. Default is zero.
- **Total size**: 20-bit field containing total number of 32-bit words in the frame, including header, error block and trailer.

Data transmission errors or synchronization errors can be detected by FEMs and RUs in three ways:

1. Incoming frames are already marked in the status field as erroneous by the upstream stage.
2. An error condition is reported by S-Link.
3. An error is detected by the RU logic itself.

In the last two cases, the corresponding bit in the Status field is marked in the outgoing frame and, if error logging is enabled, an entry is appended at the end of the Error Block (a 32-bit word consisting of the error code and the identifier of the unit in which the error was generated).

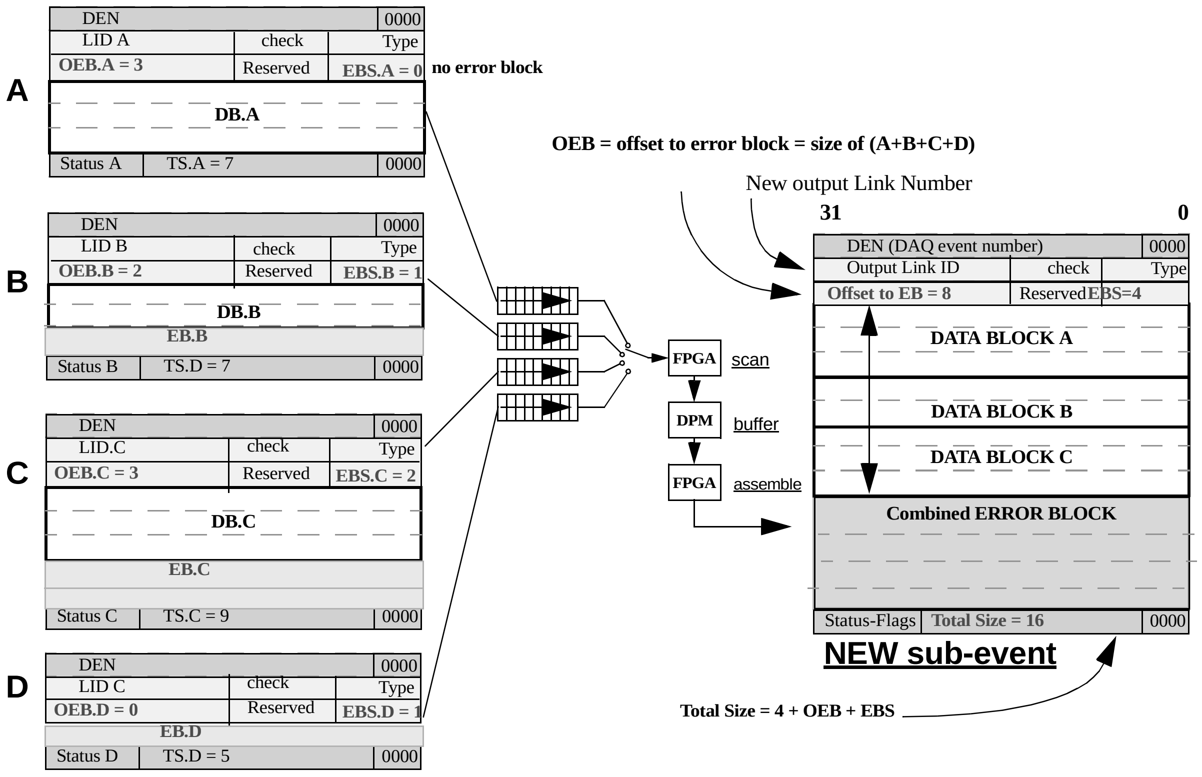


Fig. 3-35. Example of sub-event building.

The described STF transport format is applied in every stage where a new sub-event is built, i.e., both in the FEM and the DAQ RU. Data from a number of input links are reorganized every time into a new, larger sub-event. The full event-building of a single event is performed between the

DAQ RUs and a SFC. Figure 3-35 shows an example of a sub-event construction. A time out condition was reached for fragment D and a dummy frame with an error block logging this occurrence was produced by the input stage.

The described sub-event building format can be used recurrently and thus allows for an arbitrary number of sub-event building stages. All header and trailer words in the incoming framed are stripped off, reducing the framing overhead.

3.9. Proposed input and output link technologies

Input and output link technologies for the Readout Unit are not defined in the Technical Proposal for the good reason that in this fast evolving market it is not possible to foresee what gigabit link and network switch technologies will be the cheapest and long-term supported by 2005, when the experiment starts running.

There are four different types of links:

- Copper links for differential transmission of up to 20 m at 1 Gbit/s using parallel to serial converters. Example: Texas Instruments Flatlink [Texas1].
- Non-serialized parallel transmission using copper ribbon cables for up to 100 Mbit/s. Example: National LVDS line drivers/receivers [National1].
- Optical fibers for serially encoded transmission across 20-1000 meters at more than 1 Gbit/s. Example: HP Glink [Hewlett1].
- Parallel optical ribbon fibers, either DC or AC coupled for up to 100 meters. Example: Siemens Paroli [Siemens1].

For instance, twisted-pair cable links may be used at the Front-end region, whilst Gbit/s optical links are needed for the longer distance interconnection to the multiplexing stages in the DAQ. Between the two industry serial encodings, CIMT (synchronous) and 8b/10b[1] (asynchronous), today’s choice for Gbit/s transmission would be the Agilent Glink technology which implements CIMT for up to 1.6 Gbit/s using low voltage devices. Nevertheless, today’s choice has a low probability to be the same that would be made in three or four years time. The same applies to the DAQ network technology.

The incoming front-end links to the DAQ RUs the are assumed to be 8 or 16 bit wide on the physical level, corresponding to current gigabit link technology, LVDS copper ribbon cables or PAROLI optical parallel links. The physical width is converted to 32 bit via a simple multiplexer logic since the rate at which data are written to the FPGA is critical and defines the operating frequency of the FPGA.

At least until a decision is taken, the Readout Unit must implement a flexible link scheme that allows to test different technologies. The proposed solution, already approved by the LHCb Collaboration, suggest the use of the technology-independent CERN S-Link interface for FEM, DAQ RUs and VELO RUs inputs as well as FEM outputs, and the use of PCI as an interface bus

1. Used by Fiber Channel and Gigabit Ethernet.

to a commercial NIC for DAQ RU and VELO RU output. These two technologies are presented and evaluated in the following sections.

3.9.1. *CERN S-Link: a technology-independent interface to the Level-1 electronics*

The use of the CERN S-Link mezzanine convention [S-Link] was proposed to the collaboration in the framework of this thesis since it allows to choose from a variety of available mezzanine boards for either copper or optical links, which may be plugged on the RU via the S-Link connector (see Figure 3-36). A Link Source Card (LSC) transmits data and a Link Destination Card (LDC) receives data. **S-Link does not describe the physical link itself**. What S-Link defines is:

- The mezzanine and the connector, following the CMC standard [CMCstd].
- The 64-bit connector pinout. There are 32 data lines, clock, write enable, control/data, error and linkdown lines generated by the source and transmitted to the destination. For duplex S-link applications only, the destination (LDC) generates the signals: XOFF, RESET, TEST and four asynchronous return lines.
- The FIFO-like data transfer protocol between the mezzanine and the motherboard.
- Other protocols (like reset, link down, transmission error reporting, self test).

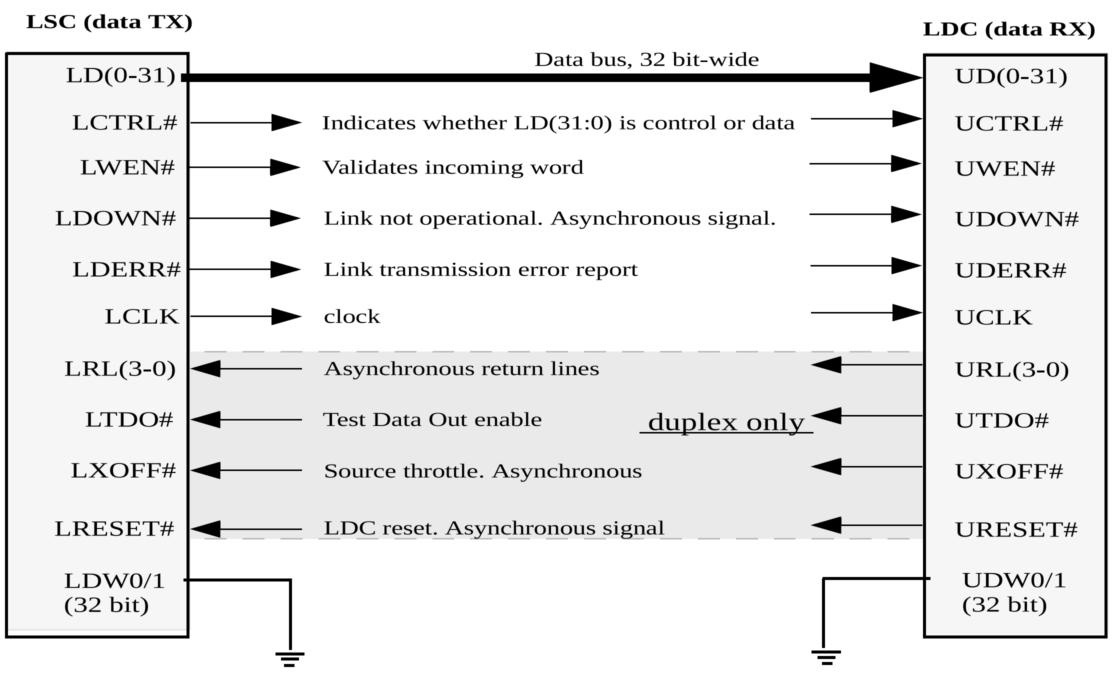


Fig. 3-36. S-Link connector signals.

The error detection mechanism (parity, CRC, etc.) is implementation dependent (i.e., not defined by the specification). Errors are reported either word-by-word (via the LDERR# line) or in a block-by-block basis (LDERR# line and the four least significant bits in a control word). S-Link obliges to perform error checking in all control words. This is used in the proposed STF to guarantee error checking in the first and last words in the frame, which carry the critical event number, frame size and status flag information.

The link physical implementation is transparent to the motherboard designer. The maximum link clock frequency is 40 MHz and the maximum data width is 32 bit[1], thus allowing for a maximum 160 MByte/s throughput.The link technology may be any current technology. Popular S-Link technologies are Glink (serial high speed) or LVDS for parallel short distance applications.

LCLK is a free running clock if LDOWN# is high, undefined otherwise. This might imply clock glitches and can be a cause of metastability if used by the receiving logic (SEM) as a master clock. In the RU, the receiving FIFOs decouple the S-Link and SEM clock domains.

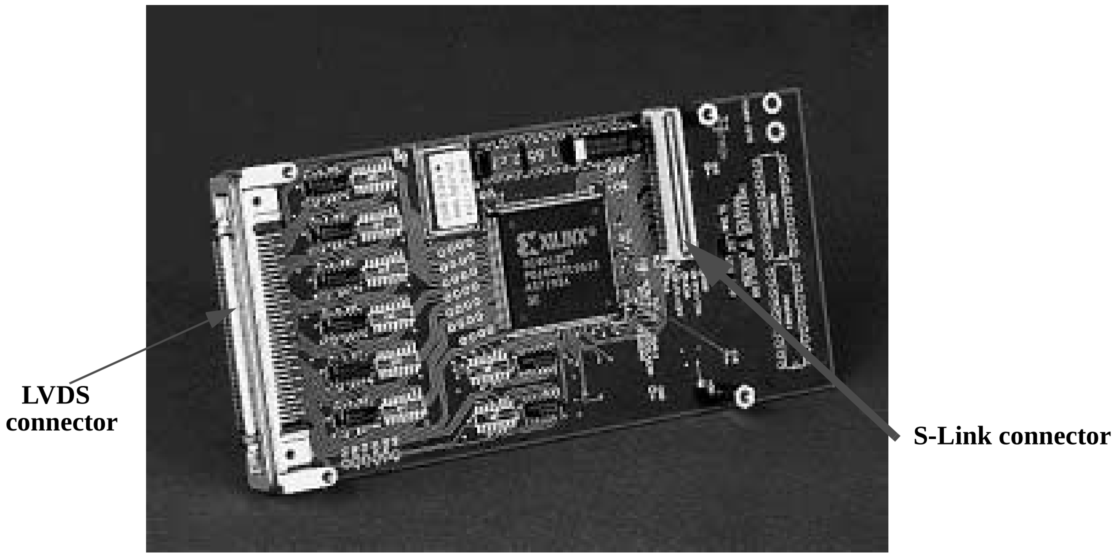


Fig. 3-37. Parallel copper S-Link card (LVDS signaling, 160 MByte/s).

The STF frame delimiting with S-Link control signals is depicted in Figure 3-38. A frame starts and ends with low levels in LCTRL# and LWEN# signals (control words). Two consecutive control words (ignoring idles) identify a start of frame.

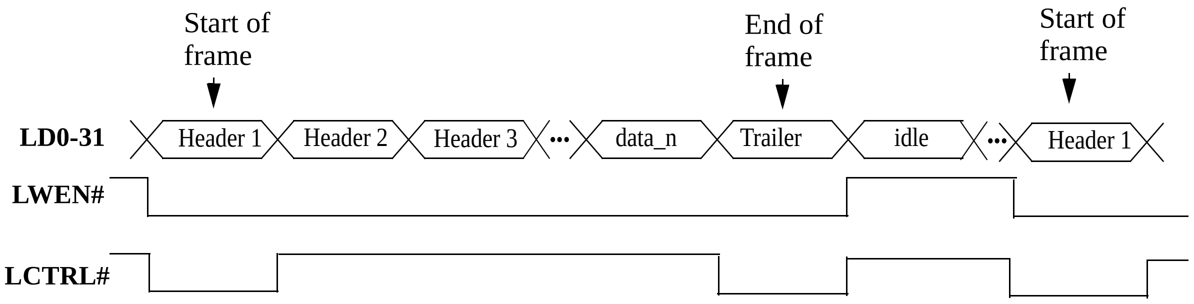


Fig. 3-38. Frame delimiting with S-Link control signals.

1. A 64-bit 100-MHz version of S-Link capable of handling up to 800 MByte/s has been recently defined. See http://edms02.cern.ch/tmpfiles/edms_tmp_1478521_slink64.pdf

3.9.2. *PCI: a technology-independent interface to the Network Interface Controller*

The PCI (Peripheral Component Interconnect) is an I/O expansion bus designed for the desktop PC market, aimed at replacing ISA, EISA, VESA and other legacy I/O standards. In its basic configuration (32-bit 33 MHz) it provides a 132 MByte/s raw bandwidth. A factor of four increase in throughput (i.e., up to 528 MByte/s) can be achieved with a 64-bit 66-MHz implementation.

PCI is an unterminated CMOS bus [PCIstd], what means very low static currents. This energy-saving feature is enhanced by the fact that PCI uses reflected-wave switching, which implies moderate dynamic currents (around 300 mA). In a reflected-wave switching bus, unterminated lines are driven with the incident wave at half the transition point. The reflected wave propagates back to the loads to duplicate the voltage step and drive a valid logic level.

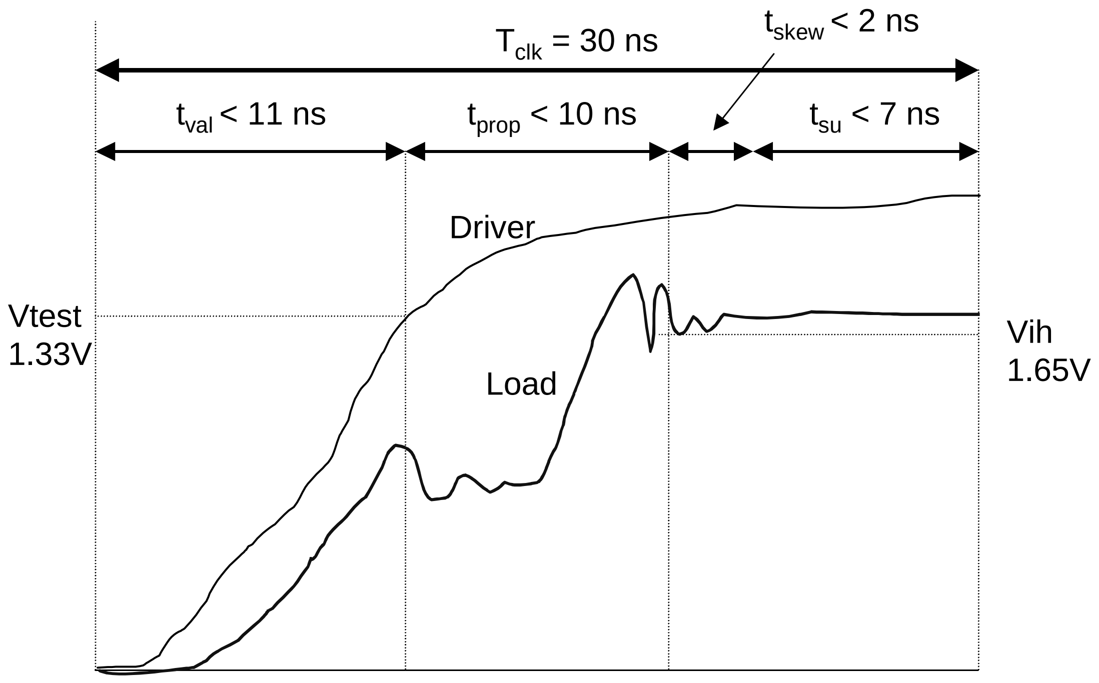


Fig. 3-39. Timing budget for 3.3V PCI implementation in 33-MHz mode.

Figure 3-39 shows the typical voltage step in the load produced by the reflected wave and the timing constraints involved in a 33-MHz implementation. Allowing a clock skew of 2 ns and assuming a setup time in the receivers of 7 ns, only 21 ns are available for signal propagation. This includes the time taken by the driver to reach V_{test} and twice the propagation time through the bus and the settling time in the loads. The more loads on the bus, the more capacitance and, as a result, the slower signals propagate. The longer the bus, the more time it takes the signals to propagate.

This highlights the two major drawbacks of PCI: the reduced number of loads allowed in a bus (between four and six) and the limited bus length. These limitations are partially avoided by the use of PCI bridges (Figure 3-40). From the electrical point of view, a PCI bridge is seen from a PCI bus like a single load, regardless of what is on the other port. This allows to add an arbitrary number of loads in the system and to extend physically the bus a few more centimeters.

PCI systems are hierarchical, consisting of a primary PCI bus where the PCI host resides and secondary PCI buses created by the addition of PCI bridges (Figure 3-40). The use of PCI bridges is an efficient way of traffic isolation, forwarding upstream (towards the host) or downstream only those transactions requiring the participation of a PCI device that belongs to other bus. This eases scalability, as simultaneous transactions (one per bus) can happen in the system. As a drawback, inter-bus transactions suffer from additional latencies introduced by the PCI bridges, resulting in reduced effective bandwidth.

Despite this hierarchical architecture, PCI defines a shared memory space model. All devices in a system share a common 32-bit memory space. In a transaction, the destination is indicated by a memory address, not by a geographical address (bus, slot, device, position). Thus, implementing DMA is straightforward in PCI. The PCI host assigns during system configuration the base addresses for each device.

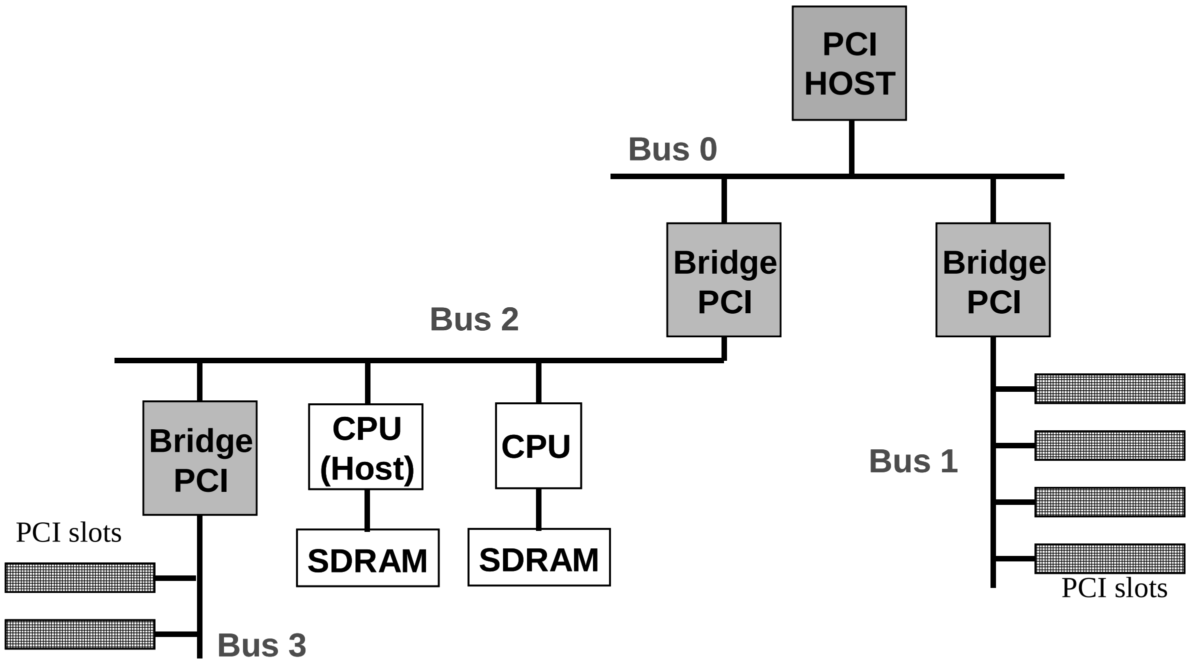


Fig. 3-40. An example of hierarchical PCI architecture.

Devices behave either as masters or as targets. Masters initiate transactions and targets respond to transactions. Thus, a device can implement both master and a target interfaces or only a target interface. To initiate a transaction, a master must request ownership of the bus and this request must be granted. PCI implements a hidden arbitration scheme (the arbitration is done ortogonally to the main data flow by using separate signals and thus transactions and arbitration can happen simultaneously) in which one dedicated pair of signals (request and grant) connect each device to the bus arbiter. The arbitration algorithm is not defined in the PCI specification. Once a master has obtained the bus mastership it can initiate a transaction consisting basically of one address/command phase, an arbitrary number of data phases and a termination cycle. This implies that all transactions are indeed burst transactions with a number of data phases limited by:

- The arbiter, who can be programmed to prevent a master from monopolizing the bus.

- The target, who can have a limited buffer space and interrupts the transaction when the limit is reached.
- The master itself, who can decide to give other masters a chance to transfer data.

Figure 3-41 depicts the memory write transaction protocol. Address and command are validated when the master asserts FRAME# (1). This cycle is followed by a number of data phases (2). The last one is indicated by the master by deasserting FRAME# (3). A turnaround cycle is needed before the next transaction can take place (4), unless there is no change of master and then this cycle can be saved (fast back to back transactions). Wait cycles can be inserted at any time by both master (IRDY# deasserted) and target (TRDY# deasserted).

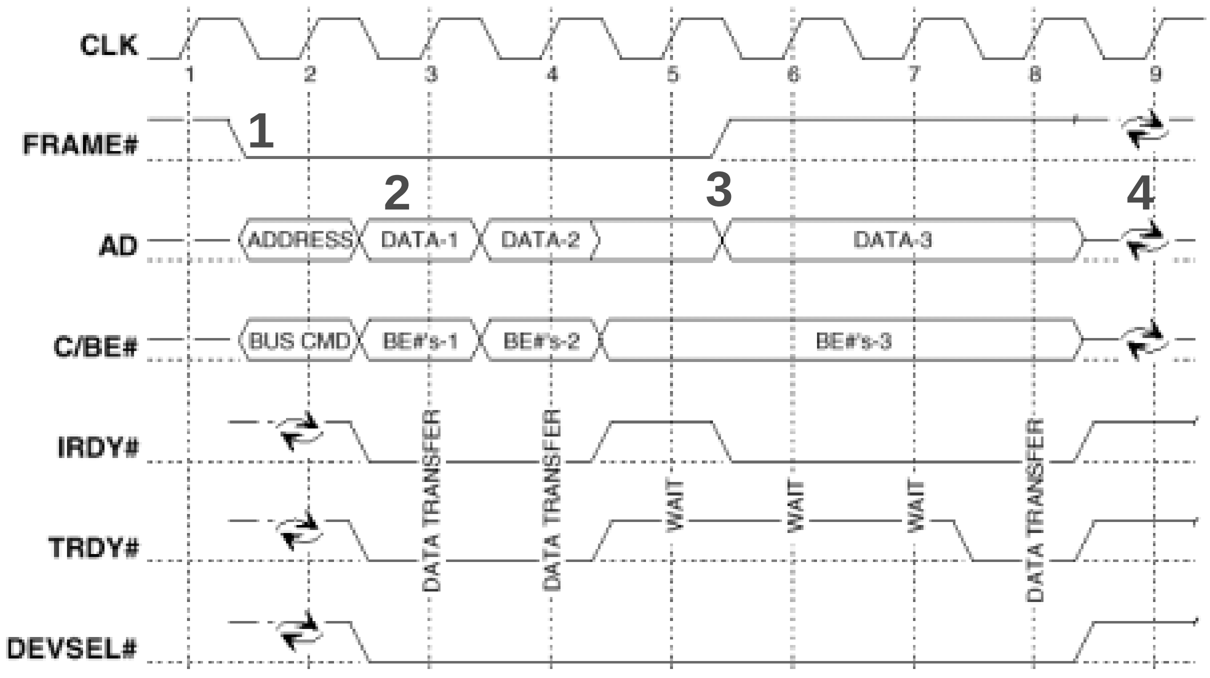


Fig. 3-41. Write transaction protocol in PCI.

Memory read transactions (Figure 3-42) are also initiated by an address/command phase validated by FRAME# (1). A turnaround cycle is needed next as the target has to drive the data bus (2). The target validates each data phase by asserting TRDY# (3) until the master has read enough data, which is indicated by FRAME# deassertion (4). A turnaround cycle is needed before the next transaction can take place (5). Wait cycles can be inserted at any time by both master (IRDY# deasserted) and target (TRDY# deasserted).

For both write and read transactions, the target must assert DEVSEL# within a certain number of clock cycles. This number and the delay introduced to complete the first data phase account for the target latency.

From the performance point of view, a write operation requires one address phase, *L* target latency cycles, *N* data phases and a turnaround cycle. The maximum achievable bandwidth (assuming *L*=3) can be calculated by expression 3.10. A read transaction requires an extra turnaround cycle and thus the performance is slightly slower (expression 3.11).

$$Throughput(Mbyte/s) = \frac{4 \cdot 10^3 \cdot N}{30 \cdot (5 + N)} \tag{3.10}$$

$$Throughput(Mbyte/s) = \frac{4 \cdot 10^3 \cdot N}{30 \cdot (6 + N)} \tag{3.11}$$

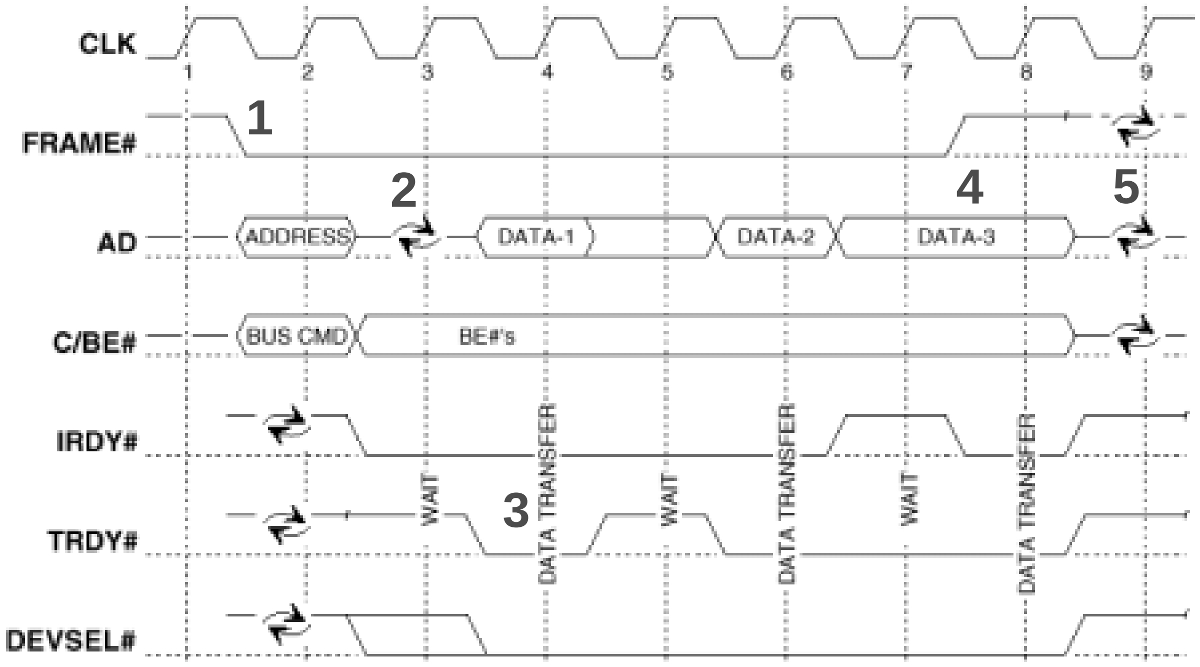


Fig. 3-42. Read transaction protocol in PCI.

Figure 3-43 depicts the theoretical effective bandwidth for a 32-bit 33-MHz bus.

Figure 3-44 shows effective bandwidth measurements on DMA write operations from a FPGA-based DMA engine (residing in a PCI slot) into the PC main memory [Costi&Toledo99]:

- **System A**: Pentium 100 MHz CPU with chipset 430HX and 32-bit 33-MHz PCI bus. EDO RAM 60 ns. One PCI slot populated by a video card.
- **System B**: Pentium II 400 MHz CPU with chipset 440BX and 32-bit 33-MHz PCI bus. SDRAM 100 MHz. Video card in a AGP slot.

The peak value of 66 MByte/s for system A corresponds to the limit imposed by the main memory bandwidth (32-bit bus width and 60 ns cycle time), whilst System B has a higher main memory bandwidth and measurements are closer to theoretical values.

Performance in a PCI system is highly dependent on the burst length and on the chosen platform and thus system evaluation and tuning is required to achieve an efficiency higher than 80%.

In a different approach, using non-DMA transactions (i.e., PCI read/write with the CPU), performance tests moving data blocks larger than 64 KByte to an from an ATI 3D RAGE PRO video card have been carried out [David01] on several platforms running Linux 2.2.x. Test utilities were executed in kernel mode for higher performance. Results are summarized in Figure 3-45.

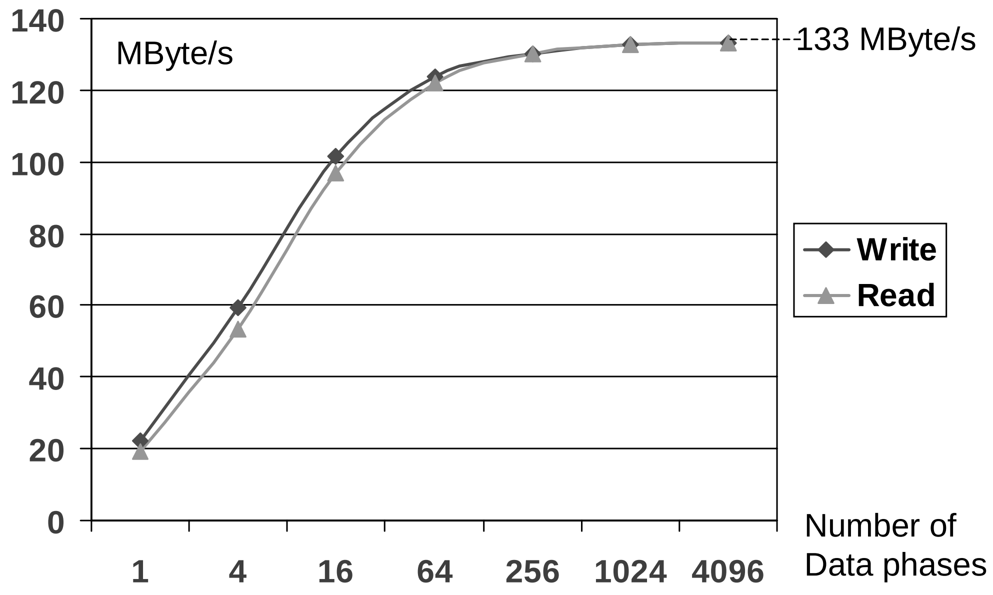


Fig. 3-43. PCI bus theoretical performance for a 32-bit 33-MHz bus.

- **Write tests** reported 103 MByte/s on a 400-MHz Pentium II with i440BX chipset, 112 MByte/s on an 800-MHz PentiumIII with i815EP chipset, but only 63 MByte/s on an Athlon-based PC with VIA-KT133 chipset. All platforms had 32-bit 33-MHz PCI. These results show promising bus efficiency for Intel chipsets.
- **Read tests**, on the other hand, did not yield more than 9 MByte/s in any of the tested platforms. This is explained by the fact that PCI chipsets are not optimized[1] for CPU reads from PCI, as DMA is a more efficient paradigm to handle this data flow. Nevertheless, read performance can be boosted as described in detail in [David01]. Firstly, the PCI board's prefetchable memory must be defined as cacheable in the chipset's MTTR (Memory Type Range Registers) [IA-32] in order to force a cache fill request rather than a data request. This yielded a 21 MByte/s performance on the i815EP platform, no improvement at all on the VIA-KT133 and 26.2 MByte/s on an AMD761-based platform. Secondly, the cache line size register in the PCI bridge in the PCI board (an i21154 PCI-to-PCI bridge in our tests) was set to the maximum value (sixteen 32-bit words), allowing longer bursts and yielding 37.8 MByte/s on an AMD761 platform.

The widespread use of PCI in the desktop arena stimulated its application in other fields (including industrial applications and HEP experiments [Praag95], [Müller95]) and led to the development of a number of derivatives designed to supersede PCI's mechanical and electrical limitations. Indeed, a PCI backplane is limited to a few centimeters in length and a few (normally four or six) electrical loads. Moreover, no physical expansion mechanism is foreseen in the PCI specification. Three of these PCI derivatives are:

- **Compact PCI** [PICMG97]. Defined in 1995 by Ziatech as an open standard which combines PCI performance, Eurocard 3U and 6U form factors, hot swap capabilities and a higher bus

1. North Bridges in PCI chipsets lack of buffers for CPU-initiated read operations.

loading capability, more adequate for industrial applications than the original PCI specification. Compact PCI is based on interconnected segments, each segment consisting of up to eight boards. The high loading in the bus segments is achieved by adding 10-ohm series terminations in each board in order to damp the reflections. As a result, a modular and scalable system using Eurocard mechanics expands PCI's applications range.

- **PXI** (PCI eXtensions for Instrumentation, 1997) [PXI00]: Based on the Compact PCI specification, adds to the backplane a set of signals for synchronization, timing and trigger that are prevalent in instrumentation applications. The chosen software platform is Windows and VXI plug&play drivers.
- **PMC** (PCI Mezzanine Card) [PMCstd]: Defines a connector pinout to combine the PCI specification with the CMC (Common Mezzanine Card) mechanical standard [CMCstd]. This reduced form factor (15 by 7.5 cm aprox.) is ideal for equipping 6U or 9U Eurocard boards (like VME) with plugable mezzanine cards for I/O, memory or CPUs.

  The extensions defined in VITA-32 standard allow to implement a PCI host on a PMC, widening the range of applications and system configurations [VITA32std].

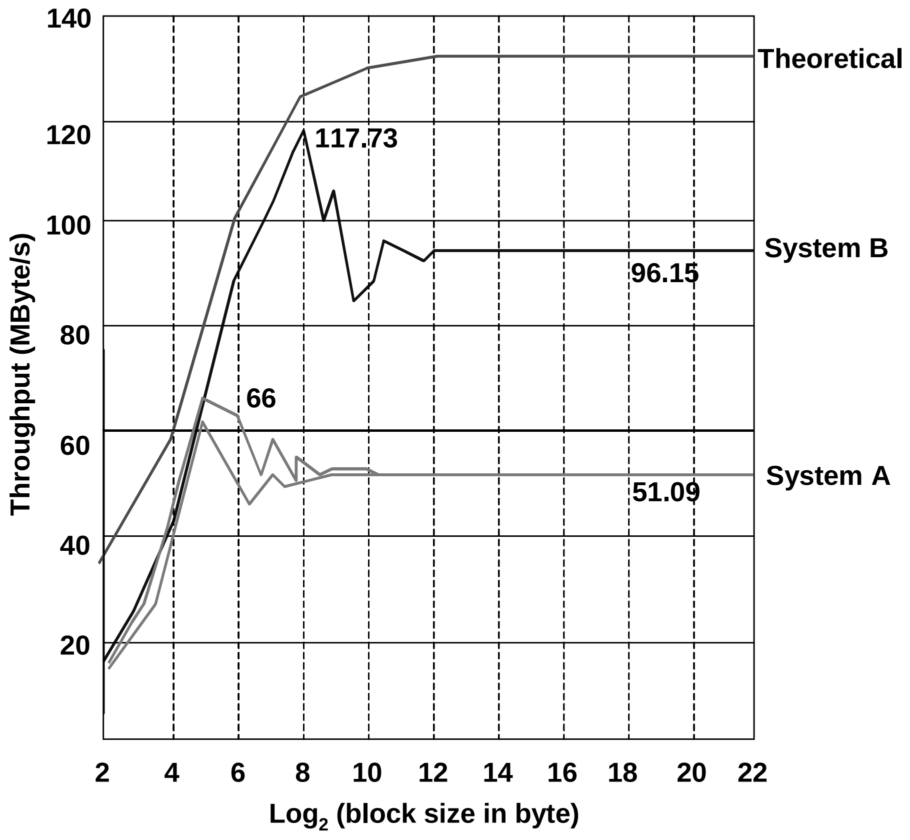


Fig. 3-44. Throughput measurement on two commodity PCs.

A recent extension to PCI 2.2, the **PCI-X** specification [PCIX], conserves the mechanical specifications of PCI and admits up to 133 MHz clock rates, thus allowing up to 1 GByte/s on a 64-bit system. The electrical limitations of PCI apply and as a result, only one load per PCI-X segment at 133 MHz is allowed. Two loads are allowed at 100 MHz and four at 66 MHz. Being backwards compatible with PCI 2.2 and allowing a 10-30% performance increase in multi-master systems due to modified protocols like the new split transactions, PCI-X is bound to replace PCI.

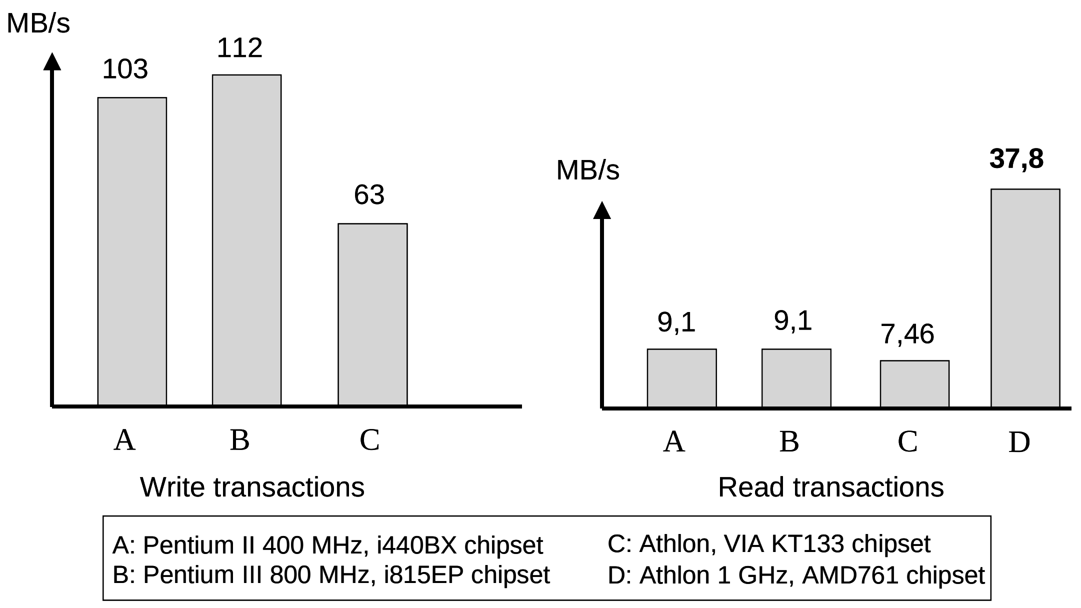


Fig. 3-45. Performance for CPU-initiated PCI transactions in several platforms.

3.10. Summary

The needed functionalities and design constraints for the three target applications are presented in sections 3.1 and 3.2, together with a basic architecture for a RU module. Different options are analyzed in sections 3.3 to 3.7, demonstrating the weak dependence of the performance with the specific implementation (determined by the number of processing elements in the input and output stages).

Input and sub-event buffer size estimation result in very relaxed needs for input FIFOs (a few kilobytes) but more demanding requirements for the SEB (2 MByte minimum). This implies, due to high cost and low density of integration of true dual-port memory, that an efficient memory usage is mandatory, as suggested in section 3.4.

Sub-event building algorithms for each application are proposed in section 3.5. In the FEM application, a 16-bit output to an S-Link transmitter card provides the required performance with a safety factor greater than two. PCI-based sub-event building between the RU output stage and the NIC aims at higher throughput applications like DAQ and Level-1 VELO trigger. In the former case, a “pull” protocol ruled by an intelligent NIC results in unnecessary overhead which can be tolerated as performance is not affected due to the large frame sizes involved. The benefits of intelligent NICs are in their capability to perform traffic shaping and complex readout protocols with the readout network.

The Level-1 VELO trigger application cannot afford the overhead implied in a “pull” protocol due to the small sub-event size (around one quarter of a kilobyte, sixteen times less than in the

DAQ application). A “push” protocol is used in which the RU output stage writes sub-events into the NIC buffer via PCI transactions. The PCI bus protocols reduce the bus efficiency to a poor 64% in a single-processing element scheme. A tandem-FPGA output stage architecture is proposed in section 3.6 which can potentially yield a higher performance. Nevertheless, the sub-event size, FPGA and NIC latencies (measured in laboratory tests) combine in this specific application to annul this superiority.

A sub-event transport format for encapsulating and handling data through the different DAQ stages is proposed. Solutions for I/O technologies in the RU are also proposed (S-Link at the input and at the output for the FEM module, and PCI at the output for the other two applications).

CHAPTER 4 The first Readout Unit prototype

"Any path that narrows future possibilities may become a lethal trap"

From the novel "Children of Dune" by Frank Herbert.

4.1. Overview

The first RU prototype module (Figure 4-1) was presented to the collaboration in the May 2000 LHCb Electronics Workshop [Müller00-2]. This prototype module aimed at evaluating some crucial components (like the chosen FPGAs[1] and Monitoring and Control Unit[2]), a number of design issues (like CMOS switches on the PCI bus and the planar PCI bus implementation) and to demonstrate the Readout Unit concept before developing a final module. The experience gained in the design, manufacturing and test of this prototype was of great importance to the successful implementation of its successor: the Readout Unit II.

This prototype was conceived as a modular system, consisting of a 9U-sized Fastbus motherboard with plug-in mezzanine cards for I/O and MCU, targeting the DAQ RU and FEM applications only. The board has a multiplexing factor fully configurable via the monitoring and control network to be any number between one and four, as four is the maximum number of mezzanine cards that fit in the front panel of a VME 9U board.

The crate backplane is not involved in the dataflow, all I/O links are point-to-point connections which are implemented via mezzanines on front and rear-panels of the RU motherboard. The Fastbus crates are only used for power supply and mechanical support. Double-width modules are used, as CMC mezzanine cards with high connectors like RJ-45 would not fit in the height

1. Lucent Technologies' ORCA 3TP12 was the first 64-bit 66-MHz master/target PCI interface implementation on an FPGA. This device could be used in a compact and high-performance solution for the RU output stage, but had not been tested before at CERN.
2. A commercial single-board computer based on the Intel i960 processor family [Pxecore00] was chosen as Monitoring and Control Unit (MCU) for the RU prototype, and thus had to be evaluated.

available in a single-width IEEE960 module (16.51 mm). The reasons to use Fastbus and not VME mechanics are:

1. The availability of a large number of Fastbus crates which could be acquired for free as a result of the dismantling of LEP experiments.
2. The upper half of the Fastbus backplane can be removed and thus Fastbus boards can accommodate two PMC slots on the back-panel. VME, on the other hand, inconveniently requires to use transition modules to use the back-panel. This would imply long buses from the motherboard logic to the output connector.

As it can be observed in Figure 4-1, there are connectors for two PMC (IEEE P1386.1) mezzanines on the rear side: one for a Network Interface Controller card (NIC) and a second one for an auxiliary PMC card. Physically, the connectors for the NIC and the S-Link cards are on the same mezzanine slot and thus both cards are mutually exclusive. Four S-Link receiver mezzanines are located on the front-panel side. The MCU, a i960-based subsystem [Pxecore00] on a card has the LAN connection routed via the printed-circuit board to the front panel. There are reset input and Throttle output signals available via NIM coaxial connectors.

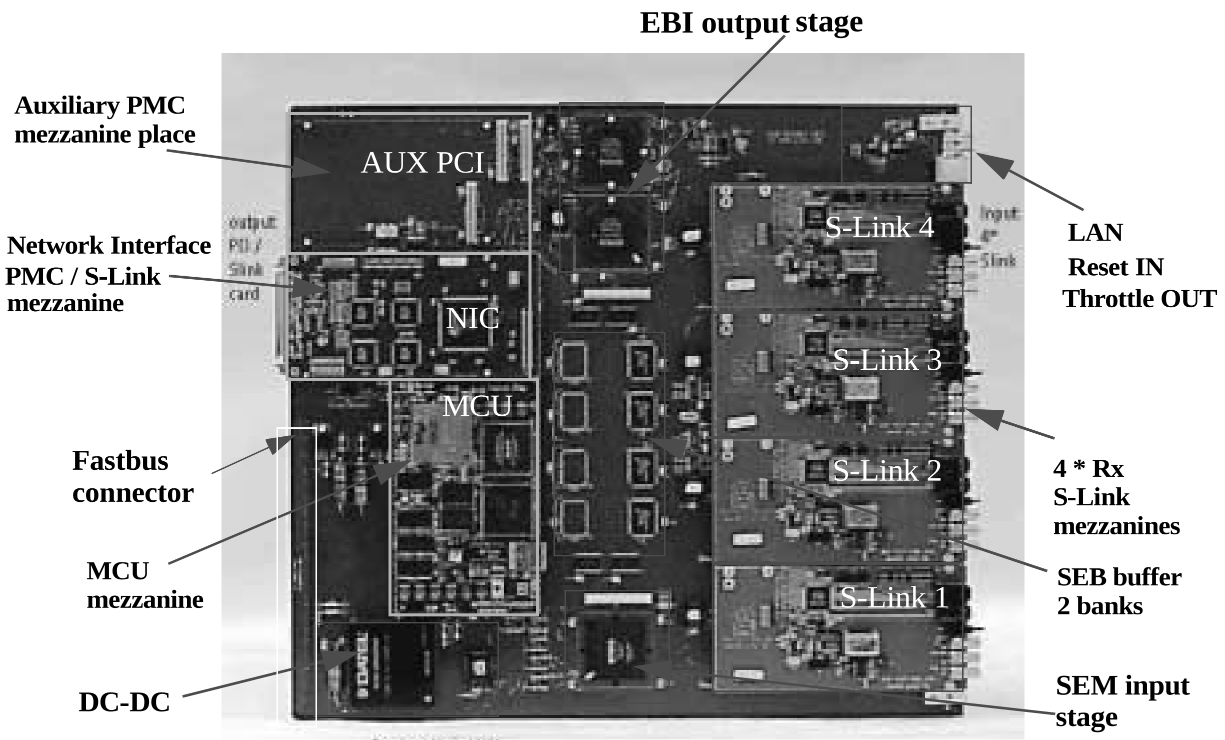


Fig. 4-1. First RU prototype.

Apart from these I/O connectors and plug-in cards, there are a number of ICs on the RU motherboard that implement the following functional blocks:

- **The sub-event buffer** (SEB) buffer is implemented as two separate memory banks. One or two banks (each consisting of four 16-bit wide true dual-port memory[1] ICs) can be populated, depending on the buffering needs.

- **The input stage FPGA**, sub-event merger (SEM), reads the input FIFOs (placed below the S-Link mezzanines) and stores the event fragments in the SEB.
- **The output stage FPGAs**, or event-builder interface (EBI). Only one is needed, though a second one has been provided for extended user logic (in case the sub-event building logic does not fit in a single device).
- **The voltage regulators**, to convert 5V from the Fastbus crate power supply down to *3.3 V*. Additionally, the $\pm 15V$ from Fastbus are regulated to $\pm 12V$ as these voltages are mandatory in the PMC slots. There is no need for *2.5 V* supply, as low-voltage devices did not spread in market until year 2000.
- The **timing circuitry** (not visible in Figure 4-1) consisting of a programmable clock generator and associated clock buffers, to distribute clean clock signals through the board.

4.2. Module architecture

Figure 4-2 shows the RU prototype architecture with four data link receiver inputs and a PCI (optionally S-Link) output. The S-Link output option allows to use the RU also as FEM. The RU prototype is optionally equipped[1] with a monitoring and control unit (MCU) for communication, error handling and configuration of the FPGAs, PCI bus, and programmable clock chips. The MCU card in the prototype is a commercial i960 subsystem with LAN controller, PCI bus and I/O functions supporting the I2C, JTAG, GPIO and interruption lines as required by the RU logic (see “The Monitoring and Control Unit” on page 100).

The input and output stages are implemented as two independent FPGA functions: SEM (Sub-Event Merger) and EBI (Event-builder Interface). The PCI bus, an integral part of both the EBI FPGAs and the MCU, interconnects the output of the MCU and the EBI FPGAs to the NIC, which is assumed to be a standard PCI device on a PMC form factor. The input and outputs stages work independently with their proper programmable clock domains.

The 64-bit architecture of the RU is conceived for a maximum raw bandwidth of 528 MByte/s (66-MHz PCI mode) or 264 MByte/s (33-MHz PCI). Its capability to cope with high data bandwidths relies on the extensive use of 64-bit data paths and fast hardware algorithms in FPGAs. Performance tests carried out on this prototype have shown that the DAQ working point of 1 KByte input sub-events at 40 kHz can be reached already with the prototype logic operating at 28 MHz.

1. In 1998 and 1999 the maximum commercial FIFO width was 36 bits, whilst the maximum DPM width was 18 bits, reaching 36 bits in year 2000. So, 16-bit memories were used for the SEB.

1. Not required for the FEM operation as the PCI bus is not used. This is true as far as FEM modules are not connected to the ECS.

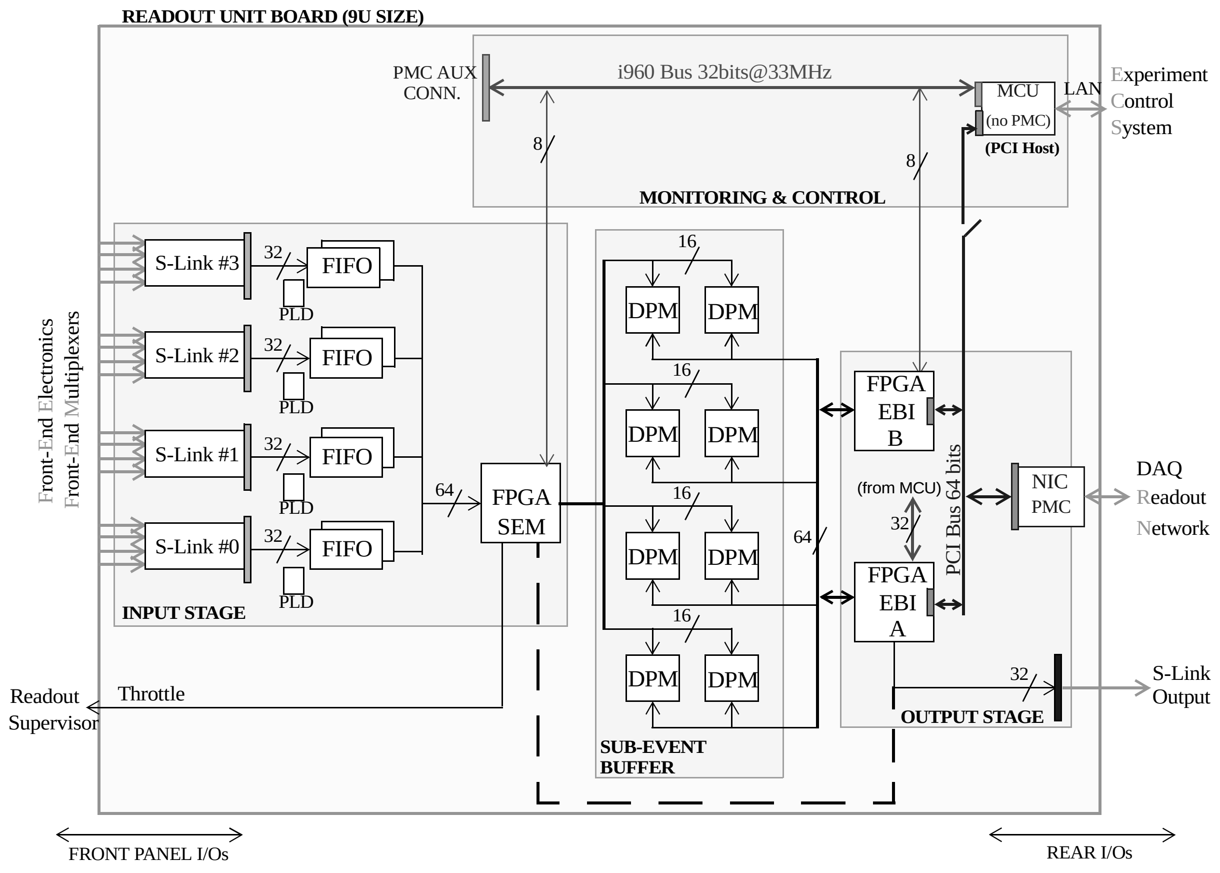


Fig. 4-2. Architecture of the first RU prototype.

4.2.1. *Input stage*

From several possibilities (see discussion in “Input stage” on page 53) the chosen architecture is based on single-processing-element input and output stages, which requires 64-bit FIFO memories. Each 64-bit FIFO memory consist of two 32-bit FIFOs working in parallel, where an interleaved writing scheme (implemented in a GAL) performs the 32-to-64 bit demultiplexing, as depicted in Figure 4-3. A snapshot of the S-Link connector and the two FIFOs below the mezzanine is shown in Figure 4-4. FIFO size can be selected between 64 KByte and 1056 KByte per chip using pin-compatible memories.

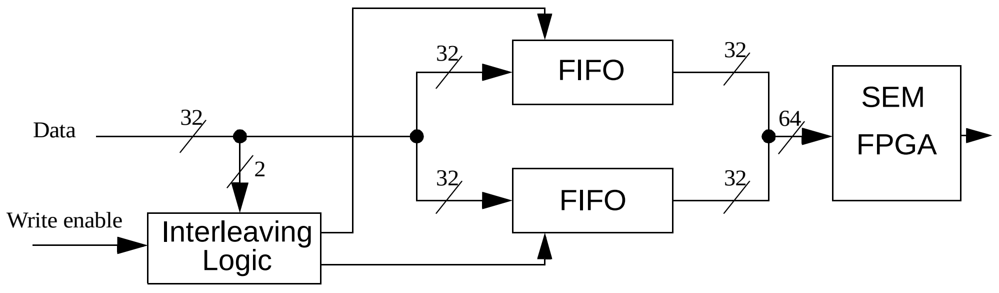


Fig. 4-3. Implementation of a 64-bit FIFO with 32-bit buffers.

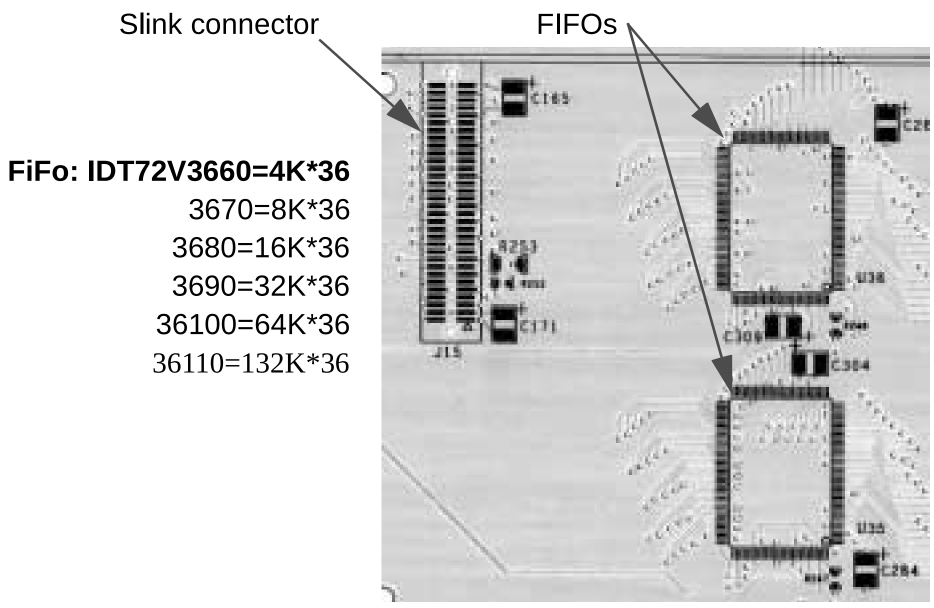


Fig. 4-4. Detail of the S-Link connector and the two FIFOs below the mezzanine.

In the prototype, a single 3.3-Volt 55-kgate FPGA [Lucent3T55] in a 352-BGA package is used for the SEM input logic. The choice was based on previous experience in several projects at the CERN EP/ED group, though there is no constraint on using other vendor devices. The FPGA code design methodology is based on VHDL description and simulation (Innoveda's VisualHDL), logic synthesis (Specctrum's Leonardo and Synplicity's Synplify) and FPGA vendor place and route tools (Lucent Technologies' ORCA Foundry), as described in the Appendix I.

The merging/multiplexing functionality is performed by the SEM FPGA which polls the FIFO outputs and forwards events either to the output link interface[1] (shown in Figure 4-2 as a dashed line) or to the output buffer (SEB). The former was foreseen for non-buffered applications. The functional block diagram of the SEM FPGA is sketched in Figure 4-5. This FPGA communicates with other devices via three interfaces:

- **FIFO interface**: Allows the SEM logic to read data from the 64-bit FIFO bus, monitor empty and full FIFO flags, and control FIFO operation.
- **S-Link and DPM interfaces**: Depending on if the multiplexer motherboard is used in buffered or unbuffered applications, the built sub-event is sent to either S-Link or to the DPM.
- **CPU interface**: Links the FPGA with the i960-based MCU using an embedded 8-bit i960 interface.

In unbuffered applications, the *Main Control* (a state machine in SEM logic) creates the header for the next sub-event and, according to the status of each link (enabled, disabled, unavailable, empty FIFO, time out reached, etc.), determines if a new block has to be read (activating the *Read Block from FIFO* state machine), if a dummy block has to be created (activating the *Create Dummy Block* state machine) or if errors have happened and an error block must be appended at

1. The FEM is considered today to be a buffered application, though both options (buffered and non-buffered) were under discussion by the time the first RU prototype was designed and thus both are supported.

the end of the frame (*Create Error Block* state machine). Finally, the trailer is appended at the end of the frame.

In buffered applications, the synchronization and operation of the functional blocks is programmed in a different way to implement the scheme described in Section 3.4.1. "Memory management scheme" in page 61. This way, a state machine (Main Control) controls other four state machines, simplifying the SEM logic. Three-state buses are used to avoid big multiplexers (buses are 64-bit wide) between pipeline stages.

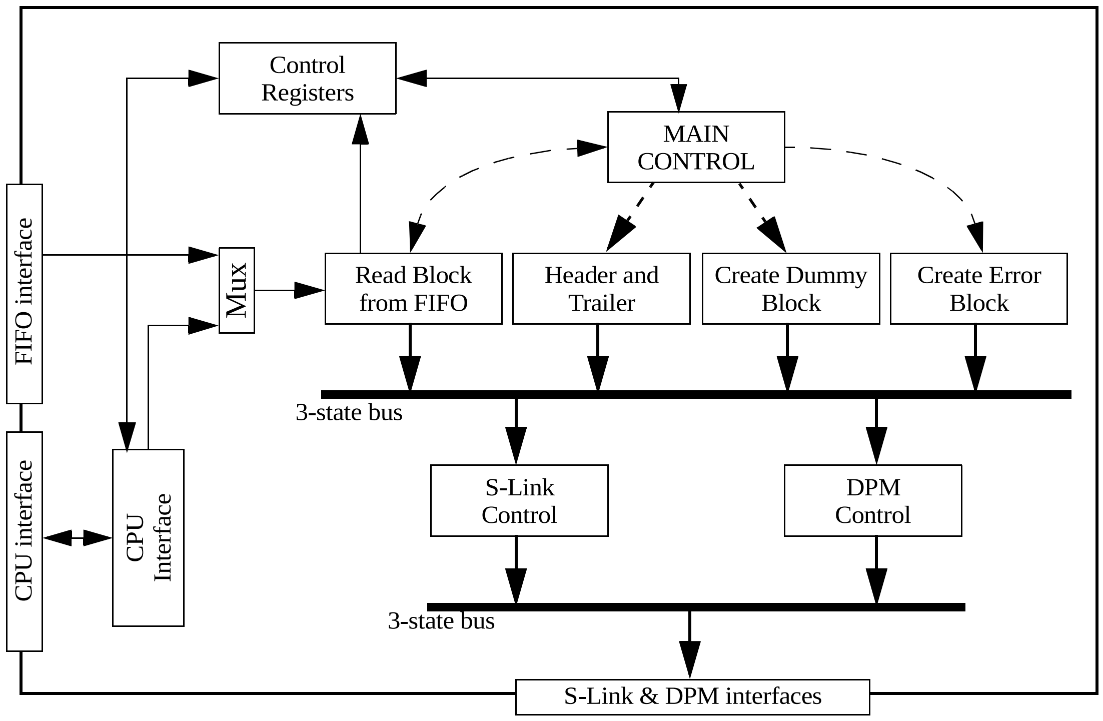


Fig. 4-5. Block diagram of the SEM FPGA logic.

The CPU interface block can be used for several purposes:

- **Test, diagnostics and Debugging**: A multiplexer (see Figure 4-5) can be configured to read data from the CPU instead of from the FIFO bus, allowing the simulation of error transmission and frame corruption, as well as other scenarios like self-test patterns.
- **Monitoring**: The CPU can access a wide range of counters and status registers in the FPGA.
- **Control**: Turning links on/off, forcing self-test or debug mode, time out window acceptance, etc. can be controlled remotely via LAN.

4.2.2. *Sub-event buffer*

Four 16-bit wide memory chips form a 64-bit memory bank, whilst two memory banks form the SEB. The 64-bit wide DPM has a storage capacity of 512 KByte, upgradable to 1 MByte in the prototype. This is insufficient as justified in “Sub-event buffer” on page 60, but enough to evaluate the RU prototype design.

The sub-event buffer implementation is shown in Figure 4-6. Signal integrity simulations carried out with Cadence's SpecctraQuest pointed out the need to buffer the memory address signals driven from the FPGAs (SEM and EBI), due to the eight-load topology (see 16LVC16244 buffers in the figure).

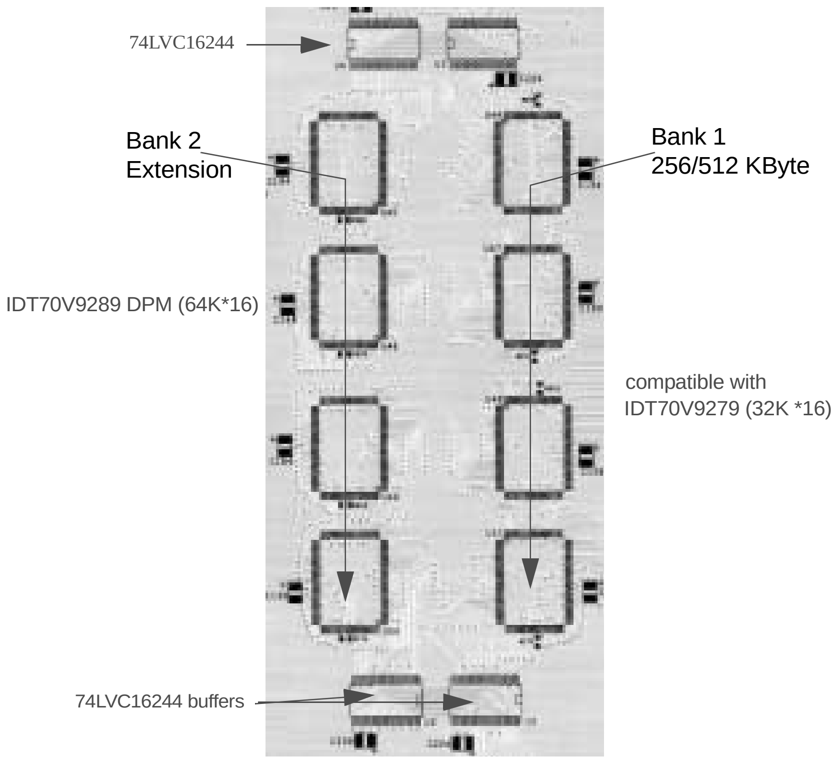


Fig. 4-6. Sub-event buffer implementation.

4.2.3. Output stage

Two parallel, 45-kgate FPGAs [Lucent3TP12] with embedded 32/64-bit 33/66-MHz PCI core are used to implement the EBI output stage logic. In the largest available physical package (352-pin BGA) up to 187 user I/Os are available. Normally only one FPGA is needed, the second one is present to accommodate extra user logic in case there is not enough with a single device. Despite the fact that the embedded PCI interface in the output stage FPGAs is 64-bit wide externally, the data path with the user logic is hardwired to 16 or 32 bits only, thus limiting the performance to 264 MByte/s maximum. The embedded PCI interface can be used in two modes: dual-port (Figure 4-7 right, master and target interfaces share 32-bit FIFOs) and quad-port (Figure 4-7 left, master and target interfaces have independent 16-bit FIFOs).

The RU contains two PCI buses (see Figure 4-8):

- The **secondary PCI bus** (33 MHz, 32 bits, 5V signaling environment) is fully configured and controlled from the MCU and is intended to connect the MCU to an auxiliary PCI card for debugging purposes. The MCU's on-board LAN chip is also connected to this PCI bus.
- The **primary PCI bus** (33/66 MHz, 32/64 bits, 3.3V or 5V) is controlled and configured by either the MCU or the EBI FPGA, depending on which of them is the PCI configuration agent

(selected via a switch on the RU motherboard). Other on-board switches allow to select the electrical environment and bus speed.

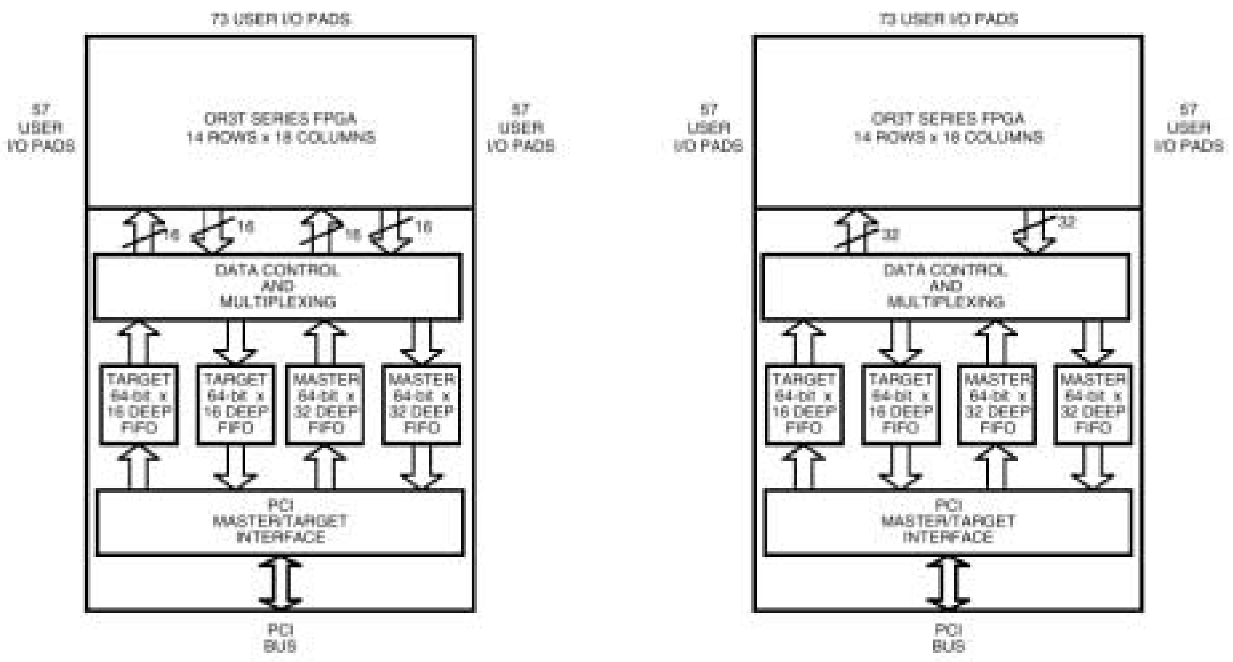

73 USER I/O PADS
57 USER I/O PADS
OR3T SERIES FPGA 14 ROWS x 18 COLUMNS
16
32
DATA CONTROL AND MULTIPLEXING
TARGET 64-bit x 16 DEEP FIFO
MASTER 64-bit x 32 DEEP FIFO
PCI MASTER/TARGET INTERFACE
PCI BUS

Fig. 4-7. Dual-port and quad-port modes in the ORCA3TP12 FPGA PCI interface.

The output stage architecture (Figure 4-8, only one EBI FPGA shown) allows the MCU to take part in the DAQ event-building protocols if desired, or even to take full control of the sub-event building tasks (at a reduced performance) by using the EBI FPGA just as a memory interface.

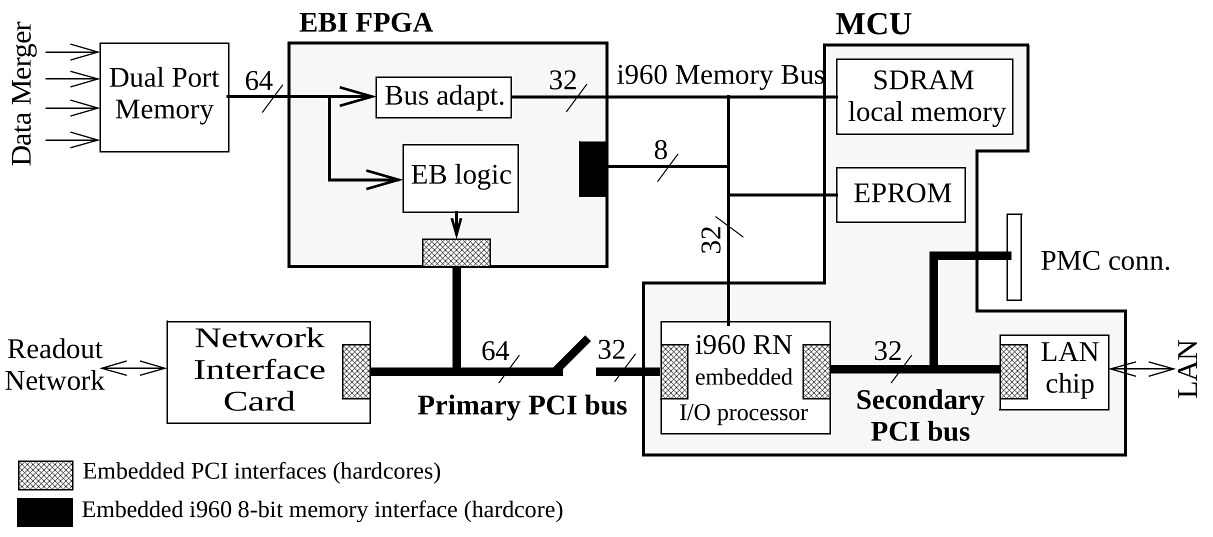


Fig. 4-8. Output stage block diagram.

Nevertheless, full-speed operation at 66 MHz implies 3.3 V electrical environment, which is not compatible with the MCU. Thus, a PCI switch based on CMOS pass transistors[1] is used to isolate the MCU from the primary PCI bus. Output stage performance, in both push and pull scenarios, is analyzed in Chapter 3.

4.2.4. Synchronization, front panel signals and RU reset

A bussed DPM_Full return line is required from the EBI FPGAs to the SEM FPGA to avoid overwriting valid data in case of a SEB full condition. This signal, together with the FIFO full lines in the input stage determine the status of the RU throttle output. This is a TTL open-collector line on a NIM connector available in the RU front panel.

A line named MAILBOX is driven by the SEM FPGA to indicate that a new sub-event block has been written into the SEB and thus the MAILBOX memory position in SEB has been updated. The output stage FPGAs react to a toggle in this signal by reading the MAILBOX position, recalculating free space in SEB and driving DPM_Full to the appropriate level.

The RU implements a hierarchical reset system, which is described here briefly. At the top of the hierarchy, a reset request line driven by the MCU, a reset push button and a NIM reset input in the front panel are fed into a logic or gate that triggers a monostable circuit. This monostable generates a logic low pulse with a duration between 3.3 and 12 ms: the ***system reset*** signal. The NMI[1] input in the MCU is connected to this *system reset* signal. This way, the MCU can be configured either to ignore the *system reset* or to reset itself together with the rest of the RU.

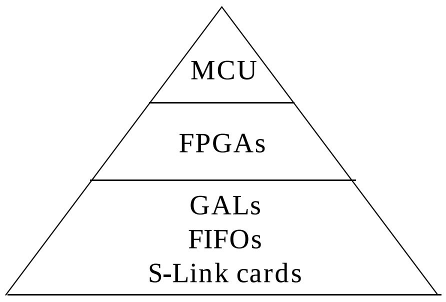


Fig. 4-9. RU Reset hierarchy.

The FPGAs are located at the second level of the reset hierarchy. The *system reset* signal forces the FPGAs to enter a reprogramming sequence. FPGAs can be programmed from a Flash memory (default) or from the embedded i960 interface. Alternatively, the MCU can force a reset in a specific FPGA just by setting bit 0 in the FPGA Control Register to ‘1’ via the i960 interface. The difference between ‘reprogramming’ and ‘reset’ in a FPGA is that the latter only resets flip-flops and I/Os, while the former also forces a reload of the FPGA configuration bitstream.

The S-Link receiver and transmitter cards, the FIFOs and the GALs are at the third level of the reset hierarchy. All these elements are reset by the FPGAs after a reset/reconfiguration or during normal operation.

1. IDT QuickSwitch family devices, see http://www.idt.com/products/pages/Bus_Switches-QS32XR245.html
1. Non-Maskable Interrupt.

4.2.5. The Monitoring and Control Unit

The functionalities of the MCU in the first RU prototype are:

- **FPGA configuration loading**. At power up, the FPGAs read the configuration data from an on-board flash EPROM. The three FPGAs are connected in a daisy-chain fashion, with the SEM FPGA as the first member in the chain. At any time, the MCU can force a reconfiguration in the FPGAs via a GPIO signal. EBI FPGAs can be also configured from the MCU via the PCI bus.
- **FPGA monitoring and control**. The MCU can access user-defined internal registers in all FPGAs via the i960 local bus.
- **Programmable clock configuration**. The two clock domains in the RU (input stage FPGA and output stage FPGAs clocks) are generated in a programmable clock generator (AMI FS-6370-1). This chips can be configured via an I2C interface or can read already programmed parameters from an internal EPROM. This means that the clock generator only needs to be programmed when a change is required in the output frequencies. As a change in the output frequencies might cause glitches, this operation should only be performed while the FPGAs are in reset mode.
- **PCI configuration agent**. The MCU is the configuration agent in the secondary PCI bus, and can also act as the configuration agent in the primary PCI bus (selectable via a jumper on the MCU board). The MCU can force a reset in the primary PCI bus. This reset does not affect the MCU itself unless it is programmed to do so when attending a NMI.
- **Experiment Control System interface**. This functionality is over the entire control of the MCU as no interface signals between the MCU and the on-board logic exist for this purpose. An on-board Ethernet chip is used to interface the ECS network.
- **Readout Unit debugging and protocol testing**. This functionality requires that the MCU accesses internal registers in the FPGAs and has read/write access to the DPM. These functions ar performed via the i960 local bus.
- **Participation in sub-event building protocols**. As an intelligent device attached to the primary PCI bus, the MCU can take part in the sub-event building.

The chosen embedded CPU for the RU prototype is the **PXECORE** module from CompuLab [Pxecore00]. It consists of an Intel 80960-RD RISC CPU, a 10/100 Ethernet controller, embedded Flash EPROM, DRAM memory and additional features like up to 50 programmable I/O lines (see Figure 4-10).

The embedded I/O processor controls and bridges two 32-bit PCI buses. The interconnection between the MCU and the RU is shown in full detail in Figure 4-11. The PXECORE card (133 by 86 mm forma factor, similar to the CMC form factor) interfaces with the RU via seven non-standard connectors, which makes the RU dependent on this company-specific development.

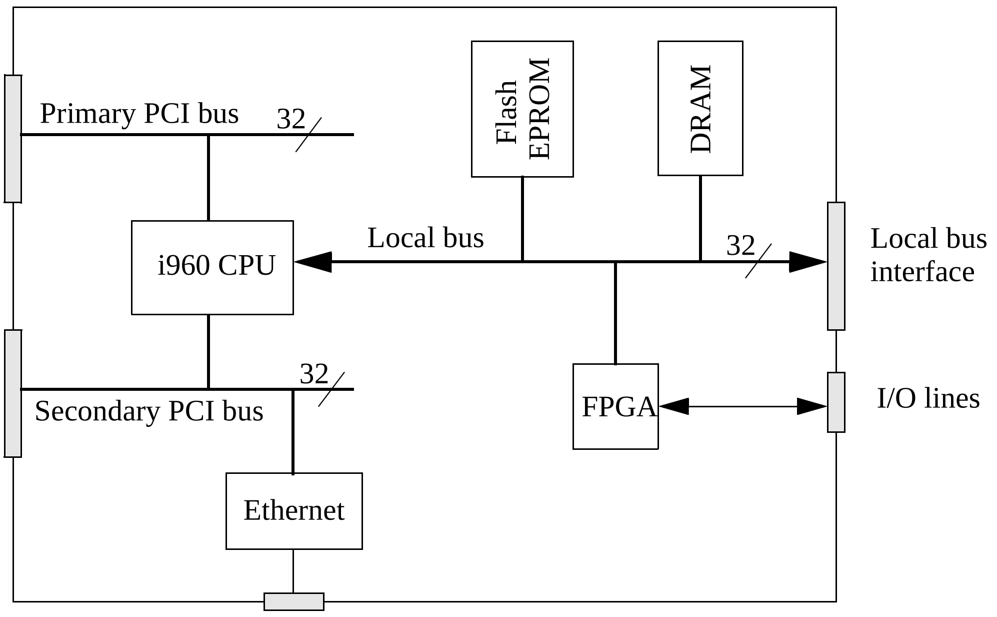


Fig. 4-10. MCU block diagram for the first RU prototype.

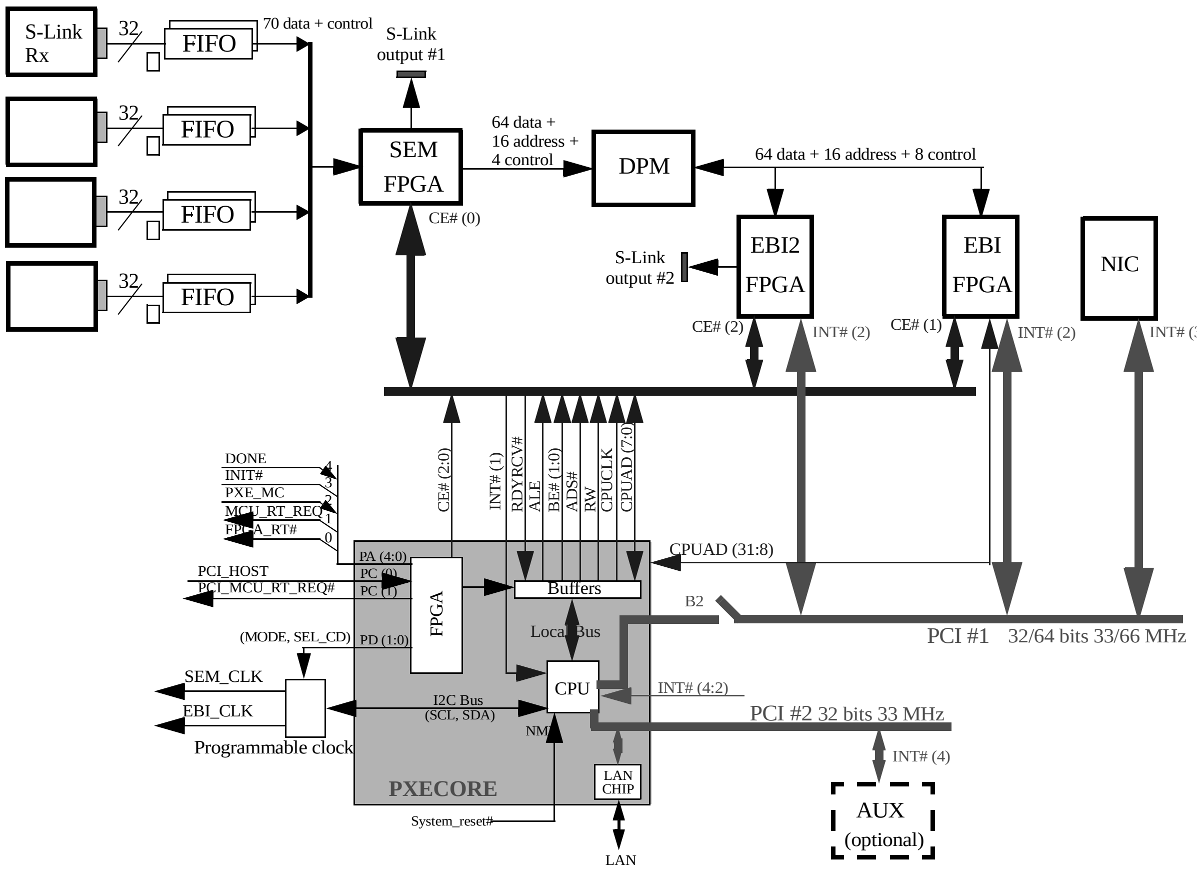


Fig. 4-11. MCU-RU interconnection in the first RU prototype.

4.3. Conclusions

The Readout Unit prototype board was successfully tested (S-Link input to S-Link output sub-event building in a buffered application) in January 2000, demonstrating the feasibility of the RU concept despite some minor design errors. The PCI-based sub-event building could not be tested, as the PCI core in the FPGAs required an specific configuration software for application development which was not delivered until summer. Nevertheless, the PCI bus could be tested in order to validate the physical design.

The chosen MCU turned out to be problematic for the RU application. The MCU LAN had problems, the RU FPGAs configuration via the dedicated i960 bus did not work, and several bugs were found on the card. Moreover, being a single-company product with non-standard connectors, there was no possibility to use a different module. This lack of flexibility was found unacceptable.

Other drawbacks in this first RU prototype are summarized below:

1. Too many components and connectors resulted in a very expensive 12-layer PCB requiring the highest fabrication quality level.
2. Too many switches and jumpers on the board provided flexibility, but restricted ease of operation and maintenance.
3. The sub-event buffer capacity (1 MByte maximum) was below the requirements to perform trigger throttling (see “Flow control in LHCb” on page 35).
4. Two FIFO chips per input channel were required in the prototype, which added a significant cost factor.
5. The SEB could not be written from the output stage, reducing the flexibility.
6. Lacking of TagNet interface, the module could not be used for the VELO application.

A module design revision was needed to reduce the module cost, remove some unnecessary features, increase the sub-event buffer size and to include the features required by the VELO application, resulting in the RU-II board described in the next chapter.

CHAPTER 5 The Readout Unit II

“There is in all things a pattern that is part of our universe. It has symmetry, elegance, and grace (...) We try to copy these patterns in our lives and our society, seeking the rhythms, the dances, the forms that comfort”

From the novel “Dune” by Frank Herbert.

5.1. Overview

A module design revision was needed to reduce the module cost, increase the sub-event buffer size and to include the features required by the VELO application, resulting in a modified architecture as shown in figures 5-2 and 5-3. The RU-II’s capability to cope with high trigger rates relies on the use of parallel data paths rather than on wide buses.

The experience with the first RU prototype resulted in a list of improvements and simplifications for a re-design of a final RU module. The resulting RU-II module was considerably less expensive due to reduction of expensive components (like DPM memory, FIFO memory, Flash memory and non-standard mezzanine cards). Also the diversification of components was further reduced, in particular a single, new type of FPGA[1] was used with the benefit of a higher gate count and faster PCI interface. The use of PCI as configuration bus for all FPGAs made the company-specific interface to a MCU card redundant and by using the PMC standard also for the MCU, the special connectors could be avoided.

As a consequence of the simplifications, the new 9U module (figure 5-1) could be routed in eight PCB layers only, compared with the twelve of its predecessor. The addition of one extra FPGA in the output stage allows to test a new, parallel input architecture. Extensive analog signal integrity checks were performed, as a result of which the addition of a PCI bus bridge for the delicate 66 MHz PCI bus became mandatory.

1. Lucent OR3LP26, an FPGA with true 64-bit 66-MHz PCI core and 60-120 Kgates of user programmable logic.

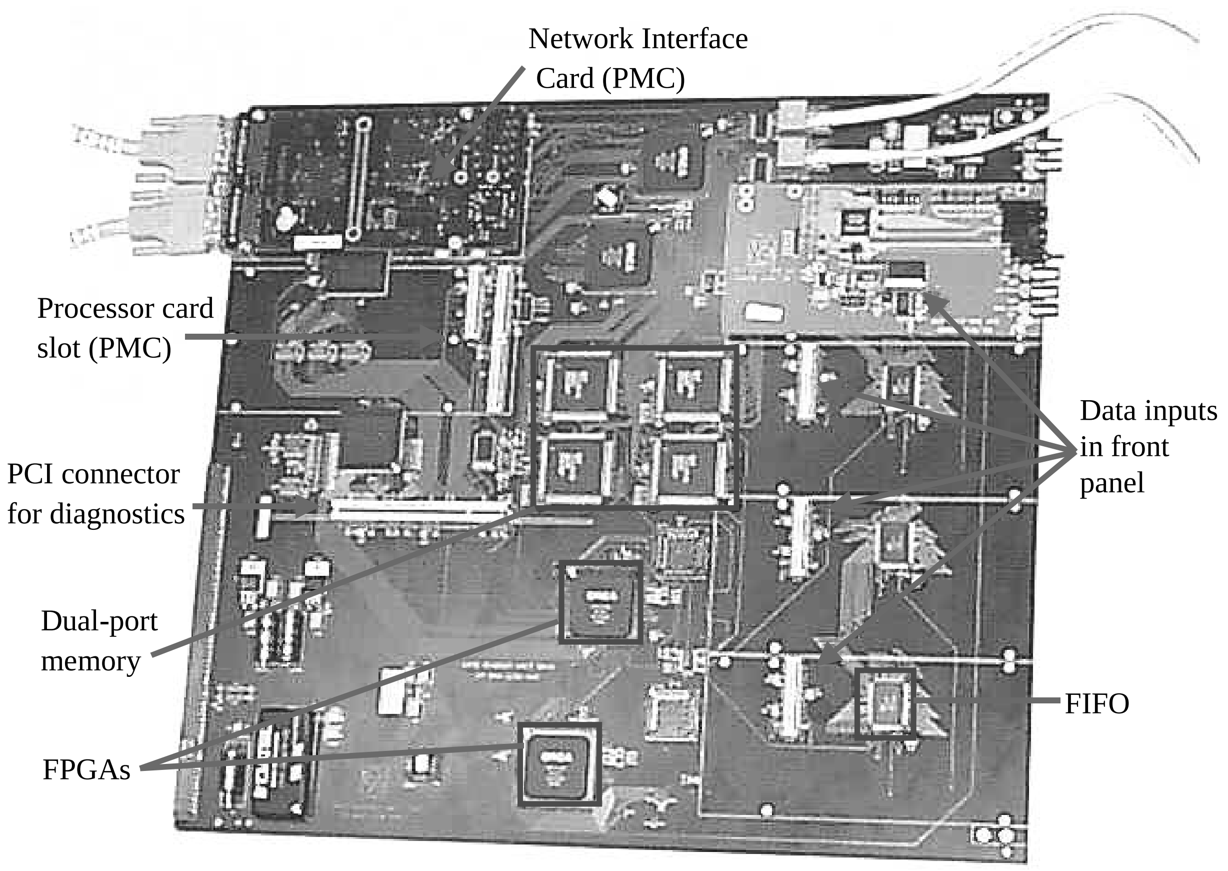

Network Interface Card (PMC)
Processor card slot (PMC)
PCI connector for diagnostics
Dual-port memory
FPGAs
Data inputs in front panel
FIFO

Fig. 5-1. Readout Unit II.

The improved features on the Readout Unit II are:

- The scheduling network TagNet is implemented as required by the VELO application.
- The FPGA configuration via the Intel i960 bus is replaced by PCI bus.
- The MCU is implemented as a standard PMC card, allowing multiple vendor choices and removing entirely non-standard connector dependencies.
- Only a single FIFO is used per S-Link input.
- The DPM size is increased to reach two megabyte (enough for fast throttling with low latency) with only four chips, instead of the eight in the first prototype.
- A diagnostic PCI connector has been added, allowing to plug PCI bus analyzers or PCI service cards.
- The SEB is now readable and writable from both input and output stages (could not be written from the output stage in the first prototype).
- Most of the mechanical configuration jumpers are replaced by programmable lines in the MCU, thus allowing remote-jumpering via the LAN interface.
- Some of the diagnostic logic which was required for the prototyping and development is removed.
- The S-Link output option in the first RU prototype which allowed to directly drive S-Link from the SEM (Figure 4-2) is removed since all RU applications require buffering for sub-event building.

- Newer FPGA devices of higher gate count (120 kgates) are used [Lucent3LP26]. All FPGAs are equipped with 66-MHz PCI cores such that their interconnection with the MCU's PCI bus is straightforward. Additionally, these FPGAs include true 64-bit interfaces between the user logic and the PCI core, thus allowing full PCI speed.

A picture of the revised RU module in shown in Figure 5-1. It has less components and includes a TagNet bus interface. One S-Link card is inserted on the top front emplacement, further emplacements are visible with a single FIFO each. One NIC card is inserted on the top rear PMC/S-Link slot. The PMC slot emplacement below can be used for the MCU card. Below the PMC there is the PCI diagnostic connector. The DPM consists of four ICs (center). Input stage logic is implemented in two FPGAs (bottom). The two upper FPGAs implement the output stage logic.

5.1.1. Architecture

Each of the two horizontal slices in the RU-II architecture (Figure 5-2) consists of a 32-bit input stage performing 2-to-1 data merging (sub-event merging, SEM), a dual-port buffer with 32-bit input and 64-bit output buses and a 64-bit output stage. The two slices generate two data paths that are connected at the output stage together to perform sub-event building (event-building interface, EBI). This generates four logic blocks, implemented in four FPGAs which are interconnected via their PCI bus pins in a PCI bus hierarchy of which the MCU card is host.

Figure 5-3 shows the full RU-II architecture. The input stage, buffer and output stage implement the main data flow. In a vertical view, control and monitoring paths are ortogonally implemented.

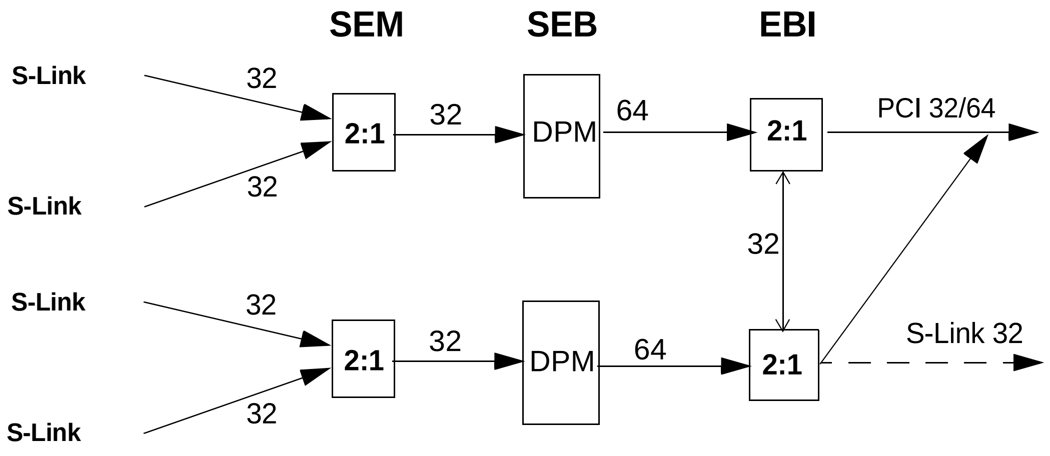


Fig. 5-2. RU-II horizontal architecture.

The MCU, hosting the primary PCI bus, is connected via PCI bridges to the input and the outputs stages. Two PCI bridges[1] create three PCI segments which can operate independently. Only the PCI bus segment at the output stage takes part in the event data transfer (from EBI

1. Intel 21154-BC 32/64-bit 33/66 MHz PCI-to-PCI bridges. This chips include a programmable PCI arbiter.

FPGAs to the Network Interface Card). The other two PCI segments are used for configuration and control.

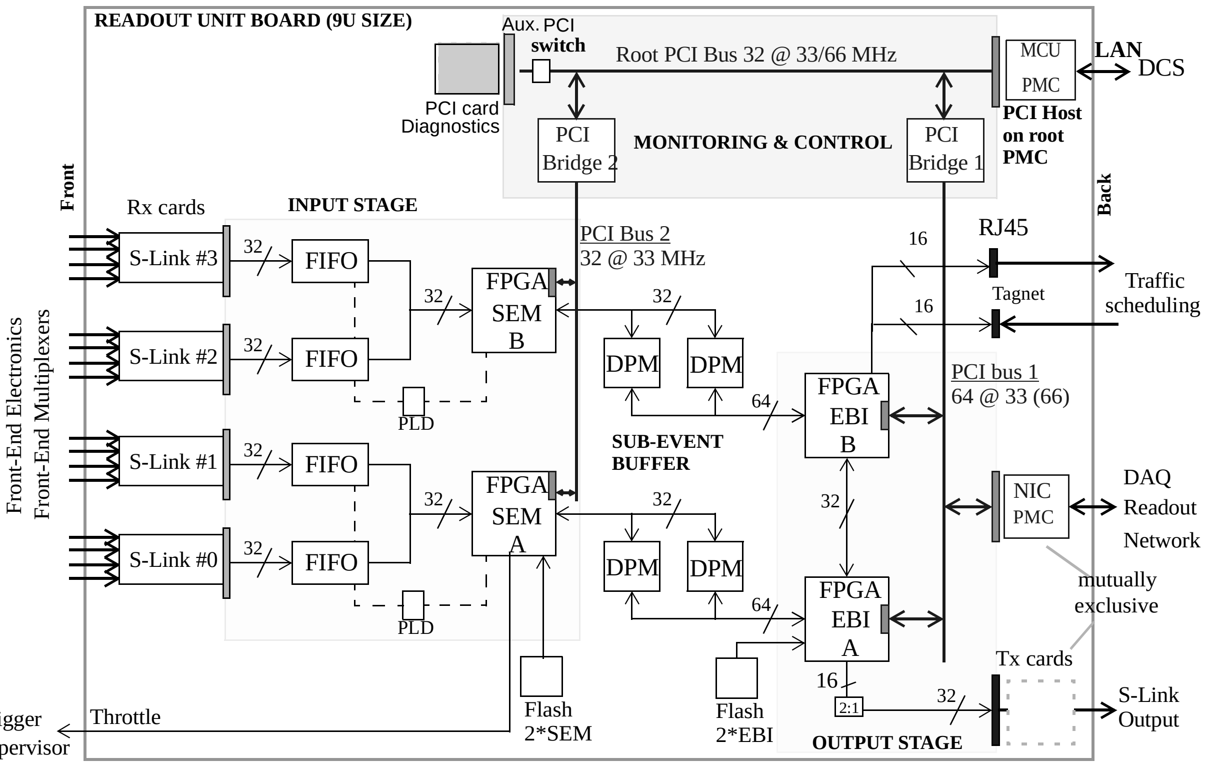


Fig. 5-3. RU-II architecture.

5.2. *Input stage*

Figure 5-2 allows to identify the two parallel input stage data paths: incoming frames from two links are read and processed in an FPGA and stored in DPM. In order to implement part of the sub-event building tasks in the input stage, the FPGA must merge incoming frames with the same event number into a single data block and a single descriptor in DPM. The resulting data structure in memory is shown in Figure 5-4.

The pointer in the directory block (DYB) points to the start of the event fragment from link 0. The length field in the DYB descriptor results from the addition of the length of the two fragments. The flag field is the bitwise logic OR of the flags for the two event fragments. This is seen by the output stage as a single data block and a single descriptor.

A simplified version[1] of the proposed algorithm for the DAQ application is shown in Figure 5-5. This algorithm can be modified for the VELO application frame format (which might be a simplified version of the DAQ transport data format) and can be also generalized and used in the FEM application. In fact, the latter option has been implemented in VHDL and allows to turn the

1. All error handling mechanisms disabled, including support for Error Blocks.

DAQ application into a particular case of the FEM application with only one fragment per event number and link. This means that the same FPGA code can be used in the input stage for both DAQ and FEM applications, simplifying code maintenance and development.

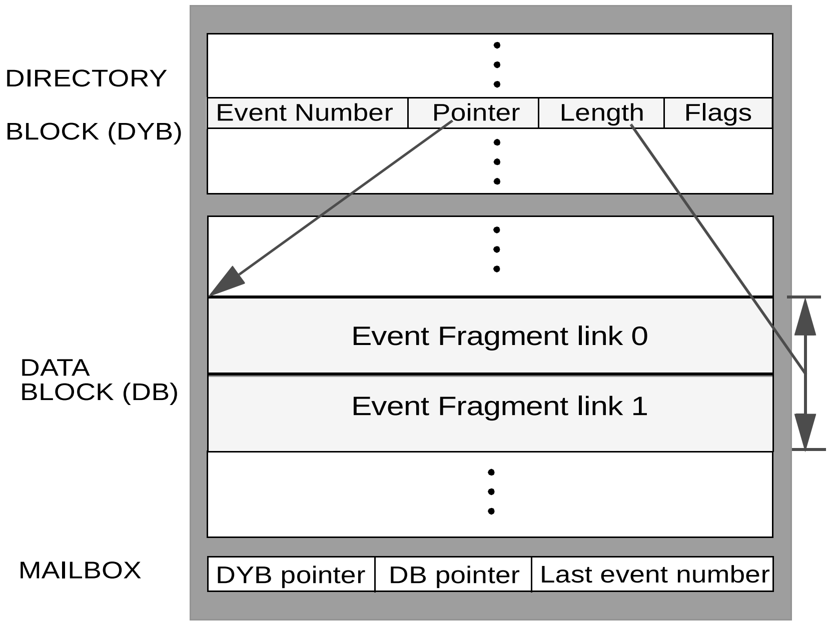


Fig. 5-4. Event fragment merging in the sub-event buffer.

The algorithm in Figure 5-5 is divided into three tasks (identified as dashed boxes in the figure):

1. Read one sub-event from the first FIFO and store its data block in the sub-event buffer (top left dashed box).
2. Read one sub-event from second FIFO and store its data block in the sub-event buffer so that the two data blocks are merged (bottom left box).
3. Write a single entry for the merged data block in the Directory Block (bottom right box), update the MAILBOX memory position and toggle the mailbox signal to let the output stage know that a new sub-event can be built.

Task 3 for the event *N* is concurrent with task 1 for the event *N+1*, resulting in zero overhead for memory management and 2-to-1 data merging (except for the turnaround cycle needed when switching between FIFOs).

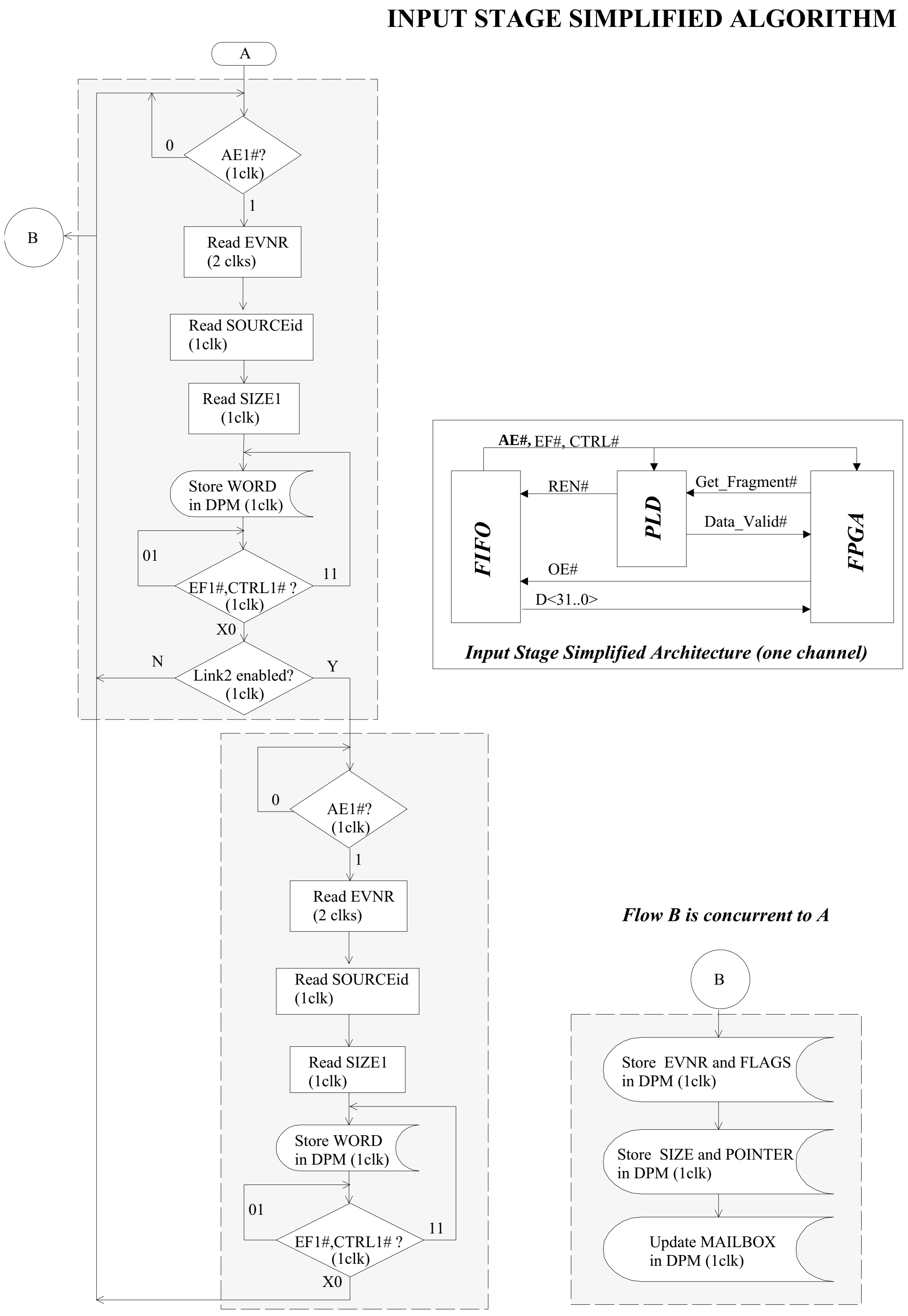


Fig. 5-5. Input stage algorithm for the DAQ application.

5.2.1. Input stage design

As described above, the input stage has two independent parallel data paths, each one consisting of two S-Link inputs, two FIFOs, one FPGA and one small auxiliary PLD[1] for FPGA support. A detailed structure of one of these data paths is shown in figure Figure 5-7.

Each S-Link input receives data which are directly written into a FIFO. The value of three S-Link signals [S-Link] (LDOWN#, LDERR#, LCTRL#) are written into the 36-bit-wide FIFO together with the 32 data bits coming from the S-Link connector and the FF# (FIFO full) flag bit coming from the FIFO. This additional information is used by the auxiliary PLD for frame synchronization and delimiting (LCTRL# signal) and by the FPGA for error detection (LDOWN#, LDERR# and FF# signals).

The FPGA monitors the state of the PAE# and EF# flags[2] (Programmable Almost Empty and Empty Flag, respectively) in both FIFOs. The value of the programmable flag can be changed depending on the application. The FPGA reads the FIFOs with the help of the PLD. The PLD is needed because the FPGA is not fast enough to immediately deassert REN# FIFO inputs (Read ENable) when an end of frame has been reached and thus an additional word (the beginning of next event fragment) would be read accidentally from FIFO. The timing diagram in Figure 5-6 illustrates this situation.

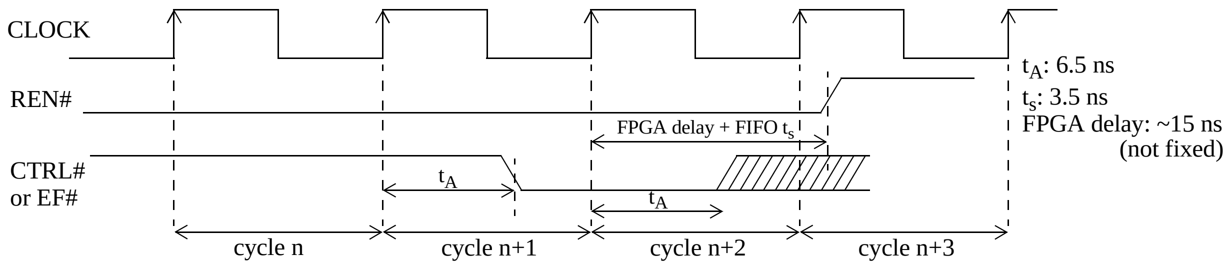


Fig. 5-6. Timing diagram illustrating a potential timing error avoided using a fast auxiliary PLD.

At the beginning of cycle n+1 (rising edge of the clock) a new word containing the end of the frame is read form FIFO (CTRL# asserted after the FIFO's access time delay t_A). The FPGA cannot deassert REN# fast enough and the result is that a new unwanted word is read on cycle n+2. The FPGA latches CTRL# at the beginning of cycle n+2 and its internal delays (**not fixed**) added to the FIFO setup time are too close to a clock period of 45-50 MHz (RU-II target frequency). Thus, a second unwanted word might be read or not depending on program-dependent FPGA delays[3]. A similar discussion can be applied to the EF# signal.

1. Altera MAX 7032AE-4 in a 44-pin PLCC, the smallest device in the MAX 7000 family. With a pin-to-pin max. combinatorial delay of 4 ns, 32 macrocells and JTAG in-system re programmability, is well suited for the RU-II application.
2. Four offset values (7, 15, 31 and 1023 words) for the PAF# and PAE# flags can be programmed via two lines (FS0 and FS1) from the SEM FPGAs.
3. This problem was a design bug if the first RU prototype that limited the maximum clock speed in the FPGA and thus the input stage performance.

In order to overcome this limitation, an auxiliary fast PLD is used. Fast PLDs have **fixed** pin-to-pin delays as short as 3.5 ns, making it possible to deassert REN# prior to the rising edge of the clock at the beginning of cycle n+2 and thus avoiding the read of unwanted extra words. Another way to avoid the problem is defining a frame format in which two empty (non-valid) words are added at the end of the frame. The resulting overhead could be acceptable for DAQ but not for VELO application. So, the addition of one PLD per input stage data path is preferable to adding framing overhead and would allow safe operation at frequencies of 50 MHz and beyond.

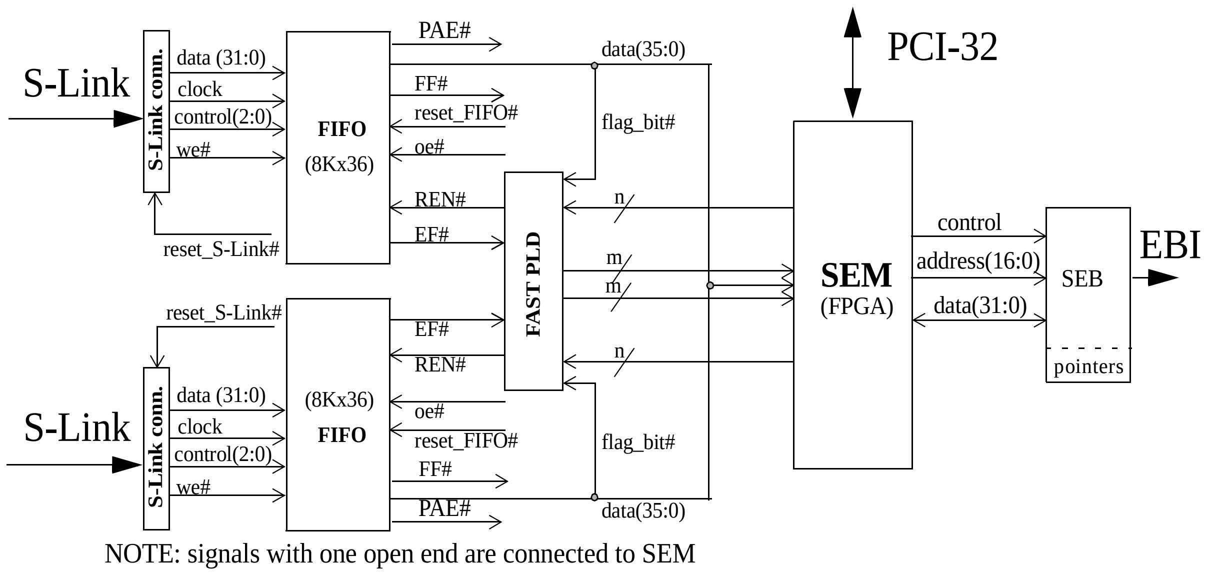


Fig. 5-7. RU-II input stage.

The input stage block diagram is depicted in Figure 5-7. S-Link receiver cards write incoming event fragments (EF) into their associated FIFO memories. The SEM identifies EF pairs that share the same event number according to an input algorithm described in the previous section. A combined sub-event block is written into the Data Block section of the SEB buffer together with a pointer to its Directory Block (see also “Memory management scheme” on page 61 and Figure 5-4).

The auxiliary PLD may be used for something else than deasserting **REN#** after an end of frame or a FIFO empty condition. For this purpose, a number of lines are connected between the FPGA, the PLD and each FIFO (Figure 5-8):

- **ctrl#** is an S-Link dedicated signal used for frame delimiting.
- FIFO read control signals: **oe#** and **ren#** (output and read enable).
- FIFO flags: **EF#**, **PAE#**, **FF#** (empty, programmable almost empty and full flags).
- PAE# threshold in FIFOs is adjusted via **fs(1:0)** signals coming from the SEM FPGA.
- S-Link defined control signals (**url(3:0)** and **reset_S-Link#**).
- **get_frgm#**: the FPGA asks the PLD to read a new event fragment from FIFO.
- **start#**: the FPGA enables the PLD operation after reset or power up.
- **gal_rt#**: the FPGA resets the PLD logic.

- **wait_sync#**: provides a way to read one FIFO when the other gets empty in the middle of a fragment read. Not used in the current version of the PLD logic.
- **dvalid#**: the PLD validates FIFO data words with this signal.
- **aux**: Additional signal reserved for future application.

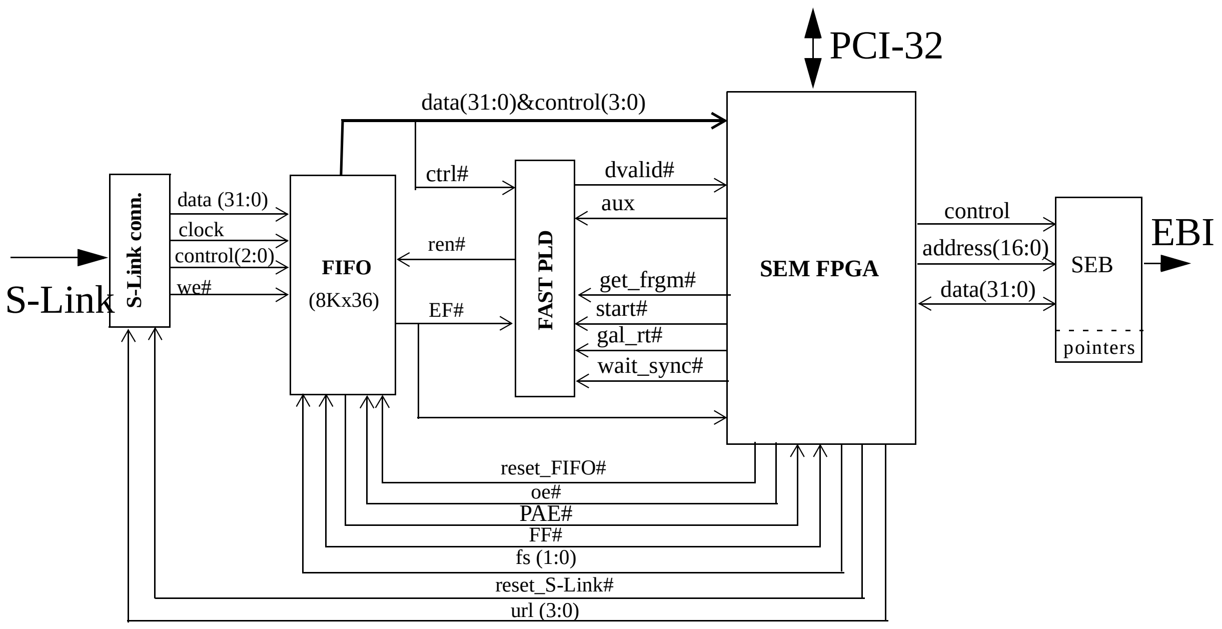


Fig. 5-8. Detailed interconnection scheme between the FIFO, PLD and FPGA.

The currently implemented 8-state machine implemented in the PLD is shown in Figure 5-9 and its operation is summarized below:

1. Wait for the FPGA to assert **get_frgm#** and **start#** (***idle*** state in the diagram).
2. Read selected FIFO until a start of frame is found (states ***ask_FIFO***, ***start_frgm***, ***icmplt_frgm*** and ***end_frgm***). A start of frame is identified by two consecutive control words (i.e., trailer followed by first word in header).
3. Read a complete event fragment from FIFO and validate each data word with **dvalid#** (state ***end_frgm***).
4. If FIFO gets empty during step 3, insert wait states (***empty1***, ***empty2***).
5. Go to step 2 (and thus read a new event fragment) when the FPGA needs it (state *next_frgm*).

Thus, frame boundary synchronization and fast REN# control are transparent to the SEM FPGA, simplifying its logic.

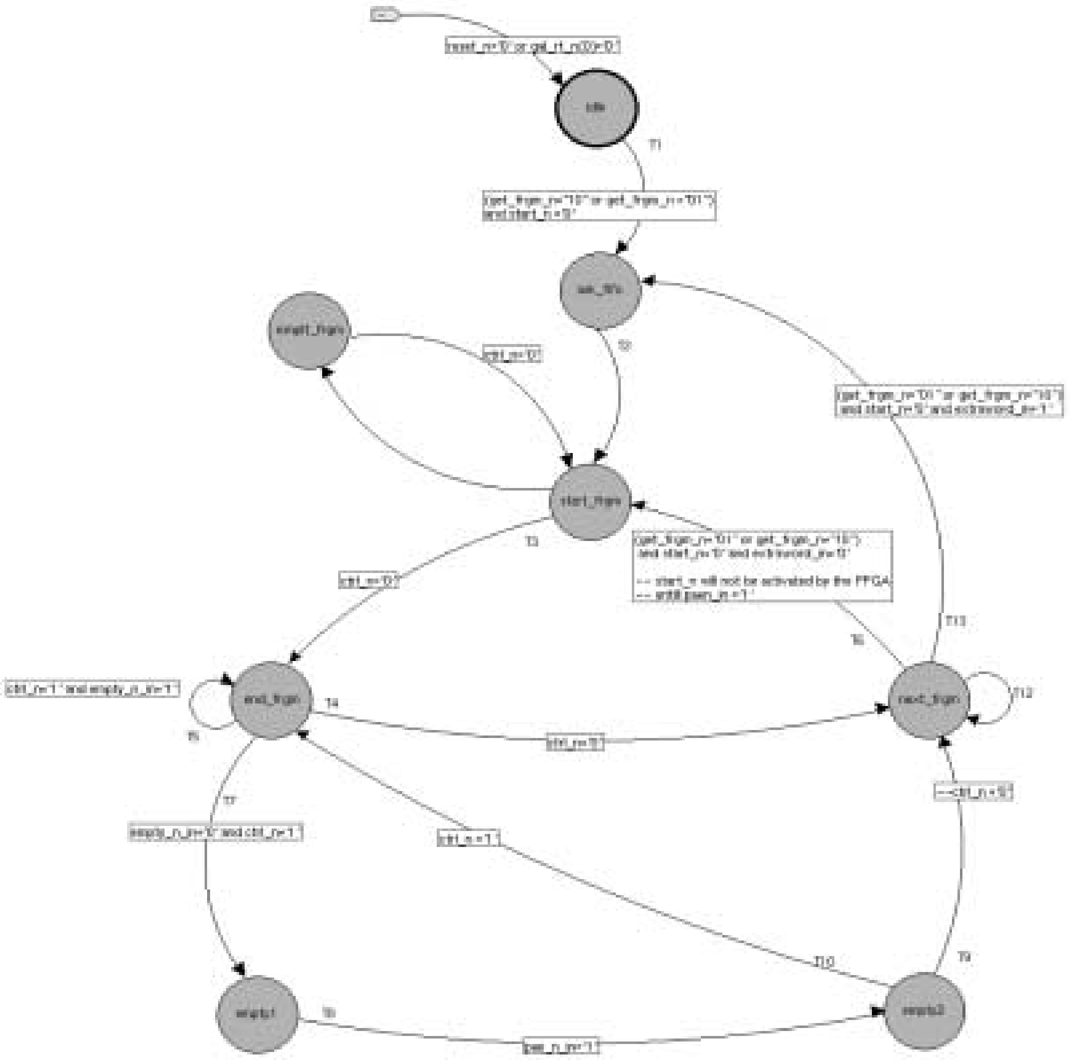


Fig. 5-9. PLD state diagram.

5.3. *Sub-event buffer*

The sub-event buffer is based on dual-port memory for simultaneous access from both ports (SEM and EBI FPGAs). It consists of two independent memory banks with a 32-bit interface to the input stage and a 64-bit interface to the output stage. Figure 5-10 depicts the structure of one of the two memory banks.

The SEM FPGA performs 32-bit interleaved operations to memory so that two consecutive memory positions lay in different ICs. The EBI FPGA performs 64-bit operations reading both DPM ICs in parallel. This is a straightforward mechanism for interfacing 32-bit with 64-bit data buses. A self-incrementing address counter in the memory ICs is loaded and enabled independently for each port via the ADS# (Address Strobe) and CNTEN# (CouNTer ENable) signals.

Even if normal operation requires only write operations to the left port (input stage) and read operations from the right port (output stage), bidirectional buses and R/W# control lines are provided for special applications like debugging or test.

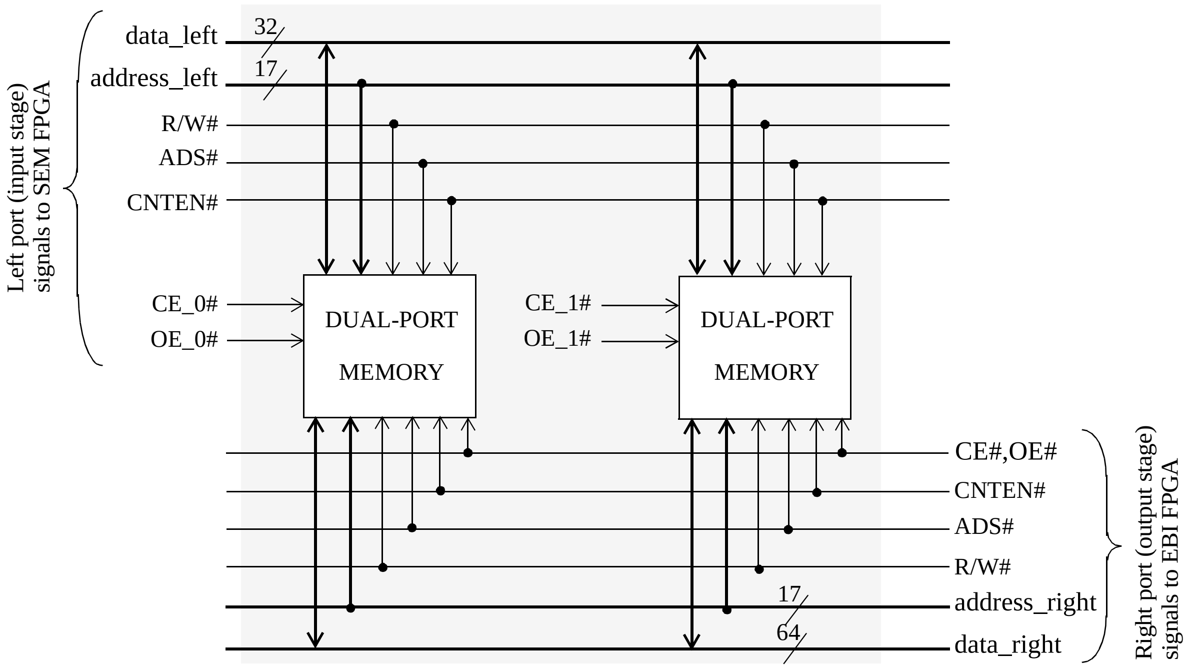


Fig. 5-10. Structure of a dual-port memory bank.

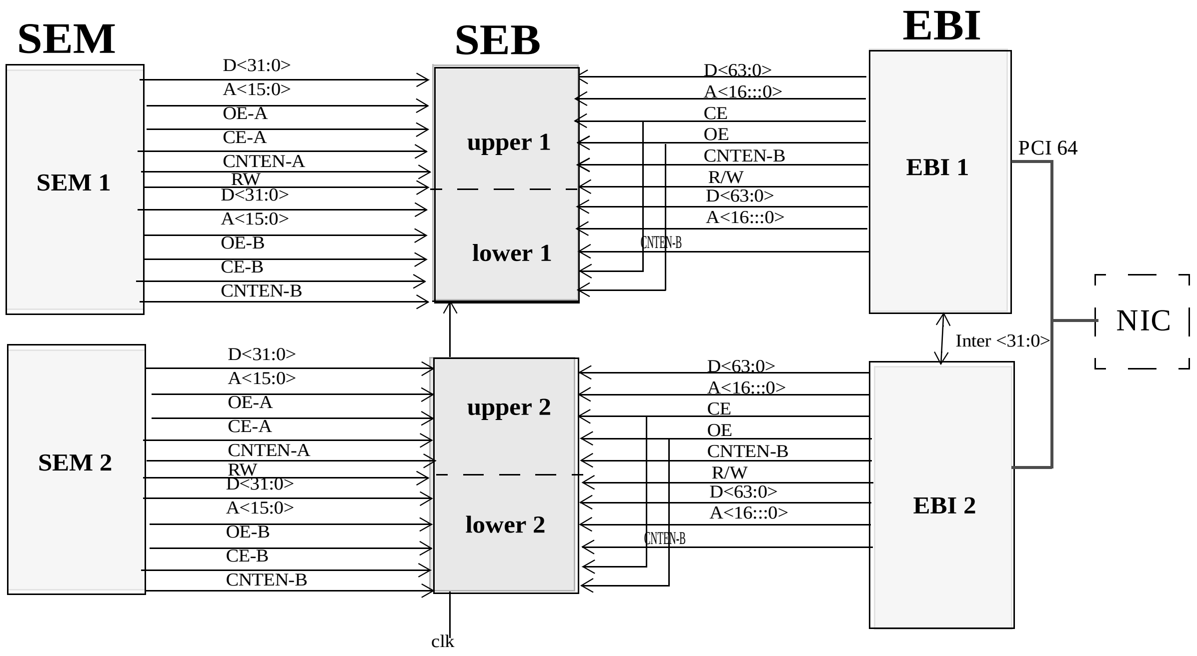


Fig. 5-11. Dual-port memory connection scheme in the SEB.

The picture below in Figure 5-12 shows the four SEB chips on the RU-II board. The chosen device is a 36-bit-wide 512-KByte true dual-port memory (IDT 70V3599S) in a 208 PQFP package. Four chips for a 2-MByte sub-event buffer.

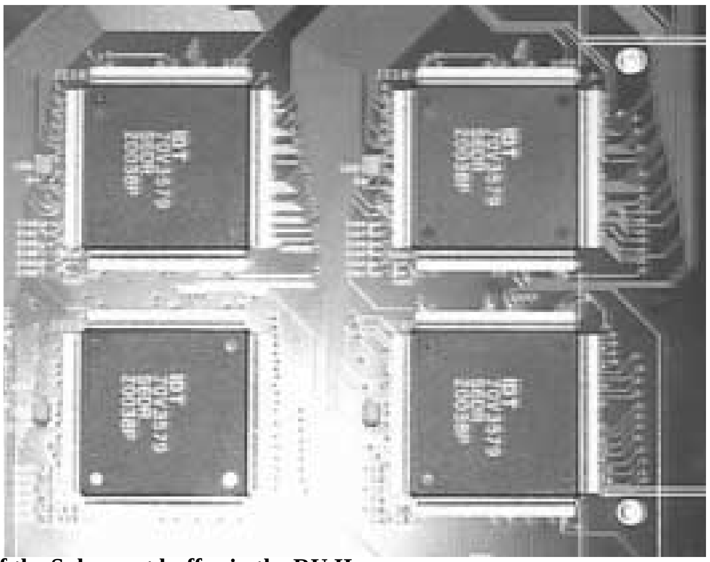


Fig. 5-12. Photography of the Sub-event buffer in the RU II.

5.4. PCI subsystem

The PCI subsystem hierarchy is depicted in Figure 5-13. It consists of three PCI bus segments:

- One 3.3V 32-bit 33/66 MHz **primary** PCI bus that interconnects the MCU, the two PCI bridges and an auxiliary PCI connector. The MCU acts as the configuration agent for the whole PCI subsystem. Alternatively, a single-board computer can be plugged on the auxiliary PCI connector (5V 32-bit) to act as the host. PCI-compliant switch/level-converter ICs are used to interface between this 5V device and the 3.3V bus and to isolate the auxiliary connector when in 66 MHz operation. This isolation reduces the bus length and enhances signal integrity to allow 66 MHz operation.
- One 3.3V 32-bit **secondary** PCI bus that interconnects the two SEM FPGAs and the corresponding PCI bridge This bus has been designed for 33 MHz operation as is intended only for FPGA monitoring and control from the MCU and for SEM FPGAs intercommunication.
- One 3.3V 64-bit **secondary** PCI bus that interconnects the two EBI FPGAs, the NIC and the corresponding PCI bridge. This bus has been designed for 33/66 MHz operation and serves as the communication path between the RU output stage and the NIC. Control and monitoring from the MCU is also foreseen (the impact on the output stage performance is marginal, due to the low bandwidth required for this task).

Apart from the benefits of traffic isolation, the two PCI bridges reduce the electrical loading and the bus segment lengths, thus allowing 66-MHz-mode[1] compliant PCI operation in both the primary and the 64-bit secondary bus. This compliance has been verified with post-layout signal

1. 66-MHz mode refer to clock speed between 33 and 66 MHz, according to the PCI specification.

integrity simulations using Cadence Specctraquest. The bridges provide the clock and arbitration signals for the devices on their secondary buses. The MCU is the arbiter in the primary bus. This allows any FPGA to become a PCI master and access all the PCI devices in the RU.

The chosen device is the Intel 21154, a transparent bridge[1] that provides 32-bit or 64-bit primary and secondary interfaces, independently. This device is available in two versions: AC and BC. The former operates up to 33-MHz, whilst the latter can be used in 66-MHz PCI systems. Both versions are compatible with the RU II design.

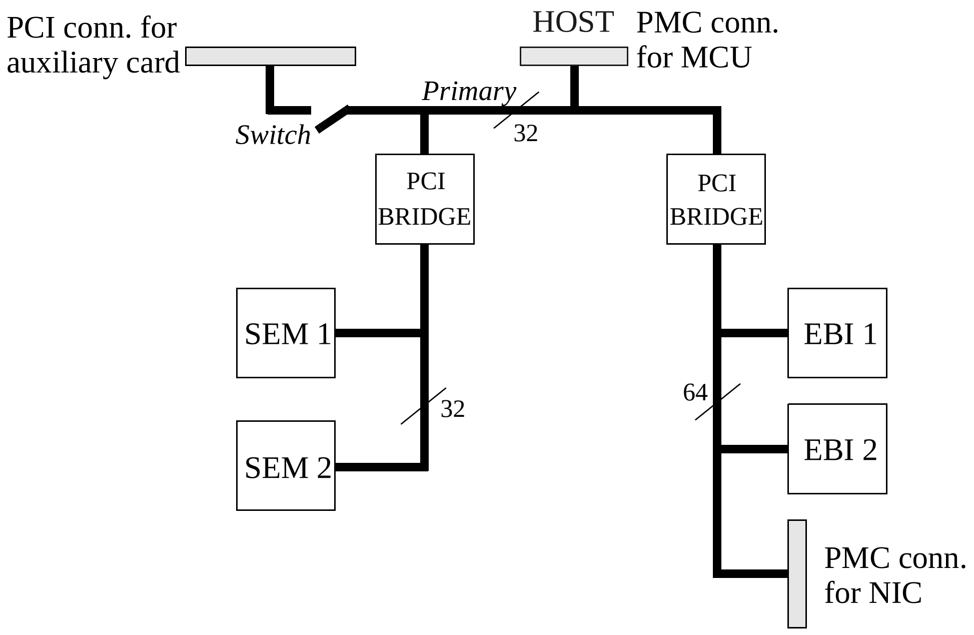


Fig. 5-13. PCI subsystem in the Readout Unit II.

Currently, there is no transparent PCI-to-PCI bridge that allows a higher clock frequency on its secondary PCI bus than on the primary bus[2]. This implies that, for 66-MHz sub-event building, a 66-MHz PCI host (i.e., the MCU) is needed. There are several 66-MHz host-capable[3] PMC embedded CPUs commercially available, though the imminent widespread of 66-MHz PCI will augment the number of such designs. The MCU in the RU-II module is a 33-MHz PCI device.

1. There are two kinds of PCI bridges: transparent and non-transparent. Transparent bridges allow to see initialize the devices on the secondary bus from a host on the primary bus. Non-transparent bridges hide and initialize all the devices on its secondary bus, allowing multi-host PCI systems.
2. The opposite is true: secondary bus clock can be slower than primary bus clock. The justification can be found in the PC architecture, in which the closer to the microprocessor (host), the higher the needed bandwidth.
3. Compliant with VITA-32 extensions for PMC.

5.5. Output stage

Event fragment merging is performed in the input stage, so the output stage will only have to assemble two data blocks into a single sub-event. Each data block resides in a separate DPM bank and is only accessible for the corresponding EBI FPGA (see Figure 5-14). The input stage FPGA code can be very similar for the three applications (DAQ, FEM and VELO), but the output stage FPGA code must be completely different from one application to the other. In this section, the sub-event building algorithm and hardware for each application is briefly described.

The 64-bit output stage architecture is shown in Figure 5-14. Two FPGA chips can access their associated SEB buffers via 64-bit data buses in parallel. The FPGA internal PCI cores allow to combine the PCI outputs together on the PCI bus as required for the tandem master operation in the L1-VELO application. The high speed data path between FPGAs and NIC is decoupled by a PCI bridge from the other PCI bus segments which are mainly used for initialization and communication. The TagNet connector for VELO high-rate traffic scheduling is implemented in logic inside one FPGA.

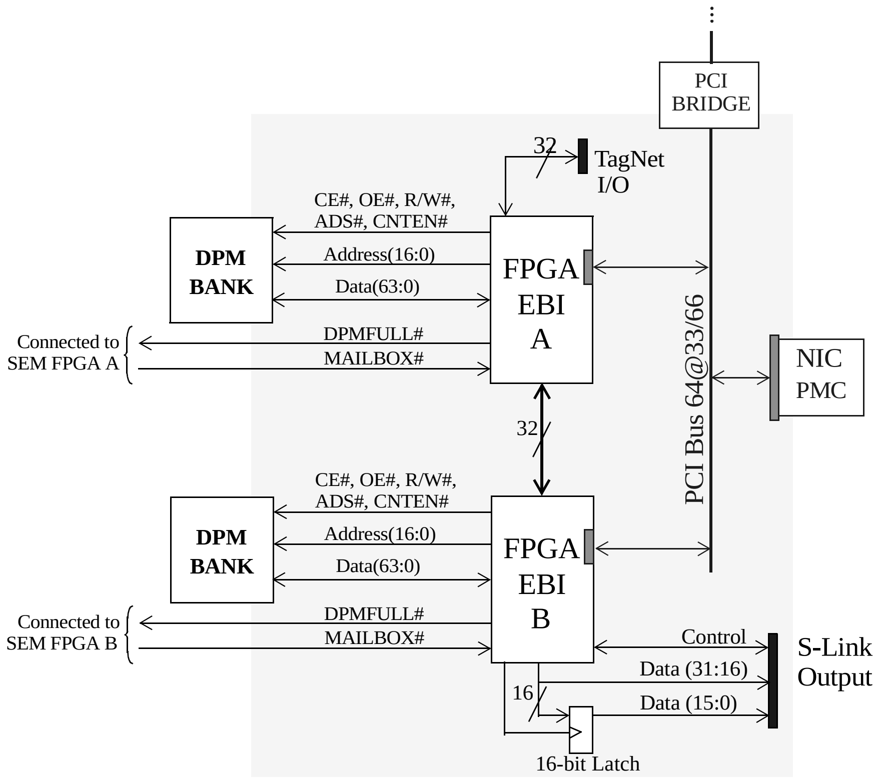


Fig. 5-14. Output stage architecture.

The S-Link and NIC cards share (exclusively) the same PMC slot. For an S-Link output, the sub event data are combined, using the 32-bit bus interconnecting the two FPGAs, and are sent to the S-Link connectors as a single sub event. A 16-to-32 bit demultiplexer (74LVC16374 latch or similar) is required for S-Link due to not enough pins on the FPGA. This is however uncritical as the performance requirement for a link multiplexer is four times less than for a RU (see “FEM application requirements” on page 65).

5.5.1. TagNet interface implementation

The TagNet interface implementation on the RU was decided on a dedicated RU meeting [Müller00] in October 2000 in Heidelberg.

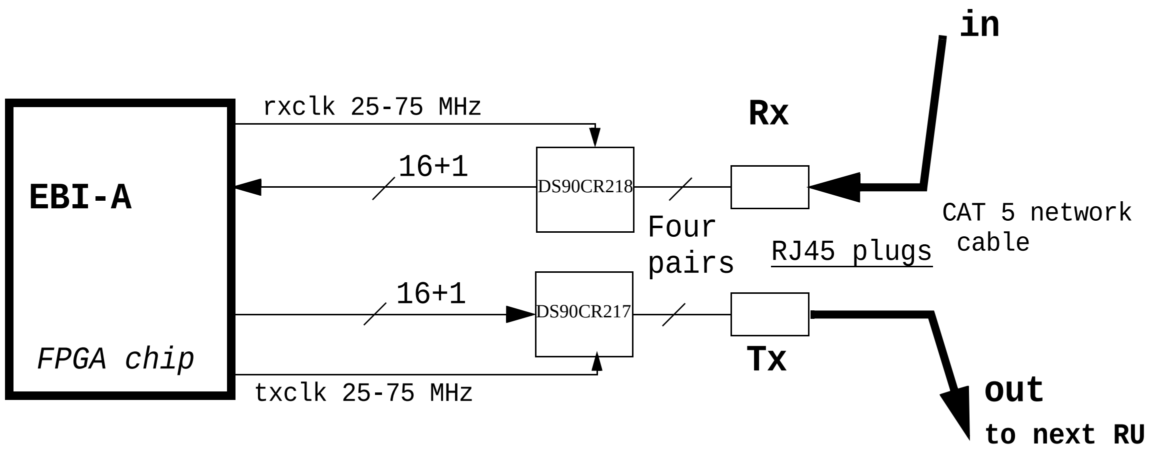


Fig. 5-15. TagNet implementation.

The TagNet interface is implemented on the RU-II board as 16-bit data plus strobe and clock. There is one TagNet input and one TagNet output per RU. At the TagNet output, the 18 lines are multiplexed, converted to LVDS levels and transmitted over a 4-pair cable with RJ-45 connector by an LVDS serializer chip[1]. At the input, the four differential pairs are de-serialized, converted to LV-TTL levels and fed into the FPGA. Standard CAT-5 network cables up to 5 m are used to interconnect RUs in a TagNet daisy-chained ring.

Two possible TagNet formats were proposed, both of which include Hamming error correcting codes (ECC) due to the vulnerability of a system to erroneous traffic scheduling information.

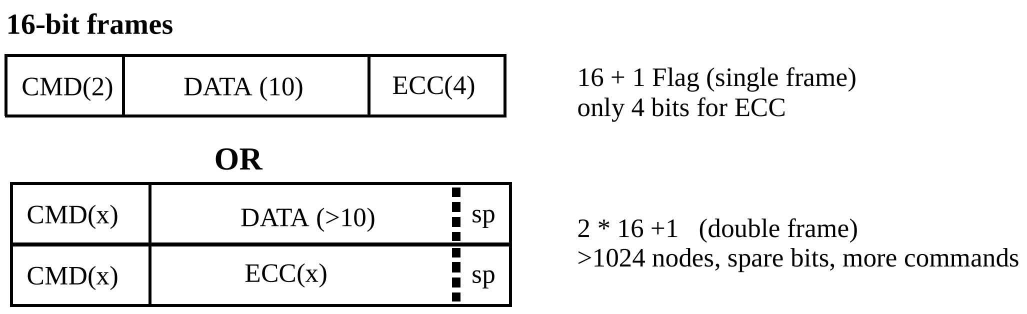


Fig. 5-16. Proposed TagNet formats.

The first proposal uses a single 16-bit frame as information unit, with 10-bit data for up to 1024 nodes. There are 2 bits for 4 commands and 4 bit for ECC. The second proposal uses double 16-bit frames as one information unit, allowing for more commands, and higher level ECC. The two-bit commands defined so ar are:

1. National Semiconductors' DS90CR217 and DS90CR218.

- SEND (code 00). The destination is coded in the DATA field as a 9-bit node and a 1-bit buffer identifiers.
- FLUSH (code 01). The 10-bit DATA field contains an event number.
- ERROR (code 10). Generated by RUs to communicate and error condition to the Scheduler Unit via an 8-bit RU identifier and two bits for error type coding in the DATA field.

5.5.2. Output to S-Link for the FEM application

The output to S-Link uses a 16-bit data path as shown in Figure 5-17. FPGA-B implements the sub-event building process. For this purpose, the dedicated 32-bit bus that interconnects the two EBI FPGAs is used for transferring 16-bit data and control signals (8 lines in each direction).

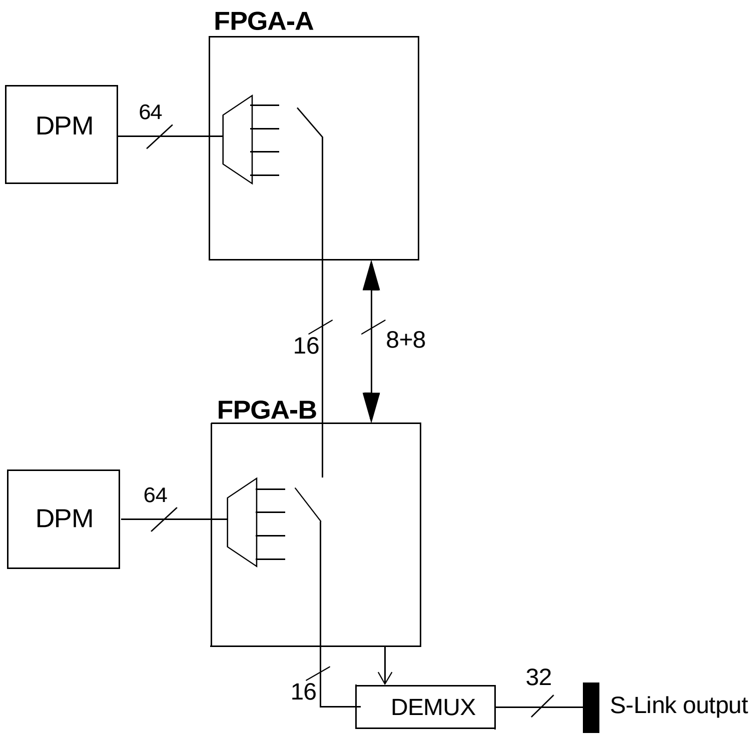


Fig. 5-17. Output data path for the FEM application.

1. FPGA-B reads the next descriptor from DPM and asks FPGA-A via to read its descriptor and send it to FPGA-B.
2. With this information, FPGA-B can generate and send the header words to S-Link.
3. FPGA-B sends its data block to S-Link.
4. FPGA-B asks FPGA-A to read and send its data block via the 16-bit inter-FPGA data lines.
5. FPGA-B adds the trailer word to complete the frame, according to the STF.

The first implemented algorithm on the RU-II module achieved 81 kHz trigger rate for a 45-MHz FPGA operation. Better results (beyond 100 kHz trigger rate) are expected with an optimized implementation.

5.5.3. Output to a PCI Network Interface card for the DAQ application

As described in “DAQ RU application requirements” on page 65, it is foreseen that an intelligent and PCI-master-capable NIC pulls data from the RU performing DMA read operations on the PCI bus. The proposed readout protocol is depicted in Figure 5-5 on page 108 and is adapted to the RU-II architecture (Figure 5-14) by defining a hierarchy in which FPGA-B is the master and FPGA-A is the slave. The former will implement the sub-event building algorithm. Sub-event building is performed as described below:

1. FPGA-B waits for the NIC to be ready to accept a new DMA descriptor list.
2. Once the NIC is ready, FPGA-B writes the DMA descriptor list in the NIC using a PCI write burst operation. When done, writes a register in the NIC: a flag that will signal an interrupt in the NIC’s processor.
3. The NIC will then pull data from DPMs using a scatter-gathered DMA engine. FPGA-A and FPGA-B will read the DPMs in request of PCI read operations initiated by the NIC.
4. When the DMA is finished, the FPGAs will create a new DMA list and then FPGA-B will poll a register in the NIC until it is ready to accept a new DMA descriptor list.

In order to simplify the DMA operation, the DMA list will consists of two blocks: one to be read from each FPGA. This implies that the frame header has to be added at the beginning of the data block in FPGA-B and that the trailer will be added at the end of the data block in FPGA-A.

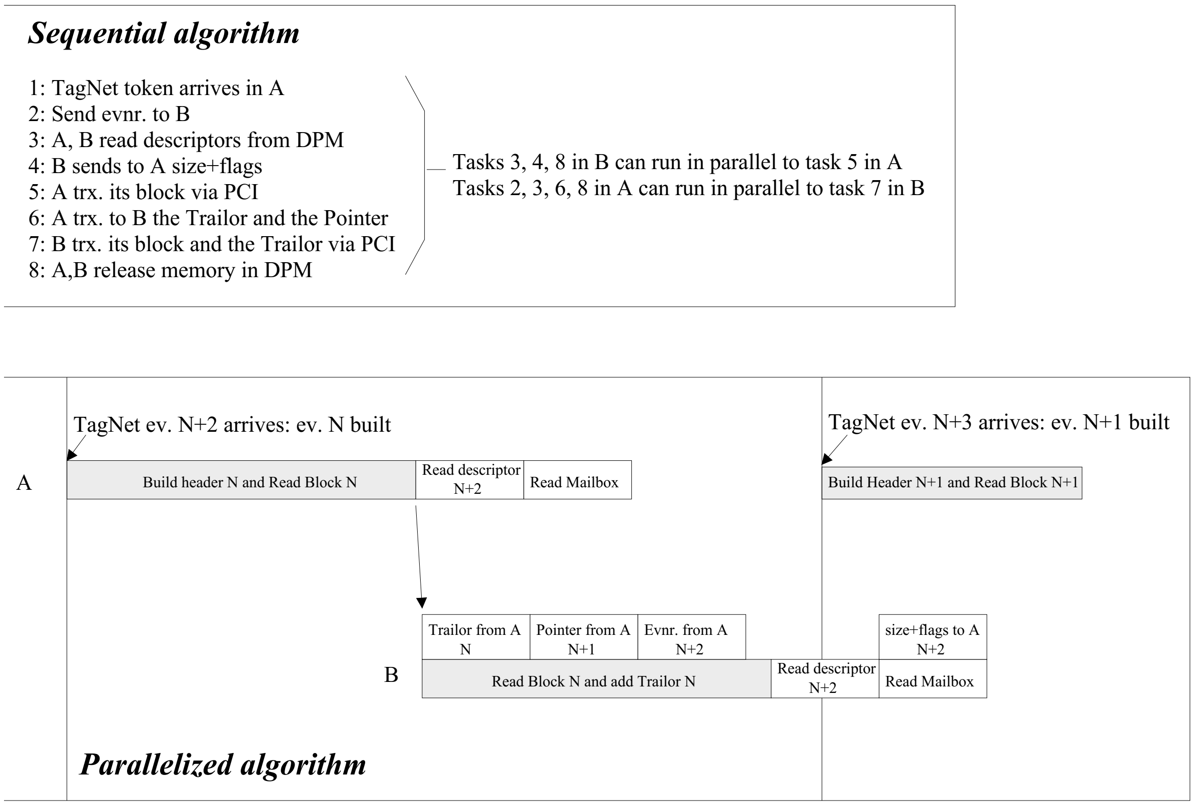
The creation of the DMA list and the addition of the framing words to the data blocks imply the following steps:

1. Both FPGAs read next descriptor from the Directory Block in DPM.
2. FPGA-A transfers its descriptor to FPGA-B using the dedicated 32-bit bus that interconnects the two FPGAs in the output stage.
3. FPGA-B creates the DMA descriptor list and the header and trailer of the frame.
4. FPGA-B transfers the trailer to FPGA-A using the dedicated 32-bit bus while transfers the DMA list to the NIC using the PCI bus.
5. When the DMA is performed, the FPGAs will include the frame header and trailer.

5.5.4. High bandwidth, tandem PCI master operation for the Level-1 VELO application

A possible algorithm, together with its parallelized version are shown in figure Figure 5-18.

In the tandem master operation, one EBI FPGA fills its internal PCI core FIFO with data while the second FPGA is flushing its FIFO to the PCI bus to the NIC card. In order to write a contiguous sub-event into the NIC memory, the FPGA that writes the first part of the sub-event gives a pointer to the other FPGA indicating the next free address in the NIC PCI-mapped memory (A1 in Figure 5-19).



Sequential algorithm

1: TagNet token arrives in A
2: Send evnr. to B
3: A, B read descriptors from DPM
4: B sends to A size+flags
5: A trx. its block via PCI
6: A trx. to B the Trailor and the Pointer
7: B trx. its block and the Trailor via PCI
8: A,B release memory in DPM

Tasks 3, 4, 8 in B can run in parallel to task 5 in A
Tasks 2, 3, 6, 8 in A can run in parallel to task 7 in B

TagNet ev. N+2 arrives: ev. N built
TagNet ev. N+3 arrives: ev. N+1 built
A
Build header N and Read Block N
Read descriptor N+2
Read Mailbox
Build Header N+1 and Read Block N+1
Trailor from A N
Pointer from A N+1
Evnr. from A N+2
size+flags to A N+2
B
Read Block N and add Trailor N
Read descriptor N+2
Read Mailbox
Parallelized algorithm

Fig. 5-18. Output stage algorithm for VELO application.

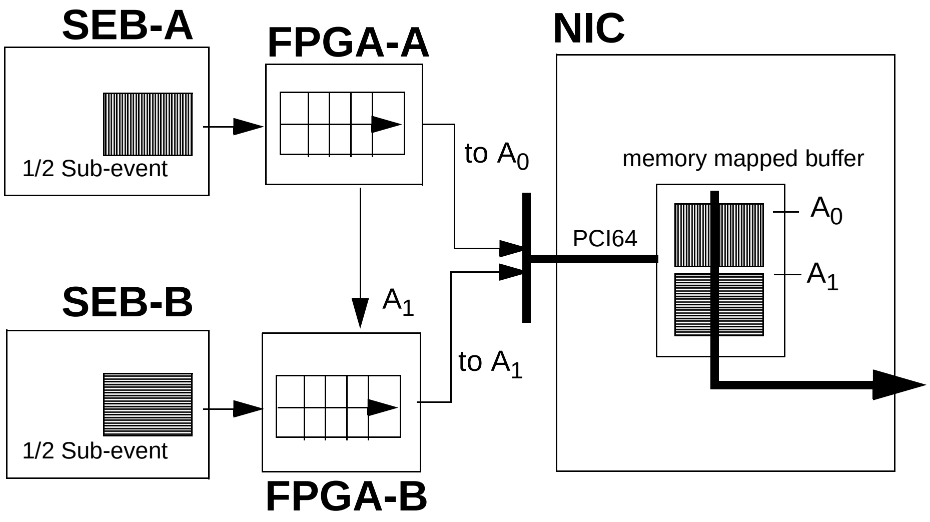


Fig. 5-19. Tandem PCI master operation.

5.6. *Physical implementation*

The RU is implemented on a 9U IEEE-960 card as depicted in figure 5-20. The use of mezzanine card standard is intentional (not necessarily cheaper) to protect against component and standard obsolescence. The six mezzanine emplacements (four S-Link inputs on the front and 2 PMC/S-Link outputs at the rear) are foreseen for CMC mezzanine cards with optional connectors in positions as shown in figure 5-20. The FPGA-based input and output stage logic is in the middle part, with the sub-event buffer in the center. The power conversion logic is close to the Fastbus backplane connector which is only used for power. The reset and throttle I/O logic is at the top of the front panel, the LAN is part of the MCU card. An auxiliary PCI slot is available for standard PCI cards.

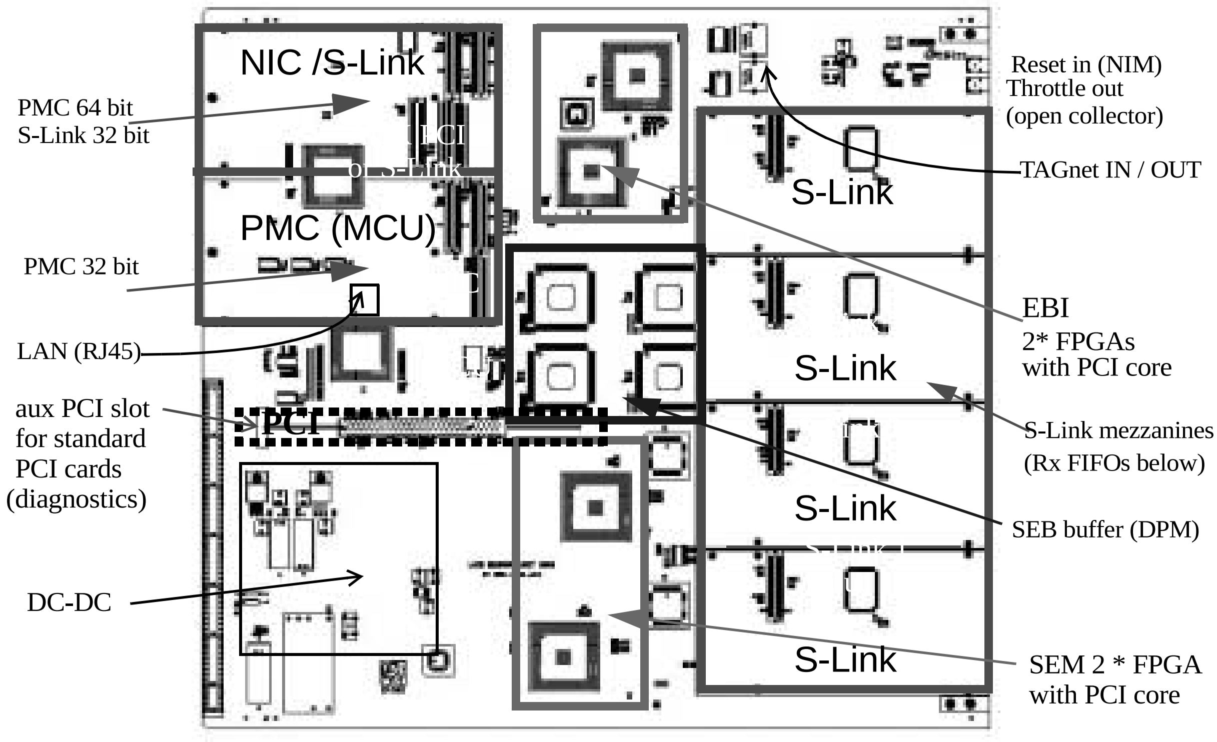


Fig. 5-20. RU-II board arrangement.

The Readout Unit II is a 9U-size (366.7x400 mm) 2.2mm-thick ($\pm 0{,}2 mm$ tolerance) 8-layer board manufactured under class 5 specifications (track width is 6 mils). The stack-up structure is shown in figure 5-21. Copper (in black in the figure) thickness is 35 µm for top and bottom layers and half this value for inner layers, according to standard values in industry. The thickness of the dielectric layers (grey) has been defined in order to keep an uniform characteristic impedance among signal layers and thus avoiding signal reflections caused by impedance mismatches. One millimeter of dielectric between layers 4 and 5 increases the total board thickness to the required 2.2 mm.

Assuming $\varepsilon_r = 4{,}2$ for the dielectric, top and bottom layer impedance is 58.3 Ω whilst inner layer impedance is 55 Ω. These values correspond to unloaded lines. Propagation delays are 59.5 ps/cm for top and bottom layers and 68.3 ps/cm for inner layers.

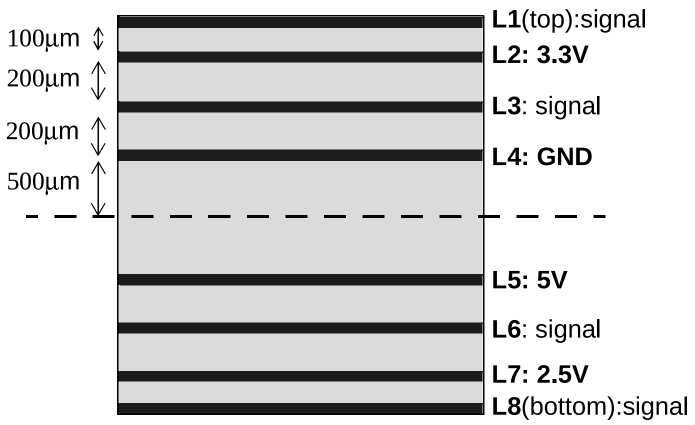


Fig. 5-21. Readout Unit II stack-up.

5.6.1. Signal integrity studies

Both pre-layout and post-layout signal integrity simulations have been carried out using Cadence SpecctraQuest SigXplorer, SpecctraQuest SigNoise and IBIS[1] models from the different chip vendors. The results of these simulations allowed to identify lines that needed terminations, helped to choose and validate bus topologies and determined component placement on the board.

Special attention was paid to the PCI bus subsystem, where simulations showed the need to include a PCI bridge between the primary bus and the SEM FPGA PCI bus and helped to design the topology of the segments that had to run at 66 MHz.

The net topology is either described based on placement assumptions (pre-layout analysis) or extracted from the PCB layout files (post-layout analysis). An example is shown in Figure 5-22 where an FPGA pin (model orca_bmz_OB12) drives two DPM ICs (model 70v9279pf_70v9279z_pf_io0_mdl). Trace parameters (length and width) are also shown.

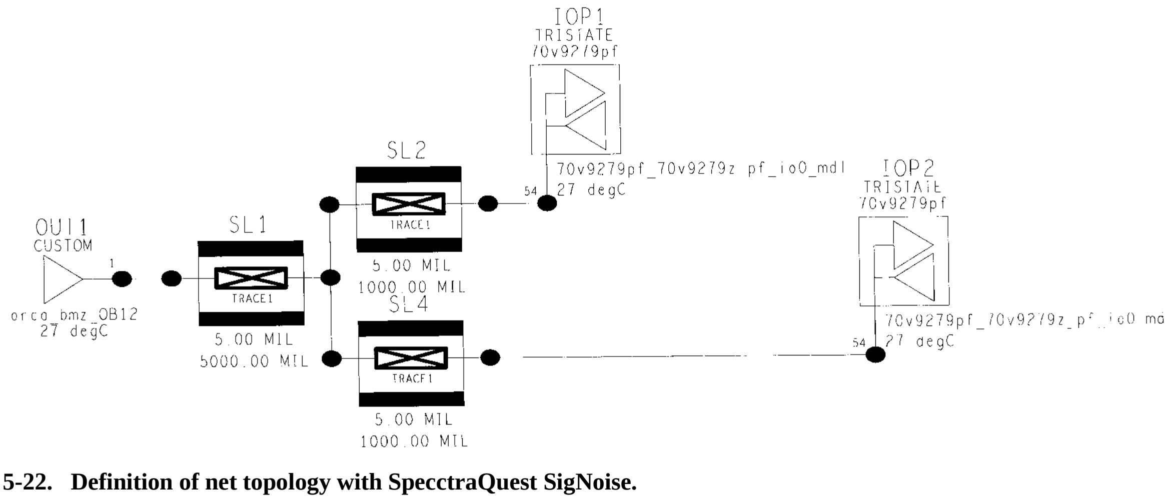


Fig. 5-22. Definition of net topology with SpecctraQuest SigNoise.

1. IBIS is an ANSI standard for integrated circuit input/output buffer specification, used for analog simulations including signal integrity. See http://www.eda.org/ibis/ for more information.

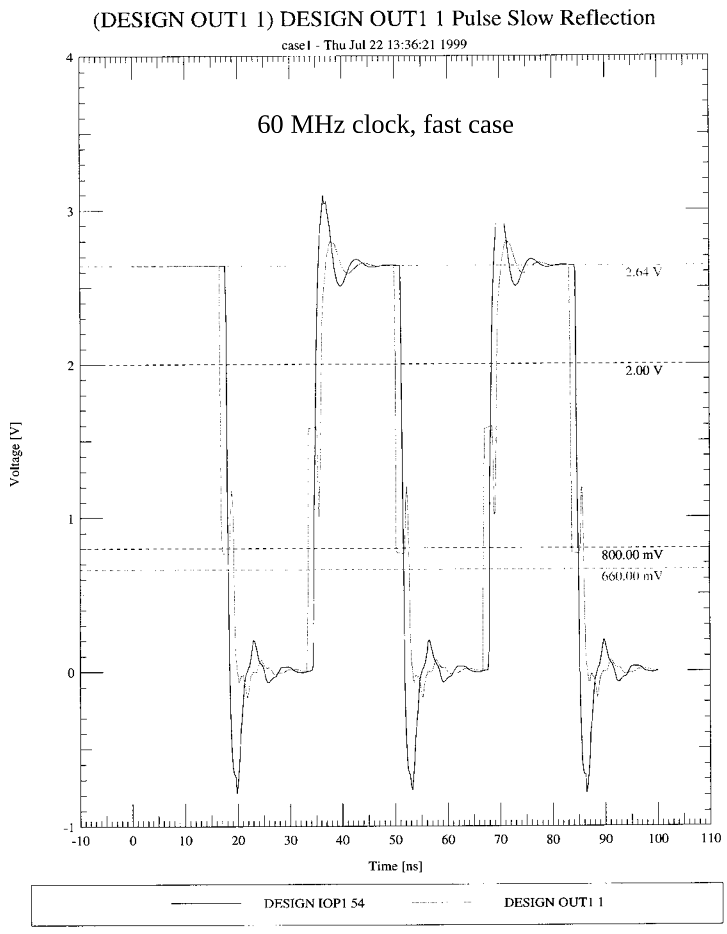


Fig. 5-23. Simulation of the net corresponding to figure 5-22.

Once the net topology is defined, it is possible to perform signal reflection simulations, like the one shown in figure 5-23 which corresponds to the example in Figure 5-22.

5.7. Connectivity and crate environment

Figure 5-24 shows the connectivity of a DAQ RU in a IEEE 960 crate environment[1]. The Experiment Control System network is connected via a local LAN distributor box. The input data cards, normally unidirectional S-Links, are connected on the front panel, using S-Link as interface. The fast throttling signals are transmitted via NIM cables to an external combinatorial logic as part of the trigger control system. The front-panel Reset input for resetting the RU logic are also on the front panel. At the rear side of the RU, the LAN and DAQ readout network cables can be routed as the crate’s upper-half rear is open.

1. IEEE 960 mechanics for double-width RUs corresponds to 13 modules per crate.

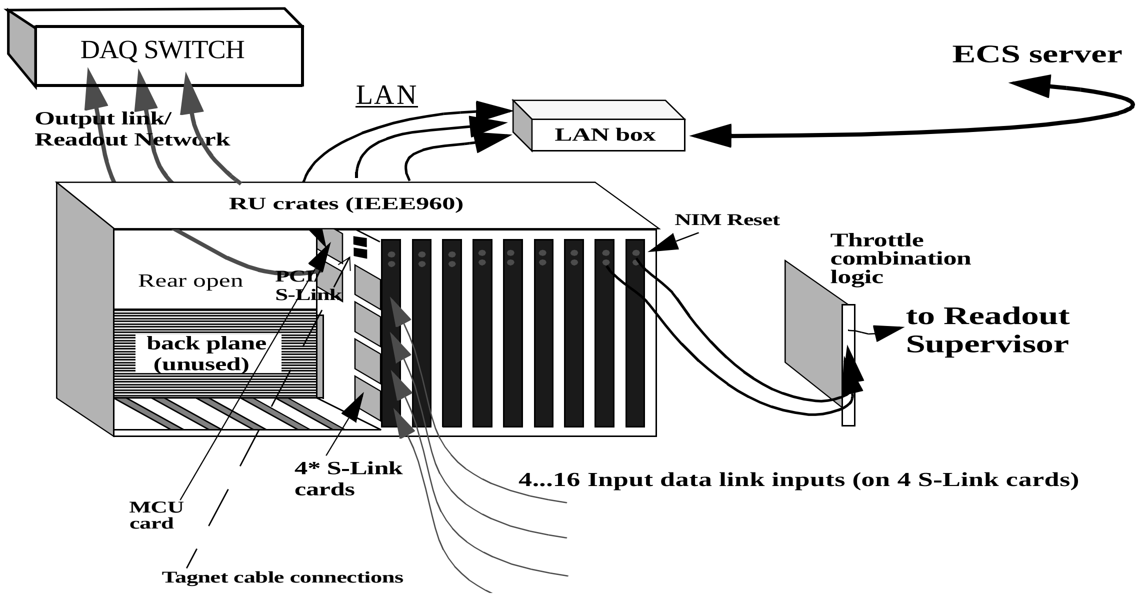


Fig. 5-24. DAQ RU connections in crate.

The power requirement per RU can be calculated as 7.5 W maximum per mezzanine card (according to the CMC specification) and less than 15 W for on-board logic, making a total of 60 W per board. This means 780 W per crate (13 boards), or equivalently 156 A from the 5 V supply. Using the F6853 CERN standard (Figure 5-25), an RU crate requires two 100-Amp 5-Volt supply modules and one 25-Amp 15 Volt modules.

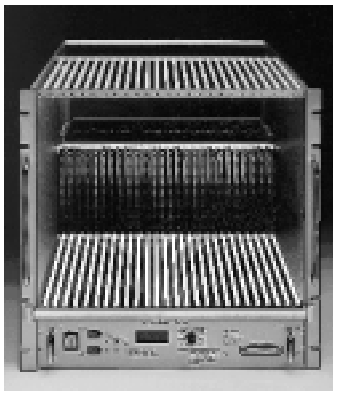


Table 5.1. RU crate power requirement

power modules	Maximum current	No of modules needed
+5 V	100 A	2
+15 V	25 A, 30 A	1

Fig. 5-25. Picture of the F6859 CERN crate. The upper rear half is free.

5.8. *The MCU for the RU-II*

The LHCb Readout Unit has been conceived as a 9U module in a Fastbus crate, using the crate for powering only without using the backplane bus. This option forces that in absence of the bus controller, an embedded controller is needed in each Readout Unit. The RU has some PCI components which need host initialization. Both needs led to the idea to design a local computer card in a PMC form factor, as the credit-card PC suggested by other LHCb members [S586pc] (which could be adapted to a PMC form factor) was not available to test the RU-II during year

2000 and early 2001. The aim was to develop in a short period of time a diskless computer to perform the following tasks for the Readout Unit boards:

1. **Initialization and configuration of the RU-II's PCI subsystem** (FPGAs, NIC and PCI bridges). This includes loading SCI drivers for the SCI NIC used in the Level-1 VELO trigger application and adding in-system programmability (ISP) capability to the hardware, allowing to program the FPGAs via the PCI bus.
2. **PCI bus arbitration**.
3. **RU-II monitoring and control** via (1) PCI-accessible registers in the FPGAs, and (2) general-purpose I/O (GPIO) signals that control some portions of the RU-II hardware.
4. **Interrupt handler** for special events signaled by the RU-II's FPGAs and NIC.
5. **I2C controller** for the programmable clock generator in the RU-II.
6. **JTAG controller** for test purposes in the RU-II.

The configuration, monitoring and control of a large quantity of MCUs is more efficiently managed and maintained from a central resource. The MCU was hence designed to be networked, to load the operating system and access a remote file system from a central server in the ECS LAN. This implies that the MCU must have a bootstrap code in non-volatile memory which allows to contact a server and obtain system files over a network link, thus avoiding the use of a hard disk for booting with the advantages of cost reduction, reliability and ease of maintenance.

5.8.1. Design choices

Three major choices determine the MCU design: (1) the form factor, (2) the remote boot mechanism and (3) the microprocessor. These three choices are justified below:

1. **Form factor**. Some of the embedded PC industry standard were not adequate for our application, company dependent, not fully specified or just quite expensive. A brief survey of standard and proprietary solutions for embedded and industrial PCs is listed bellow:
   - **PC/104 Consortium**[1]: Intended for a wide range of applications, defines 90-by-96 mm self-stacking modules communicating via 104- or 120-pin PCI bus connectors, without the need of a backplane and with a low-power consumption (1-2 Watts per module). Single-board computers with LAN interface exist, but lack of enough I/O functionalities and 66-MHz PCI operation as required for the RU-II application.
   - **PICMG (PCI Industrial Computer Manufacturers Group)**[2]: These computer systems use a passive PCI/ISA backplane and the CompactPCI Eurocard form factor. ISA bus route the interrupts, like in PC104 standard, so an special backplane is needed. This PCI form factor is not compatible with the RU applications requirements.
   - **SIM30 pin-size µCLinux computer project**[3]: The µCsimm module is a microcontroller module built specifically for the µClinux Operating System (a version of Linux for microcontrollers without a memory management unit). It stands an inch high, with a standard 30-pin SIMM form factor. I2C bus, 10Base-T Ethernet, serial port and 21 GPIO

1. See http://www.pc104.org/
2. See http://www.picmg.org/
3. See http://www.uclinux.org/index.html

pins are included. But the lacking of PCI interface, remote boot capability and the fact that is a single-company product prevent it from being considered an option.

- **Credit-Card PC (formerly Digital Logic AG, Switzerland)**[1]: Two weak points are the single-company production and the non-standard connector. Despite these drawbacks, it has found supporters in the LHCb Collaboration. The card was not available at the time of testing the RU-II.

It was decided to implement the MCU as a standard **PMC** card (74 mm x 149 mm, with 10 mm of stack-height) which is the *de facto* standard with VME in the HEP field. This standard does not define the position for the M66EN pin (required 66-MHz PCI operations) and does allow a PMC card to be the PCI system host. Nevertheless, VITA-32 extensions to PMC [VITA32std] assign a position to the M66EN pin and redefine some of the reserved pins in PMC to include the required signals to implement a PCI host on a PMC. This way, the required PCI functionalities can be implemented.

2. **Remote boot**.There are at least three solutions:
   - Use the Intel i82559 PCI LAN chip and Intel's Preboot Execution Environment (PXE)[2].
   - Use the Intel i82559ER chip (which is present in many of the last Intel Pro100 network cards) and use a PXE-like kit from Bootix Technologies GmbH[3].
   - Use any of these two chips and **Linux's Etherboot project** software, an open source initiative[4].

   The third option was chosen as it is free and well supported. In the Linux Etherboot scheme, the LAN chip uses a bootstrap code in EPROM to access[5] a DHCP (Dynamic Host Configuration Protocol) or BOOTP (Boot Protocol) server to get a host name and IP address for the MCU, the IP address for a TFTP (Trivial FTP) server and the Linux kernel image file name to download. With this information, the MCU accesses the TFTP server and gets the Linux kernel image and other files required in the boot process, like tools for building the downloaded image.

3. **Microprocessor**. The main criterion is Linux compatibility. Among a large number of possibilities, the ZFx86 chip from ZFLinux[6] was chosen because of its high degree of integration, low power consumption (as low as 0.5 W at 33 MHz) and embedded PCI interface with host capability. This chip integrates a Cyrix 486+ processor core running up to 120 MHz together with SDRAM, IDE (floppy), EIDE (hard disk), 16-bit ISA, USB, serial port, I2C, mouse and keyboard controllers. Additionally, the chip incorporates a 3.3-Volt 32-bit 33-MHz PCI interface that can act as PCI host. A PCI arbiter is also embedded in the chip.

   The chip has a typical PC architecture based on North and South bridges (Figure 5-26, left) with most of the I/O subsystem under the control of the South bridge, as well as the

1. See http://www.digitallogic.ch/english/products/datasheets/smartmodule_detail.asp?id=smartModule586PC
2. See http://developer.intel.com/ial/WfM/tools/pxesdk20linux/index.htm
3. See http://www.bootix.com/us/index.shtml
4. See http://etherboot.sourceforge.net/
5. Issuing a broadcast request, as the MCU does not know the BOOTP server address.
6. See http://www.zflinux.com

expansion PCI bus. The North bridge handles the SDRAM controller and the frontside PCI bus.

The PCI interface (Figure 5-26, right) is also typical in the sense that it does not include buffers for CPU-initiated PCI reads, yielding a poor performance. On the other hand, CPU-initiated PCI writes benefit from a 16 dual-word buffer. Nevertheless, monitoring and control throughput in the RU is the main purpose of the MCU and thus PCI performance will not be an issue.

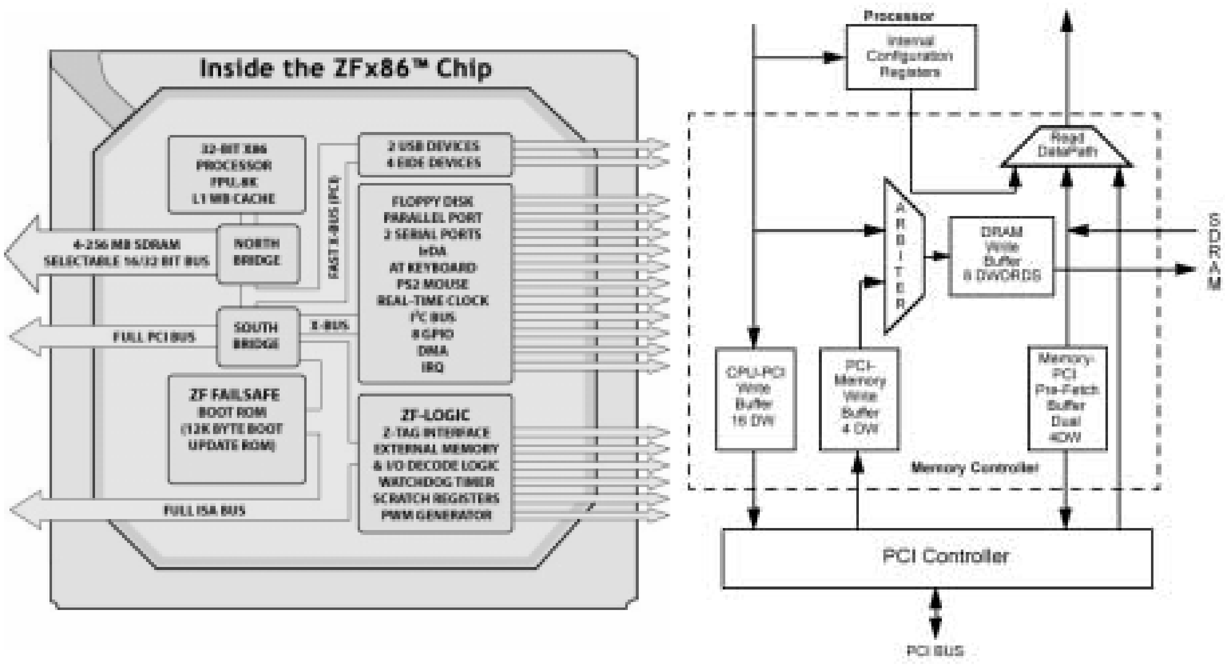

Inside the ZFx86™ Chip
32-BIT X86 PROCESSOR FPU, 8K L1 WB CACHE
4-256 MB SDRAM SELECTABLE 16/32 BIT BUS
NORTH BRIDGE
FAST X-BUS (PCI)
2 USB DEVICES 4 EIDE DEVICES
FLOPPY DISK PARALLEL PORT 2 SERIAL PORTS IrDA AT KEYBOARD PS2 MOUSE REAL-TIME CLOCK I²C BUS 8 GPIO DMA IRQ
FULL PCI BUS
SOUTH BRIDGE
X-BUS
ZF FAILSAFE BOOT ROM (12K BYTE BOOT UPDATE ROM)
ZF-LOGIC Z-TAG INTERFACE EXTERNAL MEMORY & I/O DECODE LOGIC WATCHDOG TIMER SCRATCH REGISTERS PWM GENERATOR
FULL ISA BUS
Processor
Internal Configuration Registers
Read DataPath
ARBITER
DRAM Write Buffer 8 DWORDS
SDRAM
CPU-PCI Write Buffer 16 DW
PCI-Memory Write Buffer 4 DW
Memory-PCI Pre-Fetch Buffer Dual 4DW
Memory Controller
PCI Controller
PCI BUS

Fig. 5-26. ZFx86 internal architecture and PCI interface.

5.8.2. MCU architecture

The main components in the MCU card (Figure 5-27) are:

1. **LAN subsystem**: consisting of an Intel i82559 PCI Ethernet chip, a transformer and a RJ45 connector and the Remote Boot PROM.
2. **SDRAM**: Four MT48LC8ML16A2 chips from Micron Technologies provide 64 MByte of memory. A small-form-factor SIMM would have been a cheaper option, though the difficulty to find horizontal SIMM connectors (to reduce component height) and the potential incompatibility with the ZFx86 chip resulted in a discrete implementation.
3. **Auxiliary connectors**: Serial port, keyboard, mouse, floppy and hard disk connectors are available in the MCU for development, debugging and diagnostics. These are not needed for the RU-II application.
4. **PCI connectors**: Defined in the PMC specification, for 32-bit PCI.
5. **Voltage regulator**: The 3.3 V supply available in the PMC connectors is regulated down to 2.5 V required by the ZFx86 and the LAN chip. A 1-Amp Micrel MIC39101 regulator in an 8-pin SSOP package is used.

6. **Programmable clock**: The ZFx86 chip needs external 33 MHz, 14 MHz and 48 kHz. These frequencies are generated by an on-board AMI FS6370, also used in the RU-II. The ZFx86's I2C interface can be used to program this chip, but taking into account that the FS6370 responds to all I2C addresses reserved for EPROM devices, a conflict may occur between the on-board FS6370 and any external EPROM I2C device. To solve the conflict, switches controlled by ZFx86's GPIO lines are provided (GPIO1 and GPIO2 in Figure 5-27).
7. **BIOS**: The MCU equips the old BIOS code from ZFLinux, and not the newer Phoenix BIOS release. The latter, requiring a larger memory size, would need an MCU redesign.

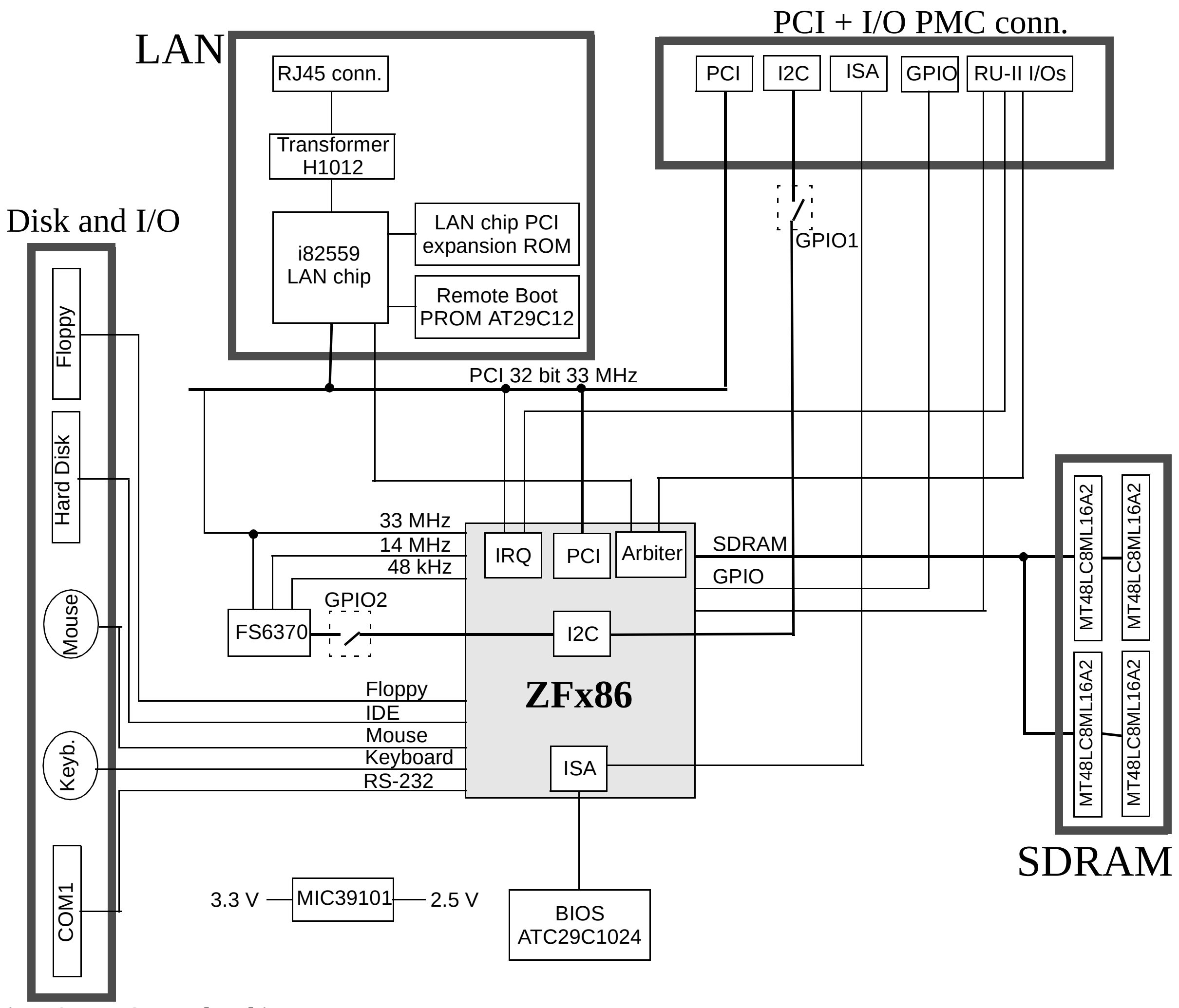
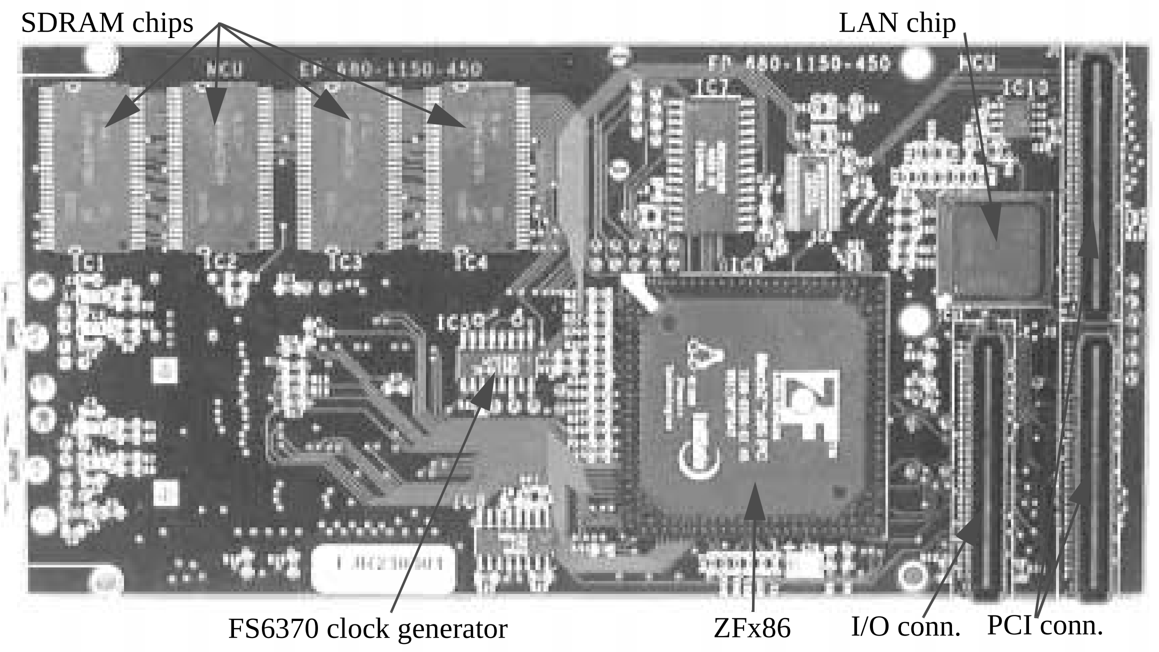


Fig. 5-27. MCU card architecture.

8. **PMC I/O connector**: This connector includes the I2C, JTAG (emulated via GPIO lines) and control signals to the RU-II. Seventeen signals in total that are described in Table 5.2. Additionally, 8-bit ISA and USB are also provided for other applications.

Figure 5-28 shows bottom and top views of the MCU card in actual size.



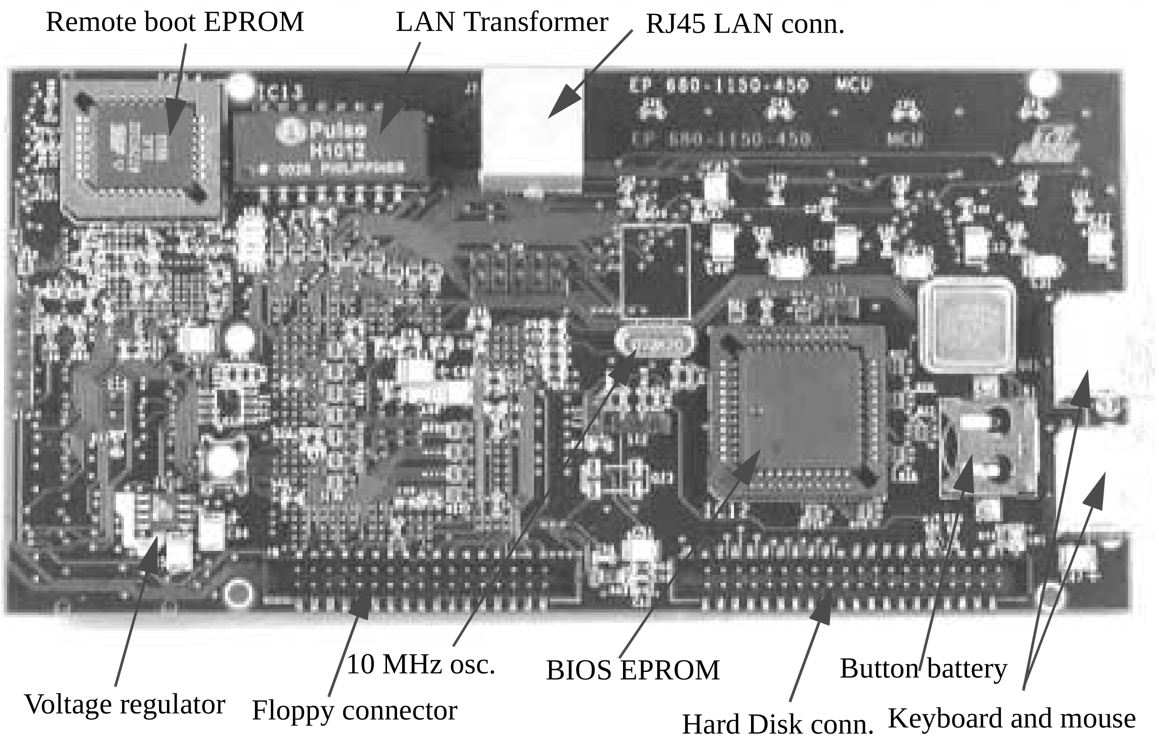


Fig. 5-28. MCU, bottom and top views (actual size).

Table 5.2. Non-standard MCU-RU interface signals

Name	Direction	Comment
REQ# (2:0)	Input	PCI arbitration lines for the PCI bridges and the auxiliary PCI connector
GNT# (2:0)	Output	
SCLK, SDA	Inout	I2C bus lines
TDI, TDO, TMS, TCK, TRST#	JTAG bus lines	
SYSRT#	Input	RU-II reset connected to the MCU’s NMI line
MCU_RT_REQ#	Output	Forces a reset in the RU-II board
SEL_CD, MODE	Output	FS6370 programmable clock signals.

CHAPTER 6 Readout Unit II laboratory tests

"We only know what we're told, and that's little enough. And for all we know it isn't even true"

Extracted from a play by Tom Stoppard.

6.1. Overview

The Readout Unit II must be tested with the twofold purpose of:

1. Verifying the module functionality, which is related to design errors.
2. Detecting unexpected behavior in the different components. This includes unknown features and unreported component bugs that escape system simulation.

The basic module functionality (several inputs combined into a single buffered output) is tested in section 6.2. An S-Link to S-Link data flow test is carried out in order to verify the sub-event merging, buffering and sub-event building functionalities (the latter only for the FEM application). No errors were found in the RU-II in these tests, demonstrating the correctness of the design.

The PCI-based sub-event building concept for the DAQ and VELO applications is tested in section 6.3, and the achievable performance for both scenarios is measured.

The PCI subsystem in the RU-II is also the most likely source of unexpected component behavior, as it includes the MCU, the FPGA's PCI interface and the Network Interface Controller, all of them of great complexity. Section 6.3 also describes the tests carried out on the PCI subsystem and the results, like more overhead in the PCI transactions than expected and an unreported error in the FPGA's PCI interface.

6.2. S-Link to S-Link dataflow test

The test setup includes FPGA-based sub-event transport format (STF) pattern generator cards for up to 1 MHz rate, designed in the context of the RU project (Figure 6-1). The frame length, data pattern (two alternating 32-bit words) and source identifier code are programmable via jumpers and switches. Connectors for trigger input and daisy-chain output are available in the front panel. Four of these cards are plugged onto the S-Link input connectors in the RU.

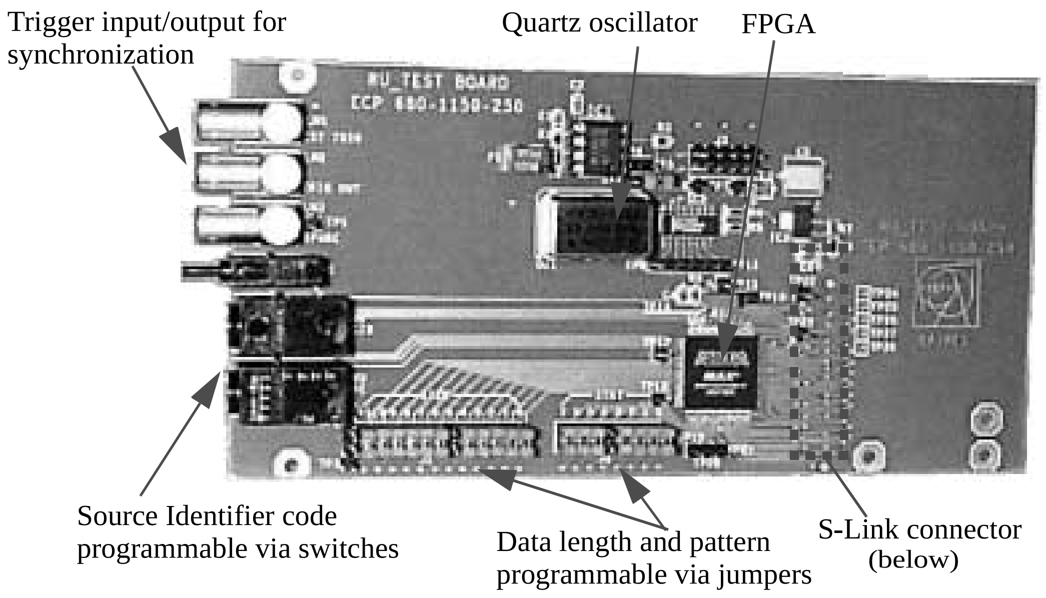

Trigger input/output for synchronization
Quartz oscillator
FPGA
RU_TEST BOARD
ECP 680-1150-230
Source Identifier code programmable via switches
Data length and pattern programmable via jumpers
S-Link connector (below)

Fig. 6-1. STF pattern generator card for RU-II test.

Figure 6-2 shows a frame generated by the STF pattern generator card. Firs word in the frame is an S-Link control word (LCTRLN, an active-low S-Link signal used to mark control words, is low) with value Ox00000010. This indicates that the event number is one and there are no link-reported errors (four least significant bits are all zeroes). Next two words identify the source (programmed via switches to Ox00F00000) and the data size (four 32-bit words, Ox00000004). After the four data words, the trailer (another S-Link control word) indicated that status flags have value OxEE and total frame length is eight (four data and four framing words).

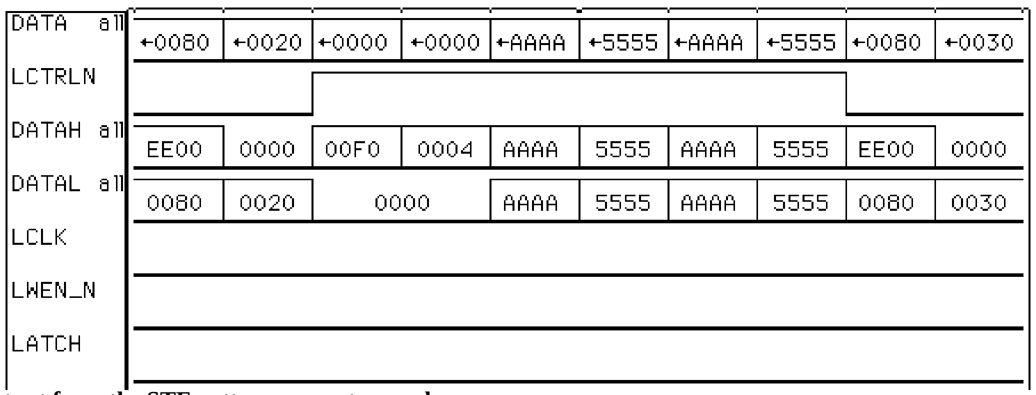


Fig. 6-2. Output from the STF pattern generator card.

Input event collection and sub-event building is performed in the RU-II, outputting to the S-Link output connector. A built sub-event captured at the RU-II S-Link output is shown in Figures 6-3 and 6-4. The sub-event header starts with an S-Link control word (LCTRLN low) and has the value Ox00000010, indicating event number one. Next two words in the header are the RU identifier code (programmed to OxFEEDBEEF) and the data size (Ox00000008). The eight data words follow with alternating pattern Ox55555555 OxAAAAAAAA.

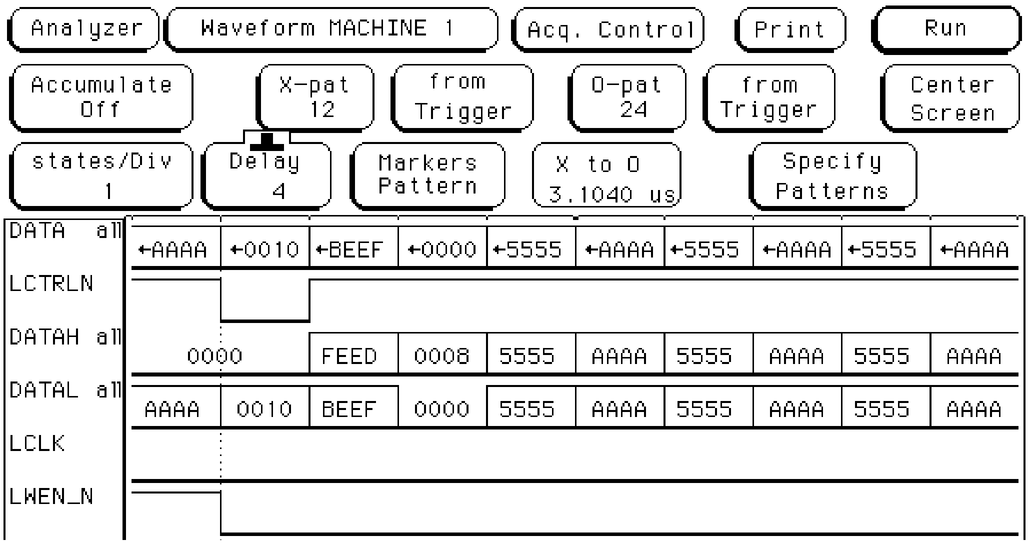


Fig. 6-3. Example of sub-event (part 1 of 2).

The frame ends with a trailer word (OxAB0000C0), indicating that the total frame length is OxC (eight data words plus four framing words). Next word is the header of sub-event number two, which also has eight data words.

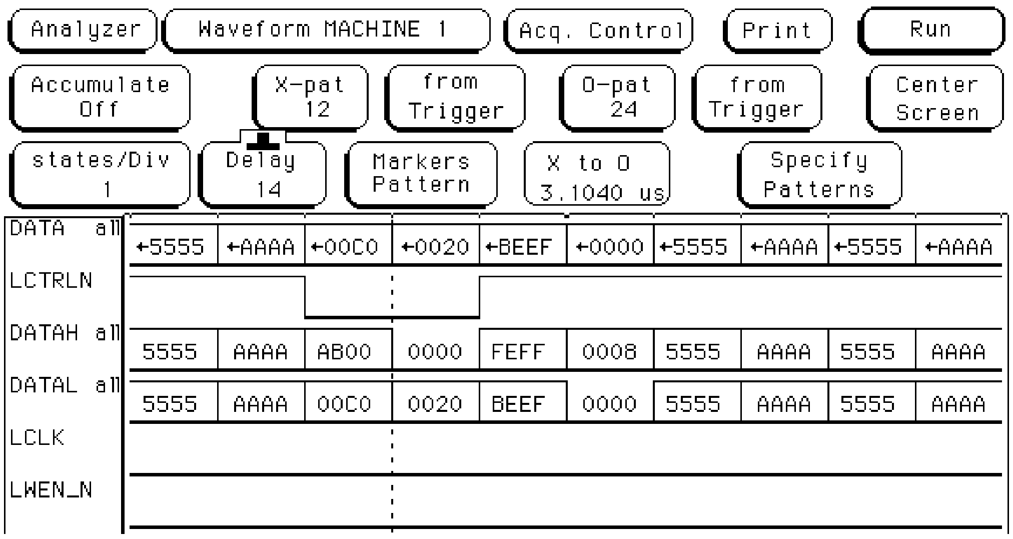


Fig. 6-4. Example of sub-event (part 2 of 2).

Figure 6-5 depicts a sub-event building measurement for the FEM application. Four 64-byte-payload frames at an equivalent 100-kHz Level-1 trigger rate are received per event and input link. Sub-event building is carried out on two input links, generating 512-byte sub-events at the S-Link output.

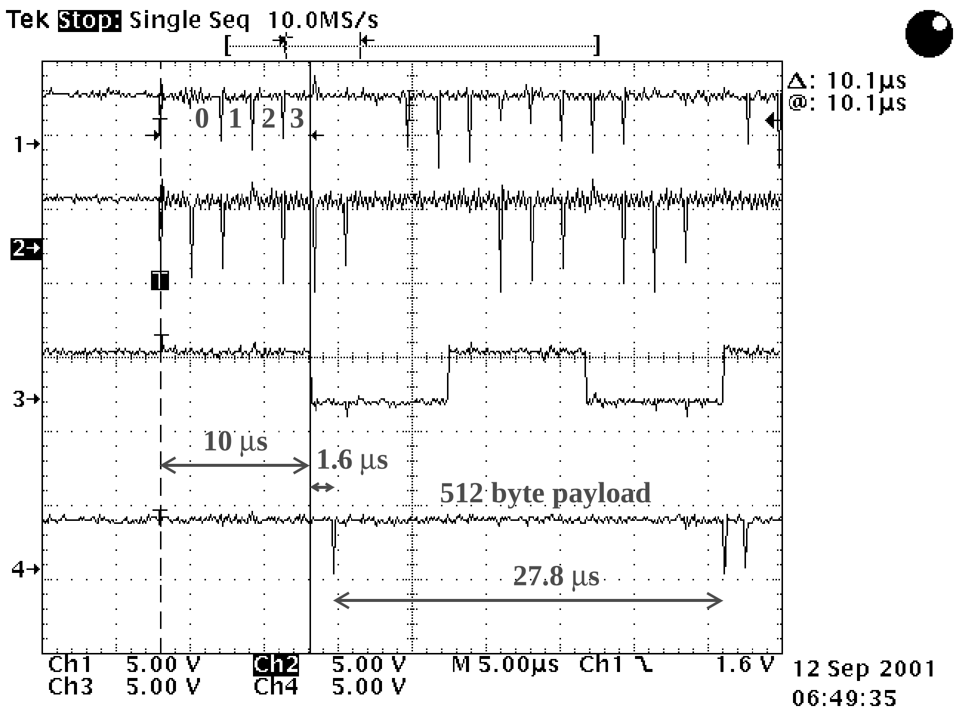


Fig. 6-5. Sub-event building measurement.

The FPGA clock in the RU-II is 15 MHz (25% of the nominal value). Trace 1 shows one of the incoming S-Link control inputs (LCTRL#), with four frames every 2.5 µs. Trace 2 is the corresponding input FIFO empty flag signal. Trace 3 shows the RU-II MAILBOX signal, toggling every time that a new 2-to-1 sub-event block has been merged and stored into the SEB. Trace 4 is the S-Link output LCTRL# line, showing a 1.6 µs output-stage latency.

Every 27.8 µs a 256-byte-payload sub-event is transmitted, corresponding to a 18.4 MByte/s throughput. A factor four of performance increase (74 MByte/s) can thus be achieved at a nominal 60 MHz FPGA operation.

6.3. PCI subsystem test

6.3.1. Access to the Sub-event buffer from PCI

The simple test utility for Linux, ***rwpci*** (described in “The rwpci utility” on page 169) can be used to provide access to the SEB from the MCU. This requires on the output stage FPGAs a target PCI interface and a dual-port memory controller programmed in VHDL. Figure 6-6 shows the data path between the MCU and one of the two sub-event buffers (SEB).

Using simple assignments like the shaded instructions in Figure I-III on page 168, the MCU accesses PCI memory space causing single-word (i.e. no burst) 32-bit PCI transactions on the root PCI bus.

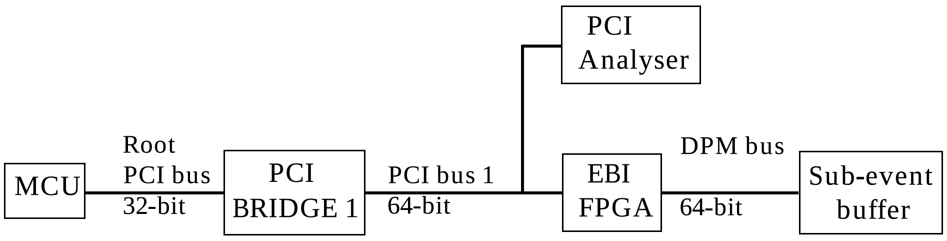


Fig. 6-6. Data path between the MCU and the Sub-event buffer.

The PCI bridge reacts initiating a transaction on the 64-bit wide PCI bus 1. Both the PCI bridge and the FPGA are 64-bit capable. Nevertheless, this does not mean that the bridge will therefore initiate a 64-bit transaction on PCI bus 1. In fact, only when the EBI FPGA declares the SEB as prefetchable memory, the transaction address is quadword[1] aligned, and at least two (memory read) or three (memory write) words are to be transferred, the bridge will perform a 64-bit transaction. Thus, single-word transactions, that are the ones generated by our test software, imply 32-bit bus accesses in PCI bus 1.

User logic inside the FPGAs cannot know if the transaction on the PCI bus 1 is 32 or 64-bit wide. This is meant to be transparent to both ends (user logic in the FPGA and C application running on the MCU) and thus have no implication in the success of the transaction.

The EBI FPGAs can only access their SEBs by means of 64-bit read and write pipelined transactions. The glue logic between the PCI core and the user logic in the FPGA can be configured to either *Quad port* or *Dual port* mode, being the PCI core in charge of the data width conversions between the current PCI transaction and glue logic data widths. These modes are depicted in Figure 6-7.

- *Quad port*: Master and target interfaces have separated 32-bit read and write paths.
- *Dual port*: Master and target share 64-bit read and write paths.

Using *Quad port* mode and non-prefetchable memory

As the SEB is declared as non-prefetchable for this test, the bridge performs single-word 32-bit transactions. This was verified with a PCI bus analyzer[2] plugged in PCI bus 1. The *rwpci* test utility combined with the PCI analyzer measurements show that read and write accesses to even memory positions are performed successfully, whilst read accesses to odd memory positions return the contents of the next memory position (i.e., read to address 5 returns data in address 6). Wrong data alignment in 32-bit to 64-bit conversion in the PCI core FIFOs was suspected and later confirmed by Lucent Technologies as a chip bug that is overcome using the Dual port mode instead of the Quad pot mode.

1. “Quadword” is 64 bits. “Dualword” is 32 bits. “Word” is used as a generic term.
2. Catalyst TA-700.

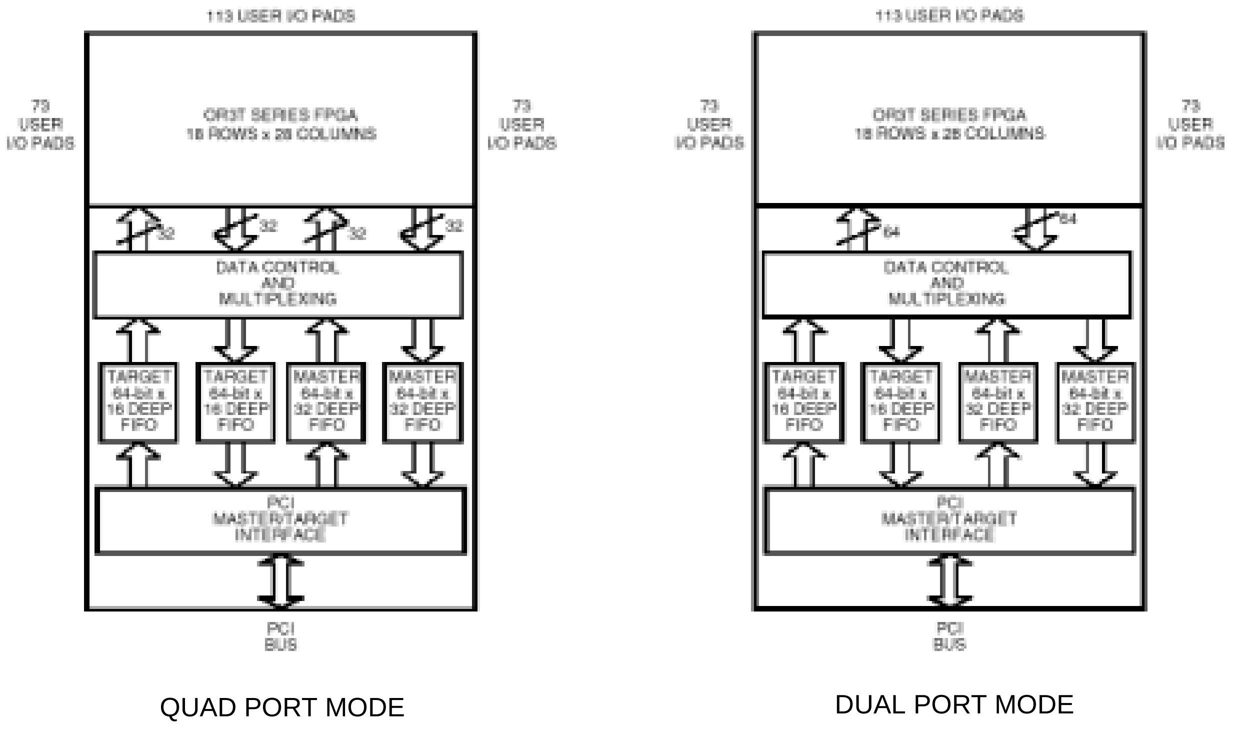


Fig. 6-7. ***Quad port* and *Dual port* modes for the PCI interface logic in the FPGA.**

Figure 6-8 shows in cycles five to eight a successful PCI write followed by a read at the first memory position in the SEB:

- Cycle 0: IDLE. No transaction taking place in the bus.
- Cycle 1: Configuration read to Ox80000010 (i.e., BAR0[1] which is in offset Ox10 in FPGA PCI configuration space starting at Ox80000000).
- Cycle 2: Read value is OxC0100000, i.e., the physical address for the start of the SEB. The transaction finishes with disconnect+data, which means that the PCI configuration space does not support burst mode transactions.
- Cycle 3: Configuration read to Ox80000000 (i.e., VendorID and DeviceID in offset Ox0 in FPGA PCI configuration space.
- Cycle 4: Read value: DeviceID is Ox5401 (registered code for OR3LP26B) and VendorID is Ox11C1 (reserved for Lucent Technologies devices).
- Cycle 5: Memory write to first position in SEB.
- Cycle 6: Written value is Ox0000049B.
- Cycle 7: Memory read to first position in SEB.
- Cycle 8: Read value is Ox0000049B.
- Cycle 9: Write to next memory position: OxC0100004.

1. BAR stands for Base Address Register. PCI devices have six of such registers (BAR0 to BAR5) in their configuration space. They are used to map memory areas connected to the PCI devices and thus make them accessible from PCI. In our application, the SEB is mapped in BAR0 and a number of general-purpose registers inside the FPGA are mapped in BAR5.

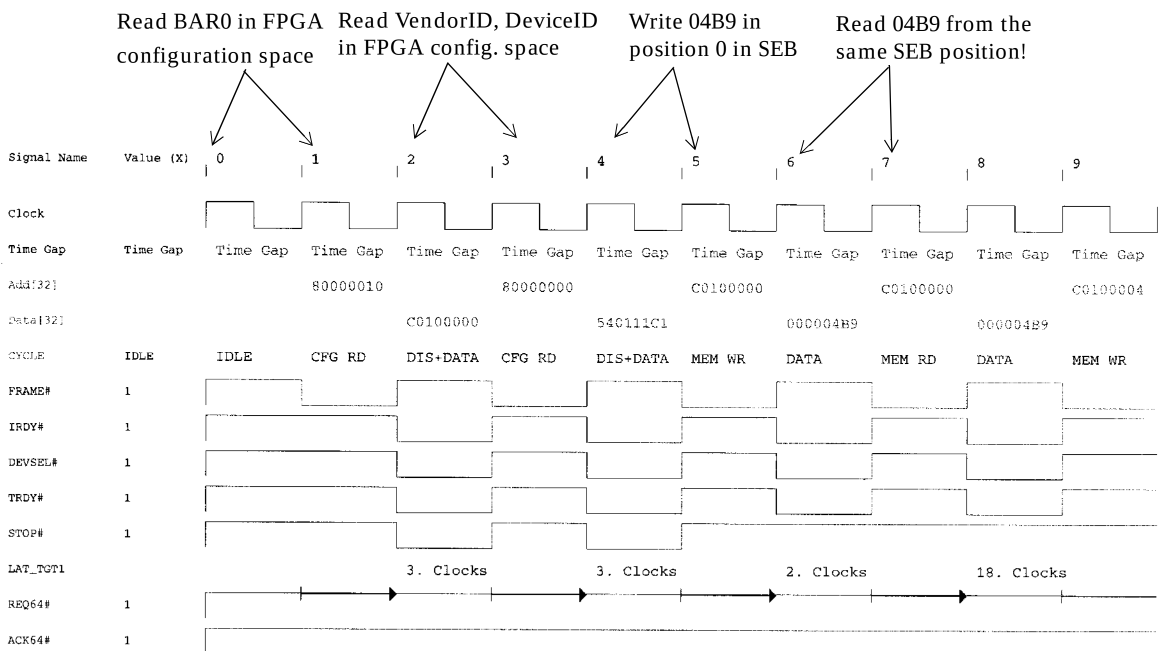


Fig. 6-8. PCI bus 1 activity captured with a PCI analyzer.

Using *Dual port* mode and prefetchable memory

A different PCI core configuration was generated with the FPGA vendor software and the VHDL code was modified for a *Dual-port* interface between user logic and embedded PCI core. The corresponding bitstream was loaded from the MCU into the EBI FPGA via the PCI bus (see section I.II.II). Figure 6-9 shows a snapshot of the MCU terminal in response to a **lspci -xvv** command. Line 7 shows that memory region 0 (where the SEB is mapped) is defined as prefetchable. This region is mapped starting at address OxC0200000.

```
01:0f.0 Signal processing controller: Lucent Microelectronics: Unknown device 5401 (rev 01)
Subsystem: Unknown device beef:dead
Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B-
Status: Cap+ 66Mhz+ UDF- FastB2B+ ParErr- DEVSEL=medium >TAbort- <TAbort- <MAbort- >SERR- <PERR
Latency: 255 min, 255 max, 64 set
Interrupt: pin A routed to IRQ 11
Region 0: Memory at c0200000 (32-bit, prefetchable)
Region 5: Memory at c0100000 (32-bit, non-prefetchable)
Capabilities: [50] #06 [0080]
00: c1 11 01 54 07 00 b0 02 01 00 80 11 00 40 00 00
10: 08 00 20 c0 00 00 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 10 c0 00 00 00 00 ef be ad de
30: 00 00 00 00 50 00 00 00 00 00 00 00 0b 01 ff ff
```

Fig. 6-9. MCU terminal dump.

Measurements show correct behavior for 64-bit transactions, but some word-swapping when transactions are 32-bit wide. Once again, wrong multiplexing in the internal PCI core FIFOs in the FPGA is suspected, which can be related to the quad-port mode bug reported by the FPGA vendor. Figure 6-10 shows a single-word read operation to DPM initiated by the MCU. As the DPM is mapped in prefetchable space, the bridge initiates a burst transactions of variable length (determined by the bridge instantaneous buffer space availability).

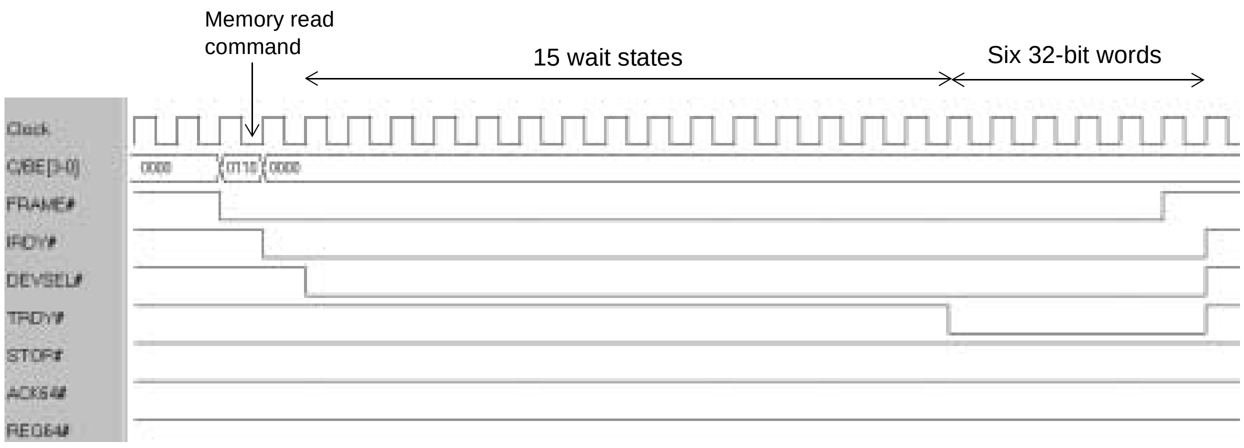


Fig. 6-10. Single-word DPM read from the MCU.

Note the fifteen wait cycles inserted by the FPGA after DEVSEL# assertion. This latency represents a low overhead for the DAQ application (4 KByte sub-events) but would be too high for the VELO application.

6.3.2. Control and monitoring registers in the FPGAs

For the tests described in this section, five 32-bit registers have been mapped in the FPGA’s BAR 5 space (see Table 6.1). These user-defined registers are the foundation to implement in the FPGAs configurable parameters, error counters and monitoring registers which can be accessed from the MCU via PCI (see data path in Figure 6-6). Figure 6-11 shows the result of the command ***rwpci ebi2a WREG 100 100*** (100h is written to the 256th 32-bit word -offset 400h- followed by a read operation for verification).

Table 6.1. User-defined register mapping

Register	**Offset (bytes)**	**Offset (32-bit words)**
Reg 1	100h	40h
Reg 2	200h	80h
Reg 3	300h	C0h
Reg 4	400h	100h
Reg 5	500h	140h

A target retry is generated as a first response to the read, and 12 wait state cycles are inserted by the FPGA after the retry (one cycle for DEVSEL# assertion and 11 cycles due to the user logic and the embedded PCI core delays. User logic delay is in the order of one or two cycles and cannot be further reduced).

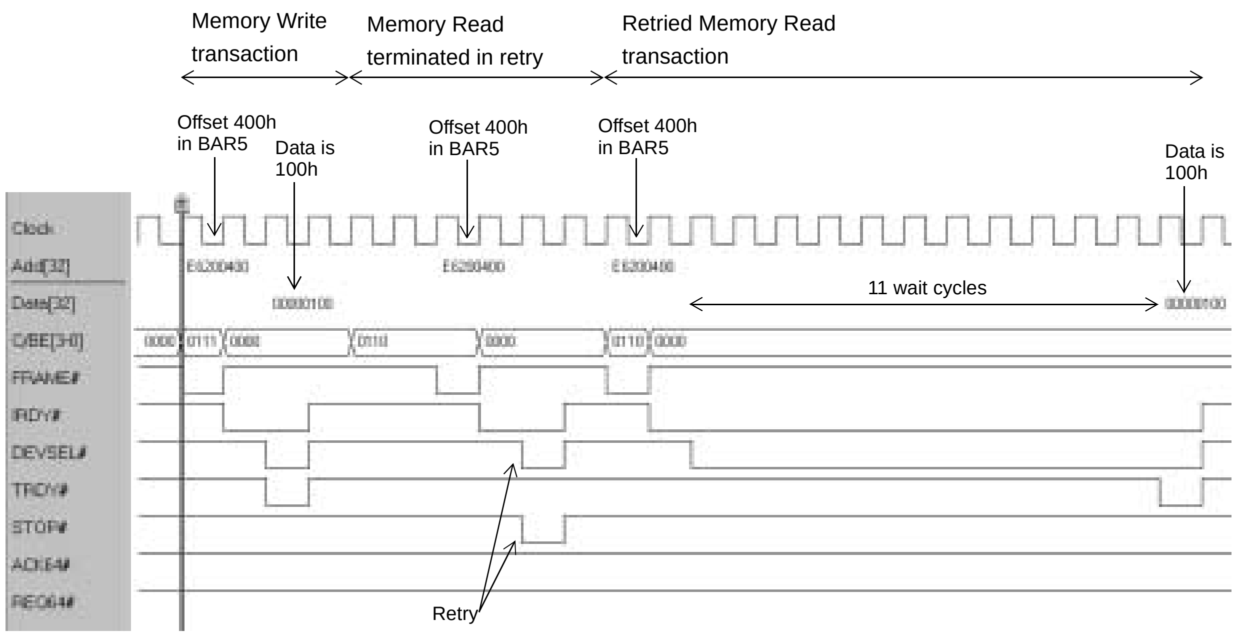


Fig. 6-11. Access to a user defined register in the FPGAs: write followed by a read.

6.3.3. Performance measurement tests

PCI performance has been measured on three different scenarios:

1. FPGA-to-FPGA write test for single-FPGA output stage architecture in the Level-1 VELO application.
2. Tandem-FPGA architecture for the Level-1 VELO application.
3. DAQ application emulation: TA-700 bus analyzer (as PCI master) reads the sub-event buffer via EBI FPGAs.

1. FPGA-to-FPGA write test for single-FPGA output stage in the VELO application

This test shows the maximum performance achievable with a single FPGA in the output stage. A *Dual-Port*-mode PCI master write interface is implemented in VHDL in one of the two output stage FPGAs (EBI2 in Figure 6-12), together with a set of registers[1] accessible via PCI. The MCU sets the burst length (in 64-bit words), destination address (start of SEB in the other EBI FPGA) and go/stop bit through these registers. When enabled via the Go/Stop bit, EBI2 FPGA will continuously produce write transactions to EBI1 FPGA, which is programmed with a target PCI interface and SEB access logic.

This test can be carried out loading the corresponding bitstream file in the FPGAs and running the line command ***rwpci ebi2a TESTWREBI1 length 1*** (where ***length*** is the desired number of 64-bit words). This command will configure the registers REG2 to REG4 in EBI2 and initiate the test.

1. See Section 6.3.2 in page 138.

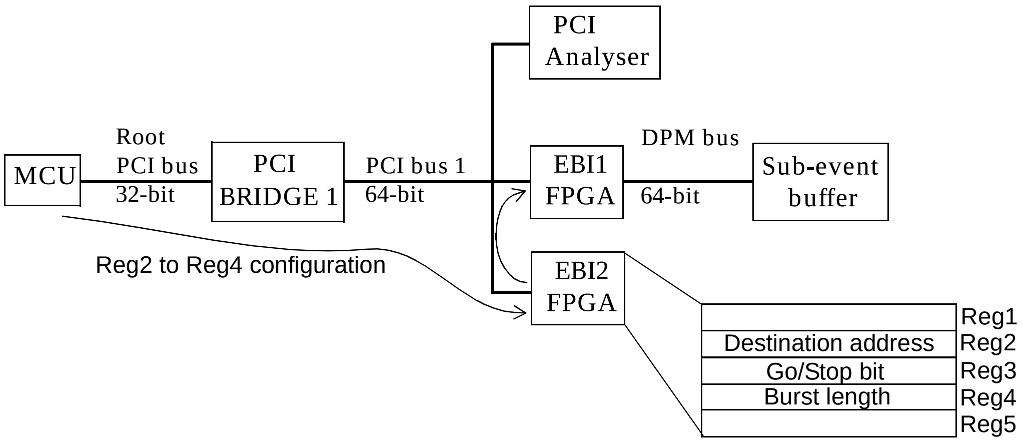


Fig. 6-12. FPGA-to-FPGA write test setup.

Figure 6-13 shows a 64-bit-wide 80-byte transfer between the two FPGAs, where a single wait state is inserted by the target at the beginning of each transaction (thus, the FPGAs are fast-decoding devices according to the PCI specification). Figure 6-14 shows that two consecutive transactions are spaced by ten clock cycles, and this is true regardless of the burst length.

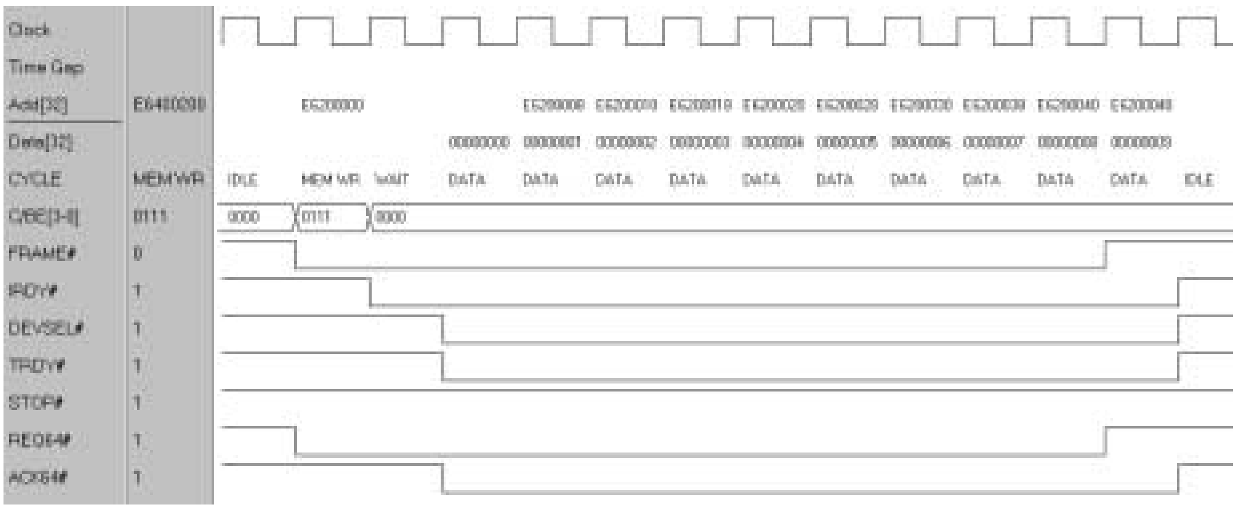


Fig. 6-13. Eight-word burst transaction between FPGAs in the output stage.

Where do these ten empty cycles come from?. The first one is a turnaround cycle (the current master releasing the bus after a transaction completion). The other nine are inserted by the FPGA embedded PCI core and cannot be avoided. If we take into account that the most performance-demanding application (Level-1 VELO) will require in average 32-quadword bursts, the maximum achievable performance in the PCI bus would be (counting 10 idle cycles, one wait state and one address cycle, see Figure 6-15) 32/44, i.e., 72.7% of the available bus bandwidth. This would make 384 MByte/s at 66 MHz clock speed or 192 MByte/s at 33 MHz for 64-bit transactions.

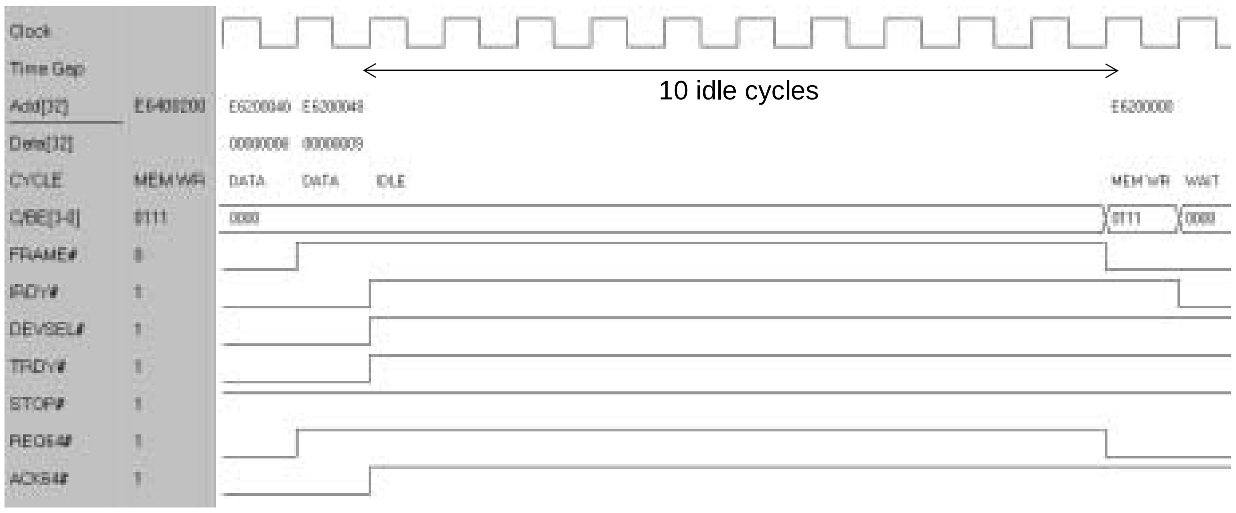


Fig. 6-14. Ten empty clock cycles between transactions.

For a real FPGA-to-NIC write scenario, seven target wait cycles in the NIC must be considered instead of just one (find the justification in the next paragraph). This decreases the PCI bus efficiency to a 64% (338 MByte/s at 66 MHz).

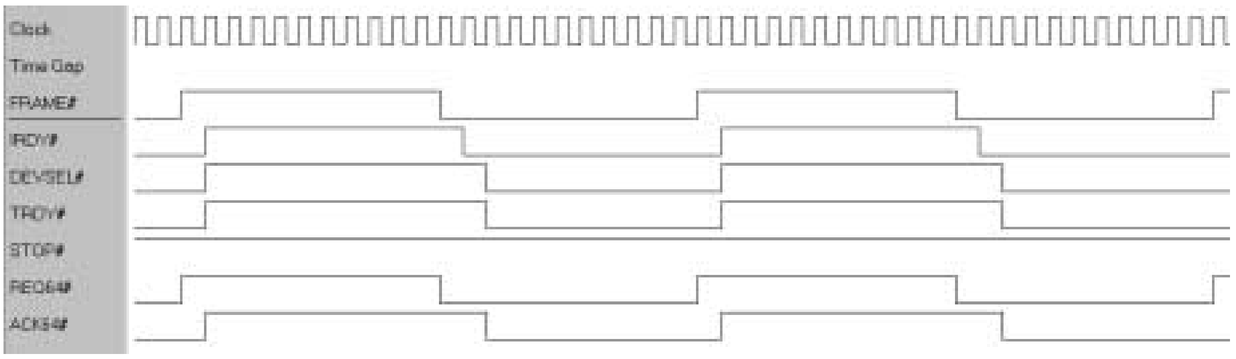


Fig. 6-15. PCI bus efficiency in the DAQ scenario.

2. Tandem-FPGA scenario for the VELO application

The tandem-FPGA operation bus efficiency is shown in Figure 6-16, where a Dolphin[1] PCI-to-SCI adapter with 33-MHz 64-bit PCI interface has been used as NIC. This card introduces four wait states as target write. Modern NIC cards should respond faster and thus increase the bus efficiency (for instance, the FPGA's PCI interface respond in just one clock cycle). Surprisingly, new 66-MHz-capable Dolphin PCI-to-SCI cards introduce seven wait states, spoiling performance for small-fragment applications like the Level-1 VELO.

1. Dolphin Interconnect LLC. See http://www.dolphinics.com/

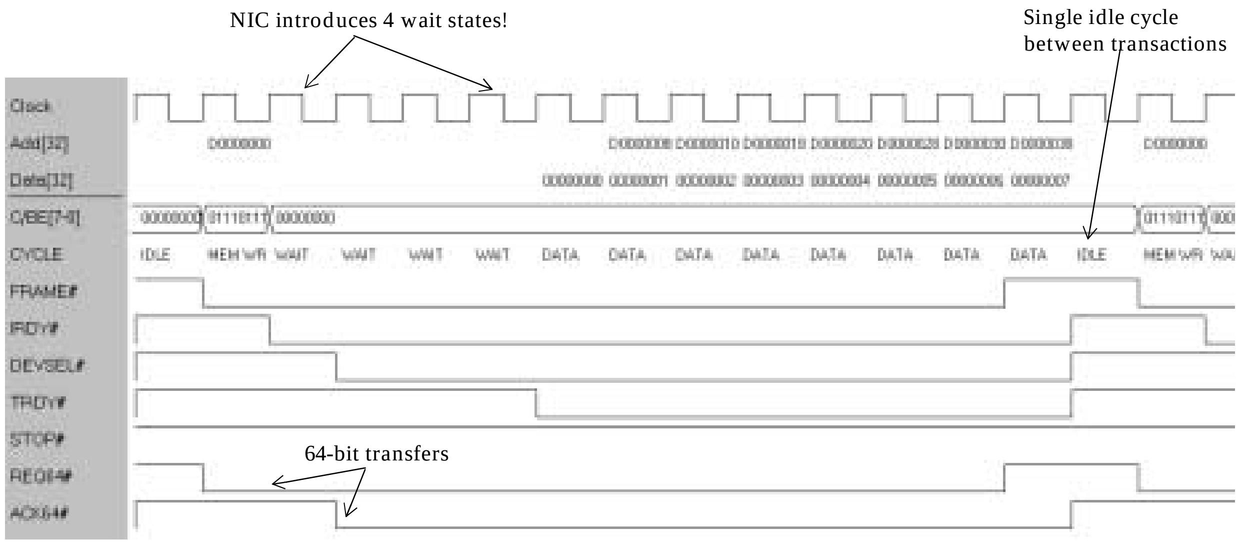


Fig. 6-16. 64-byte burst write to a Dolphin SCI NIC (LC4000 and L5B9350 chips).

According to the tests, a 16-quadword (128 byte) transfer takes 22 cycles (16 data, 1 address, 4 wait and 1 turnaround), resulting in 16/22 bus bandwidth efficiency (72.7%). This yields 193 MByte/s at 33 MHz. For a seven-cycle NIC latency, the efficiency drops to a 64%, matching the single-FPGA architecture performance (338 MByte/s). Figure 6-17 shows bus activity for tandem operation.

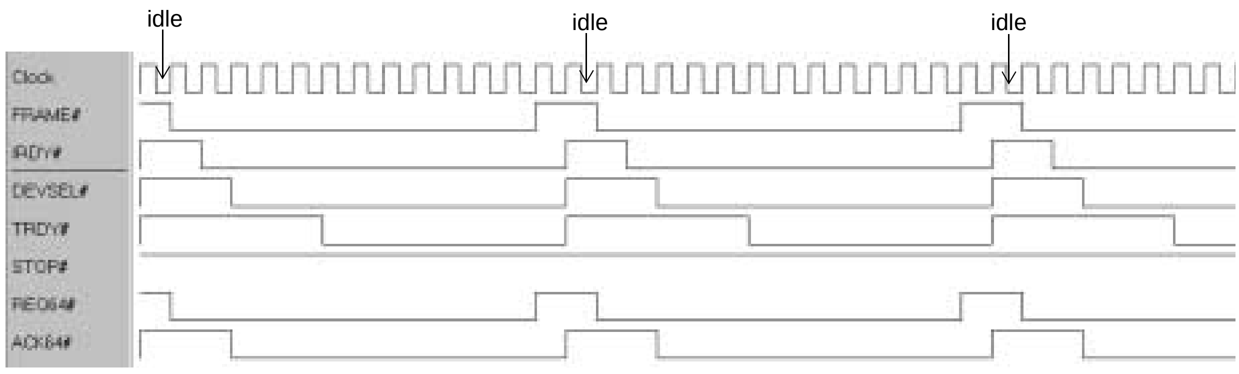


Fig. 6-17. A single idle cycle between transactions can be achieved.

3. DAQ application emulation: TA-700 read EBI FPGAs

The PCI bus performance for the DAQ application has been measured using the TA-700 bus analyzer as a PCI master reading alternatively the two EBI FPGAs. As it can be seen in Figure 6-18, the FPGAs add 15 wait states before returning data, the same result as in measurement in Figure 6-10 (the eleven cycles shown in figure Figure 6-11 and four additional cycles due to the DPM first-word latency and user logic state machines). The TA-700 adds a turnaround cycle plus eight empty cycles between transactions and a real NIC may require a similar number of cycles (ten in the case of a FPGA, see Figure 6-14).

Long burst transactions (2 KByte) will take place in the DAQ application, reducing the impact of these 15 wait cycles. The maximum bus usage for a 2-KByte sub-event fragment, taking into account the protocol overhead (1 address cycle, 1+15 wait cycles, 8 empty cycles and 1 turnaround cycle) is 95.9% for 32-bit transactions and 81.8% for 64-bit transactions. This makes 127 Mbyte/s and 217.5 Mbyte/s respectively at a 33 MHz operation.

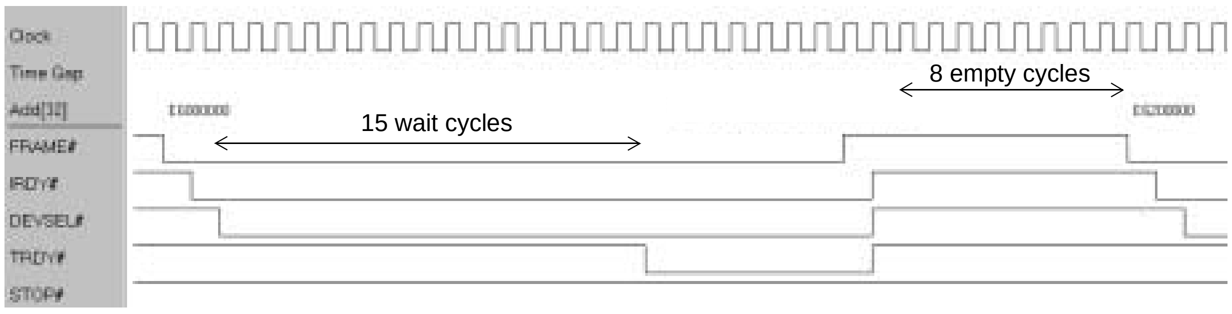


Fig. 6-18. FPGA read initiated by the TA-700 bus analyzer.

6.3.4. Conclusions

The PCI subsystem has been successfully tested at 33-MHz clock frequency (using an Advantech single-board-computer card) and 18 MHz (using our MCU card). Tests at higher bus frequencies would require a 66-MHz PCI capable MCU, which was not available[1].

The tested features include:

1. FPGA bitstream configuration from the MCU (with and without flash memories on the RU).
2. MCU access to the user-defined registers via PCI.
3. MCU access to the SEB via PCI.
4. PCI write operations between FPGAs to emulate single-FPGA output stage for the VELO application.
5. DAQ application emulation (requiring a target read/write PCI interface).
6. Tandem-FPGA VELO application emulation (requiring additionally a master write PCI interface).

The results of the first three points in the above list are exportable to the input-stage PCI subsystem. FPGA's PCI interface weak and strong points (like the single wait cycle as target write, the fifteen wait cycles as target read or the ten empty cycles between transactions as master write) have been identified and their impact on the VELO and DAQ applications studied. It has been also observed that under certain circumstances, in both the *Quad port* and the *Dual port* modes, the core swaps erroneously the lower and upper dualword in the cores' 64-bit FIFOs (recently reported by Lucent Technologies as a product design bug).

PCI bus bandwidth performance for single- and double-FPGA VELO operation has been measured, showing identical performance when using the new PCI-to-SCI adapters, as analyzed in "Tandem-FPGA output stage" on page 69.

1. Several commercially available 64-bit 66-MHz VITA-32 compliant PMC processor cards exist which could be used to carry out 66-MHz tests.

CHAPTER 7

Conclusions

"Arrakis teaches the attitude of the knife--chopping off what's incomplete and saying: Now, it's compete because it's ended here."

From the novel "Dune" by Frank Herbert.

7.1. New trends in DAQ systems for HEP experiments

Large HEP experiments cannot rely anymore in crate-based backplane bus standards like VME to cope with the high throughputs involved in their DAQ and trigger systems. Migration towards a "data processing across the on-board bus" paradigm (opposed to "data processing across the backplane") using PCI as the on-board bus, provides high performance (up to 528 MByte/s raw bandwidth per bus segment) and the benefits of an industry-supported standard solution. I/O is routed via point-to-point links in the front- and back panel, and the backplane is only used for power. The Readout Unit has been designed following this trend.

Alternatively, PCI backplanes are also an attractive solution for DAQ if compared to VME-based systems. A typical PC provides five to six PCI slots, resulting in a low-cost platform that includes mechanical support, cooling, power supply and an CPU for data processing and/or monitoring and control. A low-cost LAN card adds connection to a slow control system. Certainly, the 3.3 V regulation in today's PCs may not be adequate for some applications (10% regulation, when some 3.3 V chips require 5% stability) and the cooling may be insufficient. The number of PCI slots is low, however the bandwidth per card is below 30 MByte/s for 32-bit 33 MHz systems. Moreover, the PCI form factor is small compared to 6U and 9U VME boards, posing additional restrictions.

Using commodity PCs as DAQ platforms, flexibility and design reuse can be achieved by plugging PMC and other mezzanines onto PCI DAQ cards. This results in a new trend, the Flexible I/O concept, in which a general-purpose mezzanine-carrier PCI card with SDRAM and FPGAs can be used in a large number of applications by plugging-in the appropriate mezzanine

(ADCs, protocol conversion, legacy bus interfaces, I/O and data preprocessing, etc.) and writing the corresponding code for the carrier card’s FPGA.

These trends were discussed in the first chapter. The PCI-FLIC card designed as part of this thesis for the RU test station (section 7.3.3) was also introduced in the first chapter and its applications are described in the appendices. This card is the first 64-bit-architecture implementation of the Flexible I/O concept and is being used in the LHCb and NA-60 experiments at CERN.

The immediate future is in PCI-X. High throughput systems are possible in currently existing dual-bus 64-bit 66-MHz PCI motherboards, in which the available throughput per module is multiplied by eight. PCI-X based implementations will further enhance the available throughput (up to 64-bit bus at 133 MHz, i.e., 1 GByte/s) and solve some of the mentioned electrical and mechanical drawbacks. A proposal for a Readout Unit implementation in a PCI-X card is depicted in “A Readout Unit on a PCI-X card” on page 151.

Performance will leap upwards in the second half of this decade with the advent of the Third-Generation I/O (3GIO) [Bhatt01], a natural evolution of PCI based on point-to-point differential lines which also targets the PC market. It will have an enumeration and software device model compatible with PCI, providing a smooth technology transition in which PCI, PCI-X and 3GIO will coexist in motherboards with combined PCI-3GIO slot connectors. Allowing chip-to-chip and module-to-module low-pin-count interconnection at up to 100 MByte/s per pin, it is well suited for the coming year’s bandwidth requirements in HEP DAQ applications.

The results are presented in two publications:

1. “**A flexible PCI card concept for Data Acquisition: the PCI-FLIC**”. H. Müller, F. Bal, A. David, D. Dominguez, J. Toledo, in Proc. 12th IEEE Nuclear and Plasma Science Society Real Time Conference, Valencia, Jun. 2001, pp. 249-253.

2. “**A plug&play approach to Data Acquisition**”. H. Müller, J. Toledo, F. Bal, J. Buytaert, A. David, D. Domínguez, M. Floris, A. Guirao, in Proc. 2001 IEEE Nuclear Science Symposium, San Diego, Nov. 2001, to be published.

Additionally, a manuscript with title “A plug&play approach to Data Acquisition” has been submitted for publication in the IEEE Transactions on Nuclear Science as is pending of acceptance.

7.2. Contribution to the LHCb DAQ and Trigger systems

The thesis objectives, as stated in the preface of this thesis are:

1. Investigate the feasibility of the Readout Unit for the LHCb experiment and define its architecture and algorithms in the context of the LHCb DAQ and trigger systems.
2. Design and implement the Readout Unit.

3. Validate the design through laboratory tests and measurements.
4. Participate in the definition of the LHCb DAQ link technologies, data formats and protocols.

The achievement of the first three objectives is discussed in sections 7.2.1, 7.2.2 and 7.2.3, whilst the last one is commented in section 7.2.4.

7.2.1. Readout Unit modules

Though initially conceived for a single application (entry stage to the DAQ system), two other applications in LHCb apart from the DAQ RU have been investigated: the FEM module and the RU for the Level-1 VELO Trigger. The feasibility to build a single module which targets these three applications has been demonstrated, leading to the design of two different RU implementations.

The needed functionalities and design constraints for the three target applications have been studied. Different architectures are analyzed, demonstrating their feasibility in terms of throughput and the weak dependence of the performance with the specific implementation (determined by the number of processing elements in the input and output stages). The feasibility study is completed by input and sub-event buffer size estimation, resulting in very relaxed needs for input FIFOs (a few kilobytes) but more demanding requirements for the SEB (2 MByte minimum). This implies, due to high cost and low density of integration of true dual-port memory, that an efficient memory usage is mandatory.

Sub-event building algorithms for each application are proposed. In the FEM application, a 16-bit output to an S-Link transmitter card provides the required performance with a safety factor greater than two. PCI-based sub-event building between the RU output stage and the NIC aims at higher throughput applications like DAQ and Level-1 VELO trigger. In the former case, a “pull” protocol ruled by an intelligent NIC results in unnecessary overhead which can be tolerated as performance is not affected due to the large frame sizes involved. The benefits of intelligent NICs are in their capability to perform traffic shaping and complex readout protocols with the readout network. Gigabit Ethernet is the currently preferred technology for the DAQ (unbeatable for its widespread and low cost), though at least a 2-Gbit/s Gigabit Ethernet NIC is needed to achieve the nominal 160 MByte/s requirement.

The nominal requirement for the VELO application is building 200-240 byte sub-events at a 1 MHz trigger rate, resulting in 200-240 MByte/s throughput. This application cannot afford the overhead implied in a “pull” protocol due to the small sub-event size (around one quarter of a kilobyte, sixteen times less than in the DAQ application). A “push” protocol is used in which the RU output stage writes sub-events into the NIC buffer via PCI transactions. The PCI bus protocols reduce the bus usage to a poor 64% in a single-processing element scheme. A tandem-FPGA output stage architecture is proposed, which can potentially yield a higher performance. Nevertheless, the sub-event size, FPGA and NIC latencies (measured in laboratory tests) combine to annul this superiority. The advantage of the tandem-FPGA scheme can be found then in a reduced output-stage algorithm overhead provoked by the architecture parallelism. Both schemes (single-element and tandem output stages) yield a maximum 338 MByte/s for 64-bit 66-MHz PCI implementations, 30-to-40% above the nominal throughput.

Two RU modules have been implemented on 9U-sized cards, with S-Link CMC slots for data input on the front panel. The upper half of the rear panel includes a PMC slot for the MCU (interface to the ECS) and a shared PMC/S-Link slot for data output. The backplane is not used but as a power connector, being all I/O routed via point-to-point connections. This results in an implementation of the “data processing across the on-board bus” paradigm above mentioned, using PCI bus in the output stage as interface technology between the RU output stage and a commercial high-throughput network adapter.

The implementation of on-board PCI buses and system clocks up to 66 MHz posed the need to carry out pre- and post-layout signal integrity studies and simulations, field in which we pioneered at CERN.

A first Readout Unit prototype was built in May 2000, as described in chapter four. It implements a true 64-bit architecture, consisting on an FPGA-based input stage for data merging, a dual-port memory for sub-event buffering and an FPGA-based output stage for sub-event building and readout network protocols. Targeting the FEM and DAQ applications, the nominal requirement is building 4 KByte sub-events at a nominal 40 kHz Level-1 trigger rate (160 MByte/s throughput). At a 50 MHz FPGA operation, the prototype architecture allows 400 MByte/s raw bandwidth, exceeding the nominal requirements.

The experience with the first RU prototype resulted in a list of improvements and simplifications for a re-design of a final RU module. A module design revision was also needed to reduce the module cost, increase the sub-event buffer size and to include the features required by the VELO application, resulting in a redesign during summer 2000: the Readout Unit II.

The resulting RU-II module was considerably less expensive due to reduction of expensive components (like DPM memory, FIFO memory, Flash memory and non-standard mezzanine cards). Also the diversification of components was further reduced, in particular a single, new type of FPGA was used with the benefit of a higher gate count and faster PCI interface. The use of PCI as configuration bus for all FPGAs made the company-specific interface to a MCU card redundant and by using the PMC standard also for the MCU, the special connectors could be avoided. As a consequence of the simplifications, the new 9U module could be routed in eight PCB layers only, compared with the twelve of its predecessor. Extensive analog signal integrity checks were performed, as a result of which the addition of a PCI bus bridge for the delicate 66 MHz PCI bus became mandatory.

The addition of one extra FPGA in the input stage allowed to test a new, parallel architecture in which two parallel 32-bit data flows merge at the output stage in a tandem-FPGA sub-event building operation. Thus, its capability to cope with high trigger rates relies on the use of parallel data paths rather than on wide buses.

The results are presented in three publications:

1. “**Readout Unit for the LHCb experiment**”. J. Toledo, H. Müller, F. Bal, B. Jost, in Proc. Fifth Workshop on Electronics for LHC experiments, Snowmass, Colorado, Sept. 1999, pp. 352-356.

2. “**A Readout Unit for high data rate applications**”, J. Toledo, F. Bal, D. Dominguez, A. Guirao, H. Müller, in Proc. 12th IEEE Nuclear and Plasma Science Society Real Time Conference, Valencia, Jun. 2001, pp. 230-234.

3. “**A Readout Unit for high data rate applications**”, J. Toledo, F. Bal, D. Dominguez, A. Guirao, H. Müller. Accepted for publication at the IEEE Transactions on Nuclear Science.

7.2.2. Monitoring and Control Unit (MCU) for the Readout Unit

A single-board computer on a PMC form factor has been designed to serve as MCU for the RU. This design follows the VITA-32 extensions for Processor PMCs, which allow the implementation of a PCI host on a mezzanine card. The card includes a 10/100 Mbit Ethernet interface with remote booting capability, 32 MByte of SDRAM, standard interfaces for development (floppy disk, hard drive, keyboard and mouse) and other bus controllers via the PMC I/O connector (I2C, ISA, USB and JTAG). Additional I/O lines, specific for the RU application, complete the MCU interface.

The MCU is PC on a PMC card based on the ZF-x86 chip from ZF-Linux. Windows95, DOS and Linux 2.2.x have been successfully tested on the MCU, though Linux is the chosen development platform. The results are presented in the following publication:

1. “**A networked mezzanine card Linux processor**”. A. Guirao, J. Toledo, D. Domínguez, B. Bruder, H. Müller, in Proc. 12th IEEE Nuclear and Plasma Science Society Real Time Conference, Valencia, Jun. 2001, pp. 81-84.

7.2.3. Readout Unit II laboratory tests

S-Link to S-Link data flow tests have been successfully carried out, validating the input stage and sub-event buffer concept and design. Sub-event building towards the S-Link output results in a FEM application implementation in which 74 MByte/s are reached for a 60 MHz FPGA operation (almost twice the nominal throughput).

The PCI subsystem has been successfully tested at 33-MHz clock frequency (using an Advantech single-board-computer card) and 18 MHz (using our MCU card). FPGA’s PCI interface weak and strong points (like the single wait cycle as target write, the fifteen wait cycles as target read or the ten empty cycles between transactions as master write) have been identified and their impact on the VELO and DAQ applications studied. It has been also observed that under certain circumstances, the PCI core swaps erroneously the lower and upper halves of the data words. This result was recently reported by Lucent Technologies as a product design bug. Another surprise was the high target latency in the tested PCI-SCI adaptor cards.

The tested features include monitoring and control (FPGAs bitstream configuration from the MCU, MCU access to the user-defined registers and to the SEB) and PCI-based sub-event-building emulation for DAQ and VELO applications. In the former, a PCI bus analyzer reads alternatively from the two output stage FPGAs. In the latter, the two FPGAs write alternatively to a PCI-SCI adaptor card.

PCI bus efficiency for single- and double-FPGA Level-1 trigger operation has been measured (64% bus efficiency or 338 MByte/s), showing figures below the expected performance, though still above the nominal requirement of 240 MByte/s. Higher bus efficiency (72.7%) is shown in the DAQ application emulation due to the larger nominal sub-event size, reaching 193 MByte/s on a 64-bit 33-MHz PCI bus, exceeding in a 20% the nominal requirements for the DAQ application (160 MByte/s). The safety margin can be increased with 66-MHz PCI NIC cards.

All these tests and measurements validate the design and show that the Readout Unit II meets the requirements of all three applications.

7.2.4. Data formats and link technologies

Data formats and protocols in the LHCb DAQ system were not defined at the time the Readout Unit project started. As a part of the work carried out, a Sub-event Transport Format has been proposed in order to interface the Level-1 electronics with the FEM stage and to handle event data through the different DAQ system stages.

The sub-event building protocols between the DAQ RU and the Network Interface Controller have been defined in collaboration with other members of the LHCb DAQ group.

A solution for several module interfaces (FEE-FEM, FEM-DAQ RUs, FEE-VELO RUs) based on the CERN S-Link convention has been proposed, allowing to evaluate different link technologies during the coming years and easing maintenance and support.

PCI, an industry standard, is proposed as common denominator between the RU output stage and a network interface card (VELO and DAQ applications). No other standard can compete nowadays with PCI in widespread and bandwidth.

Performance for both I/O technologies (S-Link and PCI) is estimated and measured, demonstrating the correctness of the proposed solutions.

7.3. Current and future work

A number of HEP experiments and private companies have shown their interest on the PCI-FLIC card, opening a potential line of work in the study and implementation of the different applications.

Further work on programming the Readout Unit for the different applications, specially for the Level-1 Trigger application -together with the corresponding system integration- is being carried out as the subject of a new doctoral thesis at CERN EP/ED group.

The future of the Readout Unit is in one of the two following lines:

7.3.1. An alternative approach based on Network processors

The advent of a new generation of ICs for the high-end switch market, the Network Processors (NPs), provide an alternative approach to the Readout Unit. Network Processors aim at implementing packet processing and forwarding and other switch-related tasks on a programmable IC. This allows to update the switch to new protocols and requirements, superseding the lack of reconfigurability of ASIC-based switches. NPs embed a set of processors, memory and hardware accelerator units, making use of parallel distributed computing and pipelining techniques to achieve hardware speed.

An implementation proposal for the Readout Unit based on the IBM NP4GS3 Network Processor [NP4GS3] is briefly described in [Dufey01]. This proposal was presented to the collaboration in year 2001 by members of the LHCb DAQ group in clear competence with the implementation defended in this thesis. The competition led to an internal review, whose conclusions [Gavillet01] highlighted the technical feasibility and similar costs of the two options. Preliminary performance results of the NP-based implementation are promising for the DAQ application, but not for the Level-1 VELO application (see Figure 7-1).

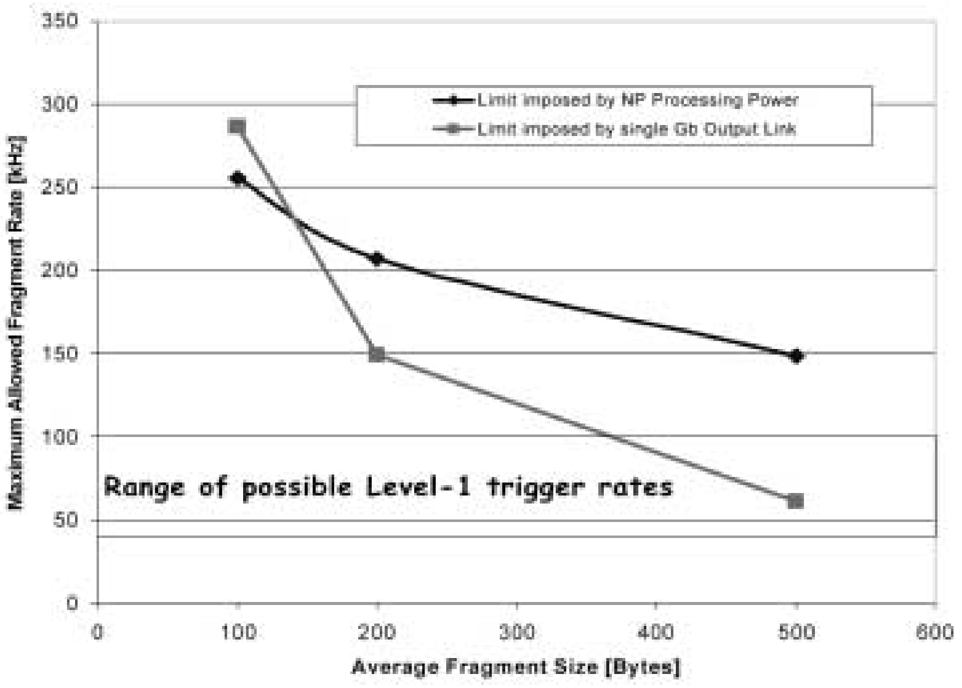


Fig. 7-1. Preliminary performance results for the Network Processor proposal.

Further work on the NP-based solution must be carried out in order to develop a prototype and demonstrate its feasibility.

7.3.2. A Readout Unit on a PCI-X card

Shrinking the Readout Unit, a 9U-sized card, on a PCI form factor does not look like and easy task. A closer look to the RU-II boards reveals that in basically consists of four FPGAs, four DPM chips, four FIFO chips, two PCI bridges, one MCU, one NIC and four data input CMCs, apart from auxiliary electronics like voltage regulators and clock buffers.

Applying the Flexible I/O concept to the Readout Unit, it results evident that the NIC can reside in a separate PCI slot, the MCU is not needed as the PC's CPU can implement its functionalities, a single FPGA in a larger package could replace the actual four FPGAs, which would also make unnecessary the PCI bridges. Moreover, the input CMCs and the four FIFOs could be shrunk into a single mezzanine plugged on the RU carrier board.

A floor plan for such a PCI Readout Unit is depicted in Figure 7-2. Board layout and component sizes are set to scale. The following components could be used:

- An Altera APEX-II FPGA in a 1.25mm-pitch 724-pin BGA package, with 492 to 536 available I/O pins. With 1.9 and 5.2 million equivalent gates and PCI-X support up to 133 MHz, the four ORCA FPGAs on the RU-II could be replaced by a single APEX-II in a 35-by-35 mm BGA package.
- IDT 36-bit synchronous FIFOs in a 128-pin TQFP (20 mm by 14 mm), same ones as in the RU-II.
- IDT 36-bit synchronous DPMs in a BGA package (17 mm by 17 mm), same ones as in the RU-II but the package.
- LVDS serializer and deserializer and two RJ-45 connectors for the TagNet implementation, same components as in the RU-II.
- NIM connectors for reset input and throttle output, same as in the RU-II.
- One CMC mezzanine card with four connectors for data, control and power.

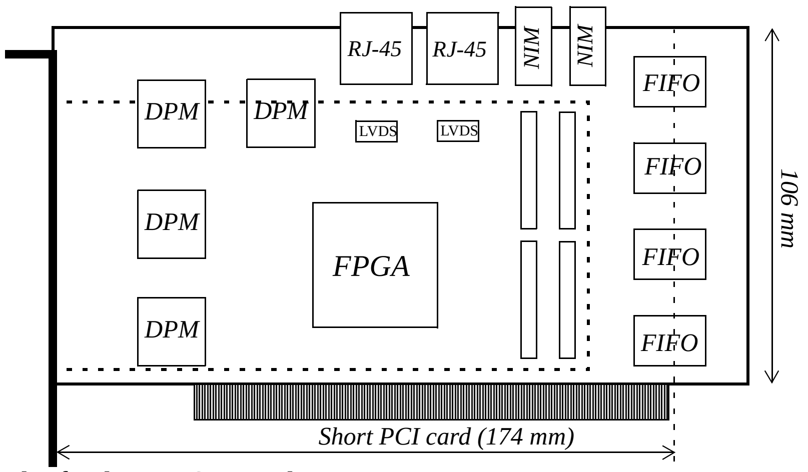


Fig. 7-2. Draft floor plan for the RU PCI-X card.

Single-processing-element input and output stages would provide the required performance over a 100-MHz PCI-X bus segment. PCI-X chipsets for high-end servers will be available soon, like the VIA APOLLO PX-266. This chipset includes a four-way PCI-X bridge which will allow architectures like shown in Figure 7-3.

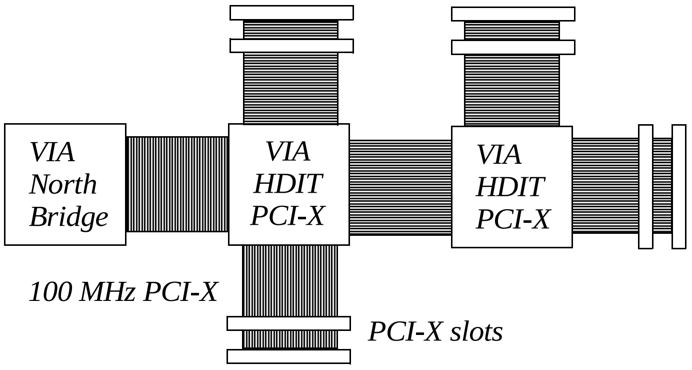


Fig. 7-3. A possible PCI-X motherboard architecture.

Two 100-MHz PCI-X slots per bus segment can accommodate one RU PCI-X and one NIC card, resulting in several RUs per PC (four in the figure) sharing a common MCU (i.e., the host CPU). The advantages of this solution are reduced cost and higher performance.

7.3.3. A test station for the Readout Unit

A test station must be built with the three-fold purpose of serving as (1) RU test and demonstrator station, (2) qualification test station for production RUs and (3) RU development station. A PC-based test station (Figure 7-4) has been designed, consisting of five main components:

1. The **Readout Unit** module to be tested.
2. A **FastBus crate** to provide power and mechanical support for the RU (not shown in the figure).
3. A six-PCI-slot commodity **PC** running the test software.
4. Ten **S-Link cards** (five on the RU and five on the PC) that can be configured as either receiver or transmitter. Such cards have been designed at EP-ED group [Bal01] using inexpensive physical media (LVDS over UTP[1] with RJ-45 connectors) and are currently in the production phase.
5. **PCI-FLIC cards** that interface the S-Link cards to the PCI bus on the PC (see “The PCI Flexible I/O Card (PCI-FLIC)” on page 22).

The implementation has not been completed though. All the test software, with the exception of the PCI-FLIC linux driver, has to be written. No further hardware development is needed, as the FLIC cards are already existing and the S-Link rx/tx cards are in the production phase. The FPGA in the PCI-FLIC need to be programmed, though most of what is required can be re-used from the existing NA-60 PCI-FLIC developments (SDRAM controller and PCI target interface). The FPGA code in the RU does not need to be modified.

1. Acronym for Unshielded Twisted Pair.

If a NIC is plugged on the RU output slot and a second NIC replaces the S-Link receiver card on the PC (either directly or as a mezzanine on top of the PCI-FLIC, depending on the NIC card form factor), the station can be used to test also the RU DAQ and L-1 VELO configurations. Nevertheless, an S-Link to S-Link configuration is intended as the baseline test setup.

Optionally (and conveniently) a connection between a LAN card on the PC and the MCU would ease FPGA reconfiguration and monitoring using the utilities described in “MCU development” on page 168. The PC can then emulate not only the data sources and destination, but also the slow-control system.

The software for the baseline RU test station will have the modular structure depicted in Figure 7-5:

- **Control Program**: Synchronizes the operation of the other software modules, sends data to S-Link transmitters and reads the buffer in the FLIC card that hosts the S-Link receiver card. Two synchronization signals between the Control Program and the RU are implemented, as described in the next section.
- **Pattern Generator**: Under the control of the Control Program, will accept data from a file, create data patterns according to the sub-event transport format and store them in a buffer in RAM.
- **Compare Utility**: Under the control of the Control Program, compares the generated data patterns and the received sub-events from the RU to generate and log error statistics.
- **FLIC driver for Linux**: This is an existing device driver from the NA-60 experiment [David01-2].

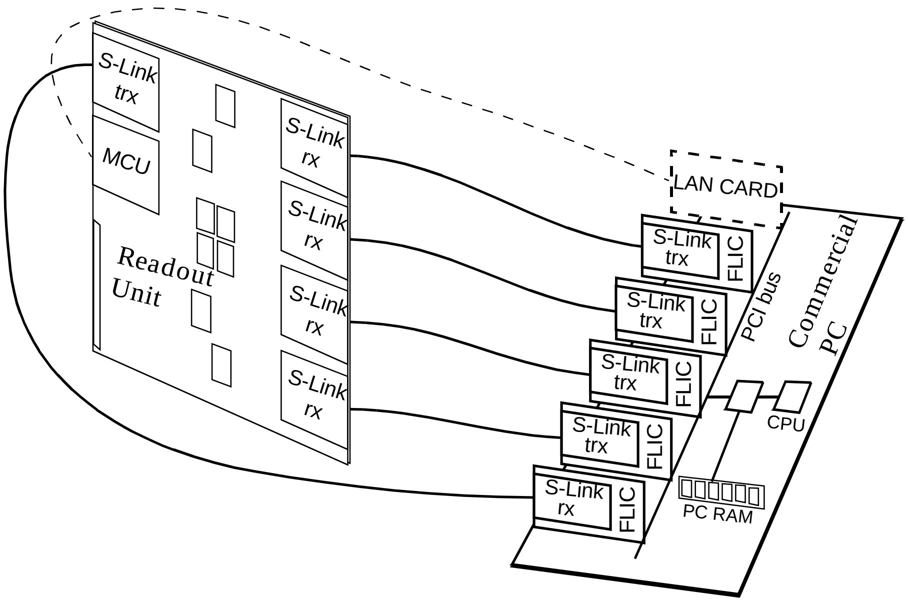


Fig. 7-4. Test station architecture with S-Link to S-Link configuration.

For the baseline test system, the following algorithm will be used:

1. If needed, the Control Program (CP) asks the Pattern Generation utility (PG) to write new patterns in the RAM buffers.

2. Once the patterns are ready, the CP will send them to the RU via the S-Link transmitter cards. The RU input stage will assemble and store incoming event fragments into the SEB. The RU output stage remains inactive until the CP enables the sub-event building operation (via a Xon/Xoff signal[1]).
3. When the patterns have been sent (note that the total size must not exceed the SEB capacity to avoid buffer overflow) the CP enables the RU output stage and sub-events are sent from the RU to the FLIC that hosts the S-Link receiver card on the PC. The sub-events are stored in the FLIC's SDRAM.
4. When the SEB empties, the RU signals this event to the CP via a dedicated signal[2] (accessible from the CP as a PCI register in a FLIC card). The CP reads the FLIC SDRAM and stores sub-events into the main PC RAM.
5. The CP asks the Compare Utility to analyze the sub-events and write results into a log file.
6. The CP returns to step 1 if further test is required.

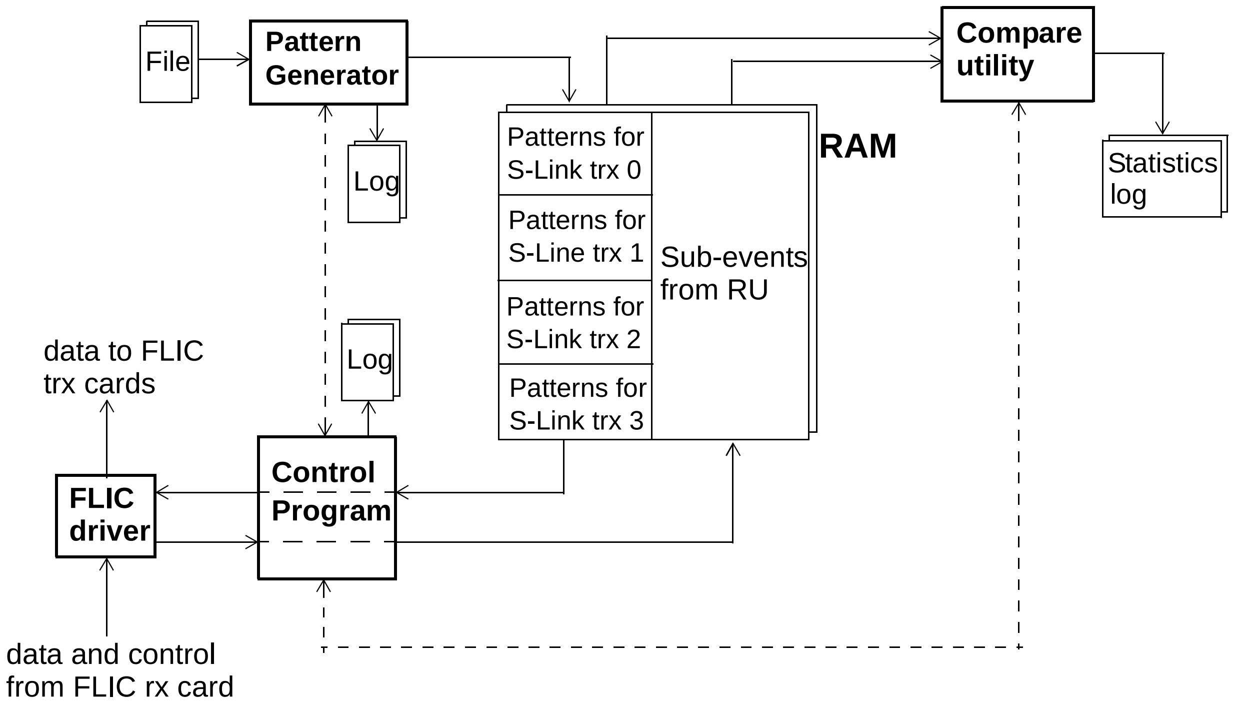

File
Pattern Generator
Log
Compare utility
Patterns for S-Link trx 0
Patterns for S-Line trx 1
Patterns for S-Link trx 2
Patterns for S-Link trx 3
Sub-events from RU
RAM
Statistics log
data to FLIC trx cards
Log
FLIC driver
Control Program
data and control from FLIC rx card

Fig. 7-5. Test station software block diagram.

1. Among several possible implementations, a register in the RU FPGA's PCI space looks like an attractive option.
2. The RU throttle output can be used for this purpose if connected to one of the generic I/Os in the PCI-FLIC card.

References

[Adam92] “Design and performance of the DELPHI Data Acquisition System”. W. Adam et al. IEEE TND, 1992. pp 166-175.

[Antchev97] “32MB PMC Dual Port Synchronous DRAM Module”. G. Antchev, T. Anguelov, I. Vankov, S. Cittolin, A. Fucci and D. Gigi, in Proc. XVII International Symposium on Nuclear Electronics. Varna, Sept. 1997.

[Baird00] “A PMC based ADC card for CMS tracker readout”, S. A. Baird, J. A. Coughland, R. Halsall, J. Hartley, W. J. Haynes and T. Parthipan, IEEE Trans. Nucl. Sci., vol. 47, no. 2, pp. 158-161, Apr. 2000.

[Bal01] “Readout Unit I/O S-Link cards”. Technical LHCb note in preparation. F. Bal et al. CERN EP-ED group.

[Barsotti90] “A progress report of the Switch-based Data Acquisition System prototype project and the application of switches from industry to high-energy physics event building”, E. Barsotti, A. Booth, M. Bowden and C. Swoboda, in Proc. Symposium on Detector Research and Development for the Superconducting Super Collider”, Fox Worth, Texas, Oct. 1990.

[Bhatt01] “Creating a Third-Generation I/O Interconnect”, Ajay V. Bhatt, Technology and Research Labs, Intel Corp. Available: http://developer.intel.com/technology/3GIO/downloads/3rdGenWhitePaper.pdf

[Bij97] “S-Link, a Data Link Interface Specification for the LHC Era”, E. van der Bij, R. McLaren, O. Boyle and G. Rubin, IEEE Trans. Nucl. Sci., vol. 44, no. 3, pp. 398-401, Jun. 1997.

[Bock01] “The Active Rob Complex: An SMP-PC and FPGA based solution for the ATLAS Readout System”, R. Bock et. al., in Proc. 12th IEEE NPSS Real Time Conference, Valencia, 2001, pp. 199-203.

[Brosch98] “MicroEnable, a Reconfigurable FPGA Coprocessor”, O. Brosch et al., in Proc. 4th Workshop on Electronics for LHC Experiments, 1998, pp. 402-406.

[Bruder00] “MCU and Linux Programming for the Readout Unit”, B.Bruder Nov, 2000. Available: http://hmuller.home.cern.ch/hmuller/RU/MCU/jtag.pdf and http://b.home.cern.ch/b/bruderbe/www/

[CAMACstd] “CAMAC, A Modular Instrumentation System for Data Handling”. EUR 4100, Office for Official Publication of the European Communities, Case Postal 1003, Luxemburg.

[Cattaneo97] The ALEPH Handbook. Volume 2. Chapter 10 (The Data Acquisition and Control System). M. Cattaneo, M. Frank, J. Harvey, B. Jost, P. Mato and W. von Rüden, CERN, 1997. Available: http://alephwww.cern.ch/HANDBOOK/VOL2/CH10/

[Charles99] “The Silicon Vertex Tracker”, SLAC Detector Physics Seminar Series. Eric Charles, Nov, 1999. Available: http://hepunx.rl.ac.uk/BFROOT/www/doc/Seminars/detector/SvtDetTalk/sld001.htm

[Charpak74] “Drift chambers”. G. Charpak. CERN, 1974.

[Charpak76] “Wire chambers: a review and forecast”. Comments Nucl. Part. Phys. 6 (1976), pp.157-171.

[CMCstd] “Draft Standard for a Common Mezzanine Card Family: CMC”, IEEE P1396 Draft 2.0, April 1995.

[Costi&Toledo99] “Evaluación de la tecnología de bus PCI para el desarrollo de tarjetas de adquisición de datos de altas prestaciones”, Final Studies Project of P. Costi Kowolik, Universidad Politécnica de Valencia, 1999.

[David01] “Performance issues in PCI reading”, A. David. Available: http://adavid.home.cern.ch/adavid/public/NA60/online/PCIReadingNote/PCIReadingNote.pdf.

[David01-2] PCI-FLIC driver web page: http://adavid.home.cern.ch/adavid/public/NA60/online/FLIC/driver/flic/

[David01-2] “Readout of NA60’s silicon strip planes”, A. David, J. Buytaert, J. Lozano and R. Shahoyan. To be published as NA60 technical note.

[Dorenbosch91] “Data Acquisition Studies for the Superconducting Super Collider”, J. Dorenbosch, E.C. Milner, A.W. Booth, M. Botlo, R. Idate and V. Kapoor, in Proc. First Annual Conference on Electronics for Future Colliders, Chestnut Ridge, New York. May, 1991.

[Drochner01] “A VME Controller for Data Acquisition with flexible Gigabit Data Link to PCI”, M. Drochner, W. Erven, M. Ramm, P. Wüstner and K. Zwoll, in Proc. 12th IEEE NPSS Real Time Conference, 2001, pp. 204-207.

[Dufey01] “Use of Network Processord for Data Multiplexing and Data Merging”, J. P. Dufey, B. Jost and N. Neufeld, in Proc. 12th IEEE-NPSS Real Time Conference, Valencia, Jun. 2001, pp. 195-198.

[Erwen92] “COSY Data Acquisition System for Physical Experiments”, W. Erven, J. Holzer, H. Kopp, H.W. Loevenich, W. Meiling and K. Zwoll. IEEE Trans. Nucl. Sci., 1992, pp 148-158.

[Essel92] “GOOSY-VME: The Data Acquisition and Analysis System at GSI”, H.G. Essel, J. Hoffmann, W. Ott, M. Richter, D. Schall, H. Sohlbach, W. Spreng. IEEE Trans. Nucl. Sci., 1992, pp. 248-251.

[Fastbus83] “FASTBUS a modular high speed data acquisition system for high energy physics and other applications” DOE/ER-0189, by the NIM Committee and ESONE/FB/01 by the ESONE Committee, 1983.

[Fastbus86] “Fastbus Modular High-Speed Data Acquisition and Control System and Fastbus Standard Routines”, IEEE 960-1986 Std, 1986.

[Fraser97] “The Quark Machines: how Europe fought the particle physics war**”**, Gordon Fraser. Institute of Physics Publishing, 1997. ISBN: 0 7503 0447 2.

[Gavillet01] Readout Unit internal review report, Ph. Gavillet, 24th July, 2001. Available: http:/ /lhcb-comp.web.cern.ch/lhcb-comp/DAQ/FEMRU%20Internal%20Review%20report.txt

[Geesaman89] “Data Acquisition for FNAL E655”, D.F. Geesaman, M.C. Green, S. Kaufman, S. Tentindo-Repond, IEEE Trans. Nucl. Sci., vol. 36, no. 5, Oct. 1989.

[Gigi99] “Dual-Port Memory with Reconfigurable Structure”, D. Gigi, G. Antchev, in Proc. Fifith Workshop on Electronics for LHC Experiments, Snowmass, Colorado, Sept. 1999.

[Groom00] Particle Physics Booklet, extracted from *The Review of Particle Physics*, D. E. Groom et al, The European Physics Journal C15 (2000) 1.

[Guirao01] “A networked mezzanine card Linux processor”, A. Guirao, J. Toledo, D. Dominguez, B. Bruder and H. Müller, in Proc. 12th IEEE NPSS Real Time Conference, Valencia, 2001, pp. 81-84.

[Harris98] “LHCb Data Flow Requirements. User requirements document”, F. Harris, M: Frank, LHCb Note 98-027, CERN, 1998.

[Hinkelbein00] “Pattern Recognition Algorithms on FPGAs and CPUs for the ATLAS LVL2 Trigger”, C. Hinkelbein et al., IEEE Trans. Nucl. Sci., vol. 47, no. 2, pp. 362-366, Apr. 2000.

[Hewlett1] 5V receiver and transmitter G-LINK chips data sheet. Available: http:// www.semiconductor.agilent.com/cgi-bin/morpheus/displayFile/ displaySecureFile.jsp?BV_SessionID=@@@@0588020311.1007544224@@@@&BV_EngineID=cadccjhdhedlbemgcgjcfijdin.0&oid=9555

[IA-32] The IA-32 Intel Architecture Software Developer's Manual, Volume 3: System Programming Guide. Available: http://developer.intel.com/design/pentium4/manuals/ 245472.htm.

[IDT70V3599] IDT’s 128Kx36 3.3V, synchronous dual port RAM data sheet. Available: http:// www.idt.com/products/pages/Multi-Ports-70V3599.html.

[Infineon1] Infineon's Paroli web page: http://www.infineon.com/cgi/ecrm.dll/ecrm/scripts/prod_ov.jsp?oid=15437&cat_oid=-8222

[LeVine00] "The STAR DAQ Receiver Board", M. J. Le Vine et al., IEEE Trans. Nucl. Sci., vol. 47, no. 2, pp. 127-131, Apr. 2000.

[Lindsay78] "RMH: A fast and flexible data acquisition system for Multiwire Proportional Chambers and other detectors", J. B. Lindsay et al., presented at the Wire Chamber Conference, Vienna, Feb. 1978.

[LHCb98-006] "The L1 Vertex Trigger algorithm and its performance", H. Dijkstra and T. Ruf. LHCb Technical Note. CERN, 1998.

[LHCb98-017] "The LHCb Level-2 trigger". T. Teubert, I. R. Tomalin, J Holt. LHCb Technical Note. CERN, 1998.

[LHCb98-022] "An All-Software Implementation of the Vertex Trigger", M. Koratzinos, P. Mato. LHCb Technical Note. CERN, 1998.

[LHCb98-028] "DAQ Architecture and Read-Out Protocols", M. Frank et al. LHCb Technical Note 98-028. CERN, 1998.

[LHCb98-029] "DAQ Implementation Studies", J. P. Dufey and I. Mandjavidze, LHCb Technical Note 98-029. CERN, 1998.

[LHCb98-030] "SCI implementation study of LHCb data acquisition". Hans Müller. LHCb Technical Note. CERN, 1998.

[LHCb98-033] "Vertex Trigger implementation using shared memory technology". H. Müller. LHCb Technical Note. CERN, 1998.

[LHCb-98-069] "Vertex detector electronics-L1 electronics prototyping". Y. Ermoline. LHCb technical Note. CERN, 1998.

[LHCb99-031] "LHCb Level-1 Vertex Topology Trigger: Requirements and Interface Specifications". Y. Ermoline, V. Lindestruth, A. Walsch. LHCb Technical Note. CERN, 1999.

[LHCb00-001] "The LHCb Vertex Locator and Level-1 trigger". Hans Dijkstra. LHCb Technical Note. CERN, 2000.

[LHCb01-034] "Level-1 decision Unit. Functional specifications". Beat Jost. LHCb Technical Note. CERN, 2001.

[LHCC95-71] ALICE Technical Proposal: A Large Ion Collider Experiment at LHC, CERN/LHCC/95-71, Dec., 1995.

[LHCC98-4] LHCb Technical Proposal: A Large Hadron Collider Beauty Experiment for Precision Measurements of CP-Violation and Rare Decays. Chapters 12 (Trigger) and 13 (DAQ). ISBN: 92-9083-123-5. Published by CERN, February, 1998.

[Lucent3T55] Lucent Technologies' Orca 3T data sheet. Available: http://www.agere.com/netcom/docs/DS99087.pdf

[Lucent3TP12] Lucent Technologies' Orca 3TP12 data sheet. Available: http://www.agere.com/netcom/docs/DS00222.pdf

[Lucent 3LP26] Lucent Technologies' Orca 3LP26B data sheet. Available: http://www.agere.com/netcom/docs/DS00151.pdf

[Martínez&Toledo] "Desarrollo hardware y software para aplicaciones de adquisición de datos basadas en la tarjeta PCI-FLIC", Final Studies Project of D. Martínez Martínez, Universidad Politécnica de Valencia. To be published.

[McLaren98] "ATLAS Read-Out Link Data Format - Version 1.1", R. Mclaren and O. Boyle. Available: http://www.cern.ch/HSI/atlas/format/

[Mohanty94] "An Integrated Design Environment for Rapid System Prototyping, Performance Modeling and Analysis using VHDL", S. Mohanty, Master of Science Thesis, University of Cincinnati, 1994.

[Morhac95] "PC-CAMAC Based Data Acquisition System for Multiparameter Measurements", M. Morhac, I. Tuerzo and J. Kristiak, IEEE Trans. Nucl. Sci., vol. 42, no. 1, Feb. 1995.

[Mota00] "Digital Implementation of a Tail Cancellation Filter for the Time Projection Chamber of the ALICE Experiment", B. Mota, J. Musa, R. Esteve and A. J. de Parga, in Proc. LEB-2000 Electronics for LHC Experiments, Krakow, Sept., 2000.

[Müller95] "A Millenium Approach to Data Acquisition: SCI and PCI", H. Müller, A. Bogaerts, V. Lindenstruth, in Proc. International Conference on Computing in High-energy Physics: CHEP '95, Rio de Janeiro, Brazil, Sept. 1995, pp. 388-393.

[Müller98] "LHCb Readout Unit Project Proposal", H. Müller, J. Toledo, F. Bal, L. Mcculloch, E. Watson. Available: http://hmuller.home.cern.ch/hmuller/~HMULLER/DOCS/ruproj.pdf

[Müller99] Presentation at the LHCb DAQ Workshop, October 1999. Available: http://hmuller.home.cern.ch/hmuller/~HMULLER/DOCS/daqws.pdf

[Müller00] Minutes of the two-day Readout Unit meeting for the Level-1 VELO meeting at KIP Heidelberg, September 2000. Available: http://hmuller.home.cern.ch/hmuller/RU/Minutes/Heidelberg/Heidelberg.html

[Müller00-2] Presentation at the May 2000 LHCb FE-DAQ Workshop. CERN. Available: http://hmuller.home.cern.ch/hmuller/RU/daqws2000.pdf

[Müller01-1] "A flexible card concept for Data Acquisition: the PCI-FLIC", H. Müller, F. Bal, A. David, D. Domínguez, J. Toledo, in Proc. 12th IEEE NPSS Real Time Conference. Valencia, 2001, pp. 259-253.

[Müller01-2] "Status on PCI readout", H. Müller et al., Feb. 2001. Available: http://hmuller.home.cern.ch/hmuller/FLIC/na60march.pdf

[National1] “LVDS Design Guide”, Second Edition, National Semiconductors. Available: http://www.national.com/appinfo/lvds/0,1798,100,00.html

[NIMstd] “Standard Nuclear Instrument Modules”, TUD-20893, Department of Energy, Physical and Technological Research Division, Office of Health and Environmental Research. Washington D.C. 20545.

[NP4GS3] More information on the IBM Network Processor NP4GS3 can be found in http://www-3.ibm.com/chips/techlib/techlib.nsf/products/IBM_PowerNP_NP4GS3

[Parkman90] “CSI buses and link groups”, Rev 1.0, C.F. Parkman, CERN/CN, 1990.

[Parkman91] “VICbus, Inter-crate Bus for the IEC821 VMEbus”, C.F. Parkman, ISO/IEC 26.11458 Draft Specification Revision 1.1. IEC, Geneva (Switzerland), 1991.

[Parkman94] Based on an article written by Chris parkman and published at the VITA Journal of September 1994. Available: http://ess.web.cern.ch/ESS/VMEbus_at_CERN/VMEbus_General.html.

[Pascual01] “Development of a DSP-based PMC board for real time data processing in HEP experiments”, J. V. Pascual, V. González, E. Sanchis, G. Torralba, J. Martos, in Proc. 12th IEEE NPSS Real Time Conference, 2001, pp. 235-239.

[PCIstd] “PCI Local Bus Specification. Revision 2.2”, PCI Special Interests Group, Dec., 1998.

[PCIX] “PCI-X Specification Revision 1.0a”, PCI Special Interests Group, 2001.

[PICMG97] CompactPCI specification Revision 2.1. PCI Industrial Computers Manufacturers group, 1997. Rogers Communications, 301 Edgewater Place, Suite 220, Wakefield MA 01880.

[PMCstd] “Draft Standard Physical and Environmental Layers for PCI Mezzanine Cards: PMC”, IEEE P1386.1 Draft 2.0, 1995.

[Pointing91] “Instrumentation Buses for High Energy Physics, Past, Present and Future”, P. Pointing, H. Verweij, IEEE Trans. Nucl. Sci., vol 38, no.2, Apr., 1991.

[Praag95] “Overview of the use of the PCI bus in present and future High Energy Physics Data Acquisition systems”, A. van Praag et al., presented at the PCI-week, 27 to 31 March, 1995 in Santa Clara, CA. Tech. Rep. CERN/ECP 95-4, 1995.

[Pxecore00] Home page of Compulab’s PXECORE product. http://www.compulab.co.il/pxecore.htm

[PXI00] PXI Specification Revision 2.0. PXI Systems Alliance, 2000. http://www.pxisa.org/

[RACE-1] University of Mannheim’s RACE-1 Reconfigurable Accelerator Computing Engine. Web page: http://www-li5.ti.uni-mannheim.de/fpga/

[Rüden89] “The ALEPH Data Acquisition System”, Wolfgang von Rüden, IEEE Trans. Nucl. Sci., vol. 36, no. 5, Oct. 1989.

[RUhist] Minutes of the Readout Unit Project Meetings. Available: http://hmuller.home.cern.ch/hmuller/RUminutes.htm

[RUtechnote] “Readout Unit. FPGA version for link multiplexers, DAQ and VELO trigger”, H. Müller, J. Toledo, A. Guirao. LHCb Technical Note LHCb 2001-136, CERN, 2001.

[Schulz01] “Architecture of the L1 Vertex Trigger Processor”, M. W. Schulz. To be published as a LHCb Technical Note.

[SCIstd] “IEEE Standard for Scalable Coherent Interface (SCI)”, IEEE Std 1596-1992, Aug., 1993.

[Simoes01] “A Simple Approach to X-ray Spectrometry with Driftless Gas Proportional Scintillation Counters”, P.C.P.S. Simoes et al., in Proc. 2001 IEEE Nuclear Science Symposium, San Diego, CA., November 2001.

[S-Link] “The S-Link Interface Specification”, O. Boyle, R. McLaren and E. van der Bij, CERN CN, March, 1997. Available: http://hsi.web.cern.ch/HSI/s-link/spec/spec/s-link.pdf

[Sphicas99] “Data Acquisition in High Energy Physics Experiments”, P. Sphicas, presented at the IEEE NSS Symposium. Seattle, Oct. 1999.

[SPSC00-10] NA60 Technical Proposal: “Study of Prompt Dimuon and Charm Production with Proton and Heavy Ion Beams at the CERN SPS”. CERN SPSC 2001-09, March 2000. Available: http://na6i.web.cern.ch/NA6i/proposal/p.ps.gz

[SPSC01-9] NA60 Status Report, CERN SPSS 2001-009, March 2001. Available: http://na6i.web.cern.ch/NA6i/proposal/spsc-2001/statrep-march12.ps.Z

[S586PC] Digital-Logic, AG S586PC credit-card PC technical user’s manual. Available: http://www.digitallogic.ch/english/products/datasheets/smartmodule_detail.asp?id=smartModule586PC

[Tam89] “An introduction to CAMAC and the CAMAC Controller Diagnostic program”, Technical report for the 1989 summerwork team, Tandy Tam, TRIUMF, 1989. Available: http://daq.triumf.ca/online/camac_primer/camac_primer/index.html

[Texas1] Texas Instruments’ Flatlink web page: http://www.ti.com/sc/docs/apps/analog/flatlink_lvds_.html

[Tol98-1] “A common approach to the Front-end Multiplexer and the Readout Unit”, J.Toledo and H. Müller, 1998. Available: http://toledo.home.cern.ch/toledo/~toledo/newarch.PDF

[Tol98-2] “A common approach to the Front-end Multiplexer and the Readout Unit”, J.Toledo and H. Müller, presentation at the LHCb Week, December 1998. http://toledo.home.cern.ch/toledo/~toledo/lhcbw_transp.PDF

[Tol98-3] “Cyclic Redundancy Code (CRC) Implementation on FPGAs”, J. Toledo. Available: http://toledo.home.cern.ch/toledo/~toledo/crc1.pdf

[Tol99-1] “Readout Unit for the LHCb experiment”. J. Toledo, H. Müller, F. Bal, B. Jost, in Proc. Fifth Workshop on Electronics for LHC Experiments, Snowmass, Colorado. Sept. 1999. Available: http://toledo.home.cern.ch/toledo/~toledo/leb99.doc

[Tol01-1] “A Readout Unit for high rate applications”, J. Toledo, F. Bal, D. Domínguez, A. Guirao, H. Müller, in Proc. 12th IEEE NPSS Real Time Conference, 2001, pp. 230-234.

[Tol01-2] “A plug&play approach to data acquisition”, J. Toledo, H. Müller, J. Buytaert, F. Bal, A. David, A. Guirao, F. J. Mora, in Proc. 2001 IEEE Nuclear Science Symposium, San Diego, CA., November 2001.

[Usai01] “The Pixel Readout Board”, G. Usai et al., NA60 Technical Note. To be published.

[VITA32std] “Processor PMC Standard for Processor PCI mezzanine cards”, VITA-32 Draft 0.41 Std., Sept. 2000.

[VMEstd] “The VME bus specification”, ANSI/IEEE 1014-1987, IEC 821.

[Walsch01] “A Hardware/Software triggered DMA engine”, A. Walsch, LHCb 2001-125, CERN, 2001. Available: http://weblib.cern.ch/database

[Walsch01-2] “Evaluation of SCI as a Fabric for a Computer based Pattern Recognition Trigger running at 1.12 MHz”, A. Walsch, V. Lindenstruth. M. W. Schulz, in Proc. 12th IEEE NPSS Real Time Conference, pp. 11-15, Valencia, June 2001.

[Watzlavik92] “FATIMA: A Data Acquisition System for Medium Scale Experiments”, K.H. Watzlavik, R. Nellen, T. Noll, M. Karnadi, H. Machner, IEEE Trans. Nucl. Sci., pp 154-158, 1992.

APPENDIX I Application development

There are two programmable components in the Readout Unit: the FPGAs and the MCU. This appendix describes the development environments for these two elements.

I.I. FPGA development environment

An HDL-based methodology is used for FPGA development. HDL languages (with VHDL and Verilog as major representatives) allow structural and behavioral description of complex digital systems, with none or little[1] technology dependence. These languages are simple in syntax and easy to learn (somewhat close to simplified Pascal or C languages) and are today's foundation of complex digital design. We have chosen VHDL as it is intensively used and well supported at CERN, though the use of Verilog would not make any difference in the tools or methodology.

I.I.I. HDL design flow

We have chosen Innoveda's VisualHDL[2] as design entry and simulation tool. Among different possibilities (top-down, bottom-up and middle-out design approaches are supported by the tool) a top-down methodology has been used. Block diagrams are drawn, with the possibility of multi-level hierarchy. Each box in a block diagram (Figure I-I, top left corner) can be visually described as a graphical state diagram (top right corner) a flowchart for asynchronous state machines or as a table for combinational logic. Raw VHDL code descriptions are, of course, also allowed.

Functional simulations can be carried out, either on individual blocks or on the whole design. When the simulations are satisfactory, VHDL code is automatically generated for the whole design, with optimizations for specific synthesis tools[3]. Visual HDL also accepts the output from

1. The instantiation of technology-specific components like macros and I/O buffers restricts portability. Nevertheless, it can take between a few minutes and one week to port any design to a different FPGA vendor.
2. Now renamed as Visual Elite. See http://www.innoveda.com/products/datasheets_HTML/vishdl.asp

the FPGA vendor place and route tools, in the form of standard delay format (SDF) files, which allow to perform post place-and-route timing simulations and compare the VHDL model with the actual implementation. The VHDL output is translated into a technology-specific netlist by the synthesis tool (Synplify[1] in our current development environment, though Leonardo and Galileo[2] have also been used).

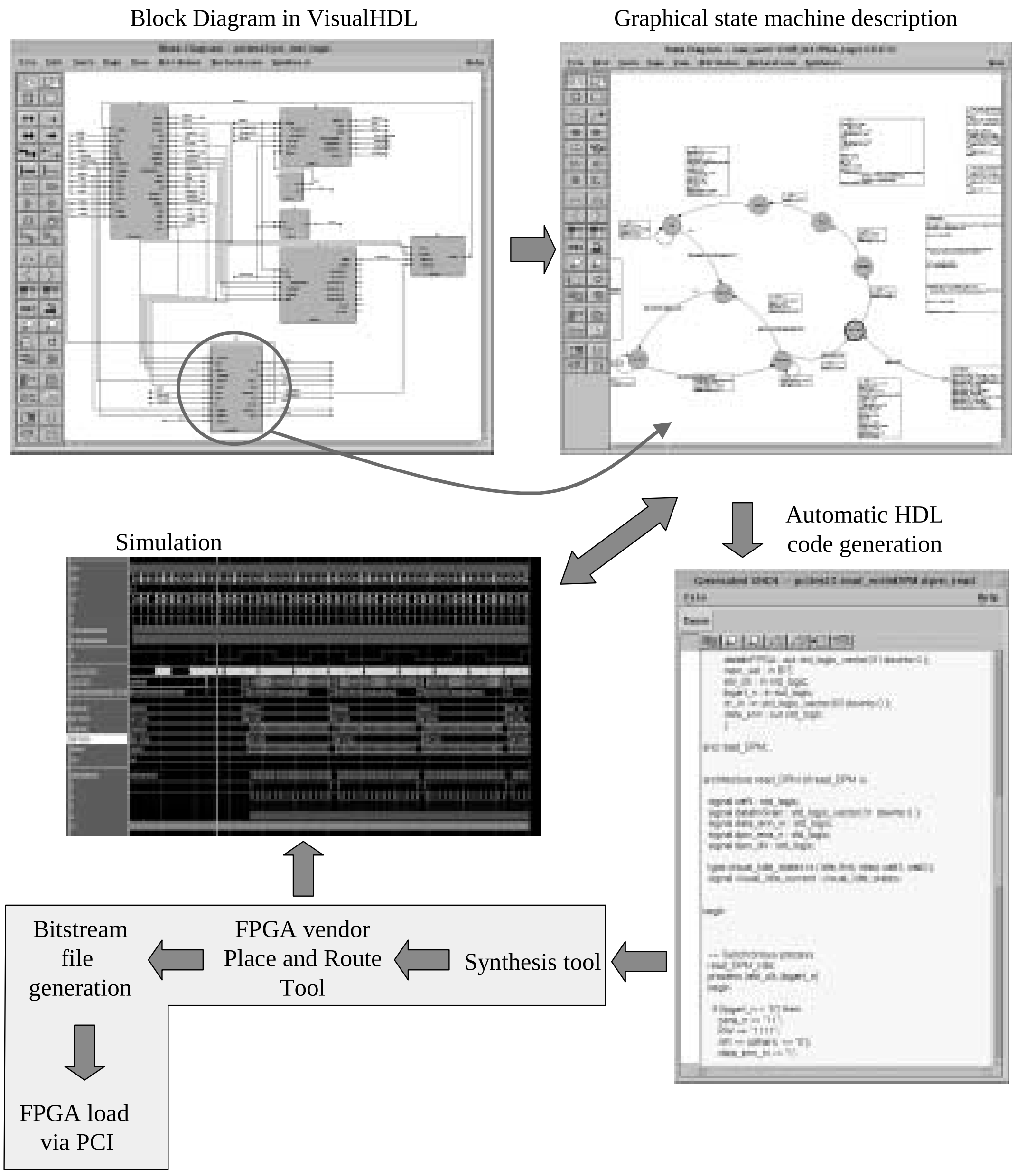

Block Diagram in VisualHDL
Graphical state machine description
Automatic HDL code generation
Simulation
Bitstream file generation
FPGA vendor Place and Route Tool
Synthesis tool
FPGA load via PCI

Fig. I-I. HDL design flow for the RU FPGAs.

3. Not all synthesis tools support all VHDL syntax constructions. In some cases, specific directives via VHDL attributes (like state-machine coding) or constructions ("*case*" instead of "*if*" clauses) produce a better result in a specific synthesis tool.
1. See http://www.synplicity.com/
2. Leonardo an Galileo are from Exemplar. See http://www.exemplar.com/

The EDIF netlist output from the synthesis tool is used as an input to the Orca Foundry 9.x tools[1].

Orca Foundry includes a **mapper** that packs the flip-flops and logic into ORCA LUTs (look-up tables), a **place and route** tool and a **bitstream generator** (bitgen) tool that creates the binary configuration file. Optionally, a graphic editor and floor planer (EPIC) is available (see Figure I-II). An additional ORCA tool, the FPSC Configuration Wizard, is used to crate a PCI core instantiation in VHDL needed for proper device configuration.

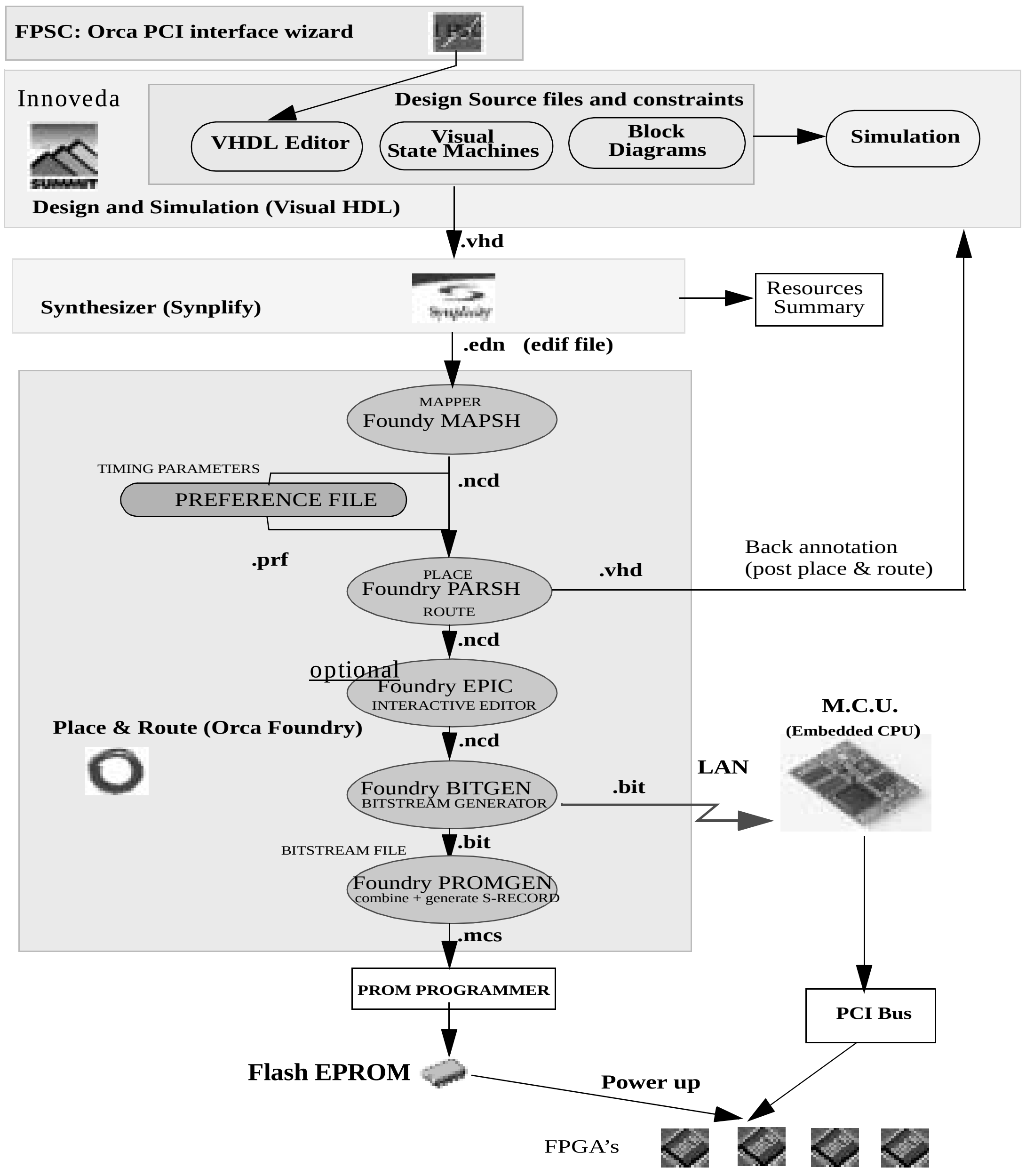


Fig. I-II. Design flow with Lucent Technologies' ORCA FPGAs.

1. See http://www.agere.com/netcom/orca/software.html

Once the bitstream file is generated, two options are available:

- Convert the file to MCS86 or other PROM-compatible format for programming the on-board FPGA-configuration EPROM memories in the RU.
- Load the FPGA configuration bitstream from the MCU via PCI using the *flasher* utility described in the following section. This option takes less than 10 minutes for a complete re-programming cycle (i.e., from the VHDL code generation until the FPGA is loaded via PCI).

I.II. MCU development

A Linux 2.2.x system is used as operating system in the MCU, though DOS and Windows95 have also been tested successfully. Hard disk, keyboard and mouse can be directly connected to the MCU. The available serial port can be used to interface a terminal, completing the development and test set up. Alternatively, a PC with terminal emulation software can be connected to the serial port in the MCU. As a third option, the MCU can be operated remotely via the LAN interface.

I.II.I. Access to PCI devices in Linux

The basic access to a PCI device in Linux is summarized in Figure I-III and relies on the use of the ***pci.h*** library. Using functions from this library, the steps needed to access a device are:

1. Once the bus, device and function numbers for the target device are know, the function ***pci_get_device*** returns a pointer to its PCI configuration space.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <pci/pci.h>
#include <sys/stat.h>
#include <fcntl.h>
#define ORCA_BAR0 0x10

int main(int argc, char *argv[])
{
  int i,fd,dev_num, bus_num;
  u32 bar0_addr,position;
  struct pci_access *pci_acc;
  struct pci_dev *orca_config_space;
  unsigned long *ptr,value,readback;

  /* Scan the PCI bus, and store the config space in pci_acc */
  pci_acc = pci_alloc();
  pci_init(pci_acc);
  pci_scan_bus(pci_acc);

  /* Select fpsc to access */
  if(strcmp(argv[1],"ebi1")==0)
bus_num = 0x1; dev_num = 0xF;  /* EBI1 */
...
...

/* Verify if an orca is there */
orca_config_space=pci_get_dev(pci_acc, bus_num,
dev_num,0);
bar0_addr = pci_read_long(orca_config_space,ORCA_BA
& PCI_BASE_ADDRESS_MEM_MASK;

if(pci_read_long(orca_config_space,0) != 0x540111c1)
 {printf("Can't find orca board\n"); return(-1);}
else { printf("Orca board has been found ...\n");
fd = open("/dev/mem", O_RDWR);
ptr = (unsigned long *) mmap(
NULL, 128, PROT_READ | PROT_WRITE,
MAP_SHARED, fd, bar0_addr);
/*----Read/write memory according to user command----
 if (strcmp(argv[2],"W")==0)
 {
  value= (unsigned long) atol(argv[3]);
  position = (u32) atol(argv[4]);
  *(ptr+position) = value;
  printf("%x written to position ", value);
  printf("[%x]",position);
  readback = *(ptr+position);
  if (readback == value) printf("...[OK]\n");
  else printf("...error: readback %x",readback);
 }
...
...
/* free buffer before leaving  */
 munmap(ptr, 128);
 return 0;
     }
}
```

Fig. I-III. Example of C code to access PCI-mapped memory in the FPGAs.

2. Using this pointer and the appropriate offsets to call the function ***pci_read_long***, the different BAR (Base Address Register) registers can be read. In the case of the EBI FPGAs

in the RU, the SEB is mapped in BAR0 and several control and monitoring registers are mapped in BAR5.

3. The value read from the BAR registers need to be masked to have 0h in the lower 4 four bits.
4. The masked BAR addresses value are used as an input to the ***mmap*** function to get pointers to the different memory regions.
5. These pointers can be used as in the highlighted assignments in Figure I-III to perform CPU-initiated read and write operations. Block transfer functions and any other function that operates with memory pointers can be used.

Before the program exits, ***munmap*** (memory unmap) must be called to free the reserved memory buffer.

I.II.II. The *flasher* utility

Written by NA-60 for the PCI-FLIC card, this utility to load a bitstream into a Lucent Technologies FPGA with embedded PCI ASIC interface is fully functional on the RU-II. It works with the MCU and has also been tested with other single-board computers plugged on the RU-II's auxiliary PCI connector. The flasher utility must be called using the following convention:

flasher device bitstream_file

Where ***device*** is ***bus:device.function*** and ***bitstream_file*** is a ".bit" file created by Orca Foundry tools. In the following usage example "***flasher 1:f.0 hope6.bit***", the FPGA located in PCI bus 1, device Oxf, function 0, is configured with the bitstream file named *hope6.bit*.

Common to all bitstreams are a read/write PCI target interface that grants access to the RU's SEB via the BAR0 and five user-defined registers located in BAR5 at offsets 100h to 500h.

I.II.III. The rwpci utility

This test utility reads and writes memory positions in the SEB. It must be called using the following conventions:

- ***rwpci device R n***: Reads the *n*th 32-bit DPM memory position.
- ***rwpci device W val n***: Writes *val* into the *n*th DPM memory position and reads back the position for verification.
- ***rwpci device I val***: Writes *val* to the first 32 DPM positions.
- ***rwpci device INC val***: Writes the first 32 DPM positions with a self-incrementing counter value starting with *val.*
- ***rwpci device DUMP***: Prints in the screen the first 32 DPM memory positions.
- ***rwpci device RREG pos***: reads the user-defined 32-bit register located in offset BAR5+*pos*
- ***rwpci device WREG val pos***: writes *val* into the user-defined 32-bit register located in offset BAR5+*pos* and verifies with a read operation.
- ***rwpci device TESTWREBI1 burst_cnt go***: Configures the destination address, burst count in quadwords and go/stop bit in the FPGA to initiate a sequence of PCI write bursts to another PCI device. The transactions can be stopped writing a 0 in *go* and enabled writing a 1.

- ***rwpci device TESTWRNIC burst_cnt go bar_nr***: Used to perform write tests to a PCI device plugged on the NIC slot. Configures the destination address -according to the selected BAR *bar_nr* in the NIC- burst count in quadwords and go/stop bit in the selected device. The transactions can be stopped writing a 0 in *go* and enabled writing a 1.

I.II.IV. I2C and JTAG controller utilities

The programming of the clock domains SEM, EBI and PCI is performed via the I2C interface (SCL and SDA lines) and two additional signals (SEL and MODE) in the PMC's I/O connector (see Figure I-IV). A utility has been written to program the clock frequencies using the on-board MCU I2C interface and two GPIO lines.

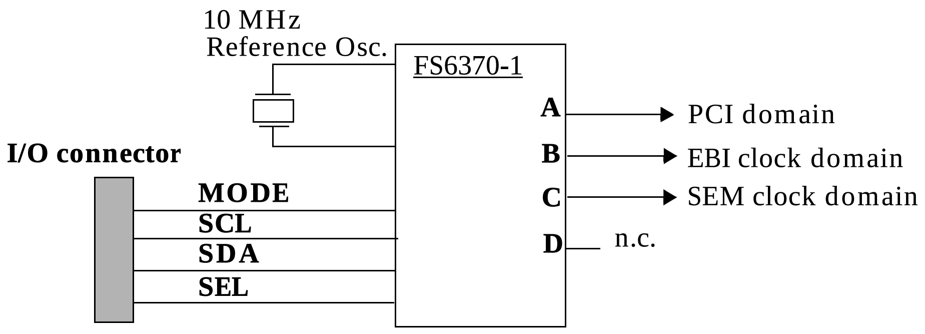


Fig. I-IV. Programming interface in the RU on-board clock generator.

A JTAG controller is emulated using GPIO signals through the PMC I/O connector [Bruder00]. Operating frequencies up to 100 kHz have been tested. This software can be used to test the FPGAs and PLDs in the RU-II.

APPENDIX II

Data Acquisition and trigger systems in NA-60, a small HEP experiment

The work described in the previous chapters has been carried out in the context of a large 21st century experiment at CERN LHC, at the cutting-edge of high-end electronics for HEP DAQ and trigger systems. Nonetheless the picture is incomplete, as the HEP arena is also populated by a large number of small experiments with less demanding requirements and much smaller budget. Both circumstances favour the reuse of already existing components, sub-detectors, electronic modules and software tools from other experiments.

This is the case of a small fixed target experiment at CERN SPS: NA-60 [SPSC00-10]. It has inherited from its predecessors (NA-50, 1994-2000 period, and NA-38, which ran from 1986 to 1992) old DAQ and trigger systems, with some concepts dating from the seventies and a number of modules from the eighties. The old system cannot cope with the new requirements and a new DAQ system has been designed [SPSC01-9], following the PC-based DAQ trend presented in "Trends in DAQ systems for the 21st century" on page 20.

One of the cards used in the RU test station, the PCI-FLIC [Müller01-1], is also a key module in the new NA-60 DAQ system. In this appendix, this new DAQ system and the applications of the PCI-FLIC card are presented, thus completing the picture of the state-of-the-art in DAQ systems for HEP experiments.

II.I. The NA-60 experiment

II.I.I. Detector geometry

The NA-60 detector consists of the following four sub-detectors:

- **Muon spectrometer**: Inherited from NA-38, it is made up of eight multi-wire proportional chambers (MWPCs, numbered 1 to 8 in Figure II-I) and four trigger hodoscopes[1] made up of plastic scintillators (R1 to R4). The muon spectrometer is used to reconstruct particle tracks.

- **Zero Degree Calorimeter** (ZDC): Inherited from NA-50, it collects and amplifies in photo-multiplier tubes (PMTs) the Cherenkov light produced in quartz fibers when a particle crosses the material. The ZDC is also used in the trigger system for heavy-ion runs.
- **Beamscope**: New in NA-60, consists of two stations, each one made up of two silicon microstrip detectors providing two-dimensional information to determine the ion beam's coordinates. This information is used to enhance the vertex calculation.
- **Silicon pixel telescope**: Also new in NA-60, is made up of ten silicon pixel planes, eight pixel chips per plane. It provides information about the interaction vertex and about the generated muons before they scatter in the carbon absorber.

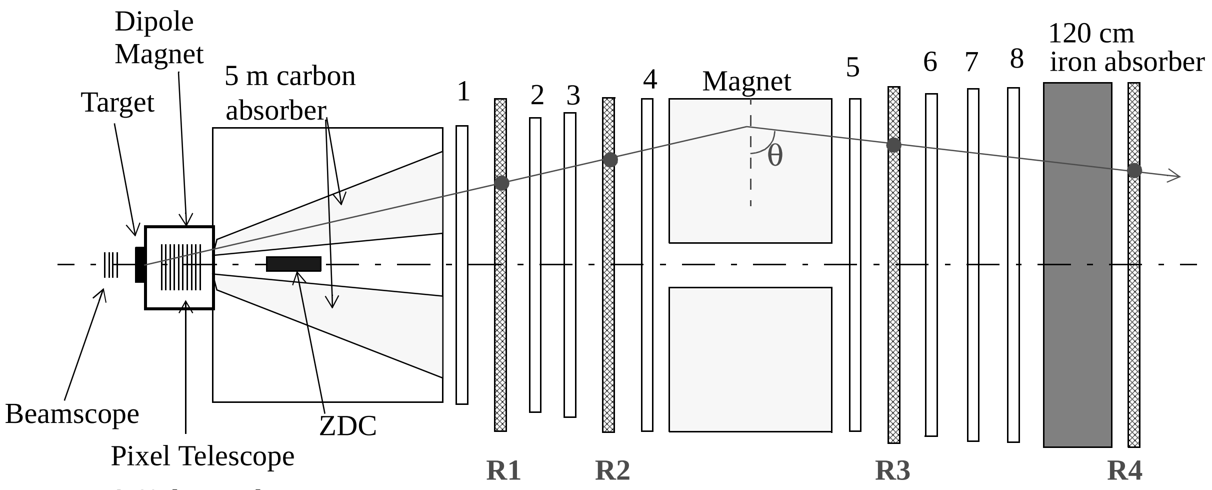


Fig. II-I. NA-60 detector layout.

II.I.II. Trigger system

The NA-60 experiment aims at measuring certain properties of dimuon systems produced in proton and heavy-ion collisions. So, the trigger system is designed for muon detection. The trigger system uses basically the information from the four hodoscope planes. A muon track is drawn in Figure II-I. The first step in the trigger algorithm is the detection of coincidences in planes R1 and R2. Indeed, the scintillator spacing in planes R1 and R2 is such that, if a track is originated at the vertex (interaction point), hits with the same coordinates are produced in R1 and R2. The momentum is determined by checking in a look-up table the value that corresponds to the R4 hit coordinate, thus providing fast momentum discrimination. Note that the track can only correspond to a muon, as by definition, it is a particle able to cross a 120-cm iron absorber. This justifies the existence of the iron wall. The data from R3 are redundant and are used for eliminating accidental coincidences.

The whole trigger system is fully implemented in hardware. Data arrive at the counting room and each channel[1] is conditioned in a pipeline consisting of a discriminator (to filter out noise), a programmable delay (to compensate for cable length tolerances) and a time averaging process (to compensate for differences in hit distance to the photo-multiplier tube).

1. Hodoscope is a term used for a combination of detector elements arranged in space and connected by logic circuitry such that particle tracks can be identified. As in this case, they are normally used in trigger systems and made up of scintillator counters, for this is a very fast kind of detector as required in trigger systems.
1. There are 768 data channels in a trigger hodoscope plane.

Coincidence and momentum checking is performed with logic gates and jumper-based look-up tables for minimum delays. The total time, from data arriving at the discriminators until a trigger decision is produced, is less than 100 ns. There are no clocks and no synchronization means in the system other than the programmable delays. Meanwhile, analog detector data have been delayed with meters of cable for exactly the same amount of time (around 100 ns) in order to arrive to the readout electronics simultaneously with the trigger decision.

Needless to say that this scheme is unconceivable in large experiments like LHCb, with one million data channels and a 40-MHz interaction rate.

II.I.III. DAQ system architecture

The DAQ system in NA-60 reads out four sub-detectors (muon spectrometer, beamscope, ZDC and pixel telescope) via PCI instead of NA-50's VME-based system. According to the architecture depicted in Figure II-II, event building is carried out in a Global Data Concentrator PC (**GDC**) using an Ethernet switch for data routing from the Local Data Concentrator PCs (**LDC**) to the GDC. Two LDCs (Pixel 1 and Pixel 2 in the figure) read out the Pixel Telescope sub-detector. One LDC reads out the Beamscope and ZDC sub-detectors. The Muon spectrometer requires one LDC. Thus, five PCs are enough to read out the different sub-detectors, resulting in a significant cost saving if compared to a VME-based solution. As described in the next section, two PCI-FLIC cards per LDC (with their corresponding mezzanine cards) are used for sub-detector readout.

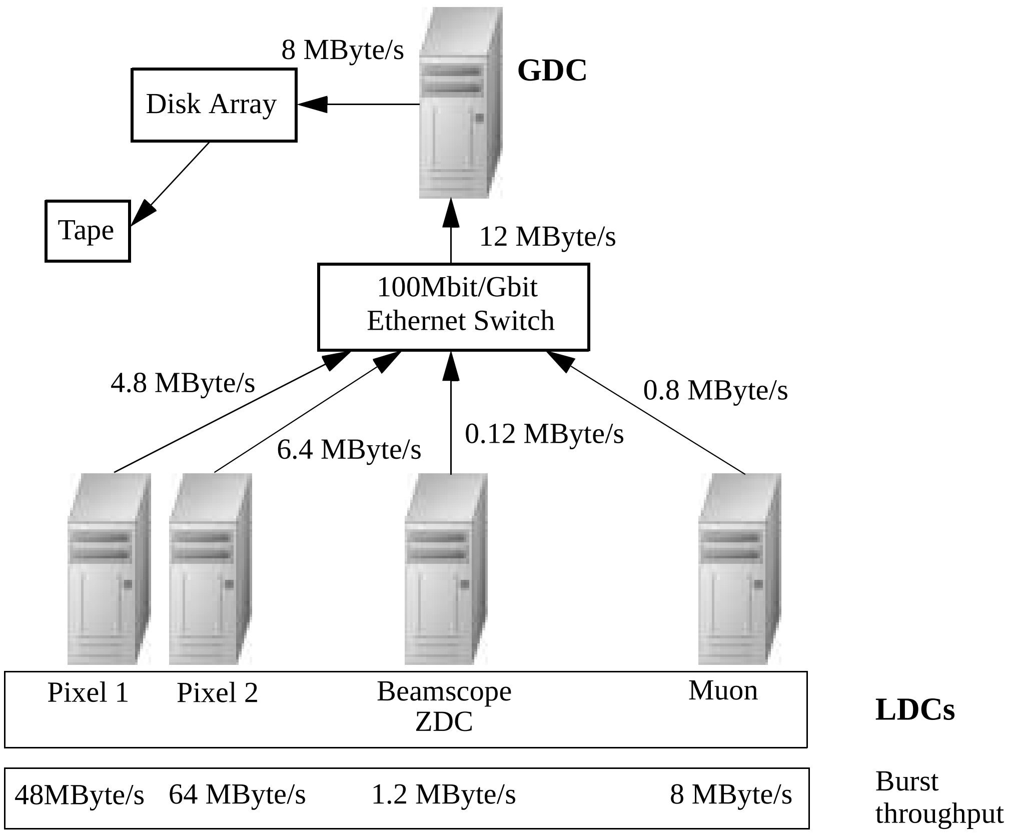


Fig. II-II. NA-60 DAQ system block diagram.

Events are not read out from the front-end electronics on a trigger-by-trigger basis, but in 5-second bursts of up to 8000 triggers each. Data bandwidth generated by an LDC ranges from 0.12 to 6.4 MByte/s, totalling 12 MByte/s though the switch, though ten times higher bandwidths are produced during the bursts.

II.II. Application of the PCI-FLIC to NA-60 detector readout

The readout of NA60 muon spectrometer's multiwire chambers (MWCs) relied on an old CERN system [Lindsey87] consisting of: (1) 32-channel readout CAMAC modules grouped in up to 22 modules per crate, (2) a RMH system encoder collecting hit data via a CAMAC branch cable and outputting to a custom ECL cable according to a protocol named RMH, (3) a VME memory module into which events are written by the RMH encoders, and (4) a VME interface to a readout computer. The 1.6 kHz trigger rate and event sizes in the order of 1 KByte require 1.6 MByte/s throughput. To enhance the event efficiency during 10 s of machine spill, the bandwidth had to be increased to 6-8 MByte/s, which required faster RMH handshake and a four times larger buffer.

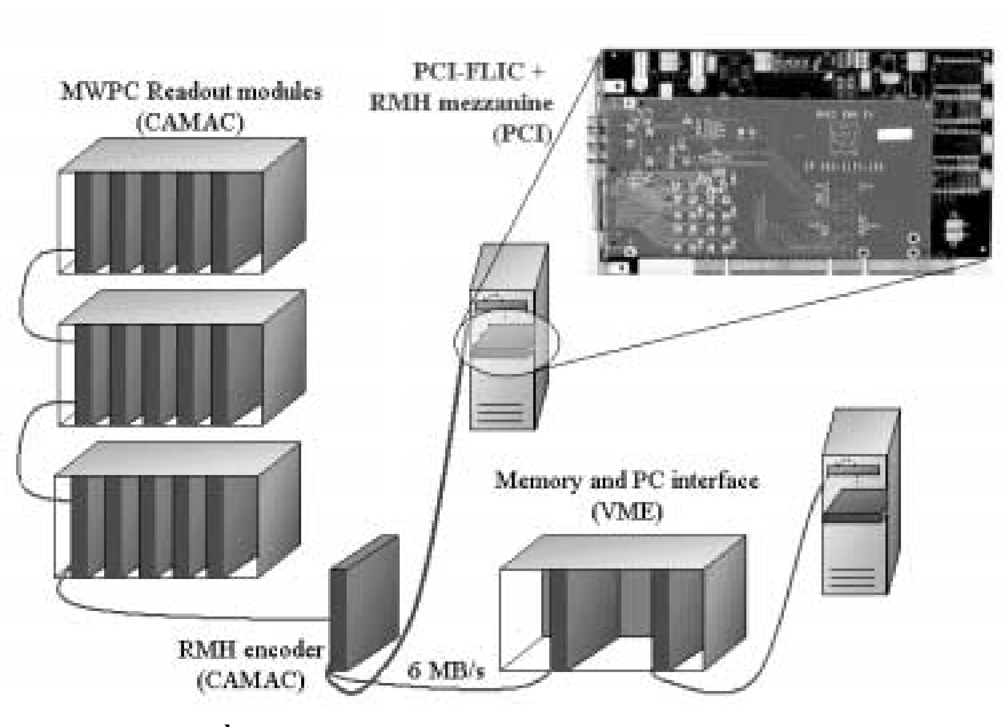


Fig. II-III. Muon spectrometer readout.

A mezzanine card has been designed [Müller01-2] to adapt the RMH cable and ECL electrical levels to the PCI-FLIC and thus replace the VME memory and PC interface modules with a faster RMH receiver with larger buffer (a PCI-FLIC card, see Figure II-III). The buffer is interfaced via PCI to the readout computer (the host PC). Several PCI-FLIC cards housed in a host PC allow to read the detector partitions in parallel from the host. In the RMH mezzanine card (Figure II-IV), NIM connectors are used for the RMH trigger interface (top right corner) and the RMH cable is connected via a 50-pin connector (bottom right). Signals are converted from ECL to TTL levels before being transmitted to the PCI-FLIC via a 64-pin connector (left).

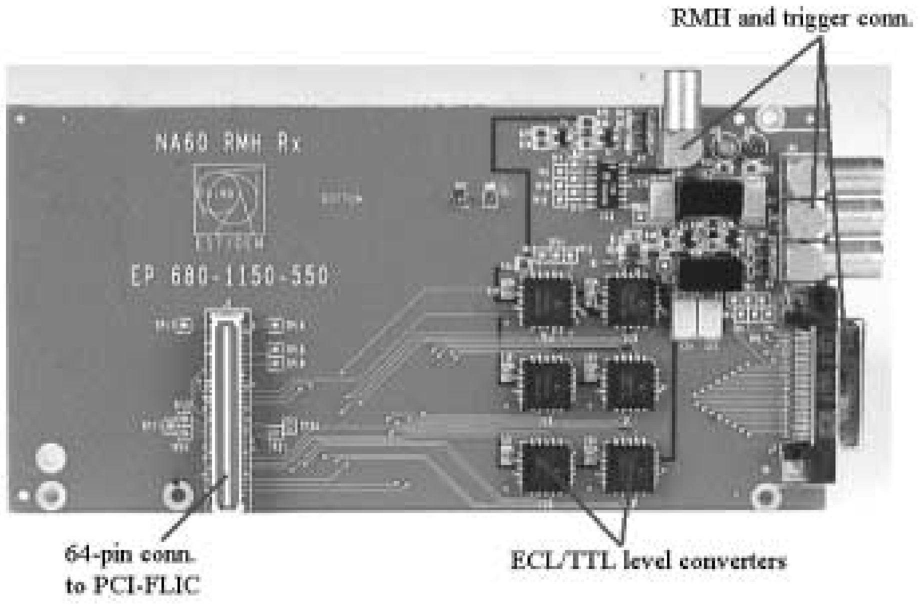


Fig. II-IV. RMH mezzanine card.

II.II.I. Other sub-detectors

A mezzanine card (PRB, Pixel Readout Board) is being designed to interface the front-end electronics of the Silicon Pixel Telescope to the PCI-FLIC [Usai01]. The PRB mezzanine will provide clock and control signal to the pixel chips and will perform zero-suppression and encoding on the read out data. Finally, processed data will be stored in FIFOs that will be read out by the FPGA in the PCI-FLIC card. There will be four PRB mezzanines reading out sixteen chips each.

The use of the PCI-FLIC is also planned for reading the ZDC and the beamscope sub-detectors.